CMOS 8-BIT MICROCONTROLLER

TMP87C833N, TMP87C33N, TMP87CH33N,

The 87C833/C33/H33 is the high speed and high performance 8-bit single chip microcomputer. This MCU contains CPU core, ROM, RAM, input/output ports, six multi-function timer/counter, serial interface, on-screen display, PWM, 6-bit A/D conversion inputs and remote control signal processor on a chip.

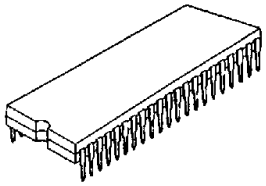
The functions of the OSD circuit conform to the on-screen display functions of closed caption decoders based on FCC standards.

PART No	ROM	RAM	PACKAGE	OTP MCU
TMP87C833N	8K bytes	1K bytes	SDIP42-P-600	TMP87PH33N
TMP87CC33N	12K bytes			
TMP87CH33N	16K bytes			

FEATURES

- ◆8-bit single chip microcomputer TLCS-870 Series
- ◆Instruction execution time : 0.5μs (at 8MHz)
- ◆412 basic instructions
  - Multiplication and Division (8bits x 8bits , 16bits ÷ 8bits)
  - Bit manipulations(Set/Clear/Complement/Move/Test/Exclusive Or)
  - 16-bit data operations
  - 1-byte jump/subroutine-call (Short relative jump / Vector call)
- ◆14 interrupt sources (External : 5, 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
- ◆6 Input/Output ports (34 pins)
  - High current output : 4pins (typ. 20mA)
- ◆Two 16-bit Timer
- ◆Two 8-bit Timer/Counters
  - Timer, Event counter, Capture (Pulse width/duty measurement) modes
- ◆Time Base Timer (Interrupt frequency : 1Hz to 16kHz)
- ◆Watchdog Timer
  - Interrupt source/reset output (programmable)
- ◆Serial Interface
  - I2C-bus/8-bits SIO modes
- ◆On-screen display circuit
  - Character patterns : 256 characters
  - Character displayed : 32 column 8 lines
  - Composition : 8 x 9 dots
  - Size of character : 3kinds (line by line)
  - Color of character : 7 kinds (character by character)
  - Variable display position : Horizontal/Vertical 128/256 steps
  - Fringing, Smoothing function
  - Conform to US CLOSED CAPTION DECODER REGULATION
- ◆PWM outputs
  - 14-bit PWM output (1 channel)
  - 7-bit PWM outputs (9 channels)
- ◆6-bit A/D conversion input (4 channels)
- ◆Pulse output (Clock for PLL IC)
- ◆Remote control signal processor
- ◆Two Power saving operating modes
  - STOP mode : Oscillation stops. Battery/Capacitor back-up. Port output hold/high-impedance.
  - IDLE mode : CPU stops, and Peripherals operate using high-frequency clock. Release by interrupts.
- ◆Wide operating voltage : 2.7~5.5V at 4.19MHz, 4.5~5.5V at 8MHz
- ◆Emulation Pod : BM87CH33N0A
- ◆Data Slicer : TA8862P

SDIP42-P-600

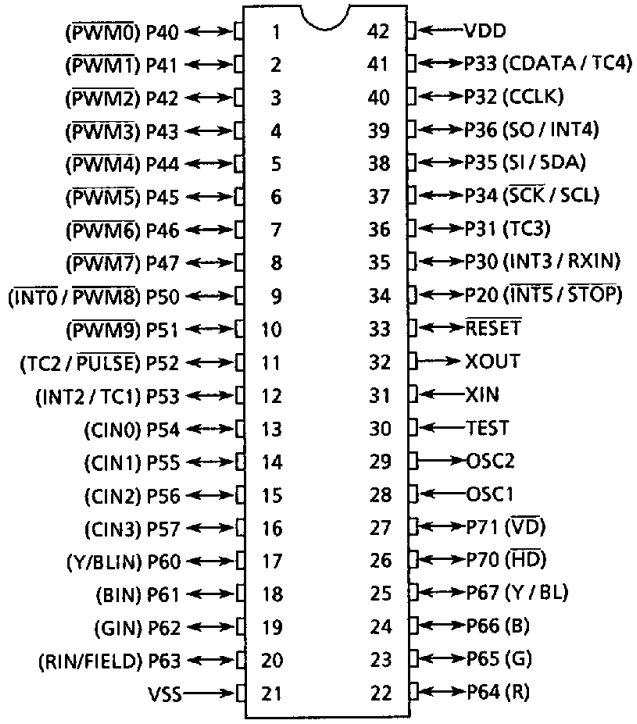


TMP87C833N
TMP87CC33N
TMP87CH33N

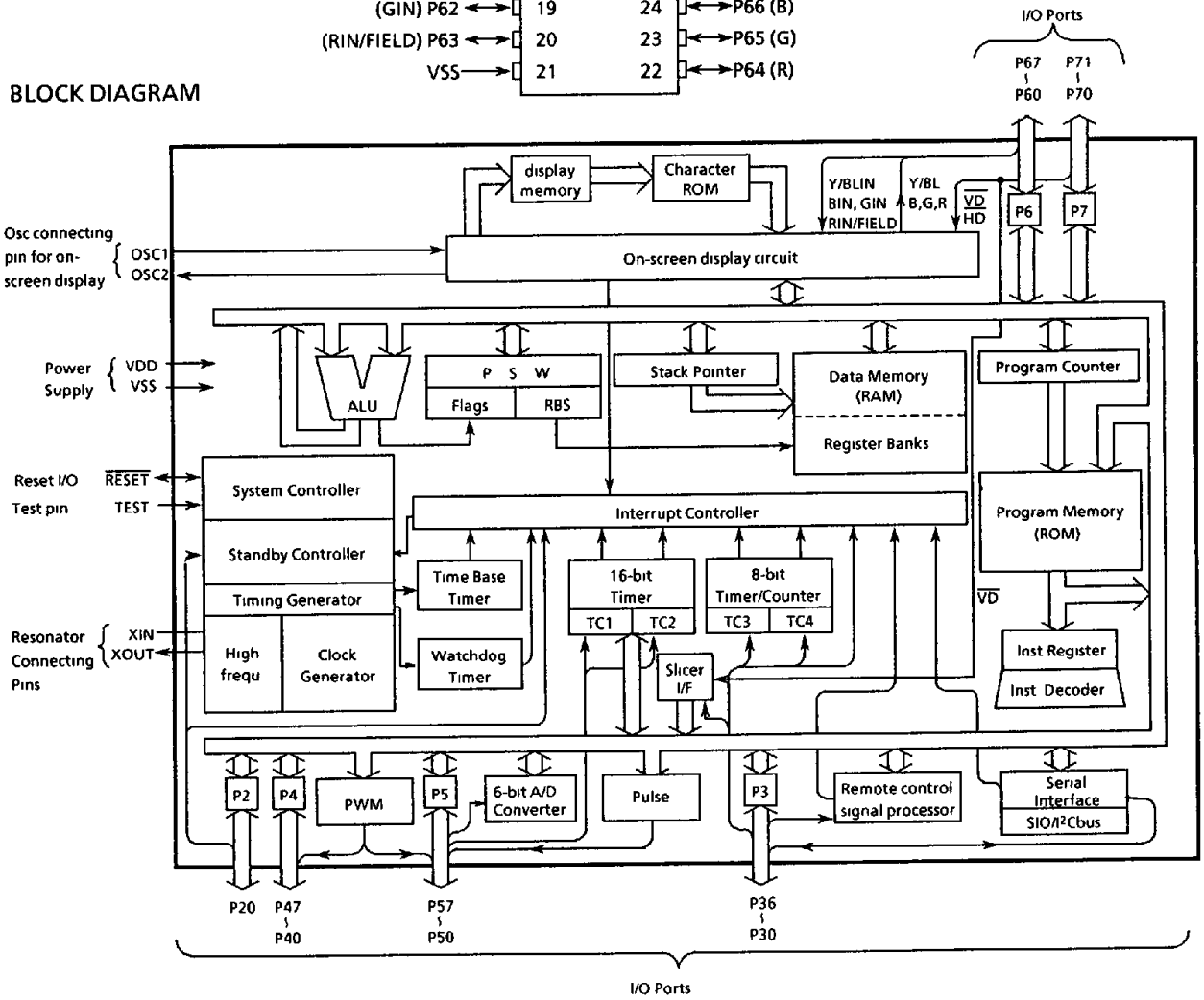
Purchase of TOSHIBA I² C components conveys a license under the Philips I² C Patent Rights to use these components in an I² C system, provided that the system conforms to the I² C Standard Specification as defined by Philips.

PIN ASSIGNMENTS (TOP VIEW)

SDIP42-P-600



BLOCK DIAGRAM



PIN FUNCTION

PIN NAME	Input/Output	Function	
P20 ($\overline{INT5}$/$\overline{STOP}$)	I/O (Input)	1-bit input / output port with latch When used as an input port, or an interrupt input/STOP mode release signal input, the latch must be set to "1"	External intrrupt input 5 / STOP mode release signal input
P36 (SO/INT4)	I/O (O / I)	7-bit input/output port with latch. When used as an input port, a serial interface input/output, a timer/counter input, a remote control signal processor input, slicer interface, or an intrrupt input, the latch must be set to "1".	SIO serial clock output/External intrrupt input4
P35 (SDA/SI)	I/O (I/O/Input)		I²Cbus serial data input/output/SIO serial data input
P34 (SCL/$\overline{SCK}$)	I/O (I/O/Input)		I²Cbus serial clock input/output or SIO serial clock input
P33 (CDATA)	I/O (Input)		Caption data input
P32 (CCLK)			Caption clock input
P31 (TC3)	I/O (Input)		Timer/Counter 3 input
P30 (INT3/RXIN)	I/O (Input/Input)		External intrrupt input 3 / remote control signal processor input
P47 ($\overline{PWM7}$) ~P41 ($\overline{PWM1}$)	I/O (Output)	8-bit programable input/output port (tri-state). Each bit of this port can be individually 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 PWM outputs
P40 ($\overline{PWM0}$)			14-bit PWM output
P57 (CIN3) ~P54 (CIN0)	I/O (Input)	8-bit input/output port with latch. When used as an input port, a PWM output, or a pluse output, the latch must be set to "1"	Comparator inputs
P53 (INT2 / TC1)	I/O		External interrupt input 2/Timer/Counter 1 input
P52 ($\overline{PULSE}$ / TC2)			Pulse output (Clock for PLL IC) /Timer/Counter 2 input
P51 ($\overline{PWM9}$) P50 ($\overline{PWM8}$ / $\overline{INT0}$)	I/O (Output)		7-bit PWM outputs/External intrrupt input 0
P71 ($\overline{VD}$)	I/O (Input)	2-bit input/output port with latch When used as an input port, a vertical synchronous signal input, or a horizontal sychronous signal input, the latch must be set to "1"	Vertical synchronous signal input
P70 ($\overline{HD}$)			Horizontal synchronous signal input
P67 (Y/BL) P66 (B) P65 (G) P64 (R)	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 as an input or an output under software control. During reset, all bits are configured as inputs. When P67 to P64 port used as port output, bits 7 to 4 of address $0F91_H$ must be set to "1". When P63 to P60 port used as RIN/FIELD, GIN, BIN, Y/BLIN input, these ports must be as inputs.	R, G, B, Y/BL output
P63 (RIN/FIELD)	I/O		R input/Field status input
P62 (GIN) P61 (BIN) P60 (Y/BLIN)			G, B, Y/BL input
OSC1, OSC2 XIN, XOUT	Input, Output	Resonator connecting pin of on-screen display circuit.	Resonator connecting pin (High frequency) For input 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	+ 5V, 0V (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 87C833/C33/H33. 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.

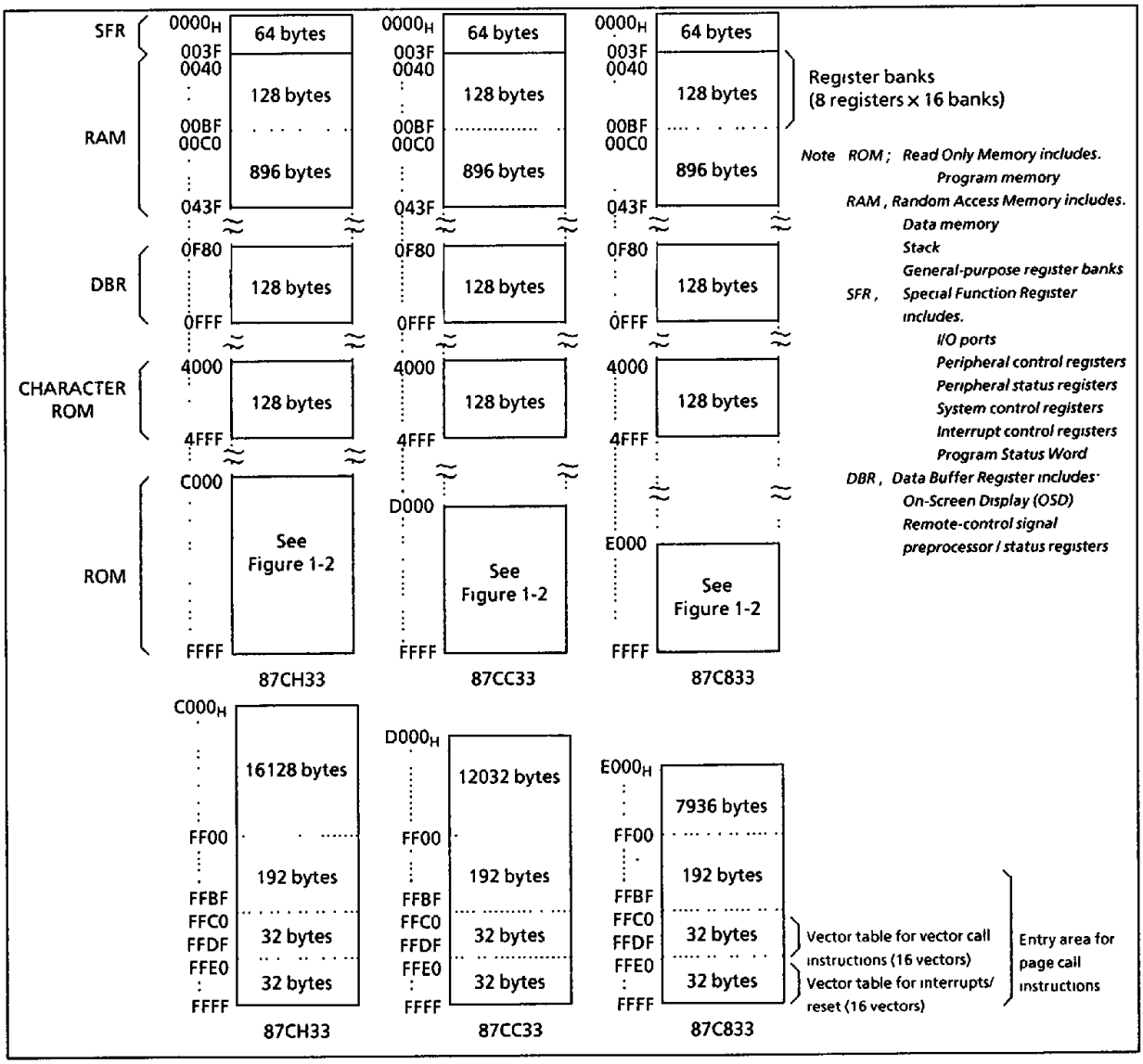


Figure 1-1. Memory Address Map

1.2 Program Memory (ROM)

The 87C833/C33/H33 has an 8K/12K/16Kbytes (addresses $E000_H$/$D000_H$/$C000_H$-$FFFF_H$) of program memory (mask programmed ROM).
Addresses $FF00_H$-$FFFF_H$ in the program memory can also be used for special purposes.

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

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

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

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

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

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

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

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

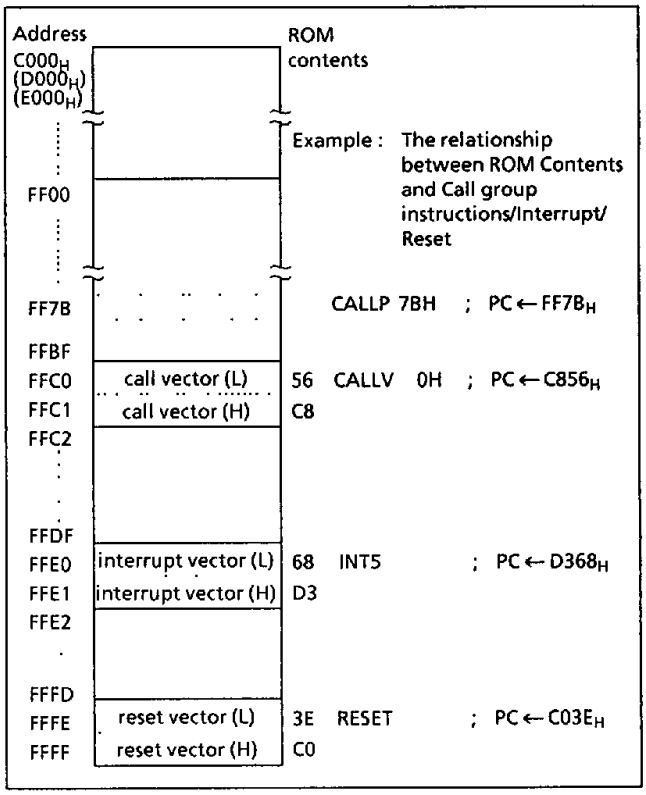


Figure 1-2. Program Memory Map

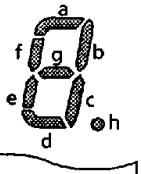
In the TLCS-870 Series, the same instruction used to access the data memory (e.g. [LD A, (HL)]) is also used to read out fixed data (ROM data) stored in the program memory. The register-offset PC-relative addressing (PC + A) instructions can also be used, and the code conversion, table look-up and n-way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

Example 1 : Loads the ROM contents at the address specified by the HL register pair contents into the accumulator (HL≧$C000_H$ for 87CH33):

```
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 :
```

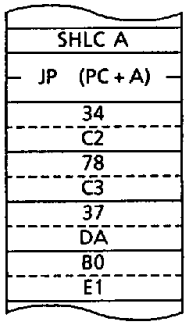


Notes : "$" is a header address of ADD instruction.
DB is a byte data difinition instruction.

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 $C125_H$.

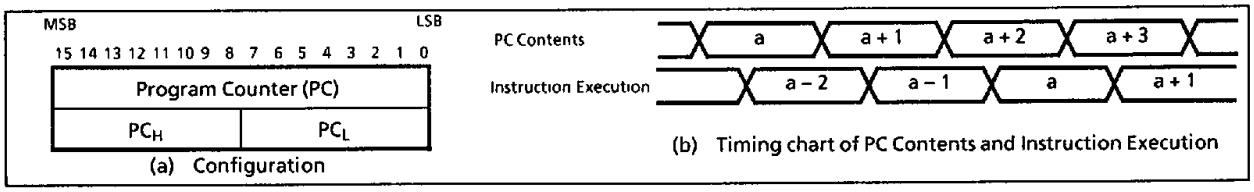


Figure 1-3. Program Counter

1.4 Data Memory (RAM)

The 87C833/C33/H33 has a 1K bytes (addresses 0040_H-$043F_H$) of data memory (static RAM). Figure 1-4 shows the data memory map.

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

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

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

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

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

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

The data memory contents become unstable when the power supply is turned on; therefore, the data memory should be initialized by an initialization routine.

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

Example : Clears RAM to "00_H" except the bank 0:

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

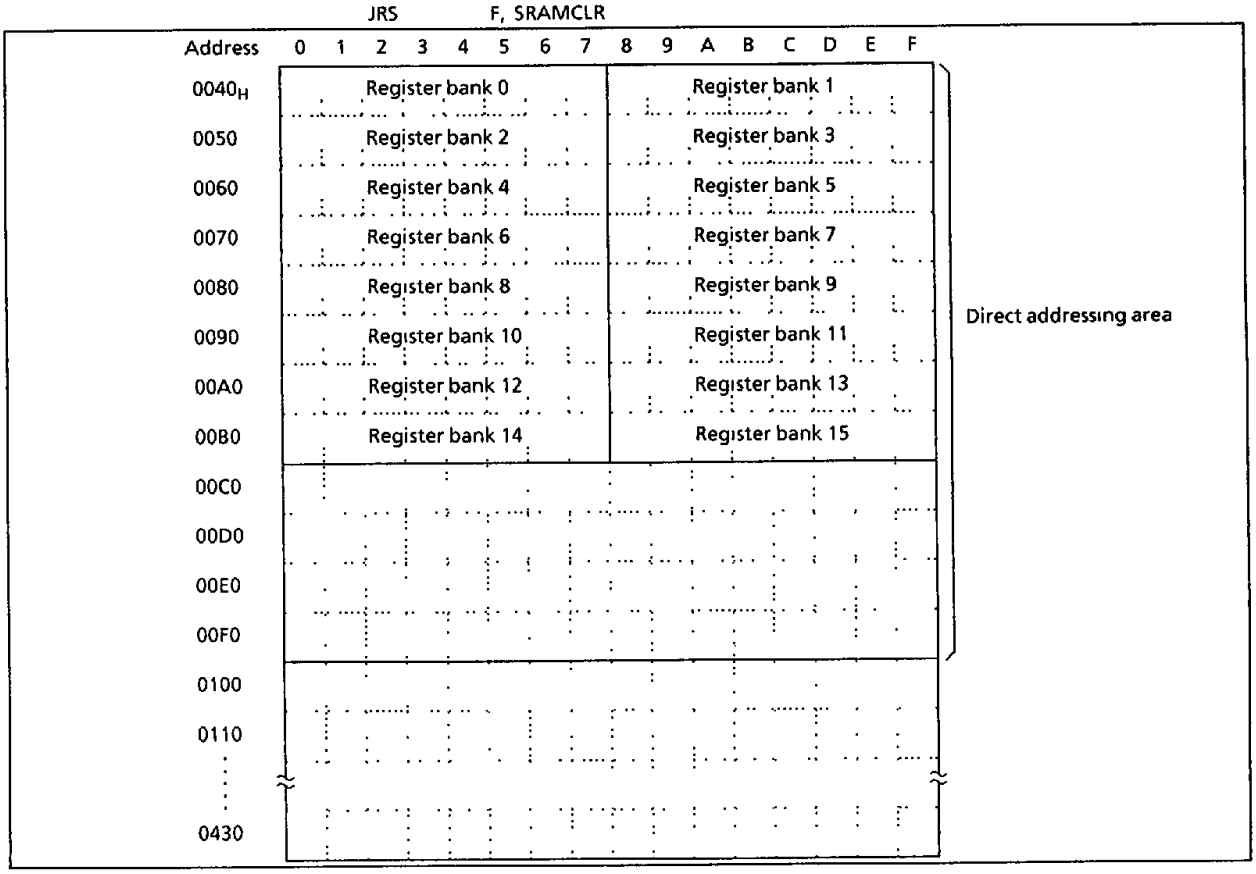


Figure 1-4. Data Memory Map

1.5 General-purpose Register Banks

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

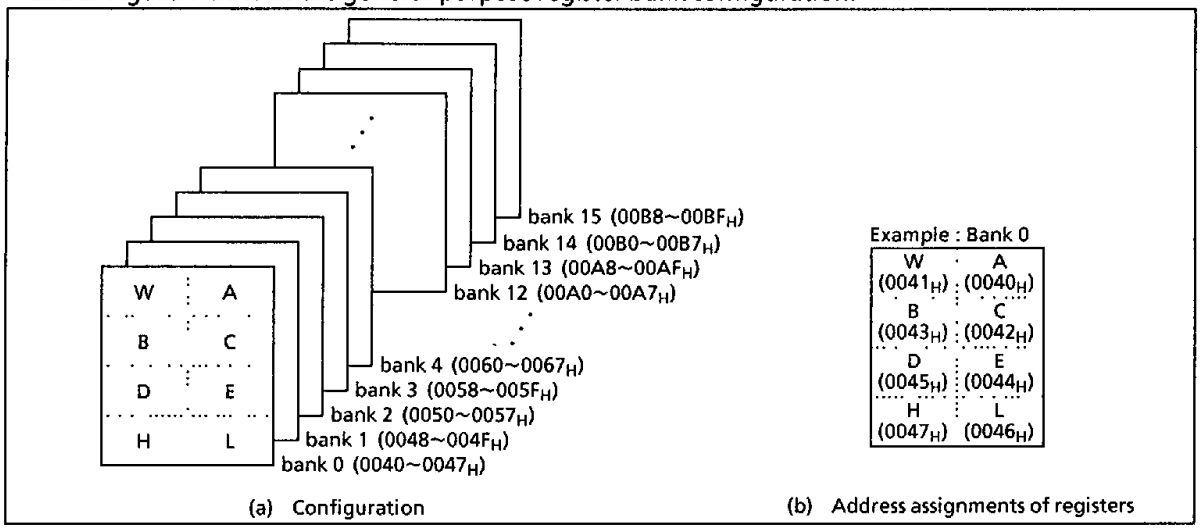


Figure 1-5. General-purpose Register Banks

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

(1) A, WA

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

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

(2) HL, DE

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

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

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

Example 2 : Block transfer

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

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

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

Example 1 : Repeat processing

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

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

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

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

Example 1 : Incrementing the RBS

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

Example 2 : Reading the RBS

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

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

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

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

1.6 Program Status Word (PSW)

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

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

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

1.6.1 Register Bank Selector (RBS)

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

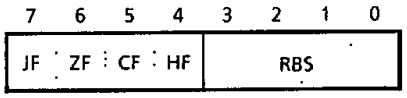


Figure 1-6. PSW (Flags, RBS) Configuration

1.6.2 Flags

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

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

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

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

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

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

The CF is set to "1" when a carry out of the MSB (most significant bit) of the result occurred during addition or when a borrow into the MSB of the result occurred during subtraction; otherwise the CF is cleared to "0". During division, this flag is set to "1" when the divisor is 00_H (divided by zero error), or when the quotient is 100_H or higher (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 1-8 shows the stacking order.

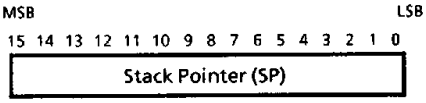


Figure 1-7. Stack Pointer

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

Example 1 : To initialize the SP

```
LD      SP, 043FH       ; SP←043F_H
```

Example 2 : To read the SP

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

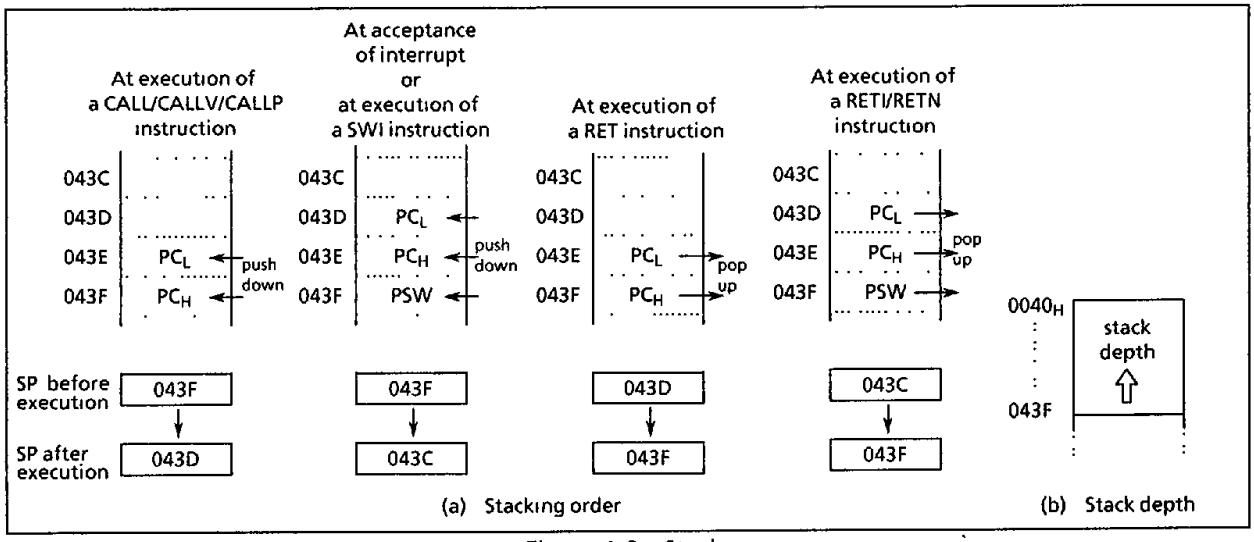


Figure 1-8. Stack

1.8 System Clock Controller

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

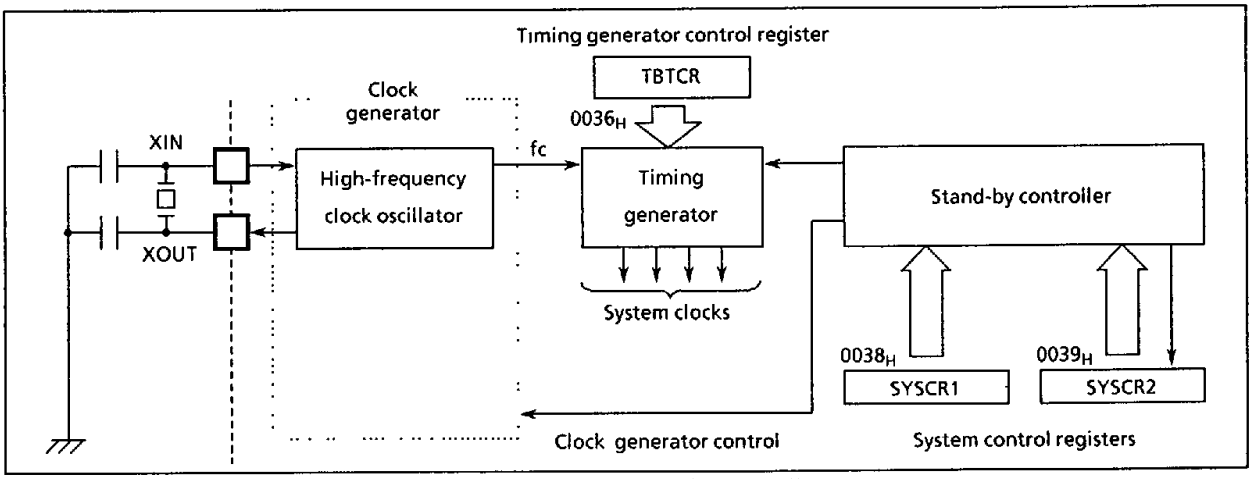


Figure 1-9. System Clock Controller

1.8.1 Clock Generator

The clock generator generates the basic clock which provides the system clocks supplied to the CPU core and peripheral hardware. It contains a oscillation circuit for the high-frequency clock.

The high-frequency (fc) clock can be easily obtained by connecting a resonator between the XIN/XOUT pins, respectively. Clock input from an external oscillator is also possible. In this case, external clock is applied to the XIN pin with the XOUT pin not connected. The 87C833/C33/H33 is not provided an RC oscillation.

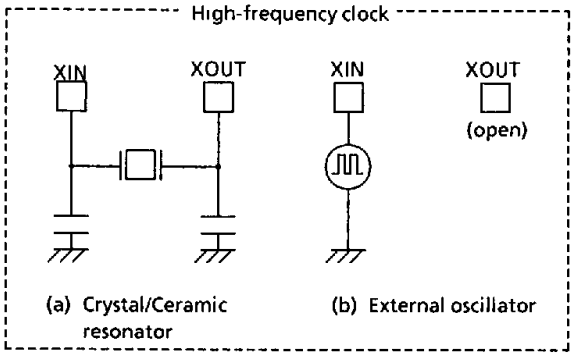


Figure 1-10. Examples of Resonator Connection

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

1.8.2 Timing Generator

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

① Generation of main system clock
② Generation of source clocks for time base timer
③ Generation of source clocks for watchdog timer
④ Generation of internal source clocks for timer/counters TC1 – TC4
⑤ Generation of warm-up clocks for releasing STOP mode
⑥ Generation of a clock for releasing reset output

(1) Configuration of Timing Generator

The timing generator consists of a 21-stage divider with a divided-by-4 prescaler, a main system clock generator, and machine cycle counters, shown in Figure 1-11 as follows. During reset and upon releasing STOP mode, the divider is cleared to "0", however, the prescaler is not cleared.

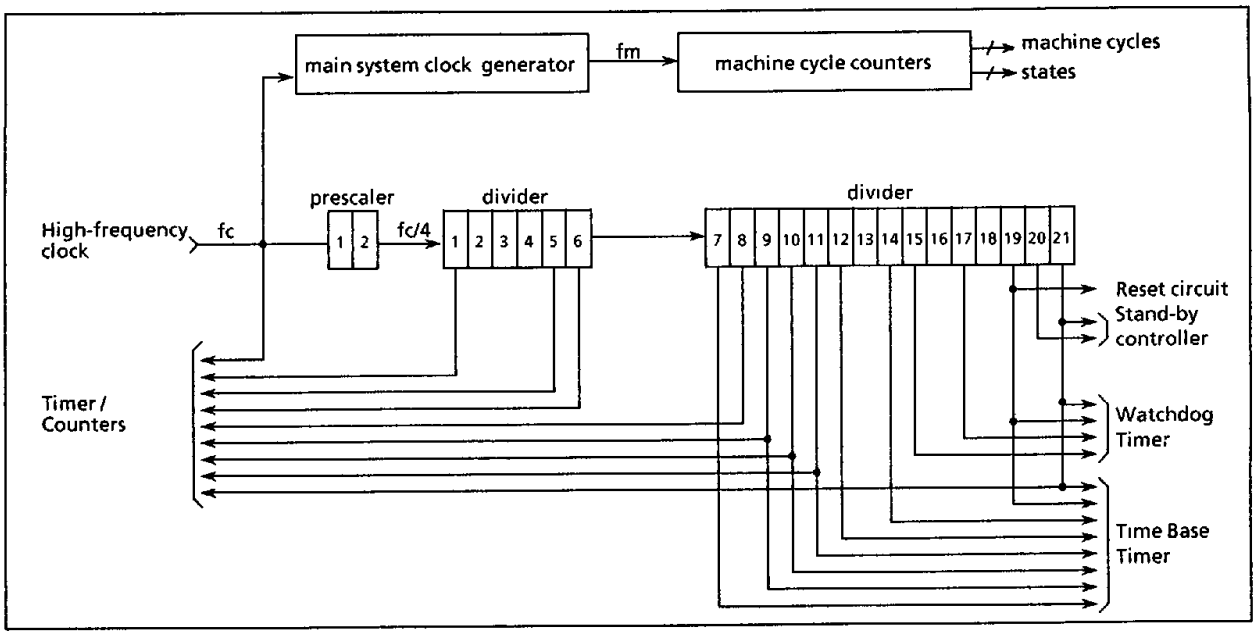


Figure 1-11. Configuration of Timing Generator

(2) Machine Cycle

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

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

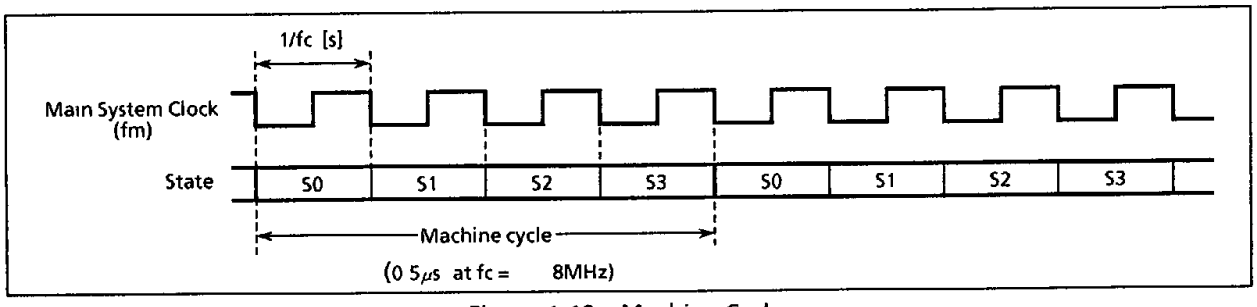


Figure 1-12. Machine Cycle

1.8.3 Stand-by Controller

The stand-by controller starts and stops the oscillation circuit for the high-frequency clock. Operating modes are controlled by the system control registers (SYSCR1, SYSCR2).

Figure 1-13 shows the operating mode transition diagram and Figure 1-14 shows the system control registers. Either the single-clock or the dual-clock mode can be selected by an option during reset.

(1) **Operating mode**

① NORMAL mode

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

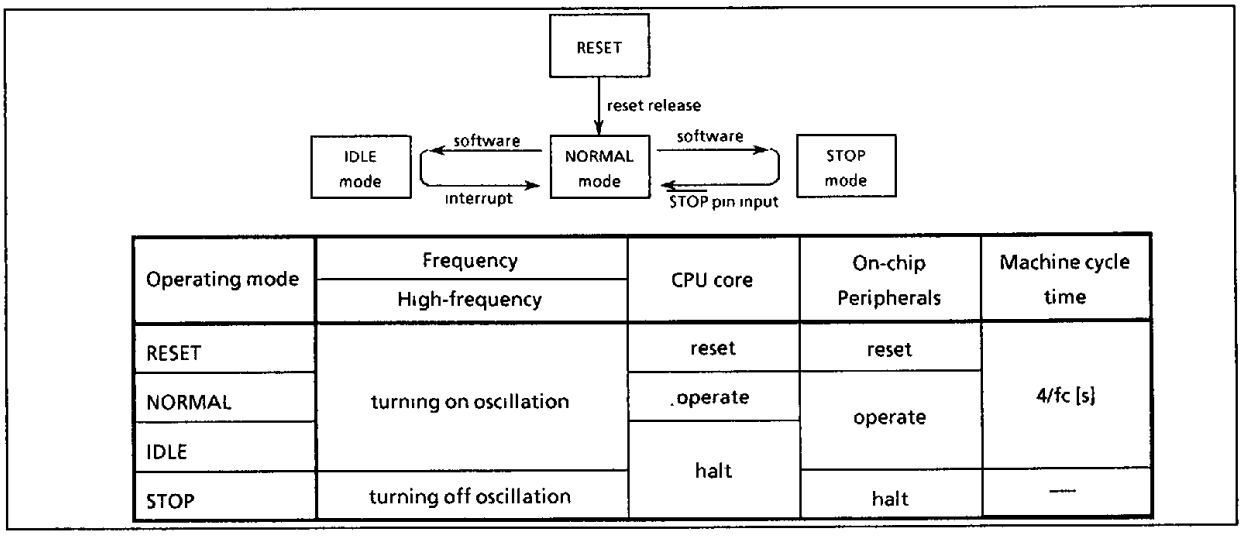
② IDLE mode

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

③ STOP 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.

STOP mode is started by setting STOP bit in the system control register 1 (SYSCR1), and STOP 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.



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

Figure 1-13. Operating Mode Transition Diagram

System Control Register 1

SYSCR1 (0038_H)

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

(Initial value: 0000 00**)

STOP	STOP mode start	0 : CPU core and peripherals remain active 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 : Reserved	
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 [s] 01 : 2^{19} / fc 1* : Reserved	

Note 1 : Always set RETM to "0" when transiting from NORMAL mode to STOP mode.
Note 2 : If 87C833/C33/H33 is moved to STOP mode while OUTEN = "0", internal inputs fix "0". Then there is a possibility to set interrupt of falling edge.
Note 3 : Bits 1 and 0 in SYSCR1 are read in as undefined data when a read instruction is executed.
Note 4 : fc ; high-frequency clock [Hz]
** ; don't care*
Note 5 : 87C833/C33/H33 returns to NORMAL mode without value of RETM, when STOP mode is retuned by input of $\overline{\text{RESET}}$ pin.

System Control Register 2

SYSCR2 (0039_H)

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

(Initial value: 1000 ****)

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

Note 1 : A reset is applied ($\overline{\text{RESET}}$ pin output goes low) if both bit 7 and bit 6 in SYSCR2 are cleared to "0".
Note 2 : Do not clear bit 7 in SYSCR2 to "0", and do not set bits 6-5 in SYSCR2 to "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.

Figure 1-14. System Control Registers

1.8.4 Operating Mode Control

(1) **STOP** mode

STOP mode is controlled by the system control register 1 (SYSCR1) and the $\overline{STOP}$ pin input. The $\overline{STOP}$ pin is also used both as a port P20 and an $\overline{INT5}$ (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 Oscillation is turned off, and all internal operations are halted.

② The data memory, registers and port output latches are all held in the status in effect before STOP mode was entered. The port output can be select either output hold or high-impedance by setting OUTEN (bit 4 in SYSCR1).

③ The divider of the timing generator is cleared to "0".

④ The program counter holds the address of the instruction following the instruction which started 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 $\overline{STOP}$ pin high. This mode is used for capacitor back-up when the main power supply is cut off and for long term battery back-up.

When the $\overline{STOP}$ 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 $\overline{STOP}$ pin input is low. The following one method can be used for confirmation:

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

Example : Starting STOP mode with an INT5 interrupt.

```
PINT5 :   TEST   (P2) . 0                ;  To reject noise, STOP mode does not start if
          JRS    F, SINT5                    port P20 is at high
          LD     (SYSCR1), 01010000B   ;  Sets up the level-sensitive release mode.
          SET    (SYSCR1) . 7          ;  Starts STOP mode
          LDW    (IL) 1110011101010111B ;  iL12, 11, 7, 5, 3 ← 0 (clears interrupt latchs)
SINT5 :   RETI
```

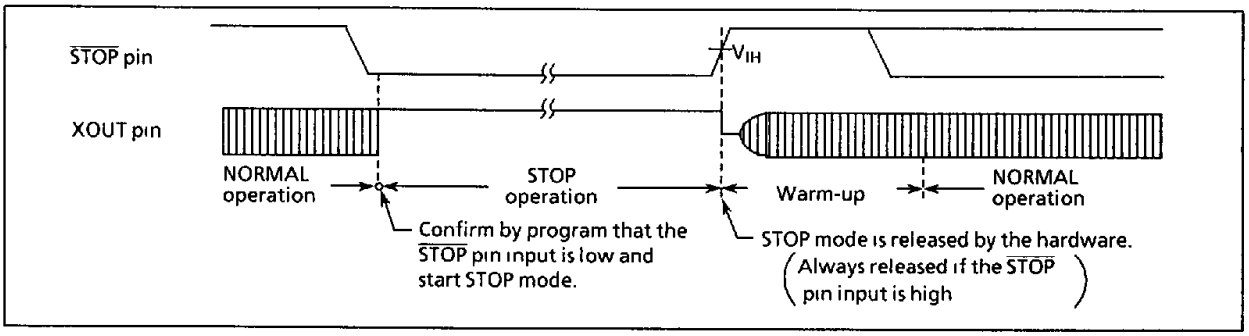


Figure 1-15. Level-sensitive Release Mode

Note1 : After warming up is started, when $\overline{STOP}$ 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 $\overline{STOP}$ 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)
```

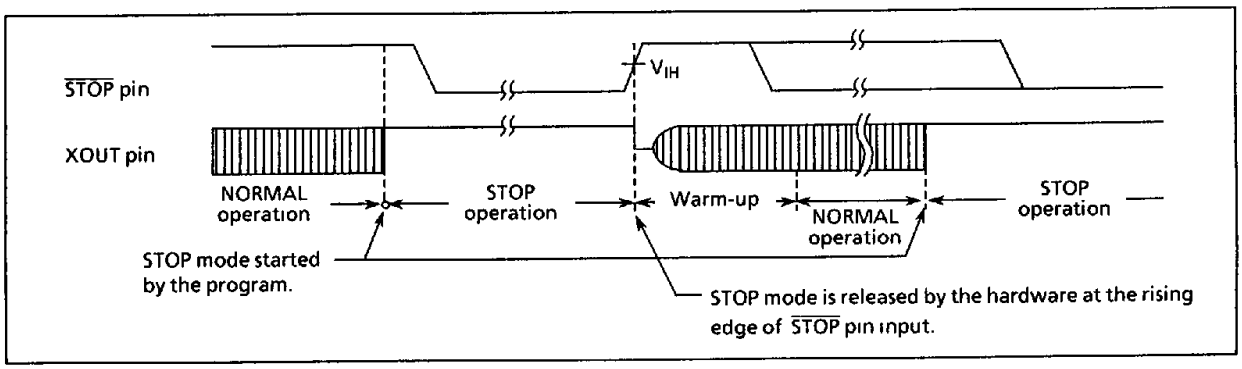


Figure 1-16. Edge-sensitive Release Mode

STOP mode is released by the following sequence:

① 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".

WUT	At fc = 4.194304MHz	At fc = 8MHz
$3\times2^{19}/fc$ [s]	375 [ms]	196.6 [ms]
$2^{19}/fc$	125	65.5

Table 1-1. Warming-up Time example

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

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

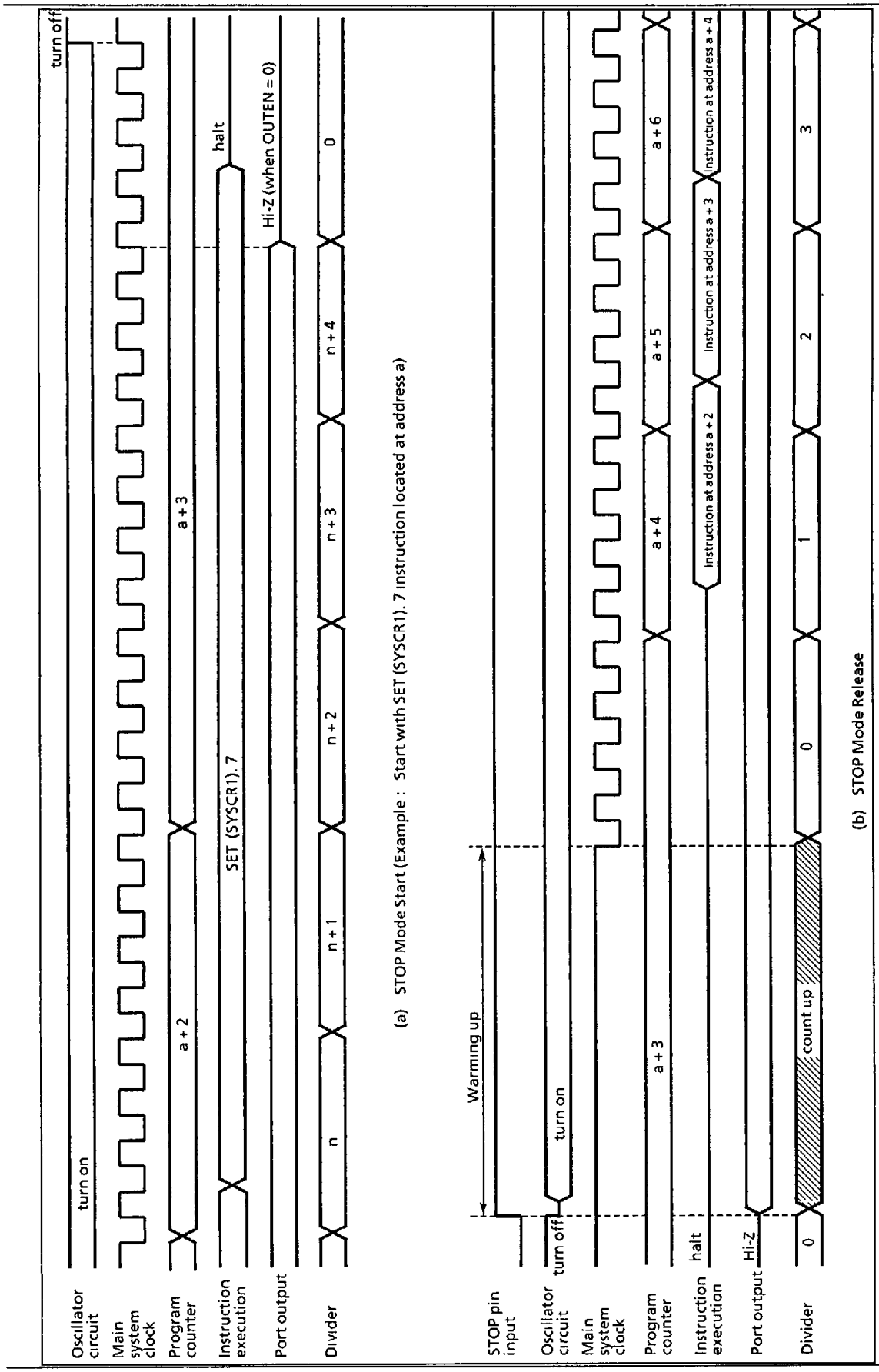


Figure 1-17. STOP Mode Start / Release

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

(2) **IDLE** mode

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 to NORMAL mode.

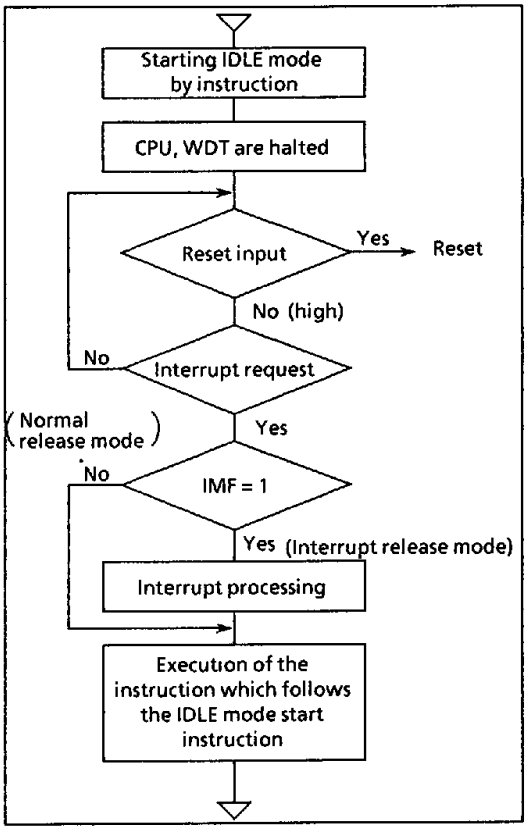


Figure 1-18. IDLE Mode

a. Normal release mode (IMF = "0")
IDLE mode is released by any interrupt source enabled by the individual interrupt enable flag (EF). Execution resumes with the instruction following the IDLE mode start instruction (e.g. [SET (SYSCR2).4]). Normally, IL (Interrupt Latch) of interrupt source to release IDEL 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). After the interrupt is processed, the execution resumes from the instruction following the instruction which started IDLE mode.

IDLE mode can also be released by setting the $\overline{\text{RESET}}$ pin low, which immediately performs the reset operation. After reset, the 87C833/C33/H33 are placed in NORMAL mode.

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

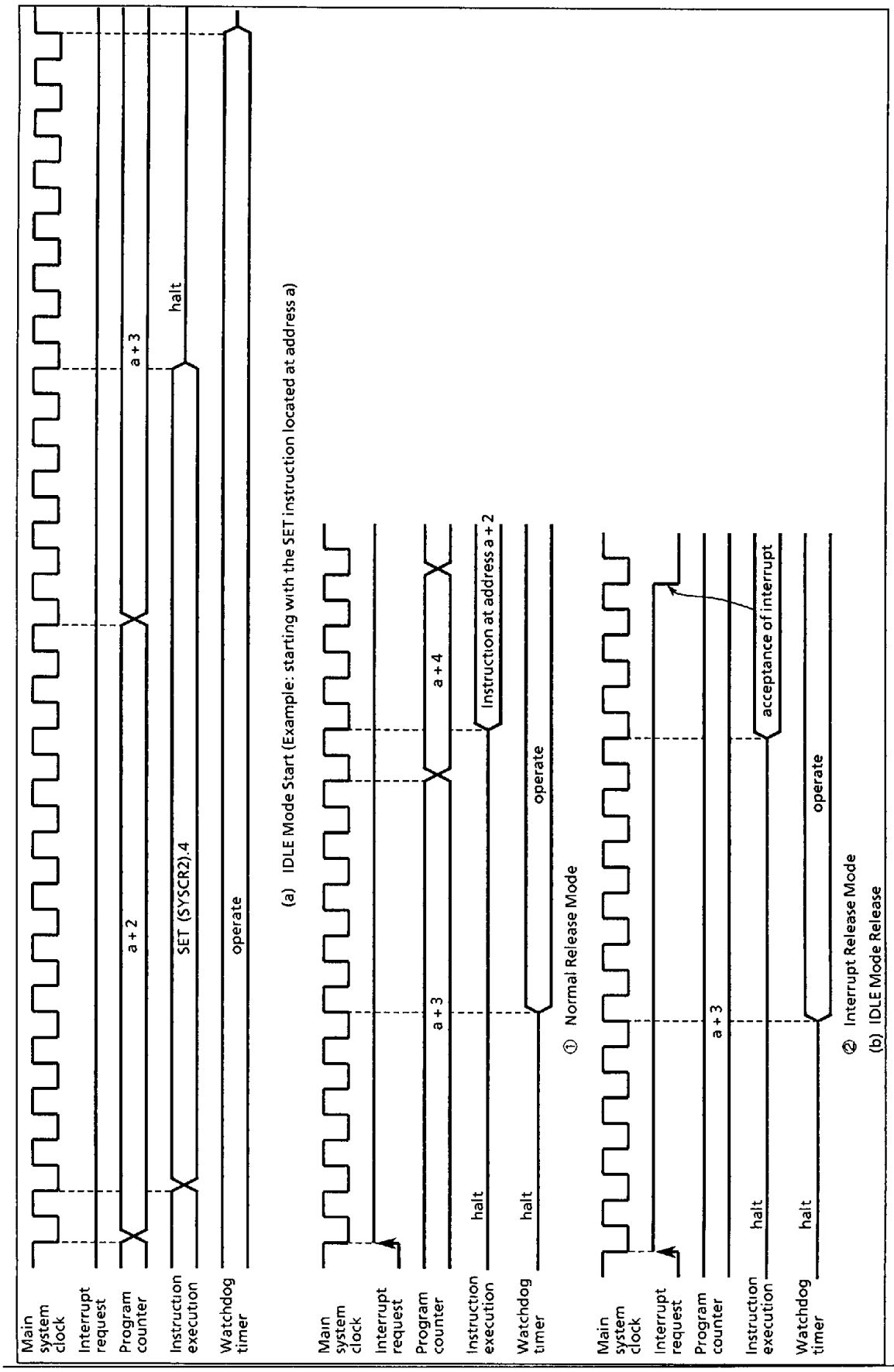


Figure 1-19. IDLE Mode Start/Release

1.9 Interrupt Controller

The 87C833/C33/H33 has a total of 14 interrupt sources: 5 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 1-20 shows the interrupt controller.

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

Table 1-2. Interrupt Sources

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

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

The interrupt latches are assigned to addresses $003C_H$ and $003D_H$ in the SFR. Each latch can be cleared to "0" individually by an instruction; however, the *read-modify-write instruction* such as bit manipulation or operation instructions *cannot be used* (Do not clear the IL2 for a watch dog timer interrupt to "0"). Thus, interrupt requests can be cancelled and initialized by the program. Note that interrupt latches cannot be set to "1" by any instruction.

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

Example 1 : Clears interrupt latches

```
LDW     (IL), 1110101010111111B     ; IL12, IL10, IL8, IL6←0
```

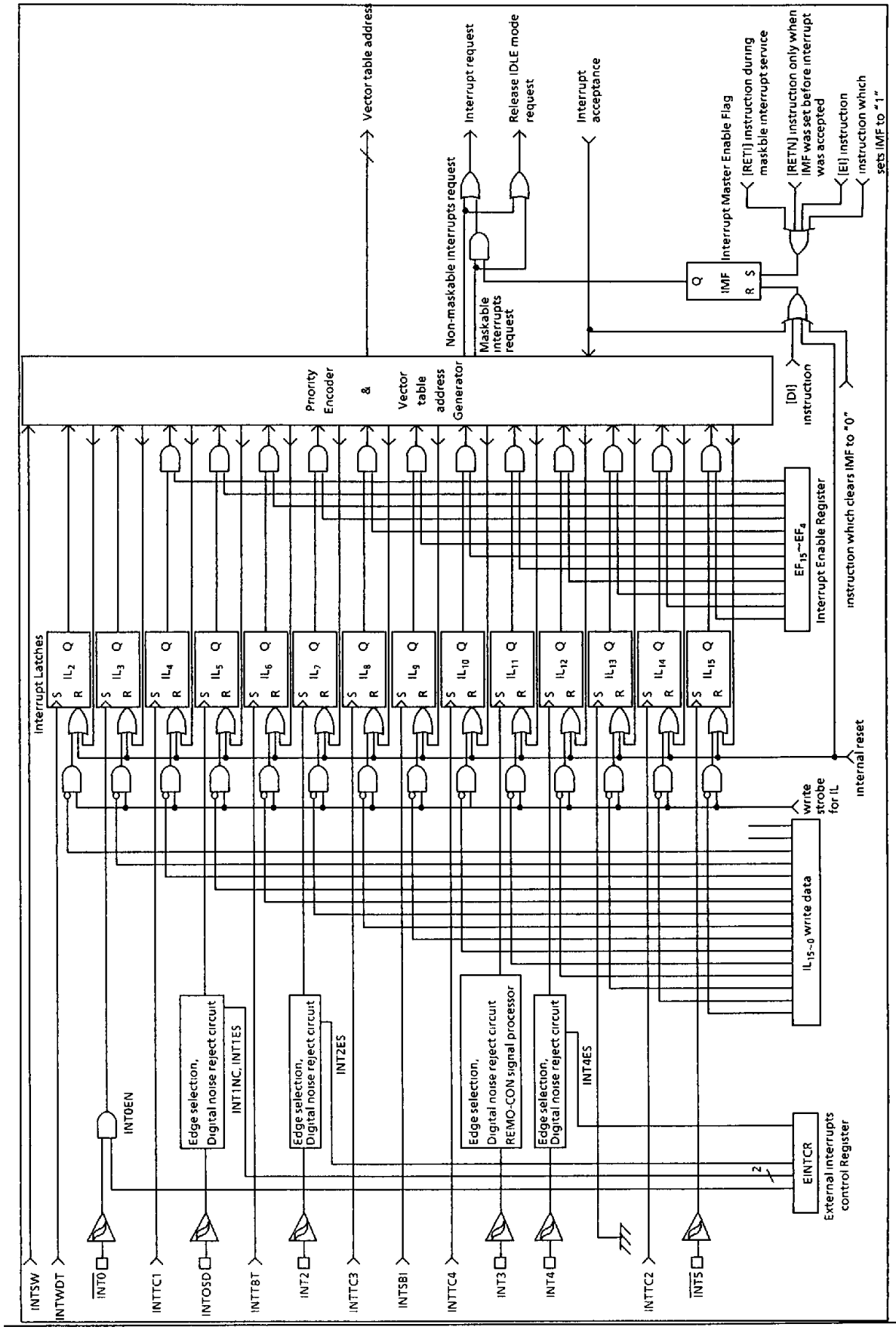


Figure 1-20. Interrupt Controller Block Diagram

Example 2 : Reads interrupt latches

```
LD      WA, (IL)        ; W←IL_H, A←IL_L
```

Example 3: Tests an interrupt latch

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

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

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

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

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

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

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

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

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

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

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

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

```
LDW     (EIR), 1110100000000001B    ;  EF15~EF13, EF11, IMF←1
```

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

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

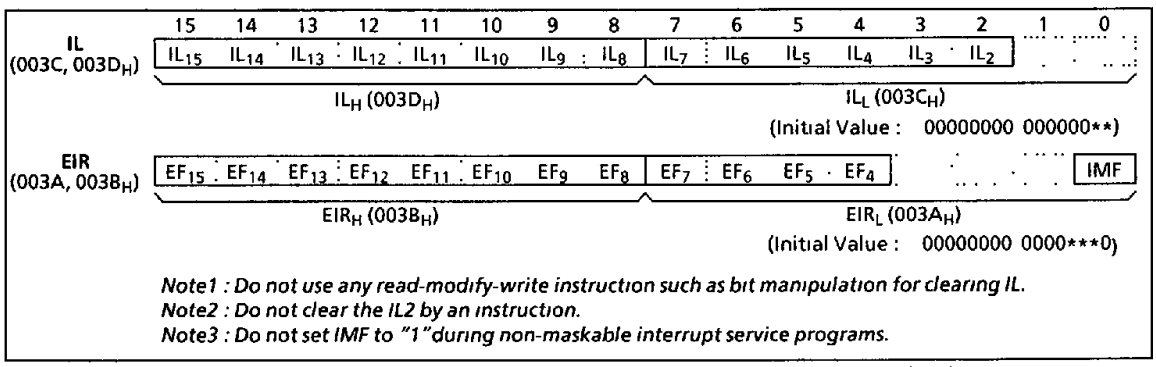


Figure 1-21. Interrupt Latch (IL) and Interrupt Enable Register (EIR)

1.9.1 Interrupt Sequence

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

(1) Interrupt acceptance processing is as follows:

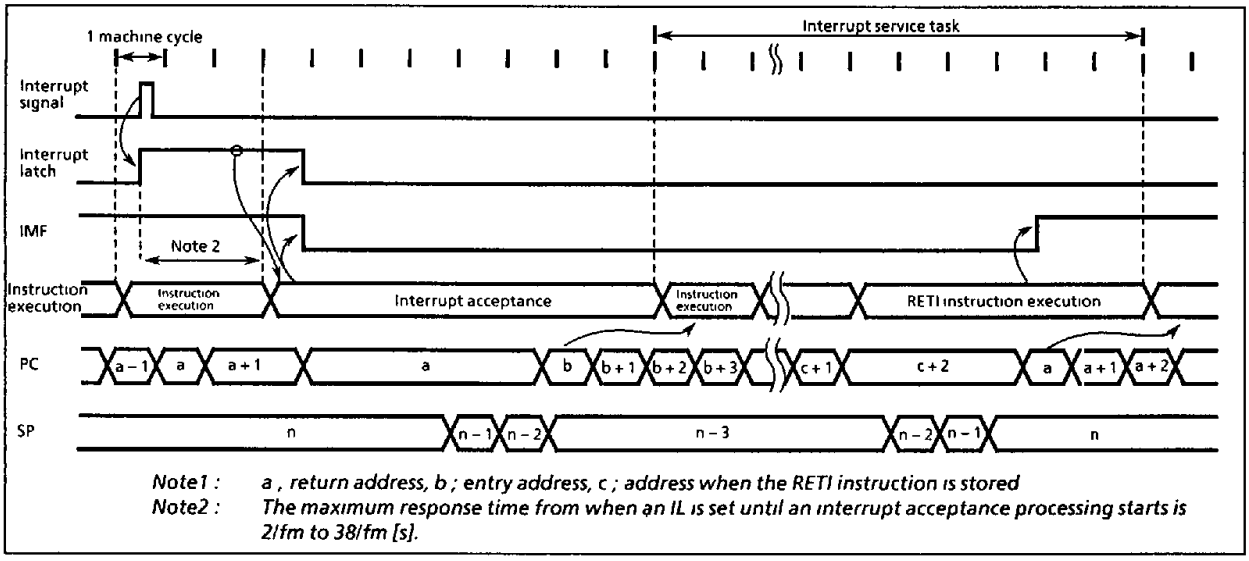
① The interrupt master enable flag (IMF) is cleared to "0" to temporarily disable the acceptance of any following maskable interrupts. When a non-maskable interrupt is accepted, the acceptance of any following interrupts is temporarily disabled.

② The interrupt latch (IL) for the interrupt source accepted is cleared to "0".

③ The contents of the program counter (return address) and the program status word are saved (pushed) onto the stack. The stack pointer is decremented 3 times.

④ The entry address of the interrupt service program is read from the vector table address, and the entry address is loaded to the program counter.

⑤ The instruction stored at the entry address of the interrupt service program is executed.

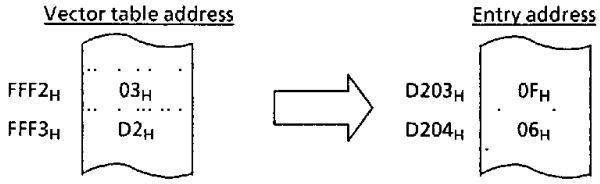


Note1 : a , return address, b ; entry address, c ; address when the RETI instruction is stored

Note2 : The maximum response time from when an IL is set until an interrupt acceptance processing starts is 2/fm to 38/fm [s].

Figure 1-22. Timing Chart of Interrupt Acceptance and Interrupt Return Instruction

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



A maskable interrupt is not accepted until the IMF is set to "1" even if a maskable interrupt of higher priority than that of the current interrupt being serviced.

When nested interrupt service is necessary, the IMF is set to "1" in the interrupt service program. In this case, acceptable interrupt sources are selectively enabled by the individual interrupt enable flags.

(2) Saving / Restoring General-purpose Register

During interrupt acceptance processing, the program counter and the program status word are automatically saved on the stack, but not the accumulator and other registers. These registers are saved by the program if necessary. Also, when nesting multiple interrupt services, it is necessary to avoid using the same data memory area for saving registers.

The following method is used to save/restore the general-purpose registers:

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

General-purpose registers can be saved at high-speed by switching to a register bank that is not in use. Normally, bank 0 is used for the main task and banks 1 to 15 are assigned to interrupt service tasks. To increase the efficiency of data memory utilization, the same bank is assigned for interrupt sources which are not nested.

The switched bank is automatically restored by executing an interrupt return instruction [RETI] or [RETN]. Therefore, it is not necessary for a program to save the RBS.

Example : Register Bank Changeover

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

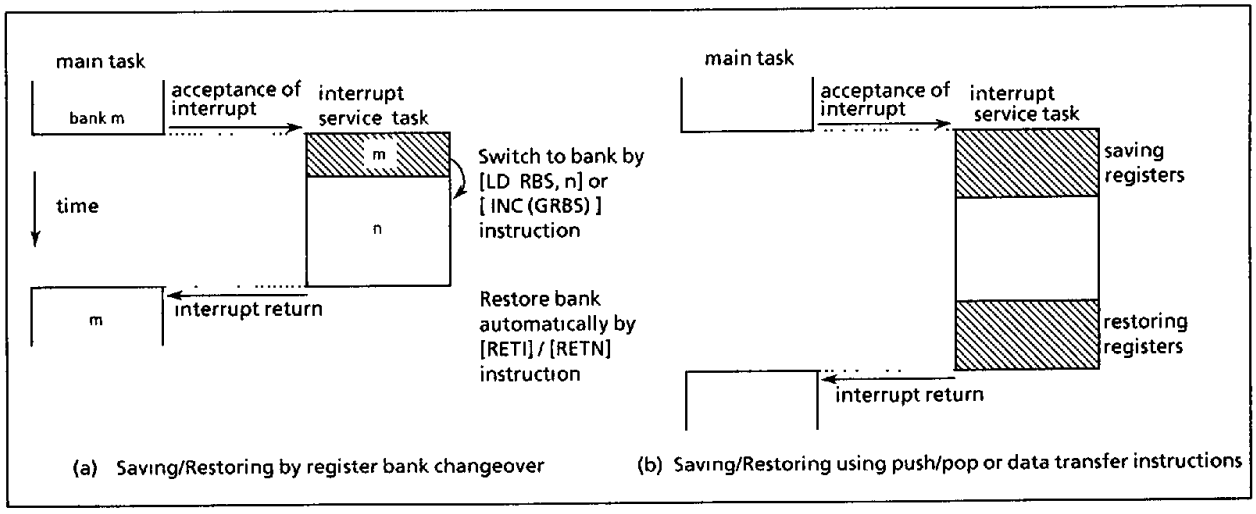


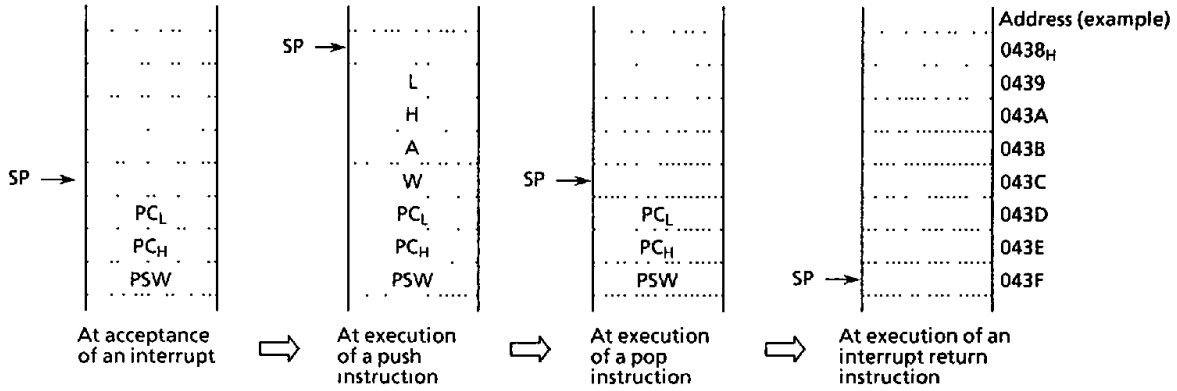
Figure 1-23. Saving/Restoring General-purpose Registers

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 instructions:
Data transfer instructions can be used to save only a specific general-purpose register during processing of a single interrupt.

Example : Saving/restoring a register using data transfer instructions

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

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

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

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

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

1.9.2 Software Interrupt (INTSW)

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

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

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

① Address Error Detection

FF_H is read if for some cause such as noise the CPU attempts to fetch an instruction from a non-existent memory address. Code FF_H is the SWI instruction, so a software interrupt is generated and an address error is detected. The address error detection range can be further expanded by writing FF_H to unused areas of the program memory. the address trap reset is generated in case that an instruction fetch from a port of RAM area or SER area .

Note : The fetch data from addresses $7F80_H$ to $7FFF_H$ (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 Interrup

The 87C833/C33/H33 have five external interrupt inputs ($\overline{INT0}$, INT2, INT3, INT4, and $\overline{INT5}$). Three of these are equipped with digital noise reject circuits (pulse inputs of less than a certain time are eliminated as noise). Edge selection is also possible with INT2, INT3 and INT4.

The $\overline{INT0}$/P50 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.

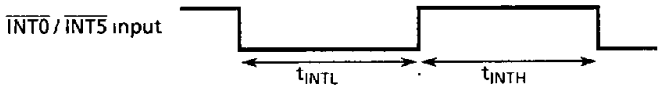
Edge selection, noise reject control and $\overline{INT0}$/P50 pin function selection are performed by the external interrupt control register (EINTCR). When INT0EN = 0, the IL_3 will not be set even if the falling edge of $\overline{INT0}$ pin input is detected.

Edge selection and noise rejection control for INT3 pin input are performed by the Remote control signal processor control registers (refer to the selection of the Remote control signal processer.)

Source	Pin	Secondary function pin	Enable conditions	Edge	Digital noise reject
INT0	$\overline{INT0}$	P50/PWM8	IMF = 1, INT0EN = 1	falling edge	— (hysteresis input)
INT2	INT2	P53/TC1	IMF · EF_7 = 1	falling edge	Pulses of less than 7/fc [s] are
INT3	INT3	P30/RX1N	IMF · EF_{11} = 1	or	eliminated as noise. Pulses of 24/fc [s]
INT4	INT4	P36/SO	IMF · EF_{12} = 1	rising edge	or more are considered to be signals.
INT5	$\overline{INT5}$	P20/STOP	IMF · EF_{15} = 1	falling edge	— (hysteresis input)

Note 1 : The noise reject function is also affected for timer/counter input (TC1 and TC3 pins).

Note 2 : The pulse width (both "H" and "L" level) for input to the $\overline{INT0}$ and $\overline{INT5}$ pins must be over 1 machine cycle.



Note 3 : 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]

Table 1-3. External Interrupts

EINTCR (0037_H)

7	6	5	4	3	2	1	0
	INT0EN		INT4ES	(TC3ES)	INT2ES		

(Initial value : *0*0 000*)

INT0EN	P50/$\overline{INT0}$ pin configuration	0 : P50 input/output port 1 : $\overline{INT0}$ pin (Port P50 should be set to an input mode)	write only
INT4 ES INT2 ES	INT4, INT2 edge select	0 : Rising edge 1 : Falling edge	

Note 1 : * ; *don't care*

Note 2 : *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 3 : *In order to change of external interrupt input by rewriting the contents of INT2ES and INT4ES during NORMAL mode, clear interrupt latches of external interrupt inputs (INT2 and INT4) after 8 machine cycles from the time of rewriting.*

Note 4 : *In order to change an edge of timer counter input by rewriting the contents of INT2ES and INT4ES during NORMAL mode, rewrite the contents after timer counter is stopped (TC*s = 0), that is, interrupt disable state. Then, clear interrupt laches of external interrupt inputs (INT2 and INT4) after 8 machine cycles from the time of rewriting to change to interrupt enable state. Finally, start timer counter.*

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

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

Note 5 : *If changing the contents of INT1ES during NORMAL mode, interrupt latch of external interrupt input INT1 must be cleared after 14 machine cycles (when INT1NC = 1) or 50 machine cycles (when INT1NC = 0) from the time of changing.*

Note 6 : *EINTCR is write-only register and must not be used with any of read-modify-write instructions.*

Note 7 : *Always write "0" to bit7, 5, 1, 0 in EINTCR.*

Note 8 : *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 (C833/C33/H33):

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

Figure 1-24. External Interrupt Control Register

1.10 Watchdog Timer (WDT)

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

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

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

1.10.1 Watchdog Timer Configuration

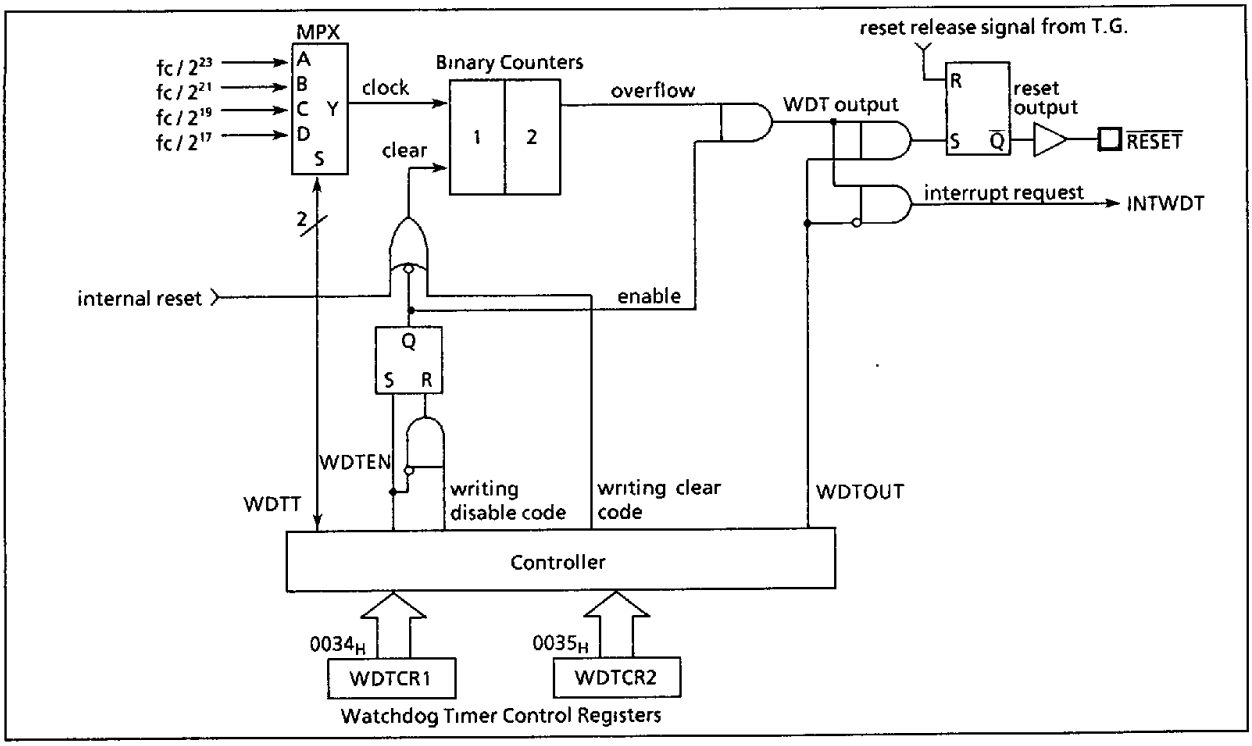


Figure 1-25. Watchdog Timer Configuration

1.10.2 Watchdog Timer Control

Figure 1-26 shows the watchdog timer control registers (WDTCR1, WDTCR2). The watchdog timer is automatically enabled after reset.

(1) Malfunction detection methods using the watchdog timer

The CPU malfunction is detected as follows:

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

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

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

The watchdog timer temporarily stops counting in STOP mode (including warm-up) or IDLE mode, and automatically restarts (continues counting) when 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 (0034H)

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

(Initial value : **** 1001)

WDTEN	Watchdog timer enable/disable	0 : Disable (It is necessary to write the disable code to WDTCR2) 1 : Enable	write only
WDTT	Watchdog timer detection time	00 : 2^{25}/fc [s] (4.194 s at fc = 8MHz) 01 : 2^{23}/fc (1.048 s at fc = 8MHz) 10 : 2^{21}/fc (262.1 ms at fc = 8MHz) 11 : 2^{19}/fc (65.5 ms at fc = 8MHz)	
WDTOUT	Watchdog timer output select	0 : Interrupt request 1 : Reset output	

Note 1 : WDTOUT cannot be set to "1" by program after clearing WDTOUT to "0"
*Note 2 : fc ; High-frequency clock [Hz] * , don't care*
Note 3 : WDTCR1 is a write-only-register and must not be used with any of read-modify-write instructions

Watchdog Timer Control Register 2

WDTCR2 (0035H)

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 1-26. Watchdog Timer Control Registers

(2) Watchdog Timer Enable

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

Example : Enables watchdog timer

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

(3) Watchdog Timer Disable

The watchdog timer is disabled by writing the disable code ($B1_H$) to WDTCR2 after clearing WDTEN (bit 3 in WDTCR1) to "0". The watchdog timer is not disabled if this procedure is reversed and the disable code is written to WDTCR2 before WDTEN is cleared to "0". The watchdog timer is halted temporarily in STOP mode (including warm-up) and IDLE mode, and restarts automaticallyafter 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) low to reset the internal hardware and the external circuits. The reset output time is 2^{20}/fc [s] (131ms at fc = 8MHz)

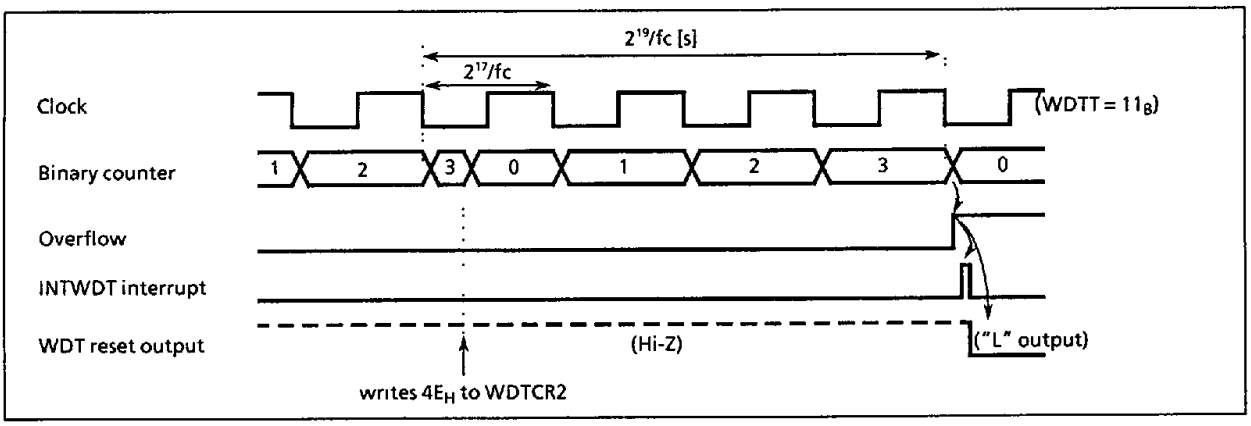


Figure 1-27. Watchdog Timer Interrupt / Reset

1.11 Reset Circuit

The TLCS-870 Series has four types of reset generation procedures: an external reset input, an address-trap-reset, a watchdog timer reset and a system-clock-reset. Table 1-4 shows on-chip hardware initialization by reset action. The internal source reset circuit (watchdog timer reset, address trap reset, and system clock reset) is not initialized when power is turned on. Thus, output from the $\overline{\text{RESET}}$ pin may go low (2^{20}/fc [s] (131ms at 8MHz) when power is turned on.

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

Table 1-4. Initializing Internal Status by Reset Action

1.11.1 External Reset Input

When the $\overline{\text{RESET}}$ pin is held at low for at least 3 machine cycles (12/fc [s]) with the power supply voltage within the operating voltage range and oscillation stable, a reset is applied and the internal state is initialized.

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

The $\overline{\text{RESET}}$ pin contains a Schmitt trigger (hysteresis) with an internal pull-up resistor. A simple power-on-reset can be applied by connecting an external capacitor and a diode.

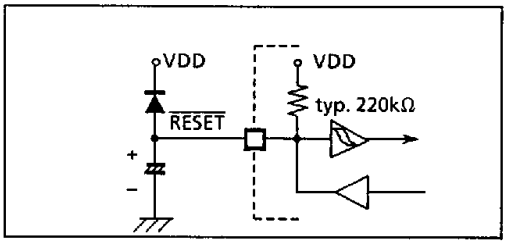


Figure 1-28. Simple Power-on-Reset Circuitry

1.11.2 Address-Trap-Reset

If a CPU malfunction occurs and an attempt is made to fetch an instruction form the RAM or the SFR area (addresses 0000_H - $043F_H$), and address-trap-reset will be generated. Then, the $\overline{\text{RESET}}$ pin output will go low. The reset time is 2^{20}/fc [s] (131ms at fc = 8MHz).

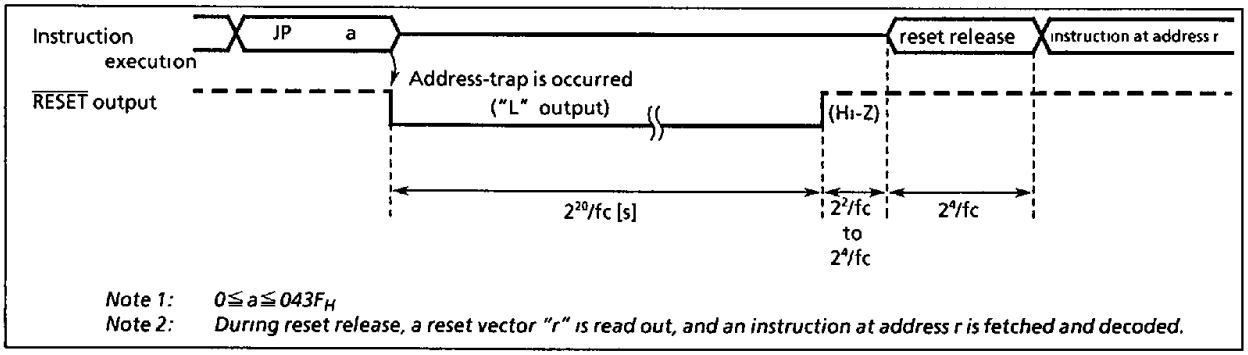


Figure 1-29. Address-Trap-Reset

1.11.3 Watchdog Timer Reset

Refer to Section "1.10 Watchdog Timer".

1.11.4 System-Clock-Reset

Clearing both bits 7 and 6 in SYSCR2 to "0" stops high-frequency oscillation, and causes the MCU to deadlock. This can be prevented by automatically generating a reset signal whenever (bit7 in SYSCR2) = (bit6 in SYSCR2) = 0 is detected to continue the oscillation. Then, the $\overline{\text{RESET}}$ pin output goes low from high-impedance. The reset time is 2^{20}/fc [s] (131ms at fc = 8MHz).

2. ON-CHIP PERIPHERALS FUNCTIONS

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

The TLCS-870 Series uses the memory mapped I/O system and all peripherals control and data transfers are performed through the special function registers (SFR) and data buffer registers (DBR).
The SFR are mapped to addresses 0000_H – $003F_H$, and the DBR to addresses $0F80_H$ – $0FFF_H$.
Figure 2-1 shows the list of the 87C833/C33/H33 SFRs and DBRs.

Address	Read	Write
0000_H	reserved	
01	reserved	
02	P2 Port	
03	P3 Port	
04	P4 Port	
05	P5 Port	
06	P6 Port	
07	P7 Port	
08	reserved	
09	reserved	
0A	reserved	
0B	reserved	
0C	–	P4CR (P4 I/O control)
0D	–	P6CR (P6 I/O control)
0E	–	CMPCR (Comparator)
0F	CMPDR (Comparator input data register input control)	
10	–	$TREG1A_L$ (Timer register 1A)
11	–	$TREG1A_H$
12	$TREG1B_L$ (Timer register 1B)	
13	$TREG1B_H$	
14	–	TC1CR (TC1 control)
15	–	TC2CR (TC2 control)
16	–	$TREG2_L$ (Timer register 2)
17	–	$TREG2_H$
18	TREG3A (Timer register 3A)	
19	TREG3B (Timer register 3B)	–
1A	–	TC3CR (TC3 control)
1B	–	TREG4 (Timer register 4)
1C	–	TC4CR (TC4 control)
1D	reserved	
1E	reserved	
1F	reserved	

Address	Read	Write
0020_H	–	SBICR1 (SBI control 1)
21	SBIDBR (SBI data buffer)	
22	–	I2CAR (I^2C-bus address)
23	SBISR (SBI status)	SBICR2 (SBI control 2)
24	reserved	
25	PWMSR (PWM status)	PWMCR (PWM control)
26	–	PWMDBR (PWM data buffer)
27	–	PULSECR (Pulse output control)
28	reserved	
29	reserved	
2A	reserved	
2B	reserved	
2C	reserved	
2D	reserved	
2E	reserved	
2F	reserved	
30	reserved	
31	reserved	
32	reserved	
33	reserved	
34	–	WDTCR1 (WDT control)
35	–	WDTCR2
36	–	TBTCR (TBT / TG / DVO control)
37	–	EINTCR (Exter interrupt control)
38	SYSCR1	(System control)
39	SYSCR2	
3A	EIR_L	(Interrupt enable register)
3B	EIR_H	
3C	IL_L	(Interrupt latch)
3D	IL_H	
3E	reserved	
3F	PSW (Program status word)	RBS (Register bank selector)

(a) Special Function Registers

Address	Read	Write
$0F80_H$	–	
	–	
90	DCTR (OSD display-line countr)	
	–	
94	Display memory	OSD control registers
95	Display memory	
96	Display on/off	
9B	–	
0F9C	reserved	
	reserved	
0FCF	reserved	
0FD0	RXCR1 (Remo-con control1)	
D1	RXCR2 (Remo-con control2)	
D2	RXCTR (Remo-con pulse width counter)	–
D3	RXDBR (Remo-con received data buffer)	–
D4	RXSR (Remo-con status)	–
D5	reserved	
	reserved	
0FFF	reserved	

(b) Data Buffer Registers

Note 1 : Do not access reserved areas by the program.
Note 2 : – : Cannot be accessed.
Note 3 : When defining address $003F_H$ with assembler symbols, use GPSW and GRBS.
Note 4 : Write-only registers and interrupt latches cannot use the read-modify-write instructions (bit manipulation instructions such as SET, CLR, etc. and logical operation instructions such as AND, OR, etc.)
Note 5 : SB1 : Serial interface.
PWM : Pulse Width Maculation
OSD : On screen display
Remo-con : Remote control

Figure 2-1. SFR & DBR

2.2 I/O Ports

The 87C833/C33/H33 has 6 parallel input/output ports (34pins) as follows:

	Primary Function	Secondary Functions
Port P2	1-bit I/O port	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/output, and serial bus interface input/output, slicer interface input
Port P4	8-bit I/O port	pulse width modulation output
Port P5	8-bit I/O port	pulse width modulation output, pulse output, and comparator input, external interrupt input, timer/counter input
Port P6	R, G, B and Y/BL output from OSD circuitry, R.G.B and Y/BL input	8-bit I/O port
Port P7	2-bit I/O port	horizontal synchronous pulse input and vertical synchronous pulse input to OSD circuitry

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

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

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

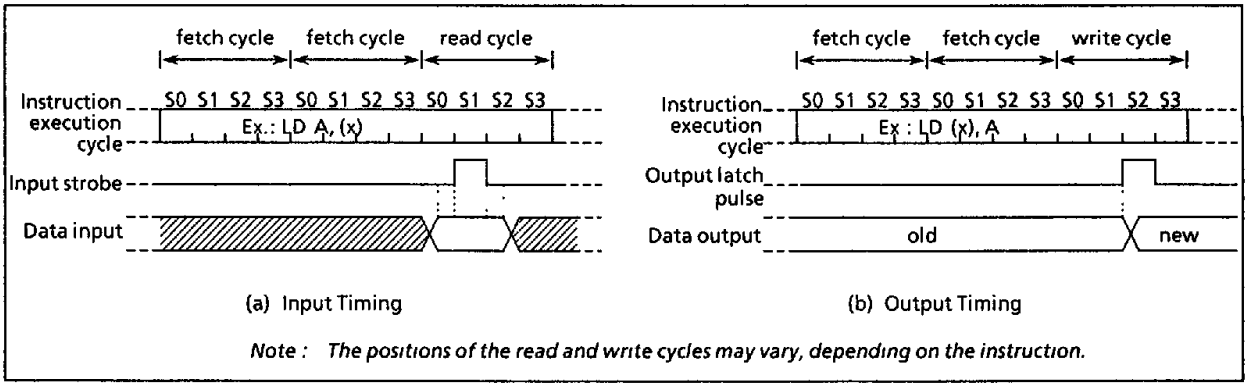


Figure 2-2. Input/Output Timing (Example)

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

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

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

2.2.1 Port P2 (P20)

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

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 1 in P2 are read in as undefined data.

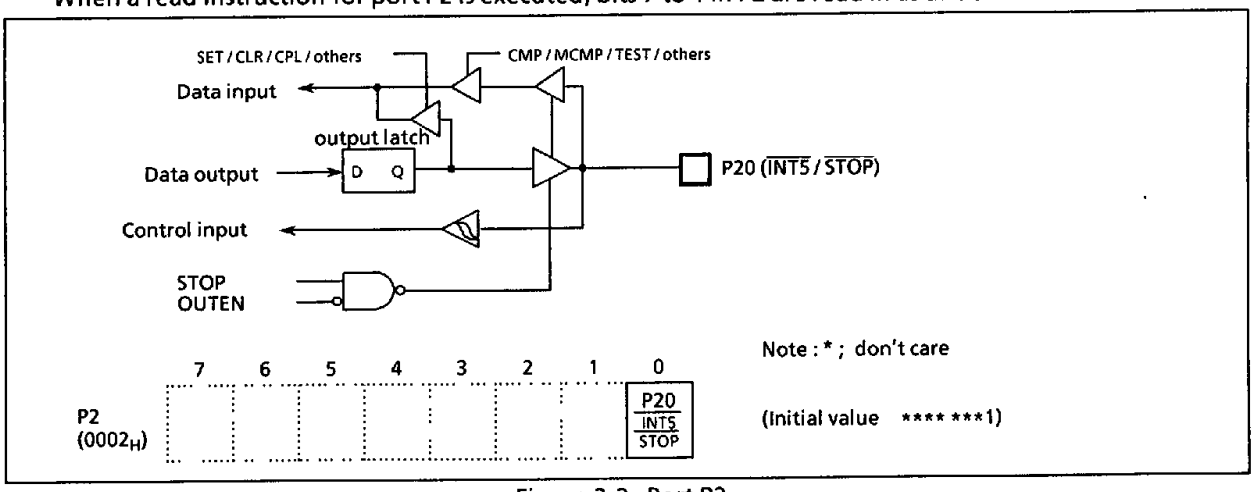


Figure 2-3 Port P2

2.2.2 Port P3 (P36 - P30)

Port P3 is a 7-bit input/output port, and is also used as serial bus interface input/output, an exrernal interrupt input a timer/counter input, and Remote-control signal input, siler interface 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←5AH
```

Example 2: Inverts the output of the lower 4bits (P33 - P30) in port P3.

```
XOR     (P3), 00001111B     ; P33~P30←P33~P30
```

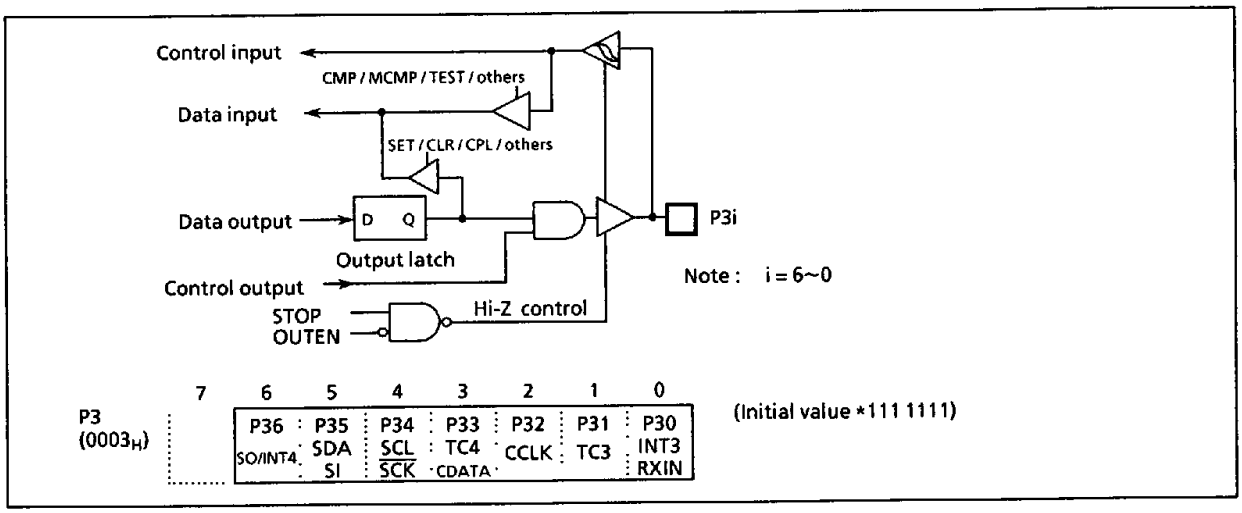


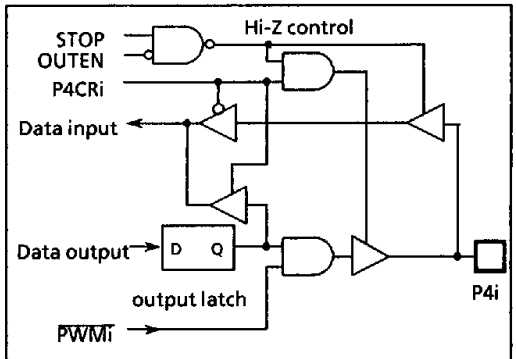
Figure 2-4 Port P3

2.2.3 Port P4 (P47 - 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. Therfore initial output data should be written into the output latch before setting P4CR. Port P4 is also used as a pulse iwdth 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 mixed, the contents of output latch setting input mode may be changed by executing bit manipulation instructions.



Note : i = 7~0

P4 (0004_H)

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

(Initial value : 1111 1111)

P4CR ($000C_H$)

7	6	5	4	3	2	1	0

(Initial value : 0000 0000)

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

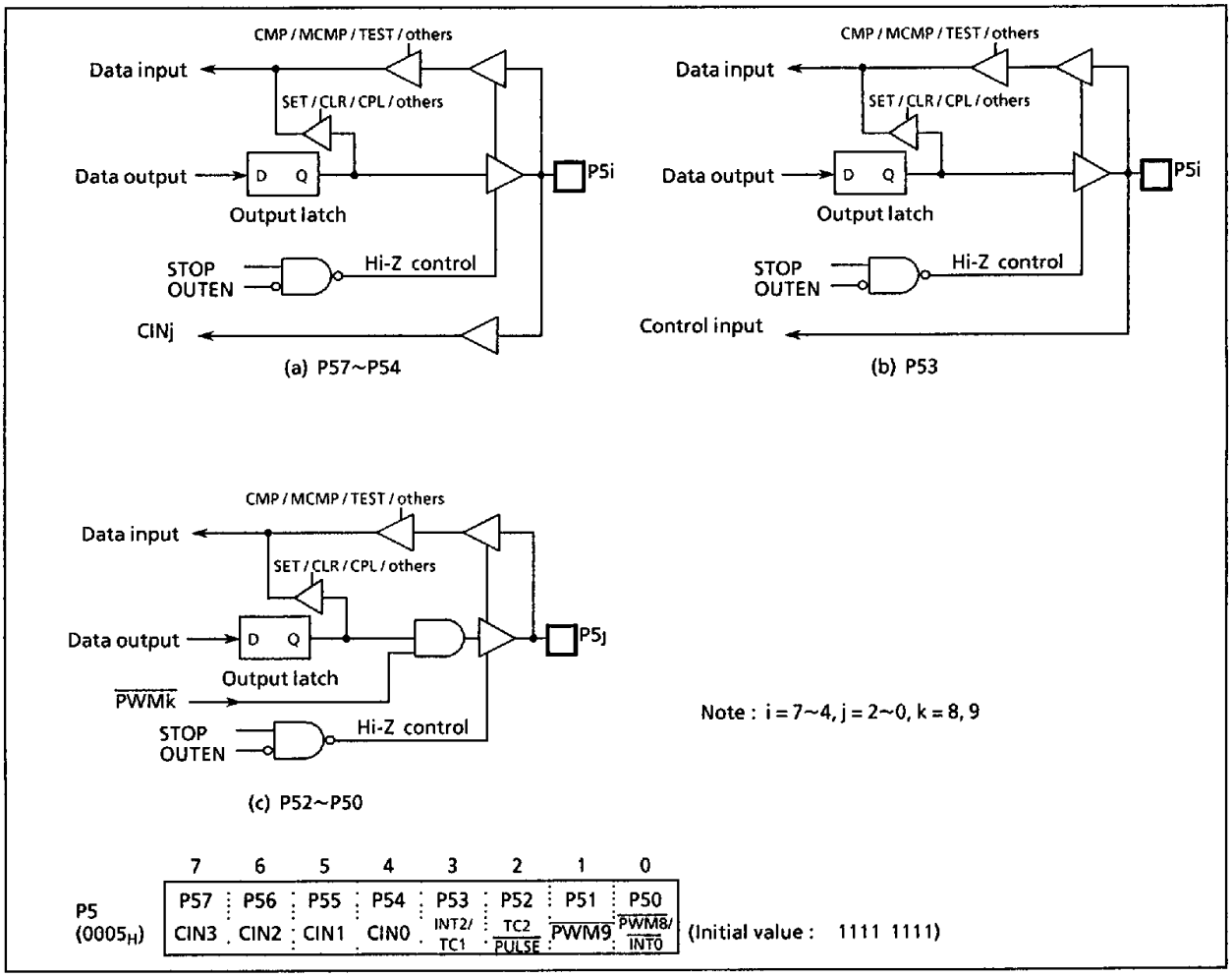
Figure 2-5. Ports P4 and P4CR

2.2.4 Port P5 (P57 - P50)

Port P5 is an 8-bit input/output port, and is also used as comparator input, a pulse output, external interrupt, timer/counter input/output, and a pulse width nodulation (PWM) output. When used as an input port or a secondary function pin, the output latch should be set to "1". The output latches are initialized to "1" during reset.

Example : Cleare of P53 pin ("L" output)

```
CLR (P5) . 3  ;  P53←0
```



	7	6	5	4	3	2	1	0
P5 (0005_H)	P57	P56	P55	P54	P53	P52	P51	P50
	CIN3	CIN2	CIN1	CIN0	INT2/ TC1	TC2 PULSE	PWM9	PWM8/ INT0

(Initial value : 1111 1111)

Figure 2-6. Ports P5

2.2.5 Port P6 (P67 - 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.

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.

Port P6 is used as an on screen display (OSD) output (R, G, B, and Y/BL signal). When used as an output pin, the output pins should be set to the output mode and beforehand the port P6 data selection register (P67S - P64S) should be set to "1".

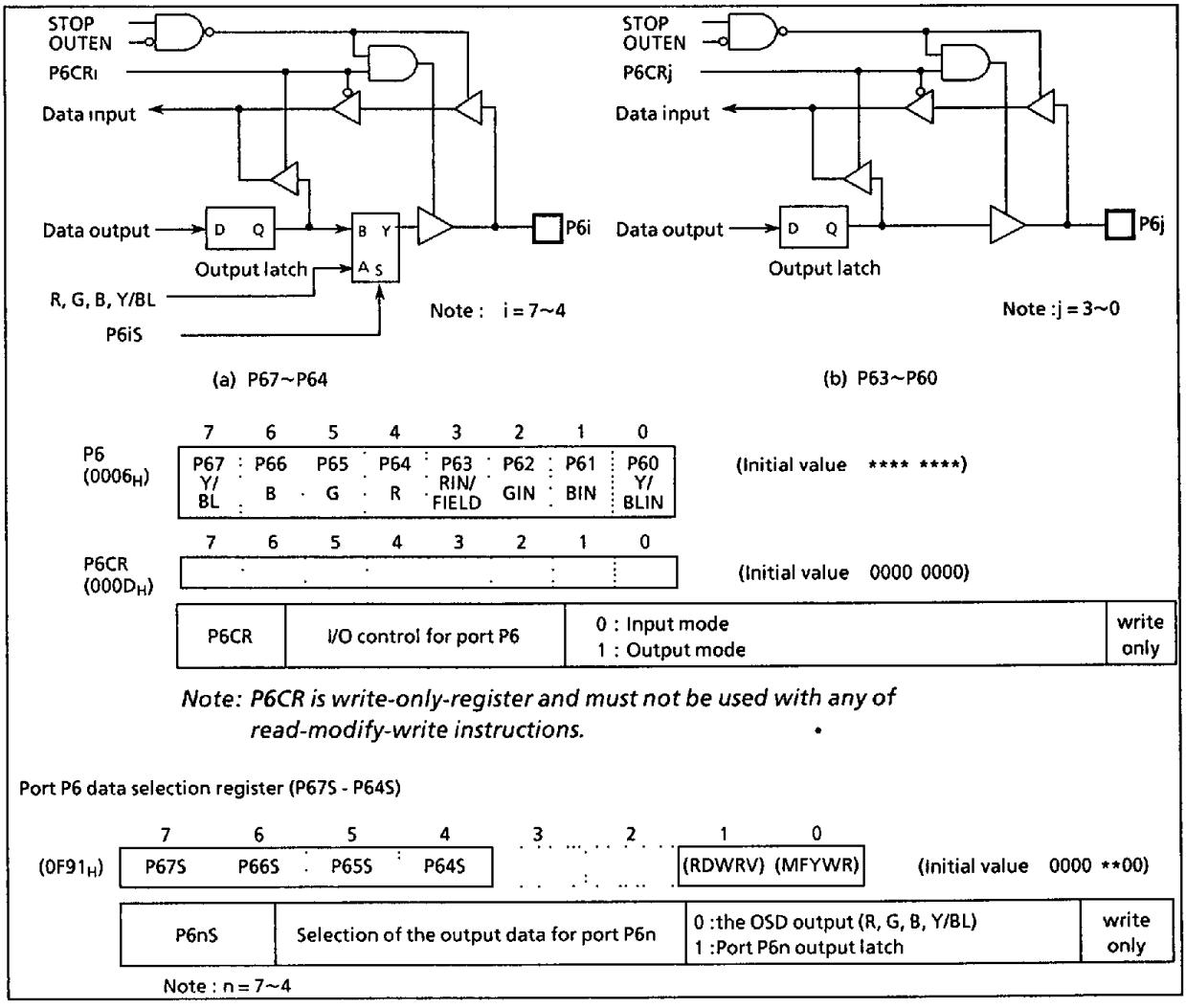


Figure 2-7. Ports P6, P6CR, and P67S - P64S

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

Exsample : Set the Lower 4 bit in port P6 to the output port and set the other to the input port

```
LD      (P6CR), 0FH     ;   P6CR ← 00001111B
```

2.2.6 Port P7 (P71~P70)

Port P7 is a 2-bit input /output port, and is also used as a vertical synchronous signal ($\overline{VD}$) input and a horizontal synchronous signal ($\overline{HD}$) input for the on screen display (OSD) circuitry.

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 2 in P7 are read in as undefined data.

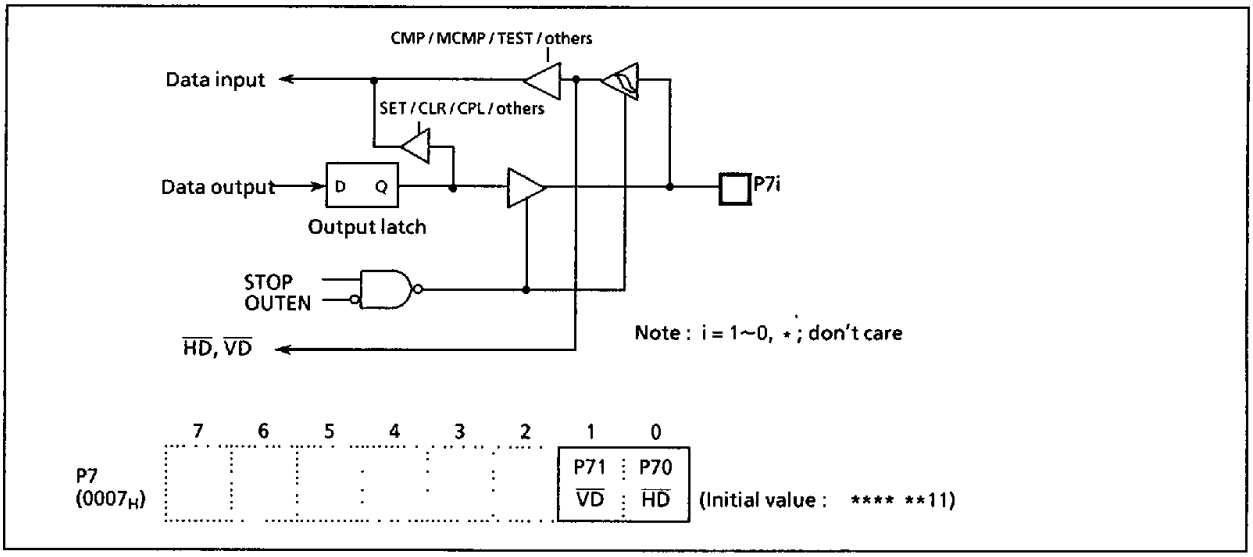


Figure 2-8. Ports P7

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 2-10.

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).

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

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

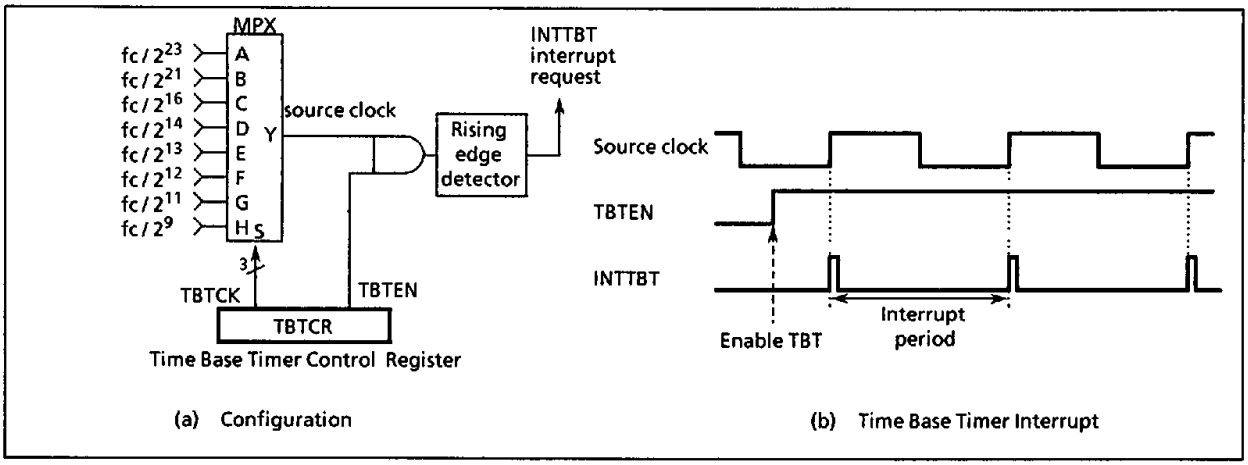


Figure 2-9. Time Base Timer

TBTCR (0036_H)

7	6	5	4	3	2	1	0
"0"	"0"	"0"	"0"	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} [Hz] (0.95 Hz at fc = 8MHz) 001 : fc/2^{21} (3.81 Hz at fc = 8MHz) 010 : fc/2^{16} (122.07 Hz at fc = 8MHz) 011 : fc/2^{14} (488.28 Hz at fc = 8MHz) 100 : fc/2^{13} (976.56 Hz at fc = 8MHz) 101 : fc/2^{12} (1953.12 Hz at fc = 8MHz) 110 : fc/2^{11} (3906.25 Hz at fc = 8MHz) 111 : fc/2^9 (15625 Hz at fc = 8MHz)	

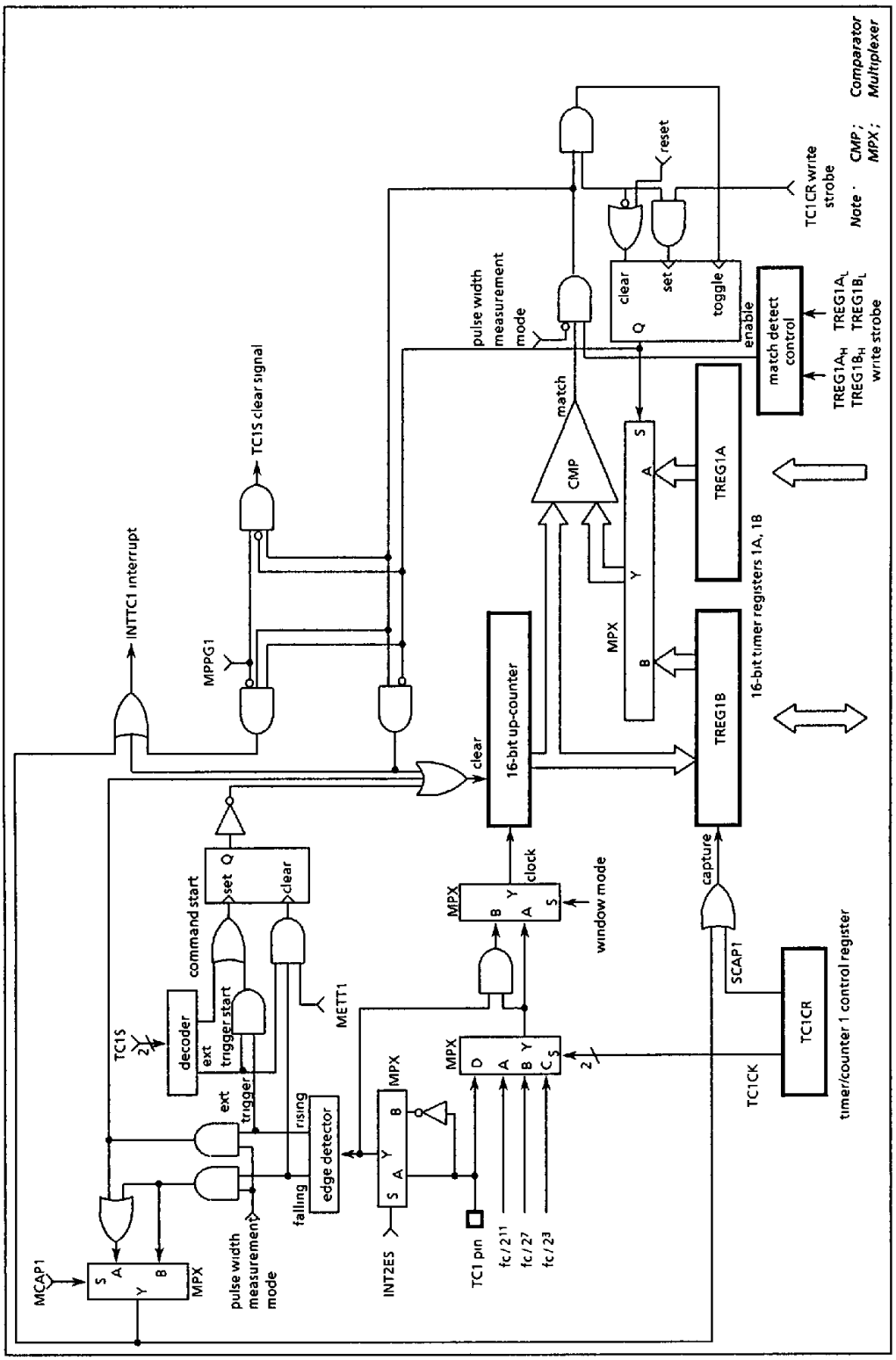
*Note1 : fc ; High-frequency clock [Hz], * ; don't care*
Note2 : The TBTCR is a write-only register and must not be used with any of the read-modify-write instructions.

Figure 2-10. Time Base Timer Control Register

2.4 16-bit Timer 1 (TC1)

The 87C833/C33/H33 has two 16-bit timers (TC1, TC2) and two multi-function 8-bit timer/counters (TC3, TC4).

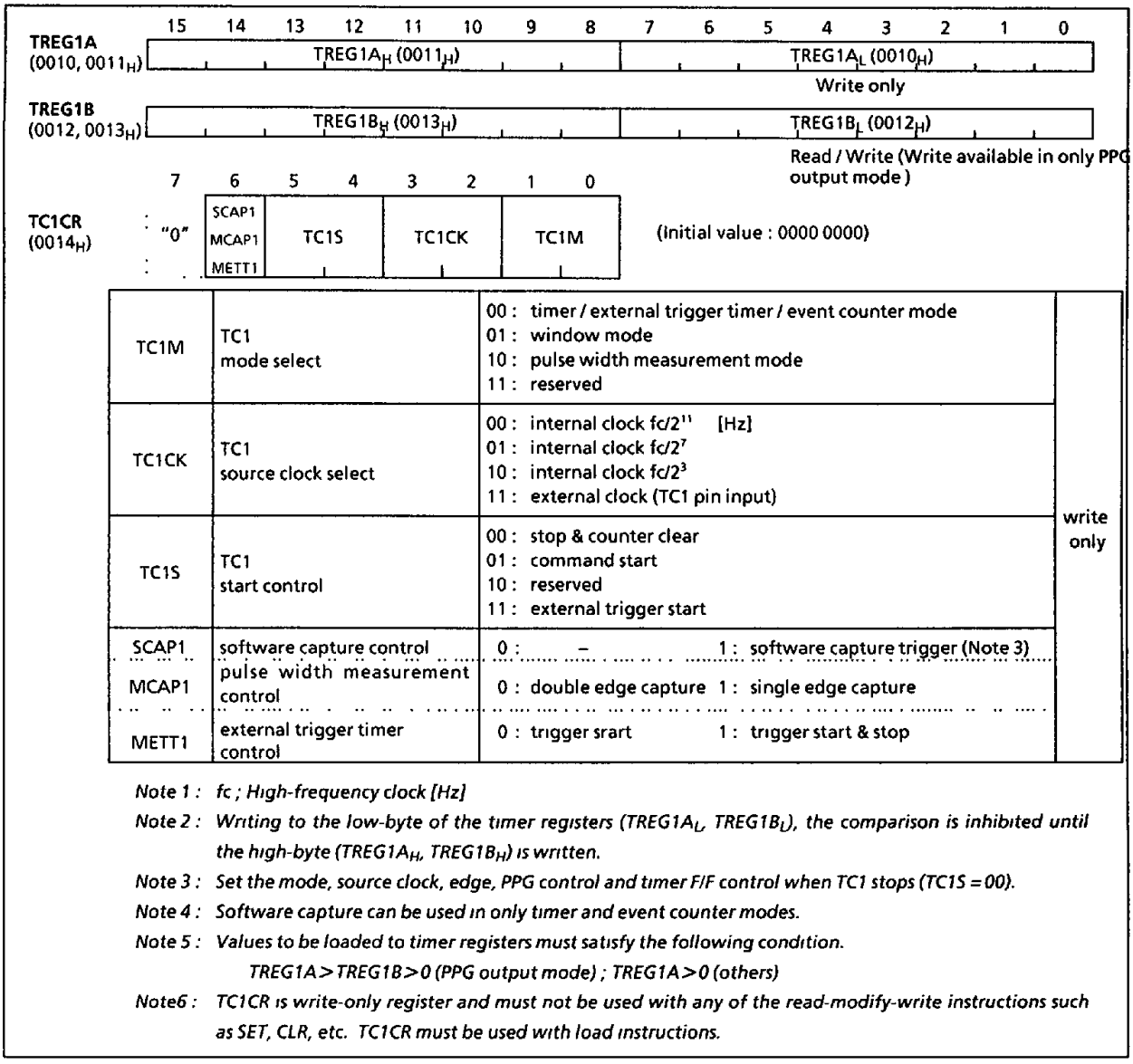
2.4.1 Configuration



Fugure 2-11. Timer/Counter 1 (TC1)

2.4.2 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 the TREG1A and TREG1B.



TC1M	TC1 mode select	00 : timer / external trigger timer / event counter mode 01 : window mode 10 : pulse width measurement mode 11 : reserved	write only
TC1CK	TC1 source clock select	00 : internal clock fc/2^{11} [Hz] 01 : internal clock fc/2^{7} 10 : internal clock fc/2^{3} 11 : external clock (TC1 pin input)	
TC1S	TC1 start control	00 : stop & counter clear 01 : command start 10 : reserved 11 : external trigger start	
SCAP1	software capture control	0 : – 1 : software capture trigger (Note 3)	
MCAP1	pulse width measurement control	0 : double edge capture 1 : single edge capture	
METT1	external trigger timer control	0 : trigger srart 1 : trigger start & stop	

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

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

Note 3 : Set the mode, source clock, edge, 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)

Note6 : TC1CR is write-only register and must not be used with any of the read-modify-write instructions such as SET, CLR, etc. TC1CR must be used with load instructions.

Figure 2-12. Timer Registers and TC1 Control Register

2.4.3 Function

Timer/counter 1 has five operating modes: timer, external trigger timer, event counter, window, pulse width measurement mode.

(1) Timer Mode

In this mode, counting up is performed using the internal clock. The contents of the timer register 1A (TREG1A) are compared with the contents of the 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 the up-counter can be transferred to the timer register 1B (TREG1B) by setting SCAP1 (bit 6 in TC1CR) to "1" (software capture function).

Source clock	Resolution (At fc = 8MHz)	Maximum time setting (At fc = 8MHz)
$fc/2^3$ [Hz]	1 μs	65.535 ms
$fc/2^7$	16 μs	1.04856 s
$fc/2^{11}$	256 μs	16.77696 s

Table 2-1. Timer 1 Source Clock (Internal Clock)

Example 1 : Sets the source clock to $fc/2^7$[Hz] and generates an interrupt 1 [s] later (at fc = 8MHz).

```
LD      (TC1CR), 00000100B      ; Sets the TC1 source clock
LDW     (TREG1A), 0F424H        ; Sets the timer register (1 s – fc / 2^7 = F424H)
SET     (EIRL) EF4              ; Enables INTTC1 interrupt
EI
LD      (TC1CR), 00010100B      ; Starts TC1
```

Example 2 : Software capture

```
LD      (TC1CR), 01010100B      ; SCAP1←1 (Captures)
LD      WA, (TREG1B)            ; Reads captured value
```

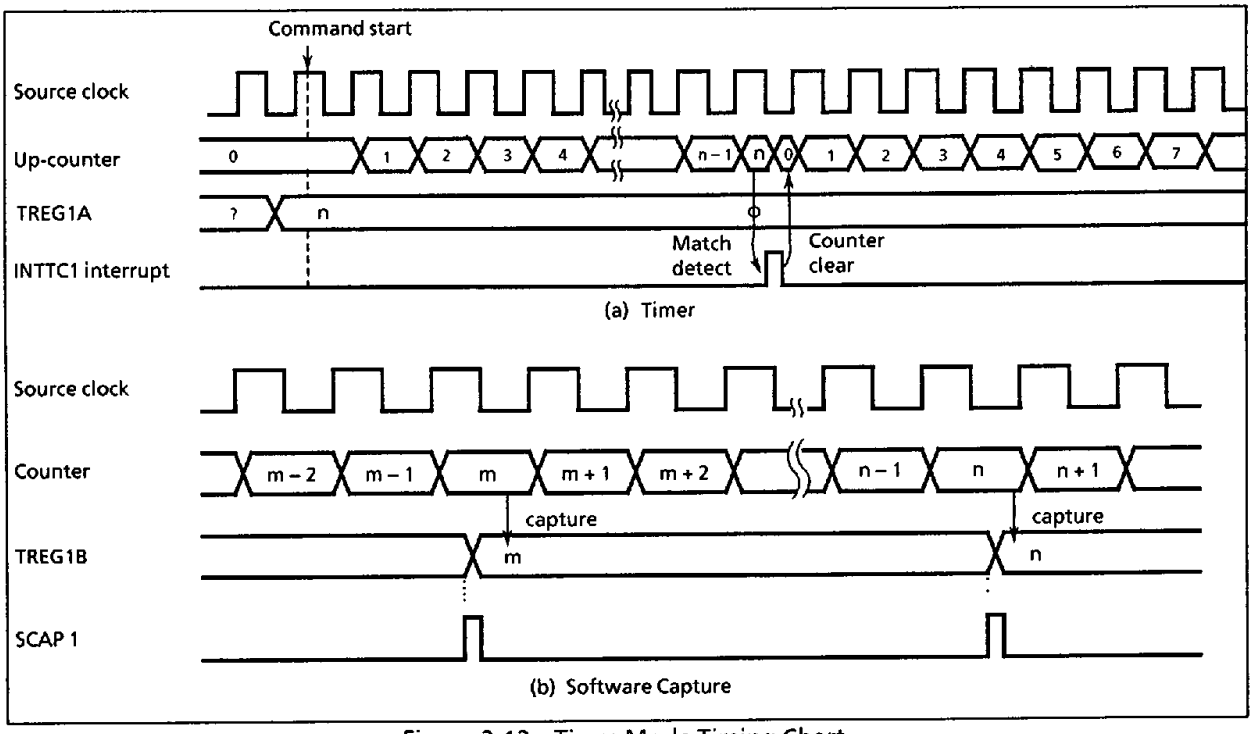


Figure 2-13. Timer Mode Timing Chart

(2) **External Trigger Timer** mode

This is the timer mode to start counting up by the external trigger. The trigger is the edge of the TC1 pin input. Either 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 the TC1CK. The contents of the TREG1A is compared with the contents of the 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 eliminated as noise. A pulse width of 24/fc [s] or more is required for edge detection in the NORMAL or IDEL mode.

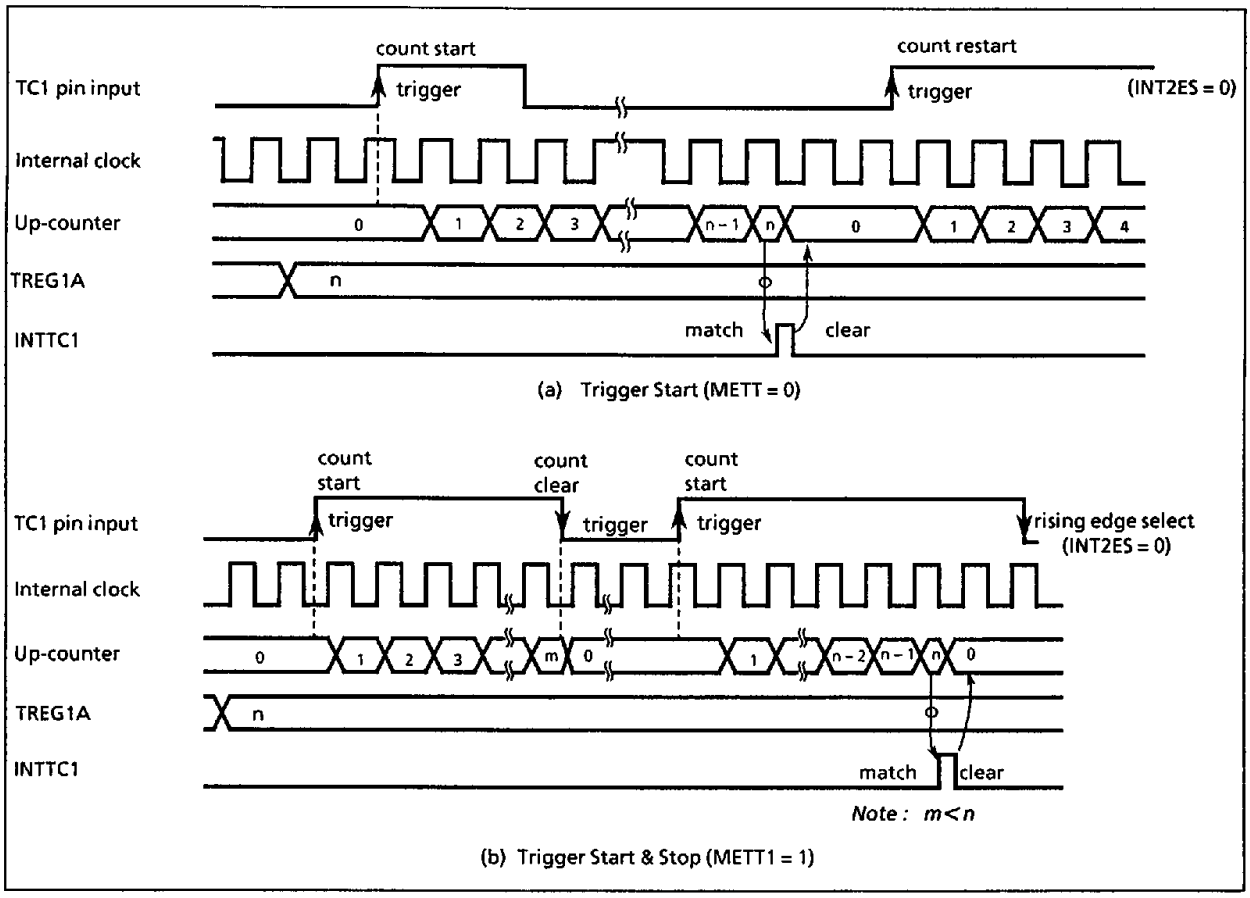


Figure 2-14. External Trigger Timer Mode Timing Chart

(3) **Event Counter** Mode

In this mode, events are counted at the edge of the TC1 pin input. Either rising or falling edge can be selected with INT2ES in EINTCR. The contents of the TREG1A are compared with the contents of the up-counter. If a match is found, an INTTC1 interrupt is generated, and the counter is cleared. The maximum applied frequency is fc/24 [Hz] in the NORMAL or IDLE mode.

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

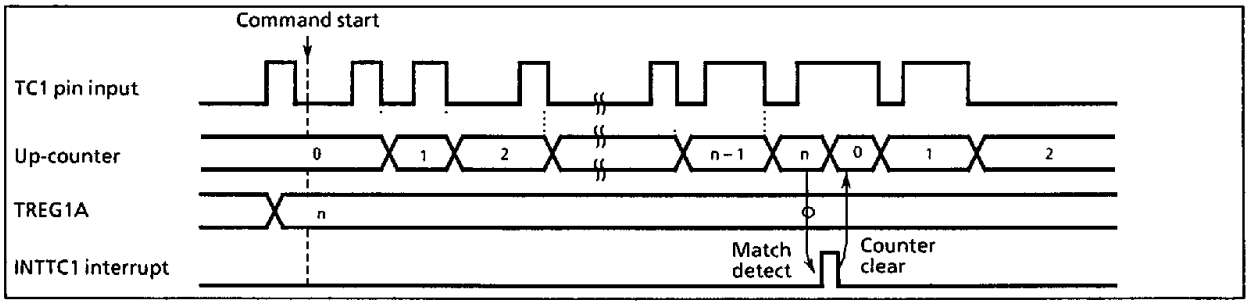


Figure 2-15. Event Counter Mode Timing Chart (INT2ES = 1)

(4) **Window** mode

Counting up is performed at 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 the TREG1A are compared with the contents of the 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 the up-counter to the 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.

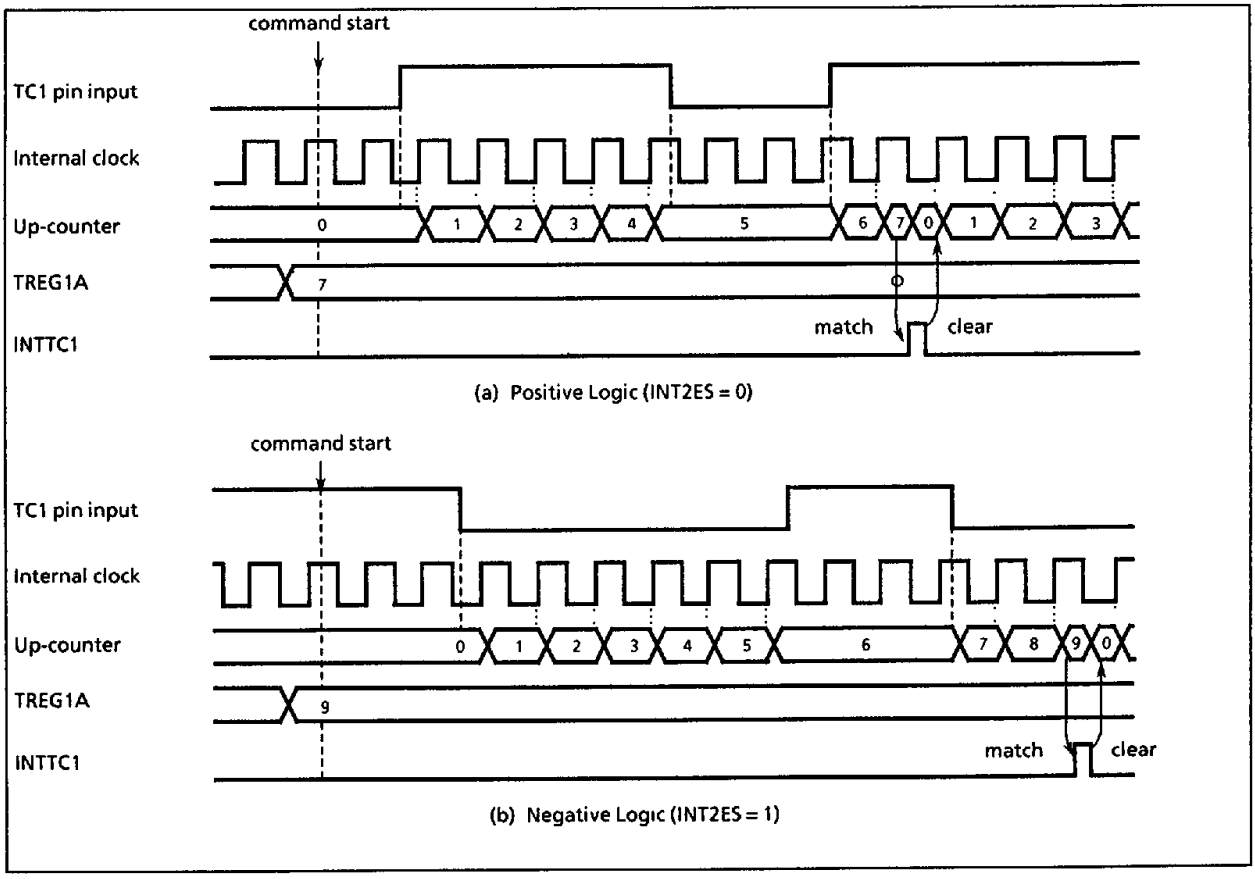


Figure 2-16. Window Mode Timing Chart

(5) **Pulse width measurement** mode

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

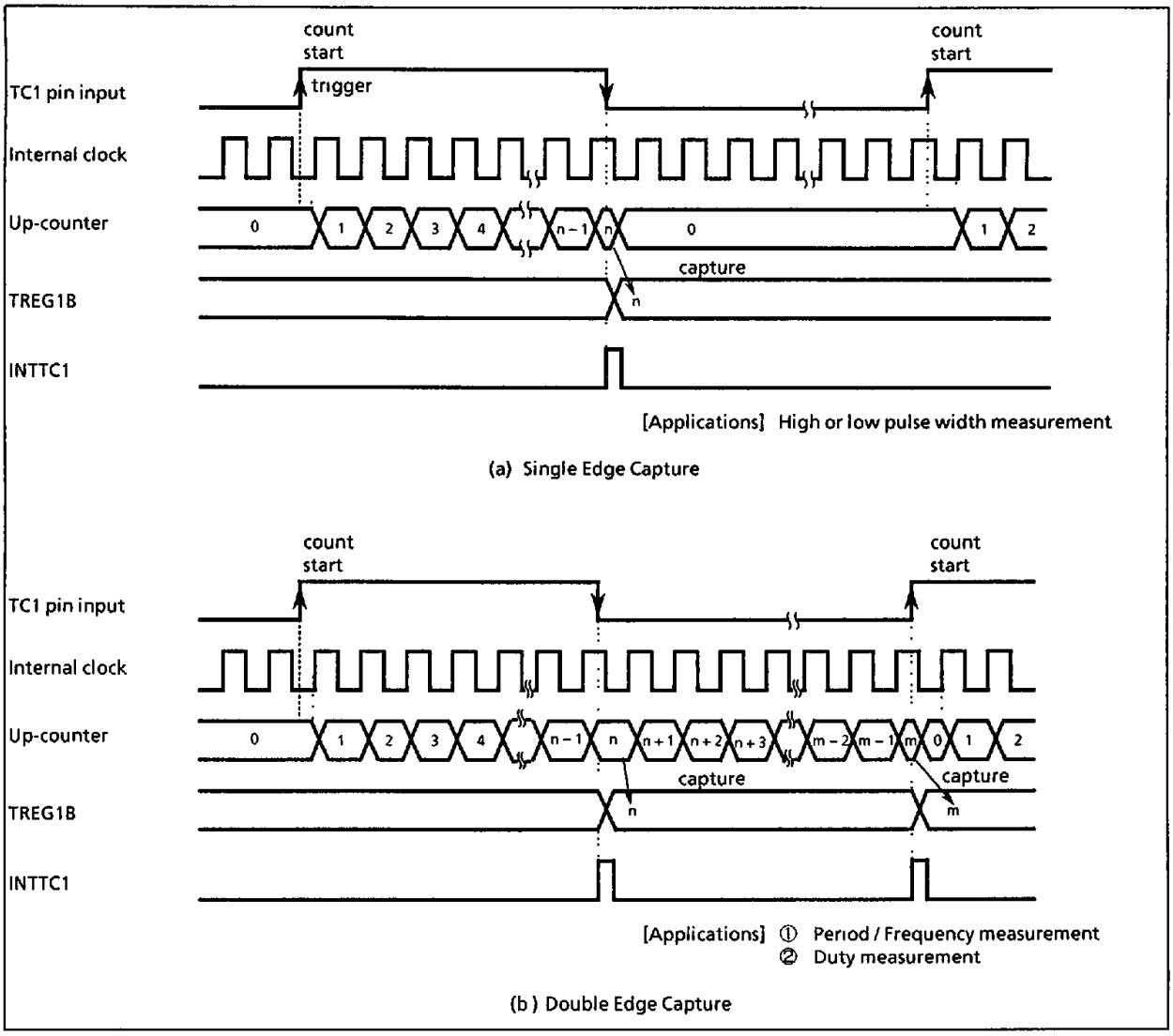


Figure 2-17. Pulse Width Measurement Mode Timing Chart

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

```
        CLR   (INTTC1C). 0          ;  INTTC1 service switch initial setting
        LD    (EINTCR), 00000000B   ;  Sets the rise edge at the INT2 edge
        LD    (TC1CR), 00000110B    ;  Sets the TC1 mode and source clock
        SET   (EIRL). 4             ;  Enables INTTC1
        LD    (TC1CR), 00110110B    ;  Starts TC1 with an external trigger
        :
PINTTC1 : CPL   (INTTC1C). 0        ;  Complements INTTC1 service switch
        JRS   F, SINTTC1
        LD    (HPULSE), (TREG1BL)   ;  Reads TREG1B
        LD    (HPULSE + 1), (TREG1BH)
        RETI
SINTTC1 : LD    (WIDTH), (TREG1BL)  ;  Reads TREG1B (Period)
        LD    (WIDTH + 1), (TREG1BH)
        .
```

2.5 16-bit Timer/Counter 2 (TC2)

2.5.1 Configuration

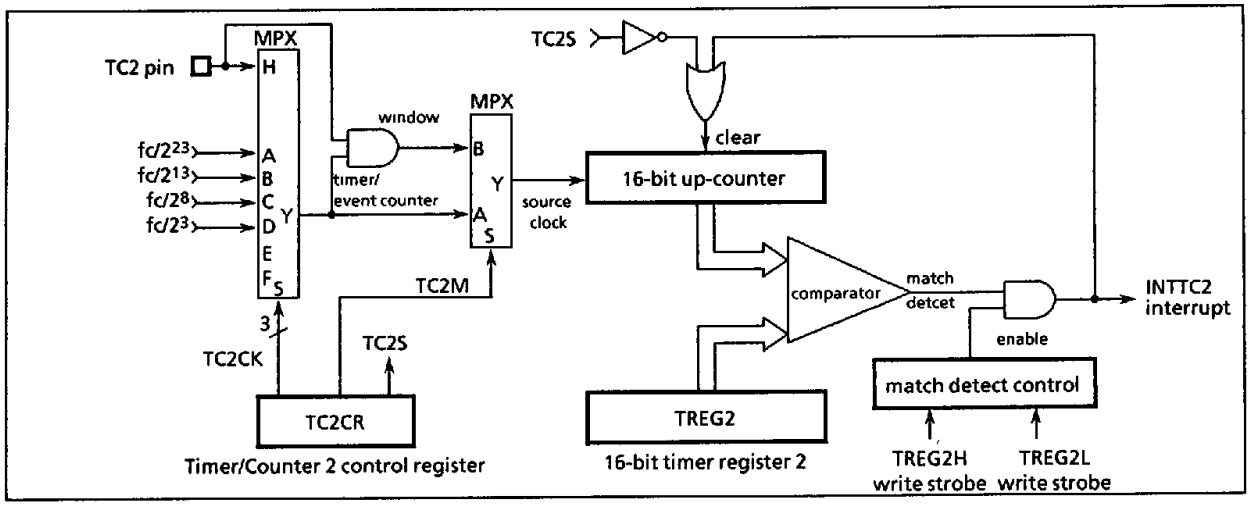


Figure 2-18. Timer/Counter 2 (TC2)

2.5.2 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} [Hz] 001 : 〃 fc / 2^{13} 010 : 〃 fc / 2^{8} 011 : 〃 fc / 2^{3} 100 : Reserved 101 : Reserved 110 : Reserved 111 : External clock (TC2 pin input)	
TC2S	Timer/counter 2 start control	0 : Stop and counter clear 1 : Start	

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

Note 2 : When writing to the low-byte of the 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$_{15\sim11}$ > 0 when warm-up).

Note 5 : The TC2CR and the TREG2 are write-only registers and must not be used with any of the read-modify-write instructions such as SET, CLR, etc.

Figure 2-19. Timer Register 2 and TC2 Control Register

2.5.3 Function

The timer/counter 2 has three operating modes: timer, event counter and window modes.

(1) **Timer Mode**

In this mode, the internal clock is used for counting up. The contents of the timer register 2 (TREG2) are compared with the contents of the 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.

Source clock	Resolution (At fc = 8MHz)	Maximum time setting (At fc = 8MHz)
$fc/2^{23}$ [Hz]	1.048576 s	19 hour 5 min 18.4 s
$fc/2^{13}$	1.024 ms	1 min 7.1 s
$fc/2^{8}$	32 μs	2.09712 s
$fc/2^{3}$	1 μs	65 535 ms

Table 2-2. Source Clock (Internal Clock) for Timer 2

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

```
LD      (TC2CR), 00001100B      ; Sets the source clock
LDW     (TREG2), 61A8H          ; Sets TREG2 (25ms ÷ 2³/fc = 61A8H)
SET     (EIRH) EF14             ; Enables INTTC2 interrupt
EI
LD      (TC2CR), 00101100B      ; Starts TC2
```

(2) **Event Counter** Mode

In this mode, events are counted at 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 the NORMAL and IDLE mode.

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

```
LD      (TC2CR), 00011100B      ; Sets the TC2 mode
LDW     (TREG2), 640            ; Sets TREG2
LD      (TC2CR), 00111100B      ; Starts TC2
```

(3) **Window** Mode

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

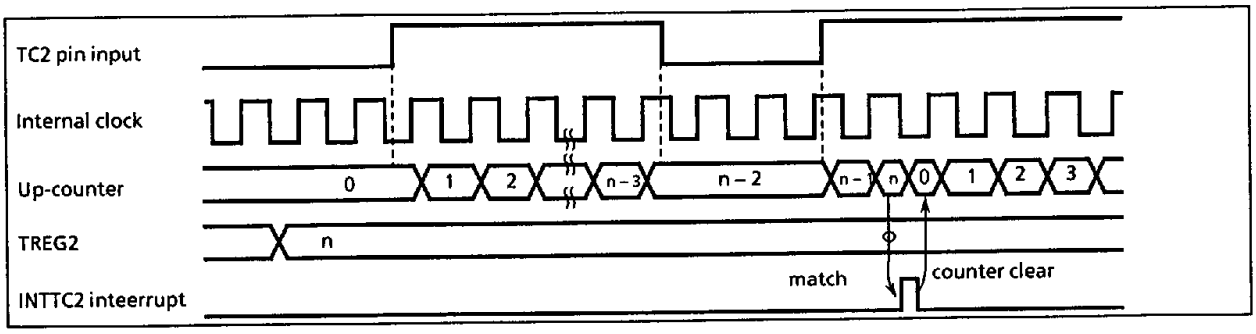


Figure 2-20. Window Mode Timing Chart

2.6 8-Bit Timer/Counter 3 (TC3)

2.6.1 Configuration

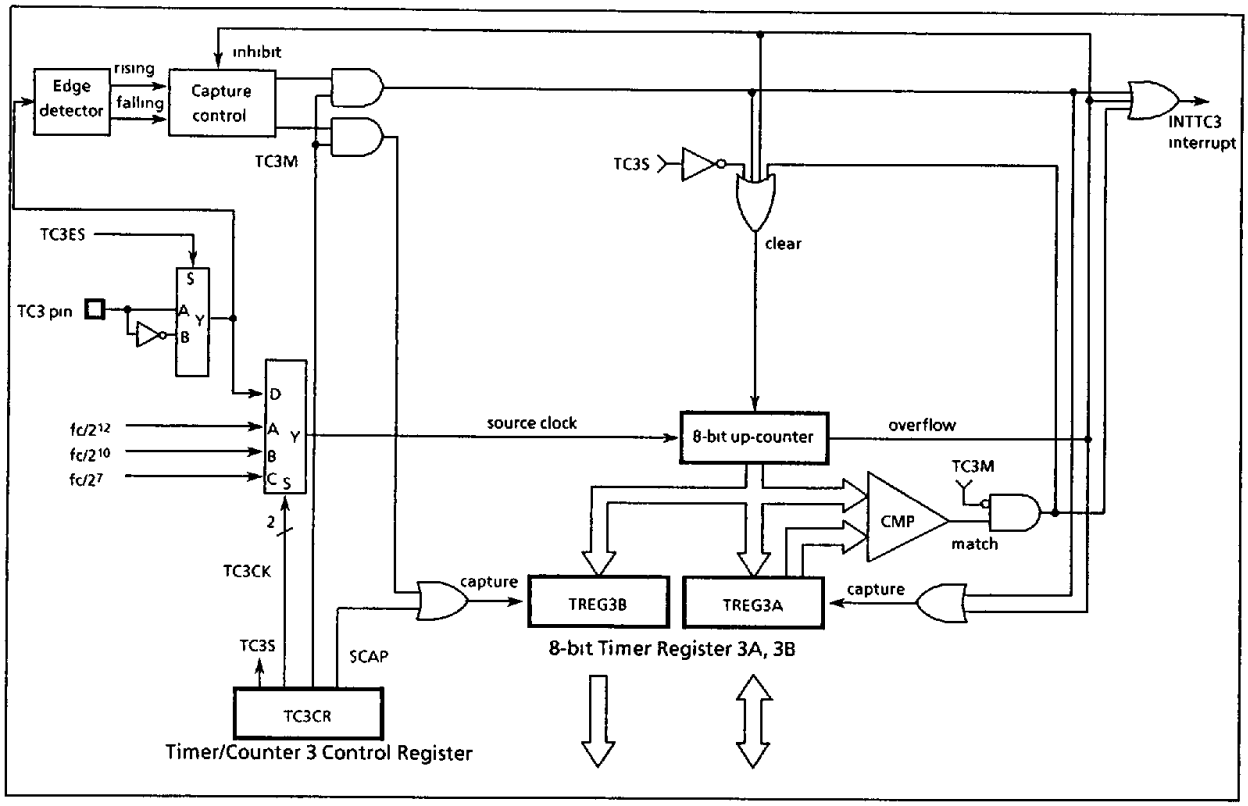
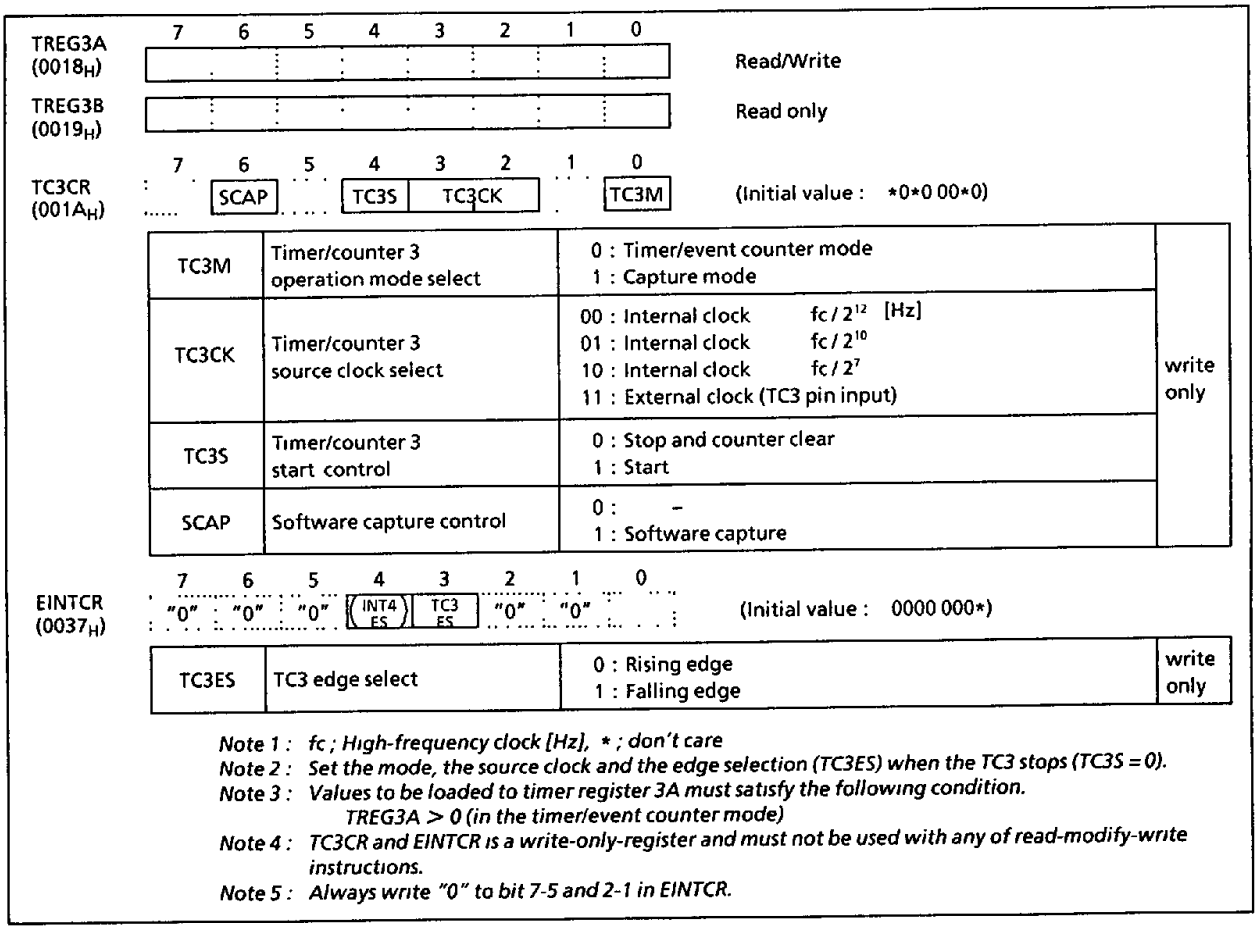


Figure 2-21. Timer/Counter 3

2.6.2 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.



*Note 1 : fc ; High-frequency clock [Hz], * ; don't care*
Note 2 : Set the mode, the source clock and the edge selection (TC3ES) when the TC3 stops (TC3S = 0).
Note 3 : Values to be loaded to timer register 3A must satisfy the following condition.
TREG3A > 0 (in the timer/event counter mode)
Note 4 : TC3CR and EINTCR is a write-only-register and must not be used with any of read-modify-write instructions.
Note 5 : Always write "0" to bit 7-5 and 2-1 in EINTCR.

Figure 2-22. Timer Register 3 and TC3 Control Registers

2.6.3 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 2-3 is used for counting up. The contents of TREG3A are compared with the contents of the up-counter. If a match is found, a timer/counter 3 interrupt (INTTC3) is generated, and the up-counter is cleared. Counting up resumes after the up-counter is cleared. The current contents of up-counter are loaded into TREG3B by setting SCAP (bit 6 in TC3CR) to "1". SCAP is automatically cleared after capturing.

Source clock	Resolution (AT fc = 8MHz)	Maximum setting time (AT fc = 8MHz)
$fc / 2^{12}$	512 μs	130.56 ms
$fc / 2^{10}$	128 μs	32.64 ms
$fc / 2^{7}$	16 μs	4.08 ms

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

(2) Event Counter Mode

In this mode, the TC3 pin input pulse 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]. 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 : Generetes 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 itnerrupt
EI
LD      (TC3CR), 00011100B      ; Starts 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 signal, etc. The counter is running free 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 continued. 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 to the TREG3A and an overflow interrupt (INTTC3) is generated. During interrupt processing, it can determine 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. Therefore, TREG3B has been read out earlier than TREG3A.

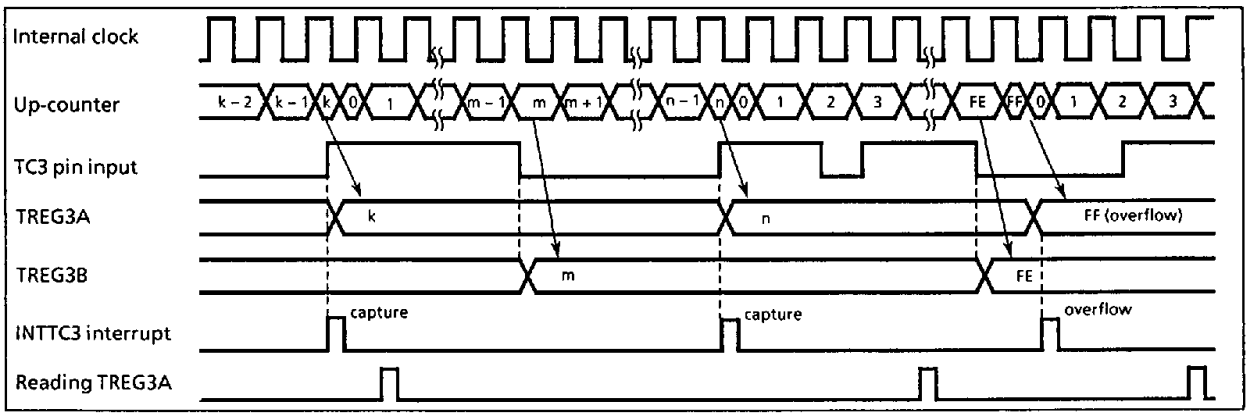


Figure 2-23. Timing Chart for Capture Mode (TC3ES = 0)

2.7 8-bit Timer/Counter (TC4)

2.7.1 Configuration

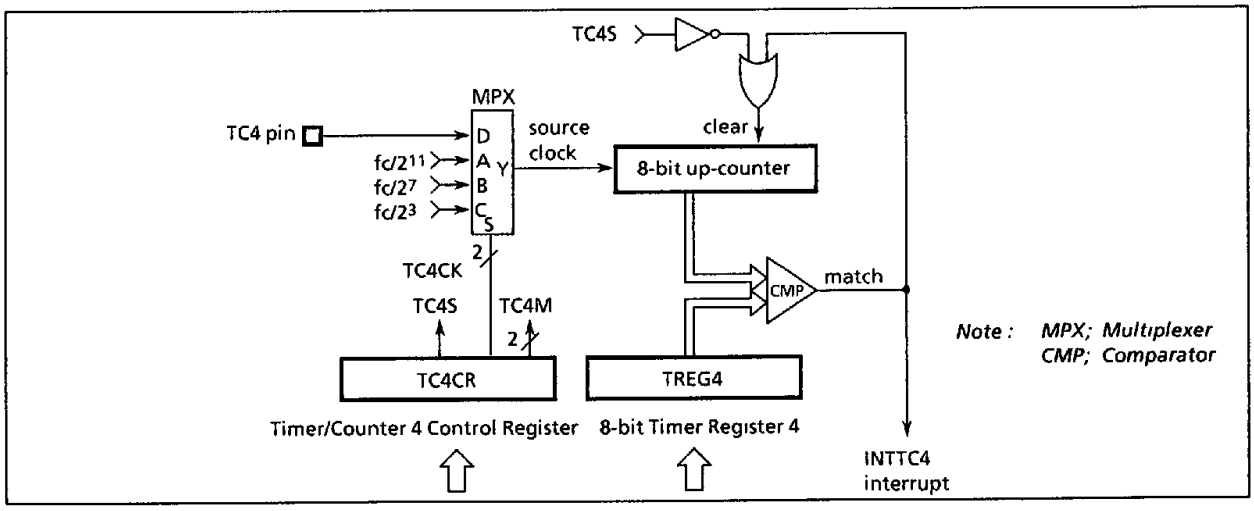
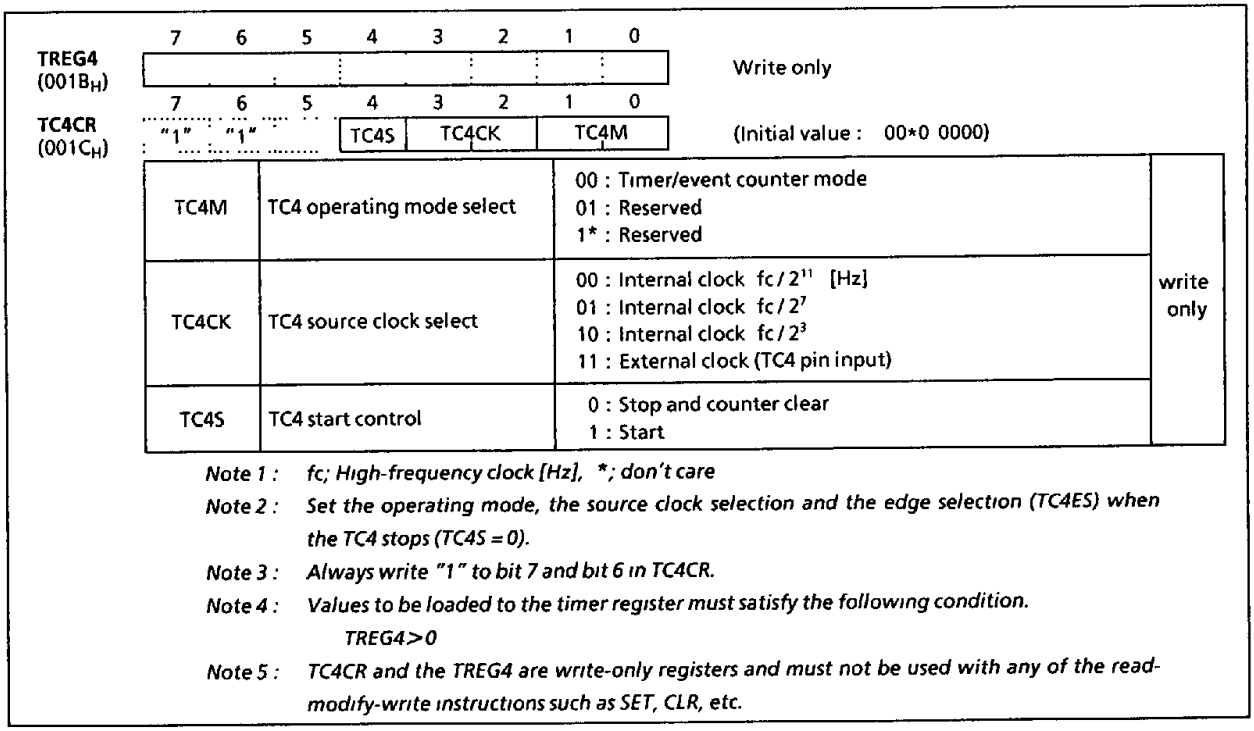


Figure 2-24. Timer/Counter 4

2.7.2 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 the TREG4.



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} [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	

*Note 1 : fc; High-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 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 2-25. Timer Register 4 and TC4 Control Registers

2.7.3 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 the 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.

Source clock	Resolution (At fc = 8MHz)	Maximum setting time (At fc = 8MHz)
$fc/2^{11}$ [Hz]	256 μs	65.28 ms
$fc/2^{7}$	16 μs	4.08 ms
$fc/2^{3}$	1 μs	255 μs

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

(2) **Event Counter** Mode

In this mode, the TC4 pin input (external clock) pulse is used for counting up. 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]. Two or more machine cycles are required for both the high and low levels of the pulse width.

2.8 Serial Bus Interface (SBI)

The 87C833/C33/H33 has a 1-channel serial bus interface which employs a clocked-synchronous 8-bit serial bus interface and an I²C bus (a bus system by Philips).

The serial interface is connected to an external device through P35 (SDA) and P34 (SCL) in the I²C bus mode; and through P34 ($\overline{SCK}$), P36 (SO), and P35 (SI) in the clocked-synchronous 8-bit SIO mode.

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

When the 87C833/C33/H33 is used as master mode, another devices on same bus must be slave mode. Because the 87C833/C33/H33 serial bus interface (SBI) does not have arbitration function. (single master-bus-system)

The data transfer is carried 9bits (8bits data and 1bit acknowledge) unit.

2.8.1 Configuration

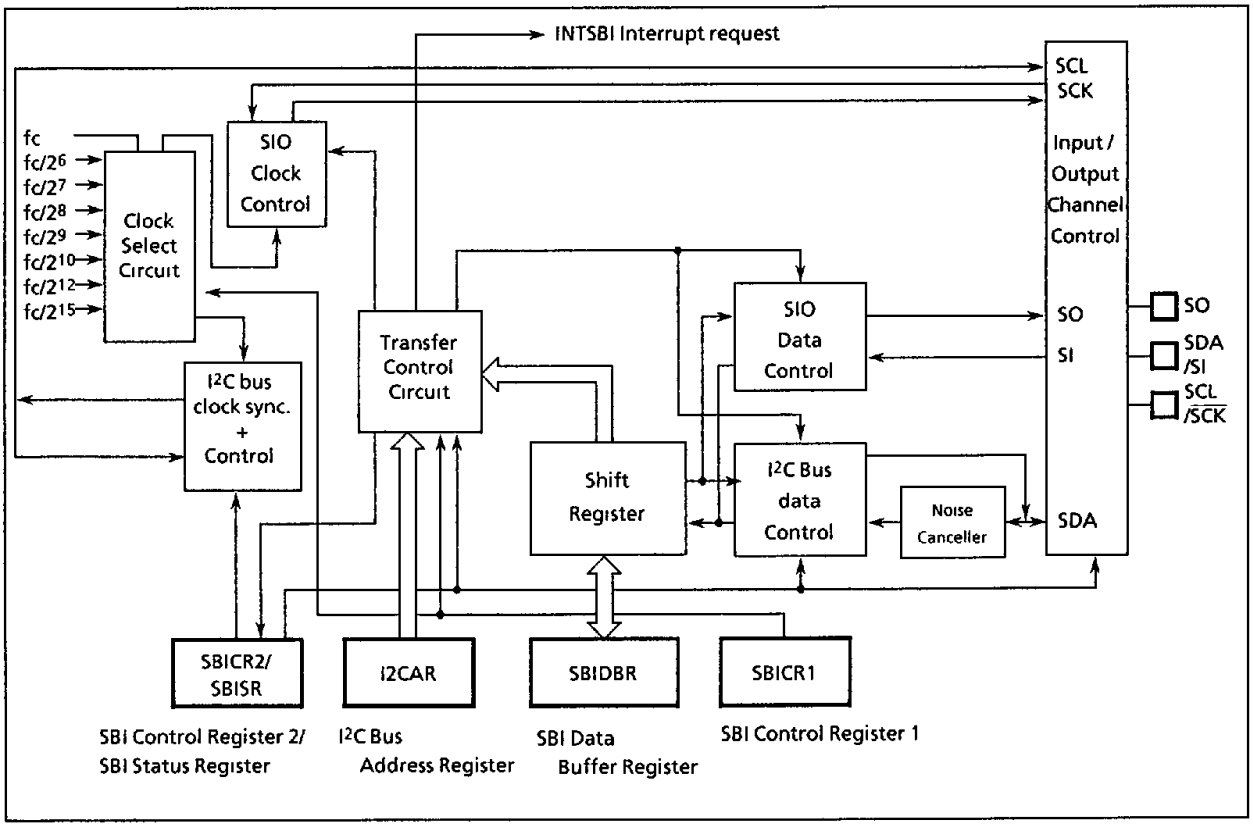


Figure 2-26. Serial Bus Interface (SBI)

2.8.2 Serial Bus Interface (SBI) Control

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

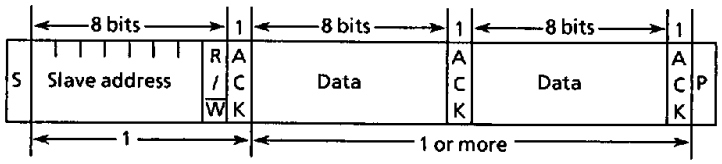
- Serial bus interface control register 1 (SBICR1)
- Serial bus interface control register 2 (SBICR2)
- Serial bus interface data buffer register (SBIDBR)
- I²C bus address register (I2CAR)
- Serial bus interface status register (SBISR)

Refer to Section "2.8.4 I²C bus Mode Control" and "2.8.6 Clocked-synchronous 8-bit SIO Mode Control".

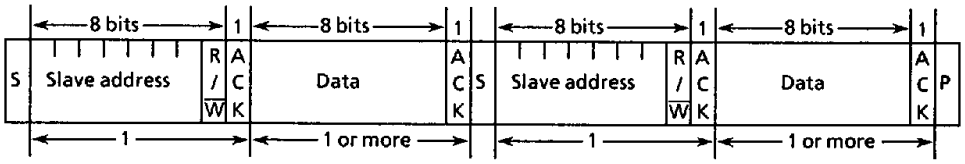
2.8.3 The Data Formats in the I²C bus Mode

The data formats when using the 87C833/C33/H33 in the I²C bus mode are shown below.

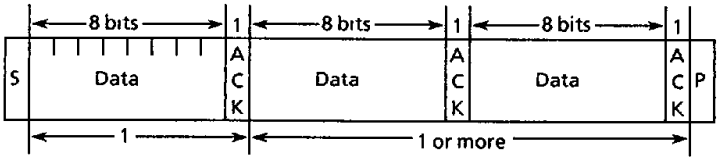
(a) Addressing format



(b) Addressing format (with restart)



(c) Free data format



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

Figure 2-27. The Data Format in the I²C bus Mode

2.8.4 I²C Bus Mode Control

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

Serial Bus Interface Control Register 1

SBICR1 (0020_H)

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

(Initial value ***0 0000)

ACK	Acknowledge mode specification	0 : Acknowledge not returned to transmitter. 1 : Acknowledge returned to transmitter.	Read/ Write
SCK	Frequency of Serial clock (f_{SCL}) selection (only master - mode)	000 : fc/2⁶ [Hz] (125kHz) 001 : fc/2⁷ [Hz] (62 5kHz) 010 : fc/2⁸ [Hz] (31 2kHz) 011 : fc/2⁹ [Hz] (15.6kHz) 100 : fc/2¹⁰ [Hz] (7.8kHz) 101 : fc/2¹² [Hz] (1.9kHz) 110 : fc/2¹⁵ [Hz] (244Hz) 111 : reserved } at fc = 8MHz (Output on SCL pin)	Write only

*Note 1 : * ; don't care*

Note 2 : The SBICR1 (bit7, 6, 5, 3) is writed to normaly "0"

Serial Bus Interface Data Buffer Register

SBIDBR (0021_H)

7	6	5	4	3	2	1	0

Read / Write

I²C bus Address Register

I2CAR (0022_H)

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

(Initial value 0000 0000)

ALS	Address recognition mode specification	0 : Slave address recognition 1 : Non slave address recognition	Write only

Figure 2-28. 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 (Write), Status monitor (Read)	0 : Slave 1 : Master	Read/ Write
TRX	Transmitter/receiver selection (Write), Status monitor (Read)	0 : Receiver 1 : Transmitter	
BB	Start/stop generation (Write), I²C bus status monitor (Read)	0 : Stop condition (Write) , Bus free (Read) 1 : Start condition (Write) , Bus busy (Read)	
PIN	Cancel interrupt service request(Write), Status monitor (Read)	0 : – (Write), Interrupt service requested (Read) 1 : Cancel interrupt service request (Write) , canceled (Read)	
SBIM	Serial bus interface operating mode selection	00 : Port mode (serial bus interface output disable) 01 : SIO mode 10 : I²C bus mode 11 : Reserved	Write only

Note 1 : * : don't care

Note 2 : Switch a mode to port after making sure that a bus is free.

SBISR (0023_H)

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

AAS	Slave address match detection monitor	0 : – 1 : Slave address match or "GENERAL CALL" detected	Read only
AD0	"GENERAL CALL" detection monitor	0 : – 1 : "GENERAL CALL" detected	
LRB	Last received bit monitor (acknowledge signal monitor)	0 : Last received bit "0" (with acknowledge signal) 1 : Last received bit "1" (with acknowledge signal)	

Figure 2-29. Serial Bus interface Control Register 2/Serial Bus interface status register in the I²Cbus Mode

(1) **Acknowledge mode specification**

Set the ACK (bit 4 in the SBICR1) to "1" for operation in the acknowledge mode. In the receiver mode during the clock pulse cycle, the SDA pin is pulled down to the low level in order to generate the acknowledge signal. When the ACK is cleared to "0", the SDA pin released high-level in the acknowledge timing.

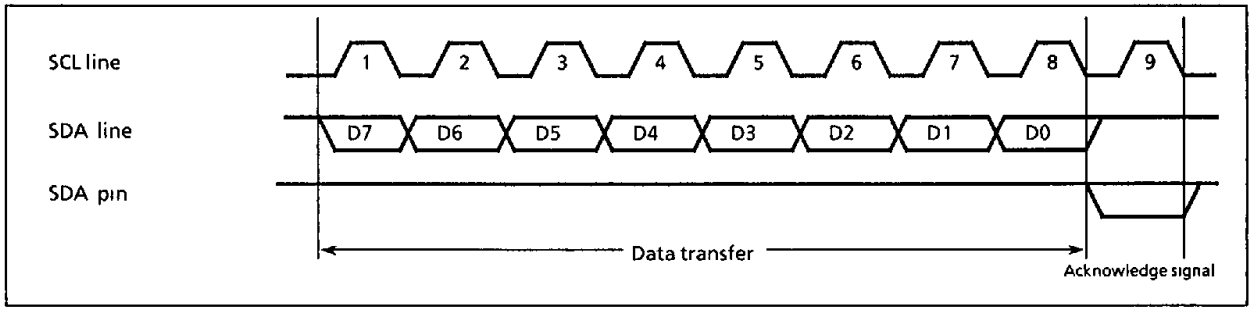
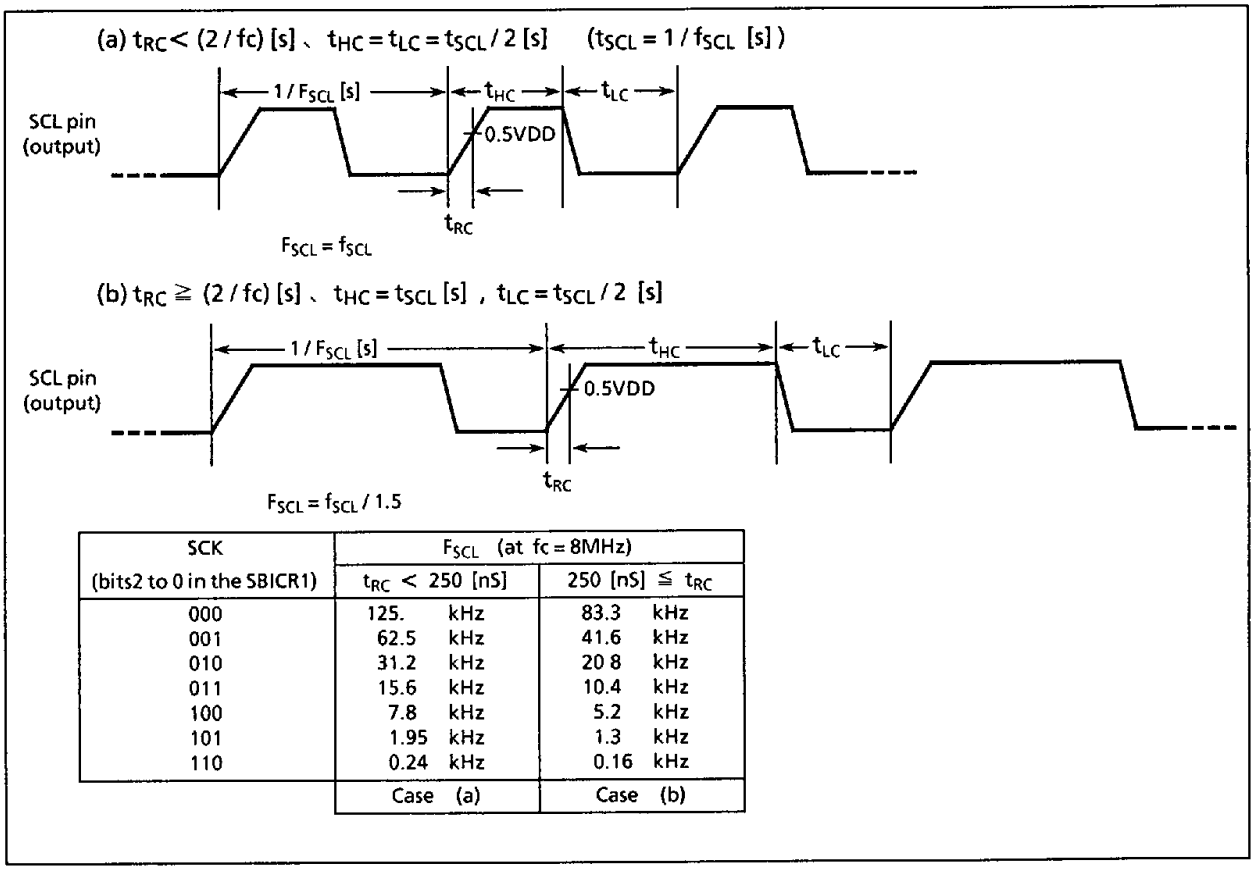


Figure 2-30. Acknowledge signal output

(2) **Serial clock**

The SCK (bits 2 to 0 in the SBICR1) is used to select a maximum transfer frequency directed from the SCL pin in the master mode. If the rising time of output clock is more 2/fc [s], "High" period of clock is extended. If the SCK is set to the "Low" Level with the slave device, the clock is stopped in this period. After restart, the t_{HC} [s] of first clock is set to the [($t_{SCL}/2$) ≦ t_{HC} ≦ t_{SCL}].



SCK (bits2 to 0 in the SBICR1)	F_{SCL} (at fc = 8MHz)	
	t_{RC} < 250 [nS]	250 [nS] ≦ t_{RC}
000	125. kHz	83.3 kHz
001	62.5 kHz	41.6 kHz
010	31.2 kHz	20 8 kHz
011	15.6 kHz	10.4 kHz
100	7.8 kHz	5.2 kHz
101	1.95 kHz	1.3 kHz
110	0.24 kHz	0.16 kHz
	Case (a)	Case (b)

Figure 2-31. Serial Clock

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

When the 87C833/C33/H33 is used as a slave device, set the slave address and ALS to the I2CAR. Set "0" to the ALS for the address recognition mode.

(4) **Master/slave selection**

Set the MST (bit 7 in the SBICR2) to "1" for operating the 87C833/C33/H33 as a master device. Reset the MST to "0" for operation as a slave device. The MST is cleared to "0" by the hardware after a stop condition on the bus is detected.

(5) **Transmitter/receiver selection**

Set the TRX (bit 6 in the SBICR2) to "1" for operating the 87C833/C33/H33 as a transmitter. Reset the TRX to "0" for operation as a receiver. When data with an addressing format is transferred in the slave mode, when a slave address with the same value that sets an I2CAR or when a GENERAL CALL is received (all 8-bit data are "0" after a start condition, the TRX is set to "1" if the direction bit (R/$\overline{W}$) sent from the master device is "1", and is cleared to "0" if the bit is "0". In the master mode, after an acknowledge signal is returned from the slave device with the hardware, the TRX is set to "0" if a transmitted direction bit is "1", and set to "1" if it is "0". When an acknowledge signal is not returned, the current condition is maintained.

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

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

A start condition and before setting slave address and the direction bit to a data buffer register are output on a bus by writing "1" to the MST, TRX, and BB when the BB (bit 5 in the SBICR2) is "0".

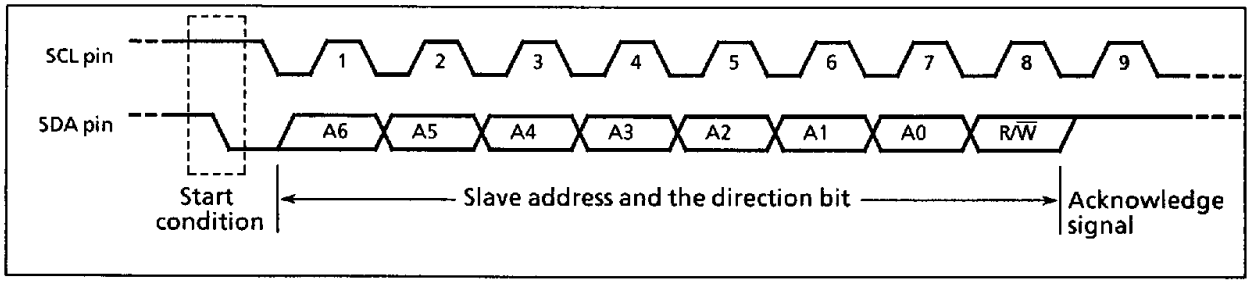


Figure 2-32. Start Condition Generation and Slave Address Generation

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

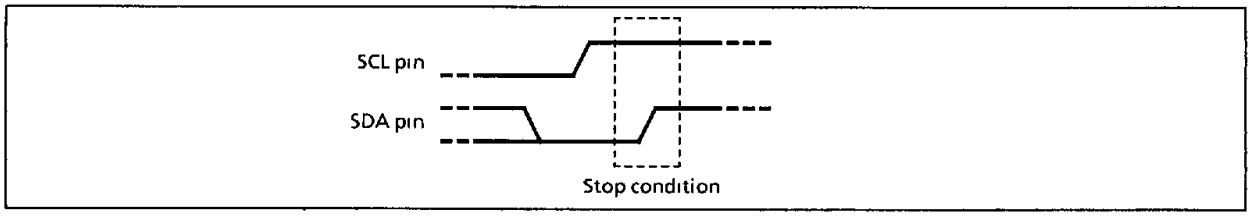


Figure 2-33. Stop Condition Generation

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. In the case of the 87C833/C33/H33 is a master transmitter, after a start condition is generated, until a stop condition is generated and until the MST is set to "0", the count of writing BB is read from BB.

(7) **Cancel interrupt service request**

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

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

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

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

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

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

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

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

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

(10) **GENERAL CALL detection monitor**

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

(11) **Last received bit monitor**

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

2.8.5 Data Transfer in I^2C bus Mode

(1) **Device Initialization**

Set the ACK, CHS and SCK in the SBICR1. Specify "0" to bits 7 to 5.

Set a slave address and the ALS (ALS = 0 when an addressing format) to the I2CAR.

For specifying the default setting to a slave receiver mode, assign "0" to the MST, TRX, and BB in the SBICR2; "1" to the PIN; "10" to the SBIM; and "0" to bits 0 and 1.

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

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

Set the ACK to "1" and specify a slave address and a direction bit to be transmitted to the SBIDBR.

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

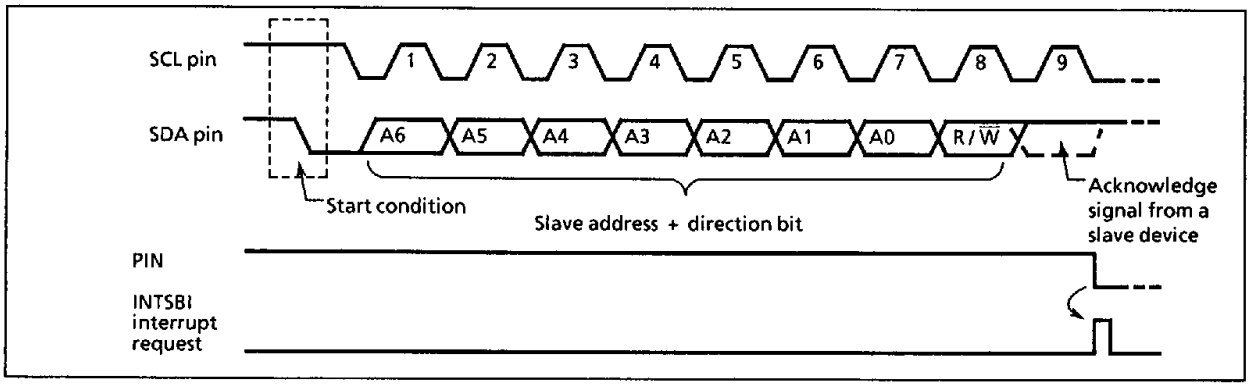


Figure 2-34. Start Condition Generation and Slave Address Transfer

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

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

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

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

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

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

When the LRB is "0", the receiver requests new data. 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 from the SDA pin. After the data is transmitted, an INTSBI interrupt request occurs. The PIN becomes "0" and the SCL pin is pulled down to the low level. If the data to be transferred is more than one word in length, repeat the procedure from the LRB test above.

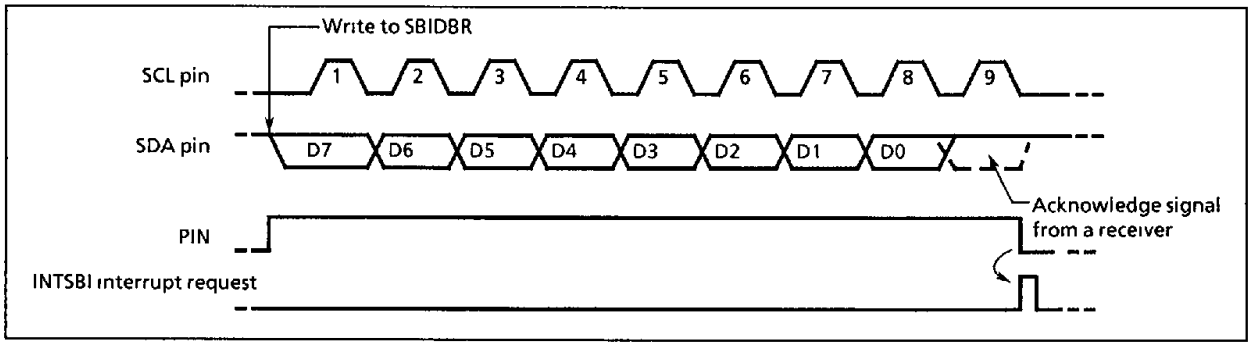


Figure2-35. Example of when BC = "000", ACK = "1"

② When the TRX is "0" (Receiver mode)
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 87C833/C33/H33 outputs a serial clock pulse to the SCL pin to transfer new 1-word of data and sets the SDA pin to "0" at the acknowledge signal timing.
An INTSBI interrupt request then occurs and the PIN becomes "0", the SCL pin is pulled down to the Low Level. The 87C833/C33/H33 outputs a clock pulse for 1-word of data transfer and the acknowledge signal each time that received data is read from the SBIDBR.

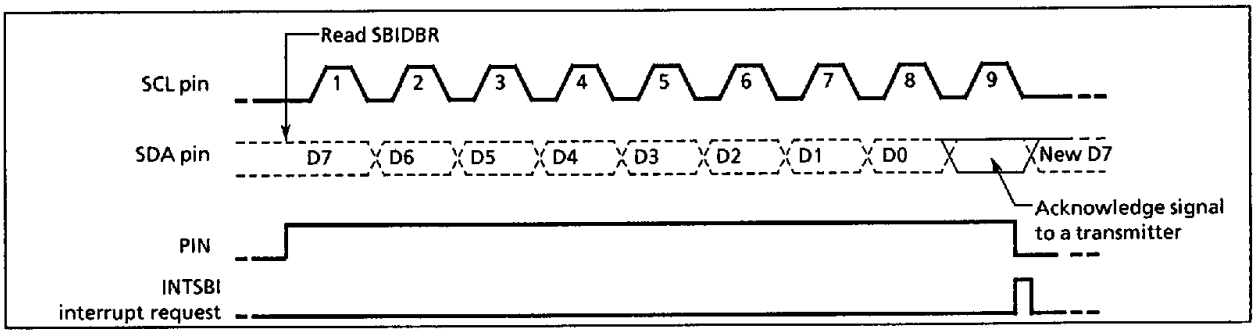


Figure 2-36. Example of when ACK = "1"

In order to terminate transmitting data to a transmitter, reset to "0" the ACK before reading data which is 1 word before the last data to be received. Then the 87C833/C33/H33 releases SDA pin with "1" at the acknowledge signal timing of the last data to be received, and informs transmitting end to a transmitter.
After data is received and an interrupt request has occurred, the 87C833/C33/H33 generates a stop condition and terminates data transfer. Even if SBIDBR is read at timing of the last data to be received, serial clock and acknowledge signal are not output. There is a reason value of ACK is "0".

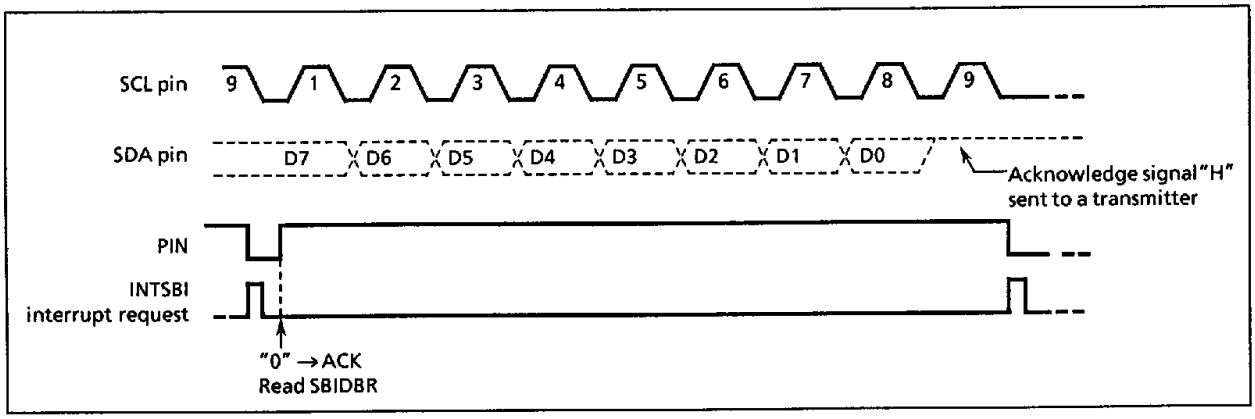


Figure 2-37. Termination of data transfer in master receiver mode

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

In the slave mode, an INTSBI interrupt request occurs when the 87C833/C33/H33 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. When an INTSBI interrupt request occurs, the PIN (bit 4 in the SBICR2) is reset to "0", and the SCL pin is pulled down to the low level. Either reading/writing from/to the SBIDBR or setting the PIN to "1" releases the SCL pin.

The 87C833/C33/H33 tests 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.

TRX	AAS	AD0	Conditions	Process
1	1	0	In the slave receiver mode, the 87C833/C33/H33 receives a slave address of which the value of the direction bit sent from the master is "1".	Write transmitted data to the SBIDBR.
	0	0	In the slave transmitter mode, 1-byte data is transmitted.	Test the LRB. If the LRB is set to "1", set the PIN to "1" since the receiverdoes not request further data. Then, reset the TRX to release the bus. If the LRB is set to "0" write transmitted data to the SBIDBR since the receiver requests further data.
0	1	1/0	In the slave receiver mode, the 87C833/C33/H33 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 write "1" to the PIN.
	0	1/0	In the slave receiver mode, the 87C833/C33/H33 terminates receiving of 1-byte data.	Read received data from the SBIDBR.

Table 2-5. Operation in the Slave Mode

(4) **Stop Condition Generation**

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

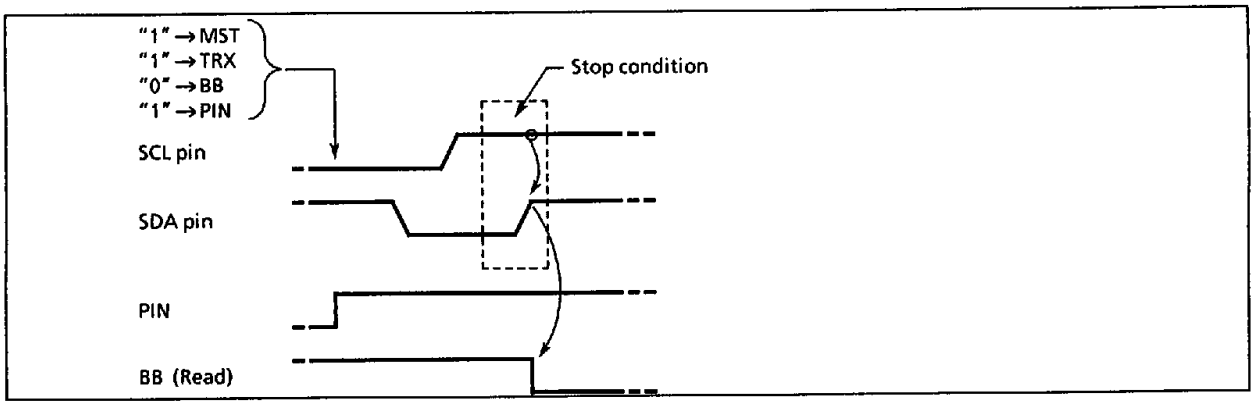


Figure 2-38. Stop Condition Generation

(5) **Restart**

Restart is used to change the direction of data transfer between a master device and a slave device during transferring data. Restart is re-generated a start condition through start condition to stop condition.

Specify "0" to the MST, TRX, and BB and "1" to the PIN and release the bus. The SDA pin retains the high level and the SCL pin is released. Since a stop condition is not generated on the bus, the bus is assumed to be in a busy state from other devices. Test the BB until it becomes "0" to check that the SCL pin of the 87C833/C33/H33 is released. Test the LRB until it becomes "1" to check that the SCL line of the bus is not pulled down to the low level by other devices. After confirming that the bus stays in a free state, generate a start condition with procedure (2).

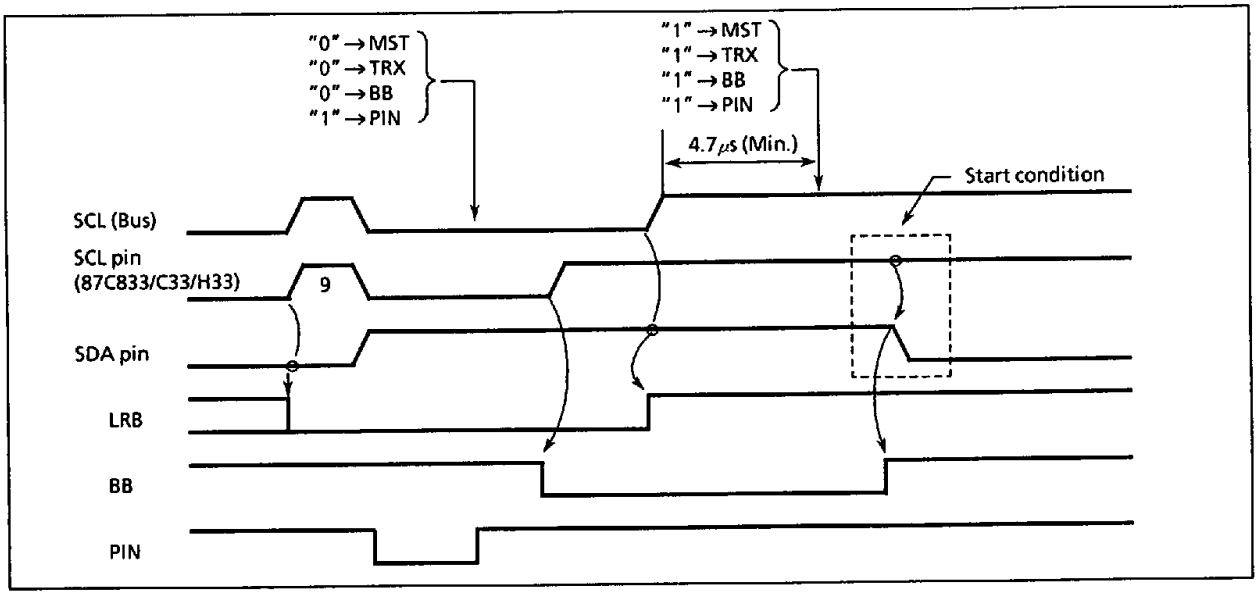


Figure 2-39. Timing diagram when restarting the 87C833/C33/H33

2.8.6 Clocked-synchronous 8-bit SIO Mode Control

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

Serial Bus Interface Control Register 1

SBICR1 (0020_H)

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

(Initial value : 0000 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	
SCK	Serial clock select	000 : fc/2^6 (125kHz) 001 : fc/2^7 (62.5kHz) 010 : fc/2^8 (31.25kHz) 011 : fc/2^9 (15.62kHz) 100 : fc/2^{10} (7.81kHz) 101 : fc/2^{12} (1.95kHz) 110 : fc/2^{15} (244Hz) 111 : External clock (input from $\overline{SCK}$ pin) } at fc = 8MHz ($\overline{SCK}$ pin : Output on)	

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

Serial Bus Interface Data Buffer Register

SBIDBR (0021_H)

7	6	5	4	3	2	1	0

Read / Write

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 : I2C bus mode 11 : reserved	Write only

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

Serial Bus Interface Status Register

SBISR (0023_H)

7	6	5	4	3	2	1	0
				SIOF	SEF		

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

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

(1) Serial Clock

a. Clock source

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

① Internal Clock

In an internal clock mode, any of seven frequencies can be selected. The serial clock is output to the outside on the $\overline{SCK}$ 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.

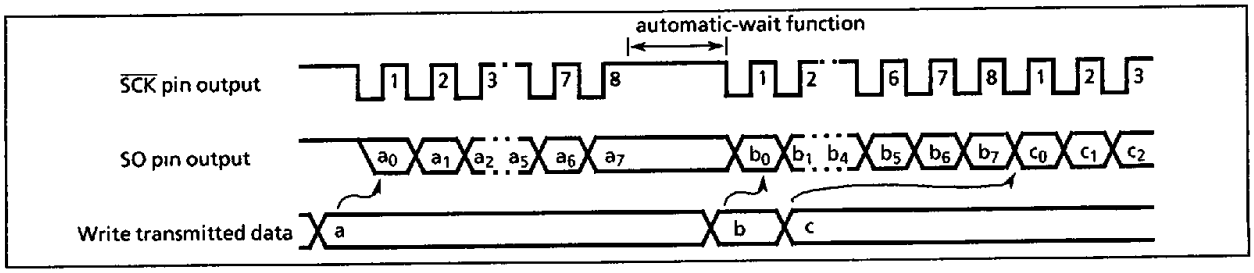


Figure 2-41. Automatic-wait Function

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 longer than 4 machine cyles is required for both high and low levels in the serial clock. The maximum data transfer frequency is 250kHz (when fc = 8MHz).

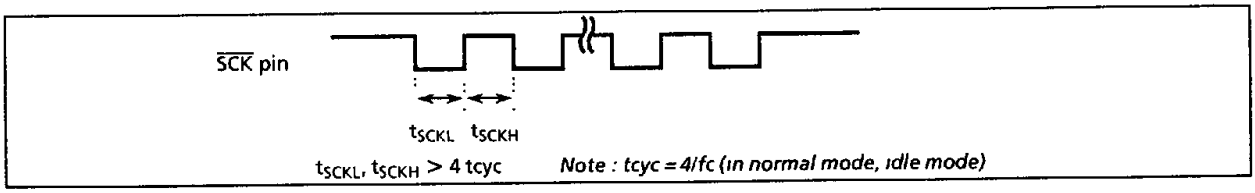


Figure 2-42. External Clock

b. Shift edge

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

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

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

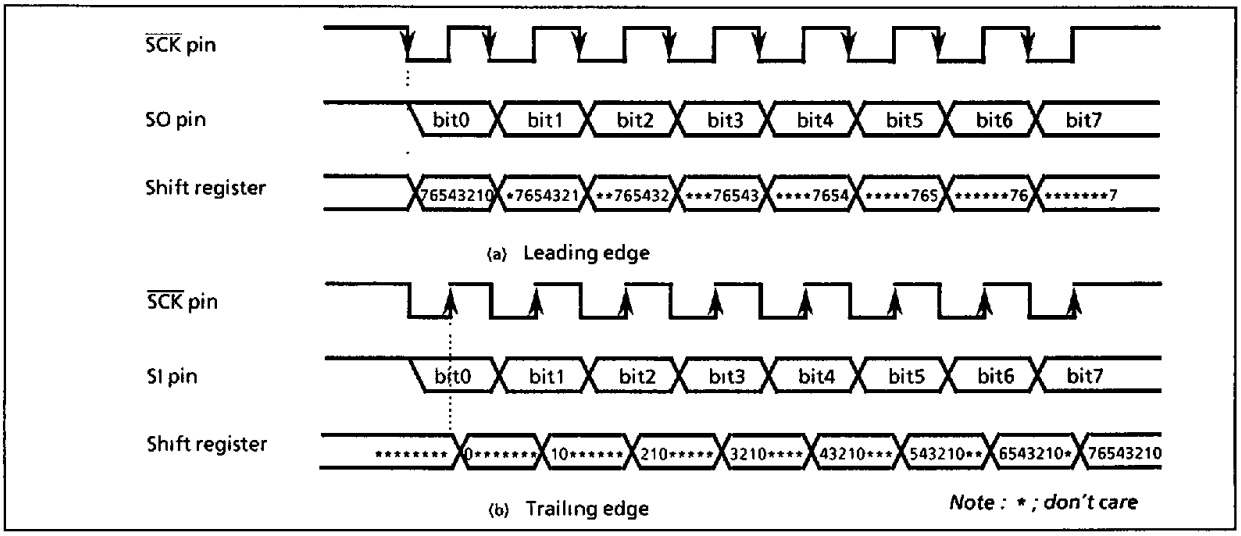


Figure 2-43. Shift Edge

(2) Transfer mode

The SIOM (bits 5 and 4 in the SIO1CR) 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 data to the SBIDBR.

After the 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 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 data is written, automatic-wait function is canceled.

When the external clock is used, data should be written to the SBIDBR before new data is shifted. The transfer speed is determined by the maximum delay time between the time when an interrupt request is generated and the time when data is written to the SBIDBR by the interrupt service program.

Transmitting data is ended by clearing the SIOS to "0" by the buffer empty interrupt service program or setting the SIOINH to "1". When the SIOS is cleared, the transmitted mode ends when all data is output. In order to confirm if data is surely transmitted by the program, set the SIOF (bit 3 in the SBISR) to be sensed. The SIOF is cleared to "0" when transmitting is complete. When the SIOINH is set, transmitting data stops. The SIOF turns "0".

When the external clock is used, it is also necessary to clear the SIOS to "0" before new data is shifted; otherwise, dummy data is transmitted and operation ends.

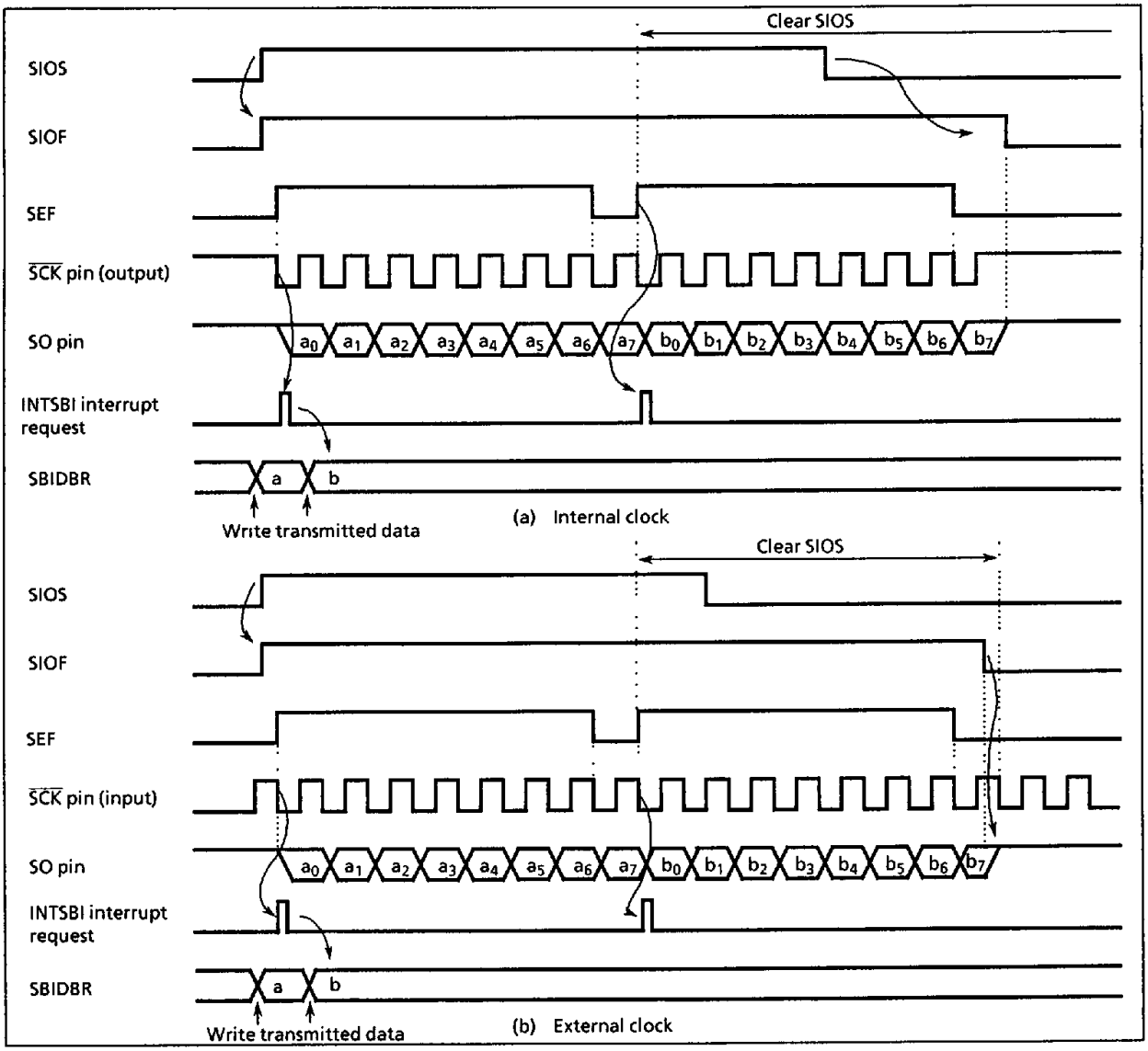


Figure 2-44. Transfer Mode

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

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

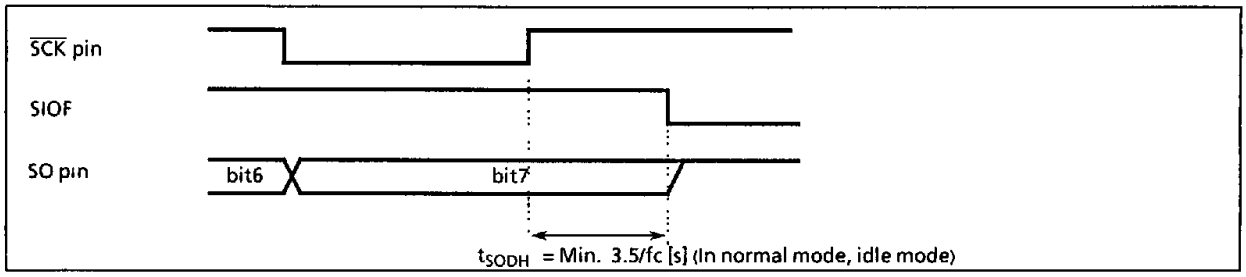


Figure 2-45. Transmitted Data Hold Time at end of transmit

b. 8-bit Receive Mode

Set a control register to a receive mode and the SIOS to "1" for switching to a 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 before new data is transferred to the SBIDBR. 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, conclude receiving data by clearing the SIOS to "0", read the last data, and then switch the mode.

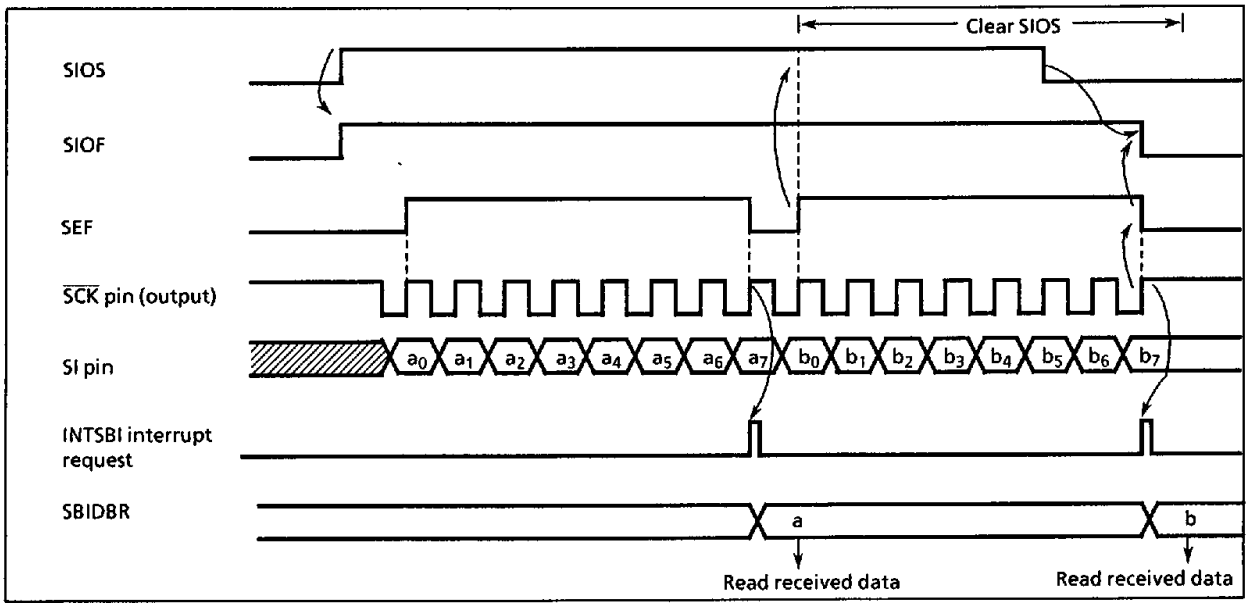


Figure 2-46. Receive Mode (Example : Internal clock)

c. 8-bit Transmit/Receive Mode

Set a control register to a transmit/receive mode and write data to the SBIDBR. After the data is written, set the SIOS to "1" to start transmitting/receiving. When transmitting, the data is output from the 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.

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 byclearing the SIOS to "0", read the last data, and then switch the transfer mode.

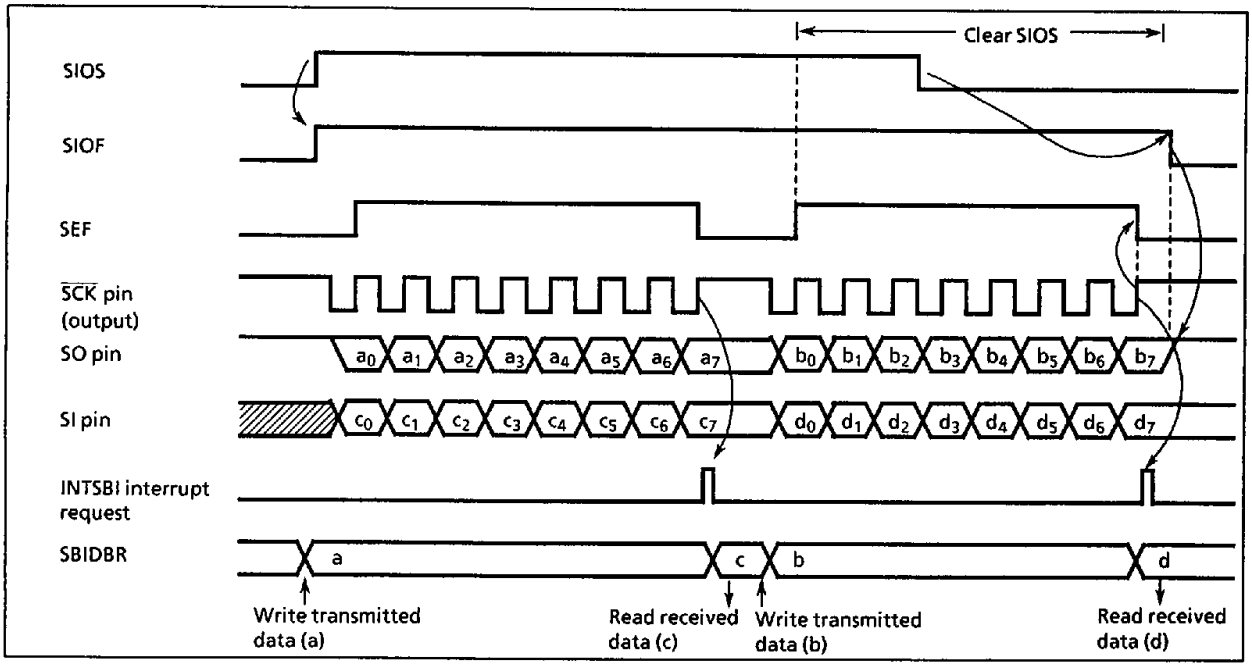


Figure 2-47. Transmit/Receive Mode (Example : Internal clock)

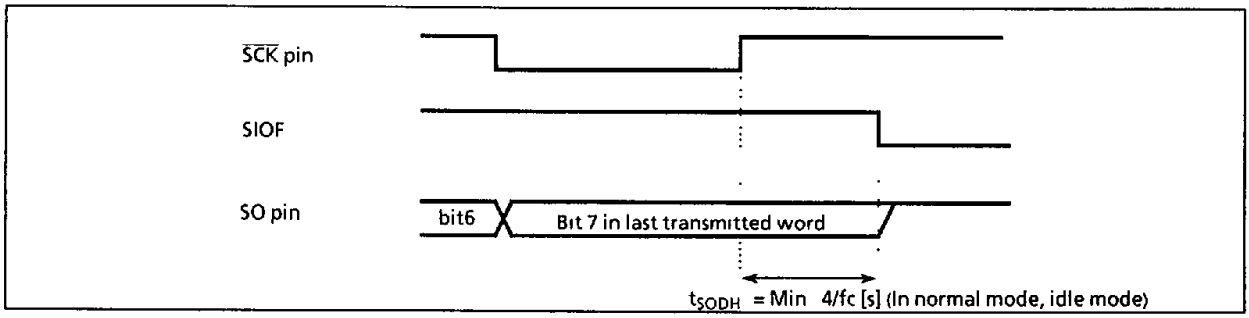


Figure 2-48. Transmitted Data Hold Time at end of transmit/receive

2.9 Remote control signal processor/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. When the remote control signal processor/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 processor/external interrupt 3 input pin, it can be used for normal I/Os.

2.9.1 Configuration

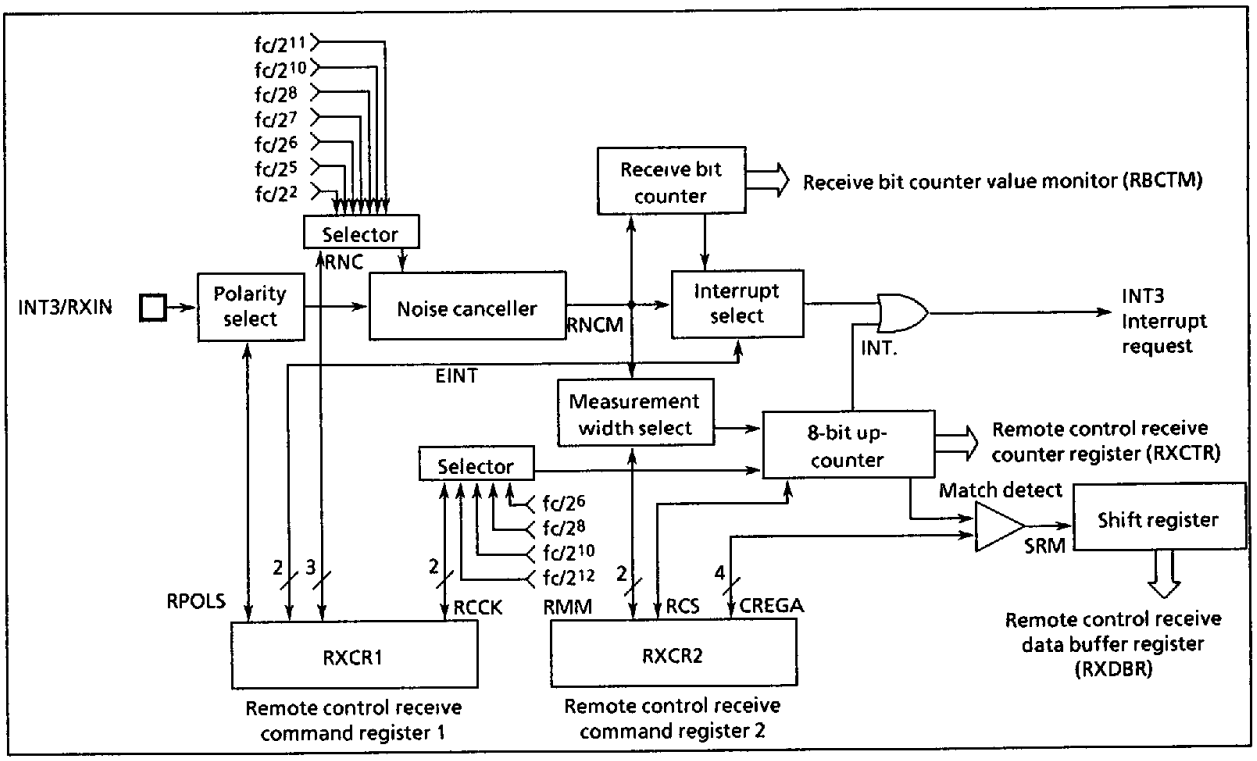


Figure2-49 Remote control signal processor

2.9.2 Remote control signal processor control

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

- Remote control receive control register 1 (RXCR1)
- Remote control receive control register 2 (RXCR2)
- Remote control receive counter register 1 (RXCTR)
- Remote control receive data buffer register 2 (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: Both edges 11: 8-bit receive end	
RNC	Noise canceler 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 : After reset, RPLOS do 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 x RCCK x CREGA [s] CREGA = 1~16h Example : CREGA = 2h, RCCK = $fc/2^6$ [Hz] ,at fc = 8MHz 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: 10: 11: See table 2.6	

Note1 : When an interrupt source is set for both edges, low and high widths are forcibly measured separately.

Note2 : Set CREGA (1~16) before EINT sets to 8-bit receive end.

Figure2-50 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 : Measurement data < when Tth = "0" 1 : Measurement data > when Tth = "1"	
RNCM	Remote control signal monitor after passing through noise canceler		

Figure2-51 Remote control receive counter register, data buffer register, status register

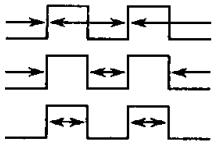
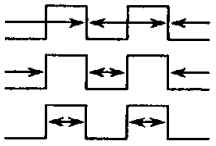
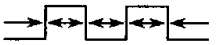
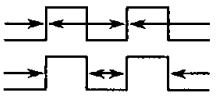
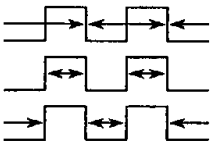
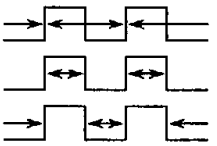
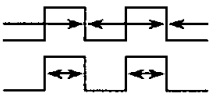
RPOLS	EINT	RMM	Interrupt source	Measurement mode
0	00	00 10 11		
	01	01 10 11		
	10	—		
	11	00 10	Receive end	
1	00	00 10 11		
	01	01 10 11		
	10	—		
	11	00 10	Receive end	

Table2-6 Combination of interrupt source and measurement mode

2.9.3 Noise elimination time setting

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

RNC	Minimum signal pulse width	at fc = 8MHz	Maximum noise width to be eliminated	at fc = 8MHz
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]

Table2-7 Noise elimination time setting

2.9.4 Operation

(1) interrupts at rising, falling, or both edges, and measurement modes

First set EINT and RMM. Next, write "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 (FFH) 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.

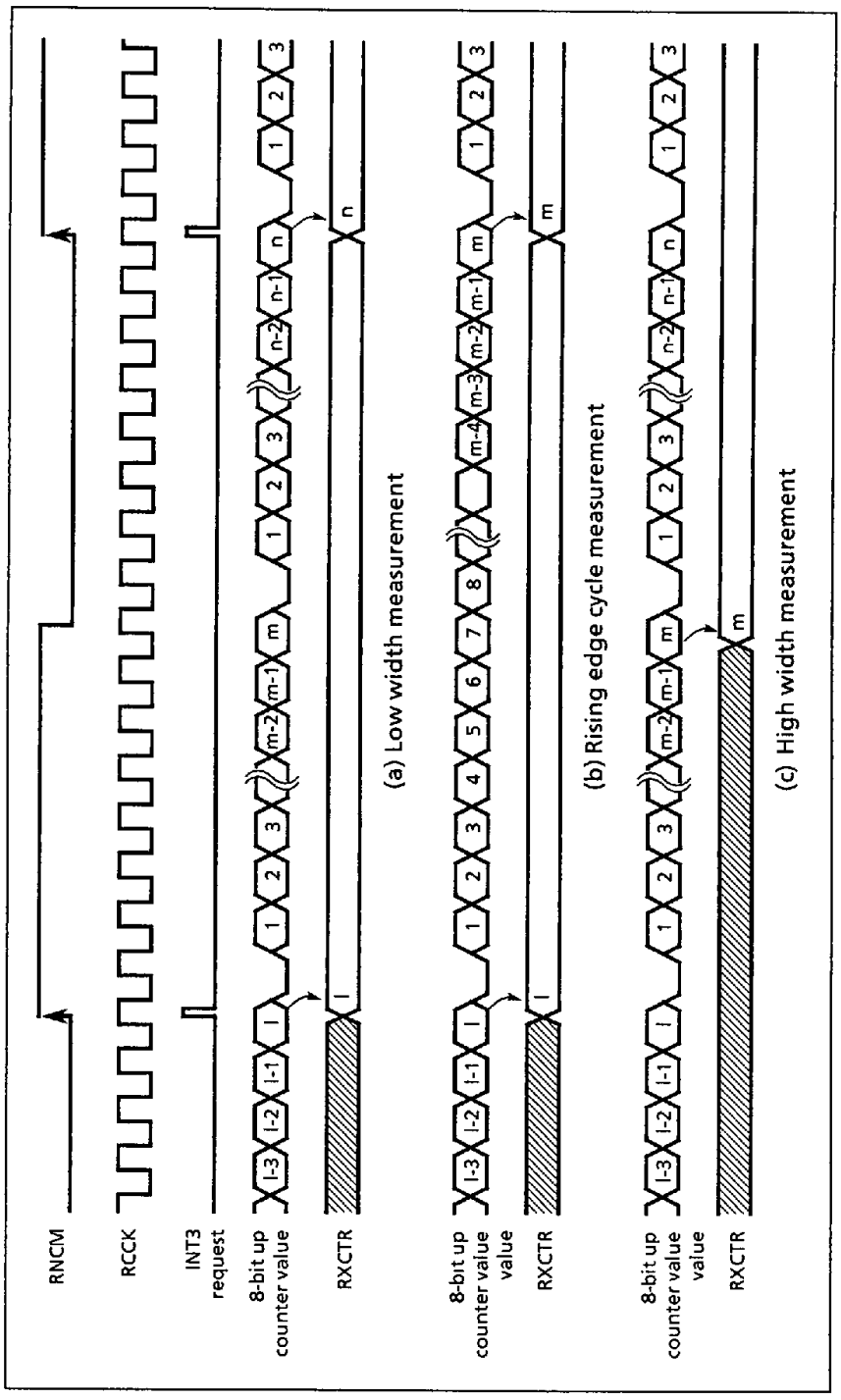


Figure2-52 Rising edge interrupt timing chart (RPOLS = 0)

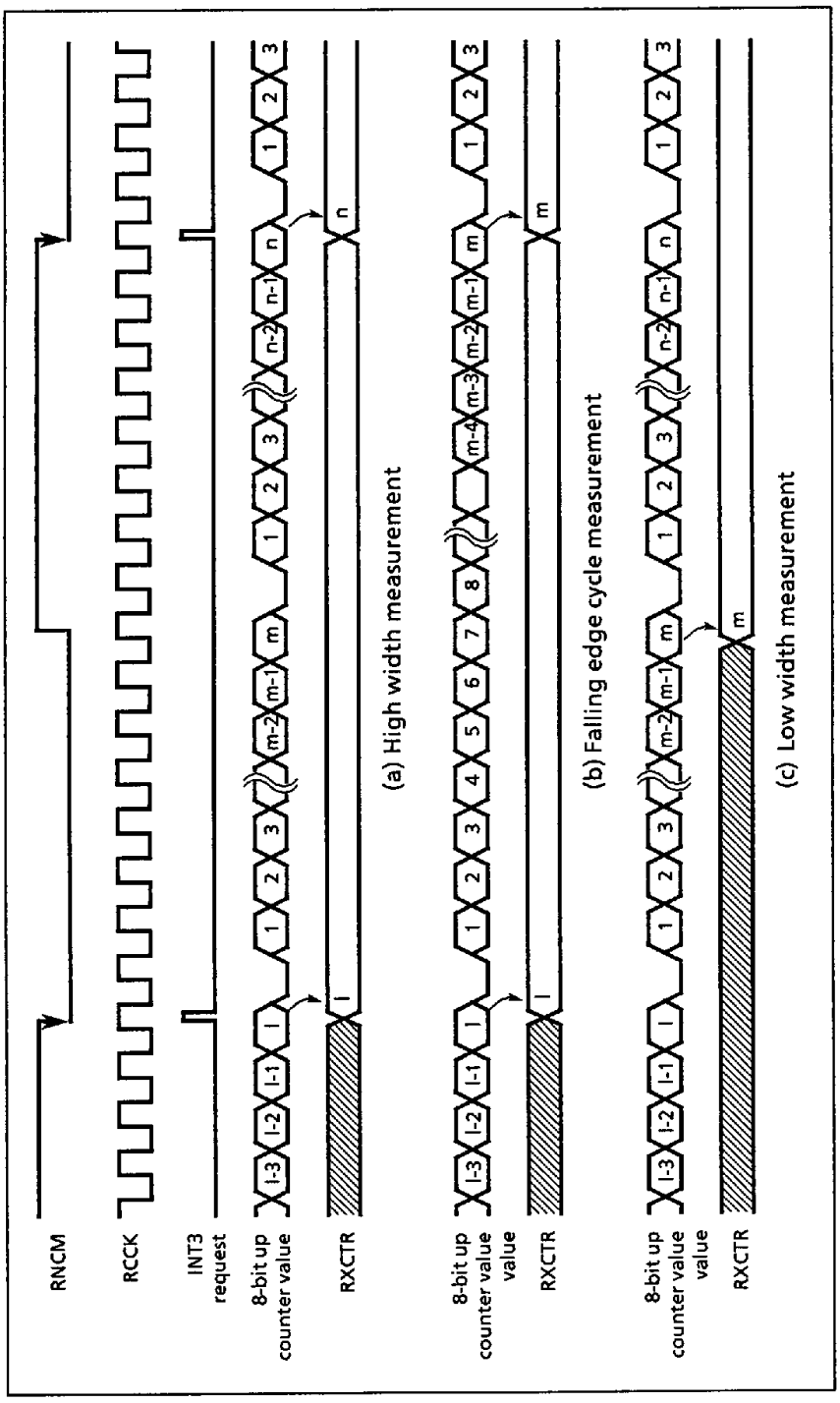


Figure2-53 Falling edge interrupt timing chart (RPOLS = 0)

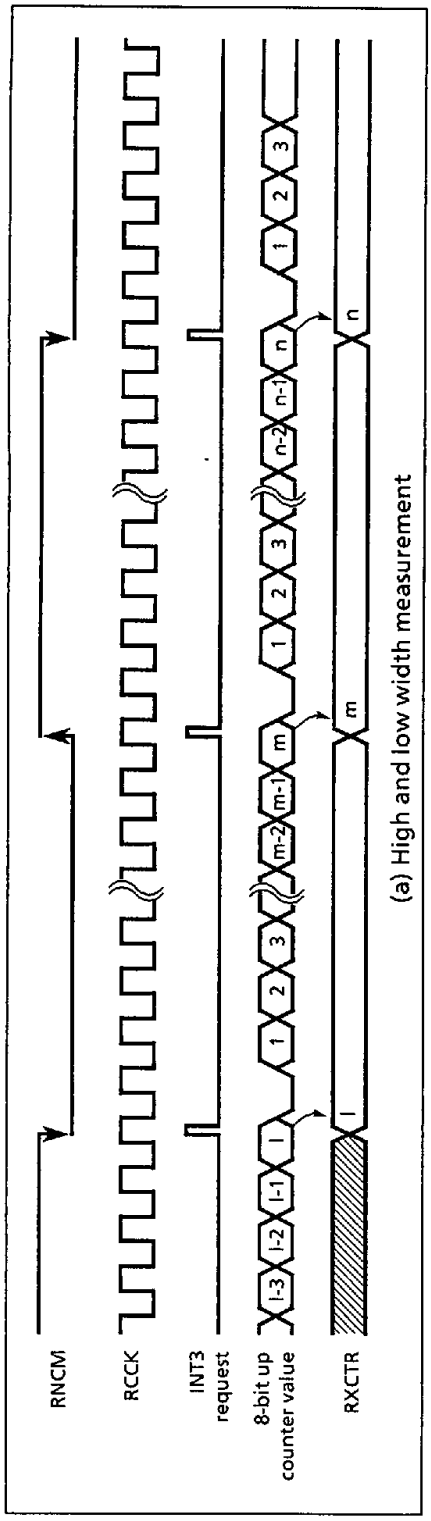


Figure2-54 Both edge interrupt timing chart

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

By determining one-cycle remote control signal as one-bit data set to "0" or one-pulse width remote control signal as one-bit data set to "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 are saved, bit by bit, in RXDBR at the rising edge of the remote control signal (when RPLOS = 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.

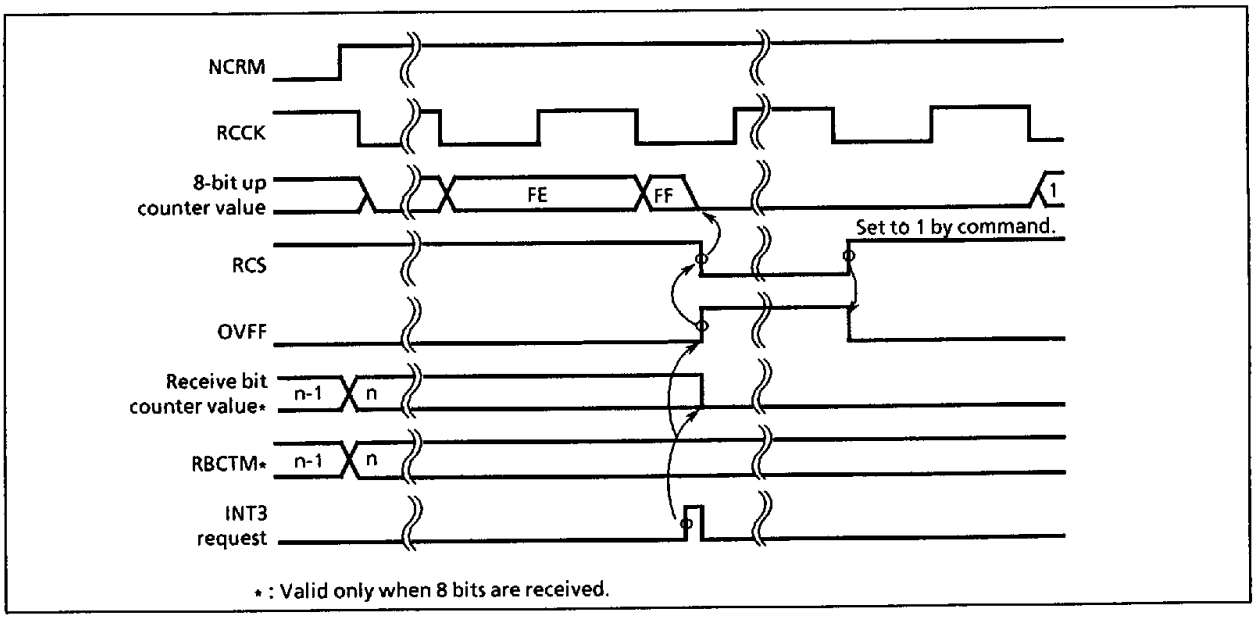


Figure2-55 Overflow interrupt timing chart

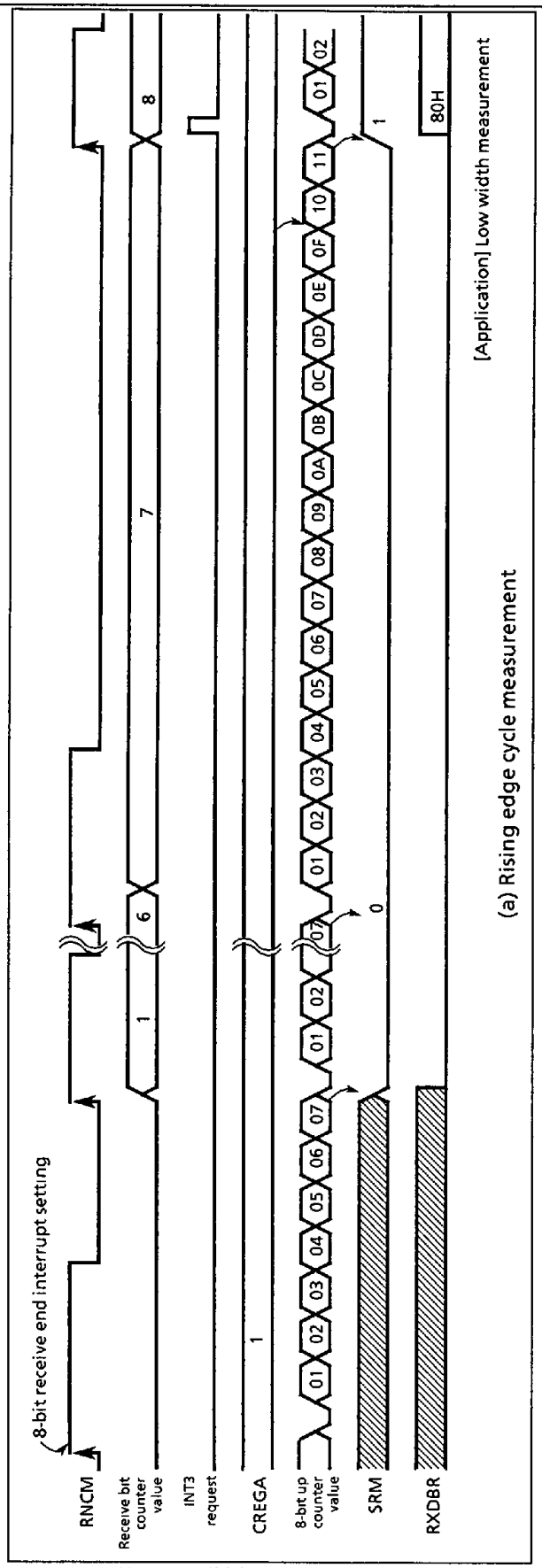


Figure2-56 8-bit receive end interrupt timing chart (RPOLS = 0)

Count clock (RCCK)	at fc = 8MHz			
	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

Table2-8 Count clock for remote control decision circuit

2.10 6-bit A/D Conversion (Comparator) Inputs

The comparator input is an analog input to discriminate key input or AFC (Auto Frequency Control) signal input, etc. The analog input voltage level (pins CIN3 - CIN0) can be detected as 64-stage by setting reference voltage.

The comparator input pins CIN3-CIN0 can also be used as ports P57 - P54.

When used as a comparator input, the output latch should be set to "1".

2.10.1 Configuration

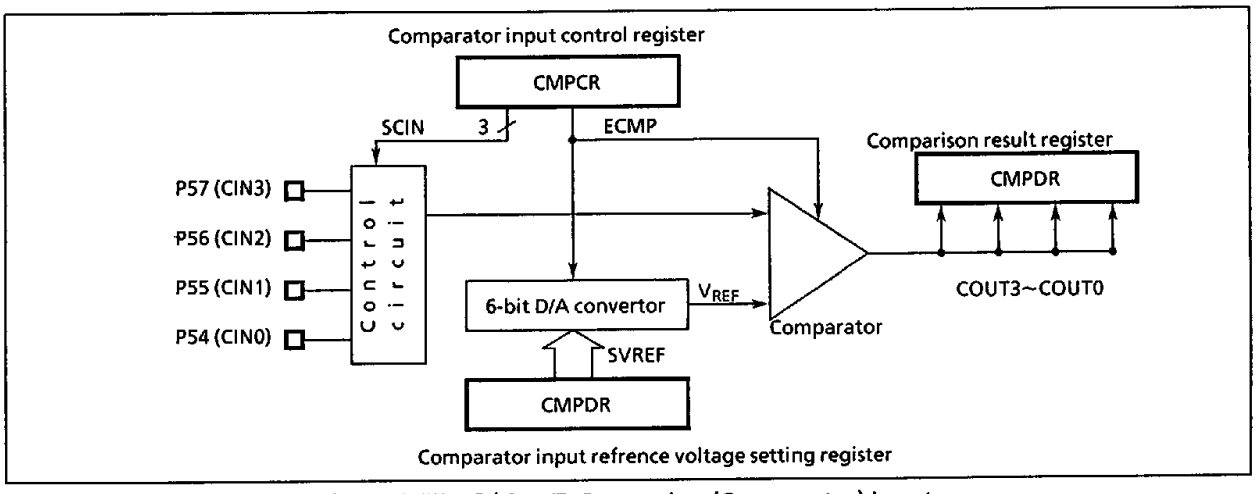


Figure 2-57. 6-bit A/D Conversion (Comparator) Input

2.10.2 Control

A/D conversion (comparator) inputs are controlled by a comparator input control register (CMPCR) and a comparator input data register (CMPDR). The CMPDR contains a reference voltage setting register (write-only) and a comparison result register (read-only).

Comparator Input Control Register

CMPCR ($000E_H$)

7	6	5	4	3	2	1	0
ECMP					SCIN		

(Initial value : 0*** *000)

ECMP	Comparator input control	0 : Disable Comparator input 1 : Enable Comparator input	write only
SCIN	Number of Comparator input channels select	000 : 1 channel (CIN0) 001 : 2 channels (CIN0 to 1) 010 : 3 channels (CIN0 to 2) 011 : 4 channels (CIN0 to 3) 1** : Reserved	

Note1 : The CMPCR is a write-only register and must not be used with any of the read-modify-write instructions.

Comparator Input Data Register

CMPDR ($000F_H$)

7	6	5	4	3	2	1	0
"1"	"1"	"1"	"1"	COUT3	COUT2	COUT1	COUT0
		SVREF					

(Initial value : **00 0000)

SVREF	Reference voltage (V_{REF}) setting	$V_{REF} = V_{DD} \times (SVREF + 1) / 64$ [V] (SVREF = 0 to 63)	write only
COUT3 to COUT0	Comparison result	Result of comparing the CIN3 - CIN0 pin analog input voltage with the reference voltage.	read only

*Note : * ; don't care*

Figure 2-58. Comparator Input Control Register and Data Register

2.10.3 Function

Reference voltage (V_{REF}) is set with SVREF (bits 5 - 0 in CMPDR).

$V_{REF} = V_{DDX}$ (SVREF + 1) / 64[V] (SVREF = 0~63)

The number of comparator input channels is selected with SCIN (bits 2 - 0 in CMPCR). Sequential comparison of the selected number of channels is started by setting ECMP (bit 7 in CMPCR) to "1".
The comparison of one channel requires two machine cycles; therefore, the comparison result register (COUT3 - COUT0) should be read out at an interval equal to [number of channels × 2 machine cycles] after setting the reference voltage (V_{REF}). COUT3 - COUT0 are set to "1" if the input voltage (pins CIN3 - CIN0) is higher than the reference voltage (V_{REF}) ; otherwise those are cleared to "0".

Note 1 : When entering STOP mode, ECMP is automatically cleared and SCIN/SVREF are held. And, COUT3 - COUT0 are always set to "1".

Note 2 : Any pins specified for comparator input with SCIN can no longer be used for normal digital input and, are read out as"0".

Note 3 : COUT3 - COUT0 are read out as "1" when not used as a comparator input. For example, bits 7 to 3 in CMPDR are always read out as "1" when SCIN = 010_B.

Example : Comparates the CIN3-CIN0 inputs with V_{REF} = 2.5V (At V_{DD} = 5V).

```
LD      (P5), 11111111B          ; Sets port P5 output latches to "1".
LD      (CMPDR), 00011111B       ; Sets VREF = 2.5V
LD      (CMPCR), 10000011B       ; Sets SCIN to 4 channels and Enables comparator input
        :                        ; 4ch x 2 machine cycles -2
        :                        ;   = 6 machine cycles wait.
LD      A, (CMPDR)               ; Reads CMPDR (COUT0~COUT3).
```

SVREF 5	4	3	2	1	0	V_{REF} [V]
0	0	0	0	0	0	0.078
0	0	0	0	0	1	0.156
0	0	0	0	1	0	0.234
:						:
1	1	1	1	0	1	4.844
1	1	1	1	1	0	4.922
1	1	1	1	1	1	5 000

Table 2-9. Reference Voltage (at V_{DD} = 5V)

2.11 Pulse Width Modulation Circuit Output

87C833/C33/H33 has 10 built-in pulse width modulation (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 ($\overline{PWM0}$) to P47 ($\overline{PWM7}$), P50 ($\overline{PWM8}$), P51 ($\overline{PWM9}$). When these ports are used PWM outputs,the corresponding bits of P4, P5 output latches should be set to "1".

2.11.1 Configuration

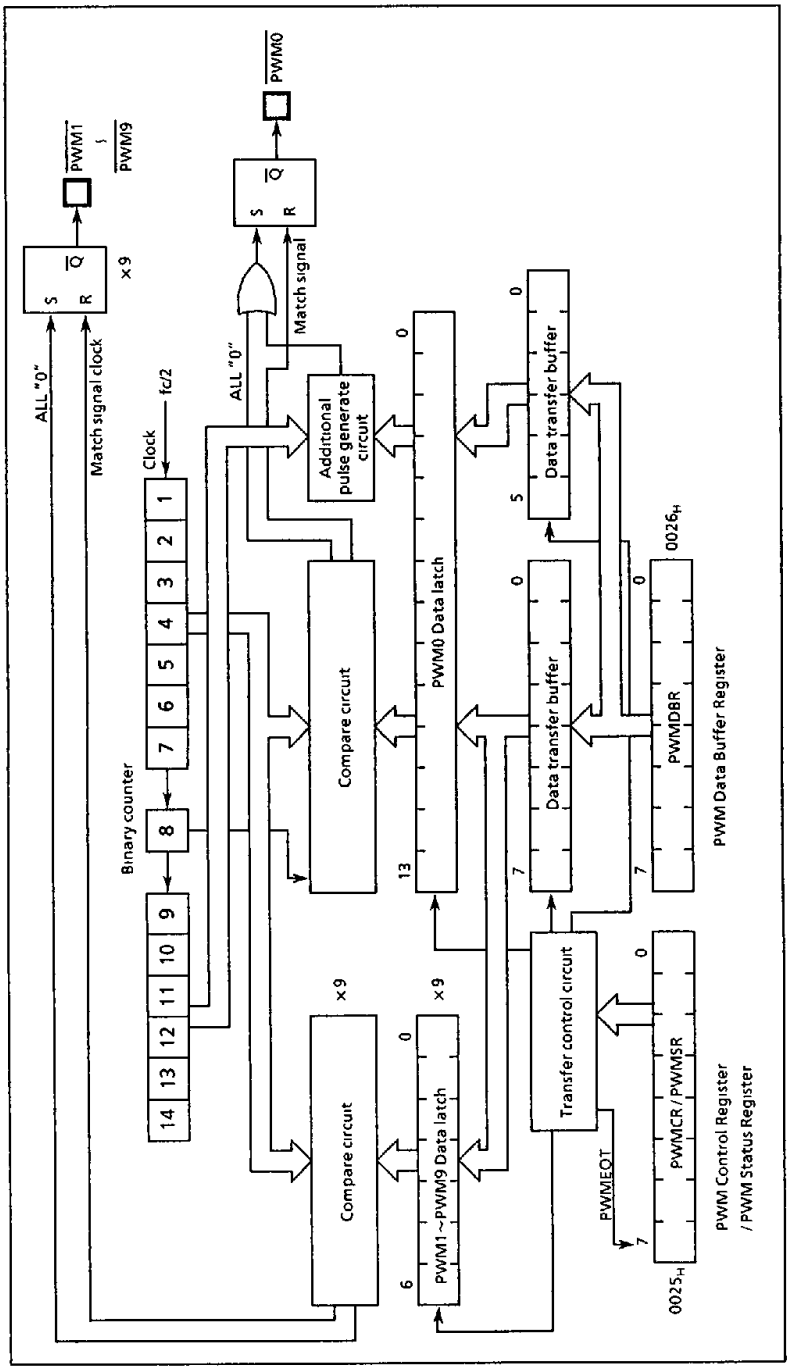


Figure 2-59. Pulse Width Modulation Circuit

2.11.2 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 TS period where the additional pulse is generated is shown in Table 2-10.

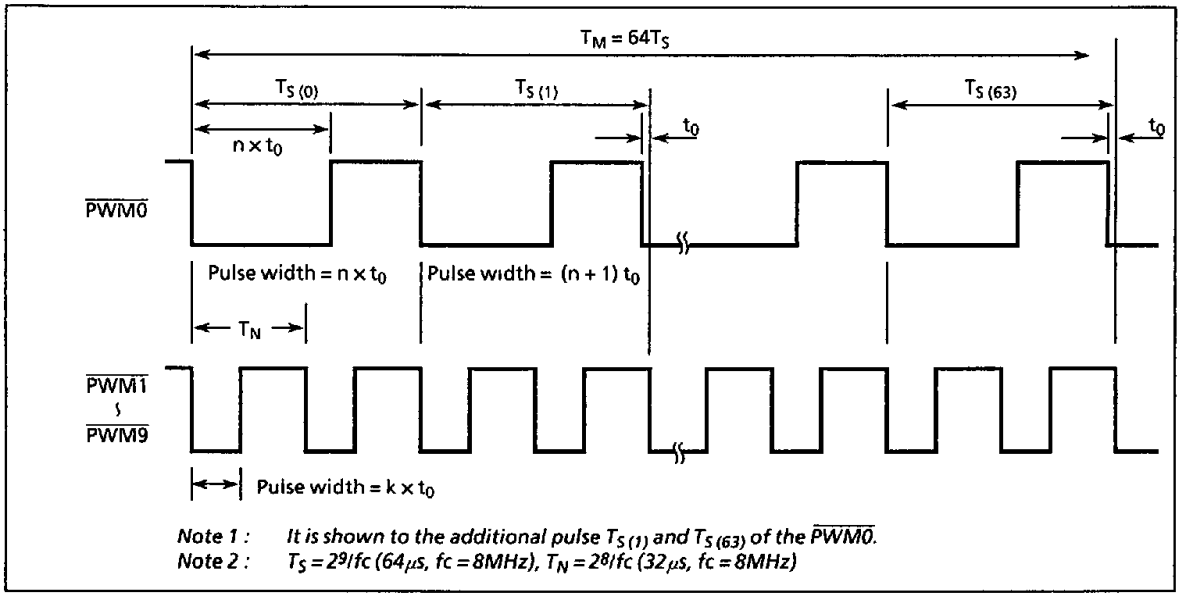
Bit position of 6 bits data	Relative position of Ts where the output pulse is generated. (No. i of $T_{S(i)}$ is listed)
Bit 0	32
Bit 1	16, 48
Bit 2	8, 24, 40, 56
Bit 3	4, 12, 20, 28, 36, 44, 52, 60
Bit 4	2, 6, 10, 14, 18, 22, 26, 30, ·····, 58, 62
Bit 5	1, 3, 5, 7, 9, 11, 13, 15, 17, ·····, 59, 61, 63

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

Table 2-10. Correspondence between 6 Bits Data and the Additional Pulse Generated TS Period

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

These are 7-bit resolution PWM outputs and one period is $T_N = 2^8/fc$ [s]. When the 7bit data are decimal k ($0 \leqq k \leqq 127$), the pulse width becomes $k \times t_0$. The wave form is illustrated in Figure 2-59.



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 = 8MHz), $T_N = 2^8/fc$ (32 μs, fc = 8MHz)

Figure 2-60. PWM Output Wave Form

2.11.3 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).

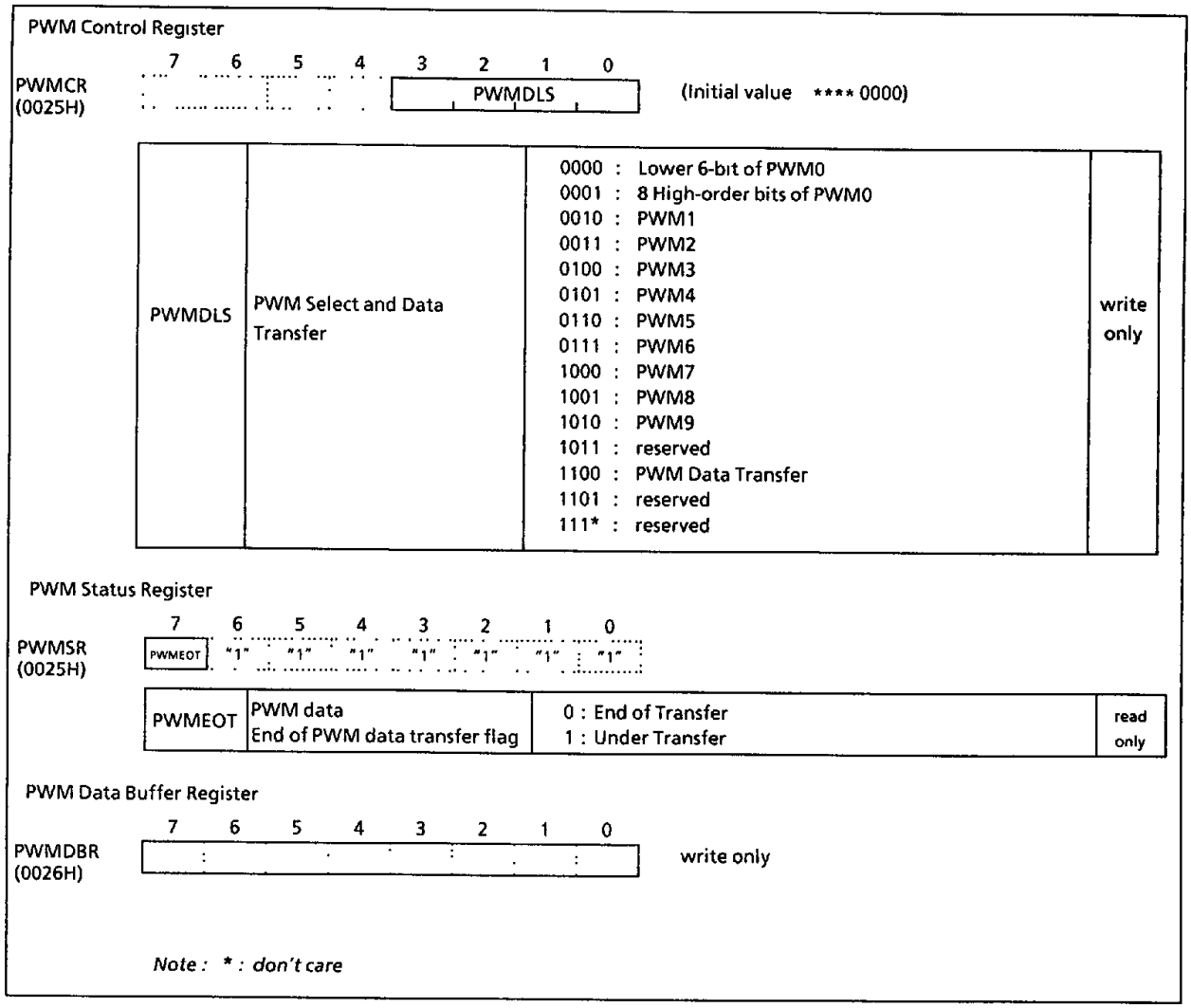


Figure 2-61. PWM Control Register / PWM Status Register / PWM Data Buffer Register

(1) Programing of PWM Data

PWM output is controlled by PWM writing the output data to data latches.
For the writing the output data are divided using the PWM Control Register (PWMCR).

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

Note : When writing the output data to PWM0, write "0CH" to the PWMDLS after writing of the 14-bits 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 "0CH" is written to the data latch until PWM output is switched is 2^{15}/fc [s] (4.096ms, at fc = 8MHz) for PWM0 output and 2^9/fc [s] (64μs, at fc = 8MHz) for PWM1 - PWM9 output.

Example : $\overline{\text{PWM0}}$ pin outputs 32μs pulse width without the additional pulse.
$\overline{\text{PWM1}}$ pin outputs 16μs pulse width.
$\overline{\text{PWM2}}$ pin outputs 8μs pulse width.
Note : at fc = 8MHz

```
        LD      (PWMCR), 00H      ; Select lower 6-bit of PWM0
        LD      (PWMDBR), 00H     ; Without the additional pulse
        LD      (PWMCR), 01H      ; Select 8 high-order bits of PWM0
        LD      (PWMDBR), 80H     ; 32μs ÷ 2/fc = 80H
        LD      (PWMCR), 0CH      ; PWM Data Transfer
WAIT0 : TEST    (PWMSR). 7        ; PWMEOT = 0?
        JRS     F, WAIT0
        LD      (PWMCR), 02H      ; Slect PWM1
        LD      (PWMDBR), 40H     ; 16μs ÷ 2/fc = 40H
        LD      (PWMCR), 0CH      ; PWM Data Transfer
WAIT1 : TEST    (PWMSR). 7        ; PWMEOT = 0?
        JRS     F, WAIT1
        LD      (PWMCR), 03H      ; Select PWM2
        LD      (PWMDBR), 20H     ; 8μs ÷ 2/fc = 20H
        LD      (PWMCR), 0CH      ; PWM Data Transfer
WAIT2 : TEST    (PWMSR). 7        ; PWMEOT = 0?
        JRS     F, WAIT2
```

2.12 Pulse Output Circuit ($\overline{PULSE}$)

Pulse output circuit generates the pulse clock of duty 50% by dividing the High-frequency clock. The pulse output is used for the basic clock for the PLL IC or peripheral ICs. When P52 port is used as the pulse output, set P52 output latch to "1".

Pulse output control command register

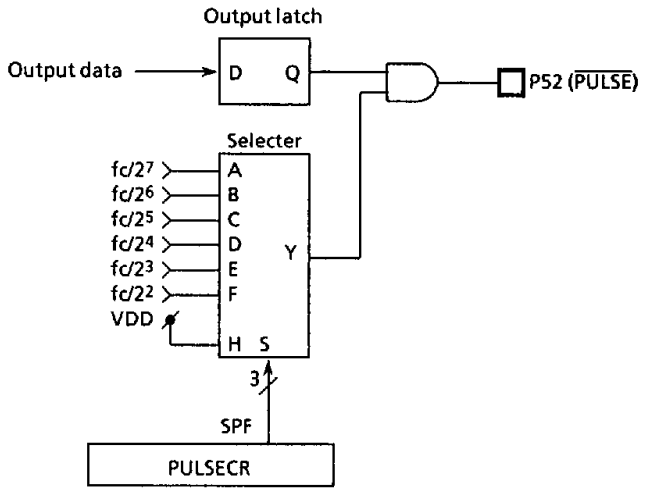
PULSECR (0027_H)

7	6	5	4	3	2	1	0
				SPF			

(Initial value **** 111*)

SPF	Select pulse frequency	000 : $fc/2^7$ [Hz] (62.5kHz, fc = 8MHz) 001 : $fc/2^6$ (125kHz, ") 010 : $fc/2^5$ (250kHz, ") 011 : $fc/2^4$ (500kHz, ") 100 : $fc/2^3$ (1MHz, ") 101 : $fc/2^2$ (2MHz, ") 110 : reserved 111 : Disable the pulse output	R/W

*Note : fc ; High-frequency clock * ; don't care*



Pulse Output Control Command Register

Figure 2-62. Pulse Output Circuit

2.13 On-Screen Display (OSD) Circuit

The TMP87C833/C33/H33 features a built-in on-screen display circuit used to display characters and symbols on the TV screen. There are 256 characters and any characters can be displayed in an area of 32 columns ×8 lines. With an OSD interrupt, additional lines can be displayed. The functions of the OSD circuit meet the requirements of on-screen display functions of closed caption decoders based on FCC standards.

OSD circuit functions are as follows:

① Number of characters : 256 (128 in 87C833).
② Number of display characters : 256 (32 columns x 8 lines). With OSD interrupt, nine or more lines can be displayed.
③ Character matrix : 8 x 9 dots (8 x 13 dots including space around character)
④ Character sizes : 3 (specified by line)
⑤ Display colors:
Character colors : 7 (one per character)
Fringe colors : 7 (one per page)
Background colors : 8 (one per page)
⑥ Fringing and smoothing functions (for large, middle, and small characters)
⑦ Display position : 128 horizontal steps and 256 vertical steps
⑧ Full-raster blanking function
⑨ Blinking function
⑩ Underline
⑪ Solid space
⑫ Slant function (italics)
⑬ Window function

2.13.1 Configuration

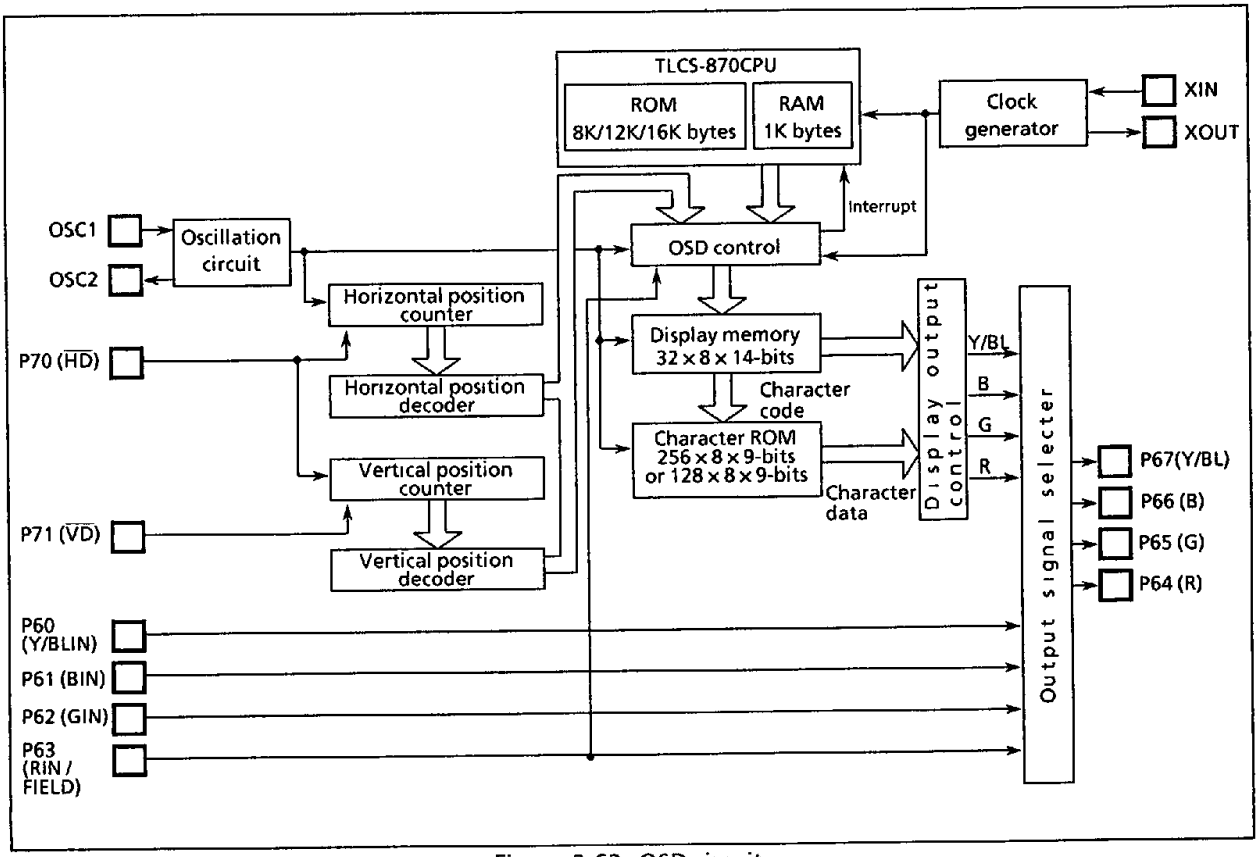


Figure 2-63 OSD circuit

2.13.2 Memories for OSD

(1) Character ROM

The character ROM contains 256 (128 in 87C833) character patterns. The user can set patterns as desired. The character ROM consists of 256 characters stored as 8 x 9 dots (character codes 00_H to FF_H). Each bit in the character ROM corresponds to one dot. 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. Using the character code, the start address in the character ROM can be Calculated as follows :

Start address in character ROM = CRA$(_{10})$ x 16 $(_{10})$

Note: CRA: Character code (00_H to FF_H)

Character code 00_H is fixed as blank data and cannot be changed to represent a different character. Write 00 in the data of character code 00 at presenting character ROM data.

Figure2-64 shows an example of the character pattern configuration for an 8 x 9 bit character (character code 01_H), with the ROM addresses and data.

Set the charactor data in a 6 x 9 dots area put left side, while using slant function.

Figure 2-65 shows the character ROM dump list for the above character pattern.

When ordering a mask, load the data to character ROM starting at address 4000_H going to $4FFF_H$ in the 256K-bit EPROM. Write FF for all the data which has ***9_H to ***F_H as an address in character ROM.

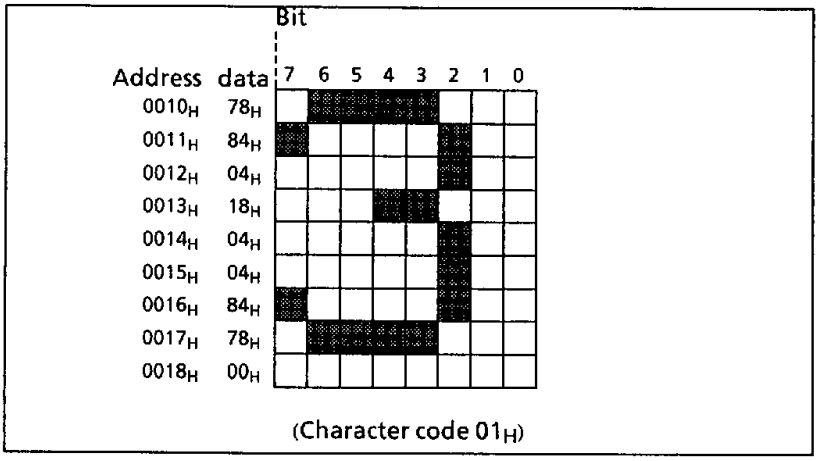


Figure 2-64. 8 x 9 dot pattern configuration

```
0010/ 78 84 04 18 04 04 84 78 00 FF FF FF FF FF FF FF
```

Figure 2-65. Character ROM dump list

(2) Display memory

Each character of the 256 characters displayed in 32 columns x 8 lines consists of 14 bits in the display memory. Five data items are written to the display memory: character code, color data, blinking specification, underline enable, and slant enable. The display memory is in an unknown state at reset.

There are two modes for writing display data to the display memory. One mode is used for writing all display data (character code, color data, blinking specification, underline enable, and slant enable) simultaneously. The other mode is used for changing either character codes or the remaining data items (color data, blinking specification, underline enable, and slant enable). How display data is written to the display memory is described in section 2.13.3 (24).

Display memory configuration

- Character code specification register (8 bits) .. CRA7 to CRA0
- Color data specification register (3 bits) ..... RDT / GDT / BDT

- Blinking specification flag (1 bit) ..... BLF
- Underline enable flag (1 bit) ..... EUL
- Slant enable flag (1 bit) ..... SLNT

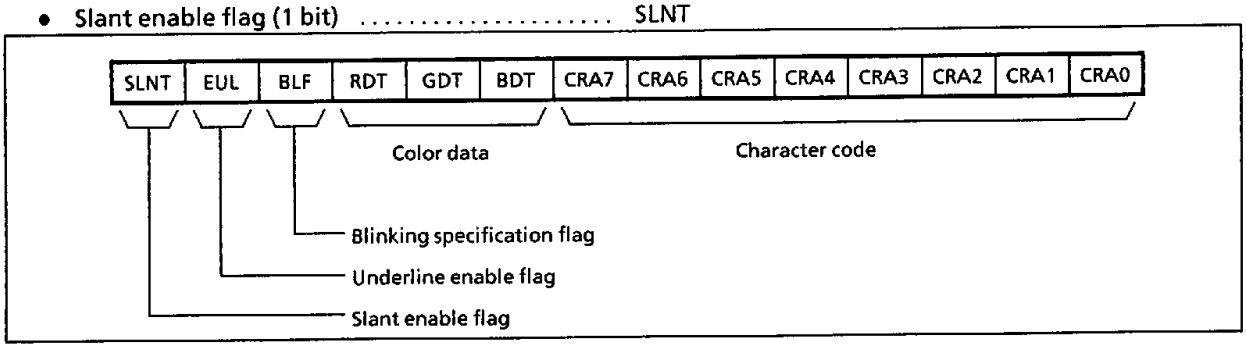


Figure 2-66. Display memory bit configuration

COLUMN / LINE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
2	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
3	40	41																														5F
4	60																															7F
5	80																															9F
6	A0																															BF
7	C0																															DF
8	E0																														FE	FF

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

Figure 2-67. Display memory address configuration

2.13.3 OSD circuit control

The OSD circuit performs control functions using the OSD control registers which reside in addresses $0F80_H$ to $0F98_H$ in the data buffer register (DBR). Section 2.13.3 (28) shows the OSD control registers. To write data to the OSD control registers, use the normal data buffer register access method. The OSD control registers are used to set display start position, display character designs (that is, fringing, smoothing, color data, character size, and etc.), display memory addresses, and character codes.

Setting the display on-off control bit, DON, (bit 0 in ORDON) to 1 enables display (starts display). Setting DON to 0 disables display (halts display).

For write to or read from the OSD control registers, see section 2.13.3 (25).

(1) Display position

In the horizontal direction the display is divided into 128 segments. The horizontal display start position can be set to any value within the 128 segments. The vertical display is divided into 256 segments. The vertical display start position can be set to any value within the 256 segments. The horizontal display start position is specified by OSD control registers HS16 to HS10 (ORHS1). The vertical display start position for line one is specified using VS17 to VS10 (ORVS1). The vertical display start position for lines 2 to 8 are specified by setting VS27 to VS20 (ORVS2) ... VS87 to VS80 (ORVS8).

Horizontal display start position
Specified Page by Page.
Specification steps : 128
Vertical display start position
Specified Line by Line.
Specification steps : 256
Horizontal display start position register (7 bits)
Lines 1 to 8 : HS16 to HS10 (ORHS1)
Vertical display start position registers (8 bits x 8)
Line 1: VS17 to VS10 (ORVS1)
Line 2: VS27 to VS20 (ORVS2)
: :
Line 8: VS87 to VS80 (ORVS8)
Horizontal display start position
$HS1 = \{(HS16 \sim HS14) \times 16^1 + (HS13 \sim HS10) \times 16^0\} \times 2T_{OSC} + 10T_{OSC}$
Vertical display start position
When VDSMD = 0, normal mode
Line n: $VSn = \{(VSn7 \sim VSn4) \times 16^1 + (VSn3 \sim VSn0) \times 16^0\} \times 1T_{HD}$
When VDSMD = 1, double scan mode
Line n: $VSn = \{(VSn7 \sim VSn4) \times 16^1 + (VSn3 \sim VSn0) \times 16^0\} \times 2T_{HD}$
T_{OSC} : One cycle of OSC oscillation
T_{HD} : One cycle of $\overline{HD}$ signal

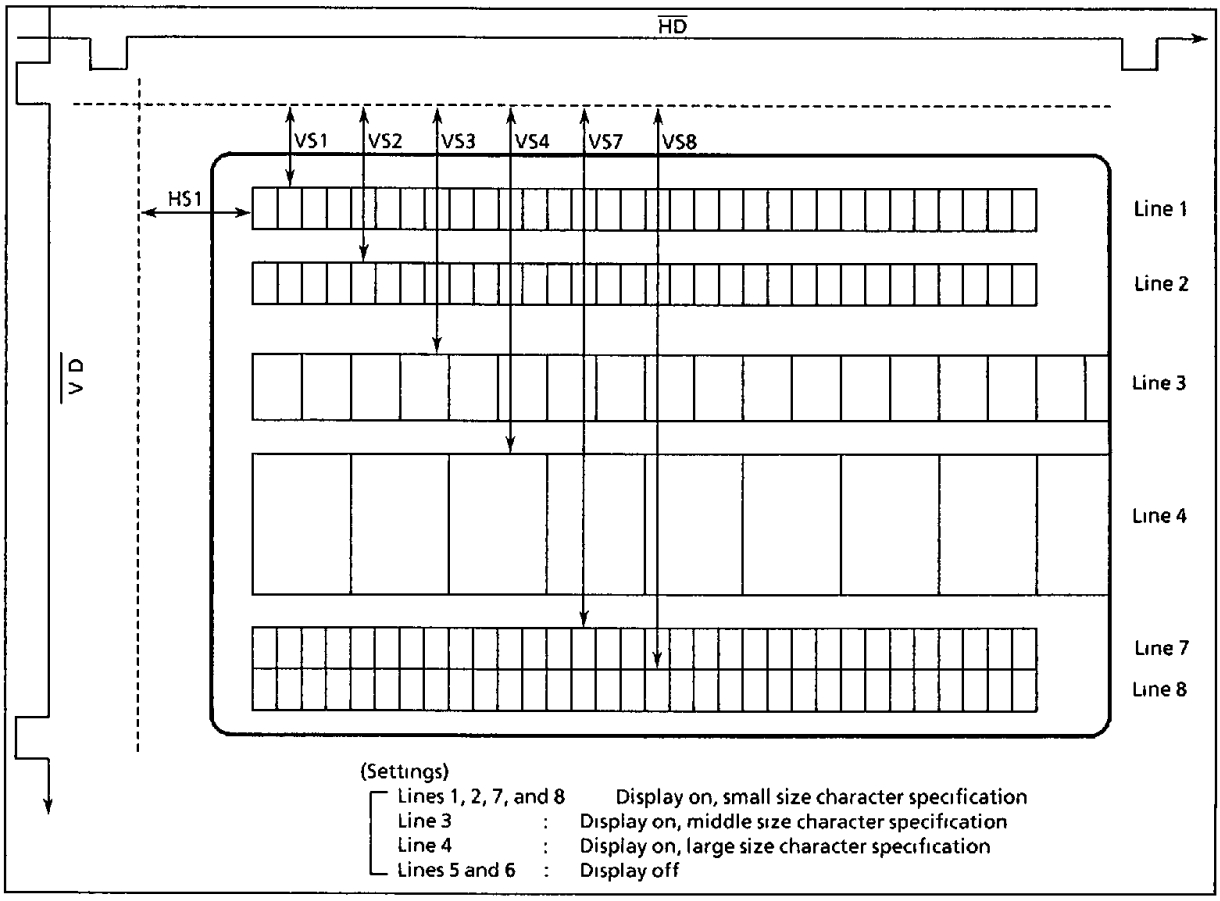


Figure 2-68. TV screen display image

- Notices for setting horizontal display start position

  Lines of OSD (VS1 to VS8) are fixed priority levels as follows:

  VS1 > VS2 > VS3 > VS4 > VS5 > VS6 > VS7 > VS8

  When horizontal display start positions are met as follows:

  VS1 ≦ VS2 ≦ VS3 ≦ VS4 ≦ VS5 ≦ VS6 ≦ VS7 ≦ VS8,

  if higher priority level line overlapps lower one, lower one is not displayed.

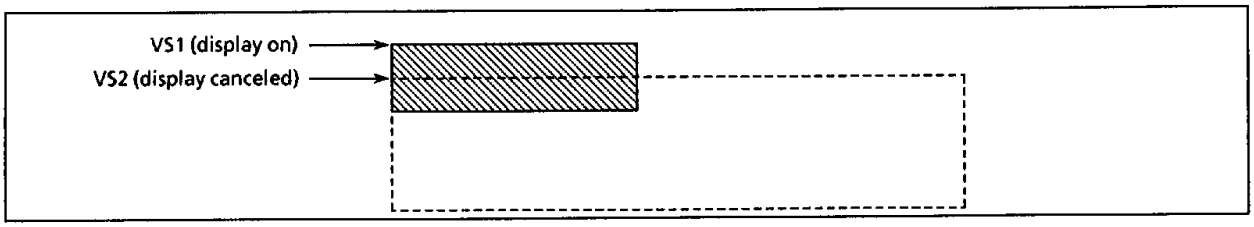


Figure 2-69. Occasion of overlapping (VS1 ≦ VS2)

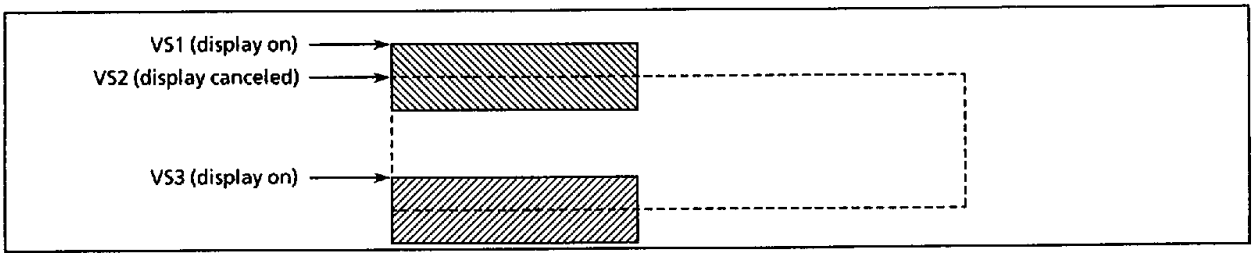


Figure 2-70. Occasion of overlapping (VS1 ≦ VS2 ≦ VS3)

Then the display line counter (refer to section 2.13. (18)) does not count up canceled lines.

When horizontal display start position is met as follows:

VS1 > VS2 > VS3 > VS4 > VS5 > VS6 > VS7 > VS8,

if higher priority level line overlapps lower one, lower one is changed to higher one in the middle of lower one.

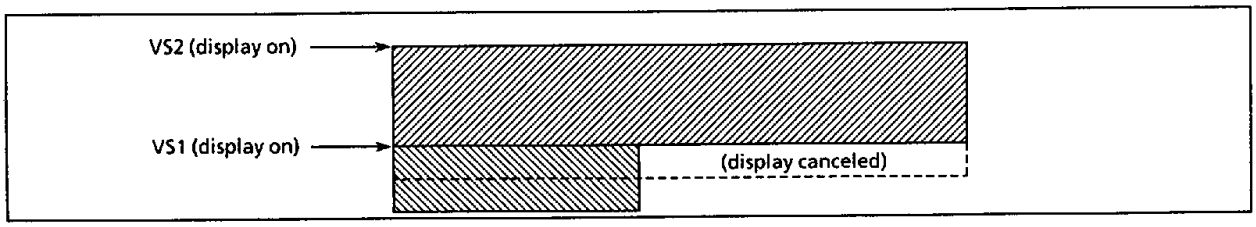


Figure 2-71. Occasion of overlapping (VS1 > VS2)

Then the display line counter counts up only one.

If higher priority level line is filled space character, under side of lower one is broken off.

If display lines are overlapped each other, there is a canceled display line. Set a space for not overlaping display lines, or set horizontal display start position out of display area.

(2) Double scan mode

The OSD circuit has a double scan mode. This enables counting by double steps in the vertical direction to handle non-interlaced scanning TVs. This mode also enables vertical display position to be set for the whole screen. Setting the OSD control register VDSMD (bit 4 in ORETC) to 1 sets double scan mode; resetting, normal mode.

Double scan mode select register (1 bit) ..... VDSMD (bit 4 in ORETC)
"0" ..... Normal mode
"1" ..... Double scan mode

(3) Character sizes and display on/off

There are three character sizes: large, middle and small. One character size can be specified for each line. Display on/off can also be specified for each line. Character size and display on/off are specified using OSD control registers CS11, CS10...CS81, CS80 (ORCS4 and ORCS8).

Character sizes : Large, middle, small
Character size and display on/off specification unit: Line
Character size select/display on/off register (2 bits x 8)
For line 1: CS11 and CS10 (bits 1 and 0 in ORCS4)
For line 2: CS21 and CS20 (bits 3 and 2 in ORCS4)
: :
For line 8: CS81 and CS80 (bits 7 and 6 in ORCS8)

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

Table 2-11. Character size and display on/off specifications (n : 1 to 8)

		VDSMD = 0, (normal mode)		VDSMD = 1, (double scan mode)	
		Dot size	Character size	Dot size	Character size
FORS = 0 (normal mode)	Small	2 T_{OSC} × 1 T_{HD}	16 T_{OSC} × 9 T_{HD}	2 T_{OSC} × 2 T_{HD}	16 T_{OSC} × 18 T_{HD}
	Middle	4 T_{OSC} × 2 T_{HD}	32 T_{OSC} × 18 T_{HD}	4 T_{OSC} × 4 T_{HD}	32 T_{OSC} × 36 T_{HD}
	Large	8 T_{OSC} × 4 T_{HD}	64 T_{OSC} × 36 T_{HD}	8 T_{OSC} × 8 T_{HD}	64 T_{OSC} × 72 T_{HD}
FORS = 1 (double frequency mode)	Small	1 T_{OSC} × 1 T_{HD}	8 T_{OSC} × 9 T_{HD}	1 T_{OSC} × 2 T_{HD}	8 T_{OSC} × 18 T_{HD}
	Middle	2 T_{OSC} × 2 T_{HD}	16 T_{OSC} × 18 T_{HD}	2 T_{OSC} × 4 T_{HD}	16 T_{OSC} × 36 T_{HD}
	Large	4 T_{OSC} × 4 T_{HD}	32 T_{OSC} × 36 T_{HD}	4 T_{OSC} × 8 T_{HD}	32 T_{OSC} × 72 T_{HD}

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

Table 2-12. Dot and character sizes

(4) Character configuration

The area for a character consists of 8 x 13 dots: character display area, underline display area, and space area. A display character is specified by a character code; underline display is enabled or disabled by the underline enable flag. Figure 2-72 shows a display character image.

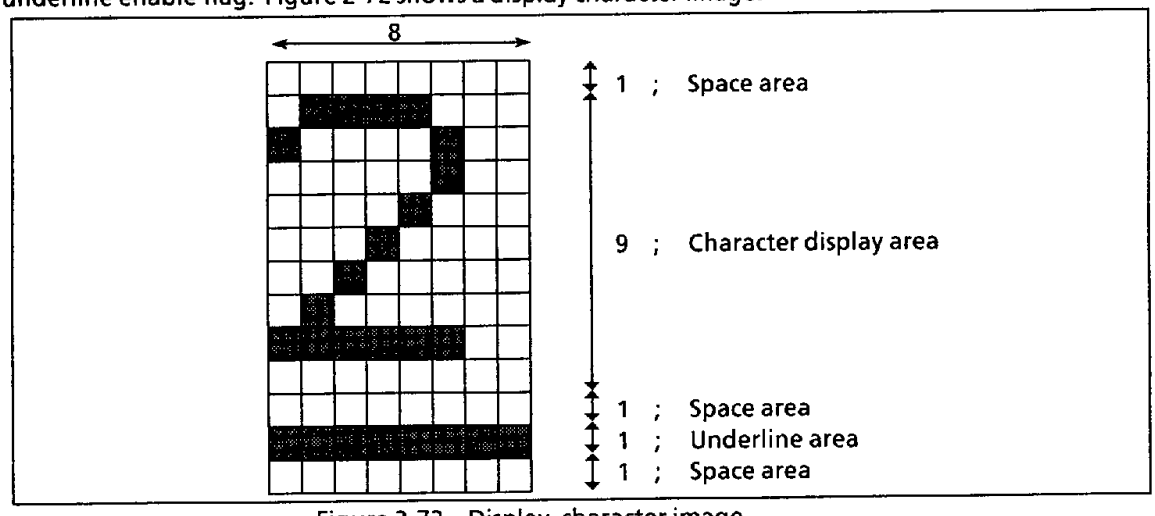


Figure 2-72. Display character image

(5) Smoothing function

The smoothing function is used to make characters look smooth. Enabling smoothing displays 1/4 of a dot between two dots which are connected corner to corner within a character as shown in the above figure 2-73. Smoothing is enabled by setting ESMZ (bit 5 in ORETC) in the OSD control register.

Smoothing specification unit: Page

Smoothing specification register (1 bit) ..... ESMZ (bit 5 in ORETC)
"0" ..... No smoothing
"1" ..... Smoothing enabled

(6) Fringing function

The fringing function is used to display a character with a fringe (width is $1/2$ dot) which has a different color from that of the character. For small characters, the fringe width is 1 dot. Fringing width is 1 dot, when character size is small. when a character has dots which are active on the edge of the 8 x 9 character area, the fringe exceeds the boundaries of the character area by $1/2$ a dot (1 dot for small characters). This occurs on the top, left, and right of the character area is displayed with the maximum of 8 vertical dots and 9 horizontal dots, the fringe exceeds right and left, and top of the character display area. If there is an adjacent character whose outer dot is active, then this dot will overrule the fringe in the horizontal direction.

Underlines are not fringed.

Fringing is enabled for each line by setting EFR1 to EFR8 (OREFR) in the OSD control register to "1".

A fringe color, which is common to all lines, is specified using OSD control registers, RFDT, GFDT, and BFDT, (bits 2 to 0 in ORBK). Do not enable both fringing and solid space simultaneously.

Specified Line by line
Fringe color specification unit: Common to all lines (1 color can be selected in 7 colors.)

Fringing enable register (1 bit x 8)

line 1	EFR1 (bit 0 in OREFR)
line 2	EFR2 (bit 1 in OREFR)
:	:
line 8	EFR8 (bit 7 in OREFR)

Fringing specification

EFRn (n: 1 to 8)

"0" ..... No fringing

"1" ..... Fringing enable

Fringe color register (3 bits) RFDT, GFDT, BFDT (bits 2 to 0 in ORBK)

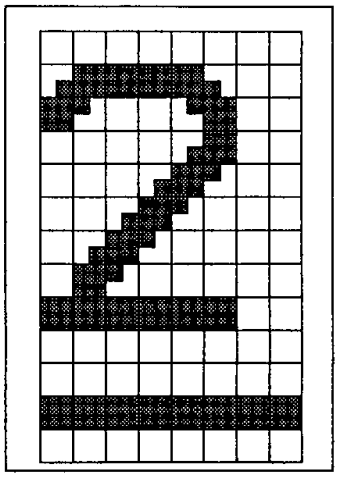


Figure 2-73. Smoothing

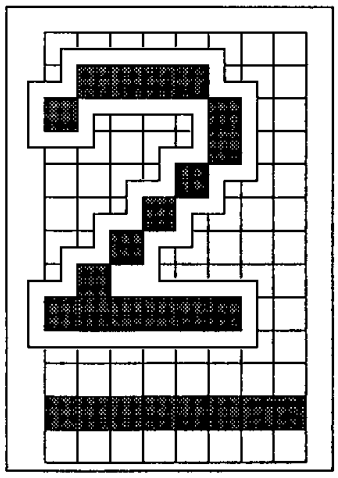


Figure 2-74. Figure Fringing

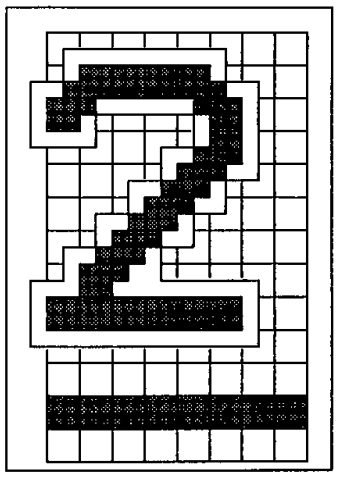


Figure 2-75. Priority of Smoothing and Fringing

(7) Background function

The background function is first used to delete the color of areas in the original display page used for displaying characters (8 x 13 dots per character), then colors those areas with a background color. (Except for areas whose character code is blank data)

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

A background color is specified for each display page by setting the OSD control registers, RBDT, GBDT, and BBDT (bits 5 to 3 in ORBK) to "1".

Background specification unit: display page (1 color can be selected in 8 colors.)

Background enable register (1 bit) EBKGD (bit 7 in ORBK)

"0" . ..... No background function

"1" .. .. Background function enable

Background color specification registers (3 bits) RBDT, GBDT, BBDT (bits 5 to 3 in ORBK)

(8) Full-raster blanking function

The full-raster blanking function is used to first delete the display page colors, then add color to it. Display page video signal is used to first delete by BL signal. When you use the full-raster blanking function, output BL signal from Y/BL pin, because you cannot first delete display page video signal by Y signal.

This function is specified for each display page by setting OSD control register EXBL (bit 6 in ORBK) to "1".

Color specification: Same as that for background.

Full-raster blanking specified display page by display page.

Full-raster blanking enable register: . ' EXBL (bit 6 in ORBK)

"0" ..... No full-raster blanking
"1" ..... Full-raster blanking

Full-raster blanking color specification registers (3 bits) ... RBDT, GBDT, BBDT (same as background color)

(9) f_{OSC} frequency select

This function is to select f_{OSC} frequency mode. By setting FORS (bit 6 in ORIV) to "1", the OSD circuit is operated with clock ($2 \times f_{OSC}$).

f_{OSC} frequency mode select register (1 bit) ··· FORS (bit 6 in ORIV)

"0" ··· Normal frequency mode
"1" ··· Double frequency mode

(10) First/second field phase select

First/second field phase priority select

First/second field phase priority select register (1 bit) ··· FMS (bit 5 in ORIV)

"0" ··· First field has higher priority.
"1" ··· Second field has higher priority.

(11) Field decision mode select

Reset timing selection for horizontal position counter:

Field decision mode select register (1bit) ··· FRSMS (bit 7 in ORIV)

"0" ··· Reset by a field (reset at falling edge of $\overline{VD}$ signal)
"1" ··· Reset by a frame (reset at falling edge of first field $\overline{HD}$ signal and falling edge of 263rd $\overline{HD}$ signal by a frame)

(12) Character

Characters: 256 (including blank data).

Character specification register (8 bits) ··· CRA7 to CRA0 (bits 7 to 0 in ORCRA)
CRA7 to CRA0 (in the display memory)

"00" ··· Blank data
"01" to "FF" ··· User programmable by character ROM

(13) Character color

Character colors: 7
Character color specification unit: character
Character color specification register (3 bits) ··· RDT, GDT, BDT (in the display memory)
RDT, GDT, BDT (bits 2 to 0 in ORDSN)

RDT	GDT	BDT	Character color
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

Table 2-13. Character color

(14) Solid space control

Solid space control is used to display one column of solid space to the left and right of 32 columns on a given display page.

Solid space control is used to delete the color in the areas where solid spaces are located in the original display page, then add color to them.

Solid space specification unit: line

Solid space specification register (16 bits)

For line 1 ··· SOL11 and SOL10 (bits 1 and 0 in ORSOL4)
For line 2 ··· SOL21 and SOL20 (bits 3 and 2 in ORSOL4)
⋮ ⋮
For line 8 ··· SOL81 and SOL80 (bits 7 and 6 in ORSOL8)

Solid space specification

The solid space control functions as follows:

SOL*1/SOL*0
"00" ··· No solid space display
"01" ··· Solid space display left for 32 columns
"10" ··· Solid space display right for 32 columns
"11" ··· Solid space display left and right for 32 columns

Solid space color specification registers (3 bits) ··· RBDT, GBDT, BBDT (bits 5 to 3 in ORBK)
(same color as that of background)

(15) Underline function

Underline function is used to underline a display character. The underline is same color as that of character.

Underline specification unit: Character

Underline enable register (1 bit) ··· EUL (bit 4 in ORDSN) (this resides in the display memory)
"0" ··· No underline
"1" ··· Underline

Underline color specification registers (3 bits) ··· RDT, GDT, BDT (bit 2 to 0 in ORDSN)
(this resides in the display memory, same color as that of character)

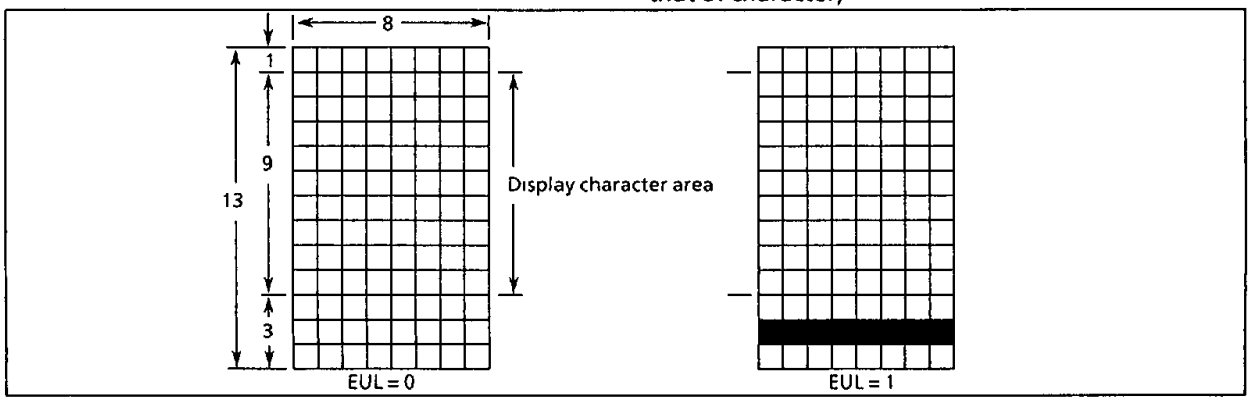


Figure 2-76. Underline

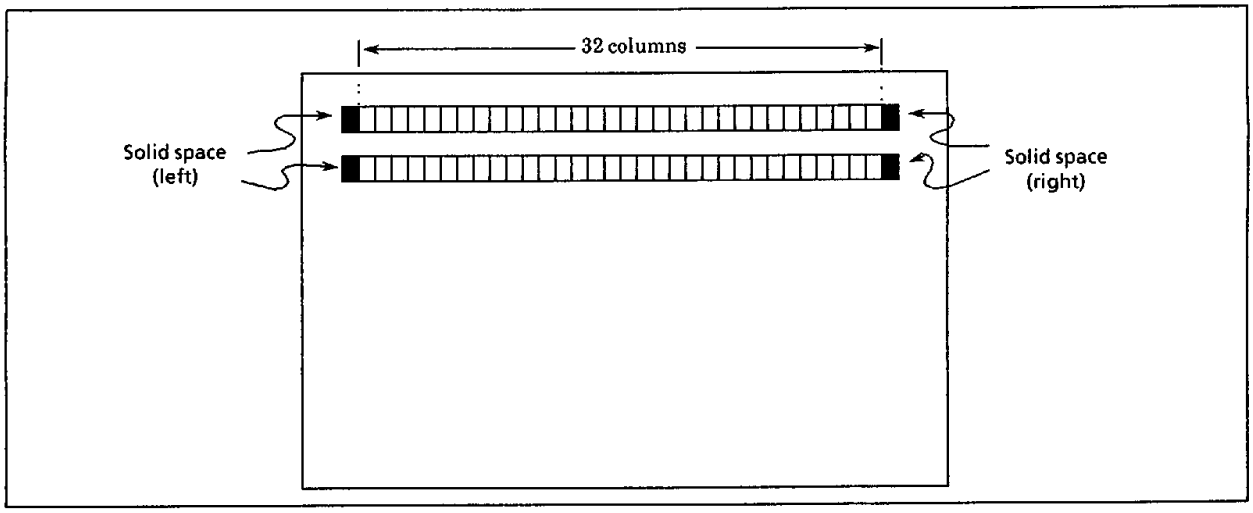


Figure 2-77. Solid space

(16) Blinking function

Blinking function is used to blink any display character.
When BKMF = 1, characters specified to blink by BLF are not displayed. (Space is displayed. That is, if the background color function is used, the background color will not disappear.)

Blinking specification unit: Character

Blinking specification register (1 bit) ··· BLF (bit 3 in ORDSN) (in the display memory)
"0" ··· No blinking
"1" ··· Blinking

Blinking master flag (1 bit) ··· BKMF (bit 6 in ORETC)
"0" ··· Blinking function disable
"1" ··· Blinking function enable
(Characters whose BLF is set to "1"are not displayed.)

(17) Slant function

Slant function is used to slant characters for italics.

Slant specification unit: Character

Slant enable register (1 bits) ··· SLNT (bit 5 in ORDSN) (in the display memory)
"0" ··· No slant
"1" ··· Slant

Note : If background and slant functions are enabled simultaneously, background area is also slanted, because slant function is affected by a character display area.
When fringe exceeds the boundaries of the character area, protruding area is not slanted.

(18) Multiple line display by OSD interrupt

Nine or more lines can be displayed using OSD interrupts. By changing the display start position and display data after each line, the additional line will appear on screen.

Interrupt source select register (1 bit) ··· SVD (bit 7 in ORIRC)

"0" ··· An interrupt request is generated when scanning of a line specified by the value set in ISDC is finished (falling edge of $\overline{HD}$ signal).

"1" ··· An interrupt request is generated at falling edge of $\overline{VD}$ signal

Display line counter

4-bit counter used to indicate a line being displayed.

This counter is cleared at the falling edge of the $\overline{VD}$ signal.

The counter is incremented after the scanning of one line. (falling edge of the $\overline{HD}$ signal).

The counter is incremented even when a line with all blank data or a line with display on/off bit off occurs.

The counter is necessary to be read twice, because it does not synchronize CPU.

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

"0000" ··· No display line or end of display of line 16
"0001" ··· End of display of line 1
"0010" ··· End of display of line 2
"0011" ··· End of display of line 3
"0100" ··· End of display of line 4
"0101" ··· End of display of line 5
"0110" ··· End of display of line 6
"0111" ··· End of display of line 7
"1000" ··· End of display of line 8
"1001" ··· End of display of line 9
"1010" ··· End of display of line 10
"1011" ··· End of display of line 11
"1100" ··· End of display of line 12
"1101" ··· End of display of line 13
"1110" ··· End of display of line 14
"1111" ··· End of display of line 15

Interrupt generation line specification register (3 bits) ··· ISDC (bits 6 to 4 in ORIRC)

When the lower 3 bits in DCTR are set as follows:

"000" ··· Interrupt request generated at display end
"001" ··· Interrupt request generated at display end
"010" ··· Interrupt request generated at display end
"011" ··· Interrupt request generated at display end
"100" ··· Interrupt request generated at display end
"101" ··· Interrupt request generated at display end
"110" ··· Interrupt request generated at display end
"111" ··· Interrupt request generated at display end

(19) P6 port output select

This selects whether the contents of port P67 - P64 will be output or R, G, B, Y/BL signals of the OSD circuit will be output on pins P67 - P64.

P6 port output select registers (4 bits) ··· P67S to P64S (bits 7 to 4 in ORP6S)

"0" ··· R, G, B, Y/BL signal output

"1" ··· Port contents output

(20) OSD pin output polarity control

Output polarity control

Output polarity control registers (3 bits)

For BL ··· BLIV (bit 4 in ORIV)

For Y ··· YIV (bit 3 in ORIV)

For R, G, and B ··· RGBIV (bit 2 in ORIV)

Output polarity control

**IV

"0" ··· Active high

"1" ··· Active low

(21) OSD pin input polarity control

Input polarity control

Input polarity control register (2 bits)

For Y/BLIN ··· YBLII (bit 1 in ORIV)

For RIN, GIN, and BIN ··· RGBII (bit 0 in ORIV)

Input polarity control

**II

"0" ··· Active high

"1" ··· Active low

(22) Y/BL signal select

Y signal ··· Logical OR for R, G, B, charactor pattern, and fringing

BL signal ··· EXBL (bit 6 in ORBK)

When EXBL = 0 (no full-raster blanking) :

Output in all areas where characters can be displayed.

(except for character code 00_H: blank data)

When EXBL = 1 (full-raster blanking) :

Output in the whole page

Selects of 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

(23) R,G, B, Y/BL signal select

Selects either R, G, B, and Y/BL signals from the internal OSD circuit, or RIN, GIN, BIN, and Y/BLIN signals externally input.

R,G, B, Y/BL signal select registers (2 bits) ··· MPXS1/MPXS0 (bits 3 and 2 in ORETC)

"00" ··· Simultaneous output (Signal from the OSD circuit has higher priority.)
"01" ··· Output of signal from internal OSD circuit
"10" ··· Output of signal from externally input
"11" ··· Simultaneous output (Externally input signal has higher priority.)

(24) Display memory access

There are two types of access: write data to the display memory and read data from the display memory.
The display memory is accessed using the following registers: DMA7 to DMA0, CRA7 to CRA0, RDT, GDT, BDT, BLF, EUL, SLNT, MBK, MFYWR, RDWRV.

- Display memory read mode specification register (1 bit) ··· MFYWR (bit 0 in ORP6S)
  "0" ··· Normal mode
  "1" ··· Read-modify-write-mode

- Read/write mode select register at normal mode (MFYWR = 0)
  ··· RDWRV (bit 1 in ORP6S)
  "0" ··· Data write mode
  "1" ··· Data read mode

- Display memory bank switching register (1 bit) ··· MBK (bit 0 in ORETC)
  "0" ··· Access to either character code or character display options
  "1" ··· Access to both character code and character display options

Display memory auto increment depends on MBK setting.

Address increment table

		RD		WR	
		Color data	Character data	Color data	Character data
MFYWR = 0	MBK = 0	INC	INC	INC	INC
	MBK = 1	–	INC	–	INC
MFYWR = 1	MBK = 0	–	–	INC	INC
	MBK = 1	–	–	–	INC

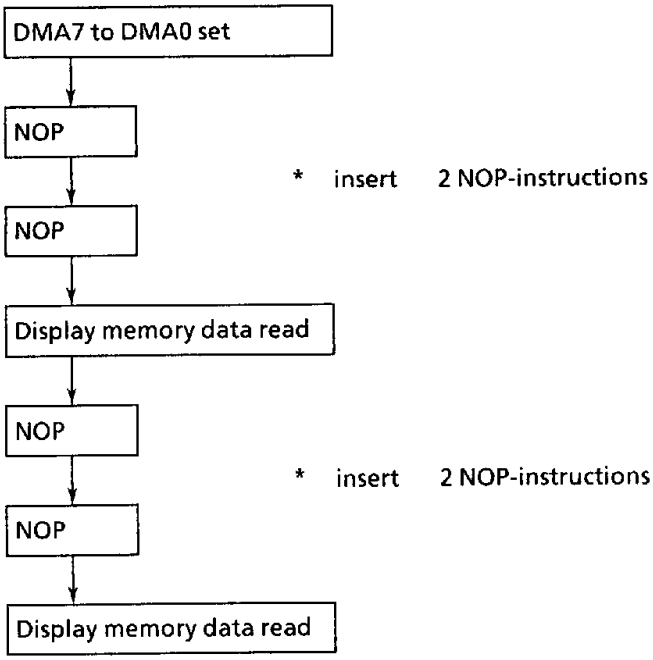
INC : Automatic address increment at read or write
– : No address change at data read or write

- Display memory address specification register (8 bits) ··· DMA7 to DMA0 (bits 7 to 0 in ORDMA)

- Display memory data access register
  ① For character code access (8 bits) ··· CRA 7 to CRA0 (bits 7 to 0 in ORCRA)
  ② For character design access (6 bits) ··· SLNT, EUL, BLF, RDT, GDT, and BDT (bits 5 to 0 in ORDSN)

There are two types of display memory access: normal mode and read-modify-write mode.

Caution 1 : Don't use the operation 「 LDW (HL), mn 」 when accessing display memory.
Caution 2 : When reading the display memory data immediately after setting the display memory address to DMA7 or DMA0, or when continuously reading the display memory data, insert 2 NOP-instructions in program as following to be stabilized the data.



Example 1 : A program before refining

```
LD    HL, 0F93H
LD    (HL+), W        ;  DMA7 to DMA0 set
LD    C, (HL)         ;  Display memory data read
```

Example 2 : A program after refining

```
LD    HL, 0F93H
LD    (HL+), W        ;  DMA7 to DMA0 set
NOP
NOP
LD    C, (HL)         ;  Display memory data read
```

1. Normal mode

In normal mode, display memory addresses are automatically incremented for every read or write. Since addresses are automatically incremented, this mode is used for simultaneously reading data from multiple consecutive addresses and for simultaneously writing data to multiple consecutive addresses.

- Display memory read sequence -

① Set MFYWR to 0. (Set to normal mode.)

② Set MBK to 0 or 1.

③ Set RDWRV to 1. (Set to data read mode.)

④ Set the display memory address to DMA7 to DMA0.

⑤ Read data from CRA7 to CRA0, SLNT, EUL, BLF, BDT, GDT, and RDT. (DMA7 to DMA0 are automatically incremented.)

⑥ For continuous read, repeat ④ and ⑤. (For data read from continuous addresses, repeat ⑤.)

- Display memory write sequence -

① Same as above.

② Same as above.

③ Set RDWRV to 0. (Set to data write mode.)

④ Same as above.

⑤ Write data to CRA7 to CRA0, SLNT, EUL, BLF, BDT, GDT, and RDT. (DMA7 to DMA0 are automatically incremented.)

⑥ Same as above.

2. Read-modify-write mode

In read-modify-write mode, display memory addresses are automatically incremented during a write; not incremented during a read. Thus, immediately after data is read from a display memory address, data can be written to the same address. After a write, the display memory address is automatically incremented.

- Read-modify-write sequence -

① Set MFYWR to 1.

② Set MBK to 0 or 1.

③ Set display memory address to DMA7 to DMA0.

④ Read data from CRA7 to CRA0, SLNT, EUL, BLF, BDT, GDT, and RDT. (DMA7 to DMA0 are not incremented.)

⑤ Write data to CRA7 to CRA0, SLNT, EUL, BLF, BDT, GDT, and RDT. (DMA7 to DMA0 are automatically incremented.)

⑥ For continuous read-modify-write, repeat ③, ④, and ⑤.
(For read-modify-write at consecutive addresses, repeat ④ and ⑤.

Note: In read-modify-write mode, read only or write only can be executed.

(25) OSD control register write/read

The addresses of the OSD control registers are assigned to the DBR register.
For writing data to or reading data from the OSD control registers, access the DBR register in the normal way.
The written data is transferred to the OSD circuit and become valid if RGWR register is set to "1".
However, while the display line is being scanned, the data written after the line is scanned is transferred to the OSD circuit and becomes valid. And, change value of OSD control register after RGWR is set to "0". The value is broken, if contents of OSD control register is modified before its data is transferred to OSD circuit.
The registers for writing data to display memory become valid, when its data is written. (DMA7 to DMA0, CRA7 to CRA0, RDT, GDT, BDT, BLF, EUL, SLNT, YBLCS, BKMF, ESMZ, VDSMD, MPX, MBK, P67S to P64S, RDWRV, and MFYWR)

- Written data transfer register (1 bit) ··· RGWR (bit 2 in ORDON)
  "0" ··· Initialized state
  "1" ··· Transfers written data to OSD circuit. (After transfer, RGWR is reset to 0.)

Coution : Don't write "0" to RGWR.

(26) Display on/off

Function used to display a line specified for on/off display.

Display on/off specified page by page

Display on/off specification register (1 bit) ··· DON (bit 0 in ORDON)

"0" ··· Display disable

"1" ··· Display enable

Caution : Don't start STOP mode during display enable.

(27) Window

Function used to set upper and lower limit of page. Window upper limit is specified by WVSH7 to WVSH0. Window lower limit is specified by WVSL7 to WVSL0. This function is specified by setting EWDW (bit 1 in ORDON) to "1".

Window upper limit specification register (8bit) ··· WVSH

Window lower limit specification register (8bit) ··· WVSL

Window upper and lower limit position

When VDSMD = 0 (normal mode) :

$WVSH = (WVSH7 \sim WVSH0)_H \times 1T_{HD}$

$WVSL = (WVSL7 \sim WVSL0)_H \times 1T_{HD}$

When VDSMD = 1 (double scan mode) :

$WVSH = (WVSH7 \sim WVSH0)_H \times 2T_{HD}$

$WVSL = (WVSL7 \sim WVSL0)_H \times 2T_{HD}$

T_{HD} : One cycle of $\overline{HD}$ signal

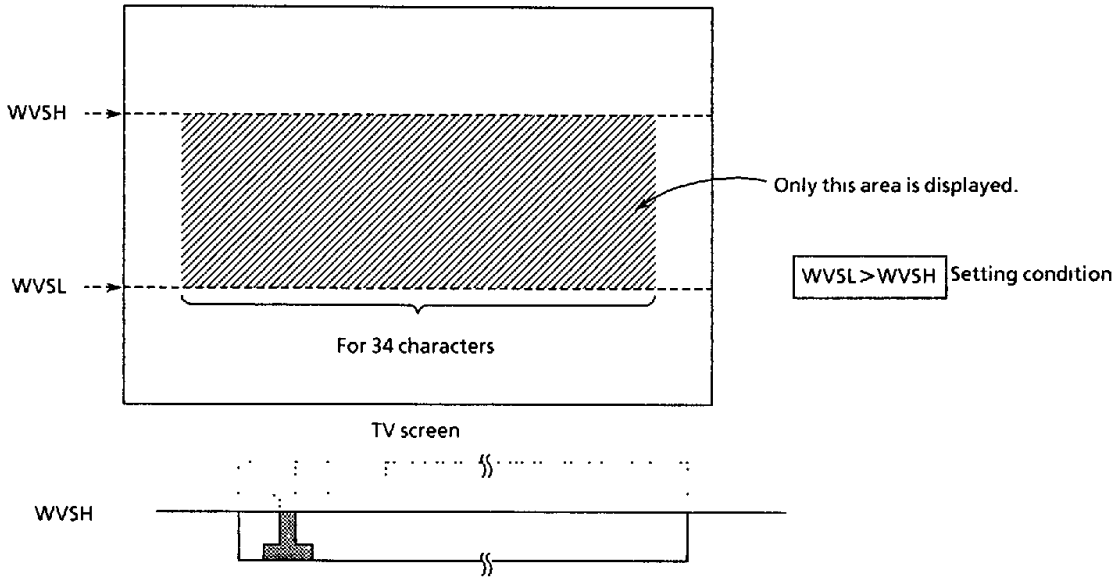
Coution : Modify value of window limit registers, while end of display all line through start of display first line or line or window upper limit.

Window enable flag (1bit) ··· EWDW (bit 1 in ORDON)

"0" ··· window specification on

"1" ··· window specification off

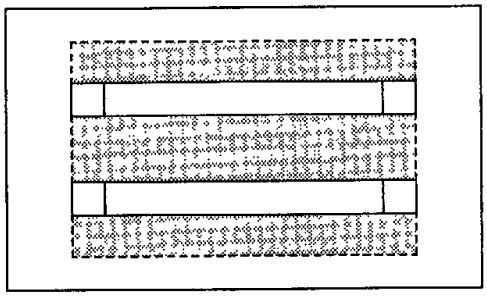
Coution : Write "0" to EWDW , after looking for rising edge of $\overline{HD}$ signal by software.



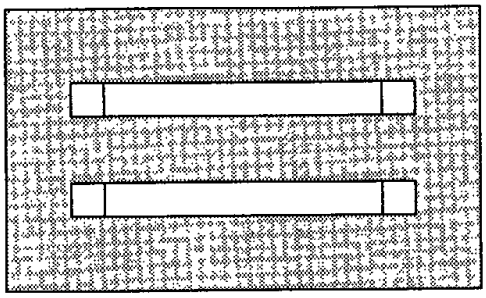
<Usage example>

The following can be displayed by combining the window and full-raster blanking functions.

EXBL	EWDW	Internal OSD BL output
0	0	Vertical direction : Character area is only displayed. Horizontal direction : Character area is only displayed.
0	1	Vertical direction : Window area is displayed. Horizontal direction : Character area is only displayed.
1	0	Vertical direction : Whole page is displayed. Horizontal direction : Whole page is displayed.
1	1	Vertical direction : Window area is displayed. Horizontal direction : Window area is only displayed.



TV screen image
(EXBL = 1, EWDW = 1)



TV screen image
(EXBL = 1, EWDW = 0)

(28) OSD control registers

ORHS1 ($0F80_H$)

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

HS16~10	Horizontal display start position	Write only

ORVS1 ($0F81_H$) **ORVS2** ($0F82_H$) **ORVS3** ($0F83_H$) **ORVS4** ($0F84_H$) **ORVS5** ($0F85_H$) **ORVS6** ($0F86_H$) **ORVS7** ($0F87_H$) **ORVS8** ($0F88_H$)

7	6	5	4	3	2	1	0	
VS17	VS16	VS15	VS14	VS13	VS12	VS11	VS10	(Initial value 0000 0000)
VS27	VS26	VS25	VS24	VS23	VS22	VS21	VS20	(Initial value 0000 0000)
VS37	VS36	VS35	VS34	VS33	VS32	VS31	VS30	(Initial value 0000 0000)
VS47	VS46	VS45	VS44	VS43	VS42	VS41	VS40	(Initial value 0000 0000)
VS57	VS56	VS55	VS54	VS53	VS52	VS51	VS50	(Initial value 0000 0000)
VS67	VS66	VS65	VS64	VS63	VS62	VS61	VS60	(Initial value 0000 0000)
VS77	VS76	VS75	VS74	VS73	VS72	VS71	VS70	(Initial value 0000 0000)
VS87	VS86	VS85	VS84	VS83	VS82	VS81	VS80	(Initial value 0000 0000)

VSn	Vertical display start position for line n	Write only

(n = 1~8)

ORCS4 ($0F89_H$)

7	6	5	4	3	2	1	0	
CS4		CS3		CS2		CS1		(Initial value 0000 0000)

ORCS8 ($0F8A_H$)

CS8		CS7	CS6	CS5	(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 = 1~8)

OREFR ($0F8B_H$)

7	6	5	4	3	2	1	0	
EFR8	EFR7	EFR6	EFR5	EFR4	EFR3	EFR2	EFR1	(Initial value 0000 0000)

EFRn	Fringing enable for line n	0: No fringing 1 : Fringing enable	Write only

(n = 1~8)

ORSOL4 ($0F8C_H$)

7	6	5	4	3	2	1	0	
SOL4		SOL3		SOL2		SOL1		(Initial value 0000 0000)

ORSOL8 ($0F8D_H$)

SOL8		SOL7	SOL6	SOL5	(Initial value 0000 0000)

SOLn	Solid space enable for line n	00: No solid space display 01 : Solid space display left for 32 columns 10 : Solid space display right for 32 columns 11 : Solid space display left and right for 32 columns	Write only

(n = 1~8)

ORBK ($0F8E_H$)

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

(Initial value 0000 0000)

EBKGD	Background function enable	0 : No background function 1 : Background function enable	Write only
EXBL	Full-raster blanking enable	0 : No Full-raster blanking 1 : 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	001 : Blue 010 : Green 011 : Cyan 100 : Red 101 : Magenta 110 : Yellow 111 : White	

ORETC ($0F8F_H$)

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

(Initial value 0000 0000)

YBLCS	Y/BL signal select	0 : Y signal output 1 : BL signal output	Write only
BKMF	Blinking master flag	0 : Blinking function disable 1 : Blinking function enable	
ESMZ	Smoothing enable	0 : No smoothing 1 : Smoothing enable	
VDSMD	Double scan mode select	0 : Normal mode 1 : Double scan mode	
MPXS	R, G, B, Y/BL signal select	00 : Simultaneous output (Signal from the OSD circuit has higher priority.) 01 : Output of signal from internal OSD circuit 10 : Output of signal from externally input 11 : Simultaneous output (Externally input signal has higher priority.)	
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	

ORIRC ($0F90_H$)

7	6	5	4	3	2	1	0
SVD	ISDC			DCTR			

(Initial value 0000 0000)

SVD	Interrupt source select	0: Interrpt request by ISDC value 1 : Interrupt request at falling edge of VD signal	Write only
ISDC	Interrupt generation line select		
DCTR	Display line counter		read only

ORP6S (0F91$_H$)

7	6	5	4	3	2	1	0
P67S	P66S	P65S	P64S			RDWRV	MFYWR

(Initial value 0000 **00)

P67S~P64S	P6 port output select	0 : R, G, B, Y/BL signal output 1 : Port contents output	Write only
RDWRV	Read/write mode select at normal mode	0 : Data write mode 1 : Data read mode	
MFYWR	Display memory read mode	0 : Normal mode 1 : Read-modify-write-mode	

ORIV (0F92$_H$)

7	6	5	4	3	2	1	0
FRSMS	FORS	FMS	BLIV	YIV	RGBIV	YBLII	RGBII

(Initial value 0000 0000)

FRSMS	Field decision mode select	0 : Field unit 1 : Frame unit	Write only
FORS	fosc frequency select	0 : Normal frequency mode 1 : Double frequency mode	
FMS	First/second field phase select	0 : First field has higher priority 1 : Second field has higher priority	
BLIV	BL output polarity select	0 : Active high 1 : Active low	
YIV	Y output polarity select	0 : Active high 1 : Active low	
RGBIV	R, G, B output polarity select	0 : Active high 1 : Active low	
YBLII	Y/BLIN input polarity select	0 : Active high 1 : Active low	
RGBII	RIN, GIN, BIN input polarity select	0 : Active high 1 : Active low	

ORDMA (0F93$_H$)

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

(Initial value 0000 0000)

DMA7~0	Display memory address	Write only

ORDSN (0F94$_H$)

7	6	5	4	3	2	1	0
		SLNT	EUL	BLF	RDT	GDT	BDT

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

SLNT	Slant enable	0 : No slant 1 : Slant	R/W
EUL	Underline enable	0 : No underline 1 : Underline	
BLF	Blinking enable	0 : No blinking 1 : Blinking	
RDT/ GDT/ BDT	Character color select	001 : Blue 010 : Green 011 : Cyan 100 : Red 101 : Magenta 110 : Yellow 111 : White	

ORCRA (0F95$_H$)

7	6	5	4	3	2	1	0
CRA7	CRA6	CRA5	CRA4	CRA3	CRA2	CRA1	CRA0

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

CRA7~0	Character code	R/W

ORDON ($0F96_H$)

7	6	5	4	3	2	1	0
				"1"	RGWR	EWDW	DON

(Initial value **** 0000)

RGWR	Written data transfer control	0 : (Initial state) 1 : Transfers written data to OSD circuit. (After transfer, RGWR is reset to 0.)	R/W
EWDW	Window enable	0 : Window specification off 1 : Window specification on	
DON	Display on/off select	0 : Display disable 1 : Display enable	

ORWVSH ($0F97_H$)

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

(Initial value 0000 0000)

WVSH7~0	Window upper limit position	Write only

ORWVSL ($0F98_H$)

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

(Initial value 0000 0000)

WVSL7~0	Window lower limit position	Write only

Note 1: *; don't care

Note 2: All OSD control registers cannot use the read-modify-write instructions (bit manipulation instructions such as SET, CLR, etc. and logical operation such as AND, OR, etc.)

Note 3: Write "0" to bit 1 of ORETC when writing to ORETC.

Note 4: Write "1" to bit 3 of ORDON when writing to ORDON.

Note 5: The registers of ORHS1, ORVS1 to ORVS8, ORCS4, ORCS8, OREFR, ORSOL4, ORSOL8, ORBK, ORIRC, ORIV, ORWVSH, and ORWVSL are changed by RGWR. Bits 2 to 1 in ORDON are also changed by RGWR.

2.14 Slicer interface (SIF)

16-bit serial interface used to receive serial data from the slicer.
Connected to the slicer via the P32 (CCLK) and P33 (CDATA) pins.
The slicer interface pin is also used as the P3 port. When the pin is used as the slicer interface pin, set port P3 to input mode.

2.14.1 Configuration

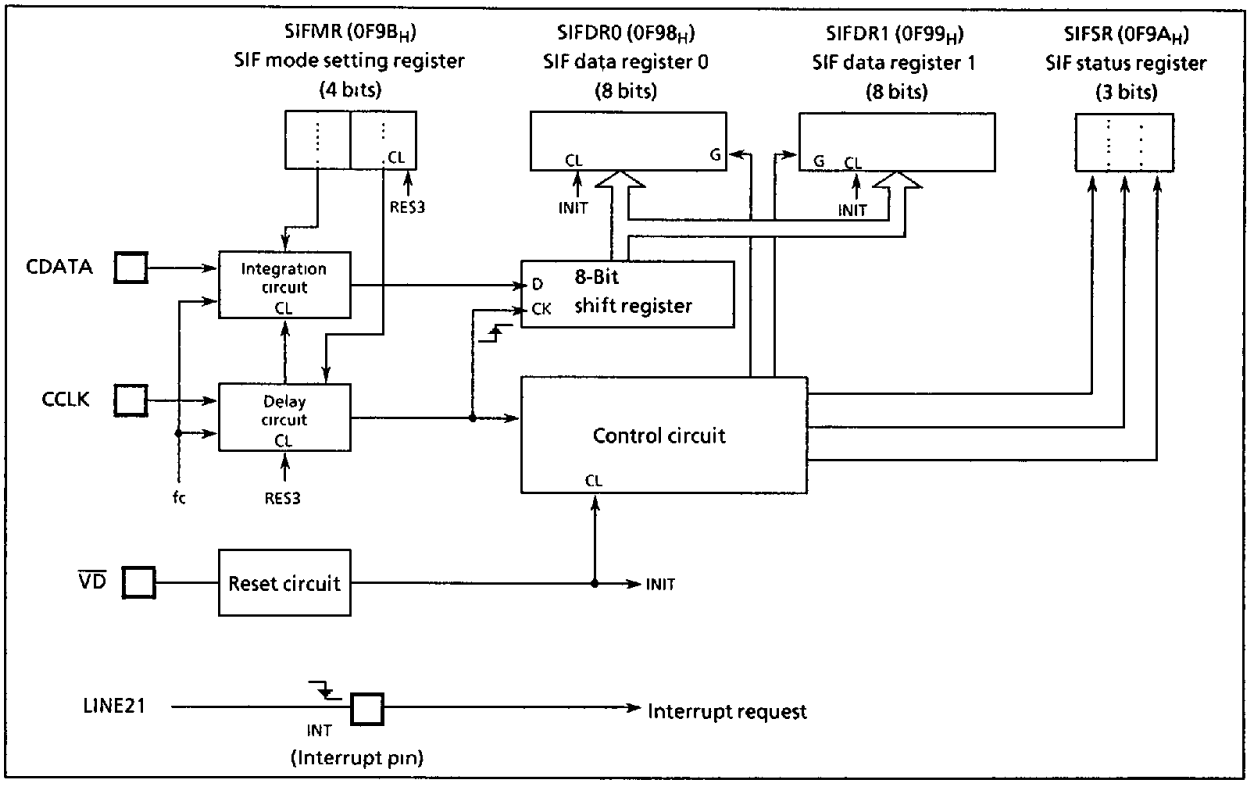


Figure 2-78. Slicer interface

2.14.2 Control

The SIF is controlled by SIFMR, SIFDR0, SIFDR1, and SIFSR.

Slicer interface mode setting register

SIFMR (0F9B$_H$)

7	6	5	4	3	2	1	0
				DTMS		SMDS	

(Initial value : **** 0000)

DTMS CCLK delay time select (fc = 8 MHz)
- "00" ··· No shift
- "01" ··· 2/fc shift
- "10" ··· 4/fc shift
- "11" ··· 6/fc shift

(Write only)

SMDS CDATA digital integration mode select
- "*0" ··· No integration.
- "01" ··· Integration by 3-bit up-down counter (mode 2)
- "11" ··· Integration by 2-bit up-down counter (mode 1)

(Write only)

Slicer interface register

SIFDR0 (0F98$_H$)

7	6	5	4	3	2	1	0
SIFD0							

(Initial value : 0000 0000)

SIFDR1 (0F99$_H$)

7	6	5	4	3	2	1	0
SIFD1							

(Initial value : 0000 0000)

SIFDR0 Receive data lower 8 bits (Read only)
SIFDR1 Receive data upper 8 bits (Read only)

Slicer interface status register

SIFSR (0F9A$_H$)

7	6	5	4	3	2	1	0
					STCK	STSB	STDE

(Initial value : **** *000)

STCK Field decision flag
- "1" ·· Period of time from the first CCLK rise to $\overline{VD}$ fall in the first field
- "0" · Other than the above

(Read only)

STSB Start bit decision flag
- "1" ··· Period of time from CCLK rise at start bit receive to $\overline{VD}$ fall in the first field
- "0" ··· Other than the above

(Read only)

STDE 16-bit data receive end decision flag
- "1" ··· Period of time from CLK rise when 16-bit data are received in the first field and its upper 8-bits of data are set in data register 1 (SIFDR1), to $\overline{VD}$ fall
- "0" ··· Other than the above

(Read only)

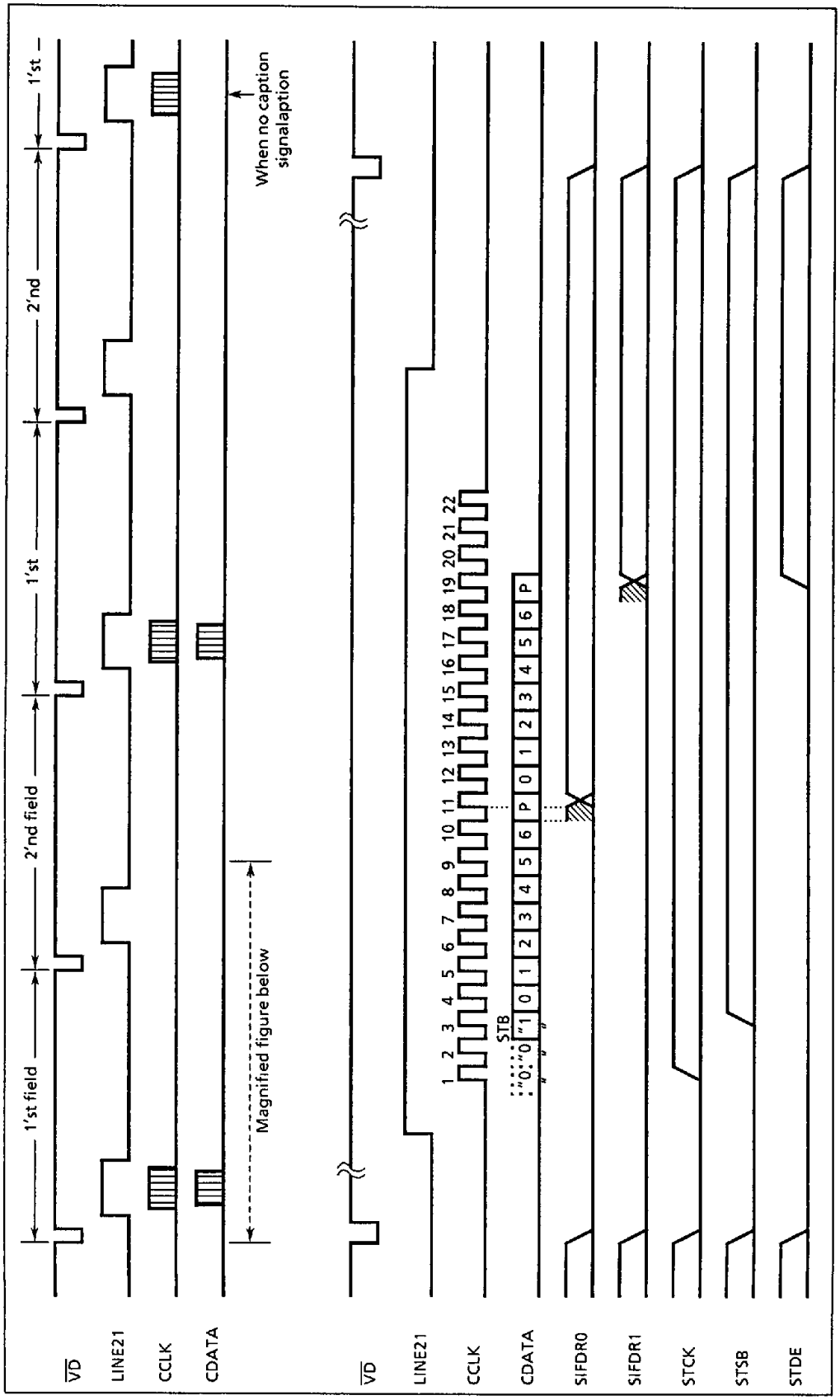
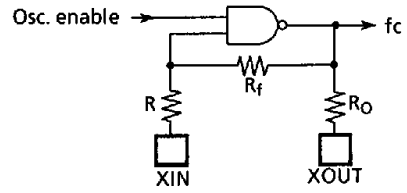
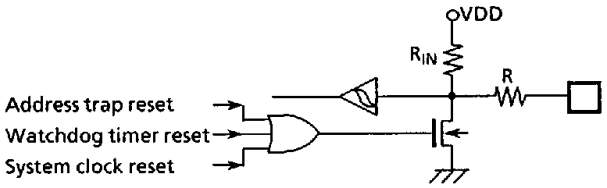
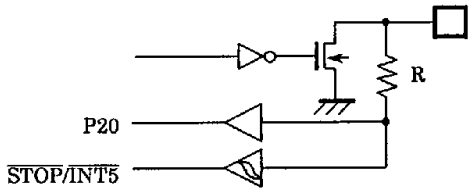
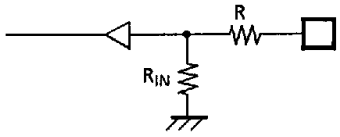
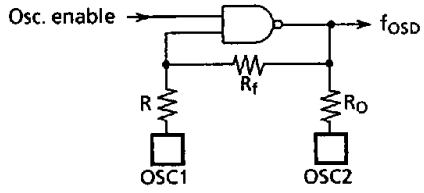


Figure 2-79. Slicer interface timing chart

INPUT/OUTPUT CIRCUITRY

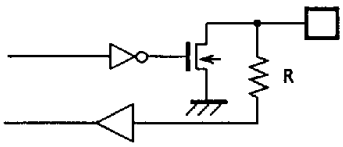
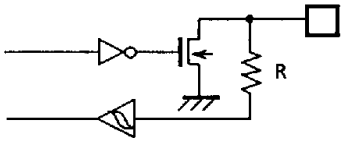
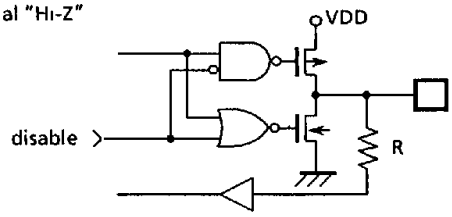
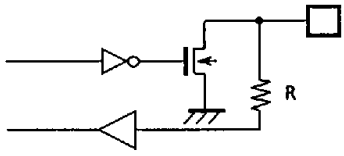
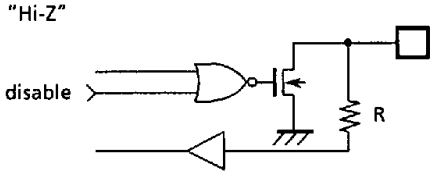
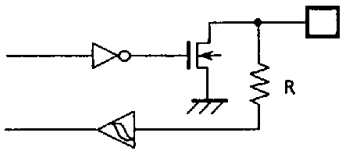
(1) Control pins

The input/output circuitries of the 87C833/C33/H33 control pins are shown below.

CONTROL PIN	I/O	INPUT/OUTPUT CIRCUITRY	REMARKS
XIN XOUT	Input Output		Resonator connecting pins (high-frequency) R_f = 1.2MΩ (typ.) R_O = 1.5kΩ (typ.) R = 1kΩ (typ.)
$\overline{\text{RESET}}$	I/O		Sink open drain output Hysteresis input Pull-up resistor R_{IN} = 220kΩ (typ.) R = 1kΩ (typ.)
$\overline{\text{STOP}}$/$\overline{\text{INT5}}$ (P20)	Input		Hysteresis input R = 1kΩ (typ.)
TEST	Input		Pull-down resistor R_{IN} = 70kΩ (typ) R = 1kΩ (typ.)
OSC1 OSC2	Input Output		Osc. connecting pin for on-screen display R_f = 1.2MΩ (typ.) R_O = 1.5kΩ (typ.) R = 1kΩ (typ.)

(2) Input/Output Ports

The input/output circuitries of the 87C833/C33/H33 I/O ports are shown below.

PORT	I/O	INPUT/OUTPUT CIRCUITRY	REMARKS
P20	I/O	initial "Hi-Z" 	Sink open drain output R = 1kΩ
P3	I/O	initial "Hi-Z" 	Sink open drain output Hysteresis input R = 1kΩ
P4 P64 ~ P67	I/O	initial "Hi-Z" 	Tri-state I/O R = 1kΩ (typ.)
P5	I/O	initial "Hi-Z" 	Sink open drain output R = 1kΩ
P60 ~ P63	I/O	initial "Hi-Z" 	Sink open drain output High current output I_{OL} = 20mA(typ.) R = 1kΩ
P70 P71	I/O	initial "Hi-Z" 	Sink open drain output Hysteresis input R = 1kΩ

ELECTRICAL CHARACTERISTICS

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

PARAMETER	SYMBOL	CONDITION	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 P2, P3, P4, P5, P64 to P67, P7	3.2	mA
	I_{OUT2}	P60 to P63	30	
Output Current (Total)	ΣI_{OUT1}	Ports P2, P3, P4, P5, P64 to P67, P7	120	mA
	ΣI_{OUT2}	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} = 0V, Topr = − 30 to 70°C)

PARAMETER	SYMBOL	PINS	CONDITIONS		Min.	Max.	UNIT
Supply Voltage	V_{DD}		fc = 8MHz	NORMAL mode	4.5	5.5	V
				IDLE mode			
			fc = 4.2MHz	NORMAL mode	2.7		
				IDLE 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_{IL3}		$V_{DD} < 4.5V$			$V_{DD} \times 0.10$	
Clock Frequency	fc	XIN, XOUT	V_{DD} = 4.5~5.5V		1	8.0	MHz
			V_{DD} = 2.7~5.5V			4.2	
	f_{OSD}	OSC1, OSC2			−	8.0	

Note : fc : The condition of power supply voltage is limited to NORMAL1, NORMAL2, IDLE1 and IDLE2 mode.

D.C. CHARACTERISTICS

(V_{SS} = 0V, T_{opr} = – 30 to 70°C)

PARAMETER	SYMBOL	PINS	CONDITIONS	Min	Typ.	Max	UNIT
Hysteresis Voltage	V_{HS}	Hysteresis inputs		–	0.9	–	V
Input Current	I_{IN1}	TEST	V_{DD} = 5.5V, V_{IN} = 5 5V / 0V	–	–	± 2	μA
	I_{IN2}	Open drain ports	V_{DD} = 5.5V, V_{IN} = 5.5V	–	–	2	
	I_{IN3}	Tri-state ports	V_{DD} = 5.5V, V_{IN} = 5 5V / 0V	–	–	± 2	
	I_{IN4}	$\overline{RESET}$, $\overline{STOP}$					
Input Resistance	R_{IN2}	$\overline{RESET}$		100	220	450	kΩ
Output Leakage Current	I_{LO}	Open drain ports and tri-state ports	V_{DD} = 5.5V, V_{OUT} = 5.5V	–	–	2	μA
Output High Voltage	V_{OH2}	Tri- state ports	V_{DD} = 4.5V, I_{OH} = – 0.7mA	4.1	–	–	V
Output Low Voltage	V_{OL}	Except XOUT	V_{DD} = 4.5V, I_{OL} = 1.6mA	–	–	0 4	V
Output Low Current	I_{OL3}	P60 to P63	V_{DD} = 4.5V, V_{OL} = 1.0V	–	20	–	mA
Supply Current in NORMAL mode	I_{DD}		V_{DD} = 5.5V fc = 8MHz V_{IN} = 5.3V/0 2V	–	10	18	mA
Supply Current in IDLE mode				–	4.5	10	mA
Supply Current in STOP mode			V_{DD} = 5.5V V_{IN} = 5 3V/0 2V	–	0.5	10	μA

Note 1 : Typical values show those at T_{opr} = 25°C, V_{DD} = 5V.

Note 2 : Input Current ; The current through pull-up or pull-down resistor is not included.

Note 3 : Typical current consumption during A/D conversion is 1.2mA.

A/D CONVERSION CHARACTERISTICS

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

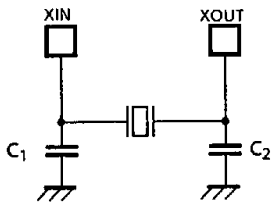
PARAMETER	SYMBOL	PINS	CONDITIONS	Min	Typ	Max.	UNIT
Analog Input Voltage Range	V_{AIN}	CIN3 to CIN0		V_{SS}	–	V_{DD}	V
Conversion Error			V_{DD} = 5 0V	–	–	± 1.5	LSB

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

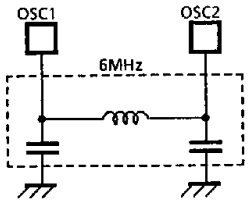
PARAMETER	SYMBOL	CONDITIONS	Min.	Typ.	Max.	UNIT
Machine Cycle Time	tcy	In NORMAL mode	0.5	–	10	μs
		In IDLE mode				
High Level Clock Pulse Width	t_{WCH}	For external clock operation (XIN input), fc = 8MHz	62.5	–	–	ns
Low Level Clock Pulse Width	t_{WCL}					

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

PARAMETER	OSCILLATOR	FREQUENCY	RECOMMENDED OSCILLATOR	RECOMMENDED CONDITIONS C_1	RECOMMENDED CONDITIONS C_2
High-frequency	Ceramic Resonator	8MHz	KYOCERA KBR8.0M	30pF	30pF
		4MHz	KYOCERA KBR4.0MS		
			MURATA CSA4.00MG		
	Crystal Oscillator	8MHz	TOYOCOM 210B 8.0000	20pF	20pF
		4MHz	TOYOCOM 204B 4 0000		
OSD	LC Resonator	6MHz	TOKO A285HCIS-13319	–	–
		12MHz	TOKO TA285HCIS-13306		



(1) High-frequency



(2) LC Resonator for OSD

Note : An electrical shield by metal shield plate on the surface of the IC package should be recommendable in order to prevent the device from the high electric fieldstress applied from CRT (Cathode Ray Tube) for continuous reliable operation.