CMOS 8-BIT MICROCONTROLLER

TMP87C841N, TMP87CC41N, TMP87CH41N, TMP87CK41N, TMP87CM41N
TMP87C841F, TMP87CC41F, TMP87CH41F, TMP87CK41F, TMP87CM41F
TMP87C841U, TMP87CC41U, TMP87CH41U, TMP87CK41U, TMP87CM41U

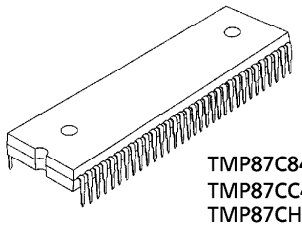
The 87C841/C41/H41/K41/M41 are the high speed and high performance 8-bit single chip microcomputers. These MCU contain CPU core, ROM, RAM, input/output ports, an A/D converter, six multi-function timer/counters, two serial interfaces, and two clock generators on a chip. The 87C841/CC41/CH41/CK41/CM41 provide high current output capability for LED direct drive.

| PART No. | ROM | RAM | PACKAGE | OTP MCU |
|---|---|---|---|---|
| TMP87C841N | 8 K × 8-bit | 256 × 8-bit | SDIP64-P-750-1.78 | TMP87PM41N |
| TMP87C841F | | | QFP64-P-1420-1.00A | TMP87PM41F |
| TMP87C841U | | | LQFP64-P-1010-0.50 | TMP87PM41U |
| TMP87CC41N | 12 K × 8-bit | 512 × 8-bit | SDIP64-P-750-1.78 | TMP87PM41N |
| TMP87CC41F | | | QFP64-P-1420-1.00A | TMP87PM41F |
| TMP87CC41U | | | LQFP64-P-1010-0.50 | TMP87PM41U |
| TMP87CH41N | 16 K × 8-bit | | SDIP64-P-750-1.78 | TMP87PM41N |
| TMP87CH41F | | | QFP64-P-1420-1.00A | TMP87PM41F |
| TMP87CH41U | | | LQFP64-P-1010-0.50 | TMP87PM41U |
| TMP87CK41N | 24 K × 8-bit | 1K × 8-bit | SDIP64-P-750-1.78 | TMP87PM41N |
| TMP87CK41F | | | QFP64-P-1420-1.00A | TMP87PM41F |
| TMP87CK41U | | | LQFP64-P-1010-0.50 | TMP87PM41U |
| TMP87CM41N | 32 K × 8-bit | | SDIP64-P-750-1.78 | TMP87PM41N |
| TMP87CM41F | | | QFP64-P-1420-1.00A | TMP87PM41F |
| TMP87CM41U | | | LQFP64-P-1010-0.50 | TMP87PM41U |

## FEATURES

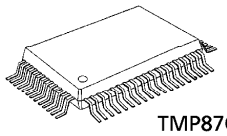
- 8-bit single chip microcomputer TLCS-870 Series
- Instruction execution time : 0.5 μs (at 8 MHz), 122 μs (at 32.768 kHz)
- 412 basic instructions
  - Multiplication and Division (8 bits × 8 bits , 16 bits ÷ 8 bits)
  - Bit manipulations (Set/Clear/Complement/Move/Test/Exclusive or)
  - 16-bit data operations
  - 1-byte jump/subroutine-call (Short relative jump / Vector call)
- 15 interrupt sources (External : 6, Internal : 9)
  - All sources have independent latches each, and nested interrupt control is available.
  - 4 edge-selectable external interrupts with noise reject
  - High-speed task switching by register bank changeover
- Input/Output ports (56 pins)
  - High current output : 8 pins (typ. 20 mA)
- Two 16-bit Timer/Counters
  - Timer, Event counter, Programmable pulse generator output, Pulse width measurement, External trigger timer, Window modes
- Two 8-bit Timer/Counters
  - Timer, Event counter, Capture (Pulse width/duty measurement), PWM output, Programmable divider output modes
- Time Base Timer (Interrupt frequency : 1 Hz to 16 kHz)
- Divider output function (frequency : 1 kHz to 8 kHz)
- Watchdog Timer
- High-speed PWM output (2 channel)
  - Cycle : 32 kHz, 64 kHz, 128 kHz.
  - Resolution : 8 bits, 7 bits, 6 bits
- Two 8-bit Serial Interfaces
  - Each 8 bytes transmit/receive data buffer
  - Internal/external serial clock, and 4/8-bit mode
- **10-bit** successive approximate type **A/D converter**
  - 16 analog inputs
  - Conversion time: 23 μs at 8 MHz
- Dual clock operation
- Five Power saving operating modes
  - STOP mode : Oscillation stops. Battery/Capacitor back-up. Port output hold/high-impedance.
  - SLOW mode: Low power consumption operation using low-frequency clock (32.8 kHz).
  - IDLE1 mode : CPU stops, and Peripherals operate using high-frequency clock. Release by interrupts.
  - IDLE2 mode : CPU stops, and Peripherals operate using high and low frequency clock. Release by interrupts.
  - SLEEP mode : CPU stops, and Peripherals operate using low-frequency clock. Release by interrupts.
- **Wide operating voltage** : 2.7 to 5.5 V at 4.19 MHz / 32.768 kHz, 4.5 to 5.5 V at 8 MHz / 32.768 kHz.
- Emulation Pod : BM87CM41N0A

SDIP64-P-750-1.78



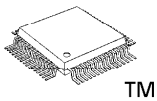
TMP87C841N
TMP87CC41N
TMP87CH41N
TMP87CK41N
TMP87CM41N
TMP87PM41N

QFP64-P-1420-1.00A



TMP87C841F
TMP87CC41F
TMP87CH41F
TMP87CK41F
TMP87CM41F
TMP87PM41F

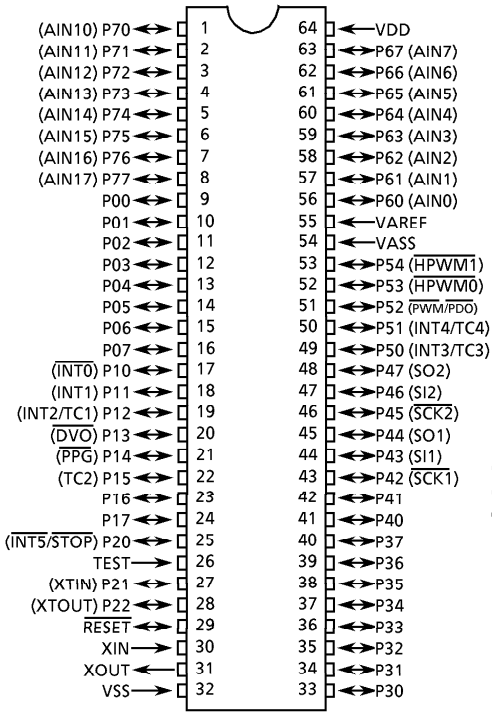
LQFP64-P-1010-0.50



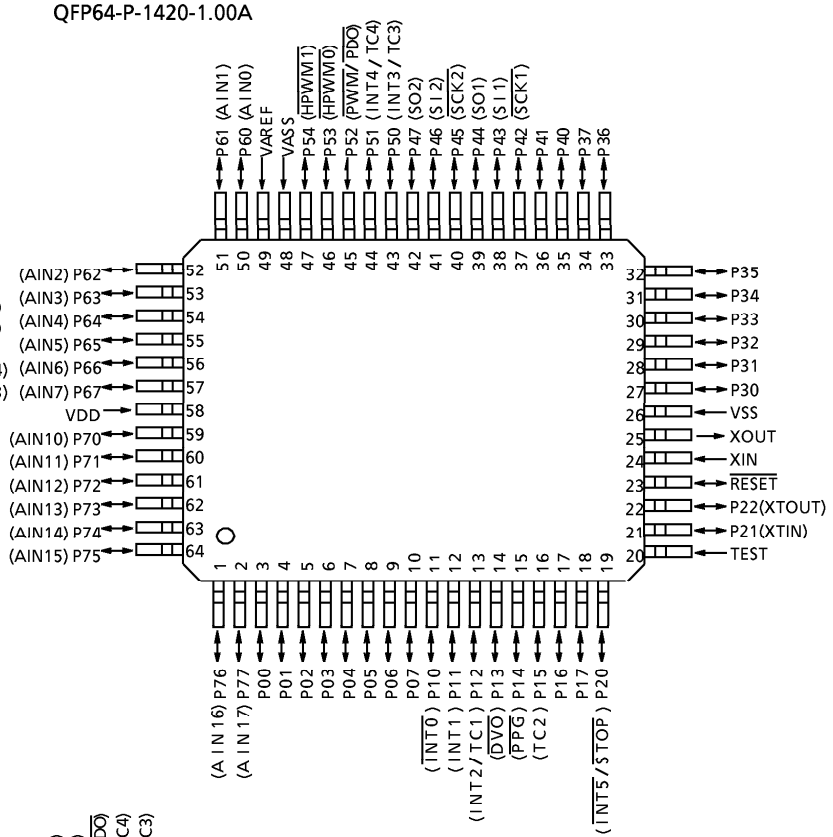
TMP87C841U
TMP87CC41U
TMP87CH41U
TMP87CK41U
TMP87CM41U
TMP87PM41U

## PIN ASSIGNMENTS (TOP VIEW)

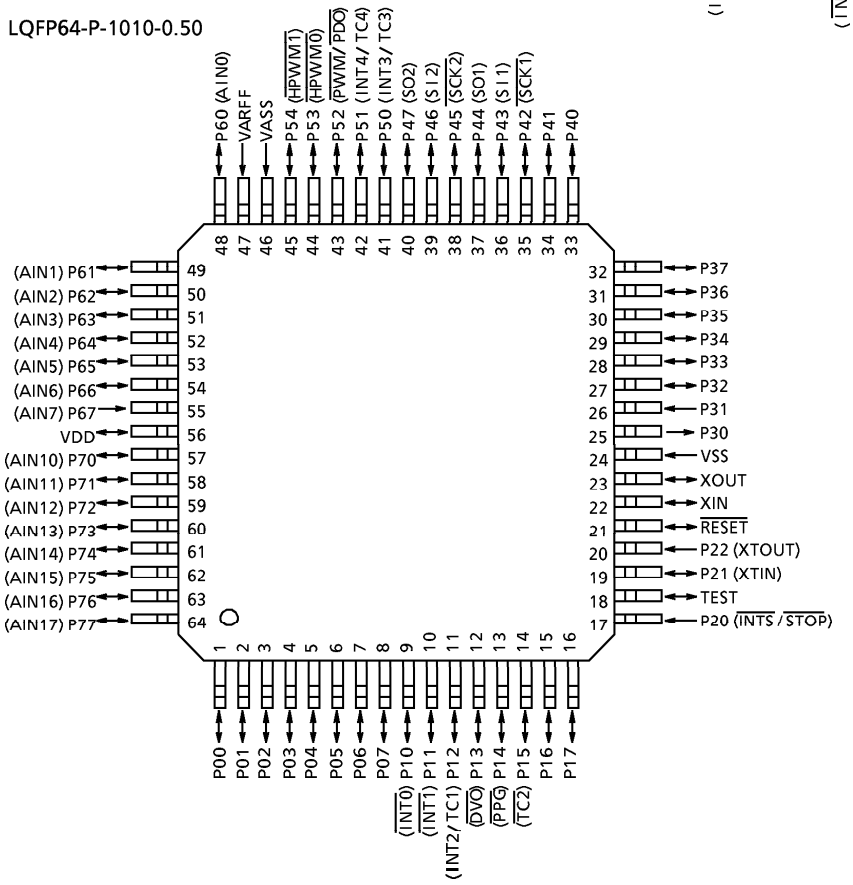
SDIP64-P-750-1.78



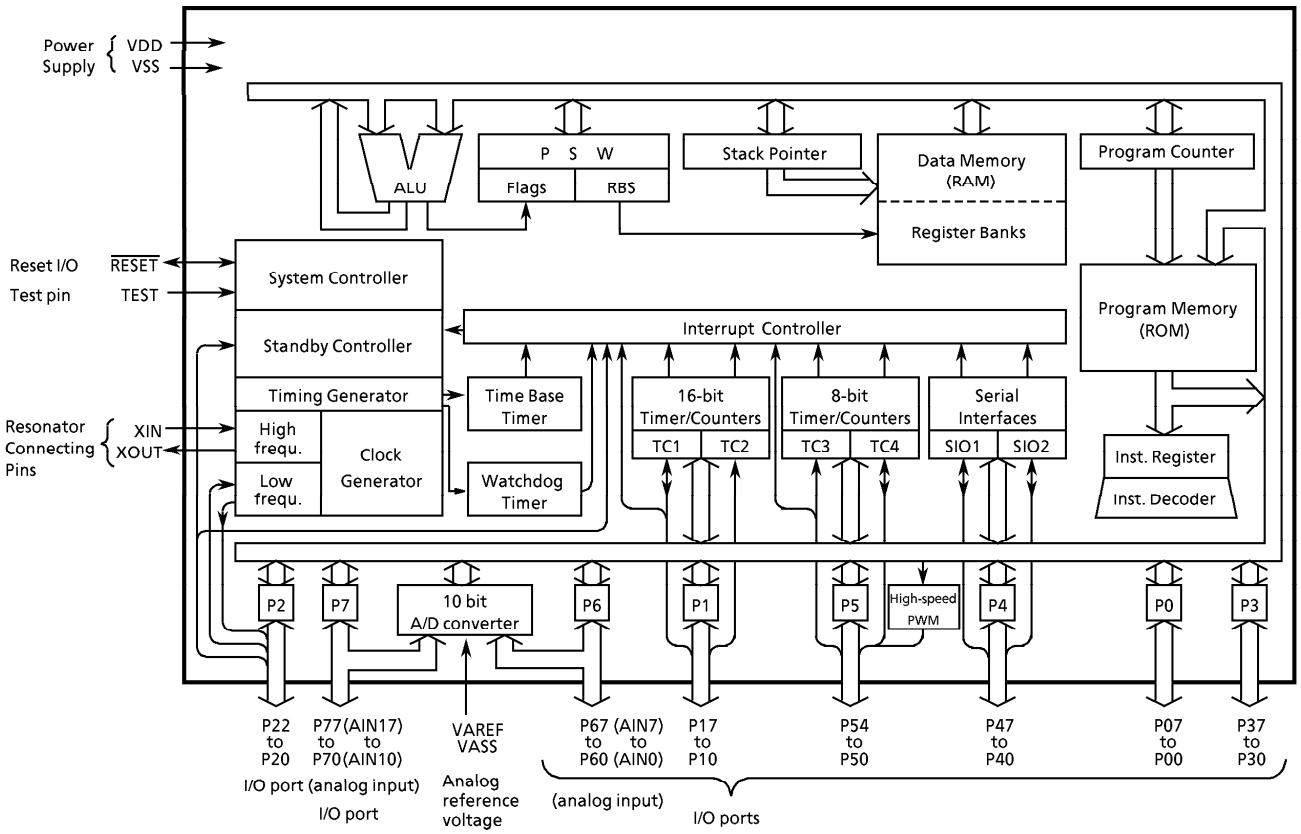

QFP64-P-1420-1.00A




LQFP64-P-1010-0.50



## BLOCK DIAGRAM



## PIN FUNCTION

| PIN NAME | Input / Output | FUNCTION | |
|---|---|---|---|
| P07 to P00 | I/O | Two 8-bit programmable input/output ports (tri-state).<br>Each bit of these ports can be individually configured as an input or an output under software control.<br>During reset, all bits are configured as inputs.<br>When used as a divider output or a PPG output, the latch must be set to "1". | |
| P17, P16 | I/O | | |
| P15 (TC2) | I/O (Input) | | Timer/Counter 2 input |
| P14 ($\overline{PPG}$) | I/O (Output) | | Programmable pulse generator output |
| P13 ($\overline{DVO}$) | | | Divider output |
| P12 (INT2 / TC1) | I/O (Input) | | External interrupt input 2 or Timer/Counter 1 input |
| P11 (INT1) | | | External interrupt input 1 |
| P10 ($\overline{INT0}$) | | | External interrupt input 0 |
| P22 (XTOUT) | I/O (Output) | 3-bit input/output port with latch.<br>When used as an input port, the latch must be set to "1". | Resonator connecting pins (32.768 kHz). For inputting external clock, XTIN is used and XTOUT is opened. |
| P21 (XTIN) | I/O (Input) | | |
| P20 ($\overline{INT5}$/$\overline{STOP}$) | | | External interrupt input 5 or STOP mode release signal input |
| P37 to P30 | I/O | 8-bit input/output port (high current output) with latch.<br>When used as an input port, the latch must be set to "1". | |
| P47 (SO2) | I/O (Output) | 8-bit input/output port with latch.<br>When used as an input port or a SIO input/output, the latch must be set to "1". | SIO2 serial data output |
| P46 (SI2) | I/O (Input) | | SIO2 serial data input |
| P45 ($\overline{SCK2}$) | I/O (I/O) | | SIO2 serial clock input/output |
| P44 (SO1) | I/O (Output) | | SIO1 serial data output |
| P43 (SI1) | I/O (Input) | | SIO1 serial data input |
| P42 ($\overline{SCK1}$) | I/O (I/O) | | SIO1 serial clock input/output |
| P41, P40 | I/O | | |
| P54 ($\overline{HPWM1}$) | I/O (Output) | 5-bit input/output port with latch.<br>When used as an input port, an external interrupt input, or a $\overline{PWM}$/$\overline{PDO}$, $\overline{HPWM0}$, $\overline{HPWM1}$ output, the latch must be set to "1". | 8-bit High-speed PWM output |
| P53 ($\overline{HPWM0}$) | I/O (Output) | | |
| P52 ($\overline{PWM}$/$\overline{PDO}$) | I/O (Input) | | 8-bit PWM output or 8-bit programmable divider output |
| P51 (INT4/TC4) | I/O (Input) | | External interrupt input 4 or Timer/Counter 4 input |
| P50 (INT3/TC3) | | | External interrupt input 3 or Timer/Counter 3 input |
| P67 (AIN7) to P60 (AIN0)<br>P77 (AIN17) to P70 (AIN10) | I/O (Input) | 8-bit programmable input/output port (tri-state). Each bit of the port can be individually configured as an input or an output under software control. (When used an analog input, the latch must be set to P6CR and P7CR analog input.) | A/D converter analog inputs |
| XIN, XOUT | Input, Output | Resonator connecting pins for high-frequency clock.<br>For inputting external clock, XIN is used and XOUT is opened. | |
| $\overline{RESET}$ | I/O | Reset signal input or watchdog timer output/address-trap-reset output/system-clock-reset output. | |
| TEST | Input | Test pin for out-going test. Be tied to low. | |
| VDD, VSS | Power Supply | + 5 V, 0 V (GND) | |
| VAREF, VASS | | Analog reference voltage inputs (High, Low) | |

# OPERATIONAL DESCRIPTION

## 1. CPU CORE FUNCTIONS

The CPU core consists of a CPU, a system clock controller, an interrupt controller, and a watchdog timer. This section provides a description of the CPU core, the program memory (ROM), the data memory (RAM), and the reset circuit.

### 1.1 Memory Address Map

The TLCS-870 Series is capable of addressing 64K bytes of memory. Figure 1-1 shows the memory address maps of the 87C841/CC41/CH41/CK41/CM41. In the TLCS-870 Series, the memory is organized 4 address spaces (ROM, RAM, SFR, and DBR). It uses a memory mapped I/O system, and all I/O registers are mapped in the SFR/DBR address spaces. There are 16 banks of general-purpose registers. The register banks are also assigned to the first 128 bytes of the RAM address space.

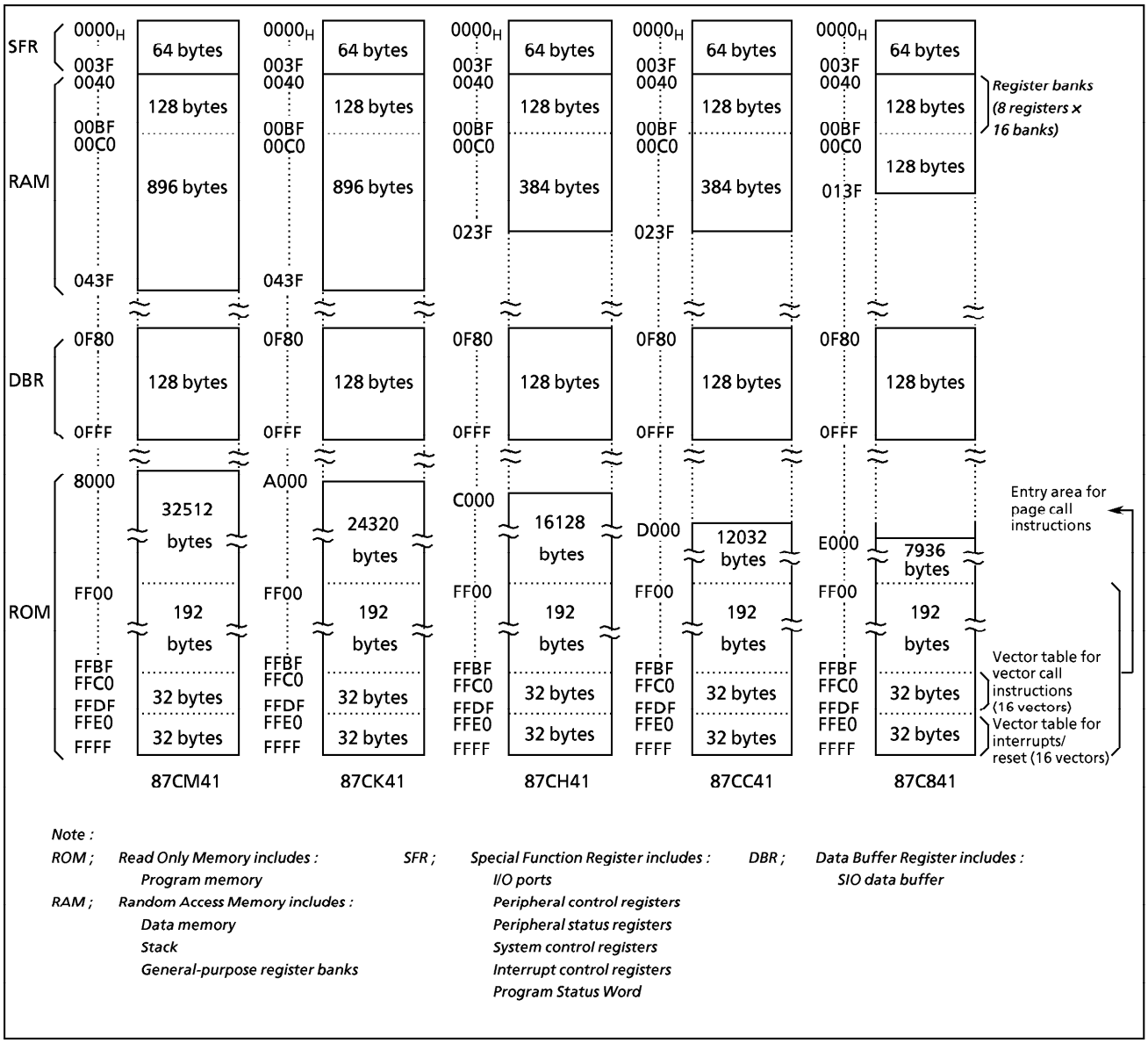



Figure 1-1. Memory Address Maps

## 1.2 Program Memory (ROM)

The 87C841 has an 8K × 8-bit (addresses $E000_H$ to $FFFF_H$), the 87CC41 has a 12K × 8-bit (addresses $D000_H$ to $FFFF_H$), the 87CH41 has a 16K × 8-bit (addresses $C000_H$ to $FFFF_H$), and the 87CK41 has a 24K × 8-bit (address $A000_H$ to $FFFF_H$), and the 87CM41 has a 32K-8-bit (address $8000_H$ to $FFFF_H$) of program memory (mask programmed ROM).
Addresses $FF00_H$ to $FFFF_H$ in the program memory can also be used for special purposes.

(1) **Interrupt / Reset** vector table (addresses $FFE0_H$ to $FFFF_H$)
This table consists of a reset vector and 15 interrupt vectors (2 bytes/vector). These vectors store a reset start address and interrupt service routine entry addresses.

(2) Vector table for **vector call** instructions (addresses $FFC0_H$ to $FFDF_H$)
This table stores call vectors (subroutine entry address, 2 bytes/vector) for the vector call instructions [CALLV n]. There are 16 vectors. The CALLV instruction increases memory efficiency when utilized for frequently used subroutine calls (called from 3 or more locations).

(3) Entry area (addresses $FF00_H$ to $FFFF_H$) for **page call** instructions
This is the subroutine entry address area for the page call instructions [CALLP n]. Addresses $FF00_H$ to $FFBF_H$ are normally used because address $FFC0_H$ to $FFFF_H$ are used for the vector tables.

Programs and fixed data are stored in the program memory. The instruction to be executed next is read from the address indicated by the current contents of the program counter (PC). There are relative jump and absolute jump instructions. The concepts of page or bank boundaries are not used in the program memory concerning any jump instruction.

Example: The relationship between the jump instructions and the PC.

① 5-bit PC-relative jump [JRS cc, $ + 2 + d]
E8C4H: JRS T, $ + 2 + 08H
When JF = 1, the jump is made to $E8CE_H$, which is $08_H$ added to the contents of the PC. (The PC contains the address of the instruction being executed + 2; therefore, in this case, the PC contents are $E8C4_H + 2 = E8C6_H$.)

② 8-bit PC-relative jump [JR cc, $ + 2 + d]
E8C4H : JR Z, $ + 2 + 80H
When ZF = 1, the jump is made to $E846_H$, which is $FF80_H$ ( − 128) added to the current contents of the PC.

③ 16-bit absolute jump [JP a]
E8C4H : JP 0C235H
An unconditional jump is made to address $C235_H$. The absolute jump instruction can jump anywhere within the entire 64K-bytes space.

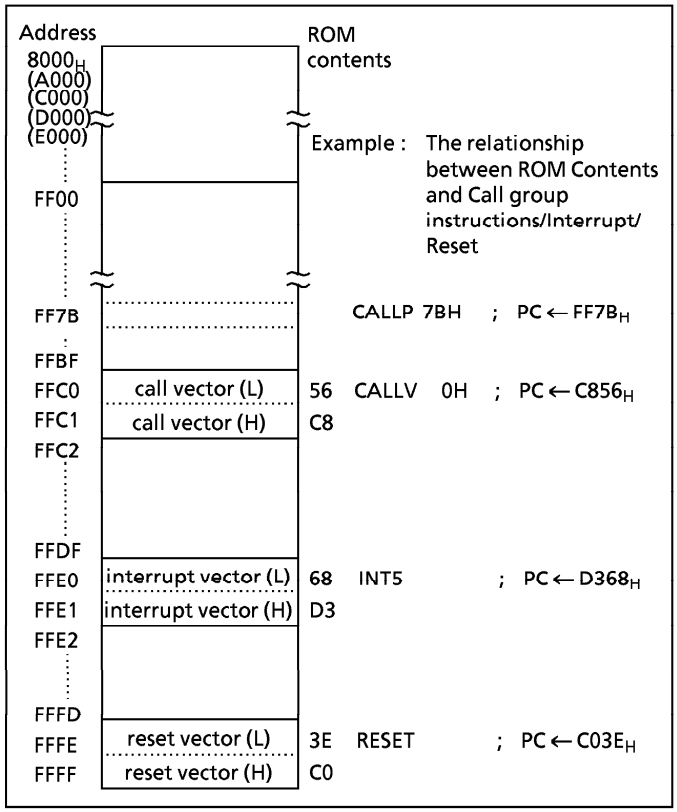



Figure 1-2. Program Memory Map

In the TLCS-870 Series, the same instruction used to access the data memory (e.g. [LD A, (HL)] ) is also used to read out fixed data (ROM data) stored in the program memory. The register-offset PC-relative addressing (PC + A) instructions can also be used, and the code conversion, table look-up and n-way multiple jump processing can easily be programmed.

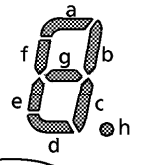
Example 1 : Loads the ROM contents at the address specified by the HL register pair contents into the accumulator (87CH41 : HL≧C000$_H$):

```
LD      A, (HL)          ; A←ROM (HL)
```

Example 2 : Converts BCD to 7-segment code (common anode LED). When A = 05$_H$, 92$_H$ is output to port P3 after executing the following program:

```
        ADD     A, TABLE – $ – 4     ; P3 ←ROM (TABLE + A)
        LD      (P3), (PC + A)
        JRS     T, SNEXT
TABLE : DB      0C0H, 0F9H, 0A4H, 0B0H, 99H, 92H, 82H, 0D8H, 80H, 98H
SNEXT :
```

*Notes : "$" is a header address of ADD instruction.*
*DB is a byte data difinition instruction.*




Example 3 : N-way multiple jump in accordance with the contents of accumulator (0≦A≦3):

```
SHLC    A                ; if A = 00H then PC←C234H
JP      (PC + A)           if A = 01H then PC←C378H
                           if A = 02H then PC←DA37H
                           if A = 03H then PC←E1B0H
DW      0C234H, 0C378H, 0DA37H, 0E1B0H
```

*Note : DW is a word data definition instruction.*

```
SHLC A
JP (PC + A)
34
C2
78
C3
37
DA
B0
E1
```

## 1.3 Program Counter (PC)

The program counter (PC) is a 16-bit register which indicates the program memory address where the instruction to be executed next is stored. After reset, the user defined reset vector stored in the vector table (addresses FFFF$_H$ and FFFE$_H$) is loaded into the PC ; therefore, program execution is possible from any desired address. For example, when C0$_H$ and 3E$_H$ are stored at addresses FFFF$_H$ and FFFE$_H$, respectively, the execution starts from address C03E$_H$ after reset.

The TLCS-870 Series utilizes pipelined processing (instruction pre-fetch); therefore, the PC always indicates 2 addresses in advance. For example, while a 1-byte instruction stored at address C123$_H$ is being executed, the PC contains C125$_H$.

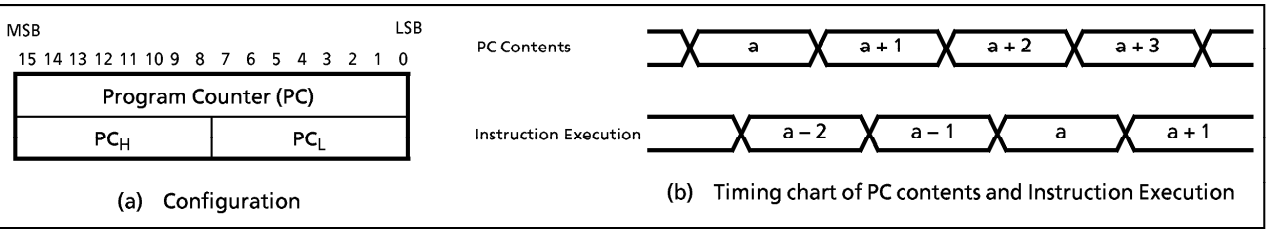



Figure 1-3. Program Counter

## 1.4 Data Memory (RAM)

The 87C841 has a 256 × 8-bit (addresses $0040_H$ to $013F_H$), the 87CC41/CH41 have a 512 × 8-bit (addresses $0040_H$ to $023F_H$), and the 87CK41/CM41 have a 1K × 8-bit (address $0040_H$ to $043F_H$) of data memory (static RAM). Figure 1-4 shows the data memory map.

Addresses $0000_H$ to $00FF_H$ are used as a direct addressing area to enhance instructions which utilize this addressing mode; therefore, addresses $0040_H$ to $00FF_H$ in the data memory can also be used for user flags or user counters. General-purpose register banks (8 registers × 16 banks) are also assigned to the 128 bytes of addresses $0040_H$ to $00BF_H$. Access as data memory is still possible even when being used for registers. For example, when the contents of the data memory at address $0040_H$ is read out, the contents of the accumulator in the bank 0 are also read out. The stack can be located anywhere within the data memory except the register bank area. The stack depth is limited only by the free data memory size. For more details on the stack, see section "1.7 Stack and Stack Pointer".

With the 87C841/CC41/CH41/CK41/CM41, programs in data memory cannot be executed. If the program counter indicates a data memory address (addresses $0040_H$ to $013F_H$ for 87C841, addresses $0040_H$ to $023F_H$ for 87CC41/CH41, addresses $0040_H$ to $043F_H$ for 87CK41/CM41) , an address-trap-reset is generated due to bus error. (Output from the $\overline{RESET}$ pin goes low. )

Example 1 : If bit 2 at data memory address $00C0_H$ is "1", $00_H$ is written to data memory at address $00E3_H$; otherwise, $FF_H$ is written to the data memory at address $00E3_H$.

```
        TEST    (00C0H).2       ;  if (00C0H)2 = 0 then jump
        JRS     T,SZERO
        CLR     (00E3H)         ;  (00E3H) ← 00H
        JRS     T,SNEXT
SZERO : LD      (00E3H), 0FFH   ;  (00E3H) ← FFH
SNEXT :
```

Example 2 : Increments the contents of data memory at address $00F5_H$, and clears to $00_H$ when $10_H$ is exceeded.

```
        INC     (00F5H)         ;  (00F5H) ← (00F5H) + 1
        AND     (00F5H), 0FH    ;  (00F5H) ← (00F5H)∧0FH
```

The data memory contents become unstable when the power supply is turned on; therefore, the data memory should be initialized by an initialization routine. Note that the general-purpuse registers are mapped in the RAM ; therefore, *do not clear RAM at the current bank addresses.*

Example 1 : Clears RAM to "$00_H$" except the bank 0 (87C841)

```
        LD      HL, 0048H       ;  Sets start address to HL register pair
        LD      WA, 0F700H      ;  Sets initial data (00H) to A register
                                ;  Sets number of byte to W register pair
SRAMCLR : LD    (HL+), A
        DEC     W
        JRS     F, SRAMCLR
```

Example 2 : Clears RAM to "$00_H$" except the bank 0 (87CC41/CH41)

```
        LD      HL, 0048H       ;  Sets start address to HL register pair
        LD      A, H            ;  Sets initial data (00H) to A register
        LD      BC, 01F7H       ;  Sets number of byte to BC register pair
SRAMCLR : LD    (HL+), A
        DEC     BC
        JRS     F, SRAMCLR
```

Example 3 : Clears RAM to "$00_H$" except the bank 0 (87CK41/CM41, 87PM41)

```
        LD      HL, 0048H       ; Sets start address to HL register pair
        LD      A, H            ; Sets initial data (00H) to A register
        LD      BC, 03F7H       ; Sets number of byte to BC register pair
SRAMCLR : LD    (HL+), A
        DEC     BC
        JRS     F, SRAMCLR
```

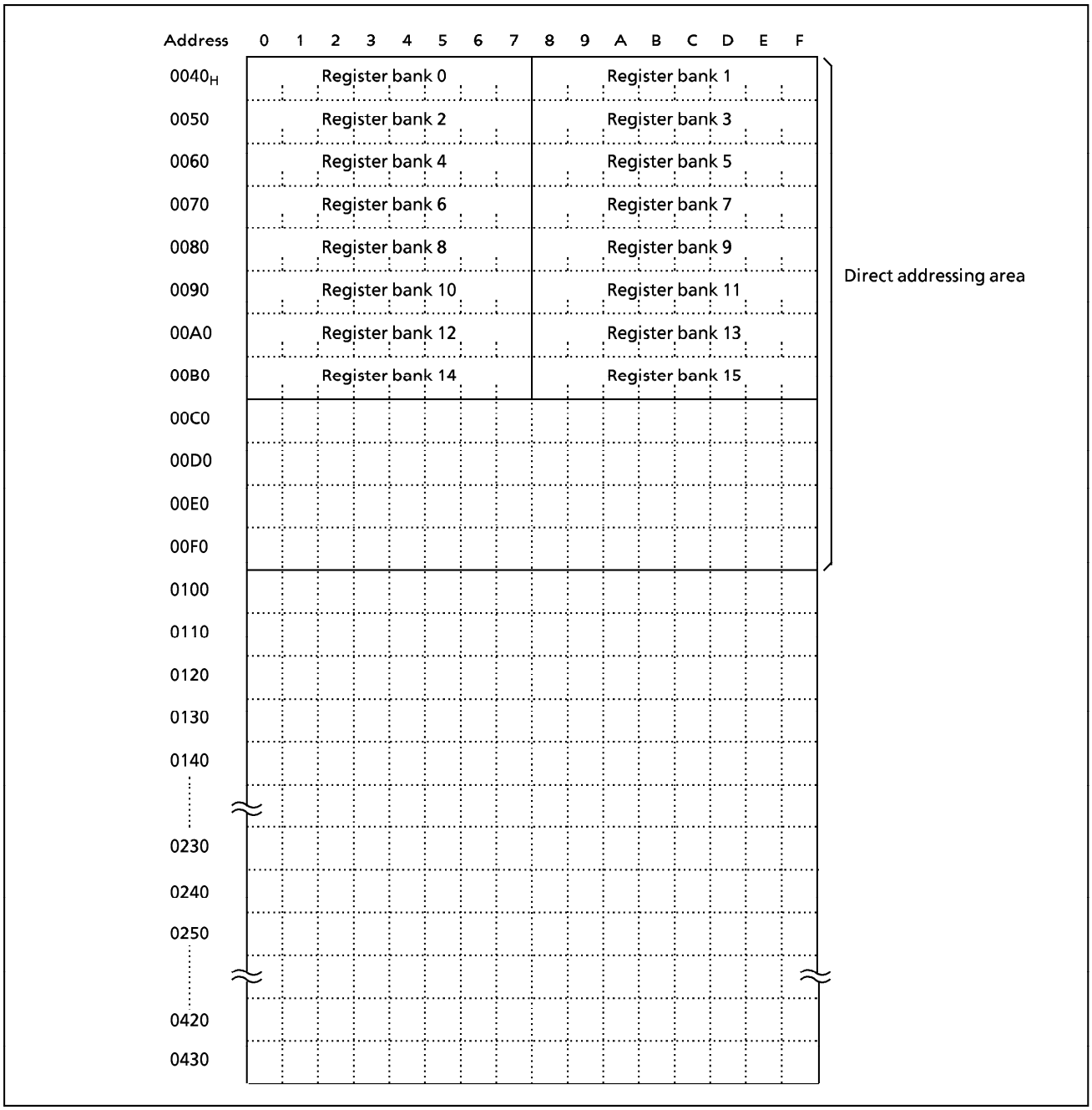



Figure 1-4. Data Memory Map

## 1.5 General-purpose Register Banks

General-purpose registers are mapped into addresses $0040_H$ to $00BF_H$ in the data memory as shown in Figure 1-4. There are 16 register banks, and each bank contains eight 8-bit registers W, A, B, C, D, E, H, and L. Figure 1-5 shows the general-purpose register bank configuration.

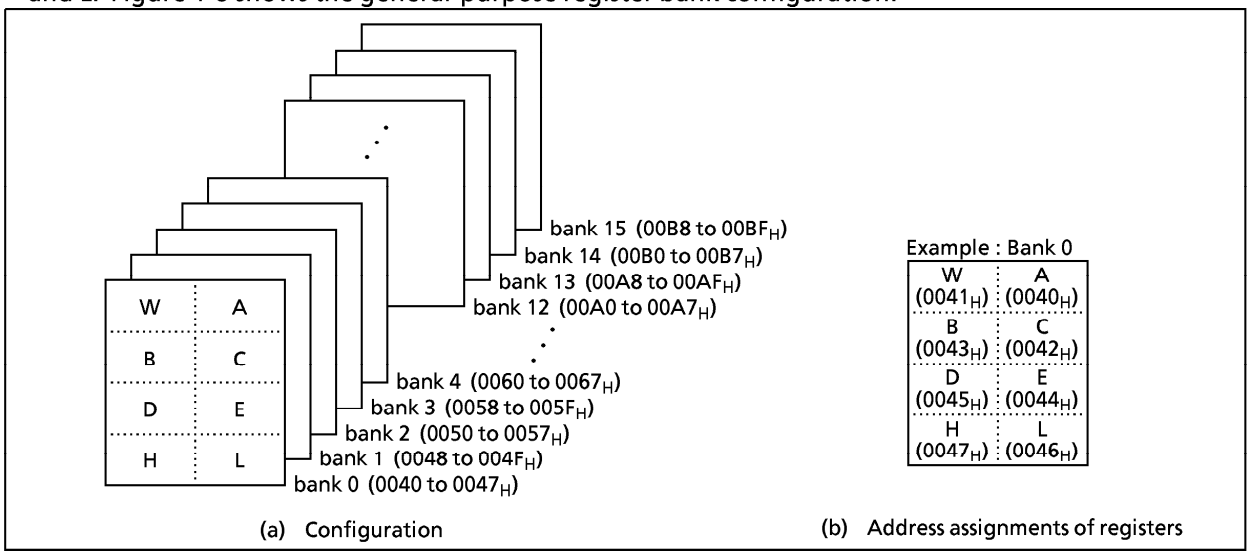



Figure 1-5. General-purpose Register Banks

In addition to access in 8-bit units, the registers can also be accessed in 16-bit units as the register pairs WA, BC, DE, and HL. Besides its function as a general-purpose register, the register also has the following functions:

(1) **A, WA**

The A register functions as an 8-bit accumulator and WA the register pair functions as a 16-bit accumulator (W is high byte and A is low byte). Registers other than A can also be used as accumulators for 8-bit operations.

```
Examples :  ①   ADD  A, B          ; Adds B contents to A contents and stores the result into A.
            ②   SUB  WA, 1234H     ; Subtracts 1234H from WA contents and stores the result into WA.
            ③   SUB  E, A          ; Subtracts A contents from E contents, and stores the result into E.
```

(2) **HL, DE**

The HL and DE specify a memory address. The HL register pair functions as data pointer (HL) /index register (HL + d) /base register (HL + C), and the DE register pair function as a data pointer (DE). The HL also has an auto-post- increment and auto-pre-decrement functions. This function simplifies multiple digit data processing, software LIFO (last-in first-out) processing, etc.

```
Example 1 :  ①   LD   A, (HL)       ; Loads the memory contents at the address specified by HL into A.
             ②   LD   A, (HL + 52H) ; Loads the memory contents at the address specified by the value
                                      obtained by adding 52H to HL contents into A.
             ③   LD   A, (HL + C)   ; Loads the memory contents at the address specified by the value
                                      obtained by adding the register C contents to HL contents into A.
             ④   LD   A, (HL + )    ; Loads the memory contents at the address specified by HL into A.
                                      Then increments HL.
             ⑤   LD   A, ( – HL)    ; Decrements HL. Then loads the memory contents at the address
                                      specified by new HL into A.
```

The TLCS-870 Series can transfer data directly memory to memory, and operate directly between memory data and memory data. This facilitates the programming of block processing.

Example 2 : Block transfer

```
        LD      B, n – 1        ; Sets (number of bytes to transfer) – 1 to B
        LD      HL, DSTA        ; Sets destination address to HL
        LD      DE, SRCA        ; Sets source address to DE
SLOOP : LD      (HL), (DE)      ; (HL) ← (DE)
        INC     HL              ; HL ← HL + 1
        INC     DE              ; DE ← DE + 1
        DEC     B               ; B ← B – 1
        JRS     F, SLOOP        ; if B ≧ 0 then loop
```

(3) **B, C, BC**

Registers B and C can be used as 8-bit buffers or counters, and the BC register pair can be used as a 16-bit buffer or counter. The C register functions as an offset register for register-offset index addressing (refer to example 1 ③ above) and as a divisor register for the division instruction [DIV gg, C].

Example 1 : Repeat processing

```
        LD      B, n            ; Sets n as the number of repetitions to B
SREPEAT : processing            ;     (n + 1 times processing)
        DEC     B
        JRS     F, SREPEAT
```

Example 2 : Unsigned integer division (16-bit ÷ 8-bit)

```
        DIV     WA, C           ; Divides the WA contents by the C contents, places the
                                ;   quotient in A and the remainder in W.
```

The general-purpose register banks are selected by the 4-bit register bank selector (RBS). During reset, the RBS is initialized to "0". The bank selected by the RBS is called the current bank.
Together with the flag, the RBS is assigned to address $003F_H$ in the SFR as the program status word (PSW). There are 3 instructions [LD RBS, n], [PUSH PSW], [POP PSW] to access the PSW. The PSW can be also operated by the memory access instruction.

Example 1 : Incrementing the RBS

```
        INC     (003FH)         ; RBS ← RBS + 1
```

Example 2 : Reading the RBS

```
        LD      A, (003FH)      ; A ← PSW (A3-0 ← RBS, A7-4 ← Flags)
```

Highly efficient programming and high-speed task switching are possible by using bank changeover to save registers during interrupt and to transfer parameters during subroutine processing.
During interrupt, the PSW is automatically saved onto the stack. The bank used before the interrupt was accepted is restored automatically by executing an interrupt return instruction [RETI]/[RETN] ; therefore, there is no need for the RBS save/restore software processing.
The TLCS-870 Series supports a maximum of 15 interrupt sources. One bank is assigned to the main program, and one bank can be assigned to each source. Also, to increase the efficiency of data memory usage, assign the same bank to interrupt sources which are not nested.

Example: Saving /restoring registers during interrupt task using bank changeover.

```
PINT1 : LD      RBS, n          ; RBS ← n (Bank changeover)
        Interrupt processing
        RETI                    ; Maskable interrupt return (Bank restoring)
```

## 1.6 Program Status Word (PSW)

The program status word (PSW) consists of a register bank selector (RBS) and four flags, and the PSW is assigned to address $003F_H$ in the SFR.

The RBS can be read and written using the memory access instruction (e. g. [LD A, (003FH)], [LD (003FH), A], however the flags can only be read. When writing to the PSW, the change specified by the instruction is made without writing data to the flags. For example, when the instruction [LD (003FH), 05H] is executed, "5" is written to the RBS and the JF is set to "1", but the other flags are not affected.

[PUSH PSW] and [POP PSW] are the PSW access instructions.

### 1.6.1 Register Bank Selector (RBS)

The register bank selector (RBS) is a 4-bit register used to select general-purpose register banks. For example, when RBS = 2, bank 2 is currently selected. During reset, the RBS is initialized to "0".

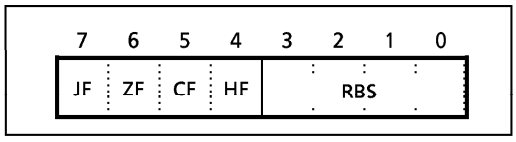



Figure 1-6. PSW (Flags, RBS) Configuration

### 1.6.2 Flags

The flags are configured with the upper 4 bits : a zero flag, a carry flag, a half carry flag and a jump status flag. The flags are set or cleared under conditions specified by the instruction. These flags except the half carry flag are used as jump condition "cc" for conditional jump instructions [JR cc, $ + 2 + d]/[JRS cc, $ + 2 + d]. After reset, the jump status flag is initialized to "1", other flags are not affected.

(1) **Zero flag (ZF)**

The ZF is set to "1" if the operation result or the transfer data is $00_H$ (for 8-bit operations and data transfers)/$0000_H$ (for 16-bit operations); otherwise the ZF is cleared to "0".

During the bit manipulation instructions [SET, CLR, and CPL], the ZF is set to "1" if the contents of the specified bit is "0"; otherwise the ZF is cleared to "0".

This flag is set to "1" when the upper 8 bits of the product are $00_H$ during the multiplication instruction [MUL], and when $00_H$ for the remainder during the division instruction [DIV]; otherwise it is cleared to "0".

(2) **Carry flag (CF)**

The CF is set to "1" when a carry out of the MSB (most significant bit) of the result occurred during addition or when a borrow into the MSB of the result occurred during subtraction; otherwise the CF is cleared to "0". During division, this flag is set to "1" when the divisor is $00_H$ (divided by zero error), or when the quotient is $100_H$ or higher (overflow error); otherwise it is cleared. The CF is also affected during the shift/rotate instructions [SHLC, SHRC, ROLC, and RORC]. The data shifted out from a register is set to the CF.

This flag is also a 1-bit register (a boolean accumulator) for the bit manipulation instructions.

Set/clear/complement are possible with the CF manipulation instructions.

Example1 : Bit manipulation

```
LD      CF, (0007H) . 5      ; (0001H)2 ← (0007H)5 ∀ (009AH)0
XOR     CF, (009AH) . 0
LD      (0001H) . 2, CF
```

Example2 : Arithmetic right shift

```
LD      CF, A . 7            ; A ← A / 2
RORC    A
```

(3) **Half carry flag (HF)**

The HF is set to "1" when a carry occurred between bits 3 and 4 of the operation result during an 8-bit addition, or when a borrow occurred from bit 4 into bit 3 of the result during an 8-bit subtraction; otherwise the HF is cleared to "0". This flag is useful in the decimal adjustment for BCD operations (adjustments using the [DAA r], or [DAS r] instructions).

Example : BCD operation
(The A becomes $47_H$ after executing the following program when A = $19_H$, B = $28_H$)

```
ADD     A, B        ; A ← 41H, HF ← 1
DAA     A           ; A ← 41H + 06H = 47H (decimal-adjust)
```

(4) **Jump status flag (JF)**

Zero or carry information is set to the JF after operation (e. g. INC, ADD, CMP, TEST).
The JF provides the jump condition for conditional jump instructions [JRS T/F, $ + 2 + d], [JR T/F, $ + 2 + d] (T or F is a condition code). Jump is performed if the JF is "1" for a true condition (T), or the JF is "0" for a false condition (F).
The JF is set to "1" after executing the load/exchange/swap/nibble rotate/jump instruction, so that [JRS T, $ + 2 + d] and [JR T, $ + 2 + d] can be regarded as an unconditional jump instruction.

Example : Jump status flag and conditional jump instruction

```
INC     A
JRS     T, SLABLE1      ; Jump when a carry is caused by the immediately
 ⋮                        preceding operation instruction.
LD      A, (HL)
JRS     T, SLABLE2      ; JF is set to "1" by the immediately preceding
 ⋮                        instruction, making it an unconditional jump
                          instruction.
```

Example : The accumulator and flags become as shown below after executing the following instructions when the WA register pair, the HL register pair, the data memory at address $00C5_H$, the carry flag and the half carry flag contents being "$219A_H$", "$00C5_H$", "$D7_H$", "1" and "0", respectively.

| Instruction | Acc. after execution | Flag after execution | | | |
|---|---|---|---|---|---|
| | | JF | ZF | CF | HF |
| ADDC A, (HL) | 72 | 1 | 0 | 1 | 1 |
| SUBB A, (HL) | C2 | 1 | 0 | 1 | 0 |
| CMP A, (HL) | 9A | 0 | 0 | 1 | 0 |
| AND A, (HL) | 92 | 0 | 0 | 1 | 0 |
| LD A, (HL) | D7 | 1 | 0 | 1 | 0 |
| ADD A, 66H | 00 | 1 | 1 | 1 | 1 |

| Instruction | Acc. after execution | Flag after execution | | | |
|---|---|---|---|---|---|
| | | JF | ZF | CF | HF |
| INC A | 9B | 0 | 0 | 1 | 0 |
| ROLC A | 35 | 1 | 0 | 1 | 0 |
| RORC A | CD | 0 | 0 | 0 | 0 |
| ADD WA, 0F508H | 16A2 | 1 | 0 | 1 | 0 |
| MUL W, A | 13DA | 0 | 0 | 1 | 0 |
| SET A.5 | BA | 1 | 1 | 1 | 0 |

## 1.7 Stack and Stack Pointer

### 1.7.1 Stack

The stack provides the area in which the return address or status, etc. are saved before a jump is performed to the processing routine during the execution of a subroutine call instruction or the acceptance of an interrupt. On a subroutine call instruction [CALL a] / [CALLP n] / [CALLV n], the contents of the PC (the return address) is saved; on an interrupt acceptance, the contents of the PC and the PSW are saved (the PSW is pushed first, followed by $PC_H$ and $PC_L$). Therefore, a subroutine call occupies two bytes on the stack; an interrupt occupies three bytes.
When returning from the processing routine, executing a subroutine return instruction [RET] restores the contents to the PC from the stack; executing an interrupt return instruction [RETI] / [RETN] restores the contents to the PC and the PSW (the $PC_L$ is popped first, followed by $PC_H$ and PSW).
The stack can be located anywhere within the data memory space except the register bank area, therefore the stack depth is limited only by the free data memory size.

### 1.7.2 Stack Pointer (SP)

The stack pointer (SP) is a 16-bit register containing the address of the next free locations on the stack.

The SP is post-decremented when a subroutine call or a push instruction is executed, or when an interrupt is accepted; and the SP is pre-incremented when a return or a pop instruction is executed. Figure 1-8 shows the stacking order.

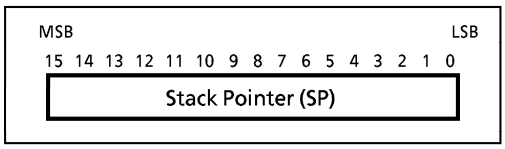



Figure 1-7. Stack Pointer

The SP is not initialized hardware-wise but requires initialization by an initialize routine (sets the highest stack address). [LD SP, mn], [LD SP, gg] and [LD gg, SP] are the SP access instructions (mn ; 16-bit immediate data, gg ; register pair).

Example 1 : To initialize the SP

```
LD      SP, 013FH       ; SP←013F_H
```

Example 2 : To read the SP

```
LD      HL, SP          ; HL←SP
```

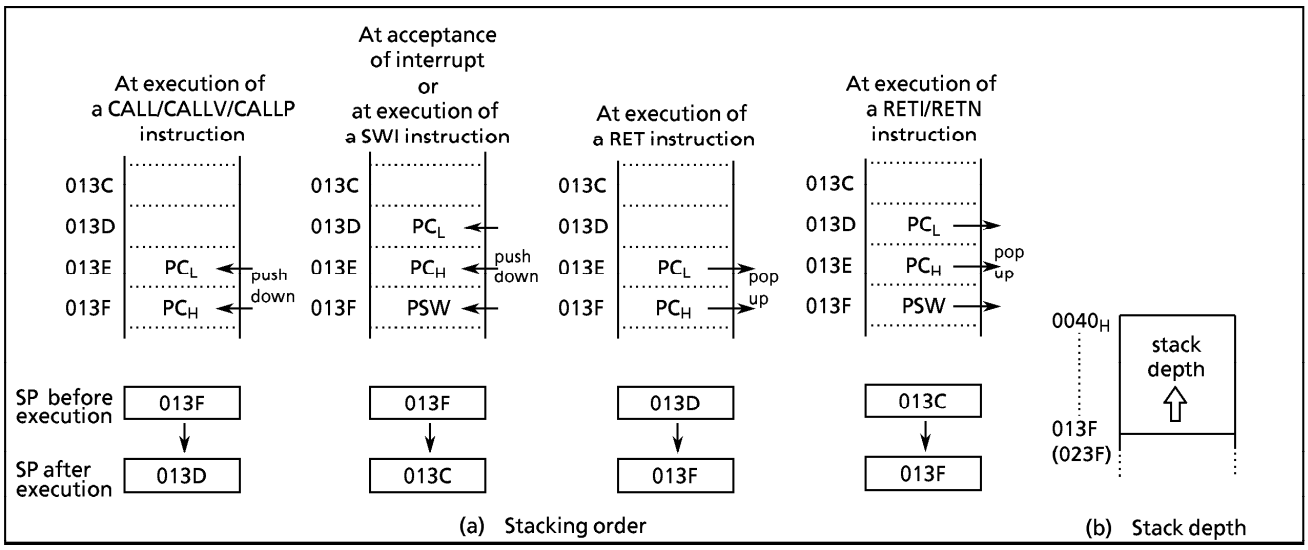



Figure 1-8. Stack

## 1.8 System Clock Controller

The system clock controller consists of a clock generator, a timing generator, and a stand-by controller.

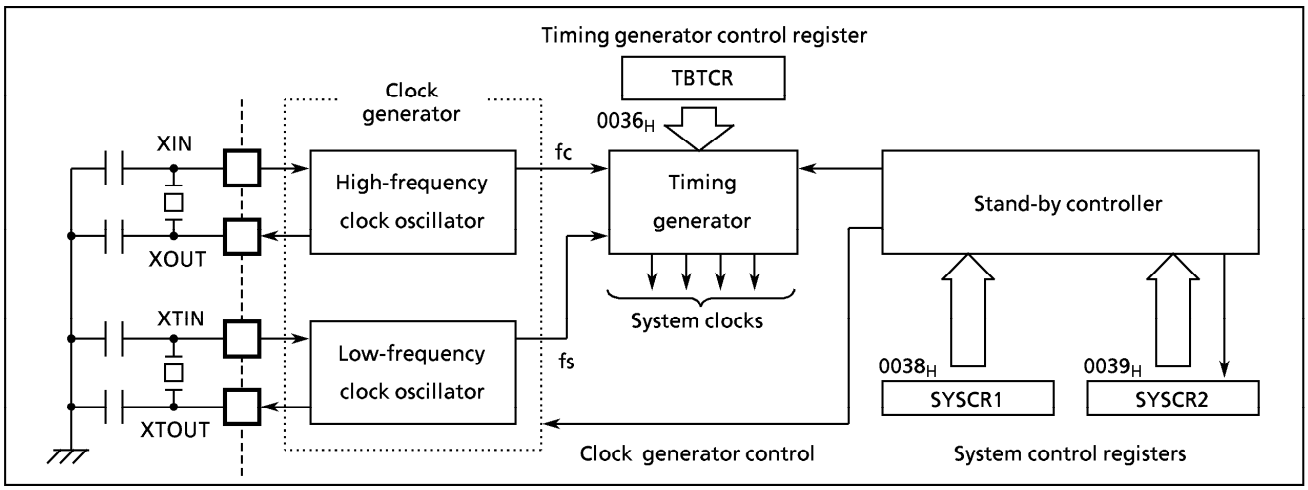



Figure 1-9. System Clock Controller

### 1.8.1 Clock Generator

The clock generator generates the basic clock which provides the system clocks supplied to the CPU core and peripheral hardware. It contains two oscillation circuits: one for the high-frequency clock and one for the low-frequency clock. Power consumption can be reduced by switching of the system clock controller to low-power operation based on the low-frequency clock.

The high-frequency (fc) and low-frequency (fs) clocks can be easily obtained by connecting a resonator between the XIN/XOUT and XTIN/XTOUT pins, respectively. Clock input from an external oscillator is also possible. In this case, external clock is applied to the XIN/XTIN pin with the XOUT/XTOUT pin not connected. The 87C841/CC41/CH41/CK41/CM41 are not provided an RC oscillation.

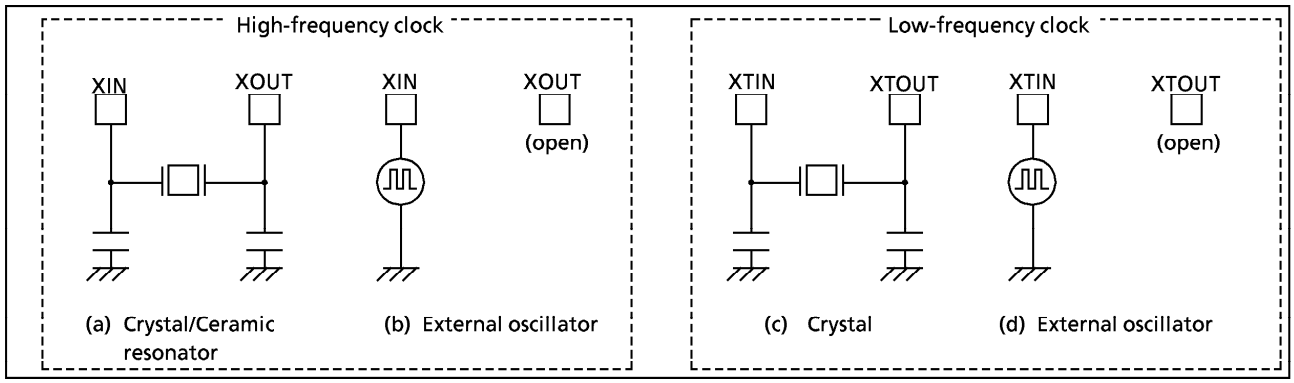
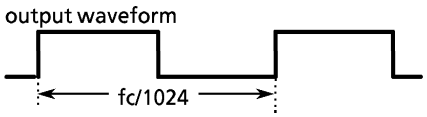



Figure 1-10. Examples of Resonator Connection

*Note : Accurate Adjustment of the Oscillation Frequency:*
*Although no hardware to externally and directly monitor the basic clock pulse is not provided, the oscillation frequency can be adjusted by making the program to output fixed frequency pulses to the port while disabling all interrupts and monitoring this pulse. With a system requiring adjustment of the oscillation frequency, the adjusting program must be created beforehand.*

Example: To output the high-frequency oscillation frequency adjusting monitor pulse to P13 ($\overline{DVO}$) pin.

```
SFCCHK:  LD   (P1CR), 00001000B  ; Configures port P13 as an output
         SET  (P1).3             ; P13 output latch ← 1
         LD   (TBTCR), 11100000B ; Enables divider output
         JRS  T, $               ; Loops endless
```




### 1.8.2 Timing Generator

The timing generator generates from the basic clock the various system clocks supplied to the CPU core and peripheral hardware. The timing generator provides the following functions :

① Generation of main system clock
② Generation of divider output ($\overline{DVO}$) pulses
③ Generation of source clocks for time base timer
④ Generation of source clocks for watchdog timer
⑤ Generation of internal source clocks for timer/counters TC1 – TC4
⑥ Generation of internal clocks for serial interfaces SIO1 and SIO2
⑦ Generation of warm-up clocks for releasing STOP mode
⑧ Generation of a clock for releasing reset output

(1) Configuration of Timing Generator

The timing generator consists of a 21-stage divider with a divided-by-4 prescaler, a main system clock generator, and machine cycle counters. An input clock to the 7th stage of the divider depends on the operating mode and DV7CK (bit 4 in TBTCR) shown in Figure 1-11 as follows.

During reset and at releasing STOP mode, the divider is cleared to "0", however, the prescaler is not cleared.

① In the single-clock mode
A divided-by-256 of high-frequency clock (fc/$2^8$) is input to the 7th stage of the divider.

② In the dual-clock mode
During NORMAL2 or IDLE2 mode (SYSCK = 0), an input clock to the 7th stage of the divider can be selected either "fc/$2^8$" or "fs" with DV7CK.
During SLOW or SLEEP mode (SYSCK = 1), fs is automatically input to the 7th stage. To input clock to the 1st stage is stopped ; output from the 1st to 6th stages is also stopped.

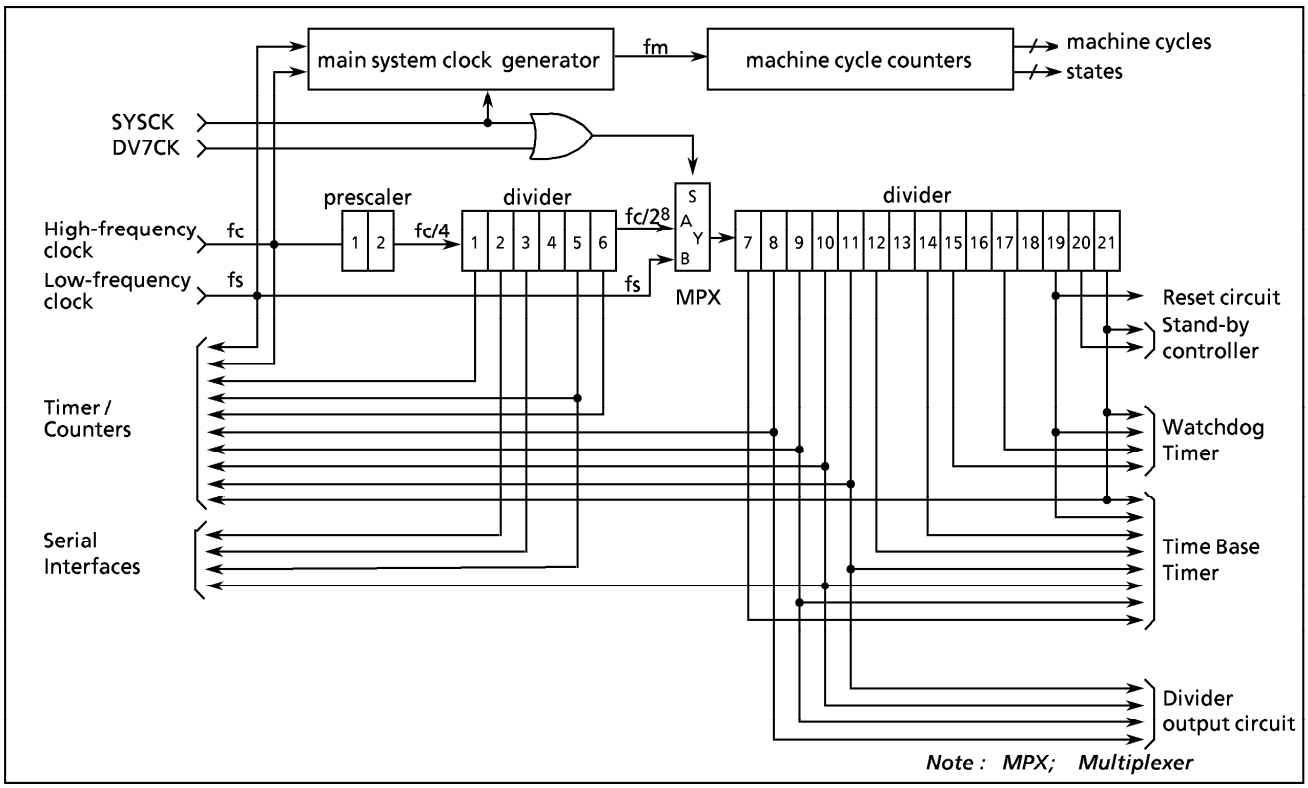



Figure 1-11. Configuration of Timing Generator

**TBTCR** ($0036_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| (DVOEN) | (DVOCK) | | DV7CK | (TBTEN) | (TBTCK) | | |

(Initial value: 0**0 0***)

| | | | |
|---|---|---|---|
| DV7CK | Selection of input clock to the 7th stage of the divider | 0 : fc/$2^8$ [Hz]<br>1 : fs | R/W |

*Note1 : fc ; high-frequency clock [Hz], fs ; low-frequency clock [Hz], * ; don't care*
*Note 2 : Do not set DV7CK to "1" in the single-clock mode.*
*Note 3 : Do not set DV7CK to "1" before low-frequency clock is stable in the dual-clock mode.*

Figure 1-12. Timing Generator Control Register

(2) Machine Cycle

Instruction execution and peripheral hardware operation are synchronized with the main system clock. The minimum instruction execution unit is called an "machine cycle". There are a total of 10 different types of instructions for the TLCS-870 Series: ranging from 1-cycle instructions which require one machine cycle for execution to 10-cycle instructions which require 10 machine cycles for execution.

A machine cycle consists of 4 states (S0 to S3), and each state consists of one main system clock.

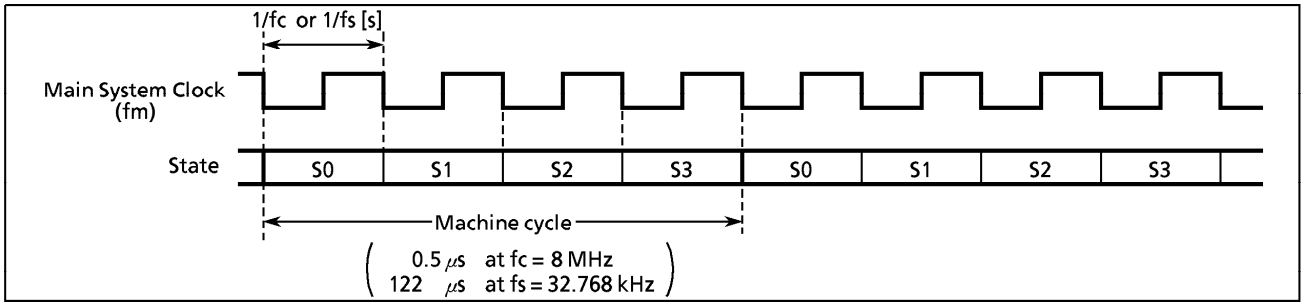



Figure 1-13. Machine Cycle

### 1.8.3 Stand-by Controller

The stand-by controller starts and stops the oscillation circuits for the high-frequency and low-frequency clocks, and switches the main system clock. There are two operating modes: single-clock and dual-clock. These modes are controlled by the system control registers (SYSCR1, SYSCR2).

Figure 1-14 shows the operating mode transition diagram and Figure 1-15 shows the system control registers. Either the single-clock or the dual-clock mode can be selected by an option during reset.

(1) **Single-clock mode**

Only the oscillation circuit for the high-frequency clock is used, and P21 (XTIN) and P22 (XTOUT) pins are used as input/output ports. In the single-clock mode, the machine cycle time is 4/fc [s] (0.5 μs at fc = 8 MHz).

① NORMAL1 mode

In this mode, both the CPU core and on-chip peripherals operate using the high-frequency clock. In the case where the single-clock mode has been selected as an option, the 87C841/CC41/CH41/CK41/CM41 are placed in this mode after reset.

② IDLE1 mode

In this mode, the internal oscillation circuit remains active, and the CPU and the watchdog timer are halted; however, on-chip peripherals remain active (operate using the high-frequency clock). IDLE1 mode is started by setting IDLE bit in the system control register 2 (SYSCR2), and IDLE1 mode is released to NORMAL1 mode by an interrupt request from on-chip peripherals or external interrupt inputs. When IMF (interrupt master enable flag) is "1" (interrupt enable), the execution will resume upon acceptance of the interrupt, and the operation will return to normal after the interrupt service is completed. When IMF is "0" (interrupt disable), the execution will resume with the instruction which follows IDLE mode start instruction.

③ STOP1 mode

In this mode, the internal oscillation circuit is turned off, causing all system operations to be halted. The internal status immediately prior to the halt is held with the lowest power consumption during this mode. The output status of all output ports can be set to either output hold or high-impedance under software control.

STOP1 mode is started by setting STOP bit in the system control register 1 (SYSCR1), and STOP1 mode is released by an input (either level-sensitive or edge-sensitive can be programmably selected) to the $\overline{\text{STOP}}$ pin. After the warming-up period is completed, the execution resumes with the next instruction which follows the STOP mode start instruction.

(2) **Dual-clock mode**

Both high-frequency and low-frequency oscillation circuits are used in this mode. Pins P21 (XTIN) and P22 (XTOUT) cannot be used as input/output ports. The main system clock is obtained from the high-frequency clock in NORMAL2 and IDLE2 modes, and is obtained from the low-frequency clock in SLOW and SLEEP modes. The machine cycle time is 4/fc [s] (0.5 $\mu$s at fc = 8 MHz) in NORMAL2 and IDLE2 modes, and 4/fs [s] (122 $\mu$s at fs = 32.768 kHz) in SLOW and SLEEP modes. Note that the 87PM41 is placed in the single-clock mode during reset. To use the dual-clock mode, the low-frequency oscillator should be turned on by executing [SET (SYSCR2).XTEN] instruction.

① NORMAL2 mode

In this mode, the CPU core operates using the high-frequency clock. On-chip peripherals operate using the high-frequency clock and/or low-frequency clock. In case that the dual-clock mode has been selected by an option, the 87C841/CC41/CH41/CK41/CM41 are placed in this mode after reset.

② SLOW mode

This mode can be used to reduce power-consumption by turning off oscillation of the high-frequency clock. The CPU core and on-chip peripherals operate using the low-frequency clock.

Switching back and forth between NORMAL2 and SLOW modes is performed by the system control register 2.

③ IDLE2 mode

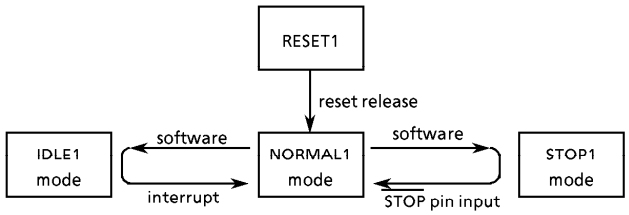
In this mode, the internal oscillation circuits remain active. The CPU and the watchdog timer are halted; however, on-chip peripherals remain active (operate using the high-frequency clock and/or the low-frequency clock). Starting and releasing of IDLE2 mode are the same as for IDLE1 mode, except that operation returns to NORMAL2 mode.

④ SLEEP mode

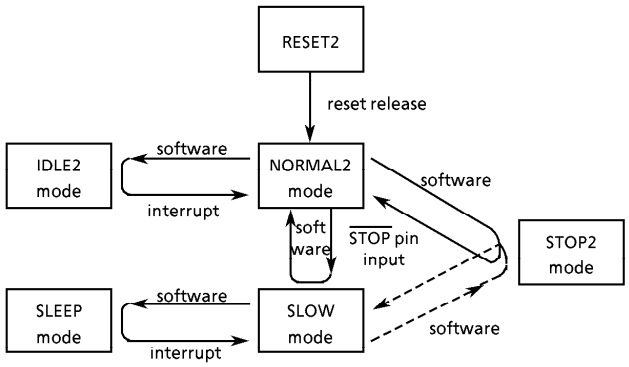
In this mode, the internal oscillation circuit of the low-frequency clock remains active. The CPU, the watchdog timer, and the internal oscillation circuit of the high- frequency clock are halted; however, on-chip peripherals remain active (operate using the low-frequency clock). Starting and releasing of SLEEP mode is the same as for IDLE1 mode, except that operation returns to SLOW mode.

⑤ STOP2 mode

As in STOP1 mode, all system operations are halted in this mode.




(a) Single-clock mode




(b) Dual-clock mode

*Note : NORMAL1 and NORMAL2 modes are generically called NORMAL; STOP1 and STOP2 are called STOP; and IDLE1, IDLE2 and SLEEP are called IDLE.*

| Operating mode | | Frequency | | CPU core | On-chip Peripherals | Machine cycle time |
|---|---|---|---|---|---|---|
| | | High-frequency | Low-frequency | | | |
| Single-Clock | RESET1 | turning on oscillation | turning off oscillation | reset | reset | 4/fc [s] |
| | NORMAL1 | | | operate | operate | |
| | IDLE1 | | | halt | | |
| | STOP1 | turning off oscillation | | | halt | — |
| Dual-Clock | RESET2 | turning on oscillation | turning on oscillation | reset | reset | 4/fc [s] |
| | NORMAL2 | | | High-frequency | operate (High and/or Low) | |
| | IDLE2 | | | halt | | |
| | SLOW | turning off oscillation | | Low-frequency | Low-frequency | 4/fs [s] |
| | SLEEP | | | halt | | |
| | STOP2 | | turning off oscillation | | halt | — |

Figure 1-14. Operating Mode Transition Diagram

System Control Register 1

**SYSCR1** ($0038_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| STOP | RELM | RETM | OUTEN | WUT | | | |

(Initial value: 0000 00**)

| | | | |
|---|---|---|---|
| STOP | STOP mode start | 0 : CPU core and peripherals remain active<br>1 : CPU core and peripherals are halted (start STOP mode) | R/W |
| RELM | Release method for STOP mode | 0 : Edge-sensitive release<br>1 : Level-sensitive release | |
| RETM | Operating mode after STOP mode | 0 : Return to NORMAL mode<br>1 : Return to SLOW mode | |
| OUTEN | Port output control during STOP mode | 0 : High-impedance<br>1 : Remain unchanged | |
| WUT | Warming-up time at releasing STOP mode | 00 : $3\times2^{19}$ / fc or $3\times2^{13}$ / fs [s]<br>01 : $2^{19}$ / fc or $2^{13}$ / fs<br>1* : Reserved | |

*Note 1 :* *Always set RETM to "0" when transiting from NORMAL1 mode to STOP1 mode and from NOMAL2 mode to STOP2 mode. Always set RETM to "1" when transiting from SLOW mode to STOP2 mode.*

*Note 2 :* *When STOP mode is released with $\overline{\text{RESET}}$ pin input, a return is made to NORMAL mode regardless of the RETM contents.*

*Note 3 :* fc ; *high-frequency clock* [Hz]
fs ; *low-frequency clock* [Hz]
* ; *don't care*

*Note 4 :* *Bits 1 and 0 in SYSCR1 are read in as undefined data when a read instruction is executed.*

System Control Register 2

**SYSCR2** ($0039_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| XEN | XTEN | SYSCK | IDLE | | | | |

(Initial value: 10/100 ****)

| | | | |
|---|---|---|---|
| XEN | High-frequency oscillator control | 0 : Turn off oscillation<br>1 : Turn on oscillation | R/W |
| XTEN | Low-frequency oscillator control | 0 : Turn off oscillation<br>1 : Turn on oscillation | |
| SYSCK | Main system clock select (write)/main system clock monitor (read) | 0 : High-frequency clock<br>1 : Low-frequency clock | |
| IDLE | IDLE mode start | 0 : CPU and watchdog timer remain active<br>1 : CPU and watchdog timer are stopped (start IDLE mode) | |

*Note 1 :* *A reset is applied ($\overline{\text{RESET}}$ pin output goes low) if both XEN and XTEN are cleared to "0".*

*Note 2 :* *Do not clear XEN to "0" when SYSCK = 0, and do not clear XTEN to "0" when SYSCK = 1.*

*Note 3 :* *WDT; watchdog timer, * ; don't care*

*Note 4 :* *Bits 3 - 0 in SYSCR2 are always read in as "1" when a read instruction is executed.*

*Note 5 :* *An optional initial value can be selected for XTEN. Always specify when ordering ES (engineering sample).*

| XTEN | operating mode after reset |
|---|---|
| 0 | Single-clock mode (NORMAL1) |
| 1 | Dual-clock mode (NORMAL2) |

Figure 1-15. System Control Registers

## 1.8.4 Operating Mode Control

(1) **STOP** mode (STOP1, STOP2)

STOP mode is controlled by the system control register 1 (SYSCR1) and the $\overline{\text{STOP}}$ pin input. The $\overline{\text{STOP}}$ pin is also used both as a port P20 and an $\overline{\text{INT5}}$ (external interrupt input 5) pin. STOP mode is started by setting STOP (bit 7 in SYSCR1 ) to "1". During STOP mode, the following status is maintained.

① Oscillations are turned off, and all internal operations are halted.
② The data memory, registers and port output latches are all held in the status in effect before STOP mode was entered. The port output can be select either output hold or high-impedance by setting OUTEN ( bit 4 in SYSCR1).
③ The divider of the timing generator is cleared to "0".
④ The program counter holds the address of the instruction following the instruction which started the STOP mode.

STOP mode includes a level-sensitive release mode and an edge-sensitive release mode, either of which can be selected with RELM (bit 6 in SYSCR1).

a. Level-sensitive release mode (RELM = 1)

In this mode, STOP mode is released by setting the $\overline{\text{STOP}}$ pin high. This mode is used for capacitor back-up when the main power supply is cut off and long term battery back-up.

When the $\overline{\text{STOP}}$ pin input is high, executing an instruction which starts the STOP mode will not place in STOP mode but instead will immediately start the release sequence (warm-up). Thus, to start STOP mode in the level-sensitive release mode, it is necessary for the program to first confirm that the $\overline{\text{STOP}}$ pin input is low. The following method can be used for confirmation:

- Using an external interrupt input $\overline{\text{INT5}}$ ($\overline{\text{INT5}}$ is a falling edge-sensitive input).

Example : Starting STOP mode with an INT5 interrupt.

```
PINT5 :   TEST   (P2) . 0              ;  To reject noise, the STOP mode does not start if
          JRS    F, SINT5                 port P20 is at high
          LD     (SYSCR1), 01000000B     ;  Sets up the level-sensitive release mode.
          SET    (SYSCR1) . 7            ;  Starts STOP mode
SINT5 :   RETI
```

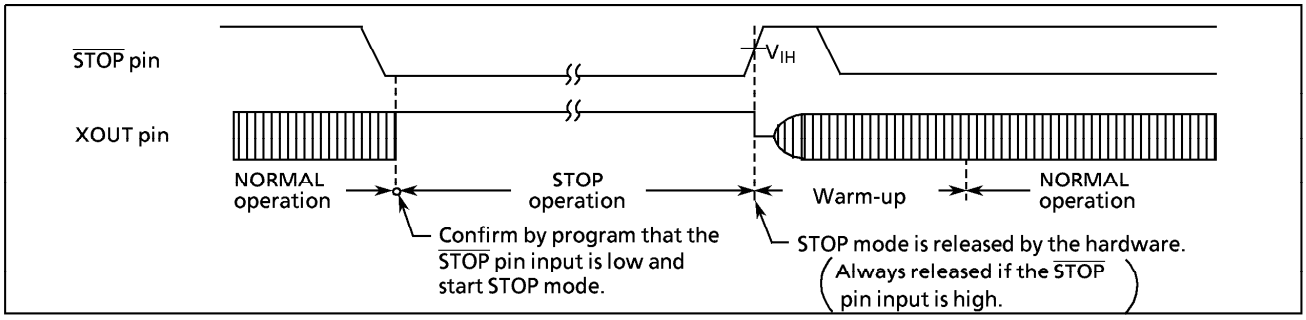



Figure 1-16. Level-sensitive Release Mode

*Note : When changing to the level-sensitive release mode from the edge-sensitive release mode, the release mode is not switched until a rising edge of the $\overline{\text{STOP}}$ pin input is detected.*

b. Edge-sensitive release mode (RELM = 0)

In this mode, STOP mode is released by a rising edge of the $\overline{STOP}$ pin input. This is used in applications where a relatively short program is executed repeatedly at periodic intervals. This periodic signal (for example, a clock from a low-power consumption oscillator) is input to the $\overline{STOP}$ pin.

In the edge-sensitive release mode, STOP mode is started even when the $\overline{STOP}$ pin input is high.

Example : Starting STOP mode operation in the edge-sensitive release mode

```
LD      (SYSCR1), 10000000B    ;  OUTEN ← 0 (specifies high-impedance)
DI                             ;  IMF ← 0 (disables interrupt service)
SET     (SYSCR1).STOP          ;  STOP ← 1 (activates stop mode)
LDW     (IL),1110011101010111B ;  IL12, 11, 7, 5, 3 ← 0
                                  (clears interrupt latches)
EI                             ;  IMF ← 1 (enables interrupt service)
```

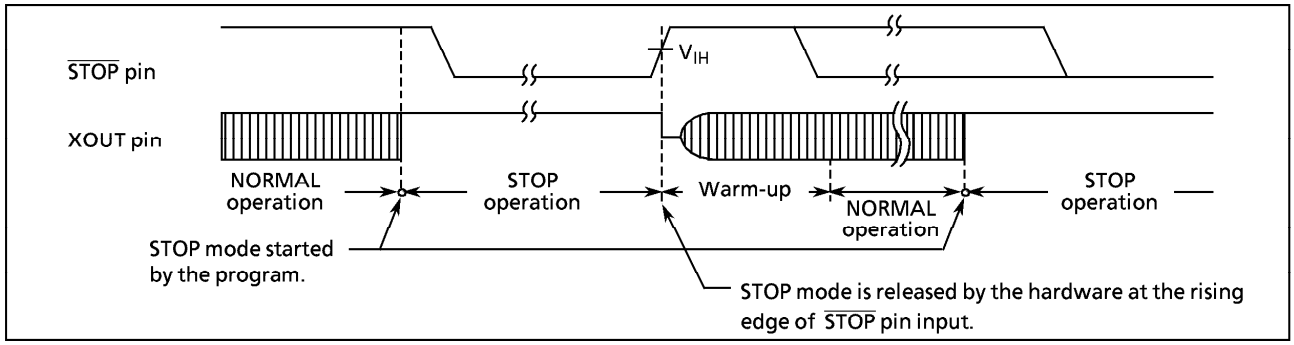



Figure 1-17. Edge-sensitive Release Mode

STOP mode is released by the following sequence:

① When returning to NORMAL2, both the high-frequency and low-frequency clock oscillators are turned on ; when returning to SLOW mode, only the low-frequency clock oscillator is turned on. When returning to NORMAL1, only the high-frequency clock oscillator is turned on.

② A warming-up period is inserted to allow oscillation time to stabilize. During warm-up, all internal operations remain halted. Two different warming-up times can be selected with WUT ( bits 2 and 3 in SYSCR1) as determined by the resonator characteristics.

③ When the warming-up time has elapsed, normal operation resumes with the instruction following the STOP mode start instruction (e.g. [SET (SYSCR1). 7]). The start is made after the divider of the timing generator is cleared to "0".

Table 1-1. Warming-up Time example

| Return to NORMAL1 mode | | | Return to SLOW mode | |
|---|---|---|---|---|
| WUT | At fc = 4.194304 MHz | At fc = 8 MHz | WUT | At fs = 32.768 kHz |
| $3 \times 2^{19}$ / fc [s] | 375 [ms] | 196.6 [ms] | $3 \times 2^{13}$ / fs [s] | 750 [ms] |
| $2^{19}$ / fc | 125 | 65.5 | $2^{13}$ / fs | 250 |

*Note : The warming-up time is obtained by dividing the basic clock by the divider: therefore, the warming-up time may include a certain amount of error if there is any fluctuation of the oscillation frequency when STOP mode is released. Thus, the warming-up time must be considered an approximate value.*

STOP mode can also be released by setting the $\overline{RESET}$ pin low, which immediately performs the normal reset operation.

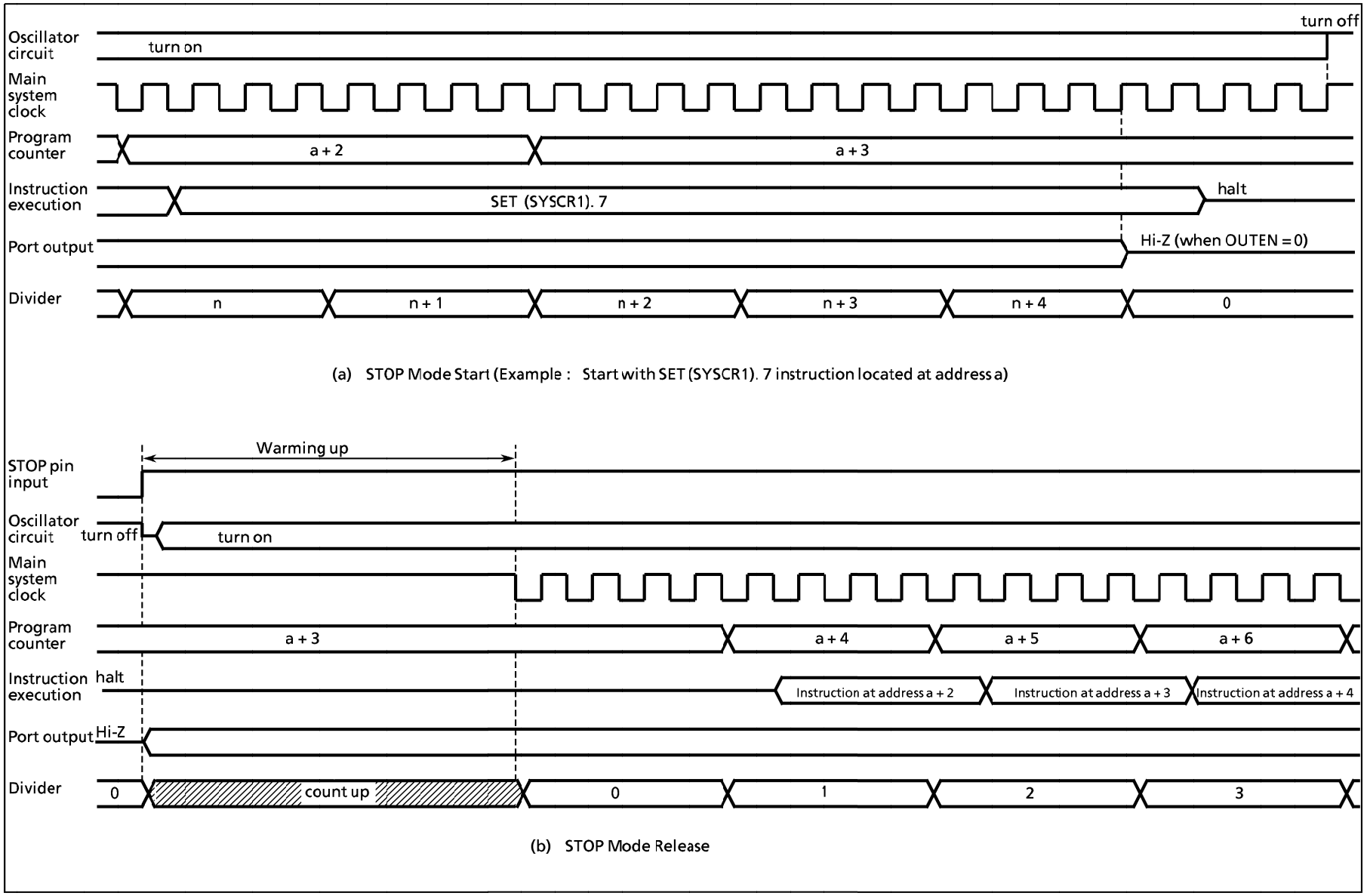



Figure 1-18. STOP Mode Start / Release

*Note : When STOP mode is released with a low hold voltage, the following cautions must be observed.*
*The power supply voltage must be at the operating voltage level before releasing STOP mode. The $\overline{\text{RESET}}$ pin input must also be high, rising together with the power supply voltage. In this case, if an external time constant circuit has been connected, the $\overline{\text{RESET}}$ pin input voltage will increase at a slower rate than the power supply voltage. At this time, there is a danger that a reset may occur if input voltage level of the $\overline{\text{RESET}}$ pin drops below the non-inverting high-level input voltage (hysteresis input).*

(2) **IDLE** mode (IDLE1, IDLE2, SLEEP)

IDLE mode is controlled by the system control register 2 and maskable interrupts. The following status is maintained during IDLE mode.

① Operation of the CPU and watchdog timer is halted. On-chip peripherals continue to operate.
② The data memory, CPU registers and port output latches are all held in the status in effect before IDLE mode was entered.
③ The program counter holds the address of the instruction following the instruction which started IDLE mode.

Example : Starting IDLE mode.

```
SET        (SYSCR2) . 4        ;  IDLE←1
```

IDLE mode includes a normal release mode and an interrupt release mode. Selection is made with the interrupt master enable flag (IMF). Releasing the IDLE mode returns from IDLE1 to NORMAL1, from IDLE2 to NORMAL2, and from SLEEP to SLOW mode.

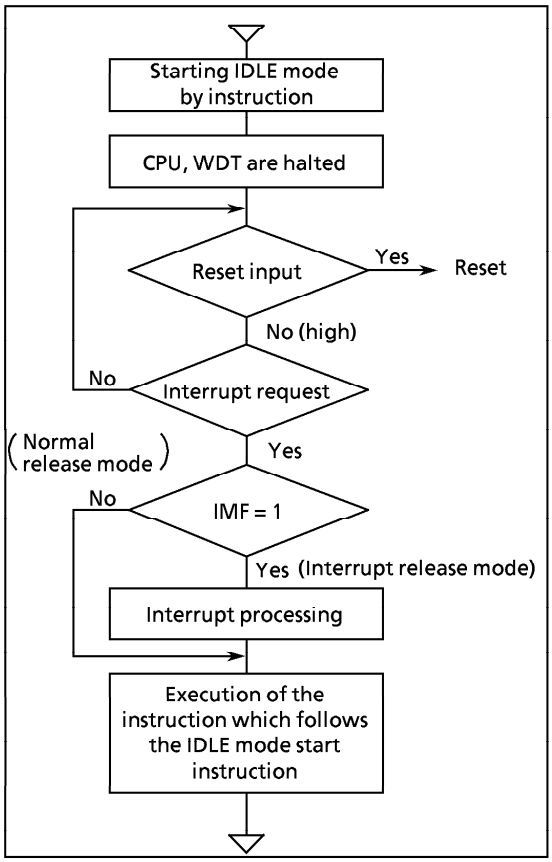



Figure 1-19. IDLE Mode

a. Normal release mode (IMF = "0")
IDLE mode is released by any interrupt source enabled by the individual interrupt enable flag (EF) or an external interrupt 0 ($\overline{\text{INT0}}$ pin) request. Execution resumes with the instruction following the IDLE mode start instruction (e.g. [SET (SYSCR2).4]).

b. Interrupt release mode (IMF = "1")
IDLE mode is released and interrupt processing is started by any interrupt source enabled with the individual interrupt enable flag (EF) or an external interrupt 0 ($\overline{\text{INT0}}$ pin) request. After the interrupt is processed, the execution resumes from the instruction following the instruction which started IDLE mode.

IDLE mode can also be released by setting the $\overline{\text{RESET}}$ pin low, which immediately performs the reset operation. After reset, the 87C841/CC41/CH41/CK41/CM41 are placed in NORMAL mode.

*Note : When a watchdog timer interrupt is generated immediately before the IDLE mode is started, the watchdog timer interrupt will be processed but IDLE mode will not be started.*

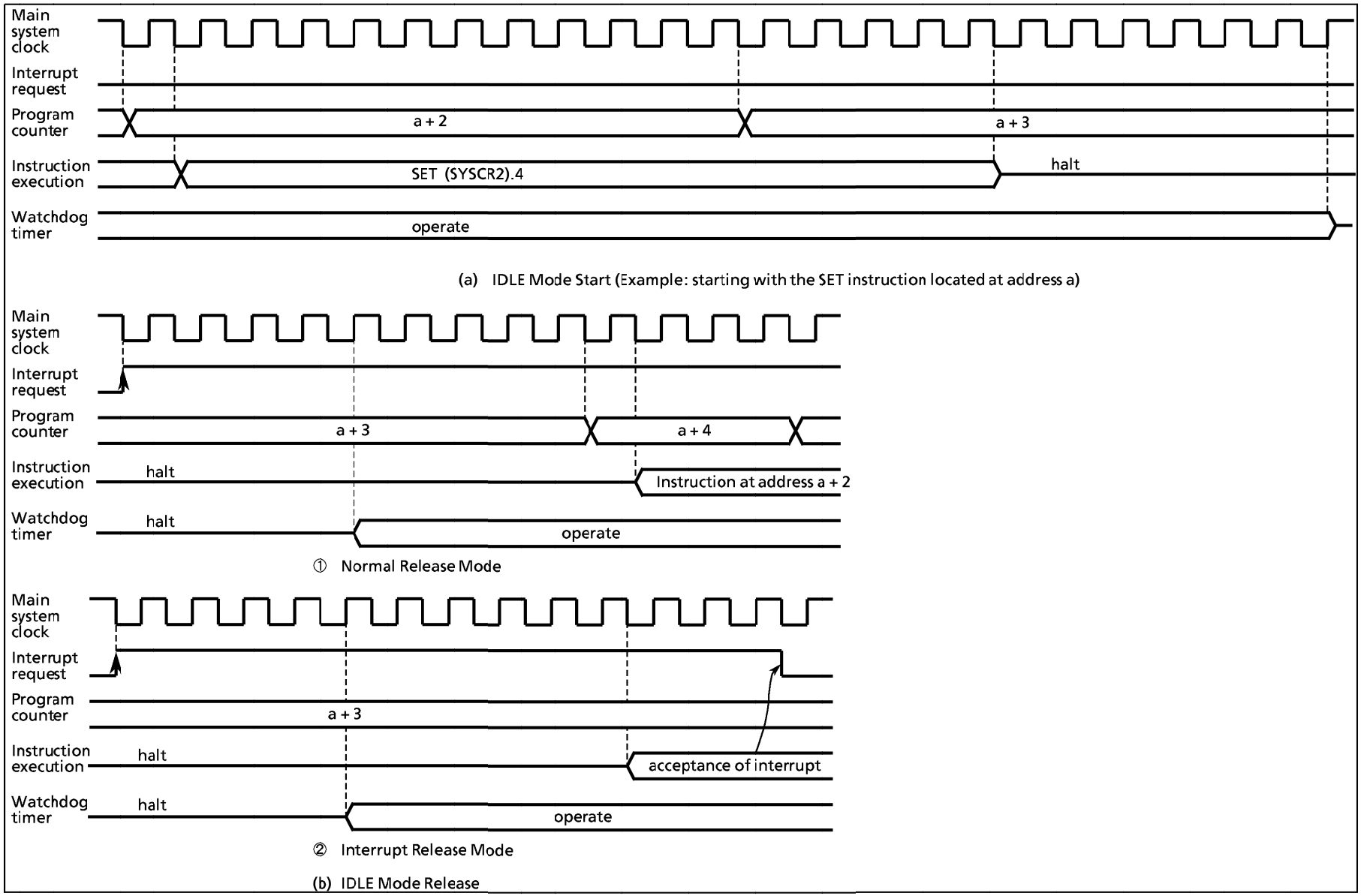



Figure 1-20. IDLE Mode Start/Release

(3) **SLOW** mode

SLOW mode is controlled by the system control register 2 and the timer/counter 2.

a. Switching from NORMAL2 mode to SLOW mode

First, set SYSCK (bit 5 in SYSCR2) to switch the main system clock to the low-frequency clock. Next, clear XEN (bit 7 in SYSCR2) to turn off high-frequency oscillation.

When the low-frequency clock oscillation is unstable, wait until oscillation stabilizes before performing the above operations. The timer/counter 2 (TC2) can conveniently be used to confirm that low-frequency clock oscillation has stabilized.

*Note : The high frequency clock can be continued oscillation in order to return to NORMAL2 mode from SLOW mode quickly. Always turn off oscillation of high frequency clock when switching from SLOW mode to STOP mode.*

Example1 : Switching from NORMAL2 mode to SLOW mode.

```
        SET     (SYSCR2) . 5        ; SYSCK←1   (Switches the main system clock to the
                                                low-frequency clock)
        CLR     (SYSCR2) . 7        ; XEN←0     (turns off high-frequency oscillation)
```

Example2 : Switching to SLOW mode after low-frequency clock oscillation has stabilized.

```
        LD      (TC2CR), 14H        ; Sets TC2 mode
                                      (timer mode, source clock : fs)
        LDW     (TREG2), 8000H      ; Sets warming-up time
                                      (according to Xtal characteristics)
        LD      (TC2CR), 34H        ; Starts TC2
        ⋮
PINTTC2 : LD    (TC2CR), 10H        ; Stops TC2
        SET     (SYSCR2) . 5        ; SYSCK←1
        CLR     (SYSCR2) . 7        ; XEN←0
        RETI
        ⋮
VINTTC2 : DW    PINTTC2             ; INTTC2 vector table
```

b. Switching from SLOW mode to NORMAL2 mode

First, set XEN (bit 7 in SYSCR2) to turn on the high-frequency oscillation. When time for stabilization (warm-up) has been taken by the timer/counter 2 (TC2), clear SYSCK (bit 5 in SYSCR2) to switch the main system clock to the high-frequency clock.

SLOW mode can also be released by setting the $\overline{\text{RESET}}$ pin low, which immediately performs the reset operation. After reset, the 87C841/CC41/CH41/CK41/CM41 are placed in NORMAL mode.

Example : Switching from SLOW mode to NORMAL2 mode (fc = 8 MHz, warming-up time is about 7.9 ms).

```
        SET     (SYSCR2) . 7        ; XEN←1     (turns on high-frequency oscillation)
        LD      (TC2CR), 10H        ; Sets TC2 mode
                                      (timer mode, source clock: fc)
        LD      (TREG2 + 1), 0F8H   ; Sets the warming-up time
                                      (according to frequency and resonator characteristics)
        LD      (TC2CR), 30H        ; Starts TC2
        ⋮
PINTTC2 : LD    (TC2CR), 10H        ; Stops TC2
        CLR     (SYSCR2) . 5        ; SYSCK←0   (Switches the main system clock to the
                                                high-frequency clcok)
        RETI
        ⋮
VINTTC2 : DW    PINTTC2             ; INTTC2 vector table
```

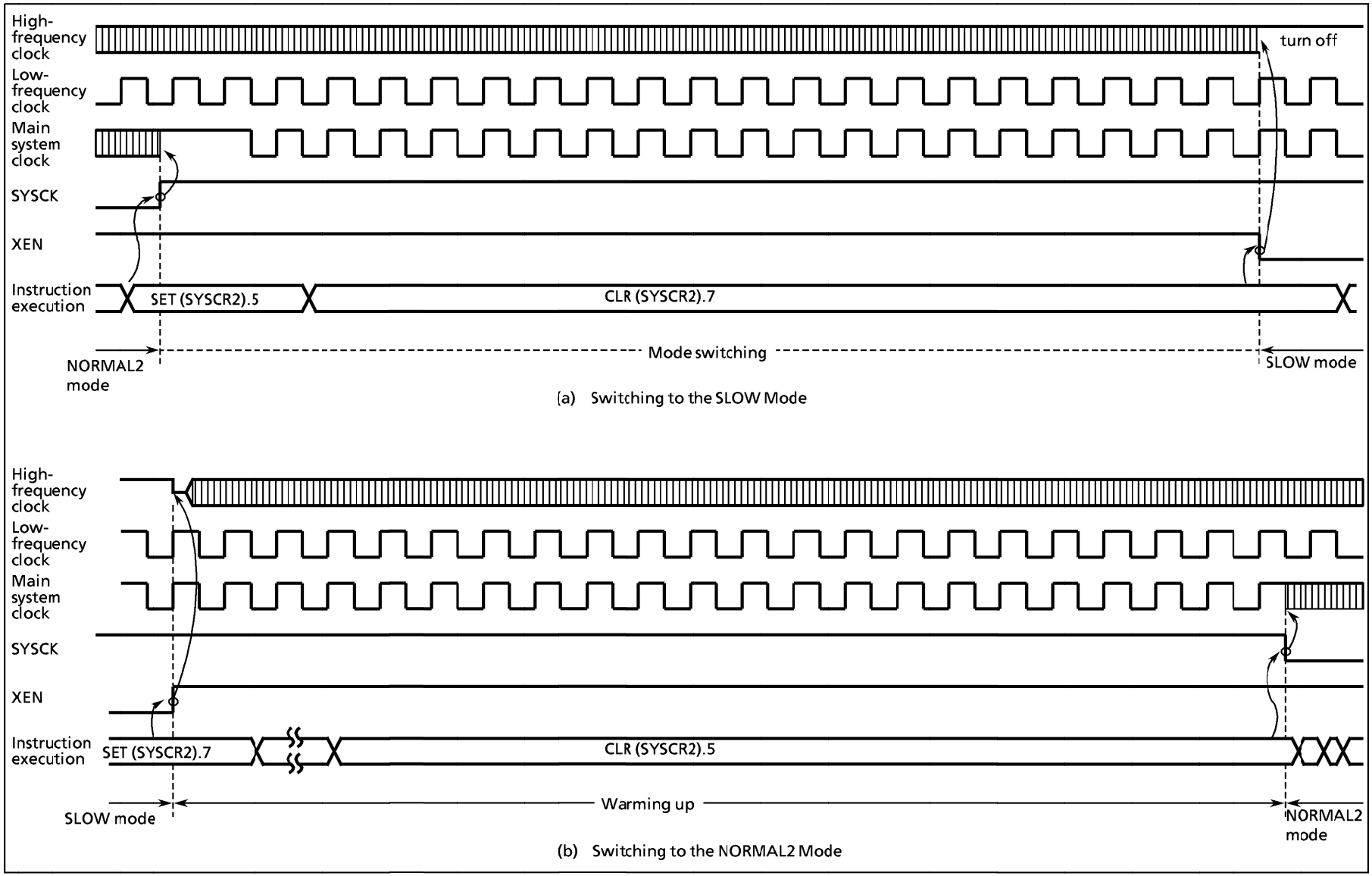



Figure 1-21. Switching between the NORMAL2 and SLOW Modes

## 1.9 Interrupt Controller

The 87C841/CC41/CH41/CK41/CM41 each have a total of 15 interrupt sources: 6 externals and 9 internals. Nested interrupt control with priorities is also possible. Two of the internal sources are pseudo non-maskable interrupts; the remainder are all maskable interrupts.

Interrupt latches (IL) that hold the interrupt requests are provided for interrupt sources. Each interrupt vector is independent.

The interrupt latch is set to "1" when an interrupt request is generated and requests the CPU to accept the interrupt. The acceptance of maskable interrupts can be selectively enabled and disabled by the program using the interrupt master enable flag (IMF) and the individual interrupt enable flags (EF). When two or more interrupts are generated simultaneously, the interrupt is accepted in the highest priority order as determined by the hardware. Figure 1-22 shows the interrupt controller.

| Interrupt Source | | | Enable Condition | Interrupt Latch | Vector Table Address | Priority |
|---|---|---|---|---|---|---|
| Internal/ External | (Reset) | | Non-Maskable | — | $FFFE_H$ | High 0 |
| Internal | INTSW | (Software interrupt) | Pseudo non-maskable | — | $FFFC_H$ | 1 |
| Internal | INTWDT | (Watchdog Timer interrupt) | | $IL_2$ | $FFFA_H$ | 2 |
| External | INT0 | (External interrupt 0) | IMF = 1, INT0EN = 1 | $IL_3$ | $FFF8_H$ | 3 |
| Internal | INTTC1 | (16-bit TC1 interrupt) | IMF · $EF_4$ = 1 | $IL_4$ | $FFF6_H$ | 4 |
| External | INT1 | (External interrupt 2) | IMF · $EF_5$ = 1 | $IL_5$ | $FFF4_H$ | 5 |
| Internal | INTTBT | (Time Base Timer interrupt) | IMF · $EF_6$ = 1 | $IL_6$ | $FFF2_H$ | 6 |
| External | INT2 | (External interrupt 2) | IMF · $EF_7$ = 1 | $IL_7$ | $FFF0_H$ | 7 |
| Internal | INTTC3 | (8-bit TC3 interrupt) | IMF · $EF_8$ = 1 | $IL_8$ | $FFEE_H$ | 8 |
| Internal | INTSIO1 | (Serial Interface 1 interrupt) | IMF · $EF_9$ = 1 | $IL_9$ | $FFEC_H$ | 9 |
| Internal | INTTC4 | (8-bit TC4 interrupt) | IMF · $EF_{10}$ = 1 | $IL_{10}$ | $FFEA_H$ | 10 |
| External | INT3 | (External interrupt 3) | IMF · $EF_{11}$ = 1 | $IL_{11}$ | $FFE8_H$ | 11 |
| External | INT4 | (External interrupt 4) | IMF · $EF_{12}$ = 1 | $IL_{12}$ | $FFE6_H$ | 12 |
| Internal | INTSIO2 | (Serial Interface 2 interrupt) | IMF · $EF_{13}$ = 1 | $IL_{13}$ | $FFE4_H$ | 13 |
| Internal | INTTC2 | (16-bit TC2 interrupt) | IMF · $EF_{14}$ = 1 | $IL_{14}$ | $FFE2_H$ | 14 |
| External | INT5 | (External interrupt 5) | IMF · $EF_{15}$ = 1 | $IL_{15}$ | $FFE0_H$ | 15 |

### (1) Interrupt Latches ($IL_{15\ to\ 2}$)

Interrupt latches are provided for each source, except for a software interrupt. The latch is set to "1" when an interrupt request is generated, and requests the CPU to accept the interrupt. The latch is cleared to "0" just after the interrupt is accepted. All interrupt latches are initialized to "0" during reset.

The interrupt latches are assigned to addresses $003C_H$ and $003D_H$ in the SFR. Each latch can be cleared to "0" individually by an instruction; however, the read-modify-write instruction such as bit manipulation or operation instructions cannot be used (Do not clear the $IL_2$ for a watchdog timer interrupt to "0"). Thus, interrupt requests can be cancelled and initialized by the program. Note that interrupt latches cannot be set to "1" by any instruction.

The contents of interrupt latches can be read out by an instruction. Therefore, testing interrupt requests by software is possible.

```
Example 1 : Clears interrupt latches
            LDW     (IL), 1110100000111111B     ; IL12, IL10 to IL6←0
Example 2 : Reads interrupt latches
            LD      WA, (IL)                    ; W←ILH, A←ILL
Example 3: Tests an interrupt latch
            TEST    (IL).7                      ; if IL7 = 1 then jump
            JR      F, SSET
```

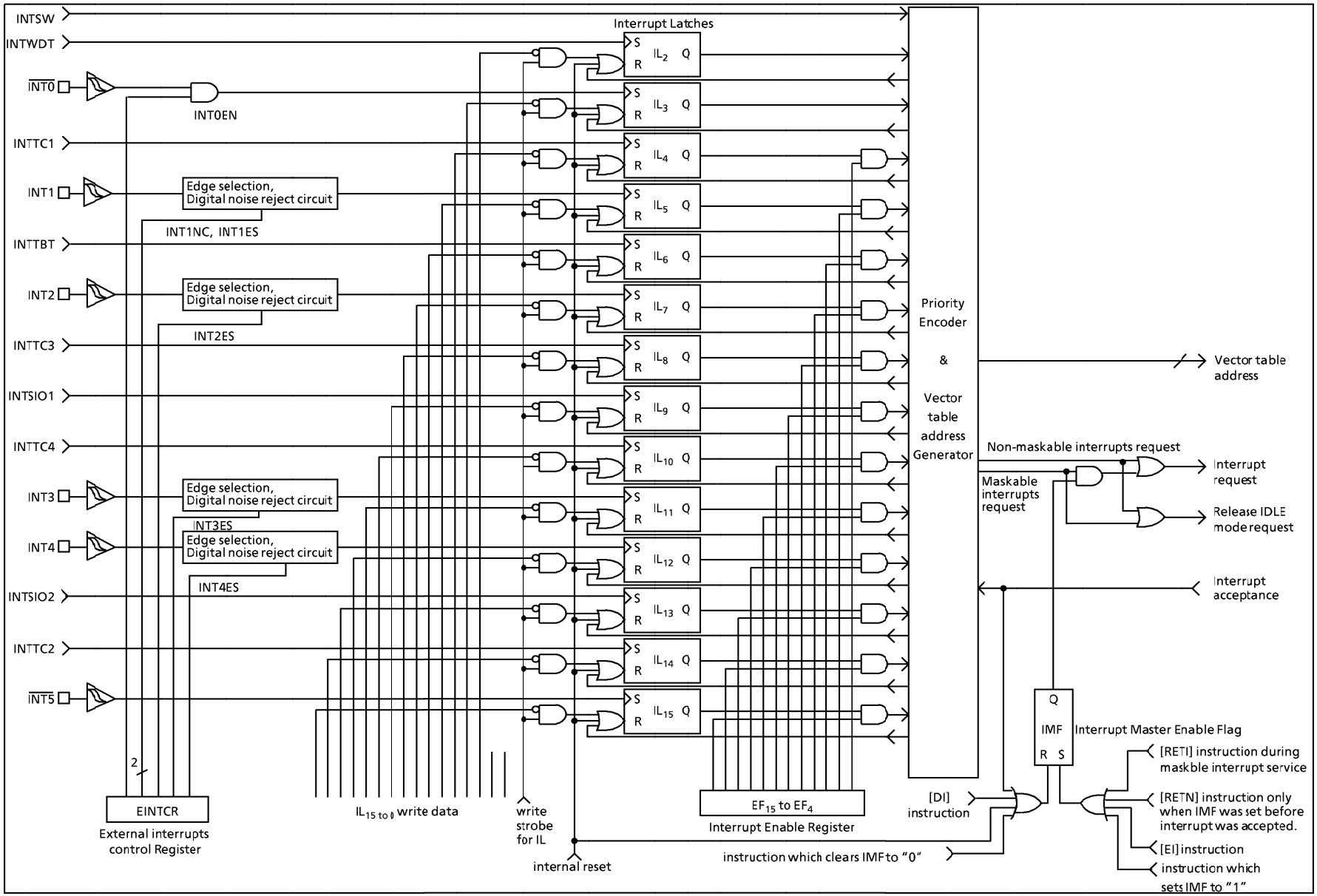



Figure 1-22. Interrupt Controller Block Diagram

(2) **Interrupt Enable Register** (EIR)

The interrupt enable registers (EIR) enable and disable the acceptance of interrupts except for the pseudo non-maskable interrupts (software and watchdog timer interrupts). Pseudo non-maskable interrupts are accepted regardless of the contents of the EIR; however, the pseudo non-maskable interrupts cannot be nested more than once at the same time. For example, the watchdog timer interrupt is not accepted during the software interrupt service.

The EIR consists of an interrupt master enable flag (IMF) and individual interrupt enable flags (EF). These registers are assigned to addresses $003A_H$ and $003B_H$ in the SFR, and can be read and written by an instruction (including read-modify-write instructions such as bit manipulation instructions).

① **Interrupt Master enable Flag** (IMF)

The interrupt master enable flag (IMF) enables and disables the acceptance of all interrupts, except for pseudo non-maskable interrupts. Clearing this flag to "0" disables the acceptance of all maskable interrupts. Setting to "1" enables the acceptance of interrupts.

When an interrupt is accepted, this flag is cleared to "0" to temporarily disable the acceptance of maskable interrupts. After execution of the interrupt service program, this flag is set to "1" by the maskable interrupt return instruction [RETI] to again enable the acceptance of interrupts. If an interrupt request has already been occurred, interrupt service starts immediately after execution of the [RETI] instruction.

Pseudo non-maskable interrupts are returned by the [RETN] instruction. In this case, the IMF is set to "1" only when pseudo non-maskable interrupt service is started with interrupt acceptance enabled (IMF = 1). Note that IMF remains "0" when cleared by the interrupt service program.

The IMF is assigned to bit 0 at address $003A_H$ in the SFR, and can be read and written by an instruction. IMF is normally set and cleared by the [EI] and [DI] instructions, and the IMF is initialized to "0" during reset.

*Note : Do not set IMF to "1"during non-maskable interrupt service programs.*

② **Individual interrupt Enable Flags** ($EF_{15}$ to $EF_4$)

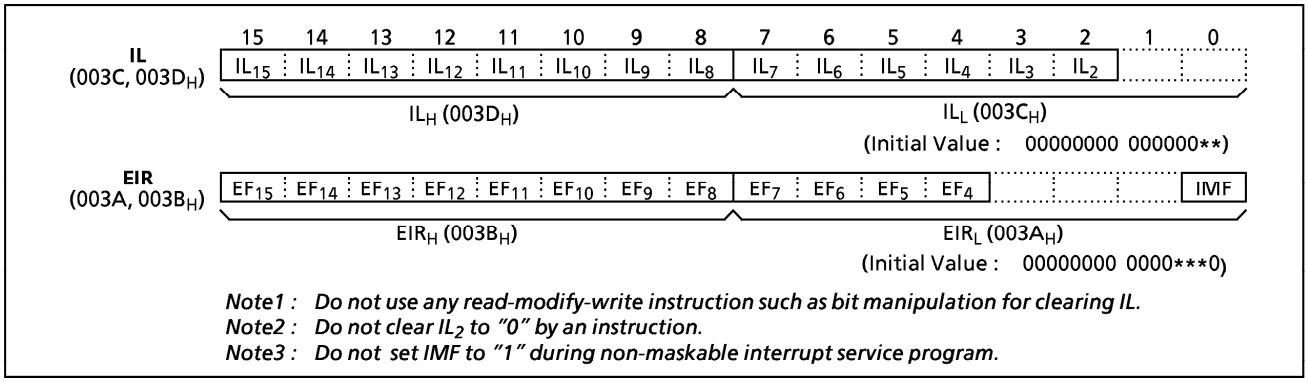
These flags enable and disable the acceptance of individual maskable interrupts, except for an external interrupt 0. Setting the corresponding bit of an individual interrupt enable flag to "1" enables acceptance of an interrupt, setting the bit to "0" disables acceptance.

Example 1 : Sets EF for individual interrupt enable, and sets IMF to "1".

```
LDW     (EIR), 1110100010100001B    ;  EF15~EF13, EF11, EF7, EF5, IMF←1
```

Example 2 : Sets an individual interrupt enable flag to "1".

```
SET     (EIRH).4                    ;  EF12←1
```




*Note1 : Do not use any read-modify-write instruction such as bit manipulation for clearing IL.*
*Note2 : Do not clear $IL_2$ to "0" by an instruction.*
*Note3 : Do not set IMF to "1" during non-maskable interrupt service program.*

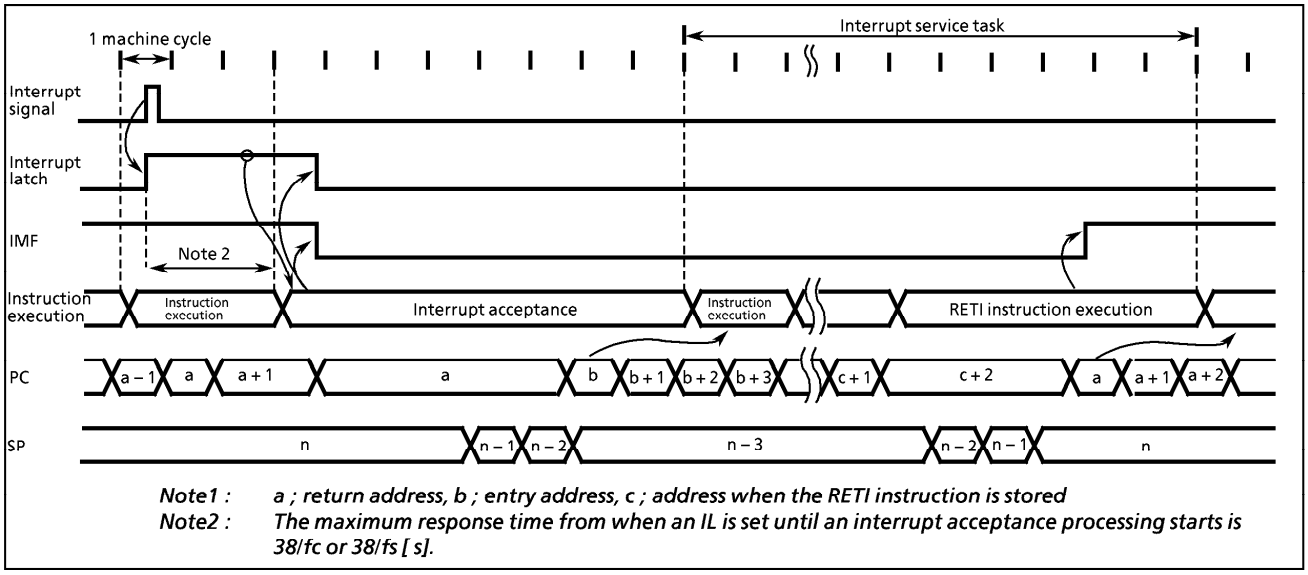
Figure 1-23. Interrupt Latch (IL) and Interrupt Enable Register (EIR)

### 1.9.1 Interrupt Sequence

An interrupt request is held until the interrupt is accepted or the interrupt latch is cleared to "0" by a reset or an instruction. Interrupt acceptance sequence requires 8 machine cycles (4 $\mu$s @ fc = 8 MHz in NORMAL mode) after the completion of the current instruction execution. The interrupt service task terminates upon execution of an interrupt return instruction [RETI] (for maskable interrupts) or [RETN] (for pseudo non-maskable interrupts).

Interrupt acceptance processing is as follows:

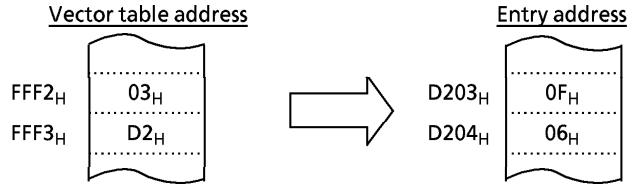
① The interrupt master enable flag (IMF) is cleared to "0" to temporarily disable the acceptance of any following maskable interrupts. When a non-maskable interrupt is accepted, the acceptance of any following interrupts is temporarily disabled.
② The interrupt latch (IL) for the interrupt source accepted is cleared to "0".
③ The contents of the program counter (return address) and the program status word are saved (pushed) onto the stack.
④ The entry address of the interrupt service program is read from the vector table address, and the entry address is loaded to the program counter.
⑤ The instruction stored at the entry address of the interrupt service program is executed.




*Note1 : a ; return address, b ; entry address, c ; address when the RETI instruction is stored*
*Note2 : The maximum response time from when an IL is set until an interrupt acceptance processing starts is 38/fc or 38/fs [ s].*

Figure 1-24. Timing Chart of Interrupt Acceptance and Interrupt Return Instruction

Example : Correspondence between vector table address for INTTBT and the entry address of the interrupt service program.




A maskable interrupt is not accepted until the IMF is set to "1" even if a maskable interrupt of higher priority than that of the current interrupt being serviced.
When nested interrupt service is necessary, the IMF is set to "1" in the interrupt service program. In this case, acceptable interrupt sources are selectively enabled by the individual interrupt enable flags. However, an acceptance of external interrupt 0 cannot be disabled by the EF; therefore, if disablement is necessary, either the external interrupt function of the $\overline{\text{INT0}}$ pin must be disabled with INT0EN in the external interrupt control register (EINTCR) or interrupt processing must be avoided by the program.

Example 1 : Disables an external interrupt 0 using INT0EN:

```
LD      (EINTCR), 00000000B ;  INT0EN←0
```

Example 2 : Disables the processing of external interrupt 0 under the software control (using bit 0 at address $00F0_H$ as the interrupt processing disable switch):

```
PINT0 :   TEST    (00F0H) . 0     ;  Returns without interrupt processing if (00F0H)0 = 1
          JRS     T, SINT0
          RETI
SINT0 :   Interrupt processing
          RETI
            :
VINT0 :   DW      PINT0
```

During interrupt acceptance processing, the program counter and the program status word are automatically saved on the stack, but not the accumulator and other registers. These registers are saved by the program if necessary. Also, when nesting multiple interrupt services, it is necessary to avoid using the same data memory area for saving registers.
The following method is used to save/restore the general-purpose registers:

① General-purpose register save/restore by register bank changeove:
General-purpose registers can be saved at high-speed by switching to a register bank that is not in use. Normally, bank 0 is used for the main task and banks 1 to 15 are assigned to interrupt service tasks. To increase the efficiency of data memory utilization, the same bank is assigned for interrupt sources which are not nested.
The switched bank is automatically restored by executing an interrupt return instruction [RETI] or [RETN]. Therefore, it is not necessary for a program to save the RBS.

Example : Register Bank Changeover

```
PINTxx :  LD      RBS, n          ;  Switches to bank n (1μs at 8MHz)
          interrupt processing
          RETI                    ;  Restores bank and Returns
```

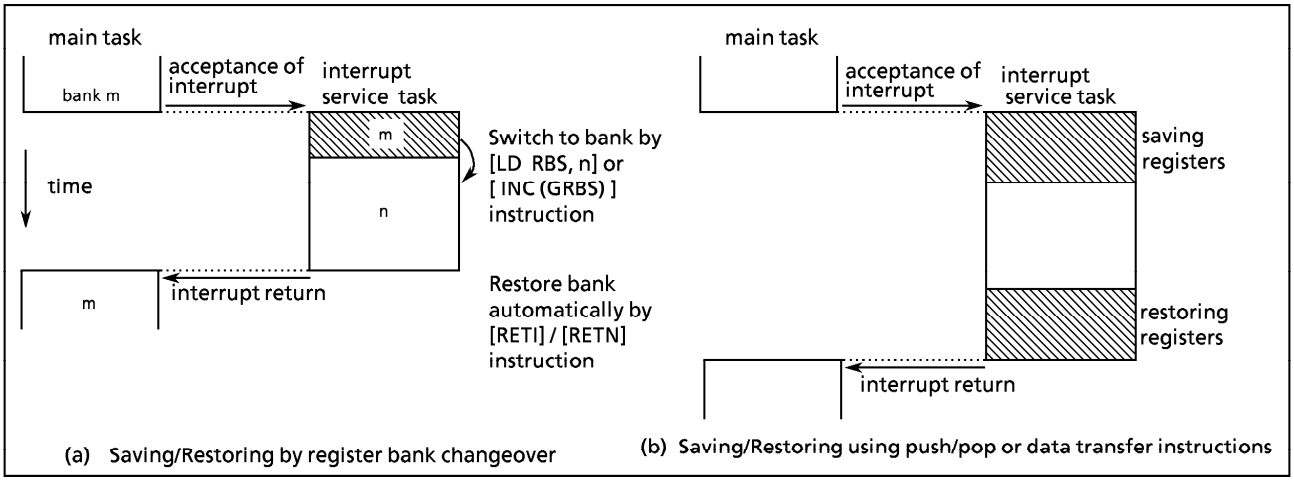
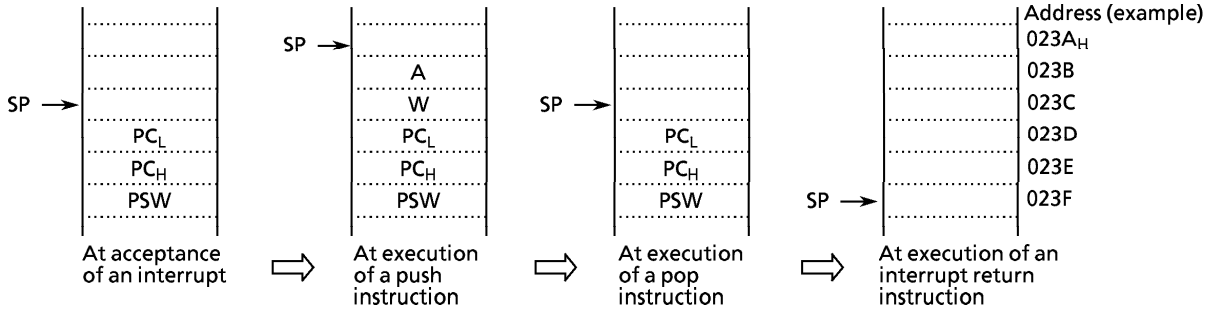



Figure 1-25. Saving/Restoring General-purpose Registers

② General-purpose register save/restore using push and pop instructions:
To save only a specific register, and when the same interrupt source occurs more than once, the general-purpose registers can be saved/restored using push/pop instructions.

Example : Register save using push and pop instructions

```
PINTxx :    PUSH    WA          ; Save WA register pair
            interrupt processing
            POP     WA          ; Restore WA register pair
            RETI                ; Return
```




③ General-purpose registers save/restore using data transfer instruction:
Data transfer instructions can be used to save only a specific general-purpose register during processing of a single interrupt.

Example : Saving/restoring a register using data transfer instructions

```
PINTxx :    LD      (GSAVA), A  ; Save A register
            interrupt processing
            LD      A, (GSAVA)  ; Restore A register
            RETI                ; Return
```

The interrupt return instructions [RETI] / [RETN] perform the following operations.

| [RETI] Maskable interrupt return | [RETN] Non-maskable interrupt return |
|---|---|
| ① The contents of the program counter and the program status word are restored from the stack. | ① The contents of the program counter and program status word are restored from the stack. |
| ② The stack pointer is incremented 3 times. | ② The stack pointer is incremented 3 times. |
| ③ The interrupt master enable flag is set to "1". | ③ The interrupt master enable flag is set to "1" only when a non-maskable interrupt is accepted in interrupt enable status. However, the interrupt master enable flag remains at "0" when so clear by an interrupt service program. |

Interrupt requests are sampled during the final cycle of the instruction being executed. Thus, the next interrupt can be accepted immediately after the interrupt return instruction is executed.

*Note : When the interrupt processing time is longer than the interrupt request generation time, the interrupt service task is performed but not the main task.*

## 1.9.2 External Interrupts

The 87C841/CC41/CH41/CK41/CM41 each have six external interrupt inputs ($\overline{INT0}$, INT1, INT2, INT3, INT4, and $\overline{INT5}$). Four of these are equipped with digital noise rejection circuits (pulse inputs of less than a certain time are eliminated as noise). Edge selection is also possible with INT1, INT2, INT3 and INT4.

The $\overline{INT0}$/P10 pin can be configured as either an external interrupt input pin or an input/output port, and is configured as an input port during reset.

Edge selection, noise rejection control and $\overline{INT0}$/P10 pin function selection are performed by the external interrupt control register (EINTCR). When INT0EN = 0, the $IL_3$ will not be set even if the falling edge of $\overline{INT0}$ pin input is detected.

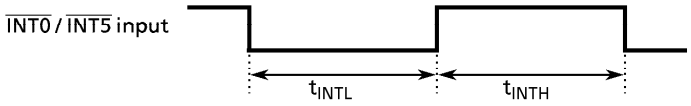
Table 1-3. External Interrupts

| Source | Pin | Secondary function pin | Enable conditions | Edge | Digital noise reject |
|---|---|---|---|---|---|
| INT0 | $\overline{INT0}$ | P10 | IMF = 1, INT0EN = 1 | falling edge | — (hysteresis input) |
| INT1 | INT1 | P11 | IMF · $EF_5$ = 1 | falling edge or rising edge | Pulses of less than 15/fc or 63/fc [s] are eliminated as noise. Pulses equal to or more than 48/fc [s] or 192/fc [s] are regarded as signals. |
| INT2 | INT2 | P12/TC1 | IMF · $EF_7$ = 1 | | Pulses of less than 7/fc [s] are eliminated as noise. Pulses equal to or more than 24/fc [s] are regarded as signals. |
| INT3 | INT3 | P50/TC3 | IMF · $EF_{11}$ = 1 | | |
| INT4 | INT4 | P51/TC4 | IMF · $EF_{12}$ = 1 | | |
| INT5 | $\overline{INT5}$ | P20/$\overline{STOP}$ | IMF · $EF_{15}$ = 1 | falling edge | — (hysteresis input) |

*Note 1 : The noise rejection function is turned off in the SLOW and SLEEP modes. Also, the noise reject times are not constant for pulses input while transiting between operating modes (NORMAL2↔SLOW)*

*Note 2 : The noise rejection function is also affected for timer/counter input (TC1 and TC3 pins).*

*Note 3 : The pulse width (both "H" and "L" level) for input to the $\overline{INT0}$ and $\overline{INT5}$ pins must be over 1 machine cycle.*




$t_{INTL}$, $t_{INTH}$ > tcyc (*Note : tcyc = 4/fm [s]*)

*Note 4 : If a noiseless signal is input to the external interrupt pin in the NORMAL 1/2 or IDLE 1/2 mode, the maximum time from the edge of input signal until the IL is set is as follows :*

① INT1 pin 49/fc [s] (INT1NC = 1), 193/fc [s] (INT1NC = 0)

② INT2,INT3, INT4 pins 25/fc [s]

*Note 5 : When high-impedance is specified for port output in stop mode, port input is forcibly fixed to low level internally. Thus, interrupt latches of external interrupt inputs except $\overline{INT5}$ (P20/$\overline{STOP}$) which are also used as ports may be set to "1". To specify high-impedance for port output in stop mode, first disable interrupt service (IMF = 0) , activate stop mode. After releasing stop mode, clear interrupt latches using load instruction, then, enable interrupt service.*

Example : Activating stop mode.

```
LD   (SYSCR1),01000000B      ; OUTEN ← 0  (specifies high-impedance)
DI                           ; IMF ← 0  (disables interrupt service)
SET (SYSCR1).STOP            ; STOP ← 1  (activates stop mode)
LDW (IL),1110011101010111B ; IL12,11,7,5,3 ← 0  (clears interrupt latches)
EI                           ; IMF ← 1  (enables interrupt service)
```

**EINTCR** ($0037_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| INT1 NC | INT0 EN | | INT4 ES | INT3 ES | INT2 ES | INT1 ES | |

(Initial value : 00*0 000*)

| | | | |
|---|---|---|---|
| INT1NC | Noise reject time select | 0 : Pulses of less than 63/fc [s] are eliminated as noise<br>1 : Pulses of less than 15/fc [s] are eliminated as noise | R/W |
| INT0EN | P10/$\overline{INT0}$ pin configuration | 0 : P10 input/output port<br>1 : $\overline{INT0}$ pin (Port P10 should be set to an input mode) | |
| INT4 ES<br>INT3 ES<br>INT2 ES<br>INT1 ES | INT4 to INT1 edge select | 0 : Rising edge<br>1 : Falling edge | |

*Note 1 : fc ; High-frequency clock [Hz] * ; don't care*

*Note 2 : Edge detection during switching edge selection is invalid.*

*Note 3 : Do not change EINTCR when IMF = 1. After changing EINTCR, interrupt latches of external interrupt inputs must be cleared to "0" using load instruction.*

*Note 4 : In order to change of external interrupt input by rewriting the contents of INT2ES, INT3ES and INT4ES during NORMAL1/2 mode, clear interrupt latches of external interrupt inputs (INT2, INT3 and INT4) after 8 machine cycles from the time of rewriting. During SLOW mode, 3 machine cycles are required.*

*Note 5 : In order to change an edge of timer counter input by rewriting the contents of INT2ES, INT3ES and INT4ES during NORMAL1/2 mode, rewrite the contents after timer counter is stopped (TC*s = 0), that is, interrupt disable state. Then, clear interrupt laches of external interrupt inputs (INT2, INT3 and INT4) after 8 machine cycles from the time of rewriting to change to interrupt enable state. Finally, start timer counter. During SLOW mode, 3 machine cycles are required.*

Example : When changing TC1 pin inputs edge in external trigger timer mode from rising edge to falling edge.

```
                LD (TC1CR),01001000B    ; TC1S ← 00 (stop TC1)
                DI                      ; IMF ← 0 (disable interrupt service)
                LD (EINTCR),00000100B   ; INT2ES ← 1 (change edge selection)
                NOP
8 machine       to
  cycles        NOP
                LD (ILL),01111111B      ; IL7 ← 0 (clear interrupt latch)
                EI                      ; IMF ← 1 (enable interrupt service)
                LD (TC1CR),01111000B    ; TC1S ← 11 (start TC1)
```

*Note 6 : If changing the contents of INT1ES during NORMAL1/2 mode, interrupt latch of external interrupt input INT1 must be cleared after 14 machine cycles (when INT1NC = 1) or 50 machine cycles (when INT1NC = 0) from the time of changing. During SLOW mode, 3 machine cycles are required.*

Figure 1-26. External Interrupt Control Register

### 1.9.3 Software Interrupt (INTSW)

Executing the [SWI] instruction generates a software interrupt and immediately starts interrupt processing (INTSW is highest prioritized interrupt). However, if processing of a non-maskable interrupt is already underway, executing the SWI instruction will not generate a software interrupt but will result in the same operation as the [NOP] instruction. Thus, the [SWI] instruction behaves like the [NOP] instruction.

Use the [SWI] instruction only for detection of the address error or for debugging.

① Address Error Detection

$FF_H$ is read if for some cause such as noise the CPU attempts to fetch an instruction from a non-existent memory address. Code $FF_H$ is the SWI instruction, so a software interrupt is generated and an address error is detected. The address error detection range can be further expanded by writing $FF_H$ to unused areas of the program memory. Address trap reset is generated for instruction fetch from a specific address (0000 to $043F_H$).

*Note : The fetch data from addresses $7F80_H$ to $7FFF_H$ (test ROM area) for 87CK41/CM41, $BF80_H$ to $BFFF_H$ for 87C841/CC41/CH41 is not "$FF_H$".*

② Debugging

Debugging efficiency can be increased by placing the SWI instruction at the software break point setting address.

## 1.10 Watchdog Timer (WDT)

The watchdog timer rapidly detects the CPU malfunction such as endless looping caused by noise or the like, and resumes the CPU to the normal state.

The watchdog timer signal for detecting malfunction can be selected either a reset output or a non-maskable interrupt request. However, selection is possible only once after reset. At first the reset output is selected.

When the watchdog timer is not being used for malfunction detection, it can be used as a timer to generate an interrupt at fixed intervals.

### 1.10.1 Watchdog Timer Configuration

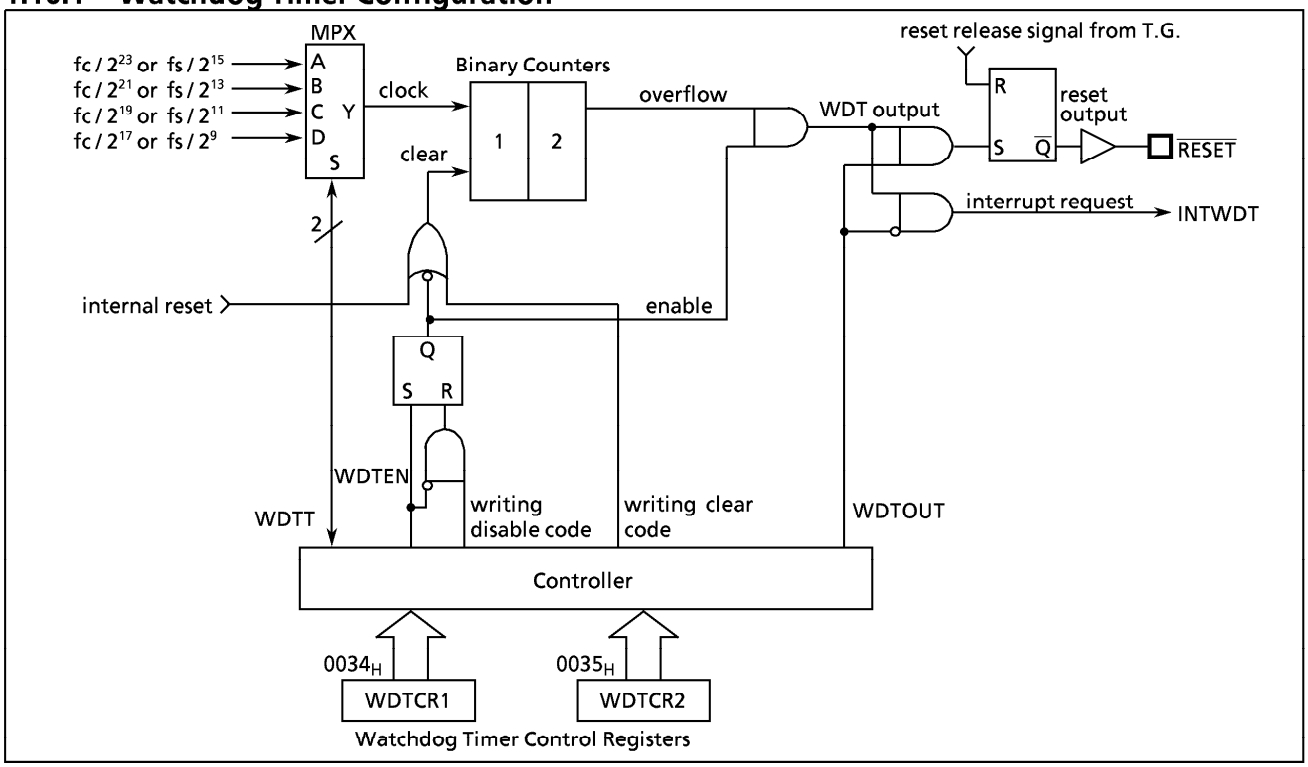



Figure 1-27. Watchdog Timer Configuration

### 1.10.2 Watchdog Timer Control

Figure 1-28 shows the watchdog timer control registers (WDTCR1, WDTCR2). The watchdog timer is automatically enabled after reset.

(1) Malfunction detection methods using the watchdog timer

The CPU malfunction is detected as follows.

① Setting the detection time, selecting output, and clearing the binary counter.

② Repeatedly clearing the binary counter within the setting detection time.

If the CPU malfunction occurs for any cause, the watchdog timer output will become active at the rising of an overflow from the binary counters unless the binary counters are cleared. At this time, when WDTOUT = 1 a reset is generated, which drives the $\overline{\text{RESET}}$ pin low to reset the internal hardware and the external circuits. When WDTOUT = 0, a watchdog timer interrupt (INTWDT) is generated.

The watchdog timer temporarily stops counting in the STOP mode including warm-up or IDLE mode, and automatically restarts (continues counting) when the STOP/IDLE mode is released.

Example : Sets the watchdog timer detection time to $2^{21}$/fc [s] and resets the CPU malfunction.

```
        LD      (WDTCR1), 00001101B   ; WDTT←10, WDTOUT←1
        LD      (WDTCR2), 4EH         ; Clears the binary counters
Within WDT                            (always clear immediately after changing WDTT)
detection time
        LD      (WDTCR2), 4EH         ; Clears the binary counters
Within WDT
detection time
        LD      (WDTCR2), 4EH         ; Clears the binary counters
```

Watchdog Timer Control Register 1

**WDTCR1** ($0034_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | WDT EN | WDTT | | WDT OUT |

(Initial value : **** 1001)

| | | | |
|---|---|---|---|
| WDTEN | Watchdog timer enable/disable | 0 : Disable (It is necessary to write the disable code to WDTCR2)<br>1 : Enable | write only |
| WDTT | Watchdog timer detection time | 00 : $2^{25}$ / fc or $2^{17}$ / fs [s]<br>01 : $2^{23}$ / fc or $2^{15}$ / fs<br>10 : $2^{21}$ / fc or $2^{13}$ / fs<br>11 : $2^{19}$ / fc or $2^{11}$ / fs | |
| WDTOUT | Watchdog timer output select | 0 : Interrupt request<br>1 : Reset output | |

*Note 1 : WDTOUT cannot be set to "1" by program after clearing WDTOUT to "0".*
*Note 2 : fc ; High-frequency clock [Hz] fs ; Low-frequency clock [Hz] * ; don't care*
*Note 3 : WDTCR1 is a write-only register and must not be used with any of read-modify-write instructions.*
*Note 4 : Disable the watchdog timer or clear the counter just before switching to STOP mode.*
*When the counter is cleared just before switching to STOP mode, clear the counter again subsequently to releasing STOP mode.*

Watchdog Timer Control Register 2

**WDTCR2** ($0035_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

(Initial value : **** ****)

| | | | |
|---|---|---|---|
| WDTCR2 | Watchdog timer control code write register | $4E_H$ : Watchdog timer binary counter clear (clear code)<br>$B1_H$ : Watchdog timer disable (disable code)<br>others : Invalid | write only |

*Note 1 : The disable code is invalid unless written when WDTEN = 0.*
*Note 2 : * ; don't care*

Figure 1-28. Watchdog Timer Control Registers

Table 1-4. Watchdog Timer Detection Time

| Operating mode | | | Detection time | |
|---|---|---|---|---|
| NORMAL1 | NORMAL2 | SLOW | At fc = 8 MHz | At fs = 32.768 kHz |
| $2^{25}$ / fc [s] | $2^{25}$ / fc, $2^{17}$ / fs | $2^{17}$ / fs | 4.194 s | 4 s |
| $2^{23}$ / fc | $2^{23}$ / fc, $2^{15}$ / fs | $2^{15}$ / fs | 1.048 ms | 1 s |
| $2^{21}$ / fc | $2^{21}$ / fc, $2^{13}$ / fs | —— | 262.1 ms | 250 ms |
| $2^{19}$ / fc | $2^{19}$ / fc, $2^{11}$ / fs | —— | 65.5 ms | 62.5 ms |

(2) Watchdog Timer Enable

The watchdog timer is enabled by setting WDTEN (bit 3 in WDTCR1) to "1". WDTEN is initialized to "1" during reset, so the watchdog timer operates immediately after reset is released.

Example : Enables watchdog timer

```
LD      (WDTCR1), 00001000B     ; WDTEN←1
```

(3) Watchdog Timer Disable

The watchdog timer is disabled by writing the disable code ($B1_H$) to WDTCR2 after clearing WDTEN (bit 3 in WDTCR1) to "0". The watchdog timer is not disabled if this procedure is reversed and the disable code is written to WDTCR2 before WDTEN is cleared to "0". The watchdog timer is halted temporarily in STOP mode (including warm-up) and IDLE mode, and restarts automatically after STOP or IDLE mode is released.

During disabling the watchdog timer, the binary counters are cleared to "0".

Example : Disables watchdog timer

```
LDW     (WDTCR1) , 0B101H       ; WDTEN←0, WDTCR2←disable code
```

### 1.10.3 Watchdog Timer Interrupt (INTWDT)

This is a pseudo non-maskable interrupt which can be accepted regardless of the contents of the EIR. If a watchdog timer interrupt or a software interrupt is already accepted, however, the new watchdog timer interrupt waits until the previous interrupt processing is completed (the end of the [RETN] instruction execution).

The stack pointer (SP) should be initialized before using the watchdog timer output as an interrupt source with WDTOUT.

Example : Watchdog timer interrupt setting up.

```
LD      SP, 013FH               ; Sets the stack pointer
LD      (WDTCR1) , 00001000B    ; WDTOUT←0
```

### 1.10.4 Watchdog Timer Reset

If the watchdog timer output becomes active, a reset is generated, which drives the $\overline{\text{RESET}}$ pin (sink open drain output) low to reset the internal hardware and the external circuits. The reset output time is $2^{20}$/fc [s] (131 ms at fc = 8MHz). The high-frequency clock oscillator also turns on when a watchdog timer reset is generated in SLOW mode.

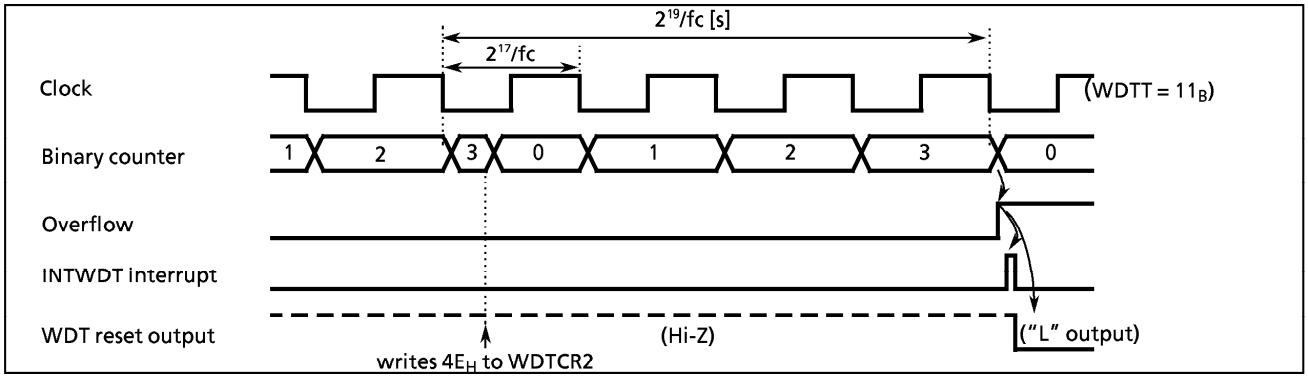



Figure 1-29. Watchdog Timer Interrupt / Reset

## 1.11 Reset Circuit

The 87C841/CC41/CH41/CK41/CM41 each have four types of reset generation procedures: an external reset input, an address trap reset, a watchdog timer reset and a system clock reset. Table 1-5 shows on-chip hardware initialization by reset action. The internal source reset circuit (watchdog timer reset, address trap reset, and system clock reset) is not initialized when power is turned on. Thus, output from the $\overline{\text{RESET}}$ pin may go low ($2^{20}$/fc [s] (131 ms at 8MHz) when power is turned on.

Table 1-5. Initializing Internal Status by Reset Action

| On-chip Hardware | | Initial Value | On-chip Hardware | Initial Value |
|---|---|---|---|---|
| Program counter | (PC) | $(FFFF_H) \cdot (FFFE_H)$ | Divider of Timing generator | 0 |
| Register bank selector | (RBS) | 0 | Watchdog timer | Enable |
| Jump status flag | (JF) | 1 | Output latches of I/O ports | Refer to I/O port circuitry |
| Interrupt master enable flag | (IMF) | 0 | Control registers | Refer to each of control register |
| Interrupt individual enable flags | (EF) | 0 | | |
| Interrupt latches | (IL) | 0 | | |

### 1.11.1 External Reset Input

When the $\overline{RESET}$ pin is held at low for at least 3 machine cycles (12/fc [s]) with the power supply voltage within the operating voltage range and oscillation stable, a reset is applied and the internal state is initialized.
When the $\overline{RESET}$ pin input goes high, the reset operation is released and the program execution starts at the vector address stored at addresses $FFFE_H$ to $FFFF_H$.
The $\overline{RESET}$ pin contains a Schmitt trigger (hysteresis) with an internal pull-up resistor. A simple power-on-reset can be applied by connecting an external capacitor and a diode.

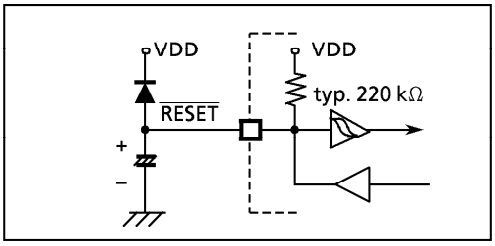
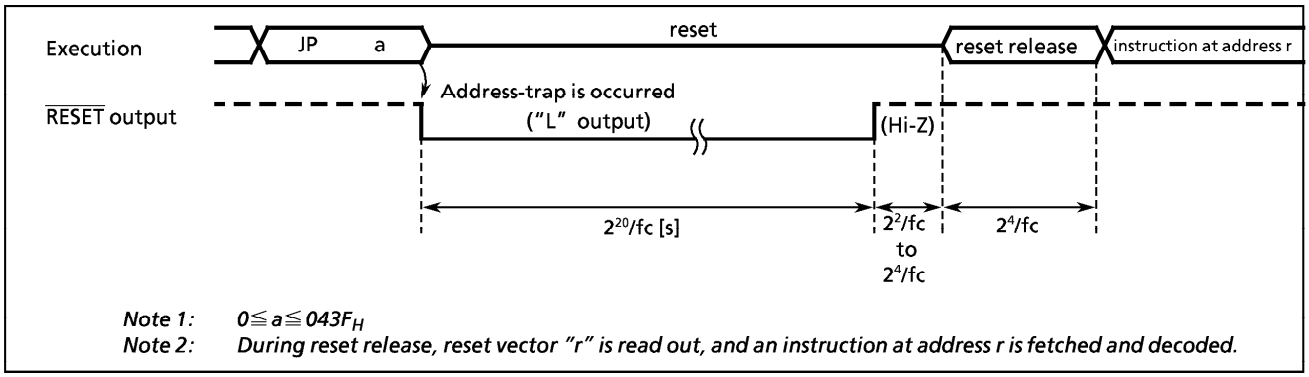



Figure 1-30. Simple Power-on-Reset Circuitry

### 1.11.2 Address-Trap-Reset

An address-trap-reset is one of fail-safe function that detects CPU malfunction such as endless looping caused by noise or the like, and returns the CPU to the normal state. If the CPU attempts to fetch an instruction from a specific address (0000 to $043F_H$), an internal reset (called address-trap-reset) will be generated. Then, the $\overline{RESET}$ pin output will go low. The reset time is $2^{20}$/fc [s] (131 ms at 8 MHz) .




*Note 1:* $0 \leqq a \leqq 043F_H$
*Note 2:* *During reset release, reset vector "r" is read out, and an instruction at address r is fetched and decoded.*

Figure 1-31. Address-Trap-Reset

### 1.11.3 Watchdog Timer Reset

Refer to Section "1.10 Watchdog Timer".

### 1.11.4 System-Clock-Reset

Clearing both XEN and XTEN (bits 7 and 6 in SYSCR2) to "0" stops both high-frequency and low-frequency oscillation, and causes the MCU to deadlock. This can be prevented by automatically generating a reset signal whenever XEN = XTEN = 0 is detected to continue the oscillation. Then, the $\overline{RESET}$ pin output goes low from high-impedance. The reset time is $2^{20}$/fc [s] (131 ms at 8MHz).

# OPERATIONAL DESCRIPTION

## 1. CPU CORE FUNCTIONS

The CPU core consists of a CPU, a system clock controller, an interrupt controller, and a watchdog timer. This section provides a description of the CPU core, the program memory (ROM), the data memory (RAM), and the reset circuit.

### 1.1 Memory Address Map

The TLCS-870 Series is capable of addressing 64K bytes of memory. Figure 1-1 shows the memory address maps of the 87C841/CC41/CH41/CK41/CM41. In the TLCS-870 Series, the memory is organized 4 address spaces (ROM, RAM, SFR, and DBR). It uses a memory mapped I/O system, and all I/O registers are mapped in the SFR/DBR address spaces. There are 16 banks of general-purpose registers. The register banks are also assigned to the first 128 bytes of the RAM address space.

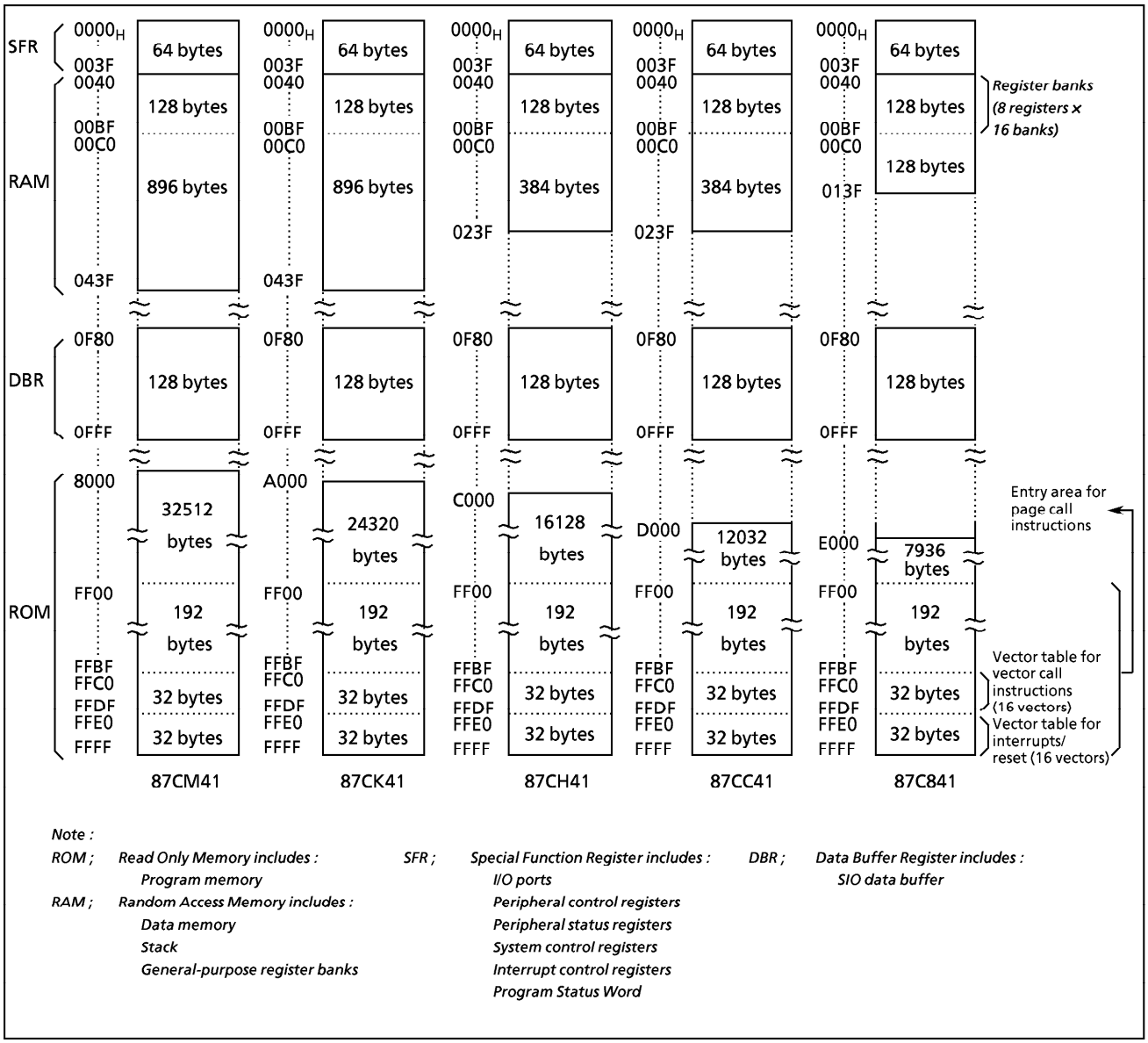



Figure 1-1. Memory Address Maps

## 1.2 Program Memory (ROM)

The 87C841 has an 8K × 8-bit (addresses $E000_H$ to $FFFF_H$), the 87CC41 has a 12K × 8-bit (addresses $D000_H$ to $FFFF_H$), the 87CH41 has a 16K × 8-bit (addresses $C000_H$ to $FFFF_H$), and the 87CK41 has a 24K × 8-bit (address $A000_H$ to $FFFF_H$), and the 87CM41 has a 32K-8-bit (address $8000_H$ to $FFFF_H$) of program memory (mask programmed ROM).
Addresses $FF00_H$ to $FFFF_H$ in the program memory can also be used for special purposes.

(1) **Interrupt / Reset** vector table (addresses $FFE0_H$ to $FFFF_H$)
This table consists of a reset vector and 15 interrupt vectors (2 bytes/vector). These vectors store a reset start address and interrupt service routine entry addresses.

(2) Vector table for **vector call** instructions (addresses $FFC0_H$ to $FFDF_H$)
This table stores call vectors (subroutine entry address, 2 bytes/vector) for the vector call instructions [CALLV n]. There are 16 vectors. The CALLV instruction increases memory efficiency when utilized for frequently used subroutine calls (called from 3 or more locations).

(3) Entry area (addresses $FF00_H$ to $FFFF_H$) for **page call** instructions
This is the subroutine entry address area for the page call instructions [CALLP n]. Addresses $FF00_H$ to $FFBF_H$ are normally used because address $FFC0_H$ to $FFFF_H$ are used for the vector tables.

Programs and fixed data are stored in the program memory. The instruction to be executed next is read from the address indicated by the current contents of the program counter (PC). There are relative jump and absolute jump instructions. The concepts of page or bank boundaries are not used in the program memory concerning any jump instruction.

Example: The relationship between the jump instructions and the PC.

① 5-bit PC-relative jump [JRS cc, $ + 2 + d]
E8C4H: JRS T, $ + 2 + 08H
When JF = 1, the jump is made to $E8CE_H$, which is $08_H$ added to the contents of the PC. (The PC contains the address of the instruction being executed + 2; therefore, in this case, the PC contents are $E8C4_H + 2 = E8C6_H$.)

② 8-bit PC-relative jump [JR cc, $ + 2 + d]
E8C4H : JR Z, $ + 2 + 80H
When ZF = 1, the jump is made to $E846_H$, which is $FF80_H$ ( − 128) added to the current contents of the PC.

③ 16-bit absolute jump [JP a]
E8C4H : JP 0C235H
An unconditional jump is made to address $C235_H$. The absolute jump instruction can jump anywhere within the entire 64K-bytes space.

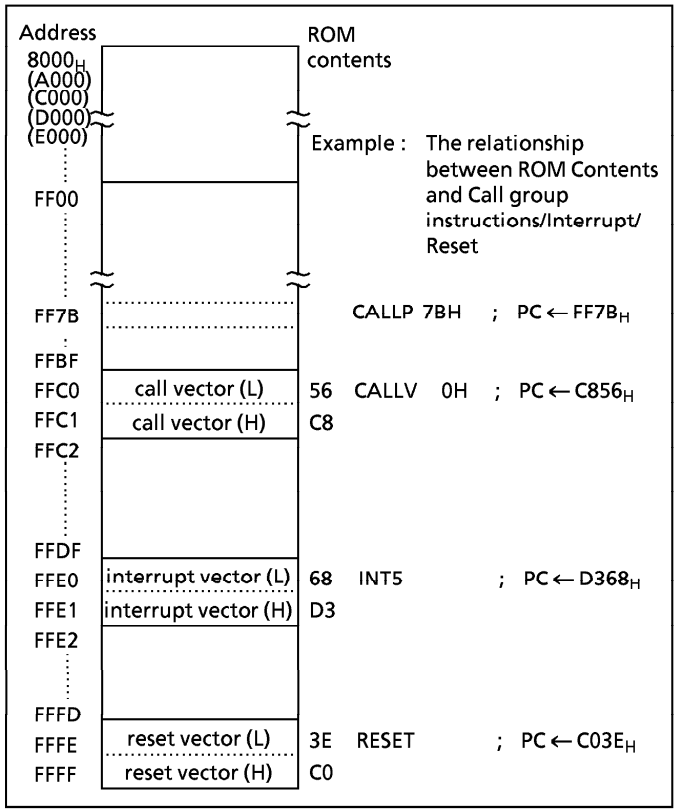



Figure 1-2. Program Memory Map

In the TLCS-870 Series, the same instruction used to access the data memory (e.g. [LD A, (HL)] ) is also used to read out fixed data (ROM data) stored in the program memory. The register-offset PC-relative addressing (PC + A) instructions can also be used, and the code conversion, table look-up and n-way multiple jump processing can easily be programmed.

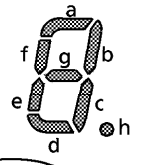
Example 1 : Loads the ROM contents at the address specified by the HL register pair contents into the accumulator (87CH41 : HL≧C000$_H$):

```
LD      A, (HL)          ; A←ROM (HL)
```

Example 2 : Converts BCD to 7-segment code (common anode LED). When A = 05$_H$, 92$_H$ is output to port P3 after executing the following program:

```
        ADD     A, TABLE – $ – 4     ; P3 ←ROM (TABLE + A)
        LD      (P3), (PC + A)
        JRS     T, SNEXT
TABLE : DB      0C0H, 0F9H, 0A4H, 0B0H, 99H, 92H, 82H, 0D8H, 80H, 98H
SNEXT :
```

*Notes : "$" is a header address of ADD instruction.*
*DB is a byte data difinition instruction.*




Example 3 : N-way multiple jump in accordance with the contents of accumulator (0≦A≦3):

```
SHLC    A                ; if A = 00H then PC←C234H
JP      (PC + A)           if A = 01H then PC←C378H
                           if A = 02H then PC←DA37H
                           if A = 03H then PC←E1B0H
DW      0C234H, 0C378H, 0DA37H, 0E1B0H
```

*Note : DW is a word data definition instruction.*

```
SHLC A
JP (PC + A)
34
C2
78
C3
37
DA
B0
E1
```

## 1.3 Program Counter (PC)

The program counter (PC) is a 16-bit register which indicates the program memory address where the instruction to be executed next is stored. After reset, the user defined reset vector stored in the vector table (addresses FFFF$_H$ and FFFE$_H$) is loaded into the PC ; therefore, program execution is possible from any desired address. For example, when C0$_H$ and 3E$_H$ are stored at addresses FFFF$_H$ and FFFE$_H$, respectively, the execution starts from address C03E$_H$ after reset.

The TLCS-870 Series utilizes pipelined processing (instruction pre-fetch); therefore, the PC always indicates 2 addresses in advance. For example, while a 1-byte instruction stored at address C123$_H$ is being executed, the PC contains C125$_H$.

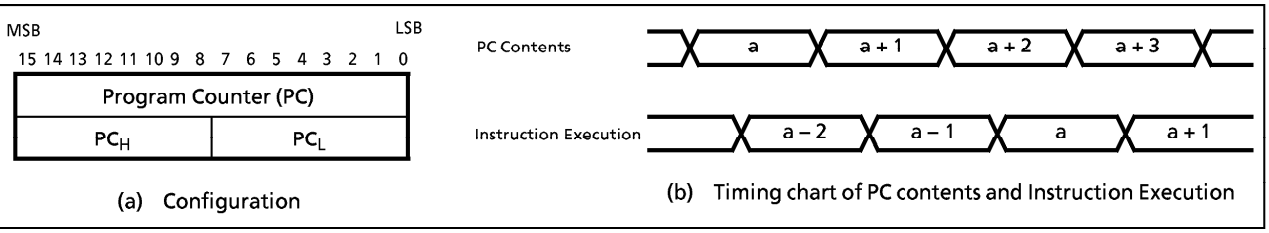



Figure 1-3. Program Counter

## 1.4 Data Memory (RAM)

The 87C841 has a 256 × 8-bit (addresses $0040_H$ to $013F_H$), the 87CC41/CH41 have a 512 × 8-bit (addresses $0040_H$ to $023F_H$), and the 87CK41/CM41 have a 1K × 8-bit (address $0040_H$ to $043F_H$) of data memory (static RAM). Figure 1-4 shows the data memory map.

Addresses $0000_H$ to $00FF_H$ are used as a direct addressing area to enhance instructions which utilize this addressing mode; therefore, addresses $0040_H$ to $00FF_H$ in the data memory can also be used for user flags or user counters. General-purpose register banks (8 registers × 16 banks) are also assigned to the 128 bytes of addresses $0040_H$ to $00BF_H$. Access as data memory is still possible even when being used for registers. For example, when the contents of the data memory at address $0040_H$ is read out, the contents of the accumulator in the bank 0 are also read out. The stack can be located anywhere within the data memory except the register bank area. The stack depth is limited only by the free data memory size. For more details on the stack, see section "1.7 Stack and Stack Pointer".

With the 87C841/CC41/CH41/CK41/CM41, programs in data memory cannot be executed. If the program counter indicates a data memory address (addresses $0040_H$ to $013F_H$ for 87C841, addresses $0040_H$ to $023F_H$ for 87CC41/CH41, addresses $0040_H$ to $043F_H$ for 87CK41/CM41) , an address-trap-reset is generated due to bus error. (Output from the $\overline{RESET}$ pin goes low. )

Example 1 : If bit 2 at data memory address $00C0_H$ is "1", $00_H$ is written to data memory at address $00E3_H$; otherwise, $FF_H$ is written to the data memory at address $00E3_H$.

```
        TEST    (00C0H).2       ;  if (00C0H)2 = 0 then jump
        JRS     T,SZERO
        CLR     (00E3H)         ;  (00E3H) ← 00H
        JRS     T,SNEXT
SZERO : LD      (00E3H), 0FFH   ;  (00E3H) ← FFH
SNEXT :
```

Example 2 : Increments the contents of data memory at address $00F5_H$, and clears to $00_H$ when $10_H$ is exceeded.

```
        INC     (00F5H)         ;  (00F5H) ← (00F5H) + 1
        AND     (00F5H), 0FH    ;  (00F5H) ← (00F5H)∧0FH
```

The data memory contents become unstable when the power supply is turned on; therefore, the data memory should be initialized by an initialization routine. Note that the general-purpuse registers are mapped in the RAM ; therefore, *do not clear RAM at the current bank addresses.*

Example 1 : Clears RAM to "$00_H$" except the bank 0 (87C841)

```
        LD      HL, 0048H       ;  Sets start address to HL register pair
        LD      WA, 0F700H      ;  Sets initial data (00H) to A register
                                ;  Sets number of byte to W register pair
SRAMCLR : LD    (HL+), A
        DEC     W
        JRS     F, SRAMCLR
```

Example 2 : Clears RAM to "$00_H$" except the bank 0 (87CC41/CH41)

```
        LD      HL, 0048H       ;  Sets start address to HL register pair
        LD      A, H            ;  Sets initial data (00H) to A register
        LD      BC, 01F7H       ;  Sets number of byte to BC register pair
SRAMCLR : LD    (HL+), A
        DEC     BC
        JRS     F, SRAMCLR
```

Example 3 : Clears RAM to "$00_H$" except the bank 0 (87CK41/CM41, 87PM41)

```
        LD      HL, 0048H       ; Sets start address to HL register pair
        LD      A, H            ; Sets initial data (00H) to A register
        LD      BC, 03F7H       ; Sets number of byte to BC register pair
SRAMCLR : LD    (HL+), A
        DEC     BC
        JRS     F, SRAMCLR
```

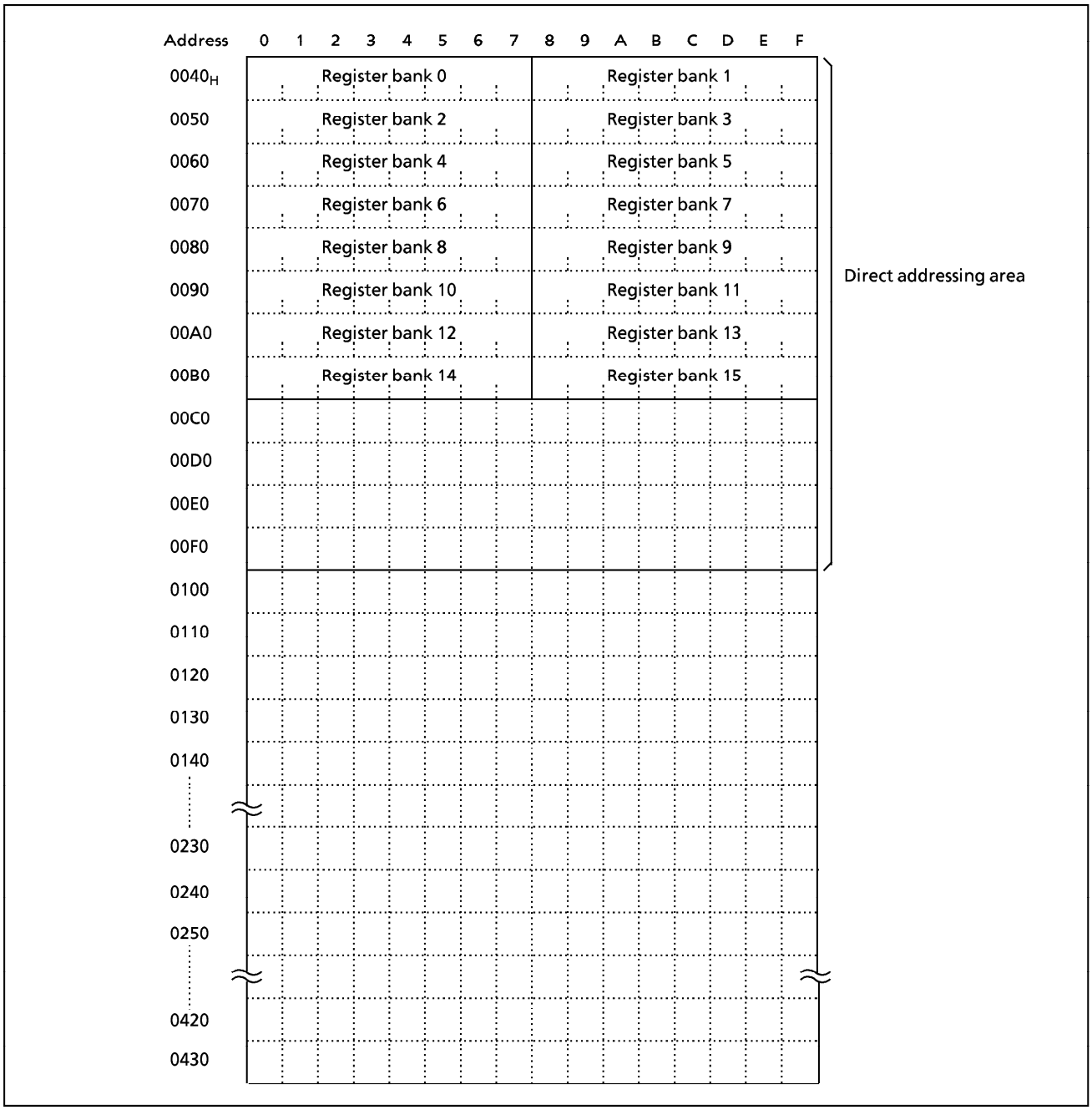



Figure 1-4. Data Memory Map

## 1.5 General-purpose Register Banks

General-purpose registers are mapped into addresses $0040_H$ to $00BF_H$ in the data memory as shown in Figure 1-4. There are 16 register banks, and each bank contains eight 8-bit registers W, A, B, C, D, E, H, and L. Figure 1-5 shows the general-purpose register bank configuration.

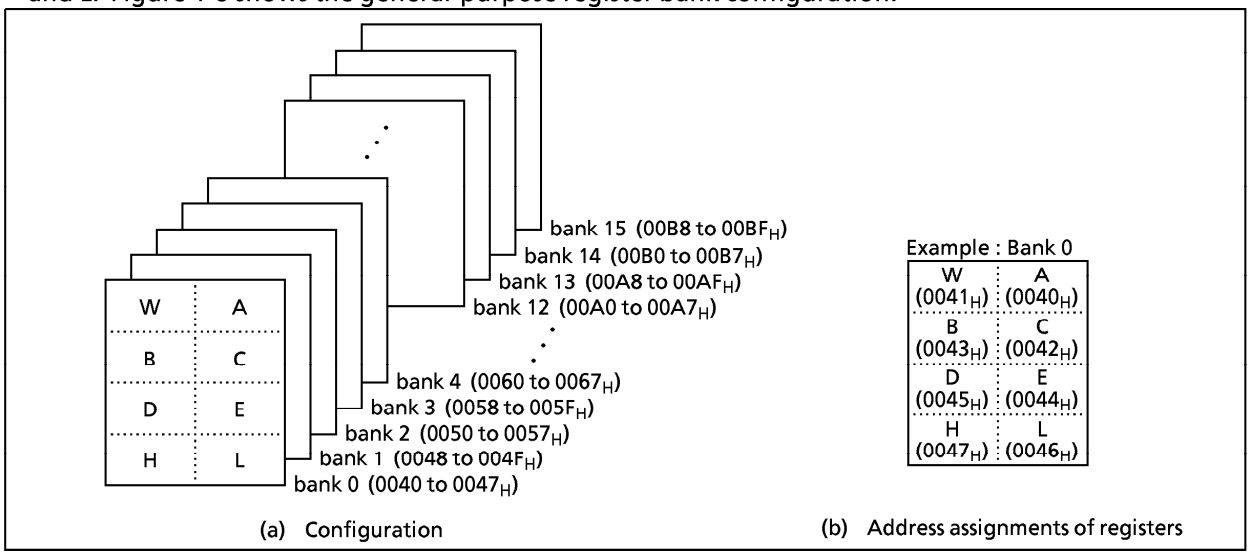



Figure 1-5. General-purpose Register Banks

In addition to access in 8-bit units, the registers can also be accessed in 16-bit units as the register pairs WA, BC, DE, and HL. Besides its function as a general-purpose register, the register also has the following functions:

(1) **A, WA**

The A register functions as an 8-bit accumulator and WA the register pair functions as a 16-bit accumulator (W is high byte and A is low byte). Registers other than A can also be used as accumulators for 8-bit operations.

```
Examples :  ①   ADD  A, B          ; Adds B contents to A contents and stores the result into A.
            ②   SUB  WA, 1234H     ; Subtracts 1234H from WA contents and stores the result into WA.
            ③   SUB  E, A          ; Subtracts A contents from E contents, and stores the result into E.
```

(2) **HL, DE**

The HL and DE specify a memory address. The HL register pair functions as data pointer (HL) /index register (HL + d) /base register (HL + C), and the DE register pair function as a data pointer (DE). The HL also has an auto-post- increment and auto-pre-decrement functions. This function simplifies multiple digit data processing, software LIFO (last-in first-out) processing, etc.

```
Example 1 :  ①   LD   A, (HL)       ; Loads the memory contents at the address specified by HL into A.
             ②   LD   A, (HL + 52H) ; Loads the memory contents at the address specified by the value
                                      obtained by adding 52H to HL contents into A.
             ③   LD   A, (HL + C)   ; Loads the memory contents at the address specified by the value
                                      obtained by adding the register C contents to HL contents into A.
             ④   LD   A, (HL + )    ; Loads the memory contents at the address specified by HL into A.
                                      Then increments HL.
             ⑤   LD   A, ( – HL)    ; Decrements HL. Then loads the memory contents at the address
                                      specified by new HL into A.
```

The TLCS-870 Series can transfer data directly memory to memory, and operate directly between memory data and memory data. This facilitates the programming of block processing.

Example 2 : Block transfer

```
        LD      B, n – 1        ; Sets (number of bytes to transfer) – 1 to B
        LD      HL, DSTA        ; Sets destination address to HL
        LD      DE, SRCA        ; Sets source address to DE
SLOOP : LD      (HL), (DE)      ; (HL) ← (DE)
        INC     HL              ; HL ← HL + 1
        INC     DE              ; DE ← DE + 1
        DEC     B               ; B ← B – 1
        JRS     F, SLOOP        ; if B ≧ 0 then loop
```

(3) **B, C, BC**

Registers B and C can be used as 8-bit buffers or counters, and the BC register pair can be used as a 16-bit buffer or counter. The C register functions as an offset register for register-offset index addressing (refer to example 1 ③ above) and as a divisor register for the division instruction [DIV gg, C].

Example 1 : Repeat processing

```
        LD      B, n            ; Sets n as the number of repetitions to B
SREPEAT : processing            ;     (n + 1 times processing)
        DEC     B
        JRS     F, SREPEAT
```

Example 2 : Unsigned integer division (16-bit ÷ 8-bit)

```
        DIV     WA, C           ; Divides the WA contents by the C contents, places the
                                ;   quotient in A and the remainder in W.
```

The general-purpose register banks are selected by the 4-bit register bank selector (RBS). During reset, the RBS is initialized to "0". The bank selected by the RBS is called the current bank.
Together with the flag, the RBS is assigned to address $003F_H$ in the SFR as the program status word (PSW). There are 3 instructions [LD RBS, n], [PUSH PSW], [POP PSW] to access the PSW. The PSW can be also operated by the memory access instruction.

Example 1 : Incrementing the RBS

```
        INC     (003FH)         ; RBS ← RBS + 1
```

Example 2 : Reading the RBS

```
        LD      A, (003FH)      ; A ← PSW (A3-0 ← RBS, A7-4 ← Flags)
```

Highly efficient programming and high-speed task switching are possible by using bank changeover to save registers during interrupt and to transfer parameters during subroutine processing.
During interrupt, the PSW is automatically saved onto the stack. The bank used before the interrupt was accepted is restored automatically by executing an interrupt return instruction [RETI]/[RETN] ; therefore, there is no need for the RBS save/restore software processing.
The TLCS-870 Series supports a maximum of 15 interrupt sources. One bank is assigned to the main program, and one bank can be assigned to each source. Also, to increase the efficiency of data memory usage, assign the same bank to interrupt sources which are not nested.

Example: Saving /restoring registers during interrupt task using bank changeover.

```
PINT1 : LD      RBS, n          ; RBS ← n (Bank changeover)
        Interrupt processing
        RETI                    ; Maskable interrupt return (Bank restoring)
```

## 1.6 Program Status Word (PSW)

The program status word (PSW) consists of a register bank selector (RBS) and four flags, and the PSW is assigned to address $003F_H$ in the SFR.

The RBS can be read and written using the memory access instruction (e. g. [LD A, (003FH)], [LD (003FH), A], however the flags can only be read. When writing to the PSW, the change specified by the instruction is made without writing data to the flags. For example, when the instruction [LD (003FH), 05H] is executed, "5" is written to the RBS and the JF is set to "1", but the other flags are not affected.

[PUSH PSW] and [POP PSW] are the PSW access instructions.

### 1.6.1 Register Bank Selector (RBS)

The register bank selector (RBS) is a 4-bit register used to select general-purpose register banks. For example, when RBS = 2, bank 2 is currently selected. During reset, the RBS is initialized to "0".

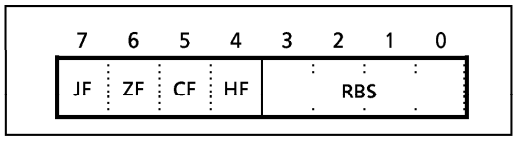



Figure 1-6. PSW (Flags, RBS) Configuration

### 1.6.2 Flags

The flags are configured with the upper 4 bits : a zero flag, a carry flag, a half carry flag and a jump status flag. The flags are set or cleared under conditions specified by the instruction. These flags except the half carry flag are used as jump condition "cc" for conditional jump instructions [JR cc, $ + 2 + d]/[JRS cc, $ + 2 + d]. After reset, the jump status flag is initialized to "1", other flags are not affected.

(1) **Zero flag (ZF)**

The ZF is set to "1" if the operation result or the transfer data is $00_H$ (for 8-bit operations and data transfers)/$0000_H$ (for 16-bit operations); otherwise the ZF is cleared to "0".

During the bit manipulation instructions [SET, CLR, and CPL], the ZF is set to "1" if the contents of the specified bit is "0"; otherwise the ZF is cleared to "0".

This flag is set to "1" when the upper 8 bits of the product are $00_H$ during the multiplication instruction [MUL], and when $00_H$ for the remainder during the division instruction [DIV]; otherwise it is cleared to "0".

(2) **Carry flag (CF)**

The CF is set to "1" when a carry out of the MSB (most significant bit) of the result occurred during addition or when a borrow into the MSB of the result occurred during subtraction; otherwise the CF is cleared to "0". During division, this flag is set to "1" when the divisor is $00_H$ (divided by zero error), or when the quotient is $100_H$ or higher (overflow error); otherwise it is cleared. The CF is also affected during the shift/rotate instructions [SHLC, SHRC, ROLC, and RORC]. The data shifted out from a register is set to the CF.

This flag is also a 1-bit register (a boolean accumulator) for the bit manipulation instructions.

Set/clear/complement are possible with the CF manipulation instructions.

Example1 : Bit manipulation

```
LD      CF, (0007H) . 5      ; (0001H)2 ← (0007H)5 ∀ (009AH)0
XOR     CF, (009AH) . 0
LD      (0001H) . 2, CF
```

Example2 : Arithmetic right shift

```
LD      CF, A . 7            ; A ← A / 2
RORC    A
```

(3) **Half carry flag (HF)**

The HF is set to "1" when a carry occurred between bits 3 and 4 of the operation result during an 8-bit addition, or when a borrow occurred from bit 4 into bit 3 of the result during an 8-bit subtraction; otherwise the HF is cleared to "0". This flag is useful in the decimal adjustment for BCD operations (adjustments using the [DAA r], or [DAS r] instructions).

Example : BCD operation
(The A becomes $47_H$ after executing the following program when A = $19_H$, B = $28_H$)

```
ADD     A, B        ; A ← 41H, HF ← 1
DAA     A           ; A ← 41H + 06H = 47H (decimal-adjust)
```

(4) **Jump status flag (JF)**

Zero or carry information is set to the JF after operation (e. g. INC, ADD, CMP, TEST).
The JF provides the jump condition for conditional jump instructions [JRS T/F, $ + 2 + d], [JR T/F, $ + 2 + d] (T or F is a condition code). Jump is performed if the JF is "1" for a true condition (T), or the JF is "0" for a false condition (F).
The JF is set to "1" after executing the load/exchange/swap/nibble rotate/jump instruction, so that [JRS T, $ + 2 + d] and [JR T, $ + 2 + d] can be regarded as an unconditional jump instruction.

Example : Jump status flag and conditional jump instruction

```
INC     A
JRS     T, SLABLE1      ; Jump when a carry is caused by the immediately
 ⋮                        preceding operation instruction.
LD      A, (HL)
JRS     T, SLABLE2      ; JF is set to "1" by the immediately preceding
 ⋮                        instruction, making it an unconditional jump
                          instruction.
```

Example : The accumulator and flags become as shown below after executing the following instructions when the WA register pair, the HL register pair, the data memory at address $00C5_H$, the carry flag and the half carry flag contents being "$219A_H$", "$00C5_H$", "$D7_H$", "1" and "0", respectively.

| Instruction | Acc. after execution | Flag after execution | | | |
|---|---|---|---|---|---|
| | | JF | ZF | CF | HF |
| ADDC A, (HL) | 72 | 1 | 0 | 1 | 1 |
| SUBB A, (HL) | C2 | 1 | 0 | 1 | 0 |
| CMP A, (HL) | 9A | 0 | 0 | 1 | 0 |
| AND A, (HL) | 92 | 0 | 0 | 1 | 0 |
| LD A, (HL) | D7 | 1 | 0 | 1 | 0 |
| ADD A, 66H | 00 | 1 | 1 | 1 | 1 |

| Instruction | Acc. after execution | Flag after execution | | | |
|---|---|---|---|---|---|
| | | JF | ZF | CF | HF |
| INC A | 9B | 0 | 0 | 1 | 0 |
| ROLC A | 35 | 1 | 0 | 1 | 0 |
| RORC A | CD | 0 | 0 | 0 | 0 |
| ADD WA, 0F508H | 16A2 | 1 | 0 | 1 | 0 |
| MUL W, A | 13DA | 0 | 0 | 1 | 0 |
| SET A.5 | BA | 1 | 1 | 1 | 0 |

## 1.7 Stack and Stack Pointer

### 1.7.1 Stack

The stack provides the area in which the return address or status, etc. are saved before a jump is performed to the processing routine during the execution of a subroutine call instruction or the acceptance of an interrupt. On a subroutine call instruction [CALL a] / [CALLP n] / [CALLV n], the contents of the PC (the return address) is saved; on an interrupt acceptance, the contents of the PC and the PSW are saved (the PSW is pushed first, followed by $PC_H$ and $PC_L$). Therefore, a subroutine call occupies two bytes on the stack; an interrupt occupies three bytes.
When returning from the processing routine, executing a subroutine return instruction [RET] restores the contents to the PC from the stack; executing an interrupt return instruction [RETI] / [RETN] restores the contents to the PC and the PSW (the $PC_L$ is popped first, followed by $PC_H$ and PSW).
The stack can be located anywhere within the data memory space except the register bank area, therefore the stack depth is limited only by the free data memory size.

### 1.7.2 Stack Pointer (SP)

The stack pointer (SP) is a 16-bit register containing the address of the next free locations on the stack.

The SP is post-decremented when a subroutine call or a push instruction is executed, or when an interrupt is accepted; and the SP is pre-incremented when a return or a pop instruction is executed. Figure 1-8 shows the stacking order.

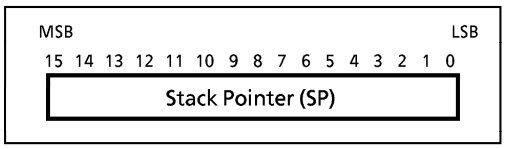



Figure 1-7. Stack Pointer

The SP is not initialized hardware-wise but requires initialization by an initialize routine (sets the highest stack address). [LD SP, mn], [LD SP, gg] and [LD gg, SP] are the SP access instructions (mn ; 16-bit immediate data, gg ; register pair).

Example 1 : To initialize the SP

```
LD      SP, 013FH       ; SP←013F_H
```

Example 2 : To read the SP

```
LD      HL, SP          ; HL←SP
```

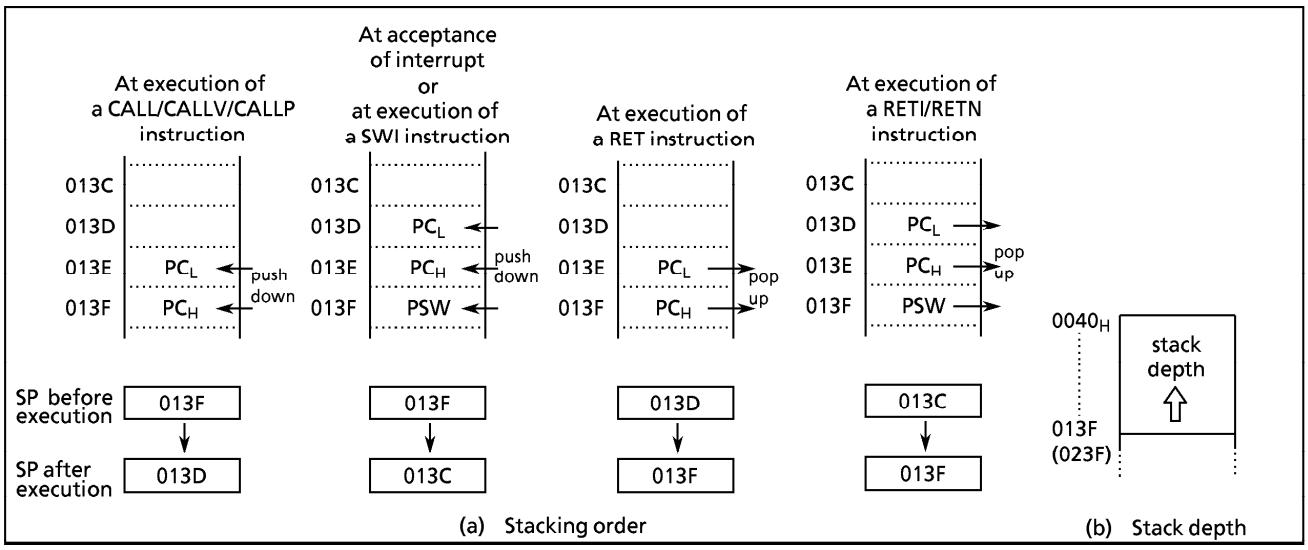



Figure 1-8. Stack

## 1.8 System Clock Controller

The system clock controller consists of a clock generator, a timing generator, and a stand-by controller.

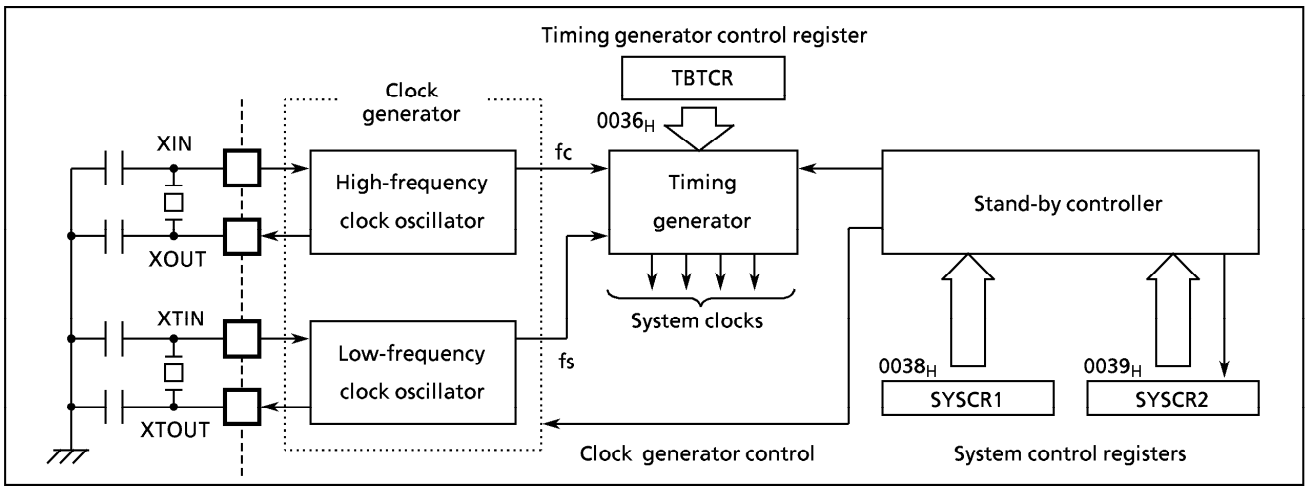



Figure 1-9. System Clock Controller

### 1.8.1 Clock Generator

The clock generator generates the basic clock which provides the system clocks supplied to the CPU core and peripheral hardware. It contains two oscillation circuits: one for the high-frequency clock and one for the low-frequency clock. Power consumption can be reduced by switching of the system clock controller to low-power operation based on the low-frequency clock.

The high-frequency (fc) and low-frequency (fs) clocks can be easily obtained by connecting a resonator between the XIN/XOUT and XTIN/XTOUT pins, respectively. Clock input from an external oscillator is also possible. In this case, external clock is applied to the XIN/XTIN pin with the XOUT/XTOUT pin not connected. The 87C841/CC41/CH41/CK41/CM41 are not provided an RC oscillation.

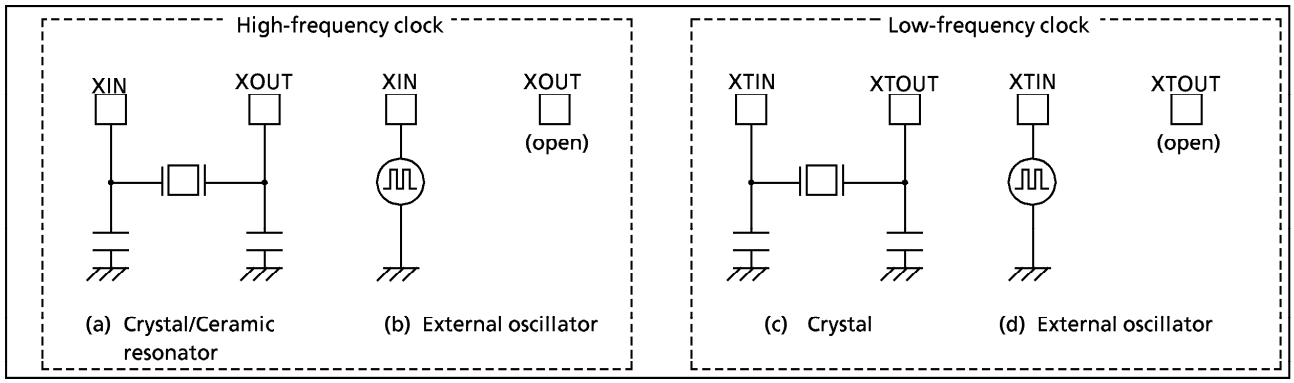
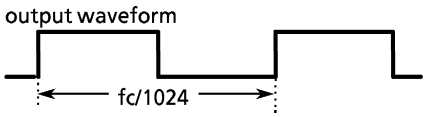



Figure 1-10. Examples of Resonator Connection

*Note : Accurate Adjustment of the Oscillation Frequency:*
*Although no hardware to externally and directly monitor the basic clock pulse is not provided, the oscillation frequency can be adjusted by making the program to output fixed frequency pulses to the port while disabling all interrupts and monitoring this pulse. With a system requiring adjustment of the oscillation frequency, the adjusting program must be created beforehand.*

Example: To output the high-frequency oscillation frequency adjusting monitor pulse to P13 ($\overline{DVO}$) pin.

```
SFCCHK:  LD   (P1CR), 00001000B  ; Configures port P13 as an output
         SET  (P1).3             ; P13 output latch ← 1
         LD   (TBTCR), 11100000B ; Enables divider output
         JRS  T, $               ; Loops endless
```




### 1.8.2 Timing Generator

The timing generator generates from the basic clock the various system clocks supplied to the CPU core and peripheral hardware. The timing generator provides the following functions :

① Generation of main system clock
② Generation of divider output ($\overline{DVO}$) pulses
③ Generation of source clocks for time base timer
④ Generation of source clocks for watchdog timer
⑤ Generation of internal source clocks for timer/counters TC1 – TC4
⑥ Generation of internal clocks for serial interfaces SIO1 and SIO2
⑦ Generation of warm-up clocks for releasing STOP mode
⑧ Generation of a clock for releasing reset output

(1) Configuration of Timing Generator

The timing generator consists of a 21-stage divider with a divided-by-4 prescaler, a main system clock generator, and machine cycle counters. An input clock to the 7th stage of the divider depends on the operating mode and DV7CK (bit 4 in TBTCR) shown in Figure 1-11 as follows.

During reset and at releasing STOP mode, the divider is cleared to "0", however, the prescaler is not cleared.

① In the single-clock mode
A divided-by-256 of high-frequency clock (fc/$2^8$) is input to the 7th stage of the divider.

② In the dual-clock mode
During NORMAL2 or IDLE2 mode (SYSCK = 0), an input clock to the 7th stage of the divider can be selected either "fc/$2^8$" or "fs" with DV7CK.
During SLOW or SLEEP mode (SYSCK = 1), fs is automatically input to the 7th stage. To input clock to the 1st stage is stopped ; output from the 1st to 6th stages is also stopped.

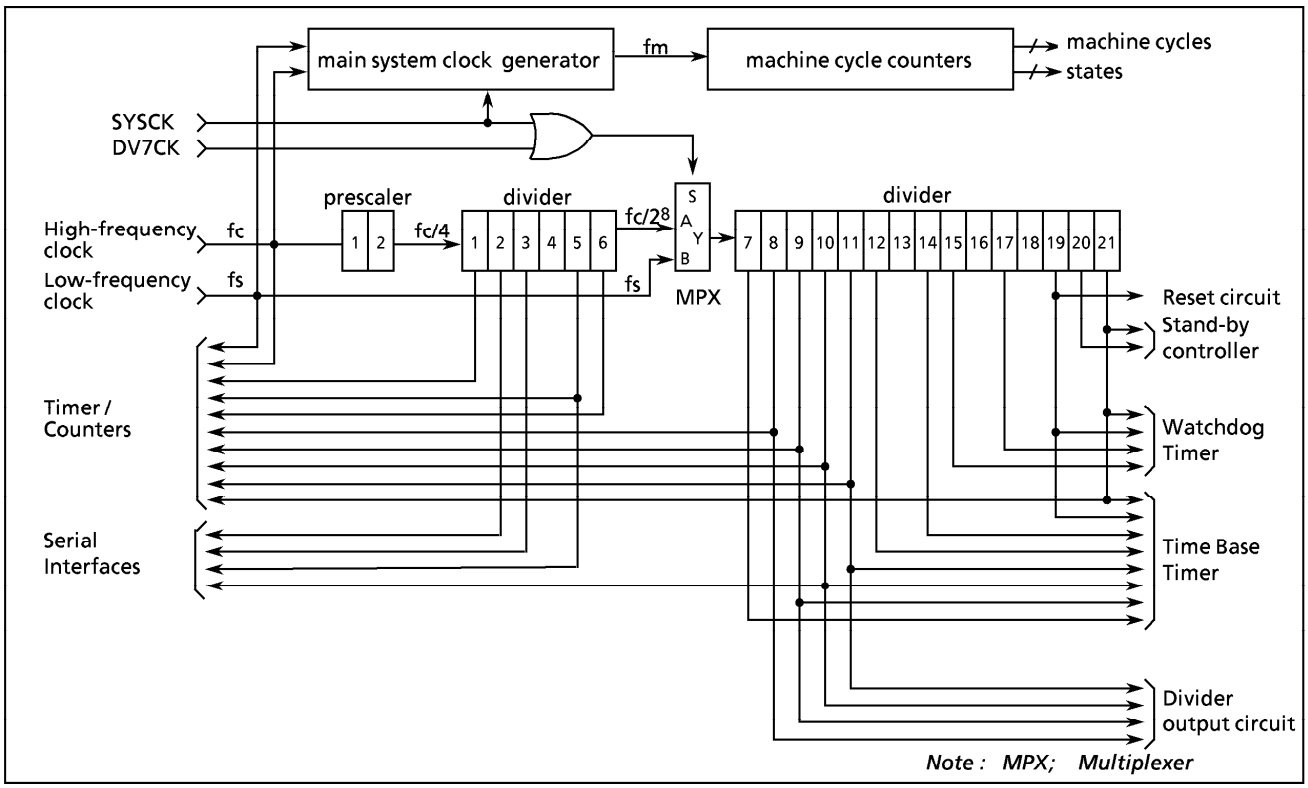



Figure 1-11. Configuration of Timing Generator

**TBTCR** ($0036_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| (DVOEN) | (DVOCK) | | DV7CK | (TBTEN) | (TBTCK) | | |

(Initial value: 0**0 0***)

| | | | |
|---|---|---|---|
| DV7CK | Selection of input clock to the 7th stage of the divider | 0 : fc/$2^8$ [Hz]<br>1 : fs | R/W |

*Note1 : fc ; high-frequency clock [Hz], fs ; low-frequency clock [Hz], * ; don't care*
*Note 2 : Do not set DV7CK to "1" in the single-clock mode.*
*Note 3 : Do not set DV7CK to "1" before low-frequency clock is stable in the dual-clock mode.*

Figure 1-12. Timing Generator Control Register

(2) Machine Cycle

Instruction execution and peripheral hardware operation are synchronized with the main system clock. The minimum instruction execution unit is called an "machine cycle". There are a total of 10 different types of instructions for the TLCS-870 Series: ranging from 1-cycle instructions which require one machine cycle for execution to 10-cycle instructions which require 10 machine cycles for execution.

A machine cycle consists of 4 states (S0 to S3), and each state consists of one main system clock.

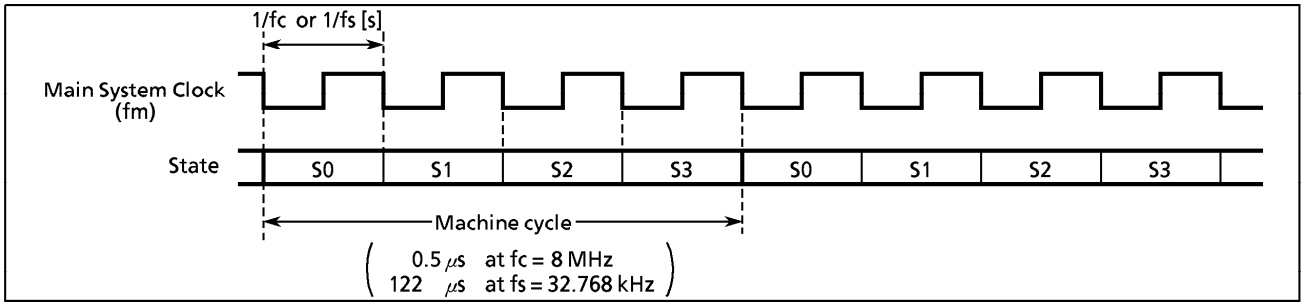



Figure 1-13. Machine Cycle

### 1.8.3 Stand-by Controller

The stand-by controller starts and stops the oscillation circuits for the high-frequency and low-frequency clocks, and switches the main system clock. There are two operating modes: single-clock and dual-clock. These modes are controlled by the system control registers (SYSCR1, SYSCR2).

Figure 1-14 shows the operating mode transition diagram and Figure 1-15 shows the system control registers. Either the single-clock or the dual-clock mode can be selected by an option during reset.

(1) **Single-clock mode**

Only the oscillation circuit for the high-frequency clock is used, and P21 (XTIN) and P22 (XTOUT) pins are used as input/output ports. In the single-clock mode, the machine cycle time is 4/fc [s] (0.5 μs at fc = 8 MHz).

① NORMAL1 mode

In this mode, both the CPU core and on-chip peripherals operate using the high-frequency clock. In the case where the single-clock mode has been selected as an option, the 87C841/CC41/CH41/CK41/CM41 are placed in this mode after reset.

② IDLE1 mode

In this mode, the internal oscillation circuit remains active, and the CPU and the watchdog timer are halted; however, on-chip peripherals remain active (operate using the high-frequency clock). IDLE1 mode is started by setting IDLE bit in the system control register 2 (SYSCR2), and IDLE1 mode is released to NORMAL1 mode by an interrupt request from on-chip peripherals or external interrupt inputs. When IMF (interrupt master enable flag) is "1" (interrupt enable), the execution will resume upon acceptance of the interrupt, and the operation will return to normal after the interrupt service is completed. When IMF is "0" (interrupt disable), the execution will resume with the instruction which follows IDLE mode start instruction.

③ STOP1 mode

In this mode, the internal oscillation circuit is turned off, causing all system operations to be halted. The internal status immediately prior to the halt is held with the lowest power consumption during this mode. The output status of all output ports can be set to either output hold or high-impedance under software control.

STOP1 mode is started by setting STOP bit in the system control register 1 (SYSCR1), and STOP1 mode is released by an input (either level-sensitive or edge-sensitive can be programmably selected) to the $\overline{\text{STOP}}$ pin. After the warming-up period is completed, the execution resumes with the next instruction which follows the STOP mode start instruction.

(2) **Dual-clock mode**

Both high-frequency and low-frequency oscillation circuits are used in this mode. Pins P21 (XTIN) and P22 (XTOUT) cannot be used as input/output ports. The main system clock is obtained from the high-frequency clock in NORMAL2 and IDLE2 modes, and is obtained from the low-frequency clock in SLOW and SLEEP modes. The machine cycle time is 4/fc [s] (0.5 $\mu$s at fc = 8 MHz) in NORMAL2 and IDLE2 modes, and 4/fs [s] (122 $\mu$s at fs = 32.768 kHz) in SLOW and SLEEP modes. Note that the 87PM41 is placed in the single-clock mode during reset. To use the dual-clock mode, the low-frequency oscillator should be turned on by executing [SET (SYSCR2).XTEN] instruction.

① NORMAL2 mode

In this mode, the CPU core operates using the high-frequency clock. On-chip peripherals operate using the high-frequency clock and/or low-frequency clock. In case that the dual-clock mode has been selected by an option, the 87C841/CC41/CH41/CK41/CM41 are placed in this mode after reset.

② SLOW mode

This mode can be used to reduce power-consumption by turning off oscillation of the high-frequency clock. The CPU core and on-chip peripherals operate using the low-frequency clock.

Switching back and forth between NORMAL2 and SLOW modes is performed by the system control register 2.

③ IDLE2 mode

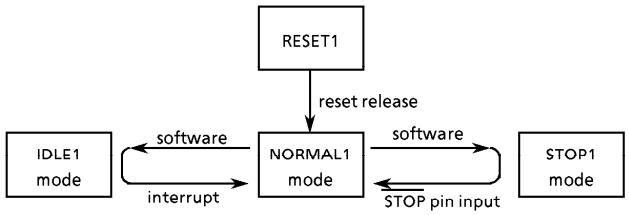
In this mode, the internal oscillation circuits remain active. The CPU and the watchdog timer are halted; however, on-chip peripherals remain active (operate using the high-frequency clock and/or the low-frequency clock). Starting and releasing of IDLE2 mode are the same as for IDLE1 mode, except that operation returns to NORMAL2 mode.

④ SLEEP mode

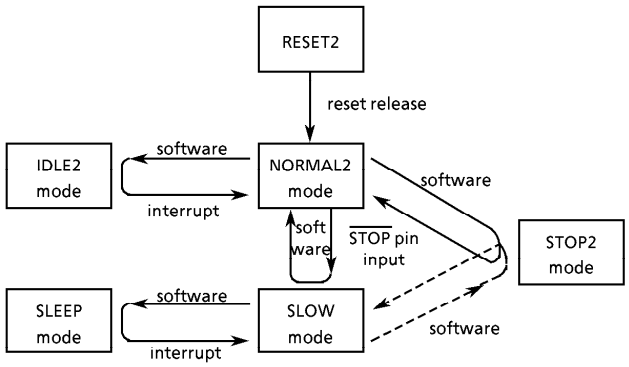
In this mode, the internal oscillation circuit of the low-frequency clock remains active. The CPU, the watchdog timer, and the internal oscillation circuit of the high- frequency clock are halted; however, on-chip peripherals remain active (operate using the low-frequency clock). Starting and releasing of SLEEP mode is the same as for IDLE1 mode, except that operation returns to SLOW mode.

⑤ STOP2 mode

As in STOP1 mode, all system operations are halted in this mode.




(a) Single-clock mode




(b) Dual-clock mode

*Note : NORMAL1 and NORMAL2 modes are generically called NORMAL; STOP1 and STOP2 are called STOP; and IDLE1, IDLE2 and SLEEP are called IDLE.*

| Operating mode | | Frequency | | CPU core | On-chip Peripherals | Machine cycle time |
|---|---|---|---|---|---|---|
| | | High-frequency | Low-frequency | | | |
| Single-Clock | RESET1 | turning on oscillation | turning off oscillation | reset | reset | 4/fc [s] |
| | NORMAL1 | | | operate | operate | |
| | IDLE1 | | | halt | | |
| | STOP1 | turning off oscillation | | | halt | — |
| Dual-Clock | RESET2 | turning on oscillation | turning on oscillation | reset | reset | 4/fc [s] |
| | NORMAL2 | | | High-frequency | operate (High and/or Low) | |
| | IDLE2 | | | halt | | |
| | SLOW | turning off oscillation | | Low-frequency | Low-frequency | 4/fs [s] |
| | SLEEP | | | halt | | |
| | STOP2 | | turning off oscillation | | halt | — |

Figure 1-14. Operating Mode Transition Diagram

System Control Register 1

**SYSCR1** ($0038_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| STOP | RELM | RETM | OUTEN | WUT | | | |

(Initial value: 0000 00**)

| | | | |
|---|---|---|---|
| STOP | STOP mode start | 0 : CPU core and peripherals remain active<br>1 : CPU core and peripherals are halted (start STOP mode) | R/W |
| RELM | Release method for STOP mode | 0 : Edge-sensitive release<br>1 : Level-sensitive release | |
| RETM | Operating mode after STOP mode | 0 : Return to NORMAL mode<br>1 : Return to SLOW mode | |
| OUTEN | Port output control during STOP mode | 0 : High-impedance<br>1 : Remain unchanged | |
| WUT | Warming-up time at releasing STOP mode | 00 : $3\times2^{19}$ / fc or $3\times2^{13}$ / fs [s]<br>01 : $2^{19}$ / fc or $2^{13}$ / fs<br>1* : Reserved | |

*Note 1 :* *Always set RETM to "0" when transiting from NORMAL1 mode to STOP1 mode and from NOMAL2 mode to STOP2 mode. Always set RETM to "1" when transiting from SLOW mode to STOP2 mode.*

*Note 2 :* *When STOP mode is released with $\overline{\text{RESET}}$ pin input, a return is made to NORMAL mode regardless of the RETM contents.*

*Note 3 :* fc ; *high-frequency clock* [Hz]
fs ; *low-frequency clock* [Hz]
* ; *don't care*

*Note 4 :* *Bits 1 and 0 in SYSCR1 are read in as undefined data when a read instruction is executed.*

System Control Register 2

**SYSCR2** ($0039_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| XEN | XTEN | SYSCK | IDLE | | | | |

(Initial value: 10/100 ****)

| | | | |
|---|---|---|---|
| XEN | High-frequency oscillator control | 0 : Turn off oscillation<br>1 : Turn on oscillation | R/W |
| XTEN | Low-frequency oscillator control | 0 : Turn off oscillation<br>1 : Turn on oscillation | |
| SYSCK | Main system clock select (write)/main system clock monitor (read) | 0 : High-frequency clock<br>1 : Low-frequency clock | |
| IDLE | IDLE mode start | 0 : CPU and watchdog timer remain active<br>1 : CPU and watchdog timer are stopped (start IDLE mode) | |

*Note 1 :* *A reset is applied ($\overline{\text{RESET}}$ pin output goes low) if both XEN and XTEN are cleared to "0".*

*Note 2 :* *Do not clear XEN to "0" when SYSCK = 0, and do not clear XTEN to "0" when SYSCK = 1.*

*Note 3 :* *WDT; watchdog timer, * ; don't care*

*Note 4 :* *Bits 3 - 0 in SYSCR2 are always read in as "1" when a read instruction is executed.*

*Note 5 :* *An optional initial value can be selected for XTEN. Always specify when ordering ES (engineering sample).*

| XTEN | operating mode after reset |
|---|---|
| 0 | Single-clock mode (NORMAL1) |
| 1 | Dual-clock mode (NORMAL2) |

Figure 1-15. System Control Registers

## 1.8.4 Operating Mode Control

(1) **STOP** mode (STOP1, STOP2)

STOP mode is controlled by the system control register 1 (SYSCR1) and the $\overline{\text{STOP}}$ pin input. The $\overline{\text{STOP}}$ pin is also used both as a port P20 and an $\overline{\text{INT5}}$ (external interrupt input 5) pin. STOP mode is started by setting STOP (bit 7 in SYSCR1 ) to "1". During STOP mode, the following status is maintained.

① Oscillations are turned off, and all internal operations are halted.
② The data memory, registers and port output latches are all held in the status in effect before STOP mode was entered. The port output can be select either output hold or high-impedance by setting OUTEN ( bit 4 in SYSCR1).
③ The divider of the timing generator is cleared to "0".
④ The program counter holds the address of the instruction following the instruction which started the STOP mode.

STOP mode includes a level-sensitive release mode and an edge-sensitive release mode, either of which can be selected with RELM (bit 6 in SYSCR1).

a. Level-sensitive release mode (RELM = 1)

In this mode, STOP mode is released by setting the $\overline{\text{STOP}}$ pin high. This mode is used for capacitor back-up when the main power supply is cut off and long term battery back-up.

When the $\overline{\text{STOP}}$ pin input is high, executing an instruction which starts the STOP mode will not place in STOP mode but instead will immediately start the release sequence (warm-up). Thus, to start STOP mode in the level-sensitive release mode, it is necessary for the program to first confirm that the $\overline{\text{STOP}}$ pin input is low. The following method can be used for confirmation:

- Using an external interrupt input $\overline{\text{INT5}}$ ($\overline{\text{INT5}}$ is a falling edge-sensitive input).

Example : Starting STOP mode with an INT5 interrupt.

```
PINT5 :   TEST   (P2) . 0              ;  To reject noise, the STOP mode does not start if
          JRS    F, SINT5                 port P20 is at high
          LD     (SYSCR1), 01000000B     ;  Sets up the level-sensitive release mode.
          SET    (SYSCR1) . 7            ;  Starts STOP mode
SINT5 :   RETI
```

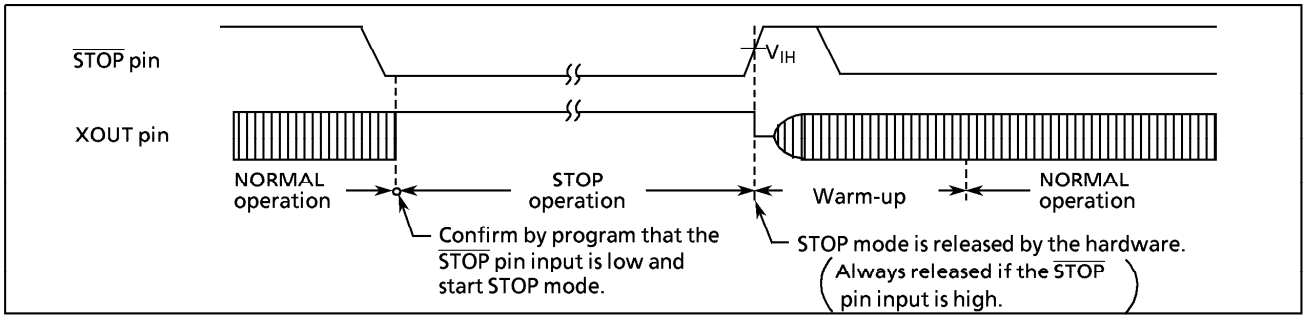



Figure 1-16. Level-sensitive Release Mode

*Note : When changing to the level-sensitive release mode from the edge-sensitive release mode, the release mode is not switched until a rising edge of the $\overline{\text{STOP}}$ pin input is detected.*

b. Edge-sensitive release mode (RELM = 0)

In this mode, STOP mode is released by a rising edge of the $\overline{STOP}$ pin input. This is used in applications where a relatively short program is executed repeatedly at periodic intervals. This periodic signal (for example, a clock from a low-power consumption oscillator) is input to the $\overline{STOP}$ pin.

In the edge-sensitive release mode, STOP mode is started even when the $\overline{STOP}$ pin input is high.

Example : Starting STOP mode operation in the edge-sensitive release mode

```
LD      (SYSCR1), 10000000B    ;  OUTEN ← 0 (specifies high-impedance)
DI                             ;  IMF ← 0 (disables interrupt service)
SET     (SYSCR1).STOP          ;  STOP ← 1 (activates stop mode)
LDW     (IL),1110011101010111B ;  IL12, 11, 7, 5, 3 ← 0
                                  (clears interrupt latches)
EI                             ;  IMF ← 1 (enables interrupt service)
```

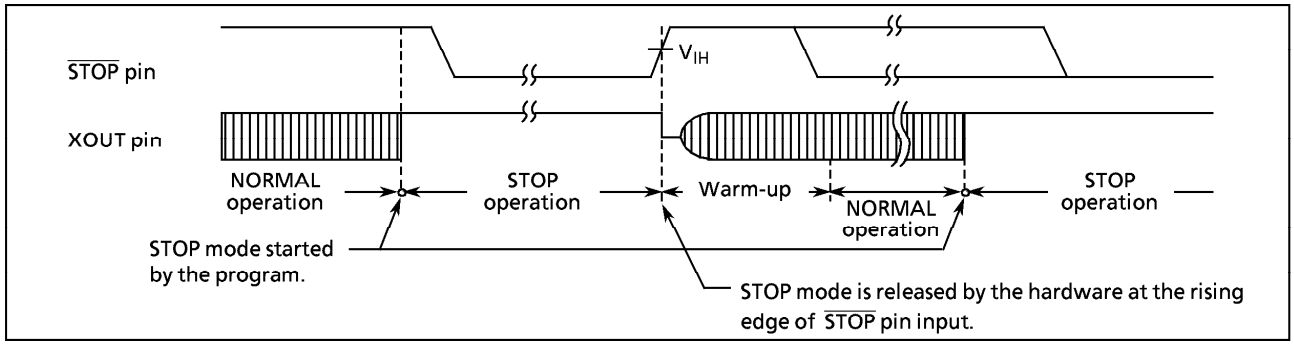



Figure 1-17. Edge-sensitive Release Mode

STOP mode is released by the following sequence:

① When returning to NORMAL2, both the high-frequency and low-frequency clock oscillators are turned on ; when returning to SLOW mode, only the low-frequency clock oscillator is turned on. When returning to NORMAL1, only the high-frequency clock oscillator is turned on.

② A warming-up period is inserted to allow oscillation time to stabilize. During warm-up, all internal operations remain halted. Two different warming-up times can be selected with WUT ( bits 2 and 3 in SYSCR1) as determined by the resonator characteristics.

③ When the warming-up time has elapsed, normal operation resumes with the instruction following the STOP mode start instruction (e.g. [SET (SYSCR1). 7]). The start is made after the divider of the timing generator is cleared to "0".

Table 1-1. Warming-up Time example

| Return to NORMAL1 mode | | | Return to SLOW mode | |
|---|---|---|---|---|
| WUT | At fc = 4.194304 MHz | At fc = 8 MHz | WUT | At fs = 32.768 kHz |
| $3 \times 2^{19}$ / fc [s] | 375 [ms] | 196.6 [ms] | $3 \times 2^{13}$ / fs [s] | 750 [ms] |
| $2^{19}$ / fc | 125 | 65.5 | $2^{13}$ / fs | 250 |

*Note : The warming-up time is obtained by dividing the basic clock by the divider: therefore, the warming-up time may include a certain amount of error if there is any fluctuation of the oscillation frequency when STOP mode is released. Thus, the warming-up time must be considered an approximate value.*

STOP mode can also be released by setting the $\overline{RESET}$ pin low, which immediately performs the normal reset operation.

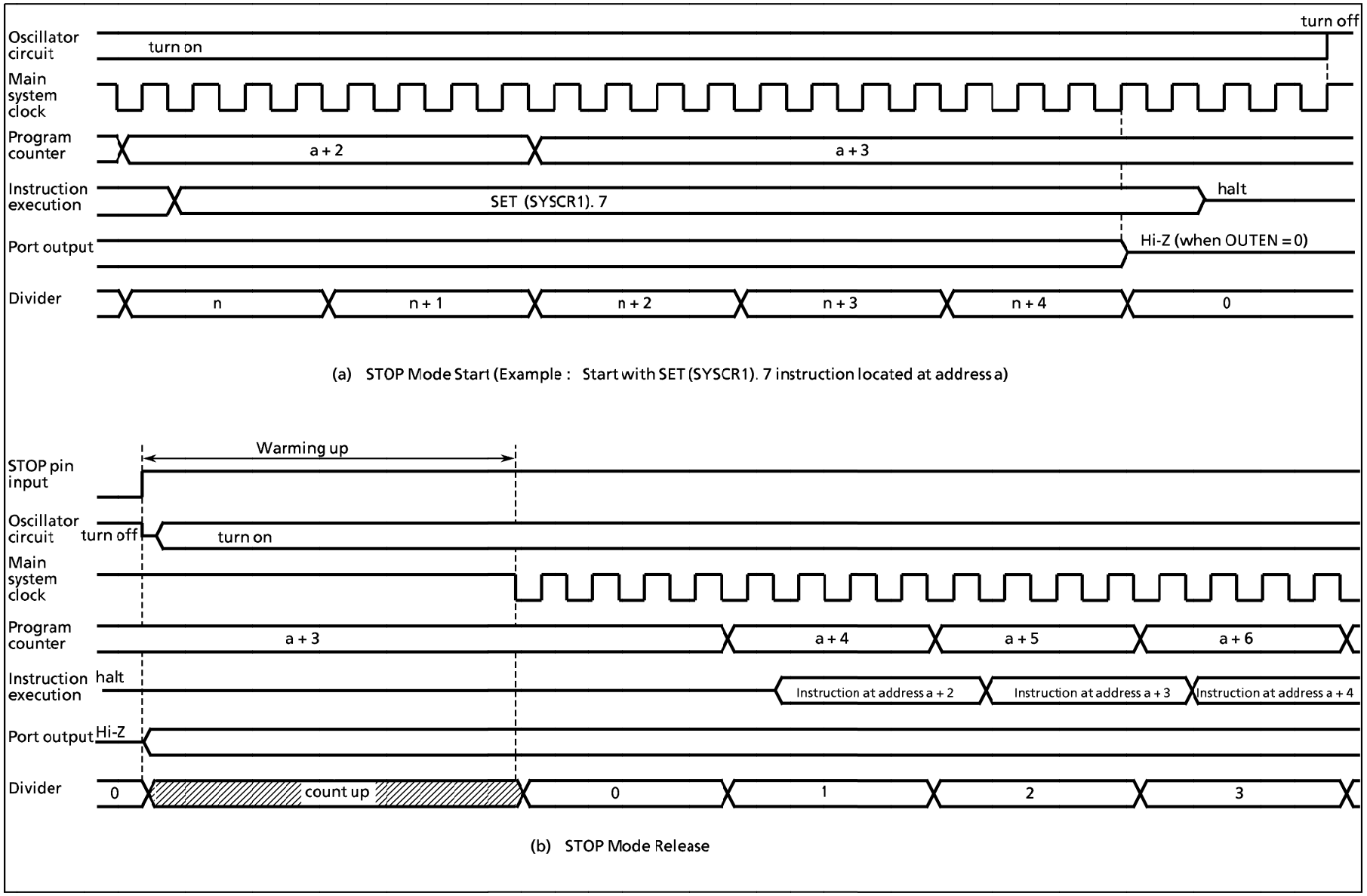



Figure 1-18. STOP Mode Start / Release

*Note : When STOP mode is released with a low hold voltage, the following cautions must be observed.*
*The power supply voltage must be at the operating voltage level before releasing STOP mode. The $\overline{\text{RESET}}$ pin input must also be high, rising together with the power supply voltage. In this case, if an external time constant circuit has been connected, the $\overline{\text{RESET}}$ pin input voltage will increase at a slower rate than the power supply voltage. At this time, there is a danger that a reset may occur if input voltage level of the $\overline{\text{RESET}}$ pin drops below the non-inverting high-level input voltage (hysteresis input).*

(2) **IDLE** mode (IDLE1, IDLE2, SLEEP)

IDLE mode is controlled by the system control register 2 and maskable interrupts. The following status is maintained during IDLE mode.

① Operation of the CPU and watchdog timer is halted. On-chip peripherals continue to operate.
② The data memory, CPU registers and port output latches are all held in the status in effect before IDLE mode was entered.
③ The program counter holds the address of the instruction following the instruction which started IDLE mode.

Example : Starting IDLE mode.

```
SET        (SYSCR2) . 4        ;  IDLE←1
```

IDLE mode includes a normal release mode and an interrupt release mode. Selection is made with the interrupt master enable flag (IMF). Releasing the IDLE mode returns from IDLE1 to NORMAL1, from IDLE2 to NORMAL2, and from SLEEP to SLOW mode.

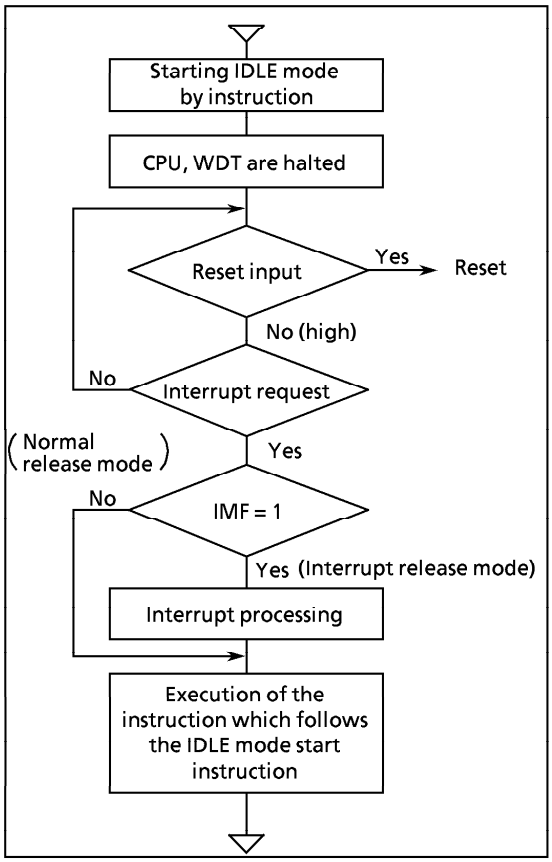



Figure 1-19. IDLE Mode

a. Normal release mode (IMF = "0")
IDLE mode is released by any interrupt source enabled by the individual interrupt enable flag (EF) or an external interrupt 0 ($\overline{\text{INT0}}$ pin) request. Execution resumes with the instruction following the IDLE mode start instruction (e.g. [SET (SYSCR2).4]).

b. Interrupt release mode (IMF = "1")
IDLE mode is released and interrupt processing is started by any interrupt source enabled with the individual interrupt enable flag (EF) or an external interrupt 0 ($\overline{\text{INT0}}$ pin) request. After the interrupt is processed, the execution resumes from the instruction following the instruction which started IDLE mode.

IDLE mode can also be released by setting the $\overline{\text{RESET}}$ pin low, which immediately performs the reset operation. After reset, the 87C841/CC41/CH41/CK41/CM41 are placed in NORMAL mode.

*Note : When a watchdog timer interrupt is generated immediately before the IDLE mode is started, the watchdog timer interrupt will be processed but IDLE mode will not be started.*

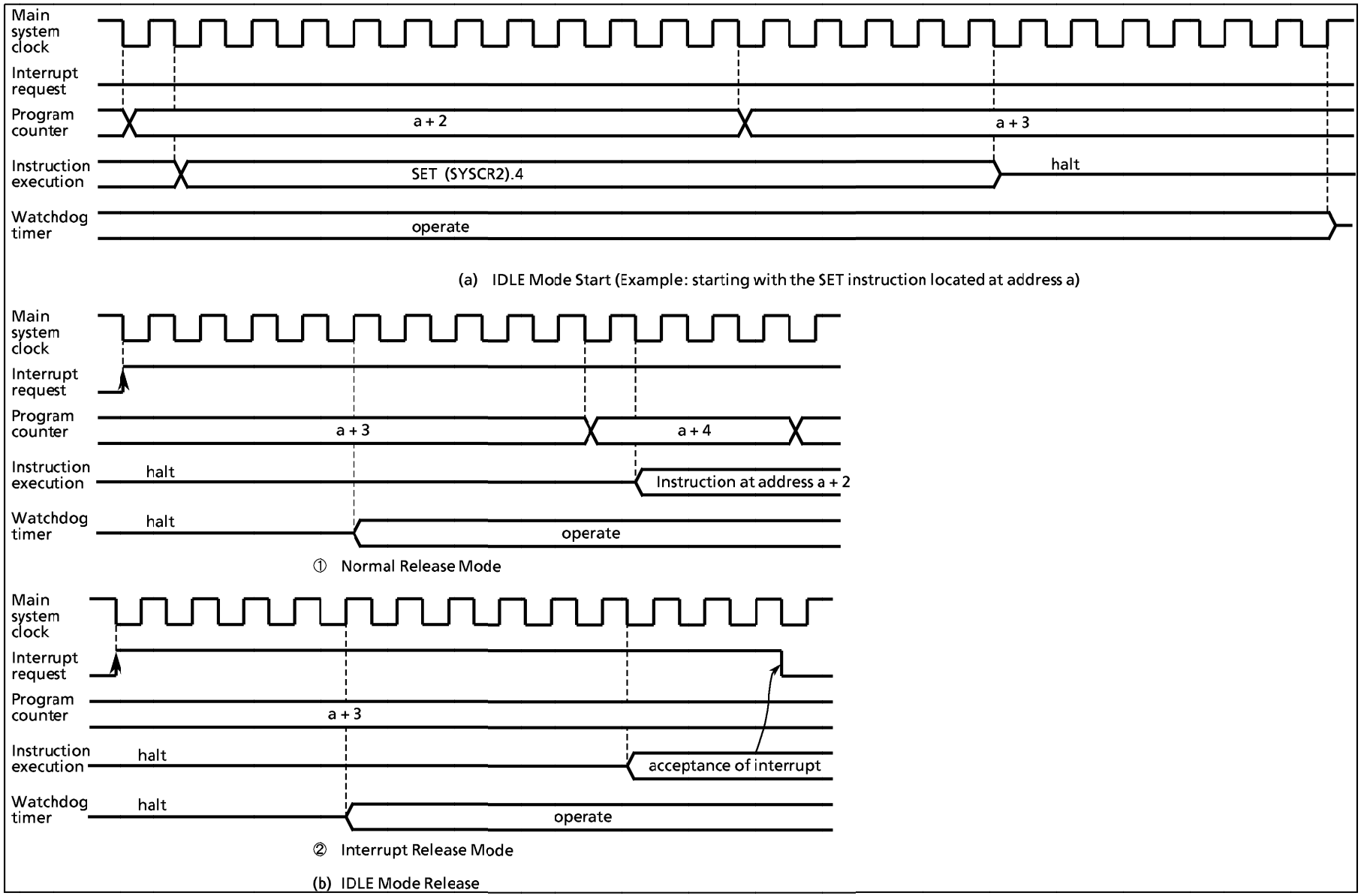



Figure 1-20. IDLE Mode Start/Release

(3) **SLOW** mode

SLOW mode is controlled by the system control register 2 and the timer/counter 2.

a. Switching from NORMAL2 mode to SLOW mode

First, set SYSCK (bit 5 in SYSCR2) to switch the main system clock to the low-frequency clock. Next, clear XEN (bit 7 in SYSCR2) to turn off high-frequency oscillation.

When the low-frequency clock oscillation is unstable, wait until oscillation stabilizes before performing the above operations. The timer/counter 2 (TC2) can conveniently be used to confirm that low-frequency clock oscillation has stabilized.

*Note : The high frequency clock can be continued oscillation in order to return to NORMAL2 mode from SLOW mode quickly. Always turn off oscillation of high frequency clock when switching from SLOW mode to STOP mode.*

Example1 : Switching from NORMAL2 mode to SLOW mode.

```
        SET     (SYSCR2) . 5        ; SYSCK←1   (Switches the main system clock to the
                                                low-frequency clock)
        CLR     (SYSCR2) . 7        ; XEN←0     (turns off high-frequency oscillation)
```

Example2 : Switching to SLOW mode after low-frequency clock oscillation has stabilized.

```
        LD      (TC2CR), 14H        ; Sets TC2 mode
                                      (timer mode, source clock : fs)
        LDW     (TREG2), 8000H      ; Sets warming-up time
                                      (according to Xtal characteristics)
        LD      (TC2CR), 34H        ; Starts TC2
        ⋮
PINTTC2 : LD    (TC2CR), 10H        ; Stops TC2
        SET     (SYSCR2) . 5        ; SYSCK←1
        CLR     (SYSCR2) . 7        ; XEN←0
        RETI
        ⋮
VINTTC2 : DW    PINTTC2             ; INTTC2 vector table
```

b. Switching from SLOW mode to NORMAL2 mode

First, set XEN (bit 7 in SYSCR2) to turn on the high-frequency oscillation. When time for stabilization (warm-up) has been taken by the timer/counter 2 (TC2), clear SYSCK (bit 5 in SYSCR2) to switch the main system clock to the high-frequency clock.

SLOW mode can also be released by setting the $\overline{\text{RESET}}$ pin low, which immediately performs the reset operation. After reset, the 87C841/CC41/CH41/CK41/CM41 are placed in NORMAL mode.

Example : Switching from SLOW mode to NORMAL2 mode (fc = 8 MHz, warming-up time is about 7.9 ms).

```
        SET     (SYSCR2) . 7        ; XEN←1     (turns on high-frequency oscillation)
        LD      (TC2CR), 10H        ; Sets TC2 mode
                                      (timer mode, source clock: fc)
        LD      (TREG2 + 1), 0F8H   ; Sets the warming-up time
                                      (according to frequency and resonator characteristics)
        LD      (TC2CR), 30H        ; Starts TC2
        ⋮
PINTTC2 : LD    (TC2CR), 10H        ; Stops TC2
        CLR     (SYSCR2) . 5        ; SYSCK←0   (Switches the main system clock to the
                                                high-frequency clcok)
        RETI
        ⋮
VINTTC2 : DW    PINTTC2             ; INTTC2 vector table
```

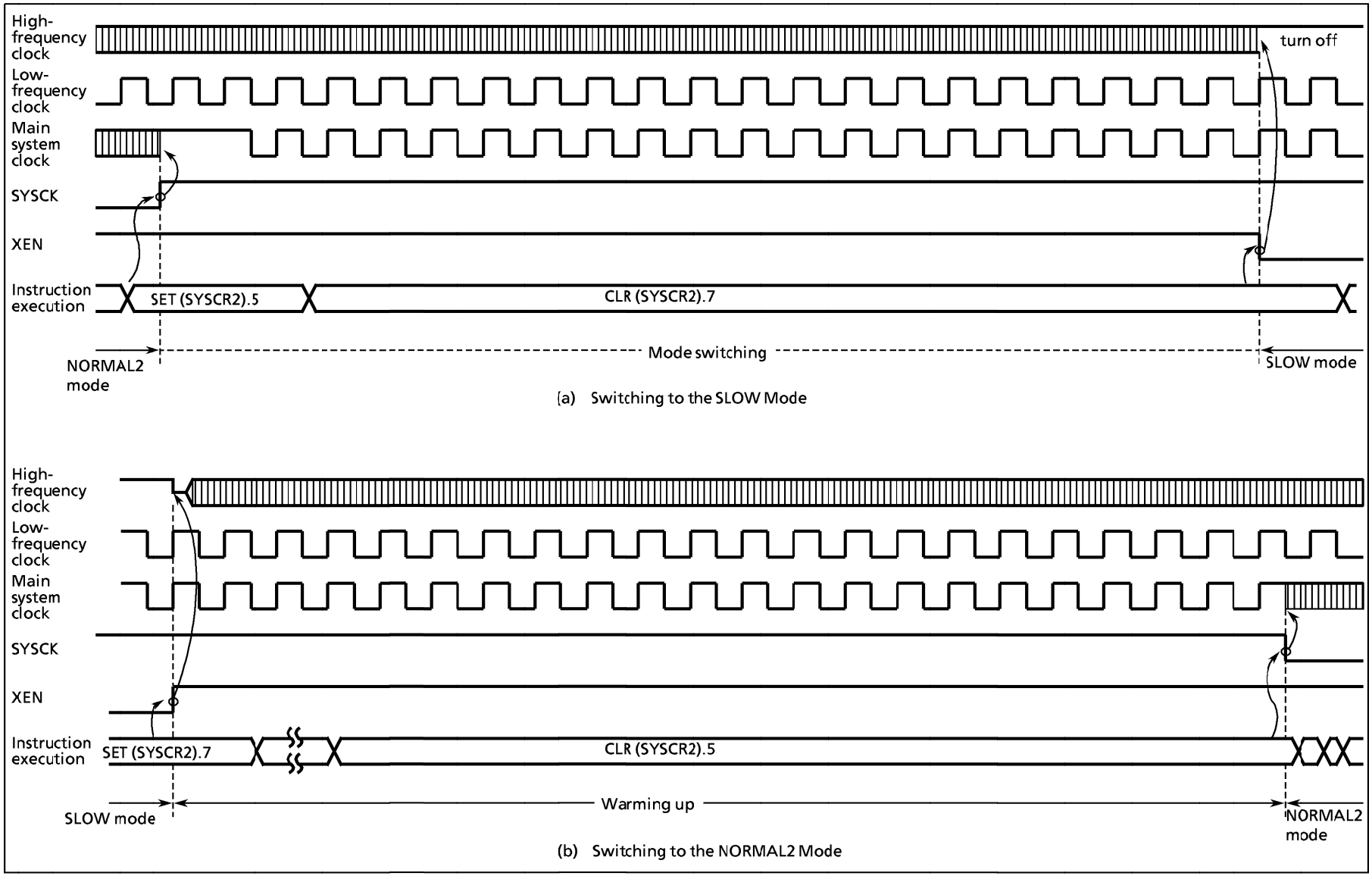



Figure 1-21. Switching between the NORMAL2 and SLOW Modes

## 1.9 Interrupt Controller

The 87C841/CC41/CH41/CK41/CM41 each have a total of 15 interrupt sources: 6 externals and 9 internals. Nested interrupt control with priorities is also possible. Two of the internal sources are pseudo non-maskable interrupts; the remainder are all maskable interrupts.

Interrupt latches (IL) that hold the interrupt requests are provided for interrupt sources. Each interrupt vector is independent.

The interrupt latch is set to "1" when an interrupt request is generated and requests the CPU to accept the interrupt. The acceptance of maskable interrupts can be selectively enabled and disabled by the program using the interrupt master enable flag (IMF) and the individual interrupt enable flags (EF). When two or more interrupts are generated simultaneously, the interrupt is accepted in the highest priority order as determined by the hardware. Figure 1-22 shows the interrupt controller.

| Interrupt Source | | | Enable Condition | Interrupt Latch | Vector Table Address | Priority |
|---|---|---|---|---|---|---|
| Internal/ External | (Reset) | | Non-Maskable | — | $FFFE_H$ | High 0 |
| Internal | INTSW | (Software interrupt) | Pseudo non-maskable | — | $FFFC_H$ | 1 |
| Internal | INTWDT | (Watchdog Timer interrupt) | | $IL_2$ | $FFFA_H$ | 2 |
| External | INT0 | (External interrupt 0) | IMF = 1, INT0EN = 1 | $IL_3$ | $FFF8_H$ | 3 |
| Internal | INTTC1 | (16-bit TC1 interrupt) | IMF · $EF_4$ = 1 | $IL_4$ | $FFF6_H$ | 4 |
| External | INT1 | (External interrupt 2) | IMF · $EF_5$ = 1 | $IL_5$ | $FFF4_H$ | 5 |
| Internal | INTTBT | (Time Base Timer interrupt) | IMF · $EF_6$ = 1 | $IL_6$ | $FFF2_H$ | 6 |
| External | INT2 | (External interrupt 2) | IMF · $EF_7$ = 1 | $IL_7$ | $FFF0_H$ | 7 |
| Internal | INTTC3 | (8-bit TC3 interrupt) | IMF · $EF_8$ = 1 | $IL_8$ | $FFEE_H$ | 8 |
| Internal | INTSIO1 | (Serial Interface 1 interrupt) | IMF · $EF_9$ = 1 | $IL_9$ | $FFEC_H$ | 9 |
| Internal | INTTC4 | (8-bit TC4 interrupt) | IMF · $EF_{10}$ = 1 | $IL_{10}$ | $FFEA_H$ | 10 |
| External | INT3 | (External interrupt 3) | IMF · $EF_{11}$ = 1 | $IL_{11}$ | $FFE8_H$ | 11 |
| External | INT4 | (External interrupt 4) | IMF · $EF_{12}$ = 1 | $IL_{12}$ | $FFE6_H$ | 12 |
| Internal | INTSIO2 | (Serial Interface 2 interrupt) | IMF · $EF_{13}$ = 1 | $IL_{13}$ | $FFE4_H$ | 13 |
| Internal | INTTC2 | (16-bit TC2 interrupt) | IMF · $EF_{14}$ = 1 | $IL_{14}$ | $FFE2_H$ | 14 |
| External | INT5 | (External interrupt 5) | IMF · $EF_{15}$ = 1 | $IL_{15}$ | $FFE0_H$ | 15 |

### (1) Interrupt Latches ($IL_{15\ to\ 2}$)

Interrupt latches are provided for each source, except for a software interrupt. The latch is set to "1" when an interrupt request is generated, and requests the CPU to accept the interrupt. The latch is cleared to "0" just after the interrupt is accepted. All interrupt latches are initialized to "0" during reset.

The interrupt latches are assigned to addresses $003C_H$ and $003D_H$ in the SFR. Each latch can be cleared to "0" individually by an instruction; however, the read-modify-write instruction such as bit manipulation or operation instructions cannot be used (Do not clear the $IL_2$ for a watchdog timer interrupt to "0"). Thus, interrupt requests can be cancelled and initialized by the program. Note that interrupt latches cannot be set to "1" by any instruction.

The contents of interrupt latches can be read out by an instruction. Therefore, testing interrupt requests by software is possible.

```
Example 1 : Clears interrupt latches
            LDW     (IL), 1110100000111111B     ; IL12, IL10 to IL6←0
Example 2 : Reads interrupt latches
            LD      WA, (IL)                    ; W←ILH, A←ILL
Example 3: Tests an interrupt latch
            TEST    (IL).7                      ; if IL7 = 1 then jump
            JR      F, SSET
```

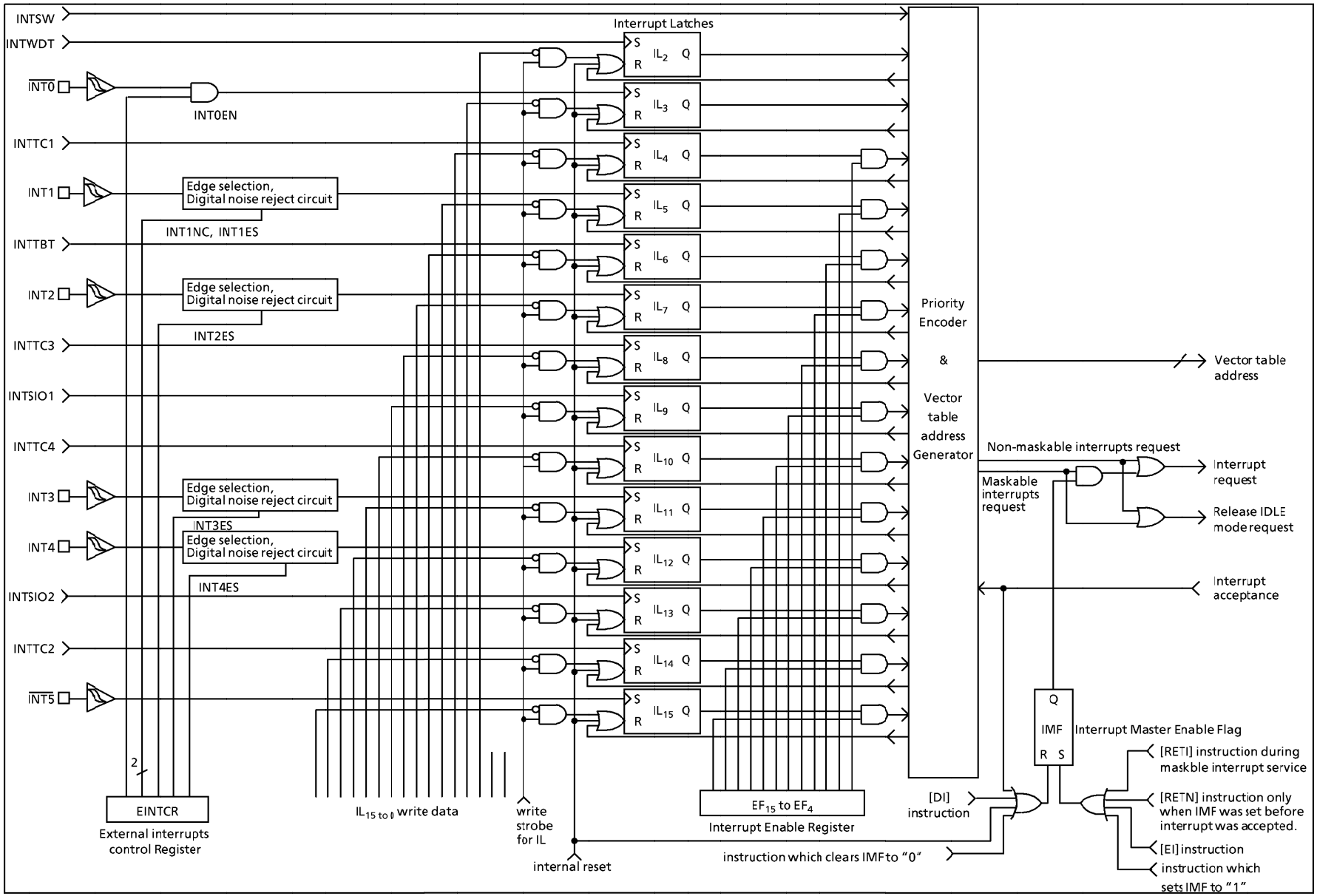



Figure 1-22. Interrupt Controller Block Diagram

(2) **Interrupt Enable Register** (EIR)

The interrupt enable registers (EIR) enable and disable the acceptance of interrupts except for the pseudo non-maskable interrupts (software and watchdog timer interrupts). Pseudo non-maskable interrupts are accepted regardless of the contents of the EIR; however, the pseudo non-maskable interrupts cannot be nested more than once at the same time. For example, the watchdog timer interrupt is not accepted during the software interrupt service.

The EIR consists of an interrupt master enable flag (IMF) and individual interrupt enable flags (EF). These registers are assigned to addresses $003A_H$ and $003B_H$ in the SFR, and can be read and written by an instruction (including read-modify-write instructions such as bit manipulation instructions).

① **Interrupt Master enable Flag** (IMF)

The interrupt master enable flag (IMF) enables and disables the acceptance of all interrupts, except for pseudo non-maskable interrupts. Clearing this flag to "0" disables the acceptance of all maskable interrupts. Setting to "1" enables the acceptance of interrupts.

When an interrupt is accepted, this flag is cleared to "0" to temporarily disable the acceptance of maskable interrupts. After execution of the interrupt service program, this flag is set to "1" by the maskable interrupt return instruction [RETI] to again enable the acceptance of interrupts. If an interrupt request has already been occurred, interrupt service starts immediately after execution of the [RETI] instruction.

Pseudo non-maskable interrupts are returned by the [RETN] instruction. In this case, the IMF is set to "1" only when pseudo non-maskable interrupt service is started with interrupt acceptance enabled (IMF = 1). Note that IMF remains "0" when cleared by the interrupt service program.

The IMF is assigned to bit 0 at address $003A_H$ in the SFR, and can be read and written by an instruction. IMF is normally set and cleared by the [EI] and [DI] instructions, and the IMF is initialized to "0" during reset.

*Note : Do not set IMF to "1"during non-maskable interrupt service programs.*

② **Individual interrupt Enable Flags** ($EF_{15}$ to $EF_4$)

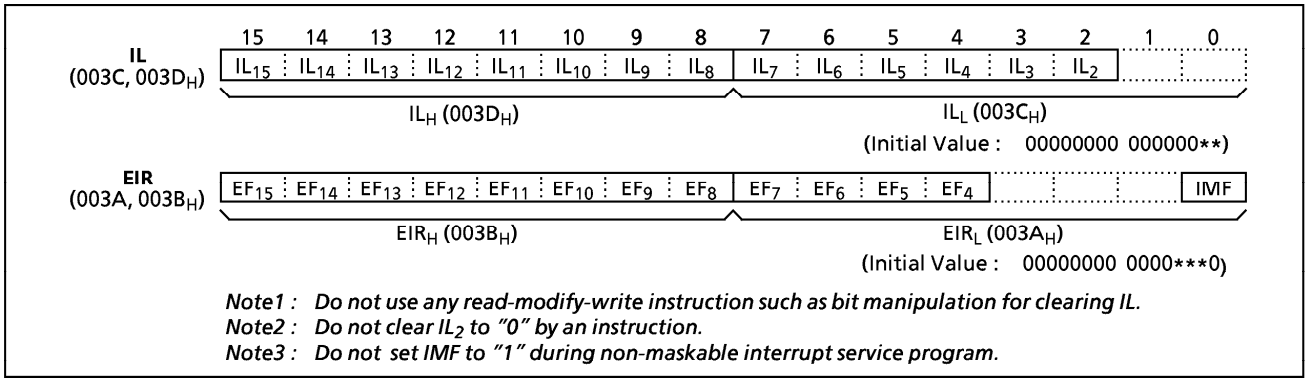
These flags enable and disable the acceptance of individual maskable interrupts, except for an external interrupt 0. Setting the corresponding bit of an individual interrupt enable flag to "1" enables acceptance of an interrupt, setting the bit to "0" disables acceptance.

Example 1 : Sets EF for individual interrupt enable, and sets IMF to "1".

```
LDW     (EIR), 1110100010100001B    ;  EF15~EF13, EF11, EF7, EF5, IMF←1
```

Example 2 : Sets an individual interrupt enable flag to "1".

```
SET     (EIRH).4                    ;  EF12←1
```




*Note1 : Do not use any read-modify-write instruction such as bit manipulation for clearing IL.*
*Note2 : Do not clear $IL_2$ to "0" by an instruction.*
*Note3 : Do not set IMF to "1" during non-maskable interrupt service program.*

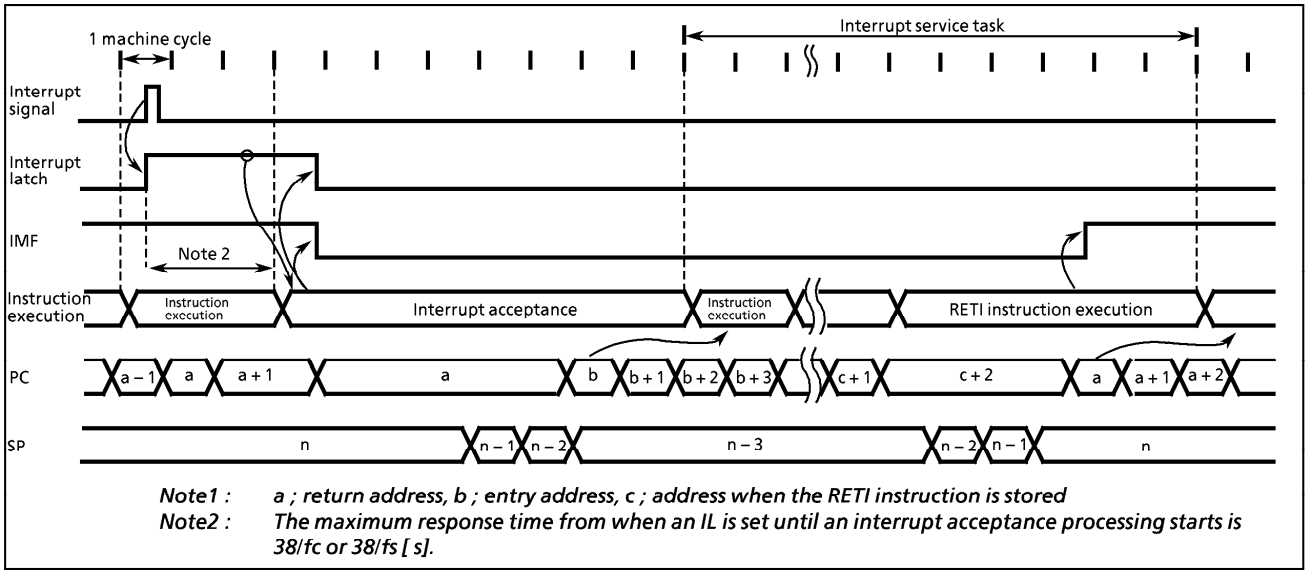
Figure 1-23. Interrupt Latch (IL) and Interrupt Enable Register (EIR)

### 1.9.1 Interrupt Sequence

An interrupt request is held until the interrupt is accepted or the interrupt latch is cleared to "0" by a reset or an instruction. Interrupt acceptance sequence requires 8 machine cycles (4 $\mu$s @ fc = 8 MHz in NORMAL mode) after the completion of the current instruction execution. The interrupt service task terminates upon execution of an interrupt return instruction [RETI] (for maskable interrupts) or [RETN] (for pseudo non-maskable interrupts).

Interrupt acceptance processing is as follows:

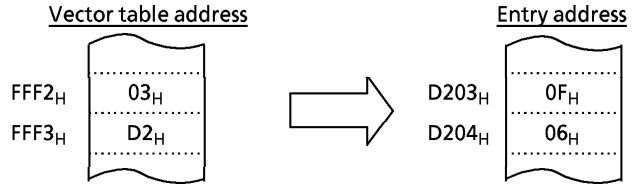
① The interrupt master enable flag (IMF) is cleared to "0" to temporarily disable the acceptance of any following maskable interrupts. When a non-maskable interrupt is accepted, the acceptance of any following interrupts is temporarily disabled.
② The interrupt latch (IL) for the interrupt source accepted is cleared to "0".
③ The contents of the program counter (return address) and the program status word are saved (pushed) onto the stack.
④ The entry address of the interrupt service program is read from the vector table address, and the entry address is loaded to the program counter.
⑤ The instruction stored at the entry address of the interrupt service program is executed.




*Note1 : a ; return address, b ; entry address, c ; address when the RETI instruction is stored*
*Note2 : The maximum response time from when an IL is set until an interrupt acceptance processing starts is 38/fc or 38/fs [ s].*

Figure 1-24. Timing Chart of Interrupt Acceptance and Interrupt Return Instruction

Example : Correspondence between vector table address for INTTBT and the entry address of the interrupt service program.




A maskable interrupt is not accepted until the IMF is set to "1" even if a maskable interrupt of higher priority than that of the current interrupt being serviced.
When nested interrupt service is necessary, the IMF is set to "1" in the interrupt service program. In this case, acceptable interrupt sources are selectively enabled by the individual interrupt enable flags. However, an acceptance of external interrupt 0 cannot be disabled by the EF; therefore, if disablement is necessary, either the external interrupt function of the $\overline{\text{INT0}}$ pin must be disabled with INT0EN in the external interrupt control register (EINTCR) or interrupt processing must be avoided by the program.

Example 1 : Disables an external interrupt 0 using INT0EN:

```
LD      (EINTCR), 00000000B ;  INT0EN←0
```

Example 2 : Disables the processing of external interrupt 0 under the software control (using bit 0 at address $00F0_H$ as the interrupt processing disable switch):

```
PINT0 :   TEST    (00F0H) . 0     ;  Returns without interrupt processing if (00F0H)0 = 1
          JRS     T, SINT0
          RETI
SINT0 :   Interrupt processing
          RETI
            :
VINT0 :   DW      PINT0
```

During interrupt acceptance processing, the program counter and the program status word are automatically saved on the stack, but not the accumulator and other registers. These registers are saved by the program if necessary. Also, when nesting multiple interrupt services, it is necessary to avoid using the same data memory area for saving registers.
The following method is used to save/restore the general-purpose registers:

① General-purpose register save/restore by register bank changeove:
General-purpose registers can be saved at high-speed by switching to a register bank that is not in use. Normally, bank 0 is used for the main task and banks 1 to 15 are assigned to interrupt service tasks. To increase the efficiency of data memory utilization, the same bank is assigned for interrupt sources which are not nested.
The switched bank is automatically restored by executing an interrupt return instruction [RETI] or [RETN]. Therefore, it is not necessary for a program to save the RBS.

Example : Register Bank Changeover

```
PINTxx :  LD      RBS, n          ;  Switches to bank n (1μs at 8MHz)
          interrupt processing
          RETI                    ;  Restores bank and Returns
```

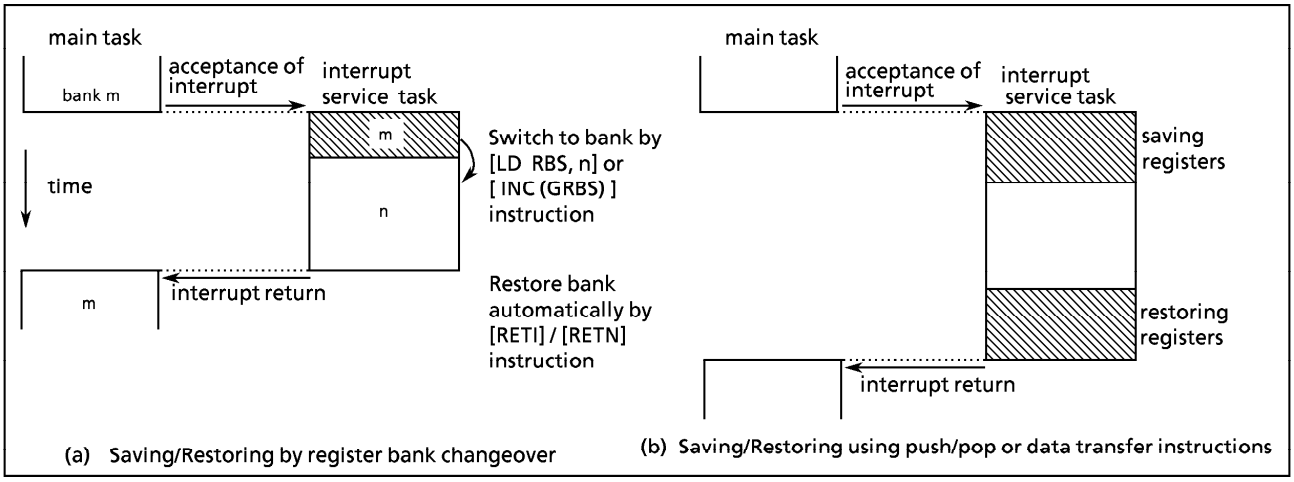
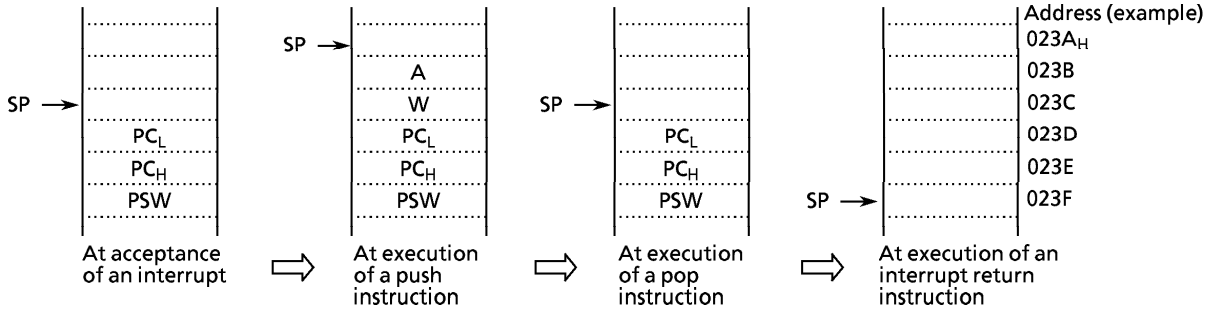



Figure 1-25. Saving/Restoring General-purpose Registers

② General-purpose register save/restore using push and pop instructions:
To save only a specific register, and when the same interrupt source occurs more than once, the general-purpose registers can be saved/restored using push/pop instructions.

Example : Register save using push and pop instructions

```
PINTxx :    PUSH    WA          ; Save WA register pair
            interrupt processing
            POP     WA          ; Restore WA register pair
            RETI                ; Return
```




③ General-purpose registers save/restore using data transfer instruction:
Data transfer instructions can be used to save only a specific general-purpose register during processing of a single interrupt.

Example : Saving/restoring a register using data transfer instructions

```
PINTxx :    LD      (GSAVA), A  ; Save A register
            interrupt processing
            LD      A, (GSAVA)  ; Restore A register
            RETI                ; Return
```

The interrupt return instructions [RETI] / [RETN] perform the following operations.

| [RETI] Maskable interrupt return | [RETN] Non-maskable interrupt return |
|---|---|
| ① The contents of the program counter and the program status word are restored from the stack. | ① The contents of the program counter and program status word are restored from the stack. |
| ② The stack pointer is incremented 3 times. | ② The stack pointer is incremented 3 times. |
| ③ The interrupt master enable flag is set to "1". | ③ The interrupt master enable flag is set to "1" only when a non-maskable interrupt is accepted in interrupt enable status. However, the interrupt master enable flag remains at "0" when so clear by an interrupt service program. |

Interrupt requests are sampled during the final cycle of the instruction being executed. Thus, the next interrupt can be accepted immediately after the interrupt return instruction is executed.

*Note : When the interrupt processing time is longer than the interrupt request generation time, the interrupt service task is performed but not the main task.*

## 1.9.2 External Interrupts

The 87C841/CC41/CH41/CK41/CM41 each have six external interrupt inputs ($\overline{INT0}$, INT1, INT2, INT3, INT4, and $\overline{INT5}$). Four of these are equipped with digital noise rejection circuits (pulse inputs of less than a certain time are eliminated as noise). Edge selection is also possible with INT1, INT2, INT3 and INT4.

The $\overline{INT0}$/P10 pin can be configured as either an external interrupt input pin or an input/output port, and is configured as an input port during reset.

Edge selection, noise rejection control and $\overline{INT0}$/P10 pin function selection are performed by the external interrupt control register (EINTCR). When INT0EN = 0, the $IL_3$ will not be set even if the falling edge of $\overline{INT0}$ pin input is detected.

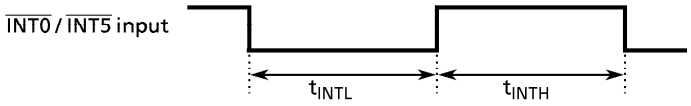
Table 1-3. External Interrupts

| Source | Pin | Secondary function pin | Enable conditions | Edge | Digital noise reject |
|---|---|---|---|---|---|
| INT0 | $\overline{INT0}$ | P10 | IMF = 1, INT0EN = 1 | falling edge | — (hysteresis input) |
| INT1 | INT1 | P11 | IMF · $EF_5$ = 1 | falling edge or rising edge | Pulses of less than 15/fc or 63/fc [s] are eliminated as noise. Pulses equal to or more than 48/fc [s] or 192/fc [s] are regarded as signals. |
| INT2 | INT2 | P12/TC1 | IMF · $EF_7$ = 1 | | Pulses of less than 7/fc [s] are eliminated as noise. Pulses equal to or more than 24/fc [s] are regarded as signals. |
| INT3 | INT3 | P50/TC3 | IMF · $EF_{11}$ = 1 | | |
| INT4 | INT4 | P51/TC4 | IMF · $EF_{12}$ = 1 | | |
| INT5 | $\overline{INT5}$ | P20/$\overline{STOP}$ | IMF · $EF_{15}$ = 1 | falling edge | — (hysteresis input) |

*Note 1 : The noise rejection function is turned off in the SLOW and SLEEP modes. Also, the noise reject times are not constant for pulses input while transiting between operating modes (NORMAL2↔SLOW)*

*Note 2 : The noise rejection function is also affected for timer/counter input (TC1 and TC3 pins).*

*Note 3 : The pulse width (both "H" and "L" level) for input to the $\overline{INT0}$ and $\overline{INT5}$ pins must be over 1 machine cycle.*




$t_{INTL}$, $t_{INTH}$ > tcyc (*Note : tcyc = 4/fm [s]*)

*Note 4 : If a noiseless signal is input to the external interrupt pin in the NORMAL 1/2 or IDLE 1/2 mode, the maximum time from the edge of input signal until the IL is set is as follows :*

① INT1 pin 49/fc [s] (INT1NC = 1), 193/fc [s] (INT1NC = 0)

② INT2,INT3, INT4 pins 25/fc [s]

*Note 5 : When high-impedance is specified for port output in stop mode, port input is forcibly fixed to low level internally. Thus, interrupt latches of external interrupt inputs except $\overline{INT5}$ (P20/$\overline{STOP}$) which are also used as ports may be set to "1". To specify high-impedance for port output in stop mode, first disable interrupt service (IMF = 0) , activate stop mode. After releasing stop mode, clear interrupt latches using load instruction, then, enable interrupt service.*

Example : Activating stop mode.

```
LD   (SYSCR1),01000000B      ; OUTEN ← 0  (specifies high-impedance)
DI                           ; IMF ← 0  (disables interrupt service)
SET (SYSCR1).STOP            ; STOP ← 1  (activates stop mode)
LDW (IL),1110011101010111B ; IL12,11,7,5,3 ← 0  (clears interrupt latches)
EI                           ; IMF ← 1  (enables interrupt service)
```

**EINTCR** ($0037_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| INT1 NC | INT0 EN | | INT4 ES | INT3 ES | INT2 ES | INT1 ES | |

(Initial value : 00*0 000*)

| | | | |
|---|---|---|---|
| INT1NC | Noise reject time select | 0 : Pulses of less than 63/fc [s] are eliminated as noise<br>1 : Pulses of less than 15/fc [s] are eliminated as noise | R/W |
| INT0EN | P10/$\overline{INT0}$ pin configuration | 0 : P10 input/output port<br>1 : $\overline{INT0}$ pin (Port P10 should be set to an input mode) | |
| INT4 ES<br>INT3 ES<br>INT2 ES<br>INT1 ES | INT4 to INT1 edge select | 0 : Rising edge<br>1 : Falling edge | |

*Note 1 : fc ; High-frequency clock [Hz] * ; don't care*

*Note 2 : Edge detection during switching edge selection is invalid.*

*Note 3 : Do not change EINTCR when IMF = 1. After changing EINTCR, interrupt latches of external interrupt inputs must be cleared to "0" using load instruction.*

*Note 4 : In order to change of external interrupt input by rewriting the contents of INT2ES, INT3ES and INT4ES during NORMAL1/2 mode, clear interrupt latches of external interrupt inputs (INT2, INT3 and INT4) after 8 machine cycles from the time of rewriting. During SLOW mode, 3 machine cycles are required.*

*Note 5 : In order to change an edge of timer counter input by rewriting the contents of INT2ES, INT3ES and INT4ES during NORMAL1/2 mode, rewrite the contents after timer counter is stopped (TC*s = 0), that is, interrupt disable state. Then, clear interrupt laches of external interrupt inputs (INT2, INT3 and INT4) after 8 machine cycles from the time of rewriting to change to interrupt enable state. Finally, start timer counter. During SLOW mode, 3 machine cycles are required.*

Example : When changing TC1 pin inputs edge in external trigger timer mode from rising edge to falling edge.

```
                LD (TC1CR),01001000B    ; TC1S ← 00 (stop TC1)
                DI                      ; IMF ← 0 (disable interrupt service)
                LD (EINTCR),00000100B   ; INT2ES ← 1 (change edge selection)
                NOP
8 machine       to
  cycles        NOP
                LD (ILL),01111111B      ; IL7 ← 0 (clear interrupt latch)
                EI                      ; IMF ← 1 (enable interrupt service)
                LD (TC1CR),01111000B    ; TC1S ← 11 (start TC1)
```

*Note 6 : If changing the contents of INT1ES during NORMAL1/2 mode, interrupt latch of external interrupt input INT1 must be cleared after 14 machine cycles (when INT1NC = 1) or 50 machine cycles (when INT1NC = 0) from the time of changing. During SLOW mode, 3 machine cycles are required.*

Figure 1-26. External Interrupt Control Register

### 1.9.3 Software Interrupt (INTSW)

Executing the [SWI] instruction generates a software interrupt and immediately starts interrupt processing (INTSW is highest prioritized interrupt). However, if processing of a non-maskable interrupt is already underway, executing the SWI instruction will not generate a software interrupt but will result in the same operation as the [NOP] instruction. Thus, the [SWI] instruction behaves like the [NOP] instruction.

Use the [SWI] instruction only for detection of the address error or for debugging.

① Address Error Detection

$FF_H$ is read if for some cause such as noise the CPU attempts to fetch an instruction from a non-existent memory address. Code $FF_H$ is the SWI instruction, so a software interrupt is generated and an address error is detected. The address error detection range can be further expanded by writing $FF_H$ to unused areas of the program memory. Address trap reset is generated for instruction fetch from a specific address (0000 to $043F_H$).

*Note : The fetch data from addresses $7F80_H$ to $7FFF_H$ (test ROM area) for 87CK41/CM41, $BF80_H$ to $BFFF_H$ for 87C841/CC41/CH41 is not "$FF_H$".*

② Debugging

Debugging efficiency can be increased by placing the SWI instruction at the software break point setting address.

## 1.10 Watchdog Timer (WDT)

The watchdog timer rapidly detects the CPU malfunction such as endless looping caused by noise or the like, and resumes the CPU to the normal state.

The watchdog timer signal for detecting malfunction can be selected either a reset output or a non-maskable interrupt request. However, selection is possible only once after reset. At first the reset output is selected.

When the watchdog timer is not being used for malfunction detection, it can be used as a timer to generate an interrupt at fixed intervals.

### 1.10.1 Watchdog Timer Configuration

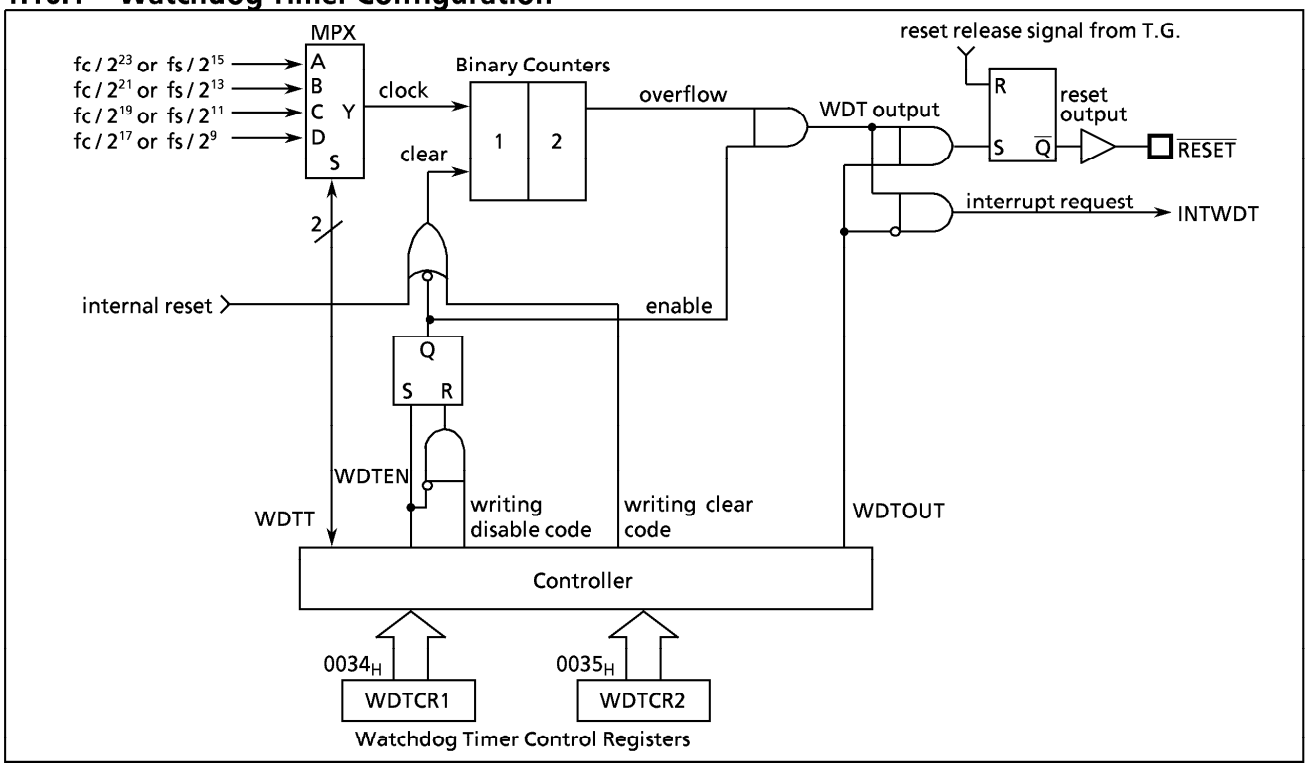



Figure 1-27. Watchdog Timer Configuration

### 1.10.2 Watchdog Timer Control

Figure 1-28 shows the watchdog timer control registers (WDTCR1, WDTCR2). The watchdog timer is automatically enabled after reset.

(1) Malfunction detection methods using the watchdog timer

The CPU malfunction is detected as follows.

① Setting the detection time, selecting output, and clearing the binary counter.

② Repeatedly clearing the binary counter within the setting detection time.

If the CPU malfunction occurs for any cause, the watchdog timer output will become active at the rising of an overflow from the binary counters unless the binary counters are cleared. At this time, when WDTOUT = 1 a reset is generated, which drives the $\overline{\text{RESET}}$ pin low to reset the internal hardware and the external circuits. When WDTOUT = 0, a watchdog timer interrupt (INTWDT) is generated.

The watchdog timer temporarily stops counting in the STOP mode including warm-up or IDLE mode, and automatically restarts (continues counting) when the STOP/IDLE mode is released.

Example : Sets the watchdog timer detection time to $2^{21}$/fc [s] and resets the CPU malfunction.

```
        LD      (WDTCR1), 00001101B   ; WDTT←10, WDTOUT←1
        LD      (WDTCR2), 4EH         ; Clears the binary counters
Within WDT                            (always clear immediately after changing WDTT)
detection time
        LD      (WDTCR2), 4EH         ; Clears the binary counters
Within WDT
detection time
        LD      (WDTCR2), 4EH         ; Clears the binary counters
```

Watchdog Timer Control Register 1

**WDTCR1** ($0034_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | WDT EN | WDTT | | WDT OUT |

(Initial value : **** 1001)

| | | | |
|---|---|---|---|
| WDTEN | Watchdog timer enable/disable | 0 : Disable (It is necessary to write the disable code to WDTCR2)<br>1 : Enable | write only |
| WDTT | Watchdog timer detection time | 00 : $2^{25}$ / fc or $2^{17}$ / fs [s]<br>01 : $2^{23}$ / fc or $2^{15}$ / fs<br>10 : $2^{21}$ / fc or $2^{13}$ / fs<br>11 : $2^{19}$ / fc or $2^{11}$ / fs | |
| WDTOUT | Watchdog timer output select | 0 : Interrupt request<br>1 : Reset output | |

*Note 1 : WDTOUT cannot be set to "1" by program after clearing WDTOUT to "0".*
*Note 2 : fc ; High-frequency clock [Hz] fs ; Low-frequency clock [Hz] * ; don't care*
*Note 3 : WDTCR1 is a write-only register and must not be used with any of read-modify-write instructions.*
*Note 4 : Disable the watchdog timer or clear the counter just before switching to STOP mode.*
*When the counter is cleared just before switching to STOP mode, clear the counter again subsequently to releasing STOP mode.*

Watchdog Timer Control Register 2

**WDTCR2** ($0035_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

(Initial value : **** ****)

| | | | |
|---|---|---|---|
| WDTCR2 | Watchdog timer control code write register | $4E_H$ : Watchdog timer binary counter clear (clear code)<br>$B1_H$ : Watchdog timer disable (disable code)<br>others : Invalid | write only |

*Note 1 : The disable code is invalid unless written when WDTEN = 0.*
*Note 2 : * ; don't care*

Figure 1-28. Watchdog Timer Control Registers

Table 1-4. Watchdog Timer Detection Time

| Operating mode | | | Detection time | |
|---|---|---|---|---|
| NORMAL1 | NORMAL2 | SLOW | At fc = 8 MHz | At fs = 32.768 kHz |
| $2^{25}$ / fc [s] | $2^{25}$ / fc, $2^{17}$ / fs | $2^{17}$ / fs | 4.194 s | 4 s |
| $2^{23}$ / fc | $2^{23}$ / fc, $2^{15}$ / fs | $2^{15}$ / fs | 1.048 ms | 1 s |
| $2^{21}$ / fc | $2^{21}$ / fc, $2^{13}$ / fs | —— | 262.1 ms | 250 ms |
| $2^{19}$ / fc | $2^{19}$ / fc, $2^{11}$ / fs | —— | 65.5 ms | 62.5 ms |

(2) Watchdog Timer Enable

The watchdog timer is enabled by setting WDTEN (bit 3 in WDTCR1) to "1". WDTEN is initialized to "1" during reset, so the watchdog timer operates immediately after reset is released.

Example : Enables watchdog timer

```
LD      (WDTCR1), 00001000B     ; WDTEN←1
```

(3) Watchdog Timer Disable

The watchdog timer is disabled by writing the disable code ($B1_H$) to WDTCR2 after clearing WDTEN (bit 3 in WDTCR1) to "0". The watchdog timer is not disabled if this procedure is reversed and the disable code is written to WDTCR2 before WDTEN is cleared to "0". The watchdog timer is halted temporarily in STOP mode (including warm-up) and IDLE mode, and restarts automatically after STOP or IDLE mode is released.

During disabling the watchdog timer, the binary counters are cleared to "0".

Example : Disables watchdog timer

```
LDW     (WDTCR1) , 0B101H       ; WDTEN←0, WDTCR2←disable code
```

### 1.10.3 Watchdog Timer Interrupt (INTWDT)

This is a pseudo non-maskable interrupt which can be accepted regardless of the contents of the EIR. If a watchdog timer interrupt or a software interrupt is already accepted, however, the new watchdog timer interrupt waits until the previous interrupt processing is completed (the end of the [RETN] instruction execution).

The stack pointer (SP) should be initialized before using the watchdog timer output as an interrupt source with WDTOUT.

Example : Watchdog timer interrupt setting up.

```
LD      SP, 013FH               ; Sets the stack pointer
LD      (WDTCR1) , 00001000B    ; WDTOUT←0
```

### 1.10.4 Watchdog Timer Reset

If the watchdog timer output becomes active, a reset is generated, which drives the $\overline{\text{RESET}}$ pin (sink open drain output) low to reset the internal hardware and the external circuits. The reset output time is $2^{20}$/fc [s] (131 ms at fc = 8MHz). The high-frequency clock oscillator also turns on when a watchdog timer reset is generated in SLOW mode.

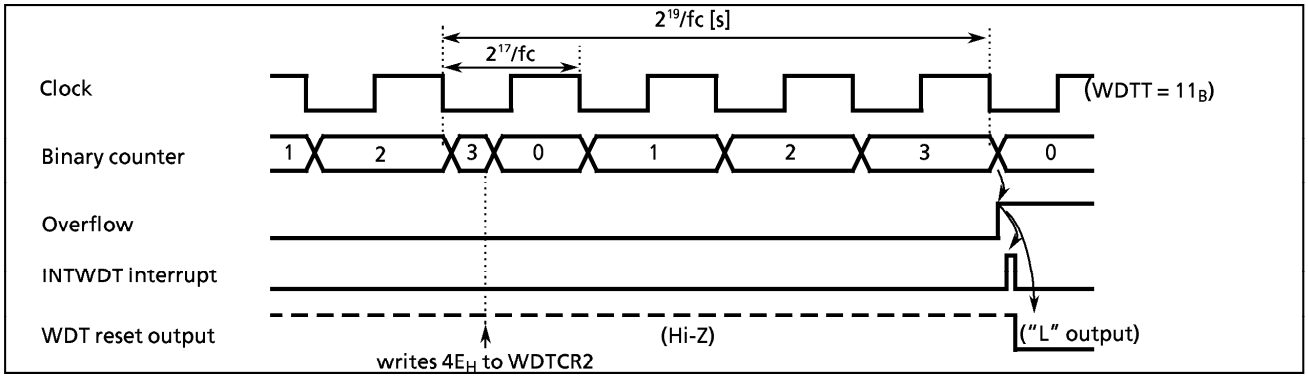



Figure 1-29. Watchdog Timer Interrupt / Reset

## 1.11 Reset Circuit

The 87C841/CC41/CH41/CK41/CM41 each have four types of reset generation procedures: an external reset input, an address trap reset, a watchdog timer reset and a system clock reset. Table 1-5 shows on-chip hardware initialization by reset action. The internal source reset circuit (watchdog timer reset, address trap reset, and system clock reset) is not initialized when power is turned on. Thus, output from the $\overline{\text{RESET}}$ pin may go low ($2^{20}$/fc [s] (131 ms at 8MHz) when power is turned on.

Table 1-5. Initializing Internal Status by Reset Action

| On-chip Hardware | | Initial Value | On-chip Hardware | Initial Value |
|---|---|---|---|---|
| Program counter | (PC) | $(FFFF_H) \cdot (FFFE_H)$ | Divider of Timing generator | 0 |
| Register bank selector | (RBS) | 0 | Watchdog timer | Enable |
| Jump status flag | (JF) | 1 | Output latches of I/O ports | Refer to I/O port circuitry |
| Interrupt master enable flag | (IMF) | 0 | Control registers | Refer to each of control register |
| Interrupt individual enable flags | (EF) | 0 | | |
| Interrupt latches | (IL) | 0 | | |

### 1.11.1 External Reset Input

When the $\overline{RESET}$ pin is held at low for at least 3 machine cycles (12/fc [s]) with the power supply voltage within the operating voltage range and oscillation stable, a reset is applied and the internal state is initialized.
When the $\overline{RESET}$ pin input goes high, the reset operation is released and the program execution starts at the vector address stored at addresses $FFFE_H$ to $FFFF_H$.
The $\overline{RESET}$ pin contains a Schmitt trigger (hysteresis) with an internal pull-up resistor. A simple power-on-reset can be applied by connecting an external capacitor and a diode.

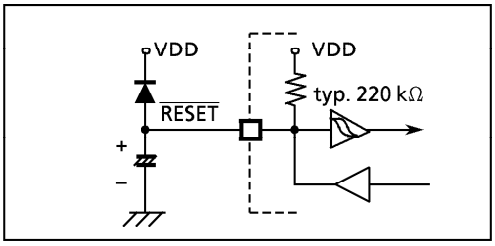
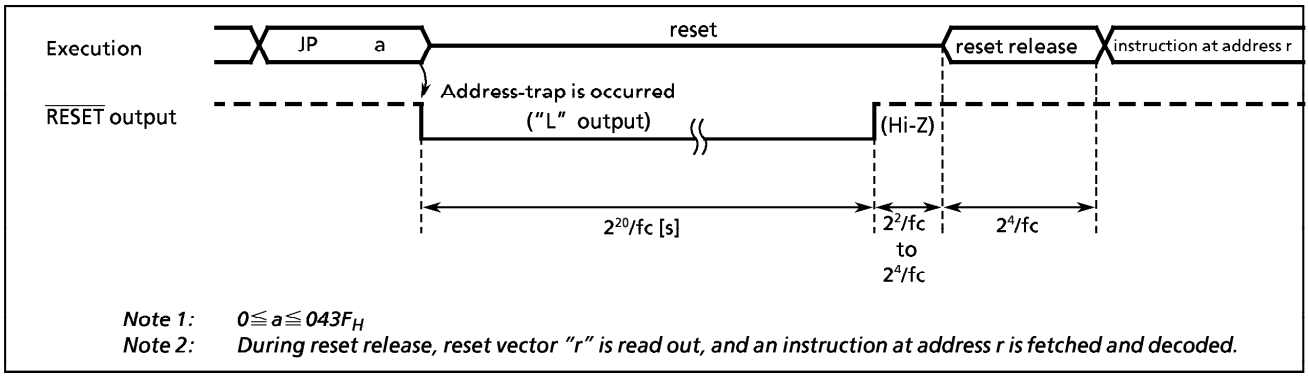



Figure 1-30. Simple Power-on-Reset Circuitry

### 1.11.2 Address-Trap-Reset

An address-trap-reset is one of fail-safe function that detects CPU malfunction such as endless looping caused by noise or the like, and returns the CPU to the normal state. If the CPU attempts to fetch an instruction from a specific address (0000 to $043F_H$), an internal reset (called address-trap-reset) will be generated. Then, the $\overline{RESET}$ pin output will go low. The reset time is $2^{20}$/fc [s] (131 ms at 8 MHz) .




*Note 1:* $0 \leqq a \leqq 043F_H$
*Note 2:* *During reset release, reset vector "r" is read out, and an instruction at address r is fetched and decoded.*

Figure 1-31. Address-Trap-Reset

### 1.11.3 Watchdog Timer Reset

Refer to Section "1.10 Watchdog Timer".

### 1.11.4 System-Clock-Reset

Clearing both XEN and XTEN (bits 7 and 6 in SYSCR2) to "0" stops both high-frequency and low-frequency oscillation, and causes the MCU to deadlock. This can be prevented by automatically generating a reset signal whenever XEN = XTEN = 0 is detected to continue the oscillation. Then, the $\overline{RESET}$ pin output goes low from high-impedance. The reset time is $2^{20}$/fc [s] (131 ms at 8MHz).

## 2. ON-CHIP PERIPHERALS FUNCTIONS

### 2.1 Special Function Registers (SFR) and Data Buffer Registers (DBR)

The TLCS-870 Series uses the memory mapped I/O system, and all peripheral control and data transfers are performed through the special function registers (SFR) and data buffer registers (DBR).
The SFR are mapped to addresses $0000_H$ to $003F_H$, and the DBR to addresses $0F80_H$ to $0FFF_H$.
Figure 2-1 shows the 87C841/CC41/CH41/CK41/CM41 SFRs and DBRs.

| Address | Read | Write |
|---|---|---|
| $0000_H$ | P0 port | |
| 01 | P1 Port | |
| 02 | P2 Port | |
| 03 | P3 Port | |
| 04 | P4 Port | |
| 05 | P5 Port | |
| 06 | P6 Port | |
| 07 | P7 Port | |
| 08 | reserved | |
| 09 | reserved | |
| 0A | – | P0CR (P0 I/O control) |
| 0B | – | P1CR (P1 I/O control) |
| 0C | – | P6CR (P6 I/O control) |
| 0D | – | P7CR (P7 I/O control) |
| 0E | ADCCR (A/D converter control) | |
| 0F | ADCDR1 (A/D conv. upper 8-bit result) | – |
| 10 | – | $TREG1A_L$ (Timer register 1A) |
| 11 | – | $TREG1A_H$ |
| 12 | $TREG1B_L$ (Timer register 1B) | |
| 13 | $TREG1B_H$ | |
| 14 | – | TC1CR (TC1 control) |
| 15 | – | TC2CR (TC2 control) |
| 16 | – | $TREG2_L$ (Timer register 2) |
| 17 | – | $TREG2_H$ |
| 18 | TREG3A (Timer register 3A) | |
| 19 | TREG3B (Timer register 3B) | – |
| 1A | – | TC3CR (TC3 control) |
| 1B | – | TREG4 (Timer register 4) |
| 1C | – | TC4CR (TC4 control) |
| 1D | reserved | |
| 1E | reserved | |
| 1F | reserved | |

| Address | Read | Write |
|---|---|---|
| $0020_H$ | SIO1SR (SIO1 status) | SIO1CR1 (SIO1 control) |
| 21 | – | SIO1CR2 |
| 22 | SIO2SR (SIO2 status) | SIO2CR1 (SIO2 control) |
| 23 | – | SIO2CR2 |
| 24 | ADCDR2L (A/D conv. lower 8-bit result) | |
| 25 | ADCDR2H (A/D conv. upper 2-bit result) | |
| 26 | | HPWMCR (HPWM control) |
| 27 | | HPWMDR0 (HPWM0 data register) |
| 28 | | HPWMDR1 (HPWM1 data register) |
| 29 | reserved | |
| 2A | reserved | |
| 2B | reserved | |
| 2C | reserved | |
| 2D | reserved | |
| 2E | reserved | |
| 2F | reserved | |
| 30 | reserved | |
| 31 | reserved | |
| 32 | reserved | |
| 33 | reserved | |
| 34 | – | WDTCR1 (WDT control) |
| 35 | – | WDTCR2 |
| 36 | TBTCR | (TBT / TG / DVO control) |
| 37 | EINTCR | (External interrupt control) |
| 38 | SYSCR1 | (System control) |
| 39 | SYSCR2 | |
| 3A | $EIR_L$ | (Interrupt enable register) |
| 3B | $EIR_H$ | |
| 3C | $IL_L$ | (Interrupt latch) |
| 3D | $IL_H$ | |
| 3E | reserved | |
| 3F | PSW (Program status word) | RBS (Register bank selector) |

(a) Special Function Registers

| Address | Read | Write |
|---|---|---|
| $0F80_H$ | reserved | |
| | reserved | |
| 0FEF | reserved | |
| 0FF0 | SIO1 transmit and receive data buffer | |
| F1 | | |
| F2 | | |
| F3 | | |
| F4 | | |
| F5 | | |
| F6 | | |
| F7 | | |
| 0FF8 | SIO2 transmit and receive data buffer | |
| F9 | | |
| FA | | |
| FB | | |
| FC | | |
| FD | | |
| FE | | |
| FF | | |

(b) Data Buffer Registers

*Note 1 : Do not access reserved areas by the program.*
*Note 2 : – : Cannot be accessed.*
*Note 3 : When defining address $003F_H$ with assembler symbols, use GPSW and GRBS.*
*Note 4 : Write-only registers and interrupt latches cannot use the read-modify-write instructions (bit manipulation instructions such as SET, CLR, etc. and logical operation instructions such as AND, OR, etc.)*

Figure 2-1. SFR & DBR

## 2.2 I/O Ports

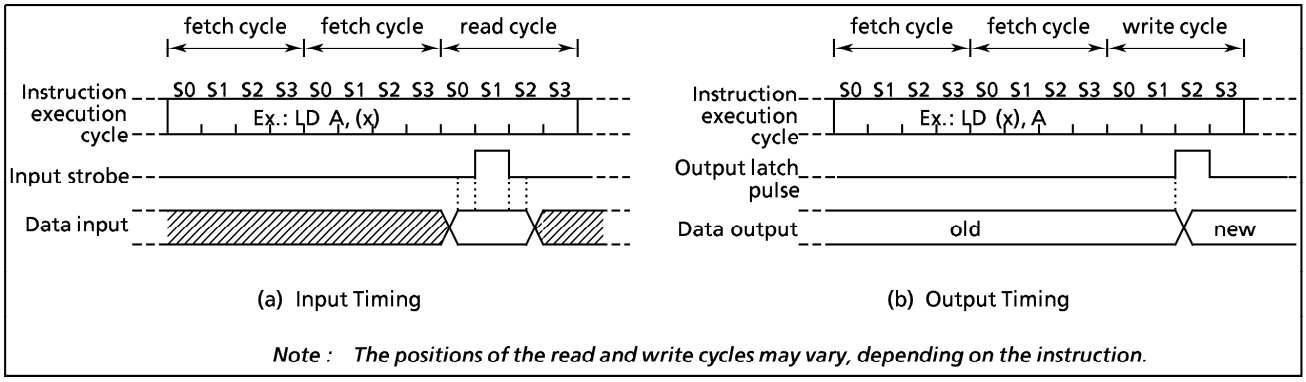
The 87C841/CC41/CH41/CK41/CM41 have 8 parallel input/output ports (56pins) each as follows:

| | Primary Function | Secondary Functions |
|---|---|---|
| Port P0 | 8-bit I/O port | ——— |
| Port P1 | 8-bit I/O port | external interrupt input, timer/counter input/output, and divider output |
| Port P2 | 3-bit I/O port | low-frequency resonator connections, external interrupt input, and STOP mode release signal input |
| Port P3 | 8-bit I/O port | ——— |
| Port P4 | 8-bit I/O port | serial interface |
| Port P5 | 5-bit I/O port | external interrupt input, and timer/counter input/output, high-speed PWM output |
| Port P6 | 8-bit I/O port | analog input |
| Port P7 | 8-bit I/O port | analog input |

Each output port contains a latch, which holds the output data. All input ports do not have latches, so the external input data should either be held externally until read or reading should be performed several times before processing. Figure 2-2 shows input/output timing examples.

External data is read from an I/O port in the S1 state of the read cycle during execution of the read instruction. This timing can not be recognized from outside, so that transient input such as chattering must be processed by the program.

Output data output changes in the S2 state of the write cycle during execution of the instruction which writes to an I/O port.




*Note : The positions of the read and write cycles may vary, depending on the instruction.*

Figure 2-2. Input/Output Timing (Example)

When reading an I/O port except programmable I/O ports, whether the pin input data or the output latch contents are read depends on the instructions, as shown below:

(1) Instructions that read the output latch contents
- ① XCH r, (src)
- ② CLR/SET/CPL (src).b
- ③ CLR/SET/CPL (pp).g
- ④ LD (src).b, CF
- ⑤ LD (pp) . b, CF
- ⑥ ADD/ADDC/SUB/SUBB/AND/OR/XOR (src), n
- ⑦ (src) side of ADD/ADDC/SUB/SUBB/AND/OR/XOR (src), (HL)

(2) Instructions that read the pin input data
- ① Instructions other than the above (1)
- ② (HL) side of ADD/ADDC/SUB/SUBB/AND/OR/XOR (src), (HL)

### 2.2.1 Port P0 (P07 to P00)

Port P0 is an 8-bit general-purpose input/output port which can be configured as either an input or an output in one-bit unit under software control. Input/output mode is specified by the corresponding bit in the port P0 input/output control register (P0CR). Port P0 is configured as an input if its corresponding P0CR bit is cleared to "0", and as an output if its corresponding P0CR bit is set to "1".

During reset, P0CR is initialized to "0", which configures port P0 as input. The P0 output latches are also initialized to "0". Data is written into the output latch regardless of the P0CR contents. Therefore initial output data should be written into the output latch before setting P0CR.

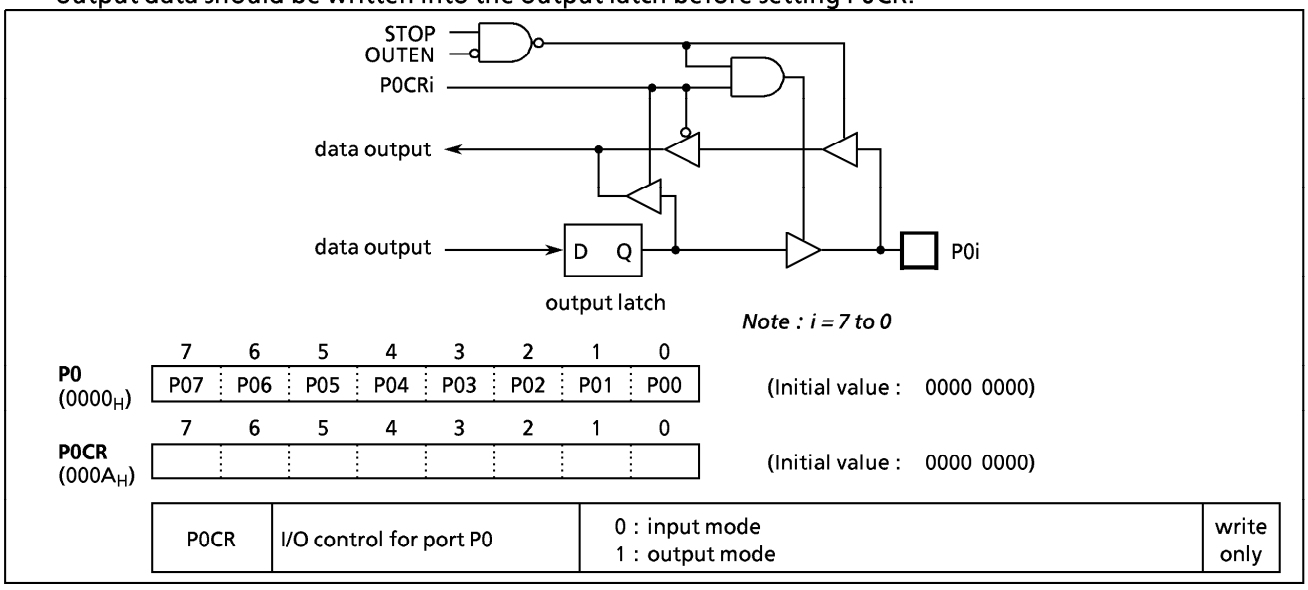



Figure 2-3. Port P0 and P0CR

Example : Setting the upper 4 bits of port P0 as an input port and the lower 4 bits as an output port (Initial output data are $1010_B$).

```
LD      (P0), 00001010B     ; Sets initial data to P0 output latches
LD      (P0CR), 00001111B   ; Sets the port P0 input/output mode
```

### 2.2.2 Port P1 (P17 - P10)

Port P1 is an 8-bit input/output port which can be configured as an input or an output in one-bit unit under software control. Input/output mode is specified by the corresponding bit in the port P1 input/output control register (P1CR). Port P1 is configured as an input if its corresponding P1CR bit is cleared to "0", and as an output if its corresponding P1CR bit is set to "1". During reset, the P1CR is initialized to "0", which configures port P1 as an input . The P1 output latches are also initialized to "0". Data is written into the output latch regardless of P1CR contents. Therfore initial output data should be written into the output latch before setting P1CR. Port P1 is also used as an external interrupt input, a timer/counter input, and a divider output. When used as secondary function pin, the input pins should be set to the input mode, and the output pins should be set to the output mode and beforehand the output latch should be set to "1".

It is recommended that pins P11 and P12 should be used as external interrupt inputs, timer/counter input, or input ports. The interrupt latch is set at the rising or falling edge of the output when used as output ports.

Pin P10 ($\overline{INT0}$) can be configured as either an I/O port or an external interrupt input with INT0EN (bit 6 in EINTCR). During reset, pin P10 ($\overline{INT0}$) is configured as an input port P10.

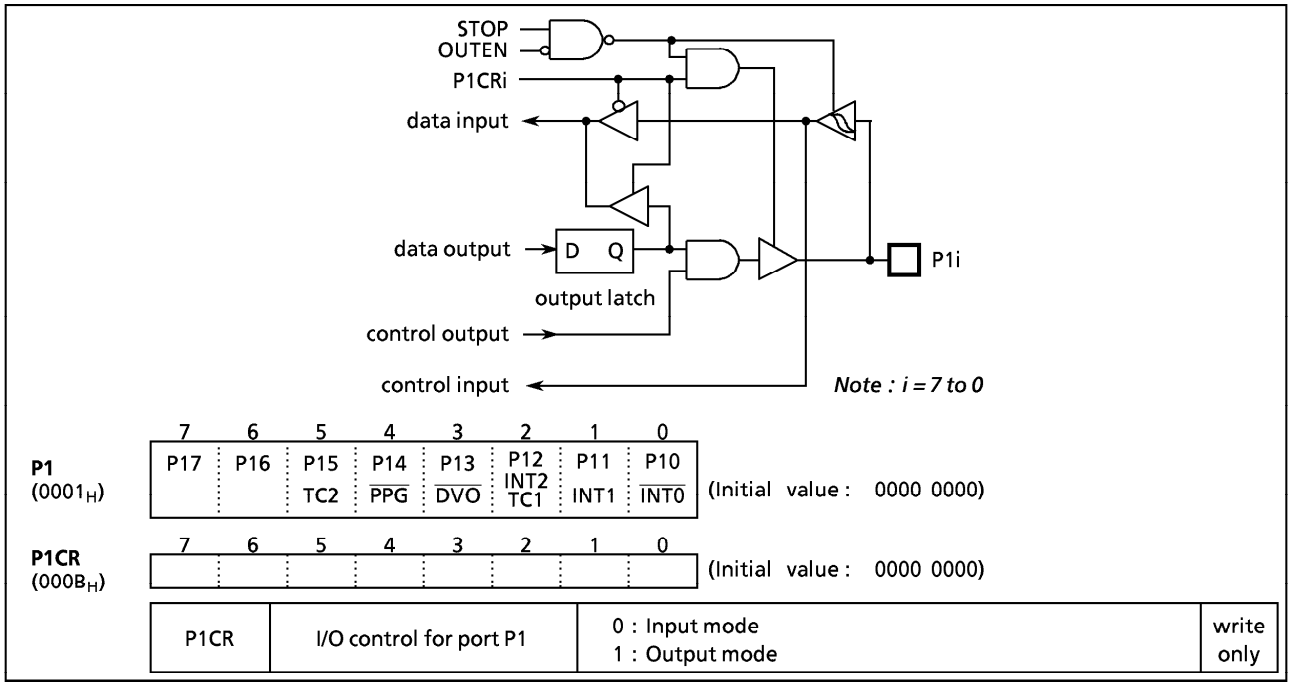



Figure 2-4. Port P1 and P1CR

Example : Sets P17, P16 and P14 as output ports, P13 and P11 as input ports, and the others as function pins. Internal output data is "1" for the P17 and P14 pins, and "0" for the P16 pin.

```
LD      (EINTCR), 01000000B     ; INT0EN←1
LD      (P1), 10111111B         ; P17←1, P14←1, P16←0
LD      (P1CR), 11010000B
```

### 2.2.3 Port P2 (P22 to P20)

Port P2 is a 3-bit input/output port. It is also used as an external interrupt input, and low-frequency crystal connection pins When used as an input port, or a secondary function pin, the output latch should be set to "1". During reset , the output latches are initialized to "1".

A low-frequency crystal (32.768 kHz) is connected to pins P21 (XTIN) and P22 (XTOUT) in the dual-clock mode. In the single-clock mode, pins P21 and P22 can be used as normal input/output ports.

It is recommended that pin P20 should be used as an external interrupt input, a STOP mode release signal input, or an input port. If used as an output port, the interrupt latch is set on the falling edge of the output pulse.

When a read instruction is executed for port P2, bits 7 to 3 read in as "1".

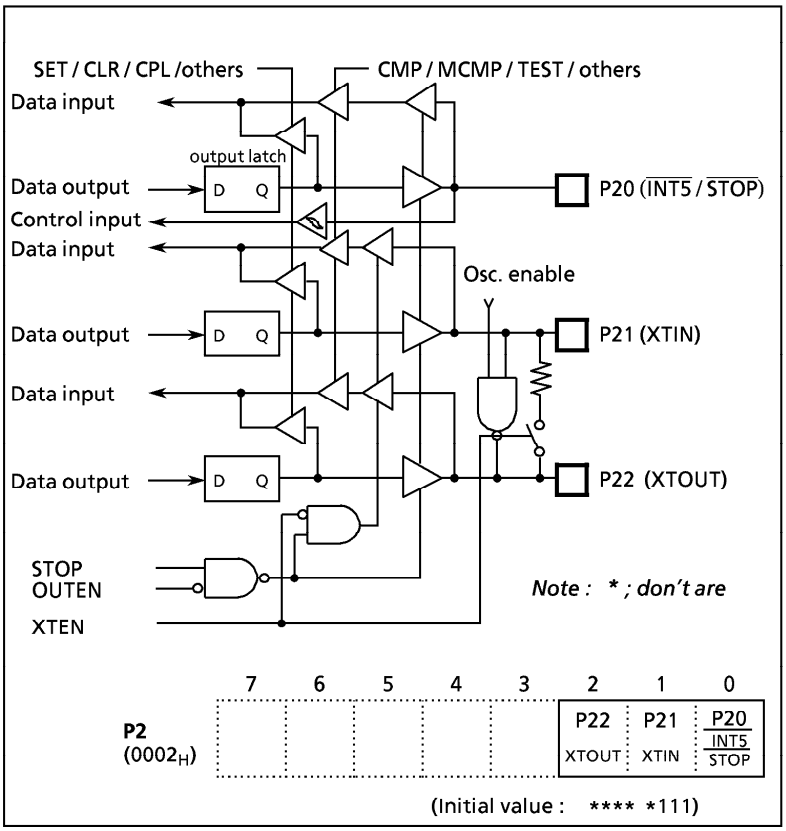



Figure 2-5. Port P2

### 2.2.4 Port P3 (P37 to P30)

Port P3 is an 8-bit input/output port. High current output is available so LEDs can be driven directly. When used as an input port, the output latch should be set to "1". The output latches are initialized to "1" during reset.

Example 1: Output the immediate data $5A_H$ to the P3 port.

```
LD      (P3), 5AH       ; P3←5AH
```

Example 2: Inverts the output of the upper 4bits (P37 - P34) of the P3 port.

```
XOR     (P3), 11110000B
                ; P37 to P34←P37 to P34
```

### 2.2.5 Port P4 (P47 to P40)

Port P4 is an 8-bit input/output port, and is also used as serial interface (SIO1, SIO2) input/output. When used as an input port or a secondary function pin, the output latch should be set to "1". The output latches are initialized to "1" during reset.

### 2.2.6 Port P5 (P54 to P50)

Port P5 is a 5-bit input/output port, and is also used as an external interrupt input, a timer/counter input/output, and a high-speed PWM output. When used as an input port or a secondary function pin, the output latch should be set to "1". The output latches are initialized to "1" during reset.

Bits 7-5 are read in as "1" when a read instruction is executed for the port P5.

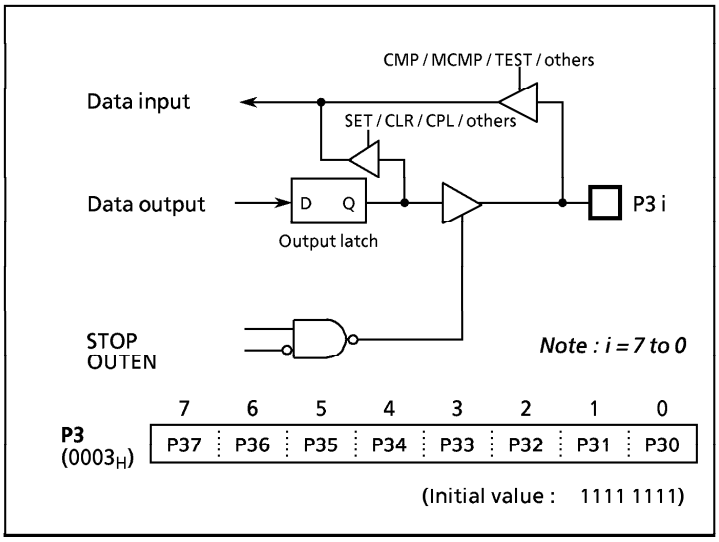



Figure 2-6. Port P3

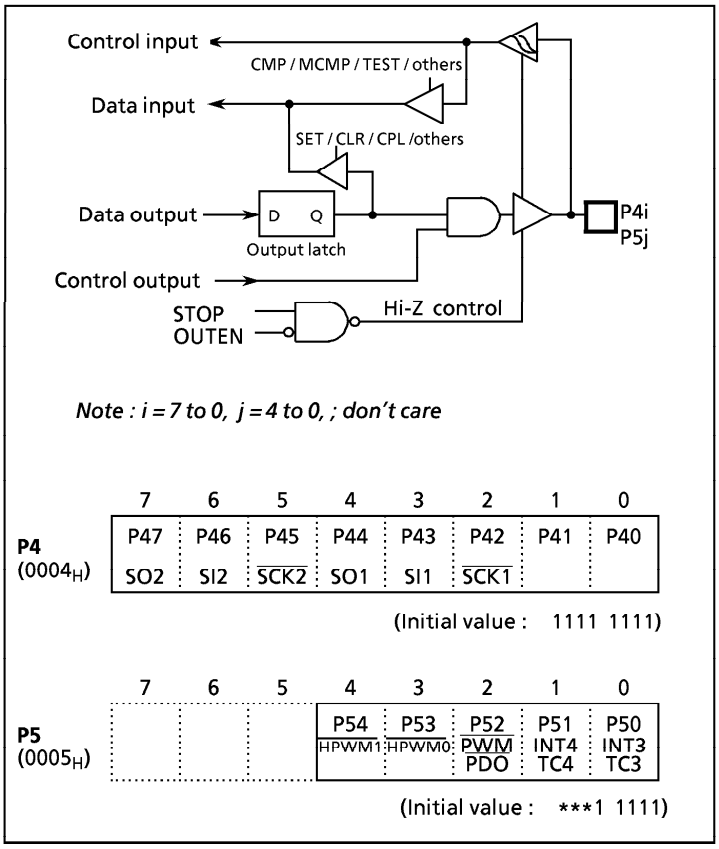



Figure 2-7. Ports P4 and P5

### 2.2.7 Port P6 (P67 to P60)

Port P6 is an 8-bit input/output port which can be configured as an input or an output in one-bit unit under software control. Input/output mode is specified by the corresponding bit in the port P6 input/output control register (P6CR).

Port P6 is also used as an analog input for the A/D converter. When used as an analog input, AINDS (bit 4 in the ADCCR) must be cleared to "0" and its corressponding P6CR bit must be set to "1". In this case, unuse pin as analog input is configured as only input port.
During reset, AINDS is initialized to "0" and all bits of P6CR are initialized to "1", which configures port P6 as analog input. The P6 output latches are initialized to "0". Data is written into the output latch regardless of the P6CR contents. Therefore initial output data should be written into the output latch before setting P6CR.

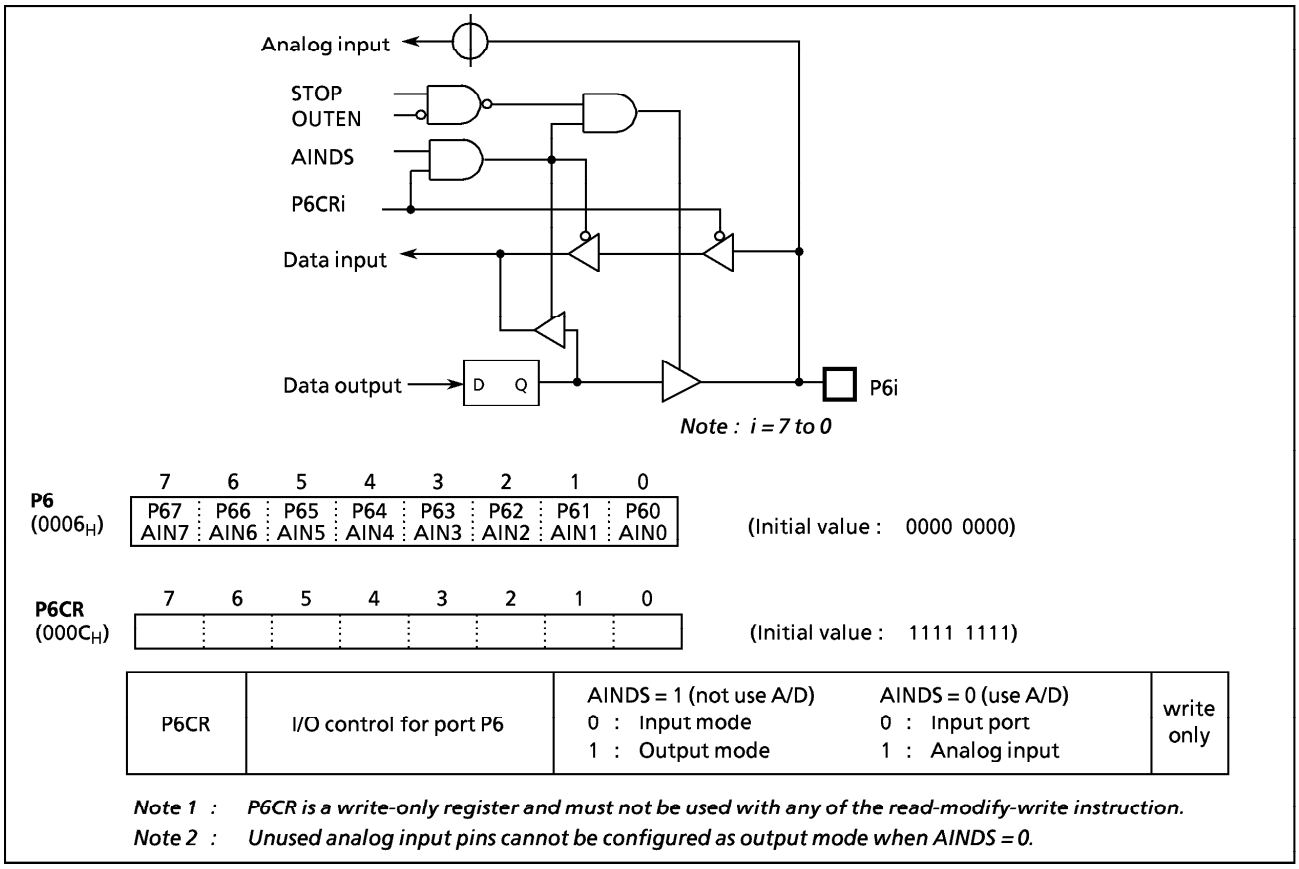


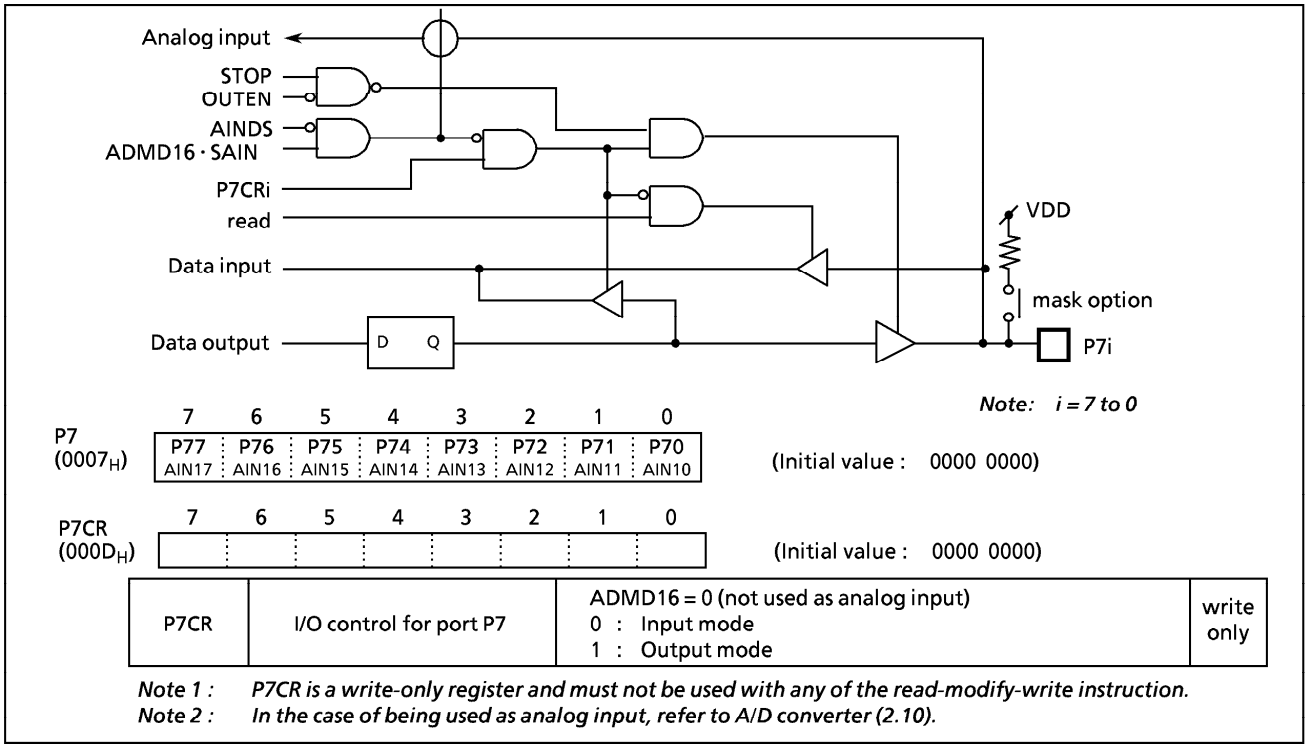

Figure 2-8. Port P6

### 2.2.8 Port P7 (P77 to P70)

Port P7 is an 8-bit input/output port which can be configured as either input or output in one-bit unit under software control. Input/output mode is specified by the corresponding bit in the port P7 input/output control register (P7CR).

Port P7 is also used as an analog input for the A/D converter. When used as an analog input, AINDS (bit 4 in the ADCCR) must be cleared to "0" and ADMD16 (bit 7 in the ADCCR) must be set to "1". In this case, unuse pin as analog input is configured as either input or output port.

During reset, AINDS and ADMD16 are initialized to "0" and all bits of P7CR are initialized to "0", which configures port P7 as input port. The P7 output latches are initialized to "0". Data is written into the output latch regardless of the P7CR contents. Therefore initial output data should be written into the output latch before setting P7CR.




*Note 1 : P7CR is a write-only register and must not be used with any of the read-modify-write instruction.*
*Note 2 : In the case of being used as analog input, refer to A/D converter (2.10).*

Figure 2.9 Port P7 and P7CR

## 2.3 Time Base Timer (TBT)

The time base timer generates time base for key scanning, dynamic displaying, etc. It also provides a time base timer interrupt (INTTBT). The time base timer is controlled by the control register (TBTCR) shown in Figure 2-11.

An INTTBT is generated on the first rising edge of source clock (the divider output of the timing generator) after the time base timer has been enabled. The divider is not cleared by the program; therefore, only the first interrupt may be generated ahead of the set interrupt period.

The interrupt frequency (TBTCK) must be selected with the time base timer disabled (both frequency selection and enabling can be performed simultaneously).

Example : Sets the time base timer frequency to fc/$2^{16}$ [Hz] and enables an INTTBT interrupt.

```
LD      (TBTCR), 00001010B
SET     (EIRL). 6
```

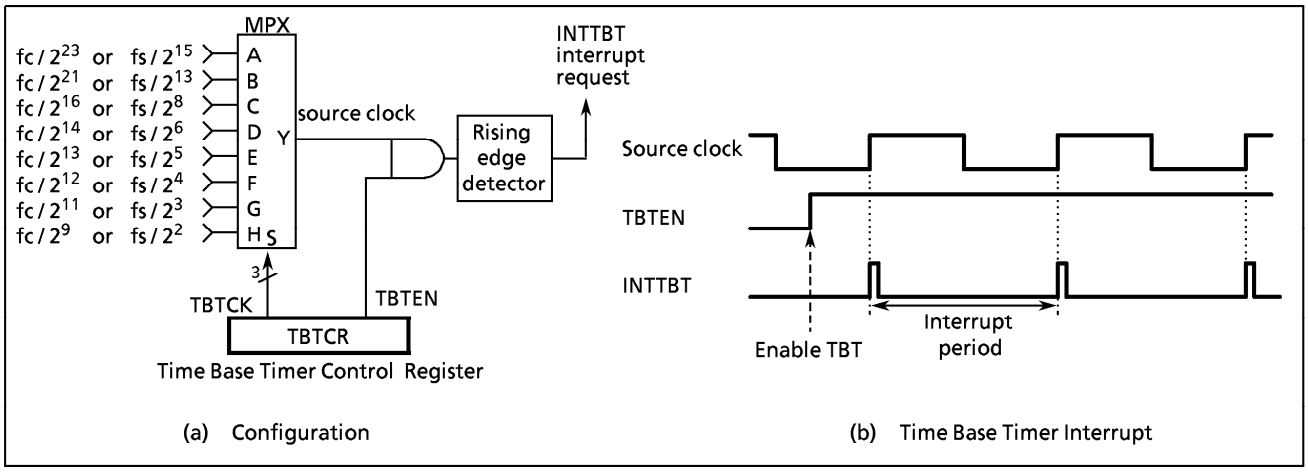
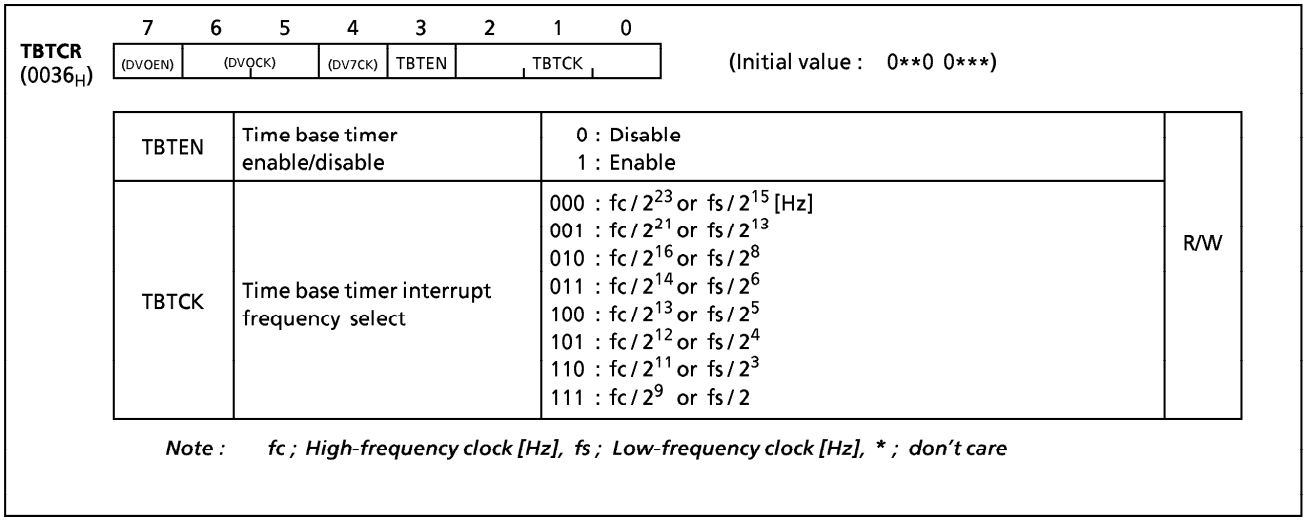



Figure 2-10. Time Base Timer




| | | | |
|---|---|---|---|
| TBTEN | Time base timer enable/disable | 0 : Disable<br>1 : Enable | R/W |
| TBTCK | Time base timer interrupt frequency select | 000 : fc / 2^23 or fs / 2^15 [Hz]<br>001 : fc / 2^21 or fs / 2^13<br>010 : fc / 2^16 or fs / 2^8<br>011 : fc / 2^14 or fs / 2^6<br>100 : fc / 2^13 or fs / 2^5<br>101 : fc / 2^12 or fs / 2^4<br>110 : fc / 2^11 or fs / 2^3<br>111 : fc / 2^9 or fs / 2 | |

*Note : fc ; High-frequency clock [Hz], fs ; Low-frequency clock [Hz], * ; don't care*

Figure 2-11. Time Base Timer and Divider Output Control Register

Table 2-1. Time Base Timer Interrupt Frequency

| TBTCK | NORMAL1/2, IDLE1/2 mode | | SLOW, SLEEP mode | Interrupt Frequency | |
|---|---|---|---|---|---|
| | DV7CK = 0 | DV7CK = 1 | | At fc = 8 MHz | At fs = 32.768 kHz |
| 000 | fc / 2^23 | fs / 2^15 | fs / 2^15 | 0.95 Hz | 1 Hz |
| 001 | fc / 2^21 | fs / 2^13 | fs / 2^13 | 3.81 | 4 |
| 010 | fc / 2^16 | fs / 2^8 | – | 122.07 | 128 |
| 011 | fc / 2^14 | fs / 2^6 | – | 488.28 | 512 |
| 100 | fc / 2^13 | fs / 2^5 | – | 976.56 | 1024 |
| 101 | fc / 2^12 | fs / 2^4 | – | 1953.12 | 2048 |
| 110 | fc / 2^11 | fs / 2^3 | – | 3906.25 | 4096 |
| 111 | fc / 2^9 | fs / 2 | – | 15625 | 16384 |

## 2.4 Divider Output ($\overline{DVO}$)

A 50 % duty pulse can be output using the divider output circuit, which is useful for piezo-electric buzzer drive. Divider output is from pin P13 ($\overline{DVO}$). The P13 output latch should be set to "1" and then the P13 should be configured as an output mode.

Divider output circuit is controlled by the control register (TBTCR) shown in Figure 2-12.

TBTCR ($0036_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DVOEN | DVOCK | | (DV7CK) | (TBTEN) | (TBTCK) | | |

(Initial value : 0**0 0***)

| | | | |
|---|---|---|---|
| DVOEN | Divider output enable/disable | 0 : Disable<br>1 : Enable | R/W |
| DVOCK | Divider output ($\overline{DVO}$) frequency selection | 00 : $fc/2^{13}$ or $fs/2^5$ [Hz]<br>01 : $fc/2^{12}$ or $fs/2^4$<br>10 : $fc/2^{11}$ or $fs/2^3$<br>11 : $fc/2^{10}$ or $fs/2^2$ | |

*Note : fc ; High-frequency clock [Hz], fs ; Low-frequency clock [Hz], * ; don't care*

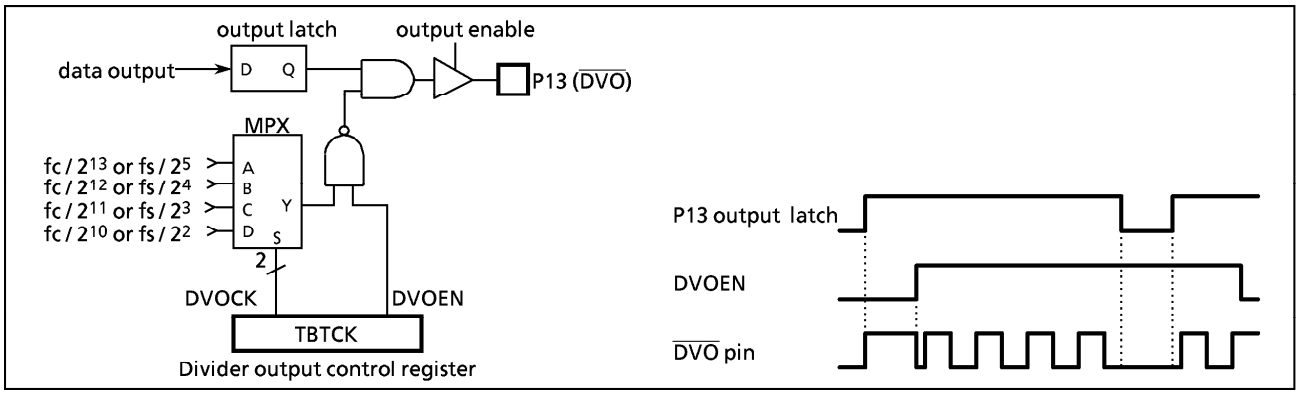
Figure 2-12. Divider Output Control Register

Example : 1 kHz pulse output (at fc = 8 MHz)

```
SET     (P1).3              ; P13 output latch ←1
LD      (P1CR), 00001000B   ; Configures P13 as an output mode
LD      (TBTCR), 10000000B  ; DVOEN←1, DVOCK←00
```

Table 2-2. Frequency of Divider Output

| DVOCK | Frequency of Divider Output | At fc = 8 MHz | At fs = 32.768 kHz |
|---|---|---|---|
| 00 | $fc/2^{13}$ or $fs/2^5$ | 0.976 [kHz] | 1.024 [kHz] |
| 01 | $fc/2^{12}$ $fs/2^4$ | 1.953 | 2.048 |
| 10 | $fc/2^{11}$ $fs/2^3$ | 3.906 | 4.096 |
| 11 | $fc/2^{10}$ $fs/2^2$ | 7.812 | 8.192 |




(a) Configuration (b) Timing Chart

Figure 2-13. Divider Output

## 2.5 16-bit Timer/Counter 1 (TC1)

### 2.5.1 Configuration

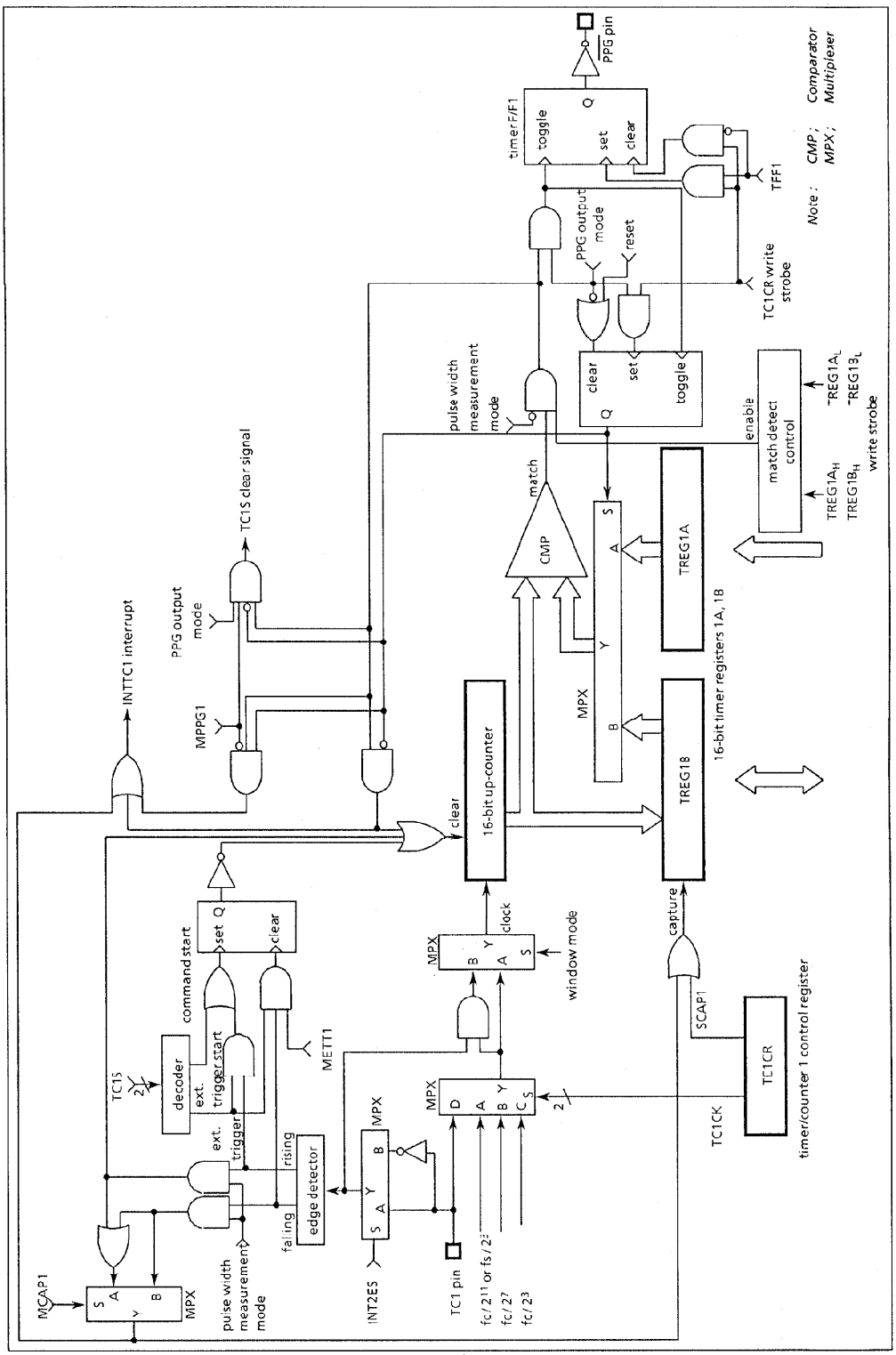
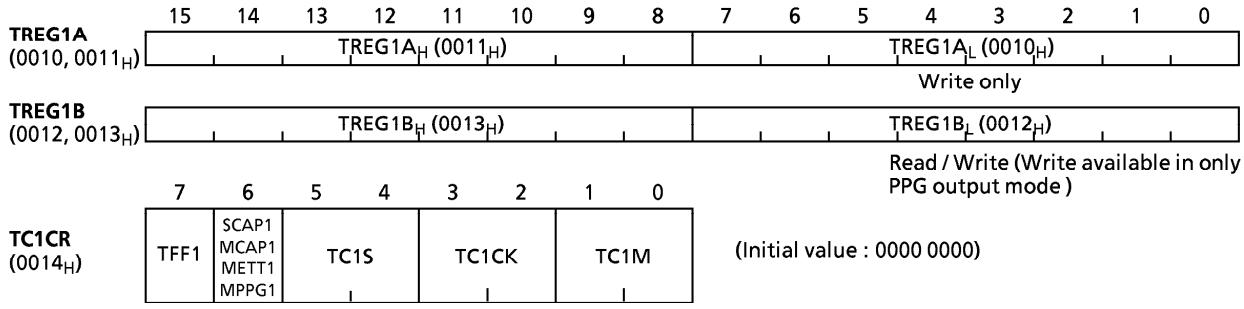



Figure 2-14. Timer/Counter 1

## 2.5.2 Control

The timer/counter 1 is controlled by a timer/counter 1 control register (TC1CR) and two 16-bit timer registers (TREG1A and TREG1B). Reset does not affect TREG1A and TREG1B.




| | | | |
|---|---|---|---|
| TC1M | TC1 mode select | 00 : timer / external trigger timer / event counter mode<br>01 : window mode<br>10 : pulse width measurement mode<br>11 : PPG output mode | Write only |
| TC1CK | TC1 source clock select | 00 : internal clock fc/$2^{11}$ or fs/$2^2$ [Hz]<br>01 : internal clock fc/$2^7$<br>10 : internal clock fc/$2^3$<br>11 : external clock (TC1 pin input) | |
| TC1S | TC1 start control | 00 : stop & counter clear<br>01 : command start<br>10 : reserved<br>11 : external trigger start | |
| SCAP1 | software capture control | 0 : – 1 : software capture trigger (Note 3) | |
| MCAP1 | pulse width measurement control | 1 : double edge capture 1 : single edge capture | |
| METT1 | external trigger timer control | 0 : trigger start 1 : trigger start & stop | |
| MPPG1 | PPG output control | 0 : continuous pulse 1 : single pulse | |
| TFF1 | timer F/F1 control for PPG output mode | 0 : clear 1 : set | |

*Note 1 : fc ; High-frequency clock [Hz], fs ; Low-frequency clock [Hz]*

*Note 2 : Writing to the low-byte of the timer registers (TREG1A$_L$, TREG1B$_L$), the comparison is inhibited until the high-byte (TREG1A$_H$, TREG1B$_H$) is written.*

*Note 3 : Set the mode, source clock, edge (INT2ES), PPG control and timer F/F control when TC1 stops (TC1S = 00).*

*Note 4 : Software capture can be used in only timer and event counter modes.*

*Note 5 : Values to be loaded to timer registers must satisfy the following condition.*
*TREG1A > TREG1B > 0 (PPG output mode) ; TREG1A > 0 (others)*

*Note 6 : Always write "0" to TFF1 except the PPG output mode.*

*Note 7 : TC1CR is a write-only register, which cannot be accessed by any read-modify-write instruction such as bit operate, etc.*

*Note 8 : TREG1B cannot be written after setting to PPG output mode.*

Figure 2-15. Timer Registers and TC1 Control Register

## 2.5.3 Function

Timer/counter 1 has six operating modes: timer, external trigger timer, event counter, window, pulse width measurement, programmable pulse generator output mode.

(1) **Timer** Mode

In this mode, counting up is performed using the internal clock. The contents of TREG1A are compared with the contents of up-counter. If a match is found, an INTTC1 interrupt is generated, and the counter is cleared to"0". Counting up resumes after the counteriscleared. The current contents of up-counter can be transfered to TREG1B by setting SCAP1 (bit 6 in TC1CR) to "1" (software capture function). SCAP1 is automatically cleared to "0" after capaturing.

Table 2-3. Timer/Counter 1 Source Clock (Internal Clock)

| Source clock | | | Resolution | | Maximum time setting | |
|---|---|---|---|---|---|---|
| NORMAL1/2, IDLE1/2 modes | | SLOW, SLEEP modes | | | | |
| DV7CK = 0 | DV7CK = 1 | | At fc = 8 MHz | At fs = 32.768 kHz | At fc = 8 MHz | At fs = 32.768 kHz |
| $fc/2^3$ [Hz] | $fc/2^3$ [Hz] | – | 1 μs | – | 65.5 ms | – |
| $fc/2^7$ | $fc/2^7$ | – | 16 μs | – | 1.0 s | – |
| $fc/2^{11}$ | $fs/2^3$ | $fs/2^3$ [Hz] | 256 μs | 244.14 μs | 16.8 s | 16.0 s |

Example 1 : Sets the timer mode with source clock $fs/2^3$[Hz] and generates an interrupt 1 s. later (at fs = 32.768 kHz).

```
LD      (TC1CR), 00000000B      ; Sets the TC1 mode and source clock
LDW     (TREG1A), 1000H         ; Sets the timer register (1 s ÷ 2^3 / fs = 1000H)
SET     (EIRL).EF4              ; enable INTTC1
EI
LD      (TC1CR), 00010000B      ; Starts TC1
```

Example 2 : Software capture

```
LD      (TC1CR), 01010000B      ; SCAP1←1 (Captures)
LD      WA, (TREG1B)            ; Reads captured value
```

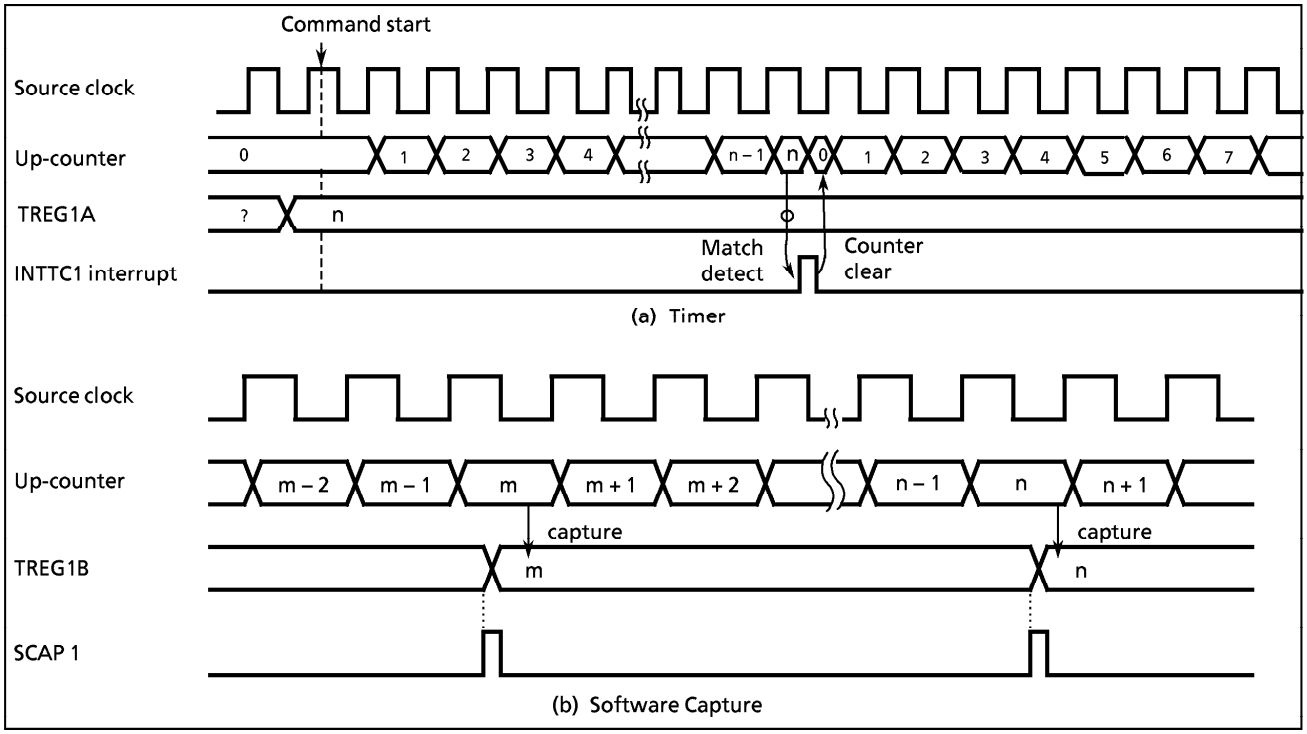



Figure 2-16. Timer Mode Timing Chart

(2) **External Trigger Timer** mode

In this mode, counting up is started by an external trigger. This trigger is the edge of the TC1 pin input. Either the rising or falling edge can be selected with INT2ES. Edge selection is the same as for the external interrupt input INT2 pin. Source clock is used an internal clock selected with TC1CK. The contents of TREG1A is compared with the contents of up-counter. If a match is found, an INTTC1 interrupt is generated, and the counter is cleared to"0" and halted. The counter is restarted by the selected edge of the TC1 pin input.

The TC1 pin input has the same noise rejection as the INT2 pin; therefore, pulses of 7/fc [s] or less are rejected as noise. A pulse width of 24/fc [s] or more is required for edge detection in NORMAL1/2 or IDLE1/2 mode. The noise rejection circuit is turned off in SLOW and SLEEP modes. But, a pulse width of 4/fs [s] or more is required.

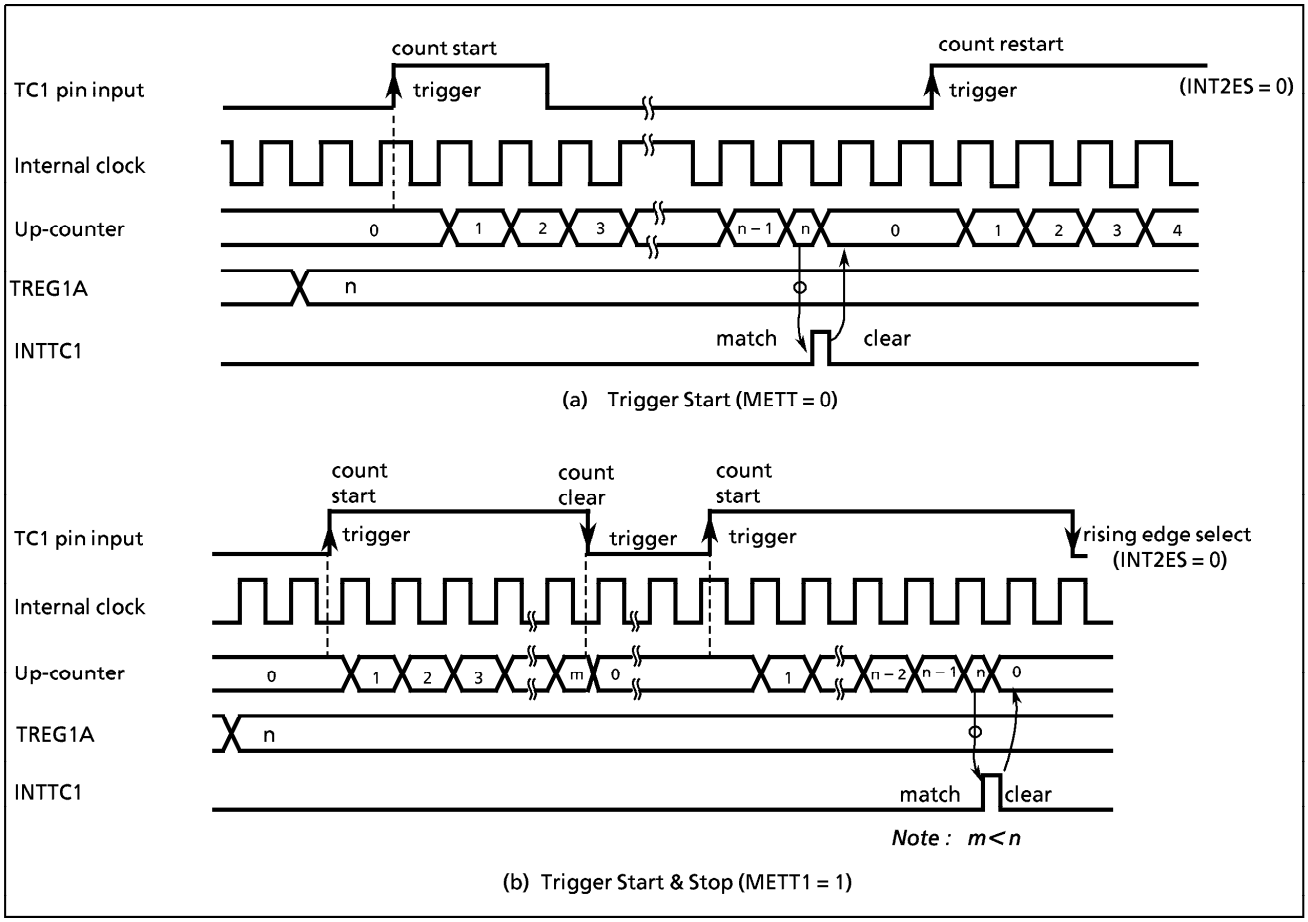



Figure 2-17. External Trigger Timer Mode Timing Chart

(3) **Event Counter** Mode

In this mode, events are counted on the edge of the TC1 pin input. Either the rising or falling edge can be selected with INT2ES in EINTCR. The contents of TREG1A are compared with the contents of up-counter. If a match is found, an INTTC1 interrupt is generated, and the counter is cleared. The maximum applied frequency is fc/$2^4$ [Hz] in NORMAL1/2 or IDLE1/2 mode and fs/$2^4$ [Hz] in SLOW or SLEEP mode.

Setting SCAP1 to "1" transferres the current contents of up-counter to TREG1B (software capture function). SCAP is automatically cleared after capturing.

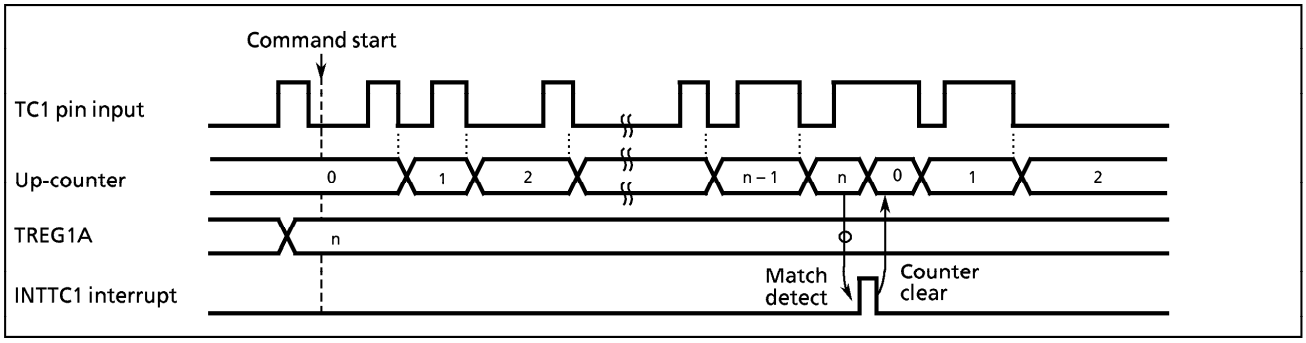



Figure 2-18. Event Counter Mode Timing Chart (INT2ES = 1)

(4) **Window** mode

Counting up is performed on the rising edge of the pulse that is the logical AND-ed product of the TC1 pin input (window pulse) and an internal clock. The contents of TREG1A are compared with the contents of up-counter. If a match is found, an INTTC1 interrupt is generated, and the counter is cleared. Positive or negative logic for the TC1 pin input can be selected with INT2ES. Setting SCAP1 to "1" transferes the current contents of up-counter to TREG1B. It is necessary that the maximum applied frequency (TC1 input) be such that the counter value can be analyzed by the program. That is, the frequency must be considerably slower than the selected internal clock.

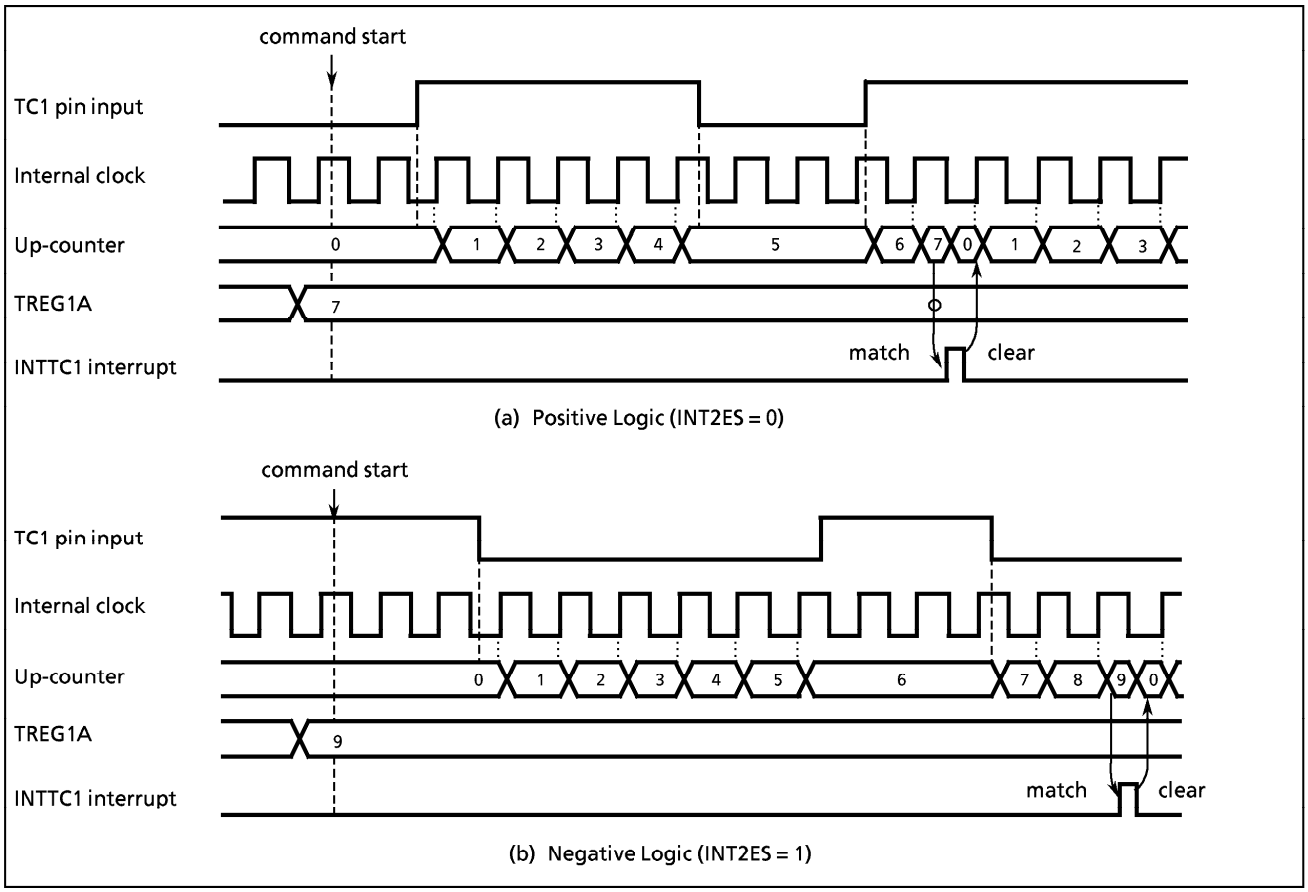



Figure 2-19. Window Mode Timing Chart

(5) **Pulse width measurement** mode

Counting is started by the external trigger (set to external trigger start by TC1S). The trigger can be selected either the rising or falling edge of the TC1 pin input. The source clock is used an internal clock. On the next falling (rising) edge, the counter contents are transferred to TREG1B and an INTTC1 interrupt is generated. The counter is cleared when the single edge capture mode is set. When double edge capture is set, the counter continues and, at the next rising (falling) edge, the counter contents are again transferred to TREG1B. If a falling (rising) edge capture value is required, it is necessary to read out TREG1B contents until a rising (falling) edge is detected. Falling or rising edge is selected with INT2ES, and single edge or double edge is selected with MCAP1 ( bit 6 in TC1CR).

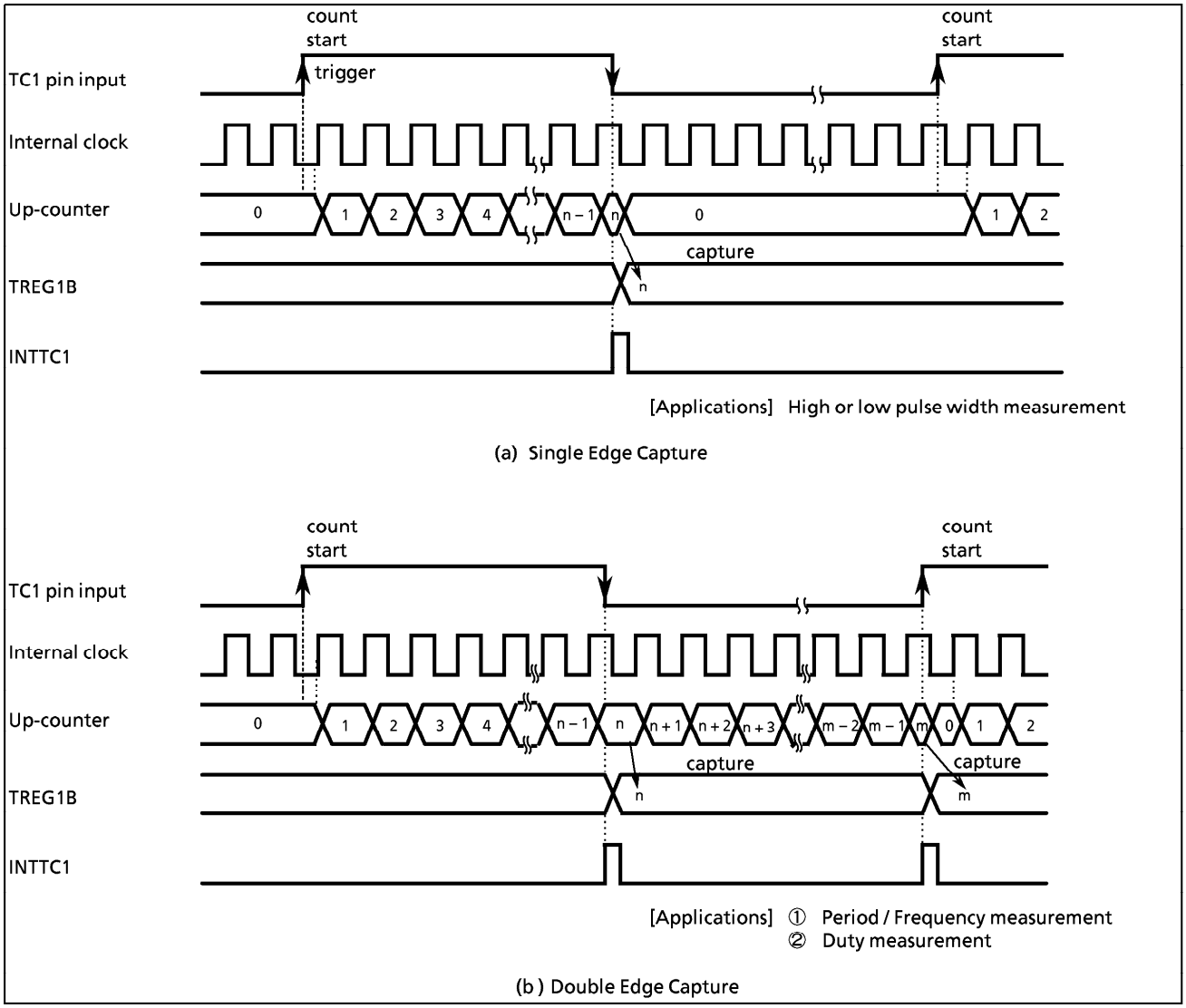



Figure 2-20. Pulse Width Measurement Mode Timing Chart

Example : Duty measurement (Resolution $fc/2^7$ [Hz])

```
            CLR     (INTTC1SW). 0           ;  INTTC1 service switch initial setting
            LD      (EINTCR), 00000000B     ;  Sets the rise edge at the INT2 edge
            LD      (TC1CR), 00000110B      ;  Sets the TC1 mode and source clock
            SET     (EIRL). 4               ;  Enables INTTC1
            EI
            LD      (TC1CR), 00110110B      ;  Starts TC1 with an external trigger
             :
PINTTC1 :   CPL     (INTTC1SW). 0           ;  Complements INTTC1 service switch
            JRS     F, SINTTC1
            LD      (HPULSE), (TREG1BL)     ;  Reads TREG1B
            LD      (HPULSE + 1), (TREG1BH)
            RETI
SINTTC1 :   LD      (WIDTH), (TREG1BL)      ;  Reads TREG1B (Period)
            LD      (WIDTH + 1), (TREG1BH)
             :
            RETI
             :
VINTTC1 :   DW      PINTTC1
```

(6) **Programmable Pulse Generate** (PPG) **output** mode

Counting is started by an edge of the TC1 pin input (either the rising or falling edge can be selected) or by a command. The source clock is used an internal clock. First, the contents of TREG1B are compared with the contents of the up-counter. If a match is found, timer F/F1 output is toggled. Next, timer F/F1 is again toggled and the counter is cleared by matching with TREG1A. An INTTC1 interrupt is generated at this time. Timer F/F output is connected to the P14 ($\overline{PPG}$) pin. In the case of $\overline{PPG}$ output, set the P14 output latch to "1" and configure as an output with $P1CR_4$. Timer F/F1 is cleared to "0" during reset. The timer F/F1 value can also be set by program and either a positive or negative logic pulse output is available. Also, writing to the TREG1B is not possible unless the timer / counter 1 is set to the PPG output mode with TC1M.

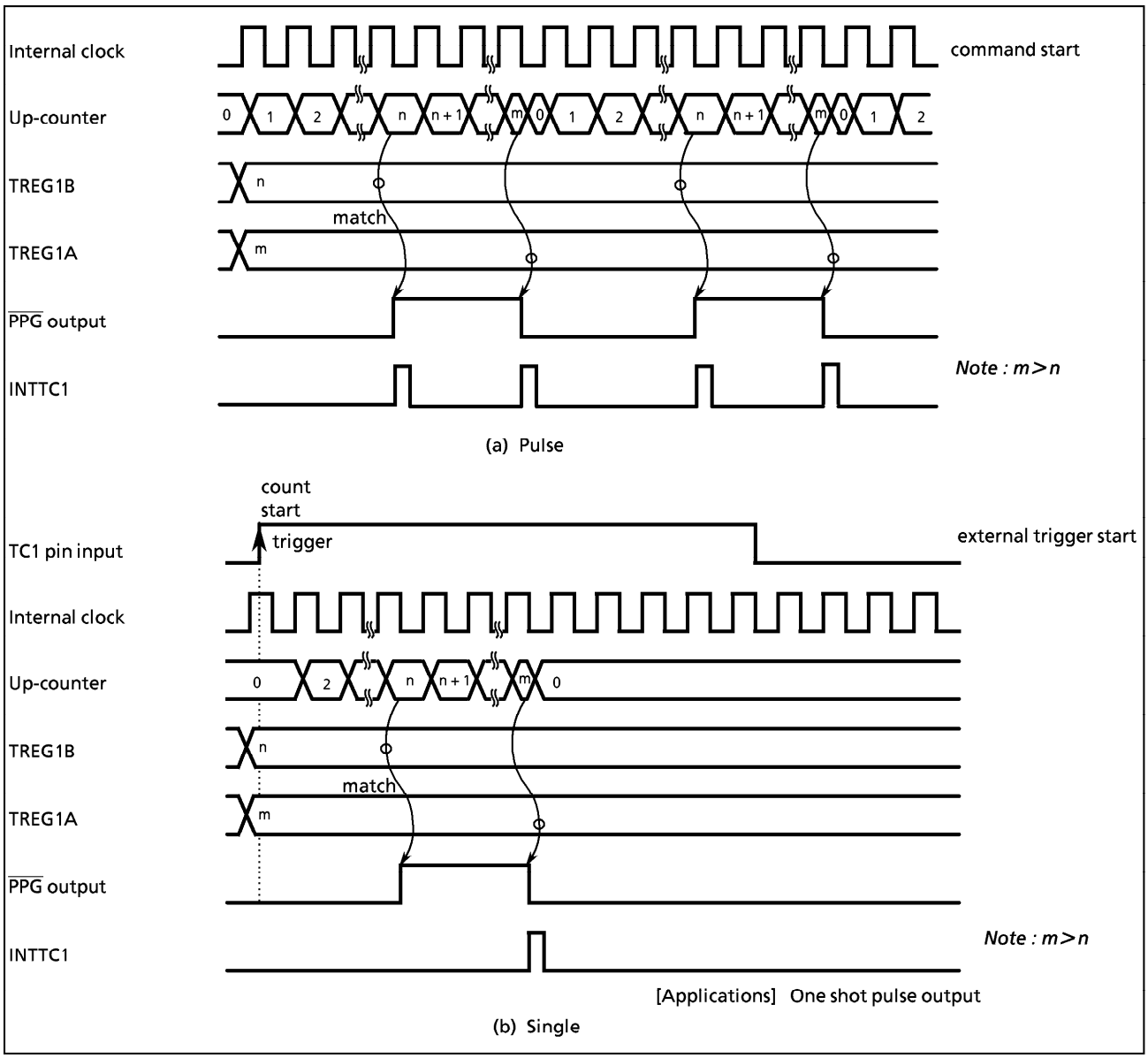



Figure 2-21. PPG Output Mode Timing Chart

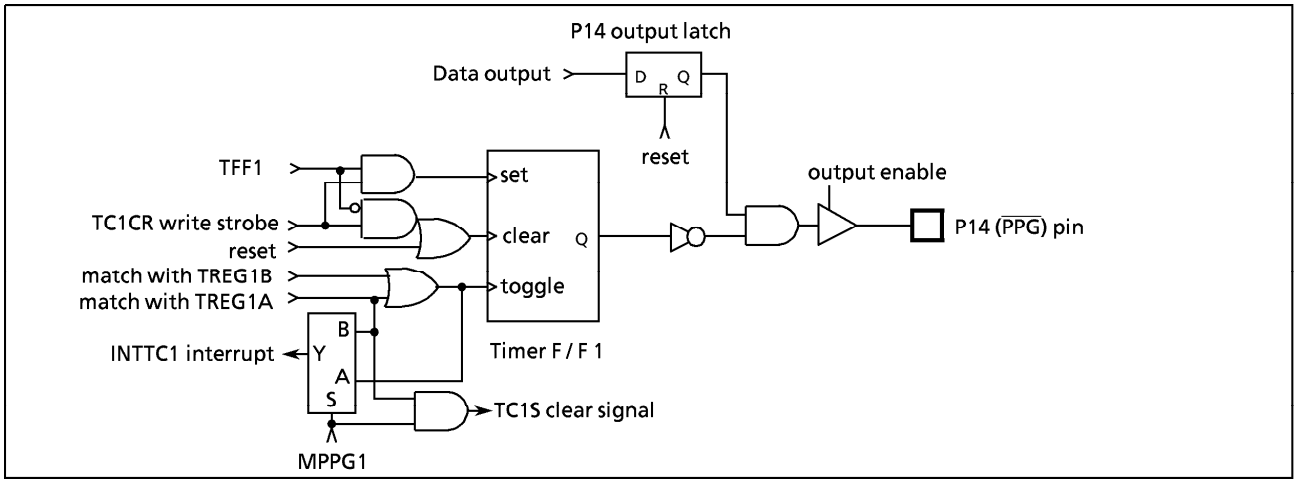



Figure 2-22. PPG Output

## 2.6 16-bit Timer/Counter 2 (TC2)

### 2.6.1 Configuration

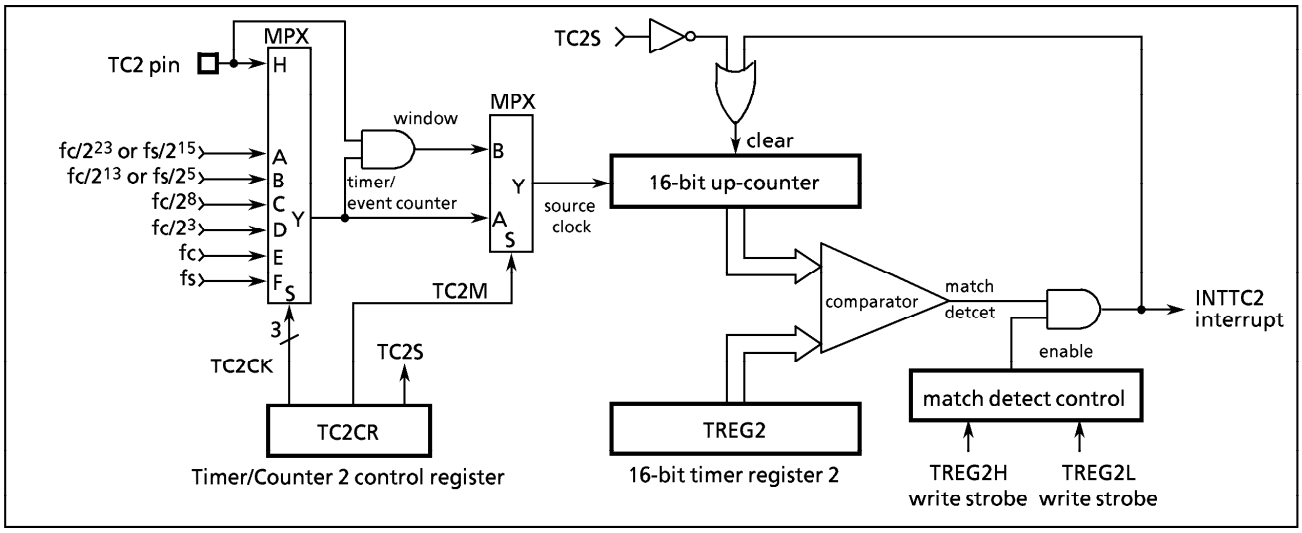



Figure 2-23. Timer/Counter 2 (TC2)

## 2.6.2 Control

The timer/counter 2 is controlled by a timer/counter 2 control register (TC2CR) and a 16-bit timer register 2 (TREG2). Reset does not affect TREG2.

**TREG2** (0016, 0017H)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TREG2H (0017H) | | | | | | | | TREG2L (0016H) | | | | | | | |

write only

**TC2CR** (0015H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | TC2S | TC2CK | | | | TC2M |

(Initial value : **00 00*0)

| | | | |
|---|---|---|---|
| TC2M | Timer/counter 2 operating mode select | 0 : Timer/Event counter mode<br>1 : Window mode | write only |
| TC2CK | Timer/counter 2 source clock select | 000 : Internal clock fc / $2^{23}$ or fs / $2^{15}$ [Hz]<br>001 : 〃 fc / $2^{13}$ or fs / $2^{5}$<br>010 : 〃 fc / $2^{8}$<br>011 : 〃 fc / $2^{3}$<br>100 : 〃 fc (Note 5)<br>101 : 〃 fs<br>110 : Reserved<br>111 : External clock (TC2 pin input) | |
| TC2S | Timer/counter 2 start control | 0 : Stop and counter clear<br>1 : Start | |

*Note 1 : fc; High-frequency clock [Hz], fs; Low-frequency clock [Hz], *; don't care*

*Note 2 : When writing to the low-byte of timer register 2 ($TREG2_L$), the comparison is inhibited until the high-byte ($TREG2_H$) is written.*
*After writing to the high-byte, any match during 1 machine cycle (instruction execution cycle) is ignored.*

*Note 3 : Set the mode and source clock when timer/counter stops (TC2S = 0).*

*Note 4 : Values to be loaded to the timer register must satisfy the following condition.*
*$TREG2 > 0$ ($TREG2_{15\ to\ 11} > 0$ when warm-up).*

*Note 5 : "fc" can be selected as the source clock only in the timer mode during the SLOW mode.*

*Note 6: TC2CR and TREG2 are write-only registers and must not be used with any of the read-modify-write instructions.*

Figure 2-24. Timer Register 2 and TC2 Control Register

## 2.6.3 Function

The timer/counter 2 has three operating modes: timer, event counter and window modes. Also timer/counter 2 is used for warm-up when switching from SLOW mode to NORMAL2 mode.

(1) **Timer** Mode

In this mode, the internal clock is used for counting up. The contents of TREG2 are compared with the contents of up-counter. If a match is found, a timer/ counter 2 interrupt (INTTC2) is generated, and the counter is cleared. Counting up is resumed after the counter is cleared.

Also, when fc is selected as the source clock during SLOW mode, the lower 11 bits of TREG2 are ignored and an INTTC2 interrupt is generated by matching the upper 5 bits. Thus, in this case, only the $TREG2_H$ setting is necessary.

Table 2-4. Source Clock (Internal Clock) for Timer/Counter 2

| Source clock | | | | Resolution | | Maximum time setting | |
|---|---|---|---|---|---|---|---|
| NORMAL1/2, IDLE1/2 mode | | SLOW mode | SLEEP mode | | | | |
| DV7CK = 0 | DV7CK = 1 | | | At fc = 8 MHz | At fs = 32.768 kHz | At fc = 8 MHz | At fs = 32.768 kHz |
| $fc/2^{23}$ [Hz] | $fs/2^{15}$ [Hz] | $fs/2^{15}$ [Hz] | $fs/2^{15}$ [Hz] | 1.05 s | 1 s | 19.1 h | 18.2 h |
| $fc/2^{13}$ | $fs/2^{5}$ | $fs/2^{5}$ | $fs/2^{5}$ | 1.02 ms | 1 ms | 1.1 min | 1 min |
| $fc/2^{8}$ | $fc/2^{8}$ | – | – | 32 μs | – | 2.1 s | – |
| $fc/2^{3}$ | $fc/2^{3}$ | – | – | 1 μs | – | 65.5 ms | – |
| – | – | fc (Note) | – | 125 ns | – | 7.9 ms | – |
| fs | fs | – | – | – | 30.5 μs | – | 2 s |

*Note : "fc" can be used only in the timer mode.*

Example : Sets the timer mode with source clock $fc/2^3$ [Hz] and generates an interrupt every 25 ms (at fc = 8 MHz).

```
LD      (TC2CR), 00001100B      ; Sets the TC2 mode and source clock
LDW     (TREG2), 61A8H          ; Sets TREG2 (25ms ÷ 2^3/fc = 61A8H)
SET     (EIRH).EF14             ; Enable INTTC2
EI
LD      (TC2CR), 00101100B      ; Starts TC2
```

(2) **Event Counter** Mode

In this mode, events are counted on the rising edge of the TC2 pin input. The contents of TREG2 are compared with the contents of the up-counter. If a match is found, an INTTC2 interrupt is generated, and the counter is cleared. The maximum frequency applied to the TC2 pin is $fc/2^4$ [Hz] in NORMAL1/2 or IDLE1/2 mode, and $fs/2^4$ [Hz] in SLOW or SLEEP mode.

Example : Sets the event counter mode and generates an INTT2 interrupt 640 counts later.

```
LD      (TC2CR), 00011100B      ; Sets the TC2 mode
LDW     (TREG2), 640            ; Sets TREG2
SET     (EIRH).EF14             ; Enable INTTC2
EI
LD      (TC2CR), 00111100B      ; Starts TC2
```

(3) **Window** Mode

In this mode, counting up is performed on the rising edge of the pulse that is the logical AND-ed product of the TC2 pin input (window pulse) and an internal clock. The internal clock is selected with TC2CK. The contents of TREG2 are compared with the contents of up-counter. If a match is found, an INTTC2 interrupt is generated, and the up-counter is cleared to "0". It is necessary that the maximum applied frequency (TC2 input) be such that the counter value can be analyzed by the program. That is, the frequency must be considerably slower than the selected internal clock.

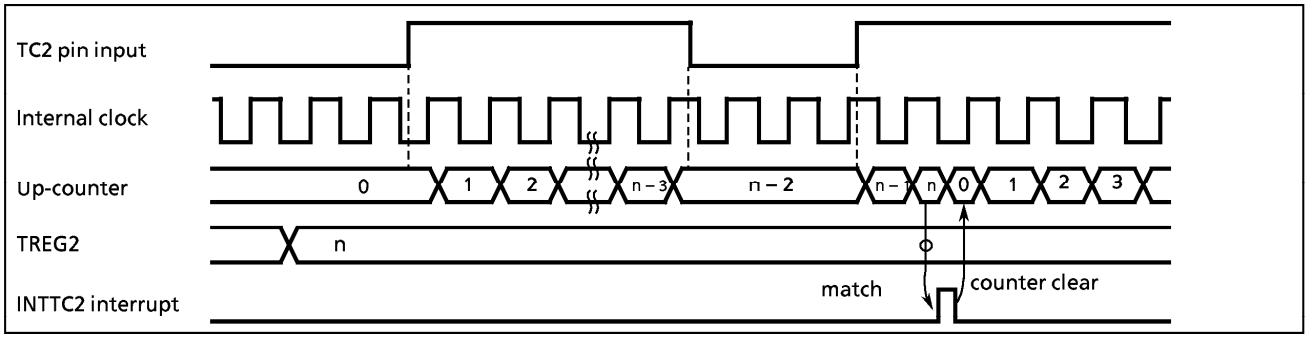



Figure 2-25. Window Mode Timing Chart

## 2.7 8-Bit Timer/Counter 3 (TC3)

### 2.7.1 Configuration

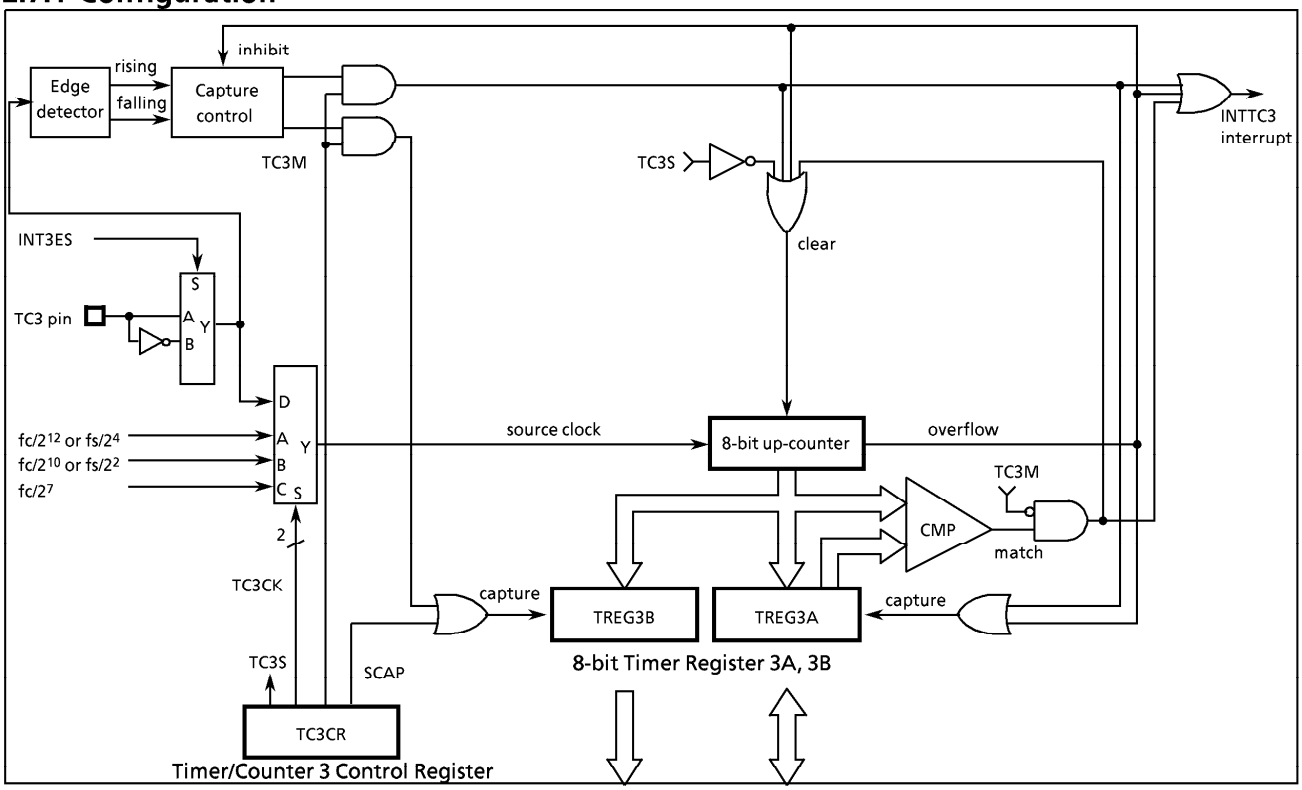
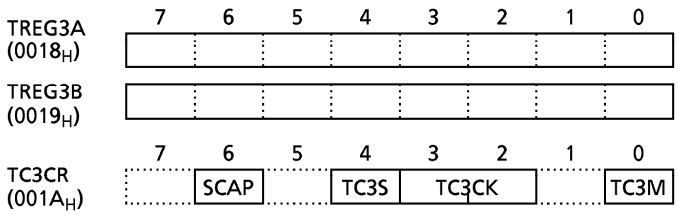



Figure 2-26. Timer/Counter 3

### 2.7.2 Control




| | | | |
|---|---|---|---|
| TC3M | Timer/counter 3 operation mode set | 0 : Timer/event counter<br>1 : Capture | Write only |
| TC3CK | Timer/counter 3 source clock select | 00 : Internal clock fc / $2^{12}$ or fs / $2^4$ [Hz]<br>01 : Internal clock fc / $2^{10}$ or fs / $2^2$<br>10 : Internal clock fc / $2^7$<br>11 : External clock (TC3 pin input) | |
| TC3S | Timer/counter 3 start select | 0 : Stop & clear<br>1 : Start | |
| SCAP | Software capture control | 0 : –<br>1 : Software capture | |

*Note 1 : fc ; High-frequency clock [Hz] fs ; Low-frequency clock [Hz] * ; don't care*
*Note 2 : Set the mode, the source clock and the edge selection (INT3ES) when the TC3 stops (TC3S = 0).*
*Note 3 : Values to be loaded into timer register 3A must satisfy the following condition.*
*TREG3A > 0 (in the timer/event counter mode)*
*Note 4 : TC3CR is a write-only register and must not be used with any of read-modify-write instructions.*

Figure 2-27. Timer Register 3A/3B and TC3 Control Register

The timer/counter 3 is controlled by a timer/counter 3 control register (TC3CR) and two 8-bit timer registers (TREG3A and TREG3B). Reset does not affect these timer registers.

### 2.7.3 Function

The timer/counter 3 has three operating modes : timer, event counter, and capture mode.

(1) **Timer** Mode

In this mode, the internal clock is used for counting up. The contents of TREG3A are compared with the contents of up-counter. If a match is found, a timer/counter 3 interrupt (INTTC3) is generated, and the up-counter is cleared. Counting up resumes after the up-counter is cleared. The current contents of up-counter are loaded into TREG3B by setting SCAP (bit 6 in TC3CR) to "1". SCAP is automatically cleared after capturing.

Table 2-5. Source Clock (Internal Clock) for Timer Counter 3

| Source clock | | Resolution | | Maximum setting time | |
|---|---|---|---|---|---|
| NORMAL1/2, IDLE1/2 mode | SLOW, SLEEP mode | fc = 8 MHz | fs = 32.768 kHz | fc = 8 MHz | fs = 32.768 kHz |
| $fc/2^{12}$ or $fs/2^4$ [Hz] | $fs/2^4$ [Hz] | 512 μs | 488.28 μs | 131.1 ms | 125 ms |
| $fc/2^{10}$ or $fs/2^2$ | – | 128 μs | 122.07 μs | 32.8 ms | 31.25 ms |
| $fc/2^7$ | – | 16 μs | – | 4.1 ms | – |

(2) **Event Counter** Mode

In this mode, the TC3 pin input pulses are used for counting up. Either the rising or falling edge can be selected with INT3ES (bit 3 in EINTCR). The contents of TREG3A are compared with the contents of the up-counter. If a match is found, an INTTC3 interrupt is generated and the counter is cleared. The maximum applied frequency is $fc/2^4$ [Hz] in the NORMAL1/2 or IDLE1/2 mode, and $fs/2^4$ [Hz] in SLOW or SLEEP mode. Two or more machine cycles are required for both the "H" and "L" levels of the pulse width.

The current contents of up-counter are loaded into TREG3B by setting SCAP (bit 6 in TC3CR) to "1". SCAP is automatically cleared after capturing.

Example : Generates an interrupt every 0.5 s, inputing 50Hz pulses to the TC3 pin.

```
LD (TC3CR), 00001100B  ;  Sets TC3 mode and source clock
LD (TREG3A), 19H       ;  0.5 s ÷ 1 / 50 = 25 = 19H
LD (TC3CR), 00011100B  ;  Start TC3
```

(3) **Capture** Mode

The pulse width, period and duty of the TC3 pin input are measured in this mode, which can be used in decoding the remote control signals, etc. The counter is free running by the internal clock. On the rising (falling) edge of the TC3 pin input, the current contents of counter is loaded into TREG3A, then the up-counter is cleared and an INTTC3 interrupt is generated. On the falling (rising) edge of the TC3 pin input, the current contents of the counter is loaded into the TREG3B. In this case, counting continues. At the next rising (falling) edge of the TC3 pin input, the current contents of counter are loaded into TREG3A, then the counter is cleared again and an interrupt is generated. If the counter overflows before the edge is detected, $FF_H$ is set to the TREG3A and an overflow interrupt (INTTC3) is generated. During interrupt processing, it can be determined whether or not there is an overflow by checking whether or not the TREG3A value is $FF_H$. Also, after an interrupt (capture to TREG3A, or overflow detection) is generated, capture and overflow detection are halted until TREG3A has been read out; however, the counter continues.

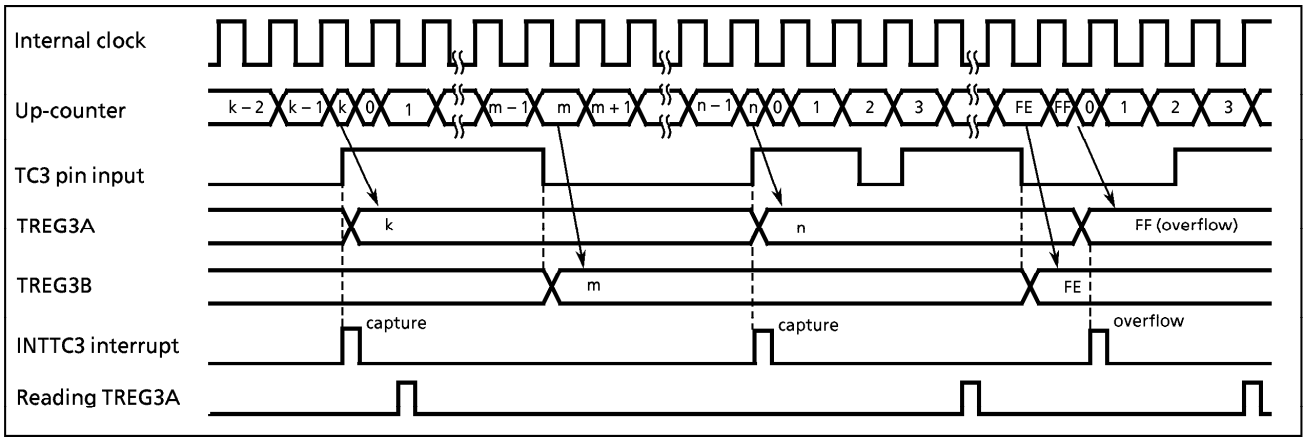



Figure 2-28. Timing Chart for Capture Mode (INT3ES = 0)

## 2.8 8-bit Timer/Counter (TC4)

### 2.8.1 Configuration

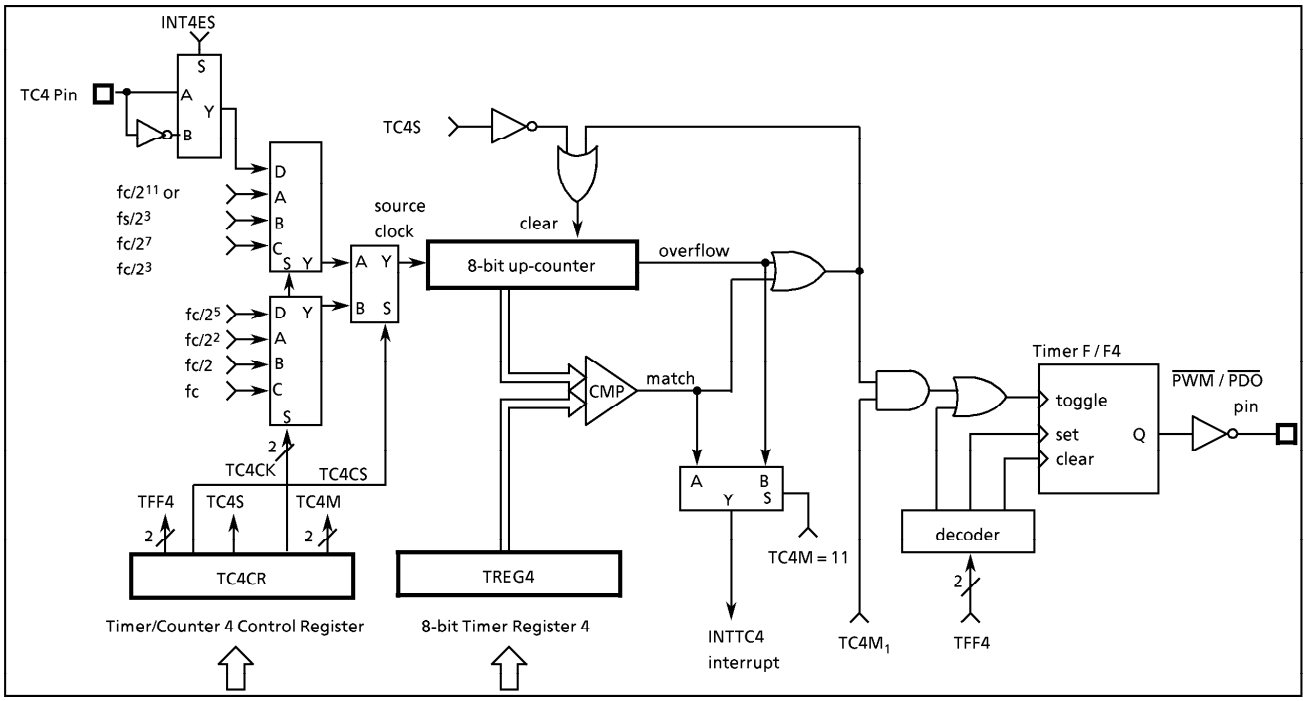



Figure 2-29. Timer/Counter 4

## 2.8.2 Control

The timer/counter 4 is controlled by a timer/counter 4 control register (TC4CR) and an 8-bit timer register 4 (TREG4). Reset does not affect TREG4.

TREG4 ($001B_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

Write only

TC4CR ($001C_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TFF4 | | TC4CS | TC4S | TC4CK | | TC4M | |

(Initial value : 0000 0000)

| | | | | |
|---|---|---|---|---|
| TC4M | TC4 operating mode select | 00 : Timer/event counter mode<br>01 : Reserved<br>10 : Programmable divider output (PDO) mode<br>11 : Pulse width modulation (PWM) output mode | | write only |
| TC4CK | TC4 source clock select | mode A (TC4CS = 0) | mode B (TC4CS = 1) | |
| | | 00 : fc / $2^{11}$ or fs / $2^3$<br>01 : fc / $2^7$<br>10 : fc / $2^3$<br>11 : External clock (TC4 pin input) | 00 : fc / $2^5$<br>01 : fc / $2^2$<br>10 : fc / 2<br>11 : fc | |
| TC4S | TC4 start control | 0 : Stop & clear<br>1 : Start | | |
| TC4CS | TC4 mode select | 0 : mode A<br>1 : mode B | | |
| TFF4 | Timer F/F 4 control | 00 : Clear<br>01 : Toggle<br>10 : Set<br>11 : – (Note 3) | | |

*Note 1 : fc; High-frequency clock [Hz], fs; Low-frequency clock [Hz], *; don't care*

*Note 2 : Set the operating mode, the source clock selection, the timer F/F 4 control and the edge selection (INT4ES) when the TC4 stops (TC4S = 0).*

*Note 3 : TFF4 must be set to "11" in the timer, event counter modes and B mode.*

*Note 4 : Values to be loaded to the timer register must satisfy the following condition.*

*(a) When in PWM output mode, 5 < TREG4 < 251*

*(b) When any other mode than PWM output mode, 0 < TREG4*

*Note 5 : Source clock fc/$2^2$, fc/2, and fc cannot be used except in PWM output mode.*

*Note 6 : TC4CR and TREG4 are write-only registers and must not be used with any of read-modify-write instructions.*

Figure 2-30. Timer Register 4 and TC4 Control Register

## 2.8.3 Function

The timer/counter 4 has four operating modes : timer, event counter, programmable divider output, and PWM output mode.

(1) **Timer** Mode

In this mode, the internal clock is used for counting up. The contents of TREG4 are compared with the contents of up-counter. If a match is found, a timer/counter 4 interrupt (INTTC4) is generated and the up-counter is cleared to "0". Counting up resumes after the up- counter is cleared.

Table 2-6. Source Clock (Internal Clock) for Timer/Counter 4

| Source clock | | | Resolution | | Maximum setting time | |
|---|---|---|---|---|---|---|
| Mode | NORMAL1/2, IDLE1/2 mode | SLOW, SLEEP mode | At fc = 8 MHz | At fs = 32.768 kHz | At fc = 8 MHz | At fs = 32.768 kHz |
| A | fc / $2^{11}$ or fs / $2^3$ [Hz] | fs / $2^3$ [Hz] | 256 μs | 244.14 μs | 65.3 ms | 62.2 ms |
| | fc / $2^7$ | – | 16 μs | – | 4.1 ms | – |
| | fc / $2^3$ | – | 1 μs | – | 255 μs | – |
| B | fc / $2^5$ | – | 4 μs | – | 1020 μs | |

(2) **Event Counter** Mode

In this mode, the TC4 pin input (external clock) pulse is used for counting up. Either the rising or falling edge can be selected with TC4ES (bit 4 in EINTCR). The contents of the TREG4 are compared with the contents of the up-counter. If a match is found, an INTTC4 interrupt is generated and the counter is cleared. The maximum applied frequency is $fc/2^4$ [Hz] in NORMAL1/2 or IDLE1/2 mode, and $fs/2^4$ [Hz] in SLOW or SLEEP mode. Two or more machine cycles are required for both the high and low levels of the pulse width.

(3) **Programmable Divider Output** (PDO) Mode

The internal clock is used for counting up. The contents of TREG4 are compared with the contents of the up-counter. Timer F/F 4 output is toggled and the counter is cleared each time a match is found. Timer F/F 4 output is inverted and output to the $\overline{\text{PDO}}$ (P55) pin. This mode can be used for 50% duty pulse output. Timer F/F 4 can be initialized by program, and it is initialized to "0" during reset. An INTTC4 interrupt is generated each time the $\overline{\text{PDO}}$ output is toggled.

Example : Output a 1024 Hz pulse (at fc = 4.194304 MHz)

```
LD      (TC4CR), 00000010B      ; Initializes the TC4 mode, source clock and timer F/F 4.
LD      (TREG4), 10H            ; 1/2048 ÷ 2^7/fc = 10H
LD      (TC4CR), 00010010B      ; Starts TC4
```

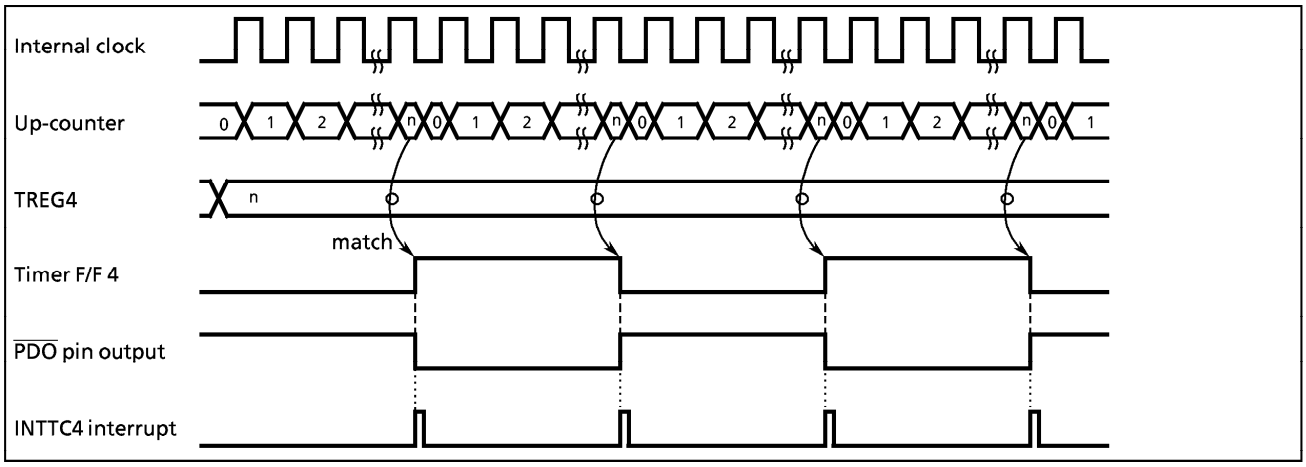



Figure 2-31. Timing Chart for PDO Mode

(4) **Pulse Width Modulation** (PWM) **Output** Mode

PWM output with a resolution of 8 bits is possible. The internal clock is used for counting up. The contents of TREG4 are compared with the contents of up-counter. If a match is found, the timer F/F 4 output is toggled. The counter continues counting. And, when an overflow occurs, the timer is again toggled and the counter is cleared. Timer F/F 4 output is inverted and output to the $\overline{\text{PWM}}$ (P55) pin. An INTTC4 interrupt is generated when an overflow occurs.

TREG4 is configured a 2-stage shift register and, during output, will not switch until one output cycle is completed even if TREG4 is overwritten; therefore, output can be altered continuously. Also, the first time, TREG4 is shifted by setting TC4S (bit 4 in TC4CR) to "1" after data are loaded to TREG4.

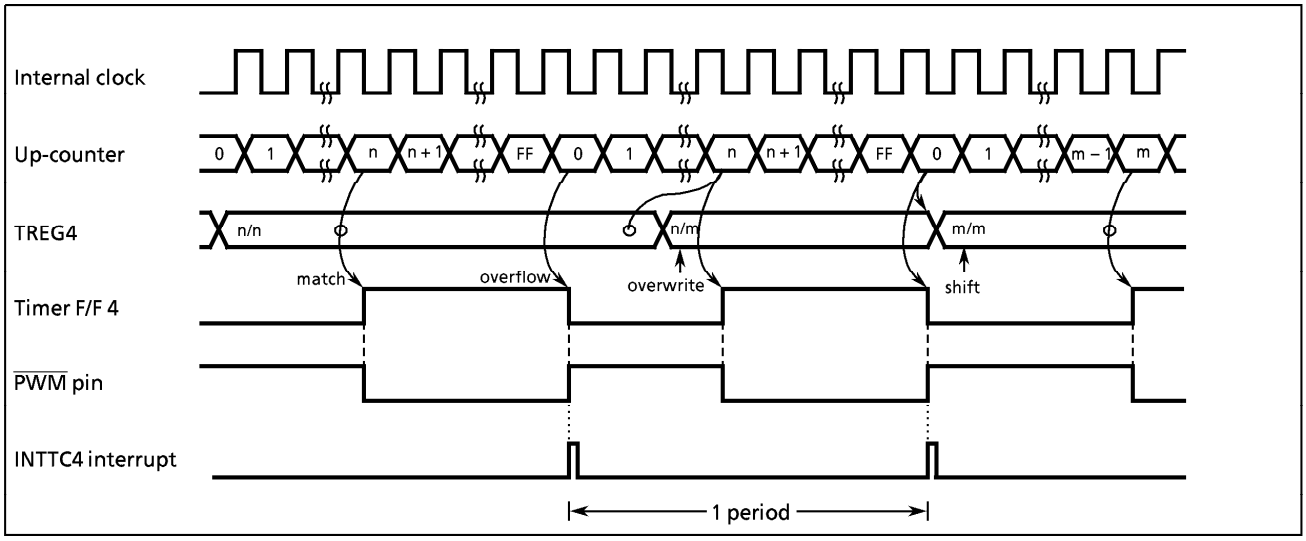



Figure 2-32. Timing Chart for PWM Mode

Table 2-7. PWM Output Mode

| Mode | Source clock: NORMAL1/2, IDLE1/2 mode DV7CK = 0 | Source clock: NORMAL1/2, IDLE1/2 mode DV7CK = 1 | Source clock: SLOW, SLEEP mode | Resolution: At fc = 8 MHz | Resolution: At fs = 32.768 kHz | Repeat cycle: At fc = 8 MHz | Repeat cycle: At fs = 32.768 kHz |
|---|---|---|---|---|---|---|---|
| A | $fc / 2^{11}$ [Hz] | $fs / 2^3$ [Hz] | $fs / 2^3$ [Hz] | 256 μs | 244.14 μs | 65.3 ms | 62.2 ms |
| | $fc / 2^7$ | – | – | 16 μs | – | 4.1 ms | – |
| | $fc / 2^3$ | – | – | 1 μs | – | 255 μs | – |
| B | $fc / 2^5$ [Hz] | – | – | 4 μs | – | 1024 μs | – |
| | $fc / 2^2$ | – | – | 500 ns | – | 128 μs | – |
| | $fc / 2$ | – | – | 250 ns | – | 64 μs | – |
| | fc | – | – | 125 ns | – | 32 μs | – |

## 2.9 Serial Interface (SIO1, SIO2)

The 87C841/CC41/CH41/CK41/CM41 each have two clocked-synchronous 8-bit serial interfaces (SIO1 and SIO2). Each serial interface has an 8-byte transmit and receive data buffer that can automatically and continuously transfer up to 64 bits of data.

The serial interfaces are connected to external devices via pins P44 (SO1), P43 (SI1), P42 ($\overline{SCK1}$) for SIO1 and P47 (SO2), P46 (SI2), P45 ($\overline{SCK2}$) for SIO2. The serial interface pins are also used as port P4. When used as serial interface pins, the output latches of these pins should be set to "1". In the transmit mode, pins P43 and P46 can be used as normal I/O ports, and in the receive mode, the pins P44 and P47 can be used as normal I/O ports.

### 2.9.1 Configuration

The SIO1 and SIO2 have the same configuration, except for the addresses/bit positions of the control/status registers and buffer registers.

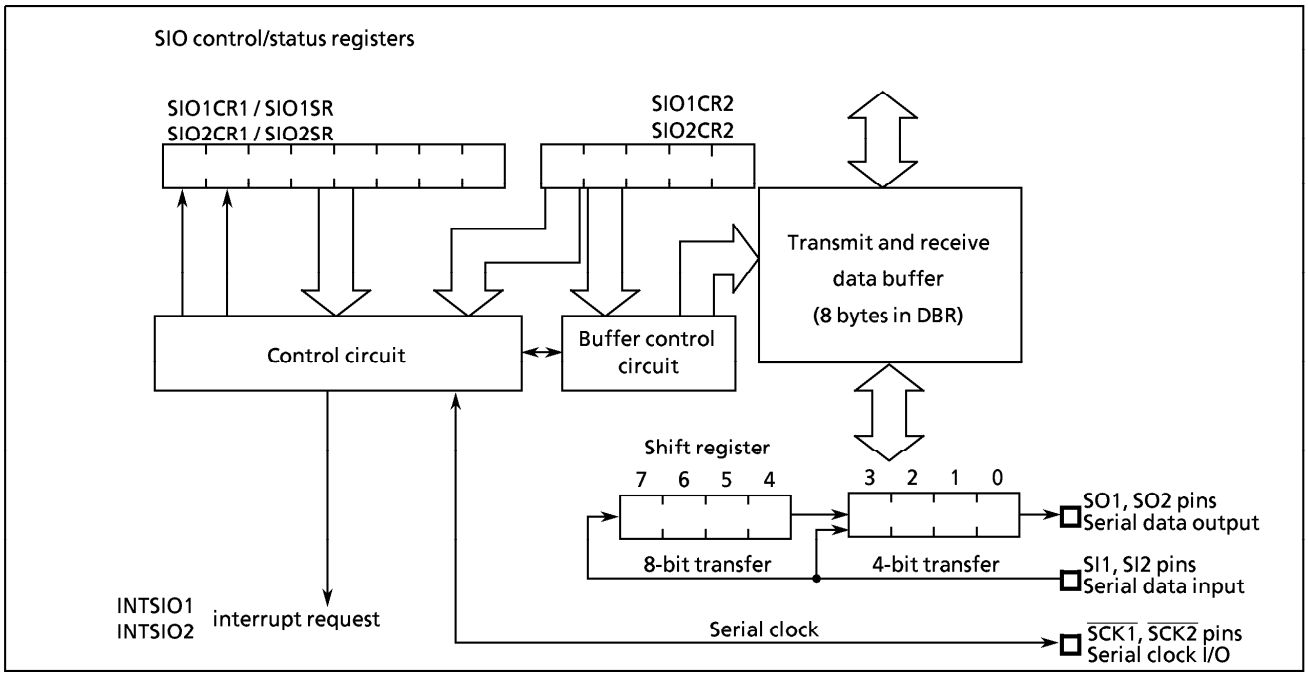



Figure 2-33. Serial Interfaces

### 2.9.2 Control

The serial interfaces are controlled by SIO control registers (SIO1CR1/SIO1CR2 or SIO2CR1/SIO2CR2). The serial interface status can be determined by reading SIO status registers (SIO1SR or SIO2SR).

The transmit and receive data buffer is controlled by the BUF (bits 2-0 in SIO1CR2/SIO2CR2). The data buffer is assigned to addresses $0FF0_H$ to $0FF7_H$ for SIO1 or $0FF8_H$ - $0FFF_H$ for SIO2 in the DBR area, and can continuously transfer up to 8 words (bytes or nibbles) at one time. When the specified number of words has been transferred, a buffer empty (in the transmit mode) or a buffer full (in the receive mode or transmit/receive mode) interrupt (INTSIO1 or INTSIO2) is generated.

When the internal clock is used as the serial clock in the 8-bit receive mode and the 8-bit transmit/receive mode, a fixed interval wait can be applied to the serial clock for each word transferred. Four different wait times can be selected with WAIT (bits 4 and 3 in SIO1CR2/SIO2CR2).

## SIO1, SIO2 Control Registers 1

**SIO1CR1** ($0020_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SIOS | SIO INH | SIOM | | | SCK | | |

(Initial value : 0000 0000)

**SIO2CR1** ($0022_H$)

| | | | |
|---|---|---|---|
| SIOS | Indicate transfer start/stop | 0 : Stop<br>1 : Start | write only |
| SIOINH | Continue/abort transfer | 0 : Continue transfer<br>1 : Abort transfer (automatically cleared after abort) | |
| SIOM | Transfer mode select | 000 : 8-bit transmit mode<br>010 : 4-bit transmit mode<br>100 : 8-bit transmit/receive mode<br>101 : 8-bit receive mode<br>110 : 4-bit receive mode | |
| SCK | Serial clock select | 000 : Internal clock fc / $2^{13}$ or fs / $2^5$ [Hz]<br>001 : Internal clock fc / $2^8$<br>010 : Internal clock fc / $2^6$<br>011 : Internal clock fc / $2^5$<br>111 : External clock (input from $\overline{SCK}$ pin)<br>(Output on $\overline{SCK}$ pin) | |

*Note 1 : fc ; High-frequency clock [Hz], fs ; Low-frequency clock [Hz]*

*Note 2 : Set SIOS to "0" and SIOINH to "1" when setting the transfer mode or serial clock.*

*Note 3 : SIO1CR2/SIO2CR2 are write-only registers, which cannot access any of in read-modify-write instruction such as bit operate, etc.*

## SIO1, SIO2 Status Registers

**SIO1SR** ($0020_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SIOF | SEF | "1" | "1" | "1" | "1" | "1" | "1" |

**SIO2SR** ($0022_H$)

| | | | |
|---|---|---|---|
| SIOF | Serial transfer operating status monitor | 0 : Transfer terminated<br>1 : Transfer in process | read only |
| SEF | Shift operating status monitor | 0 : Shift operation terminated<br>1 : Shift operation in process | |

## SIO1, SIO2 Control Registers 2

**SIO1CR2** ($0021_H$)
**SIO2CR2** ($0023_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | WAIT | | BUF | | |

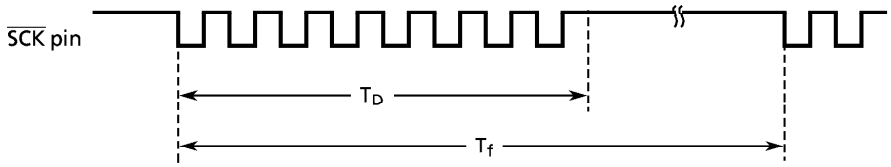
(Initial value: ***0 0000)

| | | | |
|---|---|---|---|
| WAIT | Wait control | 00 : $T_f = T_D$<br>01 : $T_f = 2T_D$<br>10 : $T_f = 4T_D$<br>11 : $T_f = 8T_D$ | Write only |
| BUF | Number of transfer words | Buffer address used<br>SIO1 / SIO2<br>000 : 1 word transfer — $0FF0_H$ / $0FF8_H$<br>001 : 2 words transfer — 0FF0 to $0FF1_H$ / 0FF8 to $0FF9_H$<br>010 : 3 words transfer — 0FF0 to $0FF2_H$ / 0FF8 to $0FFA_H$<br>011 : 4 words transfer — 0FF0 to $0FF3_H$ / 0FF8 to $0FFB_H$<br>100 : 5 words transfer — 0FF0 to $0FF4_H$ / 0FF8 to $0FFC_H$<br>101 : 6 words transfer — 0FF0 to $0FF5_H$ / 0FF8 to $0FFD_H$<br>110 : 7 words transfer — 0FF0 to $0FF6_H$ / 0FF8 to $0FFE_H$<br>111 : 8 words transfer — 0FF0 to $0FF7_H$ / 0FF8 to $0FFF_H$ | |

*Note 1 : *; don't care*

*Note 2 : WAIT is valid only in the 8-bit transmit / receive and 8-bit receive modes.*

*Note 3 : $T_f$; frame time, $T_D$ ; data transfer time*




*Note 4 : The lower 4 bits of each buffer are used during 4-bit transfers. Zeros (0) are stored to the upper 4bits when receiving.*

*Note 5 : Transmitting starts at the lowest address. Received data are also stored starting from the lowest address to the highest address. For example, in the case of SIO1, the first buffer address transmitted is $0FF0_H$.*

*Note 6 : The value to be loaded to BUF is held after transfer is completed.*

*Note 7 : SIO1CR2/SIO2CR2 are write-only registers, which cannot access any of in read-modify-write instruction such as bit operate, etc.*

Figure 2-34. SIO Control Registers and Status Registers

(1) Serial Clock

a. Clock Source

SCK (bits 2 to 0 in SIO1CR1/SIO2CR1) is able to select the following:

① Internal Clock

Any of four frequencies can be selected. The serial clock is output to the outside on the $\overline{SCK1}$ / $\overline{SCK2}$ pin. The $\overline{SCK}$ pin goes high when transfer starts.

When data writing (in the transmit mode) or reading (in the receive mode or the transmit/receive mode) cannot keep up with the serial clock rate, there is a wait function that automatically stops the serial clock and holds the next shift operation until the read/write processing is completed.

Table 2-8. Serial Clock Rate

| Serial clock | | | Maximum transfer rate | |
|---|---|---|---|---|
| NORMAL1/2, IDLE1/2 mode | | SLOW, SLEEP mode | | |
| DV7CK = 0 | DV7CK = 1 | | At fc = 8 MHz | At fs = 32.768 kHz |
| $fc / 2^{13}$ [Hz] | $fs / 2^5$ [Hz] | $fs / 2^5$ [Hz] | 0.954 kbit/s | 1 kbit/s |
| $fc / 2^8$ | $fc / 2^8$ | – | 30.5 | – |
| $fc / 2^6$ | $fc / 2^6$ | – | 122 | – |
| $fc / 2^5$ | $fc / 2^5$ | – | 244 | – |

*Note : 1Kbit = 1024bit*

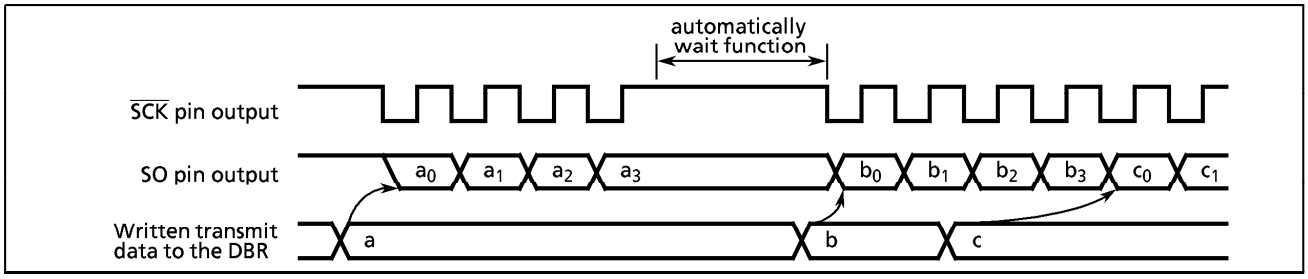
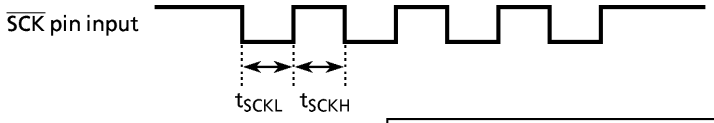



Figure 2-35. Clock Source (Internal Clock)

② External Clock

An external clock connected to the SCK1 / SCK2 pin is used as the serial clock. In this case, the P42 (SCK1) / P45 (SCK2)output latch must be set to "1". To ensure shifting, a pulse width of at least 4 machine cycles is required. Thus, the maximum transfer speed is 244K-bit/s. (at fc = 8 MHz).




*Note : tcyc = 4/fc (In NORAML1/2, IDLE1/2 modes)*
*4/fs (In SLOW, SLEEP modes)*

b. Shift edge

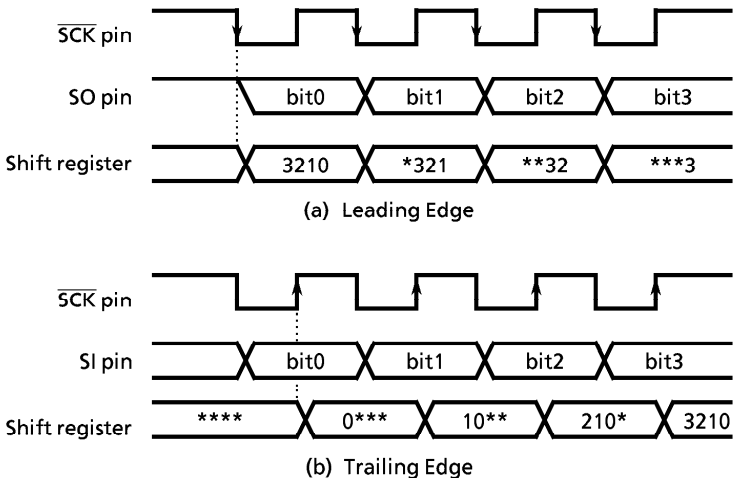
The leading edge is used to transmit, and the trailing edge is used to receive.

① Leading Edge

Transmitted data are shifted on the leading edge of the serial clock (falling edge of the SCK pin input/output).

② Trailing Edge

Received data are shifted on the trailing edge of the serial clock (rising edge of the SCK pin input/output).




*Note : * ; don't care*

Figure 2-36. Shift Edge

(2) Number of Bits to Transfer

Either 4-bit or 8-bit serial transfer can be selected. When 4-bit serial transfer is selected, only the lower 4 bits of the transmit/receive data buffer register are used. The upper 4 bits are cleared to "0" when receiving.

The data is transferred in sequence starting at the least significant bit (LSB).

(3) Number of Words to Transfer

Up to 8 words consisting of 4 bits of data (4-bit serial transfer) or 8 bits (8-bit serial transfer) of data can be transferred continuously. The number of words to be transferred is loaded to BUF1/BUF2 in SIOBCR.

An INTSIO interrupt is generated when the specified number of words has been transferred. If the number of words is to be changed during transfer, the serial interface must be stopped before making the change.

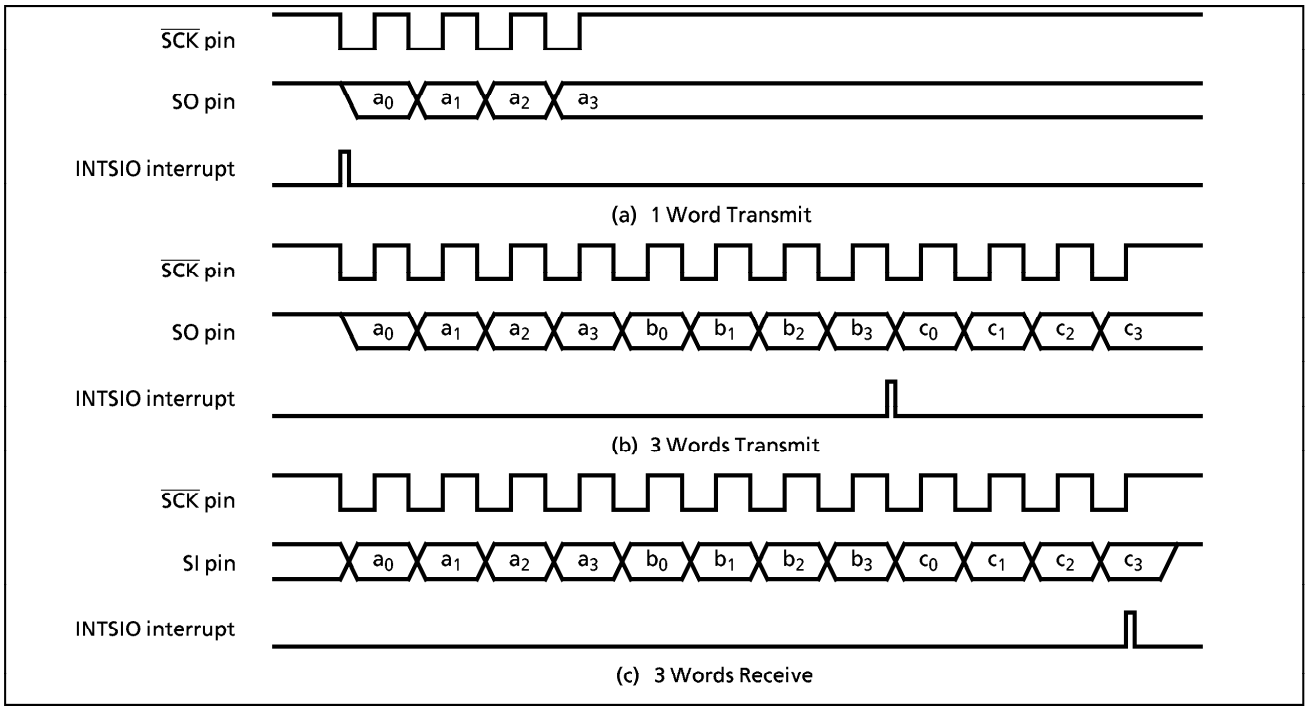



Figure 2-37. Number of Bits to Transfer (Example : 4-bit serial transfer)

## 2.9.3 Transfer Mode

SIOM (bits 5 to 3 in SIO1CR1/SIO2CR1) is used to select the transmit, receive, or transmit/receive mode.

(1) **4-bit and 8-bit Transmit** Modes

In these modes, the SIO1CR1/SIO2CR1 is set to the transmit mode and then the data to be transmitted first are written to the data buffer registers (DBR). After the data are written, the transmission is started by setting SIOS to "1". The data are then output sequentially to the SO pin in synchronous with the serial clock, starting with the least significant bit (LSB). As soon as the LSB has been output, the data are transferred from the data buffer register to the shift register. When the final data bit has been transferred and the data buffer register is empty, an INTSIO (buffer empty) interrupt is generated to request the next transmitted data.

When the internal clock is used, the serial clock will stop and an automatic-wait will be initiated if the next transmitted data are not loaded to the data buffer register by the time the number of data words specified with the BUF has been transmitted. Writing even one word of data cancels the automatic-wait; therefore, when transmitting two or more words, always write the next word before transmission of the previous word is completed.

*Note : Waits are also canceled by writing to a DBR not being used as a transmit data buffer register; therefore, during SIO do not use such DBR for other applications.*

When an external clock is used, the data must be written to the data buffer register before shifting next data. Thus, the transfer speed is determined by the maximum delay time from the generation of the interrupt request to writing of the data to the data buffer register by the interrupt service program.

When the transmit is started, after the SIOF goes "1" output from the SO pin holds final bit of the last data until falling edge of the $\overline{SCK}$.
The transmission is ended by clearing SIOS to "0" at the time that the final bit of the data being shifted out has been transferred. That the transmission has ended can be determined from the status of SIOF (bit 7 in SIO1SR/SIO2SR) because SIOF is cleared to "0" when a transfer is completed.
When an external clock is used, it is also necessary to clear SIOS to "0" before shifting the next data; otherwise, dummy data will be transmitted and the operation will end.

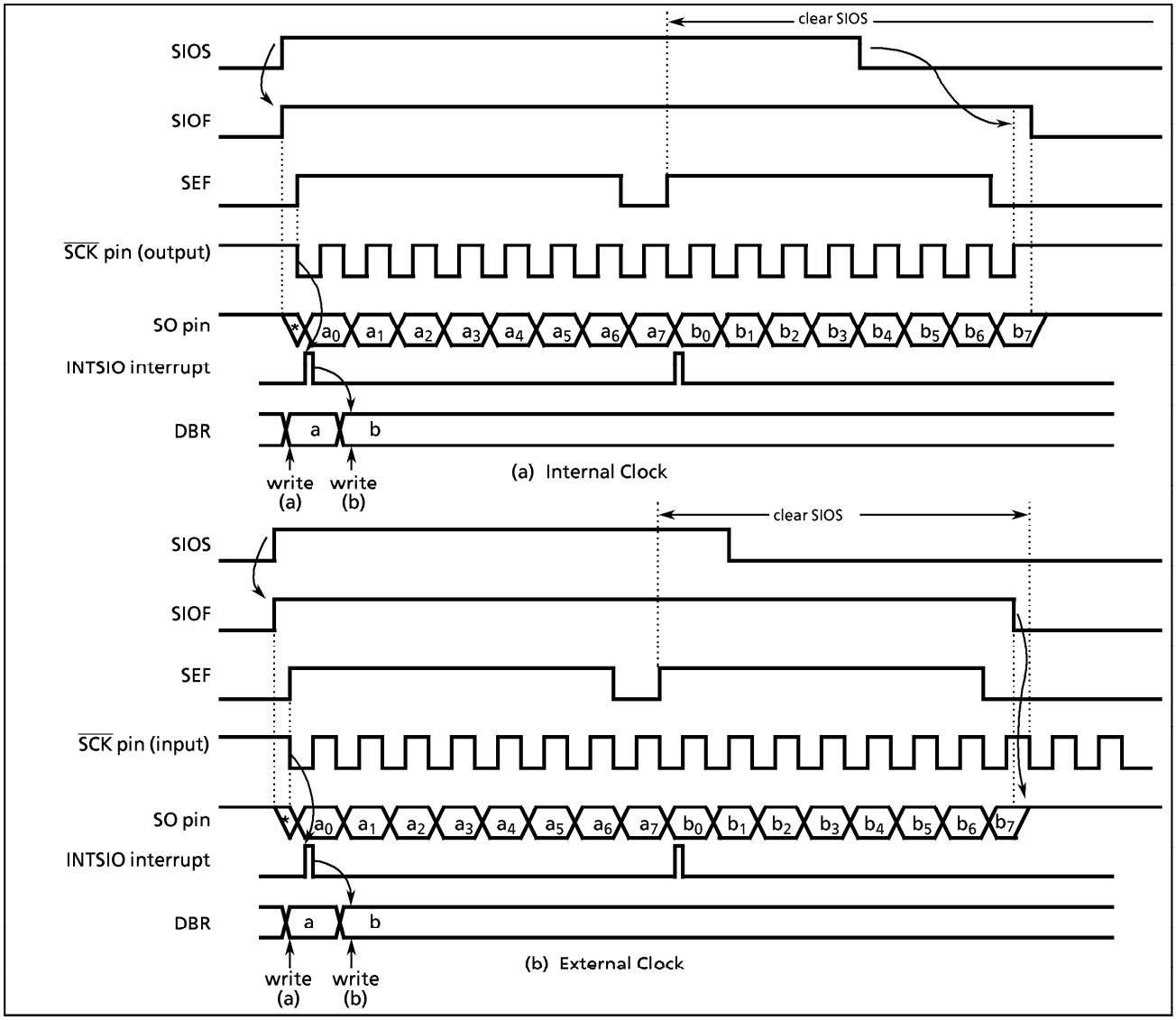



Figure 2-38. Transfer Mode (Example: 8-bit, 1 Word Transfer)

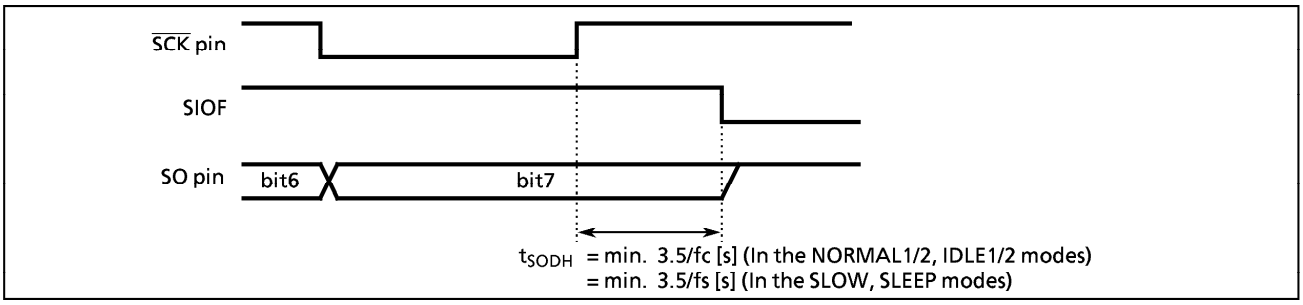



Figure 2-39. Transmitted Data Hold Time at end of transmit

(2) **4-bit and 8-bit Receive** Modes

After setting the control registers to the receive mode, set SIOS to "1" to enable receiving. The data are then transferred to the shift register via the SI pin in synchronous with the serial clock. When one word of data has been received, it is transferred from the shift register to the data buffer register (DBR). When the number of words specified with the BUF has been received, an INTSIO (buffer full) interrupt is generated to request that these data be read out. The data are then read from the data buffer registers by the interrupt service program.

When the internal clock is used, and the previous data are not read from the data buffer register before the next data are received, the serial clock will stop and an automatic-wait will be initiated until the data are read. A wait will not be initiated if even one data word has been read.

*Note : Waits are also canceled by reading a DBR not being used as a received data buffer register is read; therefore, during SIO do not use such DBR for other applications.*

When an external clock is used, the shift operation is synchronized with the external clock; therefore, the previous data are read before the next data are transferred to the data buffer register. If the previous data have not been read, the next data will not be transferred to the data buffer register and the receiving of any more data will be canceled. When an external clock is used, the maximum transfer speed is determined by the delay between the time when the interrupt request is generated and when the data received have been read.

Clear SIOS to "0" to end receiving. When SIOS is cleared, the current data are transferred to the buffer in 4-bit or 8-bit blocks. The receiving mode ends when the transfer is completed. SIOF is cleared to "0" when receiving is ended and thus can be sensed by program to confirm that receiving has ended.

*Note : The buffer contents are lost when the transfer mode is switched. If it should become necessary to switch the transfer mode, end receiving by clearing SIOS to "0", read the last data and then switch the transfer mode.*

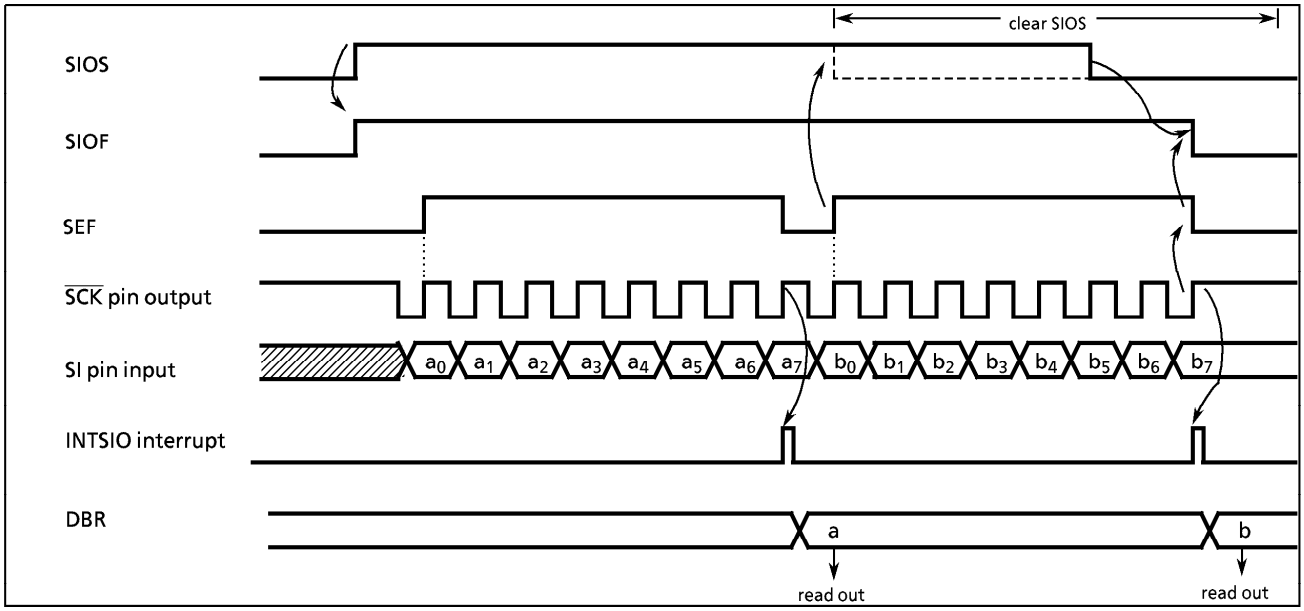



Figure 2-40. Receive Mode (Example : 8-bit, 1 word, internal clock)

(3) **8-bit Transmit/Receive** Mode

After setting the control registers to the 8-bit transmit/receive mode, write the data to be transmitted first to the data buffer registers (DBR). After that, enable transceiving by setting SIOS to "1". When transmitting, the data are output from the SO pin at leading edges of the serial clock. When receiving, the data are input to the SI pin at the trailing edges of the serial clock. 8-bit data are transferred from the shift register to the data buffer register. An INTSIO interrupt is generated when the number of data words specified with the BUF has been transferred. The interrupt service program reads the received data from the data buffer register and then writes the data to be transmitted. The data buffer register is used for both transmitting and receiving; therefore, always write the data to be transmitted after reading the received data.

When the internal clock is used, a wait is initiated until the received data are read and the next data are written.

When an external clock is used, the shift operation is synchronized with the external clock; therefore, it is necessary to read the received data and write the data to be transmitted next before starting the next shift operation. When an external clock is used, the transfer speed is determined by the maximum delay between generation of an interrupt request and the received data are read and the data to be transmitted next are written.

When the transmit is started, after the SIOF goes "1" output from the SO pin holds final bit of the last data until falling edge of the $\overline{SCK}$.

Clear SIOS to "0" to enable the transmit mode. When SIOS is cleared, the current data are transferred to the data buffer register in 8-bit blocks. The transmit mode ends when the transfer is completed. SIOF is cleared to "0" when receiving is ended and thus can be sensed by program to confirm that receiving has ended.

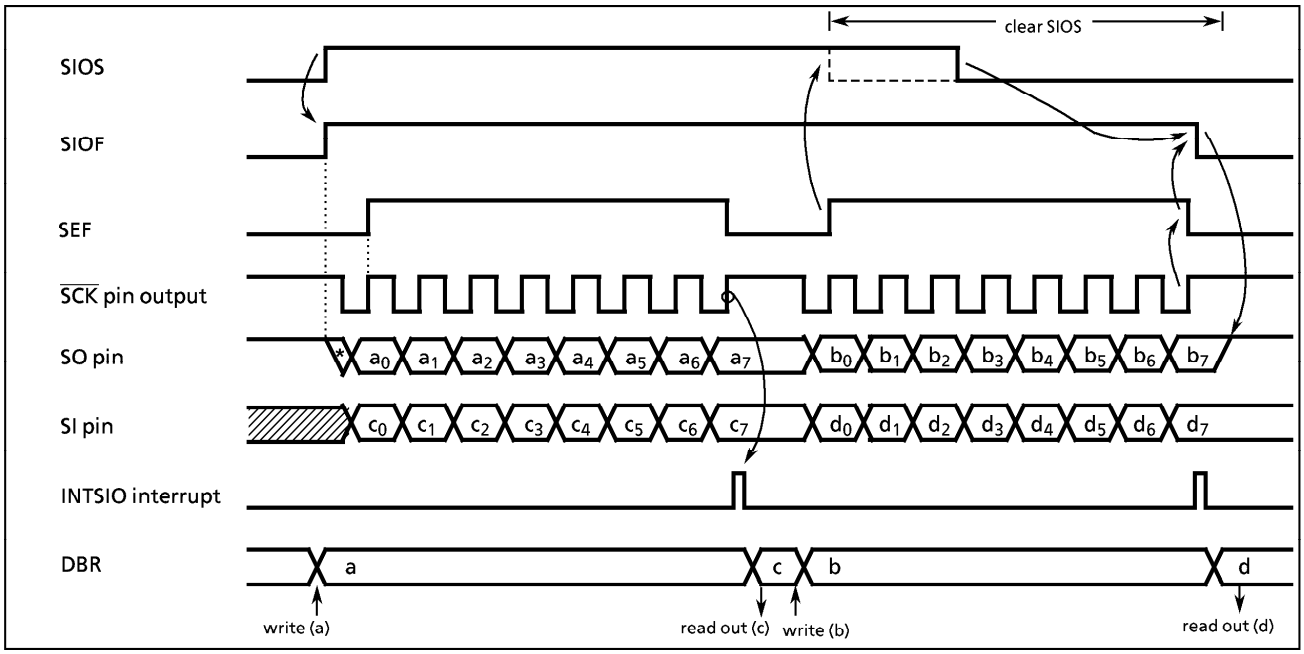



Figure 2-41. Transmit/Receive Mode (Example : 8-bit, 1word, internal clock)

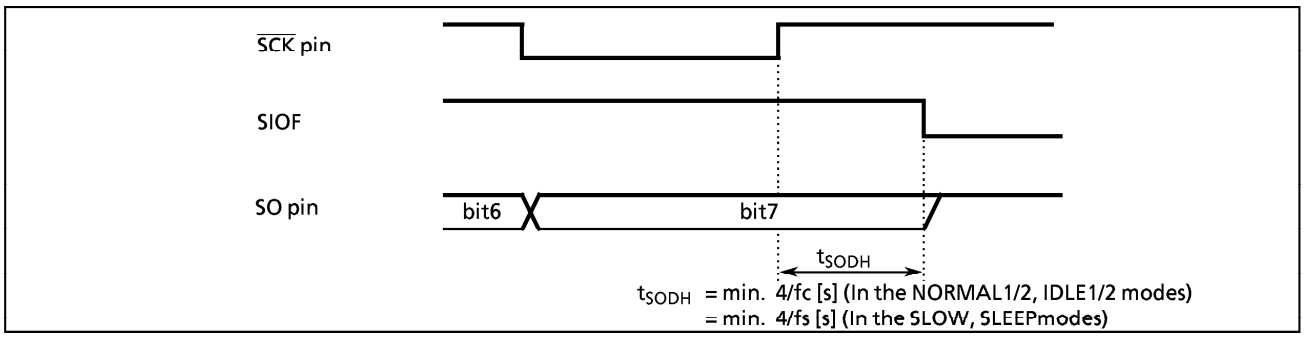



Figure 2-42. Transmitted Data Hold Time at end of transmit/receive

## 2.10 10-bit A/D Converter (ADC)

The 87C841/CC41/CH41/CK41/CM41 each have an 16-channel multiplexed-input 10-bit successive approximate type A/D converter.

### 2.10.1 Configuration

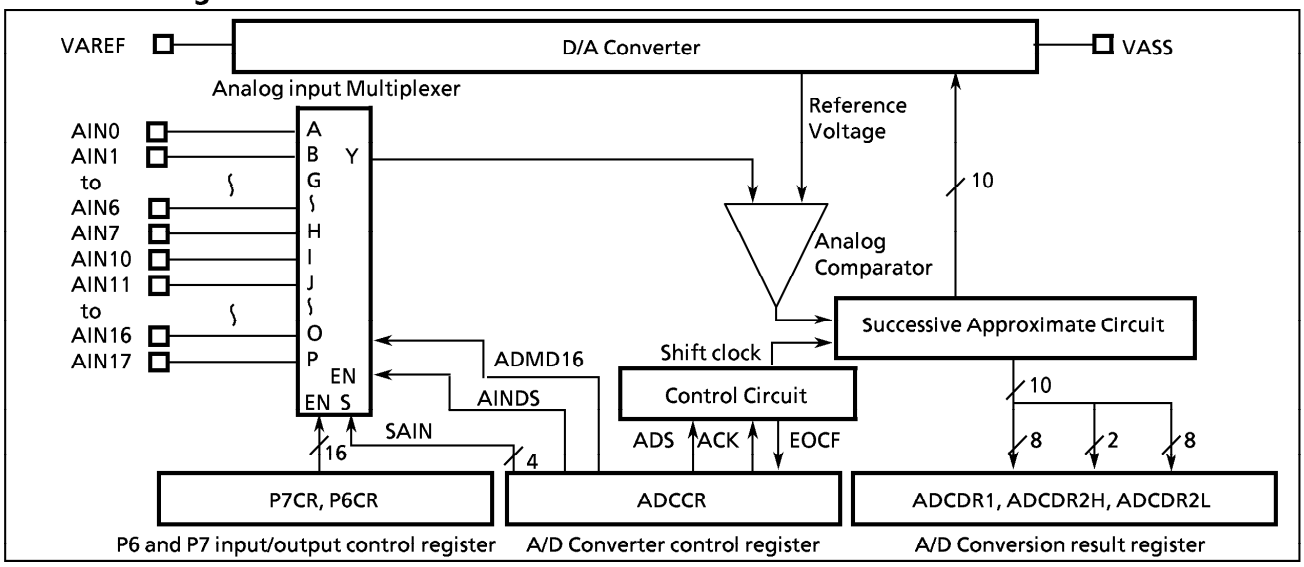



Figure 2-43. A/D Converter

### 2.10.2 Control

The A/D converter is controlled by an A/D converter control register (ADCCR), a port P6 input/output control register (P6CR) and a port P7 input/output control register (P7CR).

A/D Converter Control Register

ADCCR ($000E_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| EOCF/ ADMD16 | ADS | ACK | AINDS | SAIN | | | |

(Intial value : 0000 0000)

| | | | |
|---|---|---|---|
| SAIN | Analog input selection | 0000 : AIN0<br>0001 : AIN1<br>0010 : AIN2<br>0011 : AIN3<br>0100 : AIN4<br>0101 : AIN5<br>0110 : AIN6<br>0111 : AIN7<br>1000 : AIN10<br>1001 : AIN11<br>1010 : AIN12<br>1011 : AIN13<br>1100 : AIN14<br>1101 : AIN15<br>1110 : AIN16<br>1111 : AIN17 | R/W |
| AINDS | Analog input control | 0 : Enable<br>1 : Disable | |
| ACK | Conversion time | 0 : 196/fc : (at 8 MHz 24.5 μs). (at 4.2 MHz 46.7 μs)<br>1 : 736/fc : (at 8 MHz 92 μs). (at 4.2 MHz 176 μs) | |
| ADS | A/D convertion start | 0 : –<br>1 : A/D convertion start | |
| EOCF | End of A/D conversion flag | 0 : Under conversion or before conversion<br>1 : End of conversion | Read only |
| ADMD16 | Number of analog input selection | 0 : 8ch (AIN0 to 7)<br>1 : 16ch (AIN10 to 17) | Write only |

*Note 1 : Select analog input when A/D converter stops.*
*Note 2 : The ADS is automatically cleared to "0" after starting conversion.*
*Note 3 : The EOCF is cleared to "0" when reading the ADCDR1 or the ADCDR2H, ADCDR2L.*
*Note 4 : The EOCF is read-only.*
*Note 5 : The ADMD16 is write-only.*
*Note 6 : Select 16ch mode (ADMD16 = 1) when selecting from the AIN 10 to the AIN17.*

Figure 2-44. A/D Converter Control Register

A/D Conversion Result Register

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| ADCDR1 ($000F_H$) | DATA9 | DATA8 | DATA7 | DATA6 | DATA5 | DATA4 | DATA3 | DATA2 | Read only |
| ADCDR2H ($0025_H$) | "1" | "1" | "1" | "1" | "1" | "1" | DATA9 | DATA8 | Read only |
| ADCDR2L ($0024_H$) | DATA7 | DATA6 | DATA5 | DATA4 | DATA3 | DATA2 | DATA1 | DATA0 | Read only |

P6 input/output control register

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| P6CR ($000C_H$) | | | | | | | | | (Initial value : 1111 1111) |

| | | | |
|---|---|---|---|
| P6CR | P6 input control | 0 : input port<br>1 : analog input (AINDS = 0) | write only |

P7 input/output control register

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| P7CR ($000D_H$) | | | | | | | | | (Initial value : 0000 0000) |

| | | | |
|---|---|---|---|
| P7CR | P7 input/output control | 0 :<br>1 : } See below | write only |

| ADCCR | | P7CR | |
|---|---|---|---|
| ADMD16 | AINDS | 0 | 1 |
| 0 | 0 | input port | output port |
| 1 | 0 | input port except analog input selected by SAIN | output port except analog input selected by SAIN |
| 0 | 1 | input port | output port |
| 1 | 1 | input port | output port |

Figure 2-45. A/D Converter Result Register and P6, P7 Input/Output Control Register

## 2.10.3 Operation

Apply analog reference voltage to pins VAREF and VASS.

(1) Start of A/D conversion

First, select one of sixteen analog inputs AIN17 to AIN10, AIN7 to AIN0 with the SAIN (bits 3 to 0 in ADCCR) and the ADMD16 (bit 7 in ADCCR). Clear the AINDS (bit 4 in ADCCR) to "0" and set the corresponding P6CR bit to "1" for an analog input when AIN7 to AIN0 is selected.

A/D conversion time is set with the ACK (bit 5 in ADCCR).

A/D conversion is started by setting the ADS (bit 6 in ADCCR) to "1". The conversion is accomplished in 46 machine cycles (184/fc [s], ACK = 0).

The EOCF (bit 7 in ADCCR) is set to "1" at the end of the conversion.

If the ADS (bit 6 in ADCCR) is set to "1" during the conversion, the operation is initialized and the conversion is started again.

*Note : The pin that is not used as an analog input can be used as regular input or output pins. During conversion, do not perform output instruction to maintain a precision for all of the pins.*

(2) Reading of A/D conversion result

After the end of the conversion, read the conversion result from the ADCDR1 or the ADCDR2H, ADCDR2L.

The EOCF is automatically cleared to "0" when reading the ADCDR1 or the ADCDR2H, ADCDR2L.

Reading the conversion result during A/D conversion, an unexpected value is given.

(3) A/D conversion in STOP mode

When the MCU places in the STOP mode during the A/D conversion, the conversion is terminated and the ADCDR1 or ADCDR2H, ADCDR2L contents become indefinite.

However, if the STOP mode is started after the end of conversion (EOCF = 1), the ADCDR1 or the ADCDR2H, ADCDR2L contents are held.

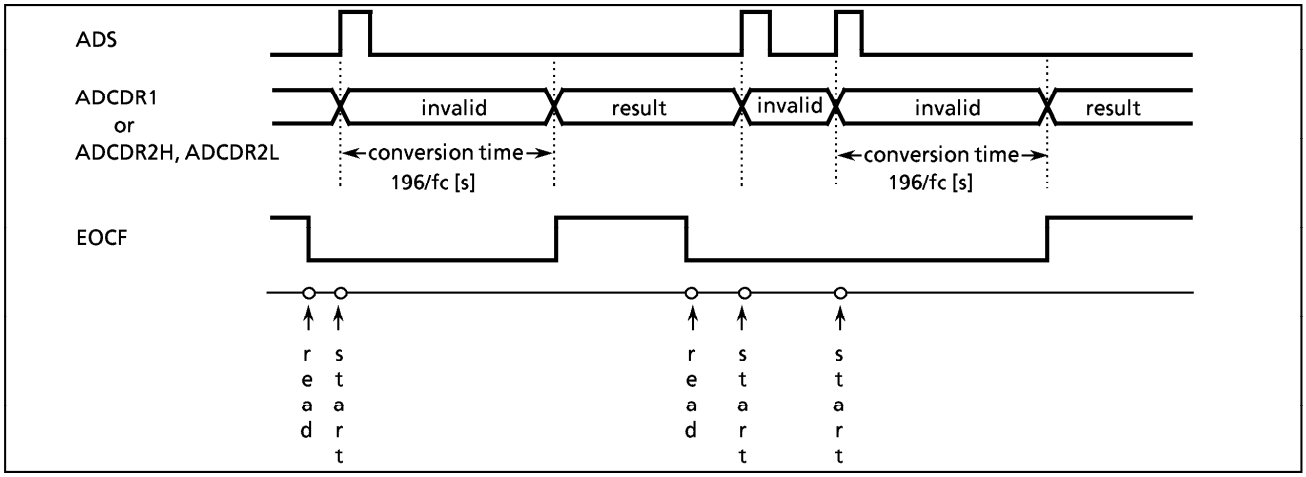



Figure 2-46. A/D Conversion Timing Chart

Example:

```
                ; AIN SELECT
                LD          (ADCCR) , 00100100B     ;   selects conversion time and AIN4
                ; A/D CONVERT START
                SET         (ADCCR) . 6             ;   ADS = 1
SLOOP    :      TEST        (ADCCR) . 7             ;   EOCF = 1 ?
                JRS         T, SLOOP
                ; RESULT DATA READ
                LD          (9EH), (ADCDR2H)
                LD          (9DH), (ADCDR2L)
```

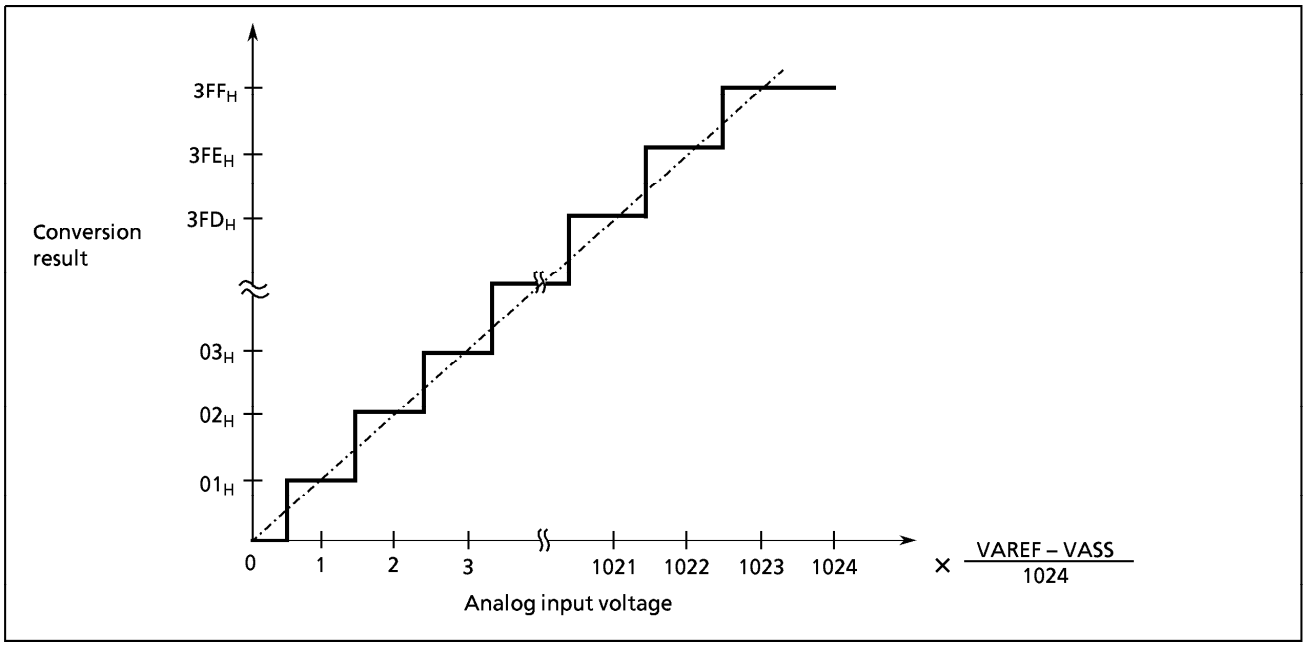



Figure2-47. Analog Input Voltage vs A/D Conversion Result (typ.)

## 2.11 8-bit High-speed PWM

The 87C841/CC41/CH41/CK41/CM41 have an internal 2-channel high-speed PWM. The 2-channel high-speed PWM can generate different waves from a data register of each channel by writing data to the register.

The 2-channel high-speed PWM are also used as a port P53 ($\overline{\text{HPWM0}}$) and a port P54 ($\overline{\text{HPWM1}}$). When used as the hig-speed PWM pin, the output latch should be set to "1".

### 2.11.1 Configuration

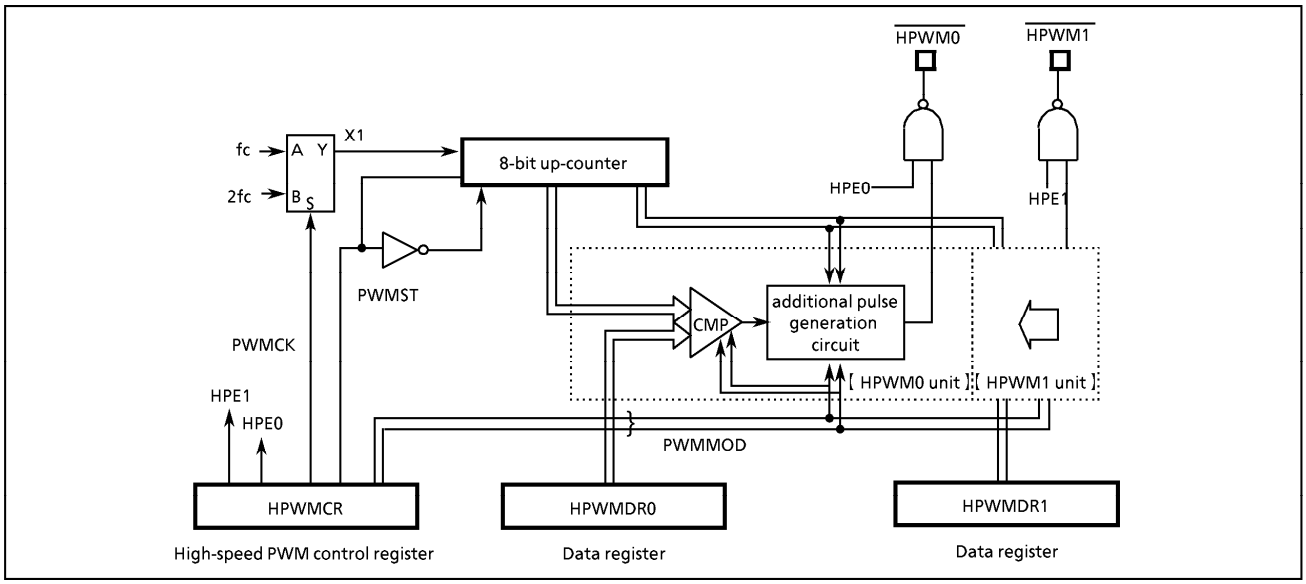



Figure 2-48. High-speed

### 2.11.2 Control

Control register

HPWMCR ($0026_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| HPE1 | HPE0 | | | PWMST | PWMCK | PWMMOD | |

Write only (initial value : 00** 0000)

| | | | |
|---|---|---|---|
| PWMMOD | PWM mode selection | 00 : mode 0 (8 bits)<br>01 : mode 1 (7 bits)<br>10 : mode 2 (6 bits)<br>11 : – | write only |
| PWMCK | PWM source clock (X1) selection | 0 : fc<br>1 : 2fc (only mode 0) | |
| PWMST | RUN/STOP of 8-bit up-counter | 0 : STOP<br>1 : RUN | |
| HPE0 | HPWM0 output control | 0 : disable<br>1 : enable | |
| HPE1 | HPWM1 output control | 0 : disable<br>1 : enable | |

Data register

HPWMDR0 ($0027_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DATA7 | DATA6 | DATA5 | DATA4 | DATA3 | DATA2 | DATA1 | DATA0 |

Write only (initial value : **** ****)

HPWMDR1 ($0028_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DATA7 | DATA6 | DATA5 | DATA4 | DATA3 | DATA2 | DATA1 | DATA0 |

Write only (initial value : **** ****)

Figure 2-49. Registers of High-speed PWM

### 2.11.3 Operation

The high-speed PWM is controlled by the control register (HPWMCR) and data register (HPWMR0, 1). To write data to the above registers, set the PWMST (bit 3 in HPWMCR) to "1", which is enable mode. When the PWMST is clear to "0", these registers are in reset mode, which sets up the high-speed PWM for a software reset.

(1) Operation mode

The high-speed PWM has three operation modes.

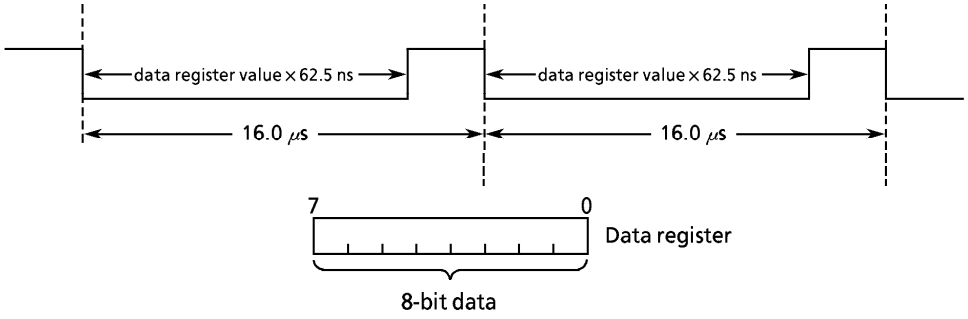
- 8-bit mode : ($T = 2^8 \times$ clock cycle $f \fallingdotseq 32$ kHz, $f \fallingdotseq$ *64 kHz)
- 7-bit mode : ($T = 2^7 \times$ clock cycle $f \fallingdotseq 64$ kHz)
- 6-bit mode : ($T = 2^6 \times$ clock cycle $f \fallingdotseq 128$ kHz)

*Note : * indicates the value when the source clock (X1) operates at 16 MHz (at 2fc : fc = 8 MHz).*

Operation mode is set by the PWMMOD (bit 0, 1 in HPWMCR). Operation mode applies commonly to all channels. Two modes cannot be used at any given time.

① 8-bit mode

8-bit mode generates a pulse with 16.0 μs cycle at a frequency of approximately 64 kHz (X1 = 16 MHz).




The minimum pulse width is 62.5 ns (data "1") and the maximum pulse width is 15.0 μs (data "F0").

Pulse width = 8-bit data × 62.5ns

A wave cycle example is shown the figure 2-50. (The value is when X1 = 16 MHz)

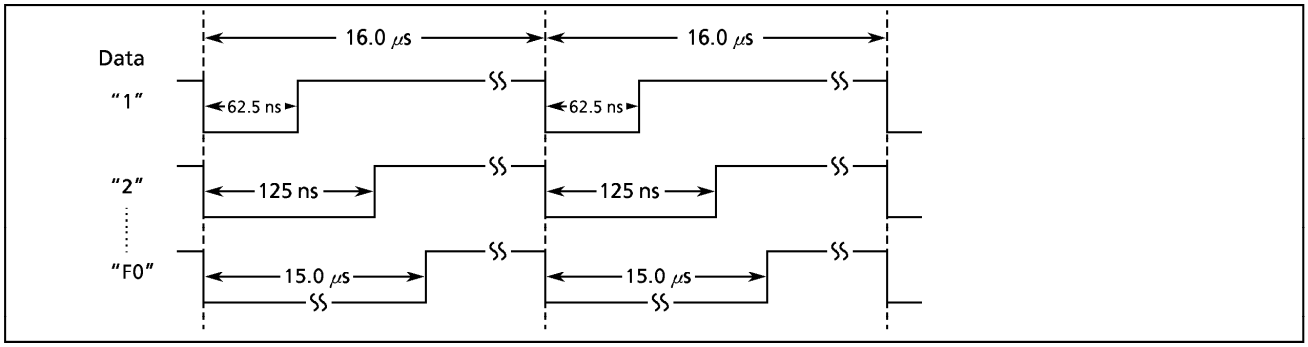
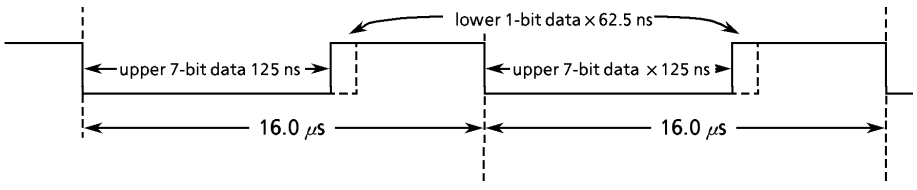


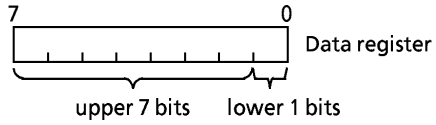

Figure 2-50. 8-bit mode

② 7-bit mode

7-bit mode generates a pulse with 16.0 μs cycle at a frequency of approximately 64 kHz (X1 = 8 MHz).



7-bit mode has 7 bits for a cycle ($2^7$ × 125 ns/cycle) and 1 bit for a 62.5 ns resolution (1/2 cycle of source clock (X1)).
When the lower 1 bit is "1", the additional 62.5 ns pulse is output.
The minimum pulse width is 62.5 ns (data "1") and the maximum pulse width is 15.0625 μs (data "F1").




Pulse width = (Upper 7-bit data × 125 ns) + (Lower 1-bit data × 62.5 ns)
A wave cycle example is shown the figure 2-51. (The value is when X1 = 8 MHz)

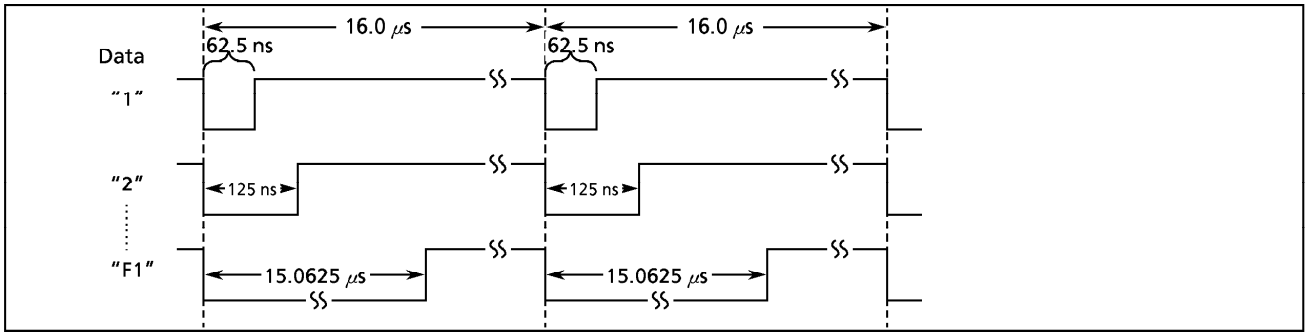
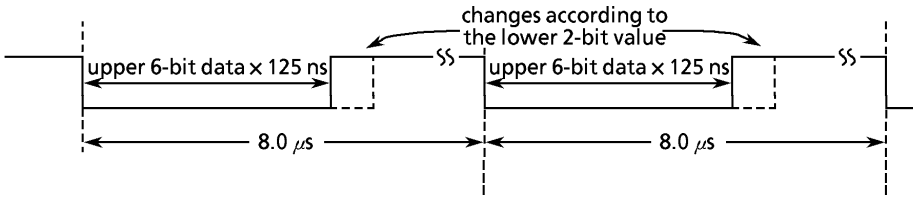
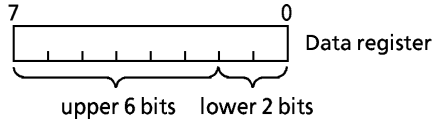



Figure 2-51. 7-bit mode

③ 6-bit mode
6-bit mode generates a pulse with 8.0 μs cycle at a frequency of approximately 128 kHz (X1 = 8 MHz).




6-bit mode has 6 bits for a cycle ($2^6$ × 125 ns/cycle) and 2 bits for a 31.25 ns resolution. Although the actual resolution every other cycle is 62.5 ns, a 31.25 ns resolution is simulated in the following way : The first cycle outputs a 62.5 ns pulse, which is averaged with the second cycle of 0ns. This pattern alternates continually and averages to a 62.5 ns (data "1") and the maximum equivalent pulse width is 7.625 μs (data "F3").




Pulse width = (upper 6-bit data × 125 ns) + (*lower 2-bit data)
*Equivalent time added for lower 2-bit data is shown below.

| 2-bit data | | Equivalent time added | |
|---|---|---|---|
| 0 | 0 | 0 | ns |
| 0 | 1 | 31.25 | ns |
| 1 | 0 | 62.5 | ns |
| 1 | 1 | 93.75 | ns |

A wave cycle example is shown the figure 2-52. (The value is when X1 = 8 MHz.)

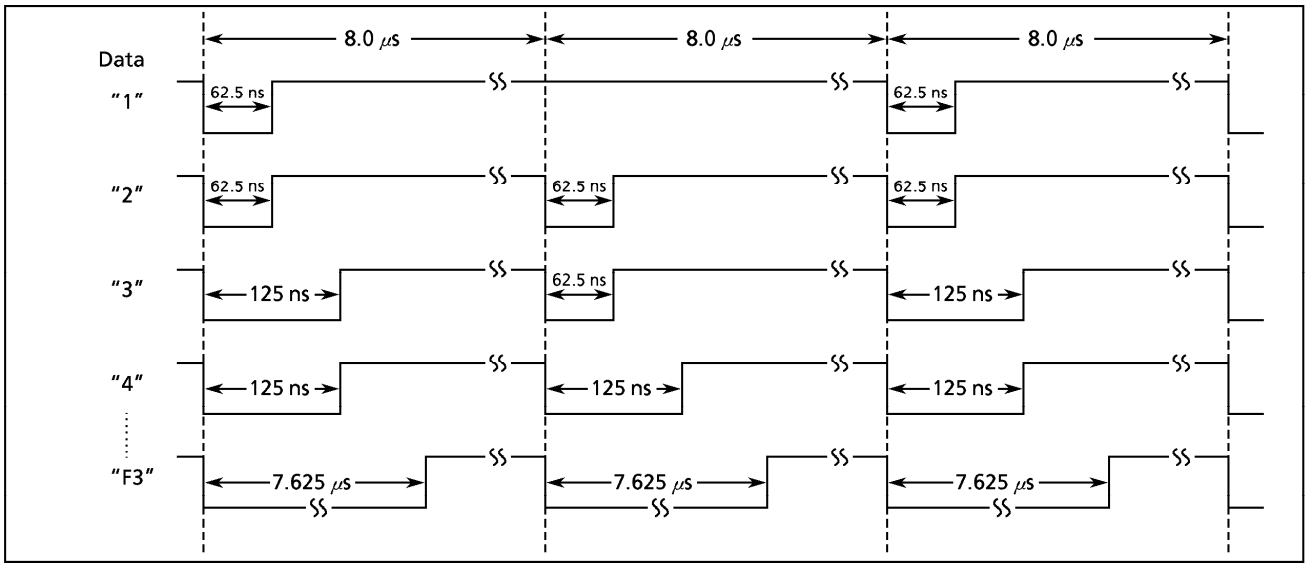



Figure 2-52. 6-bit mode

(2) Output data setting

Output data is set by writing to data registers (HPWMDR0, 1).

Example : To output 11.25 μs wave with $\overline{\text{HPWM0}}$ in 7-bit mode with a source clock (X1) = 8 MHz:

When the resolution in 7-bit mode is 62.5 ns, a 11.5μs pulse is output by setting the following to HPWMDR0.

11.5 μs ÷ 62.5 ns = 184 = B8H

## 2. ON-CHIP PERIPHERALS FUNCTIONS

### 2.1 Special Function Registers (SFR) and Data Buffer Registers (DBR)

The TLCS-870 Series uses the memory mapped I/O system, and all peripheral control and data transfers are performed through the special function registers (SFR) and data buffer registers (DBR).
The SFR are mapped to addresses $0000_H$ to $003F_H$, and the DBR to addresses $0F80_H$ to $0FFF_H$.
Figure 2-1 shows the 87C841/CC41/CH41/CK41/CM41 SFRs and DBRs.

| Address | Read | Write |
|---|---|---|
| $0000_H$ | P0 port | |
| 01 | P1 Port | |
| 02 | P2 Port | |
| 03 | P3 Port | |
| 04 | P4 Port | |
| 05 | P5 Port | |
| 06 | P6 Port | |
| 07 | P7 Port | |
| 08 | reserved | |
| 09 | reserved | |
| 0A | – | P0CR (P0 I/O control) |
| 0B | – | P1CR (P1 I/O control) |
| 0C | – | P6CR (P6 I/O control) |
| 0D | – | P7CR (P7 I/O control) |
| 0E | ADCCR (A/D converter control) | |
| 0F | ADCDR1 (A/D conv. upper 8-bit result) | – |
| 10 | – | $TREG1A_L$ (Timer register 1A) |
| 11 | – | $TREG1A_H$ |
| 12 | $TREG1B_L$ (Timer register 1B) | |
| 13 | $TREG1B_H$ | |
| 14 | – | TC1CR (TC1 control) |
| 15 | – | TC2CR (TC2 control) |
| 16 | – | $TREG2_L$ (Timer register 2) |
| 17 | – | $TREG2_H$ |
| 18 | TREG3A (Timer register 3A) | |
| 19 | TREG3B (Timer register 3B) | – |
| 1A | – | TC3CR (TC3 control) |
| 1B | – | TREG4 (Timer register 4) |
| 1C | – | TC4CR (TC4 control) |
| 1D | reserved | |
| 1E | reserved | |
| 1F | reserved | |

| Address | Read | Write |
|---|---|---|
| $0020_H$ | SIO1SR (SIO1 status) | SIO1CR1 (SIO1 control) |
| 21 | – | SIO1CR2 |
| 22 | SIO2SR (SIO2 status) | SIO2CR1 (SIO2 control) |
| 23 | – | SIO2CR2 |
| 24 | ADCDR2L (A/D conv. lower 8-bit result) | |
| 25 | ADCDR2H (A/D conv. upper 2-bit result) | |
| 26 | | HPWMCR (HPWM control) |
| 27 | | HPWMDR0 (HPWM0 data register) |
| 28 | | HPWMDR1 (HPWM1 data register) |
| 29 | reserved | |
| 2A | reserved | |
| 2B | reserved | |
| 2C | reserved | |
| 2D | reserved | |
| 2E | reserved | |
| 2F | reserved | |
| 30 | reserved | |
| 31 | reserved | |
| 32 | reserved | |
| 33 | reserved | |
| 34 | – | WDTCR1 (WDT control) |
| 35 | – | WDTCR2 |
| 36 | TBTCR | (TBT / TG / DVO control) |
| 37 | EINTCR | (External interrupt control) |
| 38 | SYSCR1 | (System control) |
| 39 | SYSCR2 | |
| 3A | $EIR_L$ | (Interrupt enable register) |
| 3B | $EIR_H$ | |
| 3C | $IL_L$ | (Interrupt latch) |
| 3D | $IL_H$ | |
| 3E | reserved | |
| 3F | PSW (Program status word) | RBS (Register bank selector) |

(a) Special Function Registers

| Address | Read | Write |
|---|---|---|
| $0F80_H$ | reserved | |
| | reserved | |
| 0FEF | reserved | |
| 0FF0 | SIO1 transmit and receive data buffer | |
| F1 | | |
| F2 | | |
| F3 | | |
| F4 | | |
| F5 | | |
| F6 | | |
| F7 | | |
| 0FF8 | SIO2 transmit and receive data buffer | |
| F9 | | |
| FA | | |
| FB | | |
| FC | | |
| FD | | |
| FE | | |
| FF | | |

(b) Data Buffer Registers

*Note 1 : Do not access reserved areas by the program.*
*Note 2 : – : Cannot be accessed.*
*Note 3 : When defining address $003F_H$ with assembler symbols, use GPSW and GRBS.*
*Note 4 : Write-only registers and interrupt latches cannot use the read-modify-write instructions (bit manipulation instructions such as SET, CLR, etc. and logical operation instructions such as AND, OR, etc.)*

Figure 2-1. SFR & DBR

## 2.2 I/O Ports

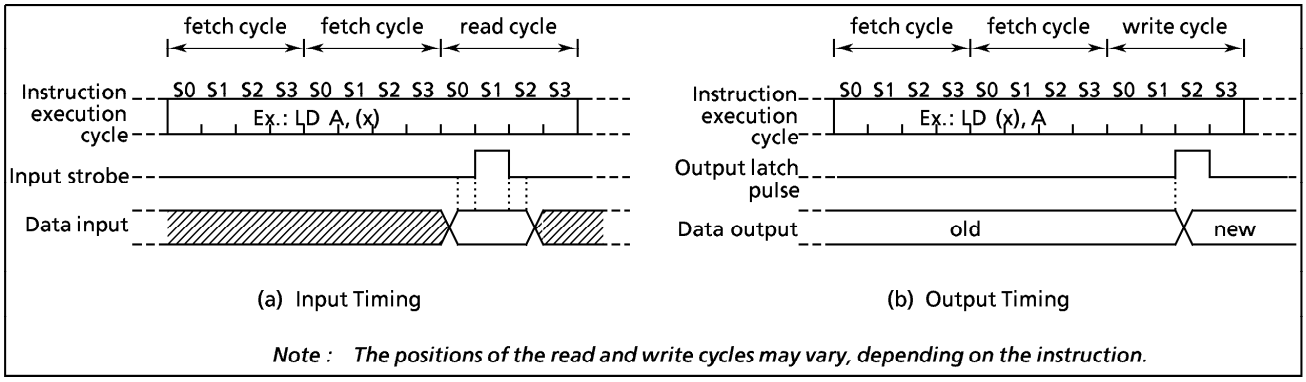
The 87C841/CC41/CH41/CK41/CM41 have 8 parallel input/output ports (56pins) each as follows:

| | Primary Function | Secondary Functions |
|---|---|---|
| Port P0 | 8-bit I/O port | ——— |
| Port P1 | 8-bit I/O port | external interrupt input, timer/counter input/output, and divider output |
| Port P2 | 3-bit I/O port | low-frequency resonator connections, external interrupt input, and STOP mode release signal input |
| Port P3 | 8-bit I/O port | ——— |
| Port P4 | 8-bit I/O port | serial interface |
| Port P5 | 5-bit I/O port | external interrupt input, and timer/counter input/output, high-speed PWM output |
| Port P6 | 8-bit I/O port | analog input |
| Port P7 | 8-bit I/O port | analog input |

Each output port contains a latch, which holds the output data. All input ports do not have latches, so the external input data should either be held externally until read or reading should be performed several times before processing. Figure 2-2 shows input/output timing examples.

External data is read from an I/O port in the S1 state of the read cycle during execution of the read instruction. This timing can not be recognized from outside, so that transient input such as chattering must be processed by the program.

Output data output changes in the S2 state of the write cycle during execution of the instruction which writes to an I/O port.




*Note : The positions of the read and write cycles may vary, depending on the instruction.*

Figure 2-2. Input/Output Timing (Example)

When reading an I/O port except programmable I/O ports, whether the pin input data or the output latch contents are read depends on the instructions, as shown below:

(1) Instructions that read the output latch contents
- ① XCH r, (src)
- ② CLR/SET/CPL (src).b
- ③ CLR/SET/CPL (pp).g
- ④ LD (src).b, CF
- ⑤ LD (pp) . b, CF
- ⑥ ADD/ADDC/SUB/SUBB/AND/OR/XOR (src), n
- ⑦ (src) side of ADD/ADDC/SUB/SUBB/AND/OR/XOR (src), (HL)

(2) Instructions that read the pin input data
- ① Instructions other than the above (1)
- ② (HL) side of ADD/ADDC/SUB/SUBB/AND/OR/XOR (src), (HL)

### 2.2.1 Port P0 (P07 to P00)

Port P0 is an 8-bit general-purpose input/output port which can be configured as either an input or an output in one-bit unit under software control. Input/output mode is specified by the corresponding bit in the port P0 input/output control register (P0CR). Port P0 is configured as an input if its corresponding P0CR bit is cleared to "0", and as an output if its corresponding P0CR bit is set to "1".

During reset, P0CR is initialized to "0", which configures port P0 as input. The P0 output latches are also initialized to "0". Data is written into the output latch regardless of the P0CR contents. Therefore initial output data should be written into the output latch before setting P0CR.

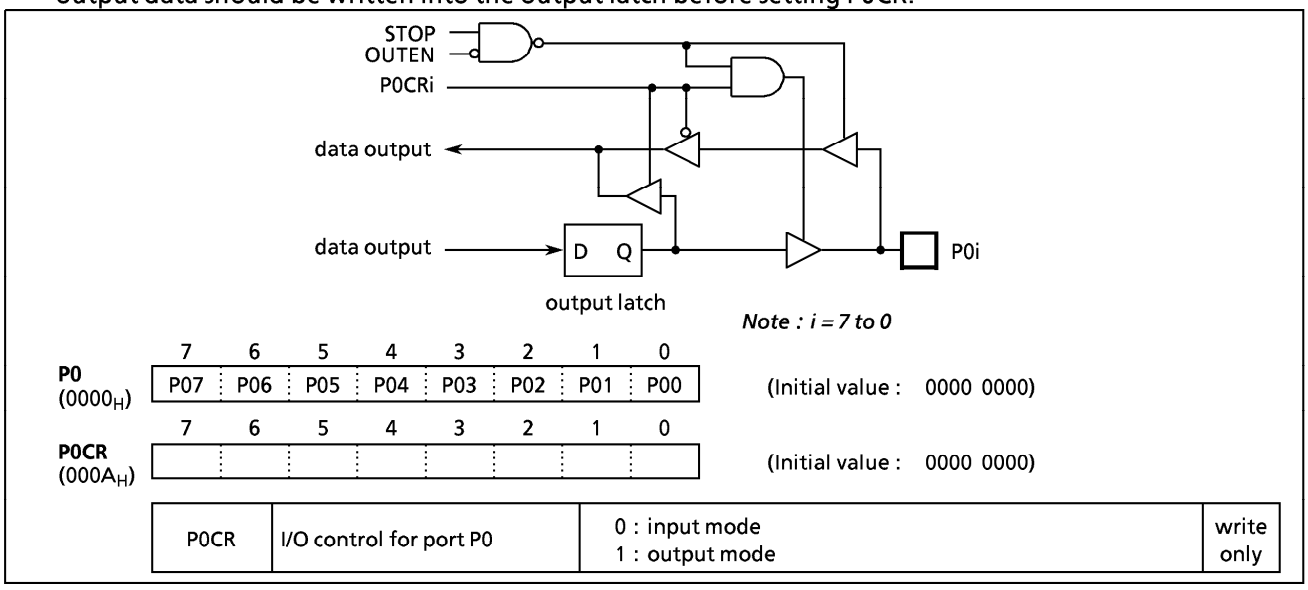



Figure 2-3. Port P0 and P0CR

Example : Setting the upper 4 bits of port P0 as an input port and the lower 4 bits as an output port (Initial output data are $1010_B$).

```
LD      (P0), 00001010B     ; Sets initial data to P0 output latches
LD      (P0CR), 00001111B   ; Sets the port P0 input/output mode
```

### 2.2.2 Port P1 (P17 - P10)

Port P1 is an 8-bit input/output port which can be configured as an input or an output in one-bit unit under software control. Input/output mode is specified by the corresponding bit in the port P1 input/output control register (P1CR). Port P1 is configured as an input if its corresponding P1CR bit is cleared to "0", and as an output if its corresponding P1CR bit is set to "1". During reset, the P1CR is initialized to "0", which configures port P1 as an input . The P1 output latches are also initialized to "0". Data is written into the output latch regardless of P1CR contents. Therfore initial output data should be written into the output latch before setting P1CR. Port P1 is also used as an external interrupt input, a timer/counter input, and a divider output. When used as secondary function pin, the input pins should be set to the input mode, and the output pins should be set to the output mode and beforehand the output latch should be set to "1".

It is recommended that pins P11 and P12 should be used as external interrupt inputs, timer/counter input, or input ports. The interrupt latch is set at the rising or falling edge of the output when used as output ports.

Pin P10 ($\overline{INT0}$) can be configured as either an I/O port or an external interrupt input with INT0EN (bit 6 in EINTCR). During reset, pin P10 ($\overline{INT0}$) is configured as an input port P10.

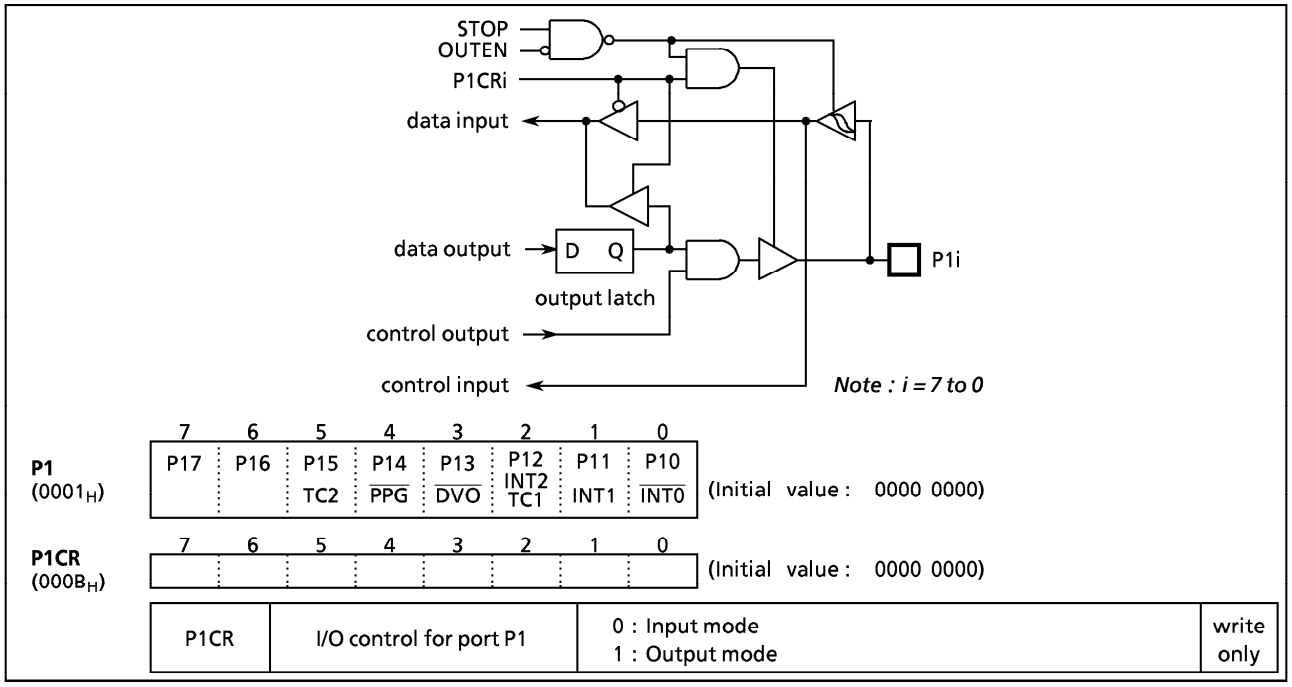



Figure 2-4. Port P1 and P1CR

Example : Sets P17, P16 and P14 as output ports, P13 and P11 as input ports, and the others as function pins. Internal output data is "1" for the P17 and P14 pins, and "0" for the P16 pin.

```
LD      (EINTCR), 01000000B     ; INT0EN←1
LD      (P1), 10111111B         ; P17←1, P14←1, P16←0
LD      (P1CR), 11010000B
```

### 2.2.3 Port P2 (P22 to P20)

Port P2 is a 3-bit input/output port. It is also used as an external interrupt input, and low-frequency crystal connection pins When used as an input port, or a secondary function pin, the output latch should be set to "1". During reset , the output latches are initialized to "1".

A low-frequency crystal (32.768 kHz) is connected to pins P21 (XTIN) and P22 (XTOUT) in the dual-clock mode. In the single-clock mode, pins P21 and P22 can be used as normal input/output ports.

It is recommended that pin P20 should be used as an external interrupt input, a STOP mode release signal input, or an input port. If used as an output port, the interrupt latch is set on the falling edge of the output pulse.

When a read instruction is executed for port P2, bits 7 to 3 read in as "1".

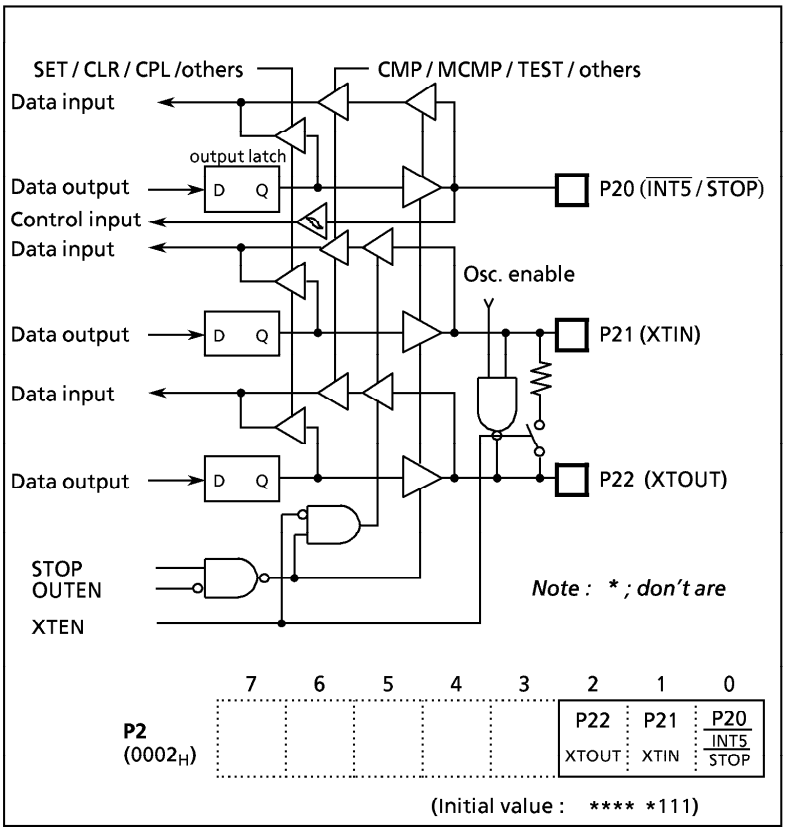



Figure 2-5. Port P2

### 2.2.4 Port P3 (P37 to P30)

Port P3 is an 8-bit input/output port. High current output is available so LEDs can be driven directly. When used as an input port, the output latch should be set to "1". The output latches are initialized to "1" during reset.

Example 1: Output the immediate data $5A_H$ to the P3 port.

```
LD      (P3), 5AH       ; P3←5AH
```

Example 2: Inverts the output of the upper 4bits (P37 - P34) of the P3 port.

```
XOR     (P3), 11110000B
                ; P37 to P34←P37 to P34
```

### 2.2.5 Port P4 (P47 to P40)

Port P4 is an 8-bit input/output port, and is also used as serial interface (SIO1, SIO2) input/output. When used as an input port or a secondary function pin, the output latch should be set to "1". The output latches are initialized to "1" during reset.

### 2.2.6 Port P5 (P54 to P50)

Port P5 is a 5-bit input/output port, and is also used as an external interrupt input, a timer/counter input/output, and a high-speed PWM output. When used as an input port or a secondary function pin, the output latch should be set to "1". The output latches are initialized to "1" during reset.

Bits 7-5 are read in as "1" when a read instruction is executed for the port P5.

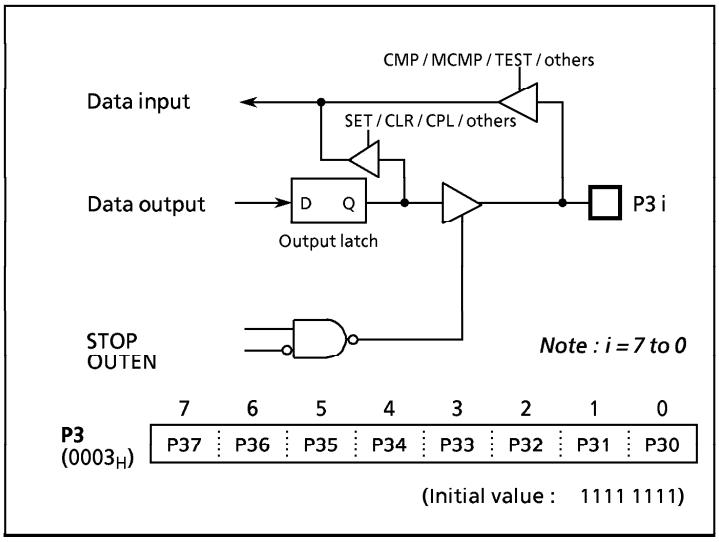



Figure 2-6. Port P3

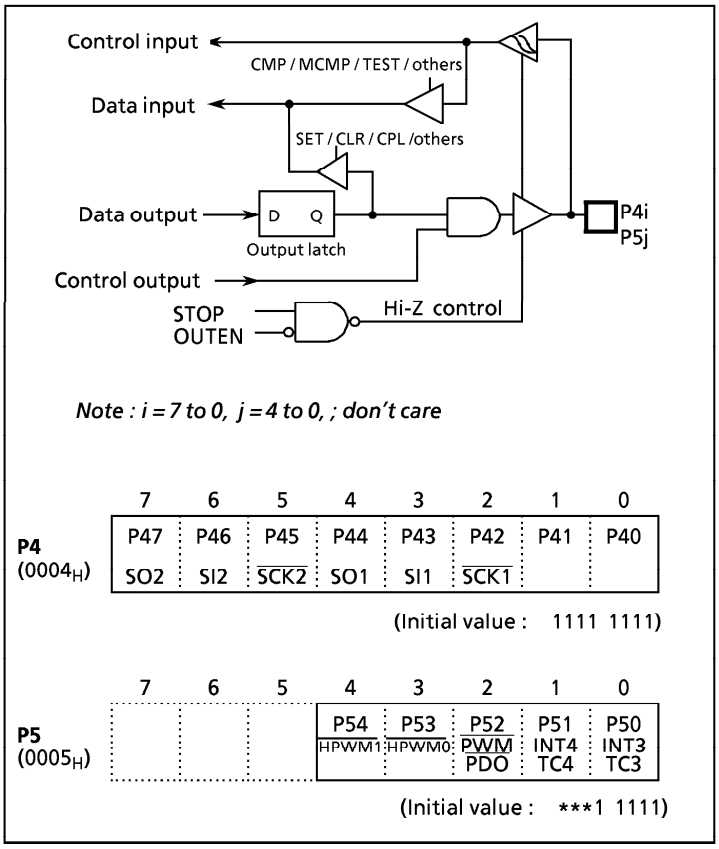



Figure 2-7. Ports P4 and P5

### 2.2.7 Port P6 (P67 to P60)

Port P6 is an 8-bit input/output port which can be configured as an input or an output in one-bit unit under software control. Input/output mode is specified by the corresponding bit in the port P6 input/output control register (P6CR).

Port P6 is also used as an analog input for the A/D converter. When used as an analog input, AINDS (bit 4 in the ADCCR) must be cleared to "0" and its corressponding P6CR bit must be set to "1". In this case, unuse pin as analog input is configured as only input port.
During reset, AINDS is initialized to "0" and all bits of P6CR are initialized to "1", which configures port P6 as analog input. The P6 output latches are initialized to "0". Data is written into the output latch regardless of the P6CR contents. Therefore initial output data should be written into the output latch before setting P6CR.

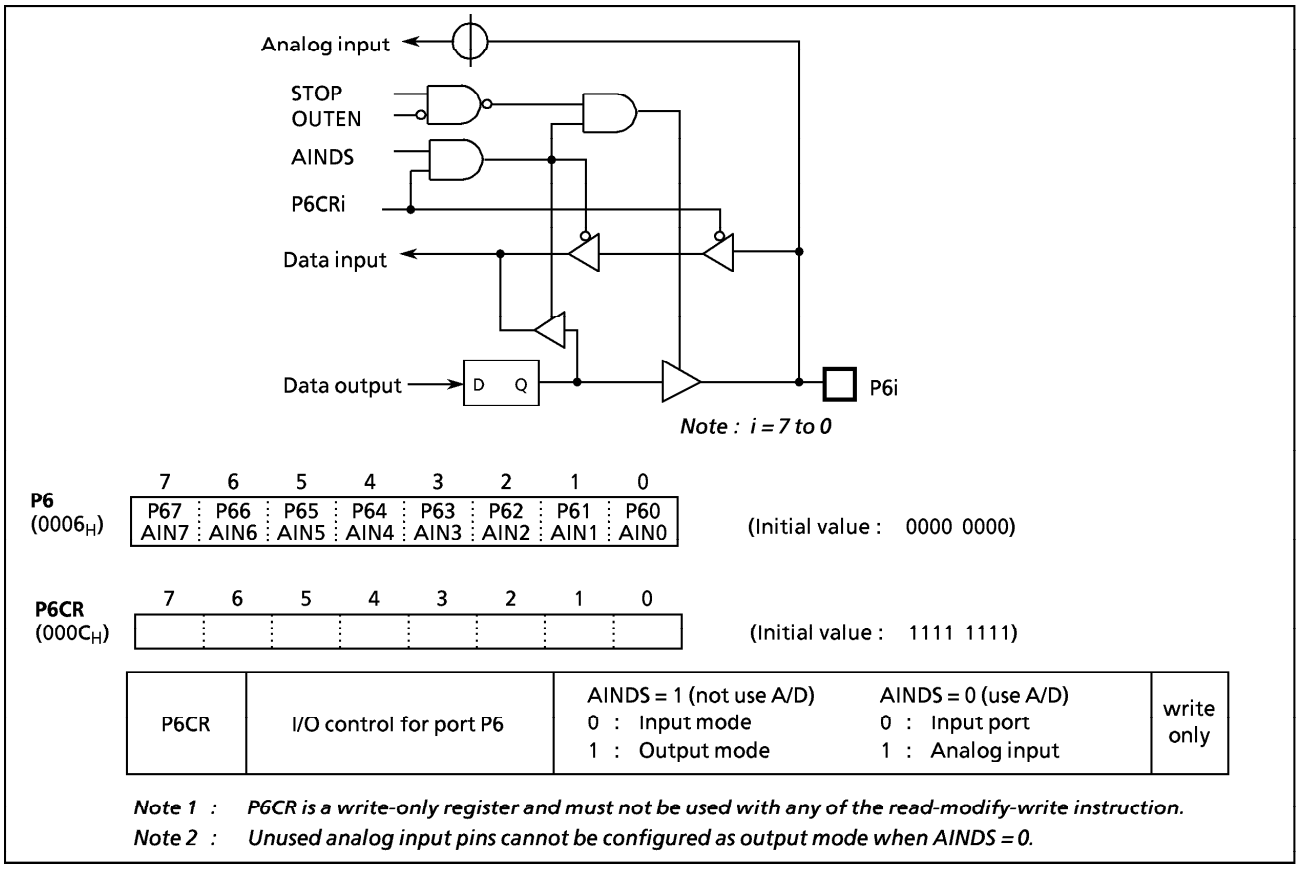


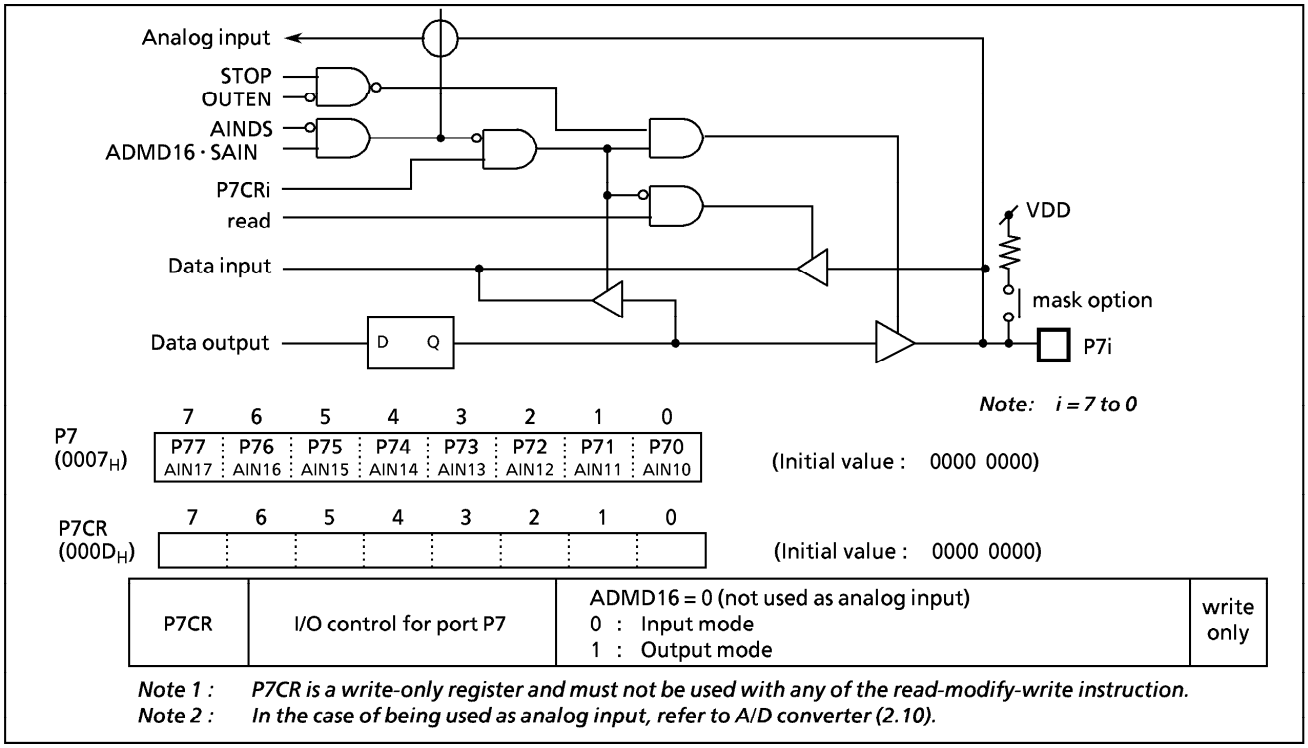

Figure 2-8. Port P6

### 2.2.8 Port P7 (P77 to P70)

Port P7 is an 8-bit input/output port which can be configured as either input or output in one-bit unit under software control. Input/output mode is specified by the corresponding bit in the port P7 input/output control register (P7CR).

Port P7 is also used as an analog input for the A/D converter. When used as an analog input, AINDS (bit 4 in the ADCCR) must be cleared to "0" and ADMD16 (bit 7 in the ADCCR) must be set to "1". In this case, unuse pin as analog input is configured as either input or output port.

During reset, AINDS and ADMD16 are initialized to "0" and all bits of P7CR are initialized to "0", which configures port P7 as input port. The P7 output latches are initialized to "0". Data is written into the output latch regardless of the P7CR contents. Therefore initial output data should be written into the output latch before setting P7CR.




*Note 1 : P7CR is a write-only register and must not be used with any of the read-modify-write instruction.*
*Note 2 : In the case of being used as analog input, refer to A/D converter (2.10).*

Figure 2.9 Port P7 and P7CR

## 2.3 Time Base Timer (TBT)

The time base timer generates time base for key scanning, dynamic displaying, etc. It also provides a time base timer interrupt (INTTBT). The time base timer is controlled by the control register (TBTCR) shown in Figure 2-11.

An INTTBT is generated on the first rising edge of source clock (the divider output of the timing generator) after the time base timer has been enabled. The divider is not cleared by the program; therefore, only the first interrupt may be generated ahead of the set interrupt period.

The interrupt frequency (TBTCK) must be selected with the time base timer disabled (both frequency selection and enabling can be performed simultaneously).

Example : Sets the time base timer frequency to fc/$2^{16}$ [Hz] and enables an INTTBT interrupt.

```
LD      (TBTCR), 00001010B
SET     (EIRL). 6
```

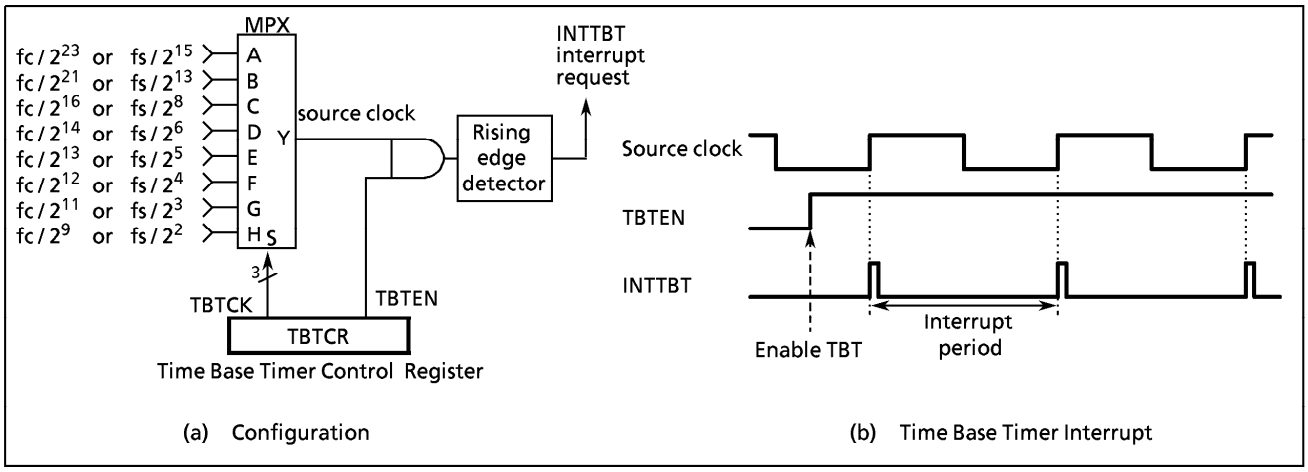
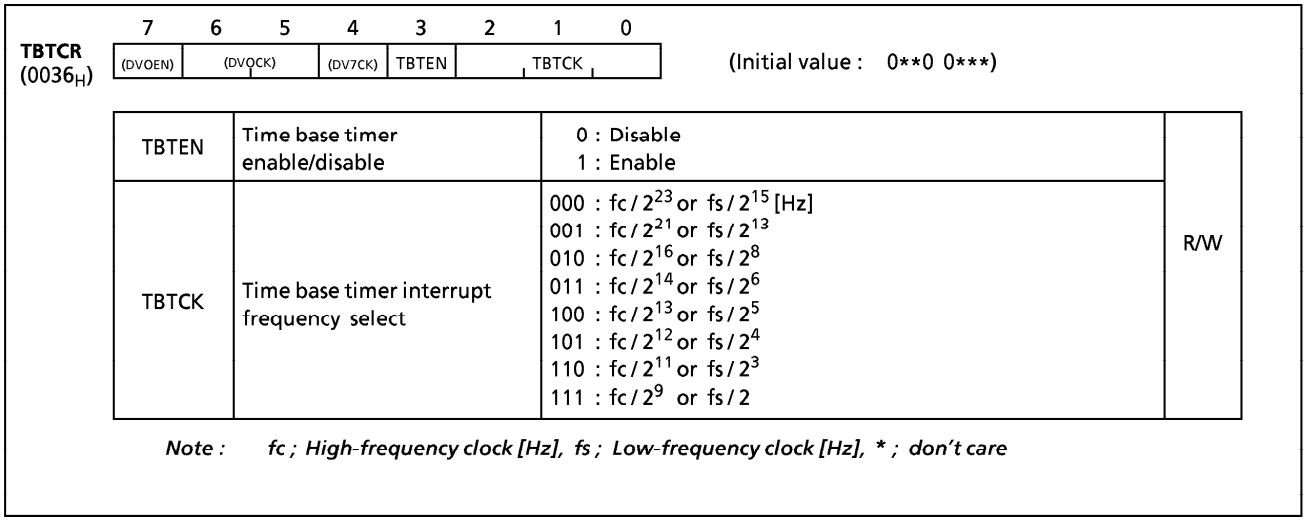



Figure 2-10. Time Base Timer




| | | | |
|---|---|---|---|
| TBTEN | Time base timer enable/disable | 0 : Disable<br>1 : Enable | R/W |
| TBTCK | Time base timer interrupt frequency select | 000 : fc / 2^23 or fs / 2^15 [Hz]<br>001 : fc / 2^21 or fs / 2^13<br>010 : fc / 2^16 or fs / 2^8<br>011 : fc / 2^14 or fs / 2^6<br>100 : fc / 2^13 or fs / 2^5<br>101 : fc / 2^12 or fs / 2^4<br>110 : fc / 2^11 or fs / 2^3<br>111 : fc / 2^9 or fs / 2 | |

*Note : fc ; High-frequency clock [Hz], fs ; Low-frequency clock [Hz], * ; don't care*

Figure 2-11. Time Base Timer and Divider Output Control Register

Table 2-1. Time Base Timer Interrupt Frequency

| TBTCK | NORMAL1/2, IDLE1/2 mode | | SLOW, SLEEP mode | Interrupt Frequency | |
|---|---|---|---|---|---|
| | DV7CK = 0 | DV7CK = 1 | | At fc = 8 MHz | At fs = 32.768 kHz |
| 000 | fc / 2^23 | fs / 2^15 | fs / 2^15 | 0.95 Hz | 1 Hz |
| 001 | fc / 2^21 | fs / 2^13 | fs / 2^13 | 3.81 | 4 |
| 010 | fc / 2^16 | fs / 2^8 | – | 122.07 | 128 |
| 011 | fc / 2^14 | fs / 2^6 | – | 488.28 | 512 |
| 100 | fc / 2^13 | fs / 2^5 | – | 976.56 | 1024 |
| 101 | fc / 2^12 | fs / 2^4 | – | 1953.12 | 2048 |
| 110 | fc / 2^11 | fs / 2^3 | – | 3906.25 | 4096 |
| 111 | fc / 2^9 | fs / 2 | – | 15625 | 16384 |

## 2.4 Divider Output ($\overline{DVO}$)

A 50 % duty pulse can be output using the divider output circuit, which is useful for piezo-electric buzzer drive. Divider output is from pin P13 ($\overline{DVO}$). The P13 output latch should be set to "1" and then the P13 should be configured as an output mode.

Divider output circuit is controlled by the control register (TBTCR) shown in Figure 2-12.

TBTCR ($0036_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DVOEN | DVOCK | | (DV7CK) | (TBTEN) | (TBTCK) | | |

(Initial value : 0**0 0***)

| | | | |
|---|---|---|---|
| DVOEN | Divider output enable/disable | 0 : Disable<br>1 : Enable | R/W |
| DVOCK | Divider output ($\overline{DVO}$) frequency selection | 00 : $fc/2^{13}$ or $fs/2^5$ [Hz]<br>01 : $fc/2^{12}$ or $fs/2^4$<br>10 : $fc/2^{11}$ or $fs/2^3$<br>11 : $fc/2^{10}$ or $fs/2^2$ | |

*Note : fc ; High-frequency clock [Hz], fs ; Low-frequency clock [Hz], * ; don't care*

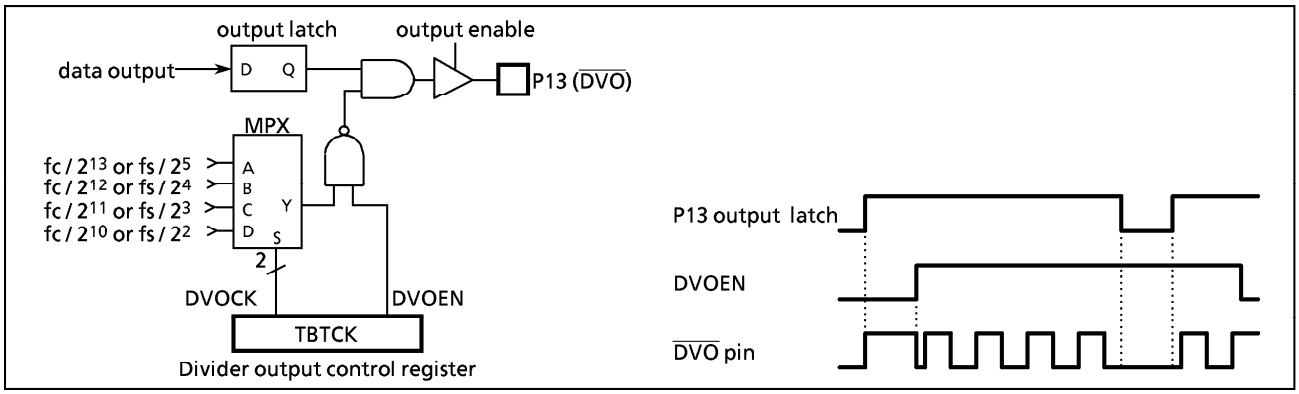
Figure 2-12. Divider Output Control Register

Example : 1 kHz pulse output (at fc = 8 MHz)

```
SET     (P1).3              ; P13 output latch ←1
LD      (P1CR), 00001000B   ; Configures P13 as an output mode
LD      (TBTCR), 10000000B  ; DVOEN←1, DVOCK←00
```

Table 2-2. Frequency of Divider Output

| DVOCK | Frequency of Divider Output | At fc = 8 MHz | At fs = 32.768 kHz |
|---|---|---|---|
| 00 | $fc/2^{13}$ or $fs/2^5$ | 0.976 [kHz] | 1.024 [kHz] |
| 01 | $fc/2^{12}$ $fs/2^4$ | 1.953 | 2.048 |
| 10 | $fc/2^{11}$ $fs/2^3$ | 3.906 | 4.096 |
| 11 | $fc/2^{10}$ $fs/2^2$ | 7.812 | 8.192 |




(a) Configuration (b) Timing Chart

Figure 2-13. Divider Output

## 2.5 16-bit Timer/Counter 1 (TC1)

### 2.5.1 Configuration

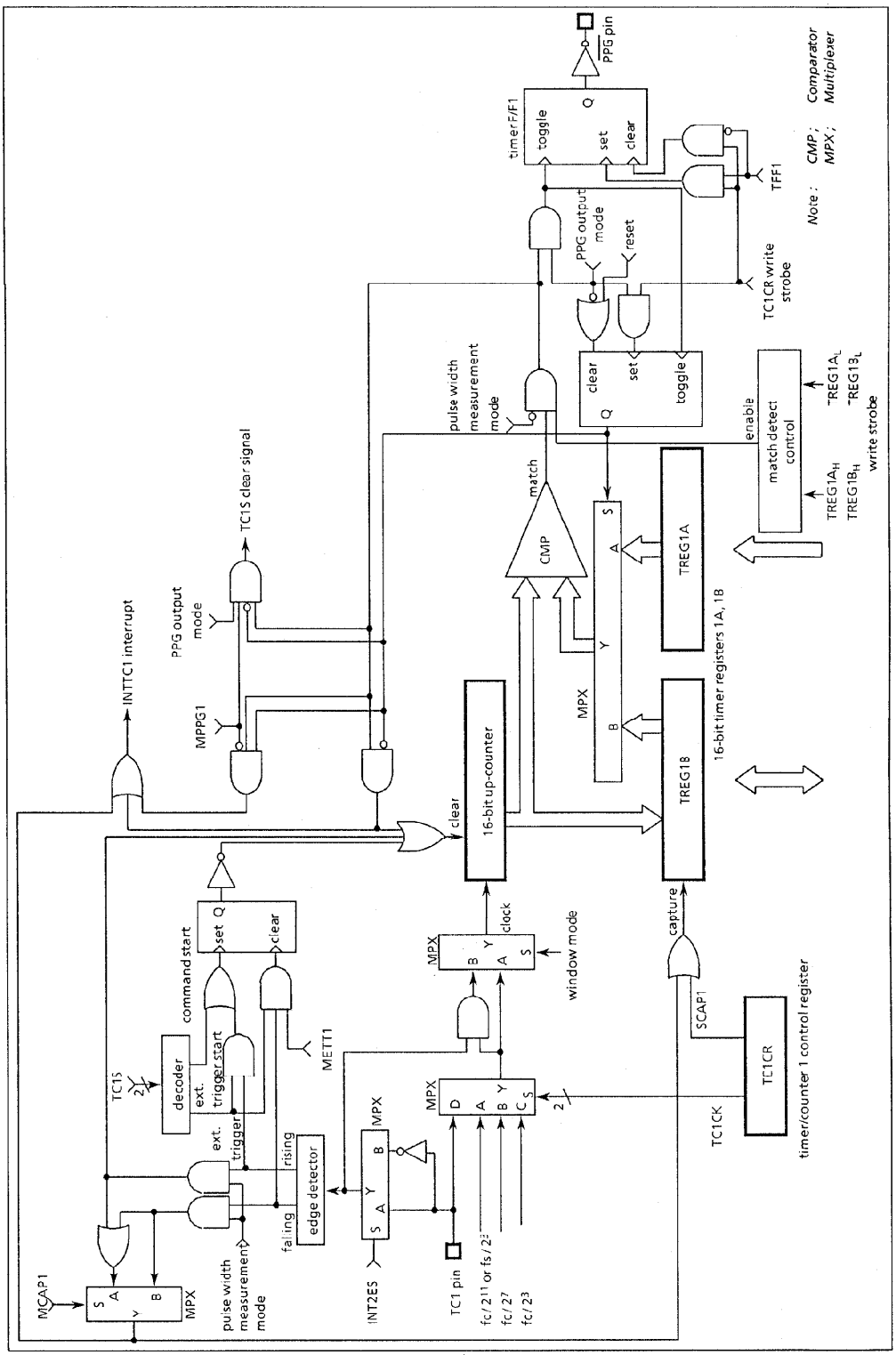
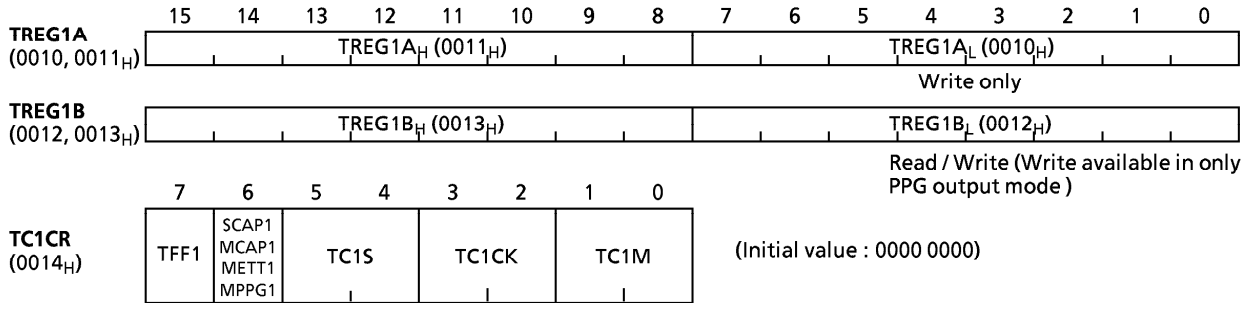



Figure 2-14. Timer/Counter 1

## 2.5.2 Control

The timer/counter 1 is controlled by a timer/counter 1 control register (TC1CR) and two 16-bit timer registers (TREG1A and TREG1B). Reset does not affect TREG1A and TREG1B.




| | | | |
|---|---|---|---|
| TC1M | TC1 mode select | 00 : timer / external trigger timer / event counter mode<br>01 : window mode<br>10 : pulse width measurement mode<br>11 : PPG output mode | Write only |
| TC1CK | TC1 source clock select | 00 : internal clock fc/$2^{11}$ or fs/$2^2$ [Hz]<br>01 : internal clock fc/$2^7$<br>10 : internal clock fc/$2^3$<br>11 : external clock (TC1 pin input) | |
| TC1S | TC1 start control | 00 : stop & counter clear<br>01 : command start<br>10 : reserved<br>11 : external trigger start | |
| SCAP1 | software capture control | 0 : – 1 : software capture trigger (Note 3) | |
| MCAP1 | pulse width measurement control | 1 : double edge capture 1 : single edge capture | |
| METT1 | external trigger timer control | 0 : trigger start 1 : trigger start & stop | |
| MPPG1 | PPG output control | 0 : continuous pulse 1 : single pulse | |
| TFF1 | timer F/F1 control for PPG output mode | 0 : clear 1 : set | |

*Note 1 : fc ; High-frequency clock [Hz], fs ; Low-frequency clock [Hz]*

*Note 2 : Writing to the low-byte of the timer registers (TREG1A$_L$, TREG1B$_L$), the comparison is inhibited until the high-byte (TREG1A$_H$, TREG1B$_H$) is written.*

*Note 3 : Set the mode, source clock, edge (INT2ES), PPG control and timer F/F control when TC1 stops (TC1S = 00).*

*Note 4 : Software capture can be used in only timer and event counter modes.*

*Note 5 : Values to be loaded to timer registers must satisfy the following condition.*
*TREG1A > TREG1B > 0 (PPG output mode) ; TREG1A > 0 (others)*

*Note 6 : Always write "0" to TFF1 except the PPG output mode.*

*Note 7 : TC1CR is a write-only register, which cannot be accessed by any read-modify-write instruction such as bit operate, etc.*

*Note 8 : TREG1B cannot be written after setting to PPG output mode.*

Figure 2-15. Timer Registers and TC1 Control Register

## 2.5.3 Function

Timer/counter 1 has six operating modes: timer, external trigger timer, event counter, window, pulse width measurement, programmable pulse generator output mode.

(1) **Timer** Mode

In this mode, counting up is performed using the internal clock. The contents of TREG1A are compared with the contents of up-counter. If a match is found, an INTTC1 interrupt is generated, and the counter is cleared to"0". Counting up resumes after the counteriscleared. The current contents of up-counter can be transfered to TREG1B by setting SCAP1 (bit 6 in TC1CR) to "1" (software capture function). SCAP1 is automatically cleared to "0" after capaturing.

Table 2-3. Timer/Counter 1 Source Clock (Internal Clock)

| Source clock | | | Resolution | | Maximum time setting | |
|---|---|---|---|---|---|---|
| NORMAL1/2, IDLE1/2 modes | | SLOW, SLEEP modes | | | | |
| DV7CK = 0 | DV7CK = 1 | | At fc = 8 MHz | At fs = 32.768 kHz | At fc = 8 MHz | At fs = 32.768 kHz |
| $fc/2^3$ [Hz] | $fc/2^3$ [Hz] | – | 1 μs | – | 65.5 ms | – |
| $fc/2^7$ | $fc/2^7$ | – | 16 μs | – | 1.0 s | – |
| $fc/2^{11}$ | $fs/2^3$ | $fs/2^3$ [Hz] | 256 μs | 244.14 μs | 16.8 s | 16.0 s |

Example 1 : Sets the timer mode with source clock $fs/2^3$[Hz] and generates an interrupt 1 s. later (at fs = 32.768 kHz).

```
LD      (TC1CR), 00000000B      ; Sets the TC1 mode and source clock
LDW     (TREG1A), 1000H         ; Sets the timer register (1 s ÷ 2^3 / fs = 1000H)
SET     (EIRL).EF4              ; enable INTTC1
EI
LD      (TC1CR), 00010000B      ; Starts TC1
```

Example 2 : Software capture

```
LD      (TC1CR), 01010000B      ; SCAP1←1 (Captures)
LD      WA, (TREG1B)            ; Reads captured value
```

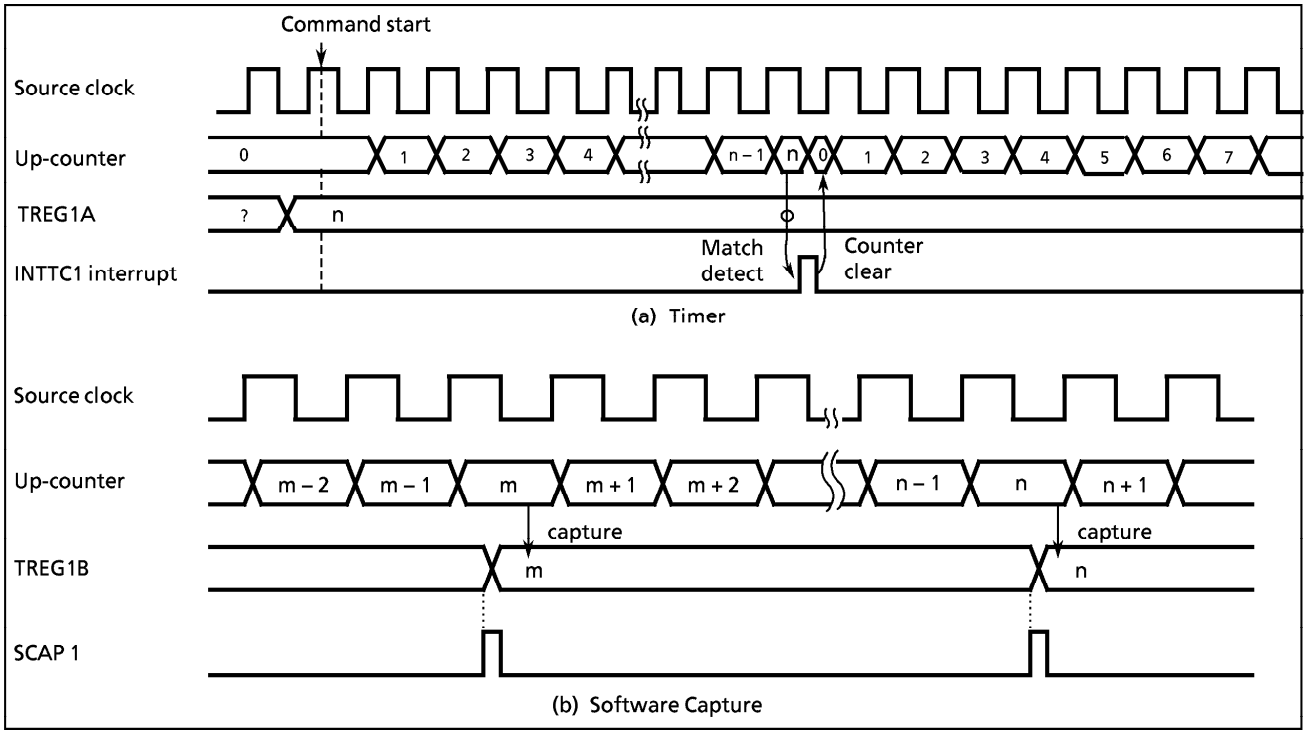



Figure 2-16. Timer Mode Timing Chart

(2) **External Trigger Timer** mode

In this mode, counting up is started by an external trigger. This trigger is the edge of the TC1 pin input. Either the rising or falling edge can be selected with INT2ES. Edge selection is the same as for the external interrupt input INT2 pin. Source clock is used an internal clock selected with TC1CK. The contents of TREG1A is compared with the contents of up-counter. If a match is found, an INTTC1 interrupt is generated, and the counter is cleared to"0" and halted. The counter is restarted by the selected edge of the TC1 pin input.

The TC1 pin input has the same noise rejection as the INT2 pin; therefore, pulses of 7/fc [s] or less are rejected as noise. A pulse width of 24/fc [s] or more is required for edge detection in NORMAL1/2 or IDLE1/2 mode. The noise rejection circuit is turned off in SLOW and SLEEP modes. But, a pulse width of 4/fs [s] or more is required.

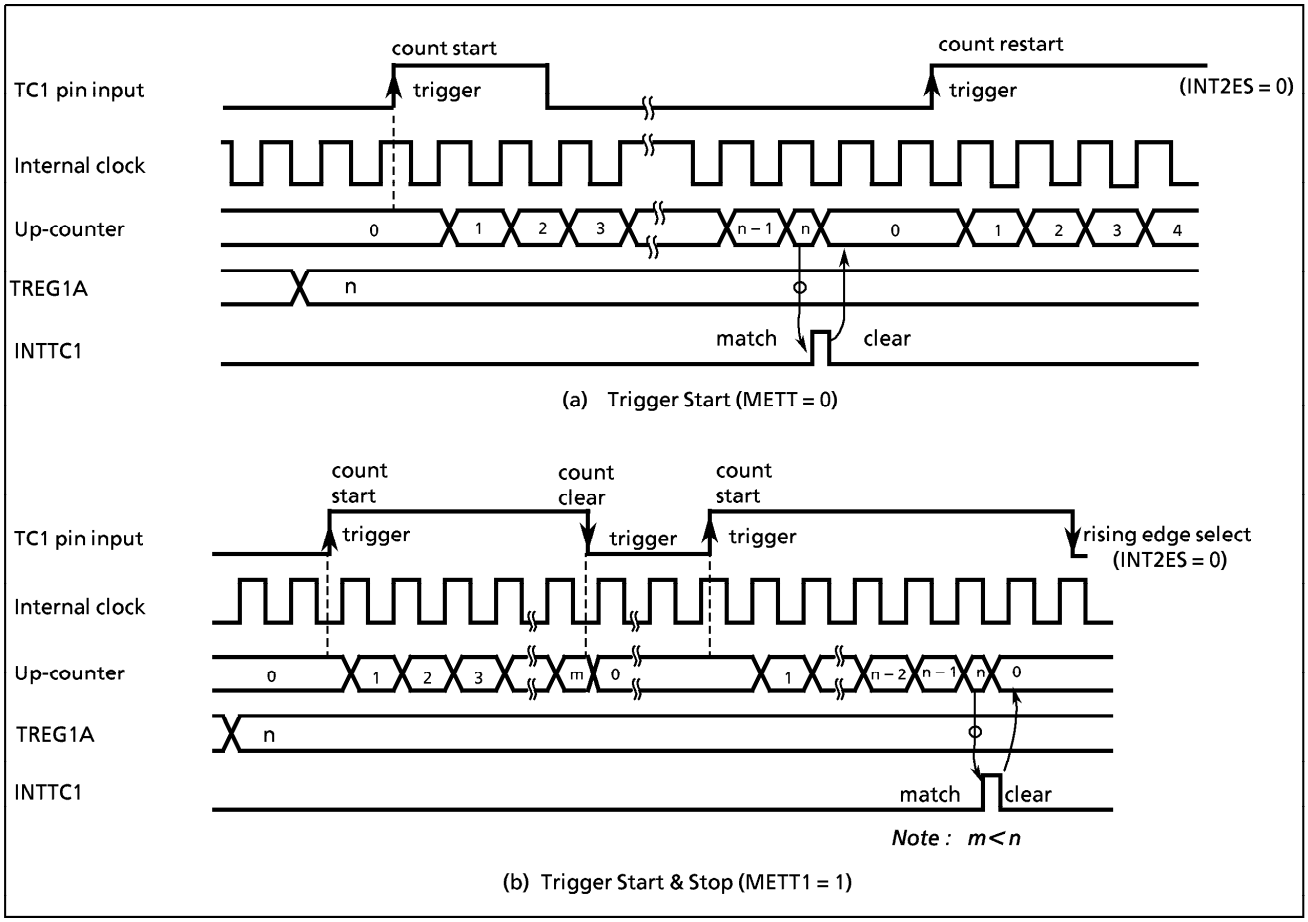



Figure 2-17. External Trigger Timer Mode Timing Chart

(3) **Event Counter** Mode

In this mode, events are counted on the edge of the TC1 pin input. Either the rising or falling edge can be selected with INT2ES in EINTCR. The contents of TREG1A are compared with the contents of up-counter. If a match is found, an INTTC1 interrupt is generated, and the counter is cleared. The maximum applied frequency is fc/$2^4$ [Hz] in NORMAL1/2 or IDLE1/2 mode and fs/$2^4$ [Hz] in SLOW or SLEEP mode.

Setting SCAP1 to "1" transferres the current contents of up-counter to TREG1B (software capture function). SCAP is automatically cleared after capturing.

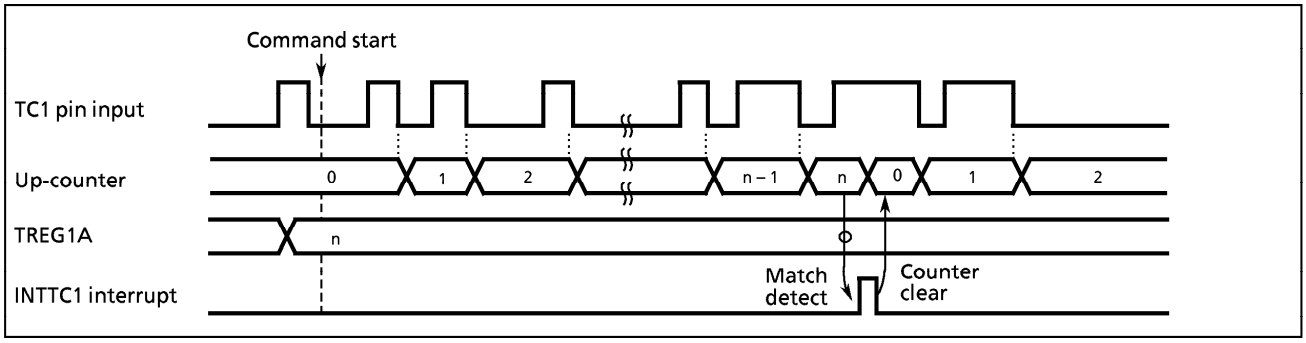



Figure 2-18. Event Counter Mode Timing Chart (INT2ES = 1)

(4) **Window** mode

Counting up is performed on the rising edge of the pulse that is the logical AND-ed product of the TC1 pin input (window pulse) and an internal clock. The contents of TREG1A are compared with the contents of up-counter. If a match is found, an INTTC1 interrupt is generated, and the counter is cleared. Positive or negative logic for the TC1 pin input can be selected with INT2ES. Setting SCAP1 to "1" transferes the current contents of up-counter to TREG1B. It is necessary that the maximum applied frequency (TC1 input) be such that the counter value can be analyzed by the program. That is, the frequency must be considerably slower than the selected internal clock.

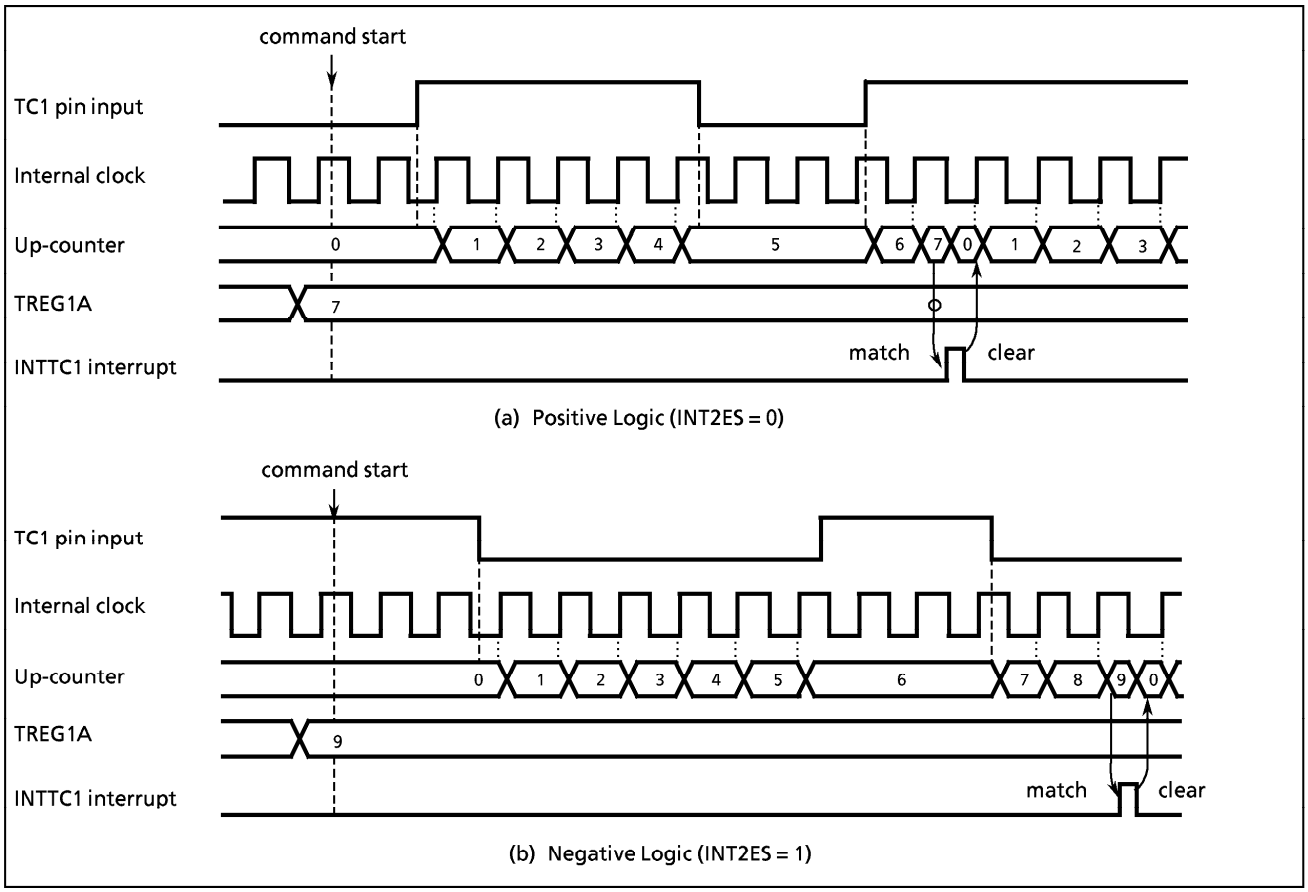



Figure 2-19. Window Mode Timing Chart

(5) **Pulse width measurement** mode

Counting is started by the external trigger (set to external trigger start by TC1S). The trigger can be selected either the rising or falling edge of the TC1 pin input. The source clock is used an internal clock. On the next falling (rising) edge, the counter contents are transferred to TREG1B and an INTTC1 interrupt is generated. The counter is cleared when the single edge capture mode is set. When double edge capture is set, the counter continues and, at the next rising (falling) edge, the counter contents are again transferred to TREG1B. If a falling (rising) edge capture value is required, it is necessary to read out TREG1B contents until a rising (falling) edge is detected. Falling or rising edge is selected with INT2ES, and single edge or double edge is selected with MCAP1 ( bit 6 in TC1CR).

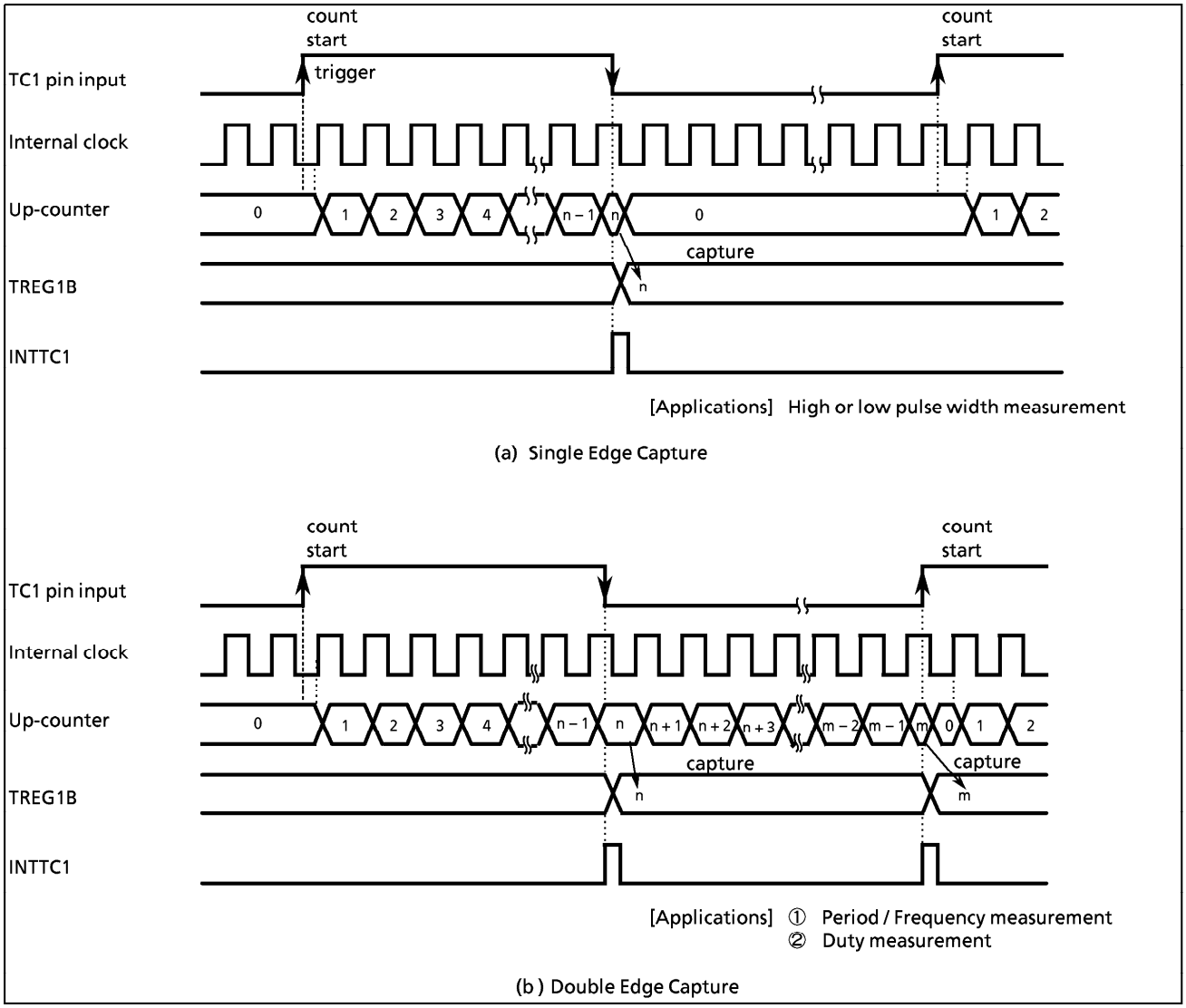



Figure 2-20. Pulse Width Measurement Mode Timing Chart

Example : Duty measurement (Resolution $fc/2^7$ [Hz])

```
            CLR     (INTTC1SW). 0           ;  INTTC1 service switch initial setting
            LD      (EINTCR), 00000000B     ;  Sets the rise edge at the INT2 edge
            LD      (TC1CR), 00000110B      ;  Sets the TC1 mode and source clock
            SET     (EIRL). 4               ;  Enables INTTC1
            EI
            LD      (TC1CR), 00110110B      ;  Starts TC1 with an external trigger
             :
PINTTC1 :   CPL     (INTTC1SW). 0           ;  Complements INTTC1 service switch
            JRS     F, SINTTC1
            LD      (HPULSE), (TREG1BL)     ;  Reads TREG1B
            LD      (HPULSE + 1), (TREG1BH)
            RETI
SINTTC1 :   LD      (WIDTH), (TREG1BL)      ;  Reads TREG1B (Period)
            LD      (WIDTH + 1), (TREG1BH)
             :
            RETI
             :
VINTTC1 :   DW      PINTTC1
```

(6) **Programmable Pulse Generate** (PPG) **output** mode

Counting is started by an edge of the TC1 pin input (either the rising or falling edge can be selected) or by a command. The source clock is used an internal clock. First, the contents of TREG1B are compared with the contents of the up-counter. If a match is found, timer F/F1 output is toggled. Next, timer F/F1 is again toggled and the counter is cleared by matching with TREG1A. An INTTC1 interrupt is generated at this time. Timer F/F output is connected to the P14 ($\overline{PPG}$) pin. In the case of $\overline{PPG}$ output, set the P14 output latch to "1" and configure as an output with $P1CR_4$. Timer F/F1 is cleared to "0" during reset. The timer F/F1 value can also be set by program and either a positive or negative logic pulse output is available. Also, writing to the TREG1B is not possible unless the timer / counter 1 is set to the PPG output mode with TC1M.

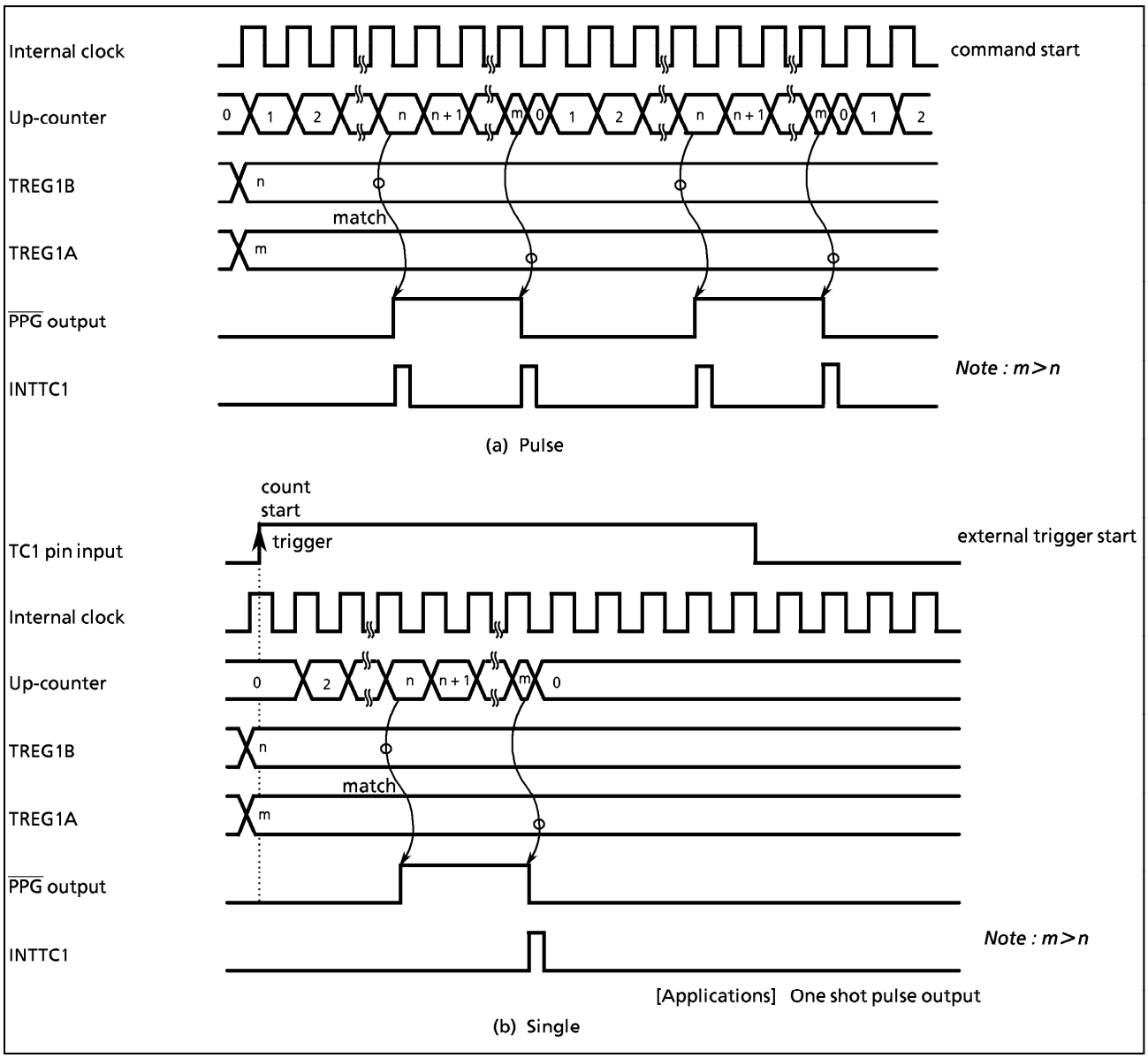



Figure 2-21. PPG Output Mode Timing Chart

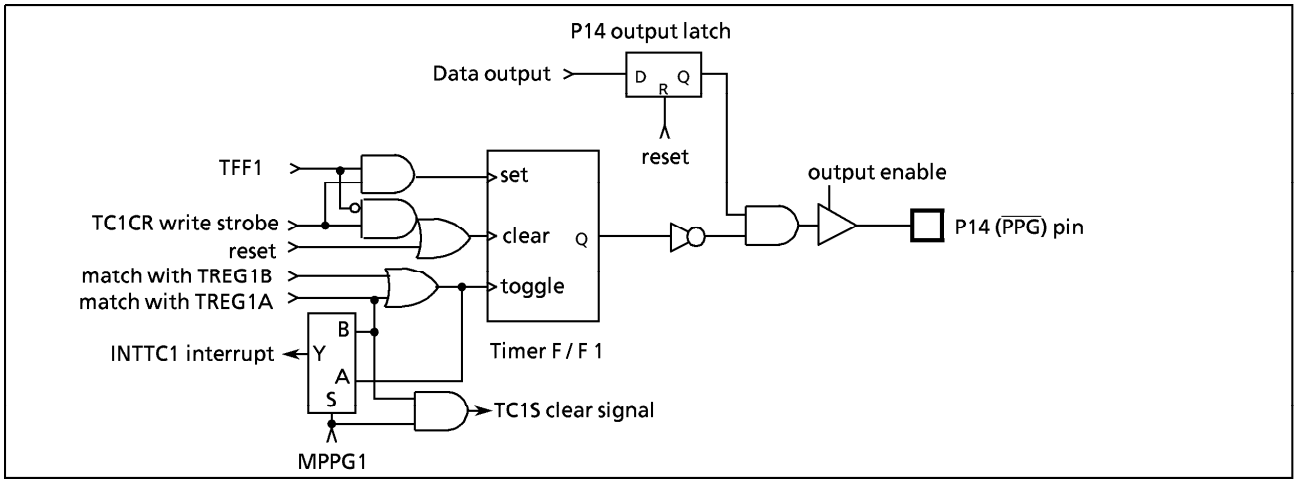



Figure 2-22. PPG Output

## 2.6 16-bit Timer/Counter 2 (TC2)

### 2.6.1 Configuration

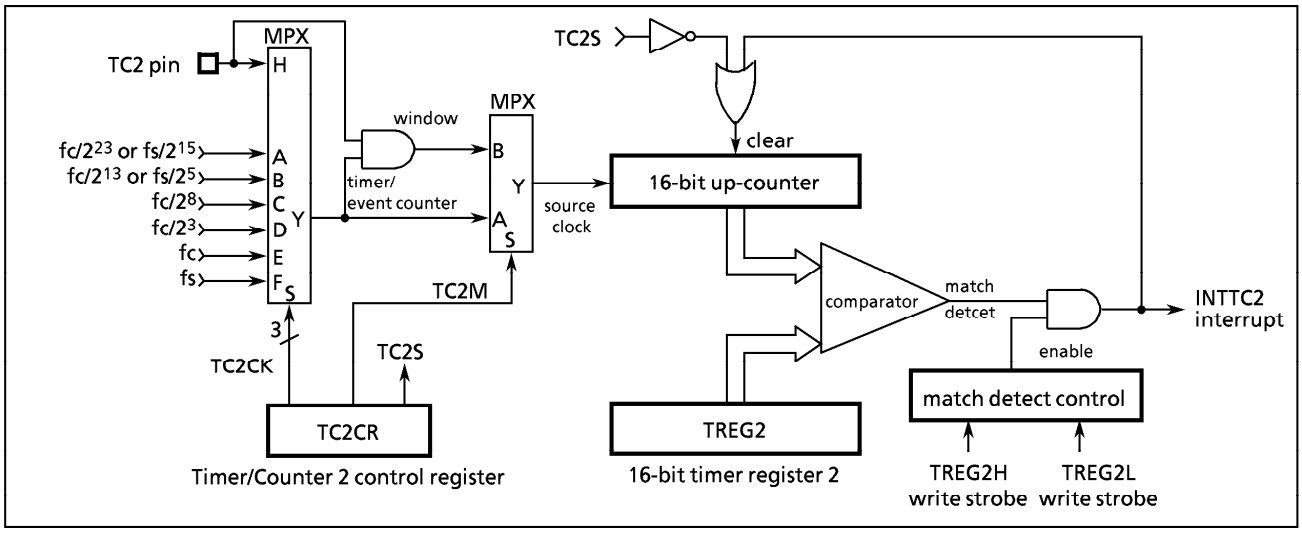



Figure 2-23. Timer/Counter 2 (TC2)

## 2.6.2 Control

The timer/counter 2 is controlled by a timer/counter 2 control register (TC2CR) and a 16-bit timer register 2 (TREG2). Reset does not affect TREG2.

**TREG2** (0016, 0017H)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TREG2H (0017H) | | | | | | | | TREG2L (0016H) | | | | | | | |

write only

**TC2CR** (0015H)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | TC2S | TC2CK | | | | TC2M |

(Initial value : **00 00*0)

| | | | |
|---|---|---|---|
| TC2M | Timer/counter 2 operating mode select | 0 : Timer/Event counter mode<br>1 : Window mode | write only |
| TC2CK | Timer/counter 2 source clock select | 000 : Internal clock fc / $2^{23}$ or fs / $2^{15}$ [Hz]<br>001 : 〃 fc / $2^{13}$ or fs / $2^{5}$<br>010 : 〃 fc / $2^{8}$<br>011 : 〃 fc / $2^{3}$<br>100 : 〃 fc (Note 5)<br>101 : 〃 fs<br>110 : Reserved<br>111 : External clock (TC2 pin input) | |
| TC2S | Timer/counter 2 start control | 0 : Stop and counter clear<br>1 : Start | |

*Note 1 : fc; High-frequency clock [Hz], fs; Low-frequency clock [Hz], *; don't care*

*Note 2 : When writing to the low-byte of timer register 2 ($TREG2_L$), the comparison is inhibited until the high-byte ($TREG2_H$) is written.*
*After writing to the high-byte, any match during 1 machine cycle (instruction execution cycle) is ignored.*

*Note 3 : Set the mode and source clock when timer/counter stops (TC2S = 0).*

*Note 4 : Values to be loaded to the timer register must satisfy the following condition.*
*$TREG2 > 0$ ($TREG2_{15\ to\ 11} > 0$ when warm-up).*

*Note 5 : "fc" can be selected as the source clock only in the timer mode during the SLOW mode.*

*Note 6: TC2CR and TREG2 are write-only registers and must not be used with any of the read-modify-write instructions.*

Figure 2-24. Timer Register 2 and TC2 Control Register

## 2.6.3 Function

The timer/counter 2 has three operating modes: timer, event counter and window modes. Also timer/counter 2 is used for warm-up when switching from SLOW mode to NORMAL2 mode.

(1) **Timer** Mode

In this mode, the internal clock is used for counting up. The contents of TREG2 are compared with the contents of up-counter. If a match is found, a timer/ counter 2 interrupt (INTTC2) is generated, and the counter is cleared. Counting up is resumed after the counter is cleared.

Also, when fc is selected as the source clock during SLOW mode, the lower 11 bits of TREG2 are ignored and an INTTC2 interrupt is generated by matching the upper 5 bits. Thus, in this case, only the $TREG2_H$ setting is necessary.

Table 2-4. Source Clock (Internal Clock) for Timer/Counter 2

| Source clock | | | | Resolution | | Maximum time setting | |
|---|---|---|---|---|---|---|---|
| NORMAL1/2, IDLE1/2 mode | | SLOW mode | SLEEP mode | | | | |
| DV7CK = 0 | DV7CK = 1 | | | At fc = 8 MHz | At fs = 32.768 kHz | At fc = 8 MHz | At fs = 32.768 kHz |
| $fc/2^{23}$ [Hz] | $fs/2^{15}$ [Hz] | $fs/2^{15}$ [Hz] | $fs/2^{15}$ [Hz] | 1.05 s | 1 s | 19.1 h | 18.2 h |
| $fc/2^{13}$ | $fs/2^{5}$ | $fs/2^{5}$ | $fs/2^{5}$ | 1.02 ms | 1 ms | 1.1 min | 1 min |
| $fc/2^{8}$ | $fc/2^{8}$ | – | – | 32 μs | – | 2.1 s | – |
| $fc/2^{3}$ | $fc/2^{3}$ | – | – | 1 μs | – | 65.5 ms | – |
| – | – | fc (Note) | – | 125 ns | – | 7.9 ms | – |
| fs | fs | – | – | – | 30.5 μs | – | 2 s |

*Note : "fc" can be used only in the timer mode.*

Example : Sets the timer mode with source clock $fc/2^3$ [Hz] and generates an interrupt every 25 ms (at fc = 8 MHz).

```
LD      (TC2CR), 00001100B      ; Sets the TC2 mode and source clock
LDW     (TREG2), 61A8H          ; Sets TREG2 (25ms ÷ 2^3/fc = 61A8H)
SET     (EIRH).EF14             ; Enable INTTC2
EI
LD      (TC2CR), 00101100B      ; Starts TC2
```

(2) **Event Counter** Mode

In this mode, events are counted on the rising edge of the TC2 pin input. The contents of TREG2 are compared with the contents of the up-counter. If a match is found, an INTTC2 interrupt is generated, and the counter is cleared. The maximum frequency applied to the TC2 pin is $fc/2^4$ [Hz] in NORMAL1/2 or IDLE1/2 mode, and $fs/2^4$ [Hz] in SLOW or SLEEP mode.

Example : Sets the event counter mode and generates an INTT2 interrupt 640 counts later.

```
LD      (TC2CR), 00011100B      ; Sets the TC2 mode
LDW     (TREG2), 640            ; Sets TREG2
SET     (EIRH).EF14             ; Enable INTTC2
EI
LD      (TC2CR), 00111100B      ; Starts TC2
```

(3) **Window** Mode

In this mode, counting up is performed on the rising edge of the pulse that is the logical AND-ed product of the TC2 pin input (window pulse) and an internal clock. The internal clock is selected with TC2CK. The contents of TREG2 are compared with the contents of up-counter. If a match is found, an INTTC2 interrupt is generated, and the up-counter is cleared to "0". It is necessary that the maximum applied frequency (TC2 input) be such that the counter value can be analyzed by the program. That is, the frequency must be considerably slower than the selected internal clock.

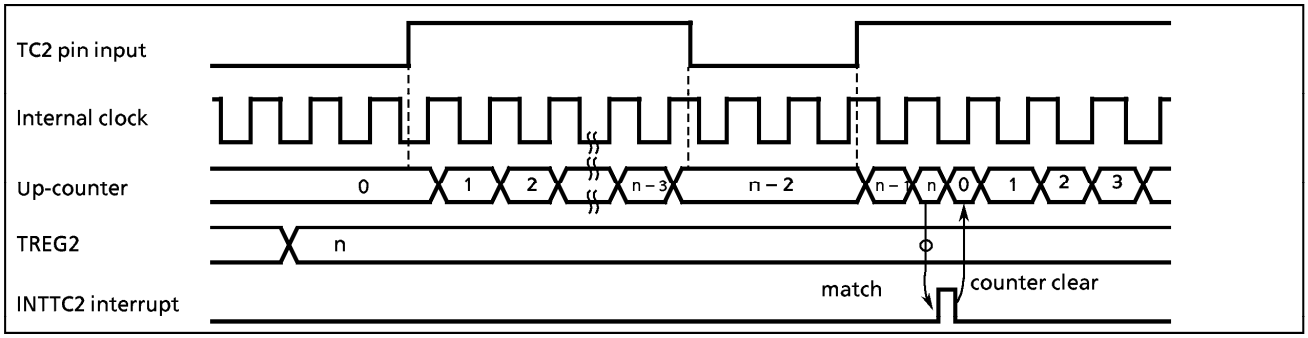



Figure 2-25. Window Mode Timing Chart

## 2.7 8-Bit Timer/Counter 3 (TC3)

### 2.7.1 Configuration

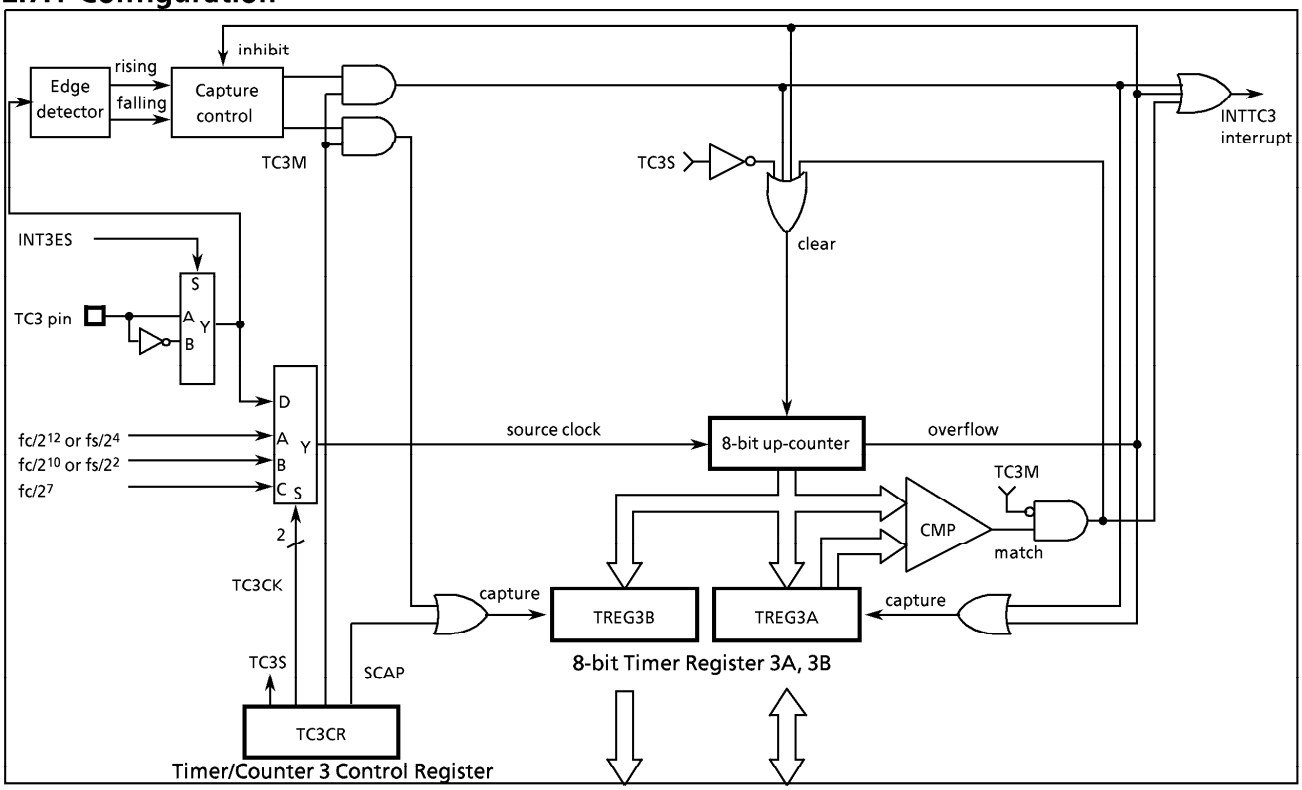
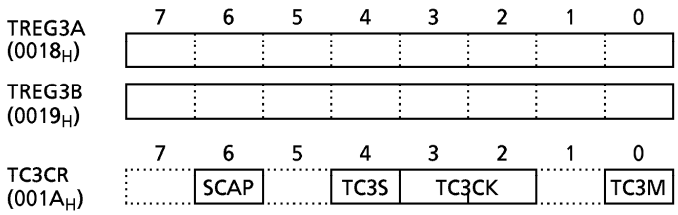



Figure 2-26. Timer/Counter 3

### 2.7.2 Control




| | | | |
|---|---|---|---|
| TC3M | Timer/counter 3 operation mode set | 0 : Timer/event counter<br>1 : Capture | Write only |
| TC3CK | Timer/counter 3 source clock select | 00 : Internal clock fc / $2^{12}$ or fs / $2^4$ [Hz]<br>01 : Internal clock fc / $2^{10}$ or fs / $2^2$<br>10 : Internal clock fc / $2^7$<br>11 : External clock (TC3 pin input) | |
| TC3S | Timer/counter 3 start select | 0 : Stop & clear<br>1 : Start | |
| SCAP | Software capture control | 0 : –<br>1 : Software capture | |

*Note 1 : fc ; High-frequency clock [Hz] fs ; Low-frequency clock [Hz] * ; don't care*
*Note 2 : Set the mode, the source clock and the edge selection (INT3ES) when the TC3 stops (TC3S = 0).*
*Note 3 : Values to be loaded into timer register 3A must satisfy the following condition.*
*TREG3A > 0 (in the timer/event counter mode)*
*Note 4 : TC3CR is a write-only register and must not be used with any of read-modify-write instructions.*

Figure 2-27. Timer Register 3A/3B and TC3 Control Register

The timer/counter 3 is controlled by a timer/counter 3 control register (TC3CR) and two 8-bit timer registers (TREG3A and TREG3B). Reset does not affect these timer registers.

### 2.7.3 Function

The timer/counter 3 has three operating modes : timer, event counter, and capture mode.

(1) **Timer** Mode

In this mode, the internal clock is used for counting up. The contents of TREG3A are compared with the contents of up-counter. If a match is found, a timer/counter 3 interrupt (INTTC3) is generated, and the up-counter is cleared. Counting up resumes after the up-counter is cleared. The current contents of up-counter are loaded into TREG3B by setting SCAP (bit 6 in TC3CR) to "1". SCAP is automatically cleared after capturing.

Table 2-5. Source Clock (Internal Clock) for Timer Counter 3

| Source clock | | Resolution | | Maximum setting time | |
|---|---|---|---|---|---|
| NORMAL1/2, IDLE1/2 mode | SLOW, SLEEP mode | fc = 8 MHz | fs = 32.768 kHz | fc = 8 MHz | fs = 32.768 kHz |
| $fc/2^{12}$ or $fs/2^4$ [Hz] | $fs/2^4$ [Hz] | 512 μs | 488.28 μs | 131.1 ms | 125 ms |
| $fc/2^{10}$ or $fs/2^2$ | – | 128 μs | 122.07 μs | 32.8 ms | 31.25 ms |
| $fc/2^7$ | – | 16 μs | – | 4.1 ms | – |

(2) **Event Counter** Mode

In this mode, the TC3 pin input pulses are used for counting up. Either the rising or falling edge can be selected with INT3ES (bit 3 in EINTCR). The contents of TREG3A are compared with the contents of the up-counter. If a match is found, an INTTC3 interrupt is generated and the counter is cleared. The maximum applied frequency is $fc/2^4$ [Hz] in the NORMAL1/2 or IDLE1/2 mode, and $fs/2^4$ [Hz] in SLOW or SLEEP mode. Two or more machine cycles are required for both the "H" and "L" levels of the pulse width.

The current contents of up-counter are loaded into TREG3B by setting SCAP (bit 6 in TC3CR) to "1". SCAP is automatically cleared after capturing.

Example : Generates an interrupt every 0.5 s, inputing 50Hz pulses to the TC3 pin.

```
LD (TC3CR), 00001100B  ;  Sets TC3 mode and source clock
LD (TREG3A), 19H       ;  0.5 s ÷ 1 / 50 = 25 = 19H
LD (TC3CR), 00011100B  ;  Start TC3
```

(3) **Capture** Mode

The pulse width, period and duty of the TC3 pin input are measured in this mode, which can be used in decoding the remote control signals, etc. The counter is free running by the internal clock. On the rising (falling) edge of the TC3 pin input, the current contents of counter is loaded into TREG3A, then the up-counter is cleared and an INTTC3 interrupt is generated. On the falling (rising) edge of the TC3 pin input, the current contents of the counter is loaded into the TREG3B. In this case, counting continues. At the next rising (falling) edge of the TC3 pin input, the current contents of counter are loaded into TREG3A, then the counter is cleared again and an interrupt is generated. If the counter overflows before the edge is detected, $FF_H$ is set to the TREG3A and an overflow interrupt (INTTC3) is generated. During interrupt processing, it can be determined whether or not there is an overflow by checking whether or not the TREG3A value is $FF_H$. Also, after an interrupt (capture to TREG3A, or overflow detection) is generated, capture and overflow detection are halted until TREG3A has been read out; however, the counter continues.

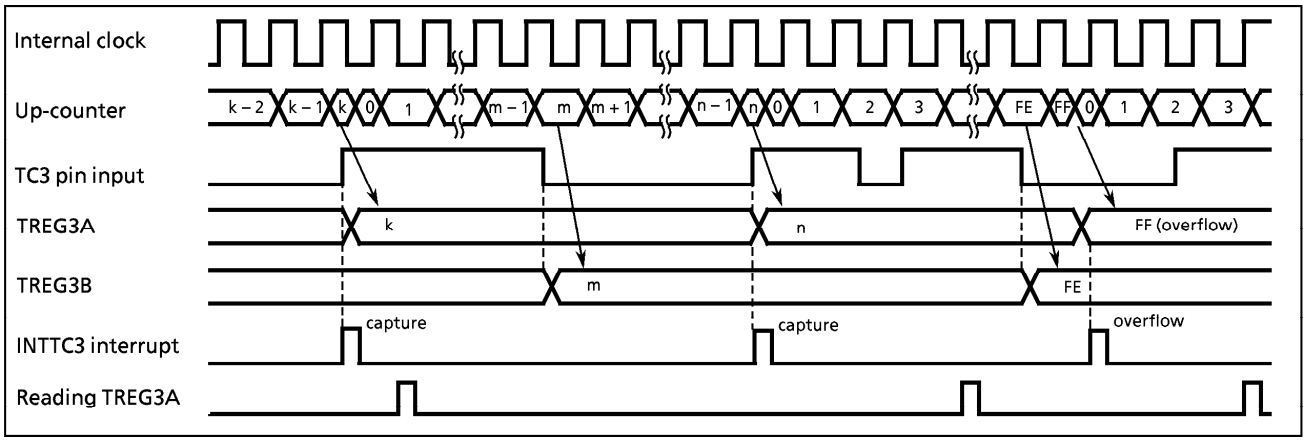



Figure 2-28. Timing Chart for Capture Mode (INT3ES = 0)

## 2.8 8-bit Timer/Counter (TC4)

### 2.8.1 Configuration

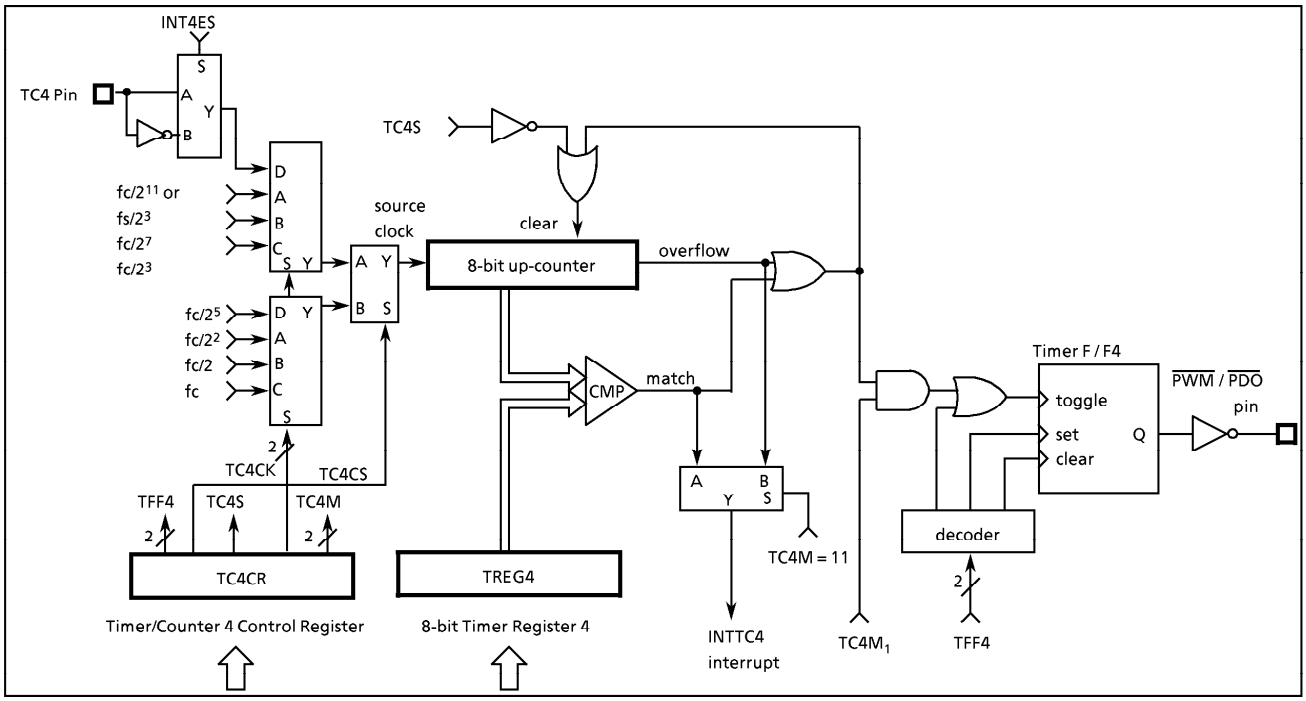



Figure 2-29. Timer/Counter 4

## 2.8.2 Control

The timer/counter 4 is controlled by a timer/counter 4 control register (TC4CR) and an 8-bit timer register 4 (TREG4). Reset does not affect TREG4.

TREG4 ($001B_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

Write only

TC4CR ($001C_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TFF4 | | TC4CS | TC4S | TC4CK | | TC4M | |

(Initial value : 0000 0000)

| | | | | |
|---|---|---|---|---|
| TC4M | TC4 operating mode select | 00 : Timer/event counter mode<br>01 : Reserved<br>10 : Programmable divider output (PDO) mode<br>11 : Pulse width modulation (PWM) output mode | | write only |
| TC4CK | TC4 source clock select | mode A (TC4CS = 0) | mode B (TC4CS = 1) | |
| | | 00 : fc / $2^{11}$ or fs / $2^3$<br>01 : fc / $2^7$<br>10 : fc / $2^3$<br>11 : External clock (TC4 pin input) | 00 : fc / $2^5$<br>01 : fc / $2^2$<br>10 : fc / 2<br>11 : fc | |
| TC4S | TC4 start control | 0 : Stop & clear<br>1 : Start | | |
| TC4CS | TC4 mode select | 0 : mode A<br>1 : mode B | | |
| TFF4 | Timer F/F 4 control | 00 : Clear<br>01 : Toggle<br>10 : Set<br>11 : – (Note 3) | | |

*Note 1 : fc; High-frequency clock [Hz], fs; Low-frequency clock [Hz], *; don't care*

*Note 2 : Set the operating mode, the source clock selection, the timer F/F 4 control and the edge selection (INT4ES) when the TC4 stops (TC4S = 0).*

*Note 3 : TFF4 must be set to "11" in the timer, event counter modes and B mode.*

*Note 4 : Values to be loaded to the timer register must satisfy the following condition.*

*(a) When in PWM output mode, 5 < TREG4 < 251*

*(b) When any other mode than PWM output mode, 0 < TREG4*

*Note 5 : Source clock fc/$2^2$, fc/2, and fc cannot be used except in PWM output mode.*

*Note 6 : TC4CR and TREG4 are write-only registers and must not be used with any of read-modify-write instructions.*

Figure 2-30. Timer Register 4 and TC4 Control Register

## 2.8.3 Function

The timer/counter 4 has four operating modes : timer, event counter, programmable divider output, and PWM output mode.

(1) **Timer** Mode

In this mode, the internal clock is used for counting up. The contents of TREG4 are compared with the contents of up-counter. If a match is found, a timer/counter 4 interrupt (INTTC4) is generated and the up-counter is cleared to "0". Counting up resumes after the up- counter is cleared.

Table 2-6. Source Clock (Internal Clock) for Timer/Counter 4

| Source clock | | | Resolution | | Maximum setting time | |
|---|---|---|---|---|---|---|
| Mode | NORMAL1/2, IDLE1/2 mode | SLOW, SLEEP mode | At fc = 8 MHz | At fs = 32.768 kHz | At fc = 8 MHz | At fs = 32.768 kHz |
| A | fc / $2^{11}$ or fs / $2^3$ [Hz] | fs / $2^3$ [Hz] | 256 μs | 244.14 μs | 65.3 ms | 62.2 ms |
| | fc / $2^7$ | – | 16 μs | – | 4.1 ms | – |
| | fc / $2^3$ | – | 1 μs | – | 255 μs | – |
| B | fc / $2^5$ | – | 4 μs | – | 1020 μs | |

(2) **Event Counter** Mode

In this mode, the TC4 pin input (external clock) pulse is used for counting up. Either the rising or falling edge can be selected with TC4ES (bit 4 in EINTCR). The contents of the TREG4 are compared with the contents of the up-counter. If a match is found, an INTTC4 interrupt is generated and the counter is cleared. The maximum applied frequency is $fc/2^4$ [Hz] in NORMAL1/2 or IDLE1/2 mode, and $fs/2^4$ [Hz] in SLOW or SLEEP mode. Two or more machine cycles are required for both the high and low levels of the pulse width.

(3) **Programmable Divider Output** (PDO) Mode

The internal clock is used for counting up. The contents of TREG4 are compared with the contents of the up-counter. Timer F/F 4 output is toggled and the counter is cleared each time a match is found. Timer F/F 4 output is inverted and output to the $\overline{\text{PDO}}$ (P55) pin. This mode can be used for 50% duty pulse output. Timer F/F 4 can be initialized by program, and it is initialized to "0" during reset. An INTTC4 interrupt is generated each time the $\overline{\text{PDO}}$ output is toggled.

Example : Output a 1024 Hz pulse (at fc = 4.194304 MHz)

```
LD      (TC4CR), 00000010B      ; Initializes the TC4 mode, source clock and timer F/F 4.
LD      (TREG4), 10H            ; 1/2048 ÷ 2^7/fc = 10H
LD      (TC4CR), 00010010B      ; Starts TC4
```

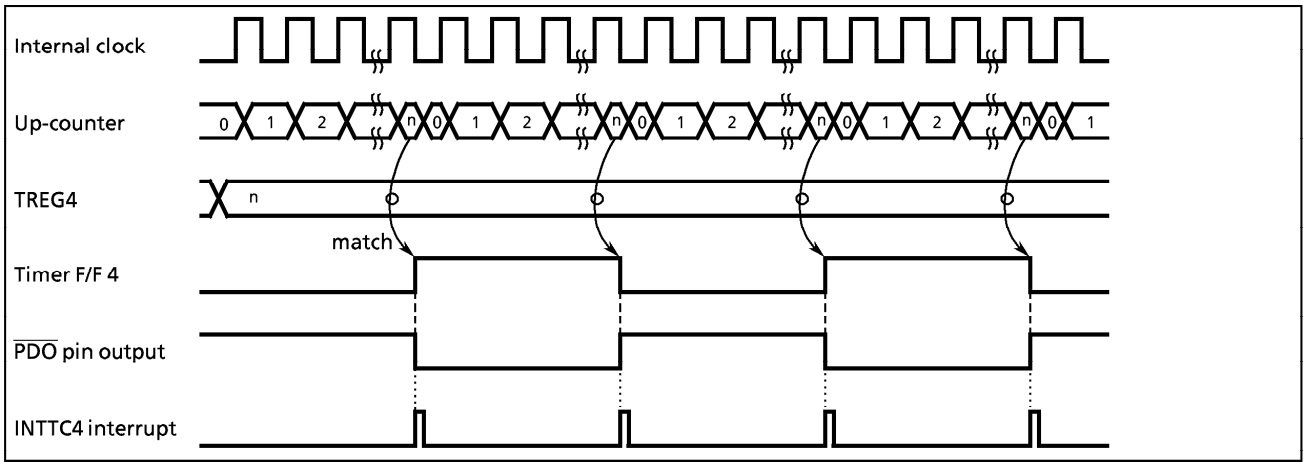



Figure 2-31. Timing Chart for PDO Mode

(4) **Pulse Width Modulation** (PWM) **Output** Mode

PWM output with a resolution of 8 bits is possible. The internal clock is used for counting up. The contents of TREG4 are compared with the contents of up-counter. If a match is found, the timer F/F 4 output is toggled. The counter continues counting. And, when an overflow occurs, the timer is again toggled and the counter is cleared. Timer F/F 4 output is inverted and output to the $\overline{\text{PWM}}$ (P55) pin. An INTTC4 interrupt is generated when an overflow occurs.

TREG4 is configured a 2-stage shift register and, during output, will not switch until one output cycle is completed even if TREG4 is overwritten; therefore, output can be altered continuously. Also, the first time, TREG4 is shifted by setting TC4S (bit 4 in TC4CR) to "1" after data are loaded to TREG4.

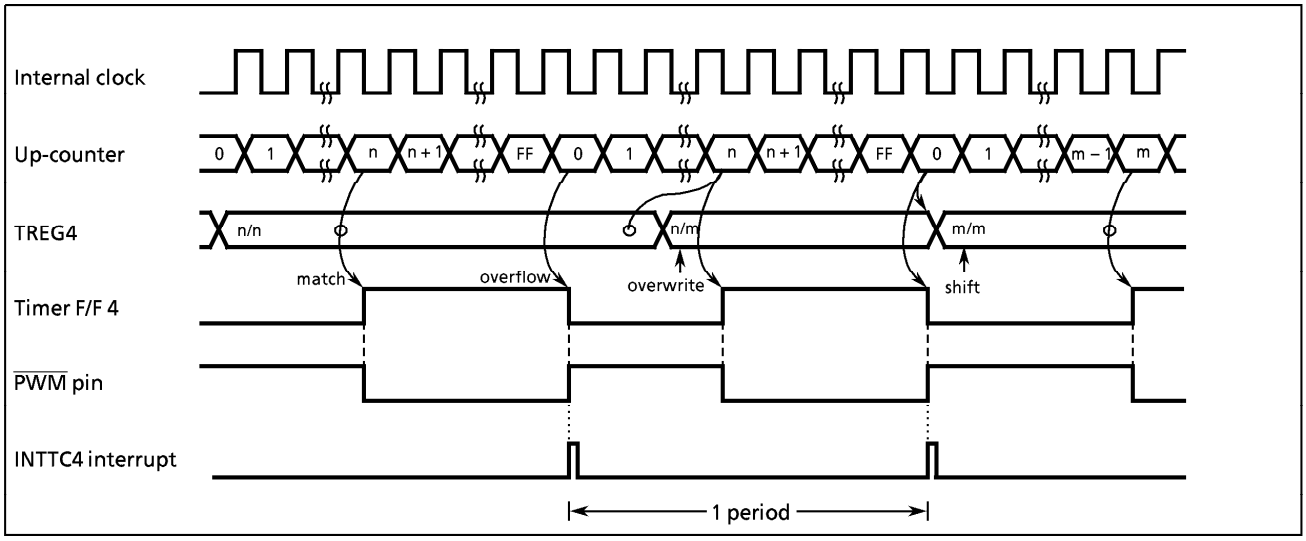



Figure 2-32. Timing Chart for PWM Mode

Table 2-7. PWM Output Mode

| Mode | Source clock: NORMAL1/2, IDLE1/2 mode DV7CK = 0 | Source clock: NORMAL1/2, IDLE1/2 mode DV7CK = 1 | Source clock: SLOW, SLEEP mode | Resolution: At fc = 8 MHz | Resolution: At fs = 32.768 kHz | Repeat cycle: At fc = 8 MHz | Repeat cycle: At fs = 32.768 kHz |
|---|---|---|---|---|---|---|---|
| A | $fc / 2^{11}$ [Hz] | $fs / 2^3$ [Hz] | $fs / 2^3$ [Hz] | 256 μs | 244.14 μs | 65.3 ms | 62.2 ms |
| | $fc / 2^7$ | – | – | 16 μs | – | 4.1 ms | – |
| | $fc / 2^3$ | – | – | 1 μs | – | 255 μs | – |
| B | $fc / 2^5$ [Hz] | – | – | 4 μs | – | 1024 μs | – |
| | $fc / 2^2$ | – | – | 500 ns | – | 128 μs | – |
| | $fc / 2$ | – | – | 250 ns | – | 64 μs | – |
| | fc | – | – | 125 ns | – | 32 μs | – |

## 2.9 Serial Interface (SIO1, SIO2)

The 87C841/CC41/CH41/CK41/CM41 each have two clocked-synchronous 8-bit serial interfaces (SIO1 and SIO2). Each serial interface has an 8-byte transmit and receive data buffer that can automatically and continuously transfer up to 64 bits of data.

The serial interfaces are connected to external devices via pins P44 (SO1), P43 (SI1), P42 ($\overline{SCK1}$) for SIO1 and P47 (SO2), P46 (SI2), P45 ($\overline{SCK2}$) for SIO2. The serial interface pins are also used as port P4. When used as serial interface pins, the output latches of these pins should be set to "1". In the transmit mode, pins P43 and P46 can be used as normal I/O ports, and in the receive mode, the pins P44 and P47 can be used as normal I/O ports.

### 2.9.1 Configuration

The SIO1 and SIO2 have the same configuration, except for the addresses/bit positions of the control/status registers and buffer registers.

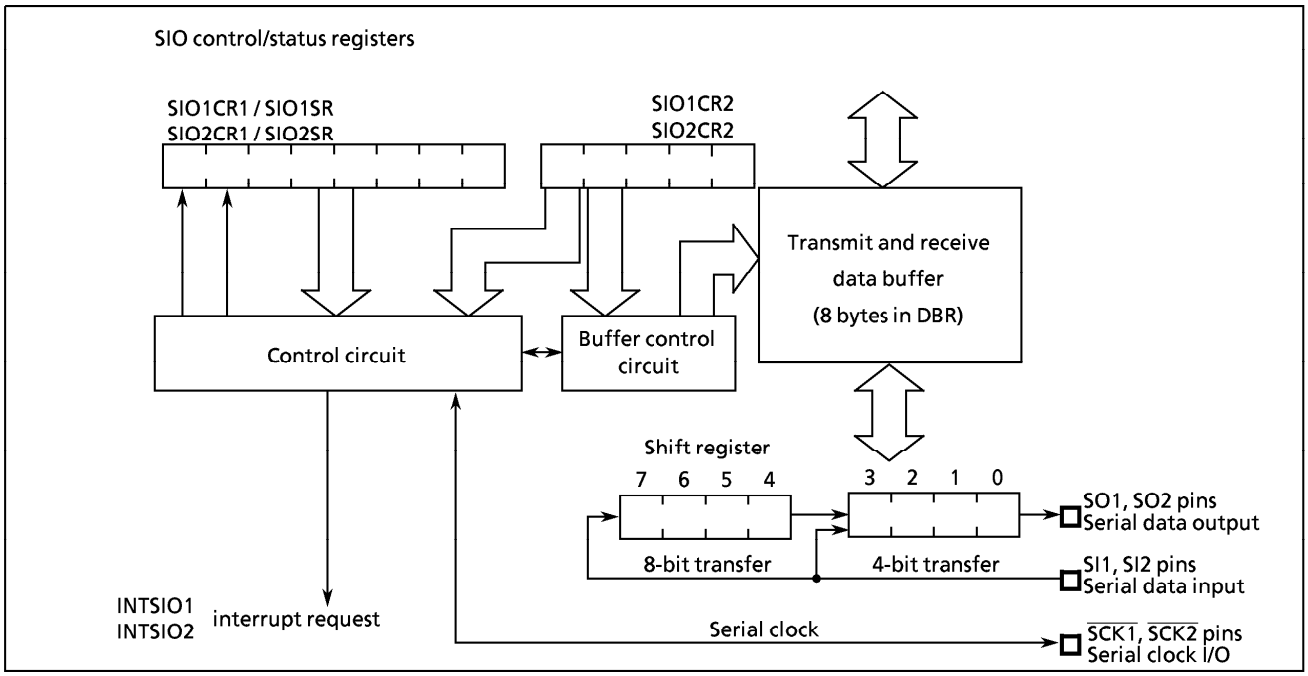



Figure 2-33. Serial Interfaces

### 2.9.2 Control

The serial interfaces are controlled by SIO control registers (SIO1CR1/SIO1CR2 or SIO2CR1/SIO2CR2). The serial interface status can be determined by reading SIO status registers (SIO1SR or SIO2SR).

The transmit and receive data buffer is controlled by the BUF (bits 2-0 in SIO1CR2/SIO2CR2). The data buffer is assigned to addresses $0FF0_H$ to $0FF7_H$ for SIO1 or $0FF8_H$ - $0FFF_H$ for SIO2 in the DBR area, and can continuously transfer up to 8 words (bytes or nibbles) at one time. When the specified number of words has been transferred, a buffer empty (in the transmit mode) or a buffer full (in the receive mode or transmit/receive mode) interrupt (INTSIO1 or INTSIO2) is generated.

When the internal clock is used as the serial clock in the 8-bit receive mode and the 8-bit transmit/receive mode, a fixed interval wait can be applied to the serial clock for each word transferred. Four different wait times can be selected with WAIT (bits 4 and 3 in SIO1CR2/SIO2CR2).

## SIO1, SIO2 Control Registers 1

**SIO1CR1** ($0020_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SIOS | SIO INH | SIOM | | | SCK | | |

(Initial value : 0000 0000)

**SIO2CR1** ($0022_H$)

| | | | |
|---|---|---|---|
| SIOS | Indicate transfer start/stop | 0 : Stop<br>1 : Start | write only |
| SIOINH | Continue/abort transfer | 0 : Continue transfer<br>1 : Abort transfer (automatically cleared after abort) | |
| SIOM | Transfer mode select | 000 : 8-bit transmit mode<br>010 : 4-bit transmit mode<br>100 : 8-bit transmit/receive mode<br>101 : 8-bit receive mode<br>110 : 4-bit receive mode | |
| SCK | Serial clock select | 000 : Internal clock fc / $2^{13}$ or fs / $2^5$ [Hz]<br>001 : Internal clock fc / $2^8$<br>010 : Internal clock fc / $2^6$<br>011 : Internal clock fc / $2^5$<br>111 : External clock (input from $\overline{SCK}$ pin)<br>(Output on $\overline{SCK}$ pin) | |

*Note 1 : fc ; High-frequency clock [Hz], fs ; Low-frequency clock [Hz]*

*Note 2 : Set SIOS to "0" and SIOINH to "1" when setting the transfer mode or serial clock.*

*Note 3 : SIO1CR2/SIO2CR2 are write-only registers, which cannot access any of in read-modify-write instruction such as bit operate, etc.*

## SIO1, SIO2 Status Registers

**SIO1SR** ($0020_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SIOF | SEF | "1" | "1" | "1" | "1" | "1" | "1" |

**SIO2SR** ($0022_H$)

| | | | |
|---|---|---|---|
| SIOF | Serial transfer operating status monitor | 0 : Transfer terminated<br>1 : Transfer in process | read only |
| SEF | Shift operating status monitor | 0 : Shift operation terminated<br>1 : Shift operation in process | |

## SIO1, SIO2 Control Registers 2

**SIO1CR2** ($0021_H$)
**SIO2CR2** ($0023_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | WAIT | | BUF | | |

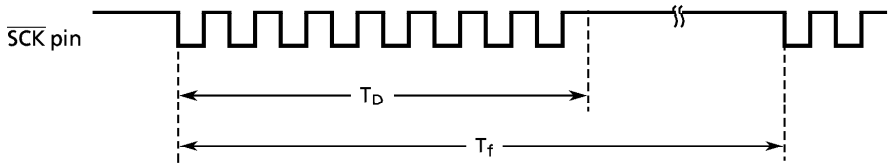
(Initial value: ***0 0000)

| | | | |
|---|---|---|---|
| WAIT | Wait control | 00 : $T_f = T_D$<br>01 : $T_f = 2T_D$<br>10 : $T_f = 4T_D$<br>11 : $T_f = 8T_D$ | Write only |
| BUF | Number of transfer words | Buffer address used<br>SIO1 / SIO2<br>000 : 1 word transfer — $0FF0_H$ / $0FF8_H$<br>001 : 2 words transfer — 0FF0 to $0FF1_H$ / 0FF8 to $0FF9_H$<br>010 : 3 words transfer — 0FF0 to $0FF2_H$ / 0FF8 to $0FFA_H$<br>011 : 4 words transfer — 0FF0 to $0FF3_H$ / 0FF8 to $0FFB_H$<br>100 : 5 words transfer — 0FF0 to $0FF4_H$ / 0FF8 to $0FFC_H$<br>101 : 6 words transfer — 0FF0 to $0FF5_H$ / 0FF8 to $0FFD_H$<br>110 : 7 words transfer — 0FF0 to $0FF6_H$ / 0FF8 to $0FFE_H$<br>111 : 8 words transfer — 0FF0 to $0FF7_H$ / 0FF8 to $0FFF_H$ | |

*Note 1 : *; don't care*

*Note 2 : WAIT is valid only in the 8-bit transmit / receive and 8-bit receive modes.*

*Note 3 : $T_f$; frame time, $T_D$ ; data transfer time*




*Note 4 : The lower 4 bits of each buffer are used during 4-bit transfers. Zeros (0) are stored to the upper 4bits when receiving.*

*Note 5 : Transmitting starts at the lowest address. Received data are also stored starting from the lowest address to the highest address. For example, in the case of SIO1, the first buffer address transmitted is $0FF0_H$.*

*Note 6 : The value to be loaded to BUF is held after transfer is completed.*

*Note 7 : SIO1CR2/SIO2CR2 are write-only registers, which cannot access any of in read-modify-write instruction such as bit operate, etc.*

Figure 2-34. SIO Control Registers and Status Registers

(1) Serial Clock

a. Clock Source

SCK (bits 2 to 0 in SIO1CR1/SIO2CR1) is able to select the following:

① Internal Clock

Any of four frequencies can be selected. The serial clock is output to the outside on the $\overline{SCK1}$ / $\overline{SCK2}$ pin. The $\overline{SCK}$ pin goes high when transfer starts.

When data writing (in the transmit mode) or reading (in the receive mode or the transmit/receive mode) cannot keep up with the serial clock rate, there is a wait function that automatically stops the serial clock and holds the next shift operation until the read/write processing is completed.

Table 2-8. Serial Clock Rate

| Serial clock | | | Maximum transfer rate | |
|---|---|---|---|---|
| NORMAL1/2, IDLE1/2 mode | | SLOW, SLEEP mode | | |
| DV7CK = 0 | DV7CK = 1 | | At fc = 8 MHz | At fs = 32.768 kHz |
| $fc / 2^{13}$ [Hz] | $fs / 2^5$ [Hz] | $fs / 2^5$ [Hz] | 0.954 kbit/s | 1 kbit/s |
| $fc / 2^8$ | $fc / 2^8$ | – | 30.5 | – |
| $fc / 2^6$ | $fc / 2^6$ | – | 122 | – |
| $fc / 2^5$ | $fc / 2^5$ | – | 244 | – |

*Note : 1Kbit = 1024bit*

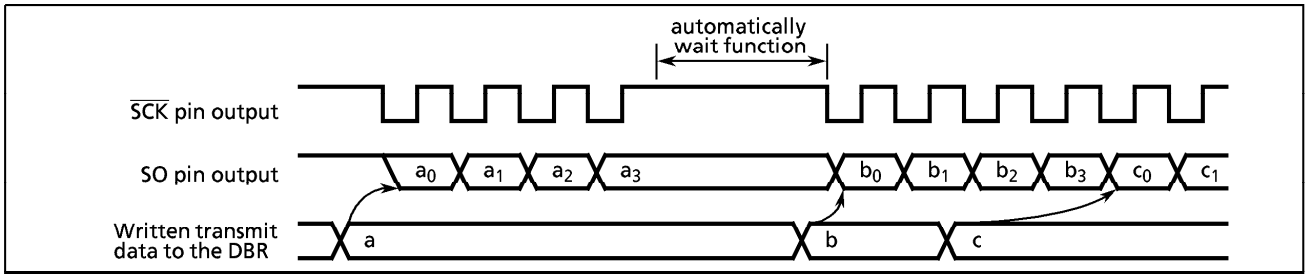
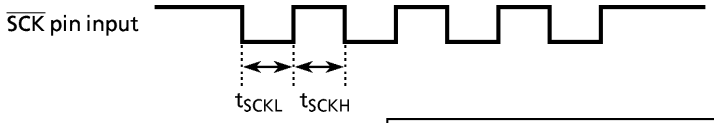



Figure 2-35. Clock Source (Internal Clock)

② External Clock

An external clock connected to the SCK1 / SCK2 pin is used as the serial clock. In this case, the P42 (SCK1) / P45 (SCK2)output latch must be set to "1". To ensure shifting, a pulse width of at least 4 machine cycles is required. Thus, the maximum transfer speed is 244K-bit/s. (at fc = 8 MHz).




*Note : tcyc = 4/fc (In NORAML1/2, IDLE1/2 modes)*
*4/fs (In SLOW, SLEEP modes)*

b. Shift edge

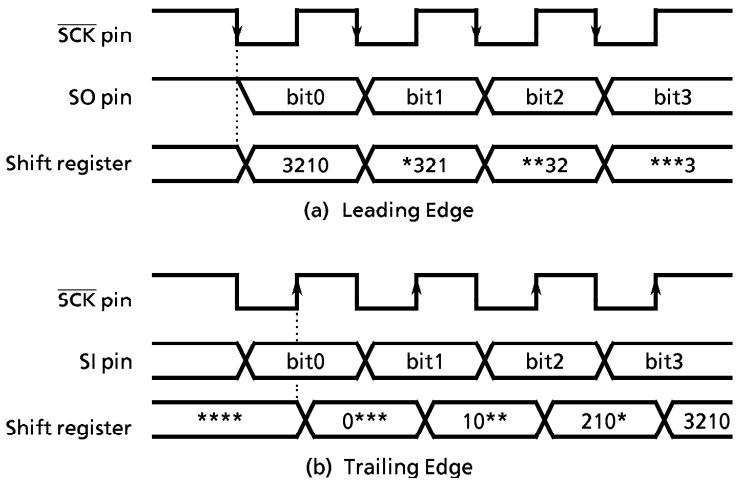
The leading edge is used to transmit, and the trailing edge is used to receive.

① Leading Edge

Transmitted data are shifted on the leading edge of the serial clock (falling edge of the SCK pin input/output).

② Trailing Edge

Received data are shifted on the trailing edge of the serial clock (rising edge of the SCK pin input/output).




*Note : * ; don't care*

Figure 2-36. Shift Edge

(2) Number of Bits to Transfer

Either 4-bit or 8-bit serial transfer can be selected. When 4-bit serial transfer is selected, only the lower 4 bits of the transmit/receive data buffer register are used. The upper 4 bits are cleared to "0" when receiving.

The data is transferred in sequence starting at the least significant bit (LSB).

(3) Number of Words to Transfer

Up to 8 words consisting of 4 bits of data (4-bit serial transfer) or 8 bits (8-bit serial transfer) of data can be transferred continuously. The number of words to be transferred is loaded to BUF1/BUF2 in SIOBCR.

An INTSIO interrupt is generated when the specified number of words has been transferred. If the number of words is to be changed during transfer, the serial interface must be stopped before making the change.

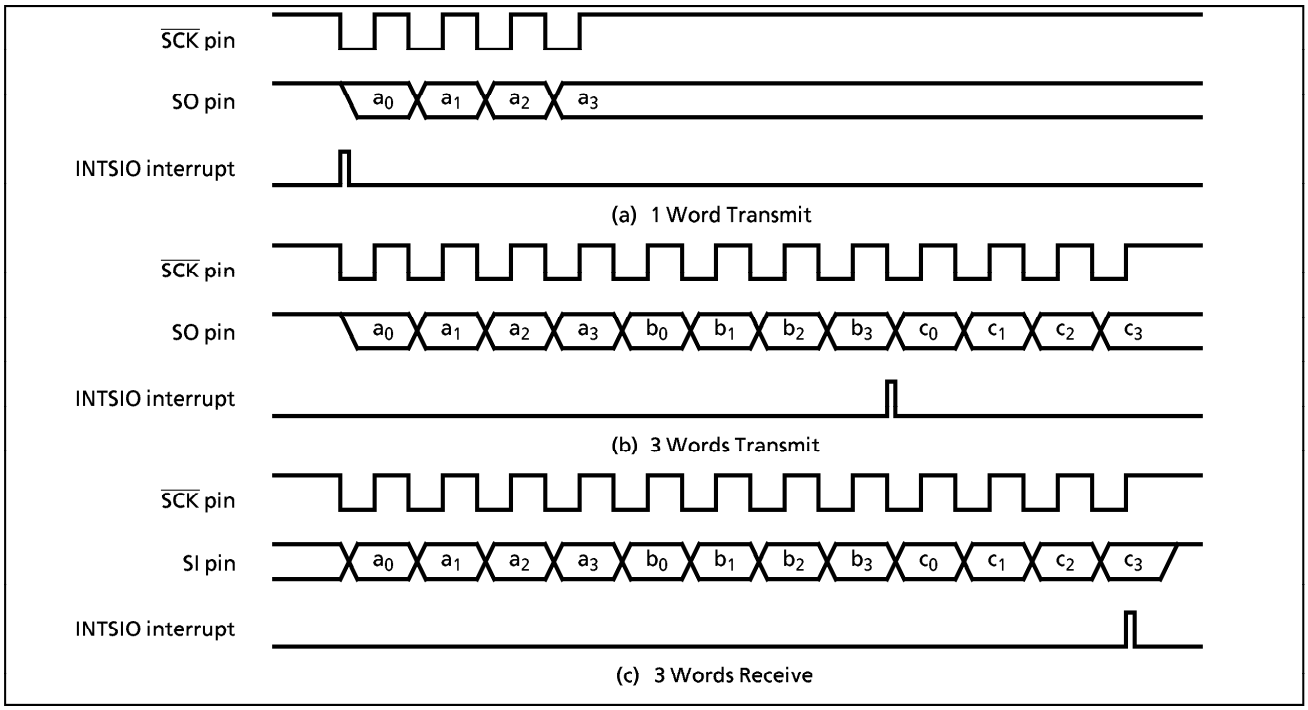



Figure 2-37. Number of Bits to Transfer (Example : 4-bit serial transfer)

## 2.9.3 Transfer Mode

SIOM (bits 5 to 3 in SIO1CR1/SIO2CR1) is used to select the transmit, receive, or transmit/receive mode.

(1) **4-bit and 8-bit Transmit** Modes

In these modes, the SIO1CR1/SIO2CR1 is set to the transmit mode and then the data to be transmitted first are written to the data buffer registers (DBR). After the data are written, the transmission is started by setting SIOS to "1". The data are then output sequentially to the SO pin in synchronous with the serial clock, starting with the least significant bit (LSB). As soon as the LSB has been output, the data are transferred from the data buffer register to the shift register. When the final data bit has been transferred and the data buffer register is empty, an INTSIO (buffer empty) interrupt is generated to request the next transmitted data.

When the internal clock is used, the serial clock will stop and an automatic-wait will be initiated if the next transmitted data are not loaded to the data buffer register by the time the number of data words specified with the BUF has been transmitted. Writing even one word of data cancels the automatic-wait; therefore, when transmitting two or more words, always write the next word before transmission of the previous word is completed.

*Note : Waits are also canceled by writing to a DBR not being used as a transmit data buffer register; therefore, during SIO do not use such DBR for other applications.*

When an external clock is used, the data must be written to the data buffer register before shifting next data. Thus, the transfer speed is determined by the maximum delay time from the generation of the interrupt request to writing of the data to the data buffer register by the interrupt service program.

When the transmit is started, after the SIOF goes "1" output from the SO pin holds final bit of the last data until falling edge of the $\overline{SCK}$.
The transmission is ended by clearing SIOS to "0" at the time that the final bit of the data being shifted out has been transferred. That the transmission has ended can be determined from the status of SIOF (bit 7 in SIO1SR/SIO2SR) because SIOF is cleared to "0" when a transfer is completed.
When an external clock is used, it is also necessary to clear SIOS to "0" before shifting the next data; otherwise, dummy data will be transmitted and the operation will end.

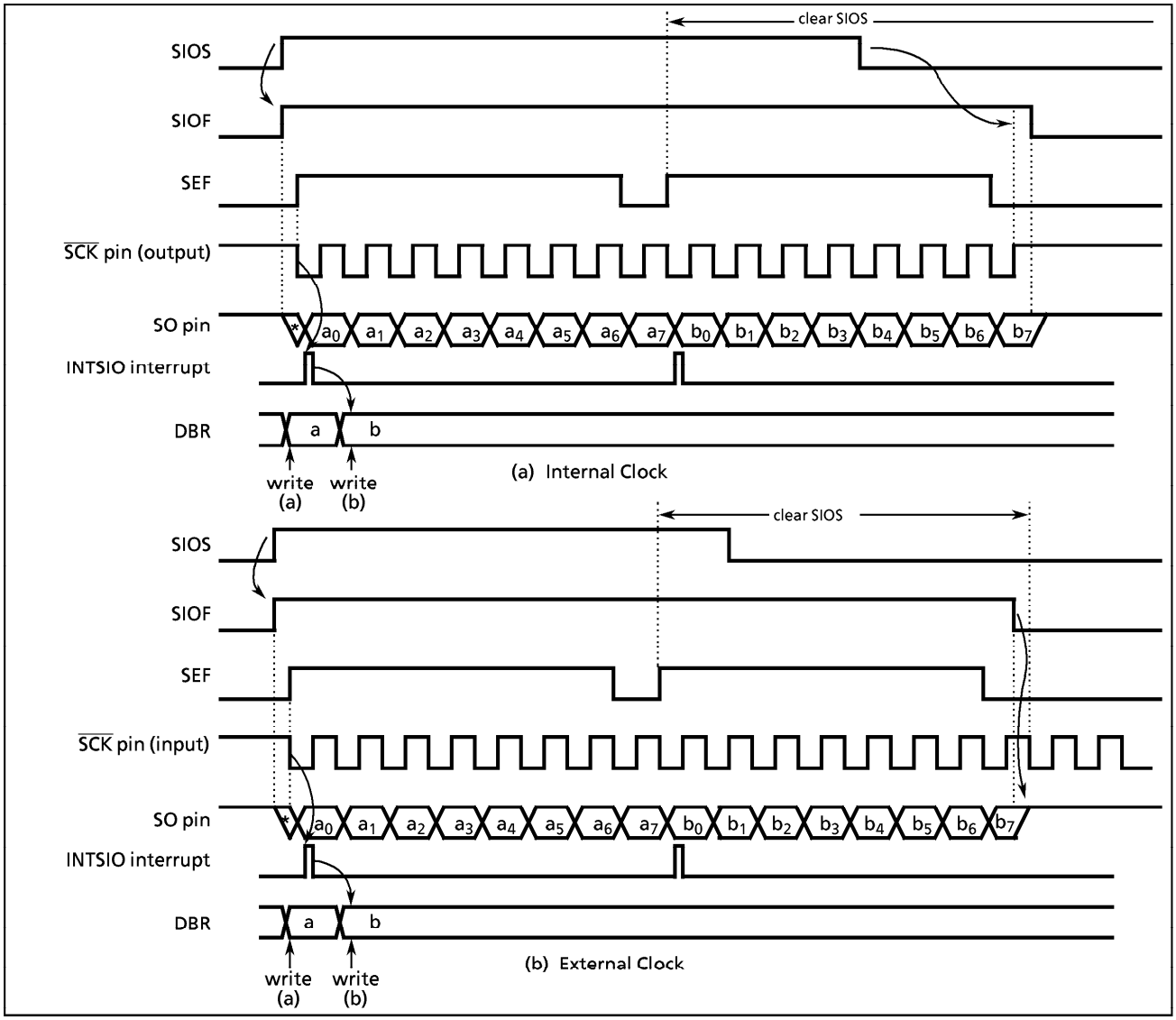



Figure 2-38. Transfer Mode (Example: 8-bit, 1 Word Transfer)

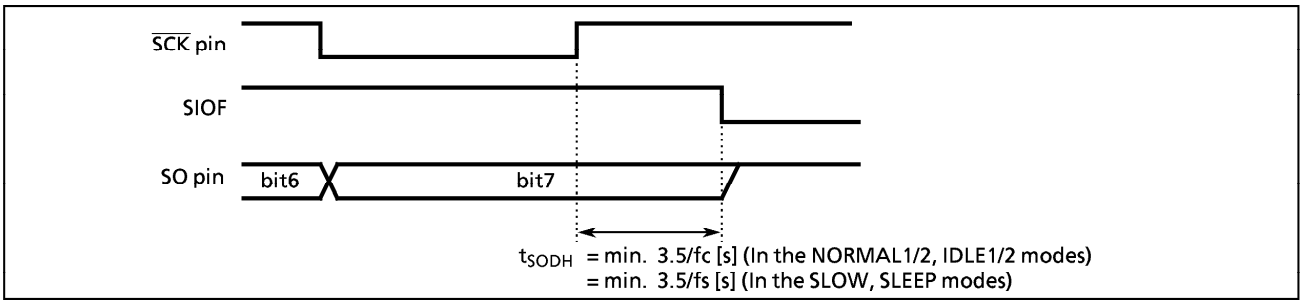



Figure 2-39. Transmitted Data Hold Time at end of transmit

(2) **4-bit and 8-bit Receive** Modes

After setting the control registers to the receive mode, set SIOS to "1" to enable receiving. The data are then transferred to the shift register via the SI pin in synchronous with the serial clock. When one word of data has been received, it is transferred from the shift register to the data buffer register (DBR). When the number of words specified with the BUF has been received, an INTSIO (buffer full) interrupt is generated to request that these data be read out. The data are then read from the data buffer registers by the interrupt service program.

When the internal clock is used, and the previous data are not read from the data buffer register before the next data are received, the serial clock will stop and an automatic-wait will be initiated until the data are read. A wait will not be initiated if even one data word has been read.

*Note : Waits are also canceled by reading a DBR not being used as a received data buffer register is read; therefore, during SIO do not use such DBR for other applications.*

When an external clock is used, the shift operation is synchronized with the external clock; therefore, the previous data are read before the next data are transferred to the data buffer register. If the previous data have not been read, the next data will not be transferred to the data buffer register and the receiving of any more data will be canceled. When an external clock is used, the maximum transfer speed is determined by the delay between the time when the interrupt request is generated and when the data received have been read.

Clear SIOS to "0" to end receiving. When SIOS is cleared, the current data are transferred to the buffer in 4-bit or 8-bit blocks. The receiving mode ends when the transfer is completed. SIOF is cleared to "0" when receiving is ended and thus can be sensed by program to confirm that receiving has ended.

*Note : The buffer contents are lost when the transfer mode is switched. If it should become necessary to switch the transfer mode, end receiving by clearing SIOS to "0", read the last data and then switch the transfer mode.*

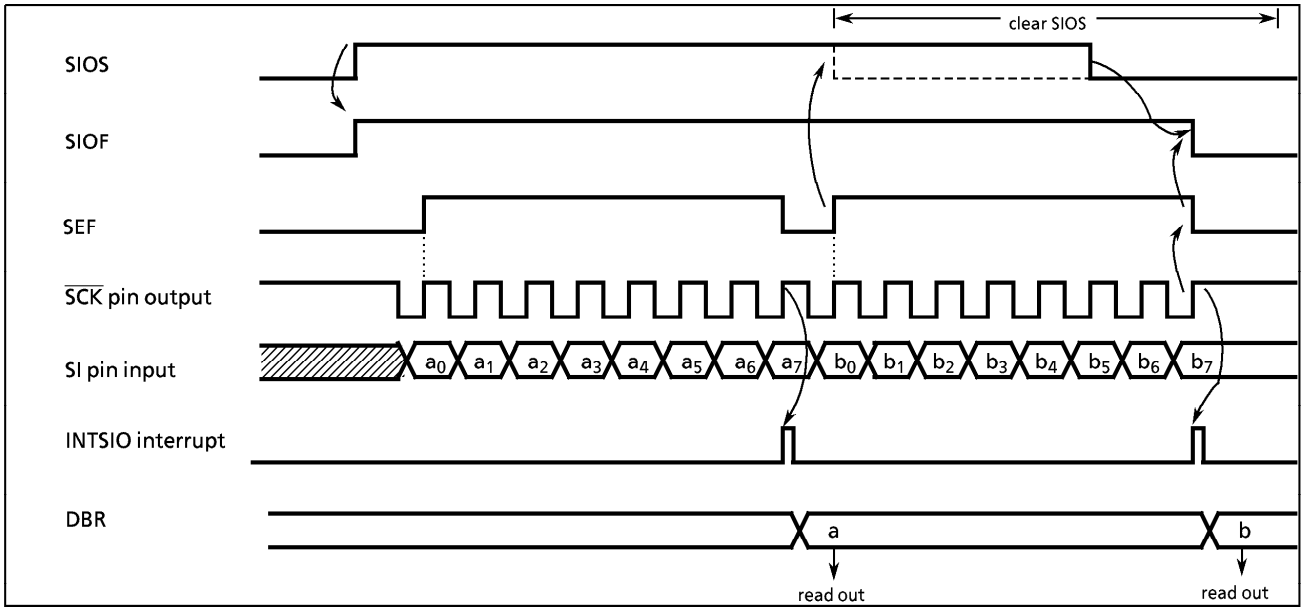



Figure 2-40. Receive Mode (Example : 8-bit, 1 word, internal clock)

(3) **8-bit Transmit/Receive** Mode

After setting the control registers to the 8-bit transmit/receive mode, write the data to be transmitted first to the data buffer registers (DBR). After that, enable transceiving by setting SIOS to "1". When transmitting, the data are output from the SO pin at leading edges of the serial clock. When receiving, the data are input to the SI pin at the trailing edges of the serial clock. 8-bit data are transferred from the shift register to the data buffer register. An INTSIO interrupt is generated when the number of data words specified with the BUF has been transferred. The interrupt service program reads the received data from the data buffer register and then writes the data to be transmitted. The data buffer register is used for both transmitting and receiving; therefore, always write the data to be transmitted after reading the received data.

When the internal clock is used, a wait is initiated until the received data are read and the next data are written.

When an external clock is used, the shift operation is synchronized with the external clock; therefore, it is necessary to read the received data and write the data to be transmitted next before starting the next shift operation. When an external clock is used, the transfer speed is determined by the maximum delay between generation of an interrupt request and the received data are read and the data to be transmitted next are written.

When the transmit is started, after the SIOF goes "1" output from the SO pin holds final bit of the last data until falling edge of the $\overline{SCK}$.

Clear SIOS to "0" to enable the transmit mode. When SIOS is cleared, the current data are transferred to the data buffer register in 8-bit blocks. The transmit mode ends when the transfer is completed. SIOF is cleared to "0" when receiving is ended and thus can be sensed by program to confirm that receiving has ended.

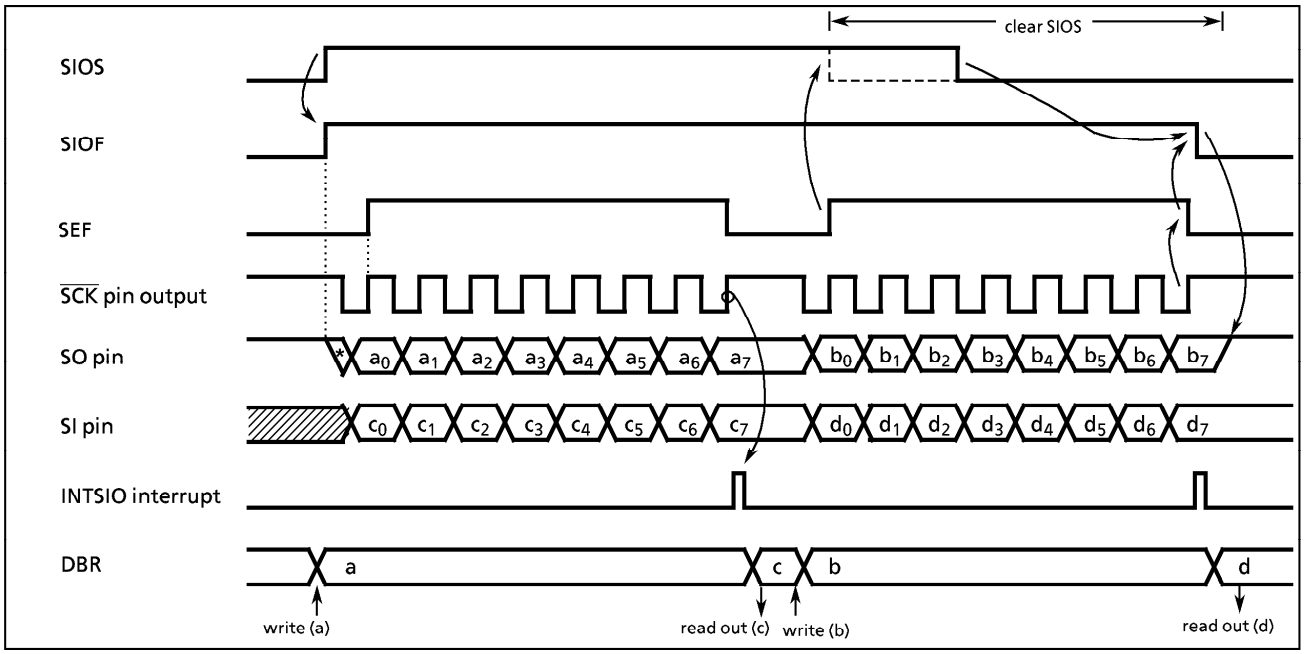



Figure 2-41. Transmit/Receive Mode (Example : 8-bit, 1word, internal clock)

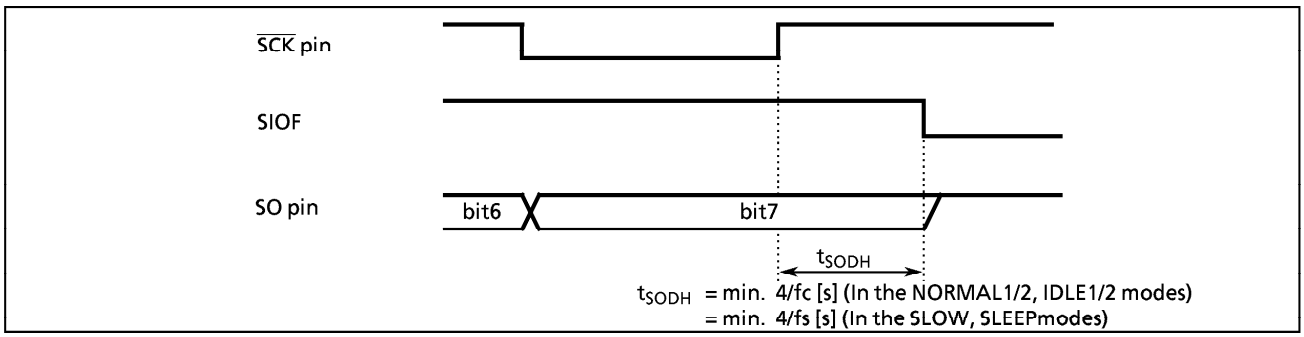



Figure 2-42. Transmitted Data Hold Time at end of transmit/receive

## 2.10 10-bit A/D Converter (ADC)

The 87C841/CC41/CH41/CK41/CM41 each have an 16-channel multiplexed-input 10-bit successive approximate type A/D converter.

### 2.10.1 Configuration

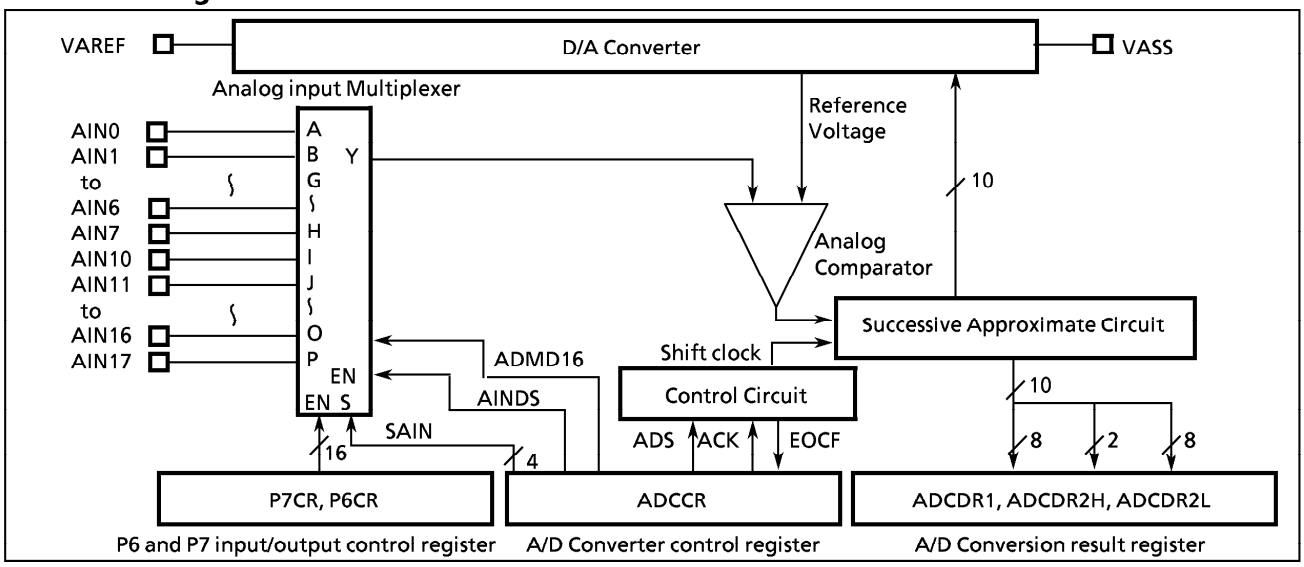



Figure 2-43. A/D Converter

### 2.10.2 Control

The A/D converter is controlled by an A/D converter control register (ADCCR), a port P6 input/output control register (P6CR) and a port P7 input/output control register (P7CR).

A/D Converter Control Register

ADCCR ($000E_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| EOCF/ ADMD16 | ADS | ACK | AINDS | SAIN | | | |

(Intial value : 0000 0000)

| | | | |
|---|---|---|---|
| SAIN | Analog input selection | 0000 : AIN0<br>0001 : AIN1<br>0010 : AIN2<br>0011 : AIN3<br>0100 : AIN4<br>0101 : AIN5<br>0110 : AIN6<br>0111 : AIN7<br>1000 : AIN10<br>1001 : AIN11<br>1010 : AIN12<br>1011 : AIN13<br>1100 : AIN14<br>1101 : AIN15<br>1110 : AIN16<br>1111 : AIN17 | R/W |
| AINDS | Analog input control | 0 : Enable<br>1 : Disable | |
| ACK | Conversion time | 0 : 196/fc : (at 8 MHz 24.5 μs). (at 4.2 MHz 46.7 μs)<br>1 : 736/fc : (at 8 MHz 92 μs). (at 4.2 MHz 176 μs) | |
| ADS | A/D convertion start | 0 : –<br>1 : A/D convertion start | |
| EOCF | End of A/D conversion flag | 0 : Under conversion or before conversion<br>1 : End of conversion | Read only |
| ADMD16 | Number of analog input selection | 0 : 8ch (AIN0 to 7)<br>1 : 16ch (AIN10 to 17) | Write only |

*Note 1 : Select analog input when A/D converter stops.*
*Note 2 : The ADS is automatically cleared to "0" after starting conversion.*
*Note 3 : The EOCF is cleared to "0" when reading the ADCDR1 or the ADCDR2H, ADCDR2L.*
*Note 4 : The EOCF is read-only.*
*Note 5 : The ADMD16 is write-only.*
*Note 6 : Select 16ch mode (ADMD16 = 1) when selecting from the AIN 10 to the AIN17.*

Figure 2-44. A/D Converter Control Register

A/D Conversion Result Register

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| ADCDR1 ($000F_H$) | DATA9 | DATA8 | DATA7 | DATA6 | DATA5 | DATA4 | DATA3 | DATA2 | Read only |
| ADCDR2H ($0025_H$) | "1" | "1" | "1" | "1" | "1" | "1" | DATA9 | DATA8 | Read only |
| ADCDR2L ($0024_H$) | DATA7 | DATA6 | DATA5 | DATA4 | DATA3 | DATA2 | DATA1 | DATA0 | Read only |

P6 input/output control register

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| P6CR ($000C_H$) | | | | | | | | | (Initial value : 1111 1111) |

| | | | |
|---|---|---|---|
| P6CR | P6 input control | 0 : input port<br>1 : analog input (AINDS = 0) | write only |

P7 input/output control register

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| P7CR ($000D_H$) | | | | | | | | | (Initial value : 0000 0000) |

| | | | |
|---|---|---|---|
| P7CR | P7 input/output control | 0 :<br>1 : } See below | write only |

| ADCCR | | P7CR | |
|---|---|---|---|
| ADMD16 | AINDS | 0 | 1 |
| 0 | 0 | input port | output port |
| 1 | 0 | input port except analog input selected by SAIN | output port except analog input selected by SAIN |
| 0 | 1 | input port | output port |
| 1 | 1 | input port | output port |

Figure 2-45. A/D Converter Result Register and P6, P7 Input/Output Control Register

## 2.10.3 Operation

Apply analog reference voltage to pins VAREF and VASS.

(1) Start of A/D conversion

First, select one of sixteen analog inputs AIN17 to AIN10, AIN7 to AIN0 with the SAIN (bits 3 to 0 in ADCCR) and the ADMD16 (bit 7 in ADCCR). Clear the AINDS (bit 4 in ADCCR) to "0" and set the corresponding P6CR bit to "1" for an analog input when AIN7 to AIN0 is selected.

A/D conversion time is set with the ACK (bit 5 in ADCCR).

A/D conversion is started by setting the ADS (bit 6 in ADCCR) to "1". The conversion is accomplished in 46 machine cycles (184/fc [s], ACK = 0).

The EOCF (bit 7 in ADCCR) is set to "1" at the end of the conversion.

If the ADS (bit 6 in ADCCR) is set to "1" during the conversion, the operation is initialized and the conversion is started again.

*Note : The pin that is not used as an analog input can be used as regular input or output pins. During conversion, do not perform output instruction to maintain a precision for all of the pins.*

(2) Reading of A/D conversion result

After the end of the conversion, read the conversion result from the ADCDR1 or the ADCDR2H, ADCDR2L.

The EOCF is automatically cleared to "0" when reading the ADCDR1 or the ADCDR2H, ADCDR2L.

Reading the conversion result during A/D conversion, an unexpected value is given.

(3) A/D conversion in STOP mode

When the MCU places in the STOP mode during the A/D conversion, the conversion is terminated and the ADCDR1 or ADCDR2H, ADCDR2L contents become indefinite.

However, if the STOP mode is started after the end of conversion (EOCF = 1), the ADCDR1 or the ADCDR2H, ADCDR2L contents are held.

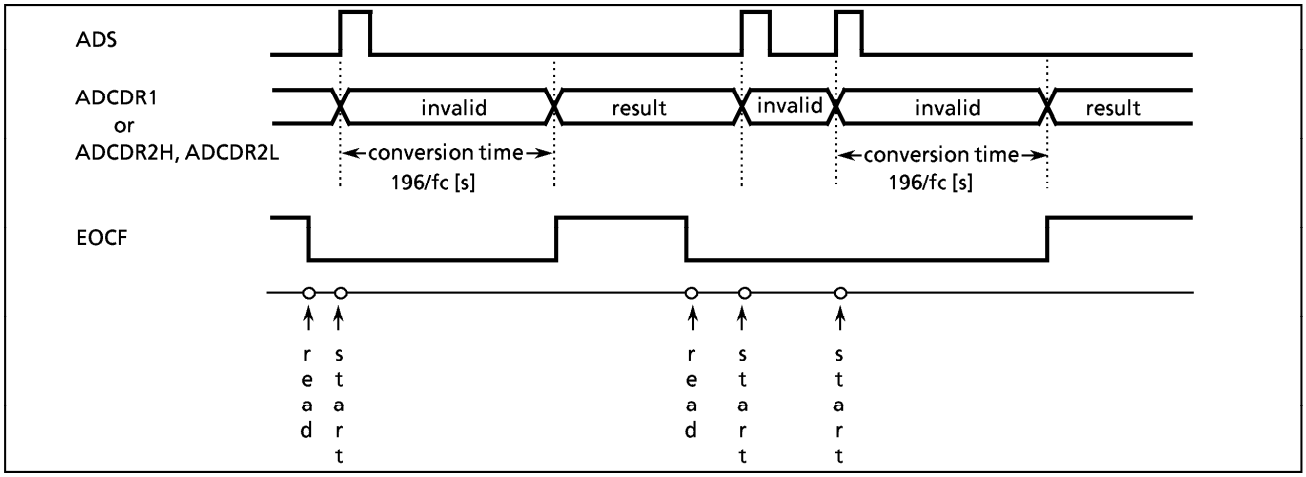



Figure 2-46. A/D Conversion Timing Chart

Example:

```
                ; AIN SELECT
                LD          (ADCCR) , 00100100B     ;   selects conversion time and AIN4
                ; A/D CONVERT START
                SET         (ADCCR) . 6             ;   ADS = 1
SLOOP    :      TEST        (ADCCR) . 7             ;   EOCF = 1 ?
                JRS         T, SLOOP
                ; RESULT DATA READ
                LD          (9EH), (ADCDR2H)
                LD          (9DH), (ADCDR2L)
```

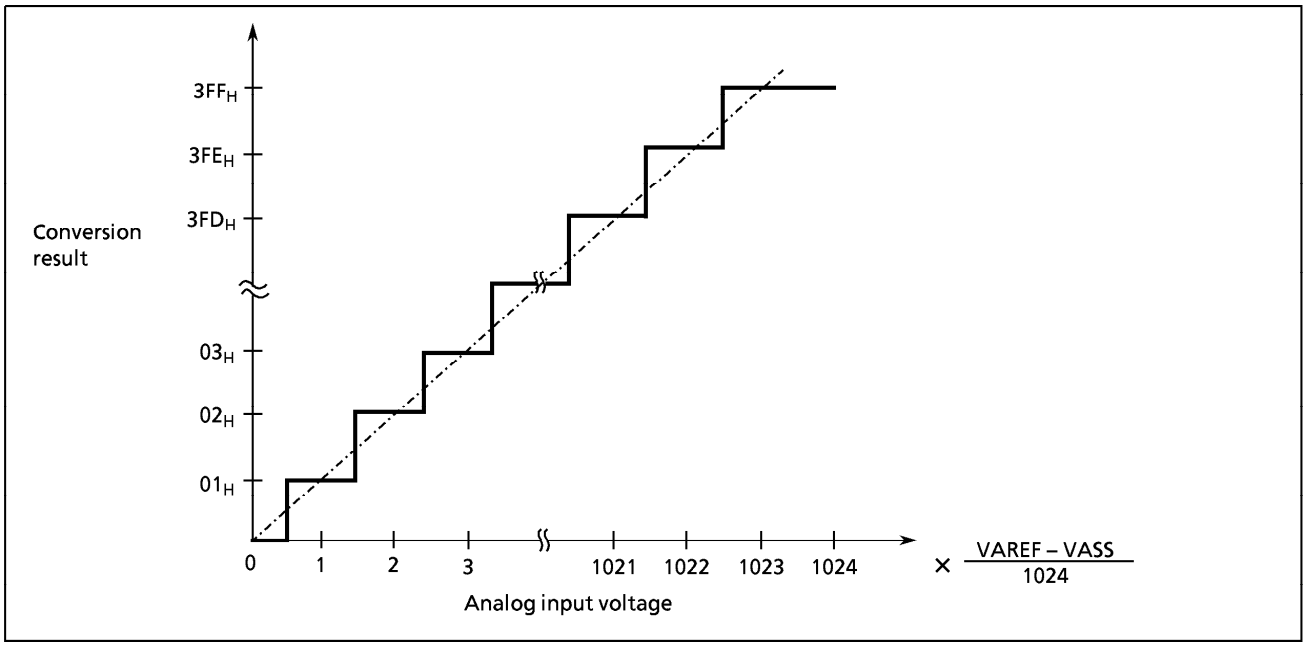



Figure2-47. Analog Input Voltage vs A/D Conversion Result (typ.)

## 2.11 8-bit High-speed PWM

The 87C841/CC41/CH41/CK41/CM41 have an internal 2-channel high-speed PWM. The 2-channel high-speed PWM can generate different waves from a data register of each channel by writing data to the register.

The 2-channel high-speed PWM are also used as a port P53 ($\overline{\text{HPWM0}}$) and a port P54 ($\overline{\text{HPWM1}}$). When used as the hig-speed PWM pin, the output latch should be set to "1".

### 2.11.1 Configuration

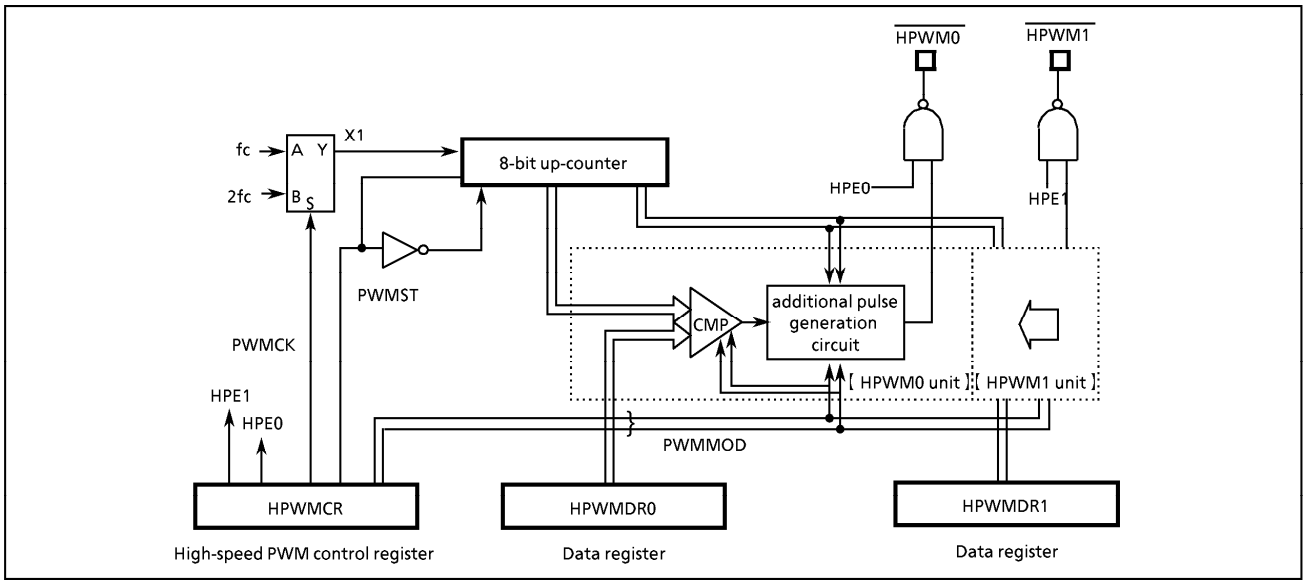



Figure 2-48. High-speed

### 2.11.2 Control

Control register

HPWMCR ($0026_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| HPE1 | HPE0 | | | PWMST | PWMCK | PWMMOD | |

Write only (initial value : 00** 0000)

| | | | |
|---|---|---|---|
| PWMMOD | PWM mode selection | 00 : mode 0 (8 bits)<br>01 : mode 1 (7 bits)<br>10 : mode 2 (6 bits)<br>11 : – | write only |
| PWMCK | PWM source clock (X1) selection | 0 : fc<br>1 : 2fc (only mode 0) | |
| PWMST | RUN/STOP of 8-bit up-counter | 0 : STOP<br>1 : RUN | |
| HPE0 | HPWM0 output control | 0 : disable<br>1 : enable | |
| HPE1 | HPWM1 output control | 0 : disable<br>1 : enable | |

Data register

HPWMDR0 ($0027_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DATA7 | DATA6 | DATA5 | DATA4 | DATA3 | DATA2 | DATA1 | DATA0 |

Write only (initial value : **** ****)

HPWMDR1 ($0028_H$)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DATA7 | DATA6 | DATA5 | DATA4 | DATA3 | DATA2 | DATA1 | DATA0 |

Write only (initial value : **** ****)

Figure 2-49. Registers of High-speed PWM

### 2.11.3 Operation

The high-speed PWM is controlled by the control register (HPWMCR) and data register (HPWMR0, 1). To write data to the above registers, set the PWMST (bit 3 in HPWMCR) to "1", which is enable mode. When the PWMST is clear to "0", these registers are in reset mode, which sets up the high-speed PWM for a software reset.

(1) Operation mode

The high-speed PWM has three operation modes.

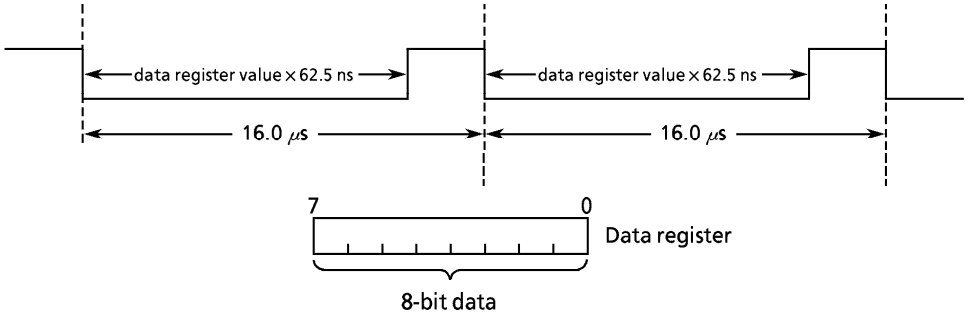
- 8-bit mode : ($T = 2^8 \times$ clock cycle $f \fallingdotseq 32$ kHz, $f \fallingdotseq$ *64 kHz)
- 7-bit mode : ($T = 2^7 \times$ clock cycle $f \fallingdotseq 64$ kHz)
- 6-bit mode : ($T = 2^6 \times$ clock cycle $f \fallingdotseq 128$ kHz)

*Note : * indicates the value when the source clock (X1) operates at 16 MHz (at 2fc : fc = 8 MHz).*

Operation mode is set by the PWMMOD (bit 0, 1 in HPWMCR). Operation mode applies commonly to all channels. Two modes cannot be used at any given time.

① 8-bit mode

8-bit mode generates a pulse with 16.0 μs cycle at a frequency of approximately 64 kHz (X1 = 16 MHz).




The minimum pulse width is 62.5 ns (data "1") and the maximum pulse width is 15.0 μs (data "F0").

Pulse width = 8-bit data × 62.5ns

A wave cycle example is shown the figure 2-50. (The value is when X1 = 16 MHz)

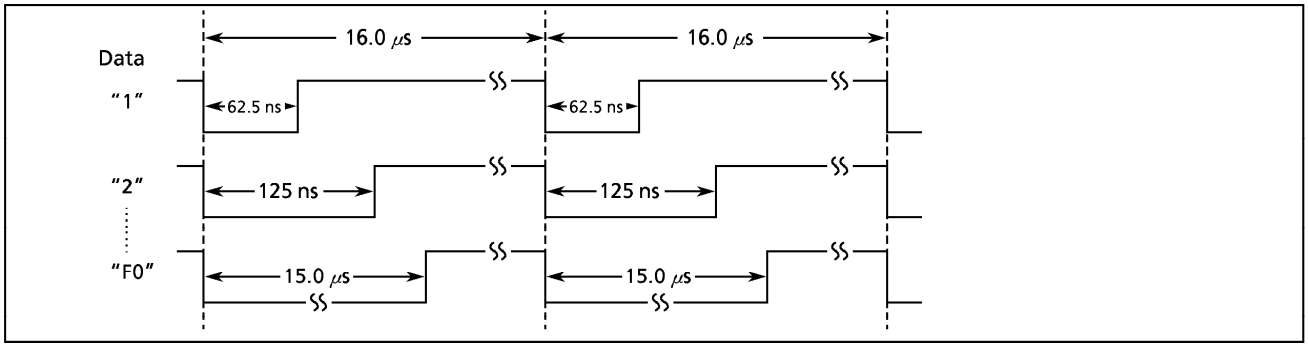
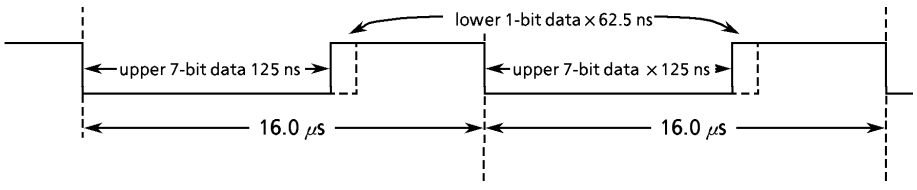


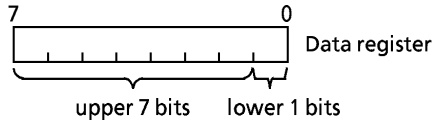

Figure 2-50. 8-bit mode

② 7-bit mode

7-bit mode generates a pulse with 16.0 μs cycle at a frequency of approximately 64 kHz (X1 = 8 MHz).



7-bit mode has 7 bits for a cycle ($2^7$ × 125 ns/cycle) and 1 bit for a 62.5 ns resolution (1/2 cycle of source clock (X1)).
When the lower 1 bit is "1", the additional 62.5 ns pulse is output.
The minimum pulse width is 62.5 ns (data "1") and the maximum pulse width is 15.0625 μs (data "F1").




Pulse width = (Upper 7-bit data × 125 ns) + (Lower 1-bit data × 62.5 ns)
A wave cycle example is shown the figure 2-51. (The value is when X1 = 8 MHz)

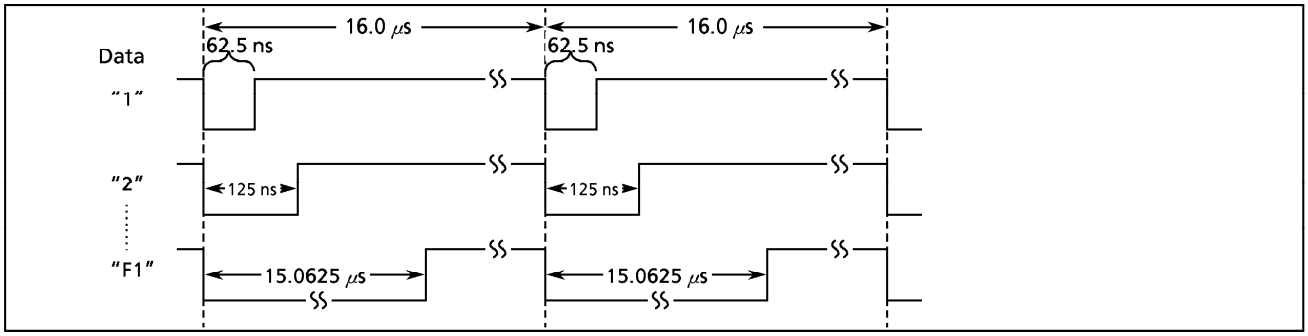
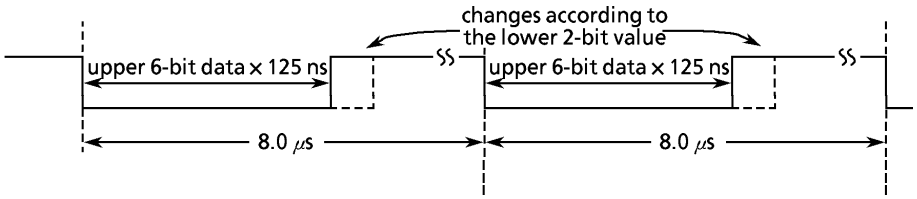
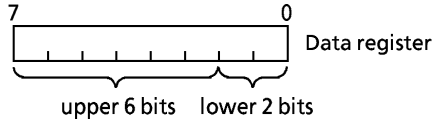



Figure 2-51. 7-bit mode

③ 6-bit mode
6-bit mode generates a pulse with 8.0 μs cycle at a frequency of approximately 128 kHz (X1 = 8 MHz).




6-bit mode has 6 bits for a cycle ($2^6$ × 125 ns/cycle) and 2 bits for a 31.25 ns resolution. Although the actual resolution every other cycle is 62.5 ns, a 31.25 ns resolution is simulated in the following way : The first cycle outputs a 62.5 ns pulse, which is averaged with the second cycle of 0ns. This pattern alternates continually and averages to a 62.5 ns (data "1") and the maximum equivalent pulse width is 7.625 μs (data "F3").




Pulse width = (upper 6-bit data × 125 ns) + (*lower 2-bit data)
*Equivalent time added for lower 2-bit data is shown below.

| 2-bit data | | Equivalent time added | |
|---|---|---|---|
| 0 | 0 | 0 | ns |
| 0 | 1 | 31.25 | ns |
| 1 | 0 | 62.5 | ns |
| 1 | 1 | 93.75 | ns |

A wave cycle example is shown the figure 2-52. (The value is when X1 = 8 MHz.)

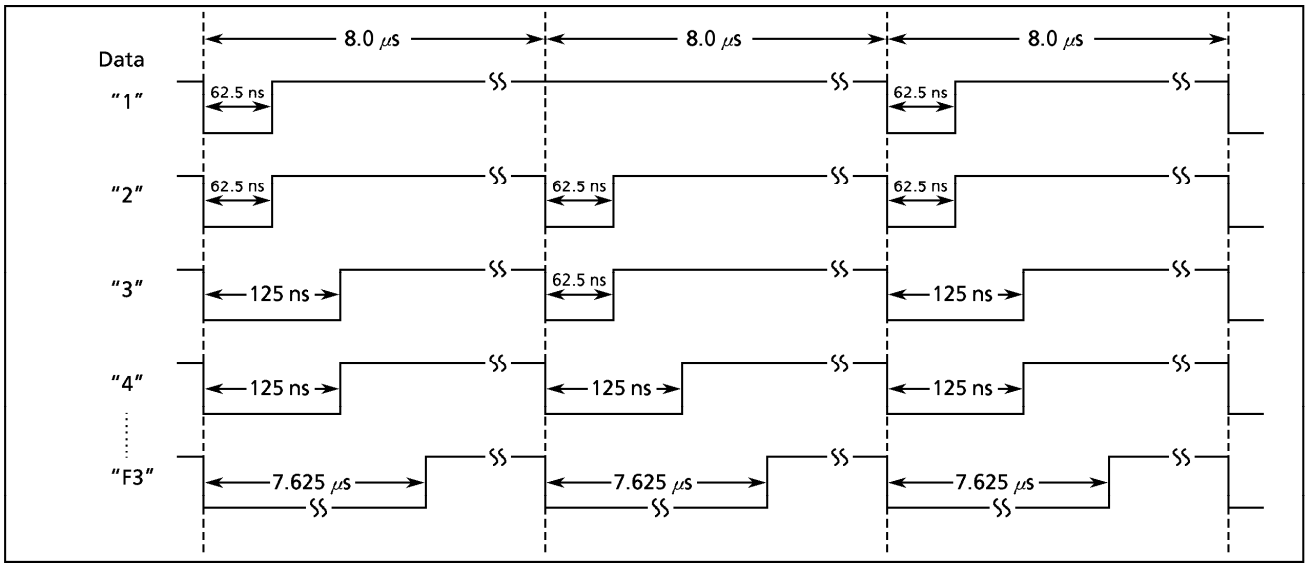



Figure 2-52. 6-bit mode

(2) Output data setting

Output data is set by writing to data registers (HPWMDR0, 1).

Example : To output 11.25 μs wave with $\overline{\text{HPWM0}}$ in 7-bit mode with a source clock (X1) = 8 MHz:

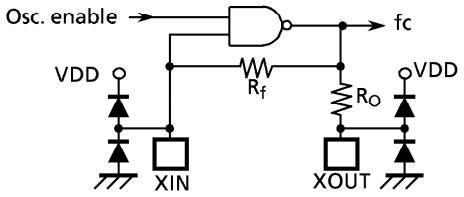
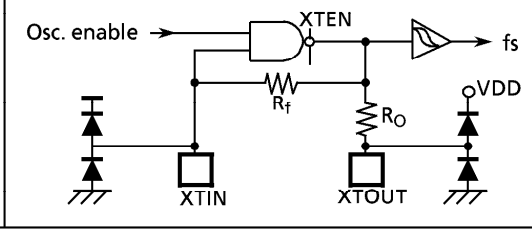
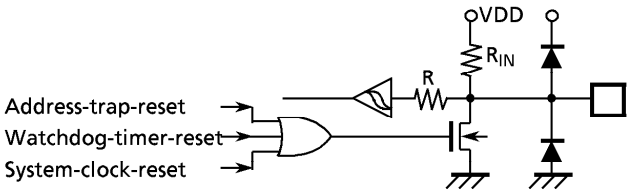
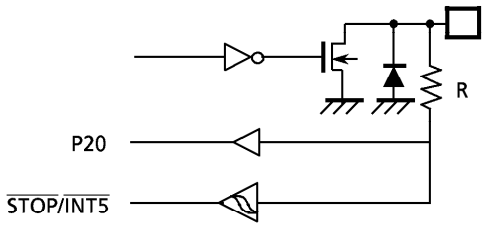
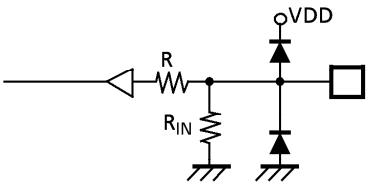
When the resolution in 7-bit mode is 62.5 ns, a 11.5μs pulse is output by setting the following to HPWMDR0.

11.5 μs ÷ 62.5 ns = 184 = B8H

## INPUT/OUTPUT CIRCUITRY

(1) Control pins

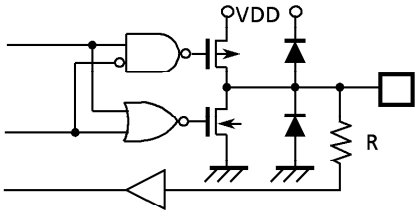
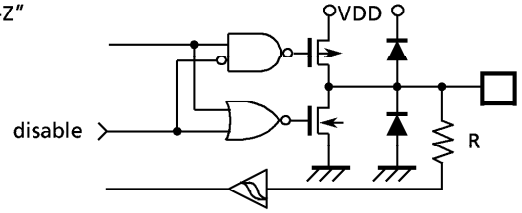
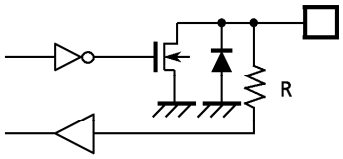
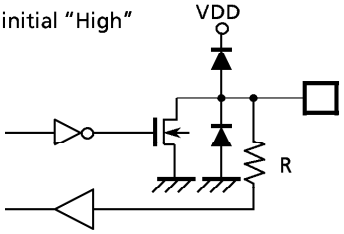
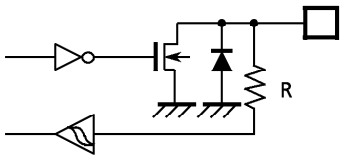
The input/output circuitries of the 87C841/CC41/CH41/CK41/CM41 control pins are shown below, any one of the circuitries can be chosen by a code (NM1 or NM2) as a mask option.

| CONTROL PIN | I/O | INPUT/OUTPUT CIRCUITRY and CODE | REMARKS |
|---|---|---|---|
| XIN<br>XOUT | Input<br>Output |  | Resonator connecting pins<br>(high-frequency)<br>$R_f$ = 1.2 MΩ (typ.)<br>$R_O$ = 1.5 kΩ (typ.)<br>R = 1k Ω (typ.) |
| XTIN (P21)<br>XTOUT (P22) | Input<br>Output | NM1: Refer to port P2<br>NM2:  | Resonator connecting pins<br>(low-frequency)<br>$R_f$ = 6 MΩ (typ.)<br>$R_O$ = 220 kΩ (typ.)<br>R = 1 kΩ (typ.) |
| $\overline{\text{RESET}}$ | I/O |  | Sink open drain output<br>Hysteresis input<br>Pull-up resistor<br>$R_{IN}$ = 220 kΩ (typ.)<br>R = 1 kΩ (typ.) |
| $\overline{\text{STOP/INT5}}$<br>(P20) | Input |  | Hysteresis input<br>R = 1 kΩ (typ.) |
| TEST | Input |  | Pull-down resistor<br>$R_{IN}$ = 70 kΩ (typ.)<br>R = 1 kΩ (typ.) |

*Note 1 : The 87PM41 does not have a pull-down resistor for TEST pin.*
*Note 2 : The input/output circuitries of the 87PM41 are the code NM1 type.*

(2) Input/Output Ports

The input/output circuitries of the 87C841/CC41/CH41/CK41/CM41 input/output ports are shown below.

| PORT | I/O | INPUT / OUTPUT CIRCUITRY and CODE | REMARKS |
|---|---|---|---|
| P0<br>P6<br>P7 | I/O | initial "Hi-Z"<br>  | Tri-state I/O<br>R = 1 kΩ (typ.) |
| P1 | I/O | initial "Hi-Z"<br>disable<br>  | Tri-state I/O<br>Hysteresis input<br>R = 1 kΩ (typ.) |
| P2<br>P3 | I/O | P20, P37~30<br>initial "Hi-Z"<br>  <br>P21. P22<br>initial "High"<br>  | Sink open drain output<br>High current output only P3<br>R = 1 kΩ |
| P4<br>P5 | I/O | initial "Hi-Z"<br>  | Sink open drain output<br>Hysteresis input<br>R = 1 kΩ (typ.) |

## ELECTRICAL CHARACTERISTICS

### ABSOLUTE MAXIMUM RATINGS ($V_{SS}$ = 0 V)

| PARAMETER | SYMBOL | PINS | RATINGS | UNIT |
|---|---|---|---|---|
| Supply Voltage | $V_{DD}$ | | – 0.3 to 6.5 | V |
| Input Voltage | $V_{IN}$ | | – 0.3 to $V_{DD}$ + 0.3 | V |
| Output Voltage | $V_{OUT1}$ | Except sink open drain pin , but include Port P21, P22 and $\overline{RESET}$ | – 0.3 to $V_{DD}$ + 0.3 | V |
| | $V_{OUT2}$ | Sink open drain pin except port P21, P22, $\overline{RESET}$ | – 0.3 to $V_{DD}$ + 0.3 | |
| Output Current (Per 1 pin) | $I_{OUT1}$ | Ports P0, P1, P2, P4, P5, P6, P7 | 3.2 | mA |
| | $I_{OUT2}$ | Port P3 | 30 | |
| Output Current (Total) | $\Sigma$ $I_{OUT1}$ | Ports P0, P1, P2, P4, P5, P6, P7 | 120 | mA |
| | $\Sigma$ $I_{OUT2}$ | Port P3 | 120 | |
| Power Dissipation [Topr = 70 °C] | PD | TMP87C841N/CC41N/CH41N/CK41N/CM41N | 600 | mW |
| | | TMP87C841F/CC41F/CH41F/CK41F/CM41F/U | 350 | |
| Soldering Temperature (time) | Tsld | | 260 (10 s) | °C |
| Storage Temperature | Tstg | | – 55 to 125 | °C |
| Operating Temperature | Topr | | – 40 to 85 | °C |

### RECOMMENDED OPERATING CONDITIONS ($V_{SS}$ = 0 V, Topr = – 40 to 85 °C)

| PARAMETER | SYMBOL | PINS | CONDITIONS | | Min. | Max. | UNIT |
|---|---|---|---|---|---|---|---|
| Supply Voltage | $V_{DD}$ | | fc = 8 MHz | NORMAL1, 2 mode | 4.5 | 5.5 | V |
| | | | | IDLE1, 2 mode | | | |
| | | | fc = 4.2 MHz | NORMAL1, 2 mode | 2.7 | | |
| | | | | IDLE1, 2 mode | | | |
| | | | fs = 32.768 kHz | SLOW mode | | | |
| | | | | SLEEP mode | | | |
| | | | | STOP mode | 2.0 | | |
| Input High Voltage | $V_{IH1}$ | Except hysteresis input | $V_{DD} \geqq 4.5$ V | | $V_{DD} \times 0.70$ | $V_{DD}$ | V |
| | $V_{IH2}$ | Hysteresis input | | | $V_{DD} \times 0.75$ | | |
| | $V_{IH3}$ | | $V_{DD} < 4.5$ V | | $V_{DD} \times 0.90$ | | |
| Input Low Voltage | $V_{IL1}$ | Except hysteresis input | $V_{DD} \geqq 4.5$ V | | 0 | $V_{DD} \times 0.30$ | V |
| | $V_{IL2}$ | Hysteresis input | | | | $V_{DD} \times 0.25$ | |
| | $V_{IL3}$ | | $V_{DD} < 4.5$ V | | | $V_{DD} \times 0.10$ | |
| Clock Frequency | fc | XIN, XOUT | $V_{DD}$ = 4.5 to 5.5 V | | 0.4 | 8.0 | MHz |
| | | | $V_{DD}$ = 2.7 to 5.5 V | | | 4.2 | |
| | fs | XTIN, XTOUT | | | 30.0 | 34.0 | kHz |

## D.C. CHARACTERISTICS

($V_{SS}$ = 0 V, Topr = – 40 to 85 °C)

| PARAMETER | SYMBOL | PINS | CONDITIONS | | Min. | Typ. | Max. | UNIT |
|---|---|---|---|---|---|---|---|---|
| Hysteresis Voltage | $V_{HS}$ | Hysteresis inputs | | | – | 0.9 | – | V |
| Input Current | $I_{IN1}$ | TEST | $V_{DD}$ = 5.5 V<br>$V_{IN}$ = 5.5 V / 0 V | | – | – | ± 2 | μA |
| | $I_{IN2}$ | Open drain ports, Tri-state ports | | | | | | |
| | $I_{IN3}$ | $\overline{RESET}$, $\overline{STOP}$ | | | | | | |
| Input Low Current | $I_{IL}$ | Push pull ports | $V_{DD}$ = 5.5 V, $V_{IN}$ = 0.4 V | | – | – | – 2 | mA |
| Input Resistance | $R_{IN2}$ | $\overline{RESET}$ | | | 90 | 220 | 510 | kΩ |
| Output Leakage Current | $I_{LO}$ | Sink open drain ports | $V_{DD}$ = 5.5 V, $V_{OUT}$ = 5.5 V | | – | – | 2 | μA |
| | | Tri-state ports | $V_{DD}$ = 5.5 V, $V_{OUT}$ = 5.5/0 V | | – | – | ± 2 | |
| Output High Voltage | $V_{OH1}$ | Push pull ports | $V_{DD}$ = 4.5 V, $I_{OH}$ = – 200 μA | | 2.4 | – | – | V |
| | $V_{OH2}$ | Tri-state ports | $V_{DD}$ = 4.5 V, $I_{OH}$ = – 0.7 mA | | 4.1 | – | – | |
| Output Low Voltage | $V_{OL}$ | Except XOUT and P3 | $V_{DD}$ = 4.5 V, $I_{OL}$ = 1.6 mA | | – | – | 0.4 | mA |
| Output Low current | $I_{OL3}$ | P3 | $V_{DD}$ = 4.5 V, $V_{OL}$ = 1.0 V | | – | 20 | – | mA |
| Supply Current in NORMAL 1, 2 modes | $I_{DD}$ | | $V_{DD}$ = 5.5 V<br>$V_{IN}$ = 5.3 V / 0.2 V<br>fc = 8 MHz<br>fs = 32.768 kHz | 87C841/CC41/CH41 | – | 8 | 14 | mA |
| | | | | 87CK41/CM41 | – | 10 | 16 | |
| Supply Current in IDLE 1, 2 modes | | | | 87C841/CC41/CH41 | – | 4 | 6 | mA |
| | | | | 87CK41/CM41 | – | 4.5 | 6 | |
| Supply Current in SLOW mode | | | $V_{DD}$ = 3.0 V<br>$V_{IN}$ = 2.8 V / 0.2 V<br>fs = 32.768 kHz | | – | 30 | 60 | μA |
| Supply Current in SLEEP mode | | | | | – | 15 | 30 | μA |
| Supply Current in STOP mode | | | $V_{DD}$ = 5.5 V<br>$V_{IN}$ = 5.3 V / 0.2 V | | – | 0.5 | 20 | μA |

*Note 1 : Typical values show those at Topr = 25 °C , $V_{DD}$ = 5 V.*
*Note 2 : Input Current $I_{IN1}$,$I_{IN3}$ ; The current through resistor is not included, when the input resistor (pull-upor pull-down) is contained.*
*Note 3 : IDD except $I_{REF}$.*

## A/D CONVERSION CHARACTERISTICS

(Topr = – 40 to 85 °C)

| PARAMETER | SYMBOL | CONDITIONS | Min. | Typ. | Max. | | | UNIT |
|---|---|---|---|---|---|---|---|---|
| | | | | | ADCDR1 | ADCDR2 | | |
| | | | | | | ACK = 0 | ACK = 1 | |
| Analog Reference Voltage | $V_{AREF}$ | $V_{AREF}$ – $V_{ASS}$ ≧ 2.5 V | 2.7 | — | $V_{DD}$ | | | V |
| | $V_{ASS}$ | | $V_{SS}$ | — | 1.5 | | | |
| Analog Input Voltage | $V_{AIN}$ | | $V_{ASS}$ | — | $V_{AREF}$ | | | V |
| Analog Supply Current | $I_{REF}$ | $V_{AREF}$ = 5.5 V,<br>$V_{ASS}$ = 0.0 V | — | 0.5 | 1 | | | mA |
| Nonlinearity Error | | $V_{DD}$ = 5.0, $V_{SS}$ = 0.0 V<br>$V_{AREF}$ = 5.000 V<br>$V_{ASS}$ = 0.000 V<br>or<br>$V_{DD}$ = 2.7, $V_{SS}$ = 0.0 V<br>$V_{AREF}$ = 2.700 V<br>$V_{ASS}$ = 0.000 V | — | — | 1.0 | | | LSB |
| Zero Point Error | | | — | — | ± 1 | ± 3 | ± 2 | |
| Full Scale Error | | | — | — | ± 1 | ± 3 | ± 2 | |
| Total Error | | | — | — | ± 2 | ± 6 | ± 4 | |

*Note 1 : $\triangle V_{AREF} = V_{AREF} - V_{ASS}$*
*Note 2 : ADCDR1 ; 8 bit – A/D conversion result (1LSB = $\triangle V_{AREF}$ / 256)*
*ADCDR2 ; 10 bit – A/D conversion result (1LSB = $\triangle V_{AREF}$ / 1024)*

## A.C. CHARACTERISTICS

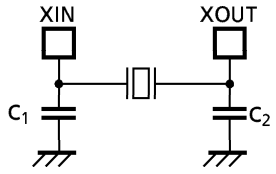
($V_{SS}$ = 0 V, $V_{DD}$ = 2.7 / 4.5 to 5.5 V, Topr = – 40 to 85 °C)

| PARAMETER | SYMBOL | CONDITIONS | Min. | Typ. | Max. | UNIT |
|---|---|---|---|---|---|---|
| Machine Cycle Time | $t_{cy}$ | In NORMAL1, 2 modes | 0.5 | – | 10 | μs |
| | | In IDLE1, 2 modes | | | | |
| | | In SLOW mode | 117.6 | – | 133.3 | |
| | | In SLEEP mode | | | | |
| High Level Clock Pulse Width | $t_{WCH}$ | For external clock operation (XIN input), fc = 8 MHz | 62.5 | – | – | ns |
| Low Level Clock Pulse Width | $t_{WCL}$ | | | | | |
| High Level Clock Pulse Width | $t_{WSH}$ | For external clock operation (XTIN input), fs = 32.768 kHz | 14.7 | – | – | μs |
| Low Level Clock Pulse Width | $t_{WSL}$ | | | | | |

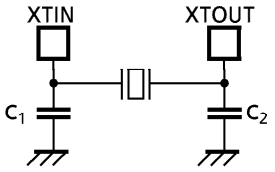
## RECOMMENDED OSCILLATING CONDITIONS

($V_{SS}$ = 0 V, $V_{DD}$ = 2.7 / 4.5 to 5.5 V, Topr = – 40 to 85 °C)

| PARAMETER | Oscillator | Oscillation Frequency | Recommended Oscillator | Recommended Constant $C_1$ | Recommended Constant $C_2$ |
|---|---|---|---|---|---|
| High-frequency Oscillation | Ceramic Resonator | 8 MHz | KYOCERA KBR8.0M | 30 pF | 30 pF |
| | | 4 MHz | KYOCERA KBR4.0MS | | |
| | | | MURATA CSA4.00MG | | |
| | Crystal Oscillator | 8 MHz | TOYOCOM 210B 8.0000 | 20 pF | 20 pF |
| | | 4 MHz | TOYOCOM 204B 4.0000 | | |
| Low-frequency Oscillation | Crystal Oscillator | 32.768 kHz | NDK MX-38T | 15 pF | 15 pF |




(1) High-frequency Oscillation




(2) Low-frequency Oscillation

*Note : When it is used in high electrical field, an electrical shield of the package is recommended to retain normal operations.*