# NEC
## ELECTRON DEVICE

# MOS INTEGRATED CIRCUIT
# μPD75518(A)

# 4 BIT SINGLE-CHIP MICROCOMPUTER

The μPD75518(A) is a 75X series four-bit single-chip microcomputer which enables data processing equivalent to that performed by an eight-bit microcomputer. It is a high-performance product, whose minimum instruction execution time is 0.67 μs, shorter than 0.95 μs for the conventional μPD75516. The ROM and RAM capacities are also larger, and the throughput of the 75X series is further increased. The μPD75517(A) is suited to controllers of electric parts of automobiles.

## FEATURES
- Higher reliable than the μPD75518
- Capacities of program memory, ROM:  32640 × 8 bits
- Capacity of data memory, RAM:  1024 × 4 bits
- Function for specifying the instruction execution time (useful for high-speed operation and saving power)
  - 0.67 μs/1.33 μs/2.67 μs/10.7 μs (when the main system clock operates at 6.0 MHz)
  - 0.95 μs/1.91 μs/3.82 μs/15.3 μs (when the main system clock operates at 4.19 MHz)
  - 122 μs (when the subsystem clock operates at 32.768 kHz)
- Built-in A/D converter operable on low voltage
  - 8-bit resolution × 8 channels (Successive approximation system)
  - $V_{DD}$ = 2.7 to 6.0 V
- Many I/O lines:  64
- Enhanced timer function:  4 channels
- Built-in 8-bit serial interface:  Two channels
  - Built-in NEC serial bus interface (SBI)
- Clock operable with ultra-low power consumption (when 5-μA TYP. operates on 3 V.)
- Product with a built-in PROM available:  μPD75P518

## APPLICATIONS
Controller of electric parts of automobiles

## ORDERING INFORMATION

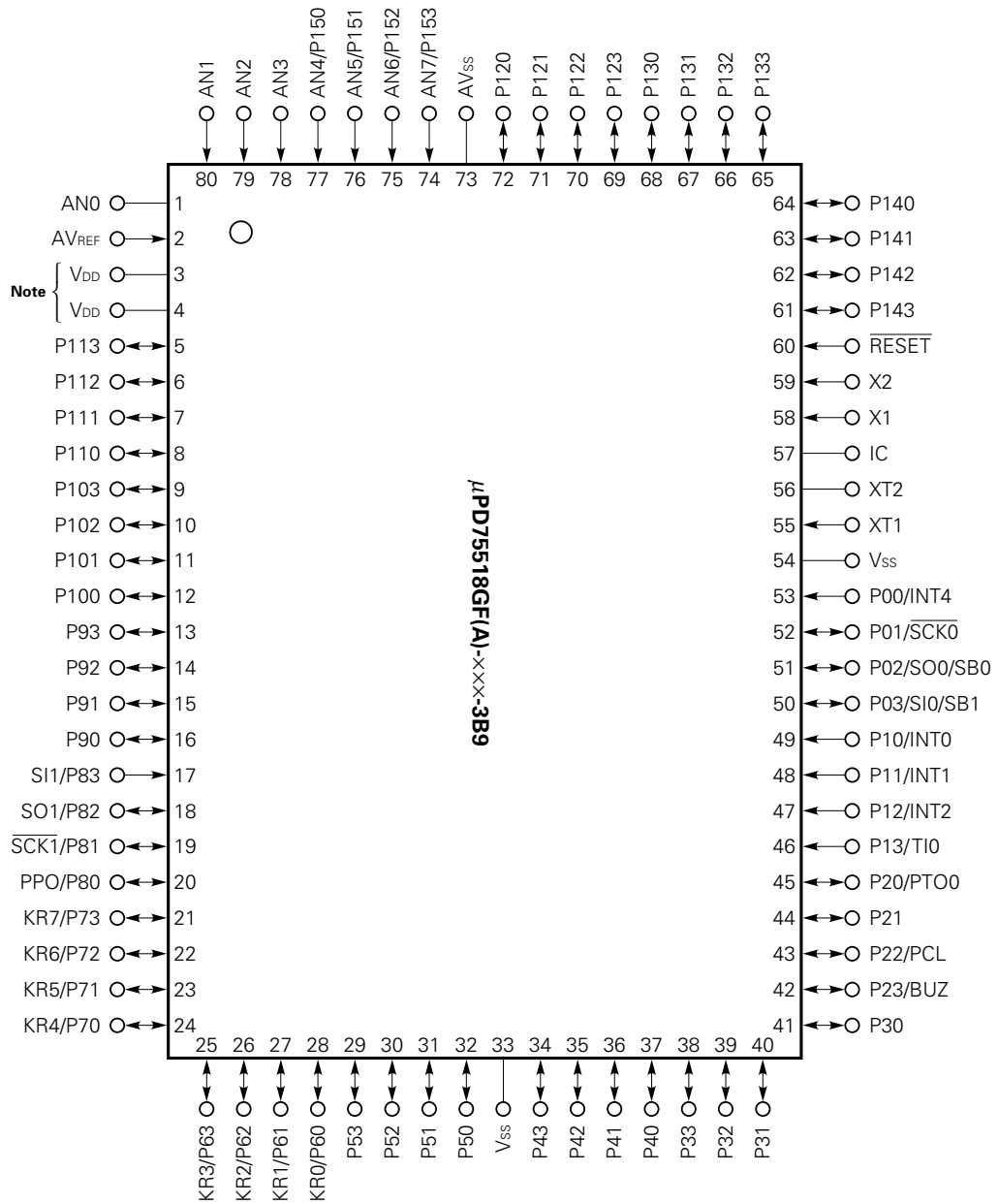| Part number | Package | Quality grade |
| --- | --- | --- |
| μPD75518GF(A)-×××-3B9 | 80-pin plastic QFP (14 mm × 20 mm) | Special |

**Remark**  ×××:  Code number

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.

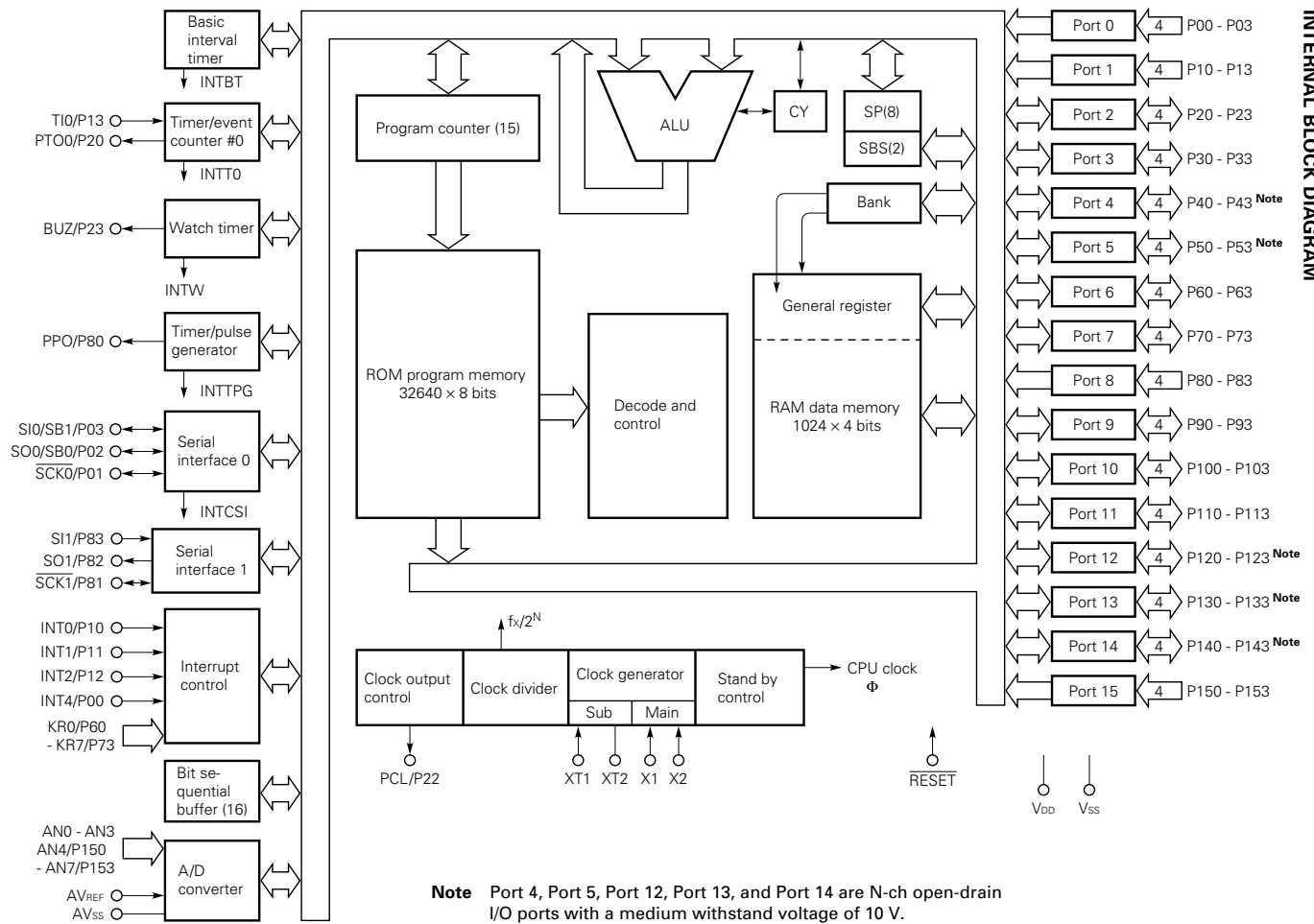**The information in this document is subject to change without notice.**

## FUNCTIONS

| Item | | Functions |
|---|---|---|
| Built-in memory | ROM | 32640 × 8 bits |
| | RAM | 1024 × 4 bits |
| General registers | | (4-bit × 8 or 8-bit × 4) × 4 banks |
| 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) |
| I/O ports | Total | 64 |
| | Number of CMOS input lines | 16 (Shared with INT, SIO, PPO, and analog input. Seven lines can be pulled up by software.) |
| | Number of CMOS I/O lines | 28 (Four lines for LED driving)<br>• 16 lines can be pulled up by software.<br>• Four lines can be pulled down by the mask option. |
| | Number of N-ch open-drain I/O lines | 20 (Eight lines for LED driving. Withstand voltage is 10 V. 20 lines can be pulled up by the mask option.) |
| A/D converter | | 8-bit resolution × 8 channels (Successive approximation system)<br>• Capable of low-voltage operation: $V_{DD}$ = 2.7 to 6.0 V |
| Timer/counter | | Four channels   • Timer/event counter<br>• Basic interval timer<br>• Timer/pulse generator (14-bit PWM output enabled)<br>• Clock timer |
| Serial interface | | Two channels   • NEC standard serial bus interface (SBI)/<br>   three-wire SIO: One channel<br>• General clock synchronous serial interface<br>   (three-wire SIO): One channel |
| Interrupt | | • Vectored interrupt: Seven sources (External: 3, internal: 4)<br>• Test input        : Two sources   (External: 1, internal: 1)<br>• Clock test flag is provided.<br>• Parallel edge detection flag for key scan input is provided. |
| Instruction set | | • Set/reset/test/Boolean operation for bit data<br>• 4-bit data transfer, arithmetic/logical, increment/decrement, and comparison instructions<br>• 8-bit data transfer, arithmetic/logical, increment/decrement, and comparison instructions |
| System clock generator | | • Ceramic/crystal oscillator for main system clock : 6.0 MHz, 4.19 MHz<br>• Crystal oscillator for subsystem clock      : 32.768 kHz |
| Operating supply voltage | | $V_{DD}$ = 2.7 to 6.0 V |
| Package | | 80-pin plastic QFP (14 × 20 mm) |

**PIN CONFIGURATION (TOP VIEW)**

Top pins (left to right): AN1 (80), AN2 (79), AN3 (78), AN4/P150 (77), AN5/P151 (76), AN6/P152 (75), AN7/P153 (74), AV$_{SS}$ (73), P120 (72), P121 (71), P122 (70), P123 (69), P130 (68), P131 (67), P132 (66), P133 (65)

Left side pins (top to bottom):
- AN0 — 1
- AV$_{REF}$ — 2
- V$_{DD}$ — 3 (Note)
- V$_{DD}$ — 4 (Note)
- P113 — 5
- P112 — 6
- P111 — 7
- P110 — 8
- P103 — 9
- P102 — 10
- P101 — 11
- P100 — 12
- P93 — 13
- P92 — 14
- P91 — 15
- P90 — 16
- SI1/P83 — 17
- SO1/P82 — 18
- $\overline{SCK1}$/P81 — 19
- PPO/P80 — 20
- KR7/P73 — 21
- KR6/P72 — 22
- KR5/P71 — 23
- KR4/P70 — 24

Center: $\mu$**PD75518GF(A)-×××-3B9**

Right side pins (top to bottom):
- 64 — P140
- 63 — P141
- 62 — P142
- 61 — P143
- 60 — $\overline{RESET}$
- 59 — X2
- 58 — X1
- 57 — IC
- 56 — XT2
- 55 — XT1
- 54 — V$_{SS}$
- 53 — P00/INT4
- 52 — P01/$\overline{SCK0}$
- 51 — P02/SO0/SB0
- 50 — P03/SI0/SB1
- 49 — P10/INT0
- 48 — P11/INT1
- 47 — P12/INT2
- 46 — P13/TI0
- 45 — P20/PTO0
- 44 — P21
- 43 — P22/PCL
- 42 — P23/BUZ
- 41 — P30

Bottom pins (left to right): KR3/P63 (25), KR2/P62 (26), KR1/P61 (27), KR0/P60 (28), P53 (29), P52 (30), P51 (31), P50 (32), V$_{SS}$ (33), P43 (34), P42 (35), P41 (36), P40 (37), P33 (38), P32 (39), P31 (40)

IC: Internally connected.  Connect the IC pin to V$_{SS}$.

**Note** Be sure to supply power to both the V$_{DD}$ pins.

Basic interval timer — INTBT

TI0/P13, PTO0/P20 — Timer/event counter #0 — INTT0

BUZ/P23 — Watch timer — INTW

PPO/P80 — Timer/pulse generator — INTTPG

SI0/SB1/P03, SO0/SB0/P02, $\overline{SCK0}$/P01 — Serial interface 0 — INTCSI

SI1/P83, SO1/P82, $\overline{SCK1}$/P81 — Serial interface 1

INT0/P10, INT1/P11, INT2/P12, INT4/P00, KR0/P60 - KR7/P73 — Interrupt control

Bit sequential buffer (16)

AN0 - AN3, AN4/P150 - AN7/P153, AV$_{REF}$, AV$_{SS}$ — A/D converter

Program counter (15)

ALU

CY

SP(8)

SBS(2)

Bank

General register

ROM program memory 32640 × 8 bits

Decode and control

RAM data memory 1024 × 4 bits

$f_X/2^N$

Clock output control | Clock divider | Clock generator (Sub, Main) | Stand by control

CPU clock Φ

PCL/P22

XT1 XT2 X1 X2

$\overline{RESET}$

V$_{DD}$ V$_{SS}$

Port 0 — 4 — P00 - P03
Port 1 — 4 — P10 - P13
Port 2 — 4 — P20 - P23
Port 3 — 4 — P30 - P33
Port 4 — 4 — P40 - P43 **Note**
Port 5 — 4 — P50 - P53 **Note**
Port 6 — 4 — P60 - P63
Port 7 — 4 — P70 - P73
Port 8 — 4 — P80 - P83
Port 9 — 4 — P90 - P93
Port 10 — 4 — P100 - P103
Port 11 — 4 — P110 - P113
Port 12 — 4 — P120 - P123 **Note**
Port 13 — 4 — P130 - P133 **Note**
Port 14 — 4 — P140 - P143 **Note**
Port 15 — 4 — P150 - P153

**Note** Port 4, Port 5, Port 12, Port 13, and Port 14 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).<br>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 | | | | Ⓜ - C |
| P10 | I | INT0 | With noise elimination function<br>4-bit input port (Port 1).<br>Pull-up resistors can be provided by software in units of 4 bits. | × | Input | Ⓑ - C |
| P11 | | INT1 | | | | |
| P12 | | INT2 | | | | |
| P13 | | TI0 | | | | |
| P20 | I/O | PTO0 | 4-bit I/O port (Port 2).<br>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).<br>Input/output can be specified bit by bit.<br>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).<br>A pull-up resistor can be provided bit by bit (mask option).<br>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).<br>A pull-up resistor can be provided bit by bit (mask option).<br>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 | KR0 | Programmable 4-bit I/O port (Port 6).<br>Input/output can be specified bit by bit.<br>Pull-up resistors can be provided by software in units of 4 bits. | ○ | Input | Ⓕ - C |
| P61 | | KR1 | | | | |
| P62 | | KR2 | | | | |
| P63 | | KR3 | | | | |
| P70 | I/O | KR4 | 4-bit I/O port (Port 7).<br>Pull-up resistors can be provided by software in units of 4 bits. | | Input | Ⓕ - A |
| P71 | | KR5 | | | | |
| P72 | | KR6 | | | | |
| P73 | | KR7 | | | | |

**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 1] circuit type |
|---|---|---|---|---|---|---|
| P80 | I | PPO | 4-bit input port (Port 8). | × | Input | E |
| P81 | | $\overline{SCK1}$ | | | | Ⓕ |
| P82 | | SO1 | | | | E |
| P83 | | SI1 | | | | Ⓑ |
| P90-P93 | I/O | – | 4-bit I/O port (Port 9).<br>A pull-down resistor can be provided bit by bit (mask option). | × | Low level (when a pull-down resistor is provided) or high impedance | V |
| P100-P103 | I/O | – | 4-bit I/O port (Port 10) | × | Input | E |
| P110-P113 | I/O | – | 4-bit I/O port (Port 11) | | Input | E |
| P120-P123 | I/O | – | N-ch open-drain, 4-bit I/O port (Port 12).<br>Pull-up resistors can be provided bit by bit (mask option).<br>Withstand voltage is 10 V in open-drain mode. | × | High level (when a pull-up resistor is provided) or high impedance | M |
| P130-P133 | I/O | – | N-ch open-drain, 4-bit I/O port (Port 13).<br>Pull-up resistors can be provided bit by bit (mask option).<br>Withstand voltage is 10 V in open-drain mode. | × | High level (when a pull-up resistor is provided) or high impedance | M |
| P140-P143 | I/O | – | N-ch open-drain, 4-bit I/O port (Port 14).<br>Pull-up resistors can be provided bit by bit (mask option).<br>Withstand voltage is 10 V in open-drain mode. | × | High level (when a pull-up resistor is provided) or high impedance | M |
| P150-P153 | I | AN4-AN7 | 4-bit input port (Port 15) | × | Input | Y - A |

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

### 1.2 NON-PORT PINS

| Pin name | I/O | Also used as | Function | | When reset | I/O[1] circuit type |
|---|---|---|---|---|---|---|
| TI0 | I | P13 | External event pulse input pin for the timer/event counter | | – | Ⓑ- C |
| PTO0 | O | P20 | Timer/event counter output pin | | Input | E - B |
| PCL | O | P22 | Clock output pin | | Input | E - B |
| BUZ | O | P23 | Fixed frequency output pin (for buzzer or system clock trimming) | | Input | E - B |
| $\overline{\text{SCK0}}$ | I/O | P01 | Serial clock I/O pin | | Input | Ⓕ- A |
| SO0/SB0 | I/O | P02 | Serial data output pin or serial bus I/O pin | | Input | Ⓕ- B |
| SI0/SB1 | I/O | P03 | Serial data input pin or serial bus I/O pin | | Input | Ⓜ- C |
| INT4 | I | P00 | Edge detection vectored interrupt input pin  (Either a rising or falling edge is detected.) | | – | Ⓑ |
| INT0 | I | P10 | Edge detection vectored interrupt input pin (The edge to be detected is selectable.) | Synchronous | – | Ⓑ- C |
| INT1 | | P11 | | Asynchronous | | |
| INT2 | I | P12 | Edge detection testable input  pin (An rising edge is detected.) | Asynchronous | – | Ⓑ- C |
| KR0-KR3 | I | P60-P63 | Parallel-falling-edge-sensitive testable input pins | | Input | Ⓕ- C |
| KR4-KR7 | I | P70-P73 | Parallel-falling-edge-sensitive testable input pins | | Input | Ⓕ- A |
| $\overline{\text{SCK1}}$ | I/O | P81 | Serial clock I/O pin | | Input | Ⓕ |
| SO1 | O | P82 | Serial data output pin | | Input | E |
| SI1 | I | P83 | Serial data input pin | | Input | Ⓑ |
| AN0-AN3 | I | – | Analog input pins to A/D converter | | – | Y |
| AN4-AN7 | | P150-P153 | | | | Y - A |
| AV$_{REF}$ | I | – | A/D converter reference voltage input pin | | – | Z |
| AV$_{SS}$ | – | – | A/D converter reference GND pin | | – | – |
| X1, X2 | I | – | Pin for 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 | – | Pin for 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 | – | | | | | |
| $\overline{\text{RESET}}$ | I | – | System reset input pin | | – | Ⓑ |
| PPO | O | P80 | Timer/pulse generator pulse output pin | | Input | E |
| V$_{DD}$ | – | – | Positive power supply pin | | – | – |
| V$_{SS}$ | – | – | Ground pin | | – | – |
| IC | – | – | Internally connected[2] | | – | – |

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

      **2.** Be sure to input V$_{SS}$ level to this pin.

## 1.3 PIN INPUT/OUTPUT CIRCUITS

Fig. 1-1 shows the input/output circuit of each μPD75518(A) pin in a simplified manner.

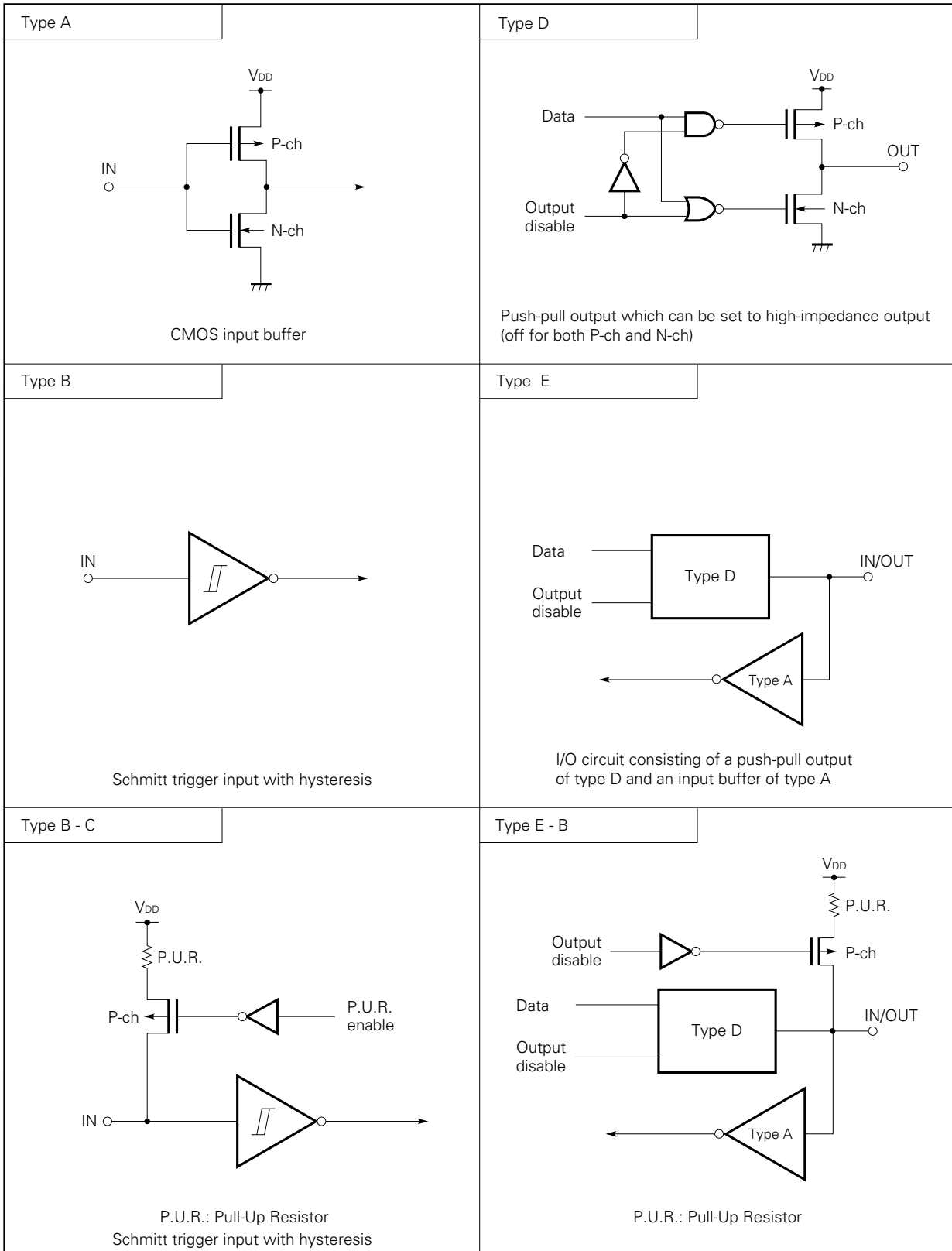**Fig. 1-1  Pin Input/Output Circuits (1/3)**



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

**Fig. 1-1  Pin Input/Output Circuits (2/3)**



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 M

Middle-voltage input buffer (Can sustain + 10 V)

P.U.R.: Pull-Up Resistor

Fig. 1-1  Pin Input/Output Circuits (3/3)

### 1.4 CONNECTION OF UNUSED PINS

**Table 1-1 Recommended Connection of Unused 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 $V_{DD}$ |
| P21 | Output state : To be left open |
| P22/PCL | |
| P23/BUZ | |
| P30-P33 | |
| P40-P43 | |
| P50-P53 | |
| P60/KR0-P63/KR3 | |
| P70/KR4-P73/KR7 | |
| P80/PPO | To be connected to $V_{SS}$ or $V_{DD}$ |
| P81/$\overline{SCK1}$ | |
| P82/SO1 | |
| P83/SI1 | |
| P90-P93 | Input state : To be connected to $V_{SS}$ or $V_{DD}$ |
| P100-P103 | Output state : To be left open |
| P110-P113 | |
| P120-P123 | |
| P130-P133 | |
| P140-P143 | |
| P150/AN4-P153/AN7 | To be connected to $V_{SS}$ |
| AN0-AN3 | |
| XT1 | To be connected to $V_{SS}$ or $V_{DD}$ |
| XT2 | To be left open |
| $AV_{REF}$ | To be connected to $V_{SS}$ |
| $AV_{SS}$ | |
| IC | |

**1.5  SELECTION OF A MASK OPTION**

The following mask options are provided for pins.

**(1)  Specification of built-in pull-up and pull-down resistors**

**Table 1-2   Selection of Pull-Up and Pull-Down Resistors**

| Pin name | Mask option | |
|---|---|---|
| P40-P43, P50-P53, P120-P123, P130-P133, P140-P143 | ① Pull-up resistors provided (Can be specified bit by bit.) | ② No pull-up resistor provided (Can be specified bit by bit.) |
| P90-P93 | ① Pull-down resistors provided (Can be specified bit by bit.) | ② No pull-down resistor provided (Can be specified bit by bit.) |

**(2)  Specification of built-in feed-back resistors for subsystem clock oscillation**

**Table 1-3   Selection of Feed-Back Resistors**

| Pin name | Mask option | |
|---|---|---|
| XT1, XT2 | ① Feed-back resistors provided (when a subsystem clock is used) | ② No feed-back resistors provided (when no subsystem clock is used) |

**Caution   Even if built-in feed-back resistors are provided when no subsystem clock is used, operation is not affected except increased power supply current I$_{DD}$.**

## 2. ARCHITECTURE AND MEMORY MAP OF THE μPD75518(A)

The μPD75518(A) 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 μPD75518(A) contains a static RAM (1024 words × 4 bits) at addresses 000H to 3FFH 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 μPD75518(A) 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 2-2 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.
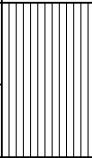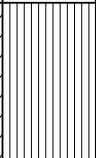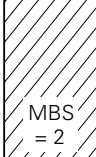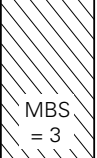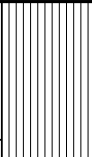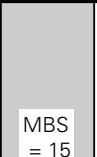
|  | Applicable program processing |
|---|---|
| MBE = 0 mode | • Interrupt processing<br>• Processing that repeats internal hardware and static RAM operations<br>• Subroutine processing |
| MBE = 1 mode | • Usual program processing |

The MBE and MBS are set as indicated below.

**Example**    SET1 MBE        ; MBE ← 1
                CLR1 MBE        ; MBE ← 0
                SEL  MB0         ; MBS ← 0
                SEL  MB1         ; MBS ← 1
                SEL  MB15      ; MBS ← 15

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

| Addressing mode | | mem mem.bit | | @HL @H + mem.bit | | @DE @DL | Stack address-ing | fmem.bit | pmem. @L |
|---|---|---|---|---|---|---|---|---|---|
| Memory bank enable flag | | MBE = 0 | MBE = 1 | MBE = 0 | MBE = 1 | – | – | – | – |
| 000H General resister area 01FH 020H 07FH Data area Static RAM (memory bank 0) 0FFH | | | MBS = 0 | | MBS = 0 | | SBS = 0 | | |
| 100H Data area Static RAM (memory bank 1) 1FFH | | | MBS = 1 | | MBS = 1 | | SBS = 1 | | |
| 200H Stack area Data area Static RAM (memory bank 2) 2FFH | | | MBS = 2 | | MBS = 2 | | SBS = 2 | | |
| 300H Data area Static RAM (memory bank 3) 3FFH | | | MBS = 3 | | MBS = 3 | | SBS = 3 | | |
| Not contained | | | | | | | | | |
| F80H FB0H FBFH Peripheral hardware area (memory bank 15) FC0H FF0H FFFH | | | MBS = 15 | | MBS = 15 | | | | |

**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<br>@HL+<br>@HL– | Address specified by MB and HL.<br>In this case, MB = MBE•MBS |
| | @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 (with the L register holding an even number).<br>In this case, MB = MBE•MBS |
| 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 μPD75518(A) 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.

**Examples 1.** The 8-bit data of port 4 and port 5 are transferred to addresses 20H and 21H.

```
CLR1    MBE          ; MBE ← 0
IN      XA, PORT4    ; XA ← Ports 5, 4
MOV     20H, XA      ; (21H, 20H) ← XA
```

**2.** When P02 is 0, P33 is set.

```
SKT     PORT0.2      ; Skip if bit 2 of port 0 is 1
SET1    PORT3.3      ; Set bit 3 of port 3
```

**3.** A different value is output to port 6, depending on the status of P10.

```
SKF     PORT1.0      ; Skip if bit 0 of port 1 is 0
MOV     A, #1010B    ; A ← 1010B (string effect)
MOV     A, #0101B    ; A ← 0101B (string effect)
SEL     MB15         ; or CLR1 MBE
OUT     PORT6, A     ; Port 6 ← A
```

**Fig. 2-2   Updating Static RAM Addresses**
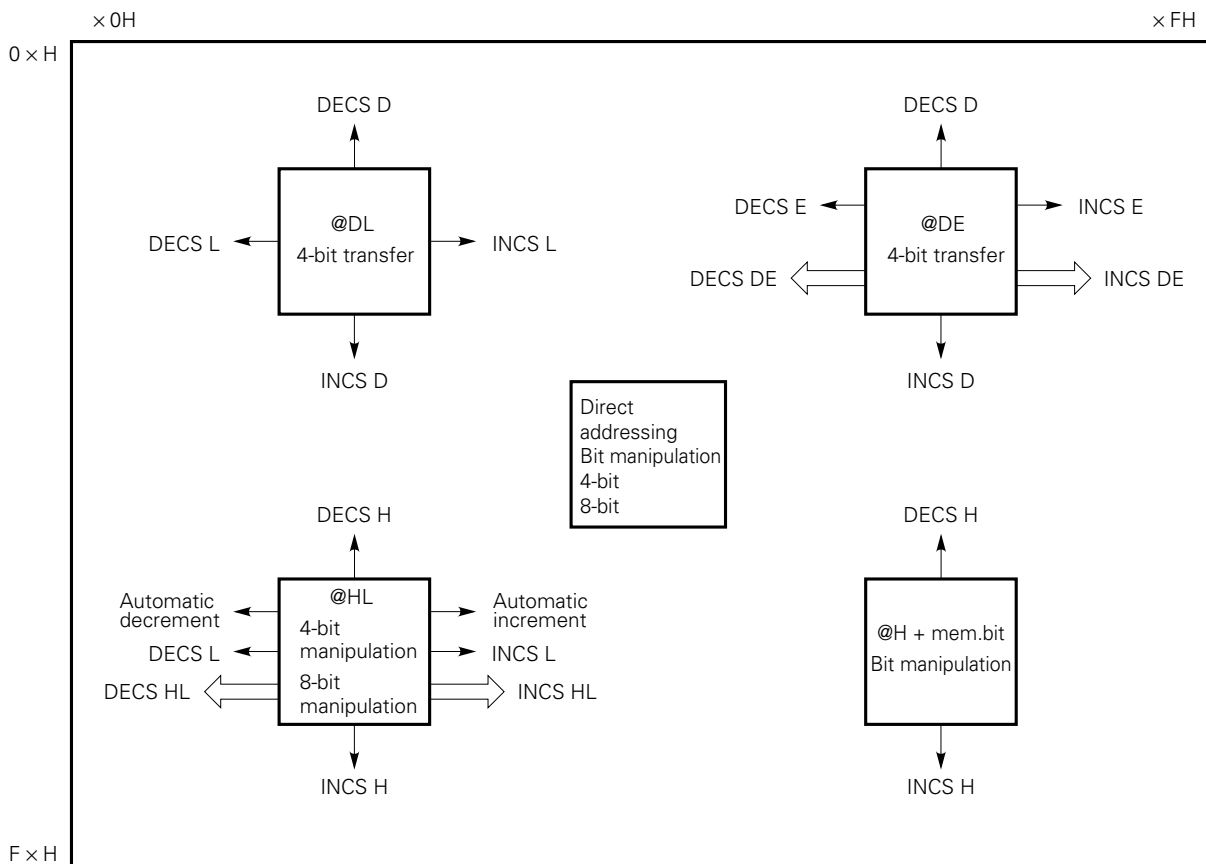
## 2.2 GENERAL REGISTER BANK CONFIGURATION

The µPD75518(A) 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. 2-3**.) 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 µPD75518(A) 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   Example of the 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. (In this case, the RBS needs to be saved and restored.) |
| Multiple (triple or more) interrupt processing | Save the registers with PUSH or POP. |

The RBE and RBS are set as indicated below.

**Example**    SET1  RBE ; RBE ← 1
             CLR1  RBE ; RBE ← 0
             SEL    RB0 ; RBS ← 0
             SEL    RB3 ; RBS ← 3

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. 2-4**.)

**Example**    SET1  RBE      ; RBE ← 1
             SEL    RB2     ; RBS ← 2
             MOV  XA, #18H ; XA ← 18H
             ADDS  HL, XA  ; HL ← HL+XA
             SUBS  HL', XA  ; HL' ← HL'–XA (HL' is HL of register bank 3)
             INCS  HL        ; HL ← HL+1
             MOV  XA, #00H ; XA ← 00H (string effect)
             MOV  XA, #10H ; XA ← 10H (string effect)

**Fig. 2-3   General Register Configuration (4-Bit Processing)**

| | | |
|---|---|---|
| X    01H | A    00H | |
| H    03H | L    02H | Register bank 0 (RBE·RBS = 0) |
| D    05H | E    04H | |
| B    07H | C    06H | |
| X    09H | A    08H | |
| H    0BH | L    0AH | Register bank 1 (RBE·RBS = 1) |
| D    0DH | E    0CH | |
| B    0FH | C    0EH | |
| X    11H | A    10H | |
| H    13H | L    12H | Register bank 2 (RBE·RBS = 2) |
| D    15H | E    14H | |
| B    17H | C    16H | |
| X    19H | A    18H | |
| H    1BH | L    1AH | Register bank 3 (RBE·RBS = 3) |
| D    1DH | E    1CH | |
| B    1FH | C    1EH | |

**Fig. 2-4   General Register Configuration (8-Bit Processing)**
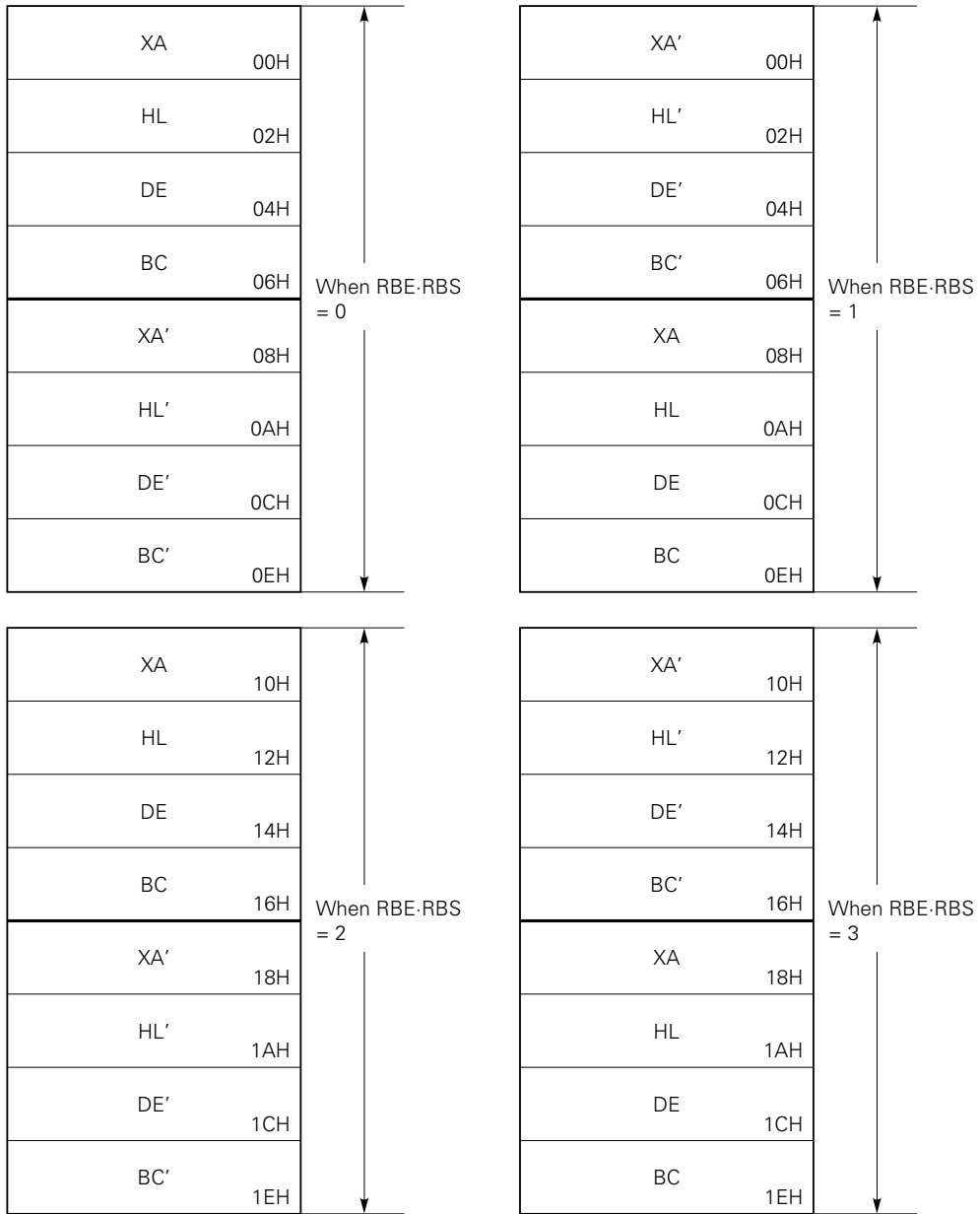
### 2.3 MEMORY-MAPPED I/O

The µPD75518(A) 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.

**Table 2-3   Addressing Modes Applicable to Peripheral Hardware**

| | 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, EOT, IE×××, IRQ×××, PORTn.× |
| | Indirect addressing mode specifying pmem.@L regardless of MBE and MBS setting | BSBn.×<br>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) | |

Fig. 2-5 summarizes the I/O map of the µPD75518(A).

The items in Fig. 2-5 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

  ◯:  Bits can be manipulated on an indicated bit (1-, 4-, or 8-bit) basis.

  △:  Particular bits can be manipulated.  For these bits, see Remarks.

  –  :  Bits cannot be manipulated on an indicated bit (1-, 4-, or 8-bit) basis.
- Bit manipulation addressing:

  Bit manipulation addressing applicable in hardware bit manipulation

**Fig. 2-5** $\mu$**PD75518(A) I/O Map (1/4)**

| Address | Hardware name (symbol) b3 \| b2 \| b1 \| b0 | | | | R/W | Number of bits that can be manipulated | | | Bit manipulation addressing | Remarks |
|---------|----|----|----|----|-----|-------|--------|--------|-----------------------------|---------|
| | | | | | | 1 bit | 4 bits | 8 bits | | |
| F80H | Stack pointer (SP) | | | | R/W | – | – | ○ | | Bit 0 is fixed to 0 |
| F82H | Register bank select register (RBS) | | Bank select register (BS) | | R | – | ○ | ○ | | **Note 1** |
| F83H | Memory bank select register (MBS) | | | | | – | ○ | | | |
| F84H | Stack bank select register (SBS) | | | | R/W | – | ○ | – | | Bits 3 and 2 are always set to 0. |
| F85H | Basic interval timer mode register (BTM) | | | | W | △ | ○ | – | mem.bit | Only bit 3 can be manipulated |
| F86H | Basic interval timer (BT) | | | | R | – | – | ○ | | |

| Address | Hardware name (symbol) b3 \| b2 \| b1 \| b0 | R/W | 1 bit | 4 bits | 8 bits | Bit manipulation addressing | Remarks |
|---------|---------------------------------------------|-----|-------|--------|--------|-----------------------------|---------|
| F90H | Timer pulse generator (TPGM) | W | △ | – | ○ | mem.bit | Only bit 3 allows bit manipulation. |
| | | | – | – | | | |
| F94H | Timer/pulse generator modulo register (MODL) | R/W | – | – | ○ | | |
| F96H | Timer/pulse generator modulo register (MODH) | R/W | – | – | ○ | | |
| F98H | Clock mode register (WM) | – | – | – | ○ | | |
| | | W | – | – | | | |

| Address | Hardware name (symbol) b3 \| b2 \| b1 \| b0 | R/W | 1 bit | 4 bits | 8 bits | Bit manipulation addressing | Remarks |
|---------|---------------------------------------------|-----|-------|--------|--------|-----------------------------|---------|
| FA0H | Timer/event counter 0 mode register (TM0) | W | △ | – | ○ | mem.bit | Only bit 3 can be manipulated |
| | | | – | – | | | |
| FA2H | TOE0 **Note 2** | W | ○ | – | – | mem.bit | |
| FA4H | Timer/event counter 0 count register (T0) | R | – | – | ○ | | |
| FA6H | Timer/event counter 0 modulo register (TMOD0) | W | – | – | ○ | | |

**Notes 1.** Can be operated separately as the RBS and MBS during 4-bit manipulation.
Can also be operated as the BS during 8-bit manipulation.

**2.** TOE0: Timer/event counter 0 output enable flag (W)

**Fig. 2-5 $\mu$PD75518(A) I/O Map (2/4)**

| Address | Hardware name (symbol) b3 | b2 | b1 | b0 | R/W | Number of bits that can be manipulated 1 bit | 4 bits | 8 bits | Bit manipulation addressing | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| FB0H | IST1 | IST0 | MBE | RBE | R/W | ○ | ○ | ○ | fmem.bit | |
| | Program status word (PSW) | | | | R | – | – | | | |
| FB2H | Interrupt priority select register (IPS) | | | | W | – | ○ | – | | Manipulated with EI/DI instruction |
| FB3H | Processor clock control register (PCC) | | | | W | – | ○ | | | |
| FB4H | INT0 mode register (IM0) | | | | W | – | ○ | – | | Bit 2 is fixed to 0 |
| FB5H | INT1 mode resistor (IM1) | | | | W | – | ○ | | | Bits 3, 2, and 1 are fixed to 0 |
| FB6H | INT2 mode register (IM2) | | | | W | – | ○ | – | | Bits 3 and 2 are fixed to 0 |
| FB7H | System clock control register (SCC) | | | | W | ○ | – | | | Bits 2 and 1 are fixed to 0 |
| FB8H | IE4 | IRQ4 | IEBT | IRQBT | R/W | ○ | ○ | – | fmem.bit | |
| FB9H | | | | EOT | R/W | ○ | ○ | | | |
| FBAH | | | IEW | IRQW | R/W | ○ | ○ | – | | |
| FBBH | | | IETPG | IRQTPG | R/W | ○ | – | | | |
| FBCH | | | IET0 | IRQT0 | R/W | ○ | ○ | – | | |
| FBDH | | | IECSI0 | IRQCSI0 | R/W | ○ | ○ | | | |
| FBEH | IE1 | IRQ1 | IE0 | IRQ0 | R/W | ○ | ○ | – | | |
| FBFH | | | IE2 | IRQ2 | R/W | ○ | ○ | | | |

| Address | Hardware name (symbol) | R/W | 1 bit | 4 bits | 8 bits | Bit manipulation addressing | Remarks |
|---|---|---|---|---|---|---|---|
| FC0H | Bit sequential buffer 0 (BSB0) | R/W | ○ | ○ | ○ | mem.bit pmem.@L | |
| 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 | Serial operation mode register 1 (CSIM1) | W | – | – | ○ | mem.bit | Only bit 7 allows bit manipulation. |
| | CSIE1 | | △ | – | | | |
| FCCH | Serial I/O shift register 1 (SIO0) | R/W | – | – | ○ | | |

**Remarks 1.** IE××× : Interrupt enable flag
　　　 **2.** IRQ×××: Interrupt request flag

**Fig. 2-5　μPD75518(A) I/O Map (3/4)**

| Address | Hardware name (symbol) | | | | R/W | Number of bits that can be manipulated | | | Bit manipulation addressing | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| | b3 | b2 | b1 | b0 | | 1 bit | 4 bits | 8 bits | | |
| FD0H | Clock output mode register (CLOM) | | | | W | – | ○ | – | | |
| FD8H | SOC | EOC | | | W | △ | – | ○ | | b3: 1-bit write b2: 1-bit read |
| | 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 0 (CSIM0) | | | | W | – | – | ○ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | CSIE0 | COI | WUP | | R/W | ○ | ○ | | mem.bit | b6: 1-bit read |
| FE2H | CMDD | RELD | CMDT | RELT | R/W | ○ | – | – | mem.bit | All bits allow bit manipulation only |
| | 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 | | | | | | |
| FEEH | PM11 | PM10 | PM9 | –Note | W | – | – | ○ | | |
| | Port mode register group C (PMGC) | | | | | | | | | |
| | –Note | PM14 | PM13 | PM12 | | | | | | |

**Note** When developing a program, set 0 to the following two bits of the port mode register group C (PMGC):

FEEH, b0 (Equivalent to PM8)

FEFH, b3 (Equivalent to PM15)

For, while this port on the chip side is used for input only, the corresponding port on the emulator is an I/O port.

**Fig. 2-5   µPD75518(A) I/O Map (4/4)**

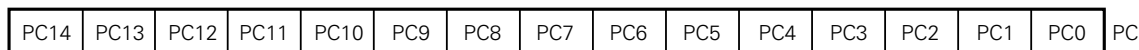| Address | Hardware name (symbol) | | | | R/W | Number of bits that can be manipulated | | | Bit manipulation addressing | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| | b3 | b2 | b1 | b0 | | 1 bit | 4 bits | 8 bits | | |
| FF0H | Port 0 | | (PORT0) | | R | ◯ | ◯ | – | | |
| FF1H | Port 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**Note** | KR3 | KR2 | KR1 | KR0 | R/W | ◯ | ◯ | ◯ | fmem.bit pmem.@L | |
| | Port 6 | | (PORT6) | | | | | | | |
| FF7H**Note** | KR7 | KR6 | KR5 | KR4 | R/W | ◯ | ◯ | | | |
| | Port 7 | | (PORT7) | | | | | | | |
| FF8H | Port 8 | | (PORT8) | | R | ◯ | ◯ | – | | |
| FF9H | Port 9 | | (PORT9) | | R/W | ◯ | ◯ | | | |
| FFAH | Port 10 | | (PORT10) | | R/W | ◯ | ◯ | – | | |
| FFBH | Port 11 | | (PORT11) | | R/W | ◯ | ◯ | | | |
| FFCH | Port 12 | | (PORT12) | | R/W | ◯ | ◯ | – | | |
| FFDH | Port 13 | | (PORT13) | | R/W | ◯ | ◯ | | | |
| FFEH | Port 14 | | (PORT14) | | R/W | ◯ | ◯ | – | | |
| FFFH | Port 15 | | (PORT15) | | R | ◯ | ◯ | | | |

**Note**   KR0 to KR7 are read-only.   In 4-bit parallel input processing, PORT6 or PORT7 is specified.

## 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 | PC |
|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|----|

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, and 0 in PC14 for initialization.

### 3.2 PROGRAM MEMORY (ROM): 32640 WORDS $\times$ 8 BITS

The program memory is a mask-programmable ROM with a configuration of 32640 words$\times$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 5F7FH.  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 5F7FH 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 000DH**
  Vector 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 steps.  (See **Section 8.1**.)

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

| | 7 | 6 | | 0 | |
|---|---|---|---|---|---|
| 0000H | MBE | RBE | Internal reset start address | (high-order 6 bits) | |
| | | | Internal reset start address | (low-order 8 bits) | |
| 0002H | MBE | RBE | INTBT/INT4 start address | (high-order 6 bits) | |
| | | | INTBT/INT4 start address | (low-order 8 bits) | |
| 0004H | MBE | RBE | INT0 start address | (high-order 6 bits) | |
| | | | INT0 start address | (low-order 8 bits) | |
| 0006H | MBE | RBE | INT1 start address | (high-order 6 bits) | |
| | | | INT1 start address | (low-order 8 bits) | |
| 0008H | MBE | RBE | INTCSI0 start address | (high-order 6 bits) | |
| | | | INTCSI0 start address | (low-order 8 bits) | |
| 000AH | MBE | RBE | INTT0 start address | (high-order 6 bits) | |
| | | | INTT0 start address | (low-order 8 bits) | |
| 000CH | MBE | RBE | INTTPG start address | (high-order 6 bits) | |
| | | | INTTPG start address | (low-order 8 bits) | |

0020H
007FH : GETI instruction reference table
0080H
07FFH
0800H
0FFFH
1000H
1FFFH
2000H
2FFFH
3000H
3FFFH
4000H
4FFFH
5000H
5FFFH
6000H
6FFFH
7000H
7F7FH

CALLF !faddr instruction entry address

BRCB !caddr instruction branch address

BR !addr instruction branch address

CALL addr instruction branch address

Branch/call address specified in GETI insturction

BR BCDE BR BCXA branch address

BRA !addr instruction branch address

CALLA !addr instruction branch address

BR $addr instruction relative branch address (–15 to –1, +2 to +16)

BRCB !caddr instruction branch address

**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.
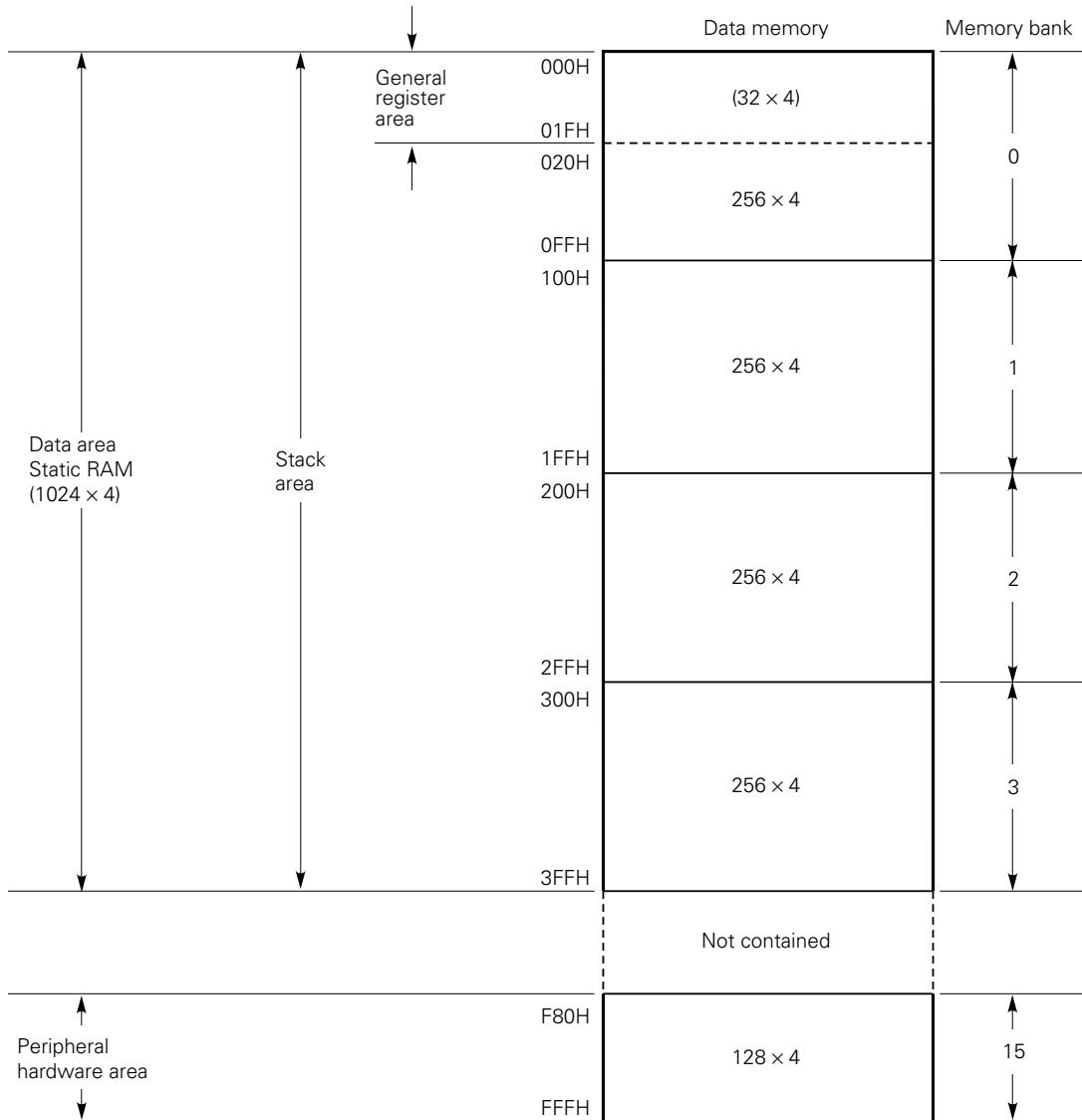
### 3.3   DATA MEMORY (RAM)

The data memory is divided into a data area and a peripheral hardware area as shown in Fig. 3-3.
The data memory consists of the following memory banks, with each bank made of 256 words × 4 bits:

• Memory banks 0, 1, 2, and 3 (data area)
• Memory bank 15 (peripheral hardware area)

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

**(1) Data area**

The data area consists of a static RAM, and is used for storing data and as stack memory for subroutine and interrupt execution. The memory can hold data even if CPU operation is stopped in the standby mode, so that it is suitable for holding memory contents with a battery for a long time. The data area can be manipulated with memory manipulation instructions.

The static RAM is mapped in memory banks 0, 1, 2, and 3, with each made up of $256 \times 4$ bits. Bank 0 is used as a data area, but can also be used as a general register area (000H to 01FH).

Whole addresses of memory banks 0, 1, 2, and 3 (000H to 3FFH) can be used as a stack area.

The static RAM has a configuration of four bits per address. However, use of manipulation instructions enables 1-, 4-, and 8-bit manipulation. Note that an even address must be specified in an 8-bit manipulation instruction.

**(a) General register area**

The general register area can be manipulated with either general register manipulation instructions or memory manipulation instructions. Up to 32 4-bit registers are available. Of the 32 general registers, registers not used by the program can be used as a data area or stack area.

**(b) Stack memory area**

The stack area can be allocated within a bank with the stack pointer (SP). The bank for the stack area is selected from the memory banks 0, 1, 2, and 3 with the stack bank select register (SBS). Stack area can be used as a save area for subroutine or interrupt execution.

Use memory manipulation instructions to manipulate the stack bank select register (SBS) and the stack pointer (SP).

**(2) Peripheral hardware area**

The peripheral hardware area is mapped at addresses F80H to FFFH of memory bank 15.

Memory manipulation instructions are used to manipulate the peripheral hardware area as well as the static RAM area. Note that, however, the number of bits to be manipulated at a time varies according to the individual addresses. Addresses to which no peripheral hardware is assigned cannot be accessed since such address locations contain no data memory.

### 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. 3-4  General Register Format**

Data memory

| Address | 3 ... 0 | |
|---|---|---|
| 000H | A register | |
| 001H | X register | |
| 002H | L register | |
| 003H | H register | Register bank 0 |
| 004H | E register | |
| 005H | D register | |
| 006H | C register | |
| 007H | B register | |
| 008H ... 00FH | Same as bank 0 | Register bank 1 |
| 010H ... 017H | Same as bank 0 | Register bank 2 |
| 018H ... 01FH | Same as bank 0 | Register bank 3 |

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

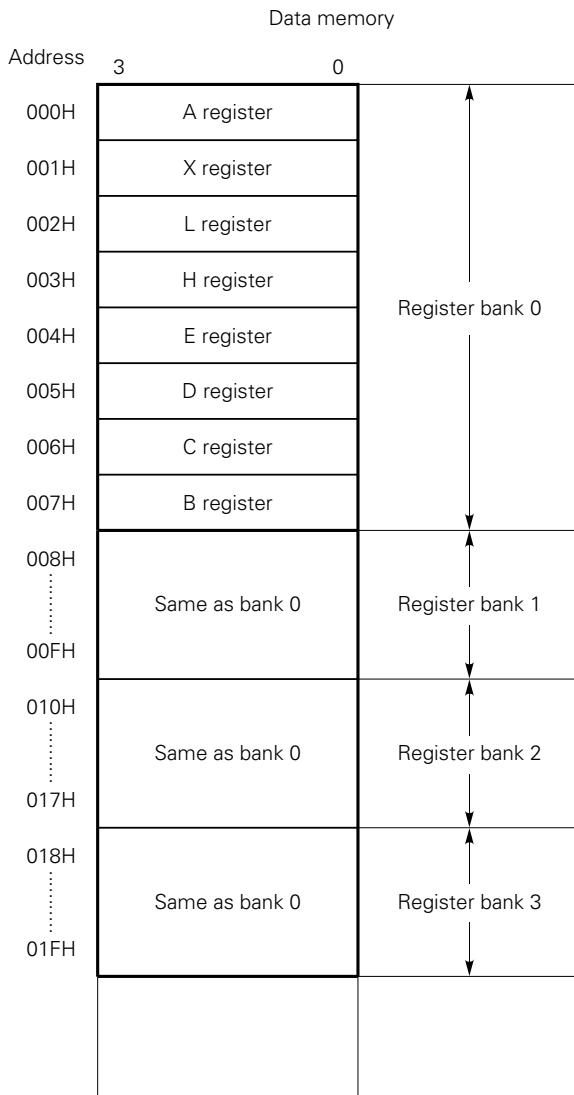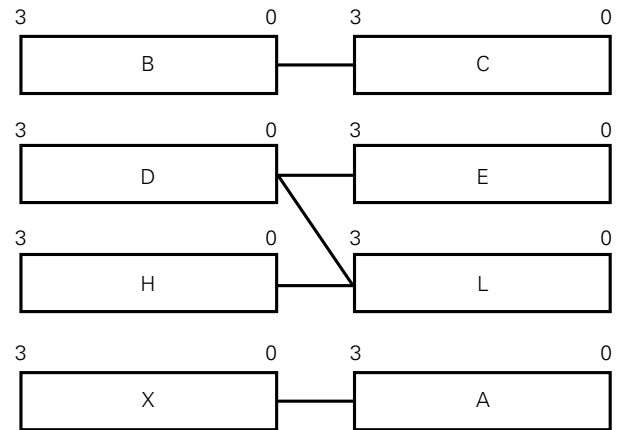| 3 ... 0 | 3 ... 0 |
|---|---|
| B | C |
| D | E |
| H | L |
| X | A |

## 3.5 ACCUMULATORS

In the $\mu$PD75518(A), 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. 3-6   Accumulators**



| CY | Bit accumulator |

| A | 4-bit accumulator |

| X | A | 8-bit accumulator |
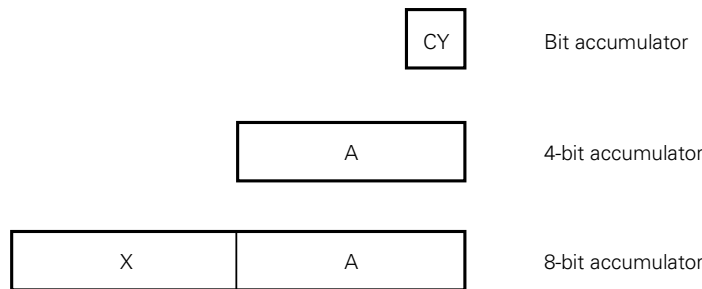
## 3.6   STACK POINTER (SP) AND STACK BANK SELECT REGISTER (SBS)

The $\mu$PD75518(A) 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 2-bit SBS.  (See **Table 3-1**.)

**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 |

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. 3-8 and 3-9 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.

When the SP is initialized to 00H, a stack operation starts at the high-order address (nFFH) of memory bank (n) 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. 3-7   Stack Pointer and Stack Bank Select Register Formats**

Address                                                                                    Symbol

| F80H | SP7 | SP6 | SP5 | SP4 | SP3 | SP2 | SP1 | 0 | SP |

| F84H | | | | | 0 | 0 | SBS1 | SBS0 | SBS |

```
                            000H  ┌─────────────────┐      ┌─ SP
        ┌────────┐                │                 │      │
        │  SBS   │────────┐──────→│  Memory bank 0  │   ⤬  └→
        └────────┘        │       │                 │
                            0FFH  │                 │
                            100H  ├─────────────────┤      ┌─ SP
                           ┌─────→│                 │      │
                           │      │  Memory bank 1  │   ⤬  └→
                           │ 1FFH │                 │
                           │ 200H ├─────────────────┤      ┌─ SP
                           ├─────→│                 │      │
                           │      │  Memory bank 2  │   ⤬  └→
                           │ 2FFH │                 │
                           │ 300H ├─────────────────┤      ┌─ SP
                           └─────→│                 │      │
                                  │  Memory bank 3  │      └→
                            3FFH  └─────────────────┘
```
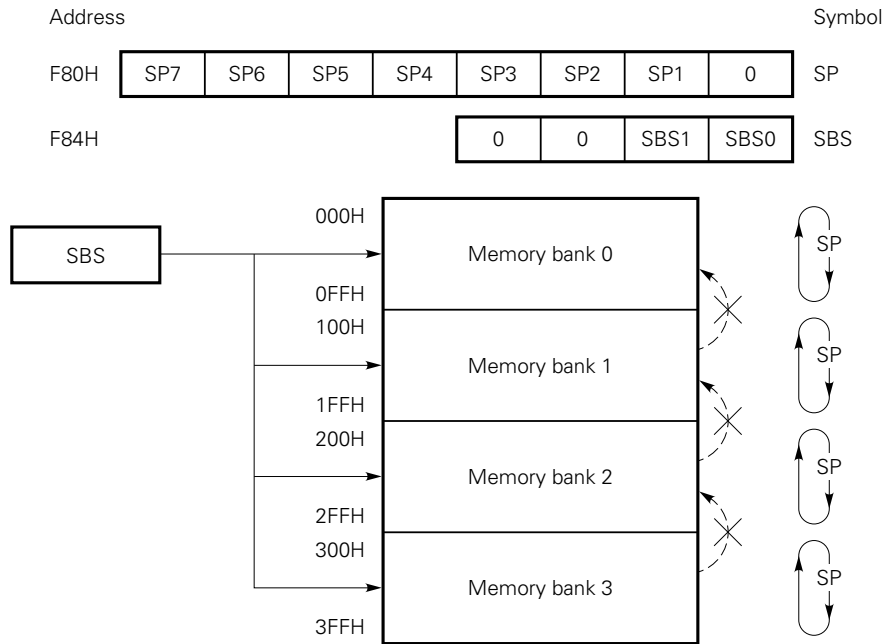
**Example**   SP initialization

In this example, stack area is allocated in memory bank 2 and stack operation starts at address 2FFH.

```
SEL   MB15      ; or CLR1 MBE
MOV   A, #2
MOV   SBS, A    ; Specify memory bank 2 as a stack area
MOV   XA, #00H
MOV   SP, XA    ; SP ← 00H
```
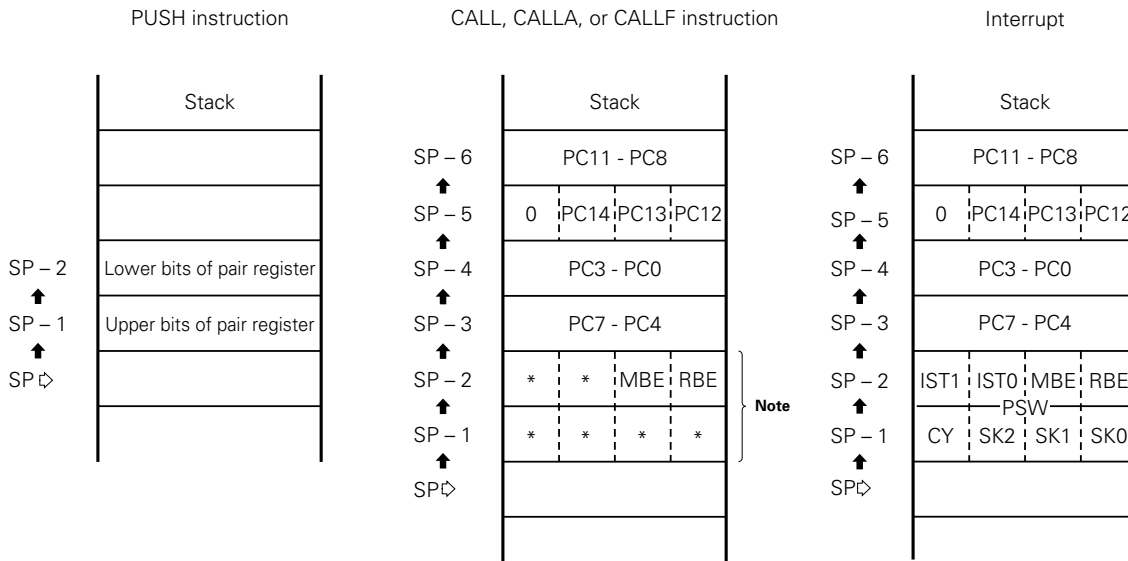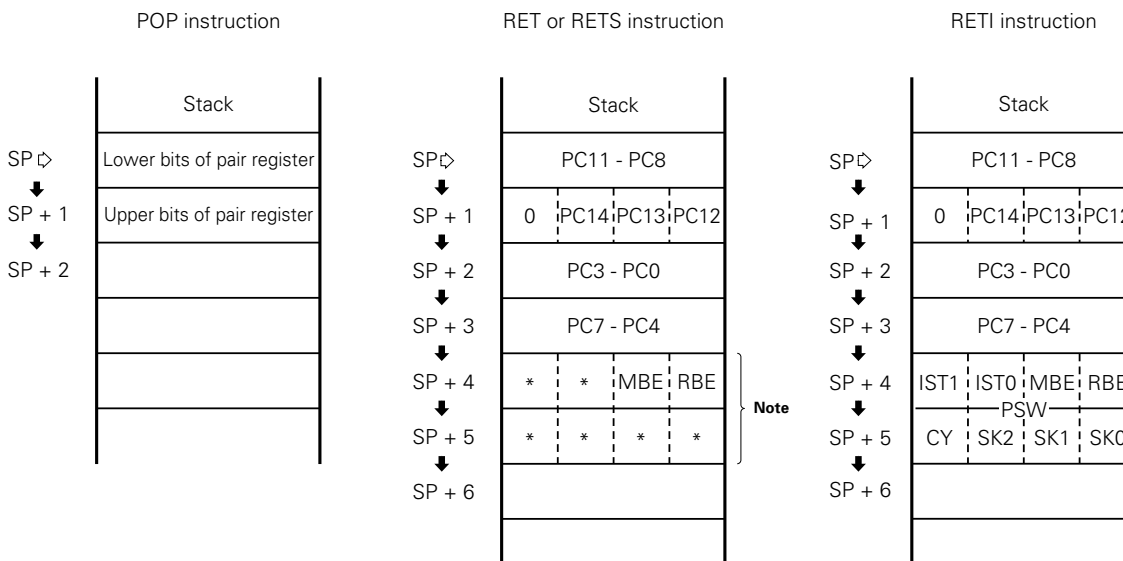
**Fig. 3-8   Data Saved to Stack Memory**

| PUSH instruction | CALL, CALLA, or CALLF instruction | Interrupt |
|---|---|---|

PUSH instruction

| | Stack |
|---|---|
| | |
| | |
| SP – 2 | Lower bits of pair register |
| ↑ | |
| SP – 1 | Upper bits of pair register |
| ↑ | |
| SP ⇨ | |
| | |

CALL, CALLA, or CALLF instruction

| | Stack |
|---|---|
| SP – 6 | PC11 - PC8 |
| ↑ | |
| SP – 5 | 0 ¦ PC14 ¦ PC13 ¦ PC12 |
| ↑ | |
| SP – 4 | PC3 - PC0 |
| ↑ | |
| SP – 3 | PC7 - PC4 |
| ↑ | |
| SP – 2 | * ¦ * ¦ MBE ¦ RBE |
| ↑ | } **Note** |
| SP – 1 | * ¦ * ¦ * ¦ * |
| ↑ | |
| SP ⇨ | |
| | |

Interrupt

| | Stack |
|---|---|
| SP – 6 | PC11 - PC8 |
| ↑ | |
| SP – 5 | 0 ¦ PC14 ¦ PC13 ¦ PC12 |
| ↑ | |
| SP – 4 | PC3 - PC0 |
| ↑ | |
| SP – 3 | PC7 - PC4 |
| ↑ | |
| SP – 2 | IST1 ¦ IST0 ¦ MBE ¦ RBE |
| ↑ | ——PSW—— |
| SP – 1 | CY ¦ SK2 ¦ SK1 ¦ SK0 |
| ↑ | |
| SP ⇨ | |
| | |

**Fig. 3-9   Data Restored from Stack Memory**

POP instruction

| | Stack |
|---|---|
| SP ⇨ | Lower bits of pair register |
| ↓ | |
| SP + 1 | Upper bits of pair register |
| ↓ | |
| SP + 2 | |
| | |
| | |
| | |

RET or RETS instruction

| | Stack |
|---|---|
| SP ⇨ | PC11 - PC8 |
| ↓ | |
| SP + 1 | 0 ¦ PC14 ¦ PC13 ¦ PC12 |
| ↓ | |
| SP + 2 | PC3 - PC0 |
| ↓ | |
| SP + 3 | PC7 - PC4 |
| ↓ | |
| SP + 4 | * ¦ * ¦ MBE ¦ RBE |
| ↓ | } **Note** |
| SP + 5 | * ¦ * ¦ * ¦ * |
| ↓ | |
| SP + 6 | |
| | |

RETI instruction

| | Stack |
|---|---|
| SP ⇨ | PC11 - PC8 |
| ↓ | |
| SP + 1 | 0 ¦ PC14 ¦ PC13 ¦ PC12 |
| ↓ | |
| SP + 2 | PC3 - PC0 |
| ↓ | |
| SP + 3 | PC7 - PC4 |
| ↓ | |
| SP + 4 | IST1 ¦ IST0 ¦ MBE ¦ RBE |
| ↓ | ——PSW—— |
| SP + 5 | CY ¦ SK2 ¦ SK1 ¦ SK0 |
| ↓ | |
| SP + 6 | |
| | |

**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.

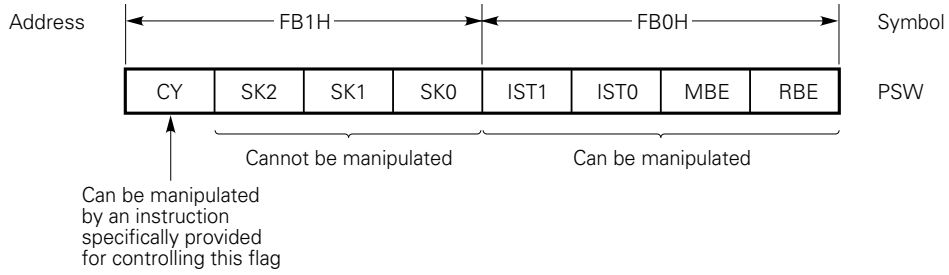**Fig. 3-10 Program Status Word Format**



**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

**Example** Bit 3 at address 3FH is ANDed with P33, then the result is output to P50.

```
MOV    H, #3H          ; Set high-order 4 bits in register H
MOV1   CY, @H+0FH.3  ; CY ← Bit 3 at 3FH
AND1   CY, PORT3.3     ; CY ← CY ∧ P33
MOV1   PORT5.0, CY     ; P50 ← CY
```

**(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. Any interrupts are acceptable. |
| 0 | 1 | Status 1 | A lower- or higher-priority interrupt is being serviced. Higher-priority interrupts are acceptable. |
| 1 | 0 | Status 2 | A higher-priority interrupt is being serviced. 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 $\overline{\text{RESET}}$ signal occurrence 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 $\overline{\text{RESET}}$ signal occurrence 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.
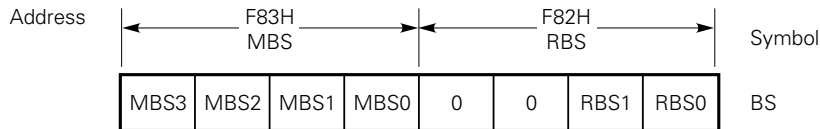
## 3.8 BANK SELECT REGISTER (BS)

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. 3-11   Bank Select Register Format**

| Address | F83H MBS | | | | F82H RBS | | | | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| | MBS3 | MBS2 | MBS1 | MBS0 | 0 | 0 | RBS1 | RBS0 | BS |

**(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.  Note, however, that the µPD75518(A) allows only memory banks 0, 1, 2, 3, and 15 to 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 $\overline{\text{RESET}}$ signal occurrence 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 $\overline{\text{RESET}}$ signal occurrence 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
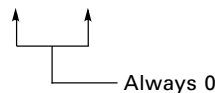
**Remark**  ×: Don't care

## 4. PERIPHERAL HARDWARE FUNCTIONS

### 4.1 DIGITAL I/O PORTS

The μPD75518(A) 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 |

Table 4-1 lists the I/O port manipulation instructions. These instructions provide a wide range of control including 8-bit I/O and bit manipulation as well as 4-bit I/O.

**Examples 1.** Test the state of P13, then output different values to ports 4 and 5 according to the test result.

```
SKT   PORT1.3    ; Skip if bit 3 of port 1 is 1
MOV   XA, #18H    ; XA ← 18H  ⎫
MOV   XA, #14H    ; XA ← 14H  ⎬ Consecutive
                             ⎭
SEL   MB15        ; or CLR1 MBE
OUT   PORT4, XA   ; Ports 5 and 4 ← XA
```

**2.** SET1 PORT4.@L   ; Set the bit of ports 4 to 7 specified by the L register to 1

**Table 4-1  I/O Pin Manipulation Instructions**

| | Port 0 | Port 1 | Port 2 | Port 3 | Port 4 | Port 5 | Port 6 | Port 7 | Port 8 | Port 9 | Port 10 | Port 11 | Port 12 | Port 13 | Port 14 | Port 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IN A, PORTn [Note 1] | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| IN XA, PORTn [Note 1] | — | — | — | — | ○ | ○ | ○ | ○ | — | — | — | — | — | — | — | — |
| OUT PORTn.A [Note 1] | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | ○ | ○ | ○ | ○ | ○ | ○ | — |
| OUT PORTn.XA [Note 1] | — | — | — | — | ○ | ○ | ○ | ○ | — | — | — | — | — | — | — | — |
| SET1 PORTn.bit | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | ○ | ○ | ○ | ○ | ○ | ○ | — |
| SET1 PORTn.@L [Note 2] | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | ○ | ○ | ○ | ○ | ○ | ○ | — |
| CLR1 PORTn.bit | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | ○ | ○ | ○ | ○ | ○ | ○ | — |
| CLR1 PORTn.@L [Note 2] | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | ○ | ○ | ○ | ○ | ○ | ○ | — |
| SKT PORTn.bit | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| SKT PORTn.@L [Note 2] | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| SKF PORTn.bit | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| SKF PORTn.@L [Note 2] | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| MOV1 CY, PORTn.bit | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| MOV1 CY, PORTn.@L [Note 2] | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| MOV1 PORTn.bit, CY | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | ○ | ○ | ○ | ○ | ○ | ○ | — |
| MOV1 PORTn.@L, CY [Note 2] | — | — | ○ | ○ | ○ | ○ | ○ | ○ | — | ○ | ○ | ○ | ○ | ○ | ○ | — |
| AND1 CY, PORTn.bit | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| AND1 CY, PORTn.@L [Note 2] | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| OR1 CY, PORTn.bit | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| OR1 CY, PORTn.@L [Note 2] | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| XOR1 CY, PORTn.bit | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| XOR1 CY, PORTn.@L [Note 2] | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |

Notes 1. Before an instruction is executed, MBE must be set to 0, or MBS must be set to 15 when MBE is 1.

2. The lower 2 bits of an address and a bit address are specified indirectly with the L register.

**(1) Types, features, and configurations of digital I/O ports**

Table 4-2 lists the types of digital I/O ports.

Fig. 4-2 through 4-8 present the configurations of the ports.

**Table 4-2  Types and Features of Digital Ports**

| Port name | Function | Operation and feature | | Remarks |
|---|---|---|---|---|
| PORT0 | 4-bit input | Allows read and test at any time regardless of the operation modes of dual function pins. | | Also used as INT4, $\overline{SCK0}$, SO0/ SB0, and SI0/SB1. |
| PORT1 | | | | Also used as INT0 to INT2, and TI0. |
| PORT2 | 4-bit I/O | Allows input or output mode setting in units of 4 bits. | | Also used as PTO0, PCL, and BUZ. |
| PORT3**Note** | | Allows input or output mode setting in units of 1 or 4 bits. | | — |
| PORT4**Note** | 4-bit I/O (N-chan-nel open-drain 10 V) | Allows input or output mode setting in units of 4 bits. | Ports 4 and 5 may be paired, allowing data I/O in units of 8 bits. | The use of pull-up resistors can be specified by mask options in units of bits. |
| PORT5**Note** | | | | |
| PORT6 | 4-bit I/O | Allows input or output mode setting in units of 1 or 4 bits. | Ports 6 and 7 may be paired, allowing data I/O in units of 8 bits. | Also used as KR0 to KR3. |
| PORT7 | | Allows input or output mode setting in units of 4 bits. | | Also used as KR4 to KR7. |
| PORT8 | 4-bit input | Allows read and test at any time regardless of the operation modes of dual function pins. | | Also used as PPO, $\overline{SCK1}$, SO1, and SI1. |
| PORT9 | 4-bit I/O | Allows input or output mode setting in units of 4 bits. | | The use of a pull-down resistor can be specified by a mask op-tion in units of bits. |
| PORT10 | 4-bit I/O | Allows input or output mode setting in units of 4 bits. | | — |
| PORT11 | | | | |
| PORT12 | 4-bit I/O (N-chan-nel open-drain 10 V) | Allows input or output mode setting in units of 4 bits. | | The use of pull-up resistors can be specified by mask options in units of bits. |
| PORT13 | | | | |
| PORT14 | | | | |
| PORT15 | 4-bit input | Allows read and test at any time regardless of the operation modes of dual function pins. | | Also used as AN4 to AN7. |

**Note**  This port can directly drive the LED.

P10 is also used as an external vectored interrupt input pin.  This input is provided with a noise eliminator. (See **Section 5.2** for details.)

The use of pull-up resistors can be specified for ports 0 (excluding pin P00/INT4), 1 to 3, 6, and 7 by software.

**(2)  Setting the I/O mode**

As shown in Fig. 4-9, the I/O mode for each I/O port is set with the port mode register.

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 bit is set to 1.

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

The generation of a $\overline{\text{RESET}}$ signal clears all the bits of each port mode register to 0.  This means that the output buffers are set off and all ports function in the input mode.
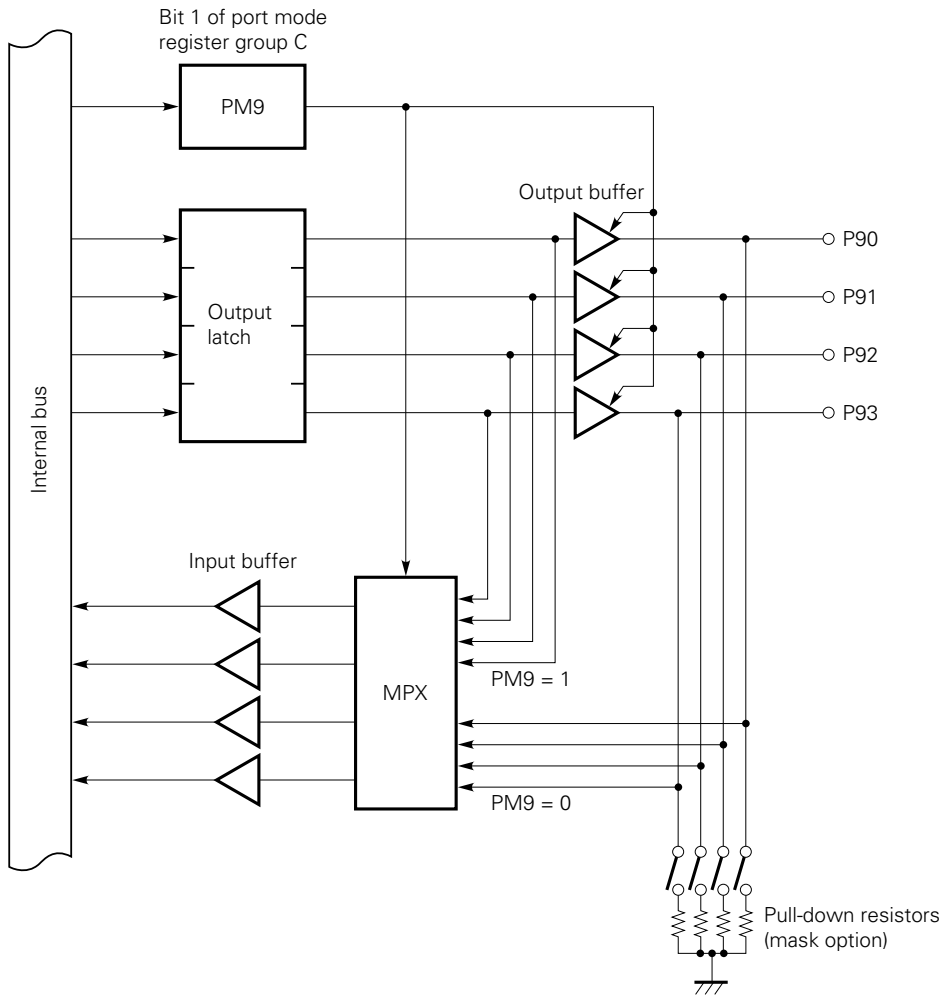
**Fig. 4-2   Configuration of Ports 0, 1, and 8**
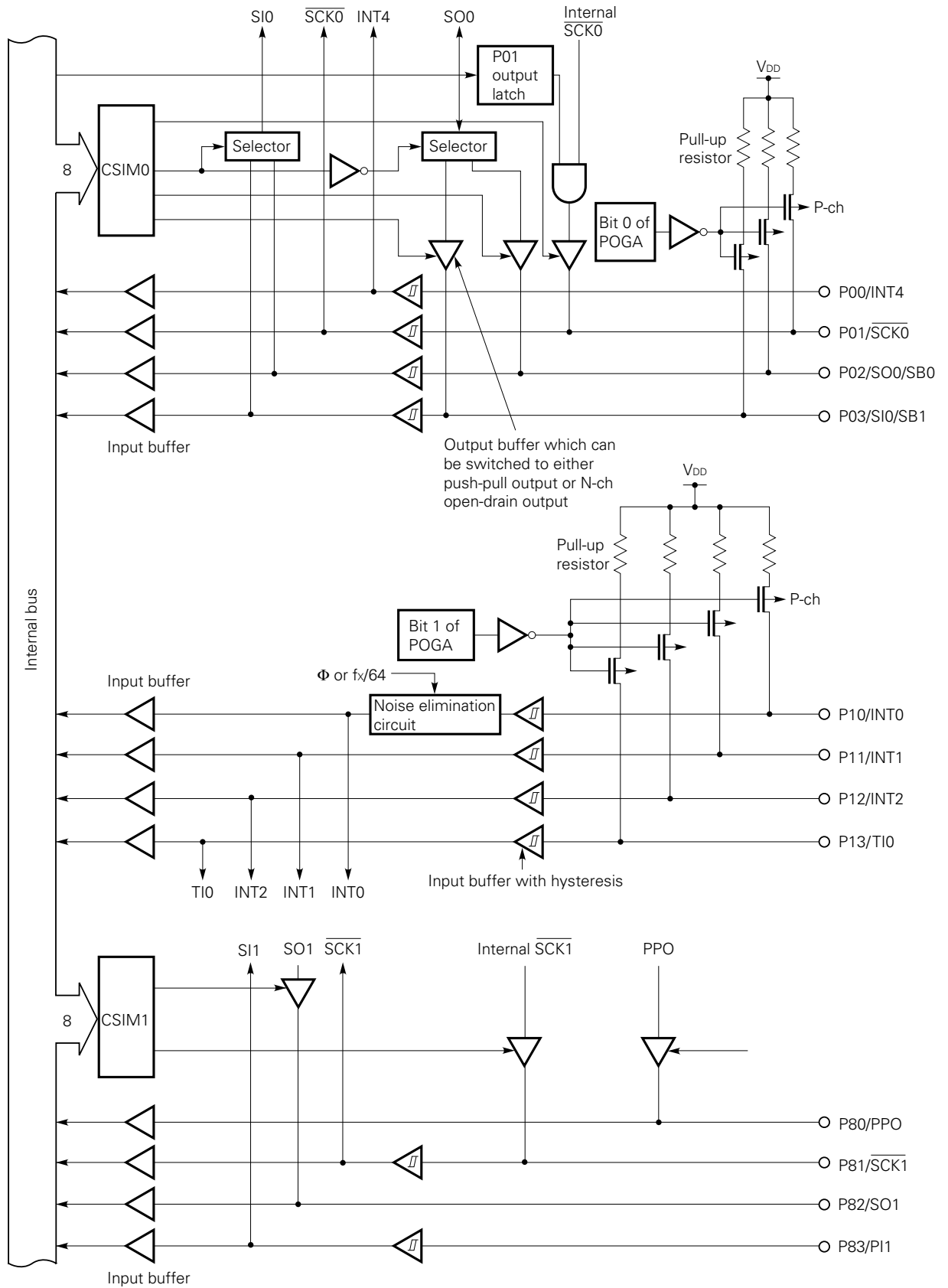
**Fig. 4-3 Configuration of Ports 3n and 6n (n = 0 to 3)**
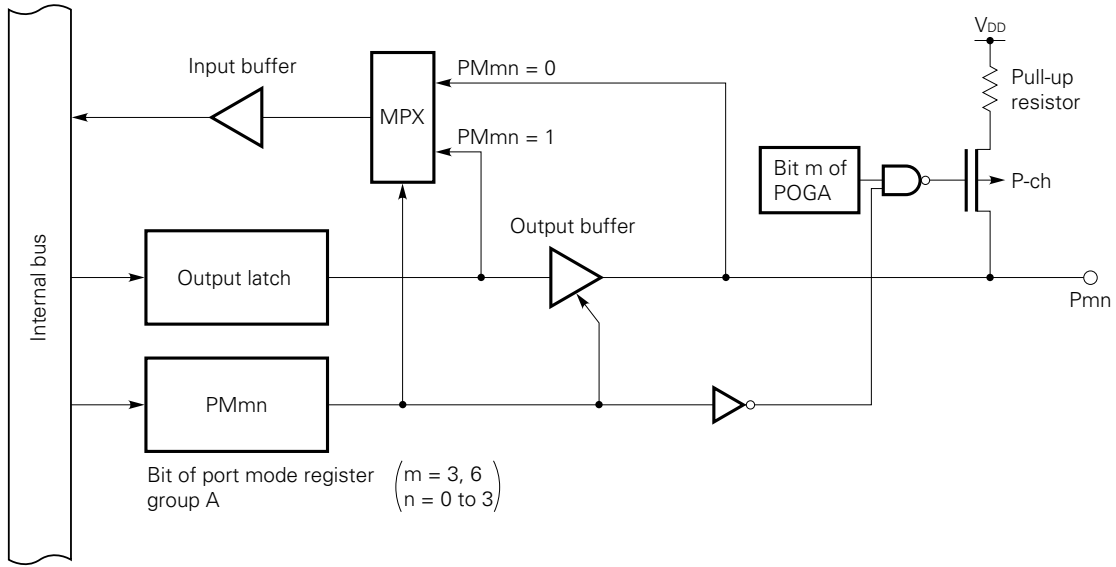


**Fig. 4-4 Configuration of Ports 2 and 7**
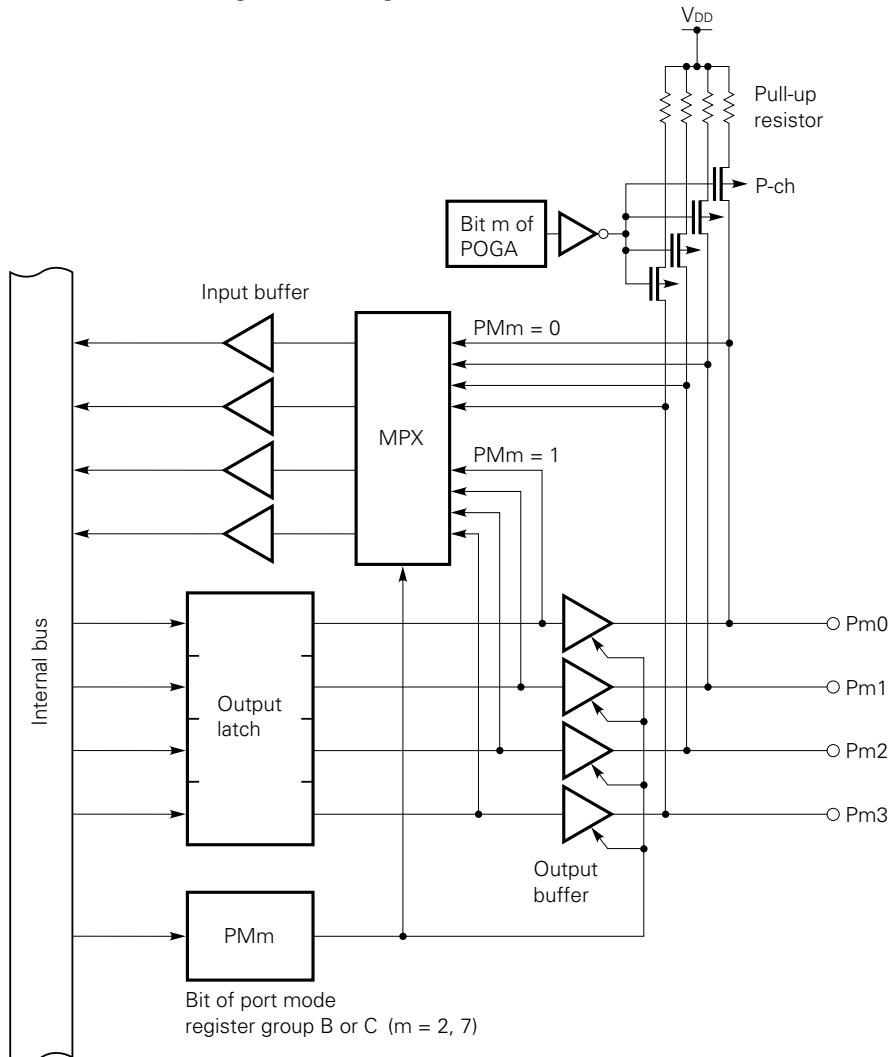
**Fig. 4-5   Configuration of Ports 4, 5, 12, 13, and 14**



Bit of port mode register group B (m = 4, 5):     Ports 4 and 5
Bit of port mode register group C (m = 4, 5, 6):  Ports 12, 13, and 14
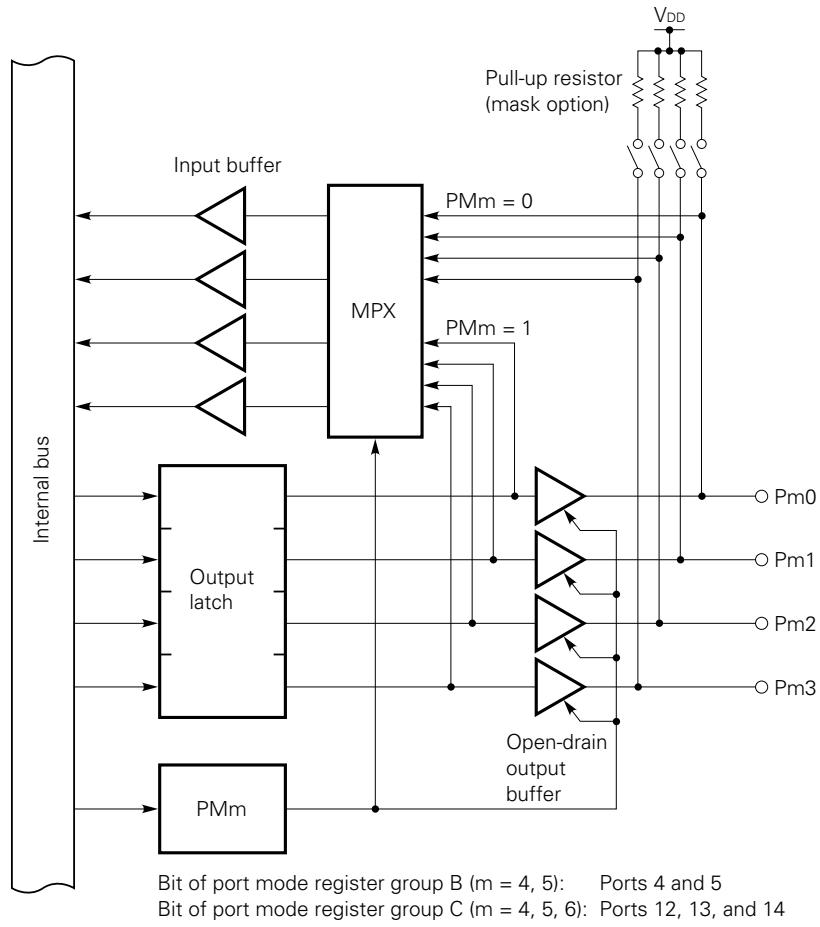
**Fig. 4-6   Configuration of Port 9**

**Fig. 4-7 Configuration of Ports 10 and 11**
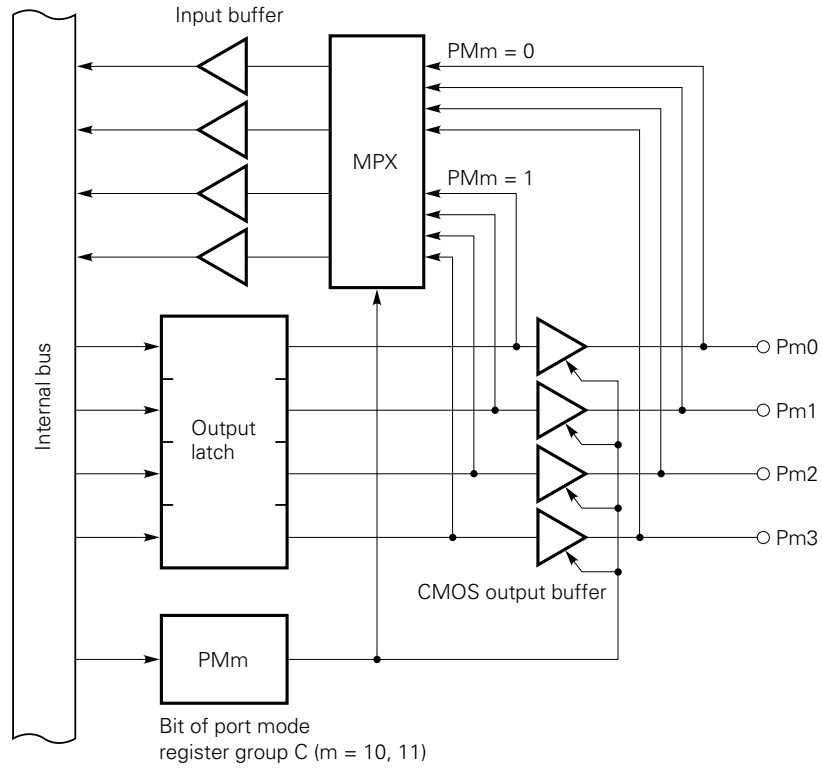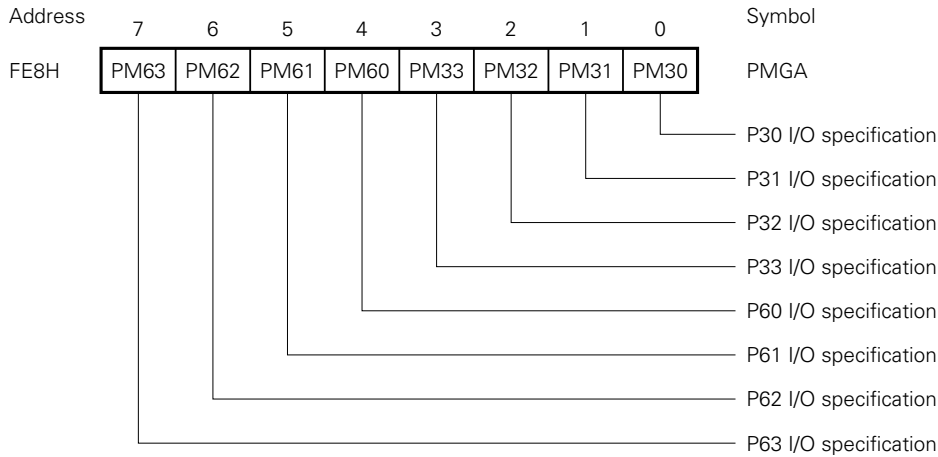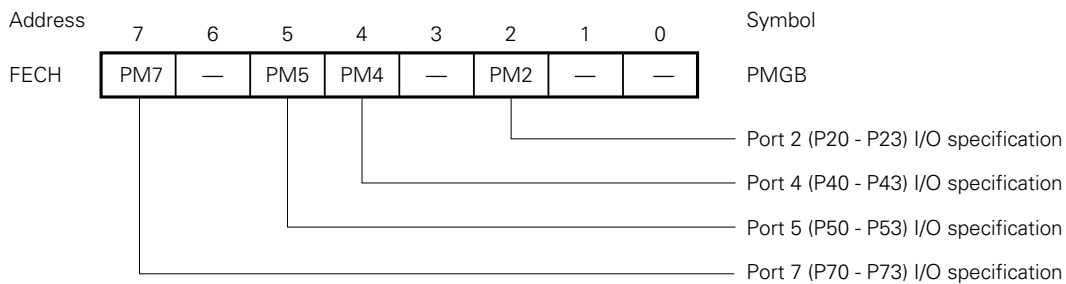


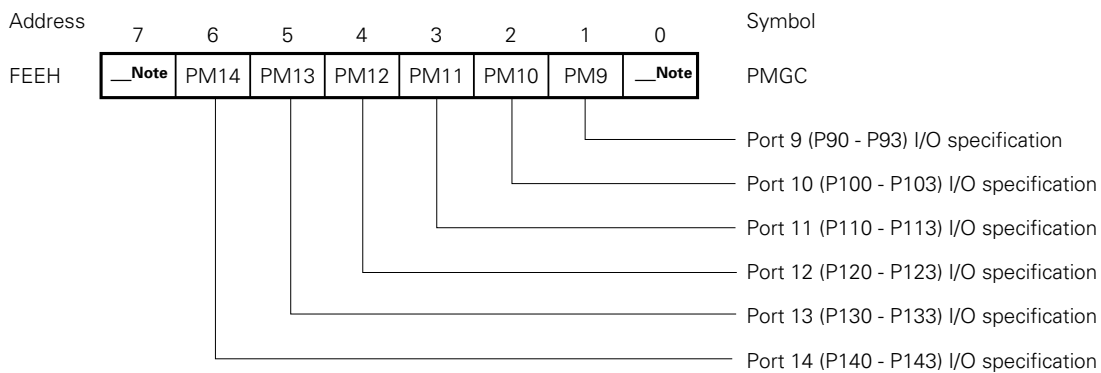**Fig. 4-8 Configuration of Port 15**
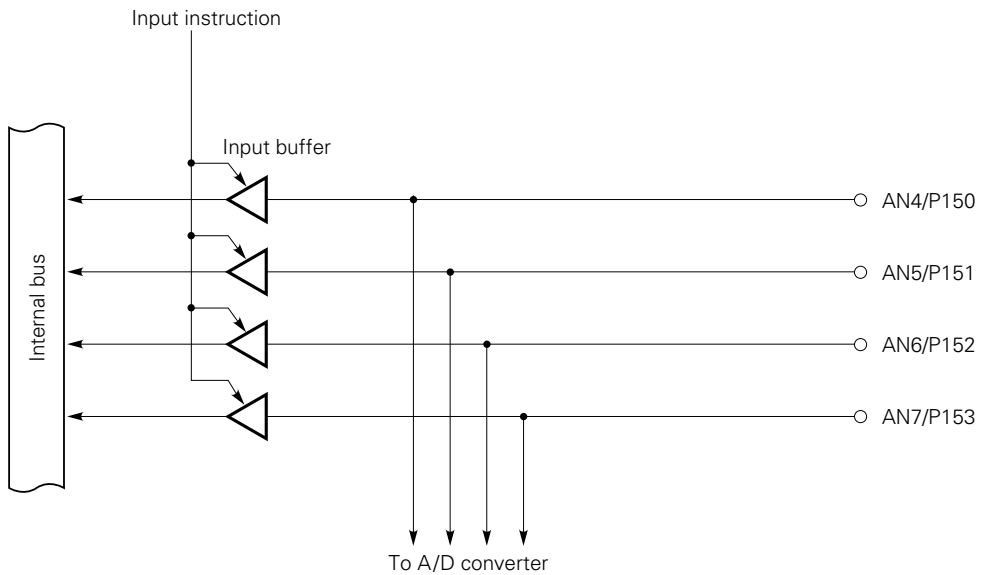
**Fig. 4-9   Formats of Port Mode Registers**

**(a)  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 |

- P30 I/O specification
- P31 I/O specification
- P32 I/O specification
- P33 I/O specification
- P60 I/O specification
- P61 I/O specification
- P62 I/O specification
- P63 I/O specification

**(b)  Port mode register group B**

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

- Port 2 (P20 - P23) I/O specification
- Port 4 (P40 - P43) I/O specification
- Port 5 (P50 - P53) I/O specification
- Port 7 (P70 - P73) I/O specification

**(c)  Port mode register group C**

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FEEH | —Note | PM14 | PM13 | PM12 | PM11 | PM10 | PM9 | —Note | PMGC |

- Port 9 (P90 - P93) I/O specification
- Port 10 (P100 - P103) I/O specification
- Port 11 (P110 - P113) I/O specification
- Port 12 (P120 - P123) I/O specification
- Port 13 (P130 - P133) I/O specification
- Port 14 (P140 - P143) I/O specification

| | Contents of specification |
|---|---|
| 0 | Input mode (Output buffer off) |
| 1 | Output mode (Output buffer on) |

**Note** To develop a program, these bits must be set to 0.  They correspond to PM8 and PM15.  While this chip is an input-only device, an emulator has I/O ports.

**(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 4-3.

**Table 4-3   I/O Port Operations by I/O Instructions**

| | Input mode $\left[\begin{array}{l}\text{corresponding bit in}\\\text{the mode register is 0}\end{array}\right]$ <br> [Output buffer is off] | Output mode $\left[\begin{array}{l}\text{corresponding bit in}\\\text{the mode register is 1}\end{array}\right]$ <br> [Output buffer is on] |
|---|---|---|
| When a 1-bit test instruction, 4-, or 8-bit 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** Instruction such as SET1 PORTn.bit or CLR1 PORTn.bit

**(4) Use of pull-up and pull-down resistors**

Ports 0 (excluding pin P00/INT4), 1 to 3, 6, and 7 can be provided with pull-up resistors by software. Ports 4, 5, and 12 to 14 can be provided with pull-up resistors by mask options. Port 9 can also be provided with pull-down resistors by mask options.

**Table 4-4   Specifying the Use of Pull-Up and Pull-Down Resistors**

| Port (pin name) | Specifying the use of pull-up and pull-down resistor | Bit in POGA |
|---|---|---|
| Port 0 (P01-P03)**Note** | The use of pull-up resistors is specified in units of 3 bits by software. | bit 0 |
| Port 1 (P10-P13) | The use of pull-up resistors is specified in units of 4 bits by software. | bit 1 |
| Port 2 (P20-P23) | | bit 2 |
| Port 3 (P30-P33) | | bit 3 |
| Port 6 (P60-P63) | | bit 6 |
| Port 7 (P70-P73) | | bit 7 |
| Port 4 (P40-P43) | The use of pull-up resistors is specified in units of bits by mask options. | – |
| Port 5 (P50-P53) | | |
| Port 12 (P120-P123) | | |
| Port 13 (P130-P133) | | |
| Port 14 (P140-P143) | | |
| Port 9 (P90-P93) | The use of pull-down resistors is specified in units of bits by mask options. | – |

**Note** The P00 pin cannot be provided with a pull-up resistor.

**Fig. 4-10   Format of the Register Group A Specifying the Use of Pull-Up Resistors**
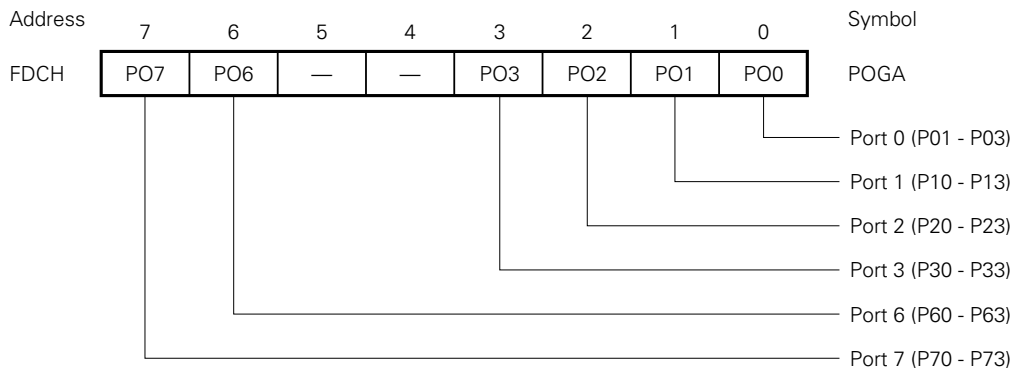


| | Description |
|---|---|
| 0 | No built-in pull-up resistor provided |
| 1 | Built-in pull-up resistor provided |

### 4.2 CLOCK GENERATOR

### (1) Configuration of the clock generator

The clock generator supplies various clock signals to the CPU and peripheral hardware. Fig. 4-11 shows the configuration of the clock generator.

### Fig. 4-11   Block Diagram of the Clock Generator
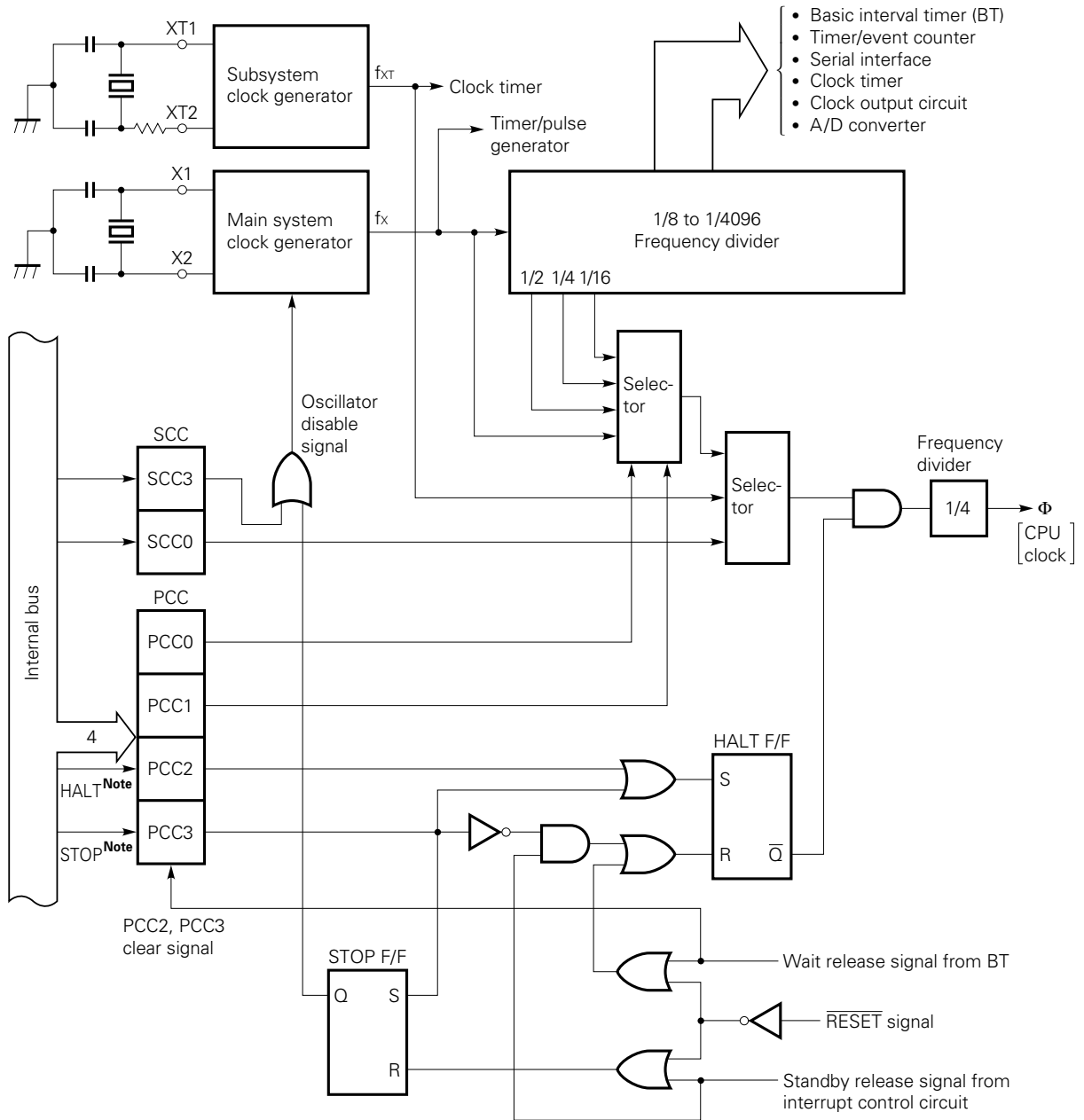


**Note** Instruction execution

**Remarks 1.** fx  : Main system clock frequency
   **2.** fxT : Subsystem clock frequency
   **3.** PCC: Processor clock control register
   **4.** SCC: System clock control register

**(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 $\Phi$
- 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) The generation of a $\overline{\text{RESET}}$ signal selects the lowest-speed mode[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[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 $\mu$s: at 32.768 kHz). In this case, the PCC set value does not affect the CPU clock signal.

(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 so that the watch and buzzer output functions can operate continuously even in a standby mode.

(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.** 10.7 $\mu$s (at 6.0 MHz) or 15.3 $\mu$s (at 4.19 MHz)
     **2.** 0.67 $\mu$s, 1.33 $\mu$s, 2.67 $\mu$s, 10.7 $\mu$s (at 6.0 MHz), or 0.95 $\mu$s, 1.91 $\mu$s, 3.82 $\mu$s, 15.3 $\mu$s (at 4.19 MHz)

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

The PCC is a 4-bit register for selecting a CPU clock with the low-order two bits and for selecting a CPU operation mode with the high-order two bits. (See **Fig. 4-12**.)

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.

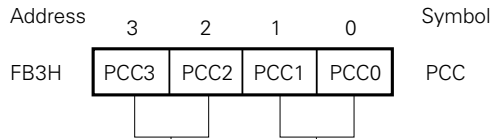The generation of a $\overline{\text{RESET}}$ signal clears the PCC to 0.

**Examples 1.** The machine cycle is set to 0.95 $\mu$s (at 4.19 MHz).

        SEL    MB15
        MOV  A, #0011B
        MOV  PCC, A

       **2.** The STOP mode is set. (A STOP instruction or HALT instruction must always be followed by an NOP instruction.)

        STOP
        NOP

**Fig. 4-12   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 | $\Phi = f_x/64$ (93.7 kHz) | 10.7 $\mu$s | $\Phi = f_{xT}/4$ (8.192 kHz) | 122 $\mu$s |
| 0 | 1 | $\Phi = f_x/16$ (375 kHz) | 2.67 $\mu$s | Not to be set | |
| 1 | 0 | $\Phi = f_x/8$ (750 kHz) | 1.33 $\mu$s | $\Phi = f_{xT}/4$ (8.192 kHz) | 122 $\mu$s |
| 1 | 1 | $\Phi = f_x/4$ (1.5 MHz) | 0.67 $\mu$s | | |

**(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 | $\Phi = f_x/64$ (65.5 kHz) | 15.3 $\mu$s | $\Phi = f_{xT}/4$ (8.192 kHz) | 122 $\mu$s |
| 0 | 1 | $\Phi = f_x/16$ (262 kHz) | 3.82 $\mu$s | Not to be set | |
| 1 | 0 | $\Phi = f_x/8$ (524 kHz) | 1.91 $\mu$s | $\Phi = f_{xT}/4$ (8.192 kHz) | 122 $\mu$s |
| 1 | 1 | $\Phi = f_x/4$ (1.05 MHz) | 0.95 $\mu$s | | |

**Remarks 1.**   $f_x$ : Output frequency from the main system clock oscillator

   **2.**   $f_{xT}$: Output frequency from the subsystem clock oscillator

**CPU operation mode control bits**

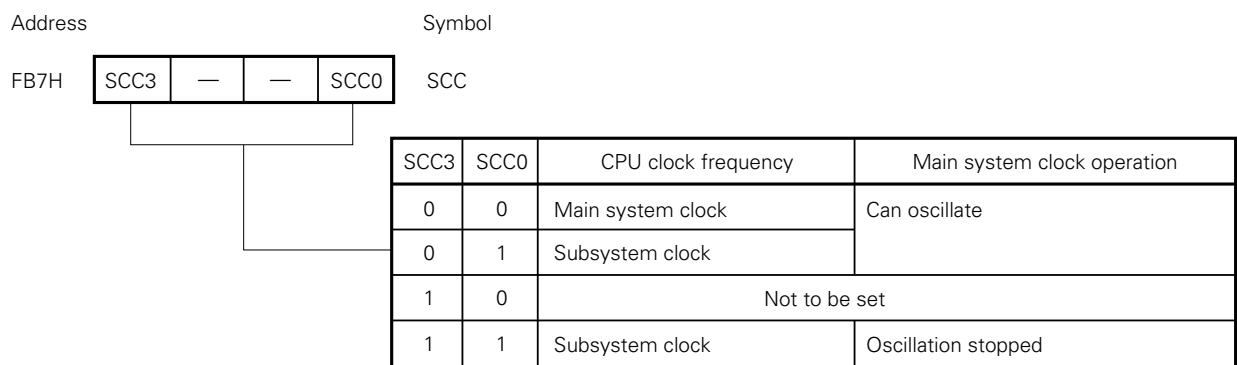| 0 | 0 | Normal operation mode |
|---|---|---|
| 0 | 1 | HALT mode |
| 1 | 0 | STOP mode |
| 1 | 1 | Not to be set |

**(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. 4-13**.)

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.

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

**Fig. 4-13   Format of the System Clock Control Register**

Address                                      Symbol

FB7H    | SCC3 | — | — | SCC0 |        SCC

| SCC3 | SCC0 | CPU clock frequency | 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 4-5 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 ($\Phi = f_X/16$), 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. 4-14   External Circuitry for the Main System Clock Oscillator**

**(a)  Crystal/ceramic oscillation**                     **(b)  External clock**
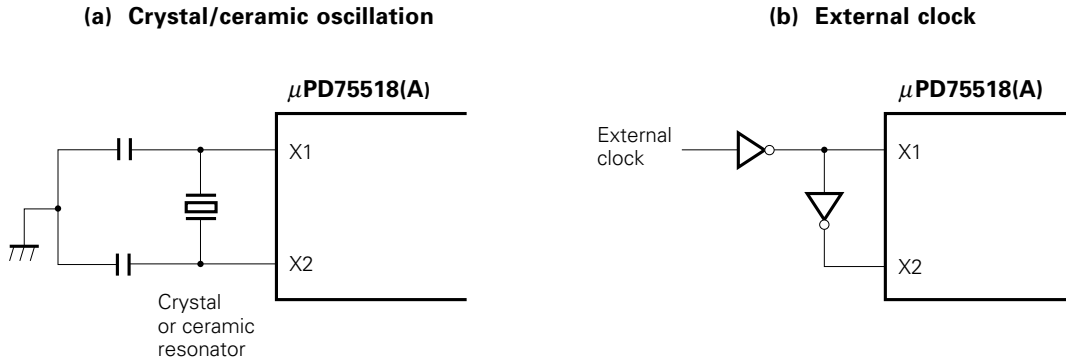


The subsystem clock oscillator operates with a crystal resonator (32.768 kHz standard) connected to the XT1 and XT2 pins.

An external clock can also be input.

**Fig. 4-15   External Circuitry for the Subsystem Clock Oscillator**

**(a)  Crystal oscillation**                               **(b)  External clock**
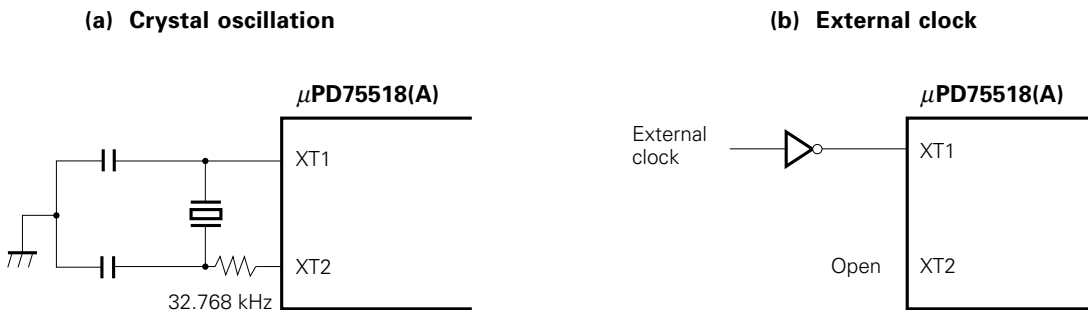


A caution on connecting the oscillator is described on the next page.

**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. 4-14 and 4-15 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 4-5 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 | | | $f_X/64f_{XT}$ 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 | | | $f_X/8f_{XT}$ machine cycles (23 machine cycles) | | |
| | 1 | 1 | 16 machine cycles | | | 16 machine cycles | | | 16 machine cycles | | | | | | $f_X/4f_{XT}$ machine cycles (46 machine cycles) | | |
| 1 | × | × | 1 machine cycle | | | Not to be set | | | 1 machine cycle | | | 1 machine cycle | | | | | |

**Remarks 1.** Time enclosed in parentheses is required when $f_X = 6.0$ MHz and $f_{XT} = 32.768$ kHz.

     **2.** ×: Don't care

     **3.** CPU clock $\Phi$ is supplied to the CPU. The reciprocal of this frequency is a minimum instruction time (defined as one machine cycle in this manual).

**Caution** When the four bits of PCC are set to 0001B ($\Phi = f_X/16$), 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. 4-16.

**Fig. 4-16  Changing the System Clock and CPU Clock**
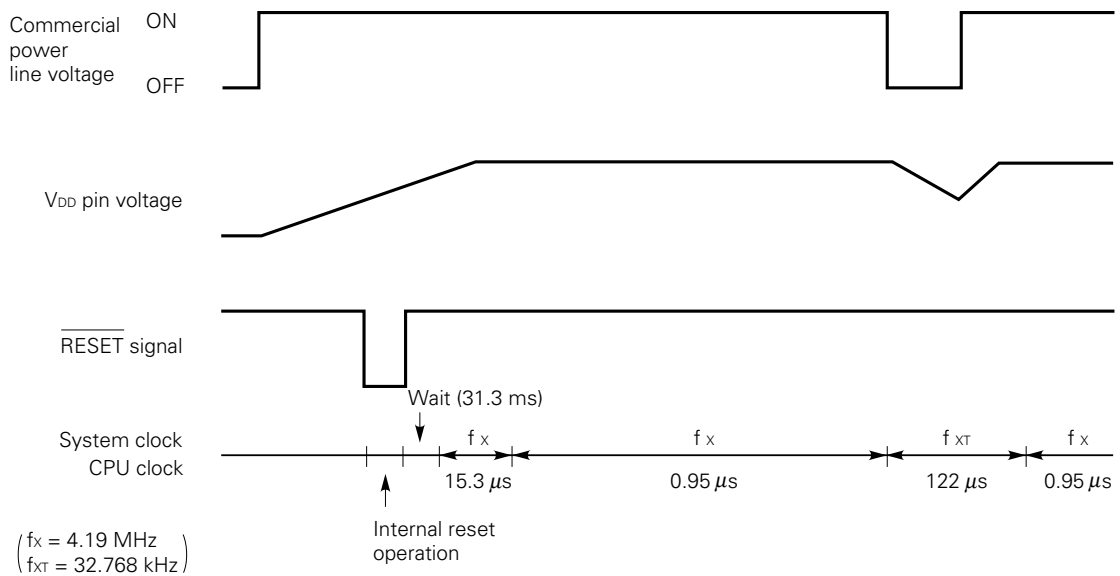


① The generation of a $\overline{\text{RESET}}$ signal starts CPU operation at the lowest speed of the main system clock[Note 1] after a wait time[Note 2] for stable oscillation.

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.** 10.7 $\mu$s (at 6.0 MHz) or 15.3 $\mu$s (at 4.19 MHz)
      **2.** 21.8 ms (at 6.0 MHz) or 31.3 ms (at 4.19 MHz)

### 4.3 CLOCK OUTPUT CIRCUIT

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

Fig. 4-17 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. 4-17   Configuration of the Clock Output Circuit**
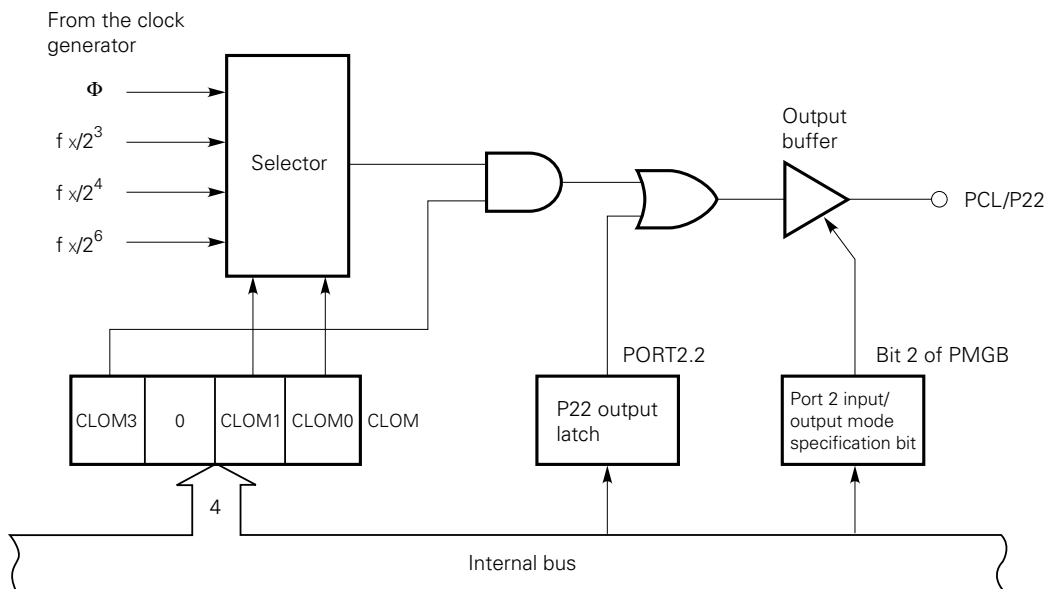


**Remark**  The clock output circuit is designed so that pulses with short widths do not appear in enabling or disabling clock output.

**(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.
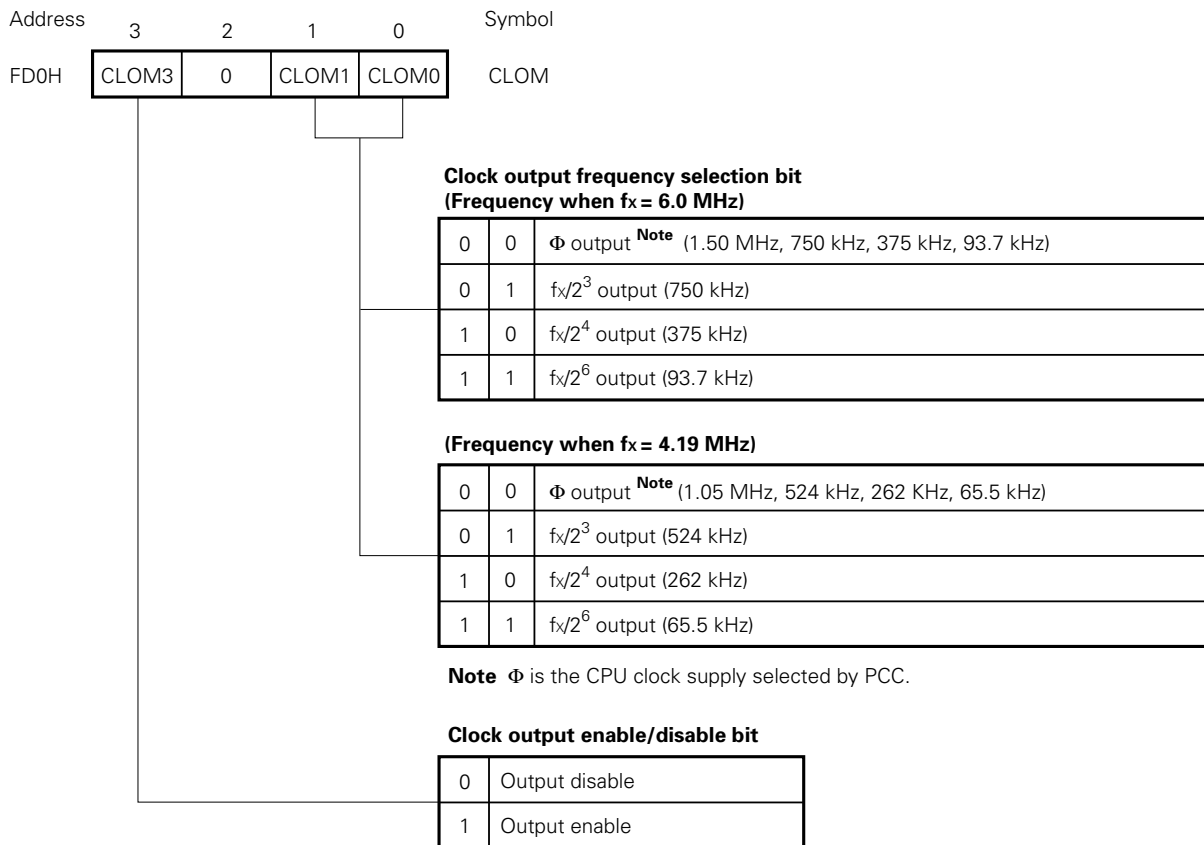
**Example**  CPU clock $\Phi$ is output on the PCL/P22 pin.

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

The generation of a $\overline{\text{RESET}}$ signal clears the CLOM to 0, disabling clock output.

**Fig. 4-18   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 | $\Phi$ output **Note** (1.50 MHz, 750 kHz, 375 kHz, 93.7 kHz) |
| 0 | 1 | $f_X/2^3$ output (750 kHz) |
| 1 | 0 | $f_X/2^4$ output (375 kHz) |
| 1 | 1 | $f_X/2^6$ output (93.7 kHz) |

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

| | | |
|---|---|---|
| 0 | 0 | $\Phi$ output **Note** (1.05 MHz, 524 kHz, 262 KHz, 65.5 kHz) |
| 0 | 1 | $f_X/2^3$ output (524 kHz) |
| 1 | 0 | $f_X/2^4$ output (262 kHz) |
| 1 | 1 | $f_X/2^6$ output (65.5 kHz) |

**Note**  $\Phi$ 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 μPD75518(A) 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. 4-19  Application to Remote Control Output**
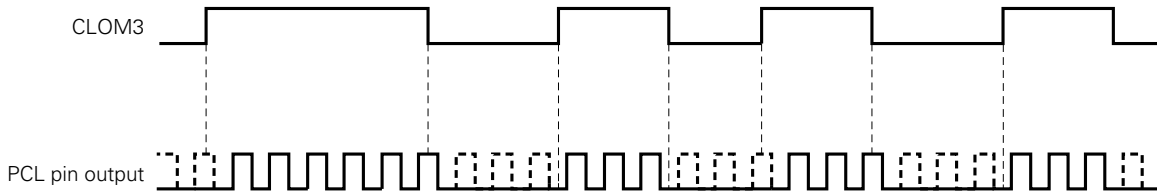
### 4.4 BASIC INTERVAL TIMER

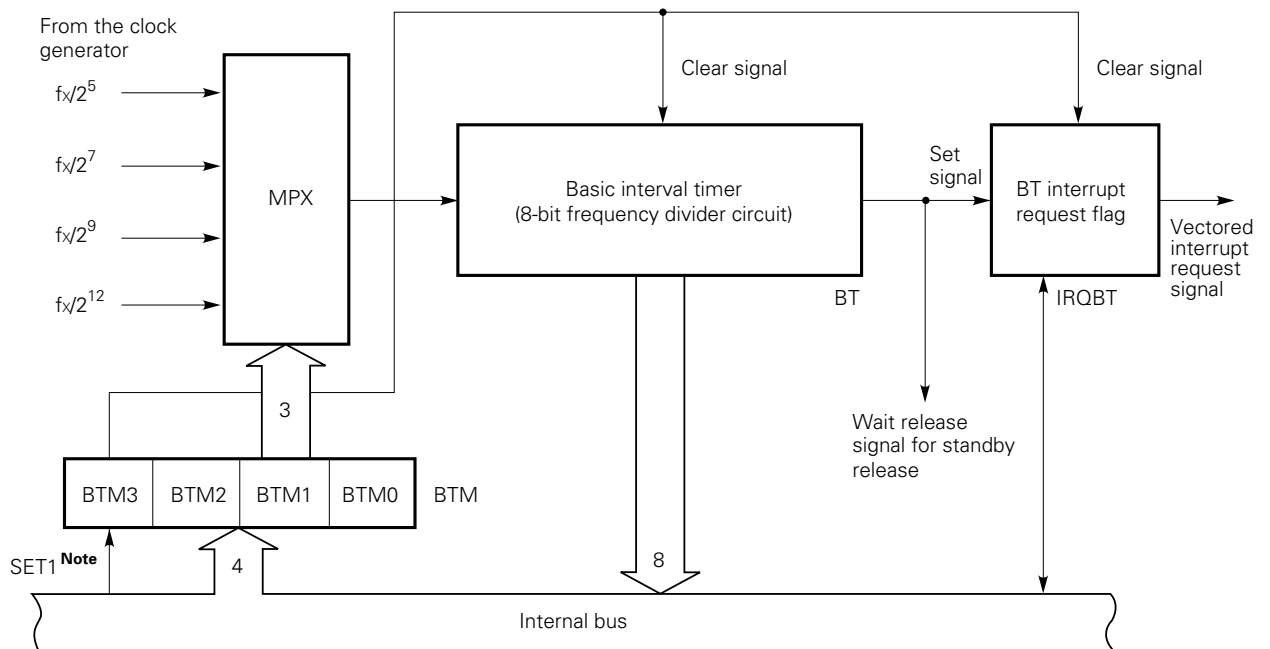**(1) Configuration of the basic interval timer**

Fig. 4-20 shows the configuration of the basic interval timer.

**(2) Basic interval timer functions**

The basic interval timer provides the following functions:

(a) Interval timer operation that generates a reference time interrupt
(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. 4-20   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).

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

**Examples 1.** Set the interrupt generation interval to 1.95 ms (4.19 MHz).

```
       SEL    MB15        ; or CLR1 MBE
       MOV    A, #1111B
       MOV    BTM, A      ; BTM ← 1111B
```
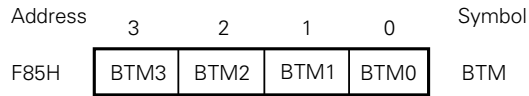
**2.** Clear BT and IRQBT (application of the watchdog timer)

```
       SEL    MB15        ; or CLR1 MBE
       SET1   BTM.3       ; Set bit 3 of BTM to 1
```

**Fig. 4-21   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. 4-21.**)

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.**

**Example**  Read the count value of BT.

```
          SET1  MBE
          SEL   MB15
          MOV   HL, #BT        ; Set the BT address in HL
LOOP:     MOV   XA, @HL        ; First read
          MOV   BC, XA
          MOV   XA, @HL        ; Second read
          SKE   XA, BC
          BR    LOOP
```

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 generation of a $\overline{\text{RESET}}$ signal 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. 4-21.  BTM must be set before the STOP mode is set.  (For details, see **Chapter 6**.)
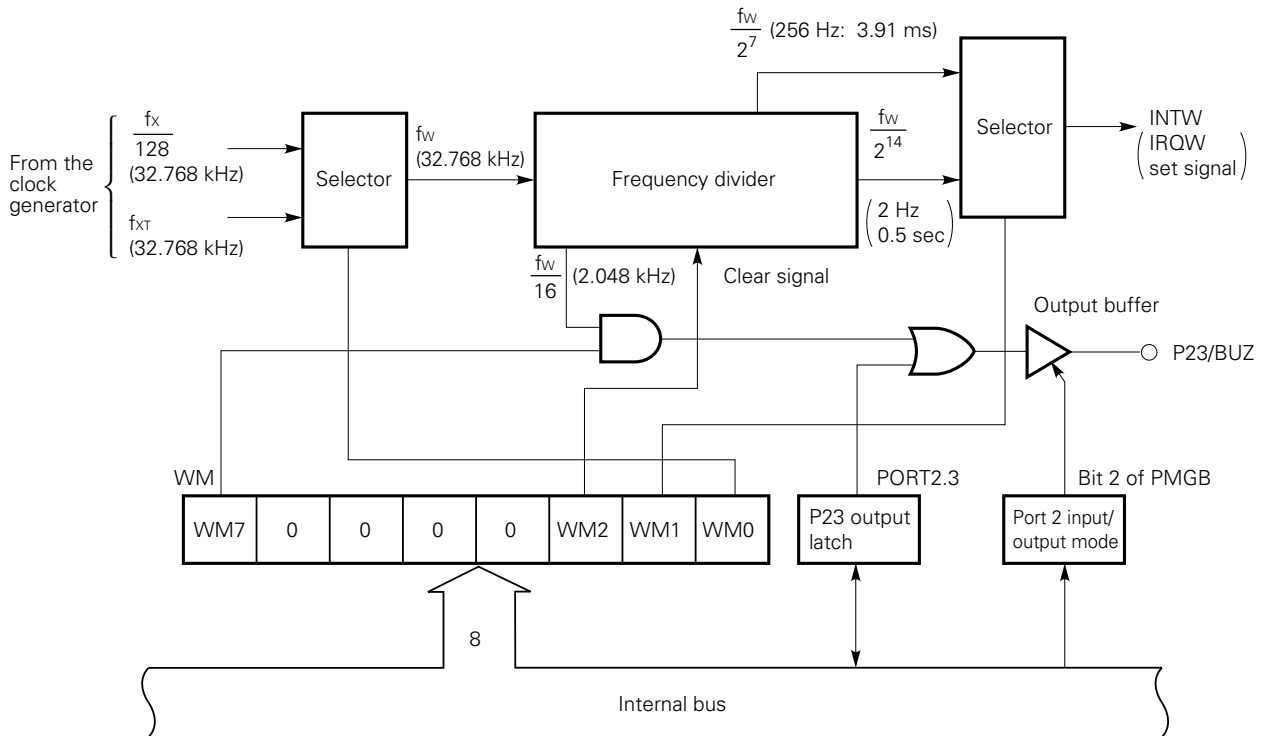
### 4.5 CLOCK TIMER

**(1) Clock timer**

The μPD75518(A) contains one channel for a clock timer.  Fig. 4-22 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 or subsystem clock 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 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.

**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.**

**Fig. 4-22    Block Diagram of the Clock Timer**



**Remark**  The values in parentheses indicates are for $f_X$ = 4.194304 MHz and $f_{XT}$ = 32.768 kHz

**(3) Clock mode register**

The clock 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. 4-23 shows the format.

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

**Example** Use the main system clock (4.19 MHz) for setting time, and enable buzzer output.

```
CLR1    MBE
MOV     XA, #84H
MOV     WM, XA      ; Set WM
```

**Fig. 4-23   Format of the Clock Mode Register**

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

**Count clock (fW) selection bit**

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

**Operation mode selection bit**

| | | |
|------|---|---|
| WM1 | 0 | Normal clock mode ( $\dfrac{f_W}{2^{14}}$ : sets IRQW at 0.5 s) |
| | 1 | Advanced clock mode ( $\dfrac{f_W}{2^{7}}$ : 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 enable/disable bit**

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

### 4.6 TIMER/EVENT COUNTER

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

The µPD75518(A) contains one channel of timer/event counter, which is configured as shown in Fig. 4-24.

**(2) 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

**(3) 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. Fig. 4-25 shows its format.

An 8-bit memory manipulation instruction is used to set 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.

**Examples 1.** The timer is started in the interval timer mode with CP = 4.09 kHz.

```
        SEL    MB15              ; or CLR1 MBE
        MOV    XA, #01001100B
        MOV    TM0, XA           ; TM0 ← 4CH
```

**2.** The timer is restarted according to the setting of the timer/event counter mode register.

```
        SEL    MB15              ; or CLR1 MBE
        SET1   TM0.3             ; TM0.bit3 ← 1
```

The generation of a $\overline{\text{RESET}}$ signal clears all bits to 0.
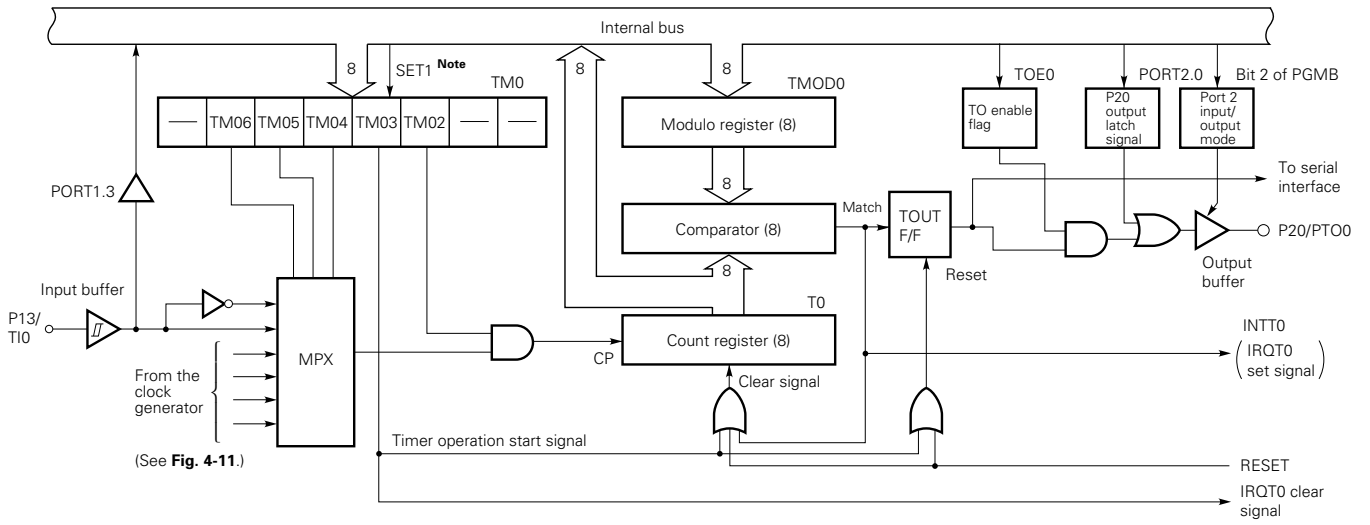
The timer/event counter output enable flag (TOE0) enables or disables output of the timer out F/F (TOUT F/F) status to the PTO0 pin.

The timer out F/F (TOUT F/F) is inverted by a match signal transmitted from the comparator.

The timer out F/F is reset when an instruction sets bit 3 of the timer mode register (TM0).

The generation of a $\overline{\text{RESET}}$ signal clears the TOE0 and TOUT F/F to 0.

**Fig. 4-24  Block Diagram of the Timer/Event Counter**



**Note**  Instruction execution

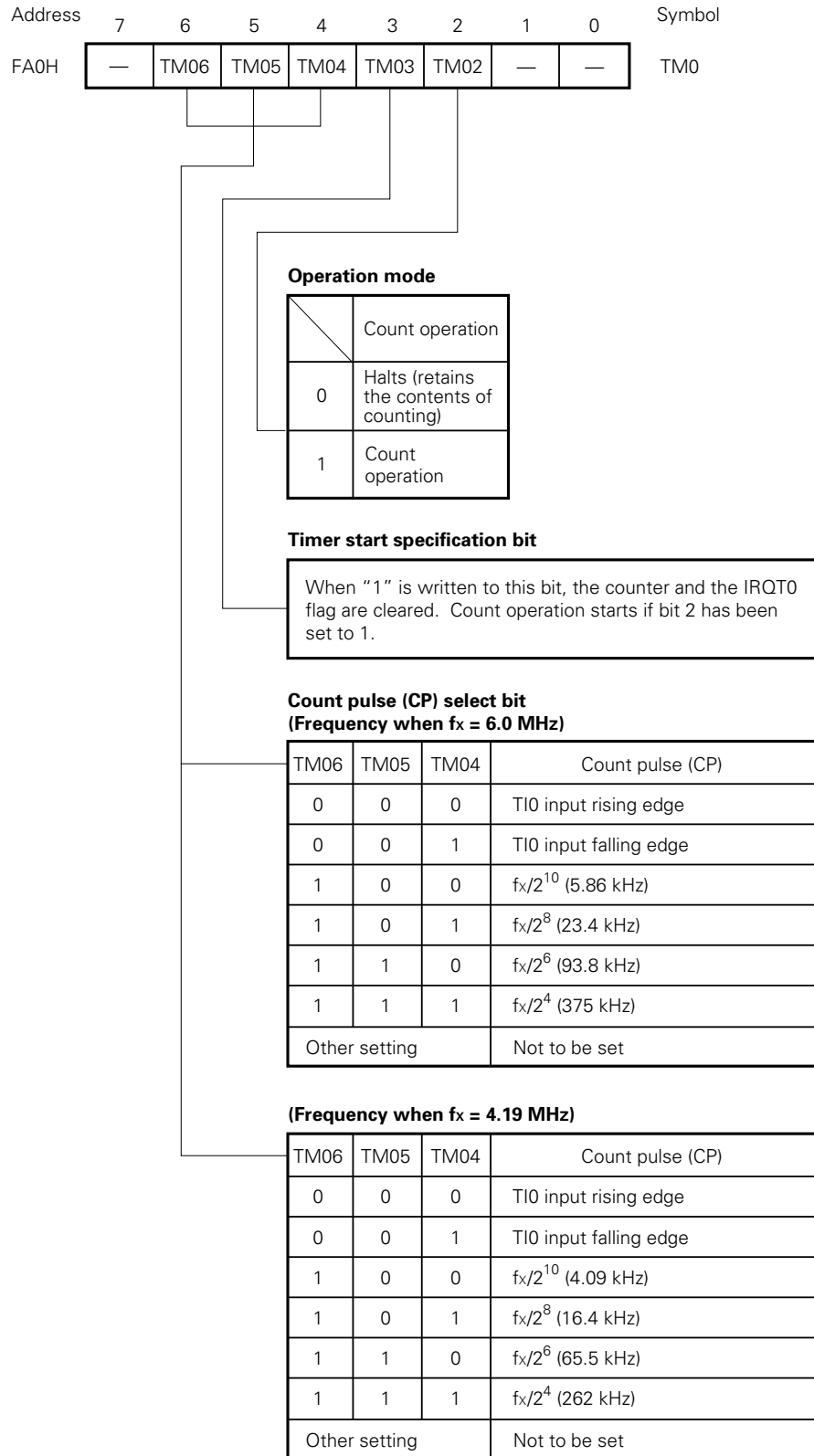**Fig. 4-25   Format of the Timer/Event Counter Mode Register**

| 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 |

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

Address

             3

FA2H    | TOE0 |

Timer/event counter output enable flag (W)

| 0 | Disables |
|---|----------|
| 1 | Enables  |
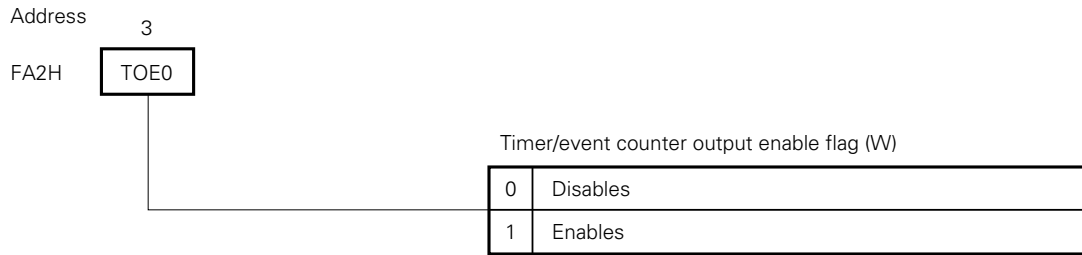
**(4)　Operation mode of the timer/event counter**

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:

① P13/TI0 pin signal input and test

② Output of the timer out F/F status to the PTO0

③ Setting of the modulo register (TMOD0)

④ Reading from the count register (T0)

⑤ 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.

**Caution   A value other than 0 must be set in the modulo register.**

**Example**   The value 3FH is set in the modulo register of channel 0.

          SEL         MB15              ; or CLR1 MBE

          MOV       XA, #3FH

          MOV       TMOD0, XA

If the value set in the modulo register matches the contents of the count register, the match signal is generated. Then, the TOUT F/F is inverted, and the counter register is cleared.

The interval time of the generation of the match signal is calculated as follows:

(Set value of the modulo register + 1) $\times$ resolution

The resolution is 1/count pulse frequency.

Table 4-6 indicates the resolution and maximum set time (when FFH is set in the modulo register), depending on a selected count pulse.

**Table 4-6  Resolution and Maximum Set Time**

**(When f$_X$ = 6.0 MHz)**

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

**(When f$_X$ = 4.19 MHz)**

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

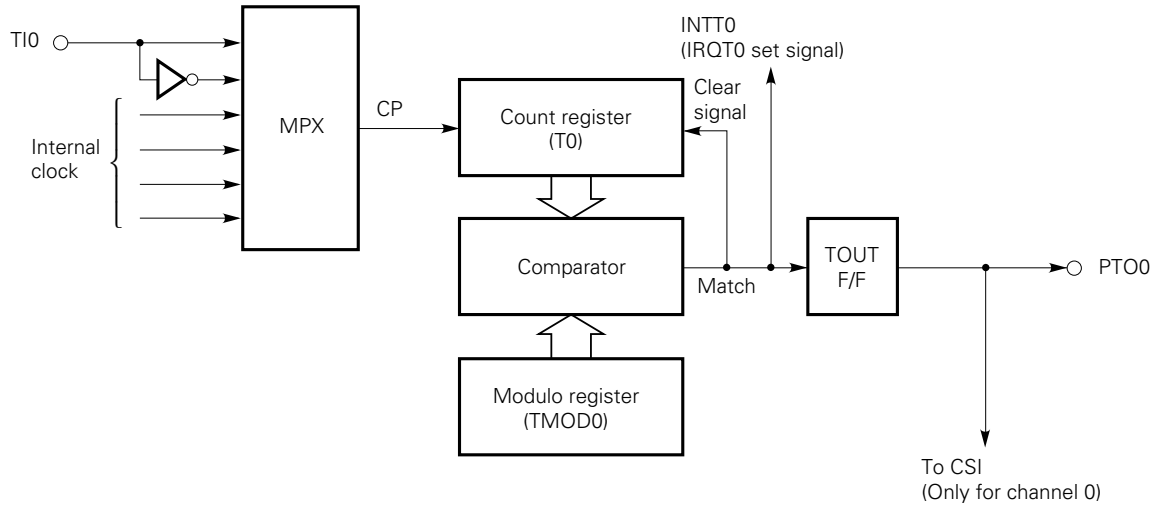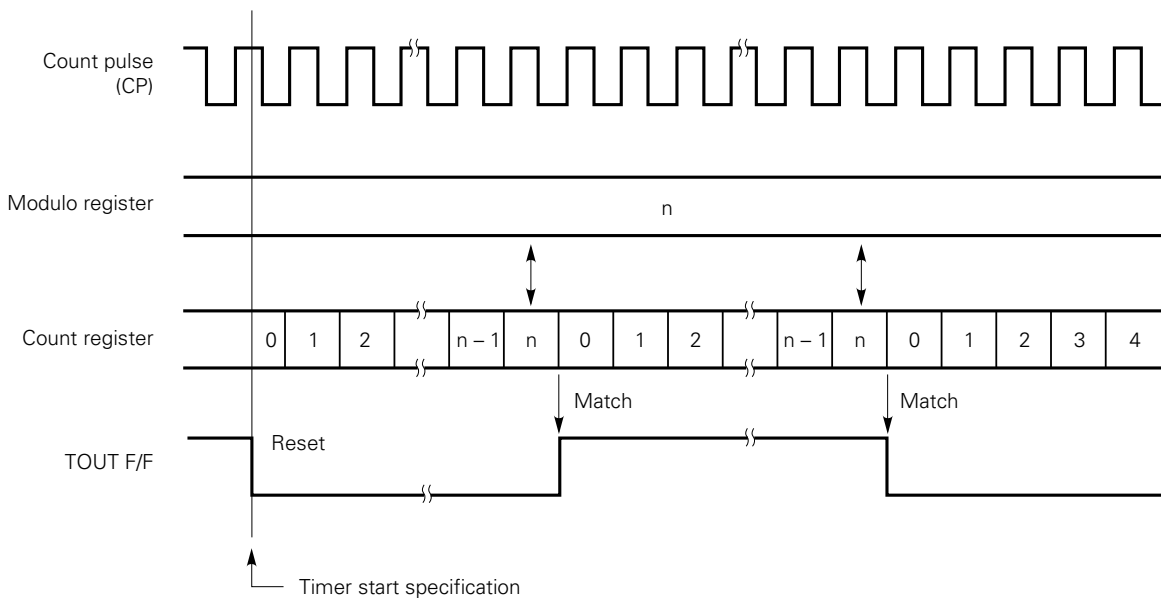**Fig. 4-27   Operation in the Count Operation Mode**



**Fig. 4-28   Timing of Count Operation**

### 4.7 TIMER/PULSE GENERATOR

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

The μPD75518(A) 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 ($2^{15}/f_X$)**Note**

    **Note**   $2^{15}/f_X$ = 5.46 ms (at 6.0 MHz) or 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. 4-29 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.

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

**Fig. 4-29   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. 4-30 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 changed from 0 to 1.

Fig. 4-31 shows the operation timing for the MODH setting, and Table 4-7 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. 4-30   Block Diagram of the Timer/Pulse Generator (Timer Mode)**
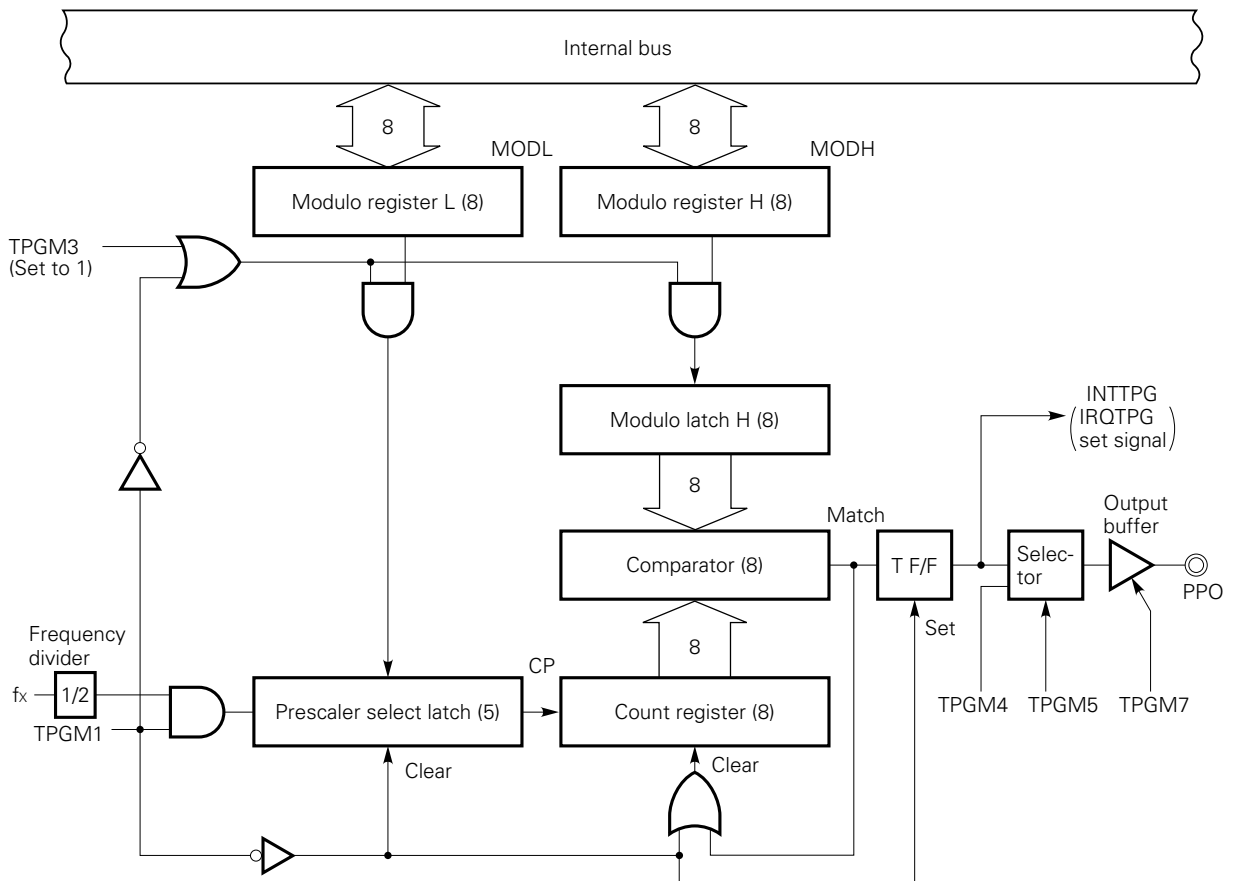
**Example**    Set IRQTPG every 1.95 ms, and set the output high on the PPO pin.

```
CLR1    MBE                 ; or SEL MB15
MOV     XA, #00100000B
MOV     MODL, XA
MOV     XA, #0FFH
MOV     MODH, XA
MOV     XA, #10011011B
MOV     TPGM, XA            ; Timer start, PPO ← 1
```

**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.**

**Example**    
```
DI
CLR1    MBE
MOV     XA, #0
MOV     TPGM, XA
CLR1    IRQTPG
EI
```

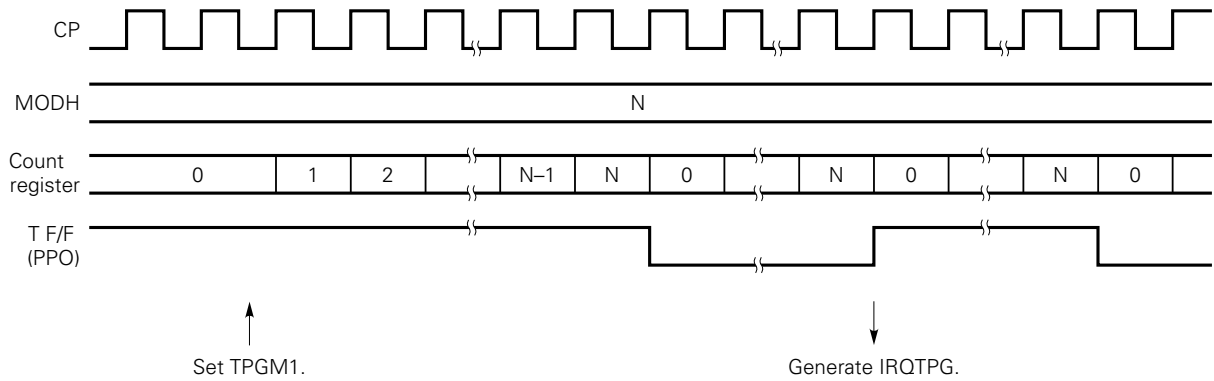**Fig. 4-31 Timer Mode Operation Timing**



**Table 4-7 Modulo Register Settings**

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

| MODL bits 2-6 | | | | | Interrupt generation interval | Square wave output frequency |
|---|---|---|---|---|---|---|
| 6 | 5 | 4 | 3 | 2 | ($f_X$ = 6.0 MHz) | ($f_X$ = 6.0 MHz) |
| 0 | 0 | 0 | 0 | 1 | 256 (N+1)/$f_X$ = 85.3 $\mu$s - 10.9 ms | $f_X$/256 (N+1) = 91.6 Hz - 11.7 kHz |
| 0 | 0 | 0 | 1 | 0 | 128 (N+1)/$f_X$ = 42.7 $\mu$s - 5.46 ms | $f_X$/128 (N+1) = 183 Hz - 23.4 kHz |
| 0 | 0 | 1 | 0 | 0 | 64 (N+1)/$f_X$ = 21.3 $\mu$s - 2.73 ms | $f_X$/64 (N+1) = 366 Hz - 46.9 kHz |
| 0 | 1 | 0 | 0 | 0 | 32 (N+1)/$f_X$ = 10.7 $\mu$s - 1.37 ms | $f_X$/32 (N+1) = 732 Hz - 93.8 kHz |
| 1 | 0 | 0 | 0 | 0 | 16 (N+1)/$f_X$ = 5.33 $\mu$s - 683 $\mu$s | $f_X$/16 (N+1) = 1465 Hz - 188 kHz |

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

| MODL bits 2-6 | | | | | Interrupt generation interval | Square wave output frequency |
|---|---|---|---|---|---|---|
| 6 | 5 | 4 | 3 | 2 | ($f_X$ = 4.19 MHz) | ($f_X$ = 4.19 MHz) |
| 0 | 0 | 0 | 0 | 1 | 256 (N+1)/$f_X$ = 122 $\mu$s - 15.6 ms | $f_X$/256 (N+1) =64 Hz - 8 kHz |
| 0 | 0 | 0 | 1 | 0 | 128 (N+1)/$f_X$ = 61.0 $\mu$s - 7.81 ms | $f_X$/128 (N+1) = 128 Hz - 16 kHz |
| 0 | 0 | 1 | 0 | 0 | 64 (N+1)/$f_X$ = 30.5 $\mu$s - 3.91 ms | $f_X$/64 (N+1) = 256 Hz - 32 kHz |
| 0 | 1 | 0 | 0 | 0 | 32 (N+1)/$f_X$ = 15.3 $\mu$s - 1.95 ms | $f_X$/32 (N+1) = 512 Hz - 65 kHz |
| 1 | 0 | 0 | 0 | 0 | 16 (N+1)/$f_X$ = 7.63 $\mu$s - 977 $\mu$s | $f_X$/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. 4-32 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 or $2^{15}$/fx = 7.81 ms: At 4.19 MHz). PWM pulses output by the $\mu$PD75518(A) 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. 4-33**.)

The PWM pulse signal is generated by combining the basic period determined by $2^{10}$/fx and the secondary period by $2^{15}$/fx so that the time constant of the external low-pass filter can be decreased.

Table 4-8 lists the basic and secondary periods by oscillator frequency.

**Table 4-8   Basic and Secondary Periods**

|  | fx = 6.0 MHz | fx = 4.19 MHz |
|---|---|---|
| Basic period ($2^{10}$/fx) | 171 $\mu$s | 244 $\mu$s |
| Secondary period ($2^{15}$/fx) | 5.46 ms | 7.81 ms |

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 transferred from the 8 bits of MODH, and the lower 6 bits of the latch are transferred 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 $\mu$PD75518(A) 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 is referred to as reloading, and it is controlled by TPGM3. If TPGM3 is 0, reloading is disabled, and if it is 1, reloading is enabled. Follow the procedure below to rewrite the modulo latch contents:

  (i)   Clear TPGM3 to disable reloading.
  (ii)   Change the MODH and MODL contents.
  (iii)   Set TPGM3 to enable reloading.

**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) is 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 $\mu$s: at 6.0 MHz or 244 $\mu$s: at 4.19 MHz), PWM pulses do not change.**

**Example**   Decrease analog output voltage to the lowest level, then increase it to the highest level.

```
        CLR1    MBE
        MOV     XA, #01H
        MOV     MODH, XA          ; MODH ← 01
        MOV     XA, #00H
        MOV     MODL, XA          ; MODL ← 00
        MOV     XA, #10101010B
        MOV     TPGM, XA          ; Enable PWM pulse output
                .
                .
                .
                .
                .
                .
                .
        CLR1    TPGM.3            ; Disable reloading
        MOV     XA, #0FFH
        MOV     MODH, XA
        MOV     XA, #0FCH
        MOV     MODL, XA
        SET1    TPGM.3            ; Enable reloading
```
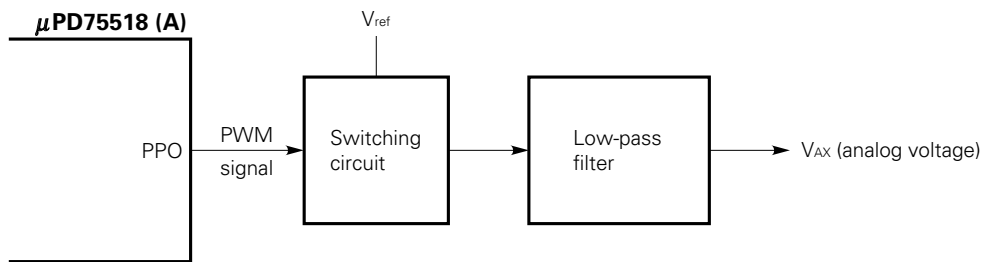
**(5) Static output to the PPO pin**

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. 4-32   Block Diagram of the Timer/Pulse Generator (PWM Pulse Generation Mode)
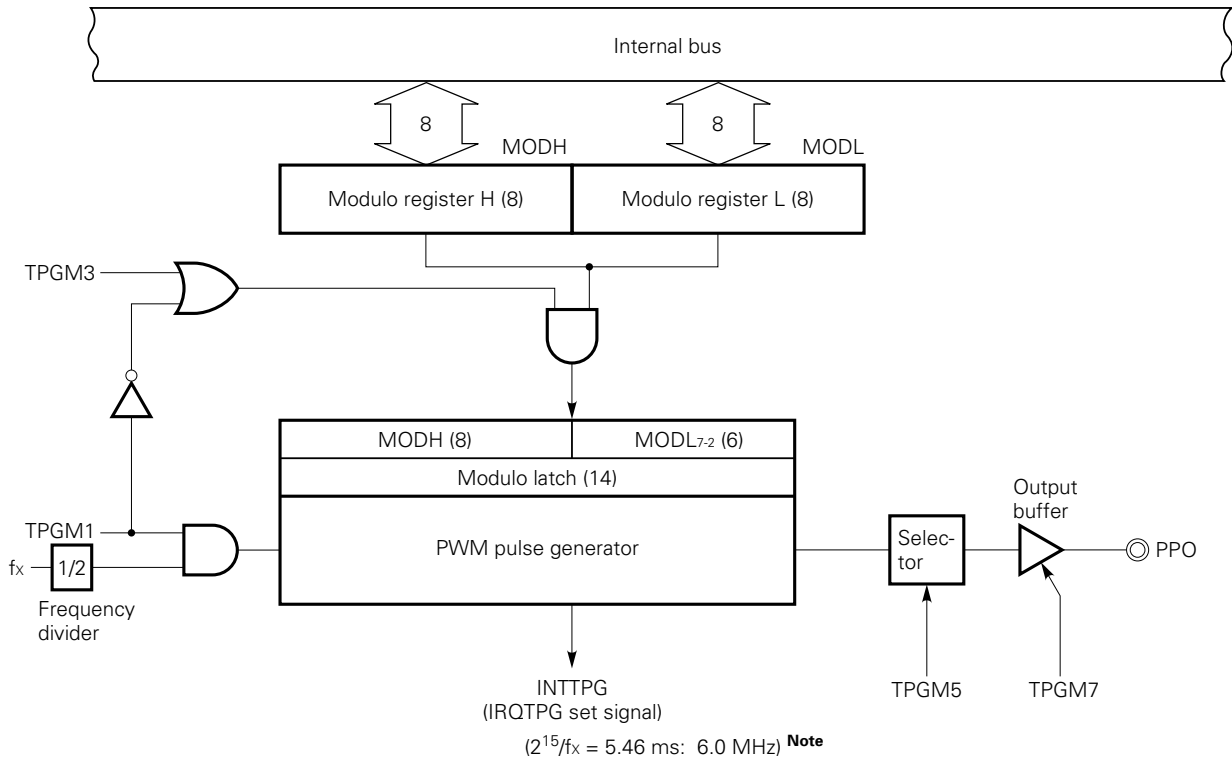


**Note**   At 4.19 MHz:  $2^{15}/f_X = 7.81$ ms

Fig. 4-33   Sample Configuration of D/A Conversion Using $\mu$PD75518(A)

**4.8 SERIAL INTERFACE (CHANNEL 0)**

The $\mu$PD75518(A) has two channels of serial interface: Channel 0 and channel 1. Table 4-9 lists the differences between channel 0 and channel 1.

**Table 4-9 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 (SBI) | | | |

### 4.8.1  Serial Interface (Channel 0) Functions

The clock synchronous 8-bit serial interface is contained in the $\mu$PD75518(A) and has four 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 ($\overline{\text{SCK0}}$), 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 ($\overline{\text{SCK0}}$) 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 $\overline{\text{SCK0}}$ 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 ($\overline{\text{SCK0}}$) and serial data bus (SB0 or SB1).

This mode conforms to the NEC serial bus format.

In this mode, the transmitter 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.


### 4.8.2  Configuration of Serial Interface (Channel 0)

Fig. 4-34 shows the block diagram of the serial interface (channel 0).

**Fig. 4-34   Block diagram of the Serial Interface (Channel 0)**
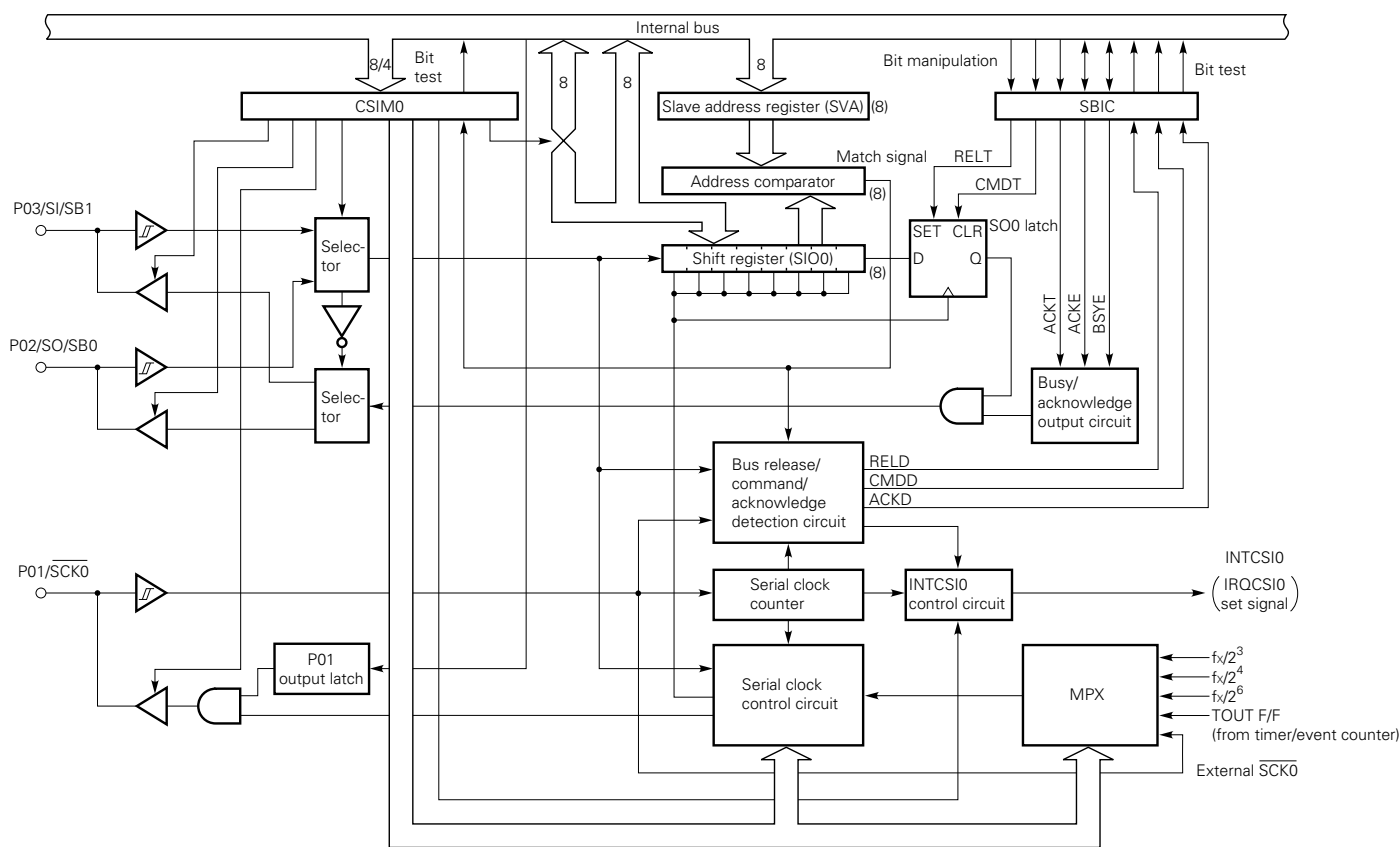
### 4.8.3 Register Functions

**(1) Serial operation mode register 0 (CSIM0)**

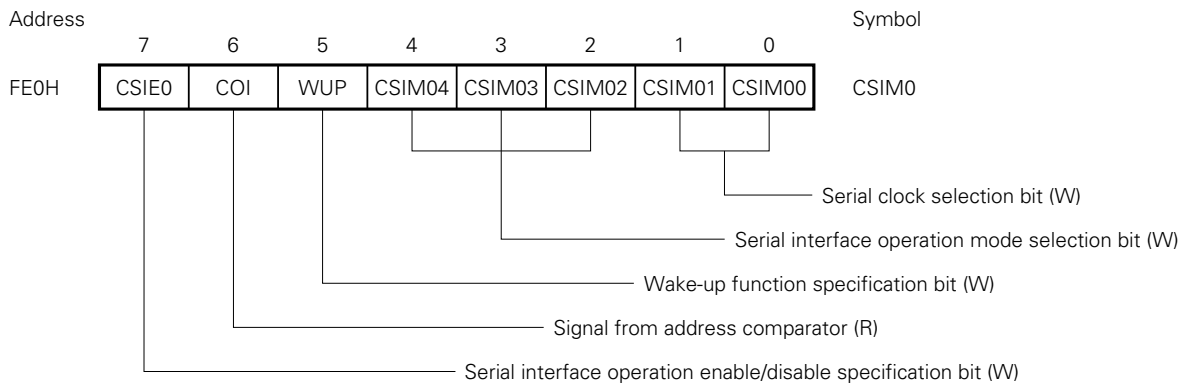Fig. 4-35 shows the format of serial operation mode register 0 (CSIM0).

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. 4-35**.) Bit 6 allows bit test operation only; any data written to this bit is invalid.

When the $\overline{\text{RESET}}$ signal is input, this register is set to 00H.

**Fig. 4-35   Format of Serial Operation Mode Register 0 (CSIM0) (1/3)**



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

**Fig. 4-35   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 | Timer/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 $f_X$ = 4.19 MHz or 6.0 MHz.

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

| CSIM04 | CSIM03 | CSIM02 | Operation mode | Shift register bit sequence | 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. 4-35  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 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 | 1 | 0 | High level output |
| 0 | 0 | 1 | |
| 0 | 1 | 1 | |
| 1 | 1 | 0 | Serial clock output (High level output) |
| 1 | 0 | 1 | |
| 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
```

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

Fig. 4-36 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. 4-36**.)

When the $\overline{\text{RESET}}$ signal is input, this register is set to 00H.

**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. 4-36   Format of Serial Bus Interface Control Register (SBIC) (1/3)**
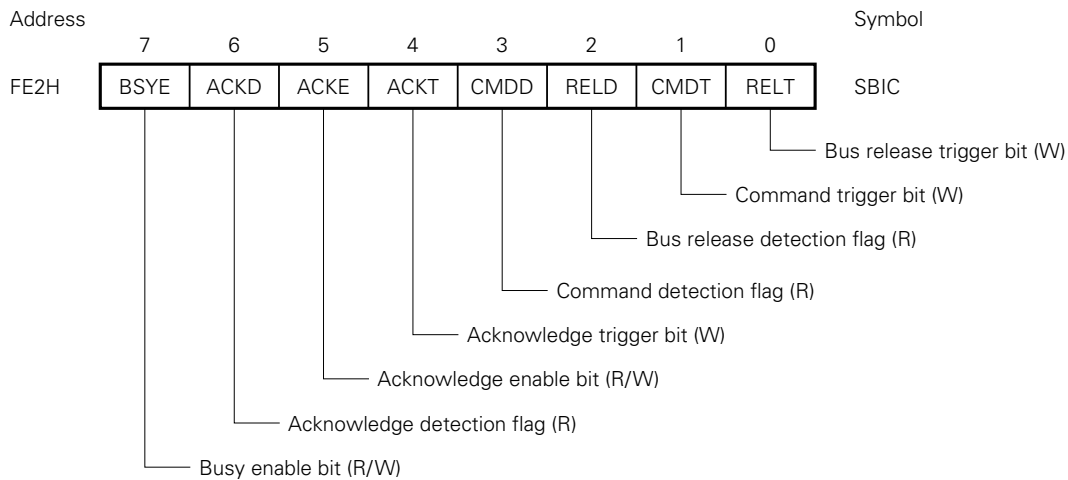


**Remark**   (R)   : Read only
   (W)   : Write only
   (R/W): Read/write

<div align="center"><strong>Fig. 4-36   Format of Serial Bus Interface Control Register (SBIC) (2/3)</strong></div>

**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)**

| RELD | Condition for being cleared (RELD = 0) | Condition for being set (RELD = 1) |
|------|----------------------------------------|-------------------------------------|
| | ① The transfer start instruction is executed<br>② The $\overline{\text{RESET}}$ signal is entered.<br>③ CSIE0 = 0 (See **Fig. 4-35**.)<br>④ SVA does not match SIO0 when an address is received. | The bus release signal (REL) is detected. |

**Command detection flag (R)**

| CMDD | Condition for being cleared (CMDD = 0) | Condition for being set (CMDD = 1) |
|------|----------------------------------------|-------------------------------------|
| | 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. 4-35**.) | 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. 4-36  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
```

**(3) Shift register (SIO0)**

Fig. 4-37 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 tansmission, data written to SIO0 is output on the serial output (SO0) or serial data bus (SB0/SB1). In reception, 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 transmission or reception is completed.

**Fig. 4-37   Peripheral Hardware of Shift Register 0**
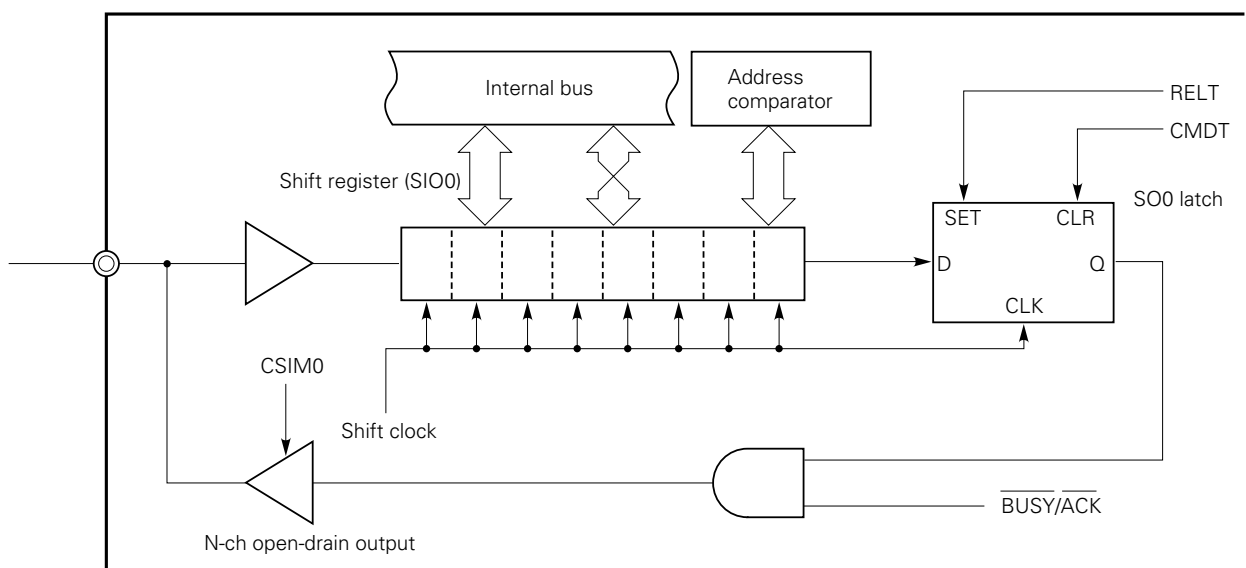


The timing for reading SIO0 and start of serial transfer (writing to SIO0) is as follows:

• 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 0 is excluded.
• When the serial clock is masked after 8-bit serial transfer
• $\overline{\text{SCK0}}$ is high.

**(4) 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 the $\overline{\text{RESET}}$ signal is entered, the value of SVA is undefined. However, the value of SVA is preserved when the $\overline{\text{RESET}}$ signal is entered in the standby mode.

- **Slave address detection**
  **[In the SBI mode]**

  SVA is used when the µPD75518(A) 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 µPD75518(A) operating as the master
  - When data is transferred with the µPD75518(A) operating as a slave

**4.8.4 Signals**

Table 4-10 lists signals. Fig. 4-38 to 4-43 show operations of signals and flags.

**Table 4-10   Various Signals (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 | $\overline{SCK0}$ "H"  SB0 (SB1) | • RELT is set. | • RELD is set.<br>• CMDD is cleared. | Indicates that CMD signal follows and data transmitted is address data. |
| Command signal (CMD) | Master | Falling edge of SB0 (SB1) when $\overline{SCK0}$ = 1 | $\overline{SCK0}$ "H"  SB0 (SB1) | • CMDT is set. | • CMDD is set. | (1) Data transmitted after REL signal output is address.<br>(2) Data transmitted, 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 reception is completed | $\overline{SCK0}$ 9 ... $\overline{BUSY}$ ... | ① ACKE = 1<br>② ACKT is set. | • ACKD is set. | Indicates completion of reception. |
| Busy signal ($\overline{BUSY}$) | Slave | Low level signal output on SB0 (SB1) after acknowledge signal | SB0 (SB1) D0 $\overline{ACK}$ ... READY<br>SB0 (SB1) D0 $\overline{ACK}$ $\overline{BUSY}$ ... READY | • BSYE = 1 | – | Indicates that serial transfer 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 | | ① BSYE = 0<br>② Execution of instruction to write data to SIO0 (Transfer start request) | – | Indicates that serial transfar is enabled. |

$\mu$PD75518(A)

**Table 4-10   Various Signals (2/2)**

| Signal name | Output device | Definition | Timing chart | Condition for output | Flag operation | Meaning of signal |
|---|---|---|---|---|---|---|
| Serial clock (SCK0) | Master | Synchronous clock for outputting address/command/data, ACK signal, synchronous BUSY signal, and so on. Address/command/data is output during first 8 clock cycles. | SCK0 1 2 7 8 9 10 / SB0 (SB1) | Execution of instruction to write data to SIO0 when CSIE0 = 1 (Transfer start request) Note 2 | IRQCSI0 is set (on rising edge of 9th clock) Note 1 | Timing of signal output on serial data bus |
| Address (A7 - A0) | Master | 8-bit data transferred in phase with SCK0 after REL signal and CMD signal output | SCK0 1 2 7 8 / SB0 (SB1) REL CMD | | | Address of slave device on serial bus |
| Command (C7 - C0) | Master | 8-bit data transferred in phase with SCK0 after only CMD signal is output, with REL signal not being output | SCK0 1 2 7 8 / SB0 (SB1) CMD | | | Directions and messages to slave device |
| Data (D7 - D0) | Master/slave | 8-bit data transferred in phase with SCK0, with neither REL signal nor CMD signal being output | SCK0 1 2 7 8 / SB0 (SB1) | | | Data processed by slave or master |

**Notes 1.** When WUP = 0, IRQCSI0 is always set on the 9th rising edge of SCK0.
When WUP = 1, IRQCSI0 is set on the 9th rising edge of SCK0 only if a received address matches the value of the slave address register (SVA).
**2.** If the BUSY state is present, data transfer is started after the READY state is set.

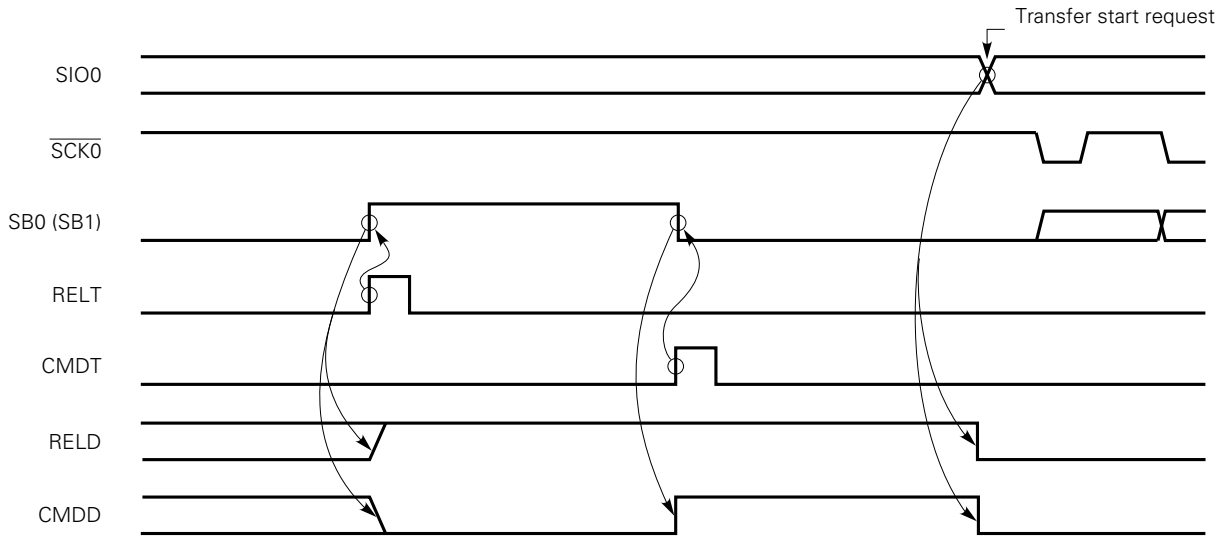**Fig. 4-38  Operations of RELT, CMDT, RELD, and CMDD (Master)**



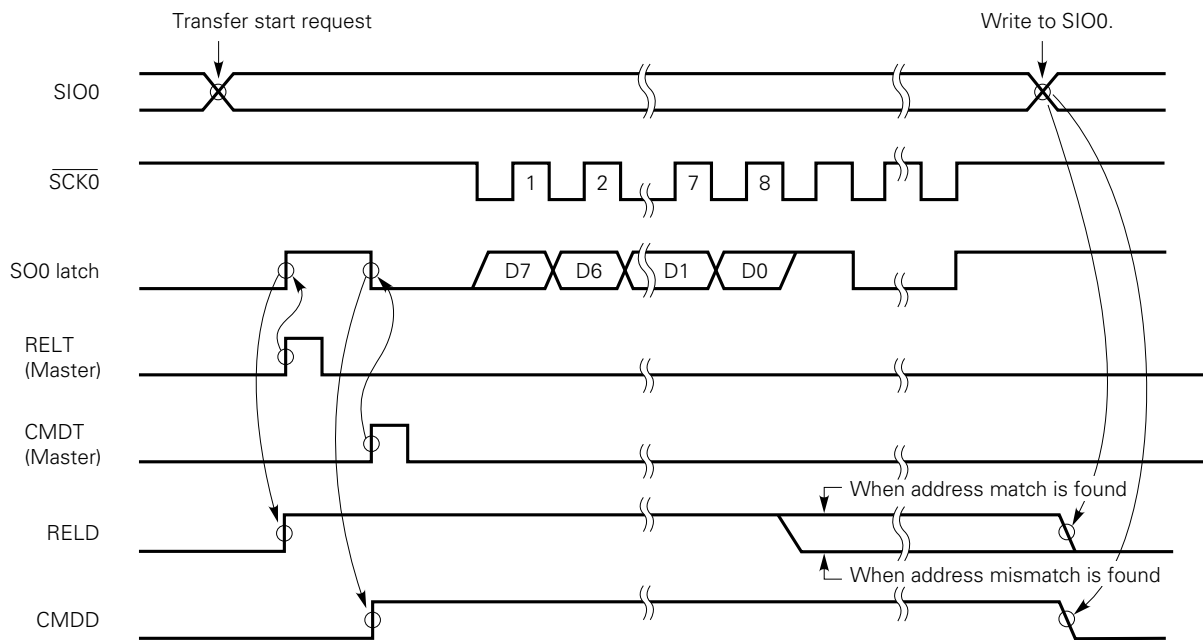**Fig. 4-39  Operations of RELT, CMDT, RELD, and CMDD (Slave)**
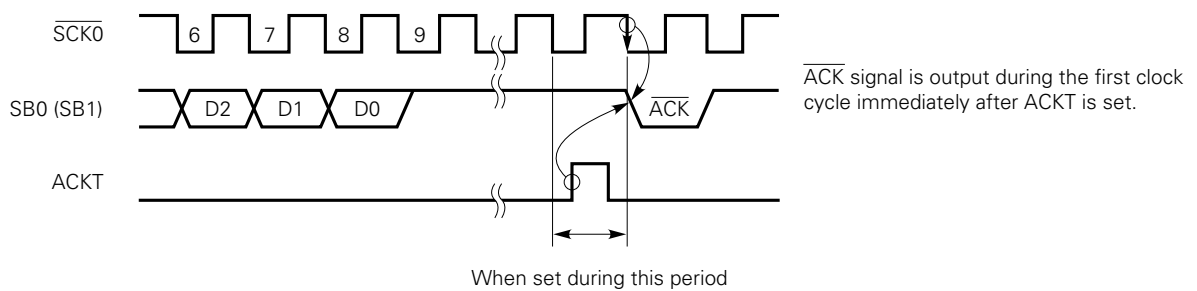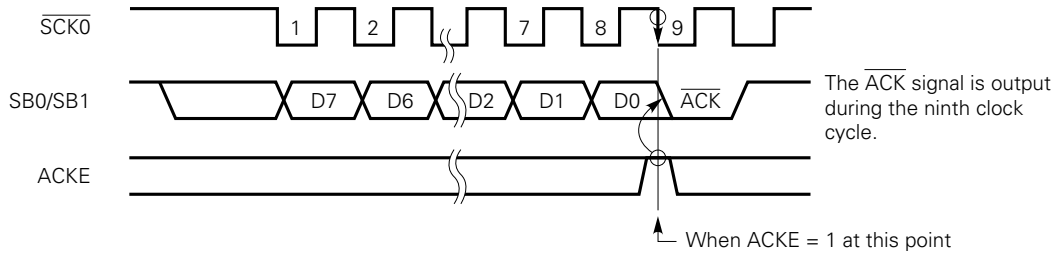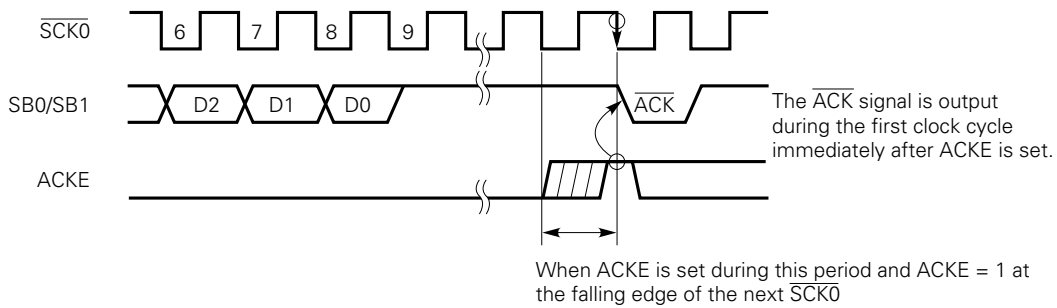


**Fig. 4-40  Operation of ACKT**



$\overline{ACK}$ signal is output during the first clock cycle immediately after ACKT is set.

When set during this period

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

**Fig. 4-41  Operation of ACKE**

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



The $\overline{ACK}$ signal is output during the ninth clock cycle.

When ACKE = 1 at this point

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



The $\overline{ACK}$ signal is output during the first clock cycle immediately after ACKE is set.

When ACKE is set during this period and ACKE = 1 at the falling edge of the next $\overline{SCK0}$

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



The $\overline{ACK}$ signal is not output.

When ACKE = 0 at this point
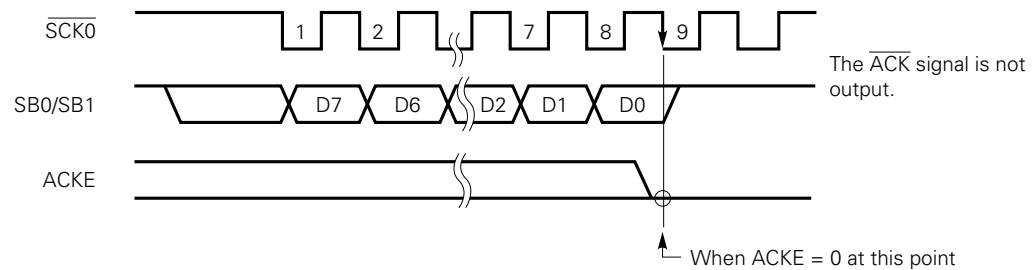
**(d)  When ACKE = 1 period is too short**



The $\overline{ACK}$ signal is not output.

When ACKE is set or cleared during this period and ACKE = 0 at the falling edge of $\overline{SCK0}$

98

**Fig. 4-42   Operation of ACKD**

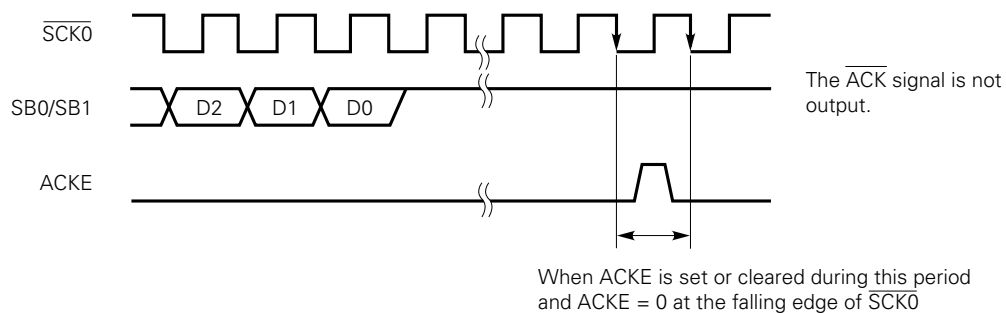**(a)  When $\overline{ACK}$ signal is output during ninth $\overline{SCK0}$ clock**
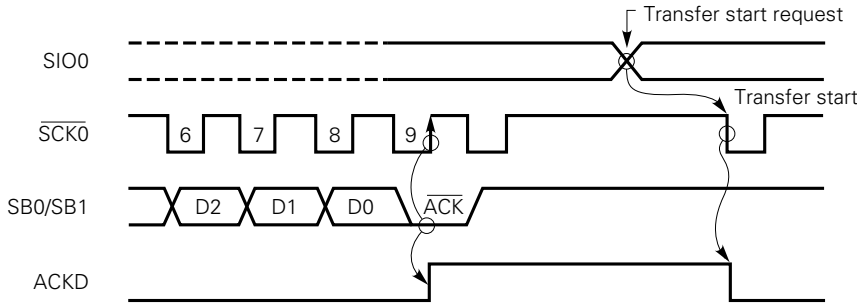


**(b)  When $\overline{ACK}$ signal is output after ninth $\overline{SCK0}$ clock**
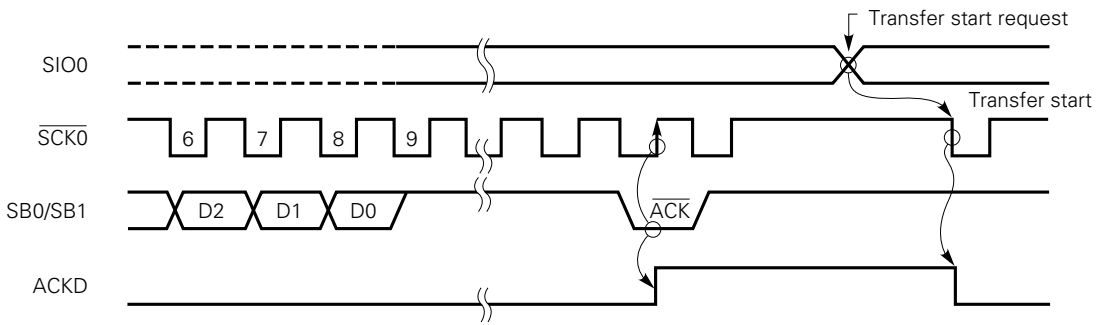


**(c)  Clear timing for case where start of transfer is requested during $\overline{BUSY}$**



**Fig. 4-43   Operation of BSYE**



When BSYE = 1 at this point

When reset operation is executed during this period and BSYE = 0 at the falling edge of $\overline{SCK0}$.

**4.8.5  Serial Interface (Channel 0) Operation**

**(1)  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.  When the $\overline{\text{RESET}}$ signal is entered, the operation halt mode is set.
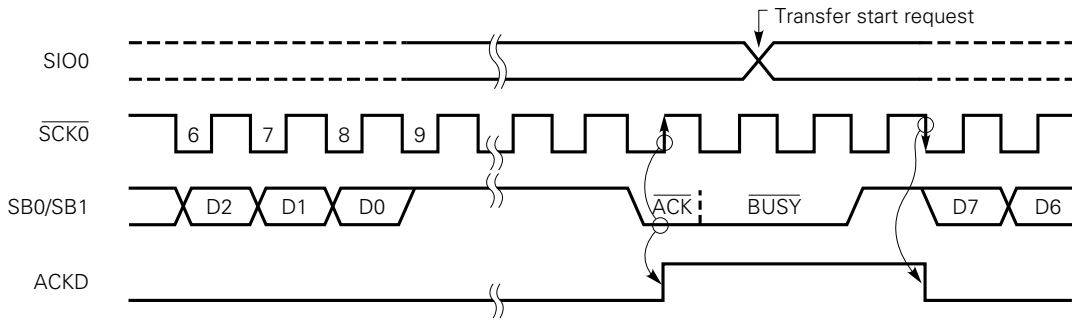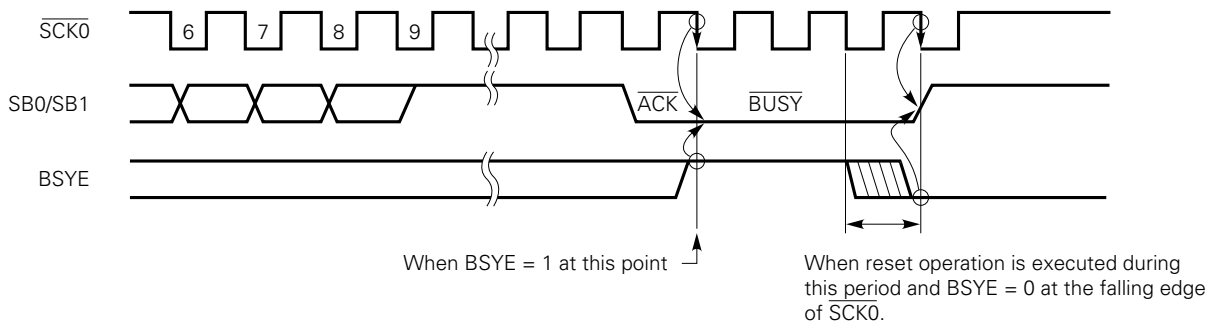
The P02/SO0/SB0 pin and P03/SI0/SBI pin function as input-only port pins.  The P01/$\overline{\text{SCK0}}$ pin can be used as an input port pin by setting the serial operation mode register 0.

**(2)  Three-wire serial I/O mode operations**

The three-wire serial I/O mode is compatible with other modes used in the 75X series, $\mu$PD7500 series, and 78K series.

Communication is performed using three lines:  Serial clock ($\overline{\text{SCK0}}$), serial output (SO0), and serial input (SI0).

**(a)  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 ($\overline{\text{SCK0}}$).  Transmit 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. 4-44  Timing of Three-Wire Serial I/O Mode**



Completion of transfer

Transfer  is started in phase with falling edge of $\overline{\text{SCK0}}$.

Execution of instruction that writes data to SIO0 (Transfer start request)

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.

The output state of the $\overline{\text{SCK0}}$ pin can be controlled by manipulating the P01 output latch in the output mode (internal system clock mode).  (See **Section 4.8.7.**)

**(b)  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. 4-45 shows the configuration of shift register 0 (SIO0) and internal bus.  As shown in Fig. 4-45, 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. 4-45    Transfer Bit Switching Circuit**
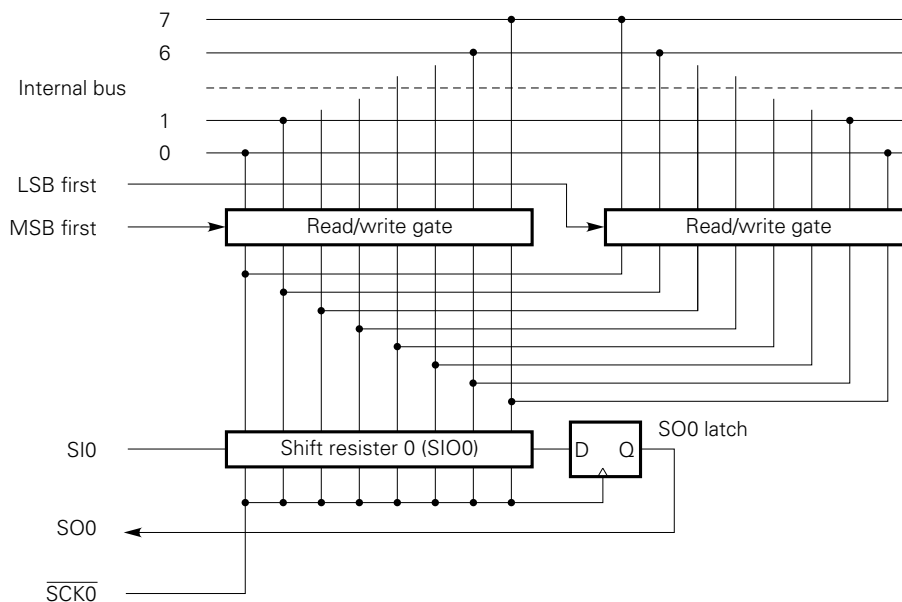


The first bit is switched by changing the order of data bits written to shift register 0 (SIO0).  The shift operation order of SIO0 is always the same.
Accordingly, the first bit must be switched between the MSB and LSB before writing data to the shift register 0.

**(3) 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).
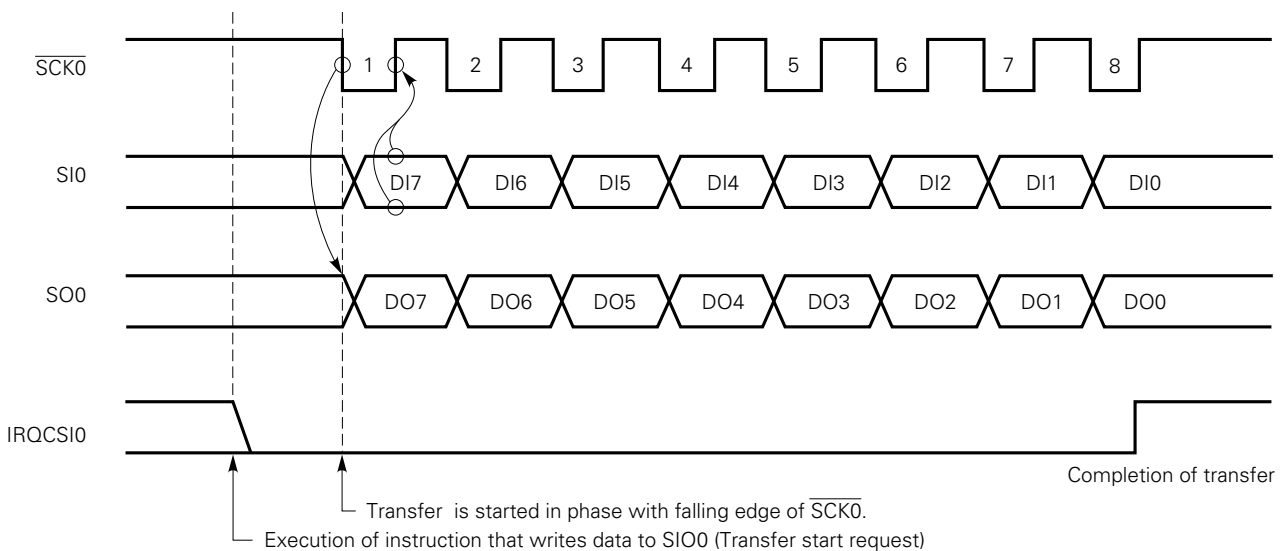
**(a) Communication operation**

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 ($\overline{SCK0}$). Transmit 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 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. 4-46   Timing of Two-Wire Serial I/O Mode**
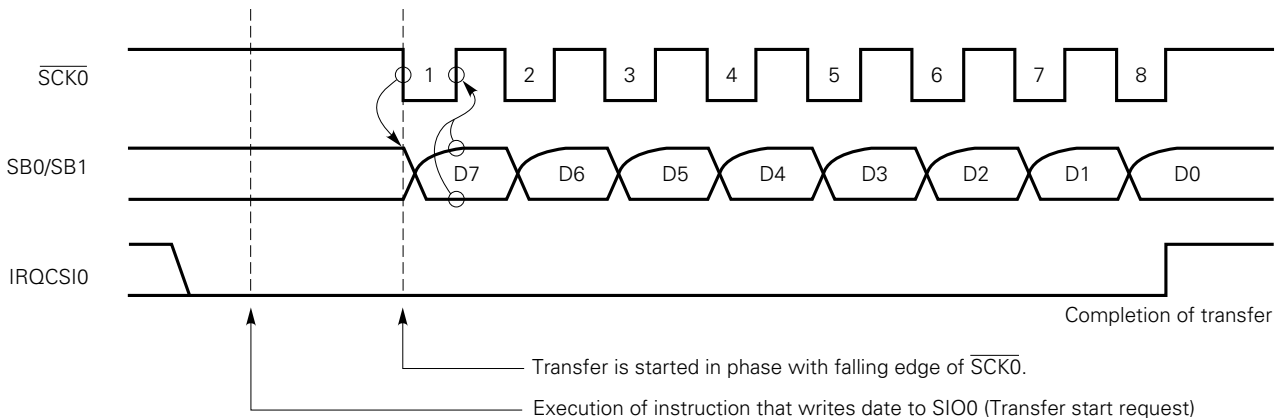


**The SB0 or SB1 pin becomes an N-ch open-drain I/O when specified as the serial data bus, so the voltage level on that pin must be pulled up externally.**

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.

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 **Section 4.8.7**.)

**(4) 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. 4-47 is an example of the SBI system configuration.

**Fig. 4-47    Example of SBI System Configuration**
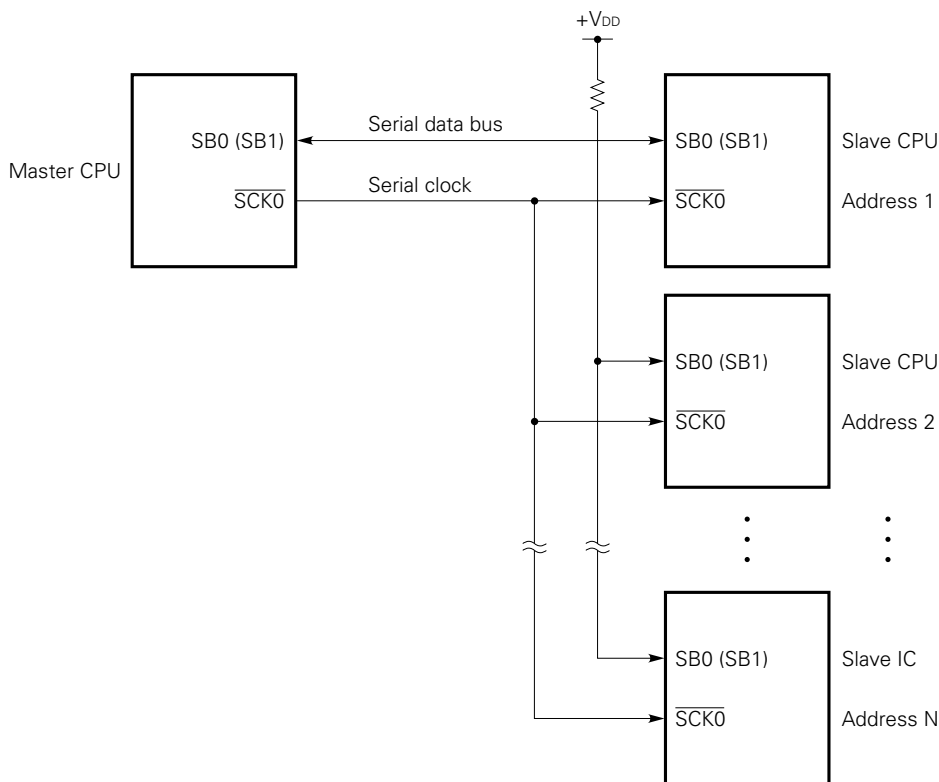


**Cautions 1.**   **In the SBI mode, the serial data bus pin SB0 (or SB1) is an open-drain output. So the serial data bus line is placed in the wired OR state. A pull-up resistor is required for the serial data bus line.**

        **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.**

**(a) 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. 4-48   Timing of SBI Transfer**

**Address transfer**



**Command transfer**

Command signal



**Data transfer**
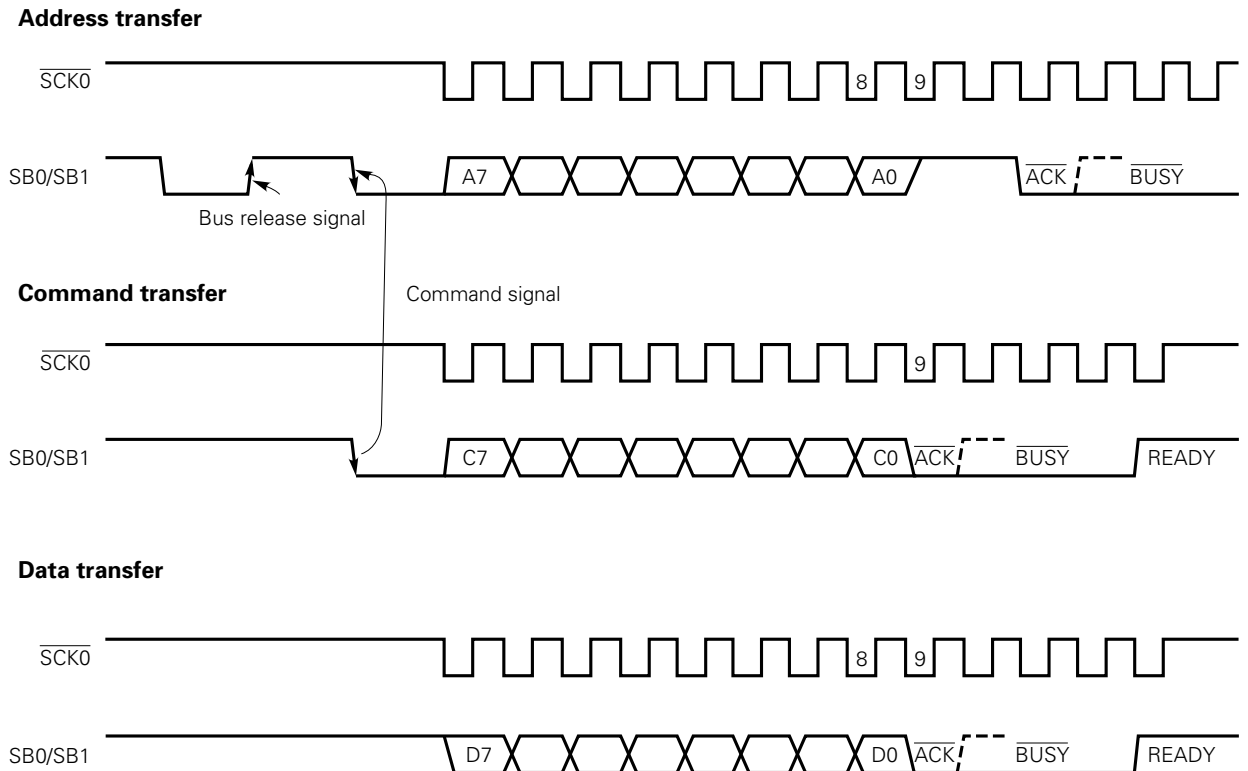


**(b) Communication operation**

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. 4-49 to 4-52 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 ($\overline{SCK0}$).  Transmit 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. 4-49  Address Transmission from Master Device to Slave Device (WUP = 1)**

Master device processing (transmitter)

Program processing — Set CMDT | Set RELT | Set CMDT | Write to SIO0 | Interrupt handling (preparation for next serial transfer)

Hardware operation — Serial transmission | Generate IRQCSI0 | Set ACKD | Stop SCK0

Transfer line

SCK0 pin — 1 2 3 4 5 6 7 8 9

SB0 pin — A7 A6 A5 A4 A3 A2 A1 A0 | ACK | BUSY | READY

Address

Slave device processing (receiver)

Program processing — WUP ← 0 | Set ACKT | Clear BUSY

Hardware operation — Set CMDD | Clear CMDD / Set RELD | Set CMDD | Serial reception | Generate IRQCSI0 | Output ACK | Output BUSY | Clear BUSY

(When SVA = SIO0)

**Fig. 4-50  Command Transmission from Master Device to Slave Device**
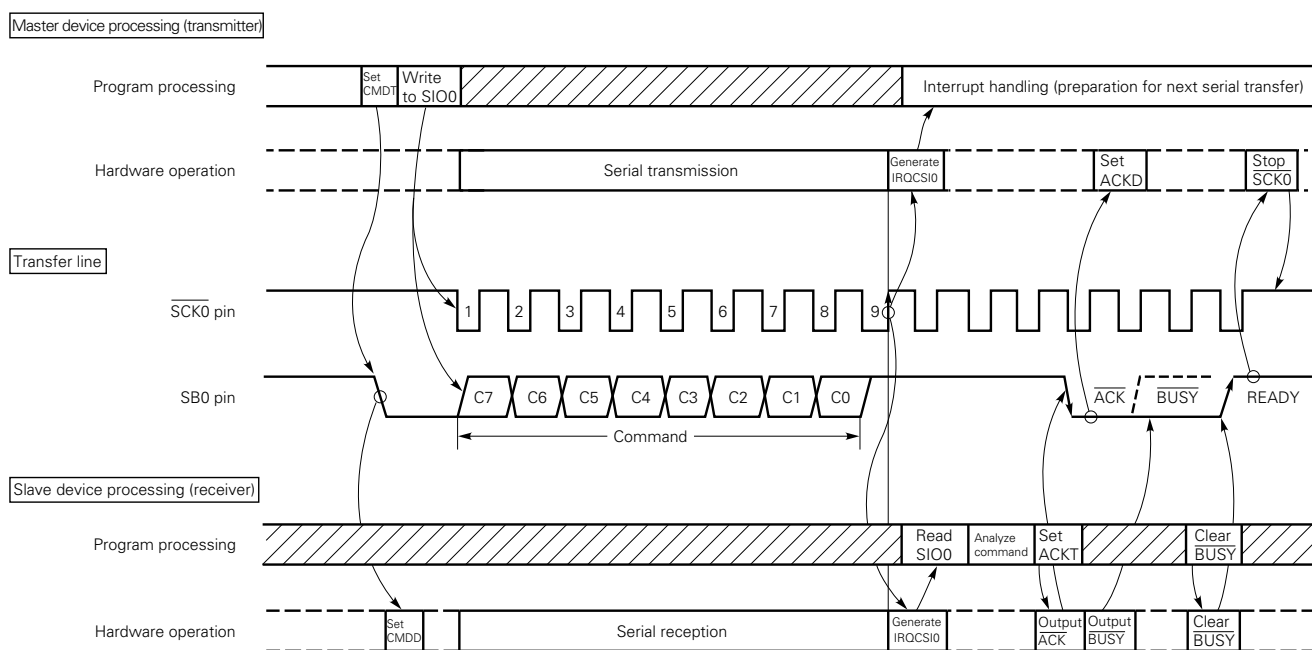
**Fig. 4-51   Data Transmission from Master Device to Slave Device**
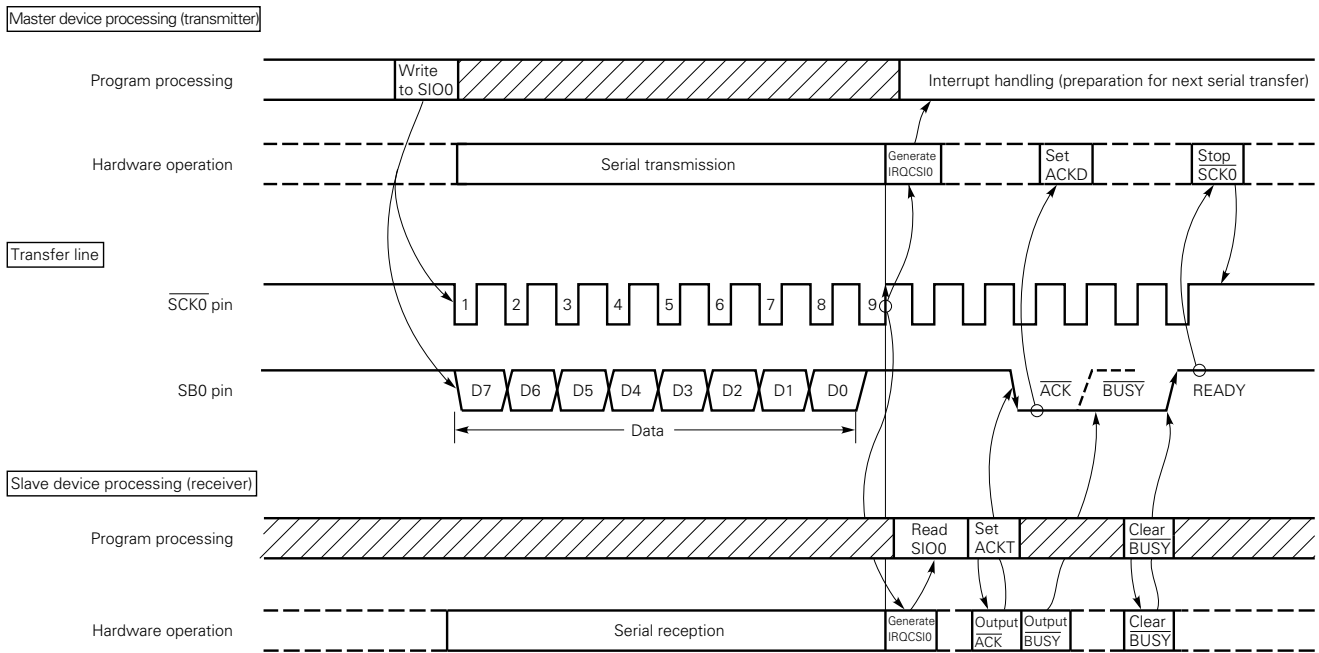
Fig. 4-52   Data Transmission from Master Device to Slave Device

Master device processing (receiver)

Program processing

Write FFH to SIO0 | Read SIO0 | Write FFH to SIO0 | Set ACKT | Receive data processing

Hardware operation

Stop SCK0 | Serial reception | Generate IRQCSI0 | Output ACK | Serial reception

Transfer line

SCK0 pin

1 2 3 4 5 6 7 8 9 | 1 2

SB0 pin

BUSY | READY | D7 D6 D5 D4 D3 D2 D1 D0 | ACK | BUSY | READY | D7 D6

Data

Slave device processing (transmitter)

Program processing

Write to SIO0 | Write to SIO0

Hardware operation

Clear BUSY | Serial transmission | Generate IRQCSI0 | Set ACKD | Output BUSY | Clear BUSY

**4.8.6  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 $\overline{\text{SCK0}}$ is high.

**Caution   Transfer 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 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 transmit data from RAM
SEL    MB15             ; or CLR1  MBE
XCH    XA, SIO0         ; Exchanges transmit data with receive data and starts transfer
```

### 4.8.7 Manipulation of $\overline{\text{SCK0}}$ Pin Output

The $\overline{\text{SCK0}}$/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 $\overline{\text{SCK0}}$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 $\overline{\text{SCK0}}$/P01 pin output is explained below.

① Set serial operation mode register 0 (CSIM0) ($\overline{\text{SCK0}}$ pin: output mode). When serial transfer is halted, $\overline{\text{SCK0}}$ 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 $\overline{\text{SCK0}}$/P01 pin by software

| | | |
|---|---|---|
| SEL | MB15 | ; or CLR1 MBE |
| MOV | XA, #10000011B | ; $\overline{\text{SCK0}}$ (fx/2³), output mode |
| MOV | CSIM0, XA | |
| CLR1 | 0FF0H.1 | ; $\overline{\text{SCK0}}$/P01 ← 0 |
| SET1 | 0FF0H.1 | ; $\overline{\text{SCK0}}$/P01 ← 1 |

**Fig. 4-53 $\overline{\text{SCK0}}$/P01 Pin Circuit Configuration**
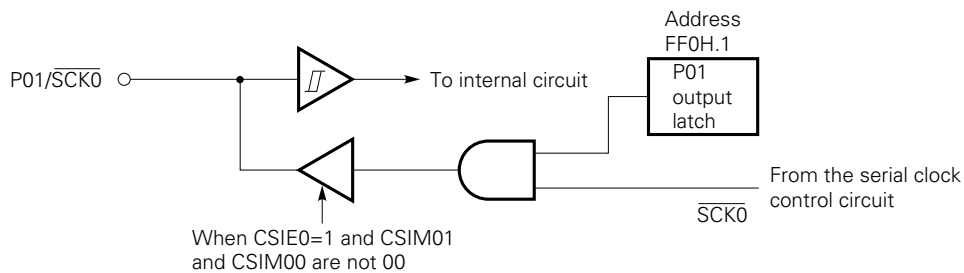


The P01 output latch is mapped to bit 1 of address FF0H. A $\overline{\text{RESET}}$ signal sets the P01 output latch to 1.

**Cautions 1. During normal serial transfer, 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**

## 4.9   SERIAL INTERFACE (CHANNEL 1)

### 4.9.1   Serial Interface (Channel 1) Functions

The $\mu$PD75518(A) has 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 ($\overline{\text{SCK1}}$), 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.

### 4.9.2  Serial Interface (Channel 1) Configuration

Fig. 4-54 shows the block diagram of the serial interface (channel 1).

**Fig. 4-54   Block Diagram of the Serial Interface (Channel 1)**
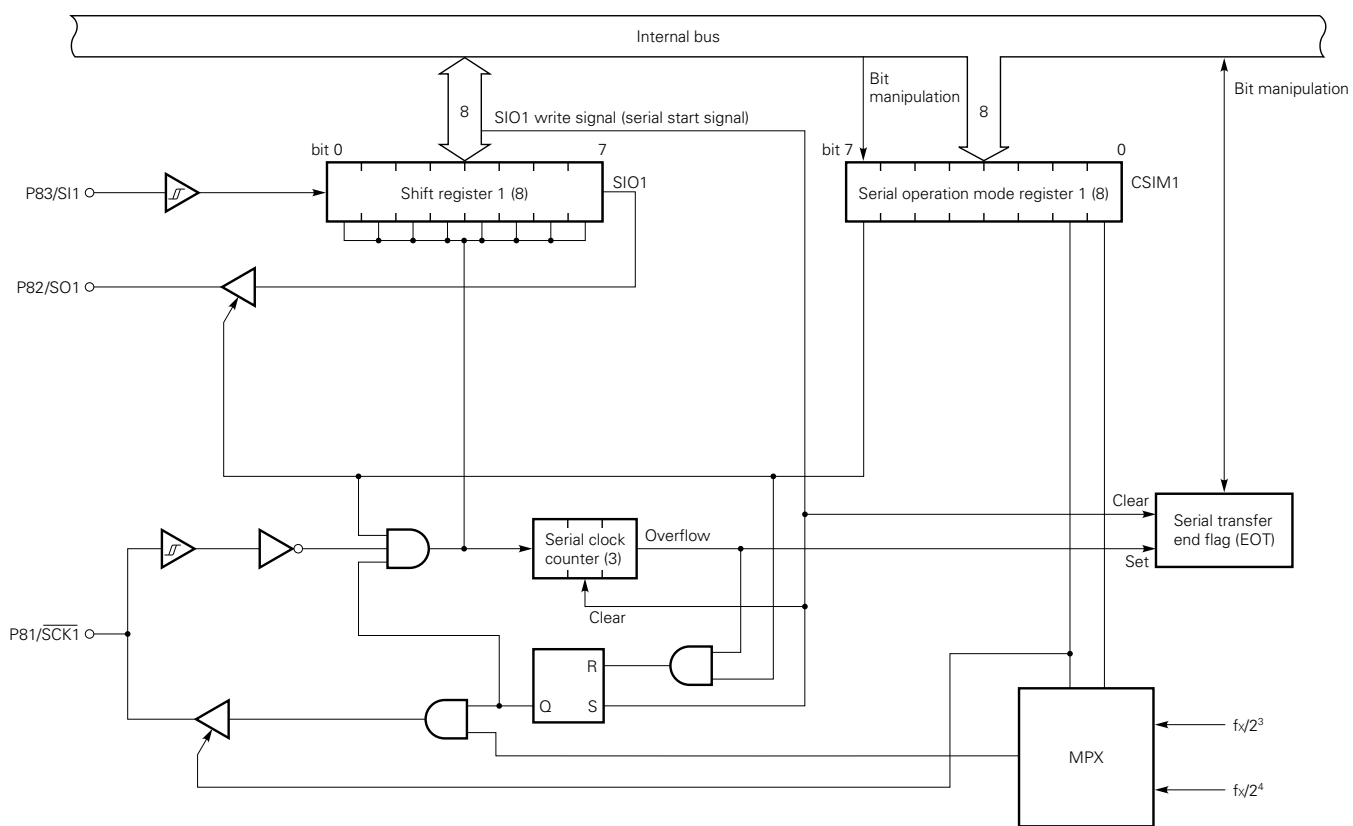
### 4.9.3 Register Functions

**(1) Serial operation mode register 1 (CSIM1)**

Fig. 4-55 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.

When the $\overline{\text{RESET}}$ signal is input, this register is set to 00H.

**Fig. 4-55 Format of Serial Operation Mode Register 1 (CSIM1)**

| 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)

Serial interface operation enable/disable specification bit (W)
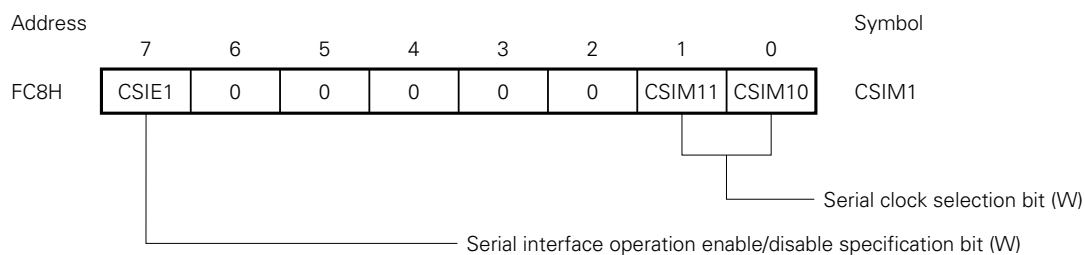
**Remark** (W): Write only

**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 at 4.19 MHz and 6.0 MHz are indicated in parentheses.

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

| | | Shift register operation | Serial clock counter | EOT 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. | Used in serial interface as well as for port 8 |

**Caution   Be sure to write 0 in bits 2 to 6 of the serial operation mode register 1 (CSIM1).**

**Example**   To select $f_X/2^4$ as the serial clock, and set the serial transfer end flag EOT to 1 each time serial transfer terminates

```
SEL   MB15            ; or CLR1  MBE
MOV   XA, #10000010B
MOV   CSIM1, XA        ; CSIM1 ← 10000010B
```

**(2)  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.  The MSB is used as the first bit of transfer.

In tansmission, data written to SIO1 is output on the serial output (SO1).  In reception, 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 transmission or reception 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 1 is excluded.
- When the serial clock is masked after 8-bit serial transfer
- When $\overline{\text{SCK1}}$ is high

**4.9.4  Serial Interface (Channel 1) Operation**

**(1)  Operation halt mode**

The operation halt mode is used when serial transfer is not performed, which is set by setting 0 in CSIE1. 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.

When the $\overline{\text{RESET}}$ signal is entered, the operation halt mode is set.  The P82/SO1 pin and P83/SI1 pin function as input-only port pins.  The P81/$\overline{\text{SCK1}}$ pin can be used as an input port pin by setting serial operation mode register 1.

**(2) Three-wire serial I/O mode operations**

The three-wire serial I/O mode is compatible with other modes used in the 75X series, $\mu$PD7500 series, and 78K series.  This mode is set by setting CSIE1 to 1.

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 ($\overline{\text{SCK1}}$).  Transmit 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).

Setting the serial transfer and flag (EOT) cannot release the standby function.

**Fig. 4-56   Timing of the Three-Wire Serial I/O Mode**



Completion of transfer

Transfer is started in phase with falling edge of $\overline{\text{SCK1}}$.

Execution of instruction that writes data to SIO1 (Transfer start request)
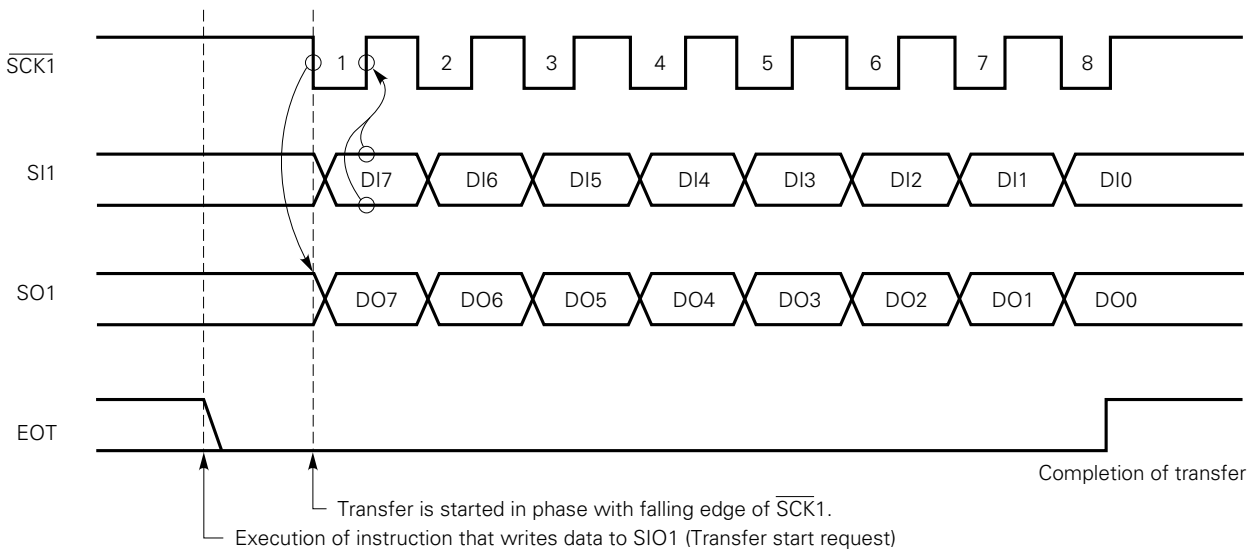
**Example**   To transfer the RAM data specified by the HL register pair to SIO1, load the SIO1 data to the accumulator, and start serial transfer:

```
MOV   XA, @HL      ; Fetch transmit data from RAM
SEL   MB15         ; or CLR1  MBE
XCH   XA, SIO1     ; Exchange transmit data and receive data, and start transfer
```

### 4.10 A/D CONVERTER

The μPD75518(A) 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. 4-57 shows the configuration of the A/D converter.

**Fig. 4-57   Block Diagram of the A/D Converter**
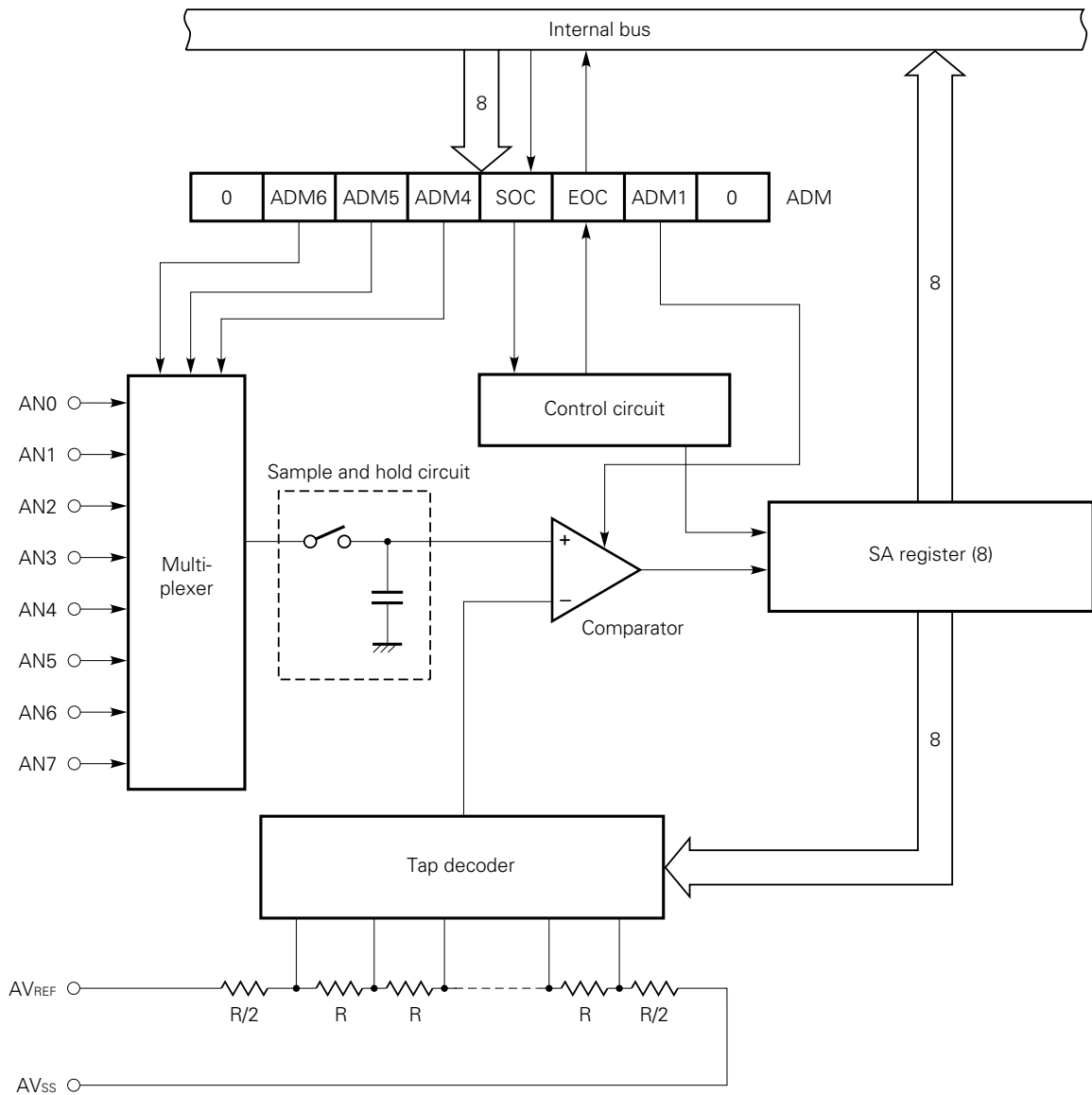
**(2)  Pins of the A/D converter**

    **(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$_{SS}$.

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

The A/D conversion mode register (ADM) is an 8-bit register which operates as follows:

- Selects analog input channels.
- Selects comparator bias voltage. **Note**
- Directs the start of conversion and detects the completion of conversion.

ADM is set with an 8-bit manipulation instruction.

Bit 2 (EOC) and bit 3 (SOC) can be manipulated on a bit-by-bit basis.

The generation of a $\overline{\text{RESET}}$ signal initializes ADM to 04H.  That is, only EOC is set to 1, with all bits cleared to 0.

**Note**  If the reference voltage (AV$_{REF}$) of the A/D converter does not exceed 0.65V$_{DD}$, the accuracy of conversion may be lowered.  To correct such lowered accuracy, selecting comparator bias voltage is provided.

**Fig. 4-58   Format of the A/D Conversion Mode Register**

Address                                                                Symbol

|     | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |     |
|-----|---|---|---|---|---|---|---|---|-----|
| FD8H | 0 | ADM6 | ADM5 | ADM4 | SOC | EOC | ADM1 | 0 | ADM |

**Comparator bias voltage selection**

| ADM1 | 0 | $AV_{REF} \geq 0.6V_{DD}$ |
|------|---|--------------------------|
|      | 1 | $AV_{REF} \leq 0.65V_{DD}$ |

**Reamark**  This bit may be set to either 0 or 1 for
$0.6V_{DD} \leq AV_{REF} \leq 0.65V_{DD}$.

**End of conversion flag (R)**

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

**Start of conversion bit (W)**

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

**Analog channel selection bit (W)**

| 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 $2^4$/fx seconds (2.67 $\mu$s at fx = 6.0 MHz)[Note] after SOC is set.  (For details, see item (5).)**

**Note**   $2^4$/fx seconds = 3.81 $\mu$s for fx = 4.19 MHz

**(4) SA register (successive approximation register)**

The SA 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.

The generation of a $\overline{\text{RESET}}$ signal makes SA undefined.

SA is mapped to address FDAH.


**(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). Comparator bias voltage selection is specified by setting bit 1 in the A/D conversion mode register (ADM1).

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, ADM bit 2 (EOC) is set to 1. Fig. 4-59 shows the timing chart of A/D conversion.

The A/D converter is used as follows:

① Select analog input channels and comparator bias voltage (by setting ADM6, ADM5, ADM4, and ADM1).

② Direct the start of A/D conversion (by setting SOC).

③ Wait for the completion of A/D 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 $2^4$/fx seconds (fx = 6.0 MHz: 2.67 $\mu$s, or fx = 4.19 MHz: 3.81 $\mu$s) 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 4-11 has elapsed after the setting of SOC. Table 4-11 also indicates A/D conversion times.


**Table 4-11  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 |
|---|---|---|---|---|---|---|
| SCC1 | SCC0 | PCC1 | PCC0 | | | |
| 0 | 0 | 0 | 0 | 168/fx s**Note** (28.0 $\mu$s/fx = 6.0 MHz) | Waiting not required | 3 machine cycles |
| | | 0 | 1 | | 1 machine cycle | 11 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 $\mu$s for fx = 4.19 MHz


**Remark**  ×: Don't care

**Fig. 4-59  Timing Chart of A/D Conversion**
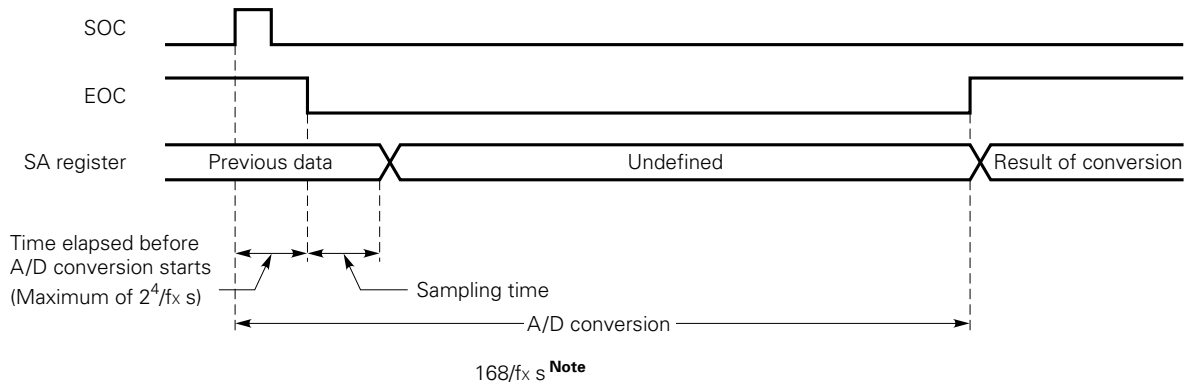


**Note**  28.0 $\mu$s (for fx = 6.0 MHz) or 40.1 $\mu$s (for fx = 4.19 MHz)

Fig. 4-60 shows the relationship between analog input voltages and 8-bit digital data obtained by A/D conversion.

**Fig. 4-60  Relationship (Ideal) between Analog Input Voltages and Results of A/D Conversion**
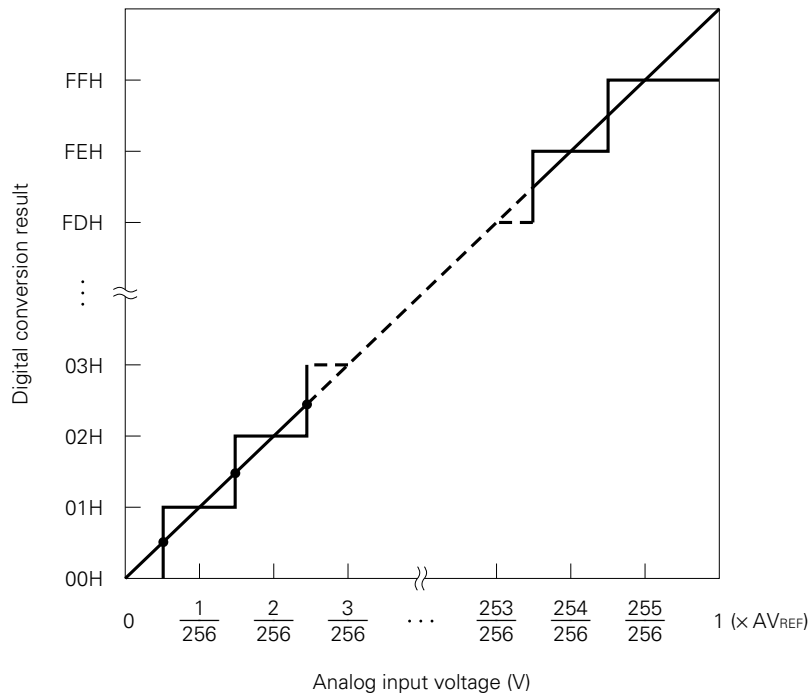
**(6) Notes on the standby mode**

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 P113 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 P113 can reduce power consumption.

**Fig. 4-61  Reducing Power Consumption in the Standby Mode**
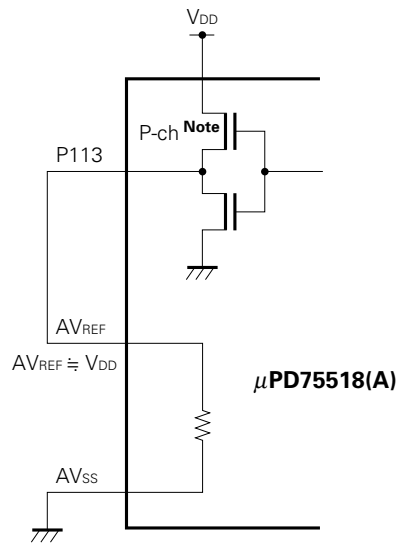


**Note**  The drive capability of P-ch is higher than that of other ports.

**(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. 4-62, a capacitor should be externally connected.

**Fig. 4-62  Analog Input Pin Connection**
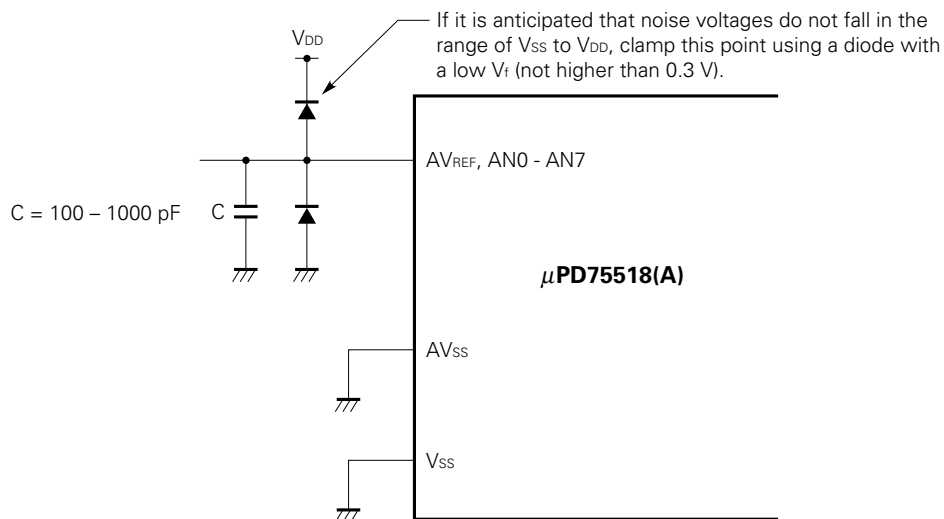


**(c) AN4/P150 to AN7/P153 pins**

The analog input pins (AN4 to AN7) are also used for an input port (PORT15).

When any of AN4 to AN7 is selected for A/D conversion, no input instruction must be executed for PORT15 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.

## 4.11  BIT SEQUENTIAL BUFFER:  16 BITS

The bit sequential buffer is special data memory for bit manipulations.  In particular, the buffer allows bit manipulations to be performed very easily by sequentially changing address and bit specifications. So the buffer is 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. 4-63  Format of the Bit Sequential Buffer
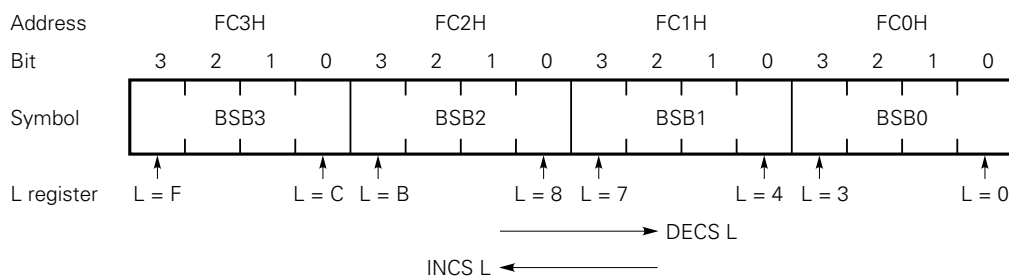


**Remark**  In pmem.@L addressing, bit specification is shifted according to the L register.

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.

**Example**  The 16-bit data of BUFF1 and BUFF2 are output from bit 0 of port 3 in serial mode.
Program example

```
            CLR1    MBE
            MOV     XA, BUFF1
            MOV     BSB0, XA        ; Set BSB0, 1
            MOV     XA, BUFF2
            MOV     BSB2, XA        ; Set BSB2, 3
            MOV     L, #0
LOOP0:      SKT     BSB0, @L        ; Test specified BSB bit
            BR      LOOP1
            NOP                     ; Dummy (timing adjustment)
            SET1    PORT3.0         ; Set bit 0 of port 3
            BR      LOOP2
LOOP1:      CLR1    PORT3.0         ; Clear bit 0 of port 3
            NOP                     ; Dummy (timing adjustment)
            NOP
LOOP2:      INCS    L               ; L ← L+1
            BR      LOOP0
            RET
```

## 5. INTERRUPT FUNCTION

The μPD75518(A) has nine interrupt sources and can handle multiple interrupts with a priority.
The μPD75518(A) is also provided with two features for accepting testable interrupts.

**Table 5-1  Interrupt Sources**

| Interrupt source | | Internal/external | Priority[Note 1] | Vectored interrupt request signal (vector table address) |
|---|---|---|---|---|
| INTBT | (Reference time interval signal from basic interval timer) | Internal | 1 | VRQ1 (0002H) |
| INT4 | (Detection of both rising and falling edges) | External | | |
| INT0 | (Rising/falling edge detection specification) | External | 2 | VRQ2 (0004H) |
| INT1 | | External | 3 | VRQ3 (0006H) |
| INTCSI0 | (Serial data transfer completion signal) | Internal | 4 | VRQ4 (0008H) |
| INTT0 | (Match signal between programmable timer/counter count register and modulo register) | Internal | 5 | VRQ5 (000AH) |
| INTTPG | (Match signal from timer/pulse generator) | Internal | 6 | VRQ6 (000CH) |
| INT2 | (Rising edge detection for an INT2 pin input signal, or falling edge detection for either of KR0 to KR7 pin input signals)[Note 2] | External | Testable input signal (Sets IRQ2 and IRQW.) | |
| INTW | (Signal from clock timer) | Internal | | |

**Notes 1.** The priority is used when two or more interrupt requests are issued at a time.
**2.** See **(3)** in **Section 5.2** for details on INT2.

The following functions are provided for the interrupt control circuit of the μPD75518(A).

(a)  Vectored 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 μPD75518(A) 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**
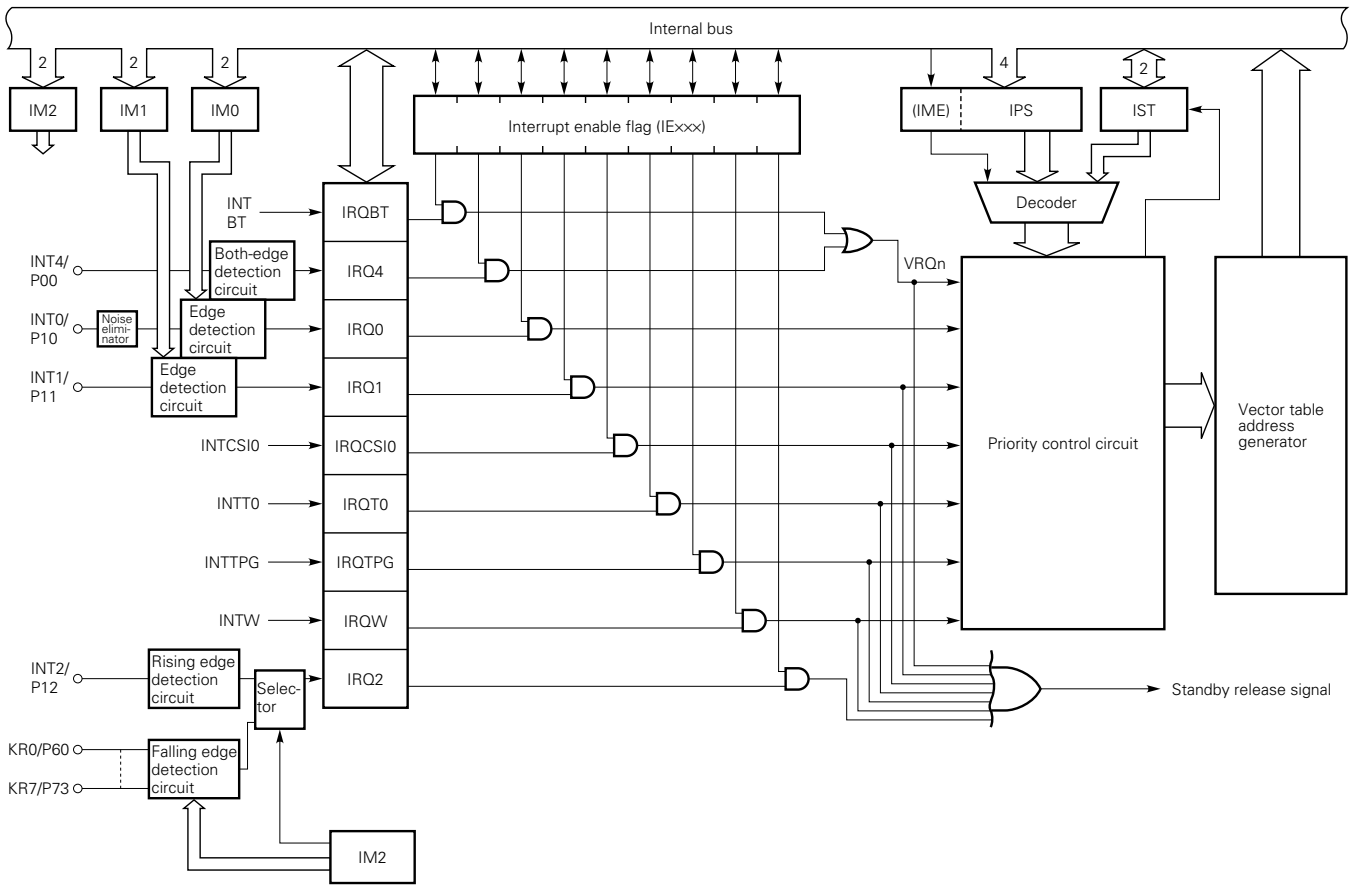
## 5.2  HARDWARE OF THE INTERRUPT CONTROL CIRCUIT

**(1)  Interrupt request flag and interrupt enable flag**

The following nine interrupt request flags (IRQ×××) corresponding to the interrupt sources are available.

| | |
|---|---|
| INT0 interrupt request flag (IRQ0) | Serial interface interrupt request flag (IRQCSI0) |
| INT1 interrupt request flag (IRQ1) | Timer/event counter 0 interrupt request flag (IRQT0) |
| INT2 interrupt request flag (IRQ2) | Timer/pulse generator interrupt request flag (IRQTPG) |
| INT4 interrupt request flag (IRQ4) | Clock timer interrupt request flag (IRQW) |
| BT interrupt request flag (IRQBT) | |

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**.)

The following nine interrupt request flags (IE×××) corresponding to the interrupt request flags are available.

| | |
|---|---|
| INT0 interrupt enable flag (IE0) | Serial interface enable flag (IECSI0) |
| INT1 interrupt enable flag (IE1) | Timer/event counter 0 interrupt enable flag (IET0) |
| INT2 interrupt enable flag (IE2) | Timer/pulse generator interrupt enable flag (IETPG) |
| INT4 interrupt enable flag (IE4) | Clock timer interrupt enable flag (IEW) |
| BT interrupt enable flag (IEBT) | |

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 vectored 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.

**Example**  EI      IE0     ; Enables INT0.
DI      IE1     ; Disables INT1.
SKTCLR  IRQCSI0  ; Skips and clears the interrupt request flag if IRQCSI0 is 1.

When an interrupt request flag is set with an instruction, a vectored interrupt is executed irrespective of whether an interrupt occurs.

When a $\overline{\text{RESET}}$ signal is generated, an interrupt request flag and its corresponding interrupt enable flag are cleared 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 |
| IRQW | Set by a signal from the clock timer. | IEW |
| IRQ2 | Set by a detected rising edge of an INT2/P12 pin input signal, or a detected falling edge of one of a KR0/P60-KR7/P73 pin input signals. | IE2 |

**(2)  Configurations of INT0, INT1, and INT4 pins**

(a)  As shown in Fig. 5-2 (a), INT0 is configured as an external interrupt pin that enables detection edge selection.

In addition, the INT0 pin is provided with a noise elimination function using a sampling clock.  The noise eliminator eliminates pulses narrower than two-sampling-clock-cycle pulses ($2t_{CY}$**Note** or $128/f_X$) as noise and accepts pulses wider than as interrupt signals.

INT0 has two sampling clocks $\Phi$ and $f_X/64$, either of which can be selected according to bit 3 (IM03) of the edge detection mode register (See **Fig. 5 - 3(a)**.)

Bits 0 and 1 (IM00 and IM01) of the edge detection mode register are used to select a detection edge. Fig. 5-3 (a) shows the format of IM0.  A 4-bit memory manipulation instruction is used to set IM0.  A $\overline{\text{RESET}}$ signal occurrence clears all bits to 0, and a rising edge is specified to be detected.

**Note**  $t_{CY}$ represents a cycle time.

**Cautions  1.  Since the INT0 input is sampled with a clock, INT0 does not operate in a standby mode.**
**2.  Input a pulse wider than two sampling clock cycles to the INT0/P10 pin.  Otherwise, the pulse is suppressed as noise by the noise eliminator when the pin is used as a port.**

(b)  As shown in Fig. 5-2 (b), INT1 is configured as an external interrupt pin that enables detection edge selection.

The edge detection mode register (IM1) is used to select a detection edge.

Fig. 5-3 (b) shows the format of IM1.  A 4-bit memory manipulation instruction is used to set IM1.  A $\overline{\text{RESET}}$ signal occurrence clears all bits to 0, and a rising edge is specified to be detected.

(c)  As shown in Fig. 5-2 (c), INT4 is configured as an external interrupt pin that enables detection of both rising and falling edges.

**Fig. 5-2  Configurations of INT0, INT1, and INT4 Pins**

**(a)  Configuration of INT0**
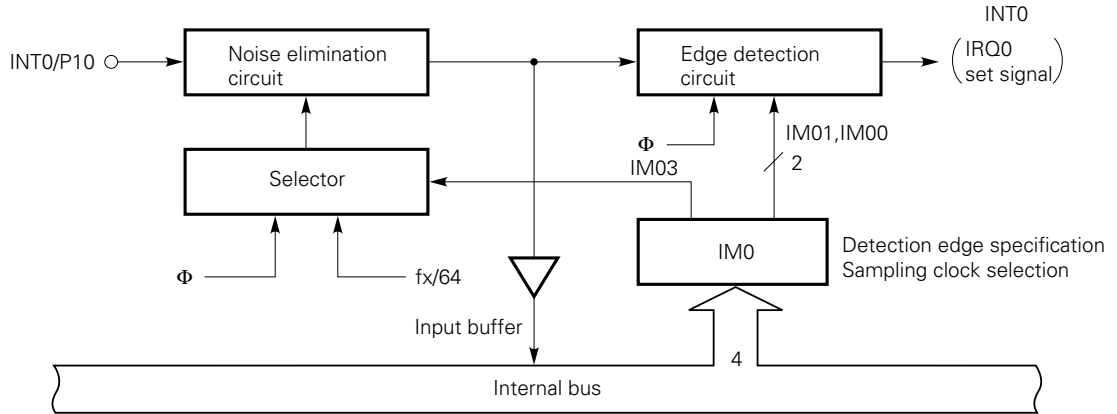


**(b)  Configuration of INT1**
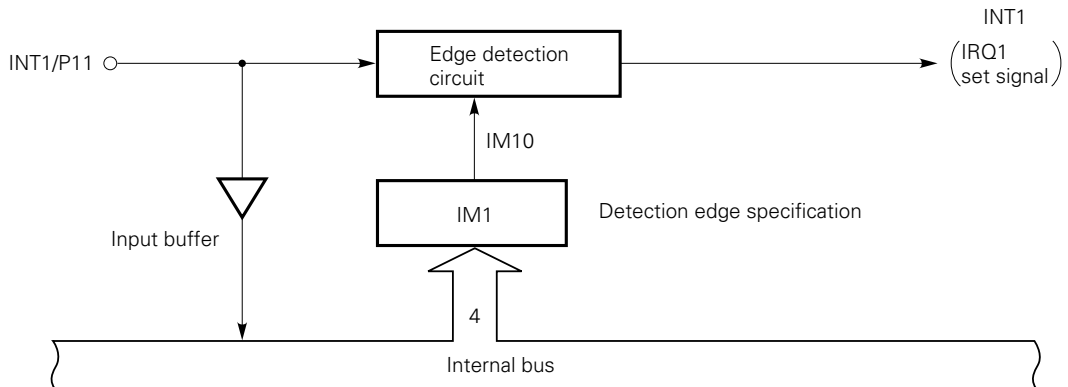


**(c)  Configuration of INT4**
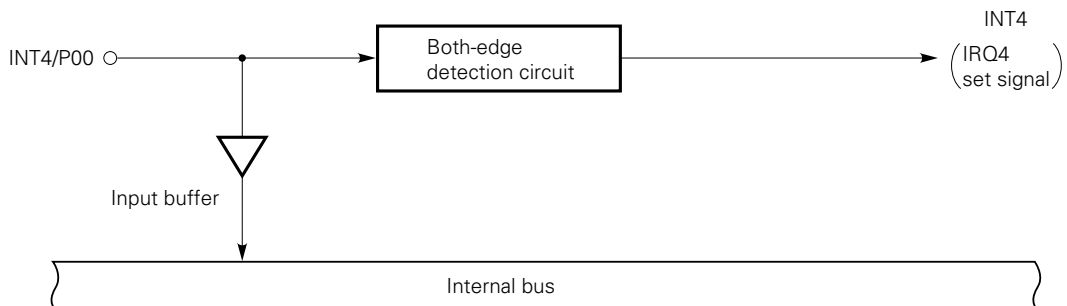
**Fig. 5-3  Format of Edge Detection Mode Registers**

**(a)  INT0 edge detection mode register (IM0)**

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| FB4H | IM03 | 0 | IM01 | IM00 | IM0 |

Detection edge specification

| | | |
|---|---|---|
| 0 | 0 | Specifies rising edge. |
| 0 | 1 | Specifies falling edge. |
| 1 | 0 | Specifies both rising and falling edges. |
| 1 | 1 | Ignored (interrupt request flag is not set.) |

Sampling clock

| | |
|---|---|
| 0 | $\Phi$ |
| 1 | $f_X/64$ (at 10.7 µs/6.0 MHz, or 15.3 µs/4.19 MHz) |

**(b)  INT1 edge detection mode register (IM1)**

| FB5H | 0 | 0 | 0 | IM10 | IM1 |
|---|---|---|---|---|---|

| | |
|---|---|
| 0 | Specifies rising edge. |
| 1 | Specifies falling edge. |

**(c)  INT2 edge detection mode register (IM2)**

| FB6H | 0 | 0 | IM21 | IM20 | IM2 |
|---|---|---|---|---|---|

| IM21 | IM20 | INT2 interrupt source | Interrupt input pin |
|---|---|---|---|
| 0 | 0 | Specifies rising edge of INT2 pin input. | INT2 (1) |
| 0 | 1 | Specifies falling edge of any of KRx pin inputs. | KR4 - KR7 (4) |
| 1 | 0 | | KR2 - KR7 (6) |
| 1 | 1 | | KR0 - KR7 (8) |

**Caution  Since changing or setting 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 $f_X/64$ 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) Configuration of INT2 and KR0 to KR7 (key interrupt) pins**
Fig. 5-4 shows the configuration of INT2 and KR0 to KR7. IRQ2 is set in one of the following modes with the edge detection mode register (IM2):

**(a) Detection of a rising edge of the INT2 pin input**
When a rising edge of the INT2 pin input is detected, IRQ2 is set.

**(b) Detection of a falling edge of one of the KR0 to KR7 pin inputs (key interrupt)**
One of the pins KR0 to KR7 is selected to be used for interrupt input with the edge detection mode register (IM2). When a falling edge of one of input signals applied to the selected pin is detected, IRQ2 is set.

**Example** If KR4 to KR7 are selected, and the level of signals input to KR4 to KR7 are all high, a falling edge appearing on any one of these inputs sets IRQ2.

**Caution If any of the selected pins has been input a low-level signal, a falling edge appearing on another pin does not set IRQ2.**

Fig. 5-3 (c) shows the format of IM2. A 4-bit memory manipulation instruction is used to set IM2. A $\overline{\text{RESET}}$ signal occurrence clears all bits to 0, and a rising edge is selected for INT2.

**Fig. 5-4  Configuration of INT2 and KR0 to KR7**
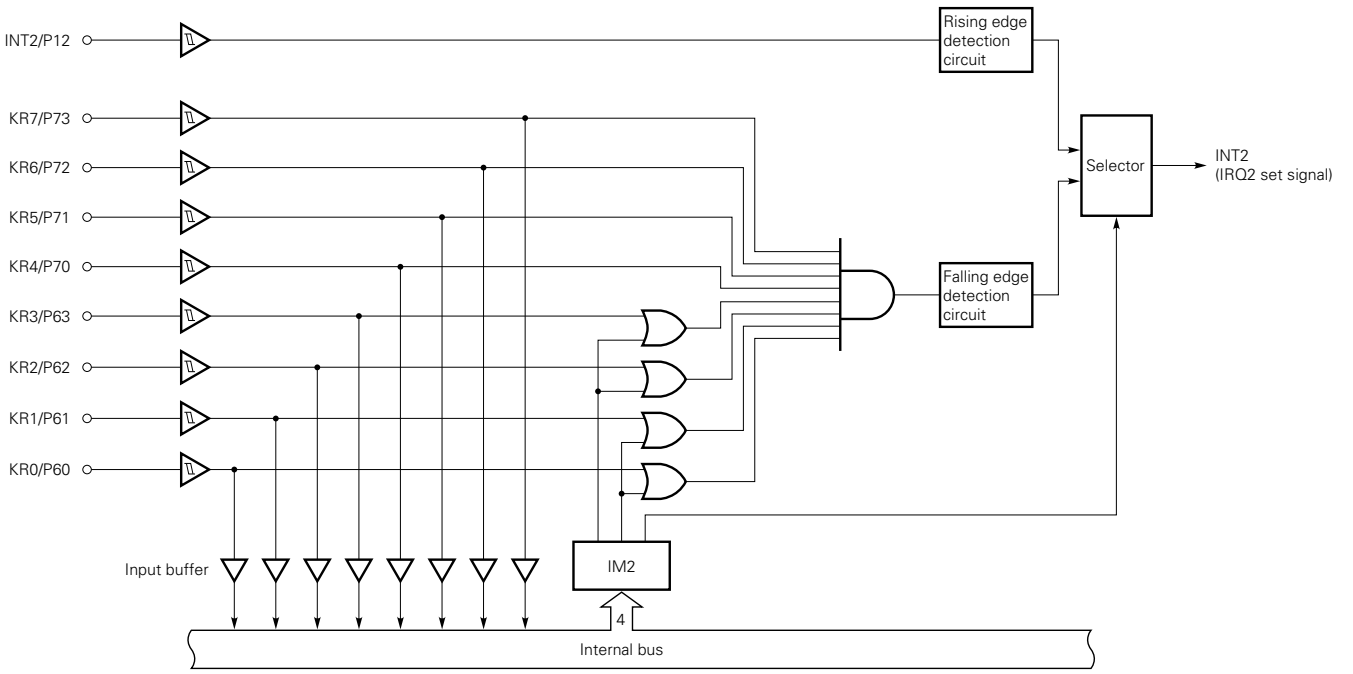
**(4) Interrupt priority specification register (IPS)**

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 a $\overline{\text{RESET}}$ signal is generated, all bits are cleared.

**Caution Disable interrupts before setting the IPS.**

**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 vectored 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 | This status is disabled. | |

**Interrupt master enable flag (IME)**

| | |
|---|---|
| 0 | All interrupts are disabled and no vectored 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.



```
              ( Interrupt (INT×××) occurrence )
                              │
                     ┌─────────────────┐
                     │  IRQ××× setting │
                     └─────────────────┘
                              │
                         ◇ IE××× set? ◇──── NO ──→ ┌─────────────────────┐
                              │                     │ Hold until IE××× is set. │
                             YES                    └─────────────────────┘
                              │
                     ┌──────────────────────────┐
                     │ Corresponding VRQn occurrence │
                     └──────────────────────────┘
                              │
                         ◇ IME = 1 ◇──── NO ──→ ┌────────────────┐
                              │                  │ Hold until IME │
                             YES                 │ is set.        │
                              │                  └────────────────┘
                   ◇ Is VRQn high-order interrupt? ◇──── NO ──→ [Hold until processing being executed is finished.]
                              │
                             YES
                              │
         ◇ Note 1 IST1, 0 = 00 or 01 ◇── NO ──→ ◇ Note 1 IST1, 0 = 00 ◇──── NO
                              │                          │
                             YES                        YES
                              │                          │
                              │            ┌────────────────────────────────┐
                              │            │ If two or more VRQns occur,     │
                              │            │ select one VRQn according to    │
                              │            │ **Table 5-1**.                  │
                              │            └────────────────────────────────┘
                              │             Selected VRQn    Remaining VRQns
                              │
  ┌──────────────────────────────────────────────────────────────────────┐
  │ Save contents of PC and PSW in stack memory and set data^Note 2 in    │
  │ vector table corresponding to activated VRQn to PC, RBE, and MBE.     │
  └──────────────────────────────────────────────────────────────────────┘
                              │
  ┌──────────────────────────────────────────┐
  │ Change contents of IST0 and IST1 from 00 │
  │ to 01 or from 01 to 10.                  │
  └──────────────────────────────────────────┘
                              │
  ┌──────────────────────────────────────────┐
  │ Reset accepted IRQ×××.                   │
  │ (See **Section 5.5** when those interrupt │
  │ sources share vector address.)           │
  └──────────────────────────────────────────┘
                              │
  ┌──────────────────────────────────────────────────────────────┐
  │ Jump to the start address for processing the interrupt service │
  │ program.                                                       │
  └──────────────────────────────────────────────────────────────┘
```

**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 $\mu$PD75518(A) can handle multiple interrupts by either of the following methods.

**(1) Multiple interrupt processing by a high-order interrupt**

In this method, the $\mu$PD75518(A) 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**.)

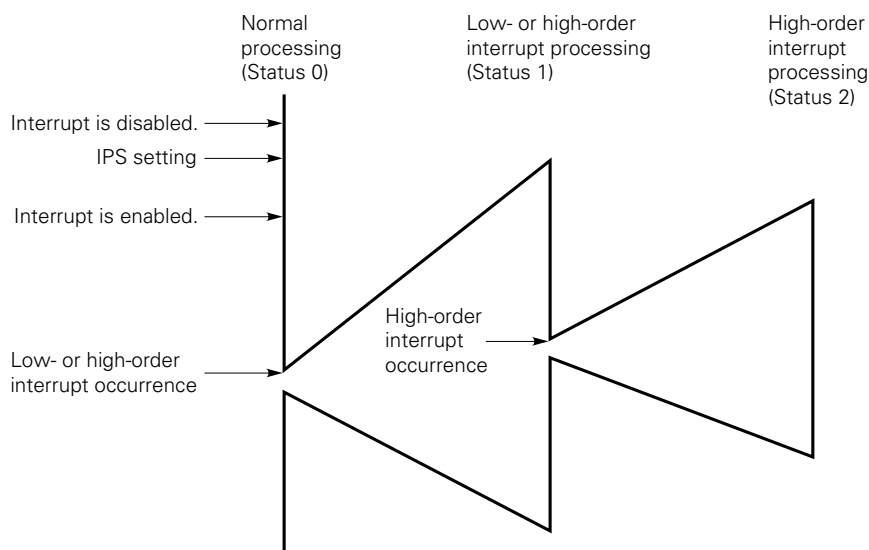**Fig. 5-6  Multiple Interrupt Processing by a High-Order  Interrupt**



**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. | No | – | – |
| 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**
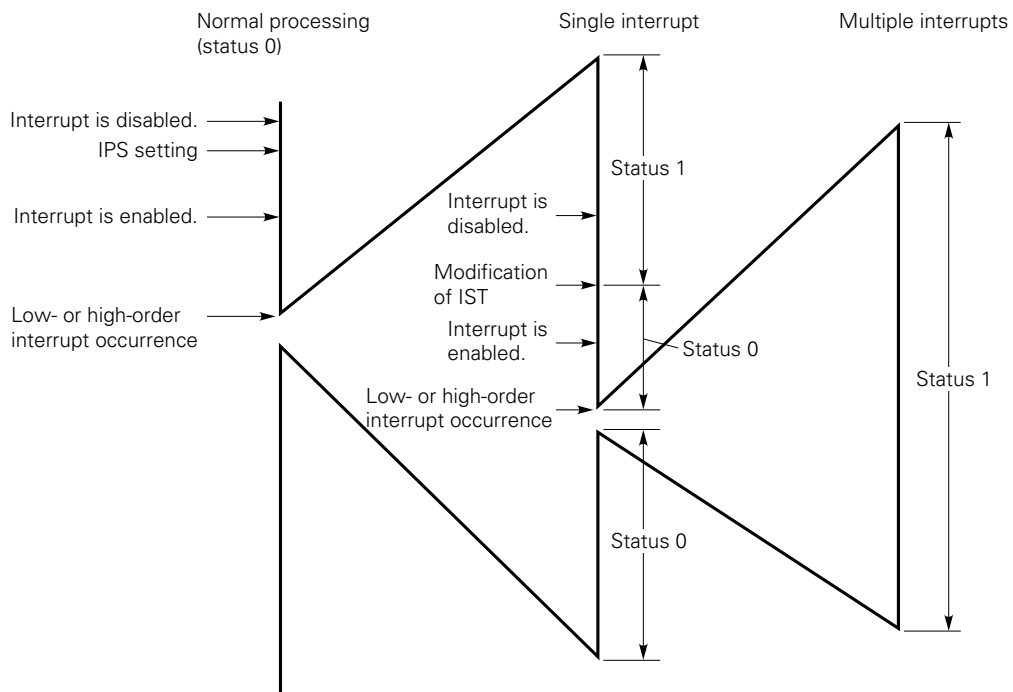
### 5.5 VECTOR ADDRESS SHARE INTERRUPT PROCESSING

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$\times\times\times$ = 1) issues an interrupt request. If this request is accepted, the corresponding interrupt request flag is reset.

**(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.

**Remark** When only one interrupt is enabled, its interrupt source can be clearly identified, so that the interrupt request flag is reset by hardware at acceptance of the interrupt. However, when both interrupts are enabled, the interrupt source cannot be identified, so that the interrupt request flags cannot be reset by hardware. For this reason, the interrupt request flags are checked by software to determine the interrupt source.

## 6. STANDBY FUNCTION

To reduce the power consumption when the program is in the wait state, the $\mu$PD75518(A) 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 $\Phi$ is stopped (with oscillation continued). |
| | Basic interval timer | Operation is stopped. | Operation is continued (to set IRQBT at reference time intervals). |
| | Serial interface (Channel 0) | Operation is possible only when external $\overline{\text{SCK0}}$ input is selected for the serial clock. | Operation is possible only when the main system clock operates or external $\overline{\text{SCK0}}$ is used. |
| | Serial interface (Channel 1) | Operation is possible only when external $\overline{\text{SCK1}}$ input is selected for the serial clock. | Operation is possible only when the main system clock operates. |
| | Timer/event counter | Operation is possible only when TI0 pin input is selected for the count clock. | Operation is possible only when the main system clock operates. |
| | Watch timer | Operation is possible only when $f_{XT}$ is selected for the count clock. | Operation is possible. |
| | A/D converter | Operation is stopped. | Operation is possible only when the main system clock operates. |
| | Timer/pulse generator | Operation is stopped. | Operation is possible only when the main system clock operates. |
| | External interrupt | INT0 is disabled.  INT1, INT2, and INT4 are enabled. | |
| | CPU | Operation is stopped. | |
| Release signal | | Interrupt request signals transmitted from hardware, which are enabled by interrupt enable flags, or $\overline{\text{RESET}}$ 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.) A STOP instruction or HALT instruction must always be followed by an NOP instruction.

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. 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 GND (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. Reset all the interrupt request flags before setting the standby mode.**
> **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 (see Fig. 5-1). When the STOP mode is set, however, the $\mu$PD75518(A) 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 that is enabled with the interrupt enable flag. Fig. 6-1 shows how the STOP and HALT modes are released.

**Fig. 6-1 Standby Mode Release Operation**

**(a) Release of the STOP mode by RESET input**

Wait
(approximately 21.8 ms/6.0 MHz) **Note**

STOP instruction

RESET signal

| Operating mode | STOP mode | HALT mode | Operating mode |

Clock | Oscillation | No oscillation | Oscillation |
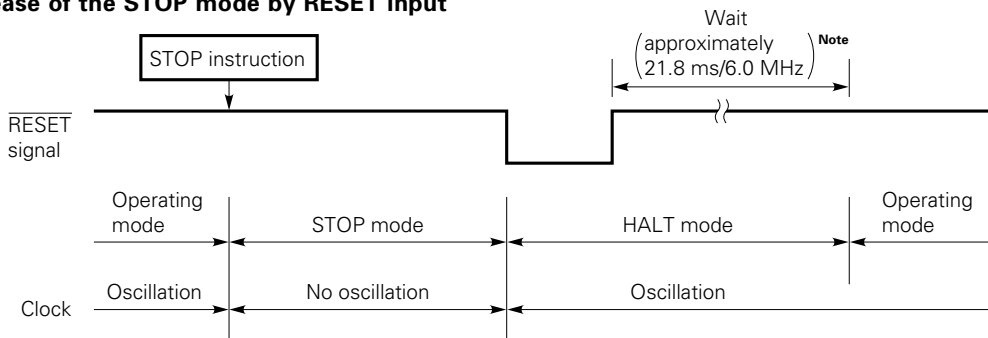
**Note** A wait time is 31.3 ms when operating at 4.19 MHz.

**(b) Release of the STOP mode by the occurrence of an interrupt**
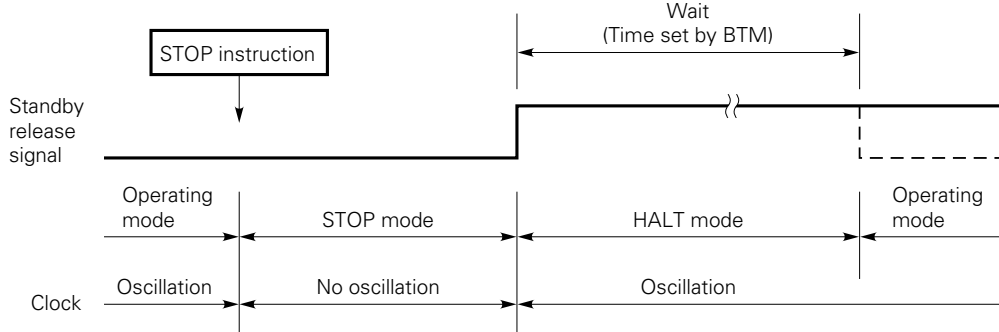
Wait
(Time set by BTM)

STOP instruction

Standby release signal

| Operating mode | STOP mode | HALT mode | Operating mode |

Clock | Oscillation | No oscillation | Oscillation |

**Remark** The dashed line indicates the case where the interrupt request that releases the standby mode is accepted.

**(c) Release of the HALT mode by RESET input**

Wait
(approximately 21.8 ms/6.0 MHz) **Note**

HALT instruction

RESET signal

| Operating mode | HALT mode | Operating mode |

Clock | Oscillation |

**Note** A wait time is 31.3 ms when operating at 4.19 MHz.

**(d) Release of the HALT mode by the occurrence of an interrupt**

HALT instruction

Standby release signal

| Operating mode | HALT mode | Operating mode |

Clock | Oscillation |

**Remark** The dashed line indicates the case where the interrupt request that releases the standby mode is accepted.

The wait times used when the STOP mode is released do not include a time (a in **Fig. 6-2**) 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.

**Fig. 6-2  Start of Clock Generation**
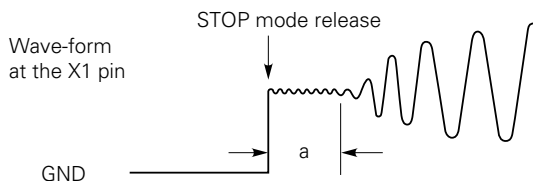


When the STOP mode is released by the occurrence of an interrupt, a wait time is determined by BTM. (See **Table 6-2**.)

**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. $2^{20}/f_x$ (Approx. 175 ms) |
| – | 0 | 1 | 1 | Approx. $2^{17}/f_x$ (Approx. 21.8 ms) |
| – | 1 | 0 | 1 | Approx. $2^{15}/f_x$ (Approx. 5.46 ms) |
| – | 1 | 1 | 1 | Approx. $2^{13}/f_x$ (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 | 1 | Approx. $2^{20}/f_x$ (Approx. 250 ms) |
| – | 0 | 1 | 1 | Approx. $2^{17}/f_x$ (Approx. 31.3 ms) |
| – | 1 | 0 | 1 | Approx. $2^{15}/f_x$ (Approx. 7.82 ms) |
| – | 1 | 1 | 1 | Approx. $2^{13}/f_x$ (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, the interrupt master enable flag (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 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 vectored interrupt is caused. However, when the standby mode is released by INTW or INT2 (input of a testable signal), no vectored interrupt is caused and the same processing as (a) above is performed.

## 7. RESET FUNCTION

The μPD75518(A) is reset by $\overline{\text{RESET}}$ signal input. When reset, the hardware is initialized as indicated in Table 7-1. Fig. 7-1 shows the timing of reset operation.
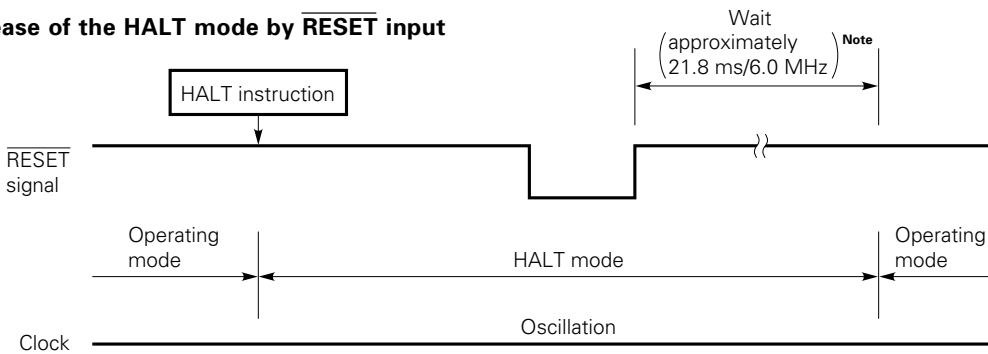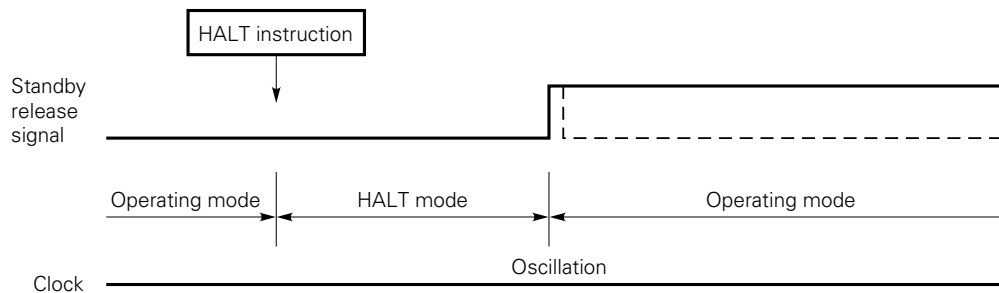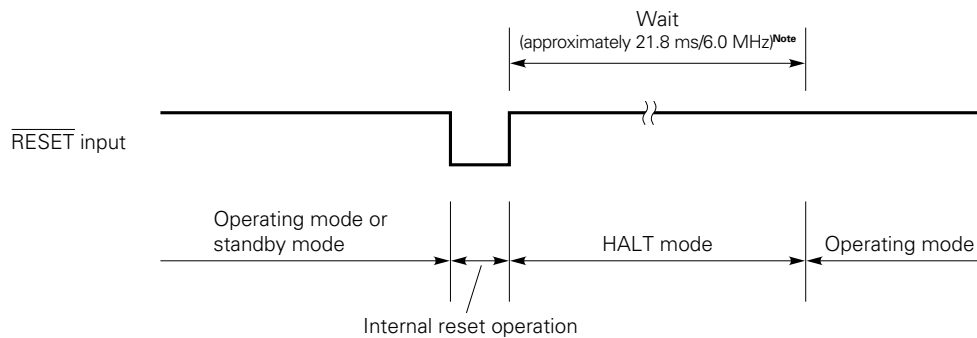
**Fig. 7-1  Reset Operation by $\overline{\text{RESET}}$ input**



**Note**  A wait time is 31.3 ms when operating at 4.19 MHz.

**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 (IST0, IST1) | 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**Note** | 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 |
| Timer/pulse generator | Modulo registers (MODH, MODL) | Held | Held |
| | Mode register (TPGM) | 0 | 0 |
| Watch timer | Mode register (WM) | 0 | 0 |

**Note**  $\overline{\text{RESET}}$ signal input causes data at addresses 0F8H-0FDH in data memory to be undefined.

142

**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 |
|---|---|---|---|
| Serial bus inter-face (Channel 0) | Shift register 0 (SIO0) | Held | Undefined |
| | Operation mode register 0 (CSIM0) | 0 | 0 |
| | SBI control register (SBIC) | 0 | 0 |
| | Slave address register (SVA) | Held | Undefined |
| | P01/$\overline{\text{SCK0}}$ output latch | 1 | 1 |
| A/D converter | Mode register (ADM), EOC | 04H (EOC = 1) | 04H (EOC = 1) |
| | SA register | Undefined | Undefined |
| Clock genera-tor, clock out-put circuit | Processor clock control register (PCC) | 0 | 0 |
| | System clock control register (SCC) | 0 | 0 |
| | Clock output mode register (CLOM) | 0 | 0 |
| Serial interface (Channel 1) | Shift register (SIO1) | Held | Undefined |
| | Operation mode register 1 (CSIM1) | 0 | 0 |
| | Serial transfer end flag (EOT) | 0 | 0 |
| Interrupt | Interrupt request flag (IRQ×××) | Reset (0) | Reset (0) |
| | Interrupt enable flag (IE×××) | 0 | 0 |
| | Interrupt master enable flag (IME) | 0 | 0 |
| | INT0, INT1 and INT2 mode regis-ters (IM0, IM1, IM2) | 0, 0, 0 | 0, 0, 0 |
| Digital ports | Output buffer | Off | Off |
| | Output latch | Clear (0) | Clear (0) |
| | I/O mode registers (PMGA, PMGB, PMGC) | 0 | 0 |
| | Pull-up resistor specification regis-ter (POGA) | 0 | 0 |
| Bit sequential buffers (BSB0 to BSB3) | | Held | Undefined |

## 8. INSTRUCTION SET

### 8.1 $\mu$PD75518(A) 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 1-byte instruction is very useful in reducing the number of program steps.

(a) A subroutine call is made to all the spaces, regarding data in a table as the call address of a call instruction.

(b) A branch is made to all the spaces, regarding data in a table as the branch address of a branch instruction.

(c) Data in a table is executed as a 2-byte or 1-byte instruction.

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. (Except for the BRCB and CALLF instructions)**

**2. When referencing two 1-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<br>DECS<br>INCS<br>DECS<br>INCS | L<br>L<br>H<br>H<br>HL |
| MOV A,@DE<br>XCH A,@DE | INCS<br>DECS<br>INCS<br>DECS<br>INCS | E<br>E<br>D<br>D<br>DE |
| MOV A,@DL<br>XCH A,@DL | INCS<br>DECS<br>INCS<br>DECS | L<br>L<br>D<br>D |

**3. Branch and subroutine instructions can be referenced by the GETI instruction only when their destination addresses 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 1-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 µPD75518(A) 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/EOT IE×××/IRQ××× | FB0H to FBFH |
| | Port 0 to Port 15 | FF0H to FFFH |
| pmem.@L | BSB0 Ports 0, 4, 8, and 12 | 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, W, CSI0, TPG
( 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 $\mu$PD75518(A) 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.

| 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 |
| addr | 0000H-3F7FH immediate data or label |
| addr1 | 0000H-7F7FH 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, IE0, IE1, IE2, IE4, IEW, IETPG |
| 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 |
| 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-3F7FH | 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 ($PC_{14,13,12}$ = 000B) or<br>            1000H-1FFFH ($PC_{14,13,12}$ = 001B) or<br>            2000H-2FFFH ($PC_{14,13,12}$ = 010B) or<br>            3000H-3FFFH ($PC_{14,13,12}$ = 011B) or<br>            4000H-4FFFH ($PC_{14,13,12}$ = 100B) or<br>            5000H-5F7FH ($PC_{14,13,12}$ = 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**Note** is skipped                    : S = 2

   **Note**   3-byte instruction:  BR !addr, BRA !addr1, CALL !addr, and CALLA !addr1 instructions

**Caution   The GETI instruction is skipped in one machine cycle.**

One machine cycle is equal to one cycle of the CPU clock, and three types of times are available for selection according to the PCC setting.

| Instruction | Mnemonic | Operand | Number of bytes | Machine cycle | Operation | Address-ing area | Skip condition |
|---|---|---|---|---|---|---|---|
| Transfer instruc-tion | 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' | | |
| | MOVT | XA,@PCDE | 1 | 3 | XA ← $(PC_{14-8}+DE)_{ROM}$ | | |
| | | XA,@PCXA | 1 | 3 | XA ← $(PC_{14-8}+XA)_{ROM}$ | | |
| | | XA, @BCDE[Note] | 1 | 3 | XA ← $(B_{2-0}+CDE)_{ROM}$ | *11 | |
| | | XA, @BCXA[Note] | 1 | 3 | XA ← $(B_{2-0}+CXA)_{ROM}$ | *11 | |

**Note** Only lower three bits are valid in the B register.

| Instruction | Mnemonic | Operand | Number of bytes | Machine cycle | Operation | Addressing area | Skip condition |
|---|---|---|---|---|---|---|---|
| Bit transfer instruction | MOV1 | CY,fmem.bit | 2 | 2 | $CY \leftarrow (fmem.bit)$ | *4 | |
| | | CY,pmem.@L | 2 | 2 | $CY \leftarrow (pmem_{7-2}+L_{3-2}.bit(L_{1-0}))$ | *5 | |
| | | CY,@H+mem.bit | 2 | 2 | $CY \leftarrow (H+mem_{3-0}.bit)$ | *1 | |
| | | fmem.bit,CY | 2 | 2 | $(fmem.bit) \leftarrow CY$ | *4 | |
| | | pmem.@L,CY | 2 | 2 | $(pmem_{7-2}+L_{3-2}.bit(L_{1-0})) \leftarrow CY$ | *5 | |
| | | @H+mem.bit,CY | 2 | 2 | $(H+mem_{3-0}.bit) \leftarrow CY$ | *1 | |
| Arithmetic/logical instruction | ADDS | A,#n4 | 1 | 1 + S | $A \leftarrow A + n4$ | | carry |
| | | XA,#n8 | 2 | 2 + S | $XA \leftarrow XA + n8$ | | carry |
| | | A,@HL | 1 | 1 + S | $A \leftarrow A + (HL)$ | *1 | carry |
| | | XA,rp' | 2 | 2 + S | $XA \leftarrow XA + rp'$ | | carry |
| | | rp'1,XA | 2 | 2 + S | $rp'1 \leftarrow rp'1 + XA$ | | carry |
| | ADDC | A,@HL | 1 | 1 | $A,CY \leftarrow A + (HL) + CY$ | *1 | |
| | | XA,rp' | 2 | 2 | $XA,CY \leftarrow XA + rp' + CY$ | | |
| | | rp'1,XA | 2 | 2 | $rp'1,CY \leftarrow rp'1 + XA + CY$ | | |
| | SUBS | A,@HL | 1 | 1 + S | $A \leftarrow A - (HL)$ | *1 | borrow |
| | | XA,rp' | 2 | 2 + S | $XA \leftarrow XA - rp'$ | | borrow |
| | | rp'1,XA | 2 | 2 + S | $rp'1 \leftarrow rp'1 - XA$ | | borrow |
| | SUBC | A,@HL | 1 | 1 | $A,CY \leftarrow A - (HL) - CY$ | *1 | |
| | | XA,rp' | 2 | 2 | $XA,CY \leftarrow XA - rp' - CY$ | | |
| | | rp'1,XA | 2 | 2 | $rp'1,CY \leftarrow rp'1 - XA - CY$ | | |
| | AND | A,#n4 | 2 | 2 | $A \leftarrow A \wedge n4$ | | |
| | | A,@HL | 1 | 1 | $A \leftarrow A \wedge (HL)$ | *1 | |
| | | XA,rp' | 2 | 2 | $XA \leftarrow XA \wedge rp'$ | | |
| | | rp'1,XA | 2 | 2 | $rp'1 \leftarrow rp'1 \wedge XA$ | | |
| | OR | A,#n4 | 2 | 2 | $A \leftarrow A \vee n4$ | | |
| | | A,@HL | 1 | 1 | $A \leftarrow A \vee (HL)$ | *1 | |
| | | XA,rp' | 2 | 2 | $XA \leftarrow XA \vee rp'$ | | |
| | | rp'1,XA | 2 | 2 | $rp'1 \leftarrow rp'1 \vee XA$ | | |
| | XOR | A,#n4 | 2 | 2 | $A \leftarrow A \veebar n4$ | | |
| | | A,@HL | 1 | 1 | $A \leftarrow A \veebar (HL)$ | *1 | |
| | | XA,rp' | 2 | 2 | $XA \leftarrow XA \veebar rp'$ | | |
| | | rp'1,XA | 2 | 2 | $rp'1 \leftarrow rp'1 \veebar XA$ | | |

| Instruction | Mnemonic | Operand | Number of bytes | Machine cycle | Operation | Address-ing area | Skip condition |
|---|---|---|---|---|---|---|---|
| Accumulator manipulation instruction | RORC | A | 1 | 1 | $CY \leftarrow A_0, A_3 \leftarrow CY, A_{n-1} \leftarrow A_n$ | | |
| | NOT | A | 2 | 2 | $A \leftarrow \overline{A}$ | | |
| Increment/ decrement instruction | INCS | reg | 1 | 1 + S | $reg \leftarrow reg + 1$ | | reg = 0 |
| | | rp1 | 1 | 1 + S | $rp1 \leftarrow rp1 + 1$ | | rp1 = 00H |
| | | @HL | 2 | 2 + S | $(HL) \leftarrow (HL) + 1$ | *1 | (HL) = 0 |
| | | mem | 2 | 2 + S | $(mem) \leftarrow (mem) + 1$ | *3 | (mem) = 0 |
| | DECS | reg | 1 | 1 + S | $reg \leftarrow reg - 1$ | | reg = FH |
| | | rp' | 2 | 2 + S | $rp' \leftarrow rp' - 1$ | | rp' = FFH |
| Comparison instruction | 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 instruction | SET1 | CY | 1 | 1 | $CY \leftarrow 1$ | | |
| | CLR1 | CY | 1 | 1 | $CY \leftarrow 0$ | | |
| | SKT | CY | 1 | 1 + S | Skip if CY = 1 | | CY = 1 |
| | NOT1 | CY | 1 | 1 | $CY \leftarrow \overline{CY}$ | | |

| Instruction | Mnemonic | Operand | Number of bytes | Machine cycle | Operation | Addressing area | Skip condition |
|---|---|---|---|---|---|---|---|
| Memory bit manipulation instruction | 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 instruction | BR | addr1 | — | — | $PC_{14-0} ← addr1$ (The assembler selects an appropriate instruction from the BR !addr, BRA !addr1, BRCB !caddr, and BR $addr instructions.) | *11 | |
| | | $addr | 1 | 2 | $PC_{14-0} ← addr$ | *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[Note] | 2 | 3 | $PC_{14-0} ← B_{2-0} + CDE$ | *11 | |
| | | BCXA[Note] | 2 | 3 | $PC_{14-0} ← B_{2-0} + CXA$ | *11 | |
| | BRA | !addr1 | 3 | 3 | $PC_{14-0} ← addr1$ | *11 | |
| | BRCB | !caddr | 2 | 2 | $PC_{14-0} ← PC_{14,13,12} + caddr_{11-0}$ | *8 | |

**Note** Only lower three bits are valid in the B register.

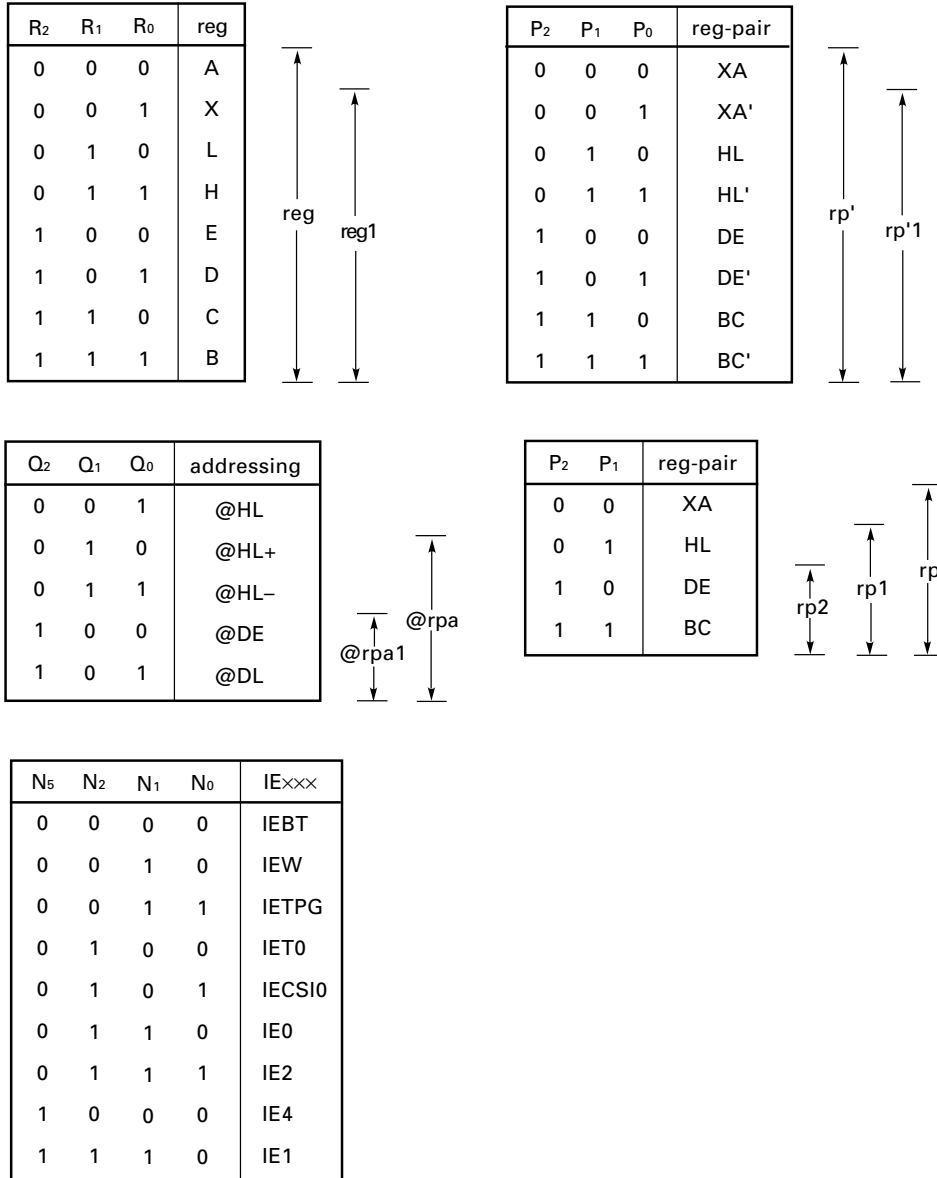| Instruction | Mnemonic | Operand | Number of bytes | Machine cycle | Operation | Addressing area | Skip condition |
|---|---|---|---|---|---|---|---|
| Subroutine stack control instruction | CALL | !addr | 3 | 4 | $(SP-2) \leftarrow \times,\times,MBE,RBE$<br>$(SP-6)(SP-3)(SP-4) \leftarrow PC_{11-0}$<br>$(SP-5) \leftarrow 0,PC_{14},PC_{13}, PC_{12}$<br>$PC_{14} \leftarrow 0, PC_{13-0} \leftarrow addr, SP \leftarrow SP - 6$ | *6 | |
| | CALLA | !addr1 | 3 | 3 | $(SP-2) \leftarrow \times,\times,MBE,RBE$<br>$(SP-6)(SP-3)(SP-4) \leftarrow PC_{11-0}$<br>$(SP-5) \leftarrow 0,PC_{14},PC_{13}, PC_{12}$<br>$PC_{14-0} \leftarrow addr, SP \leftarrow SP - 6$ | *11 | |
| | CALLF | !faddr | 2 | 3 | $(SP-2) \leftarrow \times,\times,MBE,RBE$<br>$(SP-6)(SP-3)(SP-4) \leftarrow PC_{11-0}$<br>$(SP-5) \leftarrow 0,PC_{14},PC_{13}, PC_{12}$<br>$PC_{14-0} \leftarrow 0000 + faddr, SP \leftarrow SP - 6$ | *9 | |
| | RET | | 1 | 3 | $PC_{11-0} \leftarrow (SP)(SP+3)(SP+2)$<br>$\times,PC_{14},PC_{13},PC_{12} \leftarrow (SP+1)$<br>$\times,\times,MBE,RBE \leftarrow (SP+4)$<br>$SP \leftarrow SP + 6$ | | |
| | RETS | | 1 | 3 + S | $PC_{11-0} \leftarrow (SP)(SP+3)(SP+2)$<br>$\times,PC_{14},PC_{13},PC_{12} \leftarrow (SP+1)$<br>$\times,\times,MBE,RBE \leftarrow (SP+4)$<br>$SP \leftarrow SP + 6$<br>then skip unconditionally | | Unconditionally |
| | RETI | | 1 | 3 | $PC_{11-0} \leftarrow (SP)(SP+3)(SP+2)$<br>$\times,PC_{14},PC_{13},PC_{12} \leftarrow (SP+1)$<br>$PSW \leftarrow (SP+4)(SP+5), SP \leftarrow SP + 6$ | | |
| | PUSH | rp | 1 | 1 | $(SP-1)(SP-2) \leftarrow rp, SP \leftarrow SP - 2$ | | |
| | | BS | 2 | 2 | $(SP-1) \leftarrow MBS, (SP-2) \leftarrow RBS, SP \leftarrow SP - 2$ | | |
| | POP | rp | 1 | 1 | $rp \leftarrow (SP+1)(SP), SP \leftarrow SP + 2$ | | |
| | | BS | 2 | 2 | $MBS \leftarrow (SP+1), RBS \leftarrow (SP), SP \leftarrow SP + 2$ | | |
| Interrupt control instruction | EI | | 2 | 2 | $IME(IPS.3) \leftarrow 1$ | | |
| | | IE$\times\times\times$ | 2 | 2 | $IE\times\times\times \leftarrow 1$ | | |
| | DI | | 2 | 2 | $IME(IPS.3) \leftarrow 0$ | | |
| | | IE$\times\times\times$ | 2 | 2 | $IE\times\times\times \leftarrow 0$ | | |
| I/O instruction | IN**Note** | A,PORTn | 2 | 2 | $A \leftarrow PORTn$ (n=0-15) | | |
| | | XA,PORTn | 2 | 2 | $XA \leftarrow PORTn_{+1},PORTn$ (n=4,6) | | |
| | OUT**Note** | PORTn,A | 2 | 2 | $PORTn \leftarrow A$ (n=2-7,9-14) | | |
| | | PORTn,XA | 2 | 2 | $PORTn_{+1},PORTn \leftarrow XA$ (n=4,6) | | |
| CPU control instruction | HALT | | 2 | 2 | Set HALT Mode (PCC.2 $\leftarrow$ 1) | | |
| | STOP | | 2 | 2 | Set STOP Mode (PCC.3 $\leftarrow$ 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 | Address- ing area | Skip condition |
|---|---|---|---|---|---|---|---|
| Special instruc- tion | 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> $PC_{13-0}$ ← $(taddr)_{4-0}$ + (taddr+1) <br> $PC_{14}$ ← 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> $PC_{13-0}$ ← $(taddr)_{5-0}$ + (taddr+1) <br> SP ← SP-6  $PC_{14}$ ← 0 | | |
| | | | | 3 | • For an instruction other than TBR and TCALL <br> (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 |

reg   reg1

| $P_2$ | $P_1$ | $P_0$ | 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' |

rp'   rp'1

| $Q_2$ | $Q_1$ | $Q_0$ | addressing |
|----|----|----|------------|
| 0 | 0 | 1 | @HL |
| 0 | 1 | 0 | @HL+ |
| 0 | 1 | 1 | @HL− |
| 1 | 0 | 0 | @DE |
| 1 | 0 | 1 | @DL |

@rpa1   @rpa

| $P_2$ | $P_1$ | reg-pair |
|----|----|----------|
| 0 | 0 | XA |
| 0 | 1 | HL |
| 1 | 0 | DE |
| 1 | 1 | BC |

rp2   rp1   rp

| $N_5$ | $N_2$ | $N_1$ | $N_0$ | 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 | 1 | 1 | 0 | IE1 |

$I_n$ : Immediate data for n4 or n8

$D_n$: Immediate data for mem

$B_n$: Immediate data for bit

$N_n$: Immediate data for n or IE×××

$T_n$: Immediate data for taddr × 1/2

$A_n$: Immediate data for the relative address distance (2 to 16) for the branch destination address minus one

$S_n$: 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**

$\boxed{*1}$ 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 $\boxed{*2}$ 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 |

$B_n$ : Immediate data for bit

$F_n$ : Immediate data for fmem (Low-order four bits of address)

$G_n$ : Immediate data for pmem (Bits 2 to 5 of address)

$D_n$ : Immediate data for mem (Low-order four bits of address)

| Instruc-tion | Mnemonic | Operand | Instruction code | | |
|---|---|---|---|---|---|
| | | | B₁ | B₂ | B₃ |
| Transfer instruc-tion | MOV | A, #n4 | 0 1 1 1 I₃ I₂ I₁ I₀ | | |
| | | reg1, #n4 | 1 0 0 1 1 0 1 0 | I₃ I₂ I₁ I₀ 1 R₂ R₁ R₀ | |
| | | rp, #n8 | 1 0 0 0 1 P₂ P₁ 1 | I₇ I₆ I₅ I₄ I₃ I₂ I₁ I₀ | |
| | | A, @rpa | 1 1 1 0 0 Q₂ Q₁ Q₀ | | |
| | | 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 | D₇ D₆ D₅ D₄ D₃ D₂ D₁ D₀ | |
| | | XA, mem | 1 0 1 0 0 0 1 0 | D₇ D₆ D₅ D₄ D₃ D₂ D₁ 0 | |
| | | mem, A | 1 0 0 1 0 0 1 1 | D₇ D₆ D₅ D₄ D₃ D₂ D₁ D₀ | |
| | | mem, XA | 1 0 0 1 0 0 1 0 | D₇ D₆ D₅ D₄ D₃ D₂ D₁ 0 | |
| | | A, reg | 1 0 0 1 1 0 0 1 | 0 1 1 1 1 R₂ R₁ R₀ | |
| | | XA, rp' | 1 0 1 0 1 0 1 0 | 0 1 0 1 1 P₂ P₁ P₀ | |
| | | reg1, A | 1 0 0 1 1 0 0 1 | 0 1 1 1 0 R₂ R₁ R₀ | |
| | | rp'1, XA | 1 0 1 0 1 0 1 0 | 0 1 0 1 0 P₂ P₁ P₀ | |
| | XCH | A, @rpa | 1 1 1 0 1 Q₂ Q₁ Q₀ | | |
| | | 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 | D₇ D₆ D₅ D₄ D₃ D₂ D₁ D₀ | |
| | | XA, mem | 1 0 1 1 0 0 1 0 | D₇ D₆ D₅ D₄ D₃ D₂ D₁ 0 | |
| | | A, reg1 | 1 1 0 1 1 R₂ R₁ R₀ | | |
| | | XA, rp' | 1 0 1 0 1 0 1 0 | 0 1 0 0 0 P₂ P₁ P₀ | |
| | 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 instruction | 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 instruction | 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 instruction | 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 B1 | Instruction code B2 | Instruction code B3 |
|---|---|---|---|---|---|
| Increment/decrement instruction | INCS | reg | 1 1 0 0 0 $R_2$ $R_1$ $R_0$ | | |
| | | rp1 | 1 0 0 0 1 $P_2$ $P_1$ $P_0$ | | |
| | | @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 instruction | 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 instruction | 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 instruction | 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 B1 | B2 | B3 |
|---|---|---|---|---|---|
| Branch instruction | BR | !addr | 1 0 1 0 1 0 1 1 | 0 0 ←————— addr | addr —————→ |
| | | $addr (+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 instruction | 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 ←——— 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 instruction | 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 instruction | 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 instruction | 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 instruction | 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. ELECTRICAL CHARACTERISTICS

**ABSOLUTE MAXIMUM RATINGS** ($T_a$ = 25 °C)
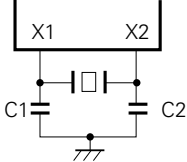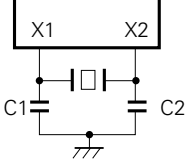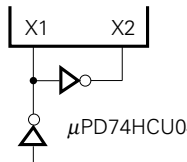
| Parameter | Symbol | Conditions | | | Rated value | Unit |
|---|---|---|---|---|---|---|
| Supply voltage | $V_{DD}$ | | | | −0.3 to +7.0 | V |
| Input voltage | $V_{I1}$ | Ports other than ports 4, 5, and 12 to 14 | | | −0.3 to $V_{DD}$ + 0.3 | V |
| | $V_{I2}$ | Ports 4, 5, and 12 to 14 | Built-in pull-up resistor | | −0.3 to $V_{DD}$ + 0.3 | V |
| | | | Open drain | | −0.3 to +13 | V |
| Output voltage | $V_O$ | | | | −0.3 to $V_{DD}$ + 0.3 | V |
| High-level output current | $I_{OH}$ | Each pin | | | −15 | mA |
| | | Total of all pins | | | −30 | mA |
| Low-level output current | $I_{OL}$**Note** | 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 11 | Peak value | | 100 | mA |
| | | | rms | | 60 | mA |
| | | Total of all pins of ports 12 to 14 | Peak value | | 40 | mA |
| | | | rms | | 25 | mA |
| Operating temperature | $T_{opt}$ | | | | −40 to +85 | ˚C |
| Storage temperature | $T_{stg}$ | | | | −65 to +150 | ˚C |

**Note** Calculate rms with [rms] = [peak value] $\times \sqrt{\text{duty}}$.

**OPERATING SUPPLY VOLTAGE**
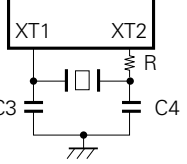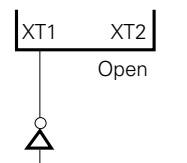
| Parameter | | Symbol | Min. | Max. | Unit | Conditions |
|---|---|---|---|---|---|---|
| A/D converter | Supply voltage | $V_{DD}$ | 2.7 | 6.0 | V | |
| | Ambient temperature | $T_a$ | −40 | +85 | ˚C | |
| Timer/pulse generator | Supply voltage | $V_{DD}$ | 4.5 | 6.0 | V | |
| | Ambient temperature | $T_a$ | −40 | +85 | ˚C | |
| Other circuits | Supply voltage | $V_{DD}$ | 2.7 | 6.0 | V | |
| | Ambient temperature | $T_a$ | −40 | +85 | ˚C | |

**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 | X1 X2 C1 C2 | Oscillator frequency ($f_X$) **Note 1** | 1.0 | | 6.2 | MHz | $V_{DD}$ = oscillation voltage range |
| | | Oscillation settling time **Note 2** | | | 4 | ms | After $V_{DD}$ reaches Min. of the oscillation voltage range |
| Crystal resonator | X1 X2 C1 C2 | Oscillator frequency ($f_X$) **Note 1** | 1.0 | 4.19 | 6.2 | MHz | |
| | | Oscillation settling time **Note 2** | | | 10 | ms | $V_{DD}$ = 4.5 to 6.0 V |
| | | | | | 30 | ms | |
| External clock | X1 X2 μPD74HCU04 | X1 input frequency ($f_X$) **Note 1** | 1.0 | | 6.2 | MHz | |
| | | X1 input high/low level width ($t_{XH}$, $t_{XL}$) | 81 | | 500 | 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 settling time means the time required for the oscillation to settle 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 | XT1 XT2 R C3 C4 | Oscillator frequency ($f_{XT}$) **Note 1** | 32 | 32.768 | 35 | kHz | |
| | | Oscillation settling time **Note 2** | | 1.0 | 2 | s | $V_{DD}$ = 4.5 to 6.0 V |
| | | | | | 10 | s | |
| External clock | XT1 XT2 Open | 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 settling time means the time required for the oscillation to settle 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 |
| Output capacitance | $C_O$ | | | 15 | pF | 0 V for pins other than pins to be measured |
| I/O capacitance | $C_{IO}$ | | | 15 | 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$_{IH1}$ | 0.7V$_{DD}$ | | V$_{DD}$ | V | Ports 2, 3, and 9 to 11, P80, and P82 | |
| | V$_{IH2}$ | 0.8V$_{DD}$ | | V$_{DD}$ | V | Ports 0, 1, 6, 7, and 15, P81, P83, and $\overline{\text{RESET}}$ | |
| | V$_{IH3}$ | 0.7V$_{DD}$ | | V$_{DD}$ | V | Ports 4, 5, and 12 to 14 | Built-in pull-up resistor |
| | | 0.7V$_{DD}$ | | 10 | V | | Open drain |
| | V$_{IH4}$ | V$_{DD}$ − 0.5 | | V$_{DD}$ | V | X1, X2, and XT1 | |
| Low-level input voltage | V$_{IL1}$ | 0 | | 0.3V$_{DD}$ | V | Ports 2 to 5 and 9 to 14, P80, and P82 | |
| | V$_{IL2}$ | 0 | | 0.2V$_{DD}$ | V | Ports 0, 1, 6, 7, and 15, P81, P83, and $\overline{\text{RESET}}$ | |
| | V$_{IL3}$ | 0 | | 0.4 | V | X1, X2, and XT1 | |
| High-level output voltage | V$_{OH}$ | V$_{DD}$ − 1.0 | | | V | V$_{DD}$ = 4.5 to 6.0 V, I$_{OH}$ = −1 mA | |
| | | V$_{DD}$ − 0.5 | | | 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 | V$_{DD}$ = 4.5 to 6.0 V, I$_{OL}$ = 1.6 mA | |
| | | | | 0.5 | V | I$_{OL}$ = 400 μA | |
| | | | | 0.2V$_{DD}$ | V | SB0 and SB1 | Open drain Pull-up resistor: 1 kΩ or more |
| High-level input leakage current | I$_{LIH1}$ | | | 3 | μA | V$_I$ = V$_{DD}$ | Other than X1, X2, and XT1 |
| | I$_{LIH2}$ | | | 20 | μA | | X1, X2, and XT1 |
| | I$_{LIH3}$ | | | 20 | μA | V$_I$ = 10 V | Ports 4, 5, and 12 to 14 (open drain) |
| Low-level input leakage current | I$_{LIL1}$ | | | −3 | μA | V$_I$ = 0 V | Other than X1, X2, and XT1 |
| | I$_{LIL2}$ | | | −20 | μA | | X1, X2, and XT1 |
| High-level output leakage current | I$_{LOH1}$ | | | 3 | μA | V$_O$ = V$_{DD}$ | Other than ports 4, 5, and 12 to 14 |
| | I$_{LOH2}$ | | | 20 | μA | V$_O$ = 10 V | Ports 4, 5, and 12 to 14 (open drain) |
| Low-level output leakage current | I$_{LOL}$ | | | −3 | μA | V$_O$ = 0 V | |
| Built-in pull-up resistor | R$_{U1}$ | 15 | 40 | 80 | kΩ | Ports 0, 1, 2, 3, 6, and 7 (excl. P00) V$_I$ = 0 V | V$_{DD}$ = 5.0 V ±10 % |
| | | 30 | | 300 | kΩ | | V$_{DD}$ = 3.0 V ±10 % |
| | R$_{U2}$ | 15 | 40 | 70 | kΩ | Ports 4, 5, and 12 to 14 V$_O$ = V$_{DD}$ − 2.0 V | V$_{DD}$ = 5.0 V ±10 % |
| | | 10 | | 60 | kΩ | | V$_{DD}$ = 3.0 V ±10 % |
| Built-in pull-down resistor | R$_D$ | 20 | 70 | 140 | kΩ | V$_O$ = 2 V | Port 9 |

| Parameter | Symbol | Min. | Typ. | Max. | Unit | Conditions | | |
|---|---|---|---|---|---|---|---|---|
| Power supply current [1] | $I_{DD1}$ | | 4.5 | 13.5 | mA | 6.0 MHz crystal resonance C1 = C2 = 22 pF | Operation mode | $V_{DD}$ = 5 V ±10 %[2] |
| | | | 0.6 | 1.8 | mA | | | $V_{DD}$ = 3 V ±10 %[3] |
| | $I_{DD2}$ | | 700 | 2100 | μA | | HALT mode | $V_{DD}$ = 5 V ±10 % |
| | | | 250 | 750 | μA | | | $V_{DD}$ = 3 V ±10 % |
| | $I_{DD1}$ | | 3 | 9 | mA | 4.19 MHz crystal resonance C1 = C2 = 22 pF | Operation mode | $V_{DD}$ = 5 V ±10 %[3] |
| | | | 0.55 | 1.5 | mA | | | $V_{DD}$ = 3 V ±10 %[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 | 32.768 kHz 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 % | |
| | | | | 5 | μA | | | $T_a$ = 25 °C |

**Notes 1.** This current excludes the current which flows through the built-in pull-up resistors.

    **2.** Value when the processor clock control register (PCC) is set to 0011 and the μPD75518(A) is operated in the high-speed mode

    **3.** Value when the PCC is set to 0000 and the μPD75518(A) is operated in the low-speed mode

**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 | |
|---|---|---|---|---|---|---|---|
| Cycle time (minimum instruction execution time)[Note 1] | $t_{CY}$ | 0.67 | | 64 | $\mu s$ | Operated by main system clock pulse | $V_{DD}$ = 4.5 to 6.0 V |
| | | 2.6 | | 64 | $\mu s$ | | |
| | | 114 | 122 | 125 | $\mu s$ | 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 | | | $\mu s$ | $V_{DD}$ = 4.5 to 6.0 V | |
| | | 1.8 | | | $\mu s$ | | |
| Interrupt input high/low level width | $t_{INTH}$, $t_{INTL}$ | **Note 2** | | | $\mu s$ | INT0 | |
| | | 10 | | | $\mu s$ | INT1, INT2, and INT4 | |
| | | 10 | | | $\mu s$ | KR0 to KR7 | |
| $\overline{RESET}$ low level width | $t_{RSL}$ | 10 | | | $\mu s$ | | |

**Notes 1.** The cycle time (minimum instruction execution time) depends on the connected resonator frequency, the system clock control register (SCC), and the processor clock control register (PCC). The figure on the right side 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).

$t_{CY}$ vs $V_{DD}$
(When the main system clock is operating)
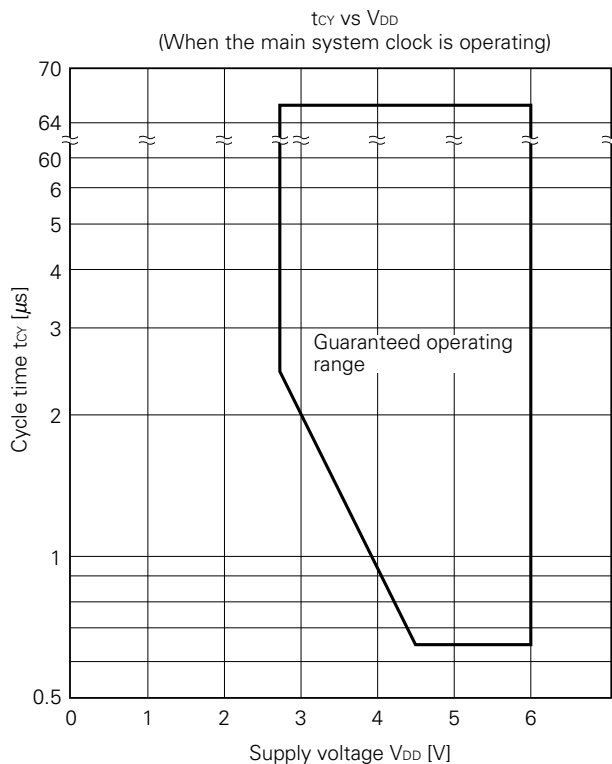
**(2) Serial transfer**

**(a) Two-wire and three-wire serial I/O modes ($\overline{\text{SCK}}$ ... Internal clock output):**

| Parameter | Symbol | Min. | Typ. | Max. | Unit | Conditions | |
|---|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY1}$ | 1600 | | | ns | $V_{DD}$ = 4.5 to 6.0 V  $f_X$ = 4.19 MHz | |
| | | 1340 | | | ns | $V_{DD}$ = 4.5 to 6.0 V  $f_X$ = 6.0 MHz | |
| | | 3800 | | | ns | $f_X$ = 4.19 MHz | |
| | | 2680 | | | ns | $f_X$ = 6.0 MHz | |
| $\overline{\text{SCK}}$ 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 $\overline{\text{SCK}}\uparrow$) | $t_{SIK1}$ | 150 | | | ns | | |
| SI hold time (referred to $\overline{\text{SCK}}\uparrow$) | $t_{KSI1}$ | 400 | | | ns | | |
| $\overline{\text{SCK}}\downarrow \rightarrow$ 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 ($\overline{\text{SCK}}$ ... External clock input):**

| Parameter | Symbol | Min. | Typ. | Max. | Unit | Conditions | |
|---|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY2}$ | 800 | | | ns | $V_{DD}$ = 4.5 to 6.0 V | |
| | | 3200 | | | ns | | |
| $\overline{\text{SCK}}$ 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 $\overline{\text{SCK}}\uparrow$) | $t_{SIK2}$ | 100 | | | ns | | |
| SI hold time (referred to $\overline{\text{SCK}}\uparrow$) | $t_{KSI2}$ | 400 | | | ns | | |
| $\overline{\text{SCK}}\downarrow \rightarrow$ 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 ($\overline{\text{SCK}}$ ... Internal clock output (master)):**

| Parameter | Symbol | Min. | Typ. | Max. | Unit | Conditions | |
|---|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY3}$ | 1600 | | | ns | $V_{DD}$ = 4.5 to 6.0 V   $f_X$ = 4.19 MHz | |
| | | 1340 | | | ns | $V_{DD}$ = 4.5 to 6.0 V   $f_X$ = 6.0 MHz | |
| | | 3800 | | | ns | $f_X$ = 4.19 MHz | |
| | | 2680 | | | ns | $f_X$ = 6.0 MHz | |
| $\overline{\text{SCK}}$ 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 $\overline{\text{SCK}}\uparrow$) | $t_{SIK3}$ | 150 | | | ns | | |
| SB0/SB1 hold time (referred to $\overline{\text{SCK}}\uparrow$) | $t_{KSI3}$ | $t_{KCY}/2$ | | | ns | | |
| $\overline{\text{SCK}}\downarrow \rightarrow$ SB0/SB1 output delay time | $t_{KSO3}$ | 0 | | 250 | ns | $R_L$ = 1 k$\Omega$ , $C_L$ = 100 pF[Note] | $V_{DD}$ = 4.5 to 6.0 V |
| | | 0 | | 1000 | ns | | |
| $\overline{\text{SCK}}\uparrow \rightarrow$ SB0/SB1$\downarrow$ | $t_{KSB}$ | $t_{KCY}$ | | | ns | | |
| SB0/SB1$\downarrow \rightarrow \overline{\text{SCK}}$ | $t_{SBK}$ | $t_{KCY}$ | | | 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 ($\overline{\text{SCK}}$ ... External clock input (slave)):**

| Parameter | Symbol | Min. | Typ. | Max. | Unit | Conditions | |
|---|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY4}$ | 800 | | | ns | $V_{DD}$ = 4.5 to 6.0 V | |
| | | 3200 | | | ns | | |
| $\overline{\text{SCK}}$ 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 $\overline{\text{SCK}}\uparrow$) | $t_{SIK4}$ | 100 | | | ns | | |
| SB0/SB1 hold time (referred to $\overline{\text{SCK}}\uparrow$) | $t_{KSI4}$ | $t_{KCY}/2$ | | | ns | | |
| $\overline{\text{SCK}}\downarrow \rightarrow$ SB0/SB1 output delay time | $t_{KSO4}$ | 0 | | 300 | ns | $R_L$ = 1 k$\Omega$ , $C_L$ = 100 pF[Note] | $V_{DD}$ = 4.5 to 6.0 V |
| | | 0 | | 1000 | ns | | |
| $\overline{\text{SCK}}\uparrow \rightarrow$ SB0/SB1$\downarrow$ | $t_{KSB}$ | $t_{KCY}$ | | | ns | | |
| SB0/SB1$\downarrow \rightarrow \overline{\text{SCK}}\downarrow$ | $t_{SBK}$ | $t_{KCY}$ | | | 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.
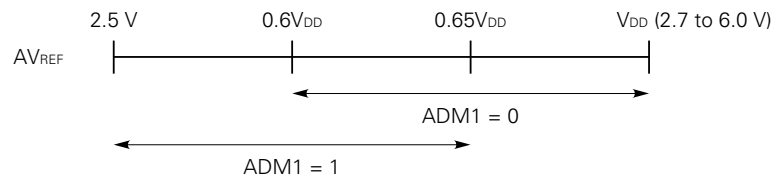
**(3) A/D converter** ($T_a$ = -40 to +85 °C, $V_{DD}$ = 2.7 to 6.0 V, $AV_{SS}$ = $V_{SS}$ = 0 V)

| Parameter | Symbol | Min. | Typ. | Max. | Unit | Conditions | |
|---|---|---|---|---|---|---|---|
| Resolution | | 8 | 8 | 8 | bit | | |
| Absolute accuracy[Note 1] | | | | ±1.5 | LSB | 2.5 V ≤ $AV_{REF}$ ≤ $V_{DD}$[Note 2] | -10 ≤ Ta ≤ +85°C |
| | | | | ±2.0 | | | -40 ≤ Ta < -10°C |
| Conversion time[Note 3] | $t_{CONV}$ | | | 168/fx | $\mu$s | | |
| Sampling time[Note 4] | $t_{SAMP}$ | | | 44/fx | $\mu$s | | |
| Analog input voltage | $V_{IAN}$ | $AV_{SS}$ | | $AV_{REF}$ | V | | |
| Analog input impedance | $R_{AN}$ | | 1000 | | M$\Omega$ | | |
| $V_{AREF}$ current | $I_{AREF}$ | | 1.0 | 2.0 | mA | | |

**Notes 1.** Absolute accuracy excluding quantization error (±1/2 LSB)

　　**2.** 2.5 V ≤ $AV_{REF}$ ≤ $V_{DD}$

　　　ADM1 is set to 0 or 1 depending on the A/D converter reference voltage ($AV_{REF}$) as follows:

```
           2.5 V        0.6VDD        0.65VDD        VDD (2.7 to 6.0 V)
  AVREF     |-------------|-------------|----------------|
                                |<----------------------->|
                                        ADM1 = 0
            |<------------------------------->|
                      ADM1 = 1
```

　　　When 0.6$V_{DD}$ ≤ $AV_{REF}$ ≤ 0.65$V_{DD}$, ADM1 can be set to either 0 or 1.

　　**3.** Time from the execution of a conversion start instruction till the end of conversion (EOC = 1) (28.0 $\mu$s: fx = 6.0 MHz, 40.1 $\mu$s: fx = 4.19 MHz)

　　**4.** Time from the execution of a conversion start instruction till the end of sampling (7.33 $\mu$s: fx = 6.0 MHz, 10.5 $\mu$s: fx = 4.19 MHz)

**AC Timing Measurement Points (Excluding (X1 and XT1 Inputs)**



**Clock Timing**



X1 input

**TI0 Timing**



TI0
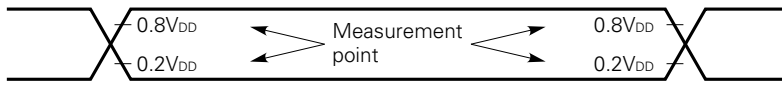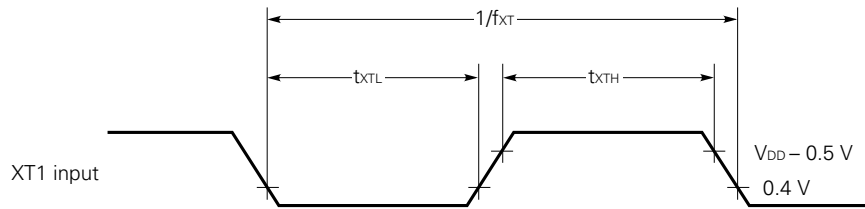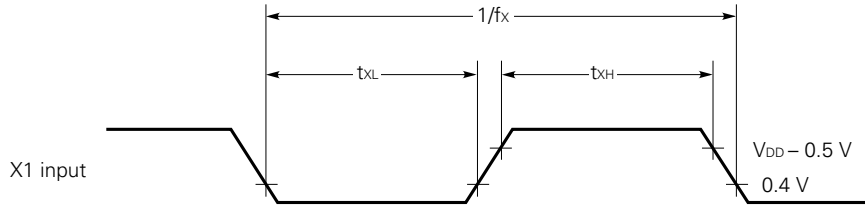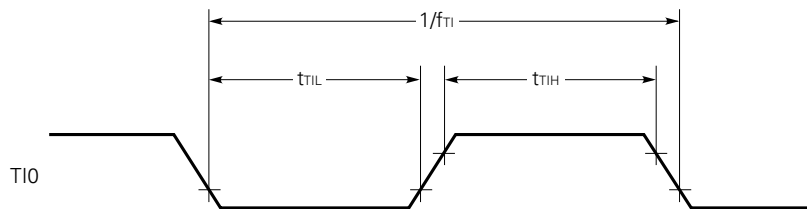
**Serial Transfer Timing**
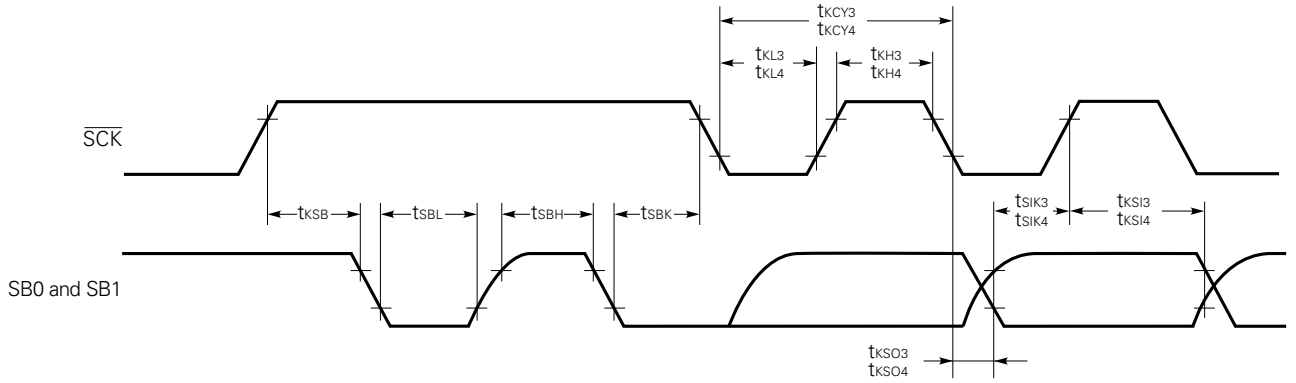
**Three-wire serial I/O mode:**
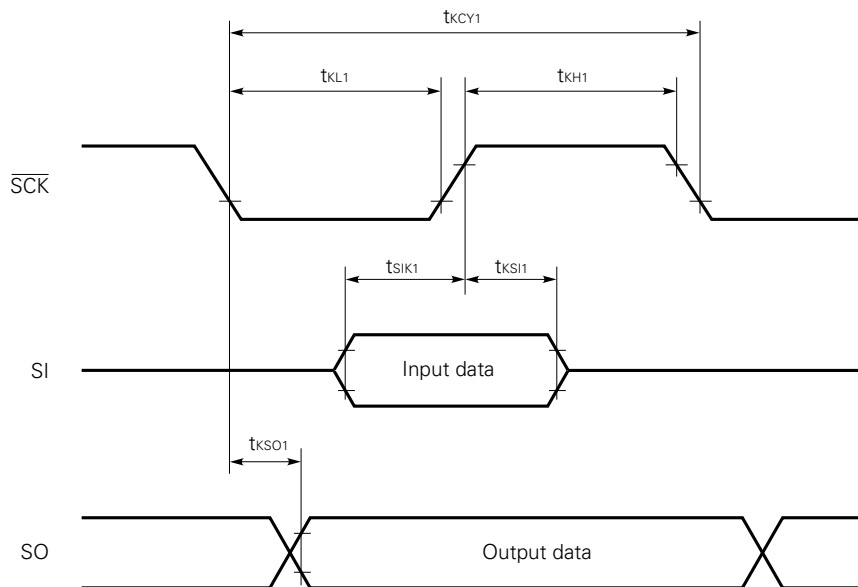


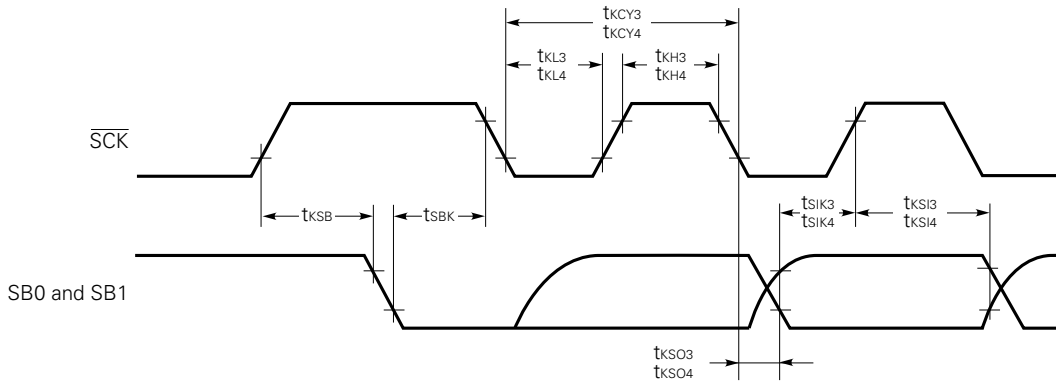**Two-wire serial I/O mode:**
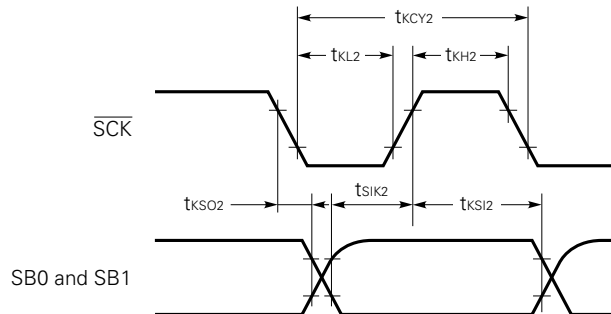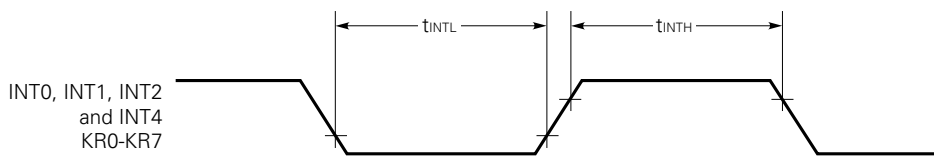
**Serial Transfer Timing**
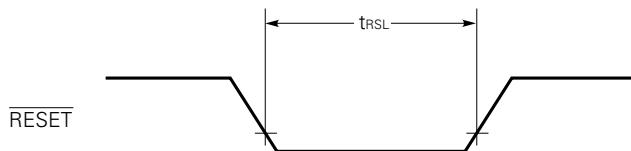
**Bus release signal transfer:**



**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 | $\mu$A | $V_{DDDR}$ = 2.0 V |
| Release signal setting time | $t_{SREL}$ | 0 | | | $\mu$s | |
| Oscillation stability wait time[Note 2] | $t_{WAIT}$ | | $2^{17}/f_X$ | | ms | Release by $\overline{RESET}$ |
| | | | Note 3 | | ms | Release by interrupt request |

**Notes 1.** Excluding the current which flows through the built-in pull-up 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 | |
|---|---|---|---|---|---|
| | | | | $f_X$ = 4.19 MHz | $f_X$ = 6.0 MHz |
| — | 0 | 0 | 0 | $2^{20}/f_X$ (approx. 175 ms) | $2^{20}/f_X$ (approx. 175 ms) |
| — | 0 | 1 | 1 | $2^{17}/f_X$ (approx. 31.3 ms) | $2^{17}/f_X$ (approx. 21.8 ms) |
| — | 1 | 0 | 1 | $2^{15}/f_X$ (approx. 7.82 ms) | $2^{15}/f_X$ (approx. 5.46 ms) |
| — | 1 | 1 | 1 | $2^{13}/f_X$ (approx. 1.95 ms) | $2^{13}/f_X$ (approx. 1.37 ms) |

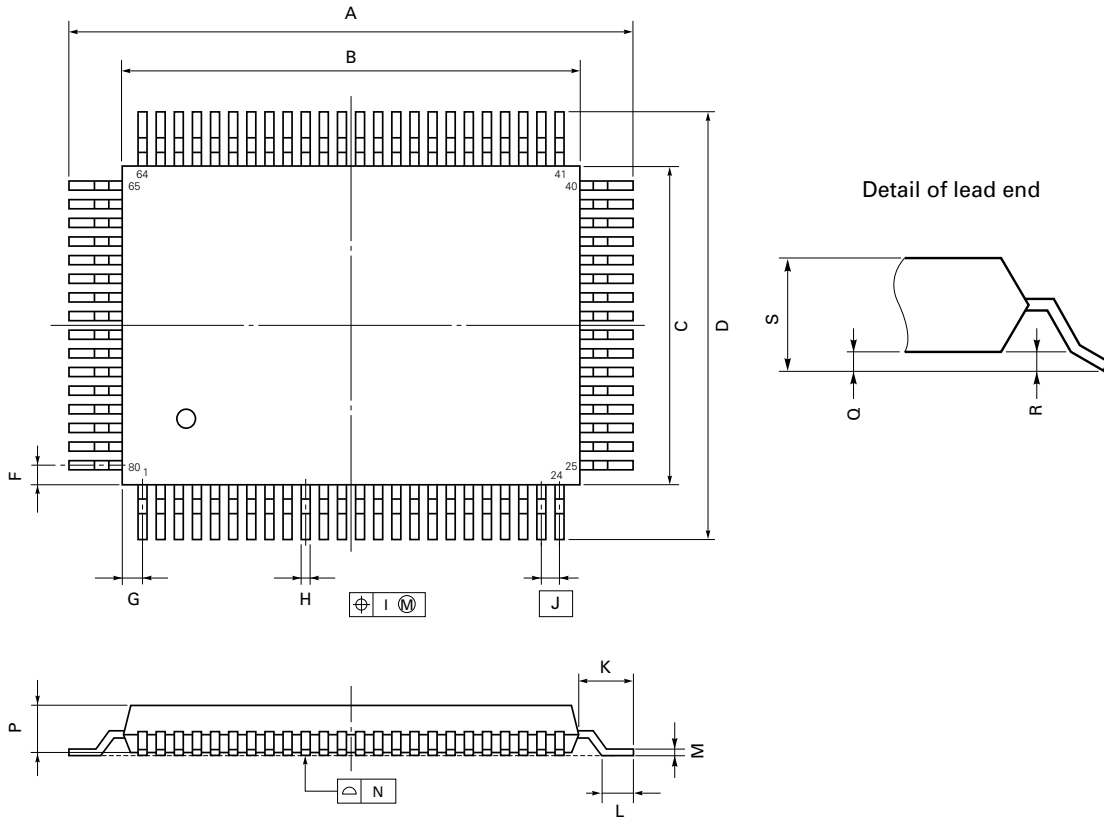### Data Hold Timing (STOP Mode Release by $\overline{RESET}$)
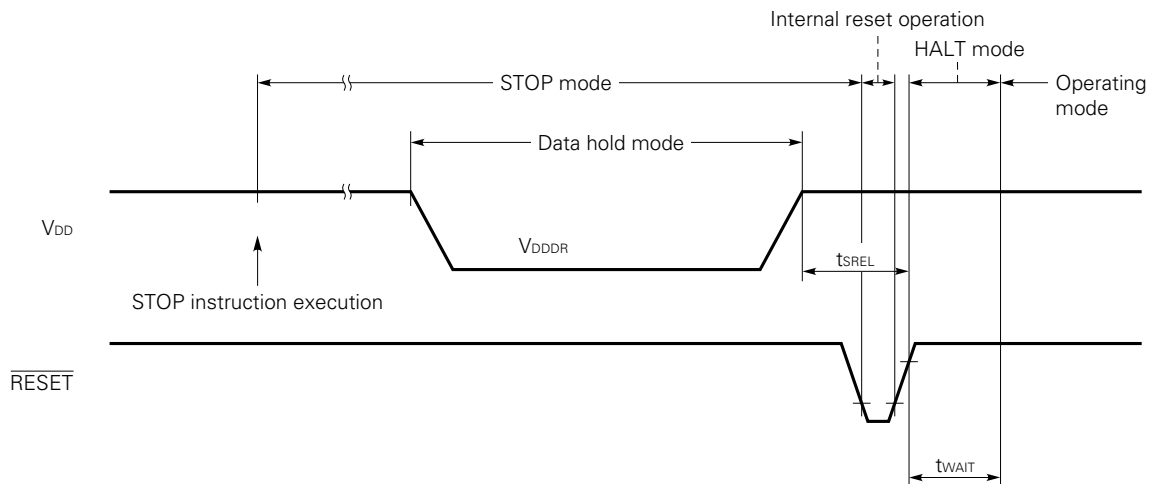


### Data Hold Timing (Standby Release Signal: STOP Mode Release by Interrupt Signal)
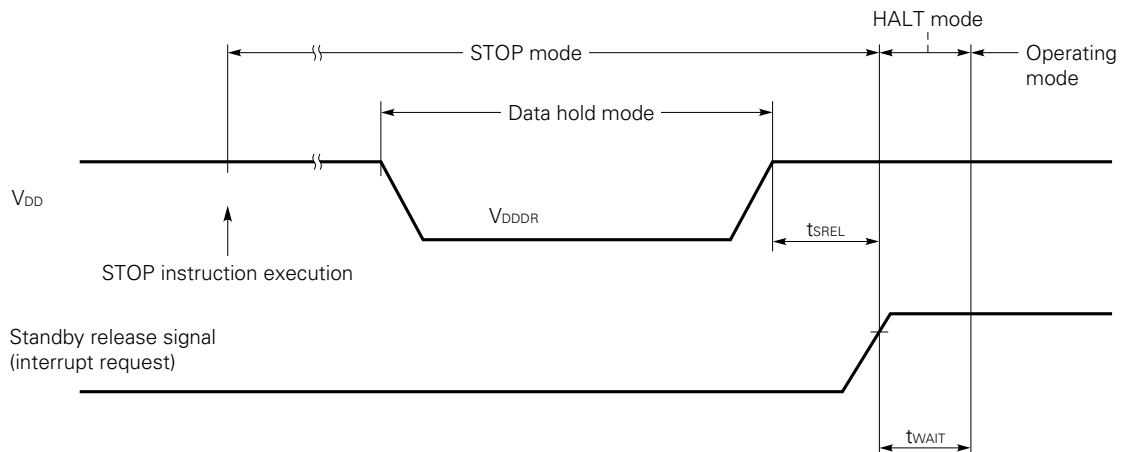
## 10. PACKAGE DIMENSIONS

**PACKAGE DIMENSIONS OF 80-PIN PLASTIC QFP (14 × 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.

P80GF-80-3B9-2E

| ITEM | MILLIMETERS | INCHES |
|------|-------------|--------|
| A | $23.6^{\pm0.4}$ | $0.929^{\pm0.016}$ |
| B | $20.0^{\pm0.2}$ | $0.795^{+0.009}_{-0.008}$ |
| C | $14.00^{\pm0.2}$ | $0.551^{+0.009}_{-0.008}$ |
| D | $17.6^{\pm0.4}$ | $0.693^{\pm0.016}$ |
| F | 1.0 | 0.039 |
| G | 0.8 | 0.031 |
| H | $0.35^{\pm0.10}$ | $0.014^{+0.004}_{-0.005}$ |
| I | 0.15 | 0.006 |
| J | 0.8 (T.P) | 0.031 (T.P.) |
| K | $1.8^{\pm0.2}$ | $0.071^{+0.008}_{-0.009}$ |
| L | $0.8^{\pm0.2}$ | $0.031^{+0.009}_{-0.008}$ |
| M | $0.15^{+0.10}_{-0.05}$ | $0.006^{+0.004}_{-0.003}$ |
| N | 0.15 | 0.006 |
| P | 2.7 | 0.106 |
| Q | $0.1^{\pm0.1}$ | $0.004^{\pm0.004}$ |
| R | $0.1^{\pm0.1}$ | $0.004^{\pm0.004}$ |
| S | 3.0 MAX. | 0.119 MAX. |

## 11. RECOMMENDED SOLDERING CONDITIONS

The following conditions must be met when soldering the μPD75518(A).

Please consult with our sales offices in case other soldering process is used, or in case soldering is done under different conditions.

**Table 11-1  Recommended Soldering Conditions**

| Part number | Package | Symbol |
|---|---|---|
| μPD75518GF(A)-×××-3B9 | 80-pin plastic QFP (14 mm × 20 mm) | WS60-162-1<br>IR30-162-1<br>VP15-162-1<br>Partial heating method |

**Table 11-2  Soldering Conditions**

| Symbol | Soldering process | Soldering conditions |
|---|---|---|
| WS60-162-1 | Wave soldering | Temperature in the soldering vessel:  260 °C or less<br>Soldering time:  10 seconds or less<br>Number of soldering processes:  1<br>Exposure limit[Note]:  2 days<br>    (16 hours of pre-baking is required at 125 °C afterward.)<br>Pre-heating temperature:  120 °C max.<br>    (package surface temperature) |
| IR30-162-1 | Infrared ray reflow | Peak package's surface temperature:  230 °C<br>Reflow time:  30 seconds or less (at 210 °C or more)<br>Number of reflow processes:  1<br>Exposure limit[Note]:  2 days<br>    (16 hours of pre-baking is required at 125 °C afterward.) |
| VP15-162-1 | VPS | Peak package's surface temperature:  215 °C<br>Reflow time:  40 seconds or less (at 200 °C or more)<br>Number of reflow processes:  1<br>Exposure limit[Note]:  2 days<br>    (16 hours of pre-baking is required at 125 °C afterward.) |
| Partial heating method | Partial heating method | Terminal temperature:  300 °C or less<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."**

For more details, refer to our document "SMT MANUAL" (IEI-1207).

## APPENDIX A  SERIES PRODUCT FUNCTIONS

| Product \ Item | $\mu$PD75517 $\mu$PD75517(A) | $\mu$PD75518 $\mu$PD75518(A) | $\mu$PD75P518 | $\mu$PD75516 $\mu$PD75516(A) | $\mu$PD75P516 |
|---|---|---|---|---|---|
| ROM (bytes) | 24448 | 32640 | 32640 (PROM) | 16256 | 16256 (PROM) |
| RAM ($\times$ 4 bits) | 1024 | | | 512 | |
| General register | (4 bits $\times$ 8 or 8 bits $\times$ 4) $\times$ 4 banks | | | | |
| Machine cycle — Main-system clock | 0.67 $\mu$s/1.33 $\mu$s/2.67 $\mu$s/10.7 $\mu$s (at 6.0 MHz) <br> 0.95 $\mu$s/1.91 $\mu$s/3.82 $\mu$s/15.3 $\mu$s (at 4.19 MHz) | | | 0.95 $\mu$s/1.91 $\mu$s/15.3 $\mu$s (at 4.19 MHz) | |
| Machine cycle — Subsystem clock | 122 $\mu$s (at 32.768 kHz) | | | | |
| I/O port [Note 1] — Total | 64 | | | | |
| I/O port — CMOS input | 16 (Shared with INT and SIO.  Seven lines can be pulled up by software) | | | | |
| I/O port — CMOS I/O | 28 (LED direct driving and shared with SIO and PPO) <br> • 16 lines can be pulled up by software. <br> • Four lines can be pulled down by the mask option. | | | | |
| I/O port — N-ch open-drain I/O | 20 (Eight lines for driving LEDs.  Withstand voltage is 10 V.  20 lines can be pulled up by the mask option.) | | | 20 (Eight lines for driving LEDs.  Withstand voltage is 9 V.  20 lines can be pulled up by the mask option.) | |
| A/D converter | • 8-bit resolution $\times$ 8 ch (successive approximation) | | | | |
| A/D converter — Operating voltage | $V_{DD}$ = 2.7 to 6.0 V | | | $V_{DD}$ = 3.5 to 6.0 V | |
| Timer/counter | 4 channels { • Timer/event counter • Basic interval timer • Timer/pulse generator (14-bit PWM output possible) • Clock timer | | | | |
| Serial interface | 2 channels { • NEC standard serial bus interface (SBI)/three-wire SIO : One channel • General synchronous serial interface (three-wire SIO) : One channel | | | | |
| Interrupt | • Vectored interrupt: Seven sources  (External: 3, Internal: 4) <br> • Test input:          Two sources     (External: 1, Internal: 1) <br> • Clock test flag is provided. <br> • Parallel-edge-sensitive flag for key scan input is provided. | | | | |
| Instruction set | The following instructions are added to 75X High-End. <br> • MOVT XA, @BCDE <br> • MOVT XA, @BCXA <br> • BRA !addr1 <br> • BR !BCDE <br> • BR !BCXA <br> • CALLA !addr1 | | | 75X High-End | |
| Operating supply voltage | $V_{DD}$ = 2.7 to 6.0 V | | | | $V_{DD}$ = 4.75 to 5.5 V |
| Package | • 80-pin plastic QFP <br> • 80-pin ceramic LCC with a window[Note 2] | | | | |

**Notes 1.**  No mask option is provided for the $\mu$PD75P516 and $\mu$PD75P518.

      **2.**  Only in the $\mu$PD75P516 and $\mu$PD75P518

## APPENDIX B  DEVELOPMENT TOOLS

The following development tools are provided for developing a system which employs the μPD75518(A).

**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 | The PG-1500 PROM programmer is used together with an accessory board and optional program adapter.  It allows the user to program a single chip microcomputer containing PROM from a standalone terminal or a host machine.  The PG-1500 can be used to program typical 256K-bit to 1M-bit PROMs. | | | |
|---|---|---|---|---|---|
| | PA-75P516GF | The PA-75P516GF is a dedicated PROM programmer for the μPD75P516GF and μPD75P518GF.  This programmer is connected to the PG-1500. | | | |
| | PA-75P516K | The PA-75P516K is a dedicated PROM programmer for the μPD75P516K and μPD75P518K.  This programmer 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**

| Hardware | IE-75000-R**Note** | | 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 μPD75518(A). For efficient debugging, the emulator is connected to the host machine and PROM programmer. |
|---|---|---|---|
| | | IE-75000-R-EM | 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 μPD75518(A). |
| | IE-75001-R | | 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 μPD75518(A). For efficient debugging, the emulator is connected to the host machine and PROM programmer. |
| | EP-75516GF-R | | The EP-75516GF-R is an emulation probe for the μPD755×× series. The emulation probe is connected to the IE-75000-R or IE-75001-R when it is used. A 80-pin conversion socket, the EV-9200G-80, attached to the probe facilitates the connection of the prove with the user system. |
| | | EV-9200G-80 | |
| Software | IE control program | | This program enables the host machine to control the IE-75000-R or IE-75001-R through the RS-232-C interface. |

| Host machine | OS | Distribution media | Part number |
|---|---|---|---|
| PC-9800 series | MS-DOS Ver. 3.10 to Ver. 3.30C | 3.5-inch 2HD | μS5A13IE75X |
| | | 5-inch 2HD | μS5A10IE75X |
| IBM PC series | PC DOS (Ver. 3.1) | 5-inch 2HC | μS7B10IE75X |

**Note**   Maintenance service only

**Remark**   NEC is not responsible for the operation of any software unless it runs on a host machine with the operating system listed above.

**Development tool configuration**

In-circuit emulator

IE-75000-R
IE-75001-R**Note 1**

IE-75000-R-EM

Emulation probe

EP-75516GF-R

Centronics interface

RS-232-C

IE control
program

PG-1500
controller

Host machine
PC-9800 series
IBM PC series
(Symbolic debugging
is possible.)

Relocatable
assembler

80-pin conver-
sion socket
EV-9200G-80

User sysytem

PROM programmer

PG-1500

Product containing PROM

µPD75P518GF
(one-time)
µPD75P518K
(with window)

+

Programmer adapter

PA75P516GF**Note 2**
PA75P516K**Note 3**

**Notes 1.** The IE-75001-R does not contain the
IE-75000-R-EM (to be ordered).
**2.** For the µPD75P518GF (one-time)
**3.** For the µPD75P518K (with window)

µ**PD75518(A)**

**[MEMO]**

**MS-DOS**[TM] **is a trademark of Microsoft Corporation.**

**PC DOS**[TM] **is a trademark of IBM Corporation.**