CMOS 8-BIT MICROCONTROLLER

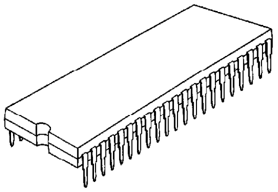
TMP87CC31N, TMP87CH31N

The 87CC31/H31 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 / counters, on-screen display, PWM, 6-bit A/D conversion inputs and remote control signal preprocessor on a chip.

PART No.	ROM	RAM	PACKAGE	OTP MCU
TMP87CC31N	12 K bytes	256 bytes	SDIP42-P-600-1.78	TMP87PM36N
TMP87CH31N	16 K bytes			

FEATURES

SDIP42-P-600-1.78

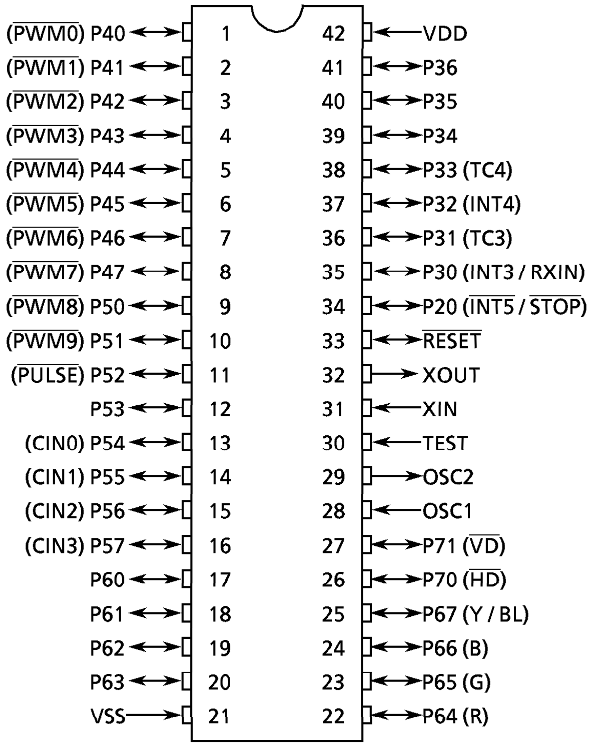


TMP87CC31N
TMP87CH31N

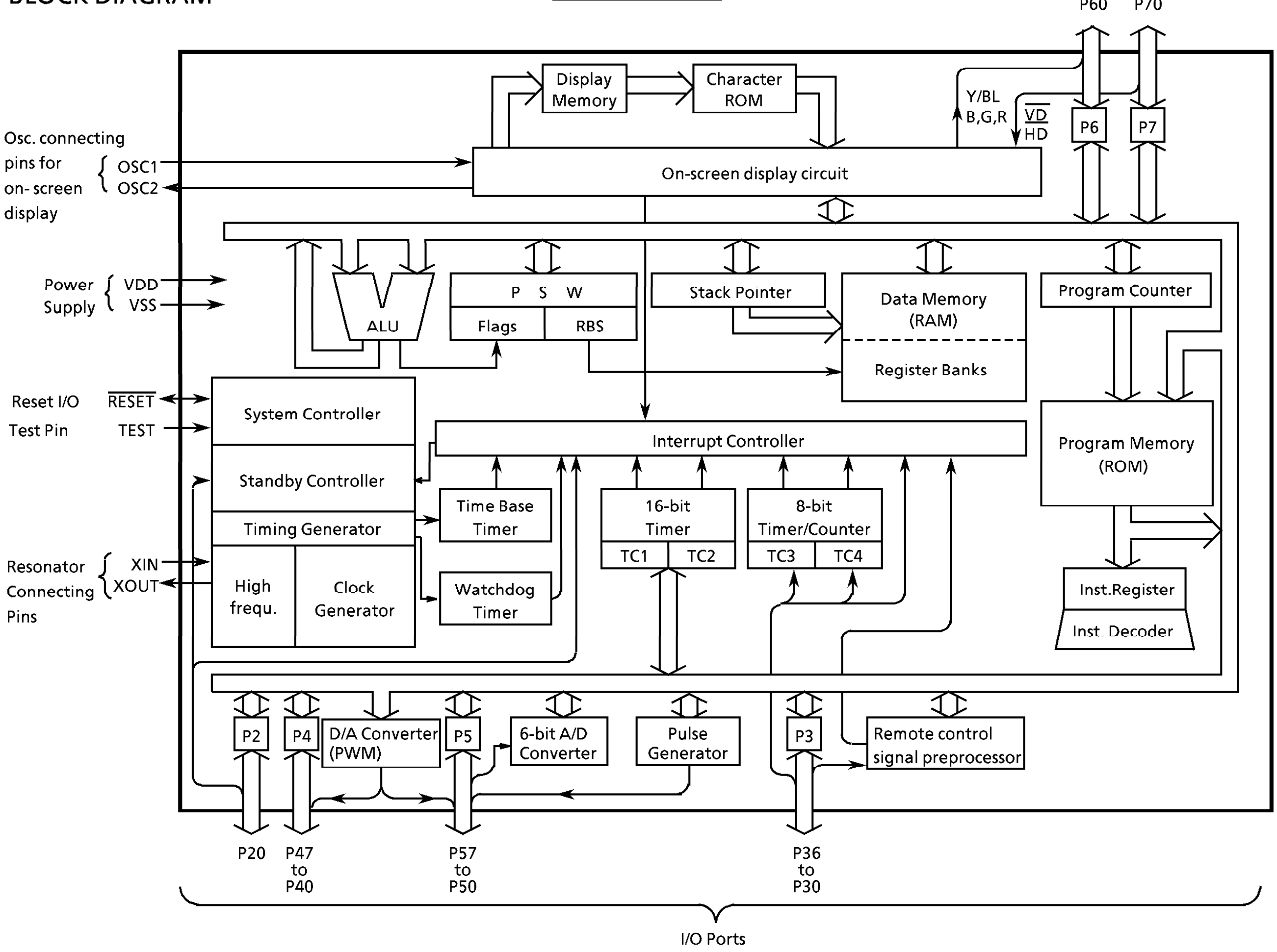
- ◆8-bit single chip microcomputer TLCS-870 Series
- ◆Instruction execution time : 0.5 μs (at 8 MHz)
- ◆412 basic instructions
  - ● Multiplication and Division (8 bits × 8 bits, 16 bits ÷ 8 bits)
  - ● Bit manipulations (Set / Clear / Complement / Move / Test / Exclusive Or)
  - ● 16-bit data operations
  - ● 1-byte jump / subroutine-call (Short relative jump / Vector call)
- ◆11 interrupt sources (External : 3, Internal : 8)
  - ● All sources have independent latches each, and nested interrupt control is available.
  - ● Edge-selectable external interrupts with noise reject
  - ● High-speed task switching by register bank changeover
- ◆6 Input/Output ports (34 pins)
  - ● High current output : 4 pins (typ. 20 mA)
- ◆Two 16-bit Timers
- ◆Two 8-bit Timer / Counters
  - ● Timer, Event counter, Capture (Pulse width / duty measurement) modes
- ◆Time Base Timer (Interrupt frequency : 1 Hz to 16384 Hz)
- ◆Watchdog Timer
  - ● Interrupt source / reset output (programmable)
- ◆On-screen display circuit
  - ● Character patterns : 96 characters
  - ● Characters displayed : 24 columns × 4 lines
  - ● Composition : 14 × 18 dots
  - ● Size of character : 3 kinds (line by line)
  - ● Color of character : 8 kinds (character by character)
  - ● Variable display position : Horizontal 128 steps, Vertical 256 steps
  - ● Fringing, Smoothing function
- ◆D/A conversion (Pulse Width Modulation) outputs
  - ● 14-bit resolution (1 channel)
  - ● 7-bit resolution (9 channels)
- ◆6-bit A/D conversion input (4 channels)
- ◆Pulse output (Clock for PLL IC)
- ◆Remote control signal preprocessor
- ◆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. Release by interrupts.
- ◆Operating voltage : 4.5 to 5.5 V at 8 MHz
- ◆Emulation Pod : BM87CM37N0A

PIN ASSIGNMENTS (TOP VIEW)

SDIP42-P-600-1.78



BLOCK DIAGRAM



PIN FUNCTION

PIN NAME	Input/Output	Function	
P20 ($\overline{\text{INT5}}$/$\overline{\text{STOP}}$)	I/O (Input)	1-bit input / output port with latch. When used as an input port, the latch must be set to "1".	External interrupt input 5 or STOP mode release signal input
P36	I/O (I/O)	7-bit input / output port with latch. When used as an input port, a timer / counter input, a remote control signal preprocessor input, or an external interrupt input, the latch must be set to "1".	
P35	I/O (I/O)		
P34	I/O (I/O)		
P33 (TC4)	I/O (Input)		Timer / Counter 4 input
P32 (INT4)			External interrupt input 4
P31 (TC3)			Timer / Counter 3 input
P30 (INT3/RXIN)	I/O (Input/Input)		External interrupt input 3 or remote control signal preprocessor input
P47 ($\overline{\text{PWM7}}$) to P41 ($\overline{\text{PWM1}}$)	I/O (Output)	8-bit programmable input / output port (tri-state). Each bit of this port can be individually configured as an input or an output under software control. During reset, all bits are configured as inputs. When used as a PWM output, the latch must be set to "1".	7-bit D/A conversion (PWM) outputs
P40 ($\overline{\text{PWM0}}$)			14-bit D/A conversion (PWM) output
P57 (CIN3) to P54 (CIN0)	I/O (Input)	8-bit input / output port with latch. When used as an input port, a comparator input, a PWM output, or a pulse output, the latch must be set to "1".	6-bit A/D conversion (Comparator) inputs
P53	I/O		
P52 ($\overline{\text{PULSE}}$)	I/O (Output)		Pulse output (Clock for PLL IC)
P51 ($\overline{\text{PWM9}}$)			7-bit D/A conversion (PWM) outputs
P50 ($\overline{\text{PWM8}}$)			
P67 (Y/BL)	I/O (Output)	8-bit programmable input / output port (P67 to P64 : tri-state, P63 to P60 : High current output). Each bit of this port can be individually configured as an input or an output under software control. During reset, all bits are configured as inputs. When used as the R, G, B, Y / BL outputs of on-screen display circuit, each bit of the P6 port data selection register (bits 7 to 4 in address $0F91_H$) must be set to "1".	Focus signal output or Background blanking control signal output
P66 (B)			RGB output
P65 (G)			
P64 (R)			
P63	I/O		High current output.
P62			
P61			
P60			
P71 ($\overline{\text{VD}}$)	I/O (Input)	2-bit input / output port with latch. When used as an input ports, or a vertical synchronous signal input and horizontal synchronous signal input, the latch must be set to "1".	Vertical synchronous signal input
P70 ($\overline{\text{HD}}$)			Horizontal synchronous signal input
OSC1, OSC2	Input, Output	Resonator connecting pins for on-screen display circuitry.	
XIN, XOUT		Resonator connecting pins. For inputting external clock, XIN is used and XOUT is opened.	
$\overline{\text{RESET}}$	I/O	Reset signal input or watchdog timer output / address-trap- reset output/system-clock-reset output.	
TEST	Input	Test pin for out-going test. Be tied to low.	
VDD, VSS	Power Supply	+ 5 V, 0 V (GND)	

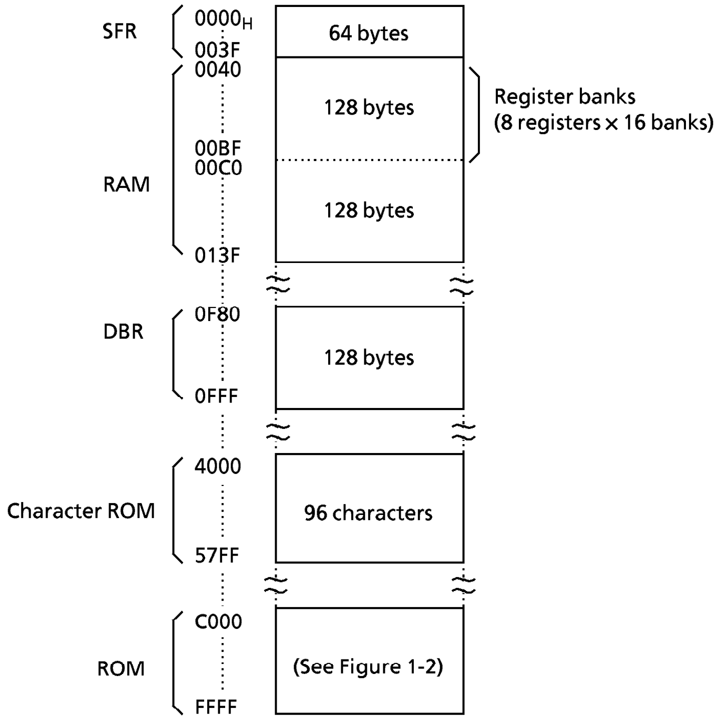
OPERATIONAL DESCRIPTION

1. CPU CORE FUNCTIONS

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

1.1 Memory Address Map

The TLCS-870 Series is capable of addressing 64 K bytes of memory. Figure 1-1 shows the memory address maps of the 87CC31/H31. In the TLCS-870 Series, the memory is organized 4 address spaces (ROM, RAM, SFR, and DBR). It uses a memory mapped I/O system, and all I/O registers are mapped in the SFR / DBR address spaces. There are 16 banks of general-purpose registers. The register banks are also assigned to the first 128 bytes of the RAM address space.



Note : ROM ; Read Only Memory includes:
Program memory
RAM ; Random Access Memory includes:
Data memory
Stack
General-purpose register banks
SFR ; Special Function Register includes:
I/O ports
Peripheral control registers
Peripheral status registers
System control registers
Interrupt control registers
Program Status Word
DBR ; Data Buffer Register includes:
On-Screen-Display (OSD) control registers
Remote-Control signal preprocessor control / status registers

Figure 1-1. Memory Address Map

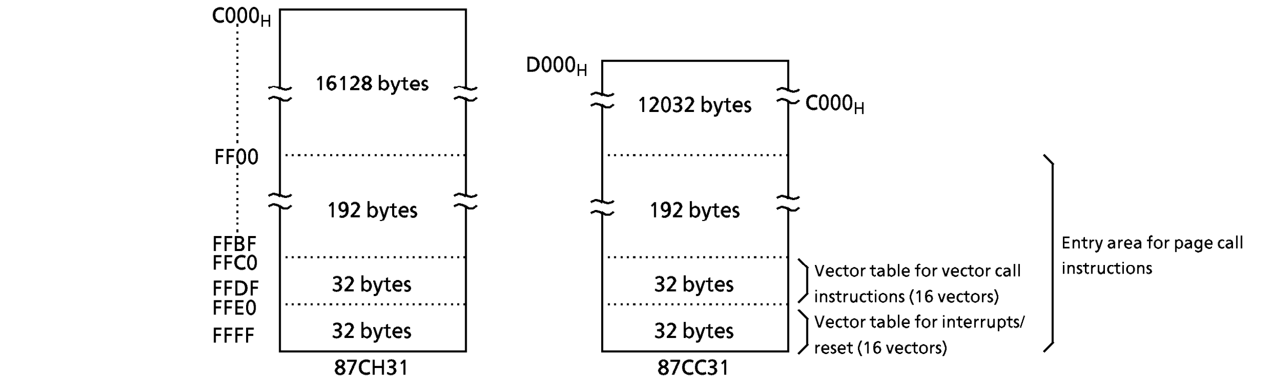

$C000_H$
16128 bytes
FF00
192 bytes
FFBF
FFC0
32 bytes
FFDF
FFE0
32 bytes
FFFF
87CH31
$D000_H$
12032 bytes
$C000_H$
192 bytes
32 bytes
32 bytes
87CC31
Vector table for vector call instructions (16 vectors)
Vector table for interrupts/ reset (16 vectors)
Entry area for page call instructions

Figure 1-2. ROM Address Maps

1.2 Program Memory (ROM)

The 87CC31 has a 12 Kbytes (addresses $D000_H$ to $FFFF_H$) and the 87CH31 has a 16 Kbytes (addresses $C000_H$ to $FFFF_H$) of program memory (mask programmed ROM). Addresses $FF00_H$ to $FFFF_H$ in the program memory can also be used for special purposes. Figure 1-2 shows the ROM address maps of the 87CC31/H31.

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

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

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

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

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

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

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

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

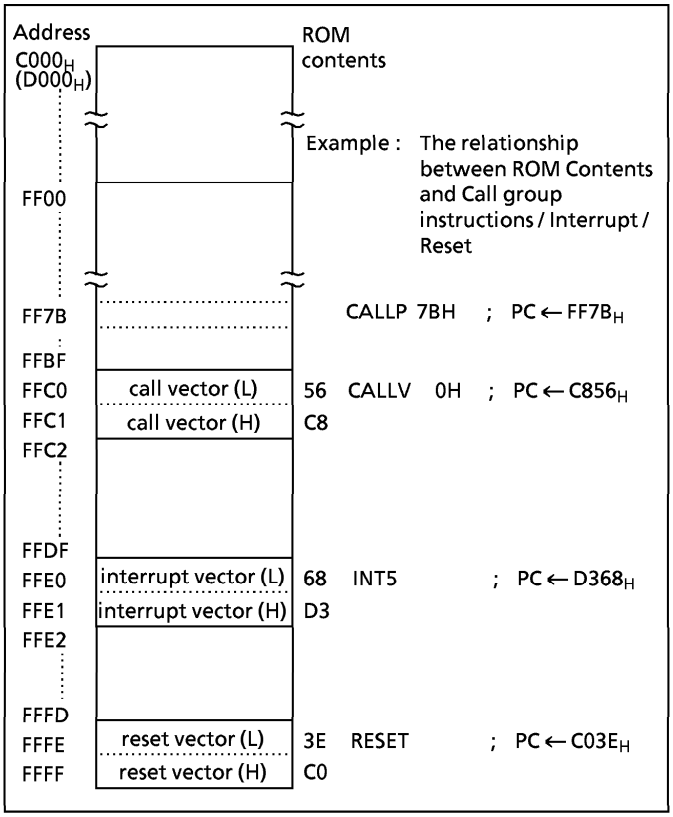


Figure 1-3. Program Memory Map

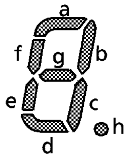
In the TLCS-870 Series, the same instruction used to access the data memory (e.g. [LD A, (HL)]) is also used to read out fixed data (ROM data) stored in the program memory. The register-offset PC-relative addressing (PC + A) instructions can also be used, and the code conversion, table look-up and n-way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

Example 1 : Loads the ROM contents at the address specified by the HL register pair contents into the accumulator (HL≧ $C000_H$ for 87CC31/H31):

```
LD  A, (HL) ; A←ROM (HL)
```

Example 2 : Converts BCD to 7-segment code (common anode LED). When A = 05_H, 92_H is output to port P5 after executing the following program:

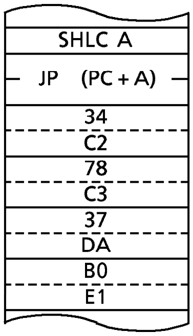
```
        ADD  A,  TABLE-$-4 ; P5 ←ROM (TABLE + A)
        LD   (P5), (PC+A)
        JRS  T,  SNEXT
TABLE : DB   0C0H,  0F9H,  0A4H,  0B0H,  99H,  92H,  82H,
             0D8H,  80H,  98H
SNEXT :
```



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

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

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

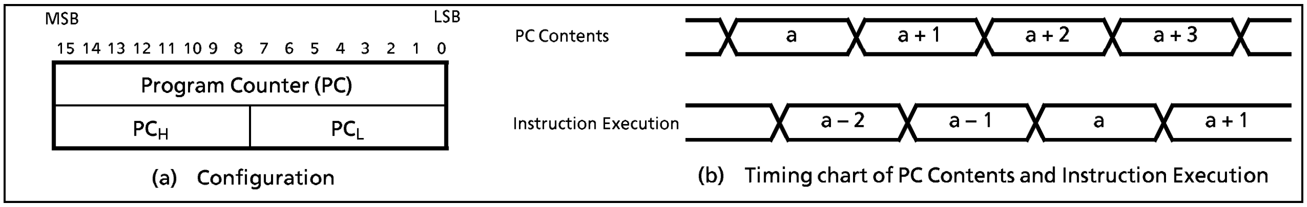


Note : DW is a word data definition instruction.

1.3 Program Counter (PC)

The program counter (PC) is a 16-bit register which indicates the program memory address where the instruction to be executed next is stored. After reset, the user defined reset vector stored in the vector table (addresses $FFFF_H$ and $FFFE_H$) is loaded into the PC ; therefore, program execution is possible from any desired address. For example, when $C0_H$ 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 $C125_H$.



(a) Configuration

(b) Timing chart of PC Contents and Instruction Execution

Figure 1-4. Program Counter

1.4 Data Memory (RAM)

The 87CC31/H31 have 256 bytes (addresses 0040_H-$013F_H$) of data memory (static RAM). Figure 1-5 shows the data memory map.

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

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

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

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

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

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

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

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

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

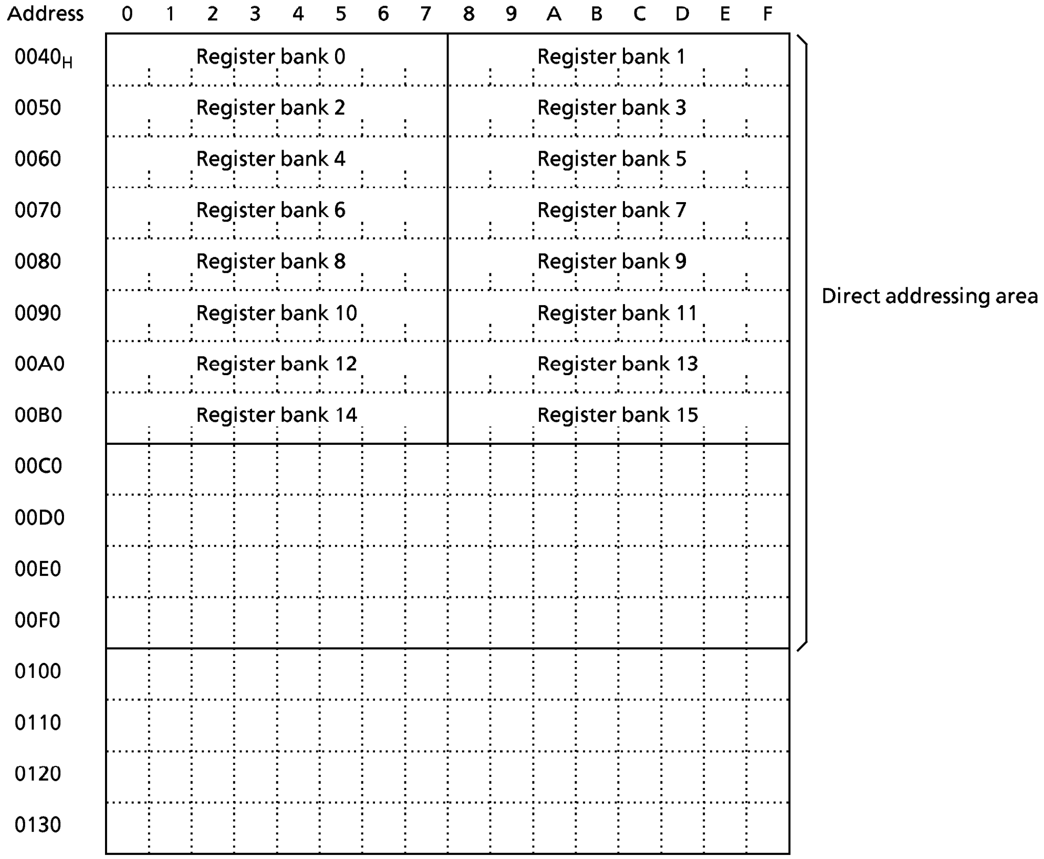
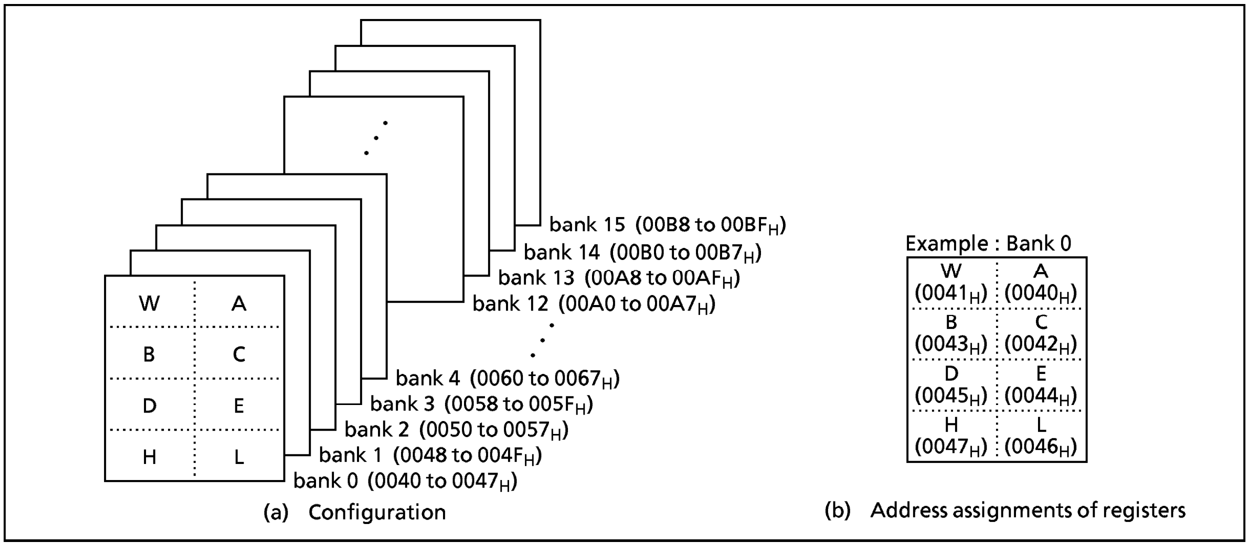


Figure 1-5. Data Memory Map

1.5 General-purpose Register Banks

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


bank 15 (00B8 to $00BF_H$)
bank 14 (00B0 to $00B7_H$)
bank 13 (00A8 to $00AF_H$)
bank 12 (00A0 to $00A7_H$)
bank 4 (0060 to 0067_H)
bank 3 (0058 to $005F_H$)
bank 2 (0050 to 0057_H)
bank 1 (0048 to $004F_H$)
bank 0 (0040 to 0047_H)
W
A
B
C
D
E
H
L

(a) Configuration

Example : Bank 0

W (0041_H)	A (0040_H)
B (0043_H)	C (0042_H)
D (0045_H)	E (0044_H)
H (0047_H)	L (0046_H)

(b) Address assignments of registers

Figure 1-6. General-purpose Register Banks

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

(1) **A, WA**

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

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

(2) **HL, DE**

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

Example 1 :

① LD A, (HL) ; Loads the memory contents at the address specified by HL into A.

② LD A, (HL + 52H) ; Loads the memory contents at the address specified by the value obtained by adding 52$_H$ to HL contents into A.

③ LD A, (HL + C) ; Loads the memory contents at the address specified by the value obtained by adding the register C contents to HL contents into A.

④ LD A, (HL +) ; Loads the memory contents at the address specified by HL into A. Then increments HL.

⑤ LD A, (− HL) ; Decrements HL. Then loads the memory contents at the address specified by new HL into A.

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

Example 2 : Block transfer

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

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

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

Example 1 : Repeat processing

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

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

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

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

Example 1 : Incrementing the RBS

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

Example 2 : Reading the RBS

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

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

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

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

1.6 Program Status Word (PSW)

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

The RBS can be read and written using the memory access instruction (e. g. [LD A, (003FH)], [LD (003FH), A], however the flags can only be read. When writing to the PSW, the change specified by the instruction

is made without writing data to the flags. For example, when the instruction [LD (003FH), 05H] is executed, "5" is written to the RBS and the JF is set to "1", but the other flags are not affected.

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

1.6.1 Register Bank Selector (RBS)

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

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

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

1.6.2 Flags

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

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

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

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

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

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

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

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

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

Example1 : Bit manipulation

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

Example2 : Arithmetic right shift

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

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

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

Example : BCD operation

(The A becomes 47_H after executing the following program when A = 19_H, B = 28_H)

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

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

Zero or carry information is set to the JF after operation (e. g. INC, ADD, CMP, TEST).

The JF provides the jump condition for conditional jump instructions [JRS T/F, $ + 2 + d], [JR T/F, $ + 2 + d] (T or F is a condition code). Jump is performed if the JF is “1” for a true condition (T), or the JF is “0” for a false condition (F).

The JF is set to “1” after executing the load / exchange / swap / nibble rotate/jump instruction, so that [JRS T, $ + 2 + d] and [JR T, $ + 2 + d] can be regarded as an unconditional jump instruction.

Example : Jump status flag and conditional jump instruction

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

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

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

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

1.7 Stack and Stack Pointer

1.7.1 Stack

The stack provides the area in which the return address or status, etc. are saved before a jump is performed to the processing routine during the execution of a subroutine call instruction or the acceptance of an interrupt. On a subroutine call instruction [CALL a] / [CALLP n] / [CALLV n], the contents of the PC (the return address) is saved; on an interrupt acceptance, the contents of the PC and the PSW are saved (the PSW is pushed first, followed by PC_H and PC_L). Therefore, a subroutine call occupies two bytes on the stack; an interrupt occupies three bytes.

When returning from the processing routine, executing a subroutine return instruction [RET] restores the contents to the PC from the stack; executing an interrupt return instruction [RETI] / [RETN] restores the contents to the PC and the PSW (the PC_L is popped first, followed by PC_H and PSW).

The stack can be located anywhere within the data memory space except the register bank area, therefore the stack depth is limited only by the free data memory size.

1.7.2 Stack Pointer (SP)

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

The SP is post-decremented when a subroutine call or a push instruction is executed, or when an interrupt is accepted; and the SP is pre-incremented when a return or a pop instruction is executed. Figure 1-9 shows the stacking order.

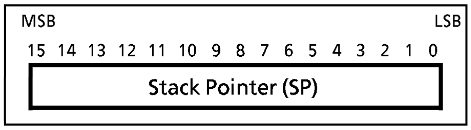


Figure 1-8. Stack Pointer

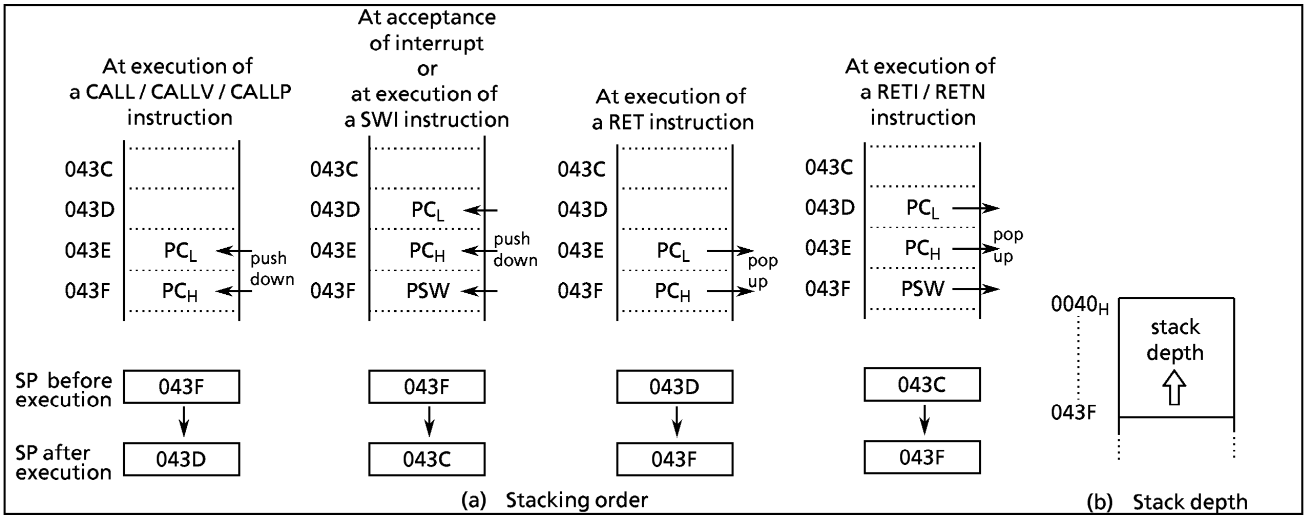
The SP is not initialized hardware-wise but requires initialization by an initialize routine (sets the highest stack address). [LD SP, mn], [LD SP, gg] and [LD gg, SP] are the SP access instructions (mn ; 16-bit immediate data, gg ; register pair).

Example 1 : To initialize the SP

```
LD  SP,  043FH  ;  SP←043FH
```

Example 2 : To read the SP

```
LD  HL,  SP     ;  HL←SP
```


At execution of a CALL / CALLV / CALLP instruction
At acceptance of interrupt or at execution of a SWI instruction
At execution of a RET instruction
At execution of a RETI / RETN instruction
043C
043D
043E
043F
PCL
PCH
PSW
push down
pop up
SP before execution
SP after execution
043F
043D
043C
043F
043D
043F
043C
043F
0040H
043F
stack depth

(a) Stacking order

(b) Stack depth

Figure 1-9. Stack

1.8 System Clock Controller

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

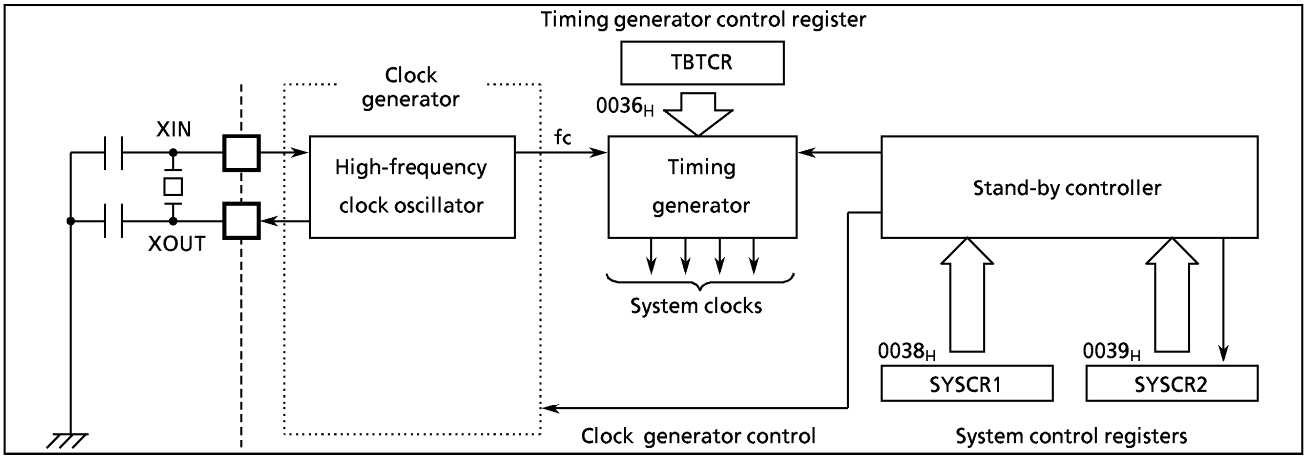


Figure 1-10. System Clock Controller

1.8.1 Clock Generator

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

The high-frequency (fc) clock can be easily obtained by connecting a resonator between the XIN / XOUT pins, respectively. Clock input from an external oscillator is also possible. In this case, external clock is applied to the XIN pin with the XOUT pin not connected. The 87CC31/H31 is not provided an RC oscillation.

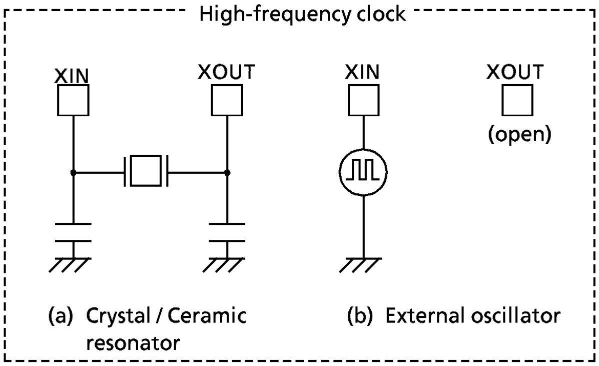


Figure 1-11. Examples of Resonator Connection

Note : Accurate Adjustment of the Oscillation Frequency:
Although hardware to externally and directly monitor the basic clock pulse is not provided, the oscillation frequency can be adjusted by providing a program to output fixed frequency pulses to the port while disabling all interrupts and monitoring this pulse. With a system requiring adjustment of the oscillation frequency, the adjusting program must be created beforehand.

1.8.2 Timing Generator

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

① Generation of main system clock
② Generation of 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 1-12 as follows. During reset and upon releasing STOP mode, the divider is cleared to "0", however, the prescaler is not cleared.

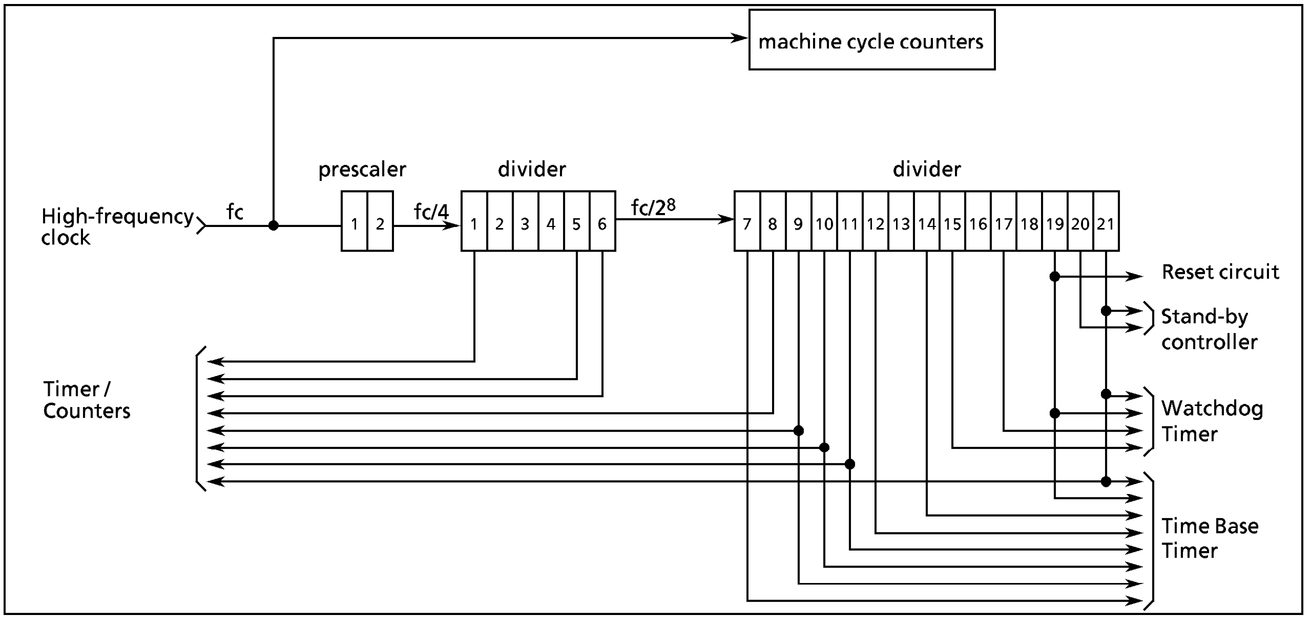


Figure 1-12. Configuration of Timing Generator

(2) Machine Cycle

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

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

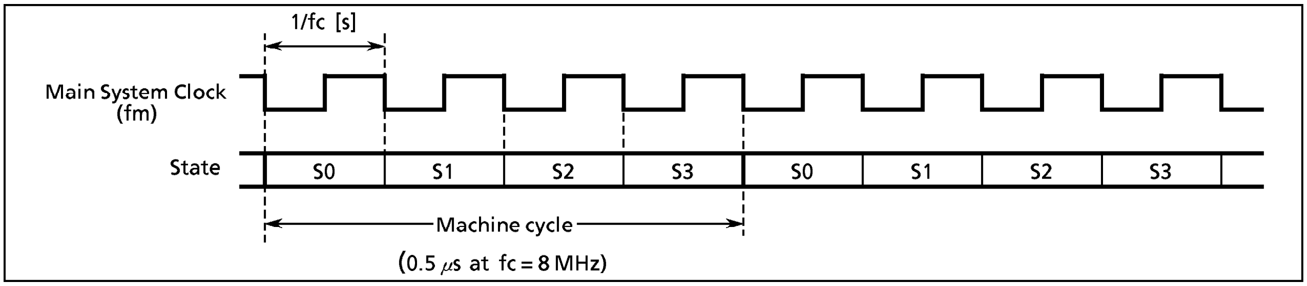


Figure 1-13. Machine Cycle

1.8.3 Stand-by Controller

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

Figure 1-14 shows the operating mode transition diagram and Figure 1-15 shows the system control registers.

(1) **Operating mode**

① NORMAL mode

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

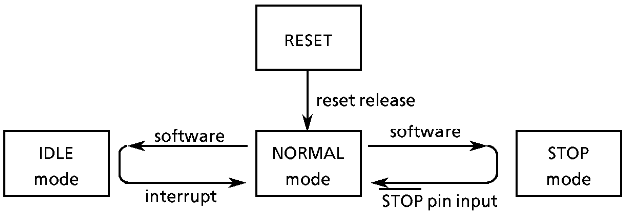
② IDLE mode

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

Figure 1-14. Operating Mode Transition Diagram

System Control Register 1

SYSCR1 (0038$_H$)

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

(Initial value: 0000 00**)

STOP	STOP mode start	0 : CPU core and peripherals remain active 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 87CC31/H31 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 : 87CC31/H31 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 bit 7 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 to 0 in SYSCR2 are always read in as "1" when a read instruction is executed.

Figure 1-15. 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 $\overline{\text{STOP}}$ pin input. The $\overline{\text{STOP}}$ pin is also used both as a port P20 and an $\overline{\text{INT5}}$ (external interrupt input 5) pin. STOP mode is started by setting STOP (bit 7 in SYSCR1) to "1". During STOP mode, the following status is maintained.

① 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)

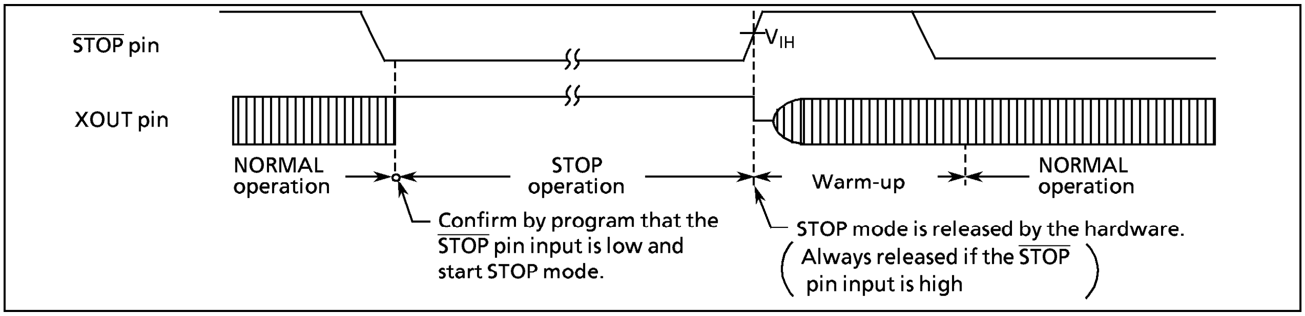
In this mode, STOP mode is released by setting the $\overline{\text{STOP}}$ pin high. This mode is used for capacitor back-up when the main power supply is cut off and for long term battery back-up.

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

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

Example : Starting STOP mode with an INT5 interrupt. (TMP87CC31/H31)

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

        SET   (SYSCR1) . 7            ;  Starts STOP mode
        LDW   (IL), 1110011111111111B ;  IL12, 11 ← 0
                                         (clears interrupt latches)
SINT5 : RETI
```



$\overline{\text{STOP}}$ pin
XOUT pin
V_{IH}
NORMAL operation
STOP operation
Warm-up
NORMAL operation
Confirm by program that the $\overline{\text{STOP}}$ pin input is low and start STOP mode.
STOP mode is released by the hardware.
(Always released if the $\overline{\text{STOP}}$ pin input is high)

Figure 1-16. Level-sensitive Release Mode

Note1 : After warming up is started, when STOP pin input is changed "L" level, $\overline{\text{STOP}}$ mode is not placed.

Note2 : When changing to the level-sensitive release mode from the edge-sensitive release mode, the release mode is not switched until a rising edge of the $\overline{\text{STOP}}$ pin input is detected.

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

In this mode, STOP mode is released by a rising edge of the $\overline{\text{STOP}}$ 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 $\overline{\text{STOP}}$ pin.

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

Example : Starting STOP mode operation in the edge-sensitive release mode (TMP87CM36)

```
LD  (SYSCR1),  00000000B       ; OUTEN ← 0 (specifies high-impedance)
DI                             ; IMF ← 0 (disables interrupt service)
SET (SYSCR1). STOP             ; STOP ← 1 (activates stop mode)
LDW (IL), 1110011111111111B    ; IL12, 11 ← 0
                                 (clears interrupt latches)
EI                             ; IMF ← 1 (enables interrupt service)
```

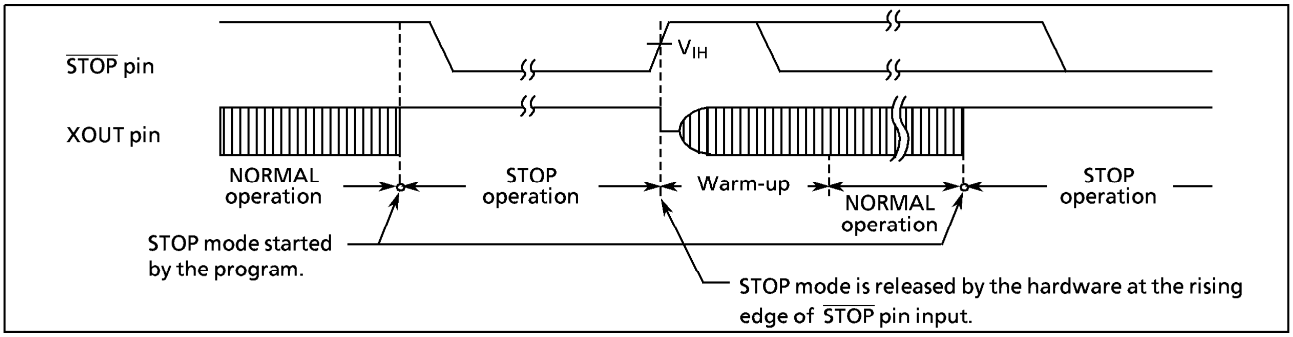

$\overline{\text{STOP}}$ pin
XOUT pin
V_{IH}
NORMAL operation
STOP operation
Warm-up
NORMAL operation
STOP operation
STOP mode started by the program.
STOP mode is released by the hardware at the rising edge of $\overline{\text{STOP}}$ pin input.

Figure 1-17. 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.

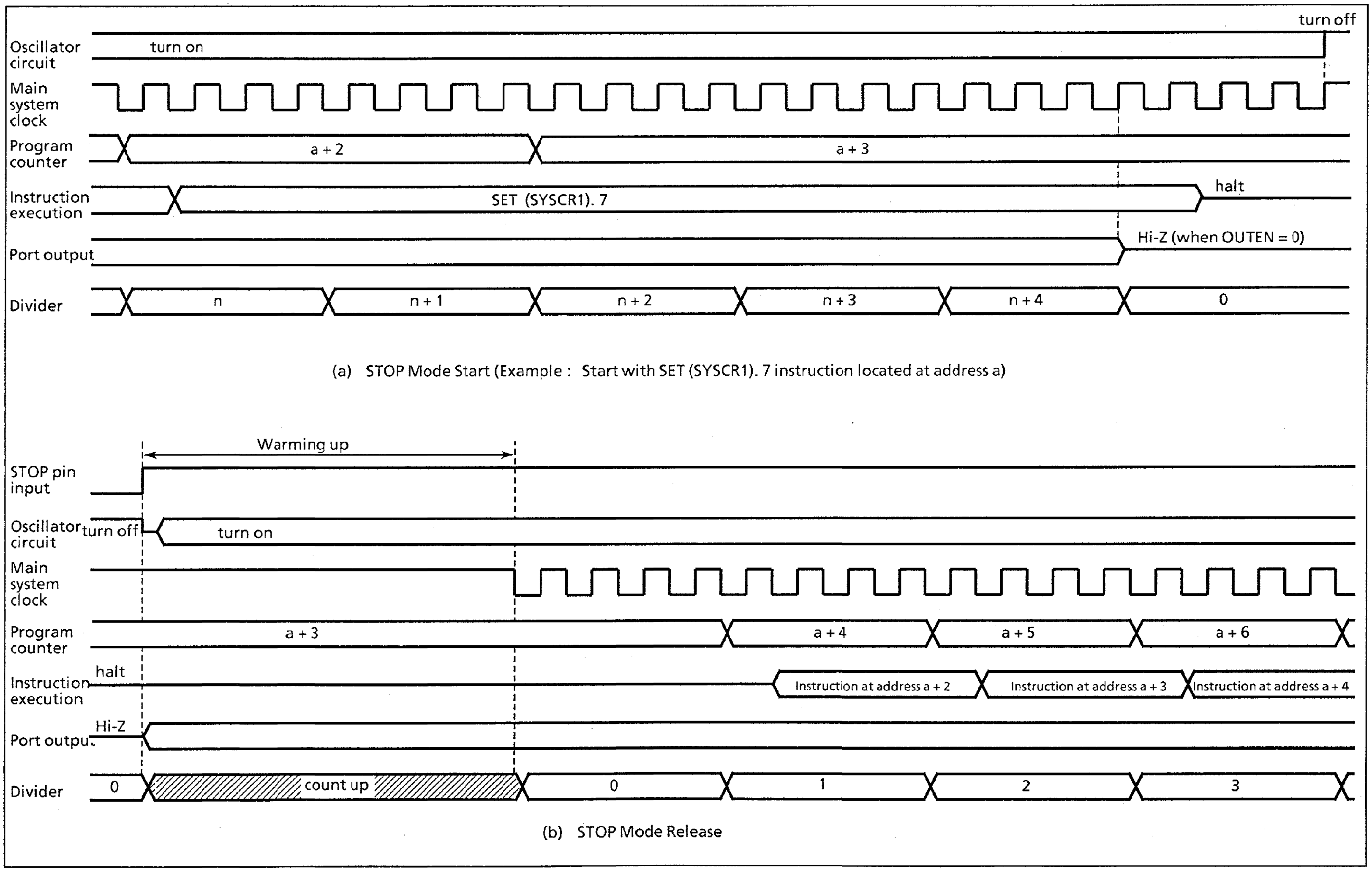
③ When the warming-up time has elapsed, normal operation resumes with the instruction following the STOP mode start instruction (e.g. [SET (SYSCR1). 7]). The start is made after the divider of the timing generator is cleared to "0".

Table 1-1. Warming-up Time example

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

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 $\overline{\text{RESET}}$ pin low, which immediately performs the normal reset operation.



(a) STOP Mode Start (Example : Start with SET (SYSCR1). 7 instruction located at address a)

(b) STOP Mode Release

Figure 1-18. STOP Mode Start / Release

Note : When STOP mode is released with a low hold voltage, the following cautions must be observed.
The power supply voltage must be at the operating voltage level before releasing the STOP mode. The 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 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 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.

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 IDLE mode must be cleared by load instructions.

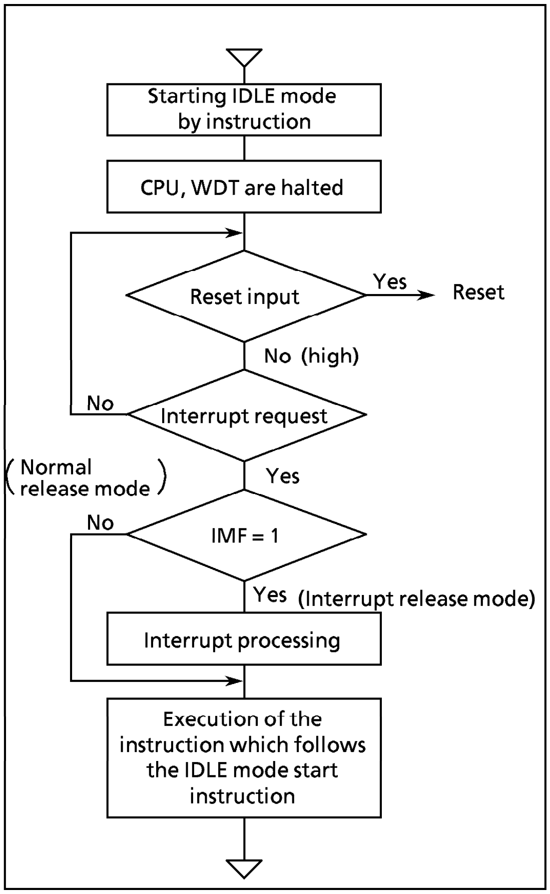
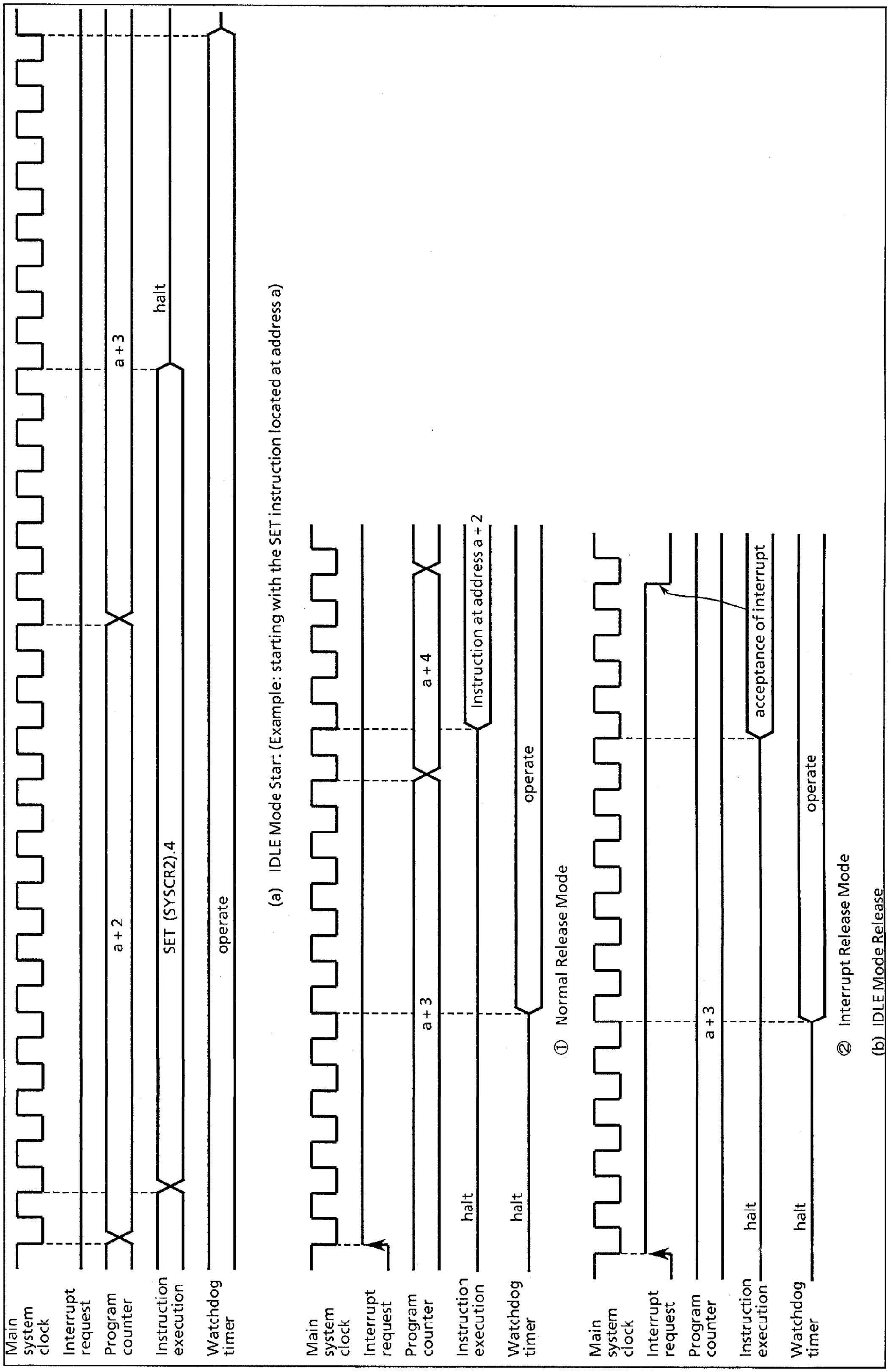

Starting IDLE mode by instruction
CPU, WDT are halted
Reset input
Yes
Reset
No (high)
No
Interrupt request
Normal release mode
Yes
No
IMF = 1
Yes (Interrupt release mode)
Interrupt processing
Execution of the instruction which follows the IDLE mode start instruction

Figure 1-19. IDLE Mode

b. Interrupt release mode (IMF = "1")
IDLE mode is released and interrupt processing is started by any interrupt source enabled with the individual interrupt enable flag (EF). After the interrupt is processed, the execution resumes from the instruction following the instruction which started IDLE mode.

IDLE mode can also be released by setting the RESET pin low, which immediately performs the reset operation. After reset, the 87CC31/H31 are placed in NORMAL mode.

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



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

① Normal Release Mode

② Interrupt Release Mode

(b) IDLE Mode Release

Figure 1-20. IDLE Mode Start/Release

1.9 Interrupt Controller

The 87CC31/H31 has a total of 11 interrupt sources: 3 externals and 8 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 1-21 shows the interrupt controller.

Table 1-2. Interrupt Sources

Interrupt Source		Enable Condition	Interrupt Latch	Vector Table Address	Priority
Internal/ External	(Reset)	Non-Maskable	—	FFFE_H	High 0
Internal	INTSW (Software interrupt)	Pseudo non-maskable	—	FFFC_H	1
Internal	INTWDT (Watchdog Timer interrupt)		IL_2	FFFA_H	2
	reserved	IMF = 1, INT0EN = 1	IL_3	FFF8_H	3
Internal	INTTC1 (16-bit TC1 interrupt)	$\text{IMF} \cdot \text{EF}_4 = 1$	IL_4	FFF6_H	4
	reserved	$\text{IMF} \cdot \text{EF}_5 = 1$	IL_5	FFF4_H	5
Internal	INTTBT (Time Base Timer interrupt)	$\text{IMF} \cdot \text{EF}_6 = 1$	IL_6	FFF2_H	6
	reserved	$\text{IMF} \cdot \text{EF}_7 = 1$	IL_7	FFF0_H	7
Internal	INTTC3 (8-bit TC3 interrupt)	$\text{IMF} \cdot \text{EF}_8 = 1$	IL_8	FFEE_H	8
	reserved	$\text{IMF} \cdot \text{EF}_9 = 1$	IL_9	FFEC_H	9
Internal	INTTC4 (8-bit TC4 interrupt)	$\text{IMF} \cdot \text{EF}_{10} = 1$	IL_{10}	FFEA_H	10
External	INT3 (External interrupt 3, Remote control receive interrupt)	$\text{IMF} \cdot \text{EF}_{11} = 1$	IL_{11}	FFE8_H	11
External	INT4 (External interrupt 4)	$\text{IMF} \cdot \text{EF}_{12} = 1$	IL_{12}	FFE6_H	12
Internal	INTOSD (OSD interrupt)	$\text{IMF} \cdot \text{EF}_{13} = 1$	IL_{13}	FFE4_H	13
Internal	INTTC2 (16-bit TC2 interrupt)	$\text{IMF} \cdot \text{EF}_{14} = 1$	IL_{14}	FFE2_H	14
External	INT5 (External interrupt 5)	$\text{IMF} \cdot \text{EF}_{15} = 1$	IL_{15}	FFE0_H	Low 15

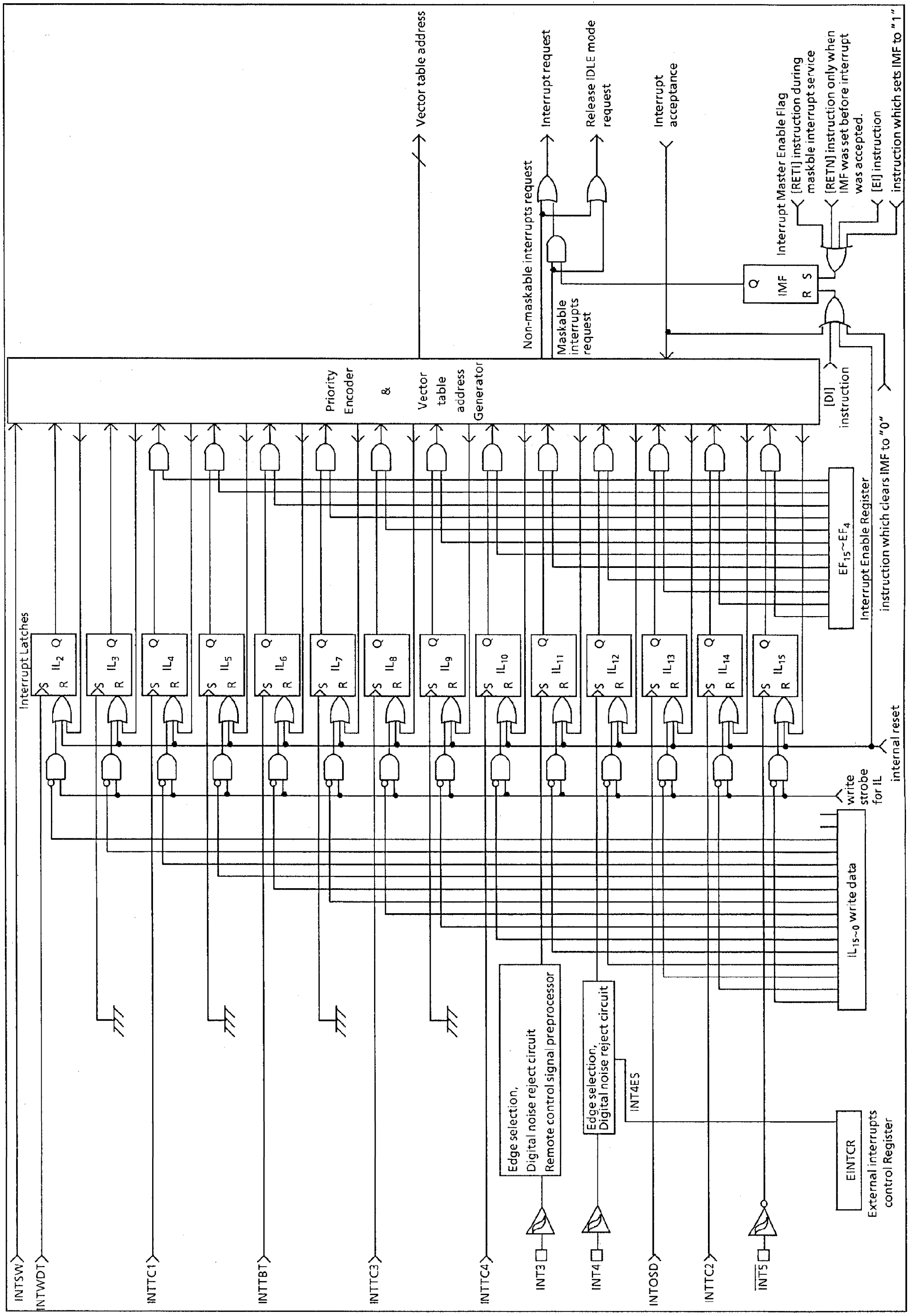

Vector table address
Non-maskable interrupts request
Interrupt request
Release IDLE mode request
Interrupt acceptance
Maskable interrupts request
Q
IMF
R S
Interrupt Master Enable Flag
[RETI] instruction during maskble interrupt service
[RETN] instruction only when IMF was set before interrupt was accepted.
[EI] instruction
instruction which sets IMF to "1"
Priority Encoder
&
Vector table address Generator
[DI] instruction
instruction which clears IMF to "0"
EF_{15}~EF_4
Interrupt Enable Register
Interrupt Latches
IL_2
IL_3
IL_4
IL_5
IL_6
IL_7
IL_8
IL_9
IL_{10}
IL_{11}
IL_{12}
IL_{13}
IL_{14}
IL_{15}
S
R
Q
internal reset
write strobe for IL
$IL_{15\sim0}$ write data
Edge selection,
Digital noise reject circuit
Remote control signal preprocessor
Edge selection,
Digital noise reject circuit
INT4ES
EINTCR
External interrupts control Register
INTSW
INTWDT
INTTC1
INTTBT
INTTC3
INTTC4
INT3
INT4
INTOSD
INTTC2
$\overline{INT5}$

Figure 1-21. Interrupt Controller Block Diagram

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

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

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

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

Example 1 : Clears interrupt latches

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

Example 2 : Reads interrupt latches

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

Example 3 : Tests an interrupt latch

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

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

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

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

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

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

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

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

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

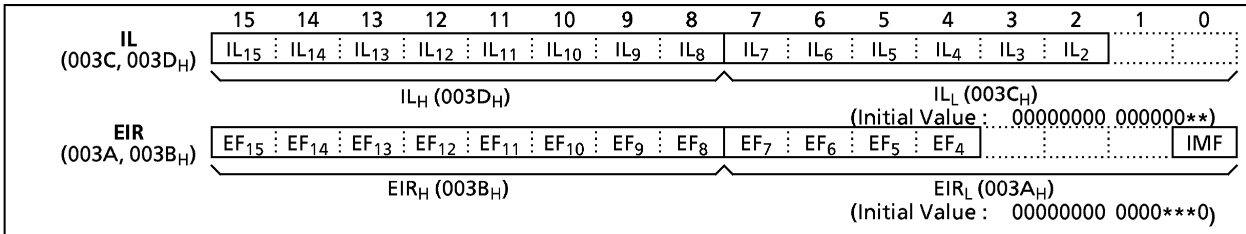
② **Individual interrupt Enable Flags** (EF$_{15}$ to EF$_4$)
These flags enable and disable the acceptance of individual maskable interrupts. Setting the corresponding bit of an individual interrupt enable flag to "1" enables acceptance of an interrupt, setting the bit to "0" disables acceptance.

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

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

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

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



Note1: Do not use any read-modify-write instruction such as bit manipulation for clearing IL.
Note2: Do not clear IL$_2$ to "0" by an instruction.
Note3: Do not set IMF to "1" during non-maskable interrupt service program.

Figure 1-22. Interrupt Latch (IL) and Interrupt Enable Register (EIR)

1.9.1 Interrupt Sequence

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

(1) Interrupt acceptance processing is as follows:

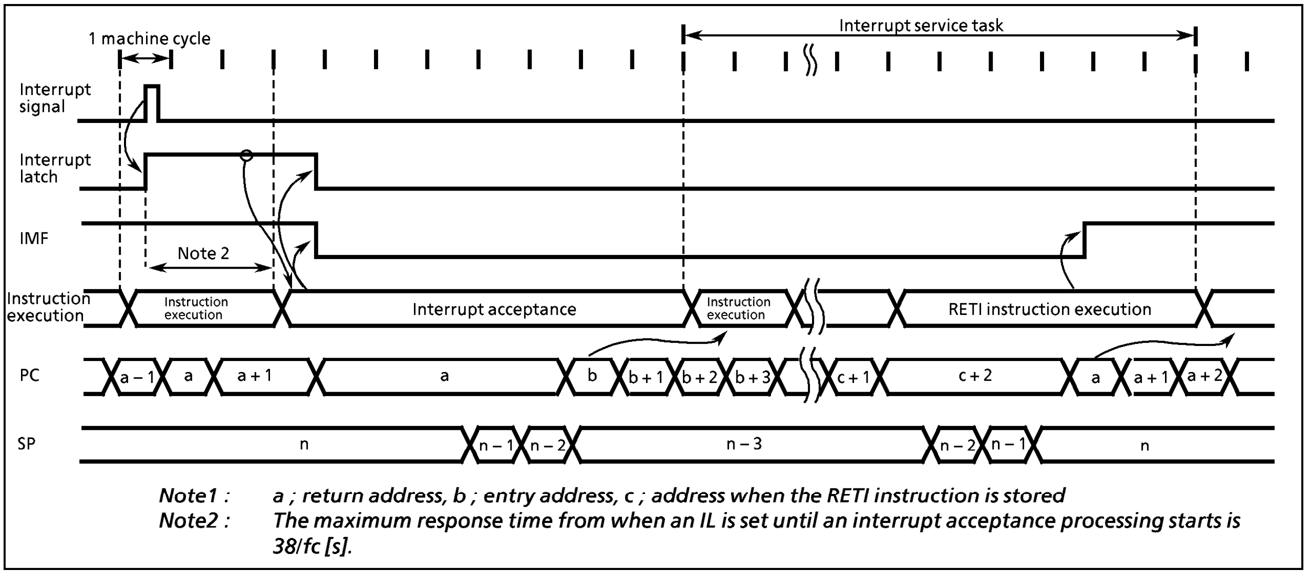
① The interrupt master enable flag (IMF) is cleared to "0" to temporarily disable the acceptance of any following maskable interrupts. When a non-maskable interrupt is accepted, the acceptance of any following interrupts is temporarily disabled.

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

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

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

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

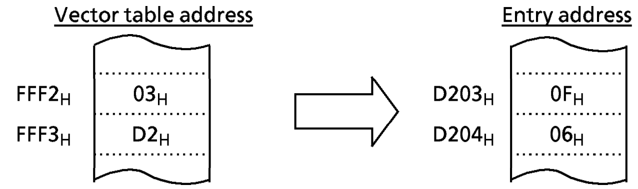


Note1 : a ; return address, b ; entry address, c ; address when the RETI instruction is stored

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

Figure 1-23. 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 (1 μs at 8 MHz)
         Interrupt processing
         RETI        ; Restores bank and Returns
```

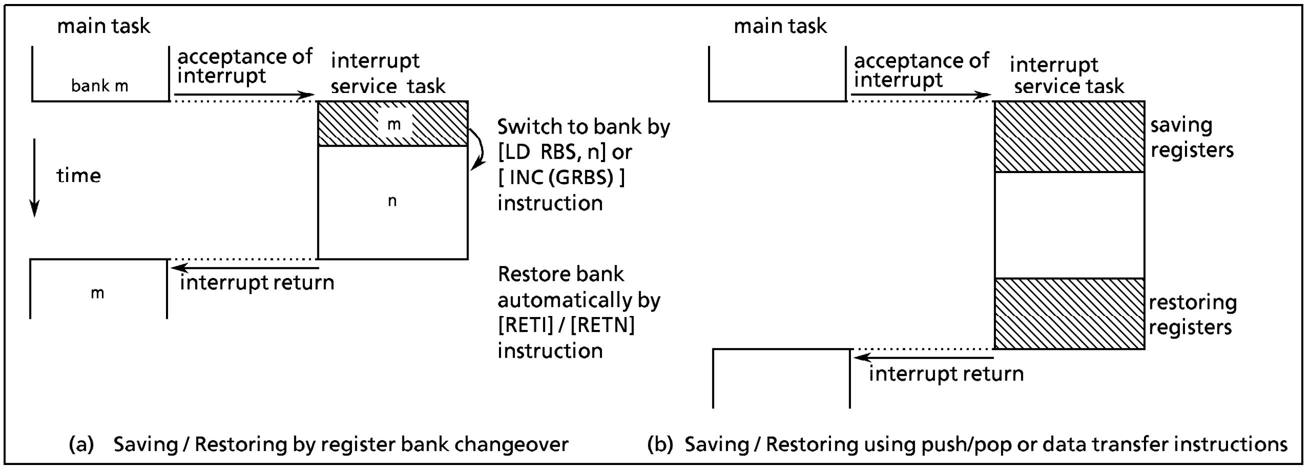
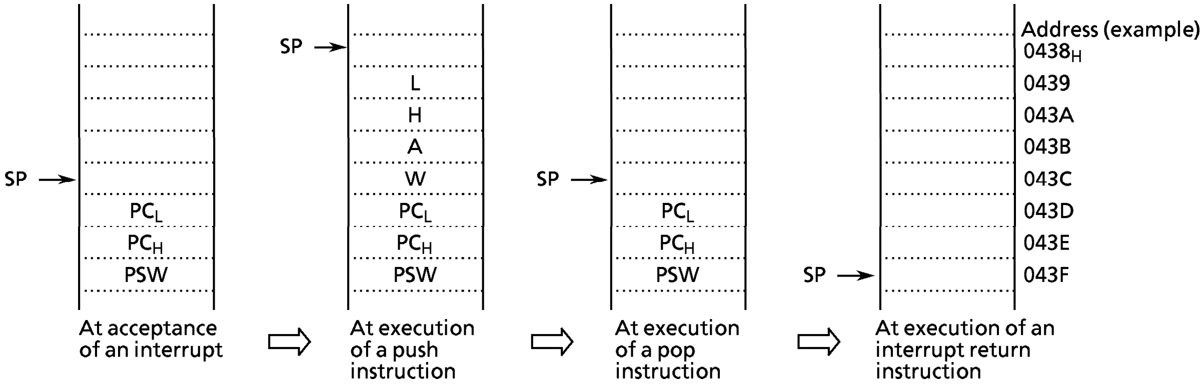

main task
bank m
acceptance of interrupt
interrupt service task
m
n
Switch to bank by [LD RBS, n] or [INC (GRBS)] instruction
time
m
interrupt return
Restore bank automatically by [RETI] / [RETN] instruction
(a) Saving / Restoring by register bank changeover
main task
acceptance of interrupt
interrupt service task
saving registers
restoring registers
interrupt return
(b) Saving / Restoring using push/pop or data transfer instructions

Figure 1-24. Saving/Restoring General-purpose Registers

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

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


SP
PCL
PCH
PSW
At acceptance of an interrupt
SP
L
H
A
W
PCL
PCH
PSW
At execution of a push instruction
SP
PCL
PCH
PSW
At execution of a pop instruction
SP
At execution of an interrupt return instruction
Address (example)
0438H
0439
043A
043B
043C
043D
043E
043F

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
```

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 is fetched from RAM or SFR areas.

Note : The fetch data from addresses BF80$_H$ to BFFF$_H$ (test ROM area) is not "FF$_H$".

② Debugging

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

1.9.3 External Interrupts

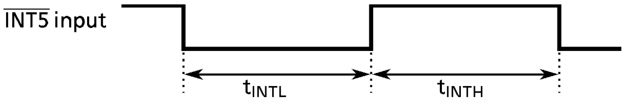
The 87CC31/H31 has three external interrupt inputs (INT3, INT4, and $\overline{\text{INT5}}$). Two of these are equipped with digital noise rejection circuits (pulse inputs of less than a certain time are eliminated as noise). Edge selection is also possible with INT3 and INT4.

Edge selection of INT4 pin input is performed by the external interrupt control register (EINTCR). 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 processor.)

Table 1-3. External Interrupts

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

Note 1 : The pulse width (both "H" and "L" level) for input to the INT5 pin must be over 1 machine cycle.



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

Note 2 : If a noiseless signal is input to the external interrupt pin, the maximum time from the edge of input signal until the IL is set is as follows :

① INT4 pin 25/fc [s]

Note 3 : 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 ($\overline{\text{INT5}}$ / $\overline{\text{STOP}}$) 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 (TMP87CC31/H31) :

```
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)
```

	7	6	5	4	3	2	1	0	
EINTCR (0037_H)	"0"	"0"	(TC4ES)	INT4 ES	(TC3ES)	"0"	"0"		(Initial value : 00*0 000*)

INT4 ES	INT4 edge select	0 : Rising edge 1 : Falling edge	write only

*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 INT4ES during NORMAL mode, clear a interrupt latch of external interrupt input (INT4) after 8 machine cycles from the time of rewriting.

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

Note 5 : Always write "0" to bit 7, 6, 2, 1 in EINTCR.

Figure 1-25. 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

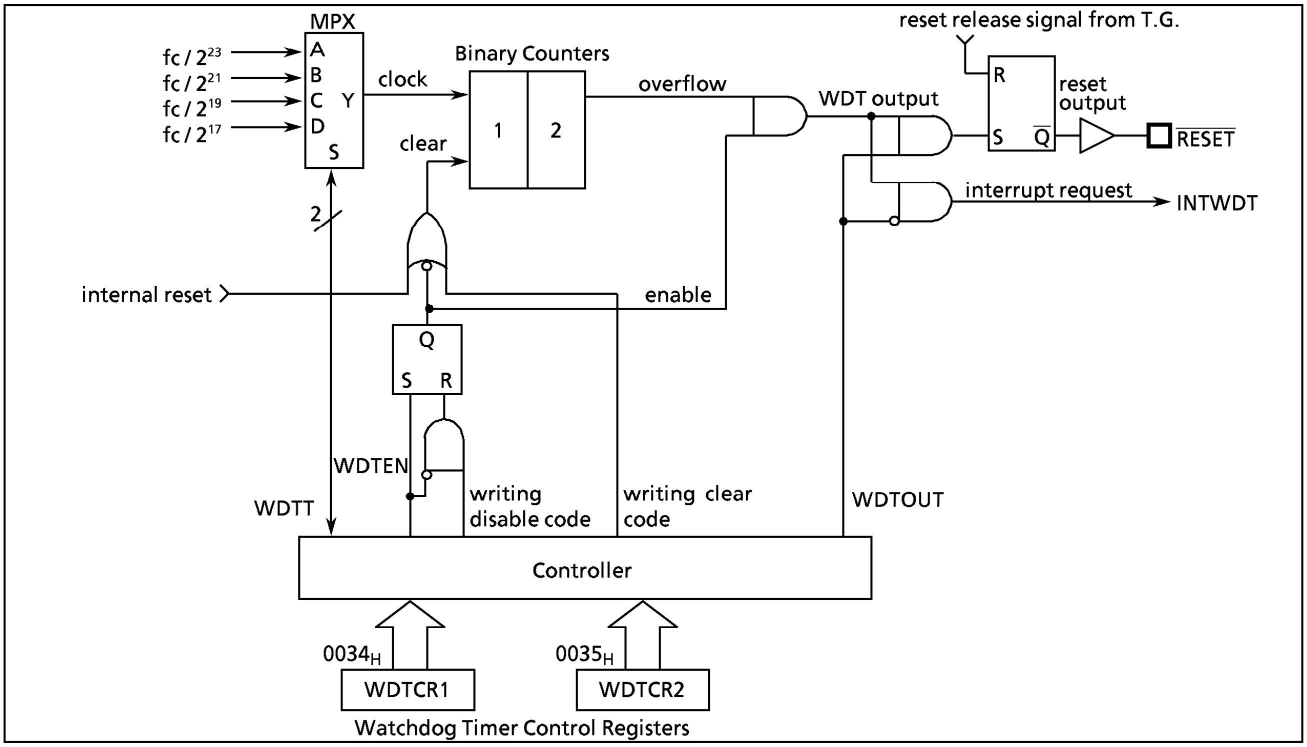


Figure 1-26. Watchdog Timer Configuration

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
Within WDT       [  LD   (WDTCR2),   4EH        ; Clears the binary counters
detection time   [  :                             (always clear immediately after changing WDTT)
                 [- LD   (WDTCR2),   4EH        ; Clears the binary counters
Within WDT       [  :
detection time   [  
                 [_ LD   (WDTCR2),   4EH        ; Clears the binary counters
                    :
```

Watchdog Timer Control Register 1

WDTCR1
(0034$_H$)

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

(Initial value : **** 1001)

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

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

Watchdog Timer Control Register 2

WDTCR2
(0035$_H$)

7	6	5	4	3	2	1	0

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

WDTCR2	Watchdog timer control code write register	4E$_H$: Watchdog timer binary counter clear (clear code) B1$_H$: Watchdog timer disable (disable code) others : Invalid	write only

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

Figure 1-27. 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 RESET pin (sink open drain output) low to reset the internal hardware and the external circuits. The reset output time is 2^{20}/fc [s] (131 ms at fc = 8 MHz).

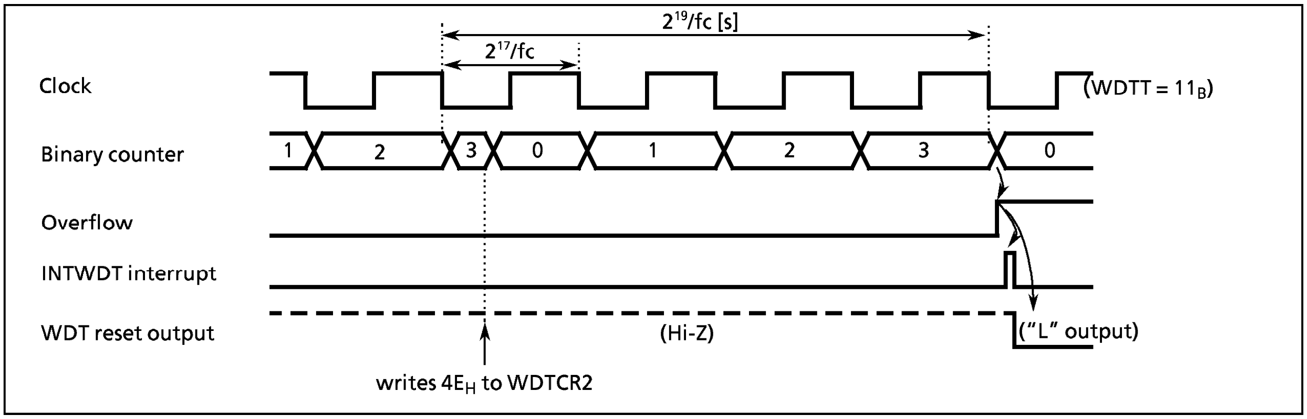


Figure 1-28. Watchdog Timer Interrupt / Reset

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 RESET pin may go low (2^{20}/fc [s] 131 ms at 8 MHz) when power is turned on.

Table 1-4. Initializing Internal Status by Reset Action

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

1.11.1 External Reset Input

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

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

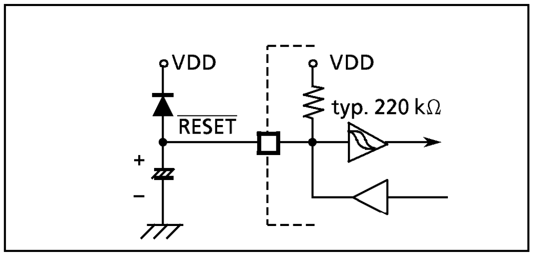
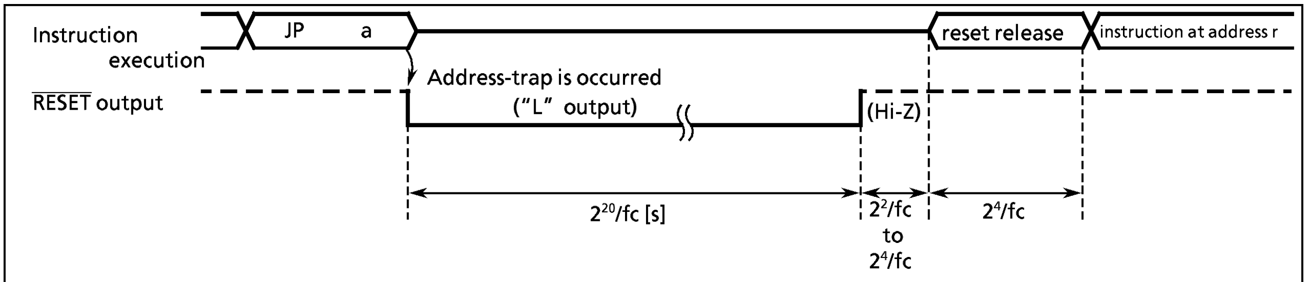


Figure 1-29. Simple Power-on-Reset Circuitry

1.11.2 Address-Trap-Reset

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



Note 1: 0≦ a≦ 043F$_H$

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

Figure 1-30. 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] (131 ms at fc = 8 MHz).

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 to $003F_H$, and the DBR to addresses $0F80_H$ to $0FFF_H$.

Figure 2-1 shows the list of the 87CC31/H31 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	reserved	
21	reserved	
22	reserved	
23	reserved	
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 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
$0F80_H$	–	OSD control registers
	–	
0F94	–	
95	DCTR (OSD display-line counter)	
96	–	
	–	
9A	ORDON	
0F9B	reserved	
	reserved	
0FCF	reserved	
0FD0	RXCR1 (Remo-con control1)	
D1	RXCR2 (Remo-con control2)	
D2	RXCTR (Remo-con receive counter)	–
D3	RXDBR (Remo-con receive 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 : PWM : Pulse Width Modulation
OSD : On Screen Display
Remo-con : Remote control

Figure 2-1. SFR & DBR

2.2 I/O Ports

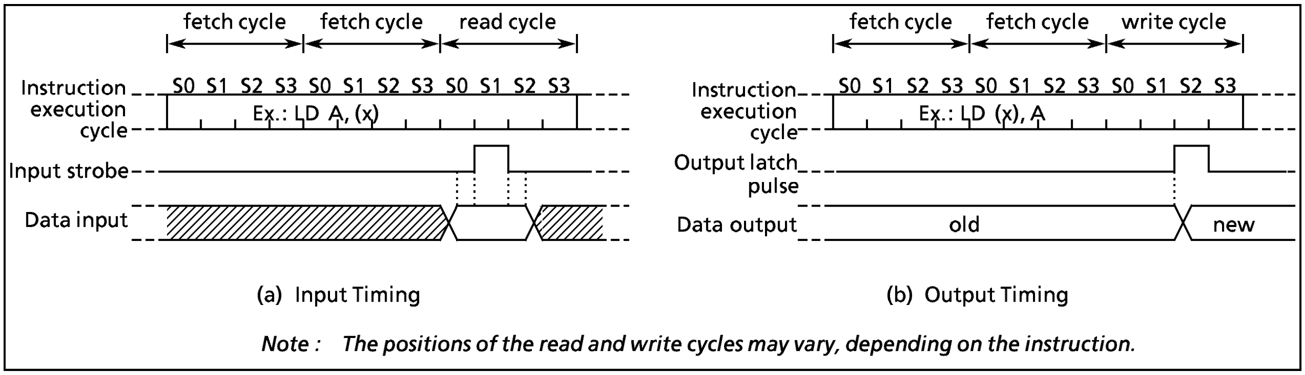
The 87CC31/H31 has 6 parallel input / output ports (34pins) as follows:

	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 and timer / counter 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
Port P6	8-bit I/O port	R, G, B and Y/BL output from OSD circuitry
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.



(a) Input Timing

(b) Output Timing

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

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

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

(1) Instructions that read the output latch contents

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

(2) Instructions that read the pin input data

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

2.2.1 Port 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.

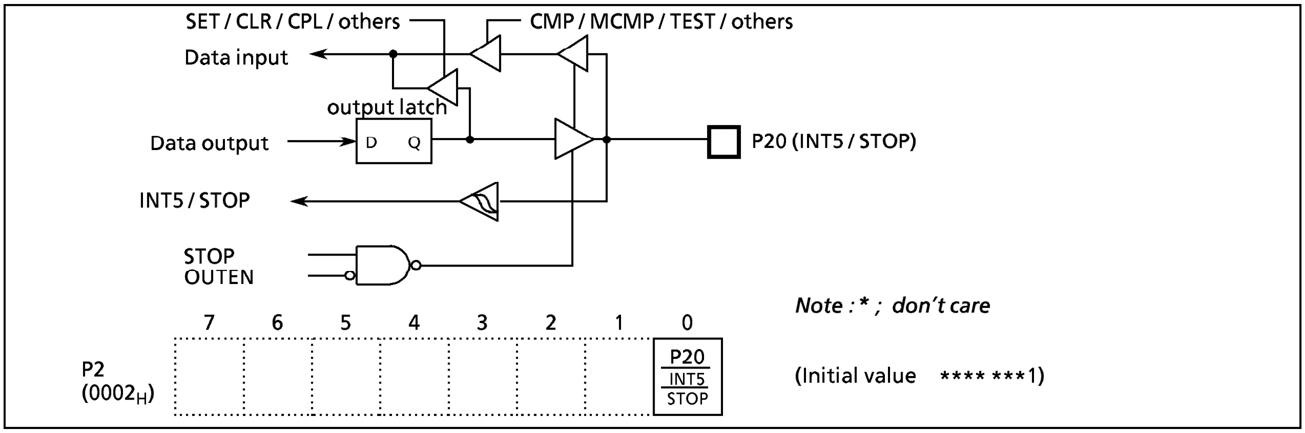


Figure 2-3. Port P2

2.2.2 Port P3 (P36 to P30)

Port P3 is a 7-bit input / output port, and is also used as an exrernal interrupt input a timer / counter input, and Remote-control signal input. When used as an input port or a secondary function pin, the output latch should be set to "1". The output latches are initialized to "1" during reset.

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

```
LD  (P3), 5AH        ; P3←5AH
```

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

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

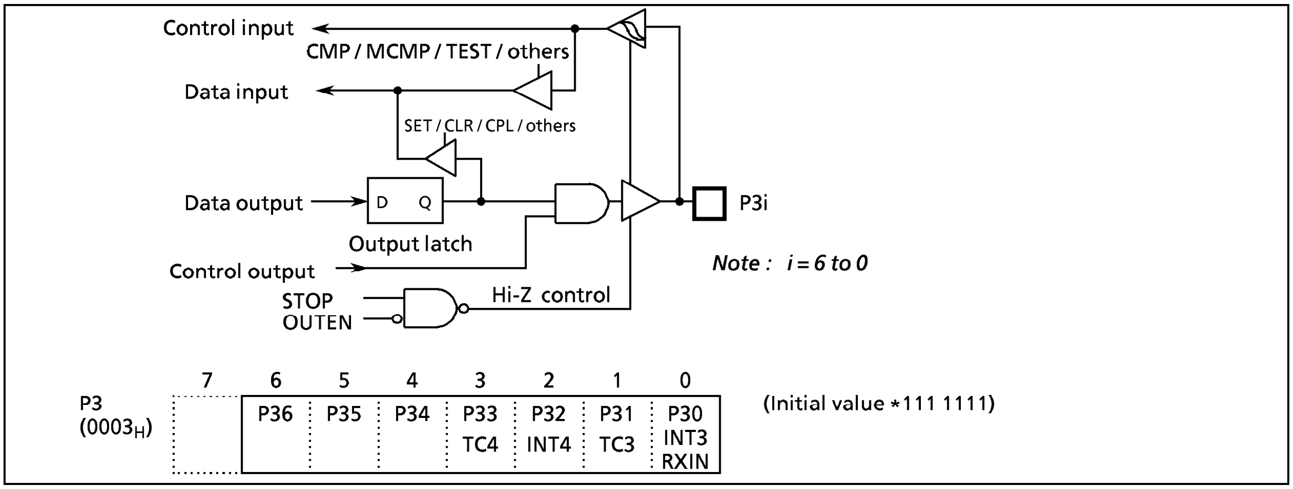


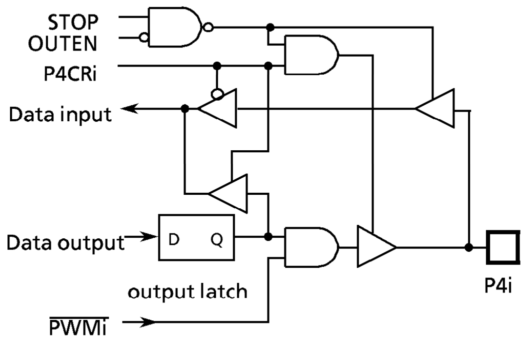
Figure 2-4. Port P3

2.2.3 Port P4 (P47 to P40)

Port P4 is an 8-bit input / output port which can be configured as an input or an output in one-bit unit under software control. Input / output mode is specified by the corresponding bit in the port P4 input / output control register (P4CR). Port P4 is configured as an input if its corresponding P4CR bit is cleared to "0", and as an output if its corresponding P4CR bit is set to "1". During reset, P4CR is initialized to "0", which configures port P4 as an input. The P4 output latches are also initialized to "1".

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

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



Note : i = 7 to 0

P4 (0004$_H$)

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

(Initial value : 1111 1111)

P4CR (000C$_H$)

7	6	5	4	3	2	1	0

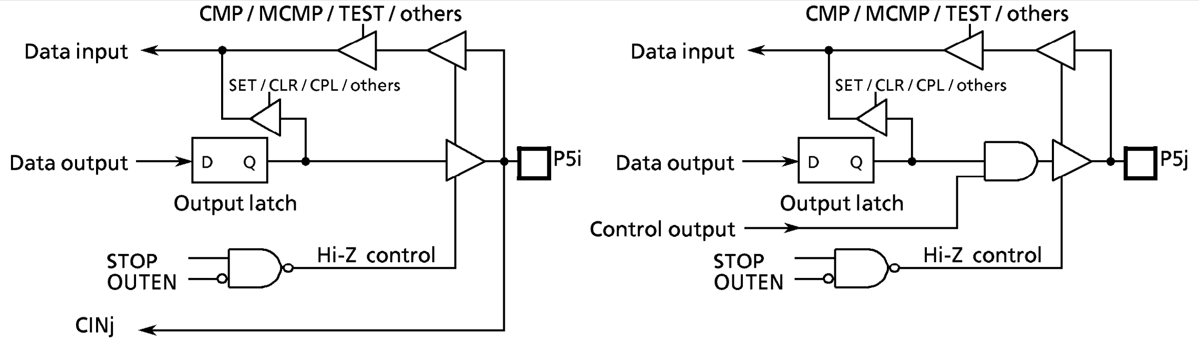
(Initial value : 0000 0000)

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 to P50)

Port P5 is an 8-bit input / output port, and is also used as comparator input, a pulse output, and a pulse width modulation (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.



(a) P57 to P54 *Note : i = 7 to 4, j = 3 to 0* (b) P53 to P50

P5 (0005$_H$)

7	6	5	4	3	2	1	0
P57 CIN3	P56 CIN2	P55 CIN1	P54 CIN0	P53	P52 $\overline{PULSE}$	P51 $\overline{PWM9}$	P50 $\overline{PWM8}$

(Initial value : 1111 1111)

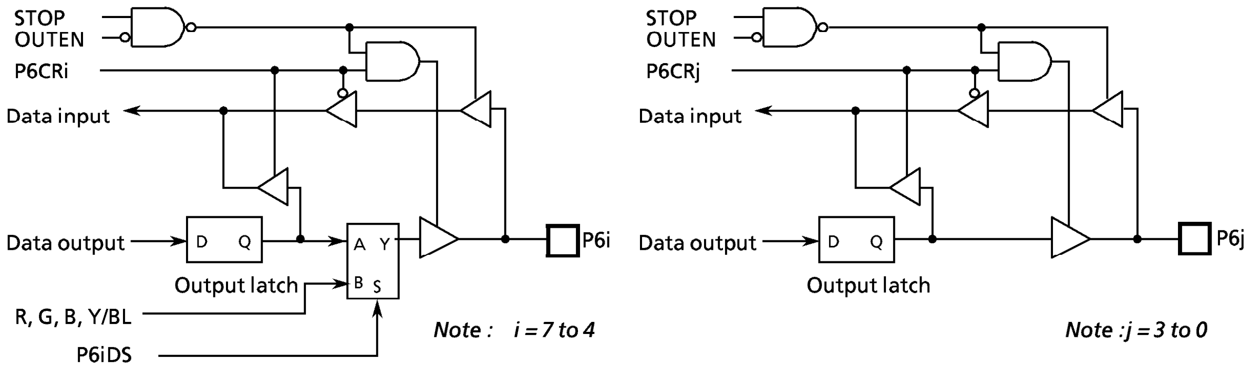
Figure 2-6. Ports P5

2.2.5 Port P6 (P67 to P60)

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

Data is written into the output latch regardless of the P6CR contents. Therefore initial output data should be written into the output latch before setting P6CR. Pins P63 to P60 are available high current output, so LEDs can be driven directly.

Port P6 is also used as an on screen display (OSD) output (R, G, B, and Y/BL signal). When used as an OSD output pin, the OSD output pins should be set to the output mode and beforehand the port P6 data selection register (P67DS to P64DS) should be set to "1".



(a) P67 to P64

(b) P63 to P60

P6 (0006$_H$)

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

(Initial value 1111 1111)

P6CR (000D$_H$)

7	6	5	4	3	2	1	0

(Initial value 0000 0000)

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

Port P6 data selection register (P67DS to P64DS)

(0F91$_H$)

7	6	5	4	3	2	1	0
P67DS	P66DS	P65DS	P64DS				

(Initial value 0000 ****)

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

Note1 : n = 7 to 4
Note2 : P6CR, P67DS to P64DS is write-only-register and must not be used with any of read-modify-write instructions.

Figure 2-7. Ports P6, P6CR, and P67DS to P64DS

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

Example : Set the lower 4 bit in port P6 (P63 to 60) to the output port and Set the other to the input port.

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

2.2.6 Port P7 (P71 to P70)

Port P7 is a 2-bit input / output port, and is also used as a vertical synchronous signal ($\overline{\text{VD}}$) input and a horizontal synchronous signal ($\overline{\text{HD}}$) 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.

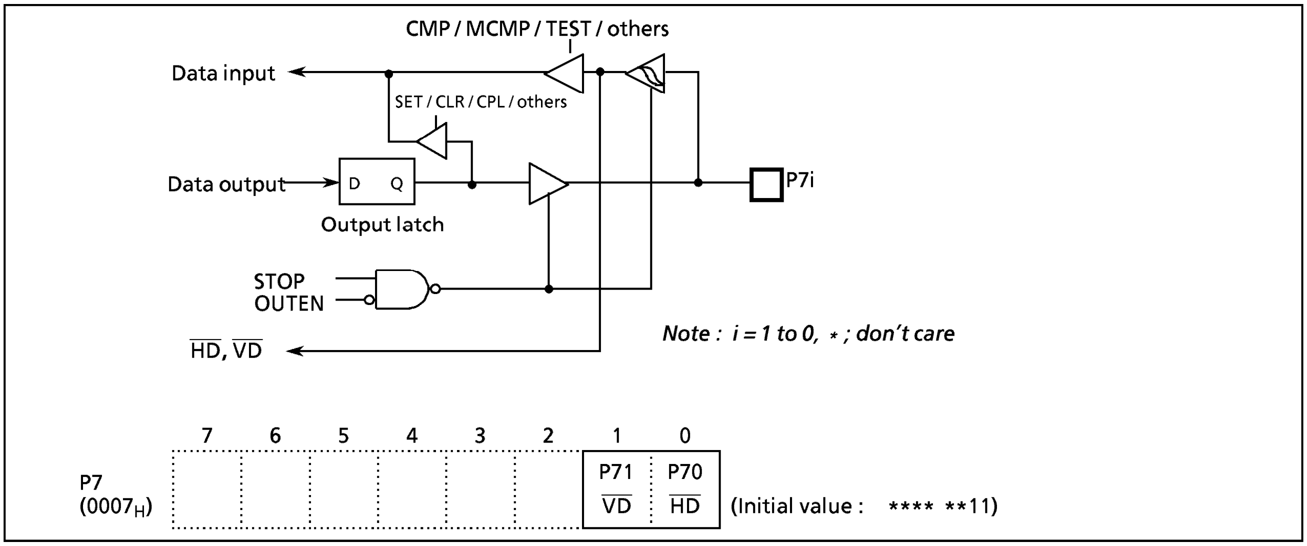


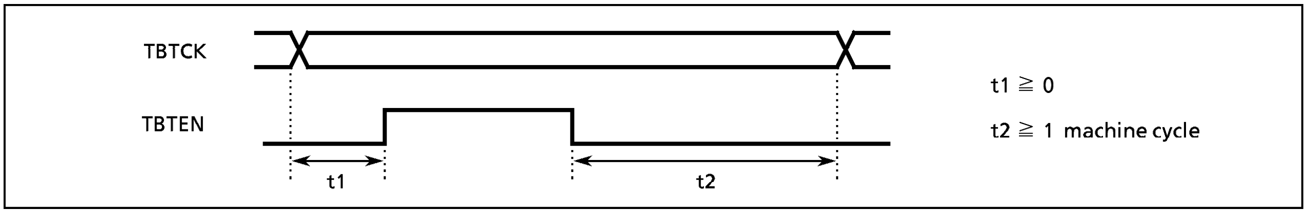
Figure 2-8. Ports P7

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
```

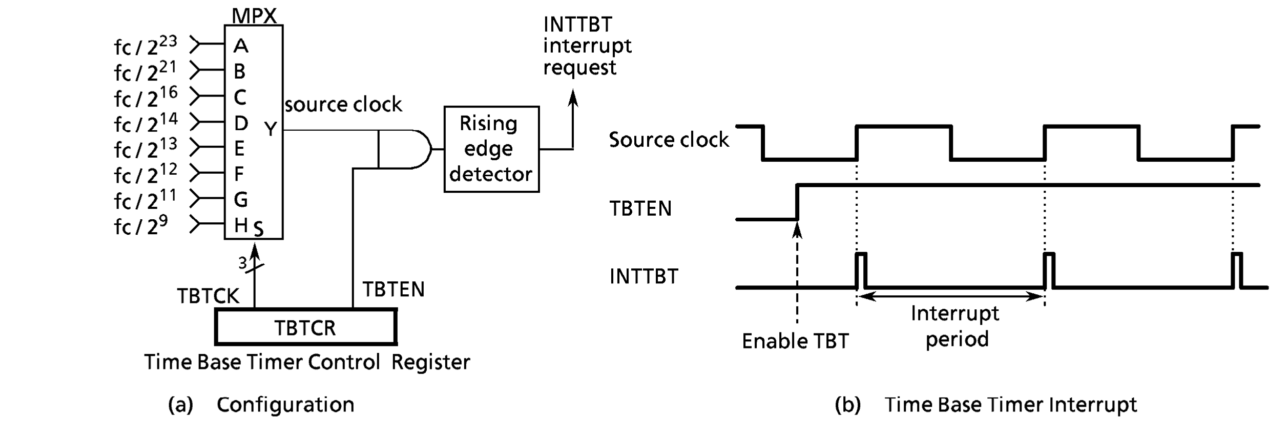


Figure 2-9. Time Base Timer

	7	6	5	4	3	2	1	0
TBTCR (0036_H)	"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 = 8 MHz) 001 : fc / 2^{21} (3.81 at fc = 8 MHz) 010 : fc / 2^{16} (122.07 at fc = 8 MHz) 011 : fc / 2^{14} (488.28 at fc = 8 MHz) 100 : fc / 2^{13} (976.56 at fc = 8 MHz) 101 : fc / 2^{12} (1953.12 at fc = 8 MHz) 110 : fc / 2^{11} (3906.25 at fc = 8 MHz) 111 : fc / 2^{9} (15625 at fc = 8 MHz)	

*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)

2.4.1 Configuration

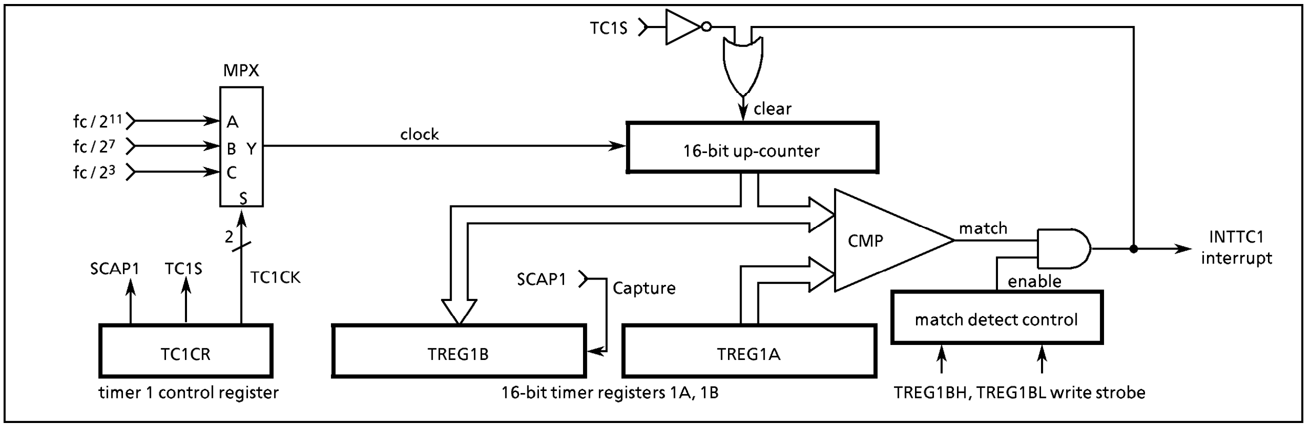


Figure 2-11. Timer 1 (TC1)

2.4.2 Control

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
TREG1A (0010, 0011$_{H}$)	TREG1A$_{H}$ (0011$_{H}$)								TREG1A$_{L}$ (0010$_{H}$)								Write only
TREG1B (0012, 0013$_{H}$)	TREG1B$_{H}$ (0013$_{H}$)								TREG1B$_{L}$ (0012$_{H}$)								Read only

	7	6	5	4	3	2	1	0	
TC1CR (0014$_{H}$)	"0"	SCAP1	TC1S		TC1CK		TC1M		(Initial value : 0000 0000)

TC1M	TC1 mode select	00 : timer mode 01 : reserved 1* : 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 : reserved	
TC1S	TC1 start control	00 : stop & counter clear 01 : command start 1* : reserved	
SCAP1	software capture control	0 : – 1 : software capture trigger	

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

Note 2 : Writing to the low-byte of the timer register (TREG1A$_{L}$), the comparison is inhibited until the high-byte (TREG1A$_{H}$) is written.
After writing to the high-byte, any match during 1 machine cycle (instruction cycle) is ignored.

Note 3 : Set the source clock, when TC1 stops (TC1S = 00).

Note 4 : SCAP1 is automatically cleared to "0" after capturing.

Note 5 : Values to be loaded to timer registers must satisfy the following condition.
TREG1A > 0

Note 6 : Always write "0" to bit 7 in TC1CR.

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

Note 8 : The TREG1B can not be written.

Figure 2-12. Timer Registers and TC1 Control Register

2.4.3 Function

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

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

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

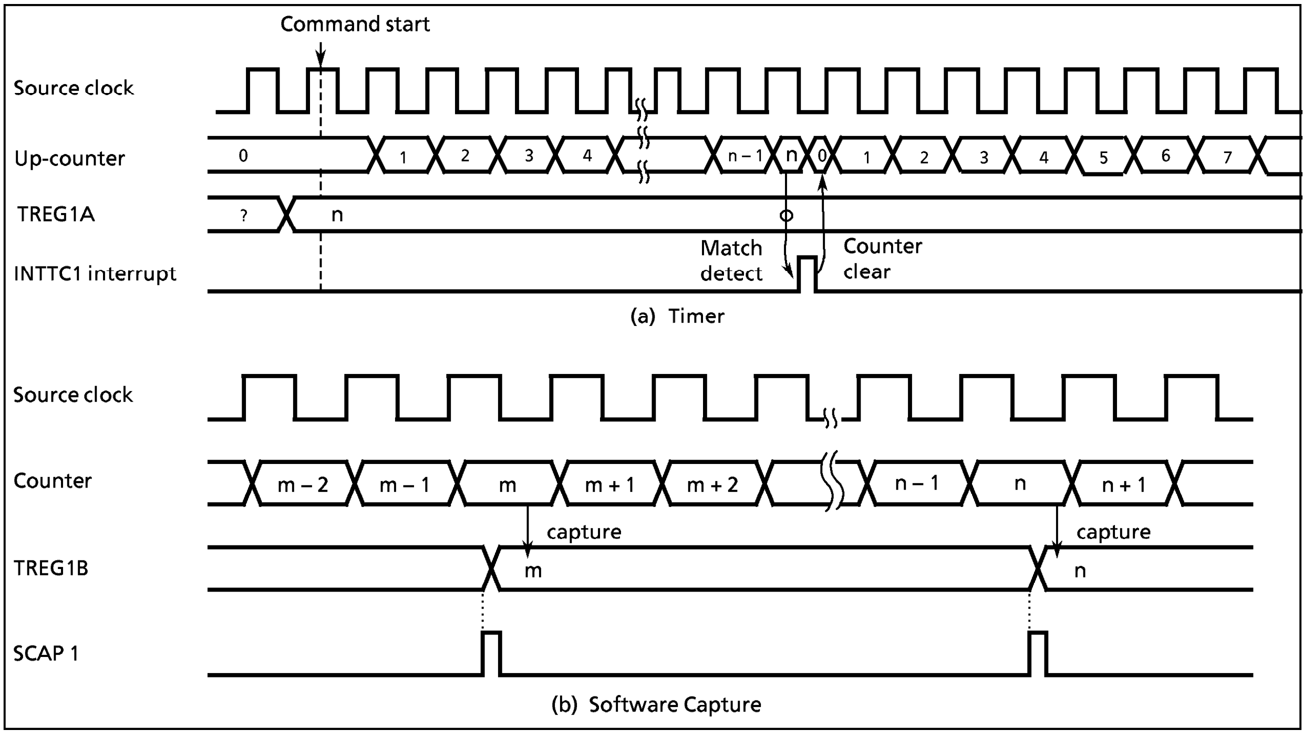
Example 1 : Sets the source clock to fc/2^7 [Hz] and generates an interrupt 1 [s]. later (at fc = 8 MHz).

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

Note : The TC1CR is write-only register and can not be started by [SET (TC1CR). 4] instruction.

Example 2 : Software capture

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



(a) Timer

(b) Software Capture

Figure 2-13. Timer Mode Timing Chart

2.5 16-bit Timer 2 (TC2)

2.5.1 Configuration

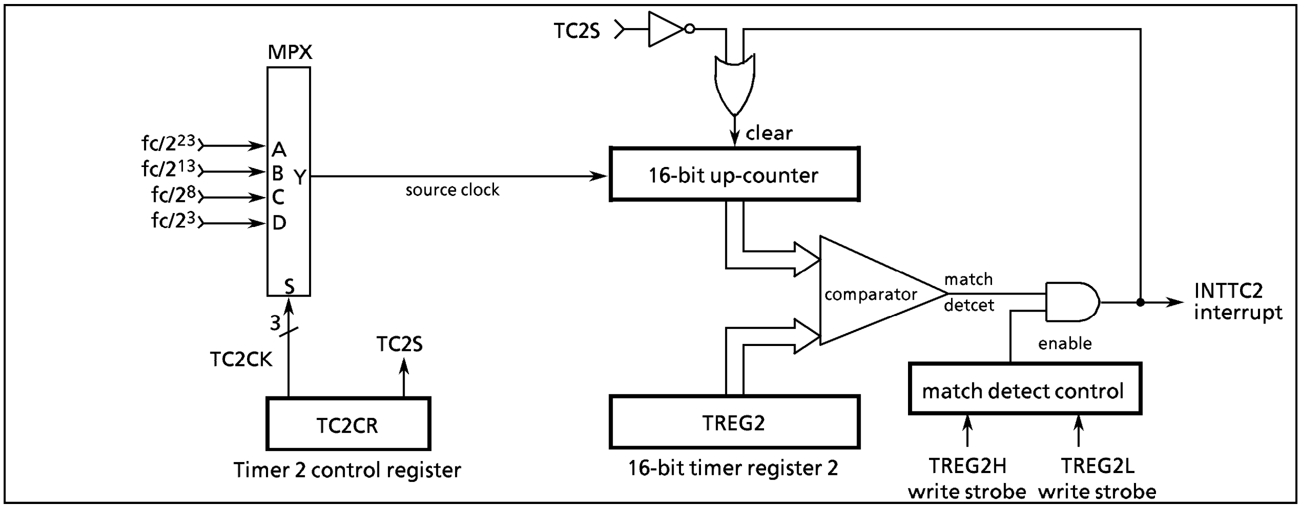
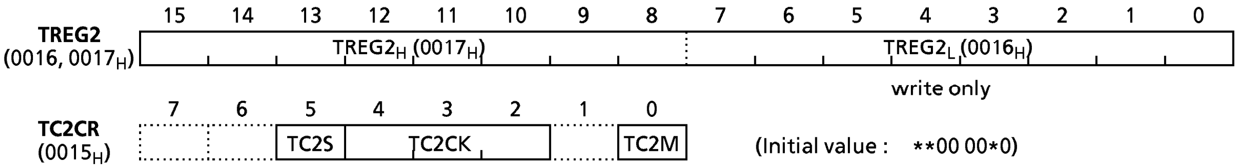


Figure 2-14. Timer 2 (TC2)

2.5.2 Control

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



TC2M	Timer 2 operating mode select	0 : Timer mode 1 : Reserved	write only
TC2CK	Timer 2 source clock select	000 : fc / 2^{23} [Hz] 001 : fc / 2^{13} 010 : fc / 2^{8} 011 : fc / 2^{3} 1** : Reserved	
TC2S	Timer 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 source clock when timer stop (TC2S = 0).

Note 4 : Values to be loaded to the timer register must satisfy the following condition.
TREG2 > 0.

Note 5 : Always write "0" to bit 0 and bit 4 in TC2CR.

Note 6 : The TC2CR and the TREG2 are write-only registers, which cannot access any of in read-modify-write instructions such as bit operate, etc.

Figure 2-15. Timer Register 2 and TC2 Control Register

2.5.3 Function

The contents of TREG2 are compared with the contents of the up-counter. If a match is found, a timer 2 interrupt (INTTC2) is generated, and the counter is cleared. Counting up is resumed after the counter is cleared.

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

Source clock	Resolution (At fc = 8 MHz)	Maximum time setting (At fc = 8 MHz)
$fc/2^{23}$ [Hz]	1.048576 s	19 hour 5 min 18.4 s
$fc/2^{13}$	1.024 ms	67.1 s
$fc/2^{8}$	32 μs	2.09712 s
$fc/2^{3}$	1 μs	65.535 ms

Example : Sets the source clock fc/2³ [Hz] and generates an interrupt every 25 ms (at fc = 8 MHz).

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

2.6 8-Bit Timer / Counter 3 (TC3)

2.6.1 Configuration

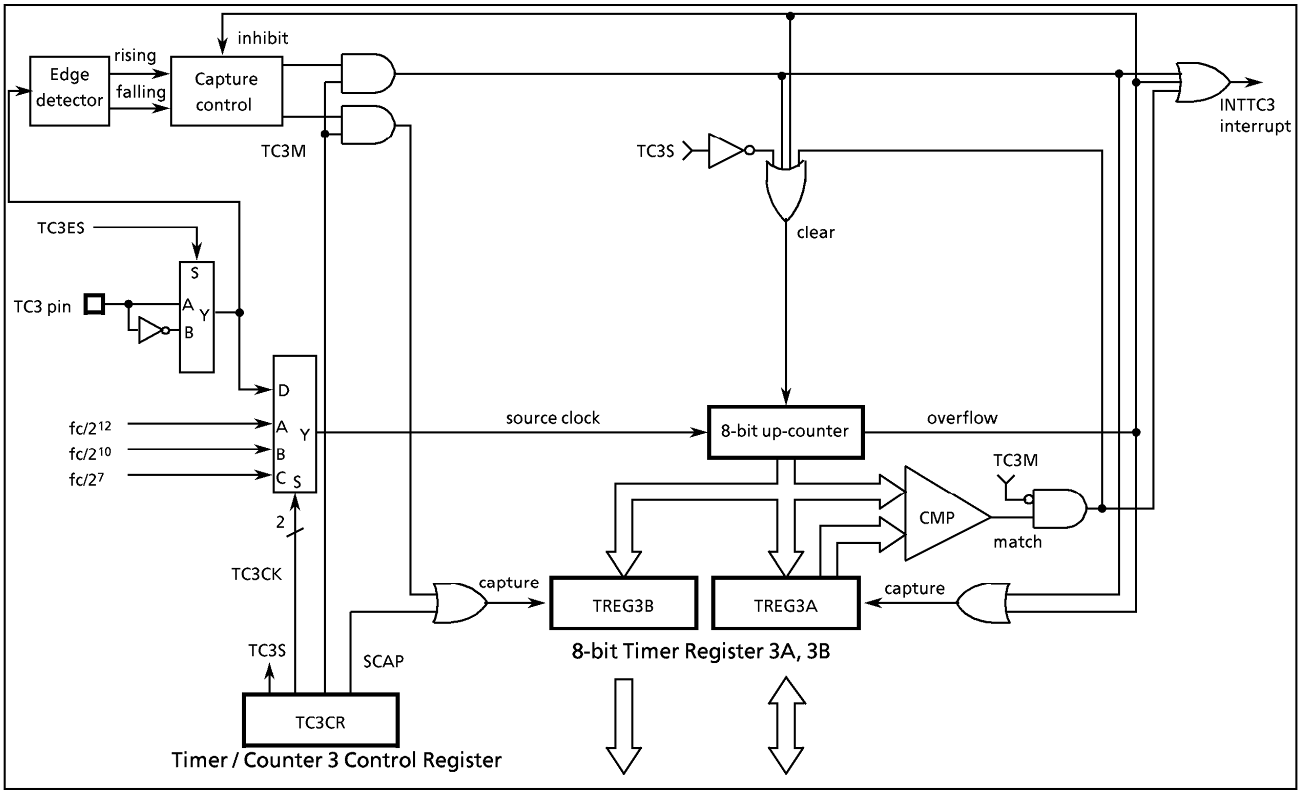


Figure 2-16. Timer/Counter 3

2.6.2 Control

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

	7	6	5	4	3	2	1	0	
TREG3A (0018_H)									Read/Write
TREG3B (0019_H)									Read only

	7	6	5	4	3	2	1	0	
TC3CR ($001A_H$)		SCAP		TC3S	TC3CK			TC3M	(Initial value : *0*0 00*0)

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

	7	6	5	4	3	2	1	0	
EINTCR (0037_H)	"0"	"0"	(TC4 ES)	(INT4 ES)	TC3 ES	"0"	"0"		(Initial value : 0000 000*)

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

*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 is a write-only-register, which cannot any of in read-modify-write instructions such as bit operate, etc.
Note 5 : Always write "0" to bit 7-6 and 2-1 in EINTCR.

Figure 2-17. 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.

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

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

(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 50 Hz pulses to the TC3 pin.

```
LD   (TC3CR), 00001100B ; Sets TC3 mode and source clock
LD   (TREG3A), 19H      ; 0.5 [s] ÷ 1 / 50 = 25 = 19H
SET  (EIRH).EF8         ; Enables INTTC3 interrupt
EI
LD   (TC3CR), 00011100B ; 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 contineus. On the next rising (falling) edge of the TC3 pin input, the current contents of counter are loaded into TREG3A, then the counter is cleared again and an interrupt is generated. If the counter overflows before the edge is detected, FF_H is set into TREG3A and an overflow interrupt (INTTC3) is generated. During interrupt processing, it can be determined whether or not there is an overflow by checking whether or not the TREG3A value is FF_H. Also, after an interrupt (capture to TREG3A, or overflow detection) is generated, capture and overflow detection are halted until TREG3A has been read out; however, the counter continues. Therefore, TREG3B has been read out earlier than TREG3A.

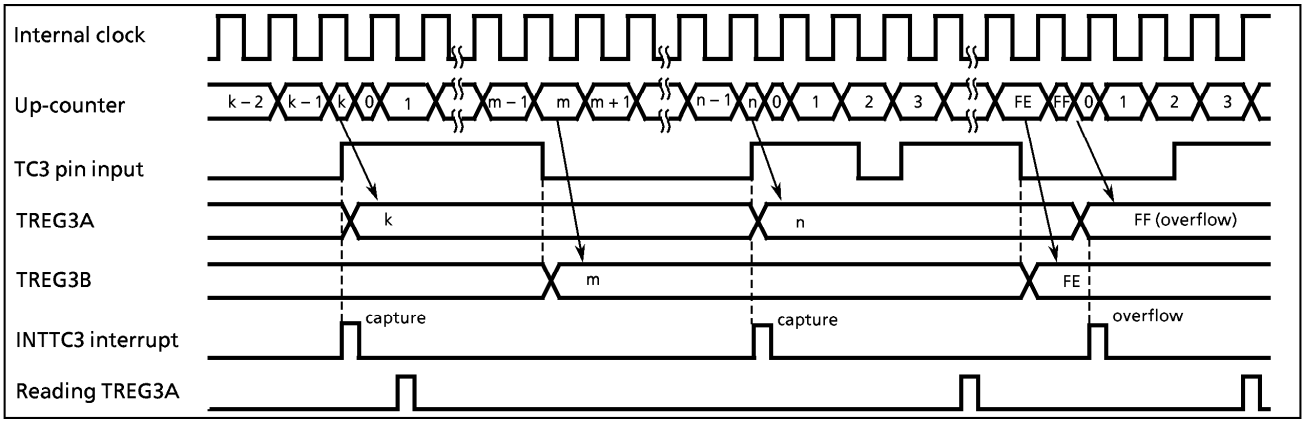


Figure 2-18. Timing Chart for Capture Mode (TC3ES = 0)

2.7 8-bit Timer / Counter (TC4)

2.7.1 Configuration

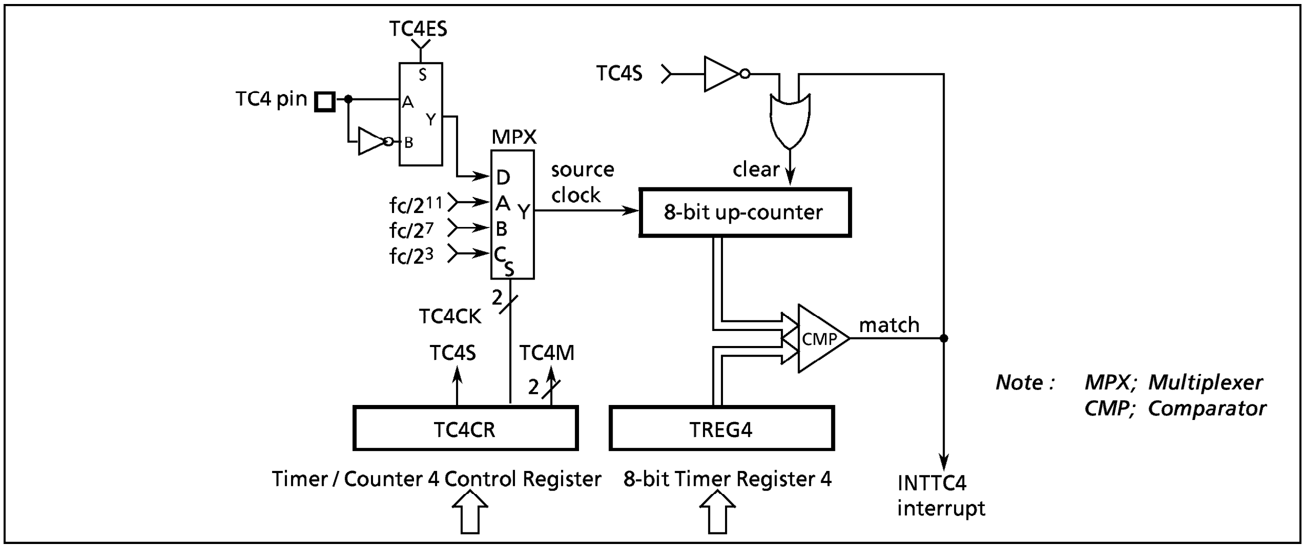


Figure 2-19. Timer / Counter 4

2.7.2 Control

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

TREG4 ($001B_H$)

7	6	5	4	3	2	1	0	
								Write only

TC4CR ($001C_H$)

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

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

EINTCR (0037_H)

7	6	5	4	3	2	1	0	
"0"	"0"	TC4 ES	(INT4 ES)	(TC3 ES)	"0"	"0"		(Initial value : 0000 000*)

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

*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, EINTCR and the TREG4 are write-only registers, which cannot access any of in read-modify-write instructions such as bit operate, etc.

Note 6 : Always write "0" to bit 7 - 6 and bit 2 - 1 in EINTCR.

Figure 2-20. 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.

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

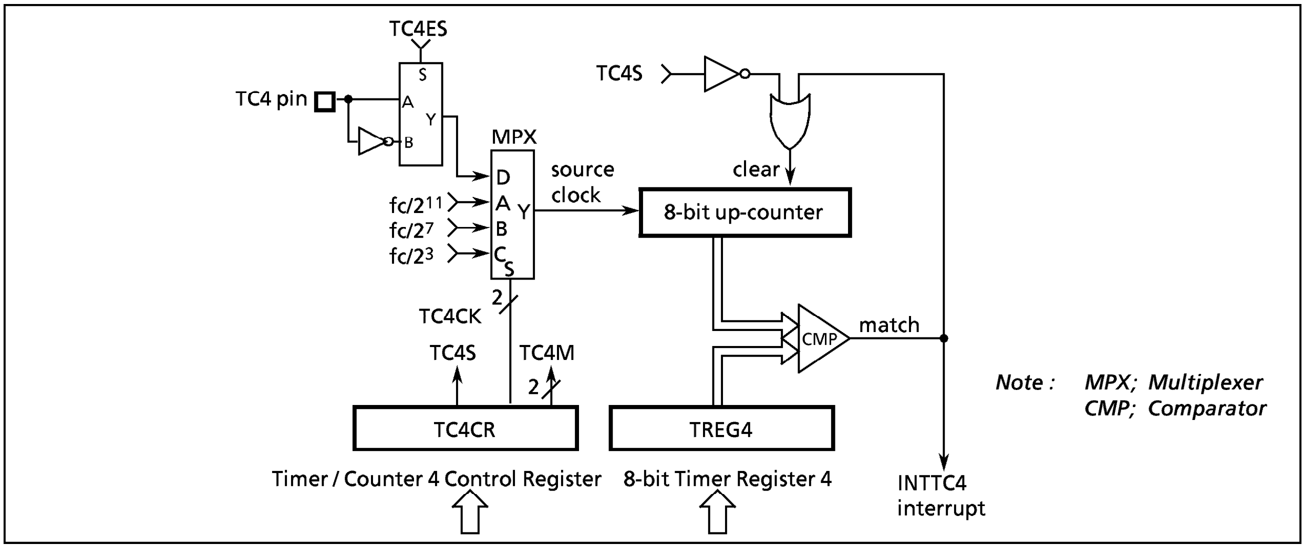
Source clock	Resolution (At fc = 8 MHz)	Maximum setting time (At fc = 8 MHz)
fc / 2^{11} [Hz]	256 μs	65.28 ms
fc / 2^{7}	16 μs	4.08 ms
fc / 2^{3}	1 μs	255 μs

(2) **Event Counter** Mode

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

2.7 8-bit Timer / Counter (TC4)

2.7.1 Configuration

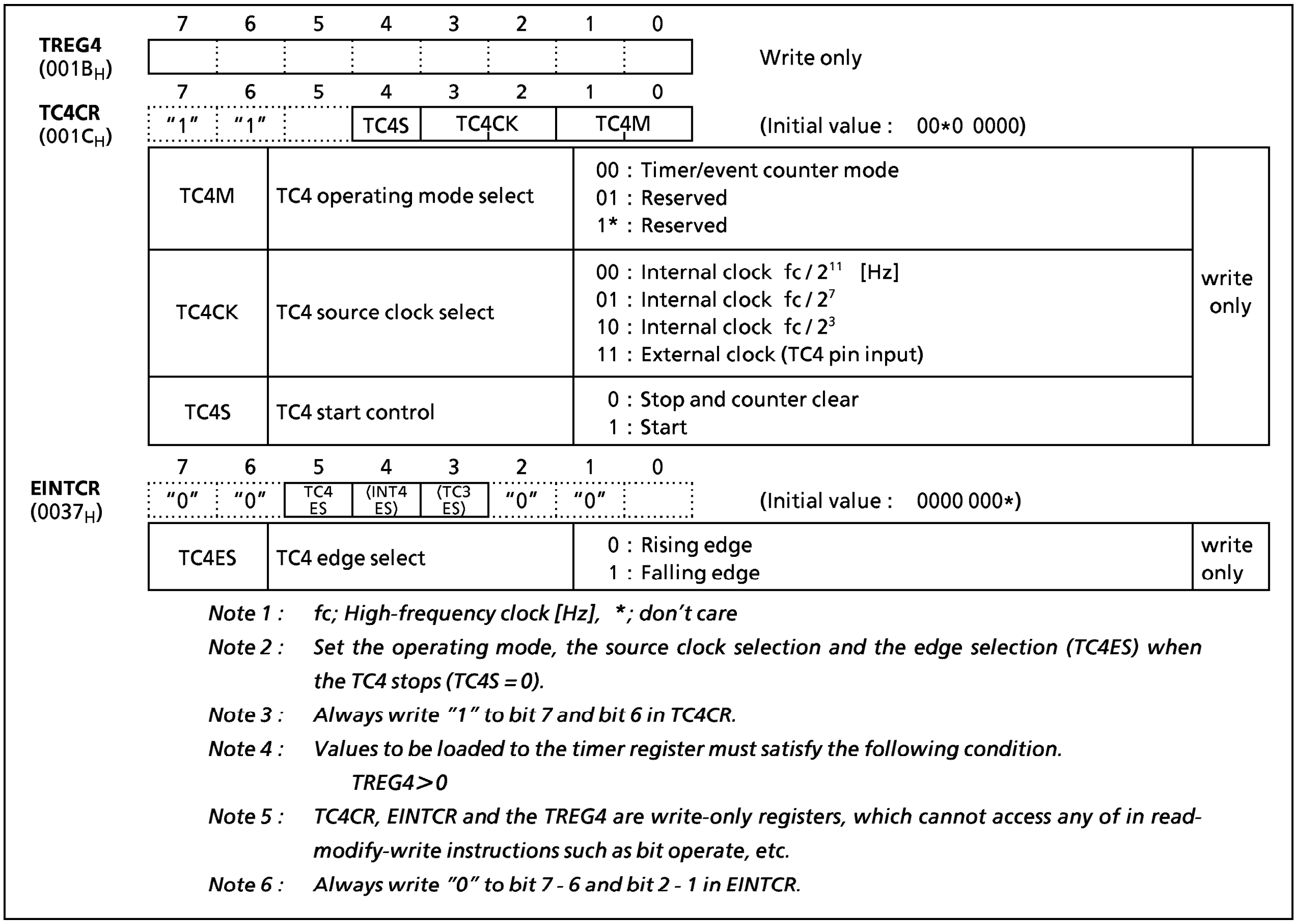


Note : MPX; Multiplexer
CMP; Comparator

Figure 2-19. Timer / Counter 4

2.7.2 Control

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


TREG4
(001B$_H$)
7 6 5 4 3 2 1 0
Write only
TC4CR
(001C$_H$)
"1" "1" TC4S TC4CK TC4M
(Initial value : 00*0 0000)
EINTCR
(0037$_H$)
"0" "0" TC4 ES (INT4 ES) (TC3 ES) "0" "0"
(Initial value : 0000 000*)

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	

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

*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, EINTCR and the TREG4 are write-only registers, which cannot access any of in read-modify-write instructions such as bit operate, etc.

Note 6 : Always write "0" to bit 7 - 6 and bit 2 - 1 in EINTCR.

Figure 2-20. 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.

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

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

(2) **Event Counter** Mode

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

2.8 Remote Control Signal Preprocessor / External Interrupt 3 Input Pin

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

2.8.1 Configuration

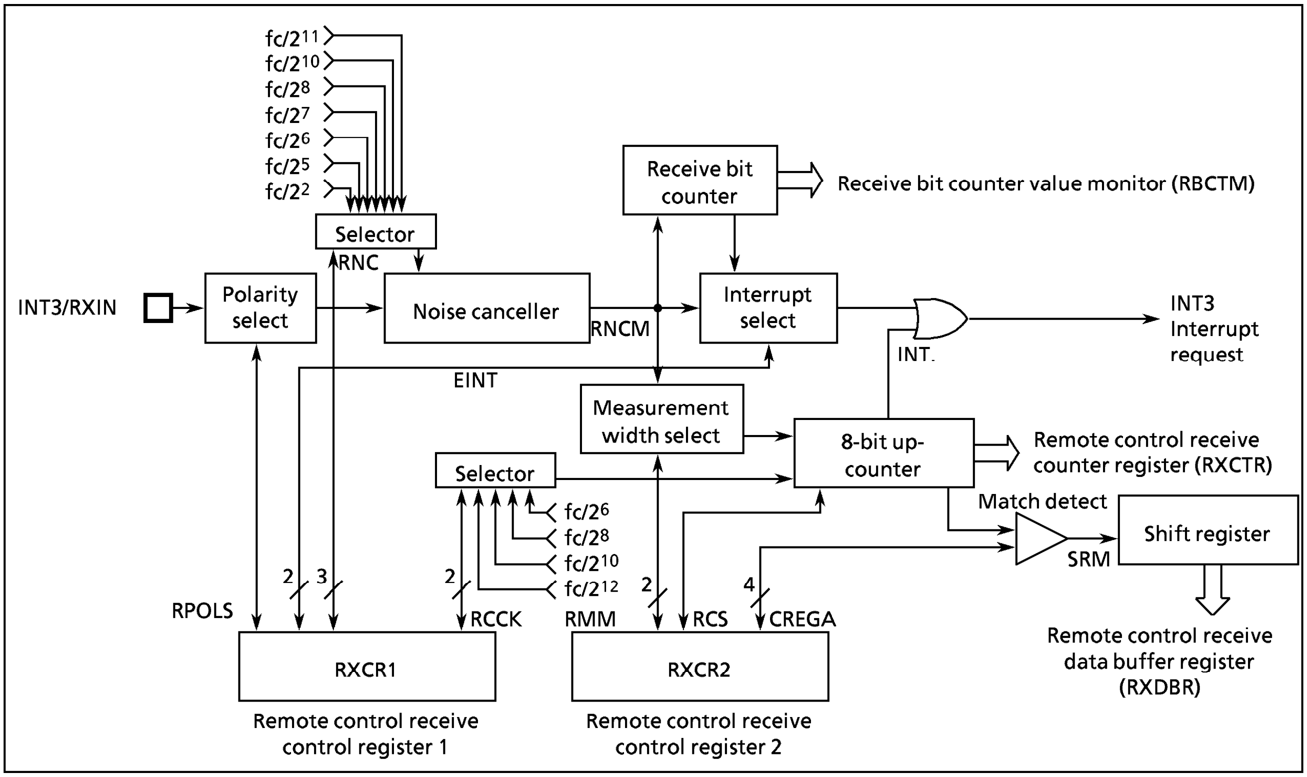


Figure 2-21. Remote Control Signal Preprocessor

2.8.2 Remote Control Signal Preprocessor Control

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

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

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

Remote control receive control register 1

RXCR1 (0FD0$_H$)	7	6	5	4	3	2	1	0
	RCCK		RPOLS	EINT		RNC		

(Initial value : 0000 0000)

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

Note1 : fc ; High-frequency clock [Hz].

Note2 : After reset, RPOLS does not change the set value in the receiving remote control signal. For setting interrupt edge and measurement data, use EINT and RMM.

Remote control receive control register 2

RXCR2 (0FD1$_H$)	7	6	5	4	3	2	1	0
	CREGA				RCS		RMM	

(Initial value : 0000 0*00)

CREGA	Setting of detect time for match with 8-bit up-counter upper 4 bits	Match detect time (Tth) = 16 × CREGA/RCCK [s]
		CREGA = 0_H to F_H
		Example : CREGA = 2_H, RCCK = fc/2^6 [Hz] , at fc = 8 MHz
		Tth = 256 [μs]
			Read/ Write

RCS	8-bit up-counter start control	0 : Stop and counter clear 1 : Start	
RMM	Measurement mode select (invalid when EINT = "10")	00 : 01 : Reter to table 2-5 10 : 11 :	

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

Note2 : When an interrupt source is set for rising/falling edge, low and high widths are forcibly measured separately.

Note3 : Set CREGA (O_H to F_H) before EINT sets to 8-bit receive end.

Figure 2-22. Remote Control Receive Control Register 1, 2

Remote control receive counter register

RXCTR
(0FD2$_H$)

7	6	5	4	3	2	1	0

Read Only
(Initial value : 0000 0000)

Remote control receive data buffer register

RXDBR
(0FD3$_H$)

7	6	5	4	3	2	1	0

Read Only
(Initial value : 0000 0000)

Remote control receive status register

RXSR
(0FD4$_H$)

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

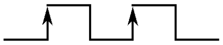
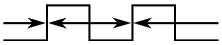
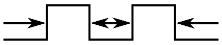
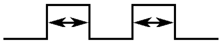
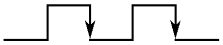
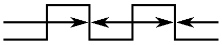
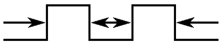
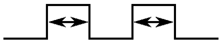
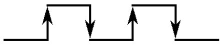
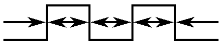
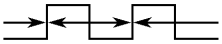
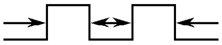
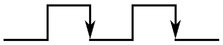
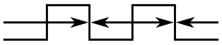
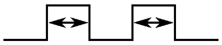
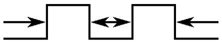
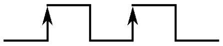
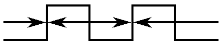
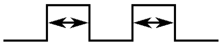
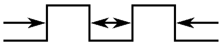
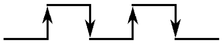
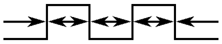
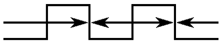
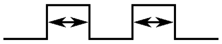
(Initial value : 0000 * 000)

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

*Note1 : * ; Don't care.*

Figure 2-23. Remote Control Receive Counter Register, Data Buffer Register, Status Register

Table 2-5. Combination of Interrupt Source and Measurement Mode

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

2.8.3 Noise elimination time setting

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

Table 2-6. Noise Elimination Time Setting

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

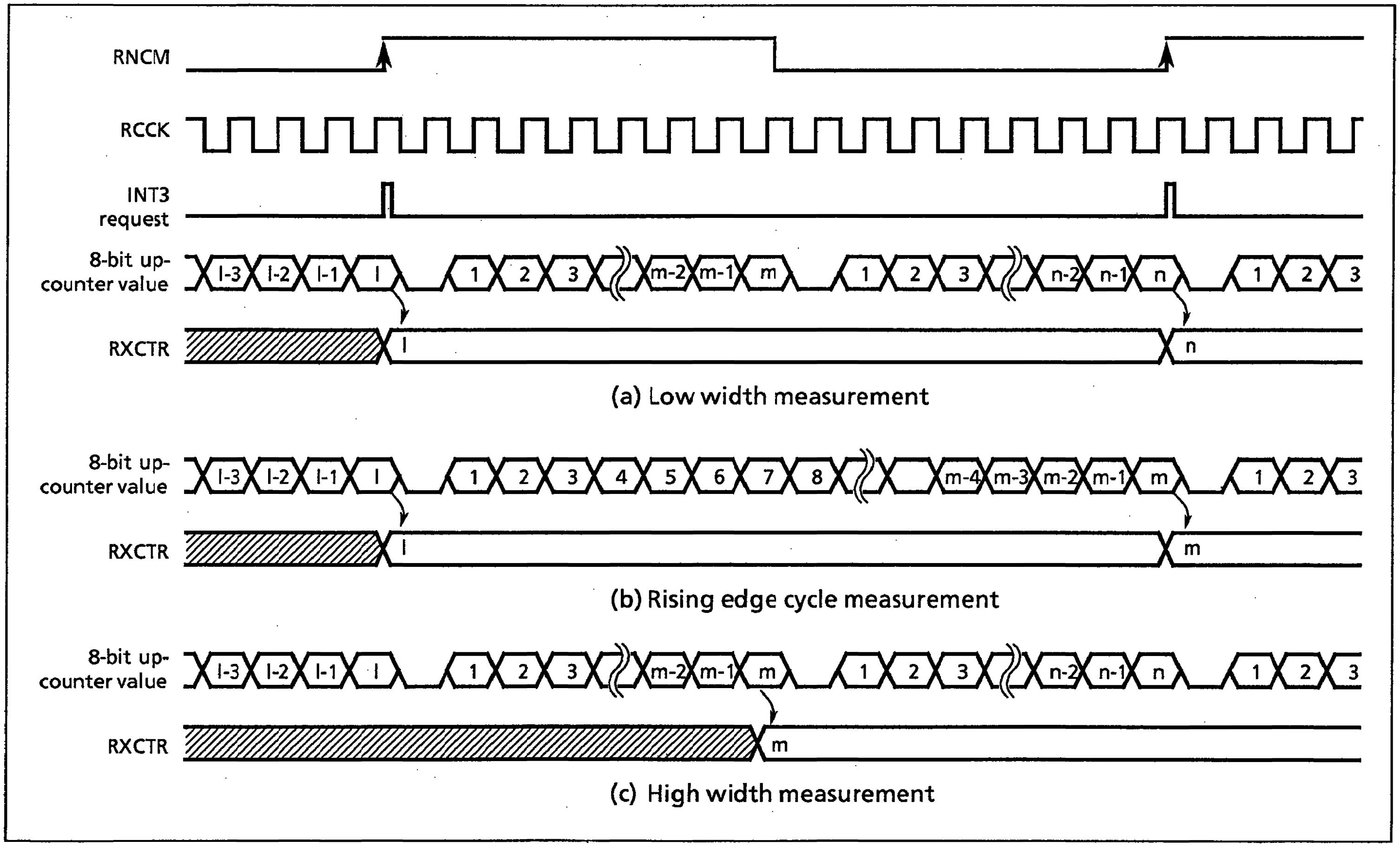
2.8.4 Operation

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

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

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

Setting RCS to "1" zero-clears OVFF.

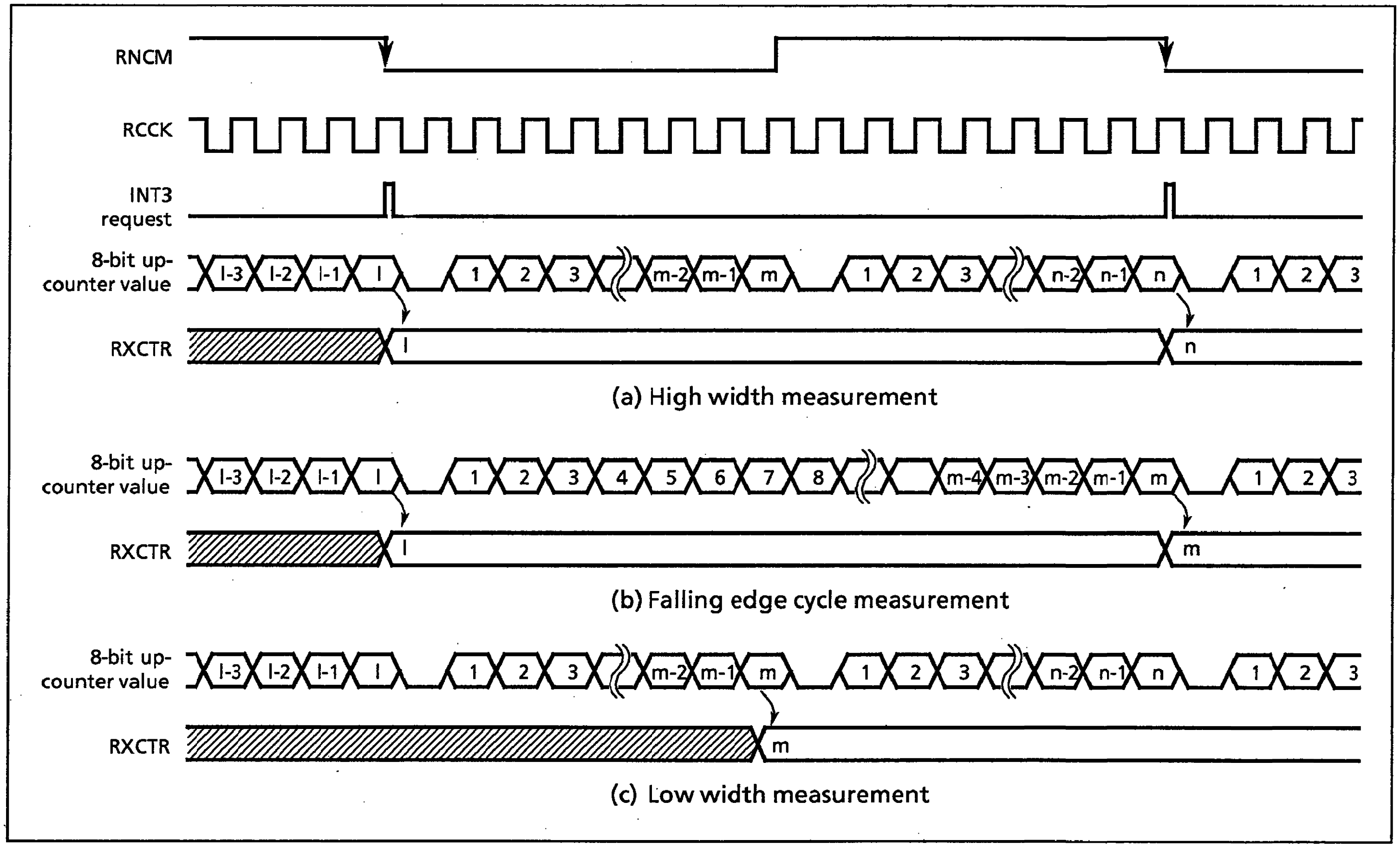


(a) Low width measurement

(b) Rising edge cycle measurement

(c) High width measurement

Figure 2-24. Rising Edge Interrupt Timing Chart (RPOLS = 0)

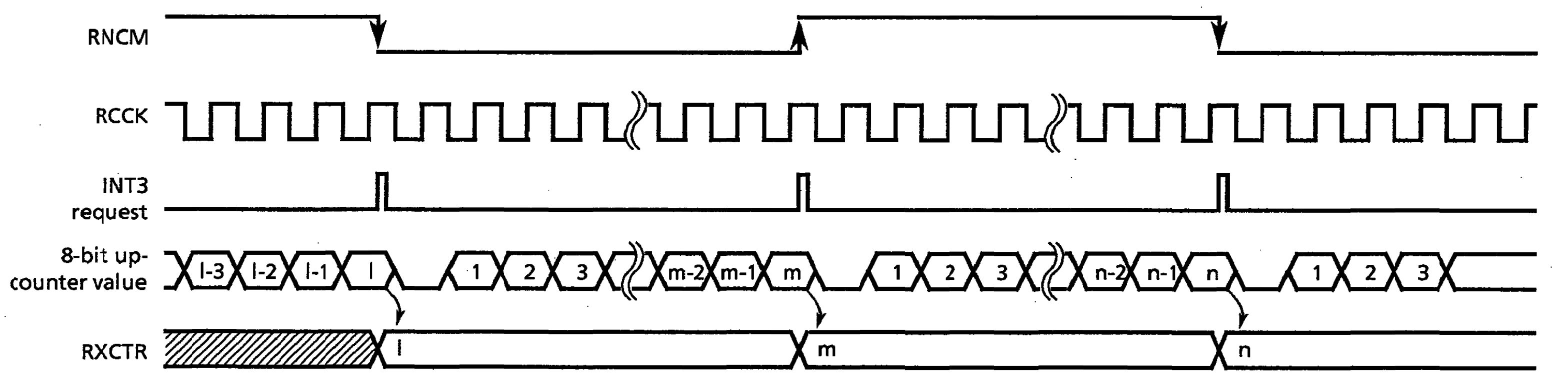


(a) High width measurement

(b) Falling edge cycle measurement

(c) Low width measurement

Figure 2-25. Falling Edge Interrupt Timing Chart (RPOLS = 0)

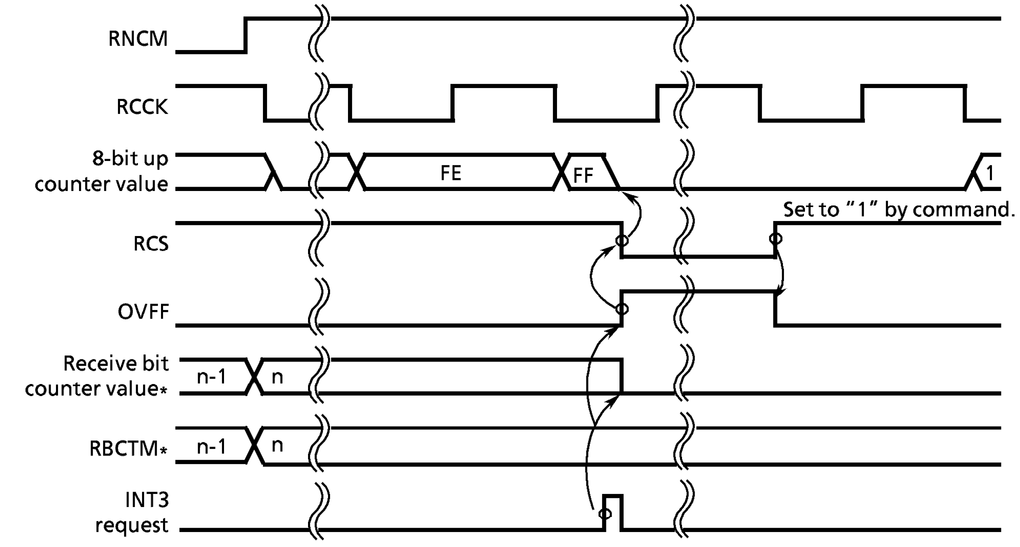


(a) High and low width measurement

Figure 2-26. Rising / Falling Edge Interrupt Timing Chart

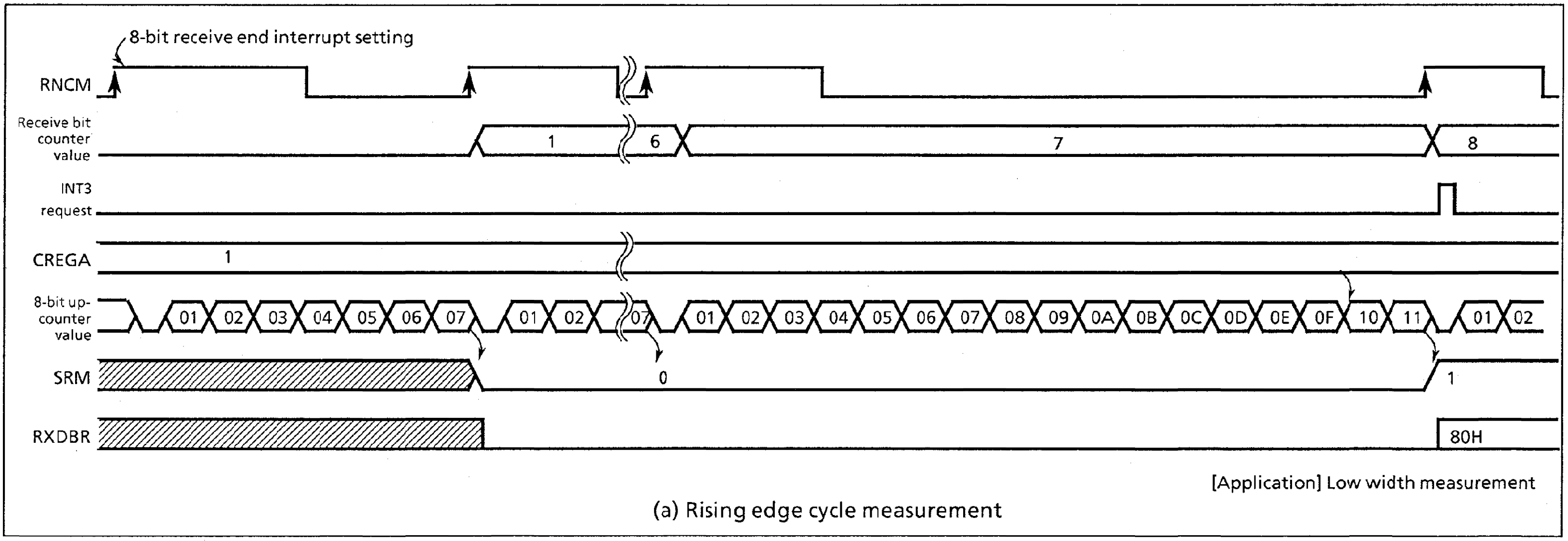
(2) 8-bit receive end interrupts and measurement modes

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



*Note : * ; Valid only when 8 bits are received.*

Figure 2-27. Overflow Interrupt Timing Chart



[Application] Low width measurement

(a) Rising edge cycle measurement

Figure 2-28. 8-bit Receive End Interrupt Timing Chart (RPOLS = 0)

Table 2-7. Count Clock for Remote Control Preprocessor Circuit

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

2.9 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 to CIN0) can be detected as 64-stage by setting reference voltage.

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

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

2.9.1 Configuration

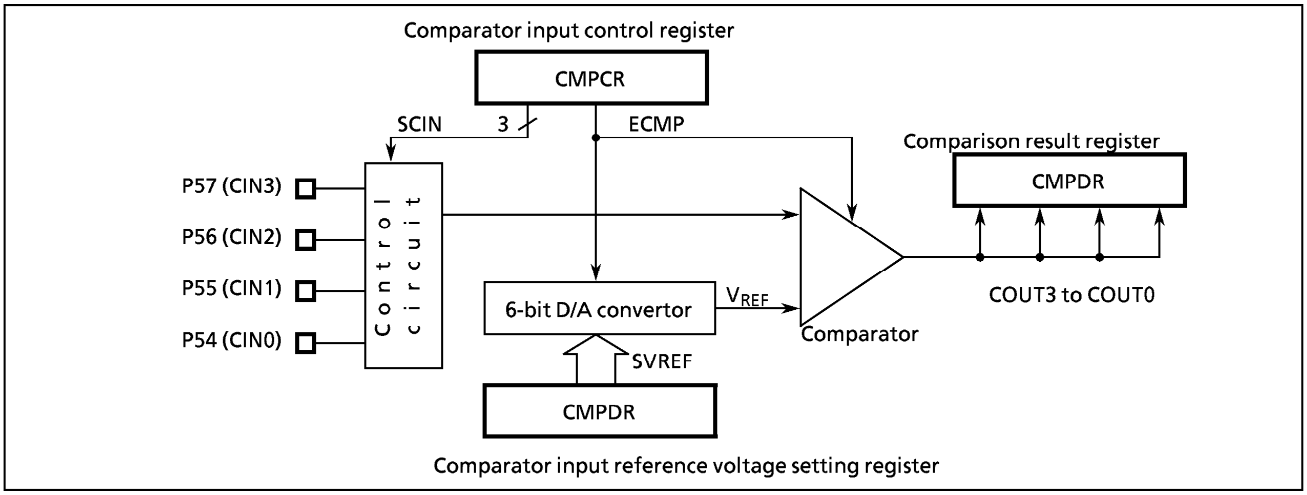


Figure 2-29. 6-bit A/D Conversion (Comparator) Inputs

2.9.2 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

	7	6	5	4	3	2	1	0	
CMPCR (000E$_H$)	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 : * ; don't care*
Note2 : The CMPCR is a write-only register and must not be used with any of the read-modify-write instructions such as bit operate, etc.

Comparator Input Data Register

	7	6	5	4	3	2	1	0	
CMPDR (000F$_H$)			SVREF						(Initial value : **00 0000)
					COUT3	COUT2	COUT1	COUT0	

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 to CIN0 pin analog input voltage with the reference voltage.	read only

*Note1 : * ; don't care*
Note2 : The SVREF as write-only bits and must not be used with any of the read-modify-write instructions such as bit operate, etc.
Note3 : Wich a read instruction for CMPDR, bits 7 to 4 are read in undetined data.

Figure 2-30. Comparator Input Control Register and Data Register

2.9.3 Function

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

$V_{REF} = V_{DD} \times (SVREF + 1) / 64$ [V] (SVREF = 0 to 63)

The number of comparator input channels is selected with SCIN (bits 2 to 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 to COUT0) should be read out at an interval equal to [number of channels × 2 machine cycles] after setting the reference voltage (V_{REF}). COUT3 to COUT0 are set to "1" if the input voltage (pins CIN3 to 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 to 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 to COUT0 are read out as "1" when not used as a comparator input. For example, bit 3 in CMPDR is always read out as "1" when SCIN = 010_B.

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

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

Table 2-8. Reference Voltage (at V_{DD} = 5 V)

SVREF						V_{REF} [V]
5	4	3	2	1	0	
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

2.10 Pulse Width Modulation Circuit Output

87CC31/H31 has a 14-bit resolution pulse width modulation (PWM) channel and 9 7-bit resolution PWM channels. D/A converter output can easily be obtained by connecting an external low-pass filter. PWM outputs are multiplexed with general purpose I/O ports as; P40 ($\overline{\text{PWM0}}$) to P47 ($\overline{\text{PWM7}}$), P50 ($\overline{\text{PWM8}}$), P51 ($\overline{\text{PWM9}}$). When these ports are used as PWM outputs,the corresponding bits of P4, P5 output latches should be set to "1".

2.10.1 Configuration

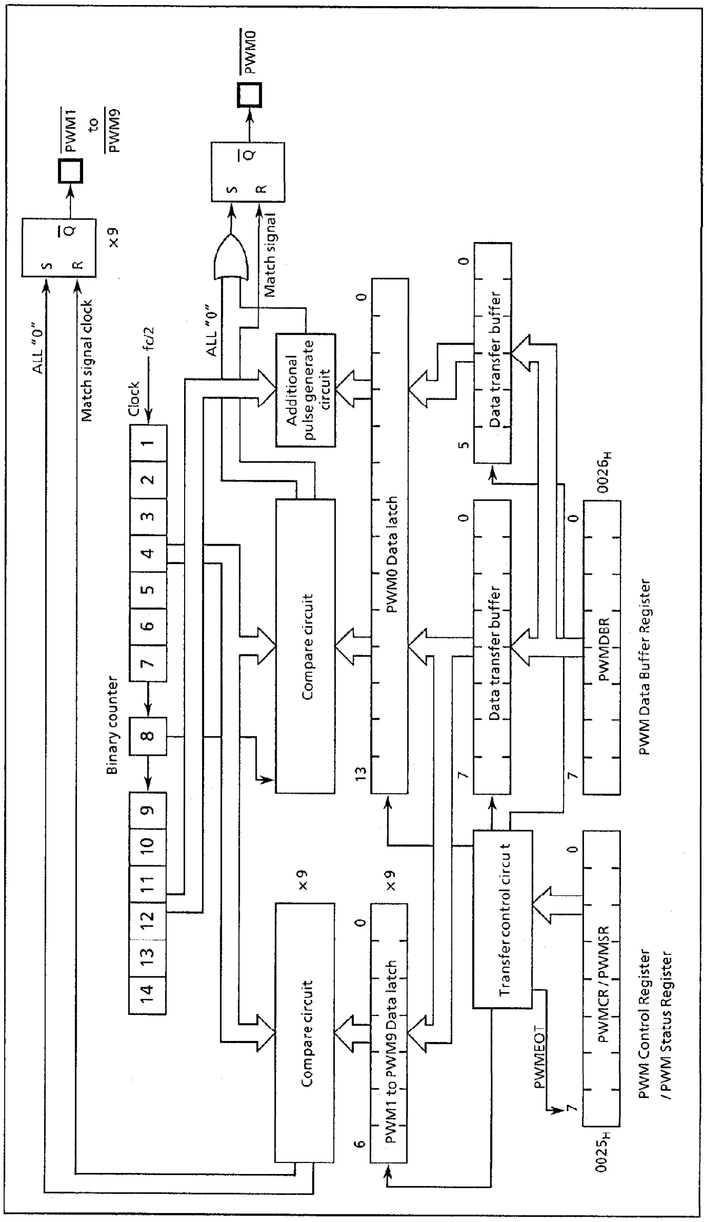


Figure 2-31. Pulse Width Modulation Circuit

2.10.2 PWM Output Wave Form

(1) $\overline{\text{PWM0}}$ 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 $\overline{\text{PWM0}}$. When the 8-bit data are decimal *n* ($0 \leqq n \leqq 255$), this pulse width becomes $n \times t_0$, where $t_0 = 2/fc$.

The lower 6-bit of 14 bit data are used to control the generation of additional to wide pulse in each T_S period. When the 6-bit data are decimal *m* ($0 \leqq m \leqq 63$), the additional pulse is generated in each of *m* periods out of 64 periods contained in a T_M period. The relationship between the 6 bits data and the position of T_S period where the additional pulse is generated is shown in Table 2-9.

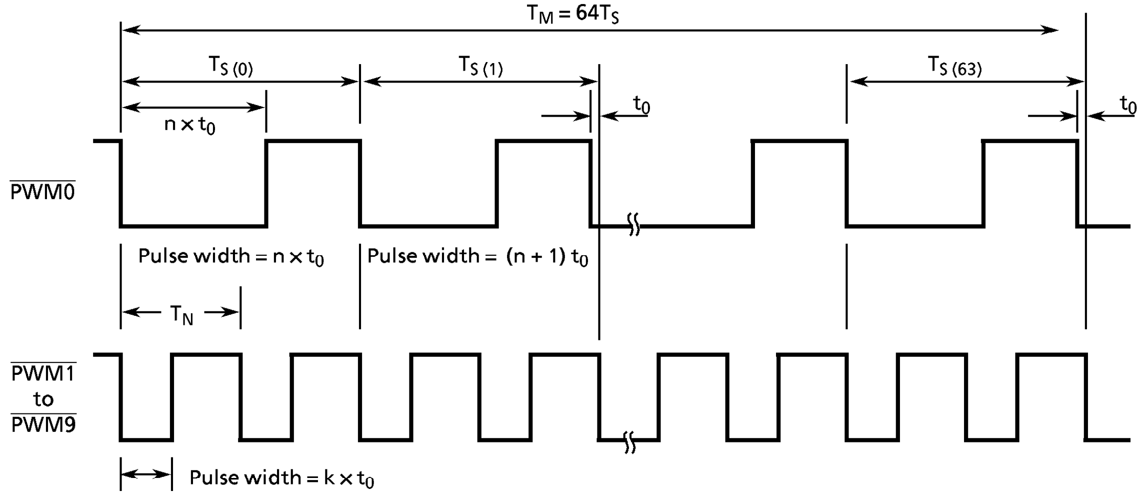
Table 2-9. Correspondence between 6 Bits Data and the Additional Pulse Generated TS Period

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

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

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

These are 7-bit resolution PWM outputs and one period is $T_N = 2^8/fc$ [s]. When the 7-bit data are decimal *k* ($0 \leqq k \leqq 127$), the pulse width becomes $k \times t_0$. The wave form is illustrated in Figure 2-32.


$T_M = 64T_S$
$T_{S\,(0)}$
$T_{S\,(1)}$
$T_{S\,(63)}$
$n \times t_0$
t_0
t_0
$\overline{\text{PWM0}}$
Pulse width = $n \times t_0$
Pulse width = $(n + 1)\, t_0$
T_N
$\overline{\text{PWM1}}$ to $\overline{\text{PWM9}}$
Pulse width = $k \times t_0$

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

Figure 2-32. PWM Output Wave Form

2.10.3 Control

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

PWM Control Register

	7	6	5	4	3	2	1	0	
PWMCR (0025H)					PWMDLS				(Initial value **** 0000)

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

PWM Status Register

	7	6	5	4	3	2	1	0	
PWMSR (0025H)	PWMEOT	"1"	"1"	"1"	"1"	"1"	"1"	"1"	(Initial value 0111 1111)

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

*Note : * ; don't care*

PWM Data Buffer Register

	7	6	5	4	3	2	1	0	
PWMDBR (0026H)									write only

Figure 2-33. PWM Control Register / PWM Status Register / PWM Data Buffer Register

(1) Programing of PWM Data

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

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

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

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

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

Note : at fc = 8 MHz

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

2.11 Pulse Output Circuit ($\overline{\text{PULSE}}$)

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

	7	6	5	4	3	2	1	0	
PULSECR (0027$_H$)						SPF			(Initial value **** 111*)

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

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

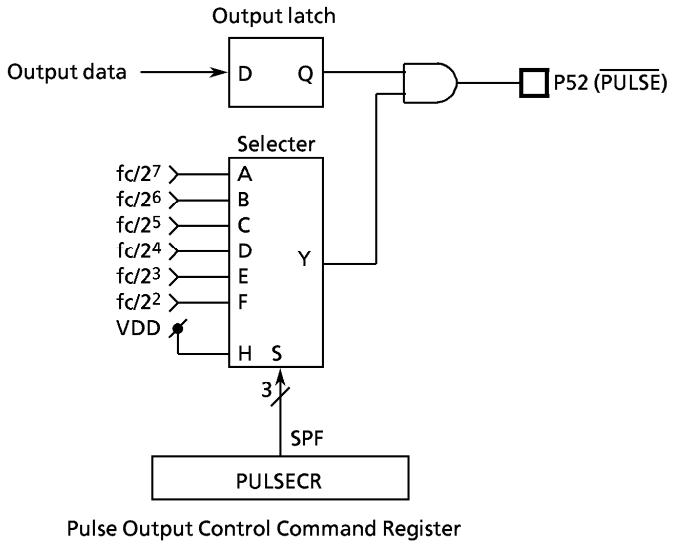


Figure 2-34. Pulse Output Circuit

2.12 On-Screen Display (OSD) Circuit

The TMP87CC31/H31 features a built-in on-screen display circuit used to display characters and symbols on the TV screen. 96 characters in any of 96 character fonts can be displayed in 24 character x 4 rows.

OSD circuit functions are as follows:

① Number of character fonts 96 (including the blank character)
② Number of display characters 96 (24 character x 4 rows).
③ Composition of a character 14 x 18 dots
④ Character sizes 3 (selectable line by line)
⑤ Display colors Character colors : 8 (selectable character by character),
Fringe color : 8 (selectable page by page)
Background color : 8 (selectable page by page)
⑥ Fringing function (for large, middle, and small characters)
⑦ Smoothing function (for large and middle characters)
⑧ Display position horizontal : 128 steps ; vertical : 256 steps
⑨ Full-raster blanking function
⑩ Blinking function
⑪ Reverse function
⑫ Reverse Blinking function
⑬ Window function

2.12.1 Configuration

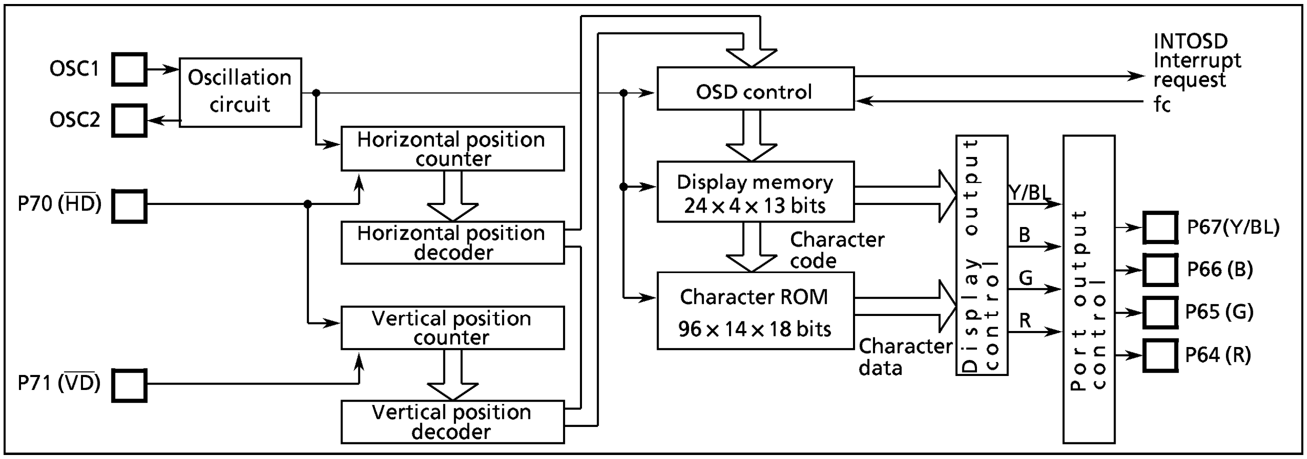


Figure 2-35. OSD Circuit

2.12.2 Character ROM and Display Memory

(1) Character ROM

The character ROM contains 96 character fonts. The user can set fonts as desired. The character ROM consists of 96 characters in 14 × 18 dots (character codes 00_H to $5F_H$). Each dot corresponds to one bit in the character ROM. When a bit in the character ROM is set to "1", the corresponding dot is displayed; if set to "0", the dot is not displayed. The start address in the character ROM corresponding to a character code is determined by the following expression:

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

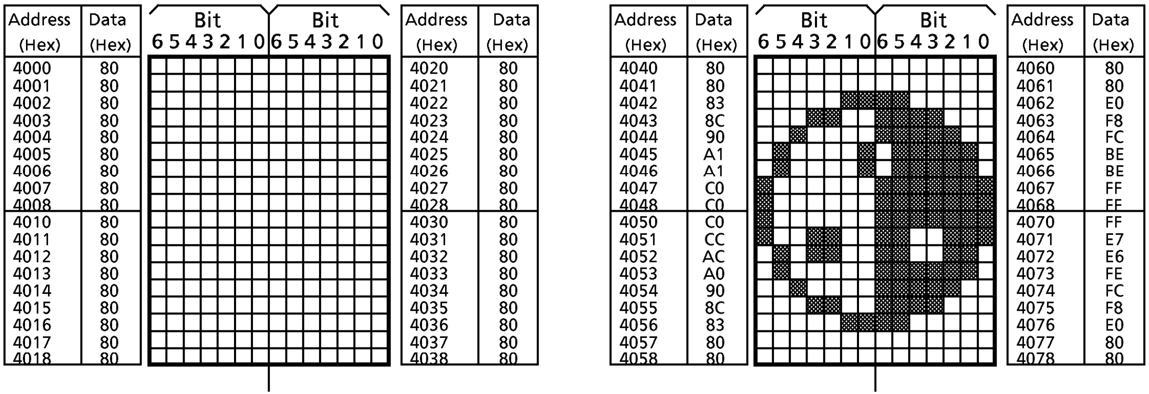
Since character code 00_H is used as blank character, the character font for this character code cannot be changed. Write "0" in the data of character code 00_H.

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

Figure 2-36. (a) shows an example of the character font configuration for the character code 00_H and 01_H, together with the ROM addresses and data.

Figure 2-36. (b) shows the character ROM dump list for these 2 character fonts .

Note 1 : CRA ; Character code (00_H to $5F_H$)
Note 2 : A data can not be read from the character ROM by software.
Note 3 : When ordering a mask, load the data to character ROM at address 4000_H to $57FF_H$.



Address (Hex)	Data (Hex)	Address (Hex)	Data (Hex)	Address (Hex)	Data (Hex)	Address (Hex)	Data (Hex)
4000	80	4020	80	4040	80	4060	80
4001	80	4021	80	4041	80	4061	80
4002	80	4022	80	4042	83	4062	E0
4003	80	4023	80	4043	8C	4063	F8
4004	80	4024	80	4044	90	4064	FC
4005	80	4025	80	4045	A1	4065	BE
4006	80	4026	80	4046	A1	4066	BE
4007	80	4027	80	4047	C0	4067	FF
4008	80	4028	80	4048	C0	4068	FF
4010	80	4030	80	4050	C0	4070	FF
4011	80	4031	80	4051	CC	4071	E7
4012	80	4032	80	4052	AC	4072	E6
4013	80	4033	80	4053	A0	4073	FE
4014	80	4034	80	4054	90	4074	FC
4015	80	4035	80	4055	8C	4075	F8
4016	80	4036	80	4056	83	4076	E0
4017	80	4037	80	4057	80	4077	80
4018	80	4038	80	4058	80	4078	80

(Character code 00_H) (Character code 01_H)

(a) Character font configuration

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

(b) ROM dump list

Figure 2-36. Character font configuration and ROM dump list

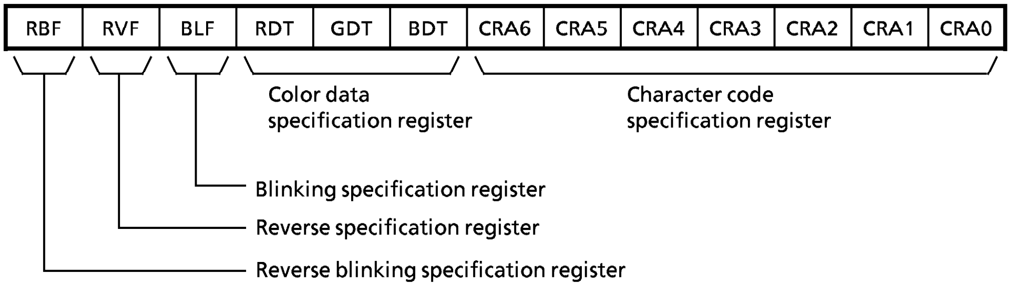
(2) Display memory

Each character out of the 96 characters displayed in 24 characters × 4 rows consists of 13 bits in the display memory. Five data items are written to the display memory: character code, color data, blinking specification, reverse specification, and reverse blinking specification. The display memory contents become unstable after the reset operation is released.

There are two modes for writing display data to the display memory. One mode is for writing all display data (character code, color data, blinking specification, reverse specification, and reverse blinking specification) simultaneously. The other mode is for changing either character code or character ornamentation data (color data, blinking specification, reverse specification, and reverse blinking specification). How the display data is written to the display memory is described in section 2.12.3 (18).

Display memory configuration

- Character code specification register (7 bits) ··· CRA6 to CRA0
- Color data specification register (3 bits) ··· RDT / GDT / BDT
- Blinking specification flag (1 bit) ··· BLF
- Reverse specification flag (1 bit) ··· RVF
- Reverse blinking specification flag (1 bit) ··· RBF



RBF	RVF	BLF	RDT	GDT	BDT	CRA6	CRA5	CRA4	CRA3	CRA2	CRA1	CRA0

Color data specification register
Character code specification register
Blinking specification register
Reverse specification register
Reverse blinking specification register

Figure 2-37. Display Memory Bit Configuration

row \ character	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	000	001	002	003	004	005	006	007	008	009	00A	00B	00C	00D	00E	00F	010	011	012	013	014	015	016	017
2	020	021	022	023	024	025	026	027	028	029	02A	02B	02C	02D	02E	02F	030	031	032	033	034	035	036	037
3	040	041	042	043	044	045	046	047	048	049	04A	04B	04C	04D	04E	04F	050	051	052	053	054	055	056	057
4	060	061	062	063	064	065	066	067	068	069	06A	06B	06C	06D	06E	06F	070	071	072	073	074	075	076	077

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

Figure 2-38. Display Memory Address Configuration

2.12.3 OSD circuit control

The OSD circuit is controlled by using the OSD control registers assigned to addresses $0F80_H$ to $0F9A_H$ in the data buffer register (DBR). For write to or read from the OSD control registers, see section 2.12.3 (19). The OSD control registers are used to set display start position, display character ornamentations (that is, fringing, smoothing, color data, character size, and etc.), display memory addresses, and character codes, and etc.

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

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

(1) Display position

The horizontal display start position can be set in 128 steps. The vertical display start positions can be specified for each line using 256 steps. The horizontal display start position is set with OSD control registers HS16 to HS10 (bits 6 to 0 in ORHS1). The vertical display start position for line one is set with VS17 to VS10 (in ORVS1). The vertical display start position of the 2nd-4th lines are determined by setting VS27 to VS20 ... VS47 to VS40 (ORVS2 to ORVS4) in the same way.

Horizontal display start position

Specification unit : Display page

Specification steps : 128

Specification horizontal display start position : Line 1 to 4 : HS16 to HS10

When FORS is "0" (Normal mode)

HS1 = (HS16 to HS10) $_H$ × 2T_{OSC} + 11T_{OSC} (Line1 to 4)

When FORS is "1" (Double frequency mode)

HS1 = (HS16 to HS10) $_H$ × T_{OSC} + 6.5T_{OSC} (Line1 to 4)

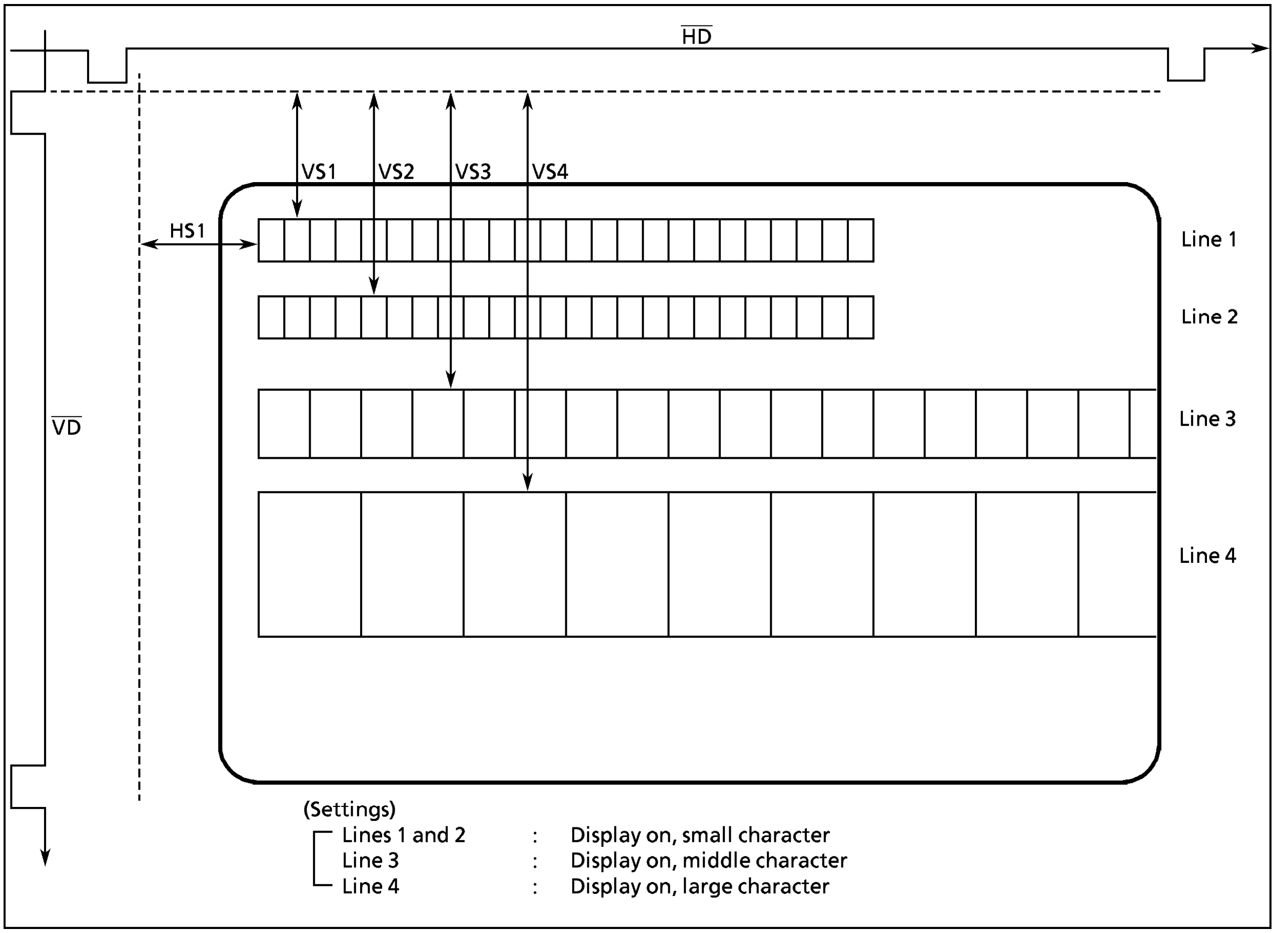
Note : T_{OSC} ; One cycle of OSC oscillation

Vertical display start position

Specification unit : Line

Specification steps : 256

Specification vertical display start position : Line1 : VS17 to VS10
Line2 : VS27 to VS20
Line3 : VS37 to VS30
Line4 : VS47 to VS40



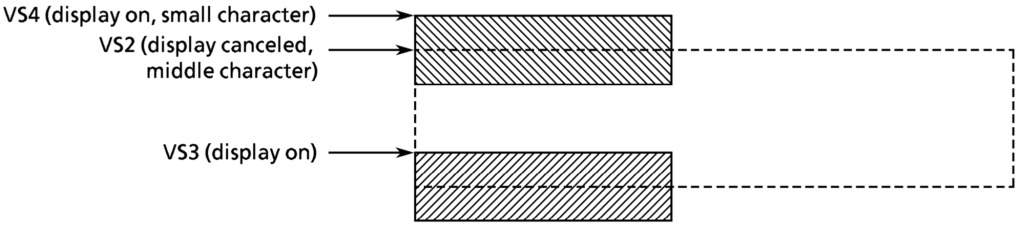
(Settings)
Lines 1 and 2 : Display on, small character
Line 3 : Display on, middle character
Line 4 : Display on, large character

Figure 2-39. TV Screen Image

When VDSMD is "0" (Normal mode)
Line n : VSn = (VSn7 to VSn0) $_H \times 2T_{HD}$ (n ; 1 to 4)

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

Note 1 : T_{HD} ; One cycle of $\overline{\text{HD}}$ signal
Note 2 : If display lines are overlapped each other, previous display line is enabled and next line is disabled. Set the vertical display start position not to overlap display lines.


VS4 (display on, small character)
VS2 (display canceled, middle character)
VS3 (display on)

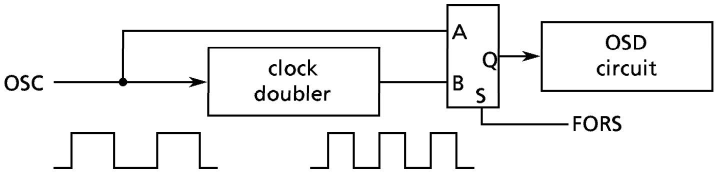
Occasion of overlapping

Note 3 : The line which is displayed off is managed as a small size character line. It is recommendable that its vertical display start position should be set out of TV screen.
Note 4 : Transfer the contents of vertical display start position registers into OSD circuit before the position of the scanning line coincides with their own vertical display start position.

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

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

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



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

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

(4) Character sizes and display on / off

Character size can be selected line by line from 3 sizes. And display on / off also can be set line by line. Small, middle and large character size and display on / off can be set with OSD control registers CS11, CS10...CS41, CS40 (ORCS4) in the OSD control registers.

Character sizes : 3 sizes (Small, middle and large)
Character size and display on / off specification unit : Line
Character size select/display on / off register (2 bits x 4)
Line 1: CS11 and CS10
Line 2: CS21 and CS20
Line 3: CS31 and CS30
Line 4: CS41 and CS40

Table 2-10. Character Size and Display On/Off Specifications (n ; 1 to 4)

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

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

Table 2-11. Dot and Character Sizes

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

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

(5) Smoothing function

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

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

(6) Fringing function

The fringing function is used to display a character with a fringe width is 1/2 dot in a different color from that of the character. For small characters, fringe width is 1 dot. When a character is displayed with the maximum of 14 vertical dots and 18 horizontal dots, the fringe exceeds right and left, top, and bottom of the character display area. The exceeded fringe can be displayed; however, display characters have higher priority to fringe horizontally.

Fringing is enabled for each line by setting EFR1 to EFR4 (bit 3 to 0 in OREFR) in the OSD control register to "1".

A color for fringe is specified common to all lines using OSD control registers, RFDT, GFDT, and BFDT, (bits 2 to 0 in ORBK).

Fringing specification unit: Line
Fringing enable register (1 bit x 4) ··· EFRn (n:1 to 4) (bit 3 to 0 in OREFR)
"0" ··· Disable fringing
"1" ··· Enable fringing

Fringe color specification unit : Display page
Fringe color register (3 bits) ··· RFDT, GFDT, BFDT

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

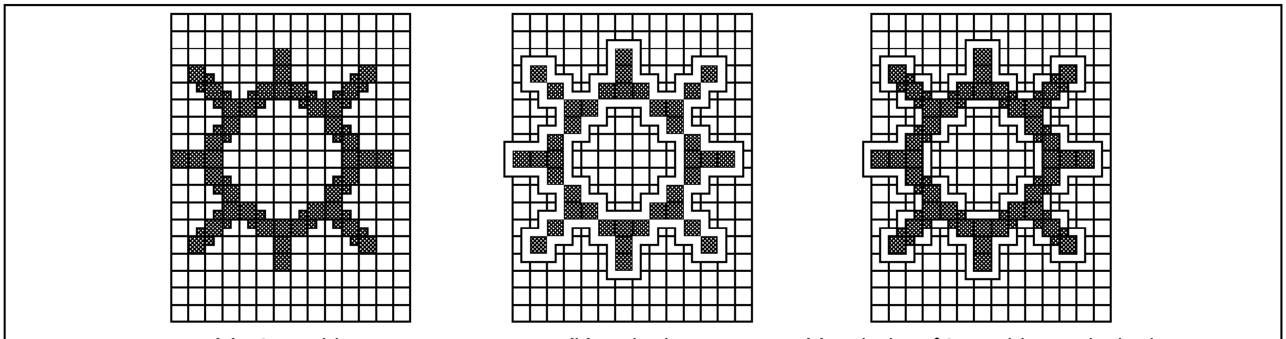

(a) Smoothing (b) Fringing (c) Priority of Smoothing and Fringing

Figure 2-40. Smoothing / Fringing / Priority of Smoothing and Fringing

(7) Background color function

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

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

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

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

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

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

(8) Full-raster blanking function

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

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

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

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

(9) Reverse function

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

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

(10) Reverse blinking function

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

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

Table 2-12. Display Mode

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

* ; don't care

(11) Blinking function

Blinking function is used to blink display characters.
When BKMF is "1", characters specified for blinking by BLF are not displayed. (Space is displayed. That is, if the background color function is used, the background color is not disappeared.)

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

(12) Character

Characters: 96 (including blank character)

Character specification register (7 bits) ··· CRA6 to CRA0 (bits 6 to 0 in ORCRA)
Character code "00_H" ··· Blank character
Character code "01_H" to "$5F_H$" ··· User programmable by character ROM

(13) Character color

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

Table 2-13. Character Color

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

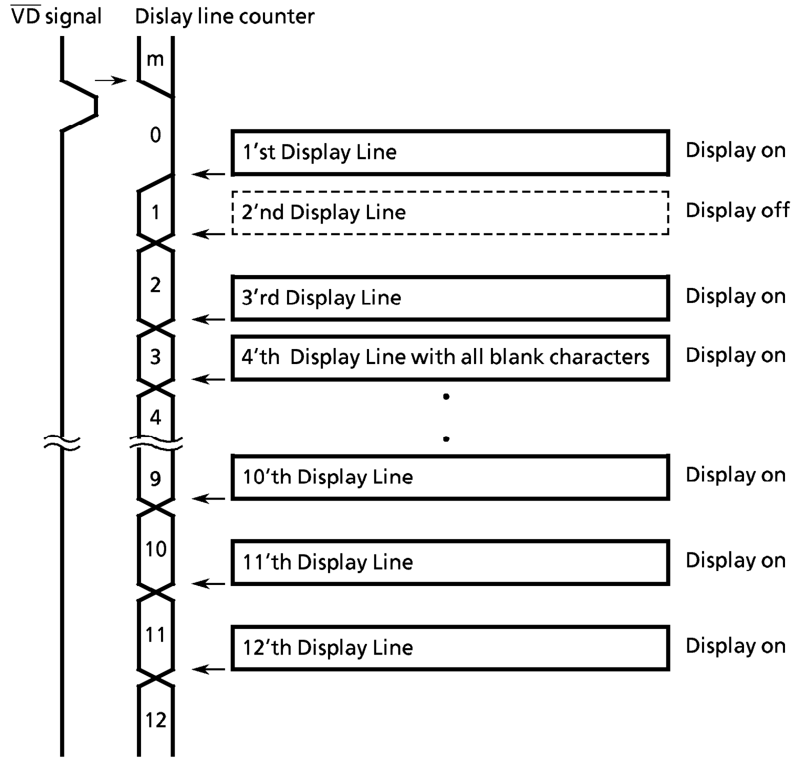
(14) OSD interrupt

1) Display line counter

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

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

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



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

Figure 2-41. Display Line Counter

2) Interrupt generator circuit

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

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

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

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

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

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

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

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

to

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

(15) P6 port output select function

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

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

(16) OSD pin output polarity control function

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

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

Table 2-16. Control of OSD Output Polarity

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

(17) Y/BL signal select

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

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

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

(18) Writing display data to the display memory

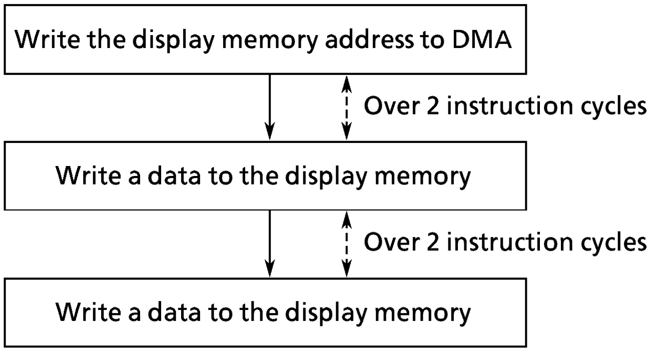
Data are written to the display memory using DMA8 to DMA0, CRA6 to CRA0, RDT, GDT, BDT, BLF, RBF, RVF, and MBK registers.

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

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

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

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


Write the display memory address to DMA
Over 2 instruction cycles
Write a data to the display memory
Over 2 instruction cycles
Write a data to the display memory

Example : Setting a character code (20_H) to the display memory (Address : 020_H to 021_H)

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

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

LD DE,ORDMA          ; Set lower 8-bit addresses to DMA7 to 0
LD (DE), 20H
LD DE, ORETC         ; Set the most upper address to DMA8 and set MBK to "0"
LD (DE),00000000B

NOP                  ; Insert 2 instruction cycles
NOP

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

NOP                  ; Insert2 instruction cycles
NOP

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

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

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

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

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

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

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

④ Write character code to CRA6 to CRA0.
At this time, character code is transferred to the display memory together with the character ornamentation data which is written in ③ and DMA8 to DMA0 are automatically incremented.

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

① Write lower 8-bit addresses of display memory to DMA7 to DMA0 (in ORDMA).
② Write the most upper address of display memory to DMA8 (bit 0 in ORETC) and clear MBK (bit 1 in ORETC) to "0".

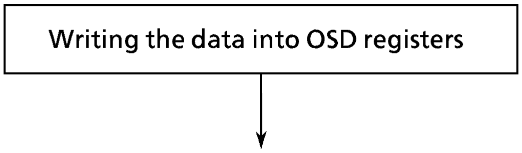
Note: It is necessary to write all bits of display memory address, writing DMA8 after DMA7 to DMA0, when writing display address, and repeat this sequence.

③ Write character ornamentation data (blinking, reverse, reverse blinking, and color data) to RDT, GDT, BDT, BLF, RVF, and RBF or write character code to CRA6 to CRA0.
At this time, written data are transferred to the display memory and DMA8 to DMA0 are automatically incremented.

(19) OSD control register write / read
The address of the OSD control registers are assigned to the DBR area.
To write or to read from the OSD control registers, the method is the same as for accessing ordinary DBR registers.
The written data are transferred to the OSD circuit at the end point of the scanning line without display by setting RGWR register to "1" and become valid. And, be able to write value of OSD control register after RGWR flag is cleared to "0".

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

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



```
            LD   A,(TEMP_ORDON)         ; Set bit 2 of work-area to "1"
            SET  A.2

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

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

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

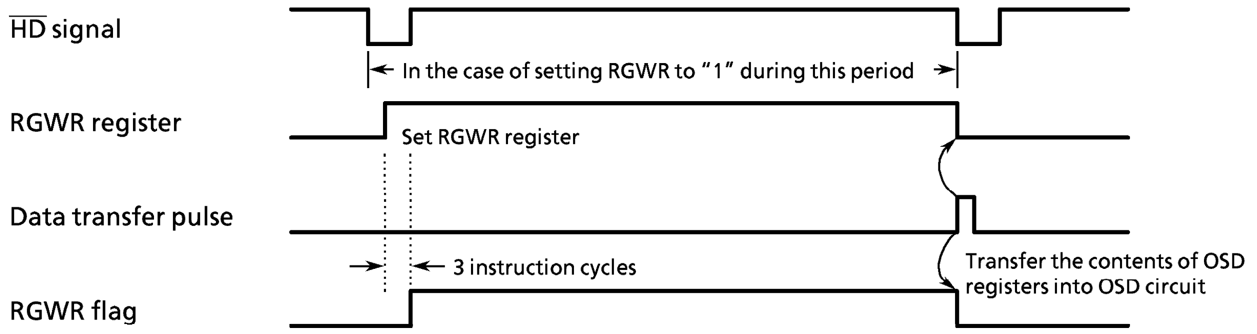
```
            LD   HL,ORDON
 CHECK_RGWR_FLAG:
            TEST (HL).2                 ; Checking RGWR flag until RGWR flag is "0"
            JR   F,CHECK_RGWR_FLAG
```

Writing the data into OSD registers

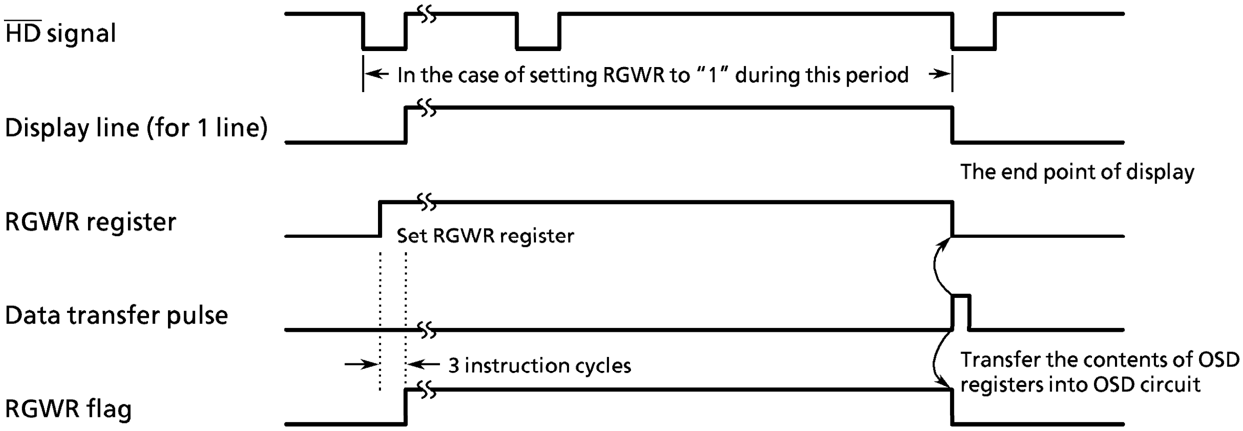
```
            LD   A,(TEMP_ORDON)         ; Set bit 2 of work-area to "1"
            SET  A.2

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

            NOP                         ; Insert 3 instruction cycles
            NOP
            NOP
```

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

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


$\overline{\text{HD}}$ signal
In the case of setting RGWR to "1" during this period
RGWR register
Set RGWR register
Data transfer pulse
3 instruction cycles
Transfer the contents of OSD registers into OSD circuit
RGWR flag

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


$\overline{\text{HD}}$ signal
In the case of setting RGWR to "1" during this period
Display line (for 1 line)
The end point of display
RGWR register
Set RGWR register
Data transfer pulse
3 instruction cycles
Transfer the contents of OSD registers into OSD circuit
RGWR flag

For registers (DMA8 to DMA0, CRA6 to CAR0, RDT, GDT, BDT, BLF, RBF, RVF, MBK) used for updating the display memory, P67DS to P64DS, YBLCS, BKMF, RBMF, ESMZ, VDSMD, FORS, and RGWR, the data become valid as soon as they are written.

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

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

(20) Display on / off
Function used to display a line specified for on / off display.

Display on / off specification unit : Page

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

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

(21) Window function

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

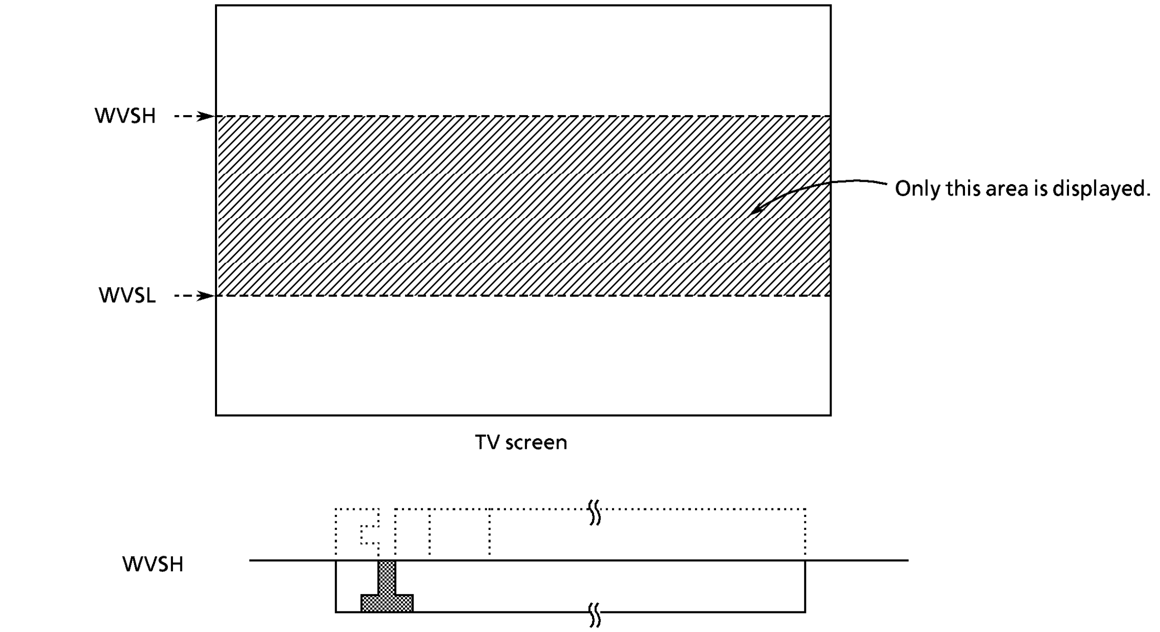
Window specification unit: Display page
Window function enable specification register (1 bit) ··· EWDW (bit 1 in ORDON)
"0" ··· Disable window function
"1" ··· Enable window function

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

Note 1 : T_{HD} ; One cycle of $\overline{\text{HD}}$ signal
Note 2 : WVSL > WVSH ≧ "1"
Note 3 : Modify the value of window upper and lower limit register as follows ;

1. *When $WVSH_{NEW}$ ≦ $WVSH_{OLD}$*
   Finish of transfer the new value, during $\overline{\text{VD}}$ signal is low or before the position of the scanning line coincides with $WVSH_{NEW}$.
2. *When WVSL > $WVSH_{NEW}$ > $WVSH_{OLD}$*
   Finish of transfer the new value, during $\overline{\text{VD}}$ signal is low or before the position of the scanning line coincides with $WVSH_{OLD}$.
3. *When $WVSL_{NEW}$ ≦ $WVSL_{OLD}$*
   Finish of transfer the new value, during $\overline{\text{VD}}$ signal is low or before the position of the scanning line coincides with $WVSL_{NEW}$.
4. *When $WVSL_{NEW}$ > $WVSL_{OLD}$*
   Finish of transfer the new value, during $\overline{\text{VD}}$ signal is low or before the position of the scanning line coincides with $WVSL_{OLD}$.

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



(22) OSD Control Registers

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

ORHS1 (0F80$_H$)

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

HS16 to 10	Horizontal display start position	Write only

ORVS1 (0F81$_H$)

7	6	5	4	3	2	1	0	
VS17	VS16	VS15	VS14	VS13	VS12	VS11	VS10	(Initial value 0000 0000)

⋮

ORVS4 (0F84$_H$)

VS47	VS46	VS45	VS44	VS43	VS42	VS41	VS40	(Initial value 0000 0000)

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

(n = 1 to 4)

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

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

ORCS4 (0F8D$_H$)

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

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

(n = 1 to 4)

OREFR
(0F90$_H$)

7	6	5	4	3	2	1	0	
				EFR4	EFR3	EFR2	EFR1	(Initial value **** 0000)

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

(n = 1 to 4)

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

ORP6DS
(0F91$_H$)

7	6	5	4	3	2	1	0	
P67DS	P66DS	P65DS	P64DS					(Initial value 0000 ****)

P67DS to P64DS	P6 port output select	0 : Port contents output 1 : R, G, B, Y/BL signal output	Write only

ORWVSH
(0F92$_H$)

7	6	5	4	3	2	1	0	
WVSH7	WVSH6	WVSH5	WVSH4	WVSH3	WVSH2	WVSH1	WVSH0	(Initial value 0000 0000)

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

ORWVSL
(0F93$_H$)

7	6	5	4	3	2	1	0	
WVSL7	WVSL6	WVSL5	WVSL4	WVSL3	WVSL2	WVSL1	WVSL0	(Initial value 0000 0000)

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

ORBK
(0F94$_H$)

7	6	5	4	3	2	1	0	
EBKGD	EXBL	RBDT	GBDT	BBDT	RFDT	GFDT	BFDT	(Initial value 0000 0000)

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

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

ORIRC
(0F95$_H$)

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

(Initial value 0000 0000)

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

ORIRC
(0F95$_H$)

7	6	5	4	3	2	1	0
				DCTR			

(Initial value **** 0000)

DCTR	Display line counter	Read only

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

ORETC
(0F96$_H$)

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

(Initial value 0000 0000)

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

Note1 : Clear "0" to bit 2 in ORETC
Note2 : It is necessary to write all bits of display memory address, writing DMA8 after DMA7 to DMA0, when writing display address, and repeat this sequence.

ORDMA
(0F97$_H$)

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

(Initial value 0000 0000)

DMA7 to 0	Display memory address	Write only

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

ORDSN
(0F98$_H$)

7	6	5	4	3	2	1	0
		RBF	RVF	BLF	RDT	GDT	BDT

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

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

	7	6	5	4	3	2	1	0	
ORCRA (0F99_H)		CRA6	CRA5	CRA4	CRA3	CRA2	CRA1	CRA0	(Initial value **** ****)

CRA6 to 0	Character code	Write only

	7	6	5	4	3	2	1	0	
ORDON (0F9A_H)	"0"	"0"	"0"	FORS	"1"	RGWR	EWDW	EDISP	(Initial value ***0 0000)

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

*Note1 : * ; don't care*

Note2 : The data written to OSD control registers except the register followed table is transmitted to OSD circuit by setting RGWR (bit2 in ORDON) to "1". RGWR is cleared to "0" automatically after the transfer is completed.

P67DS, P66DS, P65DS, P64DS	bits 7 to 4 in ORP6DS
YBLCS, BKMF, RBMF, ESMZ, VDSMD, MBK, DMA8	bits 7 to 3, 1 to 0 in ORETC
DMA7, DMA6, DMA5, DMA4, DMA3, DMA2, DMA1, DMA0	bits 7 to 0 in ORDMA
RBF, RVF, BLF, RDT, GDT, BDT	bits 5 to 0 in ORDSN
CRA6, CRA5, CRA4, CRA3, CRA2, CRA1, CRA0	bits 6 to 0 in ORCRA
FORS, RGWR	bits 4 and 2 in ORDON

Note3 : When EWDW is cleared to "0", clear EWDW to "0" independent of the value which this register has been set from detecting the rising edge of $\overline{\text{HD}}$ signal by software until the falling edge of $\overline{\text{HD}}$ signal.

Note4 : Write "1" to bit 3 of ORDON when writing to ORDON.

Note5 : Do not clear RGWR register to "0". If RGWR register is cleared to "0", the contents of OSD control registers may be transferred to OSD circuit at unexpected timing.

	7	6	5	4	3	2	1	0	
ORDON (0F9A_H)				FORS		RGWR	EWDW	EDISP	(Initial value ***0 *000)

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

Notice when developing a program of TMP87CC31/H31

When developing a program of 87CC31/H31 by using an OTP (87PM36) and an emulator (BM87CM37N0A), it is necessary to take notice as follows for emulating the operation of 87CC31/H31 with them.

On-Screen-Display Circuit

(1) Setting ORVFn (Vertical Display Start Position) of line 5 to line 12
Set "FF$_H$" into ORVF5 to ORVF12. If these registers are set other value or have an initial value, cannot emulate the operation of 87CC31/H31 with an OTP and an emulator.

									OTP/Emulator Initial value
ORVS5 (0F85$_H$)	VS57	VS56	VS55	VS54	VS53	VS52	VS51	VS50	(0000 0000)
ORVS12 (0F8C$_H$)	VS127	VS126	VS125	VS124	VS123	VS122	VS121	VS120	(0000 0000)

87CC31/H31 does not have ORVF5 to ORVF12. But the operation of OSD interrupt is similar with setting "FF$_H$" into ORVF5 to ORVF12.

(2) Setting CSn (Character size and display on/off) of line 5 to line 12
Set "00$_H$" (Initial value) into ORCS8 and ORCS12. If these registers are set other value, cannot emulate the operation of 87CC31/H31 with an OTP and an emulator.

					OTP/Emulator Initial value
ORCS8 (0F8E$_H$)	CS8	CS7	CS6	CS5	(0000 0000)
ORCS12 (0F8F$_H$)	CS12	CS11	CS10	CS9	(0000 0000)

(3) Setting EFRn (Fringing Specification) of line 5 to line 12
Set "0" (Initial value) into OREFR (EFR8 to EFR5) and ORP6DS (EFR12 to EFR9). If these registers are set other value, cannot emulate the operation of 87CC31/H31 with an OTP and an emulator.

									OTP/Emulator Initial value
	↓	↓	↓	↓					
OREFR (0F90$_H$)	ERF8	ERF7	ERF6	ERF5	(ERF4)	(ERF3)	(ERF2)	(ERF1)	(0000 0000)
					↓	↓	↓	↓	
ORP6DS (0F91$_H$)	(P67DS)	(P66DS)	(P65DS)	(P64DS)	ERF12	ERF11	ERF10	ERF9	(0000 0000)

(4) Character codes
Set a character code within 00$_H$ to 5F$_H$ into the display memory. And set "FF$_H$" into the character ROM of an OTP and an emulator at 5800$_H$ to 5FFF$_H$. If these character ROM are set other value, cannot emulate the operation of 87CC31/H31 with an OTP and an emulator.
If the display memory of 87CC31/H31 is written a character code within 60$_H$ to 7F$_H$, its font data is FF$_H$.

	TMP87CC31/H31N	TMP87CM36N
Address	4000$_H$ - 57FF$_H$ 14 × 18 × 96 characters	4000$_H$ - 5FFF$_H$ 14 × 18 × 128 characters

Serial Bus Interface circuit

87CC31/H31 does not have a serial bus interface circuit. Do not set a data into the serial bus interface control registers since OTP and emulator have it. And clear “0” into EFR9 in EIR.

Serial Bus Interface Control Register 1	SBICR1	0020_H
Serial Bus Interface Control Register 2	SBICR2	0023_H
Serial Bus Interface Data Buffer Register	SBIDBR	0021_H
I^2Cbus Address Register	I2CAR	0022_H
Serial Bus Interface Status Register	SBISR	0023_H

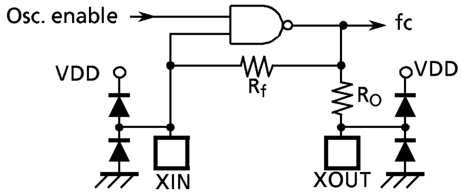
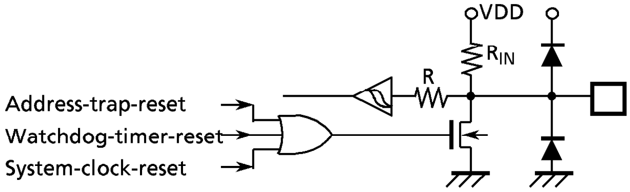
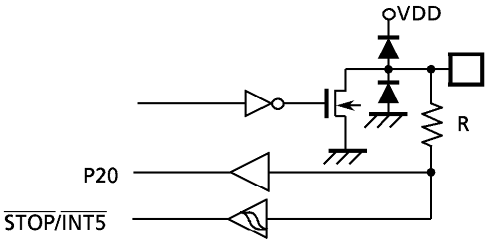
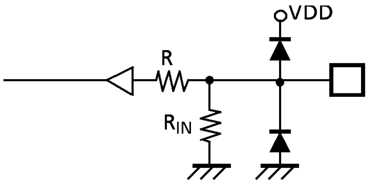
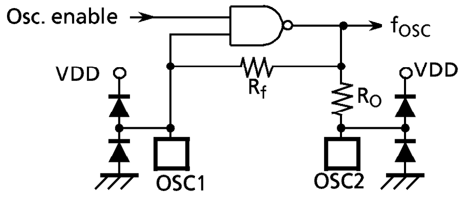
Table 1-2. Interrupt Sources

Interrupt Source		Enable Condition	Interrupt Latch	Vector Table Address	Priority
Internal/ External	(Reset)	Non-Maskable	—	$FFFE_H$	High 0
Internal	INTSW (Software interrupt)	Pseudo non-maskable	—	$FFFC_H$	1
Internal	INTWDT (Watchdog Timer interrupt)		IL_2	$FFFA_H$	2
	reserved	IMF = 1, INT0EN = 1	IL_3	$FFF8_H$	3
Internal	INTTC1 (16-bit TC1 interrupt)	$IMF \cdot EF_4 = 1$	IL_4	$FFF6_H$	4
	reserved	$IMF \cdot EF_5 = 1$	IL_5	$FFF4_H$	5
Internal	INTTBT (Time Base Timer interrupt)	$IMF \cdot EF_6 = 1$	IL_6	$FFF2_H$	6
	reserved	$IMF \cdot EF_7 = 1$	IL_7	$FFF0_H$	7
Internal	INTTC3 (8-bit TC3 interrupt)	$IMF \cdot EF_8 = 1$	IL_8	$FFEE_H$	8
	reserved (INTSBI at 87PM36 and emulator)	$IMF \cdot EF_9 = 1$	IL_9	$FFEC_H$	9
Internal	INTTC4 (8-bit TC4 interrupt)	$IMF \cdot EF_{10} = 1$	IL_{10}	$FFEA_H$	10
External	INT3 (External interrupt 3, Remote control receive interrupt)	$IMF \cdot EF_{11} = 1$	IL_{11}	$FFE8_H$	11
External	INT4 (External interrupt 4)	$IMF \cdot EF_{12} = 1$	IL_{12}	$FFE6_H$	12
Internal	INTOSD (OSD interrupt)	$IMF \cdot EF_{13} = 1$	IL_{13}	$FFE4_H$	13
Internal	INTTC2 (16-bit TC2 interrupt)	$IMF \cdot EF_{14} = 1$	IL_{14}	$FFE2_H$	14
External	INT5 (External interrupt 5)	$IMF \cdot EF_{15} = 1$	IL_{15}	$FFE0_H$	Low 15

INPUT / OUTPUT CIRCUITRY

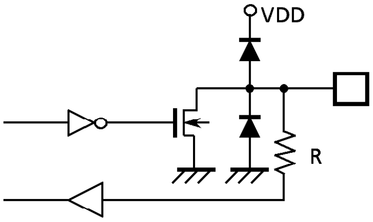
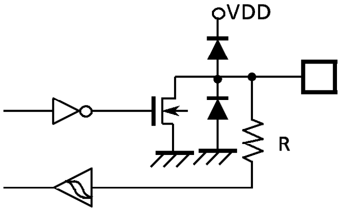
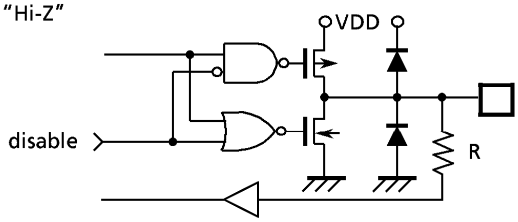
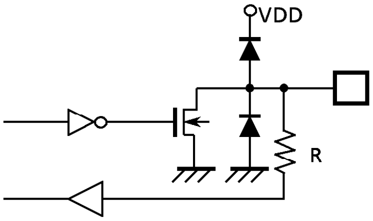
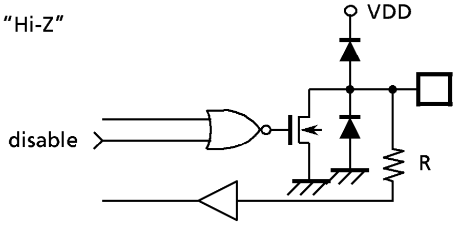
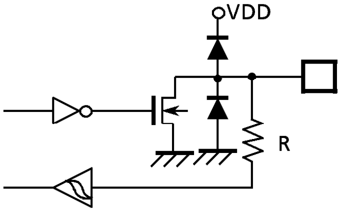
(1) Control pins

The input / output circuitries of the 87CC31/H31 control pins are shown below.

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

(2) Input / Output Ports

The input / output circuitries of the 87CC31/H31 I/O ports are shown below.

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

ELECTRICAL CHARACTERISTICS

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

PARAMETER	SYMBOL	PINS	RATINGS	UNIT
Supply Voltage	V_{DD}		– 0.3 to 6.5	V
Input Voltage	V_{IN}		– 0.3 to V_{DD} + 0.3	V
Output Voltage	V_{OUT1}		– 0.3 to V_{DD} + 0.3	V
Output Current (Per 1 pin)	I_{OUT1}	Ports P2, P3, P4, P5, P64 to P67, P7	3.2	mA
	I_{OUT2}	Ports P60 to P63	30	
Output Current (Total)	ΣI_{OUT1}	Ports P2, P3, P4, P5, P64 to P67, P7	120	mA
	ΣI_{OUT2}	Ports P60 to P63	120	
Power Dissipation [Topr = 70 °C]	PD		600	mW
Soldering Temperature (time)	Tsld		260 (10 s)	°C
Storage Temperature	Tstg		– 55 to 125	°C
Operating Temperature	Topr		– 30 to 70	°C

RECOMMENDED OPERATING CONDITIONS (V_{SS} = 0 V, Topr = – 30 to 70 °C)

PARAMETER	SYMBOL	PINS	CONDITIONS		Min.	Max.	UNIT
Supply Voltage	V_{DD}		fc = 8 MHz	NORMAL mode	4.5	5.5	V
				IDLE mode			
				STOP mode	2.0		
Input High Voltage	V_{IH1}	Except hysteresis input	$V_{DD} \geqq 4.5$ V		$V_{DD} \times 0.70$	V_{DD}	V
	V_{IH2}	Hysteresis input			$V_{DD} \times 0.75$		
Input Low Voltage	V_{IL1}	Except hysteresis input	$V_{DD} \geqq 4.5$ V		0	$V_{DD} \times 0.30$	V
	V_{IL2}	Hysteresis input				$V_{DD} \times 0.25$	
Clock Frequency	fc	XIN, XOUT	V_{DD} = 4.5 to 5.5 V		4.0	8.0	MHz
	f_{OSC}	OSC1, OSC2	Normal frequency mode (FORS = 0, V_{DD} = 4.5 to 5.5 V)		4.0	$f_{OSC} \leqq fc \times 1.2 \leqq 8.0$	
			Double frequency mode (FORS = 1, V_{DD} = 4.5 to 5.5 V)		2.0	$f_{OSC} \leqq fc \times 0.6 \leqq 4.0$	

Note : Clock Frequency fc ; The condition of supply voltage range is the value in NORMAL and IDLE modes.

D.C. CHARACTERISTICS

(V_{SS} = 0 V, 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} = 5.5 V, V_{IN} = 5.5 V / 0 V	–	–	± 2	μA
	I_{IN2}	Open drain ports	V_{DD} = 5.5 V, V_{IN} = 5.5 V	–	–	2	
	I_{IN3}	Tri-state ports	V_{DD} = 5.5 V, V_{IN} = 5.5 V / 0 V	–	–	± 2	
	I_{IN4}	$\overline{\text{RESET}}$, $\overline{\text{STOP}}$					
Input Resistance	R_{IN2}	$\overline{\text{RESET}}$		100	220	450	kΩ
Output Leakage Current	I_{LO1}	Sink open drain ports	V_{DD} = 5.5 V, V_{OUT} = 5.5 V	–	–	2	μA
	I_{LO2}	Tri-state ports	V_{DD} = 5.5 V, V_{OUT} = 5.5 V/0 V	–	–	± 2	
Output High Voltage	V_{OH2}	Tri- state port	V_{DD} = 4.5 V, I_{OH} = – 0.7 mA	4.1	–	–	V
Output Low Voltage	V_{OL}	Except XOUT and Ports P60 to P63	V_{DD} = 4.5 V, I_{OL} = 1.6 mA	–	–	0.4	V
Output Low Current	I_{OL3}	Ports P60 to P63	V_{DD} = 4.5 V, V_{OL} = 1.0 V	–	20	–	mA
Supply Current in NORMAL mode	I_{DD}		V_{DD} = 5.5 V fc = 8 MHz V_{IN} = 5.3 V/0.2 V	–	10	16	mA
Supply Current in IDLE mode				–	6	8	mA
Supply Current in STOP mode			V_{DD} = 5.5 V V_{IN} = 5.3 V/0.2 V	–	0.5	10	μA

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

Note 2 : Input Current I_{IN1}, I_{IN4} ; The current through pull-up or pull-down resistor is not included.

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

A/D CONVERSION CHARACTERISTICS

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

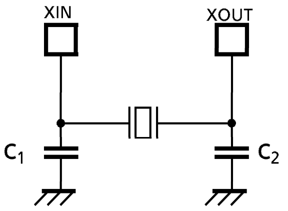
PARAMETER	SYMBOL	PINS	CONDITIONS	Min.	Typ.	Max.	UNIT
Analog Input Voltage Range	V_{AIN}	CIN3 to CIN0		V_{SS}	–	V_{DD}	V
Conversion Error			V_{DD} = 5.0 V	–	–	± 1.5	LSB

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

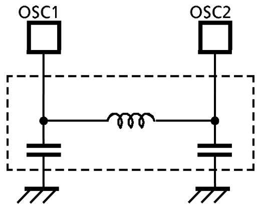
PARAMETER	SYMBOL	CONDITIONS	Min.	Typ.	Max.	UNIT
Machine Cycle Time	tcy	In NORMAL mode	0.5	–	1.0	μs
		In IDLE mode				
High Level Clock Pulse Width	t_{WCH}	For external clock operation (XIN input) , fc = 8 MHz	62.5	–	–	ns
Low Level Clock Pulse Width	t_{WCL}					

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

PARAMETER	OSCILLATOR	FREQUENCY	RECOMMENDED OSCILLATOR	RECOMMENDED CONDITIONS	
				C_1	C_2
High-frequency Osillation	Ceramic Resonator	8 MHz	KYOCERA KBR8.0M	30 pF	30 pF
		4 MHz	KYOCERA KBR4.0MS MURATA CSA4.00MG		
	Crystal Oscillator	8 MHz	TOYOCOM 210B 8.0000	20 pF	20 pF
		4 MHz	TOYOCOM 204B 4.0000		
OSD	LC Resonator	8 MHz	TOKO A285TNIS-11695	–	–
		7 MHz	TOKO TBEKSES-30375FBY		



(1) High-frequency



(2) LC Resonator for OSD

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