CMOS 8-BIT MICROCONTROLLER

TMP87C444N, TMP87C844N

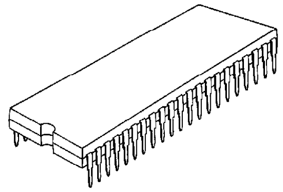
The 87C444/844 is the high speed and high performance 8-bit single chip microcomputer. This MCU contains CPU core, ROM, RAM, input/output ports, two multi-function timer/counters, serial bus interface, 8-bit D/A conversion outputs and 8-bit A/D conversion inputs on a chip.

PART No.	ROM	RAM	PACKAGE	OTP MCU
TMP87C444N	4K bytes	256 bytes	SDIP42-P-600-1.78	TMP87P844N
TMP87C844N	8K bytes			

FEATURES

- 8-bit single chip microcomputer TLCS-870 Series
- Instruction execution time : 0.5 μs (at 8 MHz)
- 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)
- 10 interrupt sources (External : 3, Internal : 7)
  - All sources have independent latches each, and nested interrupt control is available.
  - 3 edge-selectable external interrupts with noise reject
  - High-speed task switching by register bank changeover
- 5 Input/Output ports (34 pins)
- Two 16-bit Timer/Counters
  Timer, Event counter, Programmable Pulse Generator output
  Pulse width measurement, External trigger timer, Window modes
- Time Base Timer (Interrupt frequency : 1 Hz to 15625 Hz)
- Divider output function (frequency : 1 kHz to 8 kHz)
- Watchdog Timer
  - Interrupt source/reset output (programmable)
- Serial bus Interface
  - I^{2}C-bus, 8-bit SIO modes : 1 channel
  - 8-bit SIO : 1 channel
- 8-bit D/A converter
  - 8 analog outputs
  - Builtin OP-Amp.
- 8-bit successive approximate type A/D converter with sample and hold
  - 4 analog inputs
  - Conversion time : 23 μs at 8 MHz
- Power saving operating modes
  - IDLE mode : CPU stops, and Peripherals operate. Release by interrupts.
- Operating voltage : 4.5~5.5 V at 8 MHz
- Emulation Pod : BM87C844N0A

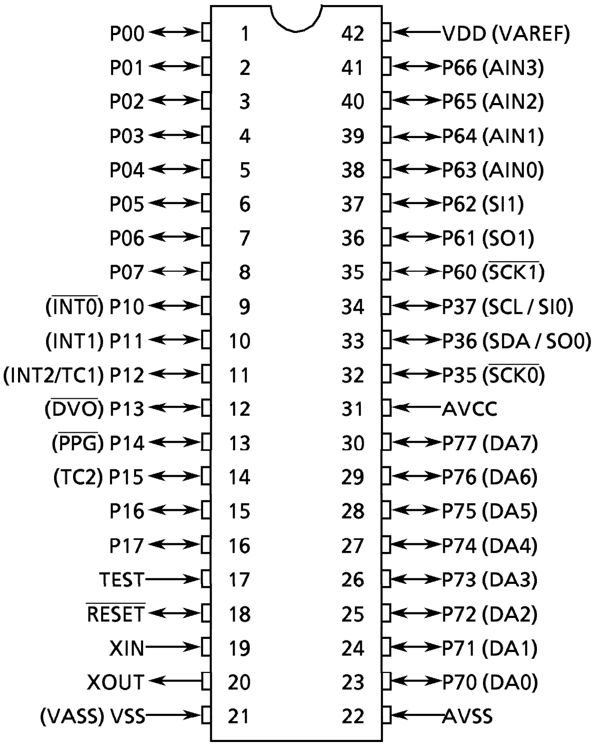
SDIP42-P-600-1.78



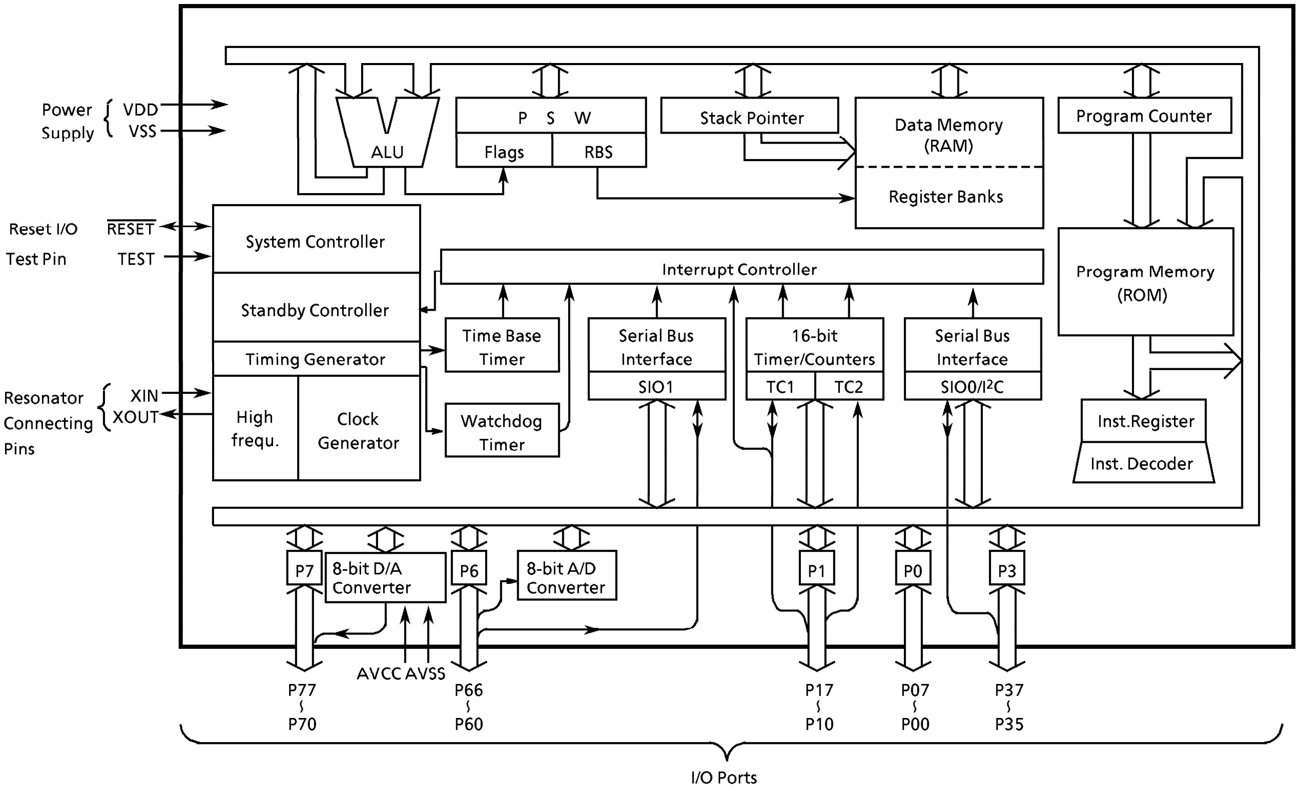
TMP87C844N
TMP87C444N
TMP87P844N

PIN ASSIGNMENTS (TOP VIEW)

SDIP42-P-600-1.78



BLOCK DIAGRAM



PIN FUNCTION

PIN NAME	Input/Output	Function	
P07~P00	I/O	Two 8-bit programmable input/output ports (tri-state) Each bit of these ports can be individually configured as an input or an output under software control. During reset, all bits are configured as inputs. 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
P37 (SCL/SI0)	I/O (I/O/Input)	3-bit input/output port with latch. When used as an input port or a SBI input/output, the latch must be set to "1".	I^2C bus serial clock input/output or SIO0 serial data input
P36 (SDA/SO0)	I/O (I/O/Output)		I^2C bus serial data input/output or SIO0 serial data output
P35 ($\overline{SCK0}$)	I/O (I/O)		SIO0 serial clock input/output
P66 (AIN3) ~P63 (AIN0)	I/O (Input)	7-bit programmable input/output port (tri-state). Each bit of this port can be individually configured as an input or an output under software control. When used as a SIO1 input/output the latch must be set to "1". When used as an analog input, select analog input enable in the ADCCR.	A/D converter analog inputs
P62 (SI1)	I/O (Input)		SIO1 serial data input
P61 (SO1)	I/O (Output)		SIO1 serial data output
P60 ($\overline{SCK1}$)	I/O (I/O)		SIO1 serial clock input/output
P77 (DA7) ~P70 (DA0)	I/O (Output)	8-bit programmable input/output port (tri-state). Each bit of this port can be individually configured as an input or an output under software control. (Only the case of the DACCR1 = "1", I/O contorol can be available). When used as an analog output, the DACCR1 must be set to "0".	D/A converter analog outputs
XIN, XOUT	Input, Output	Resonator connecting pins for high-frequency clock. 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 (VAREF)	Power Supply	+ 5V	Analog reference voltage input for the A/D converter (High)
VSS (VASS)		0V (GND)	Analog reference voltage input for the A/D converter (Low)
AVCC		Analog reference voltage input for the D/A converter (High)	
AVSS		Analog reference voltage input for the D/A converter (0V)	

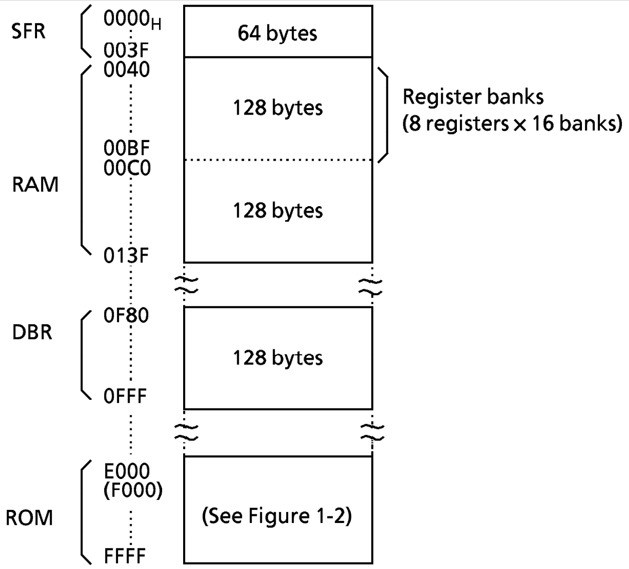
OPERATIONAL DESCRIPTION

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 87C444/844. 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.



Note : ROM ; Read Only Memory includes:
Program memory
RAM ; Random Access Memory includes:
Data memory
Stack
General-purpose register banks
SFR ; Special Function Register includes:
I/O ports
Peripheral control registers
Peripheral status registers
System control registers
Interrupt control registers
Program Status Word
DBR ; Data Buffer Register includes:
SIO1 Data Buffer
D/A Converter Data Register

Figure 1-1. Memory Address Map

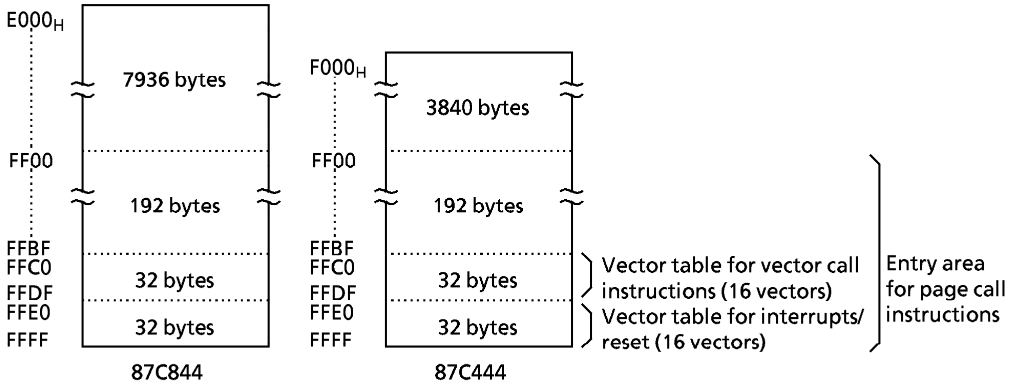


Figure 1-2. ROM Address Maps

1.2 Program Memory (ROM)

The 87C444 has a 4Kbytes (addresses F000$_H$-FFFF$_H$) and the 87C844 has a 8Kbytes (addresses E000$_H$-FFFF$_H$) of program memory (mask programmed ROM).
Addresses FF00$_H$-FFFF$_H$ in the program memory can also be used for special purposes.

(1) **Interrupt / Reset** vector table (addresses FFE0$_H$-FFFF$_H$)
This table consists of a reset vector and 15 interrupt vectors (2 bytes/vector). These vectors store a reset start address and 15 interrupt service routine entry addresses.

(2) Vector table for **vector call** instructions (addresses FFC0$_H$-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$_H$-FFFF$_H$) for **page call** instructions
This is the subroutine entry address area for the page call instructions [CALLP n]. Addresses FF00$_H$-FFBF$_H$ are normally used because address FFC0$_H$-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-byte space.

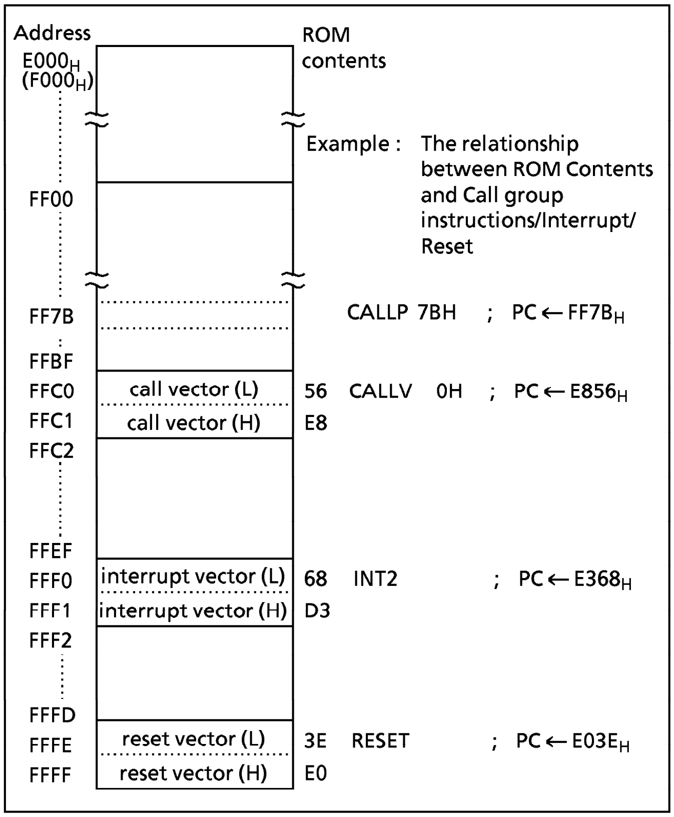


Figure 1-3. Program Memory Map

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.

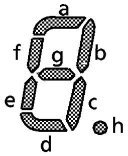
Example 1 : Loads the ROM contents at the address specified by the HL register pair contents into the accumulator (HL ≧ $E000_H$ for 87C844):

```
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 P5 after executing the following program:

```
          ADD       A, TABLE – $ – 4          ;  P5←ROM (TABLE + A)
          LD        (P5), (PC + A)
          JRS       T, SNEXT
TABLE :   DB        0C0H, 0F9H, 0A4H, 0B0H, 99H, 92H, 82H, 0D8H, 80H, 98H
SNEXT :
```

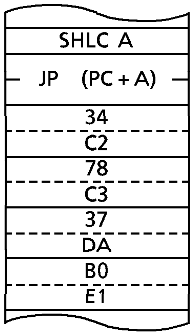
Notes : "$" is a header address of ADD instruction.
DB is a byte data difinition instruction.



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
```

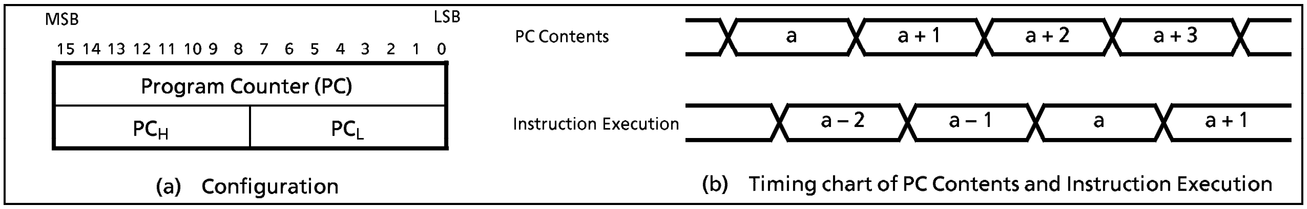
Note : DW is a word data definition instruction.



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 $E0_H$ and $3E_H$ are stored at addresses $FFFF_H$ and $FFFE_H$, respectively, the execution starts from address $E03E_H$ 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 $E123_H$ is being executed, the PC contains $E125_H$.



(a) Configuration

(b) Timing chart of PC Contents and Instruction Execution

Figure 1-4. Program Counter

1.4 Data Memory (RAM)

The 87C444/844 has a 256 bytes (addresses 0040_H-$013F_H$) of data memory (static RAM). Figure 1-5 shows the data memory map.

Addresses 0000_H-$00FF_H$ are used as a direct addressing area to enhance instructions which utilize this addressing mode; therefore, addresses 0040_H-$00FF_H$ 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-$00BF_H$. 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".

The TLCS-870 Series cannot execute programs placed in the data memory. When the program counter indicates a data memory address, a bus error occurs and an address-trap-reset applies. The $\overline{\text{RESET}}$ pin goes low during the address-trap-reset.

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-purpose registers are mapped in the RAM ; therefore, *do not clear RAM at the current bank addresses.*

Example : Clears RAM to “00_H” except the bank 0:

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

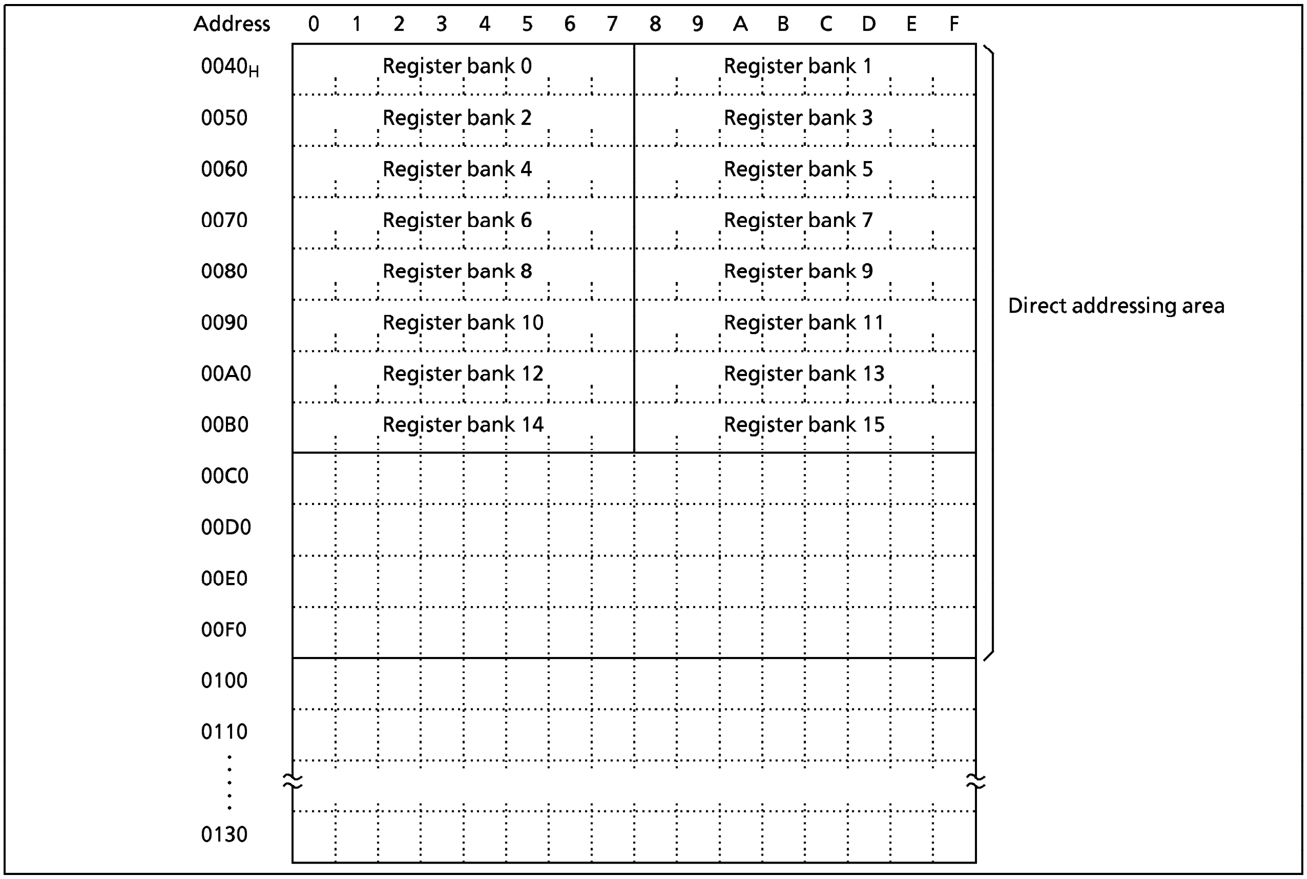
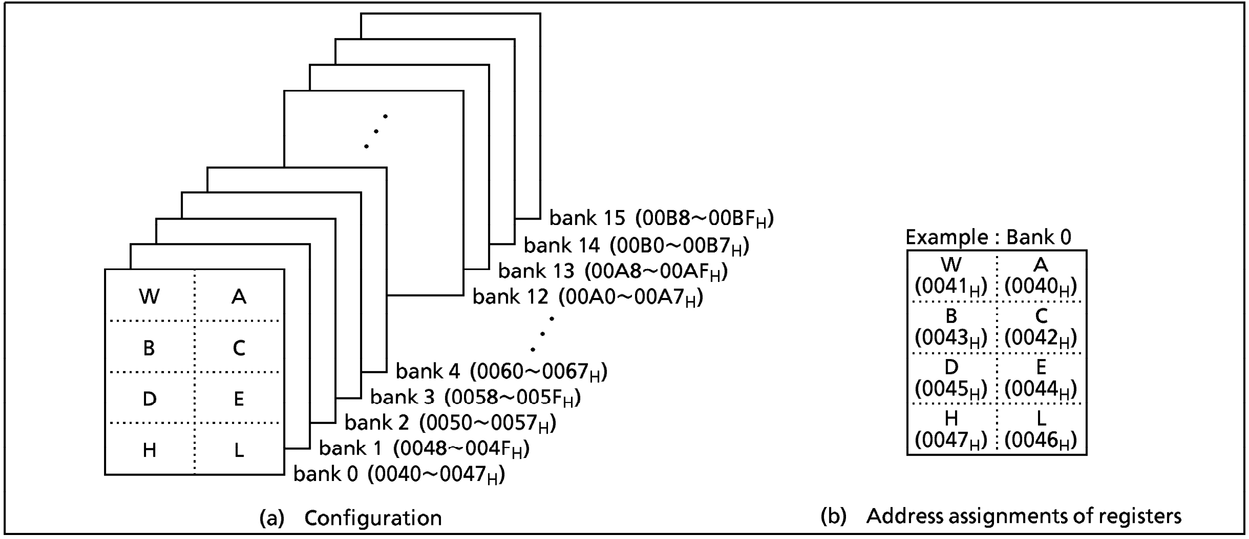


Figure 1-5. Data Memory Map

1.5 General-purpose Register Banks

General-purpose registers are mapped into addresses 0040_{H}-$00BF_{H}$ in the data memory as shown in Figure 1-5. There are 16 register banks, and each bank contains eight 8-bit registers W, A, B, C, D, E, H, and L. Figure 1-6 shows the general-purpose register bank configuration.



(a) Configuration

(b) Address assignments of registers

Figure 1-6. 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 1234_{H} 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 52_{H} 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] and [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 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".

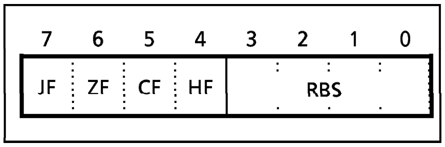


Figure 1-7. PSW (Flags, RBS) Configuration

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 (quotient 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 1-9 shows the stacking order.

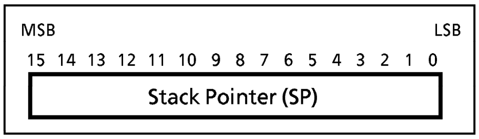


Figure 1-8. Stack Pointer

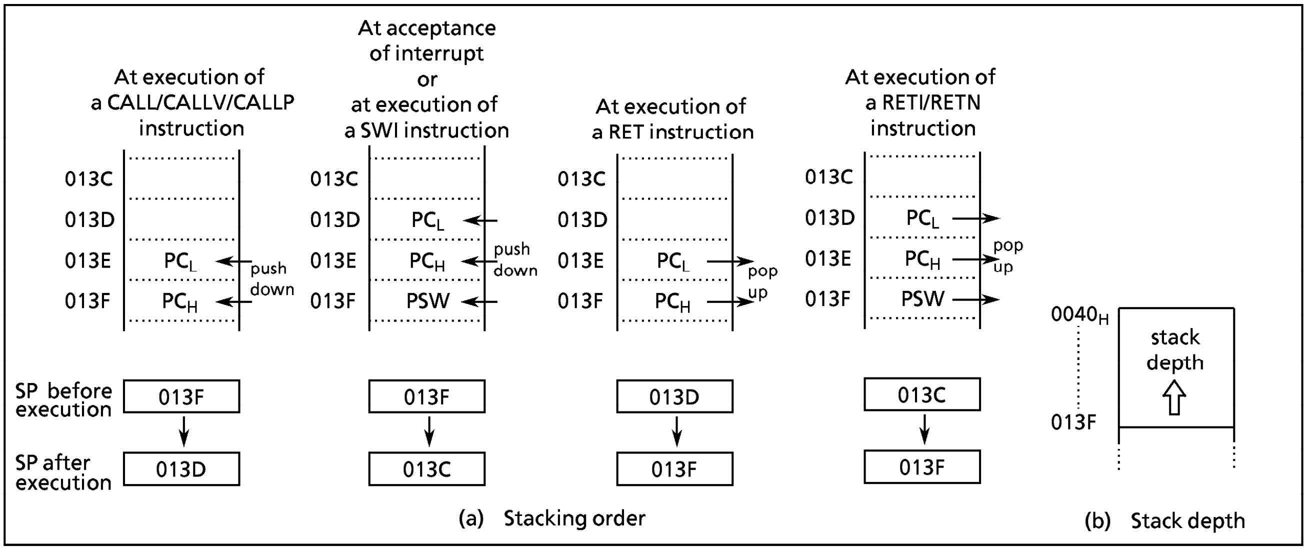
The SP is not initialized hardware-wise but requires initialization by an initialize routine (sets the highest stack address). [LD SP, mn], [LD SP, gg] and [LD gg, SP] are the SP access instructions (mn ; 16-bit immediate data, gg ; register pair).

Example 1 : To initialize the SP

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

Example 2 : To read the SP

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


At execution of a CALL/CALLV/CALLP instruction
At acceptance of interrupt or at execution of a SWI instruction
At execution of a RET instruction
At execution of a RETI/RETN instruction
013C
013D
013E
013F
PC$_L$
PC$_H$
PSW
push down
pop up
SP before execution
SP after execution
013F
013D
013F
013C
013D
013F
013C
013F
0040$_H$
013F
stack depth

(a) Stacking order (b) Stack depth

Figure 1-9. Stack

1.8 System Clock Controller

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

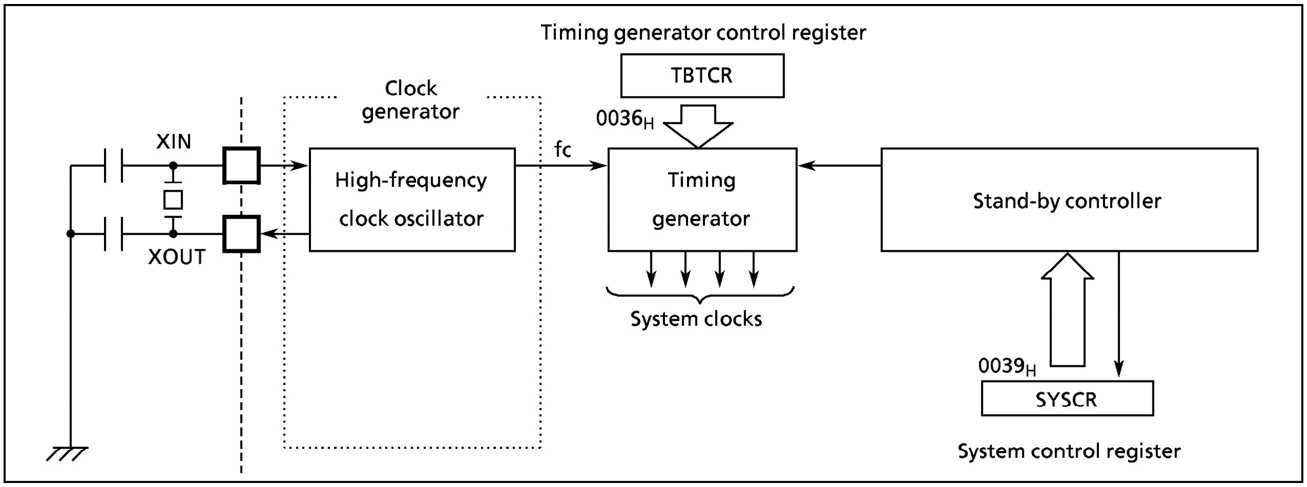


Figure 1-10. System Clock Controller

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 a oscillation circuit for the high-frequency clock.

The high-frequency (fc) clock can be easily obtained by connecting a resonator between the XIN/XOUT pins, respectively. Clock input from an external oscillator is also possible. In this case, external clock is applied to the XIN pin with the XOUT pin not connected. The 87C444/844 is not provided an RC oscillation.

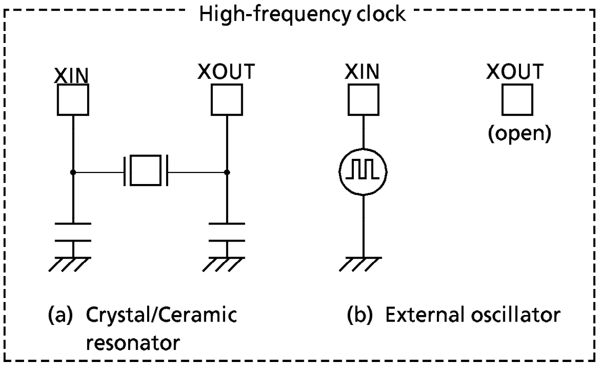


Figure 1-11. Examples of Resonator Connection

Note : Accurate Adjustment of the Oscillation Frequency:
Although hardware to externally and directly monitor the basic clock pulse is not provided, the oscillation frequency can be adjusted by providing a 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.

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 ($\overline{\text{DVO}}$) 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 – TC2)
⑥ Generation of internal clocks for serial bus interface (SIO1)
⑦ 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, shown in Figure 1-12 as follows. During reset , the divider is cleared to “0”, however, the prescaler is not cleared.

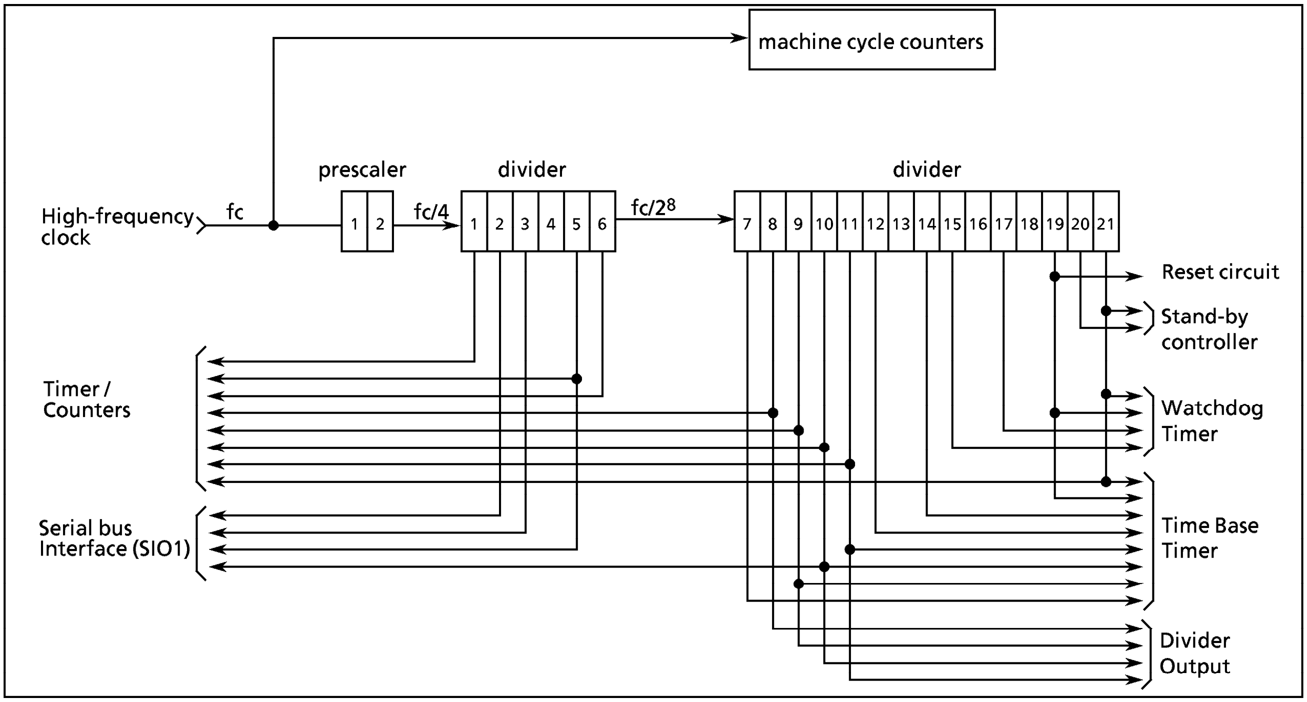


Figure 1-12. Configuration of Timing Generator

(2) Machine Cycle

Instruction execution and peripherals 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 - S3), and each state consists of one main system clock.

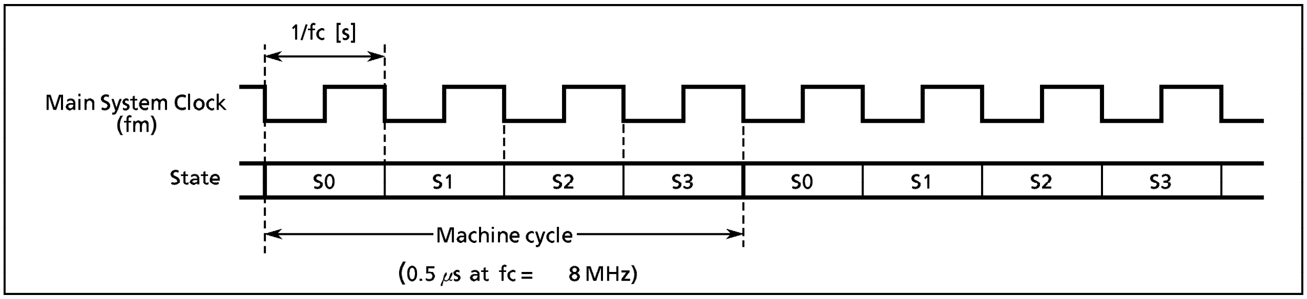


Figure 1-13. Machine Cycle

1.8.3 Stand-by Controller

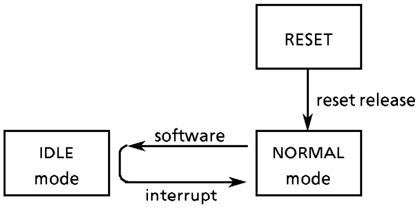
87C444/844 has IDLE mode only as stand-by mode. Setting the system control register (SYSCR) changes the operation mode from NORMAL to IDLE.
Figure 1-14 shows the operating mode transition diagram and Figure 1-15 shows the system control register.

(1) **Operating mode**

① NORMAL mode
In this mode, both the CPU core and on-chip peripherals operate.

② IDLE mode

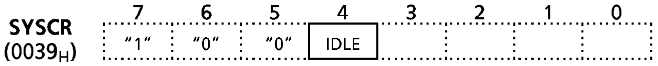
In this mode, the internal oscillation circuit remains active. The CPU and the watchdog timer are halted; however, on-chip peripherals remain active. IDLE mode is started by setting IDLE bit in the system control register (SYSCR), and IDLE mode is released to NORMAL mode by an interrupt request from 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 next instruction which follows IDLE mode start instruction.



Operating mode	Frequency	CPU core	On-chip Peripherals	Machine cycle time
	High-frequency			
RESET	turning on oscillation	reset	reset	4/fc [s]
NORMAL		operate	operate	
IDLE		halt		

Figure 1-14. Operating Mode Transition Diagram

System Control Register



(Initial value: 1000 ****)

IDLE	IDLE mode start	0 : CPU and watchdog timer remain active 1 : CPU and watchdog timer are stopped (start IDLE mode)	R/W

Note 1 : A reset is applied ($\overline{\text{RESET}}$ pin output goes low) if both bit 7 and bit 6 in SYSCR are cleared to "0".

Note 2 : Do not clear bit 7 in SYSCR to "0" , and do not set bits 6-5 in SYSCR to "1" .

*Note 3 : * ; don't care*

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

Figure 1-15. System Control Register

1.8.4 Operating Mode Control

(1) **IDLE** mode

IDLE mode is controlled by the system control register 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            (SYSCR) . 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 to NORMAL mode.

a. Normal release mode (IMF = "0")

IDLE mode is released by any interrupt source enabled by the individual interrupt enable flag (EF). Execution resumes with the instruction following the IDLE mode start instruction (e.g. [SET (SYSCR).4]). Normally, IL (Interrupt Latch) of interrupt source to release IDLE mode must be cleared by load instructions.

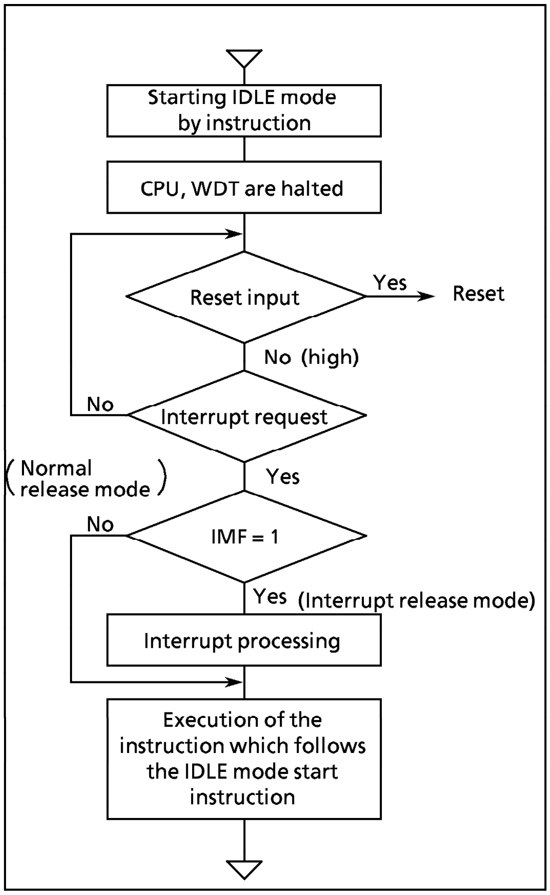

Starting IDLE mode by instruction
CPU, WDT are halted
Reset input
Yes
Reset
No (high)
No
Interrupt request
Normal release mode
Yes
No
IMF = 1
Yes (Interrupt release mode)
Interrupt processing
Execution of the instruction which follows the IDLE mode start instruction

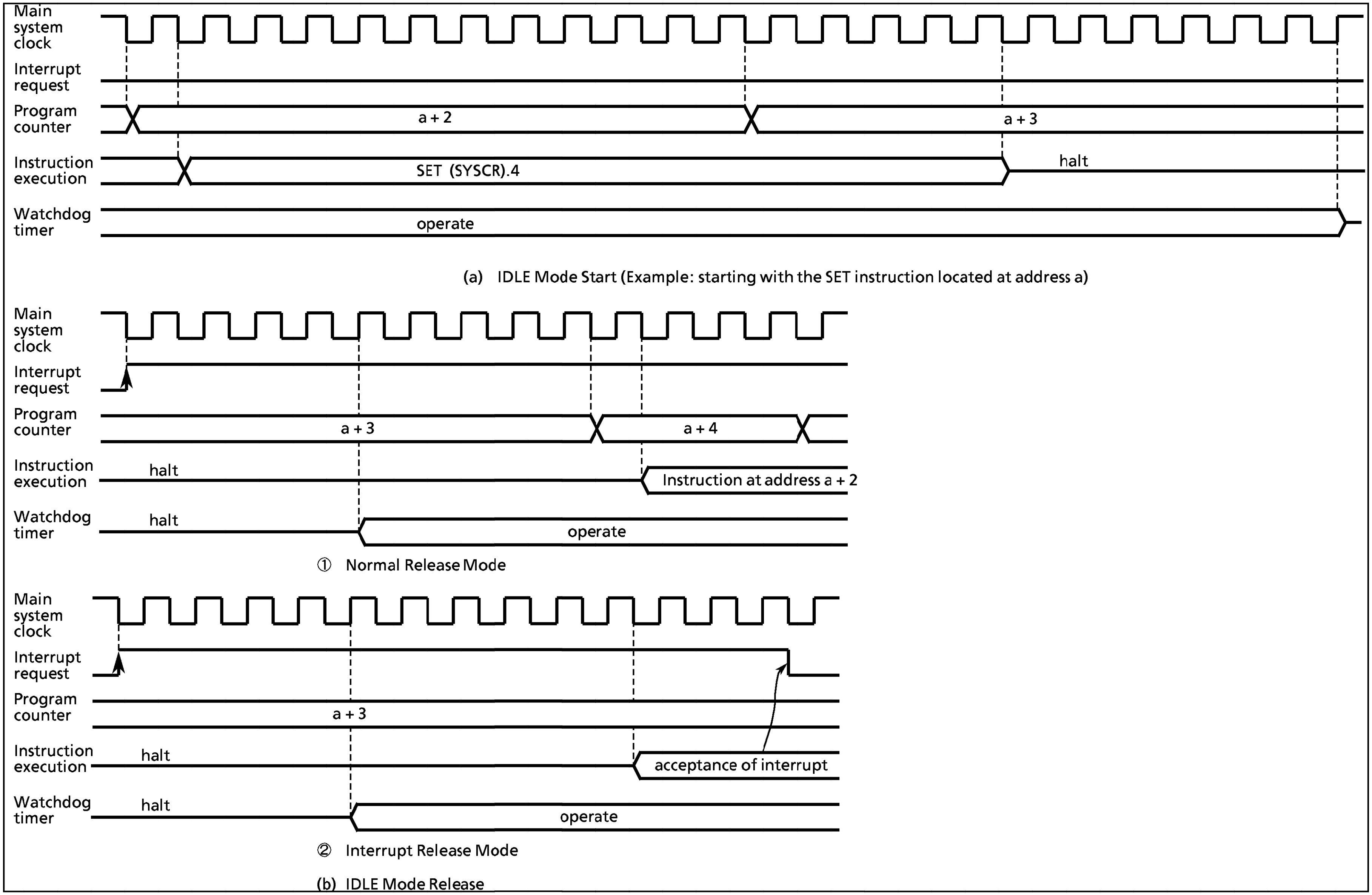
Figure 1-16. IDLE Mode

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). 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 87C444/844 are placed in NORMAL mode.

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



(a) IDLE Mode Start (Example: starting with the SET instruction located at address a)

① Normal Release Mode

② Interrupt Release Mode

(b) IDLE Mode Release

Figure 1-17. IDLE Mode Start/Release

1.9 Interrupt Controller

The 87C444/844 has a total of 10 interrupt sources: 3 externals and 7 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 1-18 shows the interrupt controller.

Table 1-1. Interrupt Sources

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 1)	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
	reserved	IMF · EF_8 = 1	IL_8	FFEE$_H$	8
Internal	INTSBI (Serial bus Interface interrupt)	IMF · EF_9 = 1	IL_9	FFEC$_H$	9
	reserved	IMF · EF_{10} = 1	IL_{10}	FFEA$_H$	10
	reserved	IMF · EF_{11} = 1	IL_{11}	FFE8$_H$	11
	reserved	IMF · EF_{12} = 1	IL_{12}	FFE6$_H$	12
Internal	INTSIO (SIO 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
	reserved	IMF · EF_{15} = 1	IL_{15}	FFE0$_H$	Low 15

(1) **Interrupt Latches** ($IL_{15\sim2}$)

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), 1011110100111111B ; IL_{14}, IL_9, IL_7, IL_6←0

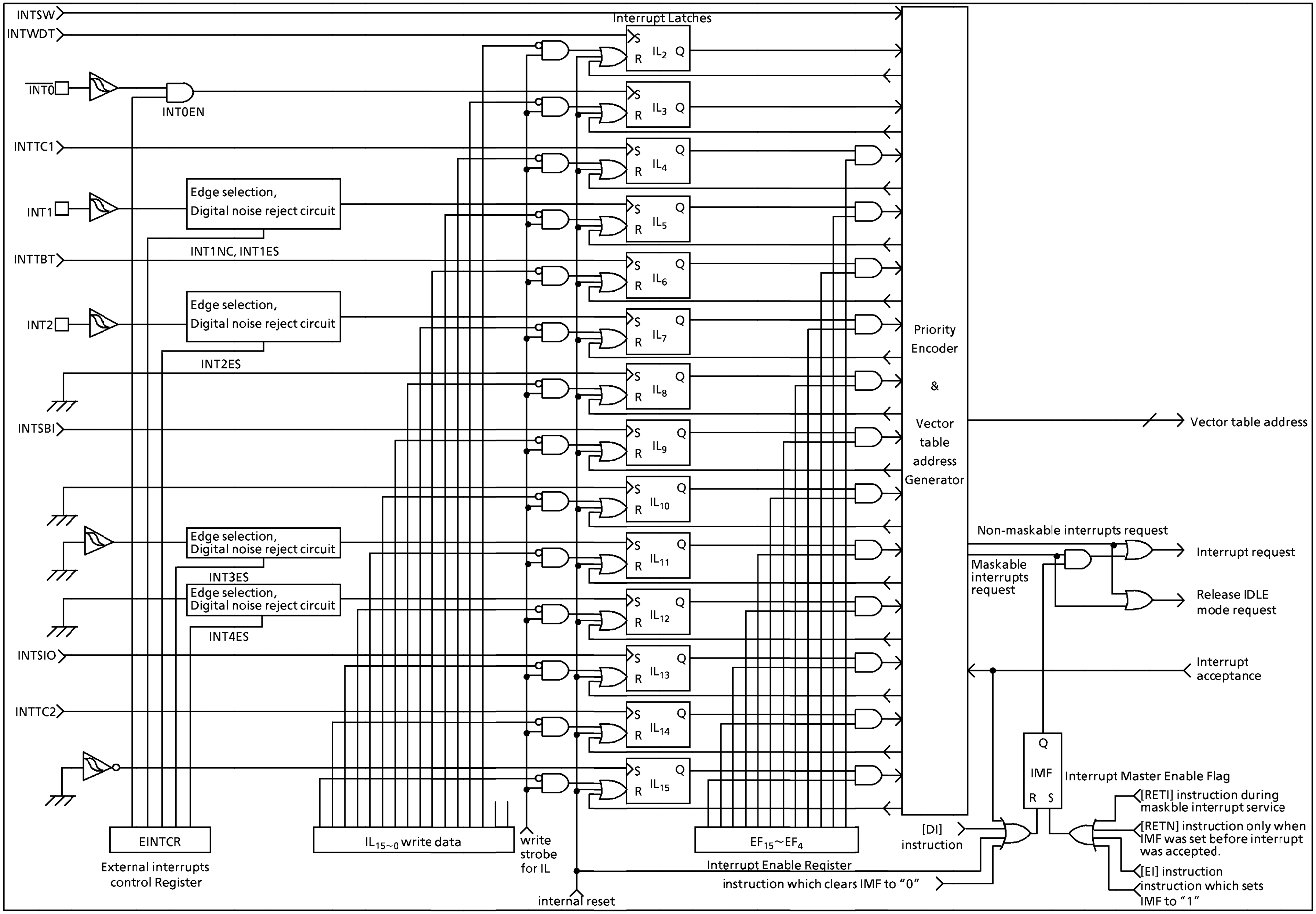

INTSW
INTWDT
INT0
INT0EN
INTTC1
INT1
Edge selection,
Digital noise reject circuit
INT1NC, INT1ES
INTTBT
INT2
Edge selection,
Digital noise reject circuit
INT2ES
INTSBI
Edge selection,
Digital noise reject circuit
INT3ES
Edge selection,
Digital noise reject circuit
INT4ES
INTSIO
INTTC2
EINTCR
External interrupts control Register
Interrupt Latches
S R IL2 Q
S R IL3 Q
S R IL4 Q
S R IL5 Q
S R IL6 Q
S R IL7 Q
S R IL8 Q
S R IL9 Q
S R IL10 Q
S R IL11 Q
S R IL12 Q
S R IL13 Q
S R IL14 Q
S R IL15 Q
IL15~0 write data
write strobe for IL
internal reset
EF15~EF4
Interrupt Enable Register
instruction which clears IMF to "0"
Priority Encoder & Vector table address Generator
Vector table address
Non-maskable interrupts request
Maskable interrupts request
Interrupt request
Release IDLE mode request
Interrupt acceptance
Q IMF R S
Interrupt Master Enable Flag
[DI] instruction
[RETI] instruction during maskble interrupt service
[RETN] instruction only when IMF was set before interrupt was accepted.
[EI] instruction
instruction which sets IMF to "1"

Figure 1-18. Interrupt Controller Block Diagram

Example 2 : Reads interrupt latches

```
        LD      WA, (IL)                ; W←ILH, A←ILL
```

Example 3: Tests an interrupt latch

```
        TEST    (ILH).4                 ; if IL12 = 1 then jump
        JR      F, SSET
```

(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 the 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. The IMF is normally set and cleared by the [EI] and [DI] instructions, and the IMF is initialized to "0" during reset.

② **Individual interrupt Enable Flags** (EF$_{15}$~EF$_4$)

These flags enable and disable the acceptance of individual maskable interrupts. 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), 1110100000000001B    ;   EF15~EF13, EF11, IMF←1
```

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IL (003C, 003D$_H$)	IL$_{15}$	IL$_{14}$	IL$_{13}$	IL$_{12}$	IL$_{11}$	IL$_{10}$	IL$_9$	IL$_8$	IL$_7$	IL$_6$	IL$_5$	IL$_4$	IL$_3$	IL$_2$		
	IL$_H$ (003D$_H$)								IL$_L$ (003C$_H$) (Initial Value : 00000000 000000**)							
EIR (003A, 003B$_H$)	EF$_{15}$	EF$_{14}$	EF$_{13}$	EF$_{12}$	EF$_{11}$	EF$_{10}$	EF$_9$	EF$_8$	EF$_7$	EF$_6$	EF$_5$	EF$_4$				IMF
	EIR$_H$ (003B$_H$)								EIR$_L$ (003A$_H$) (Initial Value : 00000000 0000***0)							

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 1-19. 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 μs at fc = 8MHz 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).

(1) Interrupt acceptance processing is as follows:

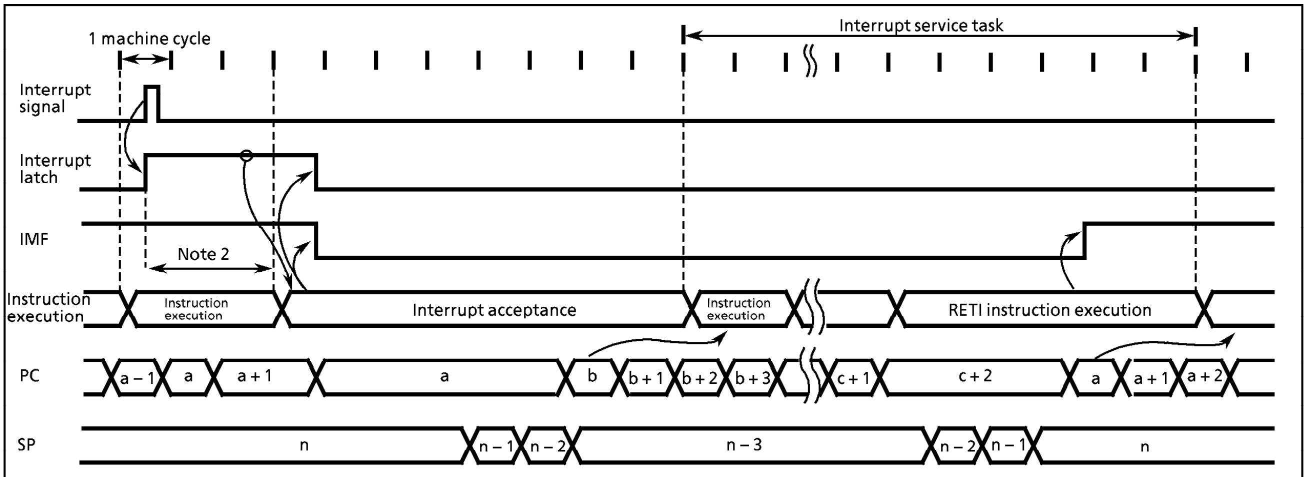
① The interrupt master enable flag (IMF) is cleared to "0" to temporarily disable the acceptance of any following maskable interrupts. When a non-maskable interrupt is accepted, the acceptance of any following interrupts is temporarily disabled.

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. The stack pointer is decremented 3 times.

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.

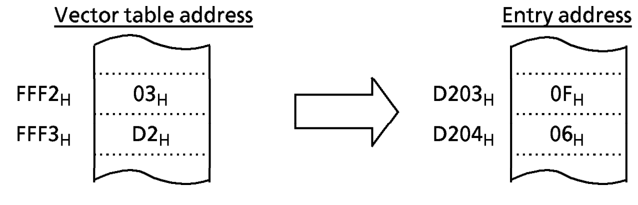


Note1 : a ; return address, b ; entry address, c ; address when the RETI instruction is stored

Note2 : The maximum response time from when an IL is set until an interrupt acceptance processing starts is 38/fc [s].

Figure 1-20. 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.

(2) Saving/Restoring General-purpose Register

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:

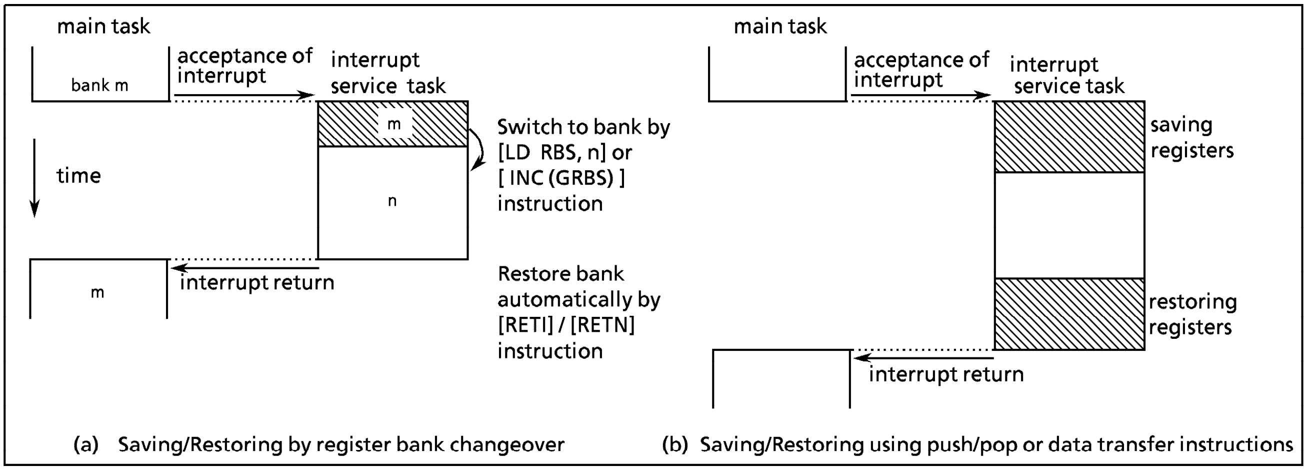
① General-purpose register save/restore by register bank changeover:

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
```



(a) Saving/Restoring by register bank changeover (b) Saving/Restoring using push/pop or data transfer instructions

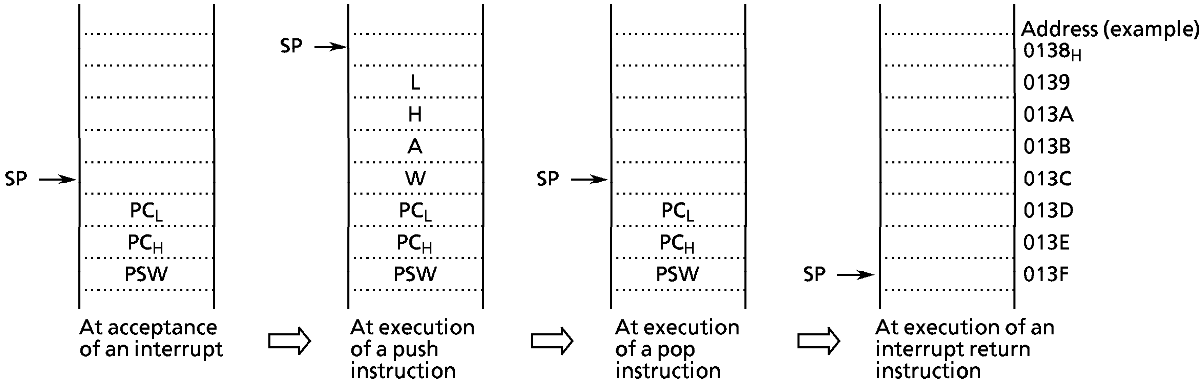
Figure 1-21. Saving/Restoring General-purpose Registers

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


SP
PC$_L$
PC$_H$
PSW
At acceptance of an interrupt
SP
L
H
A
W
PC$_L$
PC$_H$
PSW
At execution of a push instruction
SP
PC$_L$
PC$_H$
PSW
At execution of a pop instruction
SP
Address (example)
0138$_H$
0139
013A
013B
013C
013D
013E
013F
At execution of an interrupt return instruction

③ General-purpose registers save/restore using data transfer instructions:
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
```

(3) 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. ② The stack pointer is incremented 3 times. ③ The interrupt master enable flag is set to "1".	① The contents of the program counter and program status word are restored from the stack. ② The stack pointer is incremented 3 times. 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 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.

Note : At the development tool, if processing of a non-maskable interrupt is already underway, executing the SWI instruction will generate a software interrupt as a software brake.

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. the address trap reset is generated in case that an instruction is fetched from RAM or SFR areas.

Note : The fetch data from addresses $BF80_H$ to $BFFF_H$ (test ROM area) is not "FF_H".

② Debugging
Debugging efficiency can be increased by placing the SWI instruction at the software break point setting address.

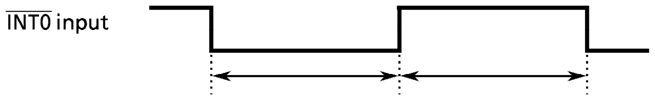
1.9.3 External Interrupts

The 87C444/844 each have three external interrupt inputs ($\overline{\text{INT0}}$, INT1, INT2). Two 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 and INT2.
The $\overline{\text{INT0}}$/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 $\overline{\text{INT0}}$/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 $\overline{\text{INT0}}$ pin input is detected.

Table 1-2. External Interrupts

Source	Pin	Secondary function pin	Enable conditions	Edge	Digital noise reject
INT0	$\overline{\text{INT0}}$	P10	IMF = 1, INT0EN = 1	falling edge	— (hysteresis input)
INT1	INT1	P11	$IMF \cdot EF_5 = 1$	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	$IMF \cdot EF_7 = 1$		Pulses of less than 7/fc [s] are eliminated as noise. Pulses equal to or more than 24/fc [s] are regareded as signals.

Note 1 : The noise rejection function is also affected for timer/counter input (TC1 pin).
Note 2 : The pulse width (both "H" and "L" level) for input to the $\overline{\text{INT0}}$ pin must be over 1 machine cycle.



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

Note 3 : If a noiseless signal is input to the external interrupt pin, 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 pin	25/fc [s]	

EINTCR
(0037_H)

7	6	5	4	3	2	1	0
INT1 NC	INT0 EN				INT2 ES	INT1 ES	

(Initial value : 00** *00*)

INT1NC	Noise reject time select	0 : Pulses of less than 63/fc [s] are eliminated as noise 1 : Pulses of less than 15/fc [s] are eliminated as noise	R/W
INT0EN	P10/$\overline{\text{INT0}}$ pin configuration	0 : P10 input/output port 1 : $\overline{\text{INT0}}$ pin (Port P10 should be set to an input mode)	
INT2 ES INT1 ES	INT2, INT1 edge select	0 : Rising edge 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 during NORMAL mode, clear interrupt latches of external interrupt inputs (INT2) after 8 machine cycles from the time of rewriting.*

Note 5 : *In order to change an edge of timer counter input by rewriting the contents of INT2ES during NORMAL mode, rewrite the contents after timer counter is stopped (TC*s = 0), that is, interrupt disable state. Then, clear interrupt latches of external interrupt inputs (INT2) after 8 machine cycles from the time of rewriting to change to interrupt enable state. Finally, start timer counter.*

Example : When changing TC1 pin inputs edge in external trigger timer mode from rising edge to falling edge. (example : TMP87C844N)

```
                   LD (TC1CR),01001000B    ;  TC1S ← 0 (stops TC1)
                   DI                      ;  IMF ← 0 (disables interrupt service)
                   LD (EINTCR),00000100B   ;  INT2ES ← 1 (change edge selection)
 ______↑______     NOP
   8 machine       ~
     cycles        NOP
 ______↓______     LD (ILL),01111111B      ;  IL7 ← 0 (clears interrupt latch)
                   EI                      ;  IMF ← 1 (enables interrupt service)
                   LD (TC1CR),01111000B    ;  TC1S ← 1 (starts TC1)
```

Note 6 : *If changing the contents of INT1ES during NORMAL 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.*

Figure 1-22. External Interrupt Control Register

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 as 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

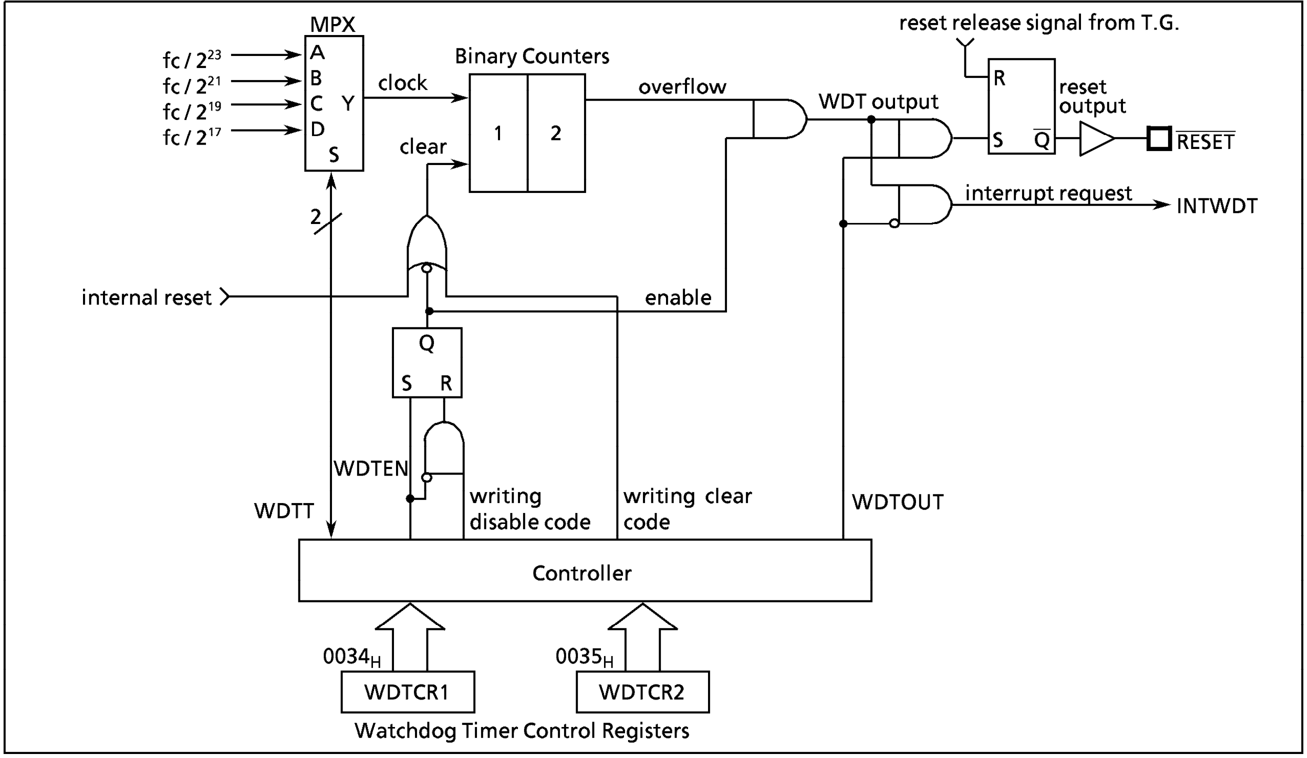


Figure 1-23. Watchdog Timer Configuration

1.10.2 Watchdog Timer Control

Figure 1-24 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 a CPU malfunction occurs for any cause, the watchdog timer output will become active on the rise 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 IDLE mode, and automatically restarts (continues counting) when IDLE mode is released.

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

```
                      LD     (WDTCR2), 4EH          ;  Clears the binary counters
                      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

	7	6	5	4	3	2	1	0	
WDTCR1 (0034_H)					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) 1 : Enable	write only
WDTT	Watchdog timer detection time	00 : 2^{25} / fc [s] (4.194 s at fc = 8 MHz) 01 : 2^{23} / fc (1.048 s at fc = 8 MHz) 10 : 2^{21} / fc (262.1 ms at fc = 8 MHz) 11 : 2^{19} / fc (65.5 ms at fc = 8 MHz)	
WDTOUT	Watchdog timer output select	0 : Interrupt request 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] * ; don't care*

Note 3 : WDTCR1 is a write-onry-register and must not be used with any of read-modify-write instructions

Watchdog Timer Control Register 2

	7	6	5	4	3	2	1	0	
WDTCR2 (0035_H)									(Initial value : **** ****)

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

Note 1 : The disable code is invalid unless written when WDTEN = 0.

*Note 2 : * ; don't care*

Figure 1-24. Watchdog Timer Control Registers

(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 IDLE mode, and restarts automatically after 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 non-maskable 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).

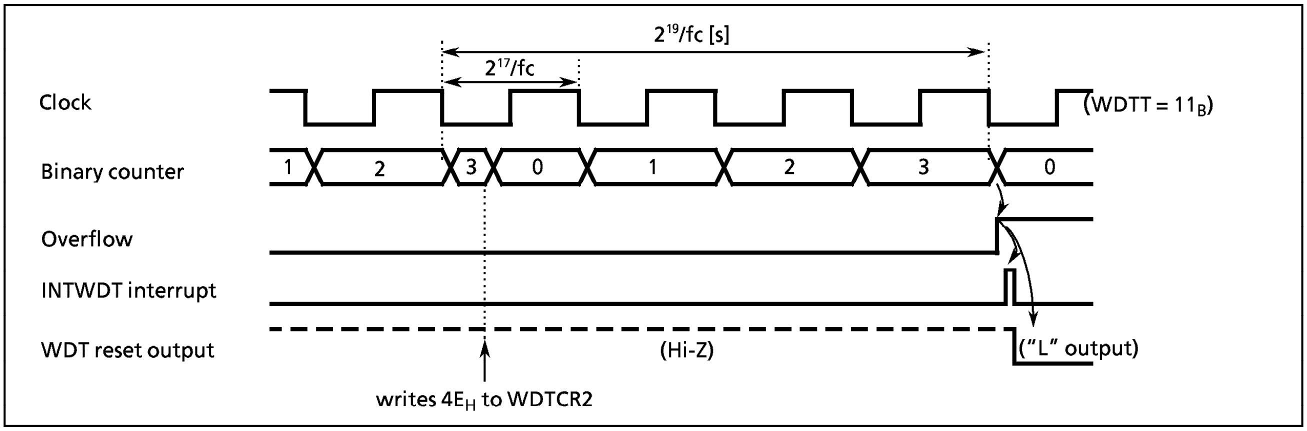


Figure 1-25. Watchdog Timer Interrupt / Reset

1.11 Reset Circuit

The TLCS-870 Series has four types of reset generation procedures: an external reset input, an address-trap-reset, a watchdog timer reset and a system-clock-reset. Table 1-3 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] 131ms at 8MHz) when power is turned on.

Table 1-3. Initializing Internal Status by Reset Action

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

1.11.1 External Reset Input

When the $\overline{\text{RESET}}$ 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 $\overline{\text{RESET}}$ pin input goes high, the reset operation is released and the program execution starts at the vector address stored at addresses FFFE_H - FFFF_H.

The $\overline{\text{RESET}}$ 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.

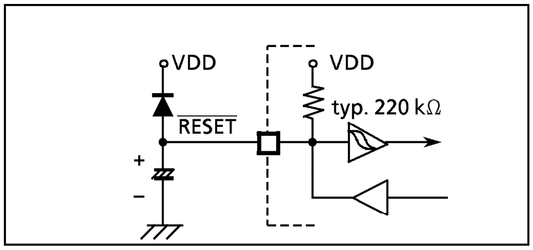
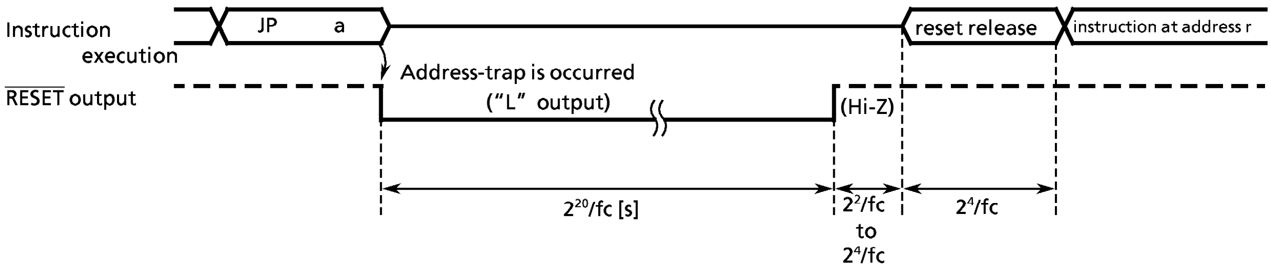


Figure 1-26. Simple Power-on-Reset Circuitry

1.11.2 Address-Trap-Reset

If a CPU malfunction occurs and an attempt is made to fetch an instruction from the RAM or the SFR area (addresses 0000_H - $013F_H$), an address-trap-reset will be generated. Then, the $\overline{\text{RESET}}$ pin output will go low. The reset time is 2^{20}/fc [s] (131ms at fc = 8MHz).



Note 1: $0 \leqq a \leqq 013F_H$

Note 2: During reset release, a reset vector "r" is read out, and an instruction at address r is fetched and decoded.

Figure 1-27. Address-Trap-Reset

1.11.3 Watchdog Timer Reset

Refer to Section "1.10 Watchdog Timer".

1.11.4 System-Clock-Reset

Clearing both bits 7 and 6 in SYSCR to "0" stops high-frequency oscillation, and causes the MCU to deadlock. This can be prevented by automatically generating a reset signal whenever (bit7 in SYSCR) = (bit6 in SYSCR) = 0 is detected to continue the oscillation. Then, the $\overline{\text{RESET}}$ pin output goes low from high-impedance. The reset time is 2^{20}/fc [s] (131ms at fc = 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 peripherals 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 – $003F_H$, and the DBR to addresses $0F80_H$ – $0FFF_H$.

Figure 2-1 shows the list of the 87C444/844 SFRs and DBRs.

Address	Read	Write	Address	Read	Write
0000_H	P0 Port		0020_H	–	SBICR1 (SBI control 1)
01	P1 Port		21	SBIDBR (SBI data buffer)	
02	reserved		22	–	I2CAR (I^2C-bus address)
03	P3 Port		23	SBISR (SBI status)	SBICR2 (SBI control 2)
04	reserved		24	reserved	
05	reserved		25	DACCR1 (D/A Conversion control 1)	
06	P6 Port		26	DACCR2 (D/A Conversion control 2)	
07	P7 Port		27	reserved	
08	reserved		28	SIOSR (SIO1 status)	SIOCR1 (SIO1 control 1)
09	reserved		29	–	SIOCR2 (SIO1 control 2)
0A	–	P0CR (P0 I/O control)	2A	reserved	
0B	–	P1CR (P1 I/O control)	2B	reserved	
0C	–	P6CR (P6 I/O control)	2C	reserved	
0D	–	P7CR (P7 I/O control)	2D	reserved	
0E	ADCCR (A/D Converter control)		2E	reserved	
0F	ADCDR (A/D Conv. Result)	–	2F	reserved	
10	–	TREG1A$_L$ (Timer register 1A)	30	reserved	
11	–	TREG1A$_H$	31	reserved	
12	TREG1B$_L$ (Timer register 1B)		32	reserved	
13	TREG1B$_H$		33	reserved	
14	–	TC1CR (TC1 control)	34	–	WDTCR1 (WDT control)
15	–	TC2CR (TC2 control)	35	–	WDTCR2
16	–	TREG2$_L$ (Timer register 2)	36	–	TBTCR (TBT / TG / DVO control)
17	–	TREG2$_H$	37	–	EINTCR (Exter. interrupt control)
18	reserved		38	reserved	
19	reserved		39	SYSCR (System control)	
1A	reserved		3A	EIR$_L$ (Interrupt enable register)	
1B	reserved		3B	EIR$_H$	
1C	reserved		3C	IL$_L$ (Interrupt latch)	
1D	reserved		3D	IL$_H$	
1E	reserved		3E	reserved	
1F	reserved		3F	PSW (Program status word)	RBS (Register bank selector)

(a) Special Function Registers

Address	Read	Write
$0F80_H$	reserved	
∫	reserved	
$0FDF_H$	reserved	
$0FE0_H$	DACDR0 (D/A Conversion data 0)	
$0FE1_H$	DACDR1 (D/A Conversion data 1)	
$0FE2_H$	DACDR2 (D/A Conversion data 2)	
$0FE3_H$	DACDR3 (D/A Conversion data 3)	
$0FE4_H$	DACDR4 (D/A Conversion data 4)	
$0FE5_H$	DACDR5 (D/A Conversion data 5)	
$0FE6_H$	DACDR6 (D/A Conversion data 6)	
$0FE7_H$	DACDR7 (D/A Conversion data 7)	
$0FE8_H$	reserved	
∫	reserved	
$0FEF_H$	reserved	
$0FF0_H$	SIO1 transmit and receive data buffer	
$0FF1_H$		
$0FF2_H$		
$0FF3_H$		
$0FF4_H$		
$0FF5_H$		
$0FF6_H$		
$0FF7_H$		
$0FF8_H$	reserved	
∫	reserved	
$0FFF_H$	reserved	

(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.)

Note 5 : SBI : Serial Bus Interface.

Figure 2-1. SFR & DBR

2.2 I/O Ports

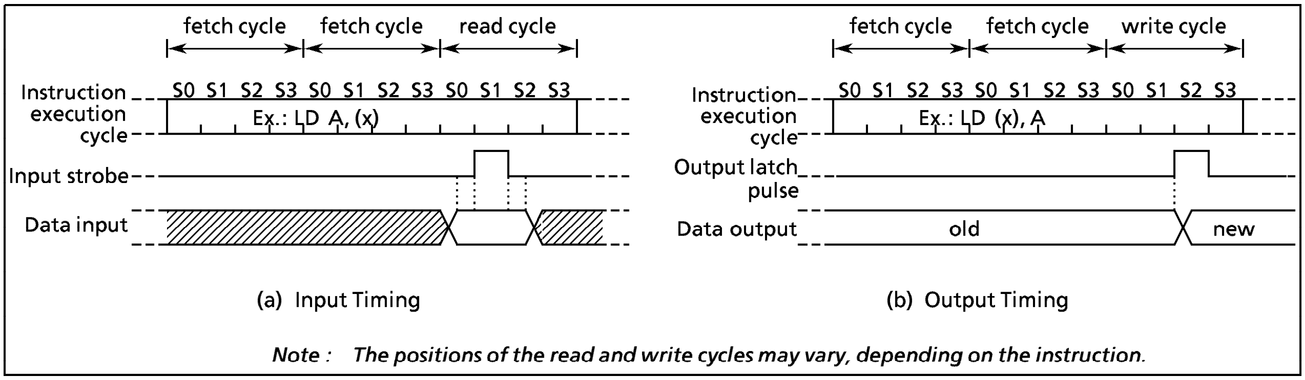
The 87C444/844 have 5parallel input/output ports (34pins) each as follows:

	Primary Function	Secondary Functions
Port P0	8-bit I/O port	————
Port P1	8-bit I/O port	external interrupt input, timer/counter input, and divider output
Port P3	3-bit I/O port	Serial bus interface input/output (I^{2}Cbus/SIO0)
Port P6	7-bit I/O port	analog input, Serial bus interface input/output (SIO1)
Port P7	8-bit I/O port	analog output

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.



(a) Input Timing

(b) Output Timing

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)

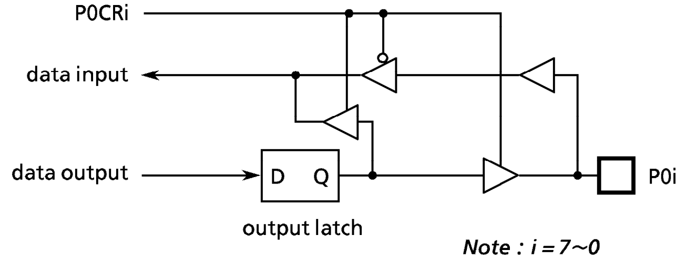
② (HL) side of ADD/ADDC/SUB/SUBB/AND/OR/XOR (src), (HL)

2.2.1 Port P0 (P07 - 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.

> *Note : Input mode port is read the state of input pin.*
> *When input/output mode is used to mixed, the contents of input mode port may be changed by executing bit manipulation instructions.*



Note : i = 7~0

	7	6	5	4	3	2	1	0	
P0 (0000_H)	P07	P06	P05	P04	P03	P02	P01	P00	(Initial value : 0000 0000)

	7	6	5	4	3	2	1	0	
P0CR ($000A_H$)									(Initial value : 0000 0000)

P0CR	I/O control for port P0	0 : input mode 1 : output mode	write only

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).

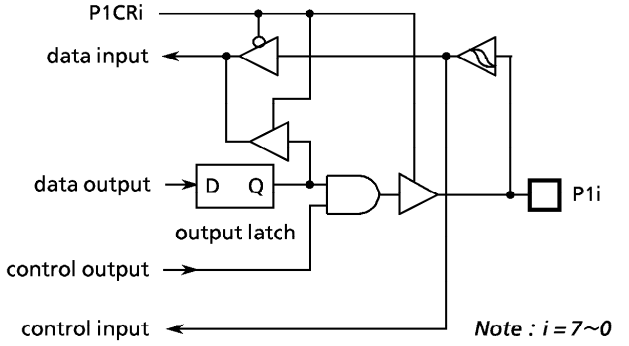
```
LD      (P0), 00001010B       ;  Sets initial data to P0 output latches
LD      (P0CR), 00001111B     ;  Sets the port P0 input/output mode
```

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



P1 (0001$_H$)

7	6	5	4	3	2	1	0
P17	P16	P15 TC2	P14 $\overline{\text{PPG}}$	P13 $\overline{\text{DVO}}$	P12 INT2 TC1	P11 INT1	P10 $\overline{\text{INT0}}$

(Initial value : 0000 0000)

P1CR (000B$_H$)

7	6	5	4	3	2	1	0

(Initial value : 0000 0000)

P1CR	I/O control for port P1	0 : Input mode 1 : Output mode	write only

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
```

Note : Input mode port is read the state of input pin.
When input/output mode is used to mixed, the contents of input mode port may be changed by executing bit manipulation instructions.

2.2.3 Port P3 (P37 - P35)

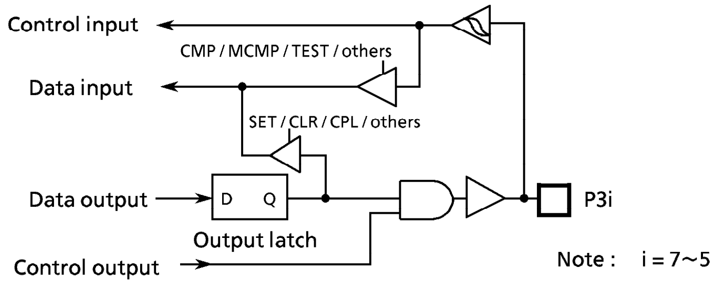
Port P3 is a 3-bit input/output port, and is also used as serial bus interface (I^{2}Cbus/SIO0) input/output. When used as an input port or a secondary function pin, the output latch should be set to "1". The output latches are initialized to "1" during reset.

Example 1: Outputs an immediate data 5A$_H$ to port P3.

LD (P3), 5AH ; P3←5A$_H$

Example 2: Inverts the output of the 3bits (P37 - P35) in port P3.

XOR (P3), 11100000B
; P37~P35←$\overline{\text{P37}}$~$\overline{\text{P35}}$



P3 (0003$_H$)	7	6	5	4	3	2	1	0
	P37 SCL SI0	P36 SDA SO0	P35 $\overline{\text{SCK0}}$					

(Initial value 111* ****)

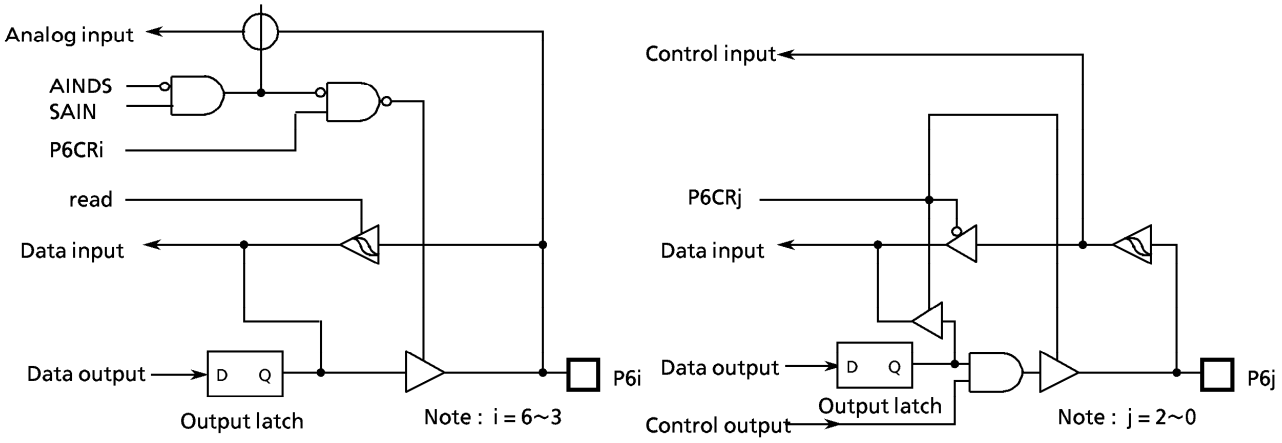
Figure 2.5 Port P3

2.2.4 Port P6 (P66 - P60)

Port P6 is an 7-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 and a serial bus interface (SIO1). 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 "0". In this case, unuse pin as analog input is configured as input/output port.
During reset, ADCCR is initialized to "0" and bits of P6CR are initialized to "*000 0000B", which configures port P6 as input port. The P6 output latches are initialized to "1". When used as a serial bus interface, the output latch should be set to "1" and set the P6CR to an input or output mode. 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.

Note : Input mode port is read the state of input pin.
When input/output mode is used to mixed, the contents of input mode port may be changed by executing bit manipulation instructions.



(a) P66~P63

(b) P62~P60

	7	6	5	4	3	2	1	0	
P6 (0006_H)		P66 AIN3	P65 AIN2	P64 AIN1	P63 AIN0	P62 SI1	P61 SO1	P60 $\overline{\text{SCK1}}$	(Initial value *111 1111)

	7	6	5	4	3	2	1	0	
P6CR ($000C_H$)									(Initial value *000 0000)

P6CR	I/O control for port P6 (Set each of a bit)	0 : Input mode 1 : Output mode	write only

Figure 2-6. Port P6 and P6CR

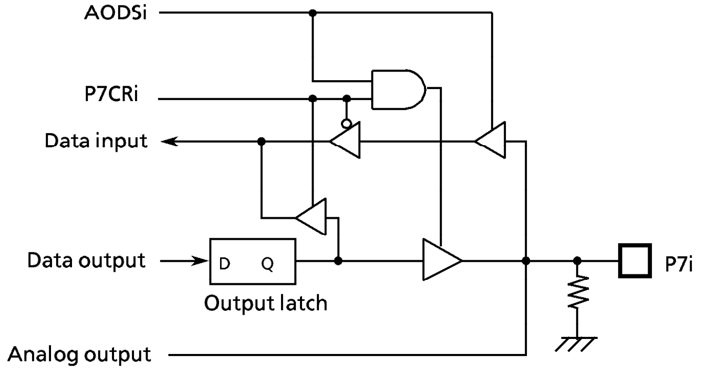
2.2.5 Port P7 (P77 - P70)

Port P7 is an 8-bit general-purpose 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 output (only bit of DACCR1 is set "0"). For example, port P7 is configured as an input if its corresponding P7CR bit is cleared to "0", and as an output if its corresponding bit is set to "1". The output latches are initialized to "0". During reset, DACCR1 is initialized to "0", which configures port P7 as an analog output. When used as an analog output, DACCR1 should be set to "0".

Data is written into the output latch regardless of the P7CR contents. Therefore initial output latch before setting P7CR.

Port P7 has Buit-in pull-down resistors.

Note : Input mode port is read the state of input pin.
When input/output mode is used to mixed, the contents of input mode port may be changed by executing bit manipulation instructions.



Note : i = 7~0

	7	6	5	4	3	2	1	0	
P7 (0007$_H$)	P77 DA7	P76 DA6	P75 DA5	P74 DA4	P73 DA3	P72 DA2	P71 DA1	P70 DA0	(Initial value 0000 0000)

	7	6	5	4	3	2	1	0	
P7CR (000D$_H$)									(Initial value 0000 0000)

P7CR	I/O control for port P7	0 : Input mode 1 : Output mode (Only the case of the DACCR1 = "1", I/O control can be available.)	write only

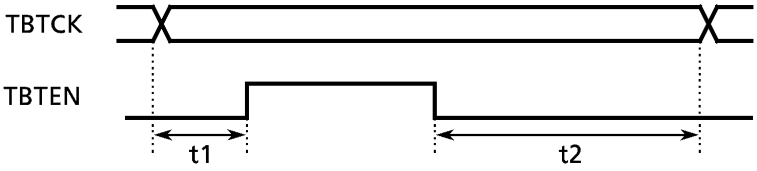
Figure 2-7. Port and P7CR

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 a control register (TBTCR) shown in Figure 2-9.

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 (When the time base timer is changed from enabling to disabling, the interrupt frequency can't be changed.) (both frequency selection and enabling can be performed simultaneously).

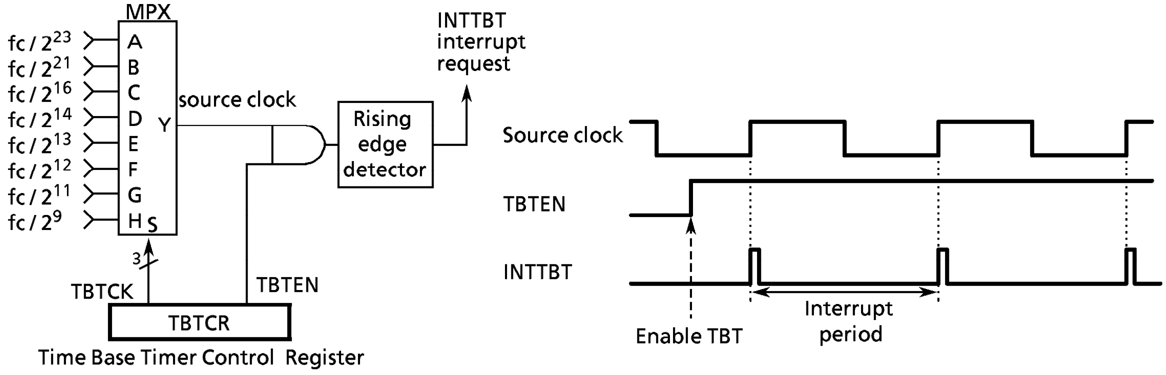


$t1 \geqq 0$

$t2 \geqq 1$ machine cycle

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

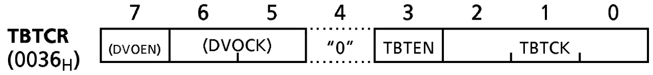
```
LD      (TBTCR) , 00001010B
SET     (EIRL) . 6
```



(a) Configuration

(b) Time Base Timer Interrupt

Figure 2-8. Time Base Timer



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

TBTEN	Time base timer enable/disable	0 : Disable 1 : Enable	write only
TBTCK	Time base timer interrupt frequency select	000 : fc / 2^{23} [Hz] (0.95 Hz at fc = 8MHz) 001 : fc / 2^{21} (3.81 at fc = 8MHz) 010 : fc / 2^{16} (122.07 at fc = 8MHz) 011 : fc / 2^{14} (488.28 at fc = 8MHz) 100 : fc / 2^{13} (976.56 at fc = 8MHz) 101 : fc / 2^{12} (1953.12 at fc = 8MHz) 110 : fc / 2^{11} (3906.25 at fc = 8MHz) 111 : fc / 2^{9} (15625 at fc = 8MHz)	

*Note1 : fc ; High-frequency clock [Hz], * ; don't care*

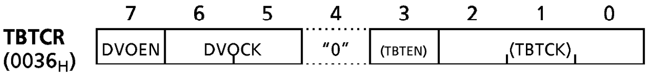
Note2 : The TBTCR is a write-only register and must not be used with any of the read-modify-write instructions.

Figure 2-9. Time Base Timer Control Register

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

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 ($\overline{\text{DVO}}$). 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 2-10.



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

DVOEN	Divider output enable/disable	0 : Disable 1 : Enable	R/W
DVOCK	Divider output ($\overline{\text{DVO}}$ pin) frequency selection	00 : fc / 2^{13} [Hz] 01 : fc / 2^{12} 10 : fc / 2^{11} 11 : fc / 2^{10}	

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

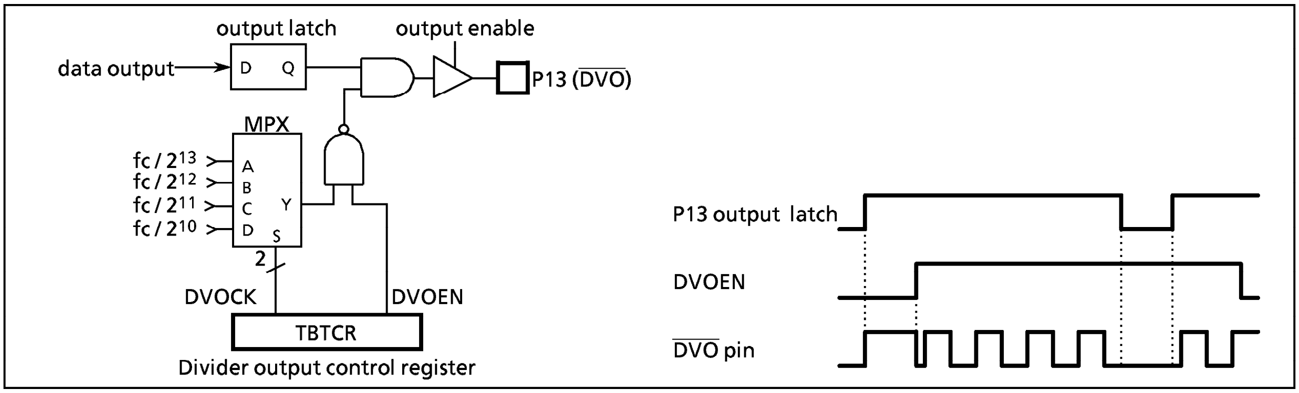
Figure 2-10. Divider Output Control Register

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 2-1. Frequency of Divider Output

DVOCK	Frequency of Divider Output	At fc = 8 MHz
00	fc / 2^{13}	0.976 [kHz]
01	fc / 2^{12}	1.953
10	fc / 2^{11}	3.906
11	fc / 2^{10}	7.812



(a) Configuration (b) Timing Chart

Figure 2-11. Divider Output

2.5 16-bit Timer/Counter 1 (TC1)

2.5.1 Configuration

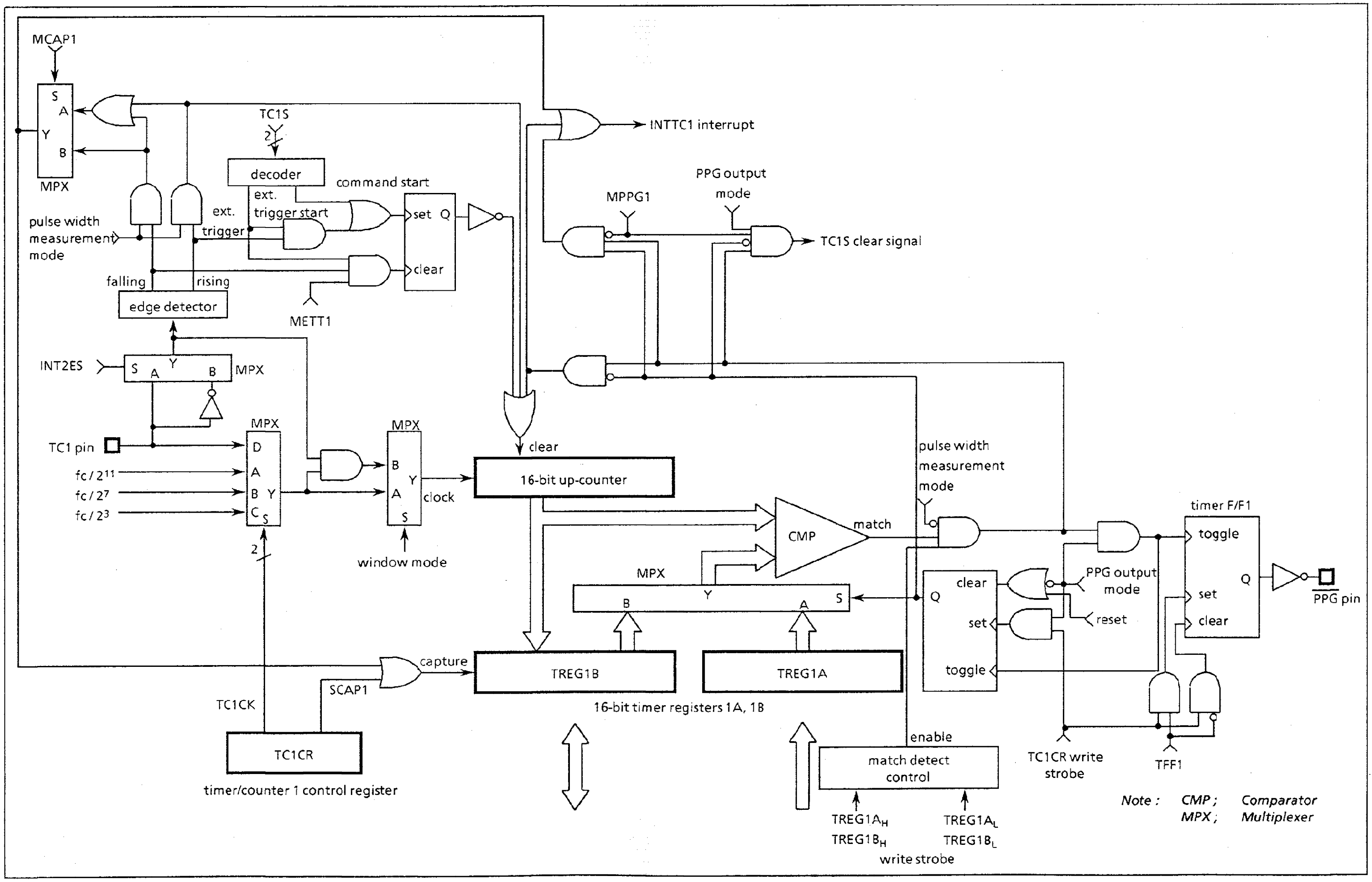


Figure 2-12. Timer / Counter 1

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.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
TREG1A (0010, 0011_H)	TREG1A$_H$ (0011_H)								TREG1A$_L$ (0010_H)								write only
TREG1B (0012, 0013_H)	TREG1B$_H$ (0013_H)								TREG1B$_L$ (0012_H)								R/W (Write available in only PPG output mode)

	7	6	5	4	3	2	1	0	
TC1CR (0014_H)	TFF1	SCAP1 MCAP1 METT1 MPPG1	TC1S		TC1CK		TC1M		(Initial value : 0000 0000)

TC1M	TC1 mode select	00 : timer / external trigger timer / event counter mode 01 : window mode 10 : pulse width measurement mode 11 : PPG output mode	write only
TC1CK	TC1 source clock select	00 : internal clock fc/2^{11} [Hz] 01 : internal clock fc/2^{7} 10 : internal clock fc/2^{3} 11 : external clock (TC1 pin input)	
TC1S	TC1 start control	00 : stop & counter clear 01 : command start 10 : reserved 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],

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/F1 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 2-13. 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 2-2. Timer/Counter 1 Source Clock (Internal Clock)

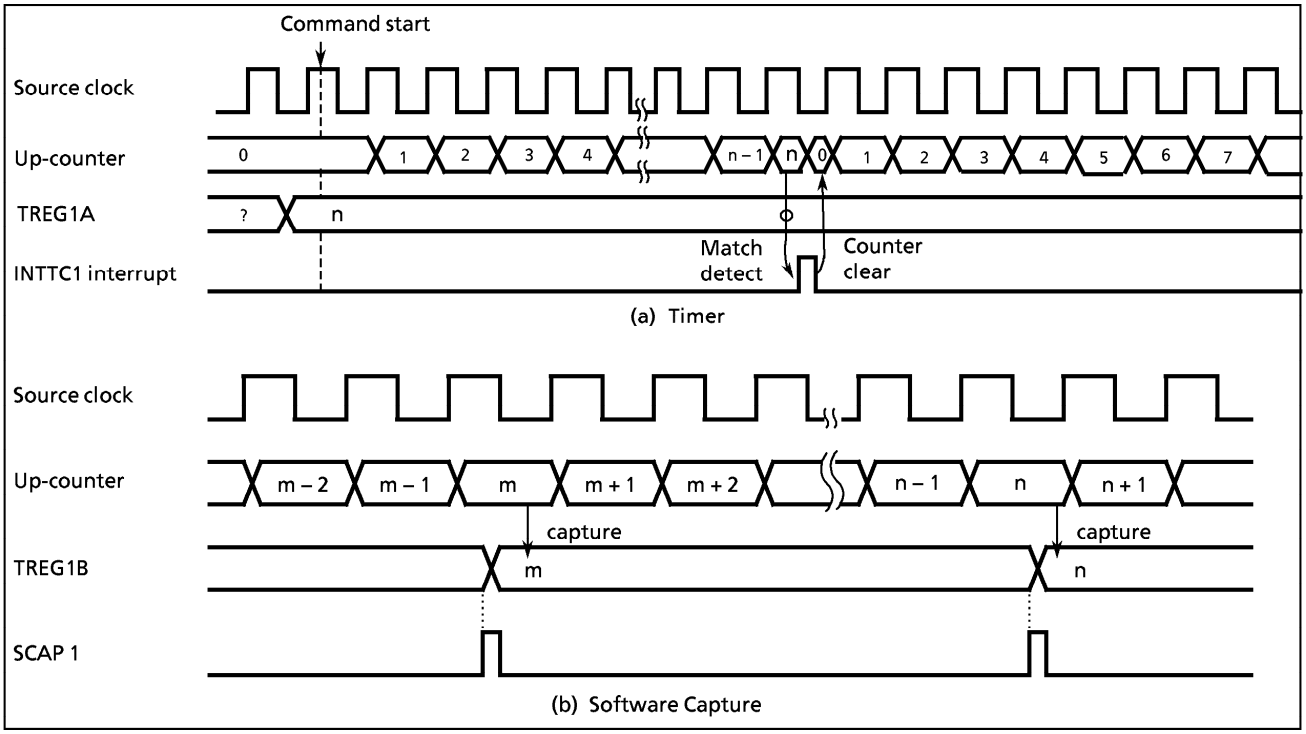
Source clock	Resolution	Maximum time setting
	At fc = 8 MHz	At fc = 8 MHz
fc / 2^{11} [Hz] fc / 2^7 fc / 2^3	256 μs 16 μs 1 μs	16.8 s 1.0 s 65.5 ms

Example 1 : Sets the timer mode with source clock fc/2^{11}[Hz] and generates an interrupt 1 s. later (at fc = 8 MHz).

```
LD      (TC1CR), 00000000B     ; Sets the TC1 mode and source clock
LDW     (TREG1A), 1000H        ; Sets the timer register (1s ÷ 2¹¹ / fc = 1000H)
LD      (TC1CR), 00010000B     ; Starts TC1
```

Example 2 : Software capture

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



(a) Timer

(b) Software Capture

Figure 2-14. 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.

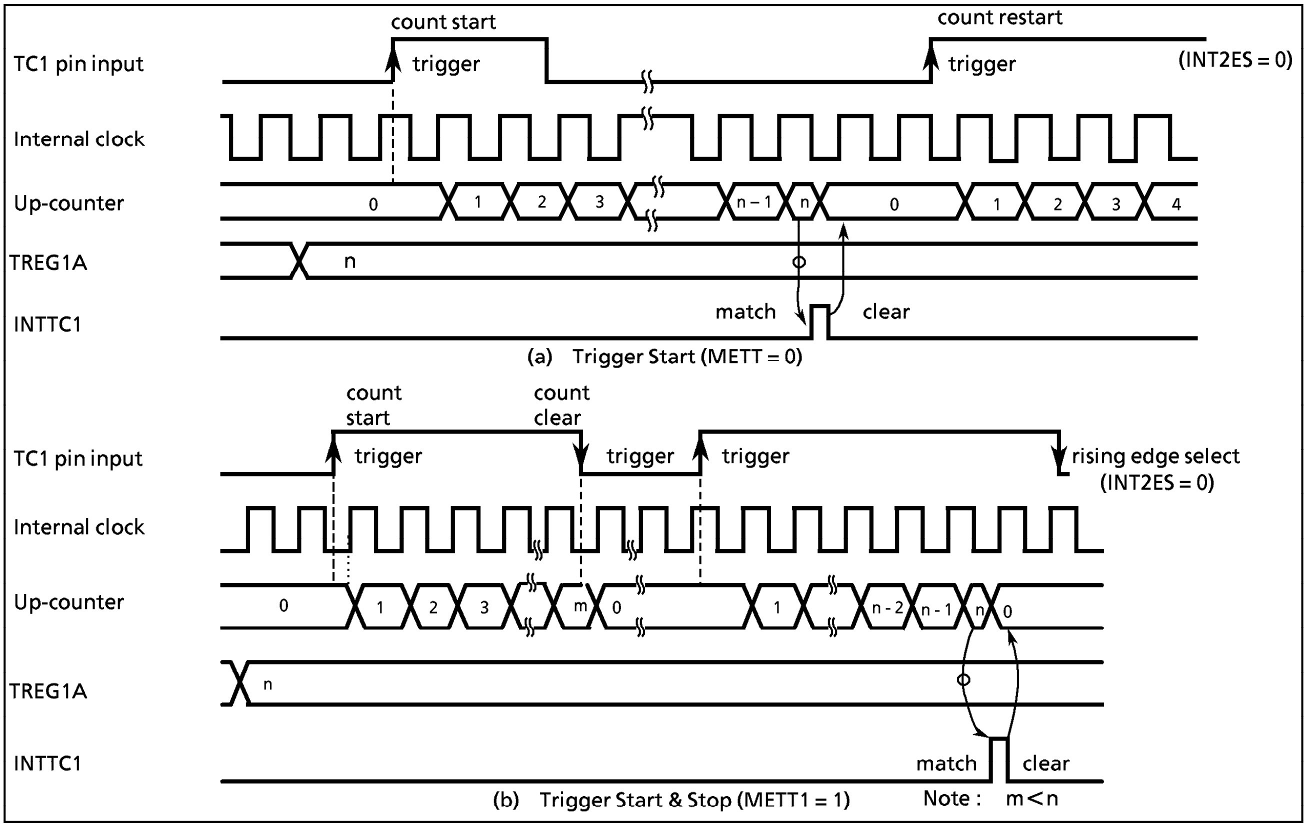


Figure 2-15. External Trigger Timer Mode Timing Chart

(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].

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

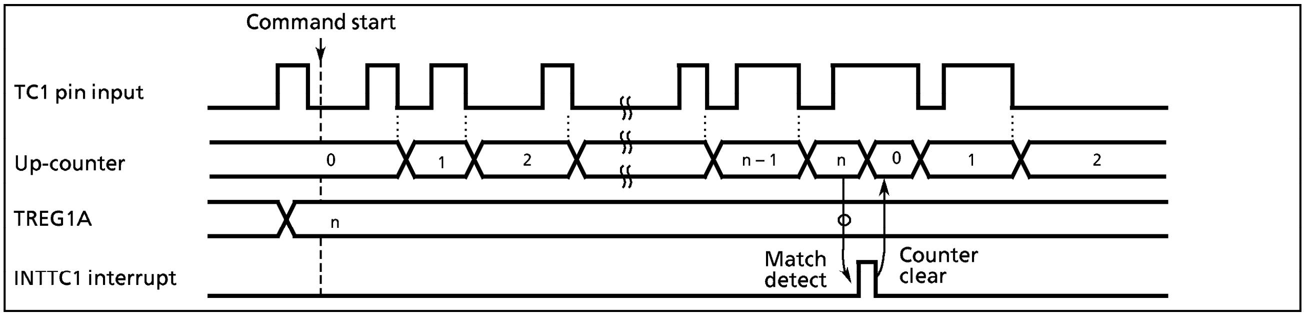
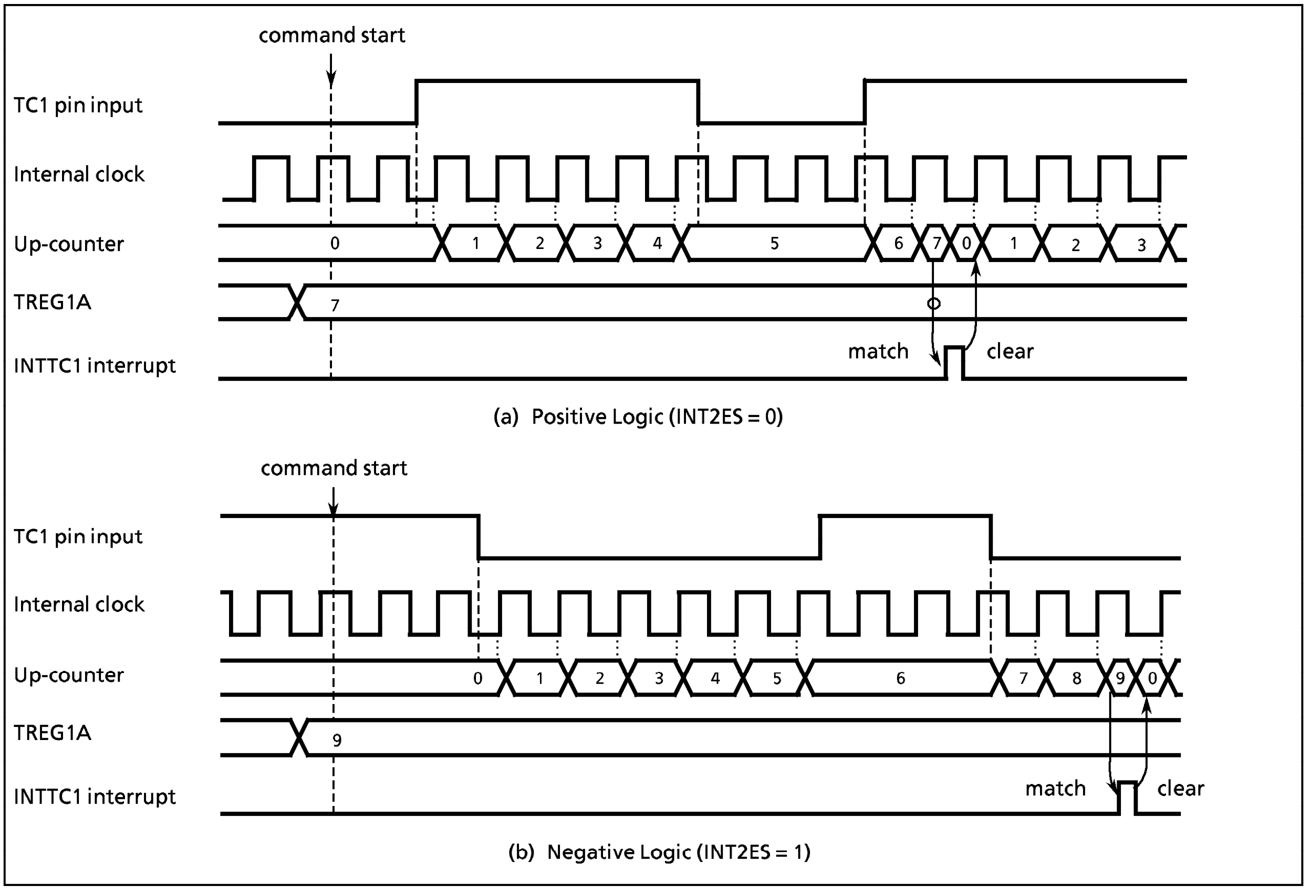


Figure 2-16. Event Counter Mode Timing Chart (INT2ES = 1)

(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.



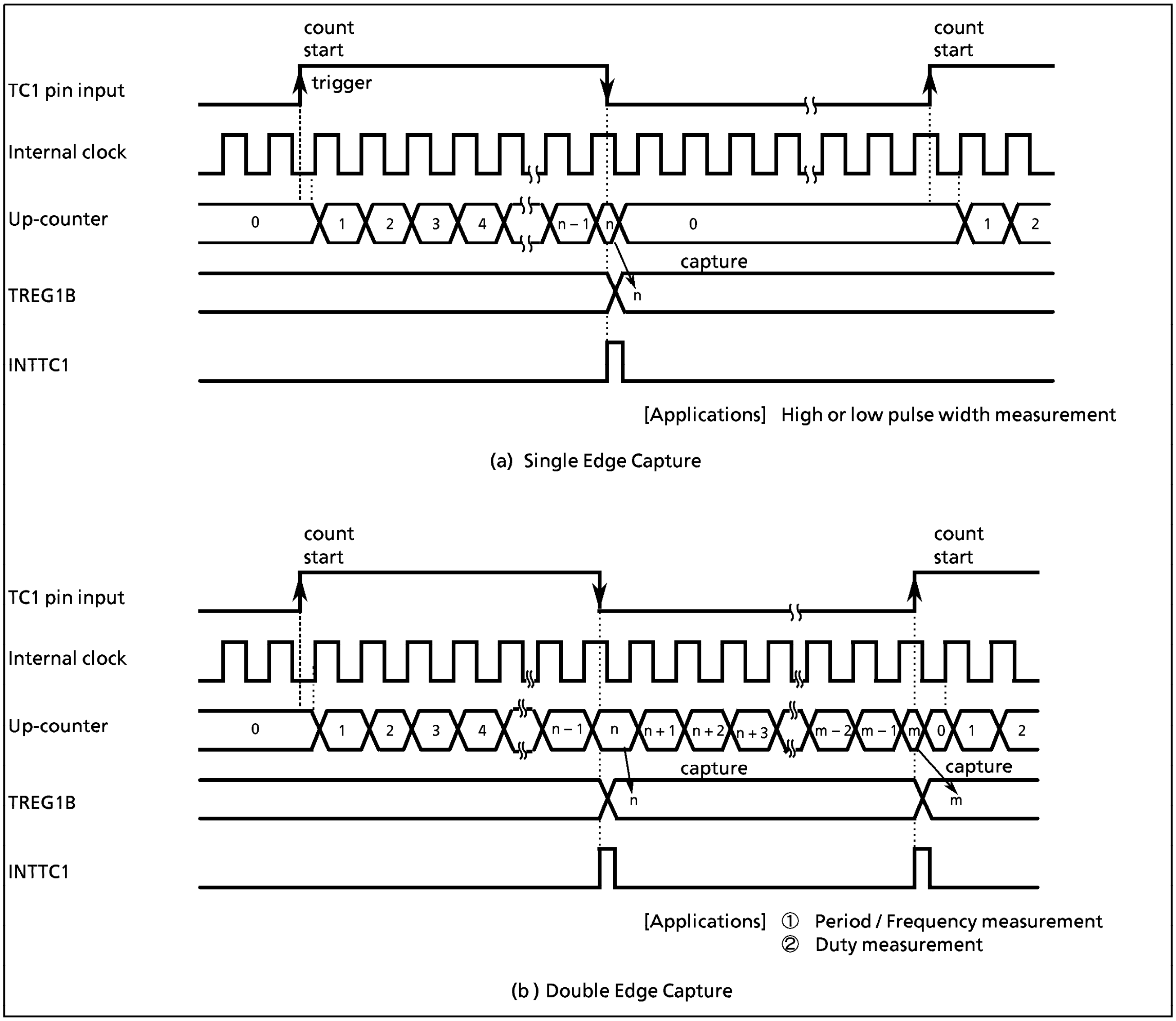
(a) Positive Logic (INT2ES = 0)

(b) Negative Logic (INT2ES = 1)

Figure 2-17. 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).



[Applications] High or low pulse width measurement

(a) Single Edge Capture

[Applications] ① Period / Frequency measurement
② Duty measurement

(b) Double Edge Capture

Figure 2-18. Pulse Width Measurement Mode Timing Chart

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

```
           CLR   (INTTC1C). 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
           LD    (TC1CR), 00110110B        ;  Starts TC1 with an external trigger
            ⋮
PINTTC1 :  CPL   (INTTC1C). 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)
            ⋮
```

(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/F1 output is connected to the P14 ($\overline{\text{PPG}}$) pin. In the case of 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.

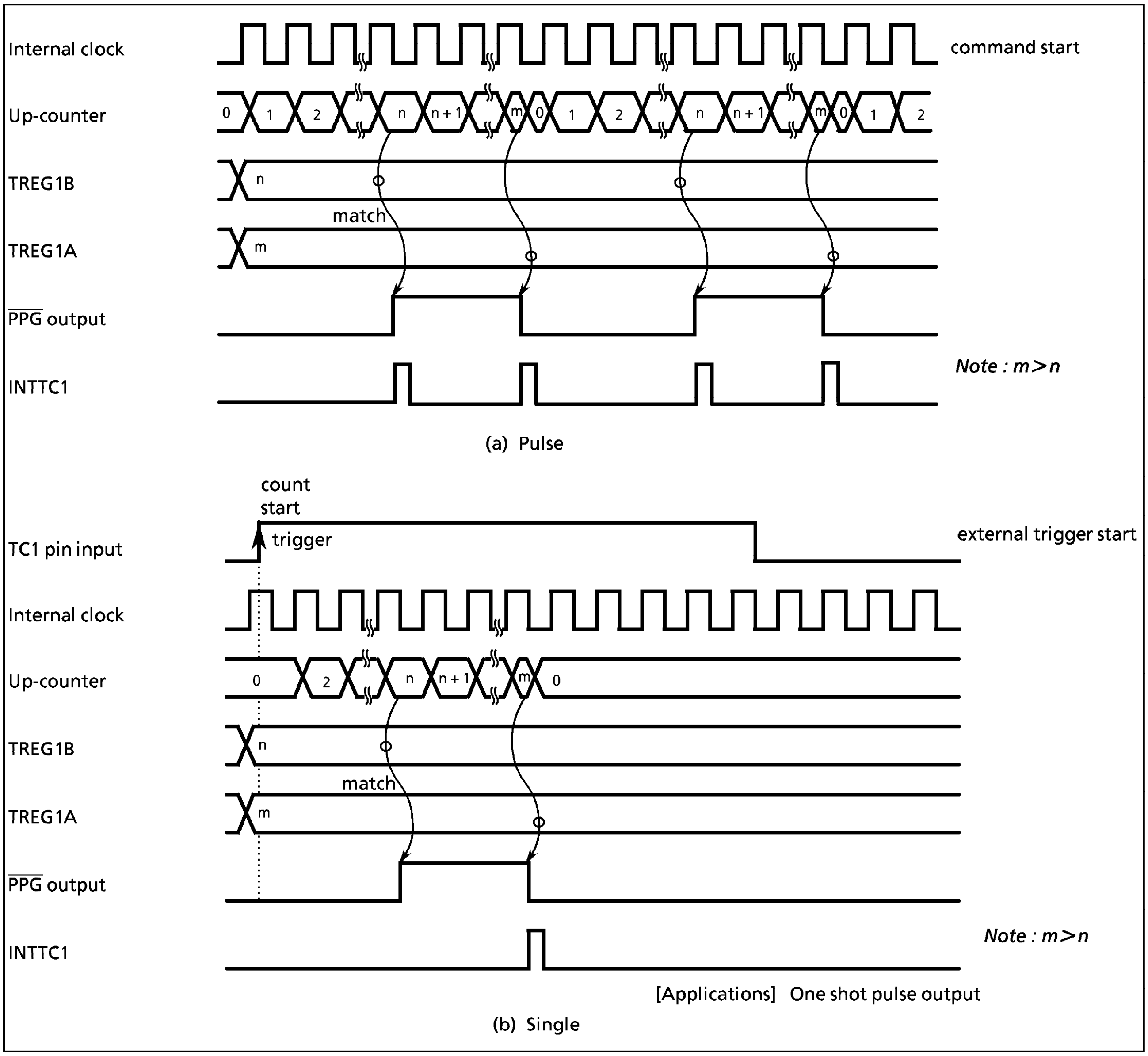


Figure 2-19. PPG Output Mode Timing Chart

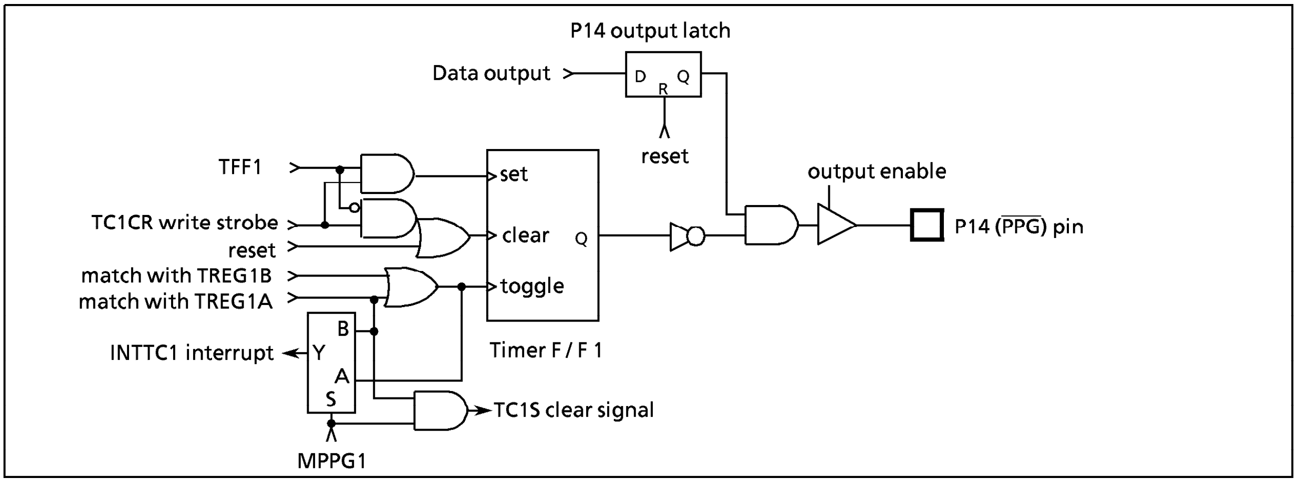


Figure 2-20. PPG Output

2.6 16-bit Timer/Counter 2 (TC2)

2.6.1 Configuration

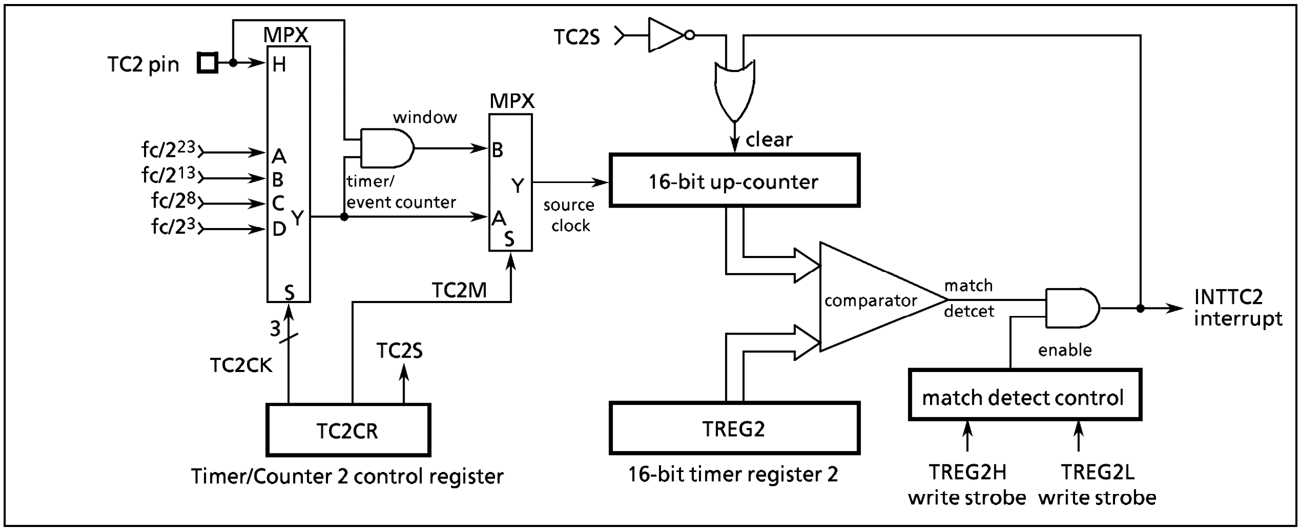
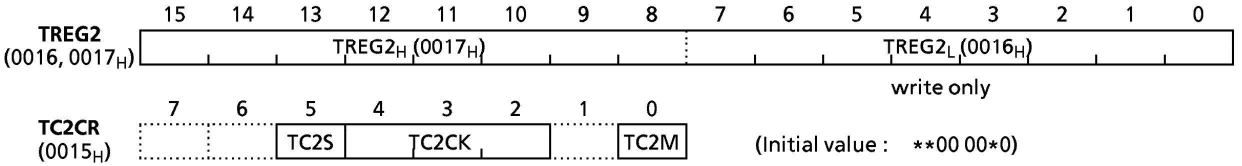


Figure 2-21. Timer/Counter 2 (TC2)

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.



TC2M	Timer/counter 2 operating mode select	0 : Timer/Event counter mode 1 : Window mode	write only
TC2CK	Timer/counter 2 source clock select	000 : Internal clock fc / 2^{23} [Hz] 001 : 〃 fc / 2^{13} 010 : 〃 fc / 2^{8} 011 : 〃 fc / 2^{3} 10* : Reserved 110 : Reserved 111 : External clock (TC2 pin input)	
TC2S	Timer/counter 2 start control	0 : Stop and counter clear 1 : Start	

*Note 1 : fc; High-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

Note 5 : Always write "0" to bit 0 in TC2CR.

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

Figure 2-22. 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.

(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.

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

Source clock	Resolution	Maximum time setting
	At fc = 8 MHz	At fc = 8 MHz
$fc / 2^{23}$ [Hz]	1.05 s	19.1 h
$fc / 2^{13}$	1.02 ms	1.1 min
$fc / 2^{8}$	32 μs	2.1 s
$fc / 2^{3}$	1 μs	65.5 ms

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 (25 ms ÷ 2^3/fc = 61A8H)
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].

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

```
LD      (TC2CR), 00011100B      ;  Sets the TC2 mode
LDW     (TREG2), 0280H          ;  Sets TREG2
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.

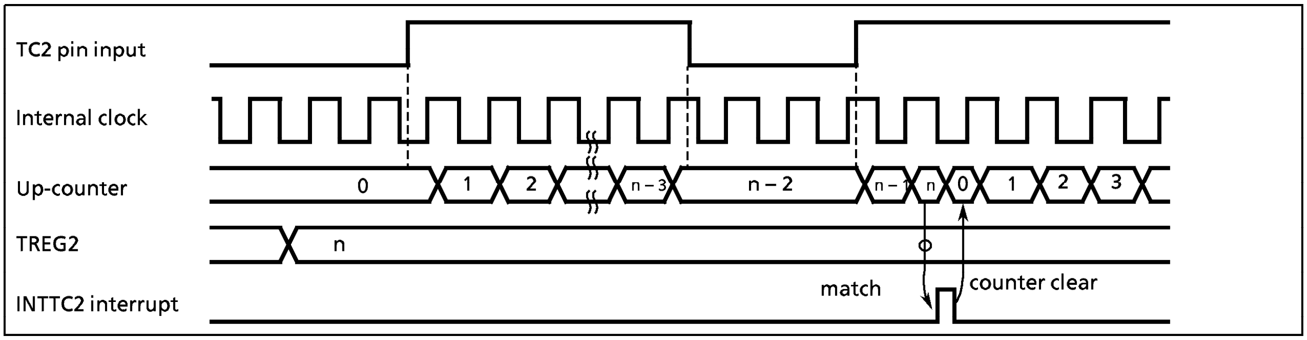


Figure 2-23. Window Mode Timing Chart

2.7 Serial Bus Interface (SBI)

The 87C444/844 has a 1-channel serial bus interface which employs a clocked-synchronous 8-bit serial bus interface (SIO0) and an I^{2}C bus (a bus system by Philips).

The serial bus interface is connected to an external device through P37 (SCL) and P36 (SDA) in the I^{2}C bus mode; and through P37 (SI0), P36 (SO0), and P35 ($\overline{\text{SCK0}}$) in the clocked-synchronous 8-bit SIO mode.

The serial bus bus interface pins are also used for the P3 port. When used for serial bus interface pins, set the P3 output latches of these pins to "1". When not used for serial bus interface pins, the P3 port is used as a normal I/O port.

2.7.1 Configuration

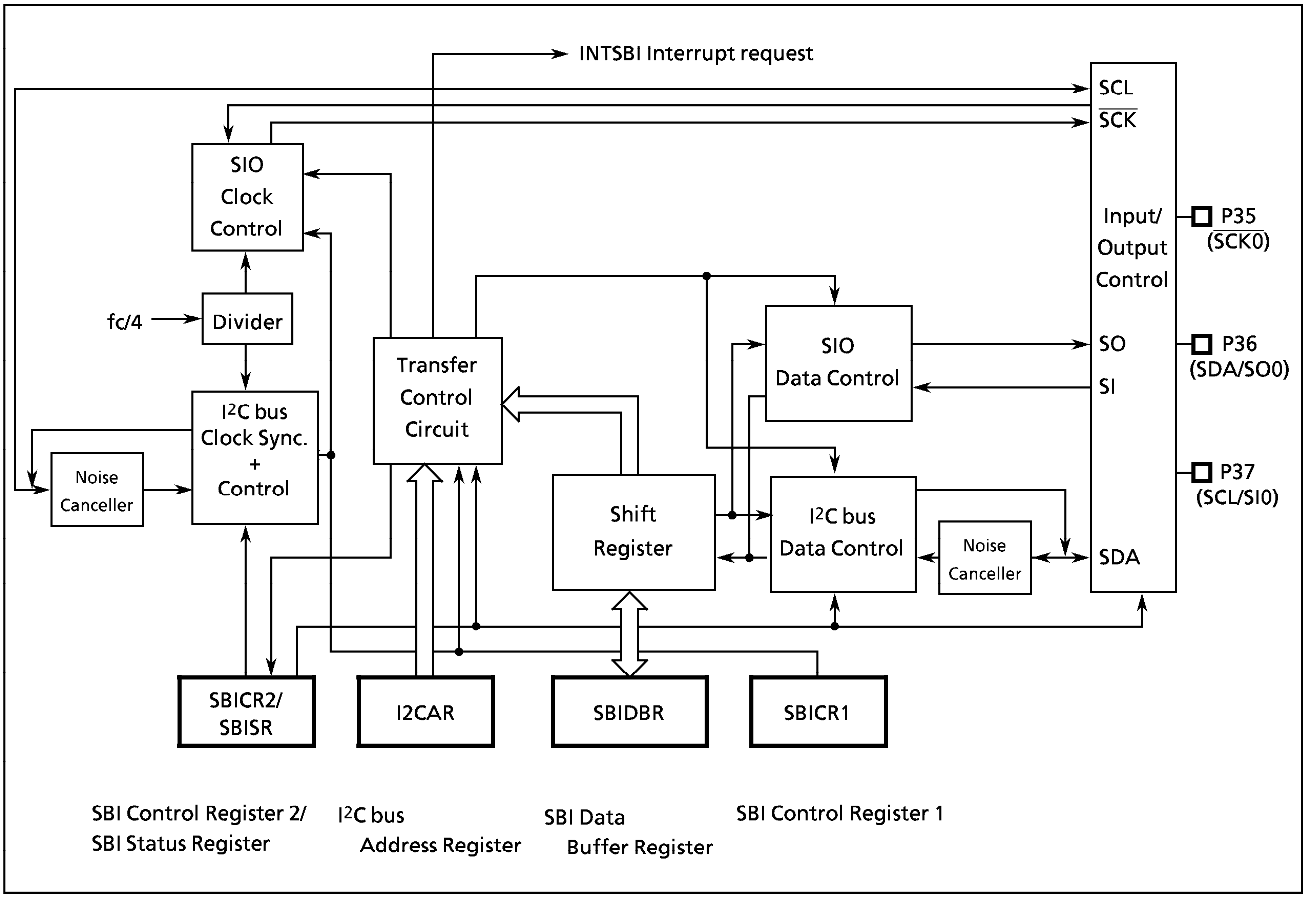


Figure 2-24. Serial Bus Interface (SBI)

2.7.2 Serial Bus Interface (SBI) Control

The following reginsters are used for control and operation status monitoring when using the serial bus interface (SBI).

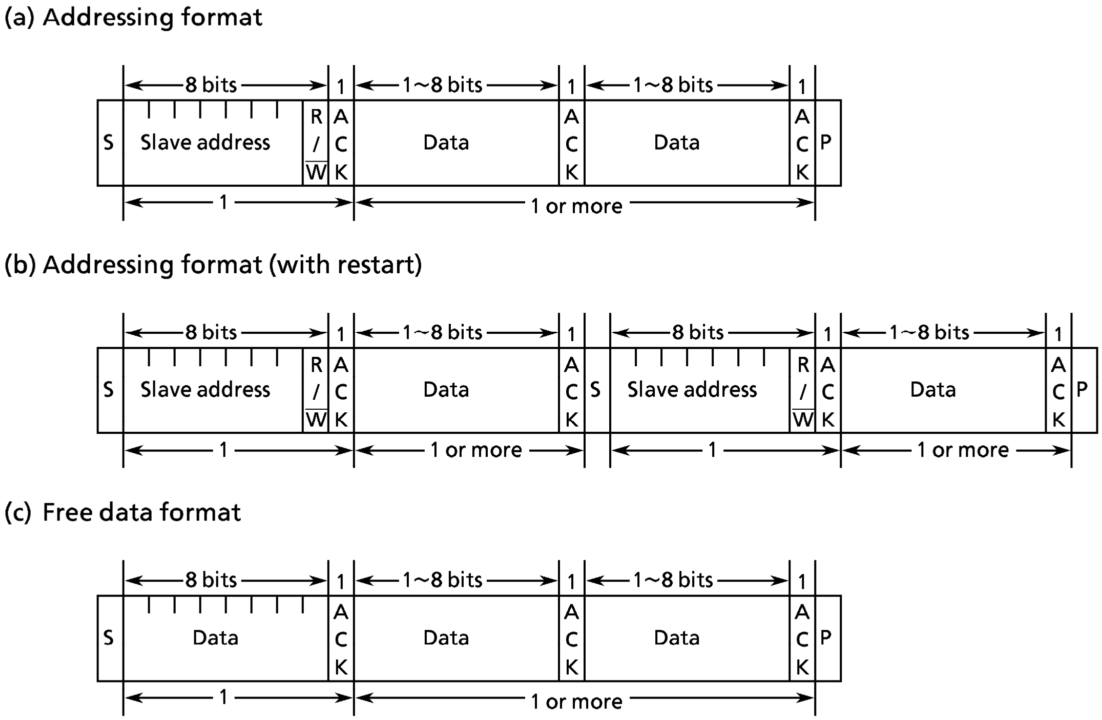
- Serial bus interface control register 1 (SBICR1)
- Serial bus interface control register 2 (SBICR2)
- Serial bus interface data buffer register (SBIDBR)
- I^{2}C bus address register (I2CAR)
- Serial bus interface status register (SBISR)

The above registers differ depending on an mode to be used.

Refer to Section "2.7.4 I^{2}C bus Mode Control" and "2.7.6 Clocked-synchronous 8-bit SIO Mode Control".

2.7.3 The Data Formats in the I^{2}C bus Mode

The data formats when using the 87C444/844 in the I^{2}C bus mode are shown below.



Notes : *S : Start condition*
R/$\overline{W}$: Direction bit
ACK : Acknowledge bit
P : Stop condition

Figure 2-25. Data format

2.7.4 I^{2}C Bus Mode Control

The following registers are used for control and operation status monitoring when using the serial bus interface (SBI) in the I^{2}C bus mode.

Serial Bus Interface Control Register 1

SBICR1 (0020$_H$)

7	6	5	4	3	2	1	0
BC			ACK	"0"	SCK		

(Initial value 0000 *000)

		BC	ACK = 0		ACK = 1		
			Number of Clock	Bits	Number of Clock	Bits	
BC	Number of transferred bits	000	8	8	9	8	write only
		001	1	1	2	1	
		010	2	2	3	2	
		011	3	3	4	3	
		100	4	4	5	4	
		101	5	5	6	5	
		110	6	6	7	6	
		111	7	7	8	7	

ACK	Acknowledge mode specification	0 : Acknowledge not returned to transmitter. 1 : Acknowledge returned to transmitter.	R/W
SCK	Serial clock selection	000 : 181.8 kHz 001 : 105.3 kHz 010 : 57.1 kHz 011 : 29.9 kHz 100 : 15.3 kHz 101 : 7.72 kHz 110 : 3.88 kHz } at fc = 8 MHz (Output on SCL pin) 111 : reserved	write only

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

Note 2 : Set the BC to "000" before switching to 8-bit SIO bus mode.

Serial Bus Interface Data Buffer Register

SBIDBR (0021$_H$)

7	6	5	4	3	2	1	0

R/W

Note : For writing transmitted data, start from the MSB.

I^{2}C bus Address Register

I2CAR (0022$_H$)

7	6	5	4	3	2	1	0
Slave address							ALS
SA6	SA5	SA4	SA3	SA2	SA1	SA0	

(Initial value 0000 0000)

SA	87C444/844 slave address selection		write only
ALS	Address recognition mode specification	0 : Slave address recognition 1 : Non slave address recognition	

Figure 2-26. Serial Bus Interface Control Register 1/Serial Bus Interface Data Buffer Register/ I^{2}C bus Address Register in the I^{2}C bus Mode

Serial Bus Interface Control Register 2

SBICR2 (0023_H)

7	6	5	4	3	2	1	0
MST	TRX	BB	PIN	SBIM		"0"	"0"

(Initial value 0001 00**)

MST	Master/slave selection (Write), Status monitor (Read)	0 : Slave 1 : Master	R/W
TRX	Transmitter/receiver selection (Write), Status monitor (Read)	0 : Receiver 1 : Transmitter	
BB	Start/stop generation (Write), I^{2}C bus status monitor (Read)	0 : Stop condition (Write) , Bus free (Read) 1 : Start condition (Write) , Bus busy (Read)	
PIN	Cancel interrupt service request(Write), Interrupt service reguest status monitor(Read)	0 : – (Write), Interrupt service requested (Read) 1 : Cancel interrupt service request (Write) , canceled (Read)	
SBIM	Serial bus interface operating mode selection	00 : Port mode (serial bus interface output disable) 01 : SIO mode 10 : I^{2}C bus mode 11 : Reserved	write only

*Note 1 : * ; don't care*

Note 2 : Switch a mode to port after confirming that the bus is free.

SBISR (0023_H)

7	6	5	4	3	2	1	0
MST	TRX	BB	PIN	AL	AAS	AD0	LRB

AL	Arbitration lost detection monitor	0 : – 1 : Arbitration lost detected	read only
AAS	Slave address match detection monitor	0 : – 1 : Slave address match or "GENERAL CALL" detected	
AD0	"GENERAL CALL" detection monitor	0 : – 1 : "GENERAL CALL" detected	
LRB	Last received bit monitor	0 : Last received bit "0" 1 : Last received bit "1"	

Figure 2-27. Serial Bus interface Control Register 2/Serial Bus interface status register in the I^{2}Cbus Mode

(1) **Acknowledge mode specification**

Set the ACK (bit 4 in the SBICR1) to "1" for operation in the acknowledge mode. The 87C444/844 generates an additional clock pulse for an acknowledge signal when operating in the master mode. In the transmitter mode during the clock pulse cycle, the SDA pin is released in order to receive the acknowledge signal from the receiver. In the receiver mode during the clock pulse cycle, the SDA pin is set to the low level in order to generate the acknowledge signal.

Reset the ACK for operation in the non-acknowledge mode. The 87C444/844 does not generate a clock pulse for the acknowledge signal when operating in the master mode.

(2) **Number of transfer bits**

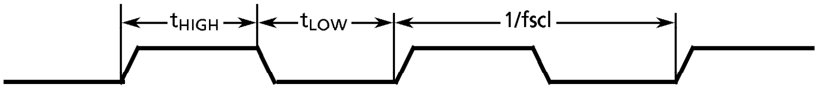
The BC (bits 7 to 5 in the SBICR1) is used to select a number of bits for next transmitting and receiving data.

Since the BC is cleared to "000" as a start condition, a slave address and direction bit transmissions are executed in 8 bits. Other than these, the BC retains a specified value.

(3) Serial clock

a. Clock source

The SCK (bits 2 to 0 in the SBICR1) is used to select a maximum transfer frequency directed from the SCL pin in the master mode.



$t_{LOW} = 2^n/fc$

$t_{HIGH} = 2^n/fc + 12/fc$

$fscl = 1/(t_{Low} + t_{HIGH})$

Note : fc ; high-frequency clock

SCK (bits2 to 0 in the SBICR1)	n
000	4
001	5
010	6
011	7
100	8
101	9
110	10

Figure 2-28. Clock Source

b. Clock synchronization

In the I^{2}C bus mode, in order to wire AND a bus, a master device which pulls down a clock pulse to low level, in the first place, invalidate a clock pulse of another master device which generates a high-level clock pulse. The master device with a high-level clock pulse needs to detect the situation and implement the following procedure.

The 87C444/844 have a clock synchronization function for normal data transfer even when more than one master exists on a bus.

The example explains clock synchronization procedures when two masters simultaneously exist on a bus.

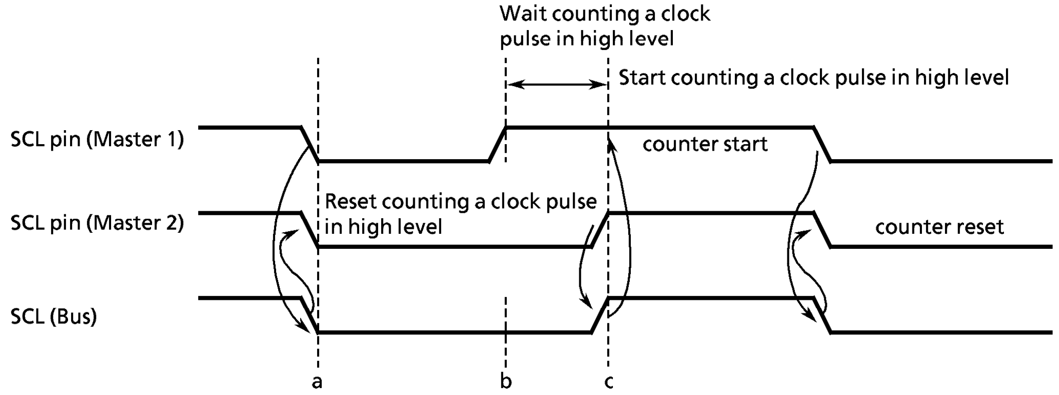

Wait counting a clock pulse in high level
Start counting a clock pulse in high level
SCL pin (Master 1)
counter start
SCL pin (Master 2)
Reset counting a clock pulse in high level
counter reset
SCL (Bus)
a
b
c

Figure 2-29. Clock Synchronization

As Master 1 pulls down the SCL pin to the low level at point "a", the SCL line of the bus becomes the low level. After detecting this situation, Master 2 resets counting a clock pulse in the high level and sets the SCL pin to the low level.

Master 1 finishes counting a clock pulse in the low level at point "b" and sets the SCL pin to the high level. Since Master 2 holds the SCL line of the bus at the low level, Master 1 waits for counting a clock pulse in the high level. After Master 2 sets a clock pulse to the high level at point “c” and detects the SCL line of the bus at the high level, Master 1 starts counting a clock pulse in the high level.

The clock pulse on the bus is determinded by the master device with the shortest high-level period and the master device with the longest low-level period from among those master devices connected to the bus.

(4) **Slave address and Address recognition mode specification**

When the 87C444/844 is used as a slave device, set the slave address and ALS to the I2CAR. Set "0" to the ALS for the address recognition mode.

(5) **Master/slave selection**

Set the MST (bit 7 in the SBICR2) to "1" for operating the 87C444/844 as a masterdevice. Reset the MST for operation as a slave device. The MST is cleared to "0" by the hardware after a stop condition on the I^2C bus is detected or arbitration is lost.

(6) **Transmitter/receiver selection**

Set the TRX (bit 6 in the SBICR2) to "1" for operating the 87C444/844 as a transmitter. Clear the TRX to "0"for operation as a receiver. When data with an addressing format is transferred in the slave mode, when a slave address with the same value that an I2CAR or a GENERAL CALL is received (all 8-bit data are "0" after a start condition), the TRX is set to "1" if the direction bit (R/$\overline{W}$) sent from the master device is "1", and is cleared to "0" if the bit is "0". In the master mode, after an acknowledge signal is returned from the slave device with the hardware, the TRX is set to "0" if a transmitted direction bit is "1", and set to "1" if it is "0". When an acknowledge signal is not returned, the current condition is maintained.

The TRX is cleared to "0" by the hardware after a stop condition on the I^2C bus is detected or arbitration is lost.

(7) **Start/Stop Condition generation**

A start condition and 8-bit of data that is set the slave address and the direction bit to a data buffer register are output on a bus by writing "1" to the MST, TRX, and BB when the BB (bit 5 in the SBICR2) is "0". It is necessary to set transmitted data to the data buffer register (SBIDBR) and set "1" to ACK beforehand.

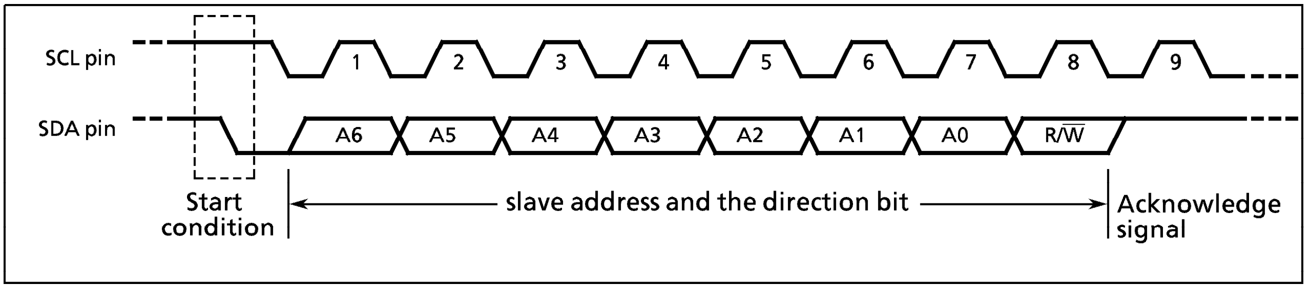


Figure 2-30. Start Condition Generation and Slave Address Generation

A stop condition is output on a bus by writing "1" to the MST , TRX and "0"to the BB when the BB is "1".

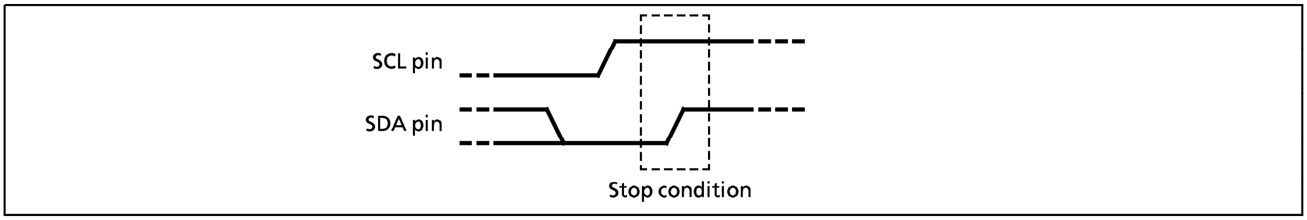


Figure 2-31. Stop Condition Generation

The bus condition can be indicated by reading the contents of the BB (bit 5 in the SBISR). The BB is set to "1" when a start condition on a bus is detected,and is cleared to "0" when a stop condition is detected.

(8) **Interrupt service request and cancel**

When a serial bus interface interrupt request (INTSBI) occurs, the PIN (bit 4 in the SBISR) is cleared to "0". During the time that the PIN is "0", the SCL pin is pulled down to the low level.

The PIN is cleared to "0" when 1-word of data is transmitted or received. Either writing/reading data to/from the SBIDBR sets the PIN to "1".

The time from the PIN being set to "1" until the SCL pin is released takes t_{LOW}.

In the address recognition mode (ALS = 0), the PIN is cleared to "0" when the received slave address is the same as the value set at the I2CAR or when a GENERAL CALL is received (all 8-bit data are "0" after a start condition). Although the PIN (bit 4 in the SBICR2) can be set to "1" by the program, the PIN is not set to "0" when "0" is written.

(9) **Serial bus interface operation mode selection**

The SBIM (bits 3, 2 in the SBICR2) is used to specify the serial bus interface operation mode. Set the SBIM to "10" when used in the I^2C bus mode.

Switch a mode to port after making sure that a bus is free.

(10) **Arbitration lost detection monitor**

Attached "ADDITIONAL INFORMATION" showed in SECTION 8 describe more detail and some notice for usage of I^2C Bus.

Please read carefully it if you are going to use I^2C Bus function in your application system.

Since more than one master device can exist simultaneously on a bus in the I^2C bus mode, a bus arbitration procedure is implemented in order to guarantee the contents of transferred data.

Data on the SDA line is used for bus arbitration of the I^2C bus.

The following shows an example of a bus arbitration procedure when two master devices exist simultaneously on the bus. Master 1 and Master 2 output the same data until point "a". After Master 1 outputs "1" and Master 2, "0", the SDA line of the bus is wire-AND and the SDA line is pulled down to the low level by Master 2. When the SCL line of the bus is pulled up at point "b", the slave device reads data on the SDA line, that is, data in Master 2. Data transmitted from Master 1 becomes invalid. The state in Master 1 is called "arbitration lost". A master device which loses arbitration releases the SDA pin in order not to effect data transmitted from other masters with arbitration. When more than one master sends the same data at the first word, arbitration occurs continuously after the second word.

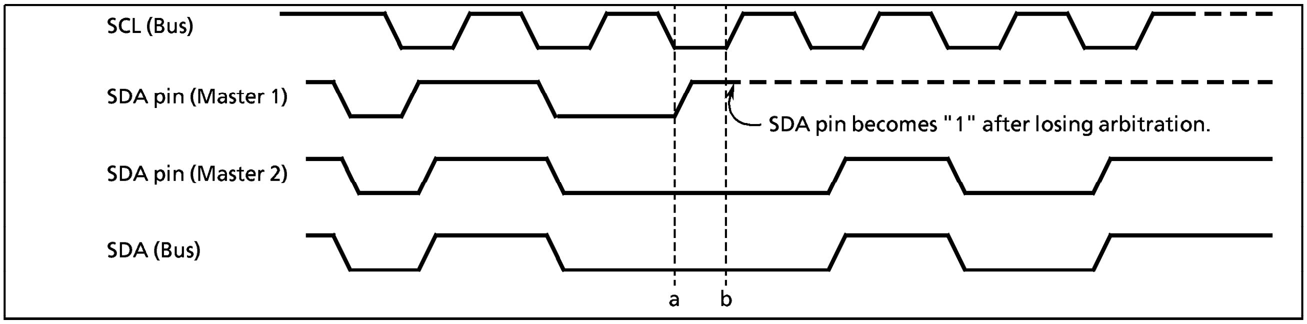


Figure 2-32. Arbitration Lost

The 87C444/844 compares levels of the SDA line of the bus with those of the 87C444/844 SDA pin at the rising edge of the SCL line. If the levels are unmatched, arbitration is lost and the AL (bit 3 in the SBISR) is set to "1".

When the AL is set to "1", the MST and TRX are reset to "0" and the mode is switched to a slave receiver mode. The 87C444/844 generates the clock pulse until data is transmitted when the AL is "1".

The AL is reset to "0" by writing/reading data to/from the SBIDBR or writing data to the SBICR2.

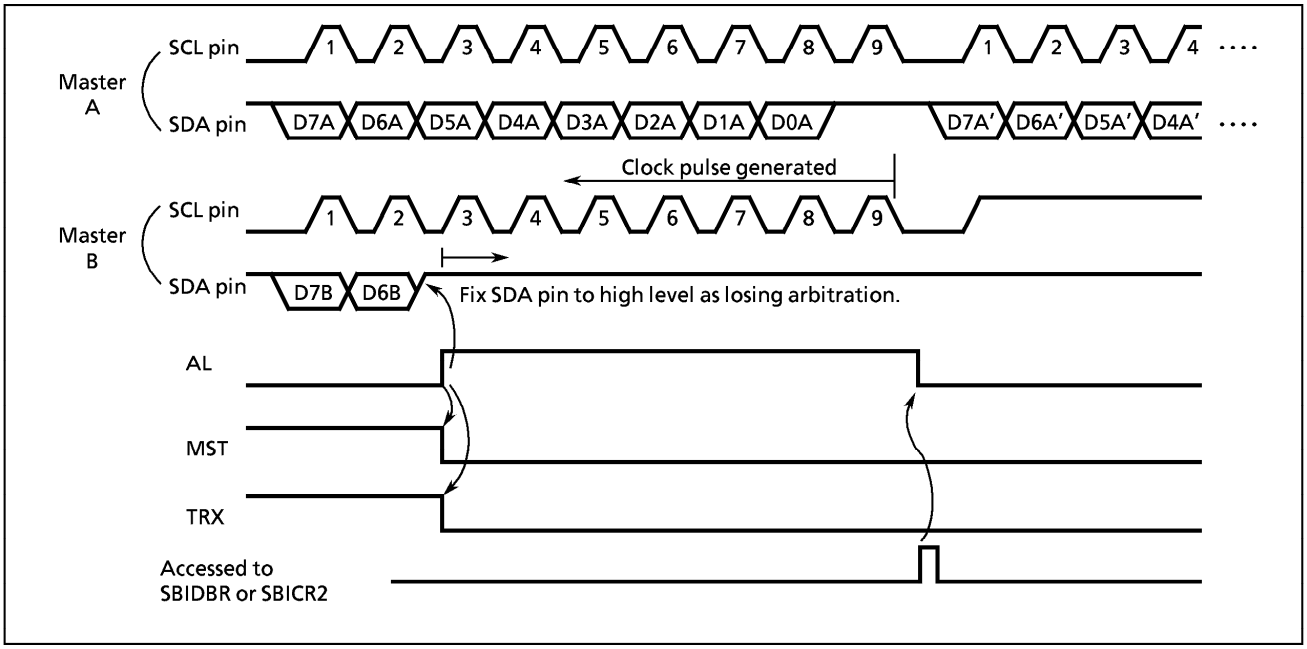

Master A
SCL pin
1 2 3 4 5 6 7 8 9 1 2 3 4 ····
SDA pin
D7A D6A D5A D4A D3A D2A D1A D0A D7A' D6A' D5A' D4A' ····
Clock pulse generated
Master B
SCL pin
1 2 3 4 5 6 7 8 9
SDA pin
D7B D6B
Fix SDA pin to high level as losing arbitration.
AL
MST
TRX
Accessed to SBIDBR or SBICR2

Figure 2-33. Example of when 87C444/844 is a Master device B

(11) **Slave address match detection monitor**

The AAS (bit 2 in the SBISR) is set to "1" in the slave mode, in the address recognition mode (ALS = 0), or when receiving a slave address with the same value that sets a GENERAL CALL or I2CAR. When the ALS is "1", the AAS is set to "1" after receiving the first 1-word of data. The AAS is reset by either writing/reading data to/from a data buffer register.

(12) **GENERAL CALL detection monitor**

The AD0 (bit 1 in the SBISR) is set to "1" in the slave mode, when all 8-bit data received immediately after a start condition are "0"(GENERAL CALL). The AD0 is reset when a start or stop condition is detected on the bus.

(13) **Last received bit monitor**

The SDA value stored at the rising edge of the SCL is sent to the LRB (bit 0 in the SBISR). When the contents of the LRB are read immediately after an INTSBI interrupt request is generated in the acknowledge mode, an ACK signal is read.

2.7.5 Data Transfer in I^{2}C bus Mode

(1) **Device Initialization**

Set the ACK and SCK in the SBICR1. Specify "0" to bits 7 to 5 and 3.
Set a slave address and the ALS (ALS = 0 when an addressing format) to the I2CAR.
For specifying the default setting to a slave receiver mode, assign "0" to the MST, TRX, and BB in the SBICR2; "1" to the PIN; "10" to the SBIM; and "0" to bits 0 and 1.

(2) Start Condition and Slave Address Generation

Observe a bus free status (when BB = 0).

Set the ACK to "1" and specify a slave address and a direction bit to be transmitted to the SBIDBR. When writing "1" to the MST, TRX, and BB, the slave address and the direction bit which are set to the SBIDBR and the start condition are output on the bus. A slave device receives these data and pulls down the SDA line of the bus to the low level at the acknowledge signal timing. An INTSBI interrupt request occurs at the 9th falling edge of the SCL clock cycle, and the PIN is cleared to "0". The SCL pin is pulled down to the low level while the PIN is "0". When an interrupt request occurs, the TRX changes by the hardware according to the direction bit only when an acknowledge signal is returned from the slave device.

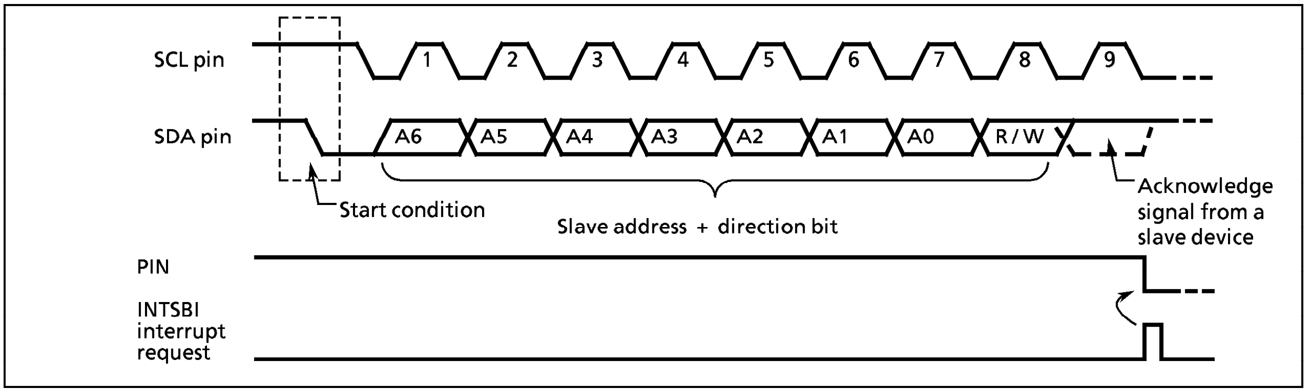


Figure 2-34. Start Condition Generation and Slave Address Transfer

(3) 1-word Data Transfer

Test the MST by the INTSBI interrupt process after a 1-word data transfer is concluded, and determine whether the mode is a master or slave.

a. When the MST is "1" (Master mode)

Test the TRX and determine whether the mode is a transmitter or receiver.

① When the TRX is "1" (Transmitter mode)

Test the LRB. When the LRB is "1", a receiver does not request data. Implement the process to generate a stop condition (described later) and terminate data transfer.

When the LRB is "0", the receiver requests new data. When the next transmitted data is other than 8 bits, set the BC and write the transmitted data to the SBIDBR. After writing the data, the PIN becomes "1", a serial clock pulse is generated for transferring a new 1-word of data from the SCL pin, and then the 1-word data is transmitted. After the data is transmitted, an INTSBI interrupt request occurs. The PIN becomes "0" and the SCL pin is pulled down to the low level. If the data to be transferred is more than one word in length, repeat the procedure from the LRB test above.

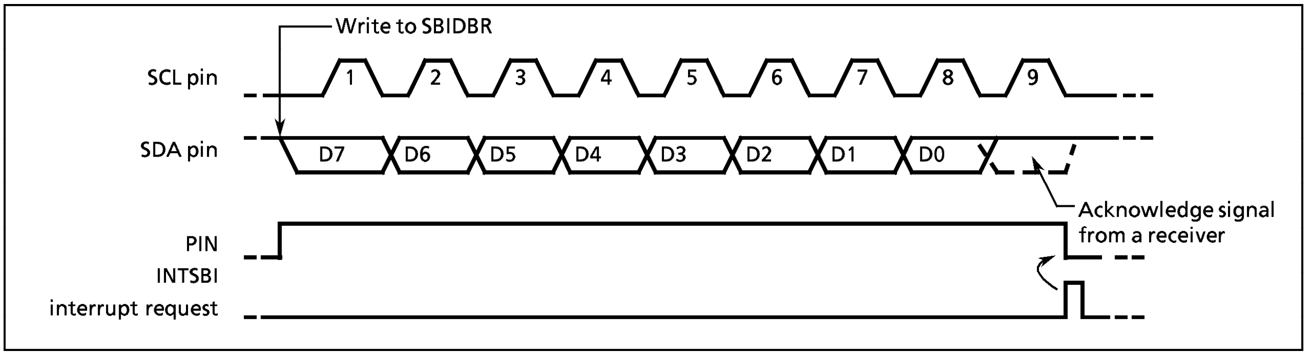


Figure2-35. Example of when BC = "000", ACK = "1"

② When the TRX is "0" (Receiver mode)
When the next transmitted data is other than 8 bits, set the BC again. Set the ACK to "1" and read the received data from the SBIDBR (data which is read immediately after a slave address is sent is undefined). After the data is read, the PIN becomes "1". The 87C444/844 outputs a serial clock pulse to the SCL to transfer new 1-word of data and sets the SDA pin to "0" at the acknowledge signal timing.
An INTSBI interrupt request then occurs and the PIN becomes "0". Then the 87C444/844 pulls down the SCL line of the bus to low level. The 87C444/844 outputs a clock pulse for 1-word of data transfer and the acknowledge signal each time that received data is read from the SBIDBR.

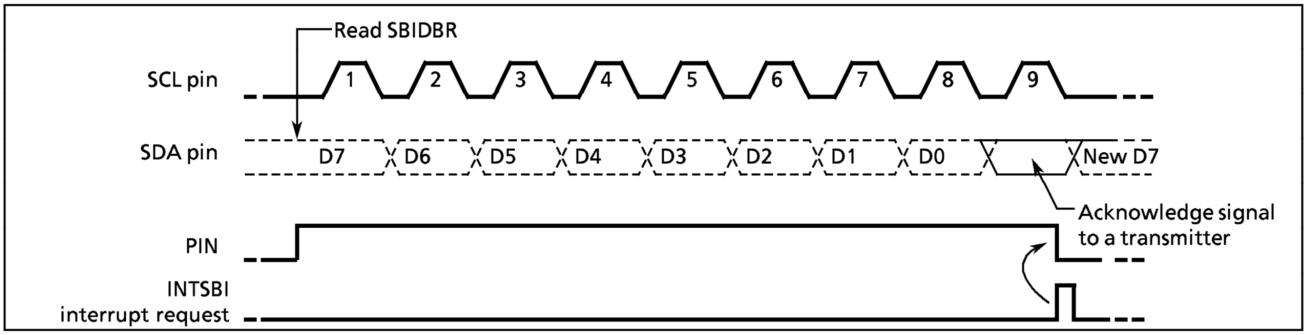


Figure 2-36. Example of when BC = "000", ACK = "1"

In order to terminate transmitting data to a transmitter, reset the ACK before reading data which is 1 word before the last data to be received. The last data does not generate a clock pulse for the acknowledge signal. After the data is transmitted and an interrupt request has occurred, set the BC to "001" and read the data. The 87C444/844 generates a clock pulse for a 1-bit data transfer. Since the master device is a receiver, the SDA line of the bus keeps the high level. The transmitter receives the high-level signal as an ACK signal. The receiver indicates to the transmitter that data transfer is complete.
After 1-bit data is received and an interrupt request has occurred, the 87C444/844 generates a stop condition and terminates data transfer.

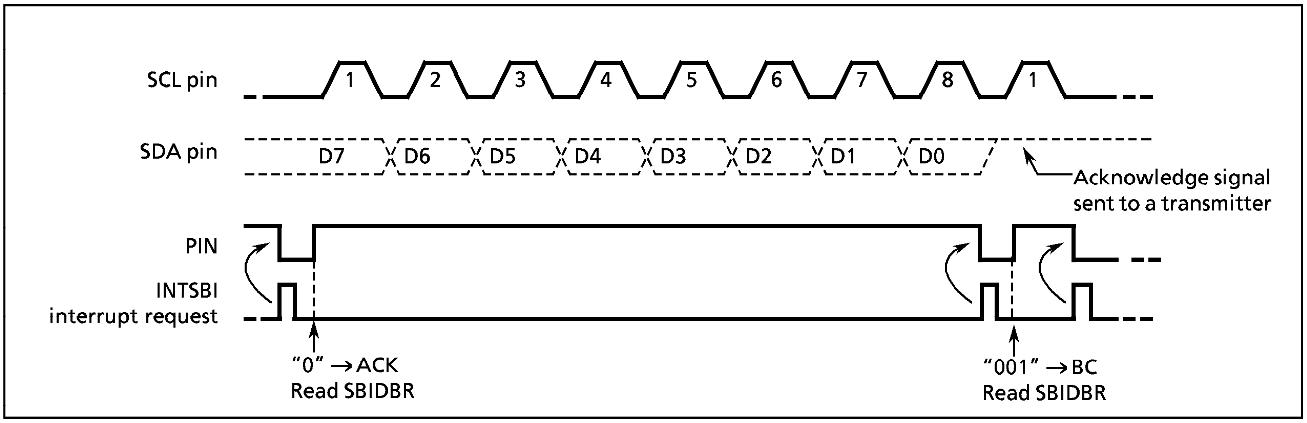


Figure 2-37. Termination of data transfer in master receiver mode

b. When the MST is "0" (Slave mode)

In the slave mode, an INTSBI interrupt request occurs when the 87C444/844 receives a slave address or a GENERAL CALL from the master device, or when a GENERAL CALL is received and data transfer is complete after matching a received slave address. In the master mode, the 87C444/844 operates in a slave mode if it is losing arbitration. An INTSBI interrupt request occurs when word data transfer terminates after losing arbitration. When an INTSBI interrupt request occurs, the PIN (bit 4 in the SBICR2) is reset, and the SCL pin is pulled down to the low level. Either reading/writing from/to the SBIDBR or setting the PIN to "1" releases the SCL pin after taking t_{LOW} time.

In the slave mode, the 87C444/844 operates either in normal slave mode or in slave mode after losing arbitration.

The 87C444/844 tests the AL (bit 3 in the SBISR), the TRX (bit 6 in the SBISR), the AAS (bit 2 in the SBISR), and the AD0 (bit 1 in the SBISR) and implements processes according to conditions listed in the next table.

Table 2-4. Operation in the Slave Mode

TRX	AL	AAS	AD0	Conditions	Process
1	1	1	0	The 87C444/844 loses arbitration when transmitting a slave address and receives a slave address of which the value of the direction bit sent from another master is "1".	Set the number of bits in 1 word to the BC and write transmitted data to the SBIDBR.
	0	1	0	In the slave receiver mode, the 87C444/844 receives a slave address of which the value of the direction bit sent from the master is "1".	
		0	0	In the slave transmitter mode, 1-word data is transmitted.	Test the LRB. If the LRB is set to "1", set the PIN to "1" since the receiver does not request further data. Then, reset the TRX to release the bus. If the LRB is set to "0", set the number of bits in a word to the BC and write transmitted data to the SBIDBR since the receiver requests further data.
0	1	1	1/0	The 87C444/844 loses arbitration when transmitting a slave address and receives a slave address or GENERAL CALL of which the value of the direction bit sent from another master is "0".	Read the SBIDBR for setting the PIN to "1" (reading dummy data) or write "1" to the PIN.
		0	0	The 87C444/844 loses arbitration when transmitting a slave address or data and terminates transferring word data.	
	0	1	1/0	In the slave receiver mode, the 87C444/844 receives a slave address or GENERAL CALL of which the value of the direction bit sent from the master is "0".	
		0	1/0	In the slave receiver mode, the 87C444/844 terminates receiving of 1 word data.	Set the number of bits in a word to the BC and read received data from the SBIDBR.

(4) Stop Condition Generation

Writing "1" to the MST, TRX, and PIN, and "0" to the BB generates a stop condition on the bus.

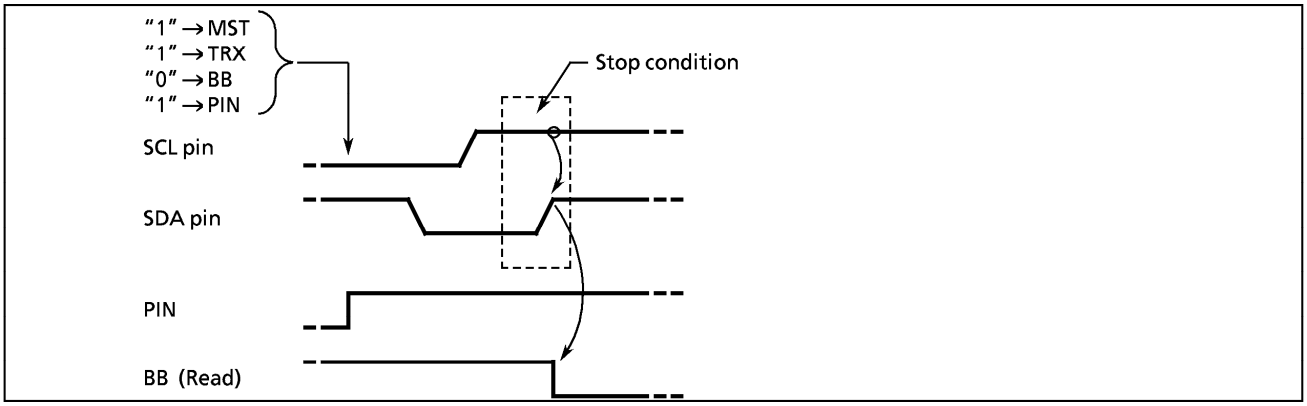


Figure 2-38. Stop Condition Generation

(5) Restart

Restart is used to change the direction of data transfer between a master device and a slave device during transferring data. The following explains how to restart when the 87C444/844 is in the master mode.

Specify "0" to the MST, TRX, and BB and "1" to the PIN and release the bus. The SDA pin retains the high level and the SCL pin is released. Since a stop condition is not generated on the bus, the bus is assumed to be in a busy state from other devices. Test the BB until it becomes "0" to check that the SCL pin of the 87C444/844 is released. Test the LRB until it becomes "1" to check that the SCL line of the bus is not pulled down to the low level by other devices. After confirming that the bus stays in a free state, generate a start condition with procedure (2).

In order to meet setup time when restarting, take at least 4.7 microseconds of waiting time by software from the time of restarting to confirm that the bus is free until the time to generate the start condition.

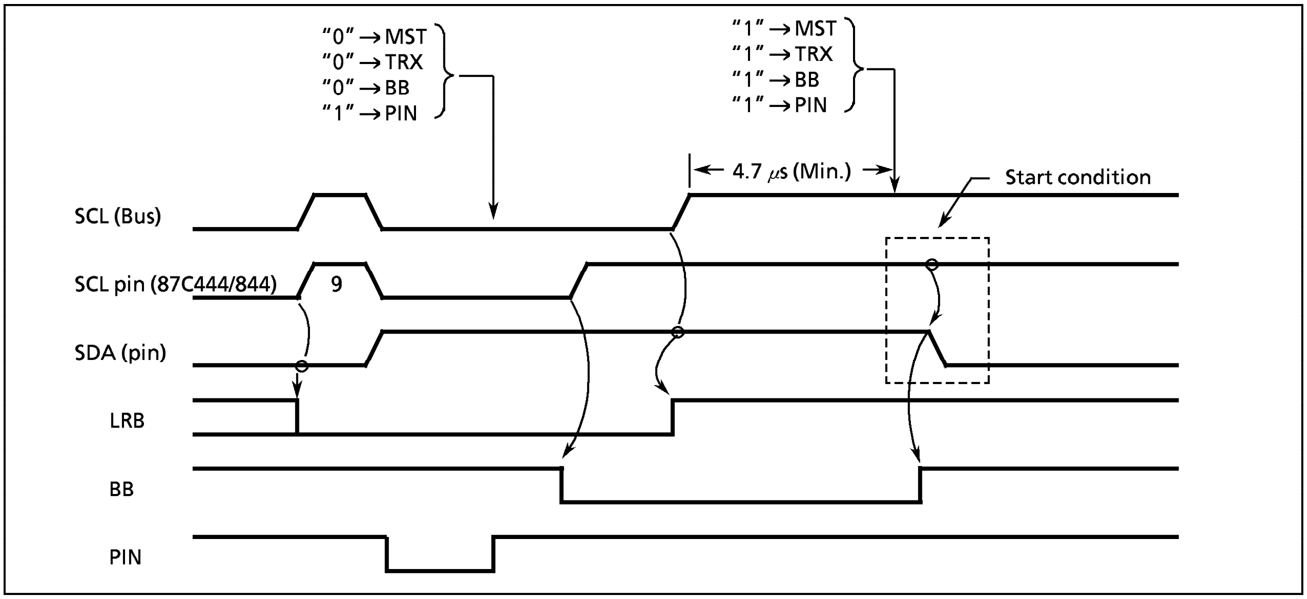


Figure 2-39. Timing diagram when restarting the 87C444/844

2.7.6 Clocked-synchronous 8-bit SIO Mode Control

The following registers are used for control and operation status monitoring when using the serial bus interface (SBI) in the clocked-synchronous 8-bit SIO mode.

Serial Bus Interface Control Register 1

SBICR1 (0020$_H$)	7	6	5	4	3	2	1	0	
	SIOS	SIO INH	SIOM		"0"	SCK			(Initial value 0000 *000)

SIOS	Indicate transfer start/stop	0 : Stop 1 : Start	write only
SIOINH	Continue/abort transfer	0 : Continue transfer 1 : Abort transfer (automatically cleared after abort)	
SIOM	Transfer mode select	00 : 8-bit transmit mode 01 : reserved 10 : 8-bit transmit/receive mode 11 : 8-bit receive mode	
SCK	Serial clock select	000 : fc/2^5 (250kHz) 001 : fc/2^6 (125kHz) 010 : fc/2^7 (62.5kHz) 011 : fc/2^8 (31.25kHz) 100 : fc/2^9 (15.62kHz) 101 : fc/2^{10} (7.81kHz) 110 : fc/2^{11} (3.90kHz) at fc = 8MHz (Output on $\overline{SCK0}$ pin) 111 : External clock (input from $\overline{SCK0}$ pin)	

*Note 1 : fc ; high-frequency clock [Hz], * ; don't care*
Note 2 : Set the SIOS to "0" when setting the transfer mode or serial clock.

Serial Bus Interface Data Buffer Register

SBIDBR (0021$_H$)	7	6	5	4	3	2	1	0	
									R/W

Serial Bus Interface Control Register 2

SBICR2 (0023$_H$)	7	6	5	4	3	2	1	0	
	"0"	"0"	"0"	"1"	SBIM		"0"	"0"	(Initial value **** 00**)

SBIM	Serial bus interface operation mode selection	00 : Port mode (serial bus interface output disable) 01 : SIO mode 10 : I^{2}C bus mode 11 : reserved	write only

*Note 1 : * ; don't care*
Note 2 : Switch a mode to port after data transfer is complete.

Serial Bus Interface Status Register

SBISR (0023$_H$)	7	6	5	4	3	2	1	0
	"1"	"1"	"1"	"1"	SIOF	SEF	"1"	"1"

SIOF	Serial transfer operating status monitor	0 : Transfer terminated 1 : Transfer in process	read only
SEF	Shift operating status monitor	0 : Shift operation terminated 1 : Shift operation in process	

Figure 2-40. Serial Bus Interface Control Register 1/Serial Bus Interface Data Buffer Register/Serial Bus Interface Control Register 2/Serial Bus Interface Status Register in SIO mode

(1) Serial Clock

a. Clock source

The SCK (bits 2 to 0 in the SBICR1) is used to select the following functions.

① Internal Clock

In an internal clock mode, any of seven frequencies can be selected. The serial clock is output to the outside on the $\overline{\text{SCK0}}$ pin. The $\overline{\text{SCK0}}$ pin becomes a high level when data transfer starts. When writing (in the transmit mode) or reading (in the receive mode) data cannot follow the serial clock rate, an automatic-wait function is executed to stop the serial clock automatically and hold the next shift operation until reading or writing is complete.

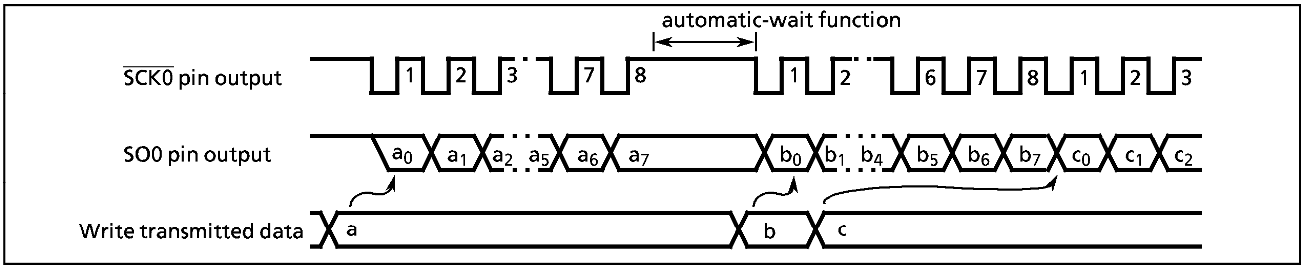


Figure 2-41. Automatic-wait Function

② External clock (SCK = "111")

An external clock supplied to the $\overline{\text{SCK0}}$ pin is used as the serial clock. In order to ensure shift operation, a pulse width of at least 4 machine cyles is required for both high and low levels in the serial clock. The maximum data transfer frequency is 250kHz (when fc = 8MHz).

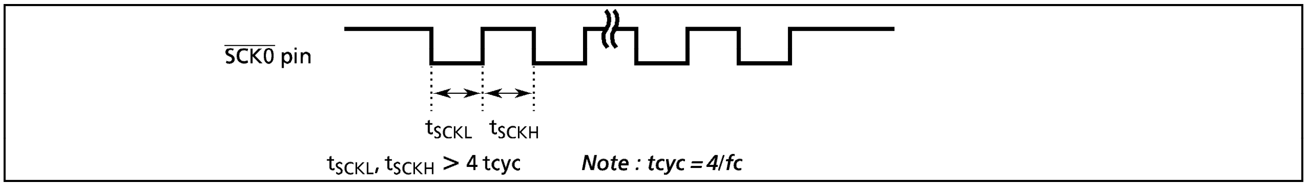


Figure 2-42. Maximum Data Transfer Freguency When External Clock Input

b. Shift edge

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

① Leading edge shift
Data is shifted on the leading edge of the serial clock (at a falling edge of the $\overline{\text{SCK0}}$ pin input/output).

② Trailing edge shift
Data is shifted on the trailing edge of the serial clock (at a rising edge of the $\overline{\text{SCK0}}$ pin input/output).

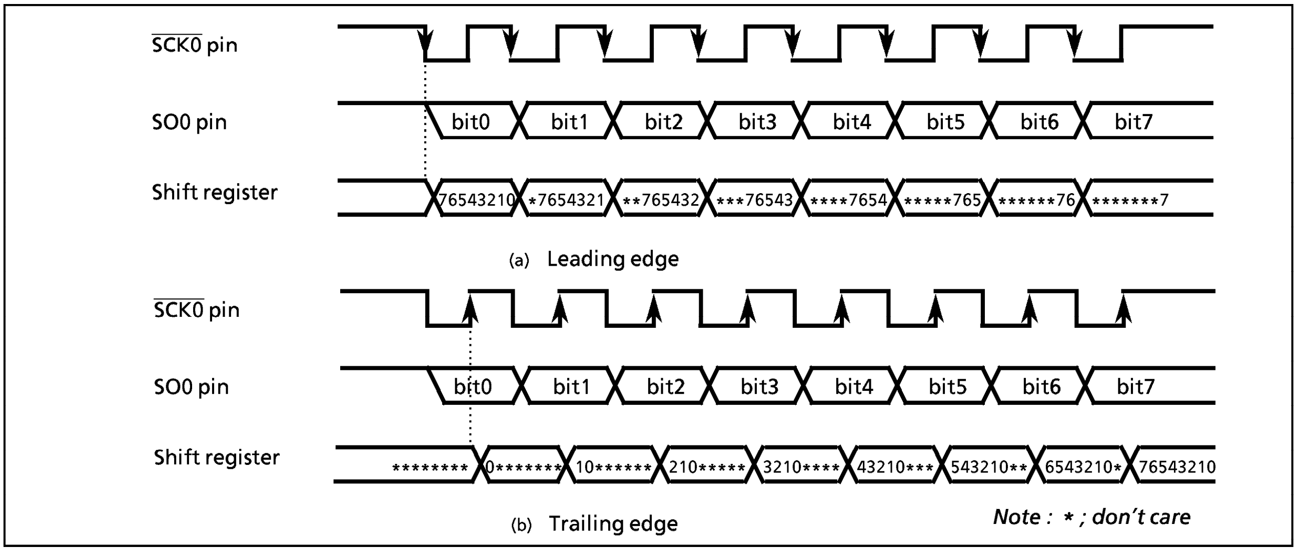


Figure 2-43. Shift Edge

(2) Transfer mode

The SIOM (bits 5 and 4 in the SBICR1) is used to select a transmit, receive, or transmit/receive mode.

a. 8-bit transmit mode

Set a control register to a transmit mode and write transmit data to the SBIDBR.
After the transmit data is written, set the SIOS to "1" to start data transfer. The transmitted data is transferred from the SBIDBR to the shift register and output to the SO0 pin in synchronous with the serial clock, starting from the least significant bit (LSB). When the transmit data is transferred to the shift register, the SBIDBR becomes empty. The INTSBI (buffer empty) interrupt request is generated to request new data.
When the internal clock is used, the serial clock will stop and automatic-wait function will be initiated if new transmit data is not loaded to the data buffer register after the specified 8-bit data is transmitted. When new data is written, automatic-wait function is canceled.
When the external clock is used, data should be written to the SBIDBR before new data is shifted. The transfer speed is determined by the maximum delay time between the time when an interrupt request is generated and the time when data is written to the SBIDBR by the interrupt service program.
Transmitting data is ended by clearing the SIOS to "0" by the buffer empty interrupt service program or setting the SIOINH to "1". When the SIOS is cleared, the transmitted mode ends when all data is output. In order to confirm if data is surely transmitted by the program, set the SIOF (bit 3 in the SBISR) to be sensed. The SIOF is cleared to "0" when transmitting is complete. When the SIOINH is set, transmitting data stops. The SIOF turns "0".
When the external clock is used, it is also necessary to clear the SIOS to "0" before new data is shifted; otherwise, dummy data is transmitted and operation ends.

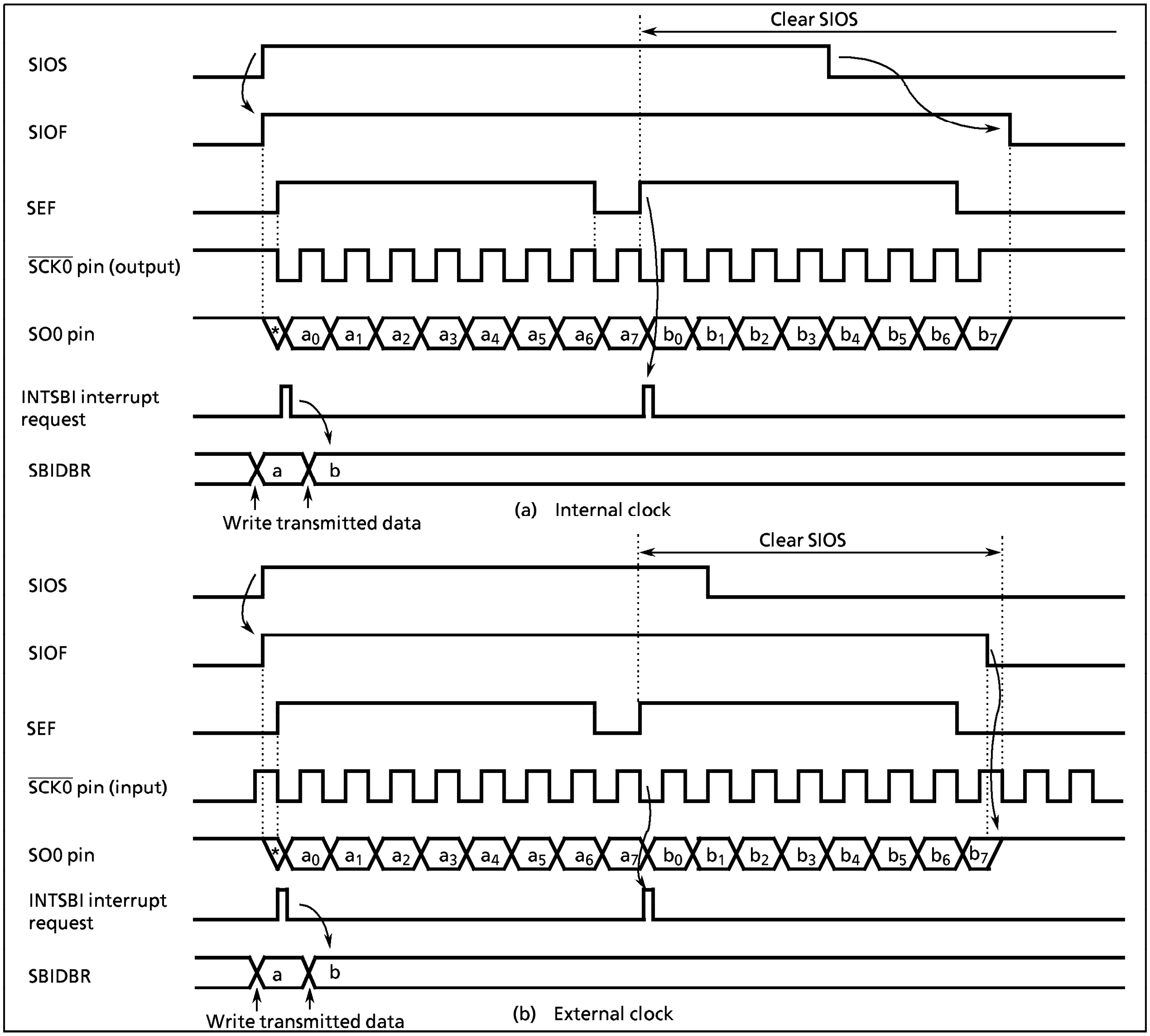


Figure 2-44. Transfer Mode

Example: Program to stop transmitting data (when external clock is used)

```
STEST1  :    TEST      (SBISR) . SEF                ;  If SEF = 1      then   loop
             JRS       F , STEST1
STEST2  :    TEST      (P3) . 6                     ;  If SCK0 = 0     then   loop
             JRS       T , STEST2
             LD        (SBICR1) , 00000111B         ;  SIOS ← 0
```

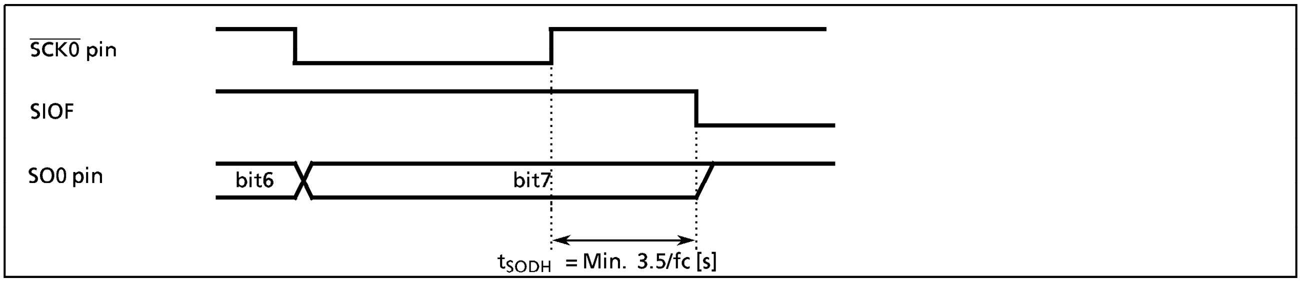


Figure 2-45. Transmitted Data Hold Time at end of transmit

b. 8-bit Receive Mode

Set the control register to receive mode and the SIOS to "1" for switching to receive mode. Data is received from the SI0 pin to the shift register in synchronous with the serial clock, starting from the least significant bit (LSB). When the 8-bit data is received, the data is transferred from the shift register to the SBIDBR. The INTSBI (buffer full) interrupt request is generated to request of reading the received data. The data is then read from the SBIDBR by the interrupt service program.

When the internal clock is used, the serial clock will stop and automatic-wait function will be initiated until the received data is read from the SBIDBR.

When the external clock is used, since shift operation is synchronized with the clock pulse provided externally, the received data should be read from the SBIDBR before next serial clock is input. If the received data is not read, further data to be received is canceled. The maximum transfer speed when the external clock is used is determined by the delay time between the time when an interrupt request is generated and the time when received data is read.

Receiving data is ended by clearing the SIOS to "0" by the buffer full interrupt service program or setting the SIOINH to "1". When the SIOS is cleared, received data is transferred to the SBIDBR in complete blocks. The received mode ends when the transfer is complete. In order to confirm if data is surely received by the program, set the SIOF (bit 3 in the SBIDBR) to be sensed. The SIOF is cleared to "0" when receiving is complete. After confirming that receiving has ended, the last data is read. When the SIOINH is set, receiving data stops. The SIOF turns "0" (the received data becomes invalid, therefore no need to read it).

Note : When the transfer mode is switched, the SBIDBR contents are lost. In case that the mode needs to be switched, receiving data is concluded by clearing the SIOS to "0", read the last data, and then switch the mode.

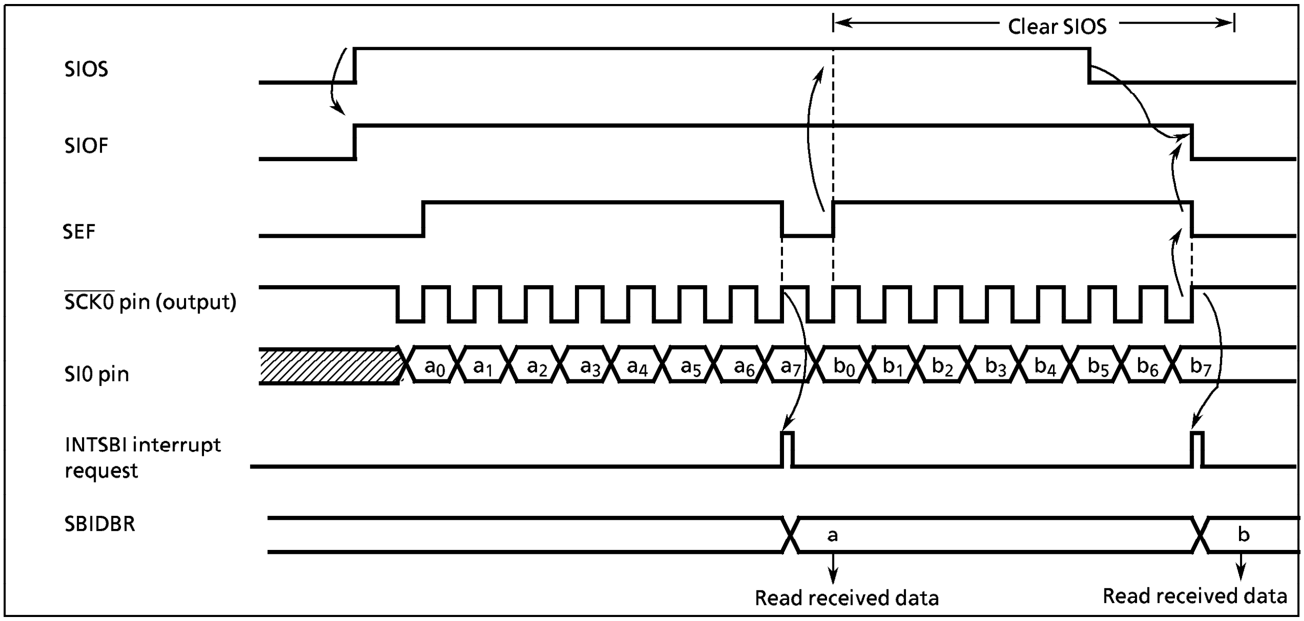


Figure 2-46. Receive Mode (Example : Internal clock)

c. 8-bit Transmit/Receive Mode

Set a control register to a transmit/receive mode and write data to the SBIDBR. After the data is written, set the SIOS to "1" to start transmitting/receiving. When transmitting, the data is output from the SO0 pin on the leading edges in synchronous with the serial clock, starting from the least significant bit (LSB). When receiving, the data is input to the SI0 pin on the trailing edges of the serial clock. 8-bit data is transferred from the shift register to the SBIDBR, and the INTSBI interrupt request occurs. The interrupt service program reads the received data from the data buffer register and writes data to be transmitted. The SBIDBR is used for both transmitting and receiving. Transmitted data should always be written after received data is read.

When the internal clock is used, automatic-wait function is initiated until received data is read and next data is written.

When the external clock is used, since the shift operation is synchronized with the external clock, received data is read and transmitted data is written before new shift operation is executed. The maximum transfer speed when the external clock is used is determined by the delay time between the time when an interrupt request is generated and the time when received data is read and transmitted data is written.

Transmitting/receiving data is ended by clearing the SIOS to "0" by the INTSBI interrupt service program or setting the SIOINH to "1". When the SIOS is cleared, received data is transferred to the SBIDBR in complete blocks. The transmit/receive mode ends when the transfer is complete. In order to confirm if data is surely transmitted/received by the program, set the SIOF (bit3 in the SBISR) to be sensed. The SIOF becomes "0" after transmitting/receiving is complete. When the SIOINH is set, transmitting/receiving data stops. The SIOF turns "0".

Note : When the transfer mode is switched, the SBIDBR contents are lost. In case that the mode needs to be switched, conclude transmitting/receiving data by clearing the SIOS to "0", read the last data, and then switch the transfer mode.

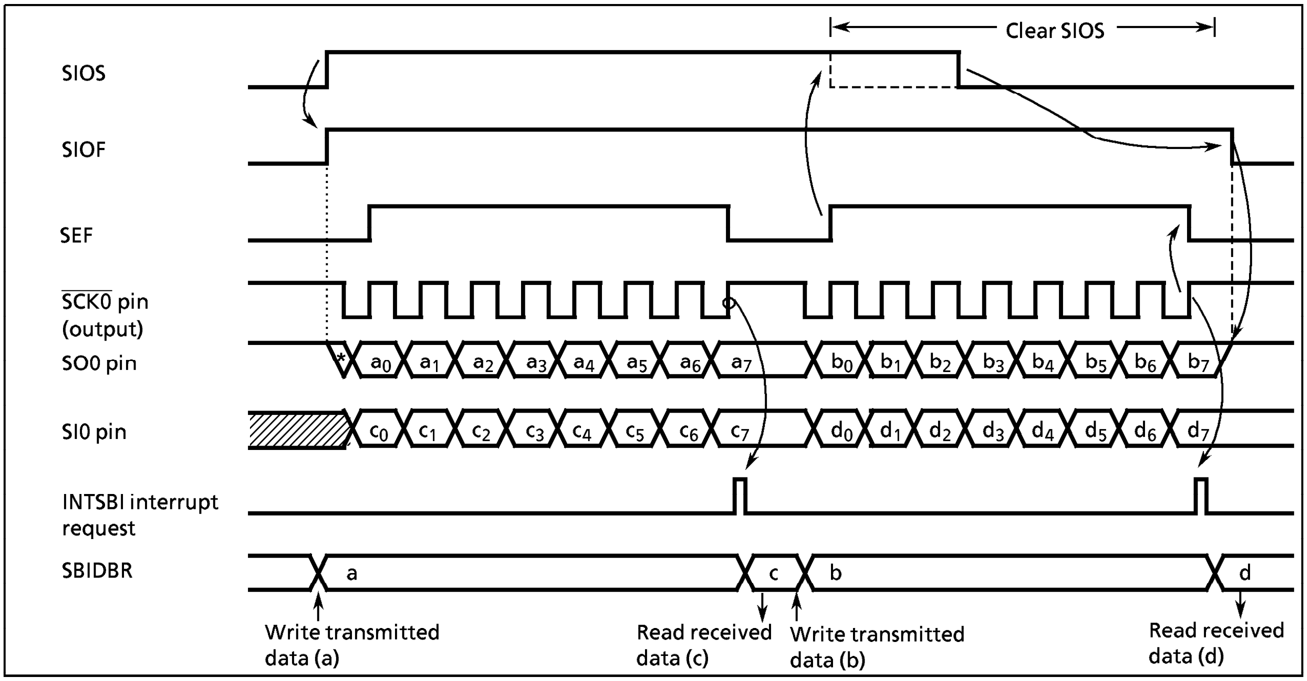


Figure 2-47. Transmit/Receive Mode (Example : Internal clock)

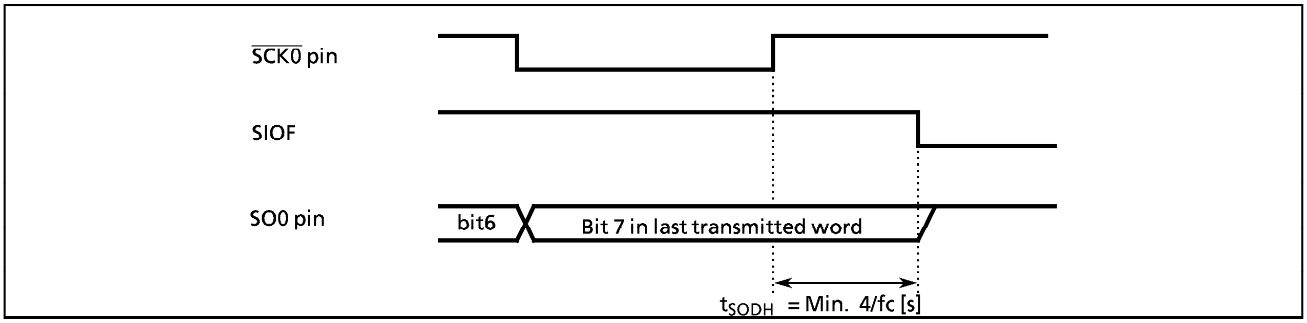


Figure 2-48. Transmitted Data Hold Time at end of transmit/receive

2.8 Serial Bus Interface (SIO1)

The 87C444/844 each have a clocked-synchronous 8-bit serial bus interface (SIO1). Each serial bus interface has an 8-byte transmit and receive data buffer that can automatically and continuously transfer up to 64 bits of data.

The serial bus interface is connected to external devices via pins P62 (SI1), P61 (SO1), P60 ($\overline{\text{SCK1}}$) for SIO1. The serial bus interface pins are also used as port P6. When used as serial bus interface pins, the output latches of these pins should be set to "1". In the transmit mode, pins P62 can be used as normal I/O ports, and in the receive mode, the pins P61 can be used as normal I/O ports.

2.8.1 Configuration

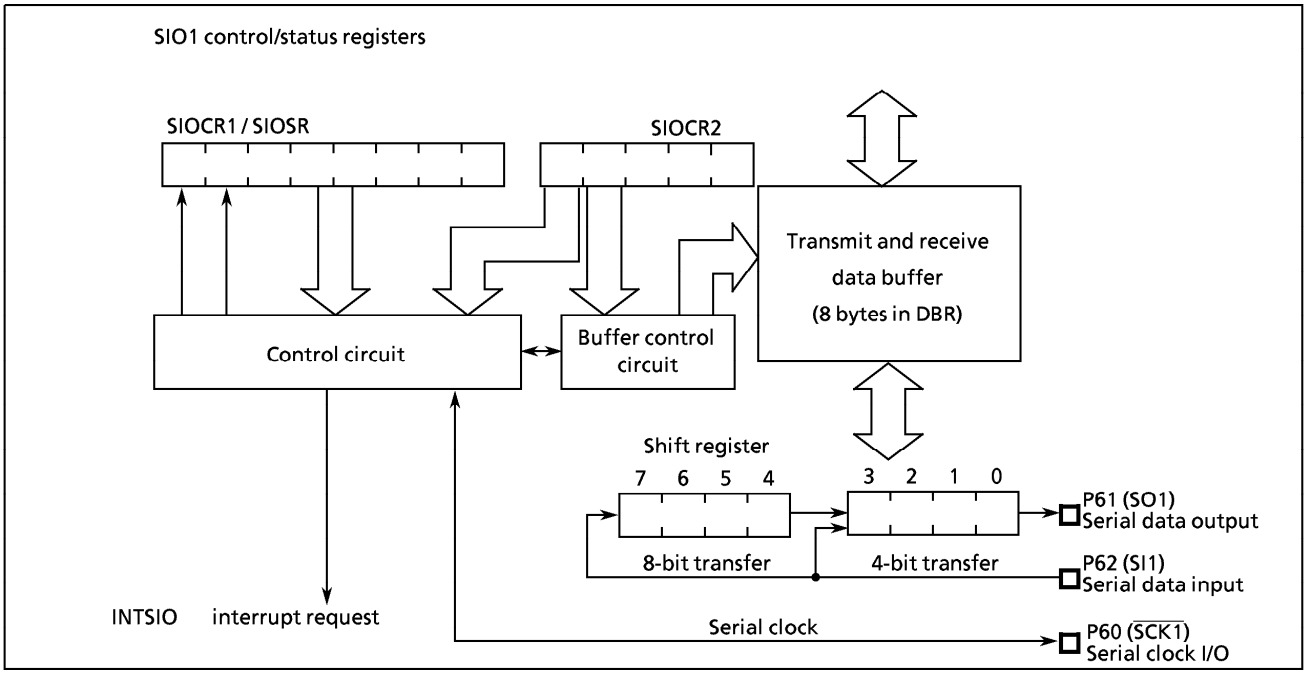

SIO1 control/status registers
SIOCR1 / SIOSR
SIOCR2
Transmit and receive data buffer (8 bytes in DBR)
Control circuit
Buffer control circuit
Shift register
7 6 5 4
3 2 1 0
8-bit transfer
4-bit transfer
P61 (SO1) Serial data output
P62 (SI1) Serial data input
P60 ($\overline{\text{SCK1}}$) Serial clock I/O
Serial clock
INTSIO interrupt request

Figure 2-49. Serial Interfaces

2.8.2 Control

The serial bus interface is controlled by SIO1 control registers (SIOCR1/SIOCR2). The serial bus interface status can be determined by reading SIO1 status register (SIOSR).

The transmit and receive data buffer is controlled by the BUF (bits 2-0 in SIOCR2). The data buffer is assigned to addresses $0FF0_H$ - $0FF7_H$ for SIO1 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 (INTSIO) 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 SIOCR2).

SIO1 Control Register 1

SIOCR1
(0028$_H$)

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

(Initial value : 0000 0000)

SIOS	Indicate transfer start/stop	0 : Stop 1 : Start	write only
SIOINH	Continue/abort transfer	0 : Continue transfer 1 : Abort transfer (automatically cleared after abort)	
SIOM	Transfer mode select	000 : 8-bit transmit mode 010 : 4-bit transmit mode 100 : 8-bit transmit / receive mode 101 : 8-bit receive mode 110 : 4-bit receive mode	
SCK	Serial clock select	000 : Internal clock $fc/2^{13}$ 001 : Internal clock $fc/2^{8}$ 010 : Internal clock $fc/2^{6}$ 011 : Internal clock $fc/2^{5}$ (Output on $\overline{\text{SCK1}}$ pin) 111 : External clock (input from $\overline{\text{SCK1}}$ pin)	

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

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

Note 3 : SIOCR1 is write-only register, which cannot access any of read-modify-write instruction such as bit operate, etc.

SIO1 Status Register

SIOSR
(0028$_H$)

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

SIOF	Serial transfer operating status monitor	0 : Transfer terminated 1 : Transfer in process	read only
SEF	Shift operating status monitor	0 : Shift operation terminated 1 : Shift operation in process	

SIO1 Control Register 2

SIOCR2
(0029$_H$)

7	6	5	4	3	2	1	0
			WAIT		BUF		

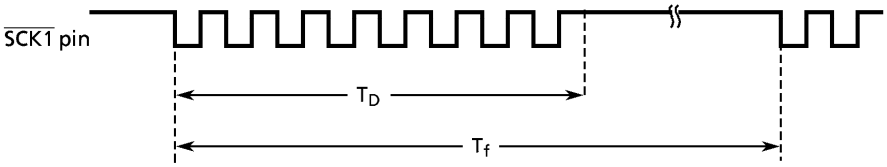
(Initial value: ***0 0000)

WAIT	Wait control	00 : $T_f = T_D$ 01 : $T_f = 2T_D$ 10 : $T_f = 4T_D$ 11 : $T_f = 8T_D$	write only
BUF	Number of transfer words	Buffer address used SIO1 000 : 1 word transfer 0FF0$_H$ 001 : 2 words transfer 0FF0 - 0FF1$_H$ 010 : 3 words transfer 0FF0 - 0FF2$_H$ 011 : 4 words transfer 0FF0 - 0FF3$_H$ 100 : 5 words transfer 0FF0 - 0FF4$_H$ 101 : 6 words transfer 0FF0 - 0FF5$_H$ 110 : 7 words transfer 0FF0 - 0FF6$_H$ 111 : 8 words transfer 0FF0 - 0FF7$_H$	

*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.

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

Note 7 : SIOCR2 is write-only register which cannot access any of in read-modify-write instruction such as bit operate, etc.

Figure 2-50. SIO1 Control Registers and Status Register

(1) Serial Clock

a. Clock Source

SCK (bits 2 - 0 in SIOCR1) 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 $\overline{SCK1}$ pin. The $\overline{SCK1}$ 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 2-5. Serial Clock Rate

Source clock	Maximum time setting
	At fc = 8 MHz
fc / 2^{13} [Hz]	0.95 K bit / s
fc / 2^8	30.5
fc / 2^6	122
fc / 2^5	244

Note : 1Kbit = 1024 bit

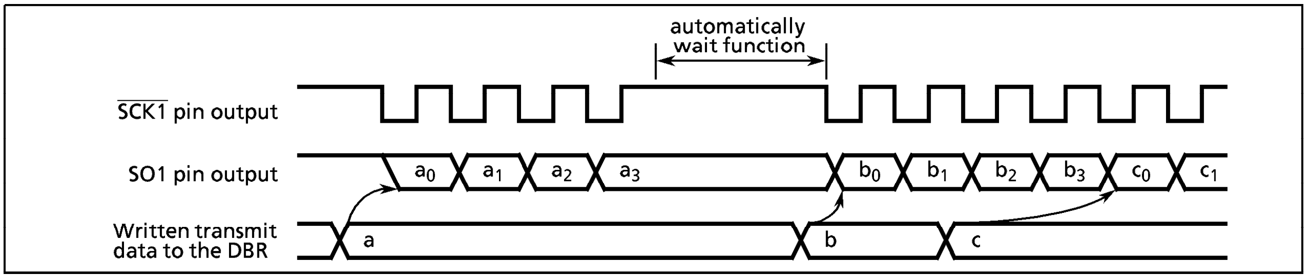
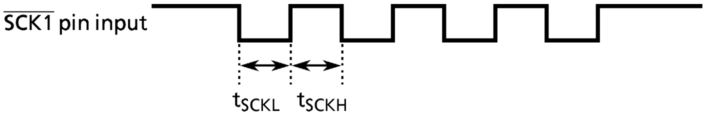


Figure 2-51. Clock Source (Internal Clock)

② External Clock

An external clock connected to the $\overline{\text{SCK1}}$ pin is used as the serial clock. In this case, the P60 ($\overline{\text{SCK1}}$) 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).



t_{SCKL}, t_{SCKH} > 4 tcyc *Note : tcyc = 4/fc*

b. Shift edge

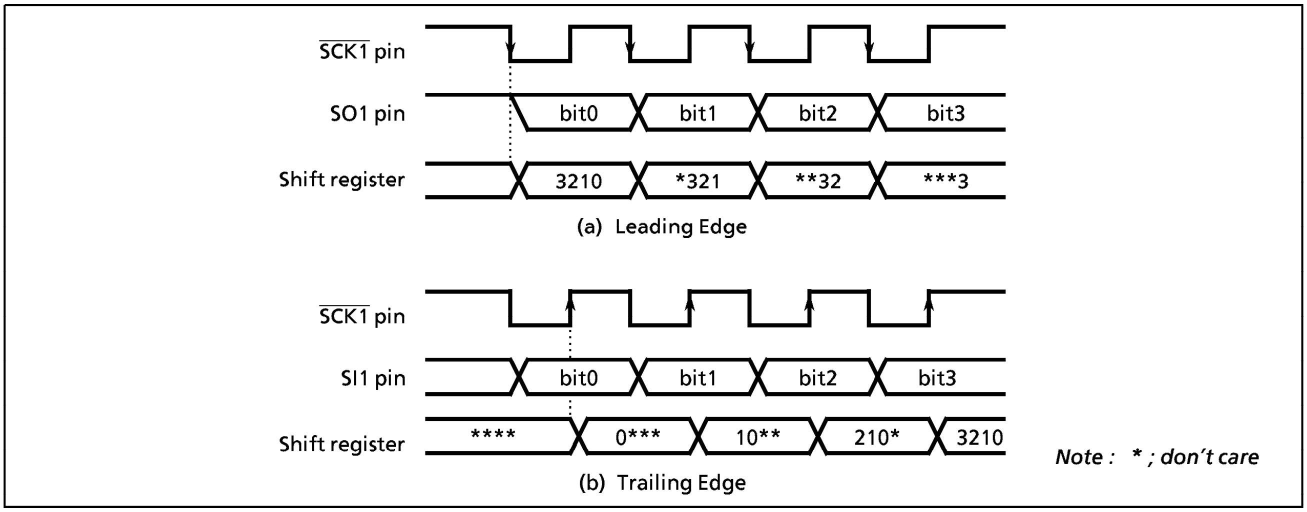
The leading edge is used to transmit, and the trailing edge is used to receive.

① Leading Edge

Transmitted data are shifted on the leading edge of the serial clock (falling edge of the $\overline{\text{SCK1}}$ pin input/output).

② Trailing Edge

Received data are shifted on the trailing edge of the serial clock (rising edge of the $\overline{\text{SCK1}}$ pin input/output).



(a) Leading Edge

(b) Trailing Edge

*Note : * ; don't care*

Figure 2-52. 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 BUF 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.

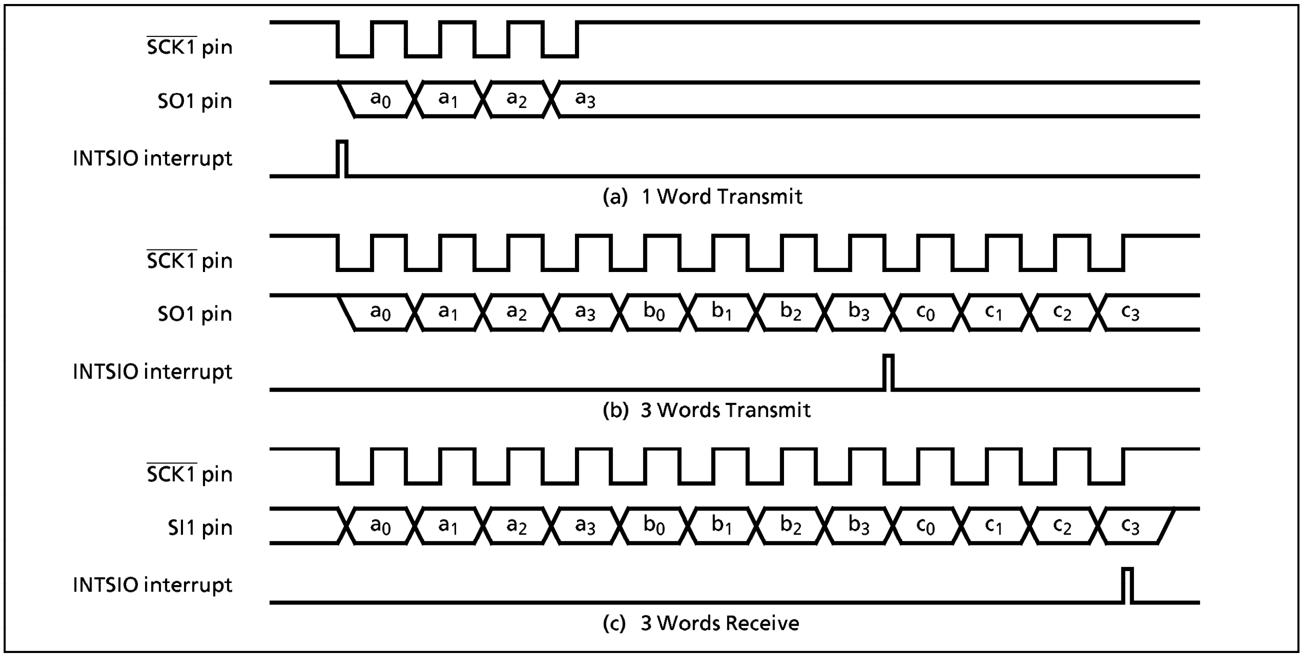


Figure 2-53. Number of Bits to Transfer (Example : 4-bit serial transfer)

2.8.3 Transfer Mode

SIOM (bits 5 - 3 in SIOCR1) is used to select the transmit, receive, or transmit/receive mode.

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

In these modes, the SIOCR1 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 SO1 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 SIO1 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 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 SIOSR) 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.

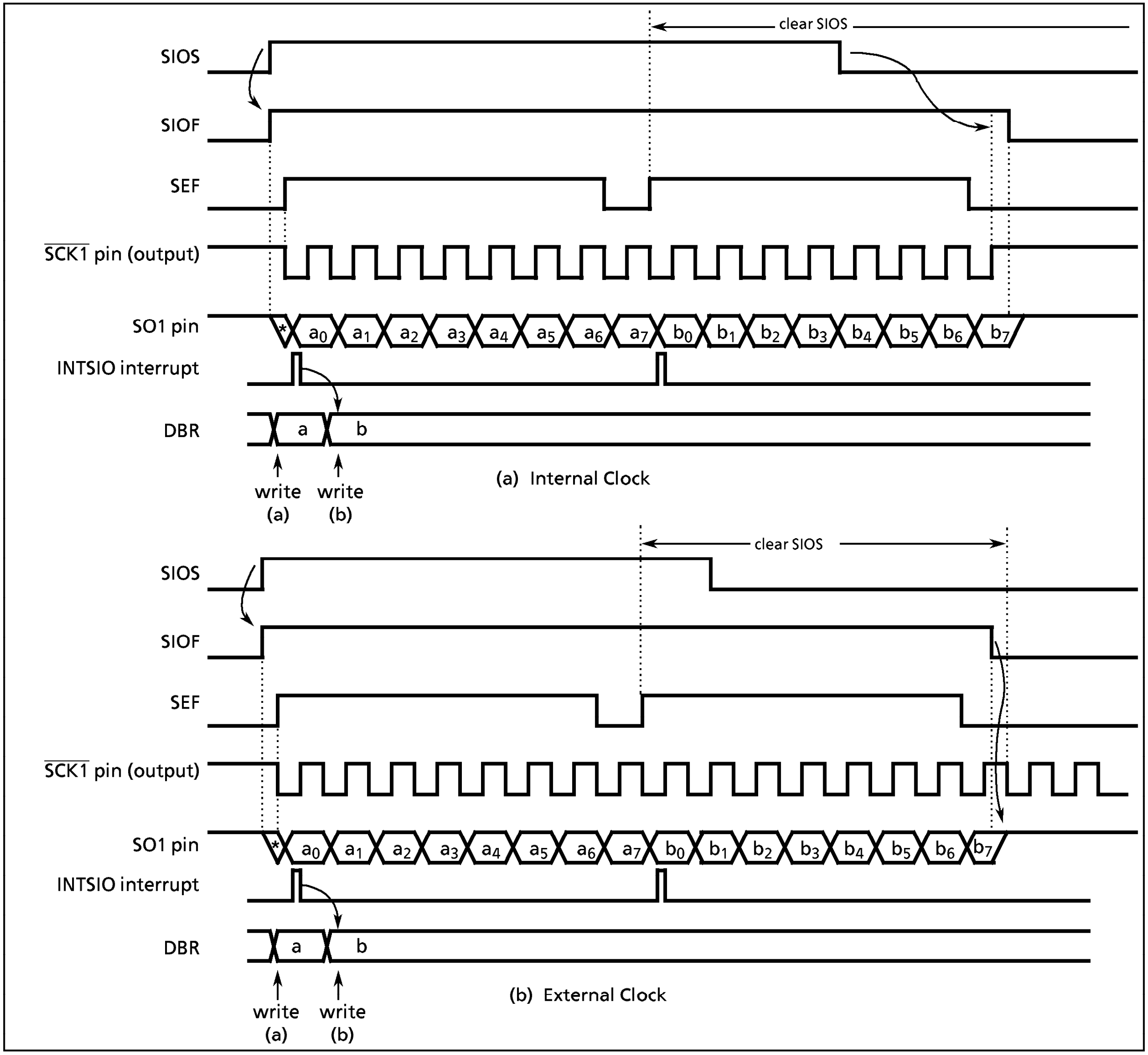


Figure 2-54. Transfer Mode (Example: 8-bit, 1 Word Transfer)

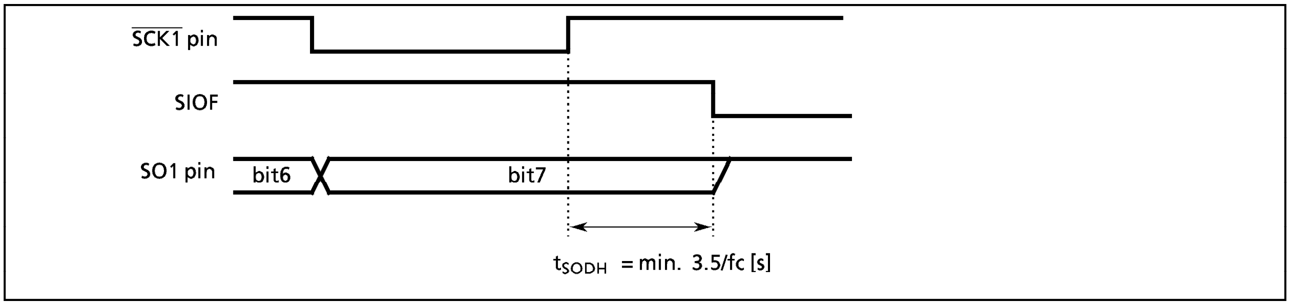


Figure 2-55. Transmitted Data Hold Time at end of transmit

(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 SIO1 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.

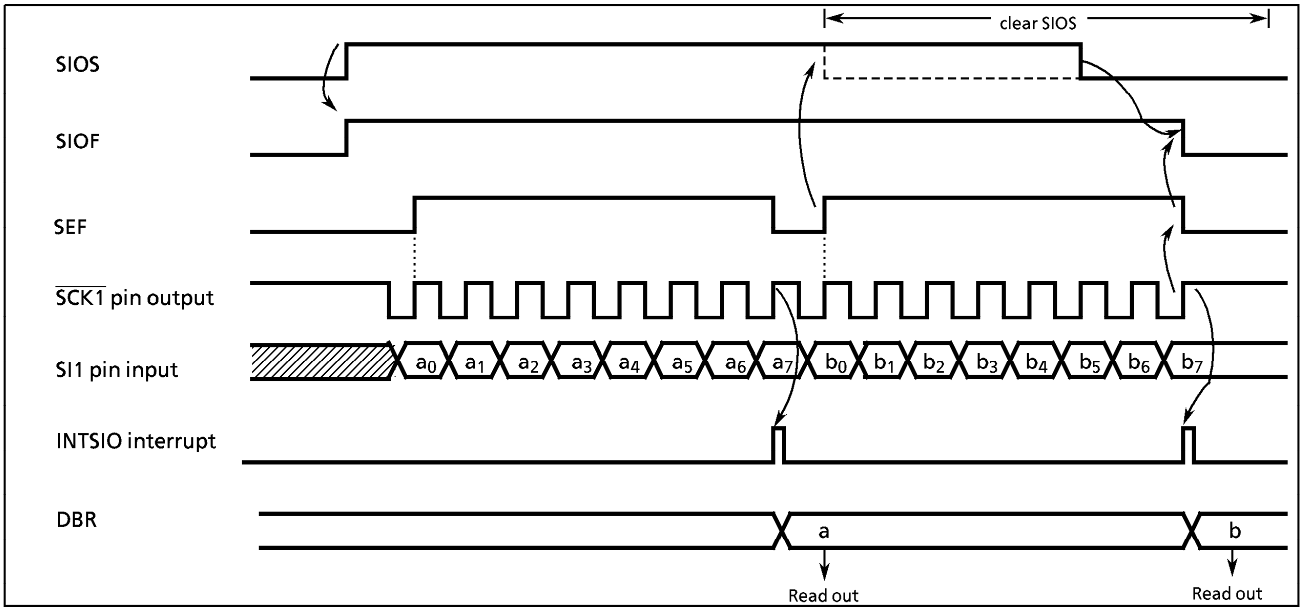


Figure 2-56. Receive Mode (Example : 8-bit, 1 word, internal clock)

(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 SO1 pin at leading edges of the serial clock. When receiving, the data are input to the SI1 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 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.

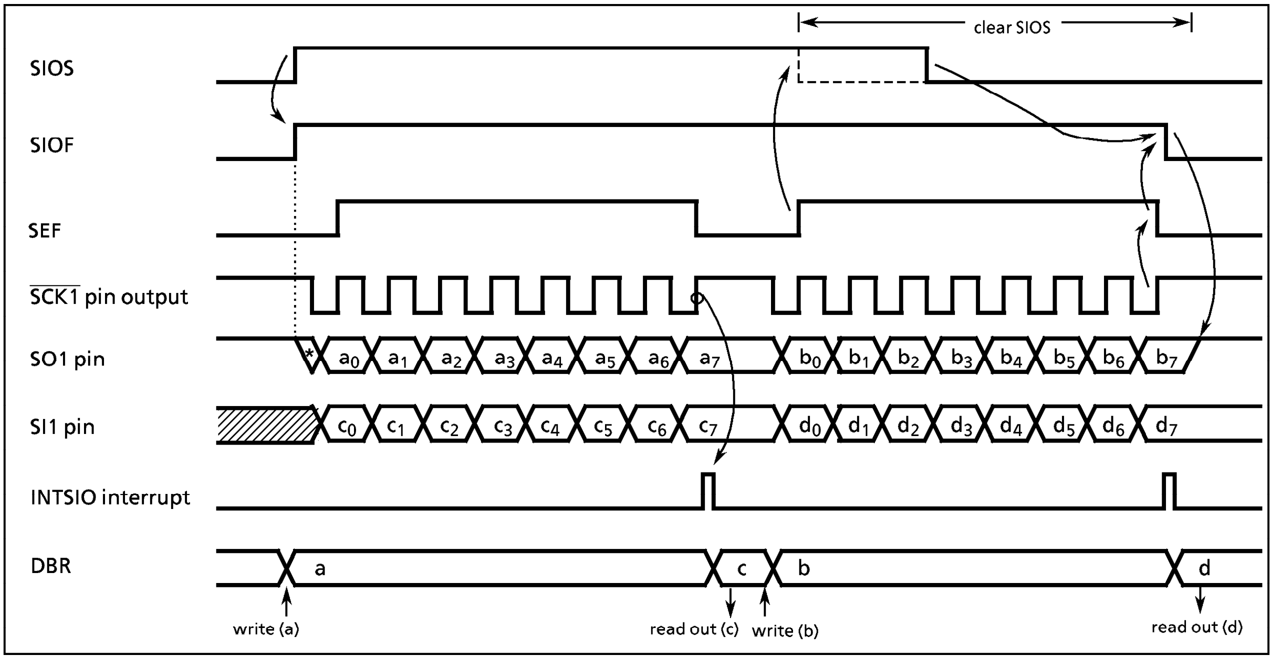


Figure 2-57. Transmit/Receive Mode (Example : 8-bit, 1word, internal clock)

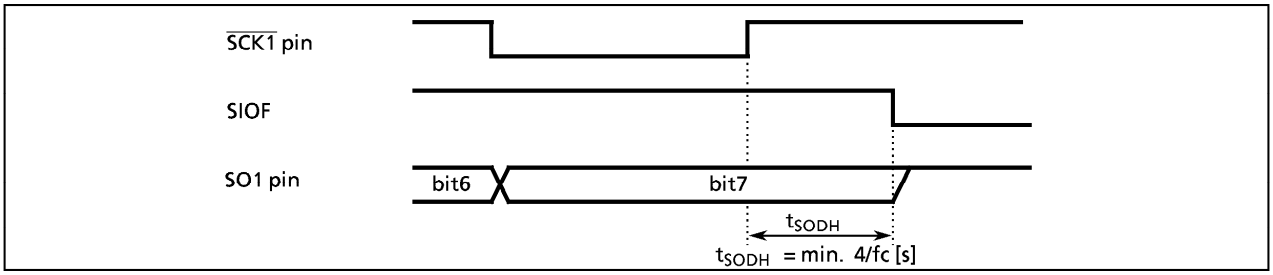


Figure 2-58. Transmitted Data Hold Time at end of transmit/receive

2.9 8-bit A/D Converter (ADC)

The 87C444/844 each have a 4-channel multiplexed-input 8-bit successive approximate type A/D converter with sample and hold.

VDD pin is also used as an analog reference voltage pin (VAREF), and VSS pin is also used as an analog reference GND pin (VASS).

2.9.1 Configuration

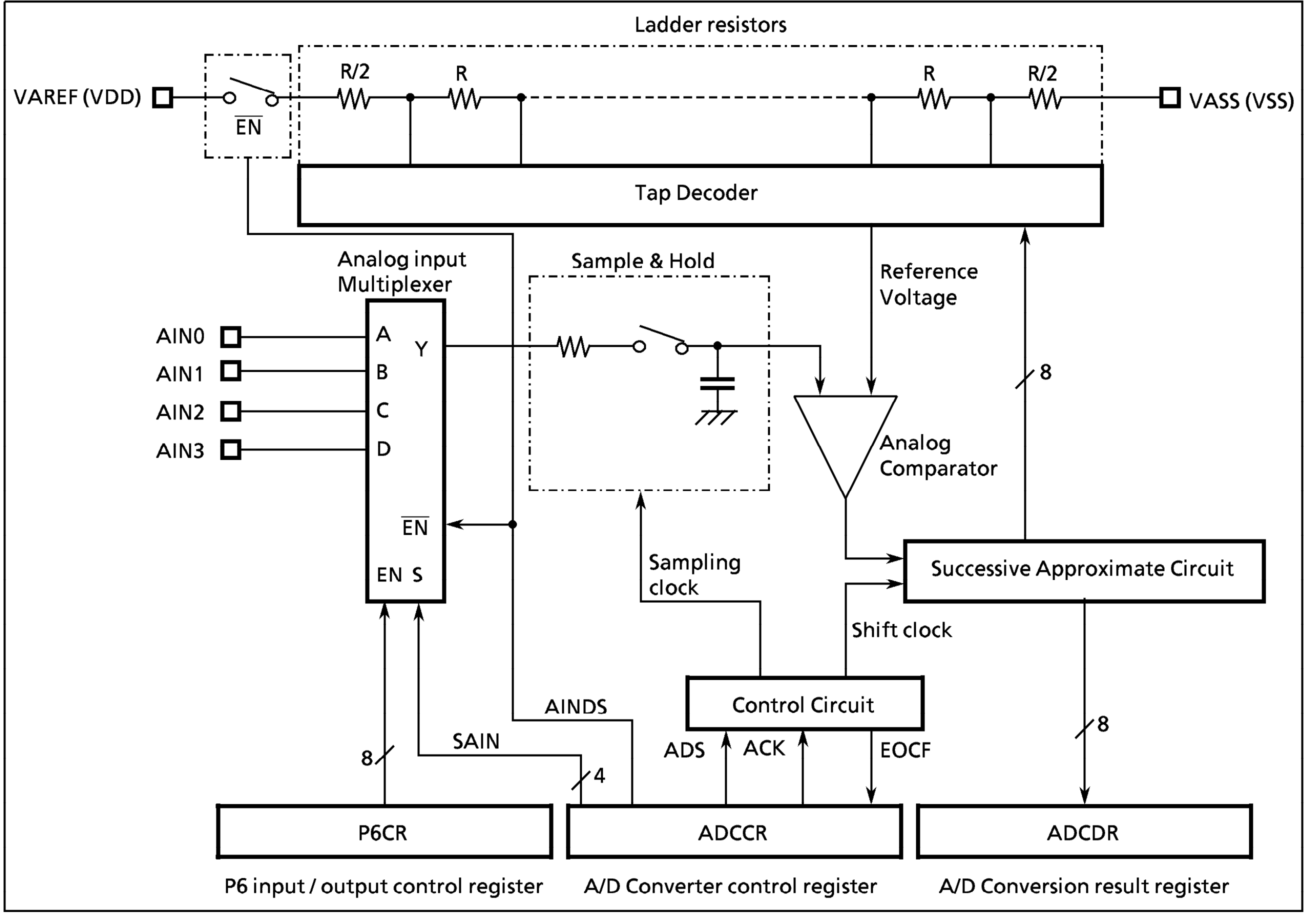


Figure 2-59. A/D Converter

2.9.2 Control

The A/D converter is controlled by an A/D converter control register (ADCCR) and a port P6 input/output control register (P6CR).

A/D Conversion Result Register

	7	6	5	4	3	2	1	0	
ADCDR (000F$_H$)									read only

Figure 2-60. A/D Conversion result register

A/D Converter Control Register

	7	6	5	4	3	2	1	0	
ADCCR (000E$_H$)	EOCF	ADS	ACK	AINDS	SAIN				(Initial value : 0000 0000)

SAIN	Analog input selection	0000 : AIN0 0001 : AIN1 0010 : AIN2 0011 : AIN3 01** : reserved 1 *** : reserved	R/W
AINDS	Analog input control	0 : Enable 1 : Disable	
ACK	conversion time	0 : 23μs (at fc = 8MHz) 1 : 92μs	
ADS	A/D conversion start	0 : – 1 : A/D conversion start	
EOCF	End of A/D conversion flag	0 : Under conversion or Before conversion 1 : End of conversion	read only

*Note 1 : * ; don't care*

Note 2 : Select analog input when A/D converter stops.

Note 3 : The ADS is automatically cleared to "0" after starting conversion.

Note 4 : The EOCF is cleared to "0" when reading the ADCDR.

Note 5 : The EOCF is read-only.

Figure 2-61. A/D converter control register

2.9.3 Operation

Apply analog reference voltage to pins VAREF and VASS.

(1) Start of A/D conversion

First, set the corressponding P6CR bit to "0" for analog input. Clear the AINDS (bit 4 in ADCCR) to "0" and select one of four analog input AIN3-AIN0 with the SAIN (bits 3-0 in ADCCR).
A/D conversion is started by setting the ADS (bit 6 in ADCCR) to "1".
When ACK = 0, conversion is accomplished in 46 machine cycles (184/fc [s]).
The EOCF (bit 7 in ADCCR) is set to "1" at end of conversion.

Note : The pin that is not used as an analog input can be used as regular input /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 conversion, read the conversion result from the ADCDR.
The EOCF is automatically cleared to "0" when reading the ADCDR.

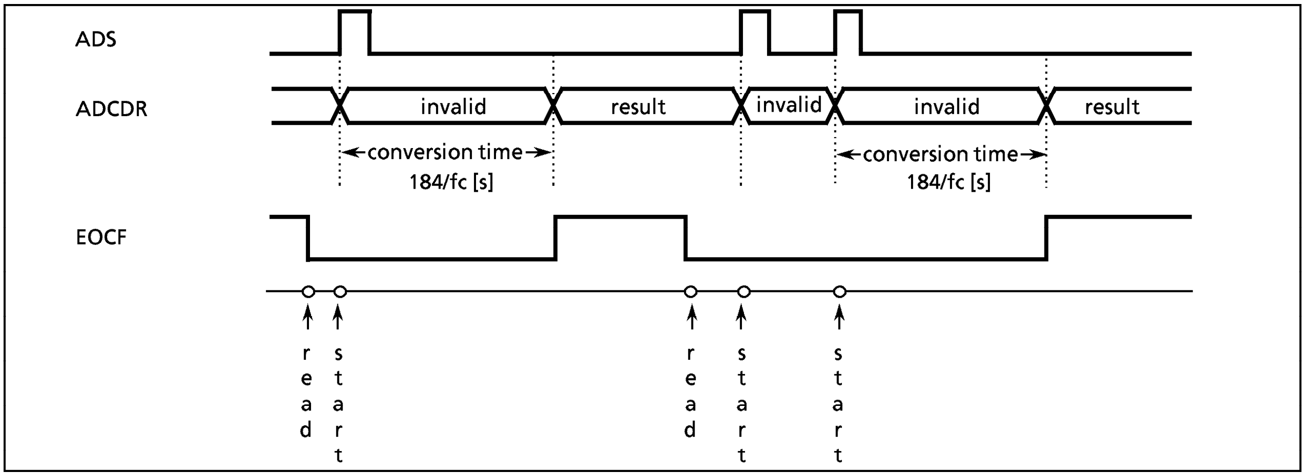


Figure 2-62. A/D conversion Timing chart

Example:

```
                ; AIN SELECT
                LD          (ADCCR) , 00000100B      ;    selects AIN4
                ; A/D CONVERT START
                SET         (ADCCR) . 6              ;    ADS = 1
SLOOP      :    TEST        (ADCCR) . 7              ;    EOCF = 1 ?
                JRS         T, SLOOP
                ; RESULT DATA READ
                LD          (9EH), (ADCDR)
```

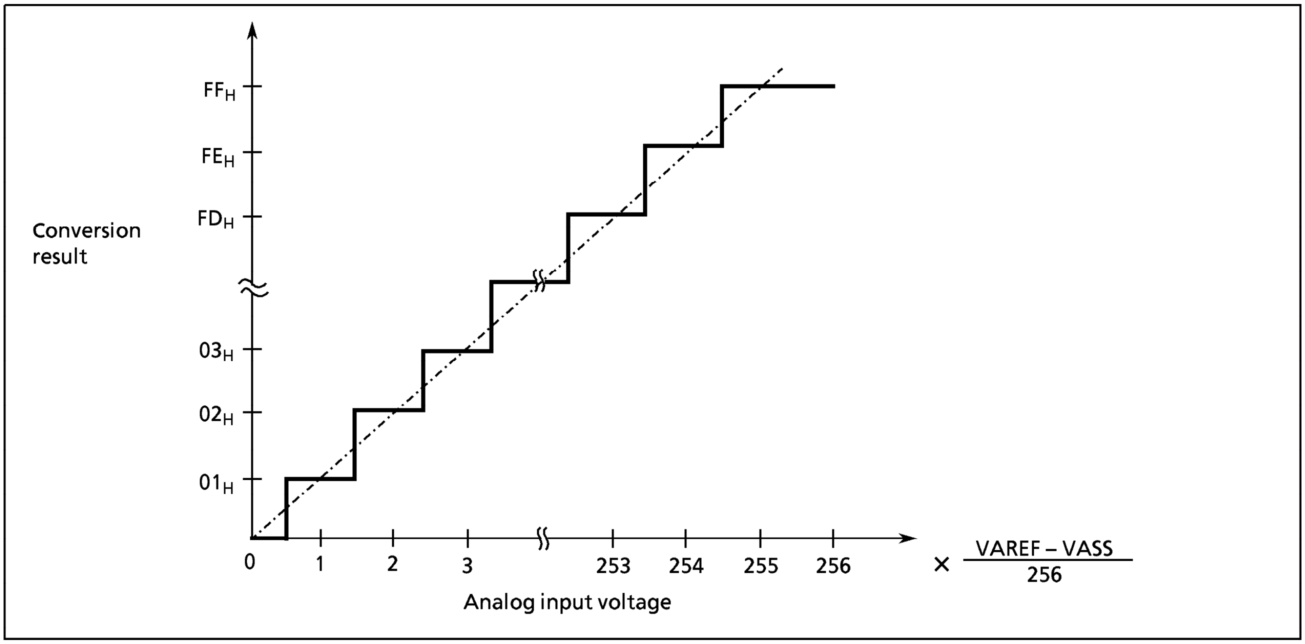


Figure2-63. Analog input voltage vs A/D conversion result (typ.)

2.10 8-bit D/A converters (DAC)

The TMP87C844/C444 has eight channels of 8 bit Digital to Analog (D/A) converters.
Each channel has an op-amp to output.
A terminal can be used for an ordinary I/O port if not used for the D/A converter.
D/A converter output has buit-in pull-down resistor.

2.10.1 Configuration

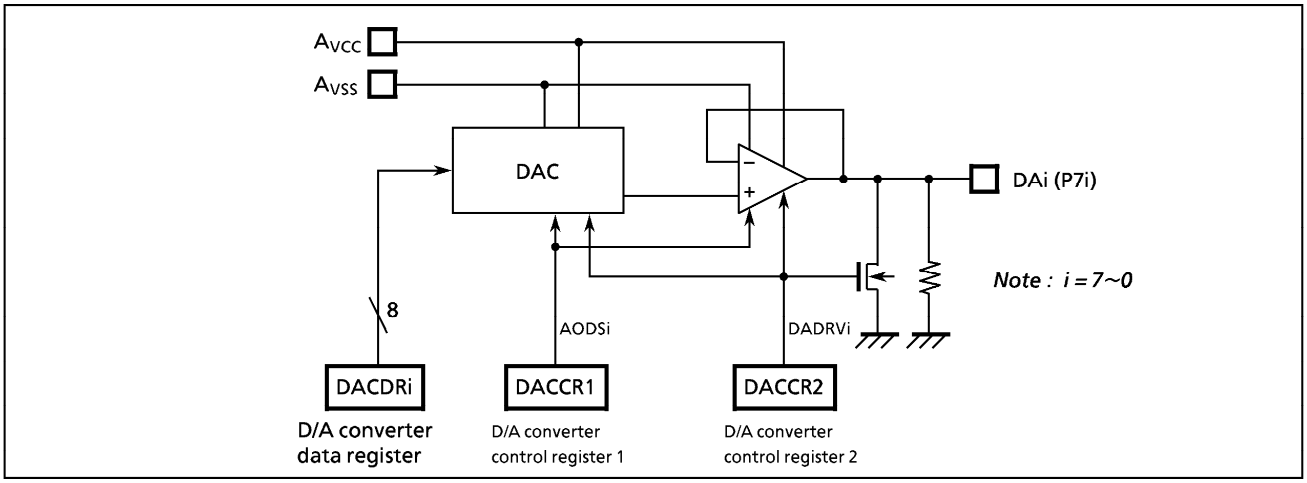

AVCC
AVSS
DAC
−
+
DAi (P7i)
Note : i = 7~0
8
AODSi
DADRVi
DACDRi
DACCR1
DACCR2
D/A converter data register
D/A converter control register 1
D/A converter control register 2

Figure 2-64. 8-bit D/A converters (DAC)

2.10.2 Control

The 8-bit D/A converters are controlled by the D/A converter control registers 1 and 2 (DACCR1, 2). For executing analog output from each channel of the D/A converters, write 8-bit data to the D/A converter data registers DACDR0 to 7, and set DACCR2 to 1.

D/A converter control register 1

	7	6	5	4	3	2	1	0	
DACCR1 (0025_H)	AODS7	AODS6	AODS5	AODS4	AODS3	AODS2	AODS1	AODS0	(Intial value : 0000 0000)

AODS	Controls analog data output (can be set on a bit base)	0 : analog data output enable 1 : input/output	R/W

Figure 2-65. D/A converter control register 1

D/A converter control register 2

	7	6	5	4	3	2	1	0	
DACCR2 (0026_H)	DADRV7	DADRV6	DADRV5	DADRV4	DADRV3	DADRV2	DADRV1	DADRV0	(Intial value : 0000 0000)

DADRV	Register conversion value output (can be set on a bit base)	0 : 0V output 1 : DACDR conversion value output	R/W

D/A converter data register (DACDR0 to DACDR7: for DA0 to DA7) R/W

(Intial value : 0000 0000)

	7	6	5	4	3	2	1	0	
DACDR0	DAO7	DAO6	DAO	DAO4	DAO3	DAO2	DAO1	DAO0	($0FE0_H$)
DACDR1									($0FE1_H$)
DACDR2									($0FE2_H$)
DACDR3									($0FE3_H$)
DACDR4									($0FE4_H$)
DACDR5									($0FE5_H$)
DACDR6									($0FE6_H$)
DACDR7									($0FE7_H$)

Figure 2-66. D/A converter control register 2 and D/A converter data register

2.10.3 D/A converter operation

Apply a high level of analog voltage reference to pin AVCC, and a GND (0V) level to pin AVSS.

(1) Starting D/A conversion operation

D/A conversion operation can be set for each channel of the D/A converters. D/A conversion operation is activated by setting DACCR1 to 0 first. When DACCR2 is set to 0, 0V output is selected. When DACCR2 is set to 1, analog output is selected for data which is written in DACDR of a specified channel.

A channel of which DACCR1 is set to 1 (regardless of DACCR2's setting) can be used as an I/O port.

After resetting, every register of the D/A converters is initialized to 00H and set 0V output. During resetting, each pin becomes Hi-z.

(2) Logical expression for converted value

The relation between a digital value and analog output voltage in DACDR can be expressed in the following equation:

$$V_{AO} = \frac{2^0 \times DAO0 + 2^1 \times DAO1 + 2^2 \times DAO2 + 2^3 \times DAO3 + 2^4 \times DAO4 + 2^5 \times DAO5 + 2^6 \times DAO6 + 2^7 \times DAO7}{256} \times AVCC$$

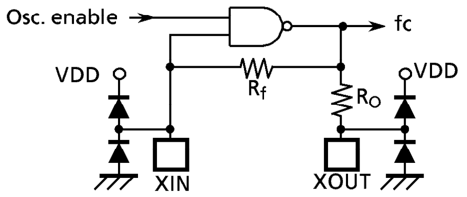
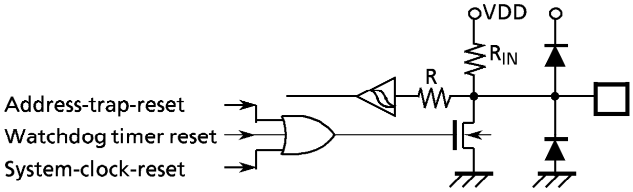
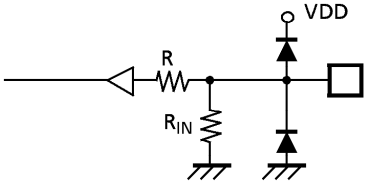
Note 1: An analog voltage reference between AVCC and AVSS is output in 8-bit resolution. However, because an analog output from DA0 to DA7 is limited depending on the characteristics of an op-amp which is connected to DAC, signal is not output from AVCC-0.5 (TYP) to AVCC and from AVSS to AVSS + 0.5 (TYP).

Note 2 : Starting up time takes 0.2 s in the worst case. (@AVCC = 5.0 V, Topr = 25 ℃)

INPUT/OUTPUT CIRCUITRY

(1) Control pins

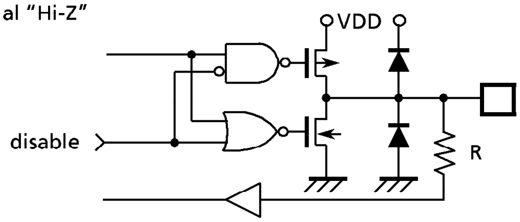
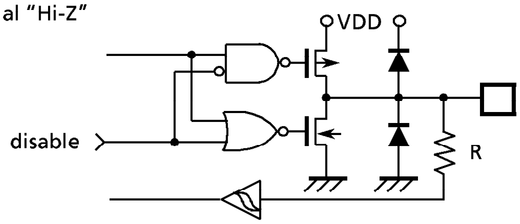
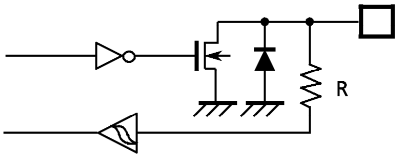
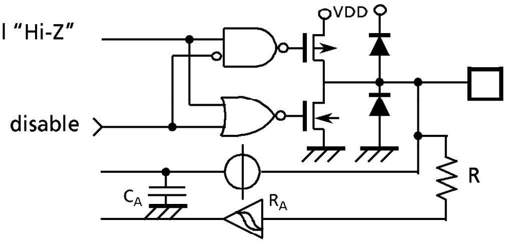
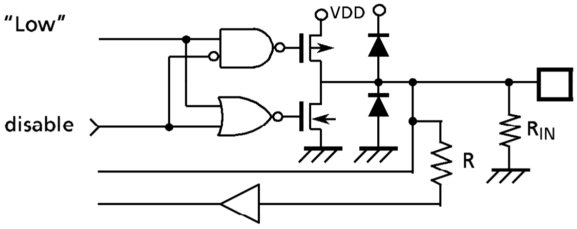
The input/output circuitries of the 87C444/844 control pins are shown below.

CONTROL PIN	I/O	INPUT/OUTPUT CIRCUITRY and CODE	REMARKS
XIN XOUT	Input Output	 Osc. enable fc VDD R_f R_O VDD XIN XOUT	Resonator connecting pins R_f = 1.2 MΩ (typ.) R_O = 1.5 kΩ (typ.) R = 1 kΩ (typ.)
$\overline{\text{RESET}}$	I/O	 VDD R_{IN} R Address-trap-reset Watchdog timer reset System-clock-reset	Sink open drain output Hysteresis input Pull-up resistor R_{IN} = 220 kΩ (typ.) R = 1 kΩ (typ.)
TEST	Input		Pull-down resistor R_{IN} = 220 kΩ (typ.) R = 1 kΩ (typ.)

Note 1 : The 87P844 does not have a pull-down resistor for TEST pin.

(2) Input/Output Ports

The input/output circuitries of the 87C444/844 input/output ports are shown below.

PORT	I/O	INPUT / OUTPUT CIRCUITRY and CODE	REMARKS
P0	I/O	initial "Hi-Z"  	Tri-state I/O R = 1 kΩ (typ.)
P1 P60 to P62	I/O	initial "Hi-Z"  	Tri-state I/O Hysteresis input R = 1 kΩ (typ.)
P3	I/O	initial "Hi-Z"  	Sink open drain output Hysteresis input R = 1 kΩ (typ.)
P63 to P66	I/O	initial "Hi-Z"  	Tri-state I/O Hysteresis input Analog input R = 1 kΩ (typ.) R_A = 5 kΩ (typ.) C_A = 12 pF (typ.)
P7	I/O	initial "Low"  	Tri-state I/O Analog output Pull-down resistor R_{IN} = 6 kΩ (typ.) R = 1 kΩ (typ.)

ELECTRICAL CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS

(V_{SS} = 0V)

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 $\overline{\text{RESET}}$	– 0.3 to V_{DD} + 0.3	V
	V_{OUT2}	Sink open drain pin except $\overline{\text{RESET}}$	– 0.3 to 5.5	V
Output Current (Per 1 pin)	I_{OUT1}	Ports P0, P1, P3, P6, P7	3.2	mA
Output Current (Total)	ΣI_{OUT1}	Ports P0, P1, P3, P6, P7	120	mA
Power Dissipation [Topr = 70°C]	PD		600	mW
Soldering Temperature (time)	Tsld		260 (10 s)	°C
Storage Temperature	Tstg		– 55 to 125	°C
Operating Temperature	Topr		– 30 to 70	°C

RECOMMENDED OPERATING CONDITIONS

(V_{SS} = 0V, Topr = – 30 to 70°C)

PARAMETER	SYMBOL	PINS	CONDITIONS		Min.	Max.	UNIT
Supply Voltage	V_{DD}		fc = 8MHz	NORMAL mode	4.5	5.5	V
				IDLE mode			
Input High Voltage	V_{IH1}	Except hysteresis input	$V_{DD} \geqq 4.5V$		$V_{DD} \times 0.70$	V_{DD}	V
	V_{IH2}	Hysteresis input			$V_{DD} \times 0.75$		
Input Low Voltage	V_{IL1}	Except hysteresis input	$V_{DD} \geqq 4.5V$		0	$V_{DD} \times 0.30$	V
	V_{IL2}	Hysteresis input				$V_{DD} \times 0.25$	
Clock Frequency	fc	XIN, XOUT	V_{DD} = 4.5~5.5V		1	8.0	MHz

D.C. CHARACTERISTICS

(V_{SS} = 0V, T_{opr} = − 30 to 70 °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} = 5.5V, V_{IN} = 5.5V / 0V	–	–	± 2	μA
	I_{IN2}	Open drain ports and Tri-state ports					
	I_{IN3}	$\overline{\text{RESET}}$					
Input Resistance	R_{IN2}	$\overline{\text{RESET}}$		100	220	450	kΩ
	R_{IN3}	Port P7		4	6	10	
Output Leakage Current	I_{LO1}	Open drain ports	V_{DD} = 5.5V, V_{OUT} = 5.5V	–	–	2	μA
	I_{LO2}	Tri-state ports	V_{DD} = 5.5V, V_{OUT} = 5.5V/0V	–	–	± 2	
Output High Voltage	V_{OH1}	Tri- state ports	V_{DD} = 4.5V, I_{OH} = − 0.7 mA	4.1	–	–	V
	V_{OH2}	Port P7	V_{DD} = 4.5V, I_{OH} = − 0.2 mA				
Output Low Voltage	V_{OL}	Except XOUT	V_{DD} = 4.5V, I_{OL} = 1.6 mA	–	–	0.4	V
Supply Current in NORMAL mode			V_{DD} = 5.5V V_{IN} = 5.3V/0.2V fc = 8 MHz	–	8	14	mA
Supply Current in IDLE mode				–	4	6	mA

Note 1 : Typical values show those at T_{opr} = 25 °C, V_{DD} = 5V.
Note 2 : Input Current : I_{IN1}, I_{IN3} ; The current through pull-up or pull-down resistor is not included.
Note 3 : I_{DD} does not include I_{AREF} / I_{DREF}.

A/D CONVERSION CHARACTERISTICS

(Topr = − 30 to 70 °C : V_{SS} = V_{ASS} = 0V)

PARAMETER	SYMBOL	CONDITIONS	Min.	Typ.	Max.	UNIT
Analog Reference Voltage	V_{AREF}	V_{DD} = V_{AREF}	4.5	–	5.5	V
Analog Input Voltage	V_{AIN}		V_{ASS}	–	V_{AREF}	V
Analog Supply Current	I_{AREF}		–	0.5	1.0	mA
Nonlinearity Error		V_{AREF} = V_{DD} = 5.000V V_{ASS} = V_{SS} = 0.000V	–	–	± 2	LSB
Zero point Error			–	–	± 2	
Full Scale Error			–	–	± 2	
Total Error			–	–	± 3	

D/A CONVERSION CHARACTERISTICS (V$_{SS}$ = AV$_{SS}$ = 0, V$_{DD}$ = 4.5 to 5.5V, Topr = – 30 to 70 °C)

PARAMETER		SYMBOL	CONDITIONS	Min.	Typ.	Max.	UNIT
Analog Reference Voltage		A$_{VCC}$		4.5	–	V$_{DD}$	V
Current Dissipation		I$_{DREF}$	No Loading, All channel operating			25	mA
Resolution						8	bits
Accuracy	Nonlinearity Error		A$_{VCC}$ = 5.000V : A$_{VSS}$ = 0.000V	–		± 2.0	LSB
	Differential Nonlinearity Error		Monotonicity Guarantee (Note1)	–		± 3/4	LSB
Settling time		T$_{SU}$	Loading condition : c = 15 pF	–		5	μS
OP-Amp output Voltage Range		V$_{AO}$	No Loading	0.03		A$_{VCC}$ – 0.25	V
			I$_{AO}$ = 1.2 mA / I$_{AO}$ = – 200 μA	0.3		A$_{VCC}$ – 0.3	
OP-Amp output Drive Range		I$_{AO}$	A$_{VCC}$ – 0.5 to 0.5V	–	+ 2/ – 1		mA
Maximum Capacitors connected to D/A output		C$_{OL}$				15	pF

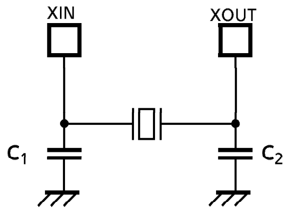
Note 1 : Differential nonlinearity error does not include quantizing error.

A.C. CHARACTERISTICS (V$_{SS}$ = 0V, V$_{DD}$ = 4.5 to 5.5V, T$_{opr}$ = – 30 to 70 °C)

PARAMETER	SYMBOL	CONDITIONS	Min.	Typ.	Max.	UNIT
Machine Cycle Time	tcy	In NORMAL mode In NORMAL mode	0.5	–	4	μs
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$_{WCL}$					

RECOMMENDED OSCILLATING CONDITION (V$_{SS}$ = 0V, V$_{DD}$ = 4.5 to 5.5V, T$_{opr}$ = – 30 to 70 °C)

PARAMETER	OSCILLATOR	FREQUENCY	RECOMMENDED OSCILLATOR		RECOMMENDED CONDITIONS	
					C$_1$	C$_2$
High-frequency	Ceramic Resonator	8 MHz	KYOCERA	KBR8.0M	30 pF	30 pF
	Crystal Oscillator	8 MHz	TOYOKOM	210B 8.0000	20 pF	20 pF



High-frequency

Note : To keep reliable operation, shield the device electrically with the metal plate on its package mold surface against the high electric field, for example, by CRT (Cathode Ray Tube).