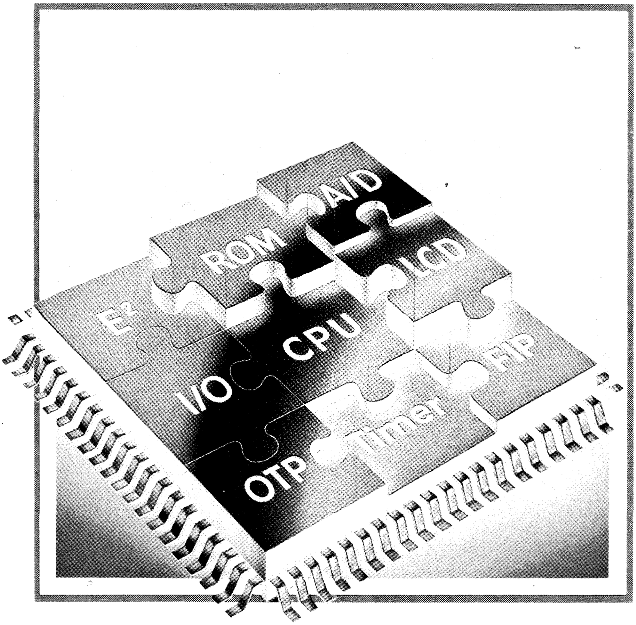
# μCOM-75x Family 4-bit CMOS Microcomputer



# User's Manual Volume II

NEC

Standard 4-bit Microcomputer
μPD75004/006/008 1

Standard 4-bit Microcomputer
with FIP Controller/Driver
μPD75268 2

Standard 4-bit Microcomputer
with LCD Controller/Driver
μPD75304/306/308/312/316 3

Standard 4-bit Microcomputer
with LCD Controller/Driver
and A/D Converter
μPD75328 4

μCOM-75X Family Standard
Instruction Set 5

Low End 4-bit Microcomputer
μPD75402A 6

# Contents

## Chapter 1
## Standard 4-bit Microcomputer μPD75004/006/008

Page

1. Overview ........ II-1- 1
   1.1 Features ........ II-1- 1
   1.2 Ordering information ........ II-1- 3
   1.3 Functional differences between family products ........ II-1- 3
   1.4 Block diagram ........ II-1- 4
   1.5 Pin configuration (Top View) ........ II-1- 5

2. Pin function ........ II-1- 6

3. Architecture and memory map ........ II-1- 15
   3.1 Data memory bank configuration and addressing mode ........ II-1- 15
   3.2 Memory mapped I/O ........ II-1- 26

4. Internal CPU function ........ II-1- 29
   4.1 Program counter (PC) ........ II-1- 29
   4.2 Program memory (ROM) ........ II-1- 30
   4.3 Data memory (RAM) ........ II-1- 34
   4.4 General purpose register ........ II-1- 35
   4.5 Accumulator ........ II-1- 36
   4.6 Stack pointer (SP) ........ II-1- 36
   4.7 Program status word (PSW) ........ II-1- 39
   4.8 Bank selection register (BS) ........ II-1- 41

5. Peripheral hardware function ........ II-1- 42
   5.1 Digital input/output ports ........ II-1- 42
   5.2 Clock generator ........ II-1- 55
   5.3 Basic interval timer ........ II-1- 62
   5.4 Watch timer ........ II-1- 66
   5.5 Timer/event counter configuration ........ II-1- 68
   5.6 Serial interface ........ II-1- 77
   5.7 Bit sequential buffer ........ II-1-104

Page

6. Interrupt function ........ II-1-105
   6.1 Interrupt control circuit configuration ........ II-1-105
   6.2 Interrupt sources types and vector table ........ II-1-107
   6.3 Hardware devices of interrupt control circuit ........ II-1-109
   6.4 Interrupt sequence ........ II-1-114
   6.5 Multiple interrupt processing control ........ II-1-115
   6.6 Processing interrupts which share vector addresses ........ II-1-115
   6.7 Machine cycles to interrupt service start ........ II-1-116
   6.8 Effective use of interrupts ........ II-1-116
   6.9 Interrupt applications ........ II-1-117

7. Standby function ........ II-1-122
   7.1 Standby mode setting and operating state ........ II-1-122
   7.2 Releasing standby mode ........ II-1-123
   7.3 Operating after standby mode is released ........ II-1-125
   7.4 Application of standby mode ........ II-1-125

8. $\overline{\text{RESET}}$ function ........ II-1-129

9. Electrical characteristics ........ II-1-131
   9.1 Package information ........ II-1-149

10. μPD75P008 4-bit microcomputer (OTPROM version) ........ II-1-152
   10.1 Features ........ II-1-152
   10.2 Ordering information ........ II-1-152
   10.3 Pin connection ........ II-1-153
   10.4 Block diagram ........ II-1-154
   10.5 Pin functions ........ II-1-155
   10.6 Differences between μPD75P008 and μPD75008 ........ II-1-160
   10.7 PROM write and verification ........ II-1-160

11. Electrical characteristics μPD75P008 ........ II-1-164
   11.1 Package information ........ II-1-177

## Chapter 2
## Standard 4-bit Microcomputer with FIP Controller/Driver μPD75268

Page

1. Overview ........ II-2- 1
 1.1 Features ........ II-2- 1
 1.2 Ordering information ........ II-2- 1
 1.3 List of functions ........ II-2- 2
 1.4 Functions of the products in the family ........ II-2- 3
 1.5 Block diagram ........ II-2- 4
 1.6 Pin diagram ........ II-2- 5

2. Pin function ........ II-2- 7

3. Architecture and memory map ........ II-2- 15
 3.1 Data memory bank structure and addressing modes ........ II-2- 15
 3.2 Memory-mapped I/O ........ II-2- 25

4. Internal CPU function ........ II-2- 28
 4.1 Program counter ........ II-2- 28
 4.2 Program memory ........ II-2- 28
 4.3 Data memory ........ II-2- 30
 4.4 General-purpose registers ........ II-2- 33
 4.5 Accumulator ........ II-2- 34
 4.6 Stack pointer ........ II-2- 34
 4.7 Program status word ........ II-2- 36
 4.8 Bank select register ........ II-2- 38

5. Peripheral hardware function ........ II-2- 39
 5.1 Digital input/output ports ........ II-2- 39
 5.2 Clock generator ........ II-2- 49
 5.3 Basic interval timer ........ II-2- 55
 5.4 Watch timer ........ II-2- 58
 5.5 Timer/event counter ........ II-2- 60
 5.6 Serial interface ........ II-2- 68
 5.7 FIP controller/driver ........ II-2- 77

Page

6. Interrupt function ........ II-2- 90
   6.1 Configuration of the interrupt control circuit ........ II-2- 90
   6.2 Hardware for interrupt control circuit ........ II-2- 93
   6.3 Interrupt sequence ........ II-2- 97
   6.4 Shared vector address interrupt service ........ II-2- 98
   6.5 Machine cycles for starting interrrupt processing ........ II-2- 99
   6.6 Effective use of interrupts ........ II-2-100
   6.7 Interrupt applications ........ II-2-100

7. Standby function ........ II-2-106
   7.1 Setting of standby modes and operation status ........ II-2-106
   7.2 Releasing standby mode ........ II-2-107
   7.3 Operating after standby mode is released ........ II-2-109
   7.4 Application of standby mode ........ II-2-109

8. $\overline{\text{RESET}}$ function ........ II-2-113

9. Electrical characteristics ........ II-2-114
   9.1 Package information ........ II-2-128

## Chapter 3
## Standard 4-bit Microcomputer with LCD Controller/Driver μPD75304/306/308/312/316

Page

1. Overview ........ II-3- 1
   1.1 Features ........ II-3- 1
   1.2 Ordering information ........ II-3- 3
   1.3 List of family product functions ........ II-3- 4
   1.4 Block diagram ........ II-3- 5
   1.5 Pin connection ........ II-3- 6

2. Pin function ........ II-3- 7

3. Architecture and memory map ........ II-3- 20
   3.1 Data memory bank configuration and addressing modes ........ II-3- 20
   3.2 Memory mapped I/O ........ II-3- 31

4. Internal CPU function ........ II-3- 34
   4.1 Program counter ........ II-3- 34
   4.2 Program memory (ROM) ........ II-3- 35
   4.3 Data memory (RAM) ........ II-3- 41
   4.4 General purpose registers ........ II-3- 43
   4.5 Accumulator ........ II-3- 44
   4.6 Stack pointer ........ II-3- 44
   4.7 Program status word ........ II-3- 48
   4.8 Bank selection register ........ II-3- 50

5. Peripheral hardware function ........ II-3- 51
   5.1 Digital input/output ports ........ II-3- 51
   5.2 Clock generator ........ II-3- 64
   5.3 Basic interval timer ........ II-3- 71
   5.4 Watch timer ........ II-3- 75
   5.5 Timer/event counter configuration ........ II-3- 77
   5.6 Serial interface ........ II-3- 86
   5.7 LCD controller/driver ........ II-3-112
   5.8 Bit sequential buffer ........ II-3-137

Page

6. Interrupt function ........ II-3-138
6.1 Interrupt control circuit configuration ........ II-3-138
6.2 Interrupt source types and vector table ........ II-3-140
6.3 Hardware devices of interrupt control circuit ........ II-3-142
6.4 Interrupt sequence ........ II-3-147
6.5 Multi interrupt service control ........ II-3-148
6.6 Vector address share interrupt service ........ II-3-148
6.7 Machine cycles to interrupt service start ........ II-3-149
6.8 Effective use of interrupts ........ II-3-149
6.9 Interrupt application ........ II-3-150

7 Standby function ........ II-3-155
7.1 Standby mode setting and operating state ........ II-3-155
7.2 Releasing standby mode ........ II-3-156
7.3 Operation after standby mode is released ........ II-3-158
7.4 Applicaton of standby mode ........ II-3-158

8. $\overline{\text{RESET}}$ function ........ II-3-162

9. Electrical characteristics ........ II-3-164
9.1 Package information ........ II-3-182

10. μPD75P308/P316 OTPROM/UVPROM 4-bit microcomputer ........ II-3-184
10.1 Overview ........ II-3-184
10.2 Features ........ II-3-184
10.3 Block diagram ........ II-3-185
10.4 Pin configuration ........ II-3-186
10.5 Pin functions ........ II-3-187
10.6 Differences between μPD75P308/P316 and μPD753XX ........ II-3-193
10.7 PROM write and verification ........ II-3-194
10.8 Electrical characteristics ........ II-3-198
10.9 Package information ........ II-3-211

## Chapter 4
## Standard 4-bit Microcomputer with LCD Controller/Driver and A/D Converter μPD75328

Page

1. Overview ........................................ II-4- 1
1.1 Features ........................................ II-4- 1
1.2 Ordering information ........................................ II-4- 3
1.3 List of family product functions ........................................ II-4- 3
1.4 Block diagram ........................................ II-4- 4
1.5 Pin connection ........................................ II-4- 5

2. Pin function ........................................ II-4- 6
2.1 Input and output ports ........................................ II-4- 6
2.2 Pin function list ........................................ II-4- 9
2.3 Pin input/output circuits ........................................ II-4- 12
2.4 Processing of unused pins ........................................ II-4- 19
2.5 Mask option selection ........................................ II-4- 19

3. Architecture and memory map ........................................ II-4- 20
3.1 Data memory bank configuration and addressing modes ........ II-4- 20
3.2 Memory mapped I/O ........................................ II-4- 31

4. Internal CPU function ........................................ II-4- 34
4.1 Program counter (PC) ........................................ II-4- 34
4.2 Program memory (ROM) ........................................ II-4- 35
4.3 Data memory (RAM) ........................................ II-4- 37
4.4 General purpose registers ........................................ II-4- 39
4.5 Accumulator ........................................ II-4- 40
4.6 Stack pointer (SP) ........................................ II-4- 40
4.7 Program status word (PSW) ........................................ II-4- 42
4.8 Bank selection register (BS) ........................................ II-4- 44

5. Peripheral hardware function ........................................ II-4- 45
5.1 Digital input/output ports ........................................ II-4- 45
5.2 Clock generator ........................................ II-4- 58
5.3 Basic interval timer ........................................ II-4- 65
5.4 Watch timer ........................................ II-4- 69
5.5 Timer/event counter configuration ........................................ II-4- 71
5.6 Serial interface ........................................ II-4- 80
5.7 LCD controller driver ........................................ II-4-106
5.8 A/D converter ........................................ II-4-131
5.9 Bit sequential buffer ........................................ II-4-136

Page

6. Interrupt function ........ II-4-137
- 6.1 Interrupt control circuit configuration ........ II-4-137
- 6.2 Interrupt source types and vector table ........ II-4-139
- 6.3 Hardware devices of interrupt control circuit ........ II-4-140
- 6.4 Interrupt sequence ........ II-4-145
- 6.5 Multi interrupt service control ........ II-4-146
- 6.6 Vector address share interrupt service ........ II-4-146
- 6.7 Machine cycles to interrupt service start ........ II-4-147
- 6.8 Effective use of interrupt ........ II-4-147
- 6.9 Interrupt application ........ II-4-148

7. Standby function ........ II-4-153
- 7.1 Standby mode setting and operating state ........ II-4-153
- 7.2 Standby mode release ........ II-4-154
- 7.3 Operation after standby mode is released ........ II-4-156
- 7.4. Application of standby mode ........ II-4-156

8. $\overline{\text{RESET}}$ function ........ II-4-160

9. Electrical characteristics ........ II-4-162
- 9.1 Package information ........ II-4-180

10. µPD75P328 (OTPROM) 4-bit microcomputer ........ II-4-181
- 10.1 Overview ........ II-4-181
- 10.2 Features ........ II-4-181
- 10.3. Block diagram ........ II-4-182
- 10.4 Pin configuration ........ II-4-183
- 10.5 Pin functions ........ II-4-184
- 10.6 Differences between µPD75P328 and µPD75328 ........ II-4-192
- 10.7 One-time PROM (Program memory) write verification ........ II-4-192
- 10.8 Electrical characteristics ........ II-4-196
- 10.9 Package information ........ II-4-209

## Chapter 5
## µCOM-75X Family Standard Instruction Set

Page

1. Featured instruction .................... II-5- 1
   - 1.1 GETI instruction .................... II-5- 1
   - 1.2 Bit manipulation instructions .................... II-5- 1
   - 1.3 String effect instructions .................... II-5- 2
   - 1.4 Base correction instructions .................... II-5- 2
   - 1.5 Skip instructions .................... II-5- 3
2. Instruction set and operation .................... II-5- 3
3. Operation codes of instruction .................... II-5-10
4. Instructions and applications .................... II-5-14
   - 4.1 Transfer instructions .................... II-5-14
   - 4.2 Table reference instructions .................... II-5-16
   - 4.3 Operating instructions .................... II-5-18
   - 4.4 Accumulator manipulation instructions .................... II-5-19
   - 4.5 Increment and decrement instructions .................... II-5-19
   - 4.6 Comparison instructions .................... II-5-20
   - 4.7 Carry flag manipulation instructions .................... II-5-20
   - 4.8 Memory bit manipulation instructions .................... II-5-21
   - 4.9 Branch instructions .................... II-5-22
   - 4.10 Subroutine stack control instructions .................... II-5-23
   - 4.11 Interrupt control instructions .................... II-5-24
   - 4.12 Input/output instructions .................... II-5-25
   - 4.13 CPU control instructions .................... II-5-26
   - 4.14 Special instructions .................... II-5-26

## Chapter 6
## Low End 4-bit Microcomputer μPD75402

Page

1. Overview ........ II-6- 1
   1.1 Function outline ........ II-6- 1
   1.2 Order information ........ II-6- 2
   1.3 Differences between μPD75402A and μPD75P402 ........ II-6- 2
   1.4 Block diagram ........ II-6- 3
   1.5 Pin connections ........ II-6- 4

2. Pin function ........ II-6- 7
   2.1 Normal operation mode ........ II-6- 7
   2.2 PROM mode ........ II-6- 8
   2.3 Pin input and output circuits ........ II-6- 9

3. Architecture features and memory map ........ II-6- 13
   3.1 Data memory bank configuration and addressing mode ........ II-6- 13
   3.2 Memory mapped I/O ........ II-6- 17

4. Internal CPU function ........ II-6- 19
   4.1 Program counter (PC) ........ II-6- 19
   4.2 Program memory (ROM) ........ II-6- 19
   4.3 Data memory ........ II-6- 20
   4.4 General purpose registers ........ II-6- 21
   4.5 Accumulators ........ II-6- 22
   4.6 Stack pointer (SP) ........ II-6- 22
   4.7 Program status word (PSW) ........ II-6- 24

5. Peripheral hardware function ........ II-6- 25
   5.1 Digital input/output ports ........ II-6- 25
   5.2 Clock generator ........ II-6- 36
   5.3 Clock output circuit ........ II-6- 39
   5.4 Basic interval timer ........ II-6- 41
   5.5 Serial interface ........ II-6- 44

6. Interrupt function ........ II-6- 67
   6.1 Interrupt control circuit configuration ........ II-6- 67
   6.2 Interrupt sources and vector table ........ II-6- 69
   6.3 Interrupt control circuit hardware ........ II-6- 70
   6.4 Interrupt sequence ........ II-6- 74
   6.5 Machine cycles until interrupt service starts ........ II-6- 75
   6.6 Interrupt application ........ II-6- 77

Page

7. Standby function ........................................ II-6- 81

7.1 Standby mode setting and operation state ........................................ II-6- 81

7.2 Standby mode release ........................................ II-6- 82

7.3 Operation after standby mode is released ........................................ II-6- 83

7.4 Standby mode application ........................................ II-6- 83

8. $\overline{\text{RESET}}$ function ........................................ II-6- 84

9. Electrical characteristics ........................................ II-6- 86

9.1 Package dimension ........................................ II-6- 95

10. μPD75P402 low end 4-bit microcomputer ........................................ II-6- 98

10.1 Overview ........................................ II-6- 98

10.2 Pin connections ........................................ II-6- 99

10.3 Block diagram ........................................ II-6-103

10.4 Pin function list ........................................ II-6-104

10.5 Differences between μPD75P402 and μPD75402 ........................................ II-6-108

10.6 One Time PROM programming ........................................ II-6-108

10.7 One Time PROM programming operating mode ........................................ II-6-109

11. Electrical characteristics ........................................ II-6-111

11.1 Package information ........................................ II-6-122

12. Instruction set μPD75402A ........................................ II-6-124

12.1 Featured instructions ........................................ II-6-124

12.2 Instruction set and operations ........................................ II-6-125

12.3 Operation code of instructions ........................................ II-6-130

12.4 Instructions and applications ........................................ II-6-133

## CHAPTER 1 μPD75004/006/008 STANDARD 4-BIT MICROCOMPUTER

### 1. Overview

μPD75004, μPD75006 and the μPD75008 are high-speed, high-performance 4-bit single-chip microcomputers in the μCOM-75X product family.

| | |
|---|---|
| • ROM size: | 8064 x 8 bits (max.) |
| • RAM size: | 512 x 4 bits |
| • General purpose registers: | 4-bit x 8 |
| • High-speed operation: | Minimum instruction execution time: 1μs |

- Eight interrupt sources and efficient interrupt processing
- Efficient instruction set which can manipulate data in 1-, 4-, or 8-bit units.
- Enhanced timer function: 3 channels
- Low-power clock operation in standby mode (with subsystem clock for low power dissipation operation)
- Internal NEC standard serial bus interface

Table 1-1 shows the program memory size for each of these products.

**Table 1-1 Program Memory Size for Each Product**

| Product | Program memory (ROM) | Remarks |
|---|---|---|
| μPD75004 | 4096 x 8 bits | Mask ROM |
| μPD75006 | 6016 x 8 bits | |
| μPD75008 | 8064 x 8 bits | |

Applications: • Consumer electronic products VCR's, audio systems (CD players, etc.), remote controllers, etc.
• Other product applications telephones, cameras, etc.

Remarks: This manual describes the μPD75004, μPD75006 and μPD75008.
The μPD7500X described here is functionally equivalent to the μPD75004, μPD75006 and μPD75008, unless stated otherwise. Therefore, except where noted, the μPD7500X is used generally to refer the μPD75004, μPD75006 and μPD75008.

### 1.1 Features

- 41 different systemized instructions
  - — Abundant bit manipulation instructions
  - — Effecient 4-bit data manipulation instructions
  - — 8-bit data transfer instructions
  - — The GETI instruction which enables execution of an arbitrary 2-byte/3-byte instruction with 1 byte.
- High-speed operation ... Minimum instruction execution time: 0.95μs/4.19MHz, 5V
- Capable of operating at low voltage and reduced power dissipation. Variable instruction execution time function
  - — With main subsystem clock selected: 0.95μs, 1.91μs, 15.3μs at 4.19MHz
  - — With the subsystem clock selected: 122μs at 32.768kHz
- Program memory size: μPD75004: 4096 x 8 bits (mask ROM)
  μPD75006: 6016 x 8 bits (mask ROM)
  μPD75008: 8064 x 8 bits (mask ROM)
- Data memory size (RAM): 512 x 4 bits
- General purpose registers
  - — For 4-bit manipulation: 8 registers (X, A, B, C, D, E, H, L)
  - — For 8-bit manipulation: 4 registers (XA, BC, DE, HL)

- Accumulators
  - Bit accumulator (CY)
  - 4-bit accumulator (A)
  - 8-bit accumulator (XA)
- 34 I/O lines
  - Input pins: 8
  - Middle voltage N-ch open drain input/output pins: 8 (can directly drive LED)
  - CMOS input/output pins: 18 (pins which can directly drive LED: 4 pins)
  - Built-in pull-up resistors can be provided for 33 I/O lines.
  - By means of software: 25 I/O lines
  - By means of mask option: 8 I/O lines
- Timer: 3-ch
  - 8-bit timer/event counter
    - 4-step clock source
    - Capable of event count operation
  - 8-bit basic interval timer
    - Reference timer generation (1.95ms, 7.81ms, 31.3ms, 250ms at 4.19MHz)
    - Can be used as a watchdog timer
  - Clock timer
    - 0.5-second interval generation
    - Count clock sources: Can be selected from the Main system clock and subsystem clock
    - Clock advance mode (3.9ms interval generation)
    - Capable of outputting buzzer signal (2kHz)
- 8-bit serial interface
  - Can accomodate three different modes
    - Three-line serial I/O mode
    - Two-line serial I/O mode
    - SBI mode
  - LSB-first/MSB-first selectable
- Bit sequential buffer (special bit manipulation memory of 16 bits)
  - Suitable for remote control applications
- Clock output functions
  - Timer/event counter output (PTO0): Square wave output of arbitrary frequency
  - Clock output (PCL): $\phi$, $f_X/2^3$, $f_X/2^4$, $f_X/2^6$
  - Buzzer output (BUZ): 2kHz at 4.19MHz (32.768kHz)
- Vector interrupt function
  - External vector interrupt sources: 3
    - Both edges detection interrupt (INT4)
    - Detection edge programmable interrupt with noise elimination function (INT0)
    - Detection edge programmable interrupt (INT1)
  - External rising edge detection/parallel port edge detection test input (INT2)
  - Internal vector interrupt sources: 3
    - Timer/event counter 0 interrupt (INTT0)
    - Basic interval timer interrupt (INTBT)
    - Serial interface interrupt (INTCSI)
  - Clock test input (INTW)
- Two built-in system clock generators
  - Ceramic/crystal oscillator for main system clock generator: 4.194304MHz (standard)
  - Crystal oscillator for subsystem clock generator: 32.768kHz (standard)
- Standby functions
  - STOP mode: The main system clock operation stops
  - HALT mode: The main system clock operation continues (CPU clock supply stops)
- CMOS

### 1.2 Ordering Information

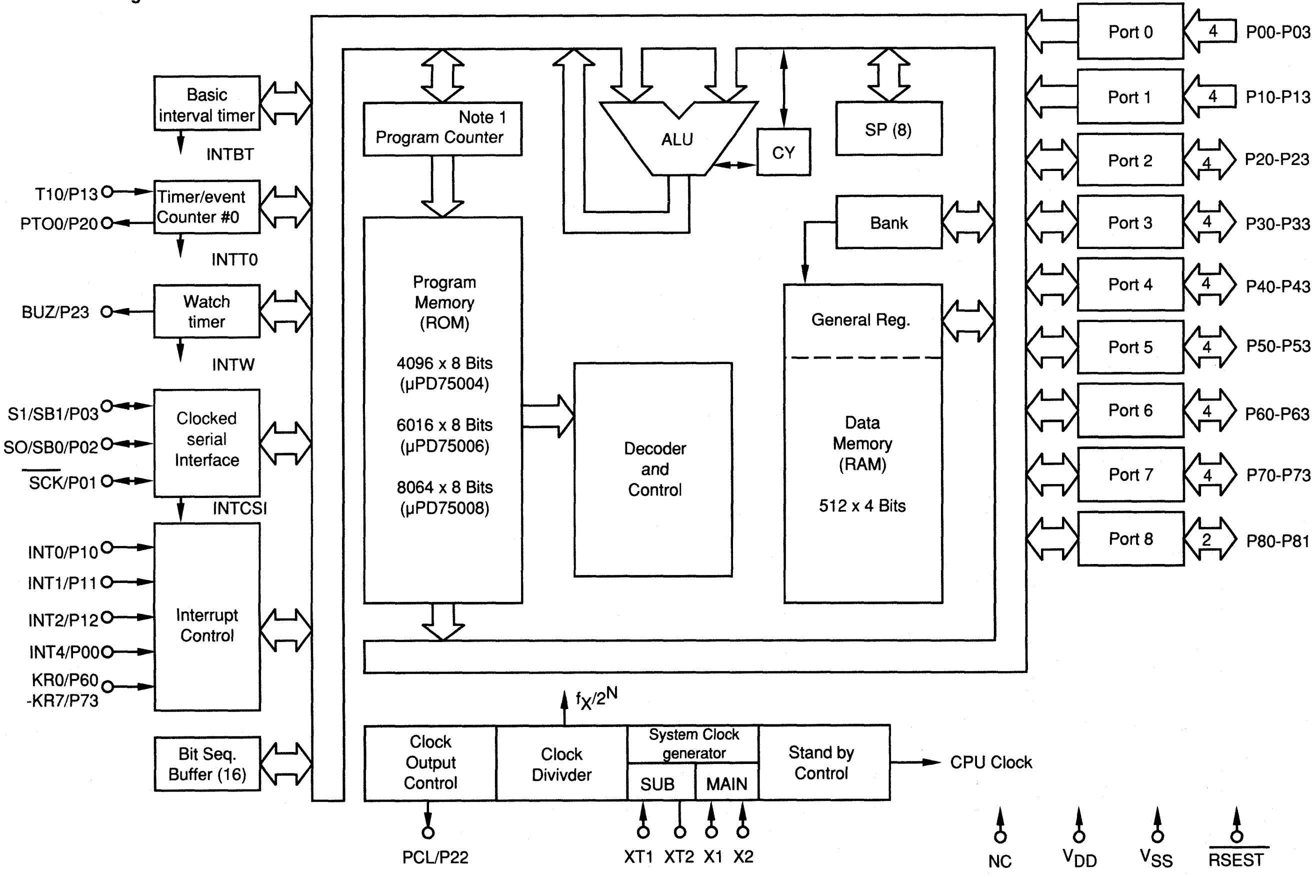
| Order code | Package | Internal ROM |
|---|---|---|
| μPD75004CU-XXX | 42-pin plastic shrink DIP | Mask ROM |
| μPD75004GB-XXX-3B4 | 44-pin plastic QFP (bent) | Mask ROM |
| μPD75006CU-XXX | 42-pin plastic shrink DIP | Mask ROM |
| μPD75006GB-XXX-3B4 | 44-pin plastic QFP (bent) | Mask ROM |
| μPD75008CU-XXX | 42-pin plastic shrink DIP | Mask ROM |
| μPD75008GB-XXX-3B4 | 44-pin plastic QFP (bent) | Mask ROM |

Remarks: XXX is the code number of mask ROM and option specification.

### 1.3 Functional Differences between Family Products

| Item | | μPD75004CU/GB | μPD75006CU/GB | μPD75008CU/GB |
|---|---|---|---|---|
| Program Memory | | • Mask ROM<br>• 000H-FFFH<br>• 4096 x 8 bits | • Mask ROM<br>• 0000H-177FH<br>• 6016 x 8 bits | • Mask ROM<br>• 0000H-1F7FH<br>• 8064 x 8 bits |
| Data memory | | 512 x 4 bits<br>Bank 0: 256 x 4<br>Bank 1: 256 x 4 | | |
| Instruction set | 3-byte branch instruction | None | Provided | |
| | Other | Common | | |
| Program counter | | 12 bits | 13 bits | |
| Pull-up resistor | Ports 0 to 3, Ports 6 to 8 | Can be specified by software | | |
| | Port 4, Port 5 | Mask option | | |
| Operating voltage range | | 2.7 V - 6.0 V | | |
| Package | | 42-pin plastic shrink DIP<br>44-pin plastic QFP (bent) | | |

### 1.4 Block Diagram

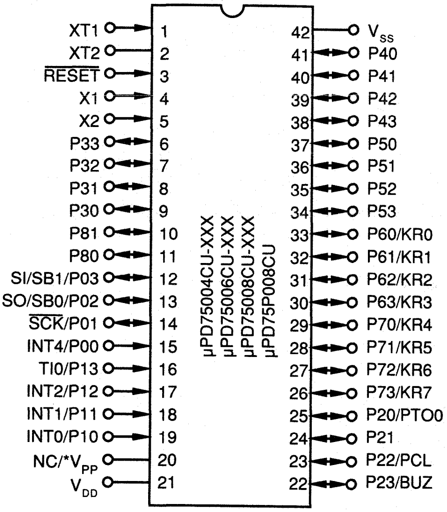


Note 1: For the µPD75004, the program is 12 bits. For the µPD75006, µPD75008 program counter size is 13 bits.

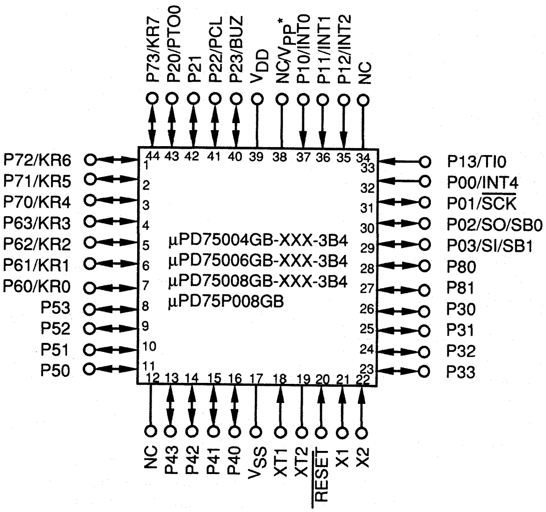
### 1.5 Pin Configuration (Top View)

μPD75008

- 42-pin plastic shrink DIP



- 44-pin plastic QFP



Pin name

| | | | |
|---|---|---|---|
| P00-03: | Port 0 | SO: | Serial Output |
| P10-13: | Port 1 | SB0; 1 | Serial Bus 0, I |
| P20-23: | Port 2 | RESET: | Reset Input |
| P30-33: | Port 3 | TI0: | Timer Input 0 |
| P40-43: | Port 4 | PTO0: | Programmable Timer Output 0 |
| P50-53: | Port 5 | BUZ: | Buzzer Clock |
| P60-63: | Port 6 | PCL: | Programmable Clock |
| P70-73: | Port 7 | INT0, 1, 4: | External Vector Interrupt 0, 1, 4 |
| P80-81: | Port 8 | INT2: | External Test Input 2 |
| KR0-7: | Key Return | X1, X2: | Main System Clock Oscillation 1, 2 |
| SCK: | Serial Clock | XT1, 2: | Subsystem Clock Oscillation 1, 2 |
| SI: | Serial Input | NC/$V_{PP}$*: | No Connection |

* When using OTP µPD75P008 in the same socket, please note that this pin is $V_{PP}$ and has to be connected to $V_{DD}$ in operating mode.

## 2. PIN FUNCTIONS

2.1.1 P00-P03 (PORT0) ... Also serves as INT4, SCK, SO/SB0, SI/SB1 input pins
P10-P13 (PORT1) ... Also serves as INT0, INT1, INT2, TI0 input pins

These pins are input pins of the 4-bit input ports: Ports 0 and Port 1.

Ports 0 and 1 are input ports. However, in addition to their input function, the pins of these ports have the following functions:

(1) Port 0: Vector interrupt input (INT4)
Serial interface input/output (SCK, SO/SB0, SI/SBI)

(2) Port 1: Vector interrupt input (INT0, INT1)
Edge detection test input (INT2) Timer/event counter external event pulse input (TI0)

Each pin of Ports 0 and 1 employs a Schmidt trigger input circuit to prevent malfunction due to noise, and can function as an input pin regardless of the operation of the shared pin.

In addition, a noise elimination circuit is provided for P10 (refer to 6.3 (2) for details).

An internal pull-up resistor can be software specified for Port 0 in 3-bit units (P01-P03), and in 4-bit units (P10-P13) for Port 1. This specification can be made by manipulating the pull-up resistor specification register group A.

When the RESET signal is generated, each of these port pins is set to the input port mode.

2.1.2 P20-P23 (PORT2) ... Also serves as PTO0, PCL, BUZ input/output pins
P30-P33 (PORT3) ... 3-state input/output
P40-P43 (PORT4),
P50-P53 (PORT5) ... N-ch open-drain middle-voltage (10V) large current output pins
P60-P63 (PORT6),
P70-P73 (PORT7) ... 3-state input/output

Ports 2 to 7, whose pin functions are described above, are 4-bit input/output ports with output latches.

The pins of Ports n (n=2, 3, 6, and 7) also have the following functions:

(1) Port 2: Timer/event counter output (PTO0). Clock output (PCL). Fixed frequency output (BUZ).

(2) Port 3: Input/output can be selected bitwise

(3) Ports 6 and 7: Key interrupt input (KR0-KR3, KR4-KR7)

Since Port 3 can output large current and Ports 4 and 5 are N-ch open-drain (middle-voltage (10V), large current output ports), these three ports can directly drive LED's.

Input/output mode selection for each port is made by setting the port mode register: Ports m (m=2, 4, 5, and 7) can be specified for input/output in 4-bit units; Ports 3 and 6 can be specified for input/output in bit units.

The internal pull-up resistor can be specified for Ports n in 4-bit units by software: this is done by manipulating pull-up resistor specification register group A (POGA). For Ports 4 and 5, the internal pull-up resistor can be specified in bit units by the mask option.

Ports 4 and 5, and Ports 6 and 7 can be paired to enable an input/output operation in 8-bit units.

When the RESET signal is generated, the output latches of these ports are cleared, Ports n are set to the input mode (high impedance output), and Ports 4 and 5 are set to a high level (when internal pull-up resistor is provided) or at high impedance.

2.1.3 P80-P81 (PORT8)

These are input/output pins for Port 8 which is a 2-bit input/output port with an output latch. The internal pull-up resistor can be specified for Port 8 in 2-bit units by software. This specification can be made by manipulating pull-up resistror specification register group B (POGB).

### 2.1.4 TI0 ... Port 1 shared input/output

This is the external event pulse pin for the programmable timer/event counter. This is Schmidt trigger input.

### 2.1.5 PTO0 ... Port 2 shared output

This is the programmable timer/event counter output pin. Squarewave pulses are output from this pin. To output the programmable timer/event counter signal, the P20 output latch must be cleared to 0 and the Port 2 bit of the port mode register must be set to the output mode (1). The output is cleared to 0 when the timer start instruction is executed.

### 2.1.6 PCL ... Port 2 shared output

This is the programmable clock output pin. This pin is used to supply the clock to peripheral LSI (slave microcomputer, A/D converter, etc.). When the RESET signal is generated, the clock mode register (CLOM) is cleared to 0, the clock output is disabled, and this pin functions as a normal port pin.

### 2.1.7 BUZ - Output Pin Also Used for Port 2

BUZ is a fixed frequency output pin. Fixed frequency (2.048 kHz) output is used for buzzer sounding and system clock oscillation frequency trimming. The BUZ pin, also used for the P23 pin, is validated only when bit 7 (WM7) of the watch mode register (WM) is set to 1.

When the RESET signal is generated, WM7 is cleared and the normal port operation mode is set.

### 2.1.8 SCK, SO/SB0, and SI/SB1 - 3-State Input/Output Pins Also Used for Port 0

SCK, SO/SB0, and SI/SB1 are input/output pins for serial interface and operate according to how the serial operation mode register (CSIM) is set.

When the RESET signal is generated, serial interface operation is stopped and the pins are used for port 0 (input port).

Every pin is Schmitt trigger input.

### 2.1.9 INT4 - Input Pin also Used for Port 0.

INT4 is an external vectored interrupt input pin (both rising and falling edged active). When the signal input to the pin changes from low to high state or from high to low state, the interrupt request flag is set.

INT4 is for asynchronous input. When a signal having a given high or low level duration is input, it is acknowledged independently of the CPU operation clock.

INT4 can also be used to release the STOP or HALT mode. It is Schmitt trigger input.

### 2.1.10 INT0 and INT1 - Input Pins Also Used for Port 1

INT0 and INT1 are edge detection vectored interrupt input pins. INT0 has the noise removal function. Detected edge selection can be made using edge detection mode registers (IM0 and IM1).

(1) INT0 (IM0 bits 0 and 1)
   (a) Rising edge active
   (b) Falling edge active
   (c) Both rising and falling edges active
   (d) External interrupt signal input inhibited

(2) INT1 (IM1 bit 0)
   (a) Rising edge active
   (b) Falling edge active

INT0 has the noise removal function; sampling clocks for noise removal can be changed two stages. The acknowledged signal width varies depending on CPU clock operation.

INT1 is for asynchronous input. When a signal having a given high level duration is input, it is acknowledged independently of the CPU clock operation.

When the RESET signal is generated, IM0 and IM1 are cleared and rising edge active is selected.

INT0 and INT1 are Schmitt trigger inputs.

### 2.1.11 INT2 - Input Pin Also Used for Port 1

INT2 is an external test input pin (both rising and falling edges active). When INT2 is selected by using the edge detection mode register (IM2) and the signal input to the INT2 pin transits from low to high state, the internal test flag (IRQ2) is set.

INT2 is for asynchronous input. When a signal having a given high level duration is input, it is acknowledged independently of the CPU clock operation.

When the RESET signal is generated, IM2 is cleared and the test flag (IRQ2) is set by inputting the rising edge to the INT2 pin.

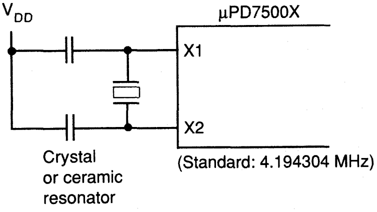
2.1.12 KR0-KR3 - Input Pins Also Used for Port 6
KR4-KR7 - Input Pins Also Used for Port 7

KR0-KR7 are key interrupt (parallel falling edge detection interrupt) input pins. The interrupt format can be specified by setting the edge detection mode register (IM2).
When the $\overline{\text{RESET}}$ signal is generated, the pins are placed in port 6 and 7 input mode.

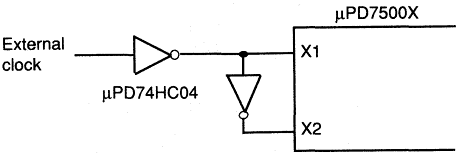
2.1.13 X1 and X2

X1 and X2 are connection pins for the main system clock crystal / ceramic oscillator.
External clocks can also be input.

(a) Crystal ceramic oscillator



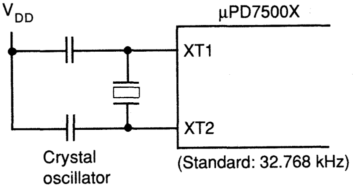
(b) External clock



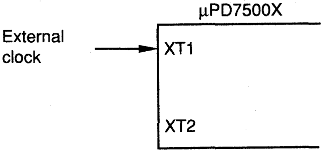
2.1.14 XT1 and XT2

XT1 and XT2 are crystal connection pins for the subsystem clock oscillation.
External clocks can also be input.

(a) Crystal oscillator



(b) External clock



2.1.15 $\overline{\text{RESET}}$

$\overline{\text{RESET}}$ is an active low reset input pin.
$\overline{\text{RESET}}$ is for asynchronous input. When a signal having a given low level duration is input independently of the operation clock, the $\overline{\text{RESET}}$ signal is generated and the system is reset overriding all other operations.
It is used for normal CPU initialization / start-up and also to release the standby (STOP or HALT) mode.
RESET is a Schnitt trigger input.

2.1.16 $V_{DD}$

$V_{DD}$ is a positive power supply pin.

2.1.17 $V_{SS}$

$V_{SS}$ is a ground potential.

### 2.2 Pin Function List

(1) Normal operation mode

**Table 2.2–1 Digital Input / Output Port Pin Function List**

| Pin name | I/O | Also used for: | Function | 8-bit I/O | When reset | I/O circuit type (Note 1) |
|---|---|---|---|---|---|---|
| P00 | I | INT4 | 4-bit input port (port 0). Internal pull-up resistors can be specified for P01-P03 in 3-bit units by using software | x | Input | Ⓑ |
| P01 | I/O | $\overline{\text{SCK}}$ | | | | Ⓕ- A |
| P02 | I/O | SO/SBO | | | | Ⓕ- B |
| P03 | I/O | SI/SB1 | | | | Ⓜ- C |
| P10 | I | INTO | With noise removal function<br>4-bit input port (port 1). Internal pull-up resistors can be specified for P10-P13 in 4-bit units by using software | x | Input | Ⓑ- C |
| P11 | | INT1 | | | | |
| P12 | | INT2 | | | | |
| P13 | | TIO | | | | |
| P20 | I/O | PTOO | 4-bit input/output port (port 2). Internal pull-up resistor can be specified for P20-P23 in 4-bit units by using software. | x | Input | E – B |
| P21 | | – | | | | |
| P22 | | PCL | | | | |
| P23 | | BUZ | | | | |
| P30 | I/O | – | Programmable 4-bit input/output port (port 3). Input or output mode can be selected bitwise.<br>Internal pull-up resistor can be specified for P30-P33 in 4-bit units by using software. (Note 2) | x | Input | E – B |
| P31 | | – | | | | |
| P32 | | – | | | | |
| P33 | | – | | | | |
| P40-P43 (Note 2) | I/O | – | N-channel open drain 4-bit input/output port (port 4). Pull-up resistor can be incorporated bitwise (mask option). 10 Volts during open drain. | o | High level (when pull-up resistor is incorporated) or high impendance | M |
| P50-P53 (Note 2) | I/O | – | N-channel open drain 4-bit input/output port (port 4). Pull-up resistor can be incorporated bitwise (mask option). 10 Volts during open drain. | | High level (when pull-up resistor is incorporated) or high impendance | M |
| P60 | I/O | KR0 | Programmable 4-bit input/output port (port 6). Input or output mode can be selected bitwise.<br>Internal pull-up resistor can be specified for P60-P63 in 4-bit units by using software. | o | Input | Ⓕ- A |
| P61 | | KR1 | | | | |
| P62 | | KR2 | | | | |
| P63 | | KR3 | | | | |
| P70 | I/O | KR4 | 4-bit input/output port (port 7). Internal pull-up resistor can be specified for P70-P73 in 4-bit units by using software. | | Input | Ⓕ- A |
| P71 | | KR5 | | | | |
| P72 | | KR6 | | | | |
| P73 | | KR7 | | | | |

(to be continued)

**Table 2.2–1 Digital Input / Output Port Pin Function List (cont`d)**

| Pin name | I/O | Also used for | Function | 8-bit I/O | When reset | I/O circuit type (Note 1) |
|---|---|---|---|---|---|---|
| P80 | Input/ Output | | 2-bit input/output port (PORT8). Built-in pull-up resistors can be specified in 2-bit units by software. | X | Input | E – B |
| P81 | | | | | | |

Note 1: The ○ mark denotes Schmitt trigger input
Note 2: LED can be driven directly

**Table 2.2–2 Pin Function List other than Port Pins**

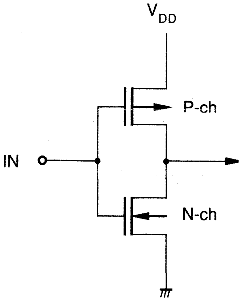
<table>
<tr><th>Pin name</th><th>I/O</th><th>Also used for:</th><th colspan="2">Function</th><th>When reset</th><th>I/O circuit type (Note 1)</th></tr>
<tr><td>TI0</td><td>I</td><td>P13</td><td colspan="2">External event pulse input pin to timer / event counter.</td><td></td><td>Ⓑ– C</td></tr>
<tr><td>PT00</td><td>I/O</td><td>P20</td><td colspan="2">Timer / event counter output pin.</td><td>Input</td><td>E – B</td></tr>
<tr><td>PCL</td><td>I/O</td><td>P22</td><td colspan="2">Clock output pin.</td><td>Input</td><td>E – B</td></tr>
<tr><td>BUZ</td><td>I/O</td><td>P23</td><td colspan="2">Fixed frequency output pin (for buzzer or system clock trimming).</td><td>Input</td><td>E – B</td></tr>
<tr><td>$\overline{\text{SCK}}$</td><td>I/O</td><td>P01</td><td colspan="2">Serial clock input / output pin.</td><td>Input</td><td>Ⓕ– A</td></tr>
<tr><td>SO/SB0</td><td>I/O</td><td>P02</td><td colspan="2">Serial data output pin.<br>Serial bus input / output pin.</td><td>Input</td><td>Ⓕ– B</td></tr>
<tr><td>SI/SB1</td><td>I/O</td><td>P03</td><td colspan="2">Serial data input pin.<br>Serial bus input / output pin.</td><td>Input</td><td>Ⓜ– C</td></tr>
<tr><td>INT4</td><td>I</td><td>P00</td><td colspan="2">Edge detection vectored interrupt input pin (detection of both rising and falling edges is active).</td><td></td><td>Ⓑ</td></tr>
<tr><td>INT0</td><td rowspan="2">I</td><td>P10</td><td rowspan="2">Edge detection vectored interrupt input pin (detected edge can be selected).</td><td>Synchronous clock</td><td rowspan="2"></td><td rowspan="2">Ⓑ– C</td></tr>
<tr><td>INT1</td><td>P11</td><td>Asynchronous</td></tr>
<tr><td>INT2</td><td>I</td><td>P12</td><td colspan="2">Edge detection testable input pin asynchronous (rising edge is detected).</td><td></td><td>Ⓑ– C</td></tr>
<tr><td>KR0-KR3</td><td>I/O</td><td>P60-P63</td><td colspan="2">Parallel falling edge detection testable input / output pins.</td><td>Input</td><td>Ⓕ– A</td></tr>
<tr><td>KR4-KR7</td><td>I/O</td><td>P70-P73</td><td colspan="2">Parallel falling edge detection testable input / output pins.</td><td>Input</td><td>Ⓕ– A</td></tr>
<tr><td>X1, X2</td><td></td><td>–</td><td colspan="2">Connection pins for main system clock crystal/ceramic oscillator. When external clock is used, it is input to X1 and its opposite phase is input to X2.</td><td>–</td><td>–</td></tr>
<tr><td>XT1</td><td>I</td><td rowspan="2">–</td><td colspan="2" rowspan="2">Subsystem clock oscillation crystal connection pins. When external clock is used, it is input to XT1, and XT2 is not connected. XT1 can be used for 1-bit input (test) pin.</td><td>–</td><td>–</td></tr>
<tr><td>XT2</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>$\overline{\text{RESET}}$</td><td>I</td><td></td><td colspan="2">System reset input pin.</td><td>–</td><td>Ⓑ</td></tr>
<tr><td>NC<br>(Note 2)</td><td>–</td><td></td><td colspan="2">No connection</td><td>–</td><td>–</td></tr>
<tr><td>$V_{DD}$</td><td>–</td><td></td><td colspan="2">Positive power supply pin.</td><td>–</td><td>–</td></tr>
<tr><td>$V_{SS}$</td><td>–</td><td></td><td colspan="2">Ground potential pin.</td><td>–</td><td>–</td></tr>
</table>

Notes: 1.The○ mark denotes Schmitt trigger input.
2. Connected the NC pin to $V_{DD}$, when μPD75P008 and printed board are also shared with.

### 2.3 Pin Input / Output Circuits

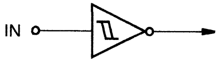
The μPD7500X pin input / output circuits are shown in schematic drawings.

(1) Type A (for Type E – B)



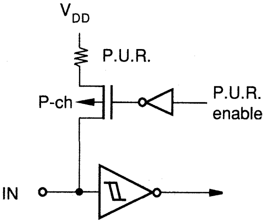
CMOS standard input buffer

(2) Type B



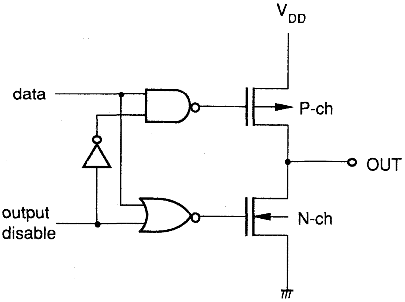
Schmitt trigger input with hysteresis characteristic

(3) Type B – C



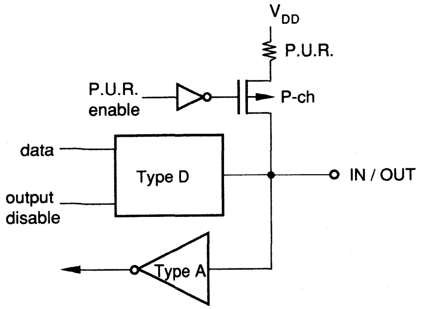
P.U.R.: Pull-Up Resistor

(4) Type D (for Type E – B, F – A)



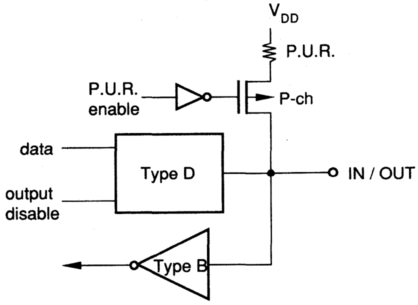
Push-pull output where output can be placed in high impedance (both P and N channels are turned off).

(5) Type E - B



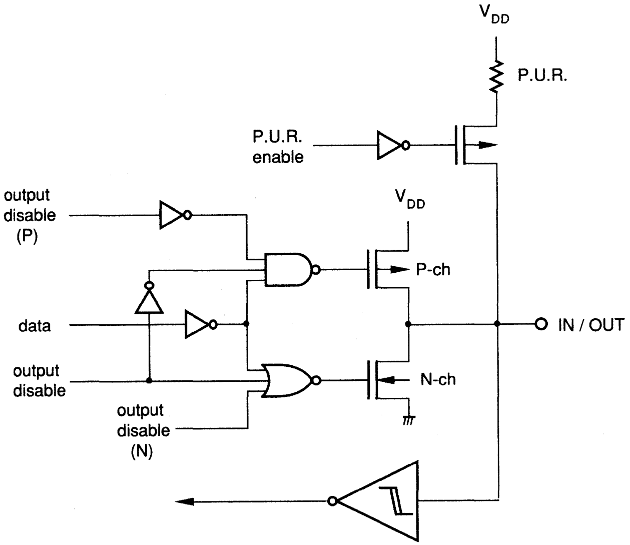
P.U.R.: Pull-Up Resistor

(6) Type F – A



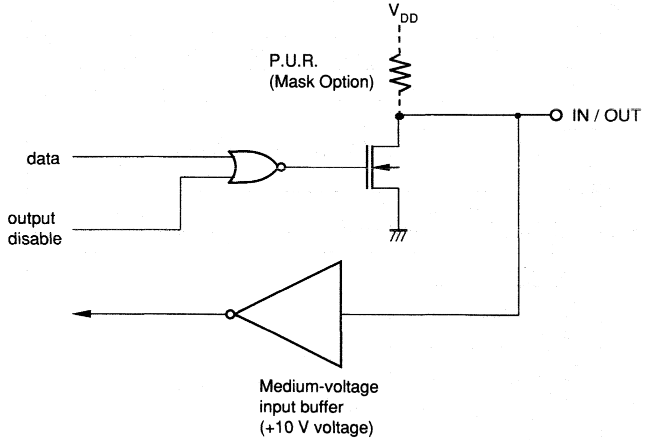
P.U.R.: Pull-Up Resistor

(7) Type F – B



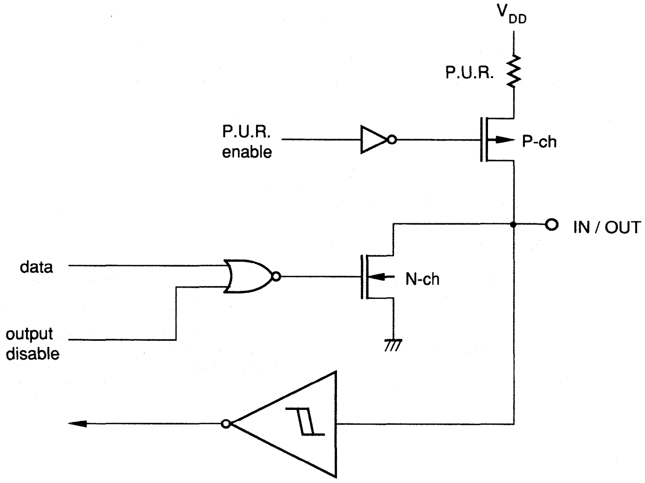
P.U.R.: Pull-Up Resistor

(8) Type M



P.U.R.: Pull-Up Resistor

(9) Type M – C



P.U.R.: Pull-Up Resistor

## 2.4 Processing of Unused Pins

### Table 2.4–1 Unused Pins Lists

| Pin | Recommended connection |
|---|---|
| P00 / INT4 | Connect to $V_{SS}$. |
| P01 / $\overline{SCK}$ | Connect to $V_{SS}$ or $V_{DD}$. |
| P02 / SO /SB0 | |
| P03 / SI / SB1 | |
| P10 / INTO–P12 / INT2 | Connect to $V_{SS}$. |
| P13 / TI0 | |
| P20 / PTO0 | Input mode: Connect to $V_{SS}$ or $V_{DD}$.<br>Output mode: Do not connect. |
| P21 | |
| P22 / PCL | |
| P23 / BUZ | |
| P30–P33 | |
| P40–P43 | |
| P50–P53 | |
| P60–P63 | |
| P70–P73 | |
| P80-P81 | |
| XT1 | Connect XT1 to $V_{SS}$ or $V_{DD}$. |
| XT2 | Do not connect XT2. |

## 2.5 Mask Option Selection

The pins contain the mask option function as listed in Table 2.5-1.
However, mask option is not contained in the μPD75P008.

### Table 2.5–1 Mask option selection

| Pin names | Mask option |
|---|---|
| P40–P43,<br>P50–P53 | Pull-up resistor is included.<br>specified bit-wise.<br>Pull-up resistor is not included. |

## 3. ARCHITECTURE AND MEMORY MAP

The μCOM-75X architecture for the μPD7500X adopts data memory bank configuration and memory mapped I / O to provide features such as:

- Internal RAM with maximum of 4K words x four bits (12-bit addresses)
- Peripheral hardware extensibility

This chapter covers these topics.

### 3.1 Data Memory Bank Configuration and Addressing Modes

3.1.1 Data memory bank configuration

A general purpose static RAM (512 words x four bits) is incorporated in data memory space addresses 000H to 1FFH. Peripheral hardware such as input/output ports and timers is allocated to addresses F80H to FFFH.

To adress the data memory space (4K words x four bits) with 12-bit addresses, the μPD7500X adopts a memory bank configuration where the low-order eight bits of an address are specified either directly or indirectly by using an instruction, and the high-order four bits of an address are specified by using a memory bank.

To specify the memory bank (MB), two hardware devices are incorporated in the μPD7500X:

- Memory bank enable flag (MBE)
- Memory bank selection register (MBS)

MBS is a register used to select a memory bank. The μPD7500X allows 0, 1, or 15 to be set in MBS. MBE is a flag used to determine whether or not the memory bank selected by using MBS is validated. As shown in Fig. 3.1–1, when MBE is set to 0, the specified memory bank is fixed regardless of how MBS is set; when MBE is set to 1, memory bank switching can be performed to extend data memory space by setting MBS.

In data memory space addressing, normally MBE is set to 1, and the data memory area of the memory bank specified by using MBS is handled. An efficient program can be prepared by using MBE = 0 mode and MBE = 1 mode appropriately in each process of the program.

| | Applicable program processing | Effects |
|---|---|---|
| MBE = 0 mode | ° Interrupt service | MBS save and restore are made unnecessary. |
| | ° Repetitive processing of internal hardware operation and general purpose RAM operation | MBS change is made unecessary. |
| | ° Subroutine processing | MBS save and restore are made unnecessary. |
| MBE = 1 mode | ° Normal program processing | |

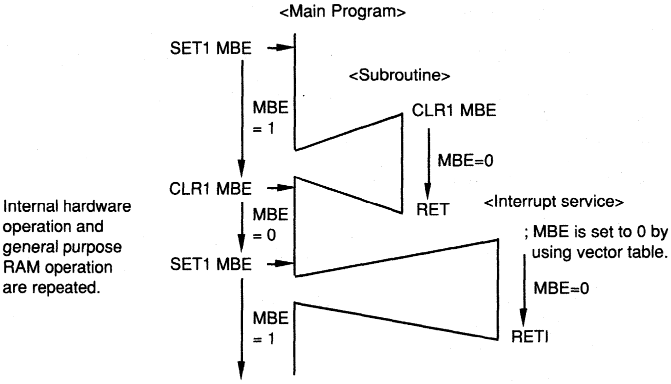

<Main Program>
SET1 MBE
<Subroutine>
MBE = 1
CLR1 MBE
MBE=0
CLR1 MBE
RET
<Interrupt service>
Internal hardware operation and general purpose RAM operation are repeated.
MBE = 0
; MBE is set to 0 by using vector table.
SET1 MBE
MBE=0
MBE = 1
RETI

**Figure 3.1–1: Proper Use of MBE = 0 and MBE = 1 Modes**

MBE is saved or restored automatically during subroutine processing and can be changed as desired. When interrupt service is made, MBE is automatically saved or restored. MBE during interrupt service can also be specified at the same time interrupt service starts by setting the interrupt vector table. Thus, it is useful for high-speed interrupt service.

To change the MBS setting in subroutine processing or interrupt service, MBS is saved and restored by using PUSH and POP instructions.

MBE is set and reset by using SET1 and CLR1 instructions.
MBS is set by using the SEL instruction.

Example 1: To clear MBE and fix memory bank

```
          CLR1   MBE  ; MBE ← 0
```

Example 2: To select memory bank 1

```
          SET1   MBE  ; MBE ← 1
          SEL    MB1  ; MBS ← 1
```

3.1.2 Data memory addressing modes
The µCOM-75X architecture adopted for the µPD7500X provides seven addressing modes, as shown in Fig. 3.1–2, for efficiently addressing the data memory space for each bit length of data to be processed.

(1) 1-bit direct adressing (mem. bit)

This addressing mode directly specifies each bit in all the data memory space by using the instruction operand.

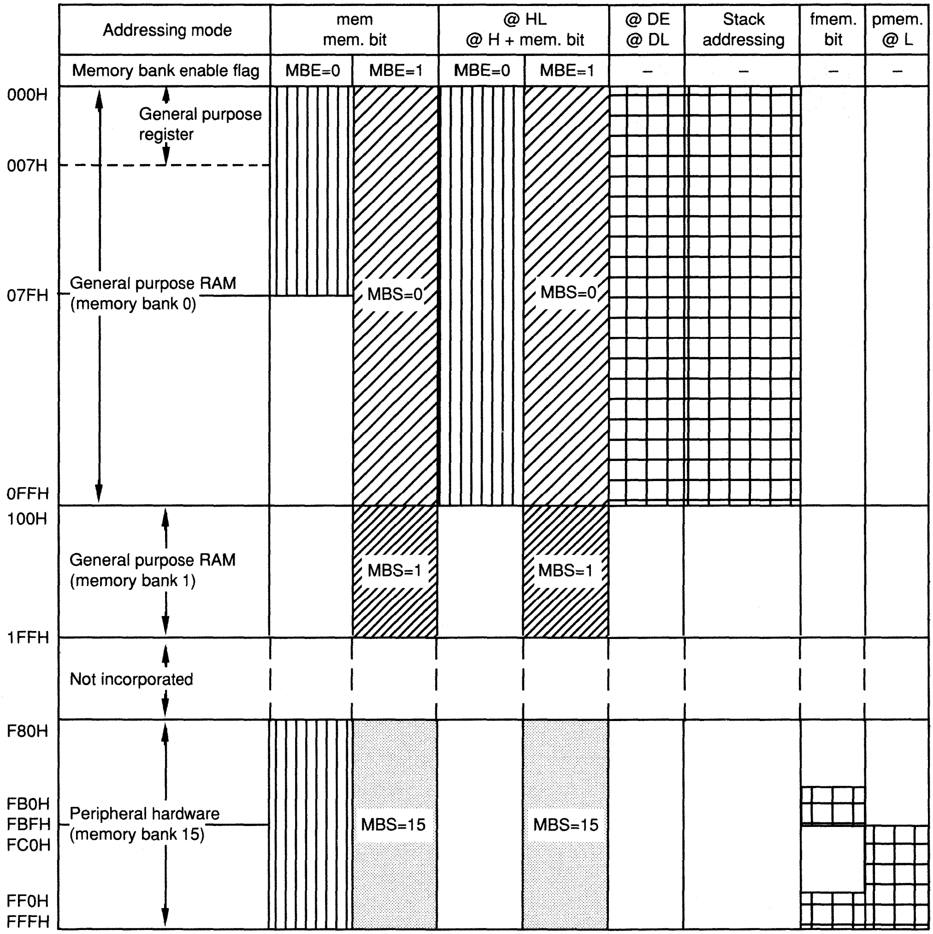
When MBE is set to 0, the specified memory bank (MB) is fixed to memory bank 0 (MB0) if the address specified in the operand is 00H-7FH; it is fixed to memory bank 15 (MB15) if the address specified in the operand is 80H-FFH. Thus, when MBE is set to 0, both the general purpose RAM area (000H-07FH) and peripheral hardware area (F80H-FFFH) can be addressed.

When MBE is set to 1, any memory bank can be specified by setting MBS; data memory space that can be specified can be extended.

The 1-bit direct addressing mode is applicable to bit set and reset instructions (SET1 and CLR1) and bit test instructions (SKT and SKF).

Example:
To set FLAG1, reset FLAG2, and test whether or not FLAG3 is set to 0.

```
FLAG1   EQU     03FH. 1   ;   3FH address bit 1
FLAG2   EQU     087H. 2   ;   87H address bit 2
FLAG3   EQU     0A7H. 0   ;   A7H address bit 0
        SET1    MBE       ;   MBE    ← 1
        SEL     MB0       ;   MBS    ← 0
        SET1    FLAG1     ;   FLAG1 ← 1
        CLR1    FLAG2     ;   FLAG2 ← 0
        SKF     FLAG3     ;   FLAG3 = 0 ?
```



–: don`t care

**Figure 3.1–2 Data Memory Configuration and Addressing Range of each Addressing Mode**

| Addressing mode | Representation format | Specified address |
|---|---|---|
| 1-bit direct addressing | mem. bit | Bit indicated by a bit at the address indicated by MB and mem. However,<br>MBE = 0, MB = 0 when mem = 00H-7FH<br>MB = 15 when mem = 80H-FFH<br>When MBE =1, MB = MBS |
| 4-bit direct addressing | mem | Address indicated by MB and mem. However,<br>MBE = 0, MB = 0 when mem = 00H-7FH<br>MB = 15 when mem = 80H-FH<br>When MBE = 1, MB = MBS |
| 8-bit direct addressing | | Address indicated by MB and mem (mem is an even address). However,<br>MBE = 0, MB = 0 when mem = 00H-7FH<br>MB = 15 when mem = 80H-FFH<br>When MBE = 1, MB = MBS |
| 4-bit register indirect addressing | @ HL | Address indicated by MB and HL. However, MB = MBE • MBS |
| | @ DE | Memory bank 0 address indicated by DE |
| | @ DL | Memory bank 0 address indicated by DL |
| 8-bit register indirect addressing | @ HL | Address indicated by MB and HL (the L register contains an even number). However, MB = MBE • MBS |
| Bit manipulation addressing | fmem. bit | Bit indicated by a bit at the address indicated by fmem.<br>However, fmem FB0H-FBFH (hardware related to interrupt)<br>FF0H-FFFH (I/O port) |
| | pmem. @ L | Bit indicated by the low-order two bits of the L register at the address indicated by the high-order 10 bits of pmem and the high-order two bits of the L register.<br>However, pmem = FC0H-FFFH |
| | @ H + mem. bit | Bit indicated by a bit at the address indicated by MB, H, and the low-order four bits of mem.<br>However, MB = MBE • MBS |
| Stack addressing | | Memory bank 0 address indicated by SP. |

**Figure 3.1–3 Addressing Modes**

(2) 4-bit direct addressing (mem)

This addressing mode directly specifies all the data memory space in 4-bit units by using the instruction operand.
As with the 1-bit direct addressing mode, when MBE is set to 0, the 4-bit direct address mode allows that only the general purpose RAM area (000H-07FH) and peripherial hardware area (F80H-FFFH) can be specified. When MBE is set to 1, any memory bank (MB) can be specified by setting MBS and the data memory space that can be specified is extended to all the space.
The 4-bit direct addressing mode is applicable to the MOV, XCH, INCS, IN, and OUT instructions.

Example 1: To input port 4 and store in "DATA1".

```
DATA1 EQU    5FH         ;  "DATA1" is address 5FH.
      CLR1   MBE         ;  MBE← 0
      IN     A, PORT4    ;  A← PORT4
      MOV    DATA1, A    ;  (DATA1)← A
```

Example 2: To output data in "BUFF" to port 5.

```
 BUFF EQU    11AH        ;  "BUFF" is address 11AH
      SET1   MBE         ;  MBE← 1
      SEL    MB1         ;  MBS← 1
      MOV    A, BUFF     ;  A← (BUFF)
      SEL    MB15        ;  MBS← 15
      OUT    PORT5, A    ;  PORT5← A
```

Caution:
If data related to the input /output ports is stored in general purpose RAM of bank 1 as in this example, the program is made less efficient. If data related to the input /output ports is stored in addresses 00H-7FH of bank 0, a program can be prepared without changing MBS as in Example 1.

(3) 8-bit direct addressing (mem)

This addressing mode directly specifies all the data memory space in 8 bit units by using the instruction operand. Only even address can be specified in the mem operand. The 4-bit data at the address specified in the operand and the 4-bit data at the address + 1 are paired and transferred to the 8-bit accumulator (XA register pair) for 8-bit processing.
The memory banks specified in the addressing mode are the same as in the 4-bit direct addressing mode.
The 8-bit direct addressing mode is applicable to the MOV, XCH, IN, and OUT instructions.

Example 1: To transfer 8-bit data in ports 4 and 5 to addresses 20H and 21H.

```
 DATA EQU    020H
      CLR1   MBE         ;  MBE← 0
      IN     XA, PORT4   ;  X← port 5, A← port 4
      MOV    DATA, XA    ;  (21H)← X, (20H)← A
```

Example 2: To read 8-bit data input to the serial interface shift register (SIO), set transfer data, and start of transfer.

```
      SEL    MB15        ;  MBS← 15
      XCH    XA, SIO     ;  XA↔ (SIO)
```

(4) 4-bit register indirect addressing (@rpa)

This addressing mode indirectly specifies the data memory space in 4-bit units by using the data pointer (general purpose register pair) specified in the instruction operand.

The data pointers are the HL register pair, which enables all the data memory space to be specified according to MB = MBE • MBS specification, and the DE and DL register pairs, which always indicate memory bank 0 regardless of how MBE and MBS are set. Efficient programs can be prepared by using the data memory bank to be used.

Example: To transfer data at 50H-57H to 110H-117H.

```
DATA1  EQU   57H
DATA2  EQU   117H
       SET1  MBE
       SEL   MB1
       MOV   D, #DATA1 SHR 4
       MOV   HL, #DATA2 AND 0FFH ; HL← 17H
LOOP:  MOV   A, @DL              ; A← (DL)
       XCH   A, @HL              ; A← (HL)
       DECS  L                   ; L← L–1
       BR    LOOP
```

The addressing mode using the HL register pair for the data pointer has wide applications such as data transfer, operation, comparison, and input / output. The addressing mode using the DE or DL register pair is applicable to the MOV and XCH instructions.

As shown in Fig. 3.1–4, data memory space addresses can be updated as desired by combining the addressing mode with general purpose register (or register pair) increment and decrement instructions.

Example 1: To compare data at 50H-57H with data at 110H-117H.

```
DATA1  EQU   57H
DATA2  EQU   117H
       SET1  MBE
       SEL   MB1
       MOV   D, #DATA1 SHR 4
       MOV   HL, #DATA2 AND 0FFH
LOOP:  MOV   A, @DL
       SKE   A, @HL    ;  A = (HL) ?
       BR    NO        ;  NO
       DECS  L         ;  YES, L← L–1
       BR    LOOP
```

Example 2: To clear data memory area 04H-FFH.

```
       SEL   MB0
       MOV   XA, #00H
       MOV   L, #04H
LOOP:  MOV   @HL, A    ;  (HL)← A
       INCS  L         ;  L← L+1
       BR    LOOP
       INCS  H         ;  H← H+1
       BR    LOOP
```

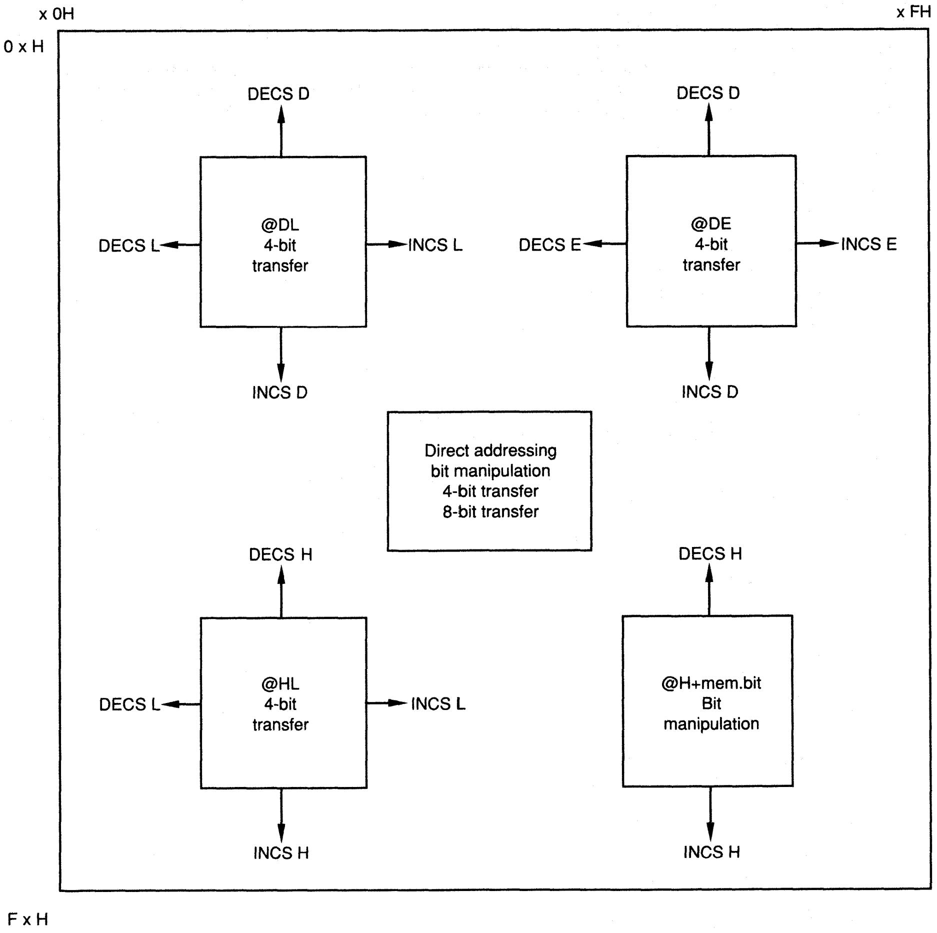


**Figure 3.1–4 General Purpose RAM Address Update Method**

(5) 8-bit register indirect addressing (@HL)

This addressing mode indirectly specifies all the data memory space in 8-bit units by using the data pointer (HL register pair). The 4-bit data at the address setting data pointer bit 0 (L register bit 0) to 0 and the 4-bit data at the address + 1 are paired and transferred to the 8-bit accumulator (XA register) for 8-bit processing.

The memory banks specified in the addressing mode are the same as those when the HL register is specified in the 4-bit register indirect addressing mode (MB = MBE • MBS). The 8-bit register indirect addressing mode is applicable to the MOV, XCH, and SKE Instructions.

Example 1: To compare the count register (T0) value of timer / event counter 0 with data at addresses 30H and 31H for equality.

```
DATA  EQU    30H
      CLR1   MBE
      MOV    HL, #DATA
      MOV    XA, T0      ;   XA←— count register 0
      SKE    A, @HL      ;   A = (HL) ?
      BR     NO
      INCS   L
      MOV    A, X        ;   A←— X
      SKE    A, @HL      ;   A = (HL) ?
```

(6) Bit manipulation addressing

This addressing mode is used to perform bit manipulations such as Boolean operation or bit transfer on any bit in all of the data memory space.

Although the 1-bit direct addressing mode is applicable only to the bit set, reset, and test instructions, the bit manipulation addressing mode enables bit manipulations such as Boolean operations by using the AND1, OR1, and XOR1 instructions and allows bit test and bit reset by using the SKTCLR instructions.

The following three types of bit manipulation addressing modes can be used according to the data memory address to be used:

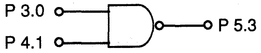
(a) Specific address bit direct addressing (fmem. bit).

This addressing mode enables, peripheral hardware that frequently uses bit manipulation, such as input / output ports and interrupt flags, to be operated at any time independently of the memory bank setting. Thus, the addressing mode is applicable to data memory addresses FF0H-FFFH (where input / output ports are mapped) and FB0H-FBFH (where hardware related to interrupts is mapped). For the hardware of the two data memory areas, bit manipulation can be performed in direct addressing at any time as desired regardless of how MBS and MBE are set.

Example 1: To test the timer 0 interrupt request flag (IRQT0); if the request flag is set, clear the flag and reset P63.

```
SKTCLR   IRQT0     ;   IRQT0 = 1 ?
BR       NO        ;   NO
CLR1     PORT6.3   ;   YES
```

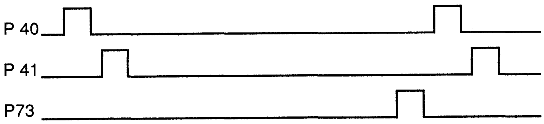
Example 2: To reset P53 if both P30 and P41 are set to 1.



```
(i)       SET1   CY              ;   CY ← 1
          AND1   CY, PORT3.0     ;   CY P3.0
          AND1   CY, PORT4.1     ;   CY P4.1
          SKT    CY              ;   CY = 1 ?
          BR     SETP
          CLR1   PORT5.3         ;   P53 ← 1
           :
           :
 SETP:    SET1 PORT5.3           ;   P53 ← 1
           :
           :

(ii)      SKT    PORT3.0         ;   P30 = 1 ?
          BR     SETP
          SKT    PORT4.1         ;   P41 = 1 ?
          BR     SETP
          CLR1   PORT5.3         ;   P53 ← 0
           :
           :
STEP:     SET1   PORT5.3         ;   P53 ← 1
```

(b) Specific address bit register indirect addressing (pmem. @L).
This addressing mode indirectly specifies each bit of peripheral hardware, such as an input / output port, by using the L register for successive operation. The addressing mode is applicable to data memory addresses FC0H-FFFH.
The addressing mode directly specifies the high-order 10 bits of a 12-bit data memory address in the pmem operand and indirectly specifies the low-order 2-bit data memory address part and the bit address by using the L register. Thus, 16 bits (four ports) can be manipulated (operated) successively by L register specification.
The address mode also enable bit manipulation to be performed at any time independently of how MBE and MBS are set.

Example: To output a pulse to each bit of ports 4 to 7 in order.

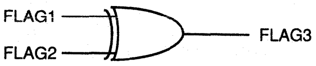


```
         MOV    L, #0
LOOP:    SET1   PORT4. @L     ;   Port 4–7 bit (L1-0) ← 1
         CLR1   PORT4. @L     ;   Port 4–7 bit (L1-0) ← 0
         INCS   L
         BR     LOOP
```

(c) Special 1-bit direct addressing (@H + mem. bit)

This addressing mode enables bit manipulation to be performed on any bit in all of the data memory space.
The addressing mode indirectly specifies the high-order four bits of data memory address of the memory bank specified by MB = MBE MBS by using the H register and directly specifies the low-order 4-bit data memory address part and the bit address in the operands. It enable various types of bit manipulation to be performed on any bit in all of the data memory space.

Example: To reset address 32H bit 2 (FLAG3) if both address 30H bit 3 (FLAG1) and address 31H bit 0 (FLAG2) are set to 0 or 1.



```
FLAG1 EQU   30H.3
FLAG2 EQU   31H.0
FLAG3 EQU   32H.2
      SEL   MB0
      MOV   H, #FLAG1 SHR 6
      CLR1  CY
      OR1   CY, @H + FLAG1   ;   CY← 0
      XOR1  CY, @H + FLAG2   ;   CY← CY-FLAG1
      SET1  @H + FLAG3       ;   CY← CY-FLAG2
      SKT   CY               ;   CY = 1 ?
      CLR1  @H + FLAG3       ;   FLAG← 30
```

(7) Stack addressing

The stack addressing mode is used for register save and restore during interrupt service or subroutine processing.
The processing mode specifies an address by using the 8-bit stack pointer (data memory bank 0).
The addressing mode can also be used for register save and restore by executing the PUSH and POP instructions.

Example 1: To save and restore register in subroutine processing.

```
SUB: PUSH  XA
     PUSH  HL
     PUSH  BS   ;  MBS is saved.
       :
       :
     POP   BS
     POP   HL
     POP   XA
     RET
```

Example 2: To transfer the HL register pair contents to the DE register pair.

```
PUSH  HL
POP   DE   ;  DE← HL
```

Example 3: To branch to the address indicated by [XABC] register.

```
PUSH  BC
PUSH  XA
RET        ;  Branch to XABC address
```

## 3.2 Memory Mapped I / O

The μPD7500X adopts memory mapped I / O where peripheral hardware such as input / ports and timers is allocated to addresses F80H-FFFH of the data memory space as shown in Fig. 3.1–2. Thus, peripheral hardware is controlled entirely by memory operation instructions rather than special instructions. (To easily understand programs, some hardware control mnemonics are provided.)
Table 3.2–1 lists the addressing modes that can be used to operate peripheral hardware.

**Table 3.2–1 Applicable Addressing Modes during Peripheral Hardware Operation**

| | Applicable addressing mode | Applicable hardware |
|---|---|---|
| Bit manipulation | With MBE=0 or (MBE=1 and MBS=15), direct addressing (specification in mem. bit). | All hardware where bit manipulation can be performed. |
| | Direct addressing regardless of how MBE and MBS are set. (specification in fmem. bit) | ISTO, MBE<br>IExxx, IRQxxx, PORTn.x |
| | Indirect addressing regardless of how MBE and MBS are set. (specification in pmem. @L) | BSBn.x<br>PORTn.x |
| 4-bit manipulation | With MBE=0 or (MBE=1 and MBS=15), direct addressing (specification in mem). | All hardware where 4-bit manipulation can be performed. |
| | With (MBE=1 and MBS=15), register indirect addressing (specification in @HL). | |
| 8-bit manipulation | With MBE=0 or (MBE=1 and MBS=15), direct addressing (specification in mem), mem must be an even address. | All hardware where 8-bit manipulation can be performed. |
| | With MBE=1 and MBS=15, register indirect addressing (specification in @HL; the L register must contain an even number. | |

```
Example: CLR1     MBE         ;  MBE = 0
         SET1     TM0.3       ;  Timer 0 starts.
         EI       IE0         ;  INT0 is enabled.
         DI       IE1         ;  INT1 is disabled.
         SKTCLR   IRQ2        ;  INT2 request flag is tested and cleared.
         SET1     PORT4. @L   ;  Port 4 is set.
         IN       A, PORT0    ;  A ← port 0
         OUT      PORT4, XA   ;  Port 5, 4 ← XA
```

Figs. 3.2–1 to 3.2–3 shows the μPD7500X I/O map.
The columns in the figures mean:

- Abbreviation:
  Name indicating internal hardware address. It can be entered in the instruction operand field.

- R/W
  Indicates whether the hardware device can be read or written.
  – R/W : Read and write are enabled.
  – R : Read only is enabled.
  – W : Write only is enabled.

- Number of bits that can be manipulated:
  Indicates the number of bits that can be processed when the hardware device is operated.
  O: Bit manipulation is enabled in 1-4, 4-, or 8-bit units as specified in the column.
  Δ: Only some bits can be manipulated. See Remarks for the bits that can be manipulated.
  –: Bit manipulation cannot be performed in 1-, 4-, or 8-bit units as specified in the column.

- Bit manipulation addressing:
  Indicates the applicable bit manipulation addressing for performing bit manipulation on the hardware device.

| Address | Hardware name (abbreviation) | | | | R/W | Number of bits that can be manipulated | | | Bit manipulation addressing | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| | b3 | b2 | b1 | b0 | | One bit | Four bit | Eight bit | | |
| F80H | Stack pointer (SP) | | | | R/W | – | – | O | | Bit 0 is fixed to 0. |
| | | | | | | | | | | |
| F85H | Basic interval timer mode register (BTM) | | | | W | Δ | O | – | mem. bit | Bit manipulation can be performed only on bit 3. |
| F86 H | Basic interval timer (BT) | | | | R | – | – | O | | |
| | | | | | | | | | | |
| F98H | Watch mode register (WM) | | | | R/W | Δ | – | O | mem. bit | Bit test can be made only on bit 3. |
| | | | | | W | – | – | | | |
| FA0H | Timer/event counter 0 mode register (TM0) | | | | W | Δ | – | O | mem. bit | Bit manipulation can be performed only on bit 3. |
| | | | | | | – | – | | | |
| FA2H | TOE0 (Note) | | | | W | O | – | – | mem. bit | |
| FA4H | Timer/event counter 0 count register (T0) | | | | R | – | – | O | | |
| | | | | | | – | – | | | |
| FA6H | Timer/event counter 0 modulo register (TMOD0) | | | | W | – | – | O | | |
| | | | | | | – | – | | | |
| FB0H | 0 | IST0 | MBE | 0 | R/W | O | O | O | fmem. bit | |
| | Program status word (PSW) | | | | R | – | – | | | |
| FB2H | (IME) | | | | – | – | – | – | | EI and DI instructions are used |
| FB3H | Processor clock control register (PCC) | | | | W | – | O | | | |
| FB4H | INT0 mode register (IM0) | | | | W | – | O | – | | Bit 2 is fixed to 0. |
| FB5H | INT1 mode register (IM1) | | | | W | – | O | | | Bit 3 to 1 are fixed to 0. |
| FB6H | INT2 mode register (IM2) | | | | W | – | O | | | Bits 3 and 2 are fixed to 0. |
| FB7H | System clock control register (SCC) | | | | W | O | – | – | | Only bits 3 and 0 can be bit manipulated |

Note: TOE0 = timer / event counter 0 output enable flag (W)

(to be continued)

**Figure 3.2–1 µPD7500X I/O Map**

| Address | Hardware name (abbreviation) | | | | R/W | Number of bits that can be manipulated | | | Bit manipulation addressing | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| | b3 | b2 | b1 | b0 | | One bit | Four bit | Eight bit | | |
| FB8H | IE4 | IRQ4 | IEBT | IRQBT | R/W | O | O | – | fmem. bit | |
| FBAH | | | IEW | IRQW | R/W | O | O | | | |
| FBCH | | | IET0 | IRQT0 | R/W | O | O | – | | |
| FBDH | | | IECSI | IRQCSI | R/W | O | O | | | |
| FBEH | IE1 | IRQ1 | IE0 | IRQ0 | R/W | O | O | – | | |
| FBFH | | | IE2 | IRQ2 | R/W | O | O | | | |
| FC0H | Bit sequential buffer 0 (BSB0) | | | | R/W | O | O | O | mem. bit<br>pmem @L | |
| FC1H | Bit sequential buffer 1 (BSB1) | | | | R/W | O | O | | | |
| FC2H | Bit sequential buffer 2 (BSB2) | | | | R/W | O | O | O | | |
| FC3H | Bit sequential buffer 3 (BSB3) | | | | R/W | O | O | | | |
| FD0H | Clock output mode register (CLOM) | | | | W | – | O | – | | |
| FDCH | Pull-up resistor specification register group A (POGA) | | | | W | – | – | O | | |
| FDEH | Pull-up resistor specification register group B (POGB) | | | | W | – | – | O | | |
| FE0H | Serial operation mode register (CSIM) | | | | W | – | – | O | | |
| | CSIE | COI | WUP | | R/W | O | O | | mem. bit | |
| FE2H | CMDD | RELD | CMDT | RELT | R/W | O | – | – | mem. bit | Only bit manipulation can be performed on any bit. |
| | SBI control register (SBIC) | | | | | | | | | |
| | BSYE | ACKD | ACKE | ACKT | | | | | | |
| FE4H | Serial I/O shift register (SIO) | | | | R/W | – | – | O | | |
| FE6H | Slave address register (SVA) | | | | W | – | – | O | | |
| FE8H | PM33 | PM32 | PM31 | PM30 | W | – | – | O | | |
| | Port mode register group A (PMGA) | | | | | | | | | |
| | PM63 | PM62 | PM61 | PM60 | | | | | | |
| FECH | – | PM2 | – | – | W | – | – | O | | |
| | Port mode register group B (PMGB) | | | | | | | | | |
| | PM7 | 0 | PM5 | PM4 | | | | | | |
| FEEH | – | – | – | PM8 | W | – | – | O | | |
| | Port mode register group C (PMGC) | | | | | | | | | |
| | – | – | – | – | | | | | | |

Remarks: 1. IExxx is an interrupt enable flag.
2. IRQxxx is an interrupt request flag.
3. IME is an interrupt master flag.

(to be continued)

**Figure 3.2–1 µPD7500X I/O Map** (con`t)

| Address | Hardware name (abbreviation) | | | | R/W | Number of bits that can be manipulated | | | Bit manipulation addressing | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| | b3 | b2 | b1 | b0 | | One bit | Four bit | Eight bit | | |
| FF0H | Port 0 (PORT 0) | | | | R | O | O | – | fmem. bit<br>pmem. @L | |
| FF1H | Port 1 (PORT 1) | | | | R | O | O | | | |
| FF2H | Port 2 (PORT 2) | | | | R/W | O | O | – | | |
| FF3H | Port 3 (PORT 3) | | | | R/W | O | O | | | |
| FF4H | Port 4 (PORT 4) | | | | R/W | O | O | O | | |
| FF5H | Port 5 (PORT 5) | | | | R/W | O | O | | | |
| FF6H (Note) | KR3 | KR2 | KR1 | KR0 | R/W | O | O | O | | |
| | Port 6 (PORT 6) | | | | | | | | | |
| FF7H (Note) | KR7 | KR6 | KR5 | KR4 | R/W | O | O | | | |
| | Port 7 (PORT 7) | | | | | | | | | |
| FF8H | Port8 (PORT8) | | | | R/W | O | O | – | | |

Note: KR0–KR7 can only be read. When 4bit parallel is input, specified at PORT6 or PORT7.

**Figure 3.2–1 μPD7500X I/O Map** (con't)

## 4 INTERNAL CPU FUNCTIONS

### 4.1 Program Counter (PC) –12–Bit (μPD75004)
### 13–Bit (μPD75006, 75008)

The program counter is binary counter which holds program memory address information. The μPD75004 is configured of 12-bit (See Fig. 4.1–1 (a)). The μPD75006 and 75008 are configured of 13-bit (See Fig. 4.4–1 (b)).0

(a) μPD75004 format

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

(b) μPD75006 and 75008 format

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

**Figure 4.1–1 Program Counter Format**

Each time one instruction is executed, normally the program counter is automatically incremented according to the number of the bytes in the instruction.
When a branch instruction (BR or BRCB) is executed, the register pair contents or immediate data indicating the branch destination address is loaded into all or some of the PC bits.

When a subroutine call instruction (CALL or CALLF) is executed or a vectored interrupt occurs, the current PC contents (return address already incremented to fetch the next instruction) are saved in stack memory (data memory indicated by the stack pointer), then the jump destination address is loaded.
When a return instruction (RET, RETS, or RETI) is executed, the stack memory contents are loaded into the PC.
When the $\overline{\text{RESET}}$ signal is generated, the program memory contents are loaded into the program counter for initialization, as follows: (Program can be started at any desired address.)

| | | | |
|---|---|---|---|
| µPD75004: | PC11–PC8 | ← | low-order four bits at address 0000H |
| | PC7 –PC0 | ← | eight bits at address 0001H |
| µPD75006, µPD75008: | PC12–PC8 | ← | low-order five bits at address 000H |
| | PC7 –PC0 | ← | eight bits at address 0001H |

## 4.2 Program Memory (ROM) - 4096 Words x Eight bit (µPD75004)<br>6016 Words x Eight bit (µPD75006)<br>8064 Words x Eight bit (µPD75008)

The program memory stores programs, interrupt vector table, GETI instruction look up table, and data such as table data. The program memory of the µPD75004, µPD75006, µPD75008, is mask programmable ROM.
Fig. 4.2–1 to 4.2–5 show the program memory map.
The program memory is addressed by using the program counter. Table data can also be referenced by using the table reference instruction (MOVT).
The address range for branching by a branch or subroutine call instruction is as shown in Fig. 4.2–1. When a relative branch instruction (BR $addr) is used, a branch can be made to [PC contents – 15 to –1, +2 to + 16] address independently of block.
The program memory addresses are shown below.

| | | |
|---|---|---|
| • 000H– FFFH | : | µPD75004 |
| • 0000H–177FH | : | µPD75006 |
| • 0000H–1F7FH | : | µPD75008 |

The following addresses are assigned for special purpose:
(All the area except address 0000H to 0001H [Note 1] can be used as normal program memory area.).

• 0000H – 0001H
Vector address table in which the program start address and MBE setup value are entered when the system is reset. (Reset start can be made at any desired address.)

• 0002H – 000BH
Vector address table in which the program start address and MBE setup value are entered for each vectored interrupt. (Interrupt service can be started at any desired address.)

• 0020H – 007FH Table area referenced by GETI [(Note 2)].

Note: 1. Their addresses are used in the µPD7500X.
2. The GETI instruction is used to convert any 2- or 3-byte instruction or any two 1-byte instructions into a 1-byte instruction. The number of program bytes can be reduced.

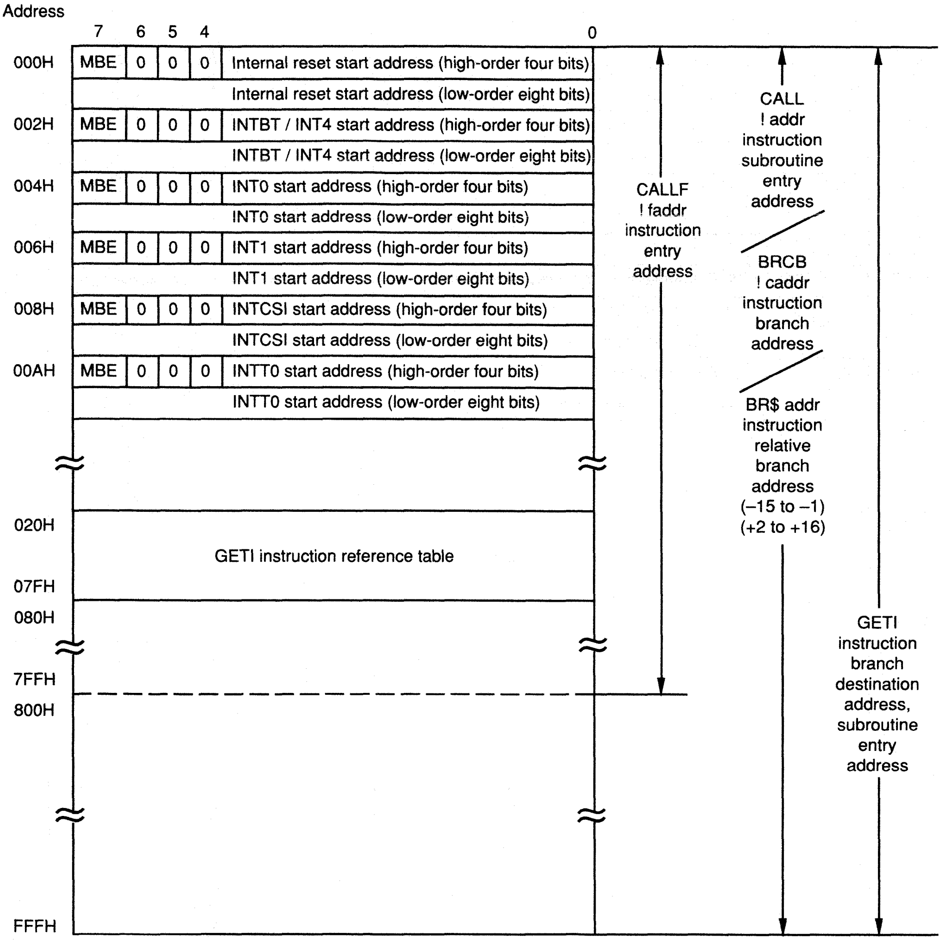
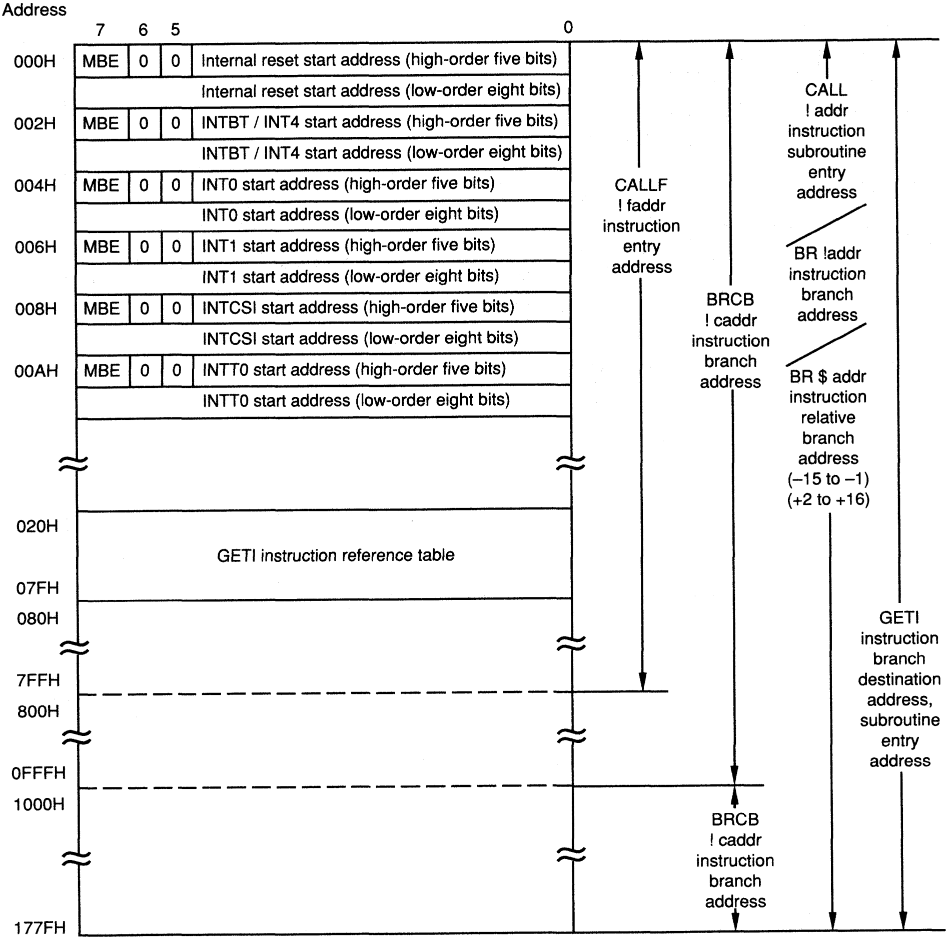

Address
7
6
5
4
0
000H
MBE
0
0
0
Internal reset start address (high-order four bits)
Internal reset start address (low-order eight bits)
002H
MBE
0
0
0
INTBT / INT4 start address (high-order four bits)
INTBT / INT4 start address (low-order eight bits)
004H
MBE
0
0
0
INT0 start address (high-order four bits)
INT0 start address (low-order eight bits)
006H
MBE
0
0
0
INT1 start address (high-order four bits)
INT1 start address (low-order eight bits)
008H
MBE
0
0
0
INTCSI start address (high-order four bits)
INTCSI start address (low-order eight bits)
00AH
MBE
0
0
0
INTT0 start address (high-order four bits)
INTT0 start address (low-order eight bits)
020H
GETI instruction reference table
07FH
080H
7FFH
800H
FFFH
CALLF ! faddr instruction entry address
CALL ! addr instruction subroutine entry address
BRCB ! caddr instruction branch address
BR$ addr instruction relative branch address (–15 to –1) (+2 to +16)
GETI instruction branch destination address, subroutine entry address

**Figure 4.2–1 Program Memory Map (µPD75004)**



| Address | 7 | 6 | 5 … 0 |
|---|---|---|---|
| 000H | MBE | 0 | 0 | Internal reset start address (high-order five bits) |
| | | | Internal reset start address (low-order eight bits) |
| 002H | MBE | 0 | 0 | INTBT / INT4 start address (high-order five bits) |
| | | | INTBT / INT4 start address (low-order eight bits) |
| 004H | MBE | 0 | 0 | INT0 start address (high-order five bits) |
| | | | INT0 start address (low-order eight bits) |
| 006H | MBE | 0 | 0 | INT1 start address (high-order five bits) |
| | | | INT1 start address (low-order eight bits) |
| 008H | MBE | 0 | 0 | INTCSI start address (high-order five bits) |
| | | | INTCSI start address (low-order eight bits) |
| 00AH | MBE | 0 | 0 | INTT0 start address (high-order five bits) |
| | | | INTT0 start address (low-order eight bits) |
| 020H–07FH | GETI instruction reference table | | |

Address
7
6
5
0
020H
07FH
080H
7FFH
800H
0FFFH
1000H
177FH
CALLF ! faddr instruction entry address
BRCB ! caddr instruction branch address
CALL ! addr instruction subroutine entry address
BR !addr instruction branch address
BR $ addr instruction relative branch address (−15 to −1) (+2 to +16)
GETI instruction branch destination address, subroutine entry address
BRCB ! caddr instruction branch address

Figure 4.2–2 Program Memory Map (μPD75006)

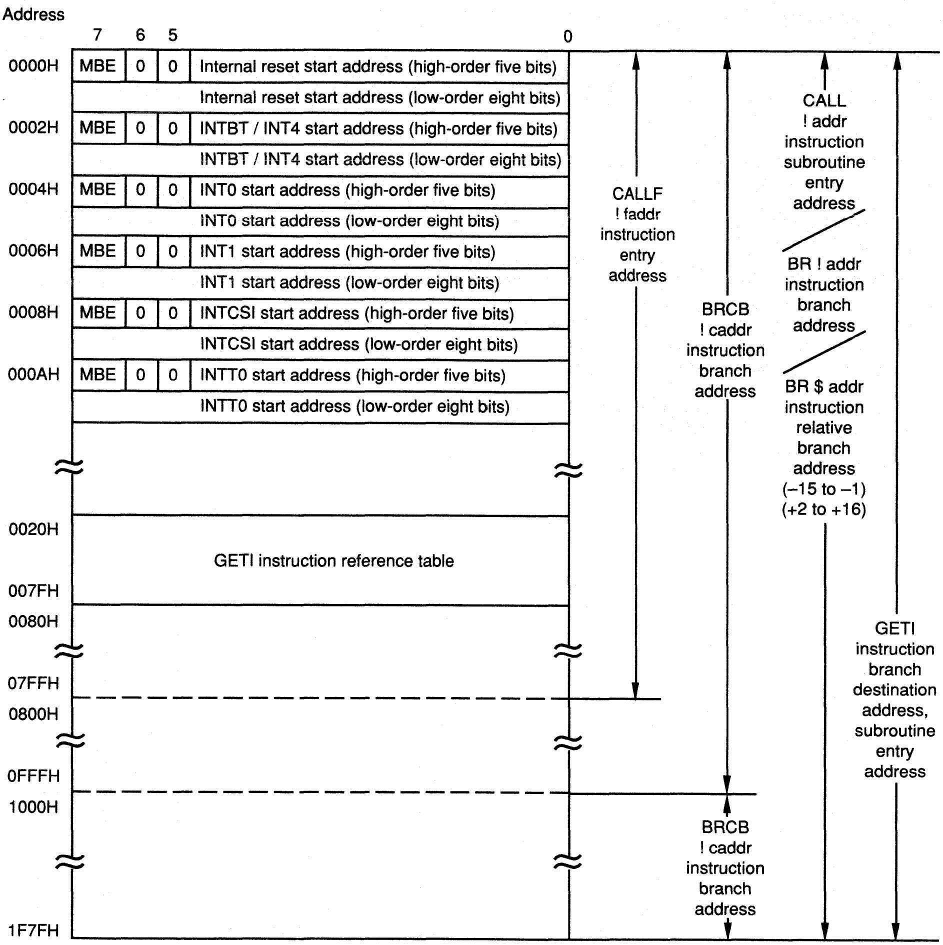

Address
7 6 5 0
0000H
MBE 0 0 Internal reset start address (high-order five bits)
Internal reset start address (low-order eight bits)
0002H
MBE 0 0 INTBT / INT4 start address (high-order five bits)
INTBT / INT4 start address (low-order eight bits)
0004H
MBE 0 0 INT0 start address (high-order five bits)
INT0 start address (low-order eight bits)
0006H
MBE 0 0 INT1 start address (high-order five bits)
INT1 start address (low-order eight bits)
0008H
MBE 0 0 INTCSI start address (high-order five bits)
INTCSI start address (low-order eight bits)
000AH
MBE 0 0 INTT0 start address (high-order five bits)
INTT0 start address (low-order eight bits)
0020H
GETI instruction reference table
007FH
0080H
07FFH
0800H
0FFFH
1000H
1F7FH
CALLF ! faddr instruction entry address
BRCB ! caddr instruction branch address
CALL ! addr instruction subroutine entry address
BR ! addr instruction branch address
BR $ addr instruction relative branch address (−15 to −1) (+2 to +16)
GETI instruction branch destination address, subroutine entry address
BRCB ! caddr instruction branch address

**Figure 4.2–3 Program Memory Map (μPD75008)**

### 4.3 Data Memory (RAM) – 512 Words x Four Bits

The data memory is a general purpose static RAM consisting of 512 words x four bits. Since a static RAM stores data during process, subroutine, or interrupt execution, it can also hold data when CPU operation is stopped in the standby mode; it is useful in that the memory contents can be held with battery power for hours.
Fig. 4.3–1 shows a μPD7500X data memory map.
The data memory adopts the bank configuration, consisting of banks 0 and 1 (each 256 words x four bits).
Peripheral hardware is mapped in the memory bank 15 area.
A memory bank is selected by setting the 4-bit memory bank selection register (MBS = 0, 1, or 15) when bank specification is enable by setting the memory bank enable flag (MBE) to 1 (MBE = 1). When bank specification is disabled (MBS = 0), memory bank 0 or 15 is automatically selected according to the current addressing mode. Each bank is addressed by using 8-bit immediate data, a register pair, etc.
Although a data memory word consists of four bits, the data memory can be handled in 1-, 4-, or 8-bit units by using various addressing modes.
See 3.1 for details of memory bank selection and addressing modes.
Specific areas of the data memory are also used for general purpose registers (bank 0; 000H – 0007H) and stack memory (bank 0; 000H – 0FFH). Data memory does not exist in bank 15. Peripheral hardware and various registers are mapped. Do not access addresses or bits to which no registers are allocated.

For the use of the specific areas of data memory, see the following:
- 4.4 for the general purpose register area
- 4.6 for the stack memory area
- CHAPTER 5 for the peripheral hardware

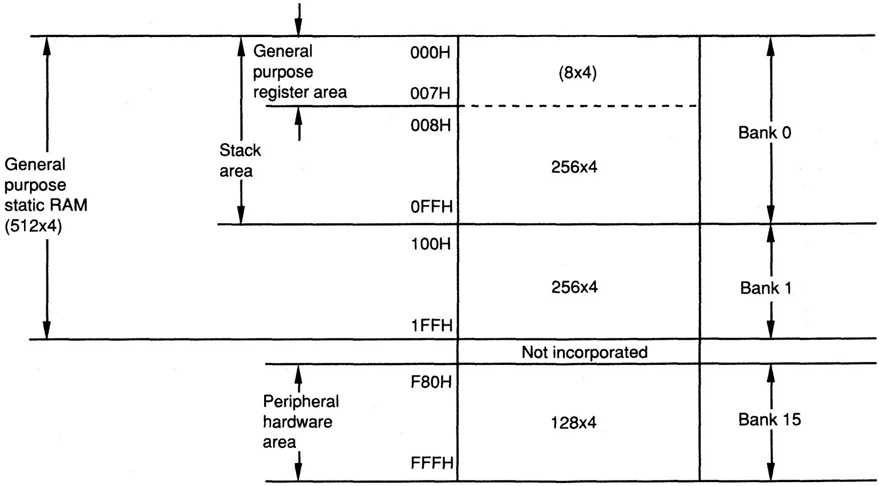


**Figure 4.3–1 Data Memory Map**

When reset is executed, the contents of the data memory are undefined. Therefore, normally, the data memory must be initialized at the beginning of the program execution (RAM clear). If this initialization is not performed, unexcepted malfunctions may result.

Example: To clear the RAM mapped to addresses 04H to 1FFH:

```
            SEL     MB0
            MOV     XA, #00H
            MOV     HL, #04H   ; Clears addresses 04H to FFH
RAMC0:      MOV     @HL, A     ; (HL) <-- A
            INCS    L          ; L = L + 1
            BR      RAMC0
            INCS    H          ; H = H + 1
            BR      RAMC0
            SEL     MB1
RAMC1:      MOV     @HL, A     ; Clears addresses 100H to 1FFH
            INCS    L
            BR      RAMC1
            INCS    H
            BR      RAMC1
             •
             •
```

### 4.4 General Purpose Registers – Eight x Four Bits

The general purpose registers are eight 4-bit registers (B, C, D, E, H, L, X, and A) mapped in specific addresses of the data memory. Every general purpose register is handled in 4-bit units; register pairs BC, DE, HL, and XA are also used for 8-bit manipulation. In addition to DE and HL, registers D and L are also paired (DL), and the three register pairs can be used for data pointers. The general purpose register area can be addressed and accessed as normal RAM regardless of whether or not it is used for registers.

| | |
|---|---|
| X<br>01H | A<br>00H |
| H<br>03H | L<br>02H |
| D<br>05H | E<br>04H |
| B<br>07H | C<br>06H |

**Figure 4.4–1 General Purpose Register Configuration (When 4-bit processing is performed)**

| |
|---|
| XA<br>00H |
| HL<br>02H |
| DE<br>04H |
| BC<br>06H |

**Figure 4.4–2 General Purpose Register Configuration (When 8-bit processing is performed)**

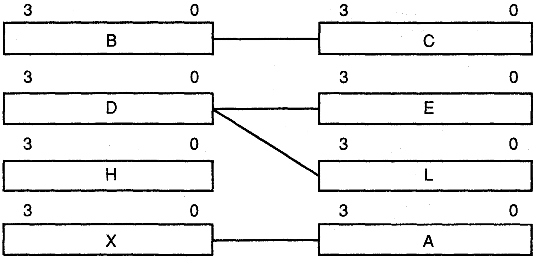


**Figure 4.4–3 Register Pair Configuration**

### 4.5 Accumulator

The μPD7500X uses the A register and XA register pair for accumulators. The A register is used as the main register during execution of 4-bit data processing instructions; the XA register pair is used as the main register pair during execution of 8-bit data processing instructions.
The carry flag (CY) is used for a bit accumulator during execution of bit manipulation instructions.

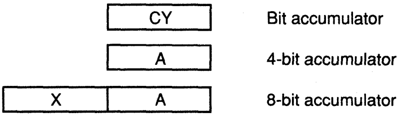


**Figure 4.5–1 Accumulator**

### 4.6 Stack Pointer (SP) – Eight Bits

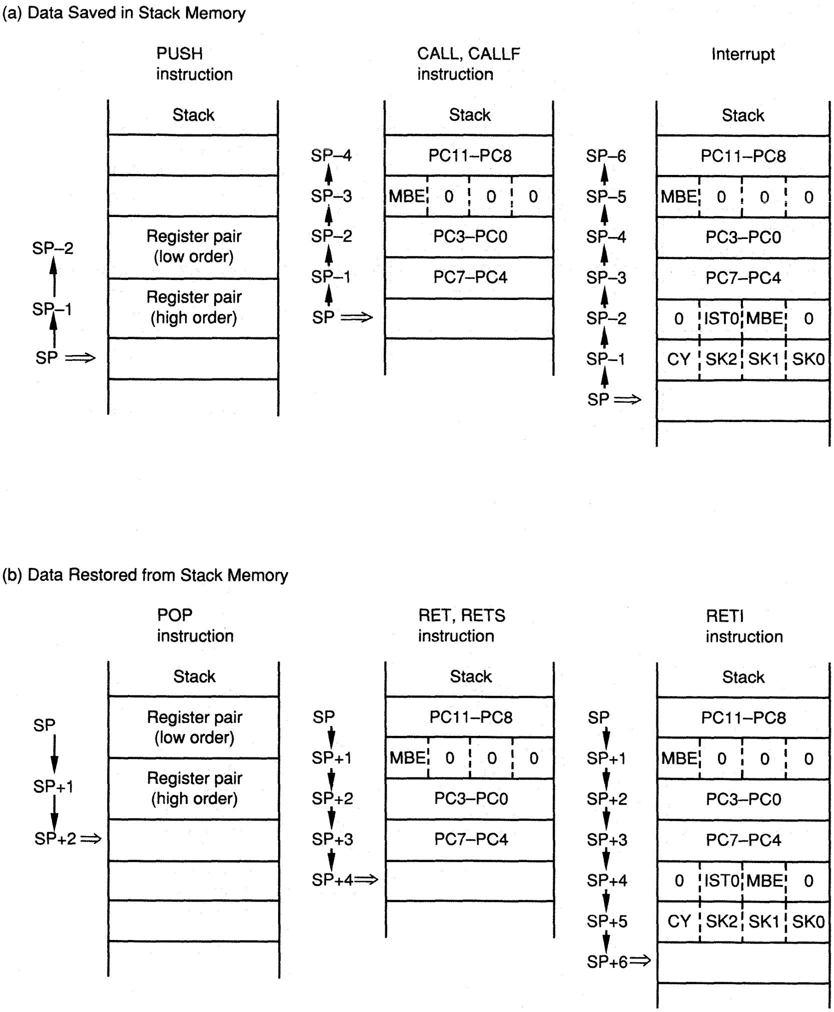
The μPD7500X uses general purpose RAM for stack memory (LIFO). The stack pointer (SP) is an 8-bit register which holds top address information of the stack area.
The stack area addresses are 000H–0FFH of memory bank 0 regardless of how MBE and MBS are set.
SP is decremented before data is saved in the stack memory (write operation); it is incremented after data is restored from the stack memory (read operation).
Figs. 4.6–2 to 4.6–4 shows data saved in and restored from the stack memory when stack operations are performed.
An initial value is set in SP by using an 8-bit memory operation instruction to determine the stack area to be used. The SP contents can also be read.
SP0 is always set to 0.
It is recommended that the initial value of SP should be set to 00H so that the stack area be used is starting at the most significant address of data memory bank 0 (0FFH).
When the RESET signal is generated, the SP contents become undefined. Be sure to initialize SP to the desired value at the start of the program.

Example: To initialize SP.

```
SEL MB15       ;  or CLR1 MBE
MOV XA, #00H
MOV SP, XA     ;  SP← 00H
```

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| F80H | SP7 | SP6 | SP5 | SP4 | SP3 | SP2 | SP1 | SP0 | SP |
|  |  |  |  |  |  |  |  | Fixed to 0 |  |

**Figure 4.6–1 Stack Pointer Configuration**



(a) Data Saved in Stack Memory

(b) Data Restored from Stack Memory

**Figure 4.6–2 Data Saved / Restore of Operation (µPD75004)**

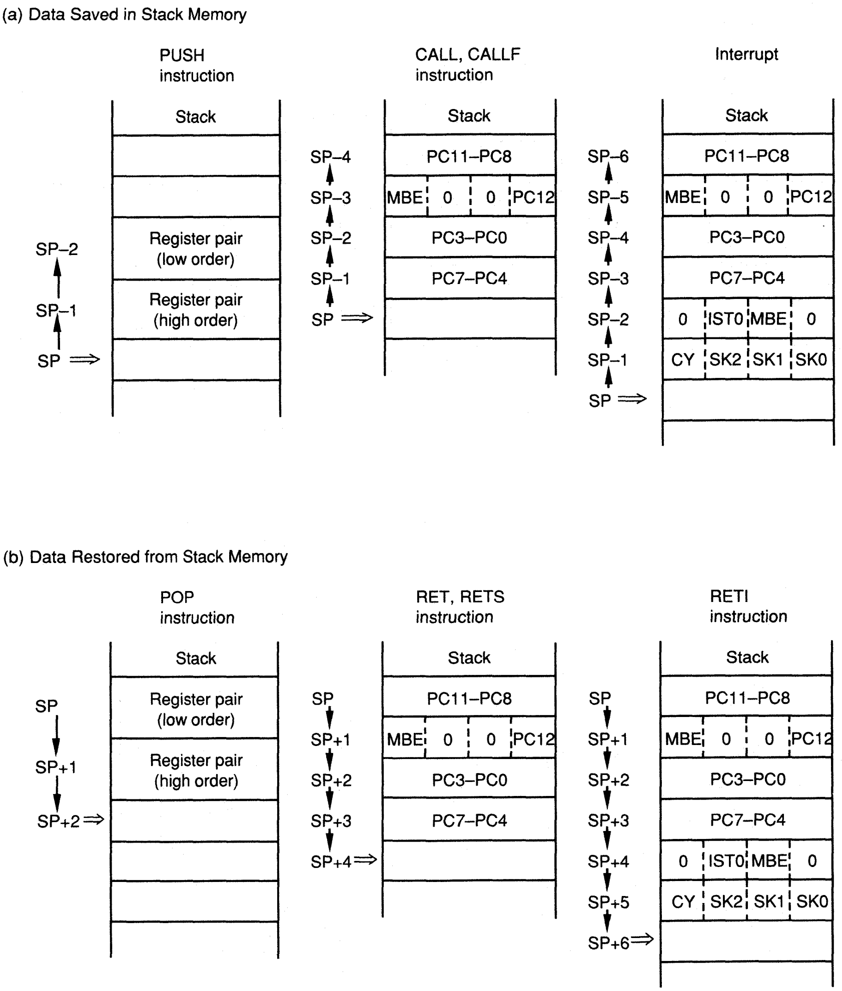


**Figure 4.6–3 Data Saved / Restored in Stack Operation (μPD75006, 75008)**

### 4.7 Program Status Word (PSW) – Eight Bits

The program status word (PSW) consists of flags closely related to processor operation.
PSW is mapped in data memory addresses FB0H and FB1H. Two bits of address FB0H can be operated by using a memory operation instruction.

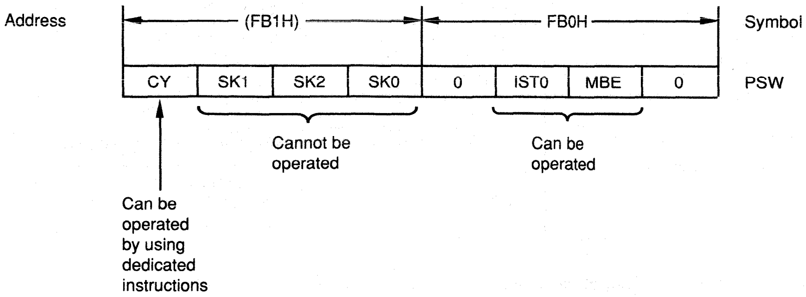


**Figure 4.7–1 Program Status Word Configuration**

**Table 4.7–1 PSW Flags Saved and Restored when Stack Operation is Performed**

| | | Saved or restored flag |
|---|---|---|
| Save | During CALL, CALLF instruction execution | MBE is saved |
| | During hardware interrupt | All PSW bits are saved |
| Restore | During RET, RETS instruction execution | MBE is restored |
| | During RETI instruction execution | All PSW bits are restored |

(1) Carry flag (CY)
The carry is a 1-bit flag that shows the occurence of overflow or underflow during execution of an instruction involving carry (ADDC or SUBC).
The carry flag also serves as a bit accumulator. Boolean algebra operation is performed between the bit accumulator and data memory specified by a bit address. The result can be stored in the bit accumulator.
The carry flag is operated by using dedicated instructions independently of other PSW bits.
When the $\overline{\text{RESET}}$ signal is generated, the carry flag becomes undefined.

**Table 4.7–2 Carry Flag Operation Instructions**

| | Instructions (mnemonic) | Carry flag processing |
|---|---|---|
| Instructions dedicated to carry flag operation | SET1 CY<br>CLR1 CY<br>NOT1 CY<br>SKT CY | CY is set to 1<br>CY is cleared<br>CY is inverted<br>Skip if CY is set to 1 |
| Bit Boolean instructions | AND1 CY, mem*.bit<br>OR1 CY, mem*.bit<br>XOR1 CY, mem*.bit | The specified bit and CY are ANDed, ORed, or XORed together |
| Interrupt service | During interrupt execution | CY is saved in stack memory in parallel with other PSW bits (eight bits) |
| | RETI | CY is restored from stack memory in parallel with other PSW bits |

Remarks: mem*.bit indicates any of the following three bit 3 manipulation addressing modes

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

Example: To AND bit 3 of address 3H and P33 together and set the result in CY.

```
SET1   CY            ;  CY← 1
CLR1   MBE           ;  Or SEL MB0
SKT    3FH. 3        ;  Skip if bit 3 of address 3FH is set to 1.
CLR1   CY            ;  CY← 0
AND1   CY, PORT3.3   ;  CY← CY ∧ P33
```

(2) Skip flags (SK2, SK1, and SK0)
The skip flags store the skip state and are automatically set or reset when the CPU executes instructions. The user cannot directly use the flags as operands.

(3) Interrupt status flag (IST0)
The interrupt status flag stores the current status of processing being performed. (For details, see Table 6.3–2.)

**Table 4.7–3 Interrupt Status Flag Indication Contents**

| IST0 | Status of processing being performed | Processing contents and interrupt control |
|---|---|---|
| 0 | Status 0 | During normal program processing.<br>Every interrupt can be acknowledged. |
| 1 | Status 1 | During interrupt processing.<br>No interrupt must be acknowledged. |

If an interrupt is acknowledged, the IST0 contents are saved in stack memory as a PSW bit, then automatically IST0 is set to 1. When an RETI instruction is executed, IST0 is set to 0.
The interrupt status flag can be operated by using a memory operation instruction. The current status of processing can also be changed under program control.

Caution: Before operating the flag, be sure to execute a DI instruction to disable interrupts. After operating the flag, execute an EI instruction to enable interrupts.

(4) Memory bank enable flag (MBE)

The memory bank enable flag is a 1-bit flag used to specify the address information generation mode of the high-order four bits of a 12-bit data memory address.
MBS can be set or reset at any time by using a bit manipulation instruction regardless of memory bank setting.

```
Example: SET1  MBE  ;  MBE← 1
         CLR1  MBE  ;  MBE← 0
```

When MBE is set to 1, the data memory address space is exceeded and all the data memory space can be addressed.
When MBE is reset to 0, the data memory address space is fixed regardless of how MBS is set. (see Fig. 3.1–2.)
When the $\overline{\text{RESET}}$ signal is generated, the contents of program memory address 0 bit 7 are set and MBE is initialized automatically.
When vectored interrupt service is made, bit 7 of the corresponding vector address table is set and the MBE state during interrupt service is set automatically.
During interrupt service, normally MBE is set to 0 and the general purpose RAM of memory bank 0 is used.

**4.8 Bank Selection Register (BS)**

The memory bank selection register (MBS) for selecting a memory bank is mapped in the bank selection register (BS). The low-order four bits of BS are fixed to 0
MBS is set by using the SEL MBn instructions.
BS can be saved in and restored from the stack area in 8-bit units by using PUSH BS instructions.

| MBS | | | | | | | | Symbol |
|---|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| MBS3 | MBS2 | MBS1 | MBS0 | 0 | 0 | 0 | 0 | BS |

**Figure 4.8–1 Bank Selection Register Configuration**

(1) Memory bank selection register (MBS)

The memory bank selection register (MBS), consisting of four bits, stores high-order 4-bit address information of a 12-bit data memory address. The memory bank to be accessed is specified according to the register contents. However, the μPD7500X allows the user to select bank 0, bank 1, or bank 15 only.
MBS is set by using the SEL MBn instruction (where n is 0, 1, or 15).
The address range applied according to how MBE and MBS are set is as shown in Fig. 3.1–2.
When the $\overline{\text{RESET}}$ signal is generated, MBS is initialized to 0.

## 5 PERIPHERAL HARDWARE FUNCTIONS

### 5.1 Digital Input / Output Ports

The μPD7500X adopts memory mapped I/O. All input / output ports are mapped in the data memory space.

| Address | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|
| 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 | — | — | P81 | P80 | PORT 8 |

**Figure 5.1–1 Data Memory Addresses of Digital Ports**

Table 5.1–2 lists the input / output port operation instructions. In addition to 4-bit input / output, 8-bit input / output and bit manipulation can be performed for PORT 4 to PORT 7 and very diverse control can be performed.

Example: To test the P13 state and output value to ports 4 and 5 depending on the result.

```
SKT   PORT1.3     ;  Skip if port 1 bit 3 is set to 1
MOV   XA, #18H    ;  XA← 18H String effect
MOV   XA, #14H    ;  XA← 14H String effect
SEL   MB15        ;  Or CLR1 MBE
OUT   PORT4, XA   ;  Ports 5,4← XA
```

Example: SET1  PORT 4. @L  ;  The port 4–7 bit specified by using the L register is set to 1.

5.1.1 Types, features, and configurations of digital input / output ports

Table 5.1–1 lists the digital input / output ports.
Figs. 5.1–1 to 5.1–6 shows the port configurations.

**Table 5.1–1 Types and Features of Digital Ports**

| Port (abbreviation) | Function | Operation and features | Remarks |
|---|---|---|---|
| PORT 0 | 4-bit input | Can always be read or tested regardless of the operation mode. | Pins also used for INT4, $\overline{\text{SCK}}$ SO/SB0, SI/SB1. Pins also used for INT0-2 and TI0. |
| PORT 1 | | | |
| PORT 3(Note 1)<br>PORT 6 | 4-bit input / output | Can be placed in input or output mode in 1-bit units. | Pins also used for MD0–MD3, (Note 2)<br>Pins also used for KR0–KR3. |
| PORT 2 | | Can be placed in input or ouput mode in 4-bit units. Ports 6 and 7 can be paired for data input / output in 8-bit units. | Port 2 pins are also used for PTO0, PCL, and BUZ. |
| PORT 7 | | | Pins also used for KR4–KR7. |
| PORT 4(Note 1)<br>PORT 5(Note 1) | 4-bit input / output (N-channel open drain 10 Volts) | Can be placed in input or output mode in 4-bit units. Ports 4 and 5 can be paired for data input / output in 8-bit units. | Internal pull-up resistor can be specified in 1-bit units by using mask option. |

Note 1: An LED can be driven directly.
Note 2: PORT 3 is also used for MD0-MD3 pin only in μPD75P008.

The P10 pin is also used for an external vectored interrupt input pin (input with a noise removal circuit). (For details, see 6.3.) When the $\overline{\text{RESET}}$ signal is generated, port 2-port 8 output latches are cleared, the output buffers are turned off, and input mode is entered.

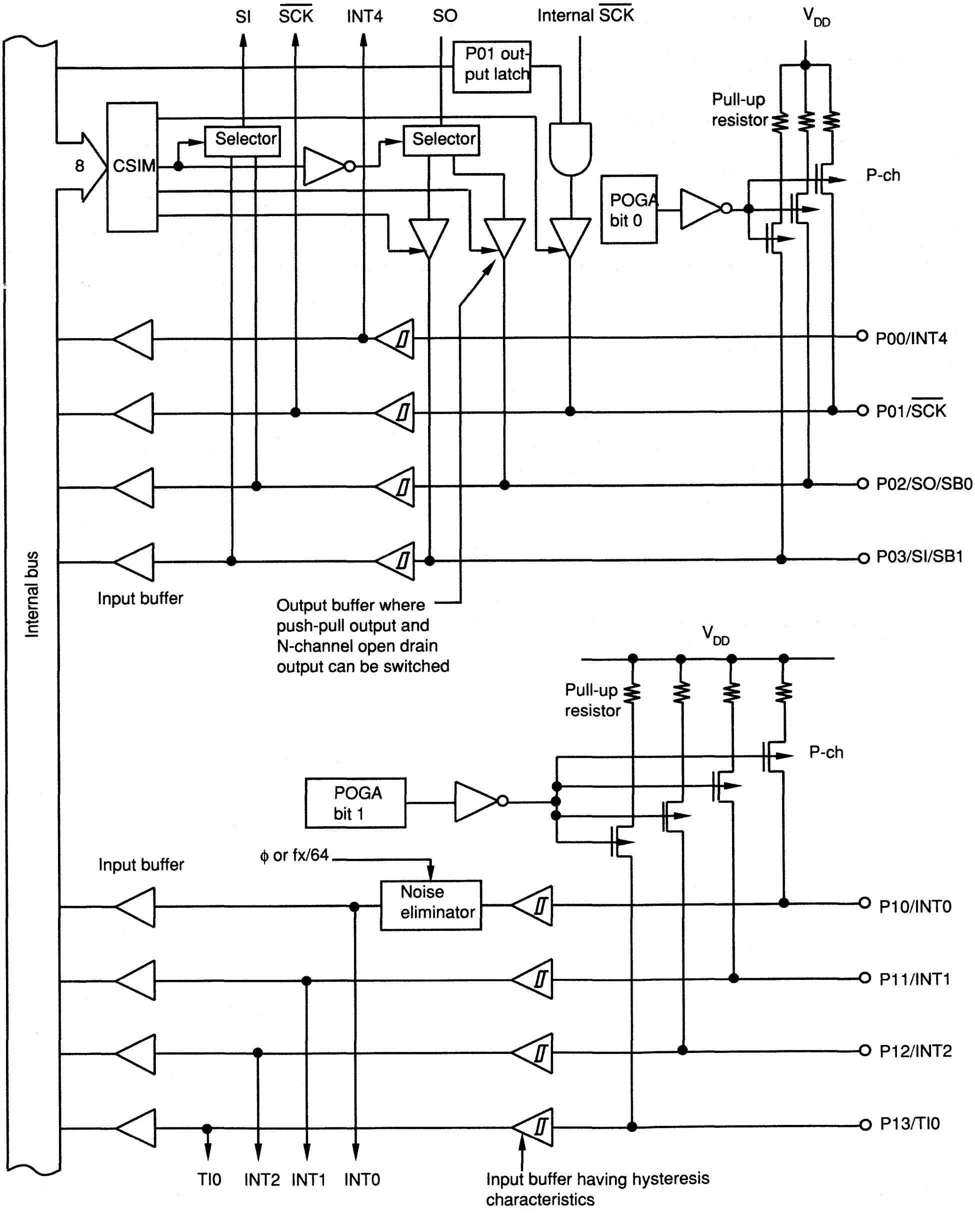


**Figure 5.1–2 Configuration of Ports 0 or 1**

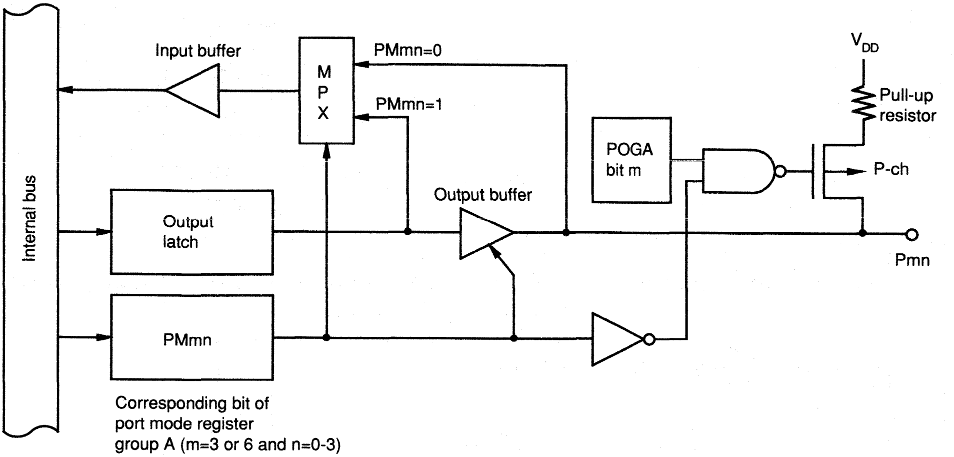


**Figure 5.1–3 Configuration of port 3n or 6n (n = 0–3)**

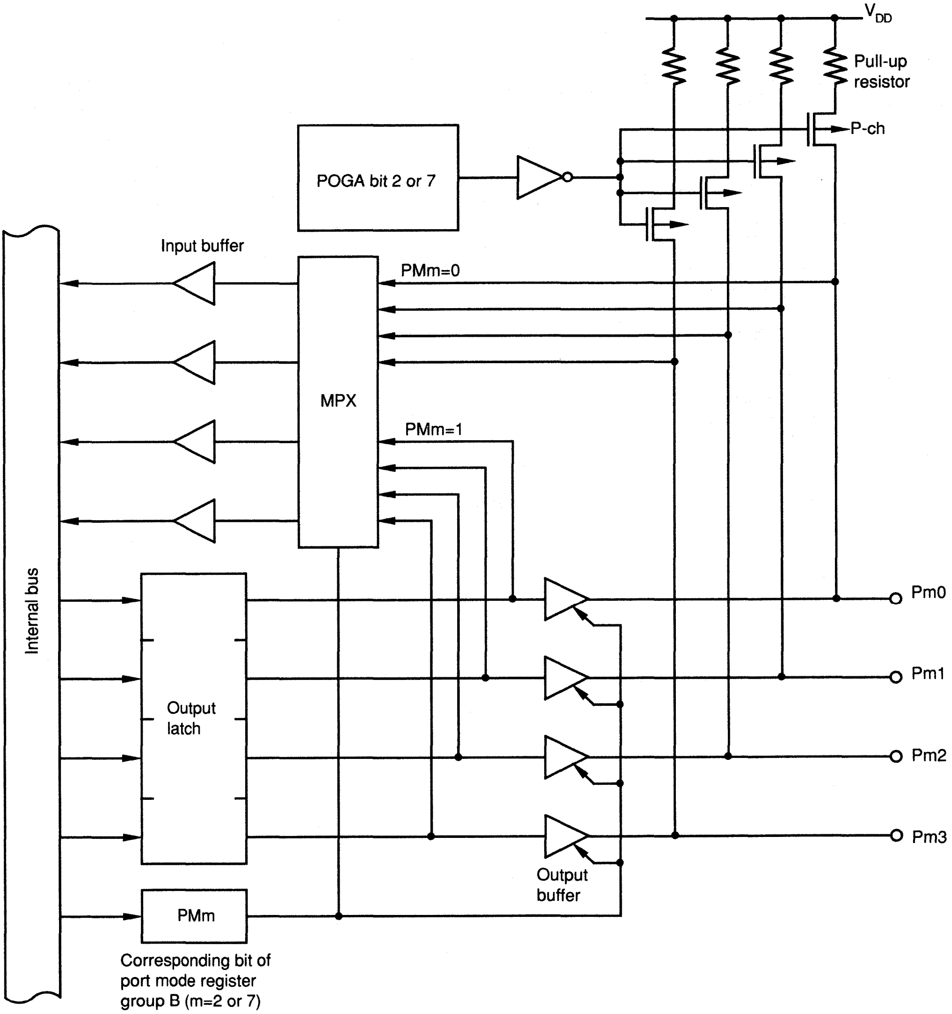


**Figure 5.1–4 Configuration of Port 2 or 7**

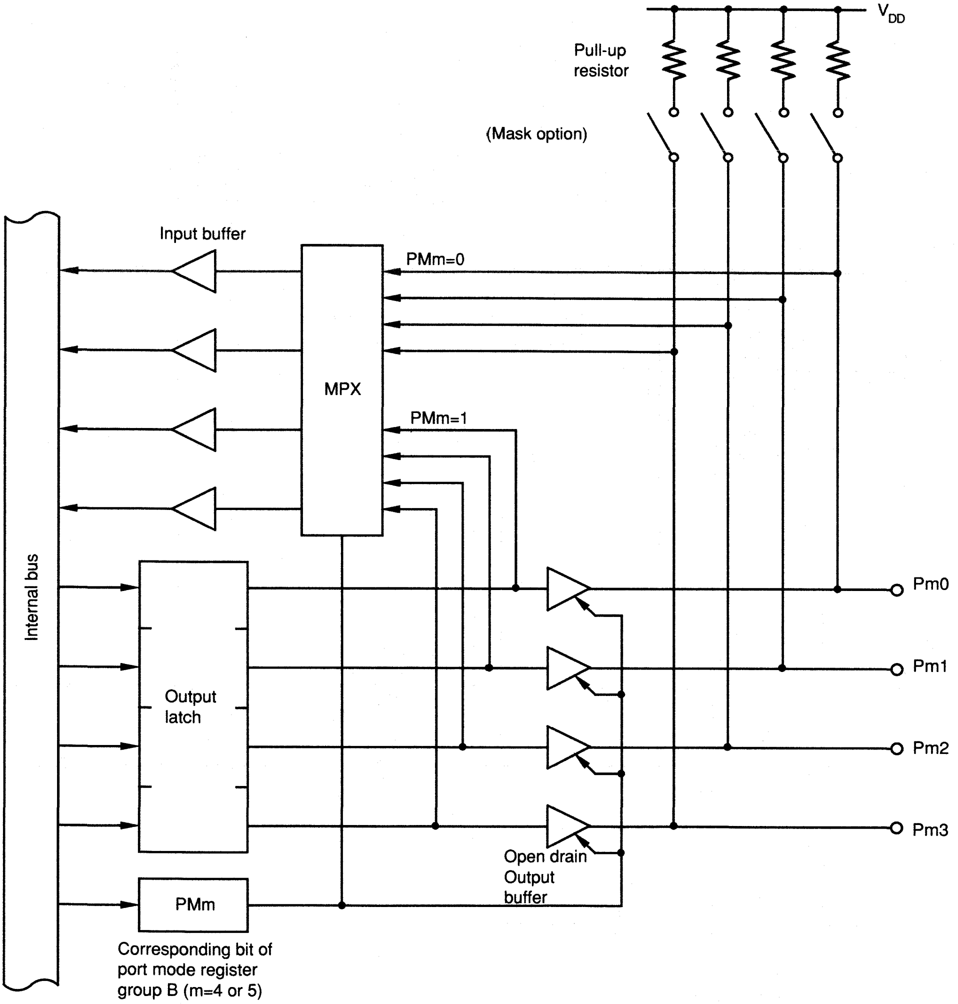


**Figure 5.1–5 Configuration of Port 4 or 5**

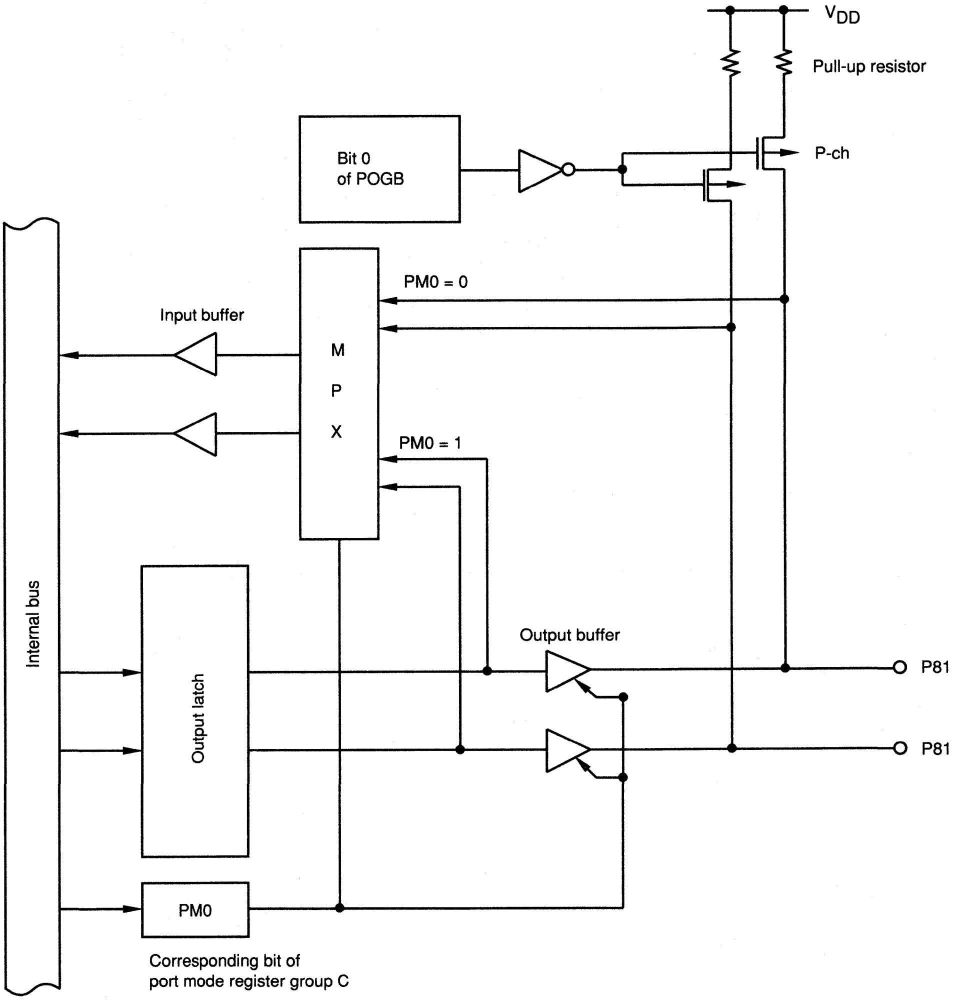


Fig. 5.1-6 Port 8 Organization

5.1.2 Input / output mode setting

The input or output mode of each input/output ports is set by using the port mode register as shown in Fig. 5.1–7. Each bit of ports 3 and 6 can be placed in input or output mode individually by using port mode register group A (PMGA). Each of ports 2, 4, 5, and 7 (four bits each) is placed in input or output mode individually by using port mode register group B (PMGB).

Each port serves as an input port when the corresponding port mode register bit is set to = 0 and serves as an output port when set to 1.

Since the output latch contents are applied to the output pins at the same time the output mode is selected by setting the port mode register, the output latch contents must previously be rewitten to the required value before the output mode is set.

Port mode register groups A and B are set by using 8-bit memory operation instructions

When the RESET signal is generated, all bits of the port mode registers are cleared; thus the output buffers are turned off, and all ports are placed in input mode.

Example: To use P30, P31, P62, and P63 for input pins and P32, P33, P60, and P61 for output pins.

```
CLR1   MBE          ;   Or SEL MB15
MOV    XA, #3CH
MOV    PMGA, XA
```

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 |

| | | |
|---|---|---|
| | PM3n<br>PM6n | P3n, P6n pin input / output specification (n = 0–3) |
| | 0 | Input mode (output buffer off) |
| PMGA | 1 | Output mode (output buffer on) |

Port mode register group B

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

| PMn | Port n input / output specification (n = 2, 4, 5, or 7) |
|---|---|
| 0 | Input mode (output buffer off) |
| 1 | Output mode (output buffer on) |

Port mode register group C

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FEEH | – | – | – | – | – | – | – | PM8 | PMGC |

| PMn | Port 8 input specification |
|---|---|
| 0 | Input mode (output buffer off) |
| 1 | Output mode (output buffer on) |

–: May be 0 or 1.

**Figure 5.1–7 Port Mode Register Formats**

5.1.3 Digital input / output port operation instructions

Since all input / output ports incorporated in the µPD7500X are mapped in the data memory space, all the data memory operation instructions are applicable. Table 5.1–2 lists the data memory operation instructions particulary useful for input / output pin operation and their application ranges.

(1) Bit operation instructions

The specific address bit direct addressing (fmem. bit) and specific address bit register indirect addressing (pmem. @L) modes are applicable to digital input / output ports PORT 0-PORT 7. The port bits can be operated at any time as desired regardless of how MBE and MBS are set.

Example: To OR P50 and P41 together and output the result to P61.

```
       SET1   CY            ;   CY← 1
       AND1   CY, PORT5.0   ;   CY← CY∧P50
       OR1    CY, PORT4.1   ;   CY← CY∨P41
       SKT    CY
       BR     CLRP
       SET1   PORT6.1       ;   P61← 1
        :
        :
CLRP:  CLR1   PORT6.1       ;   P61← 0
```

(2) 4-bit operation instructions

In addition to the IN and OUT instructions, all 4-bit memory operation instructions such as MOV, XCH, ADDS, and INCS can be used. However, memory bank 15 must have been selected before executing an instruction.

Example 1: To output the accumulator contents to port 3.

```
SEL    MB15          ;   Or CLR1 MBE
OUT PORT3, A
```

Example 2: To add the accumulator value to the data output to port 5 and output the result.

```
SET1   MBE
SEL    MB15
MOV    HL, #PORT5
ADDS   A, @HL        ;   A← A+PORT5
NOP
MOV    @HL, A        ;   PORT5← A
```

Example 3: To test whether or not data in port 4 is greater than the accumulator value.

```
SET1   MBE
SEL    MB15
MOV    HL, #PORT4
SUBS   A, @HL        ;   A<PORT4
BR     NO            ;   NO
                     ;   YES
```

(3) 8-bit operation instructions

In addition to the IN and OUT instructions, the MOV, XCH and SKE instructions can be used for ports 4 and 5 where 8-bit operation (manipulation) can be performed. As with the 4-bit operation instructions, memory bank 15 must have been selected before executing the instruction.

Example: To output data in the BC register pair to the output ports specified by the 8-bit data input from ports 4 and 5.

```
SET 1  MBE
SEL    MB15
IN     XA, PORT 4    ;   XA← ports 5 and 4
MOV    HL, XA        ;   HL← XA
MOV    XA, BC        ;   XA← BC
MOV    @HL, XA       ;   Port (L)← XA
```

**Table 5.1–2 Input / Output Port Pin Operation Instruction List**

<table>
<tr><th>PORT Instruction</th><th>PORT 0</th><th>PORT 1</th><th>PORT 2</th><th>PORT 3</th><th>PORT 4</th><th>PORT 5</th><th>PORT 6</th><th>PORT 7</th><th>PORT 8</th></tr>
<tr><td>IN A, PORTn (Note 1)</td><td colspan="9">O</td></tr>
<tr><td>IN XA, PORTn (Note 1)</td><td colspan="2">–</td><td colspan="2">–</td><td colspan="2">O</td><td colspan="2">O</td><td>–</td></tr>
<tr><td>OUT PORTn. A (Note 1)</td><td>–</td><td>–</td><td colspan="7">O</td></tr>
<tr><td>OUT PORTn. XA (Note 1)</td><td colspan="2">–</td><td colspan="2">–</td><td colspan="2">O</td><td colspan="2">O</td><td>–</td></tr>
<tr><td>SET1 PORTn. bit</td><td>–</td><td>–</td><td colspan="7">O</td></tr>
<tr><td>SET1 PORTn. @L (Note 2)</td><td>–</td><td>–</td><td colspan="7">O</td></tr>
<tr><td>CLR1 PORTn. bit</td><td>–</td><td>–</td><td colspan="7">O</td></tr>
<tr><td>CLR1 PORTn. @L (Note 2)</td><td>–</td><td>–</td><td colspan="7">O</td></tr>
<tr><td>SKT PORTn. bit</td><td colspan="9">O</td></tr>
<tr><td>SKT PORTn. @L (Note 2)</td><td colspan="9">O</td></tr>
<tr><td>SKF PORTn. bit</td><td colspan="9">O</td></tr>
<tr><td>SKF PORTn. @L (Note 2)</td><td colspan="9">O</td></tr>
<tr><td>AND1 CY, PORTn. bit</td><td colspan="9">O</td></tr>
<tr><td>AND1 CY, PORTn. @L (Note 2)</td><td colspan="9">O</td></tr>
<tr><td>OR1 CY, PORTn. bit</td><td colspan="9">O</td></tr>
<tr><td>OR1 CY, PORTn. @L (Note 2)</td><td colspan="9">O</td></tr>
<tr><td>XOR1 CY, PORTn. bit</td><td colspan="9">O</td></tr>
<tr><td>XOR1 CY, PORTn. @L (Note 2)</td><td colspan="9">O</td></tr>
</table>

Note 1: MBE = 0 or (MBE = 1 and MBS = 15) must have been set before executing the instruction.
2: The low-order two bits of address and the bit address are specified indirectly by using the L register.

5.1.4 Digital input / output port operation

When data memory operation instructions are executed for the digital input / output ports, port (pin) operation varries according to which mode (input or output) is set. (See table 5.1–3.) This is because as understood from the input / output port structure, data read into the internal bus is data for each pin in the input mode and output latch data in the output mode.

(1) Operation when input mode is set

Pin data is operated when executing a test instruction such as SKT or an instruction to read 4- or 8-bit port data into the internal bus (IN, OUT, operation, or comparison instruction).

When an instruction to transfer the accumulator contents (four or eight bits) to a port or ports is executed (OUT or MOV instruction), the accumulator data is latched in the output latches. The output buffers remain off.

When an XCH instruction is executed, pin data is input to the accumulator and the accumulator data is latched in the output latches. The output buffers remain off.

When an INCS instruction is executed, data resulting from adding 1 to 4-bit pin data is latched in the output latches. The output buffer remain off.

When an instruction to rewrite data memory bitwise is executed, such as SET1, CLR1, or SKTCLR, (the specifed bit output latch can be rewritten as specified by instruction), but the contents of the output latch for other bits become undefined.

(2) Operation when output mode is set
When a test instruction or an instruction to read 4- or 8-bit port data into the internal bus is executed, the output latch contents are operated.
When an instruction to transfer the accumulator contents (four or eight bits) is executed, output latch data is rewritten and at the same time as output from the pins.
When an XCH instruction is executed, the output latch contents are transferred to the accumulator, and the accumulator contents are held in the output latches and is output from the pins.
When an INCS instruction is executed, data resulting from incrementing the output latch contents by one is held in the output latches and output from the pins.
When a bit output instruction is executed, the specified output latch bit is rewitten and output from the pin.

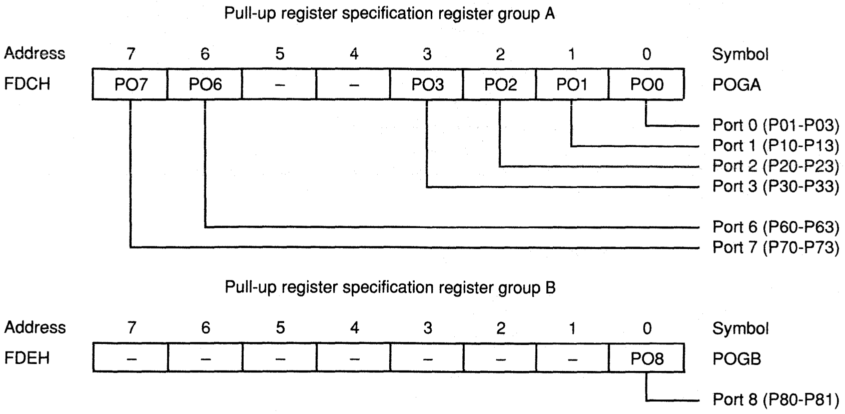
**Table 5.1–3 Operation When Input / Output Ports are Used**

| Executed instruction | Port (pin) operation | |
|---|---|---|
| | Input mode | Output mode |
| SKT ①<br>SKF ① | Pin data is tested. | Output latch is data tested. |
| AND1 CY, ①<br>OR1 CY, ①<br>XOR1 CY, ① | Operation is performed between pin data and CY. | Operation is performed between output latch data and CY. |
| IN A, PORTn<br>IN XA, PORTn<br>MOV A, @HL<br>MOV XA, @HL | Pin data is transferred to accumulator. | Output latch data is transferred to accumulator |
| ADDS A, @HL<br>ADDC A, @HL<br>SUBS A, @HL<br>SUBC A, @HL<br>AND A, @HL<br>OR A, @HL<br>XOR A, @HL | Operation is performed between pin data and accumulator. | Operation is performed between output latch data and accumulator. |
| SKE A, @HL | Pin data and accumulator contents are compared. | Output latch data and accumulator contents are compared. |
| OUT PORTn, A<br>OUT PORTn, XA<br>MOV @HL, A<br>MOV @HL, XA | Accumulator data is transferred to output latches. (Output buffers remain off.) | Accumulator data is transferred to output latches and output from the pins. |
| XCH A, PORTn<br>XCH XA, PORTn<br>XCH A, @HL<br>XCH XA, @HL | Pin data is transferred to accumulator and accumulator data is transferred to output latches.<br>(Output buffers remain off.) | Data is exchanged between output latches and the accumulator. |
| INCS PORTn<br>INCS @HL | Data resulting from incrementing pin data by one is latched in output latches. | The output latch contents are incremented by one. |
| SET1 ①<br>CLR1 ①<br>SKTCLR ① | The specified bit output latches are rewritten as specified by the instruction, but the output latch contents for other bits are undefined. | The output pin state is changed according to the instruction. |

① : Denotes two addressing modes PORTn. bit and PORTn. @L.

5.1.5 Internal pull-up resistors

Pull-up resistors can be incorporated in the μPD7500X port pins except for P00. Internal pull-up resistors are specified by using software or mask option.



| | Specification |
|---|---|
| 0 | Pull-up resistor is not incorporated. |
| 1 | Pull-up resistor is incorporated. |

**Figure 5.1–8 Pull-Up Register Specification Register Format**

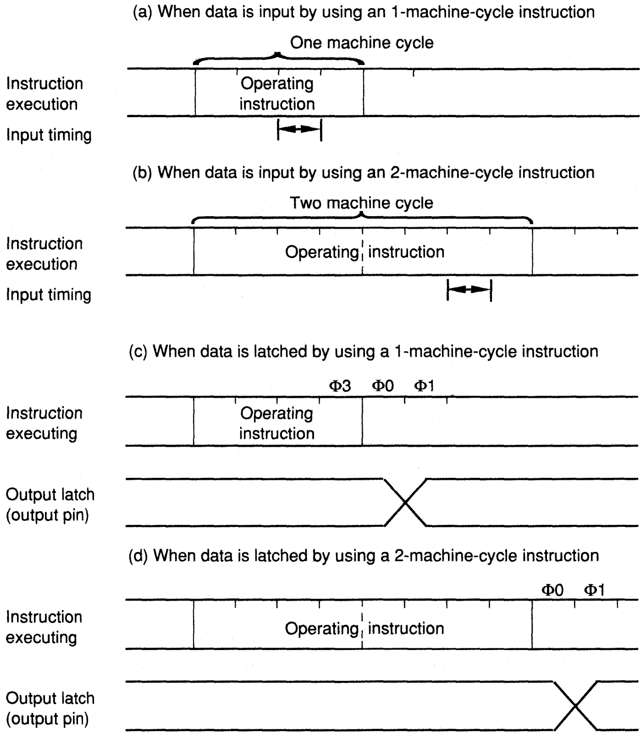
**Table 5.1–4**

| Port (Pin name) | Specification method to provide pull-up resistor | POGA bit | POGB bit |
|---|---|---|---|
| Port 0 (P01-P03) (Note 1) | By means of software in 3-bit units. | Bit 0 | |
| Port 1 (P10-P13) | By means of software in 4-bit units. | 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) | By mask option in 1-bit units. | – | – |
| Port 5 (P50-P53) | | | |
| Port 8 (P80-P81) | By means of software in 2 bit units. | – | Bit 0 |

Note: A pull-up resistor cannot be provided for the P00 Pin.

5.1.6 Digital input / output port input / output timing

Fig. 5.1–9 shows timing of data to output latch and timing of pin data or output latch data input to the internal bus.
Fig. 5.1–10 shows the ON timing at specified the internal pull-up resistor by software.



(a) When data is input by using an 1-machine-cycle instruction

(b) When data is input by using an 2-machine-cycle instruction

(c) When data is latched by using a 1-machine-cycle instruction

(d) When data is latched by using a 2-machine-cycle instruction

**Figure 5.1–9 Digital Input / Output Port Input / Output Timing**

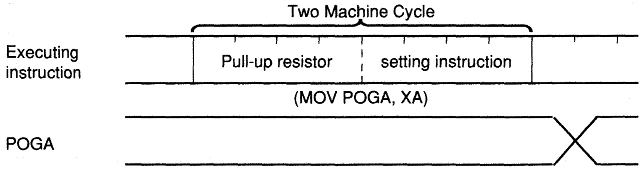


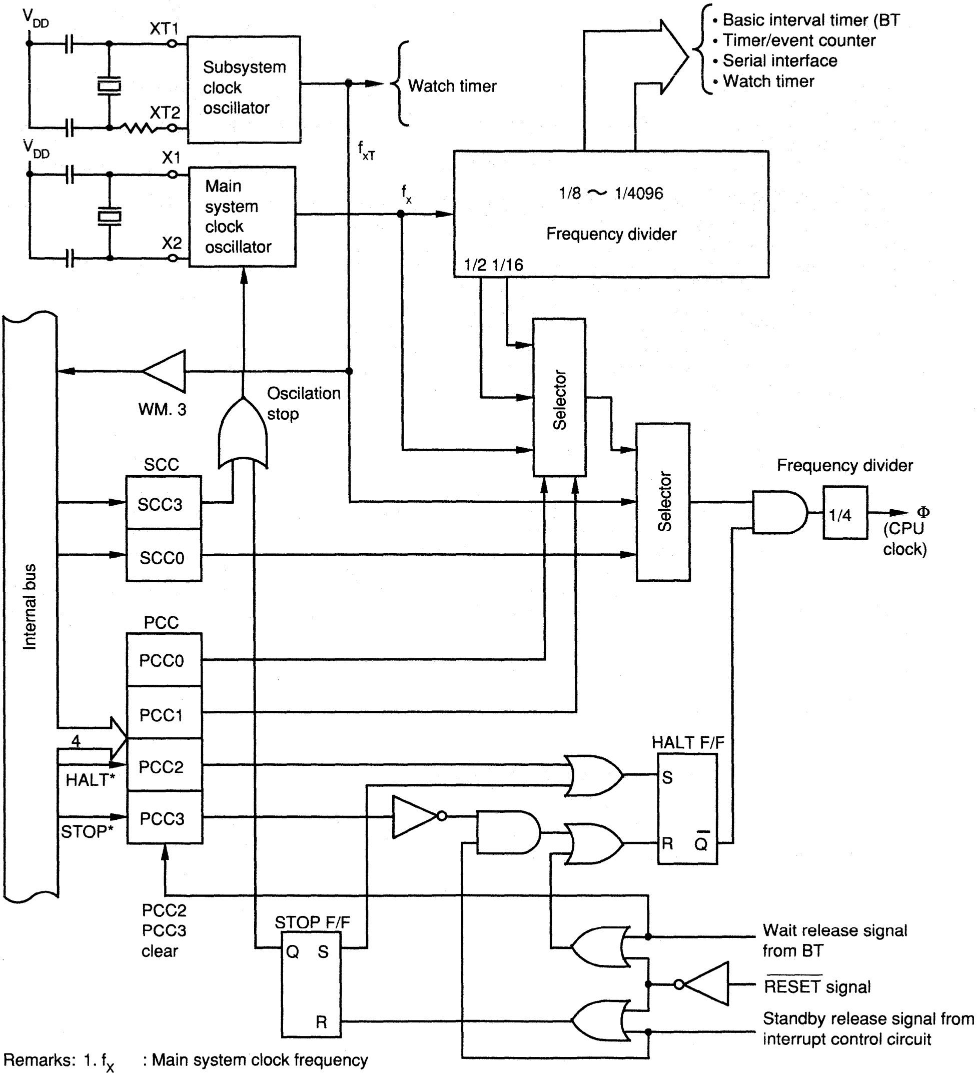
**Figure 5.1–10 ON Timing of Pull-up Resistor by using Software**

### 5.2 Clock Generator

The clock generator supplies clocks to CPU and peripheral hardware for controlling the CPU operating mode.

5.2.1 Clock generator configuration

Fig. 5.2–1 shows a clock generator block diagram.



Remarks: 1. $f_X$ : Main system clock frequency
2. $f_{XT}$ : Subsystem clock frequency
3. PCC : Processor clock control register
4. SCC : System clock control register
5. * denotes instruction execution.

**Figure 5.2–1 Clock Generator Block Diagram**

5.2.2 Clock generator function and operation

The clock generator produces the following clocks and controls CPU operating modes such as standby:
— Main system clock $f_X$
— Subsystem clock $f_{XT}$
— CPU clock $\phi$
— Clock to peripheral hardware

The clock generator operates according to how the processor clock control register (PCC) and system clock control register (SCC) are set, as described below:

(a) When the $\overline{RESET}$ signal is generated, the minimum speed mode of the main system clock (15.3 μ/4.19 MHz) is selected. (PCC = 0 and SCC = 0)

(b) When the main system clock is selected, one of three CPU clock frequencies can be selected (0.95 μs, 1.91 μs, and 15.3 μs/ 4.19 MHz) by setting PCC.

(c) When the main system clock is selected, the standby mode (STOP or HALT) can be used.

(d) Subsystem clock is selected by setting SCC.0, and operation can be performed in a very low-speed and low current consumption (122 μs/32.768 kHz). In this case, the PCC setup value does not affect the CPU clock.

(e) When the subsystem clock is selected, main system clock oscillation can be stopped by setting SCC.3. The HALT mode can also be used, but the STOP mode cannot be used. (Subsystem clock oscillation cannot be stopped.

(f) The main system clock is divided to generate a clock supplied to peripheral hardware. The subsystem clock can be supplied directly only to the watch timer. Thus, the watch function and the LCD controlller and buzzer output function which operate using the watch timer clock can also continue operation in the standby mode.

(g) When subsystem clock is selected, the watch timer and LCD controller can continue normal operation. When the main system clock is stopped, other hardware devices than the watch timer cannot be used because they operate according to the main system clock.

(1) Processor clock control register (PCC)

PCC consists of four bits; the low-order two bits are used to select CPU clock $\phi$, and the high-order two bits are used to control CPU operation mode. (See Fig. 5.2-2.)

When bit 3 or 2 is set to (1), the standby mode is set. When the standby mode is released by the standby mode release signal, automatically bits 3 and 2 are cleared and the normal operating mode is entered. (For details, see Paragraph 7).

The low-order two bits of PCC are set by using a 4-bit memory operation instruction (the high-order two bits are set 0).

Bits 3 and 2 are set to (1) by using STOP and HALT instructions respectively.

The STOP and HALT instructions can always be executed independently of the MBE contents.

CPU clock can be selected only during main system clock operation. When the μPD7500X is operated on the subsystem clock, the low-order two bits of PCC becomes invalid, and the CPU clock is fixed to $f_{XT}/4$. The stop instruction is also enabled only during main system operation.

Example 1: To set a machine cycle to 0.95 μs (4.19) MHz).

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

Example 2: To set STOP mode. (Be sure to enter a NOP instruction following the STOP or HALT instruction.)

```
STOP
NOP
```

When the $\overline{RESET}$ signal is generated, PCC is cleared.

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

CPU clock selection bits

| PCC1 | PCC0 | SCC = 0<br>The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz |  | SCC = 1<br>The value enclosed in parentheses is applied when $f_{XT}$ = 32.768 kHz |  |
|---|---|---|---|---|---|
|  |  | CPU clock frequency | 1 machine cycle | CPU clock frequency | 1 machine cycle |
| 0 | 0 | $\Phi = f_X/64$<br>(65.5 kHz) | 15.3 μs | $\Phi = f_{XT}/4$<br>(8.192 kHz) | 122 μs |
| 0 | 1 | Undefined | – |  |  |
| 1 | 0 | $\Phi = f_X/8$<br>(524 kHz) | 1.91 μs |  |  |
| 1 | 1 | $\Phi = f_X/4$<br>(1.05 MHz) | 0.95 μs |  |  |

$f_X$ : Main system clock oscillator output frequency

$f_{XT}$: Subsystem clock oscillator output frequency

CPU operation mode control bits

| PCC3 | PCC2 |  |
|---|---|---|
| 0 | 0 | Normal operating mode |
| 0 | 1 | HALT mode |
| 1 | 0 | STOP mode |
| 1 | 1 | Undefined |

**Figure 5.2–2 Processor Clock Control Register Format**

(2) System clock control register (SCC)

SCC consists of four bits; the least significant bit is used to select CPU clock, and the most significant bit is used to control (stop) main system clock oscillation. (See Fig. 5.2–3.)

SCC 0 and SCC 3 exist at the same data memory address; the bits cannot be changed at the same time. Thus, SCC 0 and SCC 3 are set by using a bit operation instruction. SCC 0 and SCC 3 can always be operated independetly of the MBE contents.

Main system clock oscillation can be stopped by setting SCC 3 only during subsystem clock operation. Main system clock oscillation is stopped by using the STOP instruction during main system clock operation.

When the $\overline{\text{RESET}}$ signal is generated, SCC is cleared.

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| FB7H | SCC3 | – | – | SCC0 | SCC |

| SCC3 | SCC0 | CPU clock slection | Main system clock oscillation |
|---|---|---|---|
| 0 | 0 | Main system clock | Oscillation is enabled. |
| 0 | 1 | Subsystem clock | |
| 1 | 0 | Undefined | |
| 1 | 1 | Subsystem clock | Oscillation is stopped. |

**Figure 5.2–3 System Clock Control Register Format**

Caution 1: Changing the system clock requires a maximum of $1/f_{XT}$ time. To stop main system clock after changing the subsystem clock, set SCC 3 after the machine cycle or cycles listed in Table 5.2–1.

2: Even if oscillation is stopped by setting SCC 3 during main system clock operation, normal STOP mode is not entered.

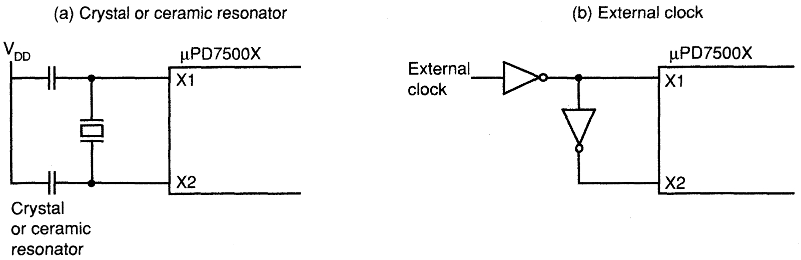
3: When SCC.3 is set to 1, X1 input is connected internally to $V_{SS}$, to avoid leakage current due to crystall oscillator. When the external clock is used in main system clock, do not set SCC.3 to 1.

4: When the system clock is switched to the subsystem clock it is necessary to disable any interrupt during the time given in table 5.2-1.

(3) System clock oscillators

The main system clock oscillator uses a crystal oscillator (4.194304 MHz standard) or cermanic oscillator connected to the X1 and X2 pins.

An external clock can also be input.


(a) Crystal or ceramic resonator

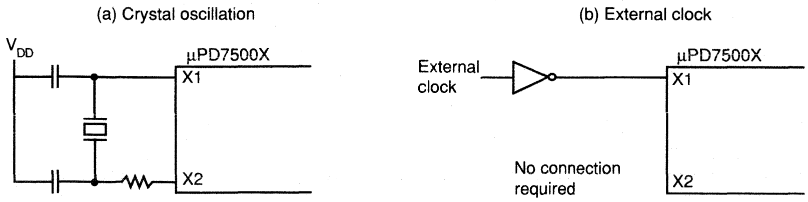
(b) External clock

**Figure 5.2–4 External Circuit to Main System Clock Oscillator**

Caution: When an external clock is input, the STOP mode cannot be set because the X1 pin is connected to VSS in the STOP mode.

- The subsystem clock oscillator uses a crystal oscillator (32.768 kHz standard) connected to the XT1 and XT2 pins.

An external clock can also be input.
The XT1 pin state can be tested by using watch mode register (WM) bit 3.



(a) Crystal oscillation

(b) External clock

**Figure 5.2–5 External Circuit to Subsystem Clock Oscillator**

(4) Frequency divider
The frequency divider divides the main system clock oscillator output ($f_X$) and generates various clocks.

5.2.3 System clock and CPU clock setting

(1) Time required to change system and CPU clocks
The system and CPU clocks can be changed by using the low-order two bits of PCC and the least significant bit of SCC. However, this clock change is not immediatly made after the register are rewritten, and the clock before the clock change is made is used for operation during given machine cycles. Thus, to stop main system clock oscillation, a STOP instruction must be executed or SCC 3 must be set after the change time elapses.

**Table 5.2–1 Maximum Time Required to Change System and CPU Clocks**

| Setup value before change | | | Setup value after change | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCC | PCC | PCC | SCC0 | PCC1 | PCC0 | SCC0 | PCC1 | PCC0 | SCC0 | PCC1 | PCC0 | SCC0 | PCC1 | PCC0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | X | X |
| 0 | 0 | 0 | | | | 1 machine cycle | | | 1 machine cycle | | | $\frac{f_X}{64f_{XT}}$ machine cycle<br>(2 machine cycles) | | |
| | 1 | 0 | 8 machine cycles | | | | | | 8 machine cycles | | | $\frac{f_X}{8f_{XT}}$ machine cycles<br>(16 machine cycles) | | |
| | 1 | 1 | 16 machine cycles | | | 16 machine cycles | | | | | | $\frac{f_X}{4f_{XT}}$ machine cycles<br>(32 machine cycles) | | |
| 1 | X | X | $\frac{f_X}{64f_{XT}}$ machine cycles<br>(2 machine cycles) | | | $\frac{f_X}{8f_{XT}}$ machine cycles<br>(16 machine cycles) | | | $\frac{f_X}{4f_{XT}}$ machine cycles<br>(32 machine cycles) | | | | | |

The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz and $f_{XT}$ = 32.768 kHz.

Remarks: The CPU clock Φ is supplied to the μPD7500X internal CPU. The reciprocal of the clock becomes the minimum instruction time. (The present manual defines it to be one machine cycle.)

(2) System and CPU clock change sequence
System and CPU clock change is explained using Fig. 5.2–6.

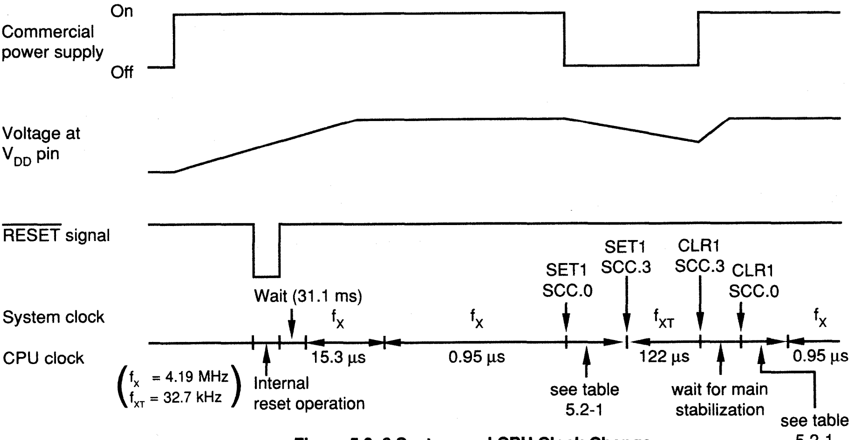


**Figure 5.2–6 System and CPU Clock Change**

1) When the RESET signal is generated, the CPU starts operating at the minimum speed of the main system clock (15.3 μs / 4.19 MHz) after the wait time (31.3 ms / 4.19 MHz) to allow time for stabilizing the oscillator.

2) After enough time has elapsed for the $V_{DD}$ pin voltage to rise to an adequate level PCC is rewritten and the μPD7500X operates at the maximum speed.

3) Turning off the commercial power supply is detected by using interrupt input (INT4 is useful); SCC. 0 is set, and the μPD7500X operates using the subsystem clock. (At the time, a check must have been made to ensure that subsystem clock oscillation has started. After the time required to change to the subsystem clock (32 machine cycles) has elapsed, SCC 3 is set and main system clock oscillation is stopped.

4) Restoration of the commercial power supply is detected by using an interrupt. SCC. 3 is cleared and main system clock oscillation is started. After the time required to stabilize oscillation has elapsed, SCC. 0 is cleared and the μPD7500X operates at the maximum speed.

5.2.4 Clock output circuit

(1) Clock output circuit configuration
Fig. 5.2-7 shows a clock output circuit block diagram.

(2) Clock output circuit function
The clock output circuit, which outputs clock pulses from the P22/PCL pin, is used to supply clock pulses to remote control output or peripheral LSIs.
Clock pulses are output in the following sequence:
(a) Clock output frequency is selected. Clock output is disabled.
(b) 0 is written into P22 output latch.
(c) Port 2 input/output mode is placed in output mode.
(d) Clock output is enabled.

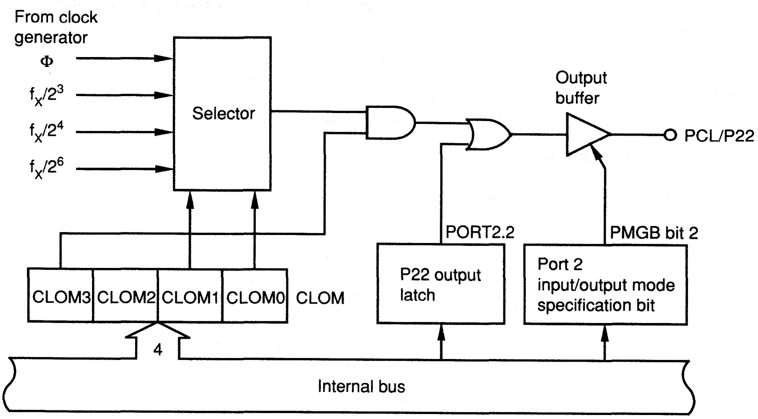


**Figure 5.2–7 Clock Output Circuit Block Diagram**

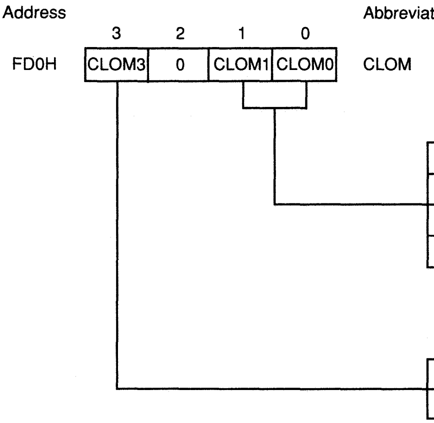
Remarks: The circuit is designed so as not to output a spurious short-width pulse when changing between clock output enable and disable.

(3) Clock output mode register (CLOM)
CLOM is a 4-bit register to control clock output.
CLOM is set by using a 4-bit memory operation instruction. It cannot be read.

Example: To output clock Φ from PCL/P22 pin.

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

When the $\overline{\text{RESET}}$ signal is generated, CLOM is cleared and clock output is disabled.



Clock output frequency selection bit

| | | |
|---|---|---|
| 0 | 0 | Φ output (Note) (1.05 MHz, 524 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: Φ is the CPU clock selected by setting PCC.

Clock output enable/disable bit

| | |
|---|---|
| 0 | Output disabled |
| 1 | Output enabled |

Caution: Be sure to write 0 into CLOM bit 2.

**Figure 5.2–8 Clock Output Mode Register Format**

(4) Application example to remote control output

The µPD7500X clock output function is applicable to remote control output. Remote control output carrier frequency is selected by using the clock frequency selection bit of the clock output mode register. Pulse output is enabled or disabled under software control by using the clock output enable/disable bit.

It is designed so that a spurious short-width pulse will not be produced when changing between clock output enable and disable.

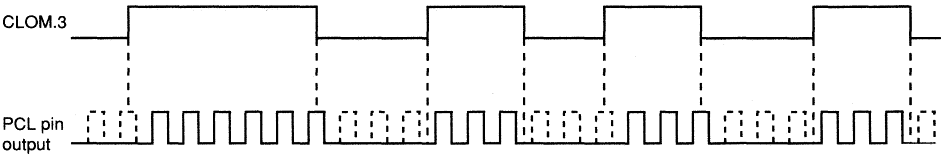


**Figure 5.2–9 Remote Control Output Application Example**

### 5.3 Basic Interval Timer

The µPD7500X contains an 8-bit basic interval timer which has the following functions:

(a) Reference time generation (four types of time interval)

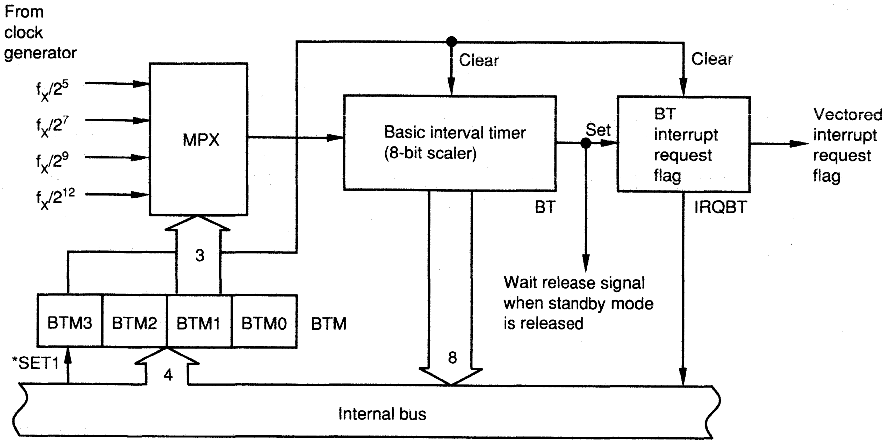
(b) Wait time selection and count when standby mode is released.

(c) Count read

The basic interval timer can also be applied to a watchdog timer to detect program overrun.

5.3.1 Basic interval timer configuration

Fig. 5.3–1 shows the configuration of the basic interval timer


From clock generator
$f_X/2^5$
$f_X/2^7$
$f_X/2^9$
$f_X/2^{12}$
MPX
Clear
Clear
Basic interval timer (8-bit scaler)
Set
BT interrupt request flag
Vectored interrupt request flag
BT
IRQBT
3
Wait release signal when standby mode is released
BTM3
BTM2
BTM1
BTM0
BTM
*SET1
4
8
Internal bus

Remarks: * denotes instruction executing.

**Figure 5.3–1 Basic Interval Timer Configuration**

5.3.2. Basic interval timer mode register (BTM)

BTM is a 4-bit register for controlling operation of the basic interval timer.
BTM is set by using a 4-bit memory operation instruction.
Bit 3 can be set individually by using a bit operation instruction.

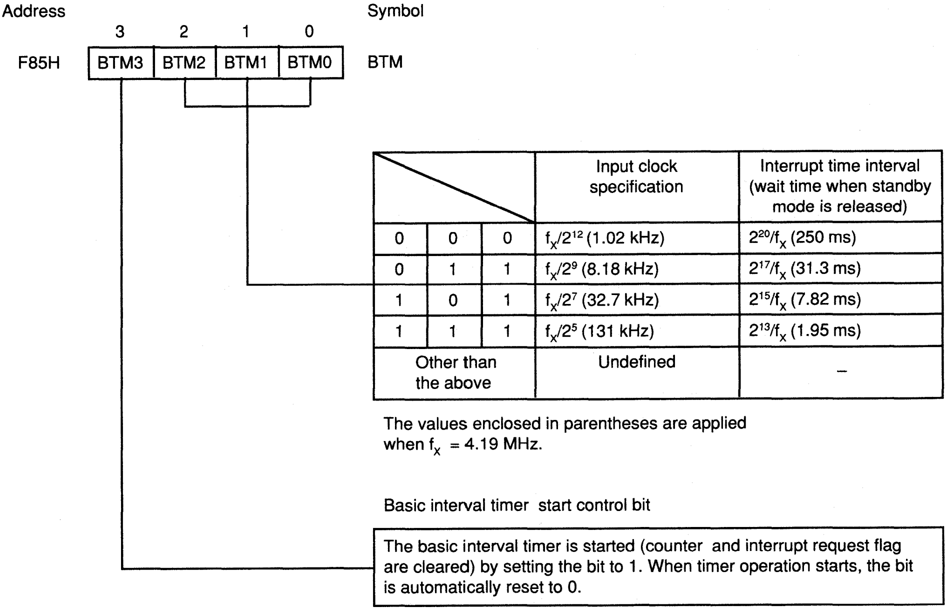
Example 1: To set the interrupt generation interval to 1.95 ms (4.19 MHz).

```
SEL     MB15       ;  or CLR1 MBE
MOV     A, # 1111B
MOV     BTM,A      ;  BTM◄— 1111B
```

Example 2: To clear BT and IRQBT (watchdog timer application).

```
SEL     MB15       ;  or CLR1 MBE
SET1    BTM.3      ;  BTM BIT 3 is set  to 1.
```

When bit 3 is set to 1, the basic interval timer contents are cleared. At the same time, the basic interval interrupt request flag (IRQBT) is also cleared. (The basic interval timer starts.)
When the $\overline{\text{RESET}}$ signal is generated, the BTM contents are cleared and interrupt request signal generation is set for the longest interval.



| | | | Input clock specification | Interrupt time interval (wait time when standby mode is released) |
|---|---|---|---|---|
| 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 | $f_X/2^{7}$ (32.7 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 than the above | | | Undefined | – |

The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz.

Basic interval timer start control bit

The basic interval timer is started (counter and interrupt request flag are cleared) by setting the bit to 1. When timer operation starts, the bit is automatically reset to 0.

**Fig. 5.3-2 Basic Interval Timer Mode Register Format**

5.3.3. Basic interval timer operation

The basic interval timer (BT) is incremented each time a pulse is received from the clock generator. When an overflow occurs, the interrupt request flag (IRQQBT) is set. BT count operation cannot be stopped.
An interrupt generation time interval can be selected from among four types by setting BTM. (See Fig. 5.3—2.)
The basic interval timer and interrupt request flag can be cleared by setting BTM bit 3 to 1 (the interval timer function starts).
The basic interval timer (BT) count can be read by using an 8-bit operation instruction. Data cannot be written into BT.

Caution:
To prevent reading of unstable data during count update when reading the basic interval timer count, execute the read instruction twice and compare the results. If they are valid values, the later read value is used as the read result, if they differ from each other completely, reexecute from the beginning.

Example: To read the BT count

```
        SET1  MBE
        SEL   MB15
        MOV   HL, #BT    ;  BT address is set in HL.
LOOP:   MOV   XA, @HL    ;  First read
        MOV   BC, XA
        MOV   XA, @ HL   ;  Second read
        SKE   A, C
        BR    LOOP
        MOV   A, X
        SKE   A, B
        BR    LOOP
```

The wait function is provided to stop CPU operation until the basic interval timer overflows in order to allow time for system clock oscillation to become stable when the STOP mode is released.
Although the wait time after the $\overline{\text{RESET}}$ signal is generated is fixed, it can be selected by setting BTM when the STOP mode is released by an interrupt. The wait time is selected by setting BTM as shown in Fig. 5.3—2. BTM setting must be performed before the STOP mode is set. (For details, see CHAPTER 7.)

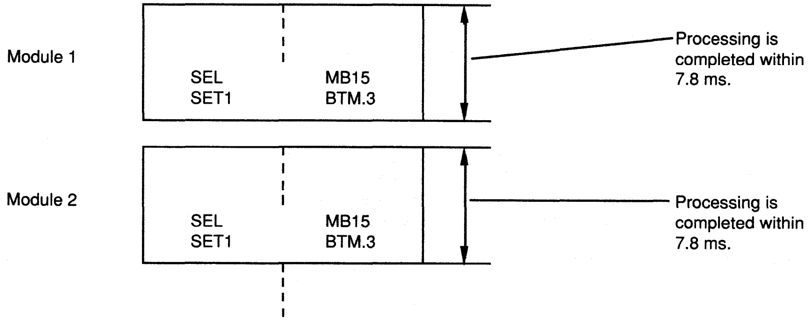
### 5.3.4 Basic interval timer application examples

Example 1: To enable basic interval timer interrupt and set interrupt generation interval at 1.95 ms (at 4.19 MHz).

```
SEL     MB 15
MOV     A, #1111B
MOV     BTM, A      ;   BTM  setting and timer function start
EI                  ;   Interrupt is enabled.
EI      IEBT        ;   BT interrupt is enabled.
```

Example 2: Watchdog timer applciation
A program is divided into serval modules which terminate processing within the BT setup time. BT and IRQBT are cleared at the end of each module. If an interrupt is generated, overrun is asumed to have occured.

```
          SEL     MB15
Initial-  MOV     A, #1101B   ;   7.8 ms interval is set.
izatiion  MOV     BTM, A      ;   BTM setting and timer function start
          EI
          EEI     IEBT
```


Module 1
Processing is completed within 7.8 ms.
Module 2
Processing is completed within 7.8 ms.

```
SEL     MB15
SET1    BTM.3
```

```
SEL     MB15
SET1    BTM.3
```

Example 3: To set the wait time, when the STOP mode is released, using an interrupt to 7.8 ms.

```
SEL     MB15        ;   or CLR1 MBE
MOV     A, #1101B
MOV     BTM, A      ;   BTM ← 1101B
STOP                ;   STOP mode is set
NOP
```

Example 4:
To set the high-level width of a pulse input to INT4 interupt (both rising and falling edge detection). (The pulse width must not exceed the BT setup value. The BT setup value must be 7.8 ms or more.)
< INT4 interrupt routine (MBE = 0) >

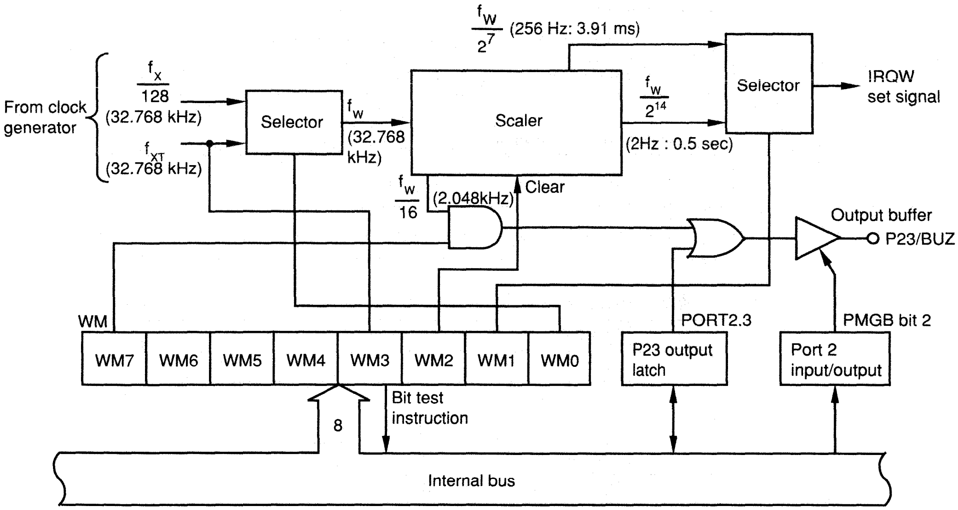
```
LOOP:   MOV     XA, BT       ;   First read
        MOV     BC, XA       ;   Data is stored.
        MOV     XA, BT       ;   Second read
        SKE     A, C
        BR      LOOP
        MOV     A, X
        SKE     A, B
        BR      LOOP
        SKT     PORT0.0      ;   P00=1?
        BR      AA           ;   NO
        MOV     XA, BC       ;   Data is stored in data memory.
        MOV     BUFF, XA
        CLR1    FLAG
        RETI
  AA:   MOV     HL, #BUFF
        MOV     A, C
        SUBC    A, @HL
        INCS    L
        MOV     C, A
        MOV     A, X
        SUBC    A, @HL
        MOV     B, A
        MOV     XA, BC
        MOV     BUFF, XA     ;   Data is stored.
        SET1    FLAG         ;   Data existence flag is set.
        RETI
```

### 5.4 Watch Timer

The µPD7500X incorporates a watch timer (one channel) which has the following function:
(a) The test flag (IRQW) is set at 0.5 s time intervales. The standby mode can be released using IRQW.
(b) The main system and subsystem clocks can be assigned 0.5-s intervals.
(c) In the rapid feed mode, time intervals multiplied by 128 (3.91 ms) are enabled. The mode is useful for program debugging and testing.
(d) A fixed frequency (2.048 kHz) can be output to P23/BUZ. It can be used for sounding the buzzer and trimming the system clock oscillation frequency.
(e) Since the scaler can be cleared, the watch can be started at zero seconds.

5.4.1 Watch timer configuration

Fig. 5.4–1 shows a watch timer block diagram.



The values enclosed in parentheses are applied when $f_X$ = 4.194304 MHz and $f_{XT}$ = 32.768 kHz.

**Figure 5.4–1 Watch Timer Block Diagram**

5.4.2 Watch mode register

The watch dog timer (WM) consists of eight bits to control the watch timer. Fig. 5.4–2 shows the watch mode register format. The watch mode register (except bit 3) is set using an 8-bit operation instruction. Bit 3 is used to test the XT1 pin input level. The input level to the XT1 pin can be tested by making a bit test. When the RESET signal is generated, all bits except bit 3 are cleared.

Example: To produce time using the main system clock (4.19 MHz). To enable buzzer output.

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

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| F98H | WM7 | 0 | 0 | 0 | WM3 | WM2 | WM1 | WM0 | WM |

Count clock ($f_W$) selection bit

| | | |
|---|---|---|
| WM0 | 0 | System clock dividing output $\frac{f_X}{128}$ is selected. |
| | 1 | Subsystem clock $f_{XT}$ is selected. |

Operation mode selection bit

| | | |
|---|---|---|
| WM1 | 0 | Normal watch mode (IRQW is set by using $\frac{f_W}{2^{14}}$, 0.5 s) |
| | 1 | Rapid feed watch mode (IRQW is set by using $\frac{f_W}{2^7}$, 3.91 ms) |

Watch operation enable/disable bit

| | | |
|---|---|---|
| WM2 | 0 | Watch operation is stopped (divider is cleared). |
| | 1 | Watch operation is enabled. |

Input level to XT1 pin (bit test only is enabled)

| | | |
|---|---|---|
| WM3 | 0 | Input level to XT1 pin is low. |
| | 1 | Input level to XT1 pin is high. |

BUZ output enable/disable bit

| | | |
|---|---|---|
| WM7 | 0 | BUZ output is disabled. |
| | 1 | BUZ output is enabled. |

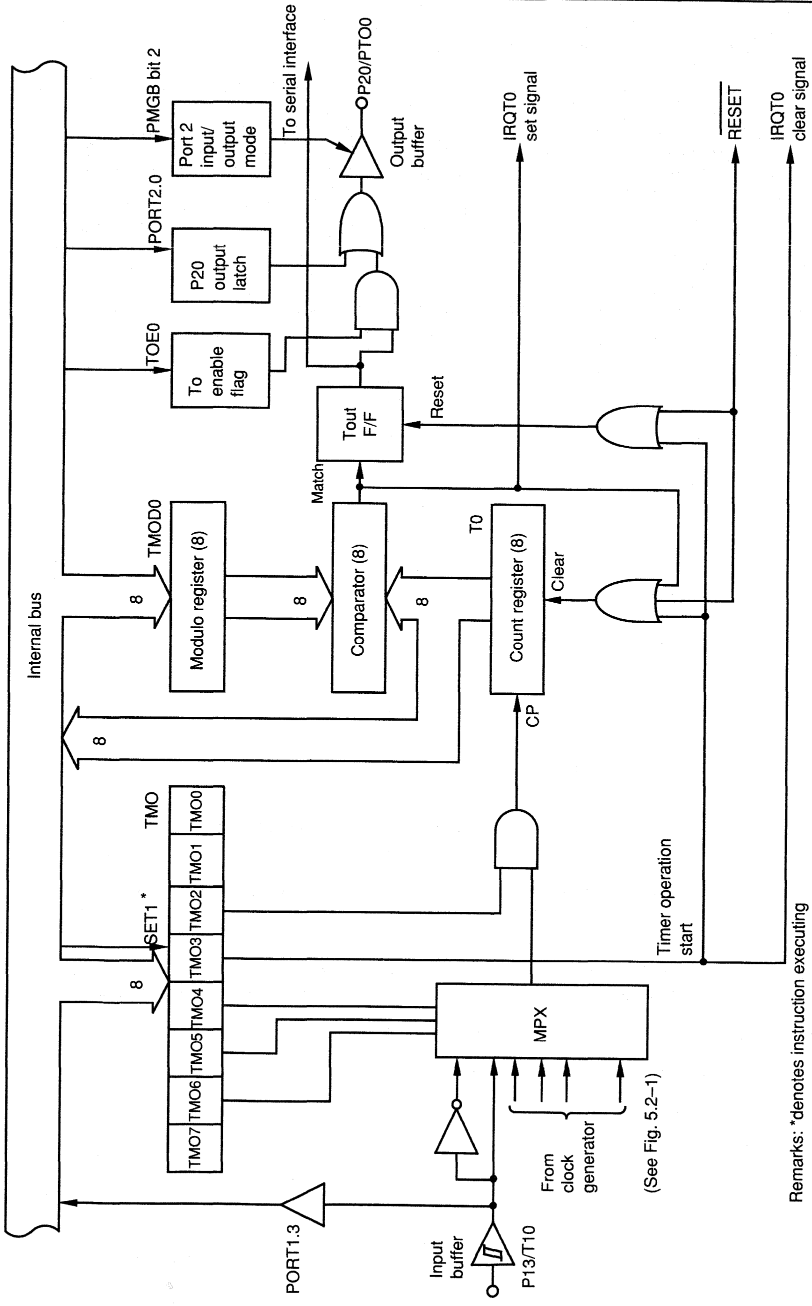
**Figure 5.4–2 Watch Mode Register Format**

### 5.5 Timer/event counter configuration

The μPD7500X incorporates a timer/event counter (one channel) as shown in Fig. 5.5–1.
The timer/event counter functions are as follows:

(a) Programmable interval timer operation
(b) Any desired frequency square wave output to PTO0 pin
(c) Event counter operation
(d) TI0 pin input is divided by N output to PTO0 pin (scaler operation).
(e) Real shift clock supply to serial interface circuit
(f) Count read function

Figure 5.5–1 Timer/Event Counter Block Diagram



Remarks: *denotes instruction executing

5.5.2 Basic configuration and operation of timer/event counter

The timer/event counter operation mode can be selected by using the timer/event counter mode register (TM0). The basic configuration and operation of the timer/event counter are explained below:

(1) Count pulse CP is selected by setting TM0 and input to the 8-bit count register T0.

(2) T0 is a binary 8-bit up counter incremented by one when CP is input. It is cleared when the RESET signal is generated, TM0 bit 3 is set (timer start), or coincidence signal is generated. T0 can be read at any time using an 8-bit memory operation instruction, but cannot be written

(3) The modulo register TMOD0 consists of eight bits to determine the T0 count. A value is set in TMOD0 using an 8-bit memory operation instruction, but TMOD0 cannot be read. When the RESET signal is generated, TMOD0 is initialized to FFH

(4) The comparator compares the T0 and TMOD0 contents. If they match, it generates a coincidence signal and sets the interrupt request flag (IRQT0)

Fig. 5.5-2 shows the count operation timing.

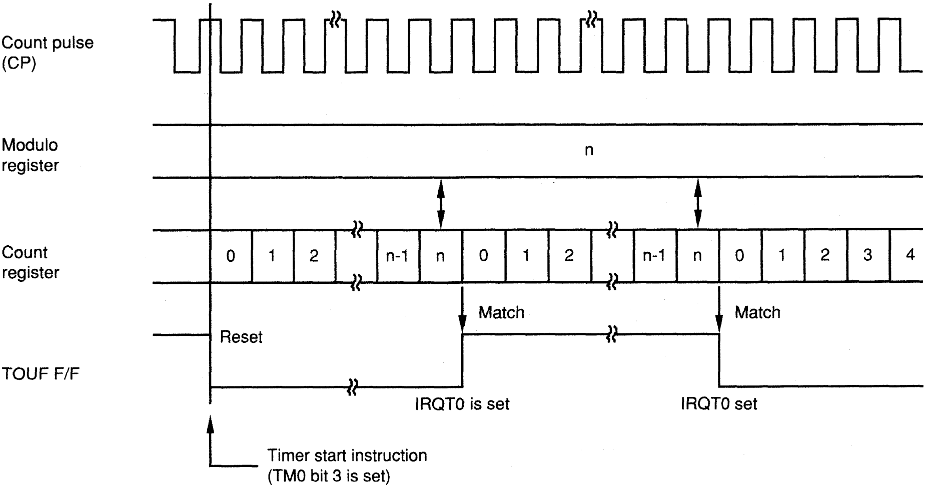


**Figure 5.5–2 Count Operation Timing Chart**

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

The mode register (TM0) consists of eight bits to control the timer/event counter. Fig. 5.5–3 shows the timer/event counter mode register format.

The timer mode register is set by using an 8-bit memory operation instruction.

Bit 3 is a timer start bit which can be set individually. Bit 3 is reset to 0 automatically when timer operation starts.

Example 1: To start the timer in interval timer mode with CP = 4.09 kHz.

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

Example 2: To restart the timer according to how the timer mode register is set.

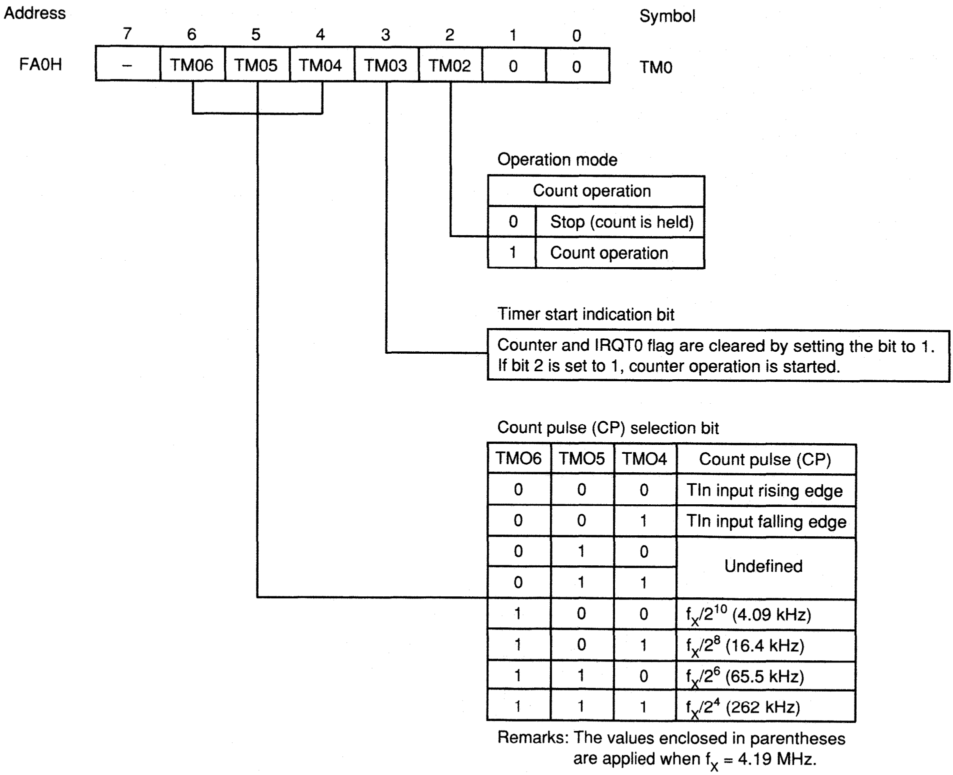
```
SEL     MB15    ;   or CLR1 MBE
SET1    TM0.3   ;   TM0.BIT3 ← 1
```

When the $\overline{\text{RESET}}$ signal is generated, all the timer mode register bits are cleared.

The timer/event counter output enable flag (TOE0) controls enable/disable of outputting the timer out F/F (TOUT F/F) state to the PTO0 pin. (See Fig. 5.5–4.) It is operated by using bit operation instruction and is enable to be writing.

The timer out F/F (TOUT F/F) is inverted by the coincidence signal received from the comparator. The timer out F/F is reset using an instruction to set timer mode register (TM0) bit 3.

When the $\overline{\text{RESET}}$ signal is generated, TOE0 and TOUT F/F are cleared.



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

Operation mode

| Count operation | |
|---|---|
| 0 | Stop (count is held) |
| 1 | Count operation |

Timer start indication bit

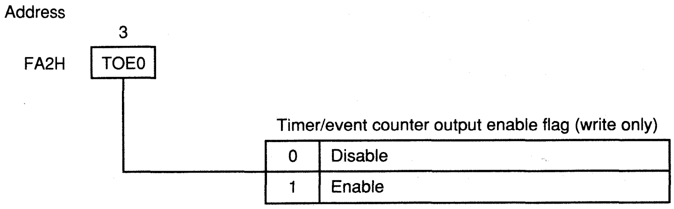
Counter and IRQT0 flag are cleared by setting the bit to 1.
If bit 2 is set to 1, counter operation is started.

Count pulse (CP) selection bit

| TMO6 | TMO5 | TMO4 | Count pulse (CP) |
|---|---|---|---|
| 0 | 0 | 0 | TIn input rising edge |
| 0 | 0 | 1 | TIn input falling edge |
| 0 | 1 | 0 | Undefined |
| 0 | 1 | 1 | |
| 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) |

Remarks: The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz.

**Figure 5.5–3 Timer/Event Coumter Mode Register Format**



Timer/event counter output enable flag (write only)

| | |
|---|---|
| 0 | Disable |
| 1 | Enable |

**Figure 5.5–4 Timer/Event Counter Output Enable Flag Format**

5.5.4 Timer/event counter operation mode

The timer/event counter operates in the count operation stop or count operation mode depending on how the mode register is set.

The following are always enabled independently of how the mode register is set

1) TI0 pin signal input and test. (P13 pin input test can be made.)
2) Output of timer out F/F state to PTO0.
3) Modulo register (TMOD0) setting.
4) Count register (T0) read.
5) Interrupt request flag (IRQT0) setting, clear, and test

(a) Count operation stop mode

The count operation stop mode is set when TM0 bit 2 is set to 0. Since count pulse (CP) supply to the count register is stopped, count operation is not performed

(b) Count operation mode

The count operation mode is set when TM0 bit 2 is set to 1. Count pulses selected by using bits 4 to 6 are supplied to the count register, and count operation is performed as shown in Fig. 5.5–2.

Normally, timer operation is started by

1) setting a count value in the modulo register (TMOD0), then
2) setting the operation mode, count clock, and start indication in the mode register (TM0).

The modulo register is set by using an 8-bit data transfer instruction.

Caution: Set a value other than 0 in the modulo register.

Example: To set 3FH in channel 0 modulo register.

```
SEL   MB15        ;   or CLR1 MBE
MOV   XA, #3FH
MOV   TMOD0, XA
```

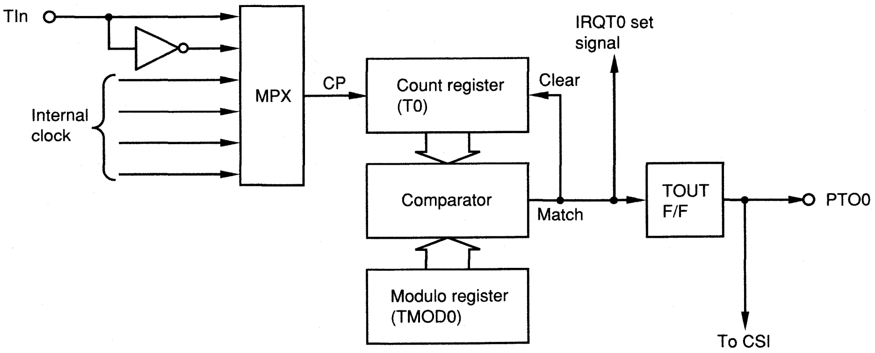


**Figure 5.5–5 Operation in Count Operation Mode**

5.5.5 Timer/event counter time setting

"Timer setup value" (period) is equal to "modulo register contents + 1" divided by "count pulse frequency" selected by setting the timer mode register.

$$T\ (s) = \frac{n+1}{F_{CP}}$$

T (s) : Timer setup value (seconds)
$F_{CP}$ (Hz) : Count pulse frequency (Hz)
n : Modulo register value ($n \neq 0$)

Once the timer is set, an interrupt request signal (IRQT0) is generated at the setup time intervals.

Table 5.5–1 lists the resolution and maximum setup value (time when FFH is set in the modulo register) of the timer/event counter for each count pulse.

Example: To produce 30 ms time intervals. ($f_X$ = 4.194304 MHz)
Use the mode with the maximum setup time 62.5 ms.

$$\frac{30\ \text{ms}}{244\ \mu\text{s}} = 122.9 = 7\text{AH}$$

Set 79H in the modulo register.

```
SEL    MB15
MOV    XA, #79H
MOV    TMOD0, XA
```

**Table 5.5–1 Resolution and Maximum Setup Value (4.19 MHz)**

| Mode register | | | Timer channel 0 | |
|---|---|---|---|---|
| TMO6 | TMO5 | TMO4 | Resolution | Maximum setup time |
| 1 | 0 | 0 | 244 μs | 62.5 ms |
| 1 | 0 | 1 | 61.1 μs | 15.6 ms |
| 1 | 1 | 0 | 15.3 μs | 3.91 ms |
| 1 | 1 | 1 | 3.81 μs | 977 μs |

### 5.5.6 Caution on timer/event counter application

(1) Error at timer start

An error with a maximum length of one clock period of the count pulse (CP) for the value calculated in 5.5.5 occurs in the time until a coincidence signal is generated after the timer starts (TM0.3 is set). This is because clearing of the count register is not synchronized with CP as shown in Fig. 5.5–6.

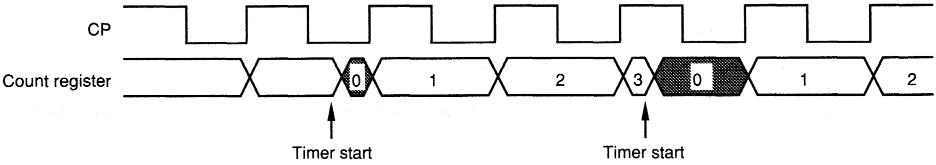


**Figure 5.5–6 Error at Timer Starting**

(2) Caution at timer start

Normally, the count register T0 and interrupt request flag IRQT0 are cleared when the timer starts (TM0 bit 3 is set). However, if the timer is placed in the operation mode, and IRQT0 setting and timer start occur simultaneously, IRQT0 may be unable to clear. There is no problem when IRQT0 is used for a vectored interrupt. In IRQT0 test application, however, a problem arises such than IRQT0 is set although the timer has been started. Thus, to start the timer at IRQT0 timing, stop the timer once (by setting TM0 bit 2 to 0), then restart it, or start the timer twice.

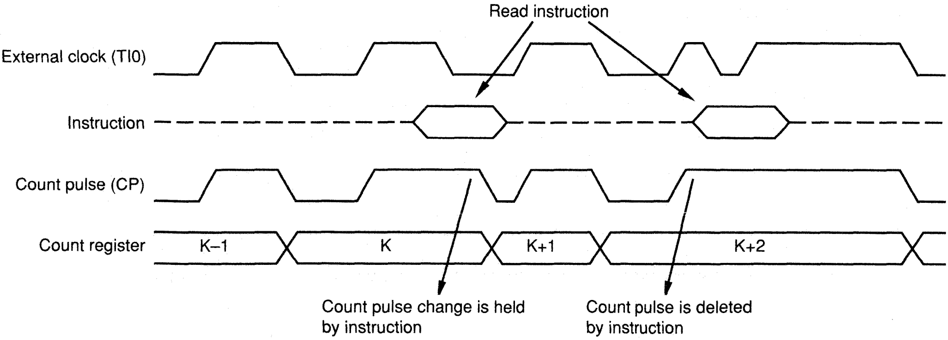
Example: Timer start at IRQT0 timing may be set

```
        SEL    MB15      ;   or CLR1 MBE
        MOV    XA, #0
        MOV    TM0, XA   ;   Timer stop
        MOV    XA, #4CH
        MOV    TM0, XA   ;   Restart
    or
        SEL    MB15      ;   or CLR1 MBE
        SET1   TM0.3
        SET1   TM0.3     ;   Restart
```

(3) Error at count register read

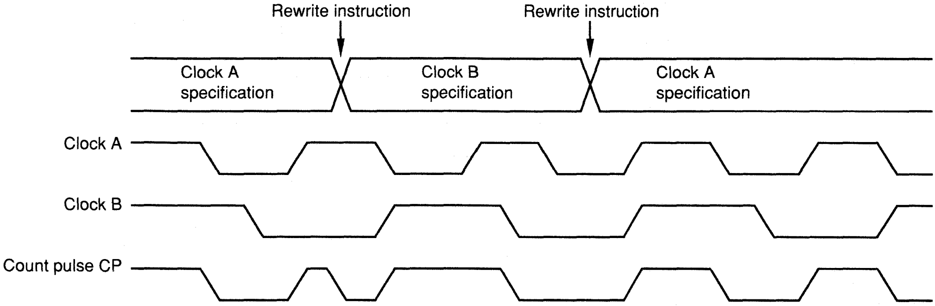
The count register contents can be read at any time using an 8-bit data memory operation instruction. During execution of the instruction, count pulse change is held and count register change is suppressed. Thus, if the count pulse signal source is input to TI0, count pulses as long as the instruction execution time are deleted. (This symptom does not occur if the internal clock is used for count pulses because they are synchronized with the instruction.)

Therefore, if TI0 is input as a count pulse and the count register contents are read, signals having a pulse width that prevents miscount, even if count pulses are deleted, must be input. Since the count hold period in read instruction execution is one machine cycle, the pulse width input to the TI0 pin must be longer than one machine cycle.

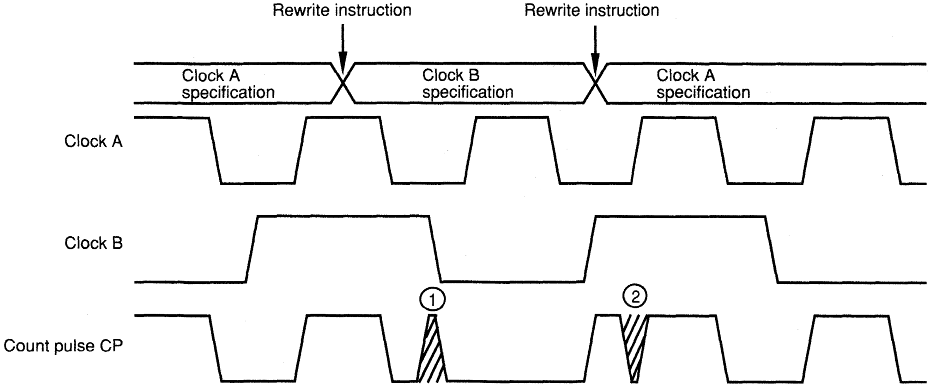


(4) Caution at count pulse change

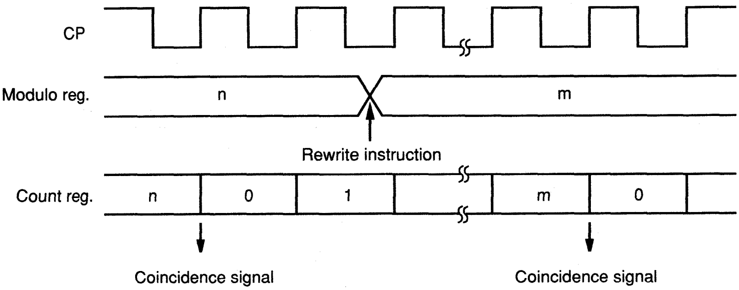
If the timer mode register is rewritten and count pulse change is made, its specification becomes effective immediately after instruction execution.



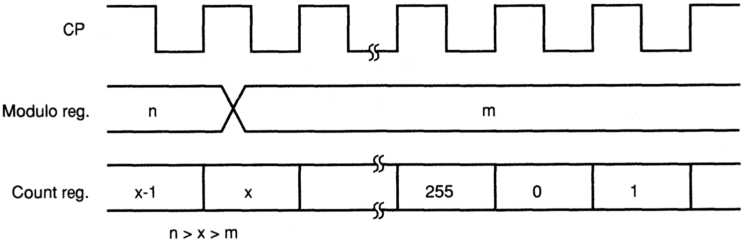
A count pulse ( ① or ② in the timing chart shown below) may occur depending on the clock combination when the count pulse changed. In this case, a miscount may occur or the count register contents may be destroyed. To change count pulse, be sure to set count mode register bit 3 to 1 and restart the timer at the same time.



(5) Operation after modulo register change
The modulo register is changed when an 8-bit data memory operation instruction is executed.



If the value appearing after the modulo register is changed and becomes smaller than the count register value, the count register continues counting and overflows, then restarts counting from 0. Thus, if the value after the modulo register is changed (m) and becomes smaller than the value before it was changed (n), the timer must be restarted after the modulo register is changed.



5.5.7 Timer/event counter application
(1) Timer 0 is applied to an interval timer generating an interrupt at 50 ms intervals.
— The high-order four bits of the mode register are set to 0100B and a maximum setup time of 62.5 ms is selected.
— The low-order four bits of the mode register are set to 1100B.
— The modulo register setup value is as follows:

$$\frac{50\text{ ms}}{244\ \mu\text{s}} = 205 = \text{CDH}$$

```
Example: SEL   MB15          ; or CLR1 MBE
         MOV   XA, #0CCH
         MOV   TMODO, XA     ; Modulo is set.
         MOV   XA, #01001100B
         MOV   TM0, XA       ; Mode is set and timer is started.
         EI                  ; Interrupts is enabled.
         EI    IET0          ; Timer interrupt is enabled.
```

Remarks: In this application, the TI0 pin can be used as an input pin.

(2) When the number of pulses input from the TI0 pin reaches 100, an interrupt is generated. (The pulses are active high.)
- — The high-order four bits of the mode register are set to 0000 and the rising edge is selected.
- — The low-order four bits of the mode register are set to 1100B.
- — The modulo register is 99 = 100 –1.

```
Example: SEL   MB15          ; or CLR1 MBE
         MOV   XA, #100 –1
         MOV   TMOD0, XA     ; Modulo is set.
         MOV   XA, #00001100B
         MOV   TM0, XA
         EI
         EI    IET0          ; INTT0 is enabled.
```

### 5.6 Serial Interface

5.6.1 Serial interface configuration
The μPD7500X has a clocked serial interface (CSI) as shown in Fig. 5.6-2.

5.6.2 Serial interface functions
The μPD7500X serial interface includes the three modes described in (1) to (3) below.
Since the serial clock line $\overline{\text{SCK}}$ and serial data bus lines SB0/SB1 enable software to determine the output level, any desired transfer format can be handled.

(1) 3-line serial I/O mode
- — Three lines of serial clock $\overline{\text{SCK}}$, serial output SO, and serial input SI
- — Clock synchronous 8-bit send and receive (simultaneous send and receive)
- — The serial transfer top can be changed between the most and least significant bits (MSB and LSB).
- — This mode enables the μPD7500X to be connected to the μPD7500 series, μCOM-75X family, μCOM-87 family, and various peripheral I/O devices.

(2) SBI (serial bus interface) mode
- — This mode conforms to the NEC serial bus format.
- — Communication can be made with a number of devices by using the serial clock $\overline{\text{SCK}}$ and serial data bus SB0 or SB1 lines.
- — Address, command, and data can be transferred, and a hardware function for discriminating the signals is included. (See Fig. 5.6-2.)
- — The acknowledge, busy signal output function and the wake-up function for handshaking are included.

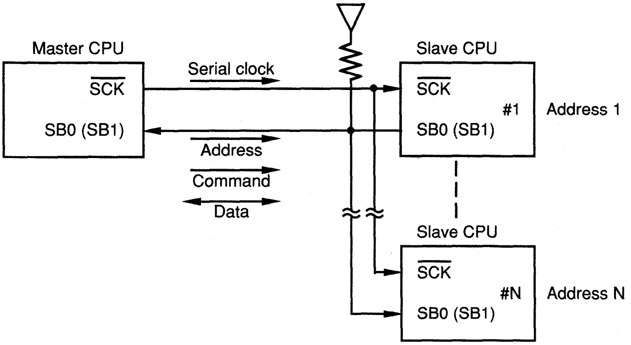


**Fig. 5.6-1 SBI System Configuration Example**

(3) 2-line serial I/O mode
- — Communication can be made by using the serial clock $\overline{\text{SCK}}$ and serial data bus SB0 or SB1 lines.
- — Communication can be made with a number of devices by using software to control the output level to the two lines. Any desired communication format can be handled.

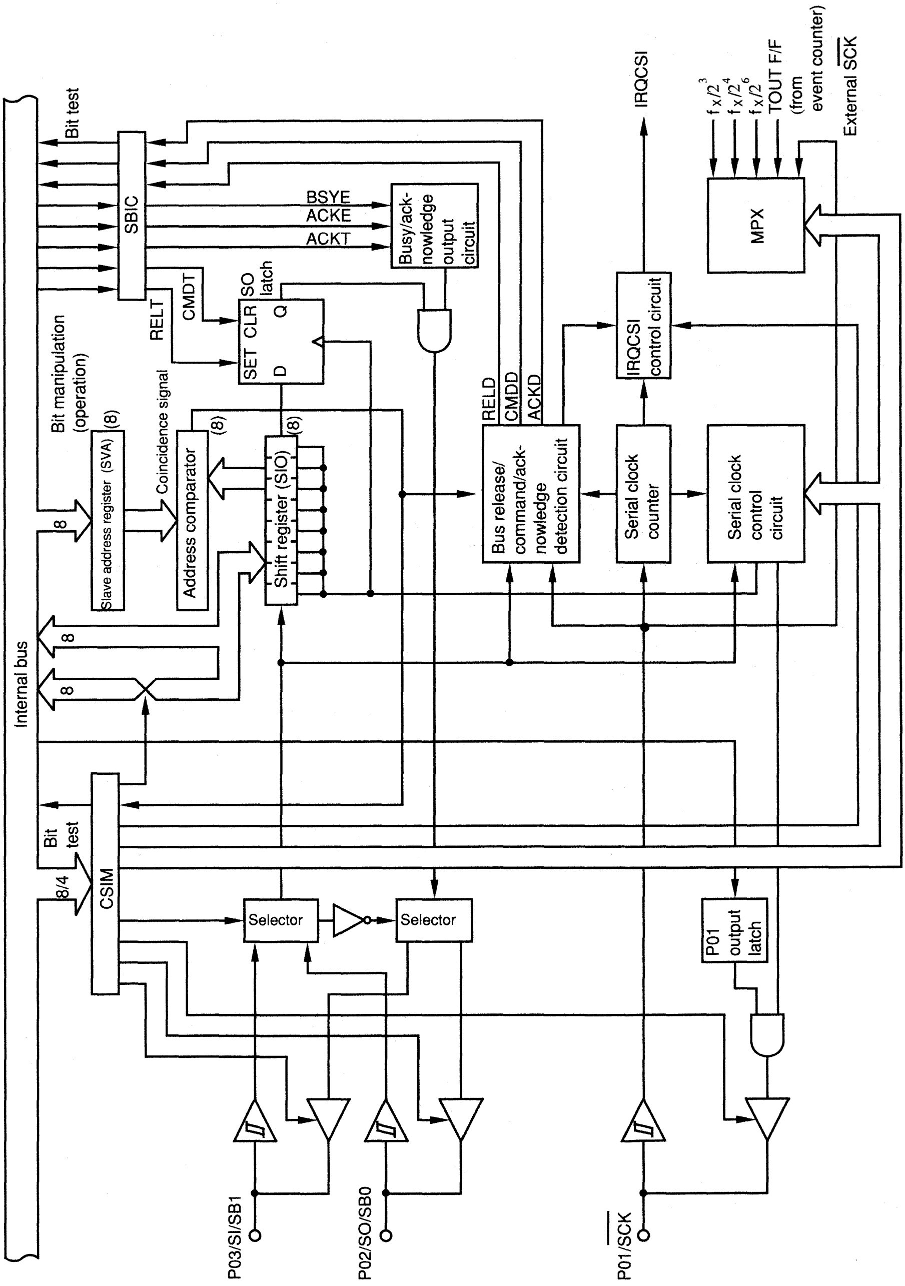


**Figure 5.6.2 Serial Interface Block Diagram**

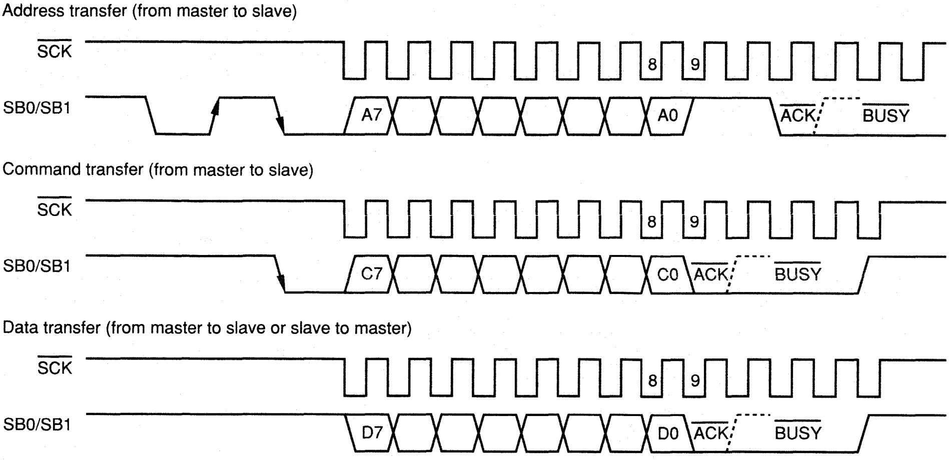

Address transfer (from master to slave)
SCK
SB0/SB1
8
9
A7
A0
ACK
BUSY
Command transfer (from master to slave)
SCK
SB0/SB1
8
9
C7
C0
ACK
BUSY
Data transfer (from master to slave or slave to master)
SCK
SB0/SB1
8
9
D7
D0
ACK
BUSY

**Figure 5.6–3 SBI Transfer Timing Chart**

5.6.3 Main register functions

(1) Serial operating mode register (CSIM)

The serial operation mode register (CSIM) consists of eight bits that specify the serial interface operating mode, serial clock, wake-up function, etc.
CSIM is set using an 8-bit memory operation instruction.
The high-order three bits of CSIM can be set bitwise using the bit name.
Bit 6 enables bit test only. Data written into the bit becomes invalid.

Example 1:
To select $f_X/2^4$ for serial clock, generate a serial interrupt IRQCSI at the end of each serial transfer, and make serial transfer in the SBI mode using the SB0 pin as the serial data bus line.

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

Example 2:
To enable serial transfer conforming to the CSIM contents.

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

Wake-up function specification bit (WUP):

(a) When WUP=0
IRQCSI is set each time a serial transfer ends.
WUP is set to 0 during normal transfer.

(b) When WUP = 1

WUP is set to 1 only during the SBI mode. IRQCSI is set only when the address received after the bus is released matches the slave address register (SVA) value (wake-up state). When the received address is not its own, an unnecessary interrupt will not be generated. The $\overline{\text{ACK}}$ signal is not output either. The SB0 (SB1) pin is placed in a high impedance state independent of the SO the latch state.

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

Fig. 5.6–4 shows the format of the serial operation mode register.

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FE0H | CSIE | COI | WUP | CSIM4 | CSIM3 | CSIM2 | CSIM1 | CSIM0 | CSIM |

Serial clock selection bits (W)

| CSIM1 | CSIM0 | Serial clock | | | $\overline{\text{SCK}}$ pin mode |
|---|---|---|---|---|---|
| | | 3-line serial I/O mode | SBI mode | 2-line serial I/O mode | |
| 0 | 0 | $\overline{\text{SCK}}$ pin input clock from the external | | | Input |
| 0 | 1 | Timer/event counter output (TO) | | | Output |
| 1 | 0 | $f_X/2^4$ (262 kHz) | | $f_X/2^6$ (65.5 kHz) | |
| 1 | 1 | $f_X/2^3$ (524 kHz) | | | |

The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz.

Serial interface operating mode selection bits (W)

| CSIM4 | CSIM3 | CSIM2 | Operating mode | Shift register bit order | SO pin function | SI pin function |
|---|---|---|---|---|---|---|
| X | 0 | 0 | 3-line serial I/O mode | $SIO_{7-0} \leftrightarrow XA$<br>(transfer starts at MBS) | SO/PO2<br>(CMOS output) | SI/PO3<br>(input) |
| | | 1 | | $SIO_{0-7} \leftrightarrow XA$<br>(transfer starts at LSB) | | |
| 0 | 1 | 0 | SBI mode | $SIO_{7-0} \leftrightarrow XA$<br>(transfer starts at MSB) | SB0/PO2<br>(N-channel open drain input/output) | PO3 input |
| 1 | | | | | PO2 input | SB1/PO3<br>(N-channel open drain input/output) |
| 0 | 1 | 1 | 2-line serial I/O mode | $SIO_{7-0} \leftrightarrow XA$<br>(transfer starts at MSB) | SB1/PO2<br>(N-channel open drain input/output) | PO3 input |
| 1 | | | | | PO2 input | SBI/PO3<br>(N-channel open drain input/output) |

(to be continued)

Fig. 5.6-4 Serial Operation Mode Register Format

Wake-up function specification bit (W)

| | | |
|---|---|---|
| WUP | 0 | IRQCSI is set each time a serial transfer ends in each mode. |
| | 1 | WUP is set to 1 only during the SBI mode. IRQCSI is set only when the address received after the bus is released matches the slave address register data (wake-up state). SB0 (SB1) is placed in the high impedance state |

Coincidence signal received from address comparator (R (bit test is only enabled)) Note 1, Note 2

| | | |
|---|---|---|
| COI | 0 | Slave address register and shift register data mismatch. |
| | 1 | Slave address register and shift register data match. |

Serial interface operation enable/disable specification bit (W) Note 3

| | | Shift register operation | Serial clock counter | IRQCSI flag | SO/SB0 and SI/SB1 pins |
|---|---|---|---|---|---|
| CSIE | 0 | Shift operation is disabled. | Clear | Hold | Port 0 function only |
| | 1 | Shift operation is enabled. | Count operation | Can be set. | Also used for port 0 according to each mode function |

**Figure 5.6–4 Serial Operation Mode Register Format (cont'd)**

Notes:

1. COI is only valid before serial transfer starts or after it is complete. An undefined value is read during serial transfer.
2. Data written into COI is ignored.
3. To use PO1/$\overline{\text{SCK}}$ pin as an input port, set the following below.
   • Set CSIM0 and CSIM1 bits to 0. (PO1/$\overline{\text{SCK}}$ pins are set to input mode.)
   • Set CSIE bit to 0. (Stop the serial interface operation.)
   When CSIM0=CSIM1=0 and CSIE=1 or CSIM0, CSIM1≠0 and CSIE=0, PO1/$\overline{\text{SCK}}$ pins are output to high level.

Remarks: (W): Data write is only enabled.
(R) : Data read is only enabled.

(2) Serial bus interface control register (SBIC)

The 8-bit serial bus interface control register (SBIC) consists of the serial bus state control bits and flags indicating the states of input data from serial bus. It is mainly used in the SBI mode.

Fig. 5.6–5 shows the SBIC format. SBIC is set or tested by using a bit operation (manipulation) instruction. When the $\overline{\text{RESET}}$ signal is generated, all the SBIC bits are cleared.

Cautions:

1. SBIC cannot be set by using 4- or 8-bit memory operation instruction.
2. In the 3- or 2-line serial I/O mode, use only the following two bits for SO latch control:
   (a) Bus release trigger bit (RELT): To set SO latch
   (b) Command trigger bit (CMDT): To clear SO latch
3. For the bus release, command, acknowledge, and busy signals, see 5.6.5 (3).

Example 1: To output command signal.

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

Example 2: To test RELD and CMDD and determine the receive data type for appropriate processing.

```
        SEL     MB15
        SKF     RELD    ;   RELD test
        BR      !ADRS
        SKT     CMDD    ;   CMDD test
        BR      !DATA
CMD:    ...             ;   Command transfer
DATA:   ...             ;   Data transfer
ADRS:   ...             ;   Address transfer
```

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FE2H | BSYE | ACKD | ACKE | ACKT | CMDD | RELD | CMDT | RELT | SBIC |

Bus release trigger bit (W)

| | |
|---|---|
| RELT | SO latch is set to 1 by setting the bit. It is used to output bus release signal. After SO latch is set, the bit is automatically cleared. |

Command trigger bit (W)

| | |
|---|---|
| CMDT | SO latch is cleared by setting the bit. It is used to output a command signal. After SO latch is cleared, the bit is automatically cleared. |

Bus release detection flag (R)

| | | |
|---|---|---|
| RELD | Clear condition (RELD=0) | 1 When the transfer start is indicated.<br>2 When the address received after the bus is released does not match the slave address register (SVA) data.<br>3 When the $\overline{\text{RESET}}$ signal is input. |
| | Setting condition (RELD=1) | When the address received after the bus is released matches the slave address register data. (Wake up) |

Command detection flag (R)

| | | |
|---|---|---|
| CMDD | Clear condition (CMDD=0) | 1 When the transfer start is indicated.<br>2 When the bus release signal is detected.<br>3 When the $\overline{\text{RESET}}$ signal is input. |
| | Setting condition (CMDD=1) | When the command signal is detected. |

Acknowledge trigger bit (W)

| | | |
|---|---|---|
| ACKT | Used only after transfer completion | Acknowledge signal is output during one clock period of $\overline{\text{SCK}}$ immediatly after execution of the set instruction. |

Remarks:

1. ACKT is automatically cleared after acknowledge signal is output. 2. ACKT cannot be cleared using software.
3. To set ACKT, set ACKE to 0.

Acknowledge enable bit (R/W)

| | | | |
|---|---|---|---|
| ACKE | 0 | Automatic acknowledge signal output is disabled (the signal can be output by setting the acknowledge trigger bit ACKT). | |
| | 1 | Before transfer completion | Acknowledge signal is output during the ninth clock period of $\overline{\text{SCK}}$ (automatically output by presetting ACKE to 1). |
| | | After transfer completion | Acknowledge signal is output during one clock period of $\overline{\text{SCK}}$ immediately after execution of the set instruction (automatically output by presetting ACKE to 1). |

Acknowledge detection flag (R)

| | | |
|---|---|---|
| ACKD | Clear condition (ACKD=0) | 1 When the transfer is started.<br>2 When the $\overline{\text{RESET}}$ is input. |
| | Setting conditions (ACKD=1) | When the acknowledge signal is detected. |

Synchronous busy enable bit (R/W)

| | | |
|---|---|---|
| BSYE | 0 | Synchronous busy signal output is disabled. Synchronous busy signal output is stopped in synchronization with the $\overline{\text{SCK}}$ falling edge immediately after execution of the clear instruction. |
| | 1 | Synchronous busy signal is output on the $\overline{\text{SCK}}$ falling edge following an acknowledge signal. |

Remarks:

(R): Read is only enabled. (W): Write is only enabled. (R/W): Both read and write are enabled.

**Figure 5.6–5 SBIC Format**

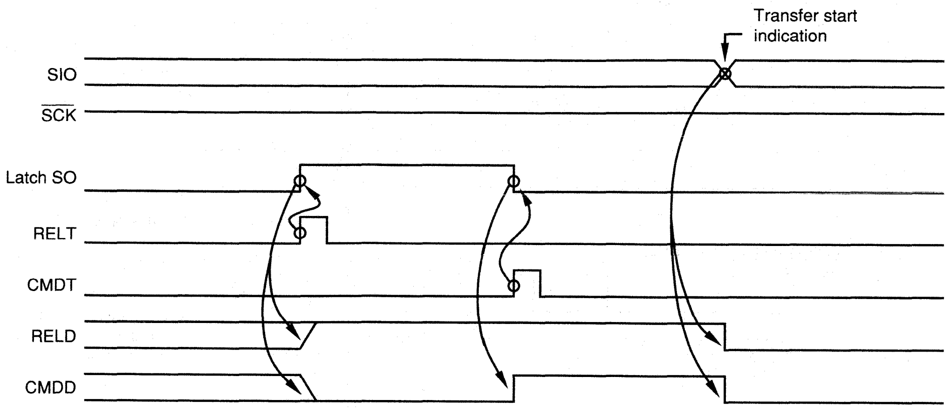


**Figure 5.6–6 RELT, CMDT, RELD, CMDD Operation**

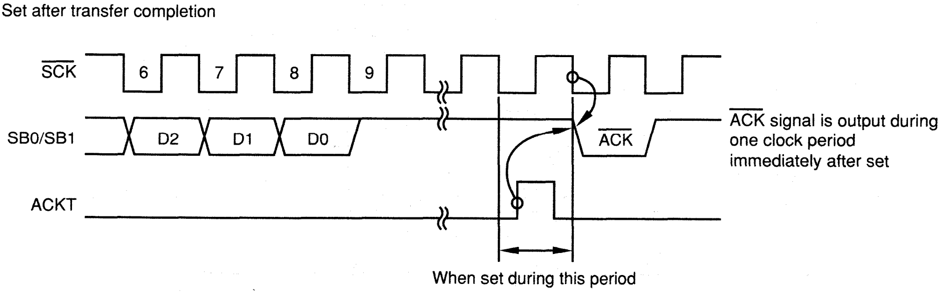
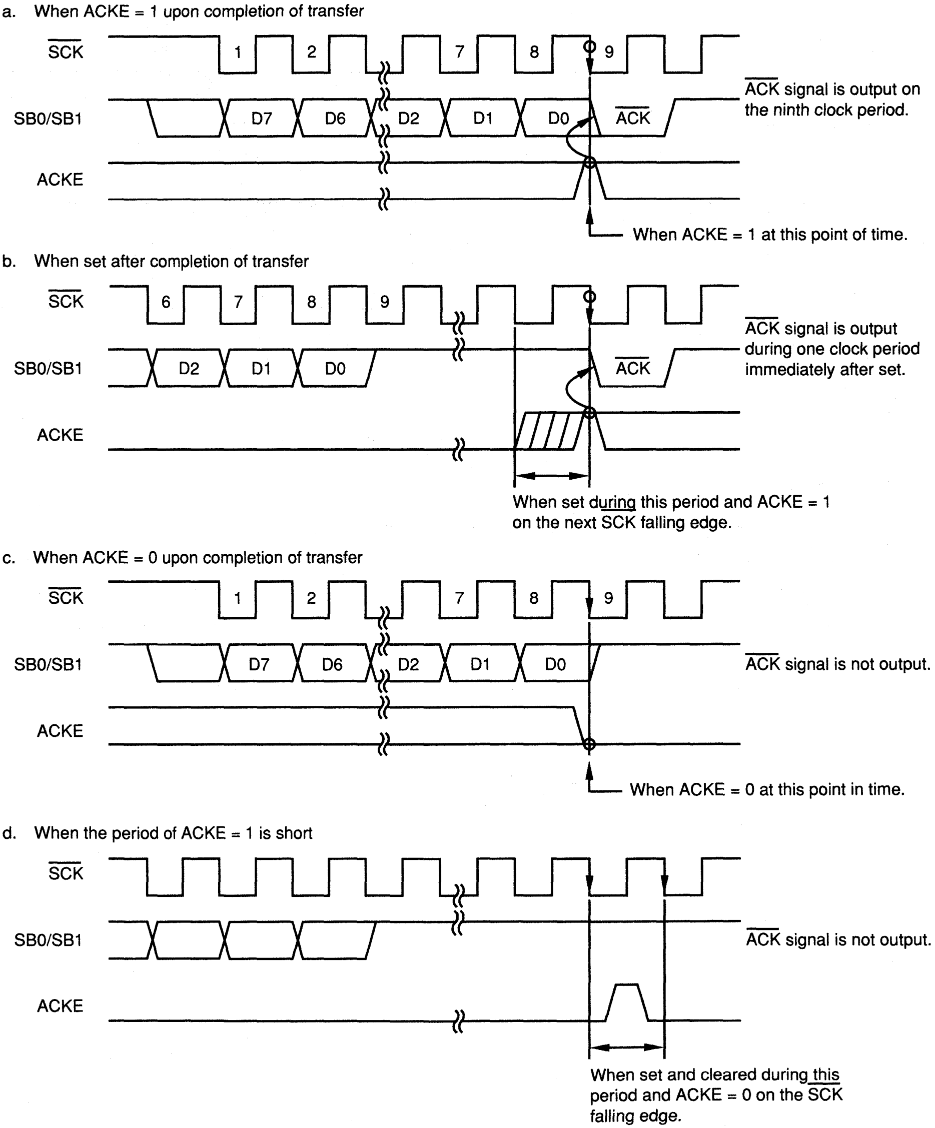

Set after transfer completion
SCK
6
7
8
9
SB0/SB1
D2
D1
D0
ACK
ACKT
ACK signal is output during one clock period immediately after set
When set during this period

**Figure 5.6–7 ACKT Operation**


a. When ACKE = 1 upon completion of transfer

$\overline{\text{ACK}}$ signal is output on the ninth clock period.

When ACKE = 1 at this point of time.

b. When set after completion of transfer

$\overline{\text{ACK}}$ signal is output during one clock period immediately after set.

When set during this period and ACKE = 1 on the next SCK falling edge.

c. When ACKE = 0 upon completion of transfer

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

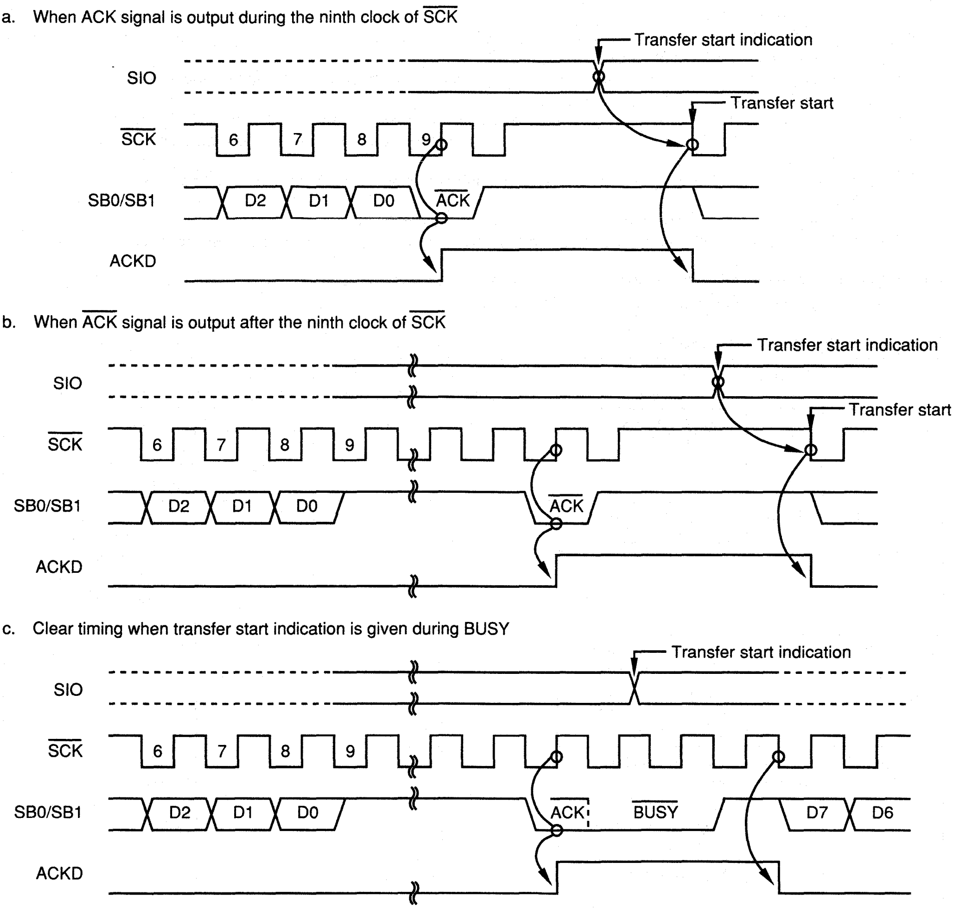
When ACKE = 0 at this point in time.

d. When the period of ACKE = 1 is short

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

When set and cleared during this period and ACKE = 0 on the $\overline{\text{SCK}}$ falling edge.

**Figure 5.6–8 ACKE Operation**

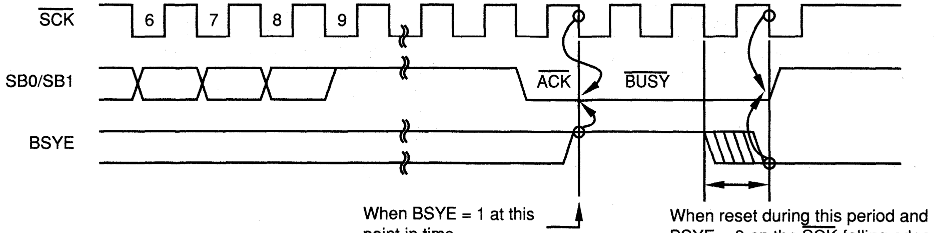


a. When ACK signal is output during the ninth clock of $\overline{\text{SCK}}$

b. When $\overline{\text{ACK}}$ signal is output after the ninth clock of $\overline{\text{SCK}}$

c. Clear timing when transfer start indication is given during BUSY

**Fig. 5.6-9 ACKD Operation**



When BSYE = 1 at this point in time

When reset during this period and BSYE = 0 on the $\overline{\text{SCK}}$ falling edge

**Figure 5.6–10 BSYE Operation**

(3) Slave address register (SVA)

When the μPD7500X is connected to the serial bus as a slave device in the SBI mode, the 8-bit slave address register (SVA) is used to set the slave address of the μPD7500X.

The SVA value is compared with received 8-bit data by the address comparator. If a match is found between them, serial operating mode register (CSIM) bit 6 (COI) is set to 1.

If a match is not found when the address is received, the bus release detection flag (RELD) is cleared. When WUP = 1, IRQCSI is set only if a match is found (wake-up is detected). This interrupt request can be used to know that a communication request is sent from the master to μPD7500X.

SVA can also be used to detect an error when the μPD7500X sends address, command, or data as a master device. (See 5.6.6.)

SVA is set using an 8-bit memory operation instruction. It can only be written.

When the $\overline{\text{RESET}}$ signal is generated in a mode which is not standby, the SVA value becomes undefined.

5.6.4 Signals in SBI mode

Tables 5.6–1 and 5.6–2 list the signals used in the SBI mode.

**Table 5.6–1 Signals in SBI Mode (I)**

| Signal name | Output device | Definition | Timing chart | Output condition | Flag influence | Explaination |
|---|---|---|---|---|---|---|
| Bus release signal (REL) | Master | SB0 (SB1) rising edge when $\overline{\text{SCK}}$ = 1 | (Figure 1) | • RELT is set. | • RELD is set.<br>• CMDD is cleared. | The signal is followed by CMD signal output indicating that the send data is an address |
| Command signal (CMD) | Master | SB0 (SB1) falling edge when $\overline{\text{SCK}}$ = 1 | (Figure 2) | • CMDT is set. | • CMDD is set. | i) After REL signal is output, send data is an address.<br>ii) When REL signal is not output, send data is a command. |
| Acknowledge signal ($\overline{\text{ACK}}$) | Slave | Low signal output to SB0 (SB1) during one clock period of $\overline{\text{SCK}}$ after completion of serial reception | (Figure 3) | 1 ACKE = 1<br>2 ACKT is set. | • ACKD is set. | Completion of reception |
| Busy signal ($\overline{\text{BUSY}}$) | Master/ Slave | (Synchronous busy signal)<br>Low signal output to SB0 (SB1) following acknowledge signal | | • BSYE = 1 | – | Serial reception cannot be done because processing is being performed. |
| | | (Asynchronous busy signal)<br>Low signal output to SB0 (SB1) (except during serial transfer). It is not synchronized with $\overline{\text{SCK}}$. | | • CMDT is set. | – | |
| Ready signal (READY) | Slave | High signal output to SB0 (SB1) before start or after completion of serial transfer | | 1 BSYE = 0<br>2 Execution of SIO data write instruction (transfer start indication) | – | Serial reception can be done. |

**Table5.6–2 Signals in SBI Mode (II)**

| Signal name | Output device | Definition | Timing chart | Output condition | Flag influence | Explanation |
|---|---|---|---|---|---|---|
| Serial clock ($\overline{\text{SCK}}$) | Master | Synchronous clock to output address, command, data, $\overline{\text{ACK}}$ signal, synchronous BUSY signal, etc. Address, command, or data is transferred with the first eight. | (Figure 4) | Execution of SIO data write instruction when CSIE =1 (serial transfer start indication) (Note 2) | IRQCSI is set (on the rising edge of ninth clock). (Note 1) | Signal output timing to serial data bus |
| Address (A7–0) | Master | 8-bit data transferred in synchronization with $\overline{\text{SCK}}$ after REL and CMD signals are output. | (Figure 5) | | (Note 1) | Slave device address value on serial bus |
| Command (C7–0) | Master | 8-bit data transferred in synchronization with $\overline{\text{SCK}}$ after CMD signal only is output (REL signal is not output). | (Figure 6) | | None | Indication message sent to slave device |
| Data (D7–0) | Master or Slave | 8-bit data transferred in synchronization with $\overline{\text{SCK}}$ when neither REL nor CMD signal is output. | (Figure 7) | | None | Numeric data processed by slave or master device |

Notes: 1. When WUP = 0, IRQCSI is always set on the ninth clock $\overline{\text{SCK}}$ rising edge.
When WUP = 1, IRQCSI is set only when the received address matches the value in the slave address register (SVA).
2. In the $\overline{\text{BUSY}}$ state, transfer is started after the READY state is set.

Figure 1:

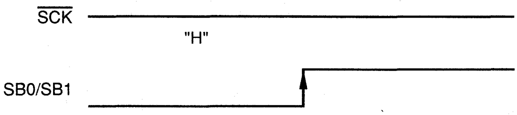


Figure 2:

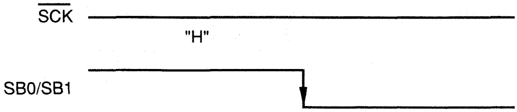


Figure 3:

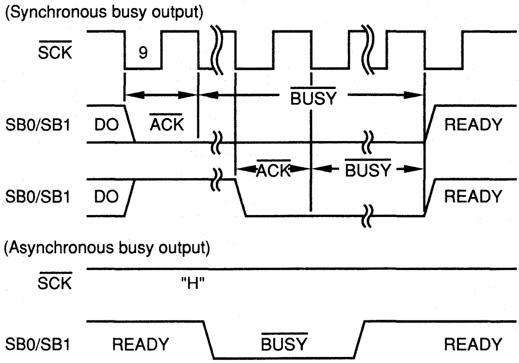


Figure 4:

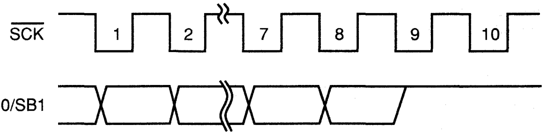


Figure 5:

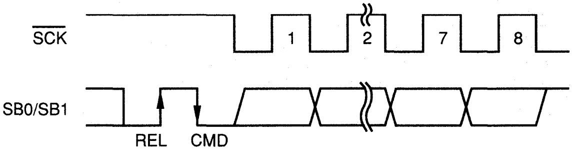


Figure 6:

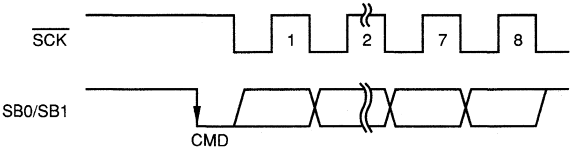
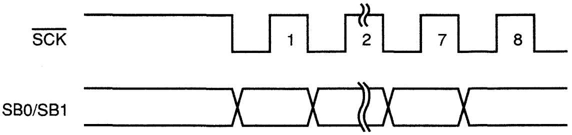


Figure 7:



5.6.5 Serial interface operation

The serial interface includes the following four operating modes.

— Operation stop mode
— 3-line serial I/O mode
— SBI mode
— 2-line serial I/O mode

**Table 5.6-3 Serial Interace Operation Mode**

| CSIM7 | CSIM3 | CSIM2 | Operating mode |
|---|---|---|---|
| 0 | — | — | Operation stop |
| 1 | 0 | — | 3-line serial I/O |
| 1 | 1 | 0 | SBI mode |
| 1 | 1 | 1 | 2-line serial I/O |

(1) Operation stop mode

When CSIE = 0, the serial interface is placed in the operation stop mode. In this mode, no serial transfer is done. The operation stop mode is set when the serial interface is not used, so that the power consumption is reduced. In this mode, the shift register does not perform shift operations and can be used as a normal 8-bit register. All the P01/$\overline{\text{SCK}}$, P02/SO/SB0, and P03/SI/SB1 pins are placed in the high impedance state and used only for the input port function. The P01/$\overline{\text{SCK}}$ pin is placed in the state listed in Table 5.6-4 according to how CSIM1 and CSIM0 are set.

**Table 5.6-4 CSIM1 and CSIM0 Settings**

| CSIM1 | CSIM0 | Serial clock | P01/$\overline{\text{SCK}}$ pin state |
|---|---|---|---|
| 0 | 0 | External clock | High impedance |
| 0 | 1 | Internal clock | A high level is output. However, a serial clock is output during serial transfer. |
| 1 | 0 | | |
| 1 | 1 | | |

Note: For the serial clock, see Fig. 5.6-4.

When the $\overline{\text{RESET}}$ signal is generated, the operation stop mode is set.

(2) 3-line serial I/O mode

The 3-line serial I/O mode is compatible with the mode used by the μPD7500 series or other μCOM-75X family devices. Fig. 5.6-11 shows the 3-line serial I/O mode operation timing.

Serial transfer start is indicated by executing an instruction to write data into the shift register (SIO), MOV or XCH.

Be sure to give a start indication when CSIE = 1.

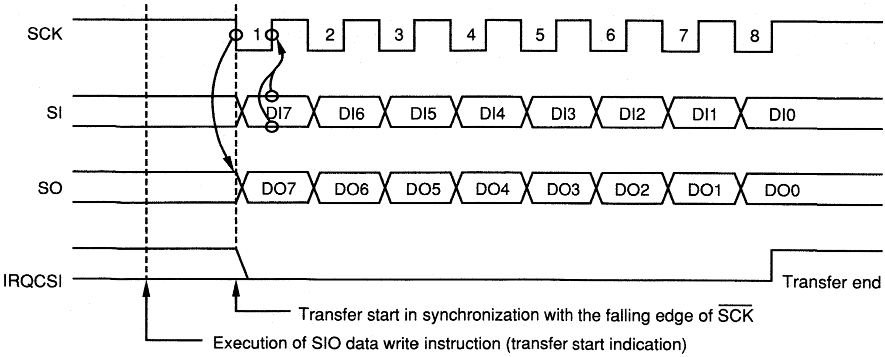

SCK
1
2
3
4
5
6
7
8
SI
DI7
DI6
DI5
DI4
DI3
DI2
DI1
DI0
SO
DO7
DO6
DO5
DO4
DO3
DO2
DO1
DO0
IRQCSI
Transfer end
Transfer start in synchronization with the falling edge of $\overline{\text{SCK}}$
Execution of SIO data write instruction (transfer start indication)

**Fig. 5.6-11 3-line Serial I/O Mode Timing**

In this mode, shift register shift operation is performed in synchronization with the serial clock ($\overline{\text{SCK}}$) falling edge, and send data is held in the SO latch and output from the SO pin. The receive data input to the SI pin is latched in the shift register on the rising edge of $\overline{\text{SCK}}$.
Since the SO pin is used as a CMOS output and outputs the SO latch state, the SO pin output state can be handled by setting the RELT and CMDT bits. However, do not try to change it during serial transfer.
The SCK pin output state can be controlled if the P01 output latch is operated in the output mode (internal clock mode), as described in 5.6.7.
A serial clock can be selected from among four clocks, as listed in Table 5.6-5, by setting the mode register.
Normally, shift operation is started by:
1) Setting the operating mode and serial clock selection data in the mode register (CSIM)
2) Setting transfer data in the shift register (SIO) (serial operation is started by executing the SIO data write instruction)
Serial transfer automatically stops at the end of 8-bit transfer, and the interrupt request flag (IRQCSI) is set.

**Table 5.6-5 Serial CLock Selection and Application**

| Mode register | | | Serial clock | Timing at which shift register can be read/written and serial transfer can be started | Application |
|---|---|---|---|---|---|
| CSIM 1 | CSIM 0 | Source | Serial clock mask | | |
| 0 | 0 | External $\overline{\text{SCK}}$ | Automatic mask at the end of 8-bit data transfer. | Only when serial transfer stops or $\overline{\text{SCK}}$ is high.<br>(Note) | Slave CPU |
| 0 | 1 | TOUT F/F | | | Half-duplex asynchronous transfer (under software control) |
| 1 | 0 | $f_X/2^4$ | | | Medium-speed serial transfer |
| 1 | 1 | $f_X/2^3$ | | | High-speed serial transfer |

Note: The serial transfer stops in the opertaion stop mode or when the serial clock is masked after 8-bit transfer has been made.

The shift register is read and written using 8-bit transfer instructions. At that time, LSB and MSB can be inverted by setting CSIM bit 2. This function enables the transfer top bit to be changed between LSB and MSB.

Example:
At the same time RAM data specified in the HL register pair is transfered to SIO, the SIO data is read into the accumulator, and serial transfer is started.

```
MOV  XA, @HL    ; Send data is taken out of RAM.
SEL  MB15       ; or CLR1 MBE
XCH  XA, SIO    ; Send data and receive data are exchanged, and transfer is started.
```

Caution:
The transfer top bit LSB or MSB is selected by changing the bit order in writing data into the shift register. The shift order of the shift register bit is always the same. Thus, even if the transfer top bit is changed (LSB to MSB or MSB to LSB) after data is written into the shift register, the data is transferred in the bit order specified before the transfer top bit is changed.

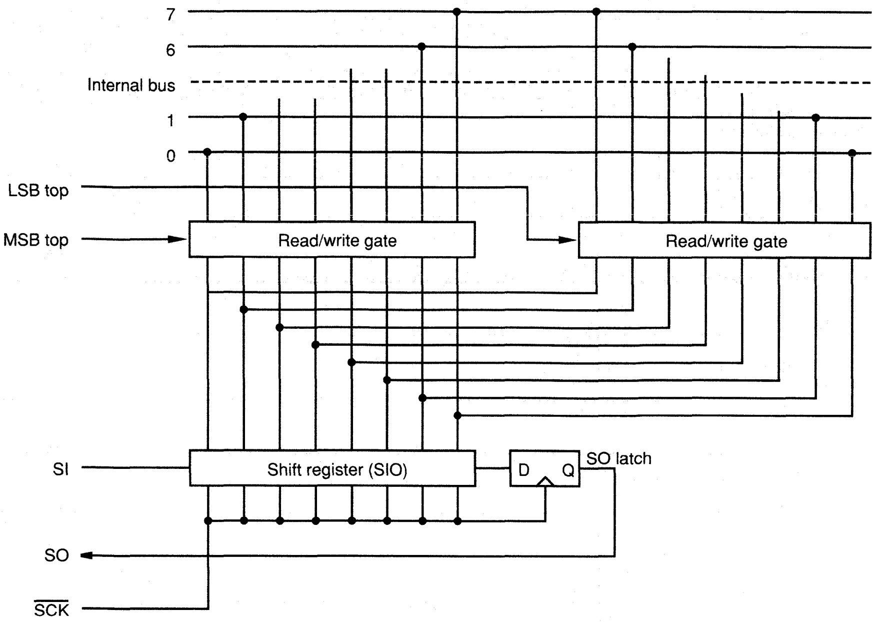


**Fig. 5.6-12 3-line I/O Mode Format**

(3) SBI mode

The SBI mode enables the μPD7500X to communicate with a number of devices by using the $\overline{\text{SCK}}$ and SB0 or SB1 lines. Figs. 5.6-13 to 5.6-15 show the timing of operations according to the type of data to be transferred.

In the SBI mode, indication of serial transfer start is also given by executing the instruction to write data into the shift register (SIO), MOV or XCH. Be sure to give a start indication when CSIE = 1.

Shift register shifting is made in synchronization with the serial clock ($\overline{\text{SCK}}$) falling edge, and send data is held in the SO latch and output from the SB0/P02 or SB1/P03 pin starting at MSB. Receive data input to the SB0 or SB1 pin is latched in the shift register on the rising edge of $\overline{\text{SCK}}$.

The SB0 or SB1 pin specified for the serial data bus is used as N-channel open drain input/output and needs pull-up. When data is received, the N-channel transistor must be turned off.

By writing FFH into SIO and shifting it, the N-channel transistor can always be turned off during transfer. When the wake-up function specification bit (WUP) is set to 1, however, the N-channel transistor is always turned off, and FFH need not be written into SIO before reception.

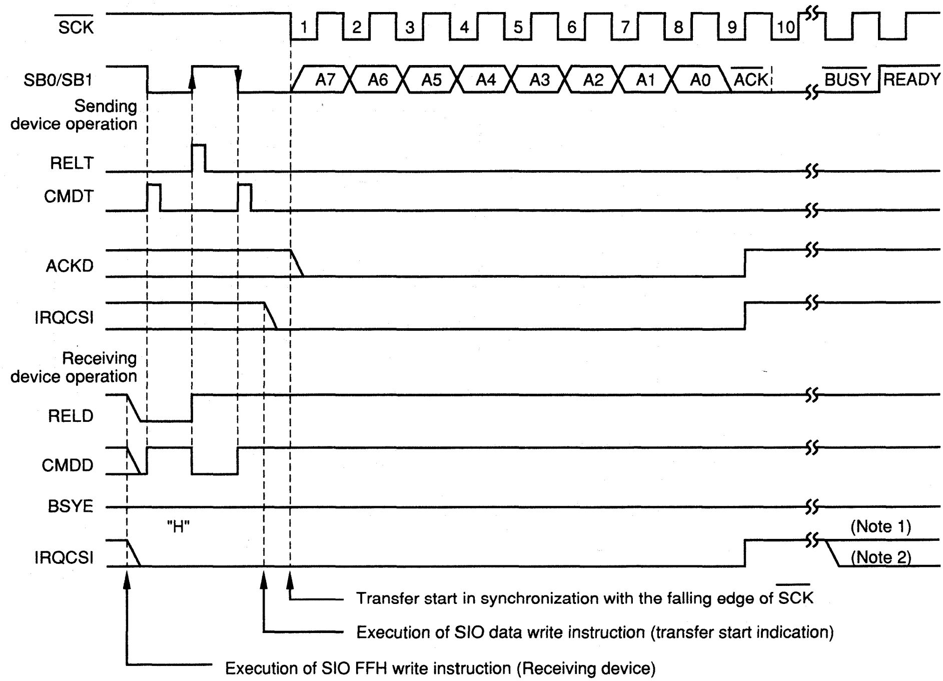

SCK
1
2
3
4
5
6
7
8
9
10
SB0/SB1
A7
A6
A5
A4
A3
A2
A1
A0
ACK
BUSY
READY
Sending device operation
RELT
CMDT
ACKD
IRQCSI
Receiving device operation
RELD
CMDD
BSYE
"H"
(Note 1)
(Note 2)
IRQCSI
Transfer start in synchronization with the falling edge of SCK
Execution of SIO data write instruction (transfer start indication)
Execution of SIO FFH write instruction (Receiving device)

**Fig. 5.6-13 SBI Mode Timing (Address Transfer)**

Note 1: Where SIO's data is read out by MOV instruction, then BSYE flag is cleared.
Note 2: Where SIO's data is exchanged (read/write) by XCH instruction.

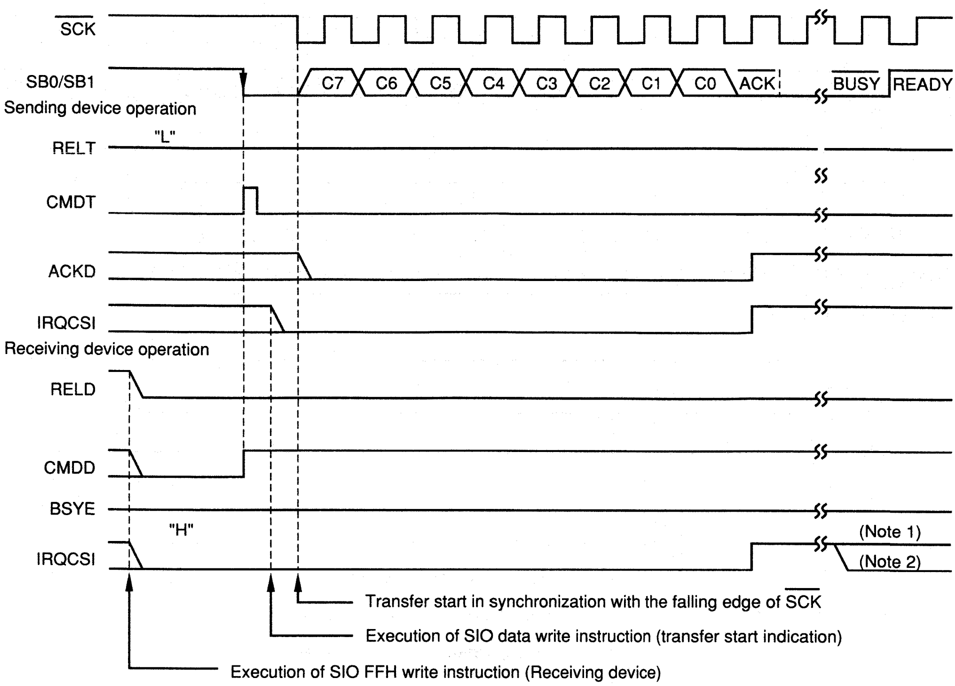

SCK
SB0/SB1
Sending device operation
C7
C6
C5
C4
C3
C2
C1
C0
ACK
BUSY
READY
RELT
"L"
CMDT
ACKD
IRQCSI
Receiving device operation
RELD
CMDD
BSYE
"H"
IRQCSI
(Note 1)
(Note 2)
Transfer start in synchronization with the falling edge of SCK
Execution of SIO data write instruction (transfer start indication)
Execution of SIO FFH write instruction (Receiving device)

**Fig. 5.6-14 SBI Mode Timing (Command Transfer)**

Note 1: Where SIO's data is read out by MOV instruction, then BSYE flag is cleared.
Note 2: Where SIO's data is exchanged (read/write) by XCH instruction.

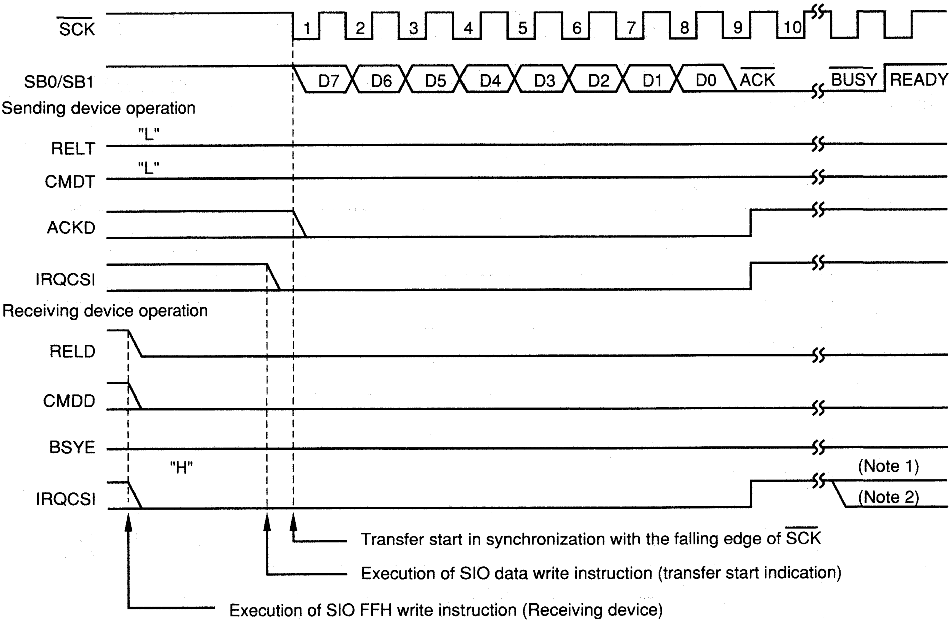

SCK
1
2
3
4
5
6
7
8
9
10
SB0/SB1
D7
D6
D5
D4
D3
D2
D1
D0
ACK
BUSY
READY
Sending device operation
RELT
"L"
CMDT
"L"
ACKD
IRQCSI
Receiving device operation
RELD
CMDD
BSYE
"H"
IRQCSI
(Note 1)
(Note 2)
Transfer start in synchronization with the falling edge of SCK
Execution of SIO data write instruction (transfer start indication)
Execution of SIO FFH write instruction (Receiving device)

**Fig. 5.6-15 SBI Mode Timing (Data Transfer)**

Note 1: Where SIO's data is read out by MOV instruction, then BSYE flag is cleared.
Note 2: Where SIO's data is exchanged (read/write) by XCH instruction.

(4) 2-line serial I/O mode

The 2-line serial I/O mode is used for communications using the SCK and SB0 or SB1 lines. The mode can handle any desired communication format by using software to control the SCK and SB0 or SB1 pin output levels.

Fig. 5.6-16 shows the 2-line serial I/O mode timing.

Serial transfer start indication is given by executing the instruction writing data into the shift register (SIO), MOV or XCH. Be sure to give start indication when CSIE = 1.

Shift register shifting is done in synchronization with the falling edge of the serial clock (SCK), and send data is held in the SO latch and output from the SB0/P02 or SB1/P03 pin starting at MSB. Receive data input from the SB0 or SB1 pin is latched in the shift register on the SCK rising edge.

The SB0 or SB1 pin specified for the serial data bus is used as an N-channel open drain input/output and needs pull-up. When data is received, the N-channel transistor must be turned off, thus FFH is previously written into SIO.

Since the SB0 or SB1 pin outputs the SO latch state, the SB0 or SB1 pin output state (high impedance or low level) can be controlled by setting the RELT and CMDT bits. However, do not change it during serial transfer.

In the output mode (internal clock mode), the SCK pin output state can also be controlled by using the P01 output latch, as described in 5.6.7.

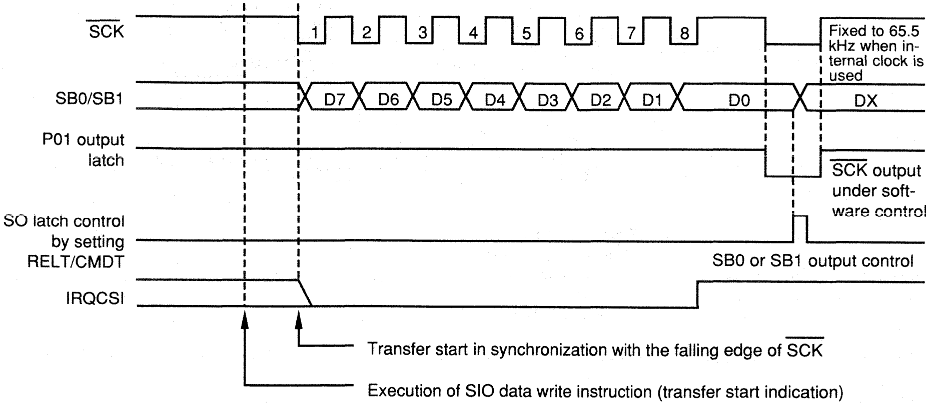

SCK
1
2
3
4
5
6
7
8
Fixed to 65.5 kHz when internal clock is used
SB0/SB1
D7
D6
D5
D4
D3
D2
D1
D0
DX
P01 output latch
SCK output under software control
SO latch control by setting RELT/CMDT
SB0 or SB1 output control
IRQCSI
Transfer start in synchronization with the falling edge of SCK
Execution of SIO data write instruction (transfer start indication)

**Fig. 5.6-16 2-line Serial I/O Mode Timing**

5.6.6 Error detection

Since the data of the sending serial bus SB0 or SB1 is also read by the shift register SIO of the device which sends data in the SBI or 2-line serial I/O mode, a send error can be detected as described below:
(1) SIO data comparison before the start of a send and after the end of a send
If both the pieces of data do not match, it is determined that a send error occurred.
(2) Use of slave address register (SVA)
Send data is also set in SVA. After the completion of a send, the COI bit (coincidence signal from the address comparator) in the serial operating mode register is tested. If the bit is set to 1, it is determined that normal sending has been performed; if 0, it is determined that a send error has occurred.

5.6.7 $\overline{\text{SCK}}$ pin output handling
The $\overline{\text{SCK}}$/P01 pin, which incorporates an output latch, can also produce static output by software control in addition to a normal serial clock.
The number of $\overline{\text{SCK}}$s can be set as desired by using software to control the P01 output latch (the SO/SB0, and SI/SB1 pins are controlled by setting the RELT and CMDT bits).
The $\overline{\text{SCK}}$/P01 pin output control method is described below:
(1) The serial operation mode register (CSIM) is set ($\overline{\text{SCK}}$ pin: Output mode, serial operation: Enabled). $\overline{\text{SCK}}$ is set to 1 when serial transfer stops.
(2) The P01 output latch is controlled by using bit manipulation (operation) instructions.
Example:
To output one clock pulse of $\overline{\text{SCK}}$ by using software.

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

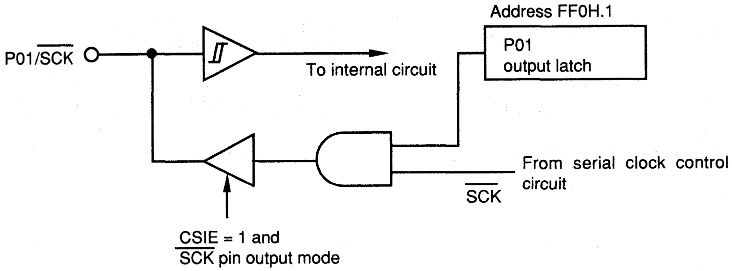


**Fig. 5.6-17 SCK/P01 Pin Configuration**

The P01 output latch is mapped in address FF0H bit 1. When the $\overline{\text{RESET}}$ signal is generated, the P01 output latch is set to 1.

Cautions:
1. The P01 output latch must be set to 1 during normal serial transfer.
2. Do not use "PORT0.1" to specify the P01 output latch address. Write directly the address (FF0H.1) in operand. At that time, set MBE to 0, or set MBE to 1 and MBS to 15.

```
CLR1  PORT0.1 ⎫
              ⎬ Do not use
SET1  PORT0.1 ⎭

CLR1  0FF0H.1 ⎫
              ⎬ Use
SET1  0FF0H.1 ⎭
```

5.6.8 Serial interface application

The serial interface for each mode is explained using examples of applications.

The normal serial interface communication sequence is as follows:

1) Set transfer mode. (Set data in CSIM.)

2) Write data into SIO and give transfer start indication. (MOV SIO, XA or XCH XA, SIO. At that time, automatic transfer start indication is given.)

3) After checking that the serial interrupt routine or interrupt request flag (IRQCSI) is set, read receive data, and start transfer.

The SBI mode communication sequence is explained in detail in (3) below.

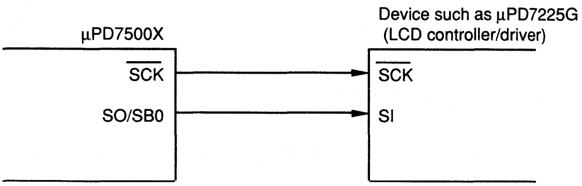
(1) 3-line serial I/O mode

(a) To transfer data starting at MBS using transfer clock of 262 kHz (at 4.19 MHz) (master operation).

```
Example:  CLR1   MBE
          MOV    XA, #10000010B
          MOV    CSIM, XA       ; Transfer mode is set.
          MOV    XA, TDATA      ; TDATA is the transfer data storage address.
          MOV    SIO, XA        ; Transfer data is set.
                                ; Transfer is started.
```

Caution:

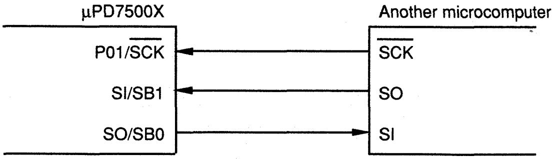
At the second time or after, transfer can be started by setting data in SIO (MOV SIO, XA or XCH XA, SIO).



In this application example, the μPD7500X SI/SB1 pin can be used for input.

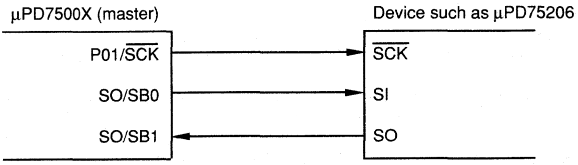
(b) To transfer data starting at the LSB using external clock (slave operation).

(In this example, the shift register read/write function to invert the LSB and MSB is used effectively.)



```
Example:  Main routine
          CLR1   MBE
          MOV    XA, #84H
          MOV    CSIM, XA       ; Serial operation stop, LSB/MSB inversion mode, external clock
          MOV    XA, TDATA
          MOV    SIO, XA        ; Transfer data is set.
                                ; Transfer is started.
          EI     IECSI
          EI

          Interrupt routine (MBE = 0)
          MOV    XA, TDATA
          XCH    XA, SIO        ; Receive data - send data, transfer start
          MOV    RDATA, XA      ; Receive data is saved.
          RETI
```

(c) To transfer data at a high speed using a transfer clock of 524 kHz (at 4.19 MHz).



Example (master)

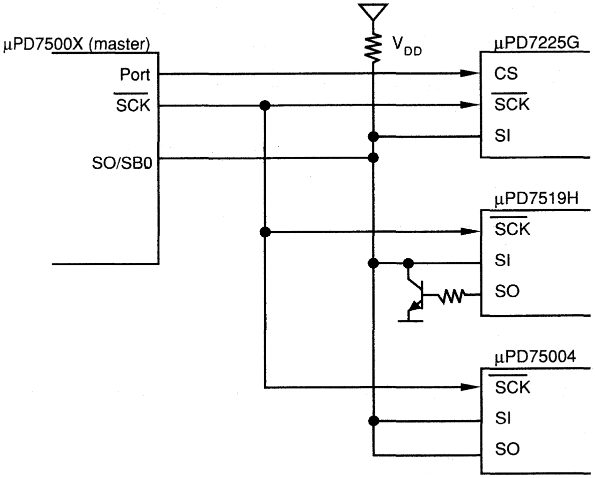
```
                CLR1    MBE
                MOV     XA, #10000011B
                MOV     CSIM, XA        ; Transfer mode is set.
                MOV     XA, TDATA
                MOV     SIO, XA         ; Transfer data is set, Transfer is started.
                 :
                 :
                 :
LOOP:           SKTCLR  IRQCSI          ; IRQCSI is tested.
                BR      LOOP
                MOV     XA, SIO         ; Receive data is read.
```

(2) 2-line serial I/O mode

The serial bus is formed and a number of devices are connected.

Example:

The example system consists of the μPD7500X as the master and μPD75004, μPD7519H, and μPD7225G connected as slaves.



To form a serial bus as in this example, FFH is previously written into the shift register, a high level is output to the shift register, a high level is output to the SO pin, the output buffers are turned off, and the bus is released except when the SI and SO pins are connected for serial data output.

Since the μPD7519H SO pin cannot be placed in the high impedance state, a transistor is connected for open collector output, as shown in the diagram above. At the time of data input, the transistor is turned off by previously writing 00H into the shift register.

The data output timing by the microcomputers is predetermined.

The serial clock is output by the master microcomputer μPD7500X; all other slave microcomputers operate on external clocks.

(3) SBI mode

An application example of serial data communication in the SBI mode is given. In the example, the μPD7500X can operate as a master or a slave CPU.

The master can also be changed by using a command.

(a) Serial bus configuration

The serial bus configuration in the application example given here assumes that the μPD7500X is connected to bus lines as one device in the serial bus.

The following two μPD7500X pins are used: Serial data bus SB0 (P02/SO) and serial clock $\overline{\text{SCK}}$ (P01).

Fig. 5.6-18 shows an example of a serial bus configuration.

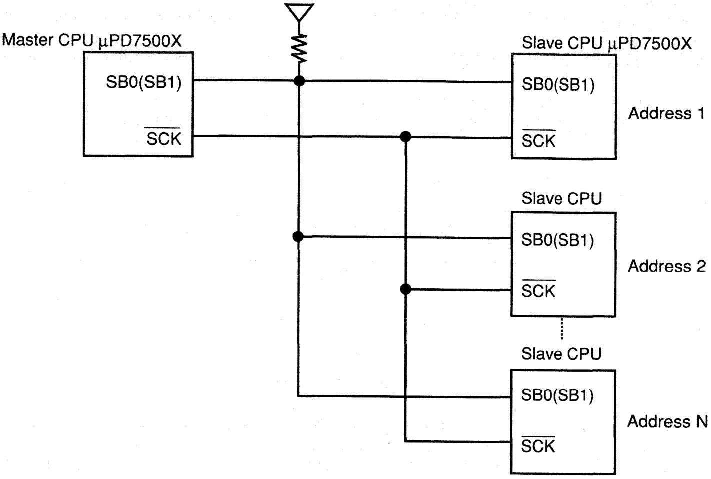


**Fig. 5.6-18 Serial Bus Configuration Example**

(b) Command explanation

Command types

The application example uses the following commands:

1) READ:
Data is transferred from slave to master.
2) WRITE:
Data is transferred from master to slave.
3) END:
WRITE command completion is reported to slave.
4) STOP:
WRITE command stop is reported to slave.
5) STATUS:
The slave state is read.
6) RESET:
The current slave being selected is made unselected.
7) CHGMST:
The master authorization is transferred to the slave.

Communication sequence

The communication sequence between the master and slave is as follows:

1) Communication is started by the master, which sends the address of the slave to communicate with and selects the slave (chip select).
The slave which receives the address returns $\overline{\text{ACK}}$ and communicates with the master. (The slave is placed in selected state.)

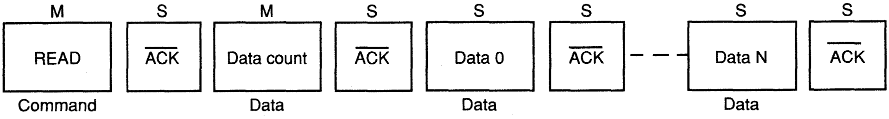
2) Command and data are transferred between the slave selected in 1 and the master.
Since command and data are transferred point-to-point (between the master and specific slave), other slaves must be deselected.

3) Communication terminates when the slave is deselected in either of the following cases:
— When the master sends the RESET command, the selected slave is deselected.
— If the master is changed by using the CHGMST command, the device changed from master to slave is deselected.

Command format

The command transfer formats are shown below:

1) READ command
The READ command reads data from a given slave. The read data count ranges from one to 256 bytes. The master specifies the data count in a parameter. If 00H is specified for the data count, 256-byte data transfer is assumed to be specified.

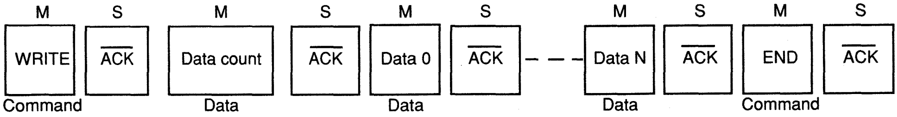


M: Output by master
S: Output by slave

**Fig. 5.6-19 READ Command Transfer Format**

After receiving the data count, the slave returns $\overline{\text{ACK}}$ if the number of data bytes that can be sent is greater than the data count. If it is less than the data count, the slave does not return $\overline{\text{ACK}}$, resulting in an error.
During data transfer, the slave compares the SIO contents before and after data transfer to check that data has been output to the bus normally. If the SIO contents before and after transfer do not match, the slave does not return $\overline{\text{ACK}}$, resulting in an error.

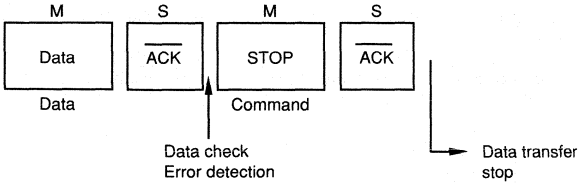
2) WRITE, END, and STOP commands
The WRITE, END, and STOP commands are used to write data into a particulary slave. The write data count ranges from 1 to 256 bytes. The master specifies the data count in a parameter. If 00H is specified for the data count, 256-byte data transfer is assumed to be specified.



M: Output by master
S: Output by slave

**Fig. 5.6-20 WRITE, END Command Transfer Format**

After receiving the data count, the slave returns $\overline{\text{ACK}}$ if the receive data store area is larger than the data count. If it is less than the data count, it does not return $\overline{\text{ACK}}$, resulting in an error.
At the termination of all data transfer, the master sends the END command to the slave. It signals that all data has been transferred normally.
The slave also receives an END command before it completes all data reception. In this case, the data which has been received immediately before the END command is received becomes valid. During data sending, the master compares the SIO contents before and after data sending to check that data has been output to the bus normally. If the SO contents before and after sending do not match, the master sends the STOP command and stops data transfer.

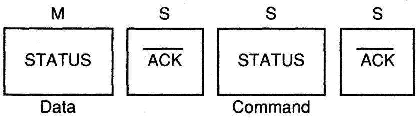


M: Output by master
S: Output by slave

**Fig. 5.6-21 STOP Command Transfer Format**

When receiving the STOP command, the slave invalidates the 1-byte data received immediately before receiving the STOP command.

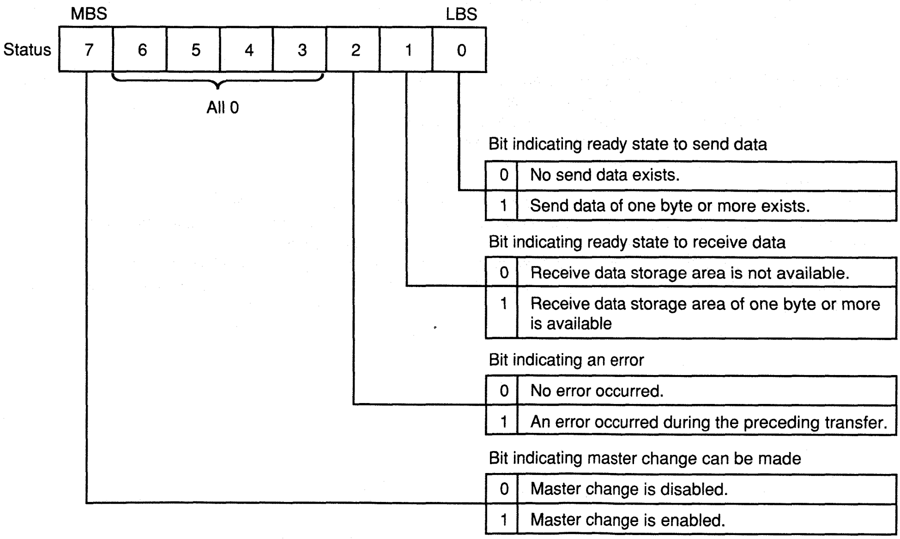
3) STATUS command
The STATUS Command Transfer Format



M: Output by master
S: Output by slave

**Fig. 5.6-22 STATUS Command Status Format**

Fig. 5.6-23 shows the format of the status returned by the slave.



Bit indicating ready state to send data

| | |
|---|---|
| 0 | No send data exists. |
| 1 | Send data of one byte or more exists. |

Bit indicating ready state to receive data

| | |
|---|---|
| 0 | Receive data storage area is not available. |
| 1 | Receive data storage area of one byte or more is available |

Bit indicating an error

| | |
|---|---|
| 0 | No error occurred. |
| 1 | An error occurred during the preceding transfer. |

Bit indicating master change can be made

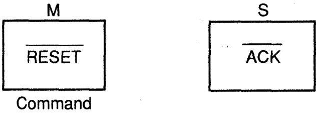
| | |
|---|---|
| 0 | Master change is disabled. |
| 1 | Master change is enabled. |

**Fig. 5.6-23 STATUS Command Status Format**

To send status data, the slave compares the contents before and after sending status data. If they do not match, the slave does not return ACK, resulting in an error.

4) $\overline{\text{RESET}}$ command

The RESET command is used to cause the currently selected slave to be selected. When the $\overline{\text{RESET}}$ command is issued, all slaves can be deselected.

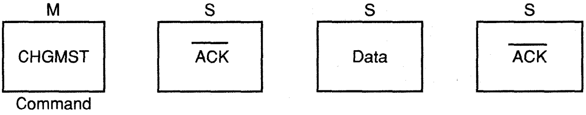


M: Output by master
S: Output ba slave

**Fig. 5.6-24 RESET Command Transfer Format**

5) CHGMST command

The CHGMST command transfers the master authorization to the currently selected slave.



M: Output by master
S: Output by slave

**Fig. 5.6-25 CHGMST Command Transfer Format**

When receiving the CHGMST command, the slave decides whether or not it can receive the master authorization, and returns either of the following data to the master:
— 0FFH: Master change is enabled.
— 00H : Master change is disabled.
When transferring data, the slave compares the SIO contents before and after data transfer. If they do not match, the slave does not return $\overline{\text{ACK}}$, resulting in an error.
If no error occurs, the master serves as a slave after 0FFH data sending is complete. If no error occurs, the slave serves as the master after 0FFH data sending is completed.

Error occurrence

When a communication error occurs, the master and slave operate as explained below:
The slave informs the master of error occurrence by returning no $\overline{\text{ACK}}$. When an error occurs, the status bit (bit 2) indicating error occurence is set to 1 and all command processing being performed is cancelled.
After completing the sending or receiving of one byte, the master checks whether or not $\overline{\text{ACK}}$ is returned from slave. If $\overline{\text{ACK}}$ is not returned from slave within a given period after sending or receiving is completed, the master decides that an error has occurred and outputs a dummy $\overline{\text{ACK}}$ signal.

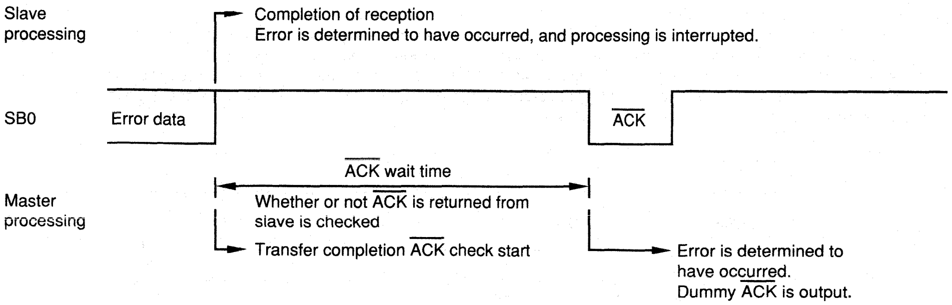

Slave processing
Completion of reception
Error is determined to have occurred, and processing is interrupted.
SB0
Error data
$\overline{\text{ACK}}$
$\overline{\text{ACK}}$ wait time
Master processing
Whether or not $\overline{\text{ACK}}$ is returned from slave is checked
Transfer completion $\overline{\text{ACK}}$ check start
Error is determined to have occurred.
Dummy $\overline{\text{ACK}}$ is output.

**Fig. 5.6-26 Master and Slave Operation when Error Occured**

The following errors are possible:

— Errors that may occur in the slave

1) Command transfer format is erroneous.
2) Undefined command is received.
3) The number of data bytes to be transferred (data count) is insufficient during READ command execution,
4) Data storage area is insufficient during WRITE command execution.
5) When READ, STATUS, or CHGMST command data is sent, data changes.

When any error in 1) or 5) occurs, no ACK is returned.

— Errors that may occur in the master

When WRITE command data is sent, if data changes, STOP command is sent to the slave.

### 5.7 Bit Sequential Buffer – 16 Bits

The bit sequential buffer is a special data memory for bit manipulation. Bit manipulation can be easily performed by changing address and bit specification in sequence. The buffer is useful for bitwise processing of long data.
The data memory consists of 16 bits. pmem.@L addressing of bit manipulation is enabled; indirect bit specification can be made by using the L register . Simply by incrementing or decrementing the L register in a program loop, processing can be performed while the specified bits are being moved in sequence.

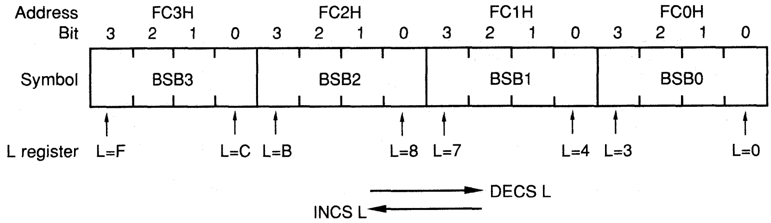


**Fig. 5.7-1 Bit Sequential Buffer Format**

Remarks:
In pmem.@L addressing, specified bits are moved according to the L register.

Data can also be handled in direct addressing. The 1–, 4–, and 8-bit direct addressing modes and pmem.@L addressing mode can be used in combination for continuous input and output of 1-bit data. In 8-bit manipulation, BSB0 and BSB2 are specified, and data is handled every high-order eight and low-order eight bits.

Example:
To output 16-bit data in BUFF 1 and BUFF2 from port 3 bit 0 serially.

```
        CLR1   MBE
        MOV    XA, BUFF1
        MOV    BSB0, XA    ; BSB0 and BSB1 are set.
        MOV    XA, BUFF 2
        MOV    BSB2, XA    ; BSB2 and BSB3 are set.
        MOV    L, #0
LOOP:   SKT    BSB0, @L    ; BSB-specified bit is tested.
        BR     LOOP1
        NOP                ; Dummy (timing adjustment)
        SET1   PORT3.0     ; Port 3 bit 0 is set.
        BR     LOOP2
LOOP1:  CLR1   PORT3.0     ; Port 3 bit 0 is cleared.
        NOP                ; Dummy (timing adjustment)
        NOP
LOOP2:  INCS   L           ; L← L+1
        BR     LOOP
               RET
```

## CHAPTER 6 INTERRUPT FUNCTION

The μPD7500X contains six vectored interrupt sources and two testable inputs for versatile application.
The μPD7500X interrupt control circuit has the following features to enable very high-speed interrupt service:

(a) Whether or not interrupts can be acknowledged can be controlled by using enable flag (IEXXX).
(b) The interrupt service start address and MBE during interrupt service can be set as desired by using a vector table. Starting the actual interrupt service program is fast.
(c) Interrupt request flag (IRQXXX) can be tested and cleared. Interrupt occurrence can be checked by using software.
(d) The standby mode (STOP or HALT) can be released by making an interrupt request. A backup release source is available by using the interrupt enable flag.

### 6.1 Interrupt Control Circuit Configuration

Fig. 6.1-1 shows configuration of the interrupt control circuit. The hardware devices are mapped in data memory.

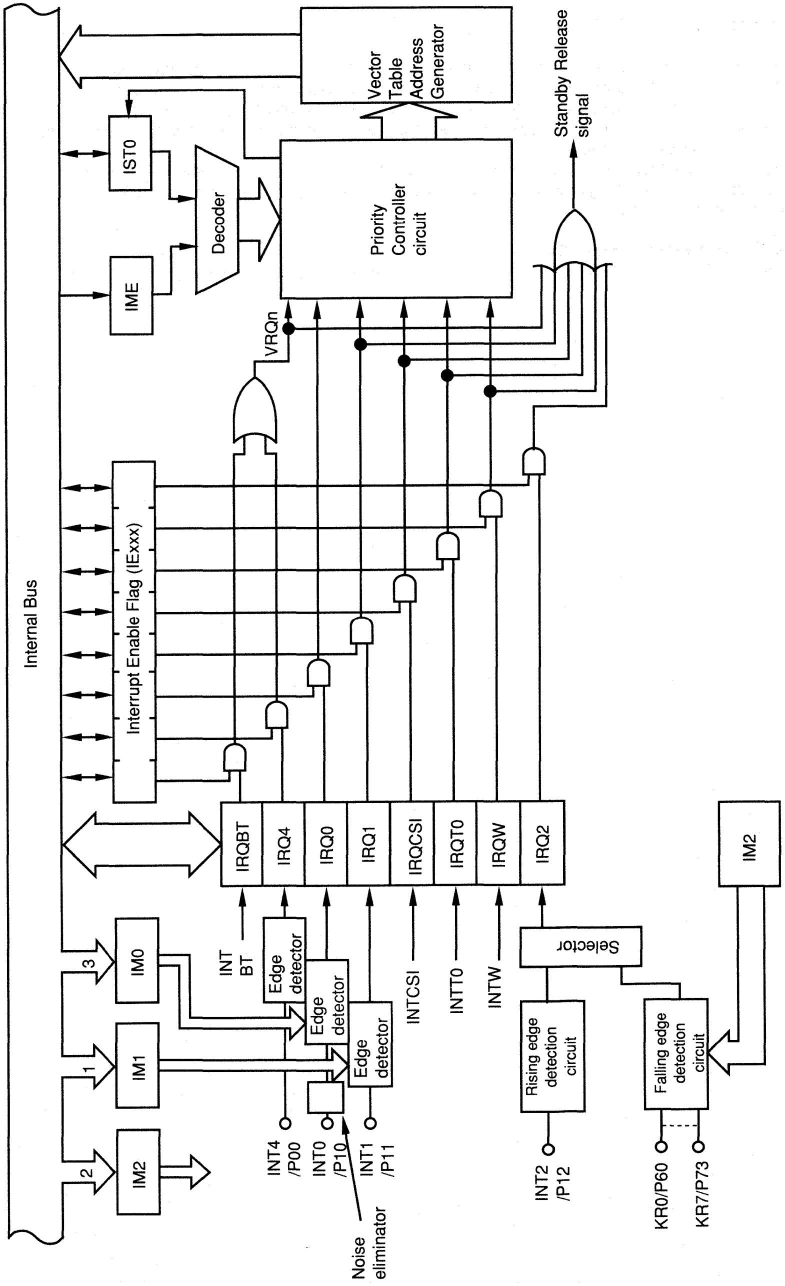


Fig. 6.1-1 Interrupt Control Circuit Block Diagram

## 6.2 Interrupt Source Types and Vector Table

Table 6.2-1 lists the types of interrupt sources for the μPD7500X. Fig. 6.2-1 shows the interrupt vector table.

**Table 6.2-1 Interrupt Source Types**

| Interrupt source | Internal /external | Interrupt priority (Note 1) | Vectored interrupt request signal (vector table address) |
|---|---|---|---|
| INTBT<br>(reference time interval signal from basic interval timer) | Internal | 1 | VRQ1 (0002H) |
| INT4<br>(both rising and falling edge detection) | External | | |
| INT0 (selection of rising or falling edge detection) | External | 2 | VRQ2 (0004H) |
| INT1 | External | 3 | VRQ3 (0006H) |
| INTCSI<br>(serial data transfer end signal) | Internal | 4 | VRQ4 (0008H) |
| INTT0<br>(coincidence signal between programmable timer/ counter count register and modulo register) | Internal | 5 | VRQ5 (000AH) |
| INT2<br>(rising edge detection of input to INT2 pin or falling edge detection of any input to KR0-KR7) (Note 2) | External | Testable input signals (IRQ2 and IRQW are set) | |
| INTW<br>(signal from watch timer) | Internal | | |

Note 1: The interrupt priority indicates the priority given to each interrupt when more than one interrupt request occurs at the same time.

Note 2: For details of INT2, see 6.3 (2).

| Address | | | | | |
|---|---|---|---|---|---|
| 002H | MBE | 0 | 0 | 0 | INTBT/INT4 start address (high-order four bits) |
| | | | | | INTBT/INT4 start address (low-order eight bits) |
| 004H | MBE | 0 | 0 | 0 | INT0 start address (high-order four bits) |
| | | | | | INT0 start address (low-order eight bits) |
| 006H | MBE | 0 | 0 | 0 | INT1 start address (high-order four bits) |
| | | | | | INT1 start address (low-order eight bits) |
| 008H | MBE | 0 | 0 | 0 | INTCSI start address (high-order four bits) |
| | | | | | INTCSI start address (low-order eight bits) |
| 00AH | MBE | 0 | 0 | 0 | INTT0 start address (high-order four bits) |
| | | | | | INTT0 start address (low-order eight bits) |

**Fig. 6.2-1 Interrupt Vector Table (a) μPD75004**

| Address | | | | |
|---|---|---|---|---|
| 002H | MBE | 0 | 0 | INTBT/INT4 start address (high-order five bits) |
| | | | | INTBT/INT4 start address (low-order eight bits) |
| 004H | MBE | 0 | 0 | INT0 start address (high-order five bits) |
| | | | | INT0 start address (low-order eight bits) |
| 006H | MBE | 0 | 0 | INT1 start address (high-order five bits) |
| | | | | INT1 start address (low-order eight bits) |
| 008H | MBE | 0 | 0 | INTCSI start address (high-order five bits) |
| | | | | INTCSI start address (low-order eight bits) |
| 00AH | MBE | 0 | 0 | INTT0 start address (high-order five bits) |
| | | | | INTT0 start address (low-order eight bits) |

**Fig. 6.2-1 Interrupt Vector Table (b) μPD75006, μPD75008**

The interrupt priority in Table 6.2-1 indicates the interrupt execution order when a number of interrupt requests occur at the same time or are held.
The interrupt service start addresses and MBE setup values during interrupt service are written into the vector table. The vector table is set by using an assembler pseudo-instruction (VENTn).

Example: To set INTBT/INT4 in vector table

```
VENT1        MBE=0, RBE=0,        GOTOBT
  ↑            ↑      ↑             ↑
  ①            ②      ③             ④
```

① Vector table address 0002.

② MBE setup value in interrupt service routine.
③ Always set RBE to 0.

④ This symbol denotes the interrupt service routine start address.

Caution 1: The vector table address specified in VENTn (n=1-5) becomes 2n address.
Caution 2: Always set RBE to 0 in VENTn.

Example: To set INTBT/INT4 and INTT0 in vector table.

```
VENT1          MBE=0, RBE=0, GOTOBT
VENT5          MBE=0, RBE=0, GOTOT0
```

### 6.3 Hardware Devices of Interrupt Control Circuit

(1) Interrupt request and enable flags

The interrupt request flag (IRQXXX) is set to 1 when a given interrupt request occurs; it is automatically cleared when interrupt service is executed.
Although IRQBT and INT4 share the vector address, they differ in clear operation. (See 6.6.)
An interrupt enable flag (IEXXX) is provided for each interrupt request flag. An interrupt is enabled when the corresponding interrupt enable flag is set to 1 and disabled when the flag is set to 0.
When the interrupt request flag is set, and the interrupt enable flag enables an interrupt, a vectored interrupt request (VRQn) occurs. This signal is also used to release the standby mode.
The interrupt request and enable flags are handled by using the bit manipulation and 4-bit memory operation instructions. The flags can be directly handled by using the bit manipulation instruction regardless of how MBE is set. The interrupt enable flags are handled by using the EI IEXXX and DI IEXXX instructions. Normally, the interrupt request flags are tested by using the SKTCLR instruction.

```
Example:  EI        IE0       ; INT0 is enabled.
          DI        IE1       ; INT1 is disabled.
          SKTCLR    IRQCSI    ; If IRQCSI is set to 1, skip and clear.
```

If the interrupt request flag is set by using the instruction, a vectored interrupt is executed as if an interrupt occured although it did not actually occurs.
When the $\overline{\text{RESET}}$ signal is generated, the interrupt request and enable flags are cleared, and all interrupts are disabled.

**Table 6.3-1 Interrupt Request and Enable Flags**

| Interrupt request flag | Interrupt request flag set signal | Interrupt enable flag |
|---|---|---|
| IRQBT | Is set by reference time interval signal generated by the basic interval timer | IEBT |
| IRQ4 | Is set when either the rising or falling edge of the INT4/P00 pin input signal is detected. | IE4 |
| IRQ0 | Is set when the INT0/P10 pin input signal edge is detected. Detected edge is selected among rising and falling edges by using the INT0 mode register (IM0). | IE0 |
| IRQ1 | Is set when the edge of the INT1/P11 pin input signal is detected. Detected edge is selected from rising and falling edges by using the INT1 mode register (IM1). | IE1 |
| IRQCSI | Is set by serial data transfer end signal on serial interface. | IECSI |
| IRQT0 | Is set by coincidence signal from timer/event counter 0. | IET0 |

**Table 6.3-1 Interrupt Request and Enable Flags (cont'd)**

| Interrupt request flag | Interrupt request flag set signal | Interrupt enable flag |
|---|---|---|
| IRQW | Is set by signal from the watch timer. | IEW |
| IRQ2 | Is set when the rising edge of the INT2/P12 pin input signal is detected or the falling edge of any input to the KR0/P60 to KR7/P73 pins is detected. | IE2 |

(2) Noise eliminator and edge detection mode registers.

Figs. 6.3-1 and 6.3-2 show the relationship of INT0 and INT1, INT2 and KR0-KR7 pins, respectivvely, within the interrupt control circuit. They are used for inputting external interrupts in cases where noise can be eliminated by the sampling clock and the detected edge can be selected.

Any pulse narrower than the sampling clock is determined to be a noise pulse and is eliminated by the noise eliminator. A pulse twice as wide (or more) as the sampling clock is acknowledged as a valid interrupt signal.

For INT0 the sampling clock can be changed to two stages.

Note that when the INT0 pin is used for P10 (port pin), the signal is also input via the noise eliminator.

Caution: Since INT0 sampling uses clock, INT0 does not operate at standby mode.

IRQ2 is set in either of the following modes:

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

(b) When the falling edge of any input to KR0-KR7 pins is detected:

When the falling edge of any input to pins selected among KR0-KR7 by using the edge detection mode register (IM2) is detected, IRQ2 is set.

For example, if KR4-KR7 are selected, IRQ2 is set if the falling edge is input to any one of KR4-KR7, because KR4-KR7 inputs are ANDed in the μPD7500X.

Fig. 6.3-4 shows the format of the edge detection mode registers used to select detected edge (IM0 to IM2). IM0 to IM2 are set individually by using a 4-bit memory operation instruction.

When RESET signal is generated, all the register bits are cleared; INT0, INT1, and INT2 detection edges are specified to rising edges.

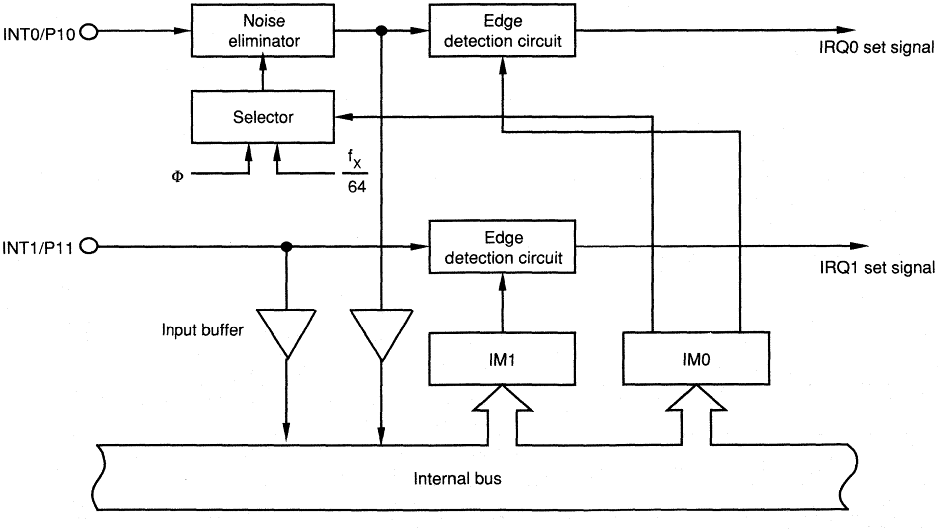


**Fig. 6.3-1 INT0 and INT1 Configuration**

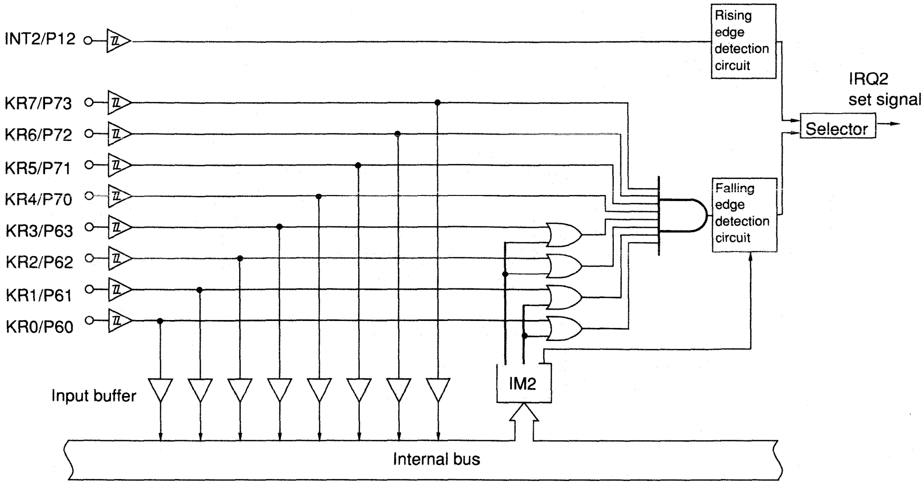


**Fig. 6.3-2 INT2 and KR0-KR7 Configuration**

When a high level is input successively, the noise eliminator outputs a high level; when a low level is input successively, it outputs a low level.

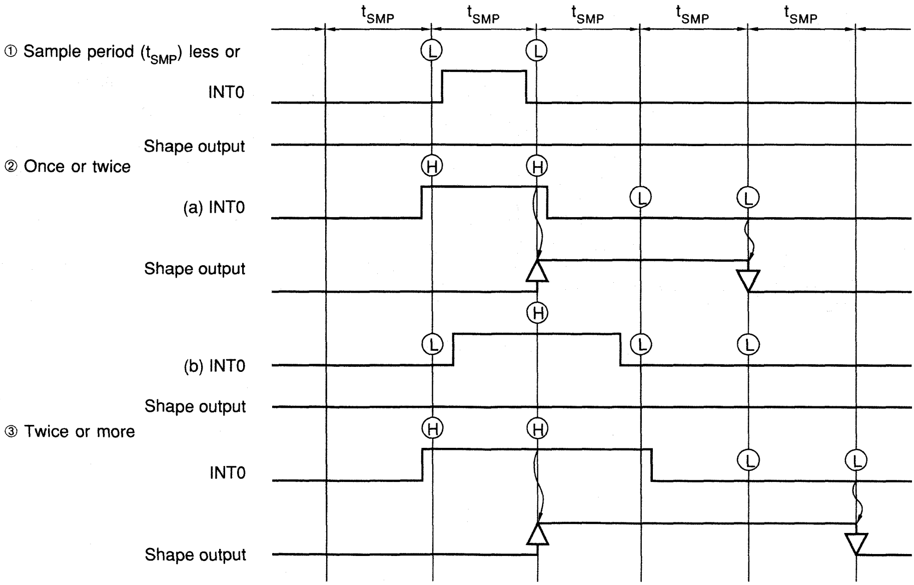
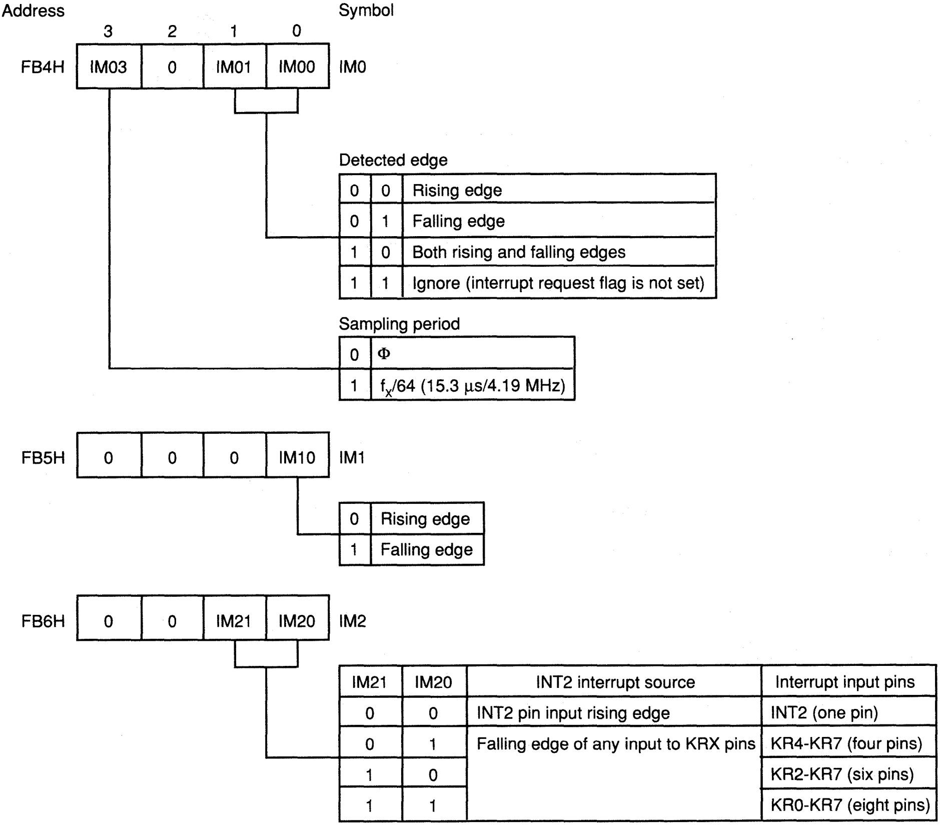


**Fig. 6.3-3 Noise Eliminator Input/Output Timing**



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

Detected edge

|  |  |  |
|---|---|---|
| 0 | 0 | Rising edge |
| 0 | 1 | Falling edge |
| 1 | 0 | Both rising and falling edges |
| 1 | 1 | Ignore (interrupt request flag is not set) |

Sampling period

|  |  |
|---|---|
| 0 | Φ |
| 1 | $f_X$/64 (15.3 μs/4.19 MHz) |

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| FB5H | 0 | 0 | 0 | IM10 | IM1 |

|  |  |
|---|---|
| 0 | Rising edge |
| 1 | Falling edge |

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| FB6H | 0 | 0 | IM21 | IM20 | IM2 |

| IM21 | IM20 | INT2 interrupt source | Interrupt input pins |
|---|---|---|---|
| 0 | 0 | INT2 pin input rising edge | INT2 (one pin) |
| 0 | 1 | Falling edge of any input to KRX pins | KR4-KR7 (four pins) |
| 1 | 0 |  | KR2-KR7 (six pins) |
| 1 | 1 |  | KR0-KR7 (eight pins) |

Caution: If the edge detection mode register is changed, the interrupt request flag may be set. Disable interrupts beforehand, and change the mode register. Clear the interrupt request flag by using the CLR1 instruction before enabling interrupts. If $f_X$/64 is selected for the sampling clock in an IM0 change, clear the interrupt request flag within 16 machine cycles after the mode register is changed.

**Fig. 6.3-4 Edge Detection Mode Register Format**

(3) Interrupt master enable flag (IME)

Interrupt master enable flag enables or disables acknowledgement of all interrupts.
IME is set to 1 or to 0 by using the EI and DI instructions.
When the $\overline{\text{RESET}}$ signal is generated, IME is cleared, disabling acknowledgement of all interrupts.

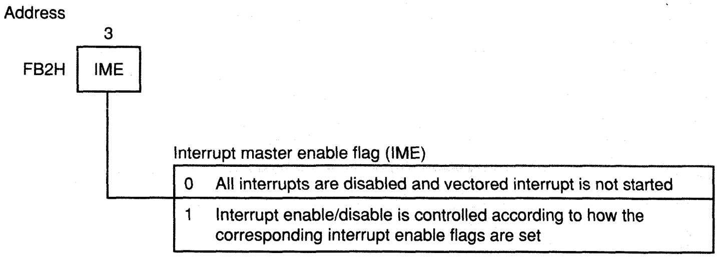

Address
3
FB2H
IME
Interrupt master enable flag (IME)
0 All interrupts are disabled and vectored interrupt is not started
1 Interrupt enable/disable is controlled according to how the corresponding interrupt enable flags are set

**Fig. 6.3-5 IME Format**

(4) Interrupt status flag

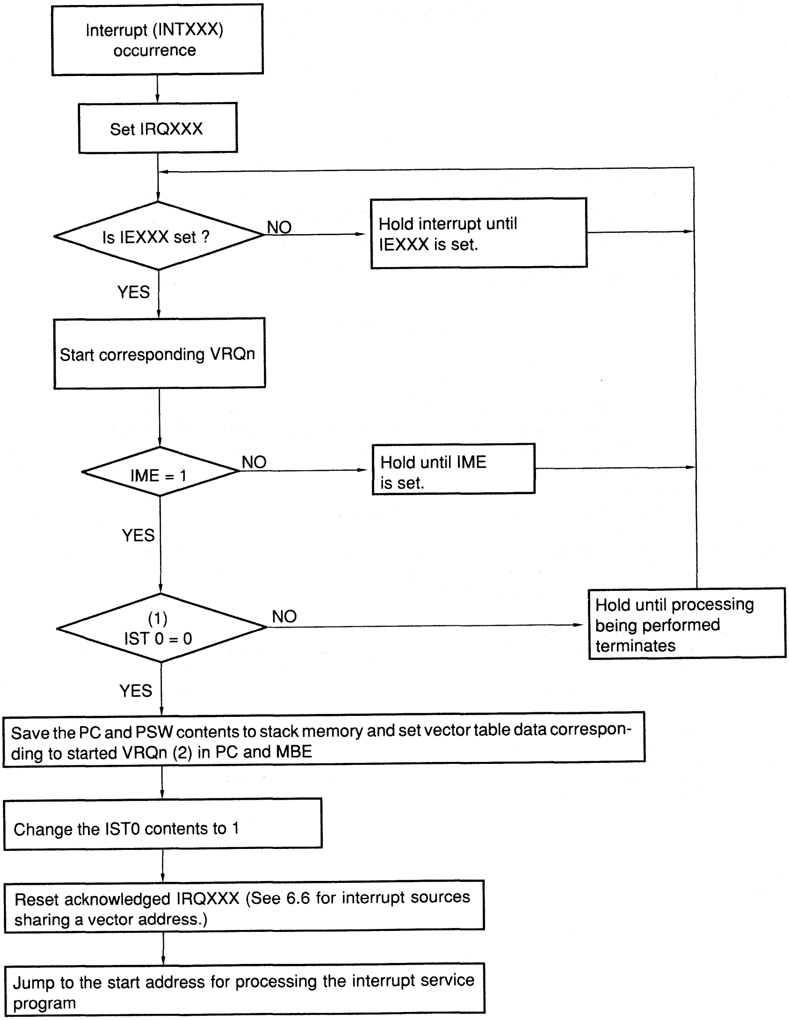
The interrupt status flag (IST0) indicates the current status of processing being performed by CPU; it is contained in the PSW. The interrupt control circuit controls multi-interrupts (as listed in Table 6.3-2) according to the contents of interrupt status flag. Since IST0 can be changed by using a 4-bit or a one-bit manipulation instruction, multi-interrupt can also be implemented by changing the status of processing being performed. Multi-interrupt can always be implemented regardless of the MBE setting when IST0 is handled bitwise.
Be sure to execute the DI instruction to disable interrupts before handling IST0 and the EI instruction to enable interrupts after handling IST0.
When an interrupt is acknowledged, IST0 is saved in stack memory together with other PSW bits, then it is automatically set to 1. When the RETI instruction is executed, the former IST0 value (0) is restored.
When the $\overline{\text{RESET}}$ signal is generated, the flag is cleared.

**Table 6.3-2 IST0 and Interrupt Service State**

| IST0 | Status of processing | CPU processing contents | Interrupt requests that can be acknowledged | After interrupt is acknowledged |
|---|---|---|---|---|
| | | | | IST0 |
| 0 | Status 0 | During normal program processing | All interrupts can be acknowledged. | 1 |
| 1 | Status 1 | During interrupt service | None of the interrupts can be acknowledged. | – |

### 6.4 Interrupt Sequence

When an interrupt occurs, it is processed as shown in Fig. 6.4-1.


Interrupt (INTXXX) occurrence
Set IRQXXX
Is IEXXX set ?
NO
Hold interrupt until IEXXX is set.
YES
Start corresponding VRQn
IME = 1
NO
Hold until IME is set.
YES
(1) IST 0 = 0
NO
Hold until processing being performed terminates
YES
Save the PC and PSW contents to stack memory and set vector table data correspon-ding to started VRQn (2) in PC and MBE
Change the IST0 contents to 1
Reset acknowledged IRQXXX (See 6.6 for interrupt sources sharing a vector address.)
Jump to the start address for processing the interrupt service program

Remarks: 1. IST0: Interrupt status flag (PSW bit 2). (See Table 6.3-2)
2. Store the interrupt service program start addresses and MBE setup values at the interrupt start in the vector table.

**Fig. 6.4-1 Interrupt Service Flow**

### 6.5 Multi-interrupt Service Control

The μPD7500X enables multi-interrupts as described below.

As understood from Table 6.3-2, multi-interrupt is enabled if the interrupts status flag is changed by a program. That is, multi-interrupt is enabled if IST0 is changed to 0 by the interrupt service program setting status 0.

To change IST0, execute the DI instruction beforehand to disable interrupts.

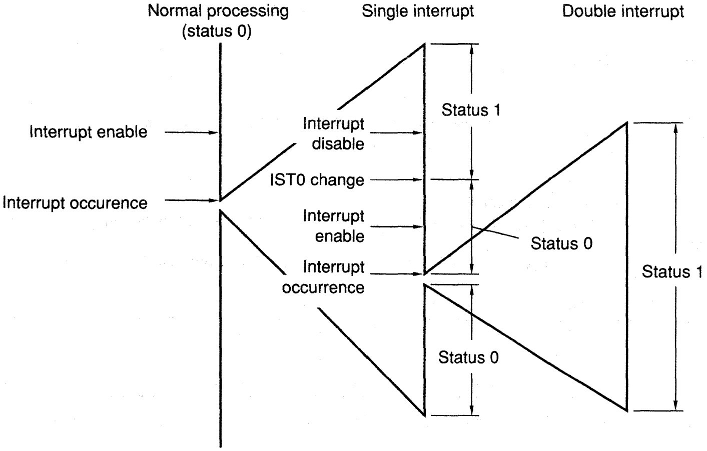


**Fig. 6.5-1 Multi-Interrupt by Changing Interrupt Status Flag**

### 6.6 Vector Address Share Interrupt Service

Since INTBT and INT4 interrupt sources share the vector table addresses, interrupt source selection is made as described below:

(1) To use one interrupt only

Set the interrupt enable flag to 1 for the required one of the two interrupt sources sharing the vector table addresses, and clear the interrupt enable flag of the other interrupt source.

In this case, an interrupt request is generated from the interrupt source corresponding to the interrupt enable flag that is set to 1 (IEXXX = 1). When the interrupt request is acknowledged, the interrupt request flag is cleared.

(2) To use both interrupts

Set both the interrupt enable flags of the two interrupt sources to 1. In this case, an interrupt request is made by ORing the interrupt request flags of the two interrupt sources.

Even if an interrupt request is acknowledged when either or both of the interrupt request flags are set to 1, the interrupt request flags are not reset.

Therefore, the interrupt service routine must decide which interrupt source the interrupt is generated from. This is accomplished by executing the SKTCLR instruction to check the interrupt request flags.

If both the interrupt request flags are set to 1 when the request flags are tested and cleared by execution of the SKTCLR instruction, the interrupt request is left even if one request flag is cleared. If IST0 is cleared, dual interrupt service is entered according to the left interrupt request.

Remarks: When only one interrupt is enabled, the source of an interrupt is known. Thus, the interrupt request flag is cleared by hardware when interrupt is acknowledged. When both interrupts are enabled, the source of an interrupt is not defined; thus, interrupt request flag cannot be cleared by hardware. Software is used to check the interrupt request flags and determine the interrupt source. Interrupt request flag is cleared by software.

Example: When INT4 takes precedence over INTBT

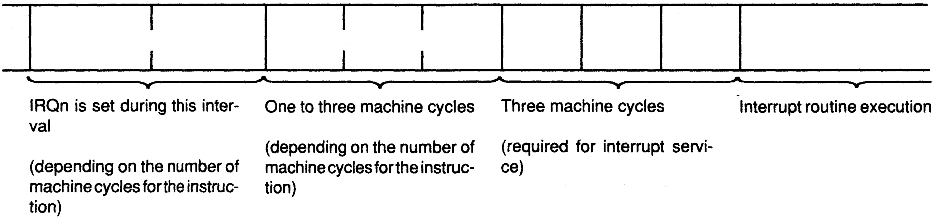
```
           DI
           SKTCLR     IRQ4         ; IRQ4 = 1 ?
           BR         VSUBBT    ┐
            :                   ├ INT4 interrupt
           EI                   │ service routine
           RETI                 ┘
            :
            :
VSUBBT:    CLR1       IRQBT
            :                   ┐
            :                   ├ INTBT interrupt
            :                   │ service routine
           EI                   ┘
           RETI
```

**6.7 Machine Cycles to Interrupt Service Start**

The number of the μPD7500X machine cycles required to start execution of the interrupt routine after an interrupt request flag is set is as shown below:

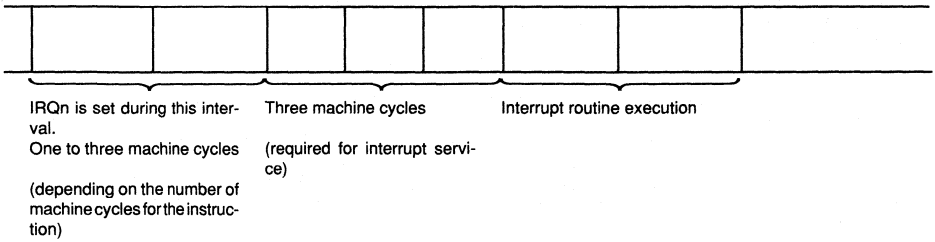
(1) When IRQn is set during execution of an operating instruction at data memory address FBxH (interrupt hardware))

FBxH address operating instruction


IRQn is set during this interval
(depending on the number of machine cycles for the instruction)
One to three machine cycles
(depending on the number of machine cycles for the instruction)
Three machine cycles
(required for interrupt service)
Interrupt routine execution

As shown above, interrupt routine processing is started a maximum of six machine cycles after the data memory (address FBxH) operating instruction is terminated. (Within a maximum of six machine cycles after the last operating instruction is terminated if the FBxH address operating instructions are successive.)

(2) When IRQn is set during execution of instruction other than in (1).


IRQn is set during this interval.
One to three machine cycles
(depending on the number of machine cycles for the instruction)
Three machine cycles
(required for interrupt service)
Interrupt routine execution

In this case, a maximum of six machine cycles are required.

**6.8 Effective Use of Interrupts**

Use the interrupt function as described below:

(1) Set MBE = 0 in the interrupt service routine.

If the data memory area used in the interrupt service routine is preferentially allocated to addresses 0-7FH and MBE = 0 is set in the interrupt vector table, a program can be prepared without considering memory banks.

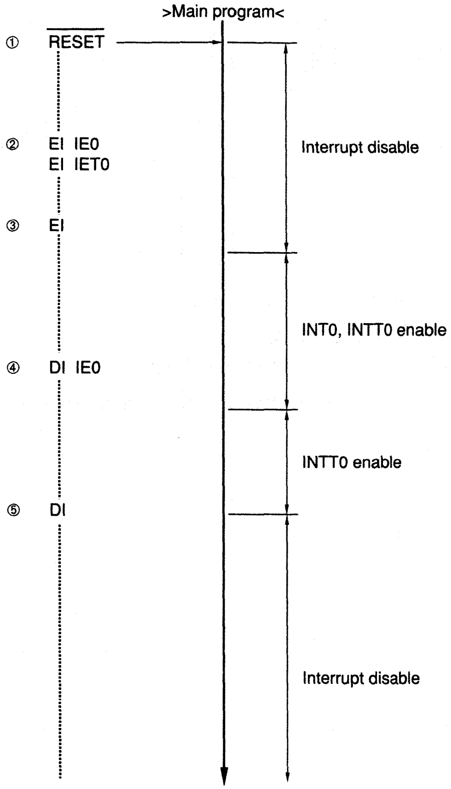
If memory bank 1 must be used for by the program, save the memory bank select register by using the PUSH BS instruction and select memory bank 1.

(2) Use software interrupt for debugging.
When an interrupt request flag is set by an instruction, operating is performed in the same manner as if an interrupt occured. Debugging when more than one interrupt occurred at the same time can be done effeciently by using an instruction to set an interrupt request flag.

### 6.9 Interrupt Application

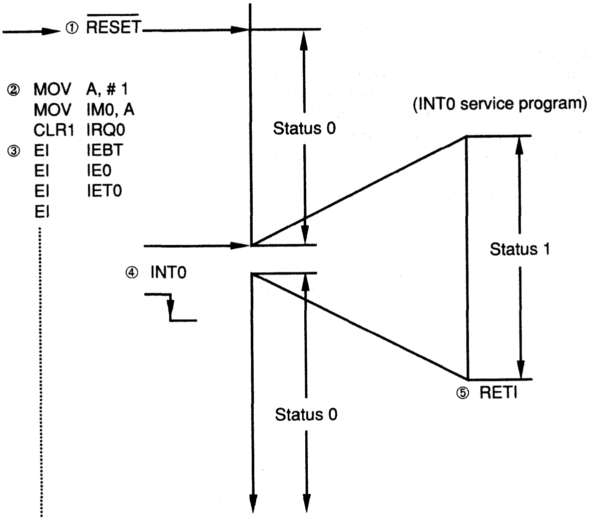
To use the interrupt function, first set the following in the main program:
(1) Set the interrupt enable flags to be used (EI IEXXX instructions).
(2) To use INT0 and INT1, select the active edge. (set IM0 and IM1.)
(3) Set the interrupt master enable flag (EI instruction).
Since MBE is set by using the vector table in an interrupt service program, registers need not be saved or restored, and the interrupt program can be started immediately.
To return from the interrupt service program, use the RETI instruction.

(1) Interrupt enable and disable



① All interrupts are disabled by the $\overline{\text{RESET}}$ signal.
② The interrupt enable flags are set by the EI IEXXX instructions. At this stage, all interrupts remain disabled.
③ The interrupt master enable flag is set by the EI instruction. At this stage, INT0 and INTT0 are enabled.
④ The interrupt enable flag is cleared by the DI IEXXX instruction and INT0 is disabled.
⑤ All interrupts are disabled by the DI instruction.

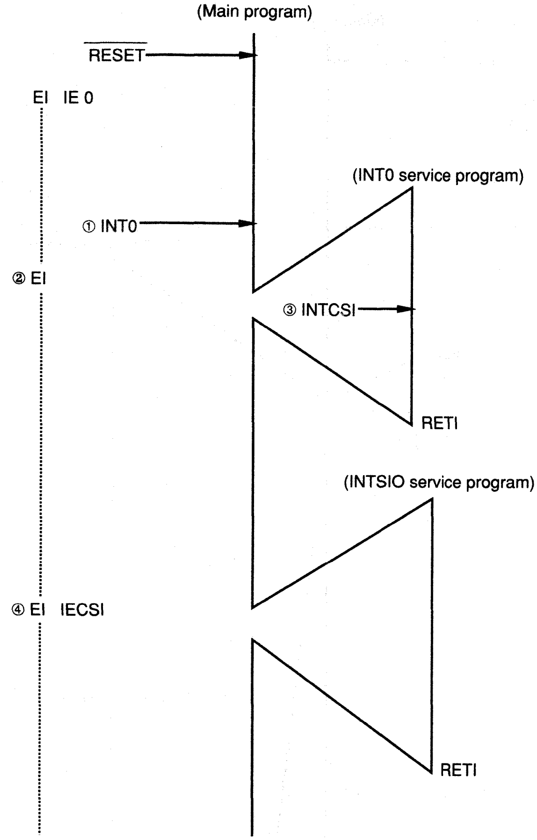
(2) Usage example of INTBT, INT0 (falling edge active), and INTT0. Multi-interrupt is not done.



```
① RESET
② MOV  A, # 1
   MOV  IM0, A
   CLR1 IRQ0
③ EI   IEBT
   EI   IE0
   EI   IET0
   EI
```

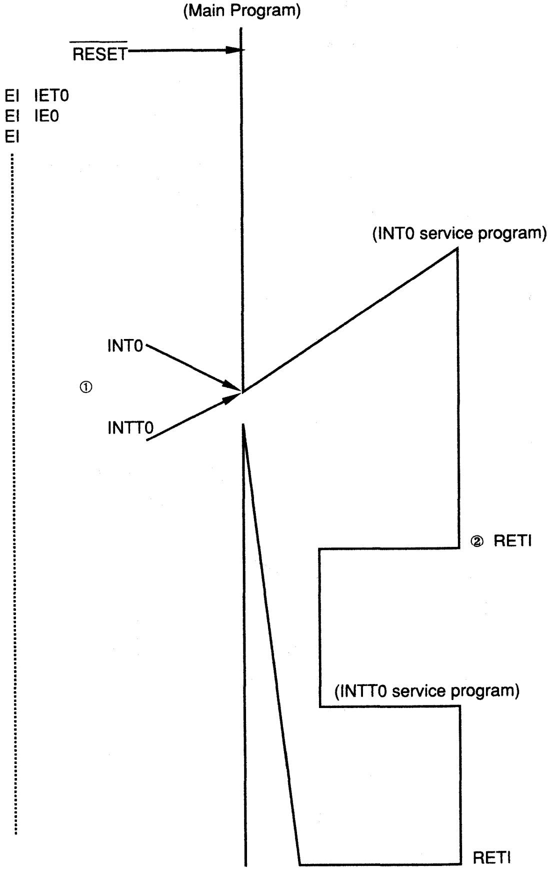
① All interrupts are disabled by the $\overline{\text{RESET}}$ signal and status 0 is set.
② Falling edge active is selected for INT0.
③ Interrupts are enabled by the EI and EI IEXXX instructions.
④ On the INT0 falling edge, the INT0 interrupt service program is started. The status is changed to status 1 and all interrupts are disabled.
⑤ A return is made from the interrupt service program by the RETI instruction. The status is restored to status 0 and interrupts are enabled.

(3) Pending interrupt execution – interrupt input during interrupt disable –



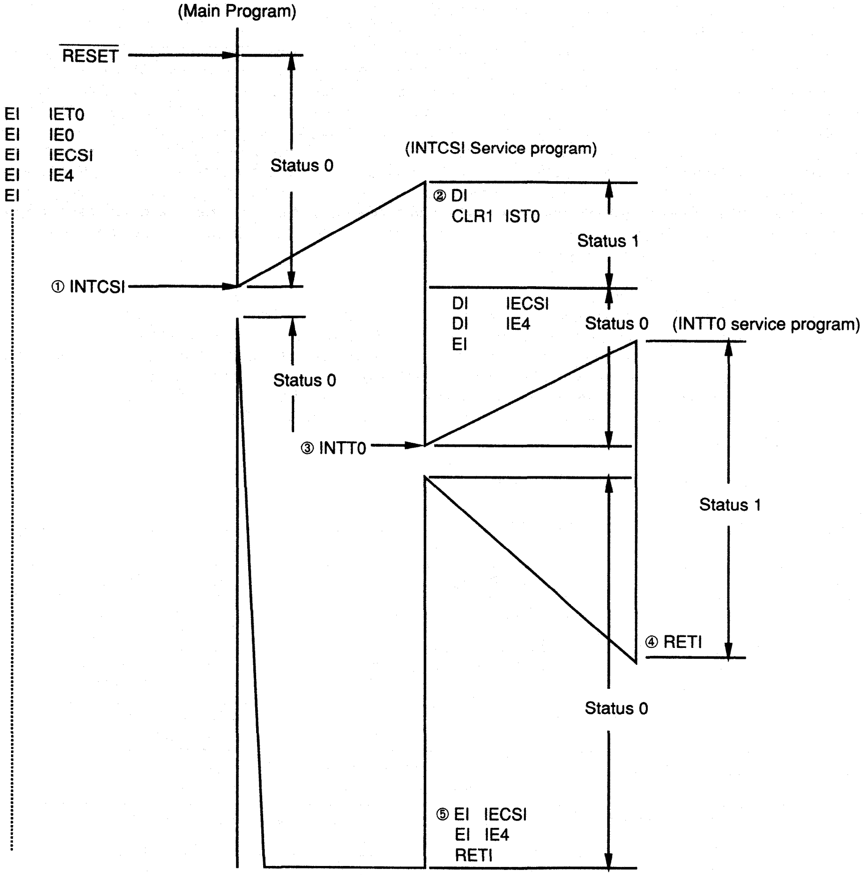
① If INT0 is set during interrupt disable, the request flag is held.
② When interrupts are enabled by the EI instruction, the INT0 service program is started.
③ Simular to ① above.
④ When INTCSI is held enabled, the INTCSI service program is started.

(4) Pending interrupt execution



① If INT0 and INTT0 are occured at the same time during execution of a single instruction, INT0, which is higher in priority than INTT0, is executed first; INTT0 is held.
② When a return is made from the INT0 service program by the RETI instruction, the held INTT0 service program is started.

(5) Enable of two double interrupts – INTT0 and INT0 enable double interrupts; INTCSI and INT4 are single interrupts –



① When interruptINTCSI occurs, which does not enable double interrupts, the INTCSI service program is started. Status 1 is set.
② The status is changed to status 0 by clearing IST0. INTCSI and INT4 do not enable double interrupt and are disabled.
③ When INTT0 enables double interrupts, the double interrupt is executed, the status is changed to status 1, and all interrupts are disabled.
④ At the termination of INTT0 service, the status is restored to status 0.
⑤ Disabled INTCSI and INT4 are enabled and a return is made by the RETI instruction.

## 7. STANDBY FUNCTION

To make the most of low current consumption, which is one feature of the CMOS process, the μPD7500X can stop CPU operation in the standby mode making current consumption by the CPU very small.
The μPD7500X standby mode includes the STOP mode and HALT mode.
The STOP mode stops the main system clock oscillator. In this mode, CPU current consumption consists almost entirely of leakage current. Data memory can also be held with low supply voltage (up to $V_{DD}$ = 2 V). This features is useful for maintaining the data memory contents with very low current consumption. Since the μPD7500X STOP mode can be released by using an interrupt request, intermittent operation can also be performed. However, if processing must be started immediately when an interrupt request is made, note that the wait time required to ensure oscillator stability is taken when the STOP mode is released.
The HALT mode continues system clock oscillator operation but stops the CPU clock ($\Phi$) supply; thus, CPU operation is stopped. Although the HALT mode is inferior to the STOP mode for reduction of current consumption, it is useful to restart processing immediately according to an interrupt request or for performing intermittent operations such as watch operation.
In either mode, all the register, flag, and data memory contents immediately before the standby mode is entered are held. The input/output port output latch state and output buffer state are also held. The input/output port state is handled beforehand so that the current consumption of the entire system is minimized.

Cautions on Use of Standby Mode:

1. The STOP mode can be used only when μPD7500X operation uses the main system clock. (Subsystem clock oscillation cannot be stopped.) The HALT mode can be used when the μPD7500X uses either main system or subsystem clock.
2. Although efficient operation with low current consumption and low voltage can be performed by using the clock change function between the CPU and system clocks in combination with the standby mode, time (described in 5.2.3) is required from selection of a new clock by setting the control register until operation is started by the newly selected clock.
   Thus, to use the clock change function and the standby mode in combination, set the standby mode within the time required for the clock change.

### 7.1 Standby Mode Setting and Operating State

**Table 7.1-1 Operating State in Standby Mode**

| | | STOP mode | HALT mode |
|---|---|---|---|
| Setting instruction | | STOP instruction | HALT instruction |
| System clock when standby mode is set. | | Can be set only during main system clock. | Can be set during either main system or subsystem clock. |
| Operating state | Clock oscillator | Only the main system clock oscillator is stopped. | Only CPU clock $\Phi$ is stopped (oscillation is continued). |
| | Basic interval timer | Operation stop | Operation (IRQBT is set at reference time intervals.) |
| | Serial interface | Can operate only when external $\overline{SCK}$ input is selected for serial clock. | Can operate. |
| | Timer/event counter | Can operate only when TI0 pin input is selected for count clock. | Can operate. |
| | Watch timer | Can operate when $f_{XT}$ is selected for count clock. | Can operate. |
| | External interrupts | INT1, INT2, and INT4 can operate. Only INT0 cannot operate. | |
| | CPU | Operation stop | |
| Release signal | | Interrupt request signal enabled with interrupt enable flag from operating hardware (exept for INT0), or $\overline{RESET}$ input. | |

The STOP mode is set by using the STOP instruction to set PCC bit 3; the HALT mode is set by the using HALT instruction to set PCC bit 2.

To change the CPU clock by using the low-order two bits of PCC, a time lag may occur from PCC rewrite to CPU clock change. Thus, to change the clock before the standby mode is entered or after the standby mode is released, set the standby mode within the number of machine cycles required to change the CPU clock after PCC is rewritten.

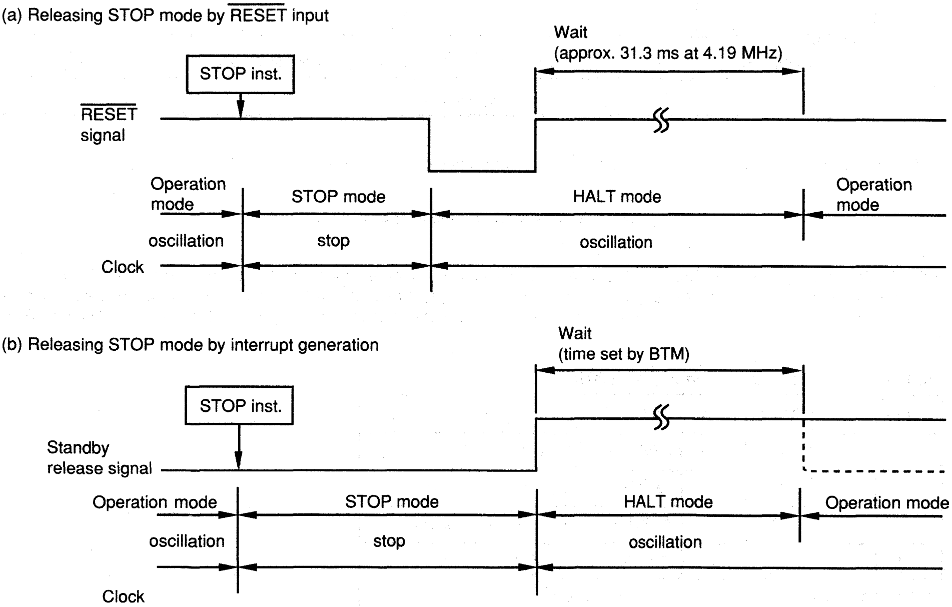
While operation stops during the standby mode, data is held in all registers and data memory such as general-purpose registers, flags, mode register, and output latches.

Caution: 1. When the STOP mode is set, X1 input is short-circuited to $V_{SS}$ (GND potential) internally to suppress crystal oscillator leakage. Therefore, so not use the STOP mode in a system using external clock as a main system slock.

2. Since an interrupt request signal is used to release the standby mode, if both interrupt request and enable flags are set for an interrupt source, the standby mode is immediately released. Thus, for the STOP mode, the HALT mode is entered immediately after execution of the STOP instruction, a wait is followed according to the setup time of the BTM register, then a return is made to the opertion mode.

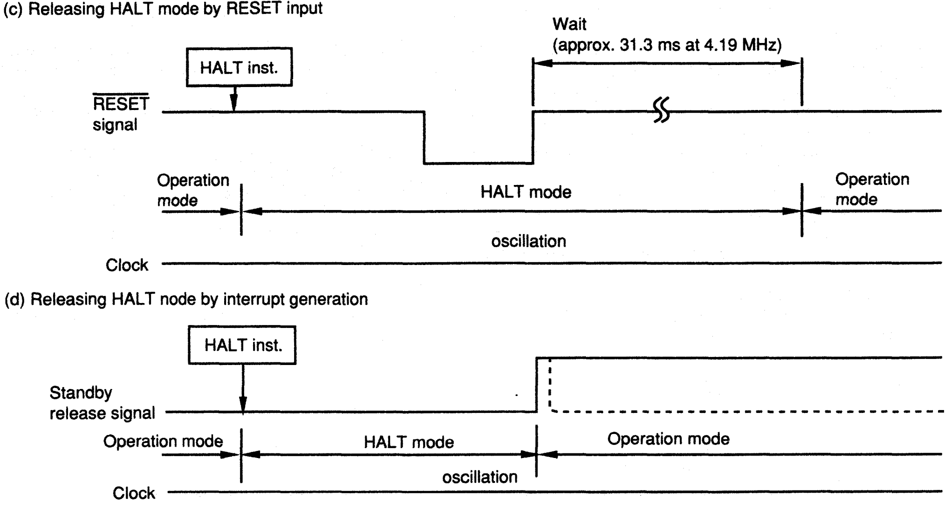
### 7.2 Realising Standby Mode

The standby mode (STOP or HALT) is released when an interrupt request signal (except INT0) enabled with an interrupt enable flag occurs or RESET is input. Fig. 6.2-1 shows the standby mode release operation.



Note: Broken line shows a case when standby releasing interrupt request is acknowledged (IME = 1).

**Fig. 7.2-1 Standby Mode Release Operation**



(c) Releasing HALT mode by $\overline{\text{RESET}}$ input

(d) Releasing HALT node by interrupt generation

Note: Broken line shows a case when standby releasing interrupt request is acknowledged (IME = 1).

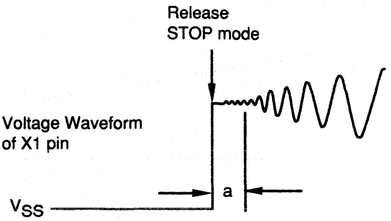
**Fig. 7.2-1 Standby Mode Release Operation (cont'd)**

If the STOP mode is released when an interrupt occurs, the wait time is determined by BTM setting. (See Table 7.2-1.)
The time until oscillation becomes stable varies according to the type of oscillator being used and the supply voltage when the STOP mode is released. Therefore, select the wait time according to the conditions of use and set BTM before setting the STOP mode.

**Table 7.2-1 Wait Time Selection by Using BTM**

| BTM3 | BTM2 | BTM1 | BTM0 | Wait time* (   ) indicates $f_{XX}$ = 4.19 MHz |
|---|---|---|---|---|
| – | 0 | 0 | 0 | APPROX. $2^{20}/f_{XX}$ (Approx. 250 ms) |
| – | 0 | 1 | 1 | APPROX. $2^{17}/f_{XX}$ (Approx. 31.3 ms) |
| – | 1 | 0 | 1 | APPROX. $2^{15}/f_{XX}$ (Approx. 7.82 ms) |
| – | 1 | 1 | 1 | APPROX. $2^{13}/f_{XX}$ (Approx. 1.95 ms) |
| Other than above | | | | Use Prohibited |

Note: The wait time when STOP mode is released does not include the time until the clock begins to oscillate after STOP mode is released ((a) in the figure below) regardless of whether STOP mode was released by $\overline{\text{RESET}}$ input or interrupt generation.



### 7.3 Operation After Standby Mode is Released

(1) If the standby mode is released when RESET is input, normal reset operation is performed.

(2) If the standby mode is released when an interrupt request occurs, the contents of the interrupt master enable flag (IME) determine whether or not a vectored interrupt is made when the CPU restarts instruction execution.

(a) When IME = 0

After the standby mode released, execution restarts at the NOP instruction next to the standby mode setting instruction. The interrupt request flags are held.

(b) When IME = 1

After the standby mode is released, two instructions following the standby mode setting instruction are executed before a vectored interrupt is executed.

However, if the standby mode is released by using INTW or INT2 (testable input), no vectored interrupt will occur; processing as in (a) above is performed.

### 7.4 Application of Standby Mode

To use the standby mode, follow the procedure described below:

1) Detect a standby mode setting source such as interrupt input or port input for power off (it is effective to use INT4 for power off detection).
2) Handle input/output ports so that current consumption is minimized.
3) Specify interrupt to release the standby mode. (It is effective to use INT4. Clear the interrupt enables flags so as not to release the standby mode.)
4) Specify operation after the standby mode is released (set IME depending on whether or not interrupt service is made).
5) Specify the CPU clock after the standby mode is released. (To change the clock, wait for the required number of machine cycles before setting the standby mode.)
6) Select the wait time in release of standby mode.
7) Set the standby mode by using the STOP or HALT instruction.

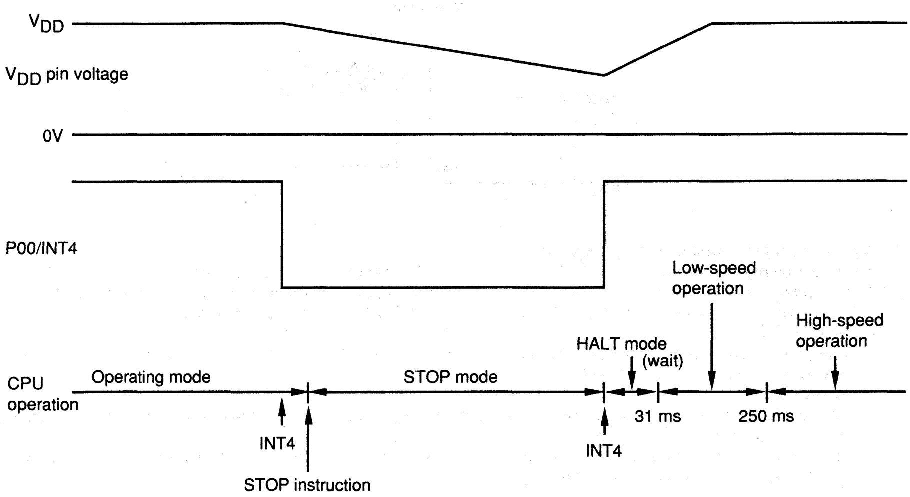
Use of the standby mode and system clock change function in combination enables the μPD7500X to operate at low current consumption and low voltage.

(1) Example of STOP mode application

Use the STOP mode under the following conditions

— Set the STOP mode when the INT4 falling edge is input and release it when the rising edge is input. (Do not use INTBT.)
— Place all input/output ports in high impedance.
— Use interrupts INT0 and INTT0 in the example program; however, do not use them to release the STOP mode.
— Enable interrupts after the STOP mode is released.
— After the STOP mode is released, start operation on the minimum speed CPU clock, and in 250 ms, change it to high-speed clock.
— Set the wait time for STOP mode release to about 31.3 ms.
— After the STOP mode is released, wait for 31.3 ms for the power supply to become stable. Check the P00/INT4 pin twice and remove chattering.

Timing Chart

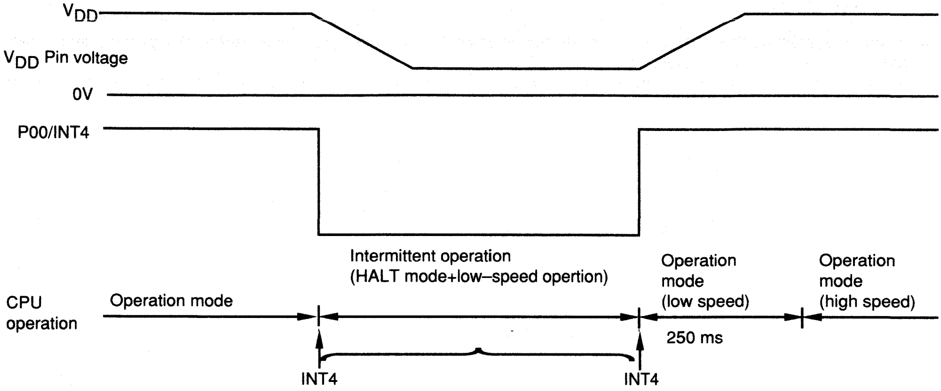


Programming example – INT4 service program with MBE = 0 –

```
VSUB4:      SKT     PORT0.0         ; P00 = 1?
            BR      PDOWN           ; Power down
            SET1    BTM.3           ; Power on
WAIT:       SKT     IRQBT           ; Waits for 31.3 ms
            BR      WAIT
            SKT     PORT0.0         ; Check chattering
            BR      PDOWN
            MOV     A, #0011B
            MOV     PCC, A          ; Set high-speed mode
            MOV     XA, #XXH        ; Set Port Mode register
            MOV     PMGm, XA        ;
            EI      IE0
            EI      IET0
            RETI
PDOWN:      MOV     A, #0           ; Minimum speed mode
            MOV     PCC, A
            MOV     XA, #00H
            MOV     PMGA, XA        ; Place input/output ports in high impedance
            MOV     PMGB, XA
            DI      IE0             ; Disables INT0 and INTT0
            DI      IET0
            MOV     A, #1011B
            MOV     BTM, A          ; Wait time = 31.3 ms
            STOP                    ; Sets STOP mode
            NOP
            RETI
```

(2) HALT mode application
Perform intermittent operation under the following conditions
— Change to the subsystem clock on the falling edge of INT4.
— Stop oscillation of the main system clock and set the HALT mode.
— Perform intermittent operation at 0.5sec. intervals during the standby mode.
— Again change to the main system clock on the rising edge of INT4.
— Do not use INTBT.

(Timing Chart)



```
Example (Initialization)
          MOV     A, #0011B
          MOV     PCC, A        ; High speed mode
          MOV     XA, #04
          MOV     WM, XA        ; Main system clock
          EI      IE4
          EI      IEW
          EI                    ; Enable interrupt
(Main routine)
          SKT     PORT0.0       ; Power OK?
          HALT                  ; Power down mode
          NOP                   ; Power OK?
          SKTCLR  IRQW          ; Is 0.5sec. flag set?
          BR      MAIN          ; NO
          CALL    WATCH         ; Watch subroutine
MAIN:       :
            :
            :
            :
(INT4 service routine)
INT4:     SKT     PORT0.0       ; Power OK?, MBE=0, RBE=0
          BR      PDOWN
          CLR1    SCC.3         ; Start oscillation of main system clock
          MOV     A, #8
          MOV     BTM, A
WAIT1:    SKT     IRQBT         ; Wait for 250 ms
          BR      WAIT1
          SKT     PORT0.0       ; Check chattering
          BR      PDOWN
          CLR1    SCC.0         ; Change to main system clock
          MOV     XA, #04H      ; Main system clock
```

```
         MOV    WM, XA
         RETI
PDOWN:   MOV    XA, #05H    ; Subsystem clock
         MOV    WM, XA
         SET1   SCC.0       ; Change to subsystem clock
         MOV    A, #6
WAIT2:   INCS   A           ; Wait for 32 machine cycles
         BR     WAIT2
         SET1   SCC.3       ; Stop oscillation of main system clock
         RETI
```

Note: When the system clock is changed after power on from main system clock to subsystem clock, change the system clock after the subsystem clock is stabilized.

## 8. RESET FUNCTION

The μPD7500X is reset when $\overline{\text{RESET}}$ is input. The hardware devices are initialized as listed inTable 8.1-1. Fig. 8.1-1 shows the reset operation timing.

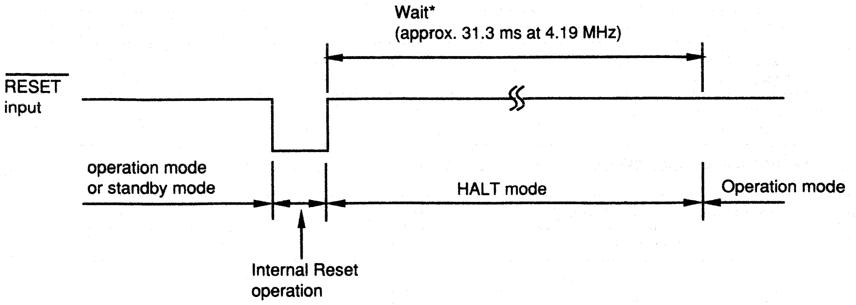

Wait*
(approx. 31.3 ms at 4.19 MHz)
$\overline{\text{RESET}}$ input
operation mode or standby mode
HALT mode
Operation mode
Internal Reset operation

**Fig. 8.1-1 Reset Operation by $\overline{\text{RESET}}$ is input**

**Table 8.1-1 State After Each Hardware Device is Reset**

| Hardware | | $\overline{\text{RESET}}$ input during standby mode | $\overline{\text{RESET}}$ input during operation |
|---|---|---|---|
| Program counter (PC) | μPD75004 | The low-order four bits of program memory address 0000H are loaded into PC11-PC8. The contents of address 0001H are loaded into PC7-PC0. | |
| | μPD75006,<br>μPD75008 | The low-order five bits of program memory address 0000H are loaded into PC12-PC8. The contents of address 0001H are loaded into PC7-PC0. | |
| PSW | Carry flag (CY) | Held | Undefined |
| | Skip flag (SK0-SK2) | 0 | 0 |
| | Interrupt status flag (IST0) | 0 | 0 |
| | Bank enable flag | Bit 7 of program memory address 0000H is loaded into MBE. | |
| Stack pointer (SP) | | Undefined | Undefined |
| Data memory (RAM) | | Held (Note) | Undefined |
| General-purpose registers (X, A, H, L, D, E, B, C) | | Held | Undefined |
| Bank selection register (MBS) | | 0 | 0 |
| Basic interval timer | Counter (BT) | Undefined | Undefined |
| | Mode register (BTM) | 0 | 0 |
| Timer event counter | Counter (T0) | 0 | 0 |
| | Modulo register (TMOD0) | FFH | FFH |
| | Mode register (TM0) | 0 | 0 |
| | TOE0, TOUT F/F | 0, 0 | 0, 0 |
| Watch timer | Mode register (WM) | 0 | 0 |
| Serial interface | Shift register (SIO) | Held | Undefined |

**Table 8.1-1 State After Each Hardware Device is Reset (Cont'd)**

| Hardware | | $\overline{\text{RESET}}$ input during standby mode | $\overline{\text{RESET}}$ input during operation |
|---|---|---|---|
| Serial interface | Operation mode register (CSIM) | 0 | 0 |
| | SBI control register (SBIC) | 0 | 0 |
| | Slave address register (SVA) | Held | Undefined |
| Clock generator and clock output circuit | Processor clock control register (PCC) | 0 | 0 |
| | System clock control register (SCC) | 0 | 0 |
| | Clock output mode register (CLOM) | 0 | 0 |
| Interrupt function | Interrupt request flags (IRQXXX) | Reset to 0 | Reset to 0 |
| | Interrupt enable flags (IEXXX) | 0 | 0 |
| | Interrupt master enable flag (IME) | 0 | 0 |
| | INT0, INT1, and INT2 mode registers (IM0, IM1, and IM2) | 0, 0, 0 | 0, 0, 0 |
| Digital ports | Output buffers | Off | Off |
| | Output latches | Cleared | Cleared |
| | Input/output mode registers (PMGA, B) | 0 | 0 |
| | Pull-up resistor specification register (POGA) | 0 | 0 |
| Bit sequential buffer (BSB0-BSB3) | | Held | Undefined |
| Pins condition | P00-P03, P10-P13, P20-P23, P30-P33, P60-P63, P70-P73, P80-P81 | Input | Input |
| | P40-P43, P50-P53, | — At incorporated pull-up resistor: ... High level<br>— At open drain: ... High impedance | |

Note: The data of data memory address 0F8H-0FDH is undefined by $\overline{\text{RESET}}$ input.

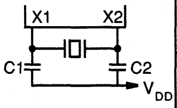
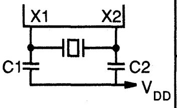
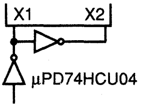
## 9. ELECTRICAL CHARACTERISTICS

**Absolute Maximum Ratings** (Ta = 25 °C)

| Item | Symbol | Conditions | | Ratings | Unit |
|---|---|---|---|---|---|
| Supply voltage | $V_{DD}$ | | | –0.3 to +7.0 | V |
| Input voltage | $V_{I1}$ | Other than ports 4 and 5 | | –0.3 to $V_{DD}$ +0.3 | V |
| | $V_{I2}$ | Ports 4 and 5 | With built-in pull-up resistor | –0.3 to $V_{DD}$ +0.3 | V |
| | | | Open drain | –0.3 to +11 | V |
| Output voltage | $V_O$ | | | –0.3 to $V_{DD}$ +0.3 | V |
| High level output current | $I_{OH}$ | Single pin | | –10 | mA |
| | | All pins | | –30 | |
| Low level output current | $I_{OL}$<br>Note | One port pin of ports 0, 3, 4 and 5 | Peak value | 30 | mA |
| | | | Effective value | 15 | |
| | | A pin other than ports 0, 3, 4 and 5 | Peak value | 20 | |
| | | | Effective value | 10 | |
| | | Total of ports 0, 3, 4, 5 and 8 | Peak value | 160 | |
| | | | Effective value | 120 | |
| | | Total of ports 2, 6 and 7 | Peak value | 66 | |
| | | | Effective value | 33 | |
| Operation Temperature | $T_{opt}$ | | | –40 to +85 | °C |
| Storage Temperature | $T_{stg}$ | | | –65 to +150 | °C |

Note: Use the following formula to calculate the effective value. (Effective value) = (Peak value) x $\sqrt{\text{duty}}$

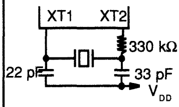
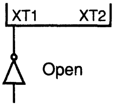
**Characteristics of Main System Clock Oscillator** (Ta = –40 to +85 °C, $V_{DD}$ = 2.7 to 6.0V)

| Oscillator | Recommended constants | Item | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Ceramic resonator |  | Note 1 Oscillation frequency ($f_{XX}$) | | 1.0 | | 5.0 | MHz |
| | | Note 2 Oscillation stabilization time | After $V_{DD}$ reaches the minimum value in the oscillator voltage range | | | 4 | ms |
| Crystal resonator |  | Note 1 Oscillation frequency ($f_{XX}$) | | 1.0 | 4.19 | 5.0 | MHz |
| | | Note 2 Oscillation stabilization time | $V_{DD}$ =4.5 to 6.0V | | | 10 | ms |
| | | | | | | 30 | ms |
| External clock |  | Note 1 X1 input frequency ($f_X$) | | 1.0 | | 5.0 | MHz |
| | | X1 input high/low level width ($t_{XH}$, $t_{XL}$) | | 100 | | 500 | ns |

Note 1: The oscillation frequency and X1 input frequency are indicated only to express the characteristics of the oscillator. Refer to the AC characteristics for instruction execution time.

Note 2: The oscillation stabilization time is the time required for the oscillator to stabillate after $V_{DD}$ is applied or after the STOP mode is released.

**Subsystem Clock Oscillator Characteristics** (Ta = –40 to +85 °C, $V_{DD}$ = 2.7 to 6.0V)

| Oscillator | Recommended constants | Item | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Crystal resonator |  | Oscillation frequency ($f_{XT}$) | | 32 | 32.768 | 35 | MHz |
| | | Oscillation stabilization time | $V_{DD}$ = 4.5 – 6.0V | | 1 | 2 | s |
| | | | | | | 10 | s |
| External clock |  | XT1 input frequency ($f_{XT}$) | | 32 | | 100 | kHz |
| | | XT1 input high/low level width ($t_{XTH}$, $t_{XTL}$) | | 5 | | 15 | μs |

**Recommended Resonator**

Main system clock: ceramic resonator

| Maunfacture | Product name | External capacitors (pF) | | Oscillator operating voltage range (V) | | Remarks |
|---|---|---|---|---|---|---|
| | | C1 | C2 | MIN. | MAX. | |
| Murata | CSA 2.00MG093 | 15 | 15 | 2.5 | 3.5 | Note |
| | CSB 1000D20 | 220 | 220 | 2.7 | 6.0 | |
| | CSA 2.00MG093<br>CSA 4.19MGU<br>CSA 4.91MGU | 30 | 30 | | | |
| | CST 2.00MG093<br>CST 4.19MGU<br>CST 4.91MGU | not required | not required | | | Internal C type |
| Kyocera | KBR–1000H | 100 | 100 | 2.7 | 6.0 | |
| | KBR–2.0MS | 47 | 47 | | | |
| | KBR–4.0MS<br>KBR–4.19MS<br>KBR–4.91MS | 33 | 33 | | | |

Note: When CSA 2.000G093 is used, VDD is 2.5 to 3.5V.

Main system clock: crystal resonator

| Maunfacture | Freg. Holder | External capacitors (pF) | | Oscillator operating voltage range (V) | | Remarks |
|---|---|---|---|---|---|---|
| | | C1 | C2 | MIN. | MAX. | |
| Kinseki | 1.00<br>2.00 HC–18/U<br>4.19 HC–49/U<br>4.91 HC–43/U | 22 | 22 | 2.7 | 6.0 | |

Subsystem clock: 32.768 KHz crystal resonator

| Maunfacture | Product name | External capacitors (pF) | | | Oscillator operating voltage range (V) | | Remarks |
|---|---|---|---|---|---|---|---|
| | | C1 (pF) | C2 (pF) | R (kΩ) | MIN. | MAX. | |
| Kinseki | P–3 | 22 | 22 | 330 | 2.7 | 6.0 | |

**Capacitance** (Ta = 25 °C, $V_{DD}$ = 0V)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Units |
|---|---|---|---|---|---|---|
| Input capacitance | $C_{IN}$ | f =1 MHz<br>For other than pins to be measured, 0V. | | | 15 | pF |
| Output capacitance | $C_{OUT}$ | | | | 15 | pF |
| Input/output capacitance | $C_{IO}$ | | | | 15 | pF |

**DC Characteristics** (Ta = –40 to +85 °C, $V_{DD}$ = 2.7 to 6.0V)

| Item | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| High level input voltage | $V_{IH1}$ | Ports 2, 3 and 8 | | $0.7V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH2}$ | Ports 0, 1, 6, 7, and $\overline{\text{RESET}}$ pin | | $0.8V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH3}$ | Ports 4 to 5 | With built-in pullup resistor | $0.7V_{DD}$ | | $V_{DD}$ | V |
| | | | Open drain | $0.7V_{DD}$ | | 10 | V |
| | $V_{IH4}$ | X1, X2, XT1 | | $V_{DD}$–0.5 | | $V_{DD}$ | V |
| Low level input voltage | $V_{IL1}$ | Ports 2, 3, 4, 5, 8 | | 0 | | $0.3V_{DD}$ | V |
| | $V_{IL2}$ | Ports 0, 1, 6, 7, and $\overline{\text{RESET}}$ pin | | 0 | | $0.2V_{DD}$ | V |
| | $V_{IL3}$ | X1, X2, XT1 | | 0 | | 0.4 | V |
| High level output voltage | $V_{OH1}$ | Ports 0, 2, 3 6, 7, 8 | $V_{DD}$ = 4.5 to 6.0V $I_{OH}$ = –1mA | $V_{DD}$–1.0 | | | V |
| | | | $I_{OH}$ = –100µA | $V_{DD}$–0.5 | | | V |
| Built-in pull-up resistor | $R_{L1}$ | Ports 0, 1, 2, 3, 6, 7, 8 (Except P00) $V_{IN}$ = 0V | $V_{DD}$ = 5.0V ± 10% | 15 | 40 | 80 | kΩ |
| | | | $V_{DD}$ = 3.0V ± 10% | 30 | | 200 | kΩ |
| | $R_{L2}$ | Ports 4 and 5 $V_{OUT}$= $V_{DD}$ –2.0V | $V_{DD}$ = 5.0V ± 10% | 15 | 40 | 70 | kΩ |
| | | | $V_{DD}$ = 3.0V ± 10% | 10 | | 60 | kΩ |
| Low level output voltage | $V_{OL1}$ | Ports 0, 2, 3, 4, 5, 6, 7, 8 | Ports 4, 5 $V_{DD}$ = 4.5-6.0V $I_{OL}$ = 15mA | | 0.4 | 2.0 | V |
| | | | Port 3 $V_{DD}$ = 4.5-6.0V $I_{OL}$ = 15mA | | 0.6 | 2.0 | V |
| | | | $V_{DD}$ = 4.5-6.0V $I_{OL}$ = 1.6mA | | | 0.4 | V |
| | | | $I_{OL}$ = 400µA | | | 0.5 | V |
| | $V_{OL2}$ | SB0, 1 open drain | Pull-up resistance: 1kΩ min. $V_{DD}$ = 4.5-6.0V | | | $0.2V_{DD}$ | V |
| | | | Pull-up resistance: 5kΩ min. | | | $0.2V_{DD}$ | V |
| High level input leakage current | $I_{LIH1}$ | $V_{IN}$ = $V_{DD}$ | Other than indicated below | | | 3 | µA |
| | $I_{LIH2}$ | | X1, X2, XT1 | | | 20 | µA |
| | $I_{LIH3}$ | $V_{IN}$ = 10V | Ports 4, 5 (with open drain) | | | 20 | µA |
| Low level input leakage current | $I_{LIL1}$ | $V_{IN}$ = 0V | Other than indicated below | | | –3 | µA |
| | $I_{LIL2}$ | | X1, X2, XT1 | | | –20 | µA |
| High level output leakage current | $I_{LOH1}$ | $V_{OUT}$ = $V_{DD}$ | Other than indicated below | | | 3 | µA |
| | $I_{LOH2}$ | $V_{OUT}$ = 10V | Ports 4, 5 (with open drain) | | | 20 | µA |
| Low level output leakage current | $I_{LOL}$ | $V_{OUT}$ = 0V | | | | –3 | µA |

**DC Characteristics** (Ta = –40 to +85 °C, $V_{DD}$ = 2.7 to 6.0V)

| Item | Symbol | Conditions | | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|---|
| Supply current (Note 1) | $I_{DD1}$ | (Note 4) 4.19 MHz crystal oscillation C1=C2=22pF | $V_{DD}$ = 5V ± 10% (Note 2) | | | 2.5 | 8 | mA |
| | | | $V_{DD}$ = 3V ± 10% (Note 3) | | | 0.35 | 1.2 | mA |
| | $I_{DD2}$ | | HALT Mode | $V_{DD}$ = 5V ± 10% | | 500 | 1500 | μA |
| | | | | $V_{DD}$ = 3V ± 10% | | 150 | 450 | μA |
| | $I_{DD3}$ | 32kHz crystal resonator (Note 5) | $V_{DD}$ = 3V ± 10% | | | 30 | 90 | μA |
| | $I_{DD4}$ | | HALT Mode | $V_{DD}$ = 3V ± 10% | | 5 | 15 | μA |
| | $I_{DD5}$ | XT1 = 0 STOP mode | $V_{DD}$ = 5V ± 10% | | | 0.5 | 20 | μA |
| | | | $V_{DD}$ = 3V ± 10% | | | 0.1 | 10 | μA |
| | | | | Ta = 25°C | | 0.1 | 5 | μA |

Note 1: The current of the built-in pull-up resistor is not included.
Note 2: When operated in the high-speed mode with the processor clock control register (PCC) set to 0011.
Note 3: When operated in the low-speed mode with the PCC set to 0000.
Note 4: Includes the power consumption for the sub oscillation.
Note 5: When the system clock control register (SCC) is set to 1001, main system clock oscillator is stopped and operated by the subsystem clock.

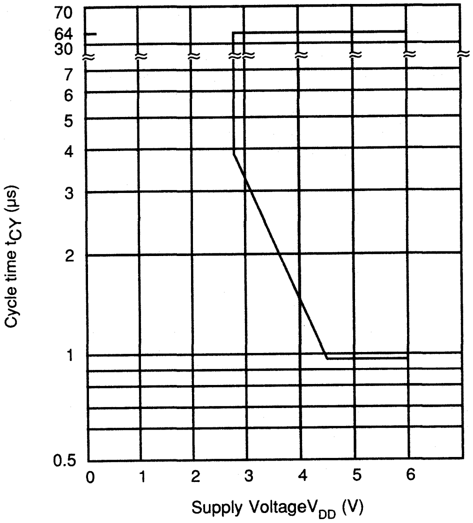
**AC Characteristics** (Ta = –40 to 85 C, $V_{DD}$ = 2.7 to 6.0V)

| Item | Symbol | Condition | | MIN. | TYP. | MAX | Unit |
|---|---|---|---|---|---|---|---|
| (Note 1) Cycle time (minimum instruction execution time) | $t_{CY}$ | Operation with main system clock | $V_{DD}$ = 4.5 –6.0V | 0.95 | | 64 | μs |
| | | | | 3.8 | | 64 | μs |
| | | Operation with sub-system clock | | 114 | 122 | 125 | μs |
| TI0 input frequency | $f_{TI}$ | $V_{DD}$ = 4.5 – 6.0V | | 0 | | 1 | MHz |
| | | | | 0 | | 275 | kHz |
| TI0 input high/low level width | $t_{TIH}$ | $V_{DD}$ = 4.5 – 6.0V | | 0.48 | | | μs |
| | $t_{TIL}$ | | | 1.8 | | | μs |
| Interrupt input high/low level width | $t_{INTH}$ | INT0 | | Note 2 | | | μs |
| | | INT1, 2, 4 | | 10 | | | μs |
| | $t_{INTL}$ | KR0-7 | | 10 | | | μs |
| $\overline{\text{RESET}}$ low level width | $t_{RSL}$ | | | 10 | | | μs |

Note 1: The cycle time (minimum instruction execution time) is determined by the oscillation frequency of the connected resonator, the system clock control register (SCC), and the processor clock control register (PCC). The figure below shows the $V_{DD}$ vs cycle time ($t_{CY}$) when operated with the main system clock.
Note 2: $2t_{CY}$ or $128/f_X$, depending on the setting of the interrupt mode register (IM0).

(with main system clock)

$t_{CY}$ vs. $V_{DD}$



**Serial Transfer Operation**

2-line/3line serial I/O mode ($\overline{SCK}$ ... Internal clock output)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{SCK}$ cycle time | $t_{KCY1}$ | $V_{DD}$ = 4.5 –6.0V | 1600 | | | ns |
| | | | 3800 | | | ns |
| $\overline{SCK}$ high/low level width | $t_{KL1}$ | $V_{DD}$ = 4.5 –6.0V | $\frac{t_{KCY1}}{2}$ -50 | | | ns |
| | $t_{KH1}$ | | $\frac{t_{KCY1}}{2}$ -150 | | | ns |
| SI set-up time (against $\overline{SCK}$ ↑ ) | $t_{SIK1}$ | | 150 | | | ns |
| SI hold time (against $\overline{SCK}$ ↑ ) | $t_{KSI1}$ | | 400 | | | ns |
| $\overline{SCK}$ ↓ → SO output delay time | $t_{KSO1}$ | $V_{DD}$ = 4.5 –6.0V | | | 250 | ns |
| | | | | | 1000 | ns |

2-line/3line serial I/O mode ($\overline{SCK}$ ... External clock input)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{SCK}$ cycle time | $t_{KCY2}$ | $V_{DD}$ = 4.5 –6.0V | 800 | | | ns |
| | | | 3200 | | | ns |
| $\overline{SCK}$ high/low level width | $t_{KL2}$ | $V_{DD}$ = 4.5 –6.0V | 400 | | | ns |
| | $t_{KH2}$ | | 1600 | | | ns |
| SI set-up time (against $\overline{SCK}$ ↑ ) | $t_{SIK2}$ | | 100 | | | ns |
| SI hold time (against $\overline{SCK}$ ↑ ) | $t_{KSI2}$ | | 400 | | | ns |
| $\overline{SCK}$ ↓ → SO output delay time | $t_{KSO2}$ | $V_{DD}$ = 4.5 –6.0V | | | 300 | ns |
| | | | | | 1000 | ns |

Note: The output delay time (for rising edge) of the serial line must be shorter than 600 ns. For example, if SB0 and SBI are pulled up with 5K ohms, the total capacitance of the serial bus line must be no greater than 120 pF.

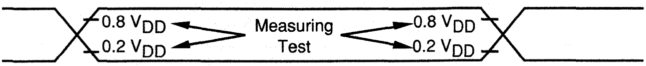
SBI mode ($\overline{SCK}$ ... Internal clock output (master))

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{SCK}$ cycle time | $t_{KCY3}$ | $V_{DD}$ = 4.5 –6.0V | 1600 | | | ns |
| | | | 3800 | | | ns |
| $\overline{SCK}$ high/low level width | $t_{KL3}$ | $V_{DD}$ = 4.5 –6.0V | $\frac{t_{KCY1}}{2}$ -50 | | | ns |
| | $t_{KH3}$ | | $\frac{t_{KCY1}}{2}$-150 | | | ns |
| SB0,1 set-up time (against $\overline{SCK}$ ↑ ) | $t_{SIK3}$ | | 150 | | | ns |
| SB0,1 hold time (against $\overline{SCK}$ ↑ ) | $t_{KSI3}$ | | $\frac{t_{KCY1}}{2}$ | | | ns |
| $\overline{SCK}$↓ → SB0,1 output delay time | $t_{KSO3}$ | $V_{DD}$ = 4.5 –6.0V | 0 | | 250 | ns |
| | | | 0 | | 1000 | ns |
| $\overline{SCK}$ ↑→ SB0,1↓ | $t_{KSB}$ | | $t_{KCY}$ | | | ns |
| SB0,1↓→ $\overline{SCK}$ ↓ | $t_{SBK}$ | | $t_{KCY}$ | | | ns |
| SB0,1 low level width | $t_{SBL}$ | | $t_{KCY}$ | | | ns |
| SB0,1 high level width | $t_{SBH}$ | | $t_{KCY}$ | | | ns |

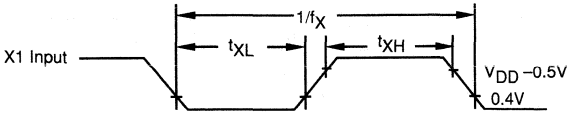
SBI mode ($\overline{SCK}$ ... External clock input (slave))

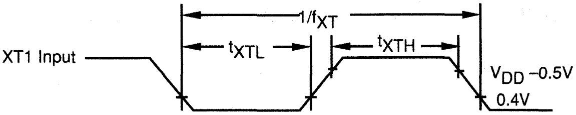
| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{SCK}$ cycle time | $t_{KCY4}$ | $V_{DD}$ = 4.5 –6.0V | 800 | | | ns |
| | | | 3200 | | | ns |
| $\overline{SCK}$ high/low level width | $t_{KL4}$ | $V_{DD}$ = 4.5 –6.0V | 400 | | | ns |
| | $t_{KH4}$ | | 1600 | | | ns |
| SB0,1 set-up time (against $\overline{SCK}$ ↑ ) | $t_{SIK4}$ | | 100 | | | ns |
| SB0,1 hold time (against $\overline{SCK}$ ↑ ) | $t_{KSI4}$ | | $\frac{t_{KCY1}}{2}$ | | | ns |
| $\overline{SCK}$↓ → SB0,1 output delay time | $t_{KSO4}$ | $V_{DD}$ = 4.5 –6.0V | 0 | | 300 | ns |
| | | | 0 | | 1000 | ns |
| $\overline{SCK}$ ↑→ SB0,1↓ | $t_{KSB}$ | | $t_{KCY}$ | | | ns |
| SB0,1↓→ $\overline{SCK}$ ↓ | $t_{SBK}$ | | $t_{KCY}$ | | | ns |
| SB0,1 low level width | $t_{SBL}$ | | $t_{KCY}$ | | | ns |
| SB0,1 high level width | $t_{SBH}$ | | $t_{KCY}$ | | | ns |

AC Timing Measurement Points (Except X1 and XT1 inputs)

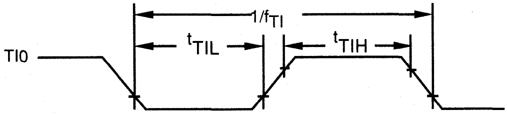


Clock Timing

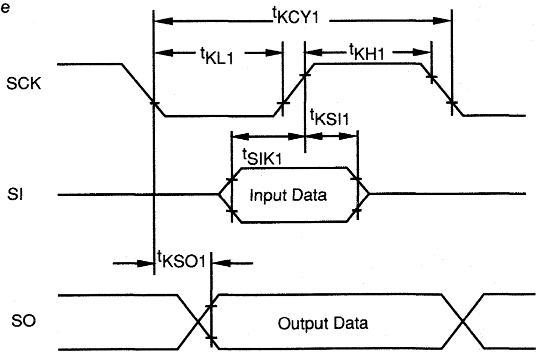




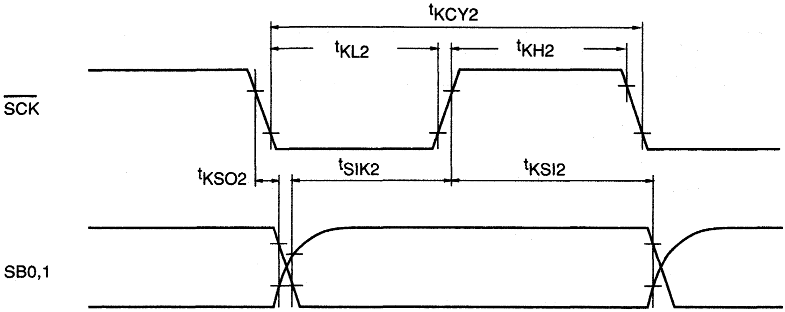
TIO Timing



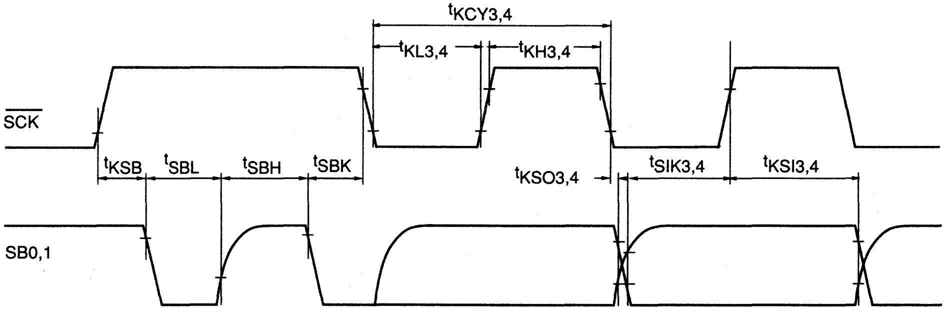
Serial Transfer Timing
3-line serial I/O mode



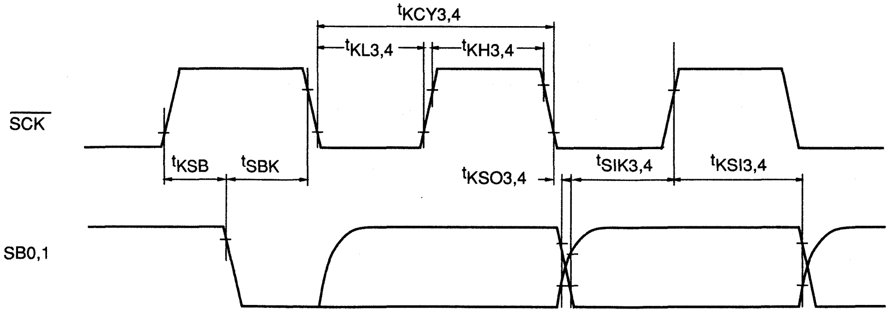
2-line serial I/O mode



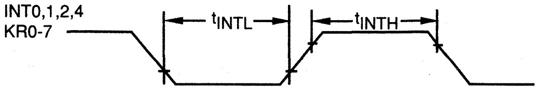
**Serial Transfer Timing**
Bus release signal transfer:



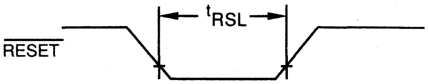
Command signal transfer:



Interrupt input timing



$\overline{\text{RESET}}$ input timing



**Data Memory STOP Mode Low Voltage Data Retention Characteristic** (Ta = –40 to +85 °C)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Data retention voltage | $V_{DDDR}$ | | 2.0 | | 6.0 | V |
| Data retention current (Note1) | $I_{DDDR}$ | $V_{DDR}$ = 2.0V | | 0.1 | 10 | μA |
| Release signal set time | $t_{SREL}$ | | 0 | | | μs |
| Oscillation stabilization time (Note 2) | $t_{WAIT}$ | Release by $\overline{\text{RESET}}$ input | | $2^{17}/f_X$ | | ms |
| | | Release by interrupt request | | Note 3 | | ms |

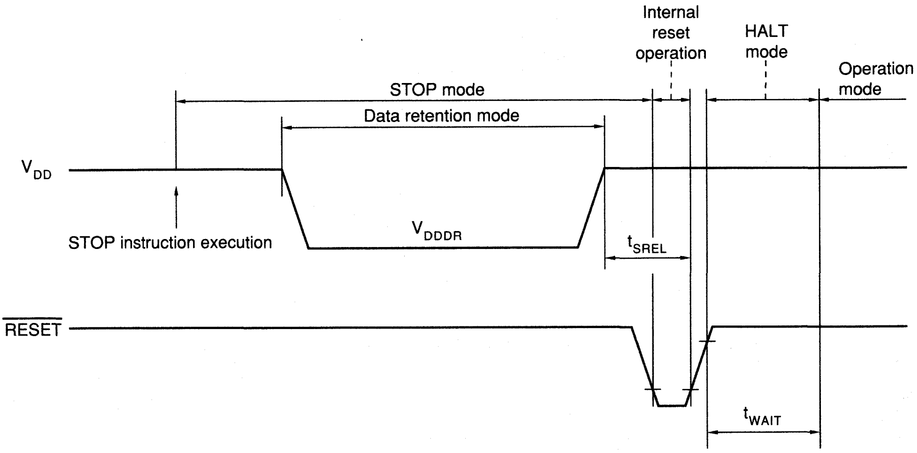
Note 1: Current in the internal pull-up resistors is not included.

Note 2: The oscillation stabilization time is the time required before beginning CPU operation in order to prevent unstable CPU operation when oscillation is initiated.

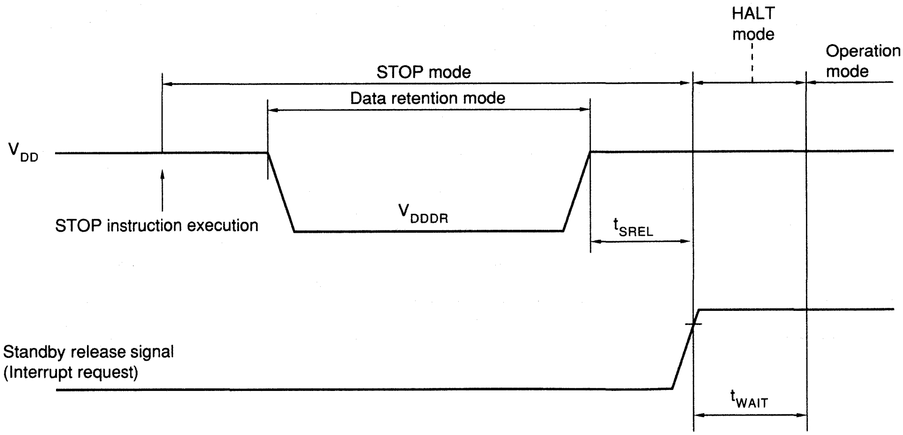
Note 3: Depends on the setting of the basic interval timer mode register (BTM) (refer to the table below).

| BTM3 | BTM2 | BTM1 | BTM0 | WAIT time ( ) indicates $f_X$ =4.19MHz |
|---|---|---|---|---|
| — | 0 | 0 | 0 | $2^{20}/f_X$ (Approximately 250 ms) |
| — | 0 | 1 | 1 | $2^{17}/f_X$ (Approximately 31.3 ms) |
| — | 1 | 0 | 1 | $2^{15}/f_X$ (Approximately 7.82 ms) |
| — | 1 | 1 | 1 | $2^{13}/f_X$ (Approximately 1.95 ms) |

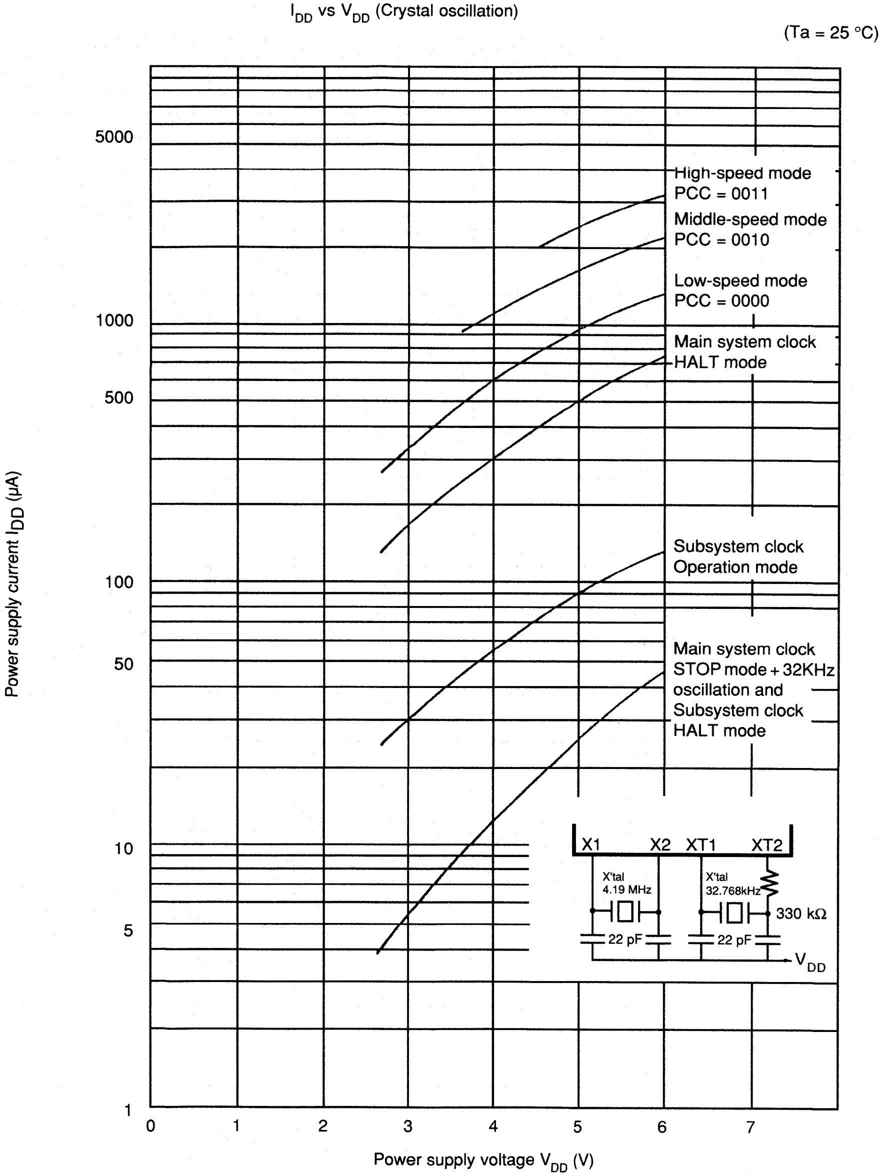
Data Retention Timing (when STOP mode is released by $\overline{\text{RESET}}$ input)



Data Retention Timing
(Standby release signal: STOP mode release by interrupt signal)

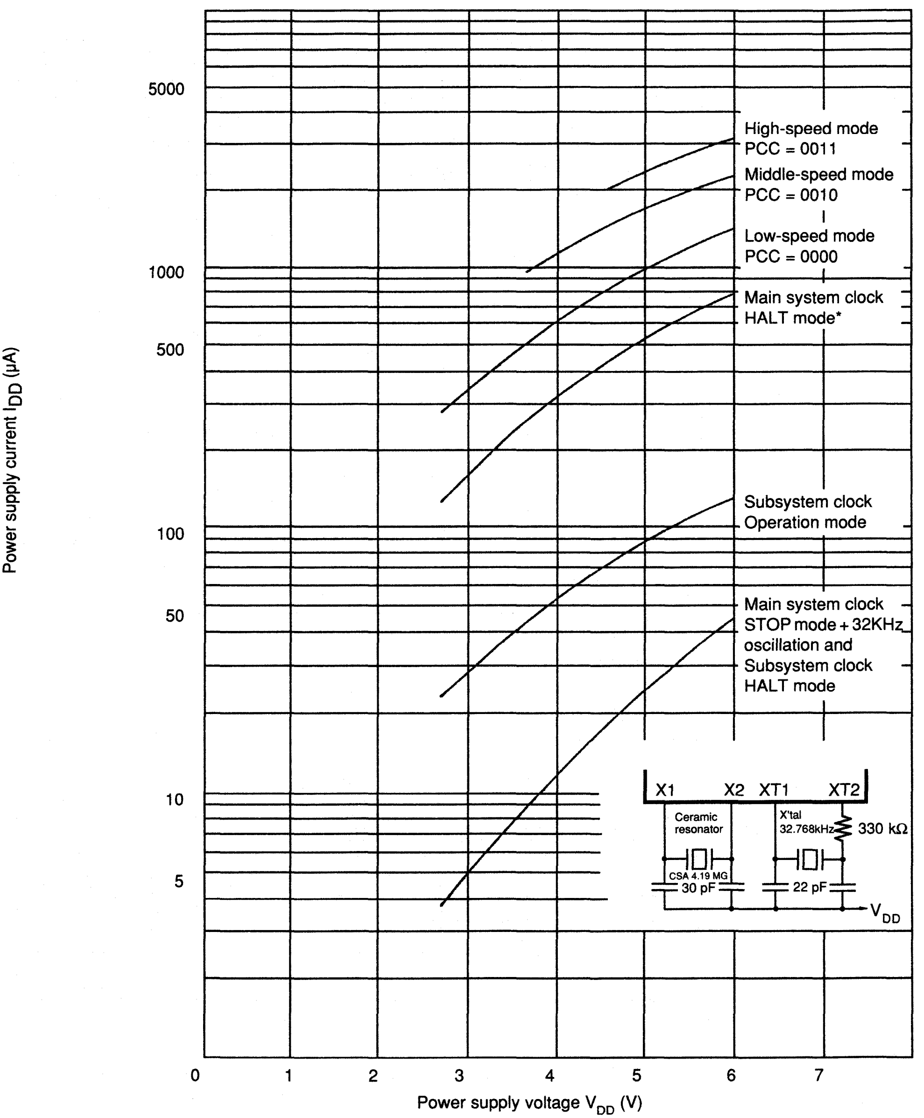


**Characteristic Curves**



$I_{DD}$ vs $V_{DD}$ (Ceramic oscillation)

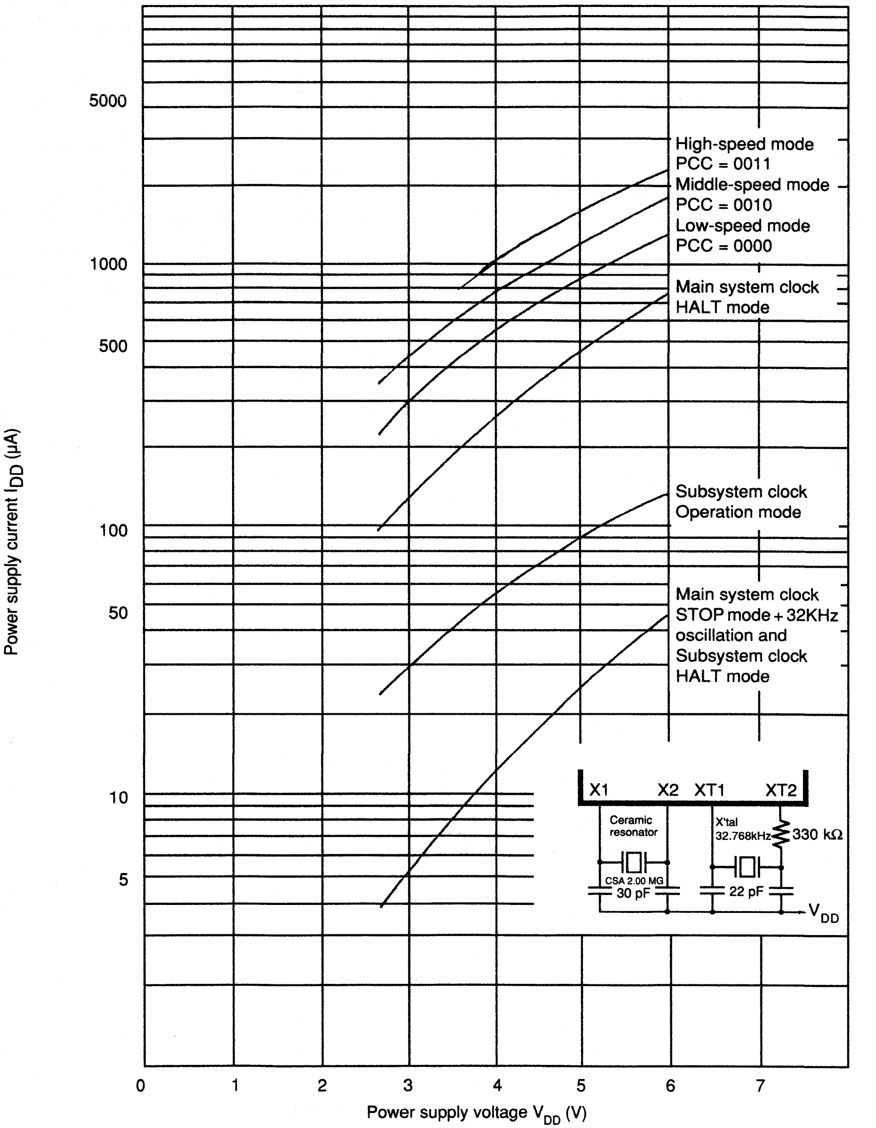
(Ta = 25 °C)



*The Current values are increased by approximately 10% compared to when operating with a crystal resonator.

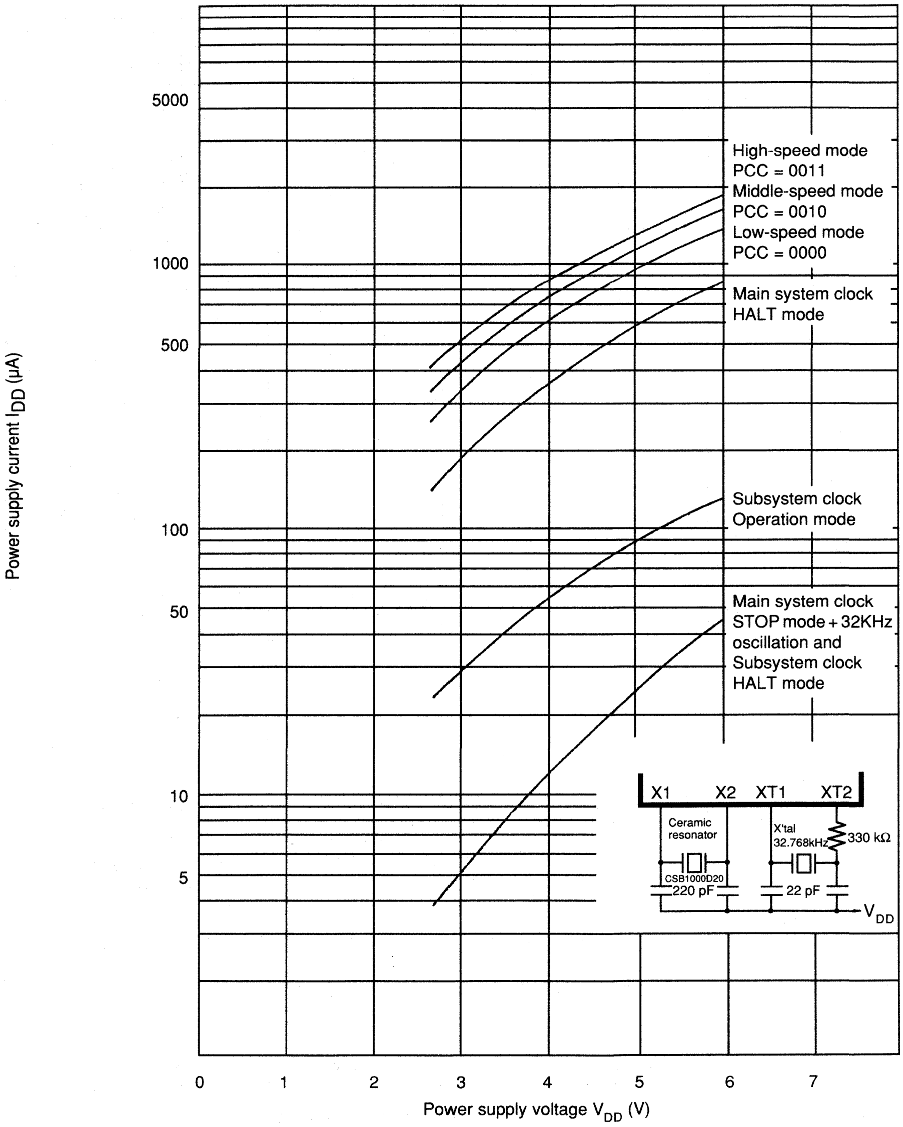
$I_{DD}$ vs $V_{DD}$ (Ceramic oscillation)

(Ta = 25 °C)

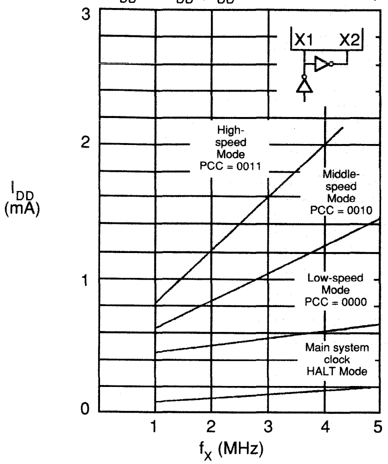


$I_{DD}$ vs $V_{DD}$ (Ceramic oscillation)

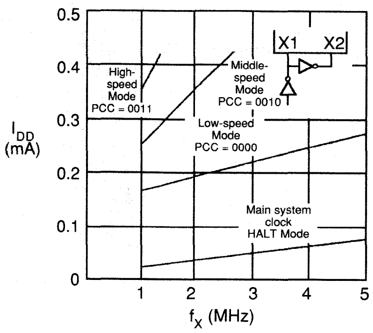
(Ta = 25 °C)



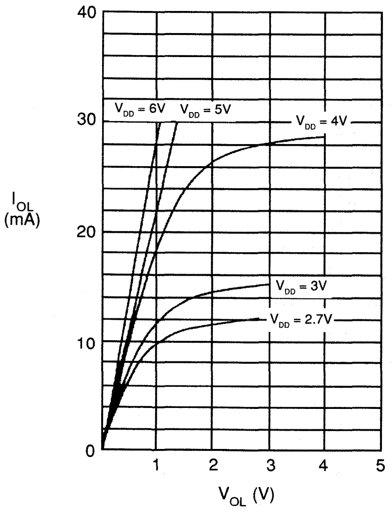
$I_{DD}$ vs $V_{DD}$ ($V_{DD}$ = 5V, Ta = 25 °C)



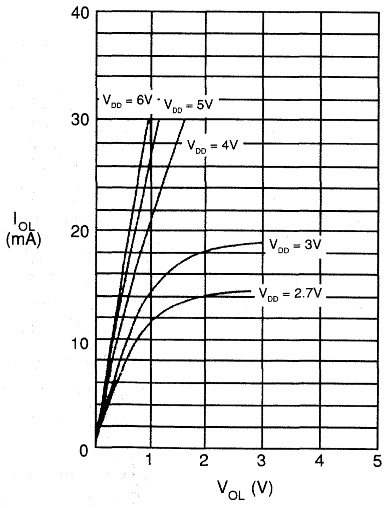
$I_{DD}$ vs $f_X$ ($V_{DD}$ = 3V, Ta = 25 °C)

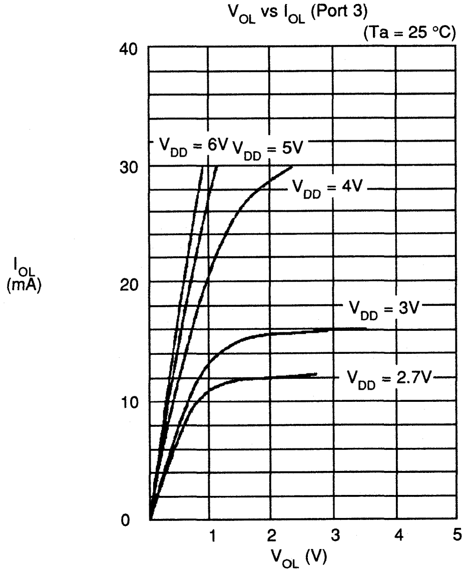


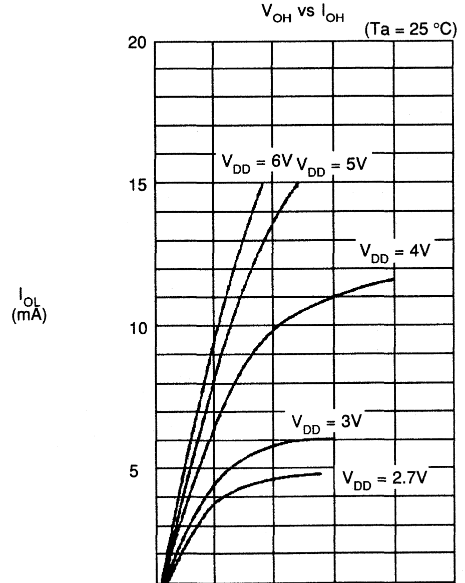
$V_{OL}$ vs $I_{OL}$ (Ports 0, 2, 6, 7, 8 Ta = 25 °C)



$V_{OL}$ vs $I_{OL}$ (Ports 4, 5, Ta = 25 °C)



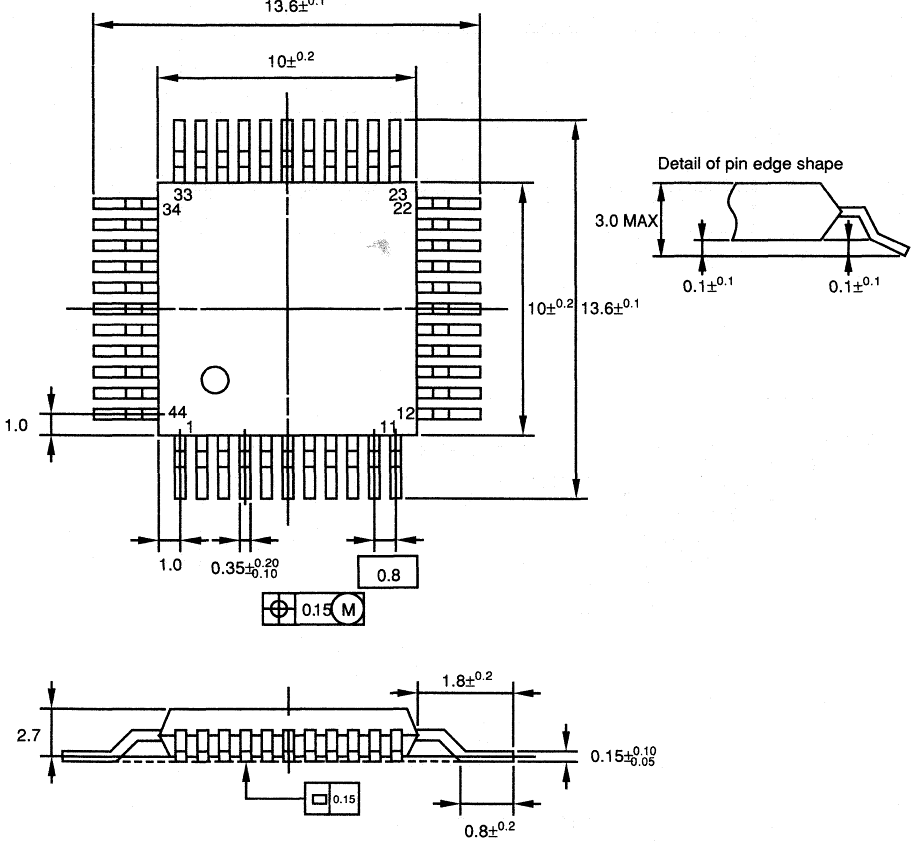

$V_{OL}$ vs $I_{OL}$ (Port 3)
(Ta = 25 °C)
$I_{OL}$ (mA)
40
30
20
10
0
1
2
3
4
5
$V_{OL}$ (V)
$V_{DD}$ = 6V
$V_{DD}$ = 5V
$V_{DD}$ = 4V
$V_{DD}$ = 3V
$V_{DD}$ = 2.7V


$V_{OH}$ vs $I_{OH}$
(Ta = 25 °C)
$I_{OL}$ (mA)
20
15
10
5
0
1
2
3
4
5
$V_{OH} - I_{OH}$ (V)
$V_{DD}$ = 6V
$V_{DD}$ = 5V
$V_{DD}$ = 4V
$V_{DD}$ = 3V
$V_{DD}$ = 2.7V

### 9.1 Package Information

Dimension of 44-pin plastic flat package (units: mm)

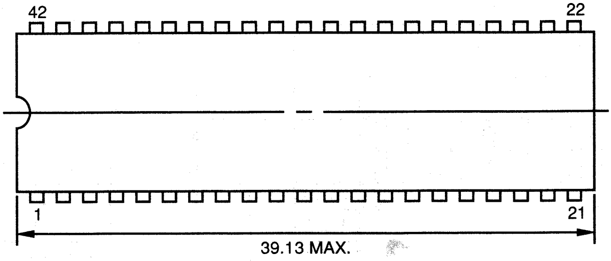
Top view

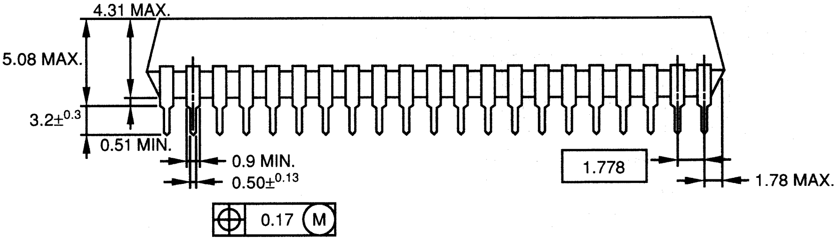


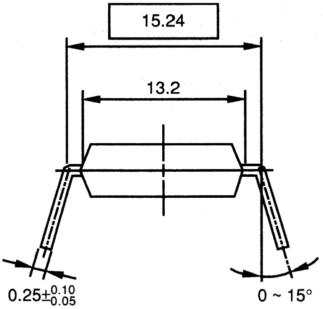
P44GB-80-3B4

42-pin plastic shrink DIP (600 mil) P42C-70-600A

Top view

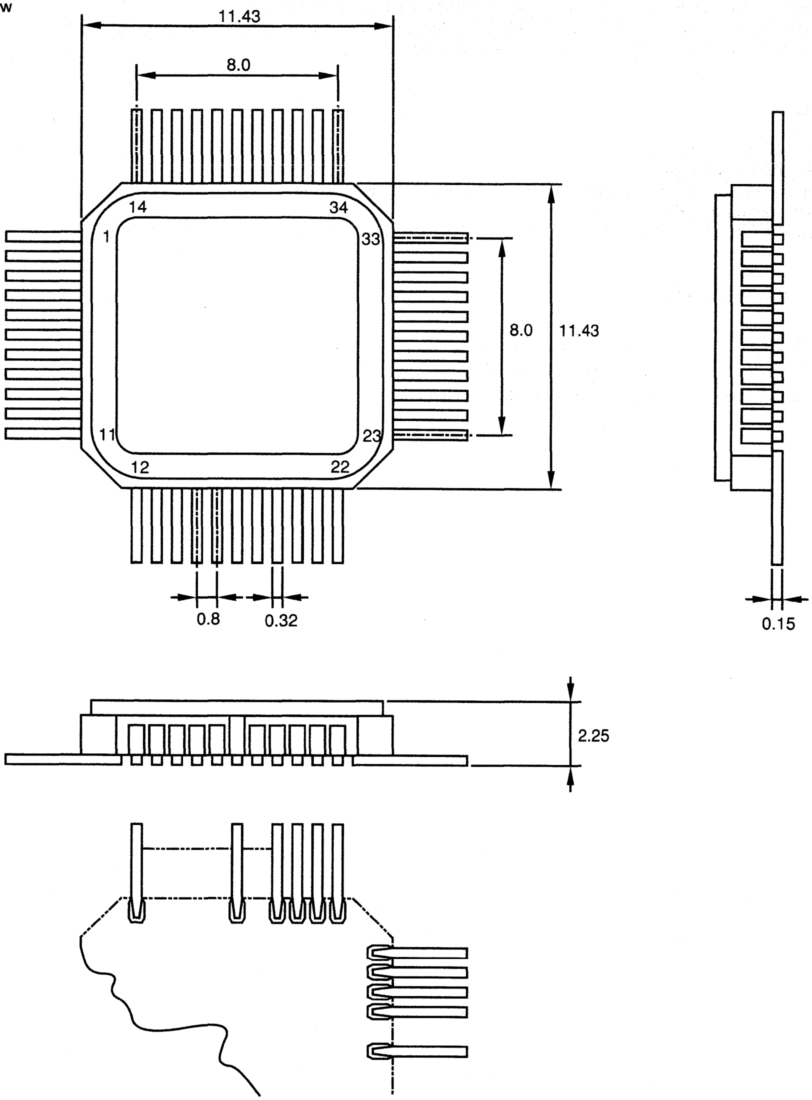






Dimension of 44-pin plastic flat package for ES (for reference) (unit: mm)

Top view



Note 1: A metal cap is connected to pin 17 and is $V_{SS}$ (GND) level.
Note 2: Lead wire length is not stipulated because cutting process of lead ends is out of production control.

## 10. μPD75P008 (OTPROM) 4-BIT MICROCOMPUTER

The μPD75P008 replaces the internal ROM of the μPD75004/006/008 with PROM.
The μPD75P008 is available as a one-time PROM version that can be only once written (for small quantity production).
μPD75P008 should not be used for final EMI and Latch up evaluation.

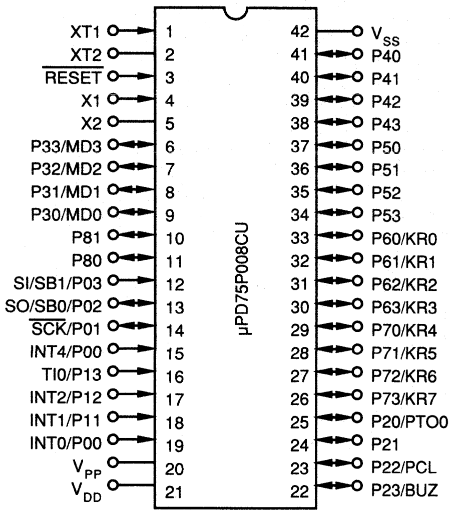
### 10.1 Features

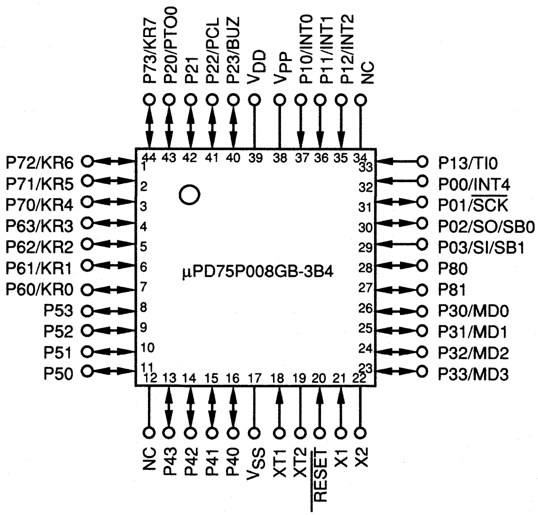
- μPD75004/006/008 compatible
- Internal PROM: 8064 x eigth bits
- Internal pull-up resistor (Ports 0-3 and 6-8) can be specified by using software.
- Open drain input/output (Ports 4 and 5)
- Single power supply: 5V ± 10%

### 10.2 Ordering Information

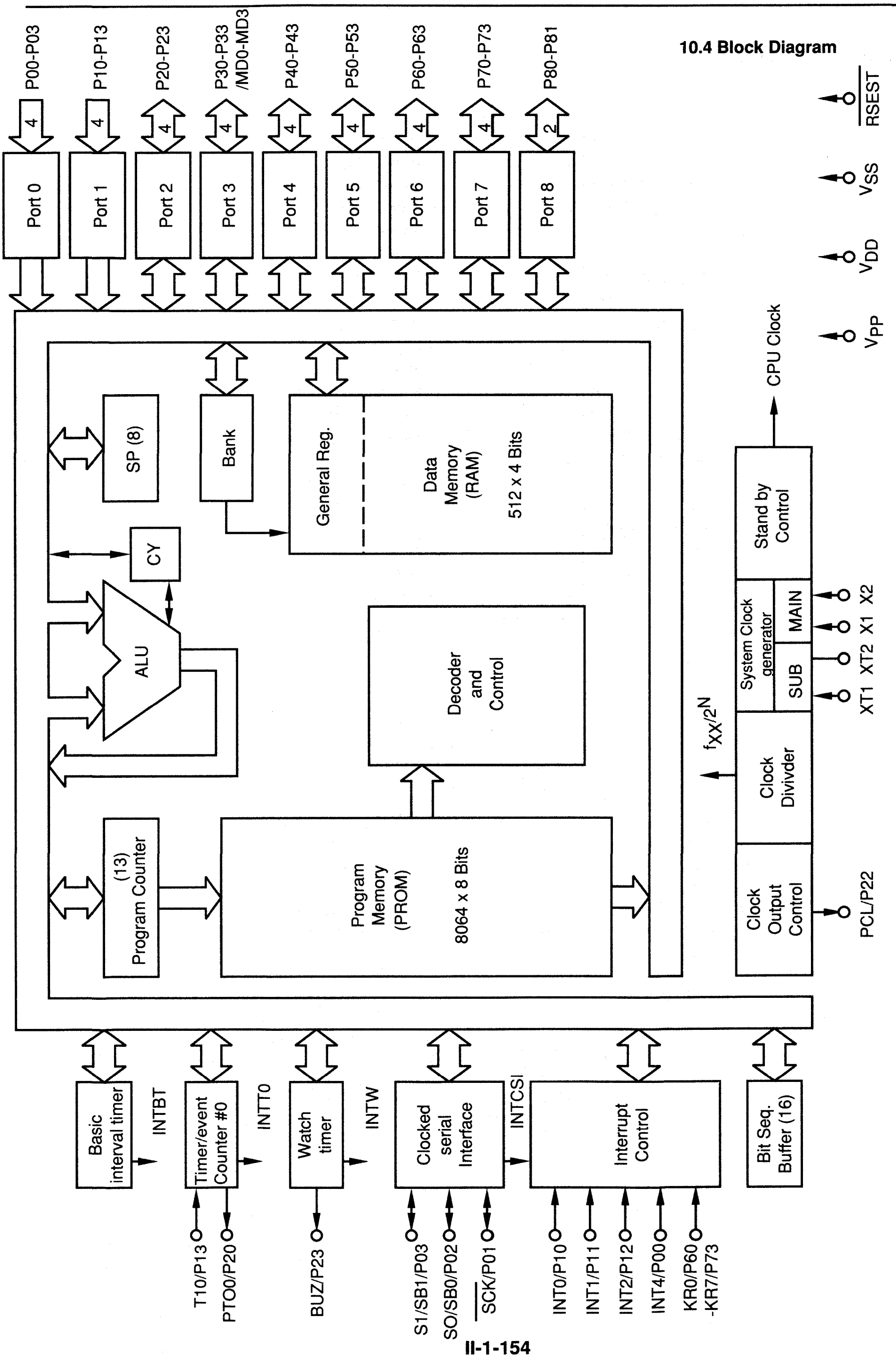
| Produce name | Package |
|---|---|
| μPD75P008CU | 42-pin plastic shrink DIP |
| μPD75P008GB-3B4 | 44-pinplastic flat-pack (bent lead) |

### 10.3 Pin Connection (Top View)





### 10.4 Block Diagram



## 10.5 Pin Functions

10.5.1 Port Pins

| Pin | Input/Output | Serves as Pin for | Functions | 8-Bit I/O | At Reset | I/O Circuit Type (Note 1) |
|---|---|---|---|---|---|---|
| P00 | Input | INT4 | 4-bit input port (PORT0).<br>Incorporation of pull-up resistors in 3-bit units for P01 to P03 specifiable by software. | X | Input | Ⓑ |
| P01 | Input/Output | $\overline{\text{SCK}}$ | | | | Ⓕ-A |
| P02 | Input/Output | SO/SB0 | | | | Ⓕ-B |
| P03 | Input/Output | SI/SBI | | | | Ⓜ-C |
| P10 | Input | INT0 | Noise elimination function available<br>4-bit input port (PORT1).<br>Incorporation of pull-up resistors in 4-bit units specifiable by software. | X | Input | Ⓑ-C |
| P11 | | INT1 | | | | |
| P12 | | INT2 | | | | |
| P13 | | TI0 | | | | |
| P20 | Input/Output | PTO0 | 4-bit input/output port (PORT2).<br>Incorporation of pull-up resistors in 4-bit units specifiable by software. | X | Input | E-B |
| P21 | | — | | | | |
| P22 | | PCL | | | | |
| P23 | | BUZ | | | | |
| P30 (Note 2) | Input /Output | MD0 | Programmable 4-bit input/output port<br>Input/output setting is possible in bit units. In corporation of pull up resistors in 4 bit units specifiable by software. | X | Input | E-B |
| P31 (Note 2) | | MD1 | | | | |
| P32 (Note 2) | | MD2 | | | | |
| P33 (Note 2) | | MD3 | | | | |
| P40-P43 (Note 2) | Input/Output | — | N-channel open drain 4-bit input/output port (PORT4).<br>Data input/output pin (lower 4-bits) for use during program memory (EPROM) write/ verify operations. | O | High Impedance | M-B |
| P50-P53 (Note 2) | Input/Output | — | N-channel open drain 4-bit input/output port (PORT5).<br>Data input/output pin (upper 4-bits) for use during program memory (EPROM) write/ verify operations. | | High Impedance | M-B |
| P60 | Input/Output | KR0 | Programmable 4-bit input/output port (PORT 6).<br>This port can be specified for input/output in bit units.<br>Incorporates software-specifiable pull-up resistors in 4-bit units. | O | Input | Ⓕ-A |
| P61 | | KR1 | | | | |
| P62 | | KR2 | | | | |
| P63 | | KR3 | | | | |
| P70 | Input/Output | KR4 | 4-bit input/output port (PORT7)<br>Incorporation of pull-up resistors in 4-bit units specifiable by software. | | Input | Ⓕ-A |
| P71 | | KR5 | | | | |
| P72 | | KR6 | | | | |
| P73 | | KR7 | | | | |
| P80 | Input/Output | — | 2-bit input/output port (PORT8)<br>Incorporation of pull-up resistors in 4-bit units specifiable by software. | X | Input | E-B |
| P81 | | — | | | | |

Note 1: Circles indicate Schmitt trigger inputs.

Note 2: LED direct drive is possible.

10.5.2 Non-Port Pins

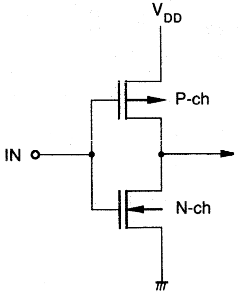
| Pin | Input/Output | Serves as Pin for | Functions | At Reset | I/O Circuit Type (Note 1) |
|---|---|---|---|---|---|
| TI0 | Input | P13 | External event pulse input pin for timer/event counter. | | Ⓑ-C |
| PTO0 | Input/Output | P20 | Timer/event counter output pin. | Input | E-B |
| PCL | Input/Output | P22 | Clock output pin. | Input | E-B |
| BUZ | Input/Output | P23 | Fixed-frequency output pin (for buzzer or system clock trimming). | Input | E-B |
| $\overline{\text{SCK}}$ | Input/Output | P01 | Serial clock input/output pin. | Input | Ⓕ-A |
| SO/SB0 | Input/Output | P02 | Serial data output pin. Serial bus input/output pin. | Input | Ⓕ-B |
| SI/SBI | Input/Output | P03 | Serial data input pin. Serial bus input/output pin. | Input | Ⓜ-C |
| INT4 | Input | P00 | Edge detector vector interrupt input pin (either rising or falling edge detection). | | Ⓑ |
| INT0 | Input | P10 | Edge detector interrupt input pin (detected edge selectable). | | Ⓑ-C |
| INT1 | | P11 | | | |
| INT2 | Input | P12 | Edge detection testable input pin (rising edge detection). | | Ⓑ-C |
| KR0-KR3 | Input/Output | P60-P63 | Testable input/output pin (for falling edge detection). | Input | Ⓕ-A |
| KR4-KR7 | Input/Output | P70-P73 | Testable input/output pin (for falling edge detection). | Input | Ⓕ-A |
| X1, X2 | Input | | Pin for connection of crystal/ceramics for main system clock generation. External clocks are input to X1, and their negative phase components are input to X2. | | |
| XT1 | Input | | Pin for connection of crystals for sub-system clock generation. External clocks are input to XT1, and XT2 is disconnected. XT1 can also be used as a 1-bit input (test) pin. | | |
| XT2 | | | | | |
| $\overline{\text{RESET}}$ | Input | | System reset input pin (low level active). | | Ⓑ |
| MD0-MD3 | Input/Output | P30-P33 | Mode selector pin for program memory (EPROM) write/verify operations. | Input | E-B |
| $V_{PP}$ | | | Program voltage application pin for program memory (PROM) write/verify operations. $V_{PP}$ is normally connected to $V_{DD}$. Connect to +12.5V for PROM write/verify operations. | | |
| $V_{DD}$ | | | Positive power supply pin. | | |
| $V_{SS}$ | | | GND potential pin. | | |

Note 1: Circles indicate Schmitt trigger inputs.

Pin I/O configurations

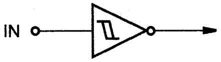
Following figures show the internal circuit configurations at the I/O ports.

(1) Type A (part of Type E–B)



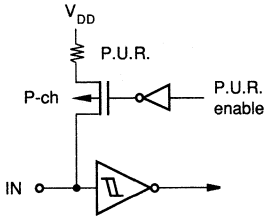
This is a CMOS standard input buffer.

(2) Type B



This is a Schmitt trigger input with hysteresis characteristics.

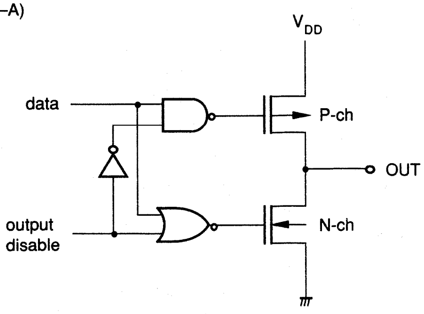
(3) Type B–C



P.U.R. : Pull-Up Resistor

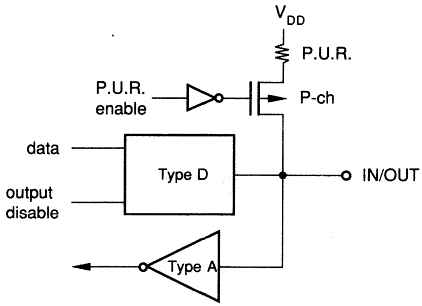
This is a Schmitt trigger input with hysteresis characteristics.

(4) Type D (Part of Type E–B, F–A)



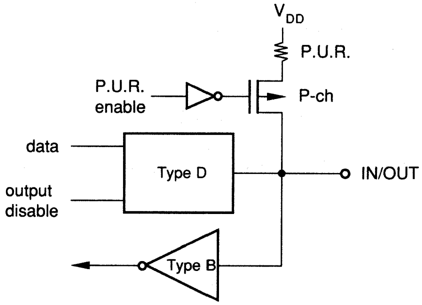
This is a push-pull output that can be set to high impedance (with both P-ch and N-ch off).

(5) Type E–B



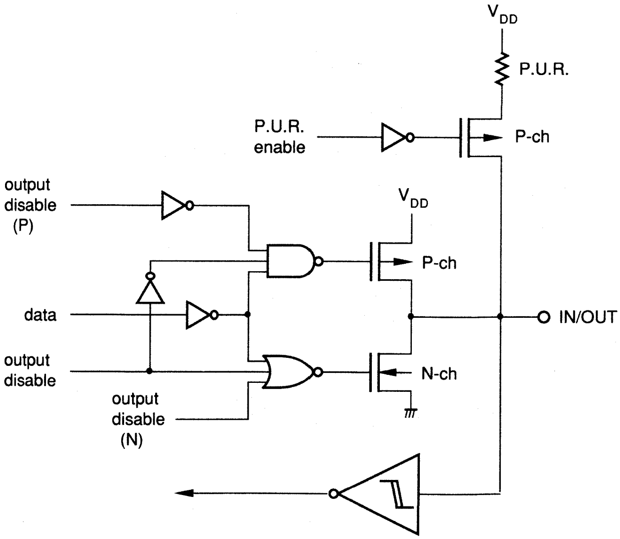
P.U.R. : Pull-Up Resistor

(6) Type F–A



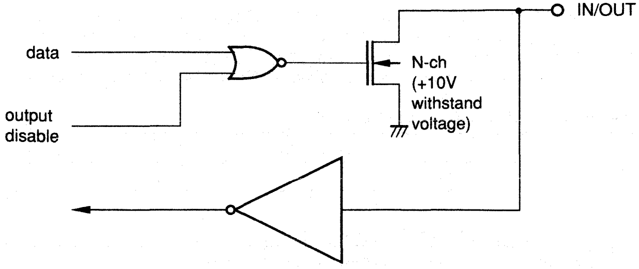
P.U.R. : Pull-Up Resistor

(7) Type F–B



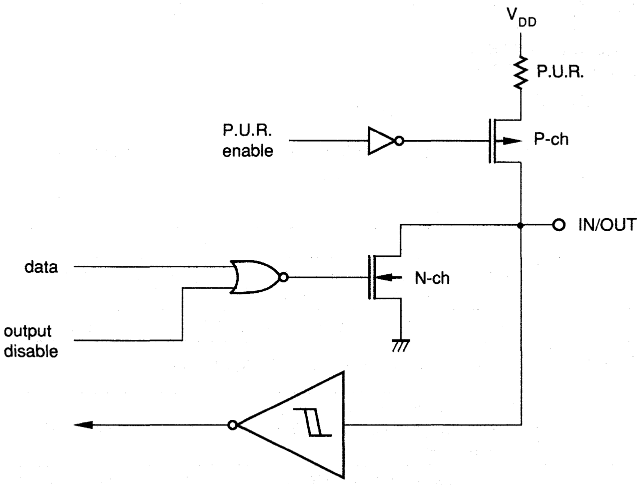
P.U.R. : Pull-Up Resistor

(10) Type M–B



Middle-high voltage
input buffer
(+10V withstand voltage)

(11) Type M–C



P.U.R. : Pull-Up resistor

### 10.6 Differences between µPD75P008 and µPD75008

Since the µPD75P008 is the result of changing the program memory of the µPD75004/006/008 from mask ROM to an PROM which can be written, it differs only in program memory and mask option; the two devices are the same in CPU functions and internal hardware. Table 10.6-1 lists the differences between the µPD75P008 and µPD75008.
For details on the CPU functions and internal hardware, refer to the µPD004/006/008 documents and user's manual.

**Table 10.6-1 Differences between µPD75P008 and µPD75008**

| Item | | µPD75P008 | µPD75008 |
|---|---|---|---|
| Program memory | | PROM | Mask ROM |
| | | • 00000-1F7FH<br>• 8064 bits x eigth bits | |
| Pull-up resistor | Ports 0-3, 6-8 | Can be specified by using software. | |
| | Ports 4, 5 | None | Mask Option |
| Supply voltage range | | 5V ± 10% | 2.7V - 6.0V |
| Pin connection | | They differ in VPP pin and pins also used for other purpose. | |

### 10.7 EPROM WRITE AND VERIFICATION

Program memory contained in the µPD75P008.
The memory capacity is as follows:

- µPD75P008: 8064 words x 8 bits

The pins as listed in Table 10.7-1 are used to write and verify the PROM. No address is input; address is updated by inputting clock from the X1 pin instead.

**Table 10.7-1 Pin function**

| Pin name | Function |
|---|---|
| X1, X2 | Address update clock during when PROM is write/verification is input to the X1 pin.<br>Its inverted signal is input to the X2 pin. |
| MD0-MD3 | Operation mode selection pins during PROM write/verification. |
| P40-P43 (low-order four bits)<br>P50-P53 (high-order four bits) | 8-bit data input/output pins during PROM write/verified. |
| $V_{DD}$ | Supply voltage apply pin.<br>During the normal operation, 5 V ± 10 % is applied; during PROM write/verification, +6 V is applied. |
| $V_{PP}$ | Voltage apply pin during PROM write/ verification. (Normally, $V_{DD}$ potential) |

10.7.1 Operating Mode during PROM is Write/Verification

When +6 V and +12.5 V are applied to the $V_{DD}$ and $V_{PP}$ pins, respctively, of the μPD75P008 the PROM write/verify mode is entered. The operation mode is selected according to the input signals to the MD0-MD3 pins as listed in Table 10.7-2.

Pins not used in PROM function should be pulled to $V_{SS}$.

**Table 10.7-2 Operating Mode**

| Operating mode specification | | | | | | Operating mode |
|---|---|---|---|---|---|---|
| $V_{DD}$ | $V_{PP}$ | MD0 | MD1 | MD2 | MD3 | |
| +12.5 V | +6 V | H | L | H | L | Clear program memory address |
| | | L | H | H | H | Write mode |
| | | L | L | H | H | Verify mode |
| | | H | X | H | H | Program inhibit mode |

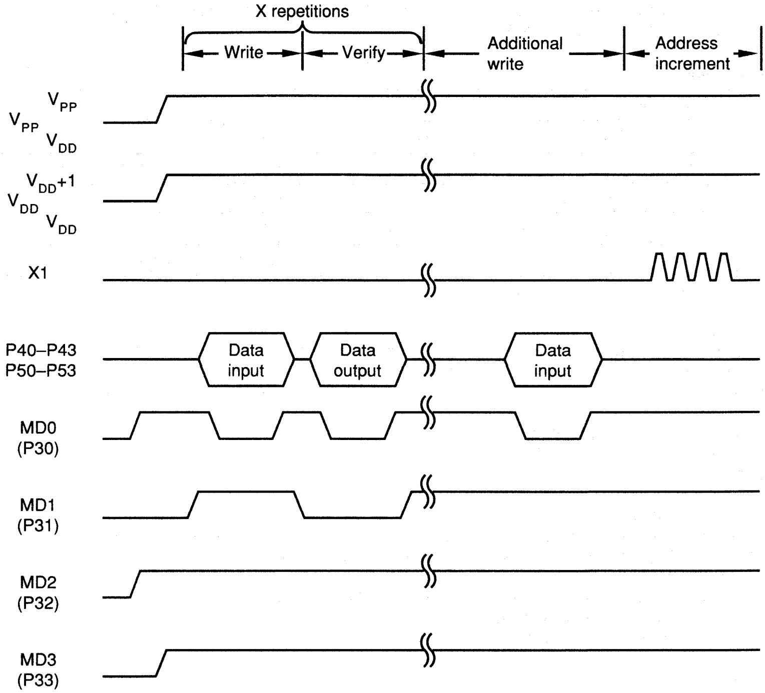
Remarks: X: L or H

10.7.2 PROM Write Procedure

PROM can be written at the high speed according to the following Procedure:

(1) Pull unused pins low to $V_{SS}$ with resistors. Set the X1 pin low.
(2) Supply 5 V to the $V_{DD}$ and $V_{PP}$ pins.
(3) Wait for 10 us.
(4) Selected the clear program memory address mode.
(5) Supply +6 V and +12.5 V to the $V_{DD}$ and $V_{PP}$ pins.
(6) Select the program inhibit mode.
(7) Write data in the 1-ms write mode.
(8) Select the program inhibit mode.
(9) Select the verify mode. If the data is written normally, proceed to (10). If it is not written normally, repeat (7) to (9).
(10) Supply X (number of (7)–(9) repetitions) x 1 ms program pulses (additional write).
(11) Select the program inhibit mode.
(12) Input four pulses to the X1 pin to update the program memory address by one.
(13) Repeat (7)–(12) until the end address is reached.
(14) Select the clear program memory address mode.
(15) Change the $V_{DD}$, $V_{PP}$ pin voltage to 5 V.
(16) Turn off the power.

Steps (2) to (12) are illustrated below:

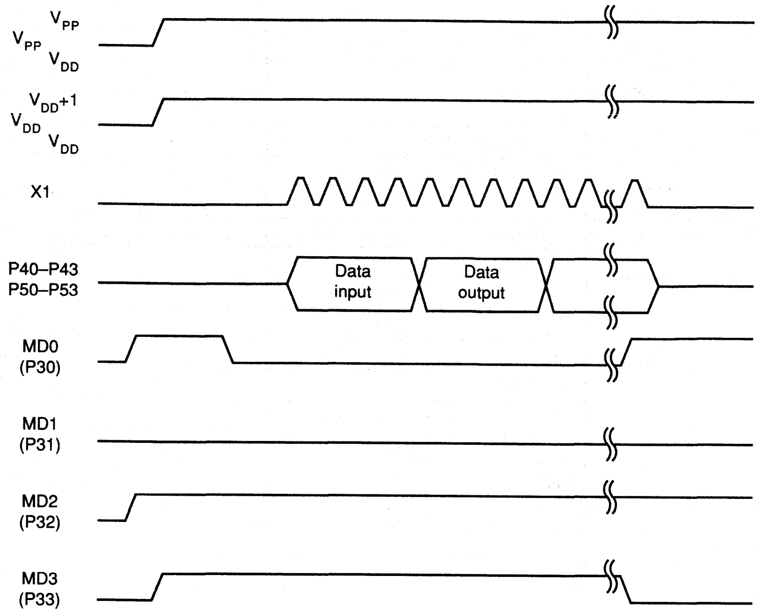


10.7.3 PROM Read Procdure

The PROM contents can be read according to the following procedure:

(1) Pull unused pins low to $V_{SS}$ with resistors.
Set the X1 pin low.
(2) Supply 5 V to the $V_{DD}$ and $V_{PP}$ pins.
(3) Wait for 10 μs.
(4) Select the clear program memory address mode.
(5) Supply + 6 V and + 12.5 V to the $V_{DD}$ and $V_{PP}$ pins.
(6) Select the program inhibit mode.
(7) Select the verify mode. Input four clock pulses to the X1 pin. The data are output from the memory addresses, one address at a time, in a cycle of hour clock pulses which are input to the X1 pin.
(8) Select the program inhibit mode.
(9) Select the clear program memory address mode.
(10) Change the $V_{DD}$, $V_{PP}$ voltage to 5 V.
(11) Turn off the power.

Steps (2) to (12) are illustrated below:



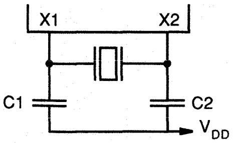
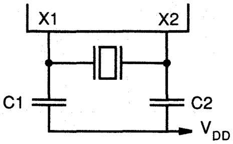
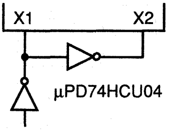
## 11. ELECTRICAL CHARACTARISTICS

**Absolute Maximum Rating** (Ta=25°C)

| Prameter | Symbol | Condition | | Rating | Unit |
|---|---|---|---|---|---|
| Supply voltage | $V_{DD}$ | | | -0.3 to +7.0 | V |
| | $V_{PP}$ | | | -0.3 to +13.5 | V |
| Input voltage | $V_{11}$ | Other than ports 4 and 5 | | -0.3 to $V_{DD}$ +0.3 | V |
| | $V_{12}$ | Ports 4 and 5 | Open drain | -0.3 to +11 | V |
| Output voltage | $V_0$ | | | -0.3 to $V_{DD}$ +0.3 | V |
| High-level output current | $I_{OH}$ | Single pin | | -10 | mA |
| | | All pins | | -30 | mA |
| Low-level output current | Note $I_{OL}$ | Ports 0, 3, 4 and 5 | Peak value | 30 | mA |
| | | | rms | 15 | mA |
| | | Other than ports 0, 3, 4 and 5 | Peak value | 20 | mA |
| | | | rms | 10 | mA |
| | | Total of ports 0, 3, 4, 5 and 8 | Peak value | 160 | mA |
| | | | rms | 120 | mA |
| | | Total of ports 2, 6 and 7 | Peak value | 66 | mA |
| | | | rms | 33 | mA |
| Operating temperature | $T_{opt}$ | | | -10 to +70 | °C |
| Storage temperature | $T_{stg}$ | | | -65 to +150 | °C |

Note : Use the following formula to calculate the effective value:
(Effective) value = (Peak value) x $\sqrt{\text{duty cycle}}$

**Main system clock Oscillator Characteristics**
(Ta = -10 to +70°C, $V_{DD}$ = 5V+/-10%)

| Resonator | Recommended constants | Item | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Note 3<br>Ceramic resonator |  | Note 1<br>Oscillation frequency ($f_{XX}$) | | 1.0 | | 5.0 | MHz |
| | | Note 2<br>Oscillation stabilization time | When $V_{DD}$ reaches the minimum oscillator operating voltage. | | | 4 | ms |
| Crystal resonator |  | Note 1<br>Oscillation frequency ($f_{XX}$) | | 1.0 | 4.19 | 5.0 | MHz |
| | | Note 2<br>Oscillation stabilization time | | | | 10 | ms |
| External clock |  | Note 1<br>X1 input frequency ($f_X$) | | 1.0 | | 5.0 | MHz |
| | | X1 input high/ low level width ($t_{XH}$ / $t_{XL}$) | | 100 | | 500 | ns |

Note 1: The oscillation frequency and X1 input frequency values indicated in this table express only the characteristics of the oscillator. Therefore, refer to the AC characteristics for the instruction execution time.

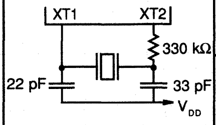
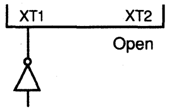
2: Oscillation stabilization time is defined as the time needed for the oscillator to stabilize after $V_{DD}$ is applied or after the STOP mode is released.

3: The following ceramic and crystal oscillators are recommended for the μPD75P008.

**Recommended Ceramic Resonators**

| Manufacture | Product | External capacitance (pF) | | Operating voltage range (V) | |
|---|---|---|---|---|---|
| | | C1 | C2 | MIN. | MAX. |
| MURATA | CSA 2.00MG | 30 | 30 | 4.75 | 5.25 |
| | CSA 4.19MG | 30 | 30 | 4.75 | 5.25 |
| | CSA 4.19MGU | 30 | 30 | 4.75 | 5.25 |
| | CST 4.19MG | 30pF (internally provided) | 30pF (internally provided) | 4.75 | 5.25 |

### Subsystem Clock Oscillator Characteristics
(Ta =-10 to +70°C, $V_{DD}$ =5V+/-10%)

| Resonator | Recommended constants | Item | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Crystal resonator |  | Oscillation frequency ($f_{XT}$) | | 32 | 32.768 | 35 | kHz |
| | | Oscillation stabilization time | | | 1.0 | 2 | s |
| External clock |  | XT1 input frequency ($f_{XT}$) | | 32 | | 100 | kHz |
| | | XT1 input high/ low level width ($t_{XTH}$/$t_{XTL}$) | | 5 | | 15 | μs |

**Capacitance** (Ta =25°C, $V_{DD}$ =0V)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Input capacitance | $C_{IN}$ | f=1 MHz Voltage should not be applied to any pin except those used for measurements. | | | 15 | pF |
| Output capacitance | $C_{OUT}$ | | | | 15 | pF |
| Input/output capacitance | $C_{IO}$ | | | | 15 | pF |

**DC Characteristics** (Ta =-10 to +70°C, $V_{DD}$ =5V±10%)

| Parameter | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| High-level input voltage | $V_{IH1}$ | Port 2, 3 and 8 | | 0.7$V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH2}$ | Port 0, 1, 6, 7, $\overline{\text{RESET}}$ | | 0.8$V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH3}$ | Port 4, 5 Open-drain | | 0.7$V_{DD}$ | | 10 | V |
| | $V_{IH4}$ | X1, X2, XT1 | | $V_{DD}$-0.5 | | $V_{DD}$ | V |
| Low-level input voltage | $V_{IL1}$ | Port 2, 3, 4, 5 and 8 | | 0 | | 0.3$V_{DD}$ | V |
| | $V_{IL1}$ | Port 0, 1, 6, 7, $\overline{\text{RESET}}$ | | 0 | | 0.2$V_{DD}$ | V |
| | $V_{IL3}$ | X1, X2, XT1 | | 0 | | 0.4 | V |
| High-level output voltage | $V_{OH1}$ | Port 0, 2, 3, 6, 7, 8 | $I_{OH}$ =-1mA | $V_{DD}$-1.0 | | | V |
| High-level input leakage current | $I_{LIH1}$ | $V_{IN}$ = $V_{DD}$ | Other than indicated below | | | 3 | μA |
| | $I_{LIH2}$ | | X1, X2, XT1 | | | 20 | μA |
| | $I_{LIH3}$ | $V_{IN}$ = 10V | Ports 4, 5 (with open drain) | | | 20 | μA |
| Low-level input leakage current | $I_{LIL1}$ | $V_{IN}$ = 0V | Other than indicated below | | | –3 | μA |
| | $I_{LIL2}$ | | X1, X2, XT1 | | | –20 | μA |
| High level output leakage current | $I_{LOH1}$ | $V_{OUT}$ = $V_{DD}$ | Other than indicated below | | | 3 | μA |
| | $I_{LOH2}$ | $V_{OUT}$ = 10V | Ports 4 and 5 (with open drain) | | | 20 | μA |

**DC Characteristics** (Ta =-10 to +70°C, $V_{DD}$ =5V±10%)

| Parameter | Symbol | Condition | | MIN. | TYP. | MAX. | Uni |
|---|---|---|---|---|---|---|---|
| Low level input leakage current | $I_{LOL}$ | $V_{OUT}$ = 0V | | | | –3 | μA |
| Internal pull-up resistor | $R_{L1}$ | Ports 0, 1, 2, 3, 6, 7 (Except P00) $V_{IN}$ = 0V | | 15 | 40 | 80 | kΩ |
| Low level output leakage | $V_{OL1}$ | Ports 0, 2, 3, 4, 5, 6, 7, 8 | Ports 4, 5 $I_{OL}$ = 15mA | | 0.4 | 2.0 | V |
| | | | Port 3 $I_{OL}$ = 15mA | | 0.6 | 2.0 | V |
| | | | $I_{OL}$ = 1.6 mA | | | 0.4 | V |
| | $V_{OL2}$ | SB0, 1 Open drain | Open drain Pull-up resistor ≥ 1kΩ | | | 0.2$V_{DD}$ | V |
| Note 1 Supply current | $I_{DD1}$ | Note 3 4.19 MHz crystal oscillation C1=C2=22pF | Note 2 | | 5 | 15 | mA |
| | $I_{DD2}$ | | HALT mode | | 500 | 1500 | μA |
| | $I_{DD3}$ | Note 4 32kHz crystal oscillation | | | 350 | 1000 | μA |
| | | | HALT mode | | 35 | 100 | |
| | $I_{DD4}$ | XT1 =0V STOP mode | | | 0.5 | 20 | μA |

Note 1. The current drained through the internal pull-up resistors is not included.
2. When operated in the high-speed mode with the processor clock control register (PCC) set to 0011.
3. Including the subsystem clock power consumption.
4. When system clock control register (SCC) is set to 1001; main system clock`s oscillation stopped once sub-system clock in operation.

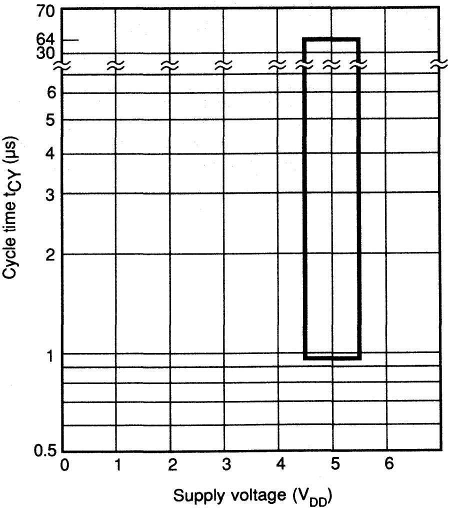
**AC Characteristics** (Ta =-10 to +70°C, $V_{DD}$ =5V±10%)
Operation other than serial transfer operation

| Parameter | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Note 1<br>Cycle time (minimum instruction execution time) | $t_{CY}$ | Operated with main system clock | 0.95 | | 64 | μs |
| | | Operated with sub-system clock | 114 | 122 | 125 | μs |
| TI0 input frequency | $f_{TI}$ | | 0 | | 1 | MHz |
| TI0 input high/low level width | $t_{TIH}$, $t_{TIL}$ | | 0.48 | | | μs |
| Interrupt input high/low level width | $t_{INTH}$, $t_{INTL}$ | INT0 | Note 2 | | | μs |
| | | KR0-7, INT1 2, 4 | 10 | | | μs |
| $\overline{\text{RESET}}$ low level width | $t_{RSL}$ | | 10 | | | μs |

Note 1. Cycle time (minimum instruction execution time) is determined by the oscillation frequency of the connected resonator, the system clock control register (SCC), and the processor clock control register (PCC). The figure to the right shows the $V_{DD}$ supply voltage vs. cycle time ($t_{CY}$) characteristic when operated with the main system clock.

2. Either $2t_{CY}$ or $64/f_{XX}$ can be selected by setting the interrupt mode register (IM0).

$t_{CY}$ vs $V_{DD}$
(With main system clock)



**Serial Transfer Operation**
**2-line/3-line serial I / 0 mode (SCK....Internal clock output)**

| Parameter | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY1}$ | | Output | 1600 | | | ns |
| $\overline{\text{SCK}}$ high / low level width | $t_{KL1}$<br>$t_{KH1}$ | | Output | $t_{KCL1}$/2<br>-50 | | | ns |
| SI set-up time (against $\overline{\text{SCK}}$↑) | $t_{SIK1}$ | | | 150 | | | ns |
| SI hold time (against $\overline{\text{SCK}}$↑) | $t_{KSI1}$ | | | 400 | | | ns |
| $\overline{\text{SCK}}$↓→ S0 output delay time | $t_{KSO1}$ | | | | | 250 | ns |

**2-line / 3-line serial I/0 mode (SCK....External clock input) :**

| Parameter | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY2}$ | | Input | 800 | | | ns |
| $\overline{\text{SCK}}$ high / low level width | $t_{KL2}$<br>$t_{KH1}$ | | Input | 400 | | | ns |
| SI set-up time (against $\overline{\text{SCK}}$↑ ) | $t_{SIK2}$ | | | 100 | | | ns |
| SI hold time (against $\overline{\text{SCK}}$ ↑ ) | $t_{KSI2}$ | | | 400 | | | ns |
| $\overline{\text{SCK}}$↓ → S0 output delay time | $t_{KSO2}$ | | | | | 300 | ns |

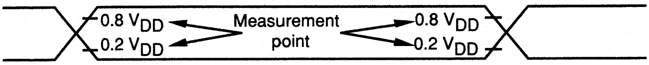
**SBI Mode (SCK....Internal clock output Master)) :**

| Parameter | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY3}$ | | 1600 | | | ns |
| $\overline{\text{SCK}}$ high / low level width | $t_{KL3}$<br>$t_{KH3}$ | | $t_{KCY}/2$<br>-50 | | | ns |
| $\overline{\text{SB0}}$, SB1 set-up time (against $\overline{\text{SCK}}$↑) | $t_{SIK3}$ | | 150 | | | ns |
| $\overline{\text{SB0}}$, SB1 hold time (against $\overline{\text{SCK}}$↑) | $t_{KSI3}$ | | $t_{KCY}/2$ | | | ns |
| $\overline{\text{SCK}}$↓ → SB0, SB1 output delay time | $t_{KSO3}$ | | 0 | | 250 | ns |
| $\overline{\text{SCK}}$↓ → SB0, SB1 ↓ | $t_{KSB}$ | | $t_{KCY}$ | | | ns |
| SB0, SB1↓ → $\overline{\text{SCK}}$↓ | $t_{SBK}$ | | $t_{KCY}$ | | | ns |
| SB0, SB1 low level width | $t_{SBL}$ | | $t_{KCY}$ | | | ns |
| SB0, SB1 high level width | $t_{SBH}$ | | $t_{KCY}$ | | | ns |

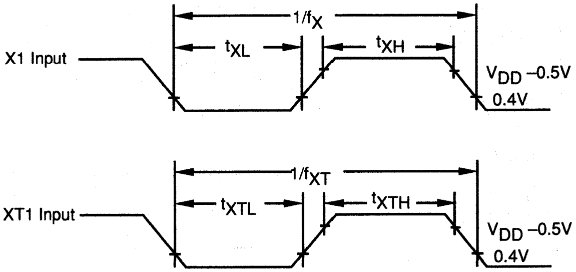
**SBI Mode (SCK.... External clock input (Slave)):**

| Parameter | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| SCK cycle time | $t_{KCY4}$ | | 800 | | | ns |
| SCK high / low level width | $t_{KL4}$<br>$t_{KH4}$ | | 400 | | | ns |
| $\overline{\text{SB0}}$, SB1 set-up time (against $\overline{\text{SCK}}$↑ ) | $t_{SIK4}$ | | 100 | | | ns |
| $\overline{\text{SB0}}$, SB1 hold time (against $\overline{\text{SCK}}$↑ ) | $t_{KSI4}$ | | $t_{KCY}/2$ | | | ns |
| $\overline{\text{SCK}}$ ↓ → SB0, SB1 output delay time | $t_{KSO4}$ | | 0 | | 300 | ns |
| $\overline{\text{SCK}}$↑ → SB0, SB1 ↓ | $t_{KSB}$ | | $t_{KCY}$ | | | ns |
| SB0, SB1↓ → $\overline{\text{SCK}}$ ↓ | $t_{SBK}$ | | $t_{KCY}$ | | | ns |
| SB0, SB1 low level width | $t_{SBL}$ | | $t_{KCY}$ | | | ns |
| SB0, SB1 high level width | $t_{SBH}$ | | $t_{KCY}$ | | | ns |

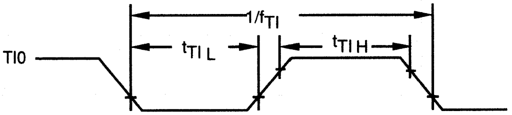
AC Timing Measurement Points (Except X1, XT1 input)



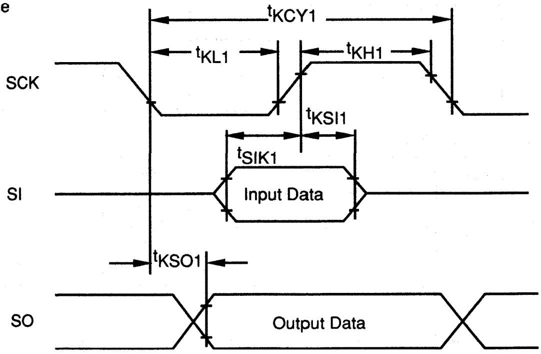
Clock Timing



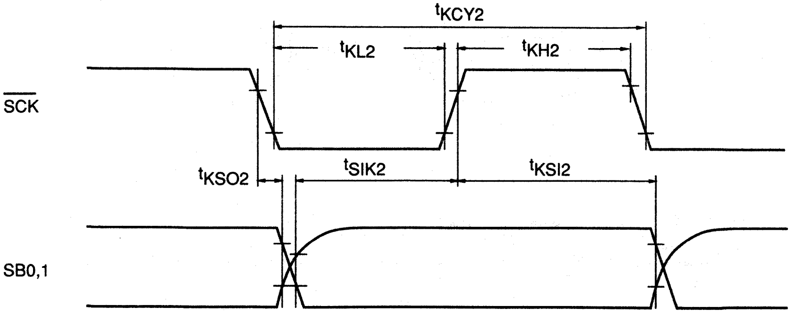
TIO Timing



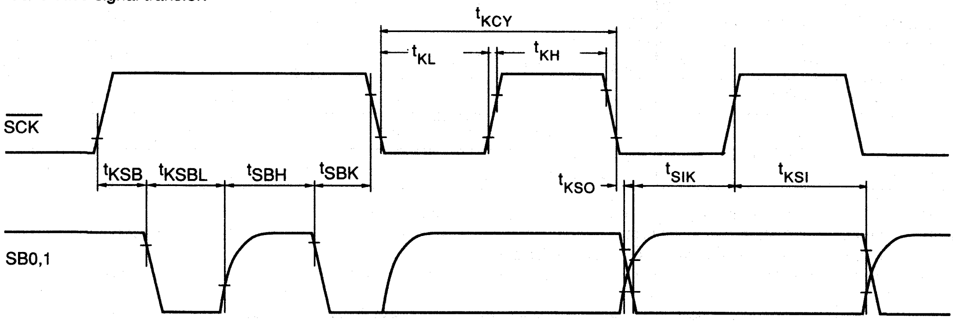
Serial Transfer Timing
3-line serial I/O mode



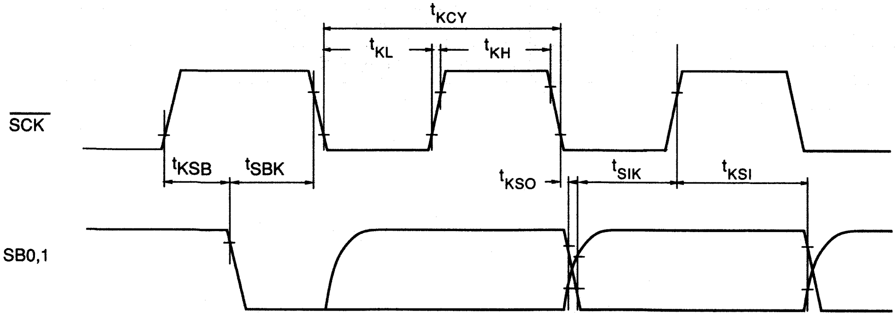
2-line serial I/O mode



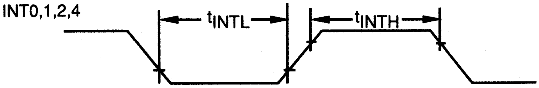
**Serial Transfer Timing** (SBI mode)
Bus release signal transfer:



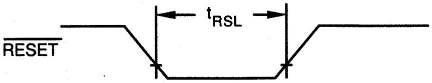
Command signal transfer:



Interrupt input timing



$\overline{\text{RESET}}$ input timing



**Data Memory STOP Mode Low Supply Voltage Data Retention Characteristic** (Ta = –10 to +70 °C)

| Parameter | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Data retention supply voltage | $V_{DDDR}$ | | 2.0 | | 6.0 | V |
| Data retention supply current (Note1) | $I_{DDDR}$ | $V_{DDDR}$ = 2.0V | | 0.1 | 10 | μA |
| Release signal set time | $t_{SREL}$ | | 0 | | | μs |
| Oscillation stabilization wait time (Note 2) | $t_{WAIT}$ | Release by $\overline{\text{RESET}}$ input | | $2^{17}/f_X$ | | ms |
| | | Release by interrupt request | | Note 3 | | ms |

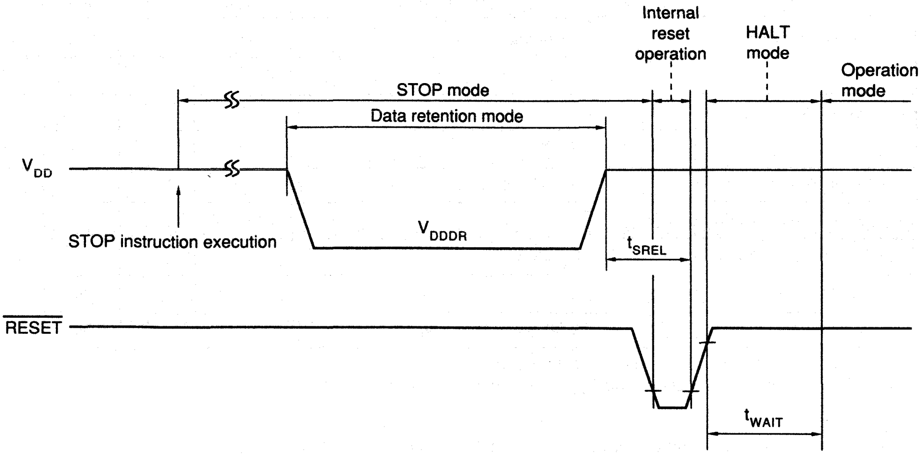
Note 1: The current drained through the internal pull-up resistor is not included

Note 2: The oscillation stabilization wait time is used to prevent unstable CPU operation at the beginning of oscillator operation, during which CPU operation is disabled.

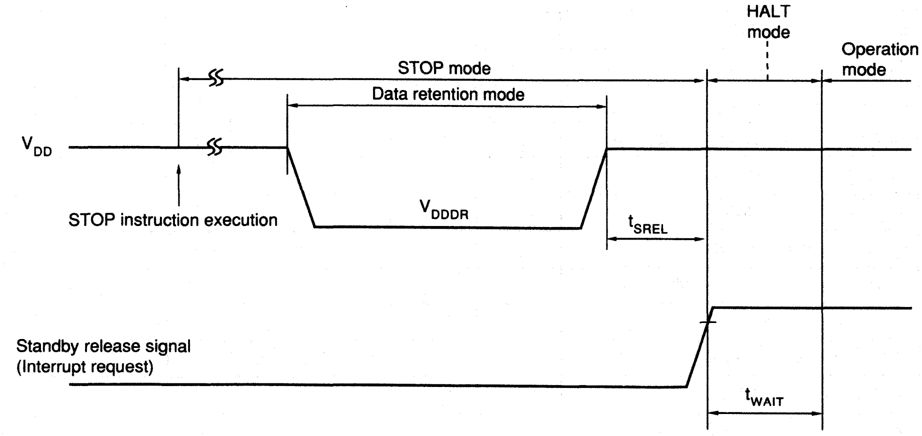
Note 3: This value depends on the setting of the basic interval timer mode register (BTM). (Refer to the table below. )

| BTM3 | BTM2 | BTM1 | BTM0 | WAIT time Parantheses ( ) indicates $f_X$ =4.19MHz |
|---|---|---|---|---|
| — | 0 | 0 | 0 | $2^{20}/f_X$ (Approximately 250 ms) |
| — | 0 | 1 | 1 | $2^{17}/f_X$ (Approximately 31.3 ms) |
| — | 1 | 0 | 1 | $2^{15}/f_X$ (Approximately 7.82 ms) |
| — | 1 | 1 | 1 | $2^{13}/f_X$ (Approximately 1.95 ms) |

Data Retention Timing (Releasing STOP mode by $\overline{\text{RESET}}$ )



Data Retention Timing
(Standby release signal: Releasing STOP mode by interrupt signal)



**DC Programming Characteristics** (Ta = 25 °C, $V_{DD}$ = 6.0 ± 0.25V, $V_{PP}$ = 12.5 ± 0.3V, $V_{SS}$ = 0V)

| Item | Symbol | Test Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| High level input voltage | $V_{IH1}$ | Other than X1, X2 | $0.7V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH2}$ | X1, X2 | $V_{DD}$ –0.5 | | $V_{DD}$ | V |
| Low level input voltage | $V_{IL1}$ | Other than X1, X2 | 0 | | 0.3 $V_{DD}$ | V |
| | $V_{IL2}$ | X1, X2 | 0 | | 0.4 | V |
| Input leak current | $I_{LI}$ | $V_{IN} = V_{IL}$ or $V_{IH}$ | | | 10 | μA |
| High level output voltage | $V_{OH}$ | $I_{OH}$ = –1 mA | $V_{DD}$ –1.0 | | | V |
| Low level output voltage | $V_{OL}$ | $I_{OL}$ = 1.6 mA | | | 0.4 | V |
| $V_{DD}$ power supply current | $I_{DD}$ | | | | 30 | mA |
| $V_{PP}$ power supply current | $I_{PP}$ | MD0 = $V_{IL}$, MD1 = $V_{IH}$ | | | 30 | mA |

Note 1: Ensure that $V_{PP}$ does not exceed +13.5V including overshoot.
Note 2: Ensure that $V_{DD}$ is applied before $V_{PP}$, and is turned off after $V_{PP}$.

**AC Programming Characteristics** (Ta = 25 °C, $V_{DD}$ = 6.0 ± 0.25V, $V_{PP}$ = 12.5 ± 0.3V, $V_{SS}$ = 0V)

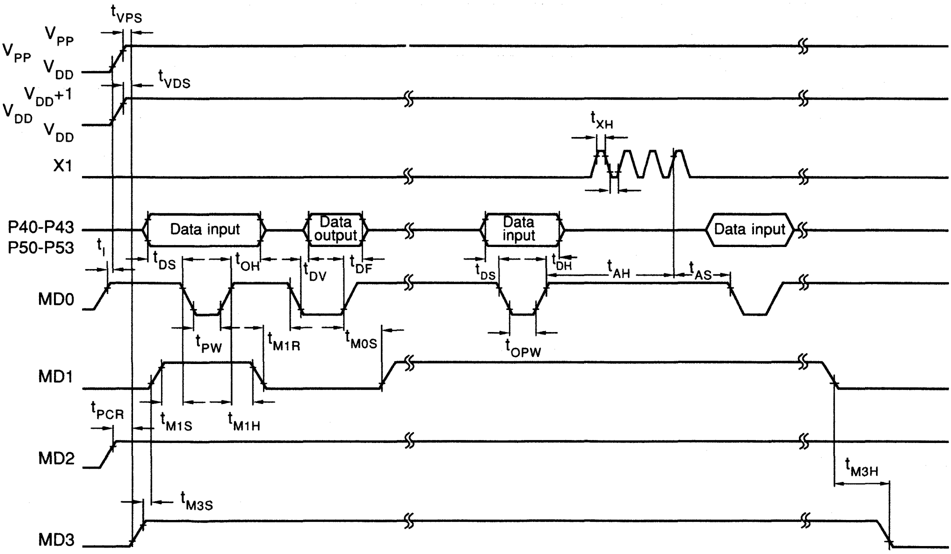
| Item | Symbol | Note 1 | Test Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Address setup time (Note 2) (to MD0 ↓ ) | $t_{AS}$ | $t_{AS}$ | | 2 | | | μs |
| MD1 setup time (to MD0 ↓ ) | $t_{M1S}$ | $t_{OES}$ | | 2 | | | μs |
| Data setup time (to MD0 ↓ ) | $t_{DS}$ | $t_{DS}$ | | 2 | | | μs |
| Address hold time (Note 2) (to MD0 ↑ ) | $t_{AH}$ | $t_{AH}$ | | 2 | | | μs |
| Data hold time (to MD0 ↑ ) | $t_{DH}$ | $t_{DH}$ | | 2 | | | μs |
| MD0 ↑ → data output float delay time | $t_{DF}$ | $t_{DF}$ | | 0 | | 130 | ns |
| $V_{PP}$ setup time (to MD3 ↑ ) | $t_{VPS}$ | $t_{VPS}$ | | 2 | | | μs |
| $V_{DD}$ setup time (to MD3 ↑ ) | $t_{VDS}$ | $t_{VCS}$ | | 2 | | | μs |
| Initial program pulse width | $t_{PW}$ | $t_{PW}$ | | 0.95 | 1.0 | 1.05 | ms |
| Additional program pulse width | $t_{OPW}$ | $t_{OPW}$ | | 0.95 | | 21.0 | ms |
| MD0 setup time (to MD1 ↑ ) | $t_{M0S}$ | $t_{CES}$ | | 2 | | | μs |
| MD0 ↓ → data output delay time | $t_{DV}$ | $t_{DV}$ | MD0 = MD1 = $V_{IL}$ | | | 1 | μs |
| MD1 hold time (to MD0 ↑ ) | $t_{M1H}$ | $t_{OEH}$ | $t_{M1H} + t_{M1R} \geq 50\mu s$ | 2 | | | μs |
| MD1 recovery time (to MD0 ↓ ) | $t_{M1R}$ | $t_{OR}$ | | 2 | | | μs |
| Program Counter reset time | $t_{PCR}$ | — | | 10 | | | μs |
| X1 input high/low level width | $t_{XH}$,$t_{XL}$ | — | | 0.125 | | | μs |
| X1 input frequency | $f_X$ | — | | | | 4.19 | MHz |
| Initial mode setting time | $t_I$ | — | | 2 | | | μs |
| MD3 setup time (to MD1 ↑ ) | $t_{M3S}$ | — | | 2 | | | μs |

**AC Programming Characteristics** (Ta = 25 °C, $V_{DD}$ = 6.0 ± 0.25V, $V_{PP}$ = 12.5 ± 0.3V, $V_{SS}$ = 0V) (cont'd)

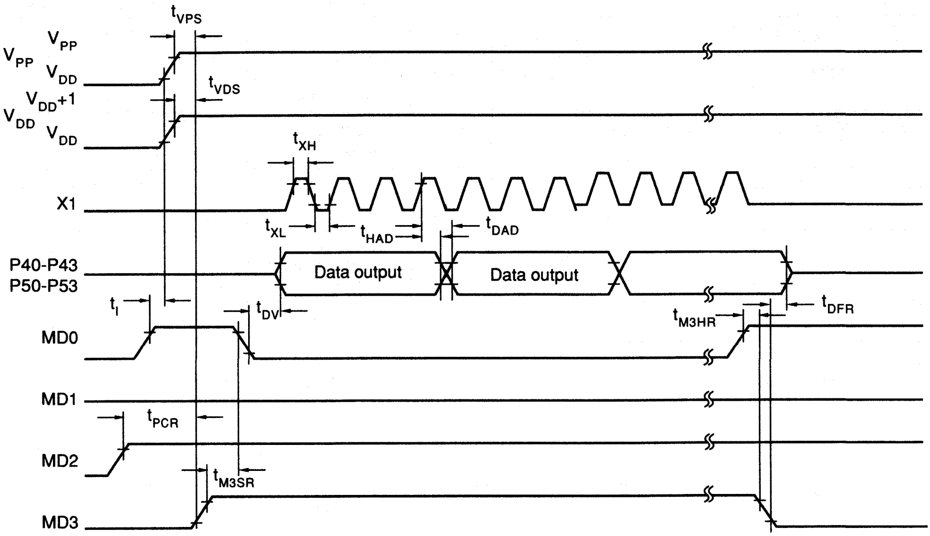
| Item | Symbol | Note 1 | Test Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| MD3 hold time (to MD1 ↓ ) | $t_{M3H}$ | — | | 2 | | | μs |
| MD3 setup time (to MD0 ↓ ) | $t_{M3SR}$ | — | For program memory read | 2 | | | μs |
| Address (Note 2) data output delay time | $t_{DAD}$ | $t_{ACC}$ | For program memory read | 2 | | | μs |
| Address (Note 2) data output hold time | $t_{HAD}$ | $t_{OH}$ | For program memory read | 0 | | 130 | ns |
| MD3 hold time (to MD0 ↑ ) | $t_{M3HR}$ | — | For program memory read | 2 | | | μs |
| MD3 ↓ → data output float delay time | $t_{DFR}$ | — | For program memory read | 2 | | | μs |

Note 1: Symbol for corresponding μPD27C256.
Note 2: The internal address signal is incremented by 1 by the rise of the 4th X1 input, and is not connected to a pin.

Program Memory Write Timing



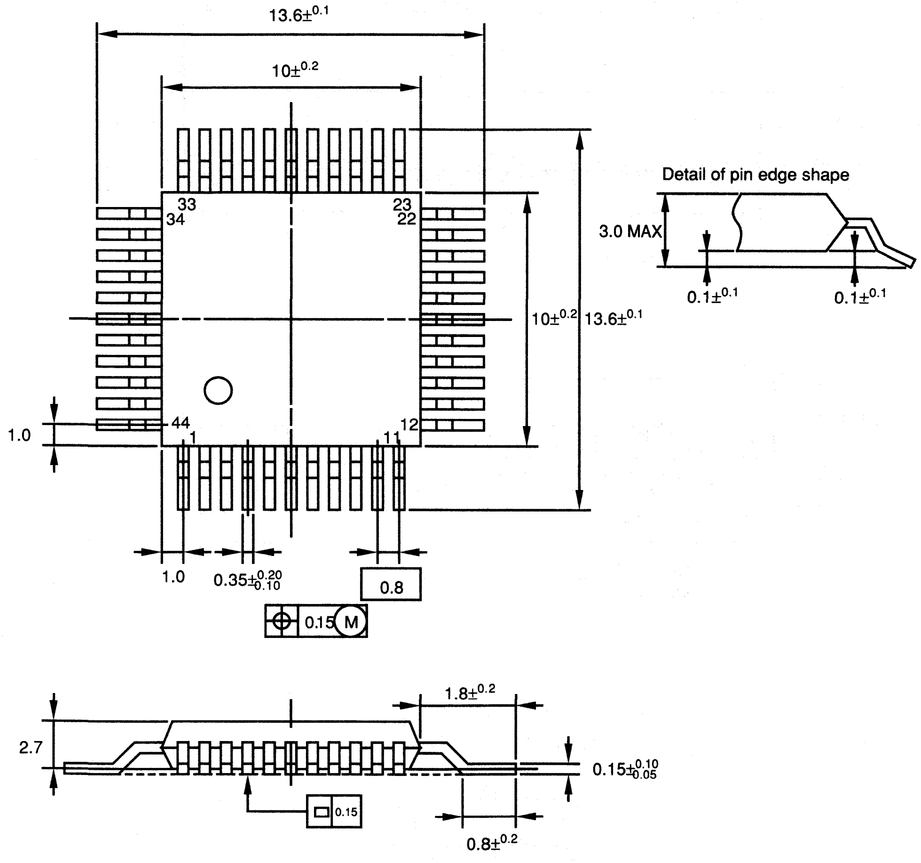
Program Memory Read Timing



### 11.1 Package Information

Dimension of 44-pin plastic flat package (units: mm)

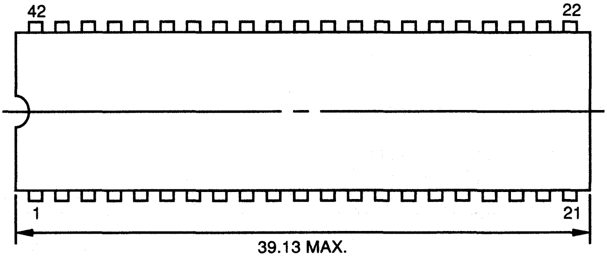
Top view

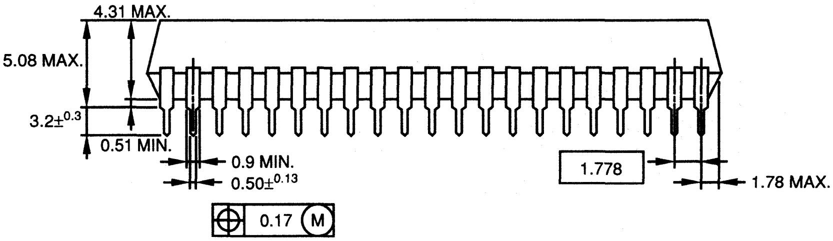


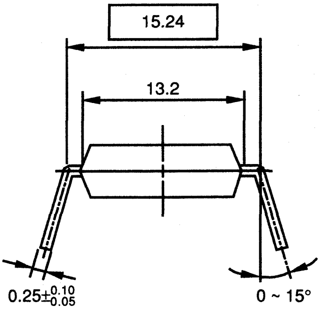
P44GB-80-B4

42-pin plastic shrink DIP (600 mil)

Top view







## CHAPTER 2 µPD75268 STANDARD MICROCOMPUTER WITH FIP CONTROLLER/DRIVER

### 1. OVERVIEW

The µPD75268 is a 4-bit single-chip microprocessor with a fluorescent indicator lamp (FIP) controller/driver built in.
The product has a 512 x 4 bit RAM, and a FIP controller/driver for controlling 9 to 16 segments.
The standard µCOM-75X instruction set, which is compatible with µPD75008 software, is used.
The µPD75268 is a high-speed microcomputer with an FIP display function, and consumes less power. The µPD75268, as a VTR timer/tuner microcomputer, is suitable for low-end to high-end VTRs. In addition, the microcomputer can find a wide range of applications:

#### 1.1 Features

- Audio equipment (such as CD players)
- Electronic cash register (ECR)
- Telephon set
- Electronic balance
- Automobile

2

#### 1.2 Ordering Information

| Nomenclature | Package |
|---|---|
| µPD75268CW-xxx | 64-pin plastic shrinked-dual-in-line-package |
| µPD75268GF-xxx-3BE | 64-pin plastic quad- flat package |

Remarks: xxx represents a ROM code number.

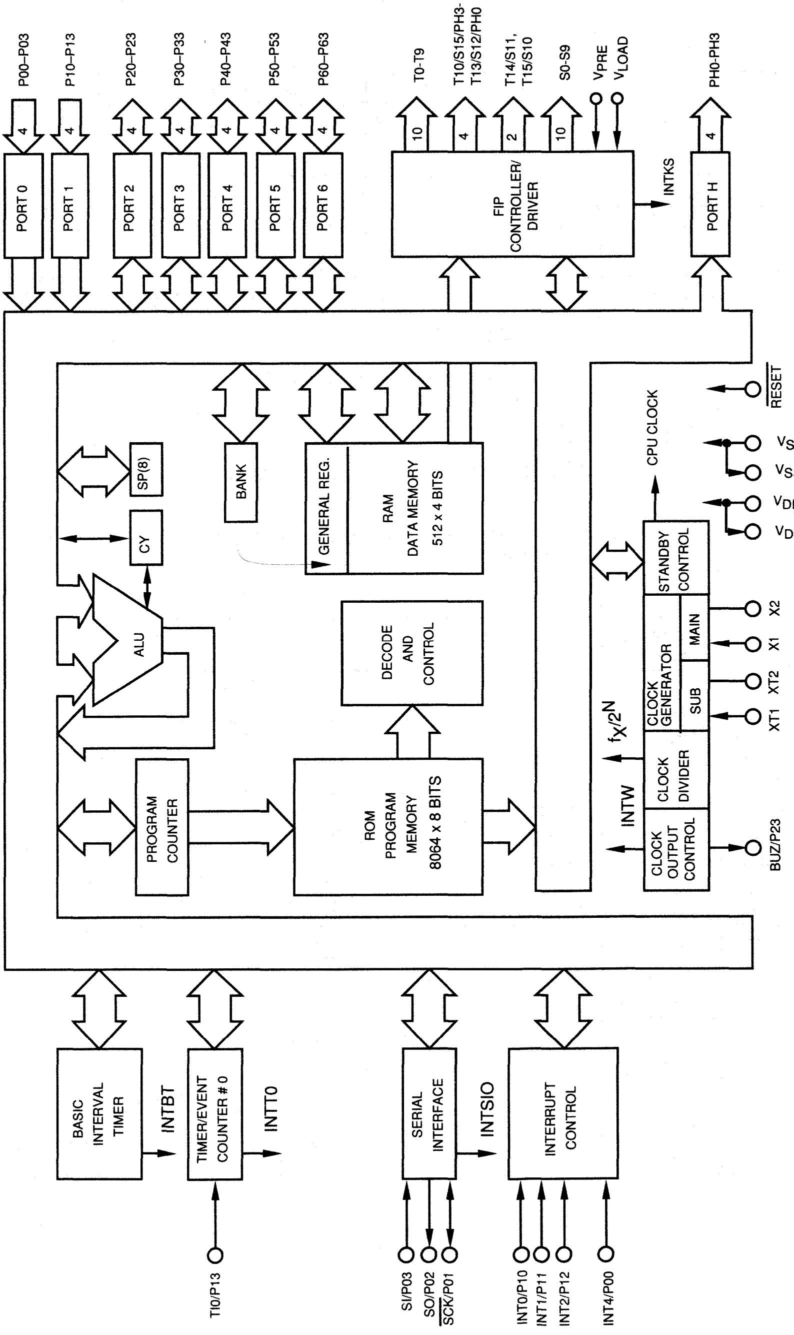
### 1.3 List of Functions

| Item | Function |
|---|---|
| Internal memory Capacity | ROM: 8064 x 8 bits, Ram: 512 x 4 bits (µPD75268) |
| I/O line | 54 lines<br>• Input : 8 lines<br>• I/O: 20 lines (LED drive: 8 lines)<br>• High-voltage output: 26 lines (40 $V_{max}$) |
| Instruction cycle | • 0.95 µs, 1.91 µs, 15.3 µs/4.19 MHz<br>• 122 µs/32.768 kHz |
| FIP controller/driver | • Number of segments: 9 to 16 segments<br>• Number of digits: 9 to 16 digits<br>• Dimmer function: 8 stages<br>• Pull-down resistor mask option<br>• Key scan interrupt generation |
| Timer | 3 channels<br>• Watch timer: With buzzer function<br>• Timer/event counter<br>• Basic interval timer: Usable as a watchdog timer |
| Serial interface | • Allows MSB/LSB inversion for starting transfer.<br>• Allows a serial bus configuration. |
| Interupt | • Allows multiple hardware interrupts.<br>• External interrupts: 3<br>• Detection of both edges<br>• Detection edge programmable (with noise eliminator)<br>• Detection edge programmable<br>• External test input: 1 (Rising edge detection)<br>• Internal interrupts 4<br>• Timer/event counter<br>• Basic interval timer<br>• Serial interface<br>• For key scan<br>• Internal test input: 1 (watch timer) |
| System clock generator | Two build-in circuits<br>• When main system clock is used: 4.194304 MHz standard<br>• When subsystem clock is used: 32.768 kHz |
| Mask option | • High-voltage port (pull-down resistor)<br>• Port 6 (pull-down resistor) |
| Operation temperature | -40 to +85°C |
| Operating power supply voltage | 2.7 to 6.0 V (standby data preservation: 2.0 to 6.0 V) |
| Package | • 64-pin plastic shrinked-dual-in-line package<br>• 64-pin plastic quad-flat package |

### 1.4 Functions of the Products in the Family

| Name | | μPD75206 | μPD75208 | μPD75268 | μPD75212A | μPD75216A | μPD75P216A |
|---|---|---|---|---|---|---|---|
| Internal memory | ROM | 6016 x 8 | 8064 x 8 | | 12160 x 8 | 16256 x 8 | 16256 x 8 (One-time PROM) |
| | RAM | 369 x 4 | 497 x 4 | 512 x 4 | | | |
| General-purpose regisger | | 4 bits x 8 x 4 banks | | 4 bits x 8 x 1 bank | 4 bits x 8 x 4 banks | | |
| I/0 line | Input only | 8 lines: Input ports (used as serial inputs, timer input, and interrupts as well) | | | | | |
| | I/O | 20 lines: 8 LED drive lines | | | | | |
| | | Port6: With a pull-down resistor (mask option) | | | | | Port 6: Without pull-down resistor |
| | Output only | 1 line: PPO | | None | 1 line: PPO | | |
| | Hight-voltage output | 26 lines: 40 $V_{MAX.}$ | | | | | |
| | | Incorporation of a pull-down resistor or open drain output is selected bit by bit (mask option). | | | | | S0-S8,T0-T9: With a pull-down resistor<br>S9, T10- T15: Open drain output |
| FIP controller/driver | Number of segments | 9 to 12 segments | | 9 to 16 segments | | | |
| | Number of digits | 9 to 16 digits | | | | | |
| Timer /pulse gemeratpr (with 14-bit PWM output capability | | Available | | Not available | Available | | |
| Serial interface | | • Allows MSB/LSB inversion for starting transfer.<br>• Allows a serial bus configuration. | | | | | |
| Interrupt sources | | Extnl: 4<br>Intnl: 6 | | Extnl: 4<br>Intnl: 5 | Extnl:4<br>Intnl: 6 | | |
| Interrupt priority specification | | Available | | Not Available | Available | | |
| Instruction set | Class | High-end | | Standard | High-end | | |
| | 1-bit transfer | Available | | Not available | Available | | |
| | 8-bit transfer | | | | | | |
| | @HL+, @HL-, Indirect branch instruction | | | | | | |
| Power-on reset circuit | | Mask option | | Not available | Mask option | | Not available |
| Power supply voltage | | 2.7 to 6.0 V | | | | | 5 V ± 10% |

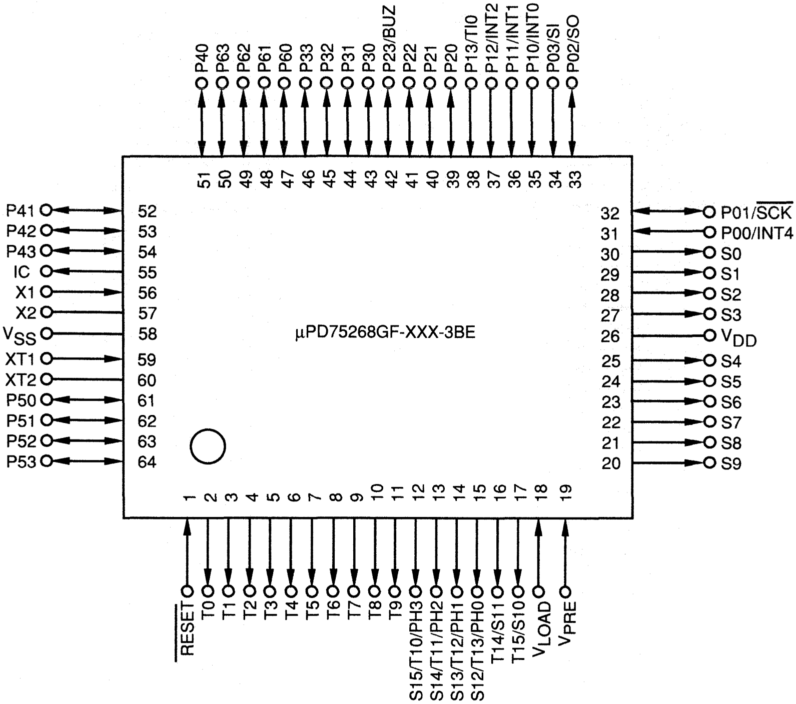
## 1.5 Block Diagram



### 1.6 Pin Configuration

64 Pin Flat Pack μPD75268

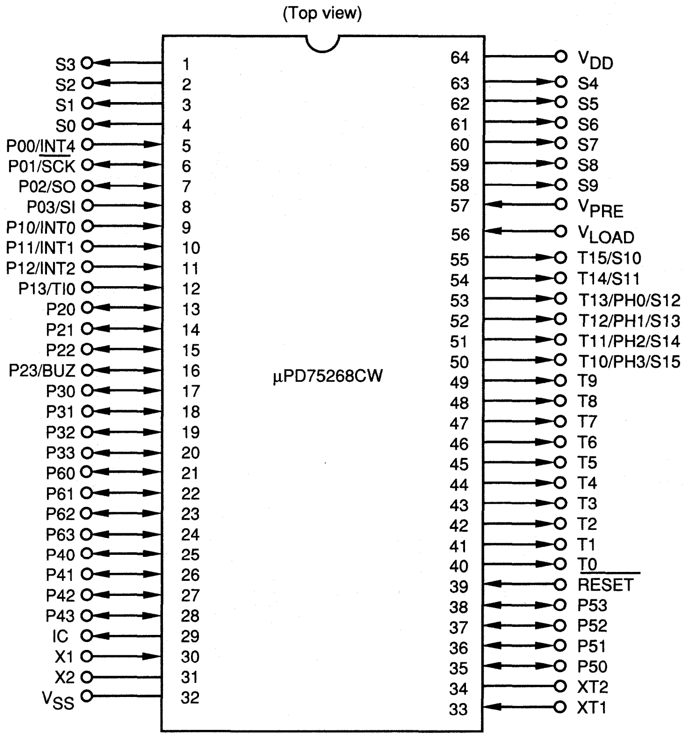
Top view



| | | | |
|---|---|---|---|
| | | $V_{SS}$ | : GND |
| | | SI | : Serial Data input |
| P00–P03 | : Port0 | $\overline{\text{SCK}}$ | : Serial Clock Input/Output |
| P10–P13 | : Port1 | $V_{DD}$ | : Positive Power Supply |
| P20–P23 | : Port2 | SO | : Serial Data Output |
| P30–P33 | : Port3 | INT0, INT1 | : Interrupt Inputs |
| P40–P43 | : Port4 | INT2 | : Interrupt Input |
| P50–P53 | : Port5 | INT4 | : Interrupt Input |
| P60–P63 | : Port6 | TI0 | : Input External Event Pulse |
| PH0-PH3 | : PortH | X1, X2 | : Main System Clock Crystal |
| T0-T15 | : Digits | XT1, XT2 | : Subsystem Clock Crystal |
| S0-S11/S15 | : Segments | $\overline{\text{RESET}}$ | : Reset |
| IC | : (*) | $V_{PRE}$ | : Power Supply for High-Voltage Output Predriver |
| BUZ | : 2kHz Buzzer Output | $V_{LOAD}$ | : Power Supply for High-Voltage Output Driver |

(*) Internal connected

64 Pin Shrink DIP μPD75268CW



## 2. PIN FUNCTIONS

2.1.1 P00–P03 (Port 0) . . . Inputs shared by INT4, $\overline{\text{SCK}}$, SI, and SO
P10–P13 (Port 1) . . . Inputs shared by INT0, 1, 2, and TI0

These are 4–bit input ports for Ports 0 and 1.
Port 0, in addition to input port function, also provides vector interrupt input (INT4) and serial interface input/output ($\overline{\text{SCK}}$, SI, SO) functions. Port 1 also functions as vector interrupt input (INT0, INT1) and edge–detection test input (INT2). P10 and P11 have a noise eliminator.
The data on the Port 0 and Port 1 lines can always be input regardless of the currently selected pin functions.
The P00 (INT4), P01 ($\overline{\text{SCK}}$), and P03 (SI) inputs of Port 0 and all inputs of Port 1 have Schmitt trigger inputs.

2.1.2 P20–P23 (Port 2), P30–P33 (Port 3),
P40–P43 (Port 4), P50–P53 (Port 5),
P60–P63 (Port 6)
. . . 3–State I/Os

These are 3–state, 4–bit CMOS input/output ports with output latches (for Ports 2 through 6). Ports 4 and 5 provide large–current output for direct LED drivability.
An internal reset signal ($\overline{\text{RES}}$) clears the output latches on all ports to zeros and places all pins in the Input mode (all outputs set to high impedance).
The Port Mode register is used to select either Input or Output mode. For Ports 3 and 6, Input/Output mode can be selectes for each bit. For Ports 2, 4, 5, mode selection is always in 4–bit units.
Ports 4 and 5, may be paired to allow input/output in 8–bit units.
Port 2, in addition to I/O port function, also functions as output (BUZ).
Port 6 has low $V_{IL}$ and can have on chip pull-down resistors as mask option, therefore it is best suited as a key input port.

2.1.3 PH0-PH3 (PORT H) . . . P-ch open drain high-voltage output pins also used for T10/S15-T13/S12

PH0-PH3 are P-ch open drain 4-bit high-voltage, high-current output port pins with output latch. High current output for direct LED drive. Internal pull-down resistor can also be included on the pins by mask option.
When the internal reset signal ($\overline{\text{RES}}$) is generated, the pins are placed in high impedance, but they will go low if the internal pull-down resistor is included.
In addition to the P-ch open drain output function, port H can also be used for high-voltage digit/segment output (T13/S15-T10/S12).

2.1.4 T0-T9 . . . P-ch open drain high-voltage output

T0-T9 are high-voltage, high-current output pins for FIP controller/driver timing (digit) signals.
An internal pull-down resistor can be included for each pin using mask option.
When the internal reset signal ($\overline{\text{RES}}$) is generated, the pins are placed in high impedance, but they will go low if the internal pull-down resistor is included.

2.1.5 T10/S15-T13/S12 . . . P-ch open drain high-voltage output pins also used for port H

T10/S15-T13/S12 are high-voltage, high-current output pins used for both FIP controller/driver digit and segment outputs.
The digit signal or segment signal can be selected for each pin. The timing signals are preferential, however, and the timing mode register is used for control.
An internal pull-down resistor van be included for each pin by mask option.
When the internal reset signal ($\overline{\text{RES}}$) is generated, the pins are placed in high impedance, but they will go low if the internal pull-down resistor is included.
The T10/S15-T13/S12 pins, when not used for digit/segment signal high voltage output, can also be used for digit/segment signal high-voltage output (PORT H).

### 2.1.6 T14/S11 and T15/S10 . . . P-ch open drain high-voltage output

T14/S11 and T15/S10 are high-voltage, high-current output pins for FIP controller/driver digit and segment outputs.
The digit signal or segment signal can be selected for each pin. The timing signals are preferential, however, and the timing mode register is used for control.
When not used for display, the pins can be used as a static output port.
An internal pull-down resistor can be used for each pin by mask option.
When the internal reset signal ($\overline{\text{RES}}$) is generated, the pins are placed in high impedance, but they will go low if the internal pull-down resistor is included.

### 2.1.7 S0-S9 . . . P-channel, open-drain, high-voltage outputs

These pins serve as high-voltage outputs to provide segment signals for the FIP controller/driver. Pin S9 can be used as a static output port when not used for display purposes. Each of these pins can be provided with a pull-down resistor as mask option.
An internal reset signal ($\overline{\text{RES}}$) sets each pin to high impedance if provided with no pull-down resistor, or to low level if provided with a pull-down resistor.

### 2.1.8 TI0 . . . Input, shared with P13

These input accept external event pulses for internal timer/event counter:
It may also be used as edge–detection vector interrupt input by setting the timer/event counter mode registers (TM0).
Since it is mapped on the data memory space, the pin status can be read or tested.
Both inputs have Schmitt trigger circuits.

### 2.1.9 BUZ . . . Output (multiplexed with Port 2)

This pin provides a fixed-frequency output (2.048kHz) usable for buzzer driving or trimming of the system clock frequency. It shares the pin function with pin P23, and is valid only if bit 7 (WM7) of the Clock Mode register (WM) is set to one.
An internal reset signal ($\overline{\text{RES}}$) resets WM7 to zero, which causes this pin to subsequently function as a general I/O port.

### 2.1.10 $\overline{\text{SCK}}$, SO, SI . . . 3–State I/O shared by Port 0

These are 3–state serial interface input/output pins.
They become effective by settings on the Serial Operation Mode register.
An internal reset signal ($\overline{\text{RES}}$) stops serial interface operation and restores the normal input port function for Port 0.
The $\overline{\text{SCK}}$ and SI pins have Schmitt trigger inputs.

### 2.1.11 INT0, INT1 . . . Interrupt inputs shared by Port 1

These are external vector interrupt inputs at Port 1.
Operation modes for these inputs can be specified with the Edge Detection Mode register from the following (see 5.3):
① Rising–edge active
② Falling–edge active
③ Both–edge active
④ External interrupt input disable
Both INT0 and INT1 are synchronous inputs with noise suppress circuits. The acceptable pulse differs depending on the CPU clock rate. This inputs are unable to clear the standby mode.
An internal reset signal ($\overline{\text{RES}}$) selects rising–edge active mode.
Both INT0 and INT1 pins have Schmitt trigger inputs.

### 2.1.12 INT2 . . . Test input shared by port 1

This is a rising–edge active, external test input at Port 1. An internal test flag is set when the signal applied to this input is set from Low to High. INT2 is an asynchronous input, which accept signals with a certain width of High level regardless of the CPU clock timing.
INT2 may also be used to clear the STOP or HALT mode. It has a Schmitt trigger input.

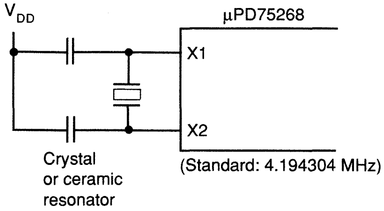
2.1.13 INT4 . . . Interrupt Input shared by Port 0

This is a both–edge (rising– and falling–edge active, external vector interrupt input). An internal Interrupt Request flag is set when the signal applied to this input is set from Low to High or vice versa.
INT4 is an asynchronous input, which accepts a signal with a certain High or Low level with no regard to the CPU clock timing. INT4 may also be used to clear the STOP or HALT mode. It has a Schmitt trigger input.

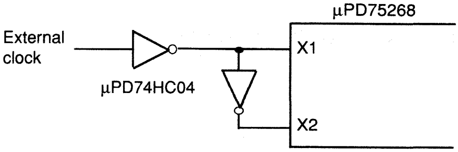
2.1.14 X1 and X2

X1 and X2 are connection pins for the main system clock crystal / ceramic oscillator.
External clocks can also be input.

(a) Crystal ceramic oscillator



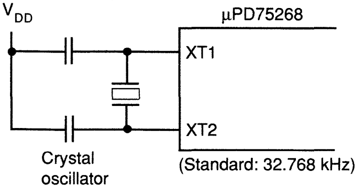
(b) External clock



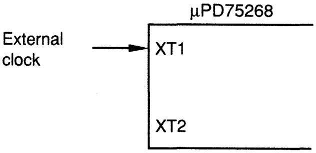
2.1.15 XT1 and XT2

XT1 and XT2 are crystal connection pins for the subsystem clock oscillation.
External clocks can also be input.

(a) Crystal oscillator



(b) External clock



### 2.1.16 $\overline{\text{RESET}}$

This is an active Low reset input. It is an asynchronous input, which accepts a signal with a certain Low level width with no regard to the system clock timing. A signal input to this pin causes an internal reset signal ($\overline{\text{RES}}$) to be generated, which provides a system reset overriding all other operations. In addition to CPU initialization and start, this input has a Schmitt trigger input.

### 2.1.17 $V_{PRE}$

This pin supplies power to the predriver for the FIP controller/driver.

### 2.1.18 $V_{LOAD}$

This pin supplies power to pull-down resistors which are provided (as a mask option) at the segment and timing output pins for the FIP controller/driver.

### 2.1.19 $V_{DD}$

This pin accepts a positive supply voltage for the device.

### 2.1.20 $V_{SS}$

$V_{SS}$ is the ground pin.

### 2.1.21 IC

This pin is internally connected. It must be open.

**Table 2.1-1 List of Digital I/O Pin Functions**

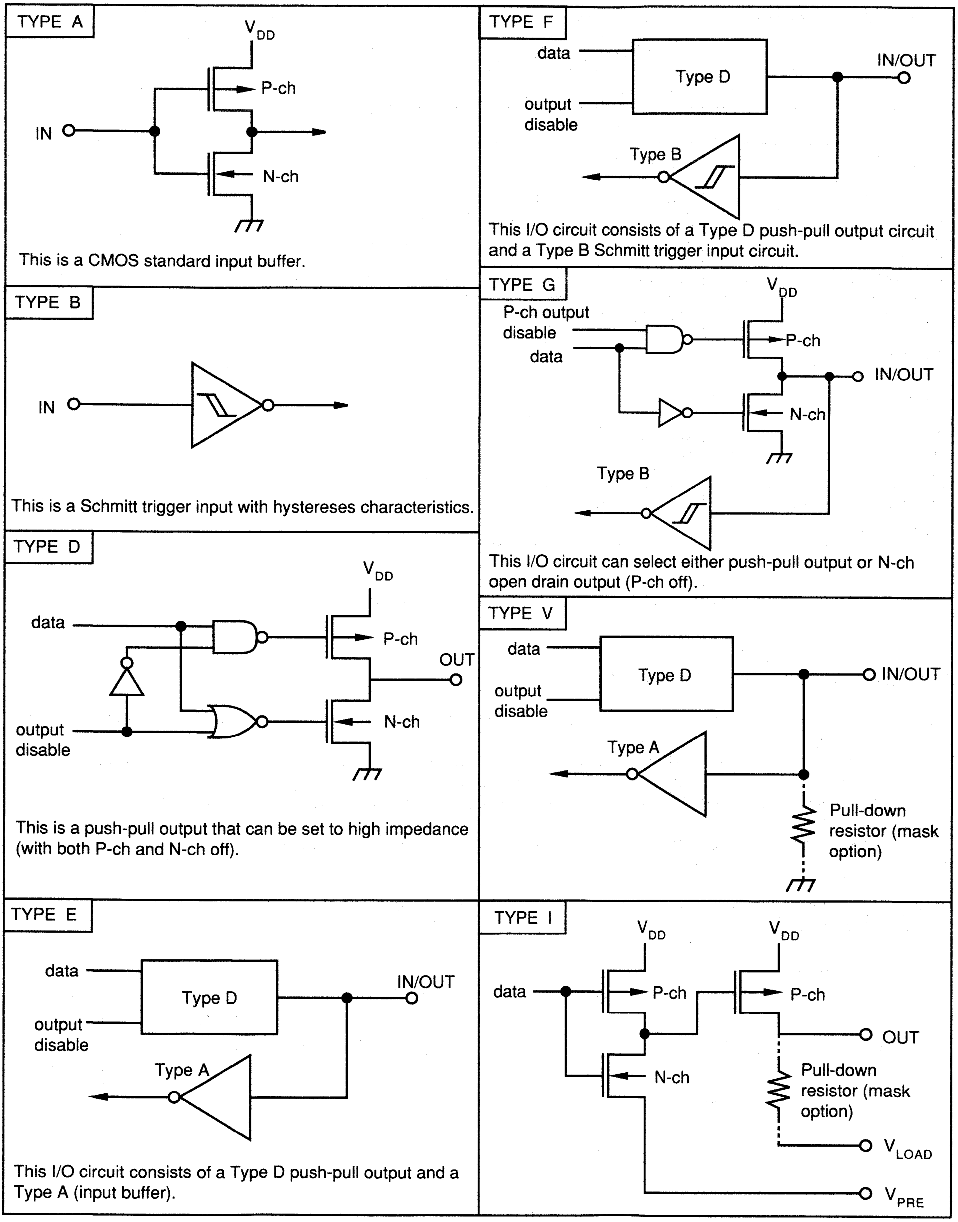
| Name | Input/output | Shared with | Function | 8-bit I/O | When reset | Circuit type |
|---|---|---|---|---|---|---|
| P00 | Input | INT4 | 4-bit input port (Port 0) | X | Input | Ⓑ *1 |
| P01 | Input/output | $\overline{\text{SCK}}$ | | | | Ⓕ *1 |
| P02 | Input/output | SO | | | | Ⓖ *1 |
| P03 | Input | SI | | | | Ⓑ *1 |
| P10 | Input | INT0 *2 | 4-bit input port (Port 1) | | Input | Ⓑ *1 |
| P11 | | INT1 *2 | | | | |
| P12 | | INT2 | | | | |
| P13 | | TI0 | | | | |
| P20 | Input/output | — | 4-bit input/output port (Port 2) | X | Input | E |
| P21 | | — | | | | |
| P22 | | — | | | | |
| P23 | | BUZ | | | | |
| P30 to P33 | Input/output | — | 4-bit programmable input/output port (Port 3). Input or output can be specified per bit. | | Input | E |
| P40 to P43 | Input/output | — | 4-bit input/output port (Port 4). Can directly drive LEDs. | O | Input | E |
| P50 to P53 | Input/output | — | 4-bit input/output port (Port 5). Can directly drive LEDs. | | Input | E |
| P60 to P63 | Input/output | — | 4-bit programmable input/output port (Port 6). Input or output can be specified per bit. Internal pull-down resistors provided (mask option). Suited for key input. | X | Input | V |
| PH0 | Output | T13/S12 | 4-bit P-ch open drain high-voltage, high-current output port (Port H). Can directly drive LEDs. Internal pull-down resistors can be built in (mask operation). | X | | I |
| PH1 | | T12/S13 | | | | |
| PH2 | | T11/S14 | | | | |
| PH3 | | T10/S15 | | | | |

*1: Indicates Schmitt trigger input.
*2: With noise eliminator circuit.

**Table 2.1-2 List of Other Pin Functions**

| Name | Input/output | Shared with | Function | | When reset | Circuit type |
|---|---|---|---|---|---|---|
| T0 to T9 | Output | — | FIP controller/driver output pins. Pull-down resistor can be built-in per bit (mask option) | High-voltage, high-current output pins for digit output. | Low level (when pull-down resistors are built-in) or high impedance (when no pull-down resistors built in) | I |
| T10/S15<br>T13/S12 | | PH3-PH0 | | High-voltage high-current output pins usable for digit/ segment output as well. Any unused pins can be used for PORT H. | | |
| T14/S11<br>T15/S10 | | — | | High-voltage, high-current output pins for segment output. Static output also possible. | | |
| S9 | | | | High-voltage output pin for segment  ouput. Static output also possible. | | |
| S0 to S8 | | | | High-voltage output to pins for segment output. | | |
| IC | — | — | Internally connected. It must be open. | | — | — |
| TI0 | Input | P13 | Inputs external event pulse to the timer/event counter. | | | Ⓑ |
| SCK | Input/output | P01 | Serial data input/ouptut pin. | | Input | Ⓕ |
| SO | Input/output | P02 | Serial data output pin or serial data input/output pin. | | | Ⓖ |
| SI | Input | P03 | Serial data input pin or normal input pin. | | | Ⓑ |
| INT4 | Input | P00 | Edge detect vectored interrupt input pin (both rising and falling edges can be detected). | | | Ⓑ |
| INT0 | Input | P10 | Edge detect vectored interrupt input pin with noise elimination circuit (either rising or falling edge can be detected). | | | Ⓑ |
| INT1 | | P11 | | | | |
| INT2 | Input | P12 | Edge detected testable input pin (rising edge detected). | | | Ⓑ |
| BUZ | Input/output | P23 | Fixed frequency output pin (for buzzer or system clock trimming). | | Input | E |
| X1, X2 | | — | Crystal or ceramic resonator for the main system clock is connected across these pins. When the external clock is used, use X1 to input the clock and input X1's reverse phase to X2. | | | |
| XT1, XT2 | | — | Crystal for subsystem clock is connected across these pins. When the external clock is used, use XT1 to input the external clock and leave open XT2. | | | |
| RESET | Input | — | System reset input pin (low-level active). | | | Ⓑ |
| $V_{PRE}$ | | | FIP controller/driver output buffer power supply pin. | | | I |
| $V_{LOAD}$ | | | FIP controller/driver pull-down resistor connection pin. | | | I |
| $V_{DD}$ | | | Positive power supply pin. | | | |
| $V_{SS}$ | | | GND | | | |

* Circuits enclosed by circle have Schmitt trigger input.



TYPE A: This is a CMOS standard input buffer.

TYPE B: This is a Schmitt trigger input with hystereses characteristics.

TYPE D: This is a push-pull output that can be set to high impedance (with both P-ch and N-ch off).

TYPE E: This I/O circuit consists of a Type D push-pull output and a Type A (input buffer).

TYPE F: This I/O circuit consists of a Type D push-pull output circuit and a Type B Schmitt trigger input circuit.

TYPE G: This I/O circuit can select either push-pull output or N-ch open drain output (P-ch off).

**Fig. 2.1-1 Pin I/O Circuit**

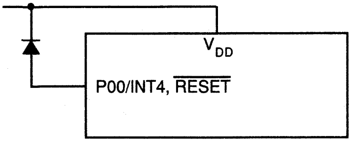
### 2.2 Recommended Conditions for Unused Pins

| Pin | Recommended conditions |
|---|---|
| P00 | Connect to $V_{SS}$. |
| P01-P03 | Connect to $V_{SS}$ or $V_{DD}$. |
| P10-P13 | Connect to $V_{SS}$. |
| P20-P23<br>P30-P33<br>P40-P43<br>P50-P53<br>P60-P63 | Input mode: Connect to $V_{SS}$ or $V_{DD}$.<br>Output mode: No connection is required. |
| IC<br>S0-S9<br>T15/S10-T14/S11<br>T0-T9<br>T10/S15/PH3-T13/S12/PH0 | No connection required. |
| XT1 | Connect to $V_{SS}$ or $V_{DD}$. |
| XT2 | No connection required. |
| $\overline{\text{RESET}}$ with internal power-on circuit | Connect to $V_{DD}$. |
| $V_{LOAD}$ without on-chip load resistor | Connect to $V_{SS}$ or $V_{DD}$. |

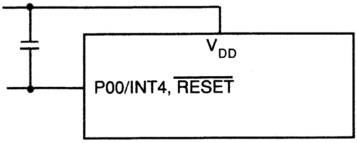
### 2.3 Cautions on Use of P00/INT4 and $\overline{\text{RESET}}$ Pins

The P00/INT4 and $\overline{\text{RESET}}$ pins have the test mode setting function (IC test) to test internal operation of μPD75268 in addition to the functions described in 2.1.11 and 2.1.13.
When voltage exceeding $V_{DD}$ is applied to either of the pins, the test mode is set. Thus, if noise exceeding $V_{DD}$ is applied during normal operation, the test mode is entered and the normal operation may not be continued.
For example, if wiring from the P00/INT4 or $\overline{\text{RESET}}$ pin is long, inter-wiring noise is applied to the pin, the pin voltage exceeds $V_{DD}$, and malfunction may be caused.
Thus, wire so as to suppress inter-wiring noise as much as possible. If noise cannot be suppressed, use an external part as shown below:

- Insert diode between the P00/INT4 or $\overline{\text{RESET}}$ pin and $V_{DD}$

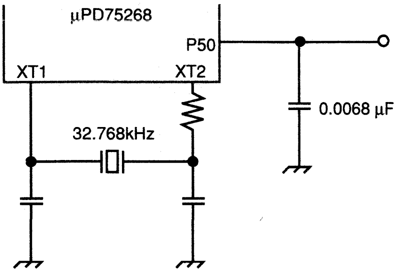


- Insert capacitor between the P00/INT4 or $\overline{\text{RESET}}$ pin and $V_{DD}$



### 2.4 Note for use of XT1, XT2 and P50 pins

When 32.768kHz subsystem clock is used for source clock of watch timer, switching frequency of input and/or output signals on P50 pin which is placed next to the XT2 pin should be lower than once in a second.
If signals of P50 pin has higher frequency, spike noises are generated on XT2 pin by capacitive coupling of P50 and XT2 pins. As a result, normal watch function will not be given (Watch will run too fast). In this case, external capacitor will be needed on P50 pin like in the following figure.



## 3. μPD75268 ARCHITECTURE AND MEMORY MAP

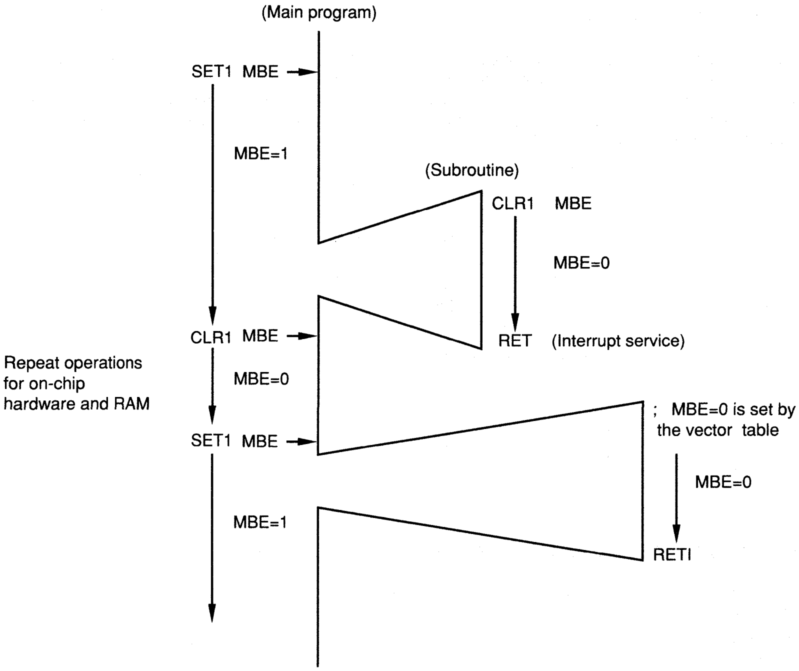
The following three points are features of the μCOM75X's that are adopted to the μPD75268 architecture.
(1) Data memory in bank configuration
(2) Memory-mapped I/O
This chapter discribes each of these features.

### 3.1 Data Memory in Bank Configuration and Addressing Mode

3.1.1 Data memory in bank configuration
The μPD75268 contain a general-purpose RAM space up to 512 x 4 words of of the data memory, and peripheral hardware (I/O ports, timer, etc.) on addresses F80H to FFFH of the data memory. The 64 x 4 bits at address 1C0H to1FFH are also used as display data memory. To address the 12-bit data address space the μPD75268 employ memory bank configuration in which the lower 8-bit addresses are specified by instructions either directly or indirectly and the higher 4-bit addresses are specified by memory bank (MB). To specify a memory bank (MB), the μPD75268 contain:
- Memory bank enable flag (MBE)
- Memory bank select register (MBS)

The MBS register is used to select a specific memory bank. On the μPD75268, values 0, 1, or 15 can be preset into this register. The MBE flag specifies whether the bank selected by the MBS register is to be made valid or not. If the MBE flag is reset (zero), a fixed memory bank is selected with no regard to the MBS (see Fig. 2.1-1). If the MBE is set (1), memory bank can be changed by the MBS setting to expand the data memory space.For addressing data memory space, the MBE flag is usually set (1) and the data memory area in the memory bank specified by the MBS is manipulated. Efficient programming is possible by using the MBE=0 and MBE=1 modes adequately.

| | Possible program processes | Effect |
|---|---|---|
| MBE=0 mode | •Interrupt process | Save/restore for MBS data is not needed |
| | • Process in which internal hardware manipulation and general-purpose RAM access are repeated. | Updating to MBS data is not needed |
| | • Subroutine process | Save/restore for MBS data is not needed |
| MBE=1 mode | • Normal program process | |



Since the MBE is automatically saved/restored in a subroutine procedure, it can be freely changed during subroutine process. In an interrupt procedure, the MBE in the interrupt service can be specified automatically at the beginning of interrupt service routine by presetting of interrupt vector table, while the MBE is stored/restored automatically.
This realizes fast interrupt operation. If the MBS is to be changed during subroutine process or interrupt service, it should be saved/restored with the PUSH/POP instruction.
The MBS is set with the SEL instruction.
Ex. 1: Clears the MBE to fix memory bank:

```
CLR1  MBE  ;     MBE←0
```

Ex. 2: Selects memory bank 1:

```
SET1  MBE  ;     MBS←1
SEL   MB1  ;     MBS←1
```

2.1.2 Data memory addressing mode
The μCOM75X architecture adopted for μPD75268 provides seven addressing modes listed in Fig. 3.1-2 and Table 3.1-1 to efficiently address the data memory space for each bit length of the data to be processed, thus enabling efficient programming.
(1) One bit direct addressing (mem.bit)
In this addressing mode, each bit of the entire data memory space is directly addressed with operands. In MBE=0 mode, the memory bank is fixed to MB=0 if the address specified by an operand is 00H-7FH, and to MB=15 if the specified address is 80H-FFH. In MBE=0 mode, therefore, both general-purpose RAM area (000H-07FH) and peripheral hardware area (F80H-FFFH) are addressable.
In MBE=1 mode, memory bank can be specified by the MBS register, allowing expansion of addressable data memory space. This addressing mode is available with four instructions: i.e., Set Bit and Reset Bit (SET1/CLR1) and Test Bit (SKT/SKF) instructions.

EX.: Sets FLAG1, resets FLAG2, and tests if FLAG3 IS ZERO:

```
FLAG1   EQU   03FH.1  ;   bit 1 of address 3FH
FLAG2   EQU   087H.2  ;   bit 2 of address 87H
FLAG3   EQU   0A7H.0  ;   bit 0 of address A7H

        SET1  MBE     ;   MBE   ←  1
        SEL   MB0     ;   MBS   ←  0
        SET1  FLAG    ;   FLAG  ←  1
        CLR1  FLAG2   ;   FLAG2 ←  0
        SKF   FLAG3   ;   FLAG3 =  0?
```

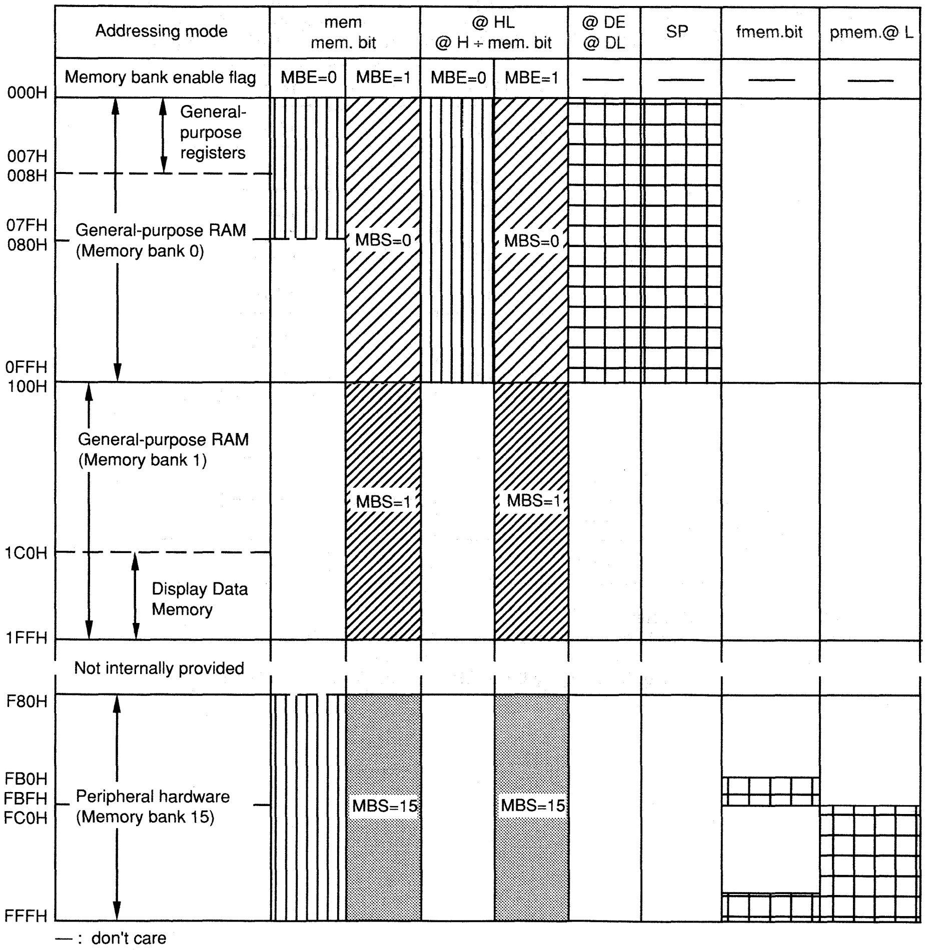


**Fig. 3.1-2 Data Memory Configuration and Addressing Range of each Addressing Mode**

| | | | | | |
|---|---|---|---|---|---|
| | | 1C3H | 1C2H | 1C1H | 1C0H |
| | | 1C7H | 1C6H | 1C5H | 1C4H |
| | | 1CBH | 1CAH | 1C9H | 1C8H |
| | | 1CFH | 1CEH | 1DDH | 1CCH |
| | | 1D3H | 1D2H | 1D1H | 1D0H |
| | | 1D7H | 1D6H | 1D5H | 1D4H |
| | | 1DBH | 1DAH | 1D9H | 1D8H |
| | | 1DFH | 1DEH | 1DDH | 1DCH |
| | | 1E3H | 1E2H | 1E1H | 1E0H |
| | | 1E7H | 1E6H | 1E5H | 1E4H |
| | | 1EBH | 1EAH | 1E9H | 1E8H |
| | | 1EFH | 1EEH | 1EDH | 1ECH |
| | | 1F3H | 1F2H | 1F1H | 1F0H |
| | | 1F7H | 1F6H | 1F5H | 1F4H |
| | | 1FBH | 1FAH | 1F9H | 1F8H |
| | | 1FFH<br>(PORTH) | 1FEH<br>(KS1) | 1FDH<br>(KS0) | 1FCH |
| Number of Manipulable Bits | 1 bit | O | O | O | O |
| | 4 bit | O | O | O | O |
| | 8 bit | O | | O | |

Note 1: KS0, KS1: Key Scan Register
Note 2: PORTH: High voltage and high current output port (shared with digit outputs)

**Fig. 3.1-3 Display Data Memory Configuration μPD75268**

**Table 3.1–1 Addressing Mode**

| Addressing mode | Identifier | Address to specify |
|---|---|---|
| 1–bit direct addressing | mem.bit | This is the bit (specified by bit) of the address specified by MB and mem. Where,<br>MBE=0 { when mem=00H to 7FH, MB=0<br>when mem=80H to FFH, MB=15<br>MBE=1 MB=MBS |
| 4–bit direct addressing | mem | Addressing specified by MB and mem. Where,<br>MBE=0 { when mem=00H to 7FH, MB=0<br>when mem=80H to FFH, MB=15<br>MBE=1 MB=MBS |
| 8–bit direct addressing | | Addressing specified by MB and mem (mem is even address). Where,<br>MBE=0 { when mem=00H to 7FH, MB=0<br>when mem=80H to FFH, MB=15<br>MBE=1 MB=MBS |
| 4–bit register indirect addressing | @ HL | Address specified by MB and HL. Where, MB=MBE MBS |
| | @ DE | Address specified by DE of memory bank 0. |
| | @ DL | Address specified by DL of memory bank 0. |
| 8–bit register indirect addressing | @ HL | Address specified by MB and HL<br>Where, MB=MBE MBS and bit 0 of the L register is ignored. |
| Bit manipulation addressing | fmem.bit | The bit (specified by bit) of the address specified by fmem. Where<br>fmem { FB0H to FBFH (hardware relating to interrupts) FF0H to FFFH (I/O ports) |
| | pmem.@ L | The bit specified by the lower 2 bits of the L register whose address is specified by the higher 10 bits of pmem and higher 2 bits of the L register.<br>Where, pmem=FC0H to FFFH |
| | @ H+mem.bit | The bit (specified by bit) of the address specified by MB, H, and lower 4 bits of mem.<br>Where, MB=MBE.MBS |
| Stack addressing | | Address specified by the SP of the memory bank 0. |

(2) Four-bit direct addressing (mem)

This addressing mode allows each 4-bit on the entire data memory space to be directly addressed with operand of instruction. In the MBE=0 mode, addressable areas are fixed to the general-purpose RAM area (000H–07FH) and peripheral hardware area (F80H–FFFH) similar to the one-bit direct addressing mode. In MBE=1 mode, the entire data memory area can be addressed with the MBS register setting.

This addressing mode is applicable to the MOV, XCH, INCS, IN, and OUT instruction.

Ex. 1: Reads the value of Port 4 and stores it to DATA1:

```
DATA1   EQU    5FH        ;   DATA1 is address 5FH.
        CLR1   MBE        ;   MBE ← 0
        IN     A, PORT4   ;   A ← PORT4
        MOV    DATA1, A   ;   (DATA1) ← A
```

Ex. 2: Outputs data at Buff to Port 6:

```
BUFF    EQU     11AH        ;    BUFF is address 11AH.
        SET1    MBE         ;    MBE←1
        SEL     MB1         ;    MBS←1
        MOV     A, BUFF     ;    A← (BUFF)
        SEL     MB15        ;    MBS← 15
        OUT     PORT6, A    ;    PORT6← A
```

Note: Program efficiency will be reduced if, as shown in example 2, data pertaining to I/O ports is stored in the general-purpose RAM in Bank 1.
Updating to the MBS will not be required in programming (as shown in the example 1) if the port data is stored in address area 00H to 7FH in Bank 0.

(3) Eight-bit direct addressing (mem)
In this addressing mode, each 8-bit on the entire data memory area can be directly addressed with operand of instruction. Only even-numbered addresses can be specified by operands. Four-bit address data that is incremented by one are paired to be 8-bit processed by the 8-bit accumulator (XA register pair).
The specified memory banks are the same as those specified in the 4-bit direct addressing mode.
This addressing mode is applicable to the MOV, XCH, IN, and OUT instructions.

Ex. 1: Transfers 8-bit data on Ports 4 and 5 to addresses 20H and 21H.

```
DATA    EQU   020H
CLR1    MBE         ;    MBE← 0
IN      XA, PORT4   ;    XA← Port5, 4
XCH     XA, DATA    ;    XA↔ (21H, 20H)
```

Ex. 2: Fetches the 8-bit data loaded in the serial interface's shift register (SIO), and sets transfer data and start transfer

```
SEL     MB15     ;    MBS← 15
XCH     XA, SIO  ;    XA↔ SIO
```

(4) Four-bit register indirect addressing (@rpa)
In this addressing mode, each 4-bit on the data memory space are indirectly addressed by a data pointer (general-purpose register pair) which is specified by an operand of instruction.
Three register pairs are available for data pointer: the HL register pair which can specify the entire data memory area by MB=MBE • MBS specifying; the DE and the DL register pairs which each always specifies memory bank zero with no regard to the MBE and MBS setting. Effecient programming is possible by choosing the appropriate data pointer according to the memory bank to be used.

Ex. 1: Transfers data in addresses 50H to 57H to addresses 110H to 117H:

```
DATA1   EQU     57H
DATA2   EQU     117H
        SET1    MBE                    ;  MBE← 1
        SEL     MB1                    ;  MBS← 1
        MOV     D, # DATA1 SHR 4       ;  D← 5
        MOV     HL, # DATA2 and 0FFH   ;  HL← 17H
LOOP:   MOV     A, @ DL                ;  A← (DL)
        XCH     A, @ HL                ;  A↔ (HL)
        DECS    L                      ;  L← L-1
        BR      LOOP
```

The addressing mode in which the HL register pair is used as a data pointer is used for data transfer, arithmetic operations, comparsion, I/O operations, and other purposes. The addressing mode using the DE or DL register pair is used for the MOV or XCH instructions.

When this addressing mode is combined with an Increment or Decrement instruction for general-purpose registers or register pairs, the RAM address on the data memory space can be freely updated as shown in Fig. 3.1-3.

Ex. 2: Compares the data in addresses 50H to 57H with those in addresses 110H to 117H.

```
DATA1   EQU    57H
DATA2   EQU    117H
        SET1   MBE
        SEL    MB1
        MOV    D, # DATA1 SHR 4
        MOV    HL, # DATA 2 and 0FFH
LOOP    MOV    A, @DL
        SKE    A, @HL              ;    A = (HL)?
        BR     NO                  ;    NO
        DECS   L                   ;    YES, L ← L-1
        BR     LOOP
```

Ex. 3: Clears the data memory area 04H to FFH.

```
        SEL    MB0
        MOV    XA, # 00H
        MOV    HL, # 04H
LOOP:   MOV    @HL, A              ;    (HL) ← A
        INCS   L                   ;    L  ← L+1
        BR     LOOP
        INCS   H                   ;    H  ← H+1
        BR     LOOP
```

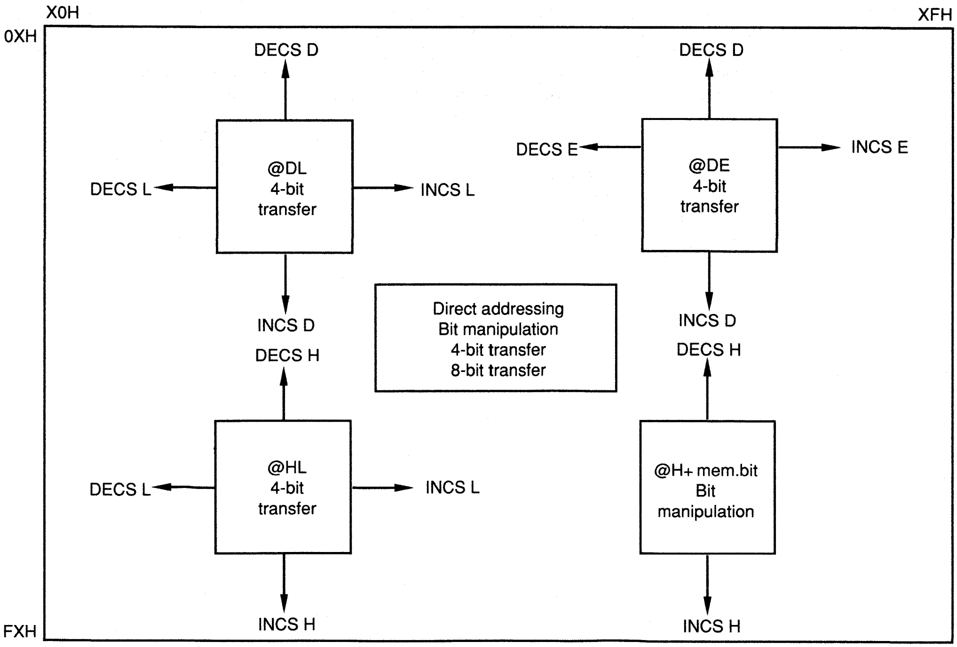


**Fig. 3.1-5 Method of General-Purpose RAM Address modification**

(5) 8-bit register indirect addressing (@HL)

This addressing mode indirectly specifies all the data memory space in 8-bit units by using the data pointer (HL register pair). The 4-bit data at the address setting data pointer bit 0 (L register bit 0) to 0 and the 4-bit data at the address + 1 are paired and transferred to the 8-bit accumulator (XA register) for 8-bit processing.

The memory banks specified in the addressing mode are the same as those when the HL register is specified in the 4-bit register indirect addressing mode (MB = MBE • MBS). The 8-bit register indirect addressing mode is applicable to the MOV, XCH, and SKE Instructions.

Example 1: To compare the count register (T0) value of timer / event counter 0 with data at addresses 30H and 31H for equality.

```
DATA  EQU   30H
      CLR1  MBE
      MOV   HL, #DATA
      MOV   XA, T0      ;  XA←  count register 0
      SKE   A, @HL      ;  A = (HL) ?
      BR    NO
      INCS  L
      MOV   A, X        ;  A← X
      SKE   A, @HL      ;  A = (HL) ?
```

(6) Bit manipulation addressing

This addressing mode is used to perform bit manipulations such as Boolean operation or bit transfer on any bit in all of the data memory space.

Although the 1-bit direct addressing mode is applicable only to the bit set, reset, and test instructions, the bit manipulation addressing mode enables bit manipulations such as Boolean operations by using the AND1, OR1, and XOR1 instructions and allows bit test and bit reset by using the SKTCLR instructions.

The following three types of bit manipulation addressing modes can be used according to the data memory address to be used:

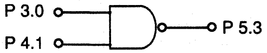
(a) Specific address bit direct addressing (fmem. bit).

This addressing mode enables, peripheral hardware that frequently uses bit manipulation, such as input / output ports and interrupt flags, to be operated at any time independently of the memory bank setting. Thus, the addressing mode is applicable to data memory addresses FF0H-FFFH (where input / output ports are mapped) and FB0H-FBFH (where hardware related to interrupts is mapped). For the hardware of the two data memory areas, bit manipulation can be performed in direct addressing at any time as desired regardless of how MBS and MBE are set.

Example 1: To test the timer 0 interrupt request flag (IRQT0); if the request flag is set, clear the flag and reset P63.

```
SKTCLR   IRQT0      ;  IRQT0 = 1 ?
BR       NO         ;  NO
CLR1     PORT6.3    ;  YES
```

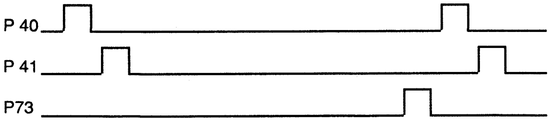
Example 2: To reset P53 if both P30 and P41 are set to 1.



```
(i)      SET1   CY            ;   CY ←— 1
         AND1   CY, PORT3.0   ;   CY P3.0
         AND1   CY, PORT4.1   ;   CY P4.1
         SKT    CY            ;   CY = 1 ?
         BR     SETP
         CLR1   PORT5.3       ;   P53 ←— 1
          :
          :
 SETP:   SET1 PORT5.3         ;   P53 ←— 1
          :
          :

(ii)     SKT    PORT3.0       ;   P30 = 1 ?
         BR     SETP
         SKT    PORT4.1       ;   P41 = 1 ?
         BR     SETP
         CLR1   PORT5.3       ;   P53 ←— 0
          :
          :
STEP:    SET1   PORT5.3       ;   P53 ←— 1
```

(b) Specific address bit register indirect addressing (pmem. @L).
This addressing mode indirectly specifies each bit of peripheral hardware, such as an input / output port, by using the L register for successive operation. The addressing mode is applicable to data memory addresses FC0H-FFFH.
The addressing mode directly specifies the high-order 10 bits of a 12-bit data memory address in the pmem operand and indirectly specifies the low-order 2-bit data memory address part and the bit address by using the L register. Thus, 16 bits (four ports) can be manipulated (operated) successively by L register specification.
The address mode also enable bit manipulation to be performed at any time independently of how MBE and MBS are set.

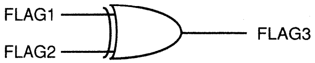
Example: To output a pulse to each bit of ports 4 to 7 in order.



```
       MOV    L, #0
LOOP:  SET1   PORT4. @L   ;   Port 4–7 bit (L1-0) ←— 1
       CLR1   PORT4. @L   ;   Port 4–7 bit (L1-0) ←— 0
       INCS   L
       BR     LOOP
```

(c) Special 1-bit direct addressing (@H + mem. bit)
This addressing mode enables bit manipulation to be performed on any bit in all of the data memory space.
The addressing mode indirectly specifies the high-order four bits of data memory address of the memory bank specified by MB = MBE MBS by using the H register and directly specifies the low-order 4-bit data memory address part and the bit address in the operands. It enable various types of bit manipulation to be performed on any bit in all of the data memory space.

Example: To reset address 32H bit 2 (FLAG3) if both address 30H bit 3 (FLAG1) and address 31H bit 0 (FLAG2) are set to 0 or 1.



```
FLAG1  EQU    30H.3
FLAG2  EQU    31H.0
FLAG3  EQU    32H.2
       SEL    MB0
       MOV    H, #FLAG1 SHR 6
       CLR1   CY
       OR1    CY, @H + FLAG1   ;   CY←  0
       XOR1   CY, @H + FLAG2   ;   CY←  CY-FLAG1
       SET1   @H + FLAG3       ;   CY←  CY-FLAG2
       SKT    CY               ;   CY = 1 ?
       CLR1   @H + FLAG3       ;   FLAG←  30
```

(7) Stack addressing
The stack addressing mode is used for register save and restore during interrupt service or subroutine processing.
The processing mode specifies an address by using the 8-bit stack pointer (data memory bank 0).
The addressing mode can also be used for register save and restore by executing the PUSH and POP instructions.

Example 1: To save and restore register in subroutine processing.

```
SUB: PUSH   XA
     PUSH   HL
     PUSH   BS   ;  MBS is saved.
      :
      :
     POP    BS
     POP    HL
     POP    XA
     RET
```

Example 2: To transfer the HL register pair contents to the DE register pair.

```
PUSH   HL
POP    DE   ;  DE←  HL
```

Example 3: To branch to the address indicated by [XABC] register.

```
PUSH   BC
PUSH   XA
RET         ;  Branch to XABC address
```

### 3.2 Memory Mapped I / O

The μPD75268 adopts memory mapped I / O where peripheral hardware such as input / ports and timers is allocated to addresses F80H-FFFH of the data memory space as shown in Fig. 3.2–2. Thus, peripheral hardware is controlled entirely by memory operation instructions rather than special instructions. (To easily understand programs, some hardware control mnemonics are provided.) The display data memory, key scan register, port H mapped to addresses 1C0H to 1FFH are to be manipulated by specifying memory bank 1.
Table 3.2–1 lists the addressing modes that can be used to operate peripheral hardware.

**Table 3.2–1 Applicable Addressing Modes during Peripheral Hardware Operation**

| | Applicable addressing mode | Applicable hardware |
|---|---|---|
| Bit manipulation | With MBE=0 or (MBE=1 and MBS=15), direct addressing (specification in mem. bit). | All hardware where bit manipulation can be performed. |
| | Direct addressing regardless of how MBE and MBS are set. (specification in fmem. bit) | ISTO, MBE<br>IExxx, IRQxxx, PORTn.x |
| | Indirect addressing regardless of how MBE and MBS are set. (specification in pmem. @L) | PORTn.x |
| 4-bit manipulation | With MBE=0 or (MBE=1 and MBS=15), direct addressing (specification in mem). | All hardware where 4-bit manipulation can be performed. |
| | With (MBE=1 and MBS=15), register indirect addressing (specification in @HL). | |
| 8-bit manipulation | With MBE=0 or (MBE=1 and MBS=15), direct addressing (specification in mem), mem must be an even address. | All hardware where 8-bit manipulation can be performed. |
| | With MBE=1 and MBS=15, register indirect addressing (specification in @HL; the L register must contain an even number | |

```
Example: CLR1     MBE          ;   MBE = 0
         SET1     TM0.3        ;   Timer 0 starts.
         EI       IE0          ;   INT0 is enabled.
         DI       IE1          ;   INT1 is disabled.
         SKTCLR   IRQ2         ;   INT2 request flag is tested and cleared.
         SET1     CY
         SET1     PORT4. @L    ;   Port 4 is set.
         AND1     CY, Port4. @L
         IN       A, PORT0     ;   A ← port 0
         OUT      PORT4, XA    ;   Port 5, 4 ← XA
```

Figs. 3.2–1 to 3.2–3 shows the μPD75268 I/O map.
The columns in the figures mean:

- Abbreviation:
  Name indicating internal hardware address. It can be entered in the instruction operand field.

- R/W
  Indicates whether the hardware device can be read or written.
  – R/W : Read and write are enabled.
  – R : Read only is enabled.
  – W : Write only is enabled.

- Number of bits that can be manipulated:
  Indicates the number of bits that can be processed when the hardware device is operated.
  O: Bit manipulation is enabled in 1-4, 4-, or 8-bit units as specified in the column.
  Δ: Only some bits can be manipulated. See Remarks for the bits that can be manipulated.
  –: Bit manipulation cannot be performed in 1-, 4-, or 8-bit units as specified in the column.

- Bit manipulation addressing:
  Indicates the applicable bit manipulation addressing for performing bit manipulation on the hardware device.

**Table 3.2-2 µPD75268 I/O Map (1/3)**

| Address | Hardware (Symbol) | | | | R/W | Number of bits to manipulate | | | Addressing mode | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| | b3 | b2 | b1 | b0 | | 1 bit | 4 bits | 8 bits | | |
| F80H | Stack pointer (SP) | | | | R/W | — | — | O | | Bit 0 is fixed to 0. |
| | | | | | | — | — | | | |
| F85H | Basic interval timer mode register (BTM) | | | | W | | O | — | mem.bit | Only bit 3 can be bit manipulated. |
| F86H | Basic interval timer (BT) | | | | R | — | — | O | | |
| | | | | | | — | — | | | |
| F88H | Display mode register (DSPM) | | | | W | — | O | — | | |
| F89H | Dimmer select register (DIMS) | | | | W | — | O | — | | |
| F8AH | KSF | Digit select register (DIGS) | | | R/W | Δ | O | — | mem.bit | Only bit 3 is testable. |
| F98H | Watch mode register (WM) | | | | W | — | — | O | | |
| | | | | | | | | | | |
| FA0H | Timer/event counter #0 mode register (TM0) | | | | W | Δ | — | O | mem.bit | Only bit 3 can be bit manipulated. |
| | | | | | | | | | | |
| FA4H | Timer/event counter #0 count register (T0) | | | | R | — | — | O | | |
| | | | | | | — | — | | | |
| FA6H | Timer/event counter #0 modulo register (TMOD0) | | | | W | — | — | O | | |
| | | | | | | | | | | |
| FB0H | 0 | IST0 | MBE | 0 | R/W | O | O | O | fmem.bit | |
| | Program status word (PSW) | | | | R | — | — | | | |
| FB2H | IME | | | | W | — | — | — | | EI and DI instruction are used |
| FB3H | Processor clock control register (PCC) | | | | W | — | O | | | |
| FB4H | INT0 mode register (IM0) | | | | W | — | O | — | | Bit 2 is fixed to 0. |
| FB5H | INT1 mode register (IM1) | | | | W | — | O | | | Bits 3, 2, 1 are fixed to 0 |
| FB7H | System clock control register (SCC) | | | | W | Δ | — | — | | Only bits 3 and 0 can be bit manipulated. |

**Table 3.2-2 μPD75268 I/O Map (2/3)**

| Address | Hardware (Symbol) | | | | R/W | Number of bits to manipulate | | | Addressing mode | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| | b3 | b2 | b1 | b0 | | 1 bit | 4 bits | 8 bits | | |
| FB8H | IE4 | IRQ4 | IEBT | IRQBT | R/W | O | O | — | fmem.bit | |
| FBAH | — | — | IEW | IRQW | R/W | O | O | — | | |
| FBBH | IEKS | IRQKS | — | — | R/W | O | O | — | | |
| FBCH | — | — | IET0 | IRQT0 | R/W | O | O | — | | |
| FBDH | — | — | IESIO | IRQSIO | R/W | O | O | | | |
| FBEH | IE1 | IRQ1 | IE0 | IRQ0 | R/W | O | O | — | | |
| FBFH | — | — | — | IRQ2 | R/W | O | O | | | |
| FE0H | Serial operation mode register (SIOM) | | | | W | Δ | — | O | mem.bit | Only bit 3 can be bit manipulated. |
| FE4H | Serial I/O shift register (SIO) | | | | R/W | — | — | O | | |
| | | | | | | — | — | | | |
| FE8H | PM33 | PM32 | PM31 | PM30 | W | — | — | O | | |
| | Port mode register group A (PMGA) | | | | | — | — | | | |
| | PM63 | PM62 | PM61 | PM60 | | | | | | |
| FECH | — | PM2 | — | — | W | — | — | O | | |
| | Port mode register group B (PMGB) | | | | | — | — | | | |
| | — | — | PM5 | PM4 | | | | | | |
| FF0H | Port 0 (PORT0) | | | | R | O | O | — | fmem.bit<br>pmem.@L | |
| FF1H | Port 1 (PORT1) | | | | R | O | O | | | |
| FF2H | Port 2 (PORT2) | | | | R/W | O | O | — | | |
| FF3H | Port 3 (PORT3) | | | | R/W | O | O | | | |
| FF4H | Port 4 (PORT4) | | | | R/W | O | O | O | | |
| FF5H | Port 5 (PORT5) | | | | R/W | O | O | | | |
| FF6H | Port 6 (PORT6) | | | | R/W | O | O | — | | |
| 1C0H+4n | Display data memory: S0-S7 (n=0-15) | | | | R/W | O | O | O | mem.bit | |
| 1C1H+4n | | | | | | O | O | | | |
| 1C2H+4n | Display data memory: S8-S15 (n=0-15) | | | | R/W | O | O | O | | |
| 1C3H+4n | | | | | | O | O | | | |
| 1FCH | Key scan register (KS0) | | | | R/W | O | O | O | | |
| 1FDH | | | | | | O | O | | | |
| 1FEH | Key scan register (KS1) | | | | R/W | O | O | O | | |
| 1FFH | Port H (PORTH) | | | | R/W | O | O | O | | |

2

## 4. INTERNAL CPU FUNCTIONS

### 4.1 Program Counter (PC) 13 bits

This is a 13-bit binary counter that stores address information of the program memory.

µPD75268

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

**Fig. 4.1-1 Program Counter Configuration**

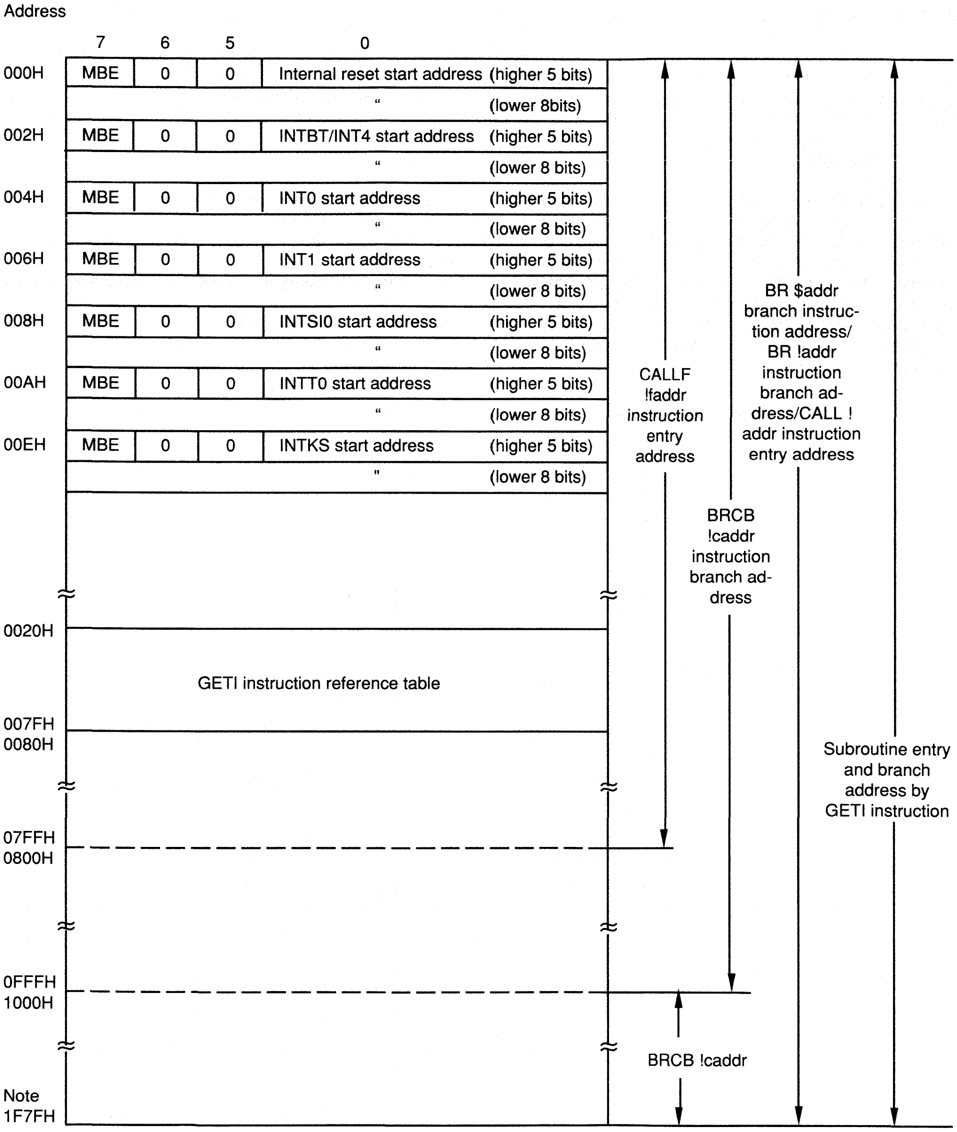
When an instruction is executed, the program counter is automatically incremented according to the number of bytes of the executed instruction.
When branch instruction (BR, BRCB) is executed, the contents of immediate data or register pair indicating address to be branched are loaded to all of some or the bits of the PC.
When a call instruction (CALL, CALLF) is executed or when a vector interrupt is generated, the contents of the PC (return address that has been incremented to fetch the next instruction) are saved to the stack memory and then the corresponding addresses to which the program execution are to be jumped are loaded.
When a return instruction (RET, RETS, RETI) is executed, the contents of the stack memory are set to PC.
Generation of internal reset signal ($\overline{\text{RES}}$) sets the lower 5 bits of address 0000H of the program memory to the PC12 to PC8. It also sets the contents of address 0001H to PC7 to PC0. Thus the PC is initialized. For this reason, program execution can be started from any address.

### 4.2 Program Memory (ROM)

**8,064 words x 8 bits (µPD75268)**

The program memory is mask programmable ROM consisting of 8064 words x 8 bits and stores programs, interrupt vector table, GETI instruction reference table, and table data.
The program memory is addressed by using the program counter. Table data can be referenced by using the table reference instruction (MOVT).
Figures 4.2-1 and 4.2-2 show the address range in which a branch can be made by using branch instruction or subroutine call instruction. In addition to the instructions, the BR PCDE and BR PCXA instructions can be used for a branch to address where only the low-order eight bits of PC are changed.
The program memory addresses are 0000H-1F7FH. The following addresses are specially assigned: (All area except address 0000H or 0001H can be used as normal program memory.)

- 0000H to 0001H (000H to 001H)
  This address area is used as vector address table to which the program start address when a reset signal is applied and set values of MBE are written.
  Program execution can be started from any address after reset.
- 0002H to 000FH (002H to 00FH)
  This address area is used for vector address table to which the program start address by vector interrupt and the set values of MBE are written. Interrupt service can be started from any address.
- 0020H to 007FH (020H to 07FH)
  This is a table area referenced by the GETI instrucion*.
  * The GETI instruction is used to execute any 2- 3-byte instruction in one byte. Using this instruction, the number of program bytes can be reduced.


Address
7
6
5
0
000H
MBE
0
0
Internal reset start address (higher 5 bits)
" (lower 8bits)
002H
MBE
0
0
INTBT/INT4 start address (higher 5 bits)
" (lower 8 bits)
004H
MBE
0
0
INT0 start address (higher 5 bits)
" (lower 8 bits)
006H
MBE
0
0
INT1 start address (higher 5 bits)
" (lower 8 bits)
008H
MBE
0
0
INTSI0 start address (higher 5 bits)
" (lower 8 bits)
00AH
MBE
0
0
INTT0 start address (higher 5 bits)
" (lower 8 bits)
00EH
MBE
0
0
INTKS start address (higher 5 bits)
" (lower 8 bits)
0020H
GETI instruction reference table
007FH
0080H
07FFH
0800H
0FFFH
1000H
Note
1F7FH
CALLF !faddr instruction entry address
BRCB !caddr instruction branch address
BR $addr branch instruction address/ BR !addr instruction branch address/CALL ! addr instruction entry address
Subroutine entry and branch address by GETI instruction
BRCB !caddr

Remarks: The BR $addr instruction enables a branch to the address indicated by PC contents +16 to PC contents +2 or PC contents – 1 to PC contents – 15 regardless of the block boundaries.

**Fig. 4.2-2 Program Memory Map (μPD75268)**

### 4.3 Data Memory ... 512 x 4

The data memory consists of up to 512 words x 4 bits of static RAM. It is used for processing data storage or as stack memory space for use in subroutine or interrupt services. Because it is a static RAM, its data can be retained when the CPU operation stopped, standby mode. This is effective for long time data memory retention with battery back-up.
Fig. 4.3-1 shows data memory map for the µPD75268. The data memory has bank configuration. A bank has up to 256 word x 4 bits. There are two memory banks;

Memory Bank 0
Memory Bank 1

Note the on-chip peripherals are memory – mapped in the bank 15. A memory bank is selected by the 4-bit Memory Bank Select register (MBS=0, 1, 15) when bank selection is enabled by the Memory Bank Enable flag (MBE=1).
When bank selection is disabled (MBE=0), bank 0 or bank 15 is automatically selected by the current addressing mode. Addresses in a bank are addressed by 8-bit immediate data, register pair, etc..
The data memory has one-word four-bit configuration, but 1/4/8-bit unit operation is possible by powerful addressing modes. Details for memory bank selection and addressing, see Section 2.1.
Special area in the data memory are also used as general purpose register (bank 0: 000H to 01FH), stack memory (bank 0: 000H to 0FFH), and display data memory (bank 1: 1C0H to 1FFH). Note the bank 15 has no data memory but has on chip peripherals memory mapping area.

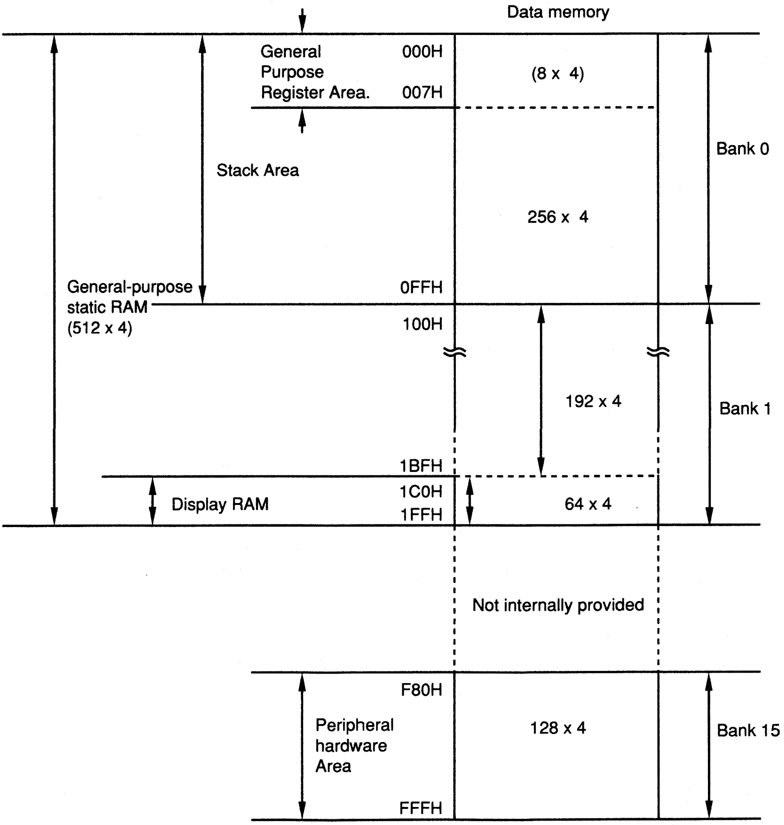


Fig. 4.3-1 Data Memory Map

Each address of the data memory consists of 4 bits; however, the data memory can be manipulated in 8-bit units using 8-bit memory manipulation instructions and in bit units using bit manipulation instructions. Even addresses only must be specified for 8-bit memory manipulation instructions.
When doing a reset during operating mode (not standby), the data memory contents may become undefined due to the possibility of reset during memory access. Therefore, the memory should be initialized at the beginning of every application program (RAM clear).

Example: Clears RAM locations 000H to 1FFH. (FFH remains in the HL register, however):

```
        SET1   MBE
        SEL    MB1
        MOV    HL, #00H
        MOV    XA, #00H
LOOP1:  MOV    @HL, A      ; Clears 100H to 1FFH
        INCS   L
        BR     LOOP1
        INCS   H
        BR     LOOP1
        SEL    MB0
LOOP2:  MOV    @HL, A      ; Clears 00H to FFH
        DECS   L
        BR     LOOP2
        DECS   H
        BR     LOOP2
```

| | | | | | |
|---|---|---|---|---|---|
| | | 1C3H | 1C2H | 1C1H | 1C0H |
| | | 1C7H | 1C6H | 1C5H | 1C4H |
| | | 1CBH | 1CAH | 1C9H | 1C8H |
| | | 1CFH | 1CEH | 1DDH | 1CCH |
| | | 1D3H | 1D2H | 1D1H | 1D0H |
| | | 1D7H | 1D6H | 1D5H | 1D4H |
| | | 1DBH | 1DAH | 1D9H | 1D8H |
| | | 1DFH | 1DEH | 1DDH | 1DCH |
| | | 1E3H | 1E2H | 1E1H | 1E0H |
| | | 1E7H | 1E6H | 1E5H | 1E4H |
| | | 1EBH | 1EAH | 1E9H | 1E8H |
| | | 1EFH | 1EEH | 1EDH | 1ECH |
| | | 1F3H | 1F2H | 1F1H | 1F0H |
| | | 1F7H | 1F6H | 1F5H | 1F4H |
| | | 1FBH | 1FAH | 1F9H | 1F8H |
| | | 1FFH<br>(PORTH) | 1FEH<br>(KS1) | 1FDH | 1FCH<br>(KS0) |
| Number of Manipulable Bits | 1 bit | O | O | O | O |
| | 4 bit | O | O | O | O |
| | 8 bit | O | | O | |
| | | 1C0H +<br>4n + 3 | 1C0H +<br>4n + 2 | 1C0H +<br>4n + 1 | 1C0H +<br>4n + 0 |

Note 1: KS0, KS1: Key Scan Register
Note 2: PORTH: High-voltage and high current output port also used for digit output.
Note 3: n = 0 to 15

**Fig. 4.3-3 Display Data Memory Configuration µPD75268**

Example: Clears 1C0H to 1FFH of the Display Data Memory.

```
         SET1   MBE
         SEL    MB1
         MOV    HL, #0C0H
         MOV    XA, #00H
LOOP:    MOV    @HL, XA     ; Clears the Display data Memory in 8-bit units.
         INCS   L
         INCS   L
         BR     LOOP
         INCS   H
         BR     LOOP
```

### 4.4 General Purpose Registers – Eight x Four Bits

The general purpose registers are eight 4-bit registers (B, C, D, E, H, L, X, and A) mapped in specific addresses of the data memory. Every general purpose register is handled in 4-bit units; register pairs BC, DE, HL, and XA are also used for 8-bit manipulation. In addition to DE and HL, registers D and L are also paired (DL), and the three register pairs can be used for data pointers. The general purpose register area can be addressed and accessed as normal RAM regardless of whether or not it is used for registers.

| | |
|---|---|
| X<br>01H | A<br>00H |
| H<br>03H | L<br>02H |
| D<br>05H | E<br>04H |
| B<br>07H | C<br>06H |

**Figure 4.4–1 General Purpose Register Configuration (When 4-bit processing is performed)**

| |
|---|
| XA<br>00H |
| HL<br>02H |
| DE<br>04H |
| BC<br>06H |

**Figure 4.4–2 General Purpose Register Configuration (When 8-bit processing is performed)**

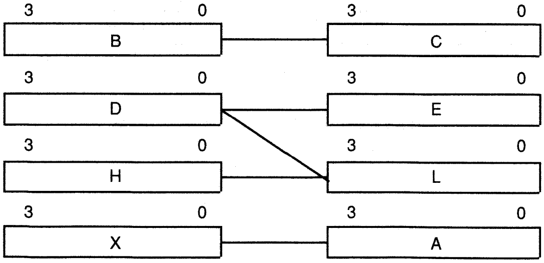


**Figure 4.4–3 Register Pair Configuration**

### 4.5 Accumulator

The μPD7500X uses the A register and XA register pair for accumulators. The A register is used as the main register during execution of 4-bit data processing instructions; the XA register pair is used as the main register pair during execution of 8-bit data processing instructions.
The carry flag (CY) is used for a bit accumulator during execution of bit manipulation instructions.

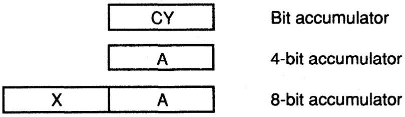


**Figure 4.5–1 Accumulator**

### 4.6 Stack Pointer (SP) – Eight Bits

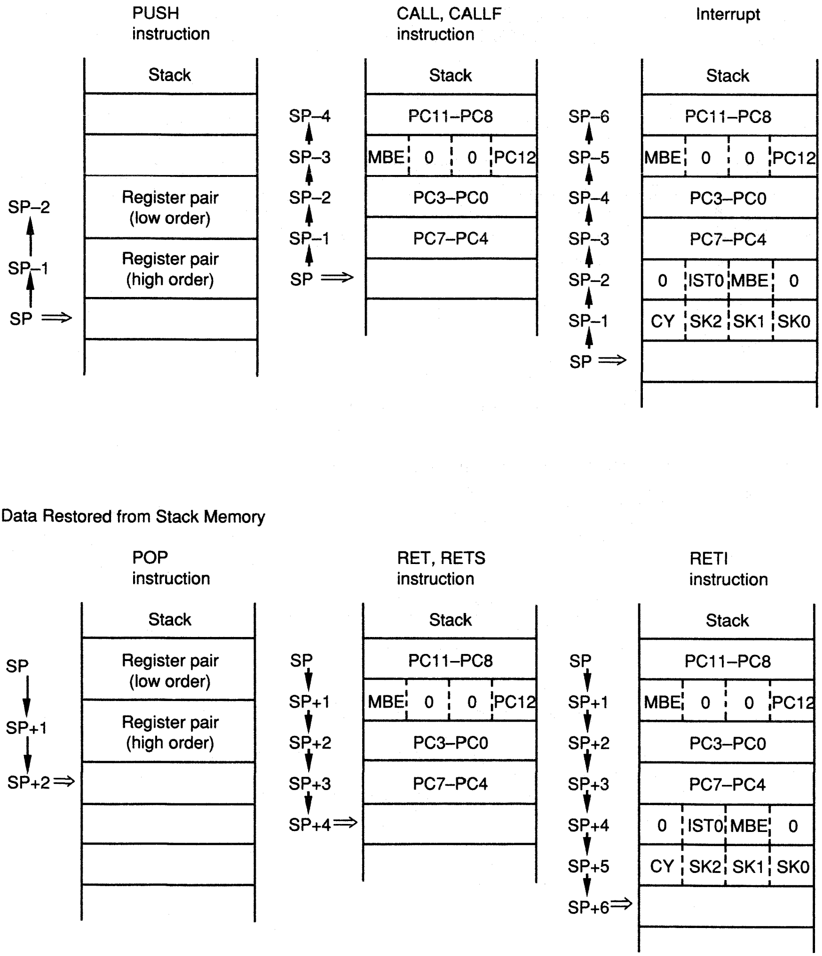
The μPD75268 uses general purpose RAM for stack memory (LIFO). The stack pointer (SP) is an 8-bit register which holds top address information of the stack area.
The stack area addresses are 000H–0FFH of memory bank 0 regardless of how MBE and MBS are set.
SP is decremented before data is saved in the stack memory (write operation); it is incremented after data is restored from the stack memory (read operation).
Fig. 4.6–2 shows data saved in and restored from the stack memory when stack operations are performed.
An initial value is set in SP by using an 8-bit memory operation instruction to determine the stack area to be used. The SP contents can also be read.
SP0 is always set to 0.
It is recommended that the initial value of SP should be set to 00H so that the stack area be used is starting at the most significant address of data memory bank 0 (0FFH).
When the RESET signal is generated, the SP contents become undefined. Be sure to initialize SP to the desired value at the start of the program.

Example: To initialize SP.

```
SEL MB15       ;   or CLR1 MBE
MOV XA, #00H
MOV SP, XA     ;   SP← 00H
```

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| F80H | SP7 | SP6 | SP5 | SP4 | SP3 | SP2 | SP1 | SP0 | SP |
| | | | | | | | | ↑<br>Fixed to 0 | |

**Figure 4.6–1 Stack Pointer Configuration**



(a) Data Saved in Stack Memory

(b) Data Restored from Stack Memory

**Figure 4.6–2 Data Saved / Restored in Stack Operation (μPD75268)**

### 4.7 Program Status Word (PSW) – Eight Bits

The program status word (PSW) consists of flags closely related to processor operation.
PSW is mapped in data memory addresses FB0H and FB1H. Two bits of address FB0Hcan be operated by using a memory operation instruction.

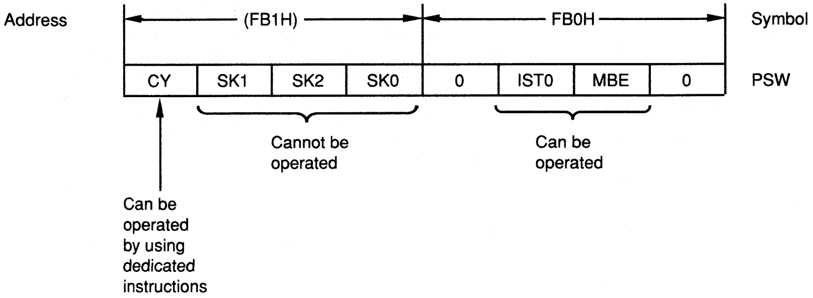


**Figure 4.7–1 Program Status Word Configuration**

**Table 4.7–1 PSW Flags Saved and Restored when Stack Operation is Performed**

| | | Saved or restored flag |
|---|---|---|
| Save | During CALL, CALLF instruction execution | MBE is saved |
| | During hardware interrupt | All PSW bits are saved |
| Restore | During RET, RETS instruction execution | MBE is restored |
| | During RETI instruction execution | All PSW bits are restored |

(1) Carry flag (CY)
The carry is a 1-bit flag that shows the occurence of overflow or underflow during execution of an instruction involving carry (ADDC or SUBC).
The carry flag also serves as a bit accumulator. Boolean algebra operation is performed between the bit accumulator and data memory specified by a bit address. The result can be stored in the bit accumulator.
The carry flag is operated by using dedicated instructions independently of other PSW bits.
When the $\overline{\text{RESET}}$ signal is generated, the carry flag becomes undefined.

**Table 4.7–2 Carry Flag Operation Instructions**

| | Instructions (mnemonic) | Carry flag processing |
|---|---|---|
| Instructions dedicated to carry flag operation | SET1 CY<br>CLR1 CY<br>NOT1 CY<br>SKT CY | CY is set to 1<br>CY is cleared<br>CY is inverted<br>Skip if CY is set to 1 |
| Bit Boolean instructions | AND1 CY, mem*.bit<br>OR1 CY, mem*.bit<br>XOR1 CY, mem*.bit | The specified bit and CY are ANDed, ORed, or XORed together |
| Interrupt service | During interrupt execution | CY is saved in stack memory in parallel with other PSW bits (eight bits) |
| | RETI | CY is restored from stack memory in parallel with other PSW bits |

Remarks: mem*.bit indicates any of the following three bit 3 manipulation addressing modes

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

Example: To AND bit 3 of address 3H and P33 together and set the result in CY.

```
SET1   CY           ;  CY←1
CLR1   MBE          ;  Or SEL MB15
SKT    3FH. 3       ;  Skip if bit 3 of address 3FH is set to 1.
CLR1   CY           ;  CY←0
AND1   CY, PORT3.3  ;  CY←CY ∧ P33
```

(2) Skip flags (SK2, SK1, and SK0)
The skip flags store the skip state and are automatically set or reset when the CPU executes instructions. The user cannot directly use the flags as operands.

(3) Interrupt status flag (IST0)
The interrupt status flag stores the current status of processing being performed. (For details, see Table 4.7-3)

**Table 4.7–3 Interrupt Status Flag Indication Contents**

| IST0 | Status of processing being performed | Processing contents and interrupt control |
|---|---|---|
| 0 | Status 0 | During normal program processing.<br>Every interrupt can be acknowledged. |
| 1 | Status 1 | During interrupt processing.<br>No interrupt must be acknowledged. |

If an interrupt is acknowledged, the IST0 contents are saved in stack memory as a PSW bit, then automatically IST0 is set to 1. When an RETI instruction is executed, IST0 is set to 0.
The interrupt status flag can be operated by using a memory operation instruction. The current status of processing can also be changed under program control.

Caution: Before operating the flag, be sure to execute a DI instruction to disable interrupts. After operating the flag, execute an EI instruction to enable interrupts.

(4) Memory bank enable flag (MBE)

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

MBS can be set or reset at any time by using a bit manipulation instruction regardless of memory bank setting.

```
Example: SET1   MBE   ;   MBE← 1
         CLR1   MBE   ;   MBE← 0
```

When MBE is set to 1, the data memory address space is exceeded and all the data memory space can be addressed.

When MBE is reset to 0, the data memory address space is fixed regardless of how MBS is set. (see Fig. 2.1–2.)

When the $\overline{\text{RESET}}$ signal is generated, the contents of program memory address 0 bit 7 are set and MBE is initialized automatically.

When vectored interrupt service is made, bit 7 of the corresponding vector address table is set and the MBE state during interrupt service is set automatically.

During interrupt service, normally MBE is set to 0 and the general purpose RAM of memory bank 0 is used.

### 4.8 Bank Selection Register (BS)

The memory bank selection register (MBS) for selecting a memory bank is mapped in the bank selection register (BS). The low-order four bits of BS are fixed to 0

MBS is set by using the SEL MBn instructions.

BS can be saved in and restored from the stack area in 8-bit units by using PUSH BS instructions.

| ← MBS → | | | | | | | | Symbol |
|---|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| MBS3 | MBS2 | MBS1 | MBS0 | 0 | 0 | 0 | 0 | BS |

**Figure 4.8–1 Bank Selection Register Configuration**

(1) Memory bank selection register (MBS)

The memory bank selection register (MBS), consisting of four bits, stores high-order 4-bit address information of a 12-bit data memory address. The memory bank to be accessed is specified according to the register contents. However, the μPD75268 allows the user to select bank 0, bank 1, or bank 15 only.

MBS is set by using the SEL MBn instruction (where n is 0, 1, or 15).

The address range applied according to how MBE and MBS are set is as shown in Fig. 2.1–2.

When the $\overline{\text{RESET}}$ signal is generated, MBS is initialized to 0.

## 5. PERIPHERAL HARDWARE FUNCTIONS

### 5.1 Digital I/O Ports

The μPD75268 contain digital I/O ports: ports 0 to 6 and port H.
For the μPD75268, memory-mapped input/output is employed; thus, all I/O ports are mapped in the data memory space. Since the data memory manipulation instruction can be applied to each of the I/O ports, bit-manipulated and 8-bit data input/output are possible in addition to 4-bit input/output, allowing the I/O ports to be controlled in serveral ways.

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| FF0H | P03 | P02 | P01 | P00 | PORT0 |
| FF1H | P13 | P12 | P11 | P10 | PORT1 |
| FF2H | P23 | P22 | P21 | P20 | PORT2 |
| FF3H | P33 | P32 | P31 | P30 | PORT3 |
| FF4H | P43 | P42 | P41 | P40 | PORT4 |
| FF5H | P53 | P52 | P51 | P50 | PORT5 |
| FF6H | P63 | P62 | P61 | P60 | PORT6 |
| 1FFH | PH3 | PH2 | PH1 | PH0 | PORTH |

**Fig. 5.1-1 Digital I/O Port Data Memory Addresses**

The I/O port control instruction are as shown in Table 6.1-2. In addition to 4-bit input/output, 8-bit input/output and bit manipulation can be done, allowing the I/O ports to be controlled several ways.

Example 1: To test the status of P13 and the output value to Ports 4 and 5 according to the test results:

```
SKT   PORT1.3        ; Skips if bit 3 of port 1 is 1.
MOV   XA, #18H       ; XA ← 18H String effect
MOV   XA, #14H       ; XA ← 14H String effect
SEL   MB15           ; or, CLR1 MBE
OUT   PORT4, XA      ; Port 5, 4 ← XA
```

Example 2:
```
SET1  PORT4.@L       ; Of ports 4 to 6, the bit specified by the L register is set to 1.
```

5.1.1 Types of digital I/O ports and either features and configuration
The types of digital I/O ports are as shown in Table 5.1-1.

**Table 5.1-1 Types of Digital Ports and Their Features**

| Port name (Symbol) | Function | operation, features | Notes |
|---|---|---|---|
| Port 0 | 4-bit input | Can be read or tested at any time regardless of the functional mode of shared pins.<br>P10 and P11 have noise eliminate function input. | The port pins are also used for SI, SO, $\overline{\text{SCK}}$, and INT4. |
| Port 1 | | | The port pins are also used for INT0, INT2 and TI0 |
| Port 3<br>Port 6 | 4-bit input/ output. | Can be set for input or output mode in 1-bit units. | Port 6 can incorporate a pull-down resistor (mask option) and its VIH is low; therefore, it is suitable for key input. |
| Port 2 | | Can be set for input or output mode in 4-bit units.<br>Ports 4 and 5 can paired to enable 8-bit data transfer. | P23 also used for BUZ. pin. |
| Port 4<br>Port 5 | | | |
| Port H | 4-bit output | P-ch open-drain high-voltage, high-current output sport. Can directly drive FIPs or LEDs. Pull-down resistors may be built in as mask option. | Shared with T10/S15 to T13/S12 pins. |

Ports 4, 5 can directly drive LED.

P10 and P11 are also used as external vector interrupt input pins, and contain a noise eliminator.
PORT H is shared with display output pins (T10/S15 to T13/S12). In Display Off mode (DSPM.3=0), in the same ways as other display outputs, these pins become low level (when internal pull-down reesistors are internally provided) or high impedance (if no pull-down resistor is provided). Therefore, the display mode must be specified by the display mode register (DSPM) before using PORT H.

The configuration of each port is shown in Figs. 5.1-1 to 5.4-5.

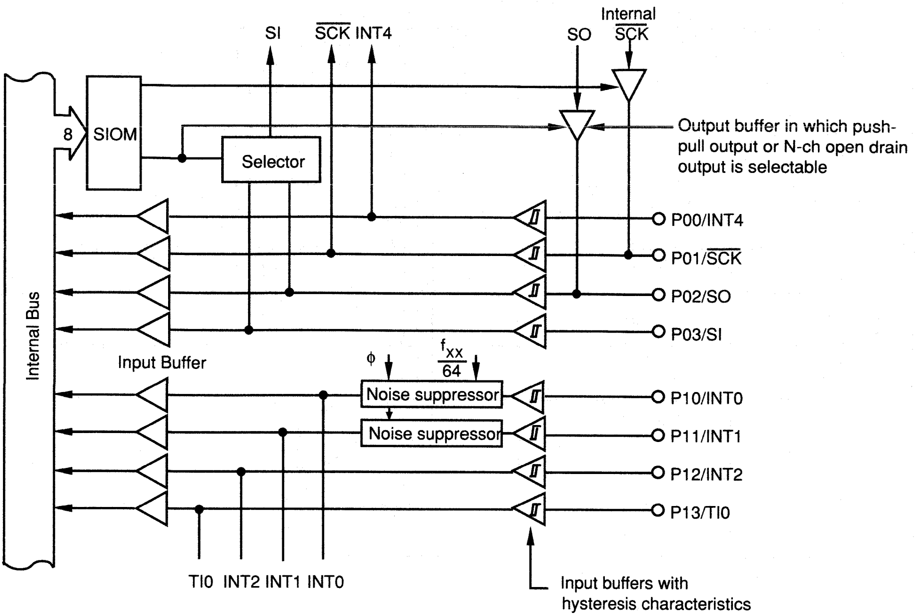


**Fig. 5.1-2 Configuration of Ports 0 and 1 (μPD75268)**

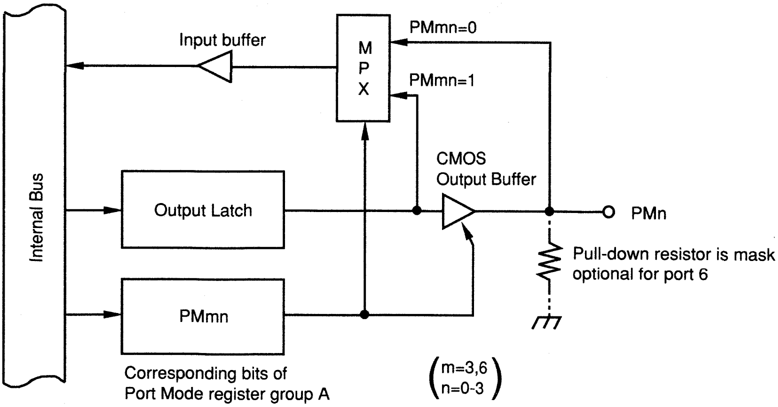

Input buffer
M
P
X
PMmn=0
PMmn=1
CMOS
Output Buffer
Internal Bus
Output Latch
PMn
Pull-down resistor is mask optional for port 6
PMmn
Corresponding bits of Port Mode register group A
m=3,6
n=0-3

**Fig. 5.1-3 Configuration of Ports 3n and 6n (n=0 to 3) (μPD75268)**

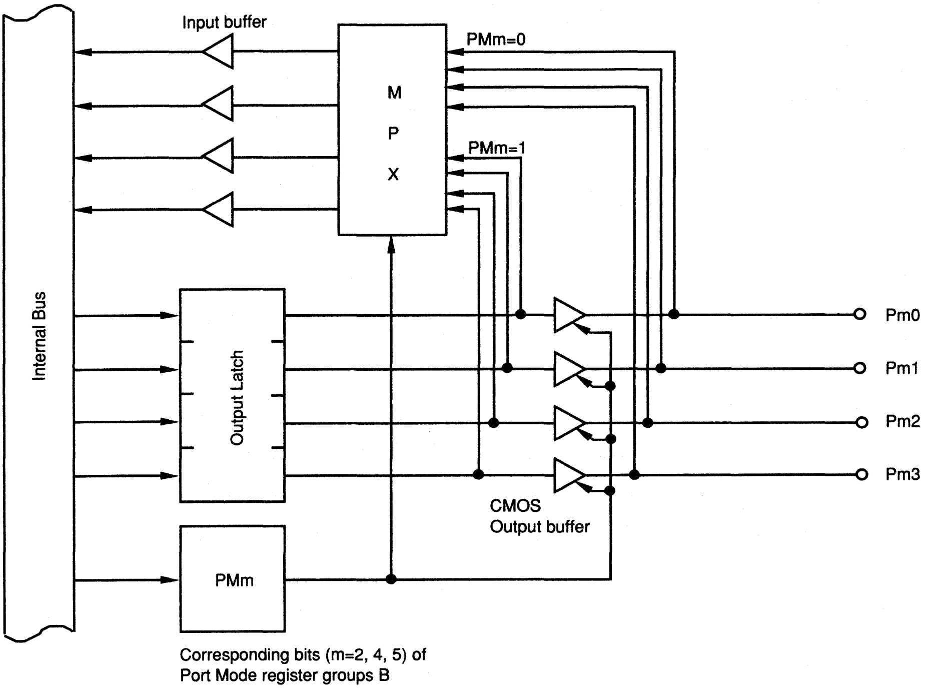


**Fig. 5.1-4 Configuration of Ports 2, 4, 5 (µPD75268)**

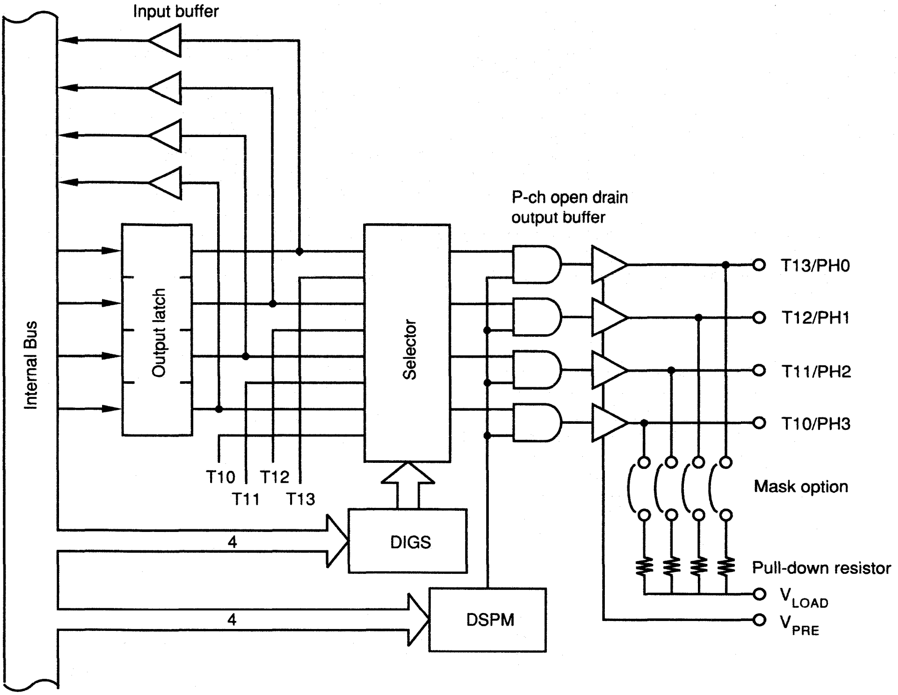


Fig. 5.1-5 Configuration of Port H

5.1.2 Setting input and output modes

The input/output mode of each input/output port is set by using the port mode register as shown in Fig, 5.1-6.

The input or output mode can be specified bitwise for port 3 and 6 by using port mode register group A (PMGA). The input or output mode is specified in 4-bit units for ports 2, 4, 5 by using PMGB.

When the corresponding bit in the port mode register is set to 0, each port functions as an input port.

When the corresponding bit is set to 1, each port functions as an output port. When the output mode is selected by setting the port mode register, at the same time, the contents of output latch are output to output pin. For this reason, setting the contents of output of output latch to the desired value is required before setting the output mode. Each port mode register group (A and B,) can be set by 8-bit memory manipulation instruction. All bits of each port mode register are cleared when an internal reset signal $\overline{\text{RES}}$ is generated, then all output buffers are turned off and all ports are set in input mode.

Example: To use P30, P31, P62 and P63 as input pins, and P32, P33, P60, and P61 as ouput pins

```
CLR1 MBE    ;   or SEL MB15
MOV  XA, #3CH
MOV  PMGA, XA
```

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

| Symbol | PM3n PM6n | P3n, P6n In/Out (n=0–3) |
|---|---|---|
| PMGA | 0 | Input mode (output buffer off) |
|  | 1 | Output mode (output buffer on) |

Port Mode Register Group A

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

| Symbol | PMn | Portn In/Out (n=2, 4, 5) |
|---|---|---|
| PMGB | 0 | Input mode (output buffer off) |
|  | 1 | Output mode (output buffer on) |

Note: –: Don't care

**Fig. 5.1-6 Format of Each Port Mode Register**

5.1.3 Digital I/O port manipulation instructions

Since all I/O ports contained in the μPD75268 are mapped to the data memory space, they can be manipulated with all data memory manipulation instructions.

Table 5.1.2 summarizes the instructions that are particulary effective to I/O port manipulation and their applicable ranges.

(1) Bit manipulation instructions

Since all digital I/O ports can be addressed with the fixed address bit register direct addressing (fmem.bit) and fixed address bit register indirect addressing (pmem.@L) modes, any individual bit on any port can be manipulated with no regard to the MBE and MBS settings.

Example: To OR P50 with P41 together and output the result to P61:

```
         SET    CY             ;  CY← 1
         AND1   CY, PORT5.0 ;  CY← CY ∧ P50
         OR1    CY, PORT4.1 ;  CY← CY ∨ P41
         SKT    CY
         BR     CLRP
         SET1   PORT6.1        ;  P61← 1
           :
           :
  CLRP:  CLR1   PORT6.1        ;  P61← 0
```

(2) Four-bit memory manipulation instructions

All four-bit memory manipulation instructions including IN/OUT, MOV, XCH, ADDS, and INCS are usable. Memory bank 15 must be selected prior to execution.

Example: Outputs the contents of accumulator to Port 3:

```
         SEL    MB15        ;  or CLR1 MBE
         OUT    PORT3, A
```

Example: Adds accumulator value to the data on Port 5 and outputs the result:

```
         SET1   MBE
         SEL    MB15
         MOV    HL, # PORT5
         ADDS   A, @HL      ;  A←  A+PORT5
         NOP
         MOV    @HL, A      ;  PORT5 ← A
```

Example: Tests if the data on Port 4 is larger than the accumulator value:

```
         SET1   MBE
         SEL    MB15
         MOV    HL, #PORT4
         SUBS   A, @HL      ;  A < PORT4
         BR     NO          ;  NO
                            ;  YES
```

(3) Eight-bit manipulation instructions

The IN, OUT, MOV, XCH, and SKE instructions can be used to the ports which allow 8-bit manipulation. Memory bank 15 must be selected prior to instruction execution.

Example: Outputs the BC register pair value to the output port specified by the 8-bit data which was input through Ports 4 and 5:

```
         SET1   MBE
         SEL    MB15
         IN     XA, PORT4 ;  XA←  port 5, 4
         MOV    HL, XA      ;  HL←  XA
         MOV    XA, BC      ;  XA ←  BC
         MOV    @HL, XA     ;  Port(L) ←  XA
```

Table 5.1-2 I/O Pin Manipulation Instruction List

| | | PORT0 | PORT1 | PORT2 | PORT3 | PORT4 | PORT5 | PORT6 | PORTH (3) |
|---|---|---|---|---|---|---|---|---|---|
| 4/8-bit Input | IN A, PORTn (1) | O | | | | | | | MOV A, PORTH |
| | IN XA, PORTn (1) | — | | — | | O | | O | — |
| 4/8-bit Output | OUT PRTn, A (1) | — | — | O | | | | | MOV PORTH, A |
| | OUT PORTn, XA (1) | — | | — | | O | | — | — |
| 1-bit Output | SET1 PORTn. bit | — | — | O | | | | | SET1 PORTH. bit |
| | SET1 PORTn. @L (2) | — | — | O | | | | | — |
| | CLR1 PORTn. bit | — | — | O | | | | | CLR1 PORTH. bit |
| | CLR1 PORTn. @L (2) | — | — | O | | | | | — |
| 1-bit Test | SKT PORTn. bit | O | | | | | | | SKT PORTH. bit |
| | SKT PORTn. @L (2) | O | | | | | | | — |
| | SKF PORTn. bit | O | | | | | | | SKF PORTH. bit |
| | SKF PORTn. @L (2) | O | | | | | | | — |
| 1-bit Operation | AND1 CY, PORTn. bit | O | | | | | | | AND1 CY, @H+PORTH. bit |
| | AND1 CY, PORTn. @L (2) | O | | | | | | | — (4) |
| | OR1 CY, PORTn. bit | O | | | | | | | OR1 CY, @H+PORTH. bit |
| | OR1 CY, PORTn. @L (2) | O | | | | | | | — (4) |
| | XOR1 CY, PORTn. bit | O | | | | | | | XOR1 CY, @H+PORTH. bit |
| | XOR1 CY, PORTn. @L (2) | O | | | | | | | — (4) |

Notes:
1. Setting MBE=0 or (MBE=1 and MBS=15 ) is required before executing any of the above instrucions.
2. Indirectly specify the lower 2 bits of address and bit address by the L register.
3. MBE=1 and MBS=1 must be selected before execution.
4. Value 0FH must be set in the H register before execution.

5.1.4 Digital I/O port function

When a data memory manipulation instruction is executed, the functions of each port and their pins differ depending on the I/O mode setting as shown in Table 5.1-3.

As can be understood from the I/O port configuration, this is because data to be placed to the internal bus are data of each pin in the input mode, where as they are data of output latch in the output mode.

(1) Function in the input mode

When executing a test instruction (such as SKT instruction), or an instruction used to input port data into the internal bus in 4 or 8 bits (such as IN, OUT, operation instruction, and compare instruction), the data of the respective pins are manipulated.

When executing an instruction (OUT or MOV instruction) used to transfer the contents of the accumulator to port, the data of the accumulator is latched by the output latch. The output buffer remains off.

When the XCH instruction is executed, data of each port pins are input to the accumulator are latched to the output latch. The output buffer remains off. When executing the INCS instruction, the 4-bit data of each port pins incremented by 1 is latched to the output latch. The output buffer remains off.

When executing an instruction that rewrites the data memory in bit units such as SET1, CLR1, and SKTCLR instructions, the specified data can be rewritten as instructed by the instruction; however, the contents of output latch of other bits becomes undefined.

(2) Function in the output mode

When executing an instruction to input port data into the internal bus in 4 or8 bits using the test or bit input instruction by which the contents of the accumulator are transferred in 4 or 8 bits, data of the output latch is rewritten and at the same time, it is output from the pins.

When the XCH instruction is executed, the contents of the output latch are transferred to the accumulator and the contents of the accumulator are latched to the output latch and output from the pins.

When the INCS instruction is executed, the contents of the output latch incremented by 1 is latched to the output latch and output from the pins.

When a bit output instruction is executed, the bit of the specified output latch is rewritten and output from the pin.

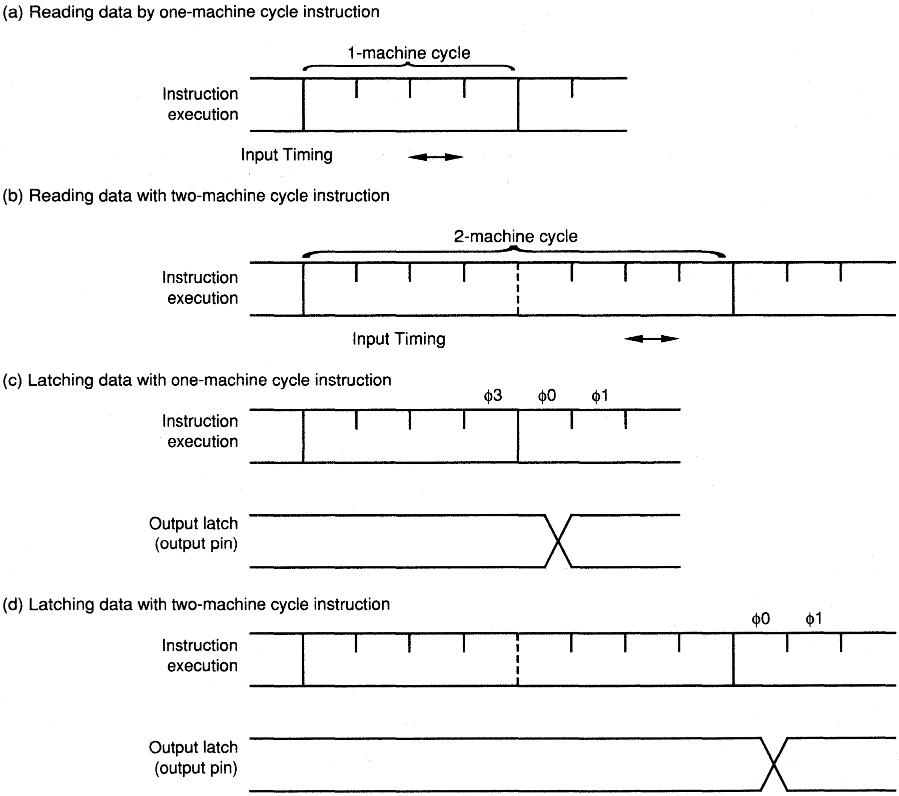
**Table 5.1-3 I/O Port Manipulation**

| Instruction | Port and pin function | |
|---|---|---|
| | Input mode | Output mode |
| SKT ①<br>SKF ① | Tests data of pin | Tests data of output latch |
| AND1 CY, ①<br>OR1 CY, ①<br>XOR1 CY, ① | Performs logical operation between data of pin and CY | Performs logical operation between data of output latch and CY |
| IN A, PORTn<br>IN XA, PORTn<br>MOV A, @HL<br>MOV XA, @HL | Transfers data of pins to accumulator | Transfers data of output latch to accumulator |
| ADDS A, @HL<br>ADDC A, @HL<br>SUBS A, @HL<br>SUBC A, @HL<br>AND A, @HL<br>OR A, @HL<br>XOR A, @HL | Performs logical operation between data of pins and contents of accumulator | Performs logical operation between data of output latch and contents of accumulator |
| SKE A, @HL | Compares data of pins and contents of accumulator | Compares data of output latch with contents of accumulator |
| OUT PORTn, A<br>OUT PORTn, XA<br>MOV @HL, A<br>MOV @HL, XA | Transfers contents of accumulator to output latch (Output buffer remains off) | Transfers contents of accumulator to output latch and outputs from pins |
| XCH A, PORTn<br>XCH XA, PORTn<br>XCH A, @HL<br>XCH XA, @HL | Transfers data of pins to accumulator and transfers contents of accumulator to output latch (output buffer remains off) | Exchanges data between output latch and accumulator |
| INCS PORTn<br>INCS @HL | Latches data of pins plus 1 to output latch | Increments the contents of output latch by 1 |
| SET1 ①<br>CLR1 ①, CY<br>MOV1 ①<br>SKTCLR ① | Rewrites output latch of the specified bit as instructed by instruction, but output latches for other bits become undefined | Changes output pin state in accordance with the instruction |

Note: ① in the column of instruction denotes two addressing modes: PORTn.bit and PORTn.@.

5.1.5 Digital I/O port input/output timings

Fig. 5.1-7 shows the data output timing to output latch and the output latch data read timing into the internal bus.



(a) Reading data by one-machine cycle instruction

(b) Reading data with two-machine cycle instruction

(c) Latching data with one-machine cycle instruction

(d) Latching data with two-machine cycle instruction

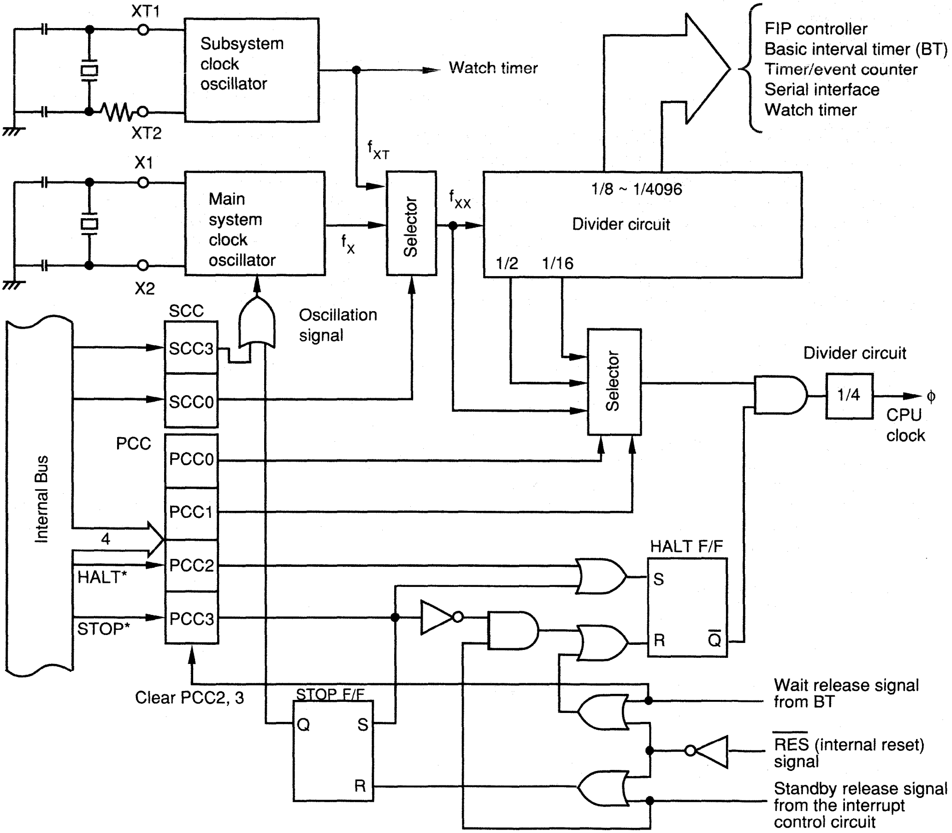
**Fig. 5.1-7 Digital I/O Port Input/Output Timing**

### 5.2 Clock Generator

The clock generator supplies clocks to the CPU and on-chip peripherals and controls CPU operation modes.

5.2.1 Clock generator configuration

The clock generator configuration is shown in Fig. 5.2-1.



Notes: 1. $f_X$: Mainsystem clock frequency
2. $f_{XT}$: Subsystem clock frequency
3. $f_{XX}$: System clock frequency
4. PCC: Processor clock control register
5. SCC: System clock control register
6. * denotes instruction execution.

**Fig. 5.2-1 Block Diagram of Clock Generator**

### 5.2.2 Clock generator function and operation

The clock generator produces the following clocks and controls CPU operating modes such as standby:
— Main system clock $f_X$
— Subsystem clock $f_{XT}$
— CPU clock $\phi$
— Clock to peripheral hardware

The clock generator operates according to how the processor clock control register (PCC) and system clock control register (SCC) are set, as described below:

(a) When the RESET signal is generated, the minimum speed mode of the main system clock (15.3 μ/4.19 MHz) is selected. (PCC = 0 and SCC = 0)

(b) When the main system clock is selected, one of three CPU clock frequencies can be selected (0.95 μs, 1.91 μs, and 15.3 μs/ 4.19 MHz) by setting PCC.

(c) When the main system clock is selected, the standby mode (STOP or HALT) can be used.

(d) Subsystem clock is selected by setting SCC.0, and operation can be performed in a very low-speed and low current consumption (122 μs/32.768 kHz). In this case, the PCC setup value does not affect the CPU clock.

(e) When the subsystem clock is selected, main system clock oscillation can be stopped by setting SCC.3. The HALT mode can also be used, but the STOP mode cannot be used. (Subsystem clock oscillation cannot be stopped).

(f) The main system clock is divided to generate a clock supplied to peripheral hardware. The subsystem clock can be supplied directly only to the watch timer. Thus, the watch function can continue operation in the standby mode.

(g) When subsystem clock is selected, the watch timer can continue normal operation. However, the FIP controller driver will not operate. When the main system clock is stopped, other hardware devices cannot be used because they operate according to the main system clock.

(1) Processor clock control register (PCC)

PCC consists of four bits; the low-order two bits are used to select CPU clock $\phi$, and the high-order two bits are used to control CPU operation mode. (See Fig. 5.2-2.)
When bit 3 or 2 is set to (1), the standby mode is set. When the standby mode is released by the standby mode release signal, automatically bits 3 and 2 are cleared and the normal operating mode is entered.
The low-order two bits of PCC are set by using a 4-bit memory operation instruction (the high-order two bits are set 0).
Bits 3 and 2 are set to (1) by using STOP and HALT instructions respectively.
The STOP and HALT instructions can always be executed independently of the MBE contents.
CPU clock can be selected only during main system clock operation. When the μPD75268 is operated on the subsystem clock, the low-order two bits of PCC becomes invalid, and the CPU clock is fixed to $f_{XT}/4$. The stop instruction is also enabled only during main system operation.

Example 1: To set a machine cycle to 0.95 μs (4.19) MHz).

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

Example 2: To set STOP mode. (Be sure to enter a NOP instruction following the STOP or HALT instruction.)

```
STOP
NOP
```

When the RESET signal is generated, PCC is cleared.

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

CPU clock selection bits

| | | SCC = 0<br>The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz | | SCC = 1<br>The value enclosed in parentheses is applied when $f_{XT}$ = 32.768 kHz | |
|---|---|---|---|---|---|
| | | CPU clock frequency | 1 machine cycle | CPU clock frequency | 1 machine cycle |
| 0 | 0 | $\Phi = f_X/64$<br>(65.5 kHz) | 15.3 μs | $\Phi = f_{XT}/4$<br>(8.192 kHz) | 122 μs |
| 0 | 1 | Undefined | – | | |
| 1 | 0 | $\Phi = f_X/8$<br>(524 kHz) | 1.91 μs | | |
| 1 | 1 | $\Phi = f_X/4$<br>(1.05 MHz) | 0.95 μs | | |

$f_X$ : Main system clock oscillator output frequency

$f_{XT}$: Subsystem clock oscillator output frequency

CPU operation mode control bits

| | | |
|---|---|---|
| 0 | 0 | Normal operating mode |
| 0 | 1 | HALT mode |
| 1 | 0 | STOP mode |
| 1 | 1 | Undefined |

**Figure 5.2–2 Processor Clock Control Register Format**

(2) System clock control register (SCC)

SCC consists of four bits; the least significant bit is used to select CPU clock, and the most significant bit is used to control (stop) main system clock oscillation. (See Fig. 5.2–3.)

SCC.0 and SCC.3 exist at the same data memory address; the bits cannot be changed at the same time. Thus, SCC.0 and SCC.3 are set by using a bit operation instruction. SCC.0 and SCC.3 can always be operated independetly of the MBE contents.

Main system clock oscillation can be stopped by setting SCC.3 only during subsystem clock operation. Main system clock oscillation is stopped by using the STOP instruction during main system clock operation.

When the $\overline{\text{RESET}}$ signal is generated, SCC is cleared.

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| FB7H | SCC3 | – | – | SCC0 | SCC |

| SCC3 | SCC0 | CPU clock slection | Main system clock oscillation |
|---|---|---|---|
| 0 | 0 | Main system clock | Oscillation is enabled. |
| 0 | 1 | Subsystem clock | |
| 1 | 0 | Undefined | |
| 1 | 1 | Subsystem clock | Oscillation is stopped. |

**Figure 5.2–3 System Clock Control Register Format**

Caution 1: Changing the system clock requires a maximum of $1/f_{XT}$ time. To stop main system clock after changing the subsystem clock, set SCC.3 after the machine cycle or cycles listed in Table 5.2–1.

2: Even if oscillation is stopped by setting SCC.3 during main system clock operation, normal STOP mode is not entered.

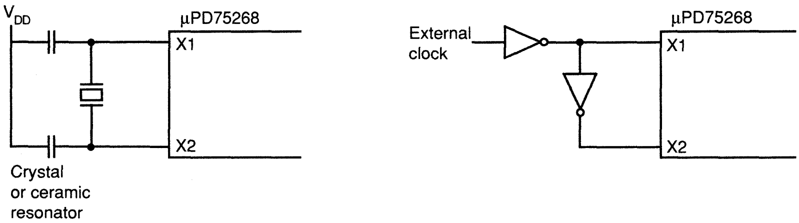
3: When SCC.3 is set to 1, X1 input is connected internally to $V_{SS}$, to avoid leakage current due to crystall oscillator. When the external clock is used in main system clock, do not set SCC.3 to 1.

4: When the system clock is switched to subsystem clock it is necessary to disable any interrupt during the time given in table 5.2-1.

(3) System clock oscillators

The main system clock oscillator uses a crystal oscillator (4.194304 MHz standard) or cermanic oscillator connected to the X1 and X2 pins.

An external clock can also be input.


(a) Crystal or ceramic resonator

(b) External clock

**Figure 5.2–4 External Circuit to Main System Clock Oscillator**

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

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

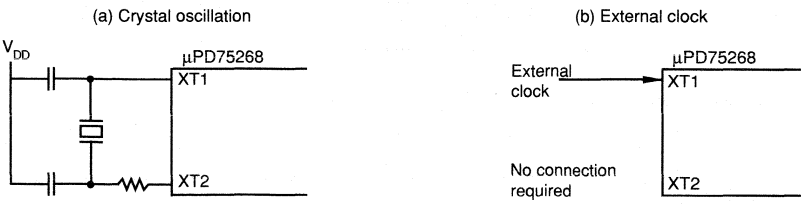


**Figure 5.2–5 External Circuit to Subsystem Clock Oscillator**

(4) Frequency divider
The frequency divider divides the main system clock oscillator output ($f_X$) and generates various clocks.

5.2.3 System clock and CPU clock setting

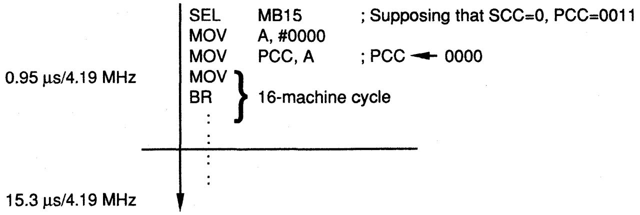
(1) Time required to change system and CPU clocks
The system and CPU clocks can be changed by using the low-order two bits of PCC and the least significant bit of SCC. However, this clock change is not immediatly made after the register are rewritten, and the clock before the clock change is made is used for operation during given machine cycles. Thus, to stop main system clock oscillation, a STOP instruction must be executed or SCC.3 must be set after the change time elapses.

**Table 5.2–1 Maximum Time Required to Change System and CPU Clocks**

| Setup value before change | | | Setup value after change | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCC | PCC | PCC | SCC0 | PCC1 | PCC0 | SCC0 | PCC1 | PCC0 | SCC0 | PCC1 | PCC0 | SCC0 | PCC1 | PCC0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | | | | 1 machine cycle | | | 1 machine cycle | | | Inhibited | | |
| | 1 | 0 | 8 machine cycles | | | | | | 8 machine cycles | | | Inhibited | | |
| | 1 | 1 | 16 machine cycles | | | 16 machine cycles | | | | | | $\frac{f_X}{4f_{XT}}$ machine cycles<br>(32 machine cycles) | | |
| 1 | 1 | 1 | Inhibited | | | Inhibited | | | $\frac{f_X}{4f_{XT}}$ machine cycles<br>(32 machine cycles) | | | | | |

The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz and $f_{XT}$ = 32.768 kHz.

Remarks: The CPU clock Φ is supplied to the μPD75268 internal CPU. The reciprocal of the clock becomes the minimum instruction time. (The present manual defines it to be one machine cycle.)



```
SEL     MB15      ; Supposing that SCC=0, PCC=0011
MOV     A, #0000
MOV     PCC, A    ; PCC ← 0000
MOV }
BR  } 16-machine cycle
 :
 :
 :
 :
```

(2) System and CPU clock change sequence
System and CPU clock change is explained using Fig. 5.2–6.

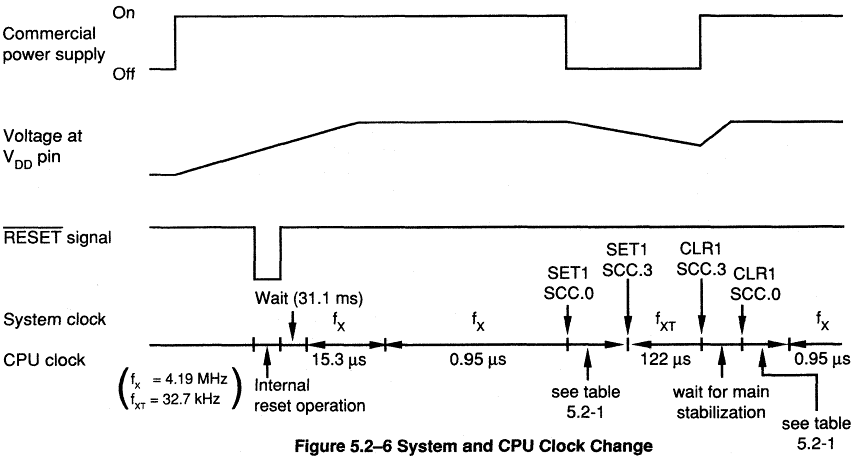


**Figure 5.2–6 System and CPU Clock Change**

1) When the $\overline{\text{RESET}}$ signal is generated, the CPU starts operating at the minimum speed of the main system clock (15.3 µs / 4.19 MHz) after the wait time (31.3 ms / 4.19 MHz) to allow time for stabilizing the oscillator.

2) After enough time has elapsed for the $V_{DD}$ pin voltage to rise to an adequate level PCC is rewritten and the µPD75268 operates at the maximum speed.

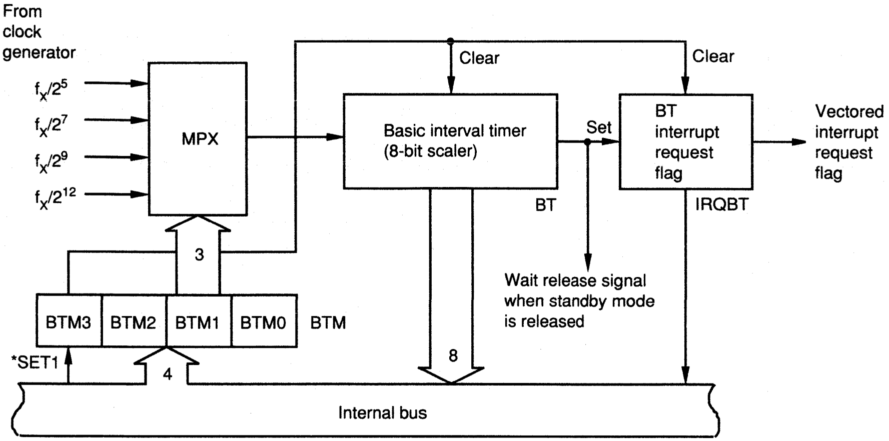
3) Turning off the commercial power supply is detected by using interrupt input (INT4 is useful); SCC. 0 is set, and the µPD75268 operates using the subsystem clock. (At the time, a check must have been made to ensure that subsystem clock oscillation has started. After the time required to change to the subsystem clock (32 machine cycles) has elapsed, SCC 3 is set and main system clock oscillation is stopped.

4) Restoration of the commercial power supply is detected by using an interrupt. SCC. 3 is cleared and main system clock oscillation is started. After the time required to stabilize oscillation has elapsed, SCC. 0 is cleared and the µPD75268 operates at the maximum speed.

## 5.3 Basic Interval Timer

5.3.1 Basic interval timer configuration

Fig. 5.3–1 shows the configuration of the basic interval timer


From clock generator
$f_X/2^5$
$f_X/2^7$
$f_X/2^9$
$f_X/2^{12}$
MPX
3
Clear
Clear
Basic interval timer (8-bit scaler)
Set
BT interrupt request flag
Vectored interrupt request flag
BT
IRQBT
Wait release signal when standby mode is released
BTM3
BTM2
BTM1
BTM0
BTM
*SET1
4
8
Internal bus

Remarks: * denotes instruction executing.

**Figure 5.3–1 Basic Interval Timer Configuration**

5.3.2. Basic interval timer mode register (BTM)

BTM is a 4-bit register for controlling operation of the basic interval timer.
BTM is set by using a 4-bit memory operation instruction.
Bit 3 can be set individually by using a bit operation instruction.

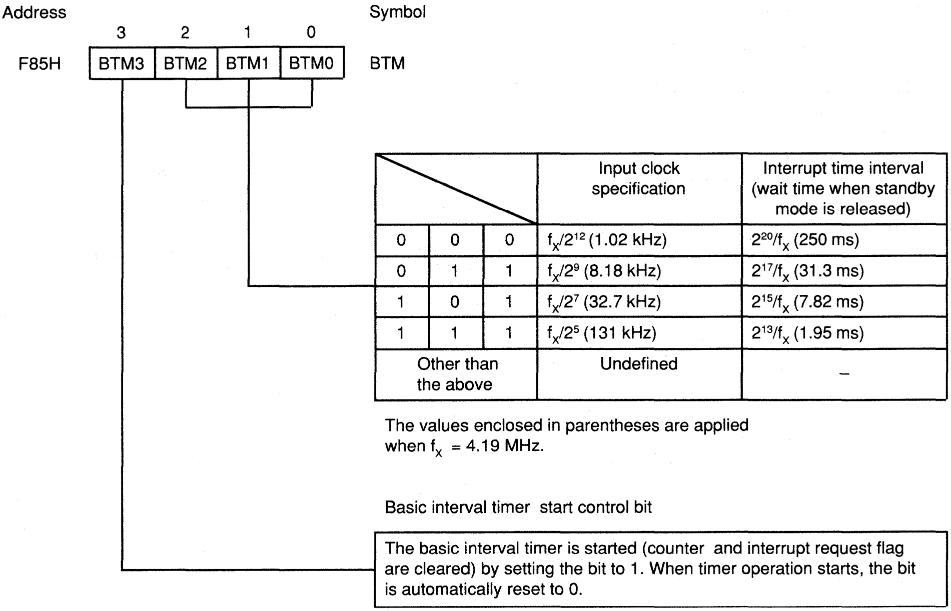
Example 1: To set the interrupt generation interval to 1.95 ms (4.19 MHz).

```
SEL    MB15        ; or CLR1 MBE
MOV    A, # 1111B
MOV    BTM, A      ; BTM←— 1111B
```

Example 2: To clear BT and IRQBT (watchdog timer application).

```
SEL    MB15        ; or CLR1 MBE
SET1   BTM.3       ; BTM BIT 3 is set to 1.
```

When bit 3 is set to 1, the basic interval timer contents are cleared. At the same time, the basic interval interrupt request flag (IRQBT) is also cleared. (The basic interval timer starts.)
When the $\overline{\text{RESET}}$ signal is generated, the BTM contents are cleared and interrupt request signal generation is set for the longest interval.



|  |  |  | Input clock specification | Interrupt time interval (wait time when standby mode is released) |
|---|---|---|---|---|
| 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 | $f_X/2^7$ (32.7 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 than the above |  |  | Undefined | – |

The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz.

Basic interval timer start control bit

The basic interval timer is started (counter and interrupt request flag are cleared) by setting the bit to 1. When timer operation starts, the bit is automatically reset to 0.

**Fig. 5.3-2 Basic Interval Timer Mode Register Format**

5.3.3. Basic interval timer operation

The basic interval timer (BT) is incremented each time a pulse is received from the clock generator. When an overflow occurs, the interrupt request flag (IRQQBT) is set. BT count operation cannot be stopped.
An interrupt generation time interval can be selected from among four types by setting BTM. (See Fig. 5.3—2.)
The basic interval timer and interrupt request flag can be cleared by setting BTM bit 3 to 1 (the interval timer function starts).
The basic interval timer (BT) count can be read by using an 8-bit operation instruction. Data cannot be written into BT.

Caution:
To prevent reading of unstable data during count update when reading the basic interval timer count, execute the read instruction twice and compare the results. If they are valid values, the later read value is used as the read result, if they differ from each other completely, reexecute from the beginning.

Example: To read the BT count

```
        SET1   MBE
        SEL    MB15
        MOV    HL, #BT     ;   BT address is set in HL.
LOOP:   MOV    XA, @HL     ;   First read
        MOV    BC, XA
        MOV    XA, @ HL    ;   Second read
        SKE    A, C
        BR     LOOP
        MOV    A, X
        SKE    A, B
        BR     LOOP
```

The wait function is provided to stop CPU operation until the basic interval timer overflows in order to allow time for system clock oscillation to become stable when the STOP mode is released.

Although the wait time after the $\overline{\text{RESET}}$ signal is generated is fixed, it can be selected by setting BTM when the STOP mode is released by an interrupt. The wait time is selected by setting BTM as shown in Fig. 5.3-2. BTM setting must be performed before the STOP mode is set.

5.3.4 Basic interval timer application examples

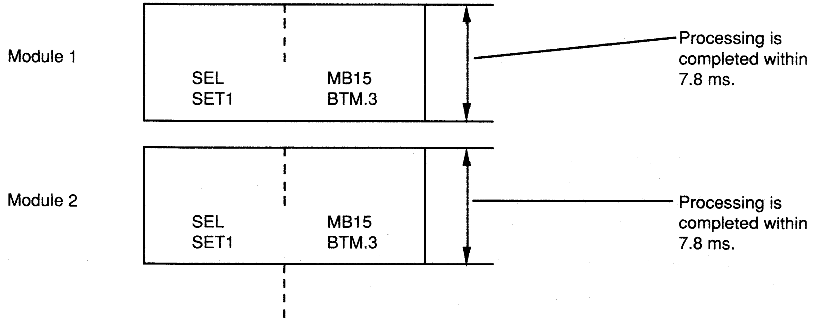
Example 1: To enable basic interval timer interrupt and set interrupt generation interval at 1.95 ms (at 4.19 MHz).

```
SEL     MB 15
MOV     A, #1111B
MOV     BTM, A      ;   BTM setting and timer function start
EI                  ;   Interrupt is enabled.
EI      IEBT        ;   BT interrupt is enabled.
```

Example 2: Watchdog timer applciation

A program is divided into several modules which terminate processing within the BT setup time. BT and IRQBT are cleared at the end of each module. If an interrupt is generated, overrun is asumed to have occured.

```
         SEL     MB15
Initial- MOV     A, #1101B   ;   7.8 ms interval is set.
izatiion MOV     BTM, A      ;   BTM setting and timer function start
         EI
         EI      IEBT
```



Module 1
SEL MB15
SET1 BTM.3
Processing is completed within 7.8 ms.
Module 2
SEL MB15
SET1 BTM.3
Processing is completed within 7.8 ms.

Example 3: To set the wait time, when the STOP mode is released, using an interrupt to 7.8 ms.

```
SEL     MB15        ;   or CLR1 MBE
MOV     A, #1101B
MOV     BTM, A      ;   BTM ← 1101B
STOP                ;   STOP mode is set
NOP
```

Example 4:
To set the high-level width of a pulse input to INT4 interupt (both rising and falling edge detection). (The pulse width must not exceed the BT setup value. The BT setup value must be 7.8 ms or more.)
< INT4 interrupt routine (MBE = 0) >

```
LOOP:  MOV    XA, BT      ;  First read
       MOV    BC, XA      ;  Data is stored.
       MOV    XA, BT      ;  Second read
       SKE    A, C
       BR     LOOP
       MOV    A, X
       SKE    A, B
       BR     LOOP
       SKT    PORT0.0     ;  P00=1?
       BR     AA          ;  NO
       MOV    XA, BC      ;  Data is stored in data memory.
       MOV    BUFF, XA
       CLR1   FLAG
       RETI
  AA:  MOV    HL, #BUFF
       MOV    A, C
       SUBC   A, @HL
       INCS   L
       MOV    C, A
       MOV    A, X
       SUBC   A, @HL
       MOV    B, A
       MOV    XA, BC
       MOV    BUFF, XA    ;  Data is stored.
       SET1   FLAG        ;  Data existence flag is set.
       RETI
```

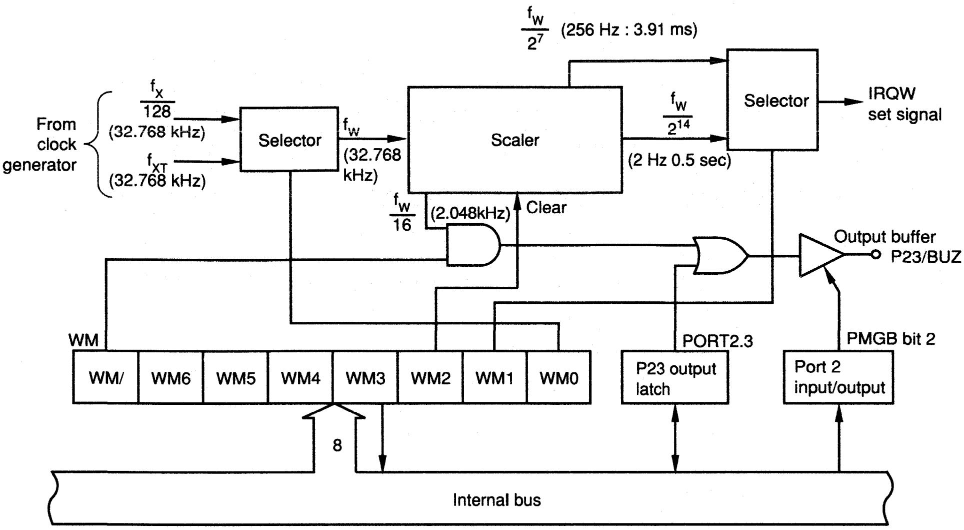
### 5.4 Watch Timer

The μPD75268 incorporates a watch timer (one channel) which has the following function:

(a) The test flag (IRQW) is set at 0.5 s time intervales. The standby mode can be released using IRQW.
(b) The main system and subsystem clocks can be assigned 0.5-s intervals.
(c) In the rapid feed mode, time intervals multiplied by 128 (3.91 ms) are enabled. The mode is useful for program debugging and testing.
(d) A fixed frequency (2.048 kHz) can be output to P23/BUZ. It can be used for sounding the buzzer and trimming the system clock oscillation frequency.
(e) Since the scaler can be cleared, the watch can be started at zero seconds.

5.4.1 Watch timer configuration

Fig. 5.4–1 shows a watch timer block diagram.



The values enclosed in parentheses are applied when $f_X$ = 4.194304 MHz and $f_{XT}$ = 32.768 kHz.

**Figure 5.4–1 Watch Timer Block Diagram**

5.4.2 Watch mode register

The watch dog timer (WM) consists of eight bits to control the watch timer. Fig. 5.4–2 shows the watch mode register format. The watch mode register is set using an 8-bit operation instruction. When the RESET signal is generated, all bits are cleared.

Example: To produce time using the main system clock (4.19 MHz). To enable buzzer output.

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

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

Count clock ($f_W$) selection bit

| | | |
|---|---|---|
| WM0 | 0 | System clock dividing output $\frac{f_X}{128}$ is selected. |
| | 1 | Subsystem clock $f_{XT}$ is selected. |

Operation mode selection bit

| | | |
|---|---|---|
| WM1 | 0 | Normal watch mode (IRQW is set by using $\frac{f_W}{2^{14}}$ , 0.5 s) |
| | 1 | Rapid feed watch mode (IRQW is set by using $\frac{f_W}{2^7}$ , 3.91 ms) |

Watch operation enable/disable bit

| | | |
|---|---|---|
| WM2 | 0 | Watch operation is stopped (divider is cleared). |
| | 1 | Watch operation is enabled. |

BUZ output enable/disable bit

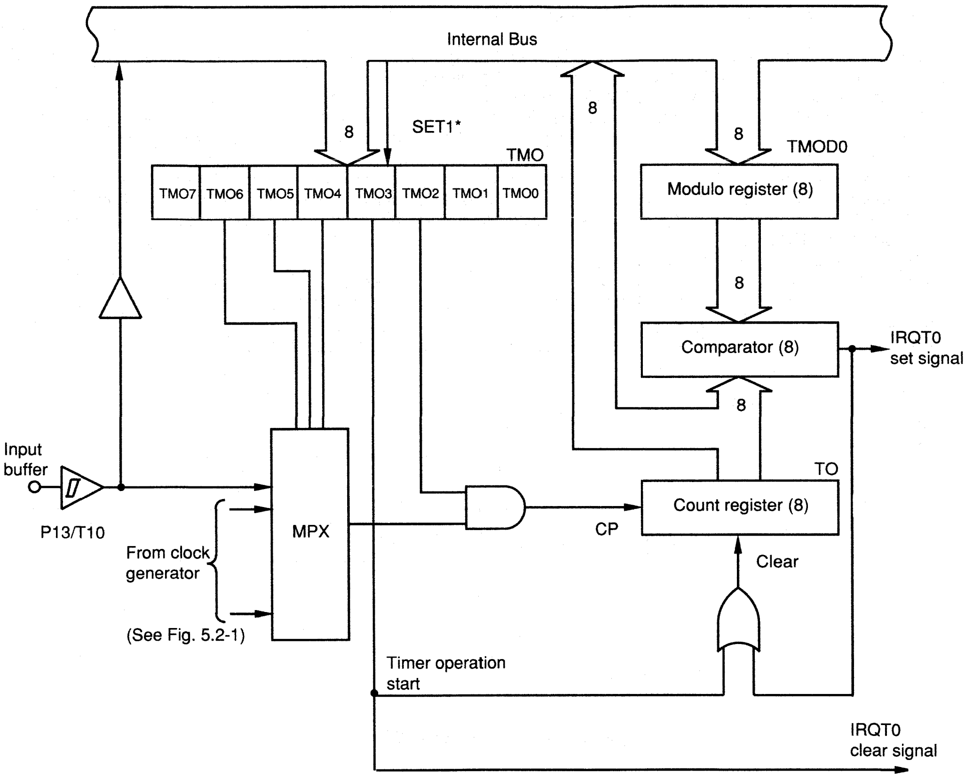
| | | |
|---|---|---|
| WM7 | 0 | BUZ output is disabled. |
| | 1 | BUZ output is enabled. |

**Figure 5.4–2 Watch Mode Register Format**

### 5.5 Timer/event counter configuration

The μPD75268 incorporates a timer/event counter (one channel) as shown in Fig. 5.5–1.
The timer/event counter functions are as follows:

(a) Programmable interval timer operation
(b) Event counter operation
(c) Count read function



Remarks: *denotes instruction executing

**Fig. 5.5-1 Timer/event Counter Block Diagram**

5.5.2 Basic configuration and operation of timer/event counter

The timer/event counter operation mode can be selected by using the timer/event counter mode register (TM0). The basic configuration and operation of the timer/event counter are explained below:

(1) Count pulse CP is selected by setting TM0 and input to the 8-bit count register T0.

(2) T0 is a binary 8-bit up counter incremented by one when CP is input. It is cleared when the RESET signal is generated, TM0 bit 3 is set (timer start), or coincidence signal is generated. T0 can be read at any time using an 8-bit memory operation instruction, but cannot be written

(3) The modulo register TMOD0 consists of eight bits to determine the T0 count. A value is set in TMOD0 using an 8-bit memory operation instruction, but TMOD0 cannot be read. When the RESET signal is generated, TMOD0 is initialized to FFH

(4) The comparator compares the T0 and TMOD0 contents. If they match, it generates a coincidence signal and sets the interrupt request flag (IRQT0)
Fig. 5.5-2 shows the count operation timing.

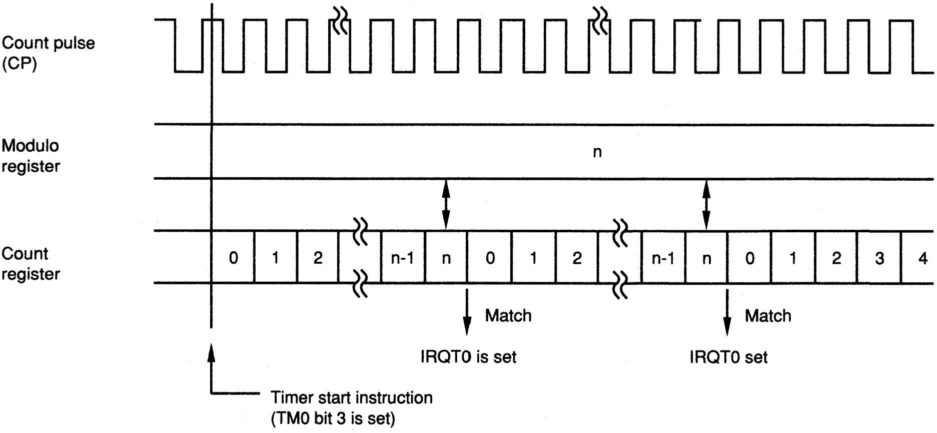


**Figure 5.5–2 Count Operation Timing Chart**

5.5.3 Timer/event counter mode register (TM0)

The mode register (TM0) consists of eight bits to control the timer/event counter. Fig. 5.5–3 shows the timer/event counter mode register format.
The timer mode register is set by using an 8-bit memory operation instruction.
Bit 3 is a timer start bit which can be set individually. Bit 3 is reset to 0 automatically when timer operation starts.

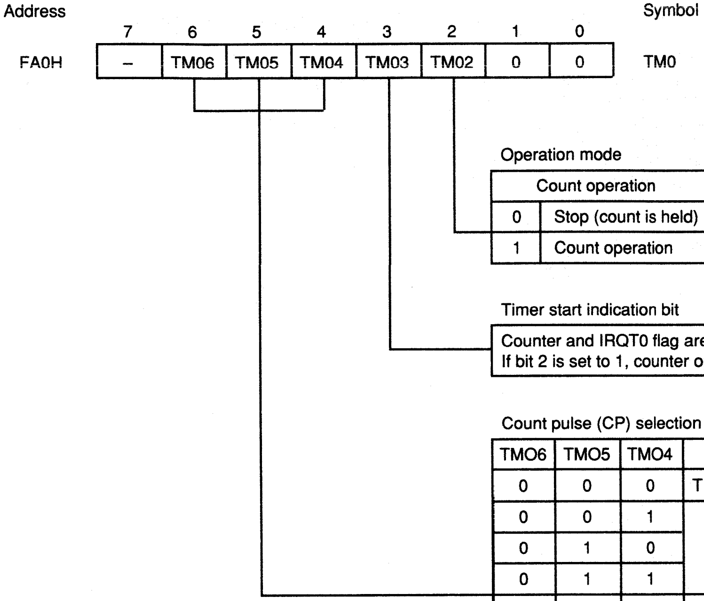
Example 1: To start the timer in interval timer mode with CP = 4.09 kHz.

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

Example 2: To restart the timer according to how the timer mode register is set.

```
SEL    MB15   ;   or CLR1 MBE
SET1   TM0.3  ;   TM0.BIT3 ← 1
```

When the $\overline{\text{RESET}}$ signal is generated, all the timer mode register bits are cleared.



Operation mode

| Count operation | |
|---|---|
| 0 | Stop (count is held) |
| 1 | Count operation |

Timer start indication bit

| Counter and IRQT0 flag are cleared by setting the bit to 1.<br>If bit 2 is set to 1, counter operation is started. |
|---|

Count pulse (CP) selection bit

| TMO6 | TMO5 | TMO4 | Count pulse (CP) |
|---|---|---|---|
| 0 | 0 | 0 | TIn input rising edge |
| 0 | 0 | 1 | Undefined |
| 0 | 1 | 0 | |
| 0 | 1 | 1 | |
| 1 | 0 | 0 | $f_X/2^{10}$ (4.09 kHz) |
| 1 | 0 | 1 | Undefined |
| 1 | 1 | 0 | $f_X/2^6$ (65.5 kHz) |
| 1 | 1 | 1 | Undefined |

Remarks: The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz.

**Figure 5.5–3 Timer/Event Counter Mode Register Format**

5.5.4 Timer/event counter operation mode

The timer/event counter operates in the count operation stop or count operation mode depending on how the mode register is set. The following are always enabled independently of how the mode register is set

1) TI0 pin signal input and test. (P13 pin input test can be made.)
2) Modulo register (TMOD0) setting.
3) Count register (T0) read.

(a) Count operation stop mode
The count operation stop mode is set when TM0 bit 2 is set to 0. Since count pulse (CP) supply to the count register is stopped, count operation is not performed

(b) Count operation mode
The count operation mode is set when TM0 bit 2 is set to 1. Count pulses selected by using bits 4 to 6 are supplied to the count register, and count operation is performed as shown in Fig. 5.5–2.
Normally, timer operation is started by

1) setting a count value in the modulo register (TMOD0), then
2) setting the operation mode, count clock, and start indication in the mode register (TM0).

The modulo register is set by using an 8-bit data transfer instruction.

Caution: Set a value other than 0 in the modulo register.

Example: To set 3FH in channel 0 modulo register.

```
SEL    MB15        ;   or CLR1 MBE
MOV    XA, #3FH
MOV    TMOD0, XA
```

5.5.5 Timer/event counter time setting

"Timer setup value" (period) is equal to "modulo register contents + 1" divided by "count pulse frequency" selected by setting the timer mode register.

$$T\ (s) = \frac{n+1}{F_{CP}}$$

T (s) : Timer setup value (seconds)
$F_{CP}$ (Hz) : Count pulse frequency (Hz)
n : Modulo register value ($n \neq 0$)

Once the timer is set, an interrupt request signal (IRQT0) is generated at the setup time intervals.

Table 5.5–1 lists the resolution and maximum setup value (time when FFH is set in the modulo register) of the timer/event counter for each count pulse.

Example: To produce 30 ms time intervals. ($f_X$ = 4.194304 MHz)
Use the mode with the maximum setup time 62.5 ms.

$$\frac{30\ \text{ms}}{244\ \mu\text{s}} = 122.9 = 7\text{AH}$$

Set 79H in the modulo register.

```
SEL    MB15
MOV    XA, #79H
MOV    TMOD0, XA
```

**Table 5.5–1 Resolution and Maximum Setup Value (4.19 MHz)**

| Mode register | | | Timer channel 0 | |
|---|---|---|---|---|
| TMO6 | TMO5 | TMO4 | Resolution | Maximum setup time |
| 1 | 0 | 0 | 244 μs | 62.5 ms |
| 1 | 1 | 0 | 15.3 μs | 3.91 ms |

### 5.5.6 Caution on timer/event counter application

(1) Error at timer start

An error with a maximum length of one clock period of the count pulse (CP) for the value calculated in 5.5.5 occurs in the time until a coincidence signal is generated after the timer starts (TM0.3 is set). This is because clearing of the count register is not synchronized with CP as shown in Fig. 5.5–6.

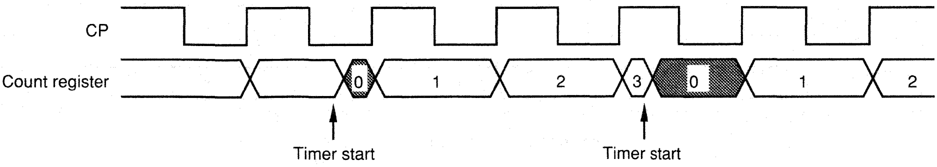


**Figure 5.5–6 Error at Timer Starting**

(2) Caution at timer start

Normally, the count register T0 and interrupt request flag IRQT0 are cleared when the timer starts (TM0 bit 3 is set). However, if the timer is placed in the operation mode, and IRQT0 setting and timer start occur simultaneously, IRQT0 may be unable to clear. There is no problem when IRQT0 is used for a vectored interrupt. In IRQT0 test application, however, a problem arises such than IRQT0 is set although the timer has been started. Thus, to start the timer at IRQT0 timing, stop the timer once (by setting TM0 bit 2 to 0), then restart it, or start the timer twice.

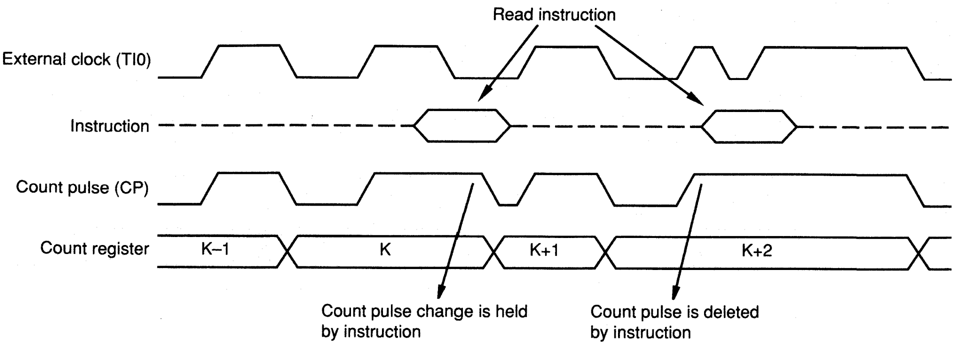
Example: Timer start at IRQT0 timing may be set

```
        SEL     MB15      ;   or CLR1 MBE
        MOV     XA, #0
        MOV     TM0, XA   ;   Timer stop
        MOV     XA, #4CH
        MOV     TM0, XA   ;   Restart
   or
        SEL     MB15      ;   or CLR1 MBE
        SET1    TM0.3
        SET1    TM0.3     ;   Restart
```

(3) Error at count register read

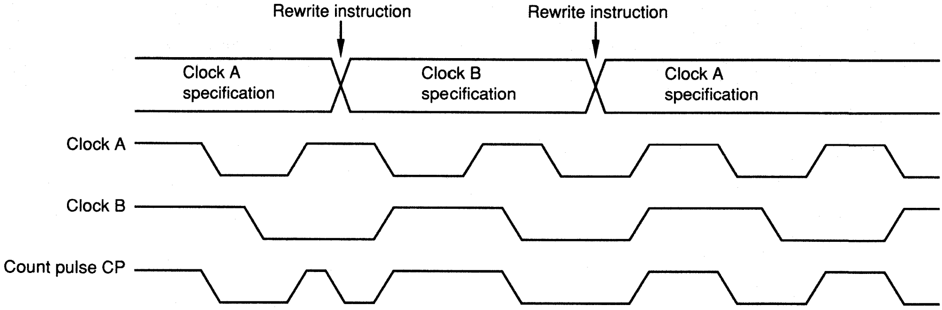
The count register contents can be read at any time using an 8-bit data memory operation instruction. During execution of the instruction, count pulse change is held and count register change is suppressed. Thus, if the count pulse signal source is input to TI0, count pulses as long as the instruction execution time are deleted. (This symptom does not occur if the internal clock is used for count pulses because they are synchronized with the instruction.)

Therefore, if TI0 is input as a count pulse and the count register contents are read, signals having a pulse width that prevents miscount, even if count pulses are deleted, must be input. Since the count hold period in read instruction execution is one machine cycle, the pulse width input to the TI0 pin must be longer than one machine cycle.

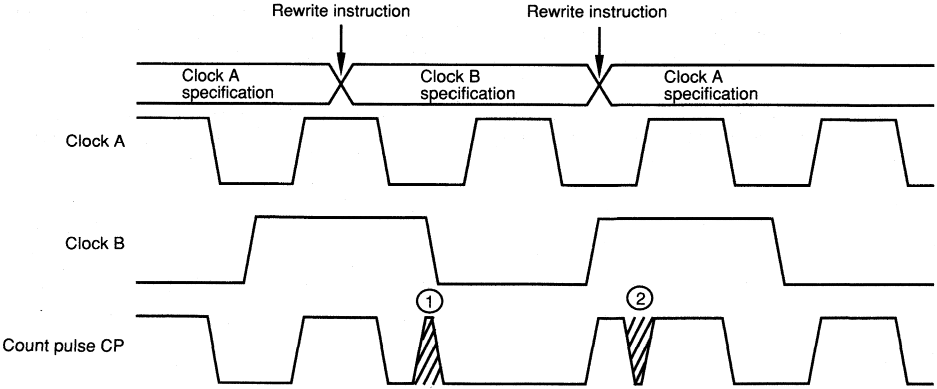


(4) Caution at count pulse change

If the timer mode register is rewritten and count pulse change is made, its specification becomes effective immediately after instruction execution.

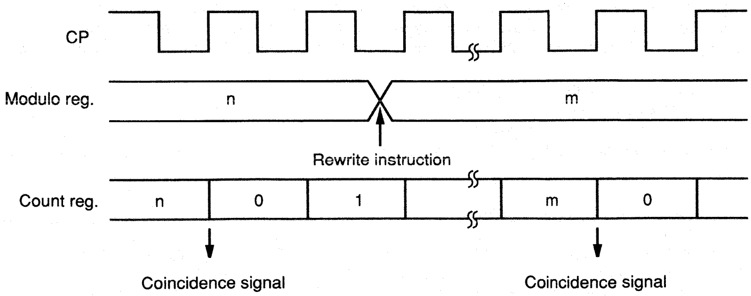


A count pulse ( ① or ② in the timing chart shown below) may occur depending on the clock combination when the count pulse changed. In this case, a miscount may occur or the count register contents may be destroyed. To change count pulse, be sure to set count mode register bit 3 to 1 and restart the timer at the same time.

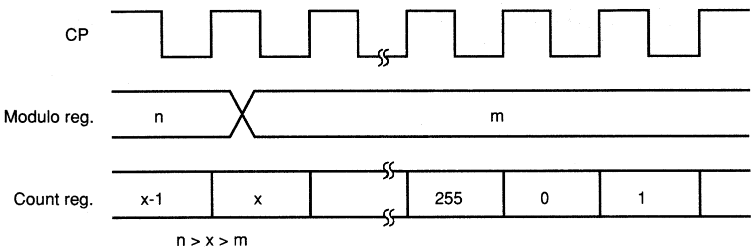


(5) Operation after modulo register change

The modulo register is changed when an 8-bit data memory operation instruction is executed.



If the value appearing after the modulo register is changed and becomes smaller than the count register value, the count register continues counting and overflows, then restarts counting from 0. Thus, if the value after the modulo register is changed (m) and becomes smaller than the value before it was changed (n), the timer must be restarted after the modulo register is changed.



5.5.7 Timer/event counter application

(1) Timer 0 is applied to an interval timer generating an interrupt at 50 ms intervals.

- The high-order four bits of the mode register are set to 0100B and a maximum setup time of 62.5 ms is selected.
- The low-order four bits of the mode register are set to 1100B.
- The modulo register setup value is as follows:

$$\frac{50\text{ ms}}{244\ \mu\text{s}} = 205 = \text{CDH}$$

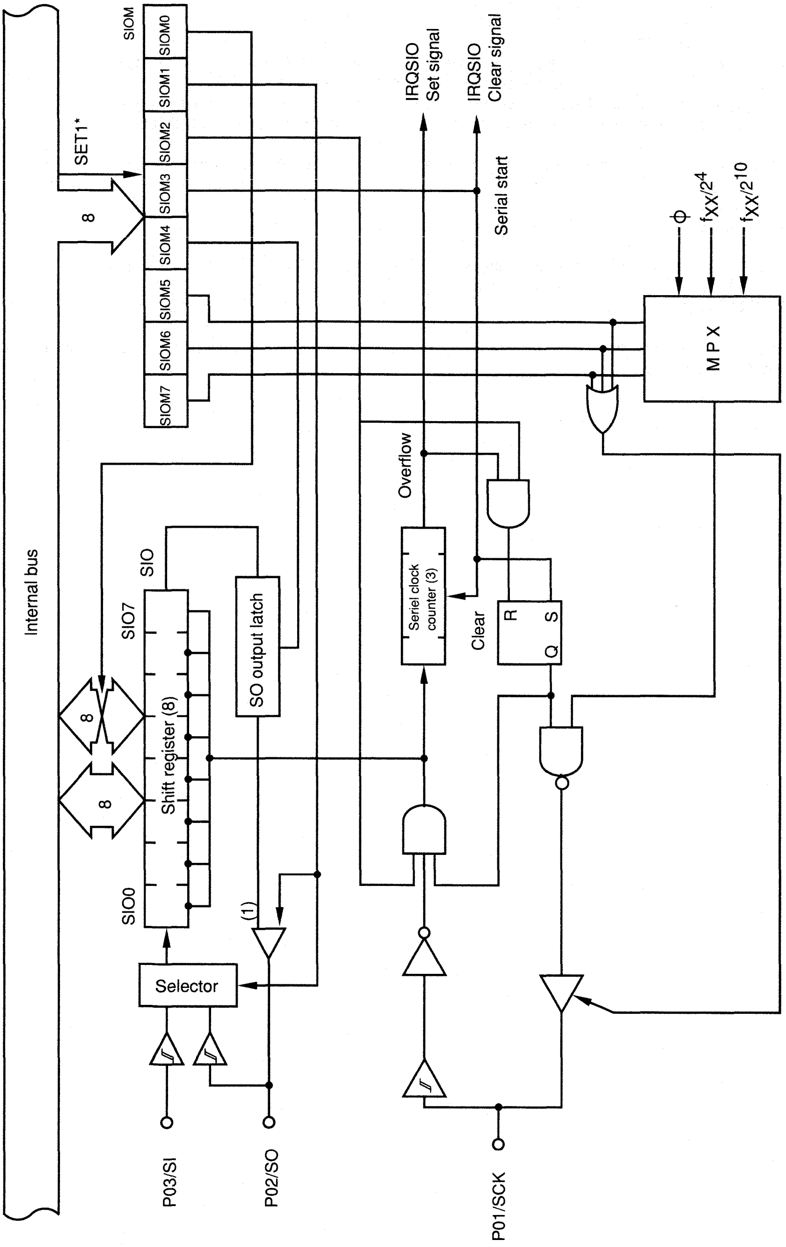
```
Example:  SEL   MB15          ; or CLR1 MBE
          MOV   XA, #0CCH
          MOV   TMOD0, XA     ; Modulo is set.
          MOV   XA, #01001100B
          MOV   TM0, XA       ; Mode is set and timer is started.
          EI                  ; Interrupts is enabled.
          EI    IET0          ; Timer interrupt is enabled.
```

Remarks: In this application, the TI0 pin can be used as an input pin.

## 5.6 Serial Interface

The µPD75268 contains a clocked 8-bit serial interface. The serial interface has the following functions.
(1) 8-bit send and receive operation synchronized with the clock (simulataneous send and receive operations)
(2) 8-bit serial bus operation synchronized with the clock (the S0 pin is used for data I/O, and the format of S0 output is N-ch open drain output.)
(3) Half-duplex start-stop transfer (controlled by sofware)
(4) Selection from four clock sources
(5) LSB/MSB inversion function for starting transfer
This serial interface facilitates data transfer with other microcomputers including the µPD7500 series and connection with peripheral devices.

5.6.1 Configuration and operation of serial interface.
Figure 5.6-1 shows the configuration of the serial interface.
The operation mode and transfer clock of the serial interface are controlled with the serial operation mode register (SIOM). The basic configuration and operation of the interface are described below.
(1) The setting of the SIOM selects a transfer clock used as the shift clock ($\overline{\text{SCK}}$) supplied to the 8-bit shift register (SIO).
(2) The SIO is a shift register that performs a shift operation in phase with an edge of SCK. An 8-bit data memory manipulation instruction is used to write to or read from the SIO. The SIOM can be set so that an SIO read or write operation can be performed starting with the LSB or MSB as desired. This function can select whether data transfer operation is to be started from the LSB or MSB.
(3) The serial clock counter is a 3-bit up-counter, incremented on the rising edge of $\overline{\text{SCK}}$. When this counter overflows, it sets the interrupts request flag IRQSIO, disables the supply of $\overline{\text{SCK}}$ to the SIO, and stops serial transfer operation.
(4) During serial transfer, SIOM.4 can be used to set the SO output state.



*: Instruction execution

1: Output buffer that can be switched between COMS, or N-ch open drain.

**Fig. 5.6-1 Serial Interface Block Diagram**

5.6.2 Serial operation mode register (SIOM)

The serial operation mode register is an 8-bit register for specifying a serial interface operation mode. (See Figure 5.6-2)

An 8-bit memory manipulation instruction is used to set the serial operation mode register.

Bit 3 can be set to 1 independently of the other bits by using the instruction SET1 SIOM.3 (serial interface start instruction) to clear the serial clock counter and interrupt request flag (IRQSIO) and start serial interface transfer operation.

Example 1: Serial clock ϕ is selected, and serial send and receive operations are started from the MSB.

```
SEL    MB15              ; Or CLR1 MBE
MOV    XA,#01001110B
MOV SIOM, XA             ; SIOM ← 01001110B
```

Example 2: Serial transfer operation is started according to the setting of the SIOM.

```
SEL    MB15              ; Or CLR1 MBE
SET1   SIOM.3
```

When the serial interface starts operation, bit 3 is automatically reset to 0. A $\overline{\text{RESET}}$ signal occurrence clears all bits to 0, except for Bit 4. Bit 4 is set to 1 after $\overline{\text{RESET}}$.

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FE0H | SIOM7 | SIOM6 | SIOM5 | SIOM4 | SIOM3 | SIOM2 | SIOM1 | SIOM0 | SIOM |

Shift Register Read/Write Bit Sequency Specification Bit

| | | |
|---|---|---|
| SIOM 0 | 0 | SIO$_7$ to SIO$_0$ ←→ XA (MSB first serial transfer) |
| | 1 | SIO$_0$ to SIO$_7$ ←→ XA (LSB first serial transfer) |

Serial Interface Operation Mode Selection Bit

| | | Operation Mode | SO Pin Output Buffer Format | Serial Data Input Pin |
|---|---|---|---|---|
| SIOM1 | 0 | Serial bus mode | N-ch open drain | P02-SO |
| | 1 | Send/receive mode | CMOS output | P03/SI |

Serial Interface Operation Enable/Disable Bit

| | | Shift Register Operation | Serial Clock Counter | IRQSIO Flag |
|---|---|---|---|---|
| SIOM2 | 0 | Shift stop | Operation stop (hold) | Hold |
| | 1 | Shift operation | Count operation | Settable |

Serial Interface Start Instruction Bit

| | |
|---|---|
| SIOM3 | Serial transfer is started by written "1". Automatically reset (0) after transfer starts. |

Output Data to SO Output Latch

| | | |
|---|---|---|
| SIOM4 | 0 | Output 0 to SO output latch |
| | 1 | Output 1 to SO output latch |

Serial Clock Selection Bits

| SIOM7 | SIOM6 | SIOM5 | Serial Clock |
|---|---|---|---|
| 0 | 0 | 0 | $\overline{\text{SCK}}$ pin input external clock |
| 0 | 0 | 1 | Reserved |
| 0 | 1 | X | $\phi$* (1.05 MHz, 524 kHz, 65.5 kHz) |
| 1 | 0 | 0 | $f_{XX}/2^{10}$ (4.09 kHz) |
| 1 | 0 | 1 | Reserved |
| 1 | 1 | 0 | |
| 1 | 1 | 1 | $f_{XX}/2^4$ (262 kHz) |

Values in ( ) are for $f_{XX}$ = 4.19 MHz.
X: Don't care
*: $\phi$ is CPU clock selected by PCC.

**Fig. 5.6-2 Serial Operation Mode Register Format**

5.6.3 Serial interface operation

Serial interface has the following operation modes:

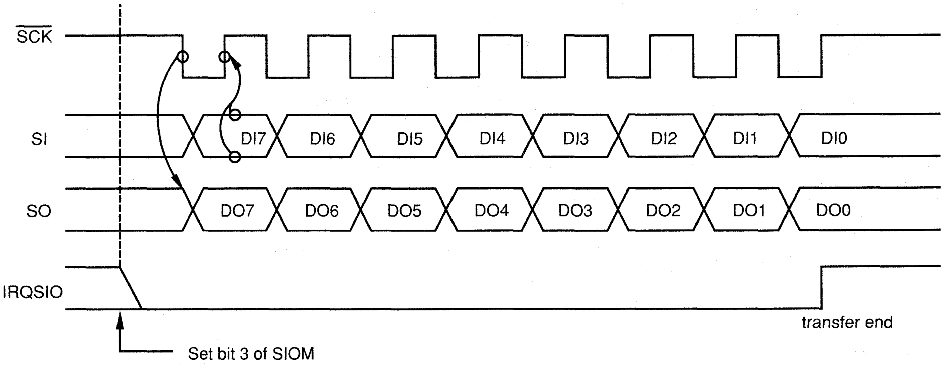
- Operation stop mode
- Serial operation mode

(i) Operation stop mode

When SIOM bit 2 is "0", the serial interface is set to the operation stop mode. In this mode, serial transfer is not performed. The serial interface is set to this mode when reducing the power consumption by not using the serial interface.
Since the shift operation is not performed in this mode, the shift register can be used as a normal 8-bit register.
In this mode, the status of the SO output latch can be set by SIOM.4 specification. Thus the SO pin can be used as normal static output.
The $\overline{\text{RES}}$ signal sets the serial interface to this mode.

(ii) Serial operation mode

When SIOM bit 2 is "1", the serial interface is set to the serial operation mode.
There are two serial operation modes, depending on the contents of SIOM bit 1:
① Serial send/receive mode (SIOM1 = 1)
② Serial bus mode (SIOM1 = 0)
Their operation timing is shown in Figure 5.6-3 and Figure 5.6-4, respectively.



Note: DI Serial Data Input
DO Serial Data Output

**Fig. 5.6-3 Serial Send/Receive Mode Timing**

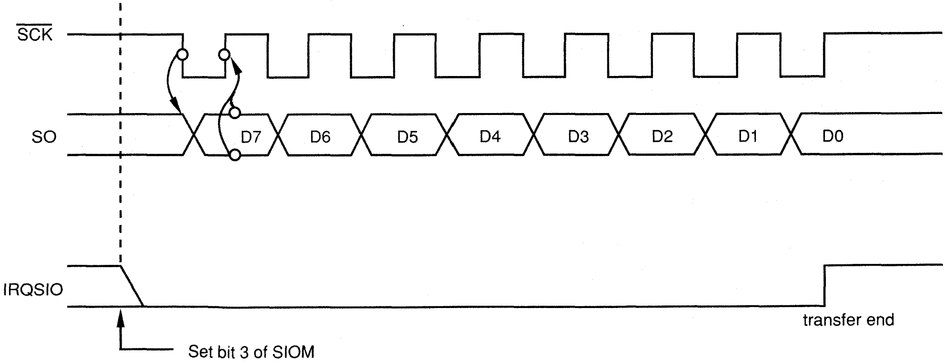


**Fig. 5.6-4 Serial Bus Mode Timing**

In the serial send/receive mode, the receive data is placed in the shift register from the SI pin in synchronization with the serial clock (SCK) and at the same time, the send data is output from the SO terminal. At this time, the SO output becomes CMOS output.
In the serial bus mode, the contents of the shift register are output to the SO pin in synchronization with SCK and at the same time, the data of the same SO pin is fetched to the shift register. At this time, the SO output becomes N-ch open drain output.
In serial bus mode use, transfer is divided into transfer when serial data is output and transfer when serial data is input. When serial data is output, the send data is written to the shift register the same as in the serial send/receive mode. When serial data is input the serial bus is opened by turning off an N-ch transistor by written FFH to the shift register beforehand.
In the serial bus mode, the P03/SI pin can be used as general-purpose input.
Four kinds of clocks can be selected as the serial clock by mode register setting as shown in Table 5.6-1. Shift operation is normally started by:
① Loading the transfer data into the shift register (SIO)
② Setting the operation mode, serial clock selection, and start order in the serial interface operation mode register (SIOM).
Serial transfer stops automatically and the interrupt request flag (IRQSIO) is set at the end of transfer of 8 bits. While serial transfer is stopped, the status of the SO output latch can be set by SIOM.4 specification.

**Table 5.6-1 Serial Clock Selection and Applications**

| Mode register | | | Serial clock | | Timing wich enables R/W of shift register and start of data transfer | Application |
|---|---|---|---|---|---|---|
| SIOM 7 | SIOM 6 | SIOM 5 | Source | Masking of serial clock | | |
| 0 | 0 | 0 | External SCK | Automatically masked at end of 8-bit data transfer. | Enable only while serial transfer is stopped, * or while SCK is at high level. | Slave CPU |
| 0 | 1 | X | φ | | Enabled at any time | High-speed serial transfer |
| 1 | 0 | 0 | $f_{XX}/2^{10}$ | | Enabled only while serial transfer is stopped,* or while SCK is at high level. | Low-speed serial transfer |
| 1 | 1 | 1 | $f_{XX}/2^{4}$ | | | Middle-speed serial transfer |

* "While serial transfer is stopped" means a condition in which the serial clock is masked after an 8-bit data transfer of during operation stop mode.
Note: In case that CPU is set in the lowest speed mode (PCC=0000), medium speed mode for serial clock (SIOM=111xxxxx) may not be used.

The shift register is read and written by 8-bit transfer instruction. At this time, the shift register can be read and written by reversing the LSB and MSB by SIOM bit 0 specification. Transfer can be switched between LSB first and MSB first with this function.

Example: Transfer the data of the RAM specified by the HL register pair to SIO and at the same time, fetch the SIO data to the accumulator.

```
MOV     XA, @HL      ; take send data from RAM.
SEL     MB15         ; or CLR1 MBE
XCH     XA, SIO      ; swap send data and receive data
```

The serial transfer timing when φ was selected as the serial clock is shown in Figure 5.6-5. (The transfer data can be read and written by waiting 7 machine cycles.)

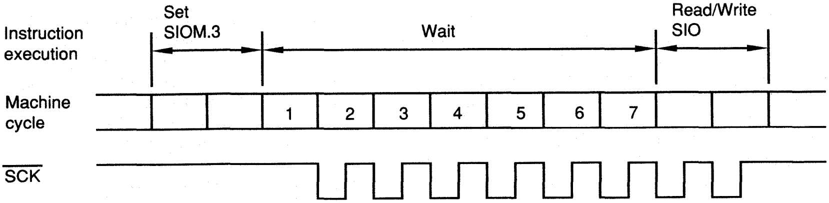


**Fig. 5.6-5 Transfer Timing when φ is selected as Serial Clock**

5.6.4 Serial interface applications

The serial interface function are usually programmed in applications as follows:

— Sets transfer data into the shift register (SIO).

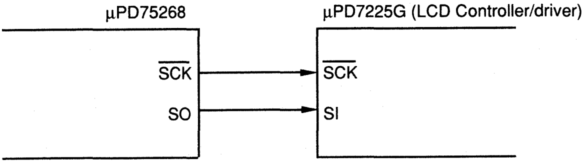
— Sets the operation mode, serial clock type, and start specification on the mode register (SIOM).

— After testing interrupt routine or interrupt request flag (IRQSIO), reads received data.

(1) Transfers data in MSB-first at a transfer clock rate of 262 kHz (at 4.19 MHz) (master option):

```
Example:  CLR1   MBE
          MOV    XA, TDATA      ; TDATA is the address where transfer data is stored.
          MOV    SIO, XA        ; Sets transfer data
          MOV    XA, #11101110B
          MOV    SIOM, XA       ; Starts transfer
```

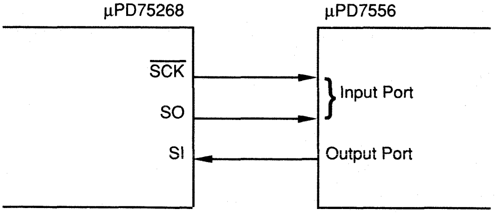
Note: The second and all the subsequent data transfer can be started with SET1 SIOM.3.



In this application the SI/P03 pin of the μPD75268 can be used as an input.

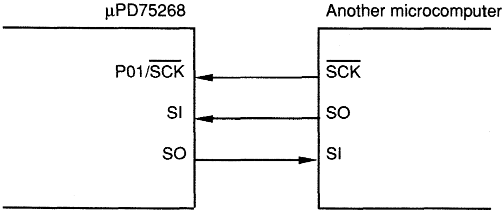
(2) Sends and receives data in MSB-first at a clock rate of 4.09 kHz (at 4.19 MHz) (master option):

This application uses a low transfer clock rate and is suitable for interfacing with a microprocessor having no serial interface such as the μPD7556.



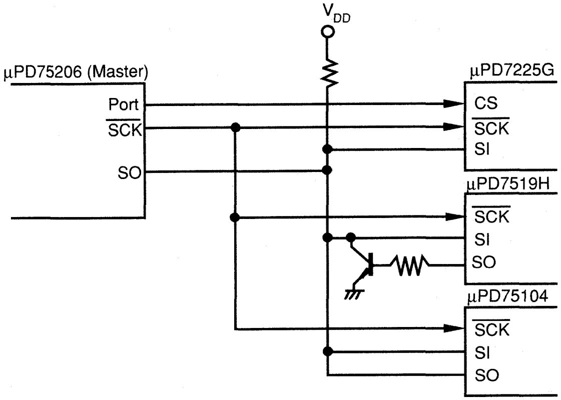
```
Exampe:   Main routine
          CLR1    MBE
          MOV     XA, TDATA
          MOV     SIO, XA           ; Sets transfer data
          MOV     XA, #10001110B
          MOV     SIOM, XA          ; Starts transfer
          EI
          EI      IESIO             ; Enable SIO interrupt

          2nd and subsequent transfer start
          CLR1    MBE
          MOV     XA, TDATA
          MOV     SIO, XA
          SET1    SIOM.3

          Interrupt routine (MBE=0)
          MOV     XA, SIO
          MOV     RDATA, XA         ; Save received data. RDATA is address where received data is stored.
          RETI
```

(3) Send and receives data in LSB-first using an external clock (slave operation).
(In this application, the shift register is accessed by using the LSB-first/MSB-first selectable function.)



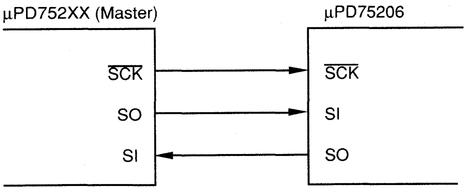
```
Example:  Main routine
          CLR1    MBE
          MOV     XA, #01H
          MOV     SIOM, XA          ; Stops serial transfer and enters LSB-first mode.
          MOV     XA, TDATA
          MOV     SIO, XA           ; Sets transfer data
          MOV     XA, #0FH          ; External clock
          MOV     SIOM, XA          ; Starts transfer
          EI
          EI      IESIO             ; Enables SIO interrupt

          Interrupt routine (MBE=0)
          MOV     XA, TDATA
          XCH     XA, SIO           ; Receive data↔send data
          MOV     RDATA, XA         ; Saves received data
          SET1    SIOM.3
          RETI
```

(4) Configures a serial bus and connects more than one device to it
Example: Configures a system in which the μPD75268 function as a master device and μPD75104, μPD7519H, and μPD7225G function as slave devices connected via a serial bus:



In this bus configuration, SI and SO pins are connected, and the SO pins should be set to high impedance to release the bus except when the SO is outputting serial data. Since the μPD7519H cannot put the SO pin to high-impedance state, the μPD7519H should be provided with an open-collector transistor as shown in the figure above. Data 00H should be written into the shift register so that the transistor can be turned off at data input timing. The timing when each microprocessor outputs data should be determined. The master processor μPD75268 delivers a serial clock and all other slave processors operate on that clock (external clock).

(5) Use the CPU clock $\phi$ (0.95 μs at 4.19 MHz) for the transfer data at high speed:



Programming example —— Master device

```
             CLR1   MBE
             MOV    XA, TDATA
             MOV    SIO, XA          ; Sets transfer data.
             MOV    XA, #01001110B
             MOV    SIOM, XA         ; Starts transfer.
           ⎧  •
7 machine  ⎨  •
cycle      ⎪  •
           ⎩  •
             MOV    XA, SIO          ; Fetches received data
```

### 5.7 FIP Controller/Driver

5.7.1 FIP controller/driver configuration and operation
The μPD75268 contains a display controller which reads the display data memory contents by DMA operation and automatically generates digit and segment signals. The high-voltage output buffer for the FIP controller/driver can directly drive a fluorescent indicator panel (FIP). Figure 5.7-1 shows the FIP controller/driver configuration.

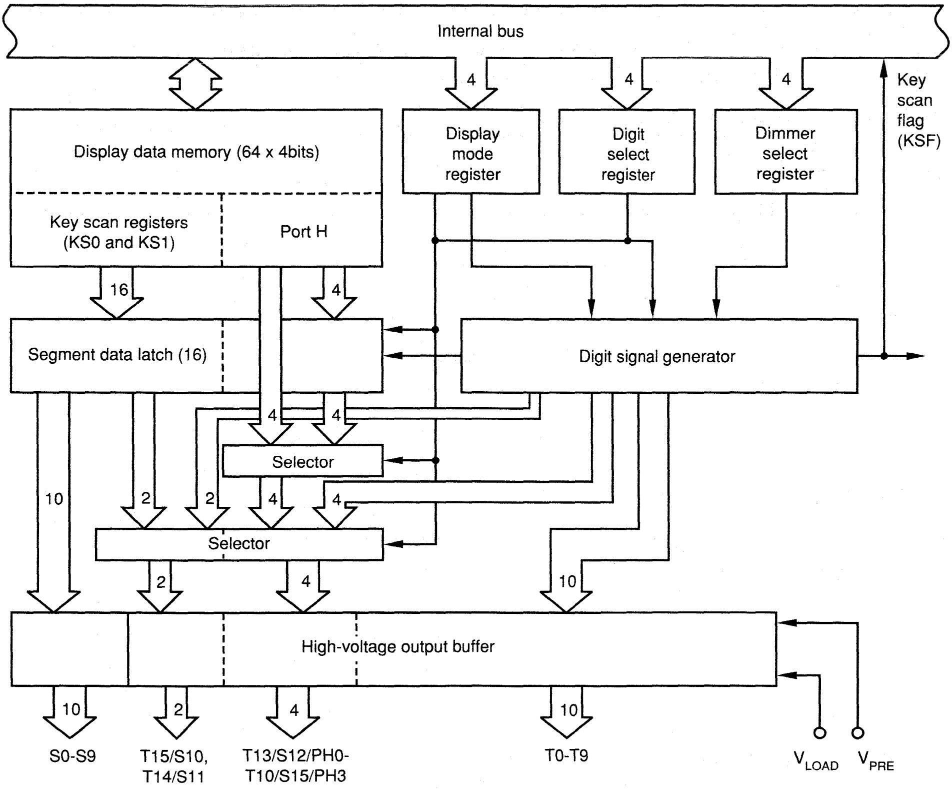


**Fig. 5.7-1 FIP Controller/Driver Block Diagram**

The Function of the μPD75268 internal FIP controller/Driver are as follows:
(i) Display data memory is read automatically by DMA operation, and segment and digit signals are generated.
(ii) A maximum of 26 display elements can be selected as desired in the range of nine to 16 segments and nine to 16 digits.
(iii) The remaining display output can be used for static output.
(iv) The intensity can be adjusted in eight stages by using the dimmer function.
(v) Key scan application is enabled.
- Interrupt generation at key scan timing (IRQKS)
- Key scan data can be output from the segment output pins.

(vi) High voltage (40 V) output pins for directly driving the FIP

- Segment-only pins (SO-S9): $V_{OD}$ = 40 V, $I_{OD}$ = 3 mA
- Digit output pins (TO-T15): $V_{OD}$ = 40 V, $I_{OD}$ = 15 mA

(vii) An internal pull-down resistor can be including for each bit by mask option.

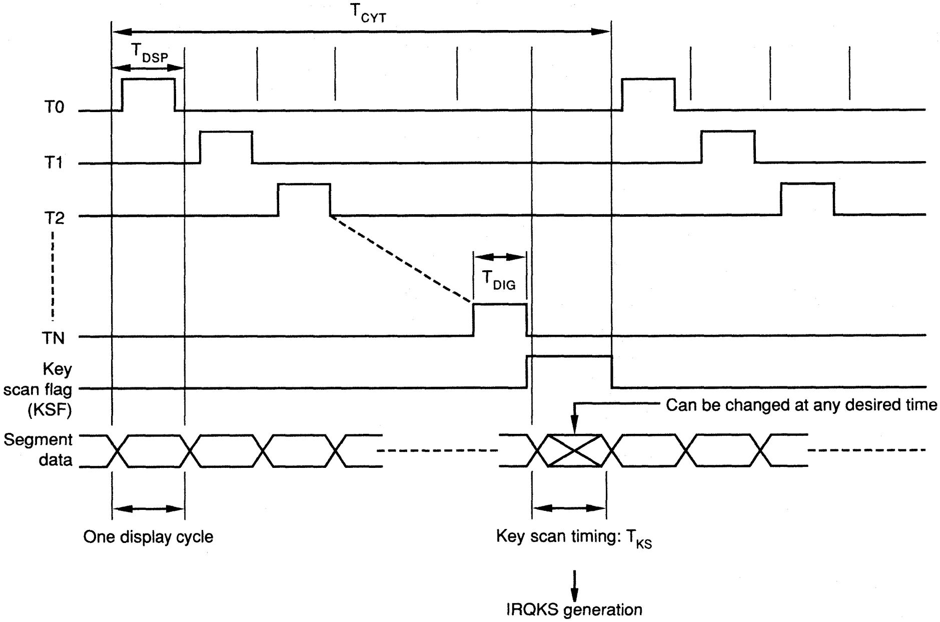
The FIP controller reads data stored in display data memory in sequency and outputs it to the segment pin ($S_N$).

Fig. 5.7-2 shows the timing for the FIP controller.

The display mode is determined by a combination of the digit signal assigned the number of digits specified in the digit select register (DIGS) and the segment signal assigned the number of segments specified in the display mode register (DSPM).

For display leakage glare, prevention or intensity adjustment (dimmer), a digit signal whose cut width is specified in the dimmer select register (DIMS) is output.

The key scan registers (KS0 and KS1) are mapped in the display data memory. Data set in KS0 and KS1 is output from the segment output pins at the key scan timing.



N: Digit select register setup value

$T_{DSP}$: One display cycle $\frac{1024}{f_{XX}}$ : 244 μs at 4.19 MHz, $\frac{2048}{f_{XX}}$ : 488 μs at 4.19 MHz)

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

$T_{DIG}$: Digit signal pulse width which can be changed in eight stages by using the dimmer select register.

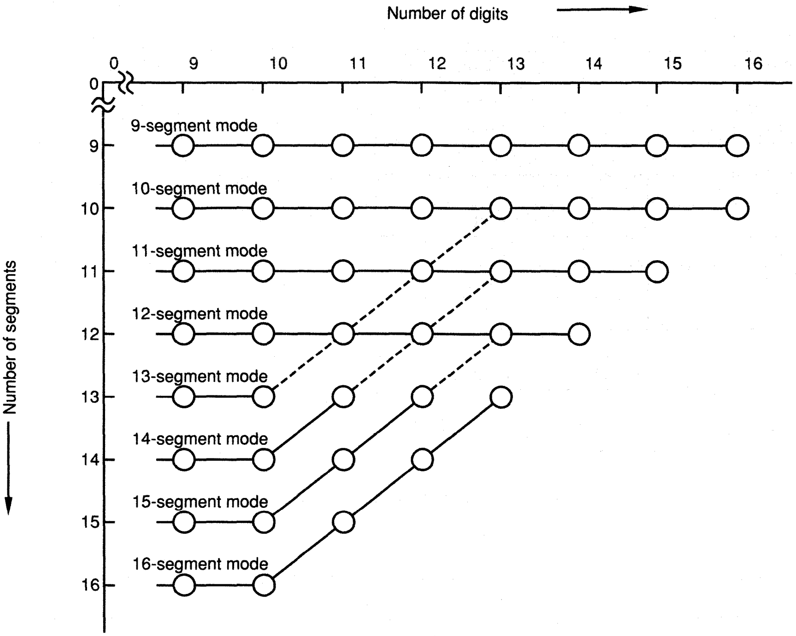
**Fig. 5.7-2 FIP Controller Operation Timing**

Caution on use:

1. The FIP controller/driver can operate only with high speed or medium speed (PCC = 0011B or 0010B) main system clock (SCC.0 = 0). In another clock mode or in standby mode, be sure to stop FIP controller operation (DSPM.3 = 0), or else the FIP controller may malfunction.
2. PORTH, when used for the digit/segment signals (T10/S15-T13/S12), is either placed in high impedance (when no pull-down resistor is contained) or goes low (when internal pull-down register is contained) in display off mode, etc.
   Thus, PORTH should be turned off when display is turned off in standby mode, etc.

5.7.2 Display mode selection

The number of segments and digits that can be displayed by the internal FIP controller/driver is determined by selecting the display mode.



Remark 1: Pins not used for display can be used for an output port.

**Fig. 5.7-3 Display Mode Selection**

5.7.3 FIP controller/driver registers

(1) Display mode register (DSPM) is a 4-bit register used to specify whether display operation is enabled or disabled and the number of display segments. Fig. 5.7-4 shows the DSPM register format.

The display mode register is set by using a 4-bit memory handling instruction.

When the Standby mode (STOP or HALT) is set or the FIP controller/driver is operated on the subsystem clock ($f_{XT}$), preset DSPM.3 to 0 to stop display operation.

When the $\overline{\text{RES}}$ signal is generated, all the DSPM bits are cleared.

Example: To use 11-segment display

```
SEL     MB15
MOV     A, #1010B
MOV     DSPM, A
```

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

Specification bit for the number of display segments

| DSPM2* | DSPM1 | DSPM0 | Number of display segments |
|---|---|---|---|
| 0 | 0 | 0 | 9 segments |
| 0 | 0 | 1 | 10 segments |
| 0 | 1 | 0 | 11segments |
| 0 | 1 | 1 | 12 segments |
| 1 | 0 | 0 | 13 segments |
| 1 | 0 | 1 | 14 segments |
| 1 | 1 | 0 | 15 segments |
| 1 | 1 | 1 | 16 segments |

Specification as to whether display operation is enabled or displayed

| | | |
|---|---|---|
| DSPM3 | 0 | Display stop |
| | 1 | Display enabled |

**Fig. 5.7-4 Display Mode Register Format**

(2) Digit select register (DIGS)

The digit select register (DIGS) is a 4-bit register used to specify the number of display digits. Fig. 5.7-5 shows the DIGS register format.

DIGS is set by using a 4-bit memory handling instruction. The number of display digits can be selected within the range of nine to 16 by setting DIGS. A value less than nine cannot be selected.

When the $\overline{\text{RES}}$ signal is generated, DIGS is set to 1000B, selecting a 9-digit display.

Example: To select a 10-digit display.

```
SEL   MB15
MOV   A, #9
MOV   DIGS, A
```

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

| DIGS-DIGS3 setup value | Number of display digits |
|---|---|
| N | N+1 |

N = 8 – 15

**Fig. 5.7-5 Digit Select Register Format**

(3) Dimmer select register (DIMS)

The dimmer select register (DIMS) is a 4-bit register used to specify the digit signal cut width for display leakage glare prevention or intensity adjustment (dimmer function). It is also used to select the display cycle ($T_{DSP}$).

Fig. 5.7-6 shows the DIMS format. Fig. 5.7-7 shows the waveform of the digit signal for each setup value.

DIMS is set by using a 4-bit memory handling instruction.

Normally, 488 μs at 4.19 MHz is selected for the display cycle by setting DIMS.0 to 1 lessen leakage glare. However, as the number of display digits increases, the display cycle approaches the commercial supply power frequency and the display flickers; 244 μs at 4.19 MHz is selected.

When $\overline{RES}$ signal is generated, all the DIMS bits are cleared.

Caution: To avoid display flickering be shure to change the DIMS register in display off phase or in key scan interrupt routine.

Example: To select a cut width of 1/16 and a display cycle of 488 μs.

```
SEL   MB15
MOV   A, #1
MOV   DIMS, A
```

Address

| | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|
| F89H | DIMS3 | DIMS2 | DIMS1 | DIMS0 | DIMS |

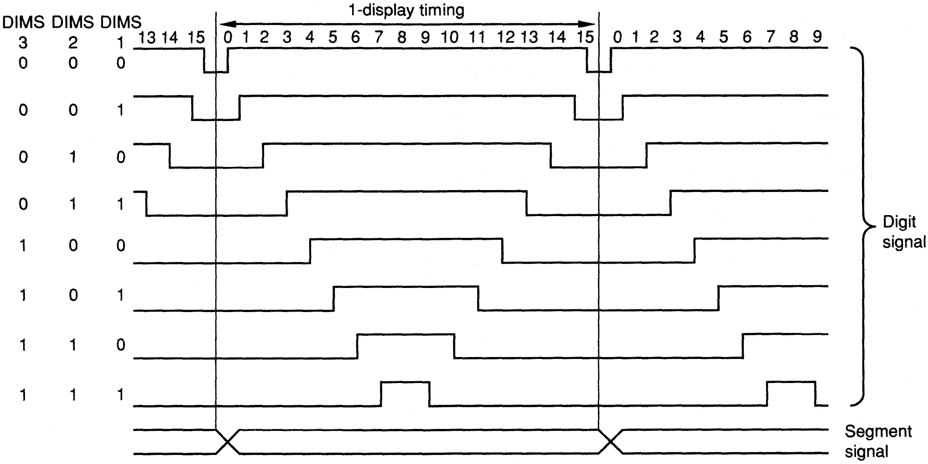
Display cyle specification bit

| | | |
|---|---|---|
| DIMS0 | 0 | $\frac{1024}{f_{XX}}$ = one display cycle. (one cycle = 244 μs at 4.19 MHz) |
| | 1 | $\frac{2048}{f_{XX}}$ = one display cycle. (one cycle = 488 μs at 4.19 MHz) |

Specification bit for the number of display segments

| DIMS3 | DIMS2 | DIMS1 | Digit signal cut width |
|---|---|---|---|
| 0 | 0 | 0 | 1/16 |
| 0 | 0 | 1 | 2/16 |
| 0 | 1 | 0 | 4/16 |
| 0 | 1 | 1 | 6/16 |
| 1 | 0 | 0 | 8/16 |
| 1 | 0 | 1 | 10/16 |
| 1 | 1 | 0 | 12/16 |
| 1 | 1 | 1 | 14/16 |

**Fig. 5.7-6 Dimmer Select Register Format**



Note: 1-display timing is 2048/$f_{XX}$ (with DIMS0 = 1) = 488 μs/4.19 MHz or 1024/$f_{XX}$ (DIMS0 = 0) = 244 μs/4.19 MHz.

**Fig. 5.7-7 Digit Signal Waveform**

5.7.4 Display data memory

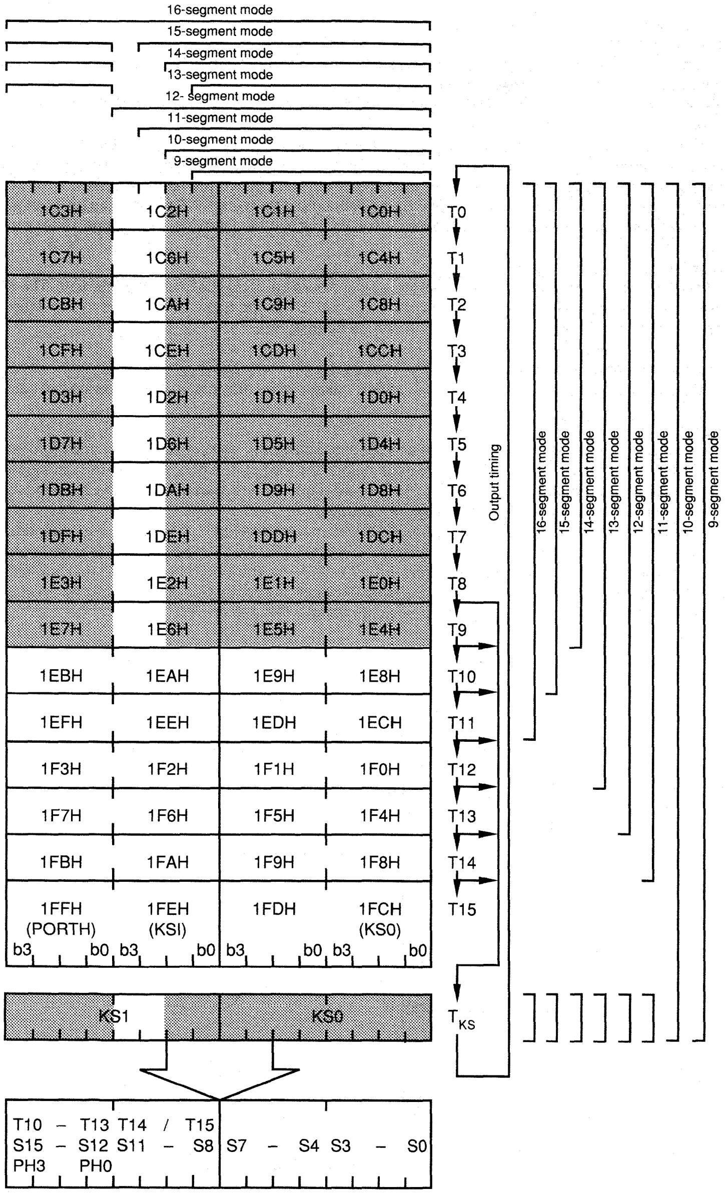
The display data memory is the area of data memory used by the display controller to write segment output data read by DMA operation. The area not used for display can be used for normal data memory.

The display data memory is mapped in data memory addresses 1COH-1FFH. Display data memory addresses 1FCH, 1FDH, 1FEH and 1FFH are also used for the key scan registers (KSO and KS1).

The display data memory can be handled in 1-, 4-, or 8-bit units by using data memory handling instructions. Fig. 5.7-9 shows an example how display data memory is related to the segment output. The shaded area in the figure is an example of how the data memory is used when the 14-segment mode is selected (14-segment 10-digit display). In this example, the S0-S9, T13/S12/PH0-T10/S15/PH3 and T0-T9 pins are used for display, and the T14/S11 and T15/S10 pins can be used for static output. In this case, KS1 bits 2 and 3 (high-order two bits of address 1FEH) are output to the T14/S11 and T15/S10 pins. (However, note that T14/S11 and T15/S10 pin output changes after a maximum delay of one display cycle after KS1 data is rewritten.)

| | | 1C3H + 4n | 1C2H + 4n | 1C1H + 4n | 1COH + 4n |
|---|---|---|---|---|---|
| Number of bits that can be manipulated | 1 bit | O | O | O | O |
| | 4 bit | O | O | O | O |
| | 8 bit | O | | O | |

**Fig. 5.7-8 Number Display Data Memory Bits that can be Manipulation (n = 0 – 15)**



| | | | | |
|---|---|---|---|---|
| 1C3H | 1C2H | 1C1H | 1C0H | T0 |
| 1C7H | 1C6H | 1C5H | 1C4H | T1 |
| 1CBH | 1CAH | 1C9H | 1C8H | T2 |
| 1CFH | 1CEH | 1CDH | 1CCH | T3 |
| 1D3H | 1D2H | 1D1H | 1D0H | T4 |
| 1D7H | 1D6H | 1D5H | 1D4H | T5 |
| 1DBH | 1DAH | 1D9H | 1D8H | T6 |
| 1DFH | 1DEH | 1DDH | 1DCH | T7 |
| 1E3H | 1E2H | 1E1H | 1E0H | T8 |
| 1E7H | 1E6H | 1E5H | 1E4H | T9 |
| 1EBH | 1EAH | 1E9H | 1E8H | T10 |
| 1EFH | 1EEH | 1EDH | 1ECH | T11 |
| 1F3H | 1F2H | 1F1H | 1F0H | T12 |
| 1F7H | 1F6H | 1F5H | 1F4H | T13 |
| 1FBH | 1FAH | 1F9H | 1F8H | T14 |
| 1FFH<br>(PORTH) | 1FEH<br>(KSI) | 1FDH | 1FCH<br>(KS0) | T15 |
| KS1 | | KS0 | | T<sub>KS</sub> |

The shaded portion denotes the area used when 14-segment mode is selected. (14-segment 10-digit display))

**Fig. 5.7-9 Example of Relationship between Display Data Memory and Segment Output (Example µPD75268)**

5.7.5 Key scan registers and flag

(1) Key scan registers (KS0 and KS1)

The Key scan registers (KS0 and KS1) are mapped in part of the display data memory (addresses 1FCH-1FFH) and are used to set segment output data at the key scan interval.

KS0 and KS1 are 8-bit registers which normally are operated by 8-bit manipulation instructions. (They can also be manipulated in 1- or 4-bit units.)

Data in KS0 and KS1 is output from the segment output pins at the key scan interval. If KS0 and KS1 are rewritten during the key scan interval, segment output data can be changed immediately. The segment output can be used for key scanning.

KS1 also serves as an output data register when any remaining display output pins are used for static output. In this case, KS1 high-order 3-bit data corresponds to the S9-S11 pins. Static data output from the S9-S11 pins can be changed immediately when KS1 is rewritten during the key scan interval. Note, however, that at other intervals, it is changed after a maximum delay of one display cycle.

(2) Key scan flag (KSF)

The key scan flag (KSF) is set to 1 during the key scan timing and cleared automatically at other intervals. The KSF is mapped in address F8AH bit 3, can be tested bitwise, and is write-protected. Since the flag can be tested to see if the current timing is for key scan, whether or not key input data is correct can be determined.

5.7.6 Fluorescent indicator panel leakage glare

When the μPD75268 is used to drive a fluorescent indicator panel, leakage glare may occur. The following two causes of leakage glare are possible:

(1) Short blanking time

Fig. 5.7-10 shows the signal waveform when the first and second digits on 2-digit display are on and off for simplicity. If the blanking time is short as shown here, the T1 signal rises before the segment signal disappears, causing leakage glare. Generally, the blanking time required is about 20 μs. This should be considered when setting the blanking mode register.

(2) Capacitance between fluorescent indicator panel segment and grid.

Leakage glare may accur although enough blanking time is allowed as shown in Fig. 5.7-12. Since capacitance exists between the grid and segment of the fluorescent indicator panel ($C_{SG}$ in Fig. 5.7-11), the timing signal pin rises through $C_{SG}$ when the segment signal is turned on. If this voltage exceeds the cut-off voltage as showsn in Fig. 5.7-12, leakage glare occurs. This hair-like voltage varies according to the value of $C_{SG}$ and the value of internal pull-down resistor RL. As $C_{SG}$ goes higher and RL goes higher, the voltage becomes higher and leakage glare is likely to occur.

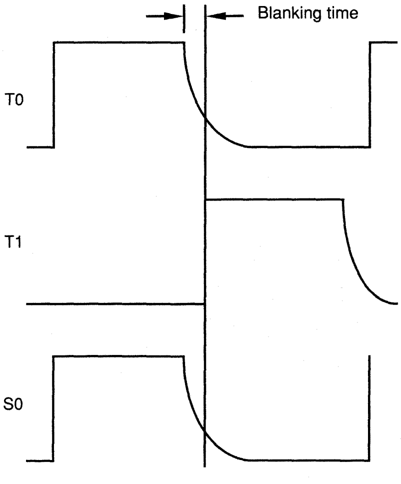


**Fig. 5.7-10 Concept of Leakage Glare Caused by Short Blanking Time**

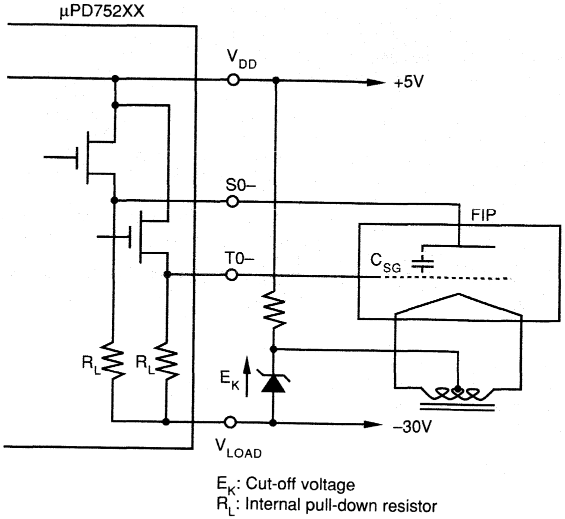


**Fig. 5.7-11 Leakage Glare Caused by $C_{SG}$**

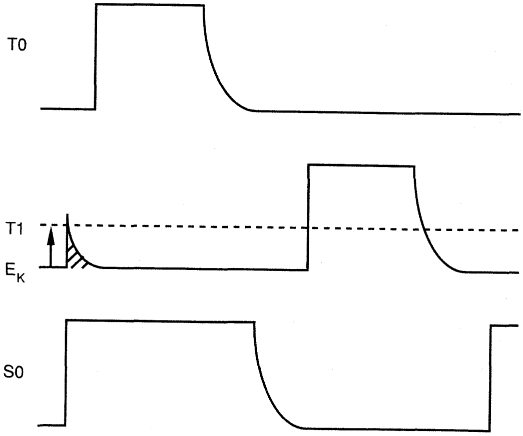
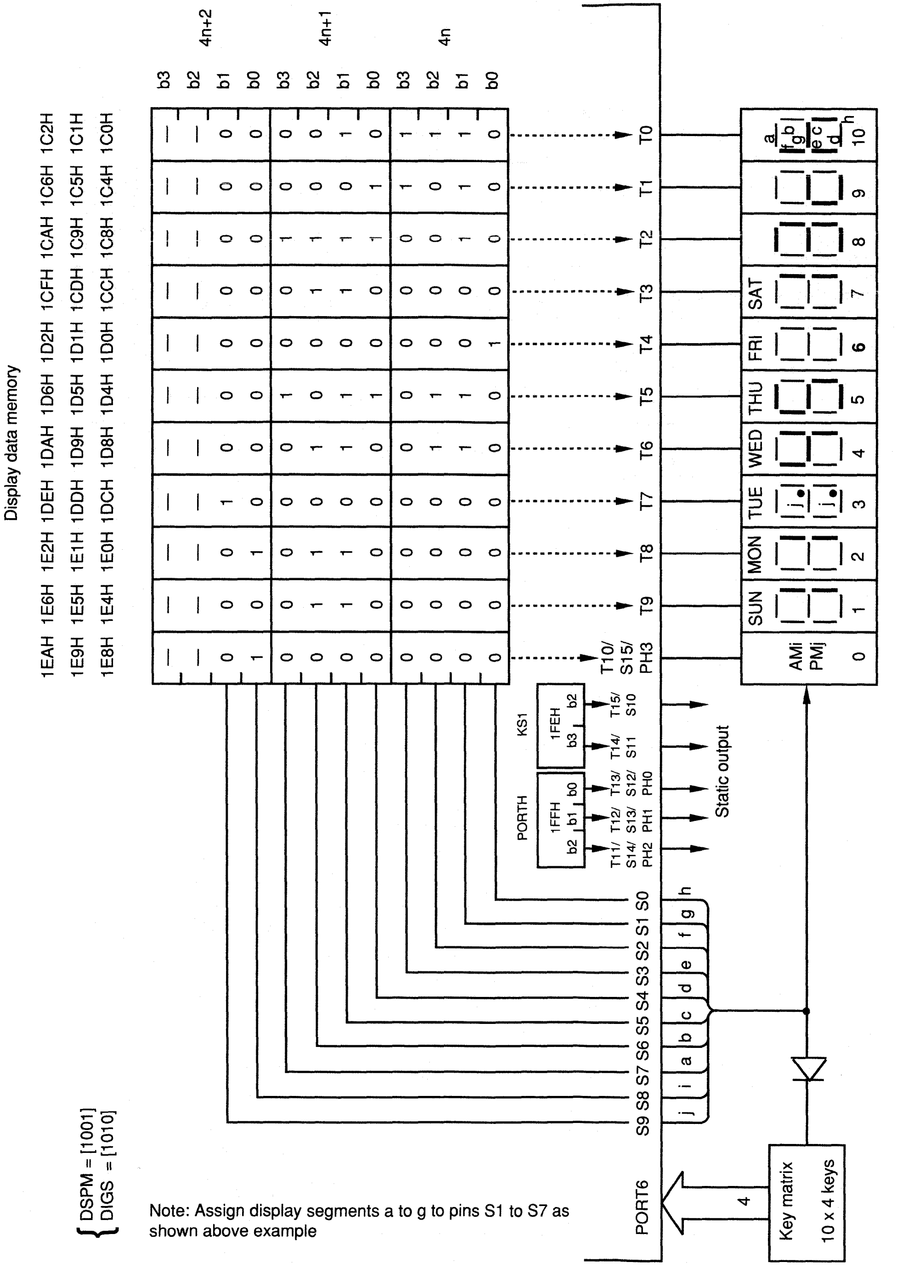


**Fig. 5.7-12 Concept of Leakage Glare Caused by $C_{SG}$**

The $C_{SG}$ value varies with the display area of the fluorescent indicator panel. As larger they are, $C_{SG}$ becomes higher. Thus, the pull-down resistor value for preventing leakage glare from occuring, varies according to the size of the fluorescent indicator panel.
On the other hand, the value of the internal pull-down resistor included in the μPD75268 by mask option is comparatively high (40-120 kΩ) due to the limitation of power loss allowable for a flat package. Therefore, the internal pull-down resistor may be unable to suppress leakage glare.
To use the μPD75268 to drive a fluorescent indicator panel, precheck to see if μPD75268 internal pull-down resistor can supress leakage glare on the fluorescent indicator panel by using the piggyback product or OTP version.
If statisfactory display quality is not obtained, take proper countermeasures such as deepening back bias (rising $E_K$), attaching a filter to the fluorescent indicator panel, or connecting an external pull-down resistor of about several 10 kΩ to the timing signal pin.
The tendency toward leakage glare caused by $C_{SG}$ varies according to the duty cycle of hair -like voltage for the entire display period. The fewer the number of display digits, the more often leakage glare occurs. Likewise, a slower display controller clock is better. Dimmer select register bit 0 (DIMS0) should be set to 1.
If the display intensity can be lowered slightly and if timing pins remain, the leakage glare can be reduced by setting many display digits (for example, setting 9-segment 16-digit display when the actual display is 9-segment 8 digits).

5.8.7 Display examples
Display examples for each segment mode are given below.
(1) 10-segment mode display example: 10-segment 11-digit display (Fig. 5.7-13)
(2) 14-segment mode display example: 14-segment 10-digit display (Fig. 5.7-14)
(3) 16-segment mode display example: 16-segment 8-digit display (Fig. 5.7-15)


Display data memory
1EAH 1E6H 1E2H 1DEH 1DAH 1D6H 1D2H 1CFH 1CAH 1C6H 1C2H
1E9H 1E5H 1E1H 1DDH 1D9H 1D5H 1D1H 1CDH 1C9H 1C5H 1C1H
1E8H 1E4H 1E0H 1DCH 1D8H 1D4H 1D0H 1CCH 1C8H 1C4H 1C0H
4n+2
4n+1
4n
b3 b2 b1 b0 b3 b2 b1 b0 b3 b2 b1 b0
T0
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10/ S15/ PH3
KS1
1FEH
b3 b2
T15/ S10
T14/ S11
PORTH
1FFH
b2 b1 b0
T13/ S12/ PH0
T12/ S13/ PH1
T11/ S14/ PH2
Static output
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9
h g f e d c b a i j
a b c d e f g h
10 9 8 7 6 5 4 3 2 1 0
SAT FRI THU WED TUE MON SUN
AMi PMj
PORT6
4
Key matrix
10 x 4 keys

DSPM = [1001]
DIGS = [1010]

Note: Assign display segments a to g to pins S1 to S7 as shown above example

Fig. 5.7.13 10-Segment 11-Digit Display

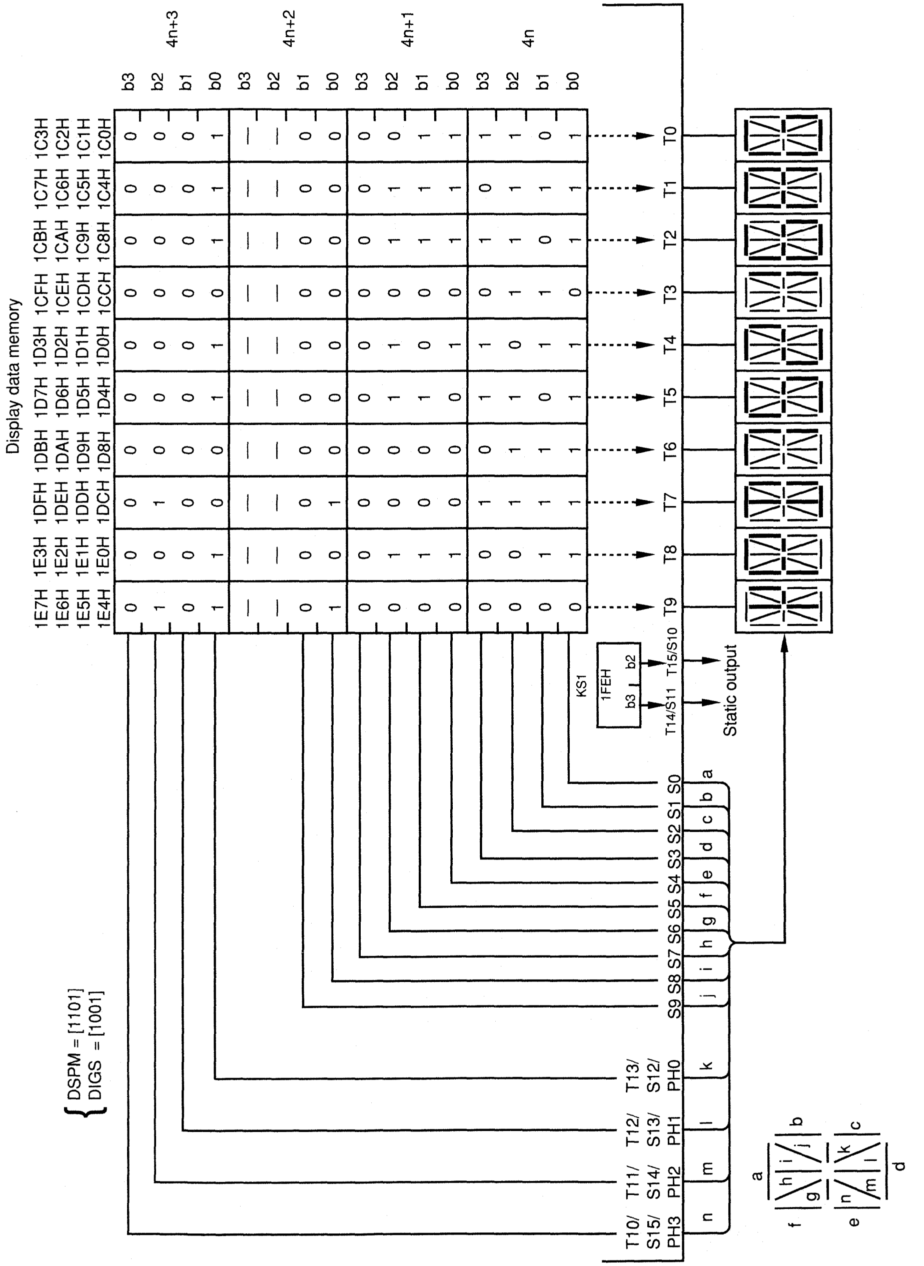
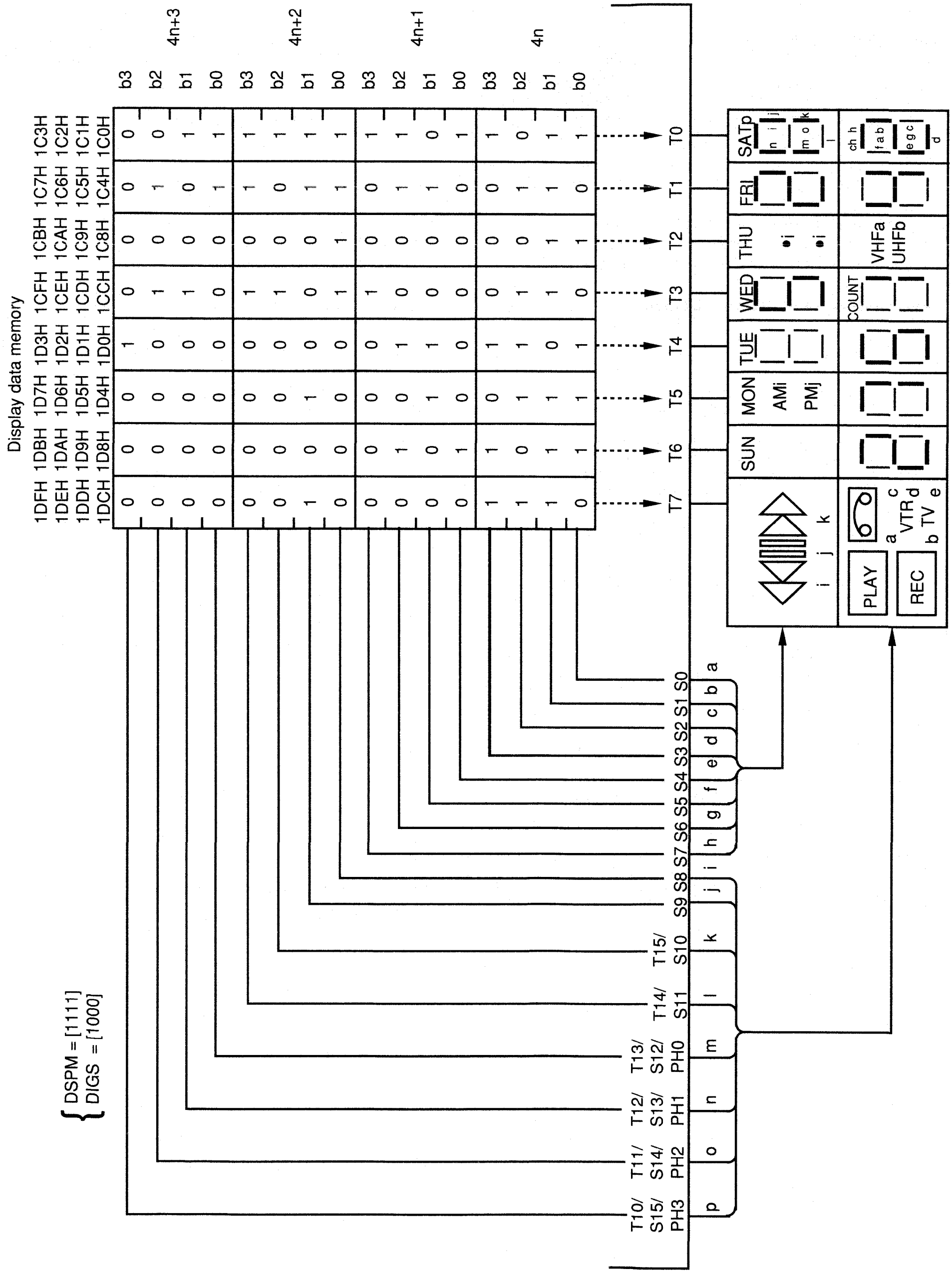

Display data memory
4n+3
4n+2
4n+1
4n
b3 b2 b1 b0 b3 b2 b1 b0 b3 b2 b1 b0 b3 b2 b1 b0
1C3H 1C2H 1C1H 1C0H — T0: 0 0 0 1 | — — 0 0 | 0 0 1 1 | 1 1 0 1
1C7H 1C6H 1C5H 1C4H — T1: 0 0 0 1 | — — 0 0 | 0 1 1 1 | 0 1 1 1
1CBH 1CAH 1C9H 1C8H — T2: 0 0 0 1 | — — 0 0 | 0 1 1 1 | 1 1 0 1
1CFH 1CEH 1CDH 1CCH — T3: 0 0 0 0 | — — 0 0 | 0 0 0 0 | 0 1 1 0
1D3H 1D2H 1D1H 1D0H — T4: 0 0 0 1 | — — 0 0 | 0 1 0 1 | 1 0 1 1
1D7H 1D6H 1D5H 1D4H — T5: 0 0 0 1 | — — 0 0 | 0 1 1 0 | 1 1 0 1
1DBH 1DAH 1D9H 1D8H — T6: 0 0 0 0 | — — 0 0 | 0 0 0 0 | 0 1 1 1
1DFH 1DEH 1DDH 1DCH — T7: 0 1 0 0 | — — 0 1 | 0 0 0 0 | 1 1 1 1
1E3H 1E2H 1E1H 1E0H — T8: 0 0 0 1 | — — 0 0 | 0 1 1 1 | 0 0 1 1
1E7H 1E6H 1E5H 1E4H — T9: 0 1 0 1 | — — 0 1 | 0 0 0 0 | 0 0 0 0
KS1
1FEH
b3 b2
T14/S11 T15/S10
Static output
S0 a
S1 b
S2 c
S3 d
S4 e
S5 f
S6 g
S7 h
S8 i
S9 j
T13/ S12/ PH0 k
T12/ S13/ PH1 l
T11/ S14/ PH2 m
T10/ S15/ PH3 n
DSPM = [1101]
DIGS = [1001]
a b c d e f g h i j k l m n

Fig. 5.7.14 14-Segment 10-Digit Display



| Display data memory | 4n+3 |  |  |  | 4n+2 |  |  |  | 4n+1 |  |  |  | 4n |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  | b3 | b2 | b1 | b0 | b3 | b2 | b1 | b0 | b3 | b2 | b1 | b0 | b3 | b2 | b1 | b0 |  |
| 1C3H 1C2H 1C1H 1C0H | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | T0 |
| 1C7H 1C6H 1C5H 1C4H | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | T1 |
| 1CBH 1CAH 1C9H 1C8H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | T2 |
| 1CFH 1CEH 1CDH 1CCH | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | T3 |
| 1D3H 1D2H 1D1H 1D0H | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | T4 |
| 1D7H 1D6H 1D5H 1D4H | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | T5 |
| 1DBH 1DAH 1D9H 1D8H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | T6 |
| 1DFH 1DEH 1DDH 1DCH | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | T7 |

DSPM = [1111]
DIGS = [1000]

Fig. 5.7-15 16-Segment 8-Digit Display

## 6. INTERRUPT OPERATION

The µPD75268 have seven vector interrupt sources and two edge-detection testable inputs. The µPD75268's interrupt control logic has the following features to enable efficient interrupt service at high speed.

(1) Interrupt master enable flag control interrupt acknowledgement.
(2) Interrupt service start address and the MBE status during interrupt service can be freely specifiedon the vector table (quick start for interrupt service program).
(3) Capable of testing and clearing the interrupt request flag.
(4) Interrupt request can clear the Standby mode (Stop, Halt) (clear source is selectable by control of interrupt enable flag).

### 6.1 Configuration of the interrupt control circuit

The µPD75268 has seven interrupt sources. Multiple interrupts with priority are possible.
Two edge detection testable inputs are also provided.

**Table 6.1-1 Kinds of Interrupt Sources**

| Interrupt Source | Internal/external | Interrupt Priority* | Vector Interrupt Reuqest Signal (Vector Table Address) |
|---|---|---|---|
| INTBT (reference time interval signal from basic interval timer) | Internal | 1 | VRQ1 (0002H) |
| INT4 (rising edge and falling edge detection) | External | | |
| INT0 (rising or falling detection edge selection) | External | 2 | VRQ2 (0004H) |
| INT1 | External | 3 | VRQ3 (0006H) |
| INTSIO (serial data transfer and signal) | Internal | 4 | VRQ4 (0008H) |
| INTT0 (coincidence signal from timer/event counter 0) | Internal | 5 | VRQ5 (000AH) |
| INTKS (key scan timing signal from display controller) | Internal | 7 | VRQ7 (000EH) |
| INT2 (rising edge detection) | External | Testable input signals (set IRQ2 and IRQW) | |
| INTW (signal from watch timer) | Internal | | |

*: When multiple interrupts are generated concurently, the interrupt order is priority order.

The interrupt control circuit of the µPD75268 has the following functions:

(i) Hardware control vector interrupt function which can control whether an interrupt is accepted or not by interrupt enable flag (IEXXX) and interrupt mask enable flag (IME).

(ii) Interrupt start address can be arbitrarily set.

(iii) Interrupt request flag (IRQXXX) test function. (Interrupt generation can be checked by software)

(iv) Standby mode release (interrupt which releases the standby mode can be selected by interrupt enable flag)

| Address | | | | | | |
|---|---|---|---|---|---|---|
| 0002H | MBE | 0 | 0 | INTBT/INT4 | Start address | (higher 5-bits) |
| | | | | INTBT/INT4 | Start address | (lower 8-bits) |
| 0004H | MBE | 0 | 0 | INT0 | Start address | (higher 5-bits) |
| | | | | INT0 | Start address | (lower 8-bits) |
| 0006H | MBE | 0 | 0 | INT1 | Start address | (higher 5-bits) |
| | | | | INT1 | Start address | (lower 8-bits) |
| 0008H | MBE | 0 | 0 | INTSIO | Start address | (higher 5-bits) |
| | | | | INTSIO | Start address | (lower 8-bits) |
| 000AH | MBE | 0 | 0 | INTT0 | Start address | (lower 5-bits) |
| | | | | INTT0 | Start address | (higher 8-bits) |
| 000EH | MBE | 0 | 0 | INTKS | Start address | (higher 5-bits) |
| | | | | INTKS | Start address | (lower 8-bits) |

**Fig. 6.1-1 Interrupt Vector Table**

Example: Sets vector table for INTBT/INT4

| VENT1 | MBE = 0 | GOTOBT |
|---|---|---|
| ↑ | ↑ | ↑ |
| Vector table at address 0002H | MBE and RBE values set in the interrupt routing | Symbol denoting the interrupt service routine start address |

Note: The address of the vector table specified by VENTn (n = 1 to 7) is address 2n.

Example: Sets vector tables for INTBT/INT4 and INTT0: (program)

```
VENT1    MBE = 0, GOTOBT
VENT5    MBE = 0, GOTOT0
```

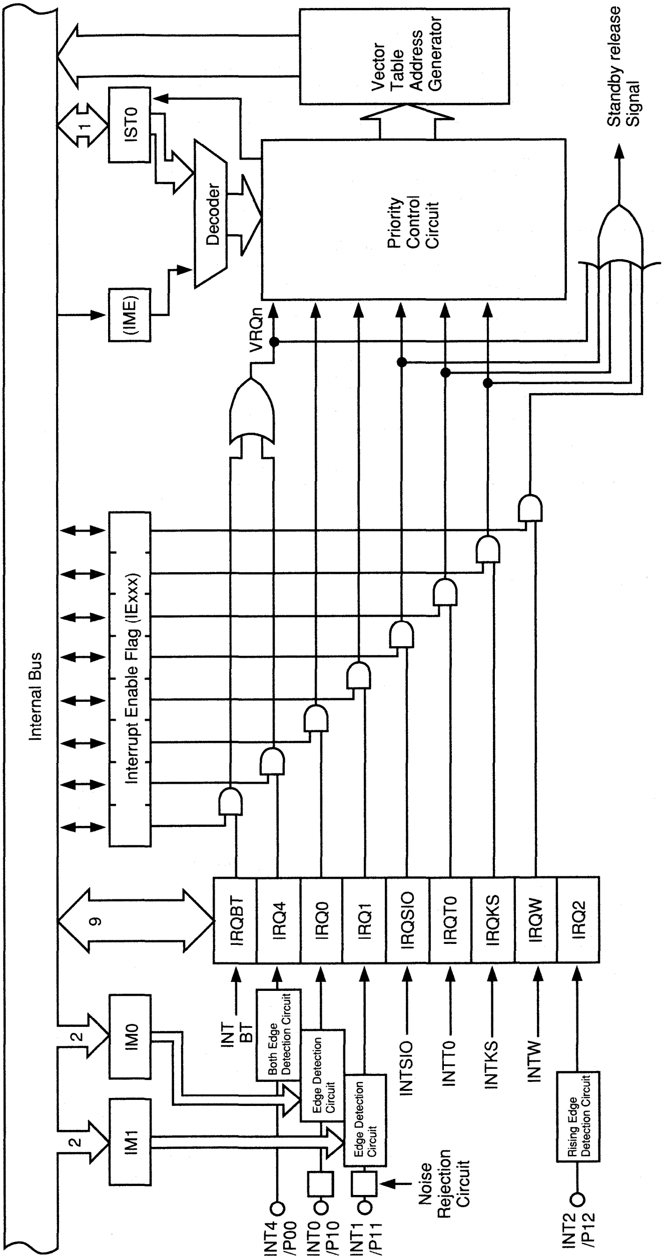


Fig. 6.1-2 Interrupt Control Circuit

### 6.2 Hardware for Interrupt Control Circuit

(1) Interrupt request flags and interrupt enable flags

The interrupt request flag (IRQXXX) is set to 1 when an interrupt request is generated and is automatically cleared to 0 when that interrupt process is executed. However, when IRQBT and IRQ4, share the vector address, the clear operation differs from this (refer to 6.6).

An interrupt enable flag (IEXXX) is provided for each interrupt request flag. An interrupt is enabled when the contents are 1, and disabled when the contents are 0.

A vector interrupt request (VRQn) is generated when the interrupt request flag has been set and the interrupt is enabled by the flag. This signal is also used to release the standby mode. The interrupt request and interrupt enable flags are controlled by bit manipulation and 4-bit memory manipulation instruction. If bit manipulation instruction is used, these flags can be directly controlled at any time regardless of the setting of the MBE flag. The interrupt enable flags are controlled by EI IEXXX and DI IEXXX instructions. SKTCLR instruction is normally used to test the interrupt request.

```
Example:   EI        IE0       ; Enable INT0
           DI        IE1       ; Disable INT1
           SKTCLR    IRQSIO    ; Skips and is IRQSIO is 1, then clears it.
```

If an interrupt request flag is set by instruction, a vector interrupt will be executed whether or not an interrupt is generated. An internal reset signal $\overline{\text{RES}}$ clears the interrupt request and interrupt enable flags to 0, and disables all interrupt.

**Table 6.2-1 Interrupt Request Flag and Interrupt Enable Flag**

| Interrupt request flag | Interrupt request flag set signal | Interrupt enable flag |
|---|---|---|
| IRQBT | Reference time interval signal from basic interval timer | IEBT |
| IRQ4 | Detection of both rising and falling edge of INT4/P00 pin input signal | IE4 |
| IRQ0 | Edge detection of INT0/P10 pin input signal. Detection edge is selected by INT0 mode register (IM0) | IE0 |
| IRQ1 | Edge detection ot INT1/P11 pin input signal. Detection edge is selected by INT1 mode register (IM1) | IE1 |
| IRQSIO | Serial data transfer completion signal of serial interface. | IESIO |
| IRQT0 | Coincidence signal from timer/event counter #0, or edge detection of TI0 pin input by setting of mode register (TM0). | IET0 |
| IRQKS | Set by Key Scan Timing signal from the display controller. | IEKS |
| IRQW | Set by the signal from the clock timer. | IEW |
| IRQ2 | Rising edge detection of INT2/P12 pin input signal. | — |

(2) Noise eliminating circuit and edge detection mode registers

Noise eliminating circuit and edge detection mode registers INT0 and INT1 are configured as shown in Fig. 6.2-1. These are external interrupt request input pins with these features: (1) eliminating noise using the sampling clock, and (2) detection edge selection. The noise eliminating circuit eliminates pulses having a pulse width narrower than the sampling clock width.

As shown in Fig. 6.2-3, a pulse two times wider than the sampling clock is accepted as an interrupt request signal. INT0 can use either one of two sampling clock rates. INT1 uses CPU clock as a sampling clock.

It should be noted that the inputs from INT0 and INT1 are made through each noise suppression circuit even when these pins are used for port input.

The edge detection mode registers (IM0, IM1) which select the edge to be detected have the formats shown in Fig. 6.2-2. Both IM0 and IM1 are set by a 4-bit memory manipulation instruction. When the $\overline{\text{RES}}$ signal is generated, all bits of this register are cleared to 0, and the rising edge is specified as the detection edge.

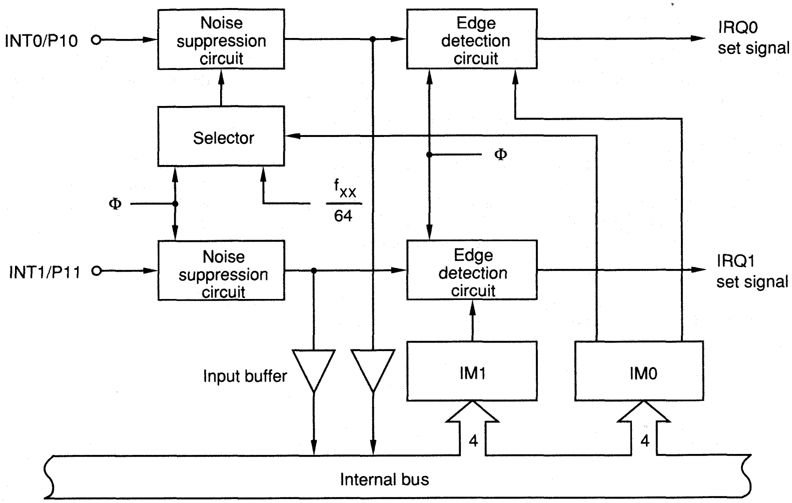
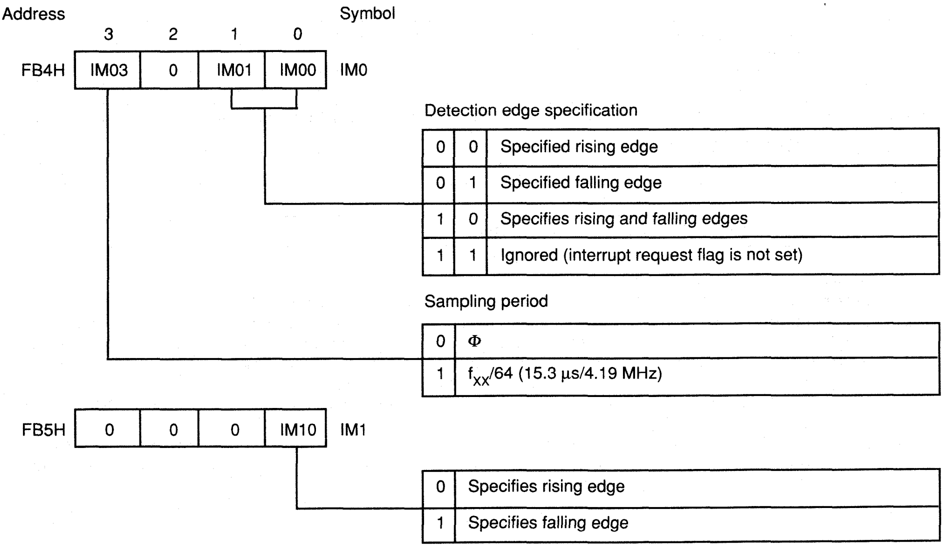


**Fig. 6.2-1**

The noise suppression circuit outputs a high level if a high level is continuously input, and outputs a low level if a low level is continuously input.



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

Detection edge specification

| IM01 | IM00 | |
|---|---|---|
| 0 | 0 | Specified rising edge |
| 0 | 1 | Specified falling edge |
| 1 | 0 | Specifies rising and falling edges |
| 1 | 1 | Ignored (interrupt request flag is not set) |

Sampling period

| IM03 | |
|---|---|
| 0 | Φ |
| 1 | $f_{XX}$/64 (15.3 µs/4.19 MHz) |

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| FB5H | 0 | 0 | 0 | IM10 | IM1 |

| IM10 | |
|---|---|
| 0 | Specifies rising edge |
| 1 | Specifies falling edge |

**Fig. 6.2-2 Edge Detection Mode Register Format**

Note: Up dating the Edge Detect Mode register may cause the Interrupt Request flag to be set. When doing this updating, disable interrupt after resetting the Interrupt Request flag by using the CLR1 instruction. If $f_{XX}$/64 is selected for the sampling clock by updating IM0, it is necessary to reset the Interrupt Request flag 16 machine cycles after updating the Mode register.

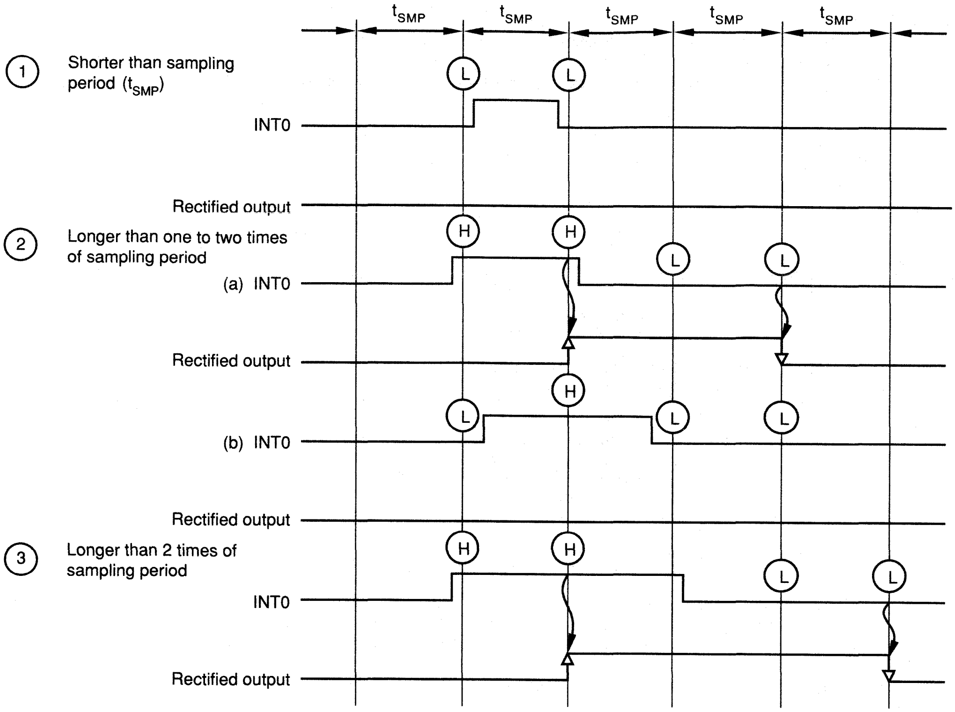
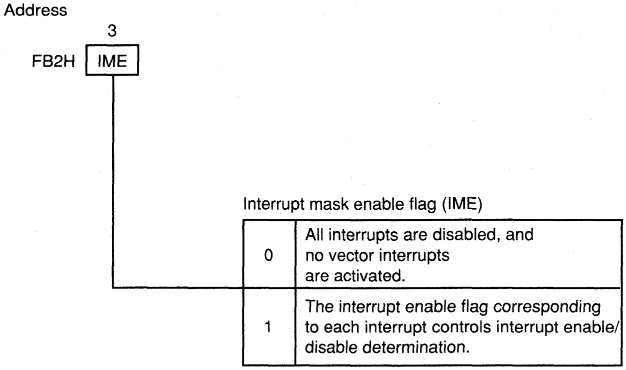


**Fig. 6.2-3 I/O Timing of Noise Suppression Circuit**

(2) Interrupt master enable flag (IME)

The interrupt master enable flag (IME) specifies whether to enable or disable the acceptance of each interrupt. The EI/DI instruction is used to set IME.

A $\overline{\text{RESET}}$ signal occurence clears IME to 0, disabling the acceptance of any interrupts.



Address

3

FB2H IME

Interrupt mask enable flag (IME)

| | |
|---|---|
| 0 | All interrupts are disabled, and no vector interrupts are activated. |
| 1 | The interrupt enable flag corresponding to each interrupt controls interrupt enable/ disable determination. |

**Fig. 6.2- 4 Format of IME**

(3) Interrupt status flag

The interrupt status flag (IST0), which is contained in the PSW, indicates the status of processing currently executed by the CPU.

By using the content of this flag, the interrupt priority control circuit controls multiple interrupts as indicated in Table 6.2-4.

A 4-bit manipulation instruction or bit manipulation instruction can be used to set or reset IST0, so that multiple interrupts are enabled by changing the current status of execution. IST0 can be manipulated on a single-bit basis at any time regardless of MBE setting.

Before IST0 is manipulated, the DI instruction must be executed to disable interrupts, then the EI instruction must be executed to enable interrupts.

IST0 is saved to stack memory together with the other PSW bits when an interrupt is accepted, then is automatically set to 1. When the RETI instruction is executed, IST0 is set to the orginal value (0).

A $\overline{\text{RESET}}$ signal occurrence clears the contet of the flag 0.

**Table 6.2-4 IST0 and Interrupt Processing Statuses**

| IST0 | Status of current processing | Processing by CPU | Acceptable interrupt request | After an interrupt is accepted |
|---|---|---|---|---|
| | | | | IST0 |
| 0 | Status 0 | CPU is executing normal program processing. | Any interrupts can be accepted. | 1 |
| 1 | Status 1 | CPU is processing interrupts. | No interrupts can be accepted. | - |

### 6.3 Interrupt Sequences

When an interrupt occurs, it is processed using the procedure shown in Figure 6.3.-1.

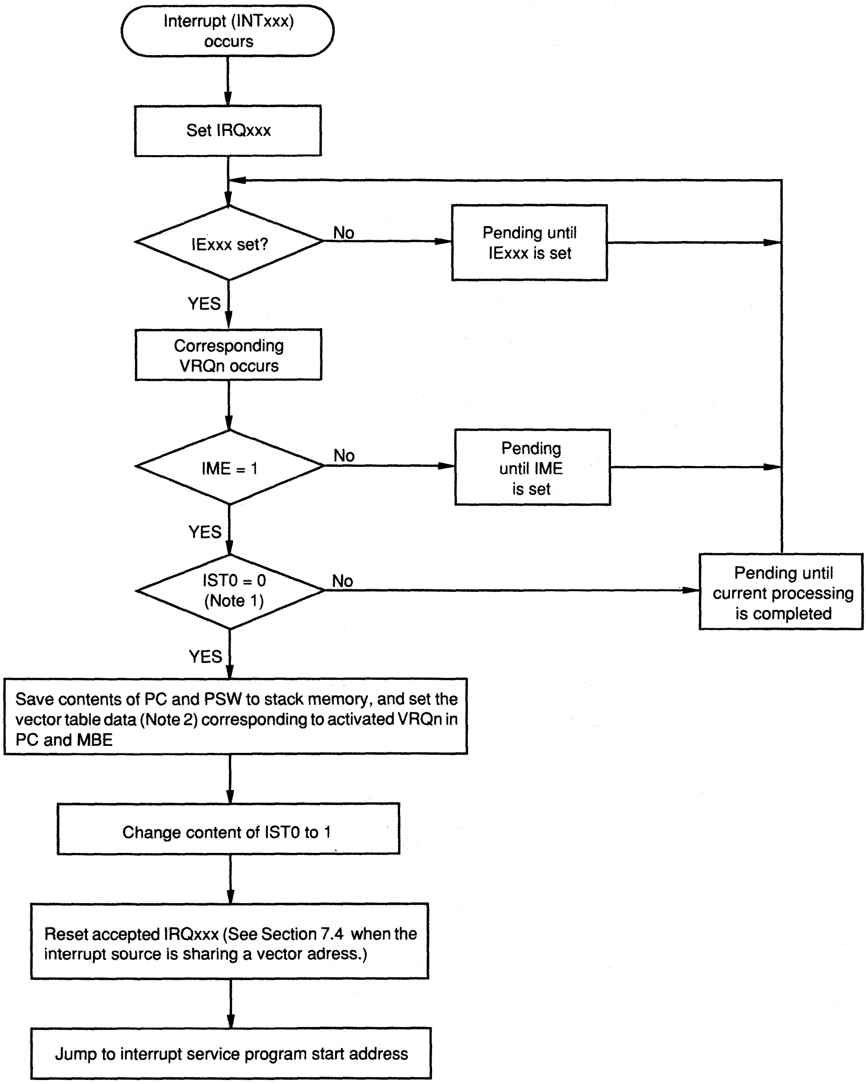

Interrupt (INTxxx) occurs
Set IRQxxx
IExxx set?
No
Pending until IExxx is set
YES
Corresponding VRQn occurs
IME = 1
No
Pending until IME is set
YES
IST0 = 0 (Note 1)
No
Pending until current processing is completed
YES
Save contents of PC and PSW to stack memory, and set the vector table data (Note 2) corresponding to activated VRQn in PC and MBE
Change content of IST0 to 1
Reset accepted IRQxxx (See Section 7.4 when the interrupt source is sharing a vector adress.)
Jump to interrupt service program start address

**Fig. 6.3.-1 Interrupt Processing Sequence**

Note 1. IST0 is the interrupt status flag (bit 2 of the PSW). (See Table 6.2-4.)

Note 2. An interrupt service program Start address and an MBE setting value at the Start of interrupt are stored in each vector table.

(3) Multiple interrupts which modify the interrupt status flag
As can be seen from Table 6.2-4, multiple interrupt is possible by modifying the interrupt status flag by program. That is, multiple interrupts can be generated by changing IST0 to 0 and 1 by interrupt service routine.
This method is used when desiring to enable two or more multiple interrupts.
IST0 is modified in the interrupt disabled state by DI instruction in advance.

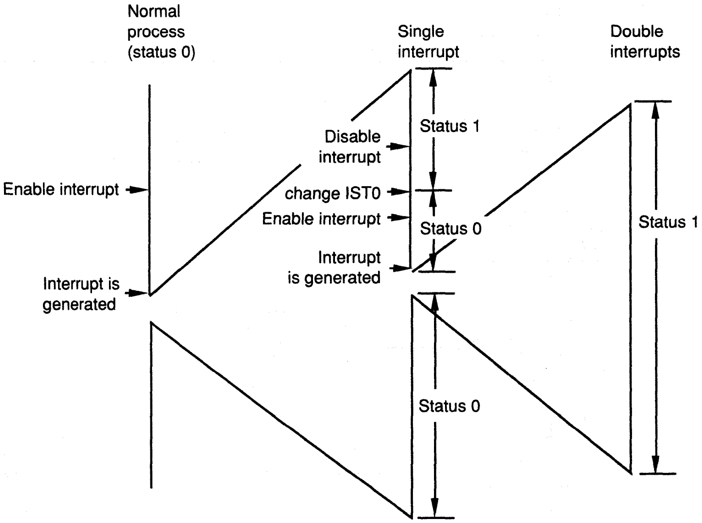


**Fig. 6.3-2 Multi-Interrupts by Modifying Interrupt Status Flag**

### 6.4 Shared Vector Address Interrupt Service

Interrupt source INTBT and INT4 share the vector table . In this case, the interrupt source is selected using the methods described below.

(1) To use only one interrupt source
The interrupt enable flag of one of the two interrupt sources sharing the vector table is set to 1, and the other interrupt enable flag is cleared to 0. The interrupt source whose interrupt enable flag is set to 1 (IExxx = 1) generates an interrupt request. The interrupt enable flag will be reset when that interrupt request is acknowledged. (Same as interrupts with unshared vector address.)

(2) To use both interrupt sources
First, the interrupt enable flags corresponding to the two interrupt sources are each set to 1. An OR-ed result of the interrupt request flags of the two interrupt sources becomes the interrupt request. In this method, neither of the interrupt request flags will be reset even if the interrupt request flag settings is acknowledged.
Therefore, in this case, an interrupt service routine needs to be used to determine the interrupt source generating the interrupt. This is done by executing DI instruction at the beginning of the interrupt service routine, then checking the interrupt request flag by executing SKTCLR instruction.
If both of the interrupt request flags are set when tested, an interrupt request will remain, even if one of the request flags is reset. If the remaining interrupt is assigned high level interrupt, it causes double interrupt service execution to be entered. This means that the interrupt request which was not tested is serviced first.
On the other hand, if the remaining interrupt is assigned low level interrupt, it is left pending.
Consequently, the interrupt request which was tested will be serviced first. Therefore, the shared interrupt identification method will differ depending on whether the interrupt is a high level interrupt or not, as shown below.

**Table 6.4-1 Shared Interrupt Identification Methods**

| | |
|---|---|
| High level interrupt | Disable interrupts and test the interrupt request flag for the interrupt to be serviced first. |
| Low level interrupt | Test the interrupt request flag for the interrupt to be serviced first. |

Example 1: Use both INTBT and INT4 as high level interrupts and service INT4 first:

```
          DI
          SKTCLR    IRQ4      ; IRQ4 = 1?
          BR        VSUBBT
            :                 INT4 service routine
          EI
          RETI
            :
VSUBBT:   CLR1      IRQBT
            :
            :                 INTBT service routine
          EI
          RETI
```

Example 2: Use both INTBT and INT4 as low level interrupts and service INT4 first:

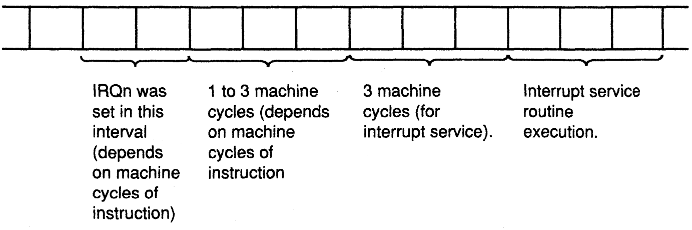
```
          SKTCLR    IRQ4      ; IRQ4 = 1
          BR        VSUBBT
            :                 INT4 service routine
            :
          RETI
            :
  VSUBBT: CLR1      IRQBT     INTBT service routine
            :
            :
          RETI
```

### 6.5 Machine Cycles before Interrupt Service Start

The following shows the machine cycle required on the μPD75268 after an interrupt request flat is set and before the execution of the interrupt service routine is started:

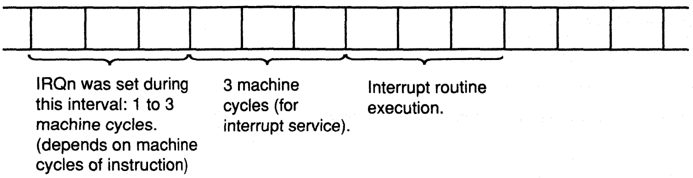
(1) IRQn was set when a manipulation instruction to data memory address FBXH (interrupt hardware) was executed:

FBXH address manipulation instruction


IRQn was set in this interval (depends on machine cycles of instruction)
1 to 3 machine cycles (depends on machine cycles of instruction
3 machine cycles (for interrupt service).
Interrupt service routine execution.

Interrupt routine service is started in a maximum of six machine cycles after handling instruction of data memory address FBXH (described above) terminates. (If FBXH address handling instructions are contiguous, interrupt routine service is started in a maximum of six machine cycle after the last handling instruction terminates.)

(2) IRQn was set when an instruction other than that given in (1) was executed:


IRQn was set during this interval: 1 to 3 machine cycles. (depends on machine cycles of instruction)
3 machine cycles (for interrupt service).
Interrupt routine execution.

This case requires up to 6 machine cycles max.

### 6.6 Effective Use of Interrupts

Note the following for effective use of interrupts:

(1) Set MBE = 0 in an interrupt service routine:
The programmer need not be aware of memory bank configuration when programming if he or she uses addresses 0 through 7FH of the data memory in interrupt service routines and specifies MBE = 0 on the interrupt vector table.
If memory bank 1 must be used for programming convenience, save the memory bank select register value by using the PUSH BS instruction before selecting memory bank 1.

(2) Use software interrupt for debugging:
If an interrupt request flag is set by an instruction, the CPU operates as if an interrupt had occurred.
Debugging for irregular interrupt service or more than one interrupt occurred at the same time can be efficiently done by setting the interrupt request flag with an instruction.

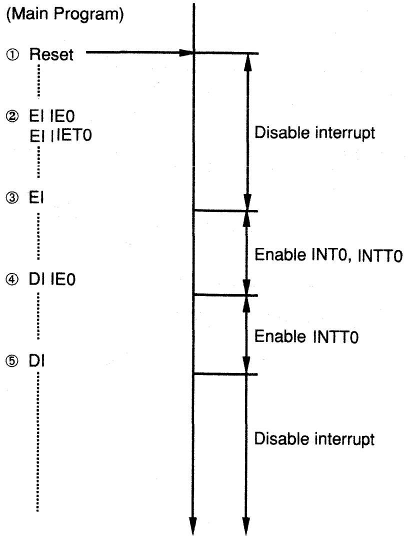
### 6.7 Applications of Interrupt

To make the interrupt function available, do the following setup in the main program:

(1) Set the interrupt enable flag to be used (EI IEXXX instruction).
(2) When using INT0 or INT1, select active edge (set IM0 or IM1).
(3) Set the interrupt master enable flag (EI instruction).

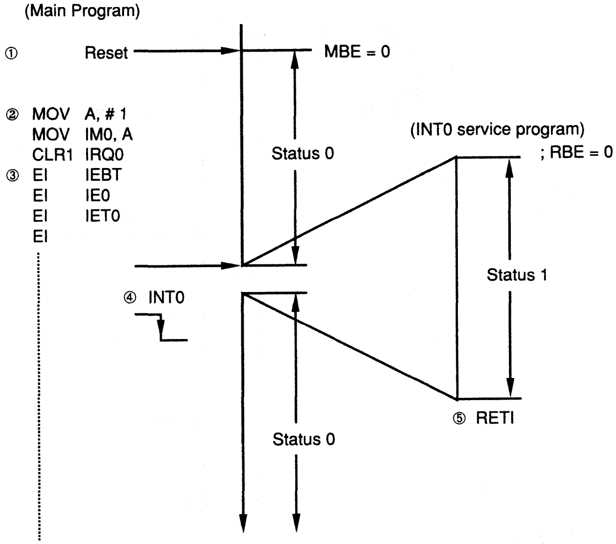
For an interrupt service program, the MBE is set by the vector table, therefore no register save/restore operation is needed, and the execution of the interrupt service program can be immediatly started. Use the RETI instruction to return control from the interrupt service program to the main program.

(1) Enable/disable interrupt



① Reset ($\overline{\text{RES}}$) disable all interrupts.
② The EI IEXXXinstruction sets an interrupt enable flag. At this point all interrupts are still disabled.
③ The EI instruction sets the interrupt master enable flag. This enables INT0 and INTT0.
④ The DI IEXXX instruction clears the interrupt enable flag, which disables INT0.
⑤ The DI instruction disables all interrupts again.

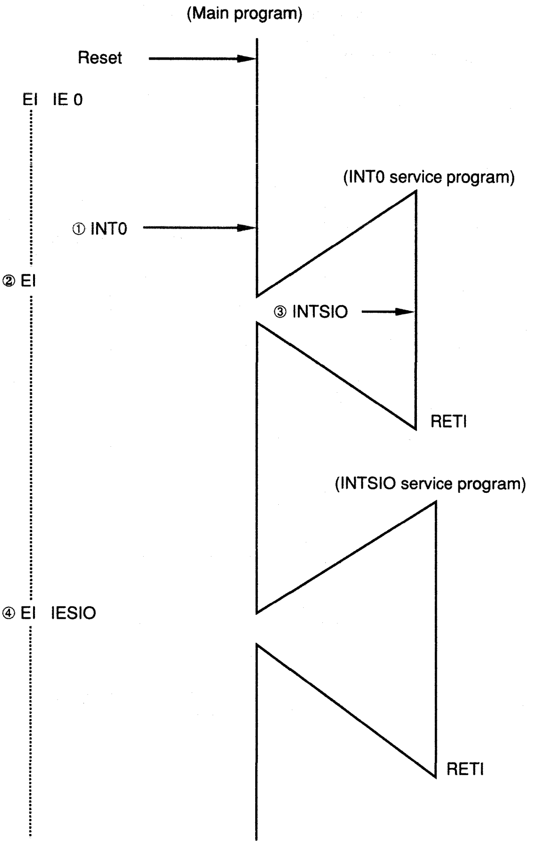
(2) Example using INTBT, INT0 (falling edge active) and INTT0.



```
①      Reset

② MOV  A, # 1
  MOV  IM0, A
  CLR1 IRQ0
③ EI   IEBT
  EI   IE0
  EI   IET0
  EI
```

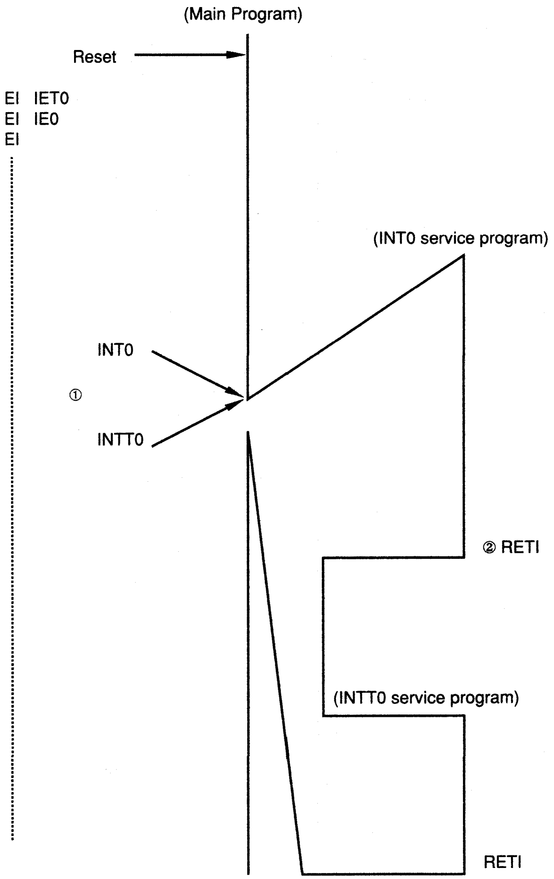
① Reset disables all interrupts and selects status 0.
② Sets INT0 for falling edge active.
③ Enables interrupt with the EI IEXXX and EI instructions.
④ INT0 interrupt service program is started at the falling edge of INT0. The status is set to one and all interrupts are disabled.
⑤ The RETI instruction returns control from the interrupt service program to main program. The status returns to 0 and all interrupts are enabled again.

(3) Execution of pending interrupt – interrupt input during interrupt disable state –



① If INT0 is set during interrupt disable state, the interrupt request flag remains pending.
② The INT0 service program is started when interrupt is enabled by the EI instruction.
③ Same as ① above.
④ The INTSIO service program is started when the suspended INTSIO is enabled.

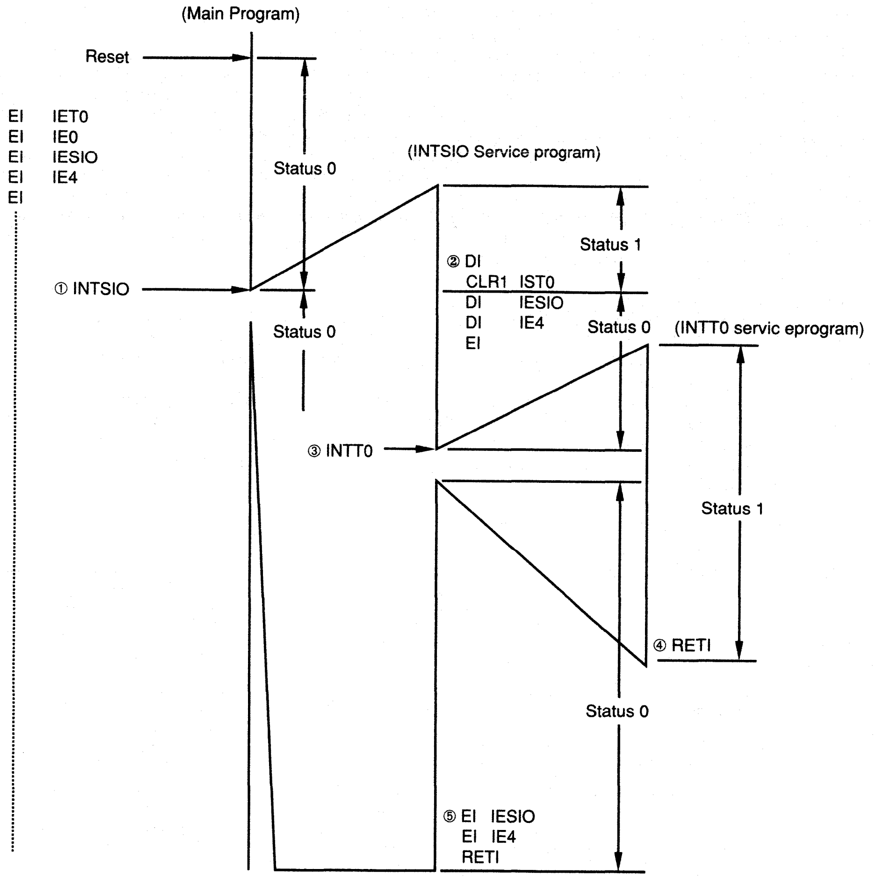
(4) Execution of pending interrupt – two low-level interrupts occuring at the same time



① If two low-level interrupts INT0 and INTT0 occured at the same time (during execution of the same instruction), the INT0 with a higher priority will be serviced first (INTT0 remains pending):

② The execution of the INTT0 service program which was pending is initiated when the INT0 service program is ended by the RETI instruction.

(5) Enable two double interrupts - INTT0 and INT0 enable double interrupt while INTSIO and INT4 enable single interrupt.-



① If an INTSIO for which double interrupt is not enabled has occured, the INTSIO service program is started with the status set to one.
② IST0 is cleared to reset status to 0. The INTSIO and INT4 for which double interrupt is not enabled are disabled.
③ Double interrupt is serviced when the INTT0 occured. The status is set to 1 to disable all interrupts.
④ Status returns to 0 when the INTT0 service is completed.
⑤ The INTSIO and INT4 are enabled and control returns to the main program.

## 7. STANDBY FUNCTION

To make the most of low current consumption, which is one feature of the CMOS process, the μPD75268 can stop CPU operation in the standby mode making current consumption by the CPU very small.
The μPD75268 standby mode includes the STOP mode and HALT mode.
The STOP mode stops the main system clock oscillator. In this mode, CPU current consumption consists almost entirely of leakage current. Data memory can also be held with low supply voltage (up to $V_{DD}$ = 2 V). This features is useful for maintaining the data memory contents with very low current consumption. Since the μPD75268 STOP mode can be released by using an interrupt request, intermittent operation can also be performed. However, if processing must be started immediately when an interrupt request is made, note that the wait time required to ensure oscillator stability is taken when the STOP mode is released.
The HALT mode continues system clock oscillator operation but stops the CPU clock (Φ) supply; thus, CPU operation is stopped. Although the HALT mode is inferior to the STOP mode for reduction of current consumption, it is useful to restart processing immediately according to an interrupt request or for performing intermittent operations such as watch operation.
In either mode, all the register, flag, and data memory contents immediately before the standby mode is entered are held. The input/output port output latch state and output buffer state are also held. The input/output port state is handled beforehand so that the current consumption of the entire system is minimized.

Cautions on Use of Standby Mode:

1. The STOP mode can be used only when μPD75268 operation uses the main system clock. (Subsystem clock oscillation cannot be stopped.) The HALT mode can be used when the μPD75268 uses either main system or subsystem clock.
2. If STOP mode is selected when the FIP controller/driver is operating, a malfunction may occur. Stop these hardware operations before selecting STOP mode.
3. Although efficient operation with low current consumption and low voltage can be performed by using the clock change function between the CPU and system clocks in combination with the standby mode, time (described in 5.2.3) is required from selection of a new clock by setting the control register until operation is started by the newly selected clock.
   Thus, to use the clock change function and the standby mode in combination, set the standby mode within the time required for the clock change.

### 7.1 Standby Mode Setting and Operating State

**Table 7.1-1 Operating State in Standby Mode**

| | | STOP mode | HALT mode |
|---|---|---|---|
| Setting instruction | | STOP instruction | HALT instruction |
| System clock when standby mode is set. | | Can be set only during main system clock. | Can be set during either main system or subsystem clock. |
| Operating state | Clock oscillator | Only the main system clock oscillator is stopped. | Only CPU clock Φ is stopped (oscillation is continued). |
| | Basic interval timer | Operation stop | Operation (IRQBT is set at reference time intervals.) |
| | Serial interface | Can operate only when external $\overline{SCK}$ input is selected for serial clock. | Can operate, when clock other than Φ is selected. |
| | Timer/event counter | Can operate only when TI0 pin input is selected for count clock. | Can operate. |
| | Watch timer | Can operate when $f_{XT}$ is selected for count clock. | Can operate. |
| | FIP controller | Disabled! | |
| | CPU | Operation stop | |
| Data to be retained | | Data is retained in all register and data memory, such as general-purpose registers, flags, mode registers, and output latches which are not operating in Standby mode. | |

**Table 7.1-1 Operating State in Standby Mode (cont'd)**

| | STOP mode | HALT mode |
|---|---|---|
| Release signal | Interrupt request signals (except INT0, INT1, and INT2) being enabled by the interrupt enable flag or the $\overline{\text{RESET}}$ input. | Interrupt request signals (except INT0, INT1, and INT2) being enabled by the interrupt enable flag or the $\overline{\text{RESET}}$ input. |

The STOP mode is set by using the STOP instruction to set PCC bit 3; the HALT mode is set by the using HALT instruction to set PCC bit 2.

To change the CPU clock by using the low-order two bits of PCC, a time lag may occur from PCC rewrite to CPU clock change. Thus, to change the clock before the standby mode is entered or after the standby mode is released, set the standby mode within the number of machine cycles required to change the CPU clock after PCC is rewritten.

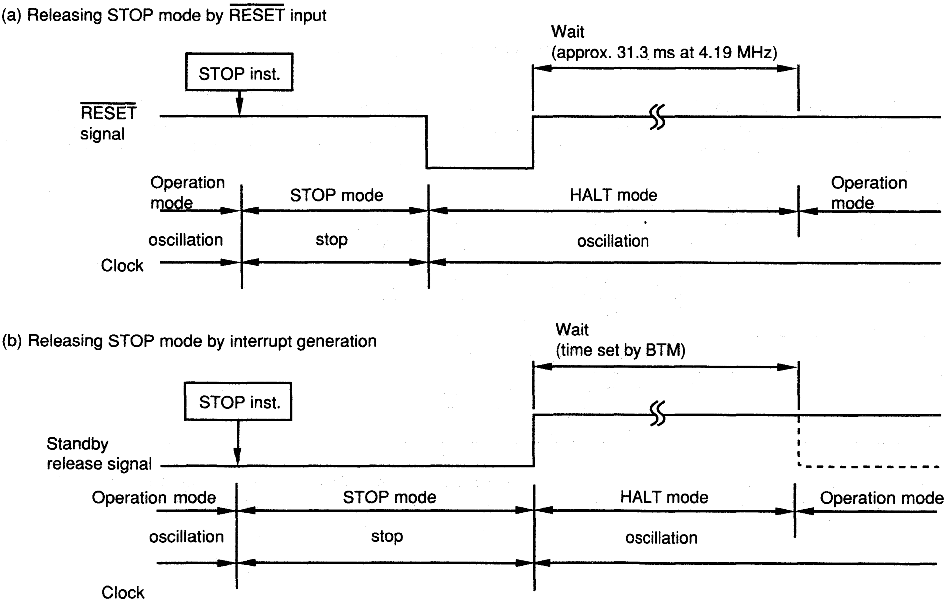
While operation stops during the standby mode, data is held in all registers and data memory such as general-purpose registers, flags, mode register, and output latches.

Caution: 1. When the STOP mode is set, X1 input is short-circuited to $V_{SS}$ (GND potential) internally to suppress crystal oscillator leakage. Therefore, so not use the STOP mode in a system using external clock as a main system slock.

2. Since an interrupt request signal is used to release the standby mode, if both interrupt request and enable flags are set for an interrupt source, the standby mode is immediately released. Thus, for the STOP mode, the HALT mode is entered immediately after execution of the STOP instruction, a wait is followed according to the setup time of the BTM register, then a return is made to the opertion mode.

**7.2 Standby Mode Release**

The standby mode (STOP or HALT) is released when an interrupt request signal (except INT0) enabled with an interrupt enable flag occurs or $\overline{\text{RESET}}$ is input. Fig. 7.2-1 shows the standby mode release operation.

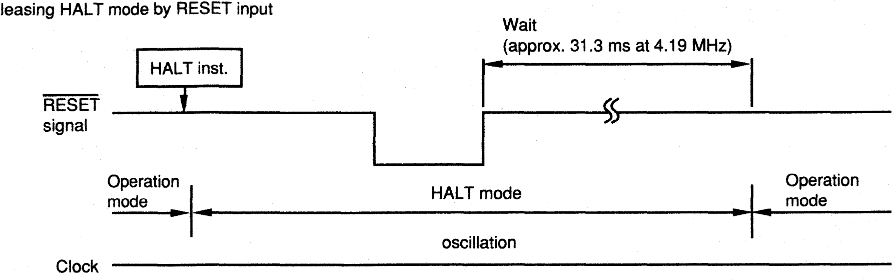

(a) Releasing STOP mode by $\overline{\text{RESET}}$ input

(b) Releasing STOP mode by interrupt generation

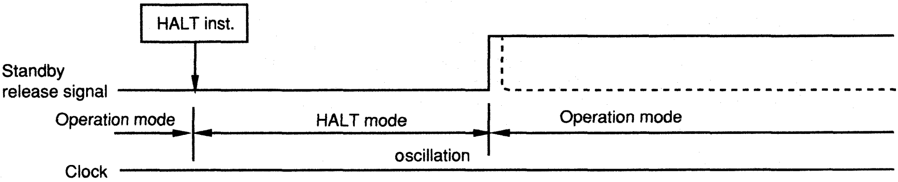
Note: Broken line shows a case when standby releasing interrupt request is acknowledged (IME = 1).

**Fig. 7.2-1 Standby Mode Release Operation**

(c) Releasing HALT mode by $\overline{\text{RESET}}$ input



(d) Releasing HALT node by interrupt generation



Note: Broken line shows a case when standby releasing interrupt request is acknowledged (IME = 1).

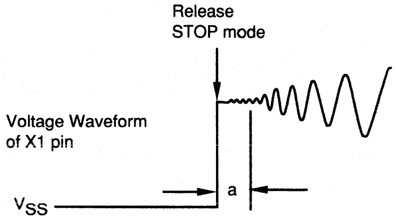
**Fig. 7.2-1 Standby Mode Release Operation (cont'd)**

If the STOP mode is released when an interrupt occurs, the wait time is determined by BTM setting. (See Table 7.2-1.) The time until oscillation becomes stable varies according to the type of oscillator being used and the supply voltage when the STOP mode is released. Therefore, select the wait time according to the conditions of use and set BTM before setting the STOP mode.

**Table 7.2-1 Wait Time Selection by Using BTM**

| BTM3 | BTM2 | BTM1 | BTM0 | Wait time* ( ) indicates $f_{XX}$ = 4.19 MHz |
|---|---|---|---|---|
| – | 0 | 0 | 0 | APPROX. $2^{20}/f_{XX}$ (Approx. 250 ms) |
| – | 0 | 1 | 1 | APPROX. $2^{17}/f_{XX}$ (Approx. 31.3 ms) |
| – | 1 | 0 | 1 | APPROX. $2^{15}/f_{XX}$ (Approx. 7.82 ms) |
| – | 1 | 1 | 1 | APPROX. $2^{13}/f_{XX}$ (Approx. 1.95 ms) |
| Other than above | | | | Use Prohibited |

Note: The wait time when STOP mode is released does not include the time until the clock begins to oscillate after STOP mode is released ((a) in the figure below) regardless of whether STOP mode was released by $\overline{\text{RESET}}$ input or interrupt generation.



### 7.3 Operation After Standby Mode is Released

(1) If the standby mode is released when $\overline{\text{RESET}}$ is input, normal reset operation is performed.

(2) If the standby mode is released when an interrupt request occurs, the contents of the interrupt master enable flag (IME) determine whether or not a vectored interrupt is made when the CPU restarts instruction execution.

(a) When IME = 0

After the standby mode released, execution restarts at the NOP instruction next to the standby mode setting instruction. The interrupt request flags are held.

(b) When IME = 1

After the standby mode is released, two instructions following the standby mode setting instruction are executed before a vectored interrupt is executed.

However, if the standby mode is released by using INTW (testable input), no vectored interrupt will occur; processing as in (a) above is performed.

### 7.4 Standby Mode Application

To use the standby mode, follow the procedure described below:

1) Detect a standby mode setting source such as interrupt input or port input for power off (it is effective to use INT4 for power off detection).
2) Handle input/output ports so that current consumption is minimized.
3) Specify interrupt to release the standby mode. (It is effective to use INT4. Clear the interrupt enables flags so as not to release the standby mode.)
4) Specify operation after the standby mode is released (set IME depending on whether or not interrupt service is made).
5) Specify the CPU clock after the standby mode is released. (To change the clock, wait for the required number of machine cycles before setting the standby mode.)
6) Select the wait time in release of standby mode.
7) Set the standby mode by using the STOP or HALT instruction.

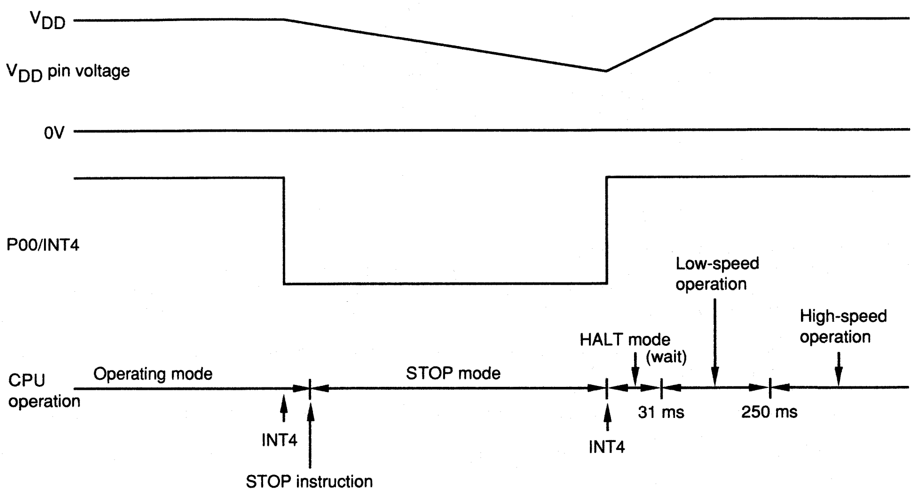
Use of the standby mode and system clock change function in combination enables the μPD75268 to operate at low current consumption and low voltage.

(1) Example of STOP mode application

Use the STOP mode under the following conditions

— Set the STOP mode when the INT4 falling edge is input and release it when the rising edge is input. (Do not use INTBT.)
— Place all input/output ports in high impedance.
— Use interrupts INT0 and INTT0 in the example program; however, do not use them to release the STOP mode.
— Enable interrupts after the STOP mode is released.
— After the STOP mode is released, start operation on the minimum speed CPU clock, and in 250 ms, change it to high-speed clock.
— Set the wait time for STOP mode release to about 31.3 ms.
— After the STOP mode is released, wait for 31.3 ms for the power supply to become stable. Check the P00/INT4 pin twice and remove chattering.

Timing Chart



<Sample program> (INT4 service program, MBE = 0)

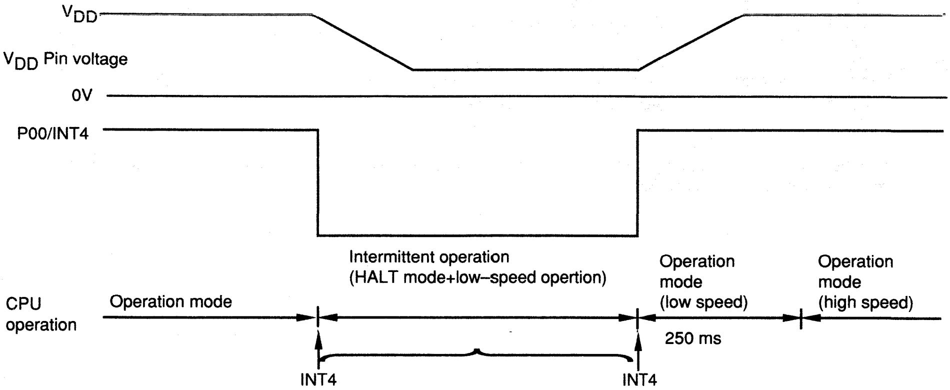
```
VSUB4:      SKT     PORT0.0       ; P00=1?
            BR      PDOWN         ; Power-off
            SET1    BTM. 3        ; Power-on
WAIT1:      SKT     IRQBT         ; Wait for 31.3 ms
            BR      WAIT1
            SKT     PORT0.0       ; Chattering check
            BR      PDOWN
            MOV     A,#1000B
            MOV     BTM, A
WAIT2:      SKT     IRQBT
            BR      WAIT2
            MOV     A, #0011B
            MOV     PCC, A        ;Set high-speed mode
            MOV     XA, #xxH      ; Set port mode register
            MOV     PMGm, XA
            EI      IE0
            EI      IET0
            RETI
PDOWN:      MOV     A, #0         ; Lowest-speed mode
            MOV     PCC, A
            MOV     XA, #00H
            MOV     DSPM, A       ; Stop FIP operation
            MOV     PMGA, XA      ; I/O port high impendance
            MOV     PMGB, XA
            DI      IE0           ; Disable INT0 and INTT0
            DI      IET0
            MOV     A, #1011B
            MOV     BTM, A        ; Wait time ≒ 31.3 ms
            STOP                  ; Set STOP mode
            NOP
            RETI
```

(2) HALT mode application

Perform intermittent operation under the following conditions

— Change to the subsystem clock on the falling edge of INT4.
— Stop oscillation of the main system clock and set the HALT mode.
— Perform intermittent operation at 0.5sec. intervals during the standby mode.
— Again change to the main system clock on the rising edge of INT4.
— Do not use INTBT.

(Timing Chart)



```
Example (Initialization)
         MOV     A, #0011B
         MOV     PCC, A      ; High speed mode
         MOV     XA, #07H    ; 32 kHz
         MOV     WM, XA
WAIT0:   SKTCLR  IRQW        ; 32 kHz O.K.?
         BR      WAIT0
         MOV     XA, #04
         MOV     WM, XA      ; Main system clock
         EI      IE4
         EI      IEW
         EI                  ; Enable interrupt
(Main routine)
         SKT     PORT0.0     ; Power OK?
         HALT                ; Power down mode
         NOP                 ; Power OK?
         SKTCLR  IRQW        ; Is 0.5sec. flag set?
         BR      MAIN        ; NO
         CALL    WATCH       ; Watch subroutine
MAIN:      :
           :
           :
           :
(INT4 service routine)
INT4:    SKT     PORT0.0     ; Power OK?, MBE=0, RBE=0
         BR      PDOWN
         CLR1    SCC.3       ; Start oscillation of main system clock
         MOV     A, #8
         MOV     BTM, A
WAIT1:   SKT     IRQBT       ; Wait for 250 ms
         BR      WAIT1
```

```
        SKT     PORT0.0     ; Check chattering
        BR      PDOWN
        CLR1    SCC.0       ; Change to main system clock
        MOV     XA, #04H    ; Main system clock
        MOV     WM, XA
        RETI
PDOWN:  MOV     XA, #05H    ; Subsystem clock
        MOV     WM, XA
        MOV     XA, #00H
        MOV     DSPM, A     ; FIP display off
        SET1    SCC.0       ; Change to subsystem clock
        MOV     A, #6
WAIT2:  INCS    A           ; Wait for 32 machine cycles
        BR      WAIT2
        SET1    SCC.3       ; Stop oscillation of main system clock
        RETI
```

Note: When the system clock is changed after power on from main system clock to subsystem clock, change the system clock after the subsystem clock is stabilized.

## 8. $\overline{\text{RESET}}$ OPERATION

The µPD75268 is reset by $\overline{\text{RESET}}$ signal input. When the µPD75268 is reset, the devices are initialized as indicated in Table 8-1. Figure 8-1 shows the timing of reset operation.

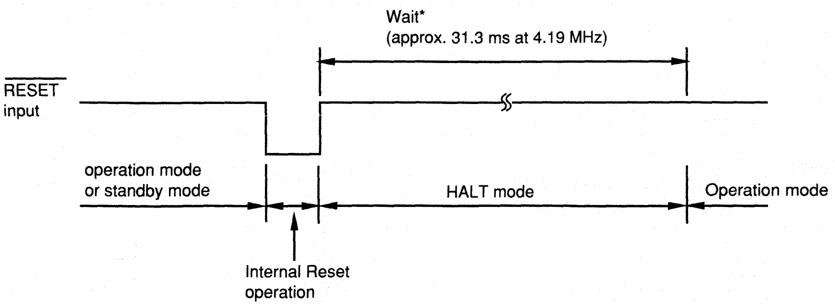

Wait*
(approx. 31.3 ms at 4.19 MHz)
$\overline{\text{RESET}}$ input
operation mode or standby mode
HALT mode
Operation mode
Internal Reset operation

**Fig. 8.1-1 Reset Operation by $\overline{\text{RESET}}$ input**

**Table 8.1-1 Status of Each Hardware After Reset**

| Hardware | | $\overline{\text{RESET}}$ input while in standby mode | $\overline{\text{RESET}}$ input while in operation mode or power-on reset. |
|---|---|---|---|
| Program counter (PC) | | Sets the lower 5 bits of address 0000H to PC 12 - 8 and the contents of address 0001H to PC7-0. | same as the left |
| PSW | Carry flag (CY) | Retained | Undefined |
| | Skip flags (SK0-2) | 0 | 0 |
| | Interrupt status flags (IST0) | 0 | 0 |
| | Bank enable flags (MBE) | Sets the contents of bit 7 of address 0000H of the program memory to the MBE flag. | same as the left |
| Stack pointer (SP) | | Undefined | Undefined |
| Data memory (RAM) | | Retained (Note 1) | Undefined |
| General-purpose registers (X, A, H, L, D, E, B, C) | | Retained | Undefined |
| Bank selection registers (MBS) | | 0 | 0 |
| Basic interval timer | Counter (BT) | Undefined | Undefined |
| | Mode register (BTM) | 0 | 0 |
| Timer/ event counter | Counter (T0) | 0 | 0 |
| | Mode register (TMOD0) | FFH | FFH |
| | Mode register (TM0) | 0 | 0 |
| Watch timer | Mode register (WM) | 0 | 0 |
| Serial interface | Shift register (SIO) | Retained | Undefined |
| | Mode register (SIOM) | Bit 4 is set to 1 all other bits are 0 | same as left |

**Table 8.1-1 Status of Each Hardware After Reset (Cont'd)**

| Hardware | | $\overline{\text{RESET}}$ input while in standby mode | $\overline{\text{RESET}}$ input while in operation mode or power-on reset. |
|---|---|---|---|
| Clock generator | Processor clock control register (PCC) | 0 | 0 |
| | System clock control register (SCC) | 0 | 0 |
| Interrupts | Interrupt request flag (IRQXXX) | Reset (to 0) | Reset (to 0) |
| | Interrupt enable flag (IEXXX) | 0 | 0 |
| | Interrupt master enable fag (IME) | 0 | 0 |
| | INT0, 1 mode register (IM0, IM1) | 0,0 | 0,0 |
| Digital port | Output buffer | Off | Off |
| | Output latch | Cleared (to 0) | Cleared (to 0) |
| | Input output mode register (PMGA, B) | 0 | 0 |
| Port H | Output latches | Retained | Undefined |
| FIP controller/ driver | Mode register | 0 | 0 |
| | Display data memory | Retained | Undefined |
| | Output buffers | Off | Off |

Notes 1: Data in address 0F8H to 0FDH of the data memory becomes undefine under the $\overline{\text{RESET}}$ signal is input.

## 9. ELECTRICAL CHARACTERISTICS

**Absolute Maximum Ratings** ($T_a$ = 25°C)

| Item | Symbol | Conditions | Rating | Units |
|---|---|---|---|---|
| Power supply voltage | $V_{DD}$ | | –0.3 to +7.0 | V |
| | $V_{LOAD}$ | | $V_{DD}$ –40 to $V_{DD}$ +3.0 | V |
| | $V_{PRE}$ | | $V_{DD}$–12 to $V_{DD}$ +0.3 | V |
| Input voltage | $V_I$ | | –0.3 to $V_{DD}$ +0.3 | V |
| Output voltage | $V_O$ | Other than display output pins | –0.3 to $V_{DD}$ +0.3 | V |
| | $V_{OD}$ | Display output pins | $V_{DD}$ –40 to $V_{DD}$ +0.3 | V |
| High level output current | $I_{OH}$ | 1 pin other than display output pins | –15 | mA |
| | | S0 to S9 1 pin | –15 | mA |
| | | T0 to T15 1 pin | –30 | mA |
| | | Total other than display output pins | –20 | mA |
| | | Display output pins total | –120 | mA |

**Absolute Maximum Ratings (Cont'd)** ($T_a$ = 25°C)

| Item | Symbol | Conditions | Rating | Units |
|---|---|---|---|---|
| Low level output current | $I_{OL}$ | 1 pin | 17 | mA |
| | | All pins total | 60 | mA |
| Total loss (Note 1) | $P_T$ | Plastic flat | 450 | mW |
| | | Plastic shrink DIP | 600 | mW |
| Operating temperature range | $T_{opt}$ | | –40 to +85 | °C |
| Storage temperature range | $T_{stg}$ | | –65 to +150 | °C |

**Operation Power Supply Voltage** ($T_a$ = –40 to +85°C)

| Item | Conditions | Min. | Max. | Units |
|---|---|---|---|---|
| CPU (Note 2) | | (Note 3) | 6.0 | V |
| Display controller | | 4.5 | 6.0 | V |
| Other hardware (Note 2) | | 2.7 | 6.0 | V |

Note 1: Total loss calculation method

The µPD75268CW/GF has the three power losses listed below. Make the design such that the sum of these three power losses is less than the total loss $P_T$. (Use below 80% of the rating is recommended.)

① CPU loss:
Calculate from $V_{DD}$ (Max.) x $I_{DD1}$ (Max.).

② Output pins loss:
This includes normal outputs and display outputs. The loss when the maximum current flows in each output pin must be added.

③ Pull-down resistor loss:
This is the loss generated by the on-chip display output pins pull-down resistors specified by mask option.

Example: 9SEG x 11DIGIT, 4LED output, $V_{DD}$ = 5V +10%, 4.19 MHz oscillation. Assume that the maximum current flowing in the segment pins is 3 mA, the maximum current flowing in the timing pin is 15 mA, and the maximum current flowing in the LED output pins is 10 mA.
Make the FIP voltage ($V_{LOAD}$ voltage) –30 V. Make the loss of the normal outputs low.

① CPU loss:
5.5 V x 9.0 mA = 49.5 mW

② Pins loss:
Segment pins ... 2 V x 3 mA x 9 = 54 mW
Timing pin ... 2V x 15 mA = 30 mW
LED output ... (10/15 x 2 V) x 10 mA x 4 = 53 mW

③ Pull-down registers loss:

$$\frac{(30 + 5.5V)^2}{40\ k\Omega} \times 10 = 315\ mW$$

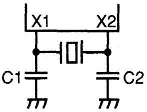
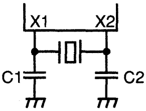
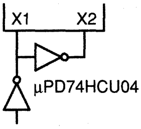
$P_T$ = ① + ② + ③ = 501.5 mW

For this example, because the allowable total loss for a shrink DIP package is 600 mW, the 501.5 mW power consumption is no problem. However, because the allowable total loss for a flat package is 450 mW, the power consumption must be lowered. The power consumption can be lowered by making the number of on-chip pull-down resistors small. For this example, the power consumption can be suppressed to 344 mW by providing an on-chip pull-down resistor at 11 of the digit outputs and four of the segment outputs and installing external pull-down resistors for the remaining five segment outputs.

Note 2: Display controller excluded.

Note 3: The power supply voltage range at which operation is possible differs with the cycle time. See the AC characteristics.

**Main System Clock Oscillator Charactaristics** (Ta = -40°C to + 85°C, $V_{DD}$ = 2.7 to 6.0 V)

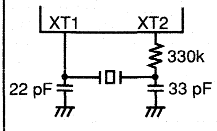
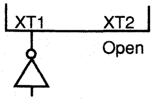
| Resonator | Recommended Constants | Item | Test Conditions | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| Ceramic oscillator (Note 3) |  | Oscillator frequency ($f_{XX}$) (Note 1) | $V_{DD}$ = oscillator voltage range | 2.0 | | 5.0 | MHz |
| | | Oscillator stabilization time (Note 2) | After $V_{DD}$ reached Min. of oscillator voltage range | | | 4 | ms |
| Crystal resonator (Note 3) |  | Oscillator frequency ($f_{xx}$) (Note 1) | | 2.0 | 4.19 | 5.0 | MHz |
| | | Oscillator stabilization time (Note 2) | $V_{DD}$ = 4.5 to 6.0 V | | | 10 | ms |
| | | | | | | 30 | ms |
| Exteral clock |  | X1 input frequency ($f_x$) (Note 1) | | 2.0 | | 5.0 | MHz |
| | | X1 input high/low level width ($t_{XH}$, $t_{XL}$) | | 100 | | 250 | ns |

Note 1: Oscillator frequency and input frequency are oscillator characteristics only. For the instruction execution time, see the AC characteristics.

Note 2: Oscillator stabilization time is the time required for oscillation to stabilize after $V_{DD}$ is applied or the STOP mode is released.

Note 3: The resonators shown below are recommended.

**Subsystem Clock Oscillator Characteristics** ($T_a$ = –40 to +85°C, $V_{DD}$ = 2.7 to 6.0 V)

| Resonator | Recommended Constants | Item | Test Conditions | Min. | Typ | Max. | Unit |
|---|---|---|---|---|---|---|---|
| Crystal resonator |  | Oscillator frequency ($f_{XT}$) (Note 1) | | 32 | 32.768 | 35 | kHz |
| | | Oscillator stabilization (Note 2) | $V_{DD}$ = 4.5 to 6.0 V | | 1.0 | 2 | s |
| | | | | | | 10 | s |
| External clock |  | XT1 input frequency ($f_{XT}$) | | 32 | | 100 | kHz |
| | | XT1 input high/low level width ($t_{XTH}$, $t_{XTL}$) | | 10 | | 32 | µs |

Note 1: Oscillator frequency and input frequency are oscillator characteristics only. For the instruction execution time, see the AC characteristics.

Note 2: Oscillator stabilization time is the time required for oscillation to stabilize after $V_{DD}$ is applied.

**Capacitance** ($T_a$ = 25°C, $V_{DD}$ = 0V)

| | Item | Symbol | Test Conditions | Min. | Typ | Max. | Unit |
|---|---|---|---|---|---|---|---|
| Input capacitance | | $C_{IN}$ | | | | 15 | pF |
| Output capacitance | Other than display output | $C_{OUT}$ | f = 1 MHz<br>other than measured pin 0 V | | | 15 | pF |
| | Display output | | | | | 35 | pF |
| I/O capacitance | | $C_{IO}$ | | | | 15 | pF |

**Recommeded Oscillator Constants Main System Clock: Ceramic** ($T_a$ = –40 to +85°C)

| Manufacturer | Product Name | External Capacitance (pF) | | Oscillator Voltage Range (V) | | Remarks |
|---|---|---|---|---|---|---|
| | | C1 | C2 | Min. | Max. | |
| Murata Seisakusho | CSA 2.00MG<br>CSA 4.19MG<br>CSA 4.91MG | 30 | 30 | 4.0 | 6.0 | |
| | CAT 2.00MG<br>CST 4.19MG<br>CST 4.91MG | Unnecessary | Unnecessary | | | On-chip C type |
| Kyocera | KBR–2.0MS | 47 | 47 | 4.0 | 6.0 | |
| | KBR–4.0MS<br>KBR–4.19MS<br>KBR–4.91MS | 33 | 33 | | | |
| TDK | FCR 3.58M2<br>FCR 4.00M2<br>FCR 4.19M2 | 30 | 30 | 4.0 | 6.0 | |
| | FCR 4.19MC | Unnecessary | Unnecessary | | | On-chip C type |

**Main System Clock $X_{TAL}$** ($T_a$ = –40 to +85°C)

| Manufacturer | Frequency | Holder | Load Capacitance (pF) | External Capacitance (pF) | | Oscillator Voltage Range (V) | | Remarks |
|---|---|---|---|---|---|---|---|---|
| | | | | C1 | C2 | Min. | Max. | |
| Kinseki | 2.00<br>4.19<br>4.91 | HC–18/U<br>HC–49/U<br>HC–43/U | 16 | 20 | 20 | 4.0 | 6.0 | |

**Subsystem Clock: 32.768 kHz $X_{TAL}$** ($T_a$ = –10 to +60°C)

| Manufacturer | Type Disignation | Load Capacitance CL (pF) | External Circuit Constants | | | Oscillator Voltage Range | | Remarks |
|---|---|---|---|---|---|---|---|---|
| | | | C1 (pF) | C2 (pF) | R (kΩ) | Min. (V) | Max. (V) | |
| Kinseki | P–3 | 12 | 22 | 22 | 330 | 2.7 | 6.0 | |
| Citizen Watch | CFS–308 | 14 | 22 | 33 | 330 | 2.7 | 6.0 | |

Note: Fine adjust the oscillator frequency of crystal resonator at external capacitance C1.

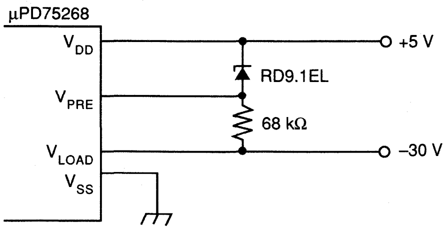
**DC Characteristics** ($T_a$ = –40 to +85°C, $V_{DD}$ = 2.7 to 6.0 V)

| Item | Symbol | Test Conditions | | Min. | Typ | Max. | Unit |
|---|---|---|---|---|---|---|---|
| High level input input voltage | $V_{IH1}$ | Other than below | | 0.7 $V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH2}$ | Port 0, 1, $\overline{\text{RESET}}$ | | 0.75 $V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH3}$ | X1, X2, XT1 | | $V_{DD}$ –0.4 | | $V_{DD}$ | V |
| | $V_{IH4}$ | Port 6 | $V_{DD}$ = 4.5 to 6.0 V | 0.65 $V_{DD}$ | | $V_{DD}$ | V |
| | | | | 0.7 $V_{DD}$ | | $V_{DD}$ | V |
| Low level input input voltage | $V_{IL1}$ | Other than below | | 0 | | 0.3 $V_{DD}$ | V |
| | $V_{IL2}$ | Port 0, 1, 6, $\overline{\text{RESET}}$ | | 0 | | 0.2 $V_{DD}$ | V |
| | $V_{IL3}$ | X1, X2, XT1 | | 0 | | 0.4 | V |
| High level output voltage | $V_{OH}$ | All output pins | $V_{DD}$ = 4.5 to 6.0 V, $I_{OH}$ = –1 mA | $V_{DD}$ –1.0 | | | V |
| | | | $I_{OH}$ = –100 µA | $V_{DD}$ –0.5 | | | V |
| Low level output voltage | $V_{OL}$ | Port 4, 5 | $V_{DD}$ = 4.5 to 6.0 V, $I_{OL}$ = 15 mA | | 0.4 | 2.0 | V |
| | | All output pins | $V_{DD}$ = 4.5 to 6.0 V, $I_{OL}$ = 1.6 mA | | | 0.4 | V |
| | | | $I_{OL}$ = 400 µA | | | 0.5 | V |
| High level input leakage current | $I_{LIH1}$ | Other than X1, X2, XT1 | $V_{IN}$ = $V_{DD}$ | | | 3 | µA |
| | $I_{LIH2}$ | X1, X2, XT1 | | | | 20 | µA |
| Low level input leakage current | $I_{LIL1}$ | Other than X1, X2, XT1 | $V_{IN}$ = 0 V | | | –3 | µA |
| | $I_{LIL2}$ | X1, X2, XT1 | | | | –20 | µA |

**DC Characteristics (Cont'd)** ($T_a$ = −40 to +85°C, $V_{DD}$ = 2.7 to 6.0 V)

| Item | Symbol | Test Conditions | | | Min. | Typ | Max. | Unit |
|---|---|---|---|---|---|---|---|---|
| High level output leakage current | $I_{LOH}$ | All output pins | $V_{OUT} = V_{DD}$ | | | | 3 | μA |
| Low level output leakage current | $I_{LOL1}$ | Other than display output | $V_{OUT}$ = 0V | | | | −3 | μA |
| | $I_{LOL2}$ | Display output | $V_{OUT} = V_{LOAD} = V_{DD}$ −35 V | | | | −10 | μA |
| Display output current | $I_{OD}$ | S0 to S9 | $V_{DD}$ = 4.5 to 6.0 V<br>$V_{OD} = V_{DD}$ −2 V | $V_{PRE}$ = (Note 1) $V_{DD}$−9 ±1 V | −3 | −5.5 | | mA |
| | | | | $V_{PRE}$ = 0V | −1.5 | −3.5 | | mA |
| | | T0 to T15 | | $V_{PRE}$ = (Note 1) $V_{DD}$−9 ±1 V | −15 | −22 | | mA |
| | | | | $V_{PRE}$ = 0 V | −7 | −15 | | mA |
| On-chip pull-down resistor (mask option) | $R_{P6}$ | Port 6 $V_{IN} = V_{DD}$ | $V_{DD}$ = 4.5 to 6.0 V | | 20 | 80 | 200 | kΩ |
| | | | | | 20 | | 1000 | kΩ |
| | $R_L$ | Display output | $V_{OD} - V_{LOAD}$ = 35 V | | 25 | 70 | 135 | kΩ |
| Power supply voltage (Note 2) | $I_{DD1}$ | 4.19 MHz crystal oscillation C1 = C2 = 15 pF | $V_{DD}$ = 5 V ± 10% (Note 3) | | | 3.0 | 9.0 | mA |
| | | | $V_{DD}$ = 3 V ± 10% (Note 4) | | | 0.55 | 1.5 | mA |
| | $I_{DD2}$ | | HALT mode | $V_{DD}$ = 5 V ± 10% | | 600 | 1800 | μA |
| | | | | $V_{DD}$ = 3 V ± 10% | | 200 | 600 | μA |
| | $I_{DD3}$ | 32 kHz (Note 5) crystal oscillation | $V_{DD}$ = 3 V ± 10% | | | 40 | 120 | μA |
| | $I_{DD4}$ | | HALT mode | $V_{DD}$ = 3 V ± 10% | | 5 | 15 | μA |
| | $I_{DD5}$ | XT1 = 0 V STOP mode | $V_{DD}$ = 5 V ± 10% | | | 0.1 | 20 | μA |
| | | | $V_{DD}$ = 3 V ± 10% | | | 0.1 | 10 | μA |

Note 1: The external circuit shown below is recommended.



RD9.1EL: Zener Diode (NEC)
Zener Voltage = 8.29 to 9.30 V

Note 2: Current flowing in on-chip pull-down resistors is not included.

Note 3: When 0011 set in processor clock control register (PCC) and operated in high-speed mode.

Note 4: When 0000 set in PCC and operated in low-speed mode.

Note 5: When 1001 set in system clock control register (SCC) and main system clock stopped and system operated by subsystem clock.

**AC Characteristics** ($T_a$ = −40 to +85°C, $V_{DD}$ = 2.7 to 6.0 V)

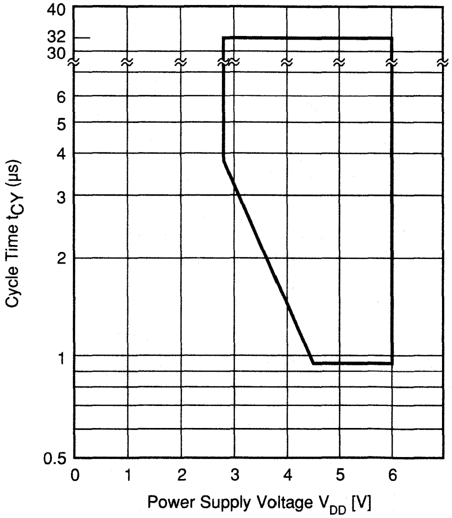
| Item | Symbol | Test Conditions | | Min. | Typ | Max. | Unit |
|---|---|---|---|---|---|---|---|
| Cycle time (minimum instruction execution time) (Note 1) | $t_{CY}$ | Operated by main system clock | $V_{DD}$ = 4.5 to 6.0 V | 0.95 | | 32 | µs |
| | | | | 3.8 | | 32 | µs |
| | | Operated by subsystem clock | | 114 | 122 | 125 | µs |
| TI0 input frequency | $f_{TI}$ | $V_{DD}$ = 4.5 to 6.0 V | | 0 | | 0.6 | MHz |
| | | | | 0 | | 165 | kHz |
| TI0 input high/low level width | $t_{TIH}$, $t_{TIL}$ | $V_{DD}$ = 4.5 to 6.0 V | | 0.83 | | | µs |
| | | | | 3 | | | µs |
| $\overline{SCK}$ cycle time | $t_{KCY}$ | $V_{DD}$ = 4.5 to 6.0 V | input | 0.8 | | | µs |
| | | | output | 0.95 | | | µs |
| | | | input | 3.2 | | | µs |
| | | | output | 3.8 | | | µs |
| $\overline{SCK}$ high/low level width | $t_{KH}$, $t_{KL}$ | $V_{DD}$ = 4.5 to 6.0 V | input | 0.4 | | | µs |
| | | | output | $t_{KCY}$/2 − 50 | | | ns |
| | | | input | 1.6 | | | µs |
| | | | output | $t_{KCY}$/2 − 150 | | | ns |

**AC Characteristics (Cont'd)** ($T_a$ = –40 to +85°C, $V_{DD}$ = 2.7 to 6.0 V)

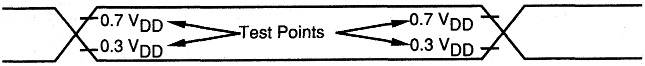
| Item | Symbol | Test Conditions | | Min. | Typ | Max. | Unit |
|---|---|---|---|---|---|---|---|
| SI setup time (for $\overline{SCK}$ ↑) | $t_{SIK}$ | | | 100 | | | ns |
| SI hold time (for $\overline{SCK}$ ↑) | $t_{KSI}$ | | | 400 | | | ns |
| $\overline{SCK}$ ↓ → SO output delay time | $t_{KSO}$ | $V_{DD}$ = 4.5 to 6.0 V | | | | 300 | ns |
| | | | | | | 1000 | ns |
| Interrupt input high/low level width | $t_{INTL}$, $t_{INTH}$ | | INT0 | (Note2) | | | μs |
| | | | INT1 | $2t_{CY}$ | | | μs |
| | | | INT2, 4 | 10 | | | μs |
| $\overline{RESET}$ low level width | $t_{RSL}$ | | | 10 | | | μs |

Note 1: The cycle time (minimum instruction execution time) is determined by the oscillator frequency of the connected resonator and the system clock control register (SCC) and processor clock control register (PCC).
The cycle time $t_{CY}$ versus power supply voltage $V_{DD}$ characteristics at main system clock operation is shown in the figure below.

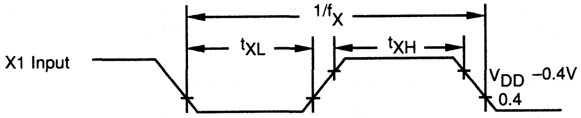
Note 2: Becomes $2t_{CY}$ or $128/f_{XX}$ by interrupt mode register (IM0) setting.



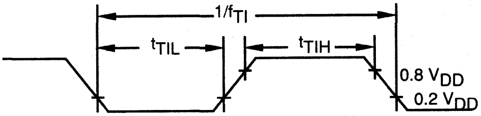
AC Timing Test Points (Except PORT0, 1, TI0, X1, X2, XT1, $\overline{\text{RESET}}$)



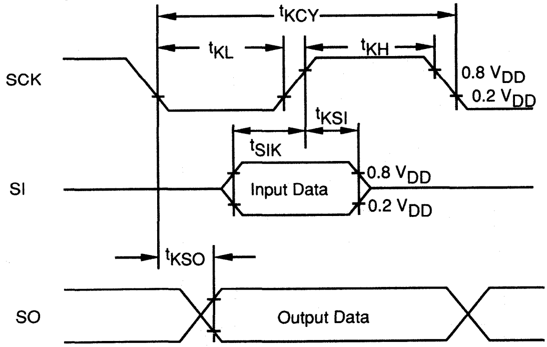
Clock Timing



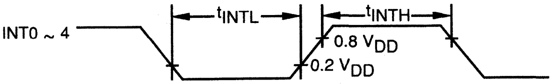
TI Timing



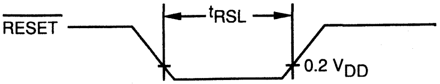
Serial Transfer Timing



Interrupt Input Timing



$\overline{\text{RESET}}$ Input Timing



**Data Memory STOP Mode Low Supply Voltage Data Retention Characteristics** (Ta = –40 to +85°C)

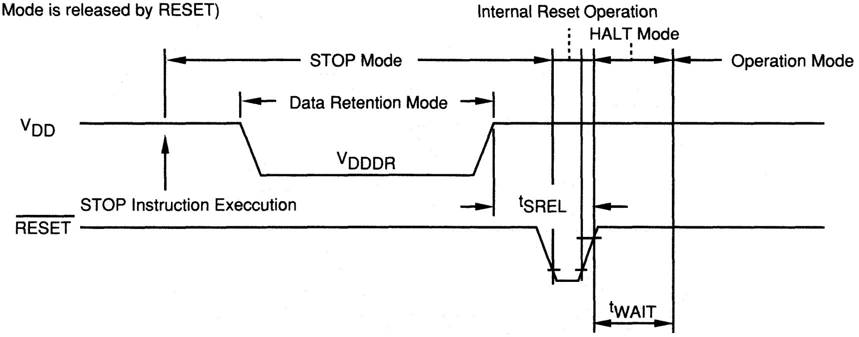
| Parameter | Symbol | Test Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Data Retention Supply Voltage | $V_{DDDR}$ | | 2.0 | | 6.0 | V |
| Data Retention Supply Current | $I_{DDDR}$ | $V_{DDDR}$ = 2.0V | | 0.1 | 10 | μA |
| Release Signal Set Time | $t_{SREL}$ | | 0 | | | μs |
| Oscillation Stable Wait Time (1) | $t_{WAIT}$ | When released by $\overline{RESET}$ | | $2^{17}/f_X$ | | ms |
| | | When released by Interrupt Request | | (2) | | ms |

Note 1: During Oscillation Stable Wait Time, CPU operation must be stopped to avoid unstable operation at oscillation start.

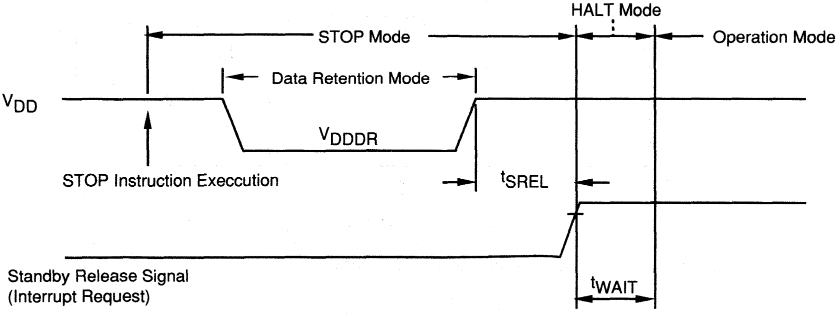
Note 2:

| BTM3 | BTM2 | BTM1 | BTM0 | tWAIT ( ): at $f_{XX}$=4.19MHz |
|---|---|---|---|---|
| – | 0 | 0 | 0 | $2^{20}/f_{XX}$ (approx. 250ms) |
| – | 0 | 1 | 1 | $2^{17}/f_{XX}$ (approx. 31.3ms) |
| – | 1 | 0 | 1 | $2^{15}/f_{XX}$ (approx. 7.82ms) |
| – | 1 | 1 | 1 | $2^{13}/f_{XX}$ (approx. 1.95ms) |

Data Retention Timing
(STOP Mode is released by $\overline{RESET}$)



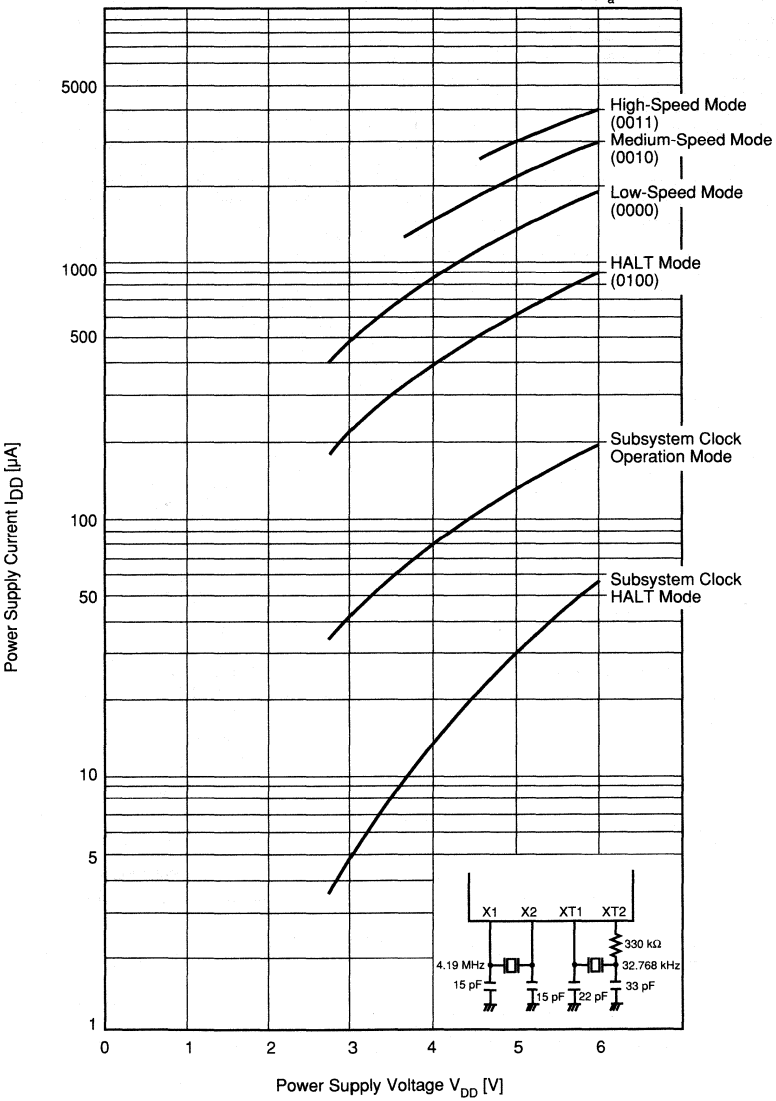
Data Retention Timing
(STOP Mode is released by Interrupt Request)



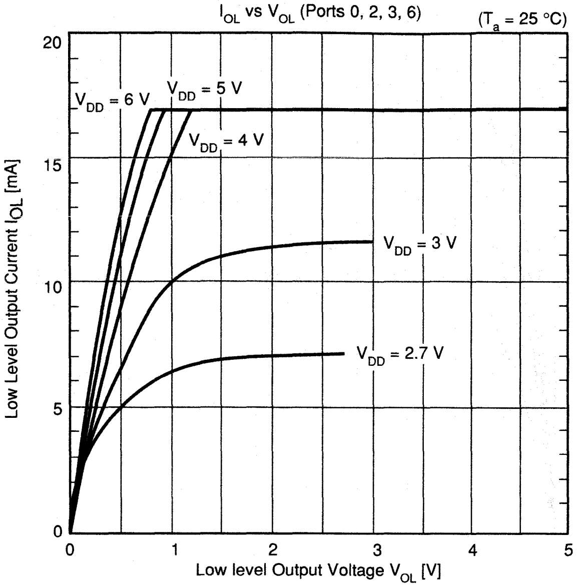
## CHARACTERISTICS CURVES

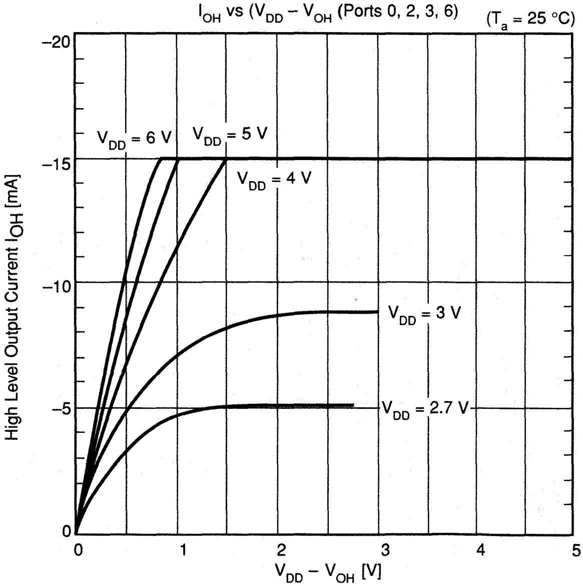
$I_{DD}$ vs $V_{DD}$

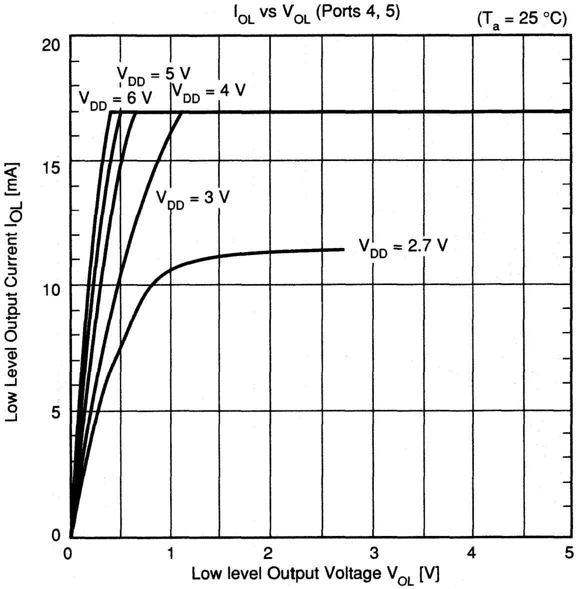
($T_a$ = 25 °C)



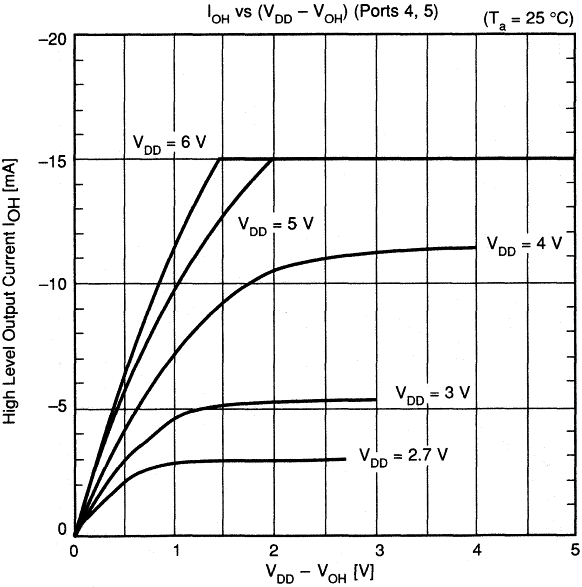
Remark: Values in ( ) are processor clock control register (PCC) values.


$I_{OL}$ vs $V_{OL}$ (Ports 0, 2, 3, 6)
($T_a$ = 25 °C)
$V_{DD}$ = 6 V
$V_{DD}$ = 5 V
$V_{DD}$ = 4 V
$V_{DD}$ = 3 V
$V_{DD}$ = 2.7 V
Low Level Output Current $I_{OL}$ [mA]
20
15
10
5
0
0
1
2
3
4
5
Low level Output Voltage $V_{OL}$ [V]

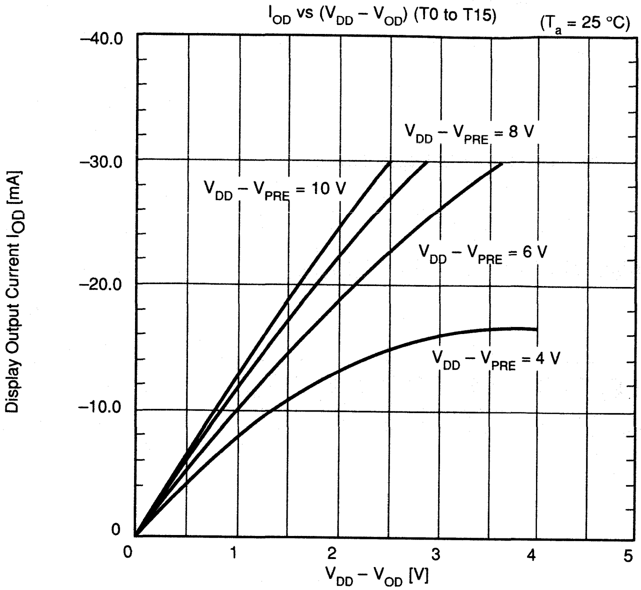

$I_{OH}$ vs ($V_{DD}$ – $V_{OH}$ (Ports 0, 2, 3, 6)
($T_a$ = 25 °C)
$V_{DD}$ = 6 V
$V_{DD}$ = 5 V
$V_{DD}$ = 4 V
$V_{DD}$ = 3 V
$V_{DD}$ = 2.7 V
High Level Output Current $I_{OH}$ [mA]
–20
–15
–10
–5
0
0
1
2
3
4
5
$V_{DD}$ – $V_{OH}$ [V]

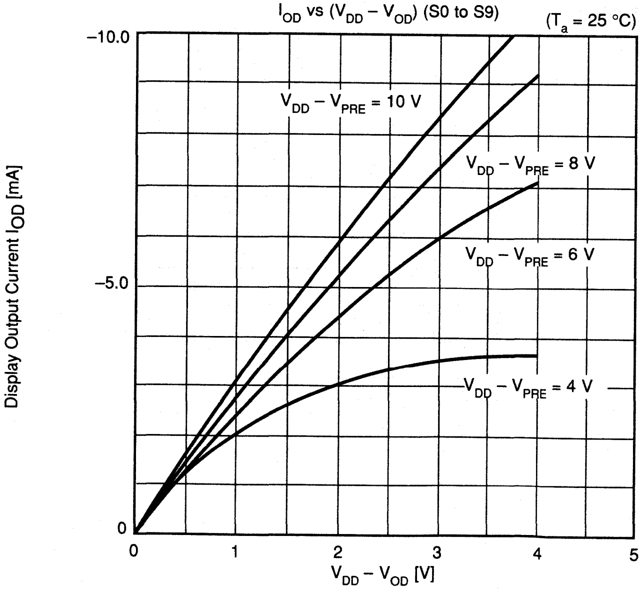


$I_{OL}$ vs $V_{OL}$ (Ports 4, 5)
($T_a$ = 25 °C)
$V_{DD}$ = 5 V
$V_{DD}$ = 6 V
$V_{DD}$ = 4 V
$V_{DD}$ = 3 V
$V_{DD}$ = 2.7 V
Low Level Output Current $I_{OL}$ [mA]
20
15
10
5
0
0
1
2
3
4
5
Low level Output Voltage $V_{OL}$ [V]



$I_{OH}$ vs ($V_{DD}$ − $V_{OH}$) (Ports 4, 5)
($T_a$ = 25 °C)
$V_{DD}$ = 6 V
$V_{DD}$ = 5 V
$V_{DD}$ = 4 V
$V_{DD}$ = 3 V
$V_{DD}$ = 2.7 V
High Level Output Current $I_{OH}$ [mA]
−20
−15
−10
−5
0
0
1
2
3
4
5
$V_{DD}$ − $V_{OH}$ [V]

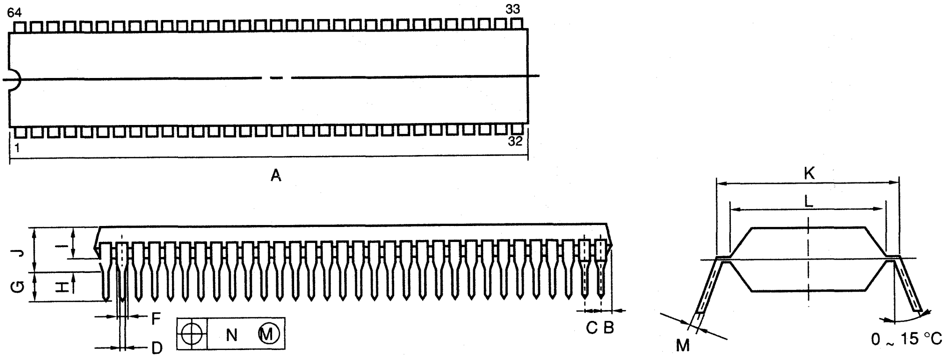

$I_{OD}$ vs ($V_{DD}$ – $V_{OD}$) (T0 to T15)
($T_a$ = 25 °C)
–40.0
–30.0
–20.0
–10.0
0
0
1
2
3
4
5
$V_{DD}$ – $V_{PRE}$ = 10 V
$V_{DD}$ – $V_{PRE}$ = 8 V
$V_{DD}$ – $V_{PRE}$ = 6 V
$V_{DD}$ – $V_{PRE}$ = 4 V
Display Output Current $I_{OD}$ [mA]
$V_{DD}$ – $V_{OD}$ [V]


$I_{OD}$ vs ($V_{DD}$ – $V_{OD}$) (S0 to S9)
($T_a$ = 25 °C)
–10.0
–5.0
0
0
1
2
3
4
5
$V_{DD}$ – $V_{PRE}$ = 10 V
$V_{DD}$ – $V_{PRE}$ = 8 V
$V_{DD}$ – $V_{PRE}$ = 6 V
$V_{DD}$ – $V_{PRE}$ = 4 V
Display Output Current $I_{OD}$ [mA]
$V_{DD}$ – $V_{OD}$ [V]

### 9.1 PACKAGE OUTLINES

**64 Pin Plastic Shrink DIP** (750 mil)

Top view



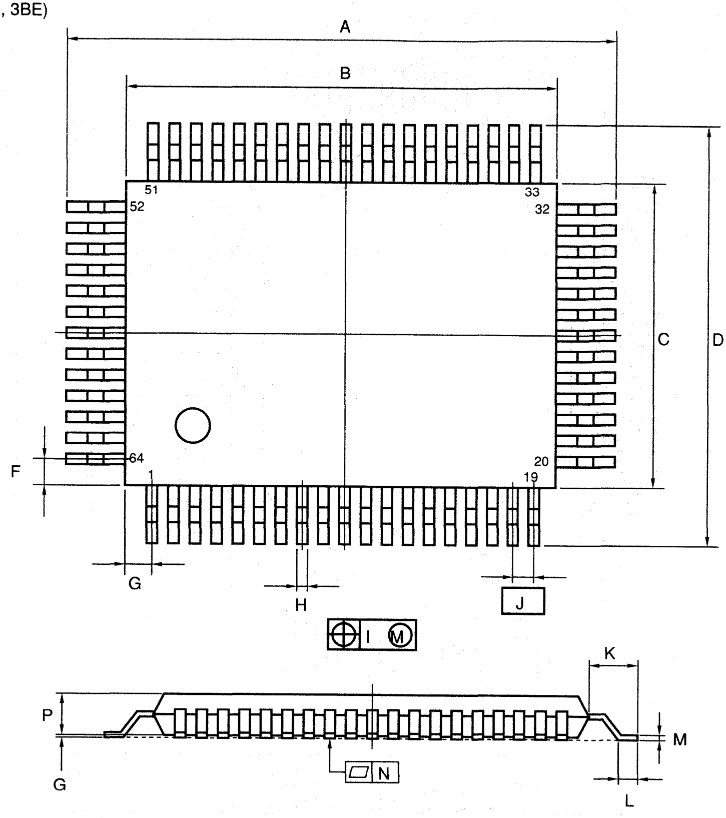
P64C-70-750A,C

Notes:
1) Each lead centerline is located within 0.17 mm (0.007 inch) of its true position (T. P.) at maximum material condition.
2) Item "K" to center of leads when formed parallel.

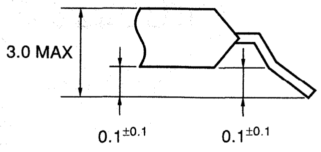
| ITEM | MILLIMETERS | INCHES |
|---|---|---|
| A | 58.68 MAX. | 2.311 MAX. |
| B | 1.78 MAX. | 0.070 MAX. |
| C | 1.778 (T.P.) | 0.070 (T.P.) |
| D | $0.50^{\pm0.10}$ | $0.020^{+0.004}_{-0.005}$ |
| F | 0.9 MIN. | 0.035 MIN. |
| G | $3.2^{\pm0.3}$ | $0.125^{\pm0.012}$ |
| H | 0.51 MIN. | 0.020 MIN. |
| I | 4.31 MAX. | 0.170 MAX. |
| J | 5.08 MAX. | 0.200 MAX. |
| K | 19.05 (T.P.) | 0.750 (T.P.) |
| L | 17.0 | 0.559 |
| M | $0.25^{+0.10}_{-0.03}$ | $0.010^{+0.004}_{-0.003}$ |
| N | 0.17 | 0.007 |

**64-pin Plastic QFP Package Dimensions** (Unit: mm)
(P64GF-100-3B8, 3BE)

Top view

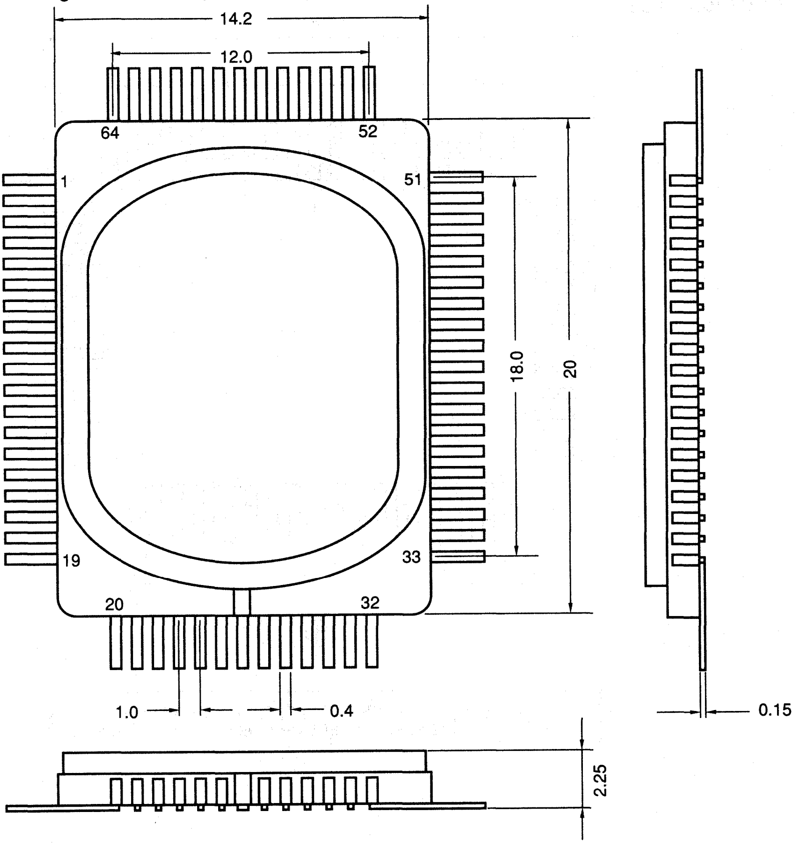


Pin Tip Shape Details

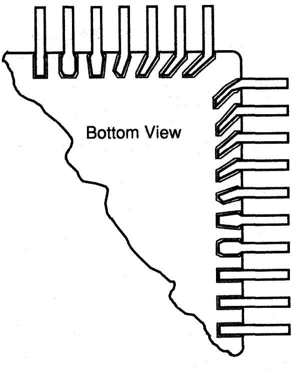


| ITEM | MILLIMETERS |
|---|---|
| A | $23.6^{\pm0.4}$ |
| B | $20^{\pm0.2}$ |
| C | $14^{\pm0.2}$ |
| D | $17.6^{\pm0.4}$ |
| F | 1.0 |
| G | 1.0 |
| H | $0.40^{\pm0.10}$ |
| I | 0.20 |
| J | 1.0 |
| K | $1.8^{\pm0.2}$ |
| L | $0.8^{\pm0.2}$ |
| M | $0.15^{+0.10}_{-0.06}$ |
| N | 0.15 |
| P | 2.7 |

**64-pin Ceramic Flat Package Outline for ES (Reference)** (Unit: mm)



Note: The metal cap of the device has $V_{SS}$ (GND Potential) level because of the metal cap is connected to pin No.26 (i.e. $V_{SS}$ pin).
The leads of the welding part at bottom of this device are formed in slant and have a chance of shorting the other lines of printed wiring boad.



## CHAPTER 3 μPD75304/306/308/312/316 4 BIT MICROCOMPUTER WITH L.C.D. CONTROLLER/DRIVER

### 1. Overview

The μPD75304, 75306, 75308, 75312, and 75316 are a product group in the series of μCOM-75X 4-bit single-chip microcomputers which contains an internal programmable LCD controller/driver and standard NEC serial bus interface, High-speed operation and high-advanced functions are more of their features.
Their Products are especially enhanced as the described below, in comparison with the μPD7514 which has contained the internal LCD controller/driver in μPD7500 series.

- ROM capacitance : Maximum 16256 words x 8 bits
- RAM capacitance: 512 x 4 bits
- General purpose register: Maximum 8 x 4 bits
- High-speed operation: Minimum instruction execution time = 1 us
- Eight interrupt sources and efficient interrupt service
- Efficient instruction system which can handle 1-, 4-, or 8-bit data by each data length
- Three-channel multifunction timer
- Very low power consumption for clock operation in standby mode (subsystem clock consumes very little power)

Application Fields

- Appliances: VTR's, Audio sets (CD player), etc.
- Others : Telephones, Cameras, Tonometers, etc.

Remarks: The manual explains the μPD75304, μPD75306, μPD75308, μPD75312, and μPD75316.
Unless otherwise noted, the μPD753XX is described as a typical product. Use the manual by replacing the μPD753XX with the μPD75304, μPD75306, μPD75308, μPD75312, or μPD75316, where appropiate.

#### 1.1 Features

- 41 systematic instructions
  - Versatile bit manipulation instructions
  - Efficient 4-bit data manipulation instructions
  - 8-bit data transfer instructions
  - GETI instructions convert any 2- or 3-byte instruction into a 1-byte instruction

- High-speed operation
  Minimum instruction execution time = 0.95 us ($f_X$ = 4.91 MHz, at 5 V operation)

- Variable instruction execution time for low voltage, low current operation
  - when main system clock is selected: 0.95 us, 1.91 us, 15.3 us/4.19 MHz
  - when subsystem clock is selected: 122 us/32.768 khz

- Program memory capacitance (ROM):
  - μPD75304 :4096 words x 8 bits (Mask ROM)
  - μPD75306 :6016 words x 8 bits (Mask ROM)
  - μPD75308 :8064 words x 8 bits (Mask ROM)
  - μPD75312 :12160 words x 8 bits (Mask ROM)
  - μPD75316 :16256 words x 8 bits (Mask ROM)

- Data memory capacity (RAM): 512 x 4 bits

- General purpose registers
  - Eight registers for 4-bit operations: X, A, B, C, D, E, H, L
  - Four pairs of registers for 8-bit operations: XA, BC, DE, HL

- Accumulators
  - One-bit accumulator (CY)
  - 4-bit accumulator (A)
  - 8-bit accumulator (XA)

- 68 I/O lines
  - 36 LCD drive output pins
  - 24 segment output dedicated pins
  - 8 dual function pins used for both segment output and CMOS output
  - 4 common output pins
  - 8 input pins
  - 8 middle-voltage N-channel open drain input/output pins (8 pins for driving an LED directly)
  - 16 CMOS input/output pins (4 pins for driving an LED directly)
  - Internal pull-up resistors for 31 I/O pins
  - 23 pins specified by using software
  - 8 pins specified by using mask option

- LCD controller/driver
  - Selection of the number of segments: 24, 28, or 32 (4 or 8 segments can be changed to bit output port.)
  - Display mode selection:
    Static
    1/2 duty (1/2 bias)
    1/3 duty (1/2 bias)
    1/3 duty (1/3 bias)
    1/4 duty (1/3 bias)
  - Split resistor for LCD drive voltage supply can be incorporated (mask option).

- 3-channel timers
  - 8-bit timer/event counter
    Four-stage clock source
    Event counting possible
  - 8-bit basic interval generation
    Reference time generation (1.95 ms, 7.82 ms, 31.3 ms, 250 ms/4.19 MHz)
    It is applicable to a watchdog timer.
  - Watch timer
    0.5 s time interval generation
    Count clock source: A change can be made between main and subsystem clocks
    Watch rapid feed mode (3.9 ms time interval generation)
    Buzzer output can be made. (2 kHz)

- 8-bit serial interface
  - It provides three modes:
    3-line serial I/O mode
    2-line serial I/O mode
    SBI mode
  - The serial transfer data top can be changed between the least and most significant bits (LSB and MSB).

- 16-bit bit sequential buffer (special bit manipulation memory):
  - Suitable for remote control applications.

- Clock output function
  - Timer/event counter output (PTO0):
    Outputs a square wave of any desired frequency
  - Clock output (PCL) : $\phi$, $f_x/2^3$, $f_x/2^4$, $f_x2^6$
  - Buzzer output (BUZ): 2 kHz (main system clock 4.19 MHz at operation/subsystem clock 32.7 kHz at operation)

- Vectored interrupt function
  - Three external vectored interrupts
    Both rising and falling edge detection interrupt (INT4)
    Detection edge programmable interrupt with noise elimination function (INT0)
    Detection edge programmable interrupt (INT1)
  - External rising edge detection/parallel port edge detection test input (INT2)
  - Three internal vectored interrupts:
    Timer/event counter 0 interrupt (INTT0)
    Basic interval timer interrupt (INTBT)

Serial interface interrupt (INTCSI)
- Watch test input (INTW)

- Two interal oscillators for system clock genertion
  - Ceramic or crystel oscillator for main system clock oscillation: 4.194304 MHz Standard
  - Crystal oscillator for subsystem clock oscillation: 32.768 kHz Standard

- Standby operation
  - STOP mode: Main system clock oscillation stops.
  - HALT mode: System clock oscillation continues. (CPU clock oscillation continues.)

- CMOS

### 1.2 Ordering Information

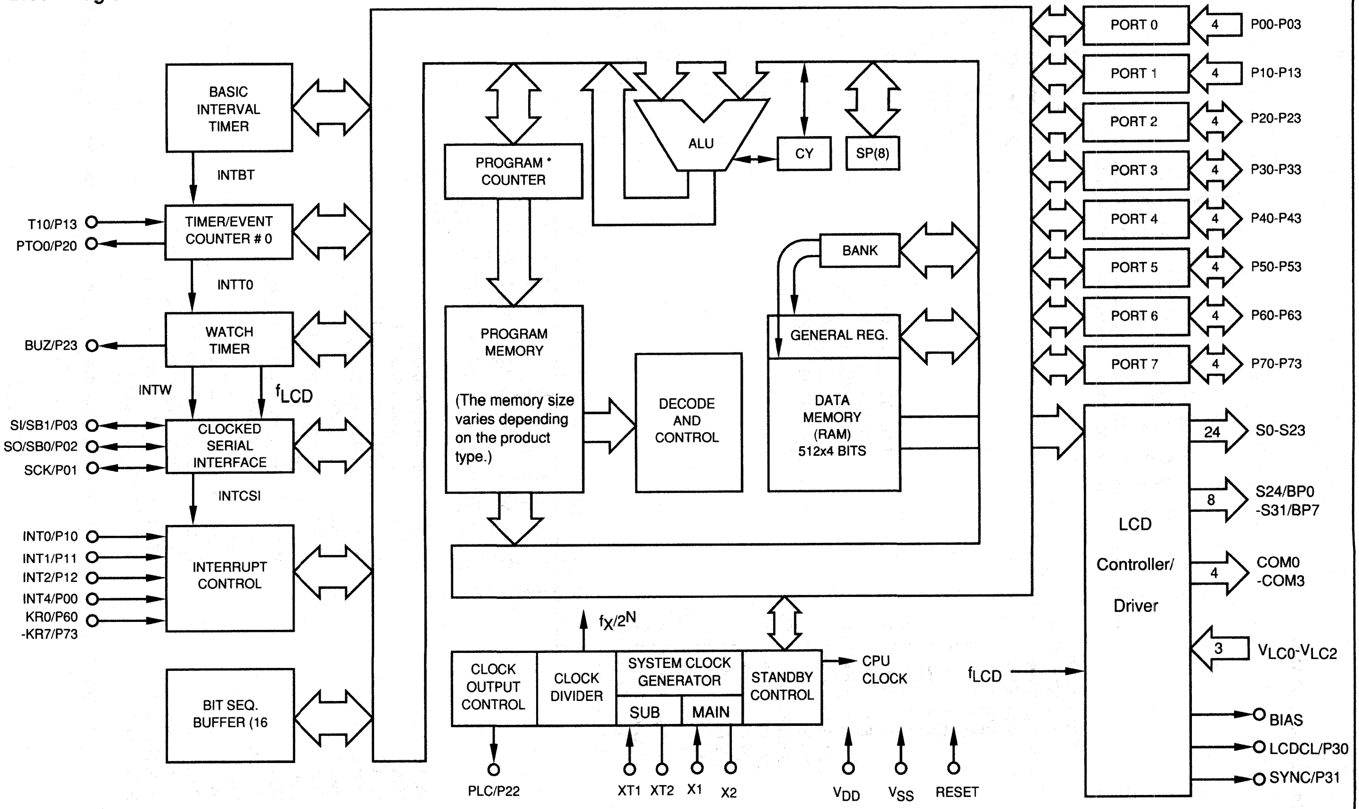
| Ordering code | Package | Internal ROM |
|---|---|---|
| μPD75304GF-xxx-3B9 | 80-pin plastic QFP (bent leads) | Mask ROM |
| μPD75306GF-xxx-3B9 | 80-pin plastic QFP (bent leads) | Mask ROM |
| μPD75308GF-xxx-3B9 | 80-pin plastic QFP (bent leads) | Mask ROM |
| μPD75312GF-xxx-3B9 | 80-pin plastic QFP (bent leads) | Mask ROM |
| μPD75316GF-xxx-3B9 | 80-pin Plastic QFP (bent leads) | Mask ROM |

Remark: xxx shows the code number of the mask ROM and option specification

### 1.3 List of Family Product Functions

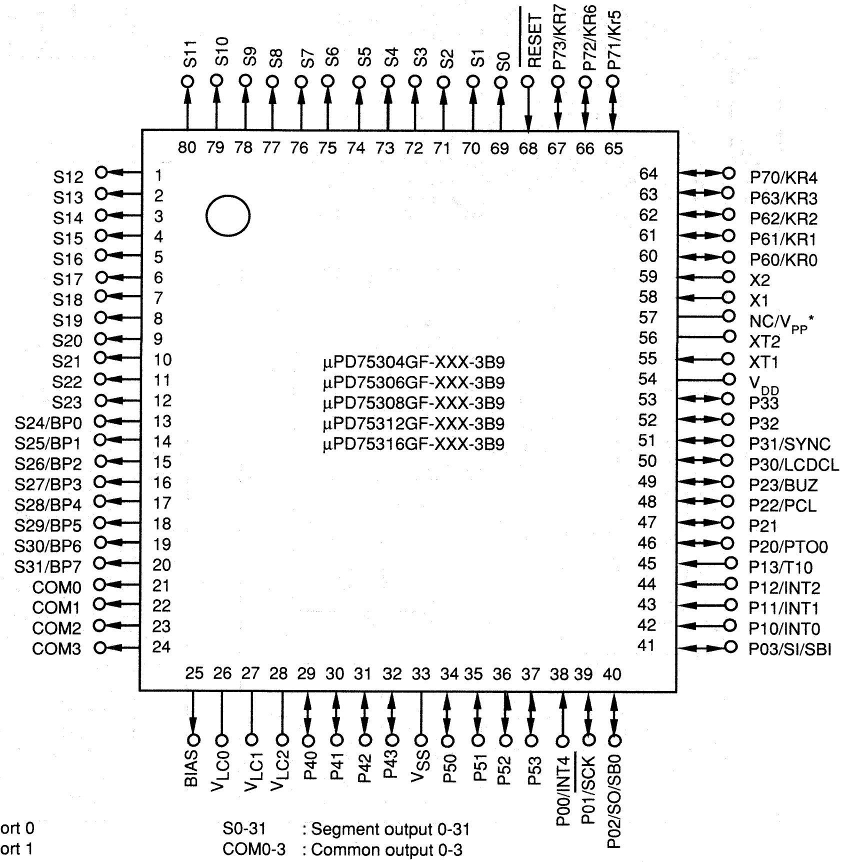
| Item | | µPΔ75304 | µPD75306 | µPD75308 | µPD75P308 | µPD75312 | µPD75316 | µPD75P316 |
|---|---|---|---|---|---|---|---|---|
| Program memory | | • Mask ROM<br>• 000H-FFFH<br>• 4096 x 8 bits | • Mask ROM<br>• 0000H-177FH<br>• 6016 x 8 bits | • Mask ROM<br>• 0000H-1F7FH<br>• 8064 x 8 bits | • EPROM/one-time PROM<br>• 0000H-1F7FH<br>• 8064 x 8 bits | • Mask ROM<br>• 0000H-2F7FHH<br>• 12160 x 8 bits | • Mask ROM<br>• 0000H-3F7FHH<br>• 16256 x 8 bits | • One-time PROM<br>• 0000H-3F7FH<br>• 16256 x 8 bits |
| Data memory | | 512 x 4 bits<br>Bank 0:256 x 4<br>Bank 1:256 x 4 | | | | | | |
| Instruction set | 3-byte branch instruction | Not included | Included | | | | | |
| | Others | Common to the products | | | | | | |
| Program counter | | 12 bits | 13 bits | | | 14 bits | | |
| Pull-up resistor (Mask option) | Ports 4 and 5 | Mask option | | | Not contained | Mask option | | Not contained |
| Split resistor for LCD drive power supply | | Mask option | | | Not included | Mask option | | Not included |
| Pin connection | | $V_{PP}$, PROM program pins not included | | | $V_{PP}$, PROM program pins included | $V_{PP}$, PROM program pins not included | | $V_{PP}$, PROM program pins included |
| $V_{DD}$ range | | 2.7-6.0 V | | | 5V±5% | 2.7-6.0 V | | 5V±5% |
| Package | | 80-pin plastic QFP | | | • 80-pin ceramic LCC (with window)<br>• 80-pin plastic QFP | 80-pin plastic QFP | | |

## 1.4 Block Diagram



*: The bit length varies depending on the product type

### 1.5 Pin Connection (Top view)



| | | |
|---|---|---|
| P00-03 | : | Port 0 |
| P10-13 | : | Port 1 |
| P20-23 | : | Port 2 |
| P30-33 | : | Port 3 |
| P40-43 | : | Port 4 |
| P50-53 | : | Port 5 |
| P60-63 | : | Port 6 |
| P70-73 | : | Port 7 |
| BP0-7 | : | Bit port |
| KR0-7 | : | Key return |
| SCK | : | Serial clock |
| SI | : | Serial input |
| SO | : | Serial output |
| SB0, 1 | : | Serial bus 0, 1 |
| RESET | : | Reset input |

| | |
|---|---|
| S0-31 | : Segment output 0-31 |
| COM0-3 | : Common output 0-3 |
| $V_{LC0-2}$ | : LCD power supply 0-2 |
| BIAS | : LCD power supply bias control |
| LCDCL | : LCD clock |
| SYNC | : LCD synchronization |
| TI0 | : Timer input 0 |
| PTO0 | : Programmable timer output 0 |
| BUZ | : Buzzer clock |
| PCL | : Programmable clock |
| INT0, 1, 4 | : External vectored interrupt 0, 1, 4 |
| INT2 | : External test input 2 |
| X1, 2 | : Main system clock oscillation 1, 2 |
| XT1, 2 | : Subsystem clock oscillation 1, 2 |
| NC/$V_{PP}$* | : No connection |

* When using OTP µPD75P3XX in the same socket, please note that this is $V_{PP}$ and has to be connected to $V_{DD}$ in operating mode.

## 2. PIN FUNCTIONS

### 2.1 Input and output ports

2.1.1 P00-P03 (port 0) - Input Pins also Used for INT4, $\overline{\text{SCK}}$, SO/SB0, SI/SB1
P10-P13 (port 1) - Input Pins also Used for INT0, INT1, INT2, and TI0

P00-P03 and P10-P13 are pins for 4-bit input port (port 0 and 1).

(1) The P00-P03 pins are also used for the INT4 pin (vectored interrupt input) and the $\overline{\text{SCK}}$, SO/SB0, and SI/SB1 pin (serial interface output).

(2) The P10-P13 pins are also used for the INT0 and INT1 pins (vectored interrupt input), the INT2 pin (edge detection test input) and the TI0 pin (external event pulse to the timer/event counter).

The state of each pin P00-P03 and P10-P13 can always be input regardless of the condition of any of the pins INT4, $\overline{\text{SCK}}$, SO/SB0, SI/SB1, INT0, INT1, INT2, and TI0.

To prevent noise from causing a malfunction, a Schmitt trigger input is applied to port 0 P00/INT4, P01/$\overline{\text{SCK}}$, P02/SO/SB0, P03/SI/SB1 input and port 1 pins. In addition, P10 is provided with a noise removal circuit. (For details, see 6.3 (2).)

An internal pull-up resistor can be specified for port 0 in 3-bit units (P01-P03) and for port 1 in 4-bit units (P10-P13) by using the software-operated pull-up resistor specification register group A.

When the $\overline{\text{RESET}}$ signal is generated, every pin is placed in input port mode.

2.1.2 P20-P23 (port 2) - Input/Output Pins also Used for PTO0, PCL, and BUZ
P30-P33 (port 3) - Input/Output Pins also Used for LCDCL and SYNC.
P40-P43 (port 4) - and P50-53 (port 5) - N-Channel Open Drain Medium-Voltage (10 V)
P60-P63 (port 6), P70-73 (port 7) - 3-State Input/Output

P20-P23, P30-P33, P40-P43, P50-P53, P60-P63, and P70-P73 are 4-bit input/output ports with output latches.

(1) The P20-P23 pins are also used for the PTO0 pin timer/event counter output), the PCL pin (programmable clock output), and the BUZ pin (fixed frequency output).

(2) The P30-P33 pins are also used for the LCDCL pin (LCD external extended driver driving clock) and the SYNC pin (LCD) external extended driver synchronizing clock).

(3) The P60-P63 and P70-P73 pins are also used for the KR0-KR3 and KR4-KR7 pins (key interrupt input).

Port 3 has high-current output, enabling an LED to be driven directly. Ports 4 and 5 outputs are N-channel open drain medium-voltage (10 V) and high-current, enabling an LED to be driven directly.

The input/output mode of each port is selected by using the port mode register. The mode of port 2, 4, 5, or 7 can be specified in 4-bit units, and that of port 3 or 6 bitwise.

An internal pull-up resistor can be specified for ports 2, 3, 6, and 7 in 4-bit units by using software to operate the pullup resistor specification register (POGA). It can be specified for ports 4 and 5 bitwise by using the mask option.

Ports 4 and 5 (6 and 7) can be paired for input/output in 8-bit units. When the $\overline{\text{RESET}}$ signal is generated, ports 2, 3, 6, and 7 are placed in input mode (output high impendance) and ports 4 and 5 are set to a high level (when internal pull-up register is contained) or to high impendance.

2.1.3 BP0-BP7 - Output Pins also Used for S24-S31 (LCD Controller/Driver Segment Signal Output)

BP0-BP7 are bit port 0-7 output pins (each port is a 1-bit output port with output latch). The BP0-BP7 pins are also used for the S24-S31 pins (LCD controller/driver segment output signal output pins).

2.1.4 TI0 - Input Pin also Used for Port 1

TI0 is an external event pulse input pin for the programmable timer/event counter.
The pin is Schmitt trigger input.

2.1.5 PTO0 - Output Pin also Used for Port 2

PTO0 is a programmable timer/event counter output pin which outputs square wave pulses. To output a programmable timer/event counter signal, the P20 output latch is cleared and the port 2 bit of the port mode register is set to 1 (output mode).
Output is cleared by execution of the timer start instruction.

2.1.6 PCL - Output Pin also Used for Port 2

PCL is a programmable clock output pin used to supply clocks to peripheral LSI devices such as a slave microcomputer and an A/D converter. When the $\overline{\text{RESET}}$ signal is generated, the clock mode register (CLOM) is cleared, clock output is inhibited, and the normal port operation mode is set.

2.1.7 BUZ - Output Pin also Used for Port 2

BUZ is a fixed frequency output pin. Fixed frequency (2.048 kHz) output is used for buzzer sounding and system clock oscillation frequency trimming. The BUZ pin, also used for the P23 pin, is validated only when bit 7 (WM7) of the watch mode register (WM) is set to 1.

When the RESET signal is generated, WM7 is cleared and the normal port operation mode is set.

2.1.8 SCK, SO/SB0, and SI/SB1 - 3-State Input/Output Pins also Used for Port 0

SCK, SO/SB0, and SI/SB1 are input/output pins for serial interface and operate according to how the serial operation mode register (CSIM) is set.

When the RESET signal is generated, serial interface operation is stopped and the pins are used for port 0 (input port).

Every pin is Schmitt trigger input.

2.1.9 INT4 - Input Pin also Used for Port 0.

INT4 is an external vectored interrupt input pin (both rising and falling edged active). When the signal input to the pin changes from low to high state or from high to low state, the interrupt request flag is set.

INT4 is for asynchronous input. When a signal having a given high or low level duration is input, it is acknowledged independently of the CPU operation clock.

INT4 can also be used to release the STOP or HALT mode. It is Schmitt trigger input.

2.1.10 INT0 and INT1 - Input Pins also Used for Port 1

INT0 and INT1 are edge detection vectored interrupt input pins. INT0 has the noise removal function. Detected edge selection can be made using edge detection mode registers (IM0 and IM1).

(1) INT0 (IM0 bits 0 and 1)
(a) Rising edge active
(b) Falling edge active
(c) Both rising and falling edges active
(d) External interrupt signal input inhibited

(2) INT1 (IM1 bit 0)
(a) Rising edge active
(b) Falling edge active

INT0 has the noise removal function; sampling clocks for noise removal can be changed two stages. The acknowledged signal width varies depending on CPU clock operation.

INT1 is for asynchronous input. When a signal having a given high level duration is input, it is acknowledged independently of the CPU clock operation.

When the RESET signal is generated, IM0 and IM1 are cleared and rising edge active is selected.

INT0 and INT1 are Schmitt trigger inputs.

2.1.11 INT2 - Input Pin also Used for Port 1

INT2 is an external test input pin (both rising and falling edges active). When INT2 is selected by using the edge detection mode register (IM2) and the signal input to the INT2 pin transits from low to high state, the internal test flag (IRQ2) is set.

INT2 is for asynchronous input. When a signal having a given high level duration is input, it is acknowledged independently of the CPU clock operation.

When the RESET signal is generated, IM2 is cleared and the test flag (IRQ2) is set by inputting the rising edge to the INT2 pin.

2.1.12 KR0-KR3 - Input Pins also Used for Port 6
KR4-KR7 - Input Pins also Used for Port 7

KR0-KR7 are key interrupt (parallel falling edge detection interrupt) input pins. The interrupt format can be specified by setting the edge detection mode register (IM2).

When the RESET signal is generated, the pins are placed in port 6 and 7 input mode.

2.1.13 S0-S23 - Output
S24-S31 - Output Pins also Used for Bit Ports 0-7

S0-S31 are segment signal output pins for directly driving LCD segments (front electrodes). Static, 2 or 3 time division (1/2 bias law), or 3 or 4 time division (1/3 bias law) driving is performed.

S0-S23 are segment-only output pins. S24-S31 are also used for the bit port 0-7 output; which mode the pins are used for, is specified by using the dislay mode register (LCDM).

2.1.14 COM0-COM3 - Output

COM0-COM3 are common signal output pins for directly driving the LCD common pins (rear electrodes). Common signals are output when static (COM0, COM1, COM2, COM3 output), while driving by 1/2 bias law 2 time division (COM0, COM1 output), while driving by 3 time division (COM0, COM1, COM2 output), while driving by 1/3 bias law 3 time division (COM0, COM1, COM2, COM3 output) is performed.

2.1.15 $V_{LC0}$-$V_{LC2}$

$V_{LC0}$-$V_{LC2}$ are power supply pins to drive the LCD. The μPD753XX allows split-resistor incorporation in the $V_{LC0}$-$V_{LC2}$ pins so that LCD drive power can be supplied according to the bias law without using an external split resistor. (Mask option)

2.1.16 BIAS

BIAS is an LCD power supply bias control pin. To deal with various LCD drive voltages, the BIAS pin is connected to the $V_{LC0}$ pin to change the resistor split ratio. By connecting an external resistor, the BIAS pin can be used together with the $V_{LC0}$-$V_{LC2}$ and $V_{SS}$ pins for fine adjustment of the LCD drive supply voltage.

2.1.17 LCDCL

LCDCL is a clock output pin for driving an external LCD extension driver.

2.1.18 SYNC

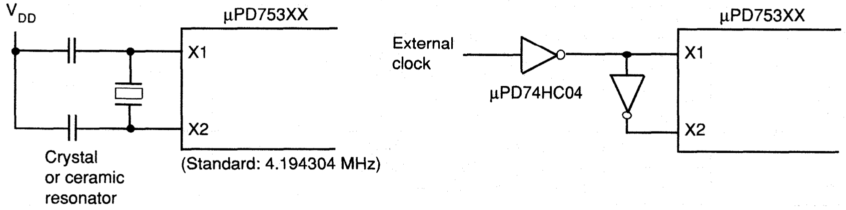
SYNC is a clock output pin for synchronizing an external LCD extension driver.

2.1.19 X1 and X2

X1 and X2 are connection pins for the main system clock crystal / ceramic oscillator.
External clocks can also be input.

(a) Crystal ceramic oscillator

(b) External clock



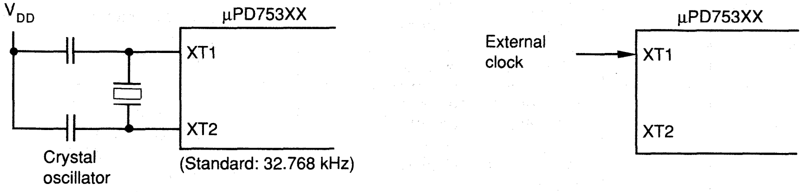
(Standard: 4.194304 MHz)

2.1.20 XT1 and XT2

XT1 and XT2 are crystal connection pins for the subsystem clock oscillation.
External clocks can also be input.

(a) Crystal oscillator

(b) External clock



(Standard: 32.768 kHz)

2.1.21 RESET

RESET is an active low reset input pin.
RESET is for asynchronous input. When a signal having a given low level duration is input independently of the operation clock, the RESET signal is generated and the system is reset overriding all other operations.
It is used for normal CPU initialization / start-up and also to release the standby (STOP or HALT) mode.
RESET is a Schnitt trigger input.

2.1.22 $V_{DD}$

$V_{DD}$ is a positive power supply pin.

2.1.23 $V_{SS}$

$V_{SS}$ is a ground potential.

## 2.2 Pin Function List

(1) Normal operation mode

**Table 2.2–1 Digital Input / Output Port Pin Function List**

| Pin name | I/O | Also used for: | Function | 8-bit I/O | When reset | I/O circuit type (Note 1) |
|---|---|---|---|---|---|---|
| P00 | I | INT4 | 4-bit input port (port 0). Internal pull-up resistors can be specified for P01-P03 in 3-bit units by using software | x | Input | Ⓑ |
| P01 | I/O | $\overline{SCK}$ | | | | Ⓕ- A |
| P02 | I/O | SO/SB0 | | | | Ⓕ- B |
| P03 | I/O | SI/SB1 | | | | Ⓜ- C |
| P10 | I | INT0 | With noise removal function | x | Input | Ⓑ- C |
| P11 | | INT1 | 4-bit input port (port 1). Internal pull-up resistors can be specified for P10-P13 in 4-bit units by using software | | | |
| P12 | | INT2 | | | | |
| P13 | | TI0 | | | | |
| P20 | I/O | PTO0 | 4-bit input/output port (port 2). Internal pull-up resistor can be specified for P20-P23 in 4-bit units by using software. | x | Input | E – B |
| P21 | | – | | | | |
| P22 | | PCL | | | | |
| P23 | | BUZ | | | | |
| P30 | I/O | LCDCL | Programmable 4-bit input/output port (port 3). Input or output mode can be selected bitwise.<br>Internal pull-up resistor can be specified for P30-P33 in 4-bit units by using software. (Note 2) | x | Input | E – B |
| P31 | | SYNC | | | | |
| P32 | | – | | | | |
| P33 | | – | | | | |
| P40-P43 (Note 2) | I/O | – | N-channel open drain 4-bit input/output port (port 4). Pull-up resistor can be incorporated bitwise (mask option). 10 Volts during open drain. | o | High level (when pull-up resistor is incorporated) or high impendance | M |
| P50-P53 (Note 2) | I/O | – | N-channel open drain 4-bit input/output port (port 4). Pull-up resistor can be incorporated bitwise (mask option). 10 Volts during open drain. | | High level (when pull-up resistor is incorporated) or high impendance | M |

(to be continued)

**Table 2.2–1 Digital Input / Output Port Pin Function List (cont'd)**

| Pin name | I/O | Also used for | Function | 8-bit I/O | When reset | I/O circuit type (Note 1) |
|---|---|---|---|---|---|---|
| P60 | I/O | KR0 | Programmable 4-bit input/output port (port 6). Input or output mode can be selected bitwise.<br>Internal pull-up resistor can be specified for P60-P63 in 4-bit units by using software. | o | Input | Ⓕ- A |
| P61 | | KR1 | | | | |
| P62 | | KR2 | | | | |
| P63 | | KR3 | | | | |
| P70 | I/O | KR4 | 4-bit input/output port (port 7). Internal pull-up resistor can be specified for P70-P73 in 4-bit units by using software. | | Input | Ⓕ- A |
| P71 | | KR5 | | | | |
| P72 | | KR6 | | | | |
| P73 | | KR7 | | | | |
| BP0 | O | S24 | 1-bit output ports (bit ports). x The pins are also used for the segment output pins. | x | Note 3 | G – C |
| BP1 | | S25 | | | | |
| BP2 | | S26 | | | | |
| BP3 | | S27 | | | | |
| BP4 | O | S28 | | | | |
| BP5 | | S29 | | | | |
| BP6 | | S30 | | | | |
| BP7 | | S31 | | | | |

Note 1: The ○ mark denotes Schmitt trigger input
Note 2: LED can be driven directly
Note 3: BP0-7 select $V_{LC1}$ as the input source.
The output level is changed by the external circuit of BP0-7 and $V_{LC1}$.

Example: Since BP0-BP7 are connected to each other through the µPD753XX as illustrated below, the BP0-BP7 output level is determined by the resistance value of R1, R2, R3.

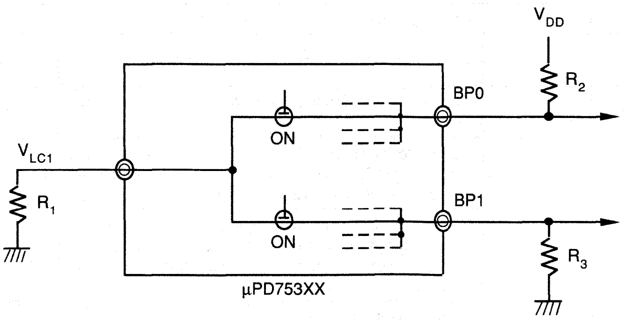


**Table 2.2–2 Pin Function List other than Port Pins**

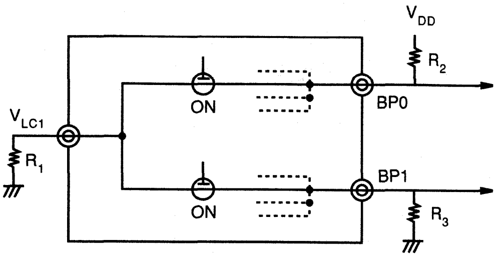
| Pin name | I/O | Also used for: | Function | When reset | I/O circuit type (Note 1) |
|---|---|---|---|---|---|
| TI0 | I | P13 | External event pulse input pin to timer / event counter. | | Ⓑ– C |
| PT00 | I/O | P20 | Timer / event counter output pin. | Input | E – B |
| PCL | I/O | P22 | Clock output pin. | Input | E – B |
| BUZ | I/O | P23 | Fixed frequency output pin (for buzzer or system clock trimming). | Input | E – B |
| $\overline{\text{SCK}}$ | I/O | P01 | Serial clock input / output pin. | Input | Ⓕ– A |
| SO/SB0 | I/O | P02 | Serial data output pin.<br>Serial bus input / output pin. | Input | Ⓕ– B |
| SI/SB1 | I/O | P03 | Serial data input pin.<br>Serial bus input / output pin. | Input | Ⓜ– C |
| INT4 | I | P00 | Edge detection vectored interrupt input pin (detection of both rising and falling edges is active). | | Ⓑ |
| INT0 | I | P10 | Edge detection vectored interrupt input pin (detected edge can be selected).<br>Synchronous clock | | Ⓑ– C |
| INT1 | | P11 | Asynchronous | | |
| INT2 | I | P12 | Edge detection testable input pin asynchronous (rising edge is detected). | | Ⓑ– C |
| KR0-KR3 | I/O | P60-P63 | Parallel falling edge detection testable input pins. | Input | Ⓕ– A |
| KR4-KR7 | I/O | P70-P73 | Parallel falling edge detection testable input pins. | Input | Ⓕ– A |
| S0-S23 | O | – | Segment signal output pins. | Note 4 | G – A |
| S24-S31 | O | BP0-7 | Segment signal output pins. | Note 4 | G – C |
| COM0-COM3 | O | – | Common signal output pins. | Note 4 | G – B |
| $V_{LC0}$-$V_{LC2}$ | – | – | LCD drive power supply pins. Internal split resistor can be used (mask option). | – | – |
| BIAS | O | – | Output pin to cut external split resistor. | Note 5 | |
| LCDCL (Note 3) | I/O | P30 | Clock output pin for external extension driver. | Input | E – B |
| SYNC (Note 3) | I/O | P31 | Clock output pin for synchronizing external extension driver. | Input | E – B |
| X1, X2 | | – | Connection pins for main system clock crystal/ceramic oscillator. When external clock is used, it is input to X1 and its opposite phase is input to X2. | – | – |
| XT1 | I | – | Subsystem clock oscillation crystal connection pins. When external clock is used, it is input to XT1, and XT2 is not connected. XT1 can be used for 1-bit input (test) pin. | – | – |
| XT2 | – | | | – | – |
| $\overline{\text{RESET}}$ | I | | System reset input pin. | – | Ⓑ |
| NC (Note 2) | – | | No connection | – | – |
| $V_{DD}$ | – | | Positive power supply pin. | – | – |
| $V_{SS}$ | – | | Ground potential pin. | – | – |

Notes:

1. The○ mark denotes Schmitt trigger input.
2. Connected the NC pin to $V_{DD}$, when μPD75P308 and printed board are also shared with.
3. The pins are provided for future system expansion; at present they are used only for the P30 and P31 pins
4. For each display output, the following $V_{LCX}$ are selected as input sources:

| | |
|---|---|
| S0-31 | : $V_{LC1}$ |
| COM0-COM2 | : $V_{LC2}$ |
| COM3 | : $V_{LC0}$ |

However, the display output level varies depending on the display output and $V_{LCX}$ external circuit.
Example:
Since BP0-BP7 are connected to each other through the μPD753XX as illustrated below, the BP0-BP7 output level is determined by the resistance value of R1, R2, R3.



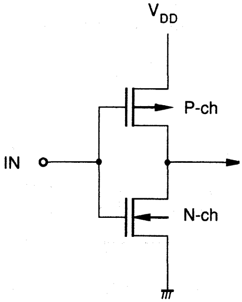
μPD753XX

5. When internal split resistor is contained : Low level
When internal split resistor is not contained: High impedance

**2.3 Pin Input / Output Circuits**

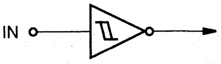
The μPD753XX pin input / output circuits are shown in schematic drawings.

(1) Type A (for Type E – B)



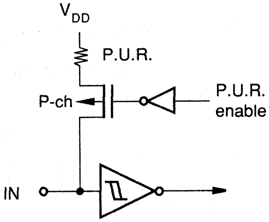
CMOS standard input buffer

(2) Type B



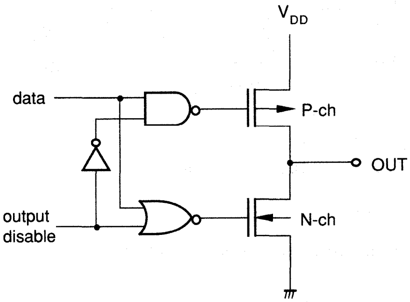
Schmitt trigger input with hysteresis characteristic

(3) Type B – C



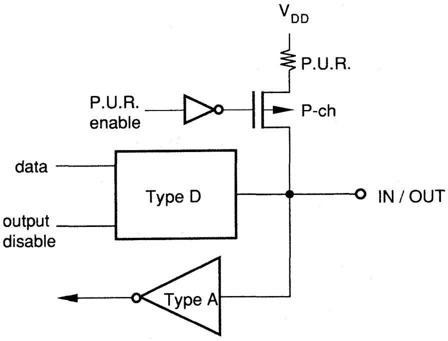
P.U.R.: Pull-Up Resistor

(4) Type D (for Type E – B, F – A)



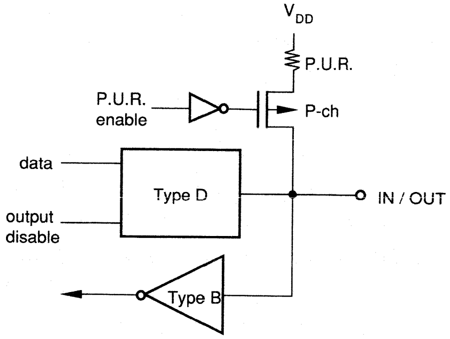
Push-pull output where output can be placed in high impedance (both P and N channels are turned off).

(5) Type E - B



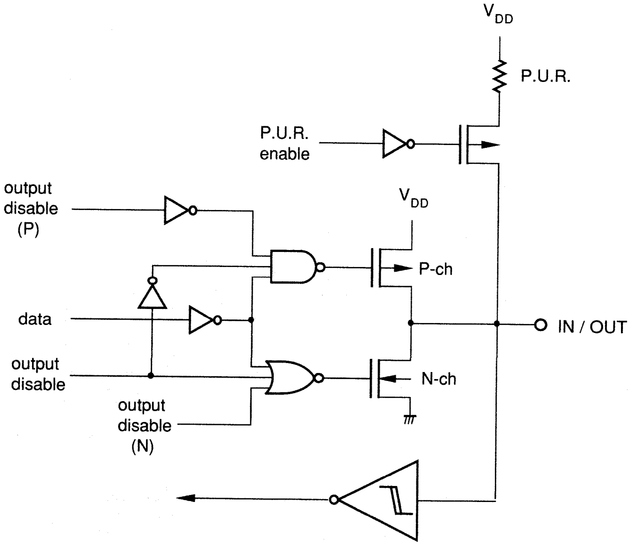
P.U.R.: Pull-Up Resistor

(6) Type F – A



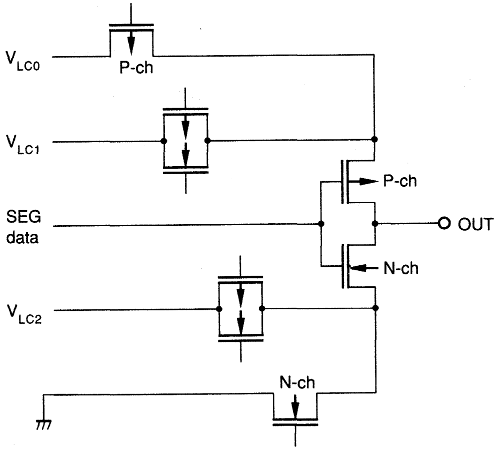
P.U.R.: Pull-Up Resistor

(7) Type F – B

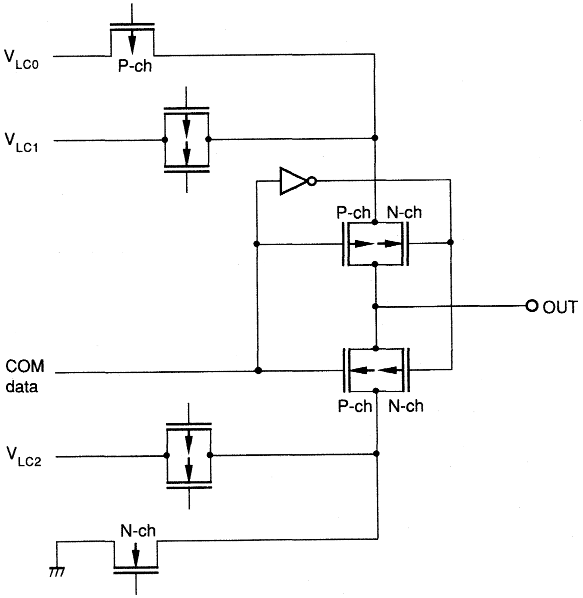


P.U.R.: Pull-Up Resistor

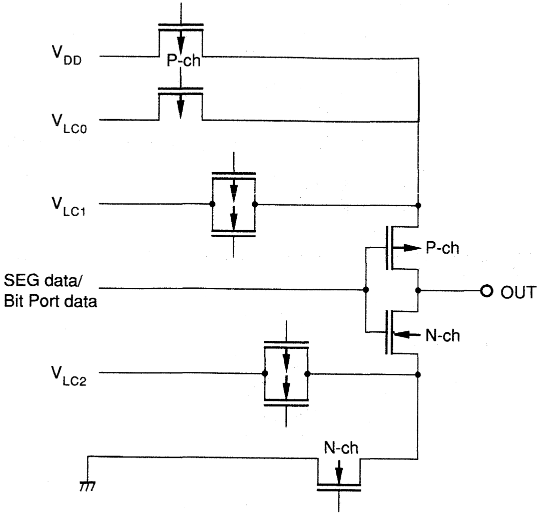
(8) Type G – A



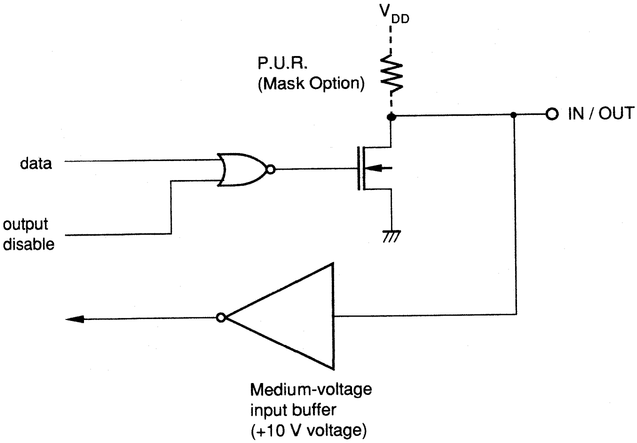
(9) Type G – B



(10) Type G – C

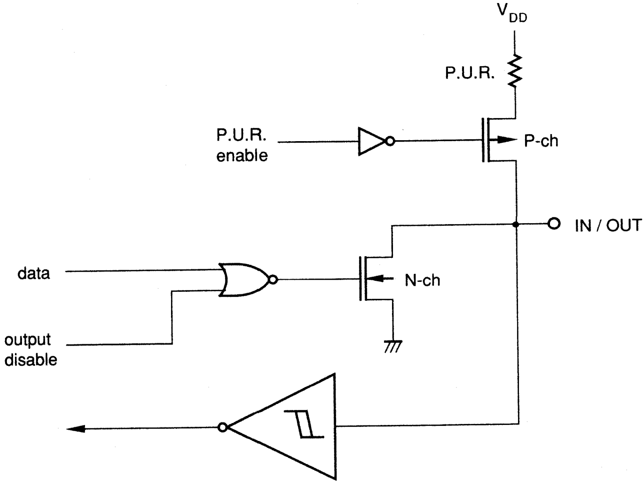


(11) Type M



P.U.R.: Pull-Up Resistor

(12) Type M – C



P.U.R.: Pull-Up Resistor

## 2.4 Processing of Unused Pins

**Table 2.4–1 Unused Pins Lists**

| Pin | Recommended connection |
|---|---|
| P00 / INT4 | Connect to $V_{SS}$. |
| P01 / $\overline{\text{SCK}}$ | Connect to $V_{SS}$ or $V_{DD}$. |
| P02 / SO /SB0 | |
| P03 / SI / SB1 | |
| P10 / INT0–P12 / INT2 | Connect to $V_{SS}$. |
| P13 / TI0 | |
| P20 / PTO0 | Input mode: Connect to $V_{SS}$ or $V_{DD}$.<br>Output mode: Do not connect. |
| P21 | |
| P22 / PCL | |
| P23 / BUZ | |
| P30–P33 | |
| P40–P43 | |
| P50–P53 | |
| P60–P63 | |
| P70–P73 | |
| S0–S23 | Do not connect. |
| S24 / BP0–S31 / BP7 | |
| COM0–COM3 | |
| $V_{LC0}$–$V_{LC3}$ | Connect to $V_{SS}$. |
| BIAS | Only when all of $V_{LC0}$–$V_{LC2}$ are unused, connect BIAS to $V_{SS}$; otherwise, do not connect. |
| XT1 | Connect XT1 to $V_{SS}$ or $V_{DD}$. |
| XT2 | Do not connect XT2. |

## 2.5 Mask Option Selection

The pins contain the mask option function as listed in Table 2.5-1.
However, mask option is not contained in the μPD75P308 or μPD75P316.

**Table 2.5–1 Mask option selection**

| Pin names | Mask option |
|---|---|
| P40–P43,<br>P50–P53 | . Pull-up resistor is included.<br>specified bit-wise.<br>. Pull-up resistor is not included. |
| Split resistor for LCD drive power supply | . Split resistor is included.<br>. Split resistor is not included. |

## 3. ARCHITECTURE AND MEMORY MAP

The μCOM-75X architecture for the μPD753XX adopts data memory bank configuration and memory mapped I / O to provide features such as:

- Internal RAM with maximum of 4K words x four bits (12-bit addresses)
- Peripheral hardware extensibility

This chapter covers these topics.

### 3.1 Data Memory Bank Configuration and Addressing Modes

3.1.1 Data memory bank configuration

A general purpose static RAM (480 words x four bits) is incorporated in data memory space addresses 000H to 1DFH, and display data memory (32 words x four bits) is incorporated in adresses 1E0H to 1FFH. Peripheral hardware such as input/output ports and timers is allocated to addresses F80H to FFFH.

To adress the data memory space (4K words x four bits) with 12-bit addresses, the μPD753XX adopts a memory bank configuration where the low-order eight bits of an address are specified either directly or indirectly by using an instruction, and the high-order four bits of an address are specified by using a memory bank.

To specify the memory bank (MB), two hardware devices are incorporated in the μPD753XX:

- Memory bank enable flag (MBE)
- Memory bank selection register (MBS)

MBS is a register used to select a memory bank. The μPD753XX allows 0, 1, or 15 to be set in MBS. MBE is a flag used to determine whether or not the memory bank selected by using MBS is validated. As shown in Fig. 3.1–1, when MBE is set to 0, the specified memory bank is fixed regardless of how MBS is set; when MBE is set to 1, memory bank switching can be performed to extend data memory space by setting MBS.

In data memory space addressing, normally MBE is set to 1, and the data memory area of the memory bank specified by using MBS is handled. An efficient program can be prepared by using MBE = 0 mode and MBE = 1 mode appropriately in each process of the program.

| | Applicable program processing | Effects |
|---|---|---|
| MBE = 0 mode | ° Interrupt service | MBS save and restore are made unnecessary. |
| | ° Repetitive processing of internal hardware operation and general purpose RAM operation | MBS change is made unecessary. |
| | ° Subroutine processing | MBS save and restore are made unnecessary. |
| MBE = 1 mode | ° Normal program processing | |

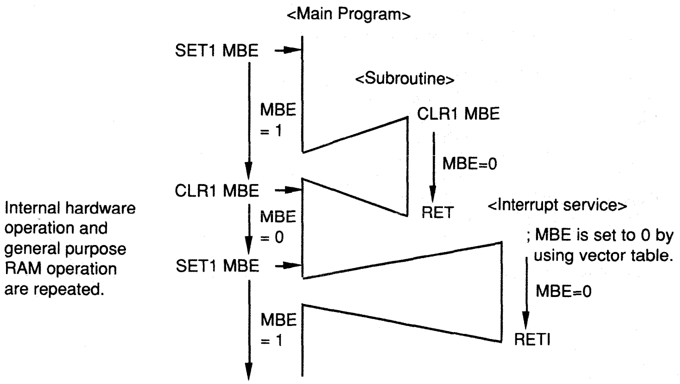

<Main Program>
SET1 MBE
MBE = 1
<Subroutine>
CLR1 MBE
MBE=0
RET
CLR1 MBE
MBE = 0
SET1 MBE
MBE = 1
Internal hardware operation and general purpose RAM operation are repeated.
<Interrupt service>
; MBE is set to 0 by using vector table.
MBE=0
RETI

**Figure 3.1–1: Proper Use of MBE = 0 and MBE = 1 Modes**

MBE is saved or restored automatically during subroutine processing and can be changed as desired. When interrupt service is made, MBE is automatically saved or restored. MBE during interrupt service can also be specified at the same time interrupt service starts by setting the interrupt vector table. Thus, it is useful for high-speed interrupt service.

To change the MBS setting in subroutine processing or interrupt service, MBS is saved and restored by using PUSH and POP instructions.

MBE is set and reset by using SET1 and CLR1 instructions.
MBS is set by using the SEL instruction.

Example 1: To clear MBE and fix memory bank

```
        CLR1   MBE  ;  MBE ← 0
```

Example 2: To select memory bank 1

```
        SET1   MBE  ;  MBE ← 1
        SEL    MB1  ;  MBS ← 1
```

3.1.2 Data memory addressing modes

The μCOM-75X architecture adopted for the μPD753XX provides seven addressing modes, as shown in Fig. 3.1–2, for efficiently addressing the data memory space for each bit length of data to be processed.

(1) 1-bit direct adressing (mem. bit)

This addressing mode directly specifies each bit in all the data memory space by using the instruction operand.

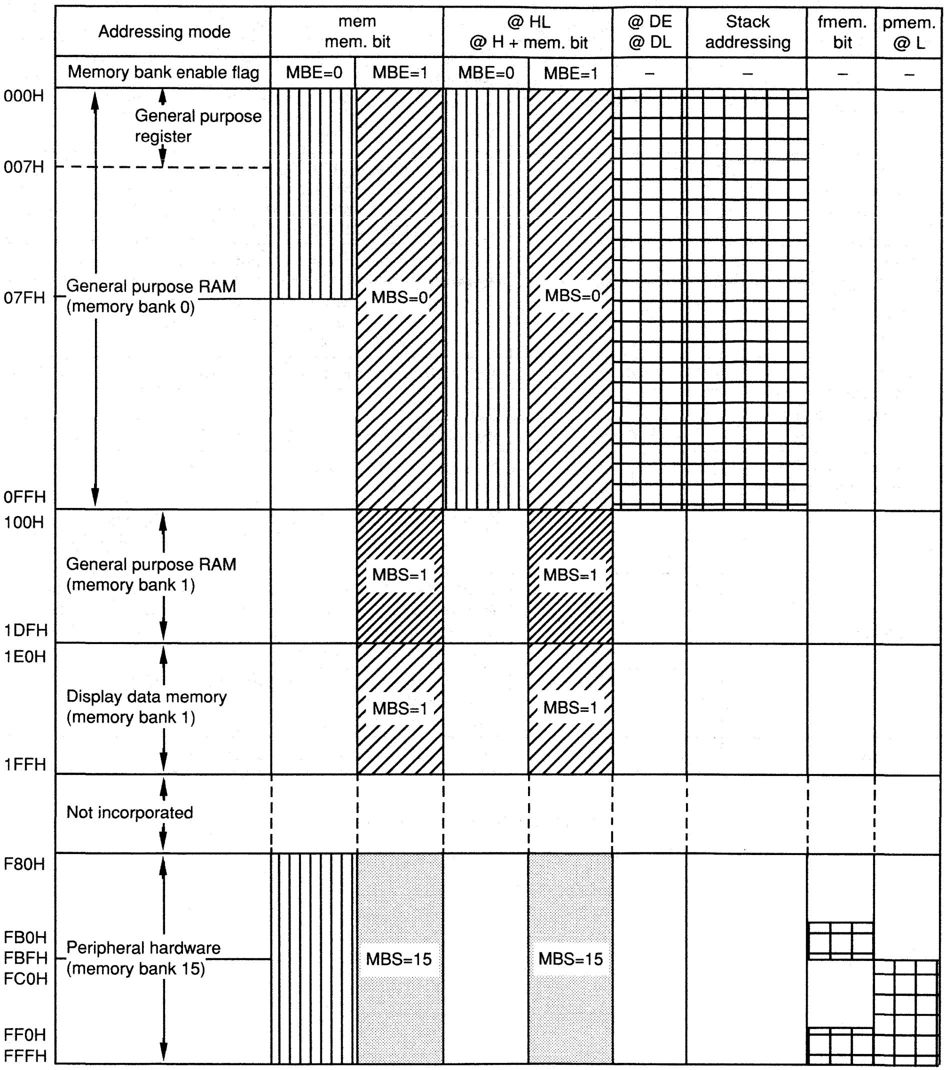
When MBE is set to 0, the specified memory bank (MB) is fixed to memory bank 0 (MB0) if the address specified in the operand is 00H-7FH; it is fixed to memory bank 15 (MB15) if the address specified in the operand is 80H-FFH. Thus, when MBE is set to 0, both the general purpose RAM area (000H-07FH) and peripheral hardware area (F80H-FFFH) can be addressed.

When MBE is set to 1, any memory bank can be specified by setting MBS; data memory space that can be specified can be extended.

The 1-bit direct addressing mode is applicable to bit set and reset instructions (SET1 and CLR1) and bit test instructions (SKT and SKF).

Example:
To set FLAG1, reset FLAG2, and test whether or not FLAG3 is set to 0.

```
FLAG1   EQU     03FH. 1   ;   3FH address bit 1
FLAG2   EQU     087H. 2   ;   87H address bit 2
FLAG3   EQU     0A7H. 0   ;   A7H address bit 0
        SET1    MBE       ;   MBE   ← 1
        SEL     MB0       ;   MBS   ← 0
        SET1    FLAG1     ;   FLAG1 ← 1
        CLR1    FLAG2     ;   FLAG2 ← 0
        SKF     FLAG3     ;   FLAG3 = 0 ?
```



–: don`t care

**Figure 3.1–2 Data Memory Configuration and Addressing Range of each Addressing Mode**

| Addressing mode | Representation format | Specified address |
|---|---|---|
| 1-bit direct addressing | mem. bit | Bit indicated by a bit at the address indicated by MB and mem.<br>However,<br>{ MBE = 0, MB = 0 when mem = 00H-7FH<br>MB = 15 when mem = 80H-FFH<br>When MBE =1, MB = MBS |
| 4-bit direct addressing | mem | Address indicated by MB and mem. However,<br>{ MBE = 0, MB = 0 when mem = 00H-7FH<br>MB = 15 when mem = 80H-FH<br>When MBE = 1, MB = MBS |
| 8-bit direct addressing | | Address indicated by MB and mem (mem is an even address).<br>However,<br>{ MBE = 0, MB = 0 when mem = 00H-7FH<br>MB = 15 when mem = 80H-FFH<br>When MBE = 1, MB = MBS |
| 4-bit register indirect addressing | @ HL | Address indicated by MB and HL. However, MB = MBE • MBS |
| | @ DE | Memory bank 0 address indicated by DE |
| | @ DL | Memory bank 0 address indicated by DL |
| 8-bit register indirect addressing | @ HL | Address indicated by MB and HL (the L register contains an even number). However, MB = MBE • MBS |
| Bit manipulation addressing | fmem. bit | Bit indicated by a bit at the address indicated by fmem.<br>However, fmem { FB0H-FBFH (hardware related to interrupt)<br>FF0H-FFFH (I/O port) |
| | pmem. @ L | Bit indicated by the low-order two bits of the L register at the address indicated by the high-order 10 bits of pmem and the high-order two bits of the L register.<br>However, pmem = FC0H-FFFH |
| | @ H + mem. bit | Bit indicated by a bit at the address indicated by MB, H, and the low-order four bits of mem.<br>However, MB = MBE • MBS |
| Stack addressing | | Memory bank 0 address indicated by SP. |

**Figure 3.1–3 Addressing Modes**

(2) 4-bit direct addressing (mem)

This addressing mode directly specifies all the data memory space in 4-bit units by using the instruction operand.

As with the 1-bit direct addressing mode, when MBE is set to 0, the 4-bit direct address mode allows that only the general purpose RAM area (000H-07FH) and peripherial hardware area (F80H-FFFH) can be specified. When MBE is set to 1, any memory bank (MB) can be specified by setting MBS and the data memory space that can be specified is extended to all the space.

The 4-bit direct addressing mode is applicable to the MOV, XCH, INCS, IN, and OUT instructions.

Example 1: To input port 4 and store in "DATA1".

```
DATA1 EQU    5FH        ;  "DATA1" is address 5HF.
      CLR1   MBE        ;  MBE← 0
      IN     A, PORT4   ;  A← PORT4
      MOV    DATA1, A   ;  (DATA1)← A
```

Example 2: To output data in "BUFF" to port 5.

```
BUFF  EQU    11AH       ;  "BUFF" is address 11AH
      SET1   MBE        ;  MBE← 1
      SEL    MB1        ;  MBS← 1
      MOV    A, BUFF    ;  A← (BUFF)
      SEL    MB15       ;  MBS← 15
      OUT    PORT5, A   ;  PORT5← A
```

Caution:

If data related to the input /output ports is stored in general purpose RAM of bank 1 as in this example, the program is made less efficient. If data related to the input /output ports is stored in addresses 00H-7FH of bank 0, a program can be prepared without changing MBS as in Example 1.

(3) 8-bit direct addressing (mem)

This addressing mode directly specifies all the data memory space in 8 bit units by using the instruction operand. Only even address can be specified in the mem operand. The 4-bit data at the address specified in the operand and the 4-bit data at the address + 1 are paired and transferred to the 8-bit accumulator (XA register pair) for 8-bit processing.

The memory banks specified in the addressing mode are the same as in the 4-bit direct addressing mode.

The 8-bit direct addressing mode is applicable to the MOV, XCH, IN, and OUT instructions.

Example 1: To transfer 8-bit data in ports 4 and 5 to addresses 20H and 21H.

```
DATA  EQU    020H
      CLR1   MBE        ;  MBE← 0
      IN     XA, PORT4  ;  X← port 5, A← port 4
      MOV    DATA, XA   ;  (21H)← X, (20H)← A
```

Example 2: To read 8-bit data input to the serial interface shift register (SIO), set transfer data, and start of transfer.

```
      SEL    MB15       ;  MBS← 15
      XCH    XA, SIO    ;  XA↔ (SIO)
```

(4) 4-bit register indirect addressing (@rpa)

This addressing mode indirectly specifies the data memory space in 4-bit units by using the data pointer (general purpose register pair) specified in the instruction operand.

The data pointers are the HL register pair, which enables all the data memory space to be specified according to MB = MBE • MBS specification, and the DE and DL register pairs, which always indicate memory bank 0 regardless of how MBE and MBS are set. Efficient programs can be prepared by using the data memory bank to be used.

Example: To transfer data at 50H-57H to 110H-117H.

```
DATA1  EQU   57H
DATA2  EQU   117H
       SET1  MBE
       SEL   MB1
       MOV   D, #DATA1 SHR 4
       MOV   HL, #DATA2 AND 0FFH ;  HL← 17H
LOOP:  MOV   A, @DL              ;  A← (DL)
       XCH   A, @HL              ;  A← (HL)
       DECS  L                   ;  L← L–1
       BR    LOOP
```

The addressing mode using the HL register pair for the data pointer has wide applications such as data transfer, operation, comparison, and input / output. The addressing mode using the DE or DL register pair is applicable to the MOV and XCH instructions.

As shown in Fig. 3.1–4, data memory space addresses can be updated as desired by combining the addressing mode with general purpose register (or register pair) increment and decrement instructions.

Example 1: To compare data at 50H-57H with data at 110H-117H.

```
DATA1  EQU   57H
DATA2  EQU   117H
       SET1  MBE
       SEL   MB1
       MOV   D, #DATA1 SHR 4
       MOV   HL, #DATA2 AND 0FFH
LOOP:  MOV   A, @DL
       SKE   A, @HL    ;  A = (HL) ?
       BR    NO        ;  NO
       DECS  L         ;  YES, L← L–1
       BR    LOOP
```

Example 2: To clear data memory area 04H-FFH.

```
       SEL   MBE
       MOV   XA, #00H
       MOV   L, #04H
LOOP:  MOV   @HL, A    ;  (HL)← A
       INCS  L         ;  L← L+1
       BR    LOOP
       INCS  H         ;  H← H+1
       BR    LOOP
```

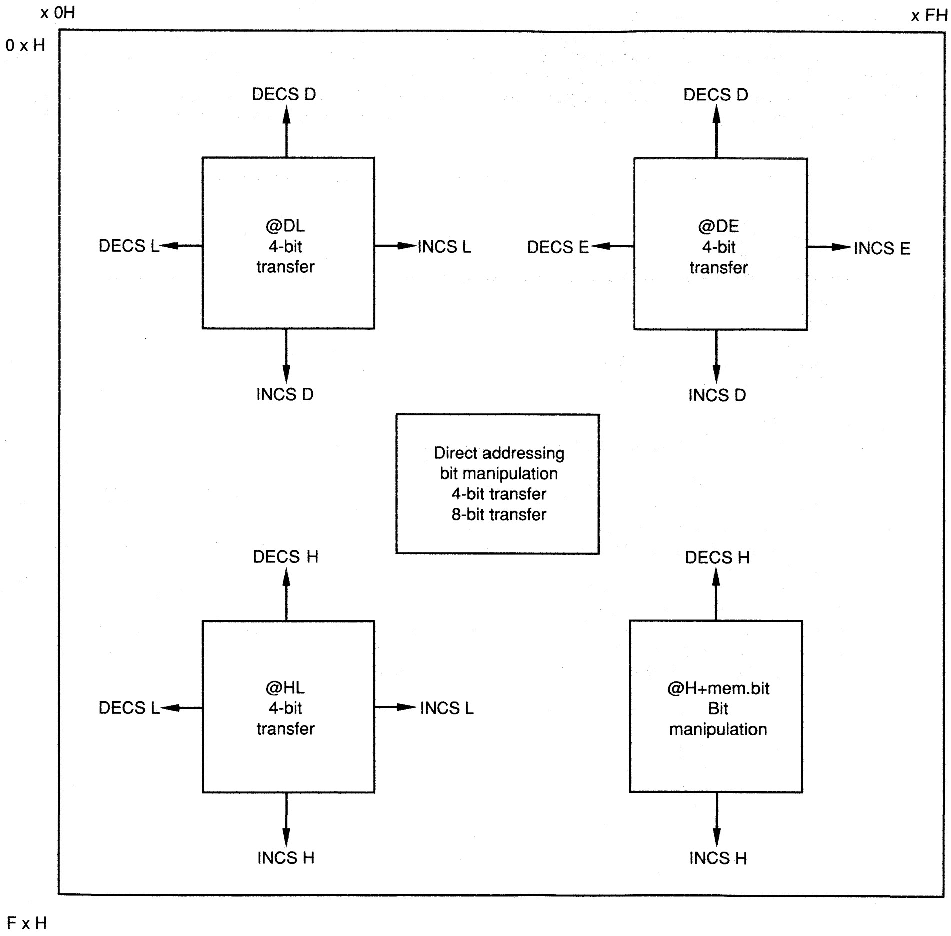


**Figure 3.1–4 General Purpose RAM Address Update Method**

(5) 8-bit register indirect addressing (@HL)

This addressing mode indirectly specifies all the data memory space in 8-bit units by using the data pointer (HL register pair). The 4-bit data at the address setting data pointer bit 0 (L register bit 0) to 0 and the 4-bit data at the address + 1 are paired and transferred to the 8-bit accumulator (XA register) for 8-bit processing.

The memory banks specified in the addressing mode are the same as those when the HL register is specified in the 4-bit register indirect addressing mode (MB = MBE • MBS). The 8-bit register indirect addressing mode is applicable to the MOV, XCH, and SKE Instructions.

Example 1: To compare the count register (T0) value of timer / event counter 0 with data at addresses 30H and 31H for equality.

```
DATA  EQU   30H
      CLR1  MBE
      MOV   HL, #DATA
      MOV   XA, T0       ;   XA← count register 0
      SKE   A, @HL       ;   A = (HL) ?
      BR    NO
      INCS  L
      MOV   A, X         ;   A← X
      SKE   A, @HL       ;   A = (HL) ?
```

(6) Bit manipulation addressing

This addressing mode is used to perform bit manipulations such as Boolean operation or bit transfer on any bit in all of the data memory space.

Although the 1-bit direct addressing mode is applicable only to the bit set, reset, and test instructions, the bit manipulation addressing mode enables bit manipulations such as Boolean operations by using the AND1, OR1, and XOR1 instructions and allows bit test and bit reset by using the SKTCLR instructions.

The following three types of bit manipulation addressing modes can be used according to the data memory address to be used:

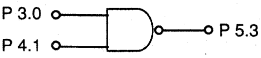
(a) Specific address bit direct addressing (fmem. bit).

This addressing mode enables, peripheral hardware that frequently uses bit manipulation, such as input / output ports and interrupt flags, to be operated at any time independently of the memory bank setting. Thus, the addressing mode is applicable to data memory addresses FF0H-FFFH (where input / output ports are mapped) and FB0H-FBFH (where hardware related to interrupts is mapped). For the hardware of the two data memory areas, bit manipulation can be performed in direct addressing at any time as desired regardless of how MBS and MBE are set.

Example 1: To test the timer 0 interrupt request flag (IRQT0); if the request flag is set, clear the flag and reset P63.

```
SKTCLR   IRQT0      ;   IRQT0 = 1 ?
BR       NO         ;   NO
CLR1     PORT6.3    ;   YES
```

Example 2: To reset P53 if both P30 and P41 are set to 1.



```
(i)     SET1   CY            ;   CY ← 1
        AND1   CY, PORT3.0   ;   CY P3.0
        AND1   CY, PORT4.1   ;   CY P4.1
        SKT    CY            ;   CY = 1 ?
        BR     SETP
        CLR1   PORT5.3       ;   P53 ← 1
          :
          :
 SETP:  SET1 PORT5.3         ;   P53 ← 1
          :
          :

(ii)    SKT    PORT3.0       ;   P30 = 1 ?
        BR     SETP
        SKT    PORT4.1       ;   P41 = 1 ?
        BR     SETP
        CLR1   PORT5.3       ;   P53 ← 0
          :
          :
STEP:   SET1   PORT5.3       ;   P53 ← 1
```

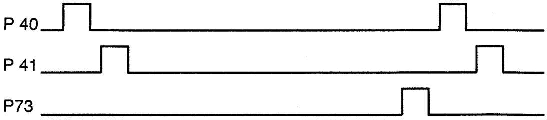
(b) Specific address bit register indirect addressing (pmem. @L).

This addressing mode indirectly specifies each bit of peripheral hardware, such as an input / output port, by using the L register for successive operation. The addressing mode is applicable to data memory addresses FC0H-FFFH.

The addressing mode directly specifies the high-order 10 bits of a 12-bit data memory address in the pmem operand and indirectly specifies the low-order 2-bit data memory address part and the bit address by using the L register. Thus, 16 bits (four ports) can be manipulated (operated) successively by L register specification.

The address mode also enable bit manipulation to be performed at any time independently of how MBE and MBS are set.

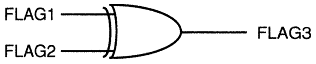
Example: To output a pulse to each bit of ports 4 to 7 in order.



```
        MOV    L, #0
LOOP:   SET1   PORT4. @L    ;   Port 4–7 bit (L1-0) ← 1
        CLR1   PORT4. @L    ;   Port 4–7 bit (L1-0) ← 0
        INCS   L
        BR     LOOP
```

(c) Special 1-bit direct addressing (@H + mem. bit)
This addressing mode enables bit manipulation to be performed on any bit in all of the data memory space.
The addressing mode indirectly specifies the high-order four bits of data memory address of the memory bank specified by MB = MBE MBS by using the H register and directly specifies the low-order 4-bit data memory address part and the bit address in the operands. It enable various types of bit manipulation to be performed on any bit in all of the data memory space.

Example: To reset address 32H bit 2 (FLAG3) if both address 30H bit 3 (FLAG1) and address 31H bit 0 (FLAG2) are set to 0 or 1.



```
FLAG1 EQU   30H.3
FLAG2 EQU   31H.0
FLAG3 EQU   32H.2
      SEL   MB0
      MOV   H, #FLAG1 SHR 6
      CLR1  CY
      OR1   CY, @H + FLAG1   ;  CY← 0
      XOR1  CY, @H + FLAG2   ;  CY← CY-FLAG1
      SET1  @H + FLAG3       ;  CY← CY-FLAG2
      SKT   CY               ;  CY = 1 ?
      CLR1  @H + FLAG3       ;  FLAG← 30
```

(7) Stack addressing
The stack addressing mode is used for register save and restore during interrupt service or subroutine processing.
The processing mode specifies an address by using the 8-bit stack pointer (data memory bank 0).
The addressing mode can also be used for register save and restore by executing the PUSH and POP instructions.

Example 1: To save and restore register in subroutine processing.

```
SUB: PUSH   XA
     PUSH   HL
     PUSH   BS   ;  MBS is saved.
       :
       :
     POP    BS
     POP    HL
     POP    XA
     RET
```

Example 2: To transfer the HL register pair contents to the DE register pair.

```
     PUSH   HL
     POP    DE   ;  DE← HL
```

Example 3: To branch to the address indicated by [XABC] register.

```
     PUSH   BC
     PUSH   XA
     RET         ;  Branch to XABC address
```

### 3.2 Memory Mapped I / O

The μPD753XX adopts memory mapped I / O where peripheral hardware such as input / ports and timers is allocated to addresses F80H-FFFH of the data memory space as shown in Fig. 3.1–2. Thus, peripheral hardware is controlled entirely by memory operation instructions rather than special instructions. (To easily understand programs, some hardware control mnemonics are provided.)
Table 3.2–1 lists the addressing modes that can be used to operate peripheral hardware.
Memory bank 1 is specified to operate the display data memory mapped in addresses 1E0H-1FFH.

**Table 3.2–1 Applicable Addressing Modes during Peripheral Hardware Operation**

| | Applicable addressing mode | Applicable hardware |
|---|---|---|
| Bit manipulation | With MBE=0 or (MBE=1 and MBS=15), direct addressing (specification in mem. bit). | All hardware where bit manipulation can be performed. |
| | Direct addressing regardless of how MBE and MBS are set. (specification in fmem. bit) | ISTO, MBE<br>IExxx, IRQxxx, PORTn.x |
| | Indirect addressing regardless of how MBE and MBS are set. (specification in pmem. @L) | BSBn.x<br>PORTn.x |
| 4-bit manipulation | With MBE=0 or (MBE=1 and MBS=15), direct addressing (specification in mem). | All hardware where 4-bit manipulation can be performed. |
| | With (MBE=1 and MBS=15), register indirect addressing (specification in @HL). | |
| 8-bit manipulation | With MBE=0 or (MBE=1 and MBS=15), direct addressing (specification in mem), mem must be an even address. | All hardware where 8-bit manipulation can be performed. |
| | With MBE=1 and MBS=15, register indirect addressing (specification in @HL; the L register must contain an even number. | |

```
Example: CLR1     MBE        ;  MBE = 0
         SET1     TM0.3      ;  Timer 0 starts.
         EI       IE0        ;  INT0 is enabled.
         DI       IE1        ;  INT1 is disabled.
         SKTCLR   IRQ2       ;  INT2 request flag is tested and cleared.
         SET1     PORT4. @L  ;  Port 4 is set.
         IN       A, PORT0   ;  A ← port 0
         OUT      PORT4, XA  ;  Port 5, 4 ← XA
```

Figs. 3.2–1 to 3.2–3 shows the μPD753XX I/O map.
The columns in the figures mean:

- Abbreviation:
  Name indicating internal hardware address. It can be entered in the instruction operand field.

- R/W
  Indicates whether the hardware device can be read or written.
  – R/W : Read and write are enabled.
  – R : Read only is enabled.
  – W : Write only is enabled.

- Number of bits that can be manipulated:
  Indicates the number of bits that can be processed when the hardware device is operated.
  O: Bit manipulation is enabled in 1-4, 4-, or 8-bit units as specified in the column.
  Δ: Only some bits can be manipulated. See Remarks for the bits that can be manipulated.
  –: Bit manipulation cannot be performed in 1-, 4-, or 8-bit units as specified in the column.

- Bit manipulation addressing:
  Indicates the applicable bit manipulation addressing for performing bit manipulation on the hardware device.

| Address | Hardware name (abbreviation) | | | | R/W | Number of bits that can be manipulated | | | Bit manipulation addressing | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| | b3 | b2 | b1 | b0 | | One bit | Four bit | Eight bit | | |
| F80H | Stack pointer (SP) | | | | R/W | – | – | O | | Bit 0 is fixed to 0. |
| | | | | | | | | | | |
| F85H | Basic interval timer mode register (BTM) | | | | W | Δ | O | – | mem. bit | Bit manipulation can be performed only on bit 3. |
| F86 H | Basic interval timer (BT) | | | | R | – | – | O | | |
| | | | | | | | | | | |
| F8CH | Display mode register (LCDM) | | | | W | Δ | – | O | mem. bit | Bit manipulation can be performed only on bit 3. |
| | | | | | | – | – | | | |
| F8EH | Display control register (LCDC) | | | | W | – | O | – | | |
| F98H | Watch mode register (WM) | | | | R/W | Δ | – | O | mem. bit | Bit test can be made only on bit 3. |
| | | | | | W | – | – | | | |
| FA0H | Timer/event counter 0 mode register (TM0) | | | | W | Δ | – | O | mem. bit | Bit manipulation can be performed only on bit 3. |
| | | | | | | – | – | | | |
| FA2H | TOE0 (Note) | | | | W | O | – | – | mem. bit | |
| FA4H | Timer/event counter 0 count register (T0) | | | | R | – | – | O | | |
| | | | | | | – | – | | | |
| FA6H | Timer/event counter 0 modulo register (TMOD0) | | | | W | – | – | O | | |
| | | | | | | – | – | | | |
| FB0H | 0 | IST0 | MBE | 0 | R/W | O | O | O | fmem. bit | |
| | Program status word (PSW) | | | | R | – | – | | | |
| FB2H | (IME) | | | | – | – | – | – | | EI and DI instructions are used |
| FB3H | Processor clock control register (PCC) | | | | W | – | O | | | |
| FB4H | INT0 mode register (IM0) | | | | W | – | O | – | | Bit 2 is fixed to 0. |
| FB5H | INT1 mode register (IM1) | | | | W | – | O | | | Bit 3 to 1 are fixed to 0. |
| FB6H | INT2 mode register (IM2) | | | | W | – | O | | | Bits 3 and 2 are fixed to 0 |
| FB7H | System clock control register (SCC) | | | | W | O | – | – | | Only bits 3 and 0 can be bit manipulated. |

Note: TOE0 = timer / event counter 0 output enable flag (W)

(to be continued)

**Figure 3.2–1 μPD753XX I/O Map**

| Address | Hardware name (abbreviation) | | | | R/W | Number of bits that can be manipulated | | | Bit manipulation addressing | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| | b3 | b2 | b1 | b0 | | One bit | Four bit | Eight bit | | |
| FB8H | IE4 | IRQ4 | IEBT | IRQBT | R/W | O | O | – | fmem. bit | |
| FBAH | | | IEW | IRQW | R/W | O | O | | | |
| FBCH | | | IET0 | IRQT0 | R/W | O | O | – | | |
| FBDH | | | IECSI | IRQCSI | R/W | O | O | | | |
| FBEH | IE1 | IRQ1 | IE0 | IRQ0 | R/W | O | O | – | | |
| FBFH | | | IE2 | IRQ2 | R/W | O | O | | | |
| FC0H | Bit sequential buffer 0 (BSB0) | | | | R/W | O | O | O | mem. bit<br>pmem @L | |
| FC1H | Bit sequential buffer 1 (BSB1) | | | | R/W | O | O | | | |
| FC2H | Bit sequential buffer 2 (BSB2) | | | | R/W | O | O | O | | |
| FC3H | Bit sequential buffer 3 (BSB3) | | | | R/W | O | O | | | |
| FD0H | Clock output mode register (CLOM) | | | | W | – | O | – | | |
| FDCH | Pull-up resistor specification register group A (POGA) | | | | W | – | – | O | | |
| FE0H | Serial operation mode register (CSIM) | | | | W | – | – | O | | |
| | CSIE | COI | WUP | | R/W | O | O | | mem. bit | |
| FE2H | CMDD | RELD | CMDT | RELT | R/W | O | – | – | mem. bit | Only bit manipulation can be performed on any bit. |
| | SBI control register (SBIC) | | | | | | | | | |
| | BSYE | ACKD | ACKE | ACKT | | | | | | |
| FE4H | Serial I/O shift register (SIO) | | | | R/W | – | – | O | | |
| FE6H | Slave address register (SVA) | | | | W | – | – | O | | |
| FE8H | PM33 | PM32 | PM31 | PM30 | W | – | – | O | | |
| | Port mode register group A (PMGA) | | | | | | | | | |
| | PM63 | PM62 | PM61 | PM60 | | | | | | |
| FECH | – | PM2 | – | – | W | – | – | O | | |
| | Port mode register group B (PMGB) | | | | | | | | | |
| | PM7 | 0 | PM5 | PM4 | | | | | | |

Remarks: 1. IExxx is an interrupt enable flag.
2. IRQxxx is an interrupt request flag.
3. IME is an interrupt master flag.

(to be continued)

**Figure 3.2–1 μPD753XX I/O Map** (con`t)

| Address | Hardware name (abbreviation) | | | | R/W | Number of bits that can be manipulated | | | Bit manipulation addressing | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| | b3 | b2 | b1 | b0 | | One bit | Four bit | Eight bit | | |
| FF0H | Port 0 (PORT 0) | | | | R | O | O | – | fmem. bit<br>pmem. @L | |
| FF1H | Port 1 (PORT 1) | | | | R | O | O | | | |
| FF2H | Port 2 (PORT 2) | | | | R/W | O | O | – | | |
| FF3H | Port 3 (PORT 3) | | | | R/W | O | O | | | |
| FF4H | Port 4 (PORT 4) | | | | R/W | O | O | O | | |
| FF5H | Port 5 (PORT 5) | | | | R/W | O | O | | | |
| FF6H<br>(Note) | KR3 | KR2 | KR1 | KR0 | R/W | O | O | O | | |
| | Port 6 (PORT 6) | | | | | | | | | |
| FF7H<br>(Note) | KR7 | KR6 | KR5 | KR4 | R/W | O | O | | | |
| | Port 7 (PORT 7) | | | | | | | | | |

Note: KR0–KR7 can only be read. When 4bit parallel is input, specified at PORT6 or PORT7.

**Figure 3.2–1 μPD753XX I/O Map** (con`t)

## 4 INTERNAL CPU FUNCTIONS

### 4.1 Program Counter (PC) –12–Bit (μPD75304)
### 13–Bit (μPD75306, 75308)
### 14–Bit (μPD75312, 75316)

The program counter is binary counter which holds program memory address information. The μPD75304 is configured of 12-bit (See Fig. 4.1–1 (a)). The μPD75306 and 75308 are configured of 13-bit (See Fig. 4.4–1 (b)). The μPD75312 and 75316 are configured of 14-bit (See Fig. 4.4–1 (c)).

(a) μPD75304 format

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

(b) μPD75306 and 75308 format

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

(c) μPD75312 and 75316 format

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

**Figure 4.1–1 Program Counter Format**

Each time one instruction is executed, normally the program counter is automatically incremented according to the number of the bytes in the instruction.

When a branch instruction (BR or BRCB) is executed, the register pair contents or immediate data indicating the branch destination address is loaded into all or some of the PC bits.

When a subroutine call instruction (CALL or CALLF) is executed or a vectored interrupt occurs, the current PC contents (return address already incremented to fetch the next instruction) are saved in stack memory (data memory indicated by the stack pointer), then the jump destination address is loaded.
When a return instruction (RET, RETS, or RETI) is executed, the stack memory contents are loaded into the PC.
When the $\overline{\text{RESET}}$ signal is generated, the program memory contents are loaded into the program counter for initialization, as follows: (Program can be started at any desired address.)

| | | | |
|---|---|---|---|
| µPD75304: | PC11–PC8 | ← | low-order four bits at address 0000H |
| | PC7 –PC0 | ← | eight bits at address 0001H |
| µPD75306, µPD75308: | PC12–PC8 | ← | low-order five bits at address 0000H |
| | PC7 –PC0 | ← | eight bits at address 0001H |
| µPD75312, µPD75316: | PC13–PC8 | ← | low-order six bits at address 0000H |
| | PC7 –PC0 | ← | eight bits at address 0001H |

### 4.2 Program Memory (ROM) - 4096 Words x Eight bit (µPD75304)

**6016 Words x Eight bit (µPD75306)**
**8064 Words x Eight bit (µPD75308)**
**12160 Words x Eight bit (µPD75312)**
**16256 Words x Eight bit (µPD75316)**

The program memory stores programs, interrupt vector table, GETI instruction look up table, and data such as table data. The program memory of the µPD75304, µPD75306, µPD75308, µPD75312, µPD75316, is mask programmable ROM.
Fig. 4.2–1 to 4.2–5 show the program memory map.
The program memory is addressed by using the program counter. Table data can also be referenced by using the table reference instruction (MOVT).
The address range for branching by a branch or subroutine call instruction is as shown in Fig. 4.2–1. When a relative branch instruction (BR $addr) is used, a branch can be made to [PC contents – 15 to –1, +2 to + 16] address independently of block.
The program memory addresses are shown below.

| | | |
|---|---|---|
| • 000H– FFFH | : | µPD75304 |
| • 0000H–177FH | : | µPD75306 |
| • 0000H–1F7FH | : | µPD75308 |
| • 0000H–2F7FH | : | µPD75312 |
| • 0000H–3F7FH | : | µPD75316 |

The following addresses are assigned for special purpose:
(All the area except address 0000H to 0001H Note 1 can be used as normal program memory area.).

- 0000H – 0001H
  Vector address table in which the program start address and MBE setup value are entered when the system is reset. (Reset start can be made at any desired address.)

- 0002H – 000BH
  Vector address table in which the program start address and MBE setup value are entered for each vectored interrupt. (Interrupt service can be started at any desired address.)

- 0020H – 007FH Table area referenced by GETI (Note 2).

Note: 1. Their addresses are used in the µPD753XX.
2. The GETI instruction is used to convert any 2- or 3-byte instruction or any two 1-byte instructions into a 1-byte instruction. The number of program bytes can be reduced.

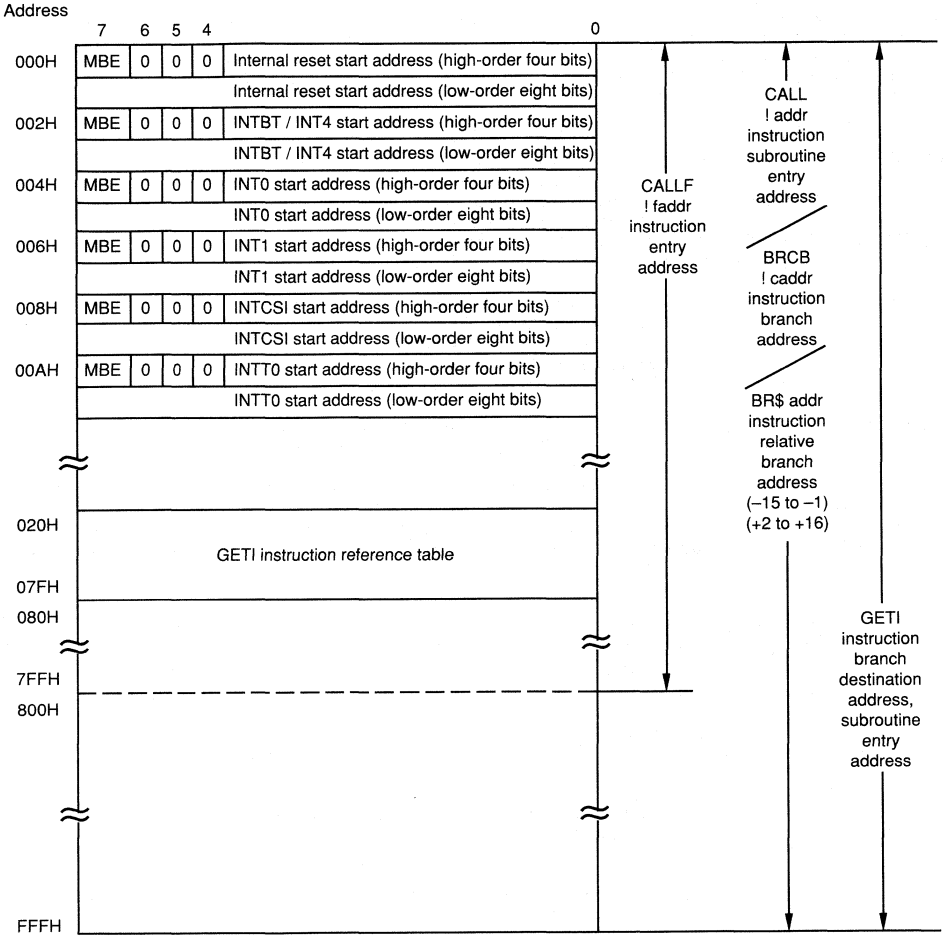

Address
7
6
5
4
0
000H
MBE
0
0
0
Internal reset start address (high-order four bits)
Internal reset start address (low-order eight bits)
002H
MBE
0
0
0
INTBT / INT4 start address (high-order four bits)
INTBT / INT4 start address (low-order eight bits)
004H
MBE
0
0
0
INT0 start address (high-order four bits)
INT0 start address (low-order eight bits)
006H
MBE
0
0
0
INT1 start address (high-order four bits)
INT1 start address (low-order eight bits)
008H
MBE
0
0
0
INTCSI start address (high-order four bits)
INTCSI start address (low-order eight bits)
00AH
MBE
0
0
0
INTT0 start address (high-order four bits)
INTT0 start address (low-order eight bits)
020H
GETI instruction reference table
07FH
080H
7FFH
800H
FFFH
CALLF
! faddr
instruction
entry
address
CALL
! addr
instruction
subroutine
entry
address
BRCB
! caddr
instruction
branch
address
BR$ addr
instruction
relative
branch
address
(–15 to –1)
(+2 to +16)
GETI
instruction
branch
destination
address,
subroutine
entry
address

**Figure 4.2–1 Program Memory Map (µPD75304)**

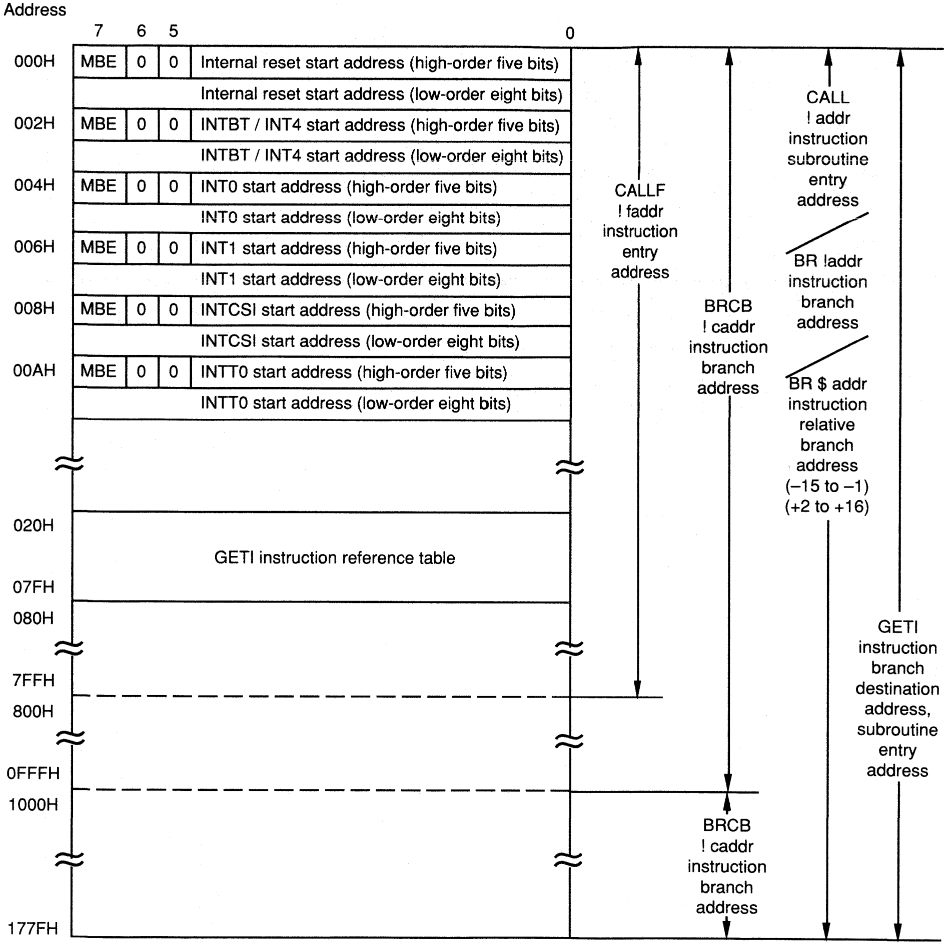

Address
7
6
5
0
000H
MBE 0 0 Internal reset start address (high-order five bits)
Internal reset start address (low-order eight bits)
002H
MBE 0 0 INTBT / INT4 start address (high-order five bits)
INTBT / INT4 start address (low-order eight bits)
004H
MBE 0 0 INT0 start address (high-order five bits)
INT0 start address (low-order eight bits)
006H
MBE 0 0 INT1 start address (high-order five bits)
INT1 start address (low-order eight bits)
008H
MBE 0 0 INTCSI start address (high-order five bits)
INTCSI start address (low-order eight bits)
00AH
MBE 0 0 INTT0 start address (high-order five bits)
INTT0 start address (low-order eight bits)
020H
GETI instruction reference table
07FH
080H
7FFH
800H
0FFFH
1000H
177FH
CALLF ! faddr instruction entry address
BRCB ! caddr instruction branch address
CALL ! addr instruction subroutine entry address
BR !addr instruction branch address
BR $ addr instruction relative branch address (–15 to –1) (+2 to +16)
GETI instruction branch destination address, subroutine entry address
BRCB ! caddr instruction branch address

**Figure 4.2–2 Program Memory Map (μPD75306)**

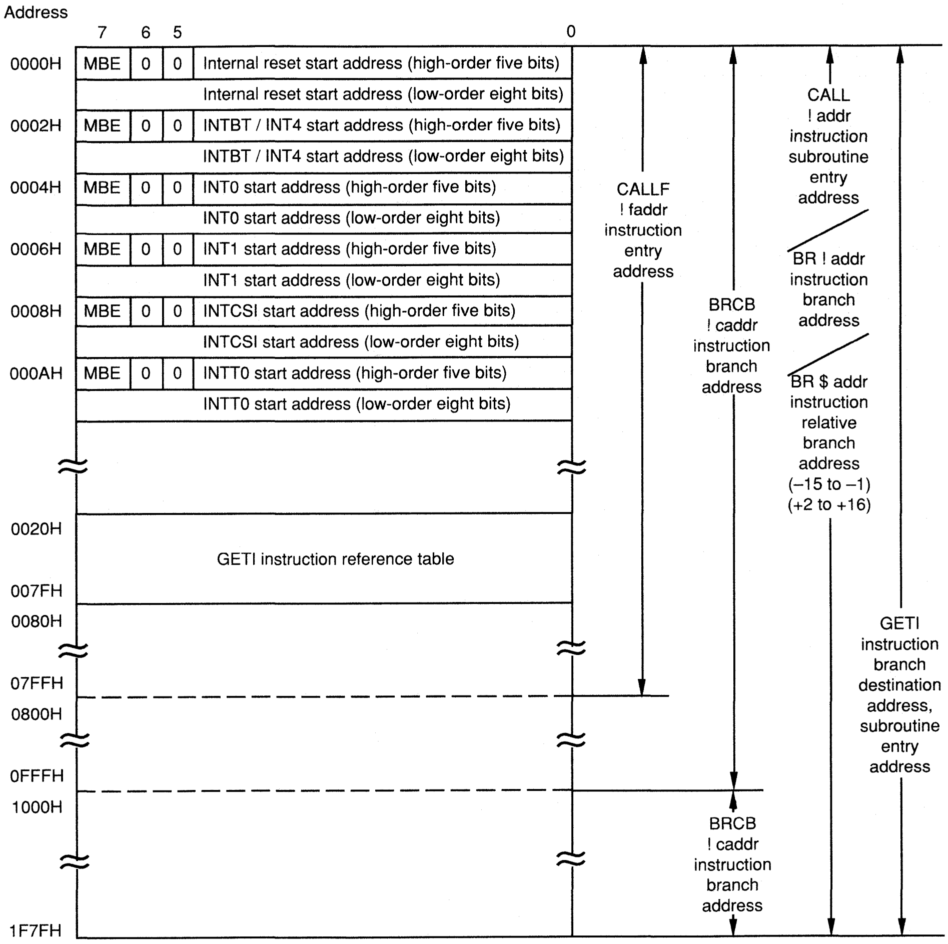

Address
7
6
5
0
0000H
MBE
0
0
Internal reset start address (high-order five bits)
Internal reset start address (low-order eight bits)
0002H
MBE
0
0
INTBT / INT4 start address (high-order five bits)
INTBT / INT4 start address (low-order eight bits)
0004H
MBE
0
0
INT0 start address (high-order five bits)
INT0 start address (low-order eight bits)
0006H
MBE
0
0
INT1 start address (high-order five bits)
INT1 start address (low-order eight bits)
0008H
MBE
0
0
INTCSI start address (high-order five bits)
INTCSI start address (low-order eight bits)
000AH
MBE
0
0
INTT0 start address (high-order five bits)
INTT0 start address (low-order eight bits)
0020H
GETI instruction reference table
007FH
0080H
07FFH
0800H
0FFFH
1000H
1F7FH
CALLF ! faddr instruction entry address
BRCB ! caddr instruction branch address
BRCB ! caddr instruction branch address
CALL ! addr instruction subroutine entry address
BR ! addr instruction branch address
BR $ addr instruction relative branch address (–15 to –1) (+2 to +16)
GETI instruction branch destination address, subroutine entry address

**Figure 4.2–3 Program Memory Map (μPD75308)**

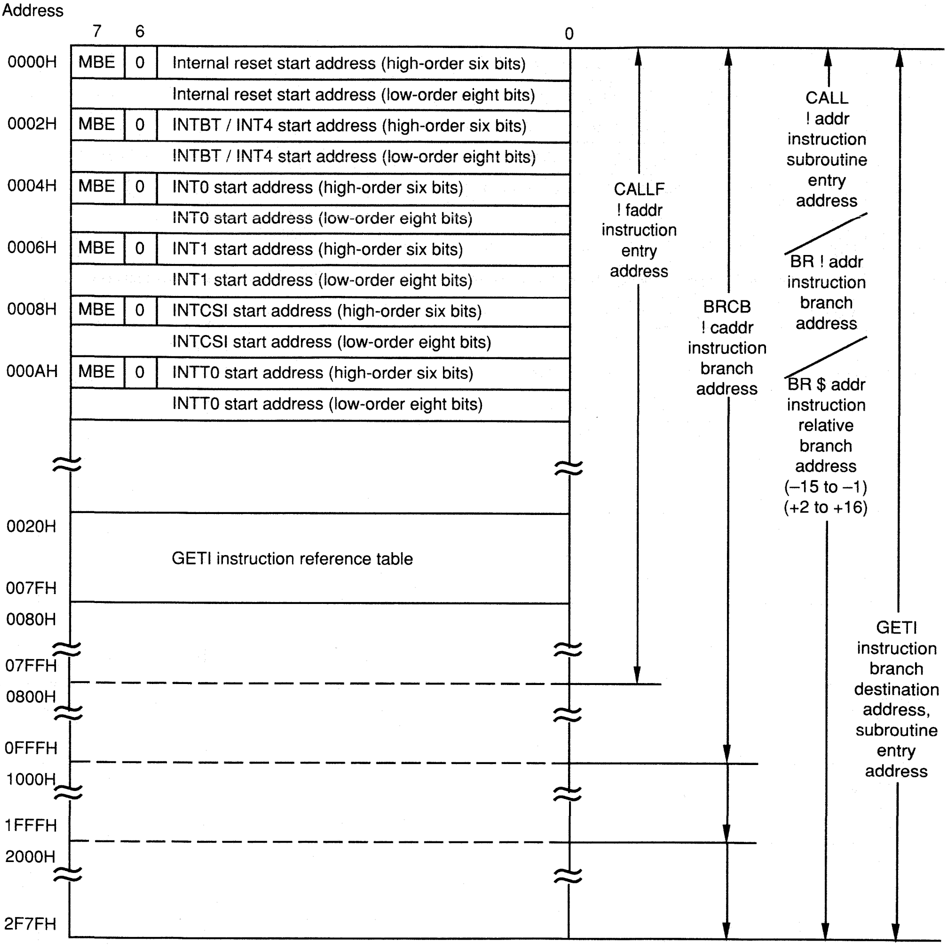

Address
7
6
0
0000H
MBE
0
Internal reset start address (high-order six bits)
Internal reset start address (low-order eight bits)
0002H
MBE
0
INTBT / INT4 start address (high-order six bits)
INTBT / INT4 start address (low-order eight bits)
0004H
MBE
0
INT0 start address (high-order six bits)
INT0 start address (low-order eight bits)
0006H
MBE
0
INT1 start address (high-order six bits)
INT1 start address (low-order eight bits)
0008H
MBE
0
INTCSI start address (high-order six bits)
INTCSI start address (low-order eight bits)
000AH
MBE
0
INTT0 start address (high-order six bits)
INTT0 start address (low-order eight bits)
0020H
GETI instruction reference table
007FH
0080H
07FFH
0800H
0FFFH
1000H
1FFFH
2000H
2F7FH
CALLF ! faddr instruction entry address
BRCB ! caddr instruction branch address
CALL ! addr instruction subroutine entry address
BR ! addr instruction branch address
BR $ addr instruction relative branch address (–15 to –1) (+2 to +16)
GETI instruction branch destination address, subroutine entry address

**Figure 4.2–4 Program Memory Map (µPD75312)**

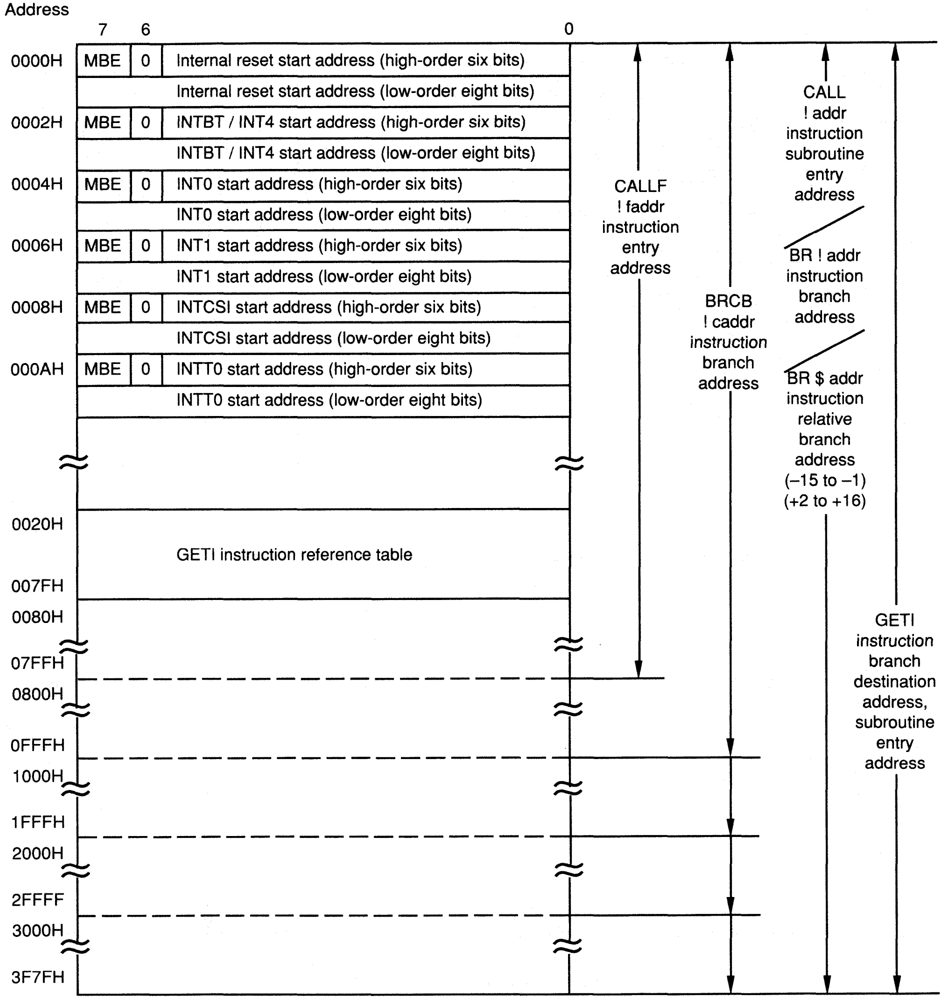

Address
7
6
0
0000H
MBE
0
Internal reset start address (high-order six bits)
Internal reset start address (low-order eight bits)
0002H
MBE
0
INTBT / INT4 start address (high-order six bits)
INTBT / INT4 start address (low-order eight bits)
0004H
MBE
0
INT0 start address (high-order six bits)
INT0 start address (low-order eight bits)
0006H
MBE
0
INT1 start address (high-order six bits)
INT1 start address (low-order eight bits)
0008H
MBE
0
INTCSI start address (high-order six bits)
INTCSI start address (low-order eight bits)
000AH
MBE
0
INTT0 start address (high-order six bits)
INTT0 start address (low-order eight bits)
0020H
GETI instruction reference table
007FH
0080H
07FFH
0800H
0FFFH
1000H
1FFFH
2000H
2FFFF
3000H
3F7FH
CALLF ! faddr instruction entry address
BRCB ! caddr instruction branch address
CALL ! addr instruction subroutine entry address
BR ! addr instruction branch address
BR $ addr instruction relative branch address (−15 to −1) (+2 to +16)
GETI instruction branch destination address, subroutine entry address

**Figure 4.2–5 Program Memory Map (μPD75316)**

### 4.3 Data Memory (RAM) – 512 Words x Four Bits

The data memory is a general purpose static RAM consisting of 512 words x four bits. Since a static RAM stores data during process, subroutine, or interrupt execution, it can also hold data when CPU operation is stopped in the standby mode; it is useful in that the memory contents can be held with battery power for hours.
Fig. 4.3–1 shows a µPD753XX data memory map.
The data memory adopts the bank configuration, consisting of banks 0 and 1 (each 256 words x four bits).
Peripheral hardware is mapped in the memory bank 15 area.
A memory bank is selected by setting the 4-bit memory bank selection register (MBS = 0, 1, or 15) when bank specification is enable by setting the memory bank enable flag (MBE) to 1 (MBE = 1). When bank specification is disabled (MBS = 0), memory bank 0 or 15 is automatically selected according to the current addressing mode. Each bank is addressed by using 8-bit immediate data, a register pair, etc.
Although a data memory word consists of four bits, the data memory can be handled in 1-, 4-, or 8-bit units by using various addressing modes.
See 3.1 for details of memory bank selection and addressing modes.
Specific areas of the data memory are also used for general purpose registers (bank 0; 000H – 0007H), stack memory (bank 0; 000H – 0FFH) and display data memory (bank 1; 1E0H – 1FFH). Data memory does not exist in bank 15. Peripheral hardware and various registers are mapped. Do not access addresses or bits to which no registers are allocated.

For the use of the specific areas of data memory, see the following:
- 4.4 for the general purpose register area
- 4.6 for the stack memory area
- 5.7.5 for the display data memory
- CHAPTER 5 for the peripheral hardware

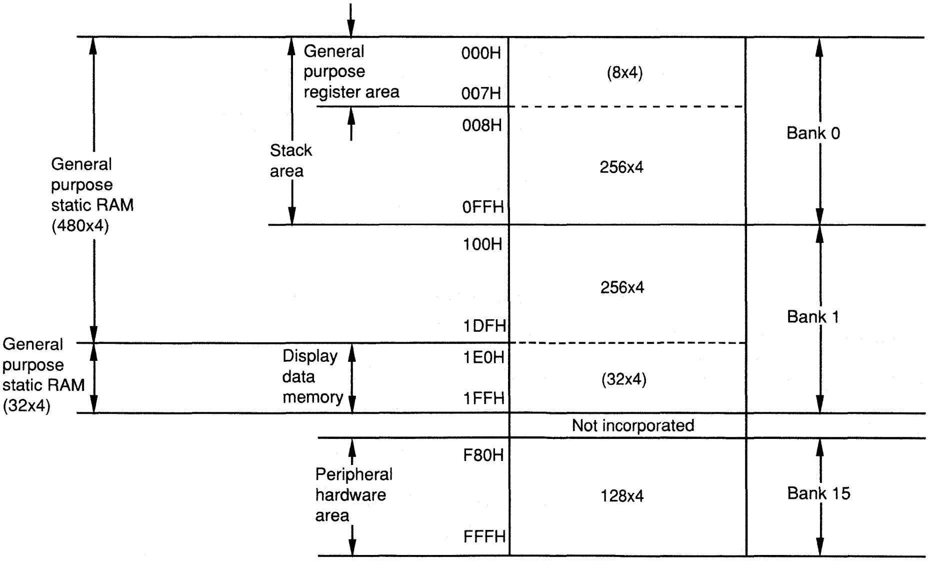


**Figure 4.3–1 Data Memory Map**

The data memory is undefined when the system is reset during normal operation mode (not standby), because of possible memory access at the time of reset. Be sure to initialize it too (clear RAM), normally at the program start.

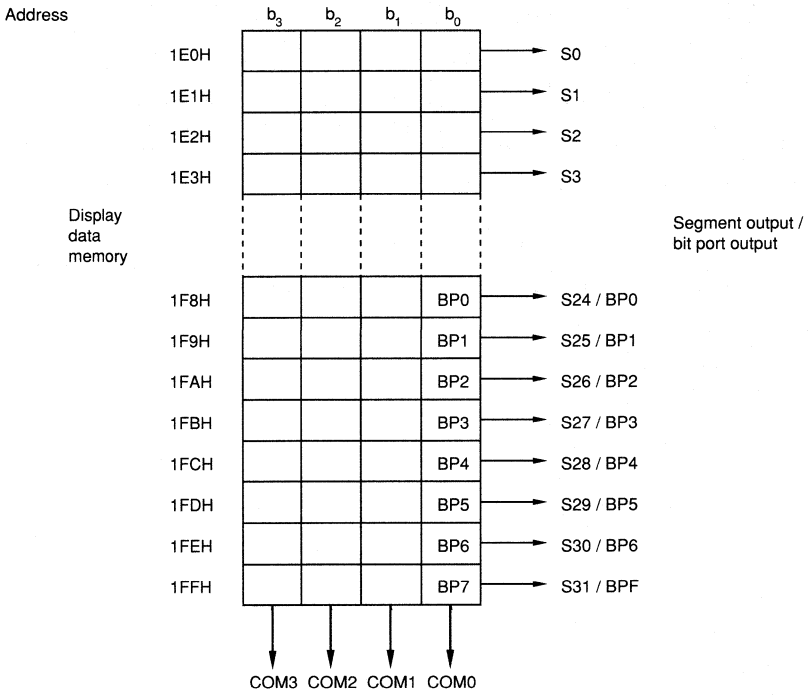


**Figure 4.3–2 Display Data Memory Configuration**

The display data memory is handled in 1- or 4-bit units.
Caution: The display data memory cannot be handled in 8-bit units.
Example: To clear 1E0H–1FFH display data memory.

```
        SET1    MBE
        SEL     MB1
        MOV     HL, #0E0H
        MOV     A, #00H
LOOP:   MOV     @HL, A      ;   Display data memory is cleared in4-bit units at a time
        INCS    L
        BR      LOOP
        INCS    H
        BR      LOOP
```

### 4.4 General Purpose Registers – Eight x Four Bits

The general purpose registers are eight 4-bit registers (B, C, D, E, H, L, X, and A) mapped in specific addresses of the data memory. Every general purpose register is handled in 4-bit units; register pairs BC, DE, HL, and XA are also used for 8-bit manipulation. In addition to DE and HL, registers D and L are also paired (DL), and the three register pairs can be used for data pointers. The general purpose register area can be addressed and accessed as normal RAM regardless of whether or not it is used for registers.

| | |
|---|---|
| X<br>01H | A<br>00H |
| H<br>03H | L<br>02H |
| D<br>05H | E<br>04H |
| B<br>07H | C<br>06H |

**Figure 4.4–1 General Purpose Register Configuration (When 4-bit processing is performed)**

| |
|---|
| XA<br>00H |
| HL<br>02H |
| DE<br>04H |
| BC<br>06H |

**Figure 4.4–2 General Purpose Register Configuration (When 8-bit processing is performed)**

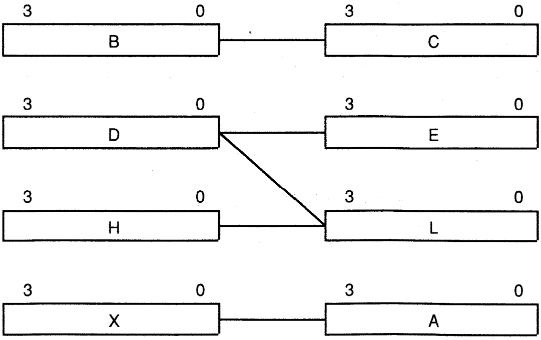


**Figure 4.4–3 Register Pair Configuration**

### 4.5 Accumulator

The μPD753XX uses the A register and XA register pair for accumulators. The A register is used as the main register during execution of 4-bit data processing instructions; the XA register pair is used as the main register pair during execution of 8-bit data processing instructions.

The carry flag (CY) is used for a bit accumulator during execution of bit manipulation instructions.

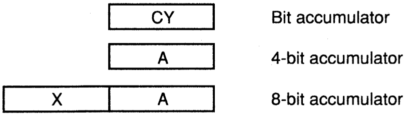


**Figure 4.5–1 Accumulator**

### 4.6 Stack Pointer (SP) – Eight Bits

The μPD753XX uses general purpose RAM for stack memory (LIFO). The stack pointer (SP) is an 8-bit register which holds top address information of the stack area.
The stack area addresses are 000H–0FFH of memory bank 0 regardless of how MBE and MBS are set.
SP is decremented before data is saved in the stack memory (write operation); it is incremented after data is restored from the stack memory (read operation).
Figs. 4.6–2 to 4.6–4 shows data saved in and restored from the stack memory when stack operations are performed.
An initial value is set in SP by using an 8-bit memory operation instruction to determine the stack area to be used. The SP contents can also be read.
SP0 is always set to 0.
It is recommended that the initial value of SP should be set to 00H so that the stack area be used is starting at the most significant address of data memory bank 0 (0FFH).
When the $\overline{\text{RESET}}$ signal is generated, the SP contents become undefined. Be sure to initialize SP to the desired value at the start of the program.

Example: To initialize SP.

```
SEL MB15        ;   or CLR1 MBE
MOV XA, #00H
MOV SP, XA      ;   SP← 00H
```

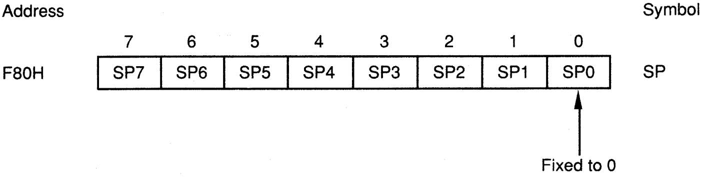
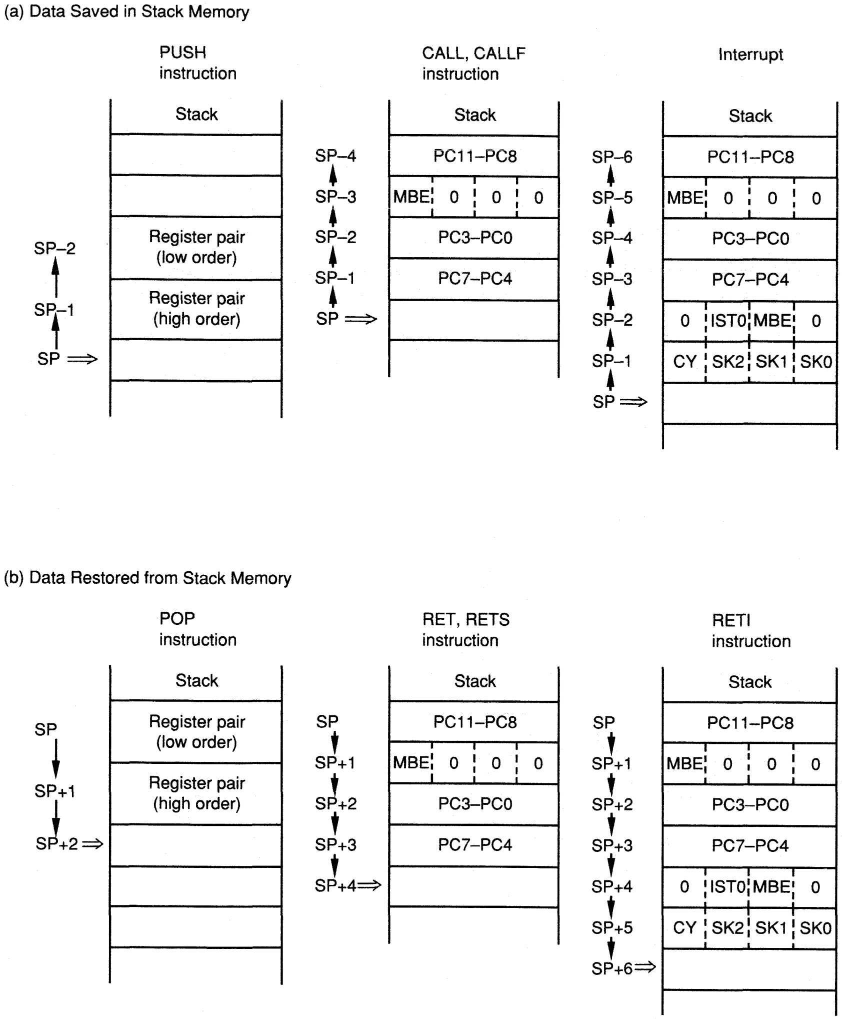


**Figure 4.6–1 Stack Pointer Configuration**



(a) Data Saved in Stack Memory

(b) Data Restored from Stack Memory

**Figure 4.6–2 Data Saved / Restore of Operation (μPD75304)**

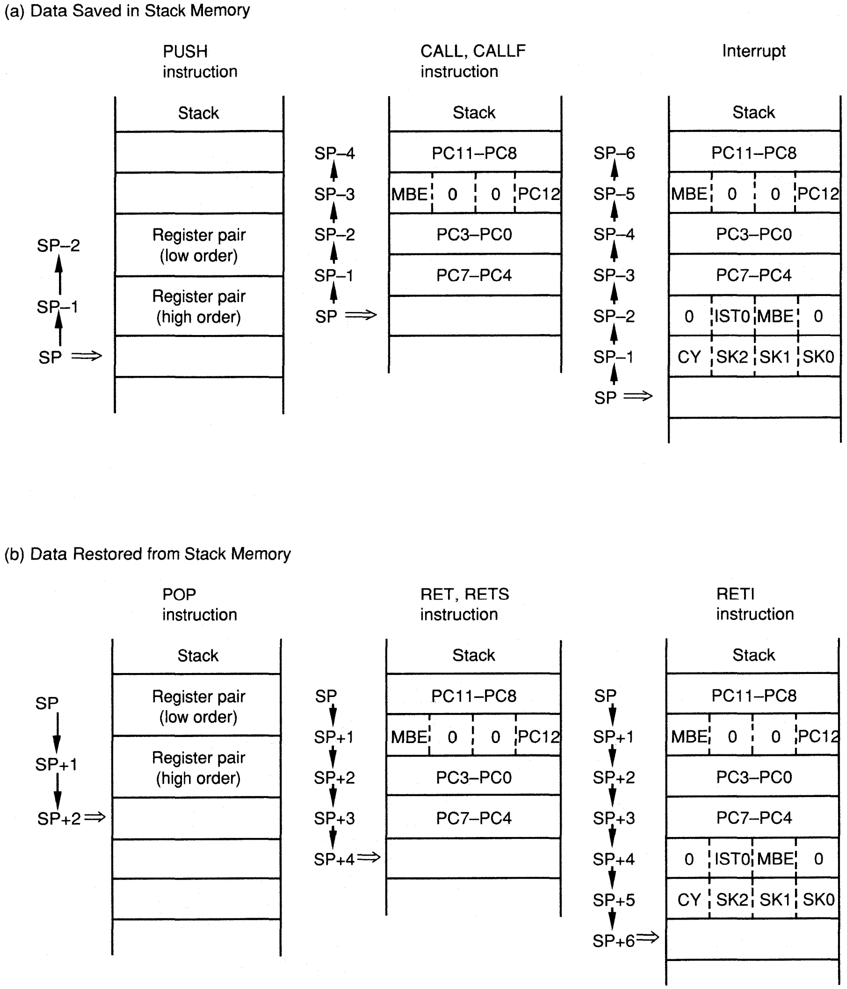
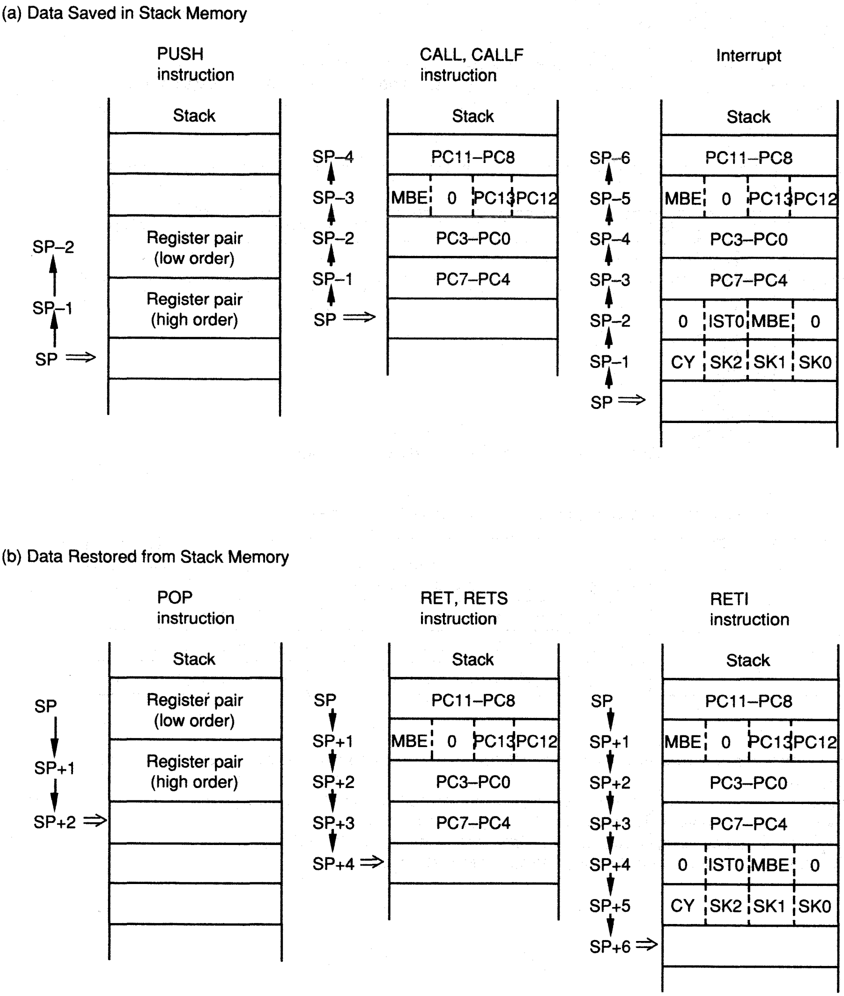


**Figure 4.6–3 Data Saved / Restored in Stack Operation (µPD75306, 75308)**


(a) Data Saved in Stack Memory

(b) Data Restored from Stack Memory

**Figure 4.6–3 Data Saved / Restored in Stack Operation (μPD75312, 75316)**

### 4.7 Program Status Word (PSW) – Eight Bits

The program status word (PSW) consists of flags closely related to processor operation.
PSW is mapped in data memory addresses FB0H and FB1H. Two bits of address FB0H can be operated by using a memory operation instruction.

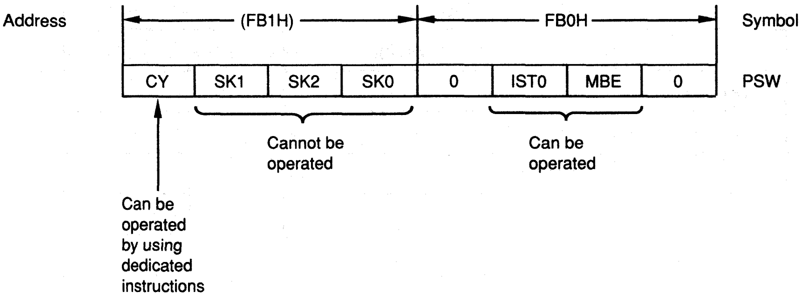


**Figure 4.7–1 Program Status Word Configuration**

**Table 4.7–1 PSW Flags Saved and Restored when Stack Operation is Performed**

| | | Saved or restored flag |
|---|---|---|
| Save | During CALL, CALLF instruction execution | MBE is saved |
| | During hardware interrupt | All PSW bits are saved |
| Restore | During RET, RETS instruction execution | MBE is restored |
| | During RETI instruction execution | All PSW bits are restored |

(1) Carry flag (CY)

The carry is a 1-bit flag that shows the occurence of overflow or underflow during execution of an instruction involving carry (ADDC or SUBC).

The carry flag also serves as a bit accumulator. Boolean algebra operation is performed between the bit accumulator and data memory specified by a bit address. The result can be stored in the bit accumulator.

The carry flag is operated by using dedicated instructions independently of other PSW bits.

When the $\overline{\text{RESET}}$ signal is generated, the carry flag becomes undefined.

**Table 4.7–2 Carry Flag Operation Instructions**

| | Instructions (mnemonic) | Carry flag processing |
|---|---|---|
| Instructions dedicated to carry flag operation | SET1 CY<br>CLR1 CY<br>NOT1 CY<br>SKT CY | CY is set to 1<br>CY is cleared<br>CY is inverted<br>Skip if CY is set to 1 |
| Bit Boolean instructions | AND1 CY, mem*.bit<br>OR1 CY, mem*.bit<br>XOR1 CY, mem*.bit | The specified bit and CY are ANDed, ORed, or XORed together |
| Interrupt service | During interrupt execution | CY is saved in stack memory in parallel with other PSW bits (eight bits) |
| | RETI | CY is restored from stack memory in parallel with other PSW bits |

Remarks: mem*.bit indicates any of the following three bit 3 manipulation addressing modes

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

Example: To AND bit 3 of address 3H and P33 together and set the result in CY.

```
SET1   CY            ;  CY←1
CLR1   MBE           ;  Or SEL MB0
SKT    3FH. 3        ;  Skip if bit 3 of address 3FH is set to 1.
CLR1   CY            ;  CY←0
AND1   CY, PORT3.3   ;  CY←CY ∧ P33
```

(2) Skip flags (SK2, SK1, and SK0)
The skip flags store the skip state and are automatically set or reset when the CPU executes instructions. The user cannot directly use the flags as operands.

(3) Interrupt status flag (IST0)
The interrupt status flag stores the current status of processing being performed. (For details, see Table 6.3–2.)

**Table 4.7–3 Interrupt Status Flag Indication Contents**

| IST0 | Status of processing being performed | Processing contents and interrupt control |
|---|---|---|
| 0 | Status 0 | During normal program processing.<br>Every interrupt can be acknowledged. |
| 1 | Status 1 | During interrupt processing.<br>No interrupt must be acknowledged. |

If an interrupt is acknowledged, the IST0 contents are saved in stack memory as a PSW bit, then automatically IST0 is set to 1. When an RETI instruction is executed, IST0 is set to 0.
The interrupt status flag can be operated by using a memory operation instruction. The current status of processing can also be changed under program control.

Caution: Before operating the flag, be sure to execute a DI instruction to disable interrupts. After operating the flag, execute an EI instruction to enable interrupts.

(4) Memory bank enable flag (MBE)

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

MBS can be set or reset at any time by using a bit manipulation instruction regardless of memory bank setting.

```
Example: SET1   MBE   ;   MBE← 1
         CLR1   MBE   ;   MBE← 0
```

When MBE is set to 1, the data memory address space is exceeded and all the data memory space can be addressed.
When MBE is reset to 0, the data memory address space is fixed regardless of how MBS is set. (see Fig. 3.1–2.)
When the $\overline{\text{RESET}}$ signal is generated, the contents of program memory address 0 bit 7 are set and MBE is initialized automatically.
When vectored interrupt service is made, bit 7 of the corresponding vector address table is set and the MBE state during interrupt service is set automatically.
During interrupt service, normally MBE is set to 0 and the general purpose RAM of memory bank 0 is used.

### 4.8 Bank Selection Register (BS)

The memory bank selection register (MBS) for selecting a memory bank is mapped in the bank selection register (BS). The low-order four bits of BS are fixed to 0
MBS is set by using the SEL MBn instructions.
BS can be saved in and restored from the stack area in 8-bit units by using PUSH BS instructions.

| MBS | | | | | | | | Symbol |
|---|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| MBS3 | MBS2 | MBS1 | MBS0 | 0 | 0 | 0 | 0 | BS |

**Figure 4.8–1 Bank Selection Register Configuration**

(1) Memory bank selection register (MBS)

The memory bank selection register (MBS), consisting of four bits, stores high-order 4-bit address information of a 12-bit data memory address. The memory bank to be accessed is specified according to the register contents. However, the µPD753XX allows the user to select bank 0, bank 1, or bank 15 only.
MBS is set by using the SEL MBn instruction (where n is 0, 1, or 15).
The address range applied according to how MBE and MBS are set is as shown in Fig. 3.1–2.
When the $\overline{\text{RESET}}$ signal is generated, MBS is initialized to 0.

# 5 PERIPHERAL HARDWARE FUNCTIONS

## 5.1 Digital Input / Output Ports

The µPD753XX adopts memory mapped I/O. All input / output ports are mapped in the data memory space.
Bit 0 in addresses 1F8H–1FFH is used for output latches for bit port outputs BP0–BP7.
For BP0–BP7, bit port output can be switched in 4-bit units by using display mode register (LCDM) bits 6 and 7. (See Fig. 5.7–2)
1F8H–1FFH bits not used for bit port output latches can be used for display memory or general purpose RAM. These addresses can be handled in 1- or 4-bit units. These addresses cannot be handled in 8-bit unit.

| Address | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|
| 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 |
| 1F8H | – | – | – | BP0 | |
| 1F9H | – | – | – | BP1 | |
| 1FAH | – | – | – | BP2 | |
| 1FBH | – | – | – | BP3 | |
| 1FCH | – | – | – | BP4 | |
| 1FDH | – | – | – | BP5 | |
| 1FEH | – | – | – | BP6 | |
| 1FFH | – | – | – | BP7 | |

Remarks: The part of – bits can be used for general purpose RAM

**Figure 5.1–1 Data Memory Addresses of Digital Ports**

Table 5.1–2 lists the input / output port operation instructions. In addition to 4-bit input / output, 8-bit input / output and bit manipulation can be performed for PORT 4 to PORT 7 and very diverse control can be performed. BP0–BP7 are 1-bit output ports.

Example: To test the P13 state and output value to ports 4 and 5 depending on the result.

```
SKT   PORT1.3     ;   Skip if port 1 bit 3 is set to 1
MOV   XA, #18H    ;   XA← 18H String effect
MOV   XA, #14H    ;   XA← 14H String effect
SEL   MB15        ;   Or CLR1 MBE
OUT   PORT4, XA   ;   Ports 5,4← XA
```

```
Example: SET1   PORT 4. @L  ;   The port 4–7 bit specified by using the L register is set to 1.
```

Example: To output 1 to BP0.

```
SET1  MBE
SEL   MB1         ;   Memory bank 1 is selected
SET1  BP0         ;   BP0← 1
```

5.1.1 Types, features, and configurations of digital input / output ports

Table 5.1–1 lists the digital input / output ports.
Figs. 5.1–1 to 5.1–5 shows the port configurations.

**Table 5.1–1 Types and Features of Digital Ports**

| Port (abbreviation) | Function | Operation and features | Remarks |
|---|---|---|---|
| PORT 0 | 4-bit input | Can always be read or tested regardless of the operation mode. | Pins also used for INT4, $\overline{\text{SCK}}$ SO/SB0, SI/SB1. Pins also used for INT0-2 and TI0. |
| PORT 1 | | | |
| PORT 3(Note 1) | 4-bit input / output | Can be placed in input or output mode in 1-bit units. | Pins also used for LCDCL SYNC and MD0–MD3, (Note 2) |
| PORT 6 | | | Pins also used for KR0–KR3. |
| PORT 2 | | Can be placed in input or ouput mode in 4-bit units. Ports 6 and 7 can be paired for data input / output in 8-bit units | Port 2 pins are also used for PTO0, PCL, and BUZ. |
| PORT 7 | | | Pins also used for KR4–KR7. |
| PORT 4(Note 1)<br>PORT 5(Note 1) | 4-bit input / output (N-channel open drain 10 Volts) | Can be placed in input or output mode in 4-bit units. Ports 4 and 5 can be paired for data input / output in 8-bit units. | Internal pull-up resistor can be specified in 1-bit units by using mask option. |
| BP0–BP7 | 1-bit output | Data is output in 1-bit units. The BP0–BP7 pins are also used as output pins (S24-S31) for the LCD driving segment signal. BP0-BP7 and S24-S31 can be changed by using software. | The capacity of drive is very small. Used for CMOS load drive. |

Note 1: An LED can be driven directly.
Note 2: PORT 3 is also used for MD0-MD3 pin only in µPD75P308 and µPD75P316.

The P10 pin is also used for an external vectored interrupt input pin (input with a noise removal circuit). (For details, see 6.3.) The BP0-BP7 pins are also used for driving LCD segment signal output pins (S24-S31);they are changed in 4- or 8-pin units by using display mode register (LCDM) bits 6 and 7. BP0-BP7 are 1-bit output ports. Data for bit 0 in each of the display data memory addresses 1F8H-1FFH is output. (See 5.7.5.)
When the $\overline{\text{RESET}}$ signal is generated, port 2-port 7 output latches are cleared, the output buffers are turned off, and input mode is entered.

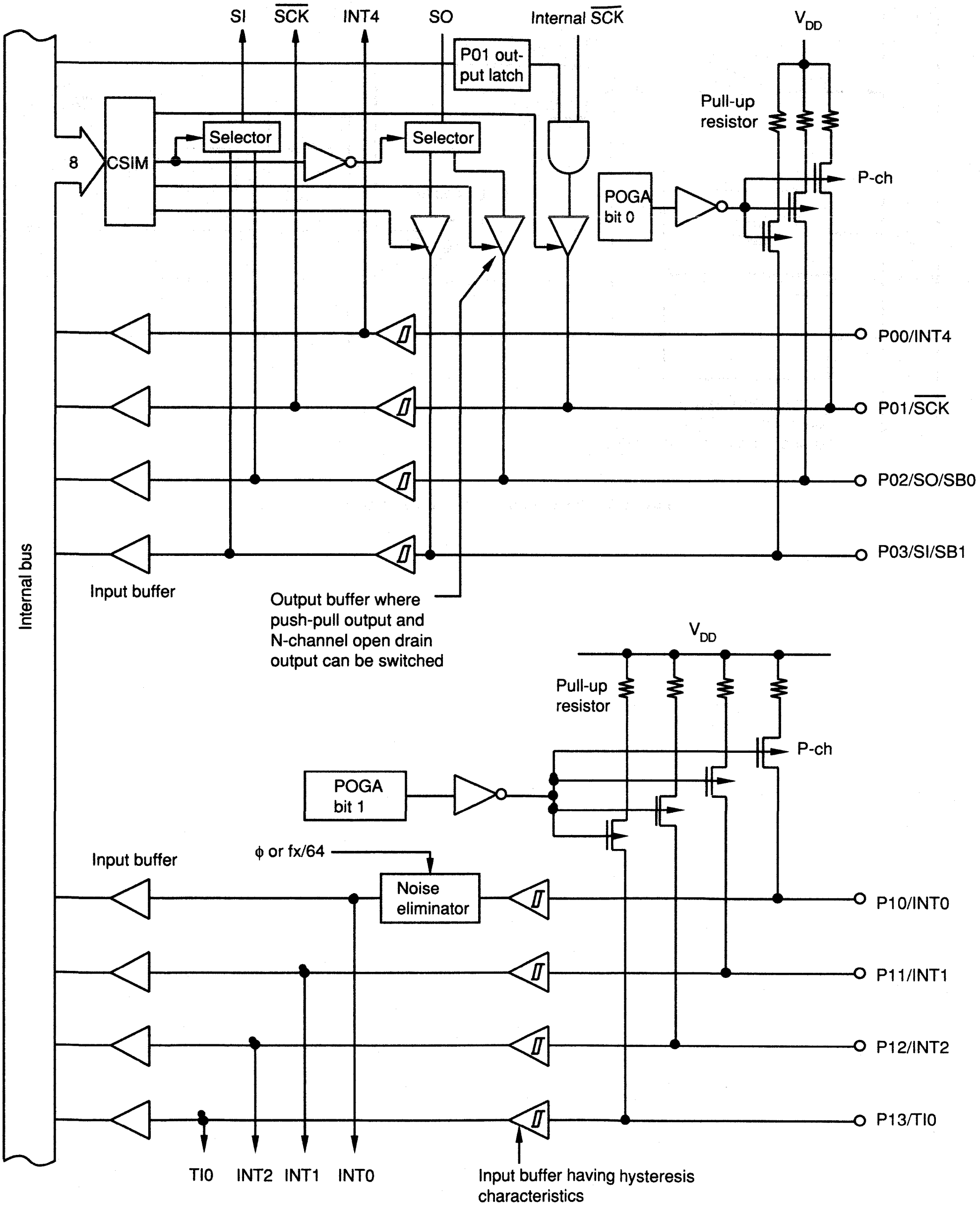


**Figure 5.1–2 Configuration of Ports 0 or 1**

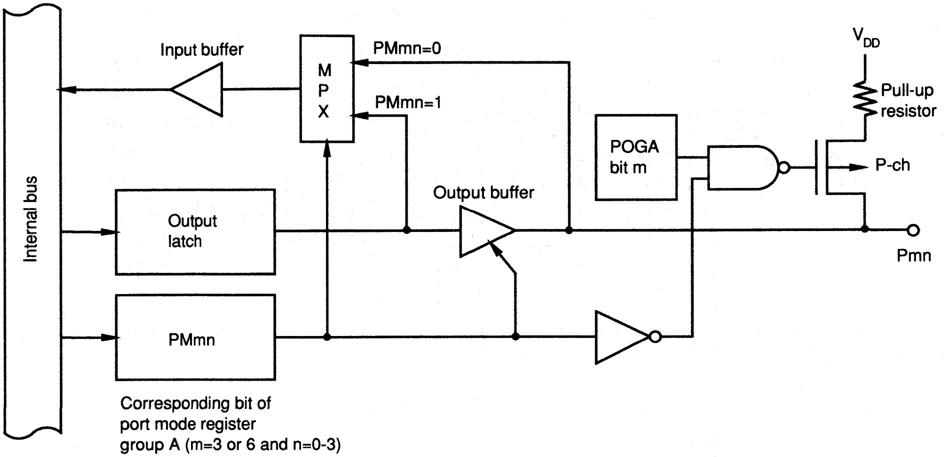


**Figure 5.1–3 Configuration of port 3n or 6n (n = 0–3)**

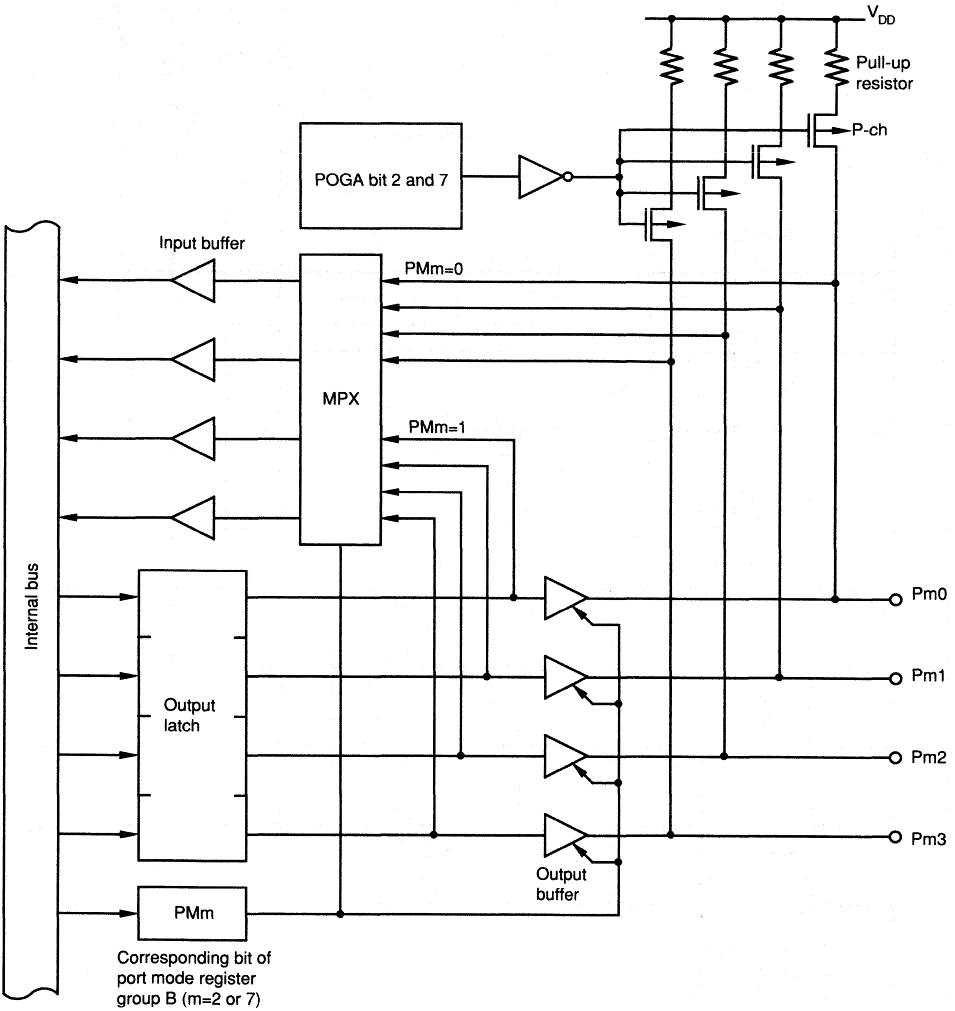


**Figure 5.1–4 Configuration of Port 2 or 7**

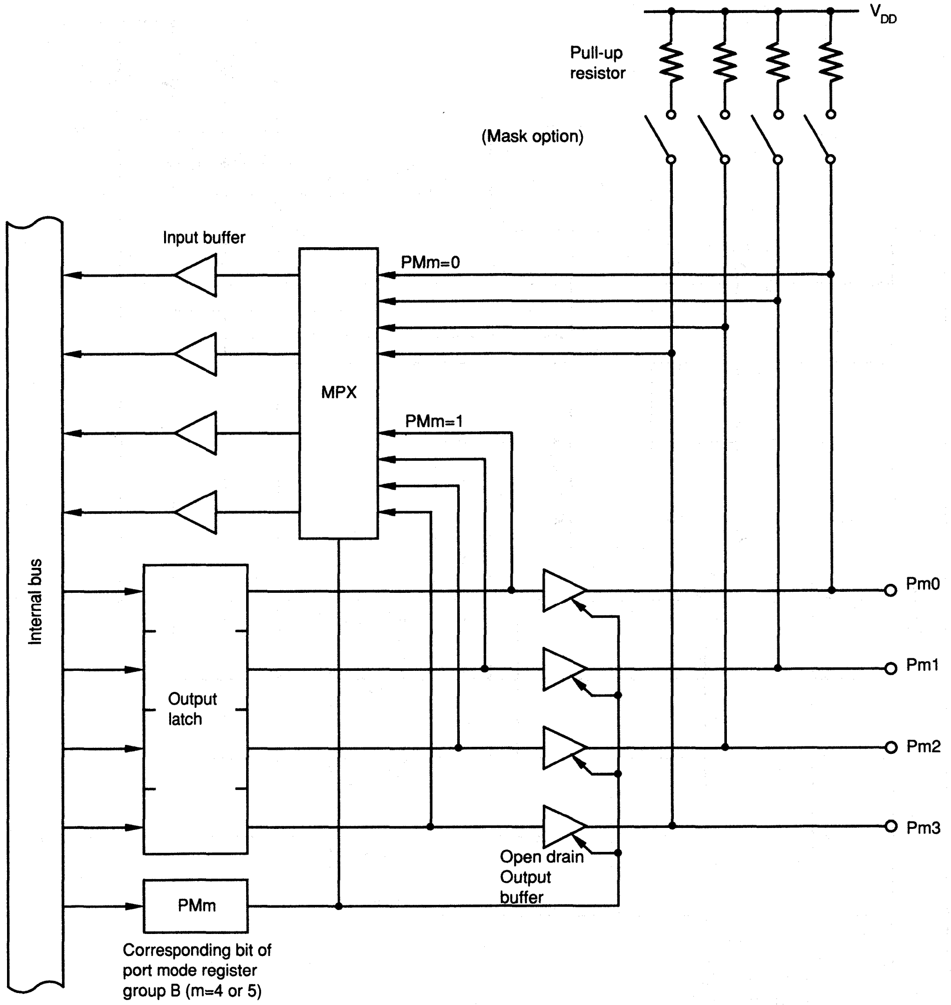


**Figure 5.1–5 Configuration of Port 4 or 5**

5.1.2 Input / output mode setting

The input or output mode of each input/output ports is set by using the port mode register as shown in Fig. 5.1–6. Each bit of ports 3 and 6 can be placed in input or output mode individually by using port mode register group A (PMGA). Each of ports 2, 4, 5, and 7 (four bits each) is placed in input or output mode individually by using port mode register group B (PMGB).
Each port serves as an input port when the corresponding port mode register bit is set to = 0 and serves as an output port when set to 1.
Since the output latch contents are applied to the output pins at the same time the output mode is selected by setting the port mode register, the output latch contents must previously be rewitten to the required value before the output mode is set.
Port mode register groups A and B are set by using 8-bit memory operation instructions
When the $\overline{\text{RESET}}$ signal is generated, all bits of the port mode registers are cleared; thus the output buffers are turned off, and all ports are placed in input mode.

Example: To use P30, P31, P62, and P63 for input pins and P32, P33, P60, and P61 for output pins.

```
CLR1   MBE        ;   Or SEL MB15
MOV    XA, #3CH
MOV    PMGA, XA
```

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 |

| | | |
|---|---|---|
| | PM3n<br>PM6n | P3n, P6n pin input / output specification (n = 0–3) |
| | 0 | Input mode (output buffer off) |
| PMGA | 1 | Output mode (output buffer on) |

Port mode register group B

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

| | |
|---|---|
| PMn | Port n input / output specification (n = 2, 4, 5, or 7) |
| 0 | Input mode (output buffer off) |
| 1 | Output mode (output buffer on) |

–: May be 0 or 1.

**Figure 5.1–6 Port Mode Register Formats**

5.1.3 Digital input / output port operation instructions

Since all input / output ports incorporated in the μPD753XX are mapped in the data memory space, all the data memory operation instructions are applicable. Table 5.1–2 lists the data memory operation instructions particulary useful for input / output pin operation and their application ranges.

(1) Bit operation instructions

The specific address bit direct addressing (fmem. bit) and specific address bit register indirect addressing (pmem. @L) modes are applicable to digital input / output ports PORT 0-PORT 7. The port bits can be operated at any time as desired regardless of how MBE and MBS are set.

Example: To OR P50 and P41 together and output the result to P61.

```
        SET1   CY            ;  CY ← 1
        AND1   CY, PORT5.0   ;  CY ← CY∧P50
        OR1    CY, PORT4.1   ;  CY ← CY∨P41
        SKT    CY
        BR     CLRP
        SET1   PORT6.1       ;  P61 ← 1
          :
          :
  CLRP: CLR1   PORT6.1       ;  P61 ← 0
```

(2) 4-bit operation instructions

In addition to the IN and OUT instructions, all 4-bit memory operation instructions such as MOV, XCH, ADDS, and INCS can be used. However, memory bank 15 must have been selected before executing an instruction.

Example 1: To output the accumulator contents to port 3.

```
        SEL    MB15          ;  Or CLR1 MBE
        OUT PORT3, A
```

Example 2: To add the accumulator value to the data output to port 5 and output the result.

```
        SET1   MBE
        SEL    MB15
        MOV    HL, #PORT5
        ADDS   A, @HL        ;  A ← A+PORT5
        NOP
        MOV    @HL, A        ;  PORT5 ← A
```

Example 3: To test whether or not data in port 4 is greater than the accumulator value.

```
        SET1   MBE
        SEL    MB15
        MOV    HL, #PORT4
        SUBS   A, @HL        ;  A<PORT4
        BR     NO            ;  NO
                             ;  YES
```

(3) 8-bit operation instructions

In addition to the IN and OUT instructions, the MOV, XCH and SKE instructions can be used for ports 4 and 5 where 8-bit operation (manipulation) can be performed. As with the 4-bit operation instructions, memory bank 15 must have been selected before executing the instruction.

Example: To output data in the BC register pair to the output ports specified by the 8-bit data input from ports 4 and 5.

```
        SET 1  MBE
        SEL    MB15
        IN     XA, PORT 4    ;  XA ← ports 5 and 4
        MOV    HL, XA        ;  HL ← XA
        MOV    XA, BC        ;  XA ← BC
        MOV    @HL, XA       ;  Port (L) ← XA
```

**Table 5.1–2 Input / Output Port Pin Operation Instruction List**

<table>
<tr><th>PORT<br>Instruction</th><th>PORT<br>0</th><th>PORT<br>1</th><th>PORT<br>2</th><th>PORT<br>3</th><th>PORT<br>4</th><th>PORT<br>5</th><th>PORT<br>6</th><th>PORT<br>7</th><th>BIT•PORT<br>0–7</th></tr>
<tr><td>IN A, PORTn<br>(Note 1)</td><td colspan="8">O</td><td>–</td></tr>
<tr><td>IN XA, PORTn<br>(Note 1)</td><td colspan="2">–</td><td colspan="2">–</td><td colspan="2">O</td><td colspan="2">O</td><td>–</td></tr>
<tr><td>OUT PORTn. A<br>(Note 1)</td><td>–</td><td>–</td><td colspan="6">O</td><td>MOV mem, A<br>(Notes 3 and 4)</td></tr>
<tr><td>OUT PORTn. XA<br>(Note 1)</td><td colspan="2">–</td><td colspan="2">–</td><td colspan="2">O</td><td colspan="2">O</td><td>–</td></tr>
<tr><td>SET1 PORTn. bit</td><td>–</td><td>–</td><td colspan="6">O</td><td>SET1 BPn<br>(Note 3)</td></tr>
<tr><td>SET1 PORTn. @L<br>(Note 2)</td><td>–</td><td>–</td><td colspan="6">O</td><td>–</td></tr>
<tr><td>CLR1 PORTn. bit</td><td>–</td><td>–</td><td colspan="6">O</td><td>CLR1 BPn<br>(Note 3)</td></tr>
<tr><td>CLR1 PORTn. @L<br>(Note 2)</td><td>–</td><td>–</td><td colspan="6">O</td><td>–</td></tr>
<tr><td>SKT PORTn. bit</td><td colspan="8">O</td><td>SKT BPn<br>(Note 3)</td></tr>
<tr><td>SKT PORTn. @L<br>(Note 2)</td><td colspan="8">O</td><td>–</td></tr>
<tr><td>SKF PORTn. bit</td><td colspan="8">O</td><td>SKF BPn<br>(Note 3)</td></tr>
<tr><td>SKF PORTn. @L<br>(Note 2)</td><td colspan="8">O</td><td>–</td></tr>
<tr><td>AND1 CY, PORTn. bit</td><td colspan="8">O</td><td>AND1 CY, @H+BPn<br>(Notes 3 and 5)</td></tr>
<tr><td>AND1 CY, PORTn. @L<br>(Note 2)</td><td colspan="8">O</td><td>–</td></tr>
<tr><td>OR1 CY, PORTn. bit</td><td colspan="8">O</td><td>OR1 CY, @H+BPn<br>(Notes 3 and 5)</td></tr>
<tr><td>OR1 CY, PORTn. @L<br>(Note 2)</td><td colspan="8">O</td><td>–</td></tr>
<tr><td>XOR1 CY, PORTn. bit</td><td colspan="8">O</td><td>XOR1 CY, @H+BPn<br>(Notes 3 and 5)</td></tr>
<tr><td>XOR1 CY, PORTn. @L<br>(Note 2)</td><td colspan="8">O</td><td>–</td></tr>
</table>

Note 1: MBE = 0 or (MBE = 1 and MBS = 15) must have been set before executing the instruction.
2: The low-order two bits of address and the bit address are specified indirectly by using the L register.
3: (MBE = 1 and MBS = 1) must have been set before executing the instruction.
4: Accumulator A bit 0 corresponds to BPn.
5: Write FH into the H register.

### 5.1.4 Digital input / output port operation

When data memory operation instructions are executed for the digital input / output ports, port (pin) operation varries according to which mode (input or output) is set. (See table 5.1–3.) This is because as understood from the input / output port structure, data read into the internal bus is data for each pin in the input mode and output latch data in the output mode.

(1) Operation when input mode is set

Pin data is operated when executing a test instruction such as SKT or an instruction to read 4- or 8-bit port data into the internal bus (IN, OUT, operation, or comparison instruction).

When an instruction to transfer the accumulator contents (four or eight bits) to a port or ports is executed (OUT or MOV instruction), the accumulator data is latched in the output latches. The output buffers remain off.

When an XCH instruction is executed, pin data is input to the accumulator and the accumulator data is latched in the output latches. The output buffers remain off.

When an INCS instruction is executed, data resulting from adding 1 to 4-bit pin data is latched in the output latches. The output buffer remain off.

When an instruction to rewrite data memory bitwise is executed, such as SET1, CLR1, or SKTCLR, (the specifed bit output latch can be rewritten as specified by instruction), but the contents of the output latch for other bits become undefined.

(2) Operation when output mode is set

When a test instruction or an instruction to read 4- or 8-bit port data into the internal bus is executed, the output latch contents are operated.

When an instruction to transfer the accumulator contents (four or eight bits) is executed, output latch data is rewritten and at the same time as output from the pins.

When an XCH instruction is executed, the output latch contents are transferred to the accumulator, and the accumulator contents are held in the output latches and is output from the pins.

When an INCS instruction is executed, data resulting from incrementing the output latch contents by one is held in the output latches and output from the pins.

When a bit output instruction is executed, the specified output latch bit is rewritten and output from the pin.

**Table 5.1–3 Operation When Input / Output Ports are Used**

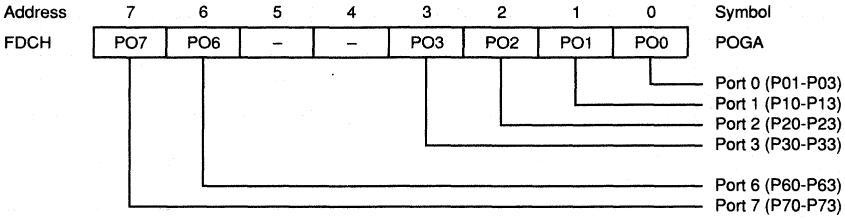
| Executed instruction | Port (pin) operation | |
|---|---|---|
| | Input mode | Output mode |
| SKT ①<br>SKF ① | Pin data is tested. | Output latch is data tested. |
| AND1 CY, ①<br>OR1 CY, ①<br>XOR1 CY, ① | Operation is performed between pin data and CY. | Operation is performed between output latch data and CY. |
| IN A, PORTn<br>IN XA, PORTn<br>MOV A, @HL<br>MOV XA, @HL | Pin data is transferred to accumulator. | Output latch data is transferred to accumulator |
| ADDS A, @HL<br>ADDC A, @HL<br>SUBS A, @HL<br>SUBC A, @HL<br>AND A, @HL<br>OR A, @HL<br>XOR A, @HL | Operation is performed between pin data and accumulator. | Operation is performed between output latch data and accumulator. |
| SKE A, @HL | Pin data and accumulator contents are compared. | Output latch data and accumulator contents are compared. |
| OUT PORTn, A<br>OUT PORTn, XA<br>MOV @HL, A<br>MOV @HL, XA | Accumulator data is transferred to output latches. (Output buffers remain off.) | Accumulator data is transferred to output latches and output from the pins. |
| XCH A, PORTn<br>XCH XA, PORTn<br>XCH A, @HL<br>XCH XA, @HL | Pin data is transferred to accumulator and accumulator data is transferred to output latches.<br>(Output buffers remain off.) | Data is exchanged between output latches and the accumulator. |
| INCS PORTn<br>INCS @HL | Data resulting from incrementing pin data by one is latched in output latches. | The output latch contents are incremented by one. |
| SET1 ①<br>CLR1 ①<br>SKTCLR ① | The specified bit output latches are rewritten as specified by the instruction, but the output latch contents for other bits are undefined. | The output pin state is changed according to the instruction. |

① : Denotes two addressing modes PORTn. bit and PORTn. @L.

### 5.1.5 Internal pull-up resistors

Pull-up resistors can be incorporated in the µPD753XX port pins except P00 or BP0-BP7. Internal pull-up resistors are specified by using software or mask option.

Pull-up register specification register group A



|  | Specification |
|---|---|
| 0 | Pull-up resistor is not incorporated. |
| 1 | Pull-up resistor is incorporated. |

**Figure 5.1–7 Pull-Up Register Specification Register Format**

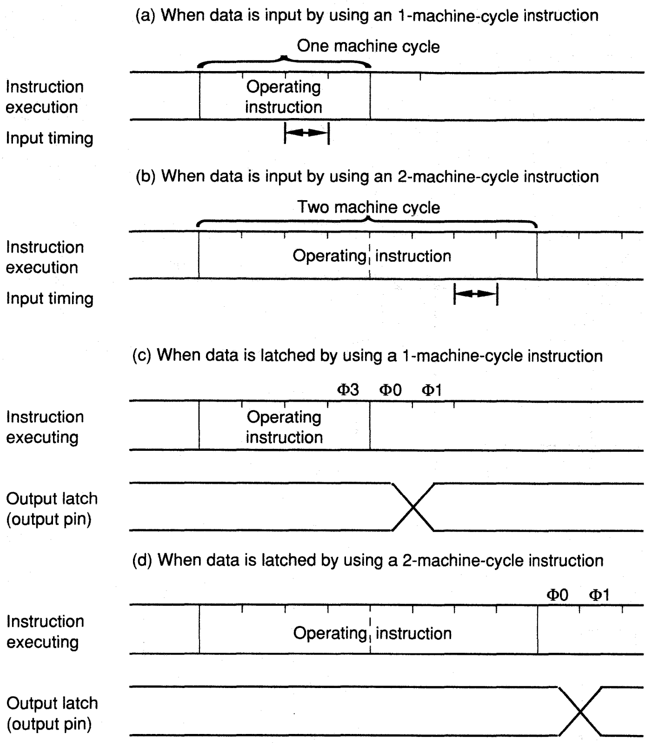
**Table 5.1–4**

| Port (pin names) | Internal pull-up resistor specification method | Specification Bit in POGA |
|---|---|---|
| Port 0 (P01-P03) (Note 1) | Software is used for internal resistor specification in 3-bit units. | Bit 0 |
| Port 1 (P10-P13) | Software is used for internal resistor specification in 4-bit units. | 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) | Mask option is used for internal resistor specification in 1-bit units. | – |
| Port 5 (P50-P53) |  |  |

Note: A pull-up resistor cannot be incorporated in the P00 Pin.

### 5.1.6 Digital input / output port input / output timing

Fig. 5.1–8 shows timing of data to output latch and timing of pin data or output latch data input to the internal bus.
Fig. 5.1–9 shows the ON timing at specified the internal pull-up resistor by software.


(a) When data is input by using an 1-machine-cycle instruction

(b) When data is input by using an 2-machine-cycle instruction

(c) When data is latched by using a 1-machine-cycle instruction

(d) When data is latched by using a 2-machine-cycle instruction

**Figure 5.1–8 Digital Input / Output Port Input / Output Timing**

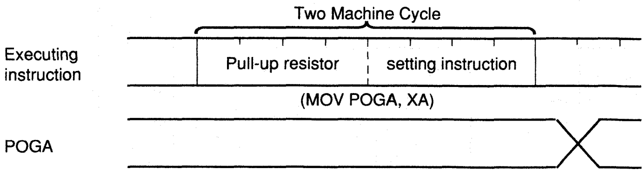


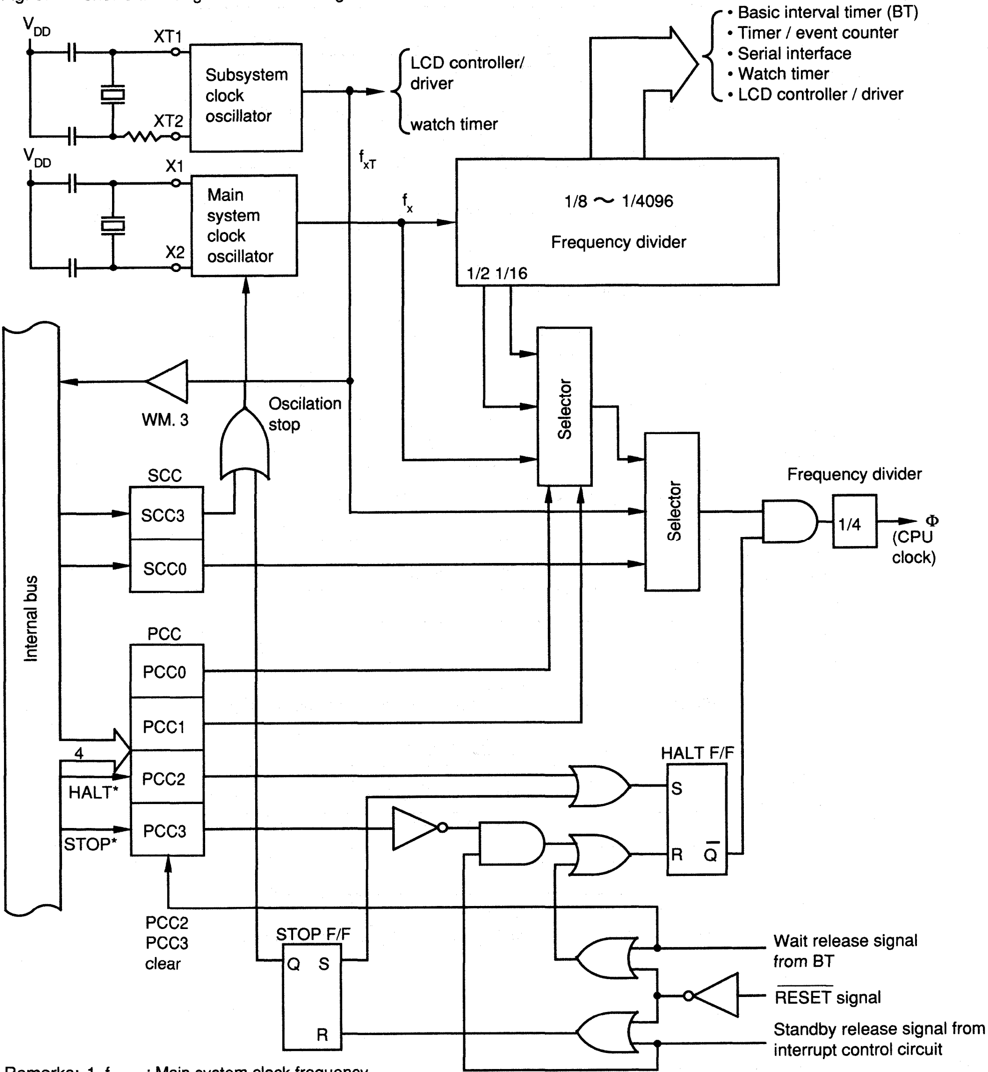
**Figure 5.1–9 ON Timing of Pull-up Resistor by using Software**

**5.2 Clock Generator**

The clock generator supplies clocks to CPU and peripheral hardware for controlling the CPU operating mode.

5.2.1 Clock generator configuration

Fig. 5.2–1 shows a clock generator block diagram.



Remarks: 1. $f_X$ : Main system clock frequency

2. $f_{XT}$ : Subsystem clock frequency

3. PCC : Processor clock control register

4. SCC : System clock control register

5. * denotes instruction execution.

**Figure 5.2–1 Clock Generator Block Diagram**

5.2.2 Clock generator function and operation

The clock generator produces the following clocks and controls CPU operating modes such as standby:
— Main system clock $f_X$
— Subsystem clock $f_{XT}$
— CPU clock φ
— Clock to peripheral hardware
The clock generator operates according to how the processor clock control register (PCC) and system clock control register (SCC) are set, as described below:
(a) When the $\overline{\text{RESET}}$ signal is generated, the minimum speed mode of the main system clock (15.3 μ/4.19 MHz) is selected. (PCC = 0 and SCC = 0)
(b) When the main system clock is selected, one of three CPU clock frequencies can be selected (0.95 μs, 1.91 μs, and 15.3 μs/ 4.19 MHz) by setting PCC.
(c) When the main system clock is selected, the standby mode (STOP or HALT) can be used.
(d) Subsystem clock is selected by setting SCC.0 and operation can be performed in a very low-speed and low current consumption (122 μs/32.768 kHz). In this case, the PCC setup value does not affect the CPU clock.
(e) When the subsystem clock is selected, main system clock oscillation can be stopped by setting SCC.3. The HALT mode can also be used, but the STOP mode cannot be used. (Subsystem clock oscillation cannot be stopped.
(f) The main system clock is divided to generate a clock supplied to peripheral hardware. The subsystem clock can be supplied directly only to the watch timer. Thus, the watch function and the LCD controlller and buzzer output function which operate using the watch timer clock can also continue operation in the standby mode.
(g) When subsystem clock is selected, the watch timer and LCD controller can continue normal operation. When the main system clock is stopped, other hardware devices than the watch timer or LCD controller cannot be used because they operate according to the main system clock.
(1) Processor clock control register (PCC)

PCC consists of four bits; the low-order two bits are used to select CPU clock φ, and the high-order two bits are used to control CPU operation mode. (See Fig. 5.2-2.)
When bit 3 or 2 is set to (1), the standby mode is set. When the standby mode is released by the standby mode release signal, automatically bits 3 and 2 are cleared and the normal operating mode is entered. (For details, see Paragraph 7).
The low-order two bits of PCC are set by using a 4-bit memory operation instruction (the high-order two bits are set 0).
Bits 3 and 2 are set to (1) by using STOP and HALT instructions respectively.
The STOP and HALT instructions can always be executed independently of the MBE contents.
CPU clock can be selected only during main system clock operation. When the μPD753XX is operated on the subsystem clock, the low-order two bits of PCC becomes invalid, and the CPU clock is fixed to $f_{XT}/4$. The stop instruction is also enabled only during main system operation.

Example 1: To set a machine cycle to 0.95 μs (4.19) MHz).

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

Example 2: To set STOP mode. (Be sure to enter a NOP instruction following the STOP or HALT instruction.)

```
        STOP
        NOP
```

When the $\overline{\text{RESET}}$ signal is generated, PCC is cleared.

3

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

CPU clock selection bits

| | | SCC = 0<br>The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz | | SCC = 1<br>The value enclosed in parentheses is applied when $f_{XT}$ = 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 μs | $\Phi = f_{XT}/4$ (8.192 kHz) | 122 μs |
| 0 | 1 | Undefined | – | | |
| 1 | 0 | $\Phi = f_X/8$ (524 kHz) | 1.91 μs | | |
| 1 | 1 | $\Phi = f_X/4$ (1.05 MHz) | 0.95 μs | | |

$f_X$ : Main system clock oscillator output frequency

$f_{XT}$: Subsystem clock oscillator output frequency

CPU operation mode control bits

| | | |
|---|---|---|
| 0 | 0 | Normal operating mode |
| 0 | 1 | HALT mode |
| 1 | 0 | STOP mode |
| 1 | 1 | Undefined |

**Figure 5.2-2 Processor Clock Control Register Format**

(2) System clock control register (SCC)

SCC consists of four bits; the least significant bit is used to select CPU clock, and the most significant bit is used to control (stop) main system clock oscillation. (See Fig. 5.2–3.)

SCC.0 and SCC.3 exist at the same data memory address; the bits cannot be changed at the same time. Thus, SCC.0 and SCC.3 are set by using a bit operation instruction. SCC.0 and SCC.3 can always be operated independetly of the MBE contents.

Main system clock oscillation can be stopped by setting SCC.3 only during subsystem clock operation. Main system clock oscillation is stopped by using the STOP instruction during main system clock operation.

When the RESET signal is generated, SCC is cleared.

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| FB7H | SCC3 | – | – | SCC0 | SCC |

| SCC3 | SCC0 | CPU clock slection | Main system clock oscillation |
|---|---|---|---|
| 0 | 0 | Main system clock | Oscillation is enabled. |
| 0 | 1 | Subsystem clock | |
| 1 | 0 | Undefined | |
| 1 | 1 | Subsystem clock | Oscillation is stopped. |

**Figure 5.2–3 System Clock Control Register Format**

Caution 1: Changing the system clock requires a maximum of $1/f_{XT}$ time. To stop main system clock after changing the subsystem clock, set SCC.3 after the machine cycle or cycles listed in Table 5.2–1.

2: Even if oscillation is stopped by setting SCC 3 during main system clock operation, normal STOP mode is not entered.

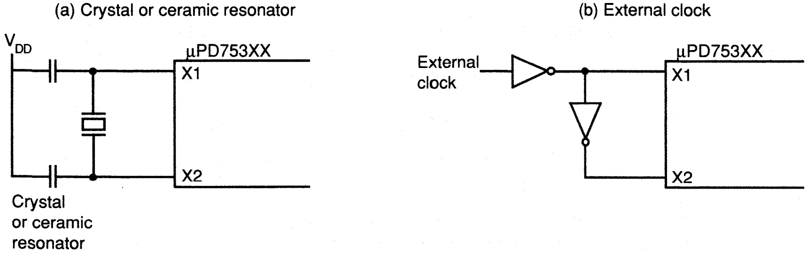
3: When SCC.3 is set to 1, X1 input is connected internally to $V_{SS}$, to avoid leakage current due to crystall oscillator. When the external clock is used in main system clock, do not set SCC.3 to 1.

4: When the system clock is switched to the subsystem clock it is necessary to disable any interrupt during the time given in table 5.2-1.

(3) System clock oscillators

The main system clock oscillator uses a crystal oscillator (4.194304 MHz standard) or cermanic oscillator connected to the X1 and X2 pins.

An external clock can also be input.


(a) Crystal or ceramic resonator

(b) External clock

**Figure 5.2–4 External Circuit to Main System Clock Oscillator**

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

- The subsystem clock oscillator uses a crystal oscillator (32.768 kHz standard) connected to the XT1 and XT2 pins.

An external clock can also be input.
The XT1 pin state can be tested by using watch mode register (WM) bit 3.

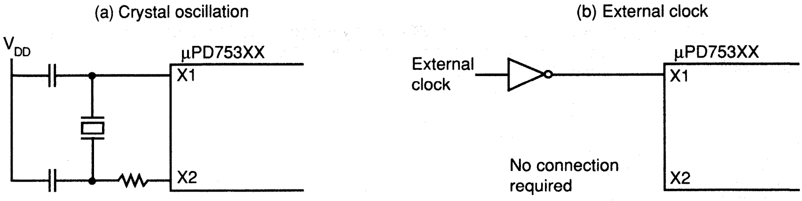


**Figure 5.2–5 External Circuit to Subsystem Clock Oscillator**

(4) Frequency divider
The frequency divider divides the main system clock oscillator output ($f_X$) and generates various clocks.

5.2.3 System clock and CPU clock setting

(1) Time required to change system and CPU clocks
The system and CPU clocks can be changed by using the low-order two bits of PCC and the least significant bit of SCC. However, this clock change is not immediatly made after the register are rewritten, and the clock before the clock change is made is used for operation during given machine cycles. Thus, to stop main system clock oscillation, a STOP instruction must be executed or SCC 3 must be set after the change time elapses.

**Table 5.2–1 Maximum Time Required to Change System and CPU Clocks**

| Setup value before change | | | Setup value after change | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCC | PCC | PCC | SCC0 | PCC1 | PCC0 | SCC0 | PCC1 | PCC0 | SCC0 | PCC1 | PCC0 | SCC0 | PCC1 | PCC0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | X | X |
| 0 | 0 | 0 | | | | 1 machine cycle | | | 1 machine cycle | | | $\frac{f_X}{64f_{XT}}$ machine cycle<br>(2 machine cycles) | | |
| | 1 | 0 | 8 machine cycles | | | | | | 8 machine cycles | | | $\frac{f_X}{8f_{XT}}$ machine cycles<br>(16 machine cycles) | | |
| | 1 | 1 | 16 machine cycles | | | 16 machine cycles | | | | | | $\frac{f_X}{4f_{XT}}$ machine cycles<br>(32 machine cycles) | | |
| 1 | X | X | $\frac{f_X}{64f_{XT}}$ machine cycles<br>(2 machine cycles) | | | $\frac{f_X}{8f_{XT}}$ machine cycles<br>(16 machine cycles) | | | $\frac{f_X}{4f_{XT}}$ machine cycles<br>(32 machine cycles) | | | | | |

The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz and $f_{XT}$ = 32.768 kHz.

Remarks: The CPU clock Φ is supplied to the μPD753XX internal CPU. The reciprocal of the clock becomes the minimum instruction time. (The present manual defines it to be one machine cycle.)

(2) System and CPU clock change sequence
System and CPU clock change is explained using Fig. 5.2–6.

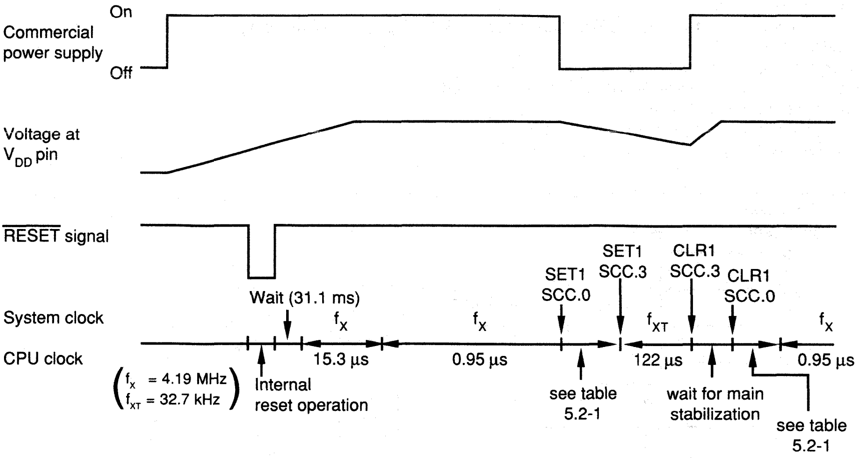


**Figure 5.2–6 System and CPU Clock Change**

1) When the RESET signal is generated, the CPU starts operating at the minimum speed of the main system clock (15.3 μs / 4.19 MHz) after the wait time (31.3 ms / 4.19 MHz) to allow time for stabilizing the oscillator.

2) After enough time has elapsed for the $V_{DD}$ pin voltage to rise to an adequate level PCC is rewritten and the μPD753XX operates at the maximum speed.

3) Turning off the commercial power supply is detected by using interrupt input (INT4 is useful); SCC. 0 is set, and the μPD753XX operates using the subsystem clock. (At the time, a check must have been made to ensure that subsystem clock oscillation has started. After the time required to change to the subsystem clock (32 machine cycles) has elapsed, SCC 3 is set and main system clock oscillation is stopped.

4) Restoration of the commercial power supply is detected by using an interrupt. SCC. 3 is cleared and main system clock osscillation is started. After the time required to stabilize oscillation has elapsed, SCC. 0 is cleared and the μPD753XX operates at the maximum speed.

5.2.4 Clock output circuit

(1) Clock output circuit configuration
Fig. 5.2-7 shows a clock output circuit block diagram.

(2) Clock output circuit function
The clock output circuit, which outputs clock pulses from the P22/PCL pin, is used to supply clock pulses to remote control output or peripheral LSIs.
Clock pulses are output in the following sequence:
(a) Clock output frequency is selected. Clock output is disabled.
(b) 0 is written into P22 output latch.
(c) Port 2 input/output mode is placed in output mode.
(d) Clock output is enabled.

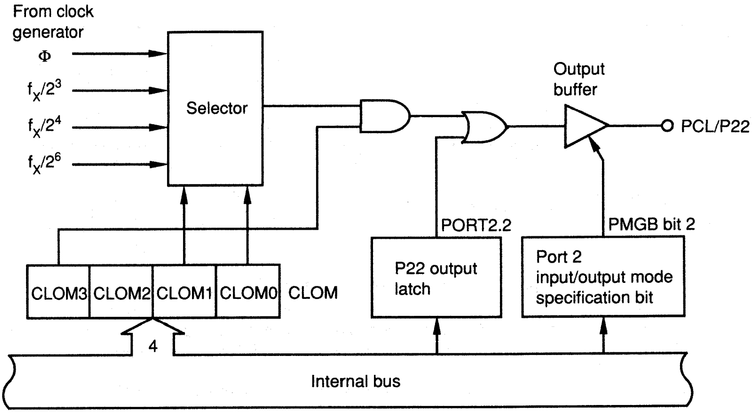


**Figure 5.2–7 Clock Output Circuit Block Diagram**

Remarks: The circuit is designed so as not to output a spurious short-width pulse when changing between clock output enable and disable.

(3) Clock output mode register (CLOM)
CLOM is a 4-bit register to control clock output.
CLOM is set by using a 4-bit memory operation instruction. It cannot be read.

Example: To output clock Φ from PCL/P22 pin.

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

When the $\overline{\text{RESET}}$ signal is generated, CLOM is cleared and clock output is disabled.

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

Clock output frequency selection bit

| CLOM1 | CLOM0 | |
|---|---|---|
| 0 | 0 | Φ output (Note) (1.05 MHz, 524 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: Φ is the CPU clock selected by setting PCC.

Clock output enable/disable bit

| CLOM3 | |
|---|---|
| 0 | Output disabled |
| 1 | Output enabled |

Caution: Be sure to write 0 into CLOM bit 2.

**Figure 5.2–8 Clock Output Mode Register Format**

(4) Application example to remote control output

The μPD753XX clock output function is applicable to remote control output. Remote control output carrier frequency is selected by using the clock frequency selection bit of the clock output mode register. Pulse output is enabled or disabled under software control by using the clock output enable/disable bit.

It is designed so that a spurious short-width pulse will not be produced when changing between clock output enable and disable.

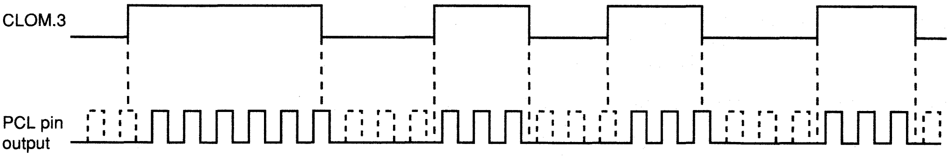


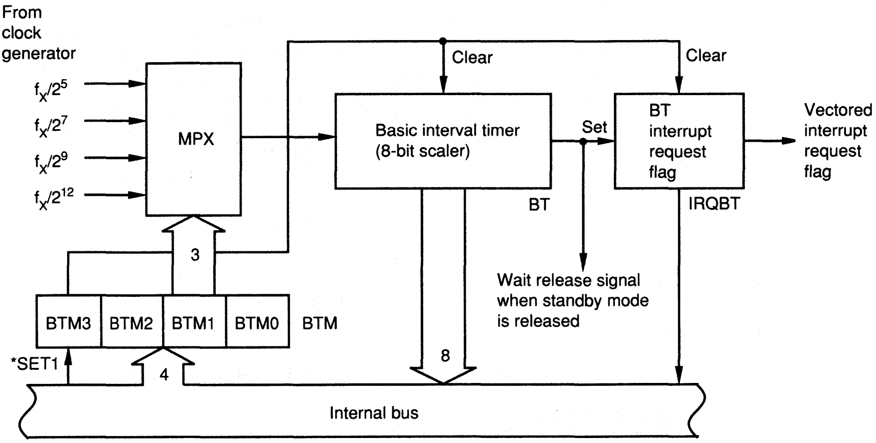
**Figure 5.2–9 Remote Control Output Application Example**

### 5.3 Basic Interval Timer

The μPD753XX contains an 8-bit basic interval timer which has the following functions:
(a) Reference time generation (four types of time interval)
(b) Wait time selection and count when standby mode is released.
(c) Count read
The basic interval timer can also be applied to a watchdog timer to detect program overrun.

5.3.1 Basic interval timer configuration

Fig. 5.3–1 shows the configuration of the basic interval timer


From clock generator
$f_X/2^5$
$f_X/2^7$
$f_X/2^9$
$f_X/2^{12}$
MPX
3
BTM3
BTM2
BTM1
BTM0
BTM
*SET1
4
Clear
Clear
Basic interval timer (8-bit scaler)
Set
BT
BT interrupt request flag
IRQBT
Vectored interrupt request flag
Wait release signal when standby mode is released
8
Internal bus

Remarks: * denotes instruction executing.

**Figure 5.3–1 Basic Interval Timer Configuration**

5.3.2. Basic interval timer mode register (BTM)

BTM is a 4-bit register for controlling operation of the basic interval timer.
BTM is set by using a 4-bit memory operation instruction.
Bit 3 can be set individually by using a bit operation instruction.

Example 1: To 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
```

Example 2: To clear BT and IRQBT (watchdog timer application).

```
SEL    MB15      ;        or CLR1 MBE
SET1   BTM.3     ;        BTM BIT 3 is set  to 1.
```

When bit 3 is set to 1, the basic interval timer contents are cleared. At the same time, the basic interval interrupt request flag (IRQBT) is also cleared. (The basic interval timer starts.)
When the $\overline{\text{RESET}}$ signal is generated, the BTM contents are cleared and interrupt request signal generation is set for the longest interval.

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

| | | | Input clock specification | Interrupt time interval (wait time when standby mode is released) |
|---|---|---|---|---|
| 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 | $f_X/2^7$ (32.7 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 than the above | | | Undefined | – |

The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz.

Basic interval timer start control bit

The basic interval timer is started (counter and interrupt request flag are cleared) by setting the bit to 1. When timer operation starts, the bit is automatically reset to 0.

**Fig. 5.3-2 Basic Interval Timer Mode Register Format**

5.3.3. Basic interval timer operation

The basic interval timer (BT) is incremented each time a pulse is received from the clock generator. When an overflow occurs, the interrupt request flag (IRQQBT) is set. BT count operation cannot be stopped.
An interrupt generation time interval can be selected from among four types by setting BTM. (See Fig. 5.3—2.)
The basic interval timer and interrupt request flag can be cleared by setting BTM bit 3 to 1 (the interval timer function starts).
The basic interval timer (BT) count can be read by using an 8-bit operation instruction. Data cannot be written into BT.

Caution:
To prevent reading of unstable data during count update when reading the basic interval timer count, execute the read instruction twice and compare the results. If they are valid values, the later read value is used as the read result, if they differ from each other completely, reexecute from the beginning.

Example: To read the BT count

```
        SET1  MBE
        SEL   MB15
        MOV   HL, #BT     ;   BT address is set in HL.
LOOP:   MOV   XA, @HL     ;   First read
        MOV   BC, XA
        MOV   XA, @ HL    ;   Second read
        SKE   A, C
        BR    LOOP
        MOV   A, X
        SKE   A, B
        BR    LOOP
```

The wait function is provided to stop CPU operation until the basic interval timer overflows in order to allow time for system clock oscillation to become stable when the STOP mode is released.
Although the wait time after the $\overline{\text{RESET}}$ signal is generated is fixed, it can be selected by setting BTM when the STOP mode is released by an interrupt. The wait time is selected by setting BTM as shown in Fig. 5.3—2. BTM setting must be performed before the STOP mode is set. (For details, see CHAPTER 7.)

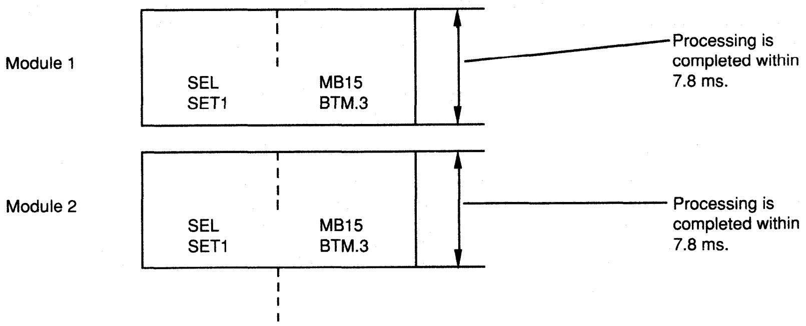
5.3.4 Basic interval timer application examples

Example 1: To enable basic interval timer interrupt and set interrupt generation interval at 1.95 ms (at 4.19 MHz).

```
SEL     MB 15
MOV     A, #1111B
MOV     BTM, A      ;   BTM setting and timer function start
EI                  ;   Interrupt is enabled.
EI      IEBT        ;   BT interrupt is enabled.
```

Example 2: Watchdog timer applciation
A program is divided into serval modules which terminate processing within the BT setup time. BT and IRQBT are cleared at the end of each module. If an interrupt is generated, overrun is asumed to have occured.

```
           SEL     MB15
Initial-   MOV     A, #1101B   ;   7.8 ms interval is set.
izatiion   MOV     BTM, A      ;   BTM setting and timer function start
           EI
           EEI     IEBT
```



Module 1
Processing is completed within 7.8 ms.
Module 2
Processing is completed within 7.8 ms.

```
SEL     MB15
SET1    BTM.3
```

```
SEL     MB15
SET1    BTM.3
```

Example 3: To set the wait time, when the STOP mode is released, using an interrupt to 7.8 ms.

```
SEL     MB15       ;   or CLR1 MBE
MOV     A, #1101B
MOV     BTM, A     ;   BTM ← 1101B
STOP               ;   STOP mode is set
NOP
```

Example 4:
To set the high-level width of a pulse input to INT4 interupt (both rising and falling edge detection). (The pulse width must not exceed the BT setup value. The BT setup value must be 7.8 ms or more.)
< INT4 interrupt routine (MBE = 0) >

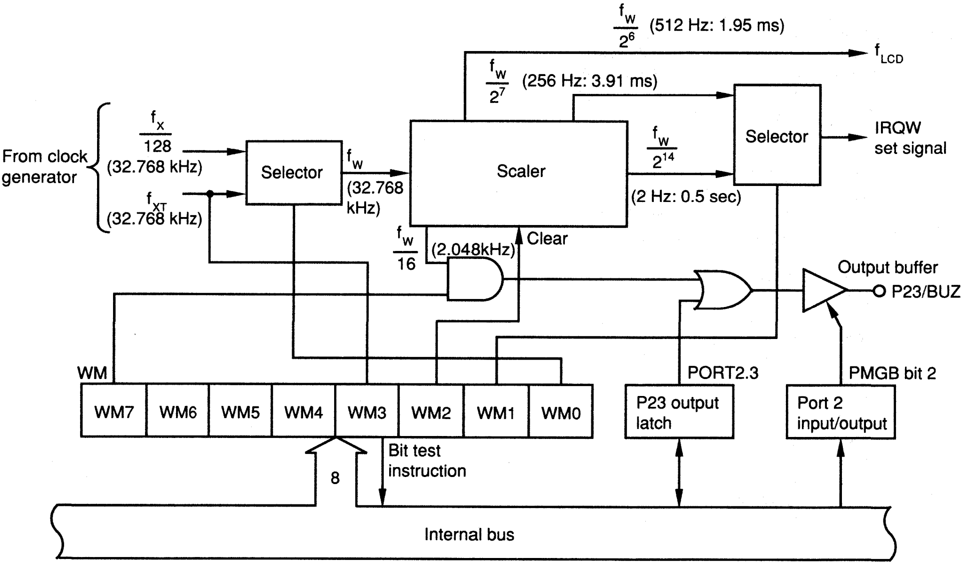
```
LOOP:   MOV     XA, BT          ;   First read
        MOV     BC, XA          ;   Data is stored.
        MOV     XA, BT          ;   Second read
        SKE     A, C
        BR      LOOP
        MOV     A, X
        SKE     A, B
        BR      LOOP
        SKT     PORT0.0         ;   P00=1?
        BR      AA              ;   NO
        MOV     XA, BC          ;   Data is stored in data memory.
        MOV     BUFF, XA
        CLR1    FLAG
        RETI
   AA:  MOV     HL, #BUFF
        MOV     A, C
        SUBC    A, @HL
        INCS    L
        MOV     C, A
        MOV     A, X
        SUBC    A, @HL
        MOV     B, A
        MOV     XA, BC
        MOV     BUFF, XA        ;   Data is stored.
        SET1    FLAG            ;   Data existence flag is set.
        RETI
```

### 5.4 Watch Timer

The μPD753XX incorporates a watch timer (one channel) which has the following function:
(a) The test flag (IRQW) is set at 0.5 s time intervales. The standby mode can be released using IRQW.
(b) The main system and subsystem clocks can be assigned 0.5-s intervals.
(c) In the rapid feed mode, time intervals multiplied by 128 (3.91 ms) are enabled. The mode is useful for program debugging and testing.
(d) A fixed frequency (2.048 kHz) can be output to P23/BUZ. It can be used for sounding the buzzer and trimming the system clock oscillation frequency.
(e) Since the scaler can be cleared, the watch can be started at zero seconds.

5.4.1 Watch timer configuration

Fig. 5.4–1 shows a watch timer block diagram.



The values enclosed in parentheses are applied when $f_X$ = 4.194304 MHz and $f_{XT}$ = 32.768 kHz.

**Figure 5.4–1 Watch Timer Block Diagram**

5.4.2 Watch mode register

The watch dog timer (WM) consists of eight bits to control the watch timer. Fig. 5.4–2 shows the watch mode register format. The watch mode register (except bit 3) is set using an 8-bit operation instruction. Bit 3 is used to test the XT1 pin input level. The input level to the XT1 pin can be tested by making a bit test. When the RESET signal is generated, all bits except bit 3 are cleared.

Example: To produce time using the main system clock (4.19 MHz). To enable buzzer output.

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

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| F98H | WM7 | 0 | 0 | 0 | WM3 | WM2 | WM1 | WM0 | WM |

Count clock ($f_W$) selection bit

| | | |
|---|---|---|
| WM0 | 0 | System clock dividing output $\frac{f_X}{128}$ is selected. |
| | 1 | Subsystem clock $f_{XT}$ is selected. |

Operation mode selection bit

| | | |
|---|---|---|
| WM1 | 0 | Normal watch mode (IRQW is set by using $\frac{f_W}{2^{14}}$, 0.5 s) |
| | 1 | Rapid feed watch mode (IRQW is set by using $\frac{f_W}{2^7}$, 3.91 ms) |

Watch operation enable/disable bit

| | | |
|---|---|---|
| WM2 | 0 | Watch operation is stopped (divider is cleared). |
| | 1 | Watch operation is enabled. |

Input level to XT1 pin (bit test only is enabled)

| | | |
|---|---|---|
| WM3 | 0 | Input level to XT1 pin is low. |
| | 1 | Input level to XT1 pin is high. |

BUZ output enable/disable bit

| | | |
|---|---|---|
| WM7 | 0 | BUZ output is disabled. |
| | 1 | BUZ output is enabled. |

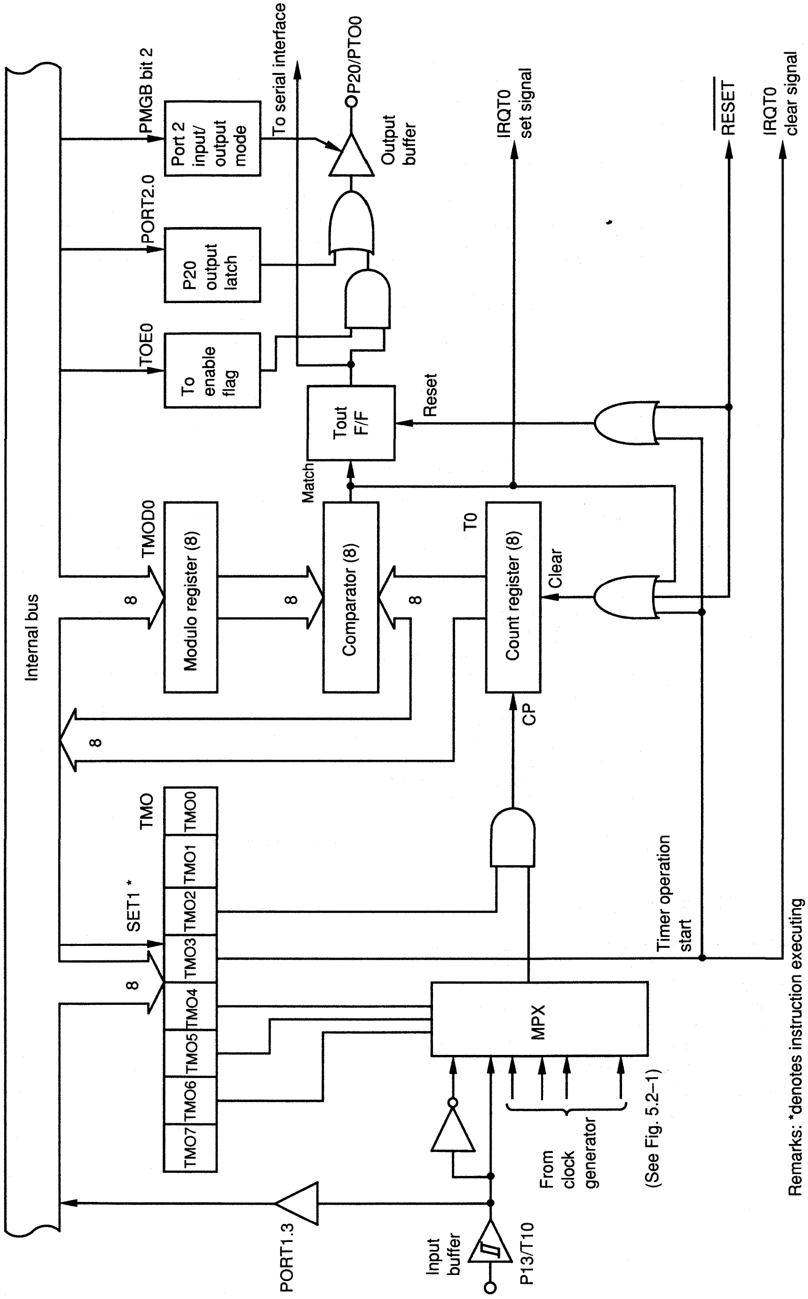
**Figure 5.4–2 Watch Mode Register Format**

### 5.5 Timer/event counter configuration

The μPD753XX incorporates a timer/event counter (one channel) as shown in Fig. 5.5–1.
The timer/event counter functions are as follows:

(a) Programmable interval timer operation
(b) Any desired frequency square wave output to PTO0 pin
(c) Event counter operation
(d) TI0 pin input is divided by N output to PTO0 pin (scaler operation).
(e) Real shift clock supply to serial interface circuit
(f) Count read function

Figure 5.5–1 Timer/Event Counter Block Diagram



Remarks: *denotes instruction executing

5.5.2 Basic configuration and operation of timer/event counter

The timer/event counter operation mode can be selected by using the timer/event counter mode register (TM0). The basic configuration and operation of the timer/event counter are explained below:

(1) Count pulse CP is selected by setting TM0 and input to the 8-bit count register T0.

(2) T0 is a binary 8-bit up counter incremented by one when CP is input. It is cleared when the $\overline{\text{RESET}}$ signal is generated, TM0 bit 3 is set (timer start), or coincidence signal is generated. T0 can be read at any time using an 8-bit memory operation instruction, but cannot be written

(3) The modulo register TMOD0 consists of eight bits to determine the T0 count. A value is set in TMOD0 using an 8-bit memory operation instruction, but TMOD0 cannot be read. When the $\overline{\text{RESET}}$ signal is generated, TMOD0 is initialized to FFH

(4) The comparator compares the T0 and TMOD0 contents. If they match, it generates a coincidence signal and sets the interrupt request flag (IRQT0)
Fig. 5.5-2 shows the count operation timing.

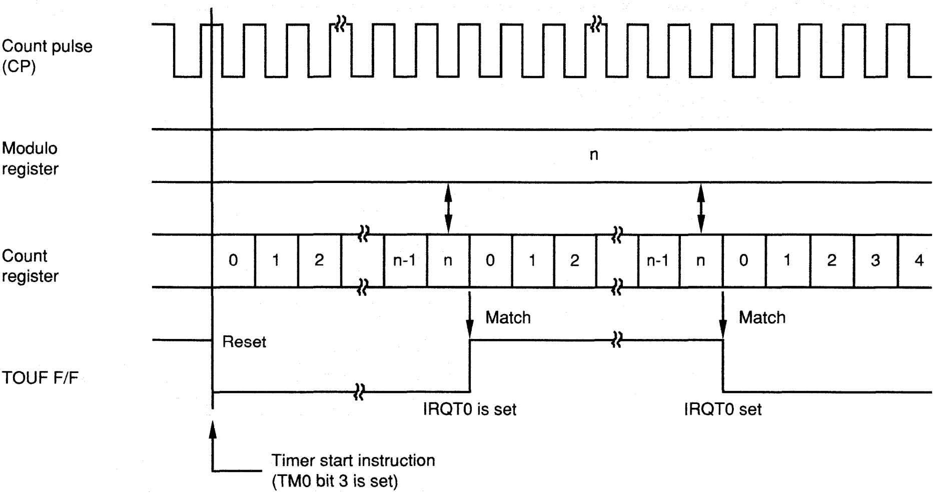


**Figure 5.5–2 Count Operation Timing Chart**

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

The mode register (TM0) consists of eight bits to control the timer/event counter. Fig. 5.5–3 shows the timer/event counter mode register format.
The timer mode register is set by using an 8-bit memory operation instruction.
Bit 3 is a timer start bit which can be set individually. Bit 3 is reset to 0 automatically when timer operation starts.

Example 1: To start the timer in interval timer mode with CP = 4.09 kHz.

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

Example 2: To restart the timer according to how the timer mode register is set.

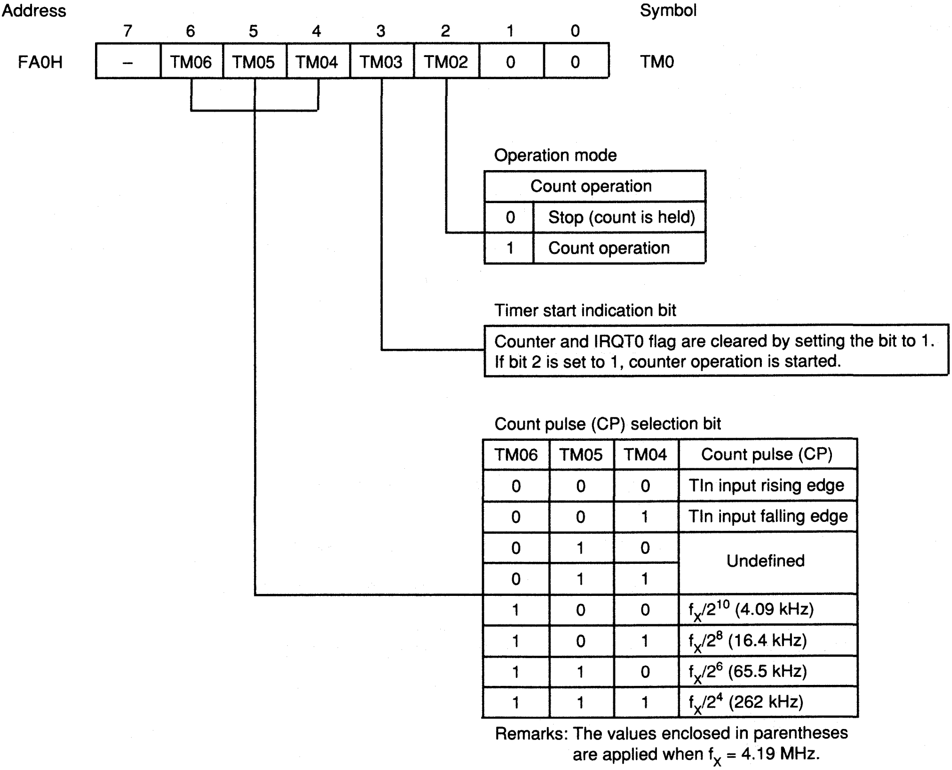
```
SEL     MB15   ;   or CLR1 MBE
SET1    TM0.3  ;   TM0.BIT3 ← 1
```

When the $\overline{\text{RESET}}$ signal is generated, all the timer mode register bits are cleared.

The timer/event counter output enable flag (TOE0) controls enable/disable of outputting the timer out F/F (TOUT F/F) state to the PTO0 pin. (See Fig. 5.5–4.) It is operated by using bit operation instruction and is enable to be writing.

The timer out F/F (TOUT F/F) is inverted by the coincidence signal received from the comparator. The timer out F/F is reset using an instruction to set timer mode register (TM0) bit 3.

When the $\overline{\text{RESET}}$ signal is generated, TOE0 and TOUT F/F are cleared.



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

Operation mode

| Count operation | |
|---|---|
| 0 | Stop (count is held) |
| 1 | Count operation |

Timer start indication bit

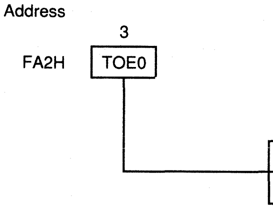
Counter and IRQT0 flag are cleared by setting the bit to 1.
If bit 2 is set to 1, counter operation is started.

Count pulse (CP) selection bit

| TM06 | TM05 | TM04 | Count pulse (CP) |
|---|---|---|---|
| 0 | 0 | 0 | TIn input rising edge |
| 0 | 0 | 1 | TIn input falling edge |
| 0 | 1 | 0 | Undefined |
| 0 | 1 | 1 | |
| 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) |

Remarks: The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz.

**Figure 5.5–3 Timer/Event Coumter Mode Register Format**



Timer/event counter output enable flag (write only)

| | |
|---|---|
| 0 | Disable |
| 1 | Enable |

**Figure 5.5–4 Timer/Event Counter Output Enable Flag Format**

5.5.4 Timer/event counter operation mode

The timer/event counter operates in the count operation stop or count operation mode depending on how the mode register is set.

The following are always enabled independently of how the mode register is set

1) TI0 pin signal input and test. (P13 pin input test can be made.)
2) Output of timer out F/F state to PTO0.
3) Modulo register (TMOD0) setting.
4) Count register (T0) read.
5) Interrupt request flag (IRQT0) setting, clear, and test

(a) Count operation stop mode

The count operation stop mode is set when TM0 bit 2 is set to 0. Since count pulse (CP) supply to the count register is stopped, count operation is not performed

(b) Count operation mode

The count operation mode is set when TM0 bit 2 is set to 1. Count pulses selected by using bits 4 to 6 are supplied to the count register, and count operation is performed as shown in Fig. 5.5–2.

Normally, timer operation is started by

1) setting a count value in the modulo register (TMOD0), then
2) setting the operation mode, count clock, and start indication in the mode register (TM0).

The modulo register is set by using an 8-bit data transfer instruction.

Caution: Set a value other than 0 in the modulo register.

Example: To set 3FH in channel 0 modulo register.

```
SEL    MB15        ;   or CLR1 MBE
MOV    XA, #3FH
MOV    TMOD0, XA
```

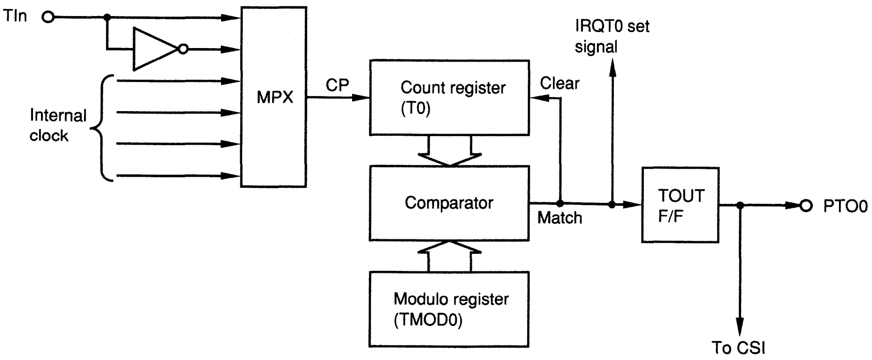


**Figure 5.5–5 Operation in Count Operation Mode**

5.5.5 Timer/event counter time setting

"Timer setup value" (period) is equal to "modulo register contents + 1" divided by "count pulse frequency" selected by setting the timer mode register.

$$T\ (s) = \frac{n+1}{F_{CP}}$$

T (s) : Timer setup value (seconds)
$F_{CP}$ (Hz) : Count pulse frequency (Hz)
n : Modulo register value ($n \neq 0$)

Once the timer is set, an interrupt request signal (IRQT0) is generated at the setup time intervals.

Table 5.5–1 lists the resolution and maximum setup value (time when FFH is set in the modulo register) of the timer/event counter for each count pulse.

Example: To produce 30 ms time intervals. ($f_X$ = 4.194304 MHz)
Use the mode with the maximum setup time 62.5 ms.

$$\frac{30\ \text{ms}}{244\ \mu\text{s}} = 122.9 = 7\text{AH}$$

Set 79H in the modulo register.

```
SEL    MB15
MOV    XA, #79H
MOV    TMOD0, XA
```

**Table 5.5–1 Resolution and Maximum Setup Value (4.19 MHz)**

| Mode register | | | Timer channel 0 | |
|---|---|---|---|---|
| TMO6 | TMO5 | TMO4 | Resolution | Maximum setup time |
| 1 | 0 | 0 | 244 μs | 62.5 ms |
| 1 | 0 | 1 | 61.1 μs | 15.6 ms |
| 1 | 1 | 0 | 15.3 μs | 3.91 ms |
| 1 | 1 | 1 | 3.81 μs | 977 μs |

### 5.5.6 Caution on timer/event counter application

(1) Error at timer start

An error with a maximum length of one clock period of the count pulse (CP) for the value calculated in 5.5.5 occurs in the time until a coincidence signal is generated after the timer starts (TM0.3 is set). This is because clearing of the count register is not synchronized with CP as shown in Fig. 5.5–6.

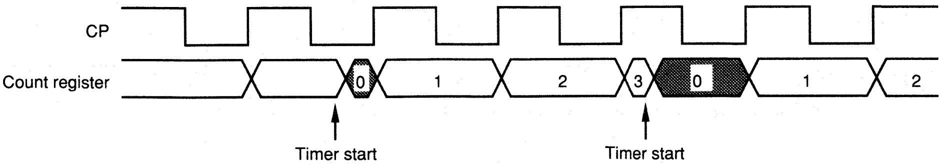


**Figure 5.5–6 Error at Timer Starting**

(2) Caution at timer start

Normally, the count register T0 and interrupt request flag IRQT0 are cleared when the timer starts (TM0 bit 3 is set). However, if the timer is placed in the operation mode, and IRQT0 setting and timer start occur simultaneously, IRQT0 may be unable to clear. There is no problem when IRQT0 is used for a vectored interrupt. In IRQT0 test application, however, a problem arises such than IRQT0 is set although the timer has been started. Thus, to start the timer at IRQT0 timing, stop the timer once (by setting TM0 bit 2 to 0), then restart it, or start the timer twice.

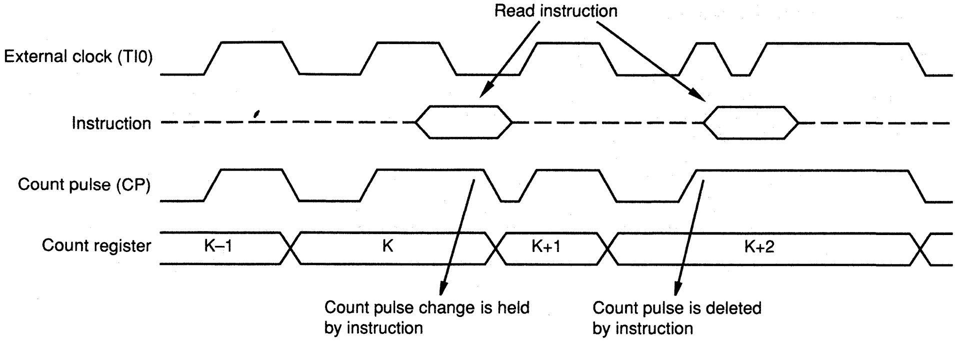
Example: Timer start at IRQT0 timing may be set

```
        SEL     MB15        ;   or CLR1 MBE
        MOV     XA, #0
        MOV     TM0, XA     ;   Timer stop
        MOV     XA, #4CH
        MOV     TM0, XA     ;   Restart
    or
        SEL     MB15        ;   or CLR1 MBE
        SET1    TM0.3
        SET1    TM0.3       ;   Restart
```

(3) Error at count register read

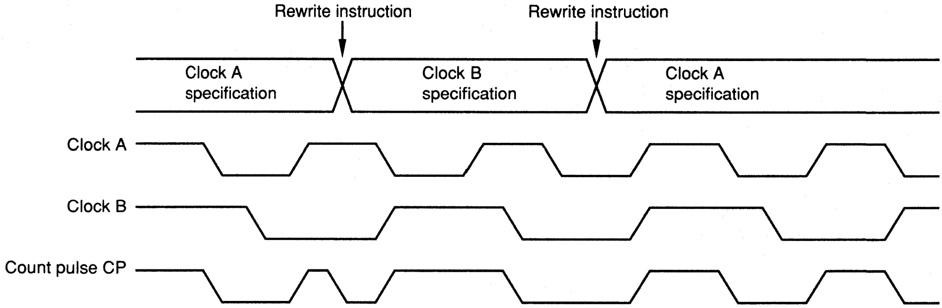
The count register contents can be read at any time using an 8-bit data memory operation instruction. During execution of the instruction, count pulse change is held and count register change is suppressed. Thus, if the count pulse signal source is input to TI0, count pulses as long as the instruction execution time are deleted. (This symptom does not occur if the internal clock is used for count pulses because they are synchronized with the instruction.)

Therefore, if TI0 is input as a count pulse and the count register contents are read, signals having a pulse width that prevents miscount, even if count pulses are deleted, must be input. Since the count hold period in read instruction execution is one machine cycle, the pulse width input to the TI0 pin must be longer than one machine cycle.

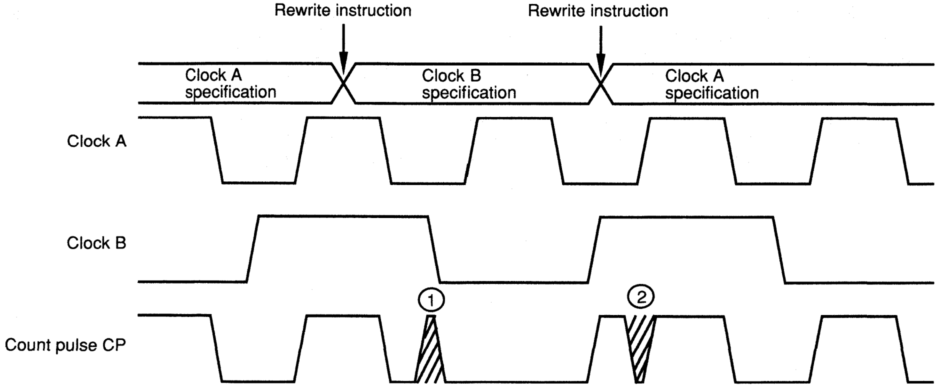


(4) Caution at count pulse change

If the timer mode register is rewritten and count pulse change is made, its specification becomes effective immediately after instruction execution.

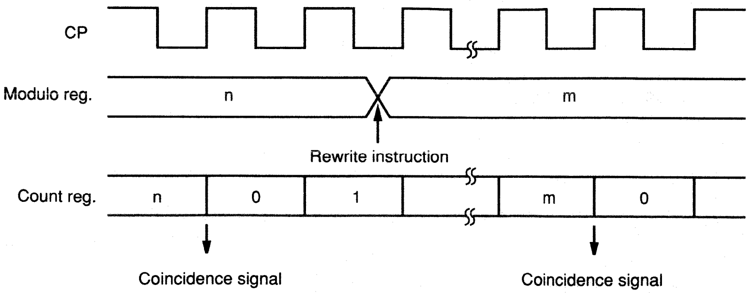


A count pulse ( ① or ② in the timing chart shown below) may occur depending on the clock combination when the count pulse changed. In this case, a miscount may occur or the count register contents may be destroyed. To change count pulse, be sure to set count mode register bit 3 to 1 and restart the timer at the same time.

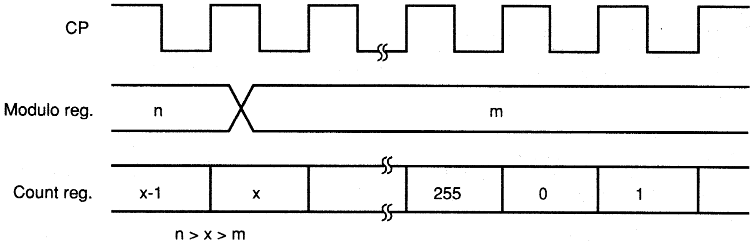


(5) Operation after modulo register change

The modulo register is changed when an 8-bit data memory operation instruction is executed.



If the value appearing after the modulo register is changed and becomes smaller than the count register value, the count register continues counting and overflows, then restarts counting from 0. Thus, if the value after the modulo register is changed (m) and becomes smaller than the value before it was changed (n), the timer must be restarted after the modulo register is changed.



5.5.7 Timer/event counter application

(1) Timer 0 is applied to an interval timer generating an interrupt at 50 ms intervals.

- The high-order four bits of the mode register are set to 0100B and a maximum setup time of 62.5 ms is selected.
- The low-order four bits of the mode register are set to 1100B.
- The modulo register setup value is as follows:

$$\frac{50\ \text{ms}}{244\ \mu\text{s}} = 205 = \text{CDH}$$

Example:
```
SEL   MB15           ; or CLR1 MBE
MOV   XA, #0CCH
MOV   TMOD0, XA      ; Modulo is set.
MOV   XA, #01001100B
MOV   TM0, XA        ; Mode is set and timer is started.
EI                   ; Interrupts is enabled.
EI    IET0           ; Timer interrupt is enabled.
```

Remarks: In this application, the TI0 pin can be used as an input pin.

(2) When the number of pulses input from the TI0 pin reaches 100, an interrupt is generated. (The pulses are active high.)
- The high-order four bits of the mode register are set to 0000 and the rising edge is selected.
- The low-order four bits of the mode register are set to 1100B.
- The modulo register is 99 = 100 –1.

```
Example: SEL   MB15            ; or CLR1 MBE
         MOV   XA, #100 –1
         MOV   TMOD0, XA       ; Modulo is set.
         MOV   XA, #00001100B
         MOV   TM0, XA
         EI
         EI    IET0            ; INTT0 is enabled.
```

### 5.6 Serial Interface

5.6.1 Serial interface configuration
The μPD753XX has a clocked serial interface (CSI) as shown in Fig. 5.6-2.

5.6.2 Serial interface functions
The μPD753XX serial interface includes the three modes described in (1) to (3) below.
Since the serial clock line $\overline{SCK}$ and serial data bus lines SB0/SB1 enable software to determine the output level, any desired transfer format can be handled.

(1) 3-line serial I/O mode
- Three lines of serial clock $\overline{SCK}$, serial output SO, and serial input SI
- Clock synchronous 8-bit send and receive (simultaneous send and receive)
- The serial transfer top can be changed between the most and least significant bits (MSB and LSB).
- This mode enables the μPD753XX to be connected to the μPD7500 series, μCOM-75X family, μCOM-87 family, and various peripheral I/O devices.

(2) SBI (serial bus interface) mode
- This mode conforms to the NEC serial bus format.
- Communication can be made with a number of devices by using the serial clock $\overline{SCK}$ and serial data bus SB0 or SB1 lines.
- Address, command, and data can be transferred, and a hardware function for discriminating the signals is included. (See Fig. 5.6-2.)
- The acknowledge, busy signal output function and the wake-up function for handshaking are included.

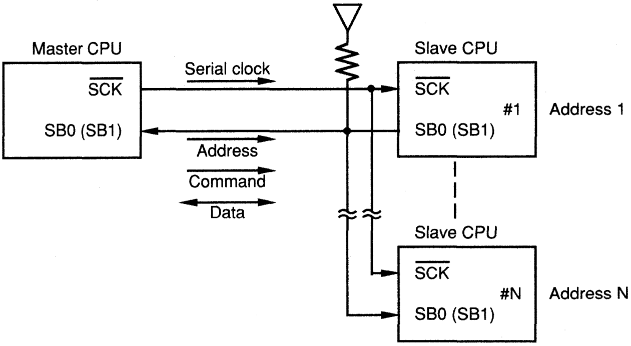


**Fig. 5.6-1 SBI System Configuration Example**

(3) 2-line serial I/O mode
- Communication can be made by using the serial clock $\overline{SCK}$ and serial data bus SB0 or SB1 lines.
- Communication can be made with a number of devices by using software to control the output level to the two lines. Any desired communication format can be handled.

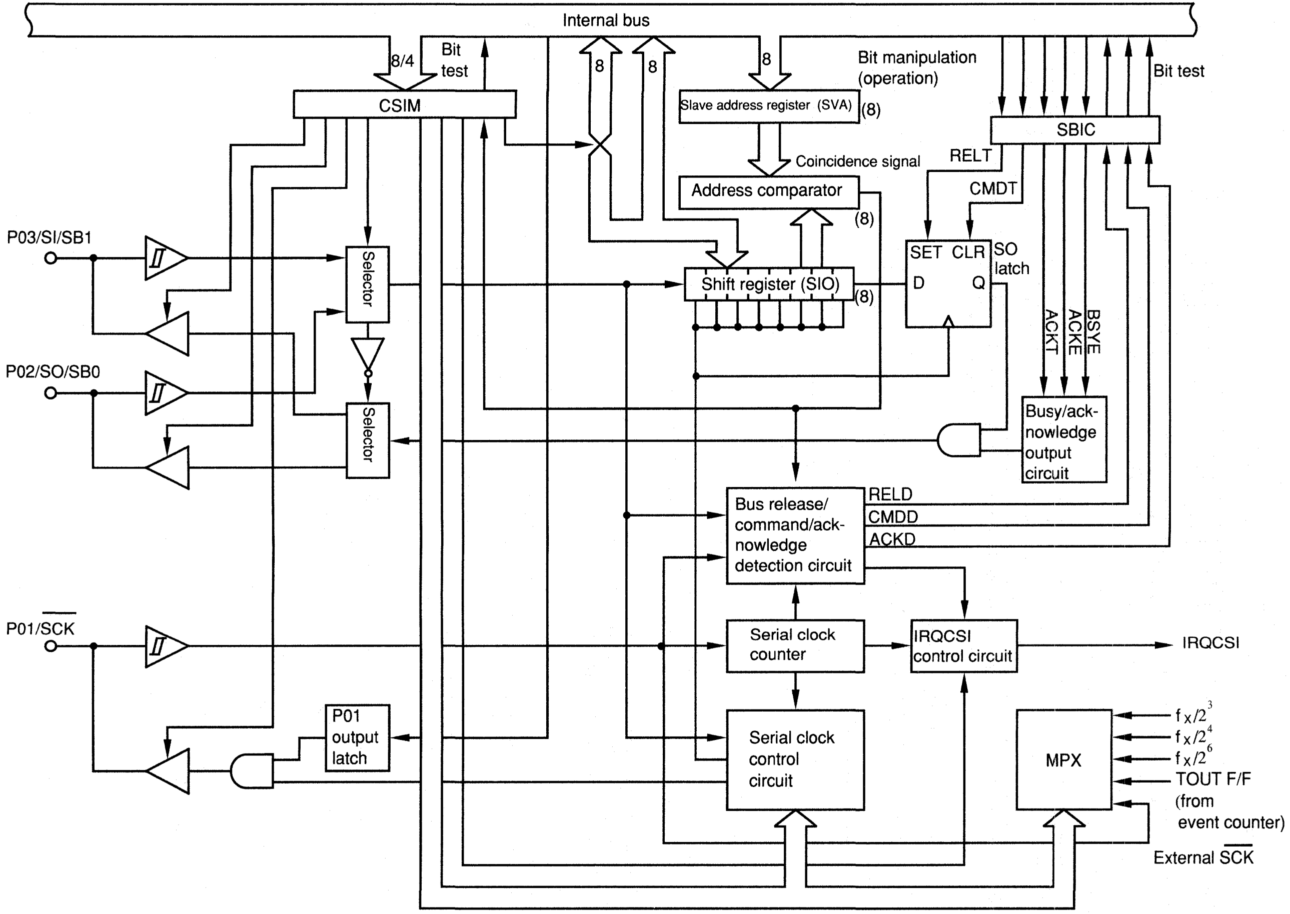
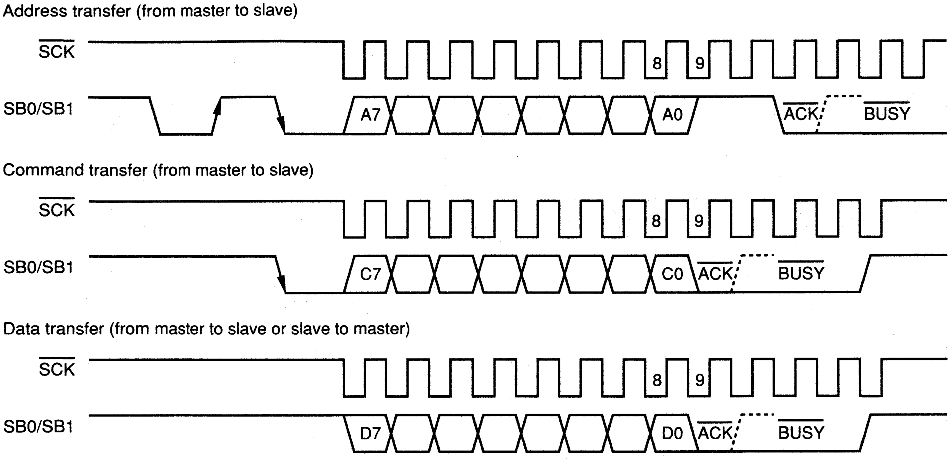


Figure 5.6.2 Serial Interface Block Diagram


Address transfer (from master to slave)

Command transfer (from master to slave)

Data transfer (from master to slave or slave to master)

**Figure 5.6–3 SBI Transfer Timing Chart**

5.6.3 Main register functions

(1) Serial operating mode register (CSIM)

The serial operation mode register (CSIM) consists of eight bits that specify the serial interface operating mode, serial clock, wake-up function, etc.

CSIM is set using an 8-bit memory operation instruction.

The high-order three bits of CSIM can be set bitwise using the bit name.

Bit 6 enables bit test only. Data written into the bit becomes invalid.

Example 1:

To select $f_X/2^4$ for serial clock, generate a serial interrupt IRQCSI at the end of each serial transfer, and make serial transfer in the SBI mode using the SB0 pin as the serial data bus line.

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

Example 2:

To enable serial transfer conforming to the CSIM contents.

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

Wake-up function specification bit (WUP):

(a) When WUP=0

IRQCSI is set each time a serial transfer ends.

WUP is set to 0 during normal transfer.

(b) When WUP = 1

WUP is set to 1 only during the SBI mode. IRQCSI is set only when the address received after the bus is released matches the slave address register (SVA) value (wake-up state). When the received address is not its own, an unnecessary interrupt will not be generated. The $\overline{\text{ACK}}$ signal is not output either. The SB0 (SB1) pin is placed in a high impedance state independent of the SO the latch state.

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

Fig. 5.6–4 shows the format of the serial operation mode register.

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FE0H | CSIE | COI | WUP | CSIM4 | CSIM3 | CSIM2 | CSIM1 | CSIM0 | CSIM |

Serial clock selection bits (W)

| CSIM1 | CSIM0 | Serial clock | | | $\overline{\text{SCK}}$ pin mode |
|---|---|---|---|---|---|
| | | 3-line serial I/O mode | SBI mode | 2-line serial I/O mode | |
| 0 | 0 | $\overline{\text{SCK}}$ pin input clock from the external | | | Input |
| 0 | 1 | Timer/event counter output (TO) | | | Output |
| 1 | 0 | $f_X/2^4$ (262 kHz) | | $f_X/2^6$ (65.5 kHz) | |
| 1 | 1 | $f_X/2^3$ (524 kHz) | | | |

The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz.

Serial interface operating mode selection bits (W)

| CSIM4 | CSIM3 | CSIM2 | Operating mode | Shift register bit order | SO pin function | SI pin function |
|---|---|---|---|---|---|---|
| X | 0 | 0 | 3-line serial I/O mode | $SIO_{7-0} \leftrightarrow XA$ (transfer starts at MBS) | SO/PO2 (CMOS output) | SI/PO3 (input) |
| | | 1 | | $SIO_{0-7} \leftrightarrow XA$ (transfer starts at LSB) | | |
| 0 | 1 | 0 | SBI mode | $SIO_{7-0} \leftrightarrow XA$ (transfer starts at MSB) | SB0/PO2 (N-channel open drain input/output) | PO3 input |
| 1 | | | | | PO2 input | SB1/PO3 (N-channel open drain input/output) |
| 0 | 1 | 1 | 2-line serial I/O mode | $SIO_{7-0} \leftrightarrow XA$ (transfer starts at MSB) | SB1/PO2 (N-channel open drain input/output) | PO3 input |
| 1 | | | | | PO2 input | SBI/PO3 (N-channel open drain input/output) |

(to be continued)

**Fig. 5.6-4 Serial Operation Mode Register Format**

Wake-up function specification bit (W)

| | | |
|---|---|---|
| WUP | 0 | IRQCSI is set each time a serial transfer ends in each mode. |
| | 1 | WUP is set to 1 only during the SBI mode. IRQCSI is set only when the address received after the bus is released matches the slave address register data (wake-up state). SB0 (SB1) is placed in the high impedance state |

Coincidence signal received from address comparator (R (bit test is only enabled)) Note 1, Note 2

| | | |
|---|---|---|
| COI | 0 | Slave address register and shift register data mismatch. |
| | 1 | Slave address register and shift register data match. |

Serial interface operation enable/disable specification bit (W) Note 3

| | | Shift register operation | Serial clock counter | IRQCSI flag | SO/SB0 and SI/SB1 pins |
|---|---|---|---|---|---|
| CSIE | 0 | Shift operation is disabled. | Clear | Hold | Port 0 function only |
| | 1 | Shift operation is enabled. | Count operation | Can be set. | Also used for port 0 according to each mode function |

**Figure 5.6–4 Serial Operation Mode Register Format (cont'd)**

Notes:
1. COI is only valid before serial transfer starts or after it is complete. An undefined value is read during serial transfer.
2. Data written into COI is ignored.
3. To use PO1/$\overline{\text{SCK}}$ pin as an input port, set the following below.
   • Set CSIM0 and CSIM1 bits to 0. (PO1/$\overline{\text{SCK}}$ pins are set to input mode.)
   • Set CSIE bit to 0. (Stop the serial interface operation.)
   When CSIM0=CSIM1=0 and CSIE=1 or CSIM0, CSIM1≠0 and CSIE=0, PO1/$\overline{\text{SCK}}$ pins are output to high level.

Remarks: (W): Data write is only enabled.
(R) : Data read is only enabled.

(2) Serial bus interface control register (SBIC)

The 8-bit serial bus interface control register (SBIC) consists of the serial bus state control bits and flags indicating the states of input data from serial bus. It is mainly used in the SBI mode.
Fig. 5.6–5 shows the SBIC format. SBIC is set or tested by using a bit operation (manipulation) instruction. When the $\overline{\text{RESET}}$ signal is generated, all the SBIC bits are cleared.

Cautions:
1. SBIC cannot be set by using 4- or 8-bit memory operation instruction.
2. In the 3- or 2-line serial I/O mode, use only the following two bits for SO latch control:
   (a) Bus release trigger bit (RELT): To set SO latch
   (b) Command trigger bit (CMDT): To clear SO latch
3. For the bus release, command, acknowledge, and busy signals, see 5.6.5 (3).

Example 1: To output command signal.

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

Example 2: To test RELD and CMDD and determine the receive data type for appropriate processing.

```
        SEL     MB15
        SKF     RELD    ;   RELD test
        BR      !ADRS
        SKT     CMDD    ;   CMDD test
        BR      !DATA
CMD:    ...     ...     ;   Command transfer
DATA:   ...     ...     ;   Data tansfer
ADRS:   ...     ...     ;   Address transfer
```

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FE2H | BSYE | ACKD | ACKE | ACKT | CMDD | RELD | CMDT | RELT | SBIC |

Bus release trigger bit (W)

| | |
|---|---|
| RELT | SO latch is set to 1 by setting the bit. It is used to output bus release signal. After SO latch is set, the bit is automatically cleared. |

Command trigger bit (W)

| | |
|---|---|
| CMDT | SO latch is cleared by setting the bit. It is used to output a command signal. After SO latch is cleared, the bit is automatically cleared. |

Bus release detection flag (R)

| | | |
|---|---|---|
| RELD | Clear condition (RELD=0) | 1 When the transfer start is indicated.<br>2 When the address received after the bus is released dies not match the slave address register (SVA) data.<br>3 When the $\overline{\text{RESET}}$ signal is input. |
| | Setting condition (RELD=1) | When the address received after the bus is released matches the slave address register data. (Wake up) |

Command detection flag (R)

| | | |
|---|---|---|
| CMDD | Clear condition (CMDD=0) | 1 When the transfer start is indicated.<br>2 When the bus release signal is detected.<br>3 When the $\overline{\text{RESET}}$ signal is input. |
| | Setting condition (CMDD=1) | When the command signal is detected. |

Acknowledge trigger bit (W)

| | | |
|---|---|---|
| ACKT | Used only after transfer completion | Acknowledge signal is output during one clock period of $\overline{\text{SCK}}$ immediatly after execution of the set instruction. |

Remarks:
1. ACKT is automatically cleared after acknowledge signal is output. 2. ACKT cannot be cleared using software.
3. To set ACKT, set ACKE to 0.

Acknowledge enable bit (R/W)

| | | | |
|---|---|---|---|
| ACKE | 0 | Automatic acknowledge signal output is disabled (the signal can be output by setting the acknowledge trigger bit ACKT). | |
| | 1 | Before transfer completion | Acknowledge signal is output during the ninth clock period of $\overline{\text{SCK}}$ (automatically output by presetting ACKE to 1). |
| | | After transfer completion | Acknowledge signal is output during one clock period of $\overline{\text{SCK}}$ immediately after execution of the set instruction (automatically output by presetting ACKE to 1). |

Acknowledge detection flag (R)

| | | |
|---|---|---|
| ACKD | Clear condition (ACKD=0) | 1 When the transfer is started.<br>2 When the $\overline{\text{RESET}}$ is input. |
| | Setting conditions (ACKD=1) | When the acknowledge signal is detected. |

Synchronous busy enable bit (R/W)

| | | |
|---|---|---|
| BSYE | 0 | Synchronous busy signal output is disabled. Synchronous busy signal output is stopped in synchronization with the $\overline{\text{SCK}}$ falling edge immediately after execution of the clear instruction. |
| | 1 | Synchronous busy signal is output on the $\overline{\text{SCK}}$ falling edge following an acknowledge signal. |

Remarks:
(R): Read is only enabled. (W): Write is only enabled. (R/W): Both read and write are enabled.

**Figure 5.6–5 SBIC Format**

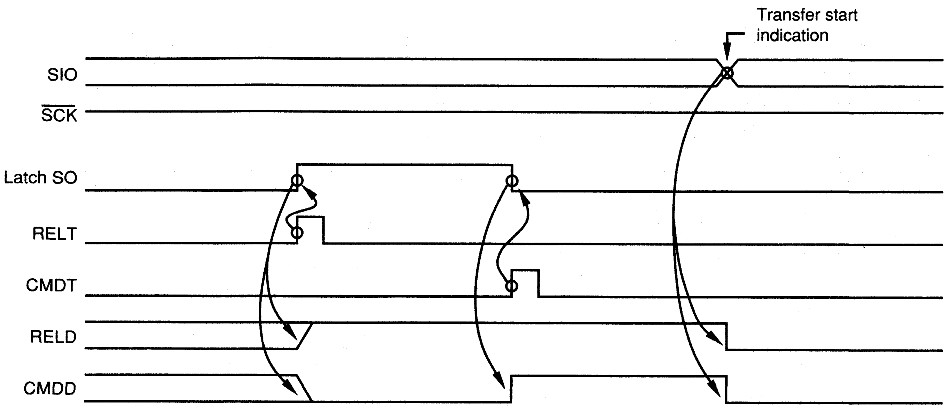


**Figure 5.6–6 RELT, CMDT, RELD, CMDD Operation**

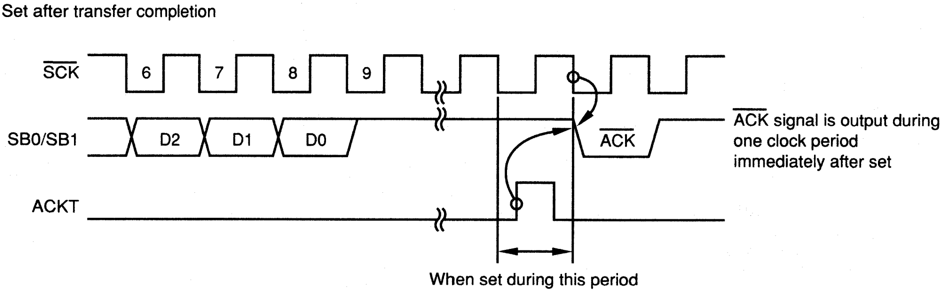
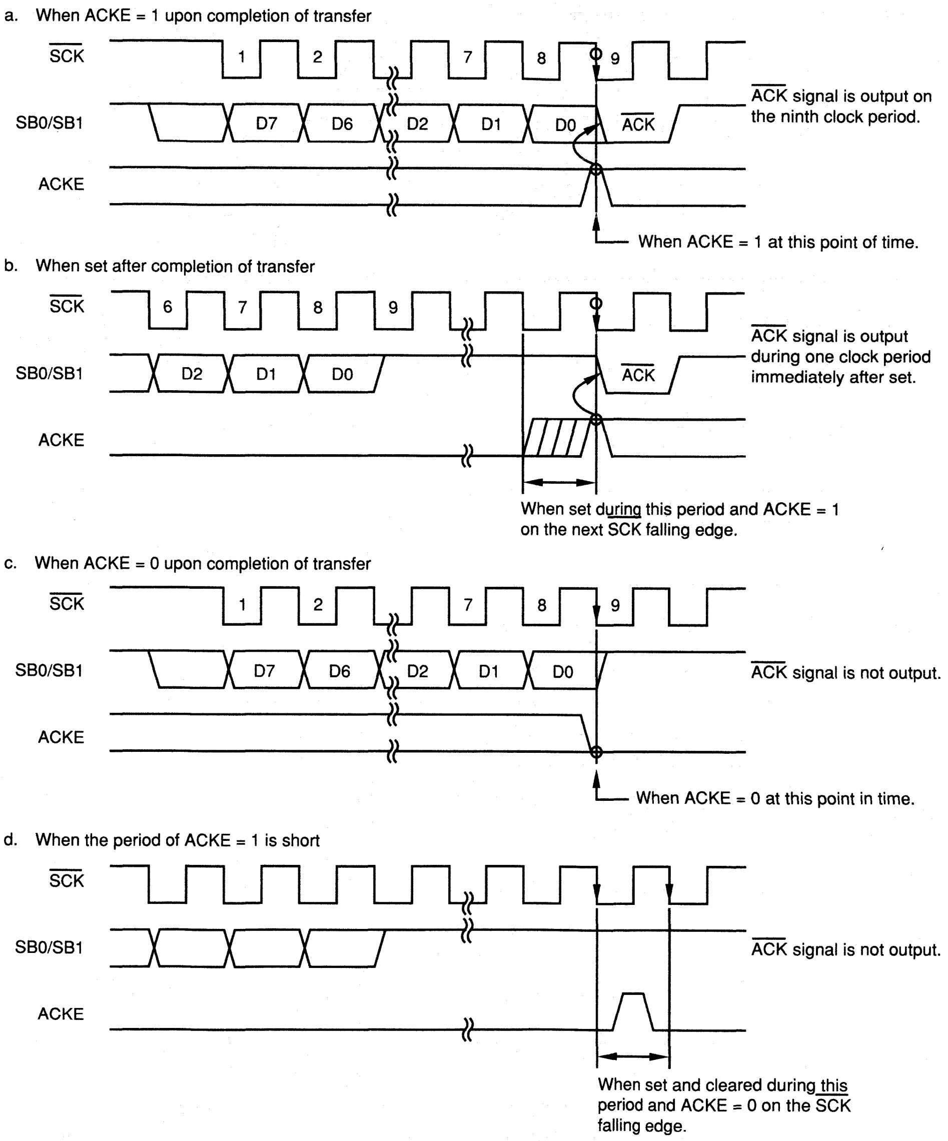

Set after transfer completion
SCK
6
7
8
9
SB0/SB1
D2
D1
D0
ACK
ACK signal is output during one clock period immediately after set
ACKT
When set during this period

**Figure 5.6–7 ACKT Operation**



a. When ACKE = 1 upon completion of transfer

ACK signal is output on the ninth clock period.

When ACKE = 1 at this point of time.

b. When set after completion of transfer

ACK signal is output during one clock period immediately after set.

When set during this period and ACKE = 1 on the next SCK falling edge.

c. When ACKE = 0 upon completion of transfer

ACK signal is not output.

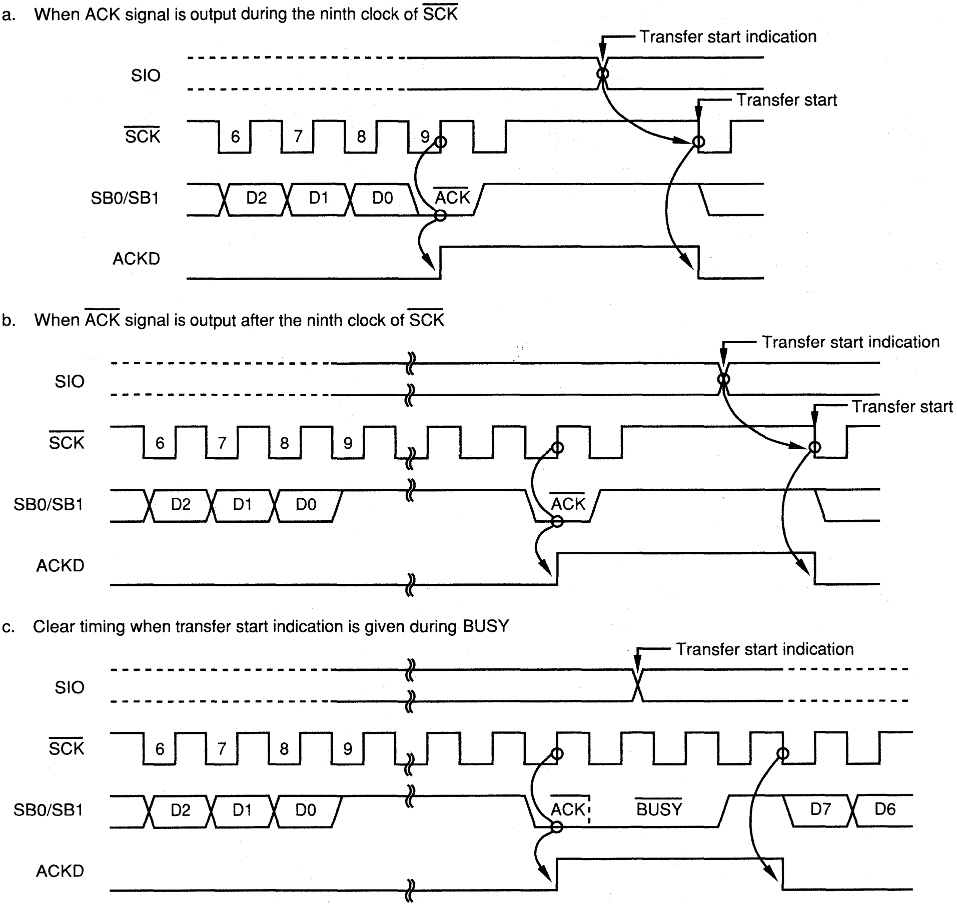
When ACKE = 0 at this point in time.

d. When the period of ACKE = 1 is short

ACK signal is not output.

When set and cleared during this period and ACKE = 0 on the SCK falling edge.

**Figure 5.6–8 ACKE Operation**

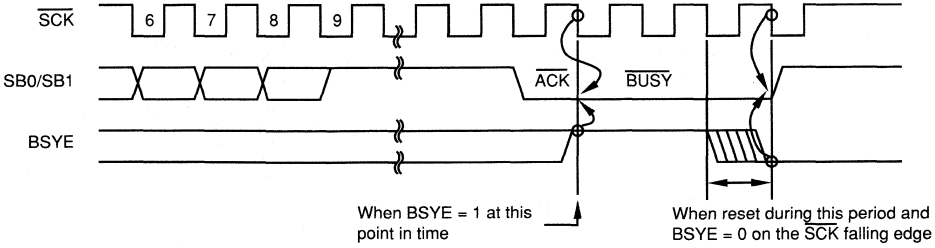


a. When ACK signal is output during the ninth clock of $\overline{\text{SCK}}$

b. When $\overline{\text{ACK}}$ signal is output after the ninth clock of $\overline{\text{SCK}}$

c. Clear timing when transfer start indication is given during BUSY

**Fig. 5.6-9 ACKD Operation**



SCK
SB0/SB1
BSYE
6
7
8
9
ACK
BUSY
When BSYE = 1 at this point in time
When reset during this period and BSYE = 0 on the $\overline{\text{SCK}}$ falling edge

**Figure 5.6–10 BSYE Operation**

(3) Slave address register (SVA)

When the μPD753XX is connected to the serial bus as a slave device in the SBI mode, the 8-bit slave address register (SVA) is used to set the slave address of the μPD753XX.

The SVA value is compared with received 8-bit data by the address comparator. If a match is found between them, serial operating mode register (CSIM) bit 6 (COI) is set to 1.

If a match is not found when the address is received, the bus release detection flag (RELD) is cleared. When WUP = 1, IRQCSI is set only if a match is found (wake-up is detected). This interrupt request can be used to know that a communication request is sent from the master to μPD753XX.

SVA can also be used to detect an error when the μPD753XX sends address, command, or data as a master device. (See 5.6.6.)

SVA is set using an 8-bit memory operation instruction. It can only be written.

When the $\overline{\text{RESET}}$ signal is generated in a mode which is not standby, the SVA value becomes undefined.

5.6.4 Signals in SBI mode

Tables 5.6–1 and 5.6–2 list the signals used in the SBI mode.

**Table 5.6–1 Signals in SBI Mode (I)**

| Signal name | Output device | Definition | Timing chart | Output condition | Flag influence | Explaination |
|---|---|---|---|---|---|---|
| Bus release signal (REL) | Master | SB0 (SB1) rising edge when $\overline{\text{SCK}}$ = 1 | (Figure 1) | • RELT is set. | • RELD is set.<br>• CMDD is cleared. | The signal is followed by CMD signal output indicating that the send data is an address |
| Command signal (CMD) | Master | SB0 (SB1) falling edge when $\overline{\text{SCK}}$ = 1 | (Figure 2) | • CMDT is set. | • CMDD is set. | i) After REL signal is output, send data is an address.<br>ii) When REL signal is not output, send data is a command. |
| Acknowledge signal ($\overline{\text{ACK}}$) | Master/ Slave | Low signal output to SB0 (SB1) during one clock period of $\overline{\text{SCK}}$ after completion of serial reception | (Figure 3) | 1 ACKE = 1<br>2 ACKT is set. | • ACKD is set. | Completion of reception |
| Busy signal ($\overline{\text{BUSY}}$) | Slave | (Synchronous busy signal)<br>Low signal output to SB0 (SB1) following acknowledge signal | | • BSYE = 1 | – | Serial reception cannot be done because processing is being performed. |
| | | (Asynchronous busy signal)<br>Low signal output to SB0 (SB1) (except during serial transfer). It is not synchronized with $\overline{\text{SCK}}$. | | • CMDT is set. | – | |
| Ready signal (READY) | Slave | High signal output to SB0 (SB1) before start or after completion of serial transfer | | 1 BSYE = 0<br>2 Execution of SIO data write instruction (transfer start indication) | – | Serial reception can be done. |

**Table5.6–2 Signals in SBI Mode (II)**

| Signal name | Output device | Definition | Timing chart | Output condition | Flag influence | Explanation |
|---|---|---|---|---|---|---|
| Serial clock ($\overline{\text{SCK}}$) | Master | Synchronous clock to output address, command, data, $\overline{\text{ACK}}$ signal, synchronous BUSY signal, etc. Address, command, or data is transferred with the first eight. | (Figure 4) | Execution of SIO data write instruction when CSIE =1 (serial transfer start indication) (Note 2) | IRQCSI is set (on the rising edge of ninth clock). (Note 1) | Signal output timing to serial data bus |
| Address (A7–0) | Master | 8-bit data transferred in synchronization with $\overline{\text{SCK}}$ after REL and CMD signals are output. | (Figure 5) | | (Note 1) | Slave device address value on serial bus |
| Command (C7–0) | Master | 8-bit data transferred in synchronization with $\overline{\text{SCK}}$ after CMD signal only is output (REL signal is not output). | (Figure 6) | | None | Indication message sent to slave device |
| Data (D7–0) | Master or Slave | 8-bit data transferred in synchronization with $\overline{\text{SCK}}$ when neither REL nor CMD signal is output. | (Figure 7) | | None | Numeric data processed by slave or master device |

Notes: 1. When WUP = 0, IRQCSI is always set on the ninth clock $\overline{\text{SCK}}$ rising edge.
When WUP = 1, IRQCSI is set only when the received address matches the value in the slave address register (SVA).
2. In the BUSY state, transfer is started after the READY state is set.

Figure 1:

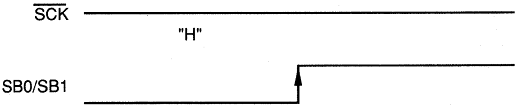


Figure 2:

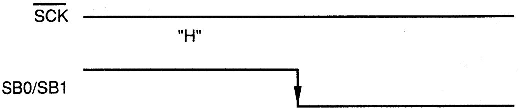


Figure 3:

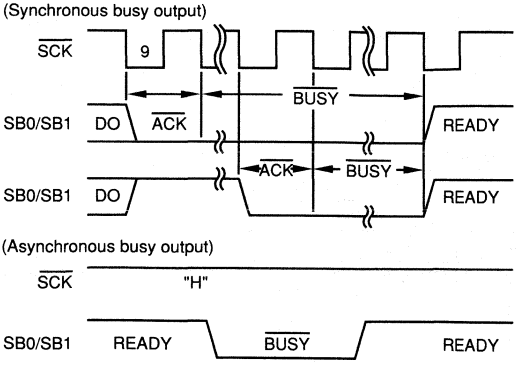


Figure 4:

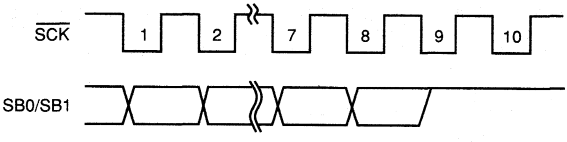


Figure 5:

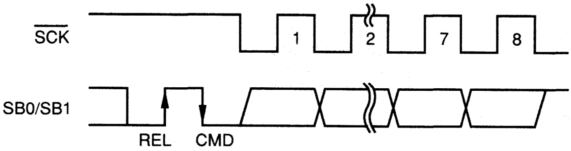


Figure 6:

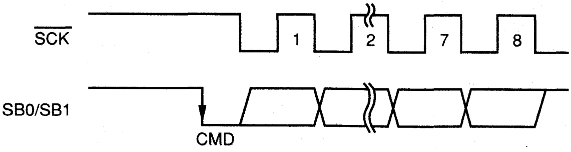
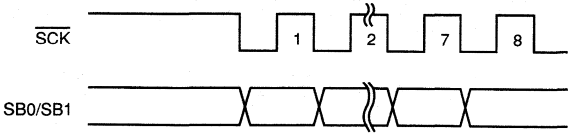


Figure 7:



5.6.5 Serial interface operation

The serial interface includes the following four operating modes.

— Operation stop mode
— 3-line serial I/O mode
— SBI mode
— 2-line serial I/O mode

**Table 5.6-3 Serial Interace Operation Mode**

| CSIM7 | CSIM3 | CSIM2 | Operating mode |
|---|---|---|---|
| 0 | — | — | Operation stop |
| 1 | 0 | — | 3-line serial I/O |
| 1 | 1 | 0 | SBI mode |
| 1 | 1 | 1 | 2-line serial I/O |

(1) Operation stop mode

When CSIE = 0, the serial interface is placed in the operation stop mode. In this mode, no serial transfer is done. The operation stop mode is set when the serial interface is not used, so that the power consumption is reduced. In this mode, the shift register does not perform shift operations and can be used as a normal 8-bit register. All the P01/$\overline{\text{SCK}}$, P02/SO/SB0, and P03/SI/SB1 pins are placed in the high impedance state and used only for the input port function. The P01/$\overline{\text{SCK}}$ pin is placed in the state listed in Table 5.6-4 according to how CSIM1 and CSIM0 are set.

**Table 5.6-4 CSIM1 and CSIM0 Settings**

| CSIM1 | CSIM0 | Serial clock | P01/$\overline{\text{SCK}}$ pin state |
|---|---|---|---|
| 0 | 0 | External clock | High impedance |
| 0 | 1 | Internal clock | A high level is output. However, a serial clock is output during serial transfer. |
| 1 | 0 | | |
| 1 | 1 | | |

Note: For the serial clock, see Fig. 5.6-4.

When the $\overline{\text{RESET}}$ signal is generated, the operation stop mode is set.

(2) 3-line serial I/O mode

The 3-line serial I/O mode is compatible with the mode used by the µPD7500 series or other µCOM-75X family devices. Fig. 5.6-11 shows the 3-line serial I/O mode operation timing.

Serial transfer start is indicated by executing an instruction to write data into the shift register (SIO), MOV or XCH.

Be sure to give a start indication when CSIE = 1.

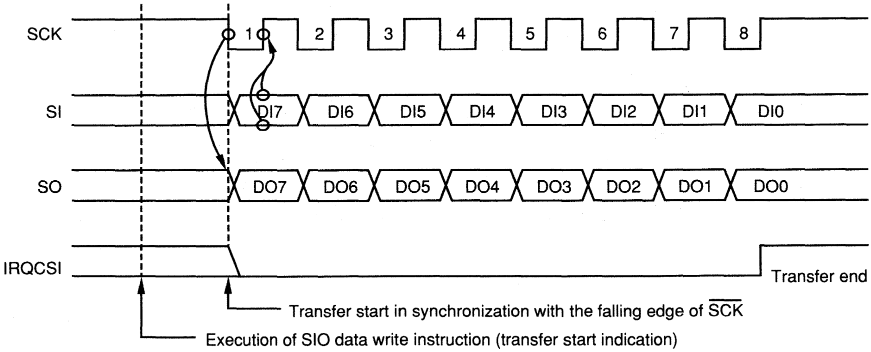

SCK
1
2
3
4
5
6
7
8
SI
DI7
DI6
DI5
DI4
DI3
DI2
DI1
DI0
SO
DO7
DO6
DO5
DO4
DO3
DO2
DO1
DO0
IRQCSI
Transfer end
Transfer start in synchronization with the falling edge of $\overline{\text{SCK}}$
Execution of SIO data write instruction (transfer start indication)

**Fig. 5.6-11 3-line Serial I/O Mode Timing**

In this mode, shift register shift operation is performed in synchronization with the serial clock ($\overline{\text{SCK}}$) falling edge, and send data is held in the SO latch and output from the SO pin. The receive data input to the SI pin is latched in the shift register on the rising edge of $\overline{\text{SCK}}$.
Since the SO pin is used as a CMOS output and outputs the SO latch state, the SO pin output state can be handled by setting the RELT and CMDT bits. However, do not try to change it during serial transfer.
The SCK pin output state can be controlled if the P01 output latch is operated in the output mode (internal clock mode), as described in 5.6.7.
A serial clock can be selected from among four clocks, as listed in Table 5.6-5, by setting the mode register.
Normally, shift operation is started by:
1) Setting the operating mode and serial clock selection data in the mode register (CSIM)
2) Setting transfer data in the shift register (SIO) (serial operation is started by executing the SIO data write instruction)
Serial transfer automatically stops at the end of 8-bit transfer, and the interrupt request flag (IRQCSI) is set.

**Table 5.6-5 Serial CLock Selection and Application**

| Mode register | | | Serial clock | Timing at which shift register can be read/written and serial transfer can be started | Application |
|---|---|---|---|---|---|
| CSIM 1 | CSIM 0 | Source | Serial clock mask | | |
| 0 | 0 | External $\overline{\text{SCK}}$ | Automatic mask at the end of 8-bit data transfer. | Only when serial transfer stops or $\overline{\text{SCK}}$ is high.<br>(Note) | Slave CPU |
| 0 | 1 | TOUT F/F | | | Half-duplex asynchronous transfer (under software control) |
| 1 | 0 | $f_X/2^4$ | | | Medium-speed serial transfer |
| 1 | 1 | $f_X/2^3$ | | | High-speed serial transfer |

Note: The serial transfer stops in the opertaion stop mode or when the serial clock is masked after 8-bit transfer has been made.

The shift register is read and written using 8-bit transfer instructions. At that time, LSB and MSB can be inverted by setting CSIM bit 2. This function enables the transfer top bit to be changed between LSB and MSB.

Eyample:
At the same time RAM data specified in the HL register pair is transfered to SIO, the SIO data is read into the accumulator, and serial transfer is started.

```
MOV  XA, @HL     ; Send data is taken out of RAM.
SEL  MB15        ; or CLR1 MBE
XCH  XA, SIO     ; Send data and receive data are exchanged, and transfer is started.
```

Caution:
The transfer top bit LSB or MSB is selected by changing the bit order in writing data into the shift register. The shift order of the shift register bit is always the same. Thus, even if the transfer top bit is changed (LSB to MSB or MSB to LSB) after data is written into the shift register, the data is transferred in the bit order specified before the transfer top bit is changed.

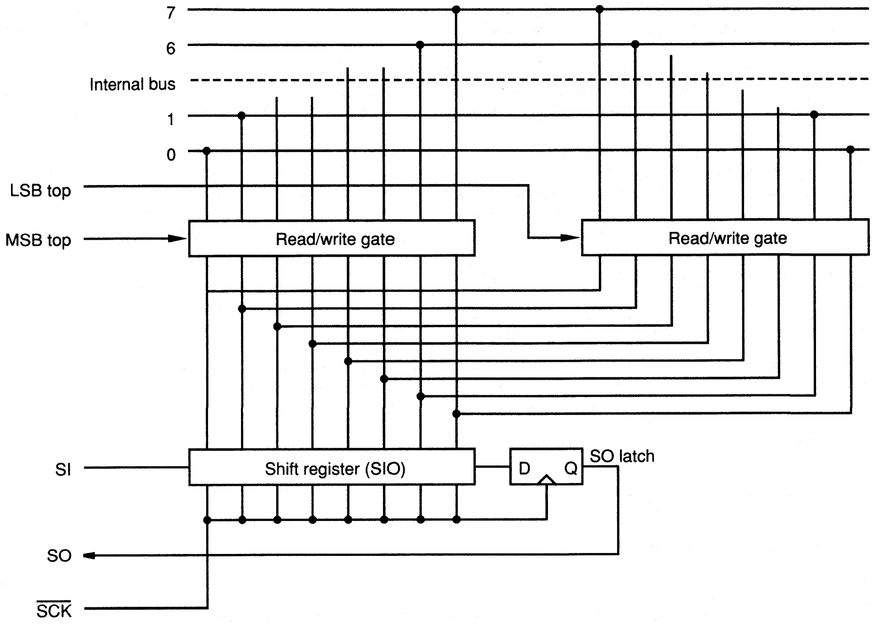


**Fig. 5.6-12 3-line I/O Mode Format**

(3) SBI mode

The SBI mode enables the μPD753XX to communicate with a number of devices by using the $\overline{\text{SCK}}$ and SB0 or SB1 lines. Figs. 5.6-13 to 5.6-15 show the timing of operations according to the type of data to be transferred.

In the SBI mode, indication of serial transfer start is also given by executing the instruction to write data into the shift register (SIO), MOV or XCH. Be sure to give a start indication when CSIE = 1.

Shift register shifting is made in synchronization with the serial clock ($\overline{\text{SCK}}$) falling edge, and send data is held in the SO latch and output from the SB0/P02 or SB1/P03 pin starting at MSB. Receive data input to the SB0 or SB1 pin is latched in the shift register on the rising edge of $\overline{\text{SCK}}$.

The SB0 or SB1 pin specified for the serial data bus is used as N-channel open drain input/output and needs pull-up. When data is received, the N-channel transistor must be turned off.

By writing FFH into SIO and shifting it, the N-channel transistor can always be turned off during transfer. When the wake-up function specification bit (WUP) is set to 1, however, the N-channel transistor is always turned off, and FFH need not be written into SIO before reception.

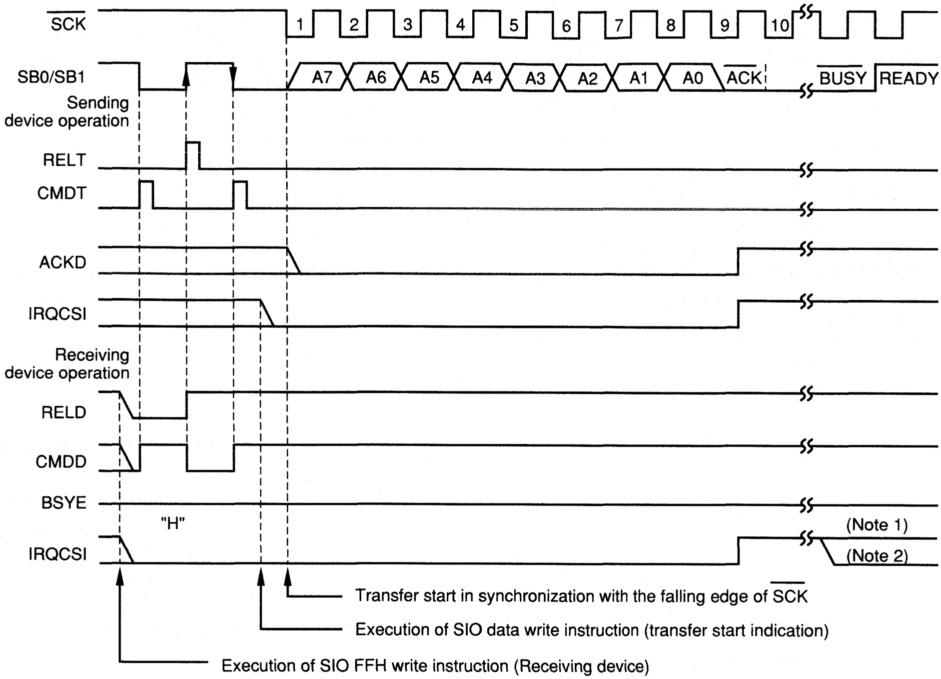

SCK
1
2
3
4
5
6
7
8
9
10
SB0/SB1
A7
A6
A5
A4
A3
A2
A1
A0
ACK
BUSY
READY
Sending device operation
RELT
CMDT
ACKD
IRQCSI
Receiving device operation
RELD
CMDD
BSYE
"H"
IRQCSI
(Note 1)
(Note 2)
Transfer start in synchronization with the falling edge of SCK
Execution of SIO data write instruction (transfer start indication)
Execution of SIO FFH write instruction (Receiving device)

**Fig. 5.6-13 SBI Mode Timing (Address Transfer)**

Note 1: Where SIO's data is read out by MOV instruction, then BSYE flag is cleared.
Note 2: Where SIO's data is exchanged (read/write) by XCH instruction.

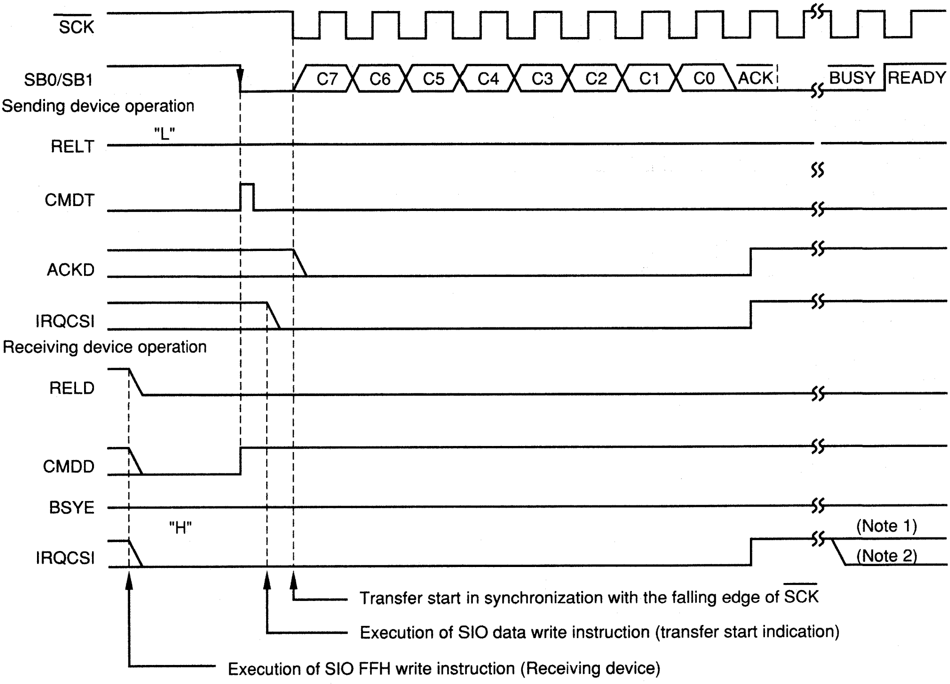

SCK
SB0/SB1
C7
C6
C5
C4
C3
C2
C1
C0
ACK
BUSY
READY
Sending device operation
"L"
RELT
CMDT
ACKD
IRQCSI
Receiving device operation
RELD
CMDD
BSYE
"H"
IRQCSI
(Note 1)
(Note 2)
Transfer start in synchronization with the falling edge of SCK
Execution of SIO data write instruction (transfer start indication)
Execution of SIO FFH write instruction (Receiving device)

**Fig. 5.6-14 SBI Mode Timing (Command Transfer)**

Note 1: Where SIO's data is read out by MOV instruction, then BSYE flag is cleared.
Note 2: Where SIO's data is exchanged (read/write) by XCH instruction.

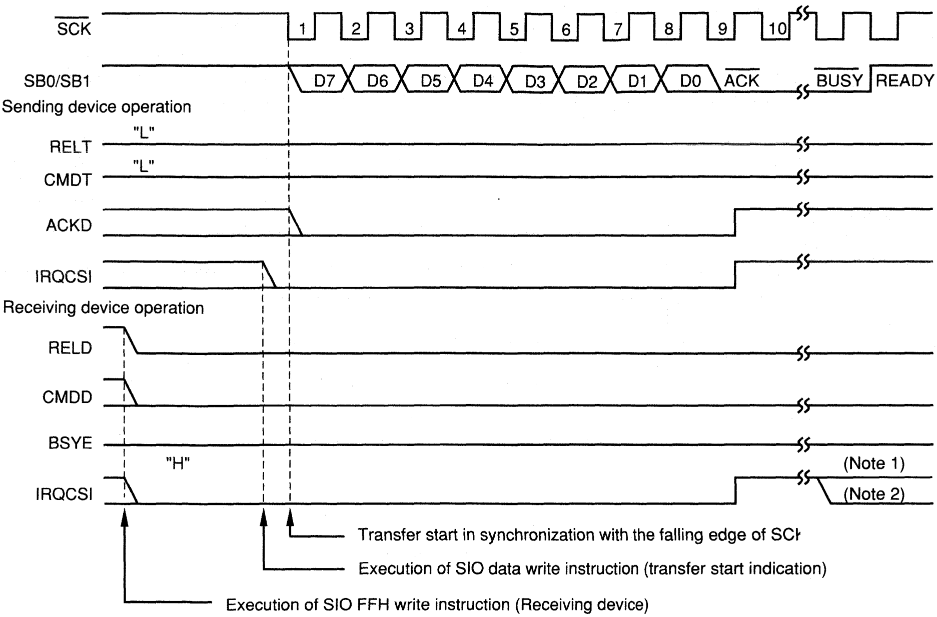

SCK
1
2
3
4
5
6
7
8
9
10
SB0/SB1
D7
D6
D5
D4
D3
D2
D1
D0
ACK
BUSY
READY
Sending device operation
"L"
RELT
"L"
CMDT
ACKD
IRQCSI
Receiving device operation
RELD
CMDD
BSYE
"H"
(Note 1)
IRQCSI
(Note 2)
Transfer start in synchronization with the falling edge of SCK
Execution of SIO data write instruction (transfer start indication)
Execution of SIO FFH write instruction (Receiving device)

**Fig. 5.6-15 SBI Mode Timing (Data Transfer)**

Note 1: Where SIO's data is read out by MOV instruction, then BSYE flag is cleared.
Note 2: Where SIO's data is exchanged (read/write) by XCH instruction.

(4) 2-line serial I/O mode

The 2-line serial I/O mode is used for communications using the SCK and SB0 or SB1 lines. The mode can handle any desired communication format by using software to control the SCK and SB0 or SB1 pin output levels.

Fig. 5.6-16 shows the 2-line serial I/O mode timing.

Serial transfer start indication is given by executing the instruction writing data into the shift register (SIO), MOV or XCH. Be sure to give start indication when CSIE = 1.

Shift register shifting is done in synchronization with the falling edge of the serial clock (SCK), and send data is held in the SO latch and output from the SB0/P02 or SB1/P03 pin starting at MSB. Receive data input from the SB0 or SB1 pin is latched in the shift register on the SCK rising edge.

The SB0 or SB1 pin specified for the serial data bus is used as an N-channel open drain input/output and needs pull-up. When data is received, the N-channel transistor must be turned off, thus FFH is previously written into SIO.

Since the SB0 or SB1 pin outputs the SO latch state, the SB0 or SB1 pin output state (high impedance or low level) can be controlled by setting the RELT and CMDT bits. However, do not change it during serial transfer.

In the output mode (internal clock mode), the SCK pin output state can also be controlled by using the P01 output latch, as described in 5.6.7.

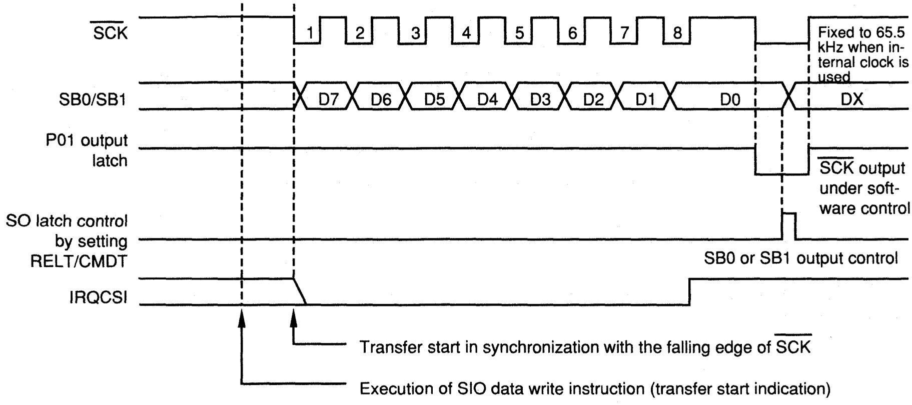

SCK
SB0/SB1
P01 output latch
SO latch control by setting RELT/CMDT
IRQCSI
1
2
3
4
5
6
7
8
D7
D6
D5
D4
D3
D2
D1
D0
DX
Fixed to 65.5 kHz when internal clock is used
SCK output under software control
SB0 or SB1 output control
Transfer start in synchronization with the falling edge of SCK
Execution of SIO data write instruction (transfer start indication)

**Fig. 5.6-16 2-line Serial I/O Mode Timing**

5.6.6 Error detection

Since the data of the sending serial bus SB0 or SB1 is also read by the shift register SIO of the device which sends data in the SBI or 2-line serial I/O mode, a send error can be detected as described below:
(1) SIO data comparison before the start of a send and after the end of a send
If both the pieces of data do not match, it is determined that a send error occurred.
(2) Use of slave address register (SVA)
Send data is also set in SVA. After the completion of a send, the COI bit (coincidence signal from the address comparator) in the serial operating mode register is tested. If the bit is set to 1, it is determined that normal sending has been performed; if 0, it is determined that a send error has occurred.

5.6.7 $\overline{\text{SCK}}$ pin output handling
The $\overline{\text{SCK}}$/P01 pin, which incorporates an output latch, can also produce static output by software control in addition to a normal serial clock.
The number of $\overline{\text{SCK}}$s can be set as desired by using software to control the P01 output latch (the SO/SB0, and SI/SB1 pins are controlled by setting the RELT and CMDT bits).
The $\overline{\text{SCK}}$/P01 pin output control method is described below:
(1) The serial operation mode register (CSIM) is set ($\overline{\text{SCK}}$ pin: Output mode, serial operation: Enabled). $\overline{\text{SCK}}$ is set to 1 when serial transfer stops.
(2) The P01 output latch is controlled by using bit manipulation (operation) instructions.
Example:
To output one clock pulse of $\overline{\text{SCK}}$ by using software.

```
SEL   MB15            ; or CLR1 MBE
MOV   XA, #10000011B  ; SCK (fX/2^3), output mode
MOV   CSIM, XA
CLR1  0FF0H.1         ; SCK/P01 ◄— 0
SET1  0FF0H.1         ; SCK/P01 ◄— 1
```

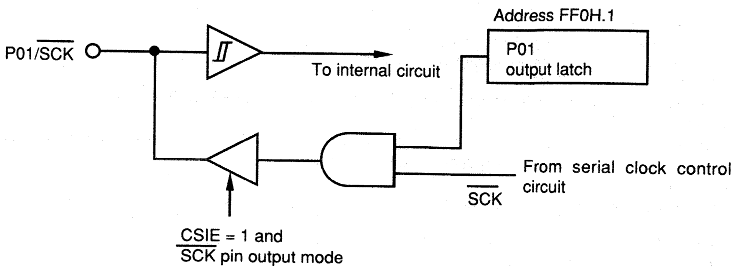

Address FF0H.1
P01/$\overline{SCK}$
To internal circuit
P01
output latch
From serial clock control circuit
$\overline{SCK}$
$\overline{CSIE}$ = 1 and
$\overline{SCK}$ pin output mode

**Fig. 5.6-17 SCK/P01 Pin Configuration**

The P01 output latch is mapped in address FF0H bit 1. When the $\overline{\text{RESET}}$ signal is generated, the P01 output latch is set to 1.

Cautions:
1. The P01 output latch must be set to 1 during normal serial transfer.
2. Do not use "PORT0.1" to specify the P01 output latch address. Write directly the address (FF0H.1) in operand. At that time, set MBE to 0, or set MBE to 1 and MBS to 15.

```
CLR1  PORT0.1 ⎫
              ⎬ Do not use
SET1  PORT0.1 ⎭

CLR1  0FF0H.1 ⎫
              ⎬ Use
SET1  0FF0H.1 ⎭
```

5.6.8 Serial interface application

The serial interface for each mode is explained using examples of applications.

The normal serial interface communication sequence is as follows:

1) Set transfer mode. (Set data in CSIM.)
2) Write data into SIO and give transfer start indication. (MOV SIO, XA or XCH XA, SIO. At that time, automatic transfer start indication is given.)
3) After checking that the serial interrupt routine or interrupt request flag (IRQCSI) is set, read receive data, and start transfer.

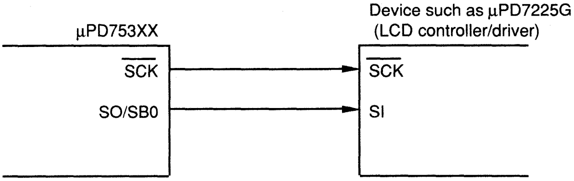
The SBI mode communication sequence is explained in detail in (3) below.

(1) 3-line serial I/O mode

(a) To transfer data starting at MBS using transfer clock of 262 kHz (at 4.19 MHz) (master operation).

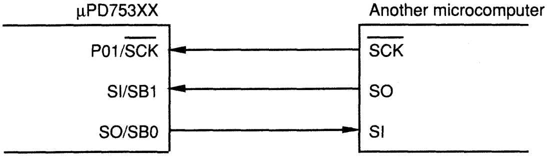
```
Example:  CLR1   MBE
          MOV    XA, #10000010B
          MOV    CSIM, XA        ; Transfer mode is set.
          MOV    XA, TDATA       ; TDATA is the transfer data storage address.
          MOV    SIO, XA         ; Transfer data is set.
                                 ; Transfer is started.
```

Caution:
At the second time or after, transfer can be started by setting data in SIO (MOV SIO, XA or XCH XA,  SIO).



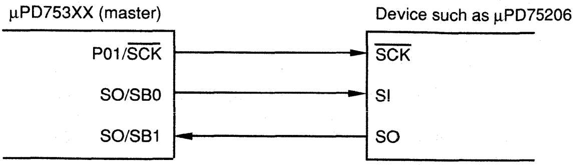
In this application example, the µPD753XX SI/SB1 pin can be used for input.

(b) To transfer data starting at the LSB using external clock (slave operation).
(In this example, the shift register read/write function to invert the LSB and MSB is used effectively.)



```
Example:  Main routine
          CLR1   MBE
          MOV    XA, #84H
          MOV    CSIM, XA        ; Serial operation stop, LSB/MSB inversion mode, external clock
          MOV    XA, TDATA
          MOV    SIO, XA         ; Transfer data is set.
                                 ; Transfer is started.
          EI     IECSI
          EI

          Interrupt routine (MBE = 0)
          MOV    XA, TDATA
          XCH    XA, SIO         ; Receive data - send data, transfer start
          MOV    RDATA, XA       ; Receive data is saved.
          RETI
```

(c) To transfer data at a high speed using a transfer clock of 524 kHz (at 4.19 MHz).



Example (master)

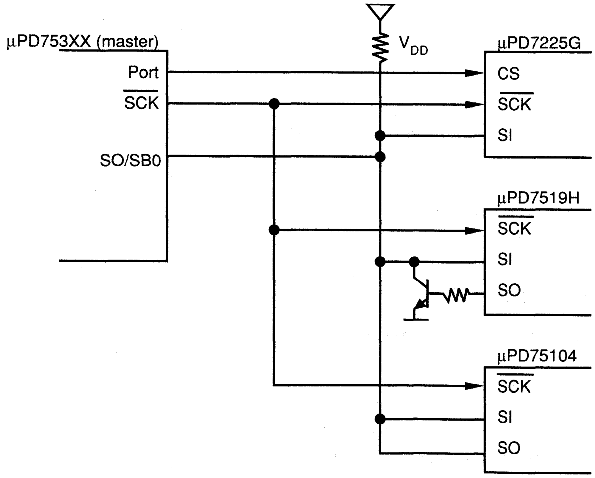
```
                CLR1    MBE
                MOV     XA, #10000011B
                MOV     CSIM, XA        ; Transfer mode is set.
                MOV     XA, TDATA
                MOV     SIO, XA         ; Transfer data is set, Transfer is started.
                  :
                  :
                  :
LOOP:           SKTCLR  IRQCSI          ; IRQCSI is tested.
                BR      LOOP
                MOV     XA, SIO         ; Receive data is read.
```

(2) 2-line serial I/O mode

The serial bus is formed and a number of devices are connected.

Example:

The example system consists of the µPD753XX as the master and µPD75104, µPD7519H, and µPD7225G connected as slaves.



To form a serial bus as in this example, FFH is previously written into the shift register, a high level is output to the shift register, a high level is output to the SO pin, the output buffers are turned off, and the bus is released except when the SI and SO pins are connected for serial data output.

Since the µPD7519H SO pin cannot be placed in the high impedance state, a transistor is connected for open collector output, as shown in the diagram above. At the time of data input, the transistor is turned off by previously writing 00H into the shift register.

The data output timing by the microcomputers is predetermined.

The serial clock is output by the master microcomputer µPD753XX; all other slave microcomputers operate on external clocks.

(3) SBI mode

An application example of serial data communication in the SBI mode is given. In the example, the μPD753XX can operate as a master or a slave CPU.

The master can also be changed by using a command.

(a) Serial bus configuration

The serial bus configuration in the application example given here assumes that the μPD753XX is connected to bus lines as one device in the serial bus.

The following two μPD753XX pins are used: Serial data bus SB0 (P02/SO) and serial clock $\overline{\text{SCK}}$ (P01).

Fig. 5.6-18 shows an example of a serial bus configuration.

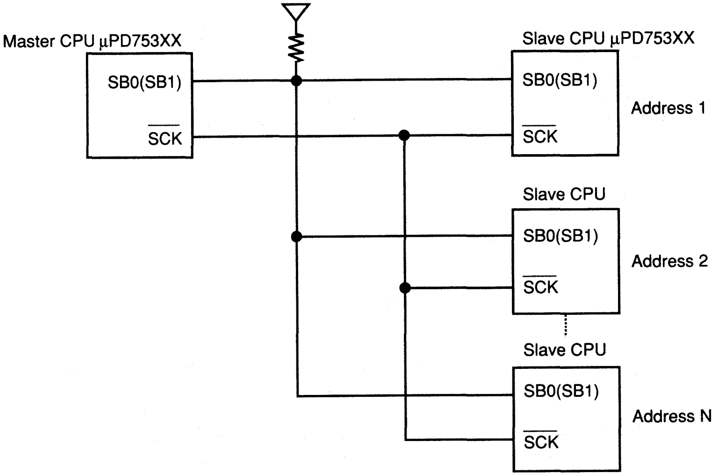


**Fig. 5.6-18 Serial Bus Configuration Example**

(b) Command explanation

Command types

The application example uses the following commands:

1) READ:
Data is transferred from slave to master.

2) WRITE:
Data is transferred from master to slave.

3) END:
WRITE command completion is reported to slave.

4) STOP:
WRITE command stop is reported to slave.

5) STATUS:
The slave state is read.

6) RESET:
The current slave being selected is made unselected.

7) CHGMST:
The master authorization is transferred to the slave.

Communication sequence

The communication sequence between the master and slave is as follows:

1) Communication is started by the master, which sends the address of the slave to communicate with and selects the slave (chip select).
The slave which receives the address returns $\overline{\text{ACK}}$ and communicates with the master. (The slave is placed in selected state.)

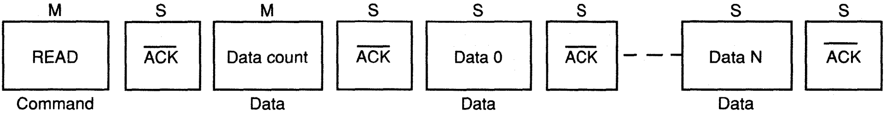
2) Command and data are transferred between the slave selected in 1 and the master.
Since command and data are transferred point-to-point (between the master and specific slave), other slaves must be deselected.

3) Communication terminates when the slave is deselected in either of the following cases:
— When the master sends the $\overline{\text{RESET}}$ command, the selected slave is deselected.
— If the master is changed by using the CHGMST command, the device changed from master to slave is deselected.

Command format

The command transfer formats are shown below:

1) READ command
The READ command reads data from a given slave. The read data count ranges from one to 256 bytes. The master specifies the data count in a parameter. If 00H is specified for the data count, 256-byte data transfer is assumed to be specified.

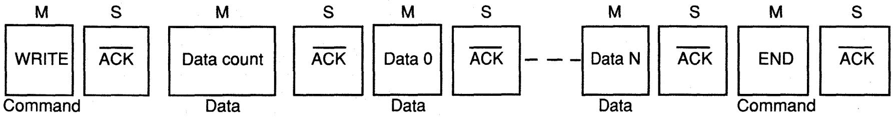


M: Output by master
S: Output by slave

**Fig. 5.6-19 READ Command Transfer Format**

After receiving the data count, the slave returns $\overline{\text{ACK}}$ if the number of data bytes that can be sent is greater than the data count. If it is less than the data count, the slave does not return $\overline{\text{ACK}}$, resulting in an error.
During data transfer, the slave compares the SIO contents before and after data transfer to check that data has been output to the bus normally. If the SIO contents before and after transfer do not match, the slave does not return $\overline{\text{ACK}}$, resulting in an error.

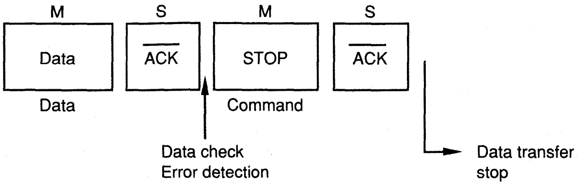
2) WRITE, END, and STOP commands
The WRITE, END, and STOP commands are used to write data into a particulary slave. The write data count ranges from 1 to 256 bytes. The master specifies the data count in a parameter. If 00H is specified for the data count, 256-byte data transfer is assumed to be specified.



M: Output by master
S: Output by slave

**Fig. 5.6-20 WRITE, END Command Transfer Format**

After receiving the data count, the slave returns $\overline{\text{ACK}}$ if the receive data store area is larger than the data count. If it is less than the data count, it does not return $\overline{\text{ACK}}$, resulting in an error.
At the termination of all data transfer, the master sends the END command to the slave. It signals that all data has been transferred normally.
The slave also receives an END command before it completes all data reception. In this case, the data which has been received immediately before the END command is received becomes valid. During data sending, the master compares the SIO contents before and after data sending to check that data has been output to the bus normally. If the SO contents before and after sending do not match, the master sends the STOP command and stops data transfer.

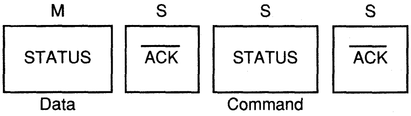


M: Output by master
S: Output by slave

**Fig. 5.6-21 STOP Command Transfer Format**

When receiving the STOP command, the slave invalidates the 1-byte data received immediately before receiving the STOP command.

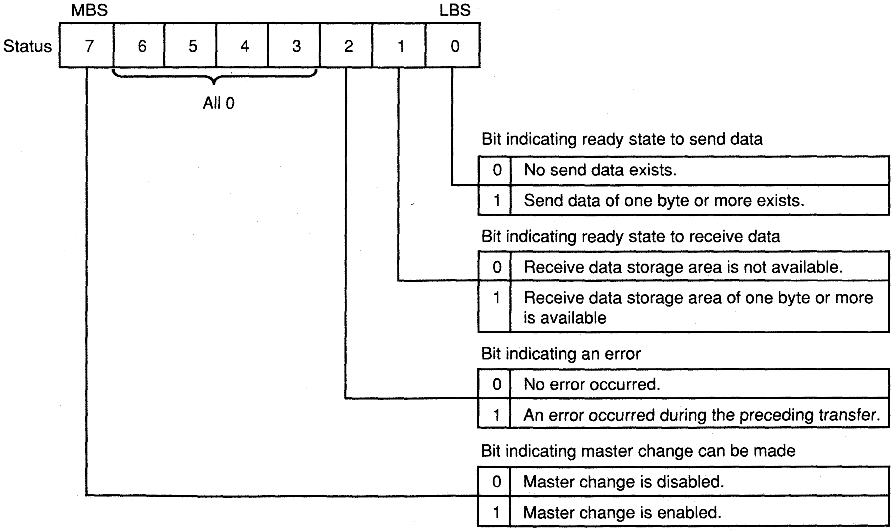
3) STATUS command
The STATUS Command Transfer Format



M: Output by master
S: Output by slave

**Fig. 5.6-22 STATUS Command Status Format**

Fig. 5.6-23 shows the format of the status returned by the slave.



Bit indicating ready state to send data

| | |
|---|---|
| 0 | No send data exists. |
| 1 | Send data of one byte or more exists. |

Bit indicating ready state to receive data

| | |
|---|---|
| 0 | Receive data storage area is not available. |
| 1 | Receive data storage area of one byte or more is available |

Bit indicating an error

| | |
|---|---|
| 0 | No error occurred. |
| 1 | An error occurred during the preceding transfer. |

Bit indicating master change can be made

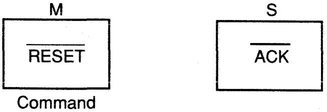
| | |
|---|---|
| 0 | Master change is disabled. |
| 1 | Master change is enabled. |

**Fig. 5.6-23 STATUS Command Status Format**

To send status data, the slave compares the contents before and after sending status data. If they do not match, the slave does not return $\overline{\text{ACK}}$, resulting in an error.

4) $\overline{\text{RESET}}$ command

The $\overline{\text{RESET}}$ command is used to cause the currently selected slave to be selected. When the $\overline{\text{RESET}}$ command is issued, all slaves can be deselected.

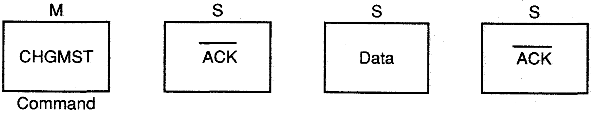


M: Output by  master
S: Output ba slave

**Fig. 5.6-24 $\overline{\text{RESET}}$ Command Transfer Format**

5) CHGMST command

The CHGMST command transfers the master authorization to the currently selected slave.



M: Output by master
S: Output by slave

**Fig. 5.6-25 CHGMST Command Transfer Format**

When receiving the CHGMST command, the slave decides whether or not it can receive the master authorization, and returns either of the following data to the master:

— 0FFH: Master change is enabled.
— 00H  : Master change is disabled.

When transferring data, the slave compares the SIO contents before and after data transfer. If they do not match, the slave does not return $\overline{\text{ACK}}$, resulting in an error.

If no error occurs, the master serves as a slave after 0FFH data sending is complete. If no error occurs, the slave serves as the master after 0FFH data sending is completed.

Error occurrence

When a communication error occurs, the master and slave operate as explained below:

The slave informs the master of error occurrence by returning no $\overline{\text{ACK}}$. When an error occurs, the status bit (bit 2) indicating error occurence is set to 1 and all command processing being performed is cancelled.

After completing the sending or receiving of one byte, the master checks whether or not $\overline{\text{ACK}}$ is returned from slave. If $\overline{\text{ACK}}$ is not returned from slave within a given period after sending or receiving is completed, the master decides that an error has occurred and outputs a dummy $\overline{\text{ACK}}$ signal.

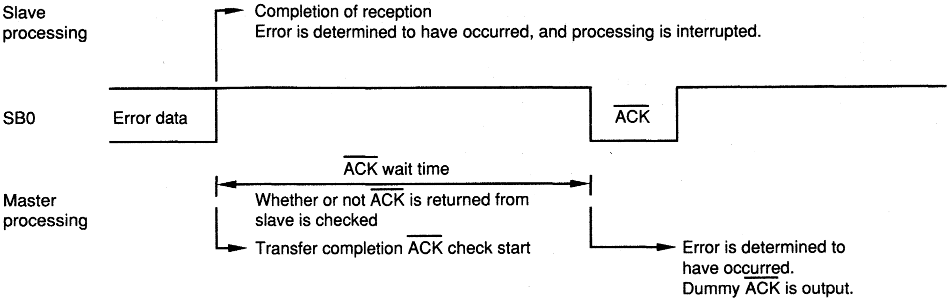

Slave processing
Completion of reception
Error is determined to have occurred, and processing is interrupted.
SB0
Error data
$\overline{\text{ACK}}$
$\overline{\text{ACK}}$ wait time
Master processing
Whether or not $\overline{\text{ACK}}$ is returned from slave is checked
Transfer completion $\overline{\text{ACK}}$ check start
Error is determined to have occurred.
Dummy $\overline{\text{ACK}}$ is output.

**Fig. 5.6-26 Master and Slave Operation when Error Occured**

The following errors are possible:

— Errors that may occur in the slave

1) Command transfer format is erroneous.
2) Undefined command is received.
3) The number of data bytes to be transferred (data count) is insufficient during READ command execution,
4) Data storage area is insufficient during WRITE command execution.
5) When READ, STATUS, or CHGMST command data is sent, data changes.

When any error in 1) or 5) occurs, no ACK is returned.

— Errors that may occur in the master

When WRITE command data is sent, if data changes, STOP command is sent to the slave.

**5.7 LCD Controller/Driver**

5.7.1 LCD controller/driver configuration

The µPD753XX incorporates a display controller which generates segment and common signals according to the display data memory contents and incorporates segment and common drivers which can drive the panel directly.

Fig. 5.7-1 shows the LCD controller/driver configuration.

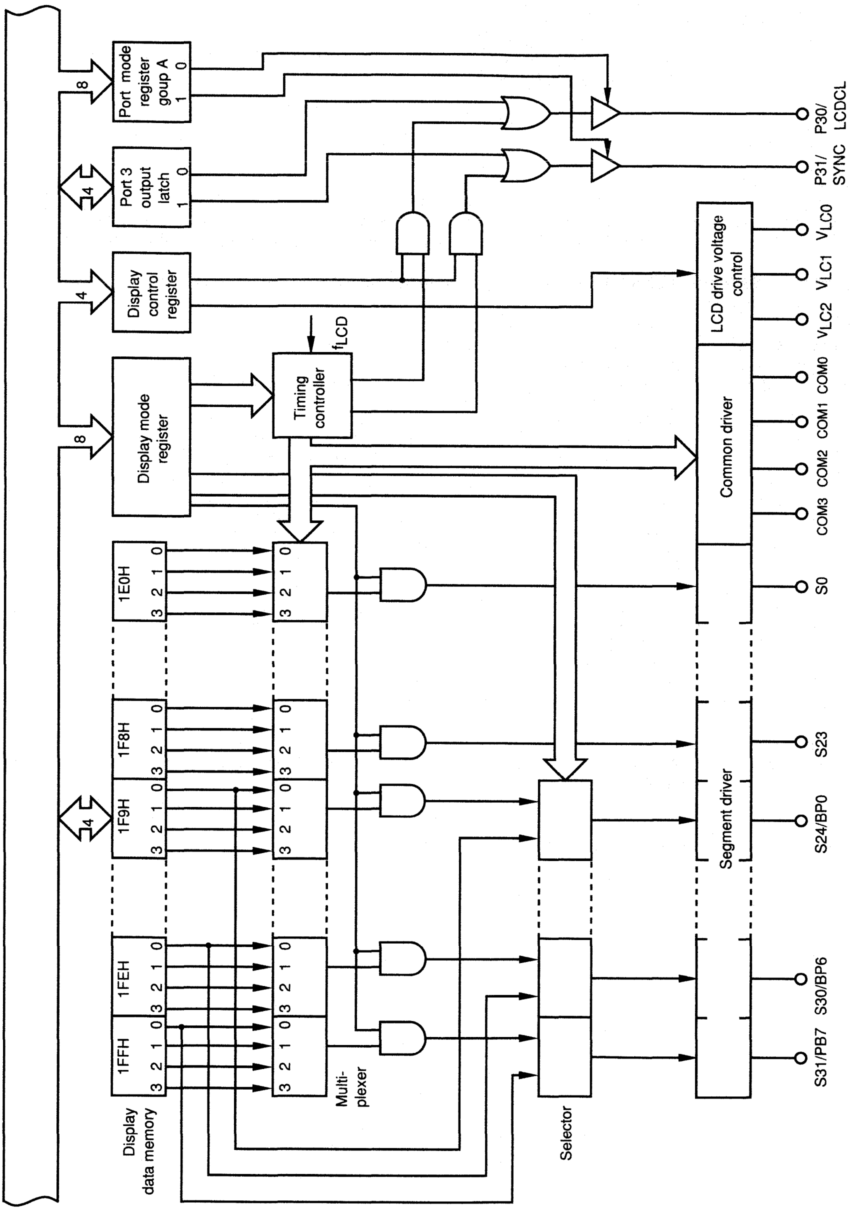


**Fig. 5.7-1 LCD Controller/Driver Block Diagram**

]

5.7.2 LCD controller/driver functions

The µPD753XX LCD controller/driver functions are as follows:

(a) Display data memory is read automatically by DMA operation and segment and common signals are generated.

(b) Display mode can be selected from among the following five.
  1) Static
  2) 1/2 duty (1/2 bias)
  3) 1/3 duty (1/2 bias)
  4) 1/3 duty (1/3 bias)
  5) 1/4 duty (1/3 bias)

(c) A frame frequency can be selected from among four in each display mode.

(d) A maximum of 32 segment signal output pins (S0-S31) and four common signal output pins (COM0 - COM3).

(e) The segment signal output pins (S24-S27 and S28-S31) can be changed to the output ports (BP0-BP3 and BP4-BP7) in 4-pin units.

(f) Split-resistor can be incorporated to supply LCD drive power. (Mask option)
  — Various bias laws and LCD drive voltage can be handled.
  — When display is off, current flow to the split resistor is cut.

(g) Display data memory not used for display can be used for normal data memory.

(h) It can also operate by using the subsystem clock.

Table 5.7-1 lists the maximum number of picture elements that can be displayed in each display mode.

**Table 5.7-1 Maximum Number of Displayed Picture Elements**

| Bias law | Time division | Used COMMON signals | Maximum number of picture elements |
|---|---|---|---|
| — | Static | COM0 (COM1, 2, 3) | 32 (segment 32 x common 1) (Note 1) |
| 1/2 | 2 | COM0, 1 | 64 (segment 32 x common 2) (Note 2) |
|  | 3 | COM0, 1, 2 | 96 (segment 32 x common 3) (Note 3) |
| 1/3 | 3 |  |  |
|  | 4 | COM0, 1, 2, 3 | 128 (segment 32 x common 4) (Note 4) |

Note 1: 4 digits (eight segment signal/digit) on LCD panel (8. display).
Note 2: 8 digits (four segment signal/digit) on LCD panel (8. display).
Note 3: 10 digits (three segment signal/digit) on LCD panel (8. display).
Note 4: 16 digits (two segment signal/digit) on LCD panel (8. display).

5.7.3 Display mode register

The display mode register (LCDM) consists of eight bits to specify the display mode, LCD clock, frame frequency, segment or bit port output, and display output on/off control.

LCDM is set by using 8-bit memory operation instruction. Only bit 3 (LCDM3) can be set and cleared by using bit operation (manipulation) instructions.

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

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| F8CH | LCDM7 | LCDM6 | LCDM5 | LCDM4 | LCDM3 | LCDM2 | LCDM1 | LCDM0 | LCDM |

Display Mode Selection

| LCDM3 | LCDM2 | LCDM1 | LCDM0 | Time division value | Bias method |
|---|---|---|---|---|---|
| 0 | X | X | X | Display off (Note) | |
| 1 | 0 | 0 | 0 | 4 | 1/3 |
| 1 | 0 | 0 | 1 | 3 | 1/3 |
| 1 | 0 | 1 | 0 | 2 | 1/2 |
| 1 | 0 | 1 | 1 | 3 | 1/2 |
| 1 | 1 | 0 | 0 | Static | |
| Other than the above | | | | Undefined | |

Note: All segment signals are unselected.

LCD Clock Selection

| LCDM5 | LCDM4 | LCDMCL |
|---|---|---|
| 0 | 0 | $f_W/2^9$ (64 Hz) |
| 0 | 1 | $f_W/2^8$ (128 Hz) |
| 1 | 0 | $f_W/2^7$ (256 Hz) |
| 1 | 1 | $f_W/2^6$ (512 Hz) |

Caution: LCDCL is supplied only when the watch timer operates. To use the LCD controller, bit 2 of watch mode register WM should be set to 1.

Segment and Bit Port Output Change Specification

| LCDM7 | LCDM6 | S24/S27 | S28/S31 | Number of segment output pins | Number of bit port output pins |
|---|---|---|---|---|---|
| 0 | 0 | Segment output | Segment output | 32 | 0 |
| 0 | 1 | Segment output | Bit port output | 28 | 4 |
| 1 | 0 | Bit port output | Segment output | 28 | 4 |
| 1 | 1 | Bit port output | Bit port output | 24 | 8 |

Frame Frequency (Hz)

| LCDCL / Display duty cycle | $f_W/2^9$ (64 Hz) | $f_W/2^8$ (128 Hz) | $f_W/2^7$ (256 Hz) | $f_W/2^6$ (512 Hz) |
|---|---|---|---|---|
| Static | 64 | 128 | 256 | 512 |
| 1/2 | 32 | 64 | 128 | 256 |
| 1/3 | 21 | 43 | 85 | 171 |
| 1/4 | 16 | 32 | 64 | 128 |

When $f_W$ = 32.768 kHz $f_W$: Input clock to watch timer ($f_X$/128 or $f_{XT}$)

**Fig. 5.7-2 Display Mode Register Format**

5.7.4 Display control register

The display control register controls LCD drive as follows:

— Enables or disables common and segment output.

— Cuts current flow to the split resistor for the LCD power supply.

— Enables or disables output of the synchronizing clock (LCDCL) and synchronizing signal (SYNC) to the controller/driver for external segment signal extension.

LCDCL is set by using a 4-bit memory operation instruction.

When the RESET signal is generated, the display control register is cleared.

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| F8EH | 0 | LCDC2 | 0 | LCDC0 | LCDC |

| LCDC0 | 0 | 1 | |
|---|---|---|---|
| LCDM3 | X | 0 | 1 |
| COM0-3 | Low level is output (display off). | Common signal corresponding to the display mode is output. | Common signal corresponding to the display mode is output. |
| S0-S23<br>S24-S31 segment specification pins | Low level is output (display off). | Segment signal corresponding to the display mode is output (no-selection level output, display off). | Segment signal corresponding to the display mode is output (display on). |
| S24-S31 bit port specification pins | Bit 0 for the corresponding display data memory is output (bit port function). | Bit 0 for the corresponding display data memory is output (bit port function). | Bit 0 for the corresponding display data memory is output (bit port function). |
| Power supply to split resistor (BIAS pin output) | Off<br>(high impedance) | On<br>(high level) | On<br>(high level) |

Note 1: ( ): When internal split register for LCD drive power supply is not contained.

LCDCL and SYNC Signal Output Enable/Disable Specification Bit

| | | |
|---|---|---|
| LCDCL2 | 0 | LCDCL and SYNC signal output is disabled. |
| | 1 | LCDCL and SYNC signal output is enabled. |

Note: LCDCL, SYNC signal output are provided for future system extension. Disable signal output at present.

**Fig. 5.7-3 Display Control Register Format**

5.7.5 Display data memory

The display data memory is mapped in 1E0H-1FFH.

The display data memory is an area read by the LCD controller/driver, which performs DMA operation independently of CPU operation. The LCD controller controls the segment signals according to data in the display data memory. When S24-S31 are used for bit ports, bit 0 of the data written into display data memory addresses 1F8H-1FFH is output from each bit port output pin.

The area not used for LCD display or ports can be used for normal data memory.

The display data memory is handled in 1- or 4-bit units. It cannot be handled in 8-bit units.

Fig. 5.7-5 shows the relationship between the display data memory bits and segment output/bit port output.

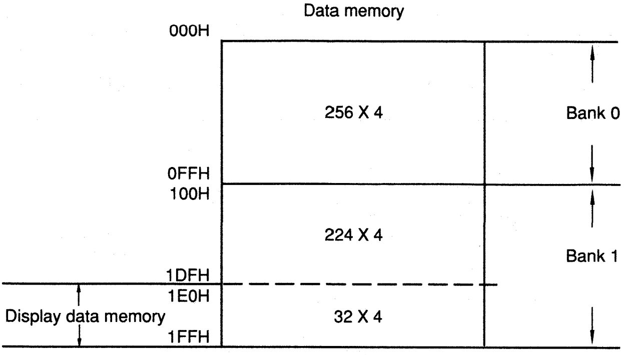


**Fig. 5.7-4 Data Memory Map**

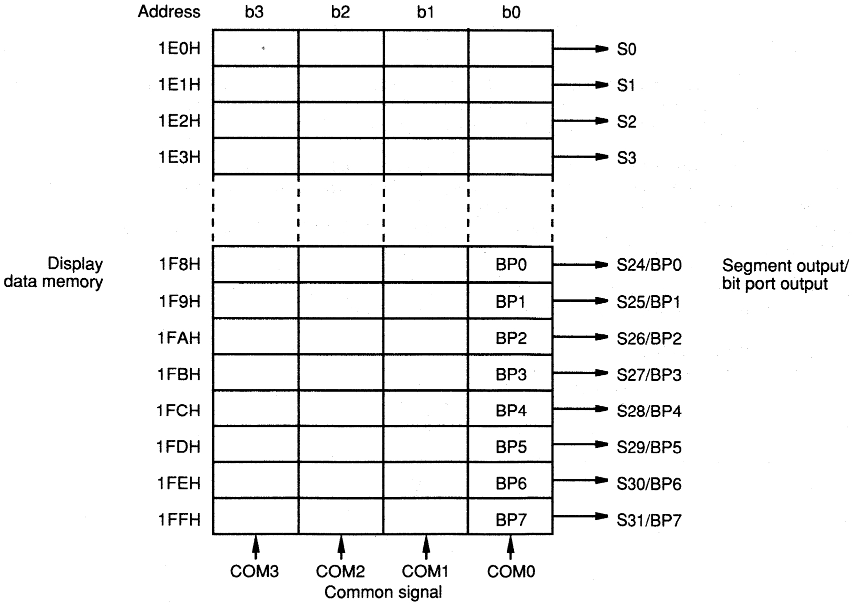


**Fig. 5.7-5 Relationship between Display Data Memory and Common Segment**

5.7.6 Common and segment signals

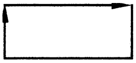
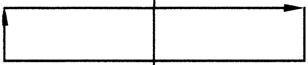
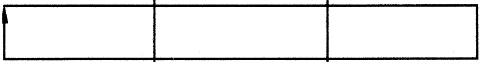
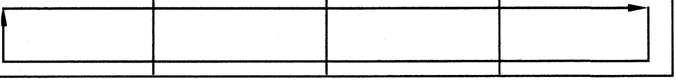
Each picture element of the LCD panel goes on when the potential difference between its corresponding common signal and segment signal reaches or exceeds a given voltage (LCD drive voltage VLCD). It goes off when the potential difference falls below VLCD or reaches 0V.

Since deterioration occurs when DC potential is maintained as the common and segment signals, the LCD panel is driven by AC.

(1) Common signals

The common signals become the selection timing in the order listed in Table 5.7-2 according to the set time division number, and repetitive operation is performed by using them as one period. In the static mode, COM0 to COM3 output the same signal. In division by 2, neither COM2 nor COM3 pin should be connected. In division by 3, do not connect the COM3 pin.

**Table 5.7-2 COM Signals**

| Time division number \ COM signal | COM0 | COM1 | COM2 | COM3 |
|---|---|---|---|---|
| Static |  | | | |
| 2 |  | | No connection required | No connection required |
| 3 |  | | | No connection required |
| 4 |  | | | |

(2) Segment signals

There are 32 segment signals corresponding to 32 locations of the display data area (1E0H-1FFH) of the data memory. Bits 0 to 3 of each location are automatically read in synchronization with the selection timing for COM0 to COM3 respectively. If the bit is set to 1, it produces the segment selection voltage; if 0, the bit produces the no-selection voltage output from the corresponding segment pin (S0-S31).

Thus, check the display patterns formed according to the combinations of the LCD panel front electrode (corresponding to segment signal) and rear electrode (corresponding to common signal). Then, on a one-to-one basis, write into the display area the bit data which corresponds to the pattern to be displayed.

Display data area bits 1 to 3 in the static mode, bits 2 and 3 in the division by 2 mode, and bit 3 in the division by 3 mode are not accessed and can be used for purposes other than display.

(3) Common and segment signal output waveforms

Tables 5.7-3 to 5.7-5 list voltages output to the common and segment signals. +VLCD/−VLCD on voltage is applied only when both signals become selection voltages; otherwise, the off voltage is applied.

**Table 5.7-3 LCD Drive Voltage (Static)**

| Common signal COM0 \ Segment signal Sn | Selection | No-selection |
|---|---|---|
| | $V_{LC0}/V_{SS}$ | $V_{SS}/V_{LC0}$ |
| $V_{SS}/V_{LC0}$ | $+V_{LCD}/-V_{LCD}$ | 0V/0V |

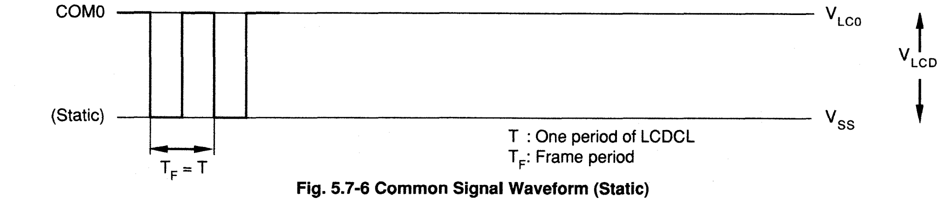
**Table 5.7-4 LCD Drive Voltage (1/2 Bias Law)**

| Common signal COMm \ Segment signal Sn | | Selection | No-selection |
|---|---|---|---|
| | | $V_{LC0}/V_{SS}$ | $V_{SS}/V_{LC0}$ |
| Selection | $V_{SS}/V_{LC0}$ | $\boxed{+V_{LCD}/-V_{LCD}}$ | 0V/0V |
| No-selection | $V_{LC1} = V_{LC2}$ | $+1/2\ V_{LCD}/-1/2\ V_{LCD}$ | $-1/2\ V_{LCD}/+1/2\ V_{LCD}$ |

**Table 5.7-5 LCD Drive Voltage (1/3 Bias Law)**

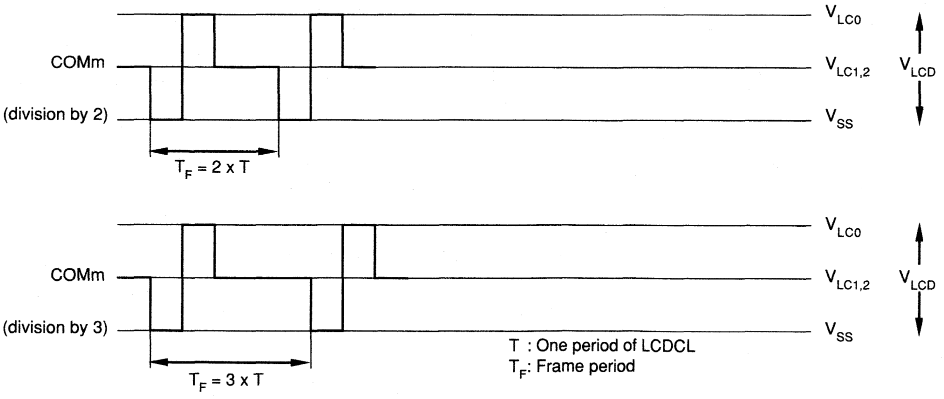
| Common signal COMm \ Segment signal Sn | | Selection | No-selection |
|---|---|---|---|
| | | $V_{LC0}/V_{SS}$ | $V_{LC2}/V_{LC1}$ |
| Selection | $V_{SS}/V_{LC0}$ | $\boxed{+V_{LCD}/-V_{LCD}}$ | $+1/3\ V_{LCD}/-1/3\ V_{LCD}$ |
| No-selection | $V_{LC1}/V_{LC2}$ | $+1/3\ V_{LCD}/-1/3\ V_{LCD}$ | $+1/3\ V_{LCD}/-1/3\ V_{LCD}$ |

Figs. 5.7-6 to 5.7-8 shows the common signal waveforms. Fig. 5.7-9 shows the common and segment signal voltages and phases.



T : One period of LCDCL
$T_F$: Frame period

**Fig. 5.7-6 Common Signal Waveform (Static)**



T : One period of LCDCL
$T_F$: Frame period

**Fig. 5.7-7 Common Signal Waveform (1/2 Bias Law)**

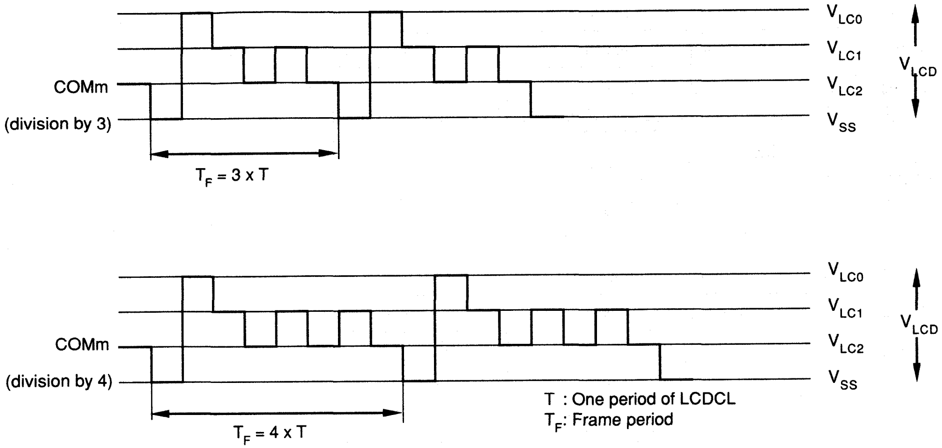


**Fig. 5.7-8 Common Signal Waveform (1/3 Bias Law)**

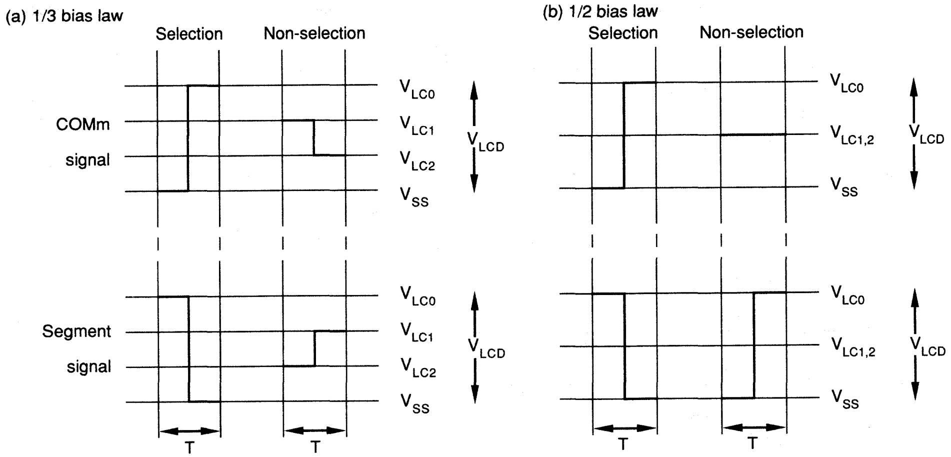


**Fig. 5.7-9 Common and Segment Signal Voltages and Phases**

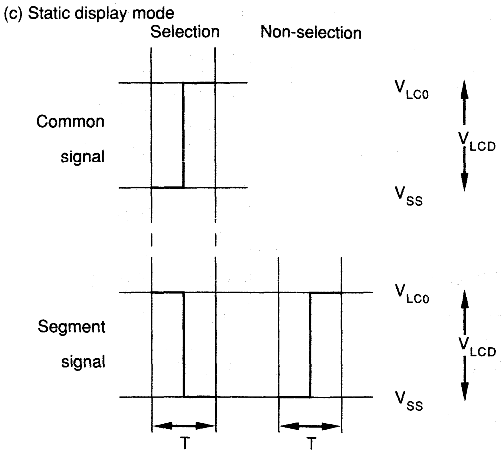


**Fig. 5.7-9 Common and Segment Signal Voltages and Phases (cont'd)**

5.7.7 Supply of LCD drive power $V_{LC0}$, $V_{LC1}$, and $V_{LC2}$

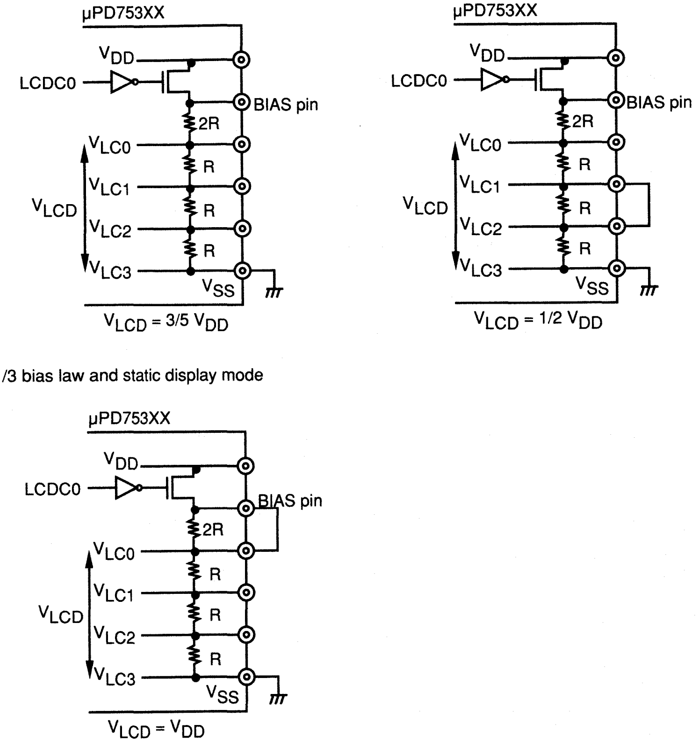
In the μPD753XX, a split resistor can be incorporated in the $V_{LC0}$-$V_{LC2}$ pins for the LCD drive power supply. According to the bias law, the LCD drive power can be supplied without an external split resistor. The μPD753XX also includes the BIAS pin to deal with various LCD drive voltages. The BIAS and $V_{LC0}$ pins are connected externally.
Table 5.7-6 lists proper LCD drive power supply values based on the static, 1/2, and 1/3 bias laws.

**Table 5.7-6 LCD Drive Power Supply Values**

| Bias law / LCD drive power | No bias (static mode) | 1/2 | 1/3 |
|---|---|---|---|
| $V_{LC0}$ | $V_{LCD}$ | $V_{LCD}$ | $V_{LCD}$ |
| $V_{LC1}$ | 2/3 $V_{LCD}$ | 1/2 $V_{LCD}$ (Note 1) | 2/3 $V_{LCD}$ |
| $V_{LC2}$ | 1/3 $V_{LCD}$ | | 1/3 $V_{LCD}$ |
| $V_{LC3}$ | 0V | 0V | 0V |

Note 1: When 1/2 bias is used, the VLC1 and VLC2 pins must be connected externally.
Note 2: When the BIAS and $V_{LC0}$ pins are not connected, $V_{LCD}$ = 3/5 $V_{DD}$ (when split resistor is incorporated). When the BIAS and $V_{LC0}$ pins are connected, $V_{LCD}$ = $V_{DD}$.

In Fig. 5.7-10, (a) to (c) show LCD drive power supply examples according to Table 5.7-6. Fig. 5.7-11 (d) shows an example in which external resistor is connected to the LCD drive voltage pins ($V_{LC0}$–$V_{LC2}$) and BIAS pin, and internal split resistor is fine-adjusted.
Current flow through the split resistor can also be cut by clearing display control register bit 0 (LCDC0).
This LCD power on/off control is also useful to prevent DC voltage from being applied to LCD when the LCD clock is stopped by execution of a STOP instruction and when the watch timer operates using the main system clock.
That is, display control register bit 0 (LCDC0) is cleared and all LCD drive power sources are placed in the same potential VSS immediatly before the STOP instruction is executed, thereby suppressing the potential difference between the LCD electrodes even if the LCD clock is stopped.
When the watch timer operates using the subsystem clock, the LCD display can still be used.



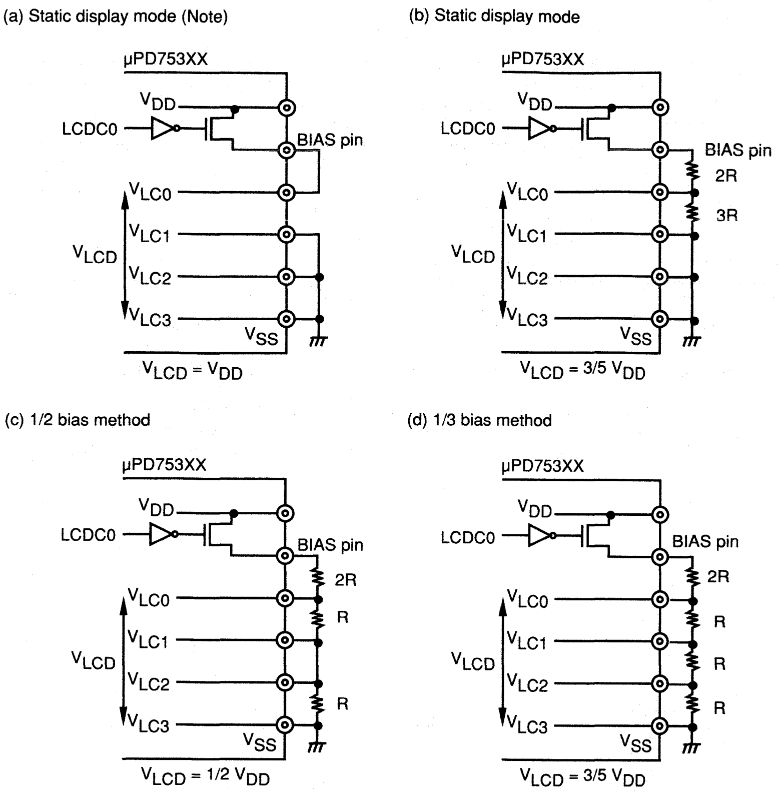
(a) 1/3 bias law and static mode display

(b) 1/2 bias law

(c) 1/3 bias law and static display mode

In example (a), $V_{DD}$ = 5V and LCD drive voltage $V_{LCD}$ = 3V.
In example (b), $V_{DD}$ = 5V and $V_{LCD}$ = 2.5V.
In example (c), $V_{DD}$ = 5V and $V_{LCD}$ = 5V.

**Fig. 5.7-10 LCD Drive Power Connection Examples (when split resistor is incorporated)**



In example (a), $V_{DD}$ = 5V and LCD drive voltage $V_{LCD}$ = 5V.
In example (b) and (d), $V_{DD}$ = 5V and LCD drive voltage $V_{LCD}$ = 3V.
In example (c), $V_{DD}$ = 5V and $V_{LCD}$ = 2.5V.

Note: Be sure to set LCDC0 to 1 (also during the standby mode)

**Fig. 5.7-11 LCD Drive Power Connection Examples (when split resistor is corporated externally)**

5.7.8 Display mode

(1) Static display example

Fig. 5.7-14 shows connection of a static 4-digit LCD panel having the display pattern shown in Fig. 5.7-13, the µPD753XX segment signals (S0-S31), and the common signal (COM0). In this example, 123.4 is displayed. The contents of the display data memory (addresses 1E0H-1FFH) correspond to the display pattern.

Here, 3. (3 .) at the second digit position is taken as an example. It is necessary to output selection and no-selection voltages as shown in Fig. 5.7-12 to the S8-S15 pins at the COM0 common signal timing according to the display pattern shown in Fig. 5.7-13.

| Segment / Common | S8 | S9 | S10 | S11 | S12 | S13 | S14 | S15 |
|---|---|---|---|---|---|---|---|---|
| COM0 | Selection | Selection | Selection | Selection | No-selection | Selection | No-selection | Selection |

**Fig. 5.7-12 Selection/Non-selection Voltages of S8-S15 Pins (Static Display Examples)**

This shows that the bit 0 string of display data memory addresses 1E8H-1EFH corresponding to S8-S15 needs to be set to 11110101.

Fig. 5.7-15 shows the S11, S12, and COM0 LCD drive waveforms. This shows that alternating current square wave of $+V_{LCD}/-V_{LCD}$ that is LCD on level is generated when S11 becomes the selection voltage at the selection timing of COM0. Since the same waveform as COM0 is output to COM1 to COM3, the drive capability can be increased by connecting COM0, COM1, COM2, and COM3.

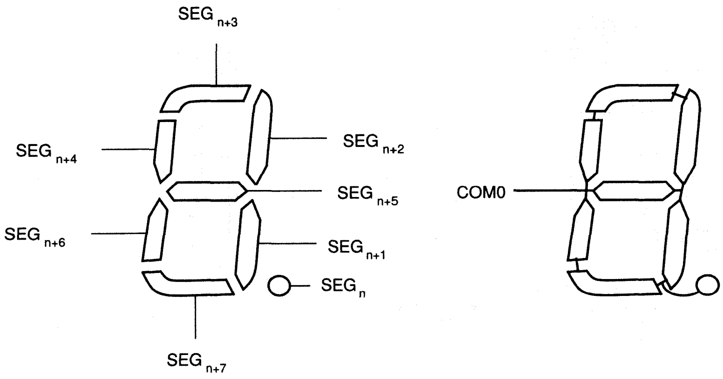
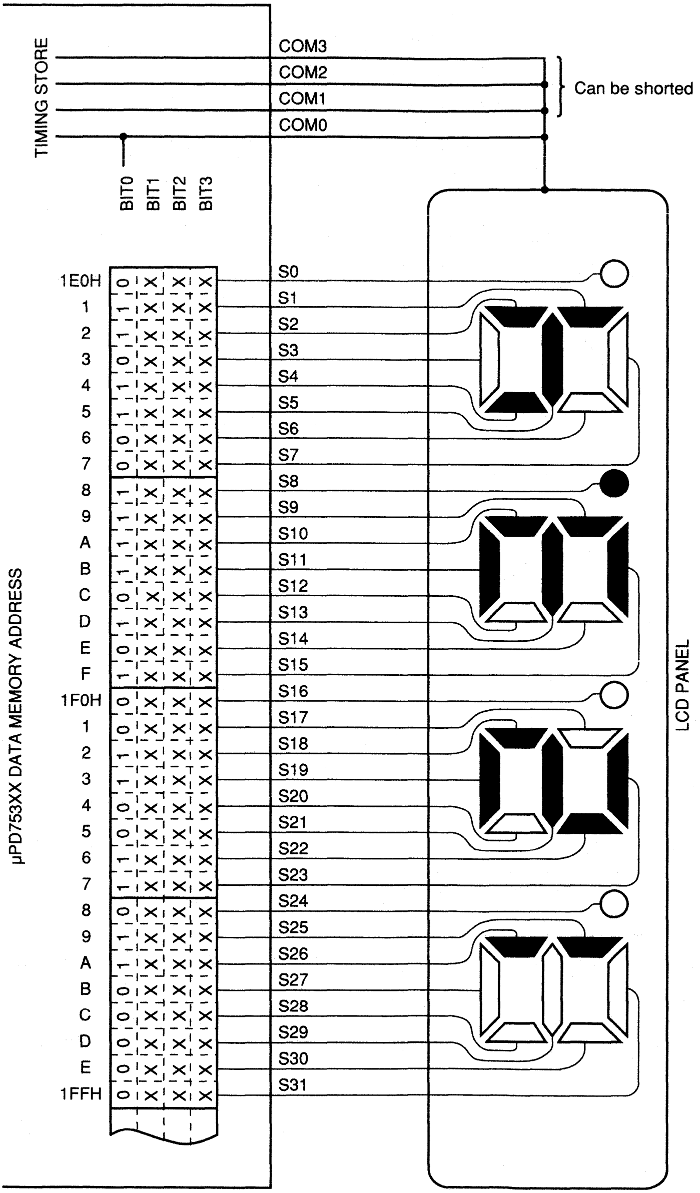


**Fig. 5.7-13 Static Mode LCD Display Pattern and Electrode Connection**



| | BIT0 | BIT1 | BIT2 | BIT3 |
|---|---|---|---|---|
| 1E0H | 0 | X | X | X |
| 1 | 1 | X | X | X |
| 2 | 1 | X | X | X |
| 3 | 0 | X | X | X |
| 4 | 1 | X | X | X |
| 5 | 1 | X | X | X |
| 6 | 0 | X | X | X |
| 7 | 0 | X | X | X |
| 8 | 1 | X | X | X |
| 9 | 1 | X | X | X |
| A | 1 | X | X | X |
| B | 1 | X | X | X |
| C | 0 | X | X | X |
| D | 1 | X | X | X |
| E | 0 | X | X | X |
| F | 1 | X | X | X |
| 1F0H | 0 | X | X | X |
| 1 | 0 | X | X | X |
| 2 | 1 | X | X | X |
| 3 | 1 | X | X | X |
| 4 | 0 | X | X | X |
| 5 | 0 | X | X | X |
| 6 | 1 | X | X | X |
| 7 | 1 | X | X | X |
| 8 | 0 | X | X | X |
| 9 | 1 | X | X | X |
| A | 1 | X | X | X |
| B | 0 | X | X | X |
| C | 0 | X | X | X |
| D | 0 | X | X | X |
| E | 0 | X | X | X |
| 1FFH | 0 | X | X | X |

**Fig. 5.7-14 Static LCD panel Connection Example**

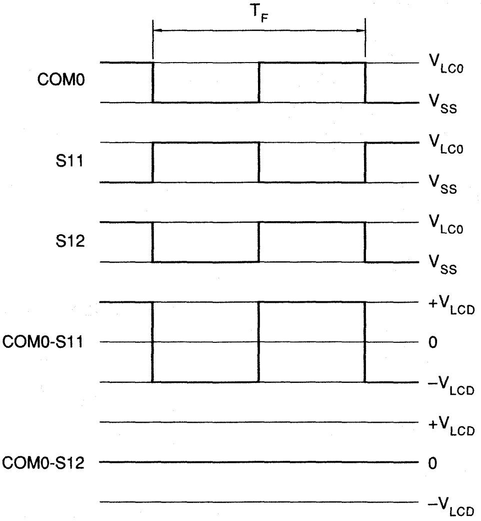


**Fig. 5.7-15 Static LCD Drive Waveform Example**

(2) Division by 2 display example

Fig. 5.7-18 shows connection of the division by 2 mode 8-digit LCD panel having the display pattern shown in Fig. 5.7-17, the μPD753XX segment signals (S0-S31), and the common signals (COM0 and COM1). In the example, 123456.78 is displayed. The contents of the display data memory (addresses 1E0H-1FFH) correspond to the display pattern.

Here, 6. ( 6.) at the third digit position is taken as an example. It is necessary to output the selection and no-selection voltages as shown in Fig. 5.7-16 to the S8-11 pins at the timing of the COM0 and COM1 common signal according to the display pattern shown in Fig. 5.7-17.

| Segment / Common | S8 | S9 | S10 | S11 |
|---|---|---|---|---|
| COM0 | Selection | No-selection | Selection | Selection |
| COM1 | Selection | Selection | Selection | Selection |

**Fig. 5.7-16 Selection/Non-selection Voltages of S8-S11 Pins (Division by 2 Mode Display Examples)**

This shows that for example, the display data memory address 1F9H bits corresponding to S9 need to be set to xx10.

Fig. 5.7-19 shows an LCD drive waveform example among S9 and common signals. This shows an alternating current square wave of $+V_{LCD}/-V_{LCD}$ that is the LCD on level being generated when S9 is the selection voltage at the COM1 selection timing.

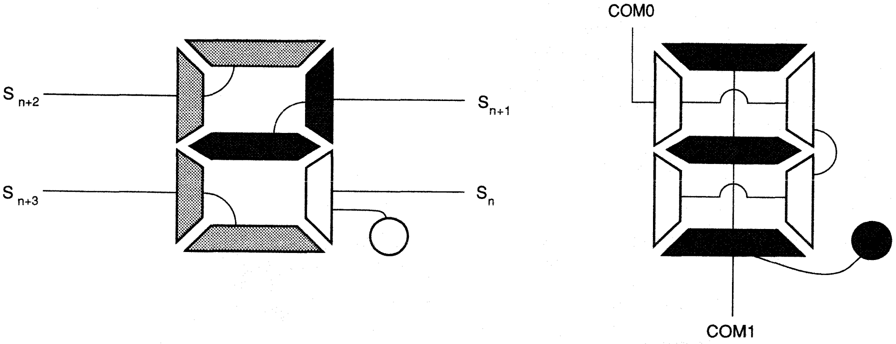
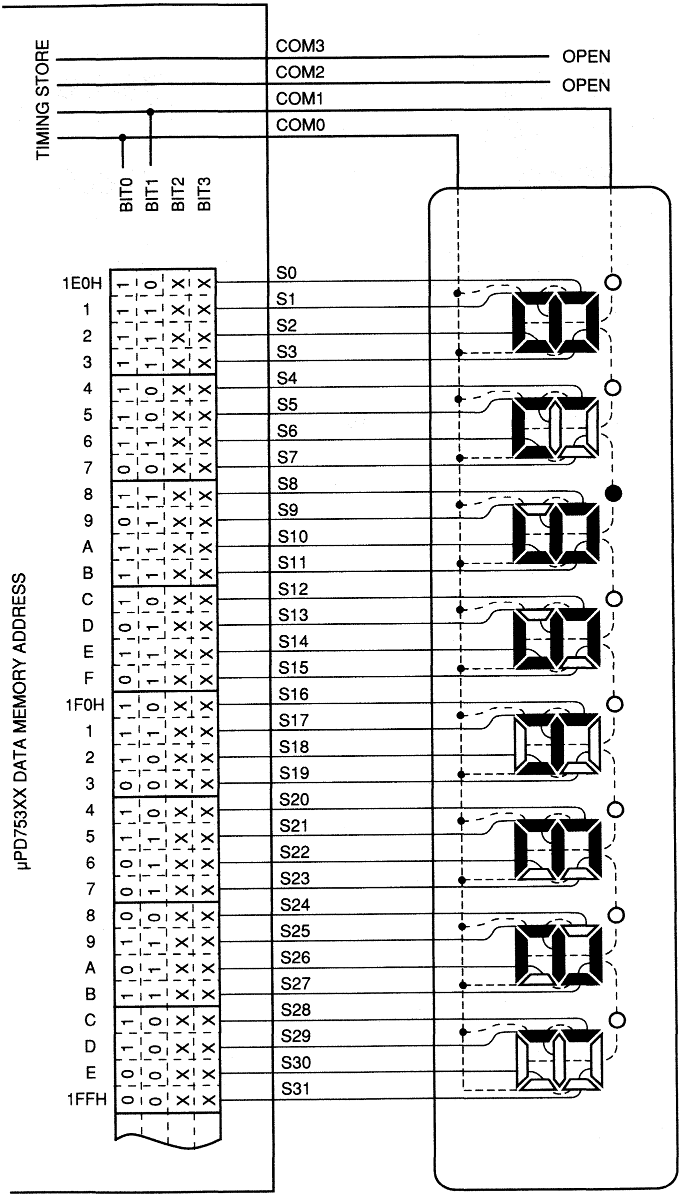


**Fig. 5.7-17 Division by 2 Mode LCD Display Pattern and Electrode Connection**



| | BIT0 | BIT1 | BIT2 | BIT3 |
|---|---|---|---|---|
| 1E0H | 1 | 0 | X | X |
| 1 | 1 | 1 | X | X |
| 2 | 1 | 1 | X | X |
| 3 | 1 | 1 | X | X |
| 4 | 1 | 0 | X | X |
| 5 | 1 | 0 | X | X |
| 6 | 1 | 1 | X | X |
| 7 | 0 | 0 | X | X |
| 8 | 1 | 1 | X | X |
| 9 | 0 | 1 | X | X |
| A | 1 | 1 | X | X |
| B | 1 | 1 | X | X |
| C | 1 | 0 | X | X |
| D | 0 | 1 | X | X |
| E | 1 | 1 | X | X |
| F | 0 | 1 | X | X |
| 1F0H | 1 | 0 | X | X |
| 1 | 1 | 1 | X | X |
| 2 | 1 | 0 | X | X |
| 3 | 0 | 0 | X | X |
| 4 | 1 | 0 | X | X |
| 5 | 1 | 1 | X | X |
| 6 | 0 | 1 | X | X |
| 7 | 0 | 1 | X | X |
| 8 | 0 | 0 | X | X |
| 9 | 1 | 1 | X | X |
| A | 0 | 1 | X | X |
| B | 1 | 1 | X | X |
| C | 1 | 0 | X | X |
| D | 1 | 0 | X | X |
| E | 0 | 0 | X | X |
| 1FFH | 0 | 0 | X | X |

X: Any data can be stored at any time because of division by 2 display.

**Fig. 5.7-18 Division by 2 LCD Panel Connection Example**

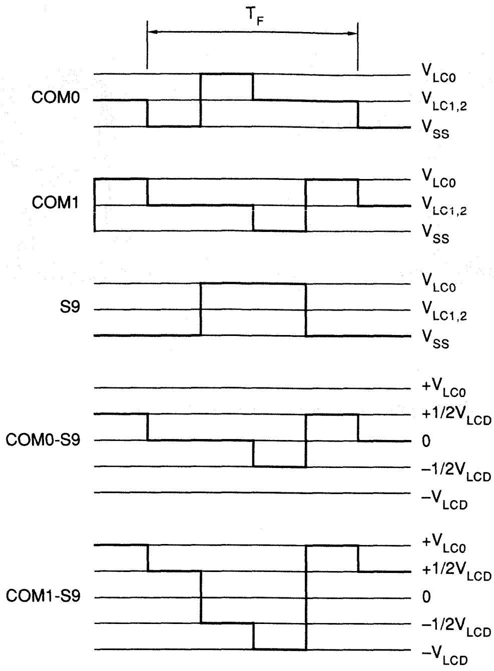


**Fig. 5.7-19 Division by 2 LCD Drive Waveform Example (1/2 Bias Law)**

(3) Division by 3 display example

Fig. 5.7-22 shows connection of a division by 3 mode 6-digit LCD panel having the display pattern shown in Fig. 5.7-21, the μPD753XX segment signals (S0-S29), and the common signals (COM0-COM2). In this example, 123456.7890 is displayed. The contents of the display data memory (addresses 1E0H-1FDH) correspond to the display pattern.

Here, 6. ( 6̲.) at the 5-digit position is taken as an example. It is necessary to output the selection and no-selection voltages as shown in Fig. 5.7-20 to the S12-S14 pins at the COM0-COM2 common signal timings according to the display pattern shown in Fig. 5.7-21.

| Segment / Common | S12 | S13 | S14 |
|---|---|---|---|
| COM0 | No-selection | Selection | Selection |
| COM1 | Selection | Selection | Selection |
| COM2 | Selection | Selection | – |

**Fig. 5.7-20 Selection/Non-selection Voltages of S12-S14 Pins (Division by 3 Mode Display Examples)**

This shows that the bits at display data memory address 1ECH corresponding to S12 need to be set to x110.

Fig. 5.7-23 and 5.7-24 show LCD drive waveforms among S12 and common signals (1/2 and 1/3 bias law). These shown an alternating current square wave of $+V_{LCD}/-V_{LCD}$ that is the selection voltage at the COM1 selection timing and S12 is the selection voltage at the COM2 selection timing.

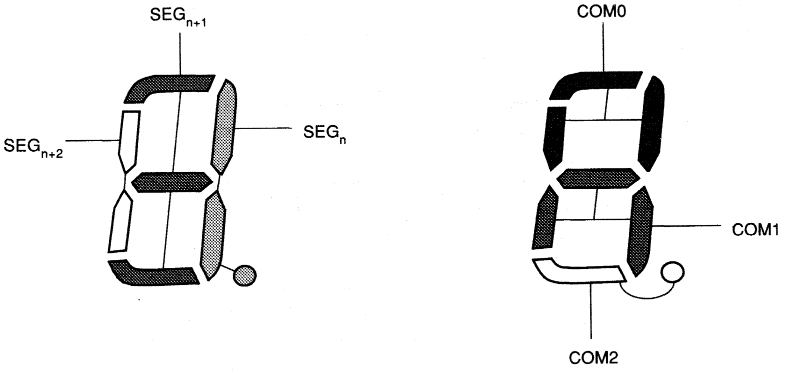
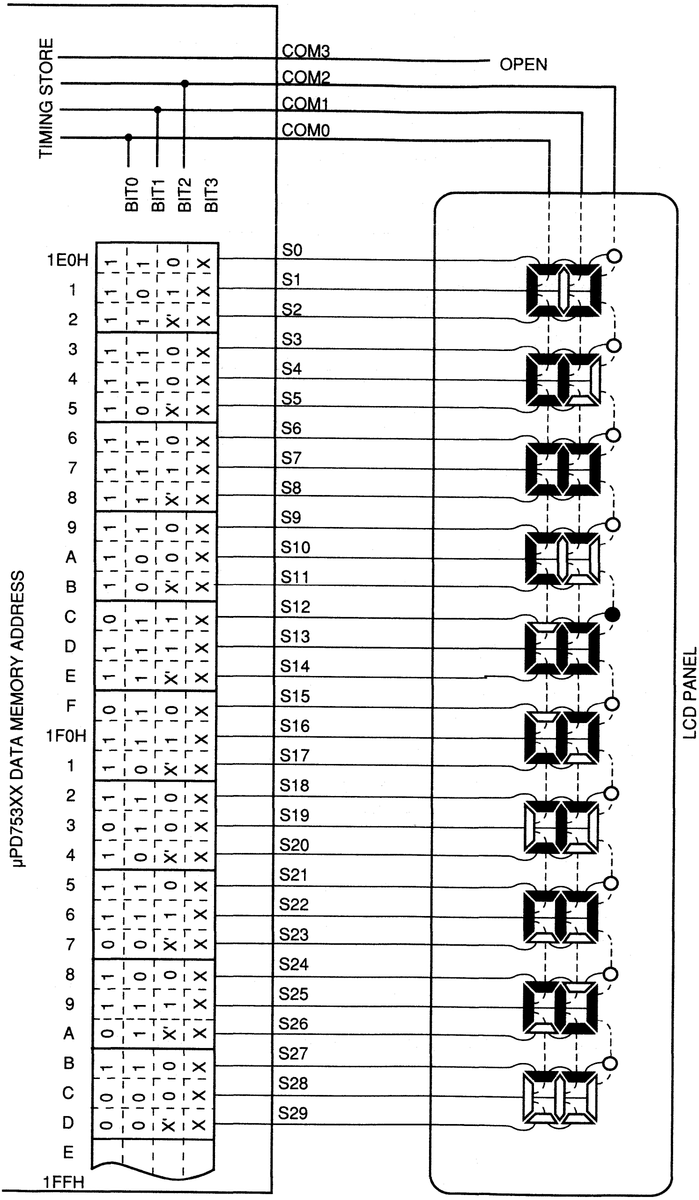


**Fig. 5.7-21 Division by 3 Mode LCD Display Pattern and Electrode Connection**



| μPD753XX DATA MEMORY ADDRESS | BIT0 | BIT1 | BIT2 | BIT3 |
|---|---|---|---|---|
| 1E0H | 1 | 1 | 0 | X |
| 1 | 1 | 0 | 1 | X |
| 2 | 1 | 1 | X' | X |
| 3 | 1 | 1 | 0 | X |
| 4 | 1 | 1 | 0 | X |
| 5 | 1 | 0 | X' | X |
| 6 | 1 | 1 | 0 | X |
| 7 | 1 | 1 | 1 | X |
| 8 | 1 | 1 | X' | X |
| 9 | 1 | 1 | 0 | X |
| A | 1 | 0 | 0 | X |
| B | 1 | 0 | X' | X |
| C | 0 | 1 | 1 | X |
| D | 1 | 1 | 1 | X |
| E | 1 | 1 | X' | X |
| F | 0 | 1 | 0 | X |
| 1F0H | 1 | 1 | 1 | X |
| 1 | 1 | 0 | X' | X |
| 2 | 1 | 1 | 0 | X |
| 3 | 0 | 1 | 0 | X |
| 4 | 1 | 0 | X' | X |
| 5 | 1 | 1 | 0 | X |
| 6 | 1 | 1 | 1 | X |
| 7 | 0 | 0 | X' | X |
| 8 | 1 | 0 | 0 | X |
| 9 | 1 | 1 | 1 | X |
| A | 0 | 1 | X' | X |
| B | 1 | 1 | 0 | X |
| C | 0 | 0 | 0 | X |
| D | 0 | 0 | X' | X |
| E |  |  |  |  |
| 1FFH |  |  |  |  |

X': Any data can be stored because the LCD panel does not have a corresponding segment.
X : Any data can be stored because of division by 3 display.

**Fig. 5.7-22 Division by 3 LCD Panel Connection Example**

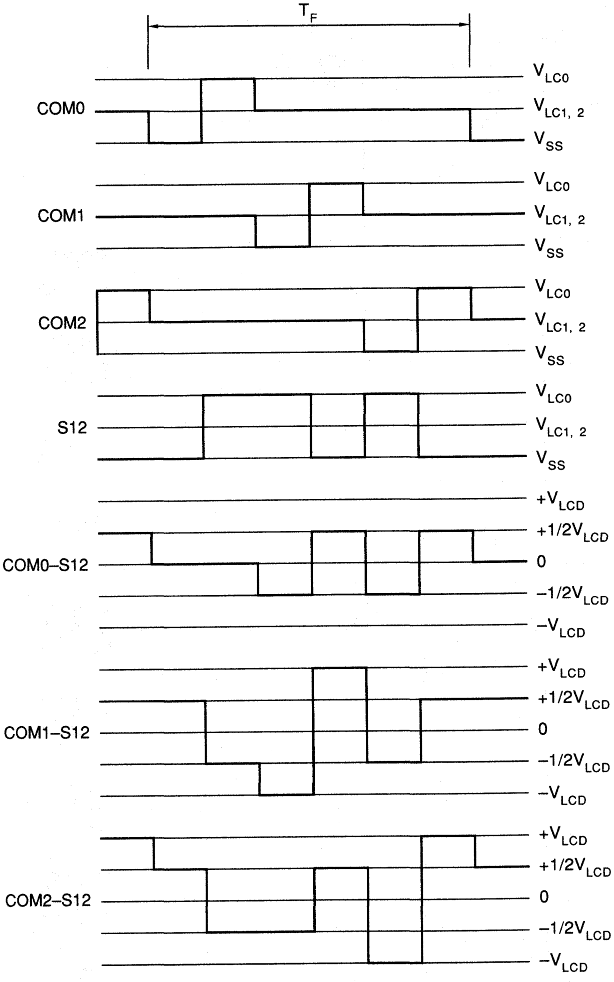


**Fig.5.7-23 Division by 3 LCD Drive Waveform Example (1/2 Bias Law)**

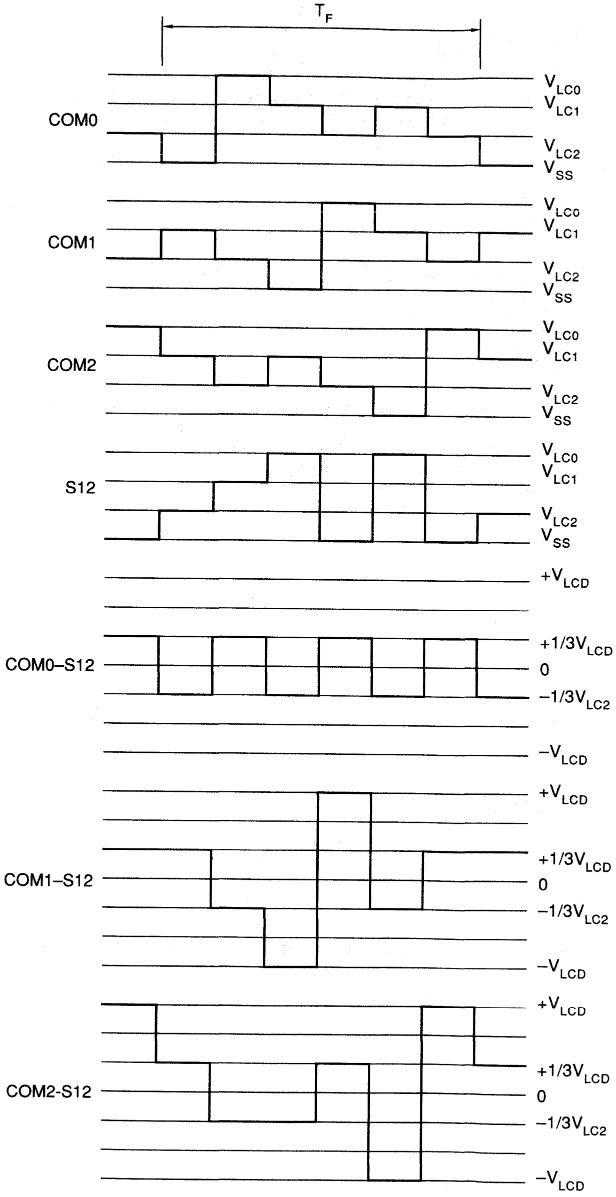


**Fig.5.7-23 Division by 3 LCD Drive Waveform Example (1/3 Bias Law)**

(4) Division by 4 Display Example

Fig. 5.7-27 shows connection of the division by 4 mode 16-digit LCD panel having the display pattern shown in Fig. 5.7-26, the μPD753XX segment signals (S0-S31), and the common signals (COM0-COM3). In this example, 123456.7890123456 is displayed. The contents of the display data memory (addresses 1E0H-1FFH) correspond to the display pattern.

Here, 6. (*6.*) at the 11-digit position is taken as an example. It is necessary to output the selection and no-selection voltages as shown in Fig. 5.7-25 to the S20 and S21 pins at the COM0-COM3 common signal timing according to the display pattern shown in Fig. 5.7-26.

| Segment / Common | S20 | S21 |
|---|---|---|
| COM0 | Selection | Selection |
| COM1 | No-selection | Selection |
| COM2 | Selection | Selection |
| COM3 | Selection | Selection |

**Fig. 5.7-25 Selection/Non-selection Voltage of S20 and S21 (Division by 4 Mode Display Examples)**

This shows that the bits at display data memory address 1F4H corresponding to S20 need to be set to 1101.

Fig. 5.7-28 shows LCD drive waveforms for S20, COM0, and COM1 signals. (Waveforms for COM2 and COM3 are omitted because of drawing space.) This shows an alternating current square wave of $+V_{LCD}/-V_{LCD}$ that is the LCD on level being generated when S20 becomes the selection voltage at the COM0 selection timing.

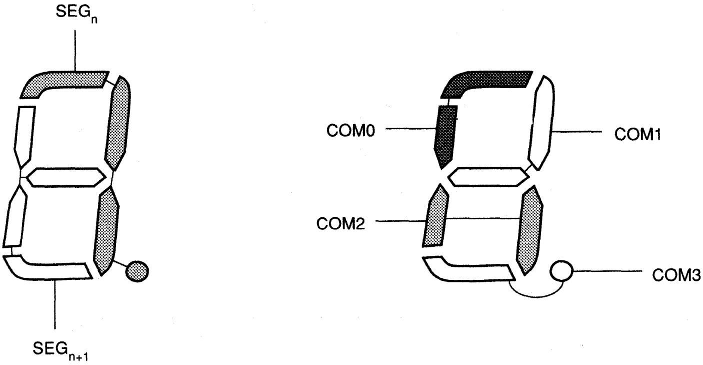
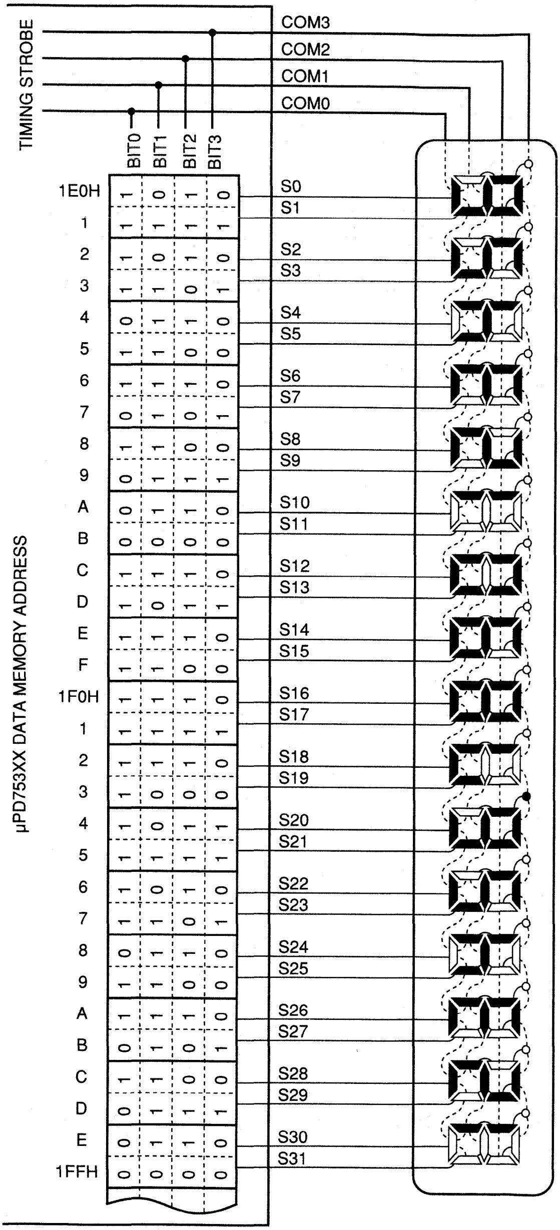


**Fig. 5.7-26 Division by 4 Mode LCD Display Pattern and Electrode Connection**



| | BIT0 | BIT1 | BIT2 | BIT3 |
|---|---|---|---|---|
| 1E0H | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 0 | 1 | 0 |
| 3 | 1 | 1 | 0 | 1 |
| 4 | 0 | 1 | 1 | 0 |
| 5 | 1 | 1 | 0 | 0 |
| 6 | 1 | 1 | 1 | 0 |
| 7 | 0 | 1 | 0 | 1 |
| 8 | 1 | 1 | 0 | 0 |
| 9 | 0 | 1 | 1 | 1 |
| A | 0 | 1 | 1 | 0 |
| B | 0 | 0 | 0 | 0 |
| C | 1 | 1 | 1 | 0 |
| D | 1 | 0 | 1 | 1 |
| E | 1 | 1 | 1 | 0 |
| F | 1 | 1 | 0 | 0 |
| 1F0H | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 1 | 0 |
| 3 | 1 | 0 | 0 | 0 |
| 4 | 1 | 0 | 1 | 1 |
| 5 | 1 | 1 | 1 | 1 |
| 6 | 1 | 0 | 1 | 0 |
| 7 | 1 | 1 | 0 | 1 |
| 8 | 0 | 1 | 1 | 0 |
| 9 | 1 | 1 | 0 | 0 |
| A | 1 | 1 | 1 | 0 |
| B | 0 | 1 | 0 | 1 |
| C | 1 | 1 | 0 | 0 |
| D | 0 | 1 | 1 | 1 |
| E | 0 | 1 | 1 | 0 |
| 1FFH | 0 | 0 | 0 | 0 |

**Fig. 5.7-27 Division by 4 LCD Panel Connection Example**

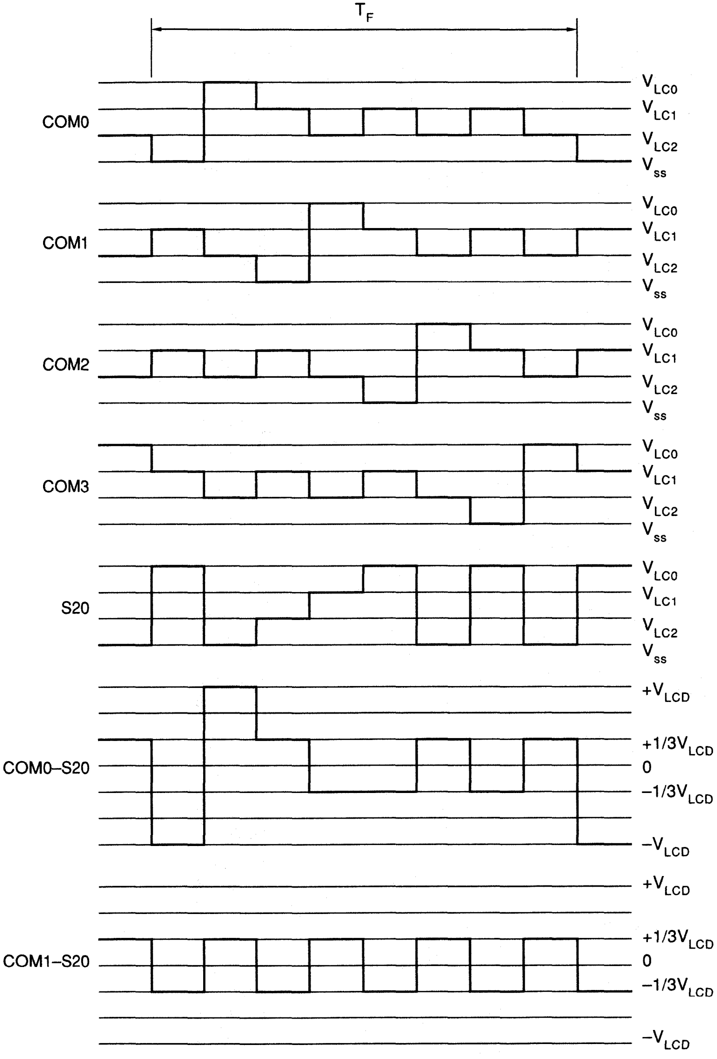


**Fig. 5.7-28 Division by 4 LCD Drive Waveform Example (1/3 Bias Method)**

### 5.8 Bit Sequential Buffer – 16 Bits

The bit sequential buffer is a special data memory for bit manipulation. Bit manipulation can be easily performed by changing address and bit specification in sequence. The buffer is useful for bitwise processing of long data.

The data memory consists of 16 bits. pmem.@L addressing of bit manipulation is enabled; indirect bit specification can be made by using the L register . Simply by incrementing or decrementing the L register in a program loop, processing can be performed while the specified bits are being moved in sequence.

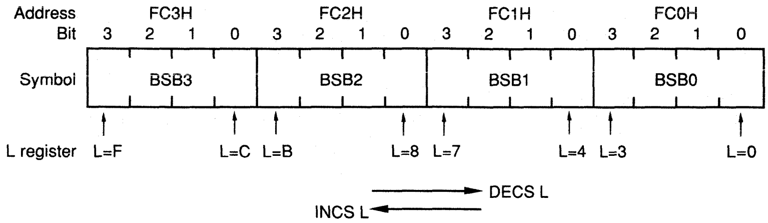

Address
FC3H
FC2H
FC1H
FC0H
Bit 3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0
Symbol
BSB3
BSB2
BSB1
BSB0
L register L=F L=C L=B L=8 L=7 L=4 L=3 L=0
DECS L
INCS L

**Fig. 5.8-1 Bit Sequential Buffer Format**

Remarks:

In pmem.@L addressing, specified bits are moved according to the L register.

Data can also be handled in direct addressing. The 1–, 4–, and 8-bit direct addressing modes and pmem.@L addressing mode can be used in combination for continuous input and output of 1-bit data. In 8-bit manipulation, BSB0 and BSB2 are specified, and data is handled every high-order eight and low-order eight bits.

Example:

To output 16-bit data in BUFF 1 and BUFF2 from port 3 bit 0 serially.

```
        CLR1   MBE
        MOV    XA, BUFF1
        MOV    BSB0, XA     ; BSB0 and BSB1 are set.
        MOV    XA, BUFF 2
        MOV    BSB2, XA     ; BSB2 and BSB3 are set.
        MOV    L, #0
LOOP:   SKT    BSB0, @L     ; BSB-specified bit is tested.
        BR     LOOP1
        NOP                 ; Dummy (timing adjustment)
        SET1   PORT3.0      ; Port 3 bit 0 is set.
        BR     LOOP2
LOOP1:  CLR1   PORT3.0      ; Port 3 bit 0 is cleared.
        NOP                 ; Dummy (timing adjustment)
        NOP
LOOP2:  INCS   L            ; L← L+1
        BR     LOOP
               RET
```

## CHAPTER 6 INTERRUPT FUNCTION

The μPD753XX contains six vectored interrupt sources and two testable inputs for versatile application.
The μPD753XX interrupt control circuit has the following features to enable very high-speed interrupt service:

(a) Whether or not interrupts can be acknowledged can be controlled by using enable flag (IEXXX).
(b) The interrupt service start address and MBE during interrupt service can be set as desired by using a vector table. Starting the actual interrupt service program is fast.
(c) Interrupt request flag (IRQXXX) can be tested and cleared. Interrupt occurrence can be checked by using software.
(d) The standby mode (STOP or HALT) can be released by making an interrupt request. A backup release source is available by using the interrupt enable flag.

### 6.1 Interrupt Control Circuit Configuration

Fig. 6.1-1 shows configuration of the interrupt control circuit. The hardware devices are mapped in data memory.

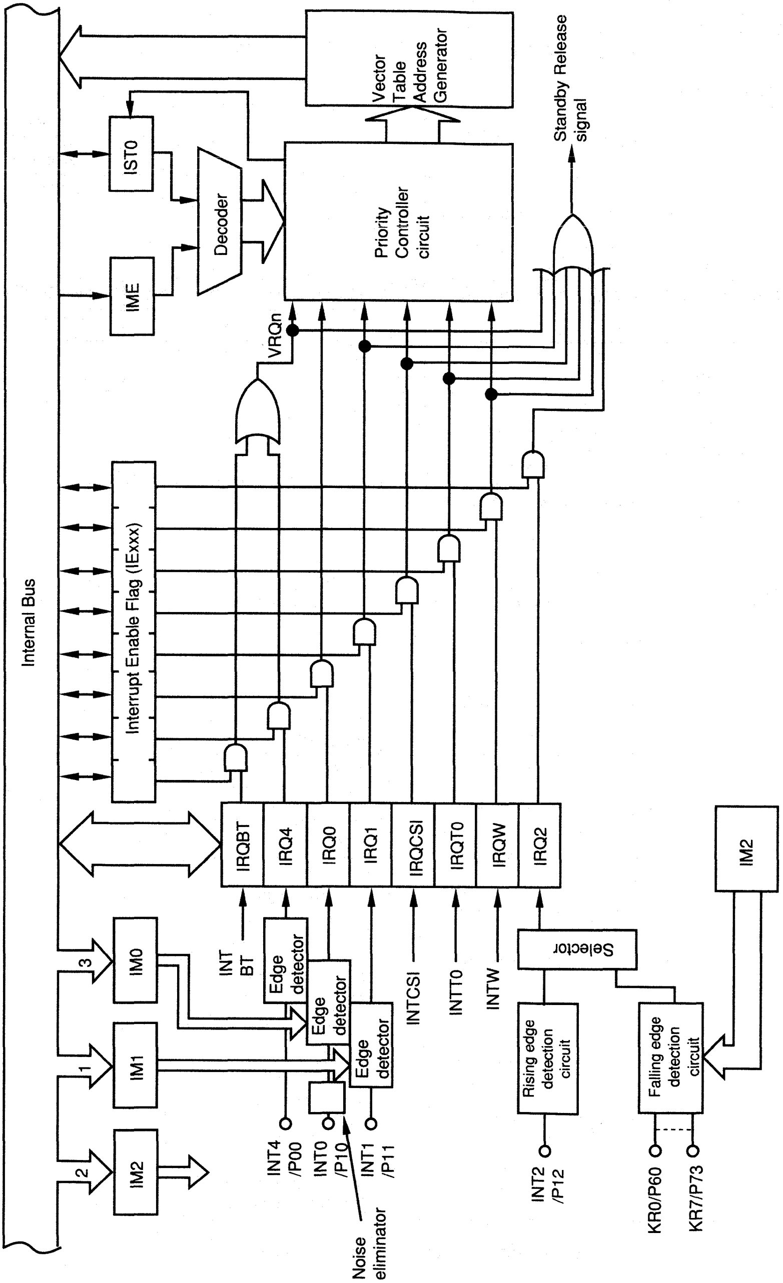


**Fig. 6.1-1 Interrupt Control Circuit Block Diagram**

## 6.2 Interrupt Source Types and Vector Table

Table 6.2-1 lists the types of interrupt sources for the μPD753XX. Fig. 6.2-1 shows the interrupt vector table.

**Table 6.2-1 Interrupt Source Types**

| Interrupt source | Internal /external | Interrupt priority (Note 1) | Vectored interrupt request signal (vector table address) |
|---|---|---|---|
| INTBT<br>(reference time interval signal from basic interval timer) | Internal | 1 | VRQ1 (0002H) |
| INT4<br>(both rising and falling edge detection) | External | | |
| INT0 (selection of rising or falling edge detection) | External | 2 | VRQ2 (0004H) |
| INT1 | External | 3 | VRQ3 (0006H) |
| INTCSI<br>(serial data transfer end signal) | Internal | 4 | VRQ4 (0008H) |
| INTT0<br>(coincidence signal between programmable timer/ counter count register and modulo register) | Internal | 5 | VRQ5 (000AH) |
| INT2<br>(rising edge detection of input to INT2 pin or falling edge detection of any input to KR0-KR7) (Note 2) | External | Testable input signals (IRQ2 and IRQW are set) | |
| INTW<br>(signal from watch timer) | Internal | | |

Note 1 : The interrupt priority indicates the priority given to each interrupt when more than one interrupt request occurs at the same time.

Note 2: For details of INT2, see 6.3 (2).

| Address | | | | | |
|---|---|---|---|---|---|
| 002H | MBE | 0 | 0 | 0 | INTBT/INT4 start address (high-order four bits) |
| | | | | | INTBT/INT4 start address (low-order eight bits) |
| 004H | MBE | 0 | 0 | 0 | INT0 start address (high-order four bits) |
| | | | | | INT0 start address (low-order eight bits) |
| 006H | MBE | 0 | 0 | 0 | INT1 start address (high-order four bits) |
| | | | | | INT1 start address (low-order eight bits) |
| 008H | MBE | 0 | 0 | 0 | INTCSI start address (high-order four bits) |
| | | | | | INTCSI start address (low-order eight bits) |
| 00AH | MBE | 0 | 0 | 0 | INTT0 start address (high-order four bits) |
| | | | | | INTT0 start address (low-order eight bits) |

**Fig. 6.2-1 Interrupt Vector Table (a) μPD75304**

| Address | | | | |
|---|---|---|---|---|
| 002H | MBE | 0 | 0 | INTBT/INT4 start address (high-order five bits) |
| | | | | INTBT/INT4 start address (low-order eight bits) |
| 004H | MBE | 0 | 0 | INT0 start address (high-order five bits) |
| | | | | INT0 start address (low-order eight bits) |
| 006H | MBE | 0 | 0 | INT1 start address (high-order five bits) |
| | | | | INT1 start address (low-order eight bits) |
| 008H | MBE | 0 | 0 | INTCSI start address (high-order five bits) |
| | | | | INTCSI start address (low-order eight bits) |
| 00AH | MBE | 0 | 0 | INTT0 start address (high-order five bits) |
| | | | | INTT0 start address (low-order eight bits) |

**Fig. 6.2-1 Interrupt Vector Table (b) μPD75306, μPD75308**

| Address | | | |
|---|---|---|---|
| 002H | MBE | 0 | INTBT/INT4 start address (high-order six bits) |
| | | | INTBT/INT4 start address (low-order eight bits) |
| 004H | MBE | 0 | INT0 start address (high-order six bits) |
| | | | INT0 start address (low-order eight bits) |
| 006H | MBE | 0 | INT1 start address (high-order six bits) |
| | | | INT1 start address (low-order eight bits) |
| 008H | MBE | 0 | INTCSI start address (high-order six bits) |
| | | | INTCSI start address (low-order eight bits) |
| 00AH | MBE | 0 | INTT0 start address (high-order six bits) |
| | | | INTT0 start address (low-order eight bits) |

**Fig. 6.2-1 Interrupt Vector Table (c) μPD75312, μPD75316**

The interrupt priority in Table 6.2-1 indicates the interrupt execution order when a number of interrupt requests occur at the same time or are held.

The interrupt service start addresses and MBE setup values during interrupt service are written into the vector table. The vector table is set by using an assembler pseudo-instruction (VENTn).

Example: To set INTBT/INT4 in vector table

```
 VENT1          MBE=0, RBE=0,          GOTOBT
   ↑              ↑      ↑               ↑
   ①              ②      ③               ④
```

① Vector table address 0002.

② MBE setup value in interrupt service routine.

③ Always set RBE to 0.

④ This symbol denotes the interrupt service routine start address.

Caution 1: The vector table address specified in VENTn (n=1-5) becomes 2n address.
Caution 2: Always set RBE to 0 in VENTn.

Example: To set INTBT/INT4 and INTT0 in vector table.

```
VENT1          MBE=0, RBE=0, GOTOBT
VENT5          MBE=0, RBE=0, GOTOT0
```

**6.3 Hardware Devices of Interrupt Control Circuit**

(1) Interrupt request and enable flags

The interrupt request flag (IRQXXX) is set to 1 when a given interrupt request occurs; it is automatically cleared when interrupt service is executed.

Although IRQBT and INT4 share the vector address, they differ in clear operation. (See 6.6.)

An interrupt enable flag (IEXXX) is provided for each interrupt request flag. An interrupt is enabled when the corresponding interrupt enable flag is set to 1 and disabled when the flag is set to 0.

When the interrupt request flag is set, and the interrupt enable flag enables an interrupt, a vectored interrupt request (VRQn) occurs. This signal is also used to release the standby mode.

The interrupt request and enable flags are handled by using the bit manipulation and 4-bit memory operation instructions. The flags can be directly handled by using the bit manipulation instruction regardless of how MBE is set. The interrupt enable flags are handled by using the EI IEXXX and DI IEXXX instructions. Normally, the interrupt request flags are tested by using the SKTCLR instruction.

```
Example:   EI         IE0       ; INT0 is enabled.
           DI         IE1       ; INT1 is disabled.
           SKTCLR     IRQCSI    ; If IRQCSI is set to 1, skip and clear.
```

If the interrupt request flag is set by using the instruction, a vectored interrupt is executed as if an interrupt occured although it did not actually occurs.

When the RESET signal is generated, the interrupt request and enable flags are cleared, and all interrupts are disabled.

**Table 6.3-1 Interrupt Request and Enable Flags**

| Interrupt request flag | Interrupt request flag set signal | Interrupt enable flag |
|---|---|---|
| IRQBT | Is set by reference time interval signal generated by the basic interval timer | IEBT |
| IRQ4 | Is set when either the rising or falling edge of the INT4/P00 pin input signal is detected. | IE4 |
| IRQ0 | Is set when the INT0/P10 pin input signal edge is detected. Detected edge is selected among rising and falling edges by using the INT0 mode register (IM0). | IE0 |
| IRQ1 | Is set when the edge of the INT1/P11 pin input signal is detected. Detected edge is selected from rising and falling edges by using the INT1 mode register (IM1). | IE1 |
| IRQCSI | Is set by serial data transfer end signal on serial interface. | IECSI |
| IRQT0 | Is set by coincidence signal from timer/event counter 0. | IET0 |

**Table 6.3-1 Interrupt Request and Enable Flags (cont'd)**

| Interrupt request flag | Interrupt request flag set signal | Interrupt enable flag |
|---|---|---|
| IRQW | Is set by signal from the watch timer. | IEW |
| IRQ2 | Is set when the rising edge of the INT2/P12 pin input signal is detected or the falling edge of any input to the KR0/P60 to KR7/P73 pins is detected. | IE2 |

(2) Noise eliminator and edge detection mode registers.

Figs. 6.3-1 and 6.3-2 show the relationship of INT0 and INT1, INT2 and KR0-KR7 pins, respectivvely, within the interrupt control circuit. They are used for inputting external interrupts in cases where noise can be eliminated by the sampling clock and the detected edge can be selected.

Any pulse narrower than the sampling clock is determined to be a noise pulse and is eliminated by the noise eliminator. A pulse twice as wide (or more) as the sampling clock is acknowledged as a valid interrupt signal.

For INT0 the sampling clock can be changed to two stages.

Note that when the INT0 pin is used for P10 (port pin), the signal is also input via the noise eliminator.

Caution: Since INT0 sampling uses clock, INT0 does not operate at standby mode.

IRQ2 is set in either of the following modes:

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

(b) When the falling edge of any input to KR0-KR7 pins is detected:

When the falling edge of any input to pins selected among KR0-KR7 by using the edge detection mode register (IM2) is detected, IRQ2 is set.

For example, if KR4-KR7 are selected, IRQ2 is set if the falling edge is input to any one of KR4-KR7, because KR4-KR7 inputs are ANDed in the μPD753XX.

Fig. 6.3-4 shows the format of the edge detection mode registers used to select detected edge (IM0 to IM2). IM0 to IM2 are set individually by using a 4-bit memory operation instruction.

When RESET signal is generated, all the register bits are cleared; INT0, INT1, and INT2 detection edges are specified to rising edges.

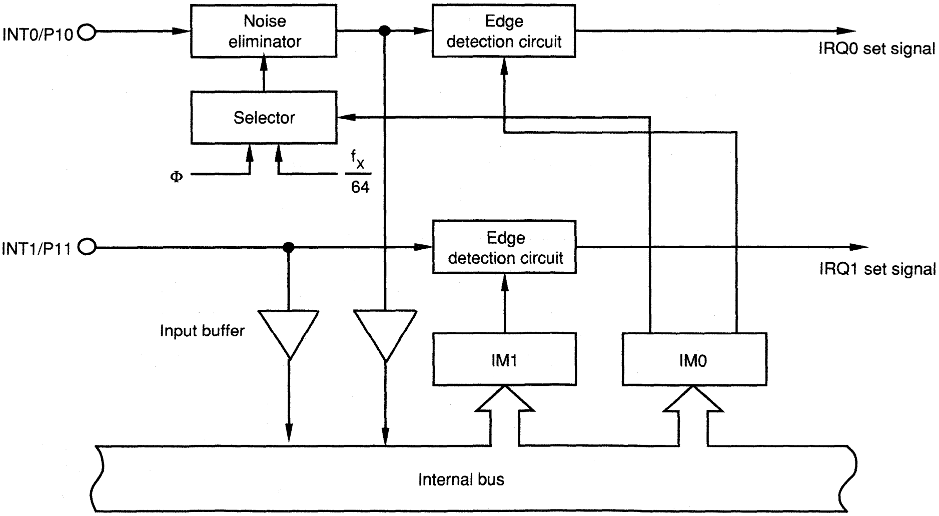


**Fig. 6.3-1 INT0 and INT1 Configuration**

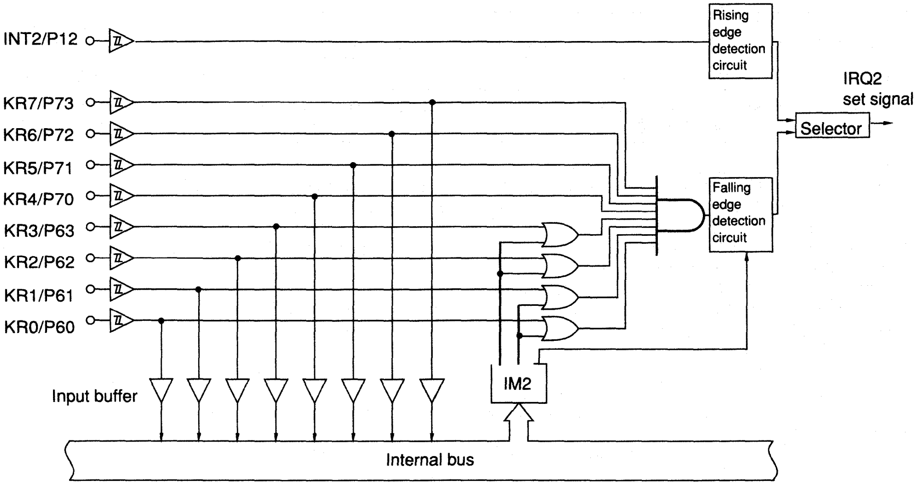


**Fig. 6.3-2 INT2 and KR0-KR7 Configuration**

When a high level is input successively, the noise eliminator outputs a high level; when a low level is input successively, it outputs a low level.

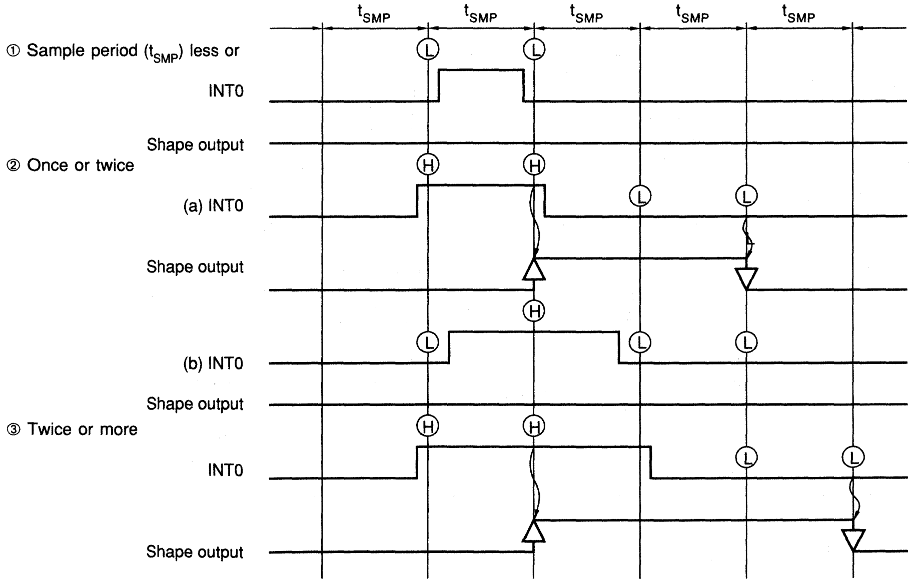
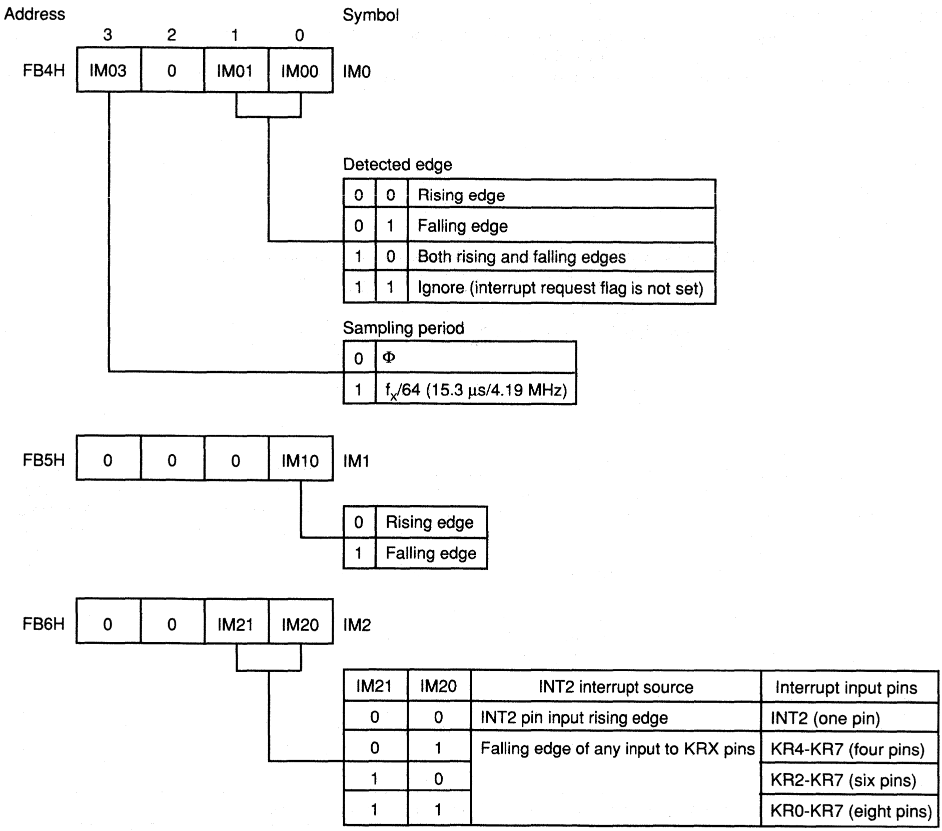


**Fig. 6.3-3 Noise Eliminator Input/Output Timing**



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

Detected edge

| IM01 | IM00 | |
|---|---|---|
| 0 | 0 | Rising edge |
| 0 | 1 | Falling edge |
| 1 | 0 | Both rising and falling edges |
| 1 | 1 | Ignore (interrupt request flag is not set) |

Sampling period

| IM03 | |
|---|---|
| 0 | Φ |
| 1 | $f_X$/64 (15.3 μs/4.19 MHz) |

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| FB5H | 0 | 0 | 0 | IM10 | IM1 |

| IM10 | |
|---|---|
| 0 | Rising edge |
| 1 | Falling edge |

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| FB6H | 0 | 0 | IM21 | IM20 | IM2 |

| IM21 | IM20 | INT2 interrupt source | Interrupt input pins |
|---|---|---|---|
| 0 | 0 | INT2 pin input rising edge | INT2 (one pin) |
| 0 | 1 | Falling edge of any input to KRX pins | KR4-KR7 (four pins) |
| 1 | 0 | | KR2-KR7 (six pins) |
| 1 | 1 | | KR0-KR7 (eight pins) |

Caution: If the edge detection mode register is changed, the interrupt request flag may be set. Disable interrupts beforehand, and change the mode register. Clear the interrupt request flag by using the CLR1 instruction before enabling interrupts. If $f_X$/64 is selected for the sampling clock in an IM0 change, clear the interrupt request flag within 16 machine cycles after the mode register is changed.

**Fig. 6.3-4 Edge Detection Mode Register Format**

(3) Interrupt master enable flag (IME)

Interrupt master enable flag enables or disables acknowledgement of all interrupts.

IME is set to 1 or to 0 by using the EI and DI instructions.

When the $\overline{\text{RESET}}$ signal is generated, IME is cleared, disabling acknowledgement of all interrupts.

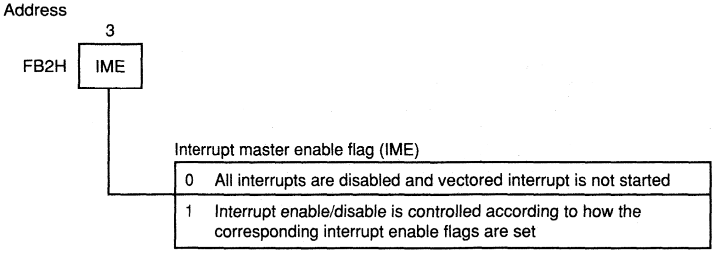

Address
3
FB2H
IME
Interrupt master enable flag (IME)
0 All interrupts are disabled and vectored interrupt is not started
1 Interrupt enable/disable is controlled according to how the corresponding interrupt enable flags are set

**Fig. 6.3-5 IME Format**

(4) Interrupt status flag

The interrupt status flag (IST0) indicates the current status of processing being performed by CPU; it is contained in the PSW. The interrupt control circuit controls multi-interrupts (as listed in Table 6.3-2) according to the contents of interrupt status flag. Since IST0 can be changed by using a 4-bit or a one-bit manipulation instruction, multi-interrupt can also be implemented by changing the status of processing being performed. Multi-interrupt can always be implemented regardless of the MBE setting when IST0 is handled bitwise.

Be sure to execute the DI instruction to disable interrupts before handling IST0 and the EI instruction to enable interrupts after handling IST0.

When an interrupt is acknowledged, IST0 is saved in stack memory together with other PSW bits, then it is automatically set to 1. When the RETI instruction is executed, the former IST0 value (0) is restored.

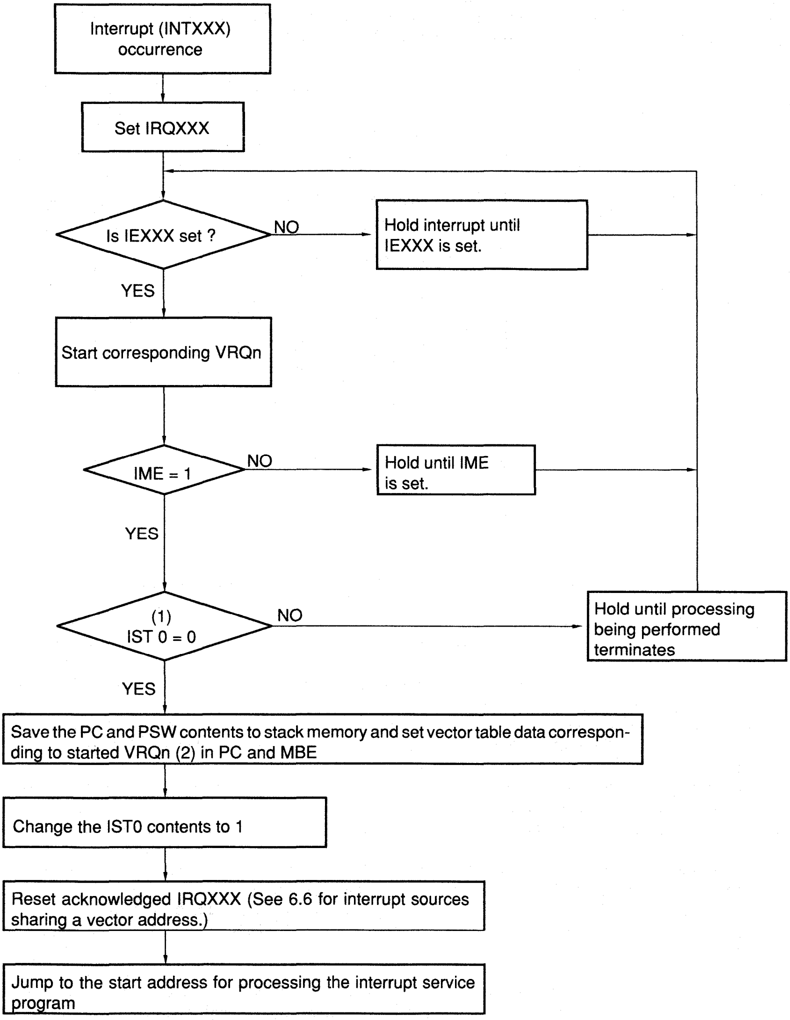
When the $\overline{\text{RESET}}$ signal is generated, the flag is cleared.

**Table 6.3-2 IST0 and Interrupt Service State**

| IST0 | Status of processing | CPU processing contents | Interrupt requests that can be acknowledged | After interrupt is acknowledged |
|---|---|---|---|---|
| | | | | IST0 |
| 0 | Status 0 | During normal program processing | All interrupts can be acknowledged. | 1 |
| 1 | Status 1 | During interrupt service | None of the interrupts can be acknowledged. | – |

### 6.4 Interrupt Sequence

When an interrupt occurs, it is processed as shown in Fig. 6.4-1.


Interrupt (INTXXX) occurrence
Set IRQXXX
Is IEXXX set ?
NO
Hold interrupt until IEXXX is set.
YES
Start corresponding VRQn
IME = 1
NO
Hold until IME is set.
YES
(1) IST 0 = 0
NO
Hold until processing being performed terminates
YES
Save the PC and PSW contents to stack memory and set vector table data correspon- ding to started VRQn (2) in PC and MBE
Change the IST0 contents to 1
Reset acknowledged IRQXXX (See 6.6 for interrupt sources sharing a vector address.)
Jump to the start address for processing the interrupt service program

Remarks: 1. IST0: Interrupt status flag (PSW bit 2). (See Table 6.3-2)

2. Store the interrupt service program start addresses and MBE setup values at the interrupt start in the vector table.

**Fig. 6.4-1 Interrupt Service Flow**

### 6.5 Multi-interrupt Service Control

The μPD753XX enables multi-interrupts as described below.

As understood from Table 6.3-2, multi-interrupt is enabled if the interrupts status flag is changed by a program. That is, multi-interrupt is enabled if IST0 is changed to 0 by the interrupt service program setting status 0.

To change IST0, execute the DI instruction beforehand to disable interrupts.

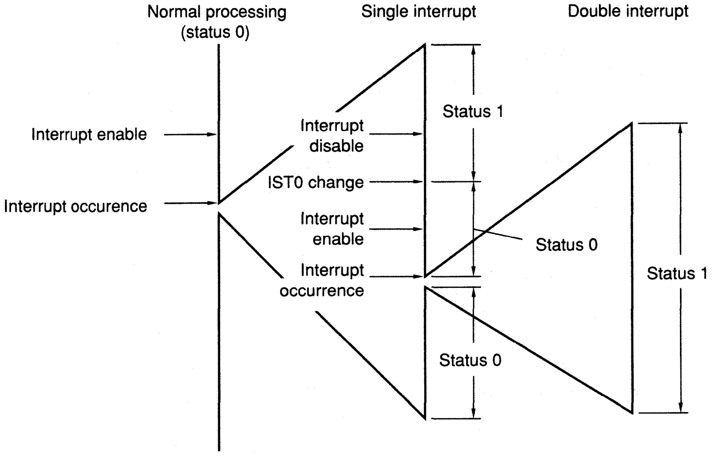


**Fig. 6.5-1 Multi-Interrupt by Changing Interrupt Status Flag**

### 6.6 Vector Address Share Interrupt Service

Since INTBT and INT4 interrupt sources share the vector table addresses, interrupt source selection is made as described below:

(1) To use one interrupt only

Set the interrupt enable flag to 1 for the required one of the two interrupt sources sharing the vector table addresses, and clear the interrupt enable flag of the other interrupt source.

In this case, an interrupt request is generated from the interrupt source corresponding to the interrupt enable flag that is set to 1 (IEXXX = 1). When the interrupt request is acknowledged, the interrupt request flag is cleared.

(2) To use both interrupts

Set both the interrupt enable flags of the two interrupt sources to 1. In this case, an interrupt request is made by ORing the interrupt request flags of the two interrupt sources.

Even if an interrupt request is acknowledged when either or both of the interrupt request flags are set to 1, the interrupt request flags are not reset.

Therefore, the interrupt service routine must decide which interrupt source the interrupt is generated from. This is accomplished by executing the SKTCLR instruction to check the interrupt request flags.

If both the interrupt request flags are set to 1 when the request flags are tested and cleared by execution of the SKTCLR instruction, the interrupt request is left even if one request flag is cleared. If IST0 is cleared, dual interrupt service is entered according to the left interrupt request.

Remarks: When only one interrupt is enabled, the source of an interrupt is known. Thus, the interrupt request flag is cleared by hardware when interrupt is acknowledged. When both interrupts are enabled, the source of an interrupt is not defined; thus, interrupt request flag cannot be cleared by hardware. Software is used to check the interrupt request flags and determine the interrupt source. Interrupt request flag is cleared by software.

Example: When INT4 takes precedence over INTBT

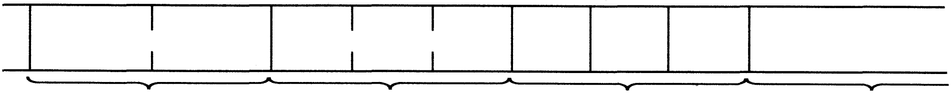
```
          DI
          SKTCLR    IRQ4      ; IRQ4 = 1 ?
          BR        VSUBBT   ⎫
           :                 ⎬ INT4 interrupt
          EI                 ⎪ service routine
          RETI               ⎭
           :
           :
VSUBBT:   CLR1      IRQBT    ⎫
           :                 ⎪
           :                 ⎬ INTBT interrupt
           :                 ⎪ service routine
          EI                 ⎭
          RETI
```

### 6.7 Machine Cycles to Interrupt Service Start

The number of the μPD753XX machine cycles required to start execution of the interrupt routine after an interrupt request flag is set is as shown below:

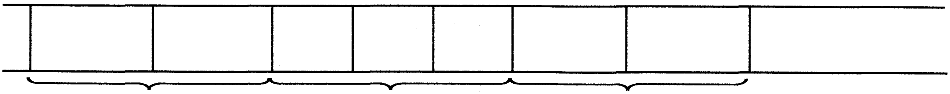
(1) When IRQn is set during execution of an operating instruction at data memory address FBxH (interrupt hardware))

FBxH address operating instruction


IRQn is set during this interval (depending on the number of machine cycles for the instruction)
One to three machine cycles (depending on the number of machine cycles for the instruction)
Three machine cycles (required for interrupt service)
Interrupt routine execution

As shown above, interrupt routine processing is started a maximum of six machine cycles after the data memory (address FBxH) operating instruction is terminated. (Within a maximum of six machine cycles after the last operating instruction is terminated if the FBxH address operating instructions are successive.)

(2) When IRQn is set during execution of instruction other than in (1).


IRQn is set during this interval.
One to three machine cycles (depending on the number of machine cycles for the instruction)
Three machine cycles (required for interrupt service)
Interrupt routine execution

In this case, a maximum of six machine cycles are required.

### 6.8 Effective Use of Interrupts

Use the interrupt function as described below:

(1) Set MBE = 0 in the interrupt service routine.

If the data memory area used in the interrupt service routine is preferentially allocated to addresses 0-7FH and MBE = 0 is set in the interrupt vector table, a program can be prepared without considering memory banks.

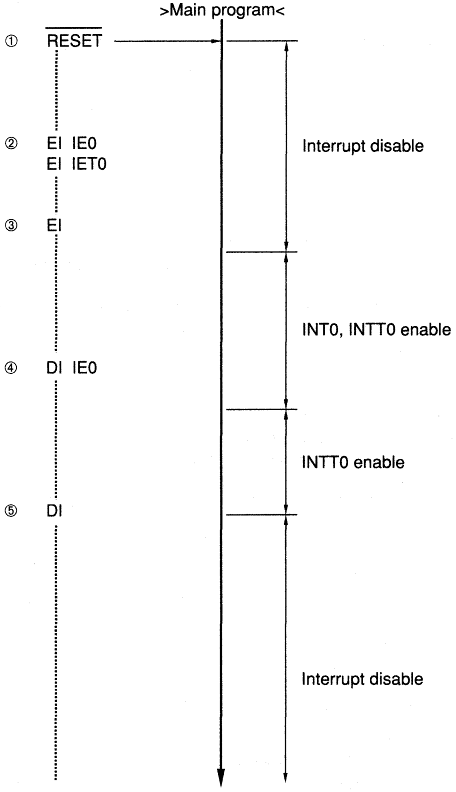
If memory bank 1 must be used for by the program, save the memory bank select register by using the PUSH BS instruction and select memory bank 1.

(2) Use software interrupt for debugging.
When an interrupt request flag is set by an instruction, operating is performed in the same manner as if an interrupt occured. Debugging when more than one interrupt occurred at the same time can be done effeciently by using an instruction to set an interrupt request flag.

### 6.9 Interrupt Application

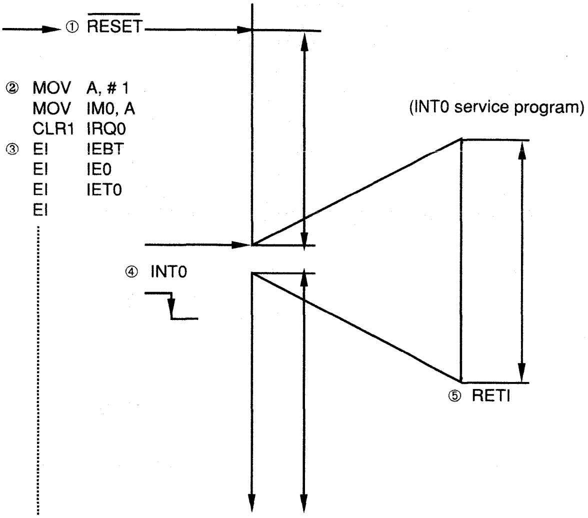
To use the interrupt function, first set the following in the main program:
(1) Set the interrupt enable flags to be used (EI IEXXX instructions).
(2) To use INT0 and INT1, select the active edge. (set IM0 and IM1.)
(3) Set the interrupt master enable flag (EI instruction).
Since MBE is set by using the vector table in an interrupt service program, registers need not be saved or restored, and the interrupt program can be started immediately.
To return from the interrupt service program, use the RETI instruction.

(1) Interrupt enable and disable



① All interrupts are disabled by the $\overline{\text{RESET}}$ signal.
② The interrupt enable flags are set by the EI IEXXX instructions. At this stage, all interrupts remain disabled.
③ The interrupt master enable flag is set by the EI instruction. At this stage, INT0 and INTT0 are enabled.
④ The interrupt enable flag is cleared by the DI IEXXX instruction and INT0 is disabled.
⑤ All interrupts are disabled by the DI instruction.

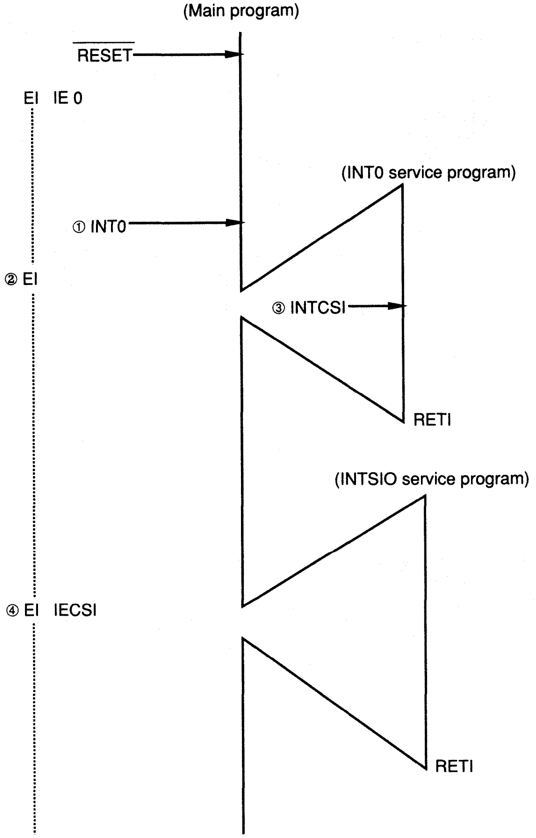
(2) Usage example of INTBT, INT0 (falling edge active), and INTT0. Multi-interrupt is not done.



```
② MOV  A, # 1
  MOV  IM0, A
  CLR1 IRQ0
③ EI   IEBT
  EI   IE0
  EI   IET0
  EI
```

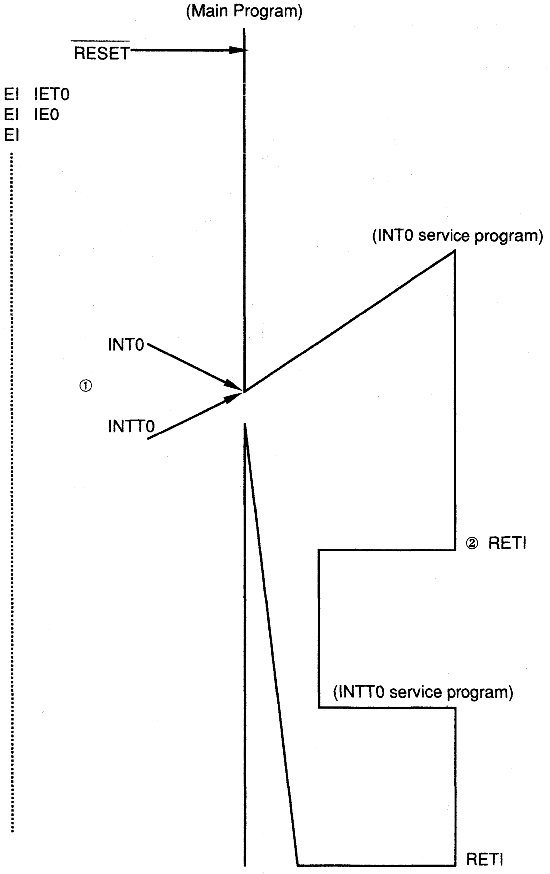
① All interrupts are disabled by the $\overline{\text{RESET}}$ signal and status 0 is set.
② Falling edge active is selected for INT0.
③ Interrupts are enabled by the EI and EI IEXXX instructions.
④ On the INT0 falling edge, the INT0 interrupt service program is started. The status is changed to status 1 and all interrupts are disabled.
⑤ A return is made from the interrupt service program by the RETI instruction. The status is restored to status 0 and interrupts are enabled.

(3) Pending interrupt execution – interrupt input during interrupt disable –



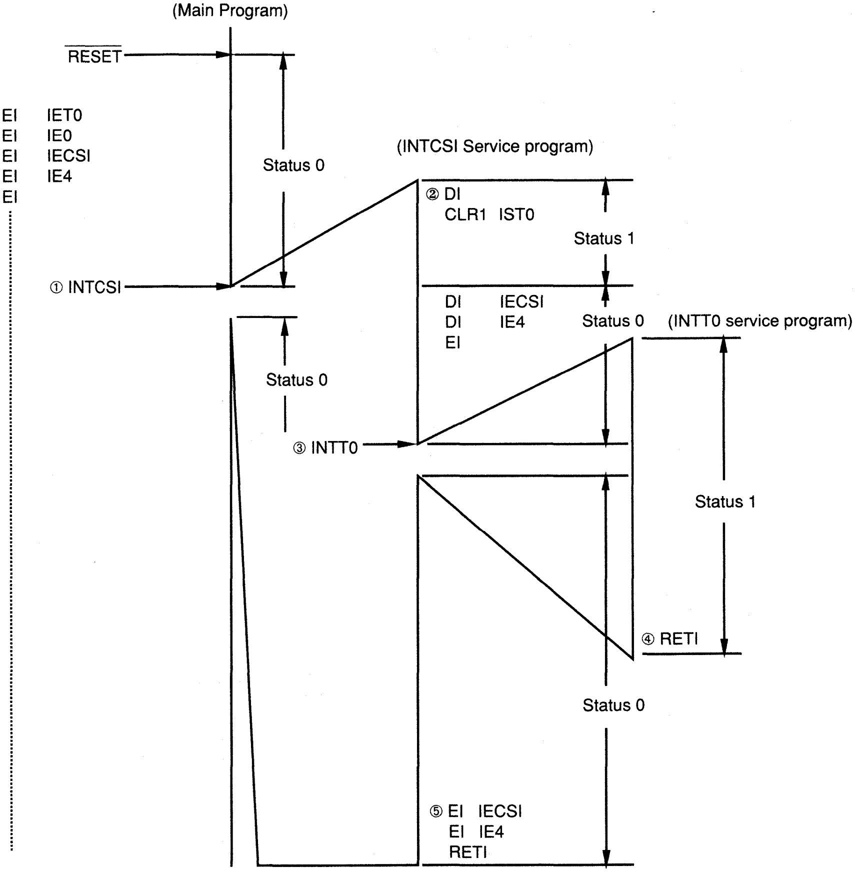
① If INT0 is set during interrupt disable, the request flag is held.
② When interrupts are enabled by the EI instruction, the INT0 service program is started.
③ Simular to ① above.
④ When INTCSI is held enabled, the INTCSI service program is started.

(4) Pending interrupt execution



① If INT0 and INTT0 are occured at the same time during execution of a single instruction, INT0, which is higher in priority than INTT0, is executed first; INTT0 is held.
② When a return is made from the INT0 service program by the RETI instruction, the held INTT0 service program is started.

(5) Enable of two double interrupts – INTT0 and INT0 enable double interrupts; INTCSI and INT4 are single interrupts –



① When interruptINTCSI occurs, which does not enable double interrupts, the INTCSI service program is started. Status 1 is set.
② The status is changed to status 0 by clearing IST0. INTCSI and INT4 do not enable double interrupt and are disabled.
③ When INTT0 enables double interrupts, the double interrupt is executed, the status is changed to status 1, and all interrupts are disabled.
④ At the termination of INTT0 service, the status is restored to status 0.
⑤ Disabled INTCSI and INT4 are enabled and a return is made by the RETI instruction.

## 7. STANDBY FUNCTION

To make the most of low current consumption, which is one feature of the CMOS process, the μPD753XX can stop CPU operation in the standby mode making current consumption by the CPU very small.
The μPD753XX standby mode includes the STOP mode and HALT mode.
The STOP mode stops the main system clock oscillator. In this mode, CPU current consumption consists almost entirely of leakage current. Data memory can also be held with low supply voltage (up to $V_{DD}$ = 2 V). This features is useful for maintaining the data memory contents with very low current consumption. Since the μPD753XX STOP mode can be released by using an interrupt request, intermittent operation can also be performed. However, if processing must be started immediately when an interrupt request is made, note that the wait time required to ensure oscillator stability is taken when the STOP mode is released.
The HALT mode continues system clock oscillator operation but stops the CPU clock (Φ) supply; thus, CPU operation is stopped. Although the HALT mode is inferior to the STOP mode for reduction of current consumption, it is useful to restart processing immediately according to an interrupt request or for performing intermittent operations such as watch operation.
In either mode, all the register, flag, and data memory contents immediately before the standby mode is entered are held. The input/output port output latch state and output buffer state are also held. The input/output port state is handled beforehand so that the current consumption of the entire system is minimized.

Cautions on Use of Standby Mode:

1. The STOP mode can be used only when μPD753XX operation uses the main system clock. (Subsystem clock oscillation cannot be stopped.) The HALT mode can be used when the μPD753XX uses either main system or subsystem clock.
2. If the STOP mode is set when the LCD controller/driver and watch clock operate on the main system clock $f_x$, they stop operation. To continue operation, change to the subsystem clock $f_{XT}$ before setting STOP mode.
3. Although efficient operation with low current consumption and low voltage can be performed by using the clock change function between the CPU and system clocks in combination with the standby mode, time (described in 5.2.3) is required from selection of a new clock by setting the control register until operation is started by the newly selected clock.
   Thus, to use the clock change function and the standby mode in combination, set the standby mode within the time required for the clock change.

### 7.1 Standby Mode Setting and Operating State

**Table 7.1-1 Operating State in Standby Mode**

| | | STOP mode | HALT mode |
|---|---|---|---|
| Setting instruction | | STOP instruction | HALT instruction |
| System clock when standby mode is set. | | Can be set only during main system clock. | Can be set during either main system or subsystem clock. |
| Operating state | Clock oscillator | Only the main system clock oscillator is stopped. | Only CPU clock Φ is stopped (oscillation is continued). |
| | Basic interval timer | Operation stop | Operation (IRQBT is set at reference time intervals.) |
| | Serial interface | Can operate only when external $\overline{SCK}$ input is selected for serial clock. | Can operate. |
| | Timer/event counter | Can operate only when TI0 pin input is selected for count clock. | Can operate. |
| | Watch timer | Can operate when $f_{XT}$ is selected for count clock. | Can operate. |
| | LCD controller | Can operate only when $f_{XT}$ is selected for LCDCL. | Can operate. |
| | External interrupts | INT1, INT2, and INT4 can operate. Only INT0 cannot operate. | |
| | CPU | Operation stop | |
| Release signal | | Interrupt request signal enabled with interrupt enable flag from operating hardware (except for INT0), or $\overline{RESET}$ input. | |

The STOP mode is set by using the STOP instruction to set PCC bit 3; the HALT mode is set by the using HALT instruction to set PCC bit 2.

To change the CPU clock by using the low-order two bits of PCC, a time lag may occur from PCC rewrite to CPU clock change. Thus, to change the clock before the standby mode is entered or after the standby mode is released, set the standby mode within the number of machine cycles required to change the CPU clock after PCC is rewritten.

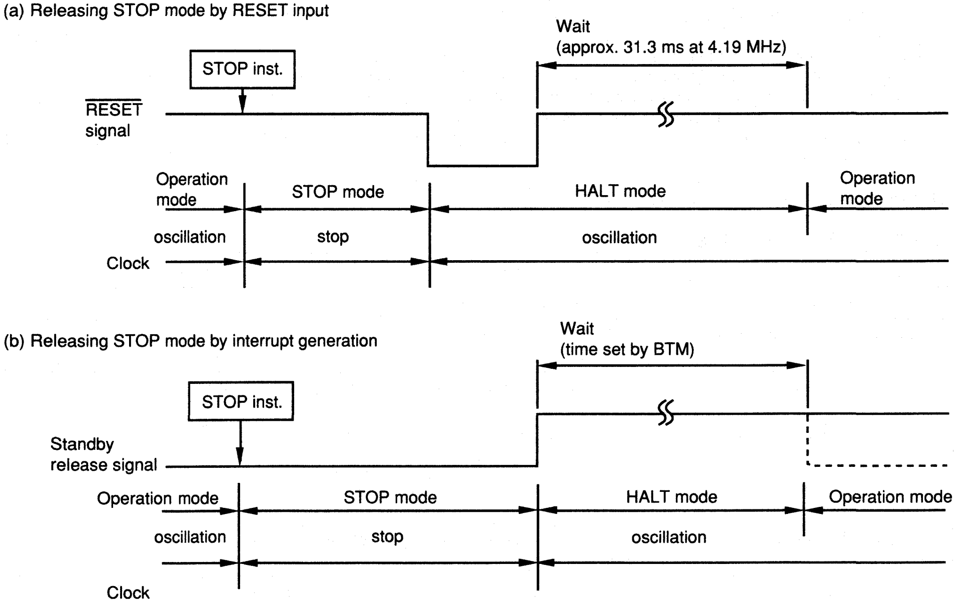
While operation stops during the standby mode, data is held in all registers and data memory such as general-purpose registers, flags, mode register, and output latches.

Caution: 1. When the STOP mode is set, X1 input is short-circuited to $V_{SS}$ (GND potential) internally to suppress crystal oscillator leakage. Therefore, so not use the STOP mode in a system using external clock as a main system slock.

2. Since an interrupt request signal is used to release the standby mode, if both interrupt request and enable flags are set for an interrupt source, the standby mode is immediately released. Thus, for the STOP mode, the HALT mode is entered immediately after execution of the STOP instruction, a wait is followed according to for the setup time of the BTM register, then a return is made to the opertion mode.

### 7.2 Realising Standby Mode

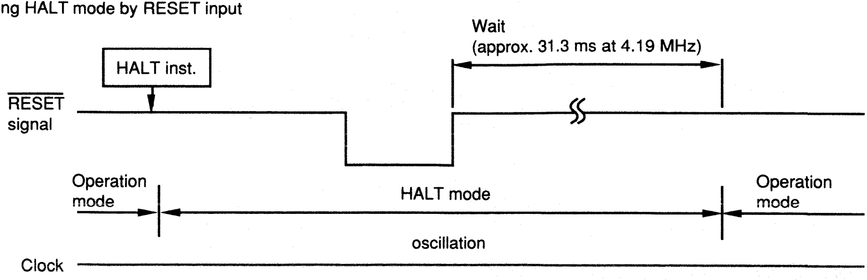
The standby mode (STOP or HALT) is released when an interrupt request signal (except INT0) enabled with an interrupt enable flag occurs or RESET is input. Fig. 6.2-1 shows the standby mode release operation.



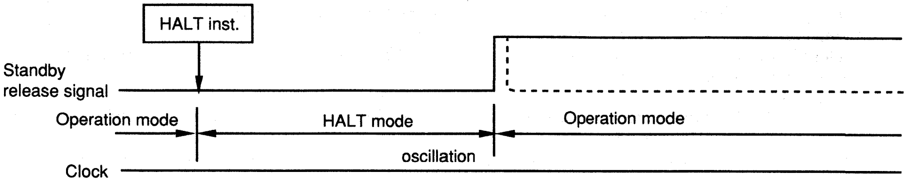
Note: Broken line shows a case when standby releasing interrupt request is acknowledged (IME = 1).

**Fig. 7.2-1 Standby Mode Release Operation**

(c) Releasing HALT mode by $\overline{\text{RESET}}$ input



(d) Releasing HALT node by interrupt generation



Note: Broken line shows a case when standby releasing interrupt request is acknowledged (IME = 1).

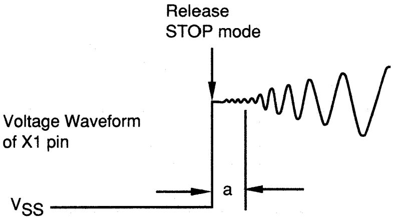
**Fig. 7.2-1 Standby Mode Release Operation (cont'd)**

If the STOP mode is released when an interrupt occurs, the wait time is determined by BTM setting. (See Table 7.2-1.) The time until oscillation becomes stable varies according to the type of oscillator being used and the supply voltage when the STOP mode is released. Therefore, select the wait time according to the conditions of use and set BTM before setting the STOP mode.

**Table 7.2-1 Wait Time Selection by Using BTM**

| BTM3 | BTM2 | BTM1 | BTM0 | Wait time* (  ) indicates $f_{XX}$ = 4.19 MHz |
|---|---|---|---|---|
| – | 0 | 0 | 0 | APPROX. $2^{20}/f_{XX}$ (Approx. 250 ms) |
| – | 0 | 1 | 1 | APPROX. $2^{17}/f_{XX}$ (Approx. 31.3 ms) |
| – | 1 | 0 | 1 | APPROX. $2^{15}/f_{XX}$ (Approx. 7.82 ms) |
| – | 1 | 1 | 1 | APPROX. $2^{13}/f_{XX}$ (Approx. 1.95 ms) |
| Other than above | | | | Use Prohibited |

Note: The wait time when STOP mode is released does not include the time until the clock begins to oscillate after STOP mode is released ((a) in the figure below) regardless of whether STOP mode was released by $\overline{\text{RESET}}$ input or interrupt generation.



### 7.3 Operation After Standby Mode is Released

(1) If the standby mode is released when $\overline{\text{RESET}}$ is input, normal reset operation is performed.

(2) If the standby mode is released when an interrupt request occurs, the contents of the interrupt master enable flag (IME) determine whether or not a vectored interrupt is made when the CPU restarts instruction execution.

(a) When IME = 0

After the standby mode released, execution restarts at the NOP instruction next to the standby mode setting instruction. The interrupt request flags are held.

(b) When IME = 1

After the standby mode is released, two instructions following the standby mode setting instruction are executed before a vectored interrupt is executed.

However, if the standby mode is released by using INTW or INT2 (testable input), no vectored interrupt will occur; processing as in (a) above is performed.

### 7.4 Application of Standby Mode

To use the standby mode, follow the procedure described below:

1) Detect a standby mode setting source such as interrupt input or port input for power off (it is effective to use INT4 for power off detection).
2) Handle input/output ports so that current consumption is minimized.
3) Specify interrupt to release the standby mode. (It is effective to use INT4. Clear the interrupt enables flags so as not to release the standby mode.)
4) Specify operation after the standby mode is released (set IME depending on whether or not interrupt service is made).
5) Specify the CPU clock after the standby mode is released. (To change the clock, wait for the required number of machine cycles before setting the standby mode.)
6) Select the wait time in release of standby mode.
7) Set the standby mode by using the STOP or HALT instruction.

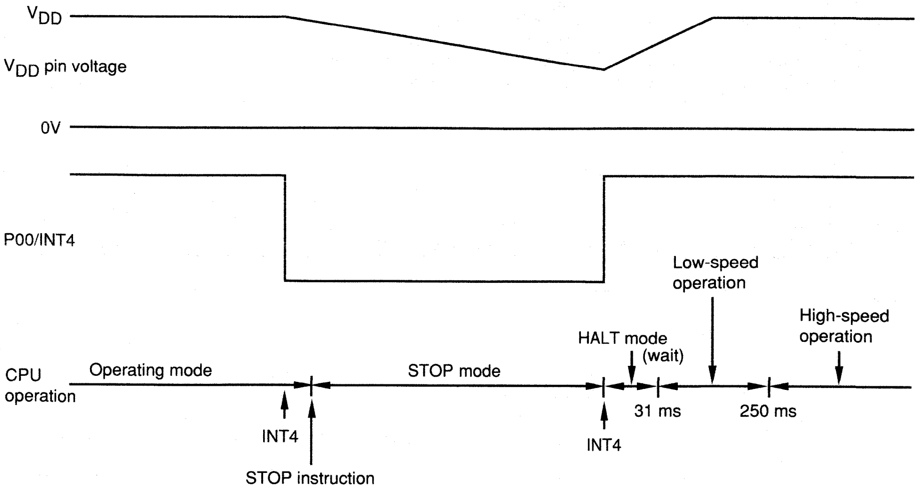
Use of the standby mode and system clock change function in combination enables the μPD753XX to operate at low current consumption and low voltage.

(1) Example of STOP mode application

Use the STOP mode under the following conditions

— Set the STOP mode when the INT4 falling edge is input and release it when the rising edge is input. (Do not use INTBT.)
— Place all input/output ports in high impedance.
— Use interrupts INT0 and INTT0 in the example program; however, do not use them to release the STOP mode.
— Enable interrupts after the STOP mode is released.
— After the STOP mode is released, start operation on the minimum speed CPU clock, and in 250 ms, change it to high-speed clock.
— Set the wait time for STOP mode release to about 31.3 ms.
— After the STOP mode is released, wait for 31.3 ms for the power supply to become stable. Check the P00/INT4 pin twice and remove chattering.

Timing Chart



Programming example – INT4 service program with MBE = 0 –

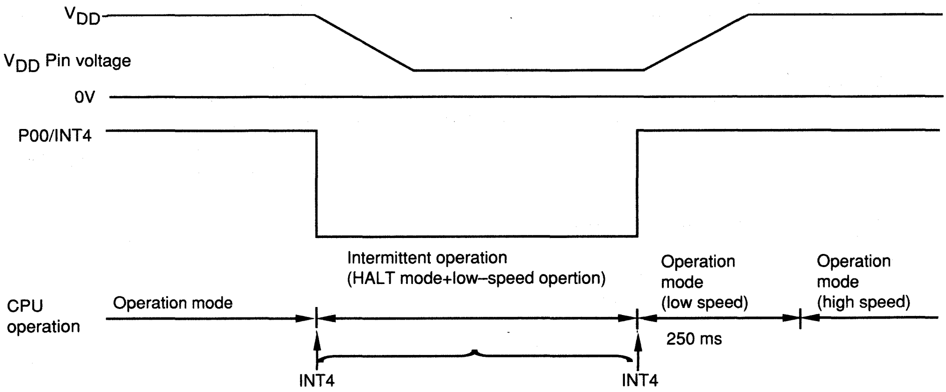
```
VSUB4:      SKT     PORT0.0          ; P00 = 1?
            BR      PDOWN            ; Power down
            SET1    BTM.3            ; Power on
WAIT:       SKT     IRQBT            ; Waits for 31.3 ms
            BR      WAIT
            SKT     PORT0.0          ; Check chattering
            BR      PDOWN
            MOV     A, #0011B
            MOV     PCC, A           ; Set high-speed mode
            MOV     XA, #XXH         ; Set Port Mode register
            MOV     PMGm, XA         ;
            EI      IE0
            EI      IET0
            RETI
PDOWN:      MOV     A, #0            ; Minimum speed mode
            MOV     PCC, A
            MOV     XA, #00H
            MOV     LCDM, XA         ; LCD display off
            MOV     LCDC, A
            MOV     PMGA, XA         ; Place input/output ports in high impedance
            MOV     PMGB, XA
            DI      IE0              ; Disables INT0 and INTT0
            DI      IET0
            MOV     A, #1011B
            MOV     BTM, A           ; Wait time = 31.3 ms
            STOP                     ; Sets STOP mode
            NOP
            RETI
```

(2) HALT mode application

Perform intermittent operation under the following conditions

— Change to the subsystem clock on the falling edge of INT4.
— Stop oscillation of the main system clock and set the HALT mode.
— Perform intermittent operation at 0.5sec. intervals during the standby mode.
— Again change to the main system clock on the rising edge of INT4.
— Do not use INTBT.

(Timing Chart)



```
Example (Initialization)
         MOV      A, #0011B
         MOV      PCC, A       ; High speed mode
         MOV      XA, #04
         MOV      WM, XA       ; Main system clock
         EI       IE4
         EI       IEW
         EI                    ; Enable interrupt
(Main routine)
         SKT      PORT0.0      ; Power OK?
         HALT                  ; Power down mode
         NOP                   ; Power OK?
         SKTCLR   IRQW         ; Is 0.5sec. flag set?
         BR       MAIN         ; NO
         CALL     WATCH        ; Watch subroutine
MAIN:      :
           :
           :
           :
(INT4 service routine)
INT4:    SKT      PORT0.0      ; Power OK?, MBE=0, RBE=0
         BR       PDOWN
         CLR1     SCC.3        ; Start oscillation of main system clock
         MOV      A, #8
         MOV      BTM, A
WAIT1:   SKT      IRQBT        ; Wait for 250 ms
         BR       WAIT1
         SKT      PORT0.0      ; Check chattering
         BR       PDOWN
         CLR1     SCC.0        ; Change to main system clock
         MOV      XA, #04H     ; Main system clock
```

```
        MOV     WM, XA
        RETI
PDOWN:  MOV     XA, #05H     ; Subsystem clock
        MOV     WM, XA
        MOV     XA, #00H
        MOV     LCDM, XA     ; LCD display off
        MOV     LCDC, A
        SET1    SCC.0        ; Change to subsystem clock
        MOV     A, #6
WAIT2:  INCS    A            ; Wait for 32 machine cycles
        BR      WAIT2
        SET1    SCC.3        ; Stop oscillation of main system clock
        RETI
```

Note: When the system clock is changed after power on from main system clock to subsystem clock, change the system clock after the subsystem clock is stabilized.

## 8. RESET FUNCTION

The μPD753XX is reset when $\overline{\text{RESET}}$ is input. The hardware devices are initialized as listed inTable 8.1-1. Fig. 8.1-1 shows the reset operation timing.

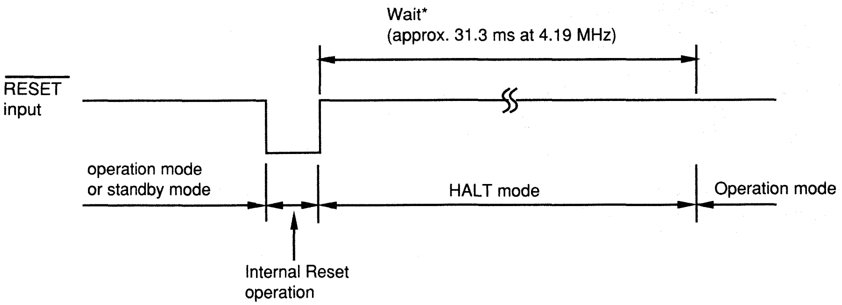

Wait*
(approx. 31.3 ms at 4.19 MHz)
$\overline{\text{RESET}}$ input
operation mode or standby mode
HALT mode
Operation mode
Internal Reset operation

**Fig. 8.1-1 Reset Operation by $\overline{\text{RESET}}$ is input**

**Table 8.1-1 State After Each Hardware Device is Reset**

| Hardware | | $\overline{\text{RESET}}$ input during standby mode | $\overline{\text{RESET}}$ input during operation |
|---|---|---|---|
| Program counter (PC) | μPD75304 | The low-order four bits of program memory address 0000H are loaded into PC11-PC8. The contents of address 0001H are loaded into PC7-PC0. | |
| | μPD75306,<br>μPD75308 | The low-order five bits of program memory address 0000H are loaded into PC12-PC8. The contents of address 0001H are loaded into PC7-PC0. | |
| | μPD75312,<br>μPD75316 | The low-order six bits of program memory address 0000H are loaded into PC13-PC8. The contents of address 0001H are loaded into PC7-PC0. | |
| PSW | Carry flag (CY) | Held | Undefined |
| | Skip flag (SK0-SK2) | 0 | 0 |
| | Interrupt status flag (IST0) | 0 | 0 |
| | Bank enable flag | Bit 7 of program memory address 0000H is loaded into MBE. | |
| Stack pointer (SP) | | Undefined | Undefined |
| Data memory (RAM) | | Held (Note) | Undefined |
| General-purpose registers (X, A, H, L, D, E, B, C) | | Held | Undefined |
| Bank selection register (MBS) | | 0 | 0 |
| Basic interval timer | Counter (BT) | Undefined | Undefined |
| | Mode register (BTM) | 0 | 0 |
| Timer event counter | Counter (T0) | 0 | 0 |
| | Modulo register (TMOD0) | FFH | FFH |
| | Mode register (TM0) | 0 | 0 |
| | TOE0, TOUT F/F | 0, 0 | 0, 0 |
| Watch timer | Mode register (WM) | 0 | 0 |
| Serial interface | Shift register (SIO) | Held | Undefined |

**Table 8.1-1 State After Each Hardware Device is Reset (Cont'd)**

| Hardware | | $\overline{\text{RESET}}$ input during standby mode | $\overline{\text{RESET}}$ input during operation |
|---|---|---|---|
| Serial interface | Operation mode register (CSIM) | 0 | 0 |
| | SBI control register (SBIC) | 0 | 0 |
| | Slave address register (SVA) | Held | Undefined |
| Clock generator and clock output circuit | Processor clock control register (PCC) | 0 | 0 |
| | System clock control register (SCC) | 0 | 0 |
| | Clock output mode register (CLOM) | 0 | 0 |
| LCD controller | Display mode register (LCDM) | 0 | 0 |
| | Display control register (LCDC) | 0 | 0 |
| Interrupt function | Interrupt request flags (IRQXXX) | Reset to 0 | Reset to 0 |
| | Interrupt enable flags (IEXXX) | 0 | 0 |
| | Interrupt master enable flag (IME) | 0 | 0 |
| | INT0, INT1, and INT2 mode registers (IM0, IM1, and IM2) | 0, 0, 0 | 0, 0, 0 |
| Digital ports | Output buffers | Off | Off |
| | Output latches | Cleared | Cleared |
| | Input/output mode registers (PMGA, B) | 0 | 0 |
| | Pull-up resistor specification register (POGA) | 0 | 0 |
| Bit sequential buffer (BSB0-BSB3) | | Held | Undefined |
| Pins condition | P00-P03, P10-P13, P20-P23, P30-P33, P60-P63, P70-P73, | Input | Input |
| | P40-P43, P50-P53, | — At incorporated pull-up resistor: ... High level<br>— At open drain: ... High impedance | |
| | S0-S23, COM0-COM3 | Note 2 | |
| | BIAS | — At incorporated split resistor: ... Low level<br>— At not incorporated split resistor: ... High impedance | |

Note: The data of data memory address 0F8H-0FDH is undefined by $\overline{\text{RESET}}$ input.

Note2: Each display output is selected as $V_{LCX}$ input source.

S0-31: $V_{LC1}$
COM0-COM2: $V_{LC2}$
COM3: $V_{LC0}$

However, each display output level is changed by each display output and $V_{LCX}$ external circuit.

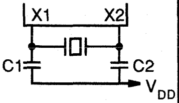
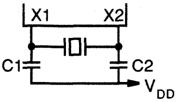
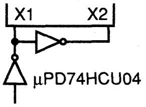
## 9. ELECTRICAL CHARACTERISTICS

**Absolute Maximum Ratings** (Ta = 25 °C)

| Item | Symbol | Conditions | | Ratings | Unit |
|---|---|---|---|---|---|
| Supply voltage | $V_{DD}$ | | | –0.3 to +7.0 | V |
| Input voltage | $V_{I1}$ | Other than ports 4 and 5 | | –0.3 to $V_{DD}$ +0.3 | V |
| | $V_{I2}$ (1) | Ports 4 and 5 | With internal pull-up resistor | –0.3 to $V_{DD}$ +0.3 | V |
| | | | Open drain | –0.3 to +11 | V |
| Output voltage | $V_O$ | | | –0.3 to $V_{DD}$ +0.3 | V |
| High level output current | $I_{OH}$ | Single pin | | –15 | mA |
| | | Total, all outputs | | –30 | |
| Low level output current | $I_{OL}$<br>Note | Single pin | Peak value | 30 | mA |
| | | | Effective value | 15 | |
| | | Ports 0, 2, 3 and 5 total | Peak value | 100 | |
| | | | Effective value | 60 | |
| | | Ports 4, 6, and 7 total | Peak value | 100 | |
| | | | Effective value | 60 | |
| Operation Temperature | $T_{opt}$ | | | –40 to +85 | °C |
| Storage Temperature | $T_{stg}$ | | | –65 to +150 | °C |

Note: Use the following formula to calculate the effective value. (Effective value) = (Peak value) x $\sqrt{\text{duty}}$

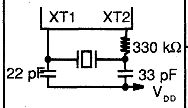
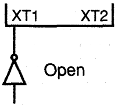
**Characteristics of Main System Clock Oscillator** (Ta = –40 to +85 °C, $V_{DD}$ = 2.7 to 6.0V)

| Oscillator | Recommended constants | Item | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Ceramic resonator |  | Note 1<br>Oscillation frequency ($f_{XX}$) | | 1.0 | | 5.0 | MHz |
| | | Note 2<br>Oscillation stabilization time | After $V_{DD}$ reaches the minimum value in the oscillator voltage range | | | 4 | ms |
| Crystal resonator |  | Note 1<br>Oscillation frequency ($f_{XX}$) | | 1.0 | 4.19 | 5.0 | MHz |
| | | Note 2<br>Oscillation stabilization time | $V_{DD}$ =4.5 to 6.0V | | | 10 | ms |
| | | | | | | 30 | ms |
| External clock |  | Note 1<br>X1 input frequency ($f_X$) | | 1.0 | | 5.0 | MHz |
| | | X1 input high/low level width ($t_{XH}$, $t_{XL}$) | | 100 | | 500 | ns |

Note 1: The oscillation frequency and X1 input frequency are indicated only to express the characteristics of the oscillator. Refer to the AC characteristics for instruction execution time.

Note 2: The oscillation stabilization time is the time required for the oscillator to stabillate after $V_{DD}$ is applied or after the STOP mode is released.

**Subsystem Clock Oscillator Characteristics** (Ta = –40 to +85 °C, $V_{DD}$ = 2.7 to 6.0V)

| Oscillator | Recommended constants | Item | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Crystal resonator |  | Oscillation frequency ($f_{XT}$) | | 32 | 32.768 | 35 | MHz |
| | | Oscillation stabilization time | $V_{DD}$ = 4.5 – 6.0V | | 1 | 2 | s |
| | | | | | | 10 | s |
| External clock |  | XT1 input frequency ($f_{XT}$) | | 32 | | 100 | kHz |
| | | XT1 input high/low level width ($t_{XTH}$, $t_{XTL}$) | | 5 | | 15 | µs |

**Recommended Resonator**

Main system clock: ceramic resonator

| Maunfacture | Product name | External capacitors (pF) | | Oscillator operating voltage range (V) | | Remarks |
|---|---|---|---|---|---|---|
| | | C1 | C2 | MIN. | MAX. | |
| Murata | CSA 2.00MG093 | 15 | 15 | 2.5 | 3.5 | Note |
| | CSB 1000D20 | 220 | 220 | 2.7 | 6.0 | |
| | CSA 2.00MG093<br>CSA 4.19MGU<br>CSA 4.91MGU | 30 | 30 | | | |
| | CST 2.00MG093<br>CST 4.19MGU<br>CST 4.91MGU | not required | not required | | | Internal C type |
| Kyocera | KBR–1000H | 100 | 100 | 2.7 | 6.0 | |
| | KBR–2.0MS | 47 | 47 | | | |
| | KBR–4.0MS<br>KBR–4.19MS<br>KBR–4.91MS | 33 | 33 | | | |

Note: When CSA 2.000G093 is used, VDD is 2.5 to 3.5V.

Main system clock: crystal resonator

| Maunfacture | Freg. Holder | External capacitors (pF) | | Oscillator operating voltage range (V) | | Remarks |
|---|---|---|---|---|---|---|
| | | C1 | C2 | MIN. | MAX. | |
| Kinseki | 1.00<br>2.00 HC–18/U<br>4.19 HC–49/U<br>4.91 HC–43/U | 22 | 22 | 2.7 | 6.0 | |

Subsystem clock: 32.768 KHz crystal resonator

| Maunfacture | Product name | External capacitors (pF) | | | Oscillator operating voltage range (V) | | Remarks |
|---|---|---|---|---|---|---|---|
| | | C1 (pF) | C2 (pF) | R (kΩ) | MIN. | MAX. | |
| Kinseki | P–3 | 22 | 22 | 330 | 2.7 | 6.0 | |

**Capacitance** (Ta = 25 °C, $V_{DD}$ = 0V)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Units |
|---|---|---|---|---|---|---|
| Input capacitance | $C_{IN}$ | f =1 MHz<br>For other than pins to be measured, 0V. | | | 15 | pF |
| Output capacitance | $C_{OUT}$ | | | | 15 | pF |
| Input/output capacitance | $C_{IO}$ | | | | 15 | pF |

**DC Characteristics** (Ta = –40 to +85 °C, $V_{DD}$ = 2.7 to 6.0V)

| Item | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| High level input voltage | $V_{IH1}$ | Ports 2 and 3 | | $0.7V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH2}$ | Ports 0, 1, 6, 7, and $\overline{\text{RESET}}$ pin | | $0.8V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH3}$ | Ports 4 to 5 | With built-in pullup resistor | $0.7V_{DD}$ | | $V_{DD}$ | V |
| | | | Open drain | $0.7V_{DD}$ | | 10 | V |
| | $V_{IH4}$ | X1, X2, XT1 | | $V_{DD}$–0.5 | | $V_{DD}$ | V |
| Low level input voltage | $V_{IL1}$ | Ports 2, 3, 4, 5 | | 0 | | $0.3V_{DD}$ | V |
| | $V_{IL2}$ | Ports 0, 1, 6, 7, and $\overline{\text{RESET}}$ pin | | 0 | | $0.2V_{DD}$ | V |
| | $V_{IL3}$ | X1, X2, XT1 | | 0 | | 0.4 | V |
| High level output voltage | $V_{OH1}$ | Ports 0, 2, 3, 6, 7, BIAS | $V_{DD}$ = 4.5 to 6.0V<br>$I_{OH}$ = –1mA | $V_{DD}$–1.0 | | | V |
| | | | $I_{OH}$ = –100µA | $V_{DD}$–0.5 | | | V |
| | $V_{OH2}$ | BP0–7 (with two $I_{OH}$ outputs) | $V_{DD}$ = 4.5 to 6.0V<br>$I_{OH}$ = –100µA | $V_{DD}$–2.0 | | | V |
| | | | $I_{OH}$ = –30µA | $V_{DD}$–1.0 | | | V |
| Low level output voltage | $V_{OL1}$ | Port 0, 2 – 7 | Ports 3, 4, 5<br>$V_{DD}$ = 4.5 to 6.0V<br>$I_{OL}$ = 15mA | | 0.4 | 2.0 | V |
| | | | $V_{DD}$ = 4.5 to 6.0V<br>$I_{OL}$ = 1.6mA | | | 0.4 | V |
| | | | $I_{OL}$ = 400µA | | | 0.5 | V |
| | | SB0, 1 | Open drain Pull-up register ≥ 1kΩ | | | $0.2V_{DD}$ | V |
| | $V_{OL2}$ | BP0-7 (with two $I_{OL}$ outputs) | $V_{DD}$ = 4.5–6.0V<br>$I_{OL}$ = 100µA | | | 1.0 | V |
| | | | $I_{OL}$ = 50µA | | | 1.0 | V |
| High level input leakage current | $I_{LIH1}$ | $V_{IN}$ = $V_{DD}$ | Other than indicated below | | | 3 | µA |
| | $I_{LIH2}$ | | X1, X2, XT1 | | | 20 | µA |
| | $I_{LIH3}$ | $V_{IN}$ = 10V | Ports 4, 5 (with open drain) | | | 20 | µA |
| Low level input leakage current | $I_{LIL1}$ | $V_{IN}$ = 0V | Other than indicated below | | | –3 | µA |
| | $I_{LIL2}$ | | X1, X2, XT1 | | | –20 | µA |
| High level output leakage current | $I_{LOH1}$ | $V_{OUT}$ = $V_{DD}$ | Other than indicated below | | | 3 | µA |
| | $I_{LOH2}$ | $V_{OUT}$ = 10V | Ports 4 and 5 (with open drain) | | | 20 | µA |
| Low level output leakage current | $I_{LOL}$ | $V_{OUT}$ = 0V | | | | –3 | µA |
| Internal pull-up resistor | $R_{L1}$ | Ports 0, 1, 2, 3, 6, 7 (Except P00)<br>$V_{IN}$ = 0V | $V_{DD}$ = 5.0V ± 10% | 15 | 40 | 80 | kΩ |
| | | | $V_{DD}$ = 3.0V ± 10% | 30 | | 200 | kΩ |

**DC Characteristics** (Ta = –40 to +85 °C, $V_{DD}$ = 2.7 to 6.0V)

| Item | Symbol | Conditions | | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|---|
| Internal pull-up resistor | $R_{L2}$ | Ports 4 and 5 | $V_{DD}$ = 5.0V ± 10% | | 15 | 40 | 70 | kΩ |
| | | $V_{OUT}$=$V_{DD}$–2.0V | $V_{DD}$ = 3.0V ± 10% | | 10 | | 60 | kΩ |
| LCD drive voltage | $V_{LCD}$ | | | | 2.5 | | $V_{DD}$ | V |
| LCD split resistor | $R_{LCD}$ | | | | 60 | 100 | 150 | kΩ |
| LCD output voltage deviation (common) | $V_{ODC}$ | $I_O$ = ±5μA | $V_{LCD0} = V_{LCD}$<br>$V_{LCD1} = V_{LCD} \times {}^2/_3$<br>$V_{LCD2} = V_{LCD} \times {}^1/_3$ (Note 4) | | 0 | | ±0.2V | V |
| LCD output voltage deviation (segment) | $V_{ODS}$ | $I_O$ = ±1μA | | | 0 | | ±0.2V | V |
| Supply current (Note 1) | $I_{DD1}$ | (Note 5) 4.19 MHz crystal oscillation C1=C2=22pF | $V_{DD}$ = 5V ± 10% (Note 2) | | | 2.5 | 8 | mA |
| | | | $V_{DD}$ = 3V ± 10% (Note 3) | | | 0.35 | 1.2 | mA |
| | $I_{DD2}$ | | HALT Mode | $V_{DD}$ = 5V ± 10% | | 500 | 1500 | μA |
| | | | | $V_{DD}$ = 3V ± 10% | | 150 | 450 | μA |
| | $I_{DD3}$ | 32kHz crystal resonator (Note 6) | $V_{DD}$ = 3V ± 10% | | | 30 | 90 | μA |
| | $I_{DD4}$ | | HALT Mode | $V_{DD}$ = 3V ± 10% | | 5 | 15 | μA |
| | $I_{DD5}$ | XT1 = 0 STOP mode | $V_{DD}$ = 5V ± 10% | | | 0.5 | 20 | μA |
| | | | $V_{DD}$ = 3V ± 10% | | | 0.1 | 10 | μA |
| | | | | Ta = 25°C | | 0.1 | 5 | μA |

Note 1: The currents of the built-in pull-up resistor and the LCD step-down resistor are not included.
Note 2: When operated in the high-speed mode with the processor clock control register (PCC) set to 0011.
Note 3: When operated in the low-speed mode with the PCC set to 0000.
Note 4: 2.7V ≤ $V_{LCD}$ ≤ $V_{DD}$
Note 5: Includes the power consumption for the sub system oscillation.
Note 6: When the system clock control registor (SCC) is set to 1001, sub system clock oscillation is stopped and operated by subsystem clock.

**AC Characteristics** (Ta = –40 to +85 C, $V_{DD}$ = 2.7 to 6.0V)

| Item | Symbol | Condition | | MIN. | TYP. | MAX | Unit |
|---|---|---|---|---|---|---|---|
| (Note 1) Cycle time (minimum instruction execution time) | $t_{CY}$ | Operation with main system clock | $V_{DD}$ = 4.5 –6.0V | 0.95 | | 64 | μs |
| | | | | 3.8 | | 64 | μs |
| | | Operation with sub-system clock | | 114 | 122 | 125 | μs |
| TI0 input frequency | $f_{TI}$ | $V_{DD}$ = 4.5 – 6.0V | | 0 | | 1 | MHz |
| | | | | 0 | | 275 | kHz |
| TI0 input high/low level width | $t_{TIH}$ | $V_{DD}$ = 4.5 – 6.0V | | 0.48 | | | μs |
| | $t_{TIL}$ | | | 1.8 | | | μs |

**AC Characteristics** (Ta = –40 to 85 C, $V_{DD}$ = 2.7 to 6.0V) (cont'd)

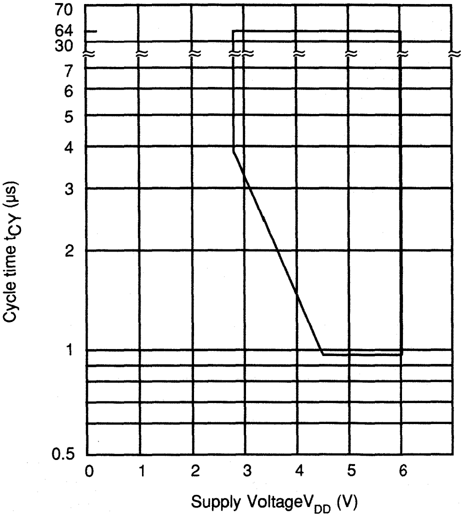
| Item | Symbol | Condition | MIN. | TYP. | MAX | Unit |
|---|---|---|---|---|---|---|
| Interrupt input high/low level width | $t_{INTH}$ | INT0 | Note 2 | | | μs |
| | | INT1, 2, 4 | 10 | | | μs |
| | $t_{INTL}$ | KR0-7 | 10 | | | μs |
| $\overline{\text{RESET}}$ low level width | $t_{RSL}$ | | 10 | | | μs |

Note 1: The cycle time (minimum instruction execution time) is determined by the oscillation frequency of the connected resonator, the system clock control register (SCC), and the processor clock control register (PCC). The figure below shows the $V_{DD}$ vs cycle time ($t_{CY}$) when operated with the main system clock.

Note 2: $2t_{CY}$ or $128/f_X$, depending on the setting of the interrupt mode register (IM0).

(with main system clock)

$t_{CY}$ vs. $V_{DD}$



**Serial Transfer Operation**

2-line/3line serial I/O mode ($\overline{\text{SCK}}$ ... Internal clock output)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY1}$ | $V_{DD}$ = 4.5 –6.0V | 1600 | | | ns |
| | | | 3800 | | | ns |
| $\overline{\text{SCK}}$ high/low level width | $t_{KL1}$<br>$t_{KH1}$ | $V_{DD}$ = 4.5 –6.0V | $\frac{t_{KCY1}}{2}$ -50 | | | ns |
| | | | $\frac{t_{KCY1}}{2}$ -150 | | | ns |
| SI set-up time (against $\overline{\text{SCK}}$ ↑ ) | $t_{SIK1}$ | | 150 | | | ns |
| SI hold time (against $\overline{\text{SCK}}$ ↑ ) | $t_{KSI1}$ | | 400 | | | ns |
| $\overline{\text{SCK}}$ ↓ → SO output delay time | $t_{KSO1}$ | $V_{DD}$ = 4.5 –6.0V | | | 250 | ns |
| | | | | | 1000 | ns |

2-line/3line serial I/O mode ($\overline{\text{SCK}}$ ... External clock input)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY2}$ | $V_{DD}$ = 4.5 –6.0V | 800 | | | ns |
| | | | 3200 | | | ns |
| $\overline{\text{SCK}}$ high/low level width | $t_{KL2}$<br>$t_{KH2}$ | $V_{DD}$ = 4.5 –6.0V | 400 | | | ns |
| | | | 1600 | | | ns |
| SI set-up time (against $\overline{\text{SCK}}$ ↑ ) | $t_{SIK2}$ | | 100 | | | ns |
| SI hold time (against $\overline{\text{SCK}}$ ↑ ) | $t_{KSI2}$ | | 400 | | | ns |
| $\overline{\text{SCK}}$ ↓ → SO output delay time | $t_{KSO2}$ | $V_{DD}$ = 4.5 –6.0V | | | 300 | ns |
| | | | | | 1000 | ns |

Note: The output delay time (for rising edge) of the serial line must be shorter than 600 ns. For example, if SB0 and SBI are pulled up with 5K ohms, the total capacitance of the serial bus line must be no greater than 120 pF.

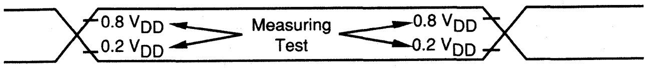
SBI mode ($\overline{\text{SCK}}$ ... Internal clock output (master))

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY3}$ | $V_{DD}$ = 4.5 –6.0V | 1600 | | | ns |
| | | | 3800 | | | ns |
| $\overline{\text{SCK}}$ high/low level width | $t_{KL3}$<br>$t_{KH3}$ | $V_{DD}$ = 4.5 –6.0V | $\frac{t_{KCY1}}{2}$ -50 | | | ns |
| | | | $\frac{t_{KCY1}}{2}$-150 | | | ns |
| SB0,1 set-up time (against $\overline{\text{SCK}}$ ↑ ) | $t_{SIK3}$ | | 150 | | | ns |
| SB0,1 hold time (against $\overline{\text{SCK}}$ ↑ ) | $t_{KSI3}$ | | $\frac{t_{KCY1}}{2}$ | | | ns |
| $\overline{\text{SCK}}$↓ → SB0,1 output delay time | $t_{KSO3}$ | $V_{DD}$ = 4.5 –6.0V | 0 | | 250 | ns |
| | | | 0 | | 1000 | ns |
| $\overline{\text{SCK}}$ ↑→ SB0,1↓ | $t_{KSB}$ | | $t_{KCY}$ | | | ns |
| SB0,1↓→ $\overline{\text{SCK}}$ ↓ | $t_{SBK}$ | | $t_{KCY}$ | | | ns |
| SB0,1 low level width | $t_{SBL}$ | | $t_{KCY}$ | | | ns |
| SB0,1 high level width | $t_{SBH}$ | | $t_{KCY}$ | | | ns |

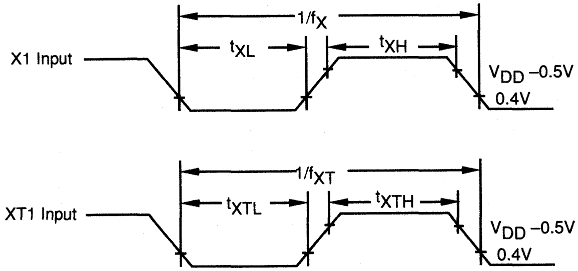
SBI mode ($\overline{\text{SCK}}$ ... External clock input (slave))

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY4}$ | $V_{DD}$ = 4.5 –6.0V | 800 | | | ns |
| | | | 3200 | | | ns |
| $\overline{\text{SCK}}$ high/low level width | $t_{KL4}$<br>$t_{KH4}$ | $V_{DD}$ = 4.5 –6.0V | 400 | | | ns |
| | | | 1600 | | | ns |
| SB0,1 set-up time (against $\overline{\text{SCK}}$ ↑ ) | $t_{SIK4}$ | | 100 | | | ns |
| SB0,1 hold time (against $\overline{\text{SCK}}$ ↑ ) | $t_{KSI4}$ | | $\frac{t_{KCY1}}{2}$ | | | ns |
| $\overline{\text{SCK}}$↓ → SB0,1 output delay time | $t_{KSO4}$ | $V_{DD}$ = 4.5 –6.0V | 0 | | 300 | ns |
| | | | 0 | | 1000 | ns |
| $\overline{\text{SCK}}$ ↑→ SB0,1↓ | $t_{KSB}$ | | $t_{KCY}$ | | | ns |
| SB0,1↓→ $\overline{\text{SCK}}$ ↓ | $t_{SBK}$ | | $t_{KCY}$ | | | ns |
| SB0,1 low level width | $t_{SBL}$ | | $t_{KCY}$ | | | ns |
| SB0,1 high level width | $t_{SBH}$ | | $t_{KCY}$ | | | ns |

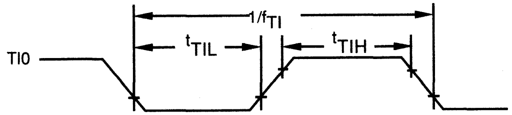
AC Timing Measurement Points (Except X1 and XT1 inputs)



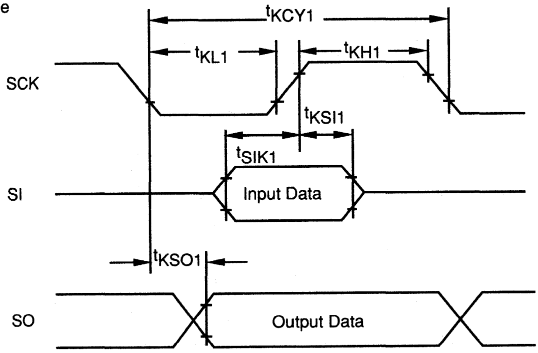
Clock Timing



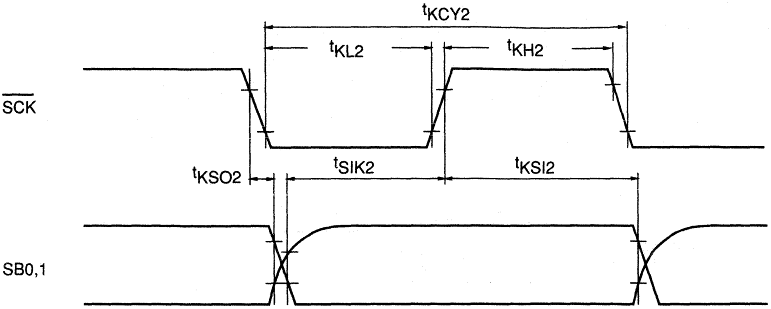
TIO Timing



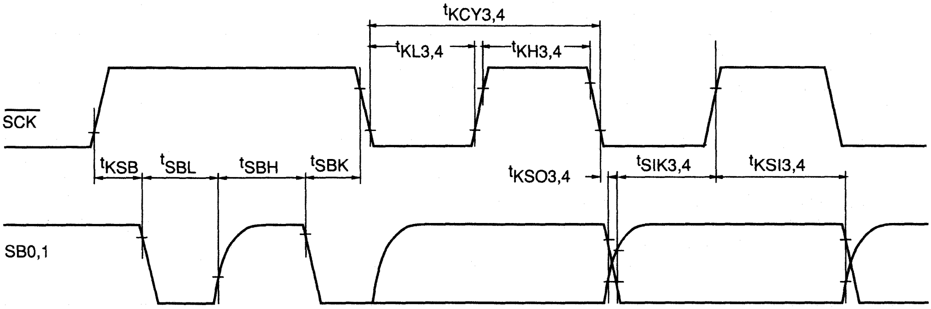
Serial Transfer Timing
3-line serial I/O mode



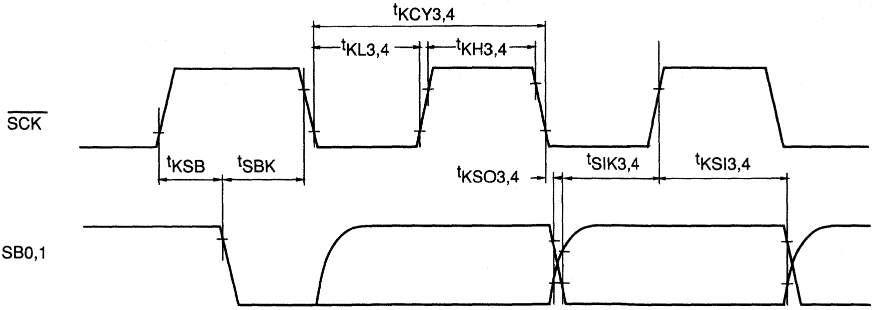
2-line serial I/O mode



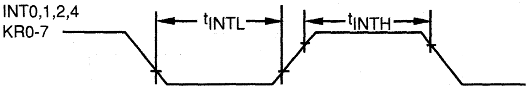
**Serial Transfer Timing**
Bus release signal transfer:



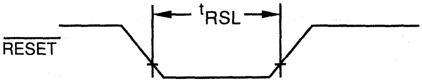
Command signal transfer:



Interrupt input timing



$\overline{\text{RESET}}$ input timing



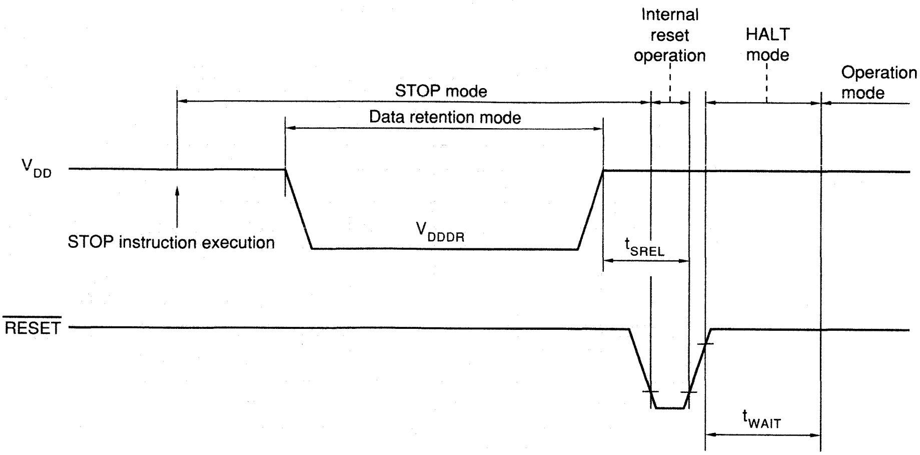
**Data Memory STOP Mode Low Voltage Data Retention Characteristic** (Ta = –40 to +85 °C)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Data retention voltage | $V_{DDDR}$ | | 2.0 | | 6.0 | V |
| Data retention current (Note1) | $I_{DDDR}$ | $V_{DDR}$ = 2.0V | | 0.1 | 10 | μA |
| Release signal set time | $t_{SREL}$ | | 0 | | | μs |
| Oscillation stabilization time (Note 2) | $t_{WAIT}$ | Release by $\overline{\text{RESET}}$ input | | $2^{17}/f_X$ | | ms |
| | | Release by interrupt request | | Note 3 | | ms |

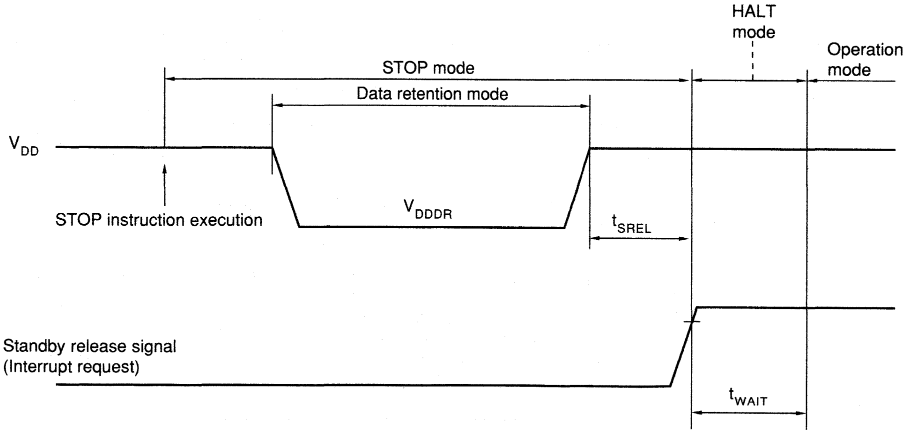
Note 1: Current in the internal pull-up resistors is not included.
Note 2: The oscillation stabilization time is the time required before beginning CPU operation in order to prevent unstable CPU operation when oscillation is initiated.
Note 3: Depends on the setting of the basic interval timer mode register (BTM) (refer to the table below).

| BTM3 | BTM2 | BTM1 | BTM0 | WAIT time ( ) indicates $f_X$ =4.19MHz |
|---|---|---|---|---|
| — | 0 | 0 | 0 | $2^{20}/f_X$ (Approximately 250 ms) |
| — | 0 | 1 | 1 | $2^{17}/f_X$ (Approximately 31.3 ms) |
| — | 1 | 0 | 1 | $2^{15}/f_X$ (Approximately 7.82 ms) |
| — | 1 | 1 | 1 | $2^{13}/f_X$ (Approximately 1.95 ms) |

Data Retention Timing (when STOP mode is released by $\overline{\text{RESET}}$ input)



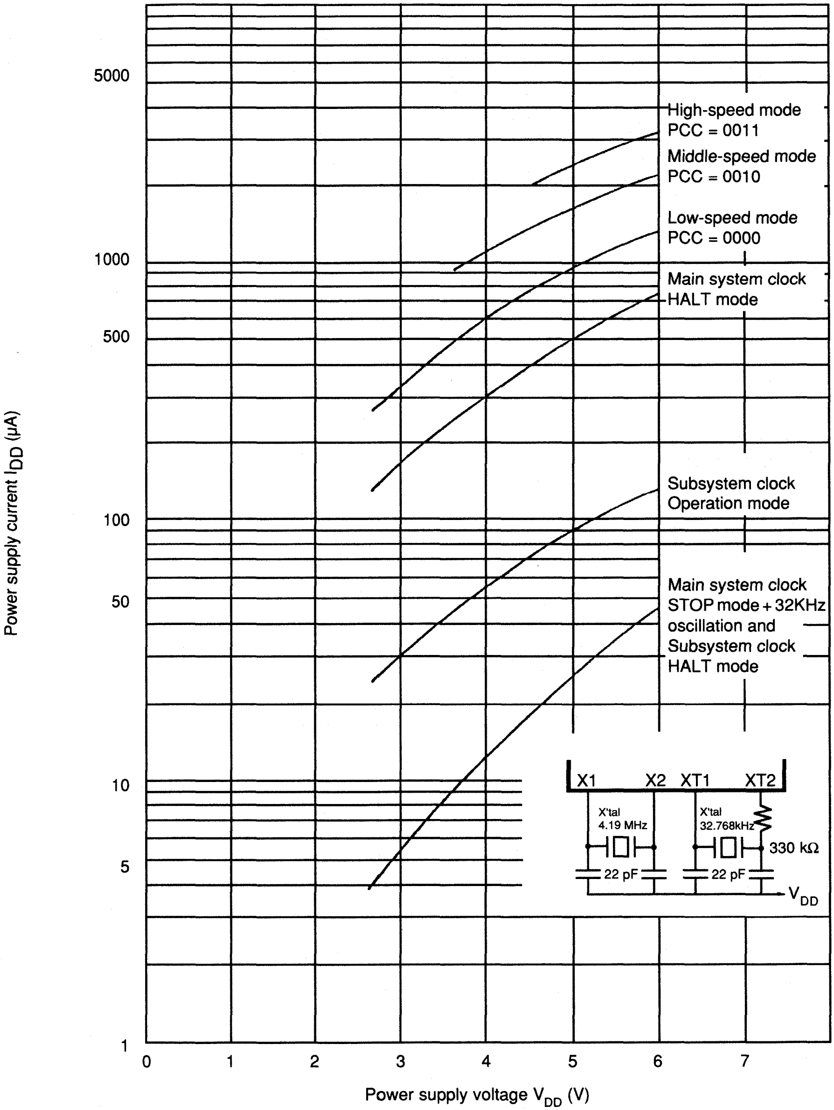
Data Retention Timing
(Standby release signal: STOP mode release by interrupt signal)



## Characteristic Curves

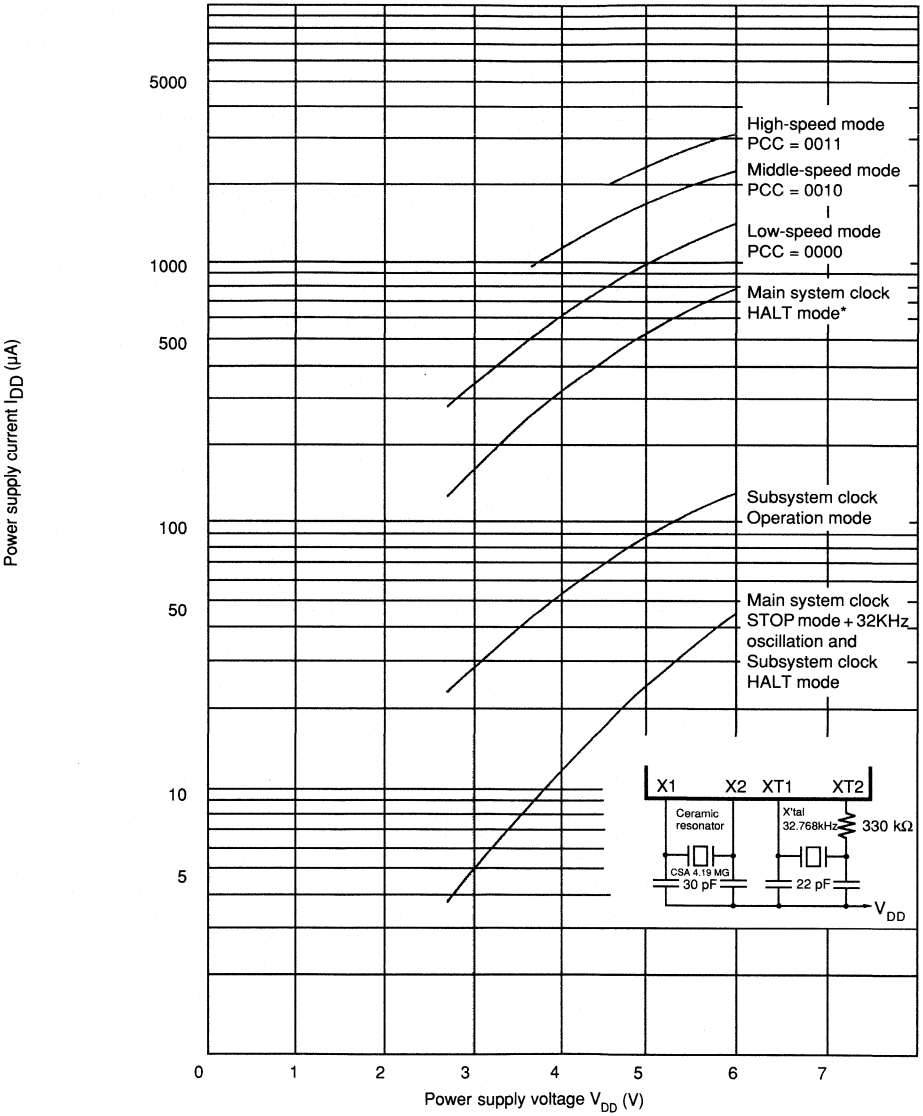
$I_{DD}$ vs $V_{DD}$ (Crystal oscillation)

(Ta = 25 °C)



$I_{DD}$ vs $V_{DD}$ (Ceramic oscillation)

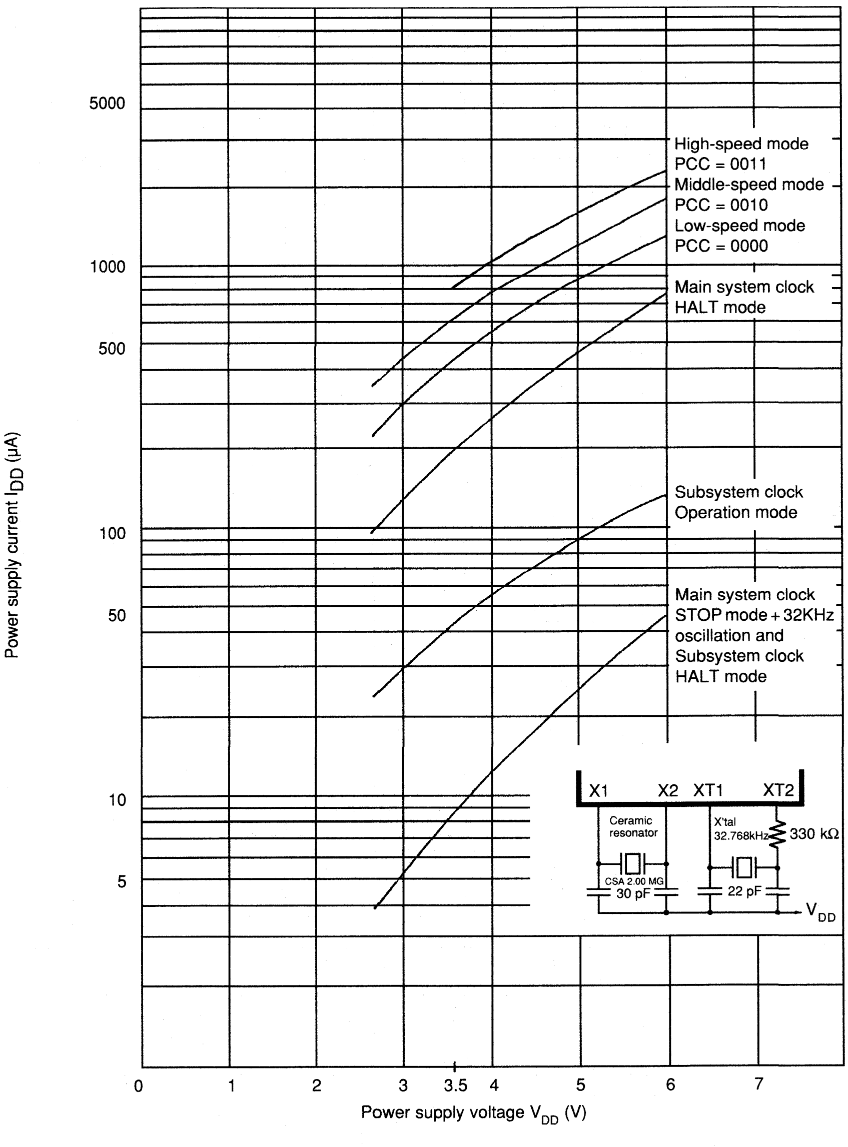
(Ta = 25 °C)



*The Current values are increased by approximately 10% compared to when operating with a crystal resonator.

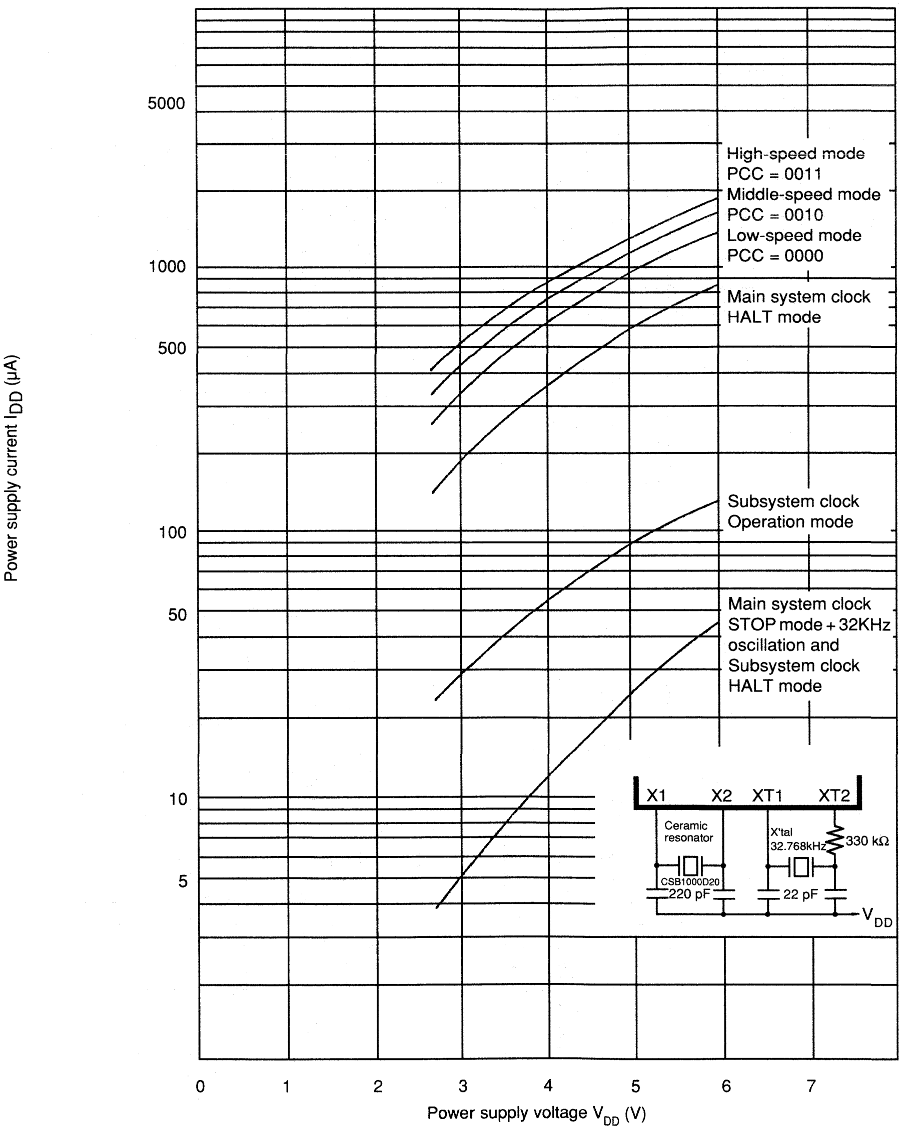
$I_{DD}$ vs $V_{DD}$ (Ceramic oscillation)

(Ta = 25 °C)

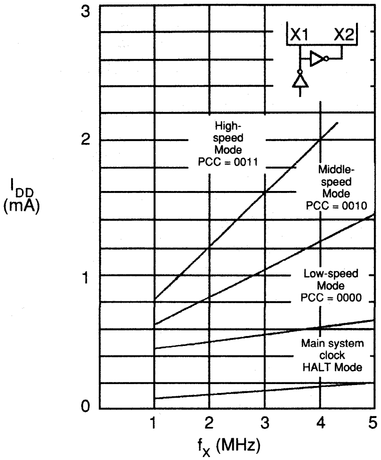


$I_{DD}$ vs $V_{DD}$ (Ceramic oscillation)

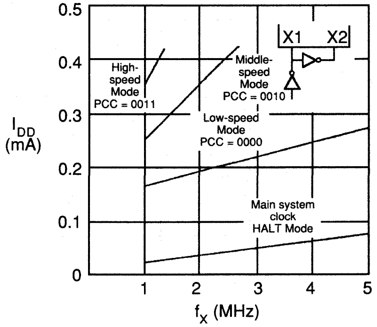
(Ta = 25 °C)



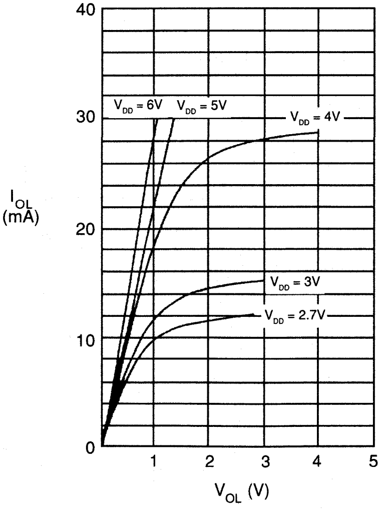
$I_{DD}$ vs $V_{DD}$ ($V_{DD}$ = 5V, Ta = 25 °C)



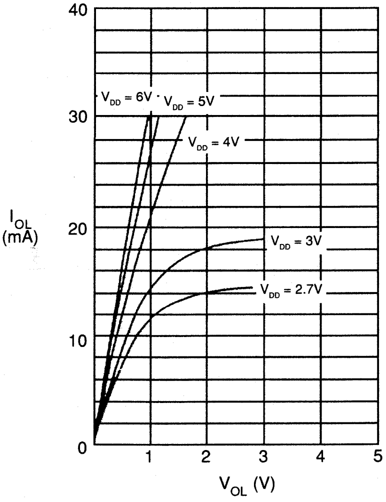
$I_{DD}$ vs $f_X$ ($V_{DD}$ = 3V, Ta = 25 °C)



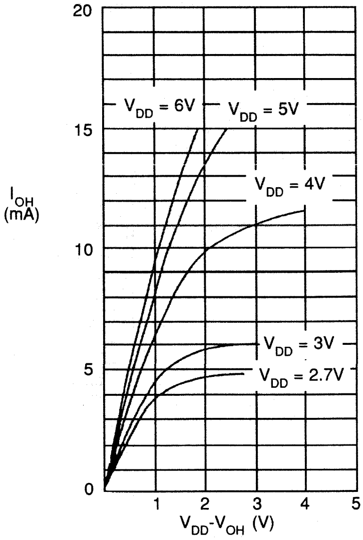
$V_{OL}$ vs $I_{OL}$ (Ports 0, 2, 6, 7, Ta = 25 °C)



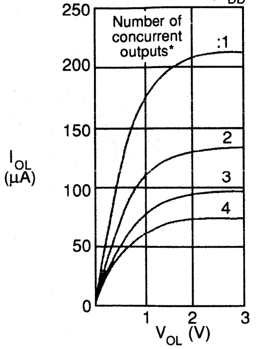
$V_{OL}$ vs $I_{OL}$ (Ports 3, 4, 5, Ta = 25 °C)



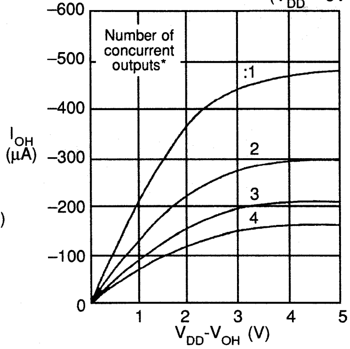
$V_{OH}$ vs $I_{OH}$ (Ta = 25 °C)



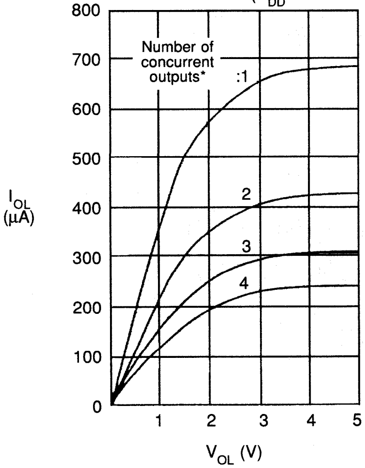
$V_{OL}$ vs $I_{OL}$ (BP0-3, BP4-7)
($V_{DD}$ = 3V Ta = 25 °C)



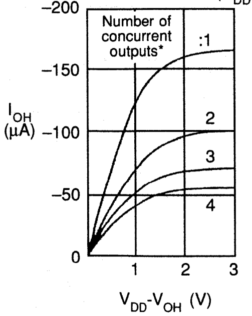
$V_{OH}$ vs $I_{OH}$ (BP0-3, BP4-7)
($V_{DD}$ = 5V Ta = 25 °C)



$V_{OL}$ vs $I_{OL}$ (BP0-3, BP4-7)
($V_{DD}$ = 5V Ta = 25 °C)



$V_{OH}$ vs $I_{OH}$ (BP0-3, BP4-7)
($V_{DD}$ = 3V Ta = 25 °C)

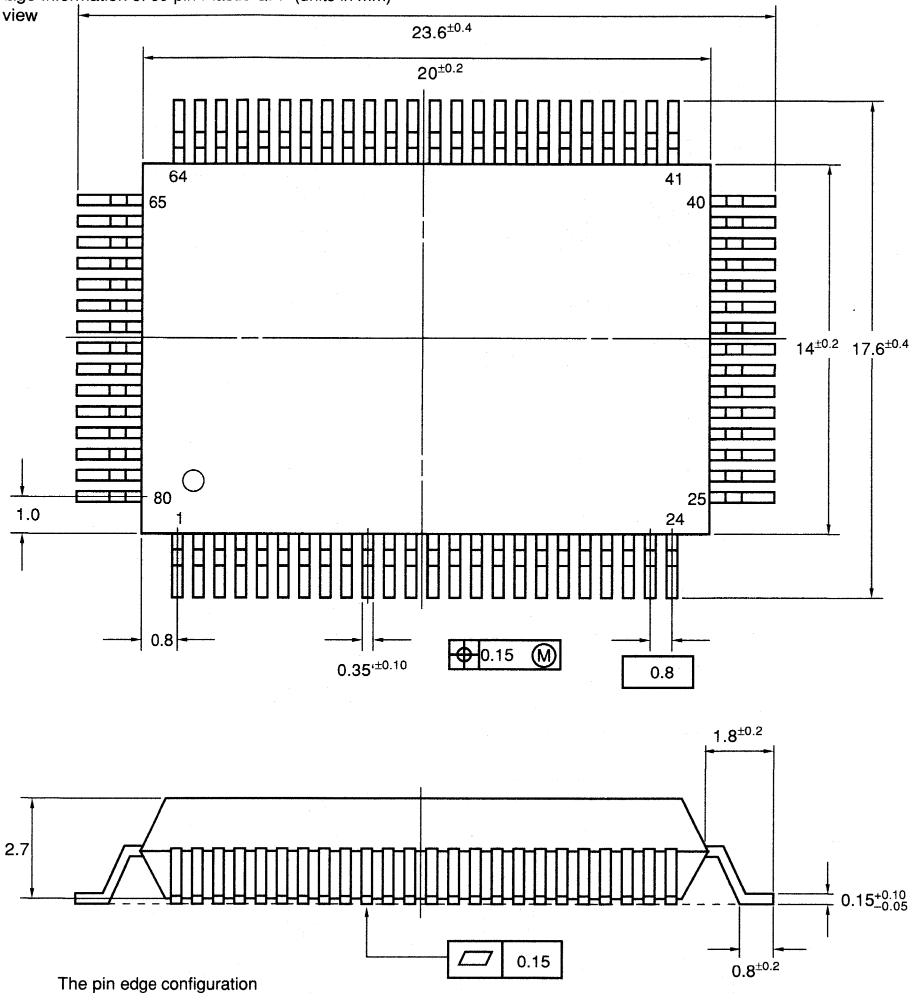


* Of pins BP0 to BP3 or BP4 to BP7, number of pins outputting the same level.

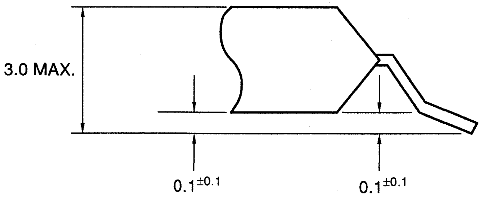
### 9.1 Package Information

Package Information of 80-pin Plastic QFP (units in mm)

Top view

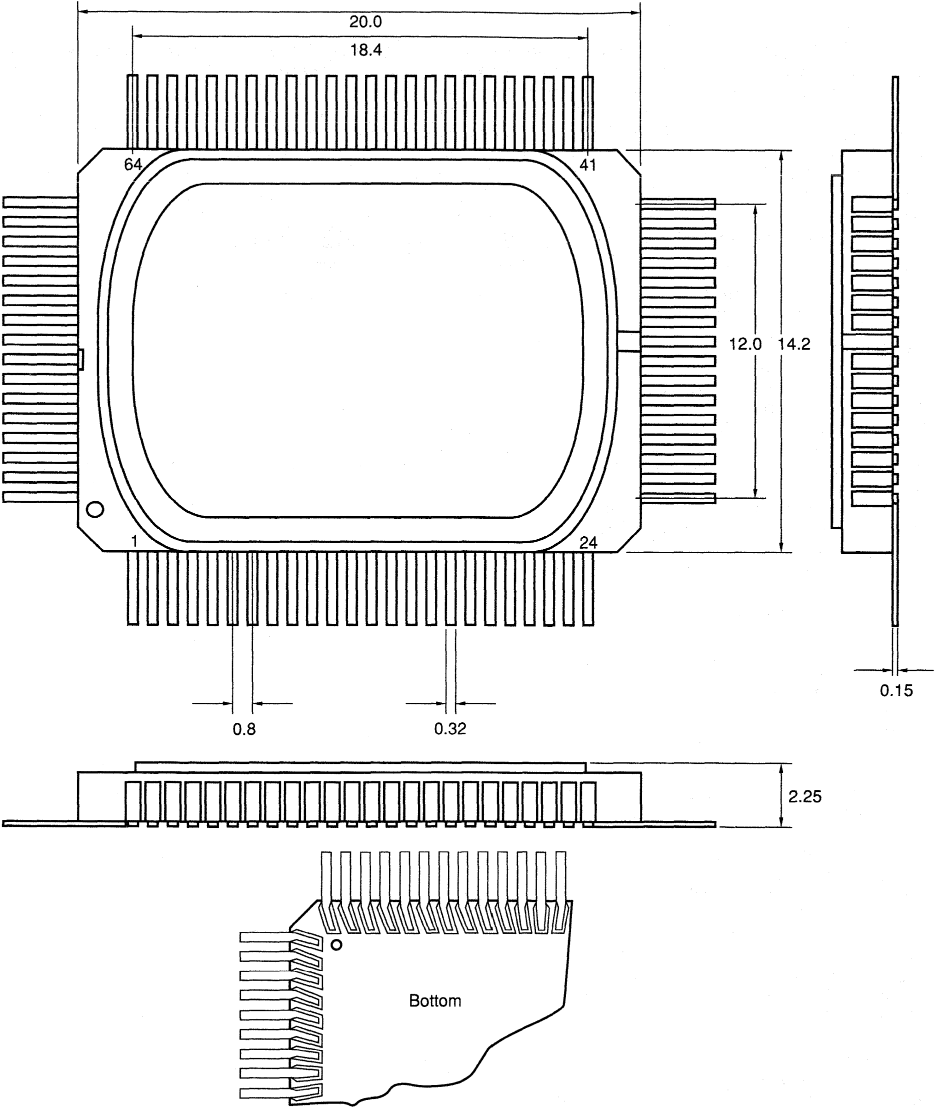


The pin edge configuration



P80GF-80-3B9

ES 80-pin Ceramic QFP (Reference Diagram) (Units in mm)
Top view



Caution:

1. If the metal cup is connected to pin 33, it becomes $V_{SS}$ level.
2. The bottom's leads are diagonally configured.
3. As the lead edges are not performed process control in metal machining, lead length is not specified.

## 10. μPD75P308K (UVPROM) / μPD75P308/P316GF (OTPROM) 4-BIT MICROCOMPUTERS.

### 10.1 Overview

The μPD75P308/16 is a single-chip 4-bit microcomputer which is supplied as a plastic otprom (one time programmable) or ceramic uvprom with window. The single-chip includes, timer/event counters, serial interface and vector interrupt. These functions are incorporated along with CPU, PROM, RAM and I/O ports.
The μPD75P308/16 is pin-compatible with the μPD75308/16 and equivalent to the device in function. Due to the simplicity of the programming requirements, the μPD75P308/16 is suitable for evaluation of the μPD75308/16.
The μPD75P308 will be use for emulation of μPD75304/306/308 and the μPD75P316 for μPD75312/316.
The μPD75P308/16 should not be used for final EMI and Latch up evaluation.

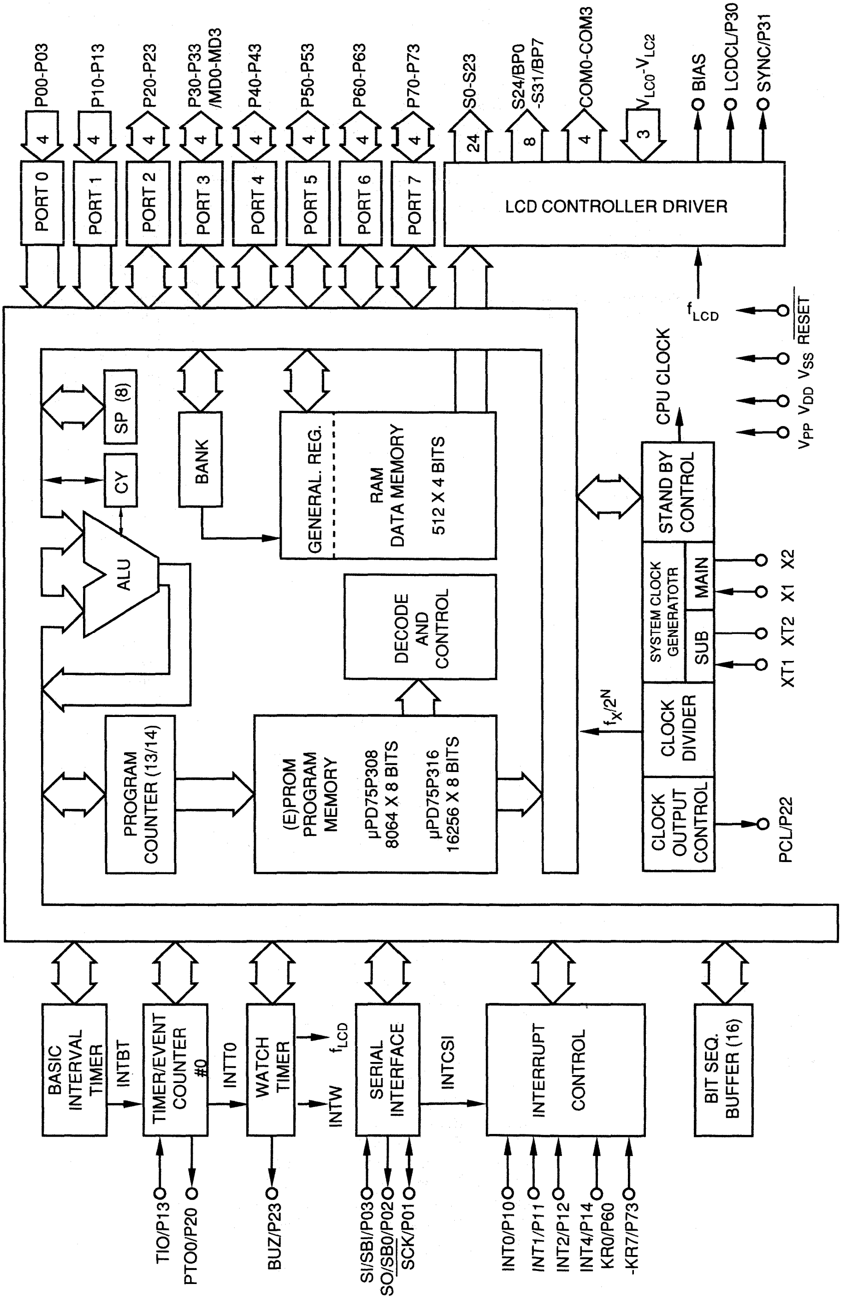
Note:

| Memory type | μPD75P308 | μPD75P316 |
|---|---|---|
| ROM | 8064 x 8 bit | 16256 x 8 bit |
| RAM | 512 x 4 bit | 512 x 4 bit |

### 10.2 Features

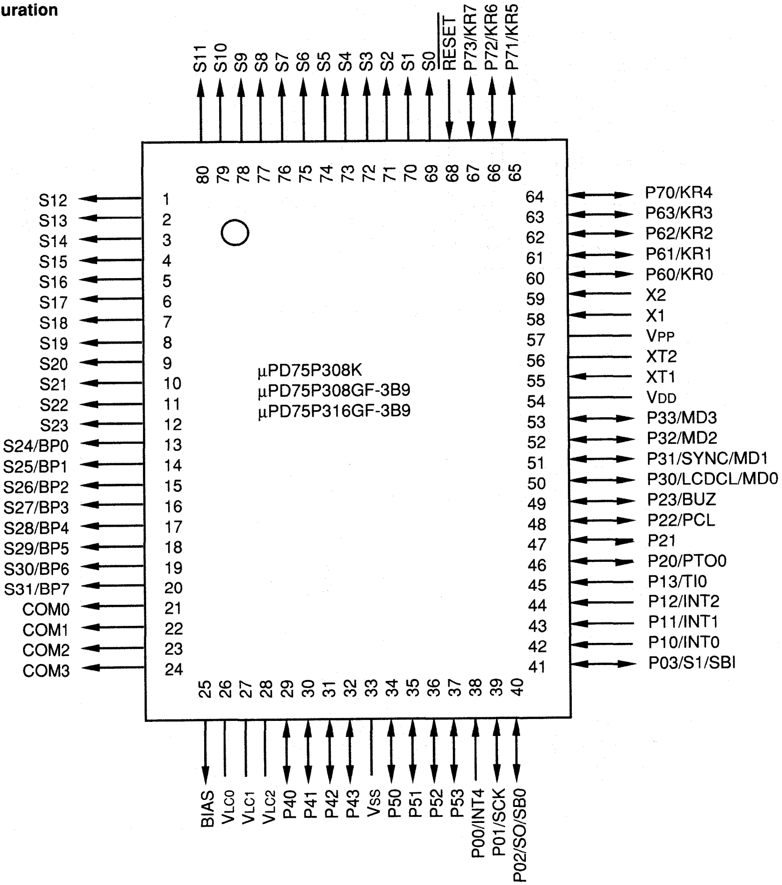
- 42 types of instructions.
  Numerous bit manipulation instructions.
  8-bit transfer instructions.
  1-byte relative branch instructions.
  GETI instruction that realizes arbitrary 2-byte instructions in 1-byte units.
- Programmable instruction cycle: 0.95μs, 1.91μs, 15.3μs / 4.19MHz (main system oscillator), 122μs / 32kHz (sub system oscillator) at 5V
- Size of data memory (RAM): 512 x 4 bit (memory bank configuration)
- Memory-mapped I/O (memory bank configuration)
- Memory for bit manipulation (bit-sequential buffer: 16 bits)
- General-purpose register: 4 bits x 8 (one register bank)
- Accumulators: Bit accumulator (CY)
  4-bit accumulator (A)
  8-bit accumulator (XA)
- LCD controller/driver
  LCD drive segment outputs: 32
  LCD drive common outputs: 4
  LCD drive mode
  — static
  — 1/2 bias – 1/2 duty, 1/3 duty
  — 1/3 bias – 1/3 duty, 1/4 duty
- External expansion port for LCD controller/driver, serial connection.
- 32 I/O lines
  CMOS In/Out Pins
  Middle-high voltage N-ch open drain In/Out ports: 2
  LED direct drive port: 3
- Vector interrupt function capable of multiple interrupts, 3 external vector interrupts
  1 external test input
  3 internal vector interrupts
  1 internal test interrupt
- Basic interval timer
- One 8-bit timer/event counter
- Watch timer (0.5 sec test flag)
- 8 bit serial interface:
  — Conventional 75, 75X compatible mode
  — Serial bus interface (SBI) mode
- Internal clock oscillator with crystal/ceramic resonator
- Subsystem and main system oscillators, run CPU on 32kHz subsystem.
- Standby operation (STOP/HALT)
- CMOS low power consumption
- μPD75P308/16GF: 80 PIN plastic flat pack one time programmable OTPROM
- μPD75P308K: 80 PIN ceramic LCC package UVPROM

### 10.3 Block Diagram



### 10.4 Pin Configuration

Top view



| | | | | |
|---|---|---|---|---|
| P00-03 | : Port 0 | | S0-31 | : Segment output 0-31 |
| P10-13 | : Port 1 | | COM0-3 | : Common output 0-3 |
| P20-23 | : Port 2 | | $V_{LC0-2}$ | : LCD power supply 0-2 |
| P30-33 | : Port 3 | | BIAS | : LCD power supply bias control |
| P40-43 | : Port 4 | | LCDCL | : LCD clock |
| P50-53 | : Port 5 | | SYNC | : LCD synchronization |
| P60-63 | : Port 6 | | TI0 | : Timer input 0 |
| P70-73 | : Port 7 | | PTO0 | : Programmable timer output 0 |
| BP0-7 | : Bit port | | BUZ | : Buzzer clock |
| KR0-7 | : Key return | | PCL | : Programmable clock |
| SCK | : Serial clock | | INT0, 1, 4 | : External vectored interrupt 0, 1, 4 |
| SI | : Serial input | | INT2 | : External test input 2 |
| SO | : Serial output | | X1, 2 | : Main system clock oscillation 1, 2 |
| SB0, I | : Serial bus 0, I | | XT1, 2 | : Subsystem clock oscillation 1, 2 |
| RESET | : Reset input | | $V_{PP}$ | : Program Power Supply |
| MD0-MD3 | : Mode 0-3 | | | |

### 10.5 Pin Functions

10.5.1 Port Pins

| Pin | Input/Output | Serves as Pin for | Functions | 8-Bit I/O | At Reset | I/O Circuit Type (Note 1) |
|---|---|---|---|---|---|---|
| P00 | Input | INT4 | 4-bit input port (PORT0).<br>Incorporation of pull-up resistors in 3-bit units for P01 to P03 specifiable by software. | X | Input | Ⓑ |
| P01 | Input/Output | $\overline{SCK}$ | | | | Ⓕ-A |
| P02 | Input/Output | SO/SB0 | | | | Ⓕ-B |
| P03 | Input/Output | SI/SBI | | | | Ⓜ-C |
| P10 | Input | INT0 | Noise elimination function available<br>4-bit input port (PORT1).<br>Incorporation of pull-up resistors in 4-bit units specifiable by software. | X | Input | Ⓑ-C |
| P11 | | INT1 | | | | |
| P12 | | INT2 | | | | |
| P13 | | TI0 | | | | |
| P20 | Input/Output | PTO0 | 4-bit input/output port (PORT2).<br>Incorporation of pull-up resistors in 4-bit units specifiable by software. | X | Input | E-B |
| P21 | | — | | | | |
| P22 | | PCL | | | | |
| P23 | | BUZ | | | | |
| P30 (Note 2) | Input /Output | LCDCL<br>MD0 | Programmable 4-bit input/output port<br>Input/output setting is possible in bit units. In corporation of pull up resistors in 4 bit units specifiable by software. | X | Input | E-B |
| P31 (Note 2) | | SYNC<br>MD1 | | | | |
| P32 (Note 2) | | MD2 | | | | |
| P33 (Note 2) | | MD3 | | | | |
| P40-P43 (Note 2) | Input/Output | — | N-channel open drain 4-bit input/output port (PORT4).<br>Data input/output pin (lower 4-bits) for use during program memory (PROM) write/ verify operations. | O | High Impedance | M-B |
| P50-P53 (Note 2) | Input/Output | — | N-channel open drain 4-bit input/output port (PORT5).<br>Data input/output pin (upper 4-bits) for use during program memory (PROM) write/ verify operations. | | High Impedance | M-B |
| P60 | Input/Output | KR0 | Programmable 4-bit input/output port (PORT 6).<br>This port can be specified for input/output in bit units.<br>Incorporates software-specifiable pull-up resistors in 4-bit units. | O | Input | Ⓕ-A |
| P61 | | KR1 | | | | |
| P62 | | KR2 | | | | |
| P63 | | KR3 | | | | |
| P70 | Input/Output | KR4 | 4-bit input/output port (PORT7)<br>Incorporation of pull-up resistors in 4-bit units specifiable by software. | | Input | Ⓕ-A |
| P71 | | KR5 | | | | |
| P72 | | KR6 | | | | |
| P73 | | KR7 | | | | |

10.5.1 Port Pins (cont'd)

| Pin | Input/Output | Serves as Pin for | Functions | 8-Bit I/O | At Reset | I/O Circuit Type (Note 1) |
|---|---|---|---|---|---|---|
| BP0 | Output | S24 | 1-bit output port (BIT PORT).<br>Also serves as the segment output pins. | X | (Note 3) | G-A |
| BP1 | | S25 | | | | |
| BP2 | | S26 | | | | |
| BP3 | | S27 | | | | |
| BP4 | Output | S28 | | | | |
| BP5 | | S29 | | | | |
| BP6 | | S30 | | | | |
| BP7 | | S31 | | | | |

Note 1: Circles indicate Schmitt trigger inputs.
Note 2: LED direct drive is possible.
Note 3: BP0-7 select $V_{LC1}$ as the input source. The output level changed by the external circuit of BP0-7 and $V_{LC1}$.

10.5.2 Non-Port Pins

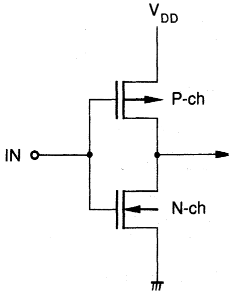
| Pin | Input/Output | Serves as Pin for | Functions | At Reset | I/O Circuit Type (Note 1) |
|---|---|---|---|---|---|
| TI0 | Input | P13 | External event pulse input pin for timer/event counter. | | Ⓑ-C |
| PTO0 | Input/Output | P20 | Timer/event counter output pin. | Input | E-B |
| PCL | Input/Output | P22 | Clock output pin. | Input | E-B |
| BUZ | Input/Output | P23 | Fixed-frequency output pin (for buzzer or system clock trimming). | Input | E-B |
| $\overline{SCK}$ | Input/Output | P01 | Serial clock input/output pin. | Input | Ⓕ-A |
| SO/SB0 | Input/Output | P02 | Serial data output pin. Serial bus input/output pin. | Input | Ⓕ-B |
| SI/SBI | Input/Output | P03 | Serial data input pin. Serial bus input/output pin. | Input | Ⓜ-C |
| INT4 | Input | P00 | Edge detector vector interrupt input pin (either rising or falling edge detection). | | Ⓑ |
| INT0 | Input | P10 | Edge detector interrupt input pin (detected edge selectable). | | Ⓑ-C |
| INT1 | | P11 | | | |
| INT2 | Input | P12 | Edge detection testable input pin (rising edge detection). | | Ⓑ-C |
| KR0-KR3 | Input/Output | P60-P63 | Testable input/output pin (for falling edge detection). | Input | Ⓕ-A |
| KR4-KR7 | Input/Output | P70-P73 | Testable input/output pin (for falling edge detection). | Input | Ⓕ-A |
| S0-S23 | Output | | Segment signal output pin. | (Note 3) | G -A |
| S24-S31 | Output | BP0-7 | Segment signal output pin. | (Note 3) | G -A |
| COM0 -COM3 | Output | | Common signal output pin. | (Note 3) | G -B |
| $V_{LC0}$-$V_{LC2}$ | | | LCD drive power supply. | | |

10.5.2 Non-Port Pins

| Pin | Input/Output | Serves as Pin for | Functions | At Reset | I/O Circuit Type (Note 1) |
|---|---|---|---|---|---|
| BIAS | | | Output pin for divided resistor cut externally installed. | Hi-Z | |
| LCDCL (Note 2) | Input/Output | P30 | Clock output pin for activating the externally extended driver. | Input | E-B |
| SYNC (Note 2) | Input/Output | P31 | Clock output pin for synchronizing the externally extended driver. | Input | E-B |
| X1, X2 | Input | | Pin for connection of crystal/ceramics for main system clock generation. External clocks are input to X1, and their negative phase components are input to X2. | | |
| XT1 | Input | | Pin for connection of crystals for sub-system clock generation. External clocks are input to XT1, and XT2 is disconnected. XT1 can also be used as a 1-bit input (test) pin. | | |
| XT2 | | | | | |
| $\overline{\text{RESET}}$ | Input | | System reset input pin (low level active). | | Ⓑ |
| MD0-MD3 | Input/Output | P30-P33 | Mode selector pin for program memory (EPROM) write/verify operations. | Input | E-B |
| $V_{PP}$ | | | Program voltage application pin for program memory (EPROM) write/verify operations. $V_{PP}$ is normally connected to $V_{DD}$. Connect to +12.5V for EPROM write/verify operations. | | |
| $V_{DD}$ | | | Positive power supply pin. | | |
| $V_{SS}$ | | | GND potential pin. | | |

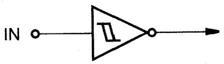
Note 1: Circles indicate Schmitt trigger inputs.
Note 2: Pins for future use in system extention. Pins LCDCL and SYNC currently serve only as P30 and P31, respectively.
Note 3: For each display output, the following $V_{LCX}$ are selected as input sources:
S0-S31: $V_{LC1}$, COM0-COM2: $V_{LC2}$, COM3: $V_{LC0}$
However, the display output level varies depending on the display output and $V_{LCX}$ external circuit.
Internal LCD split resistors are not included!

Pin I/O configurations
Following figures show the internal circuit configurations at the I/O ports.
(1) Type A (part of Type E–B)



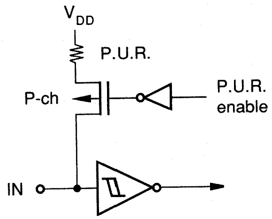
This is a CMOS standard input buffer.

(2) Type B



This is a Schmitt trigger input with hysteresis characteristics.

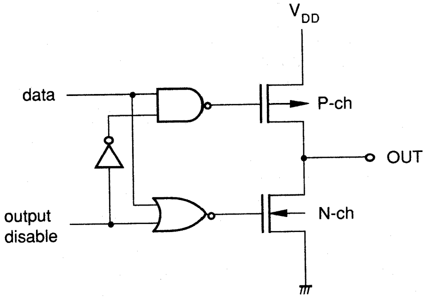
(3) Type B–C



P.U.R. : Pull-Up Resistor

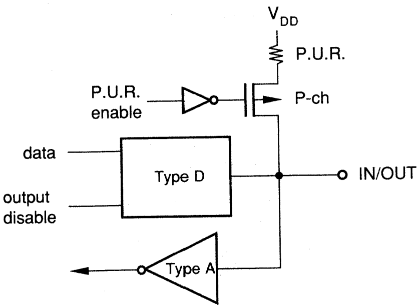
This is a Schmitt trigger input with hysteresis characteristics.

(4) Type D (Part of Type E–B, F–A)



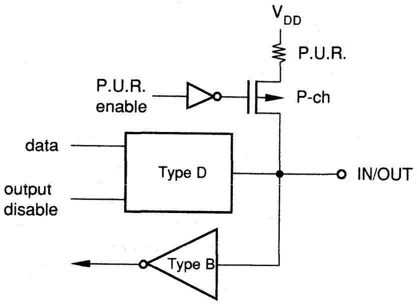
This is a push-pull output that can be set to high impedance (with both P-ch and N-ch off).

(5) Type E–B



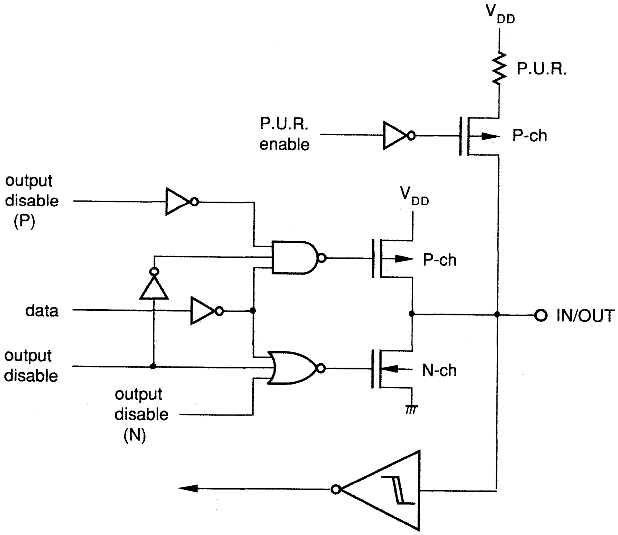
P.U.R. : Pull-Up Resistor

(6) Type F–A



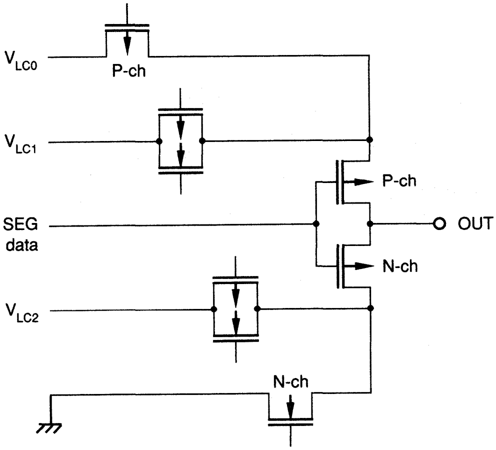
P.U.R. : Pull-Up Resistor

(7) Type F–B

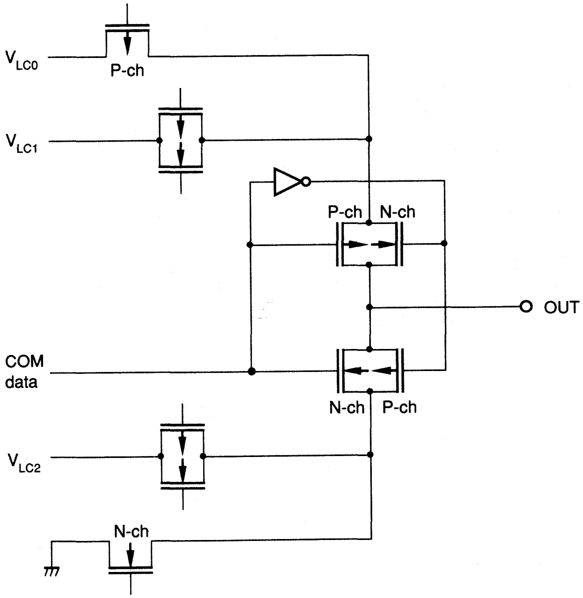


P.U.R. : Pull-Up Resistor

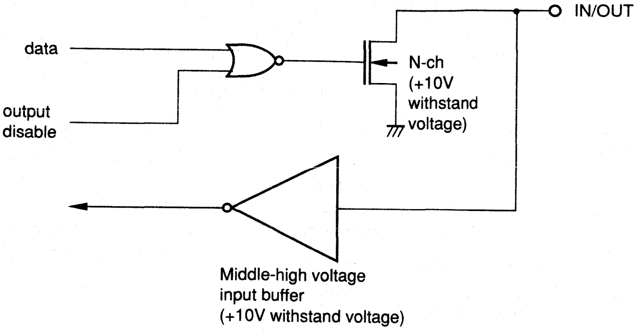
(8) Type G–A



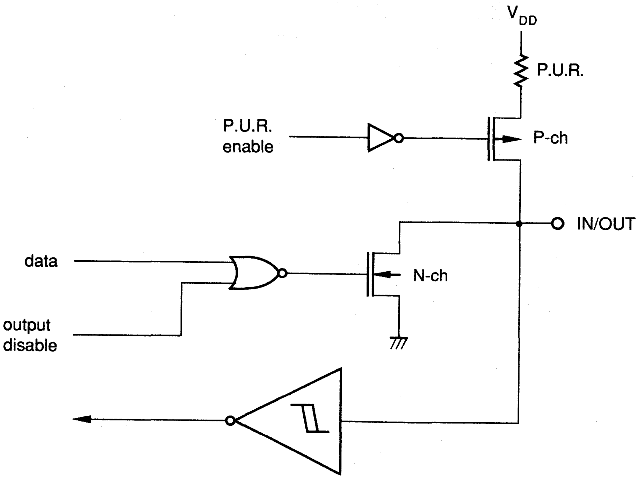
(9) Type G–B



(10) Type M–B



(11) Type M–C



P.U.R. : Pull-Up resistor

### 10.6 DIFFERENCES BETWEEN μPD75P308/P316 AND μPD753XX

Because the μPD75P308/P316 is a product that incorporates an on-chip, writeable PROM in place of the masked ROM in the μPD753XX program memory, the μPD75P308/P316 and the μPD753XX have similar hardware and perform similar CPU functions.

They differ simply in their type of program memory and mask options. Tables 10.6-1 shows these differences.

Refer to the μPD753XX Users´ Manual for details of hardware and CPU functions.

**Table 10.6-1 Differences between μPD75P308 and μPD75308**

| Item | | μPD75P308K | μPD75P308GF | μPD75308GF |
|---|---|---|---|---|
| Program memory | | • EPROM<br>• 0000–1F7FH<br>• 8064 x 8 bits | • PROM (one-time programable version)<br>• 0000–1F7FH<br>• 8064 x 8 bits | • Masked ROM<br>• 0000–1F7FH<br>• 8064 x 8 bits |
| Pull-up resistor | Ports 4,5 | None | | Mask option |
| Divided resistor for LCD drive power supply | | None | | Mask option |
| Pin configuration | | $V_{PP}$ pin and PROM program pins available. | | $V_{PP}$ pin and PROM program pins not available. |
| Power voltage range | | 5 V ± 5 % | | 2.7 to 6.0 V |
| Package | | 80-pin ceramic LCC package (with window). | 80-pin plastic QFP. | |

**Table 10.6-2 Differences between μPD75P316 and μPD75316**

| Item | | μPD75P316GF | μPD75316GF |
|---|---|---|---|
| Program memory | | • PROM (one-time programable version)<br>• 0000–1F7FH<br>• 16256 x 8 bits | • Masked ROM<br>• 0000–1F7FH<br>• 16256 x 8 bits |
| Pull-up resistor | Ports 4,5 | None | Mask option |
| Divided resistor for LCD drive power supply | | None | Mask option |
| Pin configuration | | $V_{PP}$ pin and PROM program pins available. | $V_{PP}$ pin and PROM program pins not available. |
| Power voltage range | | 5 V ± 5 % | 2.7 to 6.0 V |
| Package | | 80-pin plastic QFP. | |

### 10.7 PROM WRITE AND VERIFICATION

Program memory contained in the μPD75P308, μPD75P316 is PROM.
The memory capacity is as follows:

- μPD75P308: 8064 words x 8 bits
- μPD75P316: 16256 words x 8 bits

The pins as listed in Table 10.7-1 are used to write and verify the PROM. No address is input; address is updated by inputting clock from the X1 pin instead.

**Table 10.7-1 Pin function**

| Pin name | Function |
|---|---|
| X1, X2 | Address update clock during when PROM is write/verification is input to the X1 pin.<br>Its inverted signal is input to the X2 pin. |
| MD0-MD3 | Operation mode selection pins during PROM write/verification. |
| P40-P43 (low-order four bits)<br>P50-P53 (high-order four bits) | 8-bit data input/output pins during PROM write/verified. |
| $V_{DD}$ | Supply voltage apply pin.<br>During the normal operation, 5 V ± 5 % is applied; during PROM write/verification, +6 V is applied. |
| $V_{PP}$ | Voltage apply pin during PROM write/ verification. (Normally, $V_{DD}$ potential) |

Caution 1: Put a shading cover film on the µPD75P308K having a window, except when the PROM contents are erased.
Caution 2: µPD75P308GF, µPD75P316GF of the one-time version does not have a window, and the PROM contents cannot be erased with ultraviolet rays.

10.7.1 Operating Mode during PROM is Write/Verification
When +6 V and +12.5 V are applied to the $V_{DD}$ and $V_{PP}$ pins, respctively, of the µPD75P308, µPD75P316 the PROM write/verify mode is entered. The operation mode is selected according to the input signals to the MD0-MD3 pins as listed in Table 10.7-2. Pins not used in EPROM function should be pulled to $V_{SS}$.

**Table 10.7-2 Operating Mode**

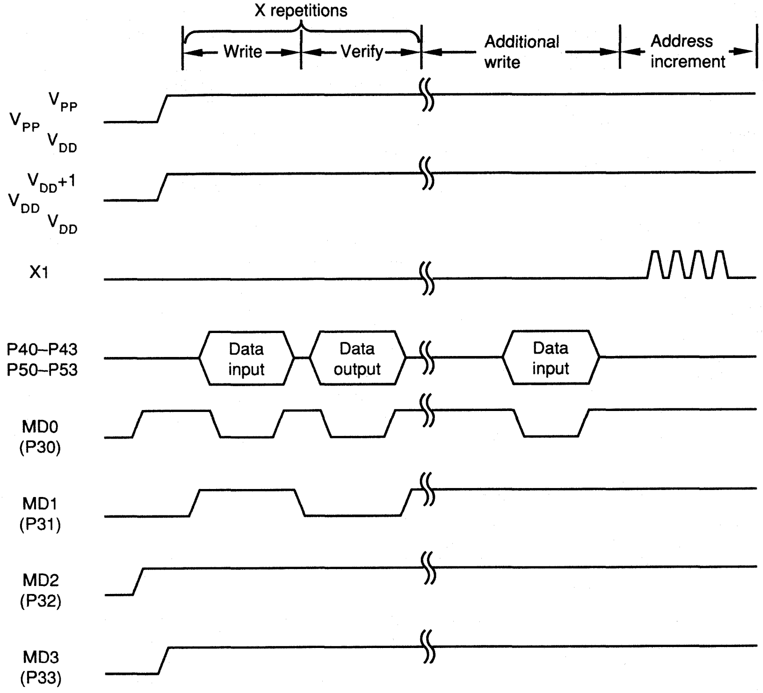
| Operating mode specification | | | | | | Operating mode |
|---|---|---|---|---|---|---|
| $V_{DD}$ | $V_{PP}$ | MD0 | MD1 | MD2 | MD3 | |
| +12.5 V | +6 V | H | L | H | L | Clear program memory address |
| | | L | H | H | H | Write mode |
| | | L | L | H | H | Verify mode |
| | | H | X | H | H | Program inhibit mode |

Remarks: X: L or H

10.7.2 PROM Write Procedure
PROM can be written at the high speed according to the following Procedure:

(1) Pull unused pins low to $V_{SS}$ with resistors. Set the X1 pin low.
(2) Supply 5 V to the $V_{DD}$ and $V_{PP}$ pins.
(3) Wait for 10 us.
(4) Selected the clear program memory address mode.
(5) Supply +6 V and +12.5 V to the $V_{DD}$ and $V_{PP}$ pins.
(6) Select the program inhibit mode.
(7) Write data in the 1-ms write mode.
(8) Select the program inhibit mode.
(9) Select the verify mode. If the data is written normally, proceed to (10). If it is not written normally, repeat (7) to (9).
(10) Supply X (number of (7)–(9) repetitions) x 1 ms program pulses (additional write).
(11) Select the program inhibit mode.
(12) Input four pulses to the X1 pin to update the program memory address by one.
(13) Repeat (7)–(12) until the end address is reached.
(14) Select the clear program memory address mode.
(15) Change the $V_{DD}$, $V_{PP}$ pin voltage to 5 V.
(16) Turn off the power.

Steps (2) to (12) are illustrated below:

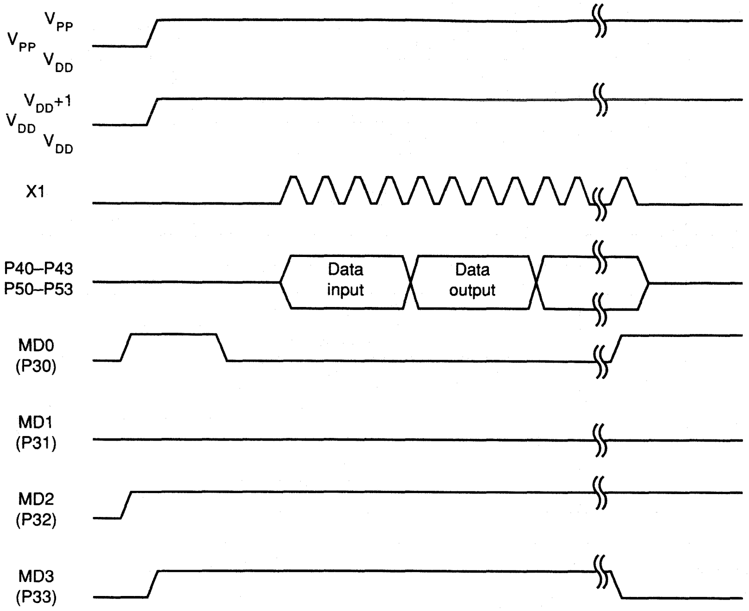


10.7.3 PROM Read Procdure

The PROM contents can be read according to the following procedure:

(1) Pull unused pins low to $V_{SS}$ with resistors.
Set the X1 pin low.
(2) Supply 5 V to the $V_{DD}$ and $V_{PP}$ pins.
(3) Wait for 10 µs.
(4) Select the clear program memory address mode.
(5) Supply + 6 V and + 12.5 V to the $V_{DD}$ and $V_{PP}$ pins.
(6) Select the program inhibit mode.
(7) Select the verify mode. Input four clock pulses to the X1 pin. The data are output from the memory addresses, one address at a time, in a cycle of hour clock pulses which are input to the X1 pin.
(8) Select the program inhibit mode.
(9) Select the clear program memory address mode.
(10) Change the $V_{DD}$, $V_{PP}$ voltage to 5 V.
(11) Turn off the power.

Steps (2) to (12) are illustrated below:



10.7.4 Erasion Method

The programmed data contents of the μPD75P308K can be erased by applying light whose wave length is shorter than about 400 nm. The programmed data contents may also be erased when the μPD75P308K is subjected to direct sunlight or fluorescent light for hours. To project the data contents, mask the μPD75P308K with a shading cover film to prevent ultraviolet rays from being entered through the upper window. Shading cover film whose quality has been tested is attached to UV PROM product for shipment by NEC.

For normal erasion, apply 254-nm ultraviolet rays to the μPD75P308K. All the radiation amount required to erase the μPD75P308K contents completely is 15 Ws/cm$^2$ (ultraviolet lenght x erasion time) at the minmum. This corresponds to about 15–20 minutes when a 12000 Ws/cm$^2$ ultraviolet ray lamp is used for erasion. However, note that the erasion time may be prolonged due to the ultraviolet lamp life, dirty package window, etc. Place the μPD75P308K and ultraviolet ray lamp within 2.5–cm distance.

### 10.8 Electrical charactaristics

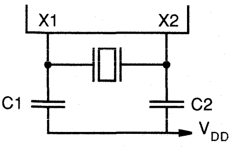
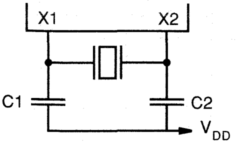
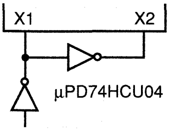
**Absolute Maximum Rating** (Ta=25°C)

| Prameter | Symbol | Condition | | Rating | Unit |
|---|---|---|---|---|---|
| Supply voltage | $V_{DD}$ | | | -0.3 to +7.0 | V |
| | $V_{PP}$ | | | -0.3 to +13.5 | V |
| Input voltage | $V_{I1}$ | Other than ports 4 and 5 | | -0.3 to $V_{DD}$ +0.3 | V |
| | (Note 1)<br>$V_{I2}$ | Ports 4 and 5 | Opendrain | -0.3 to +11 | V |
| Output voltage | $V_0$ | | | -0.3 to $V_{DD}$ +0.3 | V |
| High-level output current | $I_{OH}$ | 1 pins | | -15 | mA |
| | | All other pins | | -30 | mA |
| Low-level output current | (Note 2)<br>$I_{OL}$ | 1 pins | Peak value | 30 | mA |
| | | | Effective value | 15 | mA |
| | | Total of ports 0, 2, 3 and 5 | Peak value | 100 | mA |
| | | | Effective value | 60 | mA |
| | | Total of ports 4, 6 and 7 | Peak value | 100 | mA |
| | | | Effective value | 60 | mA |
| Operating temperature | $T_{opt}$ | | | -10 to +70 | °C |
| Storage temperature | $T_{stg}$ | | | -65 to +150 | °C |

Note 1: When applying more than 10 volts to ports 4 or 5, the pull-up resistor must be 50K ohms min.

2: Effective value is obtained as follows:
(Effective) value = (Peak value) x $\sqrt{\text{duty cycle}}$

**Main system clock Oscillator Characteristics**
(Ta = -10 to +70°C, $V_{DD}$ = 5V+/-5%)

| Resonator | Recommended constants | Item | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Note 3<br>Ceramic resonator |  | Note 1<br>Oscillation frequency ($f_{XX}$) | | 1.0 | | 5.0 | MHz |
| | | Note 2<br>Oscillation stabilization time | When $V_{DD}$ reaches the minimum oscillator operating voltage. | | | 4 | ms |
| Crystal resonator |  | Note 1<br>Oscillation frequency ($f_{XX}$) | | 1.0 | 4.19 | 5.0 | MHz |
| | | Note 2<br>Oscillation stabilization time | | | | 10 | ms |
| External clock |  | Note 1<br>X1 input frequency ($f_X$) | | 1.0 | | 5.0 | MHz |
| | | X1 input high/ low level width ($t_{XH}$ / $t_{XL}$) | | 100 | | 500 | ns |

Note 1: The oscillation frequency and X1 input frequency values indicated in this table express only the characteristics of the oscillator. Therefore, refer to the AC characteristics for the instruction execution time.

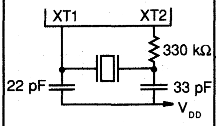
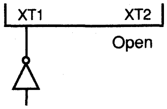
2: Oscillation stabilization time is defined as the time needed for the oscillator to stabilize after $V_{DD}$ is applied or after the STOP mode is released.

3: The following ceramic and crystal oscillators are recommended for the μPD75P308/316.

**Recommended Ceramic Resonators**

| Manufacture | Product | External capacitance (pF) | | Operating voltage range (V) | |
|---|---|---|---|---|---|
| | | C1 | C2 | MIN. | MAX. |
| MURATA | CSA 2.00MG | 30 | 30 | 4.75 | 5.25 |
| | CSA 4.19MG | 30 | 30 | 4.75 | 5.25 |
| | CSA 4.19MGU | 30 | 30 | 4.75 | 5.25 |
| | CST 4.19MG | 30pF (internally provided) | 30pF (internally provided) | 4.75 | 5.25 |

### Subsystem Clock Oscillator Characteristics
(Ta =-10 to +70°C, $V_{DD}$ =5V+/-5%)

| Resonator | Recommended constants | Item | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Crystal resonator |  | Oscillation frequency ($f_{XT}$) | | 32 | 32.768 | 35 | kHz |
| | | Oscillation stabilization time | $V_{DD}$=4.5~6.0V | | 1.0 | 2 | s |
| External clock |  | XT1 input frequency ($f_{XT}$) | | 32 | | 100 | kHz |
| | | XT1 input high/ low level width ($t_{XTH}$/$t_{XTL}$) | | 5 | | 15 | μs |

**Capacitance** (Ta =25°C, $V_{DD}$ =0V)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Input capacitance | $C_{IN}$ | f=1 MHz<br>Voltage should not be applied to any pin except those used for measurements. | | | 15 | pF |
| Output capacitance | $C_{OUT}$ | | | | 15 | pF |
| Input/output capacitance | $C_{IO}$ | | | | 15 | pF |

**DC Characteristics** (Ta =-10 to +70°C, $V_{DD}$ =5V±5%)

| Parameter | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| High-level input voltage | $V_{IH1}$ | Port 2, 3 | | $0.7V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH2}$ | Port 0, 1, 6, 7, $\overline{\text{RESET}}$ | | $0.8V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH3}$ | Port 4, 5 Open-drain | | $0.7V_{DD}$ | | 10 | V |
| | $V_{IH4}$ | X1, X2, XT1 | | $V_{DD}$-0.5 | | $V_{DD}$ | V |
| Low-level input voltage | $V_{IL1}$ | Port 2, 3, 4, 5 | | 0 | | $0.3V_{DD}$ | V |
| | $V_{IL1}$ | Port 0, 1, 6, 7, $\overline{\text{RESET}}$ | | 0 | | $0.2V_{DD}$ | V |
| | $V_{IL3}$ | X1, X2, XT1 | | 0 | | 0.4 | V |
| High-level output voltage | $V_{OH1}$ | Port 0, 2, 3, 6, 7, BIAS | $I_{OH}$ =-1mA | $V_{DD}$-1.0 | | | V |
| | $V_{OH2}$ | BPO-7 | $I_{OH}$ =-100μA<br>Note 4 | $V_{DD}$-2.0 | | | V |
| Low-level output voltage | $V_{OL1}$ | Port 0, 2, 3, 4, 5, 6, 7 | Port 3, 4, 5 $I_{OL}$ =15mA | | 0.4 | 2.0 | V |
| | | | $I_{OL}$ =1.6mA | | | 0.4 | V |
| | $V_{OL2}$ | SB0, 1 open-drain | Pull-up resistor (R) =1k ohm min. | | | $0.2V_{DD}$ | V |
| | $V_{OL3}$ | BP0-7 | $I_{OL}$ =100μA<br>Note 4 | | | 1.0 | V |
| High-level input leakage current | $I_{LIH1}$ | $V_{IN}$ =$V_{DD}$ | Other than X1, X2, and XT1 pins | | | 3 | μA |
| | $I_{LIH2}$ | | X1, X2, XT1 | | | 20 | μA |
| | $I_{LIH3}$ | $V_{IN}$ =10V | Port 4, 5 | | | 20 | μA |

**DC Characteristics** (Ta =-10 to +70°C, $V_{DD}$ =5V±5%)

| Parameter | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Low-level input leakage current | $I_{LIL1}$ | $V_{IN}$ =0V | Other than X1, X2, and XT1 pins | | | -3 | μA |
| | $I_{LIL2}$ | | X1, X2, XT1 | | | -20 | μA |
| High-level output leakage current | $I_{LOH1}$ | $V_{OUT}$ =$V_{DD}$ | Other than ports 4 and 5 | | | 3 | μA |
| | $I_{LOH2}$ | $V_{OUT}$ =10V | Port 4,5 | | | 20 | μA |
| Low-level output leakage current | $I_{LOL}$ | $V_{OUT}$ =0V | | | | -3 | μA |
| Internal pull-up resistor | $R_{LI}$ | Port 0, 1, 2, 3, 6, 7 (except P00) $V_{IN}$ =0V | | 15 | 40 | 80 | kΩ |
| LCD drive voltage | $V_{LCD}$ | | | 2.5 | | $V_{DD}$ | V |
| Common output impedance | $R_{COM}$ | Note 3 | | | 3 | 6 | kΩ |
| Segment output impedance | $R_S$ | Note 3 | | | 15 | 20 | kΩ |
| Note 1 Supply current | $I_{DD1}$ | Note 5 4.19 MHz crystal oscillation C1=C2=22pF | Note 2 | | 5 | 15 | mA |
| | $I_{DD2}$ | | HALT mode | | 500 | 1500 | μA |
| | $I_{DD3}$ | Note 6 32kHz crystal oscillation | | | 350 | 1000 | μA |
| | | | HALT mode | | 35 | 100 | |
| | $I_{DD4}$ | XT1 =0V STOP mode | | | 0.5 | 20 | μA |

Note 1. The current drained through the internal pull-up resistors is not included.

2. When operated in the high-speed mode with the processor clock control register (PCC) set to 0011.
3. $2.5V \leq V_{LCD} \leq V_{DD}$
4. When any two pins of BP0 to BP3 and any two pins of BP4 to BP7 are used simultaneously for output.
5. Including the subsystem clock power consumption.
6. When system clock control register (SCC) is set to 1001; main system clock`s oscillation halted once sub-system clock in operation.

3

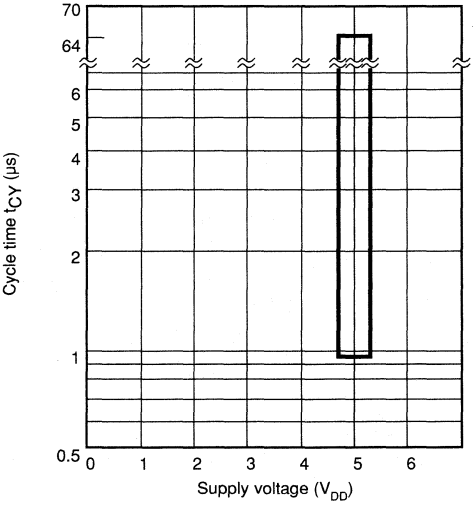
**AC Characteristics** (Ta =-10 to +70°C, $V_{DD}$ =5V±5%)
Operation other than serial transfer operation

| Parameter | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Note 1<br>Cycle time (minimum instruction execution time) | $t_{CY}$ | Operated with main system clock | 0.95 | | 64 | μs |
| | | Operated with sub-system clock | 114 | 122 | 125 | μs |
| TI0 input frequency | $f_{TI}$ | | 0 | | 1 | MHz |
| TI0 input high/low level width | $t_{TIH}$, $t_{TIL}$ | | 0.48 | | | μs |
| Interrupt input high/low level width | $t_{INTH}$, $t_{INTL}$ | INT0 | Note 2 | | | μs |
| | | KR0-7, INT1 2, 4 | 10 | | | μs |
| $\overline{\text{RESET}}$ low level width | $t_{RSL}$ | | 10 | | | μs |

Note 1. Cycle time (minimum instruction execution time) is determined by the oscillation frequency of the connected resonator, the system clock control register (SCC), and the processor clock control register (PCC). The figure to the right shows the $V_{DD}$ supply voltage vs. cycle time ($t_{CY}$) characteristic when operated with the main system clock.

2. Either $2t_{CY}$ or $64/f_{XX}$ can be selected by setting the interrupt mode register (IM0).

$t_{CY}$ vs $V_{DD}$
(With main system clock)



**Serial Transfer Operation**
**2-line/3-line serial I / 0 mode (SCK....Internal clock output)**

| Parameter | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY1}$ | | Output | 1600 | | | ns |
| $\overline{\text{SCK}}$ high / low level width | $t_{KL1}$<br>$t_{KH1}$ | | Output | $t_{KCL1}/2$ -50 | | | ns |
| SI set-up time (against $\overline{\text{SCK}}$↑ ) | $t_{SIK1}$ | | | 150 | | | ns |
| SI hold time (against $\overline{\text{SCK}}$↑) | $t_{KSI1}$ | | | 400 | | | ns |
| $\overline{\text{SCK}}$↓→ S0 output delay time | $t_{KSO1}$ | | | | | 250 | ns |

**2-line / 3-line serial I/0 mode (SCK....External clock input) :**

| Parameter | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY2}$ | | Input | 800 | | | ns |
| $\overline{\text{SCK}}$ high / low level width | $t_{KL2}$<br>$t_{KH1}$ | | Input | 400 | | | ns |
| SI set-up time (against $\overline{\text{SCK}}$↑ ) | $t_{SIK2}$ | | | 100 | | | ns |
| SI hold time (against $\overline{\text{SCK}}$ ↑ ) | $t_{KSI2}$ | | | 400 | | | ns |
| $\overline{\text{SCK}}$↓ → S0 output delay time | $t_{KSO2}$ | | | | | 300 | ns |

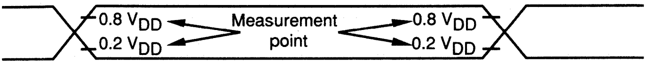
**SBI Mode (SCK....Internal clock output Master)) :**

| Parameter | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY3}$ | | 1600 | | | ns |
| $\overline{\text{SCK}}$ high / low level width | $t_{KL3}$<br>$t_{KH3}$ | | $t_{KCY}/2$<br>-50 | | | ns |
| SB0, SB1 set-up time (against $\overline{\text{SCK}}$↑) | $t_{SIK3}$ | | 150 | | | ns |
| SB0, SB1 hold time (against $\overline{\text{SCK}}$↑) | $t_{KSI3}$ | | $t_{KCY}/2$ | | | ns |
| $\overline{\text{SCK}}$↓ → SB0, SB1 output delay time | $t_{KSO3}$ | | 0 | | 250 | ns |
| $\overline{\text{SCK}}$↓ → SB0, SB1 ↓ | $t_{KSB}$ | | $t_{KCY}$ | | | ns |
| SB0, SB1↓ → $\overline{\text{SCK}}$↓ | $t_{SBK}$ | | $t_{KCY}$ | | | ns |
| SB0, SB1 low level width | $t_{SBL}$ | | $t_{KCY}$ | | | ns |
| SB0, SB1 high level width | $t_{SBH}$ | | $t_{KCY}$ | | | ns |

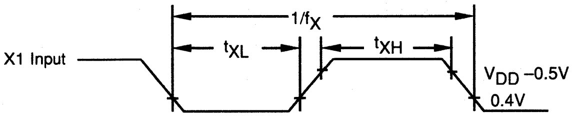
**SBI Mode (SCK.... External clock input (Slave)):**

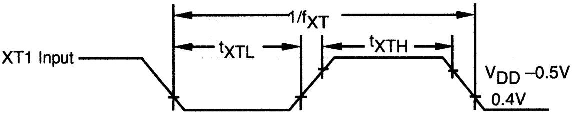
| Parameter | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| SCK cycle time | $t_{KCY4}$ | | 800 | | | ns |
| SCK high / low level width | $t_{KL4}$<br>$t_{KH4}$ | | 400 | | | ns |
| SB0, SB1 set-up time (against $\overline{\text{SCK}}$↑ ) | $t_{SIK4}$ | | 100 | | | ns |
| SB0, SB1 hold time (against $\overline{\text{SCK}}$↑ ) | $t_{KSI4}$ | | $t_{KCY}/2$ | | | ns |
| SCK ↓ → SB0, SB1 output delay time | $t_{KSO4}$ | | 0 | | 300 | ns |
| $\overline{\text{SCK}}$↑ → SB0, SB1 ↓ | $t_{KSB}$ | | $t_{KCY}$ | | | ns |
| SB0, SB1↓ → $\overline{\text{SCK}}$ ↓ | $t_{SBK}$ | | $t_{KCY}$ | | | ns |
| SB0, SB1 low level width | $t_{SBL}$ | | $t_{KCY}$ | | | ns |
| SB0, SB1 high level width | $t_{SBH}$ | | $t_{KCY}$ | | | ns |

AC Timing Measurement Points (Except X1, XT1 input)

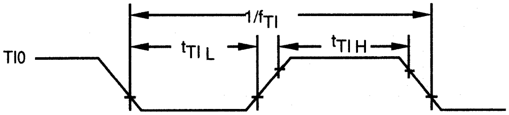


Clock Timing

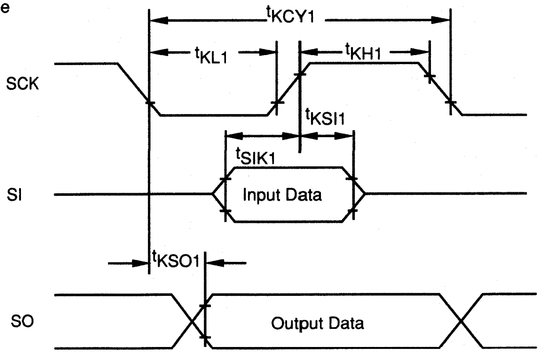




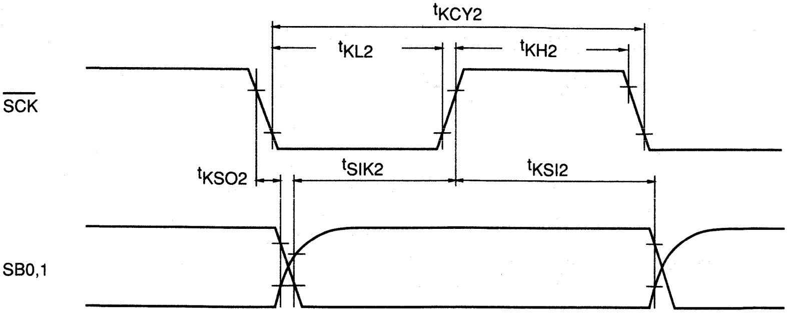
TIO Timing



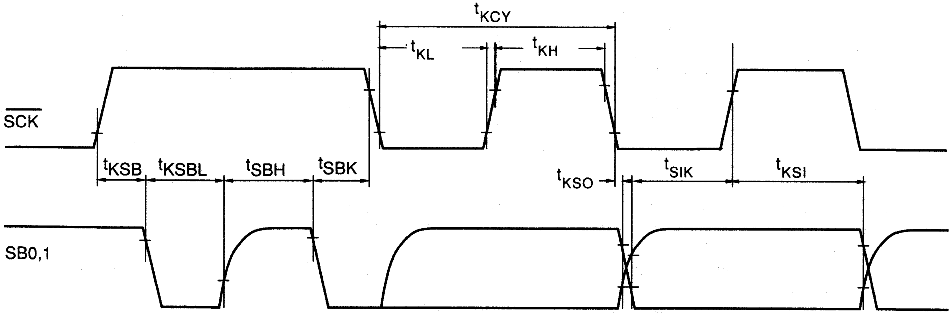
Serial Transfer Timing
3-line serial I/O mode



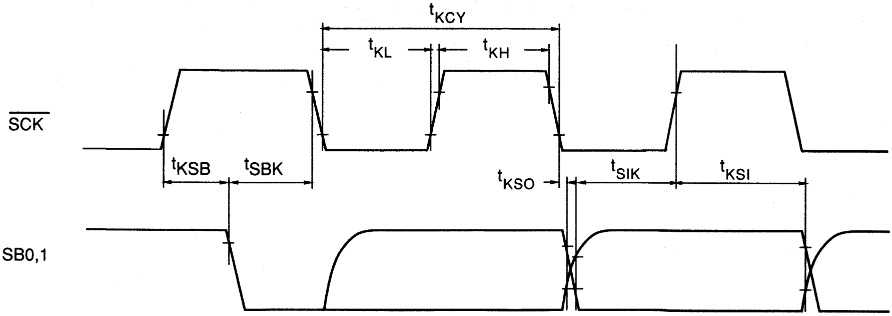
2-line serial I/O mode



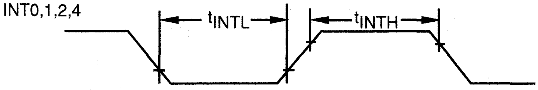
**Serial Transfer Timing** (SBI mode)
Bus release signal transfer:



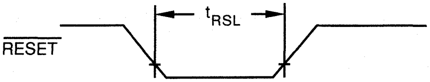
Command signal transfer:



Interrupt input timing



$\overline{\text{RESET}}$ input timing



**Data Memory STOP Mode Low Supply Voltage Data Retention Characteristic** (Ta = –10 to +70 °C)

| Parameter | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Data retention supply voltage | $V_{DDDR}$ | | 2.0 | | 6.0 | V |
| Data retention supply current (Note1) | $I_{DDDR}$ | $V_{DDDR}$ = 2.0V | | 0.1 | 10 | μA |
| Release signal set time | $t_{SREL}$ | | 0 | | | μs |
| Oscillation stabilization wait time (Note 2) | $t_{WAIT}$ | Release by $\overline{\text{RESET}}$ input | | $2^{17}/f_X$ | | ms |
| | | Release by interrupt request | | Note 3 | | ms |

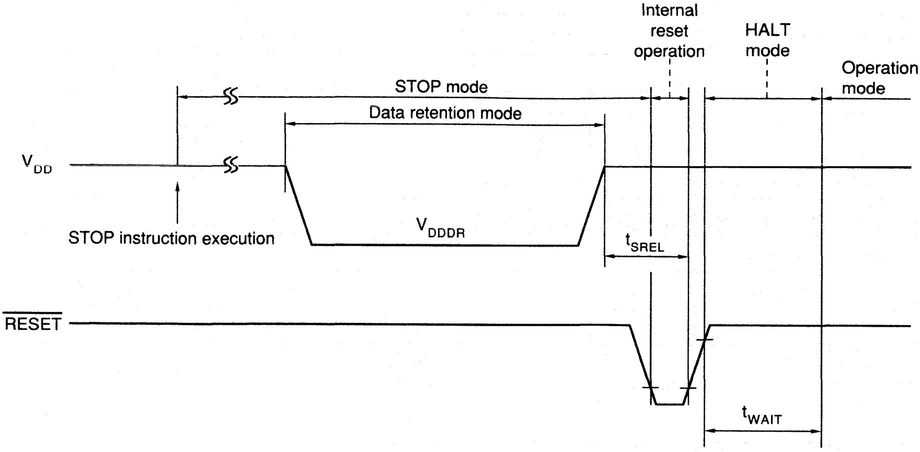
Note 1: The current drained through the internal pull-up resistor is not included

Note 2: The oscillation stabilization wait time is used to prevent unstable CPU operation at the beginning of oscillator operation, during which CPU operation is disabled.

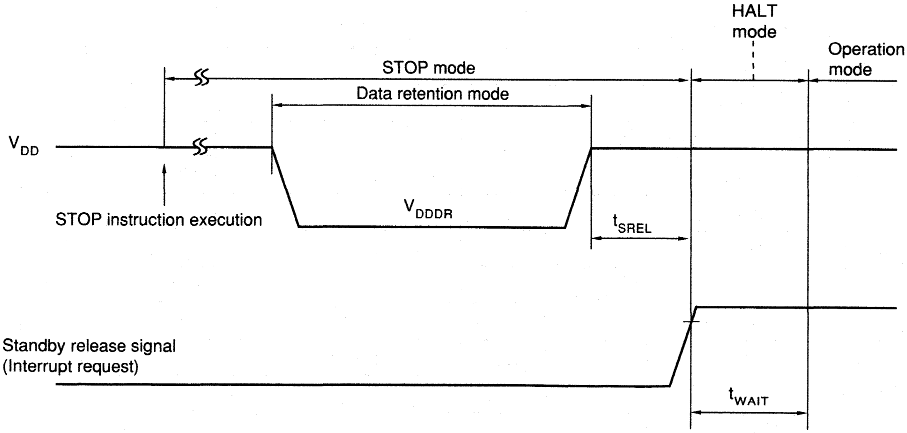
Note 3: This value depends on the setting of the basic interval timer mode register (BTM). (Refer to the table below. )

| BTM3 | BTM2 | BTM1 | BTM0 | WAIT time Parantheses ( ) indicates $f_X$ =4.19MHz |
|---|---|---|---|---|
| — | 0 | 0 | 0 | $2^{20}/f_X$ (Approximately 250 ms) |
| — | 0 | 1 | 1 | $2^{17}/f_X$ (Approximately 31.3 ms) |
| — | 1 | 0 | 1 | $2^{15}/f_X$ (Approximately 7.82 ms) |
| — | 1 | 1 | 1 | $2^{13}/f_X$ (Approximately 1.95 ms) |

Data Retention Timing (Releasing STOP mode by $\overline{\text{RESET}}$ )



Data Retention Timing
(Standby release signal: Releasing STOP mode by interrupt signal)



**DC Programming Characteristics** (Ta = 25 °C, $V_{DD}$ = 6.0 ± 0.25V, $V_{PP}$ = 12.5 ± 0.3V, $V_{SS}$ = 0V)

| Item | Symbol | Test Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| High level input voltage | $V_{IH1}$ | Other than X1, X2 | $0.7V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH2}$ | X1, X2 | $V_{DD}$ –0.5 | | $V_{DD}$ | V |
| Low level input voltage | $V_{IL1}$ | Other than X1, X2 | 0 | | 0.3 $V_{DD}$ | V |
| | $V_{IL2}$ | X1, X2 | 0 | | 0.4 | V |
| Input leak current | $I_{LI}$ | $V_{IN} = V_{IL}$ or $V_{IH}$ | | | 10 | μA |
| High level output voltage | $V_{OH}$ | $I_{OH}$ = –1 mA | $V_{DD}$ –1.0 | | | V |
| Low level output voltage | $V_{OL}$ | $I_{OL}$ = 1.6 mA | | | 0.4 | V |
| $V_{DD}$ power supply current | $I_{DD}$ | | | | 30 | mA |
| $V_{PP}$ power supply current | $I_{PP}$ | MD0 = $V_{IL}$, MD1 = $V_{IH}$ | | | 30 | mA |

Note 1: Ensure that $V_{PP}$ does not exceed +13.5V including overshoot.
Note 2: Ensure that $V_{DD}$ is applied before $V_{PP}$, and is turned off after $V_{PP}$.

**AC Programming Characteristics** (Ta = 25 °C, $V_{DD}$ = 6.0 ± 0.25V, $V_{PP}$ = 12.5 ± 0.3V, $V_{SS}$ = 0V)

| Item | Symbol | Note 1 | Test Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Address setup time (Note 2) (to MD0 ↓ ) | $t_{AS}$ | $t_{AS}$ | | 2 | | | μs |
| MD1 setup time (to MD0 ↓ ) | $t_{M1S}$ | $t_{OES}$ | | 2 | | | μs |
| Data setup time (to MD0 ↓ ) | $t_{DS}$ | $t_{DS}$ | | 2 | | | μs |
| Address hold time (Note 2) (to MD0 ↑ ) | $t_{AH}$ | $t_{AH}$ | | 2 | | | μs |
| Data hold time (to MD0 ↑ ) | $t_{DH}$ | $t_{DH}$ | | 2 | | | μs |
| MD0 ↑ → data output float delay time | $t_{DF}$ | $t_{DF}$ | | 0 | | 130 | ns |
| $V_{PP}$ setup time (to MD3 ↑ ) | $t_{VPS}$ | $t_{VPS}$ | | 2 | | | μs |
| $V_{DD}$ setup time (to MD3 ↑ ) | $t_{VDS}$ | $t_{VCS}$ | | 2 | | | μs |
| Initial program pulse width | $t_{PW}$ | $t_{PW}$ | | 0.95 | 1.0 | 1.05 | ms |
| Additional program pulse width | $t_{OPW}$ | $t_{OPW}$ | | 0.95 | | 21.0 | ms |
| MD0 setup time (to MD1 ↑ ) | $t_{M0S}$ | $t_{CES}$ | | 2 | | | μs |
| MD0 ↓ → data output delay time | $t_{DV}$ | $t_{DV}$ | MD0 = MD1 = $V_{IL}$ | | | 1 | μs |
| MD1 hold time (to MD0 ↑ ) | $t_{M1H}$ | $t_{OEH}$ | $t_{M1H} = t_{M1R} \geq 50\mu s$ | 2 | | | μs |
| MD1 recovery time (to MD0 ↓ ) | $t_{M1R}$ | $t_{OR}$ | | 2 | | | μs |
| Program Counter reset time | $t_{PCR}$ | — | | 10 | | | μs |
| X1 input high/low level width | $t_{XH}$,$t_{XL}$ | — | | 0.125 | | | μs |
| X1 input frequency | $f_X$ | — | | | | 4.19 | MHz |
| Initial mode setting time | $t_I$ | — | | 2 | | | μs |
| MD3 setup time (to MD1 ↑ ) | $t_{M3S}$ | — | | 2 | | | μs |

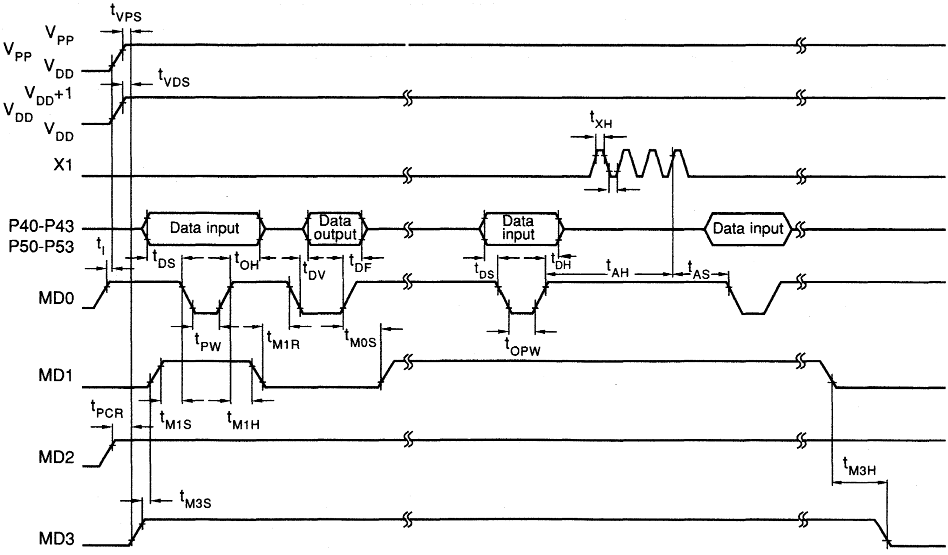
**AC Programming Characteristics** (Ta = 25 °C, $V_{DD}$ = 6.0 ± 0.25V, $V_{PP}$ = 12.5 ± 0.3V, $V_{SS}$ = 0V) (cont'd)

| Item | Symbol | Note 1 | Test Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| MD3 hold time (to MD1 ↓ ) | $t_{M3H}$ | — |  | 2 |  |  | μs |
| MD3 setup time (to MD0 ↓ ) | $t_{M3SR}$ | — | For program memory read | 2 |  |  | μs |
| Address (Note 2) data output delay time | $t_{DAD}$ | $t_{ACC}$ | For program memory read | 2 |  |  | μs |
| Address (Note 2) data output hold time | $t_{HAD}$ | $t_{OH}$ | For program memory read | 0 |  | 130 | ns |
| MD3 hold time (to MD0 ↑ ) | $t_{M3HR}$ | — | For program memory read | 2 |  |  | μs |
| MD3 ↓ → data output float delay time | $t_{DFR}$ | — | For program memory read | 2 |  |  | μs |

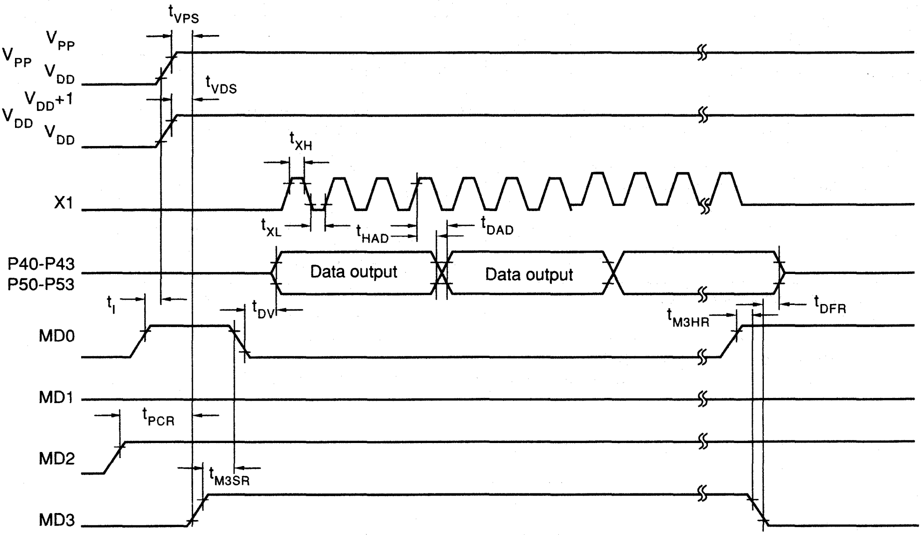
Note 1: Symbol for corresponding μPD27C256.

Note 2: The internal address signal is incremented by 1 by the rise of the 4th X1 input, and is not connected to a pin.

Program Memory Write Timing

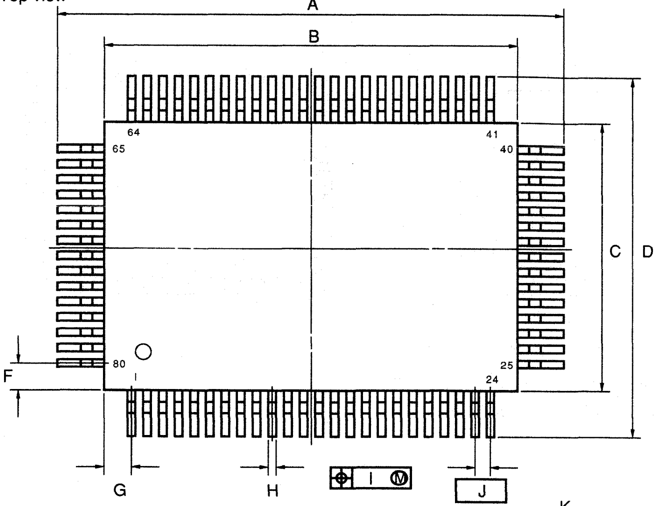


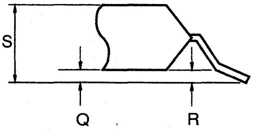
Program Memory Read Timing

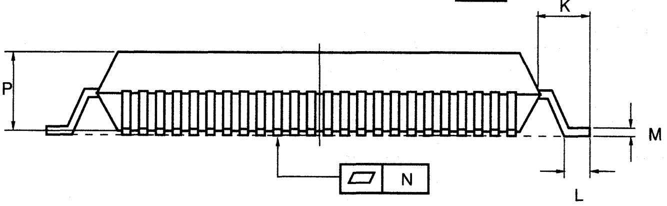


### 10.9 Package Information 80-pin Plastic QFB

Top view







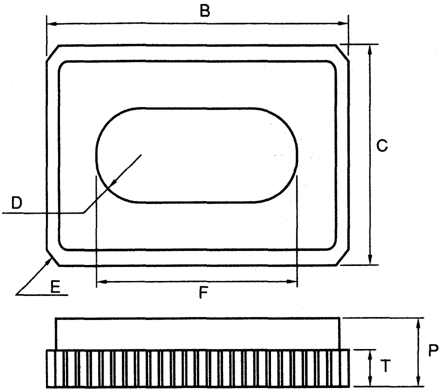
P80GF-80-3B9

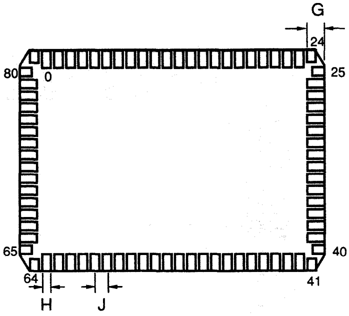
**Note**

Each lead center line is located whithin 0.15mm (0.006 inchof its true position (T.P.) at maximum material condition

| ITEM | MILLIMETERS | INCHES |
|---|---|---|
| A | $23.6\pm^{0.4}$ | $0.929\pm^{0.016}$ |
| B | $20\pm^{0.2}$ | $0.795\pm^{0.009}_{0.008}$ |
| C | $14\pm^{0.2}$ | $0.551\pm^{0.009}_{0.008}$ |
| D | $17.6\pm^{0.4}$ | $0.693\pm^{0.016}$ |
| F | 1.0 | 0.039 |
| G | 0.8 | 0.031 |
| H | $0.35\pm^{0.10}$ | $0.014\pm^{0.004}_{0.005}$ |
| I | 0.15 | 0.006 |
| J | 0.8 (T.P.) | 0.031 (T.P.) |
| K | $1.8\pm^{0.2}$ | $0.071\pm^{0.008}_{0.009}$ |
| L | $0.8\pm^{0.2}$ | $0.031\pm^{0.009}_{0.008}$ |
| M | $0.15\pm^{0.10}_{0.05}$ | $0.006\pm^{0.004}_{0.003}$ |
| N | 0.15 | 0.006 |
| P | 2.7 | 0.106 |
| Q | $0.1\pm^{0.1}$ | $0.004\pm^{0.004}$ |
| R | $0.1\pm^{0.1}$ | $0.004\pm^{0.004}$ |
| S | 3.0 MAX. | 0.119 MAX. |

Top view





X80KW-80A

| ITEM | MILLIMETERS | INCHES |
|---|---|---|
| B | $20.0\pm^{0.4}$ | $0.787\pm^{0.017}_{0.016}$ |
| C | $14.2\pm^{0.4}$ | $0.559\pm^{0.016}$ |
| D | R 3.0 | R 0.118 |
| E | C 0.3 | C 0.012 |
| F | 12.0 | 0.472 |
| G | $1.0\pm^{0.2}$ | $0.039\pm^{0.009}_{0.008}$ |
| H | $0.51\pm^{0.1}$ | $0.020\pm^{0.001}$ |
| J | $0.8\pm^{0.1}$ | $0.031\pm^{0.005}_{0.004}$ |
| P | 3.8 MAX. | 0.150 MAX. |
| T | 2.14 | 0.084 |

## CHAPTER 4 µPD75328 4 BIT MICROCOMPUTER WITH L.C.D. CONTROLLER/DRIVER

### 1. OVERVIEW

The µPD75328 is a product in the series of µCOM-75X 4-bit single-chip microcomputers which contains an internal programmable LCD controller/driver and standard NEC serial bus interface, High-speed operation and high-advanced functions are more of the features.

Their Products are especially enhanced as the described below, in comparison with the µPD7514 which has contained the internal LCD controller/driver in µPD7500 series.

- ROM capacitance : Maximum 8064 words x 8 bits
- RAM capacitance: 512 x 4 bits
- General purpose register: Maximum 8 x 4 bits
- High-speed operation: Minimum instruction execution time = 1 us
- Eight interrupt sources and efficient interrupt service
- Efficient instruction system which can handle 1-, 4-, or 8-bit data by each data length
- Three-channel multifunction timer
- Very low power consumption for clock operation in standby mode (subsystem clock consumes very little power)
- Internal 6 channel 8 bit precision A/D converter

Application Fields

- Appliances: VTR's, Audio sets (CD player), etc.
- Others : Cameras, Tonometers, Air conditioners.

### 1.1 Features

- 41 systematic instructions
  - Versatile bit manipulation instructions
  - Efficient 4-bit data manipulation instructions
  - 8-bit data transfer instructions
  - GETI instructions convert any 2- or 3-byte instruction into a 1-byte instruction

- High-speed operation
  Minimum instruction execution time = 0.95 us ($f_X$ = 4.91 MHz, at 5 V operation)

- Variable instruction execution time for low voltage, low current operation
  - when main system clock is selected: 0.95 us, 1.91 us, 15.3 us/4.19 MHz
  - when subsystem clock is selected: 122 us/32.768 khz

- Program memory capacitance (ROM):

- Data memory capacity (RAM): 512 x 4 bits

- General purpose registers
  - Eight registers for 4-bit operations: X, A, B, C, D, E, H, L
  - Four pairs of registers for 8-bit operations: XA, BC, DE, HL

- Accumulators
  - One-bit accumulator (CY)
  - 4-bit accumulator (A)
  - 8-bit accumulator (XA)

- 68 I/O lines
  - 24 LCD drive output pins
  - 12 segment output dedicated pins
  - 8 dual function pins used for both segment output and CMOS output
  - 4 common output pins
  - 8 input pins
  - 8 middle-voltage N-channel open drain input/output pins (8 pins for driving an LED directly)
  - 16 CMOS input/output pins (4 pins for driving an LED directly)

- Internal pull-up resistors for 35 I/O pins
- 27 pins specified by using software
- 8 pins specified by using mask option

- LCD controller/driver
  - Selection of the number of segments: 12, 16, or 20 (4 or 8 segments can be changed to bit output port.)
  - Display mode selection:
    Static
    1/2 duty (1/2 bias)
    1/3 duty (1/2 bias)
    1/3 duty (1/3 bias)
    1/4 duty (1/3 bias)
  - Split resistor for LCD drive voltage supply can be incorporated (mask option).

- 3-channel timers
  - 8-bit timer/event counter
    Four-stage clock source
    Event counting possible
  - 8-bit basic interval generation
    Reference time generation (1.95 ms, 7.82 ms, 31.3 ms, 250 ms/4.19 MHz)
    It is applicable to a watchdog timer.
  - Watch timer
    0.5 s time interval generation
    Count clock source: A change can be made between main and subsystem clocks
    Watch rapid feed mode (3.9 ms time interval generation)
    Buzzer output can be made. (2 kHz)

- 8-bit serial interface
  - It provides three modes:
    3-line serial I/O mode
    2-line serial I/O mode
    SBI mode
  - The serial transfer data top can be changed between the least and most significant bits (LSB and MSB).

- 8-bit A/D converter is contained
  - Analog input: six channels
  - A/D conversion rate: 40.1 us at $f_x = 4.19$ MHz
  - A/D operation range: $V_{DD} = 3..5$ to 6.0 (V) $AV_{REF} = 2.5$ to $V_{DD}$ (V), Ta = -10 to + 80 °C

- 16-bit bit sequential buffer (special bit manipulation memory):
  - Suitable for remote control applications.

- Clock output function
  - Timer/event counter output (PTO0):
    Outputs a square wave of any desired frequency
  - Clock output (PCL) : $\phi$, $f_x/2^3$, $f_x/2^4$, $f_x2^6$
  - Buzzer output (BUZ): 2 kHz (main system clock 4.19 MHz at operation/subsystem clock 32.7 kHz at operation)

- Vectored interrupt function
  - Three external vectored interrupts
    Both rising and falling edge detection interrupt (INT4)
    Detection edge programmable interrupt with noise elimination function (INT0)
    Detection edge programmable interrupt (INT1)
  - External rising edge detection/parallel port edge detection test input (INT2)
  - Three internal vectored interrupts:
    Timer/event counter 0 interrupt (INTT0)
    Basic interval timer interrupt (INTBT)
    Serial interface interrupt (INTCSI
  - Watch test input (INTW)

- Two interal oscillators for system clock genertion
  - Ceramic or crystel oscillator for main system clock oscillation: 4.194304 MHz Standard
  - Crystal oscillator for subsystem clock oscillation: 32.768 kHz Standard

- Standby operation
  - STOP mode: Main system clock oscillation stops.
  - HALT mode: System clock oscillation continues. (CPU clock oscillation continues.)

- CMOS

### 1.2 Ordering Information

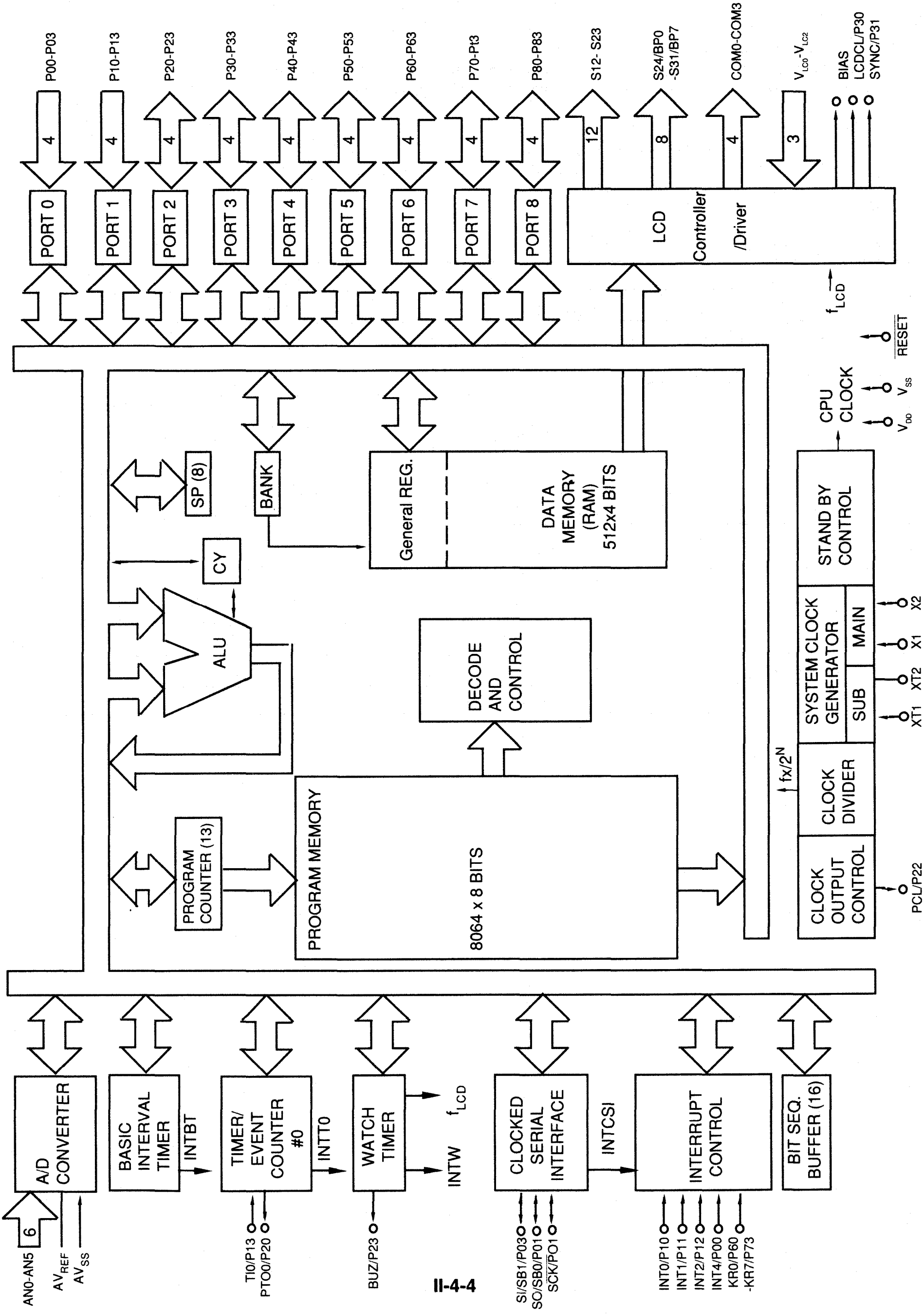
| Ordering code | Package | Internal ROM |
|---|---|---|
| μPD75328GC-xxx-3B9 | 80-pin plastic QFP (bent leads) | Mask ROM |

Remark: xxx shows the code number of the mask ROM and option specification

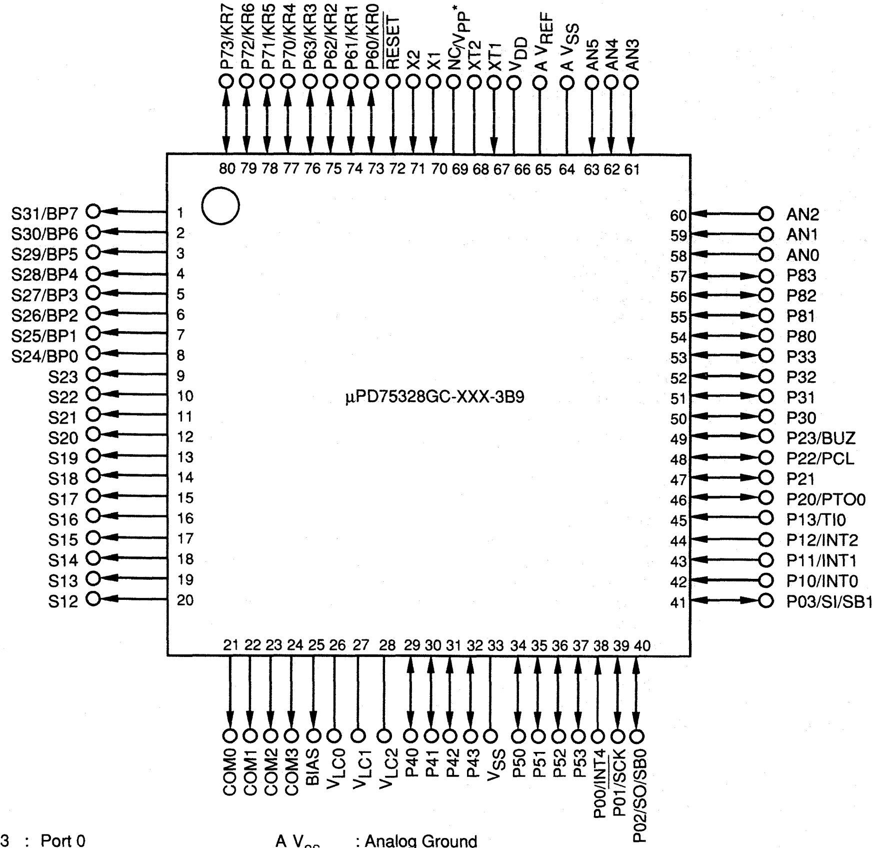
### 1.3 List of Family Product Functions

| Item | μPD75328 | μPD75P328 |
|---|---|---|
| Program memory | • Mask ROM<br>• 8064 x 8 bits<br>• 0000H-1F7FH | • One time PROM<br>• 8064 x 8 bits<br>• 0000H-1F7FH |
| Pull-up resistor in Port 4, 5 | Mask option | Not included |
| Split resistor for LCD drive power supply | Mask option | Not included |
| Feed back resistor for subsystem clock oscillation | | |
| Pin connection | $V_{PP}$ pin, one time PROM program pin not included | $V_{PP}$ pin, one time PROM program pin included |
| Supply voltage range | 2.7V – 6.0V | 5V ± 15% |
| Package | 80-pin plastic QFP (bent leads) | |

## 1.4 Block Diagram



### 1.5 Pin Connection (Top view)



| | | | |
|---|---|---|---|
| P00-03 | : Port 0 | A $V_{SS}$ | : Analog Ground |
| P10-13 | : Port 1 | AN0-5 | : Analog Input 0-5 |
| P20-23 | : Port 2 | S12-31 | : Segment Output 12-31 |
| P30-33 | : Port 3 | COM0-3 | : Common Output 0-3 |
| P40-43 | : Port 4 | $V_{LC0-2}$ | : LCD Power Supply 0-2 |
| P50-53 | : Port 5 | BIAS | : LCD Power Supply Bias Control |
| P60-63 | : Port 6 | LCDCL | : LCD Clock |
| P70-73 | : Port 7 | SYNC | : LCD Synchronization |
| P80-83 | : Port 8 | TI0 | : Timer Input 0 |
| BP0-7 | : Bit Port | PTO0 | : Programmable Timer Output 0 |
| KR0-7 | : Key Return | BUZ | : Buzzer Clock |
| SCK | : Serial Clock | PCL | : Programmable Clock |
| SI | : Serial Input | INT0, 1, 4 | : External Vectored Interrupt 0, 1, 4 |
| SO | : Serial Output | INT2 | : External Test Input 2 |
| SB0, 1 | : Serial Bus 0, 1 | X1, 2 | : Main System Clock Oscillation 1, 2 |
| RESET | : Reset Input | XT1, 2 | : Subsystem Clock Oscillation 1, 2 |
| A $V_{REF}$ | : Analog Reference | NC/$V_{PP}$* | : No Connection |

* When using the same socket for µPD75P328 and µPD75328, $V_{PP}$ pin has to be set to $V_{DD}$ in operating mode!

## 2. PIN FUNCTIONS

### 2.1 Input and output ports

2.1.1 P00-P03 (port 0) - Input Pins Also Used for INT4, $\overline{\text{SCK}}$, SO/SB0, SI/SB1
P10-P13 (port 1) - Input Pins Also Used for INT0, INT1, INT2, and TI0

P00-P03 and P10-P13 are pins for 4-bit input port (port 0 and 1).

(1) The P00-P03 pins are also used for the INT4 pin (vectored interrupt input) and the $\overline{\text{SCK}}$, SO/SB0, and SI/SB1 pin (serial interfaceb output).

(2) The P10-P13 pins are also used for the INT0 and INT1 pins (vectored interrupt input), the INT2 pin (edge detection test input) and the TI0 pin (external event pulse to the timer/event counter).

The state of each pin P00-P03 and P10-P13 can always be input regardless of the condition of any of the pins INT4, $\overline{\text{SCK}}$, SO/SB0, SI/SB1, INT0, INT1, INT2, and TI0.

To prevent noise from causing a malfunction, a Schmitt trigger input is applied to port 0 P00/INT4, P01/$\overline{\text{SCK}}$, P02/SO/SB0, P03/SI/SB1 input and port 1 pins. In addition, P10 is provided with a noise removal circuit. (For details, see 6.3 (2).)

An internal pull-up resistor can be specified for port 0 in 3-bit units (P01-P03) and for port 1 in 4-bit units (P10-P13) by using the software-operated pull-up resistor specification register group A.

When the $\overline{\text{RESET}}$ signal is generated, every pin is placed in input port mode.

2.1.2 P20-P23 (port 2) - Input/Output Pins Also Used for PTO0, PCL, and BUZ
P30-P33 (port 3) - Input/Output Pins Also Used for LCDCL and SYNC.
P40-P43 (port 4) - and P50-53 (port 5) - N-Channel Open Drain Medium-Voltage (10 V)
P60-P63 (port 6), P70-73 (port 7), P80 - P83 (port 8) - 3-State Input/Output

P20-P23, P30-P33, P40-P43, P50-P53, P60-P63, P70-P73 and P80-P83 are 4-bit input/output ports with output latches.

(1) The P20-P23 pins are also used for the PTO0 pin timer/event counter output), the PCL pin (programmable clock output), and the BUZ pin (fixed frequency output).

(2) The P30-P33 pins are also used for the LCDCL pin (LCD external extended driver driving clock) and the SYNC pin (LCD) external extended driver synchronizing clock).

(3) The P60-P63 and P70-P73 pins are also used for the KR0-KR3 and KR4-KR7 pins (key interrupt input).

Port 3 has high-current output, enabling an LED to be driven directly. Ports 4 and 5 outputs are N-channel open drain medium-voltage (10 V) and high-current, enabling an LED to be driven directly.

The input/output mode of each port is selected by using the port mode register. The mode of port 2, 4, 5, 7 or 8 can be specified in 4-bit units, and that of port 3 or 6 bitwise.

An internal pull-up resistor can be specified for ports 2, 3, 6, 7 and 8 in 4-bit units by using software to operate the pullup resistor specification register (POGA, POGB). It can be specified for ports 4 and 5 bitwise by using the mask option.

Ports 4 and 5 (6 and 7) can be paired for input/output in 8-bit units. When the $\overline{\text{RESET}}$ signal is generated, ports 2, 3, 6, 7 and 8 are placed in input mode (output high impendance) and ports 4 and 5 are set to a high level (when internal pull-up register is contained) or to high impendance.

2.1.3 BP0-BP7 - Output Pins Also Used for S24-S31 (LCD Controller/Driver Segment Signal Output)

BP0-BP7 are bit port 0-7 output pins (each port is a 1-bit output port with output latch). The BP0-BP7 pins are also used for the S24-S31 pins (LCD controller/driver segment output signal output pins).

2.1.4 TI0 - Input Pin Also Used for Port 1

TI0 is an external event pulse input pin for the programmable timer/event counter.
The pin is Schmitt trigger input.

2.1.5 PTO0 - Output Pin Also Used for Port 2

PTO0 is a programmable timer/event counter output pin which outputs square wave pulses. To output a programmable timer/event counter signal, the P20 output latch is cleared and the port 2 bit of the port mode register is set to 1 (output mode).
Output is cleared by execution of the timer start instruction.

2.1.6 PCL - Output Pin Also Used for Port 2

PCL is a programmable clock output pin used to supply clocks to peripheral LSI devices such as a slave microcomputer and an A/D converter. When the $\overline{\text{RESET}}$ signal is generated, the clock mode register (CLOM) is cleared, clock output is inhibited, and the normal port operation mode is set.

### 2.1.7 BUZ - Output Pin Also Used for Port 2

BUZ is a fixed frequency output pin. Fixed frequency (2.048 kHz) output is used for buzzer sounding and system clock oscillation frequency trimming. The BUZ pin, also used for the P23 pin, is validated only when bit 7 (WM7) of the watch mode register (WM) is set to 1.

When the RESET signal is generated, WM7 is cleared and the normal port operation mode is set.

### 2.1.8 SCK, SO/SB0, and SI/SB1 - 3-State Input/Output Pins Also Used for Port 0

SCK, SO/SB0, and SI/SB1 are input/output pins for serial interface and operate according to how the serial operation mode register (CSIM) is set.

When the RESET signal is generated, serial interface operation is stopped and the pins are used for port 0 (input port).

Every pin is Schmitt trigger input.

### 2.1.9 INT4 - Input Pin also Used for Port 0.

INT4 is an external vectored interrupt input pin (both rising and falling edged active). When the signal input to the pin changes from low to high state or from high to low state, the interrupt request flag is set.

INT4 is for asynchronous input. When a signal having a given high or low level duration is input, it is acknowledged independently of the CPU operation clock.

INT4 can also be used to release the STOP or HALT mode. It is Schmitt trigger input.

### 2.1.10 INT0 and INT1 - Input Pins Also Used for Port 1

INT0 and INT1 are edge detection vectored interrupt input pins. INT0 has the noise removal function. Detected edge selection can be made using edge detection mode registers (IM0 and IM1).

(1) INT0 (IM0 bits 0 and 1)
(a) Rising edge active
(b) Falling edge active
(c) Both rising and falling edges active
(d) External interrupt signal input inhibited

(2) INT1 (IM1 bit 0)
(a) Rising edge active
(b) Falling edge active

INT0 has the noise removal function; sampling clocks for noise removal can be changed two stages. The acknowledged signal width varies depending on CPU clock operation.

INT1 is for asynchronous input. When a signal having a given high level duration is input, it is acknowledged independently of the CPU clock operation.

When the RESET signal is generated, IM0 and IM1 are cleared and rising edge active is selected.

INT0 and INT1 are Schmitt trigger inputs.

### 2.1.11 INT2 - Input Pin Also Used for Port 1

INT2 is an external test input pin (both rising and falling edges active). When INT2 is selected by using the edge detection mode register (IM2) and the signal input to the INT2 pin transits from low to high state, the internal test flag (IRQ2) is set.

INT2 is for asynchronous input. When a signal having a given high level duration is input, it is acknowledged independently of the CPU clock operation.

When the RESET signal is generated, IM2 is cleared and the test flag (IRQ2) is set by inputting the rising edge to the INT2 pin.

### 2.1.12 KR0-KR3 - Input Pins Also Used for Port 6
### KR4-KR7 - Input Pins Also Used for Port 7

KR0-KR7 are key interrupt (parallel falling edge detection interrupt) input pins. The interrupt format can be specified by setting the edge detection mode register (IM2).

When the RESET signal is generated, the pins are placed in port 6 and 7 input mode.

### 2.1.13 S12-S23 - Output
### S24-S31 - Output Pins Also Used for Bit Ports 0-7

S12-S31 are segment signal output pins for directly driving LCD segments (front electrodes). Static, 2 or 3 time division (1/2 bias law), or 3 or 4 time division (1/3 bias law) driving is performed.

S0-S23 are segment-only output pins. S24-S31 are also used for the bit port 0-7 output; which mode the pins are used for, is specified by using the dislay mode register (LCDM).

### 2.1.14 COM0-COM3 - Output

COM0-COM3 are common signal output pins for directly driving the LCD common pins (rear electrodes). Common signals are output when static (COM0, COM1, COM2, COM3 output), while driving by 1/2 bias law 2 time division (COM0, COM1 output), while driving by 3 time division (COM0, COM1, COM2 output), while driving by 1/3 bias law 3 time division (COM0, COM1, COM2, COM3 output) is performed.

2.1.15 $V_{LC0}$-$V_{LC2}$

$V_{LC0}$-$V_{LC2}$ are power supply pins to drive the LCD. The μPD75328 allows split-resistor incorporation in the $V_{LC0}$-$V_{LC2}$ pins so that LCD drive power can be supplied according to the bias law without using an external split resistor. (Mask option)

2.1.16 BIAS

BIAS is an LCD power supply bias control pin. To deal with various LCD drive voltages, the BIAS pin is connected to the $V_{LC0}$ pin to change the resistor split ratio. By connecting an external resistor, the BIAS pin can be used together with the $V_{LC0}$-$V_{LC2}$ and $V_{SS}$ pins for fine adjustment of the LCD drive supply voltage.

2.1.17 LCDCL

LCDCL is a clock output pin for driving an external LCD extension driver.

2.1.18 SYNC

SYNC is a clock output pin for synchronizing an external LCD extension driver.

2.1.19 AN0-AN5 - Input

AN0-AN5 are six analog signal input pins to A/D converter.

2.1.20 $AV_{REF}$

$AV_{REF}$ is an A/D converter reference voltage input pin.

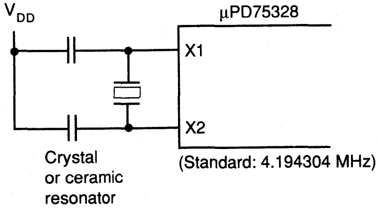
2.1.21 $AV_{SS}$

$AV_{SS}$ is an A/D converter GND pin.

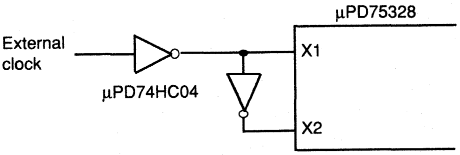
2.1.22 X1 and X2

X1 and X2 are connection pins for the main system clock crystal / ceramic oscillator.
External clocks can also be input.

(a) Crystal ceramic oscillator



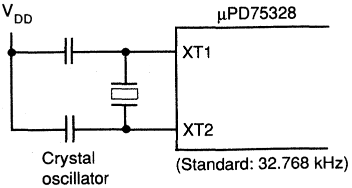
(b) External clock



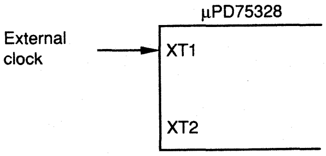
2.1.23 XT1 and XT2

XT1 and XT2 are crystal connection pins for the subsystem clock oscillation.
External clocks can also be input.

(a) Crystal oscillator



(b) External clock



2.1.24 $\overline{\text{RESET}}$

$\overline{\text{RESET}}$ is an active low reset input pin.
$\overline{\text{RESET}}$ is for asynchronous input. When a signal having a given low level duration is input independently of the operation clock, the $\overline{\text{RESET}}$ signal is generated and the system is reset overriding all other operations.
It is used for normal CPU initialization / start-up and also to release the standby (STOP or HALT) mode.
$\overline{\text{RESET}}$ is a Schnitt trigger input.

2.1.25 $V_{DD}$

$V_{DD}$ is a positive power supply pin.

2.1.26 $V_{SS}$

$V_{SS}$ is a ground potential.

## 2.2 Pin Function List

(1) Normal operation mode

**Table 2.2–1 Digital Input / Output Port Pin Function List**

| Pin name | I/O | Also used for: | Function | 8-bit I/O | When reset | I/O circuit type (Note 1) |
|---|---|---|---|---|---|---|
| P00 | I | INT4 | 4-bit input port (port 0).<br>Internal pull-up resistors can be specified for P10-P13 in 3-bit units by using software | x | Input | Ⓑ |
| P01 | I/O | $\overline{\text{SCK}}$ | | | | Ⓕ- A |
| P02 | I/O | SO/SBO | | | | Ⓕ- B |
| P03 | I/O | SI/SB1 | | | | Ⓜ- C |
| P10 | I | INT0 | With noise removal function | x | Input | Ⓑ- C |
| P11 | | INT1 | 4-bit input port (port 1).<br>Internal pull-up resistors can be specified for P10-P13 in 4-bit units by using software | | | |
| P12 | | INT2 | | | | |
| P13 | | TI0 | | | | |
| P20 | I/O | PTO0 | 4-bit input/output port (port 2).<br>Internal pull-up resistor can be specified for P20-P23 in 4-bit units by using software. | x | Input | E – B |
| P21 | | – | | | | |
| P22 | | PCL | | | | |
| P23 | | BUZ | | | | |
| P30 | I/O | LCDCL | Programmable 4-bit input/output port (port 3). Input or output mode can be selected bitwise.<br>Internal pull-up resistor can be specified for P30-P33 in 4-bit units by using software.(Note 2) | x | Input | E – B |
| P31 | | SYNC | | | | |
| P32 | | – | | | | |
| P33 | | – | | | | |
| P40-P43 (Note 2) | I/O | – | N-channel open drain 4-bit input/output port (port 4).<br>Pull-up resistor can be incorporated bitwise (mask option).<br>10 Volts during open drain. | o | High level (when pull-up resistor is incorporated) or high impendance | M |
| P50-P53 (Note 2) | I/O | – | N-channel open drain 4-bit input/output port (port 4).<br>Pull-up resistor can be incorporated bitwise (mask option).<br>10 Volts during open drain. | | High level (when pull-up resistor is incorporated) or high impendance | M |

(to be continued)

**Table 2.2–1 Digital Input / Output Port Pin Function List (cont`d)**

| Pin name | I/O | Also used for | Function | 8-bit I/O | When reset | I/O circuit type (Note 1) |
|---|---|---|---|---|---|---|
| P60 | I/O | KR0 | Programmable 4-bit input/output port (port 6). Input or output mode can be selected bitwise. | o | Input | Ⓕ- A |
| P61 | | KR1 | | | | |
| P62 | | KR2 | Internal pull-up resistor can be specified for P60-P63 in 4-bit units by using software. | | | |
| P63 | | KR3 | | | | |
| P70 | I/O | KR4 | 4-bit input/output port (port 7). Internal pull-up resistor can be specified for P70-P73 in 4-bit units by using software. | | Input | Ⓕ- A |
| P71 | | KR5 | | | | |
| P72 | | KR6 | | | | |
| P73 | | KR7 | | | | |
| P80 | I/O | – | 4-bit input/output port (Port 8). Internal pull-up resistor can be specified for P80-P83 in 4-bit units by using software. | X | Input | E – B |
| P81 | | | | | | |
| P82 | | | | | | |
| P83 | | | | | | |
| BP0 | O | S24 | 1-bit output ports (bit ports). x The pins are also used for the segment output pins. | x | Note 3 | G – C |
| BP1 | | S25 | | | | |
| BP2 | | S26 | | | | |
| BP3 | | S27 | | | | |
| BP4 | O | S28 | | | | |
| BP5 | | S29 | | | | |
| BP6 | | S30 | | | | |
| BP7 | | S31 | | | | |

Note 1: The ○ mark denotes Schmitt trigger input

Note 2: LED can be driven directly

Note 3: BP0-7 select $V_{LC1}$ as the input source.
The output level is changed by the external circuit of BP0-7 and $V_{LC1}$.

Example: Since BP0-BP7 are connected to each other through the µPD75328 as illustrated below, the BP0-BP7 output level is determined by the resistance value of R1, R2, R3.

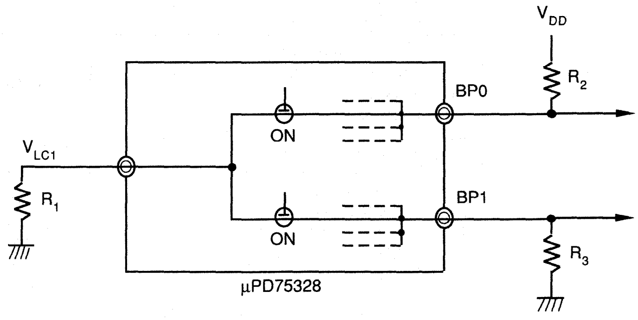


**Table 2.2–2 Pin Function List other than Port Pins**

| Pin name | I/O | Also used for: | Function | | When reset | I/O circuit type (Note 1) |
|---|---|---|---|---|---|---|
| TI0 | I | P13 | External event pulse input pin to timer / event counter. | | | Ⓑ-C |
| PT00 | I/O | P20 | Timer / event counter output pin. | | Input | E – B |
| PCL | I/O | P22 | Clock output pin. | | Input | E – B |
| BUZ | I/O | P23 | Fixed frequency output pin (for buzzer or system clock trimming). | | Input | E – B |
| $\overline{\text{SCK}}$ | I/O | P01 | Serial clock input / output pin. | | Input | Ⓕ-A |
| SO/SB0 | I/O | P02 | Serial data output pin.<br>Serial bus input / output pin. | | Input | Ⓕ-B |
| SI/SB1 | I/O | P03 | Serial data input pin.<br>Serial bus input / output pin. | | Input | Ⓜ-C |
| INT4 | I | P00 | Edge detection vectored interrupt input pin (detection of both rising and falling edges is active). | | | Ⓑ |
| INT0 | I | P10 | Edge detection vectored interrupt input pin (detected edge can be selected). | Synchronous clock | | Ⓑ-C |
| INT1 | | P11 | | Asynchronous | | |
| INT2 | I | P12 | Edge detection testable input pin asynchronous (rising edge is detected). | | | Ⓑ-C |
| KR0-KR3 | I/O | P60-P63 | Parallel falling edge detection testable input pins. | | Input | Ⓕ-A |
| KR4-KR7 | I/O | P70-P73 | Parallel falling edge detection testable input pins. | | Input | Ⓕ-A |
| S12-S23 | O | – | Segment signal output pins. | | Note 4 | G – A |
| S24-S31 | O | BP0-7 | Segment signal output pins. | | Note 4 | G – C |
| COM0-COM3 | O | – | Common signal output pins. | | Note 4 | G – B |
| $V_{LC0}$-$V_{LC2}$ | – | – | LCD drive power supply pins. Internal split resistor can be used (mask option). | | – | – |
| BIAS | O | – | Output pin to cut external split resistor. | | Note 5 | |
| LCDCL (Note 3) | I/O | P30 | Clock output pin for external extension driver. | | Input | E – B |
| SYNC (Note 3) | I/O | P31 | Clock output pin for synchronizing external extension driver. | | Input | E – B |
| AN0-AN5 | I | – | 6-bit analog input pins to A/D converter. | | | Y |
| $AV_{REF}$ | I | – | A/D converter reference voltage input pins. | | | Z |
| $AV_{SS}$ | I | – | A/D converter reference GND input pins. | | | |
| X1, X2 | | – | Connection pins for main system clock crystal/ceramic oscillator. When external clock is used, it is input to X1 and its opposite phase is input to X2. | | – | – |
| XT1 | I | – | Subsystem clock oscillation crystal connection pins. When external clock is used, it is input to XT1, and XT2 is not connected. XT1 can be used for 1-bit input (test) pin. | | – | – |
| XT2 | – | | | | – | – |
| $\overline{\text{RESET}}$ | I | | System reset input pin. | | – | Ⓑ |

4

**Table 2.2–2 Pin Function List other than Port Pins**

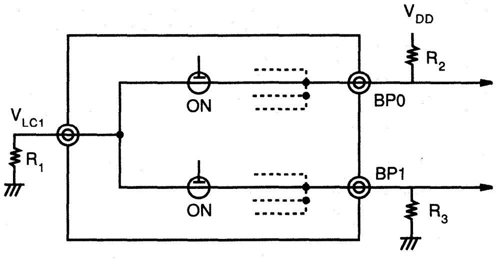
| Pin name | I/O | Also used for: | Function | When reset | I/O circuit type (Note 1) |
|---|---|---|---|---|---|
| NC<br>(Note 2) | – | | No connection | – | – |
| $V_{DD}$ | – | | Positive power supply pin. | – | – |
| $V_{SS}$ | – | | Ground potential pin. | – | – |

Notes:

1. The○ mark denotes Schmitt trigger input.
2. Connected the NC pin to $V_{DD}$, when μPD75P328 and printed board are also shared with.
3. The pins are provided for future system expansion; at present they are used only for the P30 and P31 pins
4. For each display output, the following $V_{LCX}$ are selected as input sources:

   S12-31 : $V_{LC1}$
   COM0-COM2 : $V_{LC2}$
   COM3 : $V_{LC0}$

   However, the display output level varies depending on the display output and $V_{LCX}$ external circuit.
   Example:
   Since BP0-BP7 are connected to each other through the μPD75328 as illustrated below, the BP0-BP7 output level is determined by the resistance value of R1, R2, R3.



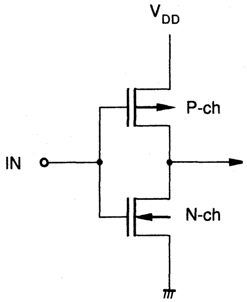
μPD75328

5. When internal split resistor is contained : Low level
   When internal split resistor is not contained: High impedance

### 2.3 Pin Input / Output Circuits

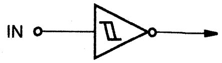
The μPD75328 pin input / output circuits are shown in schematic drawings.

(1) Type A (for Type E – B)



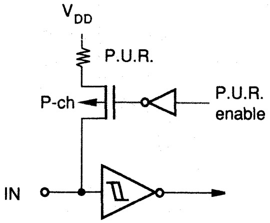
CMOS standard input buffer

(2) Type B



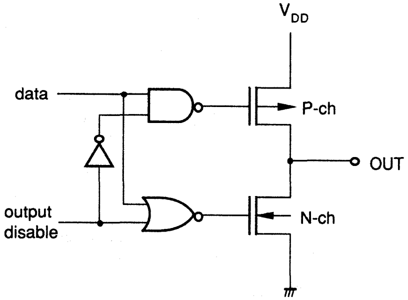
Schmitt trigger input with hysteresis characteristic

(3) Type B – C



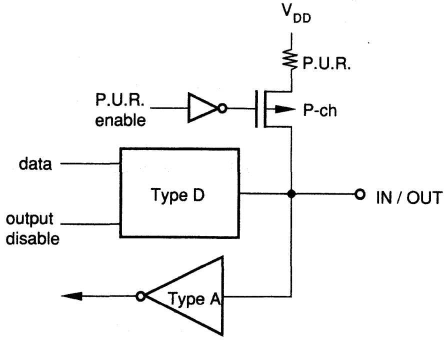
P.U.R.: Pull-Up Resistor

(4) Type D (for Type E – B, F – A)



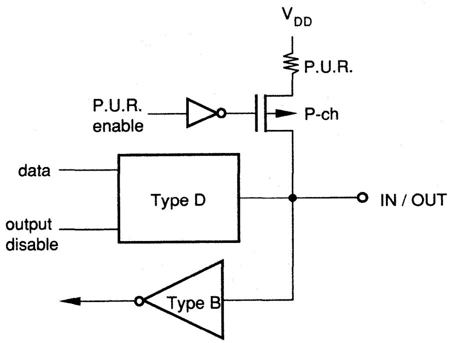
Push-pull output where output can be placed in high impedance (both P and N channels are turned off).

(5) Type E - B



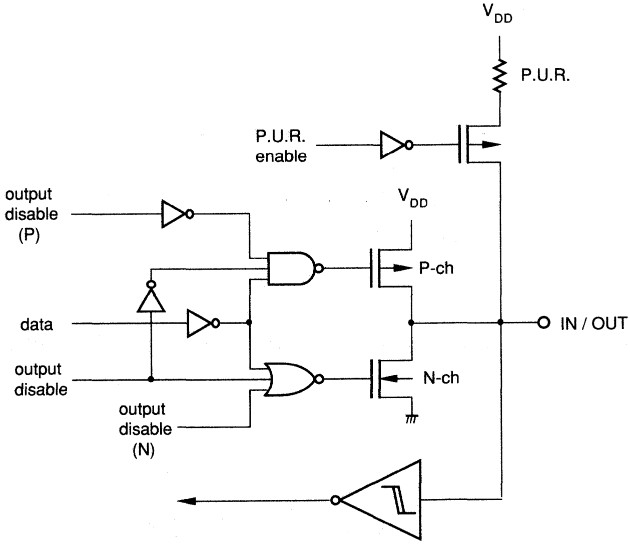
P.U.R.: Pull-Up Resistor

(6) Type F – A



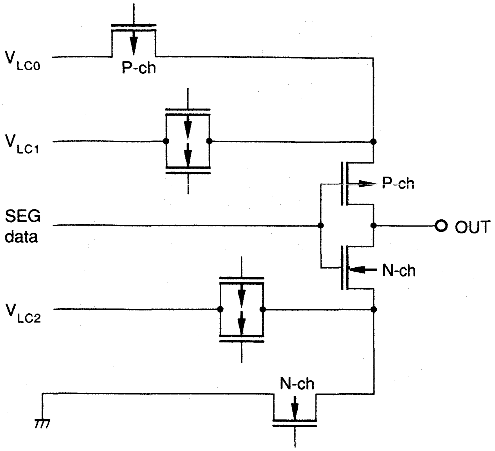
P.U.R.: Pull-Up Resistor

(7) Type F – B

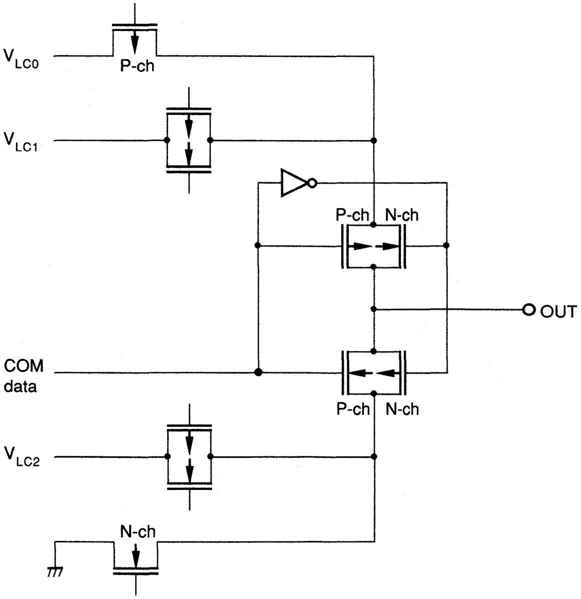


P.U.R.: Pull-Up Resistor

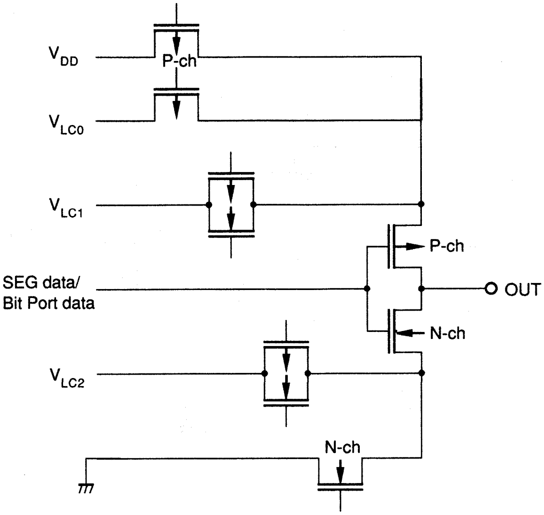
(8) Type G – A



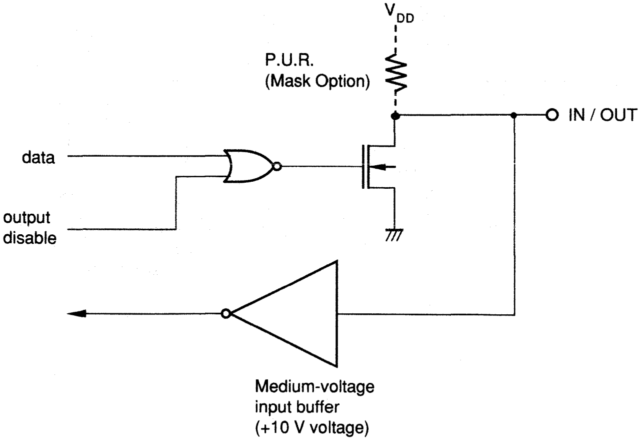
(9) Type G – B



(10) Type G – C

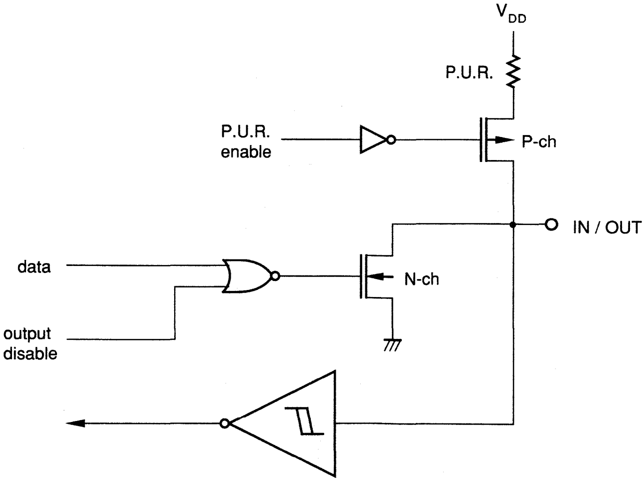


(11) Type M



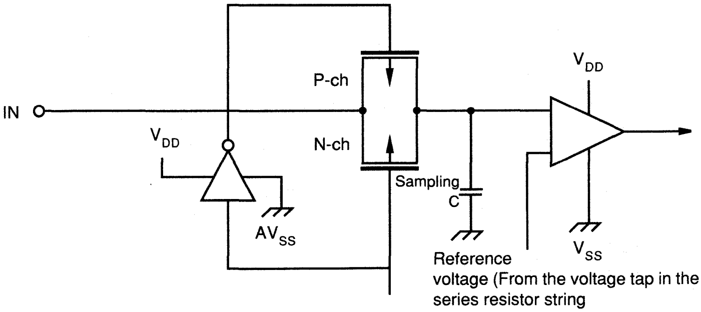
P.U.R.: Pull-Up Resistor

(12) Type M – C

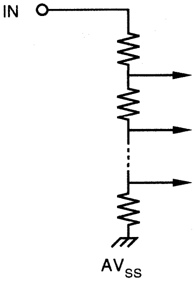


P.U.R.: Pull-Up Resistor

(13) Type Y



(14) Type Z



## 2.4 Processing of Unused Pins

### Table 2.4–1 Unused Pins Lists

| Pin | Recommended connection |
|---|---|
| P00 / INT4 | Connect to $V_{SS}$. |
| P01 / $\overline{SCK}$ | Connect to $V_{SS}$ or $V_{DD}$. |
| P02 / SO /SB0 | |
| P03 / SI / SB1 | |
| P10 / INTO–P12 / INT2 | Connect to $V_{SS}$. |
| P13 / TI0 | |
| P20 / PTO0 | Input mode: Connect to $V_{SS}$ or $V_{DD}$.<br>Output mode: Do not connect. |
| P21 | |
| P22 / PCL | |
| P23 / BUZ | |
| P30–P33 | |
| P40–P43 | |
| P50–P53 | |
| P60–P63 | |
| P70–P73 | |
| P80–P83 | |
| S12–S23 | Do not connect. |
| S24 / BP0–S31 / BP7 | |
| COM0–COM3 | |
| $V_{LC0}$–$V_{LC3}$ | Connect to $V_{SS}$. |
| BIAS | Only when all of $V_{LC0}$–$V_{LC2}$ are unused, connect BIAS to $V_{SS}$; otherwise, do not connect. |
| XT1 | Connect XT1 to $V_{SS}$ or $V_{DD}$. |
| XT2 | Do not connect XT2. |

## 2.5 Mask Option Selection

The pins contain the mask option function as listed in Table 2.5-1.
However, mask option is not contained in the μPD75P328.

### Table 2.5–1 Mask option selection

| Pin names | Mask option |
|---|---|
| P40–P43,<br>P50–P53 | . Pull-up resistor is included. specified bit-wise.<br>. Pull-up resistor is not included. |
| Split resistor for LCD drive power supply | . Split resistor is included.<br>. Split resistor is not included. |

## 3. ARCHITECTURE AND MEMORY MAP

The μCOM-75X architecture for the μPD75328 adopts data memory bank configuration and memory mapped I / O to provide features such as:

- Internal RAM with maximum of 4K words x four bits (12-bit addresses)
- Peripheral hardware extensibility

This chapter covers these topics.

### 3.1 Data Memory Bank Configuration and Addressing Modes

3.1.1 Data memory bank configuration

A general purpose static RAM (492 words x four bits) is incorporated in data memory space addresses 000H to 1DFH, and display data memory (20 words x four bits) is incorporated in adresses 1EOH to 1FFH. Peripheral hardware such as input/output ports and timers is allocated to addresses F80H to FFFH.

To adress the data memory space (4K words x four bits) with 12-bit addresses, the μPD75328 adopts a memory bank configuration where the low-order eight bits of an address are specified either directly or indirectly by using an instruction, and the high-order four bits of an address are specified by using a memory bank.

To specify the memory bank (MB), two hardware devices are incorporated in the μPD75328:

- Memory bank enable flag (MBE)
- Memory bank selection register (MBS)

MBS is a register used to select a memory bank. The μPD75328 allows 0, 1, or 15 to be set in MBS. MBE is a flag used to determine whether or not the memory bank selected by using MBS is validated. As shown in Fig. 3.1–1, when MBE is set to 0, the specified memory bank is fixed regardless of how MBS is set; when MBE is set to 1, memory bank switching can be performed to extend data memory space by setting MBS.

In data memory space addressing, normally MBE is set to 1, and the data memory area of the memory bank specified by using MBS is handled. An efficient program can be prepared by using MBE = 0 mode and MBE = 1 mode appropriately in each process of the program.

| | Applicable program processing | Effects |
|---|---|---|
| MBE = 0 mode | ° Interrupt service | MBS save and restore are made unnecessary. |
| | ° Repetitive processing of internal hardware operation and general purpose RAM operation | MBS change is made unecessary. |
| | ° Subroutine processing | MBS save and restore are made unnecessary. |
| MBE = 1 mode | ° Normal program processing | |

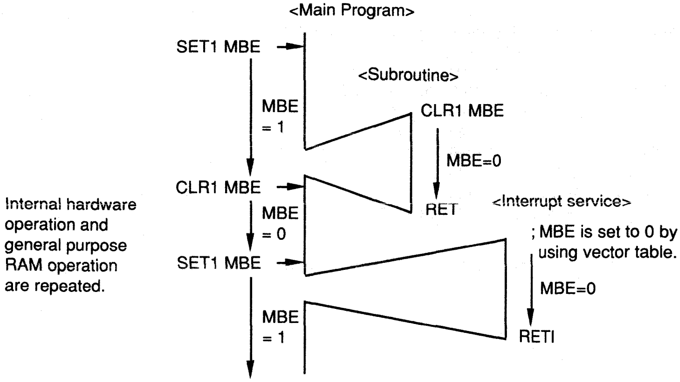

<Main Program>
SET1 MBE
<Subroutine>
MBE = 1
CLR1 MBE
MBE=0
CLR1 MBE
RET
<Interrupt service>
Internal hardware operation and general purpose RAM operation are repeated.
MBE = 0
; MBE is set to 0 by using vector table.
SET1 MBE
MBE=0
MBE = 1
RETI

**Figure 3.1–1: Proper Use of MBE = 0 and MBE = 1 Modes**

MBE is saved or restored automatically during subroutine processing and can be changed as desired. When interrupt service is made, MBE is automatically saved or restored. MBE during interrupt service can also be specified at the same time interrupt service starts by setting the interrupt vector table. Thus, it is useful for high-speed interrupt service.

To change the MBS setting in subroutine processing or interrupt service, MBS is saved and restored by using PUSH and POP instructions.

MBE is set and reset by using SET1 and CLR1 instructions.
MBS is set by using the SEL instruction.

Example 1: To clear MBE and fix memory bank

```
            CLR1    MBE  ; MBE ← 0
```

Example 2: To select memory bank 1

```
            SET1    MBE  ; MBE ← 1
            SEL     MB1  ; MBS ← 1
```

3.1.2 Data memory addressing modes

The μCOM-75X architecture adopted for the μPD75328 provides seven addressing modes, as shown in Fig. 3.1–2, for efficiently addressing the data memory space for each bit length of data to be processed.

(1) 1-bit direct adressing (mem. bit)

This addressing mode directly specifies each bit in all the data memory space by using the instruction operand.

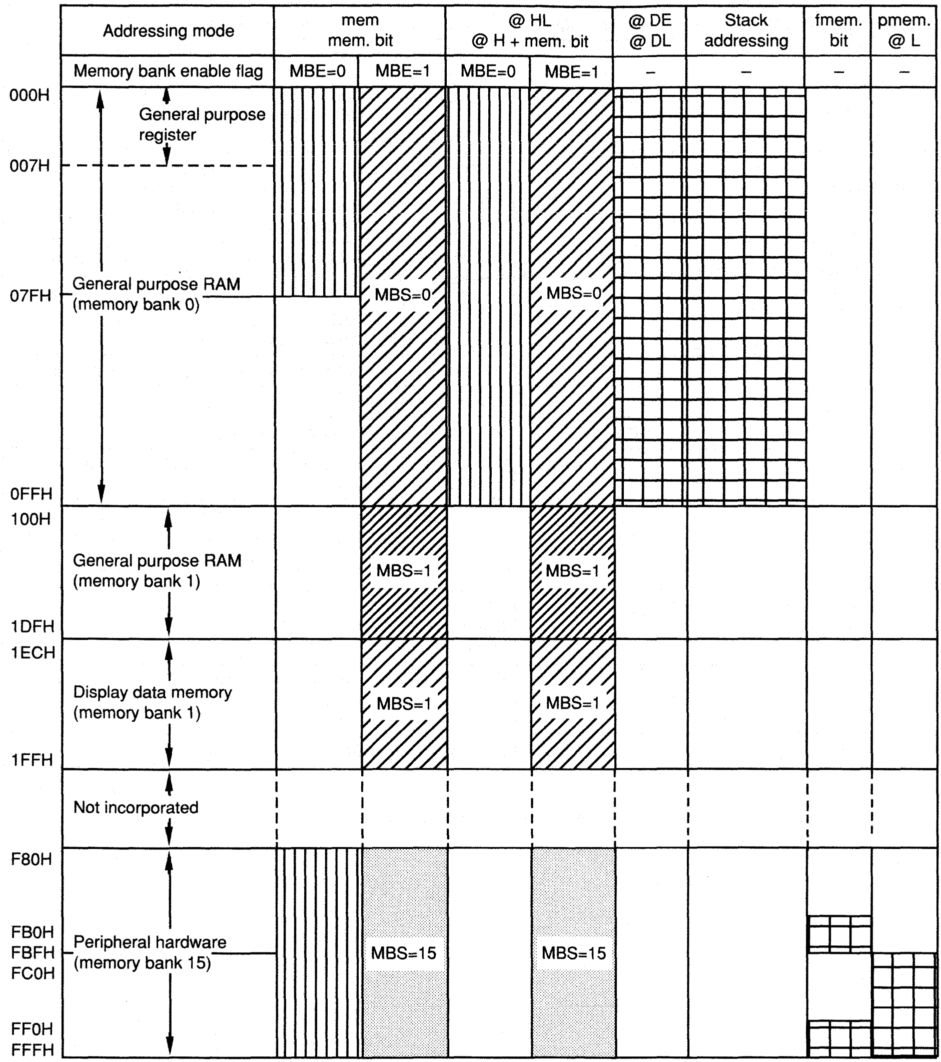
When MBE is set to 0, the specified memory bank (MB) is fixed to memory bank 0 (MB0) if the address specified in the operand is 00H-7FH; it is fixed to memory bank 15 (MB15) if the address specified in the operand is 80H-FFH. Thus, when MBE is set to 0, both the general purpose RAM area (000H-07FH) and peripheral hardware area (F80H-FFFH) can be addressed.

When MBE is set to 1, any memory bank can be specified by setting MBS; data memory space that can be specified can be extended.

The 1-bit direct addressing mode is applicable to bit set and reset instructions (SET1 and CLR1) and bit test instructions (SKT and SKF).

Example:
To set FLAG1, reset FLAG2, and test whether or not FLAG3 is set to 0.

```
FLAG1   EQU    03FH. 1  ;  3FH address bit 1
FLAG2   EQU    087H. 2  ;  87H address bit 2
FLAG3   EQU    0A7H. 0  ;  A7H address bit 0
        SET1   MBE      ;  MBE    ← 1
        SEL    MB0      ;  MBS    ← 0
        SET1   FLAG1    ;  FLAG1 ← 1
        CLR1   FLAG2    ;  FLAG2 ← 0
        SKF    FLAG3    ;  FLAG3  =  0 ?
```



–: don`t care

**Figure 3.1–2 Data Memory Configuration and Addressing Range of each Addressing Mode**

| Addressing mode | Representation format | Specified address |
|---|---|---|
| 1-bit direct addressing | mem. bit | Bit indicated by a bit at the address indicated by MB and mem. However,<br>{ MBE = 0, MB = 0 when mem = 00H-7FH<br>MB = 15 when mem = 80H-FFH<br>When MBE =1, MB = MBS |
| 4-bit direct addressing | mem | Address indicated by MB and mem. However,<br>{ MBE = 0, MB = 0 when mem = 00H-7FH<br>MB = 15 when mem = 80H-FH<br>When MBE = 1, MB = MBS |
| 8-bit direct addressing | | Address indicated by MB and mem (mem is an even address). However,<br>{ MBE = 0, MB = 0 when mem = 00H-7FH<br>MB = 15 when mem = 80H-FFH<br>When MBE = 1, MB = MBS |
| 4-bit register indirect addressing | @ HL | Address indicated by MB and HL. However, MB = MBE • MBS |
| | @ DE | Memory bank 0 address indicated by DE |
| | @ DL | Memory bank 0 address indicated by DL |
| 8-bit register indirect addressing | @ HL | Address indicated by MB and HL (the L register contains an even number). However, MB = MBE • MBS |
| Bit manipulation addressing | fmem. bit | Bit indicated by a bit at the address indicated by fmem.<br>However, fmem { FB0H-FBFH (hardware related to interrupt)<br>FF0H-FFFH (I/O port) |
| | pmem. @ L | Bit indicated by the low-order two bits of the L register at the address indicated by the high-order 10 bits of pmem and the high-order two bits of the L register.<br>However, pmem = FC0H-FFFH |
| | @ H + mem. bit | Bit indicated by a bit at the address indicated by MB, H, and the low-order four bits of mem.<br>However, MB = MBE • MBS |
| Stack addressing | | Memory bank 0 address indicated by SP. |

**Figure 3.1–3 Addressing Modes**

(2) 4-bit direct addressing (mem)

This addressing mode directly specifies all the data memory space in 4-bit units by using the instruction operand.

As with the 1-bit direct addressing mode, when MBE is set to 0, the 4-bit direct address mode allows that only the general purpose RAM area (000H-07FH) and peripherial hardware area (F80H-FFFH) can be specified. When MBE is set to 1, any memory bank (MB) can be specified by setting MBS and the data memory space that can be specified is extended to all the space.

The 4-bit direct addressing mode is applicable to the MOV, XCH, INCS, IN, and OUT instructions.

Example 1: To input port 4 and store in "DATA1".

```
DATA1 EQU    5FH          ;   "DATA1" is address 5HF.
      CLR1   MBE          ;   MBE← 0
      IN     A, PORT4     ;   A← PORT4
      MOV    DATA1, A     ;   (DATA1)← A
```

Example 2: To output data in "BUFF" to port 5.

```
BUFF  EQU    11AH         ;   "BUFF" is address 11AH
      SET1   MBE          ;   MBE← 1
      SEL    MB1          ;   MBS← 1
      MOV    A, BUFF      ;   A← (BUFF)
      SEL    MB15         ;   MBS← 15
      OUT    PORT5, A     ;   PORT5← A
```

Caution:

If data related to the input /output ports is stored in general purpose RAM of bank 1 as in this example, the program is made less efficient. If data related to the input /output ports is stored in addresses 00H-7FH of bank 0, a program can be prepared without changing MBS as in Example 1.

(3) 8-bit direct addressing (mem)

This addressing mode directly specifies all the data memory space in 8 bit units by using the instruction operand. Only even address can be specified in the mem operand. The 4-bit data at the address specified in the operand and the 4-bit data at the address + 1 are paired and transferred to the 8-bit accumulator (XA register pair) for 8-bit processing.

The memory banks specified in the addressing mode are the same as in the 4-bit direct addressing mode.

The 8-bit direct addressing mode is applicable to the MOV, XCH, IN, and OUT instructions.

Example 1: To transfer 8-bit data in ports 4 and 5 to addresses 20H and 21H.

```
DATA  EQU    020H
      CLR1   MBE          ;   MBE← 0
      IN     XA, PORT4    ;   X← port 5, A← port 4
      MOV    DATA, XA     ;   (21H)← X, (20H)← A
```

Example 2: To read 8-bit data input to the serial interface shift register (SIO), set transfer data, and start of transfer.

```
      SEL    MB15         ;   MBS← 15
      XCH    XA, SIO      ;   XA↔ (SIO)
```

(4) 4-bit register indirect addressing (@rpa)

This addressing mode indirectly specifies the data memory space in 4-bit units by using the data pointer (general purpose register pair) specified in the instruction operand.

The data pointers are the HL register pair, which enables all the data memory space to be specified according to MB = MBE • MBS specification, and the DE and DL register pairs, which always indicate memory bank 0 regardless of how MBE and MBS are set. Efficient programs can be prepared by using the data memory bank to be used.

Example: To transfer data at 50H-57H to 110H-117H.

```
DATA1  EQU   57H
DATA2  EQU   117H
       SET1  MBE
       SEL   MB1
       MOV   D, #DATA1 SHR 4
       MOV   HL, #DATA2 AND 0FFH ;  HL← 17H
LOOP:  MOV   A, @DL              ;  A← (DL)
       XCH   A, @HL              ;  A← (HL)
       DECS  L                   ;  L← L–1
       BR    LOOP
```

The addressing mode using the HL register pair for the data pointer has wide applications such as data transfer, operation, comparison, and input / output. The addressing mode using the DE or DL register pair is applicable to the MOV and XCH instructions.

As shown in Fig. 3.1–4, data memory space addresses can be updated as desired by combining the addressing mode with general purpose register (or register pair) increment and decrement instructions.

Example 1: To compare data at 50H-57H with data at 110H-117H.

```
DATA1  EQU   57H
DATA2  EQU   117H
       SET1  MBE
       SEL   MB1
       MOV   D, #DATA1 SHR 4
       MOV   HL, #DATA2 AND 0FFH
LOOP:  MOV   A, @DL
       SKE   A, @HL    ;  A = (HL) ?
       BR    NO        ;  NO
       DECS  L         ;  YES, L← L–1
       BR    LOOP
```

Example 2: To clear data memory area 04H-FFH.

```
       SEL   MB0
       MOV   XA, #00H
       MOV   L, #04H
LOOP:  MOV   @HL, A    ;  (HL)← A
       INCS  HL        ;  L← L+1
       BR    LOOP
       INCS  H         ;  H← H+1
       BR    LOOP
```

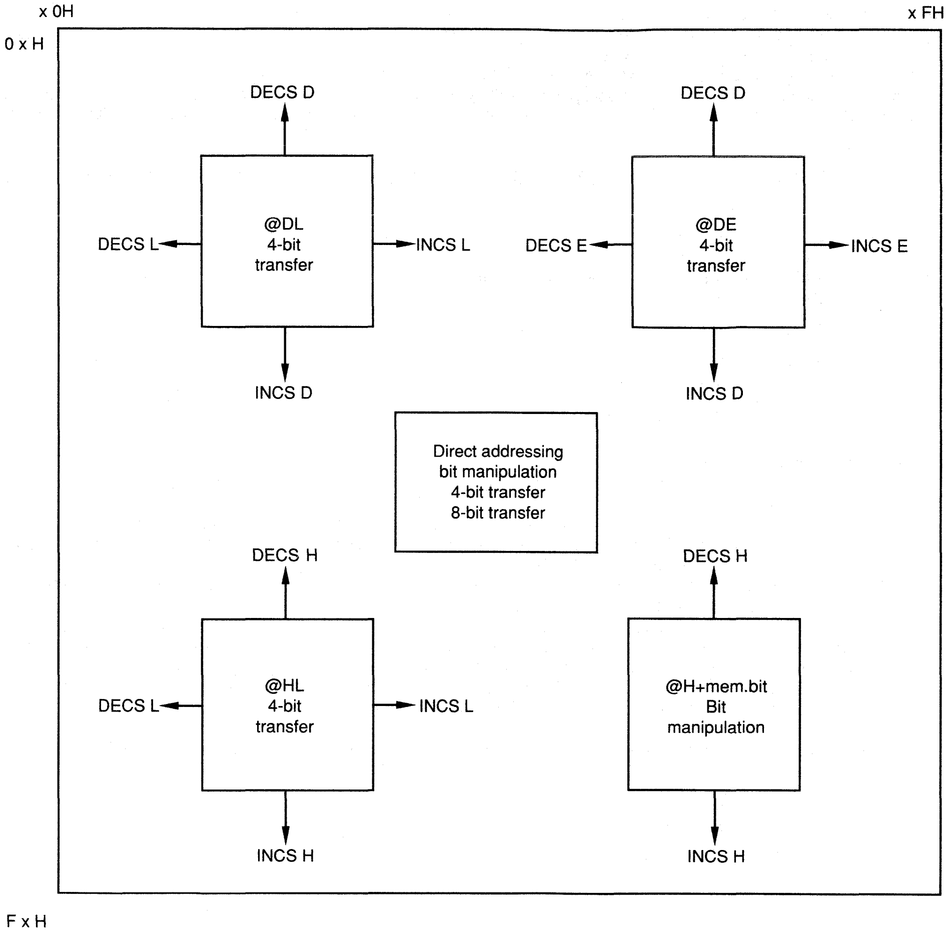


**Figure 3.1–4 General Purpose RAM Address Update Method**

(5) 8-bit register indirect addressing (@HL)

This addressing mode indirectly specifies all the data memory space in 8-bit units by using the data pointer (HL register pair). The 4-bit data at the address setting data pointer bit 0 (L register bit 0) to 0 and the 4-bit data at the address + 1 are paired and transferred to the 8-bit accumulator (XA register) for 8-bit processing.

The memory banks specified in the addressing mode are the same as those when the HL register is specified in the 4-bit register indirect addressing mode (MB = MBE • MBS). The 8-bit register indirect addressing mode is applicable to the MOV, XCH, and SKE Instructions.

Example 1: To compare the count register (T0) value of timer / event counter 0 with data at addresses 30H and 31H for equality.

```
DATA EQU    30H
     CLR1   MBE
     MOV    HL, #DATA
     MOV    XA, T0      ;   XA←— count register 0
     SKE    A, @HL      ;   A = (HL) ?
     BR     NO
     INCS   L
     MOV    A, X        ;   A←— X
     SKE    A, @HL      ;   A = (HL) ?
```

(6) Bit manipulation addressing

This addressing mode is used to perform bit manipulations such as Boolean operation or bit transfer on any bit in all of the data memory space.

Although the 1-bit direct addressing mode is applicable only to the bit set, reset, and test instructions, the bit manipulation addressing mode enables bit manipulations such as Boolean operations by using the AND1, OR1, and XOR1 instructions and allows bit test and bit reset by using the SKTCLR instructions.

The following three types of bit manipulation addressing modes can be used according to the data memory address to be used:

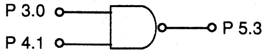
(a) Specific address bit direct addressing (fmem. bit).

This addressing mode enables, peripheral hardware that frequently uses bit manipulation, such as input / output ports and interrupt flags, to be operated at any time independently of the memory bank setting. Thus, the addressing mode is applicable to data memory addresses FF0H-FFFH (where input / output ports are mapped) and FB0H-FBFH (where hardware related to interrupts is mapped). For the hardware of the two data memory areas, bit manipulation can be performed in direct addressing at any time as desired regardless of how MBS and MBE are set.

Example 1: To test the timer 0 interrupt request flag (IRQT0); if the request flag is set, clear the flag and reset P63.

```
SKTCLR   IRQT0      ;   IRQT0 = 1 ?
BR       NO         ;   NO
CLR1     PORT6.3    ;   YES
```

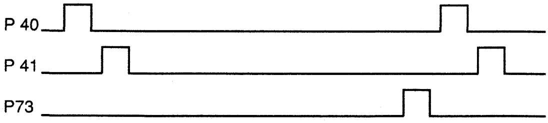
Example 2: To reset P53 if both P30 and P41 are set to 1.



```
(i)       SET1    CY              ;    CY ← 1
          AND1    CY, PORT3.0     ;    CY P3.0
          AND1    CY, PORT4.1     ;    CY P4.1
          SKT     CY              ;    CY = 1 ?
          BR      SETP
          CLR1    PORT5.3         ;    P53 ← 1
            :
            :
 SETP:    SET1 PORT5.3            ;    P53 ← 1
            :
            :

(ii)      SKT     PORT3.0         ;    P30 = 1 ?
          BR      SETP
          SKT     PORT4.1         ;    P41 = 1 ?
          BR      SETP
          CLR1    PORT5.3         ;    P53 ← 0
            :
            :
STEP:     SET1    PORT5.3         ;    P53 ← 1
```

(b) Specific address bit register indirect addressing (pmem. @L).
This addressing mode indirectly specifies each bit of peripheral hardware, such as an input / output port, by using the L register for successive operation. The addressing mode is applicable to data memory addresses FCOH-FFFH.
The addressing mode directly specifies the high-order 10 bits of a 12-bit data memory address in the pmem operand and indirectly specifies the low-order 2-bit data memory address part and the bit address by using the L register. Thus, 16 bits (four ports) can be manipulated (operated) successively by L register specification.
The address mode also enable bit manipulation to be performed at any time independently of how MBE and MBS are set.

Example: To output a pulse to each bit of ports 4 to 7 in order.



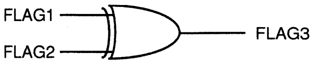
```
          MOV     L, #0
LOOP:     SET1    PORT4. @L       ;    Port 4–7 bit ($L_{1-0}$) ← 1
          CLR1    PORT4. @L       ;    Port 4–7 bit ($L_{1-0}$) ← 0
          INCS    L
          BR      LOOP
```

(c) Special 1-bit direct addressing (@H + mem. bit)

This addressing mode enables bit manipulation to be performed on any bit in all of the data memory space.

The addressing mode indirectly specifies the high-order four bits of data memory address of the memory bank specified by MB = MBE MBS by using the H register and directly specifies the low-order 4-bit data memory address part and the bit address in the operands. It enable various types of bit manipulation to be performed on any bit in all of the data memory space.

Example: To reset address 32H bit 2 (FLAG3) if both address 30H bit 3 (FLAG1) and address 31H bit 0 (FLAG2) are set to 0 or 1.



```
FLAG1  EQU    30H.3
FLAG2  EQU    31H.0
FLAG3  EQU    32H.2
       SEL    MB0
       MOV    H, #FLAG1 SHR 6
       CLR1   CY
       OR1    CY, @H + FLAG1   ;   CY← 0
       XOR1   CY, @H + FLAG2   ;   CY← CY-FLAG1
       SET1   @H + FLAG3       ;   CY← CY-FLAG2
       SKT    CY               ;   CY = 1 ?
       CLR1   @H + FLAG3       ;   FLAG← 30
```

(7) Stack addressing

The stack addressing mode is used for register save and restore during interrupt service or subroutine processing.

The processing mode specifies an address by using the 8-bit stack pointer (data memory bank 0).

The addressing mode can also be used for register save and restore by executing the PUSH and POP instructions.

Example 1: To save and restore register in subroutine processing.

```
SUB: PUSH   XA
     PUSH   HL
     PUSH   BS    ;   MBS is saved.
       :
       :
     POP    BS
     POP    HL
     POP    XA
     RET
```

Example 2: To transfer the HL register pair contents to the DE register pair.

```
     PUSH   HL
     POP    DE    ;   DE← HL
```

Example 3: To branch to the address indicated by [XABC] register.

```
     PUSH   BC
     PUSH   XA
     RET          ;   Branch to XABC address
```

### 3.2 Memory Mapped I / O

The μPD75328 adopts memory mapped I / O where peripheral hardware such as input / ports and timers is allocated to addresses F80H-FFFH of the data memory space as shown in Fig. 3.1–2. Thus, peripheral hardware is controlled entirely by memory operation instructions rather than special instructions. (To easily understand programs, some hardware control mnemonics are provided.)
Table 3.2–1 lists the addressing modes that can be used to operate peripheral hardware.
Memory bank 1 is specified to operate the display data memory mapped in addresses 1ECH-1FFH.

**Table 3.2–1 Applicable Addressing Modes during Peripheral Hardware Operation**

| | Applicable addressing mode | Applicable hardware |
|---|---|---|
| Bit manipulation | With MBE=0 or (MBE=1 and MBS=15), direct addressing (specification in mem. bit). | All hardware where bit manipulation can be performed. |
| | Direct addressing regardless of how MBE and MBS are set. (specification in fmem. bit) | ISTO, MBE<br>IExxx, IRQxxx, PORTn.x |
| | Indirect addressing regardless of how MBE and MBS are set. (specification in pmem. @L) | BSBn.x<br>PORTn.x |
| 4-bit manipulation | With MBE=0 or (MBE=1 and MBS=15), direct addressing (specification in mem). | All hardware where 4-bit manipulation can be performed. |
| | With (MBE=1 and MBS=15), register indirect addressing (specification in @HL). | |
| 8-bit manipulation | With MBE=0 or (MBE=1 and MBS=15), direct addressing (specification in mem), mem must be an even address. | All hardware where 8-bit manipulation can be performed. |
| | With MBE=1 and MBS=15, register indirect addressing (specification in @HL; the L register must contain an even number. | |

```
Example: CLR1     MBE        ;  MBE = 0
         SET1     TM0.3      ;  Timer 0 starts.
         EI       IE0        ;  INT0 is enabled.
         DI       IE1        ;  INT1 is disabled.
         SKTCLR   IRQ2       ;  INT2 request flag is tested and cleared.
         SET      PORT4. @L  ;  Port 4 is set.
         IN       A, PORT0   ;  A ← port 0
         OUT      PORT4, XA  ;  Port 5, 4 ← XA
```

Figs. 3.2–1 to 3.2–3 shows the μPD75328 I/O map.
The columns in the figures mean:

- Abbreviation:
  Name indicating internal hardware address. It can be entered in the instruction operand field.

- R/W
  Indicates whether the hardware device can be read or written.
  – R/W : Read and write are enabled.
  – R : Read only is enabled.
  – W : Write only is enabled.

- Number of bits that can be manipulated:
  Indicates the number of bits that can be processed when the hardware device is operated.
  O: Bit manipulation is enabled in 1-4, 4-, or 8-bit units as specified in the column.
  Δ: Only some bits can be manipulated. See Remarks for the bits that can be manipulated.
  –: Bit manipulation cannot be performed in 1-, 4-, or 8-bit units as specified in the column.

- Bit manipulation addressing:
  Indicates the applicable bit manipulation addressing for performing bit manipulation on the hardware device.

| Address | Hardware name (abbreviation) | | | | R/W | Number of bits that can be manipulated | | | Bit manipulation addressing | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| | b3 | b2 | b1 | b0 | | One bit | Four bit | Eight bit | | |
| F80H | Stack pointer (SP) | | | | R/W | – | – | O | | Bit 0 is fixed to 0. |
| | | | | | | | | | | |
| F85H | Basic interval timer mode register (BTM) | | | | W | Δ | O | – | mem. bit | Bit manipulation can be performed only on bit 3. |
| F86 H | Basic interval timer (BT) | | | | R | – | – | O | | |
| | | | | | | | | | | |
| F8CH | Display mode register (LCDM) | | | | W | Δ | – | O | mem. bit | Bit manipulation can be performed only on bit 3. |
| | | | | | | – | – | | | |
| F8EH | Display control register (LCDC) | | | | W | – | O | – | | |
| F98H | Watch mode register (WM) | | | | R/W | Δ | – | O | mem. bit | Bit test can be made only on bit 3. |
| | | | | | W | – | – | | | |
| FA0H | Timer/event counter 0 mode register (TM0) | | | | W | Δ | – | O | mem. bit | Bit manipulation can be performed only on bit 3. |
| | | | | | | – | – | | | |
| FA2H | TOE0 (Note) | | | | W | O | – | – | mem. bit | |
| FA4H | Timer/event counter 0 count register (T0) | | | | R | – | – | O | | |
| | | | | | | – | – | | | |
| FA6H | Timer/event counter 0 modulo register (TMOD0) | | | | W | – | – | O | | |
| | | | | | | – | – | | | |
| FB0H | 0 | IST0 | MBE | 0 | R/W | O | O | O | fmem. bit | |
| | Program status word (PSW) | | | | R | – | – | | | |
| FB2H | (IME) | | | | – | – | – | – | | EI and DI instructions are used |
| FB3H | Processor clock control register (PCC) | | | | W | – | O | | | |
| FB4H | INT0 mode register (IM0) | | | | W | – | O | – | | Bit 2 is fixed to 0. |
| FB5H | INT1 mode register (IM1) | | | | W | – | O | | | Bit 3 to 1 are fixed to 0. |
| FB6H | INT2 mode register (IM2) | | | | W | – | O | | | Bits 3 and 2 are fixed to 0. |
| FB7H | System clock control register (SCC) | | | | W | O | – | – | | only buts 3 and 0 can be bit manipulated |

Note: TOE0 = timer / event counter 0 output enable flag (W)

(to be continued)

**Figure 3.2–1 µPD75328 I/O Map**

| Address | Hardware name (abbreviation) | | | | R/W | Number of bits that can be manipulated | | | Bit manipulation addressing | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| | b3 | b2 | b1 | b0 | | One bit | Four bit | Eight bit | | |
| FB8H | IE4 | IRQ4 | IEBT | IRQBT | R/W | O | O | – | fmem. bit | |
| FBAH | | | IEW | IRQW | R/W | O | O | | | |
| FBCH | | | IET0 | IRQT0 | R/W | O | O | – | | |
| FBDH | | | IECSI | IRQCSI | R/W | O | O | | | |
| FBEH | IE1 | IRQ1 | IE0 | IRQ0 | R/W | O | O | – | | |
| FBFH | | | IE2 | IRQ2 | R/W | O | O | | | |
| FC0H | Bit sequential buffer 0 (BSB0) | | | | R/W | O | O | O | mem. bit<br>pmem @L | |
| FC1H | Bit sequential buffer 1 (BSB1) | | | | R/W | O | O | | | |
| FC2H | Bit sequential buffer 2 (BSB2) | | | | R/W | O | O | O | | |
| FC3H | Bit sequential buffer 3 (BSB3) | | | | R/W | O | O | | | |
| FD0H | Clock output mode register (CLOM) | | | | W | – | O | – | | |
| FD8H | SOC | EOC | | | R/W | Δ | – | O | mem,bit | EOC … R, SOC … W |
| | A/D convert mode register (ADM) | | | | W | – | – | | | |
| FDAH | SA register (SA) | | | | R | – | – | O | | |
| | | | | | | – | – | | | |
| FDCH | Pull-up resistor specification register group A (POGA) | | | | W | – | – | O | | |
| FDEH | Pull-up resistor specification register group B (POGB) | | | | W | - | - | O | | |
| FE0H | Serial operation mode register (CSIM) | | | | W | – | – | O | | |
| | CSIE | COI | WUP | | R/W | O | O | | mem. bit | |
| FE2H | CMDD | RELD | CMDT | RELT | R/W | O | – | – | mem. bit | Only bit manipulation can be performed on any bit. |
| | SBI control register (SBIC) | | | | | | | | | |
| | BSYE | ACKD | ACKE | ACKT | | | | | | |
| FE4H | Serial I/O shift register (SIO) | | | | R/W | – | – | O | | |
| FE6H | Slave address register (SVA) | | | | W | – | – | O | | |
| FE8H | PM33 | PM32 | PM31 | PM30 | W | – | – | O | | |
| | Port mode register group A (PMGA) | | | | | | | | | |
| | PM63 | PM62 | PM61 | PM60 | | | | | | |
| FECH | – | PM2 | – | – | W | – | – | O | | |
| | Port mode register group B (PMGB) | | | | | | | | | |
| | PM7 | 0 | PM5 | PM4 | | | | | | |

Remarks: 1. IExxx is an interrupt enable flag.
2. IRQxxx is an interrupt request flag.
3. IME is an interrupt master flag.

(to be continued)

**Figure 3.2–1 µPD75328 I/O Map** (con`t)

| Address | Hardware name (abbreviation) | | | | R/W | Number of bits that can be manipulated | | | Bit manipulation addressing | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| | b3 | b2 | b1 | b0 | | One bit | Four bit | Eight bit | | |
| FEEH | Port mode register group C (PMGC) | | | | W | – | – | O | | |
| | | | | | | – | – | | | |
| FF0H | Port 0 (PORT 0) | | | | R | O | O | – | fmem. bit<br>pmem. @L | |
| FF1H | Port 1 (PORT 1) | | | | R | O | O | | | |
| FF2H | Port 2 (PORT 2) | | | | R/W | O | O | – | | |
| FF3H | Port 3 (PORT 3) | | | | R/W | O | O | | | |
| FF4H | Port 4 (PORT 4) | | | | R/W | O | O | O | | |
| FF5H | Port 5 (PORT 5) | | | | R/W | O | O | | | |
| FF6H (Note) | KR3 | KR2 | KR1 | KR0 | R/W | O | O | O | | |
| | Port 6 (PORT 6) | | | | | | | | | |
| FF7H (Note) | KR7 | KR6 | KR5 | KR4 | R/W | O | O | | | |
| | Port 7 (PORT 7) | | | | | | | | | |
| FF8H | Port 8 (PORT8) | | | | R/W | O | O | - | | |

Note: KR0–KR7 can only be read. When 4bit parallel is input, specified at PORT6 or PORT7.

**Figure 3.2–1 µPD75328 I/O Map** (con`t)

# 4 INTERNAL CPU FUNCTIONS

## 4.1 Program Counter (PC) – 13–Bit (µPD75328)

The program counter is binary counter which holds program memory address information. The µPD75328 is configured of 13-bit (See Fig. 4.4–1).

µPD75328 format

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

**Figure 4.1–1 Program Counter Format**

Each time one instruction is executed, normally the program counter is automatically incremented according to the number of the bytes in the instruction.
When a branch instruction (BR or BRCB) is executed, the register pair contents or immediate data indicating the branch destination address is loaded into all or some of the PC bits.

When a subroutine call instruction (CALL or CALLF) is executed or a vectored interrupt occurs, the current PC contents (return address already incremented to fetch the next instruction) are saved in stack memory (data memory indicated by the stack pointer), then the jump destination address is loaded.
When a return instruction (RET, RETS, or RETI) is executed, the stack memory contents are loaded into the PC.
When the $\overline{\text{RESET}}$ signal is generated, the program memory contents are loaded into the program counter for initialization, as follows: (Program can be started at any desired address.)

|  |  |  |
|---|---|---|
| μPD75328: | PC12–PC8 ← | low-order five bits at address 0000H |
|  | PC7 –PC0 ← | eight bits at address 0001H |

### 4.2 Program Memory (ROM) - 8064 Words x Eight bit (μPD75328)

The program memory stores programs, interrupt vector table, GETI instruction look up table, and data such as table data. The program memory of the μPD75328, is mask programmable ROM.
Fig. 4.2–1 to 4.2–5 show the program memory map.
The program memory is addressed by using the program counter. Table data can also be referenced by using the table reference instruction (MOVT).
The address range for branching by a branch or subroutine call instruction is as shown in Fig. 4.2–1. When a relative branch instruction (BR $addr) is used, a branch can be made to [PC contents – 15 to –1, +2 to + 16] address independently of block.
The program memory addresses are shown below.

- 0000H–1F7FH : μPD75328

The following addresses are assigned for special purpose:
(All the area except address 0000H to 0001H Note 1 can be used as normal program memory area.).

- 0000H – 0001H
  Vector address table in which the program start address and MBE setup value are entered when the system is reset. (Reset start can be made at any desired address.)

- 0002H – 000BH
  Vector address table in which the program start address and MBE setup value are entered for each vectored interrupt. (Interrupt service can be started at any desired address.)

- 0020H – 007FH Table area referenced by GETI (Note 2).
  Note: 1. Their addresses are used in the μPD75328.
  2. The GETI instruction is used to convert any 2- or 3-byte instruction or any two 1-byte instructions into a 1-byte instruction. The number of program bytes can be reduced.

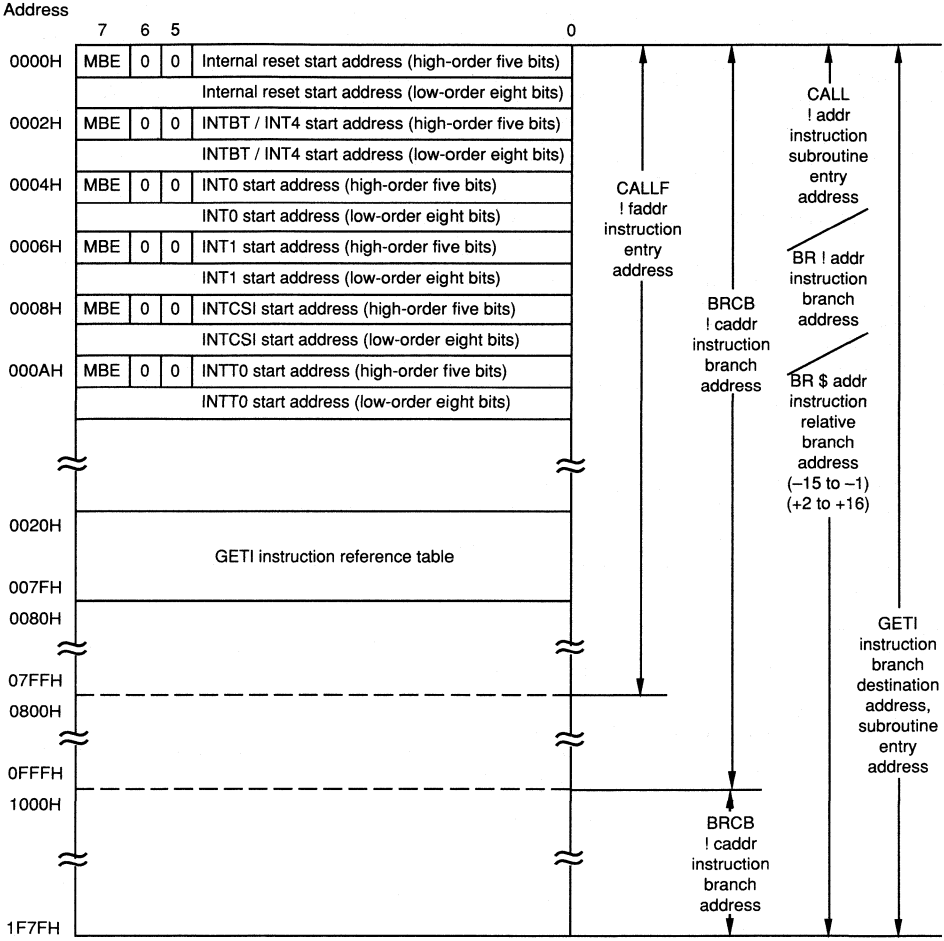

Address
7
6
5
0
0000H
MBE
0
0
Internal reset start address (high-order five bits)
Internal reset start address (low-order eight bits)
0002H
MBE
0
0
INTBT / INT4 start address (high-order five bits)
INTBT / INT4 start address (low-order eight bits)
0004H
MBE
0
0
INT0 start address (high-order five bits)
INT0 start address (low-order eight bits)
0006H
MBE
0
0
INT1 start address (high-order five bits)
INT1 start address (low-order eight bits)
0008H
MBE
0
0
INTCSI start address (high-order five bits)
INTCSI start address (low-order eight bits)
000AH
MBE
0
0
INTT0 start address (high-order five bits)
INTT0 start address (low-order eight bits)
0020H
GETI instruction reference table
007FH
0080H
07FFH
0800H
0FFFH
1000H
1F7FH
CALLF ! faddr instruction entry address
BRCB ! caddr instruction branch address
CALL ! addr instruction subroutine entry address
BR ! addr instruction branch address
BR $ addr instruction relative branch address (−15 to −1) (+2 to +16)
GETI instruction branch destination address, subroutine entry address
BRCB ! caddr instruction branch address

**Figure 4.2–1 Program Memory Map (μPD75328)**

### 4.3 Data Memory (RAM) – 512 Words x Four Bits

The data memory is a general purpose static RAM consisting of 512 words x four bits. Since a static RAM stores data during process, subroutine, or interrupt execution, it can also hold data when CPU operation is stopped in the standby mode; it is useful in that the memory contents can be held with battery power for hours.
Fig. 4.3–1 shows a μPD75328 data memory map.
The data memory adopts the bank configuration, consisting of banks 0 and 1 (each 256 words x four bits).
Peripheral hardware is mapped in the memory bank 15 area.
A memory bank is selected by setting the 4-bit memory bank selection register (MBS = 0, 1, or 15) when bank specification is enable by setting the memory bank enable flag (MBE) to 1 (MBE = 1). When bank specification is disabled (MBS = 0), memory bank 0 or 15 is automatically selected according to the current addressing mode. Each bank is addressed by using 8-bit immediate data, a register pair, etc.
Although a data memory word consists of four bits, the data memory can be handled in 1-, 4-, or 8-bit units by using various addressing modes.
See 3.1 for details of memory bank selection and addressing modes.
Specific areas of the data memory are also used for general purpose registers (bank 0; 000H – 0007H), stack memory (bank 0; 000H – 0FFH) and display data memory (bank 1; 1ECH – 1FFH). Data memory does not exist in bank 15. Peripheral hardware and various registers are mapped. Do not access addresses or bits to which no registers are allocated.

For the use of the specific areas of data memory, see the following:
- 4.4 for the general purpose register area
- 4.6 for the stack memory area
- 5.7.5 for the display data memory
- CHAPTER 5 for the peripheral hardware

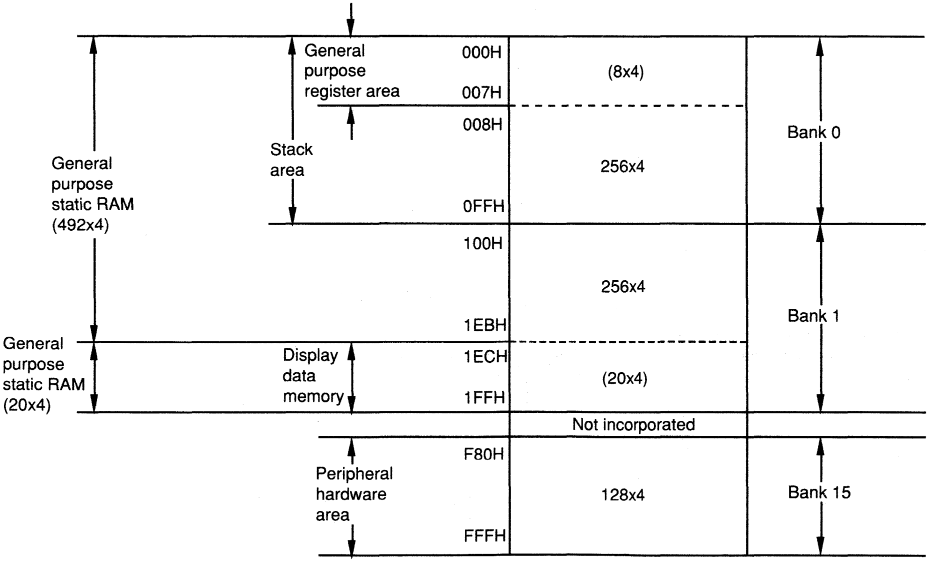


**Figure 4.3–1 Data Memory Map**

The data memory is undefined when the system is reset during normal operation mode (not standby), because of possible memory access at the time of reset. Be sure to initialize it too (clear RAM), normally at the program start.

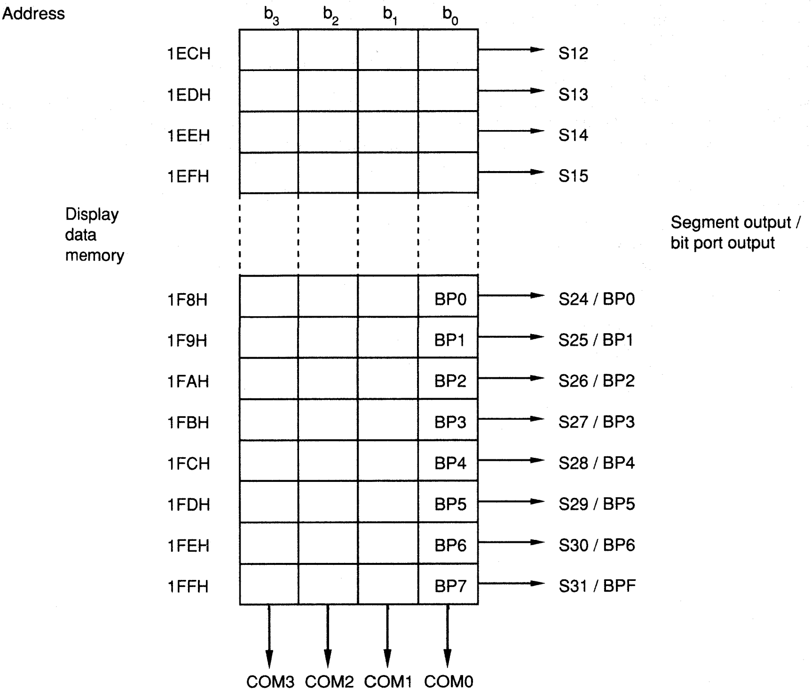


**Figure 4.3–2 Display Data Memory Configuration**

The display data memory is handled in 1- or 4-bit units.
Caution: The display data memory cannot be handled in 8-bit units.
Example: To clear 1ECH–1FFH display data memory.

```
         SET1   MBE
         SEL    MB1
         MOV    HL, #0ECH
         MOV    A, #00H
LOOP:    MOV    @HL, A       ;  Display data memory is cleared in4-bit units at a time
         INCS   L
         BR     LOOP
         INCS   H
         BR     LOOP
```

### 4.4 General Purpose Registers – Eight x Four Bits

The general purpose registers are eight 4-bit registers (B, C, D, E, H, L, X, and A) mapped in specific addresses of the data memory. Every general purpose register is handled in 4-bit units; register pairs BC, DE, HL, and XA are also used for 8-bit manipulation. In addition to DE and HL, registers D and L are also paired (DL), and the three register pairs can be used for data pointers. The general purpose register area can be addressed and accessed as normal RAM regardless of whether or not it is used for registers.

| | |
|---|---|
| X<br>01H | A<br>00H |
| H<br>03H | L<br>02H |
| D<br>05H | E<br>04H |
| B<br>07H | C<br>06H |

**Figure 4.4–1 General Purpose Register Configuration (When 4-bit processing is performed)**

| |
|---|
| XA<br>00H |
| HL<br>02H |
| DE<br>04H |
| BC<br>06H |

**Figure 4.4–2 General Purpose Register Configuration (When 8-bit processing is performed)**

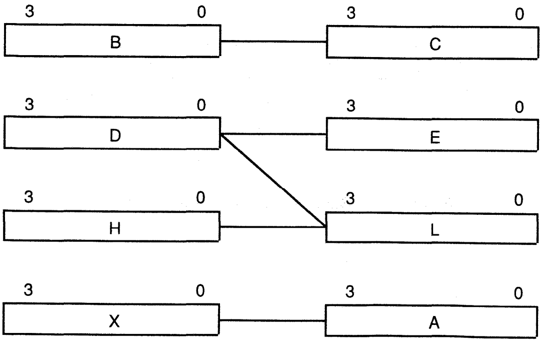


**Figure 4.4–3 Register Pair Configuration**

### 4.5 Accumulator

The µPD75328 uses the A register and XA register pair for accumulators. The A register is used as the main register during execution of 4-bit data processing instructions; the XA register pair is used as the main register pair during execution of 8-bit data processing instructions.

The carry flag (CY) is used for a bit accumulator during execution of bit manipulation instructions.

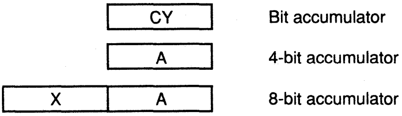


**Figure 4.5–1 Accumulator**

### 4.6 Stack Pointer (SP) – Eight Bits

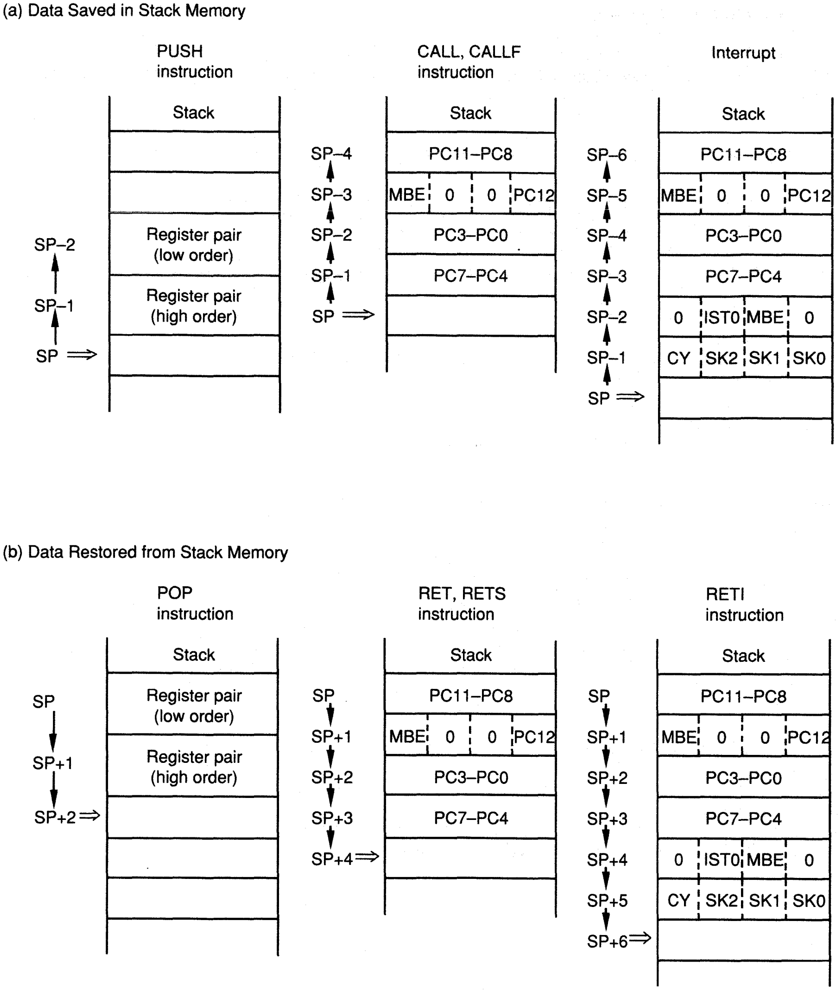
The µPD75328 uses general purpose RAM for stack memory (LIFO). The stack pointer (SP) is an 8-bit register which holds top address information of the stack area.
The stack area addresses are 000H–0FFH of memory bank 0 regardless of how MBE and MBS are set.
SP is decremented before data is saved in the stack memory (write operation); it is incremented after data is restored from the stack memory (read operation).
Figs. 4.6–2 to 4.6–4 shows data saved in and restored from the stack memory when stack operations are performed.
An initial value is set in SP by using an 8-bit memory operation instruction to determine the stack area to be used. The SP contents can also be read.
SP0 is always set to 0.
It is recommended that the initial value of SP should be set to 00H so that the stack area be used is starting at the most significant address of data memory bank 0 (0FFH).
When the RESET signal is generated, the SP contents become undefined. Be sure to initialize SP to the desired value at the start of the program.

Example: To initialize SP.

```
SEL MB15        ;   or CLR1 MBE
MOV XA, #00H
MOV SP, XA      ;   SP←00H
```

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| F80H | SP7 | SP6 | SP5 | SP4 | SP3 | SP2 | SP1 | SP0 | SP |
| | | | | | | | | Fixed to 0 | |

**Figure 4.6–1 Stack Pointer Configuration**



(a) Data Saved in Stack Memory

(b) Data Restored from Stack Memory

**Figure 4.6–3 Data Saved / Restored in Stack Operation μPD75328**

### 4.7 Program Status Word (PSW) – Eight Bits

The program status word (PSW) consists of flags closely related to processor operation.
PSW is mapped in data memory addresses FB0H and FB1H. Two bits of address FB0H can be operated by using a memory operation instruction.

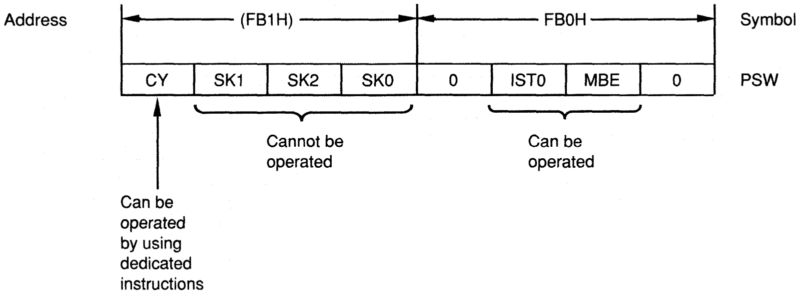


**Figure 4.7–1 Program Status Word Configuration**

**Table 4.7–1 PSW Flags Saved and Restored when Stack Operation is Performed**

| | | Saved or restored flag |
|---|---|---|
| Save | During CALL, CALLF instruction execution | MBE is saved |
| | During hardware interrupt | All PSW bits are saved |
| Restore | During RET, RETS instruction execution | MBE is restored |
| | During RETI instruction execution | All PSW bits are restored |

(1) Carry flag (CY)

The carry is a 1-bit flag that shows the occurence of overflow or underflow during execution of an instruction involving carry (ADDC or SUBC).

The carry flag also serves as a bit accumulator. Boolean algebra operation is performed between the bit accumulator and data memory specified by a bit address. The result can be stored in the bit accumulator.

The carry flag is operated by using dedicated instructions independently of other PSW bits.

When the $\overline{\text{RESET}}$ signal is generated, the carry flag becomes undefined.

**Table 4.7–2 Carry Flag Operation Instructions**

| | Instructions (mnemonic) | Carry flag processing |
|---|---|---|
| Instructions dedicated to carry flag operation | SET1 CY<br>CLR1 CY<br>NOT1 CY<br>SKT CY | CY is set to 1<br>CY is cleared<br>CY is inverted<br>Skip if CY is set to 1 |
| Bit Boolean instructions | AND1 CY, mem*.bit<br>OR1 CY, mem*.bit<br>XOR1 CY, mem*.bit | The specified bit and CY are ANDed, ORed, or XORed together |
| Interrupt service | During interrupt execution | CY is saved in stack memory in parallel with other PSW bits (eight bits) |
| | RETI | CY is restored from stack memory in parallel with other PSW bits |

Remarks: mem*.bit indicates any of the following three bit 3 manipulation addressing modes

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

Example: To AND bit 3 of address 3H and P33 together and set the result in CY.

```
SET1    CY              ;   CY←1
CLR1    MBE             ;   Or SEL MB0
SKT     3FH. 3          ;   Skip if bit 3 of address 3FH is set to 1.
CLR1    CY              ;   CY←0
AND1    CY, PORT3.3     ;   CY←CY ∧ P33
```

(2) Skip flags (SK2, SK1, and SK0)
The skip flags store the skip state and are automatically set or reset when the CPU executes instructions. The user cannot directly use the flags as operands.

(3) Interrupt status flag (IST0)
The interrupt status flag stores the current status of processing being performed. (For details, see Table 6.3–2.)

**Table 4.7–3 Interrupt Status Flag Indication Contents**

| IST0 | Status of processing being performed | Processing contents and interrupt control |
|---|---|---|
| 0 | Status 0 | During normal program processing.<br>Every interrupt can be acknowledged. |
| 1 | Status 1 | During interrupt processing.<br>No interrupt must be acknowledged. |

If an interrupt is acknowledged, the IST0 contents are saved in stack memory as a PSW bit, then automatically IST0 is set to 1. When an RETI instruction is executed, IST0 is set to 0.
The interrupt status flag can be operated by using a memory operation instruction. The current status of processing can also be changed under program control.

Caution: Before operating the flag, be sure to execute a DI instruction to disable interrupts. After operating the flag, execute an EI instruction to enable interrupts.

(4) Memory bank enable flag (MBE)

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

MBS can be set or reset at any time by using a bit manipulation instruction regardless of memory bank setting.

```
Example: SET1  MBE  ;  MBE← 1
         CLR1  MBE  ;  MBE← 0
```

When MBE is set to 1, the data memory address space is exceeded and all the data memory space can be addressed.

When MBE is reset to 0, the data memory address space is fixed regardless of how MBS is set. (see Fig. 3.1–2.)

When the $\overline{\text{RESET}}$ signal is generated, the contents of program memory address 0 bit 7 are set and MBE is initialized automatically.

When vectored interrupt service is made, bit 7 of the corresponding vector address table is set and the MBE state during interrupt service is set automatically.

During interrupt service, normally MBE is set to 0 and the general purpose RAM of memory bank 0 is used.

### 4.8 Bank Selection Register (BS)

The memory bank selection register (MBS) for selecting a memory bank is mapped in the bank selection register (BS). The low-order four bits of BS are fixed to 0

MBS is set by using the SEL MBn instructions.

BS can be saved in and restored from the stack area in 8-bit units by using PUSH BS instructions.

| MBS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
| MBS3 | MBS2 | MBS1 | MBS0 | 0 | 0 | 0 | 0 | BS |

**Figure 4.8–1 Bank Selection Register Configuration**

(1) Memory bank selection register (MBS)

The memory bank selection register (MBS), consisting of four bits, stores high-order 4-bit address information of a 12-bit data memory address. The memory bank to be accessed is specified according to the register contents. However, the μPD75328 allows the user to select bank 0, bank 1, or bank 15 only.

MBS is set by using the SEL MBn instruction (where n is 0, 1, or 15).

The address range applied according to how MBE and MBS are set is as shown in Fig. 3.1–2.

When the $\overline{\text{RESET}}$ signal is generated, MBS is initialized to 0.

## 5 PERIPHERAL HARDWARE FUNCTIONS

### 5.1 Digital Input / Output Ports

The µPD75328 adopts memory mapped I/O. All input / output ports are mapped in the data memory space.
Bit 0 in addresses 1F8H–1FFH is used for output latches for bit port outputs BP0–BP7.
For BP0–BP7, bit port output can be switched in 4-bit units by using display mode register (LCDM) bits 6 and 7. (See Fig. 5.7–2)
1F8H–1FFH bits not used for bit port output latches can be used for display memory or general purpose RAM. These addresses can be handled in 1- or 4-bit units. These addresses cannot be handled in 8-bit unit.

| Address | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|
| 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 |
| 1F8H | – | – | – | BP0 | |
| 1F9H | – | – | – | BP1 | |
| 1FAH | – | – | – | BP2 | |
| 1FBH | – | – | – | BP3 | |
| 1FCH | – | – | – | BP4 | |
| 1FDH | – | – | – | BP5 | |
| 1FEH | – | – | – | BP6 | |
| 1FFH | – | – | – | BP7 | |

Remarks: The part of – bits can be used for general purpose RAM

**Figure 5.1–1 Data Memory Addresses of Digital Ports**

Table 5.1–2 lists the input / output port operation instructions. In addition to 4-bit input / output, 8-bit input / output and bit manipulation can be performed for PORT 4 to PORT 7 and very diverse control can be performed. BP0–BP7 are 1-bit output ports.

Example: To test the P13 state and output value to ports 4 and 5 depending on the result.

```
SKT   PORT1.3    ;  Skip if port 1 bit 3 is set to 1
MOV   XA, #18H   ;  XA← 18H String effect
MOV   XA, #14H   ;  XA← 14H String effect
SEL   MB15       ;  Or CLR1 MBE
OUT   PORT4, XA  ;  Ports 5,4← XA
```

Example: SET1 PORT 4. @L ; The port 4–7 bit specified by using the L register is set to 1.

Example: To output 1 to BP0.

```
SET1  MBE
SEL   MB1        ;  Memory bank 1 is selected
SET1  BP0        ;  BP0← 1
```

5.1.1 Types, features, and configurations of digital input / output ports

Table 5.1–1 lists the digital input / output ports.
Figs. 5.1–1 to 5.1–5 shows the port configurations.

**Table 5.1–1 Types and Features of Digital Ports**

| Port (abbreviation) | Function | Operation and features | Remarks |
|---|---|---|---|
| PORT 0 | 4-bit input | Can always be read or tested regardless of the operation mode. | Pins also used for INT4, $\overline{\text{SCK}}$ SO/SB0, SI/SB1. Pins also used for INT0-2 and TI0. |
| PORT 1 | | | |
| PORT 3(Note 1) | 4-bit input / output | Can be placed in input or output mode in 1-bit units. | Pins also used for LCDCL SYNC and MD0–MD3, (Note 2) |
| PORT 6 | | | Pins also used for KR0–KR3. |
| PORT 2 | | Can be placed in input or ouput mode in 4-bit units. Ports 6 and 7 can be paired for data input / output in 8-bit units | Port 2 pins are also used for PTO0, PCL, and BUZ. |
| PORT 7 | | | Pins also used for KR4–KR7. |
| PORT 8 | | | |
| PORT 4(Note 1)<br>PORT 5(Note 1) | 4-bit input / output (N-channel open drain 10 Volts) | Can be placed in input or output mode in 4-bit units. Ports 4 and 5 can be paired for data input / output in 8-bit units. | Internal pull-up resistor can be specified in 1-bit units by using mask option. |
| BP0–BP7 | 1-bit output | Data is output in 1-bit units. The BP0–BP7 pins are also used as output pins (S24-S31) for the LCD driving segment signal. BP0-BP7 and S24-S31 can be changed by using software. | The capacity of drive is very small. Used for CMOS load drive. |

Note 1: An LED can be driven directly.
Note 2: PORT 3 is also used for MD0-MD3 pin only in μPD75P328.

The P10 pin is also used for an external vectored interrupt input pin (input with a noise removal circuit). (For details, see 6.3.)
The BP0-BP7 pins are also used for driving LCD segment signal output pins (S24-S31);they are changed in 4- or 8-pin units by using display mode register (LCDM) bits 6 and 7. BP0-BP7 are 1-bit output ports. Data for bit 0 in each of the display data memory addresses 1F8H-1FFH is output. (See 5.7.5.)
When the RESET signal is generated, port 2-port 7 output latches are cleared, the output buffers are turned off, and input mode is entered.

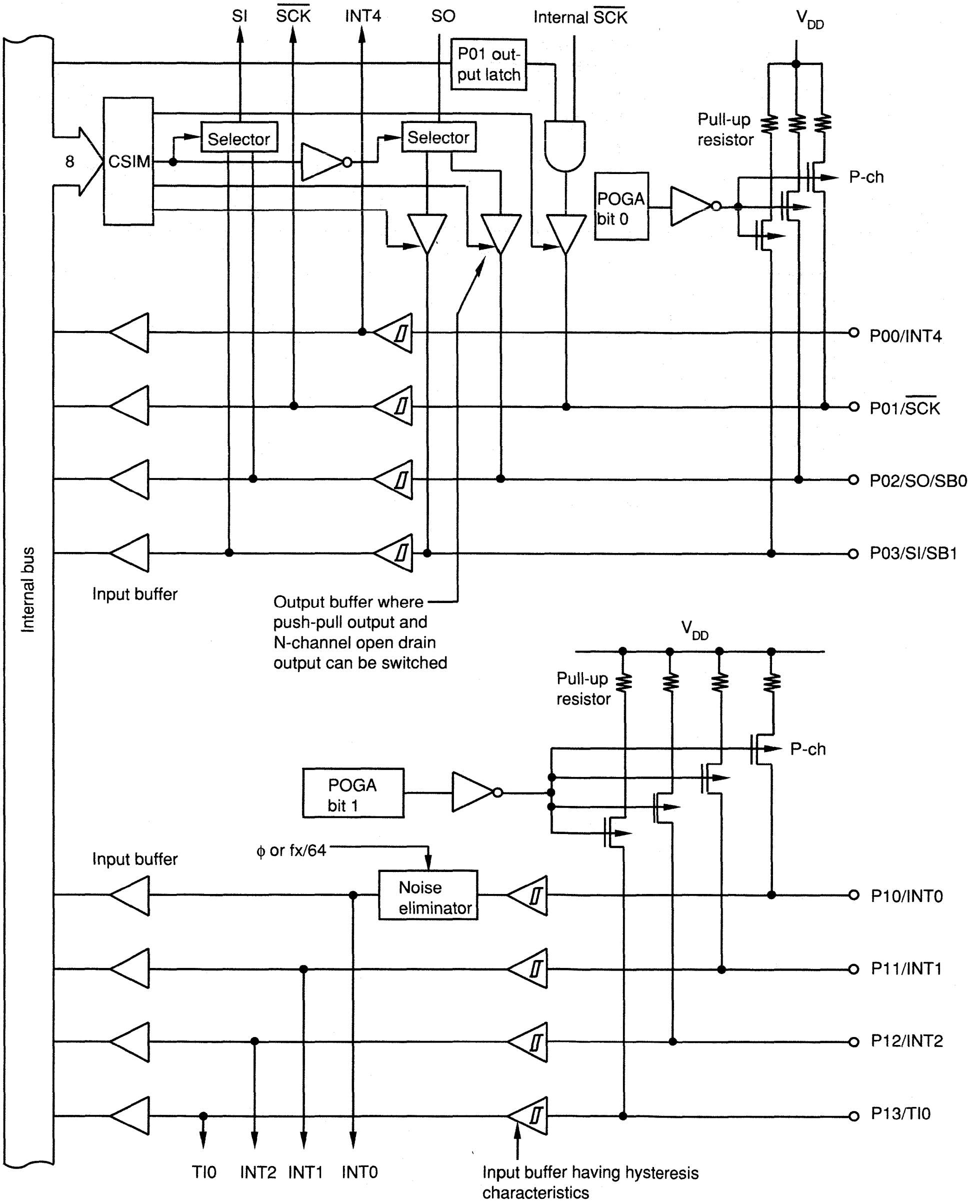


**Figure 5.1–2 Configuration of Ports 0 or 1**

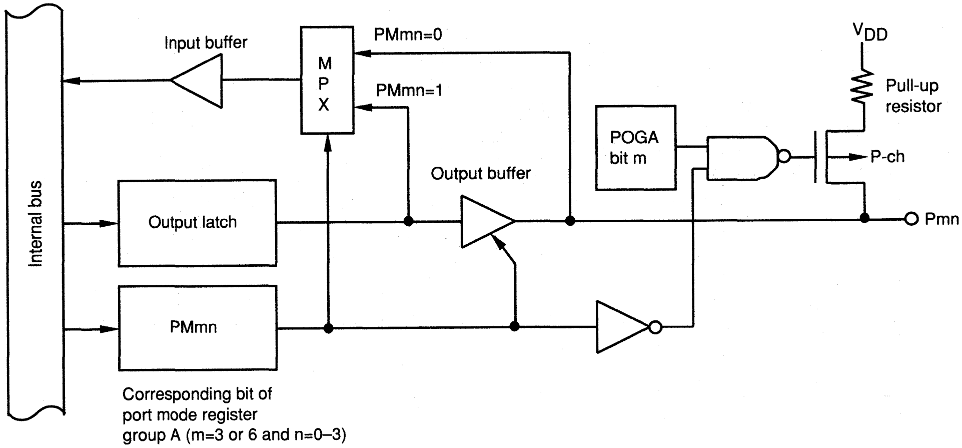


**Figure 5.1–3 Configuration of port 3n or 6n (n = 0–3)**

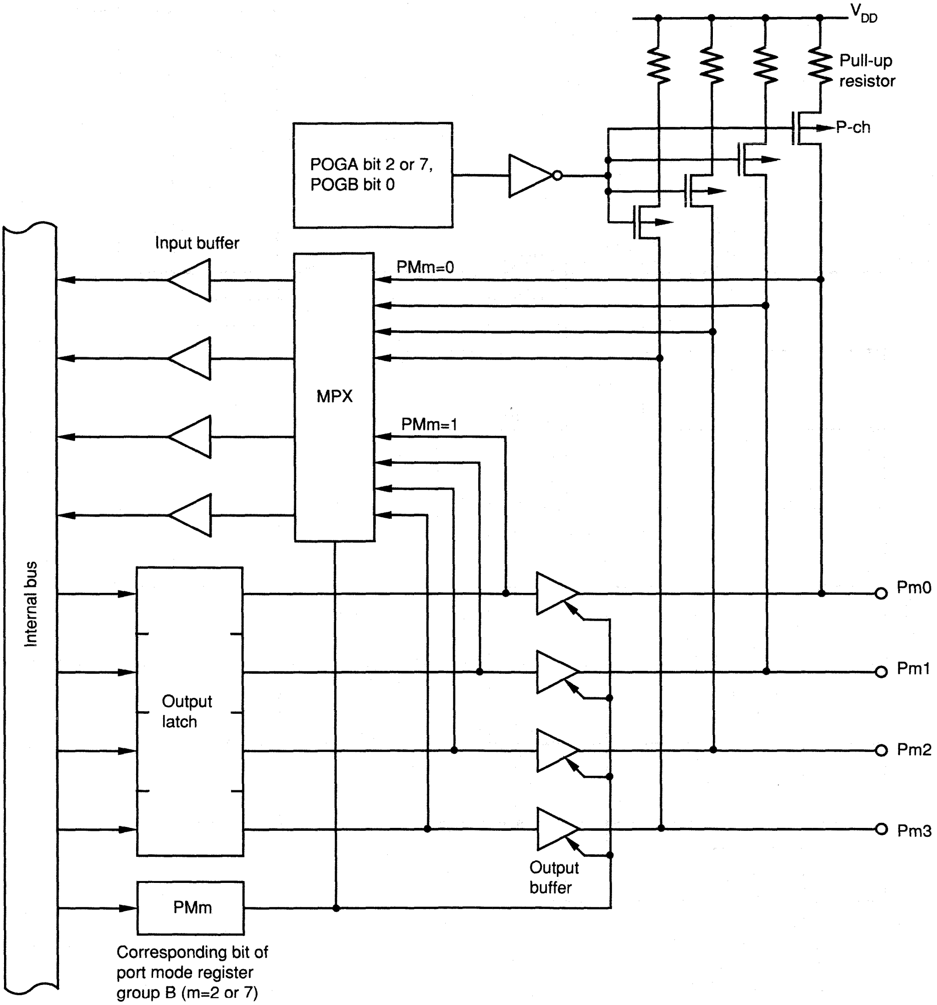


**Figure 5.1–4 Configuration of Port 2, 7 or 8**

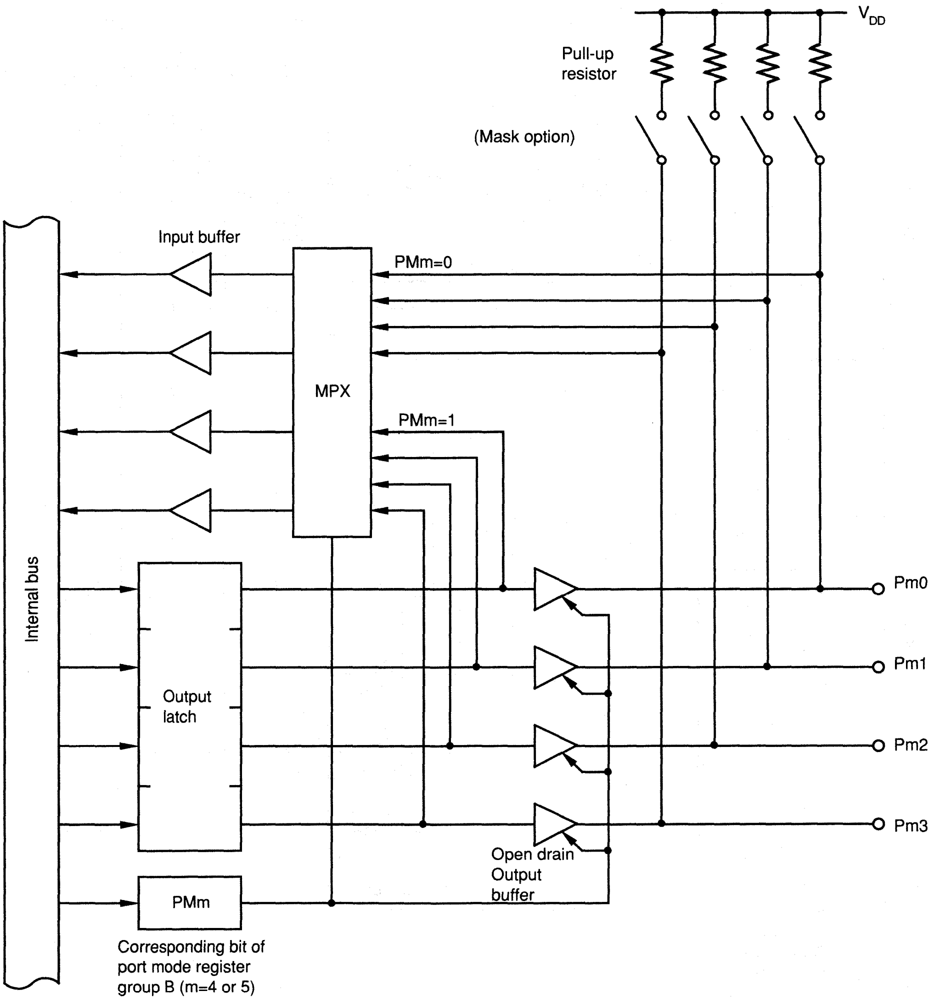


**Figure 5.1–5 Configuration of Port 4 or 5**

### 5.1.2 Input / output mode setting

The input or output mode of each input/output ports is set by using the port mode register as shown in Fig. 5.1–6. Each bit of ports 3 and 6 can be placed in input or output mode individually by using port mode register group A (PMGA). Each of ports 2, 4, 5, and 7 (four bits each) is placed in input or output mode individually by using port mode register group B (PMGB). Port 8 (four bits) is placed in input or output mode individually by using port mode register group C (PMGC).
Each port serves as an input port when the corresponding port mode register bit is set to = 0 and serves as an output port when set to 1.
Since the output latch contents are applied to the output pins at the same time the output mode is selected by setting the port mode register, the output latch contents must previously be rewitten to the required value before the output mode is set.
Port mode register groups A and B are set by using 8-bit memory operation instructions
When the $\overline{\text{RESET}}$ signal is generated, all bits of the port mode registers are cleared; thus the output buffers are turned off, and all ports are placed in input mode.

Example: To use P30, P31, P62, and P63 for input pins and P32, P33, P60, and P61 for output pins.

```
CLR1   MBE          ;   Or SEL MB15
MOV    XA, #3CH
MOV    PMGA, XA
```

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 |

| | | |
|---|---|---|
| | PM3n<br>PM6n | P3n, P6n pin input / output specification (n = 0–3) |
| | 0 | Input mode (output buffer off) |
| PMGA | 1 | Output mode (output buffer on) |

Port mode register group B

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

| | |
|---|---|
| PMn | Port n input / output specification (n = 2, 4, 5, or 7) |
| 0 | Input mode (output buffer off) |
| 1 | Output mode (output buffer on) |

Port mode register group C

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FEEH | – | – | – | – | – | – | – | PM8 | PMGB |

| | |
|---|---|
| PM8 | Port 8 input / output specification |
| 0 | Input mode (output buffer off) |
| 1 | Output mode (output buffer on) |

–: May be 0 or 1.

**Figure 5.1–6 Port Mode Register Formats**

### 5.1.3 Digital input / output port operation instructions

Since all input / output ports incorporated in the μPD75328 are mapped in the data memory space, all the data memory operation instructions are applicable. Table 5.1–2 lists the data memory operation instructions particulary useful for input / output pin operation and their application ranges.

(1) Bit operation instructions

The specific address bit direct addressing (fmem. bit) and specific address bit register indirect addressing (pmem. @L) modes are applicable to digital input / output ports PORT 0-PORT 7. The port bits can be operated at any time as desired regardless of how MBE and MBS are set.

Example: To OR P50 and P41 together and output the result to P61.

```
        SET1   CY            ;  CY←1
        AND1   CY, PORT5.0   ;  CY←CY∧P50
        OR1    CY, PORT4.1   ;  CY←CY∨P41
        SKT    CY
        BR     CLRP
        SET1   PORT6.1       ;  P61←1
         :
         :
  CLRP: CLR1   PORT6.1       ;  P61←0
```

(2) 4-bit operation instructions

In addition to the IN and OUT instructions, all 4-bit memory operation instructions such as MOV, XCH, ADDS, and INCS can be used. However, memory bank 15 must have been selected before executing an instruction.

Example 1: To output the accumulator contents to port 3.

```
        SEL    MB15          ;  Or CLR1 MBE
        OUT PORT3, A
```

Example 2: To add the accumulator value to the data output to port 5 and output the result.

```
        SET1   MBE
        SEL    MB15
        MOV    HL, #PORT5
        ADDS   A, @HL        ;  A← A+PORT5
        NOP
        MOV    @HL, A        ;  PORT5←A
```

Example 3: To test whether or not data in port 4 is greater than the accumulator value.

```
        SET1   MBE
        SEL    MB15
        MOV    HL, #PORT4
        SUBS   A, @HL        ;  A<PORT4
        BR     NO            ;  NO
                             ;  YES
```

(3) 8-bit operation instructions

In addition to the IN and OUT instructions, the MOV, XCH and SKE instructions can be used for ports 4 and 5 where 8-bit operation (manipulation) can be performed. As with the 4-bit operation instructions, memory bank 15 must have been selected before executing the instruction.

Example: To output data in the BC register pair to the output ports specified by the 8-bit data input from ports 4 and 5.

```
        SET 1  MBE
        SEL    MB15
        IN     XA, PORT 4    ;  XA← ports 5 and 4
        MOV    HL, XA        ;  HL←XA
        MOV    XA, BC        ;  XA←BC
        MOV    @HL, XA       ;  Port (L)←XA
```

**Table 5.1–2 Input / Output Port Pin Operation Instruction List**

<table>
<tr><th>PORT<br>Instruction</th><th>PORT<br>0</th><th>PORT<br>1</th><th>PORT<br>2</th><th>PORT<br>3</th><th>PORT<br>4</th><th>PORT<br>5</th><th>PORT<br>6</th><th>PORT<br>7</th><th>PORT<br>8</th><th>BIT•PORT<br>0–7</th></tr>
<tr><td>IN A, PORTn<br>(Note 1)</td><td colspan="9">O</td><td>-</td></tr>
<tr><td>IN XA, PORTn<br>(Note 1)</td><td colspan="2">–</td><td colspan="2">–</td><td colspan="2">O</td><td colspan="2">O</td><td>–</td><td>–</td></tr>
<tr><td>OUT PORTn. A<br>(Note 1)</td><td>–</td><td>–</td><td colspan="7">O</td><td>MOV mem, A<br>(Notes 3 and 4)</td></tr>
<tr><td>OUT PORTn. XA<br>(Note 1)</td><td colspan="2">–</td><td colspan="2">–</td><td colspan="2">O</td><td colspan="2">O</td><td>–</td><td>–</td></tr>
<tr><td>SET1 PORTn. bit</td><td>–</td><td>–</td><td colspan="7">O</td><td>SET1 BPn<br>(Note 3)</td></tr>
<tr><td>SET1 PORTn. @L<br>(Note 2)</td><td>–</td><td>–</td><td colspan="7">O</td><td>–</td></tr>
<tr><td>CLR1 PORTn. bit</td><td>–</td><td>–</td><td colspan="7">O</td><td>CLR1 BPn<br>(Note 3)</td></tr>
<tr><td>CLR1 PORTn. @L<br>(Note 2)</td><td>–</td><td>–</td><td colspan="7">O</td><td>–</td></tr>
<tr><td>SKT PORTn. bit</td><td colspan="9">O</td><td>SKT BPn<br>(Note 3)</td></tr>
<tr><td>SKT PORTn. @L<br>(Note 2)</td><td colspan="9">O</td><td>–</td></tr>
<tr><td>SKF PORTn. bit</td><td colspan="9">O</td><td>SKF BPn<br>(Note 3)</td></tr>
<tr><td>SKF PORTn. @L<br>(Note 2)</td><td colspan="9">O</td><td>–</td></tr>
<tr><td>AND1 CY, PORTn. bit</td><td colspan="9">O</td><td>AND1 CY, @H+BPn<br>(Notes 3 and 5)</td></tr>
<tr><td>AND1 CY, PORTn. @L<br>(Note 2)</td><td colspan="9">O</td><td>–</td></tr>
<tr><td>OR1 CY, PORTn. bit</td><td colspan="9">O</td><td>OR1 CY, @H+BPn<br>(Notes 3 and 5)</td></tr>
<tr><td>OR1 CY, PORTn. @L<br>(Note 2)</td><td colspan="9">O</td><td>–</td></tr>
<tr><td>XOR1 CY, PORTn. bit</td><td colspan="9">O</td><td>XOR1 CY, @H+BPn<br>(Notes 3 and 5)</td></tr>
<tr><td>XOR1 CY, PORTn. @L<br>(Note 2)</td><td colspan="9">O</td><td>–</td></tr>
</table>

Note 1: MBE = 0 or (MBE = 1 and MBS = 15) must have been set before executing the instruction.
2: The low-order two bits of address and the bit address are specified indirectly by using the L register.
3: (MBE = 1 and MBS = 1) must have been set before executing the instruction.
4: Accumulator A bit 0 corresponds to BPn.
5: Write FH into the H register.

### 5.1.4 Digital input / output port operation

When data memory operation instructions are executed for the digital input / output ports, port (pin) operation varries according to which mode (input or output) is set. (See table 5.1–3.) This is because as understood from the input / output port structure, data read into the internal bus is data for each pin in the input mode and output latch data in the output mode.

(1) Operation when input mode is set

Pin data is operated when executing a test instruction such as SKT or an instruction to read 4- or 8-bit port data into the internal bus (IN, OUT, operation, or comparison instruction).

When an instruction to transfer the accumulator contents (four or eight bits) to a port or ports is executed (OUT or MOV instruction), the accumulator data is latched in the output latches. The output buffers remain off.

When an XCH instruction is executed, pin data is input to the accumulator and the accumulator data is latched in the output latches. The output buffers remain off.

When an INCS instruction is executed, data resulting from adding 1 to 4-bit pin data is latched in the output latches. The output buffer remain off.

When an instruction to rewrite data memory bitwise is executed, such as SET1, CLR1, or SKTCLR, (the specifed bit output latch can be rewritten as specified by instruction), but the contents of the output latch for other bits become undefined.

(2) Operation when output mode is set

When a test instruction or an instruction to read 4- or 8-bit port data into the internal bus is executed, the output latch contents are operated.

When an instruction to transfer the accumulator contents (four or eight bits) is executed, output latch data is rewritten and at the same time as output from the pins.

When an XCH instruction is executed, the output latch contents are transferred to the accumulator, and the accumulator contents are held in the output latches and is output from the pins.

When an INCS instruction is executed, data resulting from incrementing the output latch contents by one is held in the output latches and output from the pins.

When a bit output instruction is executed, the specified output latch bit is rewritten and output from the pin.

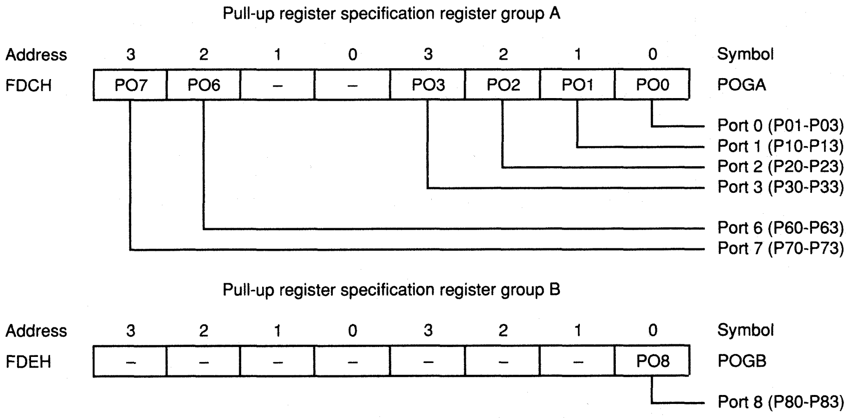
**Table 5.1–3 Operation When Input / Output Ports are Used**

| Executed instruction | Port (pin) operation | |
|---|---|---|
| | Input mode | Output mode |
| SKT ①<br>SKF ① | Pin data is tested. | Output latch is data tested. |
| AND1 CY, ①<br>OR1 CY, ①<br>XOR1 CY, ① | Operation is performed between pin data and CY. | Operation is performed between output latch data and CY. |
| IN A, PORTn<br>IN XA, PORTn<br>MOV A, @HL<br>MOV XA, @HL | Pin data is transferred to accumulator. | Output latch data is transferred to accumulator |
| ADDS A, @HL<br>ADDC A, @HL<br>SUBS A, @HL<br>SUBC A, @HL<br>AND A, @HL<br>OR A, @HL<br>XOR A, @HL | Operation is performed between pin data and accumulator. | Operation is performed between output latch data and accumulator. |
| SKE A, @HL | Pin data and accumulator contents are compared. | Output latch data and accumulator contents are compared. |
| OUT PORTn, A<br>OUT PORTn, XA<br>MOV @HL, A<br>MOV @HL, XA | Accumulator data is transferred to output latches. (Output buffers remain off.) | Accumulator data is transferred to output latches and output from the pins. |
| XCH A, PORTn<br>XCH XA, PORTn<br>XCH A, @HL<br>XCH XA, @HL | Pin data is transferred to accumulator and accumulator data is transferred to output latches.<br>(Output buffers remain off.) | Data is exchanged between output latches and the accumulator. |
| INCS PORTn<br>INCS @HL | Data resulting from incrementing pin data by one is latched in output latches. | The output latch contents are incremented by one. |
| SET1 ⏻<br>CLR1 ⏻<br>SKTCLR ⏻ | The specified bit output latches are rewritten as specified by the instruction, but the output latch contents for other bits are undefined. | The output pin state is changed according to the instruction. |

⏻ : Denotes two addressing modes PORTn. bit and PORTn. @L.

5.1.5 Internal pull-up resistors

Pull-up resistors can be incorporated in the μPD75328 port pins except P00 or BP0-BP7. Internal pull-up resistors are specified by using software or mask option.



| | Specification |
|---|---|
| 0 | Pull-up resistor is not incorporated. |
| 1 | Pull-up resistor is incorporated. |

**Figure 5.1–7 Pull-Up Register Specification Register Format**

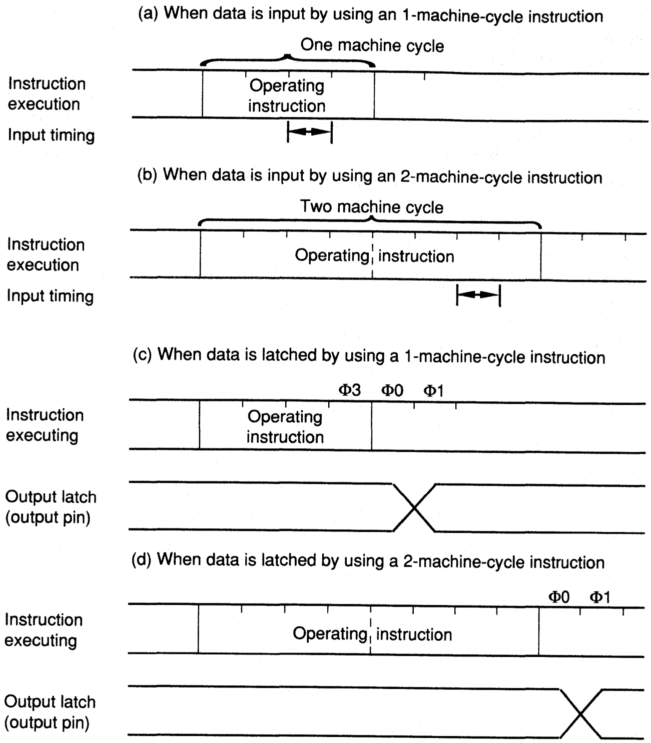
**Table 5.1–4**

| Port (pin names) | Internal pull-up resistor specification method | Specification Bit |
|---|---|---|
| Port 0 (P01-P03) (Note 1) | Software is used for internal resistor specification in 3-bit units. | POGA.0 |
| Port 1 (P10-P13) | Software is used for internal resistor specification in 4-bit units. | POGA.1 |
| Port 2 (P20-P23) | | POGA.2 |
| Port 3 (P30-P33) | | POGA.3 |
| Port 6 (P60-P63) | | POGA.6 |
| Port 7 (P70-P73) | | POGA.7 |
| Port 8 (P80-P83) | | POGB.0 |
| Port 4 (P40-P43) | Mask option is used for internal resistor specification in 1-bit units. | – |
| Port 5 (P50-P53) | | |

Note: A pull-up resistor cannot be incorporated in the P00 Pin.

5.1.6 Digital input / output port input / output timing

Fig. 5.1–8 shows timing of data to output latch and timing of pin data or output latch data input to the internal bus.
Fig. 5.1–9 shows the ON timing at specified the internal pull-up resistor by software.



(a) When data is input by using an 1-machine-cycle instruction
One machine cycle
Instruction execution
Operating instruction
Input timing
(b) When data is input by using an 2-machine-cycle instruction
Two machine cycle
Instruction execution
Operating instruction
Input timing
(c) When data is latched by using a 1-machine-cycle instruction
Φ3
Φ0
Φ1
Instruction executing
Operating instruction
Output latch (output pin)
(d) When data is latched by using a 2-machine-cycle instruction
Φ0
Φ1
Instruction executing
Operating instruction
Output latch (output pin)

**Figure 5.1–8 Digital Input / Output Port Input / Output Timing**

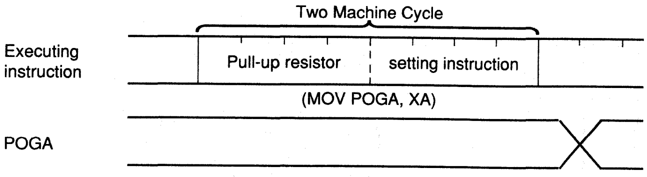


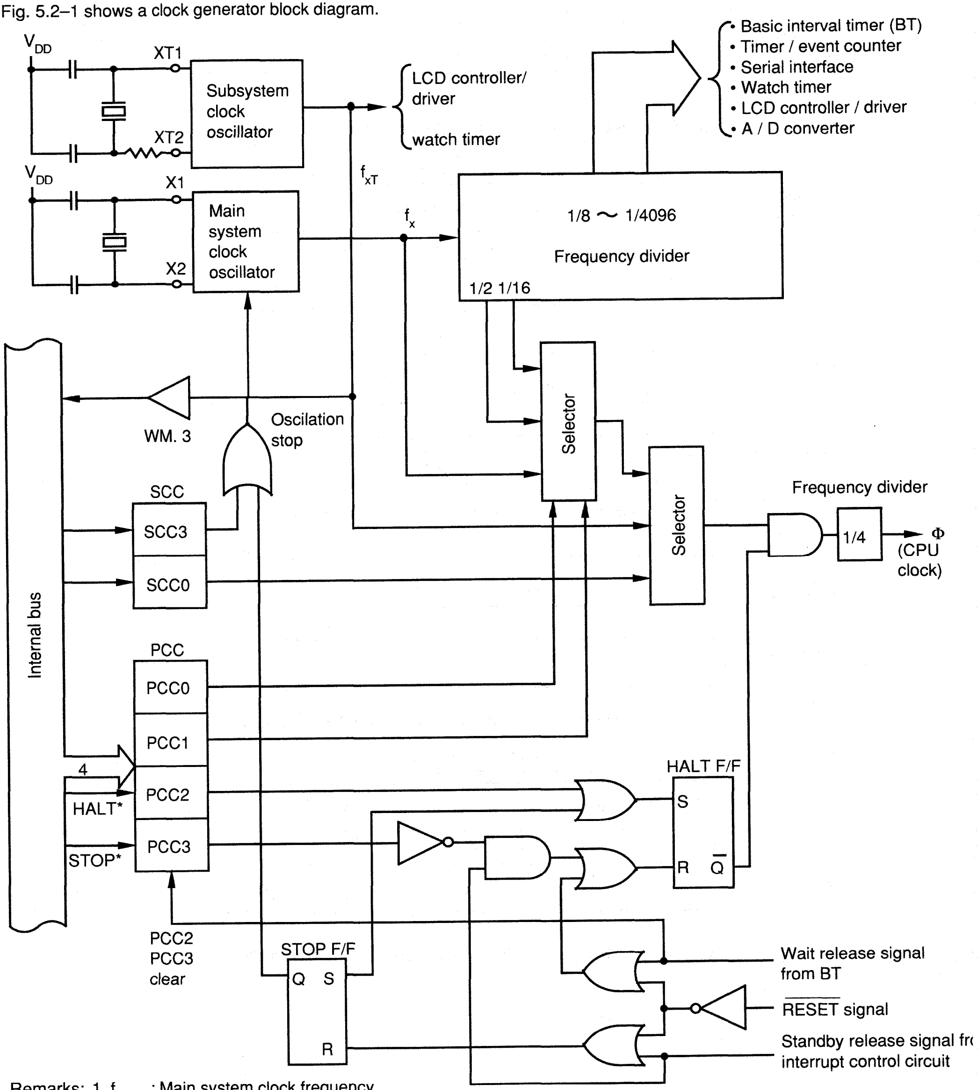
**Figure 5.1–9 ON Timing of Pull-up Resistor by using Software**

## 5.2 Clock Generator

The clock generator supplies clocks to CPU and peripheral hardware for controlling the CPU operating mode.

5.2.1 Clock generator configuration

Fig. 5.2–1 shows a clock generator block diagram.



Remarks: 1. $f_X$ : Main system clock frequency
2. $f_{XT}$ : Subsystem clock frequency
3. PCC : Processor clock control register
4. SCC : System clock control register
5. * denotes instruction execution.

**Figure 5.2–1 Clock Generator Block Diagram**

5.2.2 Clock generator function and operation

The clock generator produces the following clocks and controls CPU operating modes such as standby:
— Main system clock $f_X$
— Subsystem clock $f_{XT}$
— CPU clock φ
— Clock to peripheral hardware
The clock generator operates according to how the processor clock control register (PCC) and system clock control register (SCC) are set, as described below:

(a) When the RESET signal is generated, the minimum speed mode of the main system clock (15.3 μ/4.19 MHz) is selected. (PCC = 0 and SCC = 0)
(b) When the main system clock is selected, one of three CPU clock frequencies can be selected (0.95 μs, 1.91 μs, and 15.3 μs/ 4.19 MHz) by setting PCC.
(c) When the main system clock is selected, the standby mode (STOP or HALT) can be used.
(d) Subsystem clock is selected by setting SCC.0, and operation can be performed in a very low-speed and low current consumption (122 μs/32.768 kHz). In this case, the PCC setup value does not affect the CPU clock.
(e) When the subsystem clock is selected, main system clock oscillation can be stopped by setting SCC.3. The HALT mode can also be used, but the STOP mode cannot be used. (Subsystem clock oscillation cannot be stopped.
(f) The main system clock is divided to generate a clock supplied to peripheral hardware. The subsystem clock can be supplied directly only to the watch timer. Thus, the watch function and the LCD controlller and buzzer output function which operate using the watch timer clock can also continue operation in the standby mode.
(g) When subsystem clock is selected, the watch timer and LCD controller can continue normal operation. When the main system clock is stopped, other hardware devices than the watch timer or LCD controller cannot be used because they operate according to the main system clock.

(1) Processor clock control register (PCC)

PCC consists of four bits; the low-order two bits are used to select CPU clock φ, and the high-order two bits are used to control CPU operation mode. (See Fig. 5.2-2.)
When bit 3 or 2 is set to (1), the standby mode is set. When the standby mode is released by the standby mode release signal, automatically bits 3 and 2 are cleared and the normal operating mode is entered. (For details, see Paragraph 7).
The low-order two bits of PCC are set by using a 4-bit memory operation instruction (the high-order two bits are set 0).
Bits 3 and 2 are set to (1) by using STOP and HALT instructions respectively.
The STOP and HALT instructions can always be executed independently of the MBE contents.
CPU clock can be selected only during main system clock operation. When the μPD75328 is operated on the subsystem clock, the low-order two bits of PCC becomes invalid, and the CPU clock is fixed to $f_{XT}/4$. The stop instruction is also enabled only during main system operation.

Example 1: To set a machine cycle to 0.95 μs (4.19) MHz).

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

Example 2: To set STOP mode. (Be sure to enter a NOP instruction following the STOP or HALT instruction.)

```
STOP
NOP
```

When the RESET signal is generated, PCC is cleared.

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

CPU clock selection bits

| | | SCC = 0<br>The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz | | SCC = 1<br>The value enclosed in parentheses is applied when $f_{XT}$ = 32.768 kHz | |
|---|---|---|---|---|---|
| | | CPU clock frequency | 1 machine cycle | CPU clock frequency | 1 machine cycle |
| 0 | 0 | $\Phi = f_X/64$<br>(65.5 kHz) | 15.3 μs | $\Phi = f_{XT}/4$<br>(8.192 kHz) | 122 μs |
| 0 | 1 | Undefined | – | | |
| 1 | 0 | $\Phi = f_X/8$<br>(524 kHz) | 1.91 μs | | |
| 1 | 1 | $\Phi = f_X/4$<br>(1.05 MHz) | 0.95 μs | | |

$f_X$ : Main system clock oscillator output frequency

$f_{XT}$: Subsystem clock oscillator output frequency

CPU operation mode control bits

| | | |
|---|---|---|
| 0 | 0 | Normal operating mode |
| 0 | 1 | HALT mode |
| 1 | 0 | STOP mode |
| 1 | 1 | Undefined |

**Figure 5.2–2 Processor Clock Control Register Format**

(2) System clock control register (SCC)

SCC consists of four bits; the least significant bit is used to select CPU clock, and the most significant bit is used to control (stop) main system clock oscillation. (See Fig. 5.2–3.)

SCC 0 and SCC 3 exist at the same data memory address; the bits cannot be changed at the same time. Thus, SCC 0 and SCC 3 are set by using a bit operation instruction. SCC 0 and SCC 3 can always be operated independetly of the MBE contents. Main system clock oscillation can be stopped by setting SCC 3 only during subsystem clock operation. Main system clock oscillation is stopped by using the STOP instruction during main system clock operation.

When the RESET signal is generated, SCC is cleared.

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| FB7H | SCC3 | – | – | SCC0 | SCC |

| SCC3 | SCC0 | CPU clock slection | Main system clock oscillation |
|---|---|---|---|
| 0 | 0 | Main system clock | Oscillation is enabled. |
| 0 | 1 | Subsystem clock | |
| 1 | 0 | Undefined | |
| 1 | 1 | Subsystem clock | Oscillation is stopped. |

**Figure 5.2–3 System Clock Control Register Format**

Caution 1: Changing the system clock requires a maximum of $1/f_{XT}$ time. To stop main system clock after changing the subsystem clock, set SCC 3 after the machine cycle or cycles listed in Table 5.2–1.

2: Even if oscillation is stopped by setting SCC 3 during main system clock operation, normal STOP mode is not entered.

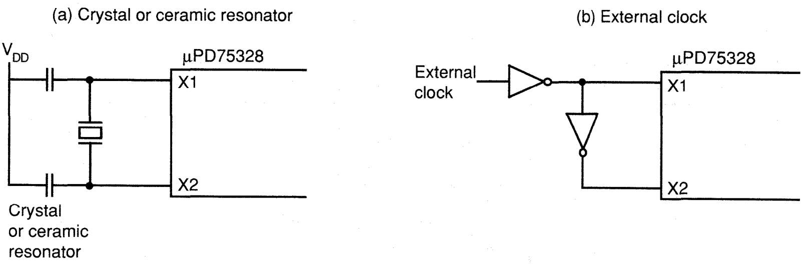
3: When SCC.3 is set to 1, X1 input is connected internally to $V_{SS}$, to avoid leakage current due to crystall oscillator. When the external clock is used in main system clock, do not set SCC.3 to 1.

4: When the system clock is switched to the subsystem clock it is necessary to disable any interrupt during the time given in table 5.2-1.

(3) System clock oscillators

The main system clock oscillator uses a crystal oscillator (4.194304 MHz standard) or cermanic oscillator connected to the X1 and X2 pins.

An external clock can also be input.


(a) Crystal or ceramic resonator

(b) External clock

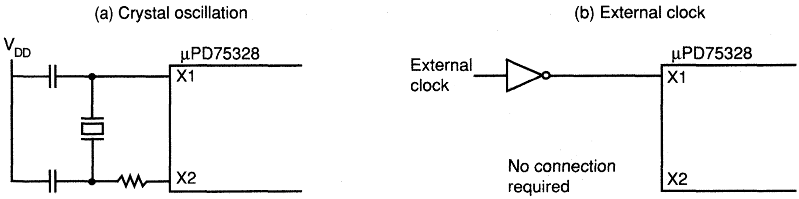
**Figure 5.2–4 External Circuit to Main System Clock Oscillator**

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

- The subsystem clock oscillator uses a crystal oscillator (32.768 kHz standard) connected to the XT1 and XT2 pins.

An external clock can also be input.

The XT1 pin state can be tested by using watch mode register (WM) bit 3.


(a) Crystal oscillation

(b) External clock

**Figure 5.2–5 External Circuit to Subsystem Clock Oscillator**

(4) Frequency divider

The frequency divider divides the main system clock oscillator output ($f_X$) and generates various clocks.

5.2.3 System clock and CPU clock setting

(1) Time required to change system and CPU clocks

The system and CPU clocks can be changed by using the low-order two bits of PCC and the least significant bit of SCC. However, this clock change is not immediatly made after the register are rewritten, and the clock before the clock change is made is used for operation during given machine cycles. Thus, to stop main system clock oscillation, a STOP instruction must be executed or SCC 3 must be set after the change time elapses.

**Table 5.2–1 Maximum Time Required to Change System and CPU Clocks**

| Setup value before change | | | Setup value after change | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCC | PCC | PCC | SCC0 | PCC1 | PCC0 | SCC0 | PCC1 | PCC0 | SCC0 | PCC1 | PCC0 | SCC0 | PCC1 | PCC0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | X | X |
| 0 | 0 | 0 | | | | 1 machine cycle | | | 1 machine cycle | | | $\frac{f_X}{64f_{XT}}$ machine cycle<br>(2 machine cycles) | | |
| | 1 | 0 | 8 machine cycles | | | | | | 8 machine cycles | | | $\frac{f_X}{8f_{XT}}$ machine cycles<br>(16 machine cycles) | | |
| | 1 | 1 | 16 machine cycles | | | 16 machine cycles | | | | | | $\frac{f_X}{4f_{XT}}$ machine cycles<br>(32 machine cycles) | | |
| 1 | X | X | $\frac{f_X}{64f_{XT}}$ machine cycles<br>(2 machine cycles) | | | $\frac{f_X}{8f_{XT}}$ machine cycles<br>(16 machine cycles) | | | $\frac{f_X}{4f_{XT}}$ machine cycles<br>(32 machine cycles) | | | | | |

The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz and $f_{XT}$ = 32.768 kHz.

Remarks: The CPU clock Φ is supplied to the µPD75328 internal CPU. The reciprocal of the clock becomes the minimum instruction time. (The present manual defines it to be one machine cycle.)

(2) System and CPU clock change sequence
System and CPU clock change is explained using Fig. 5.2–6.

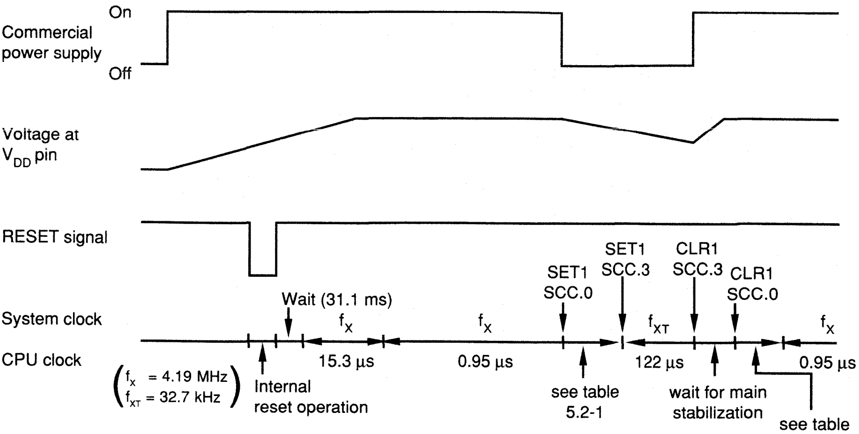


**Figure 5.2–6 System and CPU Clock Change**

1) When the RESET signal is generated, the CPU starts operating at the minimum speed of the main system clock (15.3 μs / 4.19 MHz) after the wait time (31.3 ms / 4.19 MHz) to allow time for stabilizing the oscillator.

2) After enough time has elapsed for the $V_{DD}$ pin voltage to rise to an adequate level PCC is rewritten and the μPD75328 operates at the maximum speed.

3) Turning off the commercial power supply is detected by using interrupt input (INT4 is useful); SCC. 0 is set, and the μPD75328 operates using the subsystem clock. (At the time, a check must have been made to ensure that subsystem clock oscillation has started. After the time required to change to the subsystem clock (32 machine cycles) has elapsed, SCC 3 is set and main system clock oscillation is stopped.

4) Restoration of the commercial power supply is detected by using an interrupt. SCC. 3 is cleared and main system clock osscillation is started. After the time required to stabilize oscillation has elapsed, SCC. 0 is cleared and the μPD75328 operates at the maximum speed.

5.2.4 Clock output circuit

(1) Clock output circuit configuration
Fig. 5.2-7 shows a clock output circuit block diagram.

(2) Clock output circuit function
The clock output circuit, which outputs clock pulses from the P22/PCL pin, is used to supply clock pulses to remote control output or peripheral LSIs.

Clock pulses are output in the following sequence:

(a) Clock output frequency is selected. Clock output is disabled.
(b) 0 is written into P22 output latch.
(c) Port 2 input/output mode is placed in output mode.
(d) Clock output is enabled.

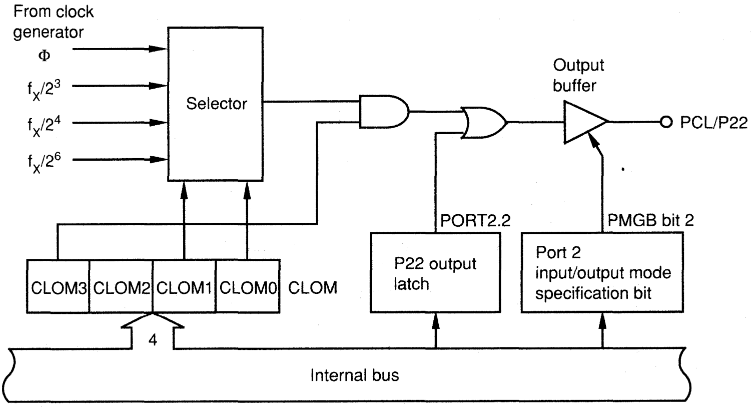


**Figure 5.2–7 Clock Output Circuit Block Diagram**

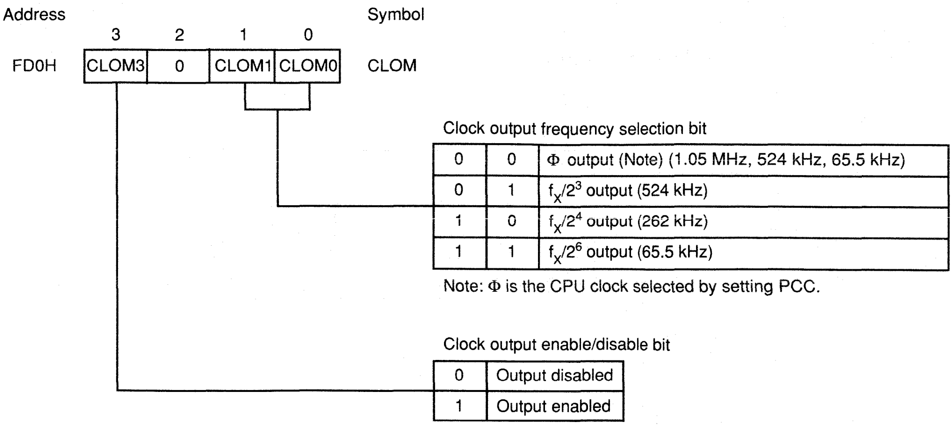
Remarks: The circuit is designed so as not to output a spurious short-width pulse when changing between clock output enable and disable.

(3) Clock output mode register (CLOM)
CLOM is a 4-bit register to control clock output.
CLOM is set by using a 4-bit memory operation instruction. It cannot be read.

Example: To output clock Φ from PCL/P22 pin.

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

When the $\overline{\text{RESET}}$ signal is generated, CLOM is cleared and clock output is disabled.



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

Clock output frequency selection bit

| | | |
|---|---|---|
| 0 | 0 | Φ output (Note) (1.05 MHz, 524 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: Φ is the CPU clock selected by setting PCC.

Clock output enable/disable bit

| | |
|---|---|
| 0 | Output disabled |
| 1 | Output enabled |

Caution: Be sure to write 0 into CLOM bit 2.

**Figure 5.2–8 Clock Output Mode Register Format**

(4) Application example to remote control output
The μPD75328 clock output function is applicable to remote control output. Remote control output carrier frequency is selected by using the clock frequency selection bit of the clock output mode register. Pulse output is enabled or disabled under software control by using the clock output enable/disable bit.
It is designed so that a spurious short-width pulse will not be produced when changing between clock output enable and disable.

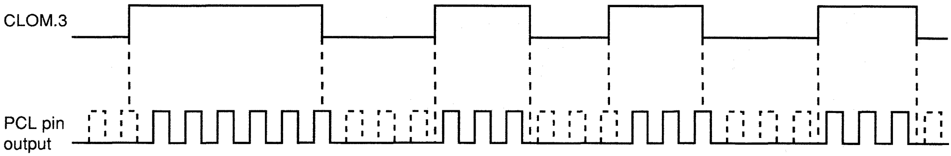


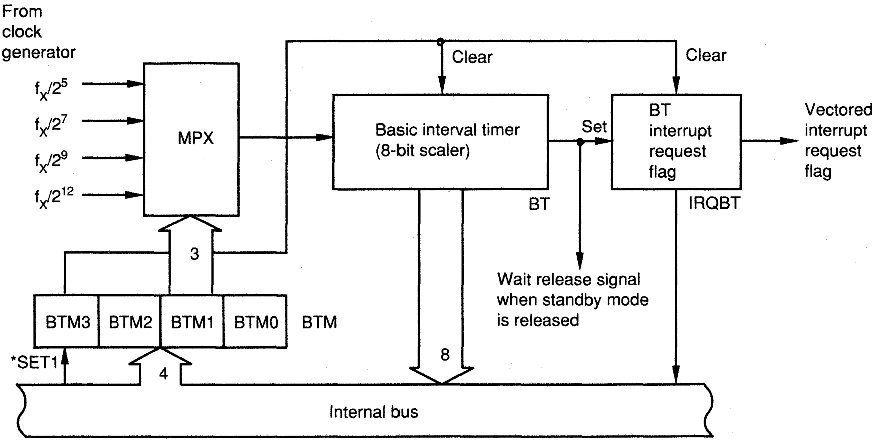
**Figure 5.2–9 Remote Control Output Application Example**

### 5.3 Basic Interval Timer

The μPD75328 contains an 8-bit basic interval timer which has the following functions:
(a) Reference time generation (four types of time interval)
(b) Wait time selection and count when standby mode is released.
(c) Count read
The basic interval timer can also be applied to a watchdog timer to detect program overrun.

5.3.1 Basic interval timer configuration

Fig. 5.3–1 shows the configuration of the basic interval timer


From clock generator
$f_X/2^5$
$f_X/2^7$
$f_X/2^9$
$f_X/2^{12}$
MPX
Clear
Clear
Basic interval timer (8-bit scaler)
Set
BT interrupt request flag
Vectored interrupt request flag
BT
IRQBT
3
Wait release signal when standby mode is released
BTM3
BTM2
BTM1
BTM0
BTM
*SET1
4
8
Internal bus

Remarks: * denotes instruction executing.

**Figure 5.3–1 Basic Interval Timer Configuration**

5.3.2. Basic interval timer mode register (BTM)

BTM is a 4-bit register for controlling operation of the basic interval timer.
BTM is set by using a 4-bit memory operation instruction.
Bit 3 can be set individually by using a bit operation instruction.

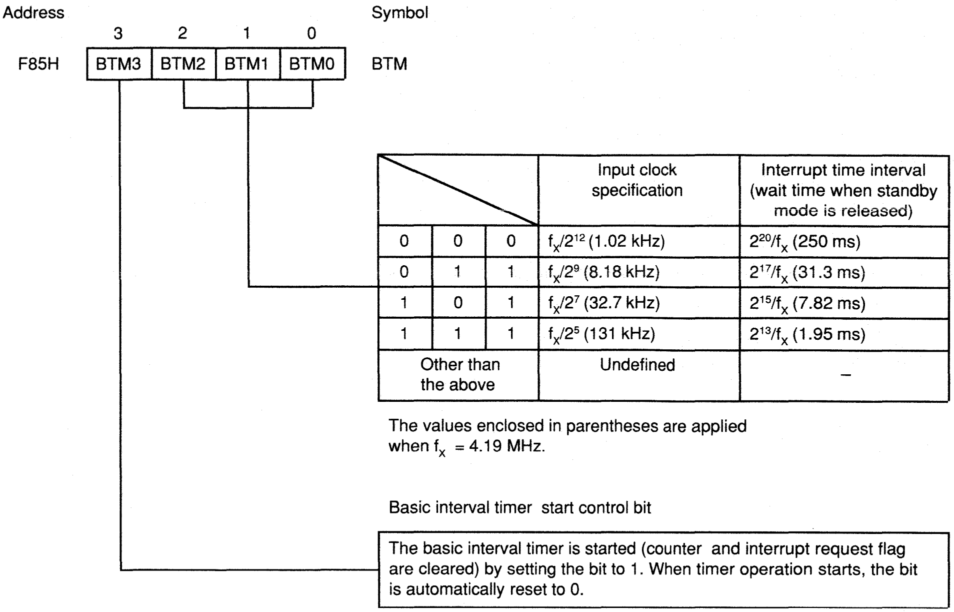
Example 1: To set the interrupt generation interval to 1.95 ms (4.19 MHz).

```
SEL    MB15      ;  or CLR1 MBE
MOV    A, # 1111B
MOV    BTM, A    ;  BTM←— 1111B
```

Example 2: To clear BT and IRQBT (watchdog timer application).

```
SEL    MB15      ;        or CLR1 MBE
SET1   BTM.3     ;        BTM BIT 3 is set  to 1.
```

When bit 3 is set to 1, the basic interval timer contents are cleared. At the same time, the basic interval interrupt request flag (IRQBT) is also cleared. (The basic interval timer starts.)
When the $\overline{\text{RESET}}$ signal is generated, the BTM contents are cleared and interrupt request signal generation is set for the longest interval.



| | | | Input clock specification | Interrupt time interval (wait time when standby mode is released) |
|---|---|---|---|---|
| 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 | $f_X/2^7$ (32.7 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 than the above | | | Undefined | – |

The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz.

Basic interval timer start control bit

The basic interval timer is started (counter and interrupt request flag are cleared) by setting the bit to 1. When timer operation starts, the bit is automatically reset to 0.

**Fig. 5.3-2 Basic Interval Timer Mode Register Format**

### 5.3.3. Basic interval timer operation

The basic interval timer (BT) is incremented each time a pulse is received from the clock generator. When an overflow occurs, the interrupt request flag (IRQQBT) is set. BT count operation cannot be stopped.
An interrupt generation time interval can be selected from among four types by setting BTM. (See Fig. 5.3—2.)
The basic interval timer and interrupt request flag can be cleared by setting BTM bit 3 to 1 (the interval timer function starts).
The basic interval timer (BT) count can be read by using an 8-bit operation instruction. Data cannot be written into BT.

Caution:
To prevent reading of unstable data during count update when reading the basic interval timer count, execute the read instruction twice and compare the results. If they are valid values, the later read value is used as the read result, if they differ from each other completely, reexecute from the beginning.

Example: To read the BT count

```
        SET1   MBE
        SEL    MB15
        MOV    HL, #BT      ;   BT address is set in HL.
LOOP:   MOV    XA, @HL      ;   First read
        MOV    BC, XA
        MOV    XA, @ HL     ;   Second read
        SKE    A, C
        BR     LOOP
        MOV    A, X
        SKE    A, B
        BR     LOOP
```

The wait function is provided to stop CPU operation until the basic interval timer overflows in order to allow time for system clock oscillation to become stable when the STOP mode is released.
Although the wait time after the $\overline{\text{RESET}}$ signal is generated is fixed, it can be selected by setting BTM when the STOP mode is released by an interrupt. The wait time is selected by setting BTM as shown in Fig. 5.3—2. BTM setting must be performed before the STOP mode is set. (For details, see CHAPTER 7.)

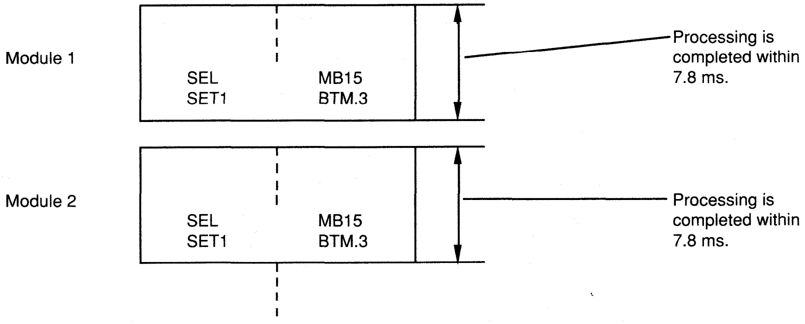
5.3.4 Basic interval timer application examples

Example 1: To enable basic interval timer interrupt and set interrupt generation interval at 1.95 ms (at 4.19 MHz).

```
SEL     MB 15
MOV     A, #1111B
MOV     BTM, A       ;   BTM  setting and timer function start
EI                   ;   Interrupt is enabled.
EI      IEBT         ;   BT interrupt is enabled.
```

Example 2: Watchdog timer applciation
A program is divided into serval modules which terminate processing within the BT setup time. BT and IRQBT are cleared at the end of each module. If an interrupt is generated, overrun is asumed to have occured.

```
           SEL     MB15
Initial-   MOV     A, #1101B   ;   7.8 ms interval is set.
izatiion   MOV     BTM, A      ;   BTM setting and timer function start
           EI
           EEI     IEBT
```



Module 1

```
SEL      MB15
SET1     BTM.3
```

Processing is completed within 7.8 ms.

Module 2

```
SEL      MB15
SET1     BTM.3
```

Processing is completed within 7.8 ms.

Example 3: To set the wait time, when the STOP mode is released, using an interrupt to 7.8 ms.

```
SEL     MB15         ;   or CLR1 MBE
MOV     A, #1101B
MOV     BTM, A       ;   BTM ← 1101B
STOP                 ;   STOP mode is set
NOP
```

Example 4:
To set the high-level width of a pulse input to INT4 interupt (both rising and falling edge detection). (The pulse width must not exceed the BT setup value. The BT setup value must be 7.8 ms or more.)
< INT4 interrupt routine (MBE = 0) >

```
LOOP:  MOV    XA, BT      ;  First read
       MOV    BC, XA      ;  Data is stored.
       MOV    XA, BT      ;  Second read
       SKE    A, C
       BR     LOOP
       MOV    A, X
       SKE    A, B
       BR     LOOP
       SKT    PORT0.0     ;  P00=1?
       BR     AA          ;  NO
       MOV    XA, BC      ;  Data is stored in data memory.
       MOV    BUFF, XA
       CLR1   FLAG
       RETI
  AA:  MOV    HL, #BUFF
       MOV    A, C
       SUBC   A, @HL
       INCS   L
       MOV    C, A
       MOV    A, X
       SUBC   A, @HL
       MOV    B, A
       MOV    XA, BC
       MOV    BUFF, XA    ;  Data is stored.
       SET1   FLAG        ;  Data existence flag is set.
       RETI
```

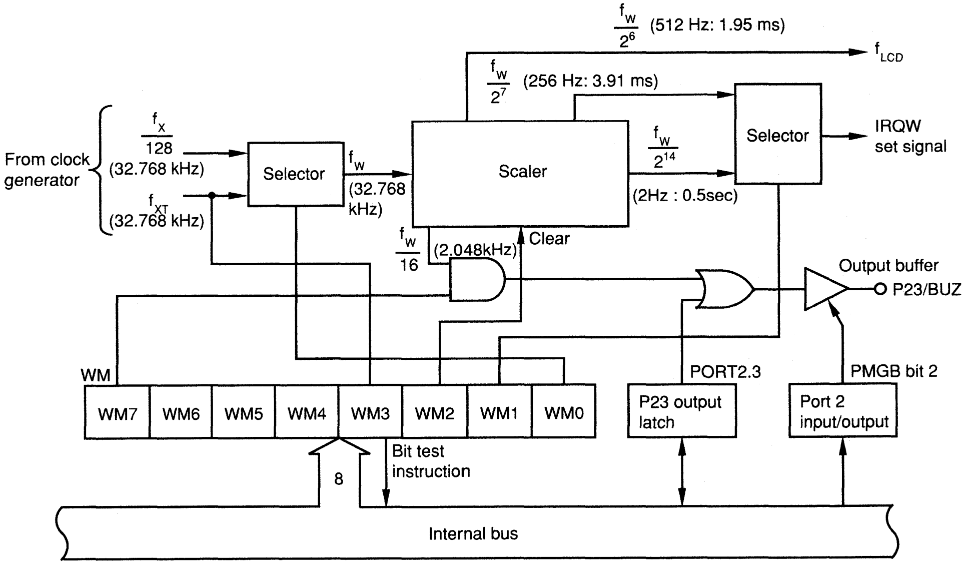
### 5.4 Watch Timer

The μPD75328 incorporates a watch timer (one channel) which has the following function:

(a) The test flag (IRQW) is set at 0.5 s time intervales. The standby mode can be released using IRQW.
(b) The main system and subsystem clocks can be assigned 0.5-s intervals.
(c) In the rapid feed mode, time intervals multiplied by 128 (3.91 ms) are enabled. The mode is useful for program debugging and testing.
(d) A fixed frequency (2.048 kHz) can be output to P23/BUZ. It can be used for sounding the buzzer and trimming the system clock oscillation frequency.
(e) Since the scaler can be cleared, the watch can be started at zero seconds.

5.4.1 Watch timer configuration

Fig. 5.4–1 shows a watch timer block diagram.



The values enclosed in parentheses are applied when $f_X$ = 4.194304 MHz and $f_{XT}$ = 32.768 kHz.

**Figure 5.4–1 Watch Timer Block Diagram**

5.4.2 Watch mode register

The watch dog timer (WM) consists of eight bits to control the watch timer. Fig. 5.4–2 shows the watch mode register format. The watch mode register (except bit 3) is set using an 8-bit operation instruction. Bit 3 is used to test the XT1 pin input level. The input level to the XT1 pin can be tested by making a bit test. When the RESET signal is generated, all bits except bit 3 are cleared.

Example: To produce time using the main system clock (4.19 MHz). To enable buzzer output.

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

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| F98H | WM7 | 0 | 0 | 0 | WM3 | WM2 | WM1 | WM0 | WM |

Count clock ($f_W$) selection bit

| | | |
|---|---|---|
| WM0 | 0 | System clock dividing output $\frac{f_X}{128}$ is selected. |
| | 1 | Subsystem clock $f_{XT}$ is selected. |

Operation mode selection bit

| | | |
|---|---|---|
| WM1 | 0 | Normal watch mode (IRQW is set by using $\frac{f_W}{2^{14}}$ , 0.5 s) |
| | 1 | Rapid feed watch mode (IRQW is set by using $\frac{f_W}{2^7}$ , 3.91 ms) |

Watch operation enable/disable bit

| | | |
|---|---|---|
| WM2 | 0 | Watch operation is stopped (divider is cleared). |
| | 1 | Watch operation is enabled. |

Input level to XT1 pin (bit test only is enabled)

| | | |
|---|---|---|
| WM3 | 0 | Input level to XT1 pin is low. |
| | 1 | Input level to XT1 pin is high. |

BUZ output enable/disable bit

| | | |
|---|---|---|
| WM7 | 0 | BUZ output is disabled. |
| | 1 | BUZ output is enabled. |

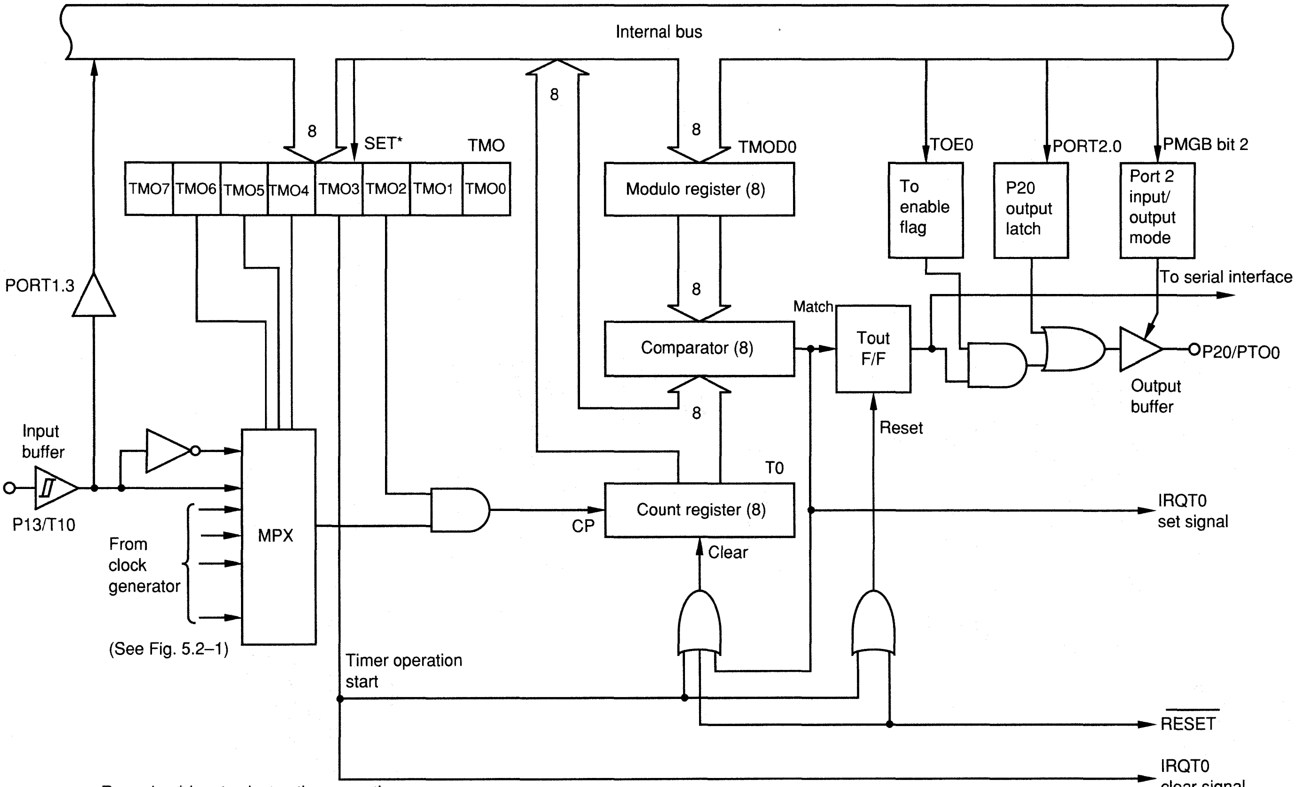
**Figure 5.4–2 Watch Mode Register Format**

### 5.5 Timer/event counter configuration

The μPD75328 incorporates a timer/event counter (one channel) as shown in Fig. 5.5–1.
The timer/event counter functions are as follows:

(a) Programmable interval timer operation
(b) Any desired frequency square wave output to PTO0 pin
(c) Event counter operation
(d) TI0 pin input is divided by N output to PTO0 pin (scaler operation).
(e) Real shift clock supply to serial interface circuit
(f) Count read function

Figure 5.5–1 Timer/Event Counter Block Diagram



Remarks: *denotes instruction executing

5.5.2 Basic configuration and operation of timer/event counter

The timer/event counter operation mode can be selected by using the timer/event counter mode register (TM0). The basic configuration and operation of the timer/event counter are explained below:

(1) Count pulse CP is selected by setting TM0 and input to the 8-bit count register T0.

(2) T0 is a binary 8-bit up counter incremented by one when CP is input. It is cleared when the $\overline{\text{RESET}}$ signal is generated, TM0 bit 3 is set (timer start), or coincidence signal is generated. T0 can be read at any time using an 8-bit memory operation instruction, but cannot be written

(3) The modulo register TMOD0 consists of eight bits to determine the T0 count. A value is set in TMOD0 using an 8-bit memory operation instruction, but TMOD0 cannot be read. When the $\overline{\text{RESET}}$ signal is generated, TMOD0 is initialized to FFH

(4) The comparator compares the T0 and TMOD0 contents. If they match, it generates a coincidence signal and sets the interrupt request flag (IRQT0)

Fig. 5.5-2 shows the count operation timing.

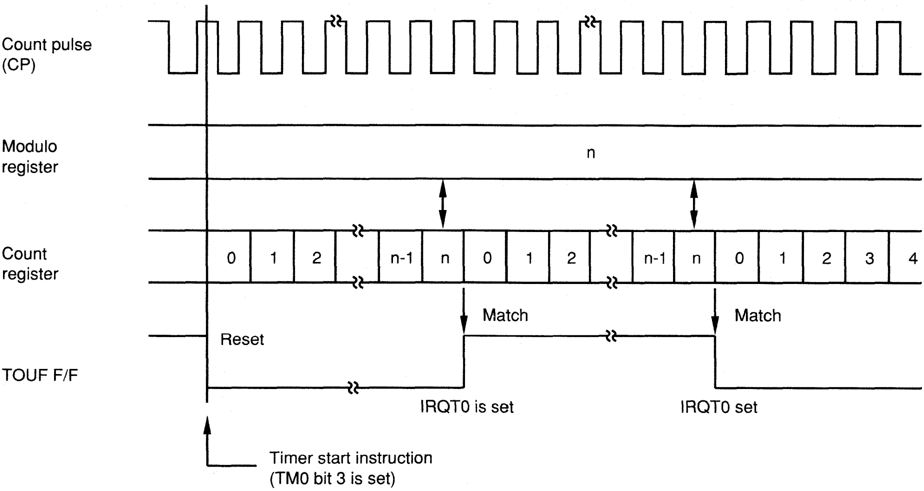


**Figure 5.5–2 Count Operation Timing Chart**

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

The mode register (TM0) consists of eight bits to control the timer/event counter. Fig. 5.5–3 shows the timer/event counter mode register format.
The timer mode register is set by using an 8-bit memory operation instruction.
Bit 3 is a timer start bit which can be set individually. Bit 3 is reset to 0 automatically when timer operation starts.

Example 1: To start the timer in interval timer mode with CP = 4.09 kHz.

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

Example 2: To restart the timer according to how the timer mode register is set.

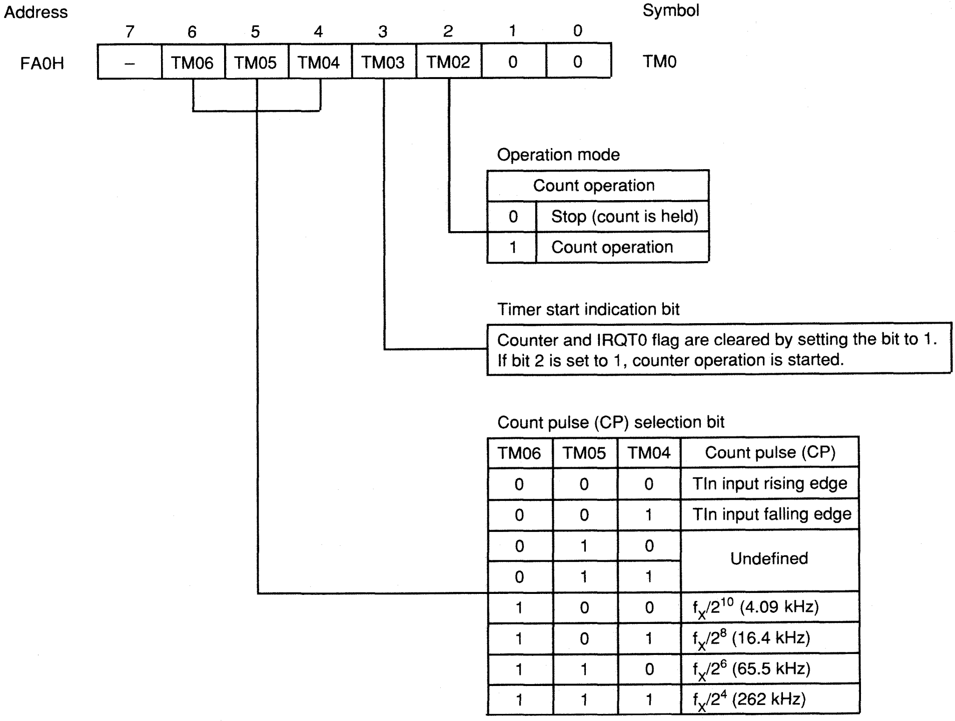
```
SEL     MB15    ;   or CLR1 MBE
SET1    TM0.3   ;   TM0.BIT3 ← 1
```

When the $\overline{\text{RESET}}$ signal is generated, all the timer mode register bits are cleared.

The timer/event counter output enable flag (TOE0) controls enable/disable of outputting the timer out F/F (TOUT F/F) state to the PTO0 pin. (See Fig. 5.5–4.) It is operated by using bit operation instruction and is enable to be writing.

The timer out F/F (TOUT F/F) is inverted by the coincidence signal received from the comparator. The timer out F/F is reset using an instruction to set timer mode register (TM0) bit 3.

When the $\overline{\text{RESET}}$ signal is generated, TOE0 and TOUT F/F are cleared.



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

Operation mode

| Count operation | |
|---|---|
| 0 | Stop (count is held) |
| 1 | Count operation |

Timer start indication bit

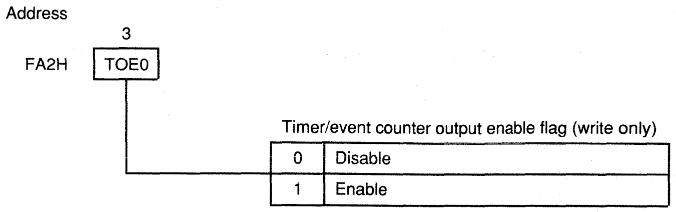
Counter and IRQT0 flag are cleared by setting the bit to 1. If bit 2 is set to 1, counter operation is started.

Count pulse (CP) selection bit

| TM06 | TM05 | TM04 | Count pulse (CP) |
|---|---|---|---|
| 0 | 0 | 0 | TIn input rising edge |
| 0 | 0 | 1 | TIn input falling edge |
| 0 | 1 | 0 | Undefined |
| 0 | 1 | 1 | |
| 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) |

Remarks: The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz.

**Figure 5.5–3 Timer/Event Coumter Mode Register Format**



Timer/event counter output enable flag (write only)

| | |
|---|---|
| 0 | Disable |
| 1 | Enable |

**Figure 5.5–4 Timer/Event Counter Output Enable Flag Format**

5.5.4 Timer/event counter operation mode

The timer/event counter operates in the count operation stop or count operation mode depending on how the mode register is set.

The following are always enabled independently of how the mode register is set

1) TI0 pin signal input and test. (P13 pin input test can be made.)
2) Output of timer out F/F state to PTO0.
3) Modulo register (TMOD0) setting.
4) Count register (T0) read.
5) Interrupt request flag (IRQT0) setting, clear, and test

(a) Count operation stop mode
The count operation stop mode is set when TM0 bit 2 is set to 0. Since count pulse (CP) supply to the count register is stopped, count operation is not performed

(b) Count operation mode
The count operation mode is set when TM0 bit 2 is set to 1. Count pulses selected by using bits 4 to 6 are supplied to the count register, and count operation is performed as shown in Fig. 5.5–2.
Normally, timer operation is started by

1) setting a count value in the modulo register (TMOD0), then
2) setting the operation mode, count clock, and start indication in the mode register (TM0).

The modulo register is set by using an 8-bit data transfer instruction.

Caution: Set a value other than 0 in the modulo register.

Example: To set 3FH in channel 0 modulo register.

```
SEL    MB15          ;   or CLR1 MBE
MOV    XA, #3FH
MOV    TMOD0, XA
```

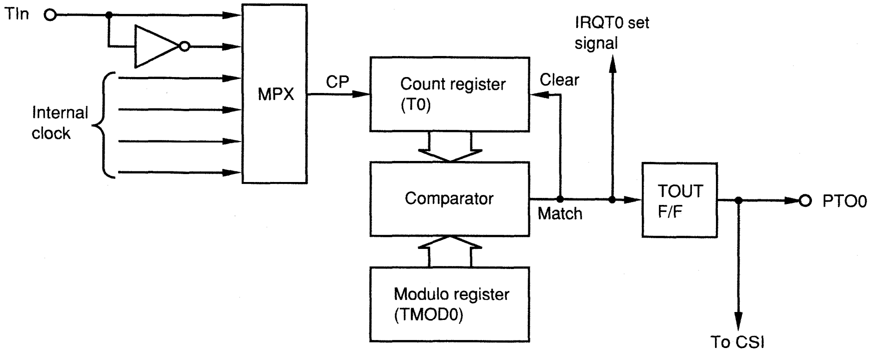


**Figure 5.5–5 Operation in Count Operation Mode**

5.5.5 Timer/event counter time setting

"Timer setup value" (period) is equal to "modulo register contents + 1" divided by "count pulse frequency" selected by setting the timer mode register.

$$T\ (s) = \frac{n+1}{F_{CP}}$$

T (s) : Timer setup value (seconds)
$F_{CP}$ (Hz) : Count pulse frequency (Hz)
n : Modulo register value ($n \neq 0$)

Once the timer is set, an interrupt request signal (IRQT0) is generated at the setup time intervals.

Table 5.5–1 lists the resolution and maximum setup value (time when FFH is set in the modulo register) of the timer/event counter for each count pulse.

Example: To produce 30 ms time intervals. ($f_X$ = 4.194304 MHz)
Use the mode with the maximum setup time 62.5 ms.

$$\frac{30\ \text{ms}}{244\ \mu\text{s}} = 122.9 = 7\text{AH}$$

Set 79H in the modulo register.

```
SEL    MB15
MOV    XA, #79H
MOV    TMOD0, XA
```

**Table 5.5–1 Resolution and Maximum Setup Value (4.19 MHz)**

| Mode register | | | Timer channel 0 | |
|---|---|---|---|---|
| TMO6 | TMO5 | TMO4 | Resolution | Maximum setup time |
| 1 | 0 | 0 | 244 μs | 62.5 ms |
| 1 | 0 | 1 | 61.1 μs | 15.6 ms |
| 1 | 1 | 0 | 15.3 μs | 3.91 ms |
| 1 | 1 | 1 | 3.81 μs | 977 μs |

5.5.6 Caution on timer/event counter application

(1) Error at timer start

An error with a maximum length of one clock period of the count pulse (CP) for the value calculated in 5.5.5 occurs in the time until a coincidence signal is generated after the timer starts (TM0.3 is set). This is because clearing of the count register is not synchronized with CP as shown in Fig. 5.5–6.

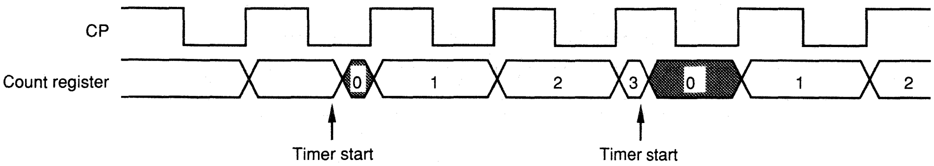


**Figure 5.5–6 Error at Timer Starting**

(2) Caution at timer start

Normally, the count register T0 and interrupt request flag IRQT0 are cleared when the timer starts (TM0 bit 3 is set). However, if the timer is placed in the operation mode, and IRQT0 setting and timer start occur simultaneously, IRQT0 may be unable to clear. There is no problem when IRQT0 is used for a vectored interrupt. In IRQT0 test application, however, a problem arises such than IRQT0 is set although the timer has been started. Thus, to start the timer at IRQT0 timing, stop the timer once (by setting TM0 bit 2 to 0), then restart it, or start the timer twice.

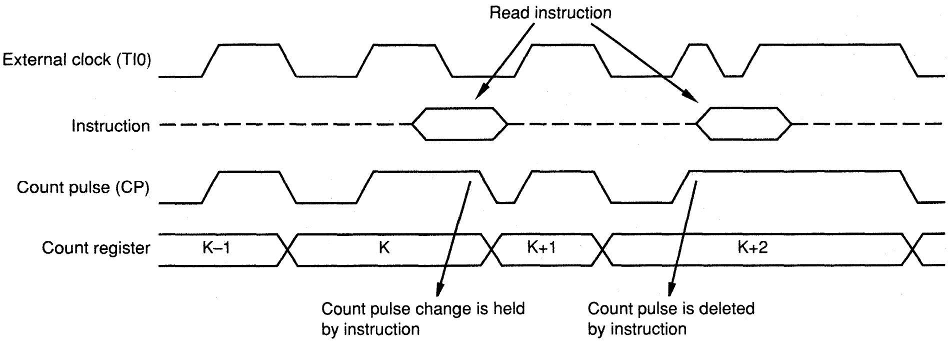
Example: Timer start at IRQT0 timing may be set

```
        SEL     MB15        ;   or CLR1 MBE
        MOV     XA, #0
        MOV     TM0, XA     ;   Timer stop
        MOV     XA, #4CH
        MOV     TM0, XA     ;   Restart
    or
        SEL     MB15        ;   or CLR1 MBE
        SET1    TM0.3
        SET1    TM0.3       ;   Restart
```

(3) Error at count register read

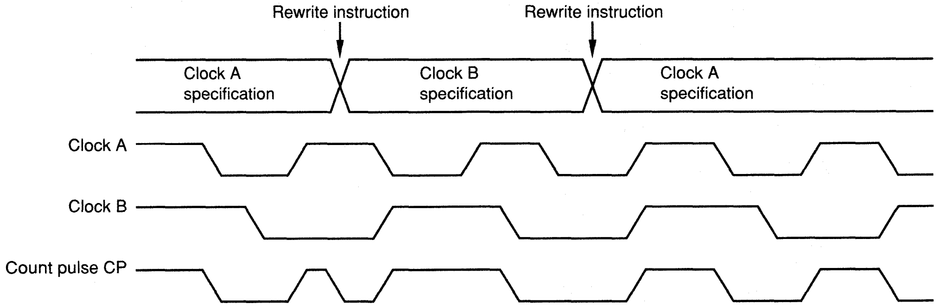
The count register contents can be read at any time using an 8-bit data memory operation instruction. During execution of the instruction, count pulse change is held and count register change is suppressed. Thus, if the count pulse signal source is input to TI0, count pulses as long as the instruction execution time are deleted. (This symptom does not occur if the internal clock is used for count pulses because they are synchronized with the instruction.)

Therefore, if TI0 is input as a count pulse and the count register contents are read, signals having a pulse width that prevents miscount, even if count pulses are deleted, must be input. Since the count hold period in read instruction execution is one machine cycle, the pulse width input to the TI0 pin must be longer than one machine cycle.

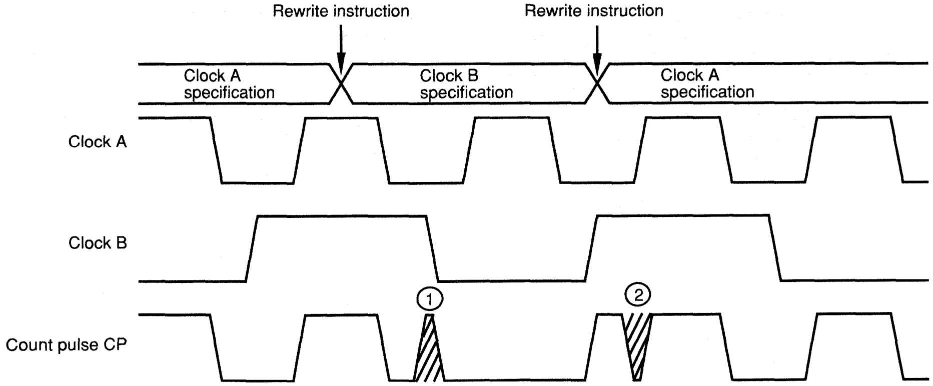


(4) Caution at count pulse change

If the timer mode register is rewritten and count pulse change is made, its specification becomes effective immediately after instruction execution.

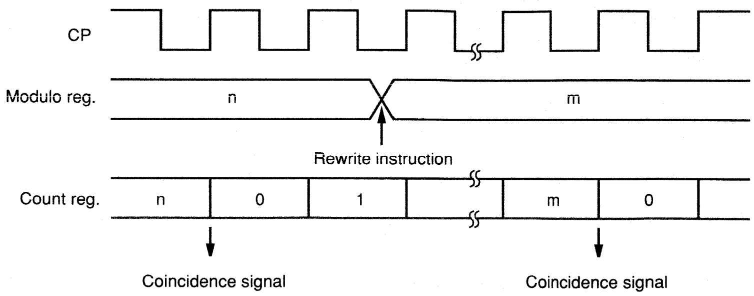


A count pulse ( ① or ② in the timing chart shown below) may occur depending on the clock combination when the count pulse changed. In this case, a miscount may occur or the count register contents may be destroyed. To change count pulse, be sure to set count mode register bit 3 to 1 and restart the timer at the same time.

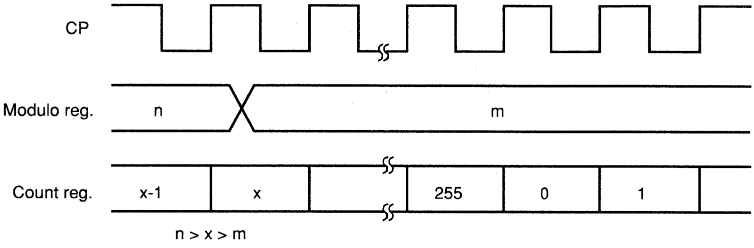


(5) Operation after modulo register change

The modulo register is changed when an 8-bit data memory operation instruction is executed.



If the value appearing after the modulo register is changed and becomes smaller than the count register value, the count register continues counting and overflows, then restarts counting from 0. Thus, if the value after the modulo register is changed (m) and becomes smaller than the value before it was changed (n), the timer must be restarted after the modulo register is changed.



5.5.7 Timer/event counter application

(1) Timer 0 is applied to an interval timer generating an interrupt at 50 ms intervals.

- The high-order four bits of the mode register are set to 0100B and a maximum setup time of 62.5 ms is selected.
- The low-order four bits of the mode register are set to 1100B.
- The modulo register setup value is as follows:

$$\frac{50\text{ ms}}{244\ \mu\text{s}} = 205 = \text{CDH}$$

```
Example:  SEL   MB15         ; or CLR1 MBE
          MOV   XA, #0CCH
          MOV   TMOD0, XA    ; Modulo is set.
          MOV   XA, #01001100B
          MOV   TM0, XA      ; Mode is set and timer is started.
          EI                 ; Interrupts is enabled.
          EI    IET0         ; Timer interrupt is enabled.
```

Remarks: In this application, the TI0 pin can be used as an input pin.

(2) When the number of pulses input from the TI0 pin reaches 100, an interrupt is generated. (The pulses are active high.)
- — The high-order four bits of the mode register are set to 0000 and the rising edge is selected.
- — The low-order four bits of the mode register are set to 1100B.
- — The modulo register is 99 = 100 –1.

```
Example: SEL   MB15          ; or CLR1 MBE
         MOV   XA, #100 –1
         MOV   TMOD0, XA     ; Modulo is set.
         MOV   XA, #00001100B
         MOV   TM0, XA
         EI
         EI    IET0          ; INTT0 is enabled.
```

### 5.6 Serial Interface

5.6.1 Serial interface configuration

The μPD75328 has a clocked serial interface (CSI) as shown in Fig. 5.6-2.

5.6.2 Serial interface functions

The μPD75328 serial interface includes the three modes described in (1) to (3) below.

Since the serial clock line $\overline{SCK}$ and serial data bus lines SB0/SB1 enable software to determine the output level, any desired transfer format can be handled.

(1) 3-line serial I/O mode
- — Three lines of serial clock $\overline{SCK}$, serial output SO, and serial input SI
- — Clock synchronous 8-bit send and receive (simultaneous send and receive)
- — The serial transfer top can be changed between the most and least significant bits (MSB and LSB).
- — This mode enables the μPD75328 to be connected to the μPD7500 series, μCOM-75X family, μCOM-87 family, and various peripheral I/O devices.

(2) SBI (serial bus interface) mode
- — This mode conforms to the NEC serial bus format.
- — Communication can be made with a number of devices by using the serial clock $\overline{SCK}$ and serial data bus SB0 or SB1 lines.
- — Address, command, and data can be transferred, and a hardware function for discriminating the signals is included. (See Fig. 5.6-2.)
- — The acknowledge, busy signal output function and the wake-up function for handshaking are included.

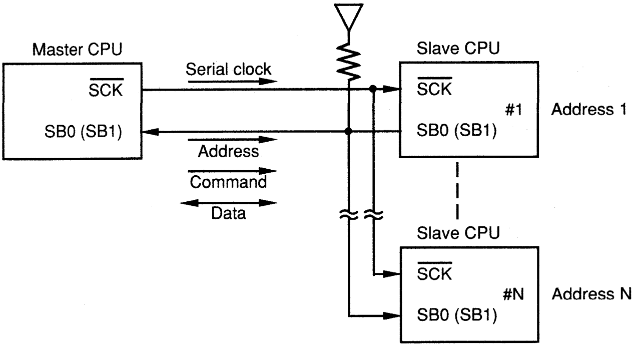


**Fig. 5.6-1 SBI System Configuration Example**

(3) 2-line serial I/O mode
- — Communication can be made by using the serial clock $\overline{SCK}$ and serial data bus SB0 or SB1 lines.
- — Communication can be made with a number of devices by using software to control the output level to the two lines. Any desired communication format can be handled.

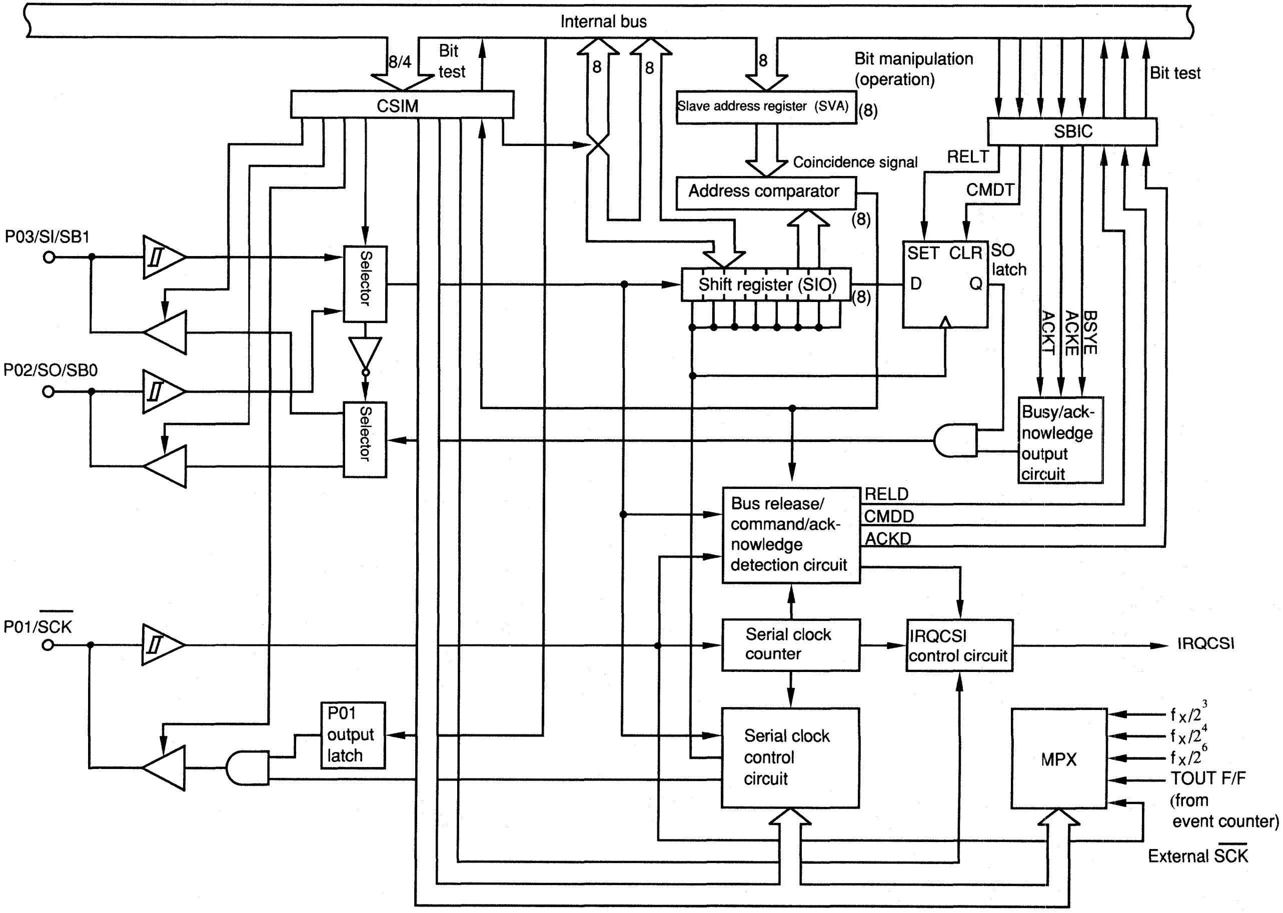
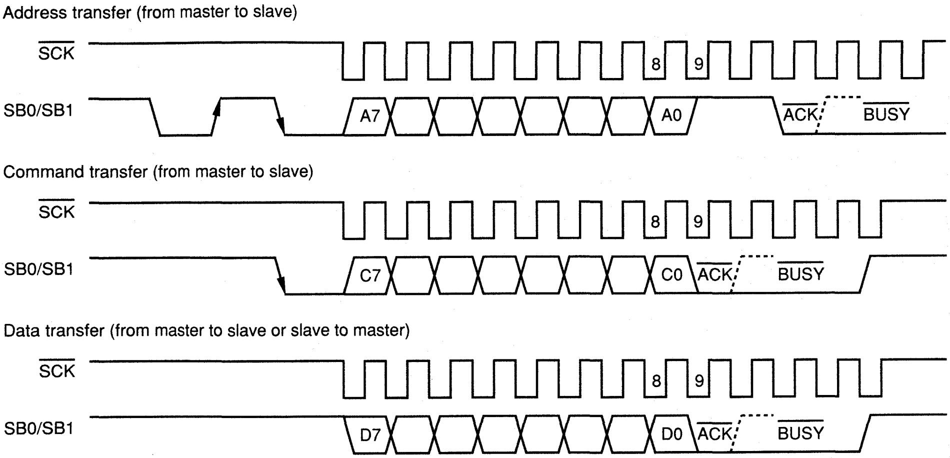


Figure 5.6.2 Serial Interface Block Diagram



Address transfer (from master to slave)

Command transfer (from master to slave)

Data transfer (from master to slave or slave to master)

**Figure 5.6–3 SBI Transfer Timing Chart**

5.6.3 Main register functions

(1) Serial operating mode register (CSIM)

The serial operation mode register (CSIM) consists of eight bits that specify the serial interface operating mode, serial clock, wake-up function, etc.
CSIM is set using an 8-bit memory operation instruction.
The high-order three bits of CSIM can be set bitwise using the bit name.
Bit 6 enables bit test only. Data written into the bit becomes invalid.

Example 1:
To select $f_X/2^4$ for serial clock, generate a serial interrupt IRQCSI at the end of each serial transfer, and make serial transfer in the SBI mode using the SB0 pin as the serial data bus line.

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

Example 2:
To enable serial transfer conforming to the CSIM contents.

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

Wake-up function specification bit (WUP):

(a) When WUP=0
IRQCSI is set each time a serial transfer ends.
WUP is set to 0 during normal transfer.

(b) When WUP = 1

WUP is set to 1 only during the SBI mode. IRQCSI is set only when the address received after the bus is released matches the slave address register (SVA) value (wake-up state). When the received address is not its own, an unnecessary interrupt will not be generated. The $\overline{\text{ACK}}$ signal is not output either. The SB0 (SB1) pin is placed in a high impedance state independent of the SO the latch state.

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

Fig. 5.6–4 shows the format of the serial operation mode register.

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FE0H | CSIE | COI | WUP | CSIM4 | CSIM3 | CSIM2 | CSIM1 | CSIM0 | CSIM |

Serial clock selection bits (W)

| CSIM1 | CSIM0 | Serial clock | | | $\overline{\text{SCK}}$ pin mode |
|---|---|---|---|---|---|
| | | 3-line serial I/O mode | SBI mode | 2-line serial I/O mode | |
| 0 | 0 | $\overline{\text{SCK}}$ pin input clock from the external | | | Input |
| 0 | 1 | Timer/event counter output (TO) | | | Output |
| 1 | 0 | $f_X/2^4$ (262 kHz) | | $f_X/2^6$ (65.5 kHz) | |
| 1 | 1 | $f_X/2^3$ (524 kHz) | | | |

The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz.

Serial interface operating mode selection bits (W)

| CSIM4 | CSIM3 | CSIM2 | Operating mode | Shift register bit order | SO pin function | SI pin function |
|---|---|---|---|---|---|---|
| X | 0 | 0 | 3-line serial I/O mode | SIO$_{7-0}$ ↔ XA<br>(transfer starts at MBS) | SO/PO2<br>(CMOS output) | SI/PO3<br>(input) |
| | | 1 | | SIO$_{0-7}$ ↔ XA<br>(transfer starts at LSB) | | |
| 0 | 1 | 0 | SBI mode | SIO$_{7-0}$ ↔ XA<br>(transfer starts at MSB) | SB0/PO2<br>(N-channel open drain input/output) | PO3 input |
| 1 | | | | | PO2 input | SB1/PO3<br>(N-channel open drain input/output) |
| 0 | 1 | 1 | 2-line serial I/O mode | SIO$_{7-0}$ ↔ XA<br>(transfer starts at MSB) | SB1/PO2<br>(N-channel open drain input/output) | PO3 input |
| 1 | | | | | PO2 input | SBI/PO3<br>(N-channel open drain input/output) |

(to be continued)

**Fig. 5.6-4 Serial Operation Mode Register Format**

Wake-up function specification bit (W)

| | | |
|---|---|---|
| WUP | 0 | IRQCSI is set each time a serial transfer ends in each mode. |
| | 1 | WUP is set to 1 only during the SBI mode. IRQCSI is set only when the address received after the bus is released matches the slave address register data (wake-up state). SB0 (SB1) is placed in the high impedance state |

Coincidence signal received from address comparator (R (bit test is only enabled)) Note 1, Note 2

| | | |
|---|---|---|
| COI | 0 | Slave address register and shift register data mismatch. |
| | 1 | Slave address register and shift register data match. |

Serial interface operation enable/disable specification bit (W) Note 3

| | | Shift register operation | Serial clock counter | IRQCSI flag | SO/SB0 and SI/SB1 pins |
|---|---|---|---|---|---|
| CSIE | 0 | Shift operation is disabled. | Clear | Hold | Port 0 function only |
| | 1 | Shift operation is enabled. | Count operation | Can be set. | Also used for port 0 according to each mode function |

**Figure 5.6–4 Serial Operation Mode Register Format (cont'd)**

Notes:
1. COI is only valid before serial transfer starts or after it is complete. An undefined value is read during serial transfer.
2. Data written into COI is ignored.
3. To use PO1/$\overline{\text{SCK}}$ pin as an input port, set the following below.
   - Set CSIM0 and CSIM1 bits to 0. (PO1/$\overline{\text{SCK}}$ pins are set to input mode.)
   - Set CSIE bit to 0. (Stop the serial interface operation.)

   When CSIM0=CSIM1=0 and CSIE=1 or CSIM0–CSIM1≠0 and CSIE=0, PO1/$\overline{\text{SCK}}$ pins are output to high level.

Remarks: (W): Data write is only enabled.
(R) : Data read is only enabled.

(2) Serial bus interface control register (SBIC)

The 8-bit serial bus interface control register (SBIC) consists of the serial bus state control bits and flags indicating the states of input data from serial bus. It is mainly used in the SBI mode.

Fig. 5.6–5 shows the SBIC format. SBIC is set or tested by using a bit operation (manipulation) instruction. When the $\overline{\text{RESET}}$ signal is generated, all the SBIC bits are cleared.

Cautions:
1. SBIC cannot be set by using 4- or 8-bit memory operation instruction.
2. In the 3- or 2-line serial I/O mode, use only the following two bits for SO latch control:
   (a) Bus release trigger bit (RELT): To set SO latch
   (b) Command trigger bit (CMDT): To clear SO latch
3. For the bus release, command, acknowledge, and busy signals, see 5.6.5 (3).

Example 1: To output command signal.

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

Example 2: To test RELD and CMDD and determine the receive data type for appropriate processing.

```
        SEL     MB15
        SKF     RELD    ;   RELD test
        BR      !ADRS
        SKT     CMDD    ;   CMDD test
        BR      !DATA
CMD:    ...     ...     ;   Command transfer
DATA:   ...     ...     ;   Data transfer
ADRS:   ...     ...     ;   Address transfer
```

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FE2H | BSYE | ACKD | ACKE | ACKT | CMDD | RELD | CMDT | RELT | SBIC |

Bus release trigger bit (W)

| | |
|---|---|
| RELT | SO latch is set to 1 by setting the bit. It it used to output bus release signal. After SO latch is set, the bit is automatically cleared. |

Command trigger bit (W)

| | |
|---|---|
| CMDT | SO latch is cleared by setting the bit. It is used to output a command signal. After SO latch is cleared, the bit is automatically cleared. |

Bus release detection flag (R)

| | | |
|---|---|---|
| RELD | Clear condition (RELD=0) | 1 When the transfer start is indicated.<br>2 When the address received after the bus is released dies not match the slave address register (SVA) data.<br>3 When the $\overline{\text{RESET}}$ signal is input. |
| | Setting condition (RELD=1) | When the address received after the bus is released matches the slave address register data. (Wake up) |

Command detection flag (R)

| | | |
|---|---|---|
| CMDD | Clear condition (CMDD=0) | 1 When the transfer start is indicated.<br>2 When the bus release signal is detected.<br>3 When the $\overline{\text{RESET}}$ signal is input. |
| | Setting condition (CMDD=1) | When the command signal is detected. |

Acknowledge trigger bit (W)

| | | |
|---|---|---|
| ACKT | Used only after transfer completion | Acknowledge signal is output during one clock period of $\overline{\text{SCK}}$ immediatly after execution of the set instruction. |

Remarks:
1. ACKT is automatically cleared after acknowledge signal is output. 2. ACKT cannot be cleared using software.
3. To set ACKT, set ACKE to 0.

Acknowledge enable bit (R/W)

| | | | |
|---|---|---|---|
| ACKE | 0 | Automatic acknowledge signal output is disabled (the signal can be output by setting the acknowledge trigger bit ACKT). | |
| | 1 | Before transfer completion | Acknowledge signal is output during the ninth clock period of $\overline{\text{SCK}}$ (automatically output by presetting ACKE to 1). |
| | | After transfer completion | Acknowledge signal is output during one clock period of $\overline{\text{SCK}}$ immediately after execution of the set instruction (automatically output by presetting ACKE to 1). |

Acknowledge detection flag (R)

| | | |
|---|---|---|
| ACKD | Clear condition (ACKD=0) | 1 When the transfer is started.<br>2 When the $\overline{\text{RESET}}$ is input. |
| | Setting conditions (ACKD=1) | When the acknowledge signal is detected. |

Synchronous busy enable bit (R/W)

| | | |
|---|---|---|
| BSYE | 0 | Synchronous busy signal output is disabled. Synchronous busy signal output is stopped in synchronization with the $\overline{\text{SCK}}$ falling edge immediately after execution of the clear instruction. |
| | 1 | Synchronous busy signal is output on the $\overline{\text{SCK}}$ falling edge following an acknowledge signal. |

Remarks:
(R): Read is only enabled. (W): Write is only enabled. (R/W): Both read and write are enabled.

**Figure 5.6–5 SBIC Format**

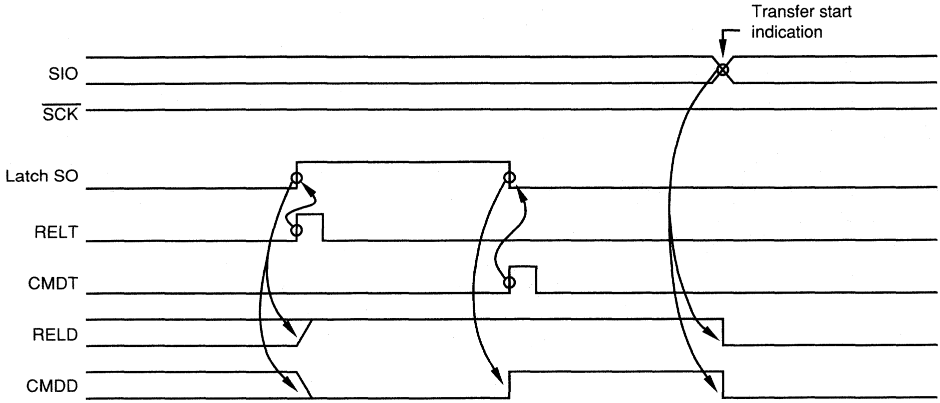


**Figure 5.6–6 RELT, CMDT, RELD, CMDD Operation**

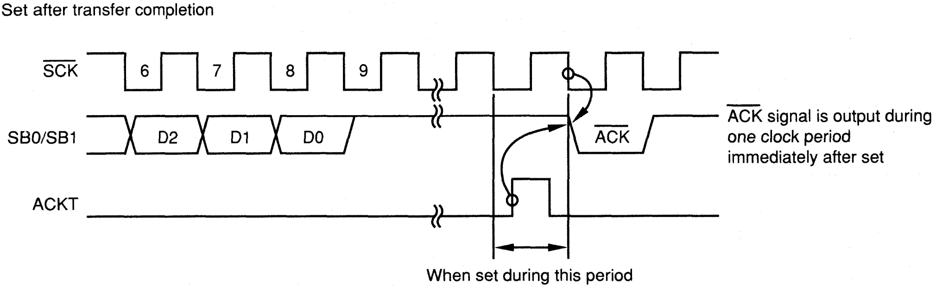
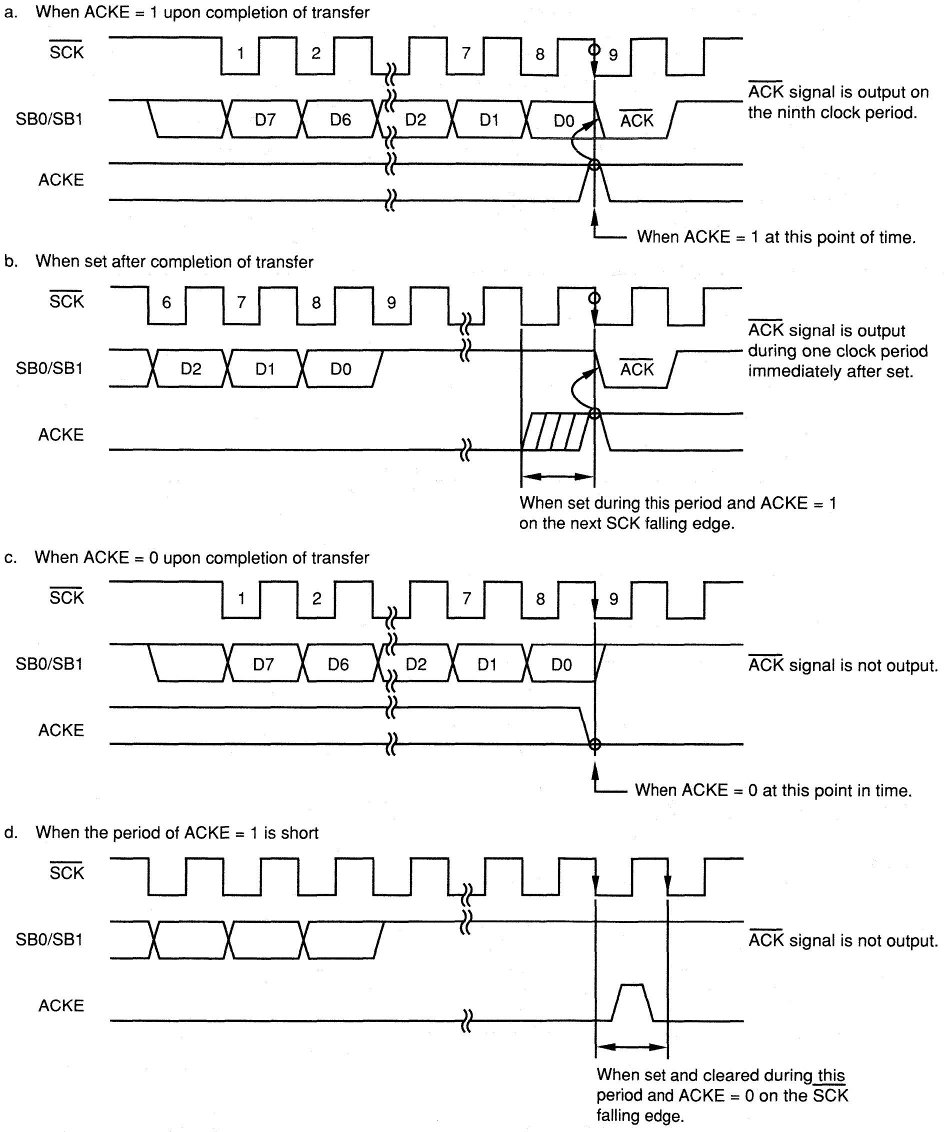

Set after transfer completion
$\overline{SCK}$
6
7
8
9
SB0/SB1
D2
D1
D0
$\overline{ACK}$
ACKT
$\overline{ACK}$ signal is output during one clock period immediately after set
When set during this period

**Figure 5.6–7 ACKT Operation**



a. When ACKE = 1 upon completion of transfer

$\overline{\text{ACK}}$ signal is output on the ninth clock period.

When ACKE = 1 at this point of time.

b. When set after completion of transfer

$\overline{\text{ACK}}$ signal is output during one clock period immediately after set.

When set during this period and ACKE = 1 on the next SCK falling edge.

c. When ACKE = 0 upon completion of transfer

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

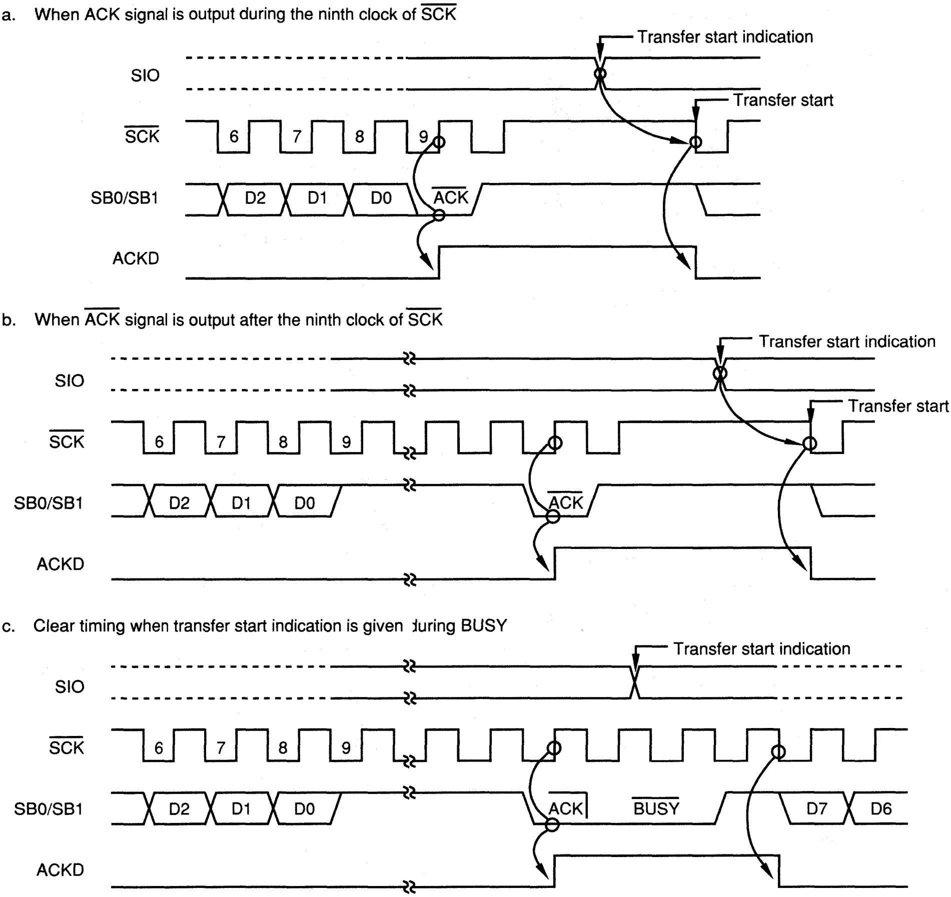
When ACKE = 0 at this point in time.

d. When the period of ACKE = 1 is short

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

When set and cleared during this period and ACKE = 0 on the $\overline{\text{SCK}}$ falling edge.

**Figure 5.6–8 ACKE Operation**

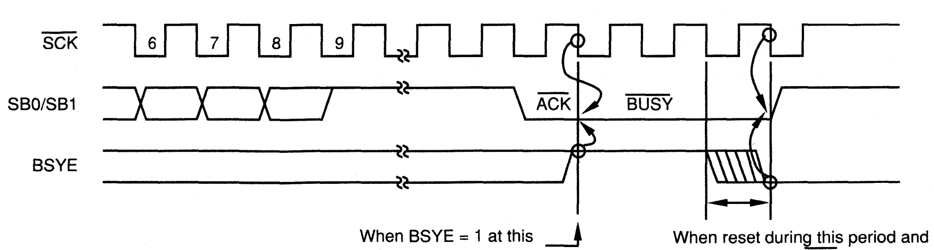


a. When ACK signal is output during the ninth clock of $\overline{\text{SCK}}$

b. When $\overline{\text{ACK}}$ signal is output after the ninth clock of $\overline{\text{SCK}}$

c. Clear timing when transfer start indication is given during BUSY

**Fig. 5 6-9 ACKD Operation**



When BSYE = 1 at this point in time

When reset during this period and BSYE = 0 on the $\overline{\text{SCK}}$ falling edge

**Figure 5.6–10 BSYE Operation**

(3) Slave address register (SVA)

When the μPD75328 is connected to the serial bus as a slave device in the SBI mode, the 8-bit slave address register (SVA) is used to set the slave address of the μPD75328.

The SVA value is compared with received 8-bit data by the address comparator. If a match is found between them, serial operating mode register (CSIM) bit 6 (COI) is set to 1.

If a match is not found when the address is received, the bus release detection flag (RELD) is cleared. When WUP = 1, IRQCSI is set only if a match is found (wake-up is detected). This interrupt request can be used to know that a communication request is sent from the master to μPD75328.

SVA can also be used to detect an error when the μPD75328 sends address, command, or data as a master device. (See 5.6.6.)

SVA is set using an 8-bit memory operation instruction. It can only be written.

When the RESET signal is generated in a mode which is not standby, the SVA value becomes undefined.

5.6.4 Signals in SBI mode

Tables 5.6–1 and 5.6–2 list the signals used in the SBI mode.

**Table 5.6–1 Signals in SBI Mode (I)**

| Signal name | Output device | Definition | Timing chart | Output condition | Flag influence | Explaination |
|---|---|---|---|---|---|---|
| Bus release signal (REL) | Master | SB0 (SB1) rising edge when $\overline{\text{SCK}}$ = 1 | (Figure 1) | • RELT is set. | • RELD is set.<br>• CMDD is cleared. | The signal is followed by CMD signal output indicating that the send data is an address |
| Command signal (CMD) | Master | SB0 (SB1) falling edge when $\overline{\text{SCK}}$ = 1 | (Figure 2) | • CMDT is set. | • CMDD is set. | i) After REL signal is output, send data is an address.<br>ii) When REL signal is not output, send data is a command. |
| Acknowledge signal ($\overline{\text{ACK}}$) | Master/ Slave | Low signal output to SB0 (SB1) during one clock period of $\overline{\text{SCK}}$ after completion of serial reception | (Figure 3) | 1 ACKE = 1<br>2 ACKT is set. | • ACKD is set. | Completion of reception |
| Busy signal ($\overline{\text{BUSY}}$) | Slave | (Synchronous busy signal) Low signal output to SB0 (SB1) following acknowledge signal | | • BSYE = 1 | – | Serial reception cannot be done because processing is being performed. |
| | | (Asynchronous busy signal) Low signal output to SB0 (SB1) (except during serial transfer). It is not synchronized with $\overline{\text{SCK}}$. | | • CMDT is set. | – | |
| Ready signal (READY) | Slave | High signal output to SB0 (SB1) before start or after completion of serial transfer | | 1 BSYE = 0<br>2 Execution of SIO data write instruction (transfer start indication) | – | Serial reception can be done. |

4

**Table5.6–2 Signals in SBI Mode (II)**

| Signal name | Output device | Definition | Timing chart | Output condition | Flag influence | Explanation |
|---|---|---|---|---|---|---|
| Serial clock (SCK) | Master | Synchronous clock to output address, command, data, ACK signal, synchronous BUSY signal, etc. Address, command, or data is transferred with the first eight. | (Figure 4) | Execution of SIO data write instruction when CSIE =1 (serial transfer start indication) (Note 2) | IRQCSI is set (on the rising edge of ninth clock). (Note 1) | Signal output timing to serial data bus |
| Address (A7–0) | Master | 8-bit data transferred in synchronization with SCK after REL and CMD signals are output. | (Figure 5) | | (Note 1) | Slave device address value on serial bus |
| Command (C7–0) | Master | 8-bit data transferred in synchronization with SCK after CMD signal only is output (REL signal is not output). | (Figure 6) | | None | Indication message sent to slave device |
| Data (D7–0) | Master or Slave | 8-bit data transferred in synchronization with SCK when neither REL nor CMD signal is output. | (Figure 7) | | None | Numeric data processed by slave or master device |

Notes: 1. When WUP = 0, IRQCSI is always set on the ninth clock SCK rising edge.
When WUP = 1, IRQCSI is set only when the received address matches the value in the slave address register (SVA).
2. In the BUSY state, transfer is started after the READY state is set.

Figure 1:

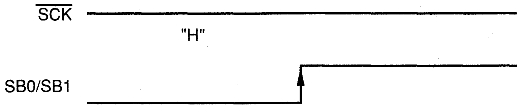


Figure 2:

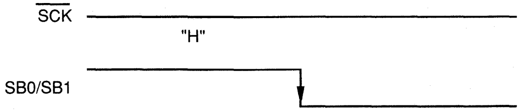


Figure 3:

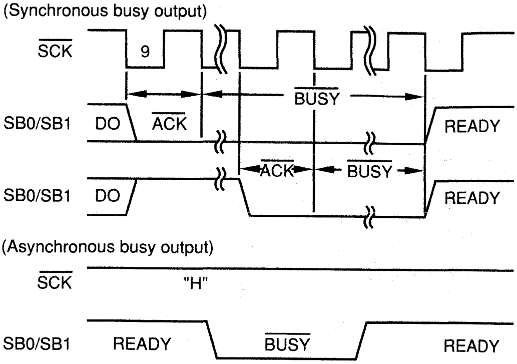


Figure 4:

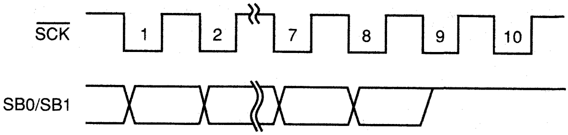


Figure 5:

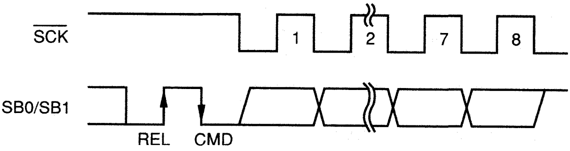


Figure 6:

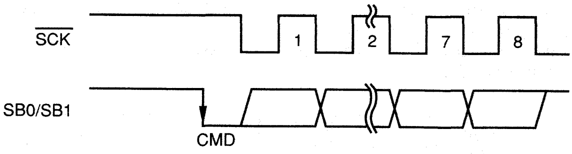
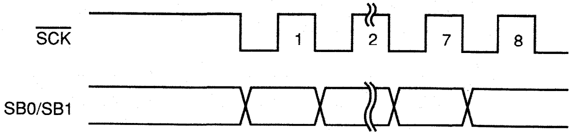


Figure 7:



5.6.5 Serial interface operation

The serial interface includes the following four operating modes.
— Operation stop mode
— 3-line serial I/O mode
— SBI mode
— 2-line serial I/O mode

**Table 5.6-3 Serial Interace Operation Mode**

| CSIM7 | CSIM3 | CSIM2 | Operating mode |
|---|---|---|---|
| 0 | — | — | Operation stop |
| 1 | 0 | — | 3-line serial I/O |
| 1 | 1 | 0 | SBI mode |
| 1 | 1 | 1 | 2-line serial I/O |

(1) Operation stop mode

When CSIE = 0, the serial interface is placed in the operation stop mode. In this mode, no serial transfer is done.
The operation stop mode is set when the serial interface is not used, so that the power consumption is reduced.
In this mode, the shift register does not perform shift operations and can be used as a normal 8-bit register.
All the P01/$\overline{\text{SCK}}$, P02/SO/SB0, and P03/SI/SB1 pins are placed in the high impedance state and used only for the input port function. The P01/$\overline{\text{SCK}}$ pin is placed in the state listed in Table 5.6-4 according to how CSIM1 and CSIM0 are set.

**Table 5.6-4 CSIM1 and CSIM0 Settings**

| CSIM1 | CSIM0 | Serial clock | P01/$\overline{\text{SCK}}$ pin state |
|---|---|---|---|
| 0 | 0 | External clock | High impedance |
| 0 | 1 | Internal clock | A high level is output. However, a serial clock is output during serial transfer. |
| 1 | 0 | | |
| 1 | 1 | | |

Note: For the serial clock, see Fig. 5.6-4.
When the $\overline{\text{RESET}}$ signal is generated, the operation stop mode is set.

(2) 3-line serial I/O mode

The 3-line serial I/O mode is compatible with the mode used by the μPD7500 series or other μCOM-75X family devices. Fig. 5.6-11 shows the 3-line serial I/O mode operation timing.
Serial transfer start is indicated by executing an instruction to write data into the shift register (SIO), MOV or XCH.
Be sure to give a start indication when CSIE = 1.

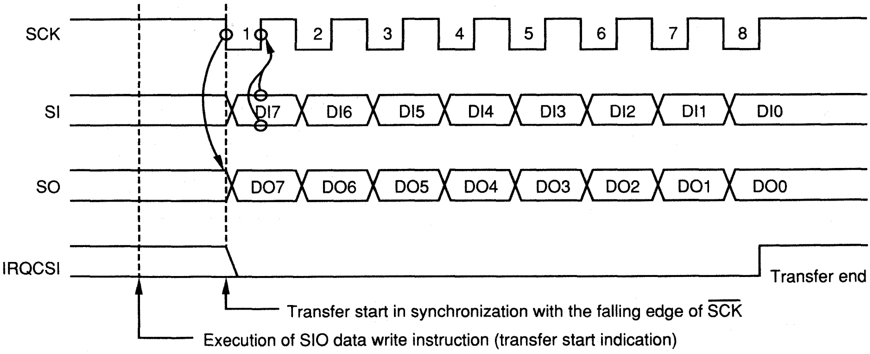

SCK
1
2
3
4
5
6
7
8
SI
DI7
DI6
DI5
DI4
DI3
DI2
DI1
DI0
SO
DO7
DO6
DO5
DO4
DO3
DO2
DO1
DO0
IRQCSI
Transfer end
Transfer start in synchronization with the falling edge of $\overline{\text{SCK}}$
Execution of SIO data write instruction (transfer start indication)

**Fig. 5.6-11 3-line Serial I/O Mode Timing**

In this mode, shift register shift operation is performed in synchronization with the serial clock (SCK) falling edge, and send data is held in the SO latch and output from the SO pin. The receive data input to the SI pin is latched in the shift register on the rising edge of SCK.

Since the SO pin is used as a CMOS output and outputs the SO latch state, the SO pin output state can be handled by setting the RELT and CMDT bits. However, do not try to change it during serial transfer.

The SCK pin output state can be controlled if the P01 output latch is operated in the output mode (internal clock mode), as described in 5.6.7.

A serial clock can be selected from among four clocks, as listed in Table 5.6-5, by setting the mode register.

Normally, shift operation is started by:

1) Setting the operating mode and serial clock selection data in the mode register (CSIM)

2) Setting transfer data in the shift register (SIO) (serial operation is started by executing the SIO data write instruction)

Serial transfer automatically stops at the end of 8-bit transfer, and the interrupt request flag (IRQCSI) is set.

**Table 5.6-5 Serial CLock Selection and Application**

| Mode register | | | Serial clock | Timing at which shift register can be read/written and serial transfer can be started | Application |
|---|---|---|---|---|---|
| CSIM 1 | CSIM 0 | Source | Serial clock mask | | |
| 0 | 0 | External SCK | Automatic mask at the end of 8-bit data transfer. | Only when serial transfer stops or SCK is high.<br>(Note) | Slave CPU |
| 0 | 1 | TOUT F/F | | | Half-duplex asynchronous transfer (under software control) |
| 1 | 0 | $f_X/2^4$ | | | Medium-speed serial transfer |
| 1 | 1 | $f_X/2^3$ | | | High-speed serial transfer |

Note: The serial transfer stops in the opertaion stop mode or when the serial clock is masked after 8-bit transfer has been made.

The shift register is read and written using 8-bit transfer instructions. At that time, LSB and MSB can be inverted by setting CSIM bit 2. This function enables the transfer top bit to be changed between LSB and MSB.

Eyample:

At the same time RAM data specified in the HL register pair is transfered to SIO, the SIO data is read into the accumulator, and serial transfer is started.

```
        MOV   XA, @HL       ; Send data is taken out of RAM.
        SEL   MB15          ; or CLR1 MBE
        XCH   XA, SIO       ; Send data and receive data are exchanged, and transfer is started.
```

Caution:

The transfer top bit LSB or MSB is selected by changing the bit order in writing data into the shift register. The shift order of the shift register bit is always the same. Thus, even if the transfer top bit is changed (LSB to MSB or MSB to LSB) after data is written into the shift register, the data is transferred in the bit order specified before the transfer top bit is changed.

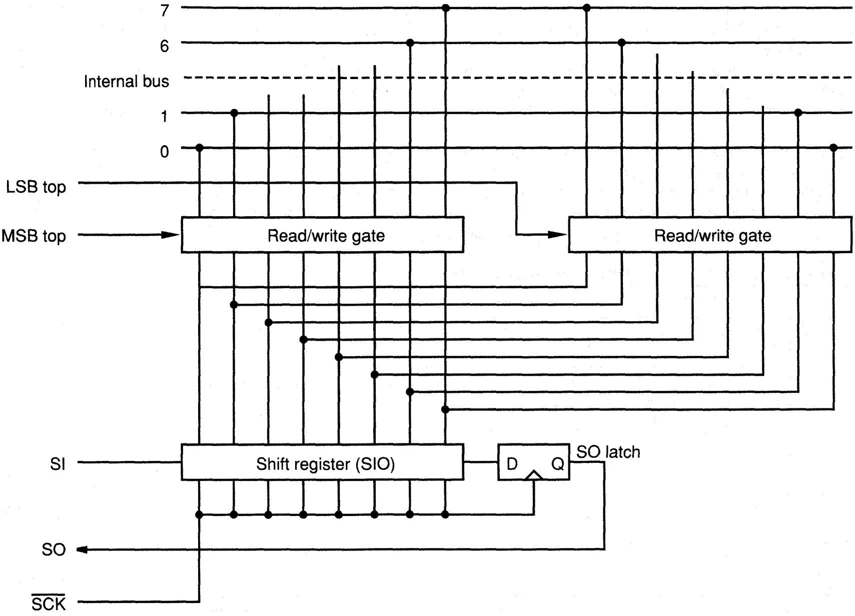


**Fig. 5.6-12 3-line I/O Mode Format**

(3) SBI mode
The SBI mode enables the μPD75328 to communicate with a number of devices by using the $\overline{SCK}$ and SB0 or SB1 lines. Figs. 5.6-13 to 5.6-15 show the timing of operations according to the type of data to be transferred.
In the SBI mode, indication of serial transfer start is also given by executing the instruction to write data into the shift register (SIO), MOV or XCH. Be sure to give a start indication when CSIE = 1.
Shift register shifting is made in synchronization with the serial clock ($\overline{SCK}$) falling edge, and send data is held in the SO latch and output from the SB0/P02 or SB1/P03 pin starting at MSB. Receive data input to the SB0 or SB1 pin is latched in the shift register on the rising edge of $\overline{SCK}$.
The SB0 or SB1 pin specified for the serial data bus is used as N-channel open drain input/output and needs pull-up. When data is received, the N-channel transistor must be turned off.
By writing FFH into SIO and shifting it, the N-channel transistor can always be turned off during transfer. When the wake-up function specification bit (WUP) is set to 1, however, the N-channel transistor is always turned off, and FFH need not be written into SIO before reception.

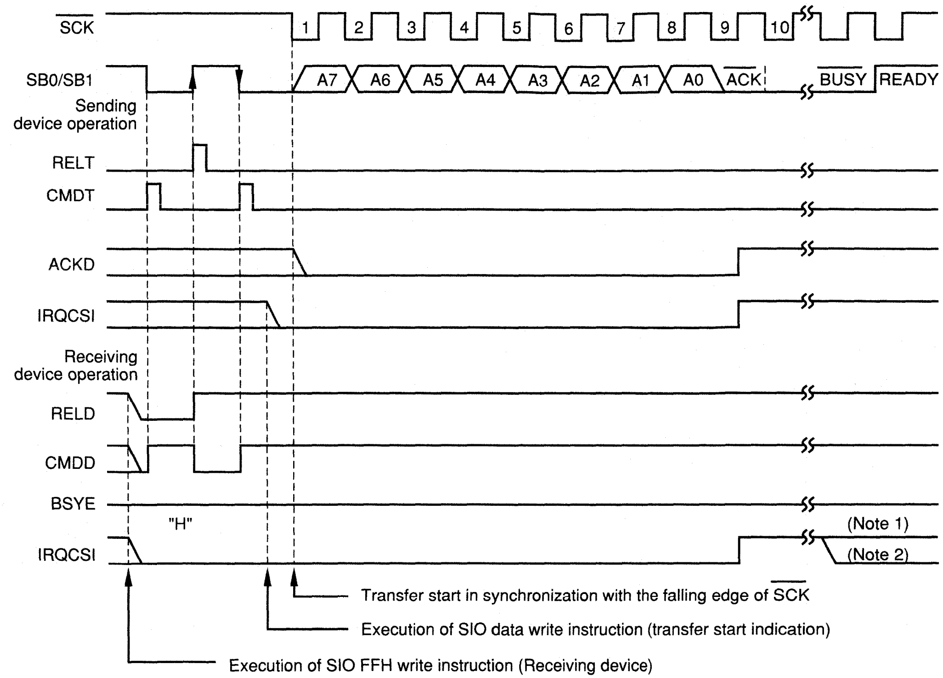

SCK
1
2
3
4
5
6
7
8
9
10
SB0/SB1
A7
A6
A5
A4
A3
A2
A1
A0
ACK
BUSY
READY
Sending device operation
RELT
CMDT
ACKD
IRQCSI
Receiving device operation
RELD
CMDD
BSYE
"H"
IRQCSI
(Note 1)
(Note 2)
Transfer start in synchronization with the falling edge of SCK
Execution of SIO data write instruction (transfer start indication)
Execution of SIO FFH write instruction (Receiving device)

**Fig. 5.6-13 SBI Mode Timing (Address Transfer)**

Note 1: Where SIO's data is read out by MOV instruction, then BSYE flag is cleared.
Note 2: Where SIO's data is exchanged (read/write) by XCH instruction.

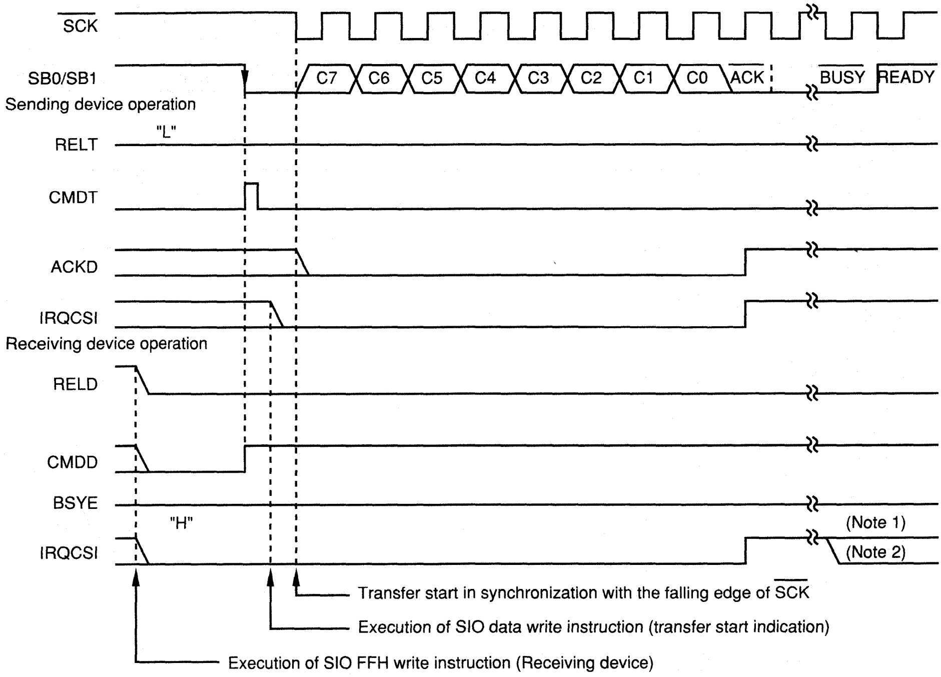

SCK
SB0/SB1
C7
C6
C5
C4
C3
C2
C1
C0
ACK
BUSY
READY
Sending device operation
"L"
RELT
CMDT
ACKD
IRQCSI
Receiving device operation
RELD
CMDD
BSYE
"H"
IRQCSI
(Note 1)
(Note 2)
Transfer start in synchronization with the falling edge of SCK
Execution of SIO data write instruction (transfer start indication)
Execution of SIO FFH write instruction (Receiving device)

**Fig. 5.6-14 SBI Mode Timing (Command Transfer)**

Note 1: Where SIO's data is read out by MOV instruction, then BSYE flag is cleared.
Note 2: Where SIO's data is exchanged (read/write) by XCH instruction.

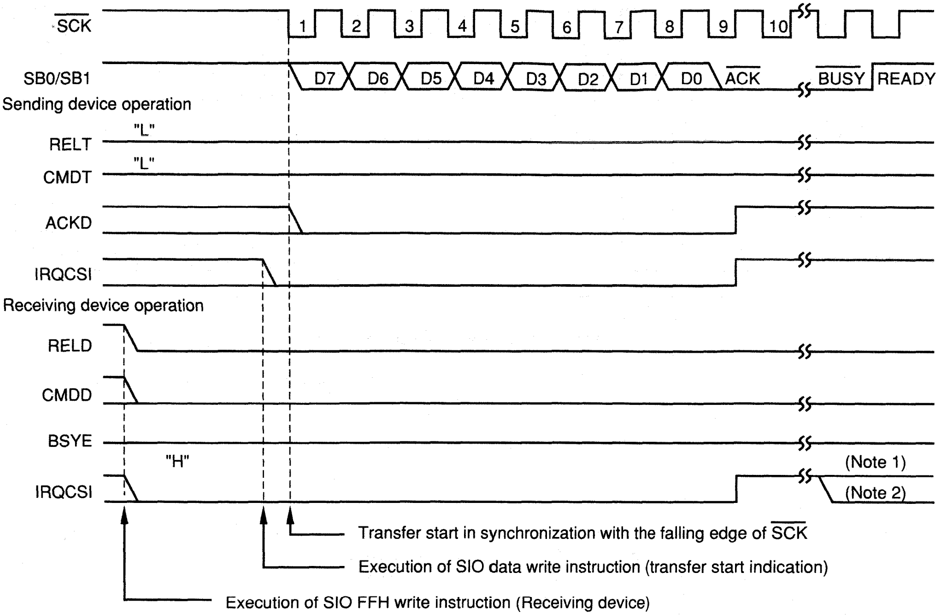

SCK
1
2
3
4
5
6
7
8
9
10
SB0/SB1
D7
D6
D5
D4
D3
D2
D1
D0
ACK
BUSY
READY
Sending device operation
RELT
"L"
CMDT
"L"
ACKD
IRQCSI
Receiving device operation
RELD
CMDD
BSYE
"H"
IRQCSI
(Note 1)
(Note 2)
Transfer start in synchronization with the falling edge of SCK
Execution of SIO data write instruction (transfer start indication)
Execution of SIO FFH write instruction (Receiving device)

**Fig. 5.6-15 SBI Mode Timing (Data Transfer)**

Note 1: Where SIO's data is read out by MOV instruction, then BSYE flag is cleared.
Note 2: Where SIO's data is exchanged (read/write) by XCH instruction.

(4) 2-line serial I/O mode

The 2-line serial I/O mode is used for communications using the SCK and SB0 or SB1 lines. The mode can handle any desired communication format by using software to control the SCK and SB0 or SB1 pin output levels.

Fig. 5.6-16 shows the 2-line serial I/O mode timing.

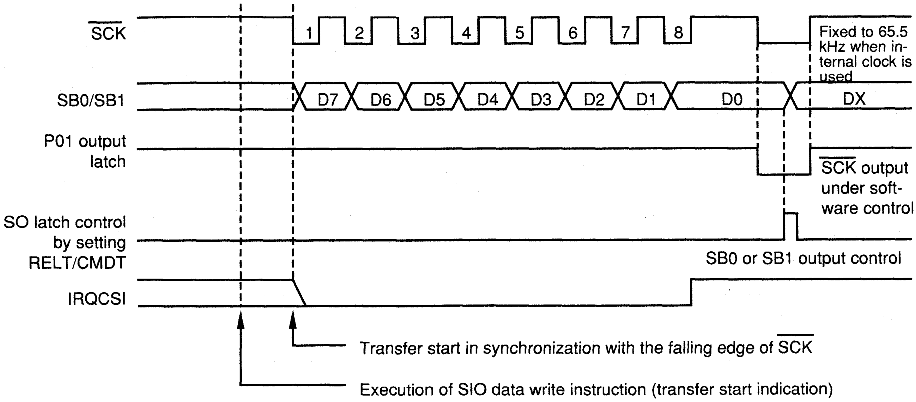
Serial transfer start indication is given by executing the instruction writing data into the shift register (SIO), MOV or XCH. Be sure to give start indication when CSIE = 1.

Shift register shifting is done in synchronization with the falling edge of the serial clock (SCK), and send data is held in the SO latch and output from the SB0/P02 or SB1/P03 pin starting at MSB. Receive data input from the SB0 or SB1 pin is latched in the shift register on the SCK rising edge.

The SB0 or SB1 pin specified for the serial data bus is used as an N-channel open drain input/output and needs pull-up. When data is received, the N-channel transistor must be turned off, thus FFH is previously written into SIO.

Since the SB0 or SB1 pin outputs the SO latch state, the SB0 or SB1 pin output state (high impedance or low level) can be controlled by setting the RELT and CMDT bits. However, do not change it during serial transfer.

In the output mode (internal clock mode), the SCK pin output state can also be controlled by using the P01 output latch, as described in 5.6.7.



$\overline{SCK}$
1
2
3
4
5
6
7
8
Fixed to 65.5 kHz when internal clock is used
SB0/SB1
D7
D6
D5
D4
D3
D2
D1
D0
DX
P01 output latch
$\overline{SCK}$ output under software control
SO latch control by setting RELT/CMDT
SB0 or SB1 output control
IRQCSI
Transfer start in synchronization with the falling edge of $\overline{SCK}$
Execution of SIO data write instruction (transfer start indication)

**Fig. 5.6-16 2-line Serial I/O Mode Timing**

5.6.6 Error detection

Since the data of the sending serial bus SB0 or SB1 is also read by the shift register SIO of the device which sends data in the SBI or 2-line serial I/O mode, a send error can be detected as described below:
(1) SIO data comparison before the start of a send and after the end of a send
If both the pieces of data do not match, it is determined that a send error occurred.
(2) Use of slave address register (SVA)
Send data is also set in SVA. After the completion of a send, the COI bit (coincidence signal from the address comparator) in the serial operating mode register is tested. If the bit is set to 1, it is determined that normal sending has been performed; if 0, it is determined that a send error has occurred.

5.6.7 $\overline{SCK}$ pin output handling
The SCK/P01 pin, which incorporates an output latch, can also produce static output by software control in addition to a normal serial clock.
The number of SCKs can be set as desired by using software to control the P01 output latch (the SO/SB0, and SI/SB1 pins are controlled by setting the RELT and CMDT bits).
The $\overline{SCK}$/P01 pin output control method is described below:
(1) The serial operation mode register (CSIM) is set ($\overline{SCK}$ pin: Output mode, serial operation: Enabled). $\overline{SCK}$ is set to 1 when serial transfer stops.
(2) The P01 output latch is controlled by using bit manipulation (operation) instructions.
Example:
To output one clock pulse of $\overline{SCK}$ by using software.

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

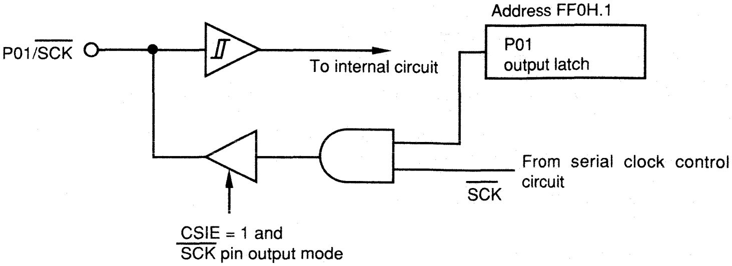

Address FF0H.1
P01/$\overline{\text{SCK}}$
To internal circuit
P01 output latch
From serial clock control circuit
$\overline{\text{SCK}}$
CSIE = 1 and
$\overline{\text{SCK}}$ pin output mode

**Fig. 5.6-17 $\overline{\text{SCK}}$/P01 Pin Configuration**

The P01 output latch is mapped in address FF0H bit 1. When the $\overline{\text{RESET}}$ signal is generated, the P01 output latch is set to 1.

Cautions:
1. The P01 output latch must be set to 1 during normal serial transfer.
2. Do not use "PORT0.1" to specify the P01 output latch address. Write directly the address (FF0H.1) in operand. At that time, set MBE to 0, or set MBE to 1 and MBS to 15.

```
CLR1  PORT0.1 ⎫
              ⎬ Do not use
SET1  PORT0.1 ⎭

CLR1  0FF0H.1 ⎫
              ⎬ Use
SET1  0FF0H.1 ⎭
```

5.6.8 Serial interface application

The serial interface for each mode is explained using examples of applications.

The normal serial interface communication sequence is as follows:

1) Set transfer mode. (Set data in CSIM.)

2) Write data into SIO and give transfer start indication. (MOV SIO, XA or XCH XA, SIO. At that time, automatic transfer start indication is given.)

3) After checking that the serial interrupt routine or interrupt request flag (IRQCSI) is set, read receive data, and start transfer.

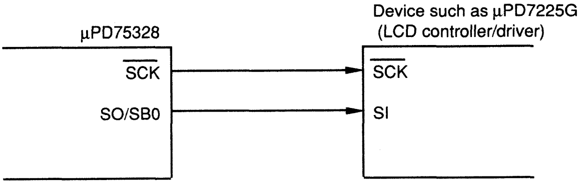
The SBI mode communication sequence is explained in detail in (3) below.

(1) 3-line serial I/O mode

(a) To transfer data starting at MBS using transfer clock of 262 kHz (at 4.19 MHz) (master operation).

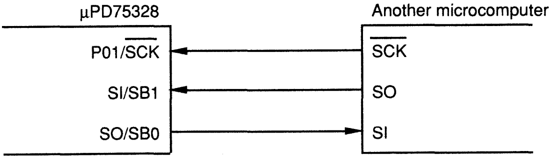
```
Example:  CLR1   MBE
          MOV    XA, #10000010B
          MOV    CSIM, XA       ; Transfer mode is set.
          MOV    XA, TDATA      ; TDATA is the transfer data storage address.
          MOV    SIO, XA        ; Transfer data is set.
                                ; Transfer is started.
```

Caution:
At the second time or after, transfer can be started by setting data in SIO (MOV SIO, XA or XCH XA, SIO).



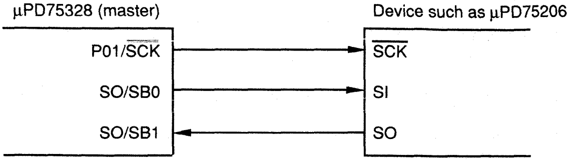
In this application example, the μPD75328 SI/SB1 pin can be used for input.

(b) To transfer data starting at the LSB using external clock (slave operation).
(In this example, the shift register read/write function to invert the LSB and MSB is used effectively.)



```
Example:  Main routine
          CLR1   MBE
          MOV    XA, #84H
          MOV    CSIM, XA       ; Serial operation stop, LSB/MSB inversion mode, external clock
          MOV    XA, TDATA
          MOV    SIO, XA        ; Transfer data is set.
                                ; Transfer is started.
          EI     IECSI
          EI

          Interrupt routine (MBE = 0)
          MOV    XA, TDATA
          XCH    XA, SIO        ; Receive data - send data, transfer start
          MOV    RDATA, XA      ; Receive data is saved.
          RETI
```

(c) To transfer data at a high speed using a transfer clock of 524 kHz (at 4.19 MHz).



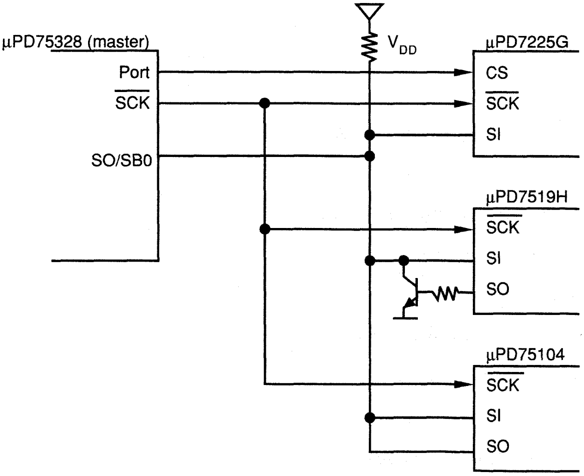
```
Example (master)
          CLR1     MBE
          MOV      XA, #10000011B
          MOV      CSIM, XA       ; Transfer mode is set.
          MOV      XA, TDATA
          MOV      SIO, XA        ; Transfer data is set, Transfer is started.
            :
            :
            :
LOOP:     SKTCLR  IRQCSI          ; IRQCSI is tested.
          BR       LOOP
          MOV      XA, SIO        ; Receive data is read.
```

(2) 2-line serial I/O mode

The serial bus is formed and a number of devices are connected.

Example:

The example system consists of the μPD75328 as the master and μPD75104, μPD7519H, and μPD7225G connected as slaves.



To form a serial bus as in this example, FFH is previously written into the shift register, a high level is output to the shift register, a high level is output to the SO pin, the output buffers are turned off, and the bus is released except when the SI and SO pins are connected for serial data output.

Since the μPD7519H SO pin cannot be placed in the high impedance state, a transistor is connected for open collector output, as shown in the diagram above. At the time of data input, the transistor is turned off by previously writing 00H into the shift register. The data output timing by the microcomputers is predetermined.

The serial clock is output by the master microcomputer μPD75328; all other slave microcomputers operate on external clocks.

(3) SBI mode

An application example of serial data communication in the SBI mode is given. In the example, the μPD75328 can operate as a master or a slave CPU.

The master can also be changed by using a command.

(a) Serial bus configuration

The serial bus configuration in the application example given here assumes that the μPD75328 is connected to bus lines as one device in the serial bus.

The following two μPD75328 pins are used: Serial data bus SB0 (P02/SO) and serial clock SCK (P01).

Fig. 5.6-18 shows an example of a serial bus configuration.

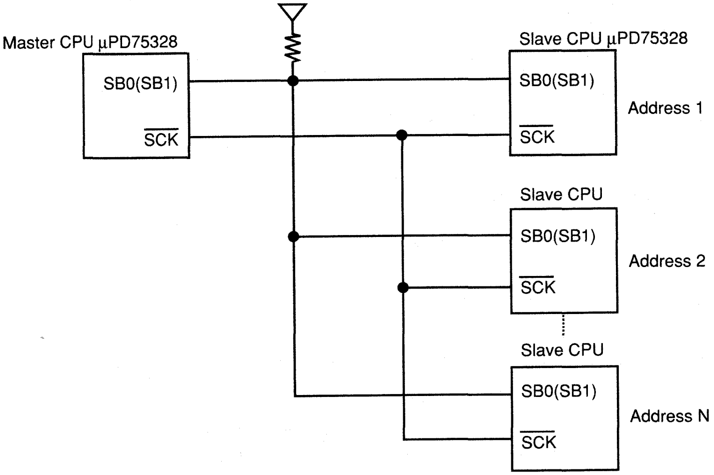


**Fig. 5.6-18 Serial Bus Configuration Example**

(b) Command explanation

Command types

The application example uses the following commands:

1) READ:
Data is transferred from slave to master.

2) WRITE:
Data is transferred from master to slave.

3) END:
WRITE command completion is reported to slave.

4) STOP:
WRITE command stop is reported to slave.

5) STATUS:
The slave state is read.

6) RESET:
The current slave being selected is made unselected.

7) CHGMST:
The master authorization is transferred to the slave.

Communication sequence

The communication sequence between the master and slave is as follows:

1) Communication is started by the master, which sends the address of the slave to communicate with and selects the slave (chip select).

The slave which receives the address returns $\overline{\text{ACK}}$ and communicates with the master. (The slave is placed in selected state.)

2) Command and data are transferred between the slave selected in 1 and the master.

Since command and data are transferred point-to-point (between the master and specific slave), other slaves must be deselected.

3) Communication terminates when the slave is deselected in either of the following cases:

— When the master sends the RESET command, the selected slave is deselected.

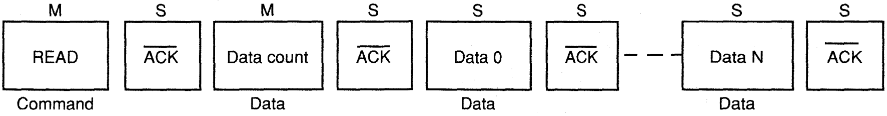
— If the master is changed by using the CHGMST command, the device changed from master to slave is deselected.

Command format

The command transfer formats are shown below:

1) READ command

The READ command reads data from a given slave. The read data count ranges from one to 256 bytes. The master specifies the data count in a parameter. If 00H is specified for the data count, 256-byte data transfer is assumed to be specified.



M: Output by master
S: Output by slave

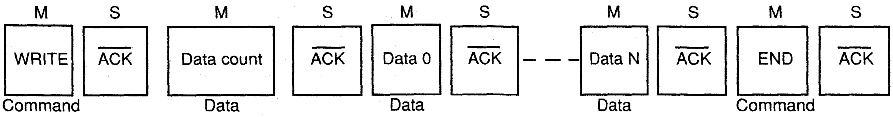
**Fig. 5.6-19 READ Command Transfer Format**

After receiving the data count, the slave returns $\overline{\text{ACK}}$ if the number of data bytes that can be sent is greater than the data count. If it is less than the data count, the slave does not return $\overline{\text{ACK}}$, resulting in an error.

During data transfer, the slave compares the SIO contents before and after data transfer to check that data has been output to the bus normally. If the SIO contents before and after transfer do not match, the slave does not return $\overline{\text{ACK}}$, resulting in an error.

2) WRITE, END, and STOP commands

The WRITE, END, and STOP commands are used to write data into a particulary slave. The write data count ranges from 1 to 256 bytes. The master specifies the data count in a parameter. If 00H is specified for the data count, 256-byte data transfer is assumed to be specified.



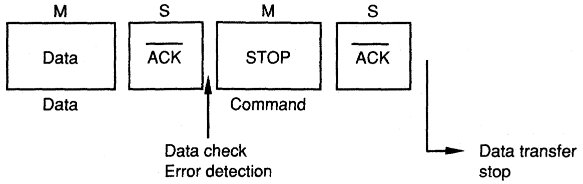
M: Output by master
S: Output by slave

**Fig. 5.6-20 WRITE, END Command Transfer Format**

After receiving the data count, the slave returns $\overline{\text{ACK}}$ if the receive data store area is larger than the data count. If it is less than the data count, it does not return $\overline{\text{ACK}}$, resulting in an error.

At the termination of all data transfer, the master sends the END command to the slave. It signals that all data has been transferred normally.

The slave also receives an END command before it completes all data reception. In this case, the data which has been received immediately before the END command is received becomes valid. During data sending, the master compares the SIO contents before and after data sending to check that data has been output to the bus normally. If the SO contents before and after sending do not match, the master sends the STOP command and stops data transfer.

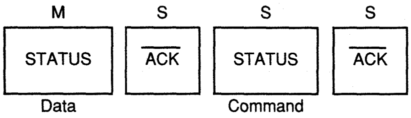


M: Output by master
S: Output by slave

**Fig. 5.6-21 STOP Command Transfer Format**

When receiving the STOP command, the slave invalidates the 1-byte data received immediately before receiving the STOP command.

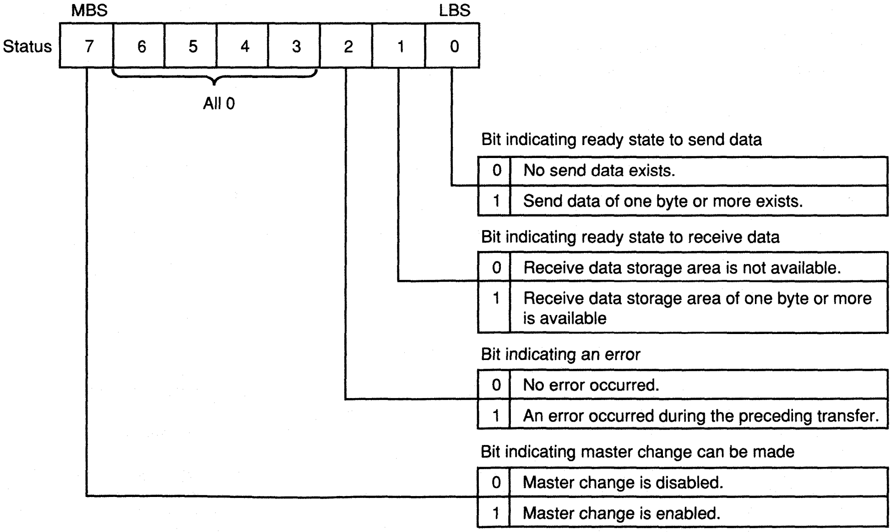
3) STATUS command
The STATUS Command Transfer Format



M: Output by master
S: Output by slave

**Fig. 5.6-22 STATUS Command Status Format**

Fig. 5.6-23 shows the format of the status returned by the slave.



Bit indicating ready state to send data

| | |
|---|---|
| 0 | No send data exists. |
| 1 | Send data of one byte or more exists. |

Bit indicating ready state to receive data

| | |
|---|---|
| 0 | Receive data storage area is not available. |
| 1 | Receive data storage area of one byte or more is available |

Bit indicating an error

| | |
|---|---|
| 0 | No error occurred. |
| 1 | An error occurred during the preceding transfer. |

Bit indicating master change can be made

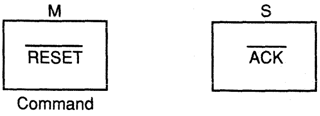
| | |
|---|---|
| 0 | Master change is disabled. |
| 1 | Master change is enabled. |

**Fig. 5.6-23 STATUS Command Status Format**

To send status data, the slave compares the contents before and after sending status data. If they do not match, the slave does not return $\overline{\text{ACK}}$, resulting in an error.

4) $\overline{\text{RESET}}$ command

The $\overline{\text{RESET}}$ command is used to cause the currently selected slave to be selected. When the $\overline{\text{RESET}}$ command is issued, all slaves can be deselected.

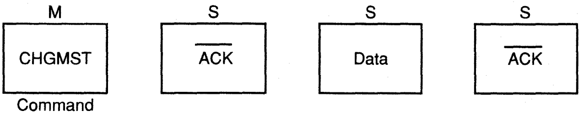


M: Output by  master
S: Output ba slave

**Fig. 5.6-24 $\overline{\text{RESET}}$ Command Transfer Format**

5) CHGMST command

The CHGMST command transfers the master authorization to the currently selected slave.



M: Output by master
S: Output by slave

**Fig. 5.6-25 CHGMST Command Transfer Format**

When receiving the CHGMST command, the slave decides whether or not it can receive the master authorization, and returns either of the following data to the master:

— 0FFH: Master change is enabled.
— 00H  : Master change is disabled.

When transferring data, the slave compares the SIO contents before and after data transfer. If they do not match, the slave does not return $\overline{\text{ACK}}$, resulting in an error.

If no error occurs, the master serves as a slave after 0FFH data sending is complete. If no error occurs, the slave serves as the master after 0FFH data sending is completed.

Error occurrence

When a communication error occurs, the master and slave operate as explained below:

The slave informs the master of error occurrence by returning no $\overline{\text{ACK}}$. When an error occurs, the status bit (bit 2) indicating error occurence is set to 1 and all command processing being performed is cancelled.

After completing the sending or receiving of one byte, the master checks whether or not $\overline{\text{ACK}}$ is returned from slave. If $\overline{\text{ACK}}$ is not returned from slave within a given period after sending or receiving is completed, the master decides that an error has occurred and outputs a dummy $\overline{\text{ACK}}$ signal.

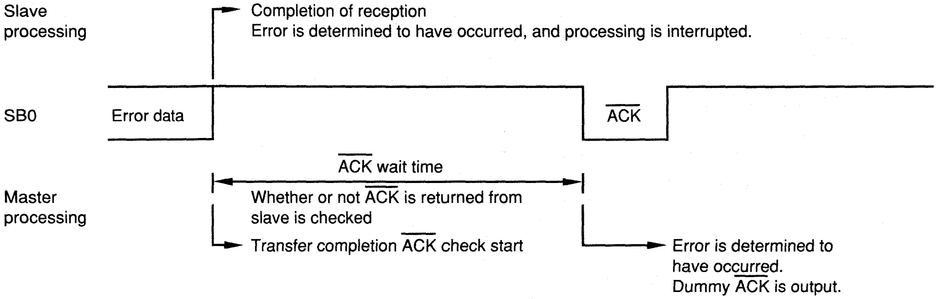

Slave processing
Completion of reception
Error is determined to have occurred, and processing is interrupted.
SB0
Error data
ACK
ACK wait time
Master processing
Whether or not ACK is returned from slave is checked
Transfer completion ACK check start
Error is determined to have occurred.
Dummy ACK is output.

**Fig. 5.6-26 Master and Slave Operation when Error Occured**

The following errors are possible:

— Errors that may occur in the slave

1) Command transfer format is erroneous.
2) Undefined command is received.
3) The number of data bytes to be transferred (data count) is insufficient during READ command execution,
4) Data storage area is insufficient during WRITE command execution.
5) When READ, STATUS, or CHGMST command data is sent, data changes.

When any error in 1) or 5) occurs, no $\overline{\text{ACK}}$ is returned.

— Errors that may occur in the master

When WRITE command data is sent, if data changes, STOP command is sent to the slave.

### 5.7 LCD Controller/Driver

5.7.1 LCD controller/driver configuration

The µPD75328 incorporates a display controller which generates segment and common signals according to the display data memory contents and incorporates segment and common drivers which can drive the panel directly.

Fig. 5.7-1 shows the LCD controller/driver configuration.

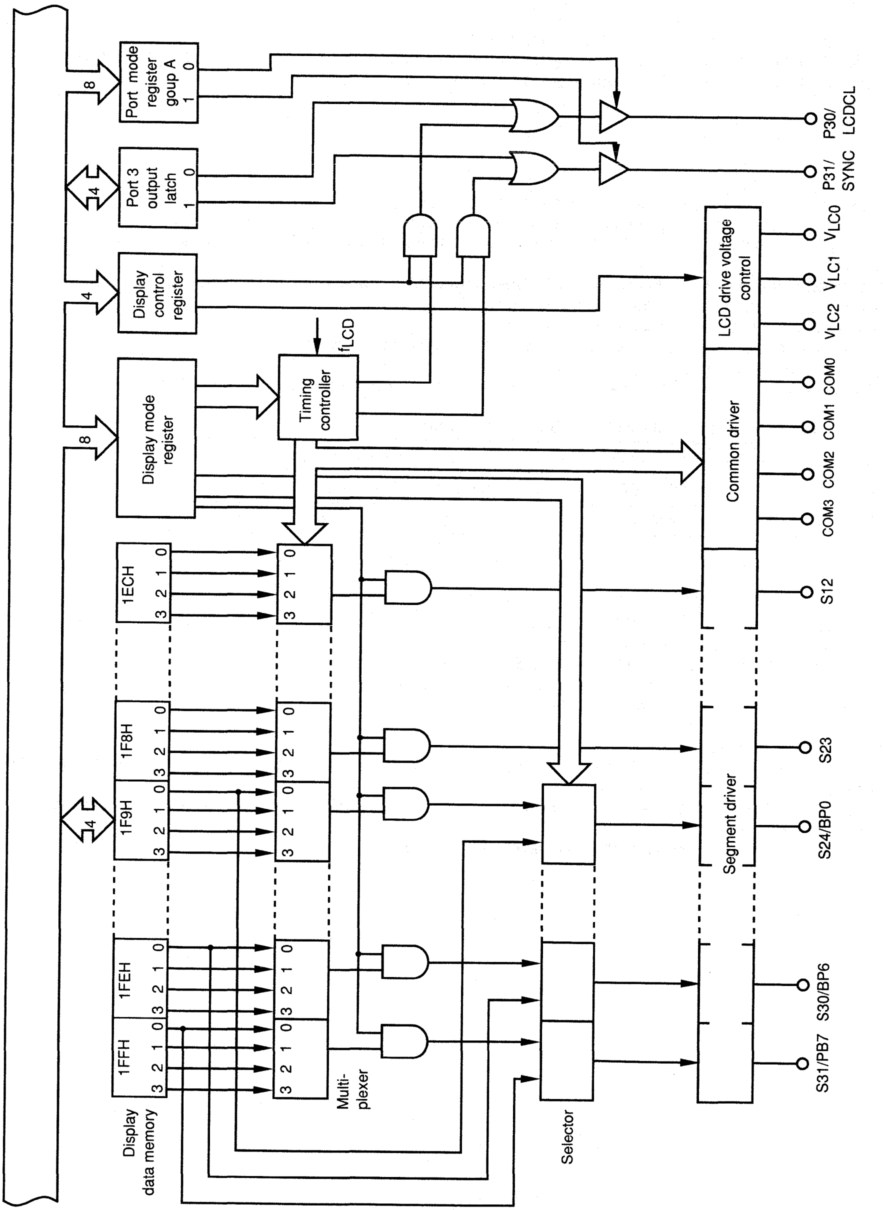


**Fig. 5.7-1 LCD Controller/Driver Block Diagram**

5.7.2 LCD controller/driver functions

The µPD75328 LCD controller/driver functions are as follows:

(a) Display data memory is read automatically by DMA operation and segment and common signals are generated.

(b) Display mode can be selected from among the following five.

1) Static
2) 1/2 duty (1/2 bias)
3) 1/3 duty (1/2 bias)
4) 1/3 duty (1/3 bias)
5) 1/4 duty (1/3 bias)

(c) A frame frequency can be selected from among four in each display mode.

(d) A maximum of 20 segment signal output pins (S12-S31) and four common signal output pins (COM0 - COM3).

(e) The segment signal output pins (S24-S27 and S28-S31) can be changed to the output ports (BP0-BP3 and BP4-BP7) in 4-pin units.

(f) Split-resistor can be incorporated to supply LCD drive power. (Mask option)

— Various bias laws and LCD drive voltage can be handled.
— When display is off, current flow to the split resistor is cut.

(g) Display data memory not used for display can be used for normal data memory.

(h) It can also operate by using the subsystem clock.

Table 5.7-1 lists the maximum number of picture elements that can be displayed in each display mode.

**Table 5.7-1 Maximum Number of Displayed Picture Elements**

| Bias law | Time division | Used COMMON signals | Maximum number of picture elements |
|---|---|---|---|
| — | Static | COM0 (COM1, 2, 3) | 20 (segment 20 x common 1) (Note 1) |
| 1/2 | 2 | COM0, 1 | 40 (segment 20 x common 2) (Note 2) |
|  | 3 | COM0, 1, 2 | 60 (segment 20 x common 3) (Note 3) |
| 1/3 | 3 |  |  |
|  | 4 | COM0, 1, 2, 3 | 80 (segment 20 x common 4) (Note 4) |

Note 1: 2 digits (eight segment signal/digit) on LCD panel (8. display).
Note 2: 5 digits (four segment signal/digit) on LCD panel (8. display).
Note 3: 6 digits (three segment signal/digit) on LCD panel (8. display).
Note 4: 10 digits (two segment signal/digit) on LCD panel (8. display).

5.7.3 Display mode register

The display mode register (LCDM) consists of eight bits to specify the display mode, LCD clock, frame frequency, segment or bit port output, and display output on/off control.

LCDM is set by using 8-bit memory operation instruction. Only bit 3 (LCDM3) can be set and cleared by using bit operation (manipulation) instructions.

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

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| F8CH | LCDM7 | LCDM6 | LCDM5 | LCDM4 | LCDM3 | LCDM2 | LCDM1 | LCDM0 | LCDM |

Display Mode Selection

| LCDM3 | LCDM2 | LCDM1 | LCDM0 | Time division value | Bias method |
|---|---|---|---|---|---|
| 0 | X | X | X | Display off (Note) | |
| 1 | 0 | 0 | 0 | 4 | 1/3 |
| 1 | 0 | 0 | 1 | 3 | 1/3 |
| 1 | 0 | 1 | 0 | 2 | 1/2 |
| 1 | 0 | 1 | 1 | 3 | 1/2 |
| 1 | 1 | 0 | 0 | Static | |
| Other than the above | | | | Undefined | |

Note: All segment signals are unselected.

LCD Clock Selection

| LCDM5 | LCDM4 | LCDMCL |
|---|---|---|
| 0 | 0 | $f_W/2^9$ (64 Hz) |
| 0 | 1 | $f_W/2^8$ (128 Hz) |
| 1 | 0 | $f_W/2^7$ (256 Hz) |
| 1 | 1 | $f_W/2^6$ (512 Hz) |

Caution: LCDCL is supplied only when the watch timer operates. To use the LCD controller, bit 2 of watch mode register WM should be set to 1.

Segment and Bit Port Output Change Specification

| LCDM7 | LCDM6 | S24/S27 | S28/S31 | Number of segment output pins | Number of bit port output pins |
|---|---|---|---|---|---|
| 0 | 0 | Segment output | Segment output | 20 | 0 |
| 0 | 1 | Segment output | Bit port output | 16 | 4 |
| 1 | 0 | Bit port output | Segment output | 16 | 4 |
| 1 | 1 | Bit port output | Bit port output | 12 | 8 |

Frame Frequency (Hz)

| LCDCL / Display duty cycle | $f_W/2^9$ (64 Hz) | $f_W/2^8$ (128 Hz) | $f_W/2^7$ (256 Hz) | $f_W/2^6$ (512 Hz) |
|---|---|---|---|---|
| Static | 64 | 128 | 256 | 512 |
| 1/2 | 32 | 64 | 128 | 256 |
| 1/3 | 21 | 43 | 85 | 171 |
| 1/4 | 16 | 32 | 64 | 128 |

When $f_W$ = 32.768 kHz    $f_W$: Input clock to watch timer ($f_X$/128 or $f_{XT}$)

**Fig. 5.7-2 Display Mode Register Format**

5.7.4 Display control register

The display control register controls LCD drive as follows:

— Enables or disables common and segment output.

— Cuts current flow to the split resistor for the LCD power supply.

— Enables or disables output of the synchronizing clock (LCDCL) and synchronizing signal (SYNC) to the controller/driver for external segment signal extension.

LCDCL is set by using a 4-bit memory operation instruction.

When the $\overline{\text{RESET}}$ signal is generated, the display control register is cleared.

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| F8EH | 0 | LCDC2 | 0 | LCDC0 | LCDC |

| LCDC0 | 0 | 1 | |
|---|---|---|---|
| LCDM3 | X | 0 | 1 |
| COM0-3 | Low level is output (display off). | Common signal corresponding to the display mode is output. | Common signal corresponding to the display mode is output. |
| S12-S23<br>S24-S31 segment specification pins | Low level is output (display off). | Segment signal corresponding to the display mode is output (no-selection level output, display off). | Segment signal corresponding to the display mode is output (display on). |
| S24-S31 bit port specification pins | Bit 0 for the corresponding display data memory is output (bit port function). | Bit 0 for the corresponding display data memory is output (bit port function). | Bit 0 for the corresponding display data memory is output (bit port function). |
| Power supply to split resistor (BIAS pin output) | Off (high impedance) | On (high level) | On (high level) |

Note 1: ( ): When internal split register for LCD drive power supply is not contained.

LCDCL and SYNC Signal Output Enable/Disable Specification Bit

| | | |
|---|---|---|
| LCDCL2 | 0 | LCDCL and SYNC signal output is disabled. |
| | 1 | LCDCL and SYNC signal output is enabled. |

Note: LCDCL, SYNC signal output are provided for future system extension. Disable signal output at present.

**Fig. 5.7-3 Display Control Register Format**

5.7.5 Display data memory

The display data memory is mapped in 1ECH-1FFH.

The display data memory is an area read by the LCD controller/driver, which performs DMA operation independently of CPU operation. The LCD controller controls the segment signals according to data in the display data memory. When S24-S31 are used for bit ports, bit 0 of the data written into display data memory addresses 1F8H-1FFH is output from each bit port output pin.

The area not used for LCD display or ports can be used for normal data memory.

The display data memory is handled in 1- or 4-bit units. It cannot be handled in 8-bit units.

Fig. 5.7-5 shows the relationship between the display data memory bits and segment output/bit port output.

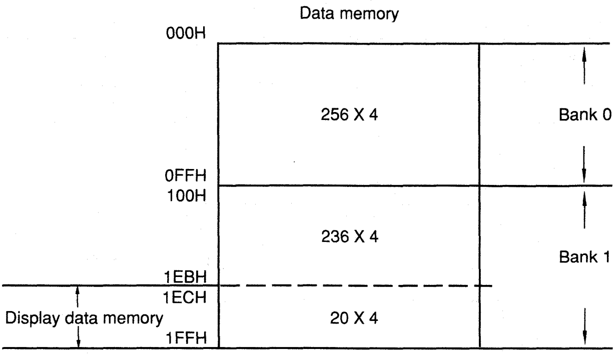


**Fig. 5.7-4 Data Memory Map**

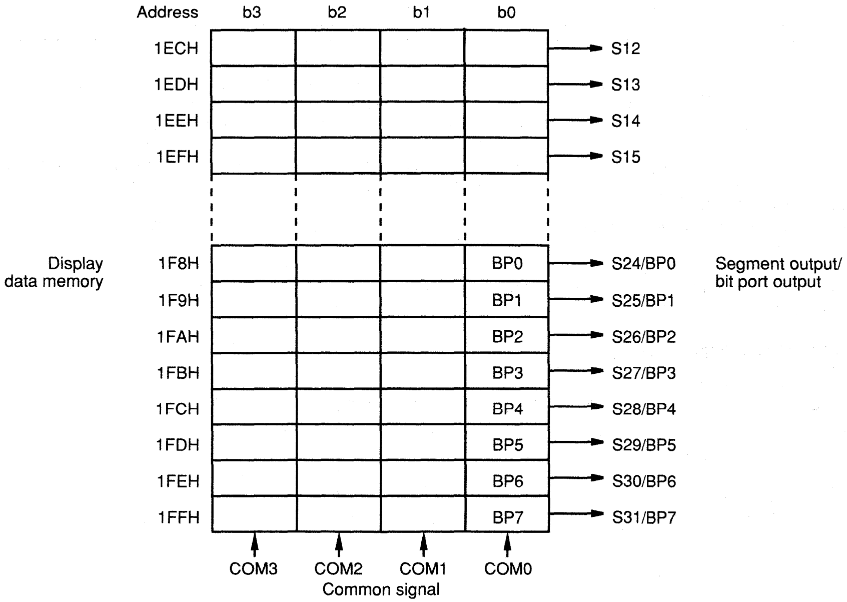


**Fig. 5.7-5 Relationship between Display Data Memory and Common Segment**

### 5.7.6 Common and segment signals

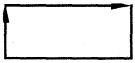
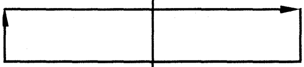
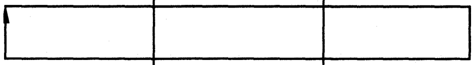
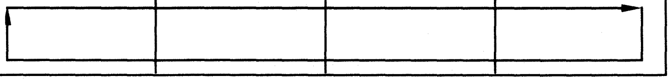
Each picture element of the LCD panel goes on when the potential difference between its corresponding common signal and segment signal reaches or exceeds a given voltage (LCD drive voltage VLCD). It goes off when the potential difference falls below VLCD or reaches 0V.

Since deterioration occurs when DC potential is maintained as the common and segment signals, the LCD panel is driven by AC.

(1) Common signals

The common signals become the selection timing in the order listed in Table 5.7-2 according to the set time division number, and repetitive operation is performed by using them as one period. In the static mode, COM0 to COM3 output the same signal. In division by 2, neither COM2 nor COM3 pin should be connected. In division by 3, do not connect the COM3 pin.

**Table 5.7-2 COM Signals**

| COM signal<br>Time division number | COM0 | COM1 | COM2 | COM3 |
|---|---|---|---|---|
| Static |  | | | |
| 2 |  | | No connection required | No connection required |
| 3 |  | | | No connection required |
| 4 |  | | | |

(2) Segment signals

There are 20 segment signals corresponding to 20 locations of the display data area (1ECH-1FFH) of the data memory. Bits 0 to 3 of each location are automatically read in synchronization with the selection timing for COM0 to COM3 respectively. If the bit is set to 1, it produces the segment selection voltage; if 0, the bit produces the no-selection voltage output from the corresponding segment pin (S12-S31).

Thus, check the display patterns formed according to the combinations of the LCD panel front electrode (corresponding to segment signal) and rear electrode (corresponding to common signal). Then, on a one-to-one basis, write into the display area the bit data which corresponds to the pattern to be displayed.

Display data area bits 1 to 3 in the static mode, bits 2 and 3 in the division by 2 mode, and bit 3 in the division by 3 mode are not accessed and can be used for purposes other than display.

(3) Common and segment signal output waveforms

Tables 5.7-3 to 5.7-5 list voltages output to the common and segment signals. +VLCD/–VLCD on voltage is applied only when both signals become selection voltages; otherwise, the off voltage is applied.

**Table 5.7-3 LCD Drive Voltage (Static)**

| Segment signal Sn<br>Common signal COM0 | Selection | No-selection |
|---|---|---|
| | $V_{LC0}/V_{SS}$ | $V_{SS}/V_{LC0}$ |
| $V_{SS}/V_{LC0}$ | $+V_{LCD}/-V_{LCD}$ | 0V/0V |

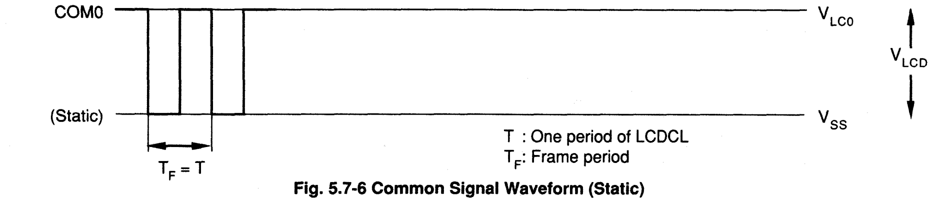
**Table 5.7-4 LCD Drive Voltage (1/2 Bias Law)**

| Segment signal Sn<br>Common signal COMm | | Selection | No-selection |
|---|---|---|---|
| | | $V_{LC0}/V_{SS}$ | $V_{SS}/V_{LC0}$ |
| Selection | $V_{SS}/V_{LC0}$ | $\boxed{+V_{LCD}/-V_{LCD}}$ | 0V/0V |
| No-selection | $V_{LC1} = V_{LC2}$ | $+1/2\ V_{LCD}/-1/2\ V_{LCD}$ | $-1/2\ V_{LCD}/+1/2\ V_{LCD}$ |

**Table 5.7-5 LCD Drive Voltage (1/3 Bias Law)**

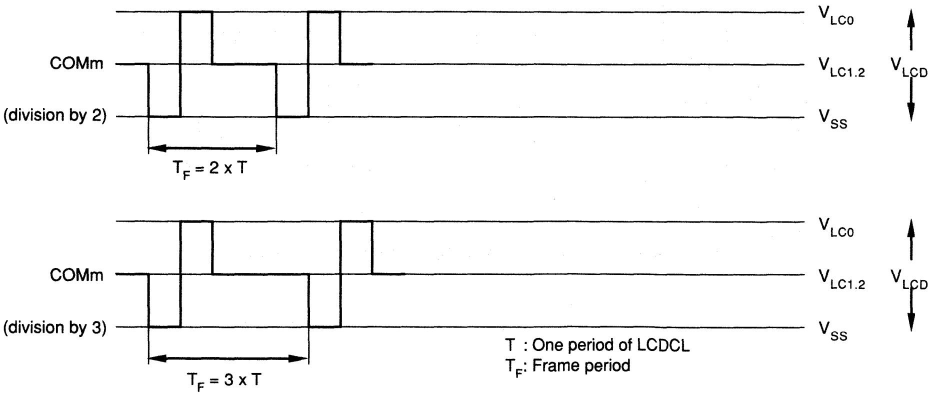
| Segment signal Sn<br>Common signal COMm | | Selection | No-selection |
|---|---|---|---|
| | | $V_{LC0}/V_{SS}$ | $V_{LC2}/V_{LC1}$ |
| Selection | $V_{SS}/V_{LC0}$ | $\boxed{+V_{LCD}/-V_{LCD}}$ | $+1/3\ V_{LCD}/-1/3\ V_{LCD}$ |
| No-selection | $V_{LC1}/V_{LC2}$ | $+1/3\ V_{LCD}/-1/3\ V_{LCD}$ | $+1/3\ V_{LCD}/-1/3\ V_{LCD}$ |

Figs. 5.7-6 to 5.7-8 shows the common signal waveforms. Fig. 5.7-9 shows the common and segment signal voltages and phases.



T : One period of LCDCL
$T_F$: Frame period

**Fig. 5.7-6 Common Signal Waveform (Static)**



T : One period of LCDCL
$T_F$: Frame period

**Fig. 5.7-7 Common Signal Waveform (1/2 Bias Law)**

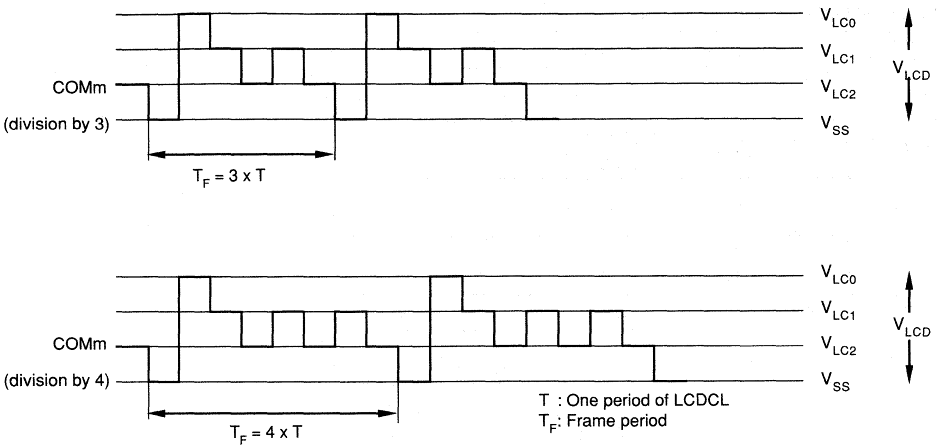


**Fig. 5.7-8 Common Signal Waveform (1/3 Bias Law)**

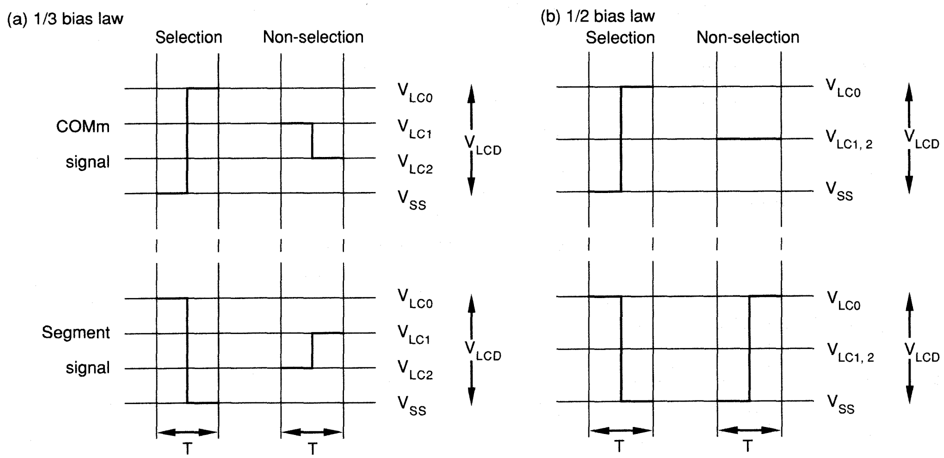


**Fig. 5.7-9 Common and Segment Signal Voltages and Phases**

(c) Static display mode

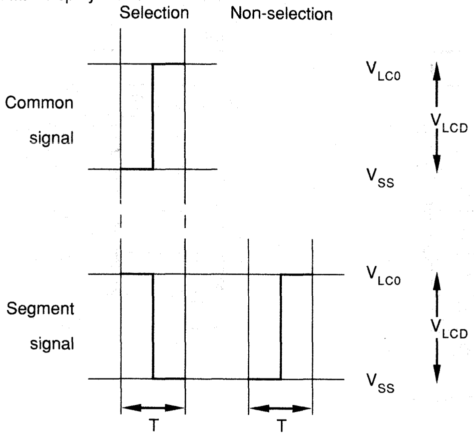


**Fig. 5.7-9 Common and Segment Signal Voltages and Phases (cont'd)**

5.7.7 Supply of LCD drive power $V_{LC0}$, $V_{LC1}$, and $V_{LC2}$

In the μPD75328, a split resistor can be incorporated in the $V_{LC0}$-$V_{LC2}$ pins for the LCD drive power supply. According to the bias law, the LCD drive power can be supplied without an external split resistor. The μPD75328 also includes the BIAS pin to deal with various LCD drive voltages. The BIAS and $V_{LC0}$ pins are connected externally.

Table 5.7-6 lists proper LCD drive power supply values based on the static, 1/2, and 1/3 bias laws.

**Table 5.7-6 LCD Drive Power Supply Values**

| Bias law / LCD drive power | No bias (static mode) | 1/2 | 1/3 |
|---|---|---|---|
| $V_{LC0}$ | $V_{LCD}$ | $V_{LCD}$ | $V_{LCD}$ |
| $V_{LC1}$ | 2/3 $V_{LCD}$ | 1/2 $V_{LCD}$ (Note 1) | 2/3 $V_{LCD}$ |
| $V_{LC2}$ | 1/3 $V_{LCD}$ | | 1/3 $V_{LCD}$ |
| $V_{LC3}$ | 0V | 0V | 0V |

Note 1: When 1/2 bias is used, the VLC1 and VLC2 pins must be connected externally.

Note 2: When the BIAS and $V_{LC0}$ pins are not connected, $V_{LCD}$ = 3/5 $V_{DD}$ (when split resistor is incorporated). When the BIAS and $V_{LC0}$ pins are connected, $V_{LCD}$ = $V_{DD}$.

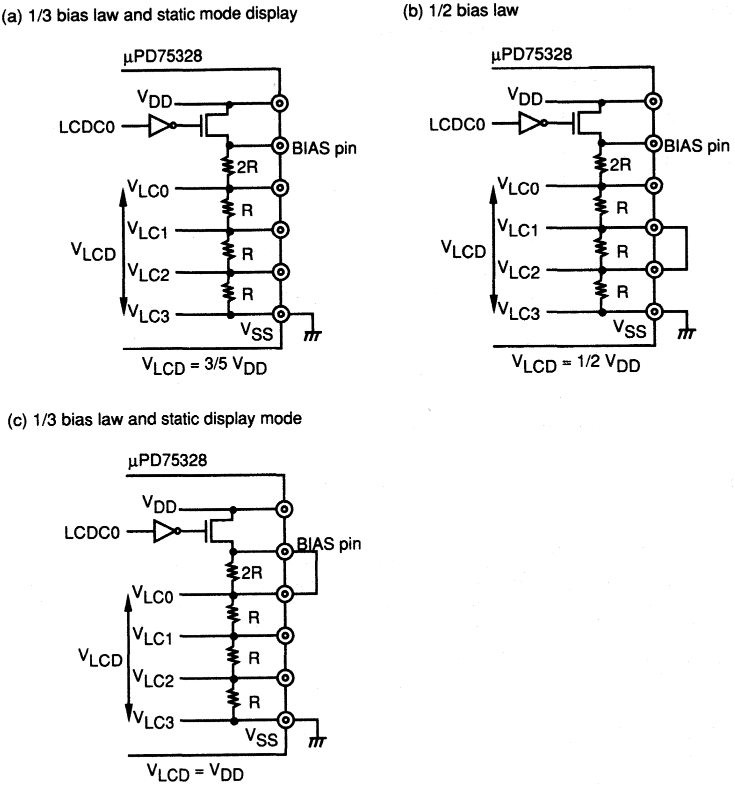
In Fig. 5.7-10, (a) to (c) show LCD drive power supply examples according to Table 5.7-6. Fig. 5.7-11 (d) shows an example in which external resistor is connected to the LCD drive voltage pins ($V_{LC0}$–$V_{LC2}$) and BIAS pin, and internal split resistor is fine-adjusted.

Current flow through the split resistor can also be cut by clearing display control register bit 0 (LCDC0).

This LCD power on/off control is also useful to prevent DC voltage from being applied to LCD when the LCD clock is stopped by execution of a STOP instruction and when the watch timer operates using the main system clock.

That is, display control register bit 0 (LCDC0) is cleared and all LCD drive power sources are placed in the same potential VSS immediatly before the STOP instruction is executed, thereby suppressing the potential difference between the LCD electrodes even if the LCD clock is stopped.

When the watch timer operates using the subsystem clock, the LCD display can still be used.



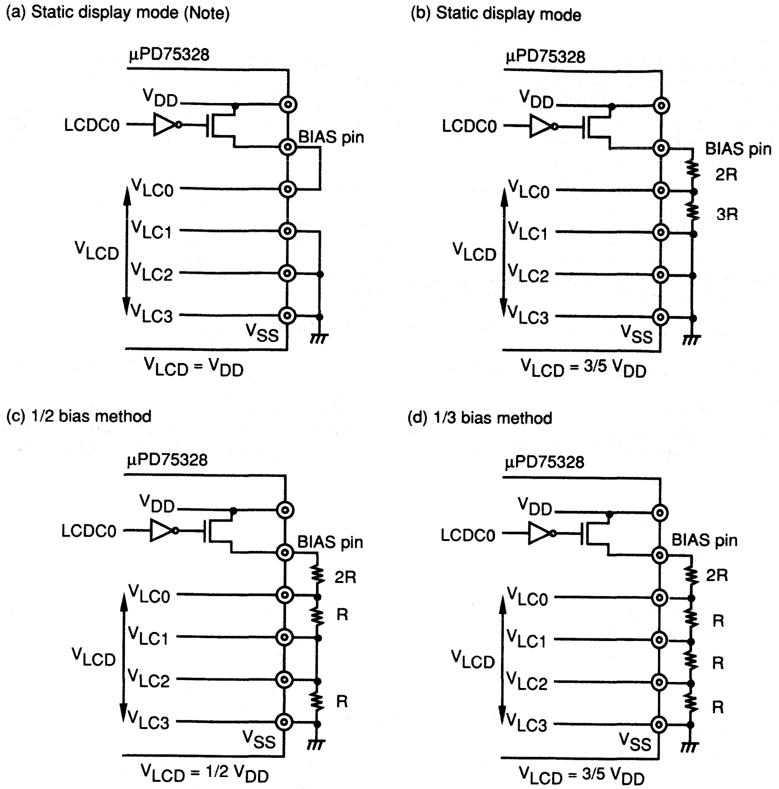
(a) 1/3 bias law and static mode display

(b) 1/2 bias law

(c) 1/3 bias law and static display mode

In example (a), $V_{DD}$ = 5V and LCD drive voltage $V_{LCD}$ = 3V.
In example (b), $V_{DD}$ = 5V and $V_{LCD}$ = 2,5V.
In example (c), $V_{DD}$ = 5V and $V_{LCD}$ = 5V.

**Fig. 5.7-10 LCD Drive Power Connection Examples (when split resistor is incorporated)**



In example (a), $V_{DD}$ = 5V and LCD drive voltage $V_{LCD}$ = 5V.
In example (b) and (d), $V_{DD}$ = 5V and LCD drive voltage $V_{LCD}$ = 3V.
In example (c), $V_{DD}$ = 5V and $V_{LCD}$ = 2.5V.

Note: Be sure to set LCDC0 to 1 (also during the standby mode)

**Fig. 5.7-11 LCD Drive Power Connection Examples (when split resistor is corporated externally)**

5.7.8 Display mode

(1) Static display example

Fig. 5.7-14 shows connection of a static 3-digit LCD panel having the display pattern shown in Fig. 5.7-13, the μPD75328 segment signals (S12-S31), and the common signal (COM0). In this example, 123 is displayed. The contents of the display data memory (addresses 1ECH-1FFH) correspond to the display pattern.

Here, 3 ( 3 ) at the third digit position is taken as an example. It is necessary to output selection and no-selection voltages as shown in Fig. 5.7-12 to the S12-S18 pins at the COM0 common signal timing according to the display pattern shown in Fig. 5.7-13.

| Segment / Common | S12 | S13 | S14 | S15 | S16 | S17 | S18 |
|---|---|---|---|---|---|---|---|
| COM0 | Selection | Selection | Selection | No-selection | Selection | No-selection | Selection |

**Fig. 5.7-12 Selection/Non-selection Voltages of S12-S18 Pins (Static Display Examples)**

This shows that the bit 0 string of display data memory addresses 1ECH-1EFH corresponding to S12-S18 needs to be set to 1110101.

Fig. 5.7-15 shows the S14, S15, and COM0 LCD drive waveforms. This shows that alternating current square wave of $+V_{LCD}$/ $-V_{LCD}$ that is LCD on level is generated when S14 becomes the selection voltage at the selection timing of COM0. Since the same waveform as COM0 is output to COM1 to COM3, the drive capability can be increased by connecting COM0, COM1, COM2, and COM3.

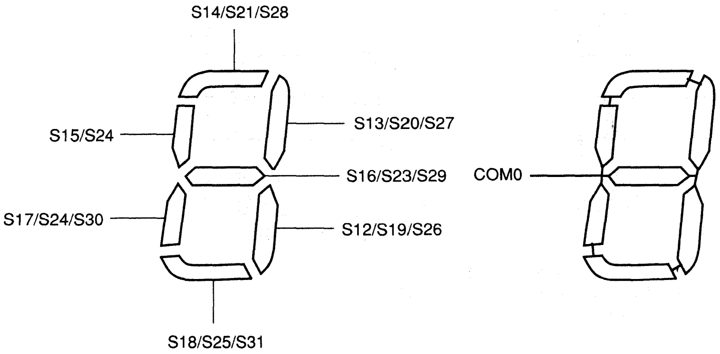
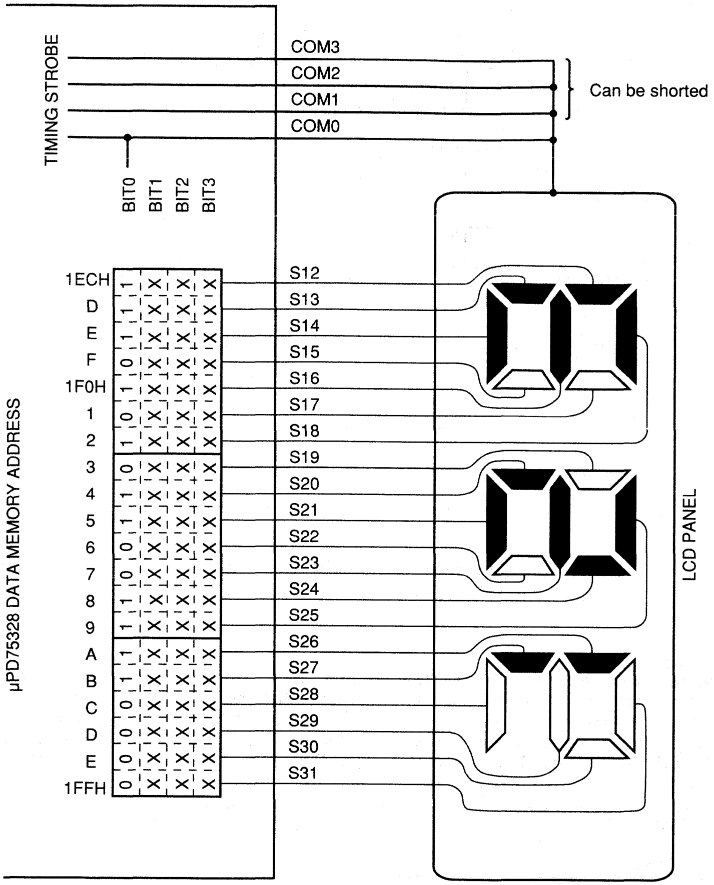


**Fig. 5.7-13 Static Mode LCD Display Pattern and Electrode Connection**



| | BIT0 | BIT1 | BIT2 | BIT3 |
|---|---|---|---|---|
| 1ECH | 1 | X | X | X |
| D | 1 | X | X | X |
| E | 1 | X | X | X |
| F | 0 | X | X | X |
| 1F0H | 1 | X | X | X |
| 1 | 0 | X | X | X |
| 2 | 1 | X | X | X |
| 3 | 0 | X | X | X |
| 4 | 1 | X | X | X |
| 5 | 1 | X | X | X |
| 6 | 0 | X | X | X |
| 7 | 0 | X | X | X |
| 8 | 1 | X | X | X |
| 9 | 1 | X | X | X |
| A | 1 | X | X | X |
| B | 1 | X | X | X |
| C | 0 | X | X | X |
| D | 0 | X | X | X |
| E | 0 | X | X | X |
| 1FFH | 0 | X | X | X |

**Fig. 5.7-14 Static LCD panel Connection Example**

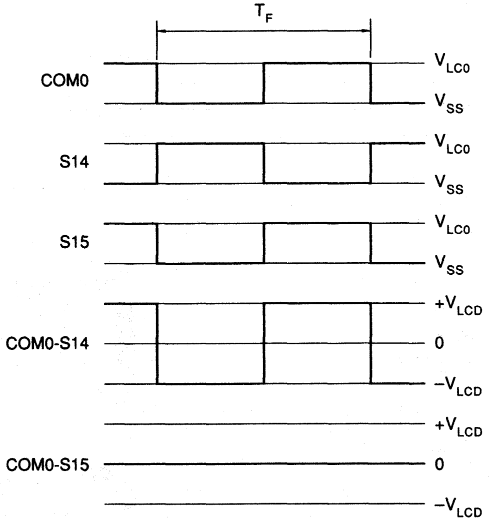


**Fig. 5.7-15 Static LCD Drive Waveform Example**

(2) Division by 2 display example

Fig. 5.7-18 shows connection of the division by 2 mode 5-digit LCD panel having the display pattern shown in Fig. 5.7-17, the μPD75328 segment signals (S12-S31), and the common signals (COM0 and COM1). In the example, 123.45 is displayed. The contents of the display data memory (addresses 1ECH-1FFH) correspond to the display pattern.

Here, 3. ( 3.) at the third digit position is taken as an example. It is necessary to output the selection and no-selection voltages as shown in Fig. 5.7-16 to the S20-23 pins at the timing of the COM0 and COM1 common signal according to the display pattern shown in Fig. 5.7-17.

| Segment / Common | S20 | S21 | S22 | S23 |
|---|---|---|---|---|
| COM0 | Selection | Selection | No-selection | No-selection |
| COM1 | Selection | Selection | Selection | Selection |

**Fig. 5.7-16 Selection/Non-selection Voltages of S8-S11 Pins (Division by 2 Mode Display Examples)**

This shows that for example, the display data memory address 1F7H bits corresponding to S23 need to be set to xx10. Fig. 5.7-19 shows an LCD drive waveform example among S23 and common signals. This shows an alternating current square wave of $+V_{LCD}/-V_{LCD}$ that is the LCD on level being generated when S23 is the selection voltage at the COM1 selection timing.

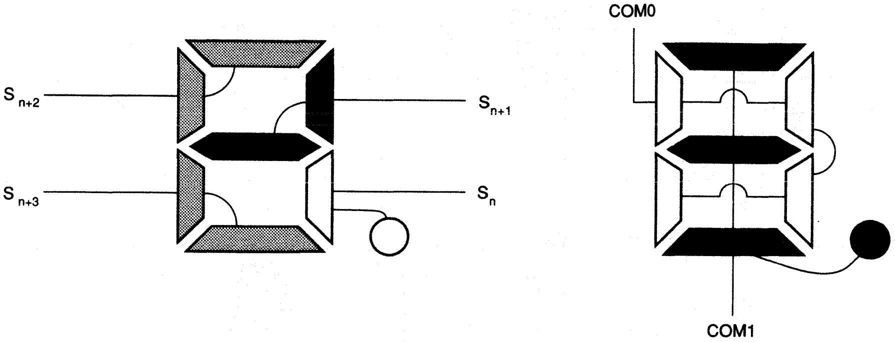
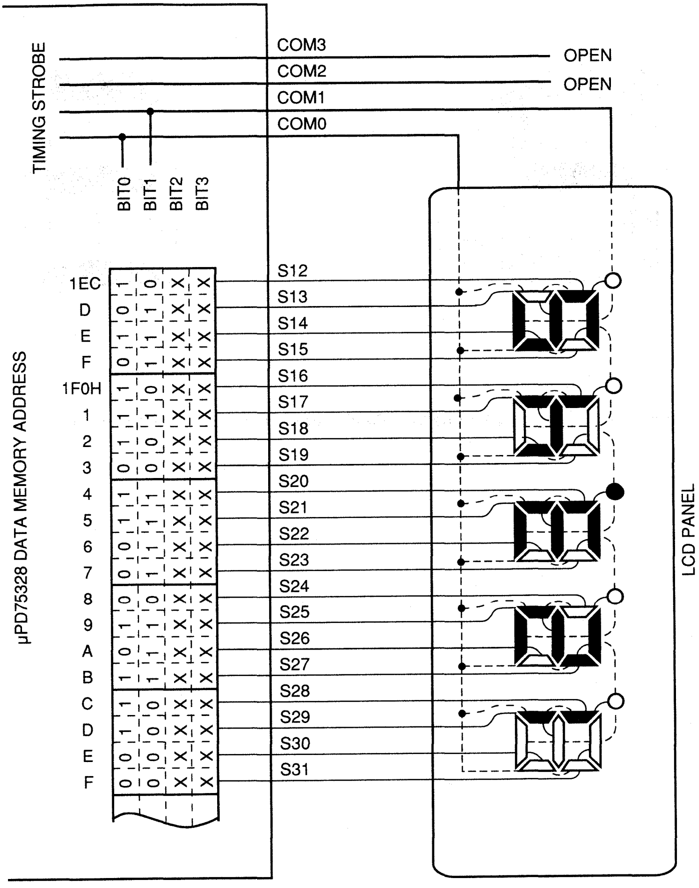


**Fig. 5.7-17 Division by 2 Mode LCD Display Pattern and Electrode Connection**



| | BIT0 | BIT1 | BIT2 | BIT3 |
|---|---|---|---|---|
| 1EC | 1 | 0 | X | X |
| D | 0 | 1 | X | X |
| E | 1 | 1 | X | X |
| F | 0 | 1 | X | X |
| 1F0H | 1 | 0 | X | X |
| 1 | 1 | 1 | X | X |
| 2 | 1 | 0 | X | X |
| 3 | 0 | 0 | X | X |
| 4 | 1 | 1 | X | X |
| 5 | 1 | 1 | X | X |
| 6 | 0 | 1 | X | X |
| 7 | 0 | 1 | X | X |
| 8 | 0 | 0 | X | X |
| 9 | 1 | 1 | X | X |
| A | 0 | 1 | X | X |
| B | 1 | 1 | X | X |
| C | 1 | 0 | X | X |
| D | 1 | 0 | X | X |
| E | 0 | 0 | X | X |
| F | 0 | 0 | X | X |

X: Any data can be stored at any time because of division by 2 display.

**Fig. 5.7-18 Division by 2 LCD Panel Connection Example**

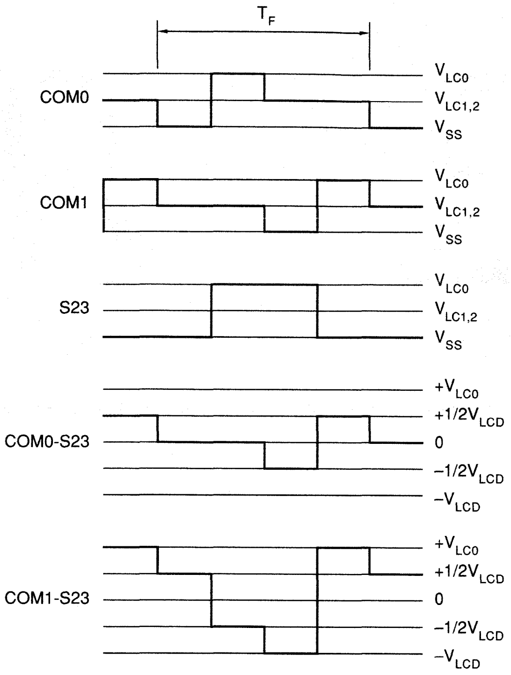


**Fig. 5.7-19 Division by 2 LCD Drive Waveform Example (1/2 Bias Law)**

(3) Division by 3 display example

Fig. 5.7-22 shows connection of a division by 3 mode 6-digit LCD panel having the display pattern shown in Fig. 5.7-21, the μPD75328 segment signals (S12-S29), and the common signals (COM0-COM2). In this example, 12345.6 is displayed. The contents of the display data memory (addresses 1ECH-1FDH) correspond to the display pattern.

Here, 5. (5.) at the second-digit position is taken as an example. It is necessary to output the selection and no-selection voltages as shown in Fig. 5.7-20 to the S15-S17 pins at the COM0-COM2 common signal timings according to the display pattern shown in Fig. 5.7-21.

| Segment / Common | S15 | S16 | S17 |
|---|---|---|---|
| COM0 | No-selection | Selection | Selection |
| COM1 | Selection | Selection | No-selection |
| COM2 | Selection | Selection | - |

**Fig. 5.7-20 Selection/Non-selection Voltages of S15-S17 Pins (Division by 3 Mode Display Examples)**

This shows that the bits at display data memory address 1EFH corresponding to S15 need to be set to x110.

Fig. 5.7-23 and 5.7-24 show LCD drive waveforms among S15 and common signals (1/2 and 1/3 bias law). These shown an alternating current square wave of $+V_{LCD}/-V_{LCD}$ that is the selection voltage at the COM1 selection timing and S15 is the selection voltage at the COM2 selection timing.

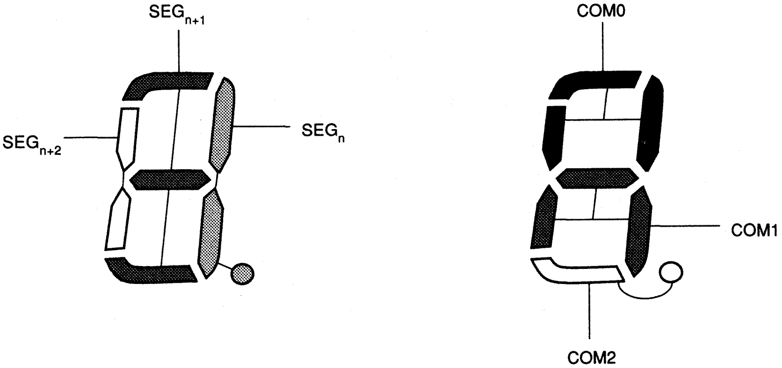
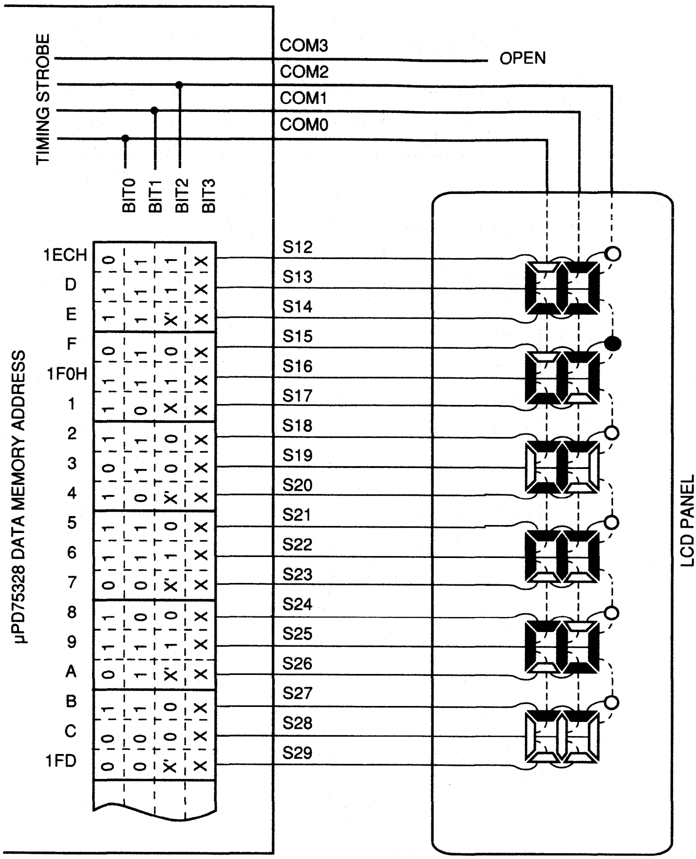


**Fig. 5.7-21 Division by 3 Mode LCD Display Pattern and Electrode Connection**



| μPD75328 DATA MEMORY ADDRESS | BIT0 | BIT1 | BIT2 | BIT3 |
|---|---|---|---|---|
| 1ECH | 0 | 1 | 1 | X |
| D | 1 | 1 | 1 | X |
| E | 1 | 1 | X' | X |
| F | 0 | 1 | 0 | X |
| 1F0H | 1 | 1 | 1 | X |
| 1 | 1 | 0 | X | X |
| 2 | 1 | 1 | 0 | X |
| 3 | 0 | 1 | 0 | X |
| 4 | 1 | 0 | X' | X |
| 5 | 1 | 1 | 0 | X |
| 6 | 1 | 1 | 1 | X |
| 7 | 0 | 0 | X' | X |
| 8 | 1 | 0 | 0 | X |
| 9 | 1 | 1 | 1 | X |
| A | 0 | 1 | X' | X |
| B | 1 | 1 | 0 | X |
| C | 0 | 0 | 0 | X |
| 1FD | 0 | 0 | X' | X |

X': Any data can be stored because the LCD panel does not have a corresponding segment.
X : Any data can be stored because of division by 3 display.

**Fig. 5.7-22 Division by 3 LCD Panel Connection Example**

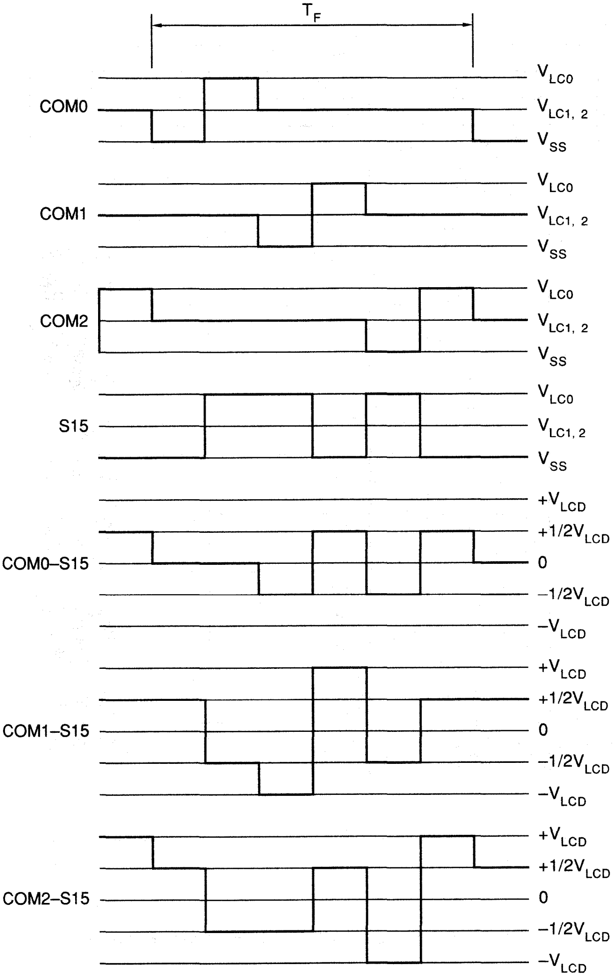


**Fig.5.7-23 Division by 3 LCD Drive Waveform Example (1/2 Bias Law)**

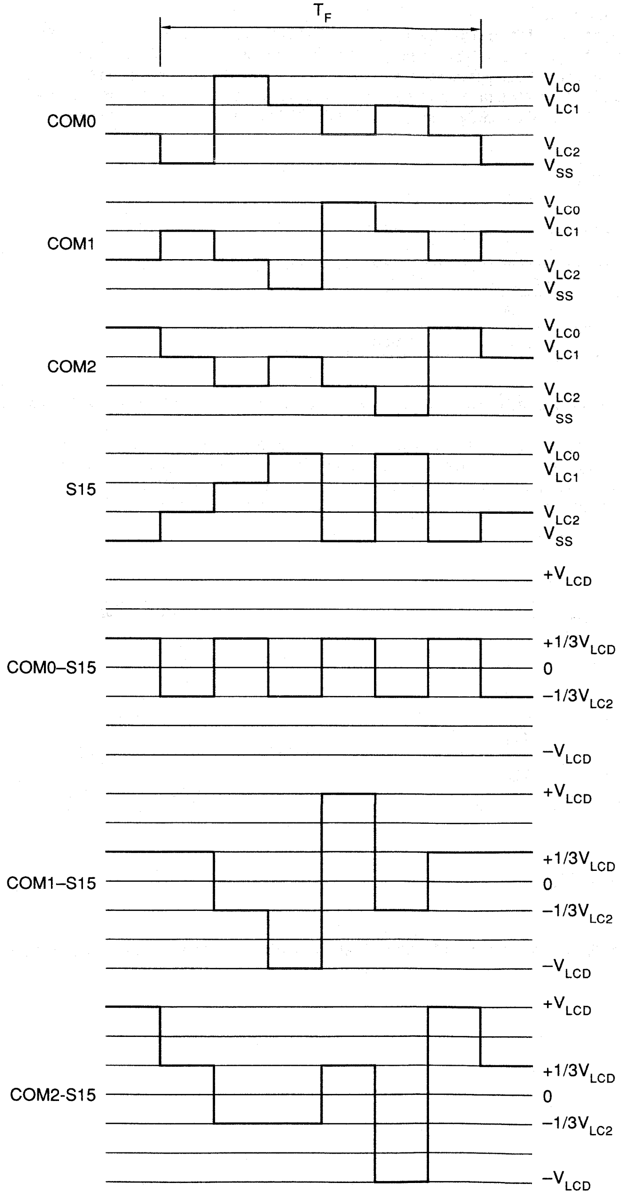


**Fig.5.7-23 Division by 3 LCD Drive Waveform Example (1/3 Bias Law)**

(4) Division by 4 Display Example

Fig. 5.7-27 shows connection of the division by 4 mode 10-digit LCD panel having the display pattern shown in Fig. 5.7-26, the μPD75328 segment signals (S12-S31), and the common signals (COM0-COM3). In this example, 123456.7890 is displayed. The contents of the display data memory (addresses 1ECH-1FFH) correspond to the display pattern.

Here, 6. (б.) at the 5-digit position is taken as an example. It is necessary to output the selection and no-selection voltages as shown in Fig. 5.7-25 to the S20 and S21 pins at the COM0-COM3 common signal timing according to the display pattern shown in Fig. 5.7-26.

| Segment / Common | S20 | S21 |
|---|---|---|
| COM0 | Selection | Selection |
| COM1 | No-selection | Selection |
| COM2 | Selection | Selection |
| COM3 | Selection | Selection |

**Fig. 5.7-25 Selection/Non-selection Voltage of S20 and S21 (Division by 4 Mode Display Examples)**

This shows that the bits at display data memory address 1F4H corresponding to S20 need to be set to 1101.

Fig. 5.7-28 shows LCD drive waveforms for S20, COM0, and COM1 signals. (Waveforms for COM2 and COM3 are omitted because of drawing space.) This shows an alternating current square wave of +$V_{LCD}$/–$V_{LCD}$ that is the LCD on level being generated when S20 becomes the selection voltage at the COM0 selection timing.

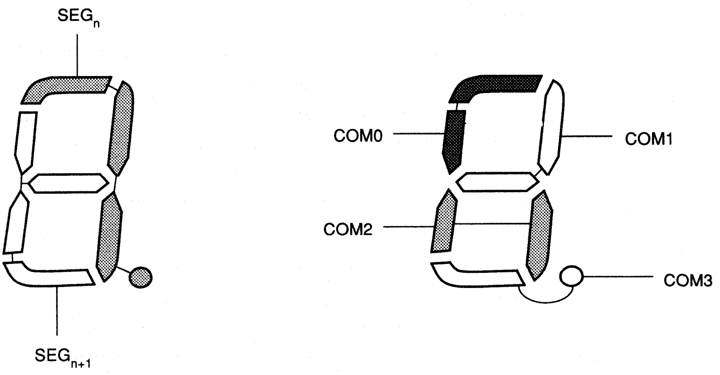
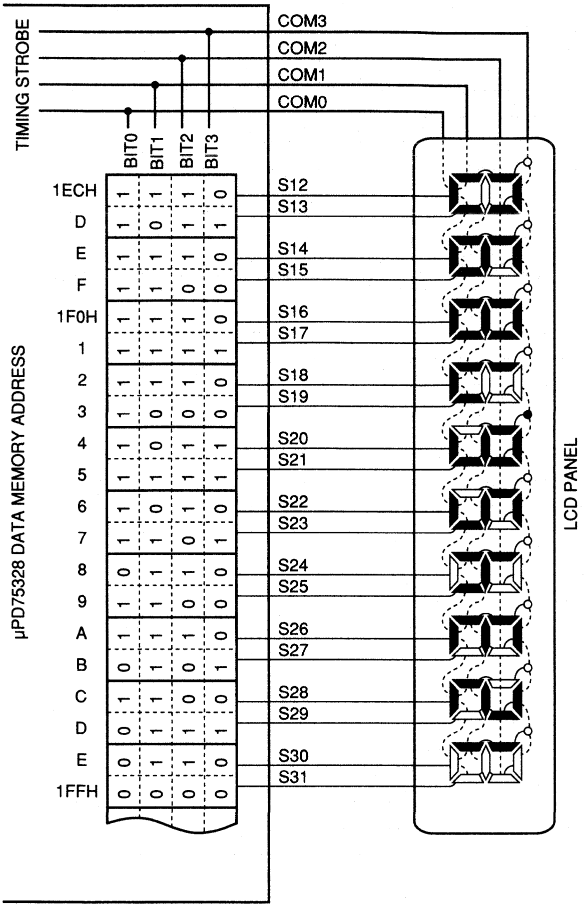


**Fig. 5.7-26 Division by 4 Mode LCD Display Pattern and Electrode Connection**



| | BIT0 | BIT1 | BIT2 | BIT3 |
|---|---|---|---|---|
| 1ECH | 1 | 1 | 1 | 0 |
| D | 1 | 0 | 1 | 1 |
| E | 1 | 1 | 1 | 0 |
| F | 1 | 1 | 0 | 0 |
| 1F0H | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 1 | 0 |
| 3 | 1 | 0 | 0 | 0 |
| 4 | 1 | 0 | 1 | 1 |
| 5 | 1 | 1 | 1 | 1 |
| 6 | 1 | 0 | 1 | 0 |
| 7 | 1 | 1 | 0 | 1 |
| 8 | 0 | 1 | 1 | 0 |
| 9 | 1 | 1 | 0 | 0 |
| A | 1 | 1 | 1 | 0 |
| B | 0 | 1 | 0 | 1 |
| C | 1 | 1 | 0 | 0 |
| D | 0 | 1 | 1 | 1 |
| E | 0 | 1 | 1 | 0 |
| 1FFH | 0 | 0 | 0 | 0 |

**Fig. 5.7-27 Division by 4 LCD Panel Connection Example**

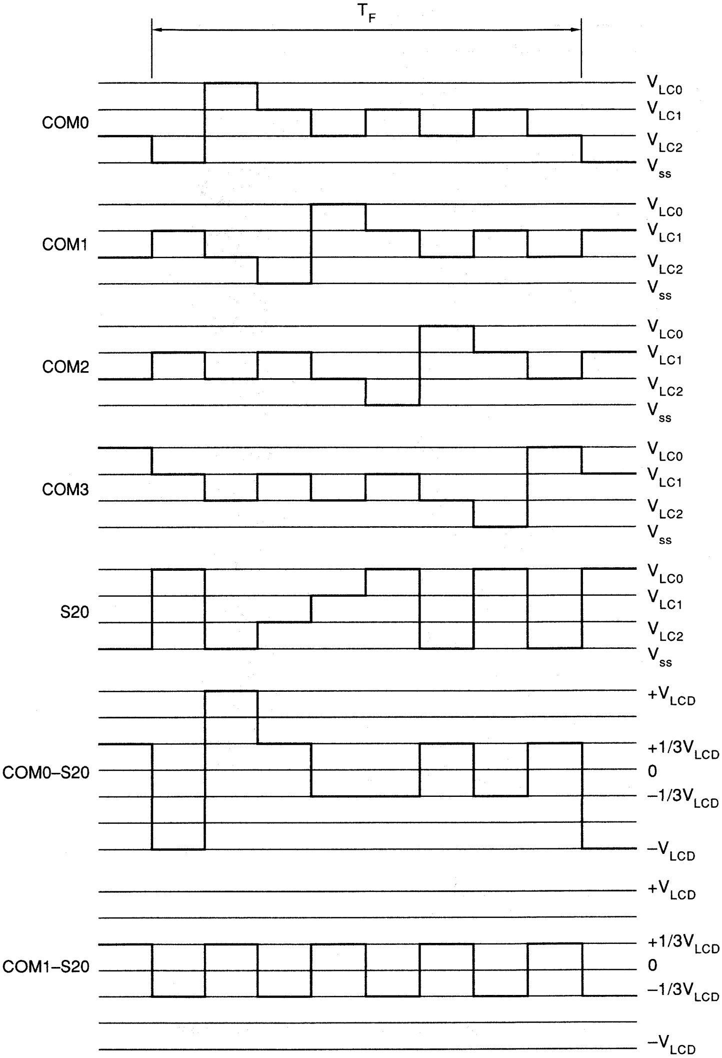


**Fig. 5.7-28 Division by 4 LCD Drive Waveform Example (1/3 Bias Method)**

### 5.8 A/D Converter

The μPD75328 contains an internal analog/digital (A/D) converter of 8-bit precision having six channels of analog input (AN0-AN5).
The A/D converter adopts the successive comparison method.

5.8.1 A/D converter configuration
Figure 5.8.1 shows the A/D converter configuration.

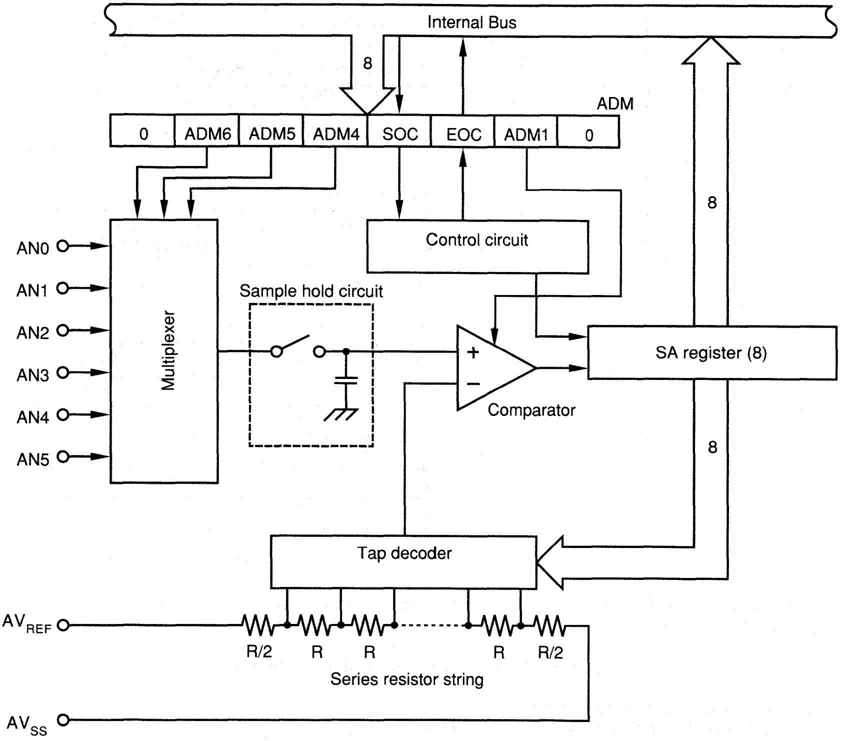


**Fig. 5.8.1 A/D Converter Block Diagram**

(1) A/D converter pins

(a) AN0-AN5

AN0-AN5 are 6-channel analog signal input pins to the A/D converter. Analog signals to be converted into digital form are input.

The A/D converter contains a sample hold circuit which holds the analog input voltage during the A/D conversion.

Caution: Use the AN0-AN5 input voltage within the prescribed limits. If particulary a voltage more than $V_{DD}$, less than $V_{SS}$ (even if the voltage is within the maximum absolute limits) is input, the channel conversion value becomes undefined and other channel conversion values may also be adversely affected.

(b) A/D conversion mode register

The A/D conversion mode register (ADM) is an 8-bit register which selects an analog input channel, indicates conversion start, and detects the conversion termination.

ADM is set by using an 8-bit handling instruction. Bit 2 (EOC) and bit 3 (SOC) can be handled bitwise. When the $\overline{\text{RESET}}$ signal is generated, ADM is initialized to 04H (only EOC is set to 1 and other bits are cleared).

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

Comarator bias voltage selection

| | | |
|---|---|---|
| ADM1 | 0 | When $AV_{REF} \geq 0.65\ V_{DD}$ |
| | 1 | $2.5V \leq AV_{REF} \leq 0.65\ V_{DD}$ |

Conversion terminatrion detection flag (R)

| | | |
|---|---|---|
| EOC | 0 | During conversion |
| | 1 | Conversion termination |

Conversion start indication bit (W)

| | |
|---|---|
| SOC | A/D conversion is started by setting the bit<br>The bit is automatically cleared after the conversion starts. |

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 |

Caution: A/D conversion starts at the maximum $2^4/f_X$ second delay (3.81 μs when the clock is 4.19 MHz) after SOC is set. (See 5.8.2)

**Fig. 5.8.2 A/D Conversion Mode Register Format**

(2) SA register (successive approximation register)

The SA register is an 8-bit register which stores the A/D conversion results.

The SA register is read by using an 8-bit handling instruction. It is a read-only register; write operation or bit handling can not be performed on the register.

When the $\overline{\text{RESET}}$ signal is generated, the SA register is initialized to 7FH.

Caution: When A/D conversion is started by setting ADM register bit 3 (SOC) to 1, the conversion results in the SA register are destroyed and the SA register becomes undefined until new conversion results are stored.

Note: If the A/D converter reference voltage ($AV_{REF}$) is 0.65 $V_{DD}$ or less, the conversion precision may lower. To correct it, comparator bias voltage selection is made.

5.8.2 A/D converter operation

Analog input signal to be converted into digital form is specified by setting A/D conversion mode register bits 6 to 4 (ADM6-ADM4). A/D conversion is started by setting ADM bit 3 (SOC) to 1. SOC is automatically cleared after it is set. A/D conversion is made by the successive comparison method by using hardware. The 8-bit data resulting from the conversion is stored in the SA register. When the conversion terminates, ADM bit 2 (EOC) is set to 1. Fig. 5.8-3 shows the A/D conversion timing chart.

The use of the A/D converter is as follows:

① Select an analog input channel. (Set ADM6-ADM4.)
② Indicate A/D conversion start. (Set SOC.)
③ Wait for the A/D conversion termination. (Wait for EOC to be set or wait with software timer.)
④ Read the A/D conversion results. (Read the SA register.)

Caution 1: ① and ② can also be performed at the same time.

Caution 2: After SOC is set, the maximum $2^4/f_X$ second delay (3.81 μs when the clock is 4.19 MHz) is made from A/D conversion start to EOC clear. Test EOC in the time listed in Table 5.8-1 after SOC is set. Table 5.8-1 also lists the A/D conversion time.

**Table 5.8-1 Setting of SCC and PCC**

| SCC, PCC setup value | | | | A/D conversion time | Wait time until EOC test after SOC is set | Wait time until the A/D conversion termination after SOC is set. |
|---|---|---|---|---|---|---|
| SCC3 | SCC0 | PCC1 | PCC0 | | | |
| 0 | 0 | 0 | 0 | $2^4/f_X$ x 9 (34.3 μs/$f_X$ = 4.19 MHz) | Wait not required | 3 machine cycles |
| | | 1 | 0 | | 2 machine cycles | 20 machine cycles |
| | | 1 | 1 | | 4 machine cycles | 40 machine cycles |
| 0 | 1 | X | X | | Wait not required | Wait not required |
| 1 | X | X | X | Conversion operation stop | – | – |

X: don't care

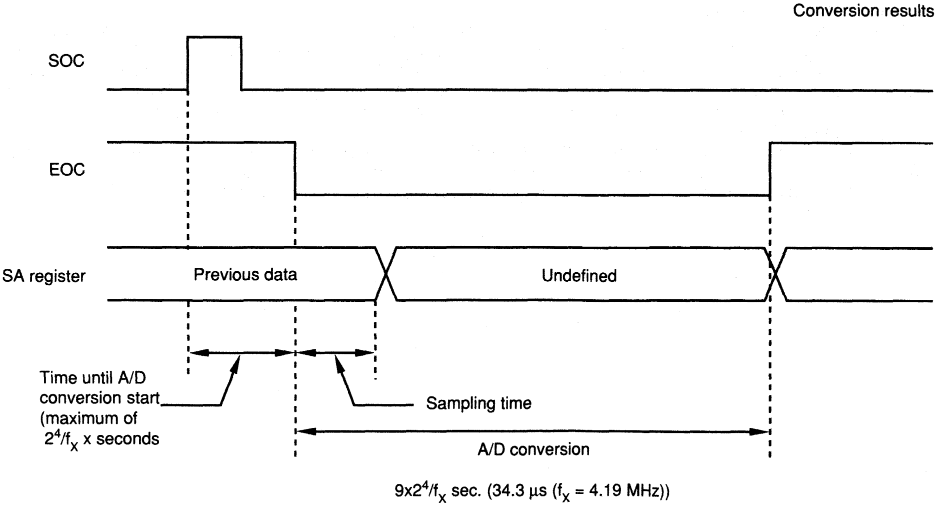

Conversion results
SOC
EOC
SA register
Previous data
Undefined
Time until A/D conversion start (maximum of $2^4/f_X$ x seconds
Sampling time
A/D conversion
$9x2^4/f_X$ sec. (34.3 μs ($f_X$ = 4.19 MHz))

**Fig. 5.8-3 A/D Conversion Timing Chart**

Fig. 5.8-4 showes the relationchip between analog input voltage and the A/D conversion results (8-bit digital data).

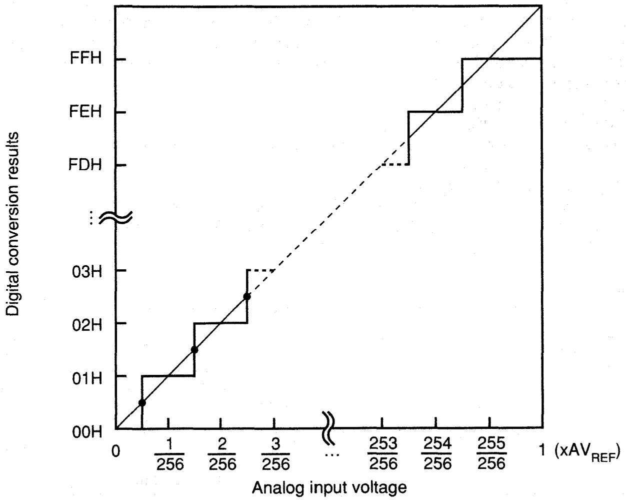


**Fig. 5.8-4 Relationchip Between Analog Input Voltage and A/D Conversion Results (ideal instance)**

5.8.3 Caution during standby mode

The A/D converter operates on the main system clock. Thus, operation stops in the STOP mode or the HALT mode (subsystem clock). Since current also flows into the $AV_{REF}$ pin at the time ,the current must be cut to reduce the entire system power consumption. The P83 pin which has the drive capability higher than other port pins can directly supply voltage to the $AV_{REF}$ pin. (See Fig. 5.8-5.) In this case, however, there is no precision in the actual $AV_{REF}$ voltage, thus the conversion value itself has no precision and can only be used for relative comparison. If a low level is output to the P83 pin during the standby mode, power consumption can be lessened.

The P83 pin of peripheral hardware emulator μPD75390 used during emulation using an evaluation kit has the same drive capability as other port pins. Thus, if an example shown in Fig. 5.8-5 is used, $AV_{REF}$ during the emulation is lower than $AV_{REF}$ when the μPD75328 is used actually, and the A/D conversion values are not made equal to each other.

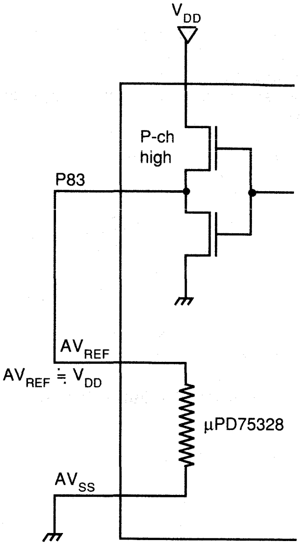
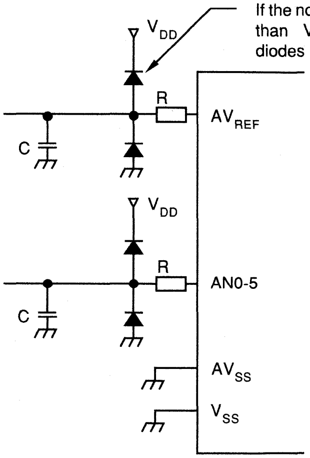


**Fig. 5.8-5 Example of Reducting Method of Power Consumption during Standby Mode**

5.8.4 A/D converter noise countermeasures

To keep the 8-bit precision, the noise to $AV_{REF}$ and AN0-5 pins must be carefully avoided. The higher the output impedance in analog input is, as more effective it is. The noise countermeasures are made by setting C shown as Fig. 5.8-6.



If the noise has the possibility of higher value than $V_{DD}$ and lower than $V_{SS}$, use clamp diodes lower than 0.3V.

Remark: C = 100 ~ 1000pF
R = ≥ 5kΩ

**Fig. 5.8-6 Analog Input Pin Processing**

### 5.9 Bit Sequential Buffer – 16 Bits

The bit sequential buffer is a special data memory for bit manipulation. Bit manipulation can be easily performed by changing address and bit specification in sequence. The buffer is useful for bitwise processing of long data.
The data memory consists of 16 bits. pmem.@L addressing of bit manipulation is enabled; indirect bit specification can be made by using the L register . Simply by incrementing or decrementing the L register in a program loop, processing can be performed while the specified bits are being moved in sequence.

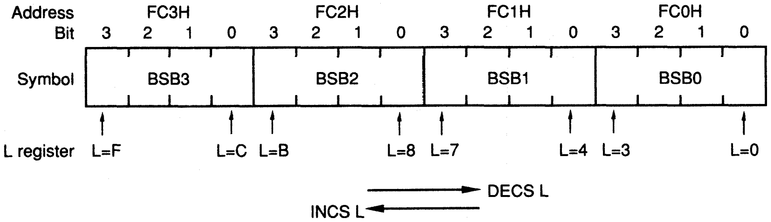


**Fig. 5.9-1 Bit Sequential Buffer Format**

Remarks:
In pmem.@L addressing, specified bits are moved according to the L register.

Data can also be handled in direct addressing. The 1–, 4–, and 8-bit direct addressing modes and pmem.@L addressing mode can be used in combination for continuous input and output of 1-bit data. In 8-bit manipulation, BSB0 and BSB2 are specified, and data is handled every high-order eight and low-order eight bits.

Example:
To output 16-bit data in BUFF 1 and BUFF2 from port 3 bit 0 serially.

```
        CLR1   MBE
        MOV    XA, BUFF1
        MOV    BSB0, XA    ; BSB0 and BSB1 are set.
        MOV    XA, BUFF 2
        MOV    BSB2, XA    ; BSB2 and BSB3 are set.
        MOV    L, #0
LOOP:   SKT    BSB0, @L    ; BSB-specified bit is tested.
        BR     LOOP1
        NOP                ; Dummy (timing adjustment)
        SET1   PORT3.0     ; Port 3 bit 0 is set.
        BR     LOOP2
LOOP1:  CLR1   PORT3.0     ; Port 3 bit 0 is cleared.
        NOP                ; Dummy (timing adjustment)
        NOP
LOOP2:  INCS   L           ; L← L+1
        BR     LOOP
               RET
```

## CHAPTER 6 INTERRUPT FUNCTION

The μPD75328 contains six vectored interrupt sources and two testable inputs for versatile application.
The μPD75328 interrupt control circuit has the following features to enable very high-speed interrupt service:

(a) Whether or not interrupts can be acknowledged can be controlled by using enable flag (IEXXX).
(b) The interrupt service start address and MBE during interrupt service can be set as desired by using a vector table. Starting the actual interrupt service program is fast.
(c) Interrupt request flag (IRQXXX) can be tested and cleared. Interrupt occurrence can be checked by using software.
(d) The standby mode (STOP or HALT) can be released by making an interrupt request. A backup release source is available by using the interrupt enable flag.

### 6.1 Interrupt Control Circuit Configuration

Fig. 6.1-1 shows configuration of the interrupt control circuit. The hardware devices are mapped in data memory.

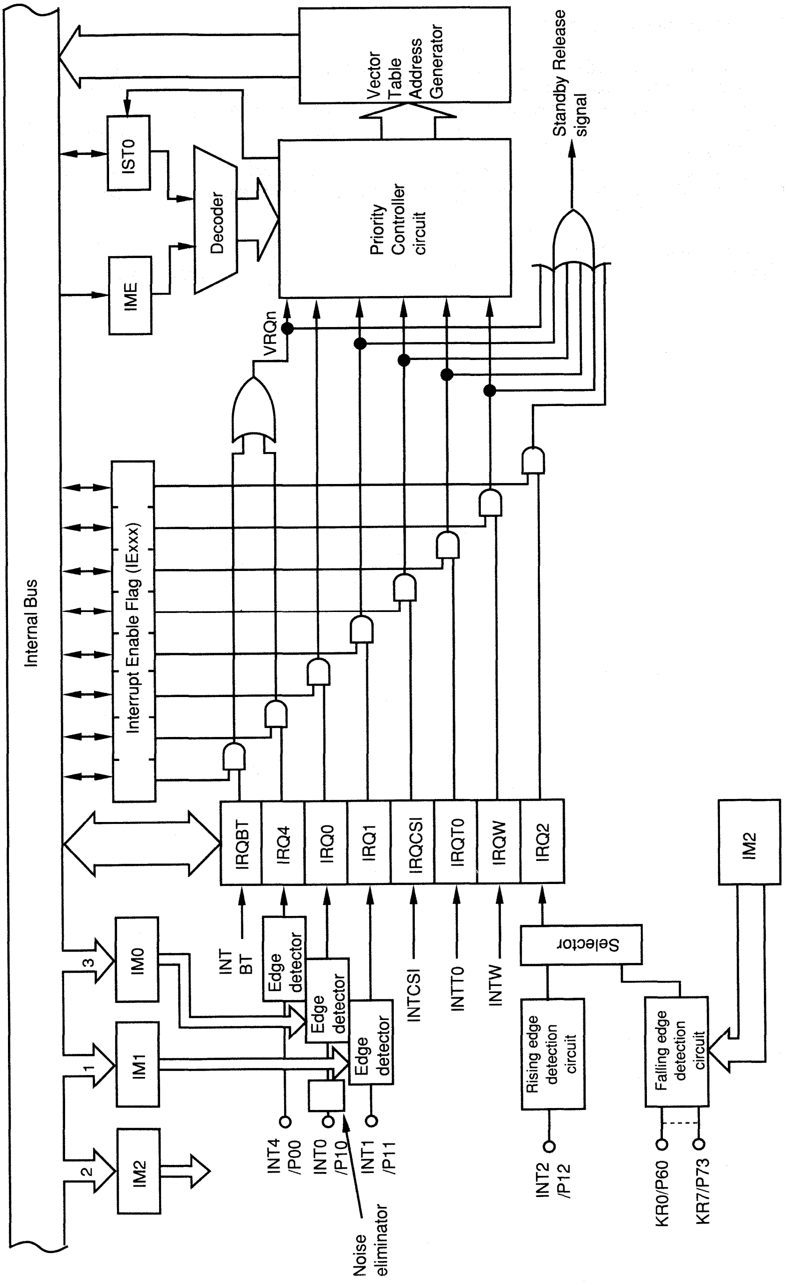


**Fig. 6.1-1 Interrupt Control Circuit Block Diagram**

### 6.2 Interrupt Source Types and Vector Table

Table 6.2-1 lists the types of interrupt sources for the μPD75328. Fig. 6.2-1 shows the interrupt vector table.

**Table 6.2-1 Interrupt Source Types**

| Interrupt source | | Internal /external | Interrupt priority (Note 1) | Vectored interrupt request signal (vector table address) |
|---|---|---|---|---|
| INTBT<br>(reference time interval signal from basic interval timer) | | Internal | 1 | VRQ1 (0002H) |
| INT4<br>(both rising and falling edge detection) | | External | | |
| INT0 | (selection of rising or falling edge detection) | External | 2 | VRQ2 (0004H) |
| INT1 | | External | 3 | VRQ3 (0006H) |
| INTCSI<br>(serial data transfer end signal) | | Internal | 4 | VRQ4 (0008H) |
| INTT0<br>(coincidence signal between programmable timer/ counter count register and modulo register) | | Internal | 5 | VRQ5 (000AH) |
| INT2<br>(rising edge detection of input to INT2 pin or falling edge detection of any input to KR0-KR7) (Note 2) | | External | Testable input signals (IRQ2 and IRQW are set) | |
| INTW<br>(signal from watch timer) | | Internal | | |

Note 1: The interrupt priority indicates the priority given to each interrupt when more than one interrupt request occurs at the same time.

Note 2: For details of INT2, see 6.3 (2).

| Address | | | | |
|---|---|---|---|---|
| 002H | MBE | 0 | 0 | INTBT/INT4 start address (high-order five bits) |
| | | | | INTBT/INT4 start address (low-order eight bits) |
| 004H | MBE | 0 | 0 | INT0 start address (high-order five bits) |
| | | | | INT0 start address (low-order eight bits) |
| 006H | MBE | 0 | 0 | INT1 start address (high-order five bits) |
| | | | | INT1 start address (low-order eight bits) |
| 008H | MBE | 0 | 0 | INTCSI start address (high-order five bits) |
| | | | | INTCSI start address (low-order eight bits) |
| 00AH | MBE | 0 | 0 | INTT0 start address (high-order five bits) |
| | | | | INTT0 start address (low-order eight bits) |

**Fig. 6.2-1 Interrupt Vector Table μPD75328**

The interrupt priority in Table 6.2-1 indicates the interrupt execution order when a number of interrupt requests occur at the same time or are held.

The interrupt service start addresses and MBE setup values during interrupt service are written into the vector table. The vector table is set by using an assembler pseudo-instruction (VENTn).

Example: To set INTBT/INT4 in vector table

```
VENT1        MBE=0, RBE=0,        GOTOBT
  ①            ②      ③             ④
```

① Vector table address 0002.

② MBE setup value in interrupt service routine.

③ Always set RBE to 0.

④ This symbol denotes the interrupt service routine start address.

Caution 1: The vector table address specified in VENTn (n=1-5) becomes 2n address.
Caution 2: Always set RBE to 0 in VENTn.

Example: To set INTBT/INT4 and INTTO in vector table.

```
VENT1              MBE=0, RBE=0, GOTOBT
VENT5              MBE=0, RBE=0, GOTOT0
```

### 6.3 Hardware Devices of Interrupt Control Circuit

(1) Interrupt request and enable flags

The interrupt request flag (IRQXXX) is set to 1 when a given interrupt request occurs; it is automatically cleared when interrupt service is executed.

Although IRQBT and INT4 share the vector address, they differ in clear operation. (See 6.6.)

An interrupt enable flag (IEXXX) is provided for each interrupt request flag. An interrupt is enabled when the corresponding interrupt enable flag is set to 1 and disabled when the flag is set to 0.

When the interrupt request flag is set, and the interrupt enable flag enables an interrupt, a vectored interrupt request (VRQn) occurs. This signal is also used to release the standby mode.

The interrupt request and enable flags are handled by using the bit manipulation and 4-bit memory operation instructions. The flags can be directly handled by using the bit manipulation instruction regardless of how MBE is set. The interrupt enable flags are handled by using the EI IEXXX and DI IEXXX instructions. Normally, the interrupt request flags are tested by using the SKTCLR instruction.

```
Example:   EI         IE0        ; INT0 is enabled.
           DI         IE1        ; INT1 is disabled.
           SKTCLR     IRQCSI     ; If IRQCSI is set to 1, skip and clear.
```

If the interrupt request flag is set by using the instruction, a vectored interrupt is executed as if an interrupt occured although it did not actually occurs.

When the RESET signal is generated, the interrupt request and enable flags are cleared, and all interrupts are disabled.

**Table 6.3-1 Interrupt Request and Enable Flags**

| Interrupt request flag | Interrupt request flag set signal | Interrupt enable flag |
|---|---|---|
| IRQBT | Is set by reference time interval signal generated by the basic interval timer | IEBT |
| IRQ4 | Is set when either the rising or falling edge of the INT4/P00 pin input signal is detected. | IE4 |
| IRQ0 | Is set when the INT0/P10 pin input signal edge is detected. Detected edge is selected among rising and falling edges by using the INT0 mode register (IM0). | IE0 |
| IRQ1 | Is set when the edge of the INT1/P11 pin input signal is detected. Detected edge is selected from rising and falling edges by using the INT1 mode register (IM1). | IE1 |
| IRQCSI | Is set by serial data transfer end signal on serial interface. | IECSI |
| IRQT0 | Is set by coincidence signal from timer/event counter 0. | IET0 |

**Table 6.3-1 Interrupt Request and Enable Flags (cont'd)**

| Interrupt request flag | Interrupt request flag set signal | Interrupt enable flag |
|---|---|---|
| IRQW | Is set by signal from the watch timer. | IEW |
| IRQ2 | Is set when the rising edge of the INT2/P12 pin input signal is detected or the falling edge of any input to the KR0/P60 to KR7/P73 pins is detected. | IE2 |

(2) Noise eliminator and edge detection mode registers.

Figs. 6.3-1 and 6.3-2 show the relationship of INT0 and INT1, INT2 and KR0-KR7 pins, respectivvely, within the interrupt control circuit. They are used for inputting external interrupts in cases where noise can be eliminated by the sampling clock and the detected edge can be selected.

Any pulse narrower than the sampling clock is determined to be a noise pulse and is eliminated by the noise eliminator. A pulse twice as wide (or more) as the sampling clock is acknowledged as a valid interrupt signal.

For INT0 the sampling clock can be changed to two stages.

Note that when the INT0 pin is used for P10 (port pin), the signal is also input via the noise eliminator.

Caution: Since INT0 sampling uses clock, INT0 does not operate at standby mode.

IRQ2 is set in either of the following modes:

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

(b) When the falling edge of any input to KR0-KR7 pins is detected:

When the falling edge of any input to pins selected among KR0-KR7 by using the edge detection mode register (IM2) is detected, IRQ2 is set.

For example, if KR4-KR7 are selected, IRQ2 is set if the falling edge is input to any one of KR4-KR7, because KR4-KR7 inputs are ANDed in the μPD75328.

Fig. 6.3-4 shows the format of the edge detection mode registers used to select detected edge (IM0 to IM2). IM0 to IM2 are set individually by using a 4-bit memory operation instruction.

When RESET signal is generated, all the register bits are cleared; INT0, INT1, and INT2 detection edges are specified to rising edges.

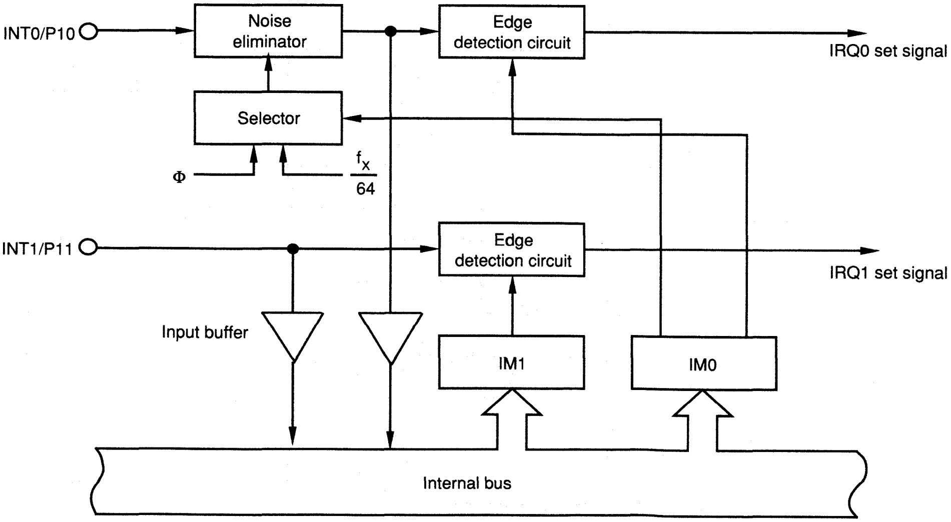


**Fig. 6.3-1 INT0 and INT1 Configuration**

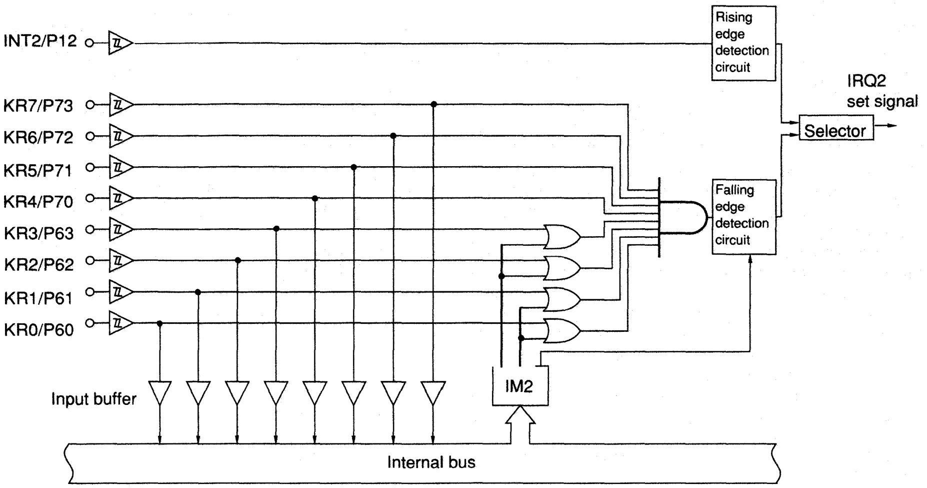


**Fig. 6.3-2 INT2 and KR0-KR7 Configuration**

When a high level is input successively, the noise eliminator outputs a high level; when a low level is input successively, it outputs a low level.

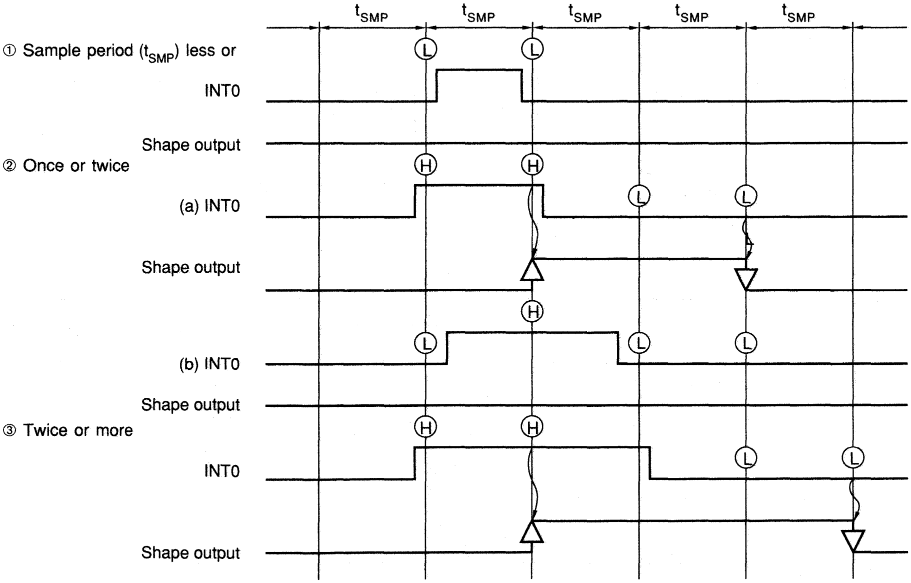
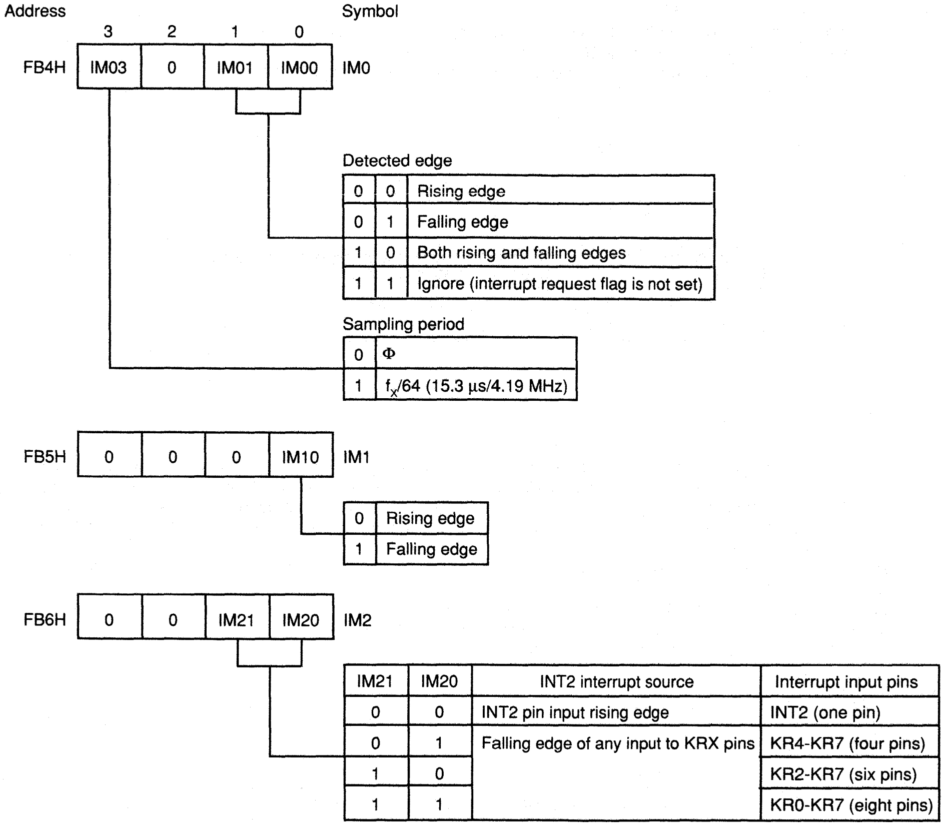


**Fig. 6.3-3 Noise Eliminator Input/Output Timing**



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

Detected edge

| | | |
|---|---|---|
| 0 | 0 | Rising edge |
| 0 | 1 | Falling edge |
| 1 | 0 | Both rising and falling edges |
| 1 | 1 | Ignore (interrupt request flag is not set) |

Sampling period

| | |
|---|---|
| 0 | Φ |
| 1 | $f_X$/64 (15.3 μs/4.19 MHz) |

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

| | |
|---|---|
| 0 | Rising edge |
| 1 | Falling edge |

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

| IM21 | IM20 | INT2 interrupt source | Interrupt input pins |
|---|---|---|---|
| 0 | 0 | INT2 pin input rising edge | INT2 (one pin) |
| 0 | 1 | Falling edge of any input to KRX pins | KR4-KR7 (four pins) |
| 1 | 0 | | KR2-KR7 (six pins) |
| 1 | 1 | | KR0-KR7 (eight pins) |

Caution: If the edge detection mode register is changed, the interrupt request flag may be set. Disable interrupts beforehand, and change the mode register. Clear the interrupt request flag by using the CLR1 instruction before enabling interrupts. If $f_X$/64 is selected for the sampling clock in an IM0 change, clear the interrupt request flag within 16 machine cycles after the mode register is changed.

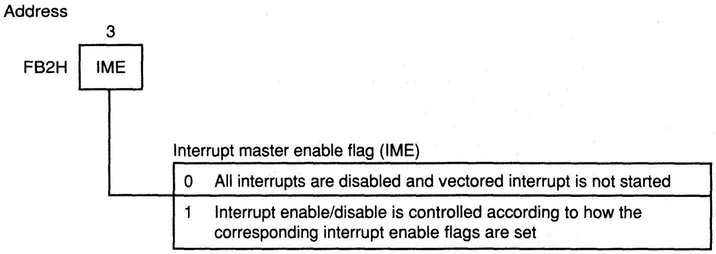
**Fig. 6.3-4 Edge Detection Mode Register Format**

(3) Interrupt master enable flag (IME)

Interrupt master enable flag enables or disables acknowledgement of all interrupts.

IME is set to 1 or to 0 by using the EI and DI instructions.

When the RESET signal is generated, IME is cleared, disabling acknowledgement of all interrupts.



Address

3

FB2H IME

Interrupt master enable flag (IME)

| | |
|---|---|
| 0 | All interrupts are disabled and vectored interrupt is not started |
| 1 | Interrupt enable/disable is controlled according to how the corresponding interrupt enable flags are set |

**Fig. 6.3-5 IME Format**

(4) Interrupt status flag

The interrupt status flag (IST0) indicates the current status of processing being performed by CPU; it is contained in the PSW. The interrupt control circuit controls multi-interrupts (as listed in Table 6.3-2) according to the contents of interrupt status flag. Since IST0 can be changed by using a 4-bit or a one-bit manipulation instruction, multi-interrupt can also be implemented by changing the status of processing being performed. Multi-interrupt can always be implemented regardless of the MBE setting when IST0 is handled bitwise.

Be sure to execute the DI instruction to disable interrupts before handling IST0 and the EI instruction to enable interrupts after handling IST0.

When an interrupt is acknowledged, IST0 is saved in stack memory together with other PSW bits, then it is automatically set to 1. When the RETI instruction is executed, the former IST0 value (0) is restored.

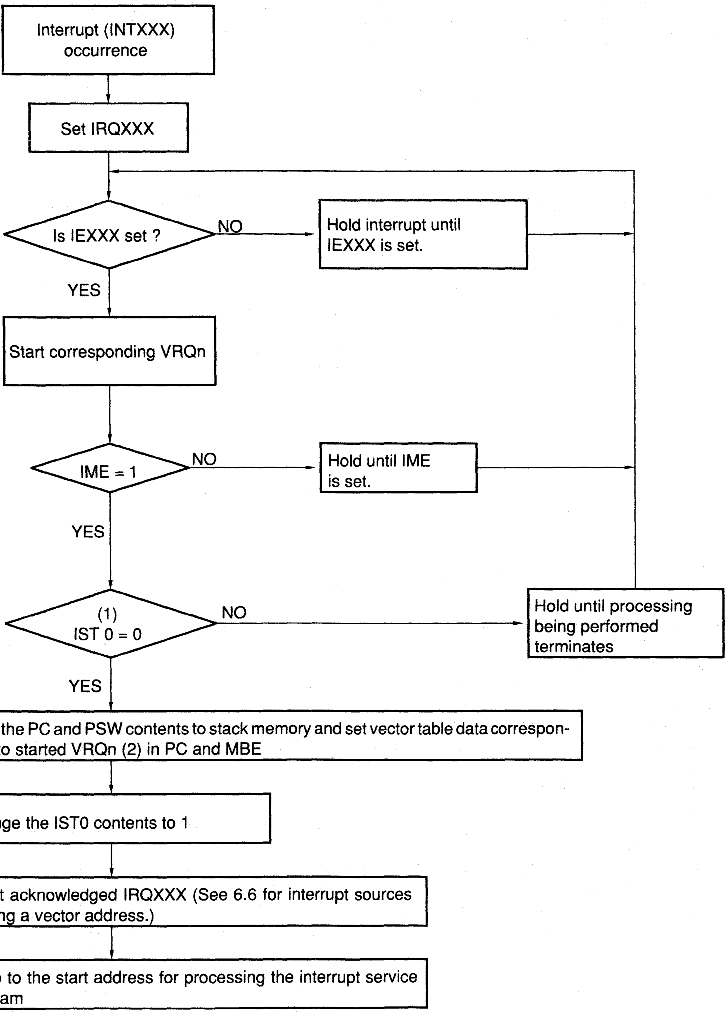
When the $\overline{\text{RESET}}$ signal is generated, the flag is cleared.

**Table 6.3-2 IST0 and Interrupt Service State**

| IST0 | Status of processing | CPU processing contents | Interrupt requests that can be acknowledged | After interrupt is acknowledged |
|---|---|---|---|---|
| | | | | IST0 |
| 0 | Status 0 | During normal program processing | All interrupts can be acknowledged. | 1 |
| 1 | Status 1 | During interrupt service | None of the interrupts can be acknowledged. | – |

### 6.4 Interrupt Sequence

When an interrupt occurs, it is processed as shown in Fig. 6.4-1.


Interrupt (INTXXX) occurrence
Set IRQXXX
Is IEXXX set ?
NO
Hold interrupt until IEXXX is set.
YES
Start corresponding VRQn
IME = 1
NO
Hold until IME is set.
YES
(1) IST 0 = 0
NO
Hold until processing being performed terminates
YES
Save the PC and PSW contents to stack memory and set vector table data correspon-ding to started VRQn (2) in PC and MBE
Change the IST0 contents to 1
Reset acknowledged IRQXXX (See 6.6 for interrupt sources sharing a vector address.)
Jump to the start address for processing the interrupt service program

Remarks: 1. IST0: Interrupt status flag (PSW bit 2). (See Table 6.3-2)

2. Store the interrupt service program start addresses and MBE setup values at the interrupt start in the vector table.

**Fig. 6.4-1 Interrupt Service Flow**

### 6.5 Multi-interrupt Service Control

The μPD75328 enables multi-interrupts as described below.
As understood from Table 6.3-2, multi-interrupt is enabled if the interrupts status flag is changed by a program. That is, multi-interrupt is enabled if IST0 is changed to 0 by the interrupt service program setting status 0.
To change IST0, execute the DI instruction beforehand to disable interrupts.

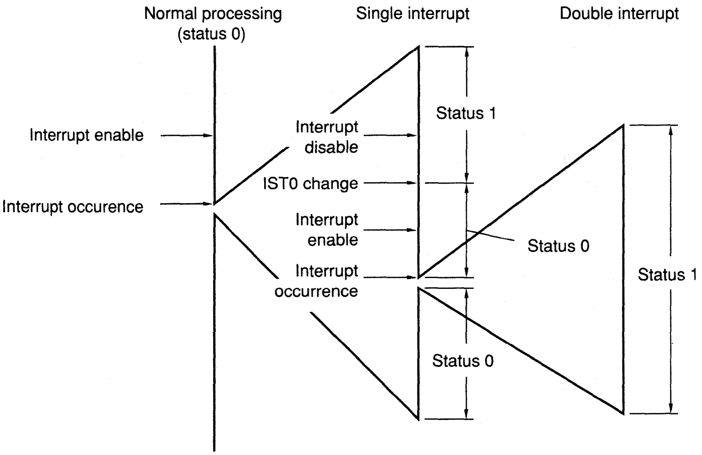


**Fig. 6.5-1 Multi-Interrupt by Changing Interrupt Status Flag**

### 6.6 Vector Address Share Interrupt Service

Since INTBT and INT4 interrupt sources share the vector table addresses, interrupt source selection is made as described below:

(1) To use one interrupt only

Set the interrupt enable flag to 1 for the required one of the two interrupt sources sharing the vector table addresses, and clear the interrupt enable flag of the other interrupt source.

In this case, an interrupt request is generated from the interrupt source corresponding to the interrupt enable flag that is set to 1 (IEXXX = 1). When the interrupt request is acknowledged, the interrupt request flag is cleared.

(2) To use both interrupts

Set both the interrupt enable flags of the two interrupt sources to 1. In this case, an interrupt request is made by ORing the interrupt request flags of the two interrupt sources.

Even if an interrupt request is acknowledged when either or both of the interrupt request flags are set to 1, the interrupt request flags are not reset.

Therefore, the interrupt service routine must decide which interrupt source the interrupt is generated from. This is accomplished by executing the SKTCLR instruction to check the interrupt request flags.

If both the interrupt request flags are set to 1 when the request flags are tested and cleared by execution of the SKTCLR instruction, the interrupt request is left even if one request flag is cleared. If IST0 is cleared, dual interrupt service is entered according to the left interrupt request.

Remarks: When only one interrupt is enabled, the source of an interrupt is known. Thus, the interrupt request flag is cleared by hardware when interrupt is acknowledged. When both interrupts are enabled, the source of an interrupt is not defined; thus, interrupt request flag cannot be cleared by hardware. Software is used to check the interrupt request flags and determine the interrupt source. Interrupt request flag is cleared by software.

Example: When INT4 takes precedence over INTBT

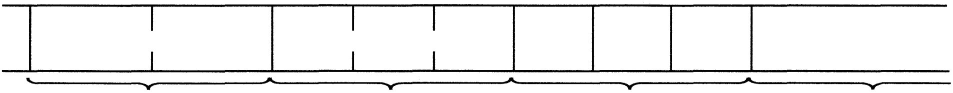
```
          DI
          SKTCLR    IRQ4      ; IRQ4 = 1 ?
          BR        VSUBBT    ⎫
           :                  ⎬ INT4 interrupt
          EI                  ⎪ service routine
          RETI                ⎭
           :
           :
VSUBBT:   CLR1      IRQBT
           :                  ⎫
           :                  ⎬ INTBT interrupt
           :                  ⎪ service routine
          EI                  ⎭
          RETI
```

### 6.7 Machine Cycles to Interrupt Service Start

The number of the μPD75328 machine cycles required to start execution of the interrupt routine after an interrupt request flag is set is as shown below:

(1) When IRQn is set during execution of an operating instruction at data memory address FBxH (interrupt hardware))

FBxH address operating instruction



IRQn is set during this interval

(depending on the number of machine cycles for the instruction)

One to three machine cycles

(depending on the number of machine cycles for the instruction)

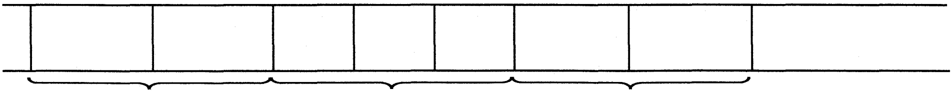
Three machine cycles

(required for interrupt service)

Interrupt routine execution

As shown above, interrupt routine processing is started a maximum of six machine cycles after the data memory (address FBxH) operating instruction is terminated. (Within a maximum of six machine cycles after the last operating instruction is terminated if the FBxH address operating instructions are successive.)

(2) When IRQn is set during execution of instruction other than in (1).



IRQn is set during this interval.

One to three machine cycles

(depending on the number of machine cycles for the instruction)

Three machine cycles

(required for interrupt service)

Interrupt routine execution

In this case, a maximum of six machine cycles are required.

### 6.8 Effective Use of Interrupts

Use the interrupt function as described below:

(1) Set MBE = 0 in the interrupt service routine.

If the data memory area used in the interrupt service routine is preferentially allocated to addresses 0-7FH and MBE = 0 is set in the interrupt vector table, a program can be prepared without considering memory banks.

If memory bank 1 must be used for by the program, save the memory bank select register by using the PUSH BS instruction and select memory bank 1.

(2) Use software interrupt for debugging.
When an interrupt request flag is set by an instruction, operating is performed in the same manner as if an interrupt occured. Debugging when more than one interrupt occurred at the same time can be done effeciently by using an instruction to set an interrupt request flag.

**6.9 Interrupt Application**

To use the interrupt function, first set the following in the main program:

(1) Set the interrupt enable flags to be used (EI IEXXX instructions).

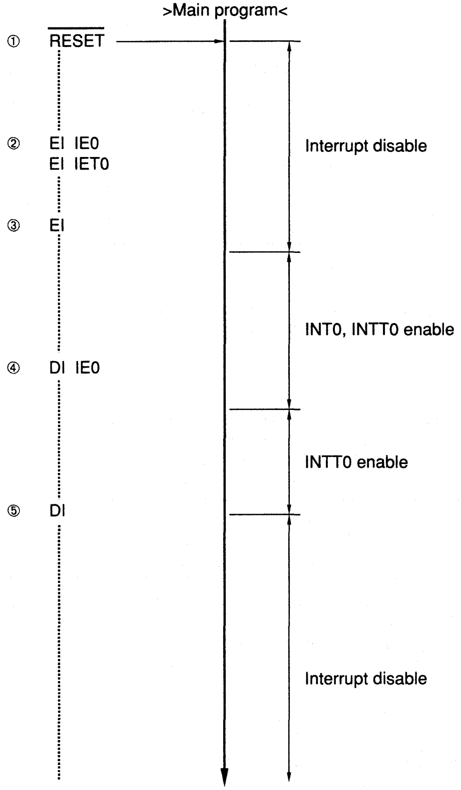
(2) To use INT0 and INT1, select the active edge. (set IM0 and IM1.)

(3) Set the interrupt master enable flag (EI instruction).

Since MBE is set by using the vector table in an interrupt service program, registers need not be saved or restored, and the interrupt program can be started immediately.

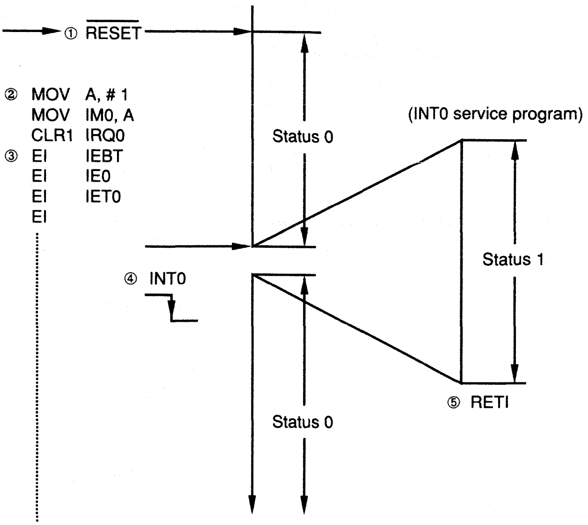
To return from the interrupt service program, use the RETI instruction.

(1) Interrupt enable and disable



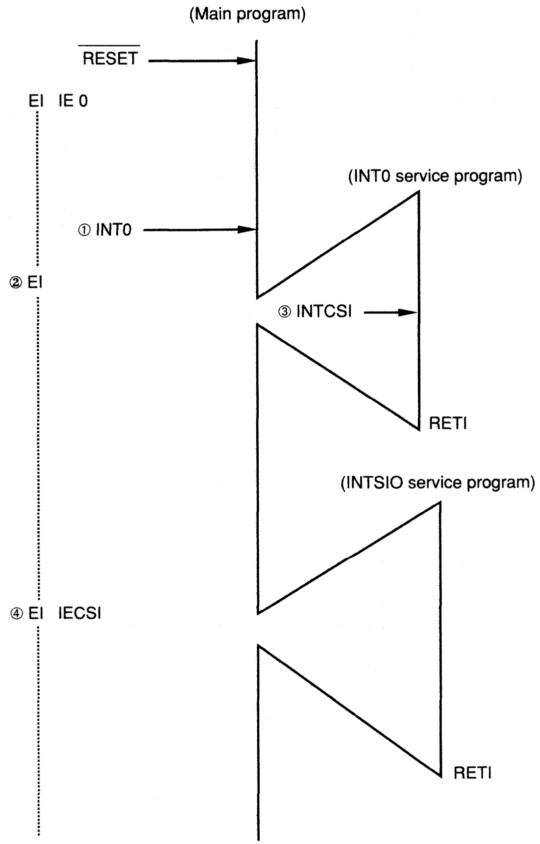
① All interrupts are disabled by the $\overline{\text{RESET}}$ signal.
② The interrupt enable flags are set by the EI IEXXX instructions. At this stage, all interrupts remain disabled.
③ The interrupt master enable flag is set by the EI instruction. At this stage, INT0 and INTT0 are enabled.
④ The interrupt enable flag is cleared by the DI IEXXX instruction and INT0 is disabled.
⑤ All interrupts are disabled by the DI instruction.

(2) Usage example of INTBT, INT0 (falling edge active), and INTT0. Multi-interrupt is not done.



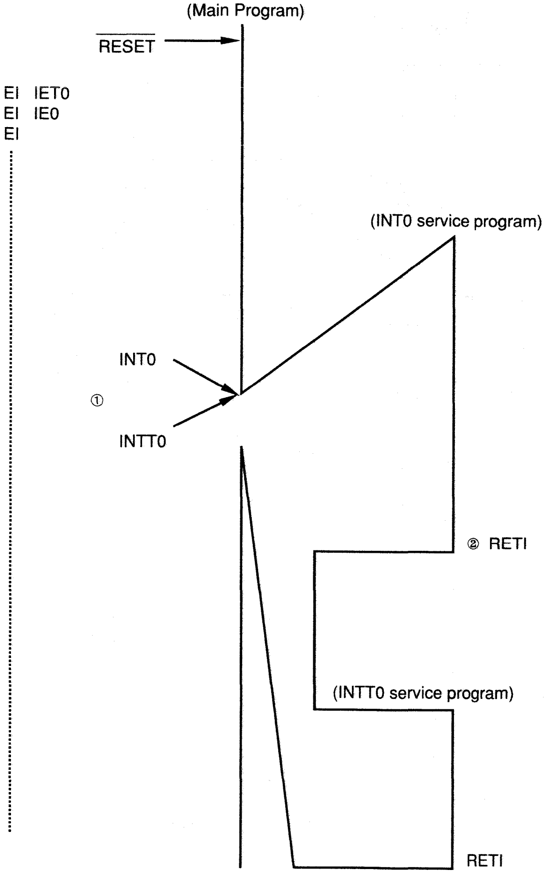
① All interrupts are disabled by the $\overline{\text{RESET}}$ signal and status 0 is set.
② Falling edge active is selected for INT0.
③ Interrupts are enabled by the EI and EI IEXXX instructions.
④ On the INT0 falling edge, the INT0 interrupt service program is started. The status is changed to status 1 and all interrupts are disabled.
⑤ A return is made from the interrupt service program by the RETI instruction. The status is restored to status 0 and interrupts are enabled.

(3) Pending interrupt execution – interrupt input during interrupt disable –



① If INT0 is set during interrupt disable, the request flag is held.
② When interrupts are enabled by the EI instruction, the INT0 service program is started.
③ Simular to ① above.
④ When INTCSI is held enabled, the INTCSI service program is started.

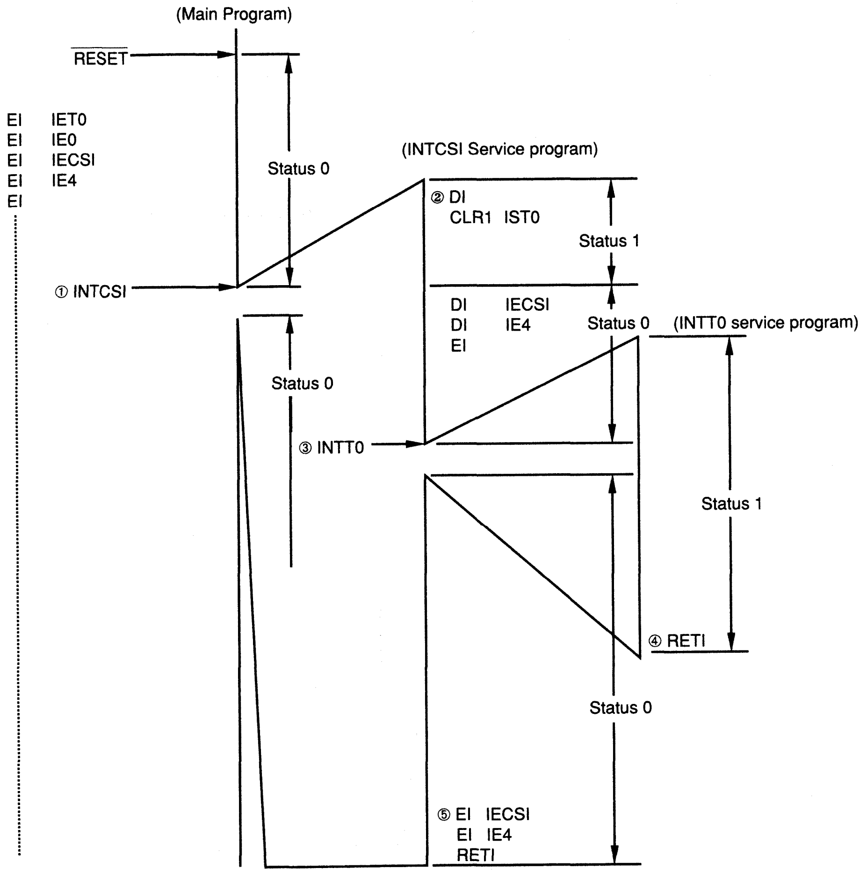
(4) Pending interrupt execution



① If INT0 and INTT0 are occured at the same time during execution of a single instruction, INT0, which is higher in priority than INTT0, is executed first; INTT0 is held.

② When a return is made from the INT0 service program by the RETI instruction, the held INTT0 service program is started.

(5) Enable of two double interrupts – INTT0 and INT0 enable double interrupts; INTCSI and INT4 are single interrupts –



① When interruptINTCSI occurs, which does not enable double interrupts, the INTCSI service program is started. Status 1 is set.
② The status is changed to status 0 by clearing IST0. INTCSI and INT4 do not enable double interrupt and are disabled.
③ When INTT0 enables double interrupts, the double interrupt is executed, the status is changed to status 1, and all interrupts are disabled.
④ At the termination of INTT0 service, the status is restored to status 0.
⑤ Disabled INTCSI and INT4 are enabled and a return is made by the RETI instruction.

## 7. STANDBY FUNCTION

To make the most of low current consumption, which is one feature of the CMOS process, the μPD75328 can stop CPU operation in the standby mode making current consumption by the CPU very small.
The μPD75328 standby mode includes the STOP mode and HALT mode.
The STOP mode stops the main system clock oscillator. In this mode, CPU current consumption consists almost entirely of leakage current. Data memory can also be held with low supply voltage (up to $V_{DD}$ = 2 V). This features is useful for maintaining the data memory contents with very low current consumption. Since the μPD75328 STOP mode can be released by using an interrupt request, intermittent operation can also be performed. However, if processing must be started immediately when an interrupt request is made, note that the wait time required to ensure oscillator stability is taken when the STOP mode is released.
The HALT mode continues system clock oscillator operation but stops the CPU clock (Φ) supply; thus, CPU operation is stopped. Although the HALT mode is inferior to the STOP mode for reduction of current consumption, it is useful to restart processing immediately according to an interrupt request or for performing intermittent operations such as watch operation.
In either mode, all the register, flag, and data memory contents immediately before the standby mode is entered are held. The input/output port output latch state and output buffer state are also held. The input/output port state is handled beforehand so that the current consumption of the entire system is minimized.

Cautions on Use of Standby Mode:

1. The STOP mode can be used only when μPD75328 operation uses the main system clock. (Subsystem clock oscillation cannot be stopped.) The HALT mode can be used when the μPD75328 uses either main system or subsystem clock.
2. If the STOP mode is set when the LCD controller/driver and watch clock operate on the main system clock $f_x$, they stop operation. To continue operation, change to the subsystem clock $f_{XT}$ before setting STOP mode.
3. Although efficient operation with low current consumption and low voltage can be performed by using the clock change function between the CPU and system clocks in combination with the standby mode, time (described in 5.2.3) is required from selection of a new clock by setting the control register until operation is started by the newly selected clock.
   Thus, to use the clock change function and the standby mode in combination, set the standby mode within the time required for the clock change.

### 7.1 Standby Mode Setting and Operating State

**Table 7.1-1 Operating State in Standby Mode**

| | | STOP mode | HALT mode |
|---|---|---|---|
| Setting instruction | | STOP instruction | HALT instruction |
| System clock when standby mode is set. | | Can be set only during main system clock. | Can be set during either main system or subsystem clock. |
| Operating state | Clock oscillator | Only the main system clock oscillator is stopped. | Only CPU clock Φ is stopped (oscillation is continued). |
| | Basic interval timer | Operation stop | Operation (IRQBT is set at reference time intervals.) |
| | Serial interface | Can operate only when external $\overline{\text{SCK}}$ input is selected for serial clock. | Can operate. |
| | Timer/event counter | Can operate only when TI0 pin input is selected for count clock. | Can operate. |
| | Watch timer | Can operate when $f_{XT}$ is selected for count clock. | Can operate. |
| | LCD controller | Can operate only when $f_{XT}$ is selected for LCDCL. | Can operate. |
| | A/D converter | Operation stop | Can operate |
| | External interrupts | INT1, INT2, and INT4 can operate. Only INT0 cannot operate. | |
| | CPU | Operation stop | |
| Release signal | | Interrupt request signal enabled with interrupt enable flag from operating hardware (exept for INT0), or $\overline{\text{RESET}}$ input. | |

The STOP mode is set by using the STOP instruction to set PCC bit 3; the HALT mode is set by the using HALT instruction to set PCC bit 2.

To change the CPU clock by using the low-order two bits of PCC, a time lag may occur from PCC rewrite to CPU clock change. Thus, to change the clock before the standby mode is entered or after the standby mode is released, set the standby mode within the number of machine cycles required to change the CPU clock after PCC is rewritten.

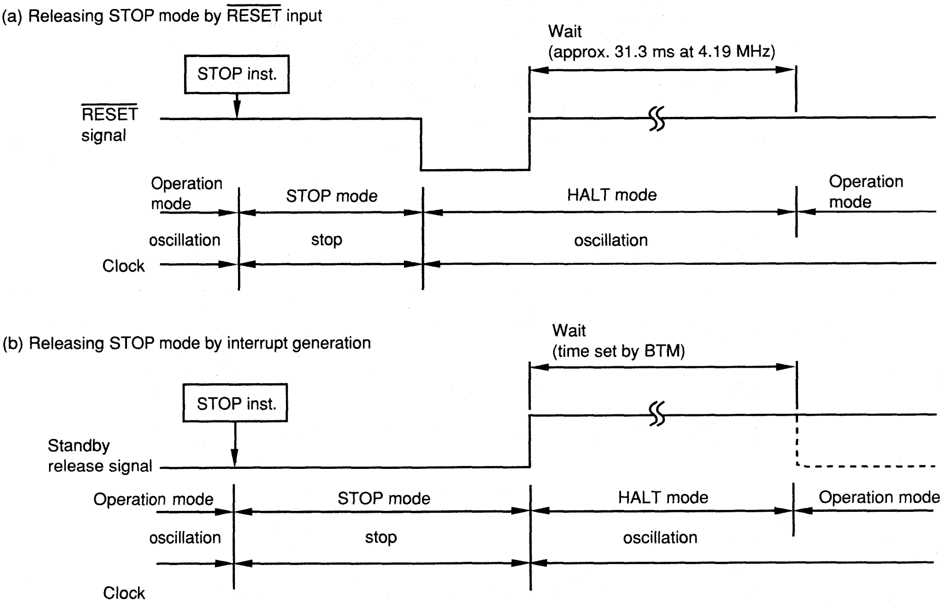
While operation stops during the standby mode, data is held in all registers and data memory such as general-purpose registers, flags, mode register, and output latches.

Caution: 1. When the STOP mode is set, X1 input is short-circuited to $V_{SS}$ (GND potential) internally to suppress crystal oscillator leakage. Therefore, so not use the STOP mode in a system using external clock as a main system slock.

2. Since an interrupt request signal is used to release the standby mode, if both interrupt request and enable flags are set for an interrupt source, the standby mode is immediately released. Thus, for the STOP mode, the HALT mode is entered immediately after execution of the STOP instruction, a wait is followed according to the setup time of the BTM register, then a return is made to the opertion mode.

### 7.2 Standby Mode Release

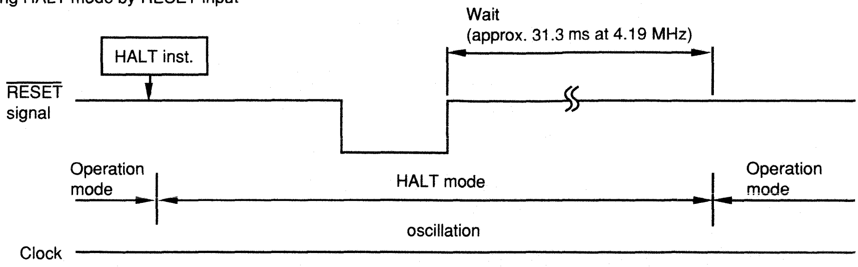
The standby mode (STOP or HALT) is released when an interrupt request signal (except INT0) enabled with an interrupt enable flag occurs or RESET is input. Fig. 6.2-1 shows the standby mode release operation.



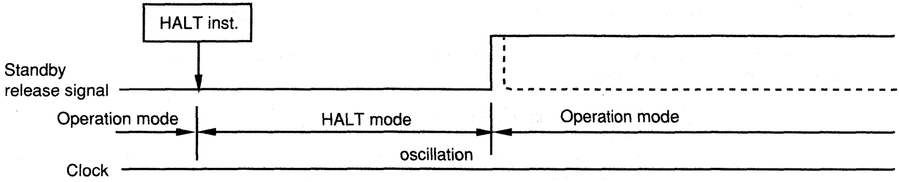
Note: Broken line shows a case when standby releasing interrupt request is acknowledged (IME = 1).

**Fig. 7.2-1 Standby Mode Release Operation**

(c) Releasing HALT mode by $\overline{\text{RESET}}$ input



(d) Releasing HALT node by interrupt generation



Note: Broken line shows a case when standby releasing interrupt request is acknowledged (IME = 1).

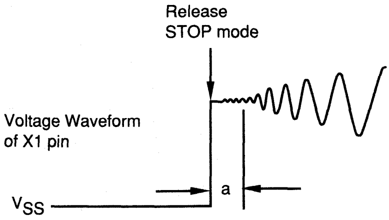
**Fig. 7.2-1 Standby Mode Release Operation (cont'd)**

If the STOP mode is released when an interrupt occurs, the wait time is determined by BTM setting. (See Table 7.2-1.)
The time until oscillation becomes stable varies according to the type of oscillator being used and the supply voltage when the STOP mode is released. Therefore, select the wait time according to the conditions of use and set BTM before setting the STOP mode.

**Table 7.2-1 Wait Time Selection by Using BTM**

| BTM3 | BTM2 | BTM1 | BTM0 | Wait time* ( ) indicates $f_{XX}$ = 4.19 MHz |
|---|---|---|---|---|
| – | 0 | 0 | 0 | APPROX. $2^{20}/f_{XX}$ (Approx. 250 ms) |
| – | 0 | 1 | 1 | APPROX. $2^{17}/f_{XX}$ (Approx. 31.3 ms) |
| – | 1 | 0 | 1 | APPROX. $2^{15}/f_{XX}$ (Approx. 7.82 ms) |
| – | 1 | 1 | 1 | APPROX. $2^{13}/f_{XX}$ (Approx. 1.95 ms) |
| Other than above | | | | Use Prohibited |

Note: The wait time when STOP mode is released does not include the time until the clock begins to oscillate after STOP mode is released ((a) in the figure below) regardless of whether STOP mode was released by $\overline{\text{RESET}}$ input or interrupt generation.



### 7.3 Operation After Standby Mode is Released

(1) If the standby mode is released when RESET is input, normal reset operation is performed.

(2) If the standby mode is released when an interrupt request occurs, the contents of the interrupt master enable flag (IME) determine whether or not a vectored interrupt is made when the CPU restarts instruction execution.

(a) When IME = 0

After the standby mode released, execution restarts at the NOP instruction next to the standby mode setting instruction. The interrupt request flags are held.

(b) When IME = 1

After the standby mode is released, two instructions following the standby mode setting instruction are executed before a vectored interrupt is executed.

However, if the standby mode is released by using INTW or INT2 (testable input), no vectored interrupt will occur; processing as in (a) above is performed.

### 7.4 Standby Mode Application

To use the standby mode, follow the procedure described below:

1) Detect a standby mode setting source such as interrupt input or port input for power off (it is effective to use INT4 for power off detection).
2) Handle input/output ports so that current consumption is minimized.
3) Specify interrupt to release the standby mode. (It is effective to use INT4. Clear the interrupt enables flags so as not to release the standby mode.)
4) Specify operation after the standby mode is released (set IME depending on whether or not interrupt service is made).
5) Specify the CPU clock after the standby mode is released. (To change the clock, wait for the required number of machine cycles before setting the standby mode.)
6) Select the wait time in release of standby mode.
7) Set the standby mode by using the STOP or HALT instruction.

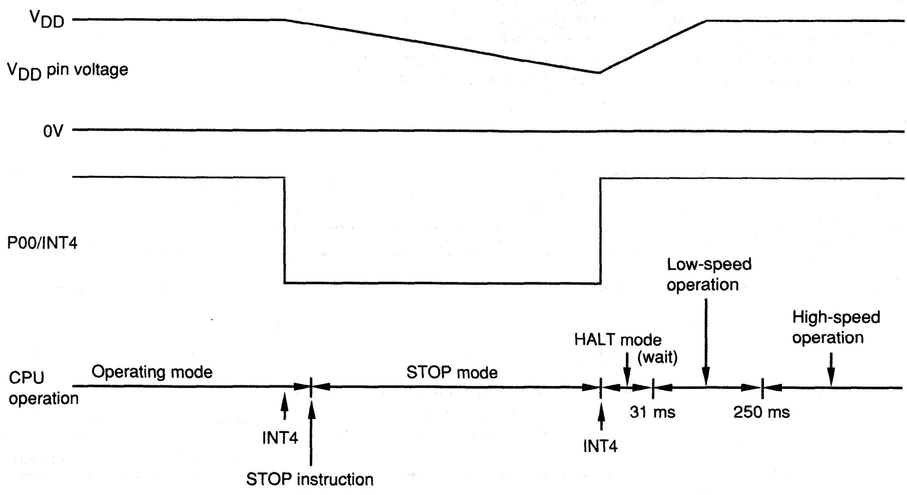
Use of the standby mode and system clock change function in combination enables the μPD75328 to operate at low current consumption and low voltage.

(1) Example of STOP mode application

Use the STOP mode under the following conditions

— Set the STOP mode when the INT4 falling edge is input and release it when the rising edge is input. (Do not use INTBT.)
— Place all input/output ports in high impedance.
— Use interrupts INT0 and INTT0 in the example program; however, do not use them to release the STOP mode.
— Enable interrupts after the STOP mode is released.
— After the STOP mode is released, start operation on the minimum speed CPU clock, and in 250 ms, change it to high-speed clock.
— Set the wait time for STOP mode release to about 31.3 ms.
— After the STOP mode is released, wait for 31.3 ms for the power supply to become stable. Check the P00/INT4 pin twice and remove chattering.

Timing Chart

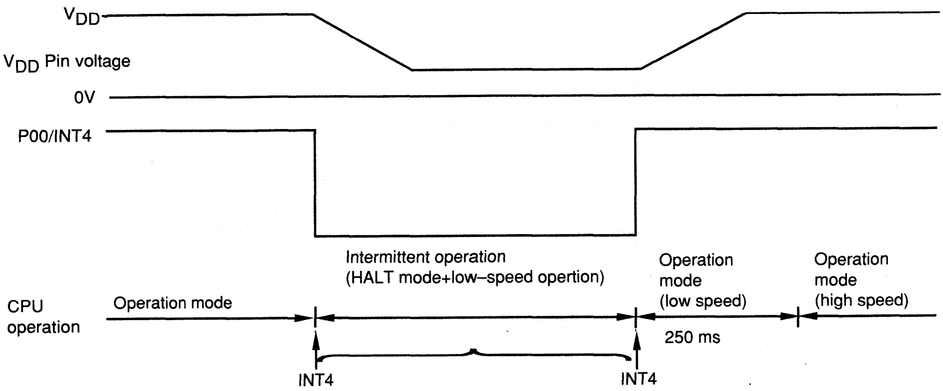


Programming example – INT4 service program with MBE = 0 –

```
VSUB4:      SKT     PORT0.0         ; P00 = 1?
            BR      PDOWN           ; Power down
            SET1    BTM.3           ; Power on
WAIT:       SKT     IRQBT           ; Waits for 31.3 ms
            BR      WAIT
            SKT     PORT0.0         ; Check chattering
            BR      PDOWN
            MOV     A, #0011B
            MOV     PCC, A          ; Set high-speed mode
            MOV     XA, #XXH        ; Set Port Mode register
            MOV     PMGm, XA        ;
            EI      IE0
            EI      IET0
            RETI
PDOWN:      MOV     A, #0           ; Minimum speed mode
            MOV     PCC, A
            MOV     XA, #00H
            MOV     LCDM, XA        ; LCD display off
            MOV     LCDC, A
            MOV     PMGA, XA        ; Place input/output ports in high impedance
            MOV     PMGB, XA
            MOV     PMGC, XA
            DI      IE0             ; Disables INT0 and INTT0
            DI      IET0
            MOV     A, #1011B
            MOV     BTM, A          ; Wait time = 31.3 ms
            STOP                    ; Sets STOP mode
            NOP
            RETI
```

(2) HALT mode application
Perform intermittent operation under the following conditions
— Change to the subsystem clock on the falling edge of INT4.
— Stop oscillation of the main system clock and set the HALT mode.
— Perform intermittent operation at 0.5sec. intervals during the standby mode.
— Again change to the main system clock on the rising edge of INT4.
— Do not use INTBT.

(Timing Chart)



Example (Initialization)

```
          MOV     A, #0011B
          MOV     PCC, A       ; High speed mode
          MOV     XA, #04
          MOV     WM, XA       ; Main system clock
          EI      IE4
          EI      IEW
          EI                   ; Enable interrupt
(Main routine)
          SKT     PORT0.0      ; Power OK?
          HALT                 ; Power down mode
          NOP                  ; Power OK?
          SKTCLR  IRQW         ; Is 0.5sec. flag set?
          BR      MAIN         ; NO
          CALL    WATCH        ; Watch subroutine
MAIN:       :
            :
            :
            :
(INT4 service routine)
INT4:     SKT     PORT0.0      ; Power OK?, MBE=0, RBE=0
          BR      PDOWN
          CLR1    SCC.3        ; Start oscillation of main system clock
          MOV     A, #8
          MOV     BTM, A
WAIT1:    SKT     IRQBT        ; Wait for 250 ms
          BR      WAIT1
          SKT     PORT0.0      ; Check chattering
          BR      PDOWN
          CLR1    SCC.0        ; Change to main system clock
          MOV     XA, #04H     ; Main system clock
```

```
        MOV     WM, XA
        RETI
PDOWN:  MOV     XA, #05H      ; Subsystem clock
        MOV     WM, XA
        MOV     XA, #00H
        MOV     LCDM, XA      ; LCD display off
        MOV     LCDC, A
        SET1    SCC.0         ; Change to subsystem clock
        MOV     A, #6
WAIT2:  INCS    A             ; Wait for 32 machine cycles
        BR      WAIT2
        SET1    SCC.3         ; Stop oscillation of main system clock
        RETI
```

Note: When the system clock is changed after power on from main system clock to subsystem clock, change the system clock after the subsystem clock is stabilized.

## 8. RESET FUNCTION

The µPD75328 is reset when $\overline{\text{RESET}}$ is input. The hardware devices are initialized as listed inTable 8.1-1. Fig. 8.1-1 shows the reset operation timing.

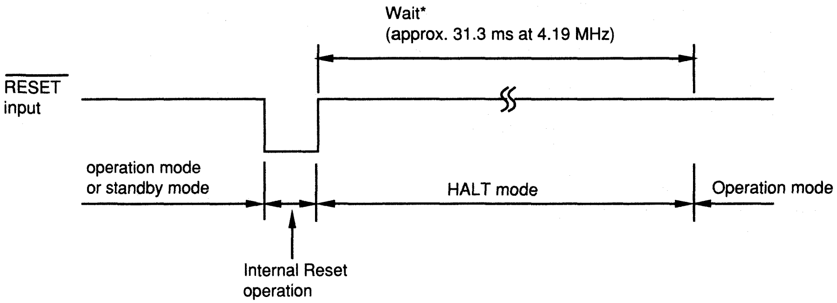

Wait*
(approx. 31.3 ms at 4.19 MHz)
$\overline{\text{RESET}}$ input
operation mode or standby mode
HALT mode
Operation mode
Internal Reset operation

**Fig. 8.1-1 Reset Operation by $\overline{\text{RESET}}$ is input**

**Table 8.1-1 State After Each Hardware Device is Reset**

| Hardware | | $\overline{\text{RESET}}$ input during standby mode | $\overline{\text{RESET}}$ input during operation |
|---|---|---|---|
| Program counter (PC | µPD75328 | The low-order five bits of program memory address 0000H are loaded into PC12-PC8. The contents of address 0001H are loaded into PC7-PC0. | |
| PSW | Carry flag (CY) | Held | Undefined |
| | Skip flag (SK0-SK2) | 0 | 0 |
| | Interrupt status flag (IST0) | 0 | 0 |
| | Bank enable flag | Bit 7 of program memory address 0000H is loaded into MBE. | |
| Stack pointer (SP) | | Undefined | Undefined |
| Data memory (RAM) | | Held (Note) | Undefined |
| General-purpose registers (X, A, H, L, D, E, B, C) | | Held | Undefined |
| Bank selection register (MBS) | | 0 | 0 |
| Basic interval timer | Counter (BT) | Undefined | Undefined |
| | Mode register (BTM) | 0 | 0 |
| Timer event counter | Counter (T0) | 0 | 0 |
| | Modulo register (TMOD0) | FFH | FFH |
| | Mode register (TM0) | 0 | 0 |
| | TOE0, TOUT F/F | 0, 0 | 0, 0 |
| Watch timer | Mode register (WM) | 0 | 0 |
| Serial interface | Shift register (SIO) | Held | Undefined |

**Table 8.1-1 State After Each Hardware Device is Reset (Cont'd)**

| Hardware | | $\overline{\text{RESET}}$ input during standby mode | $\overline{\text{RESET}}$ input during operation |
|---|---|---|---|
| Serial interface | Operation mode register (CSIM) | 0 | 0 |
| | SBI control register (SBIC) | 0 | 0 |
| | Slave address register (SVA) | Held | Undefined |
| Clock generator and clock output circuit | Processor clock control register (PCC) | 0 | 0 |
| | System clock control register (SCC) | 0 | 0 |
| | Clock output mode register (CLOM) | 0 | 0 |
| LCD controller | Display mode register (LCDM) | 0 | 0 |
| | Display control register (LCDC) | 0 | 0 |
| A/D converter | Mode register (ADM) | 04H (EOC = 1) | 04H (EOC = 1) |
| | SA register | 7FH | 7FH |
| Interrupt function | Interrupt request flags (IRQXXX) | Reset to 0 | Reset to 0 |
| | Interrupt enable flags (IEXXX) | 0 | 0 |
| | Interrupt master enable flag (IME) | 0 | 0 |
| | INT0, INT1, and INT2 mode registers (IM0, IM1, and IM2) | 0, 0, 0 | 0, 0, 0 |
| Digital ports | Output buffers | Off | Off |
| | Output latches | Cleared | Cleared |
| | Input/output mode registers (PMGA, B, C) | 0 | 0 |
| | Pull-up resistor specification register (POGA, B) | 0 | 0 |
| Bit sequential buffer (BSB0-BSB3) | | Held | Undefined |
| Pins condition | P00-P03, P10-P13, P20-P23, P30-P33, P60-P63, P70-P73, P80-P83 | Input | Input |
| | P40-P43, P50-P53, | — At incorporated pull-up resistor: ... High level<br>— At open drain: ... High impedance | |
| | S12-S23, COM0-COM3 | Note 2 | |
| | BIAS | — At incorporated split resistor: ... Low level<br>— At not incorporated split resistor: ... High impedance | |

Note: The data of data memory address 0F8H-0FDH is undefined by $\overline{\text{RESET}}$ input.

Note2: Each display output is selected as $V_{LCX}$ input source.

S12-31: $V_{LC1}$
COM0-COM2: $V_{LC2}$
COM3: $V_{LC0}$

However, each display output level is changed by each display output and $V_{LCX}$ external circuit.

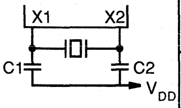
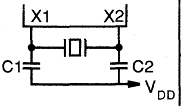
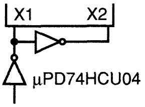
## 9. ELECTRICAL CHARACTERISTICS

**Absolute Maximum Ratings** (Ta = 25 °C)

| Item | Symbol | Conditions | | Ratings | Unit |
|---|---|---|---|---|---|
| Supply voltage | $V_{DD}$ | | | –0.3 to +7.0 | V |
| Input voltage | $V_{I1}$ | Other than ports 4 and 5 | | –0.3 to $V_{DD}$ +0.3 | V |
| | $V_{I2}$ (1) | Ports 4 and 5 | With internal pull-up resistor | –0.3 to $V_{DD}$ +0.3 | V |
| | | | Open drain | –0.3 to +11 | V |
| Output voltage | $V_O$ | | | –0.3 to $V_{DD}$ +0.3 | V |
| High level output current | $I_{OH}$ | Single pin | | –15 | mA |
| | | Total, all outputs | | –30 | |
| Low level output current | $I_{OL}$<br>Note | Single pin | Peak value | 30 | mA |
| | | | Effective value | 15 | |
| | | Ports 0, 2, 3, 5 and 8 total | Peak value | 100 | |
| | | | Effective value | 60 | |
| | | Ports 4, 6, and 7 total | Peak value | 100 | |
| | | | Effective value | 60 | |
| Operation Temperature | $T_{opt}$ | | | –40 to +85 | °C |
| Storage Temperature | $T_{stg}$ | | | –65 to +150 | °C |

Note: Use the following formula to calculate the effective value. (Effective value) = (Peak value) x $\sqrt{duty}$

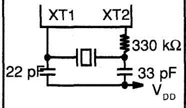
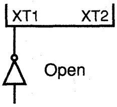
**Characteristics of Main System Clock Oscillator** (Ta = –40 to +85 °C, $V_{DD}$ = 2.7 to 6.0V)

| Oscillator | Recommended constants | Item | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Ceramic resonator |  | Note 1 Oscillation frequency ($f_{XX}$) | | 1.0 | | 5.0 | MHz |
| | | Note 2 Oscillation stabilization time | After $V_{DD}$ reaches the minimum value in the oscillator voltage range | | | 4 | ms |
| Crystal resonator |  | Note 1 Oscillation frequency ($f_{XX}$) | | 1.0 | 4.19 | 5.0 | MHz |
| | | Note 2 Oscillation stabilization time | $V_{DD}$ =4.5 to 6.0V | | | 10 | ms |
| | | | | | | 30 | ms |
| External clock |  | Note 1 X1 input frequency ($f_X$) | | 1.0 | | 5.0 | MHz |
| | | X1 input high/low level width ($t_{XH}$, $t_{XL}$) | | 100 | | 500 | ns |

Note 1: The oscillation frequency and X1 input frequency are indicated only to express the characteristics of the oscillator. Refer to the AC characteristics for instruction execution time.

Note 2: The oscillation stabilization time is the time required for the oscillator to stabillate after $V_{DD}$ is applied or after the STOP mode is released.

**Subsystem Clock Oscillator Characteristics** (Ta = –40 to +85 °C, $V_{DD}$ = 2.7 to 6.0V)

| Oscillator | Recommended constants | Item | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Crystal resonator |  | Oscillation frequency ($f_{XT}$) | | 32 | 32.768 | 35 | MHz |
| | | Oscillation stabilization time | $V_{DD}$ = 4.5 – 6.0V | | 1 | 2 | s |
| | | | | | | 10 | s |
| External clock |  | XT1 input frequency ($f_{XT}$) | | 32 | | 100 | kHz |
| | | XT1 input high/low level width ($t_{XTH}$, $t_{XTL}$) | | 5 | | 15 | µs |

**Recommended Resonator**

Main system clock: ceramic resonator

| Maunfacture | Product name | External capacitors (pF) | | Oscillator operating voltage range (V) | | Remarks |
|---|---|---|---|---|---|---|
| | | C1 | C2 | MIN. | MAX. | |
| Murata | CSA 2.00MG093 | 15 | 15 | 2.5 | 3.5 | Note |
| | CSB 1000D20 | 220 | 220 | 2.7 | 6.0 | |
| | CSA 2.00MG093<br>CSA 4.19MGU<br>CSA 4.91MGU | 30 | 30 | | | |
| | CST 2.00MG093<br>CST 4.19MGU<br>CST 4.91MGU | not required | not required | | | Internal C type |
| Kyocera | KBR–1000H | 100 | 100 | 2.7 | 6.0 | |
| | KBR–2.0MS | 47 | 47 | | | |
| | KBR–4.0MS<br>KBR–4.19MS<br>KBR–4.91MS | 33 | 33 | | | |

Note: When CSA 2.000G093 is used, VDD is 2.5 to 3.5V.

Main system clock: crystal resonator

| Maunfacture | Freg. Holder | External capacitors (pF) | | Oscillator operating voltage range (V) | | Remarks |
|---|---|---|---|---|---|---|
| | | C1 | C2 | MIN. | MAX. | |
| Kinseki | 1.00<br>2.00 HC–18/U<br>4.19 HC–49/U<br>4.91 HC–43/U | 22 | 22 | 2.7 | 6.0 | |

Subsystem clock: 32.768 KHz crystal resonator

| Maunfacture | Product name | External capacitors (pF) | | | Oscillator operating voltage range (V) | | Remarks |
|---|---|---|---|---|---|---|---|
| | | C1 (pF) | C2 (pF) | R (kΩ) | MIN. | MAX. | |
| Kinseki | P–3 | 22 | 22 | 330 | 2.7 | 6.0 | |

**Capacitance** (Ta = 25 °C, $V_{DD}$ = 0V)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Units |
|---|---|---|---|---|---|---|
| Input capacitance | $C_{IN}$ | f =1 MHz<br>For other than pins to be measured, 0V. | | | 15 | pF |
| Output capacitance | $C_{OUT}$ | | | | 15 | pF |
| Input/output capacitance | $C_{IO}$ | | | | 15 | pF |

**DC Characteristics** (Ta = –40 to +85 °C, $V_{DD}$ = 2.7 to 6.0V)

| Item | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| High level input voltage | $V_{IH1}$ | Ports 2 and 3 | | $0.7V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH2}$ | Ports 0, 1, 6, 7, and $\overline{\text{RESET}}$ pin | | $0.8V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH3}$ | Ports 4 to 5 | With built-in pullup resistor | $0.7V_{DD}$ | | $V_{DD}$ | V |
| | | | Open drain | $0.7V_{DD}$ | | 10 | V |
| | $V_{IH4}$ | X1, X2, XT1 | | $V_{DD}$–0.5 | | $V_{DD}$ | V |
| Low level input voltage | $V_{IL1}$ | Ports 2, 3, 4, 5 | | 0 | | $0.3V_{DD}$ | V |
| | $V_{IL2}$ | Ports 0, 1, 6, 7, and $\overline{\text{RESET}}$ pin | | 0 | | $0.2V_{DD}$ | V |
| | $V_{IL3}$ | X1, X2, XT1 | | 0 | | 0.4 | V |
| High level output voltage | $V_{OH1}$ | Ports 0, 2, 3, 6, 7, 8 BIAS | $V_{DD}$ = 4.5 to 6.0V<br>$I_{OH}$ = –1mA | $V_{DD}$–1.0 | | | V |
| | | | $I_{OH}$ = –100μA | $V_{DD}$–0.5 | | | V |
| | $V_{OH2}$ | BP0–7 (with two $I_{OH}$ outputs) | $V_{DD}$ = 4.5 to 6.0V<br>$I_{OH}$ = –100μA | $V_{DD}$–2.0 | | | V |
| | | | $I_{OH}$ = –30μA | $V_{DD}$–1.0 | | | V |
| Low level output voltage | $V_{OL1}$ | Port 0, 2 – 8 | Ports 3, 4, 5<br>$V_{DD}$ = 4.5 to 6.0V<br>$I_{OL}$ = 15mA | | 0.4 | 2.0 | V |
| | | | $V_{DD}$ = 4.5 to 6.0V<br>$I_{OL}$ = 1.6mA | | | 0.4 | V |
| | | | $I_{OL}$ = 400μA | | | 0.5 | V |
| | | SB0, 1 | Open drain Pull-up register ≥ 1kΩ | | | $0.2V_{DD}$ | V |
| | $V_{OL2}$ | BP0-7 (with two $I_{OL}$ outputs) | $V_{DD}$ = 4.5–6.0V<br>$I_{OL}$ = 100μA | | | 1.0 | V |
| | | | $I_{OL}$ = 50μA | | | 1.0 | V |
| High level input leakage current | $I_{LIH1}$ | $V_{IN}$ = $V_{DD}$ | Other than indicated below | | | 3 | μA |
| | $I_{LIH2}$ | | X1, X2, XT1 | | | 20 | μA |
| | $I_{LIH3}$ | $V_{IN}$ = 10V | Ports 4, 5 (with open drain) | | | 20 | μA |
| Low level input leakage current | $I_{LIL1}$ | $V_{IN}$ = 0V | Other than indicated below | | | –3 | μA |
| | $I_{LIL2}$ | | X1, X2, XT1 | | | –20 | μA |
| High level output leakage current | $I_{LOH1}$ | $V_{OUT}$ = $V_{DD}$ | Other than indicated below | | | 3 | μA |
| | $I_{LOH2}$ | $V_{OUT}$ = 10V | Ports 4 and 5 (with open drain) | | | 20 | μA |
| Low level output leakage current | $I_{LOL}$ | $V_{OUT}$ = 0V | | | | –3 | μA |
| Internal pull-up resistor | $R_{L1}$ | Ports 0,1,2,3,6,8 (Except P00) $V_{IN}$ = 0V | $V_{DD}$ = 5.0V ± 10% | 15 | 40 | 80 | kΩ |
| | | | $V_{DD}$ = 3.0V ± 10% | 30 | | 200 | kΩ |

**DC Characteristics** (Ta = –40 to +85 °C, $V_{DD}$ = 2.7 to 6.0V)

| Item | Symbol | Conditions | | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|---|
| Internal pull-up resistor | $R_{L2}$ | Ports 4 and 5 | $V_{DD}$ = 5.0V ± 10% | | 15 | 40 | 70 | kΩ |
| | | $V_{OUT}$=$V_{DD}$–2.0V | $V_{DD}$ = 3.0V ± 10% | | 10 | | 60 | kΩ |
| LCD drive voltage | $V_{LCD}$ | | | | 2.5 | | $V_{DD}$ | V |
| LCD split resistor | $R_{LCD}$ | | | | 60 | 100 | 150 | kΩ |
| LCD output voltage deviation (common) | $V_{ODC}$ | $I_O$ = ±5µA | $V_{LCD0}$ = $V_{LCD}$<br>$V_{LCD1}$ = $V_{LCD}$ x $^2/_3$<br>$V_{LCD2}$ = $V_{LCD}$ x $^1/_3$<br>Note 4 | | 0 | | ± 0.2V | V |
| LCD output voltage deviation (segment) | $V_{ODS}$ | $I_O$ = ±1µA | | | 0 | | ± 0.2V | V |
| Supply current (Note 1) | $I_{DD1}$ | (Note 5)<br>4.19 MHz crystal oscillation<br>C1=C2=22pF | $V_{DD}$ = 5V ± 10% (Note 2) | | | 2.5 | 8 | mA |
| | | | $V_{DD}$ = 3V ± 10% (Note 3) | | | 0.35 | 1.2 | mA |
| | $I_{DD2}$ | | HALT Mode | $V_{DD}$ = 5V ± 10% | | 500 | 1500 | µA |
| | | | | $V_{DD}$ = 3V ± 10% | | 150 | 450 | µA |
| | $I_{DD3}$ | 32kHz crystal resonator<br>(Note 6) | $V_{DD}$ = 3V ± 10% | | | 30 | 90 | µA |
| | $I_{DD4}$ | | HALT Mode | $V_{DD}$ = 3V ± 10% | | 5 | 15 | µA |
| | $I_{DD5}$ | XT1 = 0<br>STOP mode | $V_{DD}$ = 5V ± 10% | | | 0.5 | 20 | µA |
| | | | $V_{DD}$ = 3V ± 10% | | | 0.1 | 10 | µA |
| | | | | Ta = 25°C | | 0.1 | 5 | µA |

Note 1: The currents of the built-in pull-up resistor and the LCD step-down resistor are not included.
Note 2: When operated in the high-speed mode with the processor clock control register (PCC) set to 0011.
Note 3: When operated in the low-speed mode with the PCC set to 0000.
Note 4: 2.7V ≤ $V_{LCD}$ ≤ $V_{DD}$
Note 5: Includes the power consumption for the subsystem oscillation.
Note 6: When the system clock control registor (SCC) is set to 1001, mainsystem clock oscillation is stopped and operated by subsystem clock.

**A/D Converter characteristics** (Ta = –10 to +85 C, $V_{DD}$ = 3.5 to 6.0V, $AV_{SS}$ = $V_{SS}$ = 0V))

| Item | Symbol | Condition | MIN. | TYP. | MAX | Unit |
|---|---|---|---|---|---|---|
| Resolution | | | 8 | 8 | 8 | bit |
| Absolute accuracy (Note 1) | | 2.5 V ≤ $V_{AREF}$ ≤ $V_{DD}$ | | | ± 1.5 | LSB |
| Conversion time | $t_{CONV}$ | Note 2 | | | 168/$f_X$ | S |
| Sampling time | $t_{SAMP}$ | Note 3 | | | 44/$f_X$ | S |
| Analog input voltage | $V_{IAN}$ | | $AV_{SS}$ | | $V_{AREF}$ | V |
| Analog input impedance | $R_{AN}$ | | | 1000 | | MΩ |
| $V_{AREF}$ current | $I_{AREF}$ | | | 1.0 | 2.0 | mA |

Note 1: Absolute accuracy, which does not include the quantization error (± 1/2LSB)
Note 2: Time until EOC = 1 after conversion start instruction execution (40.1µs at 4.19 MHz ($f_X$)).
Note 3: Time until the completion of sampling after conversion start instruction execution (10.5µs at 4.19 MHz ($f_X$)).

**AC Characteristics** (Ta = –40 to +85 C, $V_{DD}$ = 2.7 to 6.0V)

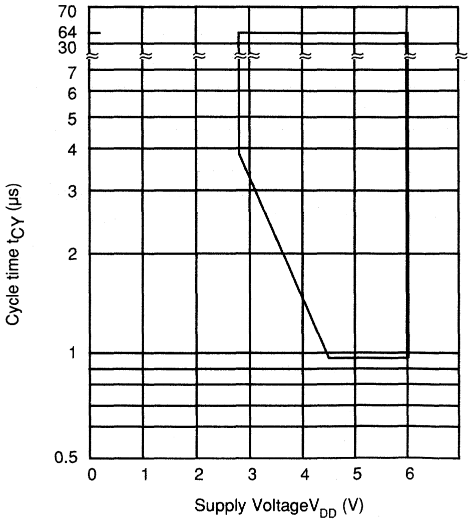
| Item | Symbol | Condition | | MIN. | TYP. | MAX | Unit |
|---|---|---|---|---|---|---|---|
| (Note 1) Cycle time (minimum instruction execution time) | $t_{CY}$ | Operation with main system clock | $V_{DD}$ = 4.5 –6.0V | 0.95 | | 64 | μs |
| | | | | 3.8 | | 64 | μs |
| | | Operation with sub-system clock | | 114 | 122 | 125 | μs |
| TI0 input frequency | $f_{TI}$ | $V_{DD}$ = 4.5 – 6.0V | | 0 | | 1 | MHz |
| | | | | 0 | | 275 | kHz |
| TI0 input high/low level width | $t_{TIH}$<br>$t_{TIL}$ | $V_{DD}$ = 4.5 – 6.0V | | 0.48 | | | μs |
| | | | | 1.8 | | | μs |
| Interrupt input high/low level width | $t_{INTH}$<br>$t_{INTL}$ | INT0 | | Note 2 | | | μs |
| | | INT1, 2, 4 | | 10 | | | μs |
| | | KR0-7 | | 10 | | | μs |
| $\overline{\text{RESET}}$ low level width | $t_{RSL}$ | | | 10 | | | μs |

Note 1: The cycle time (minimum instruction execution time) is determined by the oscillation frequency of the connected resonator, the system clock control register (SCC), and the processor clock control register (PCC). The figure below shows the $V_{DD}$ vs cycle time ($t_{CY}$) when operated with the main system clock.

Note 2: $2t_{CY}$ or $128/f_X$, depending on the setting of the interrupt mode register (IM0).

(with main system clock)

$t_{CY}$ vs. $V_{DD}$



**Serial Transfer Operation**

2-line/3line serial I/O mode ($\overline{SCK}$ ... Internal clock output)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{SCK}$ cycle time | $t_{KCY1}$ | $V_{DD}$ = 4.5 –6.0V | 1600 | | | ns |
| | | | 3800 | | | ns |
| $\overline{SCK}$ high/low level width | $t_{KL1}$ | $V_{DD}$ = 4.5 –6.0V | $\frac{t_{KCY1}}{2}$ -50 | | | ns |
| | $t_{KH1}$ | | $\frac{t_{KCY1}}{2}$ -150 | | | ns |
| SI set-up time (against $\overline{SCK}$ ↑ ) | $t_{SIK1}$ | | 150 | | | ns |
| SI hold time (against $\overline{SCK}$ ↑ ) | $t_{KSI1}$ | | 400 | | | ns |
| $\overline{SCK}$ ↓ → SO output delay time | $t_{KSO1}$ | $V_{DD}$ = 4.5 –6.0V | | | 250 | ns |
| | | | | | 1000 | ns |

2-line/3line serial I/O mode ($\overline{SCK}$ ... External clock input)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{SCK}$ cycle time | $t_{KCY2}$ | $V_{DD}$ = 4.5 –6.0V | 800 | | | ns |
| | | | 3200 | | | ns |
| $\overline{SCK}$ high/low level width | $t_{KL2}$ | $V_{DD}$ = 4.5 –6.0V | 400 | | | ns |
| | $t_{KH2}$ | | 1600 | | | ns |
| SI set-up time (against $\overline{SCK}$ ↑ ) | $t_{SIK2}$ | | 100 | | | ns |
| SI hold time (against $\overline{SCK}$ ↑ ) | $t_{KSI2}$ | | 400 | | | ns |
| $\overline{SCK}$ ↓ → SO output delay time | $t_{KSO2}$ | $V_{DD}$ = 4.5 –6.0V | | | 300 | ns |
| | | | | | 1000 | ns |

Note: The output delay time (for rising edge) of the serial line must be shorter than 600 ns. For example, if SB0 and SBI are pulled up with 5K ohms, the total capacitance of the serial bus line must be no greater than 120 pF.

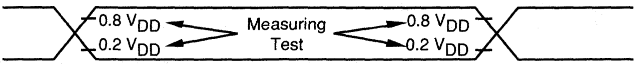
SBI mode ($\overline{\text{SCK}}$ … Internal clock output (master))

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY3}$ | $V_{DD}$ = 4.5 –6.0V | 1600 | | | ns |
| | | | 3800 | | | ns |
| $\overline{\text{SCK}}$ high/low level width | $t_{KL3}$ | $V_{DD}$ = 4.5 –6.0V | $\frac{t_{KCY1}}{2}$ -50 | | | ns |
| | $t_{KH3}$ | | $\frac{t_{KCY1}}{2}$-150 | | | ns |
| SB0,1 set-up time (against $\overline{\text{SCK}}$ ↑ ) | $t_{SIK3}$ | | 150 | | | ns |
| SB0,1 hold time (against $\overline{\text{SCK}}$ ↑ ) | $t_{KSI3}$ | | $\frac{t_{KCY1}}{2}$ | | | ns |
| $\overline{\text{SCK}}$↓ → SB0,1 output delay time | $t_{KSO3}$ | $V_{DD}$ = 4.5 –6.0V | 0 | | 250 | ns |
| | | | 0 | | 1000 | ns |
| $\overline{\text{SCK}}$ ↑→ SB0,1↓ | $t_{KSB}$ | | $t_{KCY}$ | | | ns |
| SB0,1↓→ $\overline{\text{SCK}}$ ↓ | $t_{SBK}$ | | $t_{KCY}$ | | | ns |
| SB0,1 low level width | $t_{SBL}$ | | $t_{KCY}$ | | | ns |
| SB0,1 high level width | $t_{SBH}$ | | $t_{KCY}$ | | | ns |

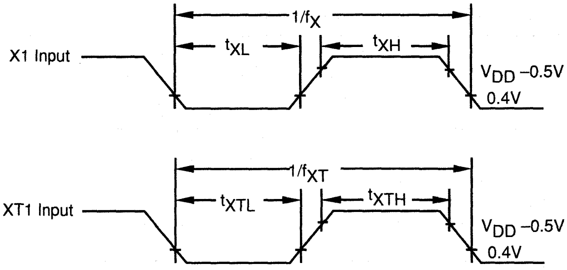
SBI mode ($\overline{\text{SCK}}$ … External clock input (slave))

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY4}$ | $V_{DD}$ = 4.5 –6.0V | 800 | | | ns |
| | | | 3200 | | | ns |
| $\overline{\text{SCK}}$ high/low level width | $t_{KL4}$ | $V_{DD}$ = 4.5 –6.0V | 400 | | | ns |
| | $t_{KH4}$ | | 1600 | | | ns |
| SB0,1 set-up time (against $\overline{\text{SCK}}$ ↑ ) | $t_{SIK4}$ | | 100 | | | ns |
| SB0,1 hold time (against $\overline{\text{SCK}}$ ↑ ) | $t_{KSI4}$ | | $\frac{t_{KCY1}}{2}$ | | | ns |
| $\overline{\text{SCK}}$↓ → SB0,1 output delay time | $t_{KSO4}$ | $V_{DD}$ = 4.5 –6.0V | 0 | | 300 | ns |
| | | | 0 | | 1000 | ns |
| $\overline{\text{SCK}}$ ↑→ SB0,1↓ | $t_{KSB}$ | | $t_{KCY}$ | | | ns |
| SB0,1↓→ $\overline{\text{SCK}}$ ↓ | $t_{SBK}$ | | $t_{KCY}$ | | | ns |
| SB0,1 low level width | $t_{SBL}$ | | $t_{KCY}$ | | | ns |
| SB0,1 high level width | $t_{SBH}$ | | $t_{KCY}$ | | | ns |

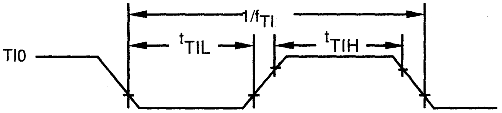
AC Timing Measurement Points (Except X1 and XT1 inputs)



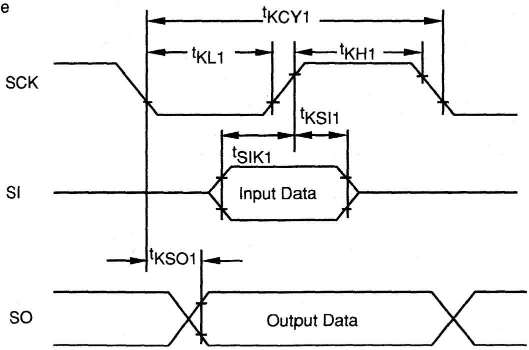
Clock Timing



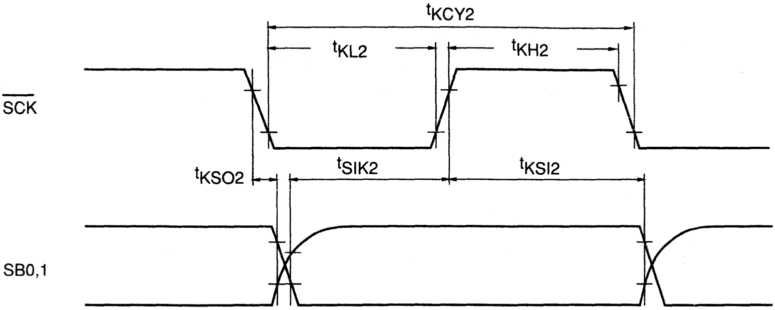
TIO Timing



Serial Transfer Timing
3-line serial I/O mode

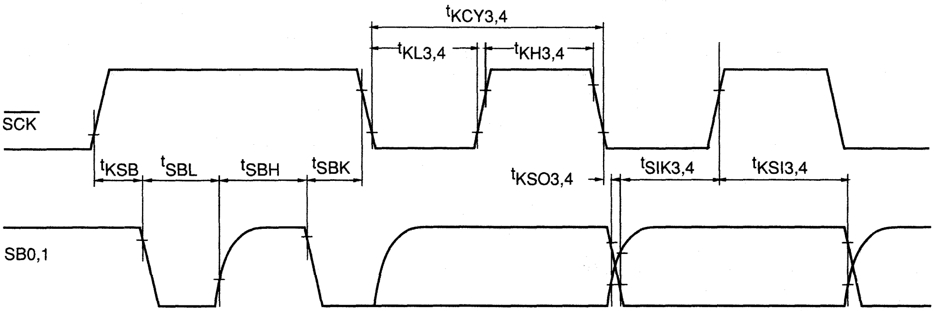


2-line serial I/O mode

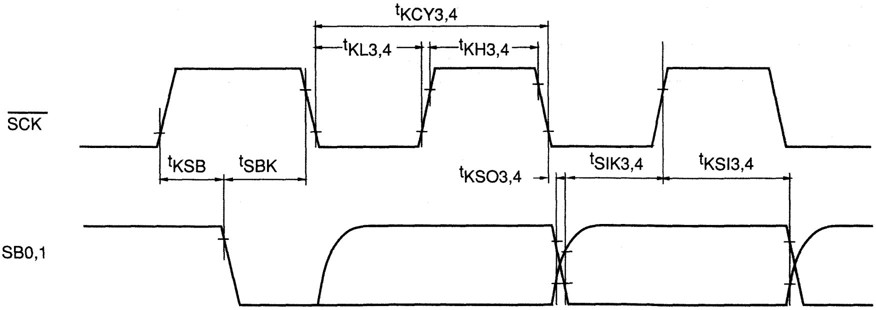


**Serial Transfer Timing**

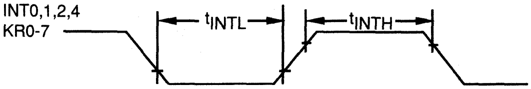
Bus release signal transfer:



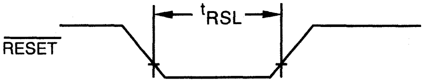
Command signal transfer:



Interrupt input timing



$\overline{\text{RESET}}$ input timing



**Data Memory STOP Mode Low Voltage Data Retention Characteristic** (Ta = –40 to +85 °C)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Data retention voltage | $V_{DDDR}$ | | 2.0 | | 6.0 | V |
| Data retention current (Note1) | $I_{DDDR}$ | $V_{DDR}$ = 2.0V | | 0.1 | 10 | μA |
| Release signal set time | $t_{SREL}$ | | 0 | | | μs |
| Oscillation stabilization time (Note 2) | $t_{WAIT}$ | Release by $\overline{\text{RESET}}$ input | | $2^{17}/f_X$ | | ms |
| | | Release by interrupt request | | Note 3 | | ms |

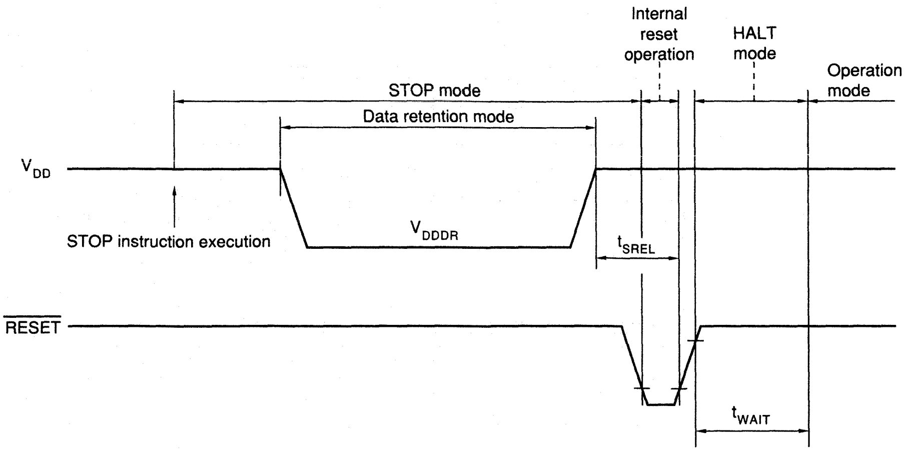
Note 1: Current in the internal pull-up resistors is not included.

Note 2: The oscillation stabilization time is the time required before beginning CPU operation in order to prevent unstable CPU operation when oscillation is initiated.

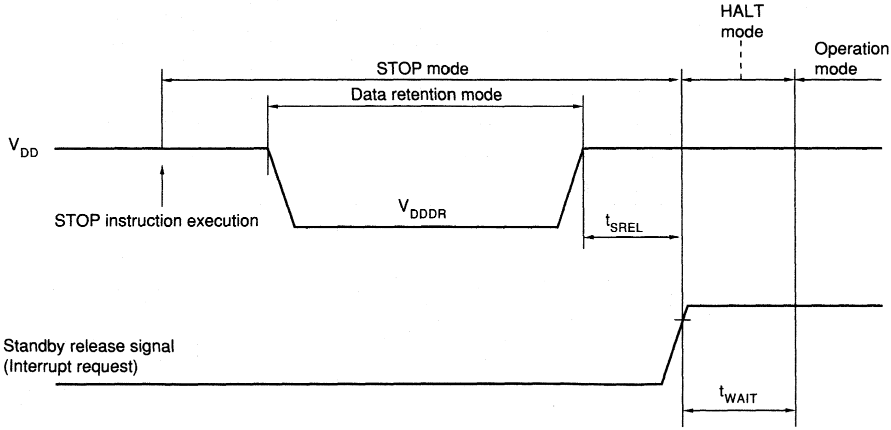
Note 3: Depends on the setting of the basic interval timer mode register (BTM) (refer to the table below).

| BTM3 | BTM2 | BTM1 | BTM0 | WAIT time ( ) indicates $f_X$ =4.19MHz |
|---|---|---|---|---|
| — | 0 | 0 | 0 | $2^{20}/f_X$ (Approximately 250 ms) |
| — | 0 | 1 | 1 | $2^{17}/f_X$ (Approximately 31.3 ms) |
| — | 1 | 0 | 1 | $2^{15}/f_X$ (Approximately 7.82 ms) |
| — | 1 | 1 | 1 | $2^{13}/f_X$ (Approximately 1.95 ms) |

Data Retention Timing (when STOP mode is released by $\overline{\text{RESET}}$ input)



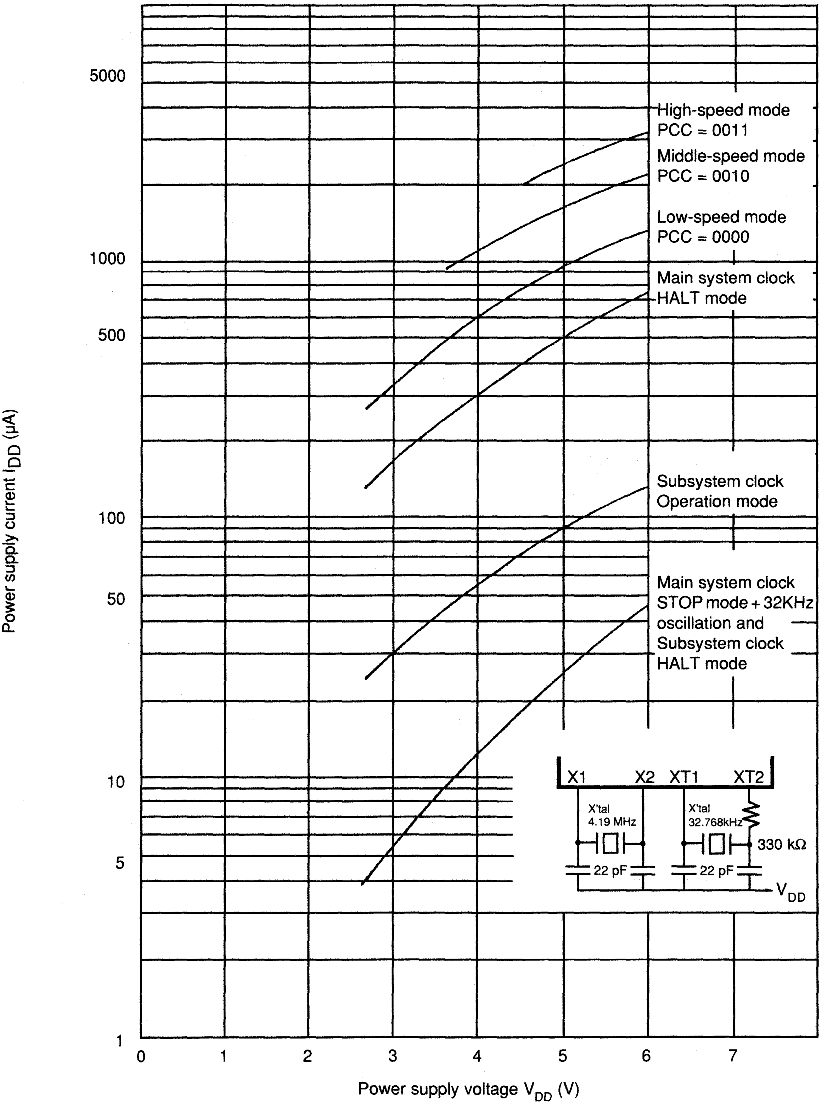
Data Retention Timing
(Standby release signal: STOP mode release by interrupt signal)



**Characteristic Curves**

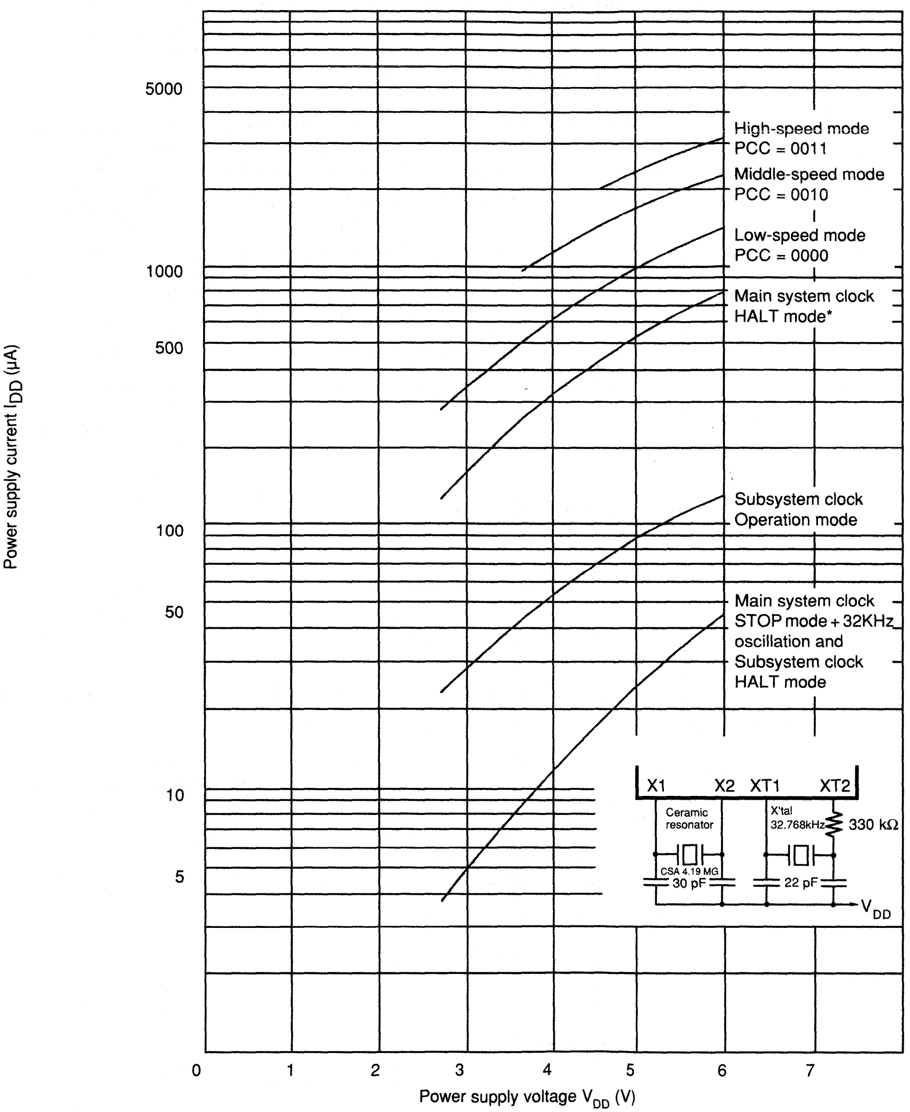
$I_{DD}$ vs $V_{DD}$ (Crystal oscillation)

(Ta = 25 °C)



$I_{DD}$ vs $V_{DD}$ (Ceramic oscillation)

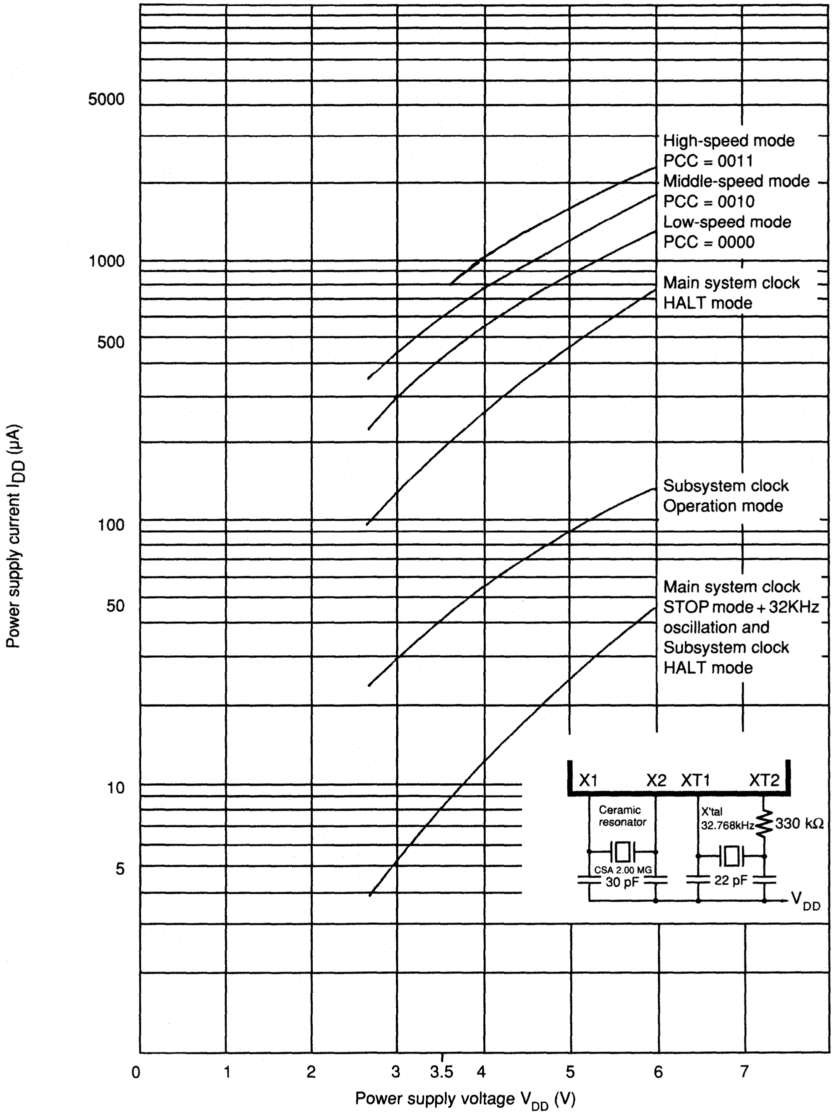
(Ta = 25 °C)



*The Current values are increased by approximately 10% compared to when operating with a crystal resonator.

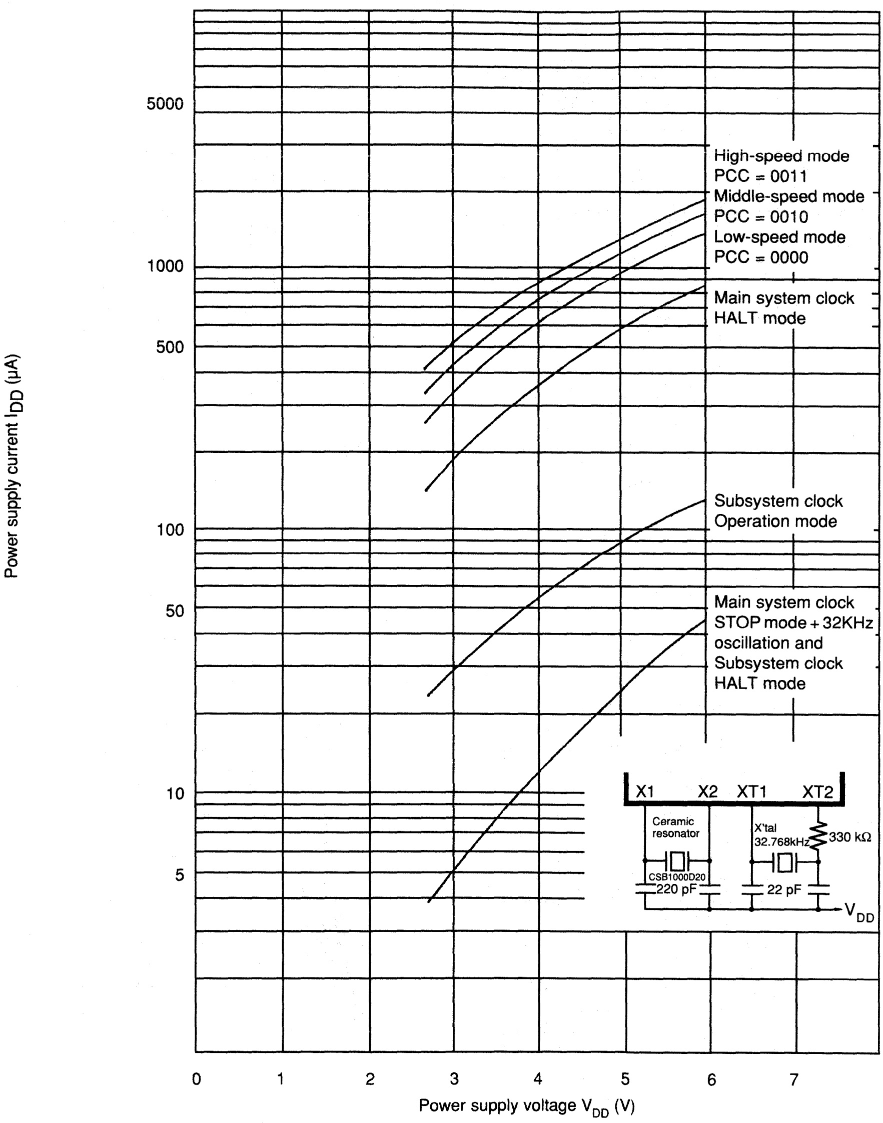
$I_{DD}$ vs $V_{DD}$ (Ceramic oscillation)

(Ta = 25 °C)

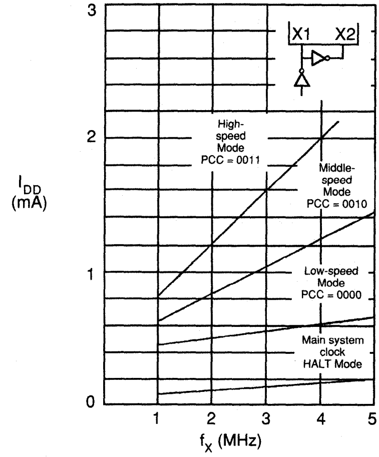


$I_{DD}$ vs $V_{DD}$ (Ceramic oscillation)

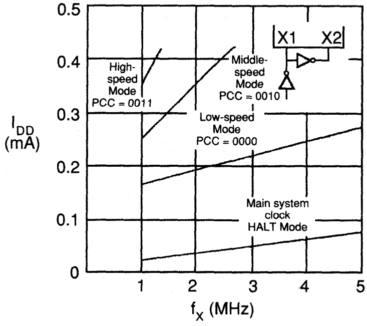
(Ta = 25 °C)



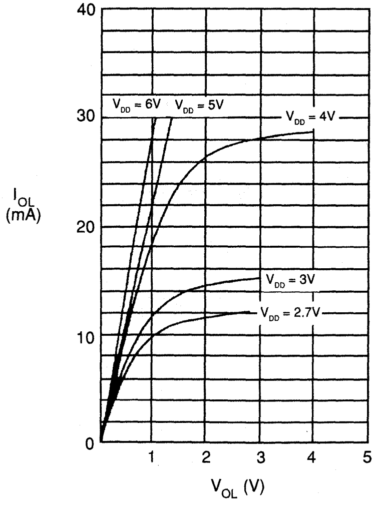
$I_{DD}$ vs $V_{DD}$ ($V_{DD}$ = 5V, Ta = 25 °C)



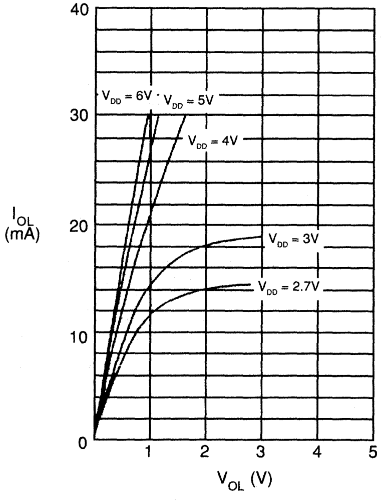
$I_{DD}$ vs $f_X$ ($V_{DD}$ = 3V, Ta = 25 °C)



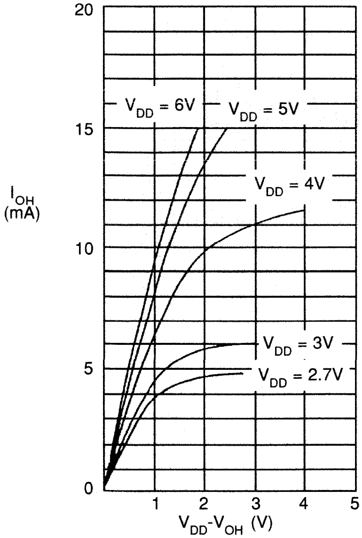
$V_{OL}$ vs $I_{OL}$ (Ports 0, 2, 6, 7, 8, Ta = 25 °C)



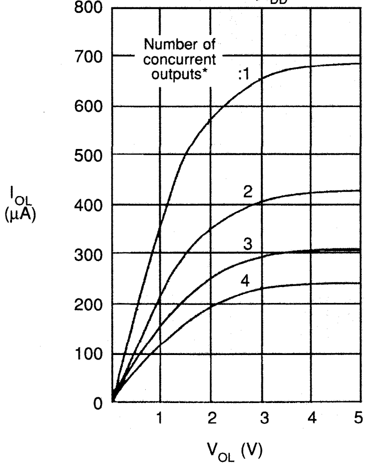
$V_{OL}$ vs $I_{OL}$ (Ports 3, 4, 5, Ta = 25 °C)



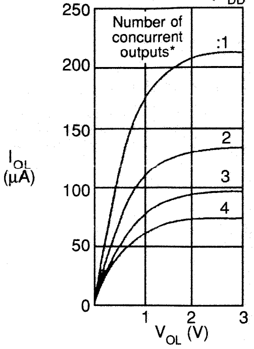
$V_{OH}$ vs $I_{OH}$ (Ta = 25 °C)



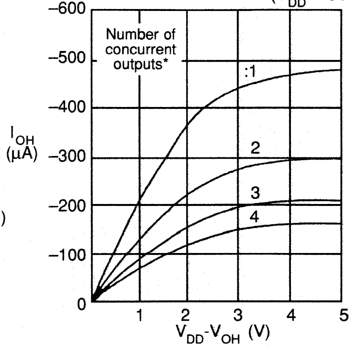
$V_{OL}$ vs $I_{OL}$ (BP0-3, BP4-7) ($V_{DD}$ = 5V Ta = 25 °C)



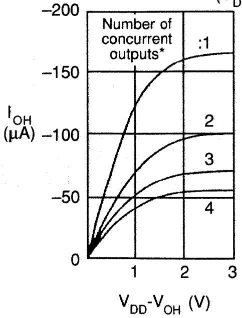
$V_{OL}$ vs $I_{OL}$ (BP0-3, BP4-7) ($V_{DD}$ = 3V Ta = 25 °C)



$V_{OH}$ vs $I_{OH}$ (BP0-3, BP4-7) ($V_{DD}$ = 5V Ta = 25 °C)



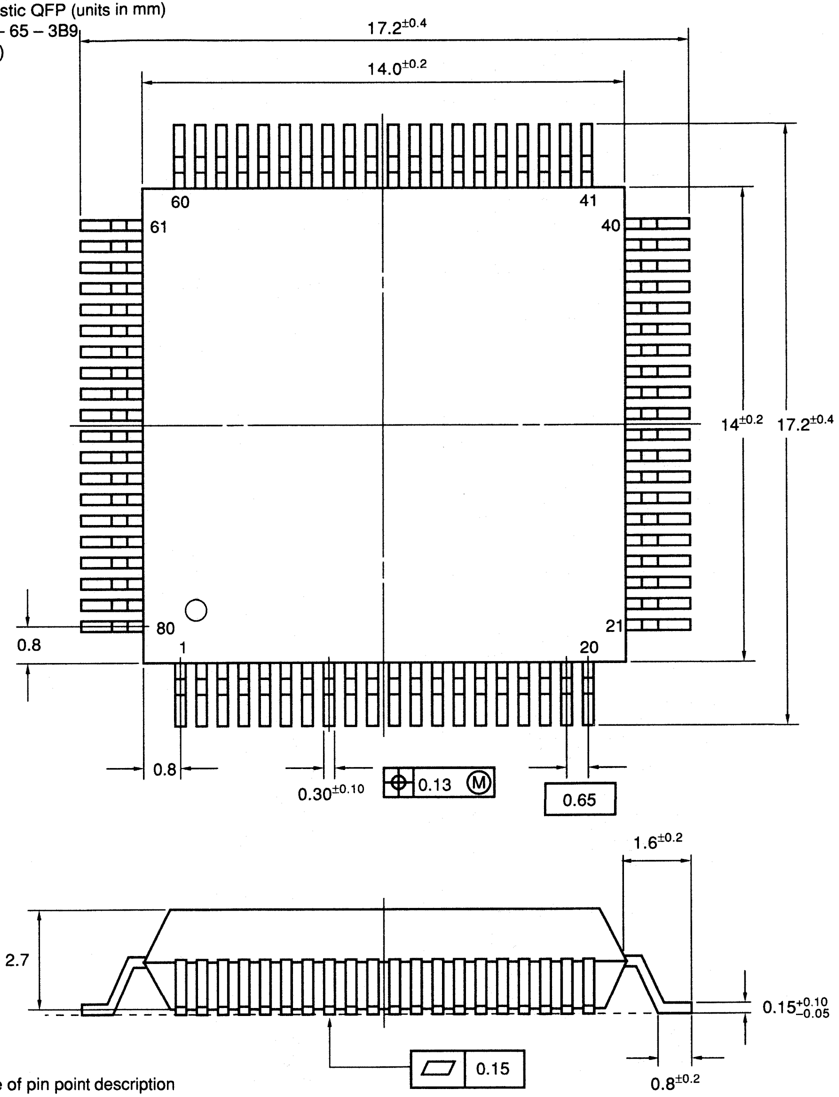
$V_{OH}$ vs $I_{OH}$ (BP0-3, BP4-7) ($V_{DD}$ = 3V Ta = 25 °C)



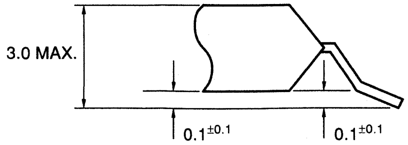
* Of pins BP0 to BP3 or BP4 to BP7, number of pins outputting the same level.

### 9.1 Package Information

80-pin plastic QFP (units in mm)
S 80 GC – 65 – 3B9
(Top view)



Shape of pin point description



## 10. μPD75P328GC (OTPROM) 4-BIT MICROCOMPUTERS.

### 10.1 Overview

The μPD75P328 is a single-chip 4-bit microcomputer which is supplied as a plastic otprom (one time programmable) or ceramic uvprom with window. The single-chip includes, timer/event counters, serial interface and vector interrupt. These functions are incorporated along with CPU, PROM, RAM and I/O ports.

The μPD75P328 is pin-compatible with the μPD75P328GC and equivalent to the device in function.

The μPD75P328 will be use for emulation of μPD75328. It should not be used for final EMI and latch up evaluation.

Note:

| Memory type | μPD75P328GC-3B9 |
|---|---|
| ROM | 8064 x 8 bit |
| RAM | 512 x 4 bit |

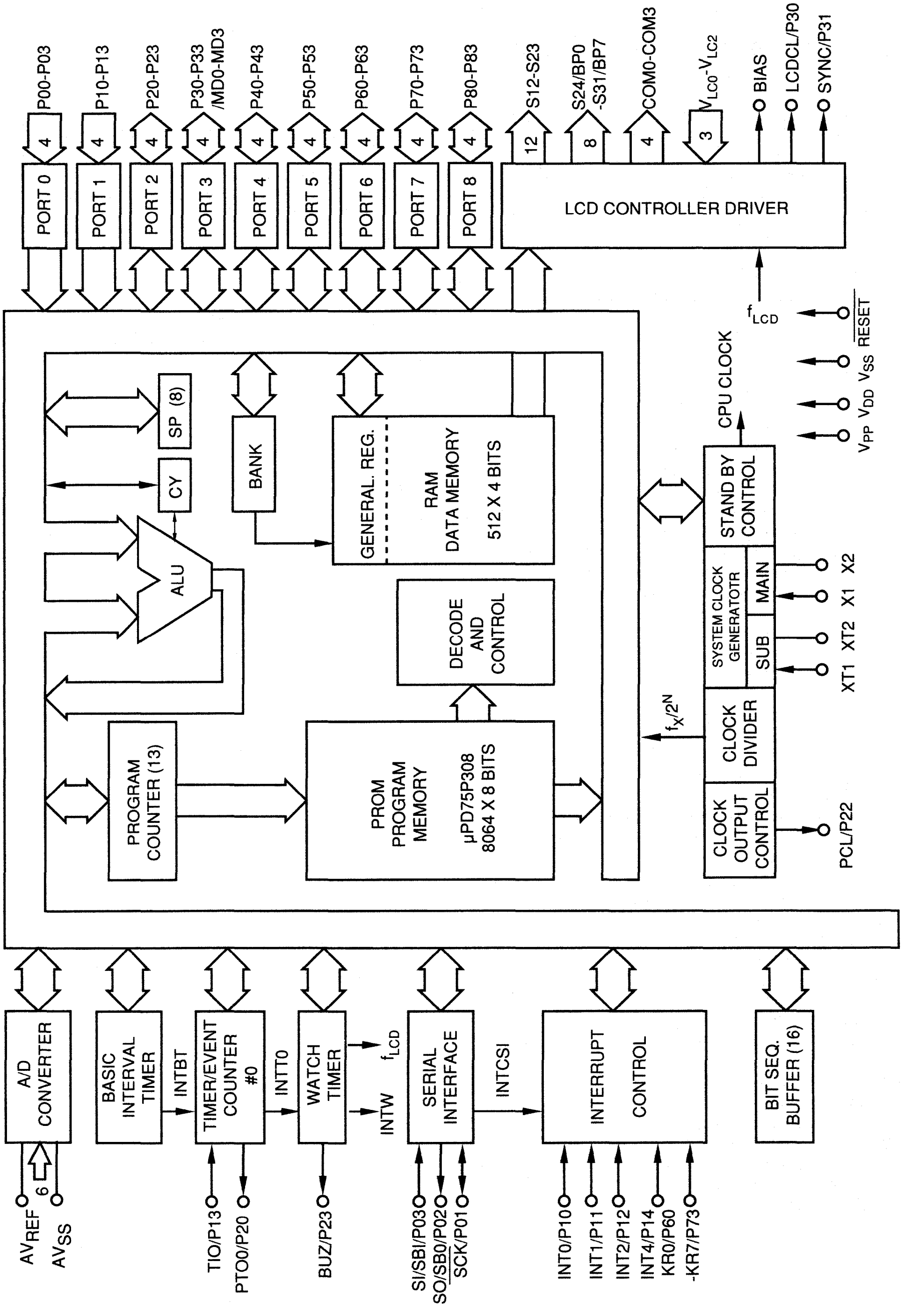
### 10.2 Features

- 42 types of instructions.
  Numerous bit manipulation instructions.
  8-bit transfer instructions.
  1-byte relative branch instructions.
  GETI instruction that realizes arbitrary 2-byte instructions in 1-byte units.
- Programmable instruction cycle: 0.95μs, 1.91μs, 15.3μs / 4.19MHz (main system oscillator), 122μs / 32kHz (sub system oscillator) at 5V
- Size of data memory (RAM): 512 x 4 bit (memory bank configuration)
- Memory-mapped I/O (memory bank configuration)
- Memory for bit manipulation (bit-sequential buffer: 16 bits)
- General-purpose register: 4 bits x 8 (one register bank)
- Accumulators:

| | | |
|---|---|---|
| Accumulators: | Bit accumulator | (CY) |
| | 4-bit accumulator | (A) |
| | 8-bit accumulator | (XA) |

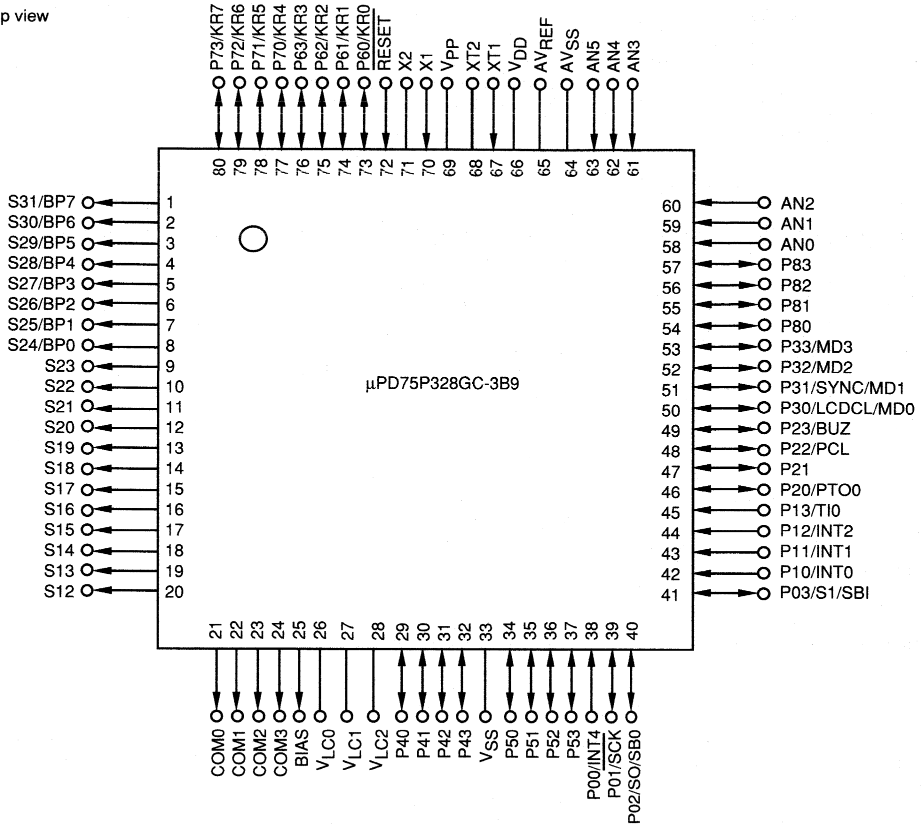
- LCD controller/driver
  LCD drive segment outputs: 20
  LCD drive common outputs: 4
  LCD drive mode
  — static
  — 1/2 bias – 1/2 duty, 1/3 duty
  — 1/3 bias – 1/3 duty, 1/4 duty
- External expansion port for LCD controller/driver, serial connection.
- Internal 6 channel 8-bit precision A/D converter.
- 36 I/O lines
  CMOS In/Out Pins
  Middle-high voltage N-ch open drain In/Out Ports: 2
  LED direct drive port: 3
- Vector interrupt function capable of multiple interrupts, 3 external vector interrupts
  1 external test input
  3 internal vector interrupts
  1 internal test interrupt
- Basic interval timer
- One 8-bit timer/event counter
- Watch timer (0.5 sec test flag)
- 8 bit serial interface:
  — Conventional 75, 75X compatible mode
  — Serial bus interface (SBI) mode
- Internal clock oscillator with crystal/ceramic resonator
- Subsystem and main system oscillators, run CPU on 32kHz subsystem.
- Standby operation (STOP/HALT)
- CMOS low power consumption
- μPD75P328GC: 80 PIN plastic flat pack one time programmable OTPROM

### 10.3 Block Diagram



### 10.4 Pin Configuration

Top view



## 10.5 Pin Functions

### 10.5.1 Port Pins

| Pin | Input/Output | Serves as Pin for | Functions | 8-Bit I/O | At Reset | I/O Circuit Type (Note 1) |
|---|---|---|---|---|---|---|
| P00 | Input | INT4 | 4-bit input port (PORT0). Incorporation of pull-up resistors in 3-bit units for P01 to P03 specifiable by software. | X | Input | Ⓑ |
| P01 | Input/Output | $\overline{\text{SCK}}$ | | | | Ⓕ-A |
| P02 | Input/Output | SO/SB0 | | | | Ⓕ-B |
| P03 | Input/Output | SI/SBI | | | | Ⓜ-C |
| P10 | Input | INT0 | Noise elimination function available<br>4-bit input port (PORT1). Incorporation of pull-up resistors in 4-bit units specifiable by software. | X | Input | Ⓑ-C |
| P11 | | INT1 | | | | |
| P12 | | INT2 | | | | |
| P13 | | TI0 | | | | |
| P20 | Input/Output | PTO0 | 4-bit input/output port (PORT2). Incorporation of pull-up resistors in 4-bit units specifiable by software. | X | Input | E-B |
| P21 | | — | | | | |
| P22 | | PCL | | | | |
| P23 | | BUZ | | | | |
| P30 (Note 2) | Input /Output | LCDCL<br>MD0 | Programmable 4-bit input/output port Input/output setting is possible in bit units. In corporation of pull up resistors in 4 bit units specifiable by software. | X | Input | E-B |
| P31 (Note 2) | | SYNC<br>MD1 | | | | |
| P32 (Note 2) | | MD2 | | | | |
| P33 (Note 2) | | MD3 | | | | |
| P40-P43 (Note 2) | Input/Output | — | N-channel open drain 4-bit input/output port (PORT4). Data input/output pin (lower 4-bits) for use during program memory (PROM) write/ verify operations. | O | High Impedance | M-B |
| P50-P53 (Note 2) | Input/Output | — | N-channel open drain 4-bit input/output port (PORT5). Data input/output pin (upper 4-bits) for use during program memory (PROM) write/ verify operations. | | High Impedance | M-B |
| P60 | Input/Output | KR0 | Programmable 4-bit input/output port (PORT 6). This port can be specified for input/output in bit units. Incorporates software-specifiable pull-up resistors in 4-bit units. | O | Input | Ⓕ-A |
| P61 | | KR1 | | | | |
| P62 | | KR2 | | | | |
| P63 | | KR3 | | | | |
| P70 | Input/Output | KR4 | 4-bit input/output port (PORT7) Incorporation of pull-up resistors in 4-bit units specifiable by software. | | Input | Ⓕ-A |
| P71 | | KR5 | | | | |
| P72 | | KR6 | | | | |
| P73 | | KR7 | | | | |
| P80-P83 | Input/Output | - | 4-bit input/output port (PORT8). Internal pull-up resistor can be specified for P80-P83 in 4-bit units by using software. | X | Input | E-B |

10.5.1 Port Pins (cont'd)

| Pin | Input/Output | Serves as Pin for | Functions | 8-Bit I/O | At Reset | I/O Circuit Type (Note 1) |
|---|---|---|---|---|---|---|
| BP0 | Output | S24 | 1-bit output port (BIT PORT).<br>Also serves as the segment output pins. | X | (Note 3) | G-A |
| BP1 |  | S25 |  |  |  |  |
| BP2 |  | S26 |  |  |  |  |
| BP3 |  | S27 |  |  |  |  |
| BP4 | Output | S28 |  |  |  |  |
| BP5 |  | S29 |  |  |  |  |
| BP6 |  | S30 |  |  |  |  |
| BP7 |  | S31 |  |  |  |  |

Note 1: Circles indicate Schmitt trigger inputs.
Note 2: LED direct drive is possible.
Note 3: BP0-7 select $V_{LC1}$ as the input source. The output level changed by the external circuit of BP0-7 and $V_{LC1}$.

10.5.2 Non-Port Pins

| Pin | Input/Output | Serves as Pin for | Functions | At Reset | I/O Circuit Type (Note 1) |
|---|---|---|---|---|---|
| TI0 | Input | P13 | External event pulse input pin for timer/event counter. |  | Ⓑ-C |
| PTO0 | Input/Output | P20 | Timer/event counter output pin. | Input | E-B |
| PCL | Input/Output | P22 | Clock output pin. | Input | E-B |
| BUZ | Input/Output | P23 | Fixed-frequency output pin (for buzzer or system clock trimming). | Input | E-B |
| $\overline{\text{SCK}}$ | Input/Output | P01 | Serial clock input/output pin. | Input | Ⓕ-A |
| SO/SB0 | Input/Output | P02 | Serial data output pin. Serial bus input/output pin. | Input | Ⓕ-B |
| SI/SBI | Input/Output | P03 | Serial data input pin. Serial bus input/output pin. | Input | Ⓜ-C |
| INT4 | Input | P00 | Edge detector vector interrupt input pin (either rising or falling edge detection). |  | Ⓑ |
| INT0 | Input | P10 | Edge detector interrupt input pin (detected edge selectable). |  | Ⓑ-C |
| INT1 |  | P11 |  |  |  |
| INT2 | Input | P12 | Edge detection testable input pin (rising edge detection). |  | Ⓑ-C |
| KR0-KR3 | Input/Output | P60-P63 | Testable input/output pin (for falling edge detection). | Input | Ⓕ-A |
| KR4-KR7 | Input/Output | P70-P73 | Testable input/output pin (for falling edge detection). | Input | Ⓕ-A |
| S12-S23 | Output |  | Segment signal output pin. | (Note 3) | G -A |
| S24-S31 | Output | BP0-7 | Segment signal output pin. | (Note 3) | G -A |
| COM0 -COM3 | Output |  | Common signal output pin. | (Note 3) | G -B |
| $V_{LC0}$-$V_{LC2}$ |  |  | LCD drive power supply. |  |  |

10.5.2 Non-Port Pins

| Pin | Input/Output | Serves as Pin for | Functions | At Reset | I/O Circuit Type (Note 1) |
|---|---|---|---|---|---|
| BIAS | | | Output pin for divided resistor cut externally installed. | Hi-Z | |
| LCDCL (Note 2) | Input/Output | P30 | Clock output pin for activating the externally extended driver. | Input | E-B |
| SYNC (Note 2) | Input/Output | P31 | Clock output pin for synchronizing the externally extended driver. | Input | E-B |
| AN0-AN5 | Input | – | 6-bit analog input pins to A/D converter. | | Y |
| $AV_{REF}$ | Input | – | A/D converter reference voltage input pins. | | Z |
| $AV_{SS}$ | | | A/D converter reference GND input pins. | | |
| X1, X2 | Input | | Pin for connection of crystal/ceramics for main system clock generation. External clocks are input to X1, and their negative phase components are input to X2. | | |
| XT1 | Input | | Pin for connection of crystals for sub-system clock generation. External clocks are input to XT1, and XT2 is disconnected. XT1 can also be used as a 1-bit input (test) pin. | | |
| XT2 | | | | | |
| $\overline{\text{RESET}}$ | Input | | System reset input pin (low level active). | | Ⓑ |
| MD0-MD3 | Input/Output | P30-P33 | Mode selector pin for program memory (EPROM) write/verify operations. | Input | E-B |
| $V_{PP}$ | | | Program voltage application pin for program memory (PROM) write/verify operations. $V_{PP}$ is normally connected to $V_{DD}$. Connect to +12.5V for PROM write/verify operations. | | |
| $V_{DD}$ | | | Positive power supply pin. | | |
| $V_{SS}$ | | | GND potential pin. | | |

Note 1: Circles indicate Schmitt trigger inputs.

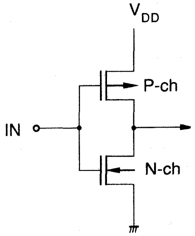
Note 2: Pins for future use in system extention. Pins LCDCL and SYNC currently serve only as P30 and P31, respectively.

Note 3: For each display output, the following $V_{LCX}$ are selected as input sources:
S12-S31: $V_{LC1}$, COM0-COM2: $V_{LC2}$, COM3: $V_{LC0}$
However, the display output level varies depending on the display output and $V_{LCX}$ external circuit.
Internal LCD split resistors are not included!

Pin I/O configurations

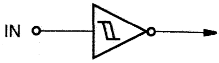
Following figures show the internal circuit configurations at the I/O ports.

(1) Type A (part of Type E–B)



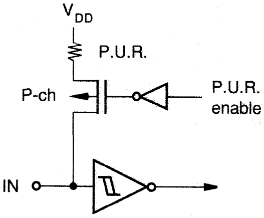
This is a CMOS standard input buffer.

(2) Type B



This is a Schmitt trigger input with hysteresis characteristics.

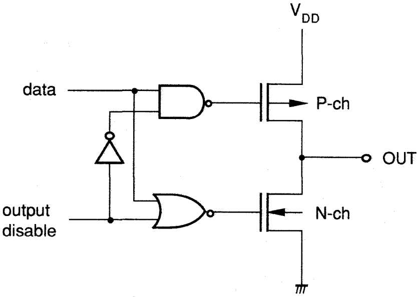
(3) Type B–C



P.U.R. : Pull-Up Resistor

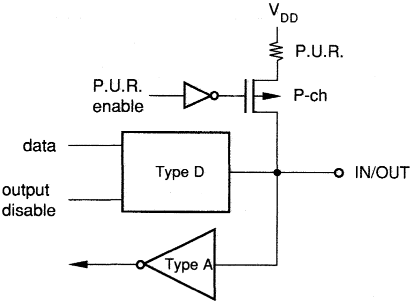
This is a Schmitt trigger input with hysteresis characteristics.

(4) Type D (Part of Type E–B, F–A)



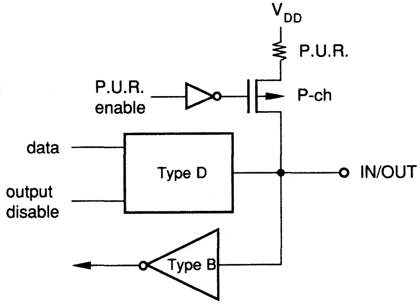
This is a push-pull output that can be set to high impedance (with both P-ch and N-ch off).

(5) Type E–B



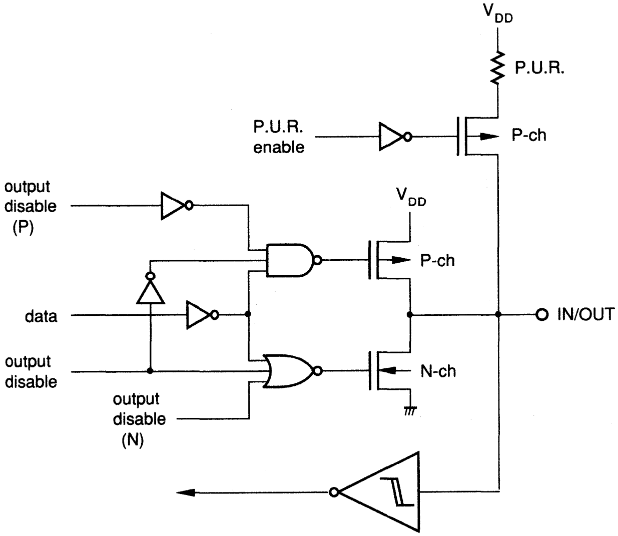
P.U.R. : Pull-Up Resistor

(6) Type F–A



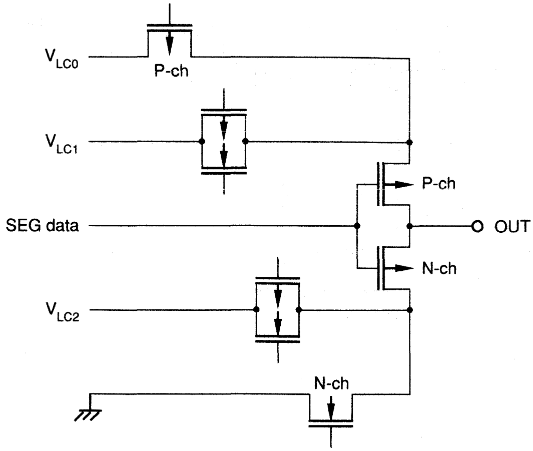
P.U.R. : Pull-Up Resistor

(7) Type F–B

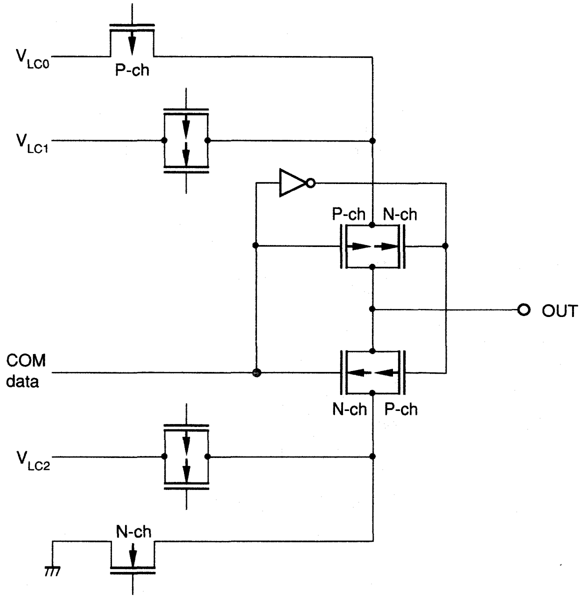


P.U.R. : Pull-Up Resistor

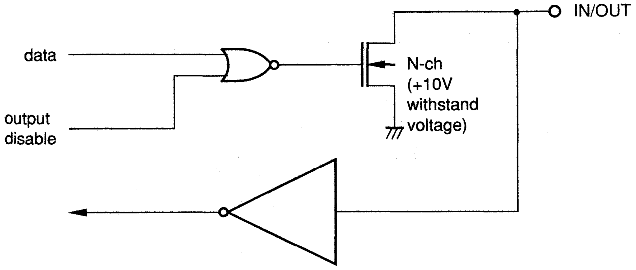
(8) Type G–A



(9) Type G–B

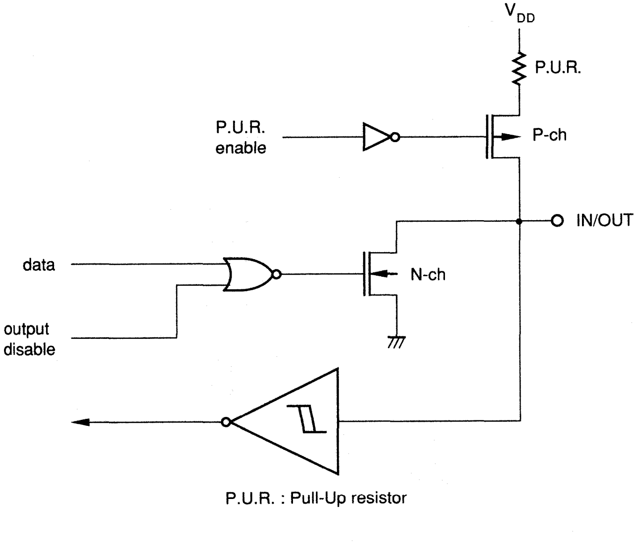


(10) Type M–B



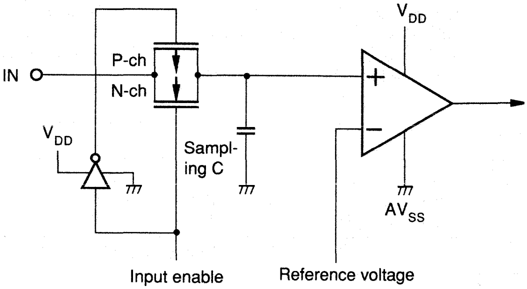
Middle-high voltage
input buffer
(+10V withstand voltage)

(11) Type M–C



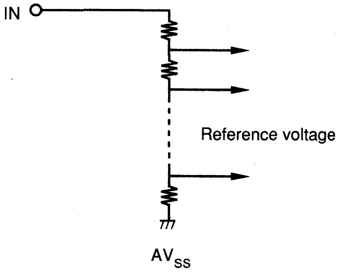
P.U.R. : Pull-Up resistor

(12) Type Y



(from series resistor string voltage tap)

(13) Type Z



### 10.6 DIFFERENCES BETWEEN μPD75P328GC AND μPD75328

Because the μPD75P328GC is a product that incorporates an on-chip, writeable PROM in place of the masked ROM in the μPD75328 program memory, the μPD75P328GC and the μPD75328 have similar hardware and perform similar CPU functions. They differ simply in their type of program memory and mask options. Tables 10.6-1 shows these differences.
Refer to the μPD75328 Users´ Manual for details of hardware and CPU functions.

**Table 10.6-1**

| Item | μPD75P328 | μPD75328 |
|---|---|---|
| Program memory | • One-time PROM<br>• 8064 x 8 bits<br>• 0000H – 1F7FH | • Mask ROM<br>• 8064 x 8 bits<br>• 0000H – 1F7FH |
| Port 4, 5 pull-up resistor | None | Mask option |
| Split resistor to supply LCD driving power | None | Mask option |
| Pin connection | $V_{PP}$ pin and one-time PROM program pins are included. | not available<br>. |
| Operation supply voltage range | 5 V ± 5% | 2.7 to 6.0 V |
| Package | 80-pin plastic QFP (bent lead) | |

### 10.7 One-Time PROM (Program Memory) Write and Verify

Program memory contained in the μPD75P328 is one-time PROM (8064 x 8 bits) which can be electrically written. The pins listed in the table given below are used for one-time PROM write and verify. Address is update by clock input from the X1 pin instead of address input.

**Table 10.7-1 Pin function**

| Pin name | Function |
|---|---|
| X1, X2 | Address update clock during when PROM is write/verification is input to the X1 pin. Its inverted signal is input to the X2 pin. |
| MD0-MD3 | Operation mode selection pins during PROM write/verification. |
| P40-P43 (low-order four bits)<br>P50-P53 (high-order four bits) | 8-bit data input/output pins during PROM write/verified. |
| $V_{DD}$ | Supply voltage apply pin.<br>During the normal operation, 5 V ± 5 % is applied; during PROM write/verification, +6 V is applied. |
| $V_{PP}$ | Voltage apply pin during PROM write/ verification. (Normally, $V_{DD}$ potential) |

Caution 1: Handle the pins not used during program memory write and verify as follows:
- Other than the XT2 pin: Connect to $V_{SS}$ via pull-down resistor.
- XT2 pin: Do not connect the pin.

Caution 2: The μPD75P328 does not have an erasion window, thus the program memory contents cannot be erased with ultraviolet rays.

10.7.1 Operating Mode during PROM is Write/Verification

When +6 V and +12.5 V are applied to the $V_{DD}$ and $V_{PP}$ pins, respctively, of the µPD75P328 the PROM write/verify mode is entered. The operation mode is selected according to the input signals to the MD0-MD3 pins as listed in Table 10.7-2.
Pins not used in PROM function should be pulled to $V_{SS}$.

**Table 10.7-2 Operating Mode**

| Operating mode specification | | | | | | Operating mode |
|---|---|---|---|---|---|---|
| $V_{DD}$ | $V_{PP}$ | MD0 | MD1 | MD2 | MD3 | |
| +12.5 V | +6 V | H | L | H | L | Clear program memory address |
| | | L | H | H | H | Write mode |
| | | L | L | H | H | Verify mode |
| | | H | X | H | H | Program inhibit mode |

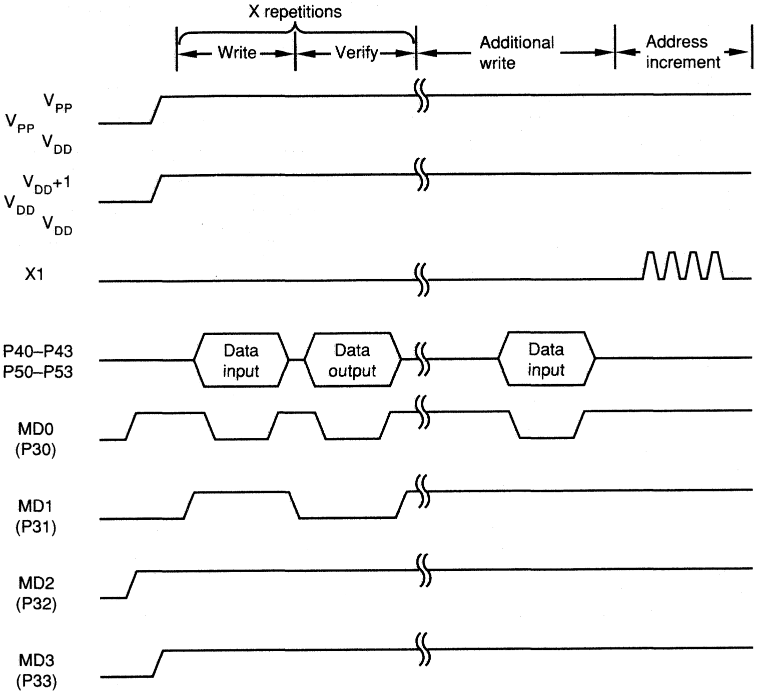
Remarks: X: L or H

10.7.2 PROM Write Procedure

PROM can be written at the high speed according to the following Procedure:

(1) Pull unused pins low to $V_{SS}$ with resistors. Set the X1 pin low.
(2) Supply 5 V to the $V_{DD}$ and $V_{PP}$ pins.
(3) Wait for 10 us.
(4) Selected the clear program memory address mode.
(5) Supply +6 V and +12.5 V to the $V_{DD}$ and $V_{PP}$ pins.
(6) Select the program inhibit mode.
(7) Write data in the 1-ms write mode.
(8) Select the program inhibit mode.
(9) Select the verify mode. If the data is written normally, proceed to (10). If it is not written normally, repeat (7) to (9).
(10) Supply X (number of (7)–(9) repetitions) x 1 ms program pulses (additional write).
(11) Select the program inhibit mode.
(12) Input four pulses to the X1 pin to update the program memory address by one.
(13) Repeat (7)–(12) until the end address is reached.
(14) Select the clear program memory address mode.
(15) Change the $V_{DD}$, $V_{PP}$ pin voltage to 5 V.
(16) Turn off the power.

Steps (2) to (12) are illustrated below:



10.7.3 PROM Read Procdure

The PROM contents can be read according to the following procedure:

(1) Pull unused pins low to $V_{SS}$ with resistors.
Set the X1 pin low.

(2) Supply 5 V to the $V_{DD}$ and $V_{PP}$ pins.

(3) Wait for 10 µs.

(4) Select the clear program memory address mode.

(5) Supply + 6 V and + 12.5 V to the $V_{DD}$ and $V_{PP}$ pins.

(6) Select the program inhibit mode.

(7) Select the verify mode. Input four clock pulses to the X1 pin. The data are output from the memory addresses, one address at a time, in a cycle of hour clock pulses which are input to the X1 pin.

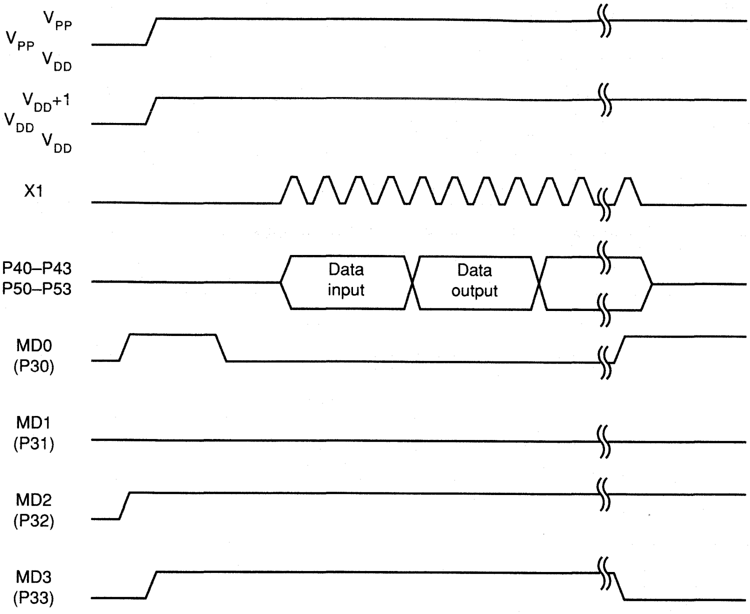
(8) Select the program inhibit mode.

(9) Select the clear program memory address mode.

(10) Change the $V_{DD}$, $V_{PP}$ voltage to 5 V.

(11) Turn off the power.

Steps (2) to (12) are illustrated below:



### 10.8 Electrical charactaristics

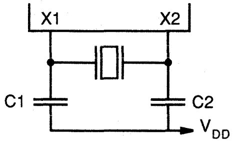
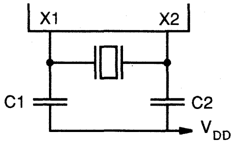
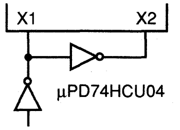
**Absolute Maximum Rating** (Ta=25°C)

| Prameter | Symbol | Condition | | Rating | Unit |
|---|---|---|---|---|---|
| Supply voltage | $V_{DD}$ | | | -0.3 to +7.0 | V |
| | $V_{PP}$ | | | -0.3 to +13.5 | V |
| Input voltage | $V_{I1}$ | Other than ports 4 and 5 | | -0.3 to $V_{DD}$ +0.3 | V |
| | (Note 1)<br>$V_{I2}$ | Ports 4 and 5 | Opendrain | -0.3 to +11 | V |
| Output voltage | $V_0$ | | | -0.3 to $V_{DD}$ +0.3 | V |
| High-level output current | $I_{OH}$ | 1 pins | | -15 | mA |
| | | All other pins | | -30 | mA |
| Low-level output current | (Note 2)<br>$I_{OL}$ | 1 pins | Peak value | 30 | mA |
| | | | Effective value | 15 | mA |
| | | Total of ports 0, 2, 3, 5 and 8 | Peak value | 100 | mA |
| | | | Effective value | 60 | mA |
| | | Total of ports 4, 6 and 7 | Peak value | 100 | mA |
| | | | Effective value | 60 | mA |
| Operating temperature | $T_{opt}$ | | | -10 to +70 | °C |
| Storage temperature | $T_{stg}$ | | | -65 to +150 | °C |

Note 1: When applying more than 10 volts to ports 4 or 5, the pull-up resistor must be 50K ohms min.

2: Effective value is obtained as follows:
(Effective) value = (Peak value) x $\sqrt{\text{duty cycle}}$

**Main system clock Oscillator Characteristics**
(Ta = -10 to +70°C, $V_{DD}$ = 5V+/-5%)

| Resonator | Recommended constants | Item | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Note 3<br>Ceramic resonator |  | Note 1<br>Oscillation frequency ($f_{XX}$) | | 1.0 | | 5.0 | MHz |
| | | Note 2<br>Oscillation stabilization time | When $V_{DD}$ reaches the minimum oscillator operating voltage. | | | 4 | ms |
| Crystal resonator |  | Note 1<br>Oscillation frequency ($f_{XX}$) | | 1.0 | 4.19 | 5.0 | MHz |
| | | Note 2<br>Oscillation stabilization time | | | | 10 | ms |
| External clock |  | Note 1<br>X1 input frequency ($f_X$) | | 1.0 | | 5.0 | MHz |
| | | X1 input high/ low level width ($t_{XH}$ / $t_{XL}$) | | 100 | | 500 | ns |

Note 1: The oscillation frequency and X1 input frequency values indicated in this table express only the characteristics of the oscillator. Therefore, refer to the AC characteristics for the instruction execution time.

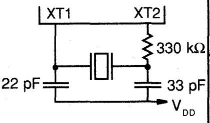
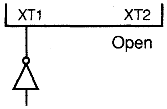
2: Oscillation stabilization time is defined as the time needed for the oscillator to stabilize after $V_{DD}$ is applied or after the STOP mode is released.

3: The following ceramic and crystal oscillators are recommended for the μPD75P328GC.

**Recommended Ceramic Resonators**

| Manufacture | Product | External capacitance (pF) | | Operating voltage range (V) | |
|---|---|---|---|---|---|
| | | C1 | C2 | MIN. | MAX. |
| MURATA | CSA 2.00MG | 30 | 30 | 4.75 | 5.25 |
| | CSA 4.19MG | 30 | 30 | 4.75 | 5.25 |
| | CSA 4.19MGU | 30 | 30 | 4.75 | 5.25 |
| | CST 4.19MG | 30pF (internally provided) | 30pF (internally provided) | 4.75 | 5.25 |

**Subsystem Clock Oscillator Characteristics**
(Ta =-10 to +70°C, $V_{DD}$ =5V+/-5%)

| Resonator | Recommended constants | Item | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Crystal resonator |  | Oscillation frequency ($f_{XT}$) | | 32 | 32.768 | 35 | kHz |
| | | Oscillation stabilization time | $V_{DD}$=4.5~6.0V | | 1.0 | 2 | s |
| External clock |  | XT1 input frequency ($f_{XT}$) | | 32 | | 100 | kHz |
| | | XT1 input high/ low level width ($t_{XTH}$/$t_{XTL}$) | | 5 | | 15 | µs |

**Capacitance** (Ta =25°C, $V_{DD}$ =0V)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Input capacitance | $C_{IN}$ | f=1 MHz Voltage should not be applied to any pin except those used for measurements. | | | 15 | pF |
| Output capacitance | $C_{OUT}$ | | | | 15 | pF |
| Input/output capacitance | $C_{IO}$ | | | | 15 | pF |

**DC Characteristics** (Ta =-10 to +70°C, $V_{DD}$ =5V±5%)

| Parameter | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| High-level input voltage | $V_{IH1}$ | Port 2, 3 | | 0.7$V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH2}$ | Port 0, 1, 6, 7, $\overline{\text{RESET}}$ | | 0.8$V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH3}$ | Port 4, 5 Open-drain | | 0.7$V_{DD}$ | | 10 | V |
| | $V_{IH4}$ | X1, X2, XT1 | | $V_{DD}$-0.5 | | $V_{DD}$ | V |
| Low-level input voltage | $V_{IL1}$ | Port 2, 3, 4, 5, 8 | | 0 | | 0.3$V_{DD}$ | V |
| | $V_{IL1}$ | Port 0, 1, 6, 7, $\overline{\text{RESET}}$ | | 0 | | 0.2$V_{DD}$ | V |
| | $V_{IL3}$ | X1, X2, XT1 | | 0 | | 0.4 | V |
| High-level output voltage | $V_{OH1}$ | Port 0, 2, 3, 6, 7, 8 BIAS | $I_{OH}$ =-1mA | $V_{DD}$-1.0 | | | V |
| | $V_{OH2}$ | BPO-7 | $I_{OH}$ =-100µA<br>Note 4 | $V_{DD}$-2.0 | | | V |
| Low-level output voltage | $V_{OL1}$ | Port 0, 2, 3, 4, 5, 6, 7,8 | Port 3, 4, 5 $I_{OL}$ =15mA | | 0.4 | 2.0 | V |
| | | | $I_{OL}$ =1.6mA | | | 0.4 | V |
| | $V_{OL2}$ | SB0, 1 open-drain | Pull-up resistor (R) =1k ohm min. | | | 0.2$V_{DD}$ | V |
| | $V_{OL3}$ | BP0-7 | $I_{OL}$ =100µA<br>Note 4 | | | 1.0 | V |
| High-level input leakage current | $I_{LIH1}$ | $V_{IN}$ =$V_{DD}$ | Other than X1, X2, and XT1 pins | | | 3 | µA |
| | $I_{LIH2}$ | | X1, X2, XT1 | | | 20 | µA |
| | $I_{LIH3}$ | $V_{IN}$ =10V | Port 4, 5 | | | 20 | µA |

**DC Characteristics** (Ta =-10 to +70°C, $V_{DD}$ =5V±5%)

| Parameter | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Low-level input leakage current | $I_{LIL1}$ | $V_{IN}$ =0V | Other than X1, X2, and XT1 pins | | | -3 | μA |
| | $I_{LIL2}$ | | X1, X2, XT1 | | | -20 | μA |
| High-level output leakage current | $I_{LOH1}$ | $V_{OUT}$ =$V_{DD}$ | Other than ports 4 and 5 | | | 3 | μA |
| | $I_{LOH2}$ | $V_{OUT}$ =10V | Port 4,5 | | | 20 | μA |
| Low-level output leakage current | $I_{LOL}$ | $V_{OUT}$ =0V | | | | -3 | μA |
| Internal pull-up resistor | $R_{LI}$ | Port 0, 1, 2, 3, 6, 7,8 (except P00) $V_{IN}$ =0V | | 15 | 40 | 80 | kΩ |
| LCD drive voltage | $V_{LCD}$ | | | 2.5 | | $V_{DD}$ | V |
| LCD output voltage deviation (Common) N. 7 | $V_{ODC}$ | $I_O$ = ± 5 μA | $V_{LCD0}$=$V_{LCD}$<br>$V_{LCD1}$=$V_{LCD}$x2/3<br>$V_{LCD2}$=$V_{LCD}$x1/3<br>2.7 V≤ $V_{LCD}$ ≤ $V_{DD}$ | 0 | | ±0.2 | V |
| LCD output voltage deviation (Segment) N. 7 | $V_{ODS}$ | $I_O$ = ± 1 μA | | 0 | | ±0.2 | V |
| Common output impedance | $R_{COM}$ | Note 3 | | | 3 | 6 | kΩ |
| Segment output impedance | $R_S$ | Note 3 | | | 15 | 20 | kΩ |
| Note 1 Supply current | $I_{DD1}$ | Note 5<br>4.19 MHz crystal oscillation C1=C2=22pF | Note 2 | | 5 | 15 | mA |
| | $I_{DD2}$ | | HALT mode | | 500 | 1500 | μA |
| | $I_{DD3}$ | Note 6<br>32kHz crystal oscillation | | | 350 | 1000 | μA |
| | | | HALT mode | | 35 | 100 | |
| | $I_{DD4}$ | XT1 =0V STOP mode | | | 0.5 | 20 | μA |

Note 1. The current drained through the internal pull-up resistors is not included.
2. When operated in the high-speed mode with the processor clock control register (PCC) set to 0011.
3. 2.5V ≤ $V_{LCD}$ ≤ $V_{DD}$
4. When any two pins of BP0 to BP3 and any two pins of BP4 to BP7 are used simultaneously for output.
5. Including the subsystem clock power consumption.
6. When system clock control register (SCC) is set to 1001; main system clock`s oscillation halted once sub-system clock in operation.
7. Voltage deviation shows differences between the expected values ($V_{LCDn}$; n = 0, 1, 2) of the segment and common output and output voltage.

**A/D Converter Characteristics** (Ta =-10 to +70°C, $V_{DD}$ =5V±5%, $AV_{SS}$ = $V_{SS}$ = 0V)

| Parameter | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Resolution | | | 8 | 8 | 8 | bit |
| Absolute accuracy *1 | | 2.5 V ≤ $V_{AREF}$ ≤ $V_{DD}$ | | | ±1.5 | LSB |
| Conversion time | $t_{CONV}$ | *2 | | | $168/f_X$ | s |
| Sampling time | $t_{SAMP}$ | *3 | | | $44/f_X$ | s |
| Analog input voltage | $V_{IAN}$ | | $AV_{SS}$ | | $V_{AREF}$ | V |
| Analog input impedance | $R_{AN}$ | | | 1000 | | MΩ |
| $V_{AREF}$current | $V_{AREF}$ | | | 1.0 | 2.0 | mA |

*1: Absolute accuracy except for quantization error (±1/2 LSB).

*2: Tme to EOC = 1 after conversion start instruction execution (40.1 µs when $f_X$ = 4.19 MHz)

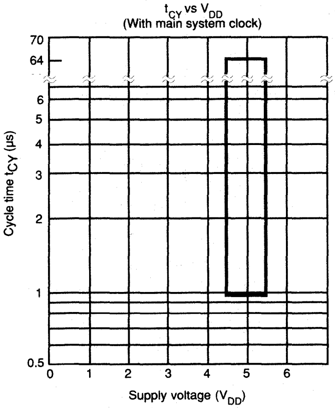
*3: Time to sampling end after conversion start instruction execution (10.5 µs when $f_X$ = 4.19 MHz)

**AC Characteristics** (Ta =-10 to +70°C, $V_{DD}$ =5V±5%) Operation other than serial transfer operation

| Parameter | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Note 1<br>Cycle time (minimum instruction execution time) | $t_{CY}$ | Operated with main system clock | 0.95 | | 64 | µs |
| | | Operated with sub-system clock | 114 | 122 | 125 | µs |
| TI0 input frequency | $f_{TI}$ | | 0 | | 1 | MHz |
| TI0 input high/low level width | $t_{TIH}$, $t_{TIL}$ | | 0.48 | | | µs |
| Interrupt input high/low level width | $t_{INTH}$, $t_{INTL}$ | INT0 | Note 2 | | | µs |
| | | KR0-7, INT1 2, 4 | 10 | | | µs |
| $\overline{\text{RESET}}$ low level width | $t_{RSL}$ | | 10 | | | µs |

Note 1. Cycle time (minimum instruction execution time) is determined by the oscillation frequency of the connected resonator, the system clock control register (SCC), and the processor clock control register (PCC). The figure to the right shows the $V_{DD}$ supply voltage vs. cycle time ($t_{CY}$) characteristic when operated with the main system clock.

2. Either $2t_{CY}$ or $64/f_{XX}$ can be selected by setting the interrupt mode register (IM0).



**Serial Transfer Operation**

**2-line/3-line serial I / 0 mode (SCK....Internal clock output)**

| Parameter | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY1}$ | | Output | 1600 | | | ns |
| $\overline{\text{SCK}}$ high / low level width | $t_{KL1}$<br>$t_{KH1}$ | | Output | $t_{KCL1}/2$<br>-50 | | | ns |
| SI set-up time (against $\overline{\text{SCK}}$↑ ) | $t_{SIK1}$ | | | 150 | | | ns |
| SI hold time (against $\overline{\text{SCK}}$↑) | $t_{KSI1}$ | | | 400 | | | ns |
| $\overline{\text{SCK}}$↓→ S0 output delay time | $t_{KSO1}$ | | | | | 250 | ns |

**2-line / 3-line serial I/0 mode (SCK....External clock input) :**

| Parameter | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY2}$ | | Input | 800 | | | ns |
| $\overline{\text{SCK}}$ high / low level width | $t_{KL2}$<br>$t_{KH1}$ | | Input | 400 | | | ns |
| SI set-up time (against $\overline{\text{SCK}}$↑ ) | $t_{SIK2}$ | | | 100 | | | ns |
| SI hold time (against $\overline{\text{SCK}}$ ↑ ) | $t_{KSI2}$ | | | 400 | | | ns |
| $\overline{\text{SCK}}$↓ → S0 output delay time | $t_{KSO2}$ | | | | | 300 | ns |

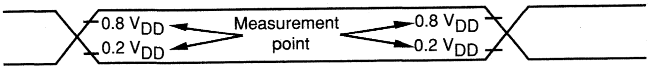
**SBI Mode (SCK....Internal clock output Master)) :**

| Parameter | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY3}$ | | 1600 | | | ns |
| $\overline{\text{SCK}}$ high / low level width | $t_{KL3}$<br>$t_{KH3}$ | | $t_{KCY}/2$<br>-50 | | | ns |
| SB0, SB1 set-up time (against $\overline{\text{SCK}}$↑) | $t_{SIK3}$ | | 150 | | | ns |
| SB0, SB1 hold time (against $\overline{\text{SCK}}$↑) | $t_{KSI3}$ | | $t_{KCY}/2$ | | | ns |
| $\overline{\text{SCK}}$↓ → SB0, SB1 output delay time | $t_{KSO3}$ | | 0 | | 250 | ns |
| $\overline{\text{SCK}}$↓ → SB0, SB1 ↓ | $t_{KSB}$ | | $t_{KCY}$ | | | ns |
| SB0, SB1↓ → $\overline{\text{SCK}}$↓ | $t_{SBK}$ | | $t_{KCY}$ | | | ns |
| SB0, SB1 low level width | $t_{SBL}$ | | $t_{KCY}$ | | | ns |
| SB0, SB1 high level width | $t_{SBH}$ | | $t_{KCY}$ | | | ns |

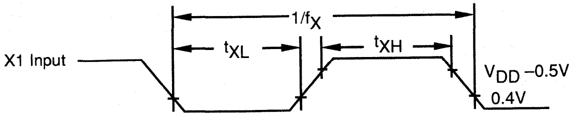
**SBI Mode (SCK.... External clock input (Slave)):**

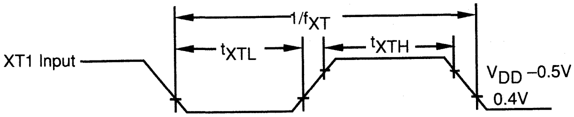
| Parameter | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| SCK cycle time | $t_{KCY4}$ | | 800 | | | ns |
| SCK high / low level width | $t_{KL4}$<br>$t_{KH4}$ | | 400 | | | ns |
| SB0, SB1 set-up time (against $\overline{\text{SCK}}$↑ ) | $t_{SIK4}$ | | 100 | | | ns |
| SB0, SB1 hold time (against $\overline{\text{SCK}}$↑) | $t_{KSI4}$ | | $t_{KCY}/2$ | | | ns |
| SCK ↓ → SB0, SB1 output delay time | $t_{KSO4}$ | | 0 | | 300 | ns |
| $\overline{\text{SCK}}$↑ → SB0, SB1 ↓ | $t_{KSB}$ | | $t_{KCY}$ | | | ns |
| SB0, SB1↓ → $\overline{\text{SCK}}$ ↓ | $t_{SBK}$ | | $t_{KCY}$ | | | ns |
| SB0, SB1 low level width | $t_{SBL}$ | | $t_{KCY}$ | | | ns |
| SB0, SB1 high level width | $t_{SBH}$ | | $t_{KCY}$ | | | ns |

AC Timing Measurement Points (Except X1, XT1 input)

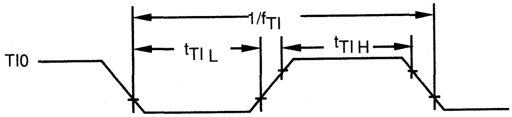


Clock Timing

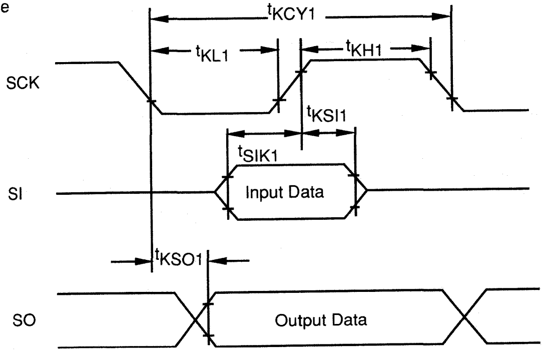




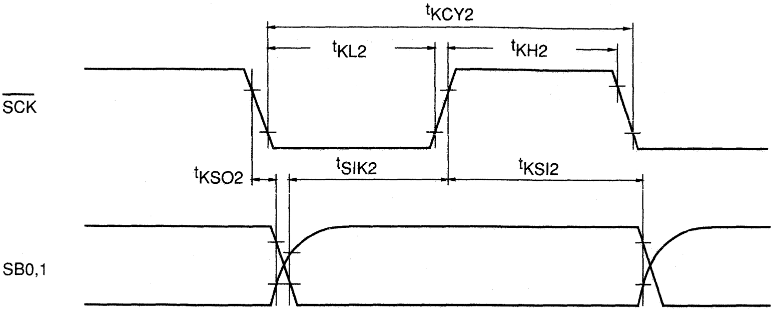
TIO Timing



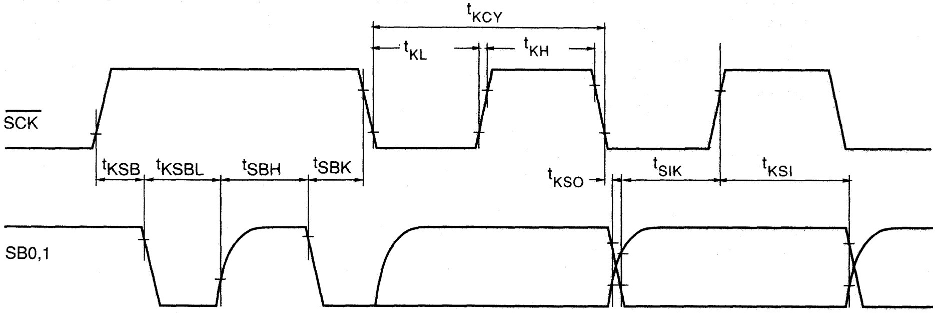
Serial Transfer Timing
3-line serial I/O mode



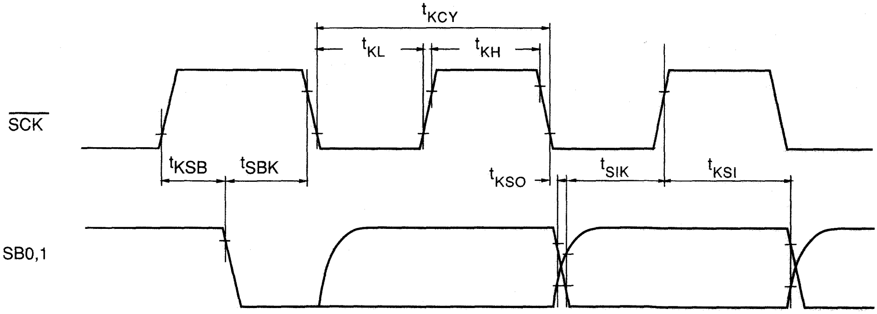
2-line serial I/O mode



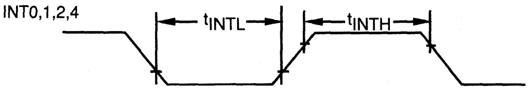
**Serial Transfer Timing** (SBI mode)
Bus release signal transfer:



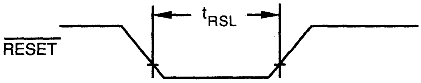
Command signal transfer:



Interrupt input timing



$\overline{\text{RESET}}$ input timing



**Data Memory STOP Mode Low Supply Voltage Data Retention Characteristic** (Ta = –10 to +70 °C)

| Parameter | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Data retention supply voltage | $V_{DDDR}$ | | 2.0 | | 6.0 | V |
| Data retention supply current (Note1) | $I_{DDDR}$ | $V_{DDDR}$ = 2.0V | | 0.1 | 10 | μA |
| Release signal set time | $t_{SREL}$ | | 0 | | | μs |
| Oscillation stabilization wait time (Note 2) | $t_{WAIT}$ | Release by $\overline{\text{RESET}}$ input | | $2^{17}/f_X$ | | ms |
| | | Release by interrupt request | | Note 3 | | ms |

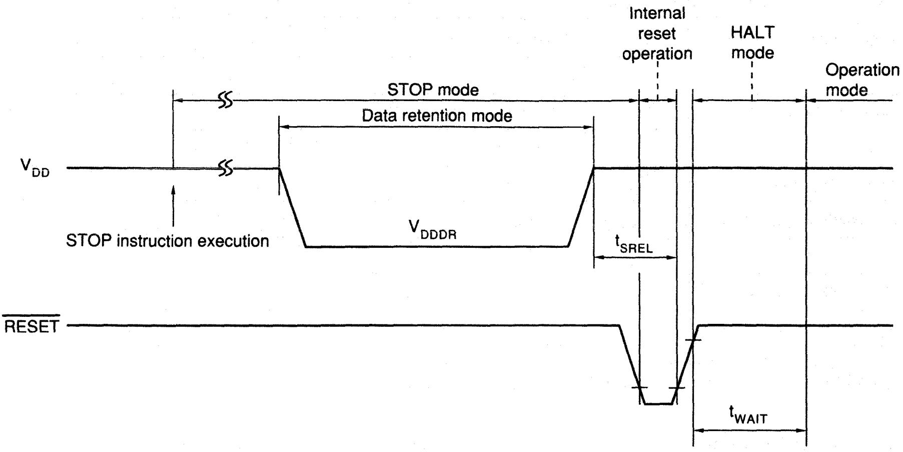
Note 1: The current drained through the internal pull-up resistor is not included

Note 2: The oscillation stabilization wait time is used to prevent unstable CPU operation at the beginning of oscillator operation, during which CPU operation is disabled.

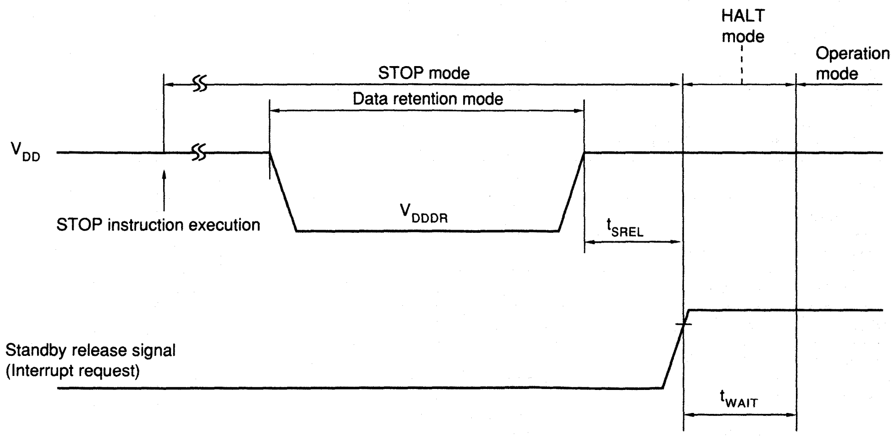
Note 3: This value depends on the setting of the basic interval timer mode register (BTM). (Refer to the table below. )

| BTM3 | BTM2 | BTM1 | BTM0 | WAIT time Parantheses ( ) indicates $f_X$ =4.19MHz |
|---|---|---|---|---|
| — | 0 | 0 | 0 | $2^{20}/f_X$ (Approximately 250 ms) |
| — | 0 | 1 | 1 | $2^{17}/f_X$ (Approximately 31.3 ms) |
| — | 1 | 0 | 1 | $2^{15}/f_X$ (Approximately 7.82 ms) |
| — | 1 | 1 | 1 | $2^{13}/f_X$ (Approximately 1.95 ms) |

Data Retention Timing (Releasing STOP mode by $\overline{\text{RESET}}$ )



Data Retention Timing
(Standby release signal: Releasing STOP mode by interrupt signal)



**DC Programming Characteristics** (Ta = 25 °C, $V_{DD}$ = 6.0 ± 0.25V, $V_{PP}$ = 12.5 ± 0.3V, $V_{SS}$ = 0V)

| Item | Symbol | Test Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| High level input voltage | $V_{IH1}$ | Other than X1, X2 | $0.7V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH2}$ | X1, X2 | $V_{DD}$ −0.5 | | $V_{DD}$ | V |
| Low level input voltage | $V_{IL1}$ | Other than X1, X2 | 0 | | 0.3 $V_{DD}$ | V |
| | $V_{IL2}$ | X1, X2 | 0 | | 0.4 | V |
| Input leak current | $I_{LI}$ | $V_{IN} = V_{IL}$ or $V_{IH}$ | | | 10 | μA |
| High level output voltage | $V_{OH}$ | $I_{OH}$ = −1 mA | $V_{DD}$ −1.0 | | | V |
| Low level output voltage | $V_{OL}$ | $I_{OL}$ = 1.6 mA | | | 0.4 | V |
| $V_{DD}$ power supply current | $I_{DD}$ | | | | 30 | mA |
| $V_{PP}$ power supply current | $I_{PP}$ | MD0 = $V_{IL}$, MD1 = $V_{IH}$ | | | 30 | mA |

Note 1: Ensure that $V_{PP}$ does not exceed +13.5V including overshoot.
Note 2: Ensure that $V_{DD}$ is applied before $V_{PP}$, and is turned off after $V_{PP}$.

**AC Programming Characteristics** (Ta = 25 °C, $V_{DD}$ = 6.0 ± 0.25V, $V_{PP}$ = 12.5 ± 0.3V, $V_{SS}$ = 0V)

| Item | Symbol | Note 1 | Test Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Address setup time (Note 2) (to MD0 ↓ ) | $t_{AS}$ | $t_{AS}$ | | 2 | | | μs |
| MD1 setup time (to MD0 ↓ ) | $t_{M1S}$ | $t_{OES}$ | | 2 | | | μs |
| Data setup time (to MD0 ↓ ) | $t_{DS}$ | $t_{DS}$ | | 2 | | | μs |
| Address hold time (Note 2) (to MD0 ↑ ) | $t_{AH}$ | $t_{AH}$ | | 2 | | | μs |
| Data hold time (to MD0 ↑ ) | $t_{DH}$ | $t_{DH}$ | | 2 | | | μs |
| MD0 ↑ → data output float delay time | $t_{DF}$ | $t_{DF}$ | | 0 | | 130 | ns |
| $V_{PP}$ setup time (to MD3 ↑ ) | $t_{VPS}$ | $t_{VPS}$ | | 2 | | | μs |
| $V_{DD}$ setup time (to MD3 ↑ ) | $t_{VDS}$ | $t_{VCS}$ | | 2 | | | μs |
| Initial program pulse width | $t_{PW}$ | $t_{PW}$ | | 0.95 | 1.0 | 1.05 | ms |
| Additional program pulse width | $t_{OPW}$ | $t_{OPW}$ | | 0.95 | | 21.0 | ms |
| MD0 setup time (to MD1 ↑ ) | $t_{M0S}$ | $t_{CES}$ | | 2 | | | μs |
| MD0 ↓ → data output delay time | $t_{DV}$ | $t_{DV}$ | MD0 = MD1 = $V_{IL}$ | | | 1 | μs |
| MD1 hold time (to MD0 ↑ ) | $t_{M1H}$ | $t_{OEH}$ | $t_{M1H} = t_{M1R} \geq 50\mu s$ | 2 | | | μs |
| MD1 recovery time (to MD0 ↓ ) | $t_{M1R}$ | $t_{OR}$ | | 2 | | | μs |
| Program Counter reset time | $t_{PCR}$ | — | | 10 | | | μs |
| X1 input high/low level width | $t_{XH}$,$t_{XL}$ | — | | 0.125 | | | μs |
| X1 input frequency | $f_X$ | — | | | | 4.19 | MHz |
| Initial mode setting time | $t_I$ | — | | 2 | | | μs |
| MD3 setup time (to MD1 ↑ ) | $t_{M3S}$ | — | | 2 | | | μs |

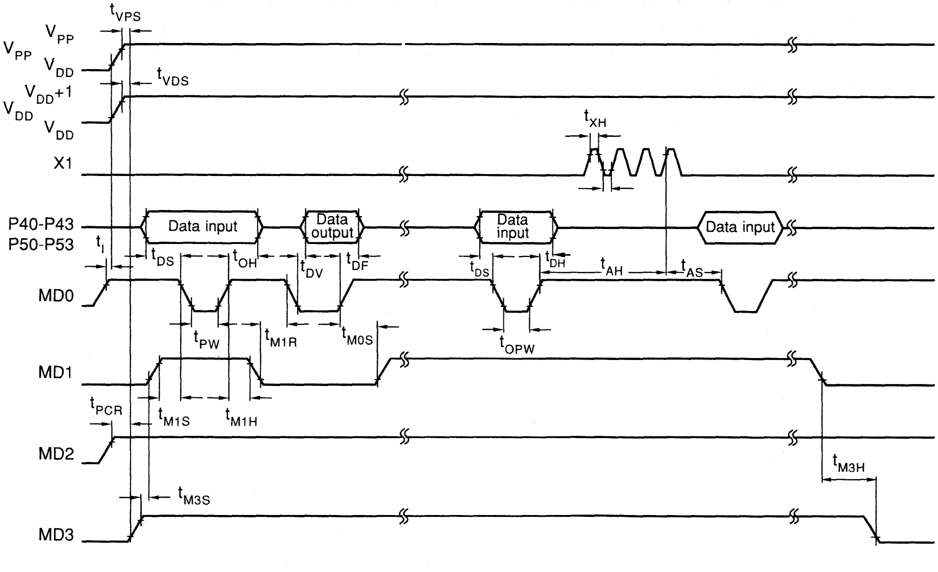
**AC Programming Characteristics** (Ta = 25 °C, $V_{DD}$ = 6.0 ± 0.25V, $V_{PP}$ = 12.5 ± 0.3V, $V_{SS}$ = 0V) (cont'd)

| Item | Symbol | Note 1 | Test Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| MD3 hold time (to MD1 ↓ ) | $t_{M3H}$ | — | | 2 | | | μs |
| MD3 setup time (to MD0 ↓ ) | $t_{M3SR}$ | — | For program memory read | 2 | | | μs |
| Address (Note 2) data output delay time | $t_{DAD}$ | $t_{ACC}$ | For program memory read | 2 | | | μs |
| Address (Note 2) data output hold time | $t_{HAD}$ | $t_{OH}$ | For program memory read | 0 | | 130 | ns |
| MD3 hold time (to MD0 ↑ ) | $t_{M3HR}$ | — | For program memory read | 2 | | | μs |
| MD3 ↓ → data output float delay time | $t_{DFR}$ | — | For program memory read | 2 | | | μs |

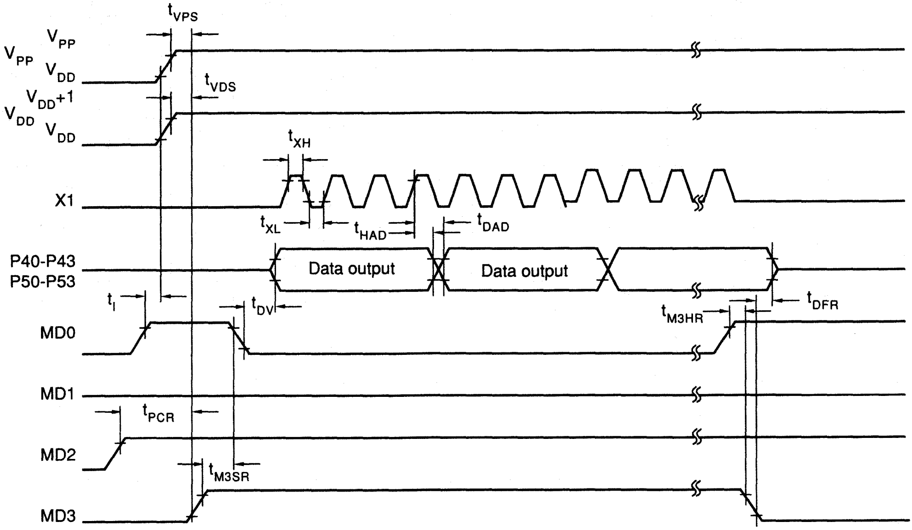
Note 1: Symbol for corresponding μPD27C256.

Note 2: The internal address signal is incremented by 1 by the rise of the 4th X1 input, and is not connected to a pin.

Program Memory Write Timing

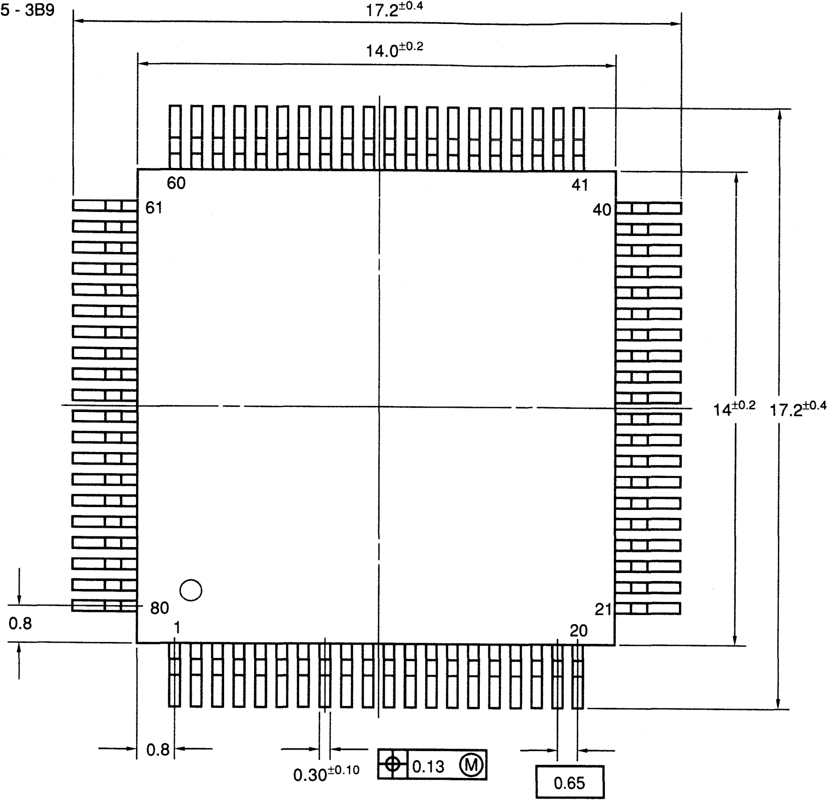


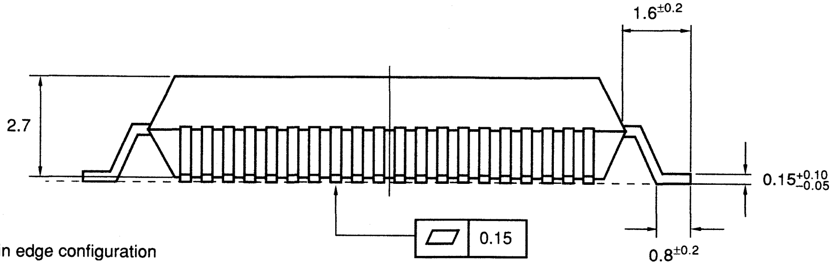
Program Memory Read Timing



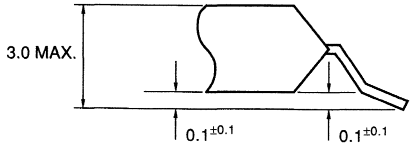
### 10.9 Package Information

80-pin plastic QFP (units in mm)
S80GC - 65 - 3B9
Top view





The pin edge configuration



## CHAPTER 5 μCOM-75X FAMILY STANDARD INSTRUCTION SET

### 1. FEATURED INSTRUCTIONS

The μCOM-75x standard instruction set is an improvement of the instruction set for the μPD7500-series, which is the predecessor of the μCOM-75X family.
It is a new epoch-making instruction set that maintains compatibility with the μPD7500 series, and has the following features:
(1) Bit manipulation instructions for various applications
(2) Efficient 4-bit manipulation instructions
(3) 8-bit manipulation instructions comparable to 8-bit microcomputer instructions
(4) GETI instruction to reduce program size
(5) String effect instructions and base correction instructions to raise program efficiency
(6) Table reference instructions appropriate for consecutive reference
(7) 1-byte relative branch instructions
(8) Easy-to-understand and well arranged NEC standard mnemonics

#### 1.1 GETI instruction

The GETI instruction is provided to convert into 1-byte instructions:

(a) Subroutine call instructions
(b) Branch instructions to all spaces
(c) Any desired 2-byte, 2-machine-cycle instruction (except BRCB or CALLF instruction)
(d) Pair of two 1-byte instructions.

When the GETI instruction is executed, the table at program memory addresses 0020H - 007FH is referenced, and the referenced 2-byte data is executed as one of the instruction (a) - (d) Thus, 48 instructions of the (a) - (d) category can be converted into 1-byte instructions.
If frequently used (a) - (d) instructions are converted into 1-byte instructions by using the GETI instruction, the number of program bytes can be drastically reduced.

#### 1.2 Bit manipulation instructions

Bit manipulation can be performed by using the following instructions:

| | | | | |
|---|---|---|---|---|
| (i) | Bit setting | : | SET1 | mem.bit |
| | | | SET1 | mem.bit* |
| (ii) | Bit clear | : | CLR1 | mem.bit |
| | | | CLR1 | mem.bit* |
| (iii) | Bit test | : | SKT | mem.bit |
| | | | SKT | mem.bit* |
| (iv) | Bit test | : | SKF | mem.bit |
| | | | SKF | mem.bit* |
| (v) | Bit test and clear | : | SKTCLR | mem.bit* |
| (vi) | Boolean operations | : | AND1 | CY, mem.bit* |
| | | | OR1 | CY, mem.bit* |
| | | | XOR1 | CY, mem.bit* |

The mem.bit* is a bit address indicated by bit manipulation addressing (fmem. bit, pmem. @ L, @ H + mem. bit).
All these bit manipulation instructions are always applicable to input/output ports; therefore, the input/output ports can be handled very efficiently.

### 1.3 String effect instructions

The instruction set provides the following two types of string effect instructions:

(a) MOV A, #n4 or MOV XA, #n8
(b) MOV HL, #n8

"String effect" places these instructions at contiguous addresses.

```
Example:  A0   : MOV A, #0
          A1   : MOV A, #1
          XA7  : MOV XA, # 07
```

If the string effect instructions are placed as in this example, when the first execution address is A0, the following two instructions are replaced with NOP instructions for execution; when the first address is A1, the following one instruction is replaced with a NOP instruction for execution. That is, only the first executed instruction is valid, and all string effect instructions following are handled as NOP instructions.
The string effect instructions enable efficient constant loading into a given accumulator (A register, register pair XA) and data pointer (register pair HL).

### 1.4 Base correction instructions

Depending on the application, the result of 4-bit data addition or subtraction (made in binary) must be converted into a decimal number or subjected to sixenary correction, as with the time of day.
Thus, the μCOM-75x standard instruction set contains base correction instructions to correct the result of 4-bit data addition or subtraction to any desired base number.

(a) Base correction during addition

Assume that the correction base value is m.
By using the combination of

```
ADDS A, #16-m
ADDC A, @ HL ; A, C ←— A + (HL) + C
ADDS A, #m
```

the accumulator and memory (HL) contents are added together, and the addition result is subjected to m correction. Overflow is left in the carry flag.
If a carry results from execution of the ADDC A, @ HL instruction, the following ADDS A, #n4 instruction is skipped. If no carry results, the ADDS A, # n4 instruction is executed. At this time, the skip function for this instruction is inhibited; even if a carry results from the addition, the following instruction will not be skipped.
Therefore, the program can be written following the ADDS A, #n4 instruction.

Example: To add the accumulator and memory together in decimal.

```
ADDS A, #6
ADDC A, @ HL ; A, C ←— A + (HL) + C
ADDS A, #10
         :
         :
```

(b) Base correction during subtraction

Assume that the correction base value is m.
By using the combination of

```
SUBC A, @ HL
ADDS A, #m
```

memory (HL) is subtracted from the accumulator, and the result is subjected to m correction.

Underflow is left in carry flag.

If a borrow does not result from execution of the SUBC A, @ HL instruction, the following ADDS A, #n4 instruction is skipped. If a borrow results, the ADDS A, #n4 instruction is executed. At this time, the skip function of this instruction is inhibited; even if a carry results from the addition, the following instruction will not be skipped.
Therefore, the program can be written following the ADDS A, #n4 instruction.

### 1.5 Skip instruction and number of machine cycles required for skip

The µCOM-75x standard instruction set contains the skip function involved in condition decision.
If the skip condition is satisfied when a skip instruction (instruction having the skip condition) is executed, the skip function skips the next instruction and executeds the instruction following the skipped instruction.
When a skip occurs, the number of machine cycles required for the skip is

(a) Two machine cycles if the skipped instruction following the skip instruction is a 3-byte instruction (BR !addr or CALL !addr).

(b) One machine cycle when the skipped instruction is not listed in (a) above.

## 2. Instruction Set and Operations

(1) Operand representation format and description method
In the operand field of each instruction, enter the operand according to the description method for the instruction operand representation format (details are based on the assembler specifications). When more than one entry exits under the description method, select one of the entries.
The uppercase alphabetic characters and the symbols + and - are keywords which must be entered exactly as shown.
For immediate data, enter the proper numeric value or label.
The register and flag abbreviations shown in Figure 2.3-1 can be entered as labels instead of mem, fmem, pmem, bit, etc., (however, labels that can be entered instead of fmem or pmem are limited).

| Representation format | Description method |
|---|---|
| reg<br>reg1 | X, A, B, C, D, E, H, L<br>X, B, C, D, E, H, L |
| rp<br>rp1<br>rp2 | XA, BC, DE, HL<br>BC, DE, HL<br>BC, DE |
| rpa<br>rpa1 | HL, DE, DL<br>DE, DL |
| n4<br>n8 | 4-bit immediate data or label<br>8-bit immediate data or label |
| mem<br>bit | 8-bit immediate data or label*<br>2-bit immediate data or label |
| fmem<br>pmem | FB0H-FBFH, FF0H-FFFH immediate data or label<br>FC0H-FFFH immediate data or label |
| addr<br>caddr<br>faddr | 0000H-3F7FH immediate data or label (Note)<br>12-bit immediate data or label<br>11-bit immediate data or label |
| taddr | 20H-7FH immediate data (where bit 0 = 0) or label |
| PORTn<br>IEXXX<br>MBn | PORT, n = number of port<br>Interrupt enable flags, XXX specifies the interrupt sorce<br>MB0, MB1, MB15 |

* For 8-bit data processing, only even addresses can be specified.
Note: Depends of ROM size.

(2) Legend in explanation of operation

| | |
|---|---|
| A | : A register, 4-bit accumulator |
| B | : B register, |
| C | : C register, |
| D | : D register, |
| E | : E register, |
| H | : H register, |
| L | : L register, |
| X | : X register, |
| XA | : XA register, pair, 8-bit accumulator |
| BC | : BC register pair |
| DE | : DE register pair |
| HL | : HL register pair |
| DL | : DL register pair |
| PC | : Program counter |
| SP | : Stack pointer |
| CY | : Carry flag, bit accumulator |
| PSW | : Program status word |
| MBE | : Memory bank enable flag |
| MBS | : Memory bank enable flag |

| | | |
|---|---|---|
| PORTn | : | Port n (n = number of port) |
| IME | : | Interrupt master enable flag |
| IExxx | : | Interrupt enable flag, xxx specifies the interrupt source |
| PCC | : | Processor clock control register |
| • | : | Separation between address and bit |
| (xx) | : | Contents addressed by xx |
| xxH | : | Hexadecimal data |

(3) Explanation of symbols under the "addressing area" column

| | | |
|---|---|---|
| *1 | MB = MBE • MBS<br>(MBS = 0, 1, 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, 15) | |
| *4 | MB = 15, fmem = FB0H - FBFH,<br>FF0H - FFFH | |
| *5 | MB = 15, pmem = FC0H - FFFH, | Program memory addressing |
| *6 | addr = 0000H - 3F7FH* | |
| *7 | addr = (Current PC) -15~(Current PC) -1,<br>(Current PC) +2~(Current PC) +16 | |
| *8 | caddr = 0000H - 0FFFH ($PC_{13,12}$=00B) or<br>1000H - 1FFFH ($PC_{13,12}$=01B) or<br>2000H - 2FFFH ($PC_{13,12}$=10B) or<br>3000H - 3F7FH ($PC_{13,12}$=11B) | |
| *9 | faddr = 0000H - 07FFH | |
| *10 | taddr = 0020H - 007FH | |

Remarks 1: MB indicates memory bank that can be accessed.
2: in *2, MB = 0 independently of how MBE and MBS are set.
3: In *4 and *5, MB = 15 independently of how MBE and MBS are set.
4: *6 to *10 indicate the areas that can be accessed.

* Depends of ROM size.

(4) Explanation of the column number of machine cycles

S denotes the number of machine cycles required by skip operation. The value of S varies as follows:

- When no skip is made: S = 0
- When the skipped instruction is a 1- or 2-byte instruction: S = 1
- When the skipped instruction is a 3-byte instruction (BR !addr or CALL !addr) : S = 2

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

One machine cycle is equal to one cycle of CPU clock Φ. The time can be selected from among three types by setting PCC.

| Instruction group | Mnemonic | Operand | Number of bytes | Number of machine cycles | Operation | Addressing area | Skip condition |
|---|---|---|---|---|---|---|---|
| Transfer instruction | MOV | A, #n4 | 1 | 1 | A ← n4 | | String effect A |
| | | reg1, #n4 | 2 | 2 | regl ← 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, @ 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, reg1 | 2 | 2 | A ← reg1 | | |
| | | XA, rp | 2 | 2 | XA ← rp | | |
| | | reg1, A | 2 | 2 | reg1 ← A | | |
| | | rp1, XA | 2 | 2 | rp1 ← A | | |
| | XCH | A, @ HL | 1 | 1 | A ↔ (HL) | *1 | |
| | | 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 ← $(PC_{13\text{-}8}+DE)_{ROM}$ | | |
| | | XA, @ PCXA | 1 | 3 | XA ← $(PC_{13\text{-}8}+XA)_{ROM}$ | | |
| Operation instruction | ADDS | A, #n4 | 1 | 1+S | A ← A+n4 | | carry |
| | | A, @ HL | 1 | 1+S | A ← A+(HL) | *1 | carry |
| | ADDC | A, @ HL | 1 | 1 | A, CY ← A + (HL) + CY | *1 | |
| | SUBS | A, @ HL | 1 | 1+S | A ← A - (HL) | *1 | borrow |
| | SUBC | A, @ HL | 1 | 1 | A, CY ← A - (HL) - CY | *1 | |
| | AND | A, #n4 | 2 | 2 | A ← A$\wedge$ n4 | | |
| | | A, @ HL | 1 | 1 | A ← A$\wedge$ (HL) | *1 | |
| | OR | A, #n4 | 2 | 2 | A ← A$\vee$ n4 | | |
| | | A, @ HL | 1 | 1 | A ← A$\vee$ (HL) | *1 | |
| | XOR | A, #n4 | 2 | 2 | A ← A$\veebar$ n4 | | |
| | | A, @ HL | 1 | 1 | A ← A$\veebar$ (HL) | *1 | |

| Instruction group | Mnemonic | Operand | Number of bytes | Number of machine cycles | Operation | Addressing area | Skip condition |
|---|---|---|---|---|---|---|---|
| Accumulator handling | RORC | A | 1 | 1 | CY ← $A_0$, $A_3$ ← CY, $A_{n-1}$ ← $A_n$ | | |
| | NOT | A | 2 | 2 | A ← $\overline{A}$ | | |
| Increment and decrement instructions | INCS | reg | 1 | 1+S | reg ← reg +1 | | reg = 0 |
| | | @ HL | 2 | 2+S | (HL) ← (HL) +1 | *1 | (HL) = 0 |
| | | mem | 2 | 2+S | (mem) ← (mem) +1 | *3 | (mem) = 0 |
| | DECS | reg | 1 | 1+S | reg ← reg - 1 | | reg = FH |
| Comparison instructions | 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) |
| | | A, reg | 2 | 2+S | Skip if A = reg | | A = reg |
| Carry flag handling instructions | SET1 | CY | 1 | 1 | CY ← 1 | | |
| | CLR1 | CY | 1 | 1 | CY ← 0 | | |
| | SKT | CY | 1 | 1+S | Skip if CY = 1 | | CY = 1 |
| | NOT1 | CY | 1 | 1 | CY ← $\overline{CY}$ | | |
| 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 |

5

| Instruction group | Mnemonic | Operand | Number of bytes | Number of machine cycles | Operation | Addressing area | Skip condition |
|---|---|---|---|---|---|---|---|
| Memory bit manipulation instruction | AND1 | CY, fmem.bit | 2 | 2 | $CY \leftarrow CY \wedge (fmem.bit)$ | *4 | |
| | | CY, pmem. @ L | 2 | 2 | $CY \leftarrow CY \wedge$<br>$(pmem_{7-2}+L_{3-2}.bit\ (L_{1-0}))$ | *5 | |
| | | CY,<br>@ H + mem.bit | 2 | 2 | $CY \leftarrow CY \wedge (H + mem_{3-0}.bit)$ | *1 | |
| | OR1 | CY, fmem.bit | 2 | 2 | $CY \leftarrow CY \vee (fmem.bit)$ | *4 | |
| | | CY, pmem.@ L | 2 | 2 | $CY \leftarrow CY \vee$<br>$(pmem_{7-2}+ L_{3-2}.bit\ (L_{1-0}))$ | *5 | |
| | | CY,<br>@ H + mem.bit | 2 | 2 | $CY \leftarrow CY \vee (H + mem_{3-0}.bit)$ | *1 | |
| | XOR1 | CY, fmem.bit | 2 | 2 | $CY \leftarrow CY \veebar (fmem.bit)$ | *4 | |
| | | CY, pmem.@ L | 2 | 2 | $CY \leftarrow CY \veebar$<br>$(pmem_{7-2}+ L_{3-2}.bit\ (L_{1-0}))$ | *5 | |
| | | CY,@H+mem.bit | 2 | 2 | $CY \leftarrow CY \veebar (H + mem_{3-0}.bit)$ | *1 | |
| Branch instructions | BR | addr | - | - | $PC_{13-0} \leftarrow addr$<br>(Select appropriate instruction from among BR !addr, BRCB !caddr, and BR \$addr according to the assembler being used.) | *6 | |
| | | !addr | 3 | 3 | $PC_{13-0} \leftarrow addr$ | *6 | |
| | | \$addr | 1 | 2 | $PC_{13-0} \leftarrow addr$ | *7 | |
| | BRCB | !caddr | 2 | 2 | $PC_{13-0} \leftarrow PC_{13,12} + caddr_{11-0}$ | *8 | |
| Subroutine stack control instructions | CALL | !addr | | | (SP -4) (SP - 1)<br>$(SP\ -2) \leftarrow PC_{11-0}$<br>$(SP\ -3) \leftarrow (MBE, 0, PC_{13,12})$<br>$PC_{13-0} \leftarrow addr, SP \leftarrow SP - 4$ | | |
| | CALLF | !faddr | 3<br>2 | 3<br>2 | (SP -4) (SP - 1)<br>$(SP\ -2) \leftarrow PC_{11-0}$<br>$(SP\ -3) \leftarrow (MBE, 0, PC_{13,12})$<br>$PC_{13-0} \leftarrow 0, faddr,$<br>$SP \leftarrow SP - 4$ | *6<br>*9 | |
| | RET | | 1 | 3 | $(MBE, PC_{13,12}) \leftarrow (SP\ +1)$<br>$PC_{11-0} \leftarrow (SP)(SP + 3)(SP + 2)$<br>$SP \leftarrow SP + 4$ | | |
| | RETS | | 1 | 3+S | $(MBE, PC_{13,12}) \leftarrow (SP\ +1)$<br>$PC_{11-0} \leftarrow (SP)\ (SP+3)\ (SP + 2)$<br>$SP \leftarrow SP+4$, then skip unconditionally | | Unconditional |
| | RETI | | 1 | 3 | $PC_{13,12} \leftarrow (SP + 1)$<br>$PC_{11-0} \leftarrow (SP)\ (SP+3)\ (SP+2)$<br>$PSW \leftarrow (SP + 4)\ (SP + 5),$<br>$SP \leftarrow SP + 6$ | | |
| | PUSH | rp | 1 | 1 | $(SP\ -1)\ (SP - 2) \leftarrow rp,$<br>$SP \leftarrow SP - 2$ | | |
| | | BS | 2 | 2 | $(SP\ -1) \leftarrow MBS,$<br>$(SP\ -2) \leftarrow 0, SP \leftarrow SP - 2$ | | |

| Instruction group | Mnemonic | Operand | Number of bytes | Number of machine cycles | Operation | Addressing area | Skip condition |
|---|---|---|---|---|---|---|---|
| Subroutine stack control instructions | POP | rp | 1 | 1 | rp ← (SP + 1) (SP),<br>SP ← SP + 2 | | |
| | | BS | 2 | 2 | MBS ← (SP+1),<br>SP ← SP + 2 | | |
| Interrupt control instructions | EI | | 2 | 2 | IME ← 1 | | |
| | | IEXXX | 2 | 2 | IEXXX ← 1 | | |
| | DI | | 2 | 2 | IME ← 0 | | |
| | | IEXXX | 2 | 2 | IEXXX ← 0 | | |
| Input output instructions | (Note 1)<br>IN | A, $PORT_n$ | 2 | 2 | A ← $PORT_n$ (n = port number) | | |
| | | XA, $PORT_n$ | 2 | 2 | XA ← $PORT_{n+1}$,$PORT_n$<br>(n = port number) | | |
| | (Note 1)<br>OUT | $PORT_n$, A | 2 | 2 | $PORT_n$ ← A (n = port number) | | |
| | | $PORT_n$, XA | 2 | 2 | $PORT_{n+1}$ ,$PORT_n$ ← XA<br>(n = port number) | | |
| CPU control instruction | HALT | | 2 | 2 | Set HALT Mode (PCC.2 ← 1) | | |
| | STOP | | 2 | 2 | Set STOP Mode (PCC.3 ← 1) | | |
| | NOP | | 1 | 1 | No Operation | | |
| Special instructions | SEL | $MB_n$ | 2 | 2 | $MBS_n$ ← n (n = 0, 1, 15) | | |
| | (Note 2)<br>GETI | taddr | 1 | 3 | • For TBR instruction<br>$PC_{13-0}$ ← $(taddr)_{5-0}$+ (taddr+1)<br>• For TCALL instruction<br>(SP -4) (SP -1)<br>(SP -2) ← $PC_{11-0}$<br>(SP -3) ← (MBE, 0, 0, $PC_{13,\ 12}$)<br>$PC_{13-0}$ ← $(taddr)_{5-0}$+ (taddr+1)<br>SP ← SP-4<br>• For any other instruction than TBR or TCALL (taddr) (taddr+1) instruction execution | *10 | Dependent on the referenced instruction |

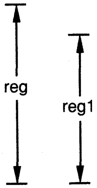
Note 1: To execute the IN or OUT instruction, preset MBE = 0 or (MBE = 1 and MBS = 15). The number of I/O ports allowing 8 bit data transfer, differs among the device types. Therfore, please check carefully the I/O pin manipulation instruction list for each device type which is available in the chapter about the on chip peripherals.

Note 2: The TBR and TCALL instructions are assembler pseudo-instructions for GETI instruction table definition.

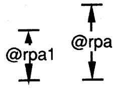
## 3. Operation Codes of Each Instructions

(1) Explaination of operation code symbols

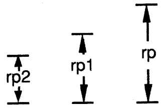
| $R_2$ | $R_1$ | $R_0$ | reg |
|---|---|---|---|
| 0 | 0 | 0 | A |
| 0 | 0 | 1 | X |
| 0 | 1 | 0 | L |
| 0 | 1 | 1 | H |
| 1 | 0 | 0 | E |
| 1 | 0 | 1 | D |
| 1 | 1 | 0 | C |
| 1 | 1 | 1 | B |



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



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



| $N_5$ | $N_2$ | $N_1$ | $N_0$ | IExxx |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | IEBT |
| 0 | 0 | 1 | 0 | IEW |
| 0 | 1 | 0 | 0 | IET0 |
| 0 | 1 | 0 | 1 | IECSI |
| 0 | 1 | 1 | 0 | IE0 |
| 1 | 0 | 0 | 0 | IE4 |
| 1 | 0 | 1 | 1 | IEKS |
| 1 | 1 | 1 | 0 | IE1 |

In : Immediate data for n4, n8
Dn : Immediate data for mem
Bn : Immediate data for bit
Nn : Immediate data for n or IExxx
Tn : Immediate data for taddr x 1/2
An : Immediate data for (relative address distance to branch destination address (2-16)) -1
Sn : Immediate data for one´s complement of (relative address distance to branch destination address (15-1))

(2) Bit manipulation addressing operation codes

$\boxed{*1}$ under the operand cloumn denotes that the following three addressing modes are available:

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

The second byte $\boxed{*2}$ of the operation code corresponding to each addressing mode shown above is as listed below:

| *1 | Second byte of operation code | Bits that can be accessed |
|---|---|---|
| fmem.bit | 1 0 $B_1$ $B_0$ $F_3$ $F_2$ $F_1$ $F_0$ | FB0H-FBFH bits that can be manipulated |
| | 1 1 $B_1$ $B_0$ $F_3$ $F_2$ $F_1$ $F_0$ | FF0H-FFFH bits that can be manipulated |
| pmem.@L | 0 1 0 0 $G_3$ $G_2$ $G_1$ $G_0$ | FC0H-FFFH bits that can be manipulated |
| @H+mem.bit | 0 0 $B_1$ $B_0$ $D_3$ $D_2$ $D_1$ $D_0$ | Bits that can be manipulated in an accessible memory bank |

Bn : Immediate data for bit
Fn : Immediate data for fmem
(low-order four bits of address are indicated)
Gn : Immediate data for pmem
(bits 5-2 of address are indicated)
Dn : Immediate data for mem
(low-order four bits of address are indicated)

| Instruction group | Mnemonic | Operand | Operation code | | |
|---|---|---|---|---|---|
| | | | $B_1$ | $B_2$ | $B_3$ |
| Transfer instructions | MOV | A, #n4 | 0 1 1 1 $I_3$ $I_2$ $I_1$ $I_0$ | | |
| | | reg1, #n4 | 1 0 0 1 1 0 1 0 | $I_3$ $I_2$ $I_1$ $I_0$ 1 $R_2$ $R_1$ $R_0$ | |
| | | rp, #n8 | 1 0 0 0 1 $P_2$ $P_1$ 1 | $I_7$ $I_6$ $I_5$ $I_4$ $I_3$ $I_2$ $I_1$ $I_0$ | |
| | | A, @rpa | 1 1 1 0 0 $Q_2$ $Q_1$ $Q_0$ | | |
| | | XA, @HL | 1 0 1 0 1 0 1 0 | 0 0 0 1 1 0 0 0 | |
| | | @HL, A | 1 1 1 0 1 0 0 0 | | |
| | | @HL, XA | 1 0 1 0 1 0 1 0 | 0 0 0 1 0 0 0 0 | |
| | | A, mem | 1 0 1 0 0 0 1 1 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | XA, mem | 1 0 1 0 0 0 1 0 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ 0 | |
| | | mem, A | 1 0 0 1 0 0 1 1 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | mem, XA | 1 0 0 1 0 0 1 0 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ 0 | |
| | | A, reg1 | 1 0 0 1 1 0 0 1 | 0 1 1 1 1 $R_2$ $R_1$ $R_0$ | |
| | | XA, rp' | 1 0 1 0 1 0 1 0 | 0 1 0 1 1 $P_2$ $P_1$ 0 | |
| | | reg1, A | 1 0 0 1 1 0 0 1 | 0 1 1 1 0 $R_2$ $R_1$ $R_0$ | |
| | | rp'1, XA | 1 0 1 0 1 0 1 0 | 0 1 0 1 0 $P_2$ $P_1$ 0 | |
| | XCH | A, @rpa | 1 1 1 0 1 $Q_2$ $Q_1$ $Q_0$ | | |
| | | XA, @HL | 1 0 1 0 1 0 1 0 | 0 0 0 1 0 0 0 1 | |
| | | A, mem | 1 0 1 1 0 0 1 1 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | XA, mem | 1 0 1 1 0 0 1 0 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ 0 | |
| | | A, reg1 | 1 1 0 1 1 $R_2$ $R_1$ $R_0$ | | |
| | | XA, rp | 1 0 1 0 1 0 1 0 | 0 1 0 0 0 $P_2$ $P_1$ 0 | |
| | MOVT | XA,@PCDE | 1 1 0 1 0 1 0 0 | | |
| | | XA,@PCXA | 1 1 0 1 0 0 0 0 | | |

| Instruction group | Mnemonic | Operand | Operation code | | |
|---|---|---|---|---|---|
| | | | B1 | B2 | B3 |
| Arithmetic and logical instruction | ADDS | A, #n4 | 0 1 1 0 $I_3$ $I_2$ $I_1$ $I_0$ | | |
| | | A, @HL | 1 1 0 1 0 0 1 0 | | |
| | ADDC | A, @HL | 1 0 1 0 1 0 0 1 | | |
| | SUBS | A, @HL | 1 0 1 0 1 0 0 0 | | |
| | SUBC | A, @HL | 1 0 1 1 1 0 0 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 | | |
| | 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 | | |
| | 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 | | |
| ACC | 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 | |
| Inc/dec | INCS | reg | 1 1 0 0 0 $R_2$ $R_1$ $R_0$ | | |
| | | @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 $R_2$ $R_1$ $R_0$ | | |
| Compare | 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 | | |
| | | A, reg | 1 0 0 1 1 0 0 1 | 0 0 0 0 1 $R_2$ $R_1$ $R_0$ | |
| Carry | 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 instruction | SET1 | mem.bit | 1 0 $B_1$ $B_0$ 0 1 0 1 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | $\boxed{*1}$ | 1 0 0 1 1 1 0 1 | *2 | |
| | CLR1 | mem.bit | 1 0 $B_1$ $B_0$ 0 1 0 0 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | $\boxed{*1}$ | 1 0 0 1 1 1 0 0 | *2 | |
| | SKT | mem.bit | 1 0 $B_1$ $B_0$ 0 1 1 1 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | $\boxed{*1}$ | 1 0 1 1 1 1 1 1 | *2 | |
| | SKF | mem.bit | 1 0 $B_1$ $B_0$ 0 1 1 0 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | $\boxed{*1}$ | 1 0 1 1 1 1 1 0 | *2 | |
| | SKTCLR | $\boxed{*1}$ | 1 0 0 1 1 1 1 1 | *2 | |
| | AND1 | CY, $\boxed{*1}$ | 1 0 1 0 1 1 0 0 | *2 | |
| | OR1 | CY, $\boxed{*1}$ | 1 0 1 0 1 1 1 0 | *2 | |
| | XOR1 | CY, $\boxed{*1}$ | 1 0 1 1 1 1 0 0 | *2 | |

| Instruction group | Mnemonic | Operand | Operation code | | |
|---|---|---|---|---|---|
| | | | $B_1$ | $B_2$ | $B_3$ |
| Branch | BR | !addr | 1 0 1 0 1 0 1 1 | 0 0 0 ← | addr → |
| | | (+16) - (+2)<br>$addr | 0 0 0 0 $A_3$ $A_2$ $A_1$ $A_0$ | | |
| | | (−1) - (−15) | 1 1 1 1 $S_3$ $S_2$ $S_1$ $S_0$ | | |
| | BRCB | !caddr | 0 1 0 1 ← | caddr → | |
| Subroutine stack | CALL | !addr | 1 0 1 0 1 0 1 1 | 0 1 0 ← | addr → |
| | CALLF | !faddr | 0 1 0 0 0 ← | faddr → | |
| | RET | | 1 1 1 0 1 1 1 0 | | |
| | RETS | | 1 1 1 0 0 0 0 0 | | |
| | RETI | | 1 1 1 0 1 1 1 1 | | |
| | PUSH | rp | 0 1 0 0 1 $P_2$ $P_1$ 1 | | |
| | | BS | 1 0 0 1 1 0 0 1 | 0 0 0 0 0 1 1 1 | |
| | POP | rp | 0 1 0 0 1 $P_2$ $P_1$ 1 | | |
| | | BS | 1 0 0 1 1 0 0 1 | 0 0 0 0 0 1 1 0 | |
| In/out | 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 | EI | | 1 0 0 1 1 1 0 1 | 1 0 1 1 0 0 1 0 | |
| | | IEXXX | 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 | |
| | | IEXXX | 1 0 0 1 1 1 0 0 | 1 0 $N_5$ 1 1 $N_2$ $N_1$ $N_0$ | |
| CPU control | HALT | | 1 0 0 1 1 1 0 1 | 1 0 1 0 0 0 1 1 | |
| | STOP | | 1 0 0 1 1 1 0 1 | 1 0 1 1 0 0 1 1 | |
| | NOP | | 0 1 1 0 0 0 0 0 | | |
| Special | SEL | 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 $T_5$ $T_4$ $T_3$ $T_2$ $T_1$ $T_0$ | | |

## 4. INSTRUCTION FUNCTIONS AND APPLICATIONS

### 4.1 Transfer instructions

**MOV A, #n4**

Function: A ← n4  n4 = $I_{3-0}$: 0-FH

This instruction transfers 4-bit immediate data n4 to the A register (4-bit accumulator).

It has a string effect (group A). If the MOV A, #n4 or MOV XA, #n8 instruction is consecutively placed, the instruction that follows the executed instruction will be replaced with an NOP instruction.

Application examples: (1) Sets 0BH into the accumulator:

```
MOV A, #0BH
```

(2) Selects data to be output to Port 3 from 0 to 2:

```
A0:   MOV A, #0
A1:   MOV A, #1
A2:   MOV A, #2
      OUT PORT3, A
```

**MOV reg1, #n4**

Function: Reg1 ← n4  n4 = $I_{3-0}$: 0-FH

This instruction transfers 4-bit immediate data n4 to the reg1 register (X, H, L, D, E, B, C).

**MOV rp, #n8**

Function: rp ← #n8  n8 = $I_{7-0}$: 00H-FFH

This instruction transfers 8-bit immediate data n8 to the register pair rp (XA, HL, DE, BC).

It has a string effect if the XA or HL is specified for rp. The string effect includes Group A (MOV A, #n4 and MOV XA, #n8 instruction). If instructions belonging to the same group are consecutively placed in a program, the string effect instruction that follows the executed instruction is replaced with an NOP instruction.

Application example: Sets 5FH into DE register pair.

```
MOV DE, #5FH
```

**MOV A, @rpa**

Function: A ← (rpa)

This instruction transfers the contents of the data memory location addressed by the register pair rpa (HL, DE, DL) to the A register.

**MOV XA, @HL**

Function: A ← (HL), X ← (HL+1)

This instruction transfers the contents of the data memory addressed by register pair HL to the A register and those of the next memory location to the X register.

If the L register value is an odd number, the least significant bit of the address information will be ignored.

Application example: Transfers data in addresses 3EH and 3FH to the register pair XA.

```
MOV HL, #3EH
MOV XA, @HL
```

**MOV @HL, A**

Function: (HL) ← A

This instruction transfers the A register value to the data memory location addressed by the register pair HL.

**MOV @HL, XA**

Function: (HL) ← A, (HL+1) ← X

This instruction transfers the A register value to the data memory location addressed by the regiser pair HL, and the X register value to the next data memory location.

If the L register value is an odd number, the least significant bit of the address information will be ignored.

**MOV A, mem**

Function: A ← (mem) mem = $D_{7-0}$: 00H-FFH

This instruction transfers the contents of the data memory location addressed by 8-bit immediate data mem to the A register.

**MOV XA, mem**

Function: A ← (mem), X ← (mem+1) mem = $D_{7-0}$: 00H-FEH

This instruction transfers the contents of the data memory location addressed by 8-bit immediate data mem to the A register, and the contents of the next data memory location to the X register.

The address specifiable with mem is an even address.

Application example: Transfers the data in location 40H and 41H to the register pair XA:
MOV XA, 40H

**MOV mem, A**

Function: (mem) ← A, mem = $D_{7-0}$ 00H-FFH

This instruction transfers the A register value to the data memory location addressed by 8-bit immediate data mem.

**MOV mem, XA**

Function: (mem) ← A, (mem+1) ← X mem = $D_{7-0}$: 00H-FEH

This instruction transfers the A register value to the data memory location addressed by 8-bit immediate data mem and the X register value to the next data memory location.

The address specifiable by mem is an even address.

**MOV A, reg1**

Function: A ← reg1

This instruction transfers the value of the register reg1 (X, H, L, D, E, B, C) to the A register.

**MOV XA, rp**

Function: XA ← rp

This instruction transfers the value of the register rp (XA, HL, DE, BC) to the register pair XA.

**MOV reg1, A**

Function: regl ← A

This instruction transfers the A register value to the register reg1 (X, H, L, D, E, B, C).

**MOV rp1, XA**

Function: rp1 ← XA

This instruction transfers the value of the register pair XA to another register pair, rp1 (HL, DE, BC).

**XCH A, @rpa**

Function: A ← (rpa)

This instruction exchanges the A register value with the contents of the data memory location addressed by the register pair rpa (HL, DE, DL).

Application example: Exchanges data in data memory locations 20 to 2FH with those in locations 30 to 3FH:

```
        SEL     MB0
        MOV     D, #2
        MOV     HL, #30H
LOOP:   XCH A,  @HL; A↔(3x)
        XCH A,  @DL; A↔(2x)
        XCH A,  @HL;  A↔(3x)
        INCS    L
        BR      LOOP
```

**XCH XA, @HL**

Function: A ↔ (HL), X ↔ (HL+1)

This instruction exchanges the A register value with the contents of the data memory location addressed by the register pair HL, and the X register value with the contents of the next data memory location.

If the Lregister value is an odd number, the least significant bit of the address information will be ignored.

**XCH A, mem**

Function: A ↔ (mem), X ↔ (mem+1)    mem = $D_{7-0}$: 00H-FFH

This instruction exchanges the A register value with the contents of the data memory location addressed by 8-bit immediate mem.

**XCH XA, mem**

Function: A ↔ (mem), X ↔ (mem+1)    mem = $D_{7-0}$: 00H-FEH

This instruction exchanges the A register value with the contents of the data memory location addressed by 8-bit immediate data mem and the X register value with the contents of the next data memory location. The address specifiable with mem is an even address.

**XCH A, reg1**

Function: A ↔ reg1

This instruction exchanges the A register value with the value of the register reg1 (X, H, L, D, E, B, C).

**XCH XA, rp**

Function: XA ↔ rp

This instruction exchanges the value of register pair XA with that of another register pair rp (XA, HL, DE, BC).

### 4.2 Table reference instructions

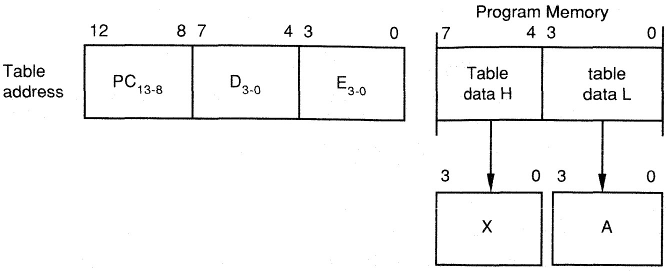
**MOVT XA, @PCDE**

Function: XA ↔ ROM ($PC_{13-8}$+DE)

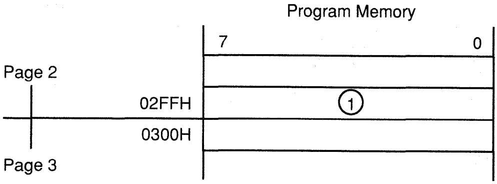
This instruction transfers the lower 4 bits of table data in the program memory location addressed by the higher 6 bits ($PC_{13-8}$) of the program counter (PC) plus DE register pair value to the A register, and the higher 4 bits of the same table data to the X register.

The higher 6 bits of the table address are determined by the program counter value at the time of the execution of this instruction.

Necessary data must be programmed on the table area beforehand by using an assembler's pseudoinstruction (DB instruction). The program counter value is not affected by the execution of this instruction. This instruction is effective to continuously refer to table data.



Note: The MOVT XA, @PCDE instruction normally references to the table data on the page where the instruction is located. If the instruction is located in address xxFFH, it references the table data on the next page.



If, for example, the MOVT XA, @PCDE instruction is placed in location ① in the above figure, it transfers the table data (specified by the value of register pair DE, not on page 2 but on page 3) to register pair XA.

Application example: Transfers the 8-byte data in program memory locations xxF8H to xxFFH to data memory locations 40 to 4FH.

```
SUB:     SEL     MB0
         MOV     HL, #40H;      HL ← 40H
         MOV     DE, #0F8H;     DE ← F8H
LOOP:    MOVT    XA, @PCDE;     XA ← table data
         MOV     @HL, XA;       (HL) ← XA
         INCS    L;             HL ← HL+2
         INCS    L
         INCS    E;             E ← E+1
         BR      LOOP
         RET
         ORG     xxF8H
         DB      xxH, xxH, ...;  table data
```

**MOVT XA, @PCXA**

Function: XA ← ROM ($PC_{13-8}$+XA)

This instruction transfers the lower 4 bits of the table data in the program memory location addressed by the upper 6 bits ($PC_{13-8}$) of the program counter (PC) plus the value of the register pair XA to the A register, and the higher 4 bits of the same table data to the X register.

The higher 6 bits of the table address are determined by the PC value at the time of the execution of this instruction. Necessary data must be programmed in the table area beforehand by using an assembler's directive (DB instruction). The program counter value is not affected by the execution of this instruction.

Note: If this instruction is placed in location xxFFH, the table data on the next program memory page will be transferred (similar to the MOVT XA, @PCDE instruction).

### 4.3 Operational instructions

**ADDS A, #n4**

Function: A ← A + n4; Skip if carry. n4 = $I_{3-0}$: 0H-FH

This instruction adds 4-bit immediate data n4 to the A register value (binary). If a carry is produced, the instruction following this instruction is skipped. The carry flag is not affected.

When this instruction is combined with the ADDC A, @HL or SUBC A, @HL instruction, it functions as a notation change instruction (see 1.4).

**ADDS A, @HL**

Function: A ← A + (HL); Skip if carry.

This instruction adds the contents of the data memory location addressed by the register pair HL to the A register valúe (binary). If a carry is produced, the instruction following this instruction is skipped. The carry flag is not affected.

**ADDC A, @HL**

Function: A, CY ← A + (HL) + CY

This instruction adds the contents (including the carry flag) of the data memory location addressed by the register pair HL to the A register value in binary. If a carry is produced, the carry flag is set; if not, the flag is reset. If a carry is produced when this instruction is followed by the ADDS A, #n4 instruction, the ADDS A, #n4 instruction will be skipped. If no carry is produced, the ADDS A, #n4 instruction will be executed with its skip function inhibited.

Therefore, these instruction are usable for notation change operation when combined (see 1.4).

**SUBS A, @HL**

Function: A ← A – (HL); Skip if borrow.

This instruction subtracts the contents of the data memory location addressed by the register pair HL from the A register value in binary and sets the result into the A register. If a borrow is produced, the instruction following this instruction is skipped. This instruction does not affect the carry flag.

**SUBC A, @HL**

Function: A, CY ← A – (HL) – CY

This instruction subtracts the contents of the data memory location addressed by register pair HL from the A register value including carry flag in binary and sets the result into the A register. If a borrow is produced, the carry flag is set; if not, the flag is reset.

If no borrow is produced when this instruction is followed by an ADDS A, #n4 instruction, the ADDS A, #n4 instruction will be skipped. If a borrow is produced, the ADDS A, #n4 instruction is executed with its skip function inhibited.

Therefore, these instructions can be used for notation change operation (see 1.4).

**AND A, #n4**

Function: A ← A $\wedge$ n4 n4 = $I_{3-0}$: 0H-FH

This instruction ANDs the A register value with 4-bit immediate data n4 and sets the result into the A register.

Application example: Sets the higher 2 bits of the accumulator to 0: AND A, #0011B

**AND A, @HL**

Function: A ← A $\wedge$ (HL)

This instruction ANDs the A register value with the contents of the data memory location addressed by the register pair HL and sets the result into the A register.

**OR A, #n4**

Function: $A \leftarrow A \vee n4 \quad n4 = I_{3\text{-}0}$: 0H-FH

This instruction ORs the A register value with 4-bit immediate data n4 and sets the result into the A register.

Application example: Sets the lower 3 bits of the accumulator to 1:
OR A, #0111B

**OR A, @HL**

Function: $A \leftarrow A \vee (HL)$

This instruction ORs the A register value with the contents of the data memory location addressed by the register pair HL and sets the result into the A register.

**XOR A, #n4**

Function: $A \leftarrow A \veebar n4 \quad n4 = I_{3\text{-}0}$: 0H-FH

This instruction XORs the A register value with 4-bit immediate data n4 and sets the result into the A register.

Application example: Inverts the MSB of the A register:
XOR A, #1000B

**XOR A, @HL**

Function: $A \leftarrow A \veebar (HL)$

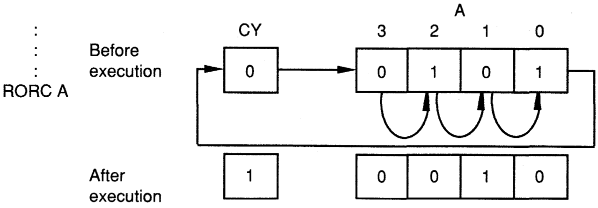
This instruction XORs the A register value with the contents of the data memory location addressed by the register pair HL and sets the result into the A register.

### 4.4 Accumulator manipulation instructions

**RORC A**

Function: $CY \leftarrow A_0, A_{n-1} \leftarrow An, A_3 \leftarrow CY$ (n = 1-3)

This instruction rotates the contents of the A register (4-bit accumulator) right by 1 bit through the carry flag.



**NOT A**

Function: $A \leftarrow \overline{A}$

This instruction complements the A register (4-bit accumulator) (invert each bit).

### 4.5 Increment/decrement instructions

**INCS reg**

Function: reg ← reg + 1; Skip if reg = 0

This instruction increments the value of the register reg (X, A, H, L, D, E, B, C) by one. If reg = 0 when incremented, the instruction following this instruction will be skipped.

**INCS @HL**

Function: (HL)← (HL + 1; Skip if (HL) = 0

This instruction increments the contents of the data memory location addressed by the register pair HL. If the contents of the data memory location become zero when incremented, the instruction following this instruction will be skipped.

**INCS mem**

Function: (mem)← (mem) + 1; Skip if (mem) = 0, (mem) = $D_{7-0}$: 00H-FFH

This instruction increments the contents of the data memory location addressed by the 8-bit immediate data mem. If the contents of the data memory location become zero when incremented, the instruction following this instruction will be skipped.

**DECS reg**

Function: reg← reg – 1; Skip if reg = FH.

This instruction decrements the value of the register reg (X, A, H, L, D, E, B, C). If reg = FH when decremented, the instruction following this instruction will be skipped.

### 4.6 Compare instructions

**SKE reg, #n4**

Function: Skip if reg = n4 n4 = $I_{3-0}$: 0H-FH

This instruction skips the instruction following it if the value of the register reg (X, A, H, L, D, E, B, C) is equal to the 4-bit immediate data n4.

**SKE @HL, #n4**

Function: Skip if (HL) = n4 n4 = $I_{3-0}$: 0H-FH

This instruction skips the instruction following it if the value of the data memory location addressed by the register pair HL is equal to the 4-bit immediate data n4.

**SKE A, @HL**

Function: Skip if A = (HL)

This instruction skips the instruction following it if the A register value is equal to the value of the data memory location addressed by the register pair HL.

**SKE A, reg**

Function: Skip if A = reg

This instruction skips the instruction following it if the A register value is equal to the value of the register reg (X, A, H, L, D, E, B, C).

### 4.7 Carry flag manipulation instructions

**SET1 CY**

Function: CY← 1

This instruction sets the carry flag.

**CLR1 CY**

Function: CY← 0

This instruction clears the carry flag.

**SKT CY**

Function: Skip if CY ←1

This instruction skips the instruction following it if the carry flag is set to 1.

**NOT1 CY**
Function: $CY \leftarrow \overline{CY}$

This instruction reverses the carry flag status, from 0 to 1 or vice versa.

### 4.8 Memory bit manipulation instructions

**SET1 mem. bit**
Function: (mem. bit) ← 1 mem = $D_{7-0}$: 00H-FFH, bit = $B_{1-0}$: 0-3

This instruction sets the data memory bit specified by the 2-bit immediate data "bit" at the location addressed by the 8-bit immediate data "mem".

**SET1 fmem. bit**
**SET1 pmem, @L**
**SET1 @H + mem. bit**
Function: (Bit specified by operand) ← 1

These instructions set the data memory bit specified by the bit manipulation addressing (fmem. bit, pmem. @L, @H + mem. bit).

**CLR1 mem. bit**
Function: (mem. bit) ← 0 mem 0 $D_{7-0}$: 00H-FFH, bit = $B_{1-0}$: 0-3

This instruction clears the data memory bit specified by the 2-bit immediate data "bit" at the location specified by the 8-bit immediate data "mem".

**CLR1 fmem. bit**
**CLR1 pmem. @L**
**CLR1 @H + mem. bit**
Function: (Bit specified by operand) ← 0

These instruction s clear the data memory bit specified by the bit manipulation addressing (fmem. bit, pmem. @L, @H + mem. bit).

**SKT mem. bit**
Function: Skip if (mem, bit) = 1 mem = $D_{7-0}$: 00H-FFH, bit = $B_{1-0}$: 0-3

This instruction skips the instruction following it if the data memory bit specified by the 2-bit immediate data "bit" at the location specified by the 8-bit immediate data "mem" is equal to 1.

**SKT fmem. bit**
**SKT pmem. @L**
**SKT @H + mem. bit**
Function: Skip if (bit specified by operand) = 1

These instructions skip the instruction following it if the data memory bit specified by the bit manipulation addressing (fmem. bit, pmem. @L, @H + mem. bit) is equal to 1.

**SKF mem. bit**
Function: Skip if (mem. bit) = 0 mem = $D_{7-0}$: 00H-FFH, bit = $B_{1-0}$: 0-3

This instruction skips the instruction following it if the data memory bit specified by the 2-bit immediate data "bit" at the location specified by the 8-bit immediate data "mem" is equal to 0.

**SKF fmem. bit**
**SKF pmem. @L**
**SKF @H + mem. bit**
Function: Skip if (bit specified by operand) = 0

These instructions skip the instruction following it if the data memory bit specified by the bit manipulation addressing (fmem. bit, pmem. @L, @H + mem. bit) is equal to 0.

**SKTCLR fmem. bit**
**SKTCLR pmem. @L**
**SKTCLR @H + mem. bit**
Function: Skip if (bit specified by operand) = 1, then clear it

These instructions skip the instruction following it if the data memory bit specified by the bit manipulation addressing (fmem. bit, pmem. @L, @H + mem. bit) is equal to 1, then clear the bit to 0.

**AND1 CY, fmem. bit**
**AND1 CY, pmem. @L**
**AND1 CY, @H + mem. bit**
Function: $CY \leftarrow CY \wedge$ (bit specified by operand)

These instructions AND the carry flag value with the data memory bit specified by the bit manipulation addressing (fmem. bit, pmem. @L, @H + mem. bit) and set the result into the carry flag.

**OR1 CY, fmem. bit**
**OR1 CY, pmem. @L**
**OR1 CY, @H + mem. bit**
Function: $CY \leftarrow CY \vee$ (bit specified by operand)

These instructions OR the carry flag value with the data memory bit specified by the bit manipulation addressing (fmem. bit, pmem. @L, @H + mem. bit) and set the result into the carry flag.

**XOR1 CY, fmem. bit**
**XOR1 CY, pmem. @L**
**XOR1 CY, @H + mem. bit**
Function: $CY \leftarrow CY \veebar$ (bit specified by operand)

These instructions XOR the carry flag value with the data memory bit specified by the bit manipulation addressing (fmem. bit, pmem. @L, @H + mem. bit) and set the result into the carry flag.

### 4.9 Branch instruction

**BR addr**
Function: $PC_{13-0} \leftarrow$ addr addr = 0000H-3F7FH (depends upon available ROM of the selected device)

This instruction causes control to branch to the address specified by the 14-bit immediate data, addr. It is an assembler's pseudoinstruction and is automatically replaced with an appropriate instruction out of the BR !addr, BRCB !caddr and BR $addr instruction when assembling.

**BR !addr**
Function: $PC_{13-0} \leftarrow$ addr addr = 0000H-3F7FH (depends upon available ROM of the selected device)

This instruction transfers the 14-bit immediate data addr to the program counter (PC) to make branch to the address specified by the PC value.

Branching is possible to any location on the program memory space.

**BR $addr**
Function: PC addr$\leftarrow$ addr = (PC-15) to (PC-1), (PC+2) to (PC+16)

This instruction causes control to branch to an address which is –15 to +16 with respect to the current address. It is not affected by page or block boundaries.

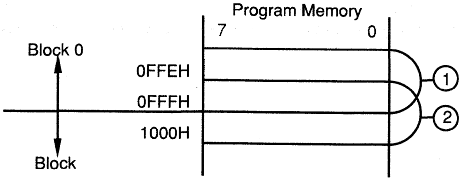
**BRCB !caddr**

Function: $PC_{11-0} \leftarrow$ caddr  caddr = 0000H-0FFFH

This instruction causes control to branch to the address which is obtained by replacing the lower address which is obtained by replacing the lower 12 bits ($PC_{11-0}$) of the program counter with the 12-bit immediate data caddr ($A_{11-0}$).

$PC_{13, 12}$ can not be modified and this branch results in an in-block branch.

Note: The BRCB !caddr instruction normally causes branching within the block where it is located. If the first byte of the instruction is located in address 0FFEH or 0FFFH, it causes branching to block 1, not to block 0.



If the BRCB !addr instruction is located in (1) or (2) of the above figure, it causes branching to block 1, not to block 0.

**TBR addr**

Function: This is an assembler directive for a table definition of the GETI instruction.

For details, see µCOM-75X Family Relocatable Assembler Package Operation Manual which is part of the µCOM-75X Family 4Bit CMOS Microcomputer Development Tool book.

### 4.10 Subroutine stack control instruction

**CALL !addr**

Function: (SP-1) $\leftarrow$ $PC_{7-4}$, (SP-2) $\leftarrow$ $PC_{3-0}$, (PC-3) $\leftarrow$ MBE, 0, $PC_{13, 12}$, (SP-4) $\leftarrow$ $PC_{11-8}$, $PC_{13} \leftarrow$ addr, SP $\leftarrow$ SP-4, addr = 0000H-3F7FH (depends on the selcted device type.

This instruction saves the values of the program counter ($PC_{13-0}$: return address) and MBE to the data memory locations (stack) addressed by the stack pointer (SP), decrements the SP, and causes branch to the address specified by the 14-bit immediate data addr. Branching is possible to any location on the program memory space.

**CALLF !faddr**

Function: (SP-1) $\leftarrow$ $PC_{7-4}$, (SP-2) $\leftarrow$ $PC_{3-0}$, (PC-3) $\leftarrow$ MBE, 0, $PC_{13, 12}$, (SP-4) $\leftarrow$ $PC_{11-8}$, SP $\leftarrow$ SP-4, PC $\leftarrow$ 000, $A_{10-0}$, faddr = $A_{10-0}$: 000H-7FFH

This instruction saves the values of the program counter (PC return address) and MBE to the data memory locations (stack) addressed by the stack pointer (SP), decrements the SP, and causes branching to the address specified by the 11-bit immediate data faddr. The program memory area which can be called is limited to addresses 0000 through 07FFH (0-2047).

**TCALL addr**

Function: This is an assembler directive for a table defination of the GETI instruction.

It is used for replacing the 3-byte CALL instruction with a GETI instruction.

Write a call address of the 3-byte CALL instruction for an addr of the TCALL.

For details, see µCOM-75X Family 4-Bit CMOS Microcomputer Development Tool book.

**RET**

Function: $PC_{11-8} \leftarrow (SP)$, MBE, $PC_{13, 12} \leftarrow (SP+1)$, $PC_{3-0} \leftarrow (SP+2)$ $PC_{7-4} \leftarrow (SP+3)$, $SP \leftarrow SP+4$

This instruction restores the contents of the data memory (stack) locations addressed by the stack pointer (SP) into the program counter (PC) and memory bank enable flag (MBE), then increments the SP.

Note: The program status word (PSW) is not restored with the exception of the MBE.

**RETS**

Function: $PC_{11-8} \leftarrow (SP)$, MBE, $PC_{13, 12} \leftarrow (SP+1)$, $PC_{3-0} \leftarrow (SP+2)$ $PC_{7-4} \leftarrow (SP+3)$, $SP \leftarrow SP+4$
Then skip unconditionally.

This instruction restores the contents of the data memory (stack) locations addressed by the stack pointer (SP) into the program counter (PC) and memory bank enable flag (MBE), increments the SP value, and makes an unconditional skip.

Note: The program status word (PSW) is not restored with the exception of the MBE.

**RETI**

Function: $PC_{11-8} \leftarrow (SP)$, $PC_{13, 12} \leftarrow (SP+1)$, $PC_{3-0} \leftarrow (SP+2)$ $PC_{7-4} \leftarrow (SP+3)$, $PSW_L \leftarrow (SP+4)$, $PSW_H \leftarrow (SP+5)$, $SP \leftarrow SP+6$

This instruction restores the contents of the data memory locations (stack) addressed by the stack pointer (SP) into the program counter (PC) and program status word (PSW), then increments the SP value. It is used to return from an interrupt service routine.

**PUSH rp**

Function: $(SP-1) \leftarrow rp_H$, $(SP-2) \leftarrow rp_L$, $SP \leftarrow SP-2$

This instruction saves the contents of the register pair rp (XA, HL, DE, BC) to the data memory locations (stack) addressed by the stack pointer (SP), then decrements the SP value. The higher 4 bits ($rp_H$: X, H, D, B) of the register pair are saved to the stack addressed by (SP-1), while the lower 4 bits ($rp_L$: A, L, E, C) of it is saved to the stack addressed by (SP-2).

**PUSH BS**

Function: $(SP-1) \leftarrow$ MBS, $(SP-2) \leftarrow 0$, $SP \leftarrow SP-2)$

This instruction saves the contents of the memory bank select register (MBS) to the data memory locations (stack) addressed by the stack pointer (SP), then decrements then SP.

**POP rp**

Function: $rp_L \leftarrow (SP)$, $rp_H \leftarrow (SP+1)$, $SP \leftarrow SP+2)$

This instruction restores the contents of the data memory locations (stack) address by the stack pointer (SP) into the register pair rp (XA, HL, DE, BC) then increments the SP. The contents of the (SP) are restored into the lower 4 bits of the register pair ($rp_L$: A, L, E, C) while those of the (SP+1) are restored into the higher 4 bits of the register pair ($rp_H$: X, H, D, B).

**POP BS**

Function: $MBS \leftarrow (SP+1)$, $SP \leftarrow SP+2$

This instruction restores the contents of the data memory locations (stack) address by the stack pointer (SP) to the memory bank select register (MBS), then increments the SP.

### 4.11 Interrupt control instructions

**EI**

Function: $IME \leftarrow 1$

This instruction sets the interrupt enable master flag to 1 to enable interrupt. Whether a specific interrupt is acknowledged or not is determined by individual interrupt enable flag.

**EI IExxx**

Function: IExxx ← 1 xxx = $N_{5-0}$

This instruction sets the interrupt flag (IExxx) to 1 to enable the corresponding interrupt request.

**DI**

Function: IME ← 0

This instruction resets the interrupt enable master flag to 0 to disable all interrupts regardless of the status of their individual interrupt enable flags.

**DI IExxx**

Function: IExxx ← 0 xxx = $N_{5-0}$

This instruction resets the interrupt flag (IExxx) to 0 to disable the corresponding interrupt request.

### 4.12 Input/output instructions

**IN A, PORTn**

Function: A ← PORTn

This instruction transfers the contents of the port specified by PORTn to the A register.

Note: When executing this instruction, the MBE = 0 or (MBE = 1 and MBS = 15) must be set. When the I/O mode specified, the output latch data (output mode) or pin data (input mode) is loaded to the A register.

**IN XA, PORTn**

Function: A ← PORTn, X ← PORTn+1

This instruction transfers the contents of the port specified by PORTn and the contents of the next port to the X register.

Note: When executing this instruction, the MBE = 0 or (MBE = 1 and MBS = 15) must be set beforehand. Only the values for "n" are allowed where input port instruction is enabled by the list of I/O pin manipulation instructions which are part of the chapter regarding on chip peripherals. Note that "n" differs among the device types!
When the I/O mode is specified, the output latch data (output mode or pin data (input mode) is loaded to the XA registers.

**OUT PORTn, A**

Function: PORTn ← A

This instruction transfers the A register value to the output latch in the port specified by PORTn.

Note: When executing this instruction, the MBE = 0 or (MBE = 1 and MBS = 15) must be set beforehand. Only the values for "n" are allowed where output port instruction is enabled by the list of I/O pin manipulation instructions which are part of the chapter regarding on chip peripherals. Note that "n" differs among the device types!
When the I/O mode is specified, the output latch data (output mode or pin data (input mode) is loaded to the A register.

**OUT PORTn, XA**

Function: PORTn ← A PORTn + 1 ← X

This instruction transfers the A register value to the output latch in the port specified by PORTn, and the X register value to the output latch on the next port.

Note: When executing this instruction, MBE = 0 or (MBE = 1 and MBS = 15) must be set beforehand. It depends on the physical availability of the port on the selected device type if this instruction is executable. Therefore check the chapter regarding the on chip peripherals first.
Note that "n" differs among the device types!

### 4.13 CPU control instructions

**HALT**

Function: PCC.2 ← 1

This instruction selects the HALT mode (sets bit 2 of the processor clock control register).

Note: The HALT instruction must be followed by a NOP instruction.

**STOP**

Function: PCC. 3 ← 1

This instruction sets the STOP mode (sets bit 3 of the processor clock control register).

Note: The STOP instruction must be followed by a NOP instruction.

**NOP**

Function: Causes no operation to consume an idle machine cycle.

### 4.14 Special instructions

**SEL MBn**

Function: MBS ← n n = $N_{3\text{-}0}$: 0, 1, 15

This instruction transfers the 4-bit immediate data n to memory bank select register (MBS).

Number 0, 1, or 15 is allowed for n presently.

**GETI taddr**

Function: taddr = $T_{5\text{-}0}$, 0: 20H-7FH

- For TBR instruction $PC_{13\text{-}0} \leftarrow (taddr)_{5\text{-}0} + (taddr + 1)$
- For TCALL instruction (SP-4) (SP-1) (SP-2) ← $PC_{11\text{-}0}$ (SP-3) ← (MBE, 0, PC13, 12)
  $PC_{13\text{-}0} \leftarrow (taddr)_{5\text{-}0} + (taddr + 1)$ SP ← SP-4
- For other than TBR, TCALL instruction Executes the instruction whose instruction code is (taddr) (taddr + 1).

This instruction references the 2-byte data located in the program memory locations specified by taddr and (taddr + 1), then executes it as an instruction.

The reference table area is 0020 to 007FH, in which data must be written beforehand. For 1- or 2-byte instruction, mnemonics should be written with a pseudoinstruction (TCALL, TBR).

*Only even-number addresses may be specified for taddr.*

Note: For 2-byte instructions, only 2-machine cycle instructions can be set on the reference table (Except BRCB and CALLF instructions). When setting two 1-byte instructions on the table, their combinations are limited to those listed in the following table:

| 1st byte instruction | 2nd byte instruction |
|---|---|
| MOV A, @HL<br>MOV @HL, A<br>XCH A, @HL | INCS L<br>DECS L<br>INCS H<br>DECS H |
| MOV A, @DE<br>XCH A, @DE | INCS E<br>DECS E<br>INCS D<br>DECS D |
| MOV A, @DL<br>XCH A, @DL | INCS L<br>DECS L<br>INCS D<br>DECS D |

Since the PC is not incremented during GETI instruction execution, execution resumes in the address next to the GETI instruction address when the execution of the reference instruction is completed. If the instruction precending the GETI instruction has skip feature, the GETI instruction is skipped in the same way as other 1-byte instructions. If the instruction referenced with the GETI instruction has a skip function, the instruction that follows the GETI instruction will be skipped.

If an instruction having the string effect is referenced with the GETI instruction, the following will occur:

- If the GETI instruction is preceded by an instruction belonging to the same string effect group, the execution of the GETI instruction cancels the effect, and the instruction which was referenced is not skipped.
- If the GETI instruction is followed by an instruction belonging to the same string effect group, the string effect of the instruction which was referenced remains valid, and the instruction following it will be skipped.

Application example:

```
Replace   MOV HL, #00H
          MOV XA, #FFH        with the GETI
          CALL SUB1           instruction
          BR SUB2

 CODE0    CSEG    IENT
  HL00:   MOV     HL, #00H
  XAFF:   MOV     XA, #FFH
CSUB1:    TCALL   SUB1
BSUB2:    TBR     SUB2
                  :
                  :
          GETI    HL00        ; MOV HL, #00H
                  :
                  :
          GETI    BSUB2       ; BR SUB2
                  :
                  :
          GETI    CSUB1       ; CALL SUB1
                  :
                  :
          GETI    XAFF        ;MOV XA, #0FFH
```

## CHAPTER 6 μPD75402A LOW END 4 BIT MICROCOMPUTER

### 1. OVERVIEW

The μPD75402A, μPD75P402 is a CMOS 4-bit single chip computer adopting the μCOM-75X architecture. The microcomputer which enables high speed processing and has internal NEC standard serial bus interface is suitable for a slave microcomputer in the multiprocessor system configuration in which the μCOM-75X or μCOM-78K series is used as the host microcomputer.

The host microcomputer load can be relieved by using the μPD75402A, μPD75P402 as a slave microcomputer for distributed processing. It is applicable to slave processing such as key input control and LED display conntrol or remote control transfer control of the following:

- Facsimile
- PPC
- Printer
- Electronic cash register
- Video tape recorder
- Remote control commander

The μPD75P402 is QTOP$^{TM}$ microcomputer provided by replacing μPD75402A internal mask ROM with one time PROM. It is compatible with the μPD75402A except for program memory or mask option. Pin connection at write is the same as that of standard EPROM μPD27C256A and the μPD75P402 has the same write characteristic as the μPD27C256A. Thus, the μPD75P402 can be written directly by using a general purpose PROM writer.

It is suitable for test production at system development or few-of-a-kind parts production.

| Product name | Program memory | Data memory |
|---|---|---|
| μPD75402A | 1920 x 8 (mask ROM) | 64 x 4 (RAM) |
| μPD75P402 | 1920 x 8 (one time PROM) | 64 x 4 (RAM) |

Remarks: This manual explains the μPD75402A as representative product unless otherwise noted. If you use the manual for the μPD75P402, replace the μPD75402A with the μPD75P402 in reading.

QTOP is a trademark of NEC

### 1.1 Function Outline

| Item | | Description |
|---|---|---|
| Number of basic instructions | | 37 |
| Minimum instruction execution time | | • 0,95μs, 1.91 or 15.3μs (at 4.19 MHz) Either can be selected. |
| Internal memory | Program memory | 1920 x 8 bits (mask ROM (μPD75402A) or one time PROM (μPD75P402)) |
| | Data memory | 64 x 4 bits (RAM) |
| General purpose register | | 4 bits x 4 or 8 bits x 2 (memory mapping) |
| Accumulator | | Three accumulators conforming to handled data length<br>• 1-bit accumulator (CY), 4-bit accumulator (A), and 8-bit accumulator (XA) |
| I/O line | | 22 lines in total<br>• CMOS input ports: 6 lines<br>• CMOS input/output ports (eight pins can directly drive LED's): 12 lines<br>• N-ch open drain input/output ports (which can directly drive LED's): 4 lines |

### 1.1 Function Outline (cont'd)

| Item | Description |
|---|---|
| Pull-up resistor | • 16 pull-up resistor internal control can be performed by using software<br>• Four pull-up resistor internal control can be performed by using mask option (μPD75402A only) |
| Clock output | • 1.05 MHz, 524 or 65.5 kHz (at 4.19 MHz)<br>• Applicable to remote control output |
| Timer/counter | 8-bit basic interval timer<br>• Reference time generation (1.91 or 31.3 ms at 4.19 MHz)<br>• Applicable to watchdog timer |
| Serial interface | • Eight bits<br>• Two transfer modes (clock synchronization 3-wire mode and SBI mode) |
| Interrupt | • Three vectored interrupts (one external and two internal)<br>• One external testable input |
| Standby | STOP or HALT mode |
| Instruction set | • Bit handling instruction (set, clear, test, and Boolean operations)<br>• 1-byte relative branch instructions<br>• 4-bit operation instructions (addition, Boolean operations, and comparison)<br>• 4- and 8-bit data transfer instructions |
| Package | • 28-pin plastic DIP<br>• 28-pin plastic shrink DIP<br>• 44-pin plastic QFP (normal bend) |

### 1.2 Order Information

| order nome | Package | Program memory |
|---|---|---|
| μPD75402AC-XXX | 28-pin plastic DIP | Mask ROM |
| μPD75402ACT-XXX | 28-pin plastic shrink DIP | |
| μPD75402AGB-XXX-3B4 | 44-pin plastic QFP | |
| μPD75P402C | 28-pin plastic DIP | One time PROM |
| μPD75P402CT | 28-pin plastic shrink DIP | |
| μPD75P402GB-3B4 | 44-pin plastic QFP | |

XXX: ROM code number

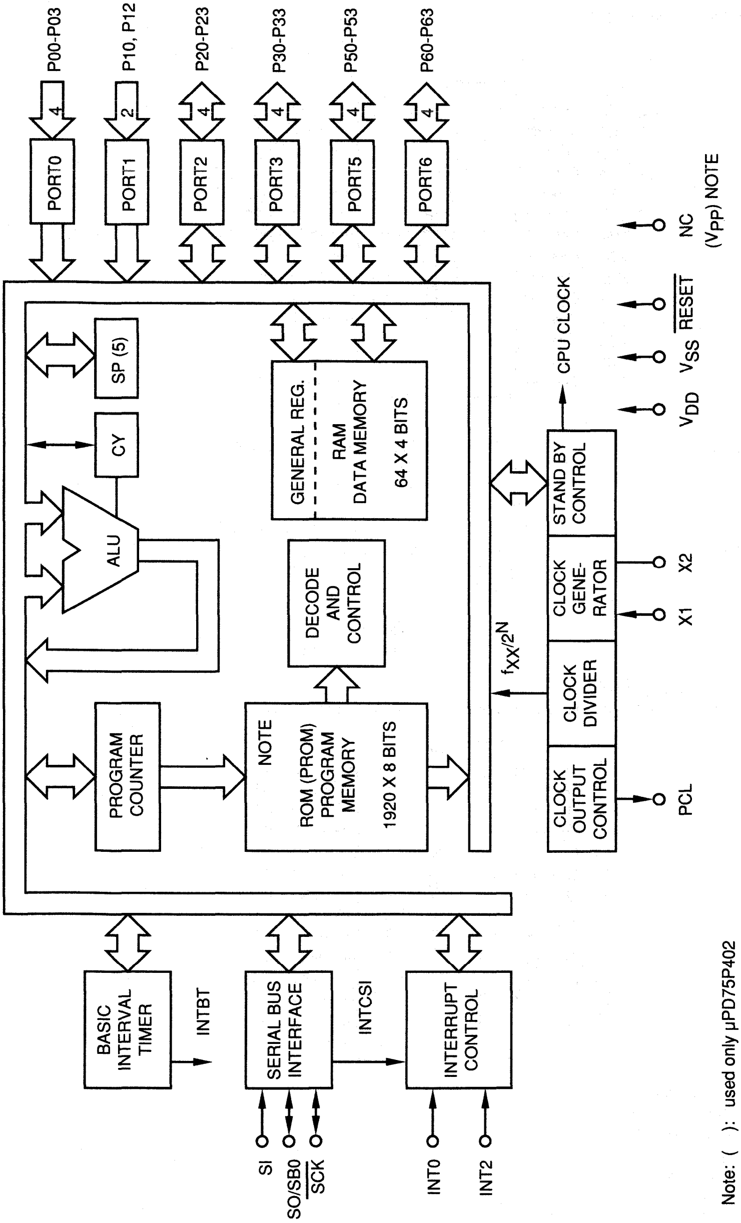
### 1.3 Differences between μPD75402A and μPD75P402

Table 1.3-1 lists the differences between the μPD75402A and μPD75P402. They have the same function except listed in the table and are pin-compatible with each other.

**Table 1.3-1 Differences between μPD75402A and μPD75P402**

| Item | μPD75402A | μPD75P402 |
|---|---|---|
| Program memory | • Mask ROM<br>• 1920 x 8 bits | • One time PROM<br>• 1920 x 8 bits |
| Port 5 pull-up resistor | Internal pull up resistor is enabled by using mask option. | None |
| Pin connection | One-time PROM write/verify/read pin function (PROM mode) is added to the μPD75P402 as compared with the μPD75402A. (See Chapter 2) | |
| Operation supply voltage range | 2.7 V to 6.0 V | 5.0 V ± 10% |
| Operating temperature range | -40 to +85°C | -10 to +70°C |
| Package | • 28-pin plastic DIP<br>• 28-pin plastic shrink DIP<br>• 44-pin plastic QFP | |

## 1.4 Block Diagram

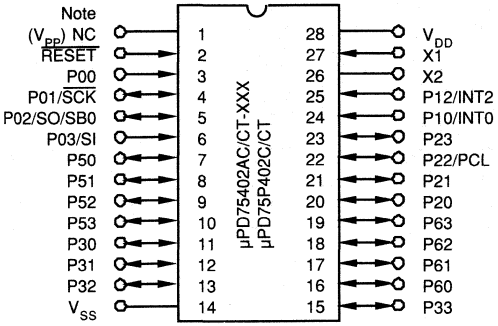


Note: ( ): used only µPD75P402

### 1.5 Pin Connections

1.5.1 28-pin plastic DIP/shrink DIP

(1) Normal operation mode

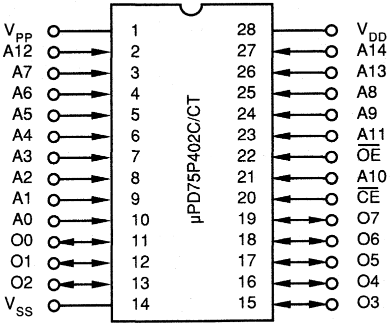


| | |
|---|---|
| P00-P03 : | Port 0 |
| P10-P13 : | Port 1 |
| P20-P23 : | Port 2 |
| P30-P33 : | Port 3 |
| P50-P53 : | Port 5 |
| P60-P63 : | Port 6 |

| | | |
|---|---|---|
| SCK | : | Serial clock input/output |
| SO/SB0 | : | Serial input/output |
| SI | : | Serial input |
| PCL | : | Clock output |
| INT0 | : | External vectored interrupt |
| INT2 | : | External test input |
| X1, X2 | : | Main system clock |
| RESET | : | Reset input |
| $V_{DD}$ | : | Power supply |
| $V_{SS}$ | : | Ground |
| $V_{PP}$ | : | Setting the GND from external |

Note (    ): used by only μPD75P402.
To share printed circuit board with μPD75P402 in the μPD75402A, set the NC pin to the GND potential.

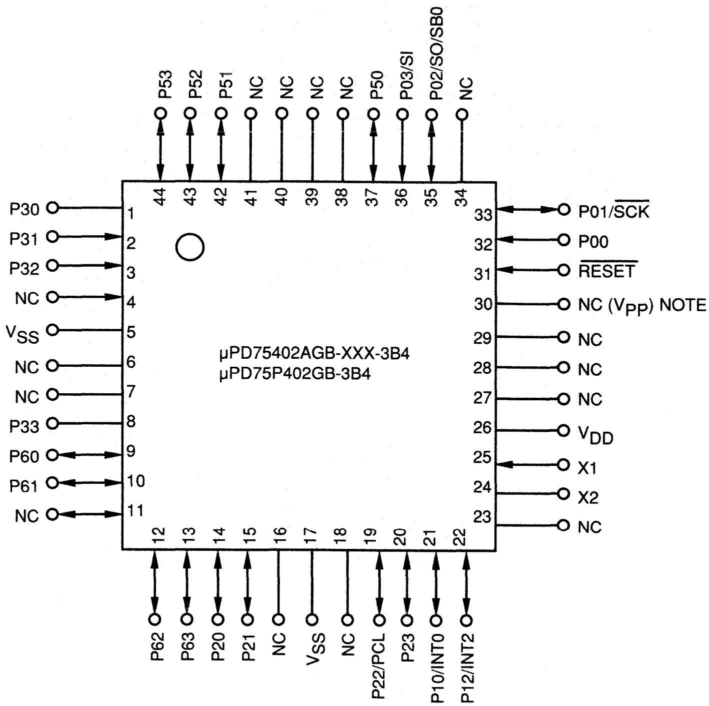
(2) PROM mode



| | | |
|---|---|---|
| A0-A14 | : | Address input |
| O0-O7 | : | Data input/output |
| CE | : | Chip enable input |
| OE | : | Output enable input |
| $V_{DD}$ | : | Power supply |
| $V_{PP}$ | : | Program power supply |
| $V_{SS}$ | : | Ground |

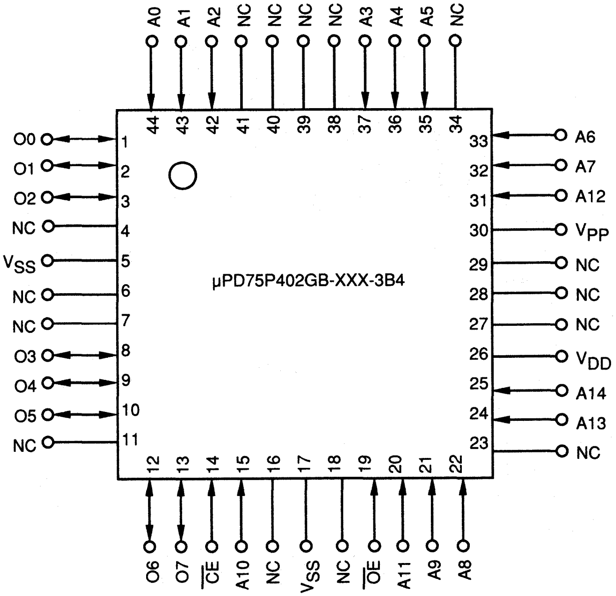
1.5.2 44-pin plastic QFP (bent lead)

(1) Normal operation mode



Note: ( ): Used by only μPD75P402.
To share printed circuit board with μPD75P402 in the μPD75402, set the NC pin which is applicable to the $V_{PP}$ in the μPD75402, to GND potential.

(2) PROM mode


μPD75P402GB-XXX-3B4
1 O0
2 O1
3 O2
4 NC
5 $V_{SS}$
6 NC
7 NC
8 O3
9 O4
10 O5
11 NC
12 O6
13 O7
14 $\overline{CE}$
15 A10
16 NC
17 $V_{SS}$
18 NC
19 $\overline{OE}$
20 A11
21 A9
22 A8
23 NC
24 A13
25 A14
26 $V_{DD}$
27 NC
28 NC
29 NC
30 $V_{PP}$
31 A12
32 A7
33 A6
34 NC
35 A5
36 A4
37 A3
38 NC
39 NC
40 NC
41 NC
42 A2
43 A1
44 A0

## 2. PIN FUNCTION

The μPD75402A operates in the normal operation mode of the pin function.
The μPD75P402 operates in either the normal operation mode (μPD75402A mode) or PROM mode of the pin function.
The mode is selected according to the $V_{PP}$ pin level as listed below:

| $V_{PP}$ | Operation mode | |
|---|---|---|
| Low (GND potential) | Normal operation mode | |
| High (+5 V) | PROM mode | PROM read mode |
| High (+12.5 V) | | PROM write/verify mode |

### 2.1 Normal Operation Mode

2.1.1 P00-P03 (port 0) – input also used for $\overline{\text{SCK}}$, SO/SB0, and SI
P10 and P12 (port 1) – input also used for INT0 and INT2
P00-P03 are 4-bit input port (port 0) input pins. P10 and P12 are 2-bit input pins.
The input port pins of ports 0 and 1 are also used for the control signal pins listed in Table 2.1-1. The state of each pin of ports 0 and 1 can always be input regardless of $\overline{\text{SCK}}$, SO/SB0, SI, INT0, or INT2 pin operation.
Every pin of ports 0 and 1 is Schmitt trigger input to prevent noise from causing malfunction. In addition, P10 has an internal noise eliminator using a sampling clock and P12 has an internal noise eliminator using analog delay.
Internal pull-up resistor can be specified for port 0 in 3-bit units (P01-P03) and port 1 bitwise (P12 only) by using the pull-up resistor specification register (POGA). Internal pull-up resistor cannot be contained in P00 or P10.
When $\overline{\text{RESET}}$ is input, every pin is placed in the input port mode.

**Table 2.1-1 Port 0 and 1 Pins also Used for:**

| Port 0 | Also used for | Port 1 | Also used for |
|---|---|---|---|
| P00 | — | P10 | INT0 |
| P01 | $\overline{\text{SCK}}$ | P12 | INT2 |
| P02 | SO/SB0 | | |
| P03 | SI | | |

2.1.2 P20-P23 (Port 2) – 3-state input/output also used for PCL
P30-P33 (Port 3) – 3-state input/output
P50-P53 (Port 5) – N-ch open drain medium voltage (10 V) input/output
P60-P63 (Port 6) – 3-state input/output
P20-P23, P30-P33, P50-P53, and P60-P63 are input/output pins of 4-bit input/output ports with output latches (ports 2, 3, 5, and 6). Port 2 contains P22 which is also used for the programmable clock output (PCL) function. Port 5 is N-ch open drain medium voltage (10 V) output.
Input or output mode can be selected for port 3 bitwise by using the port mode register (PMGA). Input or output mode can be selected for ports 2, 3, and 6 in 4-bit units by using the port mode registers (PMGA and PMGB).
Internal pull-up resistor can be specified for ports 2, 3, and 6 in 4-bit units by using software with the pull-up resistor specification register (POGA).
Internal pull-up resistor can be specified for μPD75402A port 5 bitwise by using mask option. μPD75P402 port 5 cannot contain internal pull-up resistor.
Ports 3, 5, and 6 output high current and can directly drive LED's.
When $\overline{\text{RESET}}$ is input , ports 2, 3, and 6 are used as input ports (output high impedance). Port 5 goes high (when internal pull-up resistor is contained) or is placed in high impedance.

2.1.3 $\overline{\text{SCK}}$, SO/SB0, and SI – 3-state input/output also used for port 0
$\overline{\text{SCK}}$, SO/SB0, and SI – 3-state input/output pins for serial interface. The pins operate according to how the serial operation mode register (CSIM) is set.
Every pin is Schmitt trigger input.
When $\overline{\text{RESET}}$ is input, the serial interface stops and the pins are used for input port (port 0).

2.1.4 INT0 – input also used for port 1

INT0 is an external interrupt request input pin. The detection edge can be selected among rising edge, falling edge, and both rising and falling edges by using the external interrupt mode register (IM0).
INT0 is Schmitt trigger input and contains an internal noise eliminator using sampling clocks.

2.1.5 INT2 – input also used for port 1

INT2 is an external test input pin. The detection edge is fixed to the rising edge. INT2 is Schmitt trigger input and contains an internal noise eliminator using analog delay.
INT2 is asynchronous input and acknowledged when a signal having a given high level width is input independently of the CPU operation clock.

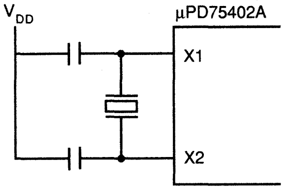
2.1.6 PCL – output also used for port 2

PCL is a programmable clock output pin. It is used to supply a clock to peripheral LSI. PLC output is applicable to carrier signal for remote control.
When RESET is input, the clock output function stops and PCL is used for input port (P22).

2.1.7 X1 and X2 (crystal)

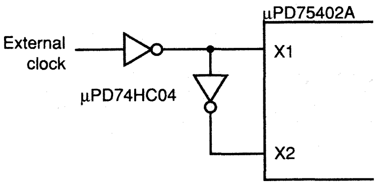
X1 and X2 are internal clock oscillation crystal or ceramic resonator connection pins.
Extenal clock can also be supplied.

(a) Crystal or ceramic oscillation



Crystal or ceramic resonator
(Standard 4.194304 MHz)

(b) External clock



2.1.8 RESET (reset)

RESET is a system reset input pin (active low). It is Schmitt trigger input and contains an internal noise eliminator using analog delay.
RESET is asynchronous input and acknowledged when a signal having a given low level width is input independently of the CPU operation clock. It takes precedence over every other operation and the system is reset.

2.1.9 $V_{DD}$

$V_{DD}$ is a positive power supply pin.

2.1.10 $V_{SS}$

$V_{SS}$ is a GND potential pin.

**2.2 PROM Mode**

The PROM mode can be selected only for the µPD75P402.

2.2.1 A0-A14 (address) – input

A0-A14 are 15-bit address input pins in the PROM write/verify or read mode. Since µPD75P402 internal PROM is 2K bytes, the low-order 11 bits (A0-A10) are used for addressing. Fix A11-A14 to high level.

2.2.2 O0-O7 (data – input/output

O0-O7 are 8-bit data input/output pins in the PROM write/verify or read mode.

2.2.3 CE (chip enable) – input

CE is a chip enable signal input pin.

2.2.4 $\overline{\text{OE}}$ (output enable) – input
OE is an output enable signal input pin.

2.2.5 $V_{PP}$
$V_{PP}$ is a high-voltage apply pin in the PROM write/verify mode. Be sure to connect the pin to $V_{SS}$ during the normal operation.

2.2.6 $V_{DD}$
$V_{DD}$ is a power supply voltage apply pin.

2.2.7 $V_{SS}$
$V_{SS}$ is a GND potential pin.

### 2.3 Pin Input and Output Circuits

The pin input and output circuits are shown schematically.

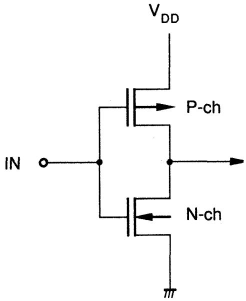
**Table 2.3-1 Pin Input/Output Types**

| Pin | Input/output type | |
|---|---|---|
| | µPD75402A | µPD75P402 |
| P00 | Ⓑ | |
| P01/SCK | Ⓕ–A | |
| P02/SO/SB0 | Ⓕ–B | |
| P03/SI | Ⓑ–C | |
| P10/INT0 | Ⓑ | |
| P12/INT2 | Ⓑ–C | |
| P20, P21, P23 | E–B | |
| P22/PCL | | |
| P30-P33 | E–B | |
| P50-P53 | M | M–A |
| P60-P63 | E–B | |
| $\overline{\text{RESET}}$ | Ⓑ | |

Remarks: ○: Schmitt trigger input

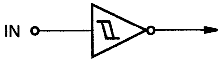
The μPD75402A pin input / output circuits are shown in schematic drawings.

(1) Type A (for Type E – B)



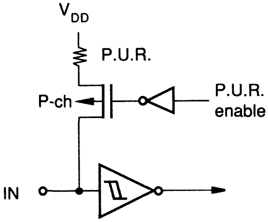
CMOS standard input buffer

(2) Type B



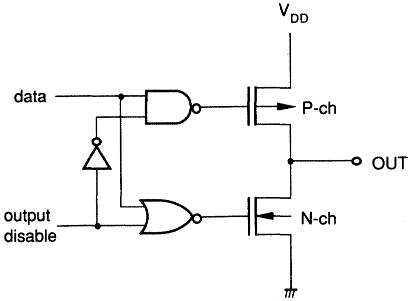
Schmitt trigger input with hysteresis characteristic

(3) Type B – C



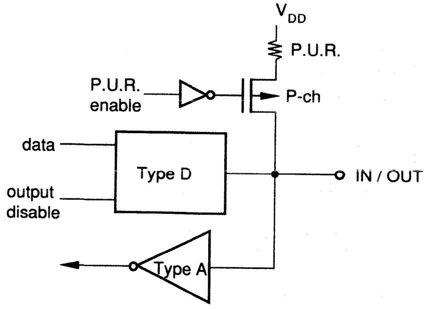
P.U.R.: Pull-Up Resistor

(4) Type D (for Type E – B, F – A)



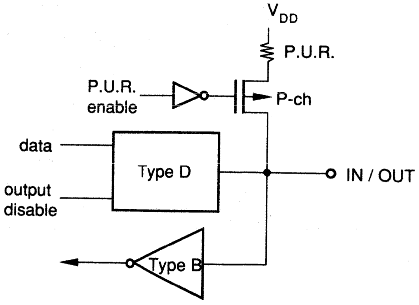
Push-pull output where output can be placed in high impedance (both P and N channels are turned off).

(5) Type E - B



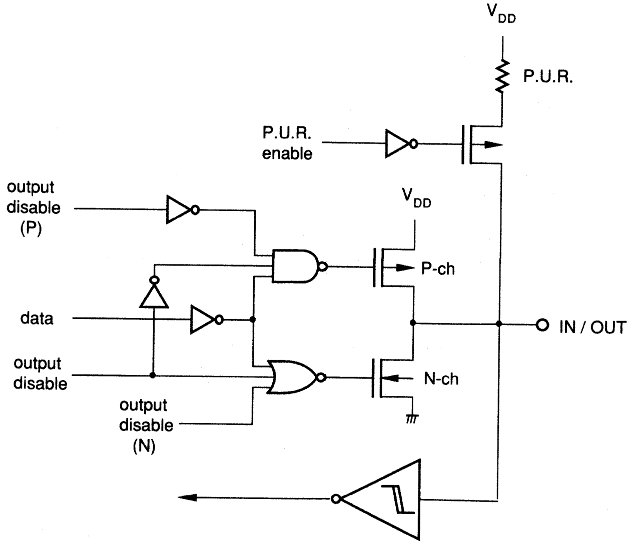
P.U.R.: Pull-Up Resistor

(6) Type F – A



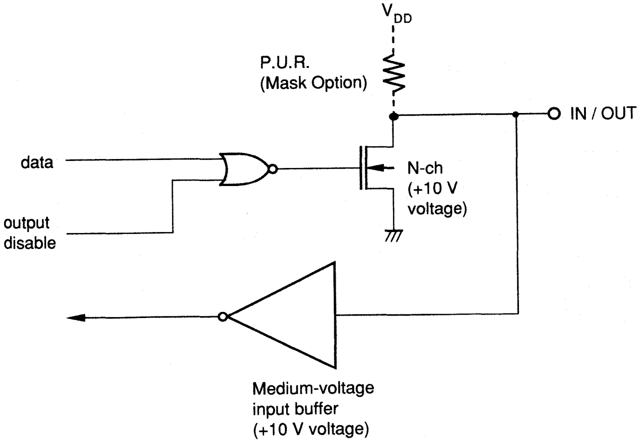
P.U.R.: Pull-Up Resistor

(7) Type F – B



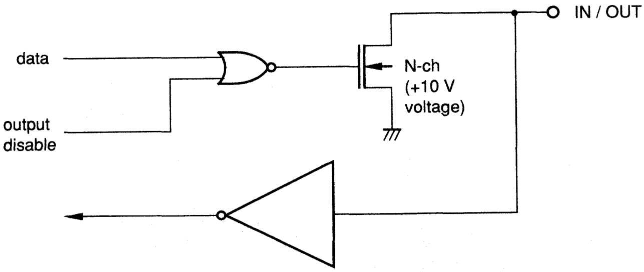
P.U.R.: Pull-Up Resistor

(8) Type M



P.U.R.: Pull-Up Resistor

(9) Type M – A



Middle-high voltage input buffer (+10 V voltage)

## 3. ARCHITECTURE FEATURES AND MEMORY MAP

The μPD75402A architecture is a subset of the μCOM-75X architecture. The features are described here.

### 3.1 Data Memory Bank Configuration and Addressing Mode

3.1.1 Data memory bank configuration
The μPD75402A data memory space consists of banks. The μPD75402A contains general purpose RAM (64 x 4 bits) at address 000H-03FH (bank 0) and peripheral hardware (such as input/output ports and serial interface) at addresses F80H-FFFH (bank 15), as listed in Table 3.1-1. To address the data memory space of 12-bit addresses, the low-order 8-bit address part is specified directly or indirectly by using an instruction. The high-order 4-bit address part is determined by the memory bank (MB) to be accessed.
The μPD75402 contains banks 0 and 15 only and does not contain the memory bank switching function. The memory bank to be accessed is determined by the addressing mode and the specified address. (See tables 3.1-1 and 3.1-2).

**Table 3.1-1 Data Memory Configuration and Address Range in Each Addressing Mode**

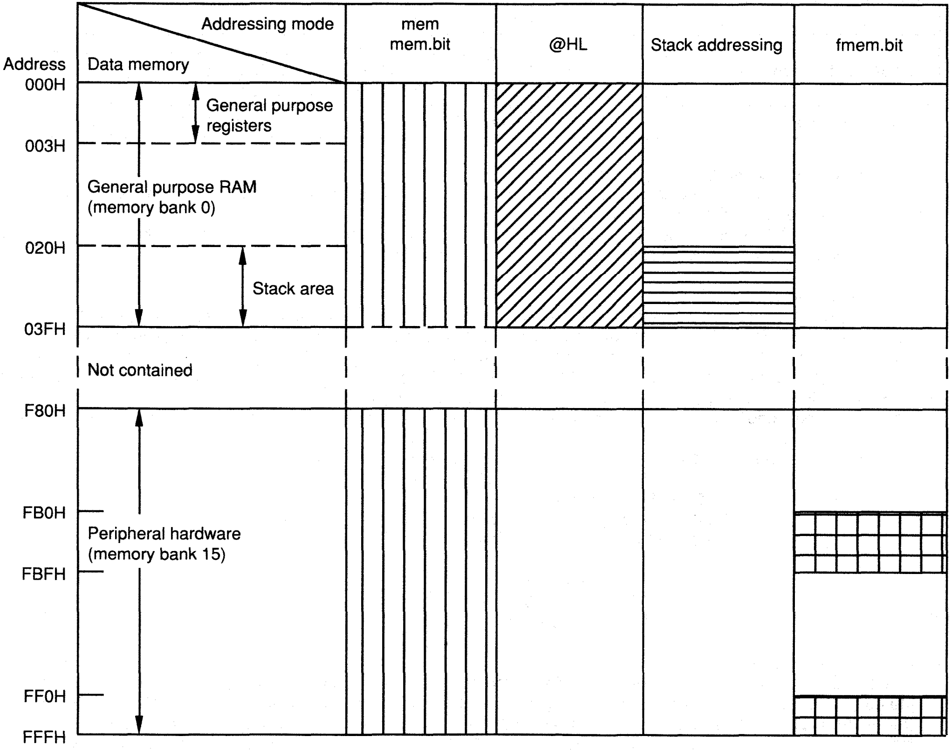


**Table 3.1-2 Addressing Mode List**

| Addressing mode | Representation format | Specified address |
|---|---|---|
| 1-bit direct addressing | mem.bit | Bit specified in bit, of address specified in mem.<br>{ When mem = 00H-3FH, memory bank 0 is accessed.<br>When mem = 80H-FFH, memory bank 15 is accessed. |
| 4-bit direct addressing | mem | Address specified in mem.<br>{ When mem = 00H-3FH, memory bank 0 is accessed.<br>When mem = 80H-FFH, memory bank 15 is accessed. |
| 8-bit direct addressing | | Address specified in mem (mem is an even address).<br>{ When mem = 00H-3FH, memory bank 0 is accessed.<br>When mem = 80H-FFH, memory bank 15 is accessed. |
| 4-bit register indirect addressing | @ HL | Address indicated by the HL contents, of memory bank 0.<br>HL = 00H-3FH |
| Bit manipulation addressing | fmem. bit | Bit specified in bit, of address specified in fmem, of memory bank 15.<br>{ fmem = FB0H-FBFH (interrupt hardware)<br>fmem = FF0H-FFFH (input/output ports) |
| Stack addressing | — | Addressing indicated by SP, memory bank 0. SP = 20H-3FH |

3.1.2 Data memory addressing mode

The µPD75402A provides the six data memory space addressing modes listed in Table 3.1-2 for efficient addressing for each bit length of the data to be processed.

Unlike other µCOM-75X family devices, the µPD75402A enables you to perform programming without taking care of memory bank switching because the memory bank to be accessed is fixed by the addressing mode.

(1) 1-bit direct addressing (mem.bit)

The 1-bit direct addessing mode directly specifies each bit of the data memory space in the instruction operand.

Memory bank 0 (MB = 0) is specified when the address specified in the mem operand is any of 00H-3FH or memory bank 15 (MB = 15) when the address specified in the mem operand is any of 80H-FFH. Thus, the mode enables addressing all bits of both the general purpose RAM area (00H-3FH) and peripheral hardware area (FF0H-FFFH). However, the bits that can be handled bitwise in the peripheral hardware area are limited. (See Table 3.2.2).

The addressing mode is applicable to bit set and clear instructions (SET1 and CLR1) and bit test instructions (SKT and SKF).

Example: Set FLAG1, clear FLAG2, and check whether or not FLAG3 is set to 0.

```
FLAG1 EQU 03FH.1   ; Address 3FH bit 1
FLAG2 EQU 027H.2   ; Address 27H bit 2
FLAG3 EQU 017H.0   , Address 17H bit 0

      SET1  FLAG1  ; FLAG←1
      CLR1  FLAG2  ; FLAG←0
      SKF   FLAG3  ; FLAG = 0 ?
```

(2) 4-bit direct addressing (mem)

The 4-bit direct addressing mode directly specifies all the data memory space in 4-bit units in the instruction operand.

Memory bank 0 (MB = 0) is specified when the address specified in the mem operand is any of 00H-3FH or memory bank 15 (MB = 15) is specified when the address is any of 80H-FFH. Thus, the mode enables addressing both the general purpose RAM area (000H-03FH) and peripheral hardware area (FF0H-FFFH). The addressing mode is applicable to the MOV, XCH, INCS, IN, and OUT instructions.

Example 1: Input port 2 and store in "DATA1".

```
DATA1    EQU  2FH        ; "DATA1" is address 2FH
         IN   A, PORT2   ; A ← PORT2
         MOV  DATA1, A   ; (DATA1) ← A
```

Example 2: Output "BUFF" data into port 5.

```
BUFF        EQU   01AH        ; "BUFF" is address 01AH
            MOV   A, BUFF     ; A ← (BUFF)
            OUT   PORT5, A    ; PORT5 ← A
```

(3) 8-bit direct addressing (mem)

The 8-bit direct addressing mode directly specifies all the data memory space in 8-bit units in the instruction operand.
Memory bank 0 (MB = 0) is specified when the address specified in the mem operand is any of 00H-3FH or memory bank 15 (MB = 15) is specifeid when the address is any of 80H-FFH. Thus, the mode enables addressing both the general purpose RAM area (000H-03FH) and peripheral hardware area (FF0H-FFFH). However, the peripheral hardware area addresses that can be handled in 8-bit units are limited. (See. Table 3.2-2.)
The addressing mode is applicable to the MOV and XCH instructions.

Example 1: Store 8-bit data of the serial interface shift register (SIO) in addresses 20H and 21H.

```
DATA        EQU   020H
            MOV   XA, SIO     ; XA ← SIO
            MOV   DATA, XA    ; (21H) ← X, (20H) ← A
```

Example 2: Read 8-bit data input to SIO into the XA register pair. At the same time set the transfer data stored in the XA register pair and give the transfer start instruction.

```
XCH   XA, SIO      ; XA ↔ SIO
```

(4) 4-bit register indirect addressing (@HL)

The 4-bit register indirect addressing mode indirectly specifies the data memory space in 4-bit units by using the HL register pair.
Memory bank 0 (MB = 0) is always addressed in the addressing mode. Thus, only the general purpose RAM area (000H-03FH) can be addressed. The peripheral hardware area cannot be addressed. Store data in the range of 00H-3FH in the HL register pair.
The addressing mode has wide applications such as data transfer, operations, and comparison.
Data memory space addresses can be updated as desired by using the addressing mode and HL register pair increment and decrement instructions (INCS and DECS) in combination.

Example: Set all the 20H-2FH contents to FH.

```
            MOV   HL, #2FH
            MOV   A, #0FH     ; A ← FH
LOOP:       MOV   @HL,A       ; (HL) ← A
            DECS  L
            BR    LOOP
```

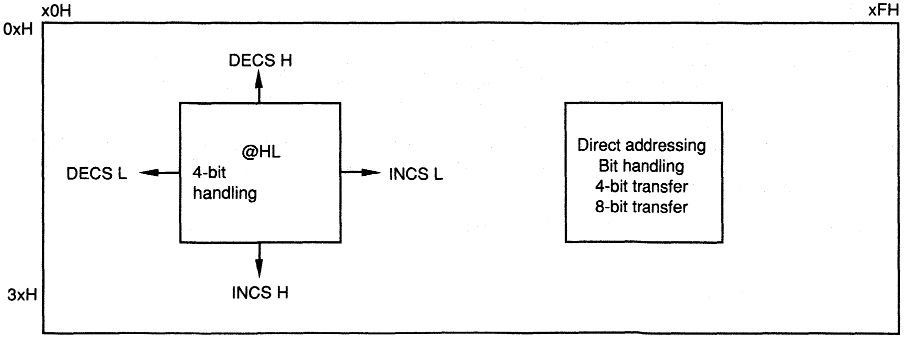


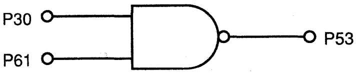
**Fig. 3.1-1 General Purpose RAM Address Update Method**

(5) Specific address bit handling addressing (fmem. bit)
This addressing mode directly specifies each bit of the input/output ports and flags such as interrupt flags of peripheral hardware in the instruction operand. Thus, the addressing mode is applicable to data memory addresses FB0H-FBFH and FF0H-FFFH.
The 1-bit direct addressing mode (mem. bit) is applicable only to bit set, clear, and test instructions; the specific address bit handling addressing mode enables diversified bit handling such as Boolean processing (AND1, OR1, and XOR1 instructions) and test and clear (SKTCLR instruction) as well as bit set, reset, and test instruction handling.

Example 1: Test basic interval timer interrupt request flag (IRQBT). If the flag is set, clear IRQBT and clear the P63 pin level.

```
SKTCLR    IRQBT       ; IRQBT = 1 ?
BR        NO          ; NO
CLR1      PORT6.3     ; YES
```

Example 2: If both P30 and P61 are set to 1, clear P53.



```
(i)                SET1    CY             ; CY ← 1
                   AND1    CY, PORT3.0    ; CY ← P3.0
                   AND1    CY, PORT6.1    ; CY ∧ P.6.1
                   SKT     CY             ; CY = 1 ?
                   BR      SETP
                   CLR1    PORT5.3        ; P53 ← 0
                     •
                     •
          SETP:    SET1    PORT5.3        ; P53 ← 1

(ii)               SKT     PORT3.0        ; P30 = 1 ?
                   BR      SETP
                   SKT     PORT6.1        ; P61 = 1 ?
                   BR      SETP
                   CLR1    PORT5.3        ; P53 ← 0
                     •
                     •
          SETP:    SET1    PORT5.3        ; P53 ← 1
```

(6) Stack addressing
The stack addressing mode is used for save and restore operation when interrupt service or subroutine is performed. The data memory is indirectly addressed by using an 8-bit stack pointer (SP).
Memory bank 0 (MB = 0) is always addressed in the stack addressing mode. Since the high-order three bits of the stack pointer are fixed to 001, the 020H-03FH area can only be addressed. The stack addressing mode is also used to save and restore registers when PUSH and POP instructions are executed.

Caution: Unlike the µPD75402A, the evaluation chip installed on the evaluation board enables the stack addressing mode to address all the area of memory bank 0. To eliminate the difference during the evaluation, set the stack pointer to a value so as not to access exceeding the 20H-3FH range.

Example 1: Save and restore registers in subroutine processing.

```
SUB:      PUSH      XA
          PUSH      HL
                :
                :
          POP       HL
          POP       XA
          RET
```

Example 2: Transfer the HL register pair contents to the XA register pair.

```
PUSH    HL
POP     XA;   XA ← HL
```

Example 3: Branch to the address indicated by [XAHL] register.

```
PUSH    HL
PUSH    XA
RET           ; Branch to address XAHL
```

### 3.2 Memory Mapped I/O

The μPD75402A adopts memory mapped I/O which maps peripheral hardware such as input/output ports and serial interface in data memory space addresses F80H-FFFH as listed in Table 3.1-1. Thus, special instructions to control the peripheral hardware are not provided. The peripheral hardware is all controlled by using memory handling instructions. (Some hardware control mnemonics are provided to easily understand programs.)
The addressing modes listed in Table 3.2-1 can be used when the peripheral hardware is handled.

**Table 3.2-1 Applicable Addressing Modes when Peripheral Hardware is Handled**

| | Applicable addressing mode | Applicable hardware |
|---|---|---|
| bit manipulation | Direct addressing. Specify the bit to be handled in mem.bit | All hardware devices where bit handling can be performed. |
| | Direct addressing. Specify the bit to be handled in fmem. bit | IEXXX, IRQXXX, PORTn.X |
| 4-bit handling | Direct addressing. Specify the address to be handled in mem. | All hardware devices where 4-bit handling can be performed |
| 8-bit handling | Direct addressing. Specify the address to be handled in mem (mem is an even address). | All hardware devices where 8-bit handling can be performed |

Table 3.2-2 lists the μPD75402A I/O map. The items in the table mean:

- Abbreviation: Name indicating the internal hardware address. It can be entered in the instruction operand field. However, IME cannot be entered.
- Number of bits that can be handed. Indicates the number of applicable bits to be processed when the hardware device is handled. Symbols such as R/W indicate whether or not the hardware device can be read/written.
  R/W : Read/written can be made. R: Read only can be made. W: Write only can be made.
- Bit handling addressing: Indicates the applicable bit handling addressing when the hardware device is handled bitwise.

**Table 3.2-2 μPD75402A I/O Map**

| | Hardware name (abbreviation) | Number of bits that can be manipulated | | | Bit manipulation addressing | |
|---|---|---|---|---|---|---|
| Address | b3 \| b2 \| b1 \| b0 | 1-bit | 4-bit | 8-bit | | Remarks |
| F80H | Stack pointer (SP) | — | — | W | | Bit 0 is fixed to 0. |
| | | — | — | | | |
| F85H | Basic interval timer mode register (BTM) | — | W | — | | Be sure to write 11 into bits 1 and 0. |
| F86H | Basic interval timer (BT) | — | — | R | | |
| | | — | — | | | |

**Table 3.2-2 μPD75402A I/O Map (cont'd)**

| Address | Hardware name (abbreviation) | | | | Number of bits that can be manipulated | | | Bit manipulation addressing | Remarks |
|---|---|---|---|---|---|---|---|---|---|
| | b3 | b2 | b1 | b0 | 1-bit | 4-bit | 8-bit | | |
| FB2H | (IME) | | | | — | — | — | | Handle it by using EI and DI instructions |
| FB3H | Processor clock control register (PCC) | | | | — | W | | | |
| FB4H | INT0 mode register (IM0) | | | | — | W | — | fmem. bit | Bit 2 is fixed to 0. |
| FB8H | 0 | 0 | IEBT | IRQBT | R/W | R/W | — | | |
| FBDH | 0 | 0 | IECSI | IRQCSI | R/W | R/W | — | | |
| FBEH | 0 | 0 | IE0 | IRQ0 | R/W | R/W | — | | |
| FBFH | 0 | 0 | IE2 | IRQ2 | R/W | R/W | | | |
| FD0H | Clock output mode register (CLOM) | | | | — | W | — | | |
| FDCH | Pull-up resistor specification register A (POGA) | | | | — | — | W | | |
| | | | | | — | — | | | |
| FE0H | Serial operation mode register (CSIM) | | | | — | — | W | | |
| FE1H | CSIE | C0I | WUP | 0 | Note 1 | — | | mem. bit | Be sure to write 0 into bit 0. |
| FE2H | CMDD | RELD | CMDT | RELT | Note 2 | — | — | mem. bit | All bits can be handled only bitwise |
| | SBI control register (SBIC) | | | | | | | | |
| FE3H | BSYE | ACKD | ACKE | ACKT | Note 3 | — | | | |
| FE4H | Serial I/O shift register (SI0) | | | | — | — | R/W | | |
| | | | | | — | — | | | |
| FE6H | Slave address register (SVA) | | | | — | — | W | | Be sure to write 11000 into the high-order five bits. |
| | | | | | — | — | | | |
| FE8H | Port mode register group A (PMGA) | | | | — | — | W | | |
| | | | | | — | — | | | |
| FECH | Port mode register group B (PMGB) | | | | — | — | W | | |
| | | | | | — | — | | | |
| FF0H | Port 0 (PORT 0) | | | | R | R | — | fmem. bit | |
| FF1H | Port 1 (PORT 1) | | | | R | R | | | Bits 3 and 1 are fixed to 0. |
| FF2H | Port 2 (PORT 2) | | | | R/W | R/W | — | | |
| FF3H | Port 3 (PORT 3) | | | | R/W | R/W | | | |
| FF5H | Port 5 (PORT 5) | | | | R/W | R/W | — | | |
| FF6H | Port 6 (PORT 6) | | | | R/W | R/W | — | | |

Note 1: Bits 3 and 1 = W, bit 2 = R

Note 2: Bits 3 and 2 = R, bits 1 and 0 = W

Note 3: Bits 3 and 1 = R/W, bit 2 = R, bit 0 = W

Remarks: IEXXX is an interrupt enable flag., IRQXXX is an interrupt request flag.

# 4. INTERNAL CPU FUNCTION

## 4.1 Program Counter

The program Counter (PC) is an 11-bit binary counter which retains program memory address information.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| PC10 | PC9 | PC8 | PC7 | PC6 | PC5 | PC4 | PC3 | PC2 | PC1 | PC0 |

**Fig. 4.1-1 Program Counter Format**

The program counter operates as follows:

- When normal operation is performed
  Each time one instruction is executed, the program counter contents are automatically incremented according to the number of bytes of the instruction.
- When branch instruction (BR or BRCB) is executed
  The immediate data indicating the branch destination address is stored in the PC.
- When subroutine call instruction (CALLF) is executed or vectored interrupt occurs
  The PC contents are saved in stack memory, then the branch destination address is stored in the PC.
- When return instruction (RET, RETS, or RETI) is executed
  The stack memory contents are restored in the PC.
- When $\overline{\text{RESET}}$ is input
  The low-order three bits of program memory address 000H are stored in PC10-PC8 and the address 001H contents are stored in PC7-PC0, then initialization is made. Program can be started at any desired address.

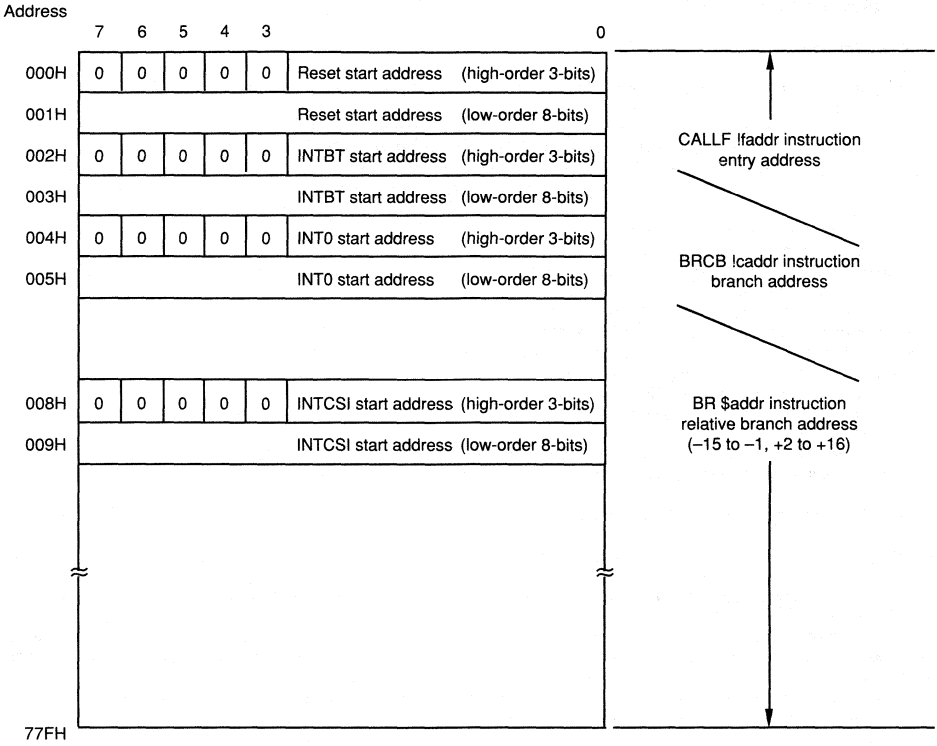
## 4.2 Program Memory (ROM) – 1920 words x 8 bits

The program memory (ROM) is mask programmable ROM consisting of 1920 word x 8 bits. It stores programs and data such as table data.

The program memory is addressed by the program counter. The table data in ROM can also be read by using a table reference instruction (MOVT).

A branch can be made to the entire program area by using a branch or subroutine call instruction. (See Fig.4.2-1) A relative branch instruction (BR $addr) enables a branch in the range of (–15 to +16) from the address indicated by the PC after instruction execution.

Program memory addresses are 000H-77FH. The addresses shown in Fig. 4.2-1 are specially assigned. All area except 00H or 001H can be used as a normal memory area.



| Address | 7 | 6 | 5 | 4 | 3 | 0 |
|---|---|---|---|---|---|---|
| 000H | 0 | 0 | 0 | 0 | 0 | Reset start address (high-order 3-bits) |
| 001H | Reset start address (low-order 8-bits) | | | | | |
| 002H | 0 | 0 | 0 | 0 | 0 | INTBT start address (high-order 3-bits) |
| 003H | INTBT start address (low-order 8-bits) | | | | | |
| 004H | 0 | 0 | 0 | 0 | 0 | INT0 start address (high-order 3-bits) |
| 005H | INT0 start address (low-order 8-bits) | | | | | |
| | | | | | | |
| 008H | 0 | 0 | 0 | 0 | 0 | INTCSI start address (high-order 3-bits) |
| 009H | INTCSI start address (low-order 8-bits) | | | | | |
| | | | | | | |
| 77FH | | | | | | |

CALLF !faddr instruction entry address
BRCB !caddr instruction branch address
BR $addr instruction relative branch address (–15 to –1, +2 to +16)

**Fig. 4.2-1 Program Memory Map**

### 4.3 Data Memory

The data memory consists of general purpose static RAM (memory bank 0) and peripheral hardware area (memory bank 15) as shown in Fig. 4.3-1.

The general purpose static RAM contains 64 x 4 bits of memory bank 0. It stores process data and is also used as stack memory when a subroutine instruction or interrupt is executed.

Specific addresses of the general purpose RAM are assigned as general purpose registers which are handled by using general purpose register access instructions.

Addresses 020H-03FH of memory bank 0 can be used as a stack area.

Although one address of general purpose RAM consists of four bits, data can be handled in 8-bit units by using 8-bit memory handling instructions or bitwise by using bit handling instructions. Specify even addresses in the 8-bit handling instructions.

Peripheral hardware is mapped in F80H-FFFH of memory bank 15 (peripheral hardware area). Like general purpose RAM, it can be accessed by using memory handling instructions. However, the number of bits that can be handled for each address of the peripheral hardware is limited. (See table 3.2-2) Data memory is not contained in addresses to which peripheral hardware is not assigned in Table 3.2-2. Do not access the addresses.

Since general purpose RAM becomes undefined at reset, normally it should be initialized to 0 (RAM clear) at the beginning of a program, or an unexpected bug may be caused. Be sure to initialize the general purpose RAM to 0.

Example: Clear the entire general purpose RAM area (00H-3FH). (However, FFH remains in the HL register pair).

```
        MOV     HL, #3FH
        MOV     A, #0H
LOOP:   MOV     @HL, A      ; Clear 00H-3FH
        DECS    L
        BR      LOOP
        DECS    H
        BR      LOOP
```

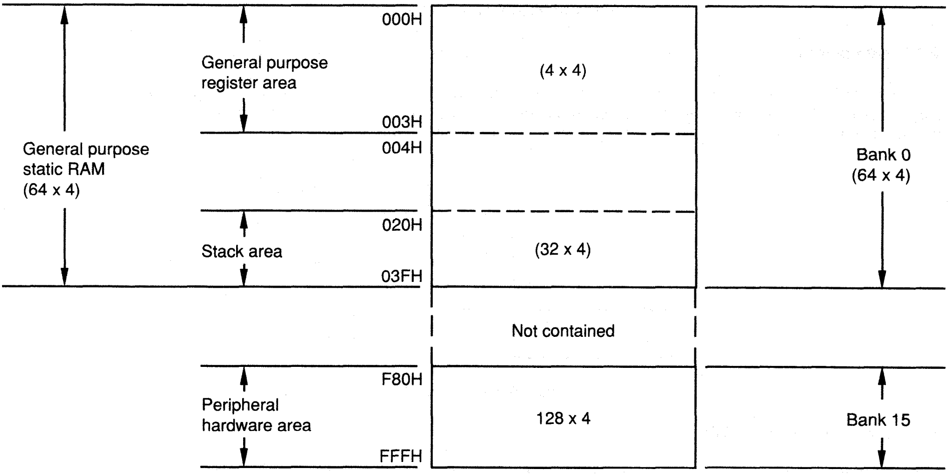


**Fig. 4.3-1 Data Memory Map**

### 4.4 General Purpose Registers – 4 x 4 bits

The general purpose registers are four 4-bit registers H, L, X, and A which are assigned to specific addresses of the data memory. Each general purpose register is handled in 4-bit units. In addition, HL and XA form register pairs which are handled in 8-bit units. The HL register pair can also be used as a data pointer for indirectly addressing memory.

The general purpose register area can be addressed and accessed as normal RAM regardless of whether or not it is used as registers.

| | |
|---|---|
| X<br>01H | A<br>00H |
| H<br>03H | L<br>02H |

Remarks: The number shown in the lower right corner is the data memory address to which the register is assigned.

**Fig. 4.4-1 General Purpose Register Configuration**

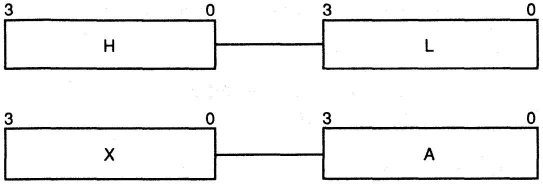


**Fig. 4.4-2 Register Pair Configuration**

### 4.5 Accumulators

On the µPD75402A, the A register and XA register pair function as accumulators. 4-bit data processing instructions are executed centering around the A register; 8-bit data processing instruction are executed centering around the XA register pair.
When bit handling are executed, the carry flag (CY) functions as a bit accumulator.

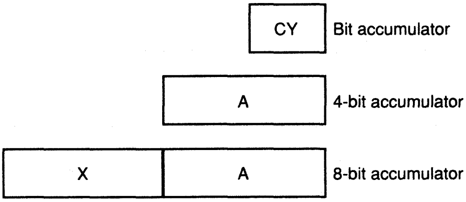


**Fig. 4.5-1 Accumulators**

### 4.6 Stack Pointer (SP) – 8 bits

The µPD75402A uses general purpose RAM as stack memory (LIFO). The stack pointer (SP) is an 8-bit register which retains the stack memory area top address information . Fig. 4.6-1 shows the stack pointer (SP) format.
Since the high-order three bits of the SP are fixed to 001, general purpose RAM addresses 020H-03FH are used for the stack area.
The SP is decremented before write (save) operation into stack memory and incremented after read (restore) operation from the stack memory. Figs. 4.6-2 and 4.6-3 show data saved in and restored from the stack memory.
The initial value is set in the SP by using an 8-bit data transfer instruction to determine the stack area. The SP contents cannot be read.
0 is always written into SP bit 0.
It is recommended to write 40H into the SP for initialization and use the area from the highest address of internal RAM (03FH) to lower addresses as the stack area.
When RESET is input, the SP contents become undefined. Be sure to initialize the SP to any desired value by using the program initialization routine.

Caution: Unlike the µPD75402A, the evaluation chip installed on the evaluation board enables addressing all the area of 000H-0FFH. To eliminate the difference during evaluation, set the SP to a value so as not to access exceeding the range of 020H-03FH.

Example: Initialize SP.

```
MOV     XA, #40H
MOV     SP, XA      ; SP ← 40H
```

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 1 | SP4 | SP3 | SP2 | SP1 | SP0 | SP |

↑ Fixed to 0

**Fig. 4.6-1 Stack Pointer Format**

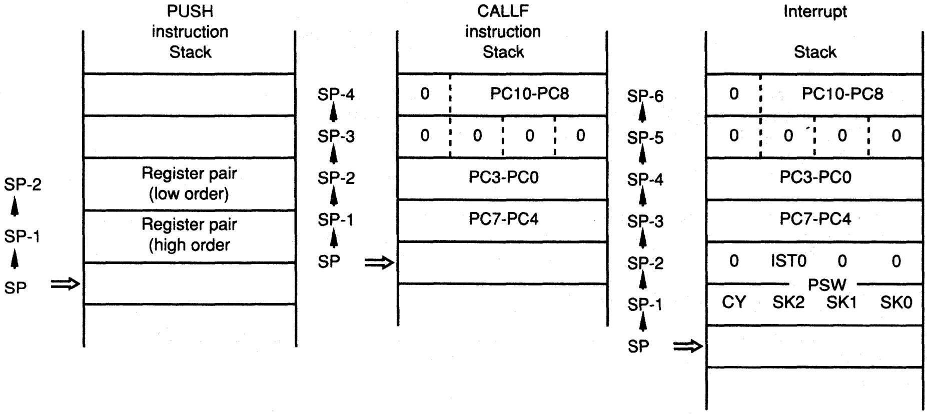


**Fig. 4.6-2 Data Saved in Stack Memory**

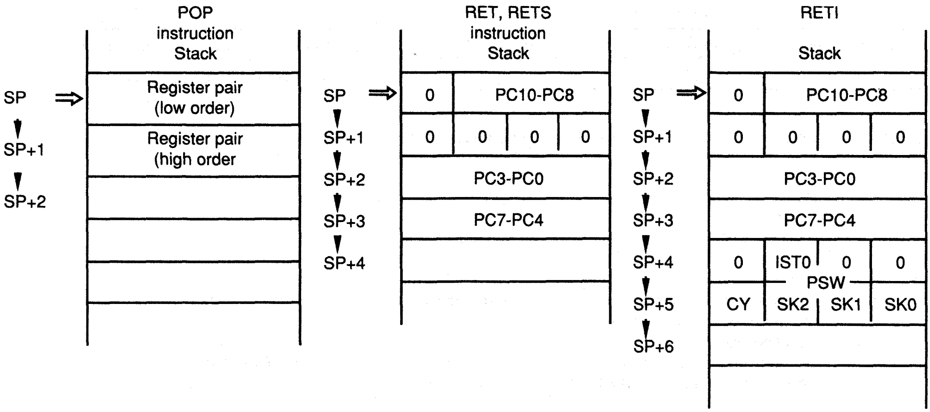


**Fig. 4.6-3 Data Restored from Stack Memroy**

### 4.7 Program Status Word (PSW) – 8 bits

The program status word (PSW) consists of flags related closely to processor operation. Fig. 4.7-1 shows the program status word (PSW) format.

When an interrupt is acknowledged, the PSW contents are saved in the stack memory in 8-bit units. When a RETI instruction is executed, the PSW contents are restored from the stack memory in 8-bit units. (See Figs. 4.6-2 and 4.6-3).

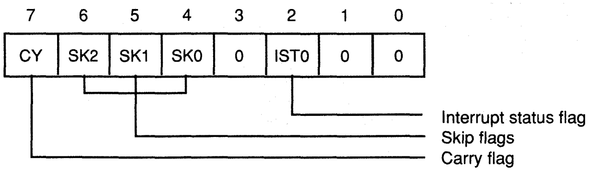


**Fig. 4.7-1 Program Status Word Format**

Since the PSW is not assigned to the data memory space, the flags cannot be handled by using memory handling instructions. However, only the carry flag (CY) can be handled by using its dedicated instructions.

When $\overline{\text{RESET}}$ is input, IST0 is set to 0 and SK0-SK2 and CY become undefined.

(1) Carry flag (CY)

The carry flag is a 1-bit flag to store overflow occurrent information when an operation instruction with carry (ADDC) is executed.

The carry flag also has the bit accumulator function. Boolean algebra operation is performed between the bit accumulator and data memory specified in the bit address, and the result can be stored in the bit accumulator.

When the $\overline{\text{RESET}}$ signal is generated, the carry flag becomes undefined.

**Table 4.7-1 Carry Flag Handling Instructions**

| | Instruction (mnemonic) | Carry flag operation or processing |
|---|---|---|
| Carry flag handling dedicated instruction | SET1 CY<br>CLR1 CY<br>NOT1 CY<br>SKT CY | Set CY to 1<br>Clear CY<br>Invert the CY contents<br>Skip if CY contains 1 |
| Bit Boolean instructions | AND1 CY, fmem. bit<br>OR1 CY, fmem. bit<br>XOR1 CY, fmem. bit | The specified bit contents and CY contents are ANDed (ORed, or exclusively ORed) together, and the result is stored in CY. |
| Interrupt service | When interrupt is executed | CY is saved in stack memory in parallel with other seven PSW bits. |
| | RETI | CY is restored from stack memory in parallel with other PSW bits. |

Example: AND address 3FH bit 3 and P33 together and store the result in CY.

```
SET1  CY            ; CY ← 1
SKT   3FH.3         ; Skip if address 3FH bit 3 is set to 1
CLR1  CY            ; CY ← 1
AND1  CY, PORT3.3   ; CY ← CY ∧ P33
```

(2) Skip flag (SK2, SK1, SK0)

The skip flag stores the skip state. It is automatically set or cleared by the CPU which executeds a given instruction. The skip flag cannot directly be handled by user.

(3) Interrupt status flag (IST0)

The interrupt status flag stores the status of the current processing being performed. (See Table 6.3-2 for details.) It cannot directly be handled by the user in a program.

**Table 4.7-2 Interrupt Status Flag Indication Contents**

| IST0 | Status of current processing being performed | Processing contents and interrupt control |
|---|---|---|
| 0 | Status 0 | Normal program processing is being performed. Every interrupt can be acknowledged. |
| 1 | Status 1 | Interrupt service is being executed. No interrupts can be acknowledged. |

When an interrupt is acknowledged, the IST0 contents are saved in stack memory as a part of PSW, then automatically set to 1. When a RETI instruction is executed, IST0 is set to 0.
Since IST0 cannot be handled by using an instruction, it is always set to 1 during interrupt service.
Thus, interrupts are not multiplexed and all interrupt requests occuring during the interrupt service are held until the interrupt service being executed terminates. (See 6. for details).

## 5. PERIPHERAL HARDWARE FUNCTION

### 5.1 Digital Input/Output Ports

The µPD75402A contains internal digital input/output ports (ports 0-3, 5 and 6).
The µPD75402A adopts memory mapped I/O which maps all input/output ports in the data memory space.
All data memory handling instructions are applicable to the ports; diversified bit handling as well as 4-bit input/output can be performed.
Caution: The µPD75402A does not enable 8-bit input/output using ports in pairs.

| Address | 3 | 2 | 1 | 0 | Abbreviation |
|---|---|---|---|---|---|
| FF0H | P03 | P02 | P01 | P00 | PORT0 |
| FF1H | 0 | P12 | 0 | P10 | PORT1 |
| FF2H | P23 | P22 | P21 | P20 | PORT2 |
| FF3H | P33 | P32 | P31 | P30 | PORT3 |
| | — | | | | |
| FF5H | P53 | P52 | P51 | P50 | PORT5 |
| FF6H | P63 | P62 | P61 | P60 | PORT6 |

**Fig. 5.1-1 Digital Input/Output Port Data Memory Addresses**

5.1.1 Types, features, and configuration of digital input/output ports
Table 5.1-1 lists the digital input/output port types. Figs. 5.1-2 to 5.1-5 show the port configuration.

**Table 5.1-1 Digital Input/Output Port Types and Features**

| Port (abbreviation) | Function | Operation and features | Remarks |
|---|---|---|---|
| PORT0<br>PORT1 | 4-bit input | Can always be read or tested regardless of the operation mode of the SO/SB0, SI, $\overline{\text{SCK}}$, INT0 or INT2 pin. | The pins are also used for SO/SB0, SI, $\overline{\text{SCK}}$, INT0 and INT2. (See 1.) |
| PORT3 (Note) | 4-bit input/output | Can be placed in the input or output mode bitwise. | |
| PORT2<br>PORT6 | | Can be placed in the input or output mode in 4-bit units. | The P22 (port 2) pin is also used for PCL. |

**Table 5.1-1 Digital Input/Output Port Types and Features (Cont'd)**

| Port (abbreviation) | Function | Operation and features | Remarks |
|---|---|---|---|
| PORT5 (Note) | 4-bit input/output (N-ch open drain 10 V voltage) | Can be placed in the input or output mode in 4-bit units. | Internal pull-up resistor can be specified bitwise by using mask option. (μPD75402A only) |

Note: LED can be directly driven.

The μPD75402A enables an internal pull-up resistor to be contained in all port pins except P00 or P10. The μPD75P402 enables internal an pull-up resistor to be contained in port pins except P00, P10 or P50-P53. (See 5.5)
P10 is also used for the external vectored interrupt input pin and contains an internal noise eliminator using sampling clocks. (See 6.2 for details).
When $\overline{\text{RESET}}$ is input, the output latches of ports 2, 3, 5 and 6 are cleared, the output buffers are turned off, and the input mode is entered.

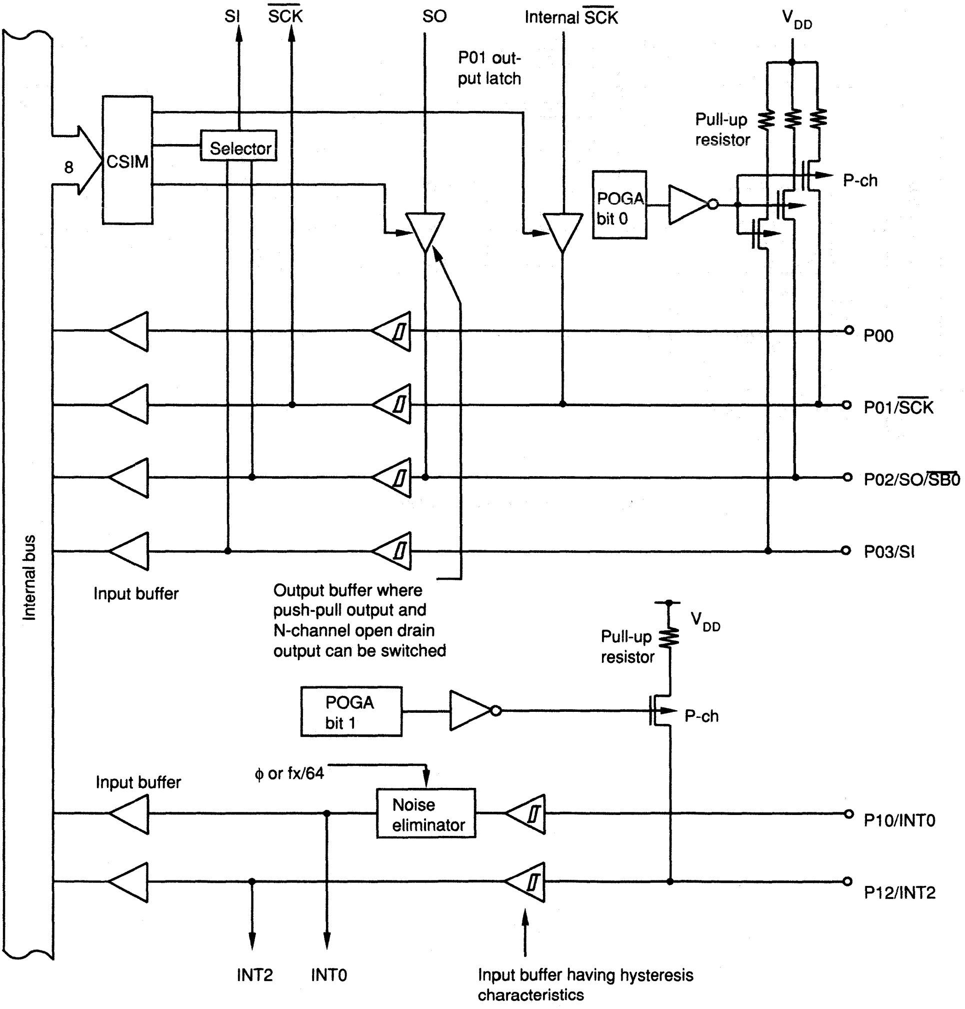


**Figure 5.1–2 Configuration of Ports 0 or 1**

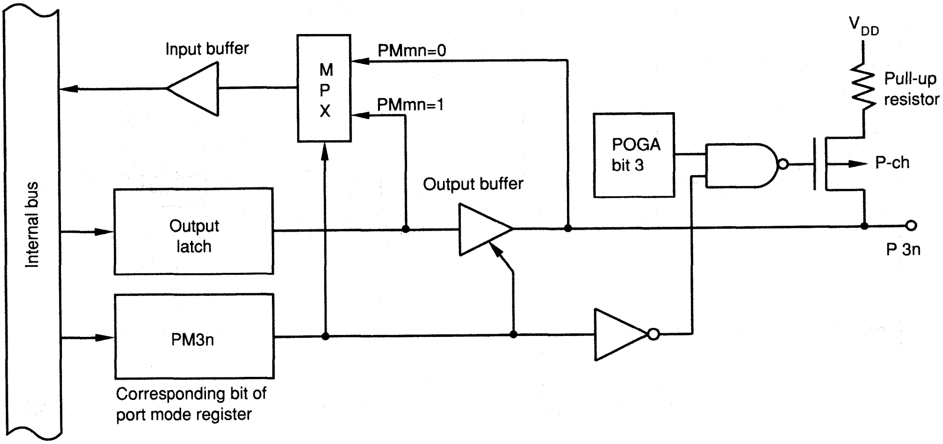


**Figure 5.1–3 Configuration of port3 (n = 0 –3)**

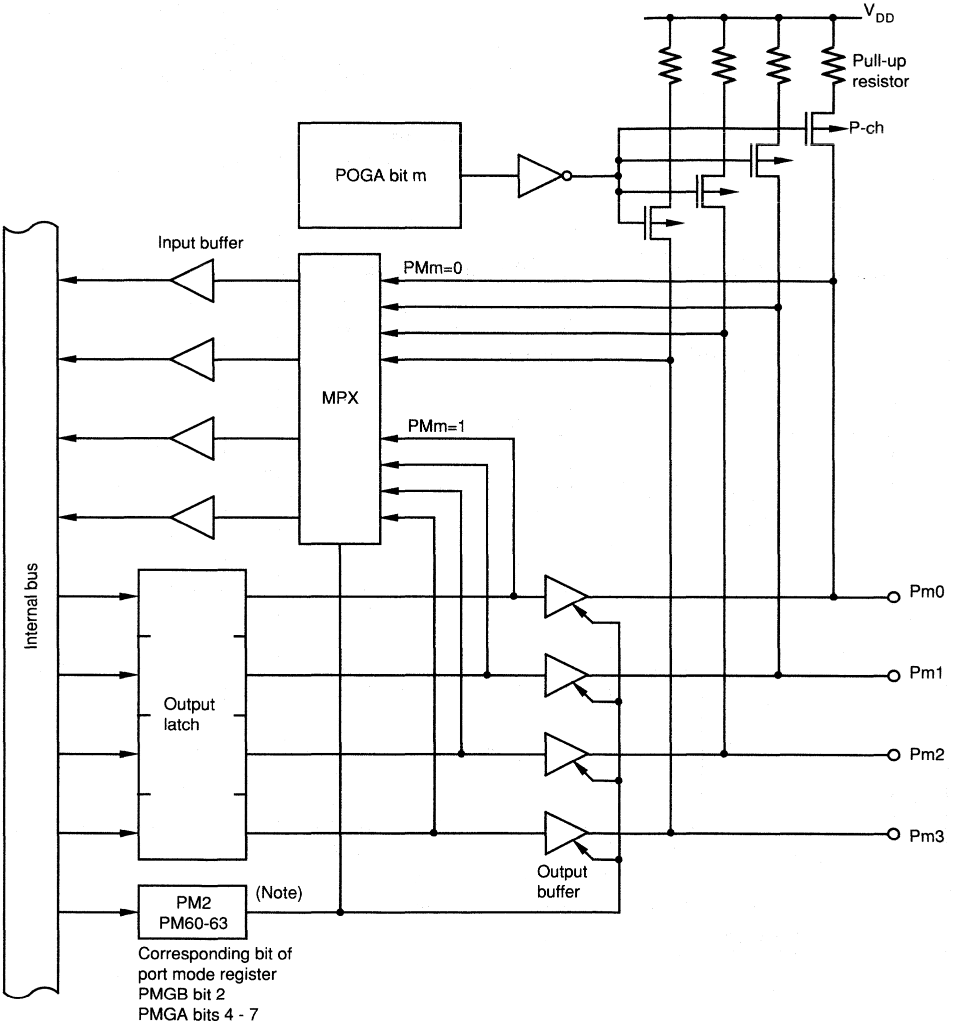


**Figure 5.1–4 Configuration of Port 2 or 6**

Note: The input or output mode is selected for port 2 by using PMGB bit 2 (PM2) and for port 6 by using PMGA bits 4 - 7 (PM60-PM63).

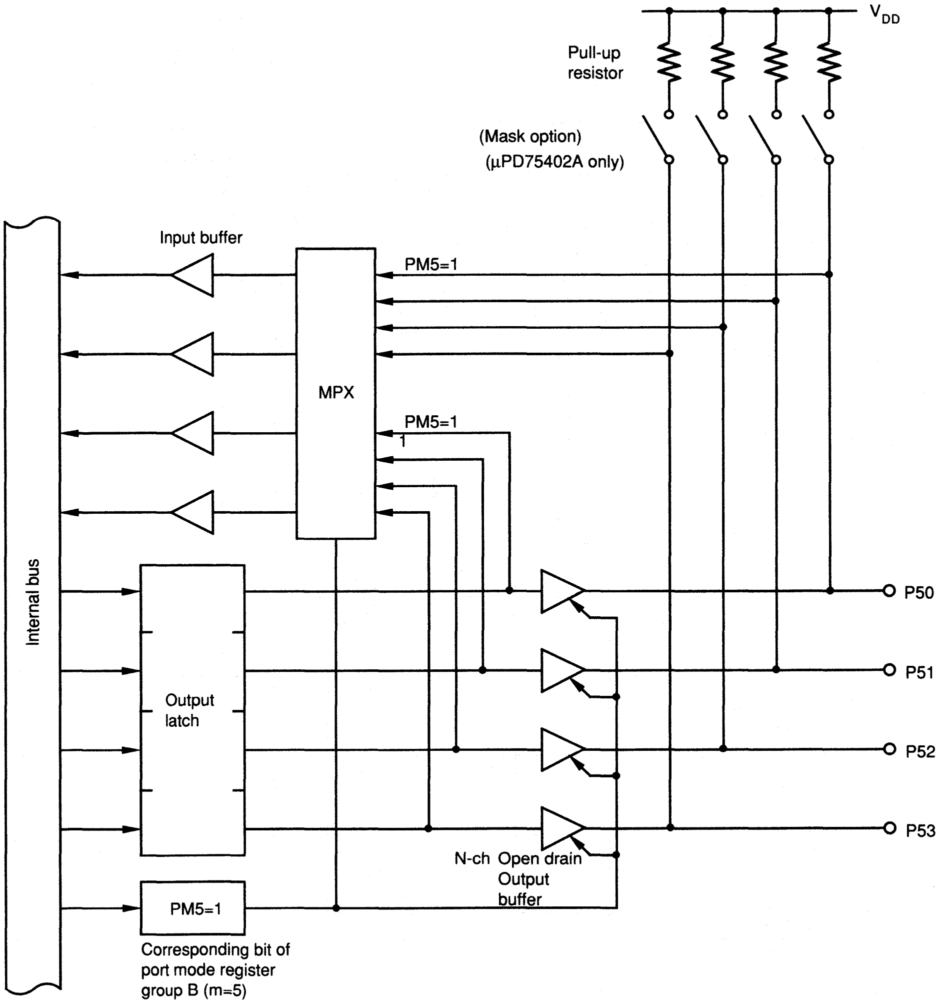


**Figure 5.1–5 Configuration of Port 5**

5.1.2 Input/output mode setting

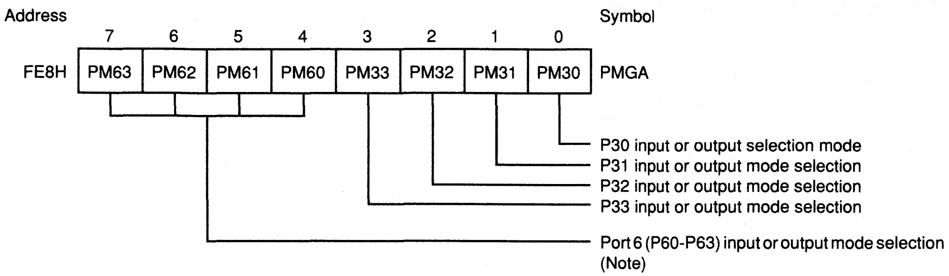
The input or output mode is selected for each input/output port as shown in Fig. 5.1-6. The input or output mode can be selected for port 3 bitwise by using the port mode register A (PMGA). The input or output mode is selcted for ports 2 and 5 by using PMGB and for port 6 by using PMGA in 4-bit units.

Each port serves as an input port when its corresponding port mode register bit is set to 0; it serves as an output port when 1. The output latch contents are output to the output pin at the same time the output mode is selected by setting the port mode register. Thus, the output latch contents must be preset to the required value before the output mode is selected.

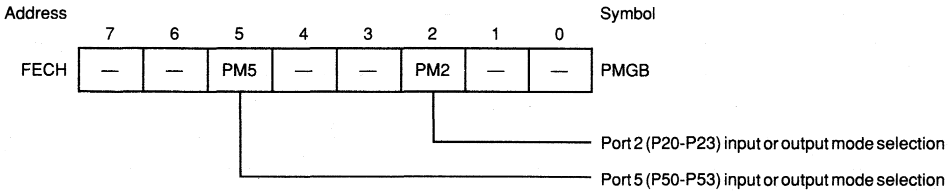
Port mode registers A and B are set by using 8-bit memory handling instructions.

When the $\overline{\text{RESET}}$ signal is generated, all bits of the port mode registers are cleared; the output buffers are turned off and all ports are placed in the input mode.

Port Mode register A


Address
Symbol
7
6
5
4
3
2
1
0
FE8H
PM63
PM62
PM61
PM60
PM33
PM32
PM31
PM30
PMGA
P30 input or output selection mode
P31 input or output mode selection
P32 input or output mode selection
P33 input or output mode selection
Port 6 (P60-P63) input or output mode selection
(Note)

Port mode register B


Address
Symbol
7
6
5
4
3
2
1
0
FECH
—
—
PM5
—
—
PM2
—
—
PMGB
Port 2 (P20-P23) input or output mode selection
Port 5 (P50-P53) input or output mode selection

| | Selection |
|---|---|
| 0 | Input mode (output buffer off) |
| 1 | Output mode (output buffer on) |

Note: Select the input or output mode for port 6 in 4-bit units. Be sure to write 0000 or 1111 into PMGA bits 7-4.

**Fig. 5.1-6 Port Mode Register Formats**

5.1.3 Digital input/output port handling instructions

Since all the input/output ports contained in the µPD75402A are mapped in the data memory space, all data memory handling instructions can be used. Table 5.1-2 lists the data memory handling instruction particularly useful for input/output pin handling and the instruction application.

(1) Bit handling instructions

Specific address bit direct addressing (fmem.bit) can be used for all digital input/output ports.

Example: OR P50 and P31 together and output the result to P61.

```
                SET1    CY              ; CY ← 1
                AND1    CY, PORT5.0     ; CY ← CY ∧ P50
                OR1     CY, PORT3.1     ; CY ← CY ∨ P31
                SKT     CY
                BR      CLRP
                SET1    PORT6.1         ; P61 ← 1
                  :
                  :
        CLRP:   CLR1    PORT6.1         ; P61 ← 0
```

(2) 4-bit handling instructions

All 4-bit memory handling instructions such as MOV, XCH, ADDS, and INCS, as well as the IN and OUT instructions, can be used.

Example 1: Output the accumulator contents to port 3.

```
OUT     PORT3, A
```

Example 2: Add the accumulator value to the data output to port 5 and output the result to port 5.

```
MOV     HL, #PORT5      ; A ← A+PORT5
ADDS    A, @HL
NOP
MOV     @HL, A          ; PORT5 ← A
```

**Table 5.1-2 Input/Output Pin Handling Instruction Lins**

| | | Port 0 | Port 1 | Port 2 | Port 3 | Port 5 | Port 6 |
|---|---|---|---|---|---|---|---|
| IN | A, PORTn | O | | | | | |
| OUT | PORTn, A | — | — | O | | | |
| SET1 | PORTn.bit | — | — | O | | | |
| CLR1 | PORTn.bit | — | — | O | | | |
| SKT | PORTn.bit | O | | | | | |
| SKF | PORTn.bit | O | | | | | |
| AND1 | CY, PORTn.bit | O | | | | | |
| OR1 | CY, PORTn.bit | O | | | | | |
| XOR | CY, PORTn.bit | O | | | | | |

### 5.1.4 Digital input/output port operation

When a data memory handling instruction is executed for each digital input/output port, the port (pin) operation varies depending on which mode, input or output mode, is selected. (See Table 5.1-3.) This is because pin data is read onto the internal data bus in the input mode and output latch data in the output mode, as understood from the input/output port configuration.

(1) Operation when port (pin) is placed in input mode

When a test instruction such as SKT or an instruction to read port data onto the internal bus in four bits (IN, MOV, or bit operation instruction) is executed, pin data is handled.

When an instruction to transfer the accumulator contents to a port in four bits (OUT or MOV instruction) is executed, the accumulator data is latched in the output latch. The output buffer remains off. When an XCH instruction is executed, pin data is input to a given accumulator and the accumulator data is latched in the output latch. The output buffer remains off.

When INCS instruction is executed, data resulting from adding 1 to pin data (four bits) is latched in the output latch. The output buffer remains off.

When an instruction to rewrite the data memory bitwise, such as the SET1, CLR1, or SKTCLR instruction, is executed, the output latch of the specified bit can be rewritten as designated by the instruction, but the output latch contents of other bits become undefined.

(2) Operation when port (pin) is placed in output mode

When a test instruction, bit input instruction , or instruction to read port data onto the internal bus in four bits is executed, the ouput latch contents are handled.

When an instruction to transfer the accumulator contents in four bits executed, output latch data is rewritten and output from the pin at the same time.

When an XCH instruction is executed, the output latch contents are transferred to a given accumulator and the accumulator contents are latched in the output latch and output from the pin.

When INCS instruction is executed, data resulting from adding 1 to the output latch contents is latched in the output latch and and output from the pin.

When a bit output instruction is executed, the specified output latch bit is rewritten and output from the pin.

**Table 5.1-3 Operation when Input/Output Port is Handled**

| Executed instruction | Port (pin) operation | |
|---|---|---|
| | Input mode | Output mode |
| SKT PORTn.bit<br>SKF PORTn.bit | Test pin data | Test output latch |
| AND1 CY, PORTn.bit<br>OR1 CY, PORTn.bit<br>XOR1 CY, PORTn.bit | Perform operation between pin data and CY | Perform operation between output latch data and CY |
| IN A, PORTn<br>MOV A, PORTn | Transfer pin data to accumulator | Transfer output latch data to accumulator |
| OUT PORTn, A<br>MOV PORTn, A | Transfer accumulator data to output latch (Output buffer remains off.) | Transfer accumulator data to output latch and output from pin |
| XCH A, PORTn | Transfer pin data to accumulator. Transfer accumulator data to output latch. (Output buffer remains off.) | Exchange data between output latch and accumulator |
| INCS PORTn | Latch data resulting from adding 1 to pin data in output latch | Increment the output latch contents by one |
| SET1 PORTn.bit<br>CLR1 PORTn.bit<br>SKTCLR PORTn.bit | Although the specified bit output latch is rewritten as designated by the instruction, the output latch of other bits is undefined. | Change the output pin state according to a given instruction |

5.1.5 Internal pull-up resistor

The μPD75402A enables an internal pull-up resistor to be contained in port pins except P00 or P10.

Internal pull-up resistors can be specified by using software or mask option as listed in Table 5.1-4.

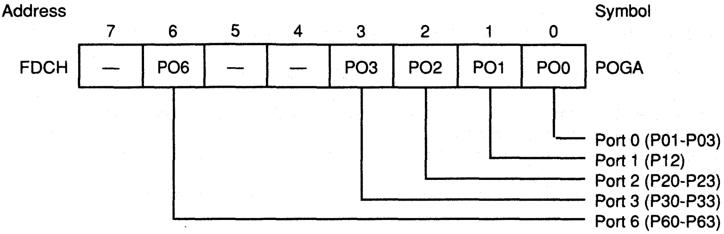
The pull-up resistor specification register (POGA) to specify internal pull-up resistor in ports 0-3 and 6 is a write-only 8-bit register. Fig. 5.1-7 shows the POGA format. POGA is set by using an 8-bit write instruction. It cannot be read or handled bitwise. When RESET is input, POGA is cleared.

Internal pull-up resistor in port 3 can be specified only for the pins for which the input mode is selected. The pins for which the output mode is selected do not contain an internal resistor regardless of how POGA is set.

**Table 5.1-4 Internal Pull-up Resistor Specification Method for Ports**

| Port (pin name) | Internal pull-up resistor specification method | Specification bit |
|---|---|---|
| Port 0 (P01-P03) (Note 1) | Use software in 3-bit units | POGA.0 |
| Port 1 (P12) (Note 1) | Use software in 1-bit units | POGA.1 |
| Port 2 (P20-P23)<br>Port 3 (P30-P33)<br>Port 6 (P60-P63) | Use software in 4-bit units | POGA.2<br>POGA.3<br>POGA.6 |
| Port 5 (P50-P53) | Use mask option bitwise | — |

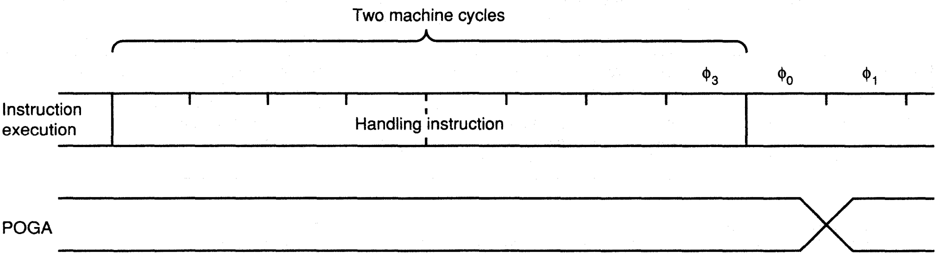
Note 1: Internal pull-up resistor cannot be contained in the P00 or P10 pin.



| | Selection |
|---|---|
| 0 | Internal pull-up resistor is not contained |
| 1 | Internal pull-up resistor is contained |

**Fig. 5.1-7 Pull-up Resistor Specification Register Format**

Fig. 5.1-8 shows the pull-up resistor change timing when the pull-up resistor specification registre (POGA) is set.



Remarks: $\phi_0$ to $\phi_3$ are internal operation timing clocks

**Fig. 5.1-8 Internal Pull-up Register Change Timing**

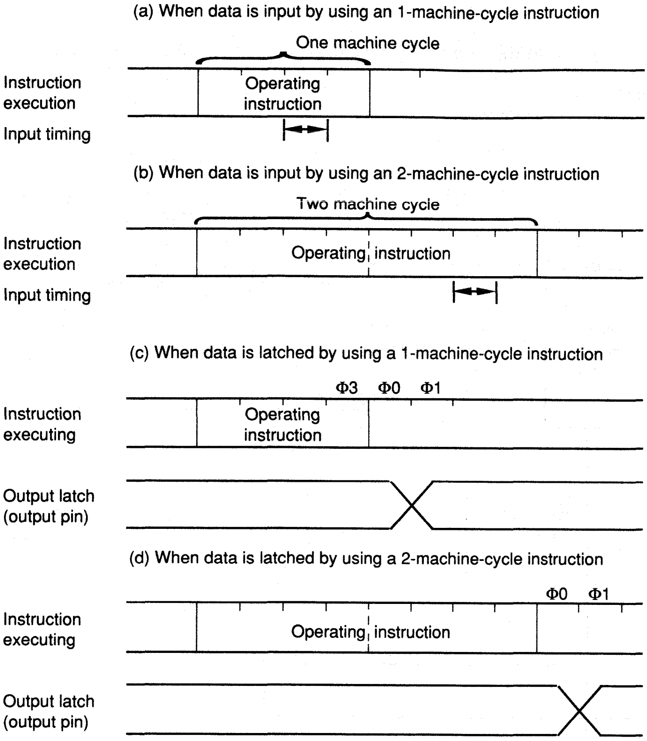
After internal pull-up resistor is specified by rewriting POGA, consider the extenal load capacitance and execute NOP instruction, etc., to stablilize the pin level before executing an input/output instruction.

Example: Input after internal pull-up resistor is specified for port 1.

```
MOV     XA, #02H    ;  Internal pull-up resistor for port 1
MOV     POGA, XA   ┐
 :                 │
 :                 ├ Consider external load capacitance and wait until the pin level becomes stable.
 :                 │
IN      A, PORT1   ┘
```

5.1.6 Digital input/output port input/output timings

Fig. 5.1-9 shows the data output timing to an output latch and the pin or output latch data read timing onto the internal bus.



(a) When data is input by using an 1-machine-cycle instruction

(b) When data is input by using an 2-machine-cycle instruction

(c) When data is latched by using a 1-machine-cycle instruction

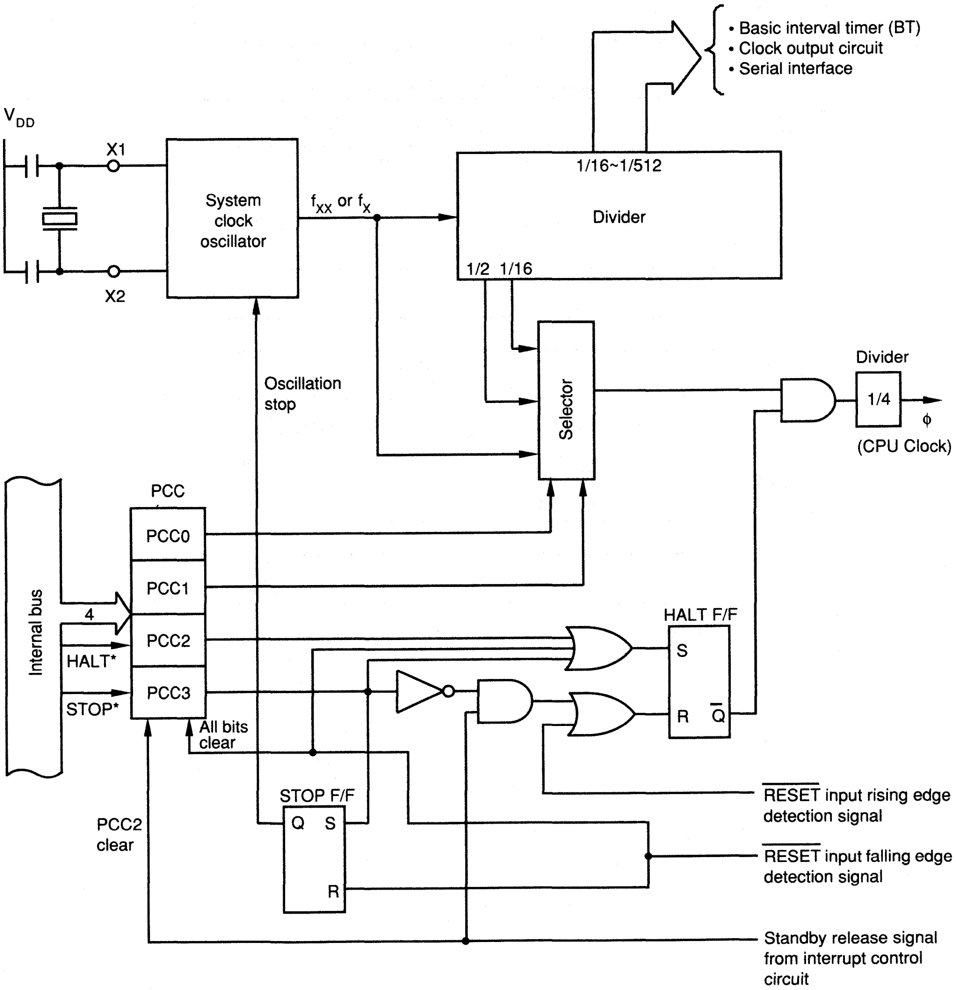
(d) When data is latched by using a 2-machine-cycle instruction

**Figure 5.1–9 Digital Input / Output Port Input / Output Timing**

### 5.2 Clock Generator

The clock generator supplies clocks to the CPU and peripheral hardware controls the CPU operation mode.

5.2.1 Clock generator configuration
Fig. 5.2-1 shows the clock generator block diagram.



Remarks: 1. $f_{XX}$ = crystal or ceramic oscillation frequency
2. $f_X$ = external clock frequency
3. The instruction marked an asterisk (*) is executed.
4. PCC: Processor clock control register

**Fig. 5.2-1 Clock Generator Block Diagram**

5.2.2 Clock generator function and operation

The clock generator generates the CPU clock and clocks supplied to the peripheral hardware and controls the CPU operation mode such as the standby mode.

The clock generator operation is determined by the processor clock control register (PCC).

When $\overline{\text{RESET}}$ is input, PCC is set to 0000 and the μPD75402A operates in the low speed mode (15.3 μs at 4.19 MHz).

Clocks are provided by dividing system clock generator output ($f_{XX}$ (crystal or ceramic oscillation) or $f_X$ (external clock)) by the divider to the peripheral hardware.

In the description to follow, only $f_{XX}$ is used to represent the clock speed. Replace it with $f_X$ for the external clock.

Block operation is explained below:

(1) Processor clock control register (PCC)

The processor clock control register (PCC) is a 4-bit register to select the CPU clock and control the CPU operation mode. Fig. 5.2-2 shows the PCC format.

When bit 3 or 2 is set to 1, the standby mode (STOP or HALT) is selected. When the standby mode is released by the standby release signal, bits 3 and 2 are automatically cleared and normal operation mode is selected. (For details, see Chapter 7.)

PCC bit 1 is set by using a 4-bit memory handling instruction.

Bits 3 and 2 are set to 1 by using the STOP and HALT instructions, respectively.

When $\overline{\text{RESET}}$ is input, the PCC is cleared (0000).

Example 1: Set machine cycle to 1.91 μs (at 4.19 MHz).

```
MOV     A, #0010B
MOV     PCC, A
```

Example 2: Select the STOP mode. (Be sure to write a NOP instruction following the STOP or HALT instruction.)

```
STOP
NOP
```

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

CPU clock selection bits

| | | The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz | |
|---|---|---|---|
| | | CPU clock frequency | machine cycle |
| 0 | 0 | Φ = $f_X$/64 (65.5 kHz) | 15.3 μs |
| 0 | 1 | Undefined | – |
| 1 | 0 | Φ = $f_X$/8 (524 kHz) | 1.91 μs |
| 1 | 1 | Φ = $f_X$/4 (1.05 MHz) | 0.95 μs |

$f_X$ : Main system clock oscillator output frequency

CPU operation mode control bits

| | | |
|---|---|---|
| 0 | 0 | Normal operating mode |
| 0 | 1 | HALT mode |
| 1 | 0 | STOP mode |
| 1 | 1 | Undefined |

**Figure 5.2–2 Processor Clock Control Register Format**

(2) System clock oscillators

The main system clock oscillator uses a crystal oscillator (4.194304 MHz standard) or ceramic oscillator connected to the X1 and X2 pins.

An external clock can also be input

(a) Crystal or ceramic resonator (b) External clock

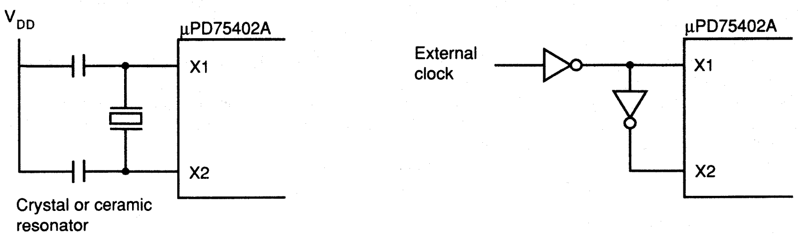


**Fig. 5.2-3 External Circuit to Main System Clock Oscillator**

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

5.2.3 CPU clock selection

The CPU clock is supplied to the µPD75402A internal CPU. The reciprocal of the clock becomes the minimum instruction execution time (which is defined to be one machine cycle in the manual).

The µPD75402A enables to be selected among three steps by setting the PCC. This means that the minimum instruction execution time can be selected among three steps by using a single system clock oscillation frequency $f_{XX}$. This function enables CPU operation at high speed (0,95 µs at 4.19 MHz) when the supply voltage is 5 V or at low speed (15.3 µs at 4.19 MHz) in the backup mode, thereby reducing consumption current and performing operation at low voltage.

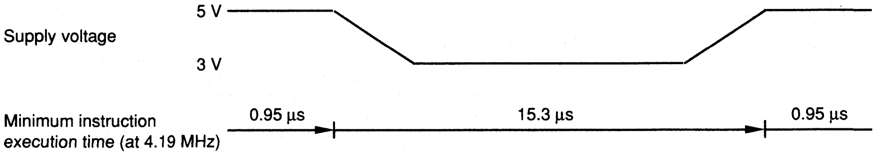
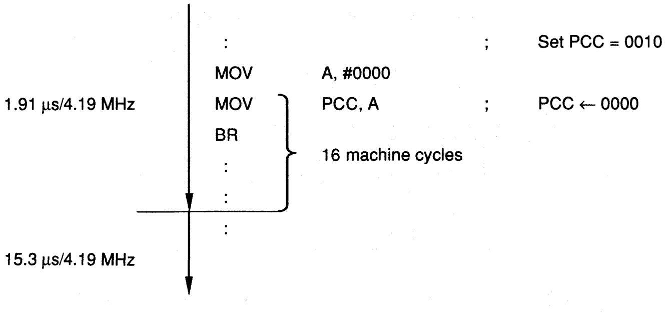


**Fig. 5.2-4 Minimum Instruction Execution Time Change Application**

Although ϕ is changed by using a 4-bit memory handling instruction as described in (1) above, the CPU operates on the old speed before the change during a number of machine cycles which are listed in Table 5.2-1, after the PCC is changed.

Example:



```
        :                       ;   Set PCC = 0010
        MOV     A, #0000
        MOV     PCC, A          ;   PCC ← 0000
        BR
        :                       16 machine cycles
        :
        :
```

Since the PCC is cleared when the $\overline{\text{RES}}$ signal is generated, the μPD75402A starts at the minimum speed (where the operation supply voltage range is wide). This enables normal operation although sufficient supply voltage cannot be obtained after power on reset in a slow voltage rise system (to which a high capacitance capacitor is connected).

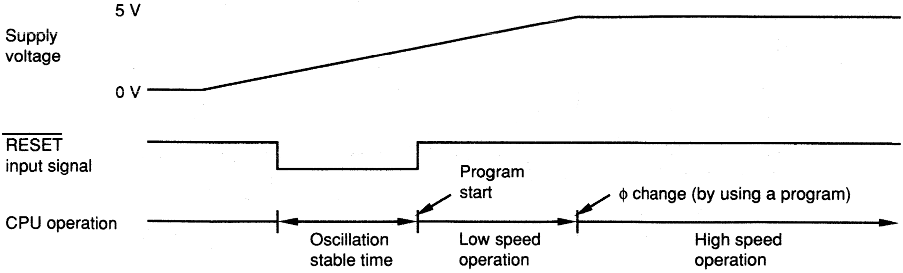


**Fig. 5.2-5 φ Change after Power on Reset**

**Table 5.2-1 Maximum Time Required for CPU Clock Change**

| PCC before φ change | PCC after φ change | Maximum number of machine cycles required for φ change | Maximum time required for φ change (Note) (when $f_{XX}$ = 4.19 MHz) |
|---|---|---|---|
| 0000 | 0010 | 1 | 16μs |
| | 0011 | 1 | |
| 0010 | 0000 | 8 | |
| | 0011 | 8 | |
| 0011 | 0000 | 16 | |
| | 0010 | 16 | |

Note: When the standby mode is not entered until φ has been changed

### 5.3 Clock Output Circuit

(1) Clock output circuit configuration
Fig. 5.3-1 shows a clock output circuit block diagram.

(2) Clock output circuit function
The clock output circuit, which outputs clock pulses from the P22/PCL pin, is used to supply clock pulses to remote control output or peripheral LSIs.
Clock pulses are output in the following sequence:
(a) Clock output frequency is selected. Clock output is disabled.
(b) 0 is written into P22 output latch.
(c) Port 2 input/output mode is placed in output mode.
(d) Clock output is enabled.

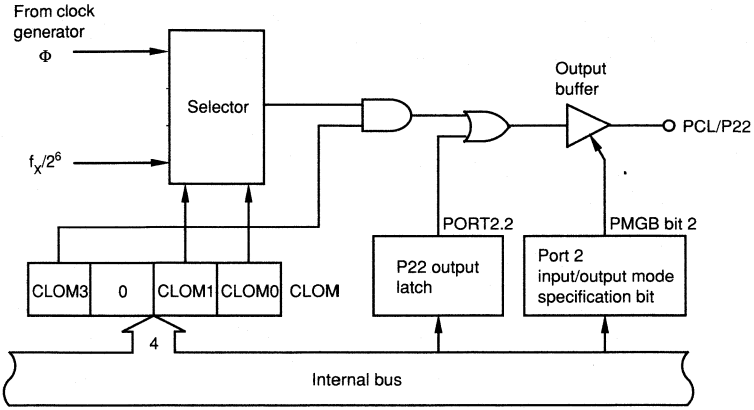


**Fig. 5.3-1 Clock Output Circuit Block Diagram**

Remarks: The circuit is designed so as not to output a spurious short-width pulse when changing between clock output enable and disable.

(3) Clock output mode register (CLOM)
CLOM is a 4-bit register to control clock output.
CLOM is set by using a 4-bit memory operation instruction. It cannot be read.

Example: To output clock Φ from PCL/P22 pin.

```
MOV    A, #1000B
MOV    CLOM, A
CLR1   PORT2.2
MOV    XA, #04H
MOV    PMGA, XA   ;    Set Port2 to Outputmode
```

When the $\overline{\text{RESET}}$ signal is generated, CLOM is cleared and clock output is disabled.

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

Clock output frequency selection bit

| CLOM1 | CLOM0 | |
|---|---|---|
| 0 | 0 | Φ output (Note) (1.05 MHz, 524 kHz, 65.5 kHz) |
| 0 | 1 | Undefined, do not use |
| 1 | 0 | Undefined, do not use |
| 1 | 1 | $f_X/2^6$ output (65.5 kHz) |

Note: Φ is the CPU clock selected by setting PCC.

Clock output enable/disable bit

| CLOM3 | |
|---|---|
| 0 | Output disabled |
| 1 | Output enabled |

Caution: Be sure to write 0 into CLOM bit 2.

**Fig. 5.3-2 Clock Output Mode Register Format**

Example 1: Output 65.5 kHz (during 4.19 MHz operation) clock from the PCL/P22 pin. (Output clock when the PCL/P22 pin is in the high impedance state.)

```
MOV     A, #1011B
MOV     CLOM, A        ; CLOM = 1011B
CLR1    PORT2.2        ; P22 ← 0
MOV     XA, #04H
MOV     PMGB, XA       ; PMGB = 00000100B
```

Example 2: Output Φ (Output clock when the PCL/P22 pin is low.)

```
MOV     A, #0
OUT     PORT2, A       ; P22 ← 0
MOV     XA, #04A
MOV     PMGB, XA
MOV     A, #1000B
MOV     CLOM, A        ; CLOM ← 1000B
```

(4) Application example to remote control output

The μPD75402A clock output function is applicable to remote control output. Remote control output carrier frequency is selected by using the clock frequency selection bit of the clock output mode register. Pulse output is enabled or disabled under software control by using the clock output enable/disable bit.

It is designed so that a spurious short-width pulse will not be produced when changing between clock output enable and disable.

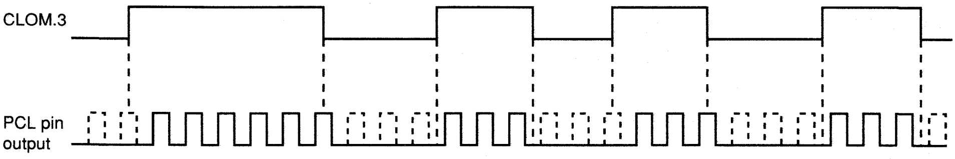


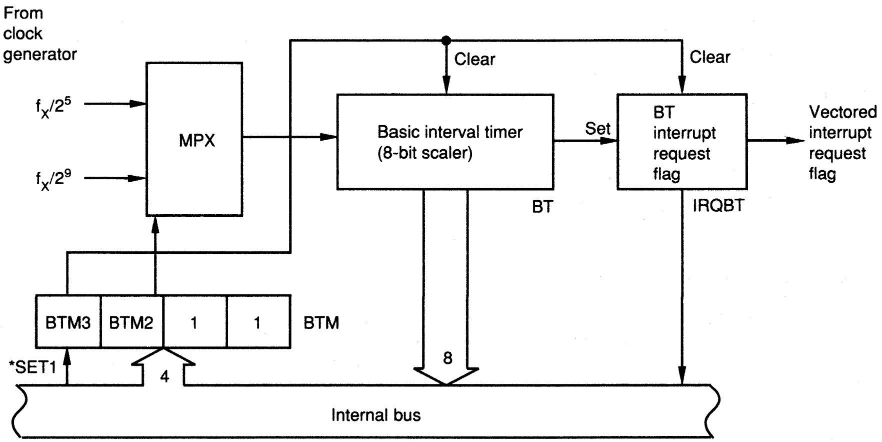
Fig. 5.3-3 Remote Control Output Application Example

### 5.4 Basic Interval Timer

The μPD7500X contains an 8-bit basic interval timer which has the following functions:
(a) Reference time generation (four types of time interval)
(b) Wait time selection and count when standby mode is released.
The basic interval timer can also be applied to a watchdog timer to detect program overrun.

5.4.1 Basic interval timer configuration

Fig. 5.4–1 shows the configuration of the basic interval timer



Remarks: * denotes instruction executing.

**Fig. 5.4–1 Basic Interval Timer Configuration**

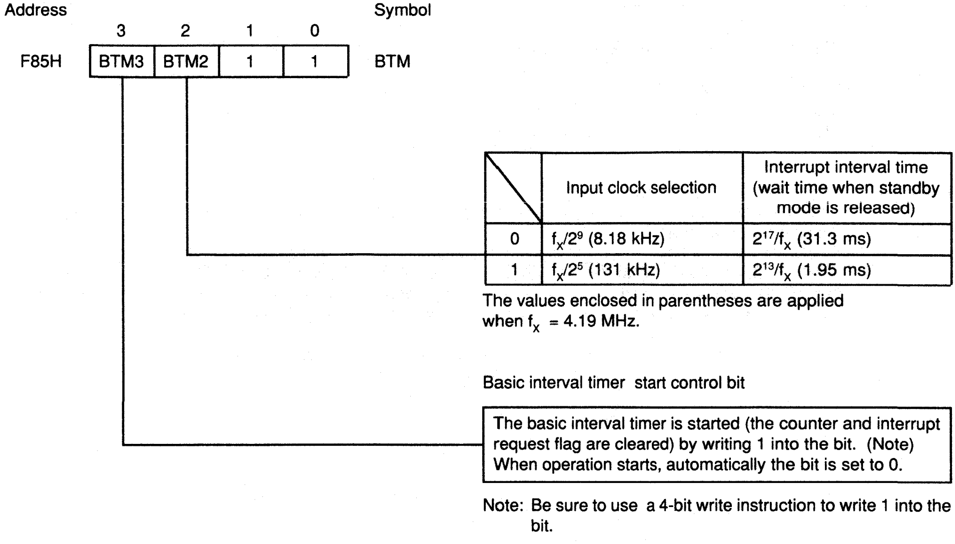
5.4.2. Basic interval timer mode register (BTM)

BTM is a 4-bit register for controlling operation of the basic interval timer.
BTM is set by using a 4-bit memory operation instruction.
Bit 3 can be set individually by using a bit operation instruction.

Example 1: To set the interrupt generation interval to 1.95 ms (4.19 MHz).

```
MOV             A, # 1111B
MOV   BTM. A  ;  BTM ← 1111B
```

When bit 3 is set to 1, the basic interval timer contents are cleared. At the same time, the basic interval timer interrupt request flag (IRQBT) is also cleared. (The basic interval timer starts.)
When the $\overline{\text{RESET}}$ signal is generated, the BTM contents are cleared and interrupt request signal generation is set for the longest interval.



|  | Input clock selection | Interrupt interval time (wait time when standby mode is released) |
|---|---|---|
| 0 | $f_X/2^9$ (8.18 kHz) | $2^{17}/f_X$ (31.3 ms) |
| 1 | $f_X/2^5$ (131 kHz) | $2^{13}/f_X$ (1.95 ms) |

The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz.

Basic interval timer start control bit

The basic interval timer is started (the counter and interrupt request flag are cleared) by writing 1 into the bit. (Note) When operation starts, automatically the bit is set to 0.

Note: Be sure to use a 4-bit write instruction to write 1 into the bit.

Caution: Be sure to write 1 into bits 1 and 0 (11).

**Fig. 5.4-2 Basic Interval Timer Mode Register Format**

5.4.3. Basic interval timer operation

The basic interval timer (BT) is always increment when a clock is input from the clock generator. When it overflows, the interrupt request flag (IRQBT) is set. The BT count operation cannot be stopped.
The interrupt generation interval can be selected among two types of time by setting the BTM. (See Fig. 5.4-2.)
The basic interval timer and interrupt request flag can be cleared by setting BTM bit 3 to 1. (Interval timer start instruction)
The basic interval timer (BT) count state can be read by using an 8-bit handling instruction. Data cannot be written into the basic interval timer.

Caution: To read the basic interval timer count contents, execute a read instruction twice to prevent reading unstable data during count update. Compare the first and second read contents. If a proper value is obtained, use the second read contents as the read result. If they differ completely, retry from the beginning.

Example: Read the BT count contents.

```
         MOV    HL, #TEMP    ; Store table address in HL
LOOP:    MOV    XA, BT       ; First read
         MOV    TEMP, XA     ; Save the read value
         MOV    XA,  BT      ; Second read
         SKE    A, @HL       ; Compare the low-order four bits
         BR     LOOP
         INCS   L
         XCH    A, X
         SKE    A, @HL              ; Compare the high-order four bits
         BR     LOOP
```

5.4.4 Basic interval timer application examples

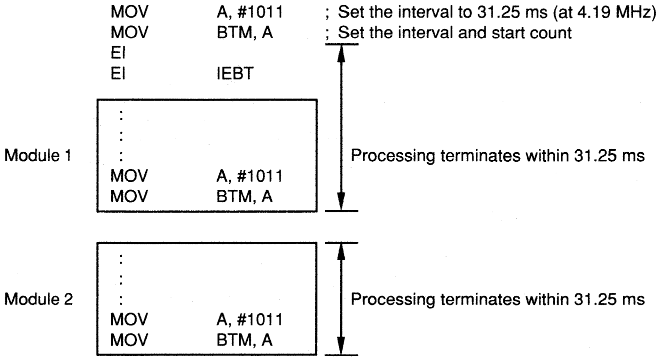
Example 1: Enable basic interval timer interrupt and set the interrupt generation interval to 1.95 ms (at 4.19 MHz).

```
MOV     A, #1111B
MOV     BTM, A          ; Set and start
EI                      ; Enable interrupt
EI      IEBT            ; Enable BT interrupt
```

Example 2: Watchdog timer applciation

To use the basic interval timer as a watchdog timer, use the basic interval timer interrupt (INTBT) generation function every setup time.

First, determine the basic interval timer interrupt generation reference time. Next, divide a program into several modules where processing terminates within the reference time. Clear the counter (BT) and interrupt request flag (IRQBT) at the end of each module. Prepare the program to suppress an interrupt (INTBT) if the program normally operates. That is, if an interrupt occurs, it is decided that the program overruns.



```
            MOV     A, #1011    ; Set the interval to 31.25 ms (at 4.19 MHz)
            MOV     BTM, A      ; Set the interval and start count
            EI
            EI      IEBT

            :
            :
Module 1    :                   Processing terminates within 31.25 ms
            MOV     A, #1011
            MOV     BTM, A

            :
            :
Module 2    :                   Processing terminates within 31.25 ms
            MOV     A, #1011
            MOV     BTM, A
```

**5.5 Serial Interface**

5.5.1 Serial interface configuration

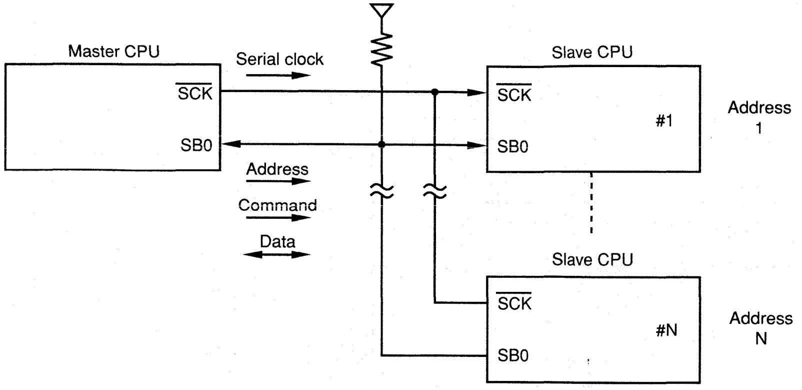
The µPD75402A serial interface is a clocked serial interface (CSI). Fig. 5.5.2 shows the serial interface block diagram.

5.5.2 Serial interface function

The µPD75402A serial interface contains two modes described in (1) and (2) below. The serial clock line $\overline{SCK}$ and serial data bus line SB0 can cover any desired transfer format because the output level can be handled by using software.

(1) 3-wire serial I/O mode
- 3-wire system of serial clock $\overline{SCK}$, serial output SO, and serial input SI.
- Clock synchronization 8-bit transfer operation (send and receive at the same time).
- Enables connection to any of the µPD7500 series, µCOM-75X family, µCOM-78X family, and peripheral I/O devices.

(2) SBI (serial bus interface) mode
- Conforms to the NEC serial bus format.
- Enables the µPD75402A to communicate with a number of devices by using the two lines of serial clock $\overline{SCK}$ and serial data bus SB0.
- Provides the address, command, and data transfer function and the signal distinction function using the hardware. (See Fig. 5.5-2.)
- Provides the wake-up function for handshaking and the acknowledge and busy signal ouput function.



Caution: When the μPD75402A operates as a slave CPU, the μPD75402A addresses are limited to C0H-C7H.

**Fig. 5.5-1 SBI System Configuration Example**

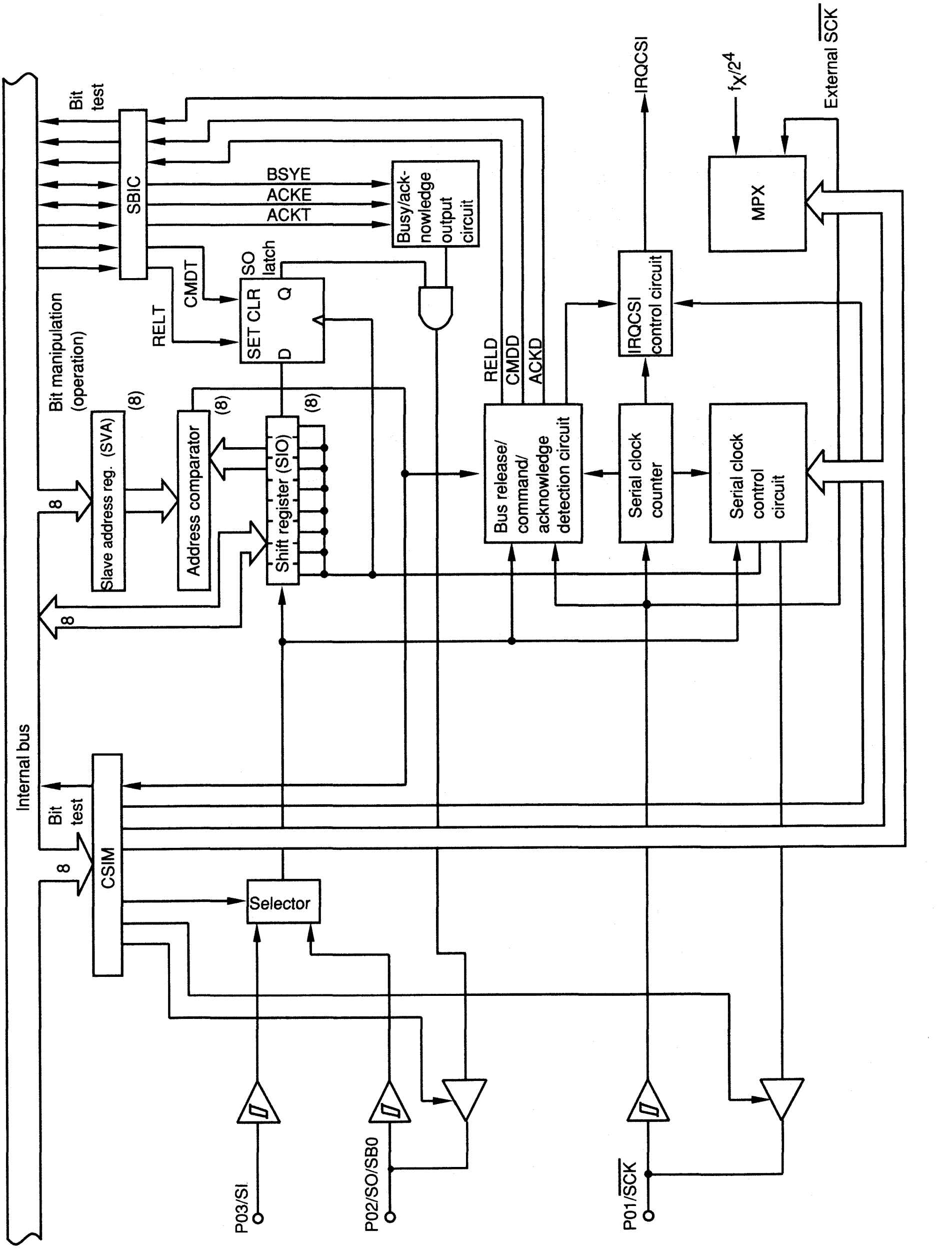


Fig. 5.5.2 Serial Interface Diagram

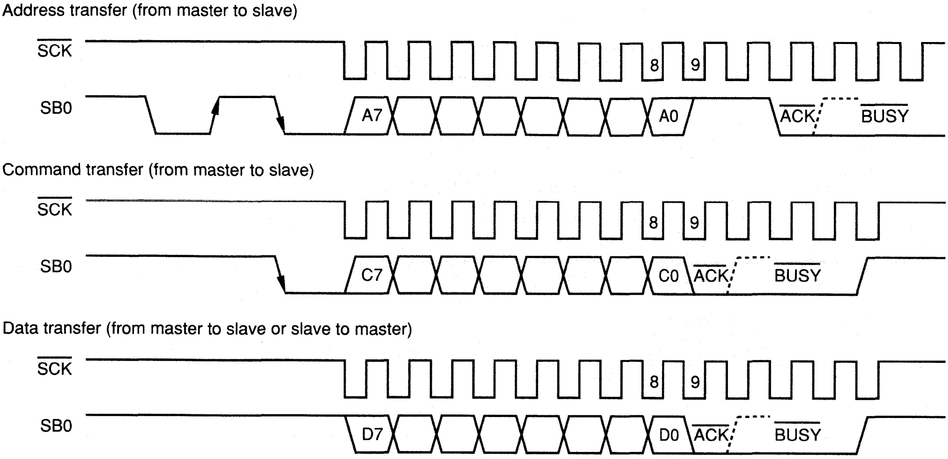


**Fig. 5.5-3 SBI Transfer Timing Chart**

5.5.3 Main register functions

(1) Serial operating mode register (CSIM)

The serial operation mode register (CSIM) consists of eight bits that specify the serial interface operating mode, serial clock, wake-up function, etc.
CSIM is set using an 8-bit memory operation instruction.
The high-order three bits of CSIM can be set bitwise using the bit name.
Bit 6 enables bit test only. Data written into the bit becomes invalid.

Example 1:
To select $f_X/2^4$ for serial clock, generate a serial interrupt IRQCSI at the end of each serial transfer, and make serial transfer in the SBI mode using the SB0 pin as the serial data bus line.

```
MOV    XA, #10001010B
MOV    CSIM, XA          ;   CSIM 10001010B
```

Example 2:
To enable serial transfer conforming to the CSIM contents.

```
SET1   CSIE
```

Wake-up function specification bit (WUP):

(a) When WUP=0
IRQCSI is set each time a serial transfer ends.
WUP is set to 0 during normal transfer.

(b) When WUP = 1
WUP is set to 1 only during the SBI mode. IRQCSI is set only when the address received after the bus is released matches the slave address register (SVA) value (wake-up state). When the received address is not its own, an unnecessary interrupt will not be generated. The $\overline{ACK}$ signal is not output either. The SB0 pin is placed in a high impedance state independent of the SO the latch state.
When the $\overline{RESET}$ signal is generated, all bits are cleared.
Fig. 5.5–4 shows the format of the serial operation mode register.

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FE0H | CSIE | COI | WUP | 0 | CSIM3 | 0 | CSIM1 | 0 | CSIM |

Serial clock selection bits (W)

| | Serial clock | | |
|---|---|---|---|
| CSIM1 | 3-line I/O mode | SBI mode | $\overline{\text{SCK}}$ pin mode |
| 0 | $\overline{\text{SCK}}$ pin input clock from the external | | Input |
| 1 | $f_X/2^4$ (262 kHz) | | Output |

The values enclosed in parentheses are applied when $f_X$ = 4.19 MHz.

Serial interface operating mode selection bits (W)

| CSIM3 | Operating mode | Shift register bit order | SO pin function | SI pin function |
|---|---|---|---|---|
| 0 | 3-line serial I/O mode | $SIO_{7-0}$ ↔ XA<br>(transfer starts at MBS) | SO/PO2<br>(CMOS output) | SI/PO3<br>(input) |
| 1 | SBI mode | $SIO_{7-0}$ ↔ XA<br>(transfer starts at MSB) | SB0/PO2<br>(N-channel open drain input/output) | PO3 input |

Wake-up function specification bit (W)

| | | |
|---|---|---|
| WUP | 0 | IRQCSI is set each time a serial transfer ends in each mode. |
| | 1 | WUP is set to 1 only during the SBI mode. IRQCSI is set only when the address received after the bus is released matches the slave address register data (wake-up state). SB0 is placed in the high impedance state |

Coincidence signal received from address comparator (R (bit test is only enabled)) Note 1, Note 2

| | | |
|---|---|---|
| COI | 0 | Slave address register and shift register data mismatch. |
| | 1 | Slave address register and shift register data match. |

Serial interface operation enable/disable specification bit (W)

| | | Shift register operation | Serial clock counter | IRQCSI flag | SO/SB0 and SI/SB1 pins |
|---|---|---|---|---|---|
| CSIE | 0 | Shift operation is disabled. | Clear | Hold | Port 0 function only |
| | 1 | Shift operation is enabled. | Count operation | Can be set. | Also used for port 0 according to each mode function |

**Fig. 5.5-4 Serial Operation Mode Register Format**

Notes:
1. COI is only valid before serial transfer starts or after it is complete. An undefined value is read during serial transfer.
2. Data written into COI is ignored.
3. Be sure to write 0 into CSIM bits 4, 2, and 0.

Remarks: (W): Data write is only enabled.
(R) : Data read is only enabled.

(2) Serial bus interface control register (SBIC)

The 8-bit serial bus interface control register (SBIC) consists of the serial bus state control bits and flags indicating the states of input data from serial bus.'It is mainly used in the SBI mode.

Fig. 5.5–5 shows the SBIC format. SBIC is set or tested by using a bit operation (manipulation) instruction. When the $\overline{\text{RESET}}$ signal is generated, all the SBIC bits are cleared.

Cautions:

1. SBIC cannot be set by using 4- or 8-bit memory operation instruction.
2. In the 3-line serial I/O mode, use only the following two bits for SO latch control:
   (a) Bus release trigger bit (RELT): To set SO latch
   (b) Command trigger bit (CMDT): To clear SO latch
3. For the bus release, command, acknowledge, and busy signals, see 5.5.5 (3).

Example 1: To output command signal.

```
        SET1    CMDT
```

Example 2: To test RELD and CMDD and determine the receive data type for appropriate processing.

```
        SKF     RELD    ;   RELD test
        BR      !ADRS
        SKT     CMDD    ;   CMDD test
        BR      !DATA
CMD  :  .........       ;   Command transfer
DATA :  .........       ;   Data transfer
ADRS :  .........       ;   Address transfer
```

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FE2H | BSYE | ACKD | ACKE | ACKT | CMDD | RELD | CMDT | RELT | SBIC |

Bus release trigger bit (W)

| | |
|---|---|
| RELT | SO latch is set to 1 by setting the bit. It is used to output bus release signal. After SO latch is set, the bit is automatically cleared. |

Command trigger bit (W)

| | |
|---|---|
| CMDT | SO latch is cleared by setting the bit. It is used to output a command signal. After SO latch is cleared, the bit is automatically cleared. |

Bus release detection flag (R)

| | | |
|---|---|---|
| RELD | Clear condition (RELD=0) | 1 When the transfer start is indicated.<br>2 When the address received after the bus is released does not match the slave address register (SVA) data.<br>3 When the $\overline{\text{RESET}}$ signal is input. |
| | Setting condition (RELD=1) | When the address received after the bus is released matches the slave address register data. (Wake up) |

Command detection flag (R)

| | | |
|---|---|---|
| CMDD | Clear condition (CMDD=0) | 1 When the transfer start is indicated.<br>2 When the bus release signal is detected.<br>3 When the $\overline{\text{RESET}}$ signal is input. |
| | Setting condition (CMDD=1) | When the command signal is detected. |

Acknowledge trigger bit (W)

| | | |
|---|---|---|
| ACKT | Used only after transfer completion | Acknowledge signal is output during one clock period of $\overline{\text{SCK}}$ immediatly after execution of the set instruction. |

Remarks:
1. ACKT is automatically cleared after acknowledge signal is output. 2. ACKT cannot be cleared using software.
3. To set ACKT, set ACKE to 0.

Acknowledge enable bit (R/W)

| | | | |
|---|---|---|---|
| ACKE | 0 | Automatic acknowledge signal output is disabled (the signal can be output by setting the acknowledge trigger bit ACKT). | |
| | 1 | Before transfer completion | Acknowledge signal is output during the ninth clock period of $\overline{\text{SCK}}$ (automatically output by presetting ACKE to 1). |
| | | After transfer completion | Acknowledge signal is output during one clock period of $\overline{\text{SCK}}$ immediately after execution of the set instruction (automatically output by presetting ACKE to 1). |

Acknowledge detection flag (R)

| | | |
|---|---|---|
| ACKD | Clear condition (ACKD=0) | 1 When the transfer is started.<br>2 When the $\overline{\text{RESET}}$ is input. |
| | Setting conditions (ACKD=1) | When the acknowledge signal is detected. |

Synchronous busy enable bit (R/W)

| | | |
|---|---|---|
| BSYE | 0 | Synchronous busy signal output is disabled. Synchronous busy signal output is stopped in synchronization with the $\overline{\text{SCK}}$ falling edge immediately after execution of the clear instruction. |
| | 1 | Synchronous busy signal is output on the $\overline{\text{SCK}}$ falling edge following an acknowledge signal. |

Remarks:
(R): Read is only enabled. (W): Write is only enabled. (R/W): Both read and write are enabled.

**Fig. 5.5-5 SBIC Format**

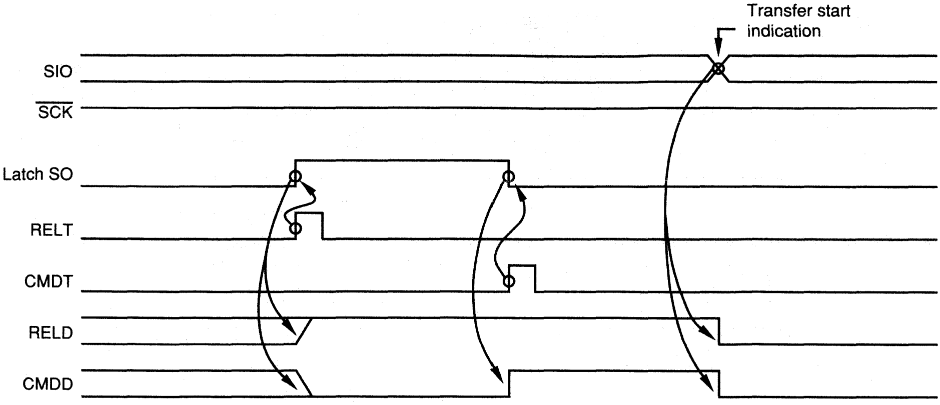
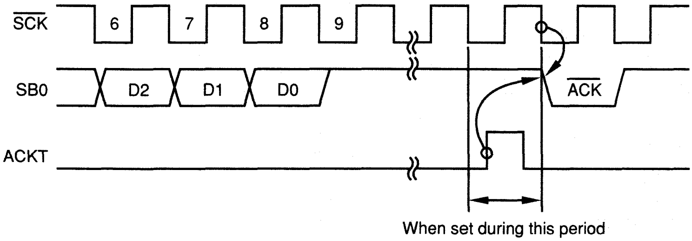


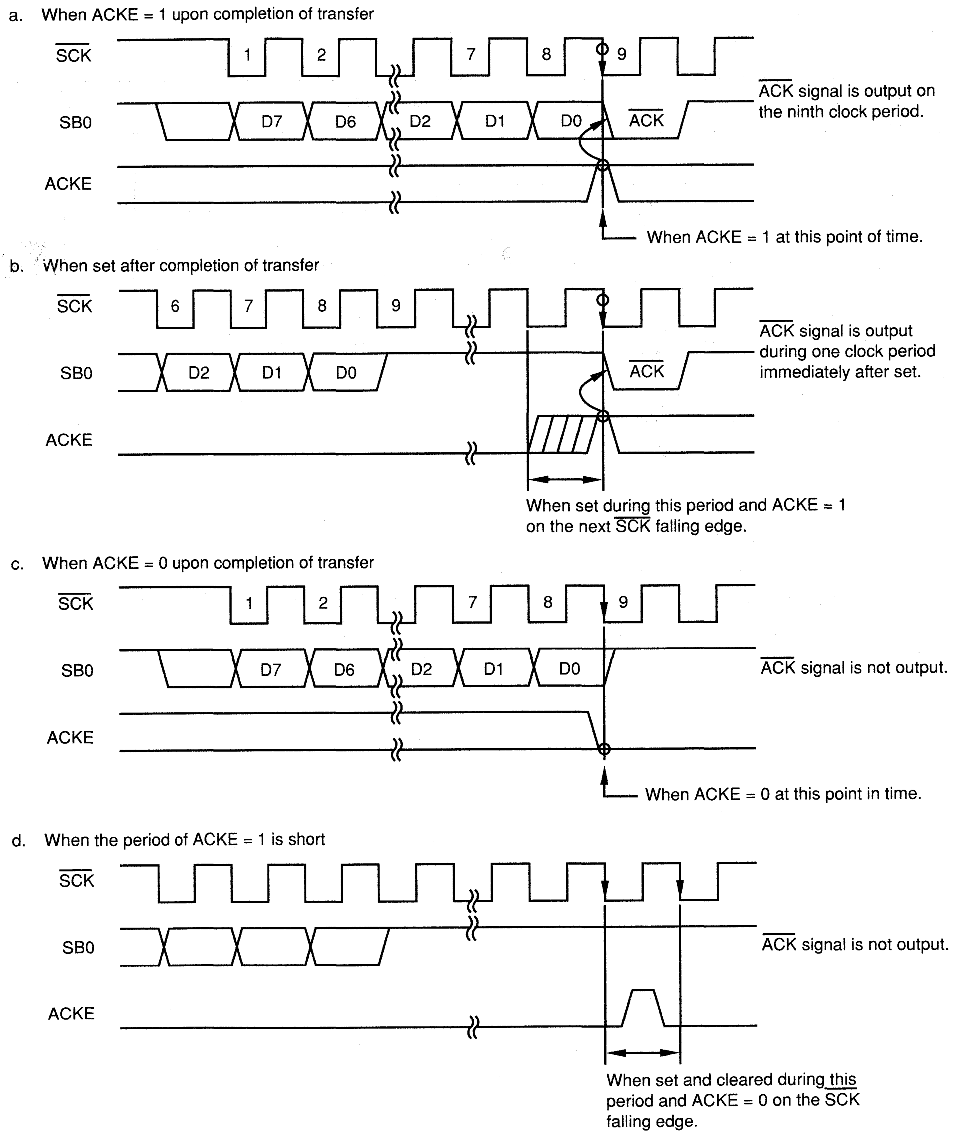
**Figure 5.5–6 RELT, CMDT, RELD, CMDD Operation**

Set after transfer completion



ACK signal is output during one clock period immediately after set
Do not set ACKT before transfer is completed.

**Fig. 5.5-7 ACKT Operation**



a. When ACKE = 1 upon completion of transfer

$\overline{\text{ACK}}$ signal is output on the ninth clock period.

When ACKE = 1 at this point of time.

b. When set after completion of transfer

$\overline{\text{ACK}}$ signal is output during one clock period immediately after set.

When set during this period and ACKE = 1 on the next $\overline{\text{SCK}}$ falling edge.

c. When ACKE = 0 upon completion of transfer

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

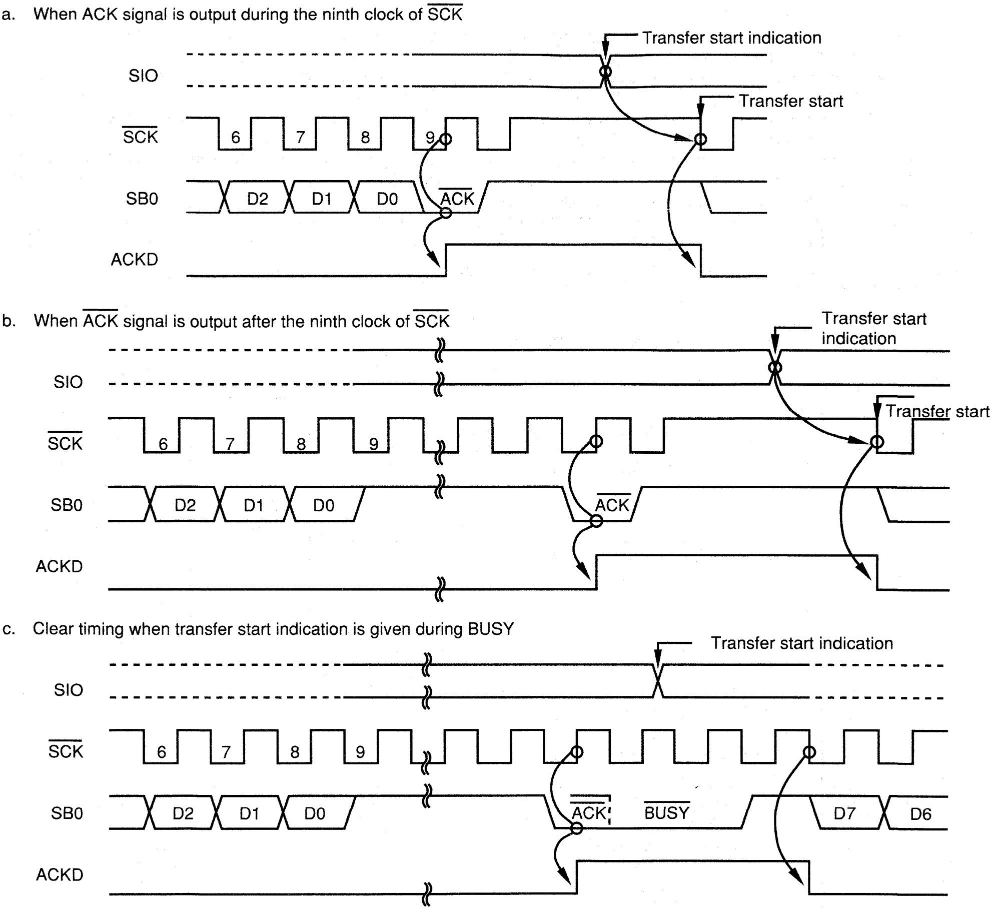
When ACKE = 0 at this point in time.

d. When the period of ACKE = 1 is short

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

When set and cleared during this period and ACKE = 0 on the SCK falling edge.

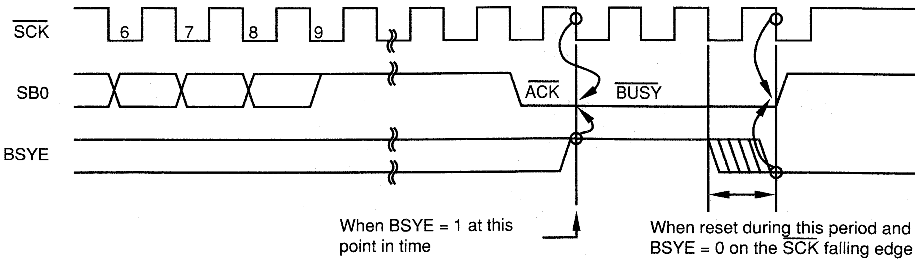
Fig. 5.5-8 ACKE Operation


a. When ACK signal is output during the ninth clock of $\overline{\text{SCK}}$

b. When $\overline{\text{ACK}}$ signal is output after the ninth clock of $\overline{\text{SCK}}$

c. Clear timing when transfer start indication is given during BUSY

**Fig. 5.5-9 ACKD Operation**



When BSYE = 1 at this point in time

When reset during this period and BSYE = 0 on the $\overline{\text{SCK}}$ falling edge

**Figure 5.5-10 BSYE Operation**

(3) Slave address register (SVA)

The slave address register (SVA) is an 8-bit register to set the slave address of the μPD75402A when the μPD75402A is connected to the serial bus as a slave device in the SBI mode.

Since the high-order five bits of SVA are fixed to 11000 on hardware, the μPD75402A assignment addresses are limited to COH – C7H.

The SVA value is compared with the received 8-bit data by the address comparator. If a match is found between them, serial operation mode register (CSIM) bit 6 (COI) is set to 1.

If no match is found when the address is received, the bus release detection flag (RELD) is cleared. When WUP is set to 1, IRQCSI is set to 1 only when a match is found (wakeup detection). This interrupt request informs the μPD75402A that a communication request is sent by master to the μPD75402A.

The SVA is handled by using an 8-bit memory handling instruction. It can only be written.

When the RESET signal is generated not during the standby mode, the SVA value becomes undefined.

Caution: Be sure to write data in the range of COH to C7H into the SVA.

5.5.4 Signals in SBI mode

Table 5.5-1 and 5.5-2 list the signals used in the SBI mode.

**Table 5.5-1 Signals in SBI Mode (I)**

| Signal name | Output device | Definition | Timing chart | Output condition | Flag influence | Explaination |
|---|---|---|---|---|---|---|
| Bus release signal (REL) | Master | SB0 rising edge when SCK = 1 | (Figure 1) | • RELT is set. | • RELD is set.<br>• CMDD is cleared. | The signal is followed by CMD signal output indicating that the send data is an address |
| Command signal (CMD) | Master | SB0 falling edge when SCK = 1 | (Figure 2) | • CMDT is set. | • CMDD is set. | i) After REL signal is output, send data is an address.<br>ii) When REL signal is not output, send data is a command. |
| Acknowledge signal (ACK) | Master/ Slave | Low signal output to SB0 during one clock period of SCK after completion of serial reception | (Figure 3) | 1 ACKE = 1<br>2 ACKT is set. | • ACKD is set. | Completion of reception |
| Busy signal (BUSY) | Slave | (Synchronous busy signal) Low signal output to SB0 following acknowledge signal | | • BSYE = 1 | – | Serial reception cannot be done because processing is being performed. |
| | | (Asynchronous busy signal) Low signal output to SB0 (except during serial transfer). It is not synchronized with SCK. | | • CMDT is set. | – | |
| Ready signal (READY) | Slave | High signal output to SB0 before start or after completion of serial transfer | | 1 BSYE = 0<br>2 Execution of SIO data write instruction (transfer start indication) | – | Serial reception can be done. |

**Table5.6–2 Signals in SBI Mode (II)**

| Signal name | Output device | Definition | Timing chart | Output condition | Flag influence | Explanation |
|---|---|---|---|---|---|---|
| Serial clock (SCK) | Master | Synchronous clock to output address, command, data, ACK signal, synchronous BUSY signal, etc. Address, command, or data is transferred with the first eight. | (Figure 4) | Execution of SIO data write instruction when CSIE =1 (serial transfer start indication) (Note 2) | IRQCSI is set (on the rising edge of ninth clock). (Note 1) | Signal output timing to serial data bus |
| Address (A7–0) | Master | 8-bit data transferred in synchronization with SCK after REL and CMD signals are output. | (Figure 5) | | (Note 1) | Slave device address value on serial bus |
| Command (C7–0) | Master | 8-bit data transferred in synchronization with SCK after CMD signal only is output (REL signal is not output). | (Figure 6) | | None | Indication message sent to slave device |
| Data (D7–0) | Master or Slave | 8-bit data transferred in synchronization with SCK when neither REL nor CMD signal is output. | (Figure 7) | | None | Numeric data processed by slave or master device |

Notes: 1. When WUP = 0, IRQCSI is always set on the ninth clock SCK rising edge.
When WUP = 1, IRQCSI is set only when the received address matches the value in the slave address register (SVA).
2. In the BUSY state, transfer is started after the READY state is set.

Figure 1:

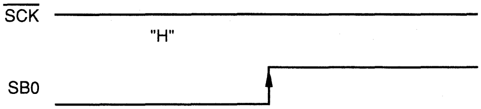


Figure 2:

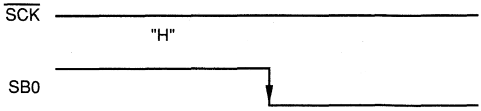
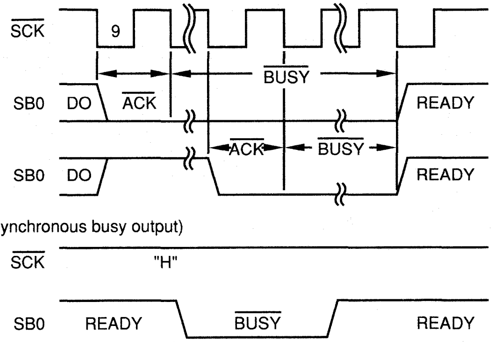


Figure 3:



(Synchronous busy output)

(Asynchronous busy output)

Figure 4:

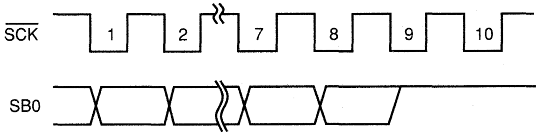


Figure 5:

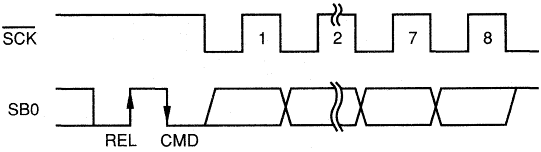


Figure 6:

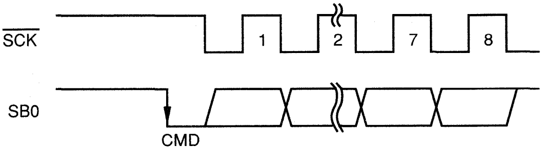
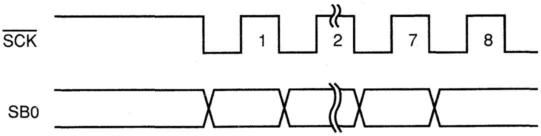


Figure 7:



5.5.5 Serial interface operation

The serial interface operates in any of the following three modes:

- Operation stop mode
- 3-wire serial I/O mode
- SBI mode

**Table 5.5-3 Serial Interace Operation Mode**

| CSIE | CSIM3 | Operation mode |
|---|---|---|
| 0 | — | Operation stop mode |
| 1 | 0 | 3-wire serial I/O mode |
| 1 | 1 | SBI mode |

(1) Operation stop mode

When CSIE = 0, the serial interface is placed in the operation stop mode. In this mode, serial transfer is not made.
The operation stop mode is selected to reduce power consumption when the serial interface is not used.
In the mode, the shift register does not perform shift operation and can be used as a normal 8-bit register.
When CSIM1, CSIM0 = 00 ($\overline{SCK}$ = external clock), the P01/SCK pin is placed in high impedance; when CSIM1, CSIM0 are not set to 00 ($\overline{SCK}$ = internal clock), the P01/SCK pin outputs high level.
The P02/SO/SB0 and P03/SI pins are placed in high impedance and used only for the input port function.
When $\overline{RESET}$ is input, the operation stop mode is selected. To select the mode during serial transfer, previously disable a serial interrupt. After selecting the mode, clear the interrupt request flag (IRQCSI).

(2) 3-line serial I/O mode

The 3-line serial I/O mode is compatible with the mode used by the µPD7500 series or other µCOM-75X family devices.
Fig. 5.5-11 shows the 3-line serial I/O mode operation timing.
Serial transfer start is indicated by executing an instruction to write data into the shift register (SIO), MOV or XCH.
Be sure to give a start indication when CSIE = 1.

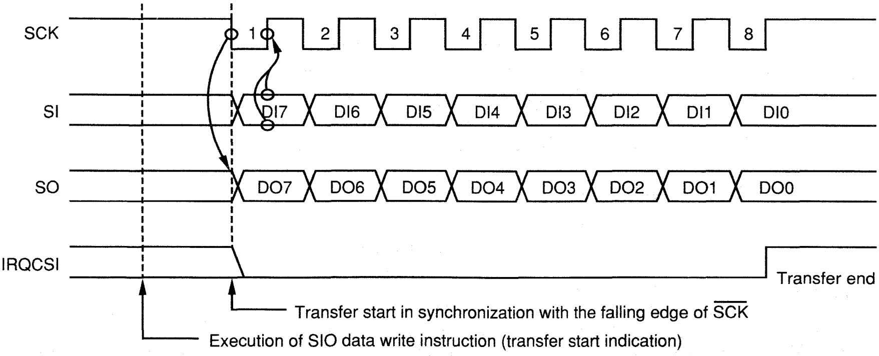

SCK
1
2
3
4
5
6
7
8
SI
DI7
DI6
DI5
DI4
DI3
DI2
DI1
DI0
SO
DO7
DO6
DO5
DO4
DO3
DO2
DO1
DO0
IRQCSI
Transfer end
Transfer start in synchronization with the falling edge of $\overline{SCK}$
Execution of SIO data write instruction (transfer start indication)

**Fig. 5.5-11 3-line Serial I/O Mode Timing**

In this mode, shift register shift operation is performed in synchronization with the serial clock ($\overline{SCK}$) falling edge, and send data is held in the SO latch and output from the SO pin. The receive data input to the SI pin is latched in the shift register on the rising edge of $\overline{SCK}$.
Since the SO pin is used as a CMOS output and outputs the SO latch state, the SO pin output state can be handled by setting the RELT and CMDT bits. However, do not try to change it during serial transfer.
A serial clock can be selected from among four clocks, as listed in Table 5.5-4, by setting the mode register.
Normally, shift operation is started by:
1) Setting the operating mode and serial clock selection data in the mode register (CSIM)
2) Setting transfer data in the shift register (SIO) (serial operation is started by executing the SIO data write instruction)
Serial transfer automatically stops at the end of 8-bit transfer, and the interrupt request flag (IRQCSI) is set.

**Table 5.5-4 Serial Clock Selection and Application**

| Mode register | Serial clock | | Timing at which shift register read/write or serial transfer start is enabled | Application |
|---|---|---|---|---|
| CSIM1 | Source | Serial clock mask | | |
| 0 | External $\overline{SCK}$ | Serial clock is automatically masked when 8-bit data transfer terminates | Only when serial transfer stops (Note) or $\overline{SCK}$ is high | Slave CPU |
| 1 | $f_{XX}/2^4$ | | Only when serial transfer stops (Note) or $\overline{SCK}$ is high | Medium speed serial transfer |

Note: The clause "when serial transfer stops" means the operation stop mode or serial clock mask state after 8-bit data transfer.

The shift register is read or written by using an 8-bit transfer instruction.

Example: Transfer RAM data specified in the HL register pair to SIO. At the same time, read SIO data into a given accumulator and start serial transfer.

```
MOV     XA, @HL    ; Read send data from RAM
XCH     XA, SIO    ; Exchange send data and receive data and start transfer.
```

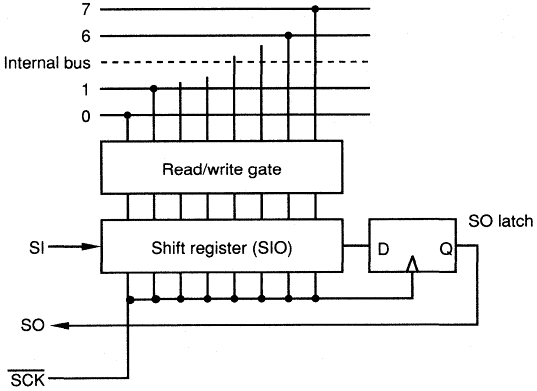


**Fig. 5.5-12**

(3) SBI mode

The SBI mode enables the μPD75402A to communicate with a number of devices by using the $\overline{SCK}$ and SB0 lines. Figs. 5.5-13 to 5.5-15 show the timing of operations according to the type of data to be transferred.

In the SBI mode, indication of serial transfer start is also given by executing the instruction to write data into the shift register (SIO), MOV or XCH. Be sure to give a start indication when CSIE = 1.

Shift register shifting is made in synchronization with the serial clock ($\overline{SCK}$) falling edge, and send data is held in the SO latch and output from the SB0/P02 pin starting at MSB. Receive data input to the SB0 pin is latched in the shift register on the rising edge of $\overline{SCK}$.

The SB0 pin specified for the serial data bus is used as N-channel open drain input/output and needs pull-up. When data is received, the N-channel transistor must be turned off.

By writing FFH into SIO and shifting it, the N-channel transistor can always be turned off during transfer. When the wake-up function specification bit (WUP) is set to 1, however, the N-channel transistor is always turned off, and FFH need not be written into SIO before reception.

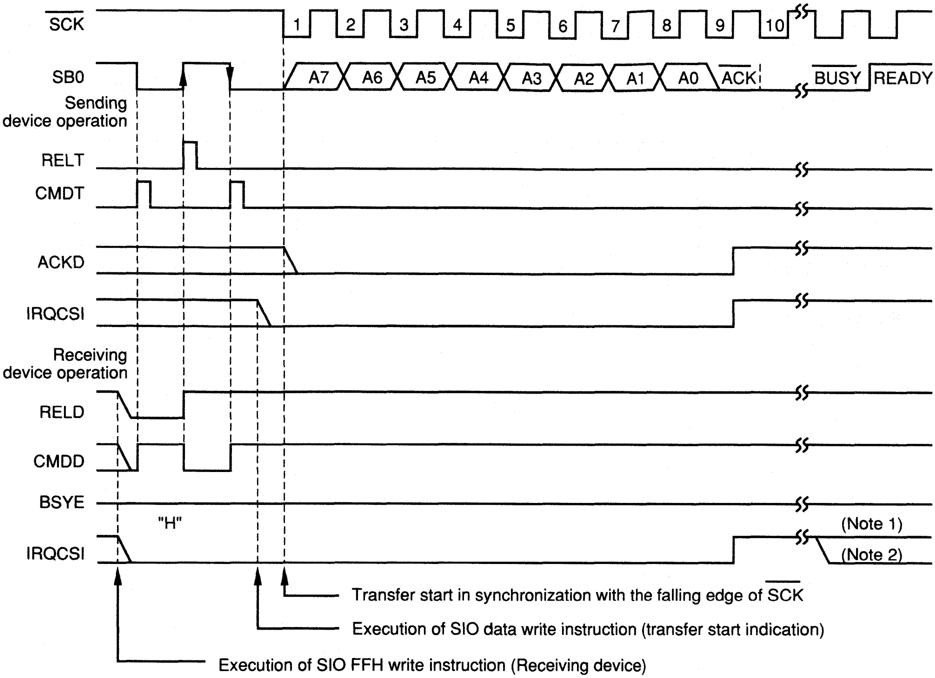

SCK
1
2
3
4
5
6
7
8
9
10
SB0
A7
A6
A5
A4
A3
A2
A1
A0
ACK
BUSY
READY
Sending device operation
RELT
CMDT
ACKD
IRQCSI
Receiving device operation
RELD
CMDD
BSYE
"H"
IRQCSI
(Note 1)
(Note 2)
Transfer start in synchronization with the falling edge of SCK
Execution of SIO data write instruction (transfer start indication)
Execution of SIO FFH write instruction (Receiving device)

**Fig. 5.5-13 SBI Mode Timing (Address Transfer)**

Note 1: Where SIO's data is read out by MOV instruction, then BSYE flag is cleared.
Note 2: Where SIO's data is exchanged (read/write) by XCH instruction.

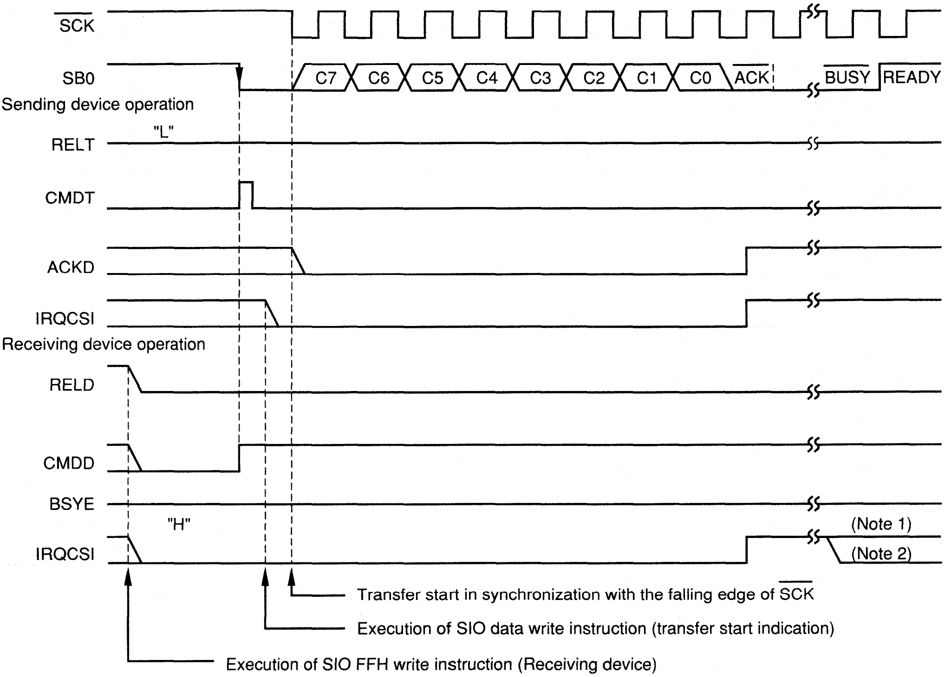

SCK
SB0
C7
C6
C5
C4
C3
C2
C1
C0
ACK
BUSY
READY
Sending device operation
"L"
RELT
CMDT
ACKD
IRQCSI
Receiving device operation
RELD
CMDD
BSYE
"H"
IRQCSI
(Note 1)
(Note 2)
Transfer start in synchronization with the falling edge of SCK
Execution of SIO data write instruction (transfer start indication)
Execution of SIO FFH write instruction (Receiving device)

**Fig. 5.5-14 SBI Mode Timing (Command Transfer)**

Note 1: Where SIO's data is read out by MOV instruction, then BSYE flag is cleared.
Note 2: Where SIO's data is exchanged (read/write) by XCH instruction.

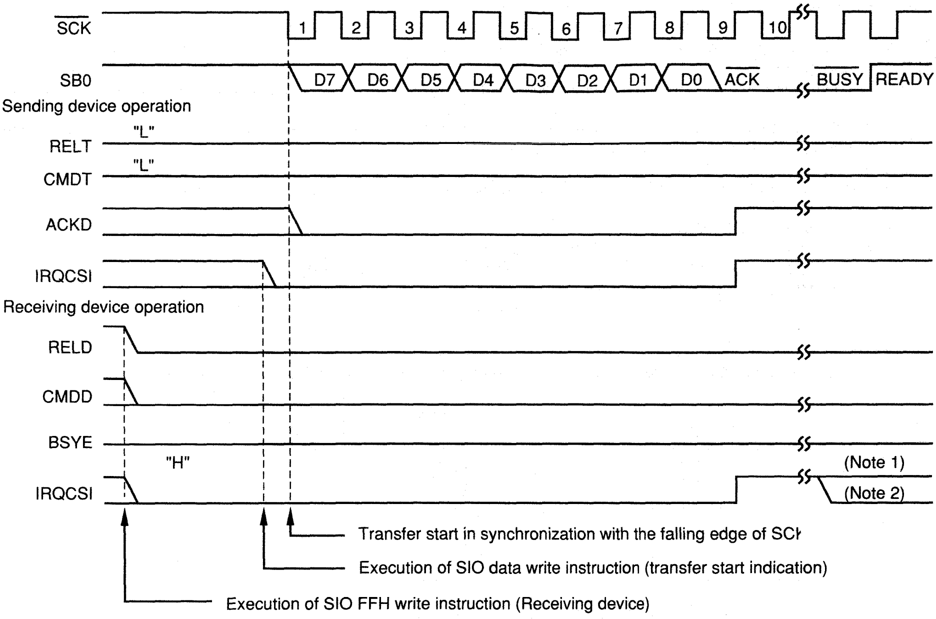

SCK
1
2
3
4
5
6
7
8
9
10
SB0
D7
D6
D5
D4
D3
D2
D1
D0
ACK
BUSY
READY
Sending device operation
RELT
"L"
CMDT
"L"
ACKD
IRQCSI
Receiving device operation
RELD
CMDD
BSYE
"H"
IRQCSI
(Note 1)
(Note 2)
Transfer start in synchronization with the falling edge of SCK
Execution of SIO data write instruction (transfer start indication)
Execution of SIO FFH write instruction (Receiving device)

**Fig. 5.5-15 SBI Mode Timing (Data Transfer)**

Note 1: Where SIO's data is read out by MOV instruction, then BSYE flag is cleared.
Note 2: Where SIO's data is exchanged (read/write) by XCH instruction.

### 5.5.6 Error detection

Since the serial bus SB0 during sending is also read into the sending device shift register (SIO) in the SBI mode, a send error can be detected by comparing the SIO data before the sending starts with that after the sending terminates. In this case, if the two data pieces differ from each other, it is decided that a send error occurred.

5.5.7 Serial interface application

The serial interface for each mode is explained using examples of applications.

The normal serial interface communication sequence is as follows:

1) Set transfer mode. (Set data in CSIM.)
2) Write data into SIO and give transfer start indication. (MOV SIO, XA or XCH XA, SIO. At that time, automatic transfer start indication is given.)
3) After checking that the serial interrupt routine or interrupt request flag (IRQCSI) is set, read receive data, and start transfer.

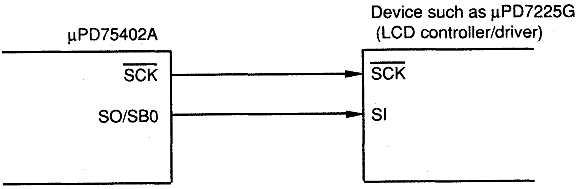
The SBI mode communication sequence is explained in detail in (3) below.

(1) 3-line serial I/O mode

(a) To transfer data starting at MBS using transfer clock of 262 kHz (at 4.19 MHz) (master operation).

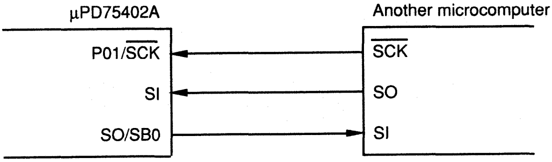
```
Example:  MOV    XA, #10000010B
          MOV    CSIM, XA        ; Transfer mode is set.
          MOV    XA, TDATA       ; TDATA is the transfer data storage address.
          MOV    SIO, XA         ; Transfer data is set.
                                 ; Transfer is started.
```

Caution:
At the second time or after, transfer can be started by setting data in SIO (MOV SIO, XA or XCH XA, SIO).



In this application example, the μPD75402 SI pin can be used for input.

(b) To transfer data starting at the MSB using external clock (slave operation).



```
Example:  Main routine
          MOV    XA, #80H
          MOV    CSIM, XA        ; Serial operation stop, external clock
          MOV    XA, TDATA
          MOV    SIO, XA         ; Transfer data is set.
                                 ; Transfer is started.
          EI     IECSI
          EI

          Interrupt routine (MBE = 0)
          MOV    XA, TDATA
          XCH    XA, SIO         ; Receive data - send data, transfer start
          MOV    RDATA, XA       ; Receive data is saved.
          RETI
```

(2) SBI mode

An application example of serial data communication in the SBI mode is given. In the example, the μPD75402A can operates as a slave CPU.

Althogh the master can also be changed by using a command, the μPD75402A can not serve as a master CPU.

(a) Serial bus configuration

The serial bus configuration in the application example given here assumes that the μPD75402A is connected to bus lines as one device in the serial bus.

The following two μPD75402A pins are used: Serial data bus SB0 (P02/SO) and serial clock SCK (P01).

Fig. 5.5-16 shows an example of a serial bus configuration.

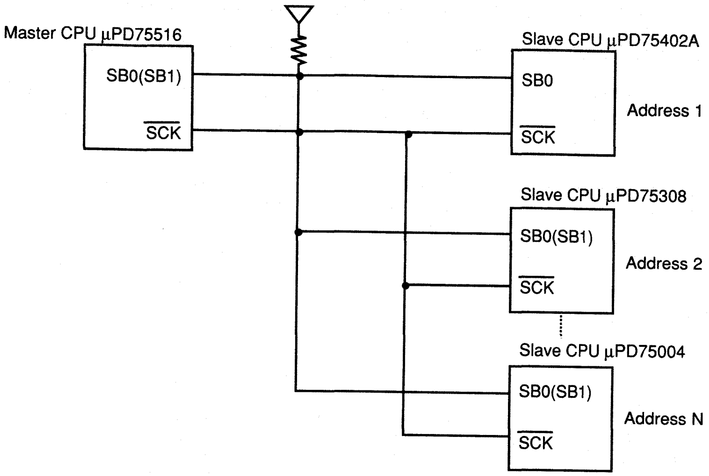


**Fig. 5.5-16 Serial Bus Configuration Example**

(b) Command explanation

Command types

The application example uses the following commands:

1) READ:
Data is transferred from slave to master.
2) WRITE:
Data is transferred from master to slave.
3) END:
WRITE command completion is reported to slave.
4) STOP:
WRITE command stop is reported to slave.
5) STATUS:
The slave state is read.
6) RESET:
The current slave being selected is made unselected.
7) CHGMST:
The master authorization is transferred to the slave.

Communication sequence

The communication sequence between the master and slave is as follows:

1) Communication is started by the master, which sends the address of the slave to communicate with and selects the slave (chip select).
The slave which receives the address returns $\overline{\text{ACK}}$ and communicates with the master. (The slave is placed in selected state.)

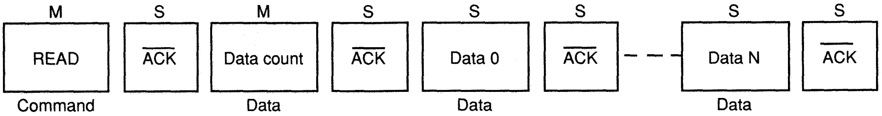
2) Command and data are transferred between the slave selected in 1) and the master.
Since command and data are transferred point-to-point (between the master and specific slave), other slaves must be deselected.

3) Communication terminates when the slave is deselected in either of the following cases:
— When the master sends the RESET command, the selected slave is deselected.
— If the master is changed by using the CHGMST command, the device changed from master to slave is deselected.

Command format

The command transfer formats are shown below:

1) READ command
The READ command reads data from a given slave. The read data count ranges from one to 256 bytes. The master specifies the data count in a parameter. If 00H is specified for the data count, 256-byte data transfer is assumed to be specified.

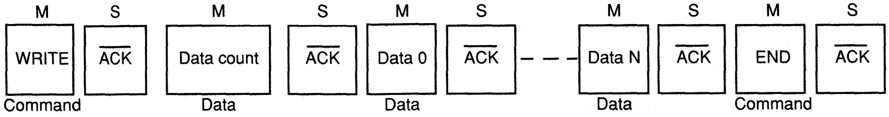


M: Output by master
S: Output by slave

**Fig. 5.5-17 READ Command Transfer Format**

After receiving the data count, the slave returns $\overline{\text{ACK}}$ if the number of data bytes that can be sent is greater than the data count. If it is less than the data count, the slave does not return $\overline{\text{ACK}}$, resulting in an error.
During data transfer, the slave compares the SIO contents before and after data transfer to check that data has been output to the bus normally. If the SIO contents before and after transfer do not match, the slave does not return ACK, resulting in an error.

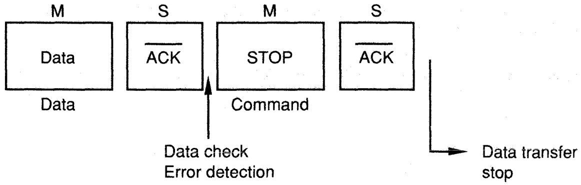
2) WRITE, END, and STOP commands
The WRITE, END, and STOP commands are used to write data into a particulary slave. The write data count ranges from 1 to 256 bytes. The master specifies the data count in a parameter. If 00H is specified for the data count, 256-byte data transfer is assumed to be specified.



M: Output by master
S: Output by slave

**Fig. 5.5-18 WRITE, END Command Transfer Format**

After receiving the data count, the slave returns $\overline{\text{ACK}}$ if the receive data store area is larger than the data count. If it is less than the data count, it does not return $\overline{\text{ACK}}$, resulting in an error.
At the termination of all data transfer, the master sends the END command to the slave. It signals that all data has been transferred normally.
The slave also receives an END command before it completes all data reception. In this case, the data which has been received immediately before the END command is received becomes valid. During data sending, the master compares the SIO contents before and after data sending to check that data has been output to the bus normally. If the SO contents before and after sending do not match, the master sends the STOP command and stops data transfer.

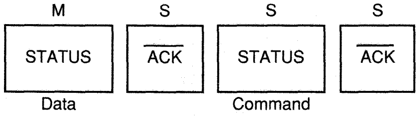


M: Output by master
S: Output by slave

**Fig. 5.5-19 STOP Command Transfer Format**

When receiving the STOP command, the slave invalidates the 1-byte data received immediately before receiving the STOP command.

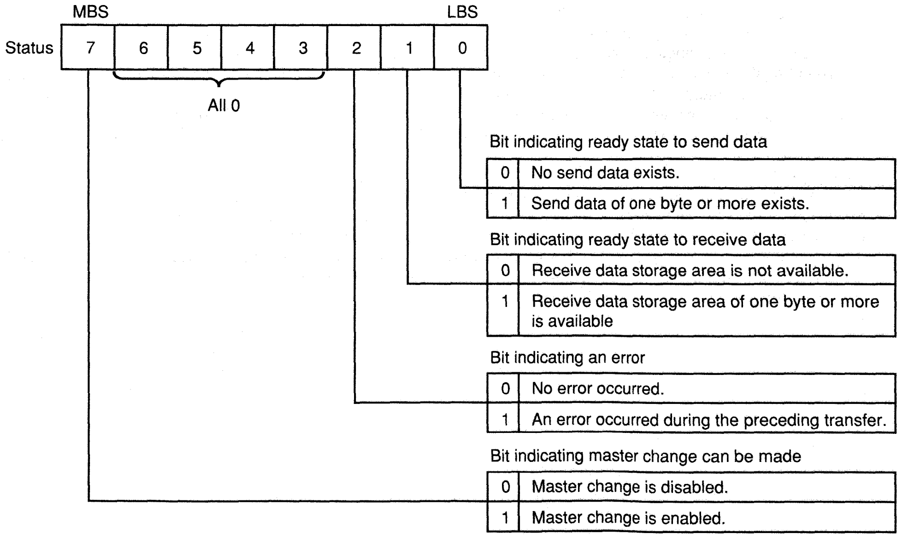
3) STATUS command
The STATUS Command Transfer Format



M: Output by master
S: Output by slave

**Fig. 5.5-20 STATUS Command Status Format**

Fig. 5.5-21 shows the format of the status returned by the slave.



Bit indicating ready state to send data

| | |
|---|---|
| 0 | No send data exists. |
| 1 | Send data of one byte or more exists. |

Bit indicating ready state to receive data

| | |
|---|---|
| 0 | Receive data storage area is not available. |
| 1 | Receive data storage area of one byte or more is available |

Bit indicating an error

| | |
|---|---|
| 0 | No error occurred. |
| 1 | An error occurred during the preceding transfer. |

Bit indicating master change can be made

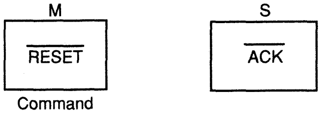
| | |
|---|---|
| 0 | Master change is disabled. |
| 1 | Master change is enabled. |

**Fig. 5.5-21 STATUS Command Status Format**

To send status data, the slave compares the contents before and after sending status data. If they do not match, the slave does not return ACK, resulting in an error.

4) $\overline{\text{RESET}}$ command

The $\overline{\text{RESET}}$ command is used to cause the currently selected slave to be selected. When the $\overline{\text{RESET}}$ command is issued, all slaves can be deselected.

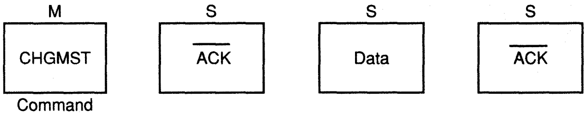


M: Output by  master
S: Output ba slave

**Fig. 5.5-22 RESET Command Transfer Format**

5) CHGMST command

The CHGMST command transfers the master authorization to the currently selected slave.



M: Output by master
S: Output by slave

**Fig. 5.5-23 CHGMST Command Transfer Format**

When receiving the CHGMST command, the slave decides whether or not it can receive the master authorization, and returns either of the following data to the master:

— 0FFH: Master change is enabled.
— 00H  : Master change is disabled.

When transferring data, the slave compares the SIO contents before and after data transfer. If they do not match, the slave does not return $\overline{\text{ACK}}$, resulting in an error.

If no error occurs, the master serves as a slave after 0FFH data sending is complete. If no error occurs, the slave serves as the master after 0FFH data sending is completed.

Error occurrence

When a communication error occurs, the master and slave operate as explained below:

The slave informs the master of error occurrence by returning no $\overline{\text{ACK}}$. When an error occurs, the status bit (bit 2) indicating error occurence is set to 1 and all command processing being performed is cancelled.

After completing the sending or receiving of one byte, the master checks whether or not $\overline{\text{ACK}}$ is returned from slave. If $\overline{\text{ACK}}$ is not returned from slave within a given period after sending or receiving is completed, the master decides that an error has occurred and outputs a dummy $\overline{\text{ACK}}$ signal.

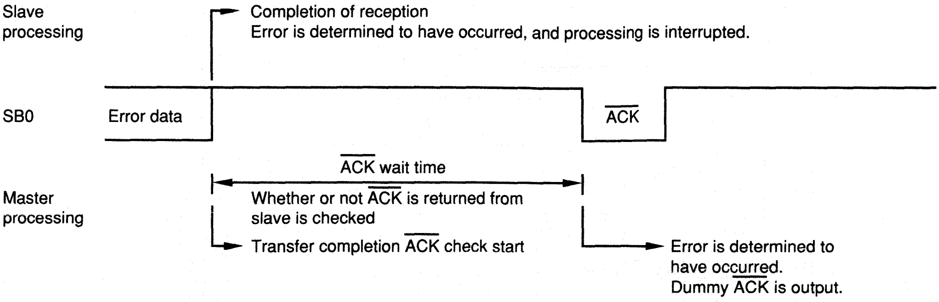

Slave processing
Completion of reception
Error is determined to have occurred, and processing is interrupted.
SB0
Error data
$\overline{\text{ACK}}$
$\overline{\text{ACK}}$ wait time
Master processing
Whether or not $\overline{\text{ACK}}$ is returned from slave is checked
Transfer completion $\overline{\text{ACK}}$ check start
Error is determined to have occurred.
Dummy $\overline{\text{ACK}}$ is output.

**Fig. 5.5-24 Master and Slave Operation when Error Occured**

The following errors are possible:

— Errors that may occur in the slave

1) Command transfer format is erroneous.
2) Undefined command is received.
3) The number of data bytes to be transferred (data count) is insufficient during READ command execution,
4) Data storage area is insufficient during WRITE command execution.
5) When READ, STATUS, or CHGMST command data is sent, data changes.

When any error in 1) or 5) occurs, no $\overline{\text{ACK}}$ is returned.

— Errors that may occur in the master

When WRITE command data is sent, if data changes, STOP command is sent to the slave.

## 6. INTERRUPT FUNCTION

The μPD75402A contains three vectored interrupt sources and one testable inputs for versatile application.

The μPD75402A interrupt control circuit has the following features to enable very high-speed interrupt service:

(a) Whether or not interrupts can be acknowledged can be controlled by using enable flag (IEXXX) and interrupt master enable flag (IME).

(b) The interrupt service start address during interrupt service can be set as desired by using a vector table. Starting the actual interrupt service program is fast.

(c) Interrupt request flag (IRQXXX) can be tested and cleared. Interrupt occurrence can be checked by using software.

(d) The standby mode (HALT) can be released by making an interrupt request (except for INT0).

### 6.1 Interrupt Control Circuit Configuration

Fig. 6.1-1 shows configuration of the interrupt control circuit. The hardware devices are mapped in data memory.

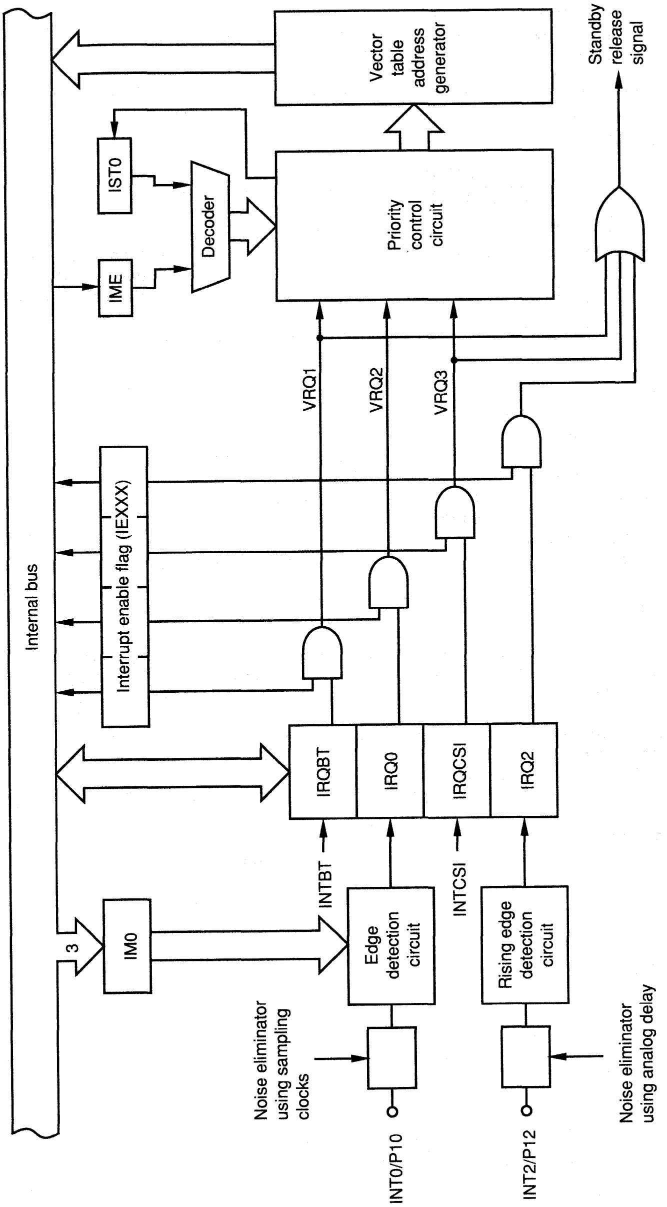

Internal bus
Vector table address generator
Standby release signal
IST0
IME
Decoder
Priority control circuit
Interrupt enable flag (IEXXX)
VRQ1
VRQ2
VRQ3
IRQBT
IRQ0
IRQCSI
IRQ2
INTBT
INTCSI
3
IM0
Edge detection circuit
Rising edge detection circuit
Noise eliminator using sampling clocks
Noise eliminator using analog delay
INT0/P10
INT2/P12

**Fig. 6.1-1 Interrupt Control Circuit Block Diagram**

### 6.2 Interrupt Sources and Vector Table

Table 6.2-1 lists the μPD75402A interrupt sources. Figure 6.2-1 shows the μPD75402A interrupt vector table.

**Table 6.2-1 Interrupt Request Sources**

| Interrupt request occurrence source | Internal/ external | Interrupt priority (Note) | Vectored interrupt request signal (vector table address) |
|---|---|---|---|
| INTBT<br>(reference time interval signal from basic interval timer) | Internal | 1 | VRQ1 (0002H) |
| INT0<br>(INT0 pin input specification edge detection) | External | 2 | VRQ2 (0004H) |
| INTCSI<br>(serial data transfer termination signal) | Internal | 3 | VRQ3 (0008H) |
| INT2<br>(INT2 pin input rising edge detection) | External | Testable input signal (IRQ2 flag is set) | |

Note: The interrupt priority is the priority assigned to the interrupt request source when a number of interrupt requests occur.

| Address | | | | | | |
|---|---|---|---|---|---|---|
| 0002H | 0 | 0 | 0 | 0 | 0 | INTBT start address (high-order three bits) |
| | INTBT start address (Low-order eight bits) | | | | | |
| 0004H | 0 | 0 | 0 | 0 | 0 | INT0 start address (high-order three bits) |
| | INT0 start address (Low-order eight bits) | | | | | |
| ≈ | | | | | | |
| 0008H | 0 | 0 | 0 | 0 | 0 | INTCSI start address (high-order three bits) |
| | INTCSI start address (Low-order eight bits) | | | | | |

**Fig. 6.2-1 Interrupt Vector Table**

The interrupt priority in Table 6.2-1 indicates the interrupt execution priority when a number of interrupt requests occur at the same time or are held.

The interrupt service start addresses are written into the vector table. The vector address table is set by using an assembler pseudo instruction (VENTn).

Example: Set INTBT vector table.

VENT1 GOTOBT

↑ Vector table at address 0002

↑ Symbol indicating the interrupt service routine start address

Caution: The vecto table address specified in VENTn (n = 1; 2; 4) becomes address 2n.

Example: Set INTBT and INT0 vector table.

```
VENT1     GOTOBT
VENT2     GOTO0
```

### 6.3 Interrupt Control Circuit Hardware

(1) Interrupt request flags and interrupt enable flags

The interrupt request flag (IRQXXX) is set to 1 when an interrupt request occurs; it is automatically cleared when a given interrupt service is executed.

The interrupt enable flag (IEXXX) is provided for each interrupt request flag separately. When it contains 1, its corresponding interrupt is enabled; when 0, disabled.

When an interrupt request flag is set to 1 and its corresponding interrupt enable flag is set to 1, a vectored interrupt request (VRQn) occurs. This signal except VRQ2 is also used to release the standby mode (HALT mode).

The interrupt request and enable flags are handled by using bit handling and 4-bit memory handling instructions. The interrupt enable flag is handled by using EI IEXXX and DI IEXXX instructions. Normally, the interrupt request flag is tested by using the SKTCLR instruction.

```
Example:    EI      IE0      ; Enable INT0
            DI      IEBT     ; Disable INTBT
            SKTCR   IRQCSI   ; Skip and clear if IRQCSI is set to 1
```

If the interrupt request flag is set to 1 by using an instruction, a vectored interrupt is executed as if it occurred although it does not occur.

When the $\overline{\text{RESET}}$ signal is generated, the interrupt request and enable flags are cleared and all interrupts are disabled.

**Table 6.3-1 Interrupt Request Flags and Interrupt Enable Flags**

| Interrupt request flag | Interrupt request flag set signal | Interrupt enable flag |
|---|---|---|
| IRQBT | Reference time interval signal from basic interval timer | IEBT |
| IRQ0 | INT0/P10 pin input signal edge detection. The detection edge is selected by using the INT0 mode register (IM0) | IE0 |
| IRQCSI | Serial interface serial data transfer operation termination signal | IECSI |
| IRQ2 | INT2/P12 pin input signal rising edge detection | IE2 |

(2) External interrupt input pin hardware

Fig. 6.3-1 shows the INT0 and INT2 configuration.

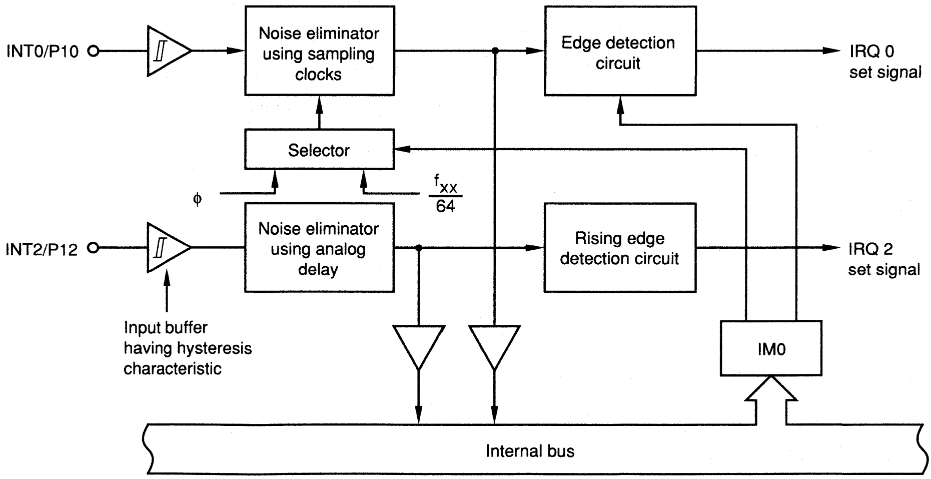

INT0/P10
Noise eliminator using sampling clocks
Edge detection circuit
IRQ 0 set signal
Selector
φ
$\frac{f_{XX}}{64}$
INT2/P12
Noise eliminator using analog delay
Rising edge detection circuit
IRQ 2 set signal
Input buffer having hysteresis characteristic
IM0
Internal bus

**Fig. 6.3-1 INT0 and INT2 Configuration**

The INT0 pin is external interrupt input where noise can be eliminated by using sampling clocks and the detection edge can be selected. The INT0 noise eliminator detects level change by using two sampling clocks. Thus, a pulse narrower than the sampling clock width is removed as a noise and a pulse wider than twice the sampling clock width is securely acknowledge as an interrupt signal. (See Fig. 6.3-2.) Either of the clocks can be selected for the sampling clock.

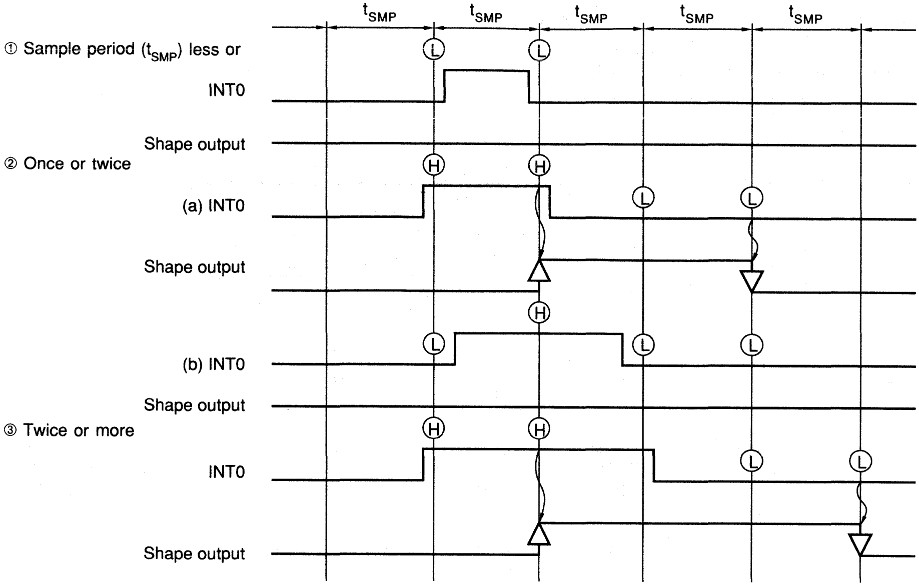


**Fig. 6.3-2 INT0 Noise Eliminator Input/Output Timing**

The INT0 input detection edge and sampling clock are selected by using the edge detection mode register (IM0).
When the INT0 pin is used for the port pin function, data is also input through the noise eliminator. Input data with sufficient width so as not to remove the data as a noise.
The INT2 pin is external testable input where testable flag is set when the rising edge is detected. It does not have a noise eliminator using sampling clocks, but contains the function of eliminating a narrow pulse by using analog delay. As with INT0, input suffeciently wide signals. (See Fig. 6.3-3.)

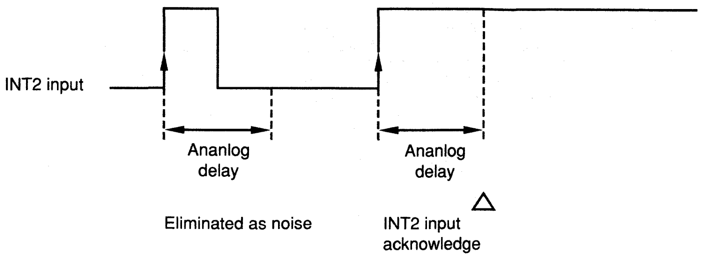
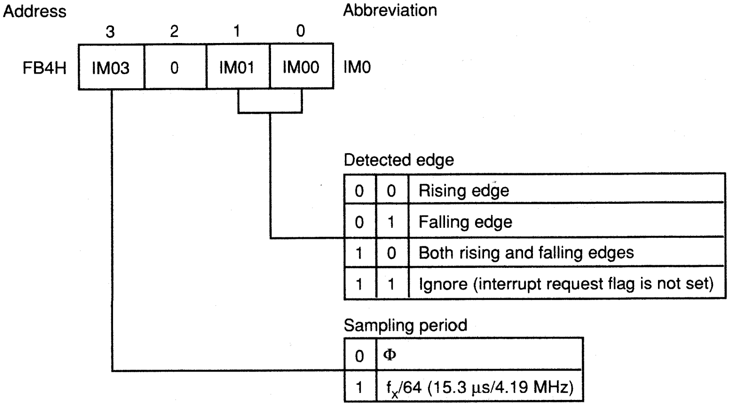


**Fig. 6.3-3 INT2 Input Noise Elimination**

Fig. 6.3-4 shows the format of the edge detection mode register (IM0) to select the detection edge. The IM0 is set by using a 4-bit memory handling instruction.
When the $\overline{\text{RESET}}$ signal is generated, all the IM0 bits are cleared and the rising edge is selected for INT0 and INT2 input detection edges.



Detected edge

| | | |
|---|---|---|
| 0 | 0 | Rising edge |
| 0 | 1 | Falling edge |
| 1 | 0 | Both rising and falling edges |
| 1 | 1 | Ignore (interrupt request flag is not set) |

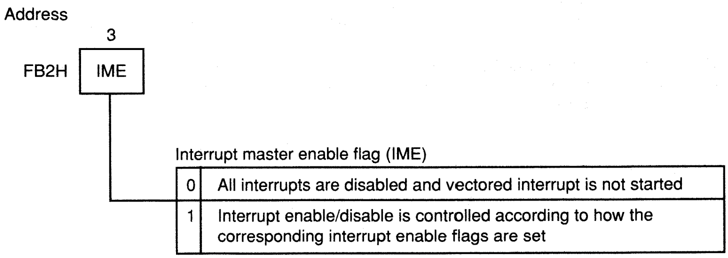
Sampling period

| | |
|---|---|
| 0 | Φ |
| 1 | $f_X$/64 (15.3 μs/4.19 MHz) |

Caution: If the edge detection mode register is changed, the interrupt request flag may be set. Disable interrupts beforehand, and change the mode register. Clear the interrupt request flag by using the CLR1 instruction before enabling interrupts. If $f_X$/64 is selected for the sampling clock in an IM0 change, clear the interrupt request flag within 16 machine cycles after the mode register is changed.

**Fig. 6.3-4 Edge Detection Mode Register Format**

(3) Interrupt master enable flag (IME)
Interrupt master enable flag enables or disables acknowledgement of all interrupts.
IME is set to 1 or to 0 by using the EI and DI instructions.
When the $\overline{\text{RESET}}$ signal is generated, IME is cleared, disabling acknowledgement of all interrupts.



Interrupt master enable flag (IME)

| | |
|---|---|
| 0 | All interrupts are disabled and vectored interrupt is not started |
| 1 | Interrupt enable/disable is controlled according to how the corresponding interrupt enable flags are set |

**Fig. 6.3-5 IME Format**

(4) Interrupt status flag

The interrupt status flag (IST0) indicates the current status of processing being performed by CPU; it is contained in the PSW. The interrupt control circuit controls multi-interrupts (as listed in Table 6.3-2) according to the contents of interrupt status flag. IST0 can not be changed by using a 4-bit or a one-bit manipulation instruction. IST0 is always set to 1 during interrupt serving. Therefore, it is not possible to write 0 to IST0 in the interrupt service routine which would result in multiple interrupt.

When an interrupt is acknowledged, IST0 is saved in stack memory together with other PSW bits, then it is automatically set to 1. When the RETI instruction is executed, the former IST0 value (0) is restored.

When the $\overline{\text{RESET}}$ signal is generated, the flag is cleared.

**Table 6.3-2 IST0 and Interrupt Service State**

| IST0 | Status of processing | CPU processing contents | Interrupt requests that can be acknowledged | After interrupt is acknowledged |
|---|---|---|---|---|
| | | | | IST0 |
| 0 | Status 0 | During normal program processing | All interrupts can be acknowledged. | 1 |
| 1 | Status 1 | During interrupt service | None of the interrupts can be acknowledged. | – |

### 6.4 Interrupt Sequence

When an interrupt occurs, it is processed as shown in Fig. 6.4-1.

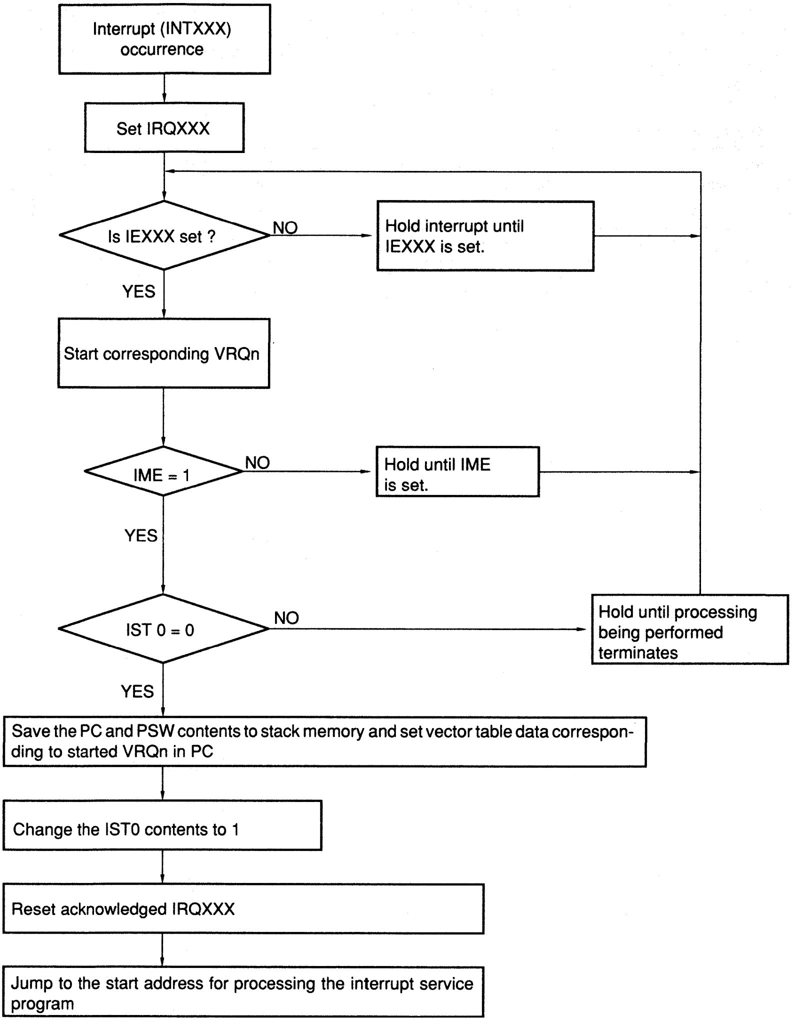

Interrupt (INTXXX) occurrence
Set IRQXXX
Is IEXXX set ?
NO
Hold interrupt until IEXXX is set.
YES
Start corresponding VRQn
IME = 1
NO
Hold until IME is set.
YES
IST 0 = 0
NO
Hold until processing being performed terminates
YES
Save the PC and PSW contents to stack memory and set vector table data correspon-ding to started VRQn in PC
Change the IST0 contents to 1
Reset acknowledged IRQXXX
Jump to the start address for processing the interrupt service program

**Fig. 6.4-1 Interrupt Service Flow**

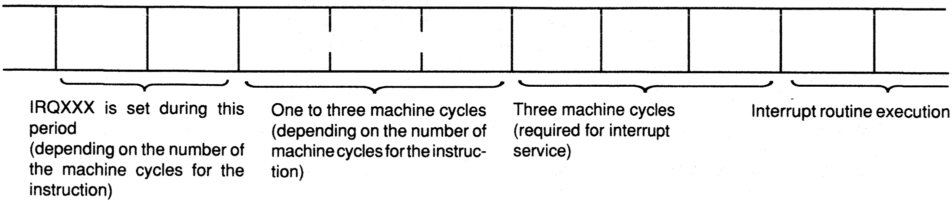
### 6.5 Machine Cycles Until Interrupt Service Starts

(1) When an interrupt request occurs in the interrupt enable state

When an interrupt request for which its corresponding interrupt enable flag is set to 1 occurs (the interrupt request flag is set to 1) in the EI state (IME = 1), the number of machine cycles required until execution of a given interrupt service routine program starts is as follows:

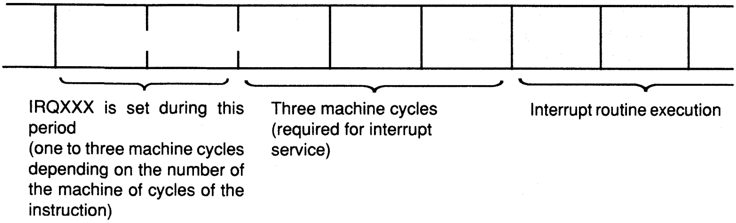
(a) When interrupt request flag (IRQXXX) is set during execution of an address FBxH (interrupt hardware) handling instruction

Address FBxH handling instruction


IRQXXX is set during this period (depending on the number of the machine cycles for the instruction)
One to three machine cycles (depending on the number of machine cycles for the instruction)
Three machine cycles (required for interrupt service)
Interrupt routine execution

As shown above, interrupt routine service starts in a maximum of six machine cycles after the data memory address FBxH handling instructions are given, interrupt routine service starts in a maximum of six machine cycles after the last handling instruction terminates.

(b) When interrupt request flag is set during execution of an instruction other than in (a)


IRQXXX is set during this period (one to three machine cycles depending on the number of the machine of cycles of the instruction)
Three machine cycles (required for interrupt service)
Interrupt routine execution

In this case, a maximum of six machine cycles are required.

(2) When interrupt is enabled after an interrupt request occurs

When an interrupt request occurs (interrupt request flag is set to 1) in the interrupt disable state, the number of machine cycles required until interrupt is enabled is as follows:

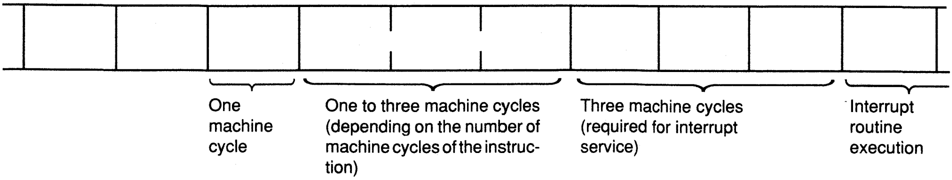
(The interrupt disable state is set when the interrupt master enable flag (IME) is cleared or the interrupt enable flag (IEXXX) corresponding to the generated interrupt request is cleared.)

(a) When interrupt enable flag is set to 1 after IME is set to 1 (EI state) by EI instruction execution

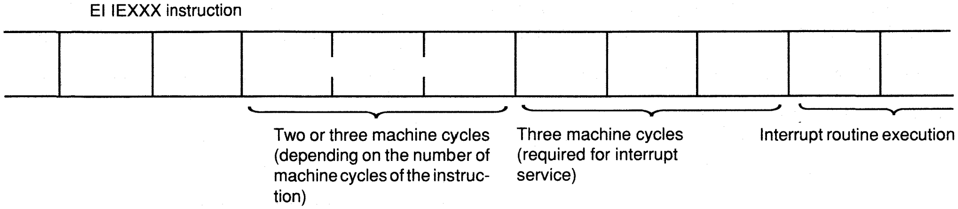
In this case, the number of machine cycles varies depending on the number of the machine cycles of the instruction following the IEXXX set instruction (EI IEXXX).

- When a 1-machine-cycle instruction follows the IEXXX set instruction, interrupt routine service is started after one more instruction is executed.

EI IEXXX instruction

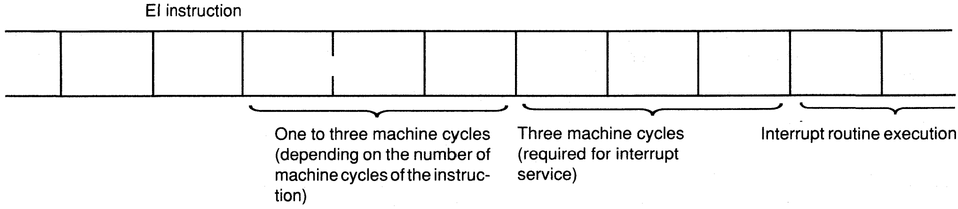

One machine cycle
One to three machine cycles (depending on the number of machine cycles of the instruction)
Three machine cycles (required for interrupt service)
Interrupt routine execution

- When a 2- or 3-machine-cycle instruction follows the IEXXX set instruction, interrupt routine service is started after that instruction is executed.


EI IEXXX instruction
Two or three machine cycles (depending on the number of machine cycles of the instruction)
Three machine cycles (required for interrupt service)
Interrupt routine execution

As shown above, interrupt routine service is started in a maximum of seven machine cycles if the IEXXX set instruction is folllowed by a 1-machine-cycle instruction; otherwise, interrupt routine service is started in a maximum of six machine cycles.

(b) When IME is set to 1 (EI state) by EI instruction execution after interrupt enable flag is set to 1.

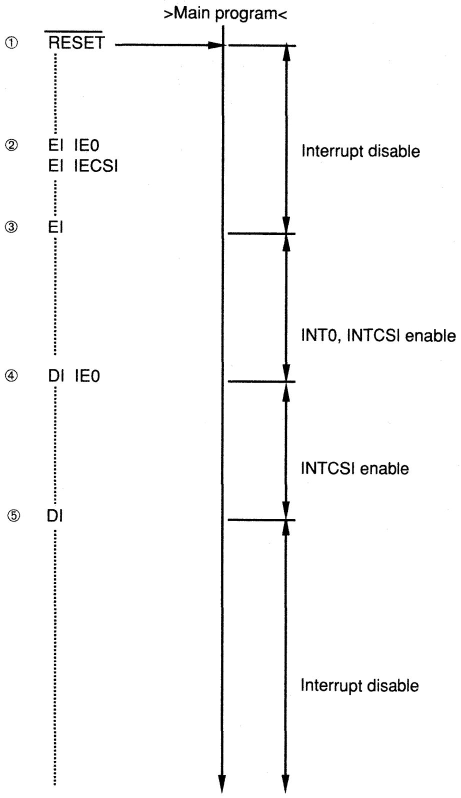

EI instruction
One to three machine cycles (depending on the number of machine cycles of the instruction)
Three machine cycles (required for interrupt service)
Interrupt routine execution

In this case, a maximum of six machine cycles are required.

### 6.6 Interrupt Application

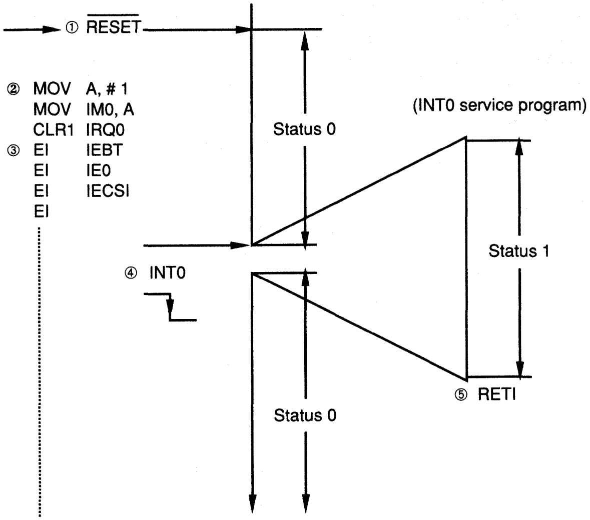
To use the interrupt function, first
① Set the interrupt enable flags of the interrupts to be used to 1 by using EI IEXXX instructions;
② To use INT0, select the active edge by setting IM0;
③ Set the interrupt master enable flag (IME) to 1 by using EI instruction in the main program.
To return from a given interrupt service program, use the RETI instruction.

(1) Interrupt enable and disable



① When the $\overline{\text{RESET}}$ signal is generated, all interrupts are disabled.
② Interrupt enable flags are set to 1 by executing the EI IEXXX instructions. In this step, all interrupts remain disabled.
③ The interrupt master enable flag is set to 1 by executing the EI instruction. In this step, INT0 and INTCSI are enabled.
④ Interrupt enable flag is cleared by executing the DI IEXXX instruction. INT0 is disabled.
⑤ All interrupts are disabled by executing the DI instruction.

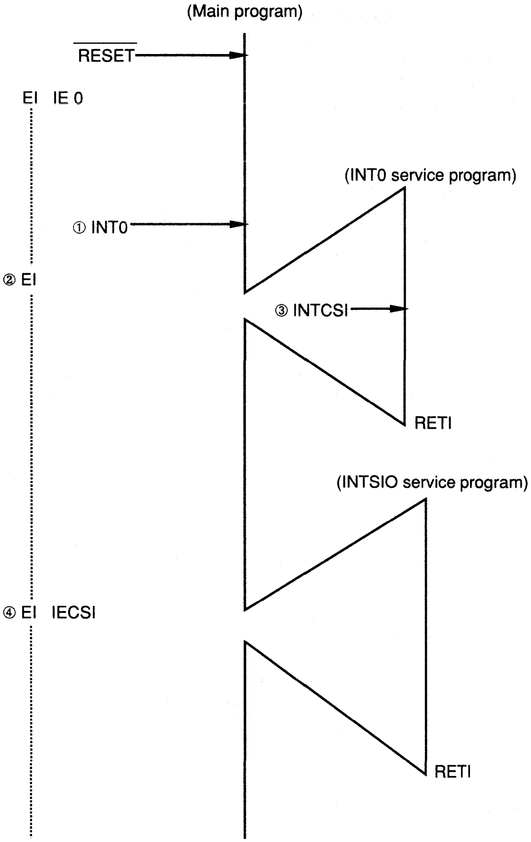
(2) Usage example of INTBT, INT0 (falling edge active), and INTCSI. Multi-interrupt is not done.



```
② MOV   A, # 1
  MOV   IM0, A
  CLR1  IRQ0
③ EI    IEBT
  EI    IE0
  EI    IECSI
  EI
```

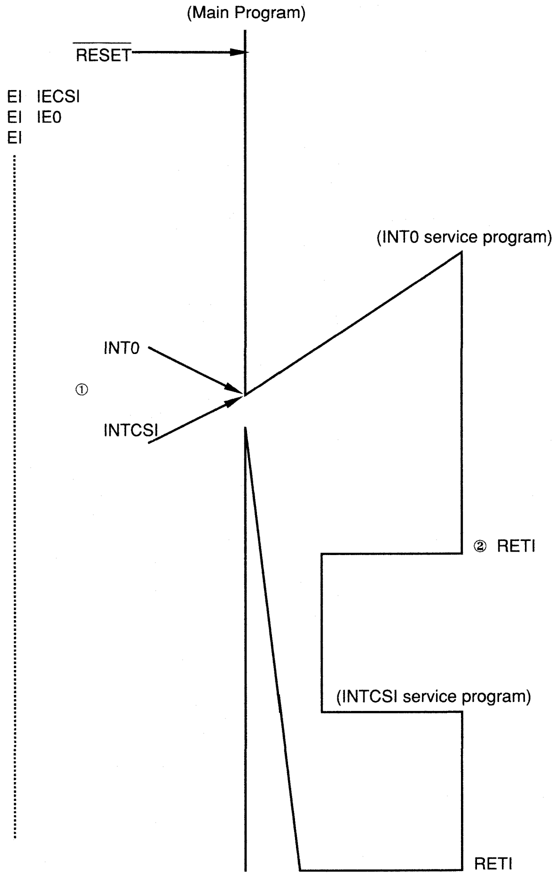
① All interrupts are disabled by the $\overline{\text{RESET}}$ signal and status 0 is set.
② Falling edge active is selected for INT0.
③ Interrupts are enabled by the EI and EI IEXXX instructions.
④ On the INT0 falling edge, the INT0 interrupt service program is started. The status is changed to status 1 and all interrupts are disabled.
⑤ A return is made from the interrupt service program by the RETI instruction. The status is restored to status 0 and interrupts are enabled.

(3) Pending interrupt execution – interrupt input during interrupt disable –



① If INT0 is set during interrupt disable, the request flag is held.
② When interrupts are enabled by the EI instruction, the INT0 service program is started.
③ Simular to ① above.
④ When INTCSI is held enabled, the INTCSI service program is started.

(4) Pending interrupt execution



① If INT0 and INTCSI are occured at the same time during execution of a single instruction, INT0, which is higher in priority than INTCSI, is executed first; INTCSI is held.

② When a return is made from the INT0 service program by the RETI instruction, the held INTCSI service program is started.

## 7. STANDBY FUNCTION

To make the most of the CMOS process features, low consumption current, the μPD75402A enables CPU operation to be stopped in the standby mode for very low CPU consumption current.

The μPD75402A standby mode is the STOP or HALT mode which can be selected according to the application.

The STOP mode stops the system clock oscillator. In this mode, almost all CPU consumption current is leakage current only. Data memory can also be retained at low supply voltage (up to $V_{DD}$ = 2 V). Thus, it is useful to retain the data memory contents at very low consumption current. The μPD75402A STOP mode is not released when an interrupt request is made. It can be released only when $\overline{\text{RESET}}$ is input.

The HALT mode continues system clock oscillator oscillation, but stops CPU clock ϕ supply. Thus, CPU operation stops. Although the HALT mode cannot reduce consumption current as compared with the STOP mode, it is useful to restart processing immediately when an interrupt request is made or for intermittent operation.

In either the STOP or HALT mode, all the contents of the registers, flags, and data memory immediately before the standby mode is selected are retained. The output latch state and output buffer state of the input/output ports are also retained. The input/output port state is previously set so that the entire system consumption current is minimized.

Caution: Although effiecient low consumption current, low voltage operation can be performed by using the standby mode and CPU clock change function, the time as described in 5.2.3 is required from new clock selection using the PCC register to operation start with the new selected clock. Thus, to use the clock change function and the standby mode in combination, select the standby mode in the time required for the change.

### 7.1 Standby Mode Setting and Operation State

**Table 7.1-1 Operating State in Standby Mode**

| | | Stop mode | HALT mode |
|---|---|---|---|
| Mode setting instruction | | STOP instruction | HALT instruction |
| Operation state | Clock oscillator | Clock oscillation stop | Stop of only CPU clock (oscillation is continued) |
| | Basic interval timer | Operation stop | Operation (IRQBT is set to 1 at reference time intervals.) |
| | Serial interface | Operation is enabled only when external $\overline{\text{SCK}}$ input is selected for the system clock. But INTCSI is not enabled | Can operate. |
| | Clock output circuit | Operation stop | Only CPU clock cannot be output |
| | External interrupts | INT2 operation is not enabled.<br>INT0 operation is not enabled. | INT2 operation is enabled<br>INT0 operation is not enabled |
| | CPU | Operation stop | |
| Release signal | | $\overline{\text{RESET}}$ input | $\overline{\text{RESET}}$ input or interrupt request flag from interrupt request flag set to 1 whose corresponding interrupt enable flag is set to 1 |

To select the STOP mode, use the STOP instruction to set PCC bit 3 to 1. To select the HALT mode, use the HALT instruction to set PCC bit 2 to 1.

Be sure to enter a NOP instruction following the STOP or HALT instruction.

When the CPU operation clock is changed by using the low-order two bits of the PCC, time lag may occur from PCC rewriting to CPU clock change. To change the operation clock before the standby mode and the CPU clock after the standby mode is released, select the standby mode in machine cycles required to change the CPU clock after the PCC is rewritten.

The standby mode retains data in data memory and all registers which stop operation during the standby mode, such as general purpose registers, flag, mode registers, and output latches.

Caution 1: When the STOP mode is selected, the X1 pin is connected internally to $V_{SS}$ (GND potential) to suppress clock oscillator leakage. Do not use the STOP mode in a system which uses an external clock.

Caution 2: The μPD75402A and the evaluation chip installed on the evaluation board differ in STOP mode release operation when an interrupt request is made as follows:

- μPD75402A : STOP mode is not released when an interrupt request is made.
- Evaluation chip : STOP mode is released when an interrupt request is made.

To eliminate this difference, disable all interrupt requests before selecting the STOP mode.

Caution 3: Since interrupt request signal is used to release the HALT mode, if there is an interrupt source with both interrupt request and enable flags set to 1, the HALT mode is released immediately when the mode is entered.

### 7.2 Standby Mode Release

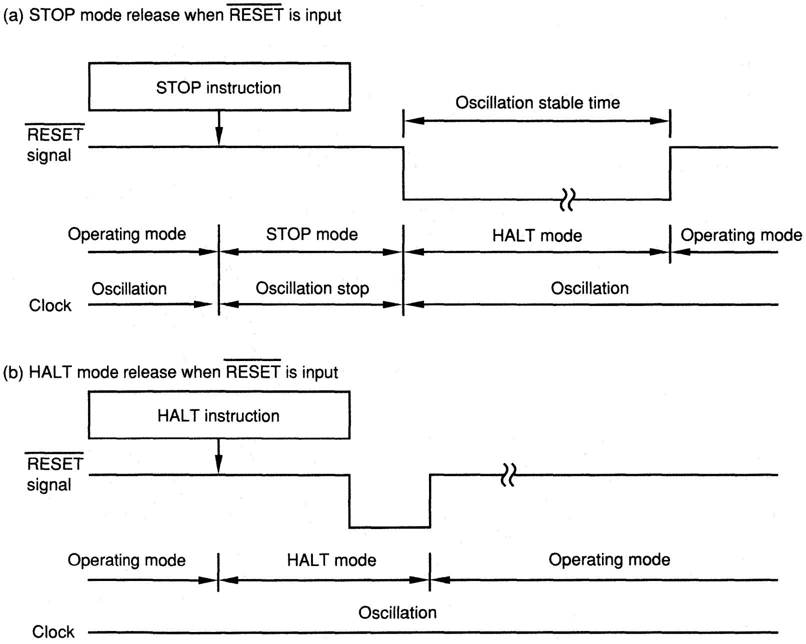
The STOP mode is released only when $\overline{\text{RESET}}$ is input. The HALT mode is released when $\overline{\text{RESET}}$ is input or by the standby release signal generated when the interrupt request flag whose corresponding interrupt enable flag is set to 1.
Fig. 7.2-1 shows standby mode release operation.

Caution: When the standby mode (STOP or HALT) is released when $\overline{\text{RESET}}$ is input, the μPD75402A does not insert wait before starting instruction execution. Instruction execution is started immediately when reset is released.

(1) STOP mode release when $\overline{\text{RESET}}$ is input
When the high-to-low transition of $\overline{\text{RESET}}$ input is made, the oscillator starts oscillation at the same time the reset state is entered.
When the low-to high transition of $\overline{\text{RESET}}$ input is made, instruction execution is started although oscillation operation does not become stable.
Thus, take sufficient low level width of $\overline{\text{RESET}}$ input and reserve the oscillation stable time.
When the reset state is released, a branch is made to the reset start address.
Unlike the normal reset operation, the data memory retains the contents before the STOP mode is selected.

(2) HALT mode release when $\overline{\text{RESET}}$ is input
When the high-to-low transition of $\overline{\text{RESET}}$ input is made, the HALT mode is released and the reset state is entered.
When the low-to-high transition of $\overline{\text{RESET}}$ input is made, a branch is made to the reset start address and instruction execution is started.
Unlike the normal reset operation, the data memory retains the contents before the HALT mode is selected.

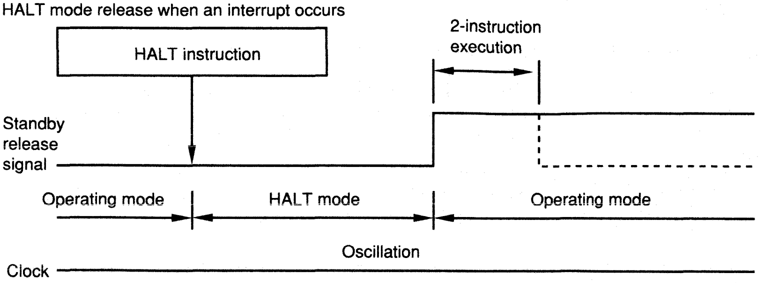
(3) HALT mode release when an interrupt occurs
When the interrupt request flag whose corresponding interrupt enable flag is set to 1, the standby release signal is generated and the HALT mode is released. However, when an INT0 interrupt request is made, the standby release signal is not generated.


(a) STOP mode release when $\overline{\text{RESET}}$ is input

(b) HALT mode release when $\overline{\text{RESET}}$ is input

**Fig. 7.2-1 Standby Mode Release Operation**

(c) HALT mode release when an interrupt occurs



Remarks: Broken line: When the interrupt request releasing the HALT mode is acknowledged (IME = 1).

**Fig. 7.2-1 Standby Mode Release Operation cont'd**

### 7.3 Operation after Standby Mode is Released

(1) If the standby mode (STOP or HALT) is released when $\overline{\text{RESET}}$ is input, the normal reset operation is performed.

(2) If the standby mode (HALT) is released when an interrupt request occurs, whether or not a vectored interrupt is to be made when the CPU restarts instruction execution, is determined by the interrupt master enable flag (IME) contents.

(a) When IME = 0

After the HALT mode released, execution is restarted at the instruction (NOP) following the HALT mode set instruction. The interrupt request flag is retained.

(b) When IME = 1

After the HALT mode is released, two instructions following th HALT mode set instruction are executed, then the vectored interrupt is executed. However, if the HALT mode is released when INT2 (testable input) occurs, no vectored interrupt occurs; processing as in (a) above is performed.

### 7.4 Standby Mode Application

To use the standby mode, perform the following sequence:

① Detect standby mode selection sources such as power off when interrupt input or port input is made.

② Set input/output ports so as to minimize consumption current.

③ Specify standby mode release interrupts. (However, in the HALT mode, interrupt enable flags not to release the HALT mode are cleared.)

④ Specify operation after the standby mode is released. (Set the IME according to whether or not interrupt service is made after the HALT mode is released).

⑤ Select the CPU clock after the standby mode is released. To change the CPU clock, select the standby mode in machine cycles required for the CPU clock change after the PCC is rewritten.)

⑥ Select the standby mode by using the STOP or HALT instruction.

## 8. RESET FUNCTION

When a low level is input to the $\overline{\text{RESET}}$ input pin, the system is reset and the hardware is placed in the state as listed in Table 8.1.

When the low-to-high transition of $\overline{\text{RESET}}$ input is made, the reset state is released. The contents of reset vector table address 001H are loaded into the program counter (PC) bits 7-0 and the contents of the low-order three bits of address 000H are loaded into PC bits 10-8, then a branch is made. Program execution is started at the branch destination address. The enabled reset starts at any desired address.

Initialize the contents of the registers as required in a program.

The $\overline{\text{RESET}}$ input pin is Schmitt trigger input having hysteresis characteristic at the threshold level. It also contains the eliminating function of noise narrower than the analog delay width to prevent noise from causing malfunction. (See Fig. 8-1.)

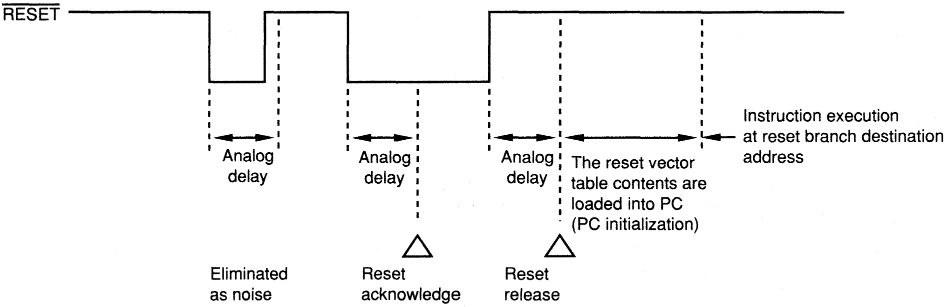

RESET
Analog delay
Analog delay
Analog delay
The reset vector table contents are loaded into PC (PC initialization)
Instruction execution at reset branch destination address
Eliminated as noise
Reset acknowledge
Reset release

**Fig. 8-1 Reset Signal Acknowledge**

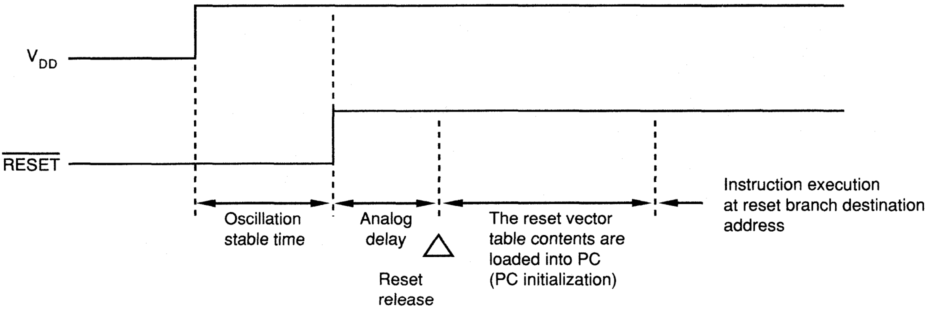

$V_{DD}$
RESET
Oscillation stable time
Analog delay
Reset release
The reset vector table contents are loaded into PC (PC initialization)
Instruction execution at reset branch destination address

**Fig. 8-2 Reset when Power is Turned On**

**Table 8-1 Hardware State after Reset**

| Hardware | | $\overline{\text{RESET}}$ input during standby mode | $\overline{\text{RESET}}$ input during operation |
|---|---|---|---|
| Program counter (PC) | | The low-order three bits of program memory address 000H are loaded into PC10-PC8 and the address 001H contents are loaded into PC7-PC0. | Same as left |
| PSW | Carry flag (CY) | Held | Undefined |
| | Skip flag (SK0-SK2) | 0 | 0 |
| | Interrupt status flag (IST0) | 0 | 0 |
| Stack pointer (SP) | Undefined | Undefined | Undefined |
| Data memory (RAM) | | Held (Note) | Undefined |
| General purpose registers (X, A, H and L) | | Held | Undefined |
| Basic interval timer | Counter (BT) | Undefined | Undefined |
| | Mode register (BTM) | 0 | 0 |
| Serial interface | Shift register (SIO) | Held | Undefined |
| | Operation mode register (CSIM) | 0 | 0 |
| | SBI control register (SBIC) | 0 | 0 |
| | Slave address register (SVA) | Held | Undefined |
| Clock generator, clock output circuit | Processor clock control register (PCC) | 0 | 0 |
| | Clock output mode register (CLOM) | 0 | 0 |
| Interrupt function | Interrupt request flag (IRQXXX) | Cleared | Cleared |
| | Interrupt enable flag (IEXXX) | 0 | 0 |
| | Interrupt master enable flag (IME) | 0 | 0 |
| | INT0 mode registers (IM0) | 0 | 0 |
| Digital ports | Output buffers | Off | Off |
| | Output latches | Cleared | Cleared |
| | Input/output mode registers (PMGA and PMGB) | 0 | 0 |
| | Pull-up resistor specification register (POGA) | 0 | 0 |
| Pin state | P00-P03, P10, P12, P20-P23, P30-P33, P60-P63, | Input | Same as left |
| | P50-P53 | • When internal pull-up resistor is specified: High level<br>• During open drain: High impedance | Same as left |

Note: When $\overline{\text{RESET}}$ is input, the contents of data memory addresses 038H-03DH become undefined.

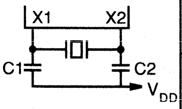
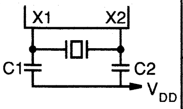
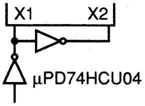
## 9. ELECTRICAL CHARACTERISTICS

**Absolute Maximum Ratings** (Ta = 25 °C)

| Item | Symbol | Conditions | | Ratings | Unit |
|---|---|---|---|---|---|
| Supply voltage | $V_{DD}$ | | | –0.3 to +7.0 | V |
| Input voltage | $V_{I1}$ | Other than port 5 | | –0.3 to $V_{DD}$ +0.3 | V |
| | $V_{I2}$ | Ports 5 | With built-in pull-up resistor | –0.3 to $V_{DD}$ +0.3 | V |
| | | | Open drain | –0.3 to +11 | V |
| Output voltage | $V_O$ | | | –0.3 to $V_{DD}$ +0.3 | V |
| High level output current | $I_{OH}$ | Single pin | | –15 | mA |
| | | All pins | | –30 | |
| Low level output current | $I_{OL}$ Note | One port pin of ports 0, 3, 5 and 6 | Peak value | 30 | mA |
| | | | Effective value | 15 | |
| | | Port 2 single pin | Peak value | 20 | |
| | | | Effective value | 10 | |
| | | Total of ports 0, 3 and 5, except P33 | Peak value | 100 | |
| | | | Effective value | 60 | |
| | | Total of ports 2, 6 and P33 | Peak value | 100 | |
| | | | Effective value | 60 | |
| Operation Temperature | $T_{opt}$ | | | –40 to +85 | °C |
| Storage Temperature | $T_{stg}$ | | | –65 to +150 | °C |

Note: Use the following formula to calculate the effective value. (Effective value) = (Peak value) x $\sqrt{\text{duty}}$

**Characteristics of Main System Clock Oscillator** (Ta = –40 to +85 °C, $V_{DD}$ = 2.7 to 6.0V)

| Oscillator | Recommended constants | Item | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Ceramic resonator |  | Note 1<br>Oscillation frequency ($f_{XX}$) | | 2.0 | | 5.0 | MHz |
| | | Note 2<br>Oscillation stabilization time | After $V_{DD}$ reaches the minimum value in the oscillator voltage range | | | 4 | ms |
| Crystal resonator |  | Note 1<br>Oscillation frequency ($f_{XX}$) | | 2.0 | 4.19 | 5.0 | MHz |
| | | Note 2<br>Oscillation stabilization time | $V_{DD}$ =4.5 to 6.0V | | | 10 | ms |
| | | | | | | 30 | ms |
| External clock |  | Note 1<br>X1 input frequency ($f_X$) | | 2.0 | | 5.0 | MHz |
| | | X1 input high/low level width ($t_{XH}$, $t_{XL}$) | | 100 | | 500 | ns |

Note 1: The oscillation frequency and X1 input frequency are indicated only to express the characteristics of the oscillator. Refer to the AC characteristics for instruction execution time.

Note 2: The oscillation stabilization time is the time required for the oscillator to stabillate after $V_{DD}$ is applied or after the STOP mode is released.

**Recommended Ceramic Resonator**

| Manufacturer | Part name* | Frequency [MHz] | Capacitance [pF] | | Oscillation voltage range [V] | |
|---|---|---|---|---|---|---|
| | | | C1 | C2 | MIN. | MAX. |
| Murata Mfg. Co., Ltd. | CSAxxxMG093 | 2.00-2.44 | 30 | 30 | 2.7 | 6.0 |
| | CSAxxxMGU | 2.45-4.91 | 30 | 30 | 2.7 | 6.0 |
| Kyocera Corp. | KBR-2.0MS | 2.0 | 47 | 47 | 3.0 | 6.0 |
| | KBR-4.19MS | 4.19 | 33 | 33 | 3.0 | 6.0 |
| | KBR-5.0MS | 5.0 | 33 | 33 | 3.3 | 6.0 |

* xxx indicates frequency.

6

**DC Characteristics** (Ta = –40 to +85 °C, $V_{DD}$ = 2.7 to 6.0V)

| Item | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| High level input voltage | $V_{IH1}$ | Ports 2 and 3 and 6 | | $0.7V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH2}$ | Ports 0, 1 and $\overline{\text{RESET}}$ pin | | $0.8V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH3}$ | Port 5 | With built-in pullup resistor | $0.7V_{DD}$ | | $V_{DD}$ | V |
| | | | Open drain | $0.7V_{DD}$ | | 10 | V |
| | $V_{IH4}$ | X1, X2 | | $V_{DD}$–0.5 | | $V_{DD}$ | V |
| Low level input voltage | $V_{IL1}$ | Ports 2, 3, 5, 6 | | 0 | | $0.3V_{DD}$ | V |
| | $V_{IL2}$ | Ports 0, 1 and $\overline{\text{RESET}}$ pin | | 0 | | $0.2V_{DD}$ | V |
| | $V_{IL3}$ | X1, X2 | | 0 | | 0.4 | V |
| High level output voltage | $V_{OH1}$ | Ports 0, 2, 3, 5, 6 | $V_{DD}$ = 4.5 to 6.0V<br>$I_{OH}$ = –1mA | $V_{DD}$–1.0 | | | V |
| | | | $I_{OH}$ = –100µA | $V_{DD}$–0.5 | | | V |
| Built-in pull-up resistor | $R_{L1}$ | Ports 0, 1, 2, 3, 6, (Except P00, P10) $V_{IN}$ = 0V | $V_{DD}$ = 5.0V ± 10% | 15 | 40 | 80 | kΩ |
| | | | $V_{DD}$ = 3.0V ± 10% | 30 | | 300 | kΩ |
| | $R_{L2}$ | Port 5 $V_{OUT}$= $V_{DD}$ –2.0V | $V_{DD}$ = 5.0V ± 10% | 15 | 40 | 70 | kΩ |
| | | | $V_{DD}$ = 3.0V ± 10% | 10 | | 60 | kΩ |
| Low level output voltage | $V_{OL1}$ | Ports 3, 5, 6 | $V_{DD}$ = 4.5-6.0V<br>$I_{OL}$ = 15mA | | 0.6 | 2.0 | V |
| | | Ports 0, 2, 3, 5, 6 | $V_{DD}$ = 4.5-6.0V $I_{OL}$ = 1.6mA | | | 0.4 | V |
| | | | $I_{OL}$ = 400µA | | | 0.5 | V |
| | $V_{OL2}$ | SB0 open drain | Pull-up resistance: 1kΩ min.<br>$V_{DD}$ = 4.5-6.0V | | | $0.2V_{DD}$ | V |
| High level input leakage current | $I_{LIH1}$ | $V_{IN}$ = $V_{DD}$ | Other than indicated below | | | 3 | µA |
| | $I_{LIH2}$ | | X1, X2 | | | 20 | µA |
| | $I_{LIH3}$ | $V_{IN}$ = 10V | Port 5 (with open drain) | | | 20 | µA |
| Low level input leakage current | $I_{LIL1}$ | $V_{IN}$ = 0V | Other than indicated below | | | –3 | µA |
| | $I_{LIL2}$ | | X1, X2 | | | –20 | µA |
| High level output leakage current | $I_{LOH1}$ | $V_{OUT}$ = $V_{DD}$ | Other than indicated below | | | 3 | µA |
| | $I_{LOH2}$ | $V_{OUT}$ = 10V | Port 5 (with open drain) | | | 20 | µA |
| Low level output leakage current | $I_{LOL}$ | $V_{OUT}$ = 0V | | | | –3 | µA |

**DC Characteristics (cont'd)** (Ta = –40 to +85 °C, $V_{DD}$ = 2.7 to 6.0V)

| Item | Symbol | Conditions | | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|---|
| Supply current (Note 1) | $I_{DD1}$ | 4.19 MHz crystal oscillation C1=C2=22pF | $V_{DD}$ = 5V ± 10% (Note 2) | | | 2.5 | 8 | mA |
| | | | $V_{DD}$ = 3V ± 10% (Note 3) | | | 0.5 | 1.5 | mA |
| | $I_{DD2}$ | | HALT Mode | $V_{DD}$ = 5V ± 10% | | 500 | 1500 | μA |
| | | | | $V_{DD}$ = 3V ± 10% | | 150 | 450 | μA |
| | $I_{DD3}$ | STOP mode | $V_{DD}$ = 5V ± 10% | | | 0.5 | 20 | μA |
| | | | $V_{DD}$ = 3V ± 10% | | | 0.1 | 10 | μA |
| | | | | Ta = 25°C | | 0.1 | 5 | μA |

Note 1: The current of the built-in pull-up resistor is not included.
Note 2: When operated in the high-speed mode with the processor clock control register (PCC) set to 0011.
Note 3: When operated in the low-speed mode with the PCC set to 0000.

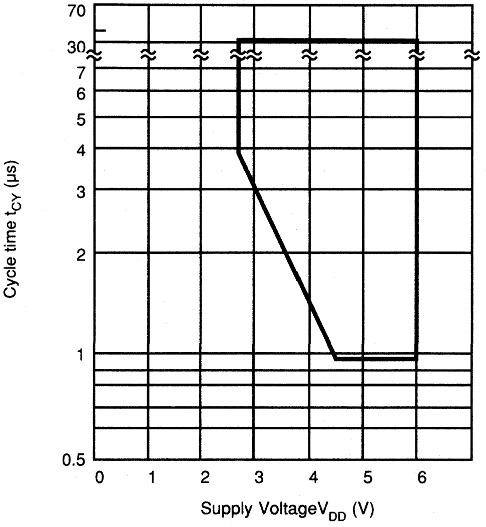
**AC Characteristics** (Ta = –40 to 85 C, VDD = 2.7 to 6.0V)

| Item | Symbol | Condition | | MIN. | TYP. | MAX | Unit |
|---|---|---|---|---|---|---|---|
| (Note 1) Cycle time (minimum instruction execution time) | $t_{CY}$ | Operation with main system clock | $V_{DD}$ = 4.5 –6.0V | 0.95 | | 32 | μs |
| | | | | 3.8 | | 32 | μs |
| Interrupt input high/low level width | $t_{INTH}$, | INT0 | | Note 2 | | | μs |
| | $t_{INTL}$ | INT2 | | 10 | | | μs |
| $\overline{\text{RESET}}$ low level width | $t_{RSL}$ | | | 10 | | | μs |

Note 1: The cycle time (minimum instruction execution time) is determined by the oscillation frequency of the connected resonator, the system clock control register (SCC), and the processor clock control register (PCC). The figure below shows the $V_{DD}$ vs cycle time ($t_{CY}$) when operated with the main system clock.
Note 2: $2t_{CY}$ or $64/f_X$, depending on the setting of the interrupt mode register (IM0).

(with main system clock)

$t_{CY}$ vs. $V_{DD}$



**Capacitance** (Ta = 25 °C, $V_{DD}$ = 0V)

| Item | Symbol | Condition | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| Input capacitance | $C_{IN}$ | f = 1 MHz<br>Unmeasured pin turned to be 0V. | | | 15 | pF |
| Output capacitance | $C_{OUT}$ | | | | 15 | pF |
| Input/output capacitance | $C_{IO}$ | | | | 15 | pF |

**Serial Transfer Operation**
3line serial I/O mode ($\overline{\text{SCK}}$ ... Internal clock output)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY1}$ | $V_{DD}$ = 4.5 –6.0V | 1600 | | | ns |
| | | | 3800 | | | ns |
| $\overline{\text{SCK}}$ high/low level width | $t_{KL1}$<br>$t_{KH1}$ | $V_{DD}$ = 4.5 –6.0V | $\frac{t_{KCY1}}{2}$ -50 | | | ns |
| | | | $\frac{t_{KCY1}}{2}$ -150 | | | ns |
| SI set-up time (against $\overline{\text{SCK}}$ ↑ ) | $t_{SIK1}$ | | 150 | | | ns |
| SI hold time (against $\overline{\text{SCK}}$ ↑ ) | $t_{KSI1}$ | | 400 | | | ns |
| $\overline{\text{SCK}}$ ↓ → SO output delay time | $t_{KSO1}$ | $V_{DD}$ = 4.5 –6.0V | | | 250 | ns |
| | | | | | 1000 | ns |

3line serial I/O mode ($\overline{\text{SCK}}$ ... External clock input)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY2}$ | $V_{DD}$ = 4.5 –6.0V | 800 | | | ns |
| | | | 3200 | | | ns |
| $\overline{\text{SCK}}$ high/low level width | $t_{KL2}$ | $V_{DD}$ = 4.5 –6.0V | 400 | | | ns |
| | $t_{KH2}$ | | 1600 | | | ns |
| SI set-up time (against $\overline{\text{SCK}}$ ↑ ) | $t_{SIK2}$ | | 100 | | | ns |
| SI hold time (against $\overline{\text{SCK}}$ ↑ ) | $t_{KSI2}$ | | 400 | | | ns |
| $\overline{\text{SCK}}$ ↓ → SO output delay time | $t_{KSO2}$ | $V_{DD}$ = 4.5 –6.0V | | | 300 | ns |
| | | | | | 1000 | ns |

Note: The output delay time (for rising edge) of the serial line must be shorter than 600 ns. For example, if SB0 is pulled up with 5K ohms, the total capacitance of the serial bus line must be no greater than 120 pF.

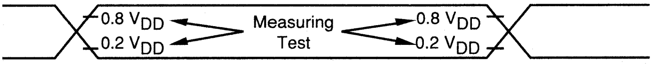
SBI mode ($\overline{\text{SCK}}$ ... Internal clock output (master))

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY3}$ | $V_{DD}$ = 4.5 –6.0V | 1600 | | | ns |
| | | | 3800 | | | ns |
| $\overline{\text{SCK}}$ high/low level width | $t_{KL3}$ | $V_{DD}$ = 4.5 –6.0V | $\frac{t_{KCY1}}{2}$ -50 | | | ns |
| | $t_{KH3}$ | | $\frac{t_{KCY1}}{2}$-150 | | | ns |
| SB0 set-up time (against $\overline{\text{SCK}}$ ↑ ) | $t_{SIK3}$ | | 150 | | | ns |
| SB0 hold time (against $\overline{\text{SCK}}$ ↑ ) | $t_{KSI3}$ | | $\frac{t_{KCY1}}{2}$ | | | ns |
| $\overline{\text{SCK}}$ ↓ → SB0 output delay time | $t_{KSO3}$ | $V_{DD}$ = 4.5 –6.0V | 0 | | 250 | ns |
| | | | 0 | | 1000 | ns |
| $\overline{\text{SCK}}$ ↑ → SB0 ↓ | $t_{KSB}$ | | $t_{KCY}$ | | | ns |
| SB0 ↓ → $\overline{\text{SCK}}$ ↓ | $t_{SBK}$ | | $t_{KCY}$ | | | ns |
| SB0 low level width | $t_{SBL}$ | | $t_{KCY}$ | | | ns |
| SB0 high level width | $t_{SBH}$ | | $t_{KCY}$ | | | ns |

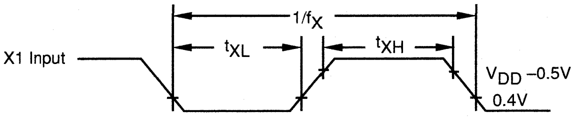
SBI mode ($\overline{SCK}$ ... External clock input (slave))

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{SCK}$ cycle time | $t_{KCY4}$ | $V_{DD}$ = 4.5 –6.0V | 800 | | | ns |
| | | | 3200 | | | ns |
| $\overline{SCK}$ high/low level width | $t_{KL4}$ | $V_{DD}$ = 4.5 –6.0V | 400 | | | ns |
| | $t_{KH4}$ | | 1600 | | | ns |
| SB0 set-up time (against $\overline{SCK}$ ↑ ) | $t_{SIK4}$ | | 100 | | | ns |
| SB0 hold time (against $\overline{SCK}$ ↑ ) | $t_{KSI4}$ | | $\frac{t_{KCY1}}{2}$ | | | ns |
| $\overline{SCK}$↓ → SB0 output delay time | $t_{KSO4}$ | $V_{DD}$ = 4.5 –6.0V | 0 | | 300 | ns |
| | | | 0 | | 1000 | ns |
| $\overline{SCK}$ ↑→ SB0 ↓ | $t_{KSB}$ | | $t_{KCY}$ | | | ns |
| SB0 ↓→ $\overline{SCK}$ ↓ | $t_{SBK}$ | | $t_{KCY}$ | | | ns |
| SB0 low level width | $t_{SBL}$ | | $t_{KCY}$ | | | ns |
| SB0 high level width | $t_{SBH}$ | | $t_{KCY}$ | | | ns |

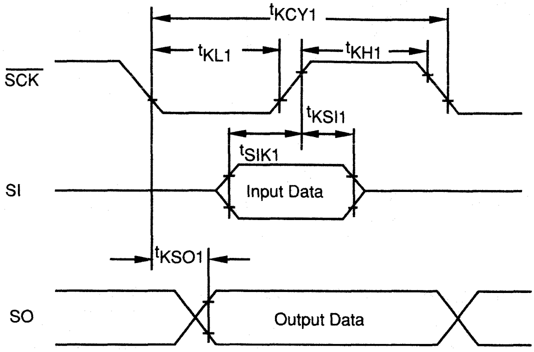
AC Timing Measurement Points (Except X1 input)



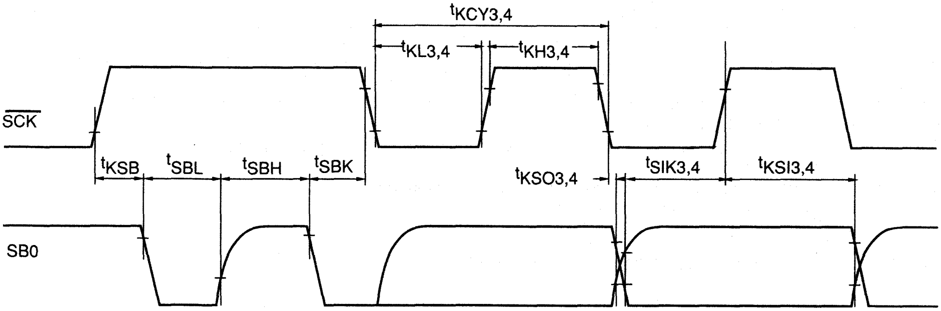
Clock Timing



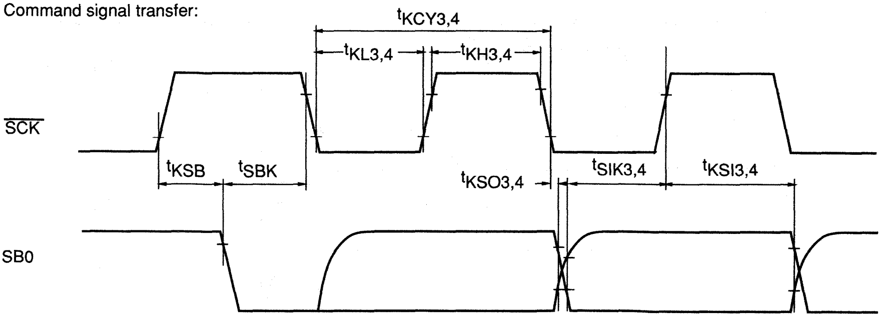
Serial Transfer Timing
3-line serial I/O mode



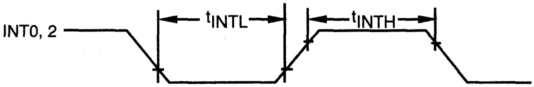
**Serial Transfer Timing**
Bus release signal transfer:



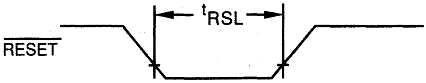
Command signal transfer:



Interrupt input timing



$\overline{\text{RESET}}$ input timing



**Data Memory STOP Mode Low Voltage Data Retention Characteristic** (Ta = –40 to +85 °C)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Data retention voltage | $V_{DDDR}$ | | 2.0 | | 6.0 | V |
| Data retention current (Note1) | $I_{DDDR}$ | $V_{DDR}$ = 2.0V | | 0.1 | 10 | μA |
| Release signal set time | $t_{SREL}$ | | 0 | | | μs |
| Oscillation stabilization time (Note 2) | $t_{OSC}$ | Release by $\overline{\text{RESET}}$ input (Note 3) | | | 4 | ms |
| | | Release by $\overline{\text{RESET}}$ input (Note 4) | | | 10 | ms |

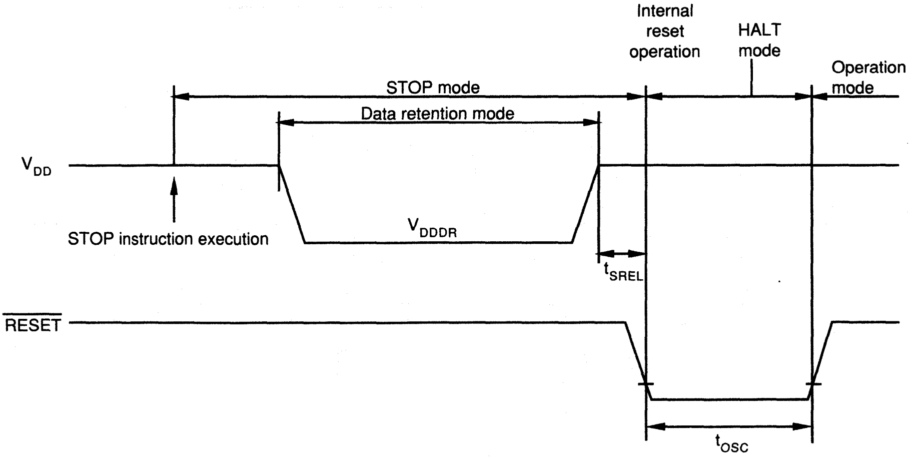
Note 1: Current in the internal pull-up resistors is not included.

Note 2: The oscillation stabilization time is the time required before beginning CPU operation in order to prevent unstable CPU operation when oscillation is initiated.

Note 3: After $V_{DD}$ reaches the oscillation voltage range, if ceramic resonator is used.

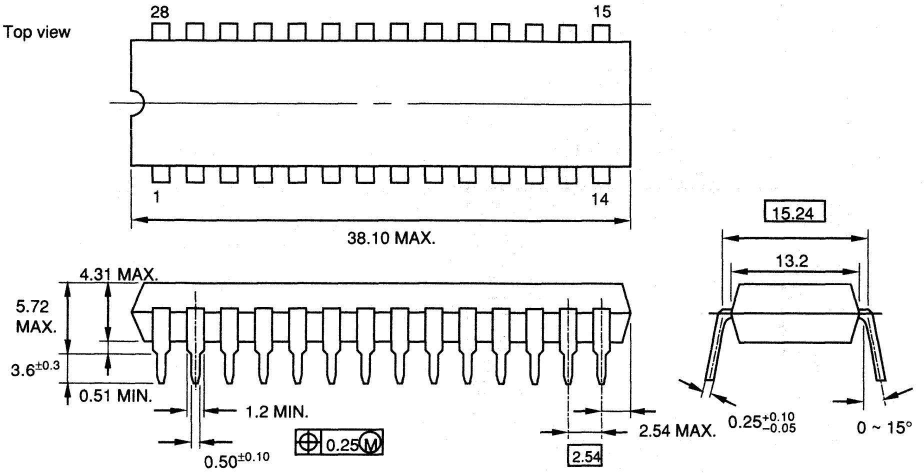
Note 4: After $V_{DD}$ reaches the oscillation voltage range, if crystal oscillator is used.

Data Retention Timing (when STOP mode is released by $\overline{\text{RESET}}$ input)

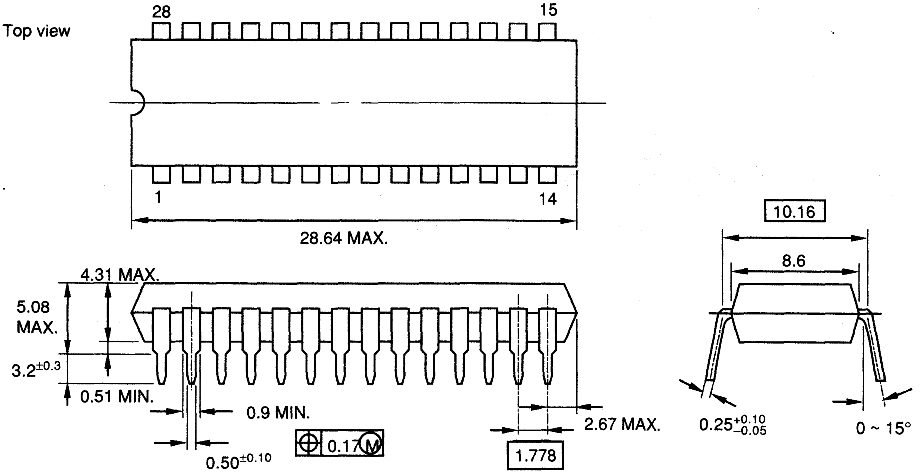


### 9.1 Package Dimension

Package Dimension of 28-pin Plastic DIP (600 mil) (Unit: mm)

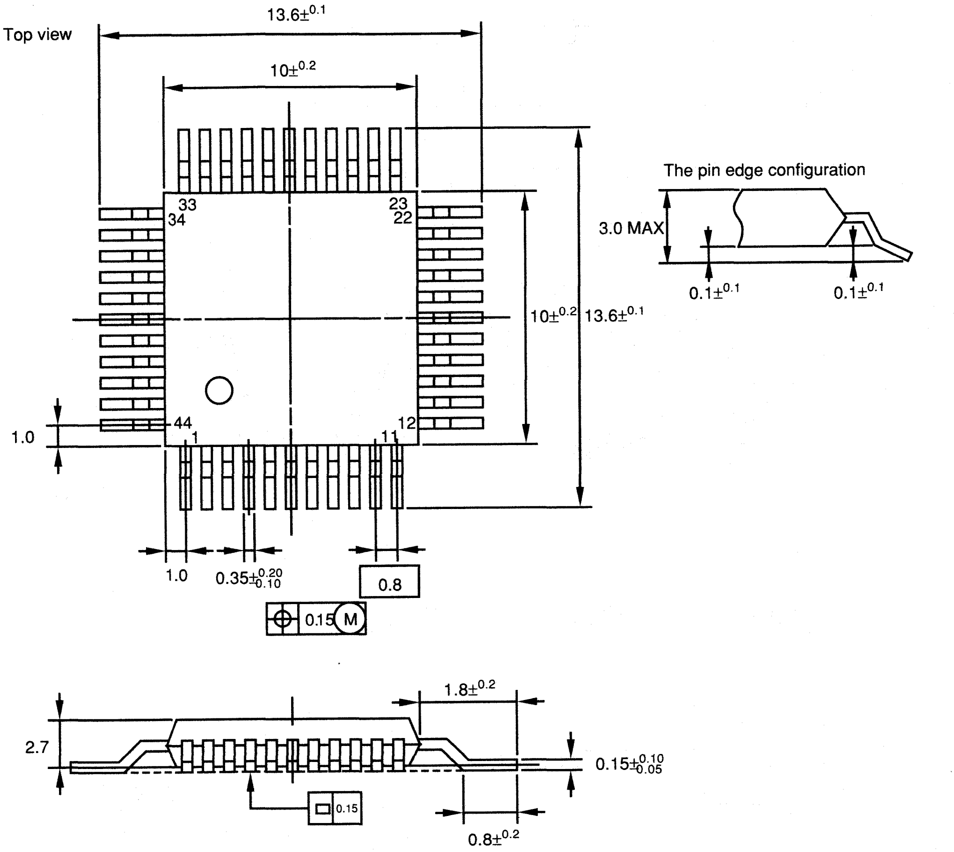


P28C-100-600A1



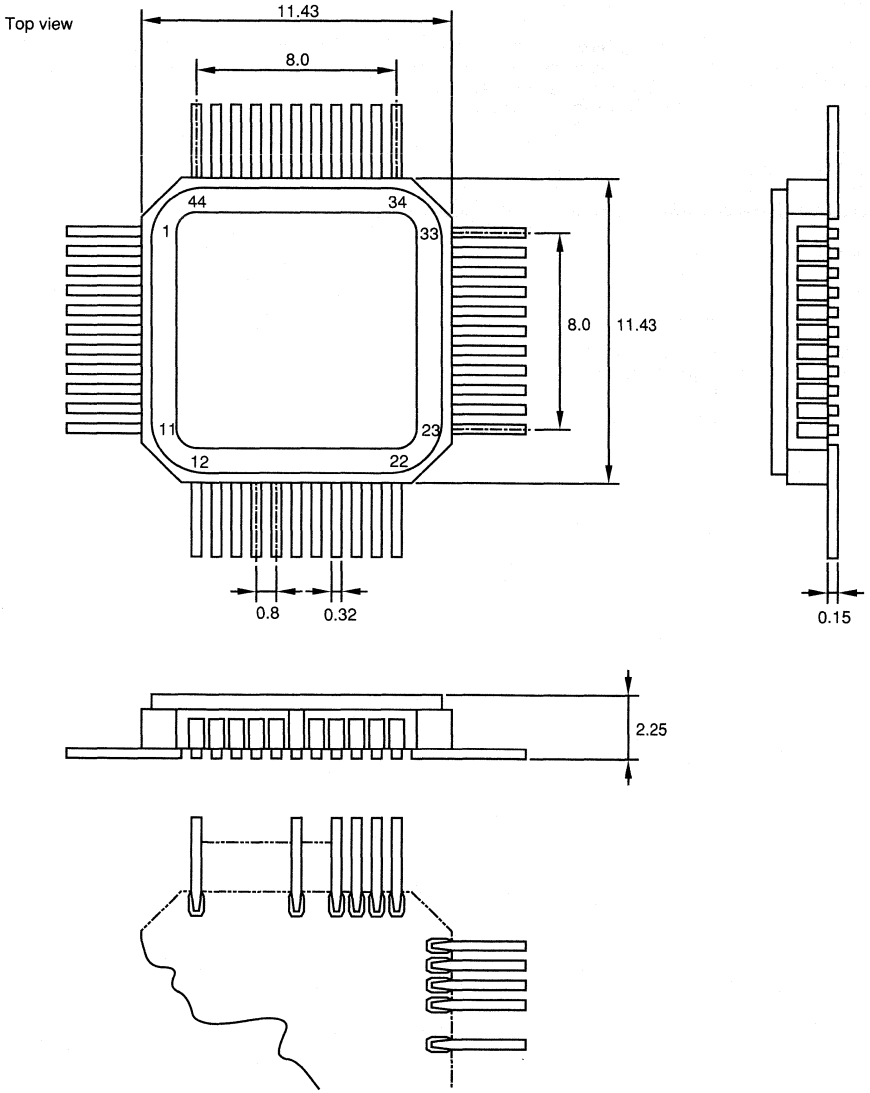
P28C-70-400A

Package Dimension of 44-pin Plastic QFP (Units: mm)



P44GB-80-3B4

Package Dimension of ES 44-pin Ceramic QFP (Reference Drawing) (Unit: mm)



Note 1: A metal cap is connected to pin 17 and is $V_{SS}$ (GND) level.
Note 2: Lead wire length is not stipulated because cutting process of lead ends is out of production control.

## 10. µPD75P402 LOW END 4-BIT MICROCOMPUTER

### 10.1 Overview

The µPD75PD402 is a QTOP™ microcomputer provided by replacing µPD75402A internal mask ROM with one-time PROM. The µPD75P402 into which the user can write programs is suitable for evaluation during system development or small quantity production. µPD75P402 should not be used for final EMI or latch up tests.
To see this document, also refer to the µPD75402A document.

Features:

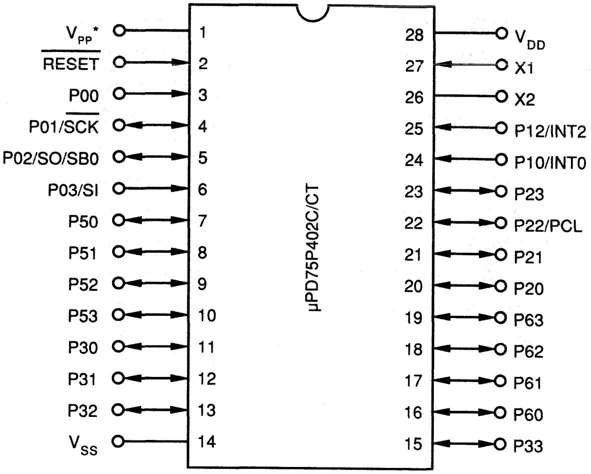
| | |
|---|---|
| • µPD75402A compatible | |
| • Internal one-time PROM: | 1920 x 8 bits |
| • PROM programming characteristics: | µPD27C256A compatible.<br>PROM can be written with general purpose PROM writer. |
| • Single power supply 5 V ± 10% | |

Order Information

| Order name | Package |
|---|---|
| µPD75P402C | 28-pin plastic DIP |
| µPD75P402CT | 28-pin plastic shrink DIP |
| µPD75P402GB-3B4 | 44-pin plastic QFP |

### 10.2 Pin Connections

28-Pin Plastic DIP
(1) Normal operation mode

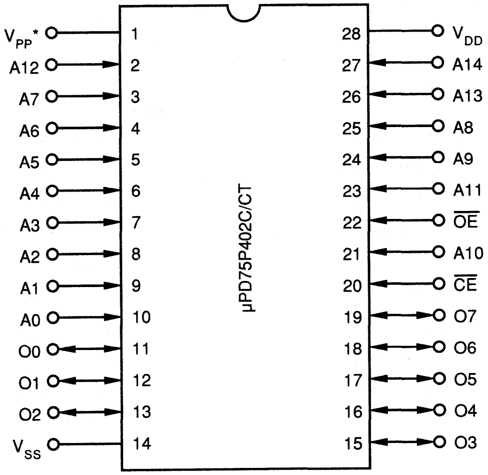


| | | |
|---|---|---|
| P00-P03 | : | Port 0 |
| P10, P12 | : | Port 1 |
| P20-P23 | : | Port 2 |
| P30-P33 | : | Port 3 |
| P50-P53 | : | Port 5 |
| P60-P63 | : | Port 6 |

| | | |
|---|---|---|
| SCK | : | Serial clock input/output |
| SO/SB0 | : | Serial input/output |
| SI | : | Serial input |
| PCL | : | Clock output |
| INT0 | : | |
| INT2 | : | External test input |
| X1, X2 | : | Main system clock |
| RESET | : | Reset input |
| $V_{DD}$ | : | Power supply |
| $V_{SS}$ | : | Ground |
| $V_{PP}$ | : | Setting the GND from external |

( * ): used by only μPD75P402. In operating mode, set to GND from external
To share printed circuit board with μPD75P402 in the μPD75402A, set the $V_{PP}$ pin to the GND potential.

(2) PROM program mode



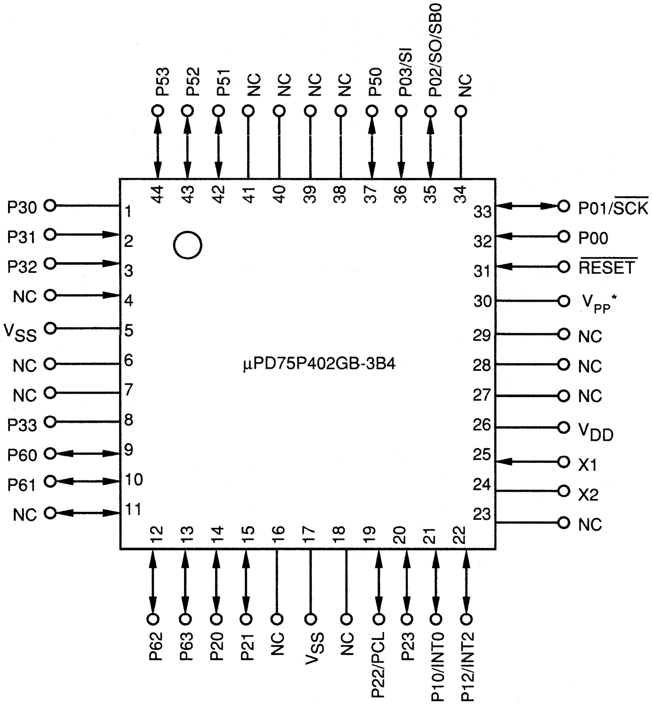
| | | |
|---|---|---|
| A0-A14 | : | Address input |
| O0-O7 | : | Data input/output |
| $\overline{CE}$ | : | Chip enable input |
| $\overline{OE}$ | : | Output enable input |
| $V_{DD}$ | : | Power supply |
| $V_{PP}$ | : | Program power supply |
| $V_{SS}$ | : | Ground |

(*): used by μPD75402 only. In operating mode, set to GND from external. Also to share pcb with μPD75402A (where this pin is NC), set pin to GND potential.

44-pin plastic QFP (bent lead)

(1) Normal operation mode

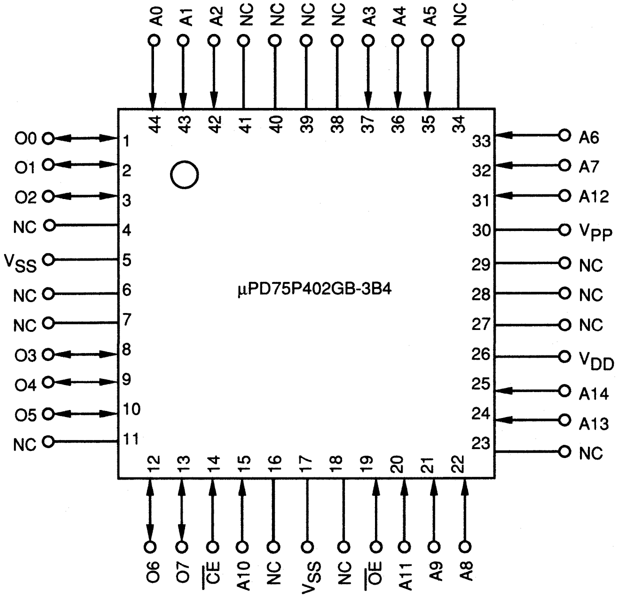
Top view



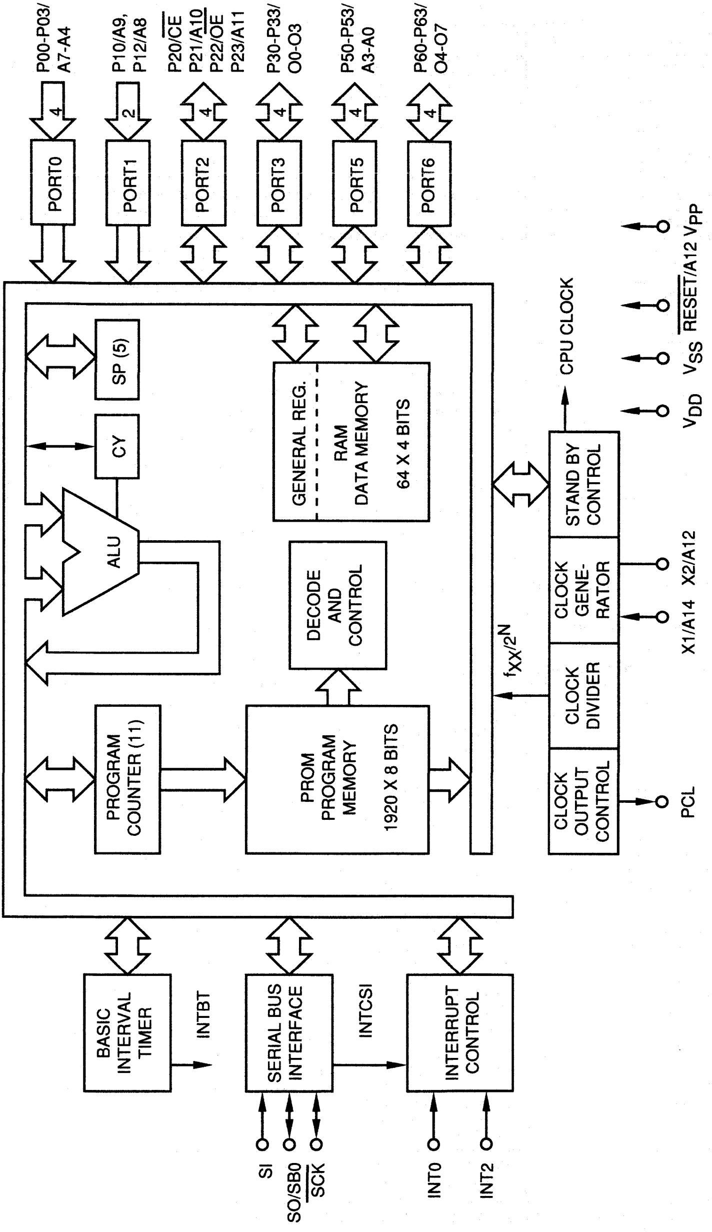
(*): Used by only μPD75P402.
To share printed circuit board with μPD75P402 in the μPD75402, set the NC pin which is applicable to the $V_{PP}$ in the μPD75402, to GND potential.

(2) PROM mode

Top view



### 10.3 Block Diagram



## 10.4 Pin Function List

**Table 4.1-1 Port Pins**

| Pin name | I/O | Also used for (Note) | Function |
|---|---|---|---|
| P00 | I | (A7) | 4-bit input port (Port 0). Internal pull-up resistor can be specified for P01-P03 in 3-bit units on software. |
| P01 | I/O | $\overline{\text{SCK}}$ (A6) | |
| P02 | I/O | SO/SB0 (A5) | |
| P03 | I | SI (A4) | |
| P10 | I | INT0 (A9) | 2-bit input port (port 1). P10 contains a sampling clock noise eliminator. P12 contains an analog delay noise eliminator. Internal pull-up resistor can be specified for P12 on software |
| P12 | | INT2 (A8) | |
| P20 | I/O | ($\overline{\text{CE}}$) | 4-bit input/output port (port 2). Input or output mode can be specified in 4-bit units. Internal pull-up resistor can be specified 4-bit units on software. |
| P21 | | (A10) | |
| P22 | | PCL ($\overline{\text{OE}}$) | |
| P23 | | (A11) | |
| P30-P33 | I/O | (O0-O3) | Programmable 4-bit input/output port (port 3). Input or output mode can be specifed bitwise. Internal pull-up resistor can be specified in 4-bit units on software. LED can be directly driven. |
| P50-P53 | I/O | (A3-A0) | 4-bit N-ch open drain input/output (port 5). Input or output mode can be specified in 4-bit units. LED can be directly driven. |
| P60-P63 | I/O | (O4-O7) | 4-bit input/output port (port 6). Input or output mode can be specified in 4-bit units. Internal pull-up resistor can be specified in 4-bit units on software. LED can be directly driven. |

Remarks: The μPD75P402 does not enable 8-bit input/output because the ports are not paired.

Note: The pins enclosed in parentheses are used in the PROM programming mode.

**Table 10.4-2 Pin Other Than Ports ( in Normal Operation Mode)**

| Pin name | I/O | Also used for (Note) | Function |
|---|---|---|---|
| INT0 | I | P10 (A9) | Edge-detected vectored interrupt request input pin (detection edge can be selected by using the mode register). Sampling clock noise elimination function is contained. |
| INT2 | I | P12 (A8) | Edge detection external test input pin (rising edge detection). Analog delay noise eliminator is contained. |
| SI | I | P03 (A4) | Serial data input pin. |
| SO | I/O | P02/SB0 (A5) | Serial data output pin. |
| $\overline{\text{SCK}}$ | I/O | P01 (A6) | Serial clock input/output pin. |
| SB0 | I/O | P02/SO (A5) | Serial bus input/output pin. |
| PCL | I/O | P22 ($\overline{\text{OE}}$) | Clock output pin. |
| X1 | | (A14) | System clock oscillation crystal or ceramic oscillator connection pins. To supply clock from the external, input the clock to X1 and its reverse phase to X2. |
| X2 | | (A13) | |
| $\overline{\text{RESET}}$ | I | (A12) | System reset input pin. Analog delay noise eliminator is contained. |
| $V_{DD}$ | — | — | Positive power supply pin. |
| $V_{SS}$ | — | — | GND potential pin. |
| $V_{PP}$ | — | — | Program voltage apply pin in the PROM programming mode.<br>In normal operation mode, connect the pin to $V_{SS}$. |

Note: The pins enclosed in parantheses are used in the PROM programming mode.

**Table 10.4-3 Pin Other Than Ports ( in PROM Programming Mode)**

| Pin name | I/O | Also used for (Note) | Function |
|---|---|---|---|
| A0-A3 | I | P53-P50 | Address input pins. |
| A4-A7 | I | P03-P00 | |
| A8 | I | P12 | |
| A9 | I | P10 | |
| A10 | I | P21 | |
| A11 | I | P23 | |
| A12 | I | $\overline{\text{RESET}}$ | |
| A13 | I | X2 | |
| A14 | I | X1 | |
| O0-O3 | I/O | P30-P33 | Data input/output pins. |
| O4-O7 | I/O | P60-P63 | |
| $\overline{\text{CE}}$ | I | P20 | Chip enable signal input pin. |
| $\overline{\text{OE}}$ | I | P22 | Output enable signal input pin. |
| $V_{PP}$ | | | Program voltage apply pin. Connect the pin to $V_{SS}$ in normal operation mode. |

10.4.1 Pins Input/Output Circuits
The µPD75P402 pin input/output circuits are shown schematically below.

**Table 10.4-4 Pin Input/Output Circuit Types**

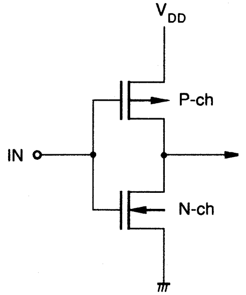
| Pin | I/O type | Pin | I/Otype |
|---|---|---|---|
| P00 | Ⓑ | P20, P21, P23 | E–B |
| P01/SCK | Ⓕ–A | P22/PCL | |
| P02/SO/SB0 | Ⓕ–B | P30-P33 | E–B |
| P03/SI | Ⓑ–C | P50-P53 | M–B |
| P10/INT0 | Ⓑ | P60-P63 | E–B |
| P12/INT2 | Ⓑ–C | RESET | Ⓑ |

Remarks: ○Schmitt trigger input

Pin I/O configurations

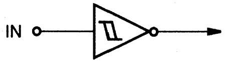
Following figures show the internal circuit configurations at the I/O ports.

(1) Type A (part of Type E–B)



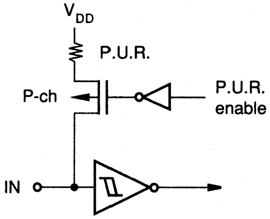
This is a CMOS standard input buffer.

(2) Type B



This is a Schmitt trigger input with hysteresis characteristics.

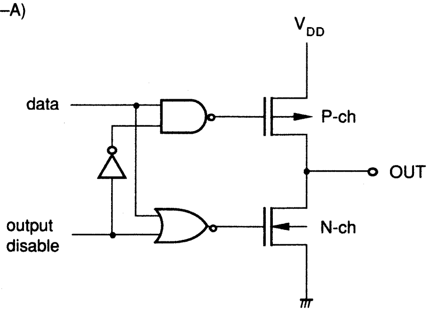
(3) Type B–C



P.U.R. : Pull-Up Resistor

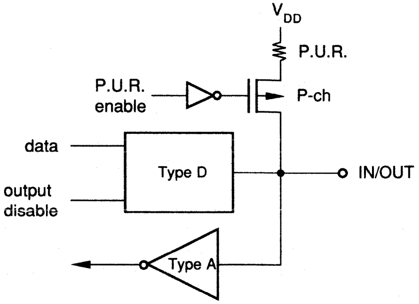
This is a Schmitt trigger input with hysteresis characteristics.

(4) Type D (Part of Type E–B, F–A)



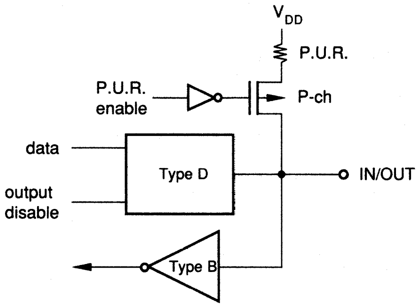
This is a push-pull output that can be set to high impedance (with both P-ch and N-ch off).

(5) Type E–B



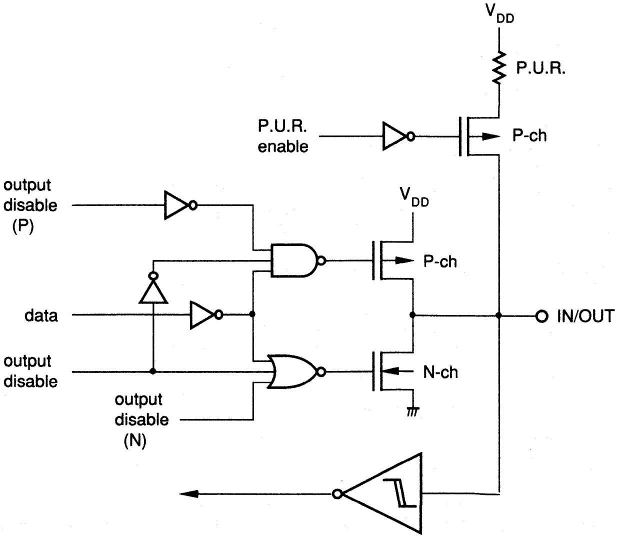
P.U.R. : Pull-Up Resistor

(6) Type F–A



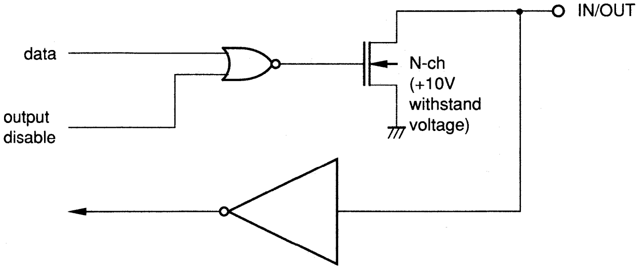
P.U.R. : Pull-Up Resistor

(7) Type F–B



P.U.R. : Pull-Up Resistor

(8) Type M–B



Middle-high voltage
input buffer
(+10V withstand voltage)

### 10.5 Differences Between μPD75P402 and μPD75402A

The μPD75P402 is a product provided by replacing μPD75402A internal mask ROM with one-time PROM. Table 10.5-1 lists the differences between the products,
Refer to the μPD75402A documents for detailed information on the CPU function and internal hardware.

**Table 10.5-1 Differences Between μPD75P402 and μPD75402A**

| Item | | μPD75P402 | μPD75402A |
|---|---|---|---|
| Program memory | | One-time PROM | Mask ROM |
| | | • 000H-77FH<br>• 1.920 x 8 bits | |
| Pull-up resistor | Ports 0-3, 6 | Can be specified on software. | |
| | Port 5 | None | Mask option |
| Operation supply voltage range | | 5V ± 10% | 2.7 V to 6.0 V |
| Pin connection | | They differ in $V_{PP}$ pin and multiplexed pins | |

### 10.6 One-Time PROM Programming

The μPD75P402 internal program memory is one-time PROM (1920 x 8 bits) that can be written electrically. To write programs into the one-time PROM, set the PROM mode and use the pins listed in Table 10.6-1.
The μPD75P402 programming characteristics are compatible with the μPD27C256A.

**Table 10.6-1 Pin Functions During One-Time PROM Programming**

| Pine name | Function |
|---|---|
| $V_{PP}$ | One-time PROM programming voltage input |
| A0-A14 (Note) | Address input |
| O0-O7 | Data input (during write) or data output (during verify or read) |
| $\overline{CE}$ | Chip enable input |
| $\overline{OE}$ | Output enable input |
| $V_{DD}$ | Supply voltage input |

Note: A11-A14 are dummy pins provided to make the same pin assignments as the μPD27C256A. Input a low level to the A11-A14 pins.

### 10.7 One-Time PROM Programming Operation Mode

When +6 V is applied to the $V_{DD}$ pin and +12.5 V to the $V_{PP}$ pin, the μPD75P402 is placed in the program write/verify mode. The mode is placed in the operation mode listed in Table 10.7-1 according to how the $\overline{CE}$ and $\overline{OE}$ pins are set.
The one-time PROM contents can be read by setting the μPD75P402 to the read mode.

**Table 10.7-1 One-Time PROM Programming Operation Mode**

| Pin / Mode | $\overline{CE}$ | $\overline{OE}$ | $V_{PP}$ | $V_{DD}$ | O0-O7 |
|---|---|---|---|---|---|
| Program write | L | H | +12.5 V | +6 V | Data input |
| Program verify | H | L | | | Data output |
| Program inhibit | H | H | | | High impedance |
| Read | L | L | +5 V | +5 V | Data output |
| Output disable | L | H | | | High impedance |
| Standby | H | L/H | | | High impedance |

Caution: When $V_{PP}$ is set to +12.5 V and $V_{DD}$ is set to +6 V, setting both $\overline{CE}$ and $\overline{OE}$ low is inhibited.

#### One-Time PROM Write Procedure

Data is written into one-time PROM at high speed as described below:

(1) Supply +6 V to $V_{DD}$ and +12.5 V to $V_{PP}$.

(2) Supply initial address.

(3) Supply write data.

(4) Supply a 1-ms program pulse (active low) to the CE pin.

(5) Enter the program inhibit mode.

(6) Enter the verify mode. If the data is not written normally, proceed to (7). If it is not written normally, repeat (3) to (6). If the data cannot be written after 25 repetitions, regard the one-time PROM as a defective device and stop write operation.

(7) Supply write data. Supply X (number of (3) - (6) repetitions) x 3 ms program pulses (additional write).

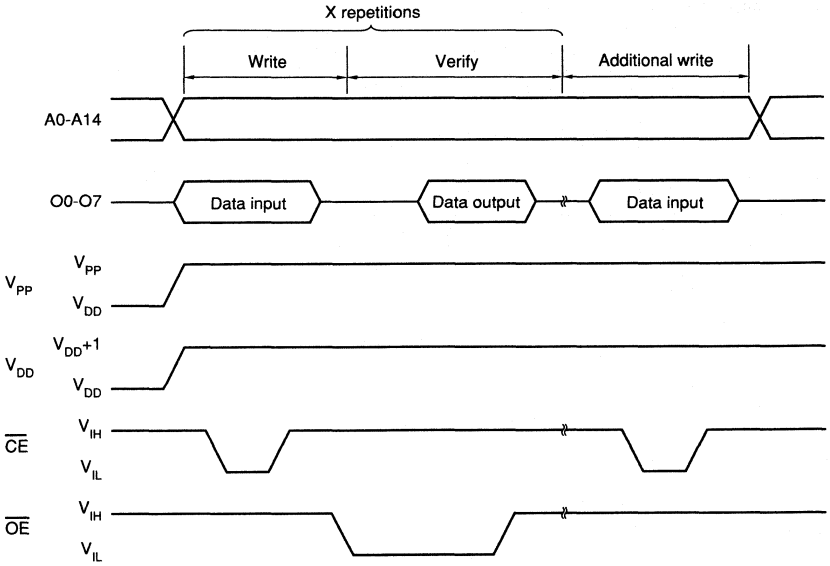
(8) Enter the program inhibit mode.

(9) Increment the address

(10) Repeat (3) - (9) until the last address is reached.

(11) Enter the read mode. Verify all byte data. If the byte is not written normally, regard the one-time PROM as a defective device.

(12) Turn off the power.



**One-Time PROM Write/Verify Timing**

### One-Time PROM Read Procedure

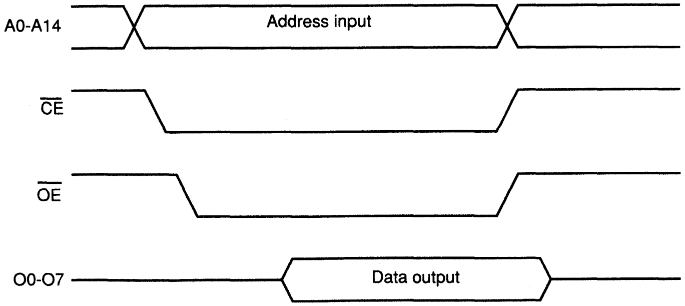
The one-time PROM contents can be read onto external data bus O0-O7 as described below:

(1) Supply 5 V to the $V_{DD}$ and $V_{PP}$ pins.

(2) Input the address of the data to be read to the A0-A14 pins.

(3) Enter the read mode.

(4) Output the data to the O0-O7 pins.



**One-Time PROM Read Timing**

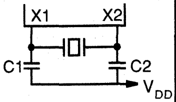
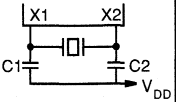
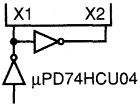
## 11. ELECTRICAL CHARACTERISTICS

**Absolute Maximum Ratings** (Ta = 25 °C)

| Item | Symbol | Conditions | | Ratings | Unit |
|---|---|---|---|---|---|
| Supply voltage | $V_{DD}$ | | | –0.3 to +7.0 | V |
| Input voltage | $V_{I1}$ | Other than port 5 | | –0.3 to $V_{DD}$ +0.3 | V |
| | $V_{I2}$ | Ports 5 | Open drain | –0.3 to +11 | V |
| Output voltage | $V_O$ | | | –0.3 to $V_{DD}$ +0.3 | V |
| High level output current | $I_{OH}$ | Single pin | | –15 | mA |
| | | All pins | | –30 | |
| Low level output current | $I_{OL}$<br>Note | One port pin of ports 0, 3, 5 and 6 | Peak value | 30 | mA |
| | | | Effective value | 15 | |
| | | Port 2 single pin | Peak value | 20 | |
| | | | Effective value | 10 | |
| | | Total of ports 0, 3 and 5, except P33 | Peak value | 100 | |
| | | | Effective value | 60 | |
| | | Total of ports 2, 6 and P33 | Peak value | 100 | |
| | | | Effective value | 60 | |
| Operation Temperature | $T_{opt}$ | | | –10 to +70 | °C |
| Storage Temperature | $T_{stg}$ | | | –65 to +150 | °C |

Note: Use the following formula to calculate the effective value. (Effective value) = (Peak value) x $\sqrt{duty}$

**Characteristics of Main System Clock Oscillator** (Ta = –10 to +70 °C, $V_{DD}$ = 5V ± to 10%)

| Oscillator | Recommended constants | Item | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Ceramic resonator |  | Note 1<br>Oscillation frequency ($f_{XX}$) | | 2.0 | | 5.0 | MHz |
| | | Note 2<br>Oscillation stabilization time | After $V_{DD}$ reaches the minimum value in the oscillator voltage range | | | 4 | ms |
| Crystal resonator |  | Note 1<br>Oscillation frequency ($f_{XX}$) | | 2.0 | 4.19 | 5.0 | MHz |
| | | Note 2<br>Oscillation stabilization time | | | | 10 | ms |
| External clock |  | Note 1<br>X1 input frequency ($f_X$) | | 2.0 | | 5.0 | MHz |
| | | X1 input high/low level width ($t_{XH}$, $t_{XL}$) | | 100 | | 500 | ns |

Note 1: The oscillation frequency and X1 input frequency are indicated only to express the characteristics of the oscillator. Refer to the AC characteristics for instruction execution time.

Note 2: The oscillation stabilization time is the time required for the oscillator to stabillate after $V_{DD}$ is applied or after the STOP mode is released.

**Recommended Ceramic Resonator**

| Manufacturer | Part name* | Frequency [MHz] | Capacitance [pF] | |
|---|---|---|---|---|
| | | | C1 | C2 |
| Murata Mfg. Co., Ltd. | CSAxxxMG093 | 2.00-2.44 | 30 | 30 |
| | CSAxxxMGU | 2.45-4.91 | 30 | 30 |
| Kyocera Corp. | KBR-2.0MS | 2.0 | 47 | 47 |
| | KBR-4.19MS | 4.19 | 33 | 33 |
| | KBR-5.0MS | 5.0 | 33 | 33 |

* xxx indicates frequency.

**DC Characteristics** (Ta = –10 to +70 °C, $V_{DD}$ = 5V ± 10%)

| Item | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| High level input voltage | $V_{IH1}$ | Ports 2 and 3 and 6 | | $0.7V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH2}$ | Ports 0, 1 and $\overline{\text{RESET}}$ pin | | $0.8V_{DD}$ | | $V_{DD}$ | V |
| | $V_{IH3}$ | Port 5 | Open drain | $0.7V_{DD}$ | | 10 | V |
| | $V_{IH4}$ | X1, X2 | | $V_{DD}$–0.5 | | $V_{DD}$ | V |
| Low level input voltage | $V_{IL1}$ | Ports 2, 3, 5, 6 | | 0 | | $0.3V_{DD}$ | V |
| | $V_{IL2}$ | Ports 0, 1 and $\overline{\text{RESET}}$ pin | | 0 | | $0.2V_{DD}$ | V |
| | $V_{IL3}$ | X1, X2 | | 0 | | 0.4 | V |
| High level output voltage | $V_{OH1}$ | Ports 0, 2, 3, 5, 6 | $I_{OH}$ = –1mA | $V_{DD}$–1.0 | | | V |
| | | | $I_{OH}$ = –100µA | $V_{DD}$–0.5 | | | V |
| Built-in pull-up resistor | $R_{L1}$ | Ports 0, 1, 2, 3, 6, (Except P00, P10) $V_{IN}$ = 0V | VDD =5.0V ± 10% | 15 | 40 | 80 | kΩ |
| Low level output voltage | $V_{OL1}$ | Ports 3, 5, 6 | $I_{OL}$ = 15mA | | 0.6 | 2.0 | V |
| | | Ports 0, 2, 3, 5, 6 | $I_{OL}$ = 1.6mA | | | 0.4 | V |
| | $V_{OL2}$ | SB0 open drain | Pull-up resistance: 1kΩ min. $V_{DD}$ = 4.5-6.0V | | | $0.2V_{DD}$ | V |
| High level input leakage current | $I_{LIH1}$ | $V_{IN} = V_{DD}$ | Other than indicated below | | | 3 | µA |
| | $I_{LIH2}$ | | X1, X2 | | | 20 | µA |
| | $I_{LIH3}$ | $V_{IN}$ = 10V | Port 5 (with open drain) | | | 20 | µA |
| Low level input leakage current | $I_{LIL1}$ | $V_{IN}$ = 0V | Other than indicated below | | | –3 | µA |
| | $I_{LIL2}$ | | X1, X2 | | | –20 | µA |
| High level output leakage current | $I_{LOH1}$ | $V_{OUT} = V_{DD}$ | Other than indicated below | | | 3 | µA |
| | $I_{LOH2}$ | $V_{OUT}$ = 10V | Port 5 (with open drain) | | | 20 | µA |
| Low level output leakage current | $I_{LOL}$ | $V_{OUT}$ = 0V | | | | –3 | µA |
| Supply current (Note 1) | $I_{DD1}$ | 4.19 MHz crystal oscillation C1=C2=22pF | $V_{DD}$ = 5V ± 10% (Note 2) | | 3.0 | 10 | mA |
| | $I_{DD2}$ | | HALT Mode $V_{DD}$ = 5V ± 10% | | 500 | 1500 | µA |
| | $I_{DD3}$ | STOP mode | $V_{DD}$ = 5V ± 10% | | 0.5 | 20 | µA |

Note 1: The current of the built-in pull-up resistor is not included.
Note 2: When operated in the high-speed mode with the processor clock control register (PCC) set to 0011.

**AC Characteristics** (Ta = –10 to 70 C, VDD = 5V ± 10%)

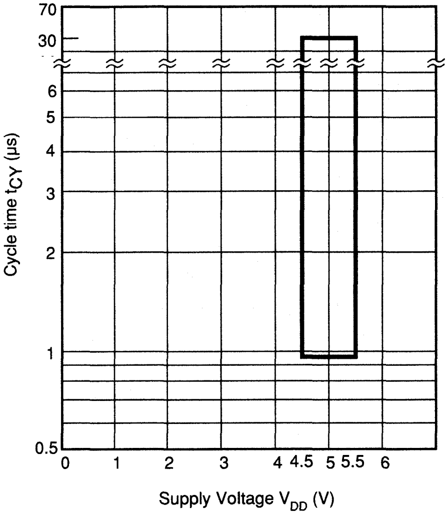
| Item | Symbol | Condition | MIN. | TYP. | MAX | Unit |
|---|---|---|---|---|---|---|
| (Note 1) Cycle time (minimum instruction execution time) | $t_{CY}$ | Operation with main system clock | 0.95 | | 32 | μs |
| Interrupt input high/low level width | $t_{INTH}$, | INT0 | Note 2 | | | μs |
| | $t_{INTL}$ | INT2 | 10 | | | μs |
| $\overline{\text{RESET}}$ low level width | $t_{RSL}$ | | 10 | | | μs |

Note 1: The cycle time (minimum instruction execution time) is determined by the oscillation frequency of the connected resonator, the system clock control register (SCC), and the processor clock control register (PCC). The figure below shows the $V_{DD}$ vs cycle time ($t_{CY}$) when operated with the main system clock.

Note 2: $2t_{CY}$ or $64/f_X$, depending on the setting of the interrupt mode register (IM0).

(with main system clock)

$t_{cy}$ vs. $V_{DD}$



**Capacitance** (Ta = 25 °C, $V_{DD}$ = 0V)

| Item | Symbol | Condition | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| Input capacitance | $C_{IN}$ | f = 1 MHz<br>Unmeasured pin turned to be 0V. | | | 15 | pF |
| Output capacitance | $C_{OUT}$ | | | | 15 | pF |
| Input/output capacitance | $C_{IO}$ | | | | 15 | pF |

**Serial Transfer Operation**

3line serial I/O mode (SCK ... Internal clock output)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY1}$ | | 1600 | | | ns |
| $\overline{\text{SCK}}$ high/low level width | $t_{KL1}$ | | $\frac{t_{KCY1}}{2}$ -50 | | | ns |
| SI set-up time (against $\overline{\text{SCK}}$ ↑ ) | $t_{SIK1}$ | | 150 | | | ns |
| SI hold time (against $\overline{\text{SCK}}$ ↑ ) | $t_{KSI1}$ | | 400 | | | ns |
| $\overline{\text{SCK}}$ ↓ → SO output delay time | $t_{KSO1}$ | $V_{dd}$ = 4.5 - 6.0V | | | 250 | ns |

3line serial I/O mode ($\overline{\text{SCK}}$ ... External clock input)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY2}$ | | 800 | | | ns |
| $\overline{\text{SCK}}$ high/low level width | $t_{KL2}$ | | 400 | | | ns |
| SI set-up time (against $\overline{\text{SCK}}$ ↑ ) | $t_{SIK2}$ | | 100 | | | ns |
| SI hold time (against $\overline{\text{SCK}}$ ↑ ) | $t_{KSI2}$ | | 400 | | | ns |
| $\overline{\text{SCK}}$↓ → SO output delay time | $t_{KSO2}$ | | | | 300 | ns |

Note: The output delay time (for rising edge) of the serial line must be shorter than 600 ns. For example, if SB0 is pulled up with 5K ohms, the total capacitance of the serial bus line must be no greater than 120 pF.

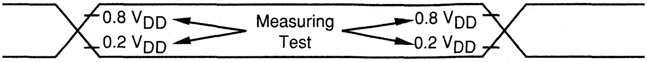
SBI mode ($\overline{\text{SCK}}$ ... Internal clock output (master))

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY3}$ | | 1600 | | | ns |
| $\overline{\text{SCK}}$ high/low level width | $t_{KL3}$ | | $\frac{t_{KCY1}}{2}-50$ | | | ns |
| SB0 set-up time (against $\overline{\text{SCK}}$ ↑ ) | $t_{SIK3}$ | | 150 | | | ns |
| SB0 hold time (against $\overline{\text{SCK}}$ ↑ ) | $t_{KSI3}$ | | $\frac{t_{KCY1}}{2}$ | | | ns |
| $\overline{\text{SCK}}$↓ → SB0,1 output delay time | $t_{KSO3}$ | | 0 | | 250 | ns |
| $\overline{\text{SCK}}$ ↑→ SB0 ↓ | $t_{KSB}$ | | $t_{KCY}$ | | | ns |
| SB0 ↓→ $\overline{\text{SCK}}$ ↓ | $t_{SBK}$ | | $t_{KCY}$ | | | ns |
| SB0 low level width | $t_{SBL}$ | | $t_{KCY}$ | | | ns |
| SB0 high level width | $t_{SBH}$ | | $t_{KCY}$ | | | ns |

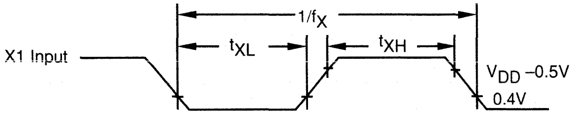
SBI mode ($\overline{\text{SCK}}$ ... External clock input (slave))

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{SCK}}$ cycle time | $t_{KCY4}$ | | 800 | | | ns |
| $\overline{\text{SCK}}$ high/low level width | $t_{KL4}$ | | 400 | | | ns |
| SB0 set-up time (against $\overline{\text{SCK}}$ ↑ ) | $t_{SIK4}$ | | 100 | | | ns |
| SB0 hold time (against $\overline{\text{SCK}}$ ↑ ) | $t_{KSI4}$ | | $\frac{t_{KCY1}}{2}$ | | | ns |
| $\overline{\text{SCK}}$↓ → SB0 output delay time | $t_{KSO4}$ | | 0 | | 300 | ns |
| $\overline{\text{SCK}}$ ↑→ SB0 ↓ | $t_{KSB}$ | | $t_{KCY}$ | | | ns |
| SB0 ↓→ $\overline{\text{SCK}}$ ↓ | $t_{SBK}$ | | $t_{KCY}$ | | | ns |
| SB0 low level width | $t_{SBL}$ | | $t_{KCY}$ | | | ns |
| SB0 high level width | $t_{SBH}$ | | $t_{KCY}$ | | | ns |

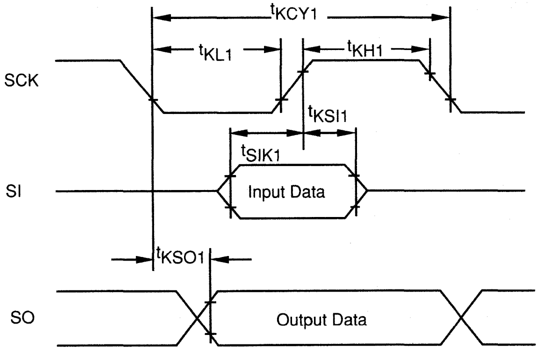
AC Timing Measurement Points (Except X1 input)



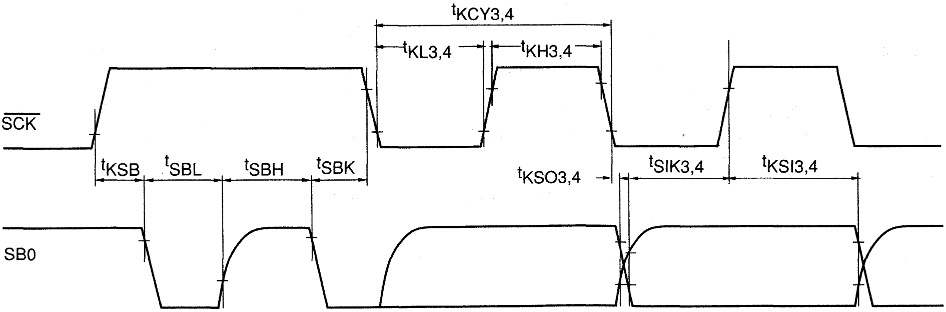
Clock Timing



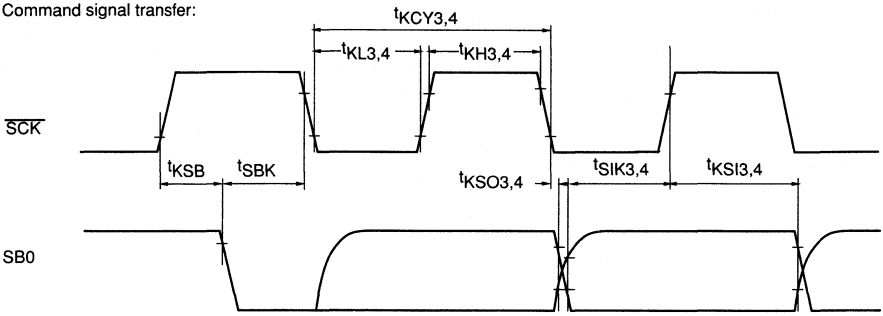
Serial Transfer Timing
3-line serial I/O mode



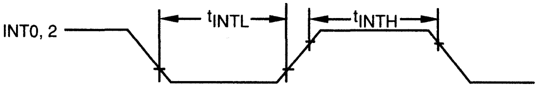
**Serial Transfer Timing**
Bus release signal transfer:



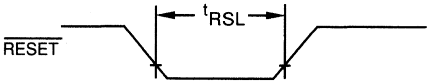
Command signal transfer:



Interrupt input timing



$\overline{\text{RESET}}$ input timing

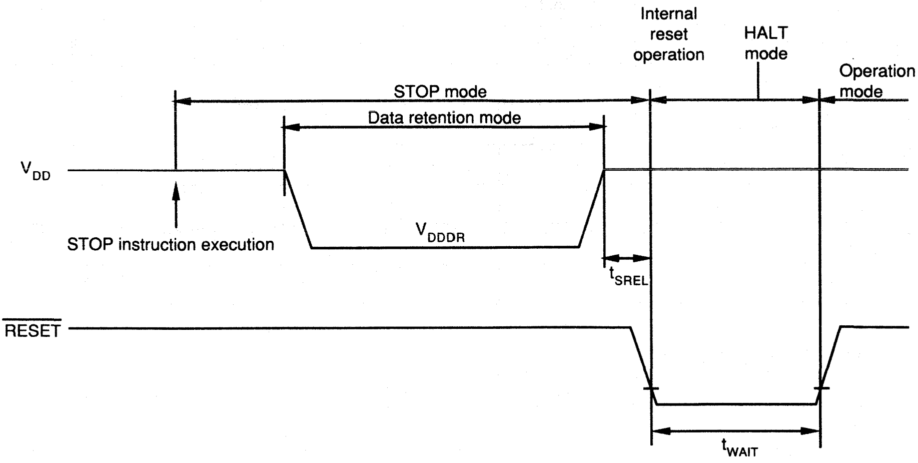


**Data Memory STOP Mode Low Voltage Data Retention Characteristic** (Ta = –10 to +70 °C)

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Data retention voltage | $V_{DDDR}$ | | 2.0 | | 6.0 | V |
| Data retention current (Note1) | $I_{DDDR}$ | $V_{DDR}$ = 2.0V | | 0.1 | 10 | µA |
| Release signal set time | $t_{SREL}$ | | 0 | | | µs |
| Oscillation stabilization time (Note 2) | $t_{WAIT}$ | Release by $\overline{\text{RESET}}$ input (Note 3) | | | 4 | ms |
| | | Release by $\overline{\text{RESET}}$ input (Note 4) | | | 10 | ms |

Note 1: Current in the internal pull-up resistors is not included.
Note 2: The oscillation stabilization time is the time required before beginning CPU operation in order to prevent unstable CPU operation when oscillation is initiated.
Note 3: After $V_{DD}$ reaches the oscillation voltage range, if ceramic resondator is used.
Note 4: After $V_{DD}$ reaches the oscillation voltage range, if crystal oscillator is used.

Data Retention Timing (when STOP mode is released by $\overline{\text{RESET}}$ input)



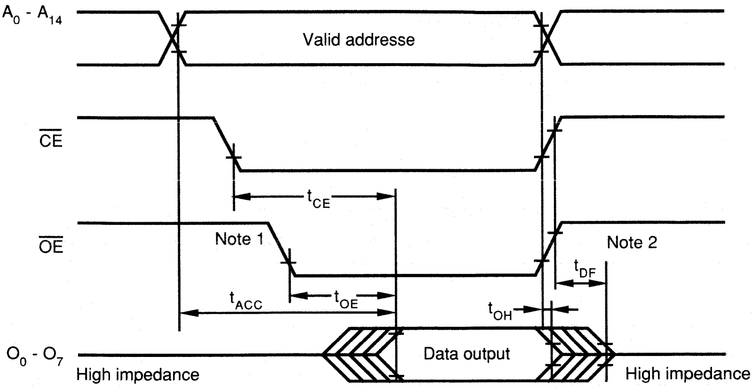
**DC Programming Charecteristics** (Ta = 25 ± 5 °C, VDD = 6.0 ± 0.25V, VPP = 12.5 ± 0.3V)

| Item | Symbol | Condition | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| High level input voltage | $V_{IH}$ | | $V_{DD}$-0.5 | | VDD | V |
| Low level input voltage | $V_{IL}$ | | 0 | | 0.4 | V |
| Input leakage current | $I_{LI}$ | $V_{IN} = V_{IL}$ or $V_{IH}$ | | | 20 | μA |
| High level output voltage | $V_{OH}$ | $I_{OH}$ = -400 μA | $V_{DD}$-1.0 | | | V |
| Low level output voltage | $V_{OL}$ | $I_{OL}$ = 1.6 mA | | | 0.4 | V |
| $V_{DD}$ supply current | $I_{DD}$ | | | | 30 | mA |
| $V_{PP}$ supply current | $I_{PP}$ | CE = $V_{IL}$, OE = $V_{IH}$ | | | 30 | mA |

**AC Programming Caracteristics** (Ta = 25 ± 5°C, $V_{DD}$ = 6.0 ± 0.25V, $V_{PP}$ = 12.5 ± 0.3V)

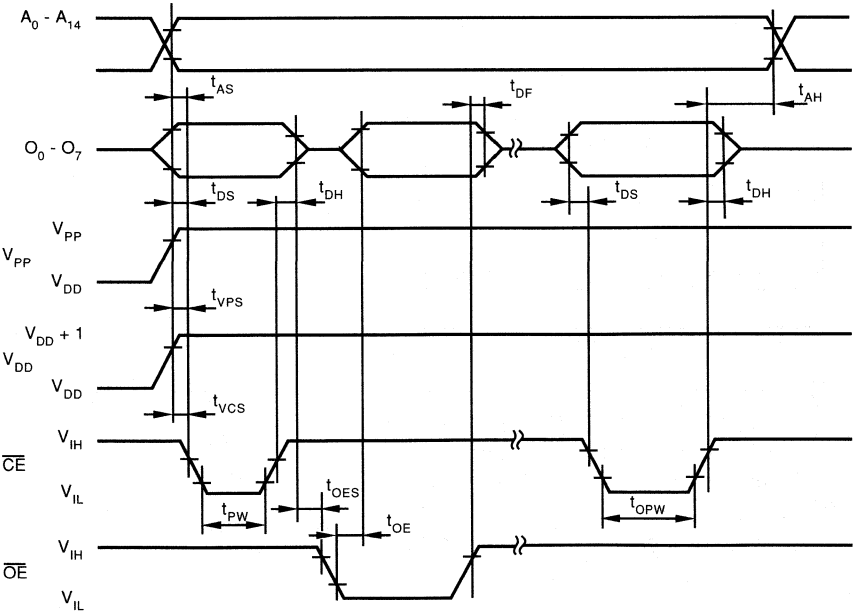
| Item | Symbol | Condition | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| Address setup time | $t_{AS}$ | | 2 | | | μs |
| $\overline{\text{OE}}$ setup time | $t_{OES}$ | | 2 | | | μs |
| Data input setup time | $t_{DS}$ | | 2 | | | μs |
| Address input retention time | $t_{AH}$ | | 2 | | | μs |
| Data input retention time | $t_{DH}$ | | 2 | | | μs |
| Data output float delay time from $\overline{\text{OE}}$ | $t_{DF}$ | | 0 | | 130 | ns |
| $V_{PP}$ setup time | $t_{VPS}$ | | 2 | | | μs |
| $V_{DD}$ setup time | $t_{VCS}$ | | 2 | | | μs |
| Initial program puls width | $t_{PW}$ | | 0.95 | 1.0 | 1.05 | ms |
| Additional program pulse width | $t_{OPW}$ | | 2.85 | | 78.75 | μs |
| Valid data delay time from $\overline{\text{OE}}$ | $t_{OE}$ | | | | 1 | μs |
| Data output delay time from address | $t_{ACC}$ | $\overline{\text{CE}} = \overline{\text{OE}} = V_{IL}$ | | | 2 | μs |
| Data output delay time from $\overline{\text{CE}}$ | $t_{CE}$ | $\overline{\text{OE}} = V_{IL}$ | | | 2 | μs |
| Data output delay time from $\overline{\text{OE}}$ | $t_{OE}$ | $\overline{\text{CE}} = V_{IL}$ | | | 1 | μs |
| Data output float delay time from $\overline{\text{OE}}$ | $t_{DF}$ | $\overline{\text{CE}} = V_{IL}$ | 0 | | 130 | ns |
| Output retention time from address | $t_{OH}$ | $\overline{\text{CE}} = \overline{\text{OE}} = V_{IL}$ | 0 | | | ns |

Read Mode Timing



Note 1 : To read within the range of $t_{ACC}$, set the delay time of $\overline{OE}$ input from the $\overline{CE}$ falling edge to $t_{ACC}$ - $t_{OE}$ (Max).
2: $t_{DF}$ is the time from the state in wich either $\overline{OE}$ or $\overline{CE}$ first becomes $V_{IH}$.

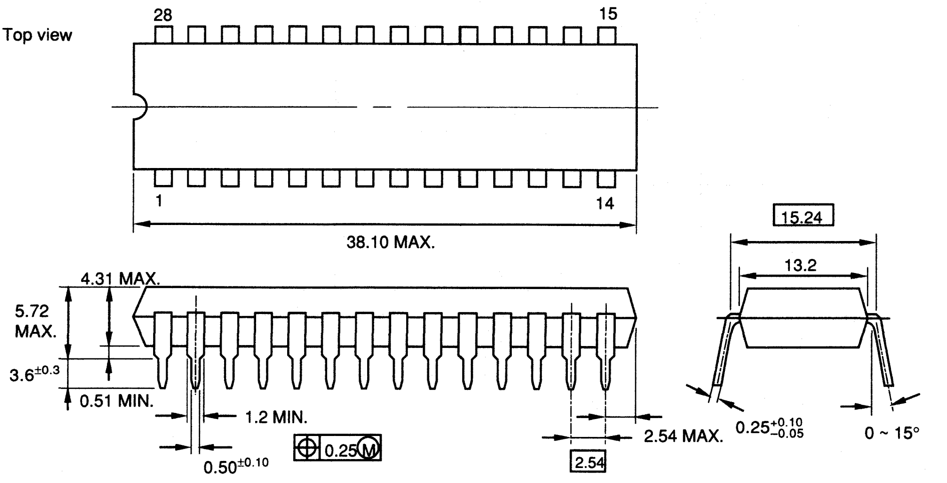
Program Mode Timing



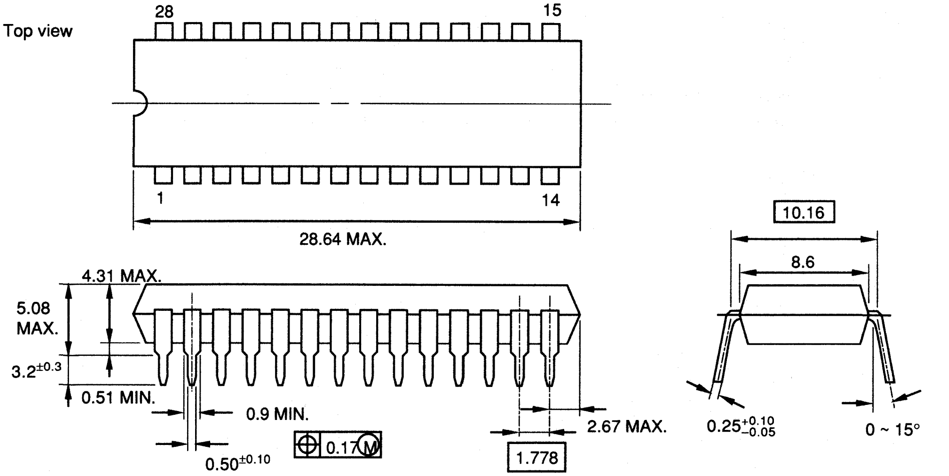
Caution 1: Apply voltage to $V_{DD}$ before $V_{PP}$ and turn off $V_{DD}$ after $V_{PP}$.
2: Present $V_{PP}$ from exceeding +13.5V containing overshoot.

### 11.1 Package Information

Package Dimension of 28-pin Plastic DIP (600 mil) (Unit: mm)

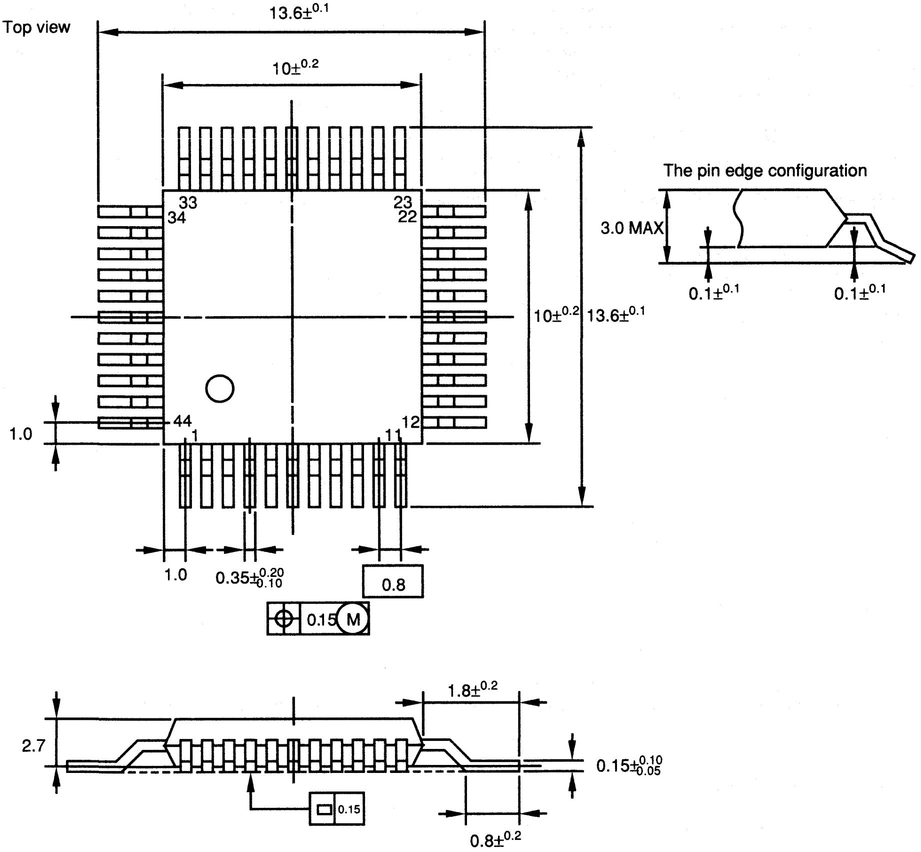


P28C-100-600A1



P28C-70-400A

Package Dimension of 44-pin Plastic QFP (Units: mm)



P44GB-80-3B4

## 12. μPD75402A INSTRUCTION SET

### 12.1 Featured Instructions

The μCOM-75x standard instruction set is an improvement of the instruction set for the μPD7500-series, which is the predecessor of the μCOM-75X family. The μPD75402A instruction set is a subset of the μCOM-75X instructions.
It is a new epoch-making instruction set that maintains compatibility with the μPD7500 series, and has the following features:
(1) Bit manipulation instructions for various applications
(2) Efficient 4-bit manipulation instructions
(3) 8-bit manipulation instructions comparable to 8-bit microcomputer instructions
(4) String effect instructions and base correction instructions to raise program efficiency
(5) Table reference instructions appropriate for consecutive reference
(6) 1-byte relative branch instructions
(7) Easy-to-understand and well arranged NEC standard mnemonics

12.1.1 Bit manipulation instructions
Bit manipulation can be performed by using the following instructions:

| | | | | |
|---|---|---|---|---|
| (i) | Bit setting | : | SET1 | mem.bit |
| | | | SET1 | fmem.bit* |
| (ii) | Bit clear | : | CLR1 | mem.bit |
| | | | CLR1 | fmem.bit* |
| (iii) | Bit test | : | SKT | mem.bit |
| | | | SKT | fmem.bit* |
| (iv) | Bit test | : | SKF | mem.bit |
| | | | SKF | fmem.bit* |
| (v) | Bit test and clear | : | SKTCLR | mem.bit* |
| (vi) | Boolean operations | : | AND1 | CY, fmem.bit* |
| | | | OR1 | CY, fmem.bit* |
| | | | XOR1 | CY, fmem.bit* |

The fmem.bit* is a bit address indicated by bit manipulation addressing.
This bit manipulation instruction is always applicable to input/output ports; therefore, the input/output ports can be handled very efficiently.

12.1.2 String effect instructions

The instruction set provides the following two types of string effect instructions:

(a) MOV A, #n4 or MOV XA, #n8
(b) MOV HL, #n8

"String effect" places these instructions at contiguous addresses.

```
Example:  A0   : MOV A, #0
          A1   : MOV A, #1
          XA7  : MOV XA, # 07
```

If the string effect instructions are placed as in this example, when the first execution address is A0, the following two instructions are replaced with NOP instructions for execution; when the first address is A1, the following one instruction is replaced with a NOP instruction for execution. That is, only the first executed instruction is valid, and all string effect instructions following are handled as NOP instructions.
The string effect instructions enable efficient constant loading into a given accumulator (A register, register pair XA) and data pointer (register pair HL).

12.1.3 Base correction instructions

Depending on the application, the result of 4-bit data addition or subtraction (made in binary) must be converted into a decimal number or subjected to sixenary correction, as with the time of day.
Thus, the μPD75402A instruction set contains base correction instructions to correct the result of 4-bit data addition to any desired base number.

(a) Base correction during addition

Assume that the correction base value is m.
By using the combination of

```
ADDS A, #16-m
ADDC A, @ HL ; A, C ← A + (HL) + C
ADDS A, #m
```

the accumulator and memory (HL) contents are added together, and the addition result is subjected to m correction. Overflow is left in the carry flag.
If a carry results from execution of the ADDC A, @ HL instruction, the following ADDS A, #n4 instruction is skipped. If no carry results, the ADDS A, # n4 instruction is executed. At this time, the skip function for this instruction is inhibited; even if a carry results from the addition, the following instruction will not be skipped.
Therefore, the program can be written following the ADDS A, #n4 instruction.

Example: To add the accumulator and memory together in decimal.

```
ADDS A, #6
ADDC A, @ HL ; A, C ← A + (HL) + C
ADDS A, #10
          :
          :
```

12.1.4 Skip instruction and number of machine cycles required for skip

The μPD75402A instruction set contains the skip function involved in condition decision.
If the skip condition is satisfied when a skip instruction (instruction having the skip condition) is executed, the skip function skips the next instruction and executeds the instruction following the skipped instruction.
When a skip occurs, one machine cycle is required for the skip.

## 12.2 Instruction Set and Operations

(1) Operand representation format and description method
In the operand field of each instruction, enter the operand according to the description method for the instruction operand representation format (details are based on the assembler specifications). When more than one entry exits under the description method, select one of the entries.
The uppercase alphabetic characters and the symbols #, @, ! and $ are keywords which must be entered exactly as shown.
For immediate data, enter the proper numeric value or label.
The register and flag abbreviations shown in Figure 12.3-1 can be entered as labels instead of mem, fmem, bit, etc. (however, labels that can be entered instead of fmem are limited).

| Representation format | Description method |
|---|---|
| reg<br>reg1 | X, A, H, L<br>X, H, L |
| rp | XA, HL |
| n4<br>n8 | 4-bit immediate data or label<br>8-bit immediate data or label |
| mem<br>bit | 8-bit immediate data or label*<br>2-bit immediate data or label |
| fmem | FB0H-FBFH, FF0H-FFFH immediate data or label |
| addr<br>caddr<br>faddr | 11-bit immediate data or label<br>11-bit immediate data or label<br>11-bit immediate data or label |
| PORTn<br>IEXXX | PORT0-PORT3, PORT5, PORT6<br>IEBT, IECSI, IE0, IE2 |

* When 8-bit data processing is performed, only an even address can be entered in mem.

(2) Legend on operation explanation

| | |
|---|---|
| A | : A register or 4-bit accumulator |
| H | : H register, |
| L | : L register, |
| X | : X register, |
| XA | : Register pair XA or 8-bit accumulator |
| HL | : Register pair HL |
| PC | : Program counter |
| SP | : Stack pointer |
| CY | : Carry flag or bit accumulator |
| PSW | : Program status word |
| PORTn | : Port n (n = 0-3, 5, or 6) |
| IME | : Interrupt master enable flag |
| IExxx | : Interrupt enable flag |
| PCC | : Processor clock control register |
| • | : Separation between address and bit |
| (xx) | : Contents addressed by xx |
| xxH | : Hexadecimal data |

(3) Explanation of symbols under the column addressing area

| | | |
|---|---|---|
| *1 | MB = 0 | Data memory addressing |
| *2 | MB = 0 (00H-3FH)<br>MB = 15 (80H-FFH) | |
| *3 | MBE = 15 : fmem = FB0H-FBFH,<br>FF0H-FFFH | |
| *4 | addr = 000H - 77FH | Program memory addressing |
| *5 | addr = (Current PC) -15~ (Current PC) -1,<br>(Current PC) +16~(Current PC) +2 | |
| *6 | caddr = 000H - 77FH | |
| *7 | faddr = 000H - 77FH | |

Remarks 1: MB denotes memory bank that can be accessed.
2: *4 and *7 indicate the areas that can be addressed.

(4) Explanation of the column number of machine cycles

S denotes the number of machine cycles required for skip instruction skip operation. The value of S varies as follows:

- When the subsequent instruction is not skipped: S = 0
- When the subsequent instruction is skipped: S = 1

One machine cycle is equal to one cycle of the CPU clock Φ for which the time can be selected among two types by setting the PCC.

| Instruction group | Mnemonic | Operand | Number of bytes | Number of machine cycles | Operation | Addressing area | Skip condition |
|---|---|---|---|---|---|---|---|
| Transfer instruction | MOV | A, #n4 | 1 | 1 | A ← n4 | | String effect A |
| | | XA, #n8 | 2 | 2 | XA ← n8 | | String effect A |
| | | HL, #n8 | 2 | 2 | HL ← n8 | | String effect B |
| | | A, @ HL | 1 | 1 | A ← (HL) | *1 | |
| | | @ HL, A | 1 | 1 | (HL) ← A | *1 | |
| | | A, mem | 2 | 2 | A ← (mem) | *2 | |
| | | XA, mem | 2 | 2 | XA ← (mem) | *2 | |
| | | mem, A | 2 | 2 | (mem) ← A | *2 | |
| | | mem, XA | 2 | 2 | (mem) ← XA | *2 | |
| | XCH | A, @ HL | 1 | 1 | A ↔ (HL) | *1 | |
| | | A, mem | 2 | 2 | A ↔ (mem) | *2 | |
| | | XA, mem | 2 | 2 | XA ↔ (mem) | *2 | |
| | | A, reg1 | 1 | 1 | A ↔ reg1 | | |
| | MOVT | XA, @ PCXA | 1 | 3 | XA ← $(PC_{13\text{-}8}+XA)_{ROM}$ | | |
| Operation instruction | ADDS | A, #n4 | 1 | 1+S | A ← A+n4 | | carry |
| | | A, @ HL | 1 | 1+S | A ← A+(HL) | *1 | carry |
| | ADDC | A, @ HL | 1 | 1 | A, CY ← A + (HL) + CY | *1 | |
| | AND | A, @ HL | 1 | 1 | A ← A $\wedge$ (HL) | *1 | |
| | OR | A, @ HL | 1 | 1 | A ← A $\vee$ (HL) | *1 | |
| | XOR | A, @ HL | 1 | 1 | A ← A $\veebar$ (HL) | *1 | |
| Accumulator handling | RORC | A | 1 | 1 | CY ← $A_0$, $A_3$ ← CY, $A_{n-1}$ ← $A_n$ | | |
| | NOT | A | 2 | 2 | A ← $\overline{A}$ | | |
| Increment and decrement instructions | INCS | reg | 1 | 1+S | reg ← reg +1 | | reg = 0 |
| | | mem | 2 | 2+S | (mem) ← (mem) +1 | *2 | (mem) = 0 |
| | DECS | reg | 1 | 1+S | reg ← reg - 1 | | reg = FH |
| Comparison instructions | SKE | reg, #n4 | 2 | 2+S | Skip if reg = n4 | | reg = n4 |
| | | A, @ HL | 1 | 1+S | Skip if A = (HL) | *1 | A = (HL) |
| Carry flag handling instructions | SET1 | CY | 1 | 1 | CY ← 1 | | |
| | CLR1 | CY | 1 | 1 | CY ← 0 | | |
| | SKT | CY | 1 | 1+S | Skip if CY = 1 | | CY = 1 |
| | NOT1 | CY | 1 | 1 | CY ← $\overline{CY}$ | | |
| Memory bit manipulation instruction | SET1 | mem.bit | 2 | 2 | (mem.bit) ← 1 | *2 | |
| | | fmem.bit | 2 | 2 | (fmem.bit) ← 1 | *3 | |
| | CLR1 | mem.bit | 2 | 2 | (mem.bit) ← 0 | *2 | |
| | | fmem.bit | 2 | 2 | (fmem.bit) ← 0 | *3 | |
| | SKT | mem.bit | 2 | 2+S | Skip if (mem.bit) = 1 | *2 | (mem.bit) = 1 |
| | | fmem.bit | 2 | 2+S | Skip if (fmem.bit) = 1 | *3 | (fmem.bit) = 1 |
| | SKF | mem.bit | 2 | 2+S | Skip if (mem.bit) = 0 | *2 | (mem.bit) = 0 |
| | | fmem.bit | 2 | 2+S | Skip if (fmem.bit) = 0 | *3 | (fmem.bit) = 0 |
| | SKTCLR | fmem.bit | 2 | 2+S | Skip if (fmem.bit) = 1 and clear | *3 | (fmem.bit) = 1 |

| Instruction group | Mnemonic | Operand | Number of bytes | Number of machine cycles | Operation | Addressing area | Skip condition |
|---|---|---|---|---|---|---|---|
| Memory bit manipulation instruction | AND1 | CY, fmem.bit | 2 | 2 | CY ← CY∧ (fmem.bit) | *3 | |
| | OR1 | CY, fmem.bit | 2 | 2 | CY ← CY ∨(fmem.bit) | *3 | |
| | XOR1 | CY, fmem.bit | 2 | 2 | CY ← CY∨ (fmem.bit) | *3 | |
| Branch instructions | BR | addr | – | – | PC$_{10-0}$ ← addr<br>(Select appropriate instruction from among BRCB !caddr, and BR $addr according to the assembler being used.) | *4 | |
| | | $addr | 1 | 2 | PC$_{10-0}$ ← addr | *5 | |
| | BRCB | !caddr | 2 | 2 | PC$_{10-0}$ ← caddr$_{10-0}$ | *6 | |
| Subroutine stack control instructions | CALLF | !faddr | 2 | 2 | (SP -4) (SP -1)<br>(SP -2) ← PC10-0<br>(SP -3) ← 0000<br>PC$_{10-0}$ ← faddr,<br>SP ← SP -4 | *7 | |
| | RET | | 1 | 3 | PC$_{10-0}$ ← (SP) (SP +3) (SP +2)<br>SP ← SP +4 | | |
| | RETS | | 1 | 3+S | PC$_{10-0}$ ← (SP) (SP +3) (SP +2)<br>SP ← SP +4, then skip unconditionally | | Unconditional |
| | RETI | | 1 | 3 | PC$_{10-0}$ ← (SP) (SP +3) (SP +2)<br>PSW ← (SP +4) (SP +5),<br>SP ← SP +6 | | |
| | PUSH | rp | 1 | 1 | (SP -1) (SP -2) ← rp,<br>SP ← SP -2 | | |
| | POP | rp | 1 | 1 | rp ← (SP +1) (SP),<br>SP ← SP +2 | | |
| Interrupt control instructions | EI | | 2 | 2 | IME ← 1 | | |
| | | IEXXX | 2 | 2 | IEXXX ← 1 | | |
| | DI | | 2 | 2 | IME ← 0 | | |
| | | IEXXX | 2 | 2 | IEXXX ← 0 | | |
| Input/output instructions | IN | A, PORT$_n$ | 2 | 2 | A ← PORT$_n$ (n=port number) | | |
| | OUT | PORT$_n$, A | 2 | 2 | PORT$_n$ ← A (n=port number) | | |
| CPU control instructions | HALT | | 2 | 2 | Set HALT Mode (PCC.2 ← 1) | | |
| | STOP | | 2 | 2 | Set STOP Mode (PCC.3 ← 1) | | |
| | NOP | | 1 | 1 | No Operation | | |

### 12.3 Operation Code of Each Instruction

(1) Explanation of operation code symbols

| $R_1$ | $R_0$ | reg | | |
|---|---|---|---|---|
| 0 | 0 | A | reg | |
| 0 | 1 | X | | reg1 |
| 1 | 0 | L | | |
| 1 | 1 | H | | |

| $P_2$ | $P_1$ | reg-pair | |
|---|---|---|---|
| 0 | 0 | XA | rp |
| 0 | 1 | HL | |

| $N_2$ | $N_1$ | $N_0$ | IExxx |
|---|---|---|---|
| 0 | 0 | 0 | IEBT |
| 1 | 0 | 1 | IECSI |
| 1 | 1 | 0 | IE0 |
| 1 | 1 | 1 | IE2 |

In : Immediate data for n4, n8
Dn : Immediate data for mem
Bn : Immediate data for bit
Nn : Immediate data for *n* or IExxx
An : Immediate data for (relative address distance to branch destination address (2-16)) -1
Sn : Immediate data for one´s complement of (relative address distance to branch destination address (15-1))

(2) Bit manipulation addressing operation codes

The second byte of the operation code corresponding to each addressing mode shown above is as listed below:

| *1 | Second byte of operation code | Bits that can be accessed |
|---|---|---|
| fmem.bit | 1 0 $B_1$ $B_0$ $F_3$ $F_2$ $F_1$ $F_0$ | FB0H-FBFH bits that can be manipulated |
| | 1 1 $B_1$ $B_0$ $F_3$ $F_2$ $F_1$ $F_0$ | FF0H-FFFH bits that can be manipulated |

Bn : Immediate data for bit
Fn : Immediate data for fmem
(low-order four bits of address are indicated)

| Instruction group | Mnemonic | Operand | Operation code | | |
|---|---|---|---|---|---|
| | | | $B_1$ | $B_2$ | $B_3$ |
| Transfer instructions | MOV | A, #n4 | 0 1 1 1 $I_3$ $I_2$ $I_1$ $I_0$ | | |
| | | rp, #n8 | 1 0 0 0 1 $P_2$ $P_1$ 1 | $I_7$ $I_6$ $I_5$ $I_4$ $I_3$ $I_2$ $I_1$ $I_0$ | |
| | | A, @HL | 1 1 1 0 0 0 0 1 | | |
| | | @HL, A | 1 1 1 0 1 0 0 0 | | |
| | | A, mem | 1 0 1 0 0 0 1 1 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | XA, mem | 1 0 1 0 0 0 1 0 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ 0 | |
| | | mem, A | 1 0 0 1 0 0 1 1 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | mem, XA | 1 0 0 1 0 0 1 0 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ 0 | |
| | XCH | A, @HL | 1 1 1 0 1 0 0 1 | | |
| | | A, mem | 1 0 1 1 0 0 1 1 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | XA, mem | 1 0 1 1 0 0 1 0 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ 0 | |
| | | A, reg1 | 1 1 0 1 1 0 $R_1$ $R_0$ | | |
| | MOVT | XA,@PCXA | 1 1 0 1 0 0 0 0 | | |
| Arithmetic and logical instructions | ADDS | A, #n4 | 0 1 1 0 $I_3$ $I_2$ $I_1$ $I_0$ | | |
| | | A, @HL | 1 1 0 1 0 0 1 0 | | |
| | ADDC | A, @HL | 1 0 1 0 1 0 0 1 | | |
| | AND | A,@HL | 1 0 0 1 0 0 0 0 | | |
| | OR | A,@HL | 1 0 1 0 0 0 0 0 | | |
| | XOR | A,@HL | 1 0 1 1 0 0 0 0 | | |
| ACC | 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 | |
| Inc/dec | INCS | reg | 1 1 0 0 0 0 $R_1$ $R_0$ | | |
| | | 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 0 $R_1$ $R_0$ | | |
| Compare | SKE | reg, #n4 | 1 0 0 1 1 0 1 0 | $I_3$ $I_2$ $I_1$ $I_0$ 0 0 $R_1$ $R_0$ | |
| | | A, @HL | 1 0 0 0 0 0 0 0 | | |
| Carry | 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 instructions | SET1 | mem.bit | 1 0 $B_1$ $B_0$ 0 1 0 1 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | fmem.bit | 1 0 0 1 1 1 0 1 | bit-addr | |
| | CLR1 | mem.bit | 1 0 $B_1$ $B_0$ 0 1 0 0 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | fmem.bit | 1 0 0 1 1 1 0 0 | bit-addr | |
| | SKT | mem.bit | 1 0 $B_1$ $B_0$ 0 1 1 1 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | fmem.bit | 1 0 1 1 1 1 1 1 | bit-addr | |
| | SKF | mem.bit | 1 0 $B_1$ $B_0$ 0 1 1 0 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | fmem.bit | 1 0 1 1 1 1 1 0 | bit-addr | |

| Instruction group | Mnemonic | Operand | Operation code | | |
|---|---|---|---|---|---|
| | | | B$_1$ | B$_2$ | B$_3$ |
| Memory bit instruction | SKTCLR | fmem.bit | 1 0 0 1 1 1 1 1 | bit-addr | |
| | AND1 | CY, fmem.bit | 1 0 1 0 1 1 0 0 | bit-addr | |
| | OR1 | CY, fmem.bit | 1 0 1 0 1 1 1 0 | bit-addr | |
| | XOR1 | CY, fmem.bit | 1 0 1 1 1 1 0 0 | bit-addr | |
| Branch | BR | (+16) - (+2)<br>$addr | 0 0 0 0 $A_3$ $A_2$ $A_1$ $A_0$ | | |
| | | (−1) - (−15) | 1 1 1 1 $S_3$ $S_2$ $S_1$ $S_0$ | | |
| | BRCB | !caddr | 0 1 0 1 0 ← | caddr → | |
| Subroutine stack | CALLF | !faddr | 0 1 0 0 0 ← | faddr → | |
| | RET | | 1 1 1 0 1 1 1 0 | | |
| | RETS | | 1 1 1 0 0 0 0 0 | | |
| | RETI | | 1 1 1 0 1 1 1 1 | | |
| | PUSH | rp | 0 1 0 0 1 0 $P_1$ 1 | | |
| | POP | rp | 0 1 0 0 1 0 $P_1$ 0 | | |
| In/out | IN | A, PORTn | 1 0 1 0 0 0 1 1 | 1 1 1 1 0 $N_2$ $N_1$ $N_0$ | |
| | OUT | PORTn, A | 1 0 0 1 0 0 1 1 | 1 1 1 1 0 $N_2$ $N_1$ $N_0$ | |
| Interrupt | EI | | 1 0 0 1 1 1 0 1 | 1 0 1 1 0 0 1 0 | |
| | | IEXXX | 1 0 0 1 1 1 0 1 | 1 0 0 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 | |
| | | IEXXX | 1 0 0 1 1 1 0 0 | 1 0 0 1 1 $N_2$ $N_1$ $N_0$ | |
| CPU control | HALT | | 1 0 0 1 1 1 0 1 | 1 0 1 0 0 0 1 1 | |
| | STOP | | 1 0 0 1 1 1 0 1 | 1 0 1 1 0 0 1 1 | |
| | NOP | | 0 1 1 0 0 0 0 0 | | |

## 12.4 Instruction Functions and Applications

### 12.4.1 Transfer instructions

**MOV A, #n4**

Function: A ← n4    n4 = $I_{3-0}$: 0-FH

This instruction transfers 4-bit immediate data n4 to the A register (4-bit accumulator).

It has a string effect (group A). If the MOV A, #n4 or MOV XA, #n8 instruction is consecutively placed, the instruction that follows the executed instruction will be replaced with an NOP instruction.

Application examples:

(1) Sets 0BH into the accumulator:
MOV A, #0BH

(2) Selects data to be output to Port 3 from 0 to 2:

```
A0:   MOV A, #0
A1:   MOV A, #1
A2:   MOV A, #2
      OUT PORT3, A
```

**MOV rp, #n8**

Function: rp ← #n8    n8 = $I_{7-0}$: 00H-FFH

This instruction transfers 8-bit immediate data n8 to the register pair rp (XA, HL).

It has a string effect if the XA or HL is specified for rp. The string effect includes Group A (MOV A, #n4 and MOV XA, #n8 instruction). If instructions belonging to the same group are consecutively placed in a program, the string effect instruction that follows the executed instruction is replaced with an NOP instruction.

Application example: Sets 5FH into HL register pair.
MOV HL, #5FH

**MOV A, @HL**

Function: A ← (HL)

This instruction transfers the contents of the data memory addressed by register pair HL to the A register.

Application example: Transfers data in addresses 3EH to the register A.
MOV HL, #3EH
MOV A, @HL

**MOV @HL, A**

Function: (HL) ← A

This instruction transfers the A register value to the data memory location addressed by the register pair HL.

**MOV A, mem**

Function: A ← (mem)    mem = $D_{7-0}$: 00H-3FH

This instruction transfers the contents of the data memory location addressed by 8-bit immediate data mem to the A register.

**MOV XA, mem**

Function: A ← (mem), X ← (mem+1)    mem = $D_{7-0}$: 00H-3EH

This instruction transfers the contents of the data memory location addressed by 8-bit immediate data mem to the A register, and the contents of the next data memory location to the X register.

The address specifiable with mem is an even address.

Application example: Transfers the data in location 40H and 41H to the register pair XA:
MOV XA, 40H

**MOV mem, A**

Function: (mem)← A, mem = $D_{7-0}$ 00H-3FH

This instruction transfers the A register value to the data memory location addressed by 8-bit immediate data mem.

**MOV mem, XA**

Function: (mem)← A, (mem+1)← X mem = $D_{7-0}$: 00H-3EH

This instruction transfers the A register value to the data memory location addressed by 8-bit immediate data mem and the X register value to the next data memory location.

The address specifiable by mem is an even address.

**XCH A, @HL**

Function: A← (HL)

This instruction exchanges the A register value with the contents of the data memory location addressed by the register pair (HL).

Application example: Exchanges data in data memory locations 20 to 2FH with those in locations 30 to 3FH:

```
        MOV      HL, #30H
LOOP:   XCH A,   @HL;  A↔ (3x)
        MOV      H, #2
        XCH A,   @HL;  A↔ (2x)
        MOV      H, #3
        XCH A,   @HL;  A↔ (3x)
        INCS     L
        BR       LOOP
```

**XCH A, mem**

Function: A↔ (mem), X↔ (mem+1) mem = $D_{7-0}$: 00H-3FH

This instruction exchanges the A register value with the contents of the data memory location addressed by 8-bit immediate mem.

**XCH XA, mem**

Function: A↔ (mem), X↔ (mem+1) mem = $D_{7-0}$: 00H-3EH

This instruction exchanges the A register value with the contents of the data memory location addressed by 8-bit immediate data mem and the X register value with the contents of the next data memory location. The address specifiable with mem is an even address.

**XCH A, reg1**

Function: A↔ reg1

This instruction exchanges the A register value with the value of the register reg1 (X, H, L).

12.4.2 Table reference instructions

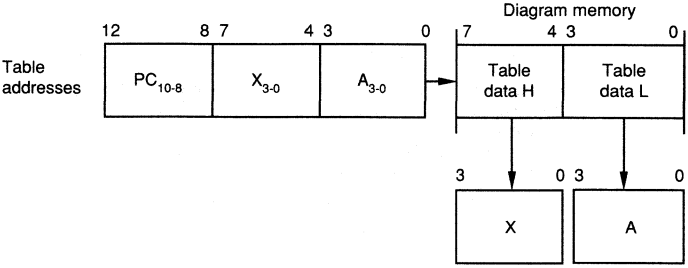
**MOVT XA, @PCXA**

Function: XA ← ROM ($PC_{10-8}$+XA)

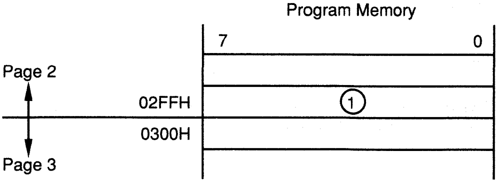
This instruction transfers the lower 4 bits of the table data in the program memory location addressed by the upper 3 bits ($PC_{10-8}$) of the program counter (PC) plus the value of the register pair XA to the A register, and the higher 4 bits of the same table data to the X register.

The higher 3 bits of the table address are determined by the PC value at the time of the execution of this instruction. Necessary data must be programmed in the table area beforehand by using an assembler's directive (DB instruction). The program counter value is not affected by the execution of this instruction.

The instruction is useful to reference table data consecutively.



Caution: Normally, the MOVT XA, @PCXA instruction references to the table data on the page where the instruction is located. If the instruction is located in address xxFFH, it references the table data on the next page.



For example, the MOVT XA, @PCXA instruction exists at the position ① as shown above, the table data on page 3 rather than page 2, specified by the register pair XA contents is transferred to the XA register pair.

12.4.3 Operational instructions

**ADDS A, #n4**

Function: A ← A + n4; Skip if carry. n4 = $I_{3\text{-}0}$: 0H-FH

This instruction adds 4-bit immediate data n4 to the A register value (binary). If a carry is produced, the instruction following this instruction is skipped. The carry flag is not affected.

When this instruction is combined with the ADDC A, @HL instruction, it functions as a base correction instruction (see 1.4).

**ADDS A, @HL**

Function: A ← A + (HL); Skip if carry.

This instruction adds the contents of the data memory location addressed by the register pair HL to the A register value (binary). If a carry is produced, the instruction following this instruction is skipped. The carry flag is not affected.

**ADDC A, @HL**

Function: A, CY ← A + (HL) + CY

This instruction adds the contents (including the carry flag) of the data memory location addressed by the register pair HL to the A register value in binary. If a carry is produced, the carry flag is set; if not, the flag is reset. If a carry is produced when this instruction is followed by the ADDS A, #n4 instruction, the ADDS A, #n4 instruction will be skipped. If no carry is produced, the ADDS A, #n4 instruction will be executed with its skip function inhibited.

Therefore, these instruction are usable for base correction operation when combined (see 1.4).

**AND A, @HL**

Function: A ← A ∧ (HL)

This instruction ANDs the A register value with the contents of the data memory location addressed by the register pair HL and sets the result into the A register.

**OR A, @HL**

Function: A ← A ∨ (HL)

ORs the contents of the A register and the data memory contents addressed by register pair HL and sets the result into the A register.

**XOR A, @HL**

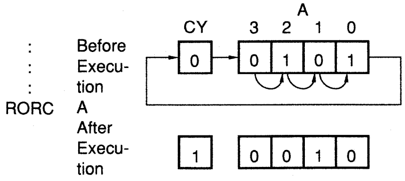
Function: A ← A ∀ (HL)

Exclusive-ORs the contents of the A register and the data memory contents addressed by register pair HL and sets the result into the A register.

**RORC A**

Function: CY ← $A_0$ $A_n$ to $A_1$ ← $A_n$, $A_3$ ← CY (n = 1 to 3)

Rotates the contents of the A register (4-bit accumulator), including the carry flag, to the right one bit at a time.



**NOT A**

Function: A ← A

Takes the one´s complement (inverts each bit) of the A register (4-bit accumulator).

**INCS reg**

Function: reg ← reg + 1; Skip if reg = 0

Increments the contents of register reg (X, A, H, L).
When the contents of register reg become 0 as the result of incrementing, skips the next instruction.

**INCS mem**

Function: (mem) ← (mem) + 1; Skip if (mem) = 0, (mem) = $D_{7\text{-}0}$:
00H-FFH

This instruction increments the contents of the data memory location addressed by the 8-bit immediate data mem. If the contents of the data memory location become zero when incremented, the instruction following this instruction will be skipped.

**DECS reg**

Function: reg ← reg – 1; Skip if reg = FH.

This instruction decrements the value of the register reg (X, A, H, L). If reg = FH when decremented, the instruction following this instruction will be skipped.

12.4.4 Compare instructions

**SKE reg, #n4**

Function: Skip if reg = n4 n4 = $I_{3\text{-}0}$: 0H-FH

This instruction skips the instruction following it if the value of the register reg (X, A, H, L) is equal to the 4-bit immediate data n4.

**SKE A, @HL**

Function: Skip if A = (HL)

This instruction skips the instruction following it if the A register value is equal to the value of the data memory location addressed by the register pair HL.

12.4.5 Carry flag manipulation instructions

**SET1 CY**

Function: CY ← 1

This instruction sets the carry flag.

**CLR1 CY**

Function: CY ← 0

This instruction clears the carry flag.

**SKT CY**

Function: Skip if CY ← 1

This instruction skips the instruction following it if the carry flag is set to 1.

**NOT1 CY**

Function: CY ← $\overline{CY}$

This instruction reverses the carry flag status, from 0 to 1 or vice versa.

12.4.6 Memory bit manipulation instructions

**SET1 mem. bit**

Function: (mem. bit) ← 1 mem = $D_{7-0}$: 00H-3FH, bit = $B_{1-0}$: 0-3

This instruction sets the data memory bit specified by the 2-bit immediate data "bit" at the location addressed by the 8-bit immediate data "mem".

**SET1 fmem. bit**

Function: (bit specified by operand) ← 1

Sets the data memory bit specified by bit operation addressing (fmem.bit).

**CLR1 mem. bit**

Function: (mem. bit) ← 0 mem 0 $D_{7-0}$: 00F-3FH, bit = $B_{1-0}$: 0-3

This instruction clears the data memory bit specified by the 2-bit immediate data "bit" at the location specified by the 8-bit immediate data "mem".

**CLR1 fmem. bit**

Function: (bit specified by operand) ← 0

Clears the data memory bit specified by bit operation addressing (fmem.bit).

**SKT mem. bit**

Function: Skip if (mem. bit) = 1 mem = $D_{7-0}$: 00H-3FH, bit = $B_{1-0}$: 0-3

This instruction skips the instruction following it if the data memory bit specified by the 2-bit immediate data "bit" at the location specified by the 8-bit immediate data "mem" is equal to 1.

**SKT fmem. bit**

Function: Skip if (bit specified by operand) = 1

If the data memory bit specified by bit operation addressing (fmem.bit) is 1, skips the next instruction.

**SKF mem. bit**

Function: Skip if (mem. bit) = 0 mem = $D_{7-0}$: 00H-3FH, bit = $B_{1-0}$: 0-3

This instruction skips the instruction following it if the data memory bit specified by the 2-bit immediate data "bit" at the location specified by the 8-bit immediate data "mem" is equal to 0.

**SKF fmen. bit**

Function: Skip if (bit specified by operand) = 0

If the contents of the data memory bit specified by bit operation addressing (fmem.bit) is 0, skips the next instruction.

**SKTCLR fmem. bit**

Function: Skip if (bit specified by operand) = 1 then clear

If the data memory bit specified by bit operation addressing (fmem.bit) is 1, skips the next instruction and clears that bit to 0.

**AND1 CY, fmem. bit**

Function: $CY \leftarrow CY \wedge$ (bit specified by operand)

ANDs the contents of the carry flag and the contents of the data memory bit specified by bit operation addressing (fmem.bit) and sets the result into the carry flag.

**OR1 CY, fmem. bit**

Function: $CY \leftarrow CY \vee$ (bit specified by operand)

ORs the contents of the carry flag and the contents of the data memory bit specified by bit operation addressing (fmem.bit) and sets the result into the carry flag.

**XOR1 CY, fmem. bit**

Function: $CY \leftarrow CY \veebar$ (bit specified by operand)

Exclusive-ORs the contents of the carry flag and the contents of the data memory bit specified by bit operation addressing (fmem.bit) and sets the result into the carry flag.

12.4.7 Branch instruction

**BR addr**

Function: $PC_{10-0} \leftarrow addr$ addr = 0000H-077FH (depends upon available ROM of the selected device)

This instruction causes control to branch to the address specified by the 11-bit immediate data, addr. It is an assembler's pseudoinstruction and is automatically replaced with an appropriate instruction out of the BRCB !caddr, and BR $addr instruction when assembling.

**BR $addr**

Function: $PC \leftarrow addr$ addr = (PC-15) to (PC-1), (PC+2) to (PC+16)

This instruction causes control to branch to an address which is (–15 to -1) and (+2 to +16) with respect to the current address. It is not affected by page or block boundaries.

**BRCB !caddr**

Function: $PC_{10-0} \leftarrow caddr$ caddr = 0000H-077FH

This instruction causes control to branch to the address which is obtained by replacing the lower address which is obtained by replacing the lower 11 bits ($PC_{10-0}$) of the program counter with the 11-bit immediate data caddr ($A_{10-0}$).

12.4.8 Subroutine stack control instruction

**CALLF !faddr**

Function: $(SP-1) \leftarrow PC_{7-4}$, $(SP-2) \leftarrow PC_{3-0}$, $(SP-3) \leftarrow 0, 0, 0, 0$, $(SP-4) \leftarrow PC_{10-8}$,
$SP \leftarrow SP-4$, $PC \leftarrow A_{10-0}$, faddr = $A_{10-0}$: 000H-7FFH

This instruction saves the values of the program counter (PC return address), to the data memory locations (stack) addressed by the stack pointer (SP), decrements the SP, and causes branching to the address specified by the 11-bit immediate data faddr. The program memory area which can be called is limited to addresses 0000 through 077FH (0-1919).

**RET**

Function: $PC_{10-8} \leftarrow (SP)$, $PC_{3-0} \leftarrow (SP+2)$ $PC_{7-4} \leftarrow (SP+3)$, $SP \leftarrow SP+4$

This instruction restores the contents of the data memory (stack) locations addressed by the stack pointer (SP) into the program counter (PC), then increments the SP.

**RETS**

Function: $PC_{10-8} \leftarrow (SP), PC_{3-0} \leftarrow (SP+2)\ PC_{7-4} \leftarrow (SP+3), SP \leftarrow SP+4$
Then skip unconditionally.

This instruction restores the contents of the data memory (stack) locations addressed by the stack pointer (SP) into the program counter (PC), increments the SP value, and makes an unconditional skip.

**RETI**

Function: $PC_{10-8} \leftarrow (SP),\ PC_{3-0} \leftarrow (SP+2)\ PC_{7-4} \leftarrow (SP+3), PSW_L \leftarrow (SP+4),$
$PSW_H \leftarrow (SP+5), SP \leftarrow SP+6$

This instruction restores the contents of the data memory locations (stack) addressed by the stack pointer (SP) into the program counter (PC) and program status word (PSW), then increments the SP value. It is used to return from an interrupt service routine.

**PUSH rp**

Function: $(SP-1) \leftarrow rp_H, (SP-2) \leftarrow rp_L, SP \leftarrow SP-2$

This instruction saves the contents of the register pair rp (XA, HL) to the data memory locations (stack) addressed by the stack pointer (SP), then decrements the SP value. The higher 4 bits ($rp_H$: X, H) of the register pair are saved to the stack addressed by (SP-1), while the lower 4 bits ($rp_L$: A, L) of it is saved to the stack addressed by (SP-2).

**POP rp**

Function: $rp_L \leftarrow (SP), rp_H \leftarrow (SP+1), SP \leftarrow SP+2)$

This instruction restores the contents of the data memory locations (stack) address by the stack pointer (SP) into the register pair rp (XA, HL) then increments the SP. The contents of the (SP) are restored into the lower 4 bits of the register pair ($rp_L$: A, L) while those of the (SP+1) are restored into the higher 4 bits of the register pair ($rp_H$: X, H).

12.4.9 Interrupt control instructions

**EI**

Function: IME ← 1

This instruction sets the interrupt enable master flag to 1 to enable interrupt. Whether a specific interrupt is acknowledged or not is determined by individual interrupt enable flag.

**EI IExxx**

Function: IExxx ← 1 $xxx = N_{2-0}$

This instruction sets the interrupt flag (IExxx) to 1 to enable the corresponding interrupt request.

**DI**

Function: IME ← 0

This instruction resets the interrupt enable master flag to 0 to disable all interrupts regardless of the status of their individual interrupt enable flags.

**DI IExxx**

Function: IExxx ← 0 $xxx = N_{2-0}$

This instruction resets the interrupt flag (IExxx) to 0 to disable the corresponding interrupt request.

12.4.10 Input/output instructions

**IN A, PORTn**

Function: A ← PORTn n = $N_3$ to $N_0$: 0 to 3, 5, 6

This instruction transfers the contents of the port specified by PORTn to the A register.

Note: When the I/O mode specified, the output latch data (output mode) or pin data (input mode) is loaded to the A register.

**OUT PORTn, A**

Function: PORTn ← A n = $N_3$ to $N_0$: 2, 3, 5, 6

This instruction transfers the A register value to the output latch in the port specified by PORTn.

12.4.11 CPU control instructions

**HALT**

Function: PCC.2 ← 1

This instruction selects the HALT mode (sets bit 2 of the processor clock control register).

Note: The HALT instruction must be followed by a NOP instruction.

**STOP**

Function: PCC. 3 ← 1

This instruction sets the STOP mode (sets bit 3 of the processor clock control register).

Note: The STOP instruction must be followed by a NOP instruction.

**NOP**

Function: Causes no operation to consume an idle machine cycle.

## CMOS-DESIGN RECOMMENDATIONS

In order to maximize circuit reliability please note the general CMOS design rules.
For example.

1) Don´t leave unused pins open, except they are outputs.

2) Never exceed the max. voltage range.

3) Avoid accurance of very fast voltage spikes or transition rate on the power supply pin.

# NEC Customer Services

## NEC's Commitment to Information

Our offices throughout Europe are always at your service for comprehensive support. Here are some of the technical services we provide:

- INSECT
- Seminars
- Update service
- Hotline
- Mailing list
- University program

## INSECT

**IN**formation **S**ystem and **E**lectronic **C**a**T**alog.

This is an on-line information service. Via a telephone link you can call up the latest data on all VLSI devices available from NEC. This includes enhancements, news and the most recent application know-how.

The service is free to our customers and other interested parties. For a menu-driven guest session, you can dial in to INSECT via the international packet switching network – in Germany this is DATEX-P – using of these numbers
45 21 10 13 020/030
and responding to the request for USERNAME and PASSWORD simply with "customer".

## Seminars

No-one is more aware than NEC of the difference that a brief intensive training course can make to your mastery of advanced and often complex devices. We hold regular workshops and seminars at local NEC offices, in our Düsseldorf headquarters, or on customer premises. For information on NEC workshops and seminars, please contact your nearest office.

## Update Service

When you buy an evaluation package from NEC, you become automatically entitled to one year's free updates for both hardware and software. All updates reach you fast and reliably via a courier service. In a field where rapid changes are the norm, you can be thus be sure of working with the most up-to-date development tools.

## Hotlines

NEC's offices located throughout Europe are responsible for technical support and customer services. On the back cover of this brochure you can check which office is most convenient for you to contact. You are also welcome to contact our European headquarters in Düsseldorf directly.

## Mailing list

Our engineering staff produces frequent additions to the available technical documentation in the form of application notes, product news and technical letters. If you would like to be included on our mailing list for this documentation, please inform us.

## University Program

Many of our products, because of their complexity and dedicated application support through EB tools, are interesting subjects for graduate studies. NEC is always ready to discuss this possibility.

## EUROPEAN DISTRIBUTORS

### AUSTRIA

G. DEMEL
HANDELSGESELLSCHAFT MBH
HOFFMEISTERGASSE 8-10/1
1120 WIEN
TEL.: (222) 81 32 50 70

### BELGIUM

INTRA ELECTRONICS P.V.B.A.
BOSUIL 80, BUS 1
2100 DEURNE
TEL.: (03) 3 25 23 20
TLX.: 31102

MALCHUS ELECTRONICS B.V.B.A.
PLANTIN EN MORETUSLEI 172
2018 ANTWERPEN
TEL.: (03) 2 35 32 72
TLX.: 33 637

INNOCIRCUIT BENELUX B.V.B.A.
PLANTIN EN MORETUSLEI 172
2018 ANTWERPEN
TEL.: (03) 2 35 32 72 (B)
TEL.: (00 31) 10-4 27 77 80
TLX.: 33 637

### DENMARK

MER-EL A/S
VED KLAEDEBO 18
2970 HOERSHOLM
TEL.: (2) 57 10 00
TLX.: 37 360

### FINLAND

OY COMDAX AB
ITAELAHDENKATU 23A
PI 1
00210 HELSINKI
TEL.: (0) 67 02 77
TLX.: 57 125 876

### FRANCE

ASAP
2 AVENUE DES CHAUMES
78180 MONTIGNY LE BRETONNEUX
TÉL.: (1) 30 43 82 33
TLX.: 698 887

C.C.I.
5, RUE MARCELIN BERTHELOT
B.P. 92
92164 ANTONY
TÉL.: (1) 46 66 21 82
TLX.: 203 881

CGE COMPOSANTS
32 RUE GRANGE DAME ROSE
92360 MEUDON
TÉL.: (1) 46 30 24 25
TLX.: 632 118

PANTEC (GROUPE SCIENTECH)
29 RUE MARCEL DASSAULT
92100 BOULOGNE
TÉL.: (1) 46 20 06 74
TLX.: 633 048

### GERMANY

BIT-ELECTRONIC AG
DINGOLFINGER STRASSE 6
8000 MÜNCHEN 80
TEL.: (0 89) 41 80 07-0
TLX.: 5 212 931

GLEICHMANN + CO ELECTRONICS
GMBH
WORMSER STRASSE 34
6710 FRANKENTHAL
TEL.: (0 62 33) 2 50 56
TLX.: 4 65 270

GLYN GMBH
SCHÖNE AUSSICHT 30
6272 NIEDERNHAUSEN
TEL.: (0 61 27) 80 77
TLX.: 4 186 911

H3W ELEKTRONIK VERTRIEBS GMBH
STAHLGRUBERRING 12
8000 MÜNCHEN 82
TEL.: (0 89) 42 92 71
TLX.: 5 214 514

MICROSCAN GMBH
ÜBERSEERING 31
2000 HAMBURG 60
TEL.: (0 40) 6 32 00 30
TLX.: 2 13 288

REIN ELEKTRONIK GMBH
LÖTSCHERWEG 66
4054 NETTETAL 1
TEL.: (0 21 53) 73 31 11
TLX.: 8 54 251

SYSTEM ELEKTRONIK VERTRIEBS
GMBH
HEESFELD 4
3300 BRAUNSCHWEIG
TEL.: (05 31) 31 40 95
TLX.: 9 52 351

ULTRATRONIK GMBH
GEWERBESTRASSE 4
8036 HERRSCHING
TEL.: (0 81 52) 37 09-0
TLX.: 5 26 459

UNIELECTRONIC VERTRIEBS GMBH
IM GEFIERTH 11 A
6072 DREIEICH 1 B. FRANKFURT
TEL.: (0 61 03) 3 51 75
TLX.: 4 11 213

### ITALY

ADELSY S.R.L.
VIA DEL FONDITORE, 5
LOCALITA ROVERI
40127 BOLOGNA
TEL.: (51) 532119
TLX.: 510 226

CLAITRON S.P.A.
VIA GALLARATE, 211
20151 MILANO
TEL.: (1) 33404000
TLX.: 313 843

DIS.EL. S.P.A.
VIA ALA DI STURA 71
10148 TORINO
TEL.: (011) 2201522
TLX.: 215 118

MELCHIONI S.P.A.
VIA COLLETTA, 37
20135 MILANO
TEL.: (01) 57941
TLX.: 315 293

### NETHERLANDS

MALCHUS B.V.
FOKKERSTRAAT 511
3125 BD SCHIEDAM
TEL.: (10) 4 27 77 77
TLX.: 21 598

INNOCIRCUIT BENELUX B.V.
POSTBUS 48
3100 AA SCHIEDAM
TEL.: (10) 4 27 77 80
TLX.: 21 598

INTRA ELECTRONICS B.V.
GULBERG 33
5674 TE NUENEN
TEL.: (40) 83 80 09
TLX.: 59 418

### NORWAY

JAKOB HATTELAND ELECTRONIC A/S
P.B. 25
5578 NEDRE VATS
TEL.: (47) 63 111
TLX.: 428 50

### PORTUGAL

AMPEREL S.A.
AV. FONTES PEREIRA DE MELO 47, 4D
1000 LISBOA
TEL.: (1) 53 26 98
TLX.: 18588

### SPAIN

COMELTA SA
EMILIO MUNOZ 41. NAVE 1-1-2
28037 MADRID
TEL.: (1) 7 54 30 77
TLX.: 42 007

VENCO
CALLE GALILEO 249
08028 BARCELONA
TEL.: (3) 330 97 51
TLX.: 98 266

### SWEDEN

NAX AB KOMPONENTBOLAGET
BOX 4115
17104 SOLNA
TEL.: (8) 98 51 40
TLX.: 17912

NORDQVIST & BERG
BOX 1458
17128 SOLNA
TEL.: (8) 7 64 67 10
TLX.: 10407

TH'S ELEKTRONIK
BOX 3027
16303 SPAANGA
TEL.: (8) 36 29 70
TLX.: 11145

### SWITZERLAND

MEMOTEC AG
GASWERKSTRASSE 32
4901 LANGENTHAL
TEL.: (63) 28 11 22
TLX.: 9 82 550

### UNITED KINGDOM

2001 ELECTRONIC COMPONENTS
WOOLMERS WAY
STEVENAGE
HERTS
SG1 3AJ
TEL.: (4 38) 74 20 01
TLX.: 827 701

ANZAC COMPONENTS LIMITED
822, YEOVIL ROAD
SLOUGH TRADING ESTATE
SLOUGH
BERKSHIRE
SL1 4JA
TEL.: (62 86) 44 11
TLX.: 847 949

CELDIS LIMITED
37, LOVEROCK ROAD
READING
BERKSHIRE
RG3 1EL
TEL.: (7 34) 58 51 71
TLX.: 848 370

IMPULSE ELECTRONICS LIMITED
HAMMOND HOUSE
CATERHAM
SURREY
CR3 6XG
TEL.: (8 83) 4 70 11
TLX.: 291 496

S.T.C. MULTICOMPONENTS LIMITED
EDINBURGH WAY
HARLOW
ESSEX
CM20 2DF
TEL.: (2 79) 44 11 44
TLX.: 818 763

V.S.I. ELECTRONICS (UK) LIMITED
ROYDONBURY INDUSTRIAL PARK
HORSECROFT ROAD
HARLOW
ESSEX
CM19 5BY
TEL.: (2 79) 2 96 66
TLX.: 81 387