CMOS 8-BIT MICROCONTROLLER

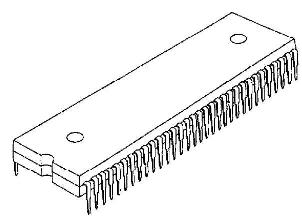
TMP87CM39N, TMP87CP39N, TMP87CS39N

The 87CM39/P39/S39 are the high speed and high performance 8-bit single chip microcomputers. These MCU contain CPU core, ROM, RAM, input / output ports, six multi-function timer / counters, serial bus interface, on-screen display, PWM outputs, 8-bit A/D converter, remote control signal preprocessor, and two clock generators on a chip.

PART No.	ROM	RAM	PACKAGE	OTP
TMP87CM39N	32K bytes	1K bytes	SDIP64-P-750-1.78	TMP87PS39N
TMP87CP39N	48K bytes	2K bytes		
TMP87CS39N	60K bytes			

FEATURES

SDIP64-P-750-1.78



TMP87CM39N
TMP87CP39N
TMP87CS39N
TMP87PS39N

- ◆8-bit single chip microcomputer TLCS-870 Series
- ◆Instruction execution time : 0.5 μs (at 8 MHz), 122 μs (at 32.768 kHz)
- ◆412 basic instructions
  - Multiplication and Division (8 bits × 8 bits , 16 bits ÷ 8 bits)
  - Bit manipulations (Set / Clear / Complement / Move / Test / Exclusive or)
  - 16-bit data operations
  - 1-byte jump / subroutine-call (Short relative jump / Vector call)
- ◆15 interrupt sources (External : 6, Internal : 9)
  - All sources have independent latches each, and nested interrupt control is available.
  - 4 edge-selectable external interrupts with noise reject
  - High-speed task switching by register bank changeover
- ◆Program Corrective Function
- ◆8 Input / Output ports (55 pins)
  - High current output : 4 pins (typ. 20 mA)
- ◆Two 16-bit Timer / Counters
  - Timer, Event counter, Programmable pulse generator output, Pulse width measurement, External trigger timer, Window modes
- ◆Two 8-bit Timer / Counters
  - Timer, Event counter, Capture (Pulse width / duty measurement) modes
- ◆Time Base Timer (Interrupt frequency : 1 Hz to 16 kHz)
- ◆Divider output function (frequency : 1 kHz to 8 kHz)
- ◆Watchdog Timer
  - Interrupt source / reset output (programmable)
- ◆Serial Bus Interface
  - I2C-bus, 8-bit SIO modes
    Selectable two I/O channels
- ◆On-screen display circuit
  - Character patterns : 256 characters
  - Characters displayed : 24 columns × 12 lines
  - Composition : 14 × 18 dots
  - Size of character : 3 kinds (line by line)
  - Color of character : 8 kinds (character by character)
  - Variable display position : Horizontal 128 steps, Vertical 256 steps
  - Fringing, Smoothing function
- ◆D/A conversion (Pulse Width Modulation) outputs
  - 14-bit resolution (1 channel)
  - 7-bit resolution (9 channels)
- ◆8-bit successive approximate type A/D converter with sample and hold
  - 8 analog inputs
  - Conversion time : 23 μs at 8 MHz

◆Remote control signal preprocessor
◆Jitter Elimination
◆Dual clock operation
- Single / Dual-clock mode (option)

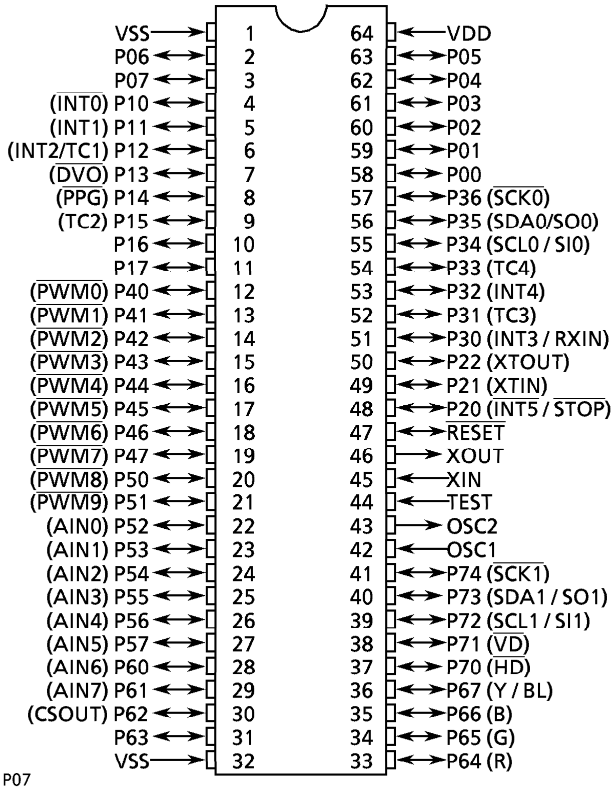
◆Five Power saving operating modes
- STOP mode : Oscillation stops. Battery / Capacitor back-up. Port output hold / high-impedance.
- SLOW mode: Low power consumption operation using low-frequency clock (32.768 kHz).
- IDLE1 mode : CPU stops, and Peripherals operate using high-frequency clock. Release by interrupts.
- IDLE2 mode : CPU stops, and Peripherals operate using high and low frequency clock. Release by interrupts.
- SLEEP mode : CPU stops, and Peripherals operate using low-frequency clock. Release by interrupts.

◆Wide operating voltage : 2.7 to 5.5 V at 32.768 kHz, 4.5 to 5.5 V at 8 MHz
◆Emulation Pod : BM87CS39N0A

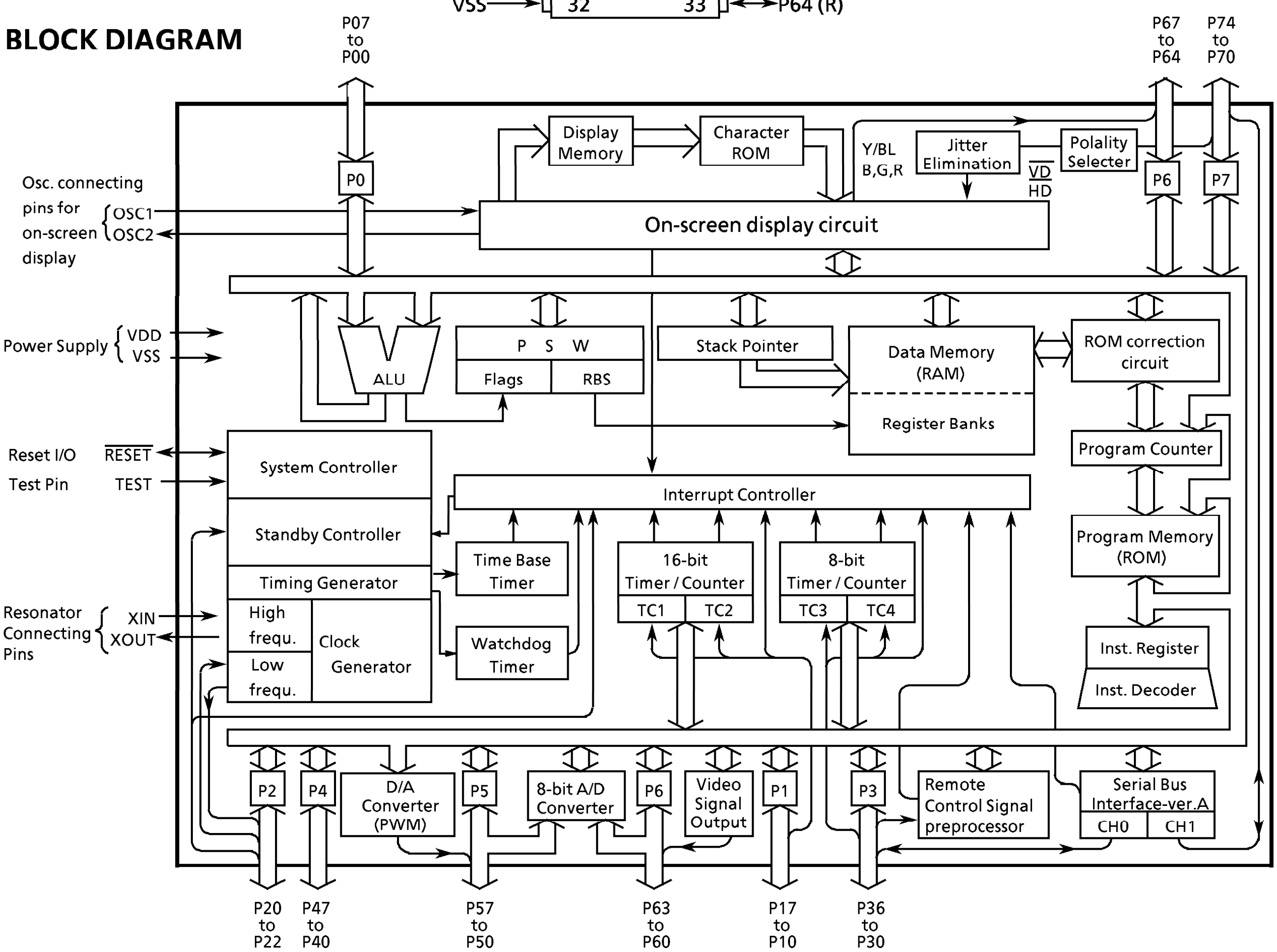
Purchase of TOSHIBA I^2C components conveys a license under the Philips I^2C Patent Rights to use these components in an I^2C system, provided that the system conforms to the I^2C Standard Specification as defined by Philips.

PIN ASSIGNMENTS (TOP VIEW)

SDIP64-P-750-1.78



BLOCK DIAGRAM



PIN FUNCTION

PIN NAME	Input/Output	Function	
P07 to 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
P22 (XTOUT)	I/O (Output)	3-bit input/output port with latch. When used as an input port, the latch must be set to "1".	Resonator connecting pins (32.768 kHz). For inputting external clock, XTIN is used and XTOUT is opened.
P21 (XTIN)	I/O (Input)		
P20 ($\overline{INT5}$ / $\overline{STOP}$)			External interrupt input 5 or STOP mode release signal input
P36 ($\overline{SCK0}$)	I/O (I/O)	7-bit input / output port with latch. When used as an input port, a serial bus interface input / output, a timer / counter input, a remote control signal preprocessor input, or an external interrupt input, the latch must be set to "1".	SIO serial clock input / output 0
P35 (SDA0 / SO0)	I/O (I/O / Output)		I^{2}Cbus serial data input / output or SIO serial data output 0
P34 (SCL0 / SI0)	I/O (I/O / Input)		I^{2}Cbus serial clock input / output or SIO serial data input 0
P33 (TC4)	I/O (Input)		Timer / Counter 4 input
P32 (INT4)			External interrupt input 4
P31 (TC3)			Timer / Counter 3 input
P30 (INT3 / RXIN)	I/O (Input / Input)		External interrupt input 3 or remote control signal preprocessor input
P47 ($\overline{PWM7}$) to P41 ($\overline{PWM1}$)	I/O (Output)	8-bit programable input / output port (tri-state). Each bit of this port 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 PWM output, the latch must be set to "1".	7-bit D/A conversion (PWM) outputs
P40 ($\overline{PWM0}$)			14-bit D/A conversion (PWM) output
P57 (AIN5) to P52 (AIN0)	I/O (Input)	8-bit programable 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 an input port, analog input, or a PWM output, the latch must be set to "1".	A/D converter analog inputs
P51 ($\overline{PWM9}$)	I/O (Output)		7-bit D/A conversion (PWM) outputs
P50 ($\overline{PWM8}$)			

PIN NAME	Input/Output	Function		
P67 (Y/BL)	I/O (Output)	8-bit programable input / output port (P67 to P64 : tri-state, P63 to P60 : High current output). Each bit of this port can be individually configured as an input or an output under software control. During reset, all bits are configured as inputs. When used as the R, G, B, Y/BL outputs of on-screen display circuit, each bit of the P6 port data selection register (bits 7 to 4 in address $0F91_H$) must be set to "1".	Focus signal output or Background blanking control signal output	
P66 (B)			RGB outputs	
P65 (G)				
P64 (R)				
P63	I/O		High current outputs	
P62 (CSOUT)				Test video signal output
P61 (AIN7)	I/O (Input)			A/D converter analog inputs
P60 (AIN6)				
P74 ($\overline{SCK1}$)	I/O (I/O)	5-bit input / output port with latch. When used as an input port, a serial bus interface input / output, or a vertical synchronous signal input and horizontal synchronous signal input, the latch must be set to "1".	SIO serial clock input / output 1	
P73 (SDA1 / SO1)	I/O (I/O / Output)		I^2Cbus serial data input / output or SIO serial data output 1	
P72 (SCL1 / SI1)	I/O (I/O / Input)		I^2Cbus serial data input / output or SIO serial data input 1	
P71 ($\overline{VD}$)	I/O (Input)		Vertical synchronous signal input	
P70 ($\overline{HD}$)			Horizontal synchronous signal input	
OSC1, OSC2	Input, Output	Resonator connecting pins for on-screen display circuitry.		
XIN, XOUT		Resonator connecting pins. For inputting external clock, XIN is used and XOUT is opened.		
$\overline{RESET}$	I/O	Reset signal input or watchdog timer output / address-trap- reset output / system-clock-reset output.		
TEST	Input	Test pin for out-going test. Be tied to low.		
VDD, VSS	Power Supply	+ 5 V, 0 V (GND)		

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 87CM39/P39/S39. In the TLCS-870 Series, the memory is organized 4 address spaces (ROM, RAM, SFR, and DBR). It uses a memory mapped I/O system, and all I/O registers are mapped in the SFR / DBR address spaces. There are 16 banks of general-purpose registers. The register banks are also assigned to the first 128 bytes of the RAM address space.

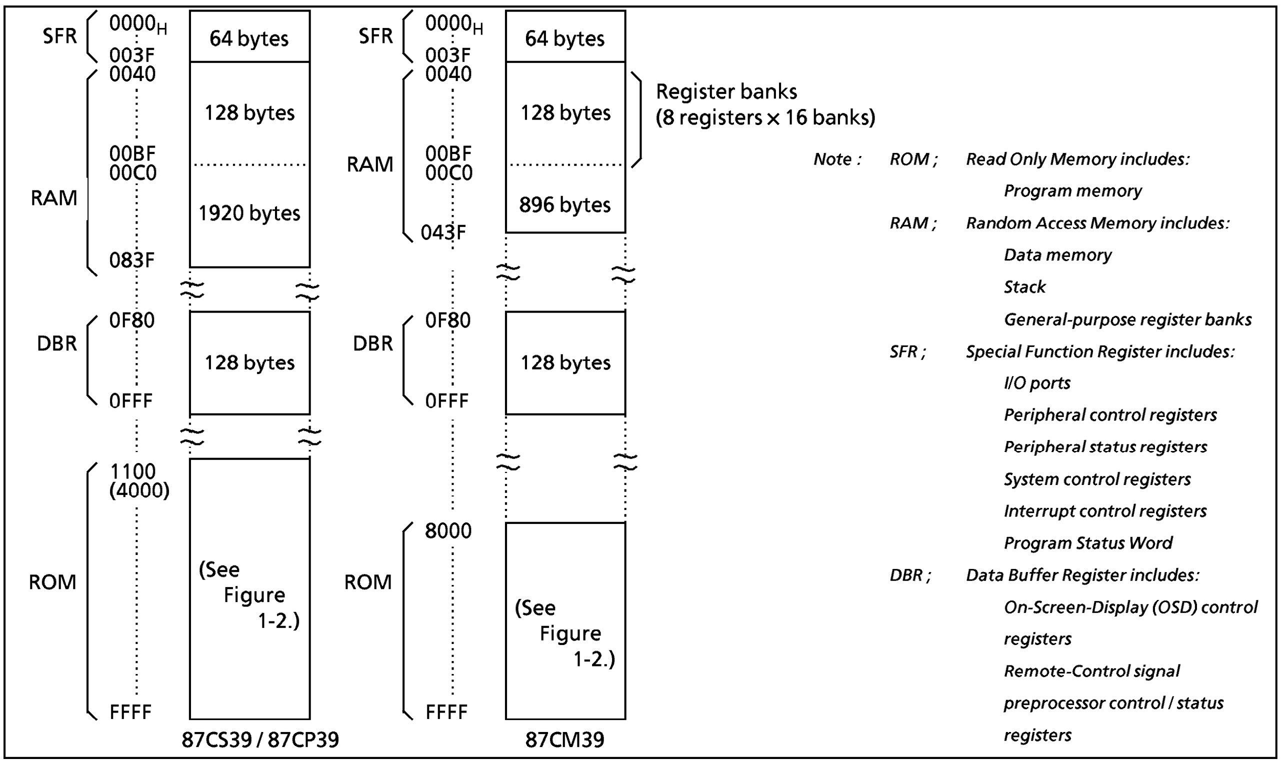


Figure 1-1. Memory Address Map

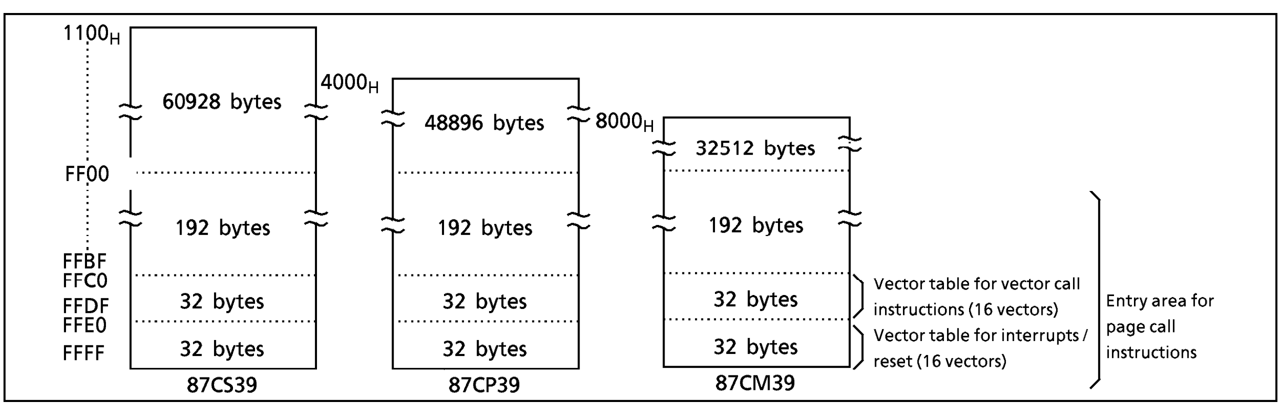


Figure 1-2. ROM Address Maps

1.2 Program Memory (ROM)

The 87CM39 has a 32K bytes (addresses 8000_H to $FFFF_H$), the 87CP39 has a 48K bytes (addresses 4000_H to $FFFF_H$), and the 87CS39 has a 60K bytes (addresses 1100_H to $FFFF_H$) of program memory (mask programmed ROM). Addresses $FF00_H$ to $FFFF_H$ in the program memory can also be used for special purposes. Figure 1-2 shows the ROM address maps of the 87CM39/P39/S39.

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

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

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

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

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

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

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

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

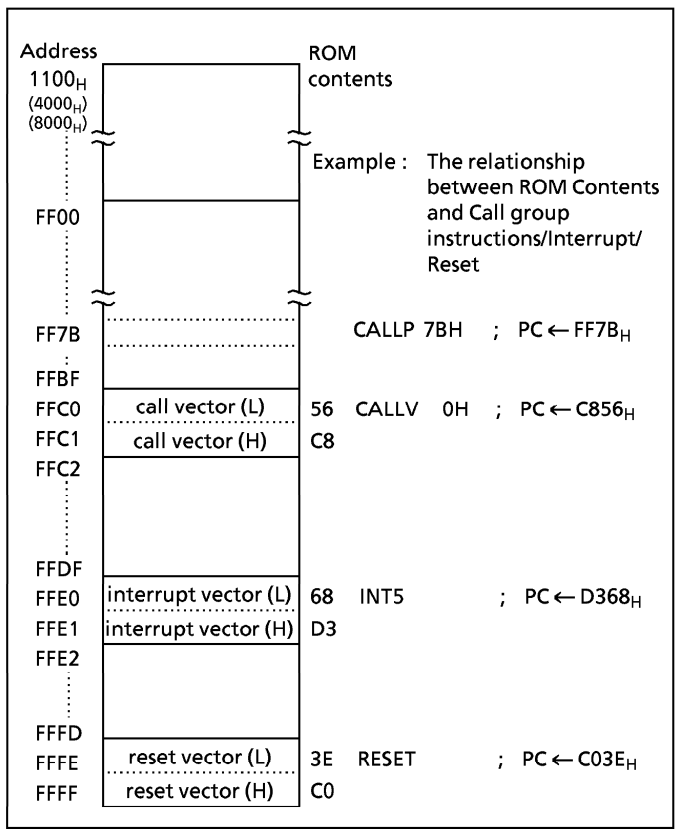


Figure 1-3. Program Memory Map

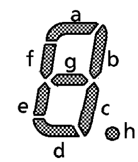
In the TLCS-870 Series, the same instruction used to access the data memory (e.g. [LD A, (HL)]) is also used to read out fixed data (ROM data) stored in the program memory. The register-offset PC-relative addressing (PC + A) instructions can also be used, and the code conversion, table look-up and n-way 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 ≧ 1100_H for 87CS39):

```
        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:

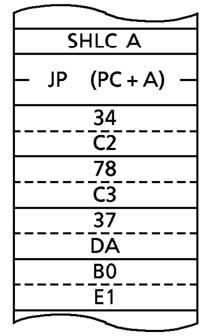
```
        ADD     A, TABLE − $ − 4    ; P5 ←ROM (TABLE + A)
        LD      (P5), (PC + A)
        JRS     T, SNEXT            ; Jump to SNEXT
TABLE : DB      0C0H, 0F9H, 0A4H, 0B0H, 99H, 92H, 82H, 0D8H, 80H, 98H
SNEXT :
```



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 $C0_H$ and $3E_H$ are stored at addresses $FFFF_H$ and $FFFE_H$, respectively, the execution starts from address $C03E_H$ after reset.

The TLCS-870 Series utilizes pipelined processing (instruction pre-fetch); therefore, the PC always indicates 2 addresses in advance. For example, while a 1-byte instruction stored at address $C123_H$ is being executed, the PC contains $C125_H$.

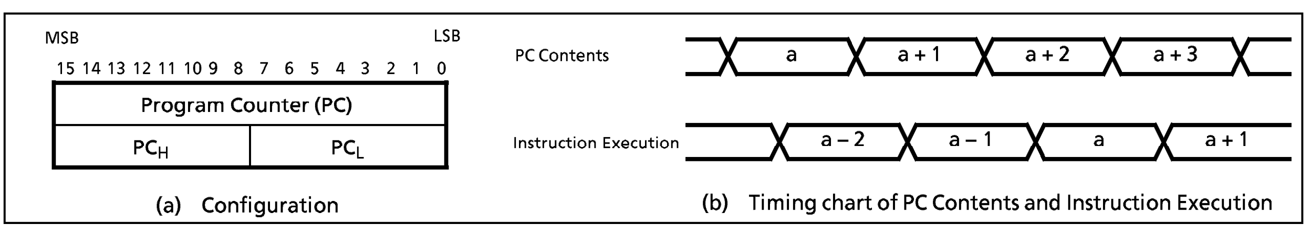


Figure 1-4. Program Counter

1.4 Data Memory (RAM)

The 87CM39 has a 1 K bytes (addresses 0040_H to $043F_H$) of data memory, and the 87CP39/CS39 have a 2 K bytes (address 0040_H to $083F_H$) of data memory. Figure 1-5 shows the data memory map.
Addresses 0000_H to $00FF_H$ are used as a direct addressing area to enhance instructions which utilize this addressing mode; therefore, addresses 0040_H to $00FF_H$ in the data memory can also be used for user flags or user counters. General-purpose register banks (8 registers × 16 banks) are also assigned to the 128 bytes of addresses 0040_H to $00BF_H$. Access as data memory is still possible even when being used for registers. For example, when the contents of the data memory at address 0040_H is read out, the contents of the accumulator in the bank 0 are also read out. The stack can be located anywhere within the data memory except the register bank area. The stack depth is limited only by the free data memory size. For more details on the stack, see section "1.7 Stack and Stack Pointer".

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 : The general-purpuse registers are mapped in the RAM ; therefore, do not clear RAM at the current bank addresses.*

Example1 : Clears RAM to "00_H" except the bank 0 (87CM39):

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

Example2 : Clears RAM to "00_H" except the bank 0 (87CP39/CS39):

```
          LD     HL, 0048H      ;  Sets start address to HL register pair
          LD     A, H           ;  Sets initial data (00H) to A register
          LD     BC, 07F7H      ;  Sets number of byte to BC register pair
SRAMCLR : LD     (HL+), A
          DEC    BC
          JRS    F, SRAMCLR
```

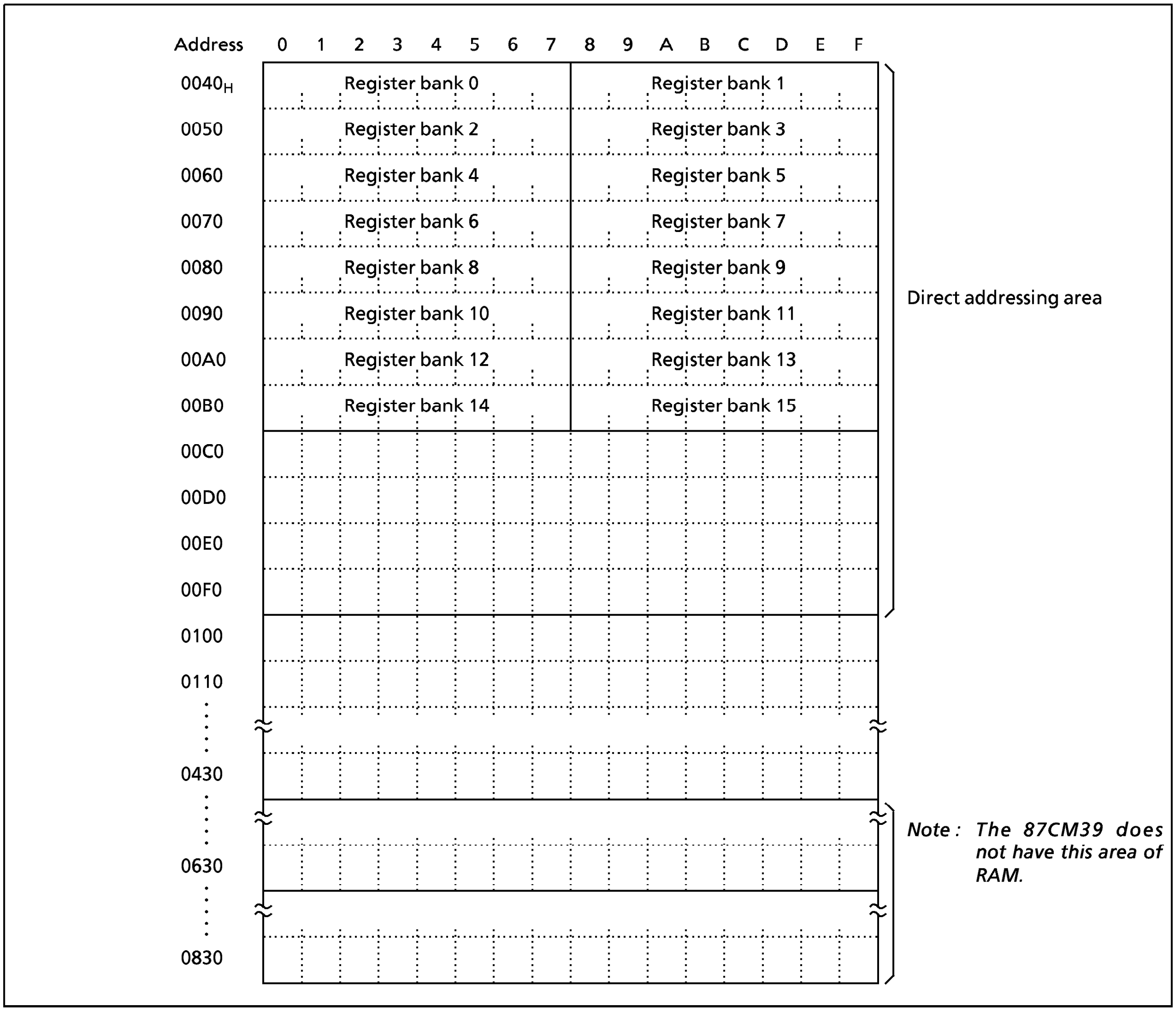


Figure 1-5. Data Memory Map

1.5 General-purpose Register Banks

General-purpose registers are mapped into addresses 0040_H to $00BF_H$ 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.

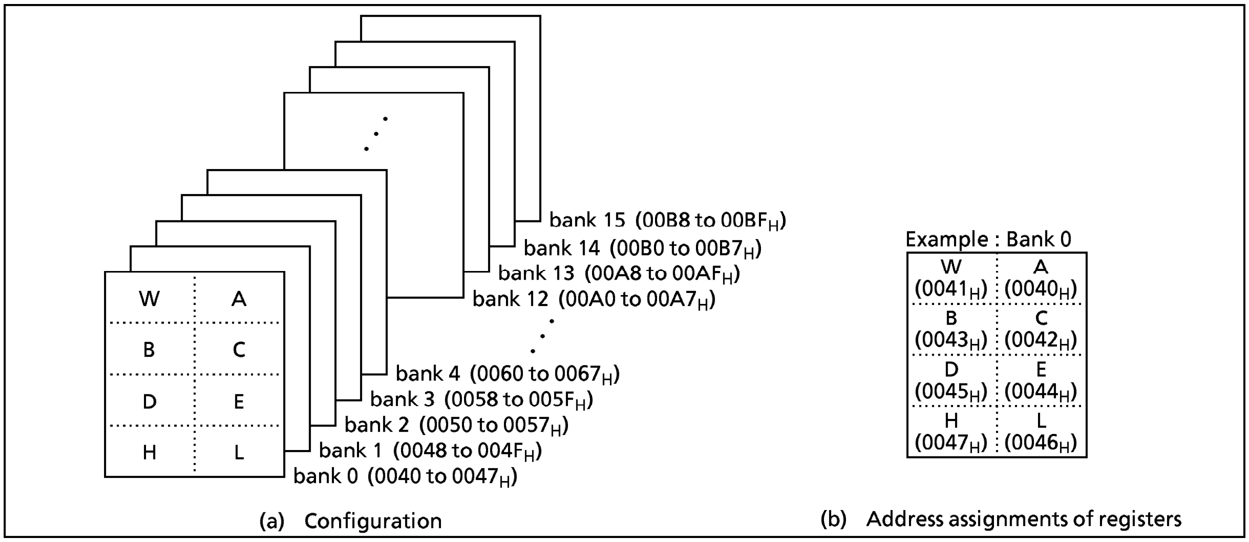


Figure 1-6. General-purpose Register Banks

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

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

```
Example 2 : Block transfer
                LD    B, n-1         ; Sets (number of bytes to transfer) – 1 to B
                LD    HL, DSTA       ; Sets destination address to HL
                LD    DE, SRCA       ; Sets source address to DE
        SLOOP : LD    (HL), (DE)     ; (HL) ← (DE)
                INC   HL             ; HL ← HL + 1
                INC   DE             ; DE ← DE + 1
                DEC   B              ; B ← B – 1   JRS   F, SLOOP ;   if B ≧ 0 then loop
```

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

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

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

Example 2  :  Unsigned integer division (16-bit ÷ 8-bit)
              DIV    WA, C          ; Divides the WA contents by the C contents, places the
                                      quotient in A and the remainder in W.
```

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

7	6	5	4	3	2	1	0
JF	ZF	CF	HF	RBS			

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, CF ← 0
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.

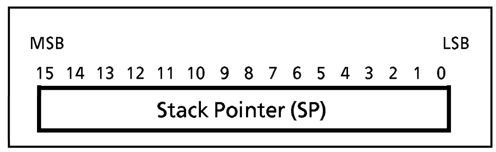


Figure 1-8. Stack Pointer

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, 043FH        ;  SP←043FH
Example 2 :  To read the SP
             LD     HL, SP           ;  HL←SP
```

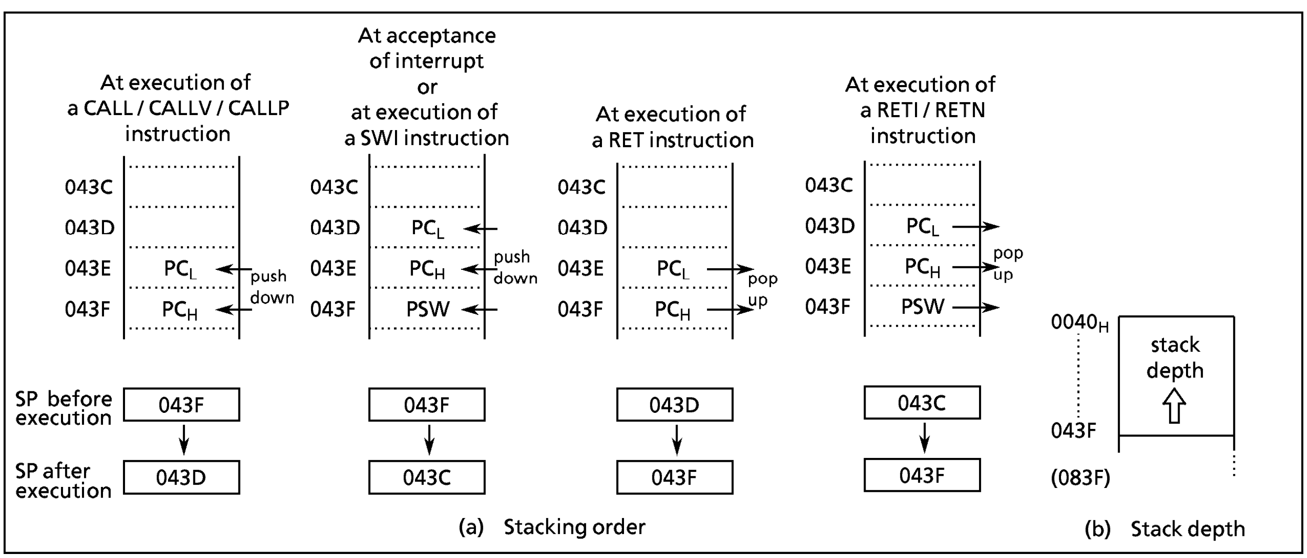


Figure 1-9. Stack

1.8 System Clock Controller

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

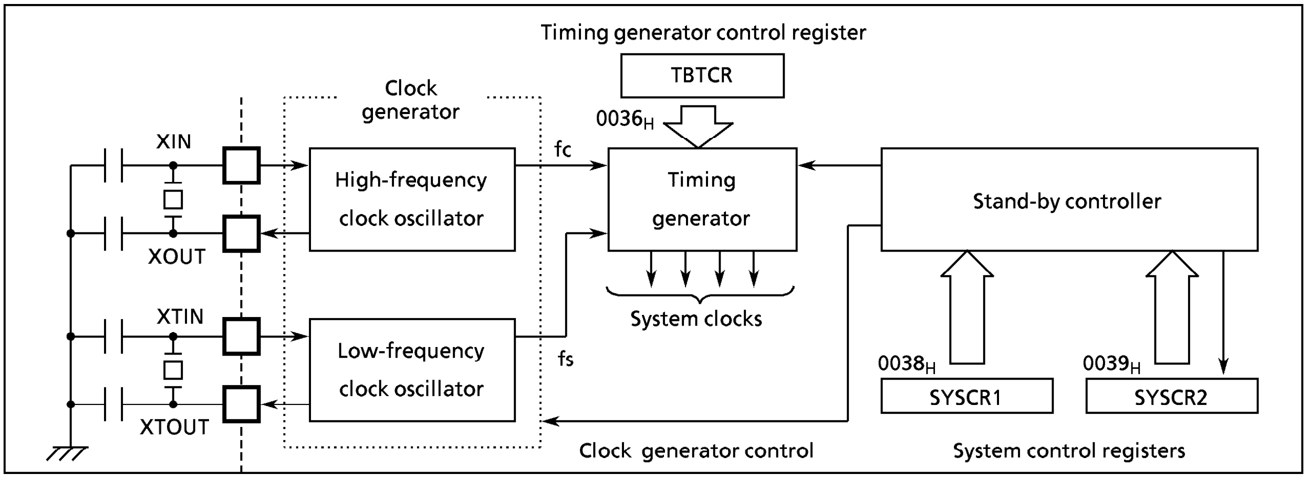


Figure 1-10. System Clock Controller

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 two oscillation circuits: one for the high-frequency clock and one for the low-frequency clock. Power consumption can be reduced by switching of the system clock controller to low-power operation based on the low-frequency clock.

The high-frequency (fc) and low-frequency (fs) clocks can be easily obtained by connecting a resonator between the XIN / XOUT and XTIN / XTOUT pins, respectively. Clock input from an external oscillator is also possible. In this case, external clock is applied to the XIN / XTIN pin with the XOUT / XTOUT pin not connected. The 87CM39/P39/S39 are not provided an RC oscillation.

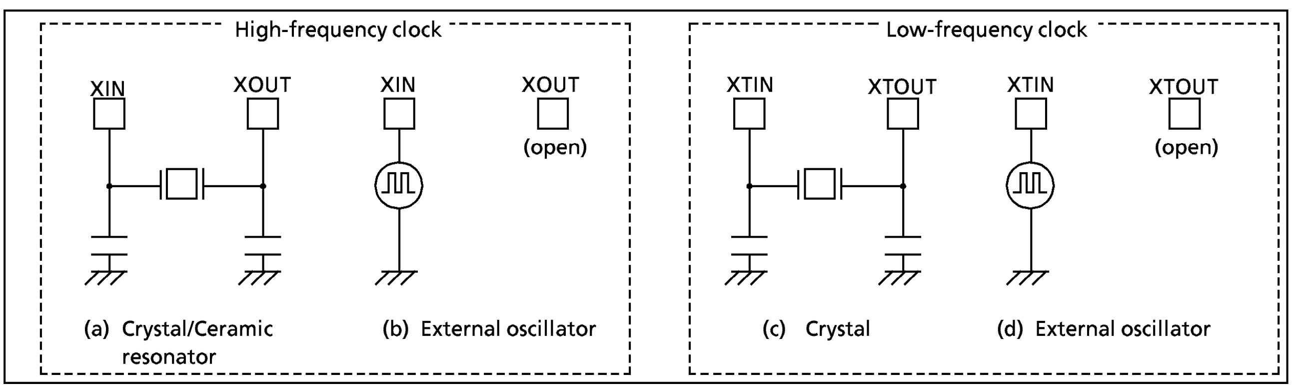
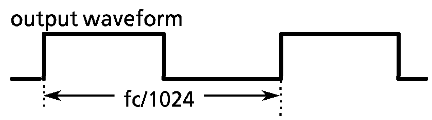


Figure 1-11. Examples of Resonator Connection

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 making the program to output fixed frequency pulses to the port while disabling all interrupts and monitoring this pulse. With a system requiring adjustment of the oscillation frequency, the adjusting program must be created beforehand.

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

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



1.8.2 Timing Generator

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

① Generation of main system clock
② Generation of divider output ($\overline{DVO}$) pulses
③ Generation of source clocks for time base timer
④ Generation of source clocks for watchdog timer
⑤ Generation of internal source clocks for timer / counters TC1 – TC4
⑥ Generation of warm-up clocks for releasing STOP mode
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. An input clock to the 7th stage of the divider depends on the operating mode and DV7CK (bit 4 in TBTCR) shown in Figure 1-12 as follows.
During reset and at releasing STOP mode, the divider is cleared to “0”, however, the prescaler is not cleared.

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

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

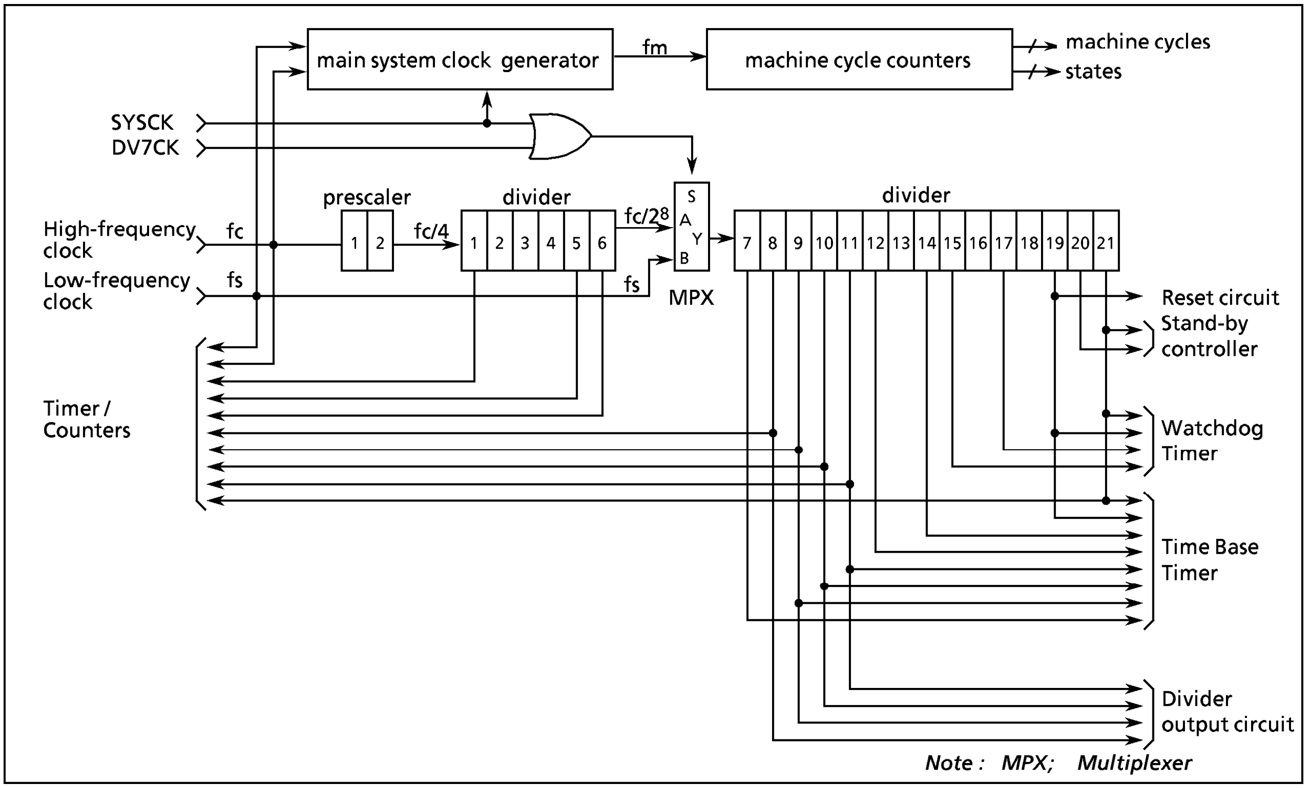


Figure 1-12. Configuration of Timing Generator

TBTCR (0036_H)

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

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

DV7CK	Selection of input clock to the 7th stage of the divider	0 : $fc/2^8$ [Hz] 1 : fs	Write only

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

Note 2 : Do not set DV7CK to "1" in the single-clock mode.

Note 3 : Do not set DV7CK to "1" before low-frequency clock is stable in the dual-clock mode.

Note 4 : TBTCR is a write-only register and must not be used with any of read-modify-write instructions.

Figure 1-13. Timing Generator Control Register

(2) Machine Cycle

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

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

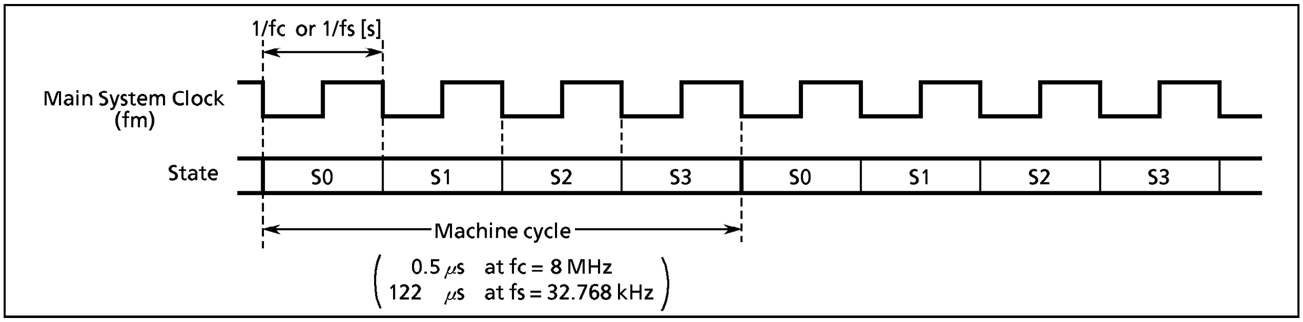


Figure 1-14. Machine Cycle

1.8.3 Stand-by Controller

The stand-by controller starts and stops the oscillation circuits for the high-frequency and low-frequency clocks, and switches the main system clock. There are two operating modes: single-clock and dual-clock. These modes are controlled by the system control registers (SYSCR1, SYSCR2).
Figure 1-15 shows the operating mode transition diagram and Figure 1-16 shows the system control registers. Either the single-clock or the dual-clock mode can be selected by an option during reset.

(1) **Single-clock mode**

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

① NORMAL1 mode

In this mode, both the CPU core and on-chip peripherals operate using the high-frequency clock. In the case where the single-clock mode has been selected as an option, the 87CM39/P39/S39 are placed in this mode after reset.

② IDLE1 mode

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

③ STOP1 mode

In this mode, the internal oscillation circuit is turned off, causing all system operations to be halted. The internal status immediately prior to the halt is held with the lowest power consumption during this mode. The output status of all output ports can be set to either output hold or high-impedance under software control.
STOP1 mode is started by setting STOP bit in the system control register 1 (SYSCR1), and STOP1 mode is released by an input (either level-sensitive or edge-sensitive can be programmably selected) to the $\overline{\text{STOP}}$ pin. After the warming-up period is completed, the execution resumes with the next instruction which follows the STOP mode start instruction.

(2) **Dual-clock mode**

Both high-frequency and low-frequency oscillation circuits are used in this mode. Pins P21 (XTIN) and P22 (XTOUT) cannot be used as input/output ports. The main system clock is obtained from the high-frequency clock in NORMAL2 and IDLE2 modes, and is obtained from the low-frequency clock in SLOW and SLEEP modes. The machine cycle time is 4/fc [s] (0.5 μs at fc = 8 MHz) in NORMAL2 and IDLE2 modes, and 4/fs [s] (122 μs at fs = 32.768 kHz) in SLOW and SLEEP modes. *Note that the 87PS39 is placed in the single-clock mode during reset.* To use the dual-clock mode, the low-frequency oscillator should be turned on by executing [SET (SYSCR2).XTEN] instruction.

① NORMAL2 mode

In this mode, the CPU core operates using the high-frequency clock. On-chip peripherals operate using the high-frequency clock and / or low-frequency clock. In case that the dual-clock mode has been selected by an option, the 87CM39/P39/S39 are placed in this mode after reset.

② SLOW mode

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

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

③ IDLE2 mode

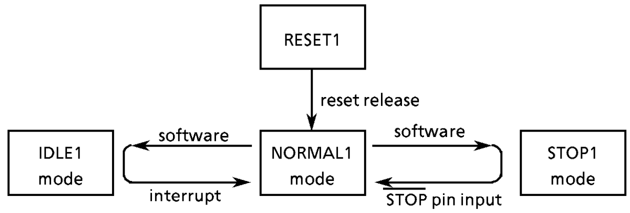
In this mode, the internal oscillation circuits remain active. The CPU and the watchdog timer are halted; however, on-chip peripherals remain active (operate using the high-frequency clock and / or the low-frequency clock). Starting and releasing of IDLE2 mode are the same as for IDLE1 mode, except that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

④ SLEEP mode

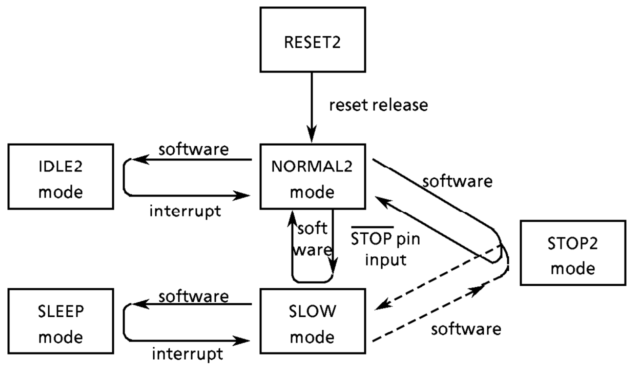
In this mode, the internal oscillation circuit of the low-frequency clock remains active. The CPU, the watchdog timer, and the internal oscillation circuit of the high- frequency clock are halted; however, on-chip peripherals remain active (operate using the low-frequency clock). Starting and releasing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

⑤ STOP2 mode

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



(a) Single-clock mode



(b) Dual-clock mode

Note 1 : NORMAL1 and NORMAL2 modes are generically called NORMAL; STOP1 and STOP2 are called STOP; and IDLE1, IDLE2 and SLEEP are called IDLE.

Note 2 : There is not a RESET2 mode in 87PS39.

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

Figure 1-15. Operating Mode Transition Diagram

System Control Register 1

SYSCR1 (0038H)

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

(Initial value: 0000 00**)

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

Note 1 : Always set RETM to "0" when transiting from NORMAL1 mode to STOP1 mode and from NORMAL2 mode to STOP2 mode. Always set RETM to "1" when transiting from SLOW mode to STOP2 mode.
Note 2 : When STOP mode is released with $\overline{\text{RESET}}$ pin input, a return is made to NORMAL mode regardless of the RETM contents.
Note 3 : fc ; *high-frequency clock* [Hz]
fs ; *low-frequency clock* [Hz]
* ; *don't care*
Note 4 : Bits 1 and 0 in SYSCR1 are read in as undefined data when a read instruction is executed.
Note5 : If 87CM39/P39/S39 is moved to STOP mode while OUTEN = "0", internal inputs fix "0". Then there is a possibility to set interrupt of falling edge.

System Control Register 2

SYSCR2 (0039H)

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

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

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

Note 1 : A reset is applied ($\overline{\text{RESET}}$ pin output goes low) if both XEN and XTEN are cleared to "0".
Note 2 : Do not clear XEN to "0" when SYSCK = 0, and do not clear XTEN to "0" when SYSCK = 1.
*Note 3 : * ; don't care*
Note 4 : Bits 3 - 0 in SYSCR2 are always read in as "1" when a read instruction is executed.
Note 5 : An optional initial value can be selected for XTEN. Always specify when ordering ES (engineering sample). Initial value of XTEN in 87PS39 is "0".

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

Figure 1-16. System Control Registers

1.8.4 Operating Mode Control

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

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

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

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

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

In this mode, STOP mode is released by setting the $\overline{\text{STOP}}$ pin high. This mode is used for capacitor back-up when the main power supply is cut off and long term battery back-up.
When the $\overline{\text{STOP}}$ pin input is high, executing an instruction which starts the STOP mode will not place in STOP mode but instead will immediately start the release sequence (warm-up). Thus, to start STOP mode in the level-sensitive release mode, it is necessary for the program to first confirm that the $\overline{\text{STOP}}$ pin input is low. The following method can be used for confirmation:

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

Example : Starting STOP mode with an INT5 interrupt.

```
PINT5 : TEST  (P2) . 0                  ; To reject noise, the STOP mode does not start if
        JRS   F, SINT5                    port P20 is at high
        LD    (SYSCR1), 01000000B       ; Sets up the level-sensitive release mode.
        SET   (SYSCR1) . 7              ; Starts STOP mode
        LDW   (IL), 1110011101010111B   ; IL12, 11, 7, 5, 3 ← 0 (Clears interrupt latches)
SINT5 : RETI
```

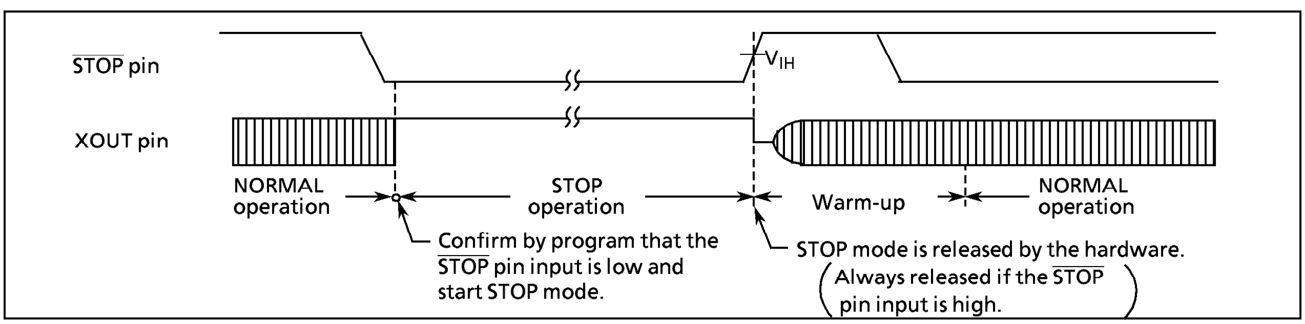


Figure 1-17. Level-sensitive Release Mode

Note1 : After warming up is started, when $\overline{\text{STOP}}$ pin input is changed "L" level, STOP mode is not placed.
Note2 : When changing to the level-sensitive release mode from the edge-sensitive release mode, the release mode is not switched until a rising edge of the $\overline{\text{STOP}}$ pin input is detected.

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

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

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

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

```
LD    (SYSCR1), 00000000B      ; OUTEN ← 0 (specifies high-impedance)
DI                             ; IMF ← 0 (disables interrupt service)
SET   (SYSCR1). STOP           ; STOP ← 1 (activates stop mode)
LDW   (IL), 1110011101010111B  ; IL12, 11, 7, 5, 3 ← 0
                                 (clears interrupt latches)
EI                             ; IMF ← 1 (enables interrupt service)
```

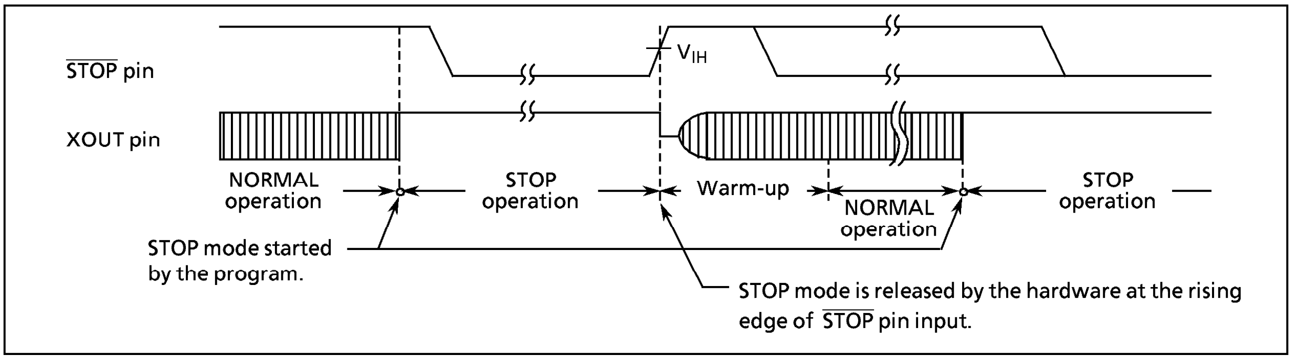


Figure 1-18. Edge-sensitive Release Mode

STOP mode is released by the following sequence:

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

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

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

Table 1-1. Warming-up Time Example

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

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

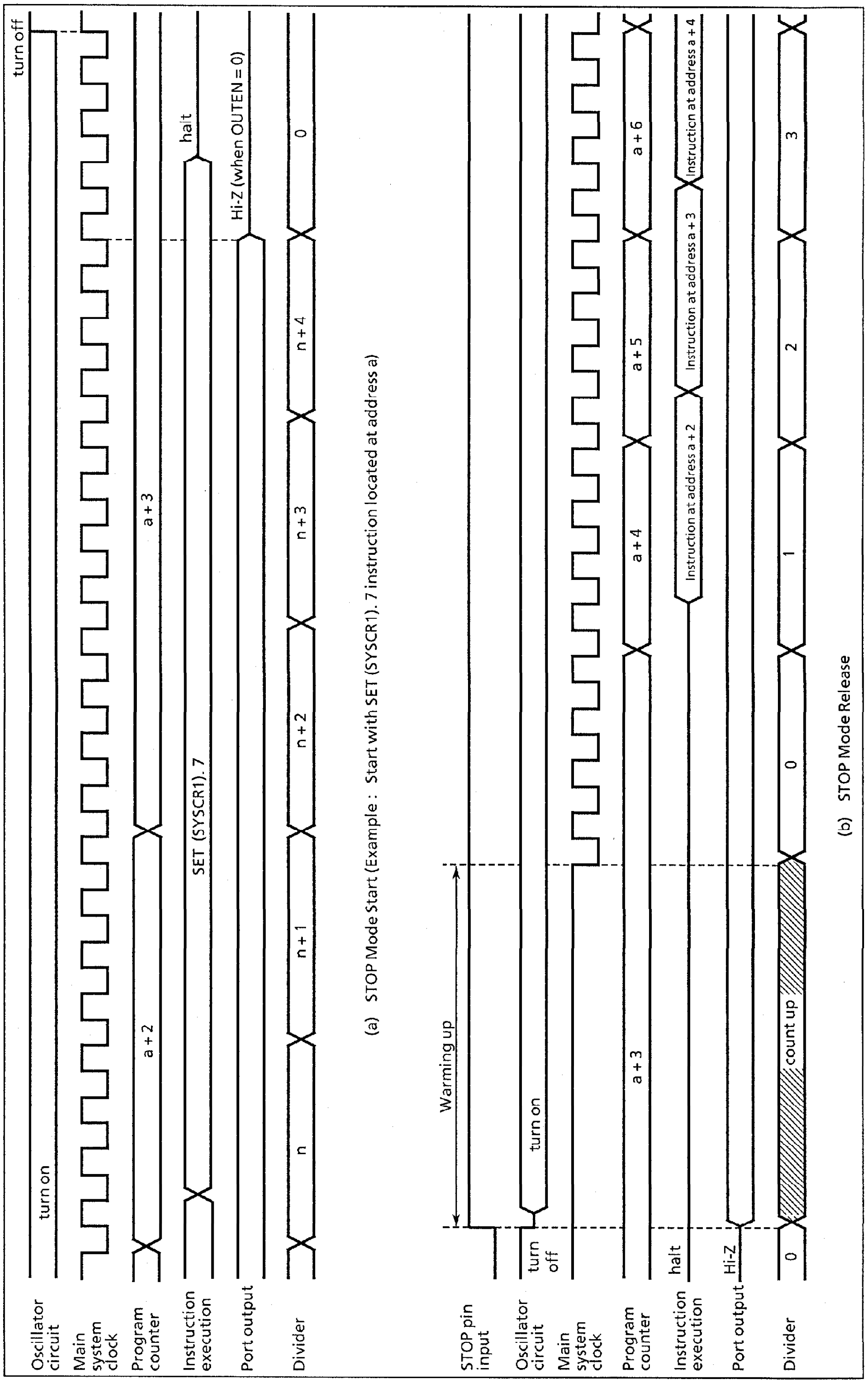


Figure 1-19. STOP Mode Start / Release

STOP mode can also be released by setting the $\overline{RESET}$ pin low, which immediately performs the normal reset operation. 87CM39/P39/S39 restarts NORMAL2 mode (NORMAL1 mode at 87PS39), even if setting is returning to SLOW mode.

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

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

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

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

Example : Starting IDLE mode.
```
SET    (SYSCR2) . 4    ; IDLE←1
```

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

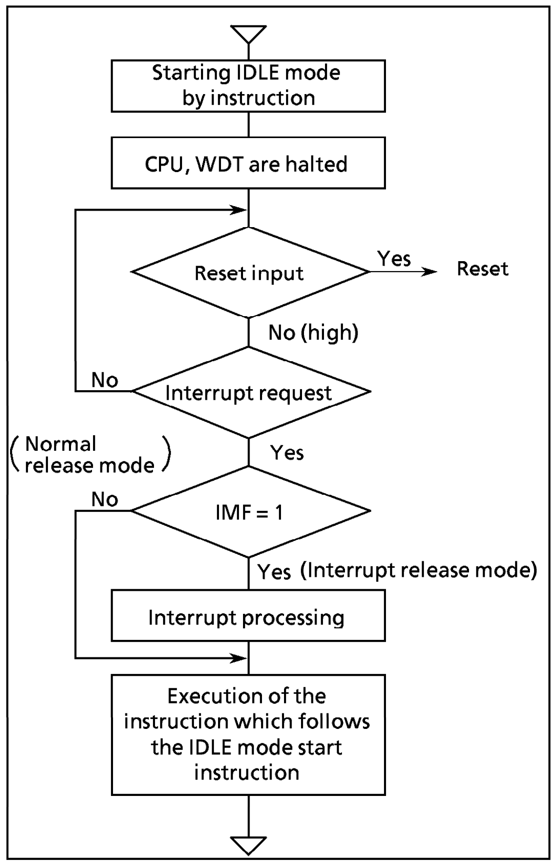


Figure 1-20. IDLE Mode

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

IDLE mode is released by any interrupt source enabled by the individual interrupt enable flag (EF) or an external interrupt 0 ($\overline{INT0}$ pin) request. Execution resumes with the instruction following the IDLE mode start instruction (e.g. [SET (SYSCR2).4]). Normally, IL (Interrupt Latch) of interrupt source to release IDLE mode must be cleared by load instructions.

b. Interrupt release mode (IMF = "1")

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

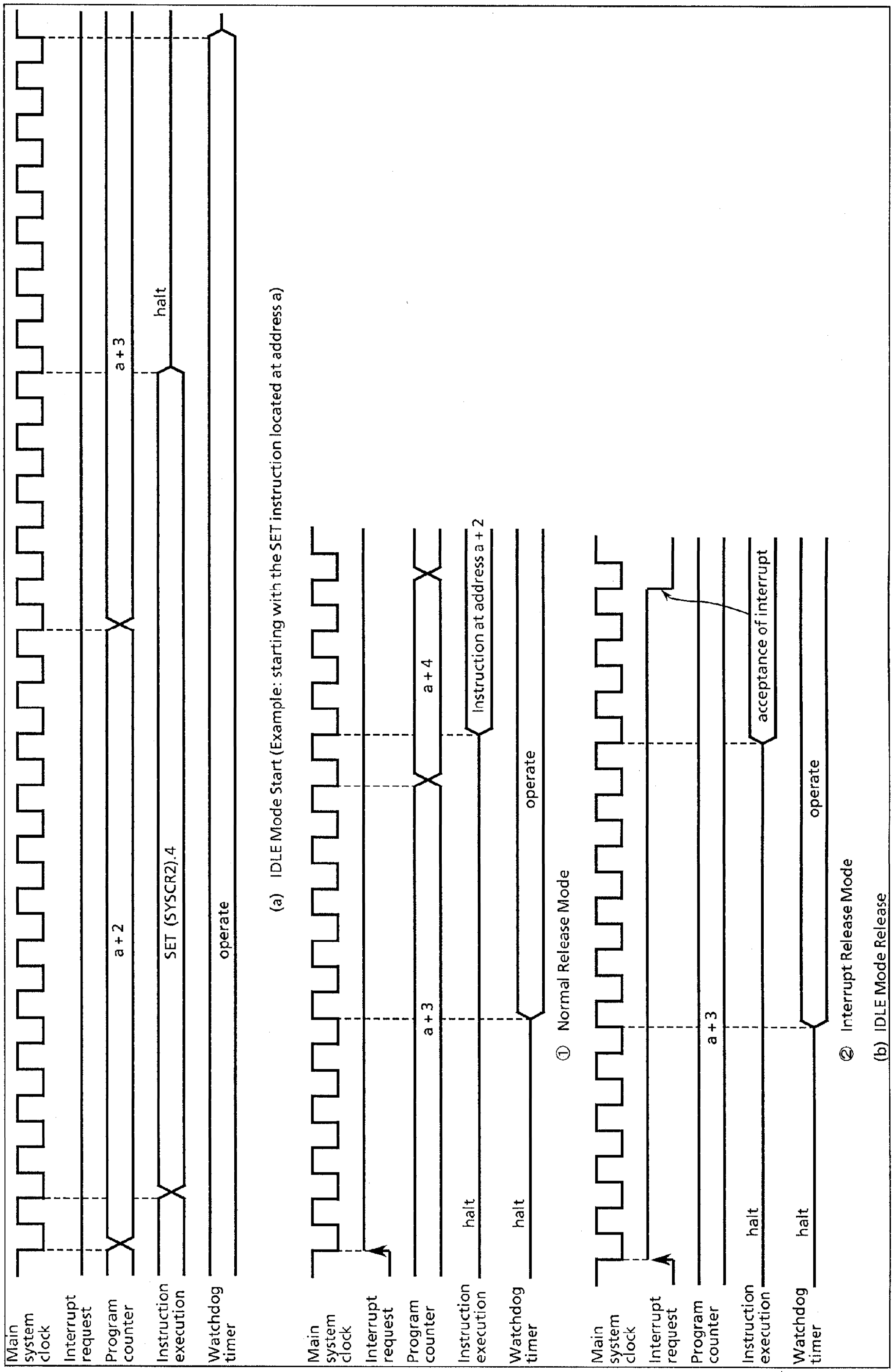


Figure 1-21. IDLE Mode Start / Release

IDLE mode can also be released by setting the $\overline{RESET}$ pin low, which immediately performs the reset operation. After reset, the 87CM39/P39/S39 are placed in NORMAL mode.

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

(3) **SLOW** mode

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

a. Switching from NORMAL2 mode to SLOW mode

First, set SYSCK (bit 5 in SYSCR2) to switch the main system clock to the low-frequency clock. Next, clear XEN (bit 7 in SYSCR2) to turn off high-frequency oscillation.
When the low-frequency clock oscillation is unstable, wait until oscillation stabilizes before performing the above operations. The timer / counter 2 (TC2) can conveniently be used to confirm that low-frequency clock oscillation has stabilized.

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

Example1 : Switching from NORMAL2 mode to SLOW mode.

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

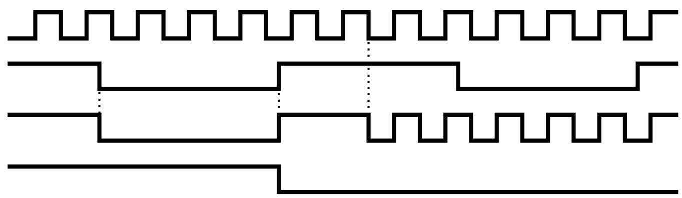
Example2 : Switching to SLOW mode after low-frequency clock oscillation has stabilized.

```
          LD      (TC2CR), 14H    ; Sets TC2 mode
                                    (timer mode, source clock : fs)
          LDW     (TREG2), 8000H  ; Sets warming-up time
                                    (according to Xtal characteristics)
          SET     (EIRH). EF14    ; INTTC2 interrupt enable
          LD      (TC2CR), 34H    ; Starts TC2
           ⋮
  PINTTC2 :       LD              (TC2CR), 10H ;  Stops TC2
          SET     (SYSCR2) . 5    ; SYSCK←1
          CLR     (SYSCR2) . 7    ; XEN←0
          RETI
           ⋮
  VINTTC2 :       DW              PINTTC2   ;  INTTC2 vector table
```

b. Switching from SLOW mode to NORMAL2 mode

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

Note1 : After SYSCK is cleared, executing instructions is continued by low-frequency clock while low-frequency clock and high-frequency clock synchronize each other.



*Note2 : **SLOW** mode can also be released by setting the* $\overline{\text{RESET}}$ *pin low, which immediately performs the reset operation. After reset, the **87CM39/P39/S39** are placed in **NORMAL2** mode. (The **87PS39** is placed in **NORMAL1** mode.)*

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

```
           SET   (SYSCR2) . 7        ; XEN←1     (turns on high-frequency oscillation)
           LD    (TC2CR), 10H        ; Sets TC2 mode
                                       (timer mode, source clock: fc)
           LD    (TREG2 + 1), 0F8H   ; Sets the warming-up time
                                       (according to frequency and resonator
                                       characteristics)
           SET   (EIRH). EF14        ; INTTC2 interrupt enable
           LD    (TC2CR), 30H        ; Starts TC2
            ⋮
PINTTC2 :  LD    (TC2CR), 10H        ; Stops TC2
           CLR   (SYSCR2) . 5        ; SYSCK←0   (Switches the main system clock to the
                                                  high-frequency clcok)
           RETI
            ⋮
VINTTC2 :  DW    PINTTC2             ; INTTC2 vector table
```

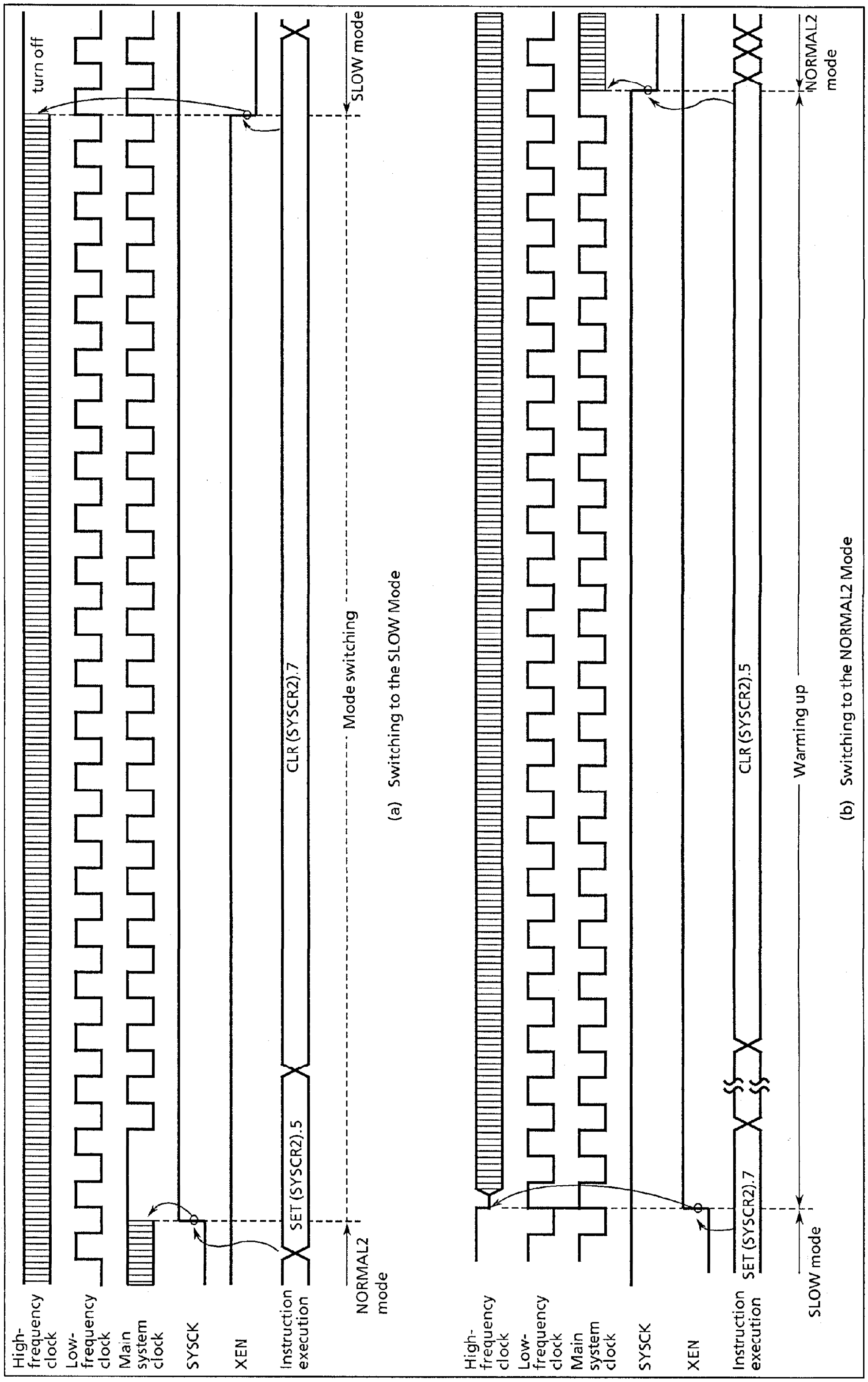


Figure 1-22. Switching between the NORMAL2 and SLOW Modes

1.9 Interrupt Controller

The 87CM39/P39/S39 has a total of 15 interrupt sources: 6 externals and 9 internals. Nested interrupt control with priorities is also possible. Two of the internal sources are pseudo non-maskable interrupts; the remainder are all maskable interrupts.
Interrupt latches (IL) that hold the interrupt requests are provided for interrupt sources. Each interrupt vector is independent.
The interrupt latch is set to "1" when an interrupt request is generated and requests the CPU to accept the interrupt. The acceptance of maskable interrupts can be selectively enabled and disabled by the program using the interrupt master enable flag (IMF) and the individual interrupt enable flags (EF). When two or more interrupts are generated simultaneously, the interrupt is accepted in the highest priority order as determined by the hardware. Figure 1-23. shows the interrupt controller.

Table 1-2. 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)	$IMF \cdot EF_4 = 1$	IL_4	$FFF6_H$	4
External	INT1	(External interrupt 2)	$IMF \cdot EF_5 = 1$	IL_5	$FFF4_H$	5
Internal	INTTBT	(Time Base Timer interrupt)	$IMF \cdot EF_6 = 1$	IL_6	$FFF2_H$	6
External	INT2	(External interrupt 2)	$IMF \cdot EF_7 = 1$	IL_7	$FFF0_H$	7
Internal	INTTC3	(8-bit TC3 interrupt)	$IMF \cdot EF_8 = 1$	IL_8	$FFEE_H$	8
Internal	INTSBI	(Serial Bus Interface interrupt)	$IMF \cdot EF_9 = 1$	IL_9	$FFEC_H$	9
Internal	INTTC4	(8-bit TC4 interrupt)	$IMF \cdot EF_{10} = 1$	IL_{10}	$FFEA_H$	10
External	INT3	(External interrupt 3, Remote control receive interrupt)	$IMF \cdot EF_{11} = 1$	IL_{11}	$FFE8_H$	11
External	INT4	(External interrupt 4)	$IMF \cdot EF_{12} = 1$	IL_{12}	$FFE6_H$	12
Internal	INTOSD	(OSD interrupt)	$IMF \cdot EF_{13} = 1$	IL_{13}	$FFE4_H$	13
Internal	INTTC2	(16-bit TC2 interrupt)	$IMF \cdot EF_{14} = 1$	IL_{14}	$FFE2_H$	14
External	INT5	(External interrupt 5)	$IMF \cdot EF_{15} = 1$	IL_{15}	$FFE0_H$	Low 15

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

Interrupt latches are provided for each source, except for a software interrupt. The latch is set to "1" when an interrupt request is generated, and requests the CPU to accept the interrupt. The latch is cleared to "0" just after the interrupt is accepted. All interrupt latches are initialized to "0" during reset. They must be cleared to "0" by software before enabling interrupt.
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.* Thus, interrupt requests can be cancelled and initialized by the program. Don't clear the IL_2 for a watchdog timer interrupt to "0". 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.

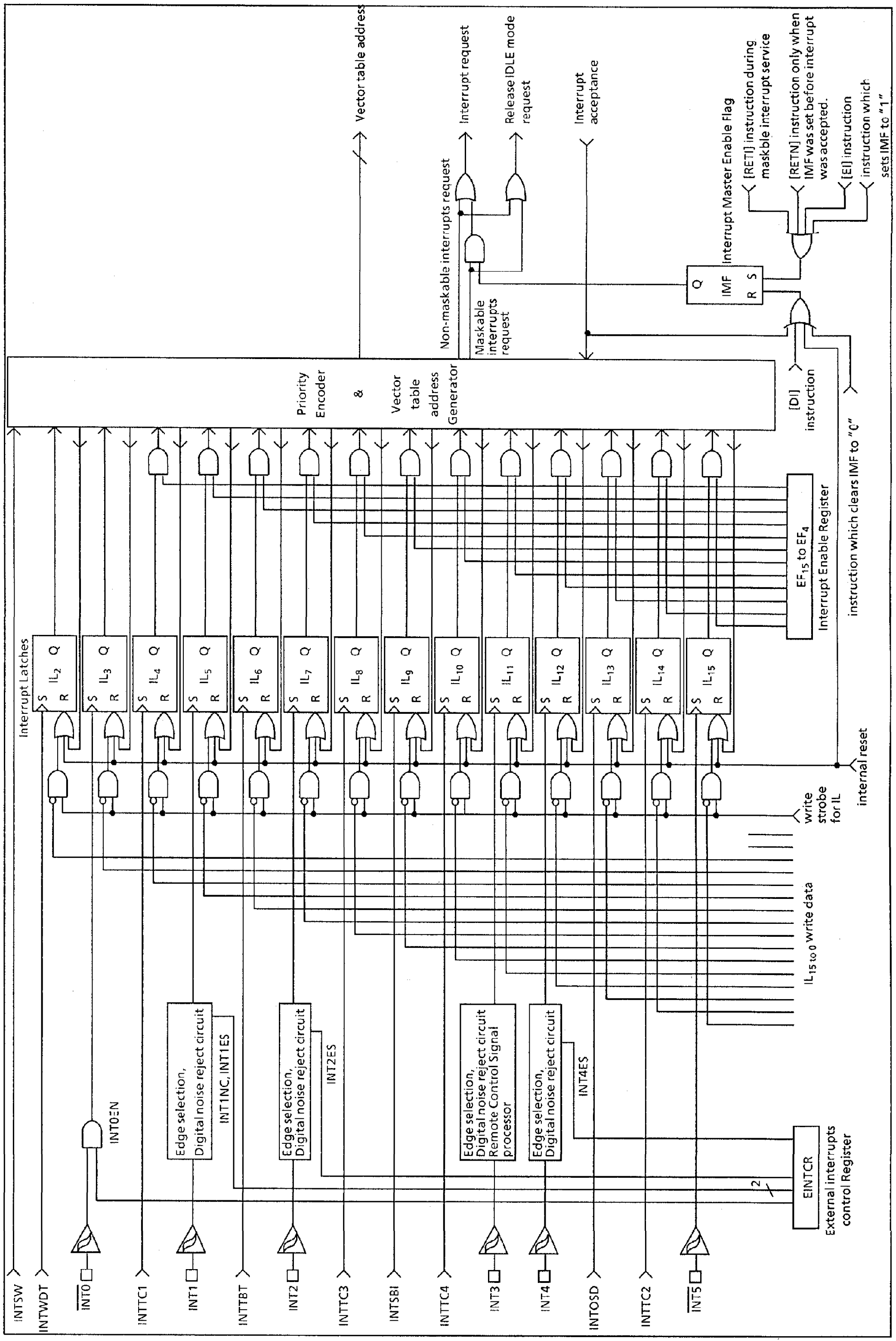


Figure 1-23. Interrupt Controller Block Diagram

Example 1 : Clears interrupt latches

```
LDW   (IL), 1110100000111111B  ;  IL12, IL10 to IL6←0
```

Example 2 : Reads interrupt latches

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

Example 3: Tests an interrupt latch

```
TEST  (IL).7                   ;  if IL7 = 1 then jump
JR    F, SSET
```

(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} to EF_4)

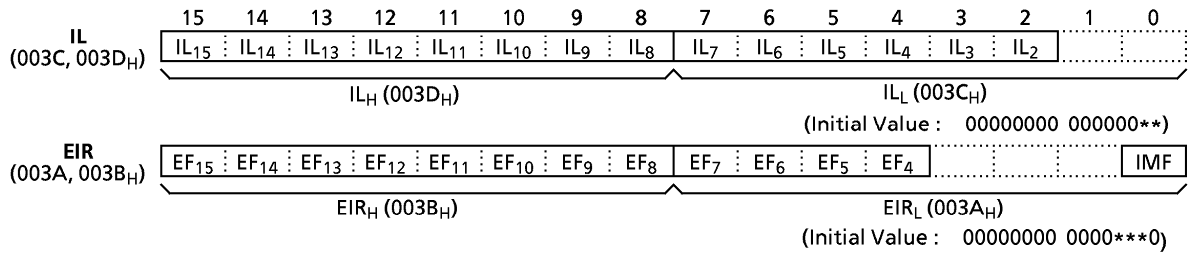
These flags enable and disable the acceptance of individual maskable interrupts, except for an external interrupt 0. Setting the corresponding bit of an individual interrupt enable flag to "1" enables acceptance of an interrupt, setting the bit to "0" disables acceptance.

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

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

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

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



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

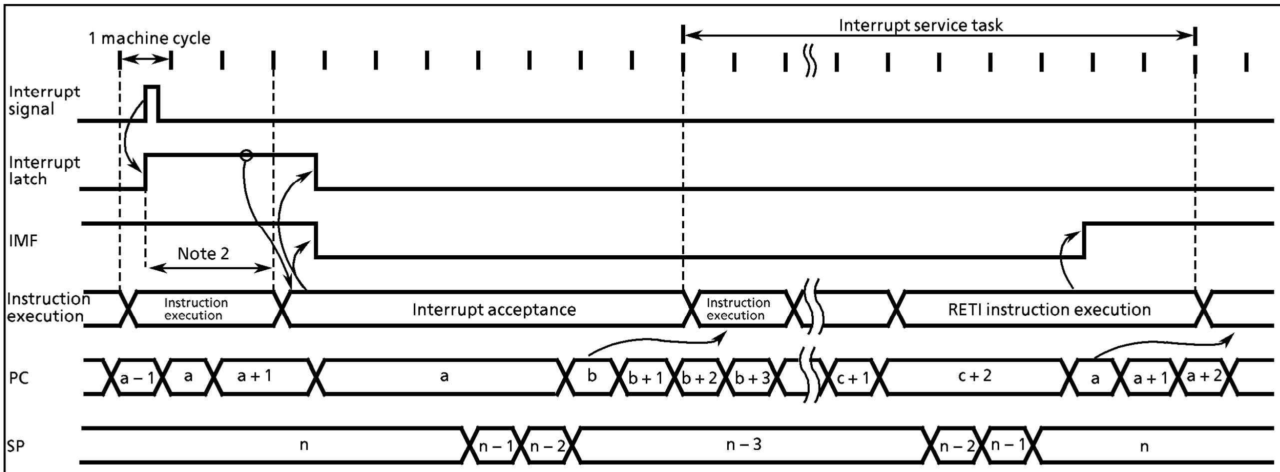
Figure 1-24. Interrupt Latch (IL) and Interrupt Enable Register (EIR)

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 = 8 MHz in NORMAL mode) after the completion of the current instruction execution. The interrupt service task terminates upon execution of an interrupt return instruction [RETI] (for maskable interrupts) or [RETN] (for pseudo non-maskable interrupts).

(1) Interrupt acceptance processing is as follows:

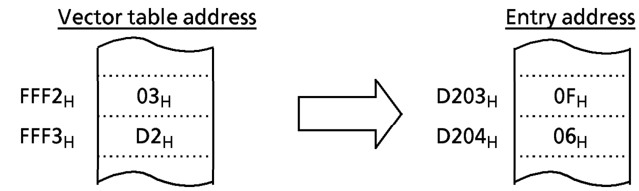
① The interrupt master enable flag (IMF) is cleared to "0" to temporarily disable the acceptance of any following maskable interrupts. When a non-maskable interrupt is accepted, the acceptance of any following interrupts is temporarily disabled.
② The interrupt latch (IL) for the interrupt source accepted 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 to 38/fs [s].

Figure 1-25. Timing Chart of Interrupt Acceptance and Interrupt Return Instruction

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



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

Example 1 : Disables an external interrupt 0 using INT0EN:

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

Example 2 : Disables the processing of external interrupt 0 under the software control (using bit 0 at address 00F0H as the interrupt processing disable switch):

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

(2) Saving / Restoring General Purpose Registers

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 (1 µs at 8 MHz)
                interrupt processing
           RETI                    ;  Restores bank and Returns
```

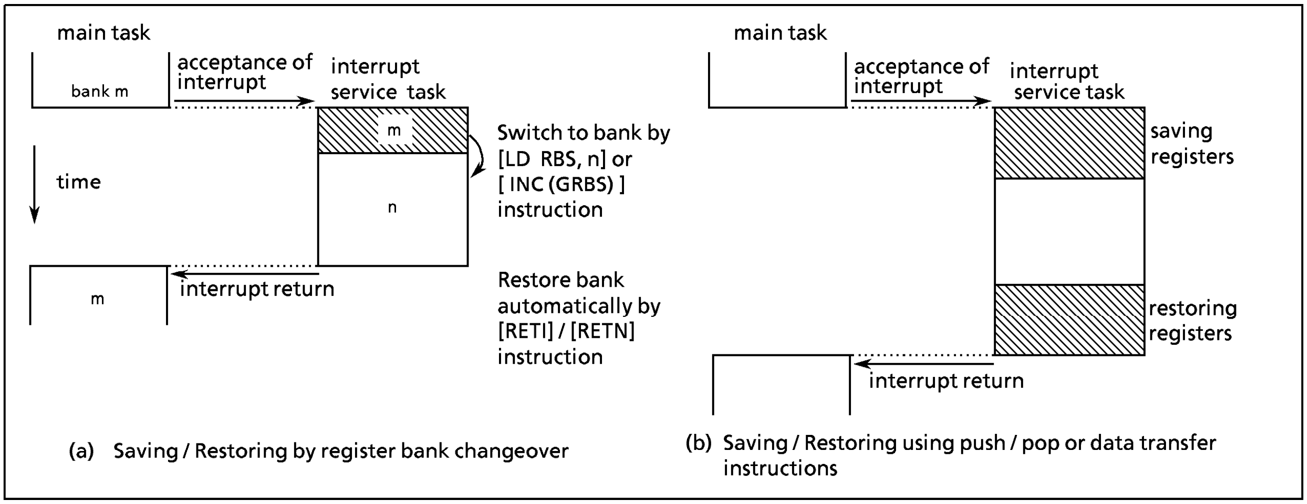
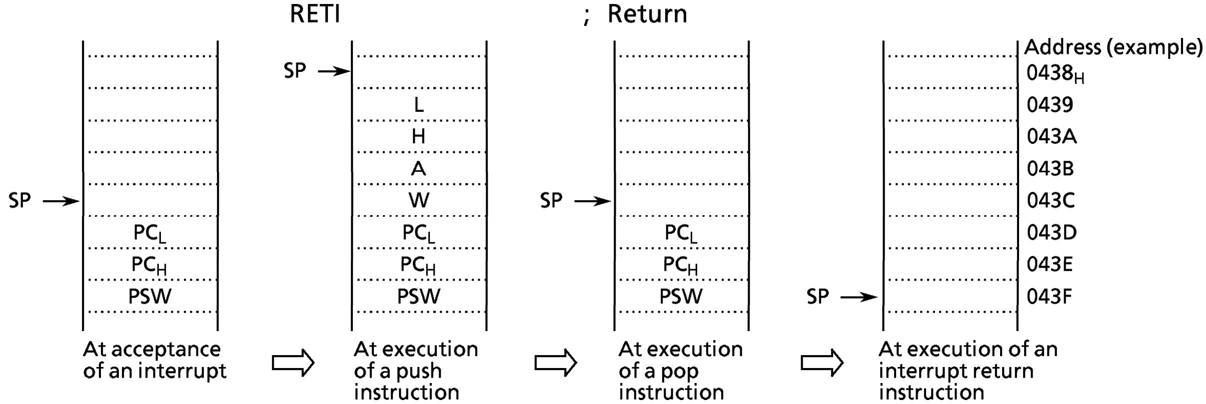


Figure 1-26. Saving / Restoring General-purpose Registers

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
```



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

Example : Saving / restoring a register using data transfer instructions

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

(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 soft ware 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 1080_H to $10FF_H$ (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 87CM39/P39/S39 each have six external interrupt inputs ($\overline{INT0}$, INT1, INT2, INT3, INT4, and $\overline{INT5}$). Four of these are equipped with digital noise rejection circuits (pulse inputs of less than a certain time are eliminated as noise). Edge selection is also possible with INT1, INT2, INT3 and INT4.

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

Edge selection, noise rejection control except INT3 pin input and $\overline{INT0}$ / P10 pin function selection are performed by the external interrupt control register (EINTCR). Edge selecting and noise rejection control for INT3 pin input are preformed by the Remote control signal preprocessor control registers. (refer to the section of the Remote control signal preprocessor.) When INT0EN = 0, the IL_3 will not be set even if the falling edge of $\overline{INT0}$ pin input is detected.

Table 1-3. External Interrupts

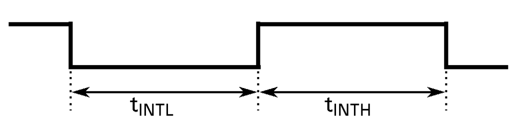
Source	Pin	Secondary function pin	Enable conditions	Edge	Digital noise rejection
INT0	$\overline{INT0}$	P10	IMF = 1, INT0EN = 1	falling edge	— (hysteresis input)
INT1	INT1	P11	$IMF \cdot EF_5 = 1$	falling edge or rising edge	Pulses of less than 15/fc or 63/fc [s] are eliminated as noise. Pulses equal 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 regarded as signals.
INT3	INT3	P30/RXIN	$IMF \cdot EF_{11} = 1$	falling edge, rising edge or falling / rising edge	Refer to the section of the Remote control preprocessor
INT4	INT4	P32	$IMF \cdot EF_{12} = 1$	falling edge or rising edge	Pulses of less than 7/fc [s] are eliminated as noise. Pulses of 24/fc [s] or more are considered to be signals.
INT5	$\overline{INT5}$	P20 / $\overline{STOP}$	$IMF \cdot EF_{15} = 1$	falling edge	

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

Note 2 : The noise rejection function is also affected for timer/counter input (TC1 pin).

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

$\overline{INT0}$ / $\overline{INT5}$ input



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

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

① INT1 pin 49/fc [s] (INT1NC = 1) , 193/fc [s] (INT1NC = 0)
② INT2,INT4 pins 25/fc [s]
③ INT3 pin Refer to the section of the Remote control preprocessor.

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

Example : Activating stop mode (TMP87CM39/P39/S39) :

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

EINTCR (0037H)

7	6	5	4	3	2	1	0
INT1 NC	INT0 EN	(TC4 ES)	INT4 ES	(TC3 ES)	INT2 ES	INT1 ES	

(Initial value : 00*0 000*)

INT1NC	Noise reject time select	0 : Pulses of less than 63/fc [s] are eliminated as noise 1 : Pulses of less than 15/fc [s] are eliminated as noise	write only
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)	
INT4 ES INT2 ES INT1 ES	INT4, INT2 and 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 and INT4ES during NORMAL1/2 mode, clear interrupt latches of external interrupt inputs (INT2 and INT4) after 8 machine cycles from the time of rewriting. During SLOW mode, 3 machine cycles are required.

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

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

```
                LD (TC1CR),01001000B     ;  TC1S ← 00 (stops TC1)
                DI                       ;  IMF ← 0 (disables interrupt service)
                LD (EINTCR),00000100B    ;  INT2ES ← 1 (change edge selection)
   ↑            NOP
8 machine       to
  cycles        NOP
   ↓            LD (ILL),01111111B       ;  IL7 ← 0 (clears interrupt latch)
                EI                       ;  IMF ← 1 (enables interrupt service)
                LD (TC1CR),01111000B     ;  TC1S ← 11 (starts TC1)
```

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

Figure 1-27. 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 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

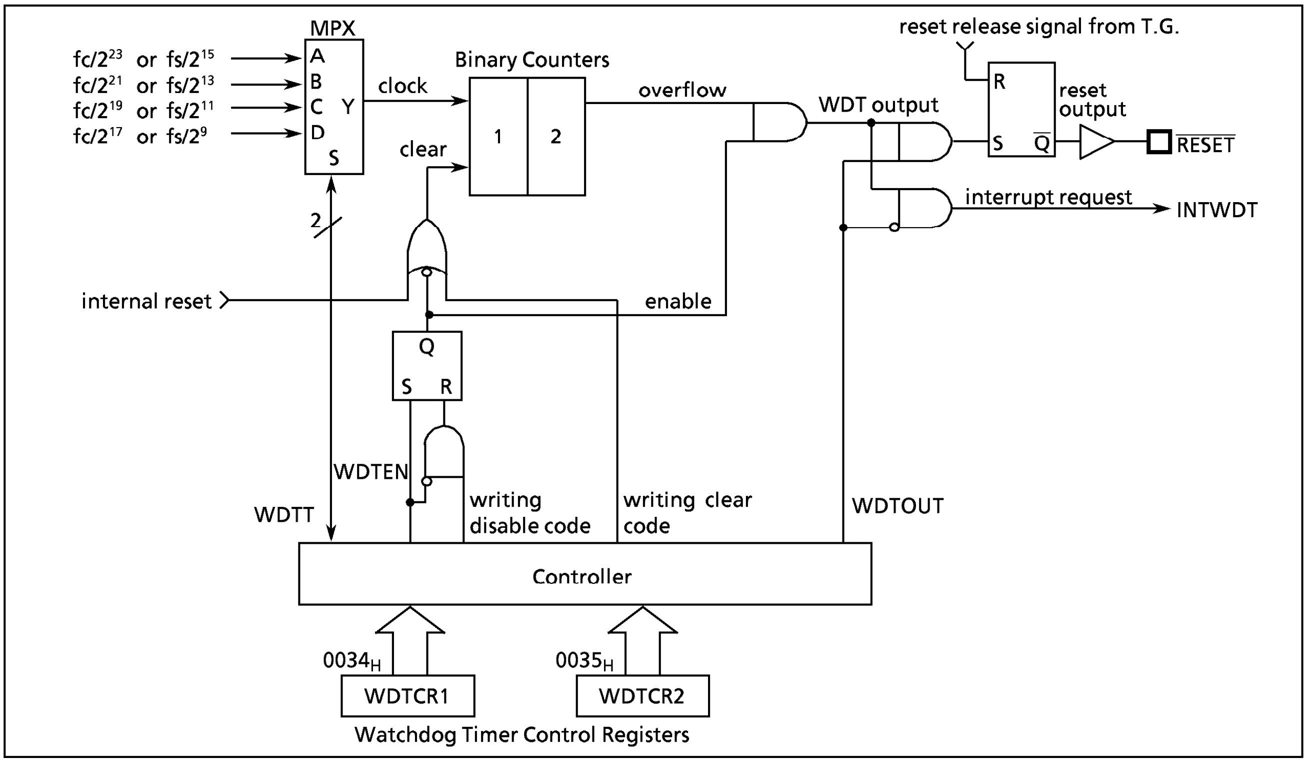


Figure 1-28. Watchdog Timer Configuration

1.10.2 Watchdog Timer Control

Figure 1-29 shows the watchdog timer control registers (WDTCR1, WDTCR2). The watchdog timer is automatically enabled after reset.

(1) Malfunction detection methods using the watchdog timer

The CPU malfunction is detected as follows.

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

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

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

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

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

```
                      LD    (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

WDTCR1 (0034_H)

7	6	5	4	3	2	1	0
				WDTEN	WDTT		WDTOUT

(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 or 2^{17}/fs [s] 01 : 2^{23}/fc or 2^{15}/fs 10 : 2^{21}/fc or 2^{13}/fs 11 : 2^{19}/fc or 2^{11}/fs	
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] fs ; Low-frequency clock [Hz] * ; don't care*
Note 3 : WDTCR1 is a write-only register and must not be used with any of read-modify-write instructions.
Note 4 : Disable the watchdog timer or clear the counter just before switching to STOP mode. When the counter is cleared just before switching to STOP mode, clear the counter again subsequently to releasing STOP mode.

Watchdog Timer Control Register 2

WDTCR2 (0035_H)

7	6	5	4	3	2	1	0

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

WDTCR2	Watchdog timer control code write register	$4E_H$: Watchdog timer binary counter clear (clear code) $B1_H$: 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 1-29. Watchdog Timer Control Registers

Table 1-4. Watchdog Timer Detection Time

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

(2) Watchdog Timer Enable

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

Example : Enables watchdog timer
```
LD      (WDTCR1), 00001000B ;   WDTEN←1
```

(3) Watchdog Timer Disable

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

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

Example : Disables watchdog timer
```
LDW     (WDTCR1) , 0B101H  ;  WDTEN←0, WDTCR2←disable code
```

1.10.3 Watchdog Timer Interrupt (INTWDT)

This is a pseudo non-maskable interrupt which can be accepted regardless of the contents of the EIR. If a watchdog timer interrupt or a software interrupt is already accepted, however, the new watchdog timer interrupt waits until the previous 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, 043FH               ;  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 with pull-up registor) low to reset the internal hardware and the external circuits. The reset output time is 12/fc [s] (1.5 μs at fc = 8 MHz). The high-frequency clock oscillator also turns on when a watchdog timer reset is generated in SLOW mode.

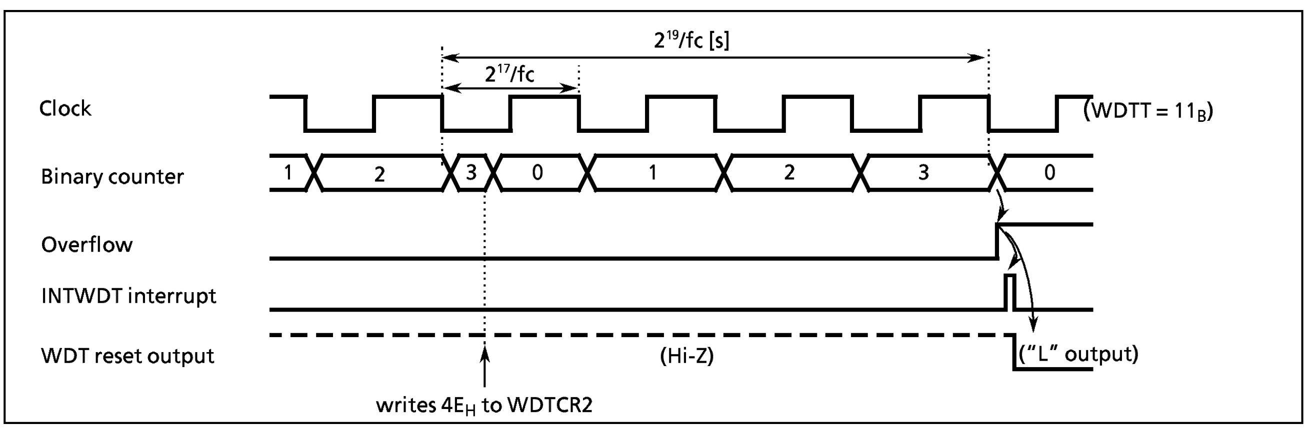


Figure 1-30. Watchdog Timer Interrupt / Reset

1.11 Reset Circuit

The TLCS-870 series has four types of reset generation procedures: an external reset input, a watchdog timer reset and a system-clock-reset. Table 1-5 shows on-chip hardware initialization by reset action.
The internal source reset circuit (watchdog timer reset, and system clock reset) is not initialized when power is turned on. Thus, output from the RESET pin may go low (12/fc [s] 1.5 μs at 8 MHz) when power is turned on.

Table 1-5. Initializing Internal Status by Reset Action

On-chip Hardware		Initial Value	On-chip Hardware	Initial Value
Program counter	(PC)	$(FFFF_H) \cdot (FFFE_H)$	Divider of Timing generator	0
Register bank selector	(RBS)	0	Watchdog timer	Enable
Jump status flag	(JF)	1		
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 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 RESET pin input goes high, the reset operation is released and the program execution starts at the vector address stored at addresses $FFFE_H$ to $FFFF_H$.
The 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.

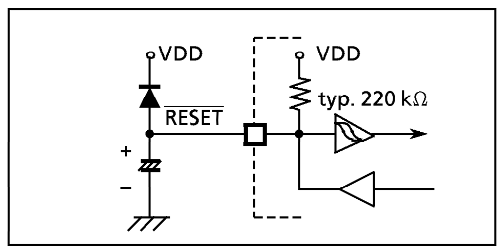
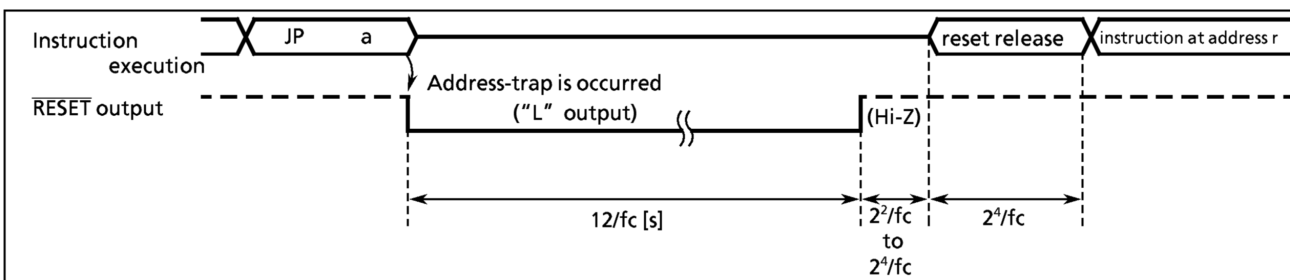


Figure 1-31. Simple Power-on-Reset Circuitry

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 - $043F_H$ [87CM39] / 0000_H to $083F_H$ [87CP38/S38]), an address-trap-reset will be generated. Then, the RESET pin output will go low. The reset time is 12/fc [s] (1.5μs at fc = 8 MHz).



Note 1: $0 \leqq a \leqq 043F_H$ (87CM39), $0 \leqq a \leqq 083F_H$ (87CP38/S38)
Note 2: During reset release, a reset vector "r" is read out, and an instruction at address r is fetched and decoded.

Figure 1-32. Address-Trap-Reset

1.11.3 Watchdog Timer Reset

Refer to Section "1.10 Watchdog Timer".

1.11.4 System-Clock-Reset

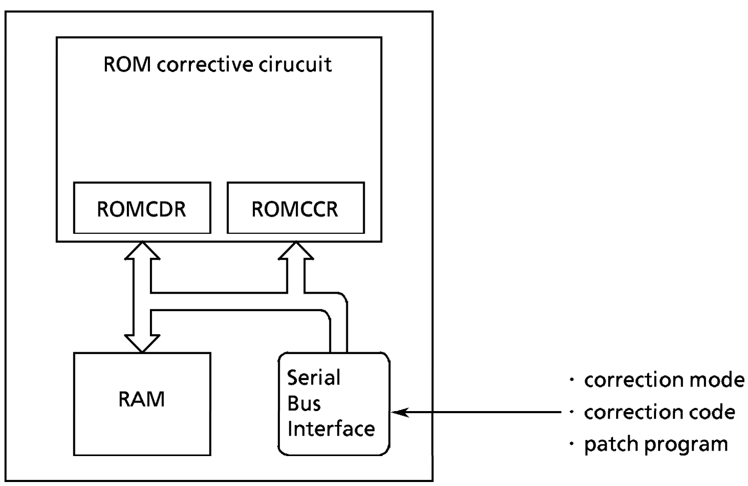
Clearing both XEN and XTEN (bits 7 and 6 in SYSCR2) to "0" stops both high-frequency and low-frequency oscillation, and causes the MCU to deadlock. This can be prevented by automatically generating a reset signal whenever XEN = XTEN = 0 is detected to continue the oscillation. Then, the $\overline{\text{RESET}}$ pin output goes low from high-impedance. The reset time is 12/fc [s] (1.5 μs at 8 MHz).

1.12 ROM Corrective Function

The ROM corrective function can patch the part (s) of on-chip ROM with some bugs. The patched data should be loaded from an external memory at the initialized routine beforehand. The figure below shows one example of configurations loading the patched data via I^{2}C-bus. The ROM corrective function have two modes. One is to replace the instruction on a certain address in the ROM with the jump instruction to branch into the RAM area where the patched codes and/or data are loaded (Program Jump Mode) . The other is to replace a byte or a word (2 byte) length data in the ROM with the patched data (Data Replacement Mode) . When the ROM corrective function is enabled, the address-trap-reset is automatically disabled on the RAM area from 0240_H where the patched program is running.

Note1 : When use ROM correction circuit, it is necessary to contain a program which operates to load patch data (program code) from external memory to internal data RAM in an initial routine.
Note2 : BM87CS39N0A does not support the ROM corrective function.

Example :



1.12.1 Configuration

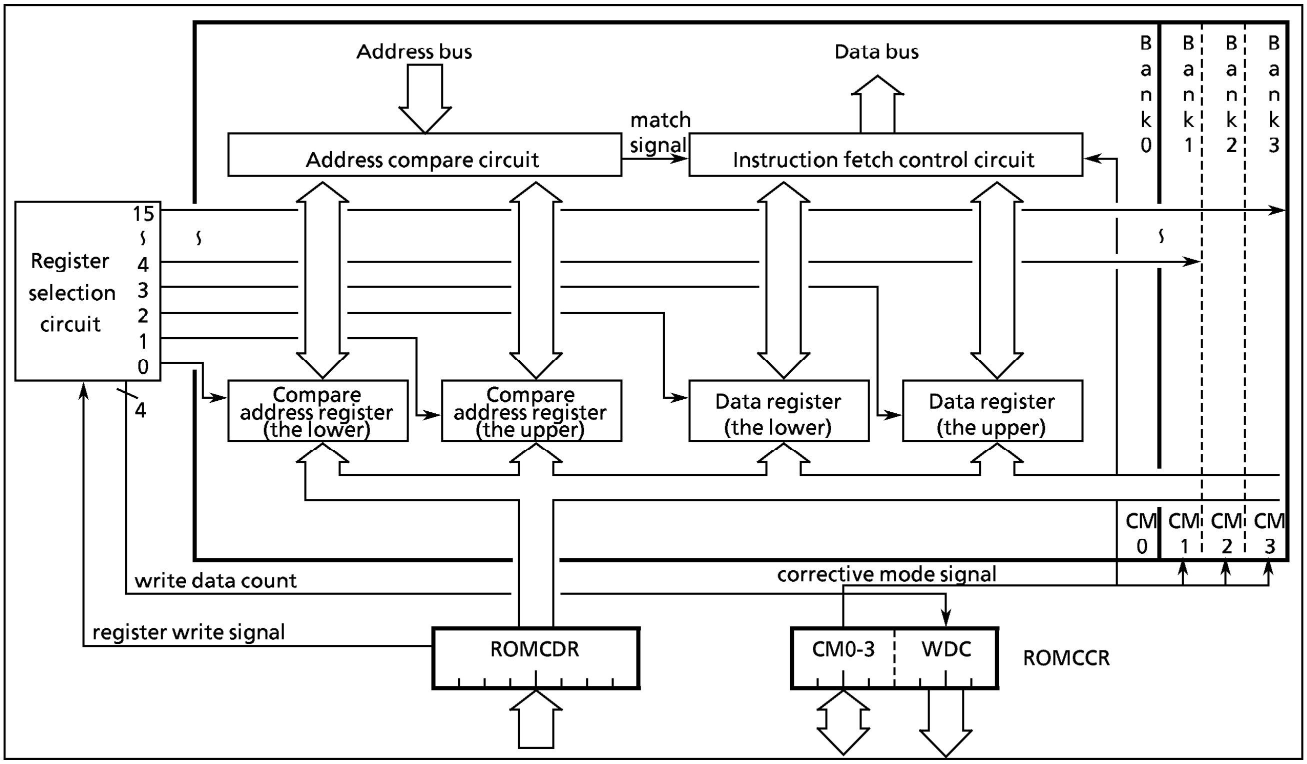


Figure 1-33. ROM Corrective Circuit

1.12.2 Control

The ROM corrective function is controlled by ROM corrective control register (ROMCCR) and ROM corrective data register (ROMCDR).

ROMCCR ($0FC0_H$)

	7	6	5	4	3	2	1	0
ROMCCR ($0FC0_H$)	CM3	CM2	CM1	CM0	WDC			

(Initial value 0000 0000)

CM3	corrective mode setting (BANK3)	0 : program jump mode 1 : data replacement mode	Read/Write
CM2	corrective mode setting (BANK2)		
CM1	corrective mode setting (BANK1)		
CM0	corrective mode setting (BANK0)		
WDC	write data counter	Counting the number of the byte written in ROMCDR	read only

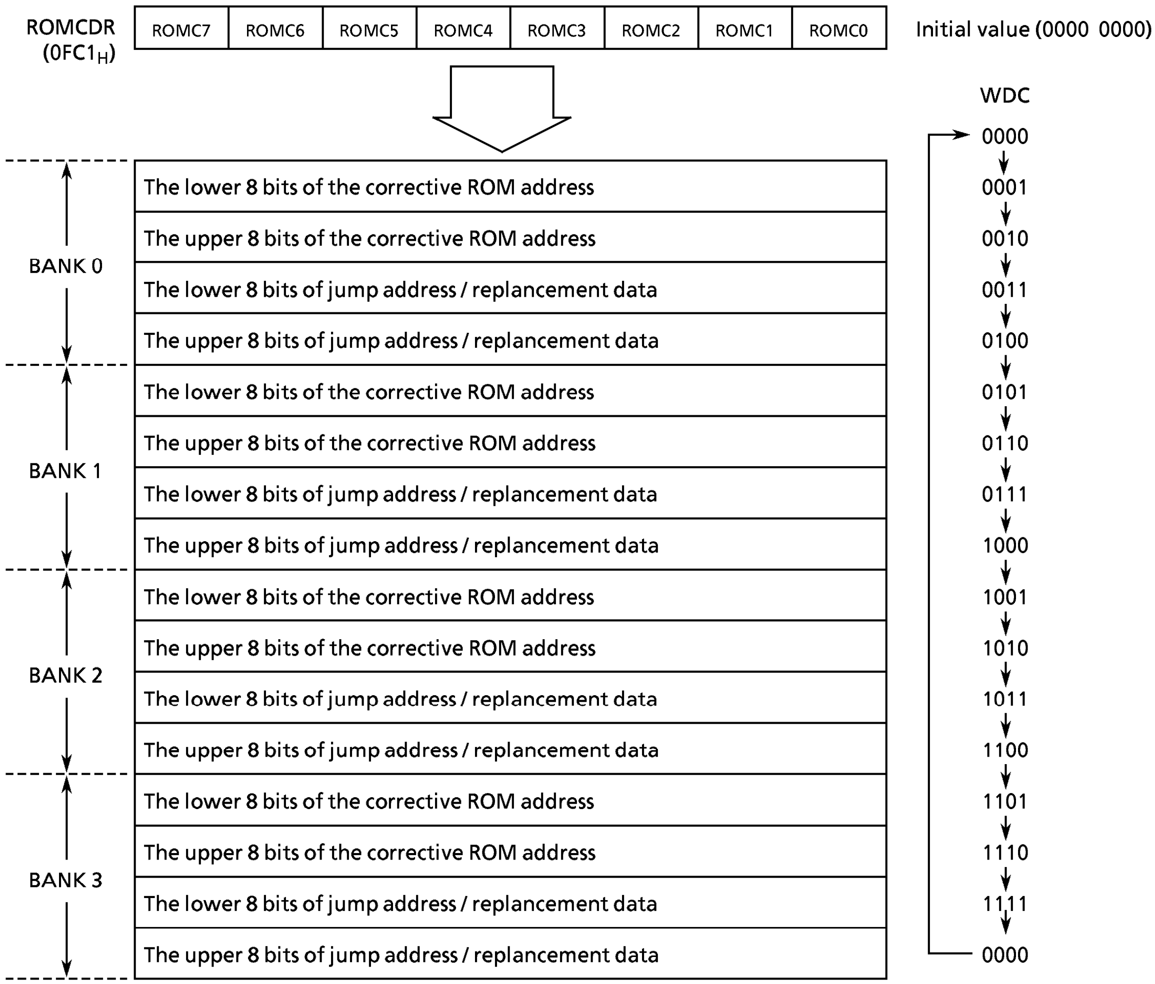
	7	6	5	4	3	2	1	0
ROMCDR ($0FC1_H$)								

(Initial value 0000 0000) write only

Figure 1-34. ROM Corrective Control Register and ROM Corrective Data Register

(1) The ROM corrective data register writing

The ROM corrective data register has four banks corresponding to four independent locations to patch. The write data counter (WDC) points each bank to set. (Figure 1-34.)



Note 1 : WDC value equals to the number of the byte stored in ROMCDR.

Note 2 : ROMCDR is set in order of the lower (8 bits) and the upper(8 bits) start address of the corrective area, the lower (8 bits) and the upper (8 bits) of the jump address / the replacement data.

Figure 1-35. Banks and WDC Value of the Program Corrective Data Register

Whenever ROMCDR is written, WDC is incremented to indicate what data is writen via ROMCDR. During reset, WDC is intialized to "0".

(1) The lower start address of the corrective area (8 bits)
(2) The upper start address of the corrective area (8 bits)
(3) The lower jump address / replacement data (8 bits)
(4) The upper jump address / replacement data (8 bits)

Note : Corrective addresses must have over five addresses each other.

1.12.3 Functions

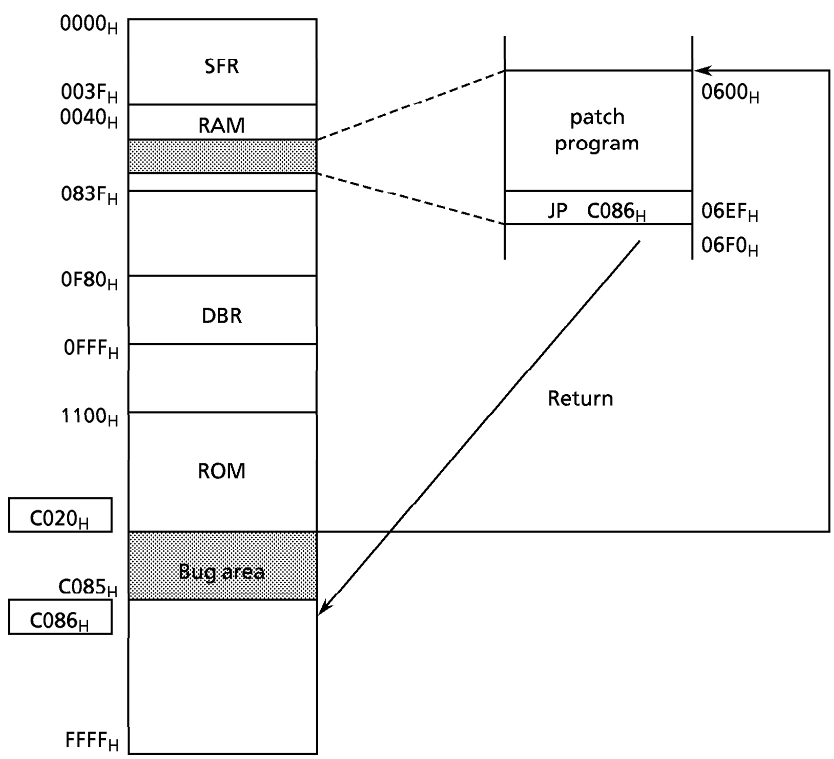
The ROM corrective function can correct maximum four ROM areas with their corresponding four banks of ROM corrective registers. Either program jump mode or data replacement mode is selected for each bank by CM0 - CM3 respectively.

(1) Program jump mode

The program jump mode is to execute the program in the RAM area to correct the bug (s) in the ROM. The start address of ROM that should be patched and the jump vector pointing the RAM area are specified by ROMCDR. When the program is about to run on the code at this start address, the jump instruction is issued, the program branches into the RAM at the jump vector, and the subsequent program codes primarily loaded into this RAM area are excuted. After this patch program execution, the program must be returned to the ROM area by any of the jump instructions at the end of this RAM area. By doing these, the correction of the bug is completed. The program jump mode can be selected at CMn = 0 (n = 0 - 3 for each bank) . The start address must point the 1 st byte of the instruction codes (Op-Code) .

Example : There is bugs on the locations from $C020_H$ to $C085_H$

The corrective address, the jump vector, the program patch codes and other information to patch the ROM with the bugs must be read out from any of memory storage that holds them during initial program routine. CMn = 0 specifies the program jump mode. Subsequently, the patch program codes are loaded into RAM (0600_H to $06EF_H$) . The start address ($C020_H$) of the ROM necessary to patch is written to the corrective ROM address registers, and the start address (0600_H) of the RAM area to patch is loaded onto the jump address registers. When the instruction at $C020_H$ is fetched, the instruction to jump into 0600_H is unconditionally executed instead of the instruction at $C020_H$, and the subsequent patch program codes are executed. The jump instruction at the end of the patch program codes returns to the ROM at $C086_H$.



Note : Corrective address must be assigned to 1st byte of instruction codes on the program jump mode.

(2) Data replacement mode

The data replacement mode is to directly replace a single byte or word (2 byte) length data with the replacement data which are written via ROMCDR.

The program jump mode can work as the equivalent data replacement mode. However, when many instructions refer a certain data in the ROM which must be patched, the program jump mode consumes the same number of banks as that of the instructions referring this (these) data. ROM data replace mode reduces this kind of bank consumption.

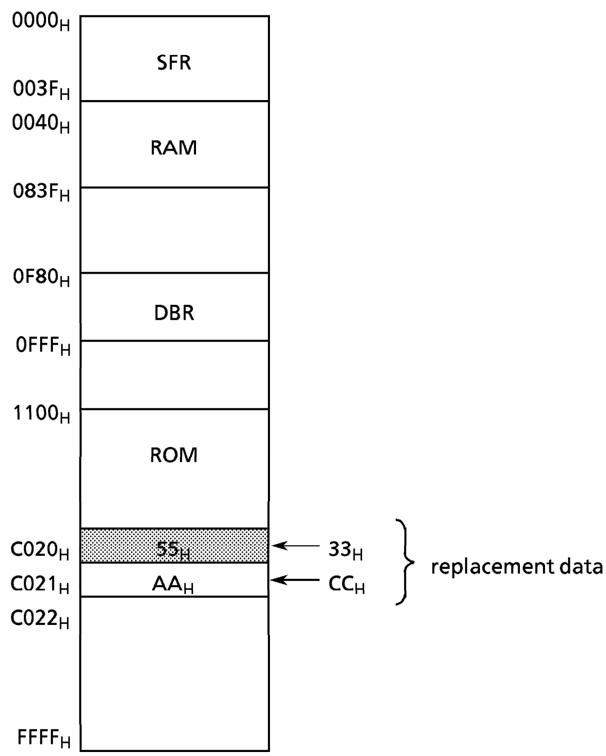
Note : The instruction that gains access to an only byte is replaced to an only start byte.

By setting CMn to 1, the data replacement mode is selected. The start address of ROM data is set to the corrective ROM address, and two bytes replacement data is set to the patch data register via ROMCDR.

The corrective address must point the constant data in the data replacement mode. It is impossible to replace opecode and operand in the data replacement mode.

Example :

The start address is set to $C020_H$ as the location of the replaced data. Two bytes of the patch data are set 33_H for $C020_H$, CC_H for $C021_H$.



1. At HL = $C020_H$, Executing LD A, (HL) loads 33_H in A. (Data replacement)
2. At HL = $C021_H$, Executing LD A, (HL) loads AA_H in A. (No data replacement)
3. At HL = $C020_H$, Executing LD WA, (HL) loads $CC33_H$ in WA. (Data replacement)

Note1 : Corrective address must be assigned to constant data area on the data replacement mode. (Ope-code and Ope-rand can't be replaced by ROM correction circuit.) *Note2 : Instructions which includes "(HL +)" or "(– HL)" operation can't be replaced by ROM corrective circuit on the data replace ment mode.*

2. ON-CHIP PERIPHERALS FUNCTIONS

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

The TLCS-870 Series uses the memory mapped I/O system, and all peripheral control and data transfers are performed through the special function registers (SFR) and data buffer registers (DBR).

The SFR are mapped to addresses 0000_H to $003F_H$, and the DBR to addresses $0F80_H$ to $0FFF_H$.

Figure 2-1. shows the 87CM39/P39/S39 SFRs and Figure 2-2. shows the 87CM39/P39/S39 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	P2 Port		22	–	I2CAR (I²C-bus address)
03	P3 Port		23	SBISR (SBI status)	SBICR2 (SBI control 2)
04	P4 Port		24	SBICR3 (SBI control 3)	
05	P5 Port		25	PWMSR (PWM status)	PWMCR (PWM control)
06	P6 Port		26	–	PWMDBR (PWM data buffer)
07	P7 Port		27	reserved	
08		P5CR (P5 I/O control)	28	reserved	
09	reserved		29	reserved	
0A	–	P0CR (P0 I/O control)	2A	reserved	
0B	–	P1CR (P1 I/O control)	2B	reserved	
0C	–	P4CR (P4 I/O control)	2C	reserved	
0D	–	P6CR (P6 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	TREG3A (Timer register 3A)		38	SYSCR1 (System control)	
19	TREG3B (Timer register 3B)	–	39	SYSCR2	
1A	–	TC3CR (TC3 control)	3A	EIR_L (Interrupt enable register)	
1B	–	TREG4 (Timer register 4)	3B	EIR_H	
1C	–	TC4CR (TC4 control)	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)

Special Function 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
PWM : Pulse Width Modulation

Figure 2-1. SFR

Address	Read	Write
$0F80_H$	–	OSD control register
≈	–	
94	–	
95	DCTR (OSD display-line counter)	
≈	–	
98	–	
99	–	
9A	ORDON	
9B	–	
0F9C	reserved	reserved
9D	reserved	reserved
AA	reserved	reserved
0FAB	–	JECR (Jitter elimination control register)
≈	reserved	reserved
0FB0	TVSCR (Test video signal output control register)	TVSCR (Test video signal output control register)
≈		
0FC0	ROMCCR (ROM Correction control)	ROMCCR (ROM Correction control)
C1	–	ROMCDR (ROM Correction data)
C2	reserved	reserved
≈	reserved	reserved
0FCF	reserved	reserved
D0	RXCR1 (Remo-con control 1)	RXCR1 (Remo-con control 1)
D1	RXCR2 (Remo-con control 2)	RXCR2 (Remo-con control 2)
D2	RXCTR (Remo-con receive counter)	–
D3	RXDBR (Remo-con receive data buffer)	–
D4	RXSR (Remo-con status)	–
D5	reserved	reserved
≈	reserved	reserved
0FFF	reserved	reserved

Data Buffer Registers

Note 1 : Do not access reserved areas by program.
Note 2 : – : Cannot be accessed.
Note 3 : 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 4 : OSD : On screen display
Remo-con : Remote control signal preprocessor

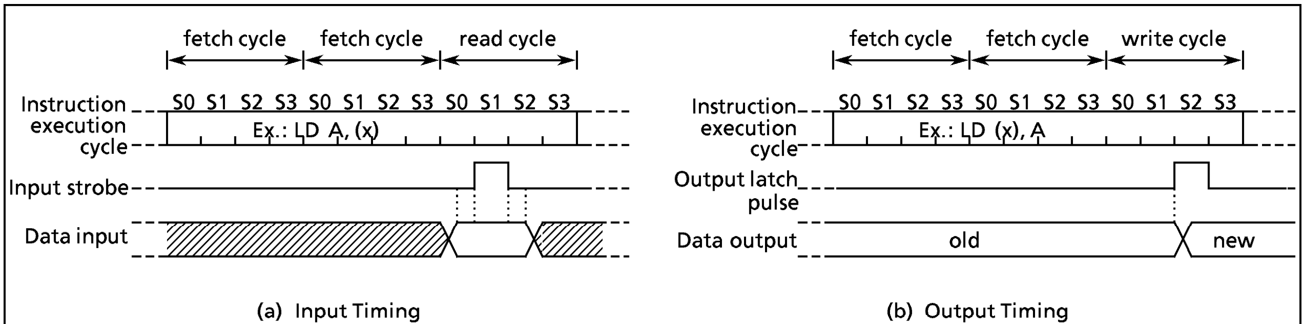
Figure 2-2. DBR

2.2 I/O Ports

The 87CM39/P39/S39 each have 8 parallel input / output ports (55 pins) each as follows:

	Primary Function	Secondary Functions
Port P0	8-bit I/O port	–
Port P1	8-bit I/O port	external interrupt input, timer / counter input / output, and divider output
Port P2	3-bit I/O port	low-frequency resonator connections, external interrupt input, and STOP mode release signal input
Port P3	7-bit I/O port	external interrupt input, remote control signal input, timer / counter input, and serial bus interface input/output
Port P4	8-bit I/O port	pulse width modulation output
Port P5	8-bit I/O port	pulse width modulation output, and analog input
Port P6	8-bit I/O port	R, G, B and Y/BL output from OSD circuitry, analog input, test video signal output.
Port P7	5-bit I/O port	horizontal synchronous pulse input, vertical synchronous pulse input to OSD circuitry, and serial bus interface input / 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 2-3. 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 changes in the S2 state of the write cycle during execution of the instruction which writes to an I/O port.



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

Figure 2-3. Input / Output Timing (Example)

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

(1) Instructions that read the output latch contents

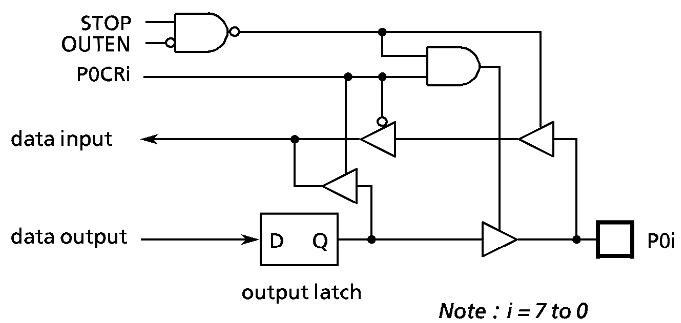
① XCH r, (src)
② CLR/SET/CPL (src).b
③ CLR/SET/CPL (pp).g
④ LD (src).b, CF
⑤ LD (pp) . b, CF
⑥ ADD / ADDC / SUB / SUBB / AND / OR / XOR (src), n
⑦ (src) side of ADD / ADDC / SUB / SUBB / AND / OR / XOR (src), (HL)

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

2.2.1 Port P0 (P07 to P00)

Port P0 is an 8-bit general-purpose input/output port which can be configured as either an input or an output in one-bit unit under software control. Input / output mode is specified by the corresponding bit in the port P0 input / output control register (P0CR). Port P0 is configured as an input if its corresponding P0CR bit is cleared to "0", and as an output if its corresponding P0CR bit is set to "1".
During reset, P0CR is initialized to "0", which configures port P0 as input. The P0 output latches are also initialized to "0". Data is written into the output latch regardless of the P0CR contents. Therefore initial output data should be written into the output latch before setting P0CR.

> *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.*



P0 (0000$_H$) (Initial value : 0000 0000)

7	6	5	4	3	2	1	0
P07	P06	P05	P04	P03	P02	P01	P00

P0CR (000A$_H$) (Initial value : 0000 0000)

7	6	5	4	3	2	1	0

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

Figure 2-4. 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
```

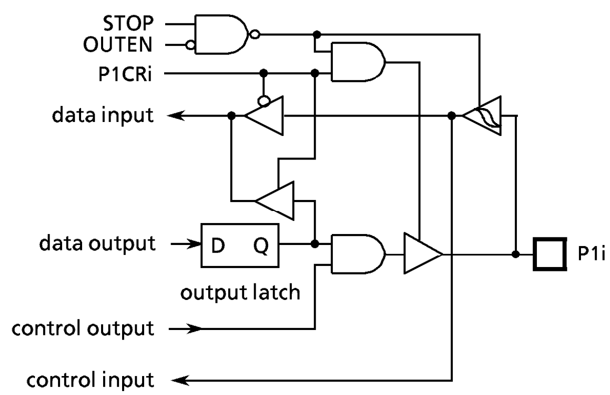
2.2.2 Port P1 (P17 to P10)

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, 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 the P1CR contents. Therefore 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 a 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 on 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.



Note : i = 7 to 0

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 2-5. 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 P2 (P22 to P20)

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

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

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

When a read instruction for port P2 is executed, bits 7 to 3 in P2 are read in as undefined data.

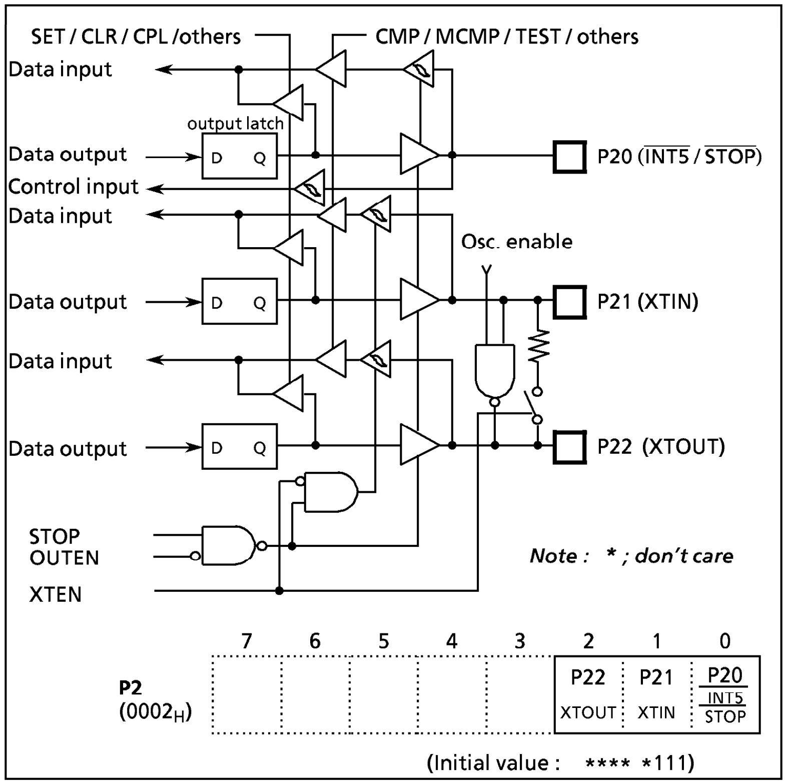


Figure 2-6. Port P2

2.2.4 Port P3 (P36 to P30)

Port P3 is an 7-bit input / output port, and is also used as a serial bus interface input / output, an external interrupt input, a timer / counter input, and Remote-control signal input. 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 lower 4bits (P33 to P30) in port P3.

XOR (P3), 00001111B ;
P33 to P30←$\overline{P33}$ to $\overline{P30}$

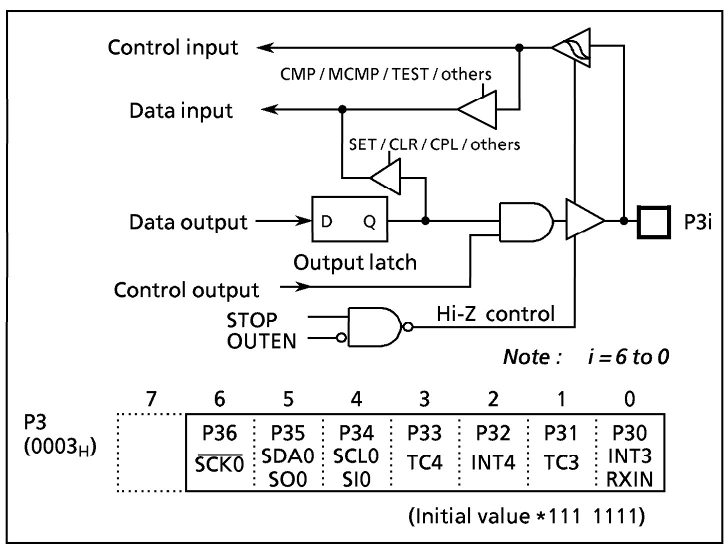


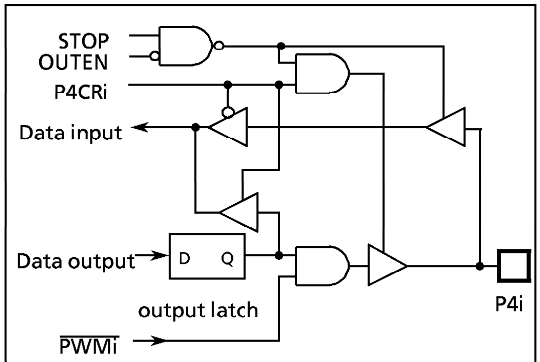
Figure 2-7. Port P3

2.2.5 Port P4 (P47 to P40)

Port P4 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 P4 input/output control register (P4CR). Port P4 is configured as an input if its corresponding P4CR bit is cleared to "0", and as an output if its corresponding P4CR bit is set to "1". During reset, P4CR is initialized to "0", which configures port P4 as an input . The P4 output latches are also initialized to "1".

Data is written into the output latch regardless of the P4CR contents. Therefore initial output data should be written into the output latch before setting P4CR. Port P4 is also used as a pulse width modulation (PWM) output. When used as a PWM output pin, the output pins should be set to the output mode and beforehand the output latch should be set to "1".

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 to 0

P4 (0004_H) (Initial value : 1111 1111)

7	6	5	4	3	2	1	0
P47	P46	P45	P44	P43	P42	P41	P40
$\overline{\text{PWM7}}$	$\overline{\text{PWM6}}$	$\overline{\text{PWM5}}$	$\overline{\text{PWM4}}$	$\overline{\text{PWM3}}$	$\overline{\text{PWM2}}$	$\overline{\text{PWM1}}$	$\overline{\text{PWM0}}$

P4CR ($000C_H$) (Initial value : 0000 0000)

7	6	5	4	3	2	1	0

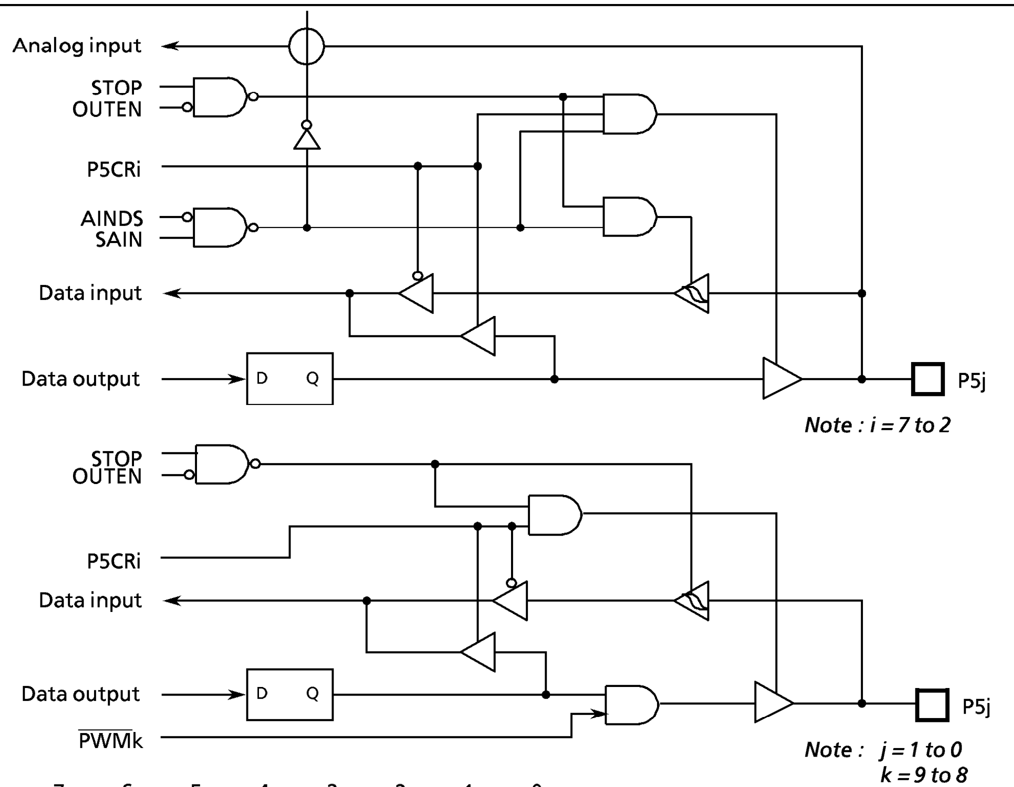
P4CR	I/O control for port P4	0 : input mode 1 : output mode	write only

Figure 2-8. Port P4 and P4CR

2.2.6 Port P5 (P57 to P50)

Port P5 is an 8-bit input / output port which can be configured as an input or output in one-bit unit under software control. Input / output mode is specified by the corresponding bit in the port P5 input / output control register (P5CR). For example, port P5 is configured as an input if its corresponding P5CR bit is cleared to "0", and as an output if its corresponding bit is set to "1". During reset, P5CR is initialized to "0", which configures port P5 as an input and AINDS (bit 4 in the ADCCR) and SAIN (bit 3 to 0 in the ADCCR) are initialized to "0", which configure port P52 as an analog input. The P5 output latches are also initialized to "1". Data is written into the output latch regardless of the P5CR contents. Therefore initial output data should be written into the output latch before setting P5CR.

Port P5 is also used as an analog input for the A/D converter and a pulse width modulation (PWM). When used as an analog input, AINDS must be cleared to "0". The bit used as an analog input must be selected by SAIN and be specified as input port by P5CR. Unused pin as analog input can be used as input / output port. But it is recommendable that the contents of output latches for output pins in P5 port should not be changed during A/D conversion, because an accuracy of A/D conversion is changed for the worse. When used as a PWM output, output pins should be set to the output mode and beforehand the output latch should be set to "1". The P5 output latches are initialized to "1". Data is written into the output latch regardless of the P5CR contents. Therefore initial output data should be written into the output latch before setting P5CR.



P5 (0005_H)

7	6	5	4	3	2	1	0
P57 AIN5	P56 AIN4	P55 AIN3	P54 AIN2	P53 AIN1	P52 AIN0	P51 $\overline{PWM9}$	P50 $\overline{PWM8}$

(Initial value : 1111 1111)

P5CR (0008_H)

7	6	5	4	3	2	1	0

(Initial value : 0000 0000)

P5CR	I/O control for port P6	0 : Input mode 1 : Output mode	write only

Note : P5CR is a write-only register and must not be used with any of the read-modify-write instruction.

Figure 2-9. Port P5

2.2.7 Port P6 (P67 to P60)

Port P6 is an 8-bit input / output port which can be configured as an input or an output in one-bit unit under software control. Input or output mode is selected by the corresponding bit in the input/output control register (P6CR). For example, port P6 is configured as an input if its corresponding P6CR bit is cleared to "0", and as an output if its corresponding bit is set to "1". During reset, P6CR is initialized to "0", which configures port P6 as an input. The P6 output latches are also initialized to "1".

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. Pins P63 to P60 are available high current output, so LEDs can be driven directly.

Port P6 is also used as an on screen display (OSD) output (R, G, B, and Y/BL signal), an analog input for A/D converter and a test video signal output.

When used as an OSD output pin, the OSD output pins should be set to the output mode and beforehand the port P6 data selection register (P67DS to P64DS) should be set to "1".

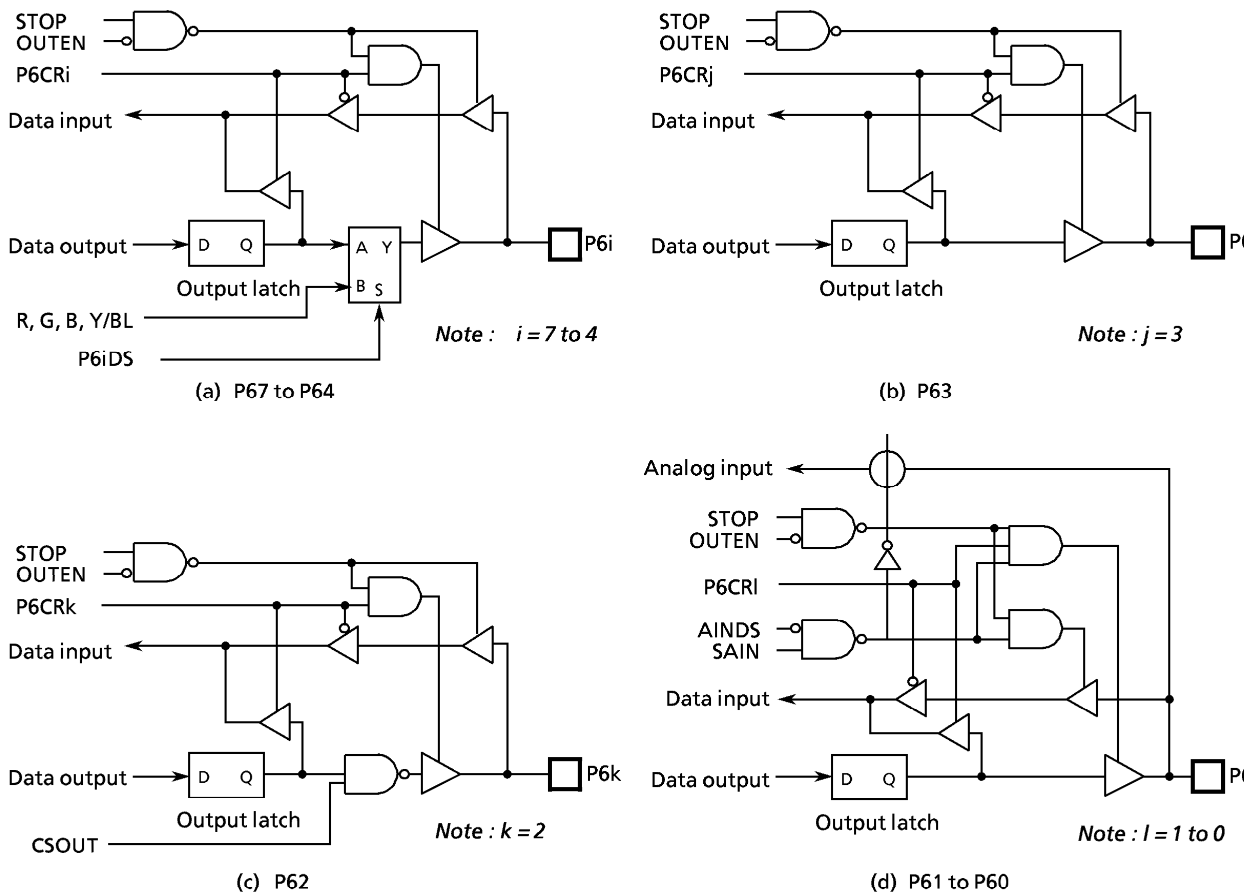
When used as an analog input, AINDS (bit 4 in the ADCCR) must be cleared to "0". The bit used as an analog input must be selected by SAIN (bit 3 to 0 in the ADCCR) and be specified as input port by P6CR.

Unused pin as analog input can be used as input / output port. But it is recommendable that the contents of output latches for output pins in P6 port should not be changed during A/D conversion, because an accuracy of A/D conversion is changed for the worse. When used as a test video signal output, the output pins should be set to the output mode and beforehand the output latch should be set to "1".

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.

Example : Set the lower 4 bit in port P6 (P63 to P60) to the output port and set the other to the input port.

```
LD      (P6CR), 0FH      ; P6CR←0000 1111B
```



P6 (0006_H)

7	6	5	4	3	2	1	0
P67 Y BL	P66 B	P65 G	P64 R	P63	P62 CSOUT	P61 AIN7	P60 AIN6

(Initial value 1111 1111)

P6CR ($000D_H$)

7	6	5	4	3	2	1	0

(Initial value 0000 0000)

P6CR	I/O control for port P6	0 : Input mode 1 : Output mode	write only

Port P6 data selection register (P67DS to P64DS)

($0F91_H$)

7	6	5	4	3	2	1	0
P67DS	P66DS	P65DS	P64DS	(EFR12)	(EFR11)	(EFR10)	(EFR9)

(Initial value 0000 0000)

P6nDS	Selection of the output data for port P6n	0 :Port P6n output latch 1 :the OSD output (R, G, B, Y/BL)	write only

Note1 : P6CR, P67DS to P64DS is write-only-register and must not be used with any of read-modify-write instructions.

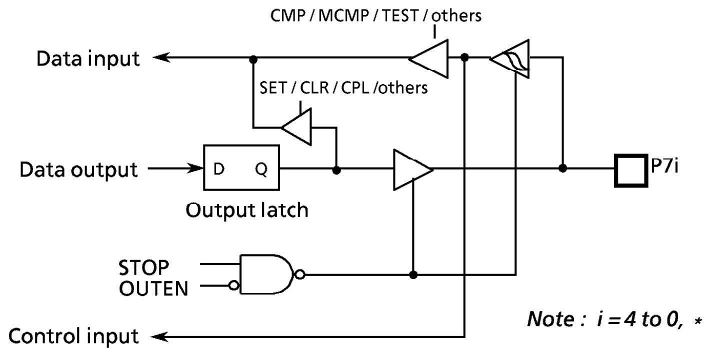
Figure 2-10. Port P6, P6CR, and P67DS to P64DS

2.2.8 Port P7 (P74 to P70)

Port P7 is a 5-bit input / output port, and is also used as a vertical synchronous signal ($\overline{VD}$) input, a horizontal synchronous signal ($\overline{HD}$) input for the on screen display (OSD) circuitry and serial bus interface input / output.

The output latches are initialized to "1" during reset. When used as an input port or a secondary function pin, the output latch should be set to "1".

When a read instruction for port P7 is executed, bits 7 to 5 in P7 are read in as undefined data.



*Note : i = 4 to 0, * ; don't care*

	7	6	5	4	3	2	1	0	
P7 (0007_H)				P74 / $\overline{SCK1}$	P73 / SDA1 / SO1	P72 / SCL1 / SI1	P71 / $\overline{VD}$	P70 / $\overline{HD}$	(Initial value : ***1 1111)

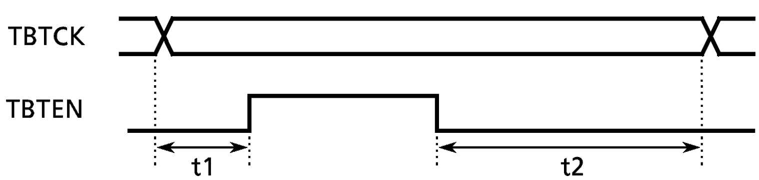
Figure 2-11. Port P7

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 2-13.

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 ≧ 0

t2 ≧ 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
```

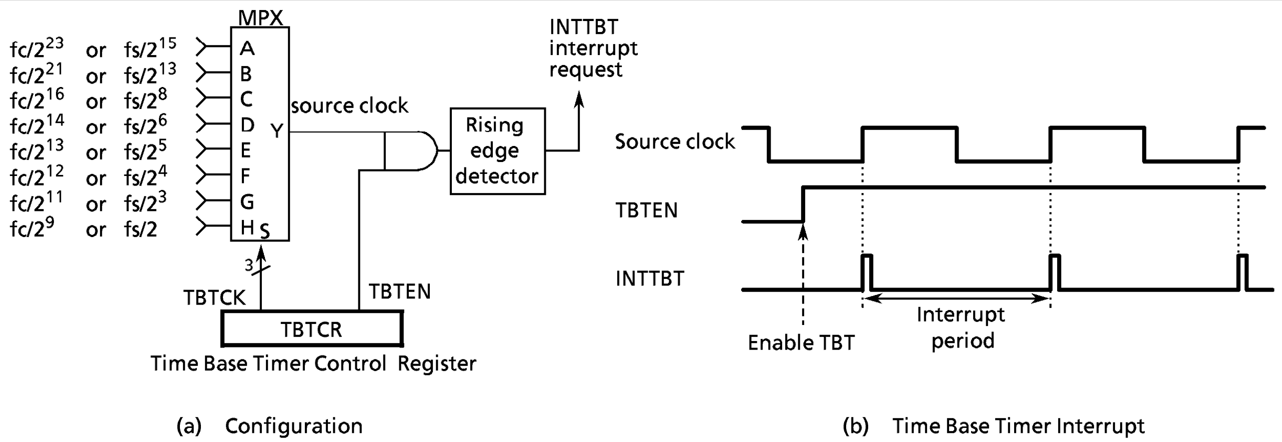


Figure 2-12. Time Base Timer

TBTCR (0036_H)

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

(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 : $fc/2^{23}$ or $fs/2^{15}$ [Hz] 001 : $fc/2^{21}$ or $fs/2^{13}$ 010 : $fc/2^{16}$ or $fs/2^{8}$ 011 : $fc/2^{14}$ or $fs/2^{6}$ 100 : $fc/2^{13}$ or $fs/2^{5}$ 101 : $fc/2^{12}$ or $fs/2^{4}$ 110 : $fc/2^{11}$ or $fs/2^{3}$ 111 : $fc/2^{9}$ or fs/2	

*Note1 : fc ; High-frequency clock [Hz], fs ; Low-frequency clock [Hz], * ; don't care*
Note2 : TBTCR is a write-only register and must not be used with any of read-modify-write instructions.

Figure 2-13. Time Base Timer and Divider Output Control Register

Table 2-1. Time Base Timer Interrupt Frequency

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

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

A 50 % duty pulse can be output using the divider output circuit, which is useful for a piezo-electric buzzer drive. Divider output is from pin P13 ($\overline{DVO}$). The P13 output latch should be set to "1" and then the P13 should be configured as an output.
The divider output circuit is controlled by the control register (TBTCR) shown in Figure 2-14.
Note that TBTCR is a write-only register.

	7	6	5	4	3	2	1	0
TBTCR (0036_H)	DVOEN	DVOCK		(DV7CK)	(TBTEN)	(TBTCK)		

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

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

*Note 1 : fc ; High-frequency clock [Hz], fs ; Low-frequency clock [Hz], * ; don't care*
Note 2 : TBTCR is write-only register and must not be used with any of the read-modify-write instructions such as SET, CLR, etc.

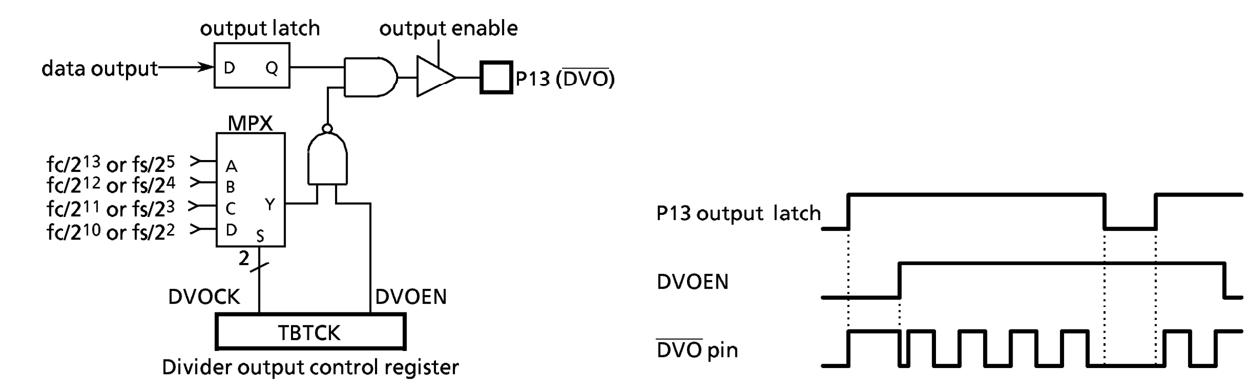
Figure 2-14. Divider Output Control Register

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

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

Table 2-2. Frequency of Divider Output

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



(a) Configuration (b) Timing Chart

Figure 2-15. Divider Output

2.5 16-bit Timer / Counter 1 (TC1)

2.5.1 Configuration

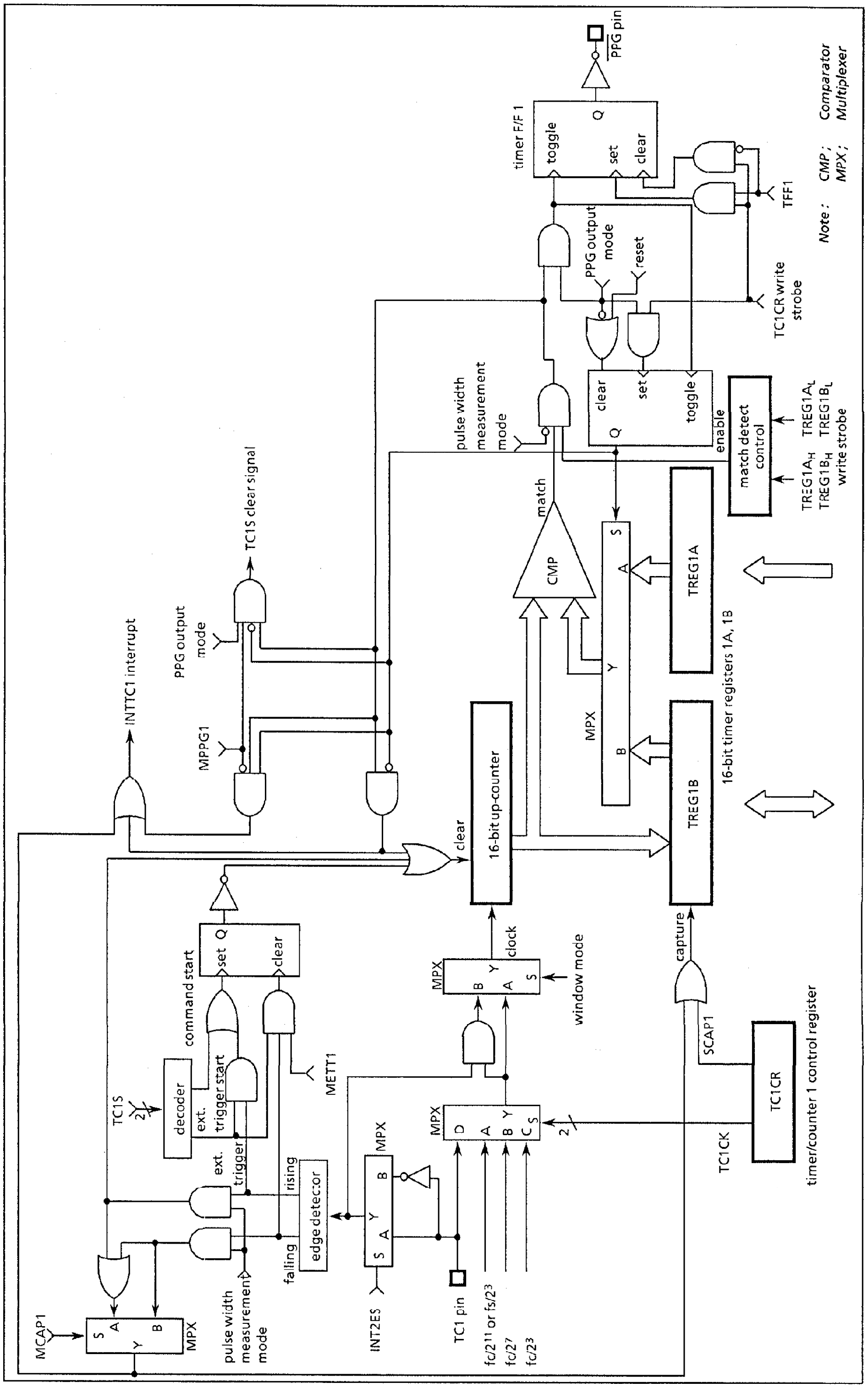


Figure 2-16. Timer / Counter 1

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.

	15–8	7–0	
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)	Read / Write (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 $fc/2^{11}$ or $fs/2^3$ [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 srart 1 : trigger start & stop	
MPPG1	PPG output control	0 : cotinuous pulse 1 : single pulse	
TFF1	timer F/F1 control for PPG output mode	0 : clear 1 : set	

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

Note 2 : Writing to the low-byte of the timer registers ($TREG1A_L$, $TREG1B_L$), the comparison is inhibited until the high-byte ($TREG1A_H$, $TREG1B_H$) is written.

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

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

Note 5 : Values to be loaded to timer registers must satisfy the following condition.

TREG1A > TREG1B > 0 (PPG output mode) ; TREG1A > 0 (others)

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

Note 7 : TC1CR is a write-only register, which cannot access any of in read-modify-write instruction such as bit operate, etc.

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

Figure 2-17. 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 counter is cleared. The current contents of up-counter can be transferred to TREG1B by setting SCAP1 (bit 6 in TC1CR) to "1" (software capture function). SCAP1 is automatically cleared after capturing.

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

Source clock			Resolution		Maximum time setting	
NORMAL1/2, IDLE1/2 modes		SLOW, SLEEP modes	At fc = 8 MHz	At fs = 32.768 kHz	At fc = 8 MHz	At fs = 32.768 kHz
DV7CK = 0	DV7CK = 1					
$fc/2^3$ [Hz]	$fc/2^3$ [Hz]	–	1 μs	–	65.5 ms	–
$fc/2^7$	$fc/2^7$	–	16 μs	–	1.0 s	–
$fc/2^{11}$	$fs/2^3$	$fs/2^3$ [Hz]	256 μs	244.14 μs	16.7 s	16.0 s

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

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

Example 2 : Software capture

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

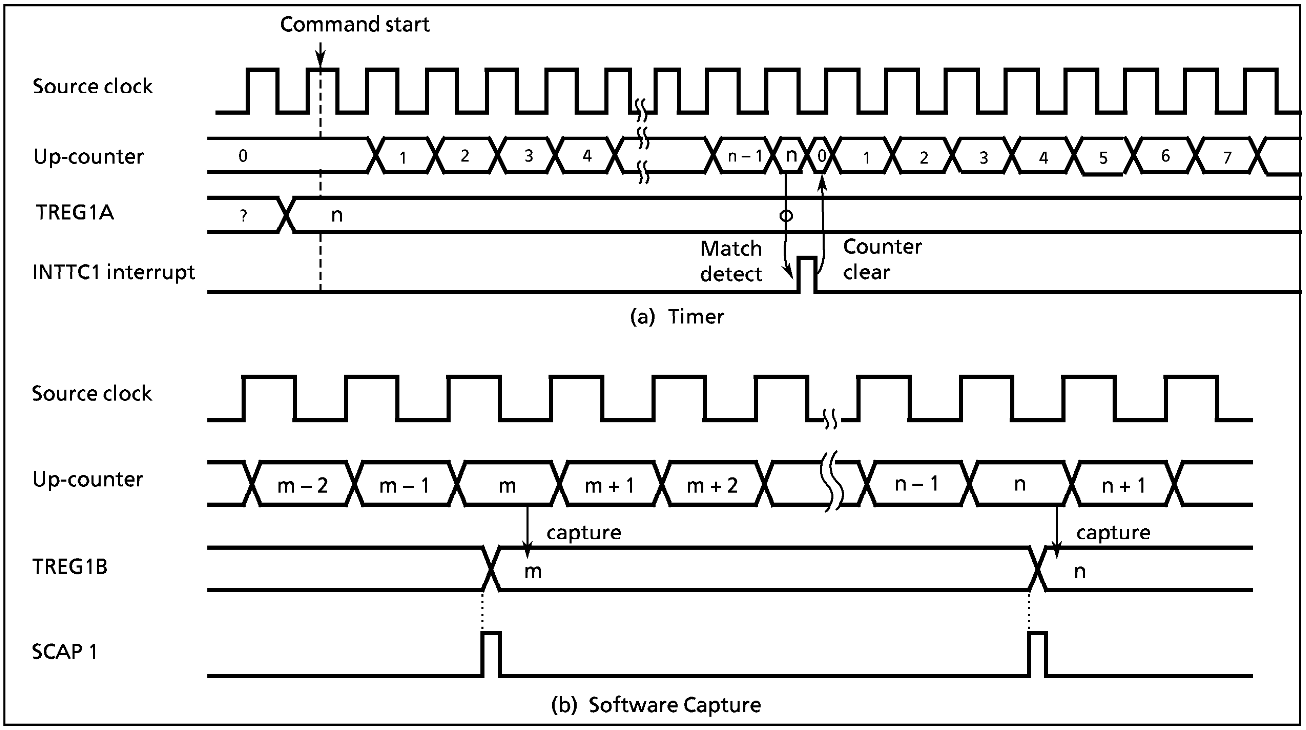


Figure 2-18. Timer Mode Timing Chart

(2) **External Trigger Timer** mode

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

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

Example 1 : Detects rising edge (in TC1 pin input) and generates an interrupt 100 μs later. (at fc = 8 MHz)

```
LD    (EINTCR), 00000000B ; INT2ES←0 (rising edge)
LDW   (TREG1A), 0064H     ; 100 μs ÷ 2^3/fc = 64H
SET   (EIRL). EF4         ; Enables INTTC1
EI
LD    (TC1CR), 00111000B  ; Starts TC1 with an external trigger, METT = 0
```

Example 2 : Generates an interrupt, inputting "L" level pulse (pulse width: 4 [ms] or more) to the TC1 pin. (at fc = 8 MHz)

```
LD    (EINTCR), 00000100B ; INT2ES←1 ("L" level)
LDW   (TREG1A), 00FAH     ; 4 ms ÷ 2^7/fc = FAH
SET   (EIRL). EF4         ; Enables INTTC1
EI
LD    (TC1CR), 01110100B  ; Starts TC1 with an external trigger, METT = 1
```

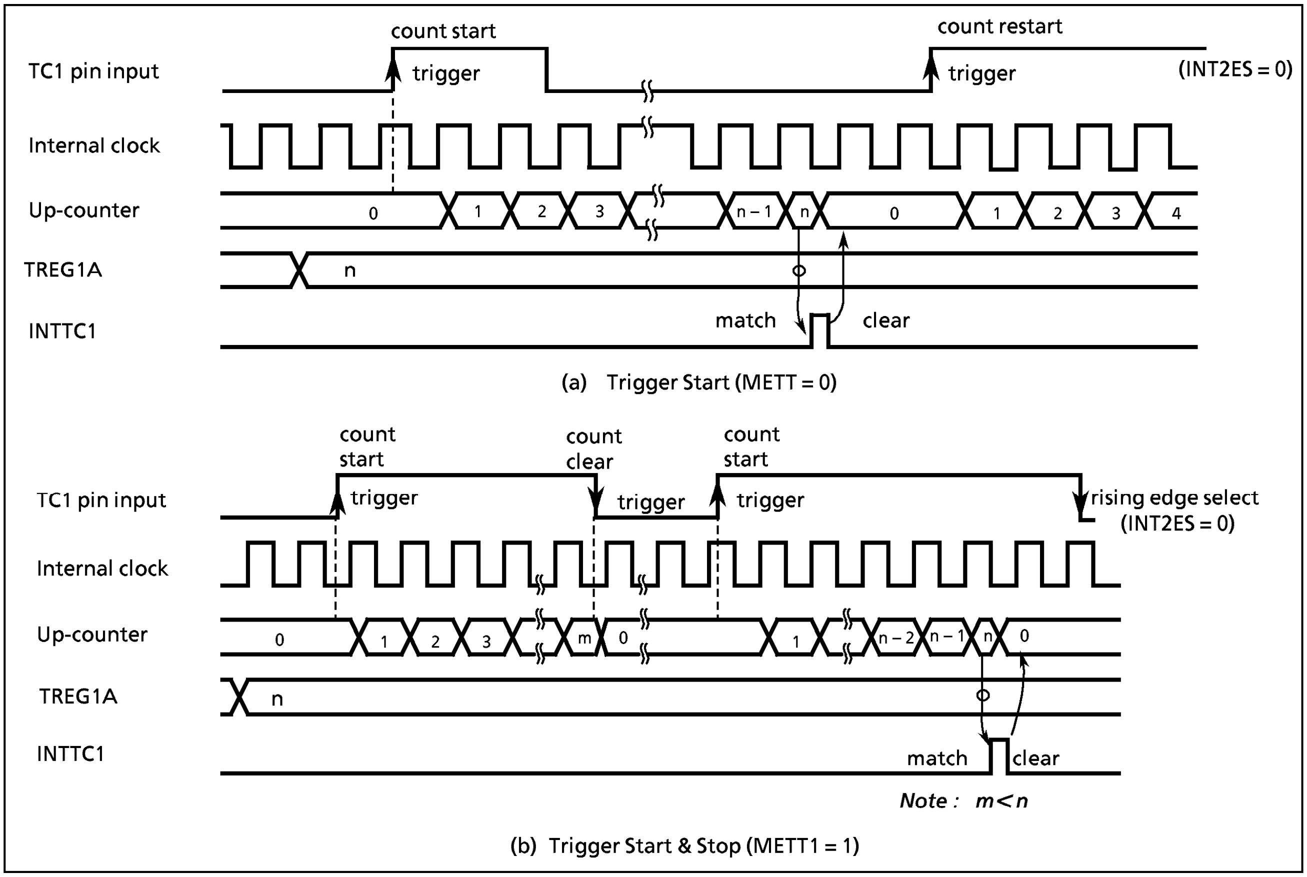


Figure 2-19. External Trigger Timer Mode Timing Chart

(3) **Event Counter** Mode

In this mode, events are counted at the edge of the TC1 pin input. Either the rising or falling edge can be selected with INT2ES in EINTCR. The contents of TREG1A are compared with the contents of up-counter. If a match is found, an INTTC1 interrupt is generated, and the counter is cleared. The maximum applied frequency is fc/2^4 [Hz] in NORMAL1/2 or IDLE1/2 mode and fs/2^4 [Hz] in SLOW or SLEEP mode.
Setting SCAP1 to "1" transferes the current contents of up-counter to TREG1B (software capture function). SCAP1 is automatically cleared after capturing.

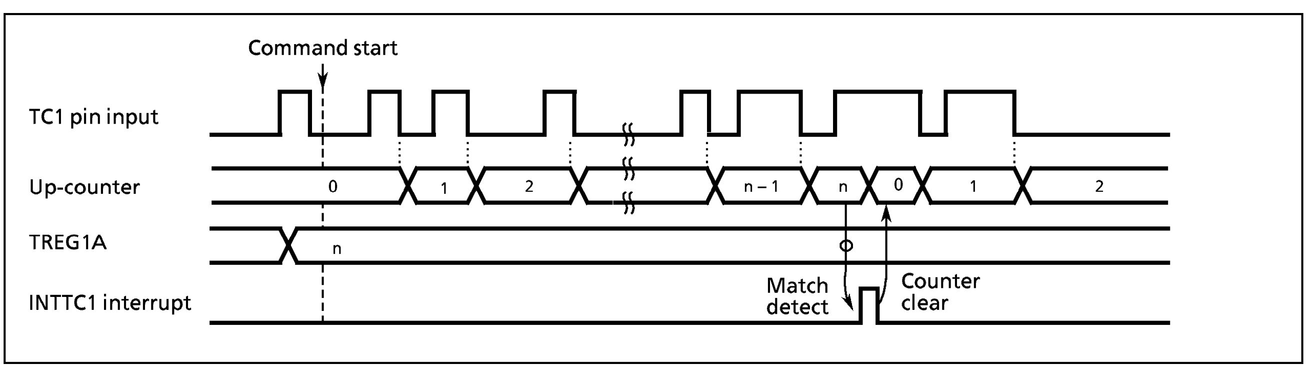


Figure 2-20. Event Counter Mode Timing Chart (INT2ES = 1)

(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" transferres the current contents of up-counter to TREG1B. It is necessary that the maximum applied frequency (TC1 input) be such that the counter value can be analyzed by the program. That is, the frequency must be considerably slower than the selected internal clock.

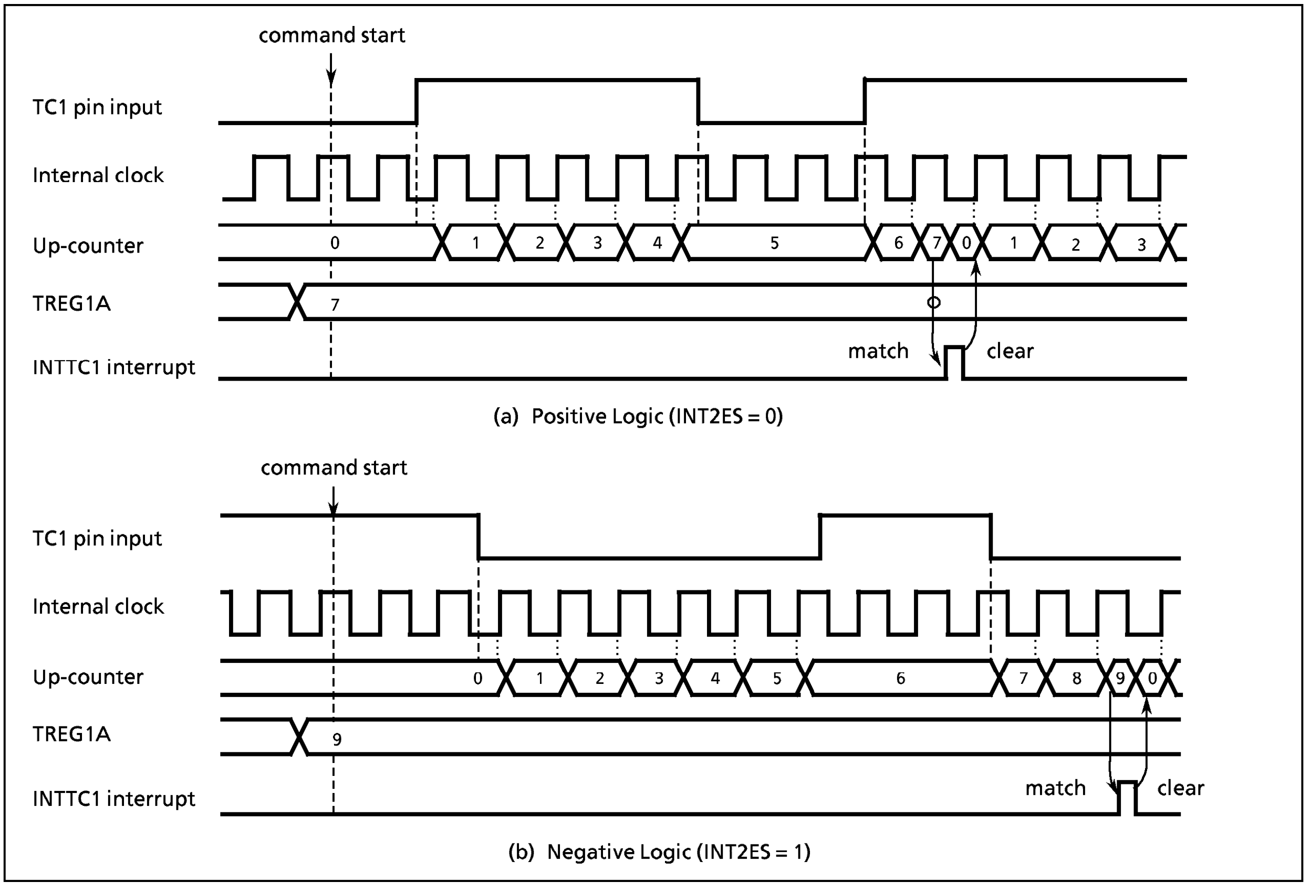
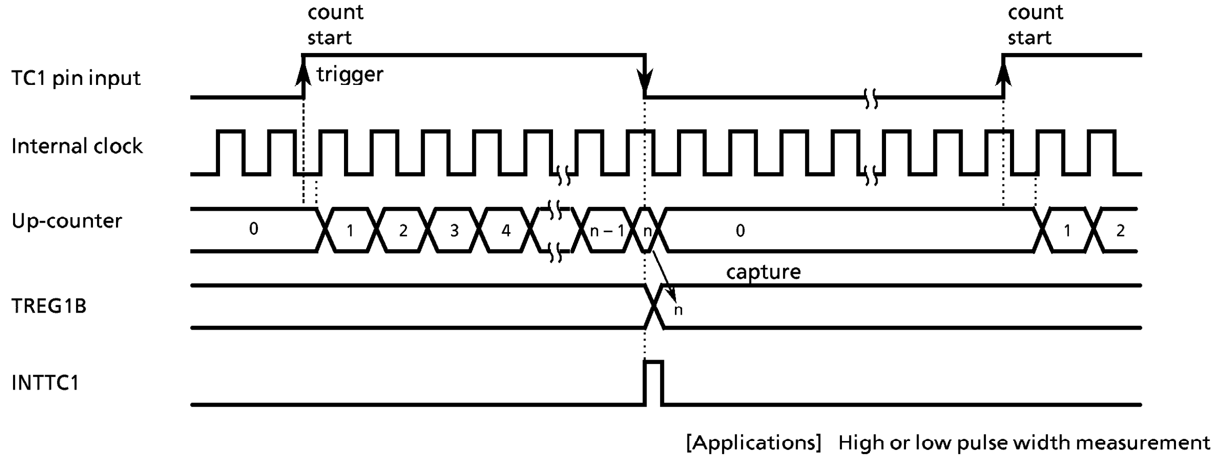


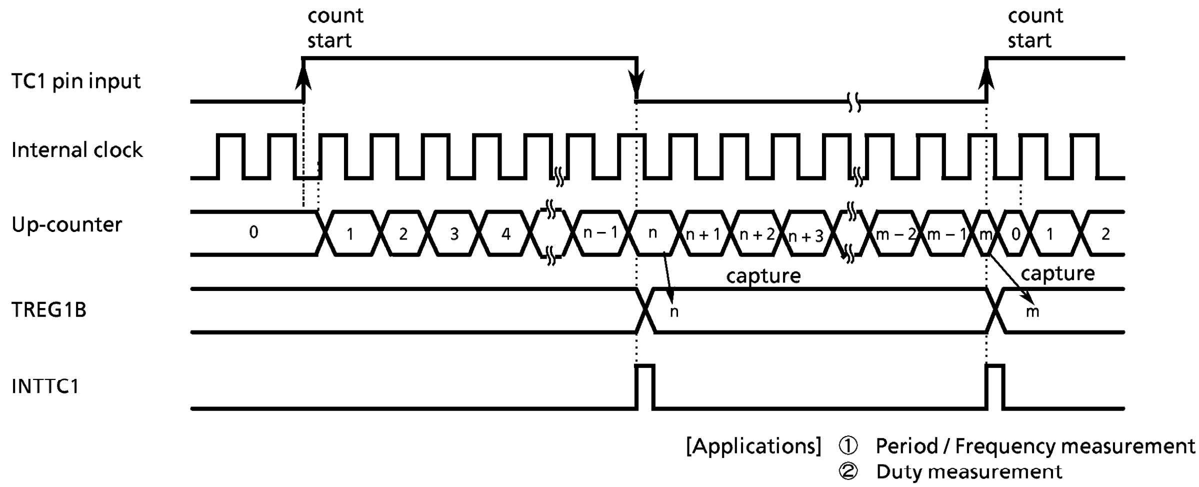
Figure 2-21. Window Mode Timing Chart

(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).



(a) Single Edge Capture



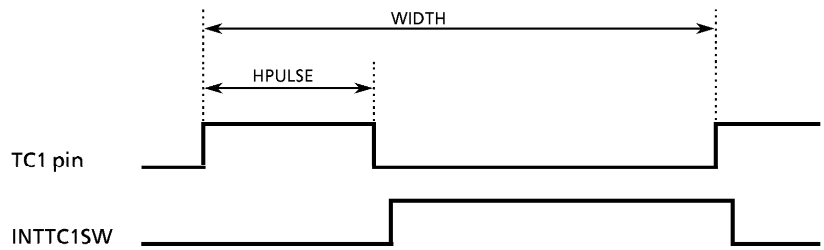
(b) Double Edge Capture

Figure 2-22. Pulse Width Measurement Mode Timing Chart

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

```
           CLR   (INTTC1SW). 0          ;  INTTC1 service switch initial setting
           LD    (EINTCR), 00000000B    ;  Sets the rise edge at the INT2 edge
           LD    (TC1CR), 00000110B     ;  Sets the TC1 mode and source clock
           SET   (EIRL). EF4            ;  Enables INTTC1
           EI
           LD    (TC1CR), 00110110B     ;  Starts TC1 with an external trigger
            ⋮
PINTTC1 :  CPL   (INTTC1SW). 0          ;  Complements INTTC1 service switch
           JRS   F, SINTTC1
           LD    (HPULSE), (TREG1BL)    ;  Reads TREG1B
           LD    (HPULSE + 1), (TREG1BH)
           RETI
```

```
SINTTC1 :  LD    (WIDTH), (TREG1BL)       ;   Reads TREG1B (Period)
           LD    (WIDTH + 1), (TREG1BH)
            ⋮
           RETI
            ⋮
VINTTC1 :  DW    PINTTC1
```



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

Counting is started by an edge of the TC1 pin input (either the rising or falling edge can be selected) or by a command. The source clock is used an internal clock. First, the contents of TREG1B are compared with the contents of the up-counter. If a match is found, timer F/F1 output is toggled. Next, timer F/F1 is again toggled and the counter is cleared by matching with TREG1A. An INTTC1 interrupt is generated at this time. Timer F/F output is connected to the P14 ($\overline{\text{PPG}}$) 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.

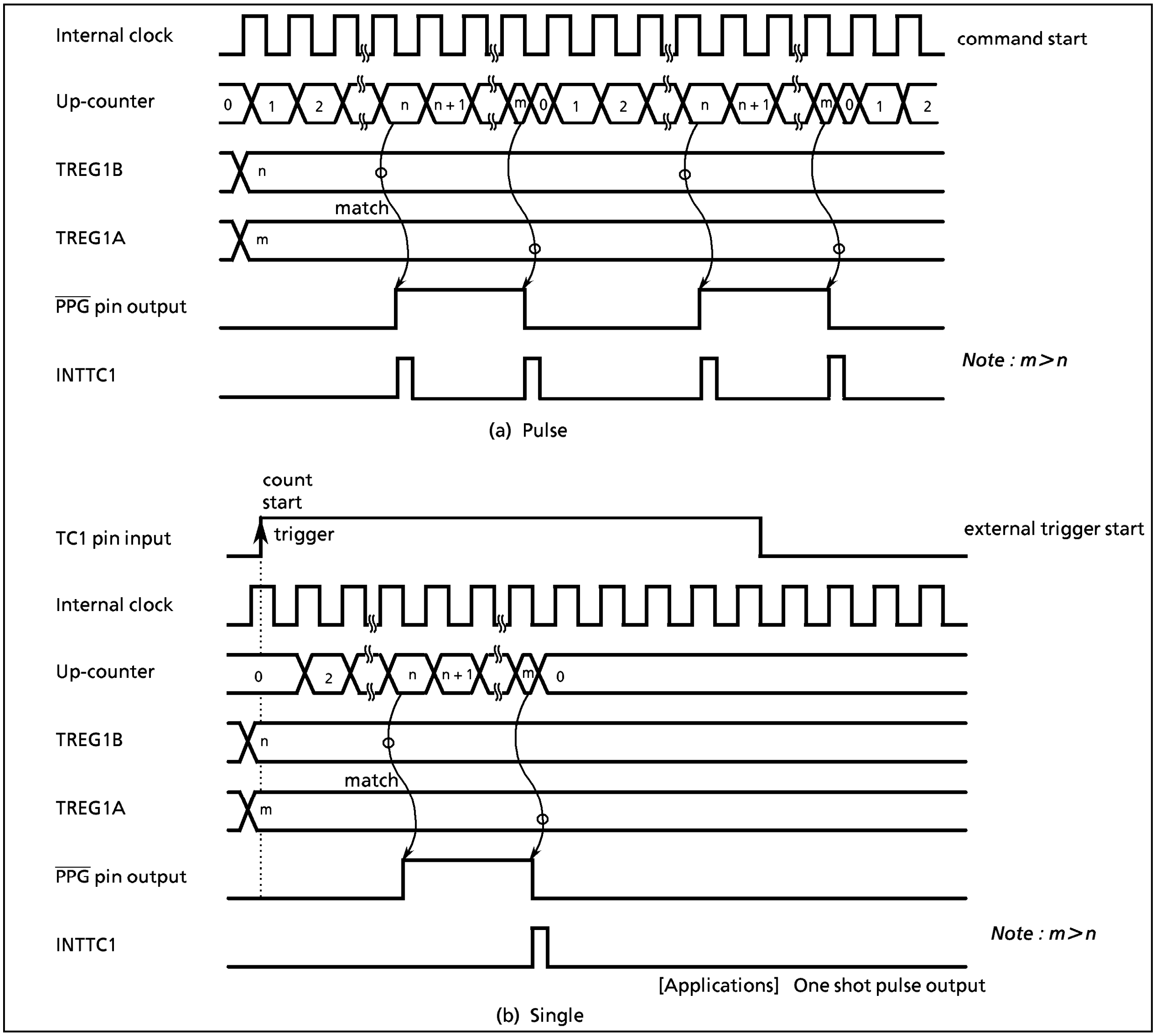


Figure 2-23. PPG Output Mode Timing Chart

Example : Pulse output ("H" level ← 800 µs, "L" level ← 200 µs) (at fc = 8 MHz)

```
SET   (P1). 4               ; P14 output latch ← 1
LD    (P1CR), 0001000B      ; Sets the P14 output mode
LD    (TC1CR), 10000011B    ; Sets the PPG output mode TFF = 1
LDW   (TREG1A), 03E8H       ; Sets the period (1 ms ÷ 1 µs = 03E8H)
LDW   (TREG1B), 00C8H       ; Sets the "L" level pulse width (200 µs ÷ 1 µs = 00C8H)
LD    (TC1CR), 10010011B    ; Command start
```

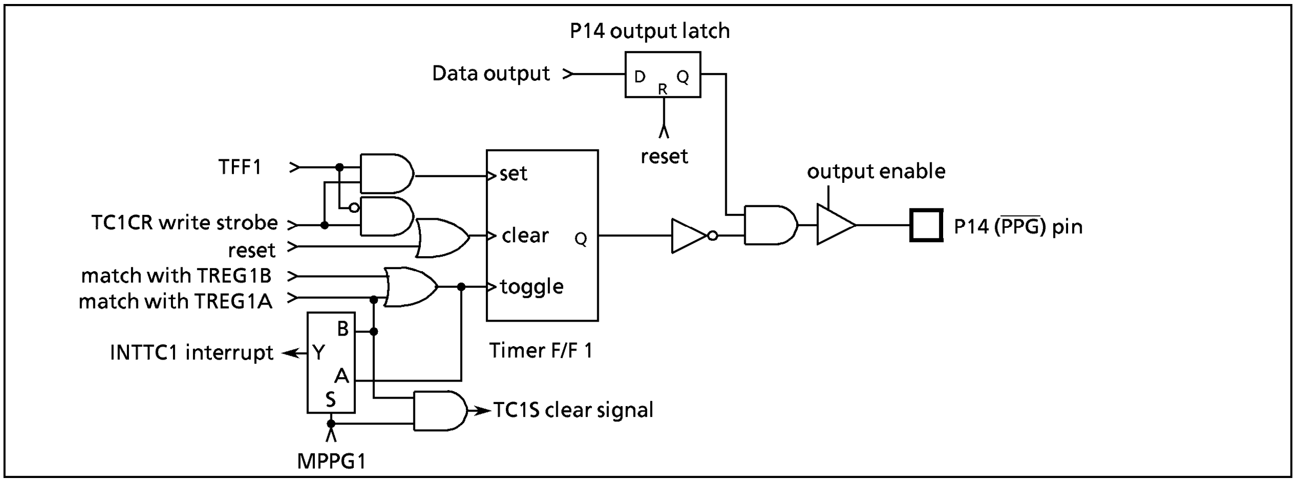


Figure 2-24. PPG Output

2.6 16-bit Timer / Counter 2 (TC2)

2.6.1 Configuration

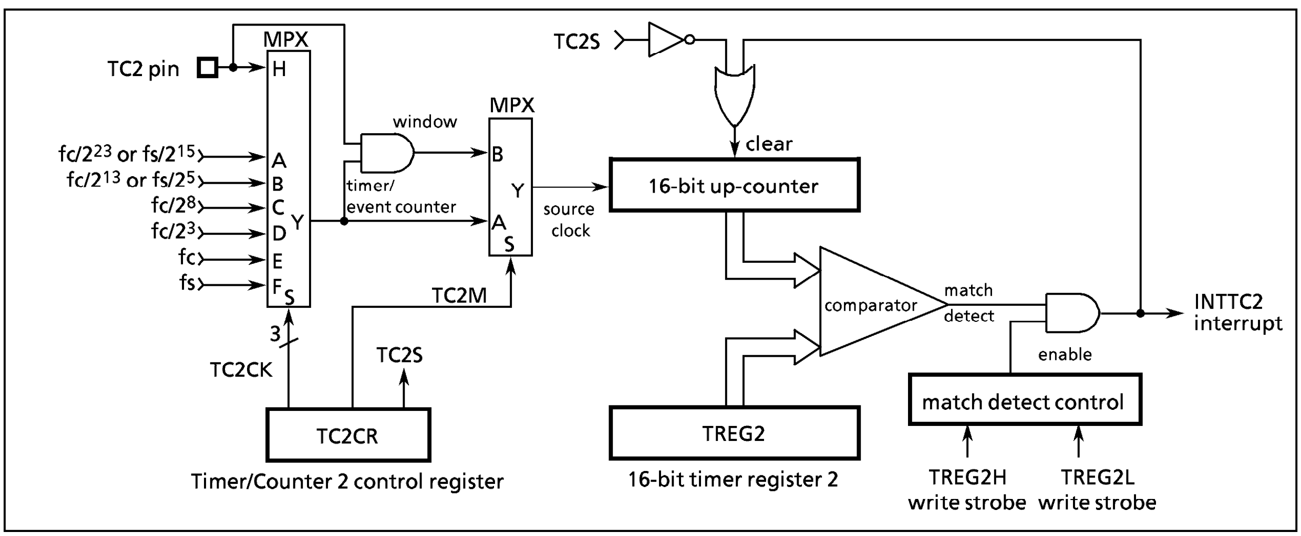


Figure 2-25. Timer / Counter 2 (TC2)

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 the TREG2.

TREG2 (0016, 0017_H)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$TREG2_H$ (0017_H)								$TREG2_L$ (0016_H)							

write only

TC2CR (0015_H)

7	6	5	4	3	2	1	0
		TC2S	TC2CK				TC2M

(Initial value : **00 00*0)

TC2M	Timer / counter 2 operating mode select	0 : Timer / Event counter mode 1 : Window mode	write only
TC2CK	Timer / counter 2 source clock select	000 : Internal clock $fc/2^{23}$ or $fs/2^{15}$ [Hz] 001 : 〃 $fc/2^{13}$ or $fs/2^{5}$ 010 : 〃 $fc/2^{8}$ 011 : 〃 $fc/2^{3}$ 100 : 〃 fc (Note 5) 101 : 〃 fs 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], fs; Low-frequency clock [Hz], *; don't care*

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

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

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

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

Note 6 : TC2CR and TREG2 are write-only registers and must not be used with any of the read-modify-write instructions such as SET, CLR, etc.

Figure 2-26. Timer Register 2 and TC2 Control Register

2.6.3 Function

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

(1) **Timer** Mode

In this mode, the internal clock is used for counting up. The contents of TREG2 are compared with the contents of up-counter. If a match is found, a timer / counter 2 interrupt (INTTC2) is generated, and the counter is cleared. Counting up is resumed after the counter is cleared.
Also, when "fc" is selected as the source clock during SLOW mode, the lower 11 bits of TREG2 are ignored and an INTTC2 interrupt is generated by matching the upper 5 bits. Thus, in this case, only the $TREG2_H$ setting is necessary.

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

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

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

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

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

(2) **Event Counter** Mode

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

Example : Sets the event counter mode and generates an INTTC2 interrupt 640 counts later.

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

(3) **Window** Mode

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

Example : Generates an interrupt, inputting "H" level pulse width of 120 [ms] or more. (at fc = 8 MHz)

```
LDW   (TREG2), 0078H      ; Sets the TREG2 (120 ms ÷ 2^13/fc = 0078H)
SET   (EIRH). EF14        ; Enables INTTC2
EI
LD    (TC2CR), 00100101B  ; Starts TC2
```

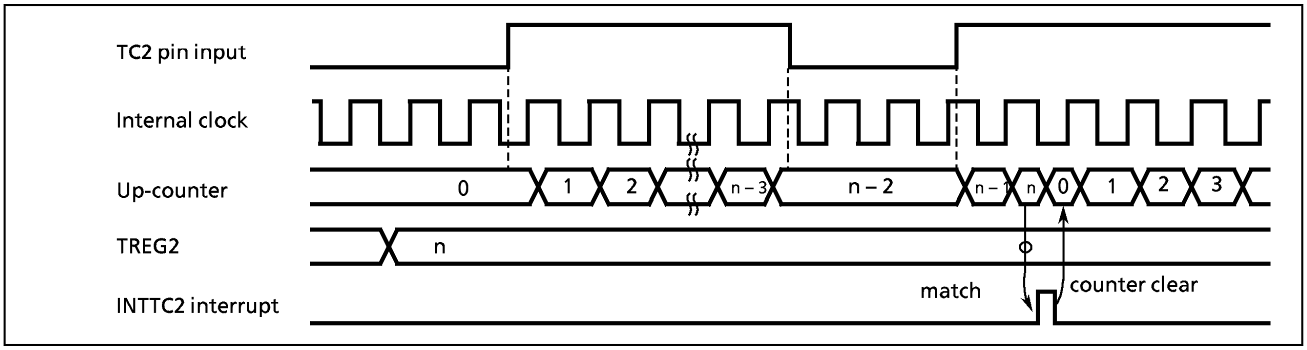


Figure 2-27. Window Mode Timing Chart

2.7 8-Bit Timer / Counter 3 (TC3)

2.7.1 Configuration

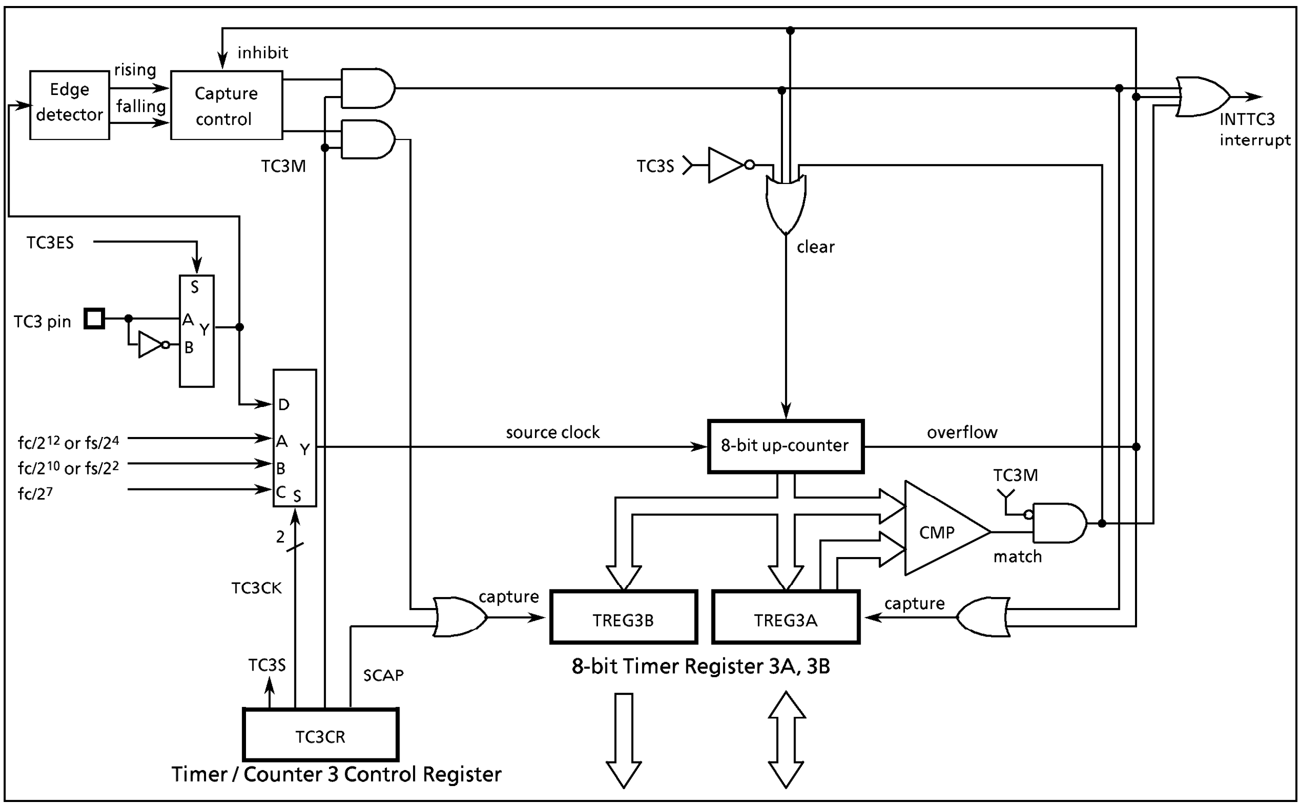


Figure 2-28. Timer / Counter 3

2.7.2 Control

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

Register	7	6	5	4	3	2	1	0	
TREG3A (0018_H)									Read / Write
TREG3B (0019_H)									Read only
TC3CR ($001A_H$)		SCAP		TC3S	TC3CK			TC3M	(Initial value : *0*0 00*0)

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

Register	7	6	5	4	3	2	1	0	
EINTCR (0037_H)	(INT1NC)	(INT0EN)	(TC4ES)	(INT4ES)	TC3ES	(INT2ES)	(INT1ES)		(Initial value : 0000 000*)

TC3ES	TC3 edge select	0 : Rising edge 1 : Falling edge	write only

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

Figure 2-29. Timer Register 3 and TC3 Control Registers

2.7.3 Function

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

(1) **Timer** Mode

In this mode, the internal clock shown in Table 2-5. is used for counting up. The contents of TREG3A are compared with the contents of up-counter. If a match is found, a timer / counter 3 interrupt (INTTC3) is generated, and the up-counter is cleared. Counting up resumes after the up-couter is cleared. The current contents of up-counter are loaded into TREG3B by setting SCAP (bit 6 in TC3CR) to "1". SCAP is automatically cleared after capturing.

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

Source clock		Resolution		Maximum setting time	
NORMAL1/2, IDLE1/2 mode	SLOW, SLEEP mode	fc = 8 MHz	fs = 32.768 kHz	fc = 8 MHz	fs = 32.768 kHz
fc/2^{12} or fs/2^4 [Hz]	fs/2^4 [Hz]	512 μs	488.28 μs	131.1 ms	124.5 ms
fc/2^{10} or fs/2^2	–	128 μs	122.07 μs	32.6 ms	31.12 ms
fc/2^7	–	16 μs	–	4.1 ms	–

(2) **Event Counter** Mode

In this mode, the TC3 pin input pulses are used for counting up. Either the rising or falling edge can be selected with TC3ES (bit 3 in EINTCR). The contents of TREG3A are compared with the contents of the up-counter. If a match is found, an INTTC3 interrupt is generated and the counter is cleared. The maximum applied frequency is $fc/2^4$ [Hz] in NORMAL1/2 or IDLE1/2 mode, and $fs/2^4$ [Hz] in SLOW or SLEEP mode. Two or more machine cycles are required for both the high and low levels of the pulse width.

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

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

```
LD    (TC3CR) , 00001100B     ;  Sets TC3 mode and source clock
LD    (TREG3A) , 19H          ;  0.5 [s] ÷ 1/50 = 25 = 19H
SET   (EIRH) .EF8             ;  Enables INTTC3 interrupt
EI
LD    (TC3CR) , 00011100B     ;  Start TC3
```

(3) **Capture** Mode

The pulse width, period and duty of the TC3 pin input are measured in this mode, which can be used in decoding the remote control signals, etc. The counter is free running by the internal clock. On the rising (falling) edge of the TC3 pin input, the current contents of counter is loaded into TREG3A, then the up-counter is cleared to "0" and an INTTC3 interrupt is generated. On the falling (rising) edge of the TC3 pin input, the current contents of the counter is loaded into TREG3B. In this case, counting continues. On the next rising (falling) edge of the TC3 pin input, the current contents of counter are loaded into TREG3A, then the counter is cleared again and an interrupt is generated. If the counter overflows before the edge is detected, FF_H is set into TREG3A, and the counter is cleared and an overflow interrupt (INTTC3) is generated. During interrupt processing, it can be determined whether or not there is an overflow by checking whether or not the TREG3A value is FF_H. Also, after an interrupt (capture to TREG3A, or overflow detection) is generated, capture and overflow detection are halted until TREG3A has been read out; however, the counter continues.

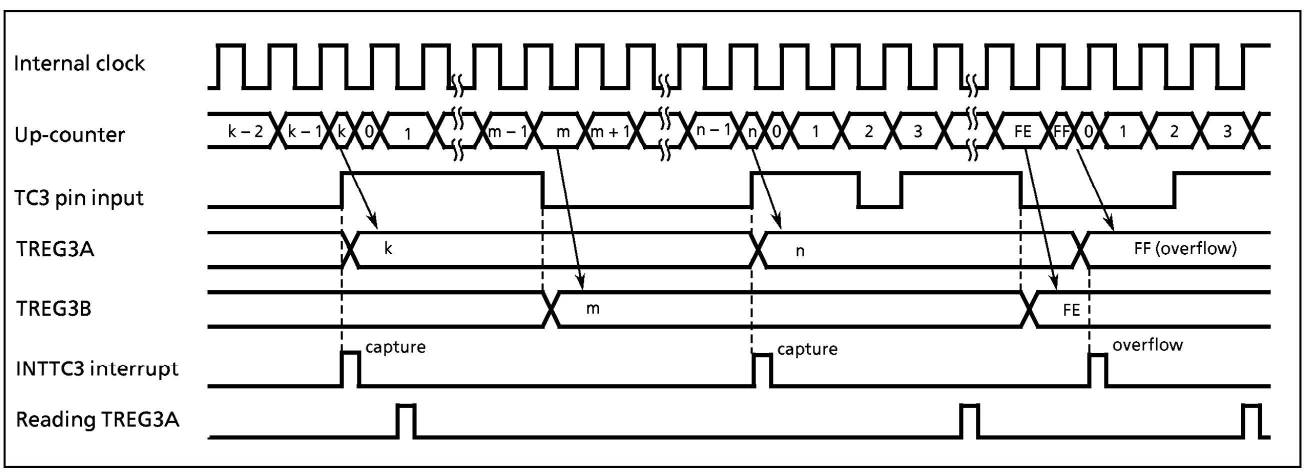


Figure 2-30. Timing Chart for Capture Mode (TC3ES = 0)

2.8 8-bit Timer / Counter (TC4)

2.8.1 Configuration

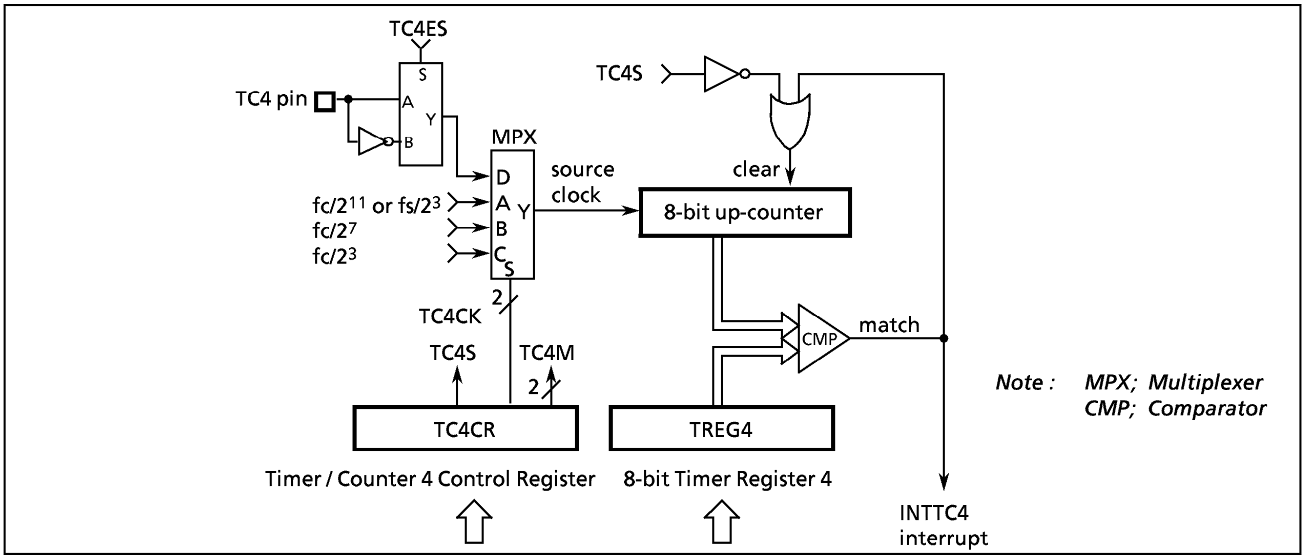


Figure 2-31. Timer / Counter 4

2.8.2 Control

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

TREG4 ($001B_H$)

7	6	5	4	3	2	1	0	
								Write only

TC4CR ($001C_H$)

7	6	5	4	3	2	1	0	
"1"	"1"		TC4S	TC4CK		TC4M		(Initial value : 00*0 0000)

TC4M	TC4 operating mode select	00 : Timer / event counter mode 01 : Reserved 1* : Reserved	write only
TC4CK	TC4 source clock select	00 : Internal clock fc/2^{11} or fs/2^3 [Hz] 01 : Internal clock fc/2^7 10 : Internal clock fc/2^3 11 : External clock (TC4 pin input)	
TC4S	TC4 start control	0 : Stop and counter clear 1 : Start	

EINTCR (0037_H)

7	6	5	4	3	2	1	0	
(INT1 NC)	(INT0 EN)	TC4 ES	(INT4 ES)	(TC3 ES)	(INT2 ES)	(INT1 ES)		(Initial value : 0000 000*)

TC4ES	TC4 edge select	0 : Rising edge 1 : Falling edge	write only

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

Note 2 : Set the operating mode, the source clock selection and the edge selection (TC4ES) when the TC4 stops (TC4S = 0).

Note 3 : Always write "1" to bit 7 and bit 6 in TC4CR.

Note 4 : Values to be loaded to the timer register must satisfy the following condition.
TREG4 > 0

Note 5 : TC4CR, EINTCR and the TREG4 are write-only registers and must not be used with any of the read-modify-write instructions such as SET, CLR, etc.

Figure 2-32. Timer Register 4 and TC4 Control Registers

2.8.3 Function

The timer / counter 4 has two operating modes : timer and event counter mode.

(1) **Timer** Mode

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

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

Source clock		Resolution		Maximum setting time	
NORMAL1/2, IDLE1/2 mode	SLOW, SLEEP mode	At fc = 8 MHz	At fs = 32.768 kHz	At fc = 8 MHz	At fs = 32.768 kHz
fc/2^{11} or fs/2^3 [Hz]	fs/2^3 [Hz]	256 μs	244.14 μs	65.28 ms	62.25 ms
fc/2^7	–	16 μs	–	4.08 ms	–
fc/2^3	–	1 μs	–	255 μs	–

(2) **Event Counter** Mode

In this mode, the TC4 pin input (external clock) pulse is used for counting up. Either the rising or falling edge can be selected with TC4ES (bit 5 in EINTCR). The contents of TREG4 are compared with the contents of the up-counter. If a match is found, an INTTC4 interrupt is generated and the counter is cleared. The maximum applied frequency is $fc/2^4$ [Hz] in NORMAL1/2 or IDLE1/2 mode, and $fs/2^4$ [Hz] in SLOW or SLEEP mode. Two or more machine cycles are required for both the high and low levels of the pulse width.

2.9 Serial Bus Interface (SBI-ver.A)

The 87CM39/P39/S39 has a 1-channel serial bus interface which employs a clocked-synchronous 8-bit serial bus interface and an I^2C bus.

The serial bus interface is connected to an external device through P35 (SDA0) / P73 (SDA1) and P34 (SCL0) / P72 (SCL1) in the I^2C bus mode; and through P36 ($\overline{SCK0}$) / P74 ($\overline{SCK1}$), P35 (SO0) / P73 (SO1), and P34 (SI0) / P72 (SI1) in the clocked-synchronous 8-bit SIO mode.

The serial bus interface pins are also used as the P3 / P7 port. When used as serial bus interface pins, set the P3 / P7 output latches of these pins to "1". When not used as serial bus interface pins, the P3 / P7 port is used as a normal I/O port.

I^2C bus has no an arbitration function which is necessary when two or more master devices scramble for the bus control. In master mode, other devices which are connected on the same bus need be slave devices. (single master)

Note : When a multi master I^2C bus system operates in I^2C bus mode of this serial bus interface circuit, there is a possibility that the following problems raise. I^2C bus mode of this serial bus interface circuit should be used by a single master I^2C bus system.

1. *The SCL line is fixed to low level and transferring stops by the serial bus interface circuit. The other devices can not run on the SCL line. Thus the bus locks.*
2. *The SCL pin is pulled down to low level regardless of the state of the SCL line by the serial bus interface circuit. A period of high-level SCL clock pulse which other devices output is shortened. The minimum value of which the SCL clock holds high level is not satisfied, which is specified with the I2C bus standard.*

2.9.1 Configuration

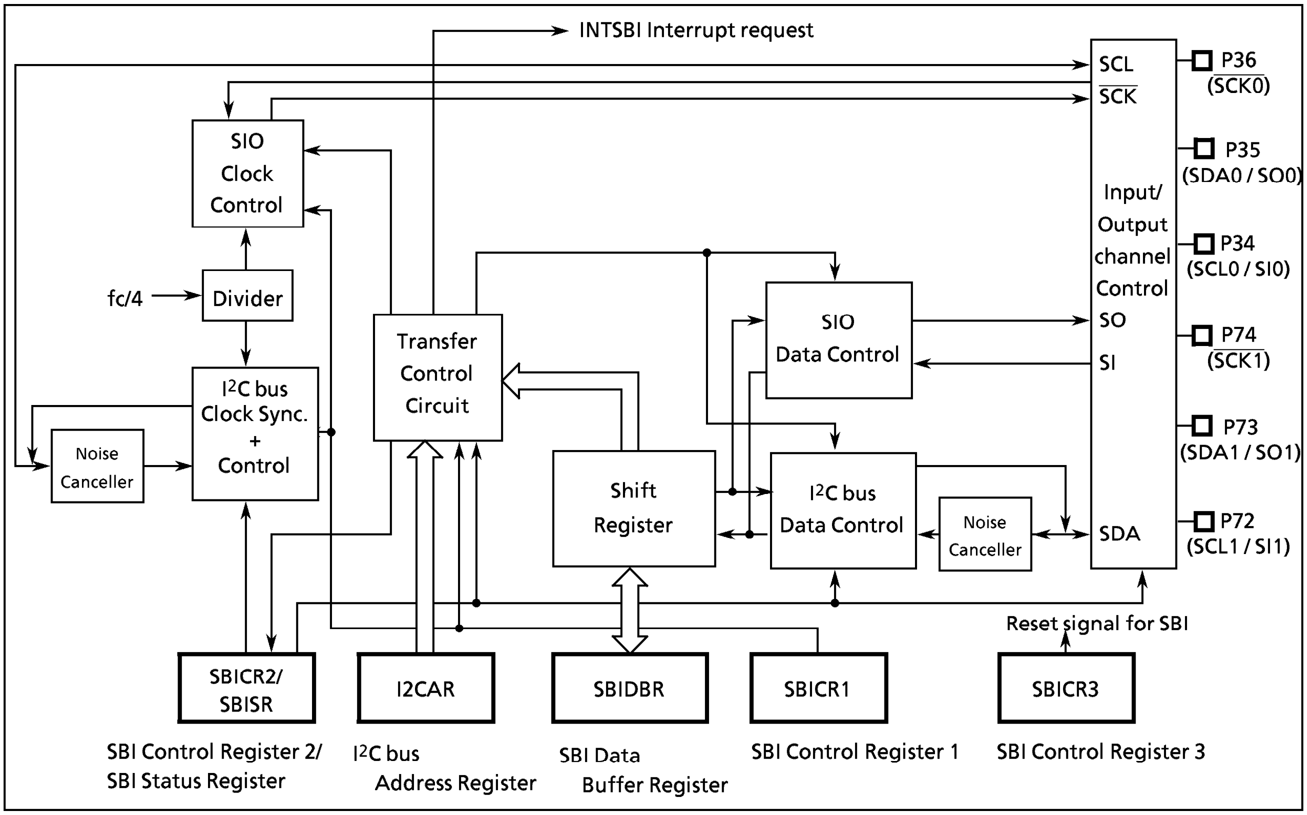


Figure 2-33. Serial Bus Interface (SBI-ver.A)

2.9.2 Serial Bus Interface (SBI-ver.A) Control

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

- Serial bus interface control register 1 (SBICR1)
- Serial bus interface control register 2 (SBICR2)
- Serial bus interface control register 3 (SBICR3)
- Serial bus interface data buffer register (SBIDBR)
- I^2C bus address register (I2CAR)
- Serial bus interface status register (SBISR)

The above registers differ depending on a mode to be used.
Refer to Section "2.9.4 I^2C bus Mode Control" and "2.9.6 Clocked-synchronous 8-bit SIO Mode Control".

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

The data formats when using the serial bus interface circuit in the I^{2}C bus mode are shown below.

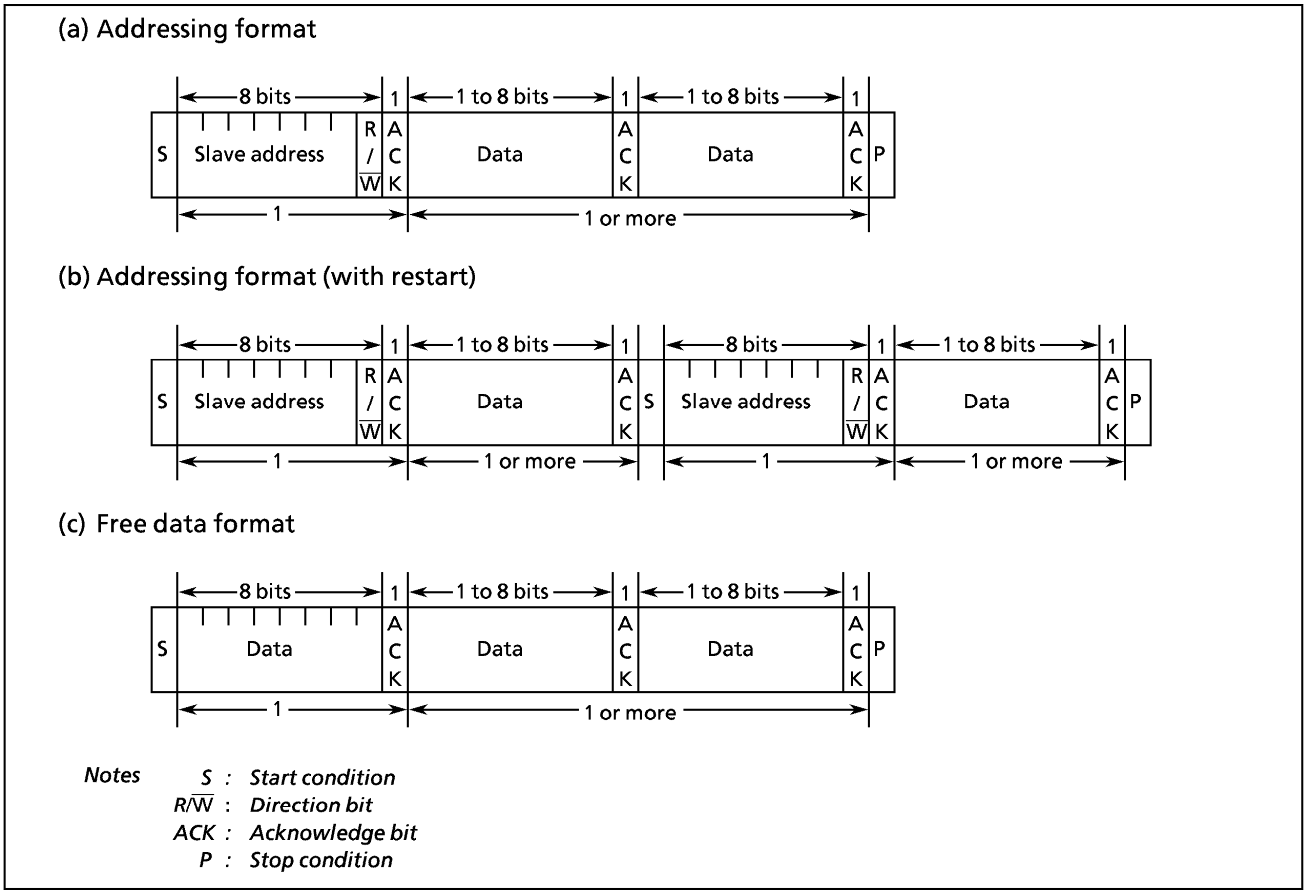


Figure 2-34. Data Format

2.9.4 I²C Bus Mode Control

The following registers are used for control and operation status monitoring when using the serial bus interface (SBI-ver.A) in the I²C bus mode.

Serial Bus Interface Control Register 1

SBICR1 (0020H)

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

(Initial value 0000 0000)

		BC	ACK = 0 Number of Clock	ACK = 0 Bits	ACK = 1 Number of Clock	ACK = 1 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 : Does not generate clock pulse for acknowledgment. (master mode) / Does not count clock pulse for acknowledgment. (slave mode) 1 : Generates clock pulse for acknowledgment. (master mode) / Counts clock pulse for acknowledgment. (slave mode)					Read/ Write
CHS	Input / Output channel selection	0 : Channel0 (SCL0, SDA0) 1 : Channel1 (SCL1, SDA1)					Read/ Write
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 111 : reserved (000–110) at fc = 8 MHz (Output on SCL pin)					Write only

*Note 1 : fc ; high-frequency clock [Hz], * ; don't care*
Note 2 : Set the BC to "000" before switching to a clock-synchronous 8-bit SIO mode.
Note 3 : SBICR1 has write-only register bits, which cannot access any of in read-modify-write instructions such as bit operate, etc.

Serial Bus Interface Data Buffer Register

SBIDBR (0021H)

7	6	5	4	3	2	1	0

(Initial value **** ****) Read / Write

Note 1 : When writing transmitted data, start from the MSB.
Note 2 : Cannot read the data which was written into SBIDBR, since a write data buffer and a read data buffer are independent in SBIDBR. Therefore, cannot access it any of in read-modify-write instructions such as bit operate, etc.
Note3 : A value written to SBIDBR is cleared to "0" by INTSBI interrupt request signal.
*Note4 : * ; don't care*

I²C bus Address Register

I2CAR (0022H)

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

(Initial value 0000 0000)

SA	87CM39/P39/S39 slave address selection		Write only
ALS	Address recognition mode specification	0 : Slave address recognition 1 : Non slave address recognition	

Note : I2CAR is a write-only register, which cannot access any of in read-modify-write instructions such as bit operate, etc.

Figure 2-35. Serial Bus Interface Control Register 1 / Serial Bus Interface Data Buffer Register/ I²C bus Address Register in the I²C 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	0 : Slave 1 : Master	Write only
TRX	Transmitter / receiver selection	0 : Receiver 1 : Transmitter	
BB	Start / stop generation	0 : Generate the stop condition when the MST, TRX, and PIN are "1". 1 : Generate the start condition when the MST, TRX, and PIN are "1".	
PIN	Cancel interrupt service request	0 : – 1 : Cancel interrupt service request	
SBIM	Serial bus interface operating mode selection	00 : Port mode (serial bus interface output disable) 01 : SIO mode 10 : I^2C bus mode 11 : Reserved	

*Note 1 : * ; don´t care*
Note 2 : Switch a mode to port mode after confirming that the bus is free.
Note 3 : Swich a mode to I2Cbus mode after confirming that input signals via port are high level.
Note 4 : SBICR2 has write-only register bits, which can not access any of in read-modify-write instructions such as bit operate, etc.
Note 5 : Clear bits 1 and 0 in SBICR2 to "0".

Serial Bus Interface Status Register

SBISR (0023_H)

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

(Initial value 0001 0000)

MST	Master / Slave selection status monitor	0 : Slave 1 : Master	Read only
TRX	Transmitter / Receiver selection status monitor	0 : Receiver 1 : Transmitter	
BB	Bus status monitor	0 : Bus free 1 : Bus busy	
PIN	Interrupt service request status monitor	0 : Requesting interrupt service 1 : Releasing interrupt service request	
AL	Noise detection monitor	0 : Does not detect noise 1 : Detects noise	
AAS	Slave address match detection monitor	0 : Does not detect slave address match or "GENERAL CALL" 1 : Detects slave address match or "GENERAL CALL"	
AD0	"GENERAL CALL" detection monitor	0 : Does not detect "GENERAL CALL" 1 : Detects "GENERAL CALL"	
LRB	Last received bit monitor	0 : Last received bit is "0" 1 : Last received bit is "1"	

Serial Bus Interface Control Register 3

SBICR3 (0024_H)

7	6	5	4	3	2	1	0
							SWRST

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

SWRST	Software Reset	0 : – 1 : Initialize SBI (After initialize SBI, SWRST is automatically cleared to 0.)	Read/ Write

Figure 2-36. Serial Bus Interface Control Register 2 / Serial Bus Interface Status Register / Serial Bus Interface Control Register 3 in the I^2Cbus Mode

(1) **Acknowledgment mode specification**

Set the ACK (bit 4 in SBICR1) to "1" for operation in acknowledgment mode. When the serial bus interface circuit is the master mode, an additional clock pulse is generated for an acknowledge signal. In the transmitter mode during this additional clock pulse cycle, the SDA pin is released in order to receive the acknowledge signal from the receiver. In the receiver mode during this additional clock pulse cycle, the SDA pin is set to low level generating the acknowledge signal.

Clear the ACK to "0" for operation in the non-acknowledgment mode. When the serial bus interface circuit is the master mode, a clock pulse for the acknowledge signal is not generated.

In the acknowledgment mode, when the serial bus interface circuit is the slave mode, clocks are counted for the acknowledge signal. During the clock for the acknowledge signal, when a received slave address matches to a slave address set to the I2CAR or a "GENERAL CALL" is received, the SDA pin is set to low level generating an acknowledge signal.

After a received slave address matches to a slave address set to the I2CAR and a "GENERAL CALL" is received, in the transmitter mode during the clock for the acknowledge signal, the SDA pin is released in order to receive the acknowledge signal from the receiver. In the receiver mode, the SDA pin is set to low level generating an acknowledge signal.

In the non-acknowledgment mode, when the serial bus interface circuit is the slave mode, clocks for the acknowledge signal are not counted.

(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.

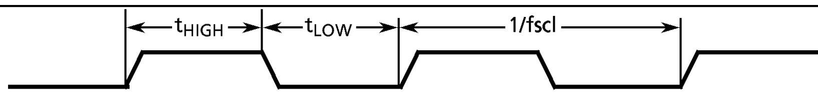
Since the BC is cleared to “000” by a start condition, a slave address and direction bit transmissions are executed in 8 bits. Other than these, the BC retains a specified value.

(3) **Serial clock**

a. Clock source

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

Four or more machine cycles are required for both the high and low levels of the pulse width of a clock which is input externally in both the master and slave 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 2-37. Clock Source

b. Clock synchronization

The I^2C bus has a clock synchronization function to meet the transfer speed to a slow processing device when a transfer is performed between devices which have different process speed.

The clock synchronization functions when the SCL pin is high level and the SCL line of the bus is low level in the serial bus interface circuit. The serial bus interface circuit waits counting a clock pulse in high level until the SCL line of the bus is high level. When the SCL line of the bus is high level, the serial bus interface circuit starts counting during high level. The clock synchronization function holds clocks which are output from the serial interface circuit to be high level.

The slave device can stop the clock output of the master device on one word or one bit basis. Additionally, the transfer speed by the master device matches to the process speed of the slave device.

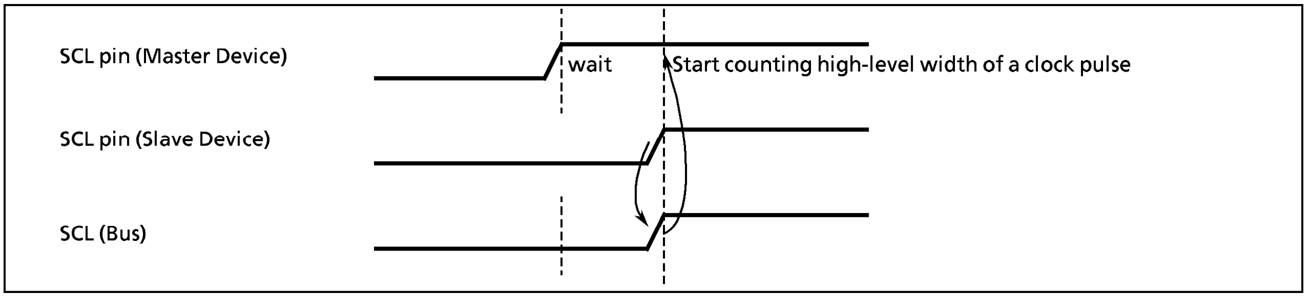


Figure 2-38. Clock Synchronization

(4) **Slave address and address recognition mode specification**
To operate the serial bus interface circuit in the addressing format which recognizes the slave address, clear the ALS (bit 0 in I2CAR) to "0" and set the slave address to the SA (bits 7 to 1 in I2CAR).
To operate the serial bus interface circuit in the free data format which does not recognize the slave address, set the ALS to "1". When the serial bus interface circuit is used in the free data format, the slave address and the direction bit are not recognized. They are handled as data just after generation of start conditions.

(5) **Master/slave selection**
Set the MST (bit 7 in the SBICR2) to “1” for operating the serial bus interface as a master device. Clear the MST to “0” for operation as a slave device. The MST is cleared to “0” by the hardware after a stop condition on a bus is detected or the noise is detected.

(6) **Transmitter / receiver selection**
Set the TRX (bit 6 in the SBICR2) to "1" for operating the serial bus interface circuit 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, the TRX is set to "1" by the hardware if the direction bit (R/W) sent from the master device is "1", and is cleared to "0" by the hardware if the bit is "0". In the master mode, after an acknowledge signal is returned from the slave device, the TRX is cleared to "0" by the hardware if a transmitted direction bit is "1", and is set to "1" by the hardware 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 bus is detected or the noise is detected.

The following shows TRX change conditions in each mode and TRX after changing.

Mode	Direction bit	Change condition	TRX after changing
Slave mode	0	A received slave address is the same as a value set to I2CAR.	0
	1		1
Master mode	0	ACK signal is returned.	1
	1		0

When the serial bus interface circuit operates in the free data format, the slave address and the direction bit are not recognized. They are handled as data just after generating a start condition. The TRX was not changed by the hardware.

(7) **Start/stop condition generation**

When the BB (bit 5 in the SBICR2) is "0", the slave address and the direction bit which are set to the SBIDBR are output on a bus after generating a start condition by writing "1" to the MST, TRX, BB, and PIN. It is necessary to set transmitted data to the data buffer register (SBIDBR) and set "1" to ACK beforehand.

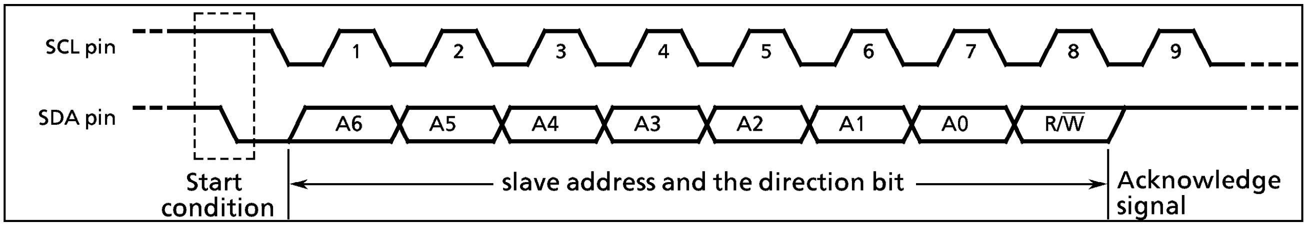


Figure 2-39. Start Condition Generation and Slave Address Generation

When the BB is "1", a sequence of generating a stop condition is started by writing "1" to the MST, TRX, and PIN, and "0" to the BB. Do not modify the contents of MST, TRX, BB and PIN until a stop condition is generated on a bus.
When a stop condition is generated and the SCL line on the bus is set to low level by another device, a stop condition is generated after releasing the SCL line.

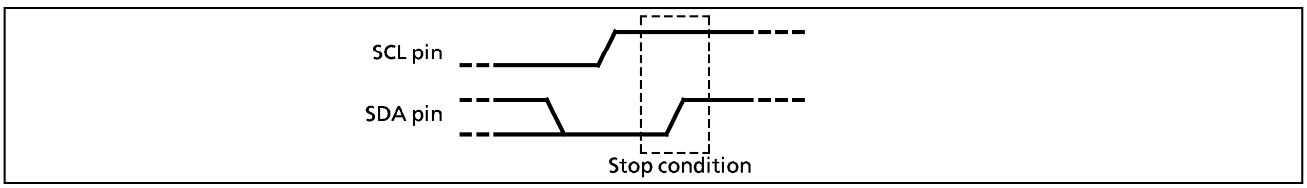


Figure 2-40. Stop Condition Generation

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 on a bus.

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

When the serial bus interface circuit is the master mode and transferring a number of clocks set by the BC and the ACK is complete, a serial bus interface interrupt request (INTSBI) is generated.
In the slave mode, the INTSBI is generated when the received slave address is the same as the value set to the I2CAR and an acknowledge signal is output, when a "GENERAL CALL" is received and an acknowledge signal is output, or when transferring / receiving data is complete after the received slave address is the same as the value set to the I2CAR and a "GENERAL CALL" is received.
When the serial bus interface interrupt request occurs, the PIN (bit 4 in the SBISR) is cleared to "0". During the time that the PIN is "0", the SCL pin is set to low level.
Either writing or reading data to or 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}.
Although the PIN (bit 4 in the SBICR2) can be set to "1" by the program, the PIN is not cleared to "0" when it is written "0".

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

The SBIM (bits 3 and 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 after confirming that the serial bus interface pin is high level. Switch a mode to port after confirming that the bus is free.

(10) **Noise detection monitor**

The I^2C bus is easy to be affected by noise, because the bus is driven by the open drain and the pull-up resistor.

With the serial bus interface circuit, the SDA pin output and the SDA line level are compared at a rise of the SCL line on the bus, and whether data are output correctly on the bus is detected only in the master transmitter mode.

When the SDA pin output differs from the SDA line level, the AL (bit 3 in the SBISR) is set to "1". When the AL is set to "1", the SDA pin is released and the MST and the TRX are cleared to "0" by the hardware. The serial bus interface circuit changes to the slave receiver mode, and the serial bus interface circuit continues outputting clocks until transferring data when the AL was set to "1" is completed.

Either writing or reading data to or from the SBIDBR, or writing data to the SBICR2 clears to the AL to "0".

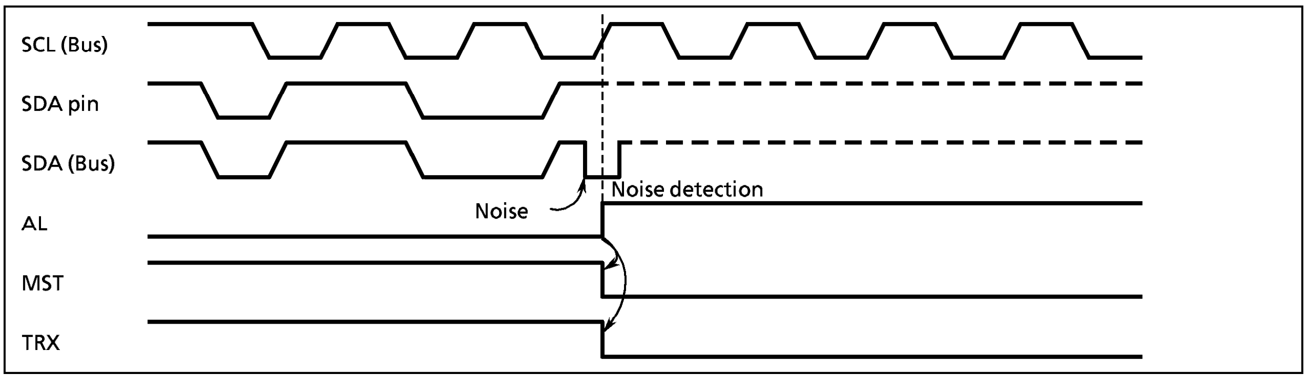


Figure 2-41. Noise Detection Monitor

(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), when receiving "GENERAL CALL" or a slave address with the same value that is set to the I2CAR. When the ALS is "1", the AAS is set to "1" after receiving the first 1-word of data. The AAS is cleared to "0" by 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 received data is "0", after a start condition (GENERAL CALL). The AD0 is cleared to "0" when a start or stop condition is detected on a bus.

(13) **Last received bit monitor**

The SDA value stored at the rising edge of the SCL is set to the LRB (bit 0 in the SBISR). In the acknowledge mode, immediately after an INTSBI interrupt request is generated, an acknowledge signal is read by reading the contents of the LSB.

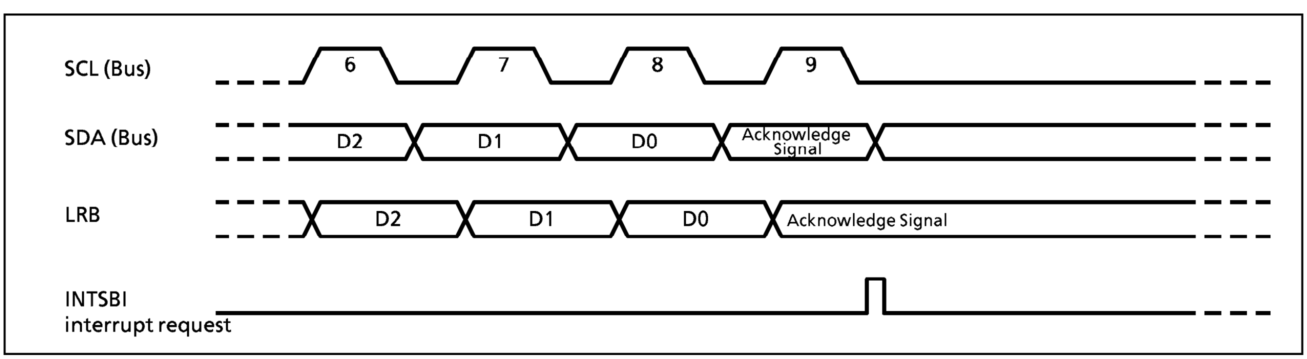


Figure 2-42. Last Received bit Monitor

(14) **Software reset function**

Software reset function is used to initialize SBI, when SBI is locked by external noise, etc.
SWRST is set to "1", internal reset signal pulse is generated and inputted into SBI circuit.
All command registers and status registers are initialized to an initial value.
SWRST is automatically cleared to "0" after initialize SBI circuit.

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

Set the ACK in the SBICR1 to "1", and the BC to 000. Specify the data length to 8 bits to count clocks for acknowledge. Set a transfer frequency to the SCK and a serial bus interface pin to the CHS.
Subsequently, set a slave address to the SA in the I2CAR and clear the ALS to "0" to set an addressing format.
After confirming that the serial bus interface pin is high-level, for specifying the default setting to a slave receiver mode, clear "0" to the MST, TRX, and BB in the SBICR2, set "1" to the PIN, "10" to the SBIM, and "0" to bits 1 and 0,

Note : The initialization of the serial bus interface circuit must be complete within the time from all devices which are connected to the bus have initialized to any device does not generate a start condition. If not, there is a possibility that another device starts transferring before an end of the initialization of the serial bus interface circuit. Data can not be received correctly.

(2) **Start Condition and Slave Address Generation**

Confirm 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 the BB is "0", the start condition are generated and the slave address and the direction bit which are set to the SBIDBR are output on a bus by writing "1" to the MST, TRX, BB and PIN. 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.

*Note 1 : Do not write a slave address to be output to the **SBIDBR** while data are transferred. If data is written to the **SBIDBR**, data to been outputting may be destroyed.*
*Note 2 : Do not start transferring due to another master from writing a slave address to be output to the **SBIDBR** to writing a start condition generation command to the **SBICR2**. The serial bus interface circuit malfunctions because it has not an arbitration function.*

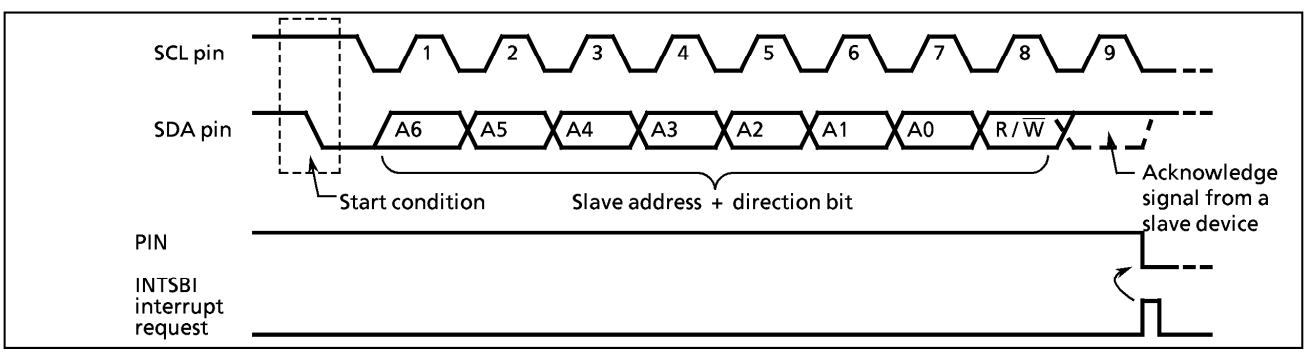


Figure 2-43. Start Condition Generation and Slave Address Transfer

(3) **1-word Data Transfer**

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

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

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

① When the TRX is "1" (Master 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, set the ACK to "1", 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, and an INTSBI interrupt request occurs. The PIN becomes "0" and the SCL pin is set to low level. If the data to be transferred is more than one word in length, repeat the procedure from the LRB test above.

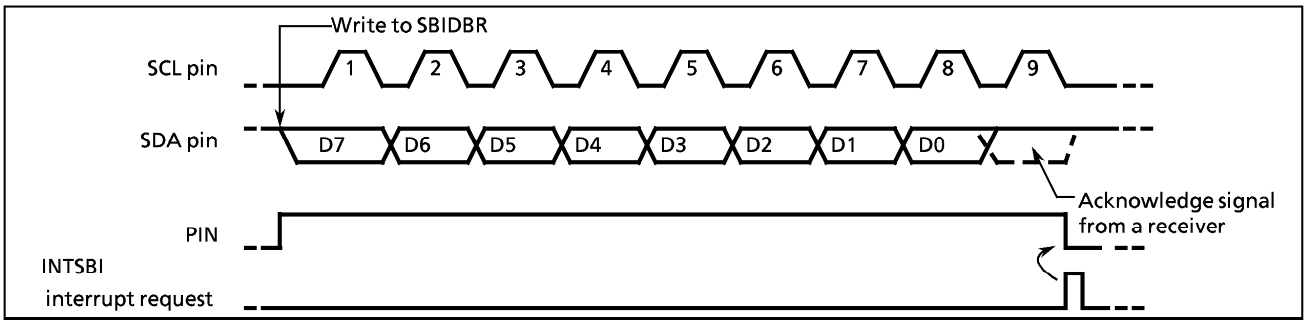


Figure 2-44. Example when BC = "000", ACK = "1" in Transmitter Mode

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 serial bus interface circuit 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 occurs and the PIN becomes "0". Then the serial bus interface circuit pulls down the SCL pin to the low level. The serial bus interface circuit outputs a clock pulse for 1-word of data transfer and the acknowledge signal each time that received data is read from the SBIDBR.

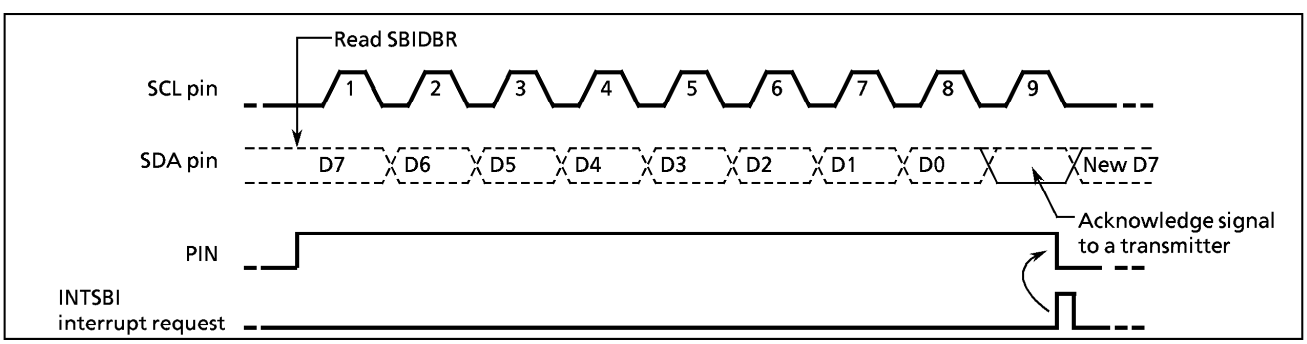


Figure 2-45. Example when BC = "000", ACK = "1" in Receiver Mode

In order to terminate transmitting data to a transmitter, clear the ACK to "0" 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 serial bus interface circuit generates a clock pulse for a 1-bit data transfer. Since the master device is a receiver, the SDA line on a 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 serial bus interface circuit generates a stop condition (Refer to 2.9.5. (4)) and terminates data transfer.

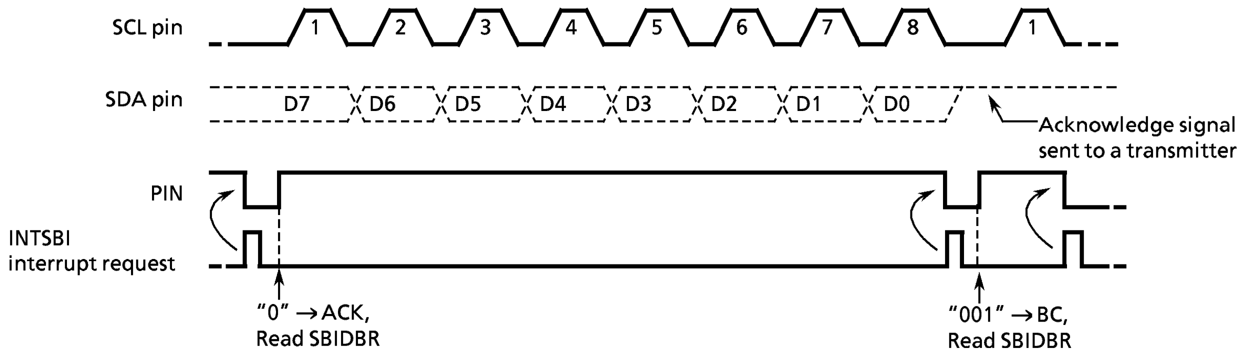


Figure 2-46. Termination of Data Transfer in Master Receiver Mode

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

In the slave mode, the serial bus interface circuit operates either in normal slave mode or in recovery process after a noise detection.
In the slave mode, an INTSBI interrupt request occurs when the serial bus interface circuit 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 serial bus interface circuit operates in a slave mode if a noise is detected. An INTSBI interrupt request occurs when word data transfer terminates after a noise detection. When an INTSBI interrupt request occurs, the PIN (bit 4 in the SBICR2) is reset, and the SCL pin is set to low level. Either reading or writing from or to the SBIDBR or setting the PIN to "1" releases the SCL pin after taking t_{LOW} time.
The serial bus interface circuit 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 2-7. Operation in the Slave Mode

TRX	AL	AAS	AD0	Conditions	Process
1	0	1	0	In the slave receiver mode, the serial bus interface circuit receives a slave address of which the value of the direction bit sent from the master is "1".	Set the number of bits in 1-word to the BC and write transmitted data to the SBIDBR.
		0	0	In the slave transmitter mode, 1-word data is transmitted.	Check the LRB. If the LRB is set to "1", set the PIN to "1" since the receiver does not request next data. Then, clear the TRX to "0" release the bus. If the LRB is cleared to "0", set the number of bits in a word to the BC and write transmitted data to the SBIDBR since the receiver requests next data.
0	1	0	0	The serial bus interface circuit detects the noise when transmitting a slave address or data and terminates transferring word data.	There is a possibility that a serial bus interface circuit does not receive data normally. The recovery process such as a data re-transfer, etc. is needed.
	0	1	1/0	In the slave receiver mode, the serial bus interface circuit receives a slave address or GENERAL CALL of which the value of the direction bit sent from the master is "0".	Read the SBIDBR for setting the PIN to "1" (reading dummy data) or set the PIN to "1".
		0	1/0	In the slave receiver mode, the serial bus interface circuit 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**

When the BB is "1", a sequence of generating a stop condition is started by setting "1" to the MST, TRX and PIN, and "0" to the BB. Do not modify the contents of the MST, TRX, BB, PIN until a stop condition is generated on a bus. When a SCL line of bus is pulled down by other devices, the serial bus interface circuit generates a stop condition after they release a SCL line.

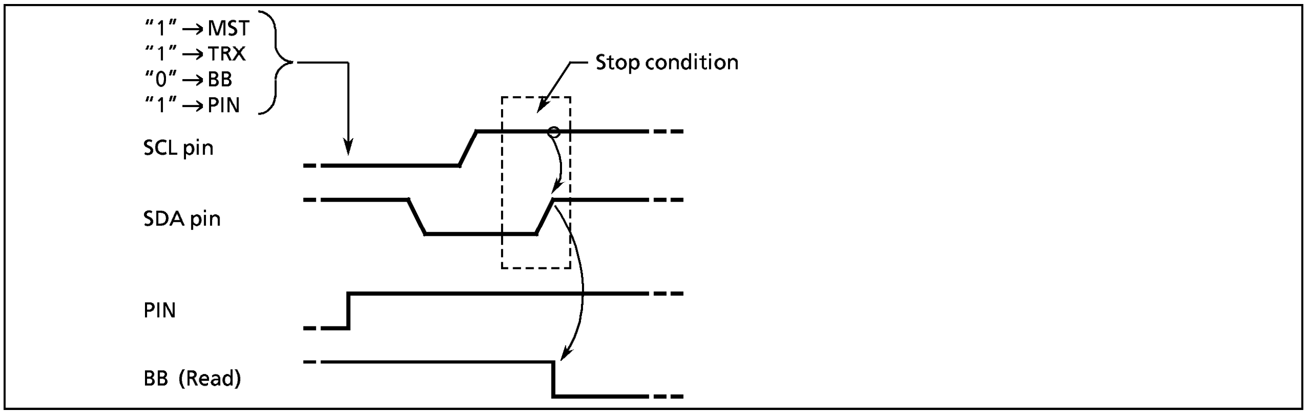


Figure 2-47. Stop Condition Generation

(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 the serial bus interface circuit.

Clear "0" to the MST, TRX, and BB and set "1" to the PIN. 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 serial bus interface circuit is released. Test the LRB until it becomes "1" to check that the SCL line of the bus is not set to 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 μs 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.

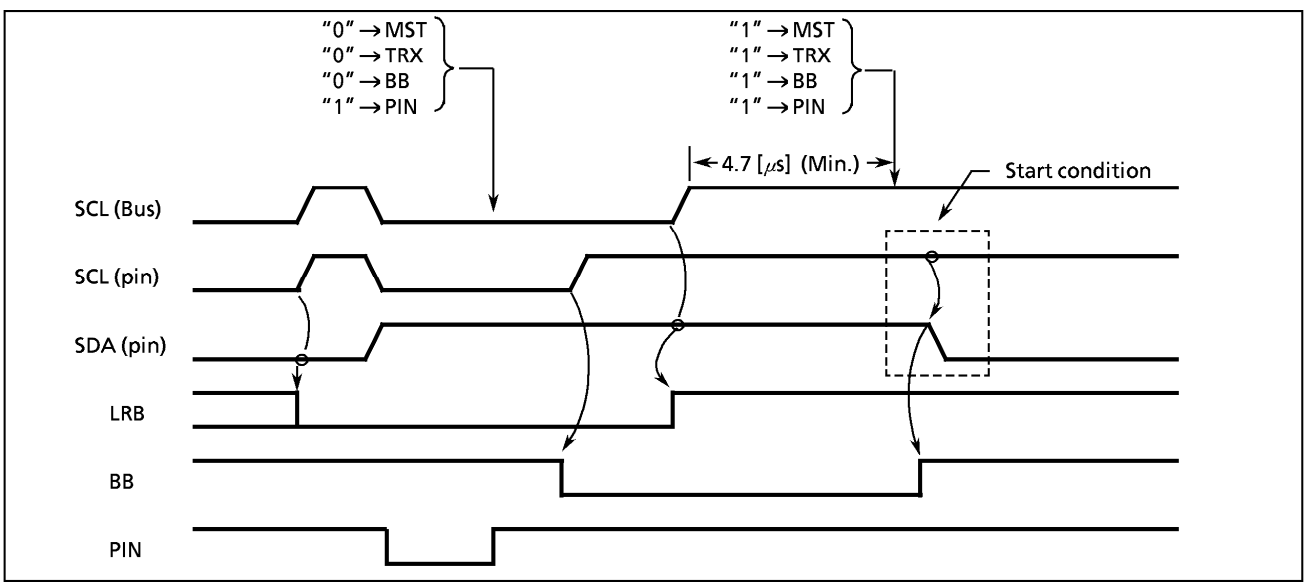


Figure 2-48. Timing Diagram when Restarting

2.9.6 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-ver.A) 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	SIOINH	SIOM		CHS	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	00 : 8-bit transmit mode 01 : reserved 10 : 8-bit transmit/receive mode 11 : 8-bit receive mode	
CHS	Input / Output channel selection	0 : channel0 (SCK0, SO0, SI0) 1 : channle1 (SCK1, SO1, SI1)	
SCK	Serial clock select	000 : fc/2^5 (250 kHz) 001 : fc/2^6 (125 kHz) 010 : fc/2^7 (62.5 kHz) 011 : fc/2^8 (31.25 kHz) 100 : fc/2^9 (15.62 kHz) 101 : fc/2^{10} (7.81 kHz) 110 : fc/2^{11} (3.90 kHz) (000–110: at fc = 8 MHz (Output on SCK pin)) 111 : External clock (input from SCK pin)	

Note 1 : fc ; high-frequency clock [Hz]
Note 2 : Clear the SISO to "0" and set the SIOINH to "1" when setting the transfer mode and serial clock.
Note 3 : SBICR1 is a write-only register, which cannot access any of in read-modify-write instructions such as bit operate, etc.

Serial Bus Interface Data Buffer Register

SBIDBR (0021_H)

7	6	5	4	3	2	1	0

(Initial value **** ****) Read / Write

Note 1 : Cannot read the data which was written into SBIDBR, since a write data buffer and a read data buffer are independent in SBIDBR. Therefore, cannot access it any of in read-modify-write instructions such as bit operate, etc.
Note 2 : don't care

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^2C bus mode 11 : reserved	Write only

*Note 1 : * ; don't care*
Note 2 : Switch a mode to port after data transfer is complete.
Note 3 : Switch a mode to SIO mode after confirming that input signals via port are high level.
Note 4 : SBICR2 is a write-only regiter, which cannot access any of in read-modify-write instructions such as bit operate, etc.
Note 5 : Clear bits 7 to 5 in the SBICR2 to "0", and set bit 4 to "1".

Figure 2-49. Serial Bus Interface Control Register 1 / Serial Bus Interface Data Buffer Register / Serial Bus Interface Control Register 2 in SIO Mode

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 2-50. Serial Bus Interface Status Register in SIO Mode

(1) Serial Clock

a. Clock source

The SCK (bit 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 $\overline{SCK}$ pin. The $\overline{SCK}$ 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.

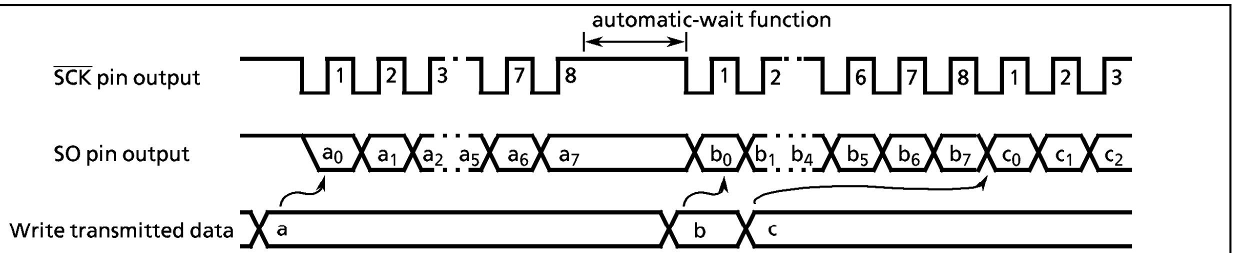


Figure 2-51. Automatic-wait Function

② External clock (SCK = "111")

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

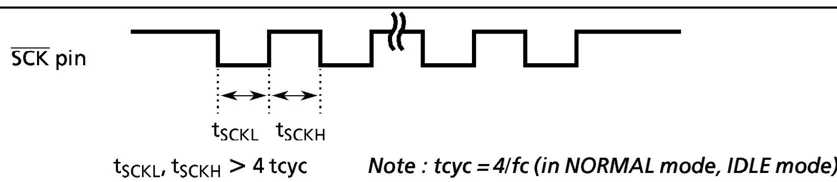


Figure 2-52. Maximum Data Transfer Frequency When External Clock Input

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 $\overline{SCK}$ pin input/output).

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

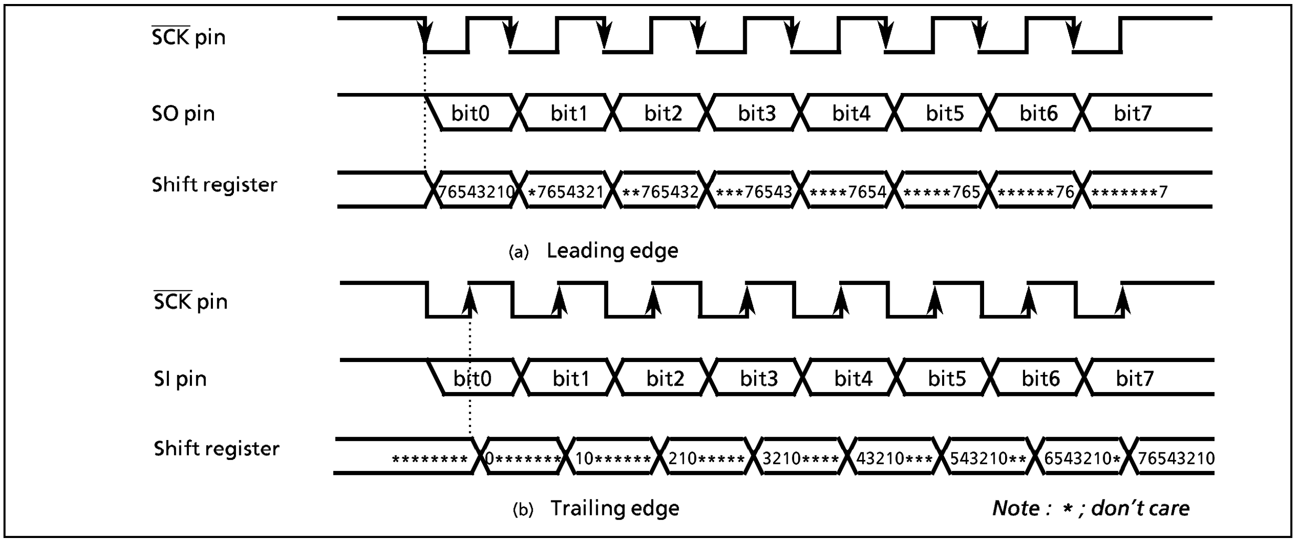


Figure 2-53. Shift Edge

(2) Transfer mode

The SIOM (bit 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 SO 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 data is not loaded to the data buffer register after the specified 8-bit data is transmitted. When new transmit 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.

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

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.

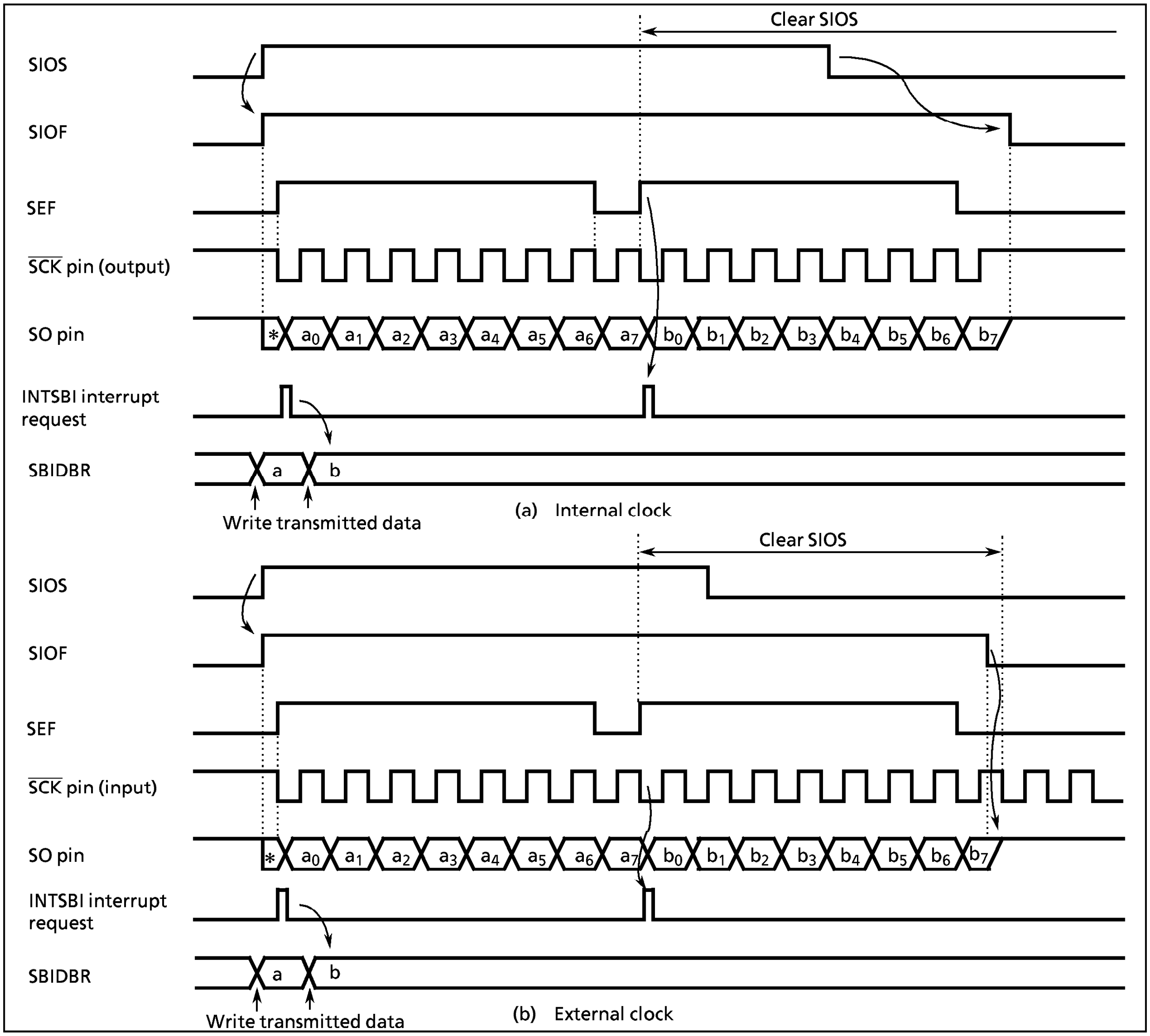


Figure 2-54. Transfer Mode

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 SCK = 0 then loop
         JRS    T , STEST2
         LD     (SBICR1) , 00000111B  ; SIOS ← 0
```

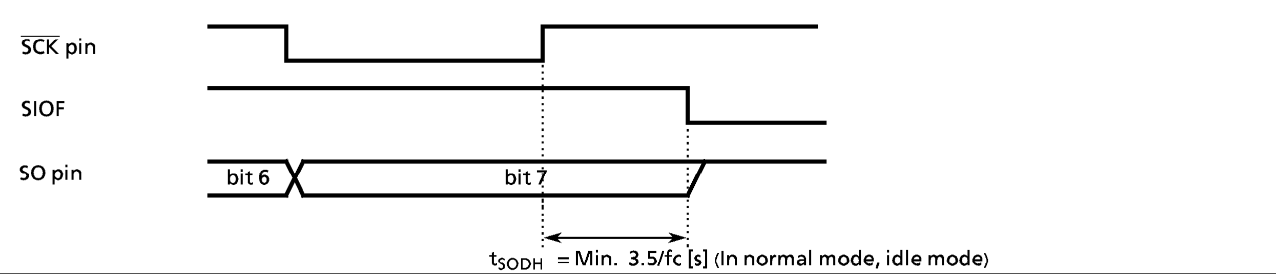


Figure 2-55. Transmitted Data Hold Time at End of Transmit

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 SI 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.*

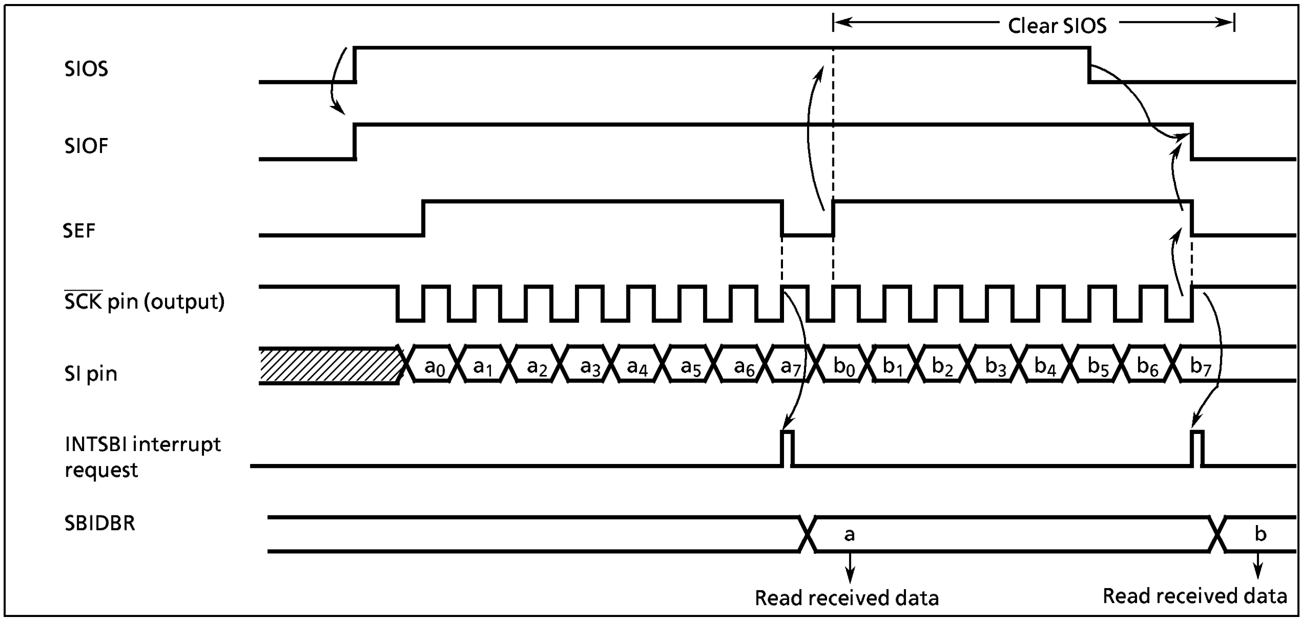


Figure 2-56. Receive Mode (Example : Internal clock)

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 SO 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 SI 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.

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

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.*

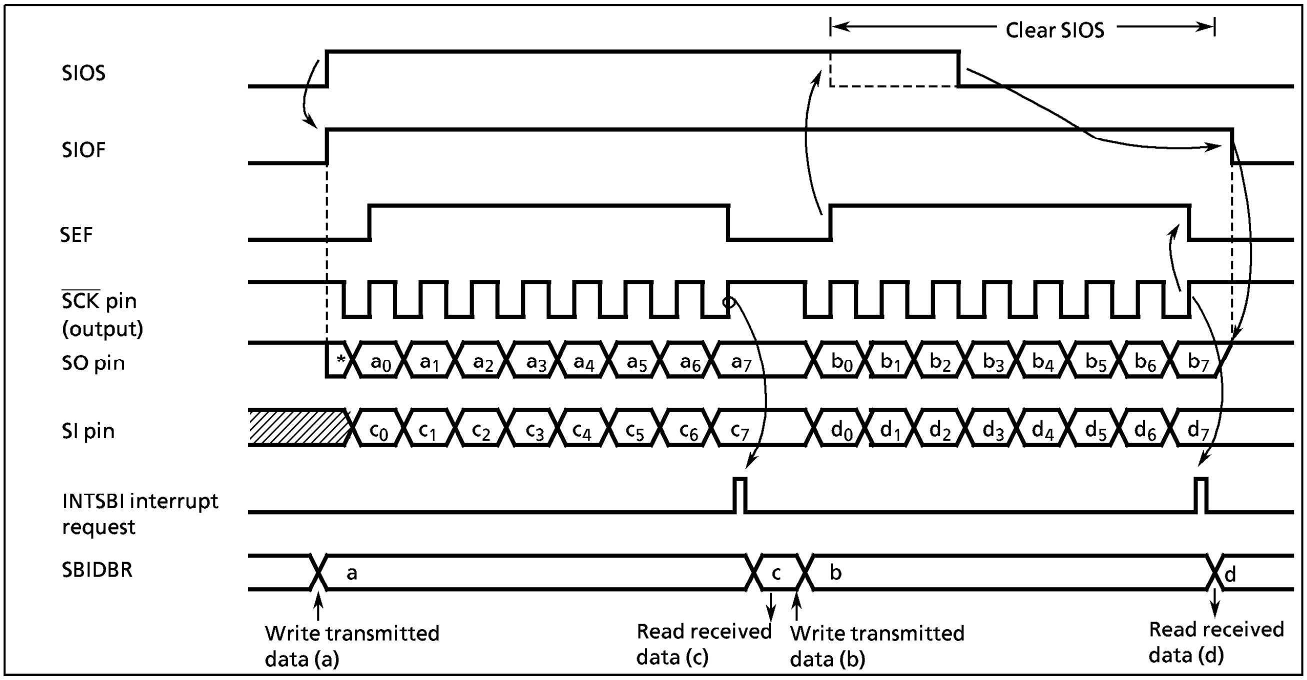


Figure 2-57. Transmit / Receive Mode (Example : Internal clock)

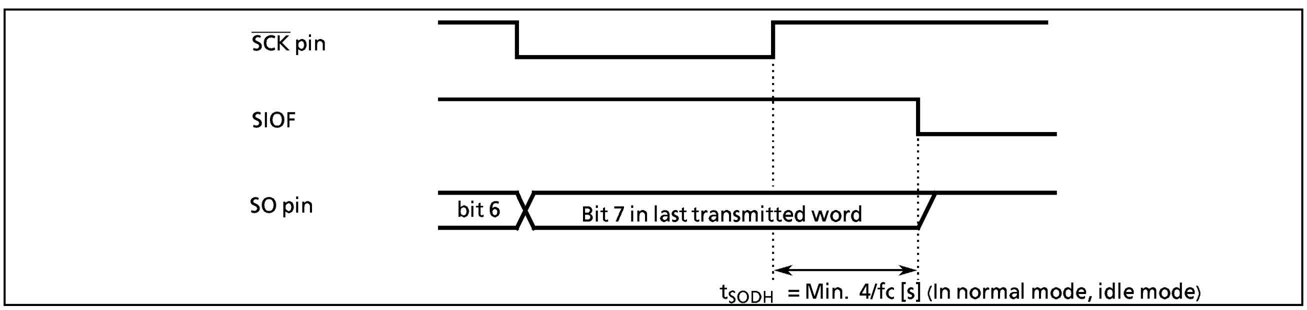


Figure 2-58. Transmitted Data Hold Time at End of Transmit / Receive

2.10 Remote Control Signal Preprocessor / External Interrupt 3 Input Pin

The remote control signal waveform can be determined by inputting the remote control signal waveform from which the carrier wave was eliminated by the receive circuit to P30 (INT3 / RXIN) pin. When the remote control signal preprocessor / external interrupt 3 pin is also used as the P30 port, set the P30 port output latch to "1". When it is not used as the remote control signal preprocessor / external interrupt 3 input pin, it can be used for normal port.

2.10.1 Configuration

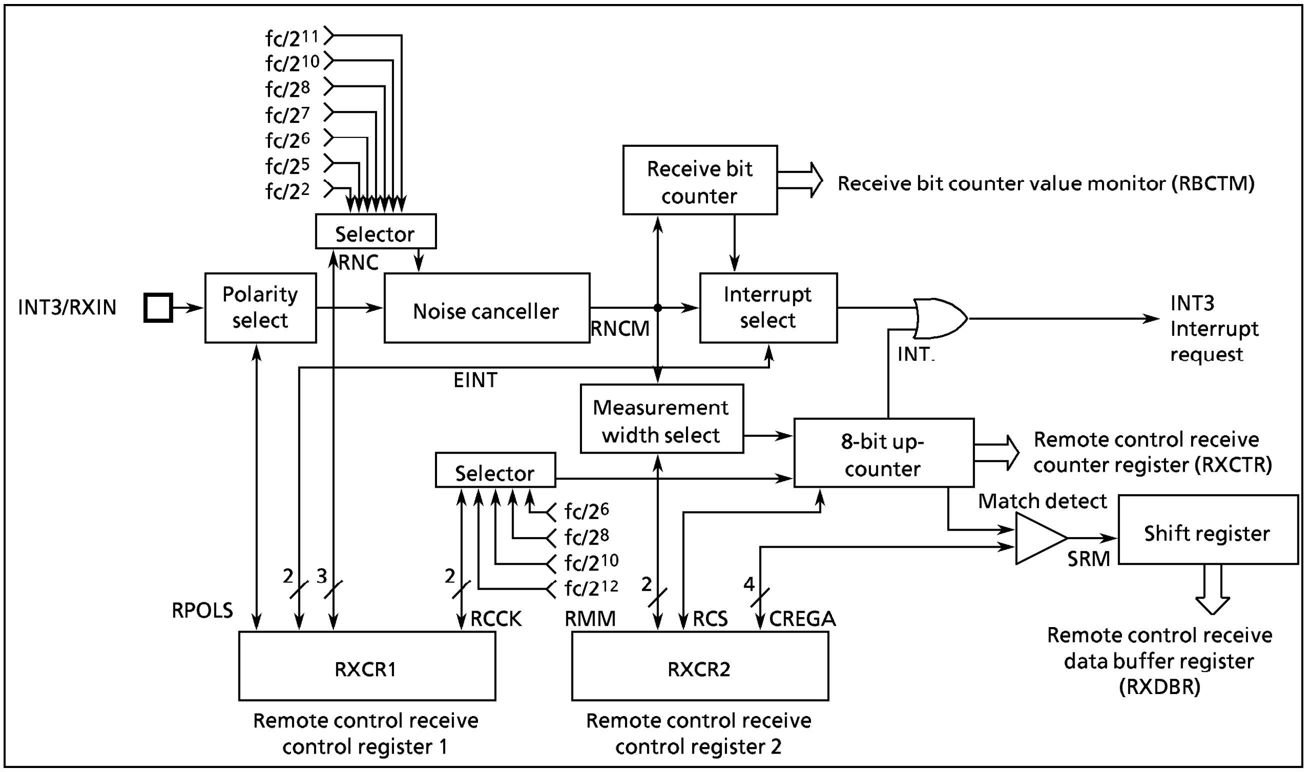


Figure 2-59. Remote Control Signal Preprocessor

2.10.2 Remote Control Signal Preprocessor Control

When the remote control signal preprocessor is used, operating states are controlled and monitored by the following registers. Interrupt requests also use the remote control signal preprocessor / external interrupt 3 input pin.

- Remote control receive control register 1 (RXCR1)
- Remote control receive control register 2 (RXCR2)
- Remote control receive counter register (RXCTR)
- Remote control receive data buffer register (RXDBR)
- Remote control receive status register (RXSR)

When this pin is used for the external interrupt 3 input, set EINT in RXCR1 to other than "11".

Remote control receive control register 1

RXCR1 ($0FD0_H$)

7	6	5	4	3	2	1	0
RCCK		RPOLS	EINT		RNC		

(Initial value : 0000 0000)

RCCK	8-bit up-counter source clock select	00: $fc/2^6$ [Hz] 01: $fc/2^8$ 10: $fc/2^{10}$ 11: $fc/2^{12}$	Read/ Write
RPOLS	Remote control signal polarity select	0 : Positive 1 : Negative	
EINT	Interrupt source select	00: Rising edge 01: Falling edge (when RPOLS = 0) 10: Rising / Falling edge 11: 8-bit receive end	
RNC	Noise canceller noise eliminating time select	000 : Noise canceler disable 001 : $2^2/fc \times 7 - 1/fc$ [s] 010 : $2^5/fc \times 7 - 1/fc$ 011 : $2^6/fc \times 7 - 1/fc$ 100 : $2^7/fc \times 7 - 1/fc$ 101 : $2^8/fc \times 7 - 1/fc$ 110 : $2^{10}/fc \times 7 - 1/fc$ 111 : $2^{11}/fc \times 7 - 1/fc$	

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

Note 2 : After reset, RPOLS does not change the set value in the receiving remote control signal. For setting interrupt edge and measurement data, use EINT and RMM.

Remote control receive control register 2

RXCR2 ($0FD1_H$)

7	6	5	4	3	2	1	0
CREGA				RCS		RMM	

(Initial value : 0000 0*00)

CREGA	Setting of detect time for match with 8-bit up-counter upper 4 bits	Match detect time (Tth) = 16 × CREGA / RCCK [s] CREGA = 0_H to F_H Example : CREGA = 2, RCCK = $fc/2^6$ [Hz] , at fc = 8 MHz Tth = 256 [μs]	Read/ Write
RCS	8-bit up-counter start control	0 : Stop and counter clear 1 : Start	
RMM	Measurement mode select (invalid when EINT = "10")	00: 01: Refer to table 2-8. 10: 11:	

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

Note 2 : When an interrupt source is set for rising / falling edge, low and high widths are forcibly measured separately.

Note 3 : Set CREGA (O_H to F_H) before EINT sets to 8-bit receive end.

Figure 2-60. Remote Control Receive Control Register 1, 2

Remote control receive counter register

RXCTR ($0FD2_H$)

7	6	5	4	3	2	1	0

Read Only
(Initial value : 0000 0000)

Remote control receive data buffer register

RXDBR ($0FD3_H$)

7	6	5	4	3	2	1	0

Read Only
(Initial value : 0000 0000)

Remote control receive status register

RXSR ($0FD4_H$)

7	6	5	4	3	2	1	0
RBCTM					OVFF	SRM	RNCM

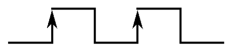
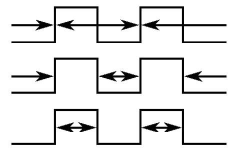
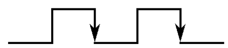
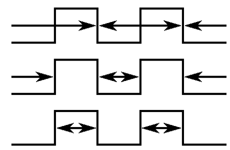
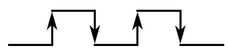
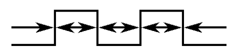
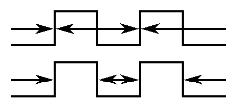
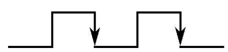
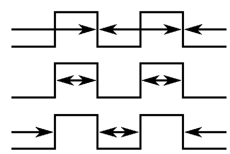
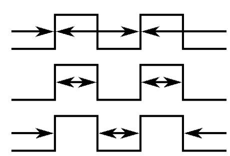
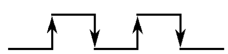
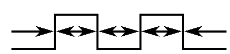
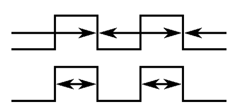
(Initial value : 0000 * 000)

RBCTM	Receive bit counter value monitor		Read Only
OVFF	8-bit up-counter overflow flag	0 : No overflow 1 : Overflow	
SRM	Data buffer register input monitor	0 : Upper 4 bits of 8-bit up-counter < CREGA 1 : Upper 4 bits of 8-bit up-counter ≧ CREGA	
RNCM	Remote control signal monitor after passing through noise canceller		

*Note : * ; don't care*

Figure 2-61. Remote Control Receive Counter Register, Data Buffer Register, Status Register

Table 2-8. Combination of Interrupt Source and Measurement Mode

RPOLS	EINT	RMM	Interrupt source	Measurement mode
0	00	00 10 11		
	01	01 10 11		
	10	—		
	11	00 10	Receive end	
1	00	00 10 11		
	01	01 10 11		
	10	—		
	11	00 10	Receive end	

2.10.3 Noise Elimination Time Setting

The remote control receive circuit has a noise canceller. By setting RNC in RXCR1, input signals shorter than the fixed time can be eliminated as noise.

Table 2-9. Noise Elimination Time Setting

RNC	Minimum signal pulse width	at fc = 8 MHz	Maximum noise width to be eliminated	at fc = 8 MHz
000	——	——	——	——
001	$(2^5 + 5) / fc$ [s]	4.63 [μs]	$(2^2 \times 7 - 1) / fc$ [s]	3.38 [μs]
010	$(2^8 + 5) / fc$	32.63	$(2^5 \times 7 - 1) / fc$	27.88
011	$(2^9 + 5) / fc$	64.63	$(2^6 \times 7 - 1) / fc$	55.88
100	$(2^{10} + 5) / fc$	128.63	$(2^7 \times 7 - 1) / fc$	111.88
101	$(2^{11} + 5) / fc$	256.63	$(2^8 \times 7 - 1) / fc$	223.88
110	$(2^{13} + 5) / fc$	1.025 [ms]	$(2^{10} \times 7 - 1) / fc$	895.88
111	$(2^{14} + 5) / fc$	2.049	$(2^{11} \times 7 - 1) / fc$	1.792 [ms]

2.10.4 Operation

(1) interrupts at rising, falling, or rising / falling edge, and measurement modes

First set EINT and RMM. Next, set RCS to "1" in RCS; the 8-bit up-counter is counted up by the internal clock. After measurement, the 8-bit up-counter value is saved in RXCTR. Then, the 8-bit up-counter is cleared, an INT3 request is generated, and the 8-bit up-counter resumes counting.

If the 8-bit up-counter overflows (FF_H) before measurement is completed, an INT3 request is generated and the overflow flag (OVFF) is set to "1". Then, the 8-bit up-counter is cleared. An overflow can be detected by reading OVFF by the interrupt processing. To restart the 8-bit up-counter, set RCS to "1".

Setting RCS to "1" zero-clears OVFF.

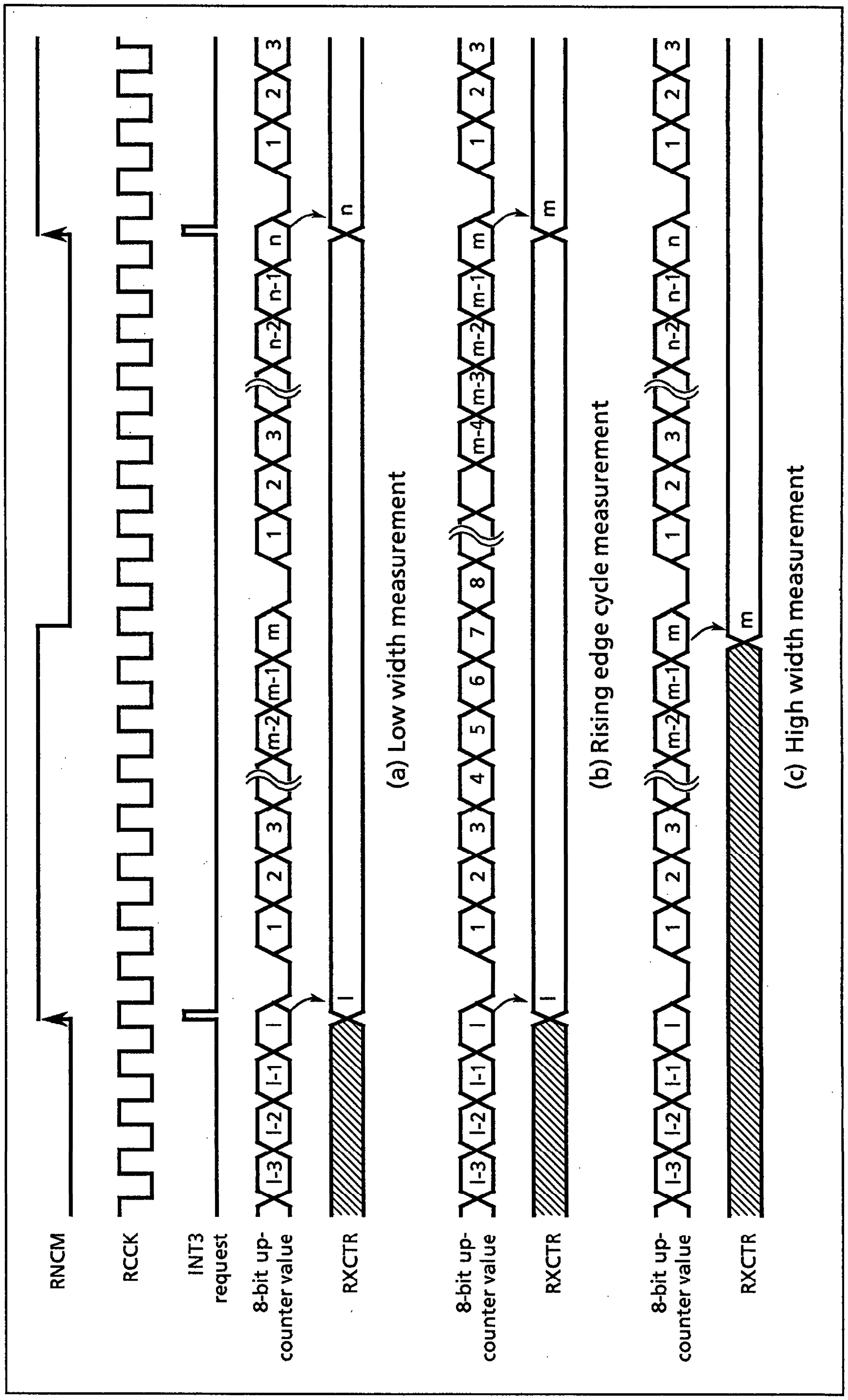


Figure 2-62. Rising Edge Interrupt Timing Chart (RPOLS = 0)

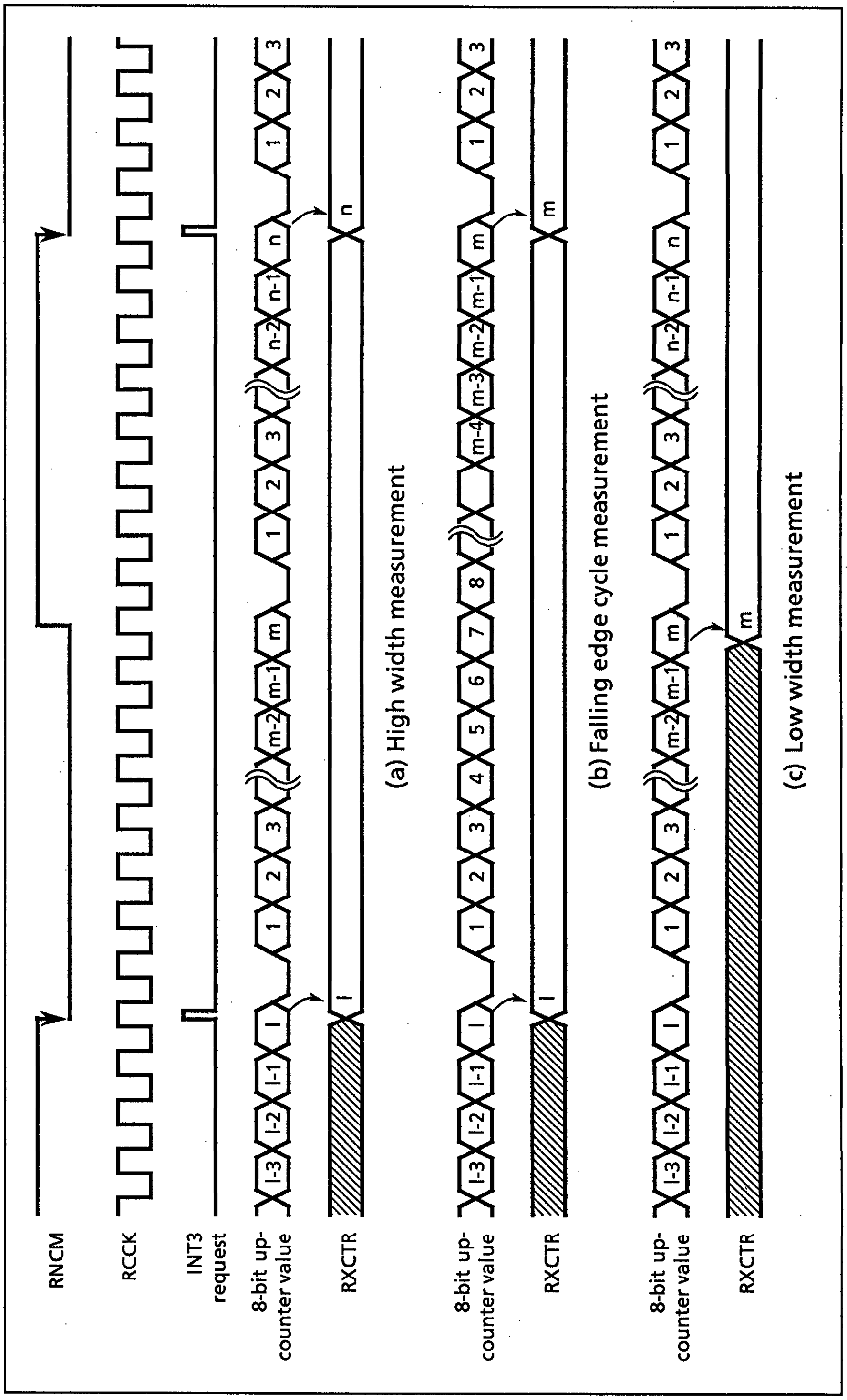


Figure 2-63. Falling Edge Interrupt Timing Chart (RPOLS = 0)

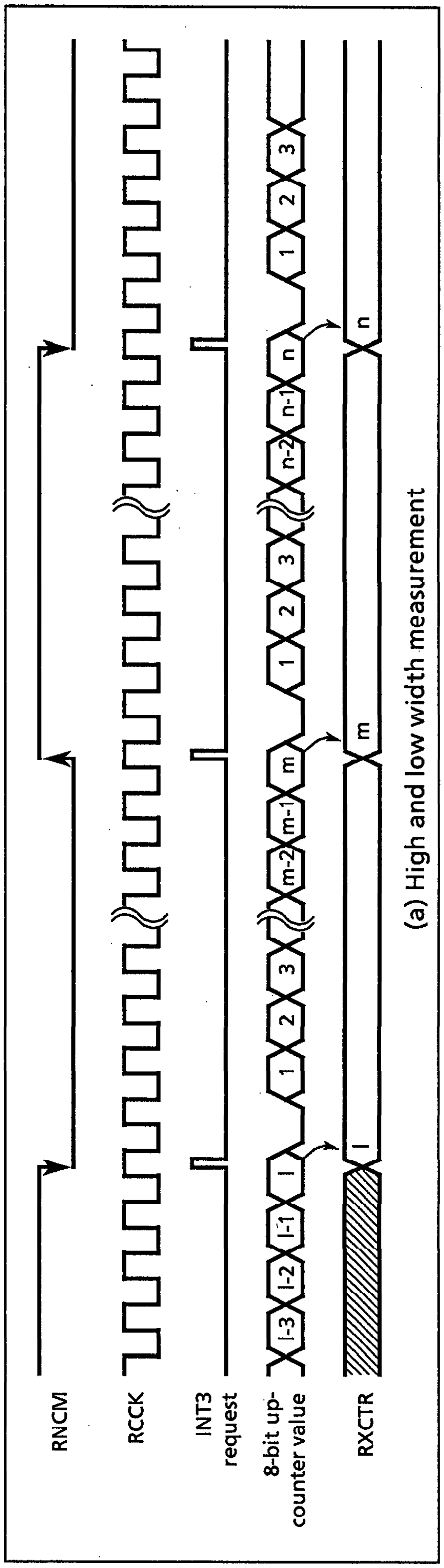


Figure 2-64. Rising / Falling Edge Interrupt Timing Chart

(2) 8-bit receive end interrupts and measurement modes

By determining one-cycle remote control signal or one-pulse width as one-bit data "0" or "1", an INT3 request is generated after 8-bit data is received. When "0" is determined, this means the upper four bits in the 8-bit up-counter have not reached the CREGA value. When "1" is determined, this means the upper four bits in the 8-bit up-counter have reached or exceeded the CREGA value. The 8-bit up-counter value is saved in RXCTR after one bit is determined. The determined data is saved, bit by bit, in RXDBR at the rising edge of the remote control signal (when RPOLS = 1, falling edge). The number of bits saved in RXDBR is counted by the receive bit counter and saved in RBCTM. RBCTM is set to "0001B" at the rising edge of the input (when RPOLS = 1, falling edge) after the INT3 request is generated.

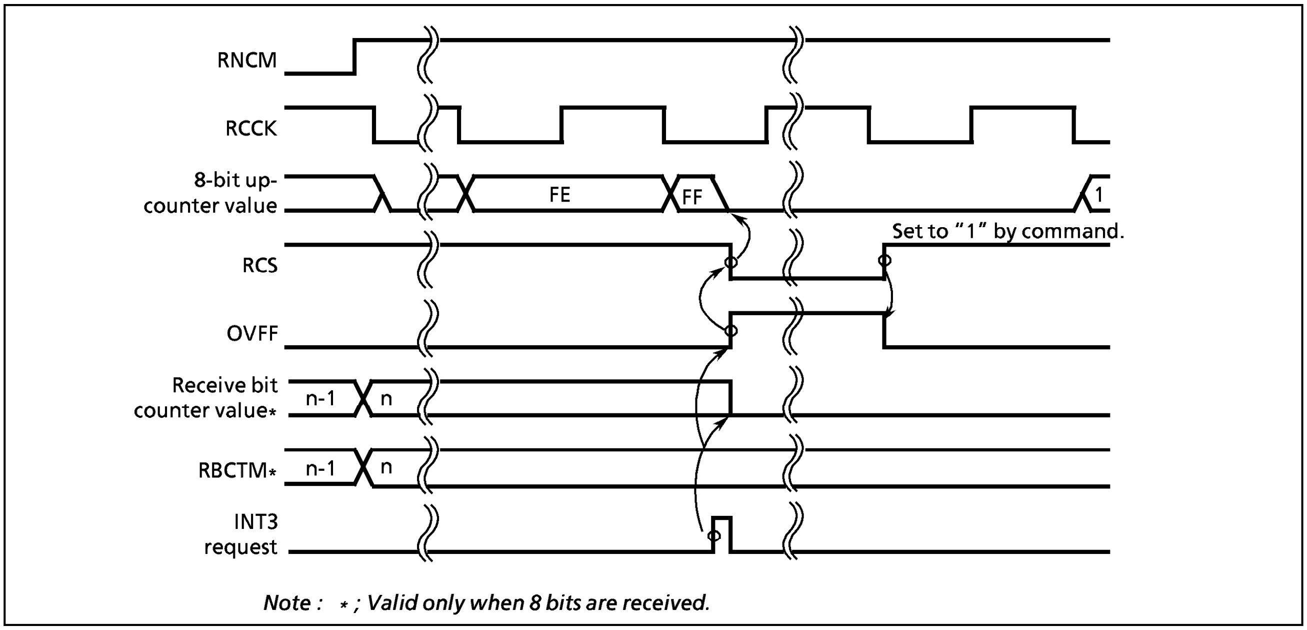
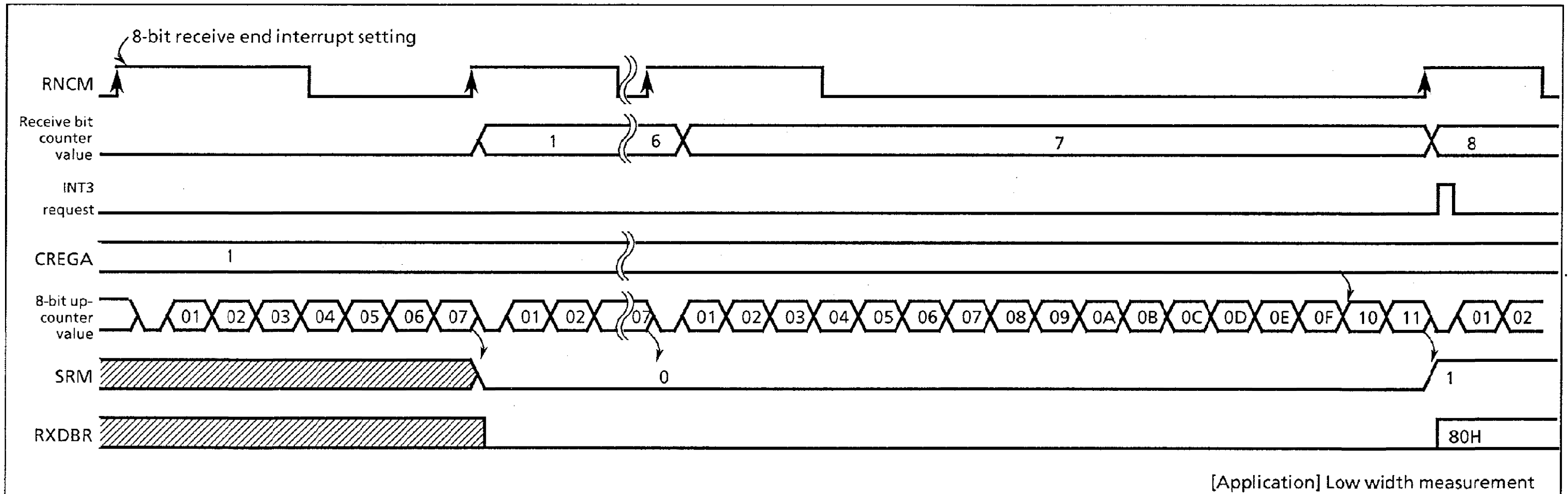


Figure2-65. Overflow Interrupt Timing Chart



(a) Rising edge cycle measurement

Figure 2-66. 8-bit Receive End Interrupt Timing Chart (RPOLS = 0)

Table 2-10. Count Clock for Remote Control Preprocessor Circuit

Count clock (RCCK)	at fc = 8 MHz	
	Resolution	Maximum setting time
$fc/2^6$ [Hz]	8 μs	2.048 ms
$fc/2^8$	32 μs	8.192 ms
$fc/2^{10}$	128 μs	32.768 ms
$fc/2^{12}$	512 μs	131.072 ms

2.11 8-bit A/D Converter (ADC)

The 87CM39/P39/S39 each have an 8-channel multiplexed-input 8-bit successive approximate type A/D converter with sample and hold.

2.11.1 Configuration

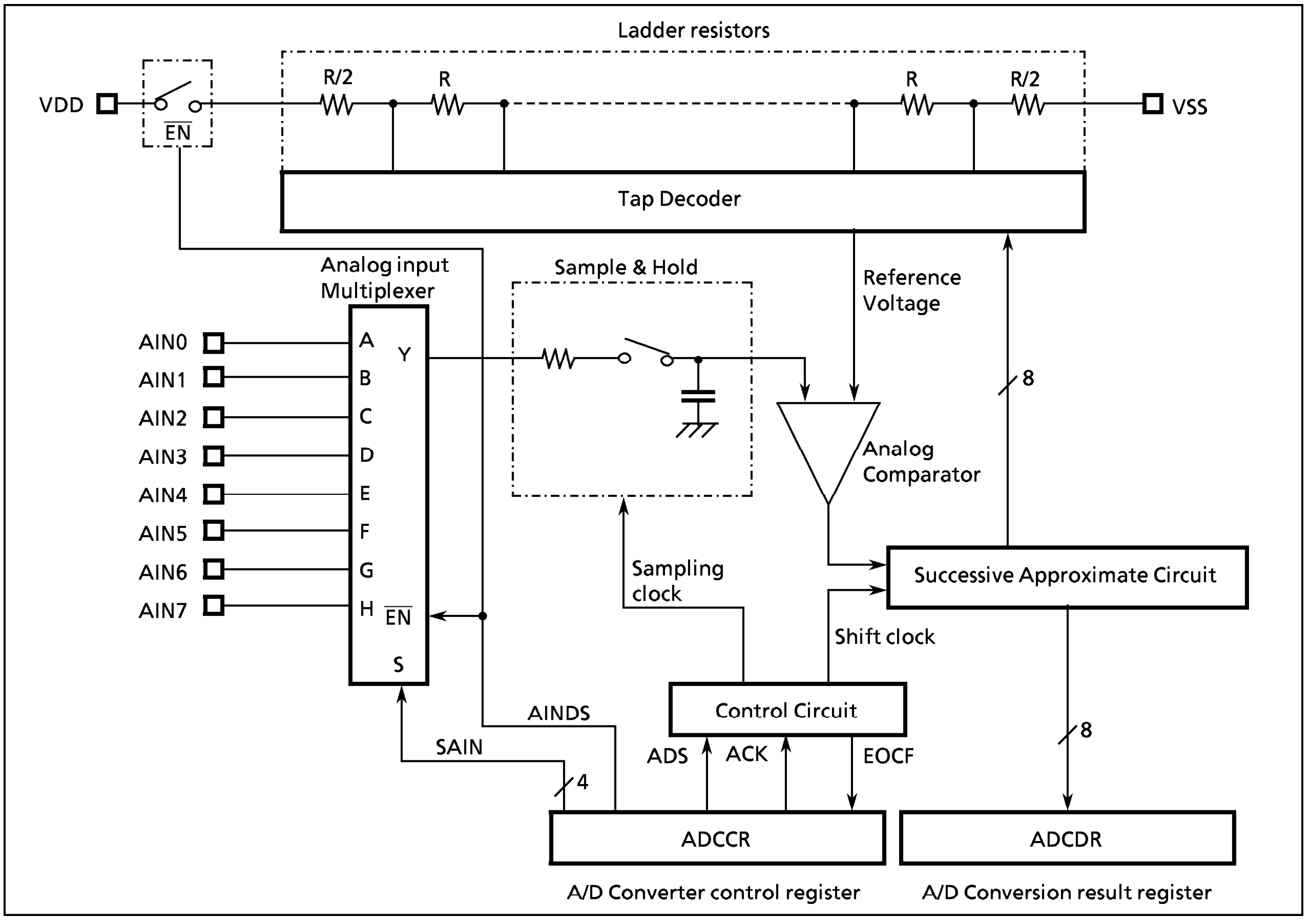


Figure 2-67. A/D Converter

2.11.2 Control

The A/D converter is controlled by an A/D converter control register (ADCCR) and a port P5/P6 input/output control register (P5CR/P6CR).

A/D Conversion Result Register

	7	6	5	4	3	2	1	0	
ADCDR ($000F_H$)									Read only

Figure 2-68. A/D Conversion Result Register

A/D Converter Control Register

	7	6	5	4	3	2	1	0	
ADCCR ($000E_H$)	EOCF	ADS	"0"	AINDS	SAIN				(Initial value : 00*0 0000)

SAIN	Analog input selection	0000 : AIN0 0001 : AIN1 0010 : AIN2 0011 : AIN3 0100 : AIN4 0101 : AIN5 0110 : AIN6 0111 : AIN7 011* : reserved 1 *** : reserved	R/W
AINDS	Analog input control	0 : Enable 1 : Disable	R/W
ADS	A/D conversion start	0 : – 1 : A/D conversion start	R/W
EOCF	End of A/D conversion flag	0 : Under conversion or Before conversion 1 : End of conversion	R

*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 2-69. A/D Converter Control Register

2.11.3 Operation

(1) Start of A/D conversion

First, set the corressponding P5CR / P6CR bit to "0" for analog input. Clear the AINDS (bit 4 in ADCCR) to "0" and select one of eight analog inputs AIN7-AIN0 with the SAIN (bit 3 to 0 in ADCCR).

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.

A/D conversion is started by setting the ADS (bit 6 in ADCCR) to "1".
Conversion is accomplished in 46 machine cycles (184/fc [s]).
The EOCF (bit 7 in ADCCR) is set to "1" at end of conversion.
When setting the ADS to "1" under A/D conversion, the A/D converter circuit is initialized and the A/D conversion try again from start.
The sampling of the analog input voltage is excuted at 4 machine cycles after setting the ADS to "1".

Note : The circuit of sample and hold is included in a condenser (12 pF (typ.)) through a register (5 kΩ (typ.)).
Therefore, until 4 machine cycles is over, this condenser must be charged.

(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.

(3) A/D conversion in STOP mode

When the MCU places in the STOP mode during the A/D conversion, the conversion is terminated and the ADCDR contents become indefinite.

However, if the STOP mode is started after the end of conversion (EOCF = 1), the ADCDR contents are held.

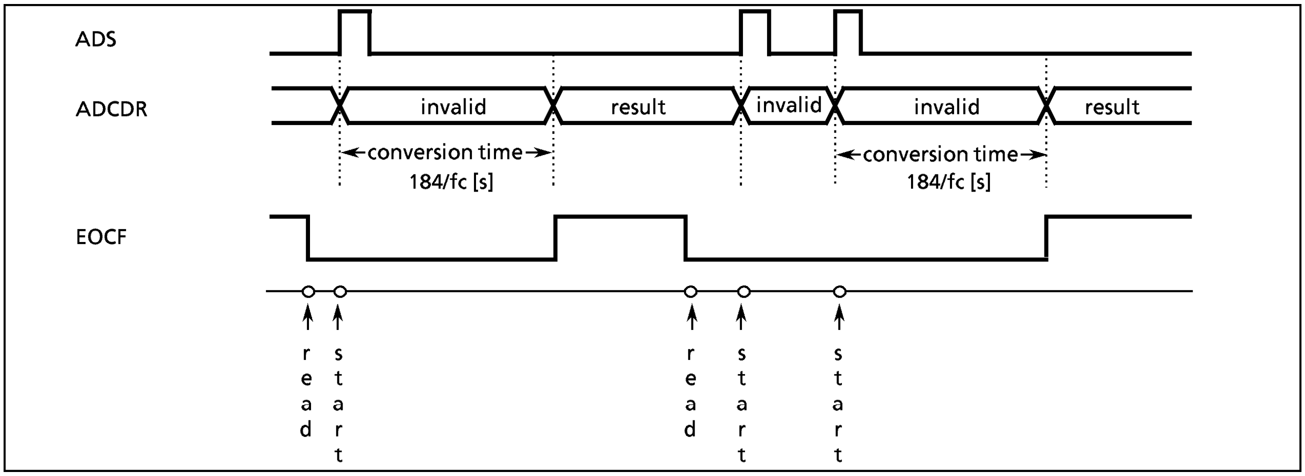


Figure 2-70. A/D Conversion Timing Chart

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)
```

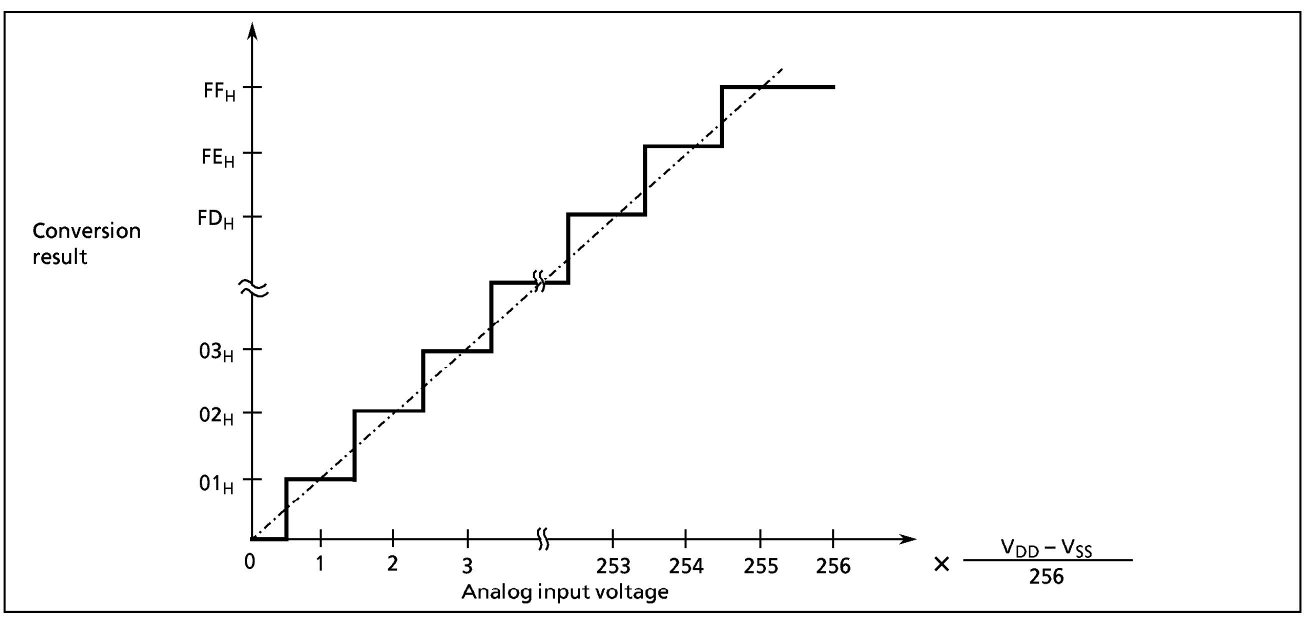


Figure2-71. Analog Input Voltage vs A/D Conversion Result (typ.)

2.12 Pulse Width Modulation Circuit Output

87CM39/P39/S39 has a 14-bit resolution pulse width modulation (PWM) channel and 9 7-bit resolution PWM channels. D/A converter output can easily be obtained by connecting an external low-pass filter. PWM outputs are multiplexed with general purpose I/O ports as; P40 ($\overline{\text{PWM0}}$) to P47 ($\overline{\text{PWM7}}$), P50 ($\overline{\text{PWM8}}$), P51 ($\overline{\text{PWM9}}$). When these ports are used as PWM outputs,the corresponding bits of P4, P5 output latches should be set to "1".

2.12.1 Configuration

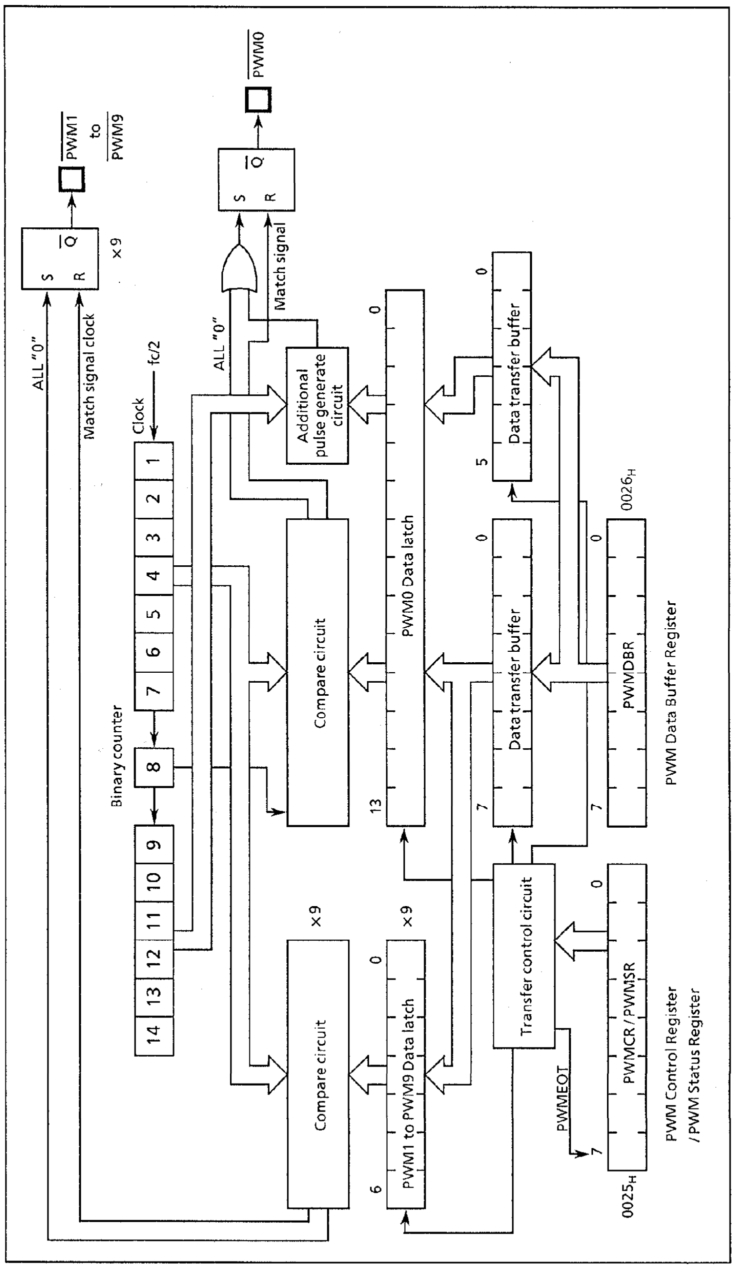


Figure 2-72. Pulse Width Modulation Circuit

2.12.2 PWM Output Wave Form

(1) $\overline{PWM0}$ output

This is 14-bit resolution PWM output and one period is $T_M = 2^{15}/fc$ [s].

The 8 high-order bits of the PWM data latch control the pulse width of the pulse output with a period of T_S ($T_S = T_M/64$), which is the sub-period of the $\overline{PWM0}$. When the 8-bit data are decimal n ($0 \leqq n \leqq 255$), this pulse width becomes $n \times t_0$, where $t_0 = 2/fc$.

The lower 6-bit of 14 bit data are used to control the generation of additional to wide pulse in each T_S period. When the 6-bit data are decimal m ($0 \leqq m \leqq 63$), the additional pulse is generated in each of m periods out of 64 periods contained in a T_M period. The relationship between the 6 bits data and the position of T_S period where the additional pulse is generated is shown in Table 2-11.

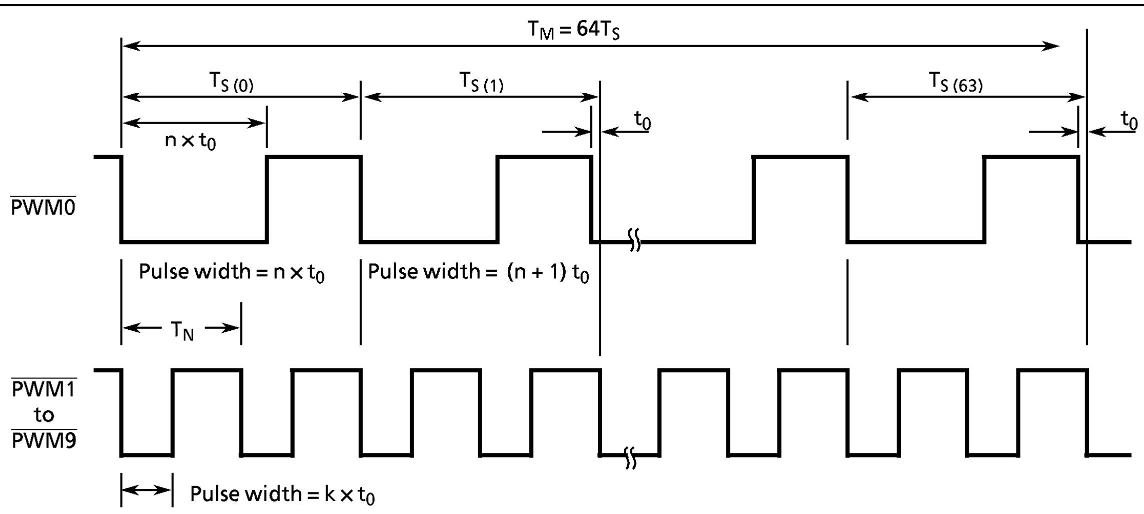
Table 2-11. Correspondence between 6 Bits Data and the Additional Pulse Generated T_S Period

Bit position of 6 bits data	Relative position of Ts where the output pulse is generated. (Number i of $T_{S\,(i)}$ is listed)
Bit 0	32
Bit 1	16, 48
Bit 2	8, 24, 40, 56
Bit 3	4, 12, 20, 28, 36, 44, 52, 60
Bit 4	2, 6, 10, 14, 18, 22, 26, 30, ·····, 58, 62
Bit 5	1, 3, 5, 7, 9, 11, 13, 15, 17, ·····, 59, 61, 63

Note : When the corresponding bit is "1", it is output.

(2) $\overline{PWM1}$ to $\overline{PWM9}$ outputs

These are 7-bit resolution PWM outputs and one period is $T_N = 2^8/fc$ [s]. When the 7-bit data are decimal k ($0 \leqq k \leqq 127$), the pulse width becomes $k \times t_0$. The wave form is illustrated in Figure 2-73.



Note 1 : It is shown to the additional pulse $T_{S\,(1)}$ and $T_{S\,(63)}$ of the $\overline{PWM0}$.
Note 2 : $T_S = 2^9/fc$ (64 μs, fc = 8 MHz), $T_N = 2^8/fc$ (32 μs, fc = 8 MHz)

Figure 2-73. PWM Output Wave Form

2.12.3 Control

PWM output is controlled by PWM Control Register (PWMCR) and PWM Data Buffer Register (PWMDBR). The status of transfer PWM data from PWMDBR to PWM data latch is read by PWMEOT of PWM status register (PWMSR).

PWM Control Register

PWMCR (0025H)

7	6	5	4	3	2	1	0
				PWMDLS			

(Initial value **** 0000)

PWMDLS	Selection of PWM data latch and request of data Transfer	0000 : Lower 6-bit of PWM0 0001 : 8 High-order bits of PWM0 0010 : PWM1 0011 : PWM2 0100 : PWM3 0101 : PWM4 0110 : PWM5 0111 : PWM6 1000 : PWM7 1001 : PWM8 1010 : PWM9 1011 : reserved 1100 : PWM Data Transfer Request 1101 : reserved 111* : reserved	write only

*Note : * ; don't care*

PWM Status Register

PWMSR (0025H)

7	6	5	4	3	2	1	0
PWMEOT	"1"	"1"	"1"	"1"	"1"	"1"	"1"

PWMEOT	End of PWM data transfer flag	0 : End of Transfer 1 : Under Transfer	read only

PWM Data Buffer Register

PWMDBR (0026H)

7	6	5	4	3	2	1	0

write only

Figure 2-74. PWM Control Register / PWM Status Register / PWM Data Buffer Register

(1) Programing of PWM Data

PWM output is controlled by writing the output data to data latches.
The sequence of writing the output data to data latch is shown as follows;

1. Write the channel number of PWM data latch to the PWMDLS.
2. Write PWM output data to the PWMDBR.
3. Write "$0C_H$" to the PWMCR.
   When transferring of the output data is completed, the PWMEOT becomes "0", indicating that the next data can be written. Do not write PWM data when the PWMEOT is "1" because write errors can occur in this case.

*Note : When writing the output data to **PWM0** data latch, write "$0C_H$" to the **PWMCR** after writing of the **14**-bit output data is completed.*

While the output data are being written to the data latch, the previously written data are being output. The maximum time from the point at which "$0C_H$" is written to the data latch until PWM output is switched is 2^{15}/fc [s] (4.096 ms, at fc = 8 MHz) for PWM0 output and 2^9/fc [s] (64 μs, at fc = 8 MHz) for PWM1 to PWM9 output.

Example : $\overline{\text{PWM0}}$ pin outputs a PWM wave form with a low-level of 32 μs width and no additional pulse.
$\overline{\text{PWM1}}$ pin outputs a PWM wave form with a low-level of 16 μs width.
$\overline{\text{PWM2}}$ pin outputs a PWM wave form with a low-level of 8 μs width.

Note : at fc = 8 MHz

```
        LD      (PWMCR),  00H    ;  Select lower 6-bit of PWM0
        LD      (PWMDBR),  00H   ;  No additional pulse
        LD      (PWMCR),  01H    ;  Select 8 high-order bits of PWM0
        LD      (PWMDBR),  80H   ;  32 μs ÷ 2/fc = 80 H
        LD      (PWMCR),  0CH    ;  Request PWM Data Transfer
WAIT0 : TEST    (PWMSR). 7       ;  PWMEOT = 0?
        JRS     F,  WAIT0
        LD      (PWMCR),  02H    ;  Select PWM1
        LD      (PWMDBR),  40H   ;  16 μs ÷ 2/fc = 40 H
        LD      (PWMCR),  0CH    ;  Request PWM Data Transfer
WAIT1 : TEST    (PWMSR).  7      ;  PWMEOT = 0?
        JRS     F,  WAIT1
        LD      (PWMCR),  03H    ;  Select PWM2
        LD      (PWMDBR),  20H   ;  8 μs ÷ 2/fc = 20 H
        LD      (PWMCR),  0CH    ;  Request PWM Data Transfer
WAIT2 : TEST    (PWMSR). 7       ;  PWMEOT = 0?
        JRS     F,  WAIT2
```

2.13 Test Video Signal Output for Adjusting TV Screen

TMP87CM39/P39/S39 have a built-in test video signal output circuit to output necessary signal for TV screen adjustment.

Mode : NTSC (at fc = 8.056 MHz)
PAL (at fc = 8.000 MHz)
Picture pattern : Total eight types, Monochromatic inversion possible
Output format : Three states (H, L, Hi-Z) output
Comp.Sync duration time L output
Black level / Pedestal duration time Hi-Z output
White level duration time H output

2.13.1 Configuration

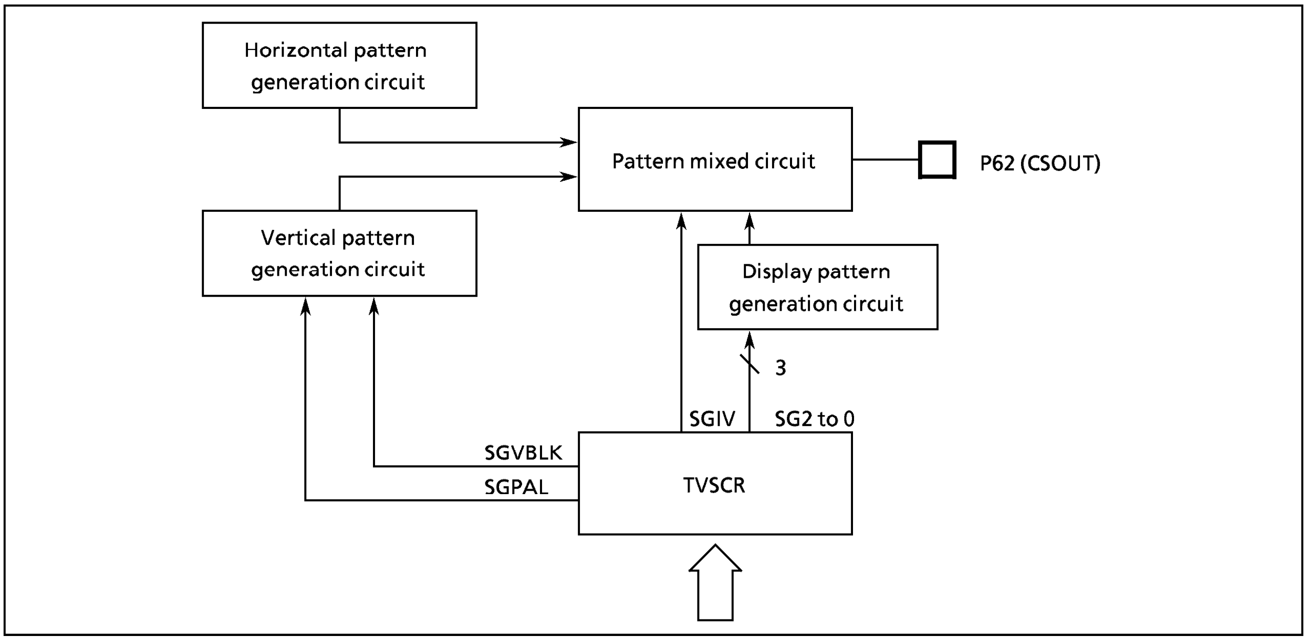


Figure 2-75. Test Video Signal Output Circuit

2.13.2 Control

The test video signal output circuit can be controlled with the signal control register.

TVSCR ($0FB0_H$) (Initial value 0000 0000)

7	6	5	4	3	2	1	0
SGEN	SGVBLK	SGPAL	SGIV	SGCHS	SGPAT		

SGEN	function selection	0 : disable 1 : enable	Write only
SGVBLK	Picture signal for VBLK duration time	0 : Output 1 : No output	
SGPAL	PAL/NTSC selection	0 : NTSC 1 : PAL	
SGIV	Pattern monochromatic inversion	0 : No inversion 1 : Inversion	
SGCHS	OSD synchronous signal selection	0 : Port 1 : Pseudo signal circuit	
SGPAT	Display pattern	000 : Black on the whole screen 001 : White on the whole screen 010 : Cross hatch 011 : Cross dot pattern 100 : Cross bar 101 : White on the upper side / Black on the lower side 110 : H signal pattern 111 : H resolution pattern	

Figure 2-76. Signal Control Register

2.13.3 Functions

Test video signal output is to generate monochromatic picture signal output to take easily the necessary tests such as TV screen white adjustment and screen distortion amplitude adjustment implemented on the final manufacturing process of a TV receiver set.

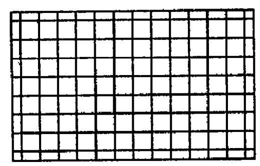
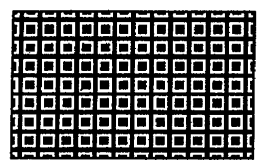
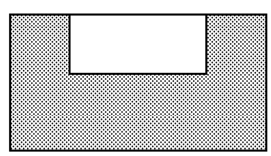
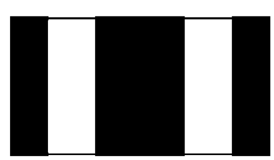
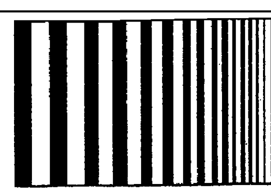
Display pattern	TV screen
000 (Black on the whole surface)	
001 (White on the whole surface)	
010 (Cross hatch)	
011 (Cross dot)	
100 (Cross bar)	
101 (White on the upper side / Black on the lower side)	
110 (H signal pattern)	
111 (H resolution pattern)	

Figure 2-77. Display Pattern and TV Screen

There are three states of the output to generate picture signal with the external circuit of the resistance divided voltage.

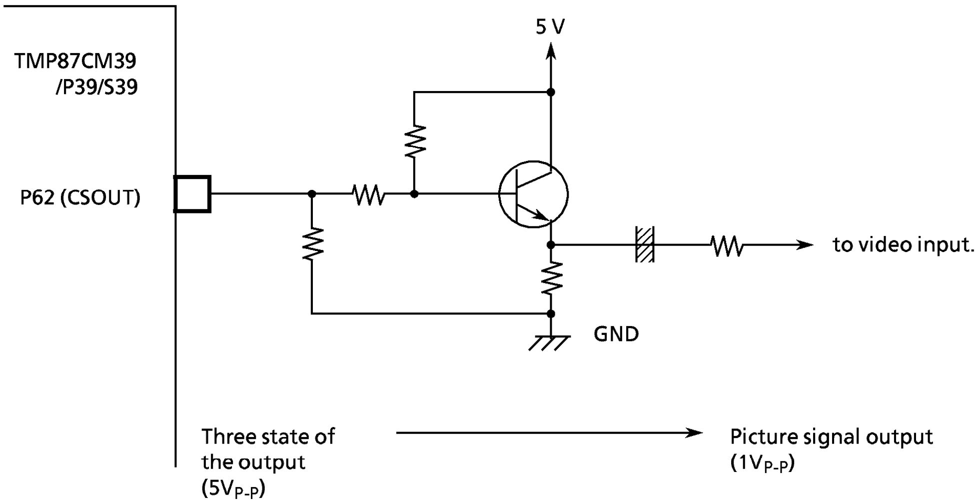


Figure 2-78. Example of Picture Output Generation

2.14 On-Screen Display (OSD) Circuit

The TMP87CM39/P39/S39 features a built-in on-screen display circuit used to display characters and symbols on the TV screen. 288 characters in any of 256 character fonts can be displayed in 24 characters x 12 rows.

OSD circuit functions are as follows:

①	Number of character fonts 256 (including blank character)			
②	Number of display characters 288 (24 characters x 12 rows)			
③	Composition of a character	14 x 18 dots		
④	Character sizes	3 (selectable line by line)		
⑤	Display colors	Character colors	:	8 (selectable character by character)
		Fringe color	:	8 (selectable page by page)
		Background color	:	8 (selectable page by page)
⑥	Fringing function (for large, middle, and small characters)			
⑦	Smoothing function (for large and middle characters)			
⑧	Display position	horizontal : 128 steps ;	vertical :	256 steps
⑨	Full-raster blanking function			
⑩	Blinking function			
⑪	Reverse function			
⑫	Reverse Blinking function			
⑬	Window function			

2.14.1 Configuration

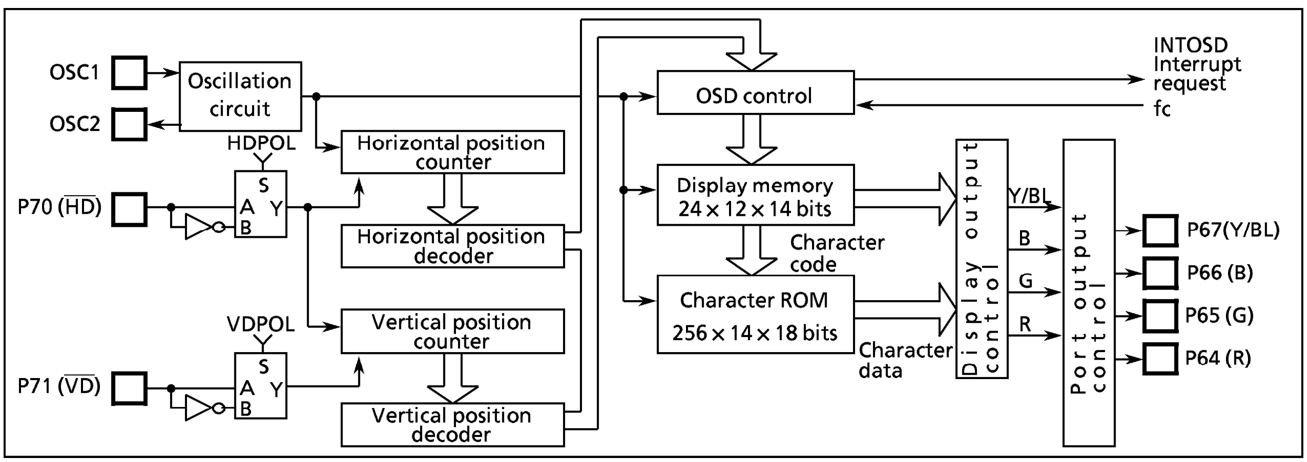


Figure 2-79. OSD Circuit

2.14.2 Character ROM and Display Memory

(1) Character ROM

The character ROM contains 256 character fonts. The user can set fonts as desired. The character ROM consists of 256 characters in 14 x 18 dots (character codes 00_H to FF_H). Each dot corresponds to one bit in the character ROM. When a bit in the character ROM is set to "1", the corresponding dot is displayed; if set to "0", the dot is not displayed. The start address in the character ROM corresponding to a character code is determined by the following expression:

$$\text{Start address in character ROM} = \text{CRA} \times 40_H + 4000_H$$

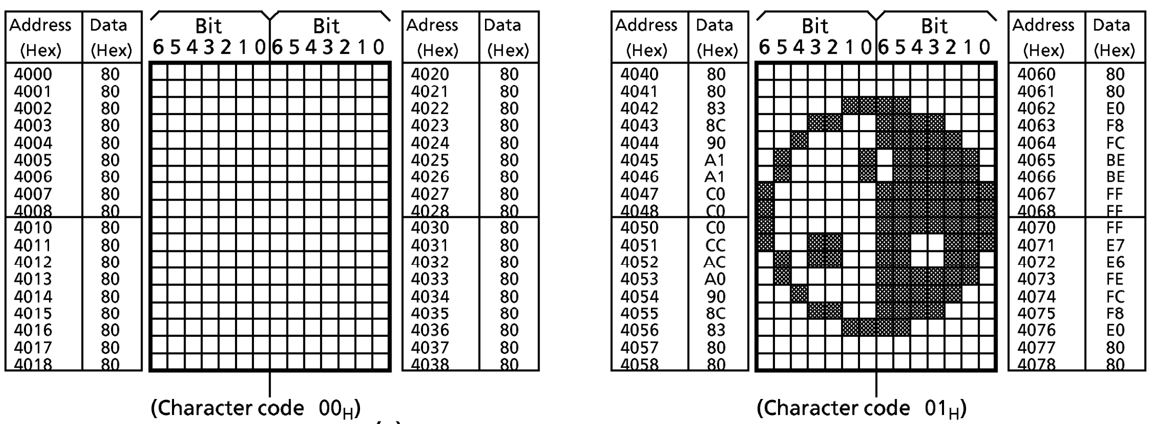
Since character code 00_H is used as blank character, the character font for this character code cannot be changed. Write "0" in the data of character code 00_H.

Set all unused bits (bit 7 with 0_H to 8_H in the lower 4-bit of an address) to "1" and write the data "FF_H" to all unused address (the lower 4-bit of an address are 9_H to F_H) in character ROM.

Figure 2-80. (a) shows an example of the character font configuration for the character code 00_H and 01_H, together with the ROM addresses and data.

Figure 2-80. (b) shows the character ROM dump list for these 2 character fonts .

> *Note 1 : CRA ; Character code (00_H to FF_H)*
> *Note 2 : A data can not be read from the character ROM by software.*
> *Note 3 : When ordering a mask, load the data to character ROM at addresses 4000_H to $7FFF_H$.*



(a) Character font configuration

```
4000/ 80 80 80 80 80 80 80 80 80 FF FF FF FF FF FF FF
4010/ 80 80 80 80 80 80 80 80 80 FF FF FF FF FF FF FF
4020/ 80 80 80 80 80 80 80 80 80 FF FF FF FF FF FF FF
4030/ 80 80 80 80 80 80 80 80 80 FF FF FF FF FF FF FF
4040/ 80 80 83 8C 90 A1 A1 C0 C0 FF FF FF FF FF FF FF
4050/ C0 CC AC A0 90 8C 83 80 80 FF FF FF FF FF FF FF
4060/ 80 80 E0 F8 FC BE BE FF FF FF FF FF FF FF FF FF
4070/ FF E7 E6 FE FC F8 E0 80 80 FF FF FF FF FF FF FF
```

(b) ROM dump list

Figure 2-80. Character Font Configuration and ROM Dump List

(2) Display memory

Each character out of the 288 characters displayed in 24 characters x 12 rows consists of 14 bits in the display memory. Five data items are written to the display memory: character code, color data, blinking specification, reverse specification, and reverse blinking specification. The display memory contents become unstable after the reset operation is released.

There are two modes for writing display data to the display memory. One mode is for writing all display data (character code, color data, blinking specification, reverse specification, and reverse blinking specification) simultaneously. The other mode is for changing either character code or character ornamentation data (color data, blinking specification, reverse specification, and reverse blinking specification). How the display data is written to the display memory is described in section 2.14.3 (19).

Display memory configuration

- Character code specification register (8 bits) ··· CRA7 TO 0
- Color data specification register (3 bits) ··· RDT / GDT / BDT
- Blinking specification register (1 bit) ··· BLF
- Reverse specification register (1 bit) ··· RVF
- Reverse blinking specification register (1 bit) ··· RBF

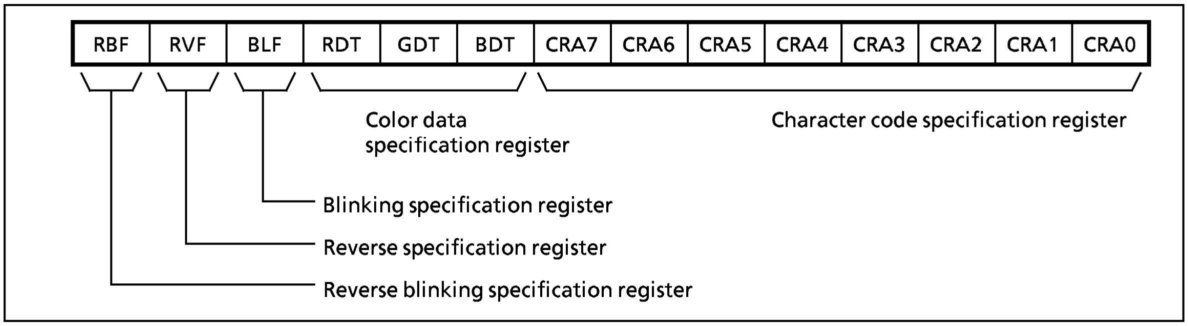


Figure 2-81. Display Memory Bit Configuration

Row \ Character	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	000	001	002	003	004	005	006	007	008	009	00A	00B	00C	00D	00E	00F	010	011	012	013	014	015	016	017
2	020	021	022	023	024	025	026	027	028	029	02A	02B	02C	02D	02E	02F	030	031	032	033	034	035	036	037
3	040	041																						057
4	060	061																						077
5	080	081																						097
6	0A0	0A1																						0B7
7	0C0	0C1																						0D7
8	0E0	0E1																						0F7
9	100	101																						117
10	120	121																						137
11	140	141																						157
12	160	161																					176	177

Note: Numerals in the table indicate (hexadecimal) addresses in the display memory.

Figure 2-82. Display Memory Address Configuration

2.14.3 OSD circuit control

The OSD circuit is controled by using the OSD control registers assigned to addresses $0F80_H$ to $0F9B_H$ in the data buffer register (DBR). For write to or read from the OSD control registers, see section 2.14.3 (20). The OSD control registers are used to set display start position, display character ornamentations (that is, fringing, smoothing, color data, character size, and etc.), display memory addresses, character codes, and etc.

After all settings are complete, setting the display on-off control bit, EDISP (bit 0 in ORDON) to 1 enables display (starts display). Setting EDISP to 0 disables display (halts display).

Note : The contents of OSD control registers are not initialized in STOP mode.

(1) Display position

The horizontal display start position can be set in 128 steps. The vertical display start positions can be specified for each line using 256 steps. The horizontal display start position is set with OSD control registers HS16 to HS10 (bit 6 to 0 in ORHS1). The vertical display start position of the line 1 is set with VS17 to VS10 (in ORVS1). The vertical display start position of the line 2 to 12 are determined by setting VS27 to VS20 ... VS127 to VS120 (ORVS2 to ORVS12) in the same way.

Horizontal display start position
Specification unit : Display page
Specification steps : 128
Specification horizontal display start position : Line 1 to 12 : HS16 to HS10

When FORS is "0" (Normal mode)
$HS1 = (HS16\ to\ HS10)_H \times 2T_{OSC} + 11T_{OSC}$ (Line1 to 12)

When FORS is "1" (Double frequency mode)
$HS1 = (HS16\ to\ HS10)_H \times 2T_{OSC} + 6.5T_{OSC}$ (Line1 to 12)

Note : T_{OSC} ; One cycle of OSC oscillation

Vertical display start position
Specification unit : Line
Specification steps : 256
Specification vertical display start position : Line1 : VS17 to VS10
Line2 : VS27 to VS20
·
Line12 : VS127 to VS120

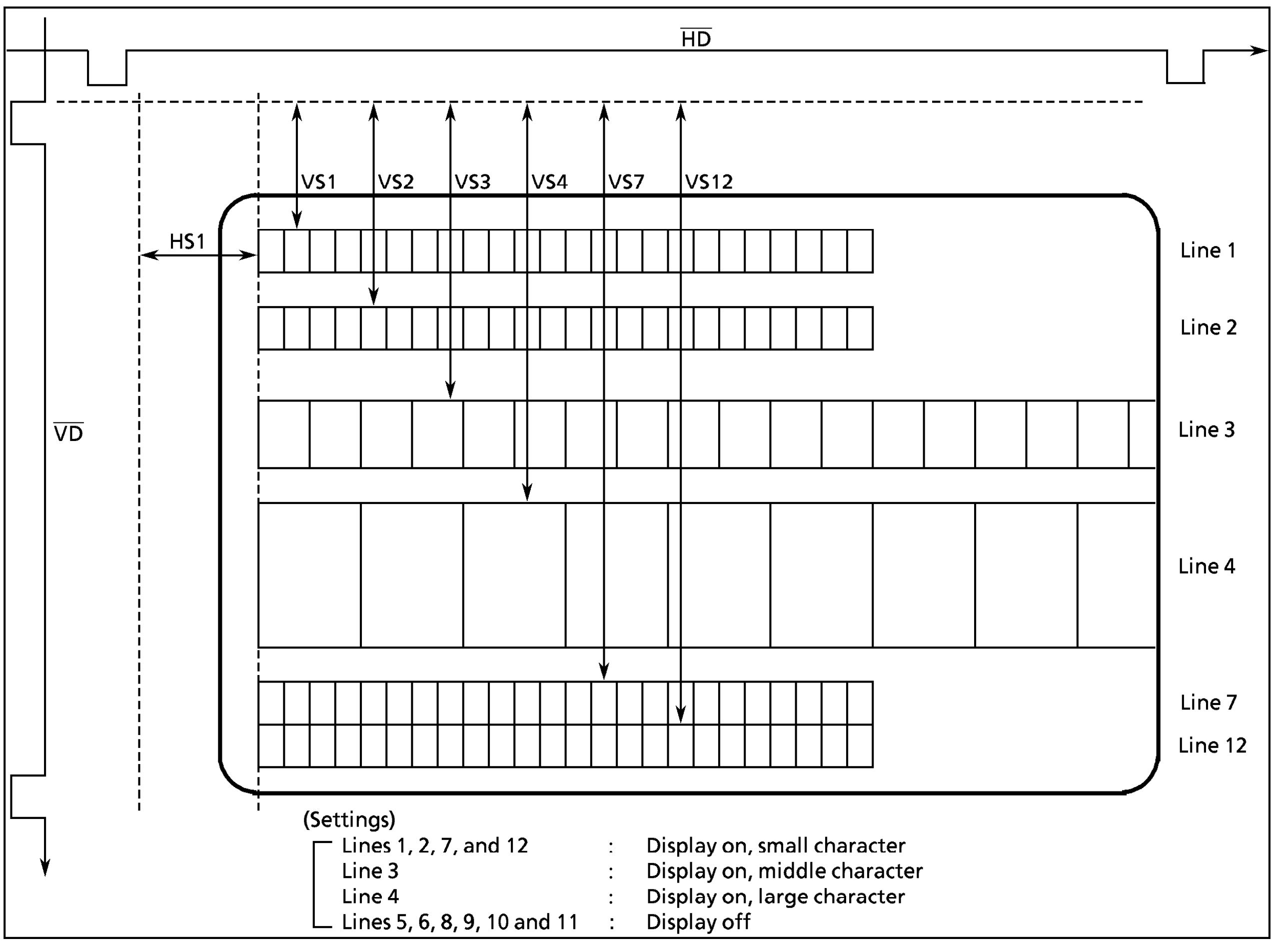
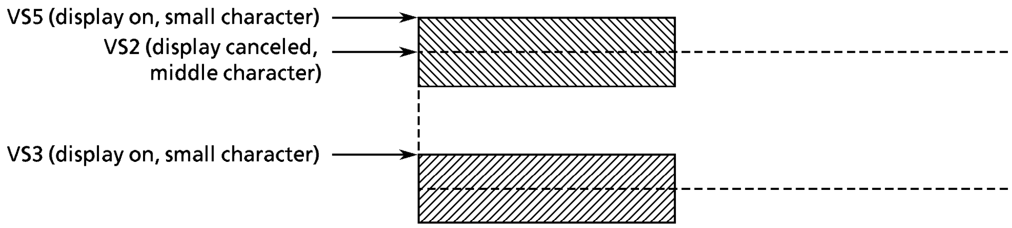


Figure 2-83. TV Screen Image

When VDSMD is "0" (Normal mode)
Line n : $VSn = (VSn7 \text{ to } VSn0)_H \times 2T_{HD}$ (n ; 1 to 12)

When VDSMD is "1" (Double scan mode)
Line n : $VSn = (VSn7 \text{ to } VSn0)_H \times 4T_{HD}$ (n ; 1 to 12)

Note1 : T_{HD} ; One cycle of $\overline{HD}$ signal
Note2 : If display lines are overlapped each other, previous display line is enabled and next line is disabled. Set the vertical display start position not to overlap display lines.



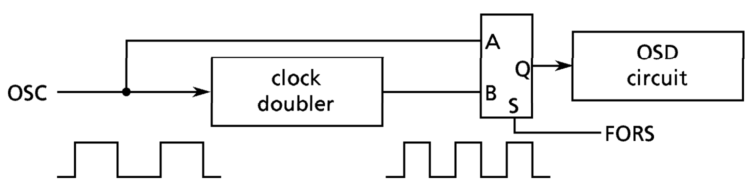
Occasion of overlapping

Note3 : The line which is displayed off is managed as a small size character line. It is recommendable that its vertical display start position should be set out of TV screen.
Note4 : Transfer the contents of vertical display start position registers into OSD circuit before the position of the scanning line coincides with their own vertical display start position

(2) Double scan mode
The double scan mode is used to handle non-interlaced scanning TV. When double scan mode is enabled, the vertical display counter increases every 4 scan lines and a vertical dot size is double. This function is enabled by setting VDSMD (bit 3 in ORETC) in the OSD control register to "1".

Scan mode select register (1 bit) ··· VDSMD (bit 3 in ORETC)
"0" ··· Normal mode
"1" ··· Double scan mode

(3) Double frequency mode
The Double frequency mode is used to display OSD by the OSC frequency doubled. When this function is enabled, the clock which is doubled by a clock doubler is inputted into an OSD circuit.



This function is ebabled by setting FORS (bit 4 in ORDON) in the OSD control register to "1".

OSC frequency select register (1 bit) ··· FORS (bit 4 in ORDON)
"0" ··· Normal mode
"1" ··· Double frequency mode

(4) $\overline{\text{HD}}$/$\overline{\text{VD}}$ signal polarity control function

This function is used to select the polarity of input signal via $\overline{\text{HD}}$/$\overline{\text{VD}}$ pin.

$\overline{\text{HD}}$ signal polarity select register (1 bit) ··· HDPOL (bit 0 in ORPOL)
"0" ··· Non-invert input signal via $\overline{\text{HD}}$ pin.
"1" ··· Invert input signal via $\overline{\text{HD}}$ pin.

$\overline{\text{VD}}$ signal polarity select register (1 bit) ··· VDPOL (bit 1 in ORPOL)
"0" ··· Non-invert input signal via $\overline{\text{VD}}$ pin.
"1" ··· Invert input signal via $\overline{\text{VD}}$ pin.

(5) Character sizes and display on / off

Character size can be selected line by line from 3 sizes. And display on / off also can be set line by line. Small, middle and large character size and display on / off can be set with OSD control registers CS11, CS10...CS121, CS120 (ORCS4, ORCS8, and ORCS12) in the OSD control registers.

Character sizes : 3 sizes (Small, middle and large)
Character size and display on / off specification unit : Line
Character size select/display on / off register (2 bits x 12)
Line 1: CS11 and CS10
Line 2: CS21 and CS20
: :
Line 12: CS121 and CS120

Table 2-12. Character Size and Display On / Off Specifications (n ; 1 to 12)

CSn1	CSn0	Character size	Display on/off
1	1	Small	On
1	0	Middle	On
0	1	Large	On
0	0	–	Off

Note : The line which is displayed off is managed as a small character size line by the overlap of vertical display start position, the display line counter function, and etc.

Table 2-13. Dot and Character Sizes

		VDSMD = 0 (Normal mode)		VDSMD = 1 (Double scan mode)	
		Dot size	Character size	Dot size	Character size
FORS = 0 (normal mode)	Small	1 T_{OSC} × 1 T_{HD}	14 T_{OSC} × 18 T_{HD}	1 T_{OSC} × 2 T_{HD}	14 T_{OSC} × 36 T_{HD}
	Middle	2 T_{OSC} × 2 T_{HD}	28 T_{OSC} × 36 T_{HD}	2 T_{OSC} × 4 T_{HD}	28 T_{OSC} × 72 T_{HD}
	Large	4 T_{OSC} × 4 T_{HD}	56 T_{OSC} × 72 T_{HD}	4 T_{OSC} × 8 T_{HD}	56 T_{OSC} × 144 T_{HD}
FORS = 1 (double frequency mode)	Small	0.5 T_{OSC} × 1 T_{HD}	7 T_{OSC} × 18 T_{HD}	0.5 T_{OSC} × 2 T_{HD}	7 T_{OSC} × 36 T_{HD}
	Middle	1 T_{OSC} × 2 T_{HD}	14 T_{OSC} × 36 T_{HD}	1 T_{OSC} × 4 T_{HD}	14 T_{OSC} × 72 T_{HD}
	Large	2 T_{OSC} × 4 T_{HD}	28 T_{OSC} × 72 T_{HD}	2 T_{OSC} × 8 T_{HD}	28 T_{OSC} × 144 T_{HD}

Note : T_{OSC}; One cycle of OSC oscillation T_{HD}; One cycle of $\overline{HD}$ signal

(6) Smoothing function

The smoothing function is used to make characters look smooth. Enabling smoothing displays 1/4 dot between two dots connecting corner to corner within a character. Small size character can not be enabled smoothing. Smoothing is enabled by setting ESMZ (bit 4 in ORETC) in the OSD control register to "1".

Smoothing specification unit: Display page
Smoothing specification register (1 bit) ··· ESMZ (bit 4 in ORETC)
"0" ··· Disable smoothing
"1" ··· Enable smoothing

(7) Fringing function

The fringing function is used to display a character with a fringe width is 1/2 dot in a different color from that of the character. For small characters, fringe width is 1 dot. When a character is displayed with the maximum of 14 vertical dots and 18 horizontal dots, the fringe exceeds right and left, top, and bottom of the character display area. The exceeded fringe can be displayed; however, display characters have higher priority to fringe horizontally.
Fringing is enabled for each line by setting EFR1 to EFR12 (OREFR and bit 3 to 0 in ORP6DS) in the OSD control register to "1".
A color for fringe is specified common to all lines using OSD control registers, RFDT, GFDT, and BFDT, (bit 2 to 0 in ORBK).

Fringing specification unit: Line
Fringing enable register (1 bit x 12) ··· EFRn (n: 1 to 12) (OREFR and bit 3 to 0 in ORP6DS)
"0" ··· Disable fringing
"1" ··· Enable fringing

Fringe color specification unit: Display page
Fringe color register (3 bits) ··· RFDT, GFDT, BFDT (bit 2 to 0 in ORBK)

Note : When a display line is enabled fringing function, its vertical size is increased by one dot (by two dots when its character size is small) independent of its character font. Therefore, when a vertical display start position is specified to no space between the lines, the display line which is overlapped with increasing dot (s) is canceled.

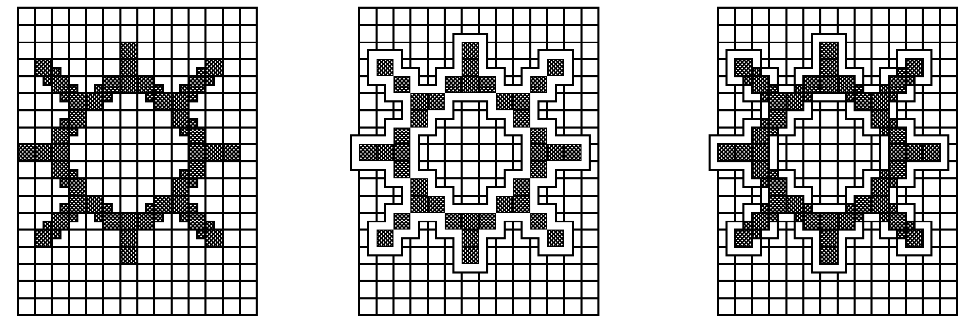


Figure 2-84. Smoothing / Fringing / Priority of Smoothing and Fringing

(8) Background color function

Background color function is used to color the entire background for the character area (14 × 18 dots). Except the character area whose character code is 00_H.

This function is specified for each display page by setting EBKGD (bit 7 in ORBK) in the OSD control register to "1".

A background color is specified for each display page by setting RBDT, GBDT and BBDT (bit 5 to 3 in ORBK) in the OSD control registers. A color specification is same as them for full-raster blanking.

Background specification unit: Display page
Background enable register (1 bit) ··· EBKGD (bit 7 in ORBK)
"0" ··· Disable background
"1" ··· Enable background

Background color specification unit: Display page
Background color specification registers (3 bits) ··· RBDT, GBDT, BBDT (bit 5 to 3 in ORBK)

Note: When the background color function is used, the blank character (Code 00_H) can not be used as the first character on the fringing line.

(9) Full-raster blanking function

Full-raster blanking function is used to color the entire background for the display area (TV screen). When using the full-raster blanking function, set YBLCS (bit7 in ORETC) to "1", output BL signal from Y/BL pin, because Y signal cannot delete whole display page from video signal.

This function is specified for each display page by setting EXBL (bit 6 in ORBK) in the OSD register to "1". Color specification is same as them for background color.

Full-raster blanking specification unit: Display page
Full-raster blanking enable register (1 bit) ··· EXBL (bit 6 in ORBK)
"0" ··· Disable full-raster blanking
"1" ··· Enable full-raster blanking

Full-raster blanking color specification registers (3 bits) ··· RBDT, GBDT, BBDT (bit 5 to 3 in ORBK)

(10) Reverse function

This function is used to reverse the background and character colors. However, when fringing is specified, the fringe color does not change. If background function is not used, background color is black.
Reverse function is enabled by setting RVF (bit 4 in ORDSN) in the OSD control register to "1".

Reverse specification: Character
Reverse enable register (1 bit) ··· RVF (bit 4 in ORDSN)
"0" ··· Disable reverse
"1" ··· Enable reverse

(11) Reverse blinking function

Reverse blinking function is used to reverse the background and character colors.
When RBMF is "1", characters specified for blinking by RBF are reversed the background and character colors. However, when fringing is specified, the fringe color does not change.

Reverse blinking specification unit: Character
Reverse blinking specification register (1 bit) ··· RBF (bit 5 in ORDSN)
"0" ··· No reverse blinking
"1" ··· Reverse blinking
Reverse blinking master specification register (1 bit) ··· RBMF (bit 5 in ORETC)
"0" ··· Disable reverse blinking
"1" ··· Enable reverse blinking
(Characters whose RBF is set to "1" are reversed the background and character colors.)

Table 2-14. Display Mode

RBF	RVF	RBMF	Display
0	0	*	Normal
0	1	*	Reverse
1	0	0	Normal
		1	Reverse
1	1	*	reserved

* ; don't care

(12) Blinking function

Blinking function is used to blink display characters.
When BKMF is "1", characters specified for blinking by BLF are not displayed.

Blinking specification unit : Character
Blinking specification register (1 bit) ··· BLF (bit 3 in ORDSN)
"0" ··· No blinking
"1" ··· Blinking
Blinking master specification register (1 bit) ··· BKMF (bit 6 in ORETC)
"0" ··· Disable blinking
"1" ··· Enable blinking (Characters whose BLF are set to "1" are not displayed.)

(13) Character

Characters: 256 (including blank character)

Character specification register (8 bits) ··· CRA7 to CRA0 (bit 7 to 0 in ORCRA)
Character code "00_H" ··· Blank character
Character code "01_H" to "FF_H" ··· User programmable by character ROM

(14) Character color

Character colors: 8
Character color specification unit: Character
Character color specification register (3 bits) ··· RDT / GDT / BDT (bit 2 to 0 in ORDSN)

Table 2-15. Character Color

RDT	GDT	BDT	Character Color
0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White

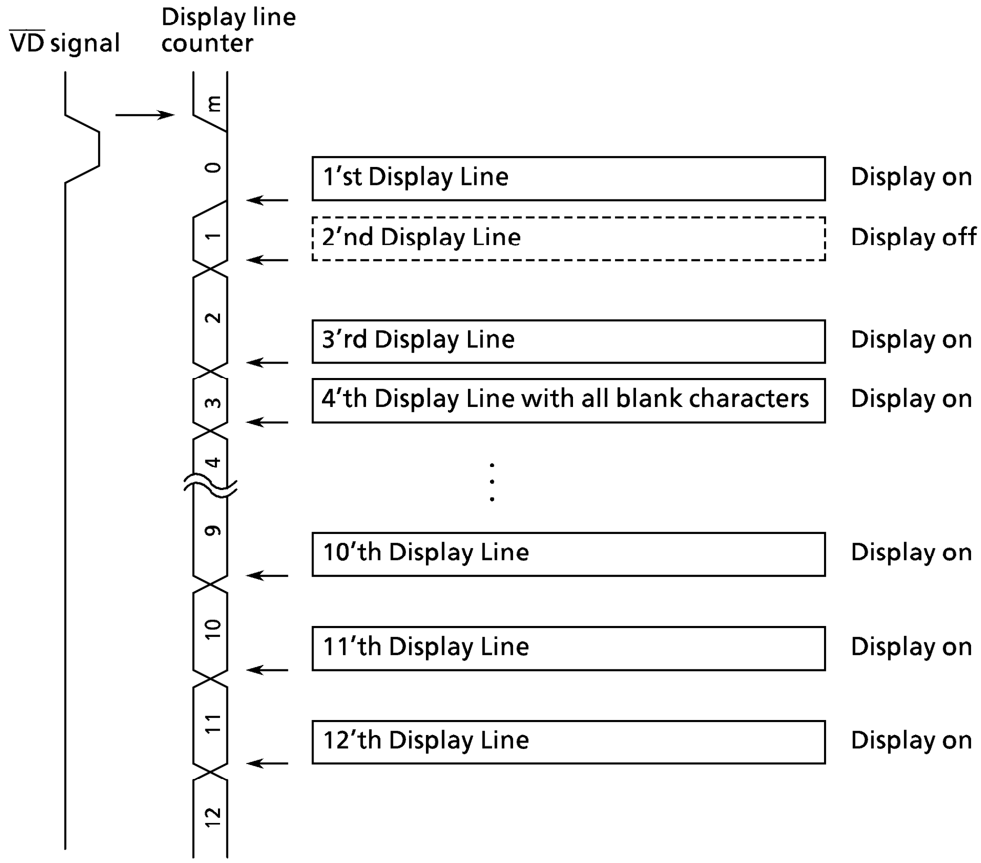
(15) OSD interrupt

1) Display line counter

The display line counter indicates number of display line(s) by OSD circuit on the TV screen. The display line counter is a 4-bit counter which is initialized to "0" by the falling edge of the $\overline{VD}$ signal and which increments when last scanning of each display line is completed (falling edge of the $\overline{HD}$ signal). It is necessary to be read out display line counter several times, because it does not synchronize CPU clock.

Display line counter register (4 bits) ··· DCTR (bit 3 to 0 in ORIRC)

"0000" ··· No display line is completed.
"0001" ··· 1'st display line is completed.
"0010" ··· 2'nd display line is completed.
to
"1111" ··· 15'th display line is completed.



Note : The display line counter also increments when a line with all blank characters or a line with display off is specified.

Figure 2-85. Display Line Counter

2) Interrupt generator circuit

An interrupt request is generated when a falling edge of $\overline{VD}$ signal or when line counter (DCTR) is counted to the certain value specified by ISDC.

Interrupt souce select register (1 bit) ··· SVD (bit4 in ORIRC)

"0" ··· Interrupt request generated when the display line counter (DCTR) is counted to the certain value which is specified by ISDC.

"1" ··· Interrupt request is generated when a falling edge of $\overline{VD}$ signal.

Interrupt generation line specification register (4 bits) ··· ISDC (bit 3 to 0 in ORIRC)

"0000" ··· Interrupt request generated when the display line counter is cleared.

"0001" ··· Interrupt request generated at end points of the last scanning line of the first displey line

"0010" ··· Interrupt request generated at end points of the last scanning line of the 2'nd display line

to

"1111" ··· Interrupt request generated at end points of the last scanning line of the 15'th display line

(16) P6 port output select function

This function is used to select whether the contents of port P67 to P64 will be output or R, G, B, Y/BL signals of the OSD circuit will be output on pins P67 to P64.

P6 port output select registers (4 bits) ··· P67DS to P64DS (bit 7 to 4 in ORP6DS)
"1" ··· R, G, B, Y/BL signal output
"0" ··· Port contents output

(17) OSD pin output polarity control function

This function is used to select the polarity of the OSD outputs for RGB and Y/BL.

Output polarity control register (3 bits) ··· BLIV, YIV, RGBIV (bit 7 to 5 in ORIRC)

Table 2-16. Control of OSD Output Polarity

Symbol	Output port	Data "0"	Data "1"
BLIV	BL	Active High	Active Low
YIV	Y	Active High	Active Low
RGBIV	RGB	Active High	Active Low

(18) Y/BL signal select function

This function is used to select either Y or BL signal output from the Y/BL pin.

Y/BL signal select register (1 bit) ··· YBLCS (bit 7 in ORETC)
"0" ··· Y signal output
"1" ··· BL signal output

Y signal ··· Logical OR for R, G, B Character data, and Fringing data.
BL signal ··· When EXBL is "0":
Output in all display character areas
(except for character code 00_H: blank character)
When EXBL is "1":
Output in the whole page

(19) Writing display data to the display memory

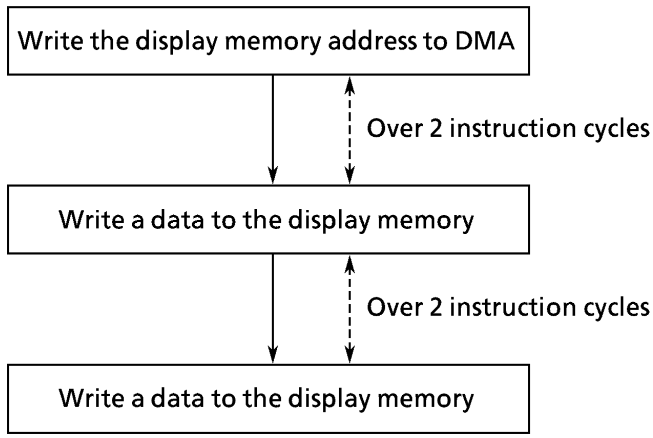
Data are written to the display memory using DMA8 to DMA0, CRA7 to CRA0, RDT, GDT, BDT, BLF, RBF, RVF, and MBK registers.

Display memory address specification register (9 bits) ···
DMA8 to DMA0 (bit0 in ORETC and ORDMA)

Display memory bank switching register (1 bit) ··· MBK (bit1 in ORETC)
"0" ··· For changing either character code or character ornamentation
"1" ··· For changing both character code and character ornamentation

Note 1: Don't use the 2 bytes transfer operation such as [LDW (HL), mn] when accessing to display memory.

Note 2: When writing a data to the display memory immediately after setting the display memory address to DMA, or when continuously writing a data to the display memory, insert over 2 instruction cycles between the instrunction for writing a data.



Example: Setting a character code (20_H) to the display memory (Address : 120_H to 121_H)

```
LD      HL, ORCRA          ; Set ORCRA address to HL reg.
LD      A, 20H             ; Load character code to A reg.

LD      DE, ORDMA          ; Set lower 8-bit addresses to DMA7 to 0
LD      (DE), 20H
LD      DE, ORETC          ; Set the most upper address to DMA8 and set MBK to "0"
LD      (DE), 00000001B

LD      DE, ORDMA          ; Set lower 8-bit addresses to DMA7 to 0
LD      (DE), 20H
LD      DE, ORETC          ; Set the most upper address to DMA8 and set MBK to "0"
LD      (DE), 00000001B

NOP                        ; Insert 2 instruction cycles
NOP

LD      (HL),A             ; Write a character code to display memory (Address : 120H)

NOP                        ; Insert 2 instruction cycles
NOP

LD      (HL),A             ; Write a character code to display memory (Address : 121H)
```

Note 3: Transfer the contents of display memory which affect displaying characters into OSD circuit, before the position of scanning line coincides with their own vertical display start position.

a. Display memory write sequence when writing both character code and character ornamentation.

① Write lower 8-bit addresses of display memory to DMA7 to DMA0 (in ORDMA).

② Write the most upper addresses of display memory to DMA8 (bit 0 in ORETC) and set MBK (bit 1 in ORETC) to “1”.

> *Note: It is necessary to write all bits of display memory address, writting DMA8 after DMA7 to DMA0, when writing display address, and repeat this sequence.*

③ Write character ornamentation data (blinking, reverse, reverse blinking, and color data) to RDT, GDT, BDT, BLF, RVF, and RBF.
At this time, the character ornamentation data is transferred to the display memory.

④ Write character code to CRA7 to CRA0.
At this time, character code is transferred to the display memory together with the character ornamentation data which is written in ③ and DMA8 to DMA0 are automatically incremented.

b. Display memory write sequence when writing either character code or character ornamentation

① Write lower 8-bit addresses of display memory to DMA7 to DMA0 (in ORDMA).

② Write the most upper address of display memory to DMA8 (bit 0 in ORETC) and clear MBK (bit 1 in ORETC) to “0”.

> *Note: It is necessary to write all bits of display memory address, writting DMA8 after DMA7 to DMA0, when writing display address, and repeat this sequence.*

③ Write character ornamentation data (blinking, reverse, reverse blinking, and color data) to RDT, GDT, BDT, BLF, RVF, and RBF or write character code to CRA7 to CRA0.
At this time, written data are transferred to the display memory and DMA8 to DMA0 are automatically incremented.

(20) OSD control register write / read

The address of the OSD control registers are assigned to the DBR area.

To write or to read from the OSD control registers, the method is the same as for accessing ordinary DBR registers.

The written data are transferred to the OSD circuit at the end point of the scanning line without display by setting RGWR register to "1" and become valid. And, be able to write value of OSD control register after RGWR flag is cleared to "0".

> *Note 1 : Do not write the contents of OSD control registers during RGWR flag is "1". If contents of OSD control registers are written during RGWR flag is "1", the written data are broken.*
> *Note 2 : Do not clear RGWR register to "0". If RGWR register is cleared to "0", the contents of OSD control registers may be transferred to OSD circuit at unexpected timing.*
> *Note 3 : Insert over 3 instruction cycles between the instruction which sets RGWR register to "1" and the instruction which checks RGWR flag.*
> *Note 4 : Transfer the contents of all OSD control registers which affect displaying characters into OSD circuit before the position of scanning line coincides with their own vertical display start position.*

Example 1 : In the case of writing the data into OSD registers, setting RGWR register to "1" and checking RGWR flag

```
                [Writing the data into OSD registers]
                                ↓
          LD     A, (TEMP_ORDON)      ; Set bit 2 of work-area to "1"
          SET    A.2

          LD     HL, ORDON            ; Set RGWR register to "1" (Request data transfer)
          LD     (HL), A

          NOP                         ; Insert 3 instruction cycles
          NOP
          NOP
CHECK_RGWR_FLAG:
          TEST   (HL).2               ; Check RGWR flag until RGWR flag is "0"
          JR     F,CHECK_RGWR_FLAG
```

Example 2 : In the case of checking RGWR flag, writing the data into OSD registers, and writing RGWR register to "1"

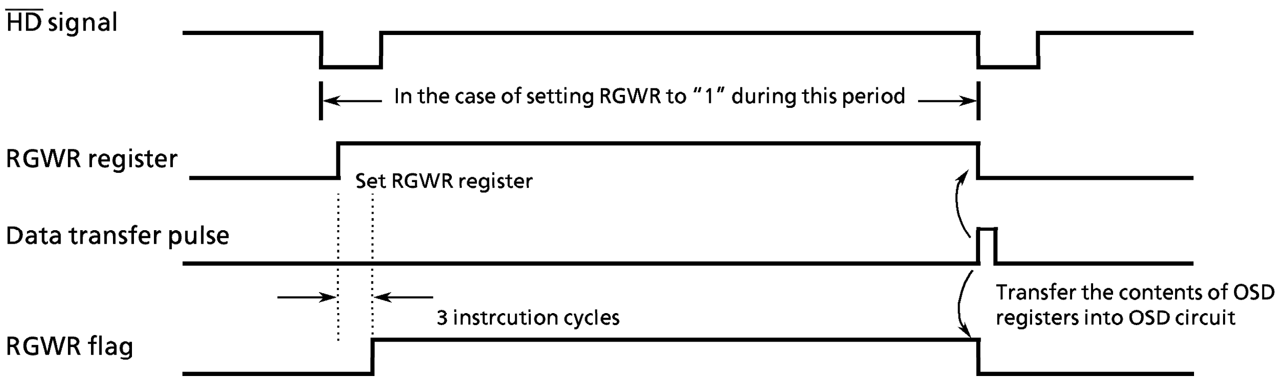
```
          LD     HL, ORDON
CHECK_RGWR_FLAG:
          TEST   (HL).2               ; Checking RGWR flag until RGWR flag is "0"
          JR     F,CHECK_RGWR_FLAG
                                ↓
                [Writing the data into OSD registers]
                                ↓
          LD     A, (TEMP_ORDON)      ; Set bit 2 of work-area to "1"
          SET    A.2

          LD     HL, ORDON            ; Set RGWR register to "1" (Request data transfer)
          LD     (HL), A
```

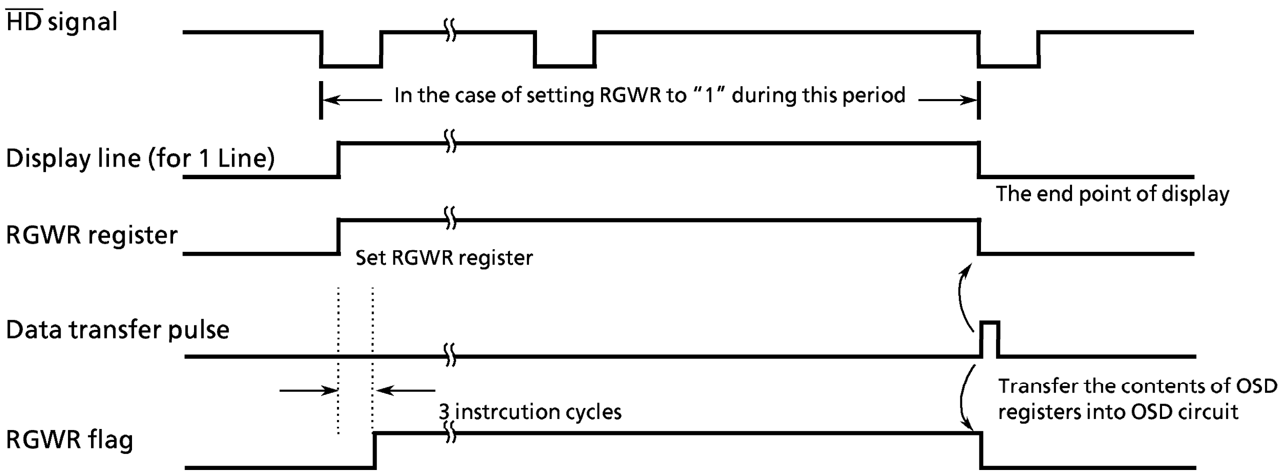
```
NOP                          ; Insert 3 instruction cycles
NOP
NOP
```

The timing chart of transferring the contents of OSD registers into OSD circuit is shown as follows;

1. In the case of setting RGWR register to "1" during the position of the scanning line is in no display area (except any lines specified as display off by CSn).
   The contents of OSD registers are transferred into OSD circuit when the position of the scanning line is at the falling edge of $\overline{HD}$ signal.



2. In the case of setting RGWR register to "1" during the position of the scanning line is in display area (including any lines specified as display off by CSn).
   The contents of OSD registers are transferred into OSD circuit when the position of the scanning line is at the falling edge of $\overline{HD}$ signal of finishing the display line.



For registers (DMA8 to DMA0, CRA7 to CAR0, RDT, GDT, BDT, BLF, RBF, RVF, MBK) used for updating the display memory, P67DS to P64DS, YBLCS, BKMF, RBMF, ESMZ, VDSMD, FORS, RGWR, HDPOL and VDPOL, the data become valid as soon as they are written.

Written data transfer register (1 bit) ··· RGWR (bit 2 in ORDON)
"0" ··· Initial state
"1" ··· Transfer written data to OSD circuit. (After transfer, RGWR register and RGWR flag are automatically cleared to "0".)

Written data transfer monitor flag (1 bit) ··· RGWR (bit 2 in ORDON)
"0" ··· Transfer completed.
"1" ··· During transfer

(21) Display on / off
This function is used to display characters specified for on / off display.

Display on / off specification unit : Display page
Display on / off specification register (1 bit) ··· EDISP (bit 0 in ORDON)
"0" ··· Disable display
"1" ··· Enable display

Note: Do not start STOP mode during display is enabled.

(22) Window function

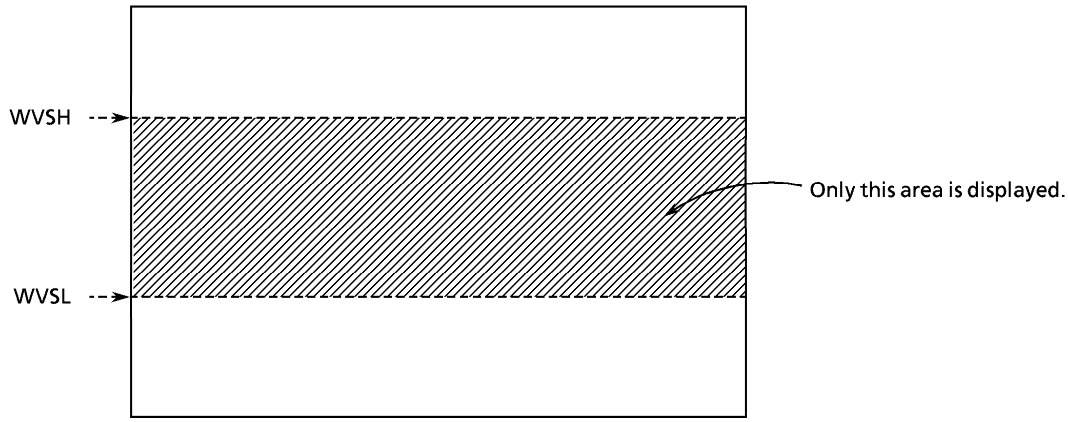
This function is used to set upper and lower limit of display page. Window upper limit is specified by WVSH (ORWVSH). Window lower limit is specified by WVSL (ORWVSL). This function is enabled by setting EWDW (bit 1 in ORDON) in the OSD control register to "1".

Window specification unit: Display page
Window function enable specification register (1 bit) ··· EWDW (bit 1 in ORDON)
"0" ··· Disable window function
"1" ··· Enable window function

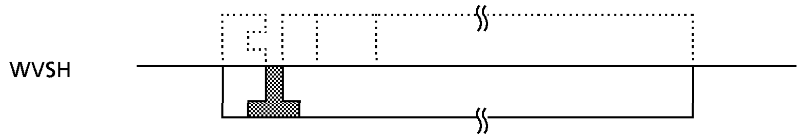
Window upper limit specification register (8 bits) ··· WVSH7 to 0 (ORWVSH)
Window lower limit specification register (8 bits) ··· WVSL7 to 0 (ORWVSL)
Window upper and lower limit position ···
When VDSMD is "0" (Normal mode) :
$WVSH = (WVSH7 \text{ to } WVSH0)_H \times 2T_{HD}$
$WVSL = (WVSL7 \text{ to } WVSL0)_H \times 2T_{HD}$
When VDSMD is "1" (Double scan mode) :
$WVSH = (WVSH7 \text{ to } WVSH0)_H \times 4T_{HD}$
$WVSL = (WVSL7 \text{ to } WVSL0)_H \times 4T_{HD}$

Note 1 : T_{HD} ; One cycle of $\overline{HD}$ signal
Note 2 : $WVSL > WVSH \geqq$ "1"
Note 3 : Modify the value of window upper and lower limit register as follows:
1. *When $WVSH_{NEW} \leqq WVSH_{OLD}$*
   Finish to transfer the new value, during $\overline{VD}$ signal is low or before the position of the scanning line coincides with $WVSH_{NEW}$.
2. *When $WVSL > WVSL_{NEW} > WVSH_{OLD}$*
   Finish to transfer the new value, during $\overline{VD}$ signal is low or before the position of the scanning line coincides with $WVSH_{OLD}$.
3. *When $WVSL_{NEW} \leqq WVSL_{OLD}$*
   Finish to transfer the new value, during $\overline{VD}$ signal is low or before the position of the scanning line coincides with $WVSL_{NEW}$.
4. *When $WVSL_{NEW} > WVSL_{OLD}$*
   Finish to transfer the new value, during $\overline{VD}$ signal is low or before the position of the scanning line coincides with $WVSL_{OLD}$.

Note 4 : It is recommendable that the window function is always enabled (EWDW = "1") and set WVSH to "01_H", WVSL to "FE_H". When the window function should be set to disable, clear EWDW to "0" independent of the value which this register has been set from detecting the rising edge of $\overline{HD}$ signal by software until the falling edge of $\overline{HD}$ signal.



TV screen



(23)OSD Control Registers

Can not access all OSD control registers in any of read-modify-write instructions such as bit operation, etc.

Register	7	6	5	4	3	2	1	0	
ORHS1 (0F80$_H$)	"0"	HS16	HS15	HS14	HS13	HS12	HS11	HS10	(Initial value *000 0000)

HS16 to 10	Horizontal display start position	Write only

Register	7	6	5	4	3	2	1	0	
ORVS1 (0F81$_H$)	VS17	VS16	VS15	VS14	VS13	VS12	VS11	VS10	(Initial value 0000 0000)
⋮		⋮			⋮				⋮
ORVS12 (0F8C$_H$)	VS127	VS126	VS125	VS124	VS123	VS122	VS121	VS120	(Initial value 0000 0000)

VSn7 to 0	Vertical display start position for line n	Write only

(n = 1 to 12)

Note 1 : If display lines are overlapped each other, previous display line is enabled and next line is disabled. Set the vertical display start position not to overlap display lines.

Note 2 : Transfer the contents of vertical display start position registers into OSD circuit before a position of the scanning line coincides with their own vertical display start position.

Register	7 6	5 4	3 2	1 0	
ORCS4 (0F8D$_H$)	CS4	CS3	CS2	CS1	(Initial value 0000 0000)
ORCS8 (0F8E$_H$)	CS8	CS7	CS6	CS5	(Initial value 0000 0000)
ORCS12 (0F8F$_H$)	CS12	CS11	CS10	CS9	(Initial value 0000 0000)

CSn	Character size and display on/off for line n	00: Display off 01 : Large size 10 : Middle size 11 : Small size	Write only

(n = 1 to 12)

OREFR
($0F90_H$)

7	6	5	4	3	2	1	0
EFR8	EFR7	EFR6	EFR5	EFR4	EFR3	EFR2	EFR1

(Initial value 0000 0000)

EFRn	Fringing enable specification register	0 : Disable fringing 1 : Enable fringing	Write only

(n = 1 to 8)

Note : When a display line is enabled fringing function, its vertical size is increased by one dot (by two dots when its character size is small) independent of its character font. Therefore, when a vertical display start position is specified to no space between the lines, the display line which is overlapped with increasing dot(s) is canceled.

ORP6DS
($0F91_H$)

7	6	5	4	3	2	1	0
P67DS	P66DS	P65DS	P64DS	EFR12	EFR11	EFR10	EFR9

(Initial value 0000 0000)

P67DS to P64DS	P6 port output select	0 : Port contents output 1 : R, G, B, Y/BL signal output	Write only
EFRm	Fringing enable specification register	0 : Disable fringing 1 : Enable fringing	

(m = 9 to 12)

Note : When a display line is enabled fringing function, its vertical size is increased by one dot (by two dots when its character size is small) independent of its character font. Therefore, when a vertical display start position is specified to no space between the lines, the display line which is overlapped with increasing dot(s) is canceled.

ORWVSH
($0F92_H$)

7	6	5	4	3	2	1	0
WVSH7	WVSH6	WVSH5	WVSH4	WVSH3	WVSH2	WVSH1	WVSH0

(Initial value 0000 0000)

WVSH7 to 0	Window upper limit position (WVSL > WVSH ≧ 1)	Write only

ORWVSL
($0F93_H$)

7	6	5	4	3	2	1	0
WVSL7	WVSL6	WVSL5	WVSL4	WVSL3	WVSL2	WVSL1	WVSL0

(Initial value 0000 0000)

WVSL7 to 0	Window lower limit position (WVSL > WVSH ≧ 1)	Write only

ORBK
($0F94_H$)

7	6	5	4	3	2	1	0
EBKGD	EXBL	RBDT	GBDT	BBDT	RFDT	GFDT	BFDT

(Initial value 0000 0000)

EBKGD	Background function enable specification register	0 : Disable background 1 : Enable background	Write only
EXBL	Full-raster blanking enable specification register	0 : Disable full-raster blanking 1 : Enable full-raster blanking	
RBDT / GBDT / BBDT	Background color select	000 : Black 001 : Blue 010 : Green 011 : Cyan 100 : Red 101 : Magenta 110 : Yellow 111 : White	
RFDT / GFDT / BFDT	Fringing color select	000 : Black 001 : Blue 010 : Green 011 : Cyan 100 : Red 101 : Magenta 110 : Yellow 111 : White	

Note : When the background color function is used, the blank character (code 00_H) can not be used as the first character on the fringing line.

ORIRC (0F95$_H$)

7	6	5	4	3	2	1	0
BLIV	YIV	RGBIV	SVD	ISDC			

(Initial value 0000 0000)

BLIV	BL output polarity select	0 : Active high 1 : Active low	Write only
YIV	Y output polarity select	0 : Active high 1 : Active low	
RGBIV	R, G, B output polarity select	0 : Active high 1 : Active low	
SVD	Interrupt source select	0: Interrpt request by ISDC value 1 : Interrupt request at falling edge of $\overline{VD}$ signal	
ISDC	Interrupt generation line select		

ORIRC (0F95$_H$)

7	6	5	4	3	2	1	0
				DCTR			

(Initial value **** 0000)

DCTR	Display line counter	Read only

Note : The display line counter also increments when a line with all blank data or a line with display off is specified.

ORETC (0F96$_H$)

7	6	5	4	3	2	1	0
YBLCS	BKMF	RBMF	ESMZ	VDSMD	"0"	MBK	DMA8

(Initial value 0000 0000)

YBLCS	Y/BL signal select	0 : Y signal output 1 : BL signal output	Write only
BKMF	Blinking master enable specification register	0 : Disable blinking 1 : Enable blinking	
RBMF	Reverse blinking master enable specification register	0 : Disable reverse blinking 1 : Enable reverse blinking	
ESMZ	Smoothing enable specification register	0 : Disable smoothing 1 : Enable Smoothing	
VDSMD	Double scan mode select	0 : Normal mode 1 : Double scan mode	
MBK	Display memory bank switching	0 : Access to either character code or character display options 1 : Access to Both character code and character display options	
DMA8	Display memory address (bit 8)		

Note 1 : Clear "0" to bit 2 in ORETC.
Note 2 : It is necessary to write all bits of display memory address, writing DMA8 after DMA7 to DMA0, when writing display address, and repeat this sequence.

ORDMA (0F97$_H$)

7	6	5	4	3	2	1	0
DMA7	DMA6	DMA5	DMA4	DMA3	DMA2	DMA1	DMA0

(Initial value 0000 0000)

DMA7 to 0	Display memory address	Write only

Note : It is necessary to write all bits of display memory address, writing DMA8 after DMA7 to DMA0, when writing display address, and repeat this sequence.

ORDSN ($0F98_H$)

7	6	5	4	3	2	1	0	
		RBF	RVF	BLF	RDT	GDT	BDT	(Initial value **** ****)

RBF	Reverse blinking enable specification register	0 : Disable reverse blinking 1 : Enable reverse blinking	Write only
RVF	Reverse enable specification register	0 : Disable reverse 1 : Enable reverse	
BLF	Blinking enable specification register	0 : Disable blinking 1 : Enable blinking	
RDT/ GDT/ BDT	Character color select	000 : Black 001 : Blue 010 : Green 011 : Cyan 100 : Red 101 : Magenta 110 : Yellow 111 : White	

ORCRA ($0F99_H$)

7	6	5	4	3	2	1	0	
CRA7	CRA6	CRA5	CRA4	CRA3	CRA2	CRA1	CRA0	(Initial value **** ****)

CRA7 to 0	Character code	Write only

ORDON ($0F9A_H$)

7	6	5	4	3	2	1	0	
"0"	"0"	"0"	FORS	"1"	RGWR	EWDW	EDISP	(Initial value ***0 0000)

FORS	fosc frequency select	0 : Normal frequency mode 1 : Double frequency mode	Write only
RGWR	Written data transfer request register	0 : (Initial state) 1 : Transfer written data to OSD circuit. (After transfer, RGWR is cleared to "0".)	
EWDW	Window function enable specification register	0 : Disable window function 1 : Enable window function	
EDISP	Display on/off specification register	0 : Disable Display 1 : Enable Display	

*Note1 : * ; don't care*
Note2 : The data written to OSD control registers except the register followed table is transmitted to OSD circuit by setting RGWR (bit2 in ORDON) to "1". RGWR is cleared to "0" automatically after the transfer is completed.

P67DS, P66DS, P65DS, P64DS	bits 7 to 4 in ORP6DS
YBLCS, BKMF, RBMF, ESMZ, VDSMD, MBK, DMA8	bits 7 to 3, 1 to 0 in ORETC
DMA7, DMA6, DMA5, DMA4, DMA3, DMA2, DMA1, DMA0	bits 7 to 0 in ORDMA
RBF, RVF, BLF, RDT, GDT, BDT	bits 5 to 0 in ORDSN
CRA7, CRA6, CRA5, CRA4, CRA3, CRA2, CRA1, CRA0	bits 7 to 0 in ORCRA
FORS, RGWR	bits 4 and 2 in ORDON
HDPOL, VDPOL	bits 1 and 0 in ORPOL

Note3 : When EWDW is cleared to "0", clear EWDW to "0" independent of the value which this register has been set from detecting the rising edge of $\overline{HD}$ signal by software until the falling edge of $\overline{HD}$ signal.
Note4 : Write "1" to bit 3 of ORDON when writing to ORDON.
Note5 : Do not clear RGWR register to "0". If RGWR register is cleared to "0", the contents of OSD control registers may be transferred to OSD circuit at unexpected timing.

ORDON ($0F9A_H$)

7	6	5	4	3	2	1	0
			FORS		RGWR	EWDW	EDISP

(Initial value ***0 *000)

FORS	fosc frequency select status	0 : Normal frequency mode 1 : Double frequency mode	Read only
RGWR	Written data transfer monitor flag	0 : Transfer completed 1 : During transfer	
EWDW	Window function enable specification status	0 : Disable window function 1 : Enable window function	
EDISP	Display on/off status	0 : Disable display 1 : Enable display	

ORPOL ($0F9B_H$)

7	6	5	4	3	2	1	0
						VDPOL	HDPOL

(Initial value **** **00)

HDPOL	HD polality select	0 : Non-invert input signal 1 : Invert input signal	Read/ Write
VDPOL	VD polality select	0 : Non-invert input signal 1 : Invert input signal	

2.15 Jitter Elimination Circuit

In an interlace TV, the 87CM39/P39/S39 has a jitter elimination circuit which can display stably without moving up and down TV even if a vertical synchronous signal input at the on-screen display is disarranged.

2.15.1 Configuration

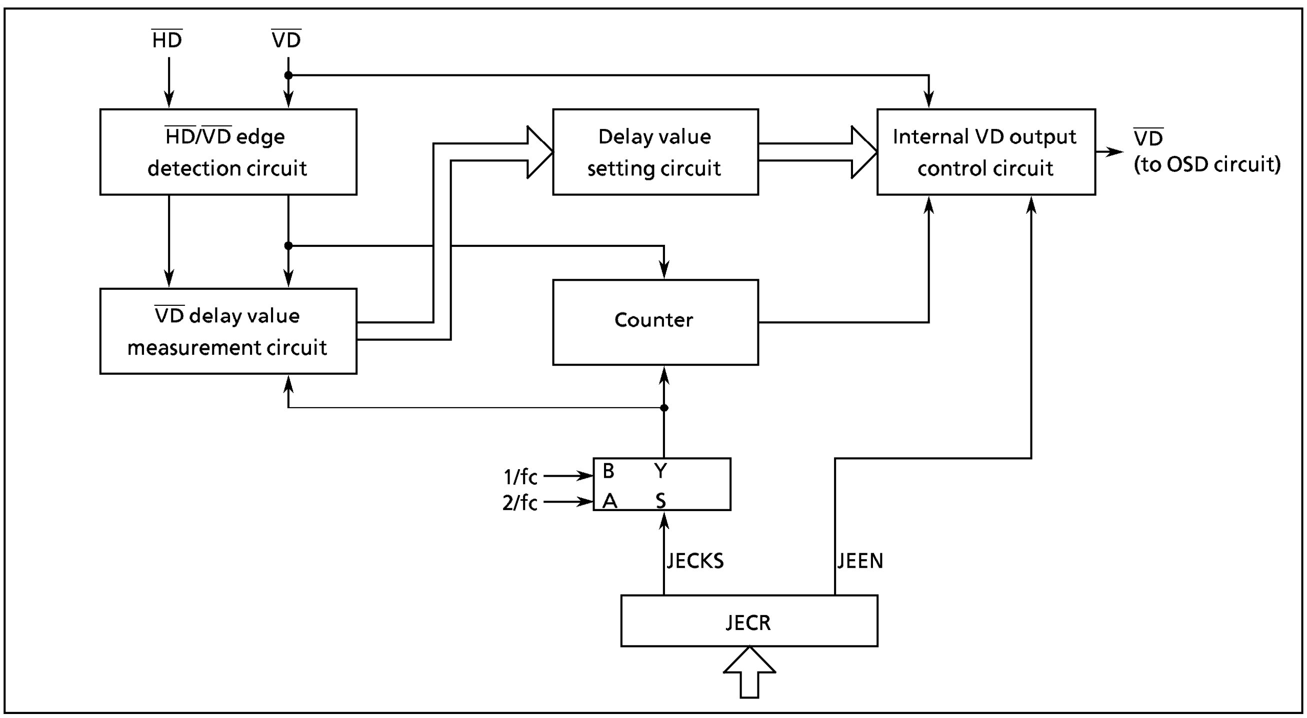


Figure 2-86. Jitter Elimination Circuit

2.15.2 Control

Jitter elimination circuit is controlled by the jitter elimination control register (JECR).

Jitter elimination control register

JECR (0FAB$_H$)	7	6	5	4	3	2	1	0	
					"0"	JEEN	"0"	JECKS	(initial value: **** *0*0)

JEEN	Jitter elimination enable specification	0 : disable 1 : enable	write only
JECKS	Source clock select	0 : 2/fc (Select this, when fc = 8 MHz) 1 : 1/fc (Select this, when fc = 4 MHz)	

Note 1: When using jitter elimination function, set the JECKS using the CPU clock of 4 MHz or 8 MHz. Using different frequency clock may cause malfunctions.

Note 2: The bit 1 and bit 3 of JECR are always cleared to "0".

Figure 2-87. Jitter Elimination Control Register

2.15.3 Functions

Jitter elimination circuit detects the different phase between the falling edge of the external $\overline{VD}$ signal and $\overline{HD}$ signal. When $\overline{VD}$ signal is falling within $\overline{HD}$ signal falling +/− 1/4 $\overline{HD}$, the jitter is automatically eliminated and internal $\overline{VD}$ signal is set to the stable location.

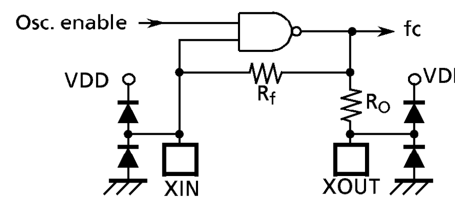
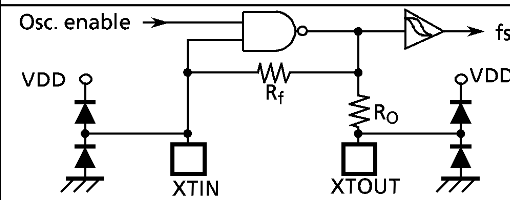
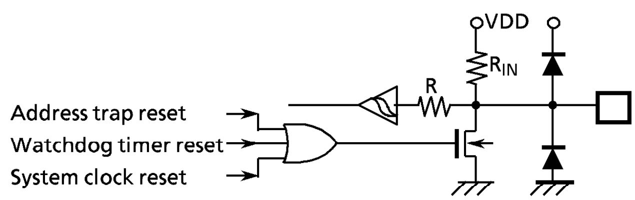
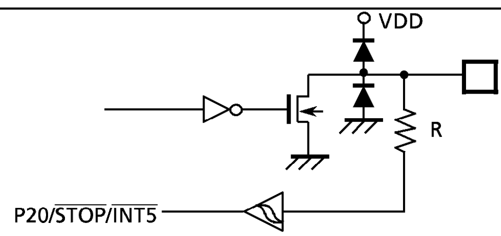
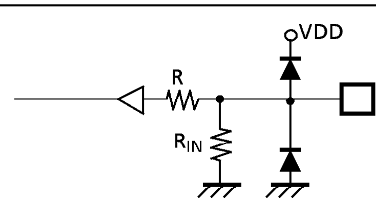
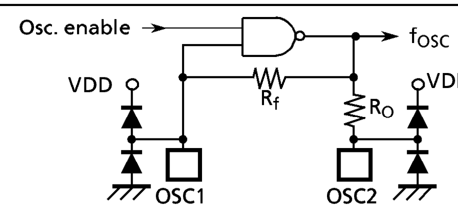
This function is enabled by setting JEEN (bit2 in JECR) in the jitter elimination control register to "1". CPU clock must be used at 8 MHz or 4 MHz at using the jitter elimination. JECKS (bit0 in JECR) must be set to "0" at 8 MHz, and "1" at 4 MHz.

INPUT / OUTPUT CIRCUITRY

(1) Control pins

The input / output circuitries of the 87CM39/P39/S39 control pins are shown below.

After reset, set the operation mode either to the single clock mode (only XIN/XOUT) or to the dual clock mode (XIN/XOUT and XTIN/XTOUT) by mask option (code NM1, NM2).

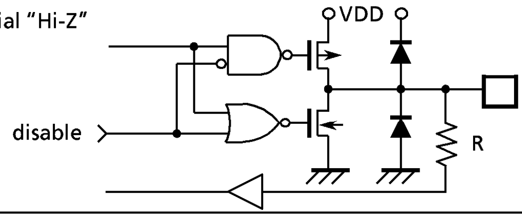
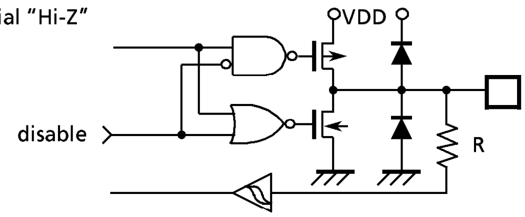
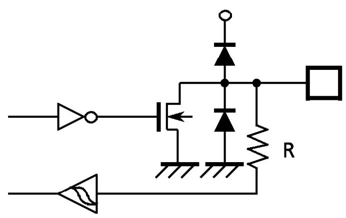
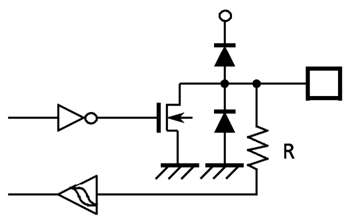
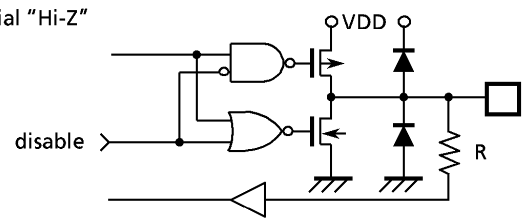
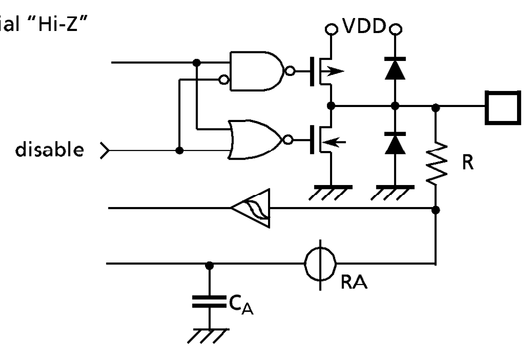
CONTROL PIN	I/O	INPUT / OUTPUT CIRCUITRY	REMARKS
XIN XOUT	Input Output		Resonator connecting pins (high-frequency) R_f = 1.2 MΩ (typ.) R_O = 1.5 kΩ (typ.)
XTIN XTOUT	Input Output	NM1: Refer to P2 port NM2: 	Resonator connecting pins (low-frequency) R_f = 6 MΩ (typ.) R_O = 220 kΩ (typ.)
$\overline{RESET}$	I/O		Sink open drain output Hysteresis input Pull-up resistor R_{IN} = 220 kΩ (typ.) R = 1 kΩ (typ.)
$\overline{STOP}/\overline{INT5}$	Input		Hysteresis input R = 1 kΩ (typ.)
TEST	Input		Pull-down resistor R_{IN} = 70 kΩ (typ.) R = 1 kΩ (typ.)
OSC1 OSC2	Input Output		Osc. connecting pin for on-screen display R_f = 1.2 MΩ (typ.) R_O = 1.5 kΩ (typ.)

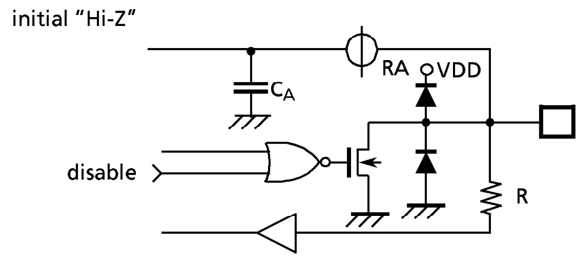
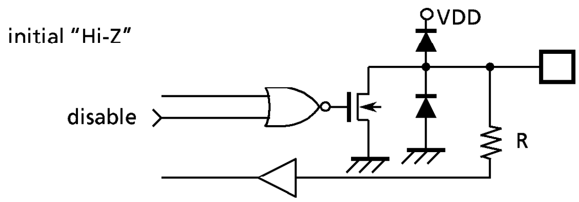
Note 1) The TMP87PS39 test pin dose not have the pull-down resistor. It must be fixed to Low-level.

Note 2) After reset, the TMP87PS39 is set to the single clock mode (NM1).

(2) Input / Output Ports

The input / output circuitries of the 87CM39/P39/S39 I/O ports are shown below.

PORT	I/O	INPUT / OUTPUT CIRCUITRY	REMARKS
P0	I/O	initial "Hi-Z" 	Tri-state I/O R = 1 kΩ (typ.)
P1 P50 to P51	I/O	initial "Hi-Z" 	Tri-state I/O Hysteresis input R = 1 kΩ (typ.)
P2	I/O	initial "Hi-Z" 	Sink open drain output Hysteresis input R = 1 kΩ (typ.)
P3 P7	I/O	initial "Hi-Z" 	Sink open drain output Hysteresis input R = 1 kΩ (typ.)
P4 P64 to P67	I/O	initial "Hi-Z" 	Tri-state I/O R = 1 kΩ (typ.)
P52 to P57	I/O	initial "Hi-Z" 	Tri-state I/O Hysteresis input R = 1 kΩ (typ.) RA = 5 kΩ (typ.) C_A = 12 pF (typ.)

PORT	I/O	INPUT/OUTPUT CIRCUITRY	REMARKS
P60 to P61	I/O		Sink open drain output High current output I_{OL} = 20 mA(typ.) R = 1 kΩ (typ.) RA = 5 kΩ (typ.) C_A = 12 pF (typ.)
P62 to P63	I/O		Sink open drain output High current output I_{OL} = 20 mA (typ.) R = 1 kΩ (typ.)

ELECTRICAL CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS
(V_{SS} = 0 V)

PARAMETER	SYMBOL	PINS	RATINGS	UNIT
Supply Voltage	V_{DD}		– 0.3 to 6.5	V
Input Voltage	V_{IN}		– 0.3 to V_{DD} + 0.3	V
Output Voltage	V_{OUT1}		– 0.3 to V_{DD} + 0.3	V
Output Current (Per 1 pin)	I_{OUT1}	Ports P0, P1, P2, P3, P4, P5, P64 to P67, P7	3.2	mA
	I_{OUT2}	Ports P60 to P63	30	
Output Current (Total)	ΣI_{OUT1}	Ports P0, P1, P2, P3, P4, P5, P64 to P67, P7	120	mA
	ΣI_{OUT2}	Ports P60 to P63	120	
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} = 0 V, Topr = – 30 to 70 °C)

PARAMETER	SYMBOL	PINS	CONDITIONS		Min.	Max.	UNIT
Supply Voltage	V_{DD}		fc = 8 MHz	NORMAL1, 2 mode	4.5	5.5	V
				IDLE1, 2 mode			
			fs = 32.768 kHz	SLOW mode	2.7		
				SLEEP mode			
				STOP mode	2.0		
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			$V_{DD} \times 0.75$		
	V_{IH3}		$V_{DD} < 4.5V$		$V_{DD} \times 0.90$		
Input Low Voltage	V_{IL1}	Except hysteresis input	$V_{DD} \geqq 4.5V$		0	$V_{DD} \times 0.30$	V
	V_{IL2}	Hysteresis input				$V_{DD} \times 0.25$	
	V_{IH3}		$V_{DD} < 4.5V$			$V_{DD} \times 0.01$	
Clock Frequency	fc	XIN, XOUT	V_{DD} = 4.5 to 5.5V		4.0	8.0	MHz
	f_{OSC}	OSC1, OSC2	Normal frequency mode (FORS = 0, V_{DD} = 4.5 to 5.5 V)		4.0	fosc $\leqq$ fc × 1.2 $\leqq$ 8.0	
			Double frequency mode (FORS = 1, V_{DD} = 4.5 to 5.5 V)		2.0	fosc $\leqq$ fc × 0.6 $\leqq$ 4.0	
	fs	XTIN, XTOUT			30.0	34.0	kHz

Note 1 : Clock frequency fc ; The condition of supply voltage range is the value in NORMAL 1/2 mode and IDLE 1/2 mode.
Note 2 : When using test video signal circuit, high frequency must be 8 MHz.
Note 3 : When the OSD circuit is used, the supply voltage must be from 4.5 V to 5.5 V.

D.C. CHARACTERISTICS

(V_{SS} = 0 V, Topr = – 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} = 5.5 V, V_{IN} = 5.5 V / 0 V	–	–	± 2	μA
	I_{IN2}	Open drain ports	V_{DD} = 5.5 V, V_{IN} = 5.5 V	–	–	2	
	I_{IN3}	Tri-state ports	V_{DD} = 5.5 V, V_{IN} = 5.5 V / 0 V	–	–	± 2	
	I_{IN4}	$\overline{RESET}$, $\overline{STOP}$					
Input Resistance	R_{IN2}	$\overline{RESET}$		100	220	450	kΩ
Output Leakage Current	I_{LO1}	Sink open drain ports	V_{DD} = 5.5 V, V_{OUT} = 5.5 V	–	–	2	μA
	I_{LO2}	Tri-state ports	V_{DD} = 5.5 V, V_{OUT} = 5.5 V / 0 V	–	–	± 2	
Output High Voltage	V_{OH2}	Tri-state ports	V_{DD} = 4.5 V, I_{OH} = – 0.7 mA	4.1	–	–	V
Output Low Voltage	V_{OL}	Except XOUT and ports P63 to P60	V_{DD} = 4.5 V, I_{OL} = 1.6 mA	–	–	0.4	V
Output Low current	I_{OL3}	Ports P63 to P60	V_{DD} = 4.5 V, V_{OL} = 1.0 V	–	20	–	mA
Supply Current in NORMAL 1, 2 modes	I_{DD}		V_{DD} = 5.5 V, V_{IN} = 5.3 V / 0.2 V fc = 8 MHz fs = 32.768 kHz	–	13	20	mA
Supply Current in IDLE 1, 2 modes				–	6.5	10	
Supply Current in SLOW mode			V_{DD} = 3.0 V fs = 32.768 kHz V_{IN} = 2.8 V / 0.2 V	–	30	70	μA
Supply Current in SLEEP mode				–	15	35	
Supply Current in STOP mode			V_{DD} = 5.5 V V_{IN} = 5.3 V / 0.2 V	–	0.5	10	μA

Note 1 : Typical values show those at Topr = 25 °C , V_{DD} = 5 V.
Note 2 : Input Current I_{IN1}, I_{IN4} ; The current through pull-up or pull-down resistor is not included.
Note 3 : Supply Current I_{DD} ; The current (Typ. 0.5 mA) through ladder resistors of ADC is included in NORMAL mode and IDEL mode.

A / D CONVERSION CHARACTERISTICS

(V_{SS} = 0 V, V_{DD} = 4.5 to 5.5 V, Topr = – 30 to 70 °C)

PARAMETER	SYMBOL	CONDITIONS	Min.	Typ.	Max.	UNIT
Analog Reference Voltage	V_{DD}	supplied from V_{DD} pin	–	V_{DD}	–	V
	V_{SS}	supplied from V_{SS} pin	–	0	0	
Analog Reference Voltage Range	ΔV_{AREF}	= V_{DD} – V_{SS}	–	V_{DD}	–	
Analog Input Voltage	V_{AIN}		V_{SS}	–	V_{DD}	
Nonlinearity Error		V_{DD} = 4.5V to 5.5V	–	–	± 1	LSB
Zero Point Error			–	–	± 2	
Full Scale Error			–	–	± 2	
Total Error			–	–	± 3	

A.C. CHARACTERISTICS

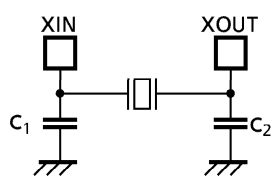
(V_{SS} = 0 V, V_{DD} = 4.5 to 5.5 V, Topr = – 30 to 70 °C)

PARAMETER	SYMBOL	CONDITIONS	Min.	Typ.	Max.	UNIT
Machine Cycle Time	t_{cy}	In NORMAL1, 2 modes In IDLE1, 2 modes	0.5	–	1.0	μs
		In SLOW mode In SLEEP mode	117.6	–	133.3	
High-Level Clock Pulse Width	t_{WCH}	For external clock operation (XIN input), fc = 8 MHz	62.5	–	–	ns
Low-Level Clock Pulse Width	t_{WCL}					
High-Level Clock Pulse Width	t_{WSH}	For external clock operation (XTIN input), fs = 32.768 kHz	14.7	–	–	μs
Low-Level Clock Pulse Width	t_{WSL}					

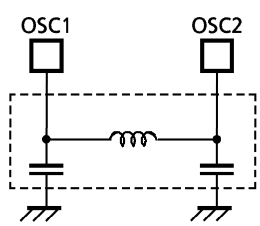
RECOMMENDED OSCILLATING CONDITIONS

(V_{SS} = 0 V, V_{DD} = 4.5 to 5.5 V, Topr = – 30 to 70 °C)

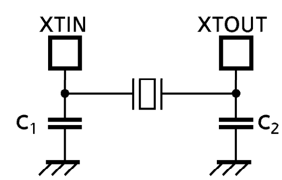
PARAMETER	Oscillator	Oscillation Frequency	Recommended Oscillator	Recommended Constant	
				C_1	C_2
High-frequency Oscillation	Ceramic Resonator	8 MHz	KYOCERA KBR8.0M	30 pF	30 pF
		4 MHz	KYOCERA KBR4.0MS		
			MURATA CSA4.00MG		
	Crystal Oscillator	8 MHz	TOYOCOM 210B 8.0000	20 pF	20 pF
		4 MHz	TOYOCOM 204B 4.0000		
OSD	LC Resonator	8 MHz	TOKO A285TNIS-11695	–	–
		7 MHz	TOKO TBEKSES-30375FBY		
Low-frequency Oscillation	Crystal Oscillator	32.768 kHz	NDK MX-38T	15 pF	15 pF



(1) High-frequency Oscillation



(2) LC Resonator for OSD



(3) Low-frequency Oscillation

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, be CRT (Cathode Ray Tube).