TOSHIBA

8 Bit Microcontroller
TLCS-870/C Series

TMP86FM26UG

TOSHIBA CORPORATION

Revision History

Date	Revision	
2007/3/27	1	First Release
2007/7/27	2	Contents Revised

Table of Contents

12.   Synchronous Serial Interface (SIO)

13.   Asynchronous Serial interface (UART0)

14.   Asynchronous Serial interface (UART1)

15. Key-on Wakeup (KWU)

16. LCD Driver

17. FLASH Memory

18. Input/Output Circuit

19. Electrical Characteristics

20. Package Dimensions

This is a technical document that describes the operating functions and electrical specifications of the 8-bit microcontroller series TLCS-870/C (LSI).

CMOS 8-Bit Microcontroller

TMP86FM26UG

Product No.	ROM (FLASH)	RAM	Package	Emulation Chip
TMP86FM26UG	32768 bytes	1024 bytes	LQFP64-P-1010-0.50E	TMP86C926XB

Note: Of the 32768 bytes of ROM (FLASH), 512 bytes can also be used as flash data memory.

1.1 Features

1. 8-bit single chip microcomputer TLCS-870/C series
    - Instruction execution time :

        0.25 μs (at 16 MHz)

        122 μs (at 32.768 kHz)
    - 132 types & 731 basic instructions
2. 21interrupt sources (External : 6 Internal : 15)
3. Input / Output ports (41 pins)

    Large current output: 23pins (Typ. 20mA), LED direct drive

4. Watchdog Timer
5. Prescaler
    - Time base timer
    - Divider output function
6. 18-bit Timer/Counter : 1ch
    - Timer Mode
    - Event Counter Mode
    - Pulse Width Measurement Mode
    - Frequency Measurement Mode
7. 8-bit timer counter : 4 ch
    - Timer, Event counter, Programmable divider output (PDO),

Pulse width modulation (PWM) output,

Programmable pulse generation (PPG) modes

8. Real Time Clock : 1ch

9. 8-bit UART/SIO: 1 ch

10. 8-bit UART : 1 ch

11. Key-on wakeup : 4 ch

12. LCD driver/controller

Built-in voltage booster for LCD driver With display memory
LCD direct drive capability (MAX 32 seg × 4 com)
1/4,1/3,1/2duties or static drive are programmably selectable

13. Clock operation

Single clock mode

Dual clock mode

14. Low power consumption operation

STOP mode: Oscillation stops. (Battery/Capacitor back-up.)

SLOW1 mode: Low power consumption operation using low-frequency clock.(High-frequency clock stop.)

SLOW2 mode: Low power consumption operation using low-frequency clock.(High-frequency clock oscillate.)

IDLE0 mode: CPU stops, and only the Time-Based-Timer(TBT) on peripherals operate using high frequency clock. Release by falling edge of the source clock which is set by TBTCR<TBTCK>.

IDLE1 mode: CPU stops and peripherals operate using high frequency clock. Release by interruputs(CPU restarts).

IDLE2 mode: CPU stops and peripherals operate using high and low frequency clock. Release by interruputs. (CPU restarts).

SLEEP0 mode: CPU stops, and only the Time-Based-Timer(TBT) on peripherals operate using low frequency clock.Release by falling edge of the source clock which is set by TBTCR<TBTCK>.

SLEEP1 mode: CPU stops, and peripherals operate using low frequency clock. Release by interruput.(CPU restarts).

SLEEP2 mode: CPU stops and peripherals operate using high and low frequency clock.   Release by interruput.

15. Wide operation voltage:

2.7 V to 3.6 V at 16MHz /32.768 kHz

1.8 V to 3.6 V at 8 MHz /32.768 kHz

1.2   Pin Assignment
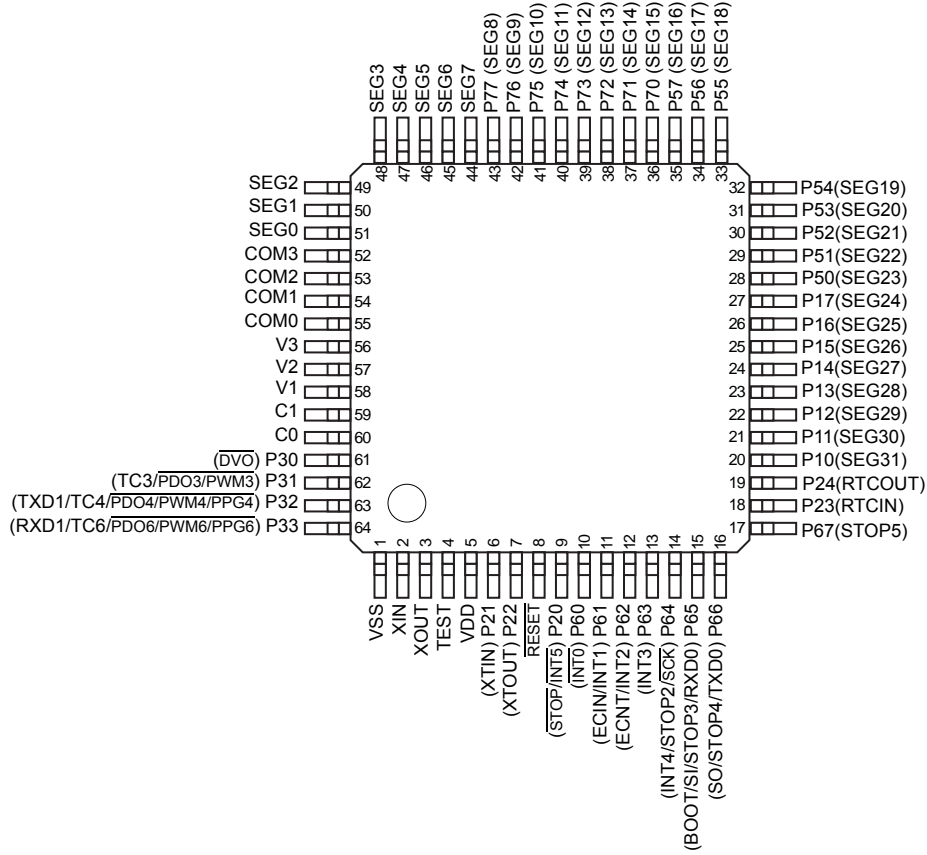
Figure 1-1 Pin Assignment

1.3 Block Diagram
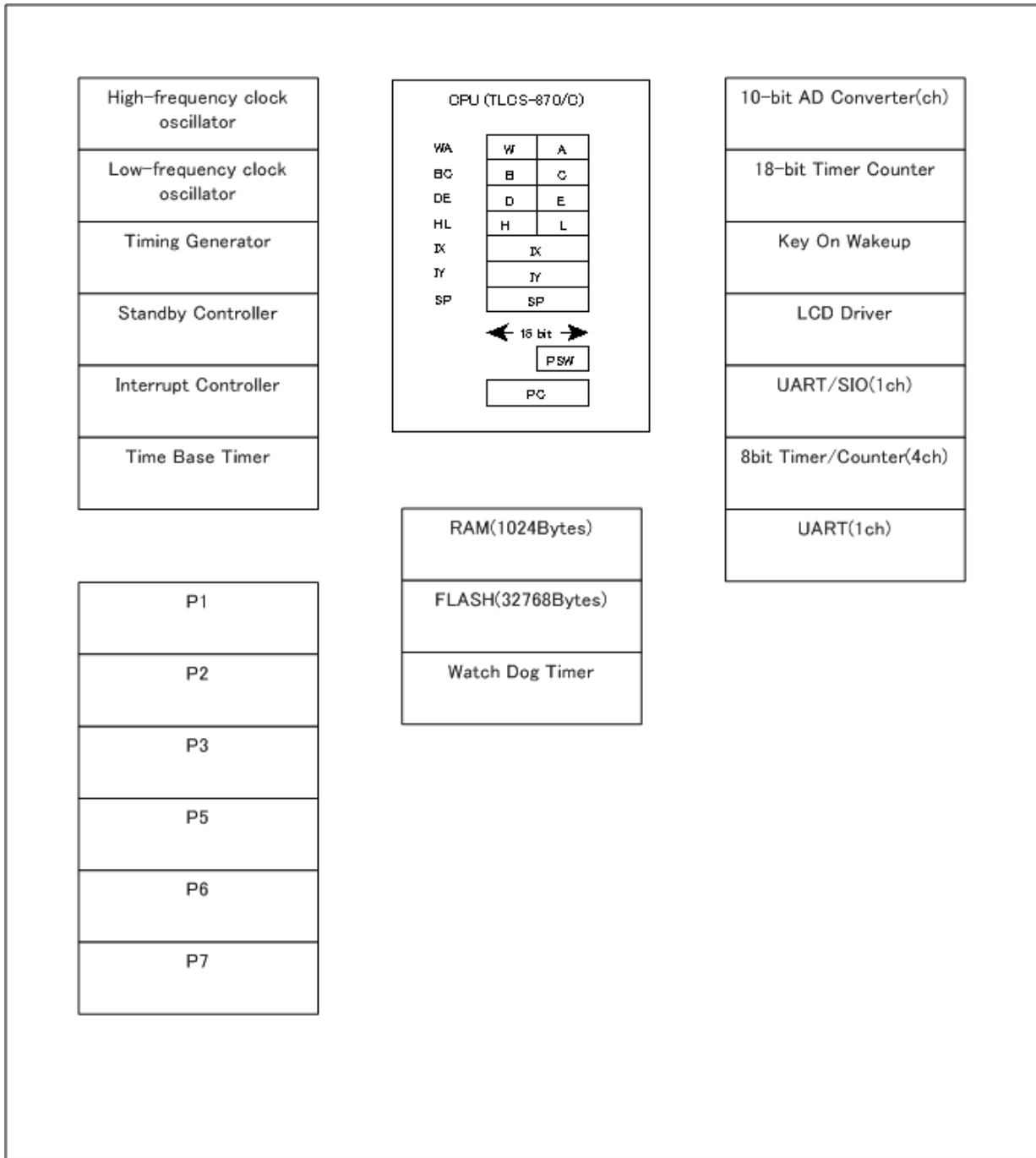
Figure 1-2 Block Diagram

1.4 Pin Names and Functions

The TMP86FM26UG has MCU mode and serial PROM mode. Table 1-1 shows the pin functions in MCU mode. The serial PROM mode is explained later in a separate chapter.

Table 1-1 Pin Names and Functions(1/3)

Pin Name	Pin Number	Input/Output	Functions
P17 SEG24	27	IO O	PORT17 LCD segment output 24
P16 SEG25	26	IO O	PORT16 LCD segment output 25
P15 SEG26	25	IO O	PORT15 LCD segment output 26
P14 SEG27	24	IO O	PORT14 LCD segment output 27
P13 SEG28	23	IO I	PORT13 LCD segment output 28
P12 SEG29	22	IO O	PORT12 LCD segment output 29
P11 SEG30	21	IO O	PORT11 LCD segment output 30
P10 SEG31	20	IO O	PORT10 LCD segment output 31
P24 RTCOUT	19	IO O	PORT24 RTC output
P23 RTCIN	18	IO I	PORT23 RTC input
P22 XTOUT	7	IO O	PORT22 Resonator connecting pins(32.768kHz) for inputting external clock
P21 XTIN	6	IO I	PORT21 Resonator connecting pins(32.768kHz) for inputting external clock
P20 $\overline{INT5}$ $\overline{STOP}$	9	IO I I	PORT20 External interrupt 5 input STOP mode release signal input
P33 $\overline{PDO6/PWM6/PPG6}$ TC6 RXD1	64	IO O I I	PORT33 PDO6/PWM6/PPG6 output TC6 input UART data input 1
P32 $\overline{PDO4/PWM4/PPG4}$ TC4 TXD1	63	IO O I O	PORT32 PDO4/PWM4/PPG4 output TC4 input UART data output 1
P31 $\overline{PDO3/PWM3}$ TC3	62	IO O I	PORT31 PDO3/PWM3 output TC3 input
P30 $\overline{DVO}$	61	IO O	PORT30 Divider Output
P57 SEG16	35	IO O	PORT57 LCD segment output 16

Table 1-1   Pin Names and Functions(2/3)

Pin Name	Pin Number	Input/Output	Functions
P56 SEG17	34	IO O	PORT56 LCD segment output 17
P55 SEG18	33	IO O	PORT55 LCD segment output 18
P54 SEG19	32	IO O	PORT54 LCD segment output 19
P53 SEG20	31	IO O	PORT53 LCD segment output 20
P52 SEG21	30	IO O	PORT52 LCD segment output 21
P51 SEG22	29	IO O	PORT51 LCD segment output 22
P50 SEG23	28	IO O	PORT50 LCD segment output 23
P67 STOP5	17	IO I	PORT67 STOP5 input
P66 TXD0 STOP4 SO	16	IO O I O	PORT66 UART data output 0 STOP4 input Serial Data Output
P65 RXD0 STOP3 SI BOOT	15	IO I I I I	PORT65 UART data input 0 STOP3 input Serial Data Input Serial PROM mode control input
P64 $\overline{\text{SCK}}$ STOP2 INT4	14	IO I I I	PORT64 Serial Clock I/O STOP2 input External interrupt 4 input
P63 INT3	13	IO I	PORT63 External interrupt 3 input
P62 INT2 ECNT	12	IO I I	PORT62 External interrupt 2 input ECNT input
P61 INT1 ECIN	11	IO I I	PORT61 External interrupt 1 input ECIN input
P60 $\overline{\text{INT0}}$	10	IO I	PORT60 External interrupt 0 input
P77 SEG8	43	IO O	PORT77 LCD segment output 8
P76 SEG9	42	IO O	PORT76 LCD segment output 9
P75 SEG10	41	IO O	PORT75 LCD segment output 10
P74 SEG11	40	IO O	PORT74 LCD segment output 11
P73 SEG12	39	IO O	PORT73 LCD segment output 12

Table 1-1   Pin Names and Functions(3/3)

Pin Name	Pin Number	Input/Output	Functions
P72 SEG13	38	IO O	PORT72 LCD segment output 13
P71 SEG14	37	IO O	PORT71 LCD segment output 14
P70 SEG15	36	IO O	PORT70 LCD segment output 15
SEG7	44	O	LCD segment output 7
SEG6	45	O	LCD segment output 6
SEG5	46	O	LCD segment output 5
SEG4	47	O	LCD segment output 4
SEG3	48	O	LCD segment output 3
SEG2	49	O	LCD segment output 2
SEG1	50	O	LCD segment output 1
SEG0	51	O	LCD segment output 0
COM3	52	O	LCD common output 3
COM2	53	O	LCD common output 2
COM1	54	O	LCD common output 1
COM0	55	O	LCD common output 0
V3	56	I	LCD voltage booster pin
V2	57	I	LCD voltage booster pin
V1	58	I	LCD voltage booster pin
C1	59	I	LCD voltage booster pin
C0	60	I	LCD voltage booster pin
XIN	2	I	Resonator connecting pins for high-frequency clock
XOUT	3	O	Resonator connecting pins for high-frequency clock
$\overline{\text{RESET}}$	8	IO	Reset signal
TEST	4	I	Test pin for out-going test. Normally, be fixed to low.
VAREF	18	I	Analog Base Voltage Input Pin for A/D Conversion
AVDD	19	I	Analog Power Supply
VDD	4	I	Power Supply
VSS	1	I	0(GND)

2. Operational Description

2.1 CPU Core Functions

The CPU core consists of a CPU, a system clock controller, and an interrupt controller.

This section provides a description of the CPU core, the program memory, the data memory, and the reset circuit.

2.1.1 Memory Address Map

The TMP86FM26UG memory is composed Flash, BOOTROM, RAM, DBR(Data buffer register) and SFR(Special function register). They are all mapped in 64-Kbyte address space. Figure 2-1 shows the TMP86FM26UG memory address map.

SFR	0000$_H$ 003F$_H$	64 bytes	SFR: Special function register includes: I/O ports Peripheral control registers Peripheral status registers System control registers Program status word
RAM	0040$_H$ 043F$_H$	1024 bytes	RAM: Random access memory includes: Data memory Stack
DBR	0F80$_H$ 0FFF$_H$	128 bytes	DBR: Data buffer register includes: Peripheral control registers Peripheral status registers LCD display memory
BOOTROM	3800$_H$ 3FFF$_H$	2048 bytes	BOOTROM: Flash programming control program
Flash	8000$_H$ 81FF$_H$ 8200$_H$	32768 bytes	Flash: Program memory The area of 8000$_H$ to 81FF$_H$ can be also used as data memory of FLASH.
	FFB0$_H$ FFBF$_H$		Vector table for interrupts (16 bytes)
	FFC0$_H$ FFDF$_H$		Vector table for vector call instructions (32 bytes)
	FFE0$_H$ FFFF$_H$		Vector table for interrupts (32 bytes)

Figure 2-1 Memory Address Map

2.1.2 Program Memory (Flash)

The TMP86FM26UG has a 32768 bytes (Address 8000H to FFFFH) of program memory (Flash). The area of 8000H to 81FFH can be used as a 512 bytes data memory of FLASH.

2.1.3   Data Memory (RAM)

The TMP86FM26UG has 1024bytes (Address 0040H to 043FH) of internal RAM. The first 192 bytes (0040H to 00FFH) of the internal RAM are located in the direct area; instructions with shorten operations are available against such an area.

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

Example :Clears RAM to "00H". (TMP86FM26UG)

```
                    LD       HL, 0040H              ; Start address setup
                    LD       A, H                   ; Initial value (00H) setup
                    LD       BC, 03FFH
    SRAMCLR:        LD       (HL), A
                    INC      HL
                    DEC      BC
                    JRS      F, SRAMCLR
```

2.2   System Clock Controller

The system clock controller consists of a clock generator, a timing generator, and a standby controller.
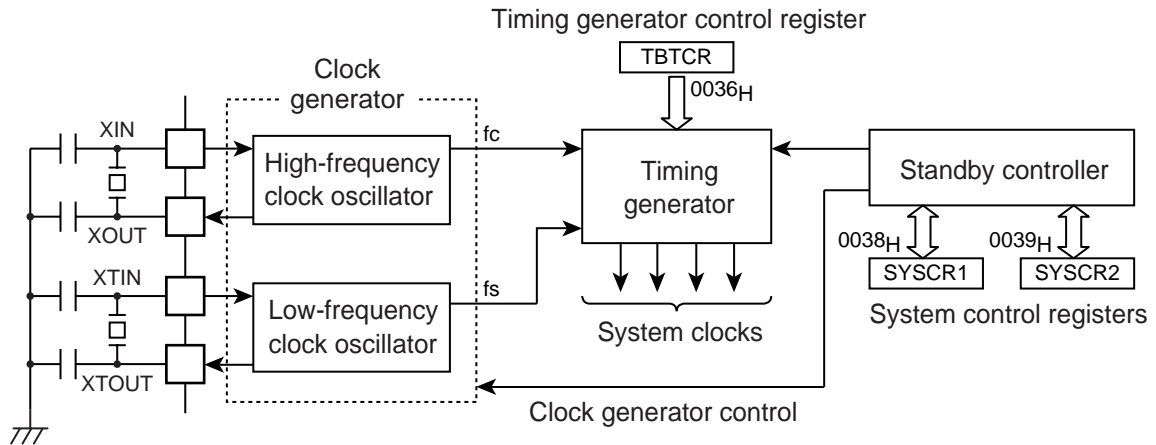
Figure 2-2 System Colck Control

2.2.1   Clock Generator

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

The high-frequency (fc) clock and low-frequency (fs) clock can easily be obtained by connecting a resonator between the XIN/XOUT and XTIN/XTOUT pins respectively. Clock input from an external oscillator is also possible. In this case, external clock is applied to XIN/XTIN pin with XOUT/XTOUT pin not connected.
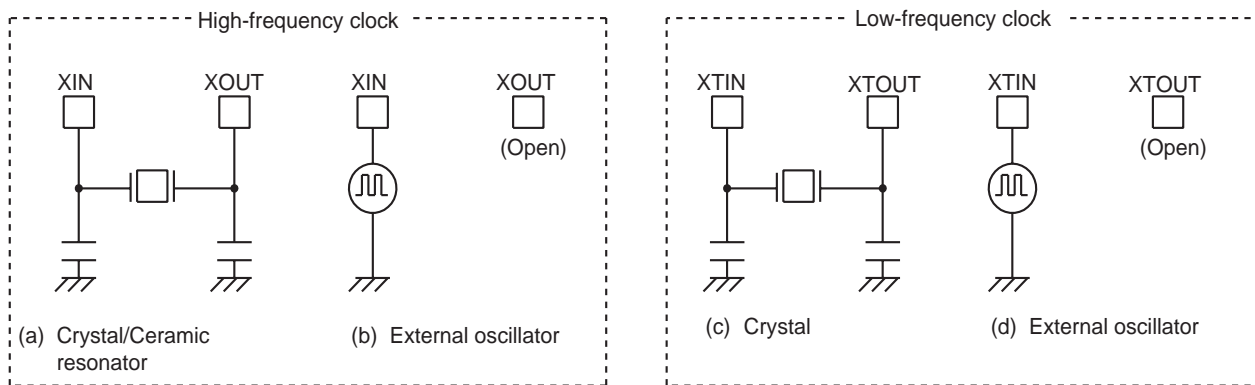
Figure 2-3 Examples of Resonator Connection

Note: The function to monitor the basic clock directly at external is not provided for hardware, however, with disabling all interrupts and watchdog timers, the oscillation frequency can be adjusted by monitoring the pulse which the fixed frequency is outputted to the port by the program.
The system to require the adjustment of the oscillation frequency should create the program for the adjustment in advance.

2.2.2   Timing Generator

The timing generator generates the various system clocks supplied to the CPU core and peripheral hardware from the basic clock (fc or fs). The timing generator provides the following functions.

1. Generation of main system clock
2. Generation of divider output ($\overline{\text{DVO}}$) pulses
3. Generation of source clocks for time base timer
4. Generation of source clocks for watchdog timer
5. Generation of internal source clocks for timer/counters
6. Generation of warm-up clocks for releasing STOP mode
7. LCD

2.2.2.1   Configuration of timing generator

The timing generator consists of a 2-stage prescaler, a 21-stage divider, a main system clock generator, and machine cycle counters.

An input clock to the 7th stage of the divider depends on the operating mode, SYSCR2<SYSCK> and TBTCR<DV7CK>, that is shown in Figure 2-4. As reset and STOP mode started/canceled, the prescaler and the divider are cleared to "0".
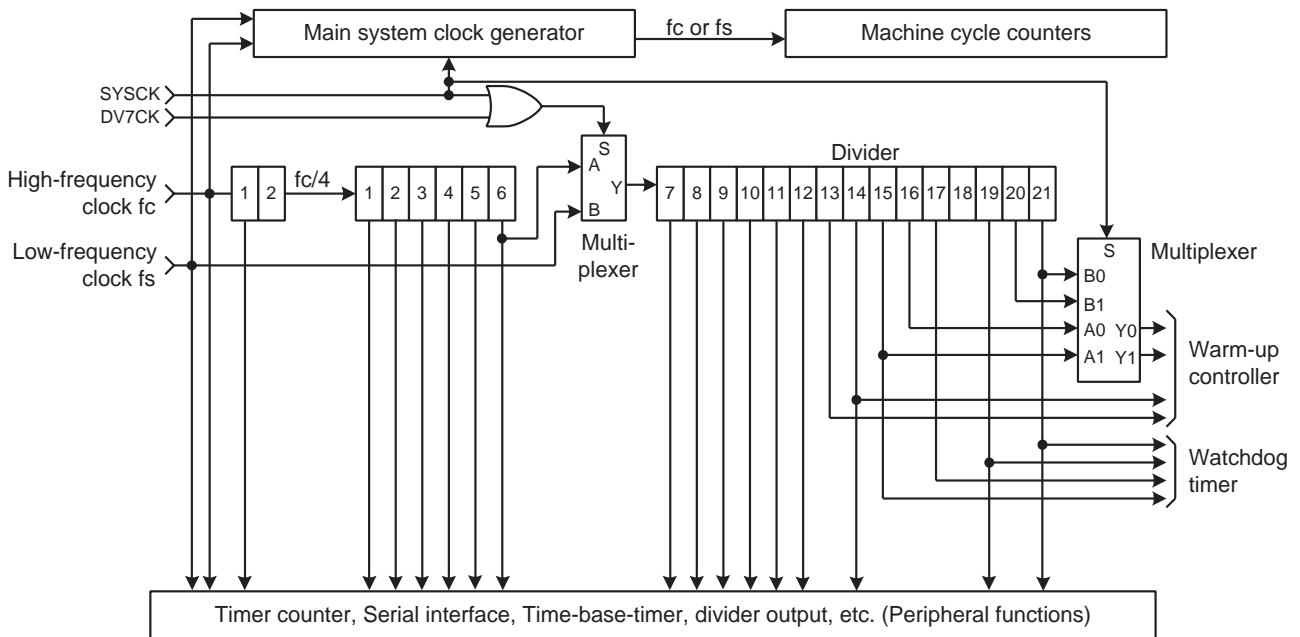
Figure 2-4 Configuration of Timing Generator

Timing Generator Control Register

TBTCR	7	6	5	4	3	2	1	0	
(0036H)	(DVOEN)	(DVOCK)		DV7CK	(TBTEN)	(TBTCK)			(Initial value: 0000 0000)

DV7CK	Selection of input to the 7th stage of the divider	0: fc/2^8 [Hz] 1: fs	R/W
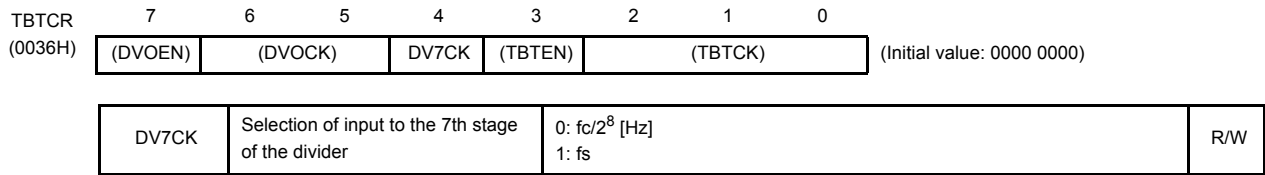
Note 1: In single clock mode, do not set DV7CK to "1".

Note 2: Do not set "1" on DV7CK while the low-frequency clock is not operated stably.

Note 3: fc: High-frequency clock [Hz], fs: Low-frequency clock [Hz], *: Don't care

Note 4: In SLOW1/2 and SLEEP1/2 modes, the DV7CK setting is ineffective, and fs is input to the 7th stage of the divider.

Note 5: When STOP mode is entered from NORMAL1/2 mode, the DV7CK setting is ineffective during the warm-up period after release of STOP mode, and the 6th stage of the divider is input to the 7th stage during this period.

2.2.2.2　Machine cycle

Instruction execution and peripheral hardware operation are synchronized with the main system clock.

The minimum instruction execution unit is called an "machine cycle". There are a total of 10 different types of instructions for the TLCS-870/C 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.
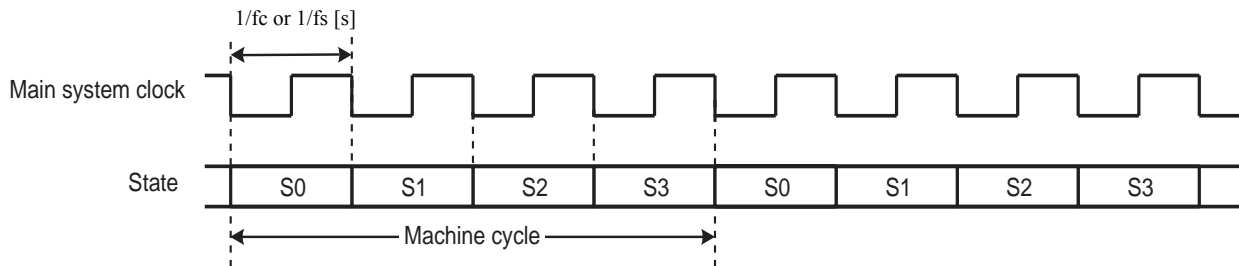
Figure 2-5　Machine Cycle

2.2.3　Operation Mode Control Circuit

The operation mode control circuit starts and stops the oscillation circuits for the high-frequency and low-frequency clocks, and switches the main system clock. There are three operating modes: Single clock mode, dual clock mode and STOP mode. These modes are controlled by the system control registers (SYSCR1 and SYSCR2). Figure 2-6 shows the operating mode transition diagram.

Note 1: When the IDLE0/1/2 and SLEEP0/1/2 modes are started with the EEPCR<ATPWDW> = "0", the CPU wait period for stabilizing of the power supply of Flash control circuit is executed after being released from these mode.

Note 2: When the STOP mode is started with the EEPCR<MNPWDW> = "1", the CPU wait period for stablizing of the power supply of flash control circuit is executed after the STOP warm-up time.

2.2.3.1　Single-clock mode

Only the oscillation circuit for the high-frequency clock is used, and P21 (XTIN) and P22 (XTOUT) pins are used as input/output ports. The main-system clock is obtained from the high-frequency clock. In the single-clock mode, the machine cycle time is 4/fc [s].

(1) NORMAL1 mode

In this mode, both the CPU core and on-chip peripherals operate using the high-frequency clock. The TMP86FM26UG is placed in this mode after reset.

(2) IDLE1 mode

In this mode, the internal oscillation circuit remains active. The CPU and the watchdog timer are halted; however on-chip peripherals remain active (Operate using the high-frequency clock).

IDLE1 mode is started by SYSCR2<IDLE> = "1", and IDLE1 mode is released to NORMAL1 mode by an interrupt request from the on-chip peripherals or external interrupt inputs. When the IMF (Interrupt master enable flag) is "1" (Interrupt enable), the execution will resume with the acceptance of the interrupt, and the operation will return to normal after the interrupt service is completed. When the IMF is "0" (Interrupt disable), the execution will resume with the instruction which follows the IDLE1 mode start instruction.

(3) IDLE0 mode

In this mode, all the circuit, except oscillator and the timer-base-timer, stops operation.

This mode is enabled by SYSCR2<TGHALT> = "1".

When IDLE0 mode starts, the CPU stops and the timing generator stops feeding the clock to the peripheral circuits other than TBT. Then, upon detecting the falling edge of the source clock selected with TBTCR<TBTCK>, the timing generator starts feeding the clock to all peripheral circuits.

When returned from IDLE0 mode, the CPU restarts operating, entering NORMAL1 mode back again. IDLE0 mode is entered and returned regardless of how TBTCR<TBTEN> is set. When IMF = "1", EF6 (TBT interrupt individual enable flag) = "1", and TBTCR<TBTEN> = "1", interrupt processing is performed. When IDLE0 mode is entered while TBTCR<TBTEN> = "1", the INTTBT interrupt latch is set after returning to NORMAL1 mode.

2.2.3.2 Dual-clock mode

Both the high-frequency and low-frequency oscillation circuits are used in this mode. P21 (XTIN) and P22 (XTOUT) pins cannot be used as input/output ports. The main system clock is obtained from the high-frequency clock in NORMAL2 and IDLE2 modes, and is obtained from the low-frequency clock in SLOW and SLEEP modes. The machine cycle time is 4/fc [s] in the NORMAL2 and IDLE2 modes, and 4/fs [s] (122 μs at fs = 32.768 kHz) in the SLOW and SLEEP modes.

The TLCS-870/C is placed in the signal-clock mode during reset. To use the dual-clock mode, the low-frequency oscillator should be turned on at the start of a program.

(1) NORMAL2 mode

In this mode, the CPU core operates with the high-frequency clock. On-chip peripherals operate using the high-frequency clock and/or low-frequency clock.

(2) SLOW2 mode

In this mode, the CPU core operates with the low-frequency clock, while both the high-frequency clock and the low-frequency clock are operated. As the SYSCR2<SYSCK> becomes "1", the hardware changes into SLOW2 mode. As the SYSCR2<SYSCK> becomes "0", the hardware changes into NORMAL2 mode. As the SYSCR2<XEN> becomes "0", the hardware changes into SLOW1 mode. Do not clear SYSCR2<XTEN> to "0" during SLOW2 mode.

(3) SLOW1 mode

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

Switching back and forth between SLOW1 and SLOW2 modes are performed by SYSCR2<XEN>. In SLOW1 and SLEEP modes, the input clock to the 1st stage of the divider is stopped; output from the 1st to 6th stages is also stopped.

(4) IDLE2 mode

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

(5) SLEEP1 mode

In this mode, the internal oscillation circuit of the low-frequency clock remains active. The CPU, the watchdog timer, and the internal oscillation circuit of the high-frequency clock are halted; however, on-chip peripherals remain active (Operate using the low-frequency clock). Starting and releasing of SLEEP mode are the same as for IDLE1 mode, except that operation returns to SLOW1 mode. In SLOW1 and SLEEP1 modes, the input clock to the 1st stage of the divider is stopped; output from the 1st to 6th stages is also stopped.

(6) SLEEP2 mode

The SLEEP2 mode is the idle mode corresponding to the SLOW2 mode. The status under the SLEEP2 mode is same as that under the SLEEP1 mode, except for the oscillation circuit of the high-frequency clock.

(7) SLEEP0 mode

In this mode, all the circuit, except oscillator and the timer-base-timer, stops operation. This mode is enabled by setting "1" on bit SYSCR2<TGHALT>.

When SLEEP0 mode starts, the CPU stops and the timing generator stops feeding the clock to the peripheral circuits other than TBT. Then, upon detecting the falling edge of the source clock selected with TBTCR<TBTCK>, the timing generator starts feeding the clock to all peripheral circuits.

When returned from SLEEP0 mode, the CPU restarts operating, entering SLOW1 mode back again. SLEEP0 mode is entered and returned regardless of how TBTCR<TBTEN> is set. When IMF = "1", EF6 (TBT interrupt individual enable flag) = "1", and TBTCR<TBTEN> = "1", interrupt processing is performed. When SLEEP0 mode is entered while TBTCR<TBTEN> = "1", the INTTBT interrupt latch is set after returning to SLOW1 mode.

2.2.3.3 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 a lowest power consumption during STOP mode.

STOP mode is started by the system control register 1 (SYSCR1), and STOP mode is released by a inputting (Either level-sensitive or edge-sensitive can be programmably selected) to the $\overline{STOP}$ pin. After the warm-up period is completed, the execution resumes with the instruction which follows the STOP mode start instruction.
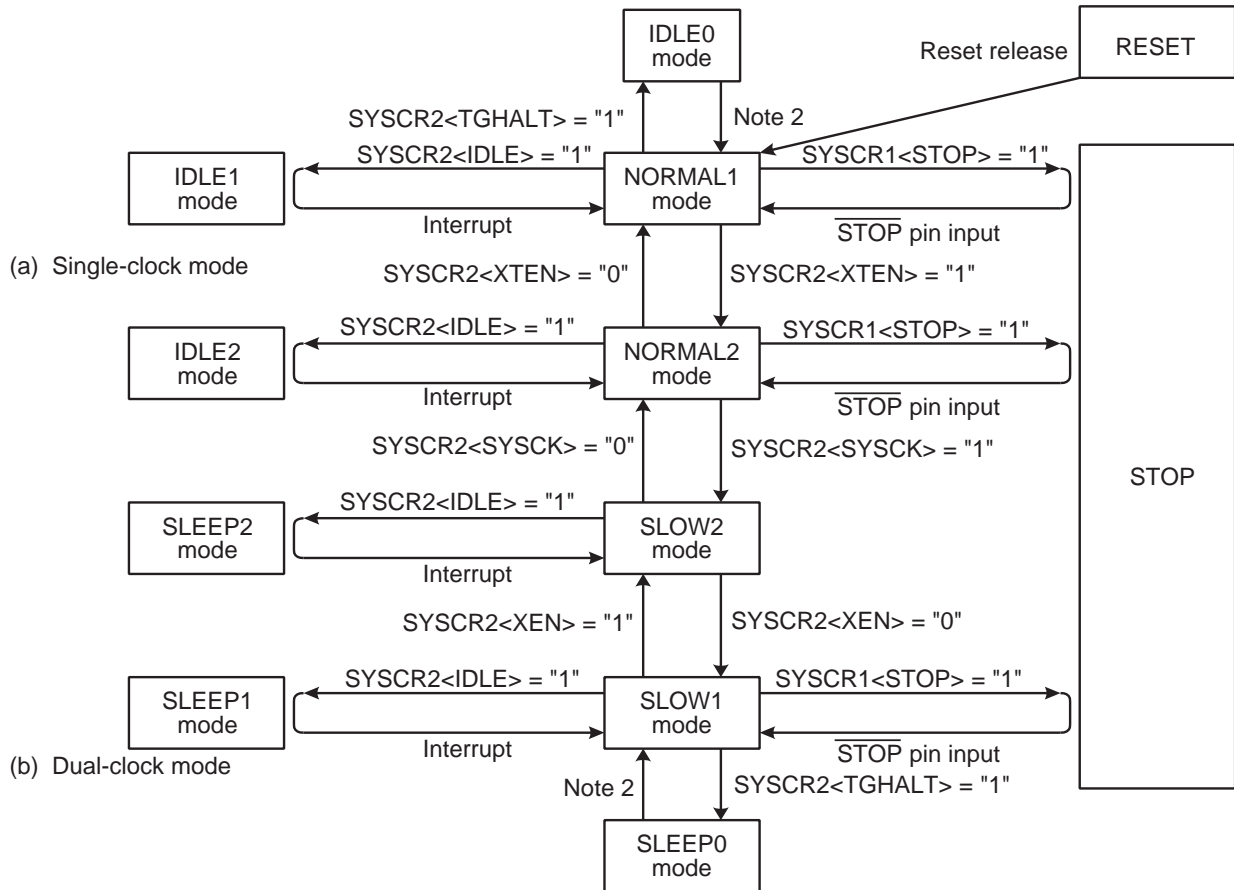
Note 1: NORMAL1 and NORMAL2 modes are generically called NORMAL; SLOW1 and SLOW2 are called SLOW; IDLE0, IDLE1 and IDLE2 are called IDLE; SLEEP0, SLEEP1 and SLEEP2 are called SLEEP.

Note 2: The mode is released by falling edge of TBTCR<TBTCK> setting.

Figure 2-6 Operating Mode Transition Diagram

Table 2-1 Operating Mode and Conditions

Operating Mode		Oscillator		CPU Core	TBT	Other Peripherals	Machine Cycle Time
		High Frequency	Low Frequency				
Single clock	RESET	Oscillation	Stop	Reset	Reset	Reset	4/fc [s]
	NORMAL1			Operate	Operate	Operate	
	IDLE1			Halt			
	IDLE0					Halt	
	STOP	Stop			Halt		–
Dual clock	NORMAL2	Oscillation	Oscillation	Operate with high frequency	Operate	Operate	4/fc [s]
	IDLE2			Halt			
	SLOW2			Operate with low frequency			4/fs [s]
	SLEEP2			Halt			
	SLOW1	Stop		Operate with low frequency			
	SLEEP1			Halt			
	SLEEP0					Halt	
	STOP		Stop	Halt	Halt		–

System Control Register 1

SYSCR1	7	6	5	4	3	2	1	0	
(0038H)	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		R/W
RETM	Operating mode after STOP mode	0: Return to NORMAL1/2 mode 1: Return to SLOW1 mode		R/W
OUTEN	Port output during STOP mode	0: High impedance 1: Output kept		R/W

			Return to NORMAL mode	Return to SLOW mode	
WUT	Warm-up time at releasing STOP mode	00	$3 \times 2^{16}/fc + (2^{10}/fc)$	$3 \times 2^{13}/fs + (2^3/fs)$	R/W
		01	$2^{16}/fc + (2^{10}/fc)$	$2^{13}/fs + (2^3/fs)$	
		10	$3 \times 2^{14}/fc + (2^{10}/fc)$	$3 \times 2^6/fs + (2^3/fs)$	
		11	$2^{14}/fc + (2^{10}/fc)$	$2^6/fs + (2^3/fs)$	

Note 1: Always set RETM to "0" when transiting from NORMAL mode to STOP mode. Always set RETM to "1" when transiting from SLOW mode to STOP mode.

Note 2: When STOP mode is released with $\overline{RESET}$ pin input, a return is made to NORMAL1 regardless of the RETM contents.

Note 3: fc: High-frequency clock [Hz], fs: Low-frequency clock [Hz], *; Don't care

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

Note 5: As the hardware becomes STOP mode under OUTEN = "0", input value is fixed to "0"; therefore it may cause external interrupt request on account of falling edge.

Note 6: When the key-on wakeup is used, RELM should be set to "1".

Note 7: Port P20 is used as $\overline{STOP}$ pin. Therefore, when stop mode is started, OUTEN does not affect to P20, and P20 becomes High-Z mode.

Note 8: The warmig-up time should be set correctly for using oscillator.

Note 9: When the STOP mode is started with the EEPCR<MNPWDW> = "1", the CPU wait period for stabilizing of the power supply of flash control circuit is executed after the STOP warm-up time.(The CPU wait period for FLASH is shown in parentheses)

System Control Register 2

SYSCR2	7	6	5	4	3	2	1	0	
(0039H)	XEN	XTEN	SYSCK	IDLE		TGHALT			(Initial value: 1000 *0**)

XEN	High-frequency oscillator control	0: Turn off oscillation 1: Turn on oscillation		
XTEN	Low-frequency oscillator control	0: Turn off oscillation 1: Turn on oscillation		R/W
SYSCK	Main system clock select (Write)/main system clock monitor (Read)	0: High-frequency clock (NORMAL1/NORMAL2/IDLE1/IDLE2) 1: Low-frequency clock (SLOW1/SLOW2/SLEEP1/SLEEP2)		
IDLE	CPU and watchdog timer control (IDLE1/2 and SLEEP1/2 modes)	0: CPU and watchdog timer remain active 1: CPU and watchdog timer are stopped (Start IDLE1/2 and SLEEP1/2 modes)		R/W
TGHALT	TG control (IDLE0 and SLEEP0 modes)	0: Feeding clock to all peripherals from TG 1: Stop feeding clock to peripherals except TBT from TG. (Start IDLE0 and SLEEP0 modes)		

Note 1: A reset is applied if both XEN and XTEN are cleared to "0", XEN is cleared to "0" when SYSCK = "0", or XTEN is cleared to "0" when SYSCK = "1".

Note 2: *: Don't care, TG: Timing generator, *; Don't care

Note 3: Bits 3, 1 and 0 in SYSCR2 are always read as undefined value.

Note 4: Do not set IDLE and TGHALT to "1" simultaneously.

Note 5: Because returning from IDLE0/SLEEP0 to NORMAL1/SLOW1 is executed by the asynchronous internal clock, the period of IDLE0/SLEEP0 mode might be shorter than the period setting by TBTCR<TBTCK>.

Note 6: When IDLE1/2 or SLEEP1/2 mode is released, IDLE is automatically cleared to "0".

Note 7: When IDLE0 or SLEEP0 mode is released, TGHALT is automatically cleared to "0".

Note 8: Before setting TGHALT to "1", be sure to stop peripherals. If peripherals are not stopped, the interrupt latch of peripherals may be set after IDLE0 or SLEEP0 mode is released.

2.2.4 Operating Mode Control

2.2.4.1 STOP mode

STOP mode is controlled by the system control register 1, the $\overline{STOP}$ pin input and key-on wakeup input (STOP5 to STOP2) which is controlled by the STOP mode release control register (STOPCR).
The $\overline{STOP}$ pin is also used both as a port P20 and an $\overline{INT5}$ (external interrupt input 5) pin. STOP mode is started by setting SYSCR1<STOP> to "1". During STOP mode, the following status is maintained.

1. Oscillations are turned off, and all internal operations are halted.
2. The data memory, registers, the program status word and port output latches are all held in the status in effect before STOP mode was entered.
3. The prescaler and the divider of the timing generator are cleared to "0".
4. The program counter holds the address 2 ahead of the instruction (e.g., [SET (SYSCR1).7]) which started STOP mode.

STOP mode includes a level-sensitive mode and an edge-sensitive mode, either of which can be selected with the SYSCR1<RELM>. Do not use any key-on wakeup input (STOP5 to STOP2) for releasing STOP mode in edge-sensitive mode.

When the STOP mode is started with the EEPCR<MNPWDW> = "1", the CPU wait for stabilizing of the power supply of flash control circuit is executed after the STOP warm-up time.

Note 1: The STOP mode can be released by either the STOP or key-on wakeup pin (STOP5 to STOP2). However, because the STOP pin is different from the key-on wakeup and can not inhibit the release input, the STOP pin must be used for releasing STOP mode.

Note 2: During STOP period (from start of STOP mode to end of warm up), due to changes in the external interrupt pin signal, interrupt latches may be set to "1" and interrupts may be accepted immediately after STOP mode is released. Before starting STOP mode, therefore, disable interrupts. Also, before enabling interrupts after STOP mode is released, clear unnecessary interrupt latches.

(1) Level-sensitive release mode (RELM = "1")

In this mode, STOP mode is released by setting the $\overline{STOP}$ pin high or setting the STOP5 to STOP2 pin input which is enabled by STOPCR. This mode is used for capacitor backup when the main power supply is cut off and long term battery backup.

Even if an instruction for starting STOP mode is executed while $\overline{STOP}$ pin input is high or $STOP5$ to $STOP2$ input is low, STOP mode does not start but instead the warm-up sequence starts immediately. Thus, to start STOP mode in the level-sensitive release mode, it is necessary for the program to first confirm that the $\overline{STOP}$ pin input is low or STOP5 to STOP2 input is high. The following two methods can be used for confirmation.

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

Example 1 :Starting STOP mode from NORMAL mode by testing a port P20.

```
                LD          (SYSCR1), 01010000B     ; Sets up the level-sensitive release mode

SSTOPH:         TEST        (P2PRD). 0              ; Wait until the STOP pin input goes low level

                JRS         F, SSTOPH

                DI                                  ; IMF ← 0

                SET         (SYSCR1). 7             ; Starts STOP mode
```

Example 2 :Starting STOP mode from NORMAL mode with an INT5 interrupt.

```
PINT5:          TEST        (P2PRD). 0              ; To reject noise, STOP mode does not start if

                JRS         F, SINT5                  port P20 is at high

                LD          (SYSCR1), 01010000B     ; Sets up the level-sensitive release mode.

                DI                                  ; IMF ← 0

                SET         (SYSCR1). 7             ; Starts STOP mode

SINT5:          RETI
```
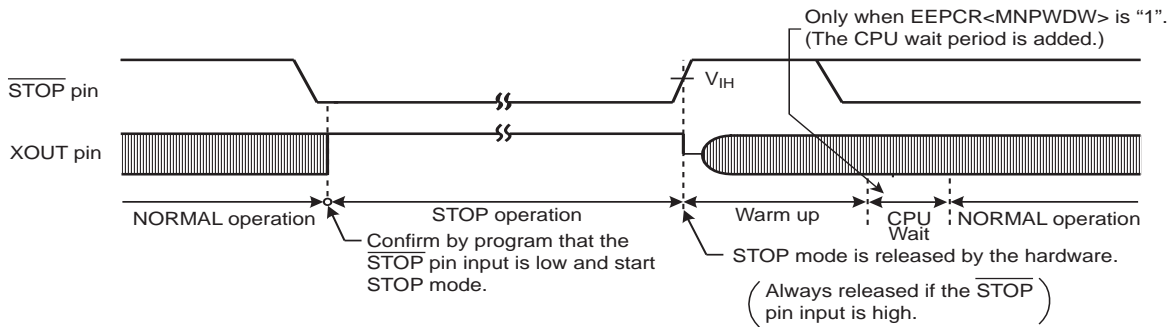

Figure 2-7   Level-sensitive Release Mode

Note 1: Even if the STOP pin input is low after warm-up start, the STOP mode is not restarted.

Note 2: In this case of changing to the level-sensitive mode from the edge-sensitive mode, the release mode is not switched until a rising edge of the STOP pin input is detected.

Note 3: When the STOP mode is started with the EEPCR<MNPWDW> = "1", the CPU wait period for stablizing of the power supply of flash control circuit is executed after the STOP warm-up time.

(2)   Edge-sensitive release mode (RELM = "0")

In this mode, STOP mode is released by a rising edge of the 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 STOP pin. In the edge-sensitive release mode, STOP mode is started even when the STOP pin input is high level. Do not use any STOP5 to STOP2 pin input for releasing STOP mode in edge-sensitive release mode.

Example :Starting STOP mode from NORMAL mode

```
                DI                                  ; IMF ← 0

                LD          (SYSCR1), 10010000B     ; Starts after specified to the edge-sensitive release mode
```
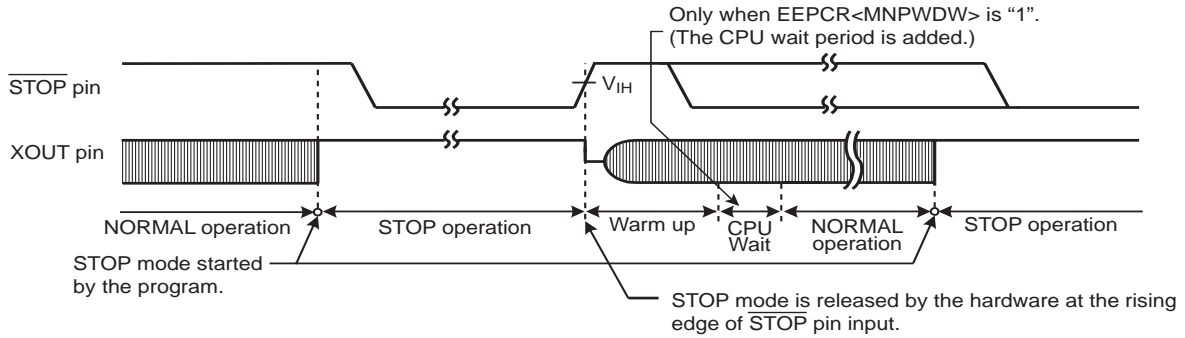
Figure 2-8 Edge-sensitive Release Mode

Note 1: When the STOP mode is started with the EEPCR<MNPWDW> = "1", the CPU wait period for stablizing of the power supply of flash control circuit is executed after the STOP warm-up time.

STOP mode is released by the following sequence.

1. In the dual-clock mode, when returning to NORMAL2, both the high-frequency and low-frequency clock oscillators are turned on; when returning to SLOW1 mode, only the low-frequency clock oscillator is turned on. In the single-clock mode, only the high-frequency clock oscillator is turned on.

2. A warm-up period is inserted to allow oscillation time to stabilize. During warm up, all internal operations remain halted. Four different warm-up times can be selected with the SYSCR1<WUT> in accordance with the resonator characteristics.

3. When the EEPCR<MNPWDW> is "1", the CPU wait period is inserted to stabilize the power supply of flash control circuit. During CPU wait, though CPU operations remain halted, the peripheral function operation is resumed, and the counting of the timing generator is restarted. After the CPU wait is finished, normal operation resumes with the instruction following the STOP mode start instruction.

4. When the EEPCR<MNPWDW> is "0", normal operation resumes with the instruction following the STOP mode start instruction after the STOP warm up.

Note 1: When the STOP mode is released, the start is made after the prescaler and the divider of the timing generator are cleared to "0".

Note 2: STOP mode can also be released by inputting low level on the $\overline{\text{RESET}}$ pin, which immediately performs the normal reset operation.

Note 3: When STOP mode is released with a low hold voltage, the following cautions must be observed. The power supply voltage must be at the operating voltage level before releasing STOP mode. The $\overline{\text{RESET}}$ pin input must also be "H" level, rising together with the power supply voltage. In this case, if an external time constant circuit has been connected, the $\overline{\text{RESET}}$ pin input voltage will increase at a slower pace than the power supply voltage. At this time, there is a danger that a reset may occur if input voltage level of the $\overline{\text{RESET}}$ pin drops below the non-inverting high-level input voltage (Hysteresis input).

Table 2-2   Warm-up Time Example (at fc = 16.0 MHz, fs = 32.768 kHz)

WUT	Warm-up Time [ms]	
	Return to NORMAL Mode	Return to SLOW Mode
00	12.288 + (0.064)	750 + (0.244)
01	4.096 + (0.064)	250 + (0.244)
10	3.072 + (0.064)	5.85 + (0.244)
11	1.024 + (0.064)	1.95 + (0.244)

Note 1: The warm-up time is obtained by dividing the basic clock by the divider. Therefore, the warm-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 warm-up time must be considered as an approximate value.

Note 2: The CPU wait period for FLASH is shown in parentheses.

(a) STOP mode start (Example: Start with SET (SYSCR1). 7 instruction located at address a)
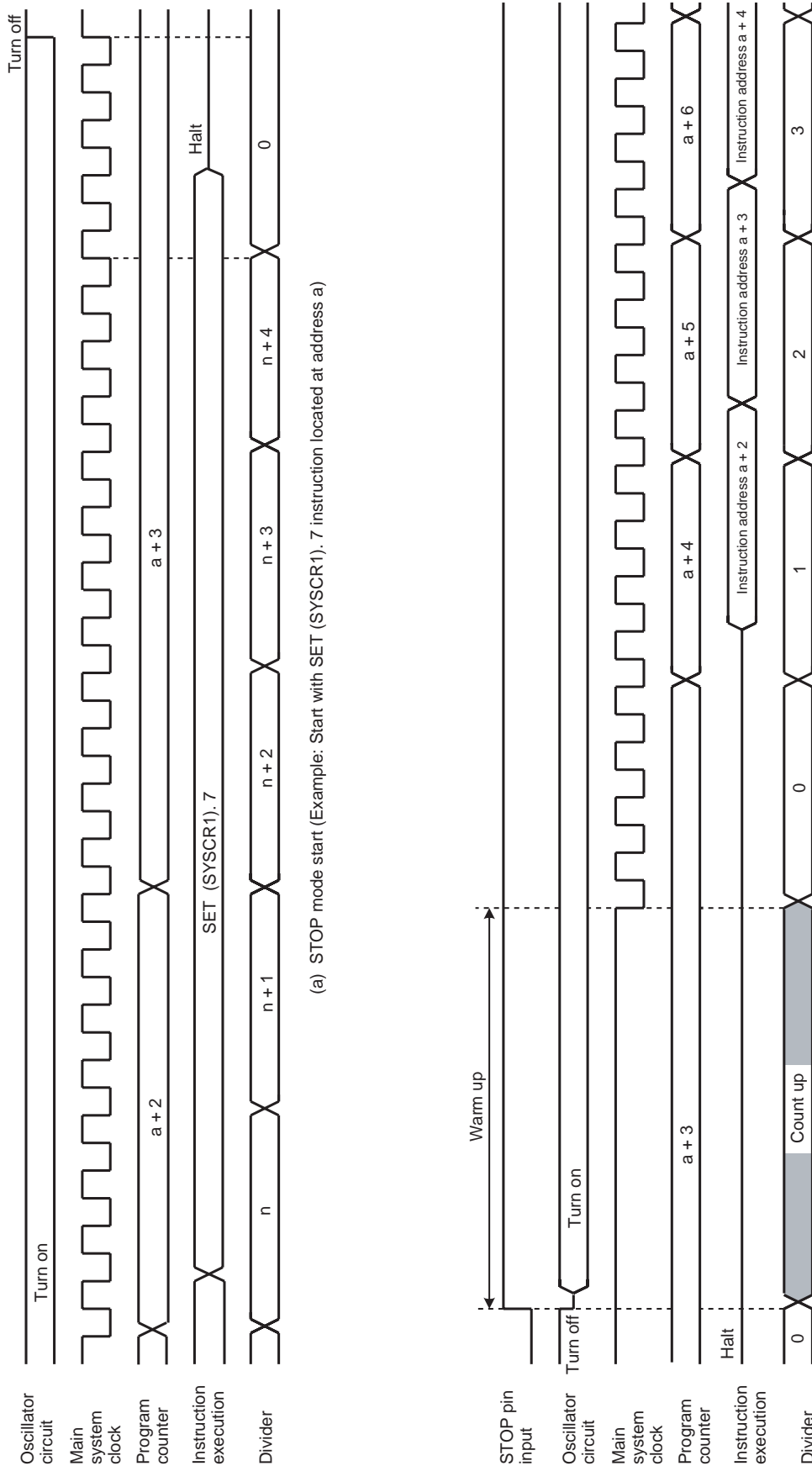
(b) STOP mode release

Figure 2-9 STOP Mode Start/Release (when EEPCR<MNPWDW> = "0")

(a) STOP mode start (Example: Start with SET (SYSCR1). 7 instruction located at address a)
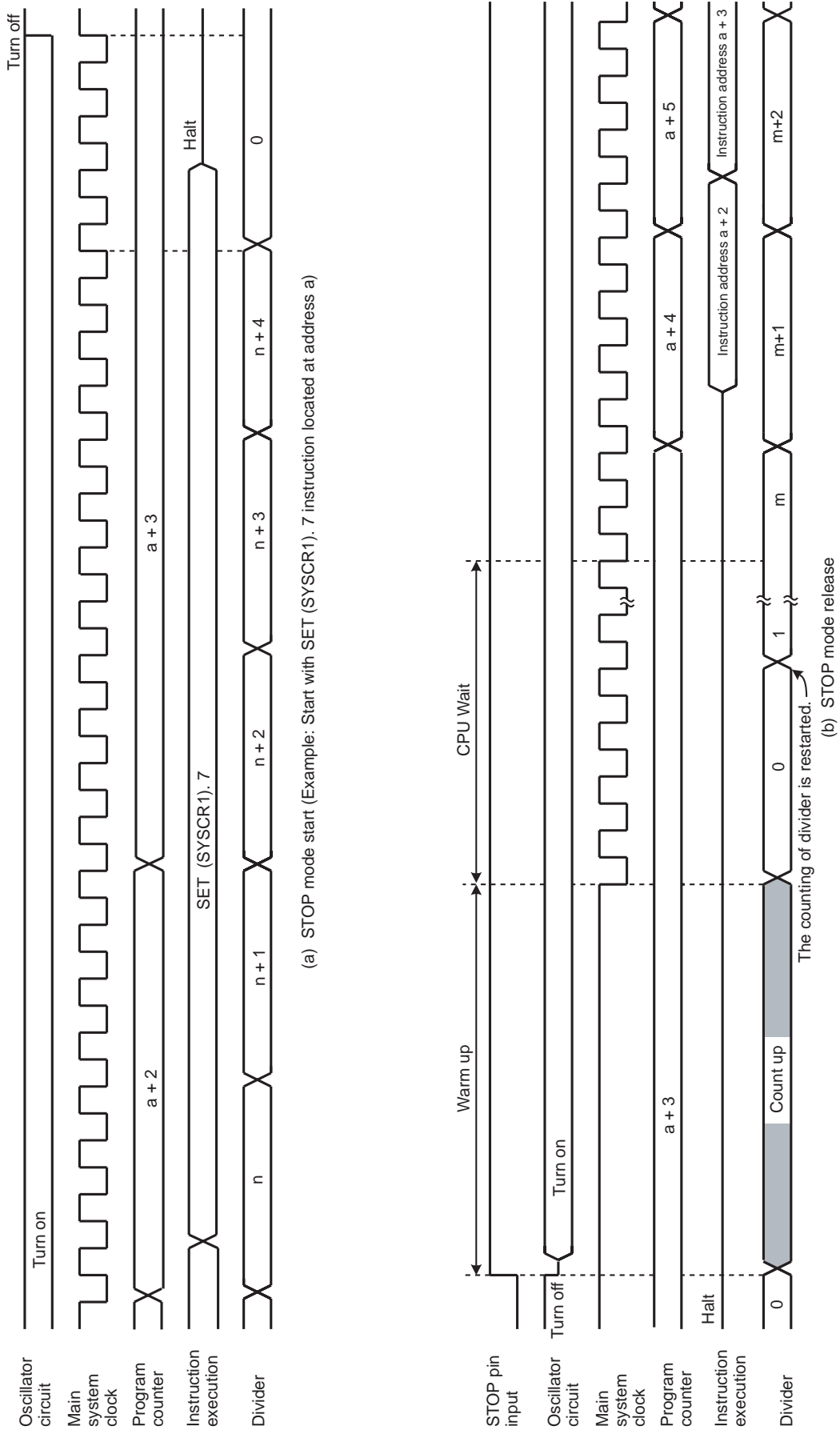
(b) STOP mode release

Figure 2-10 STOP Mode Start/Release (when EEPCR<MNPWDW> = "1")

TOSHIBA

2.2.4.2 IDLE1/2 mode and SLEEP1/2 mode

IDLE1/2 and SLEEP1/2 modes are controlled by the system control register 2 (SYSCR2) and maskable interrupts. The following status is maintained during these modes.

1. Operation of the CPU and watchdog timer (WDT) is halted. On-chip peripherals continue to operate.

2. The data memory, CPU registers, program status word and port output latches are all held in the status in effect before these modes were entered.

3. The program counter holds the address 2 ahead of the instruction which starts these modes.
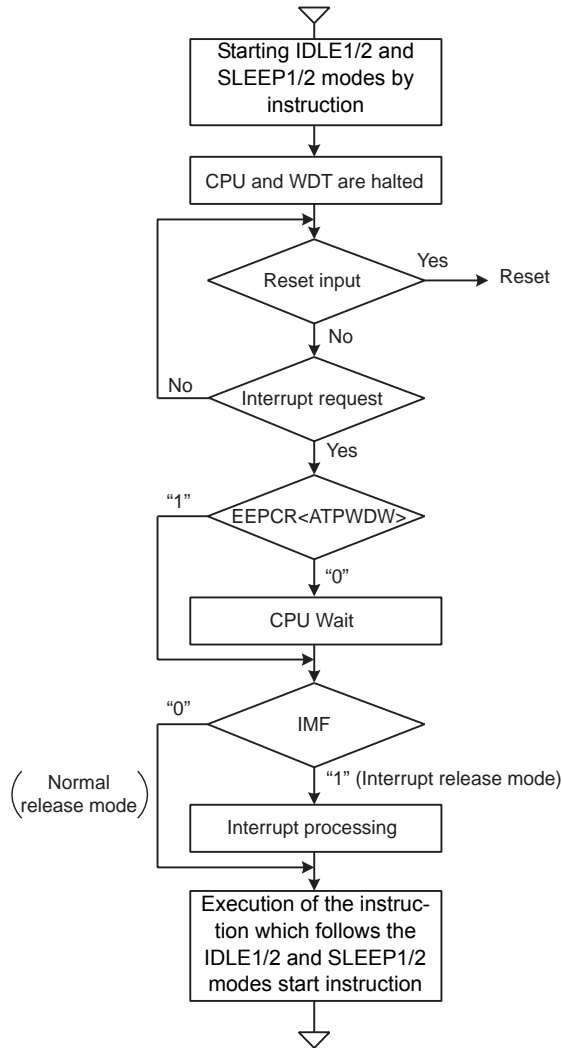
Figure 2-11   IDLE1/2 and SLEEP1/2 Modes

Note 1: EEPCR<ATPWDW> is a bit1 in EEPCR, which is a control bit of the power supply circuit for flash.

Note 2: During CPU wait, though CPU operations remain halted, the peripheral function operation is resumed. Therefore in this time, though the interrupt latch might be set, interrupt operation is not executed until the CPU wait is finished.

- Start the IDLE1/2 and SLEEP1/2 modes

     After IMF is set to "0", set the individual interrupt enable flag (EF) which releases IDLE1/2 and SLEEP1/2 modes. To start IDLE1/2 and SLEEP1/2 modes, set SYSCR2<IDLE> to "1".

- Release the IDLE1/2 and SLEEP1/2 modes

     IDLE1/2 and SLEEP1/2 modes include a normal release mode and an interrupt release mode. These modes are selected by interrupt master enable flag (IMF). After releasing IDLE1/2 and SLEEP1/2 modes, the SYSCR2<IDLE> is automatically cleared to "0" and the operation mode is returned to the mode preceding IDLE1/2 and SLEEP1/2 modes.

     When the IDLE1/2 and SLEEP1/2 modes are started with the EEPCR<ATPWDW> = "0", the CPU wait period for stabilizing of the power supply of flash control circuit is added before the operation mode is returned to the preceding modes. The CPU wait time of IDLE1/2 is $2^{10}/fc$ [s] and that of SLEEP1/2 mode is $2^3/fs$ [s].

     IDLE1/2 and SLEEP1/2 modes can also be released by inputting low level on the $\overline{\text{RESET}}$ pin. After releasing reset, the operation mode is started from NORMAL1 mode.

(1) Normal release mode (IMF = "0")

     IDLE1/2 and SLEEP1/2 modes are released by any interrupt source enabled by the individual interrupt enable flag (EF). After the interrupt is generated, the program operation is resumed from the instruction following the IDLE1/2 and SLEEP1/2 modes start instruction. Normally, the interrupt latches (IL) of the interrupt source used for releasing must be cleared to "0" by load instructions.

(2) Interrupt release mode (IMF = "1")

     IDLE1/2 and SLEEP1/2 modes are released by any interrupt source enabled with the individual interrupt enable flag (EF) and the interrupt processing is started. After the interrupt is processed, the program operation is resumed from the instruction following the instruction, which starts IDLE1/2 and SLEEP1/2 modes.

   Note: When a watchdog timer interrupts is generated immediately before IDLE1/2 and SLEEP1/2 modes are started, the watchdog timer interrupt will be processed but IDLE1/2 and SLEEP1/2 modes will not be started.
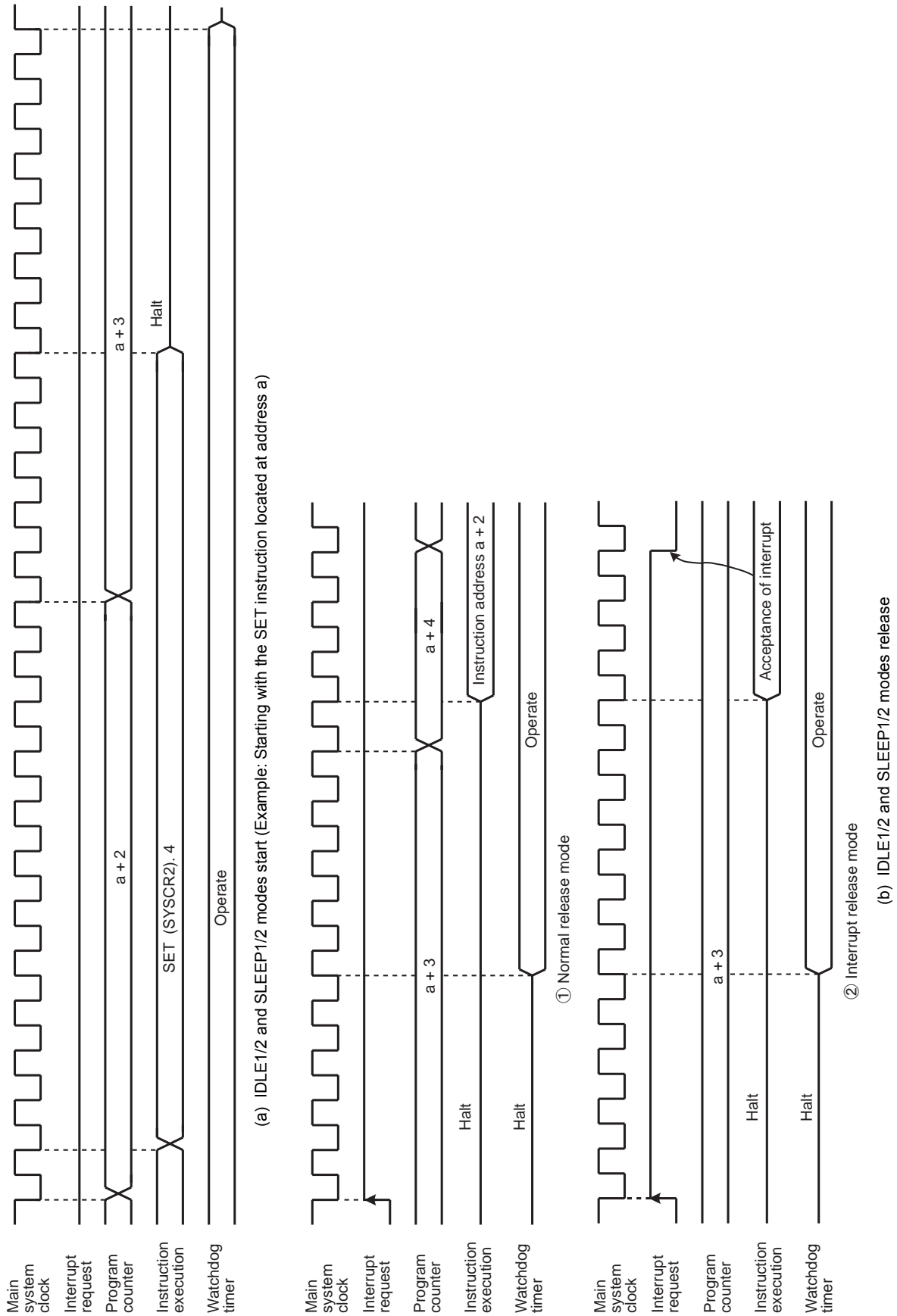
Figure 2-12 IDLE1/2 and SLEEP1/2 Modes Start/Release (when EEPCR<ATPWDW> = "1")

2.2.4.3 IDLE0 and SLEEP0 modes (IDLE0, SLEEP0)

IDLE0 and SLEEP0 modes are controlled by the system control register 2 (SYSCR2) and the time base timer control register (TBTCR). The following status is maintained during IDLE0 and SLEEP0 modes.

1. Timing generator stops feeding clock to peripherals except TBT.
2. The data memory, CPU registers, program status word and port output latches are all held in the status in effect before IDLE0 and SLEEP0 modes were entered.
3. The program counter holds the address 2 ahead of the instruction which starts IDLE0 and SLEEP0 modes.

Note: Before starting IDLE0 or SLEEP0 mode, be sure to stop (Disable) peripherals.
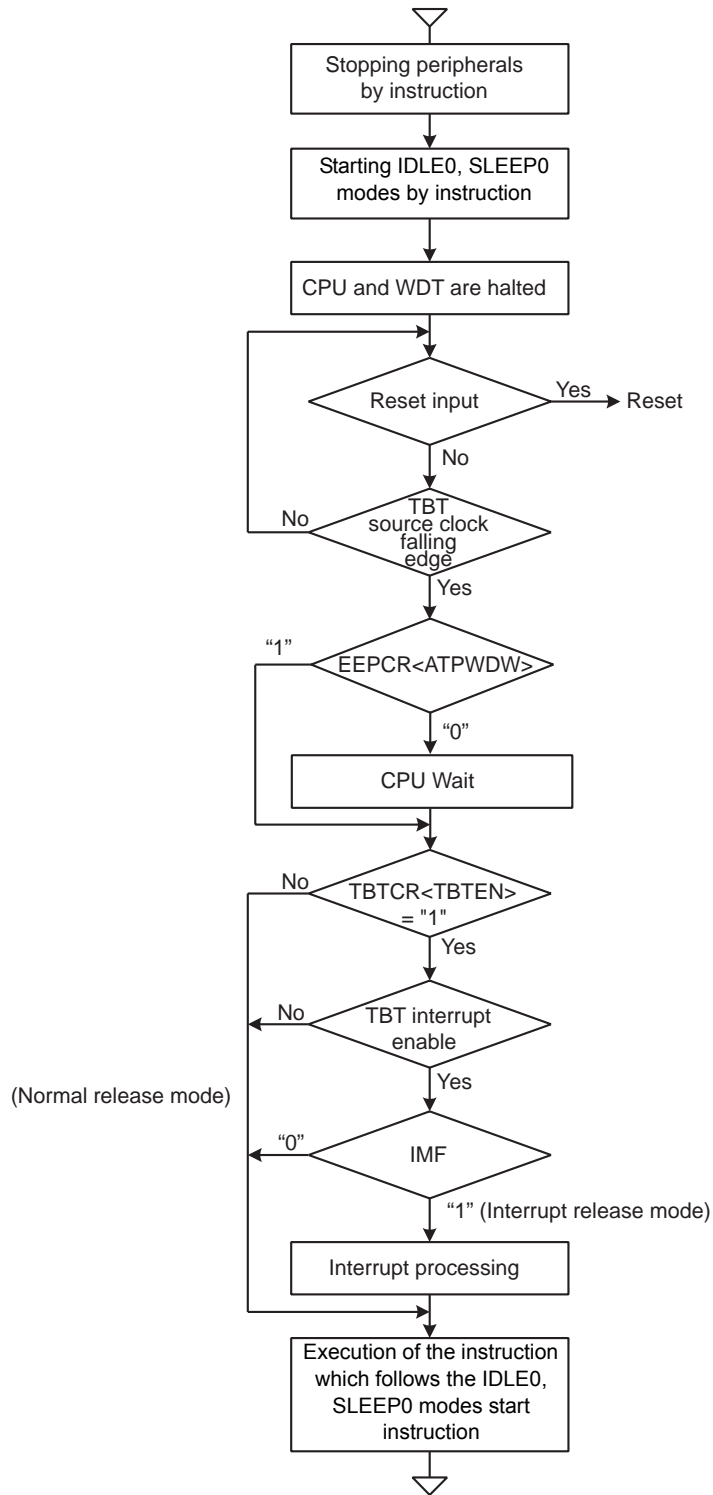
Figure 2-13   IDLE0 and SLEEP0 Modes

- Start the IDLE0 and SLEEP0 modes

    Stop (Disable) peripherals such as a timer counter.

    To start IDLE0 and SLEEP0 modes, set SYSCR2<TGHALT> to "1".

- Release the IDLE0 and SLEEP0 modes

    IDLE0 and SLEEP0 modes include a normal release mode and an interrupt release mode.

    These modes are selected by interrupt master flag (IMF), the individual interrupt enable flag of TBT and TBTCR<TBTEN>.

    After releasing IDLE0 and SLEEP0 modes, the SYSCR2<TGHALT> is automatically cleared to "0" and the operation mode is returned to the mode preceding IDLE0 and SLEEP0 modes. Before starting the IDLE0 or SLEEP0 mode, when the TBTCR<TBTEN> is set to "1", INTTBT interrupt latch is set to "1".

    When the IDLE0 and SLEEP0 modes are started with the EEPCR<ATPWDW> = "0", the CPU wait period for stabilizing of the power supply of flash control circuit is added before the operation mode is returned to the preceding modes. The CPU wait time of IDLE0 is $2^{10}/fc$ [s] and that of SLEEP0 mode is $2^3/fs$ [s].

    IDLE0 and SLEEP0 modes can also be released by inputting low level on the $\overline{\text{RESET}}$ pin. After releasing reset, the operation mode is started from NORMAL1 mode.

Note: IDLE0 and SLEEP0 modes start/release without reference to TBTCR<TBTEN> setting.

(1)   Normal release mode (IMF•EF6•TBTCR<TBTEN> = "0")

    IDLE0 and SLEEP0 modes are released by the source clock falling edge, which is setting by the TBTCR<TBTCK>. After the falling edge is detected, the program operation is resumed from the instruction following the IDLE0 and SLEEP0 modes start instruction. Before starting the IDLE0 or SLEEP0 mode, when the TBTCR<TBTEN> is set to "1", INTTBT interrupt latch is set to "1".

(2)   Interrupt release mode (IMF•EF6•TBTCR<TBTEN> = "1")

    IDLE0 and SLEEP0 modes are released by the source clock falling edge, which is setting by the TBTCR<TBTCK> and INTTBT interrupt processing is started.

Note 1: Because returning from IDLE0, SLEEP0 to NORMAL1, SLOW1 is executed by the asynchronous internal clock, the period of IDLE0, SLEEP0 mode might be the shorter than the period setting by TBTCR<TBTCK>.

Note 2: When a watchdog timer interrupt is generated immediately before IDLE0/SLEEP0 mode is started, the watchdog timer interrupt will be processed but IDLE0/SLEEP0 mode will not be started.
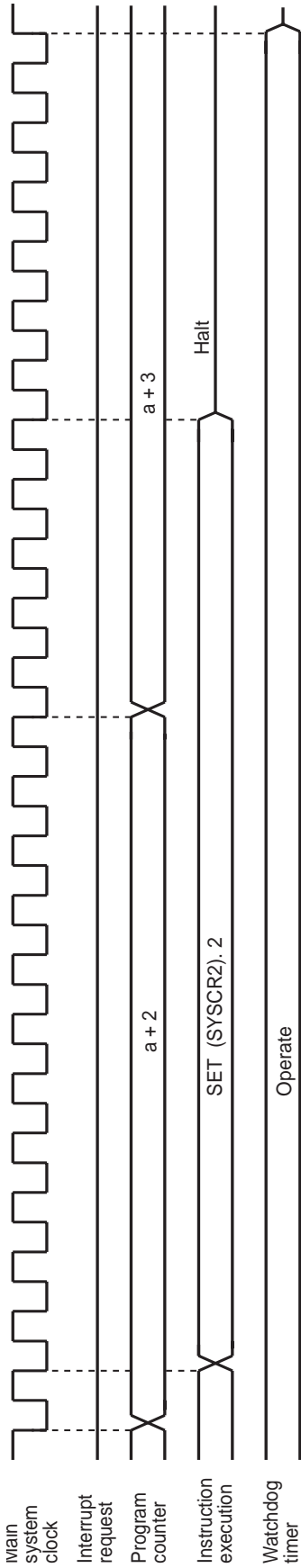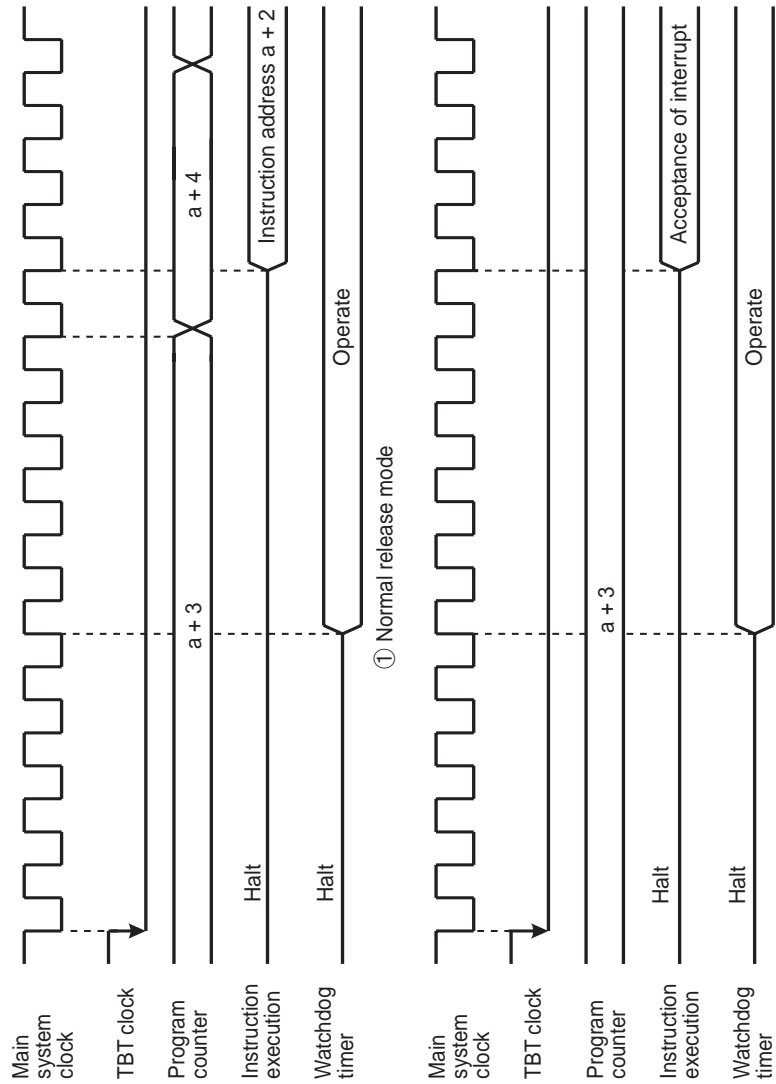
Figure 2-14 IDLE0 and SLEEP0 Modes Start/Release (when EEPCR<ATPWDW> = "1")

2.2.4.4 SLOW mode

SLOW mode is controlled by the system control register 2 (SYSCR2).

The following is the methods to switch the mode with the warm-up counter.

(1) Switching from NORMAL2 mode to SLOW1 mode

First, set SYSCR2<SYSCK> to switch the main system clock to the low-frequency clock for SLOW2 mode. Next, clear SYSCR2<XEN> to turn off high-frequency oscillation.

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

When the low-frequency clock oscillation is unstable, wait until oscillation stabilizes before performing the above operations. The timer/counter (TC4,TC3) can conveniently be used to confirm that low-frequency clock oscillation has stabilized.

Example 1 :Switching from NORMAL2 mode to SLOW1 mode.

```
        SET         (SYSCR2). 5        ; SYSCR2<SYSCK> ← 1
                                        (Switches the main system clock to the low-frequency
                                        clock for SLOW2)

        CLR         (SYSCR2). 7        ; SYSCR2<XEN> ← 0
                                        (Turns off high-frequency oscillation)
```

Example 2 :Switching to the SLOW1 mode after low-frequency clock has stabilized.

```
        SET         (SYSCR2). 6        ; SYSCR2<XTEN> ← 1

        LD          (TC3CR), 43H       ; Sets mode for TC4, 3 (16-bit mode, fs for source)

        LD          (TC4CR), 05H       ; Sets warming-up counter mode

        LDW         (TTREG3), 8000H    ; Sets warm-up time (Depend on oscillator accompanied)

        DI                             ; IMF ← 0

        SET         (EIRE). 1          ; Enables INTTC4

        EI                             ; IMF ← 1

        SET         (TC4CR). 3         ; Starts TC4, 3

            :

PINTTC4: CLR        (TC4CR). 3         ; Stops TC4, 3

        SET         (SYSCR2). 5        ; SYSCR2<SYSCK> ← 1
                                        (Switches the main system clock to the low-frequency clock)

        CLR         (SYSCR2). 7        ; SYSCR2<XEN> ← 0
                                        (Turns off high-frequency oscillation)

        RETI

            :

VINTTC4: DW         PINTTC4            ; INTTC4 vector table
```

(2)   Switching from SLOW1 mode to NORMAL2 mode

First, set SYSCR2<XEN> to turn on the high-frequency oscillation. When time for stabilization (Warm up) has been taken by the timer/counter (TC4,TC3), clear SYSCR2<SYSCK> to switch the main system clock to the high-frequency clock.

SLOW mode can also be released by inputting low level on the $\overline{RESET}$ pin. After releasing reset, the operation mode is started from NORMAL1 mode.

Note: After SYSCK is cleared to "0", executing the instructions is continiued by the low-frequency clock for the period synchronized with low-frequency and high-frequency clocks.

High-frequency clock
Low-frequency clock
Main system clock
SYSCK
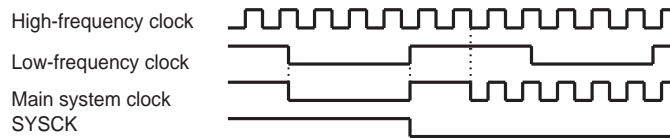
Example :Switching from the SLOW1 mode to the NORMAL2 mode (fc = 16 MHz, warm-up time is 4.0 ms).

```
            SET       (SYSCR2). 7          ; SYSCR2<XEN> ← 1 (Starts high-frequency oscillation)

            LD        (TC3CR), 63H         ; Sets mode for TC4, 3 (16-bit mode, fc for source)

            LD        (TC4CR), 05H         ; Sets warming-up counter mode

            LD        (TTREG4), 0F8H       ; Sets warm-up time

            DI                             ; IMF ← 0

            SET       (EIRE). 1            ; Enables INTTC4

            EI                             ; IMF ← 1

            SET       (TC4CR). 3           ; Starts TC4, 3

              :

PINTTC4:    CLR       (TC4CR). 3           ; Stops TC4, 3

            CLR       (SYSCR2). 5          ; SYSCR2<SYSCK> ← 0
                                             (Switches the main system clock to the high-frequency clock)

            RETI

              :

VINTTC4:    DW        PINTTC4              ; INTTC4 vector table
```
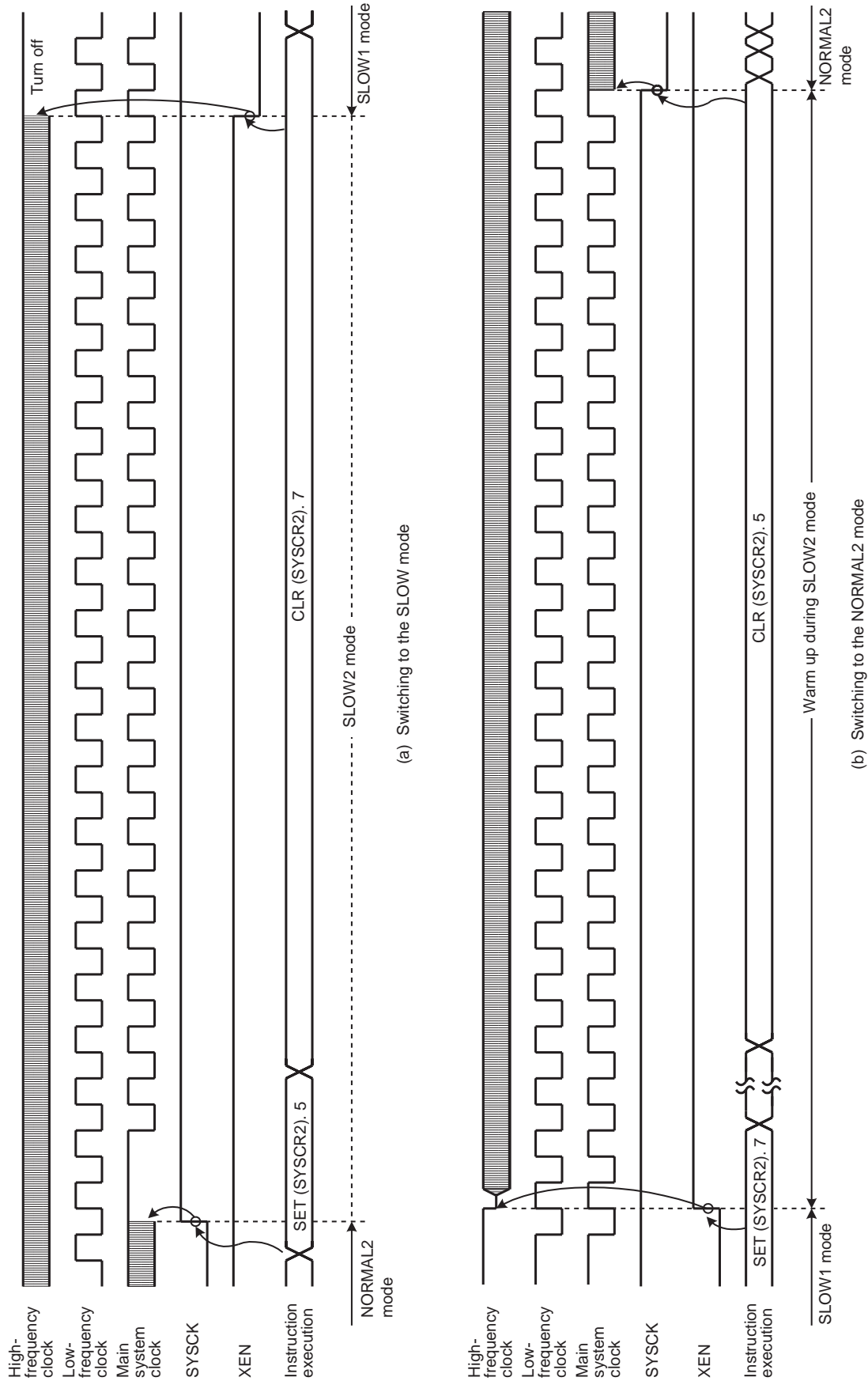
Figure 2-15 Switching between the NORMAL2 and SLOW Modes

2.3 Reset Circuit

The TMP86FM26UG has five types of reset generation procedures: An external reset input, an address trap reset, a watchdog timer reset and a system clock reset. Of these reset, the address trap reset, the watchdog timer and the system clock reset are a malfunction reset. When the malfunction reset request is detected, reset occurs during the maximum 24/fc[s].

Also a reset circuit has an 11-stage counter for generation of flash reset, and the flash reset occurs immediately after the malfunction reset and the external reset operation. The flash reset period is 2^{10}/fc [s] (64μs at 16.0MHz).

Therefore, the maximum reset period is 24/fc [s] + 2^{10}/fc [s] (65.5μs at 16.0MHz).

The malfunction reset circuit such as watchdog timer reset, address trap reset and system clock reset is not initialized when power is turned on. Therefore, reset may occur during maximum 24/fc + 2^{10}/fc[s] (65.5μs at 16.0 MHz) when power is turned on.

Table 2-3 shows on-chip hardware initialization by reset action.

Table 2-3   Initializing Internal Status by Reset Action

On-chip Hardware		Initial Value	On-chip Hardware	Initial Value
Program counter	(PC)	(FFFEH)	Prescaler and divider of timing generator	0
Stack pointer	(SP)	Not initialized		
General-purpose registers (W, A, B, C, D, E, H, L, IX, IY)		Not initialized		
Jump status flag	(JF)	Not initialized	Watchdog timer	Enable
Zero flag	(ZF)	Not initialized	Output latches of I/O ports	Refer to I/O port circuitry
Carry flag	(CF)	Not initialized		
Half carry flag	(HF)	Not initialized		
Sign flag	(SF)	Not initialized		
Overflow flag	(VF)	Not initialized		
Interrupt master enable flag	(IMF)	0		
Interrupt individual enable flags	(EF)	0	Control registers	Refer to each of control register
Interrupt latches	(IL)	0		
			LCD data buffer	Not initialized
			RAM	Not initialized

2.3.1 External Reset Input

The $\overline{\text{RESET}}$ pin contains a Schmitt trigger (Hysteresis) with an internal pull-up resistor.

When the $\overline{\text{RESET}}$ pin is held at "L" level 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 high level goes on during 2^{10}/fc[s] (65.5μs at 16MHz) after 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 FFFEH to FFFFH.
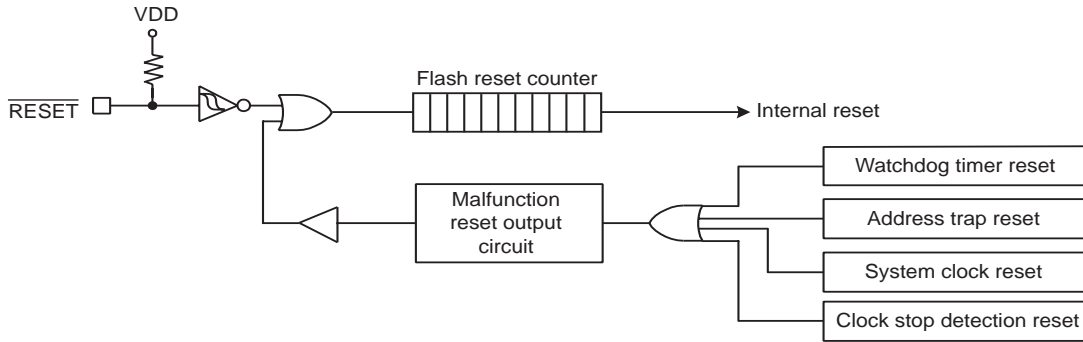
Figure 2-16   Reset Circuit

2.3.2   Address trap reset

If the CPU should start looping for some cause such as noise and an attempt be made to fetch an instruction from the on-chip RAM (when WDTCR1<ATAS> is set to "1"), DBR or the SFR area, address trap reset will be generated. The reset time is maximum $24/fc + 2^{10}/fc$[s] (65.5μs at 16.0 MHz).

Note:The operating mode under address trapped is alternative of reset or interrupt. The address trap area is alternative.
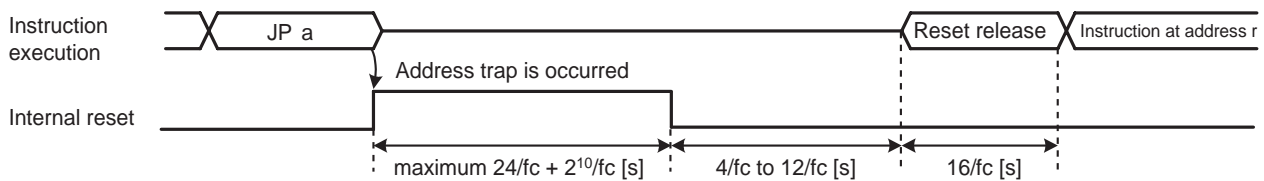
Note 1: Address "a" is in the SFR, DBR or on-chip RAM (WDTCR1<ATAS> = "1") space.
Note 2: During reset release, reset vector "r" is read out, and an instruction at address "r" is fetched and decoded.

Figure 2-17 Address Trap Reset

2.3.3   Watchdog timer reset

Refer to Section "Watchdog Timer".

2.3.4   System clock reset

If the condition as follows is detected, the system clock reset occurs automatically to prevent dead lock of the CPU. (The oscillation is continued without stopping.)

- In case of clearing SYSCR2<XEN> and SYSCR2<XTEN> simultaneously to "0".
- In case of clearing SYSCR2<XEN> to "0", when the SYSCR2<SYSCK> is "0".
- In case of clearing SYSCR2<XTEN> to "0", when the SYSCR2<SYSCK> is "1".

The flash reset occurs immediately after the system clock reset operation. The reset time is maximum $24/fc + 2^{10}/fc$ (65.5 μs at 16.0 MHz).

2.3.5 Clock Stop Detection Reset

Refer to Section "Clock Stop Detection Circuit".

2.3.6 Internal Reset Detection Flags

After an internal reset is released, the cause of this internal reset can be identified by reading the internal reset detection flag register (IRSCR). IRSCR<SYSRSF> corresponds to system clock reset, IRSCR<ADTRF> to address trap reset, IRSCR<WDTF> to watchdog timer reset, and IRSCR<CLKSF> to clock stop detection reset. Each of these bits is set to "1" when the corresponding reset is generated.

To clear IRSCR<SYSRSF, ADTRF, WDTF, CLKSF> to "0", write "1" in IRSCR<FCLR> or set the $\overline{\text{RESET}}$ pin (external reset) to "L" level.

Internal Reset Detection Flag Register

IRSCR (0FB5H)	7	6	5	4	3	2	1	0	
	FCLR				SYSRSF	ADTRF	WDTF	CLKSF	(Initial value: 0*** 0000)
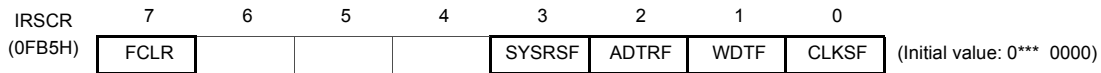
FCLR	Flag initialization control	0: - 1: Clear SYSRSF, ADTRF, WDTF, and CLKSF flags to "0".		R/W
SYSRSF	System clock reset flag	0: System clock reset not detected 1: System clock reset detected		
ADTRF	Address trap reset flag	0: Address trap reset not detected 1: Address trap reset detected		Read only
WDTF	Watchdog timer reset flag	0: Watchdog timer reset not detected 1: Watchdog timer reset detected		
CLKSF	Clock stop detection reset flag	0: Clock stop detection reset not detected 1: Clock stop detection reset detected		

Note 1: FCLR is automatically cleared to "0" after being set to "1".

Note 2: SYSRSF, ADTRF, WDTF, and CLKSF are not initialized by an internal reset. To initialize these flags, write "1" in FCLR or set the $\overline{\text{RESET}}$ pin (external reset) to "L" level.

2.4 Clock Stop Detection Circuit

The clock stop detection circuit generates a clock stop detection reset if either one of the high-frequency and low-frequency clocks stopped or became unstable. A clock stop detection reset is generated when the frequency ratio of the high-frequency clock to the low-frequency clock goes out of the range specified by the maximum and minimum values. This reset is released when the frequency ratio returns to the specified range again.

2.4.1 Configuration
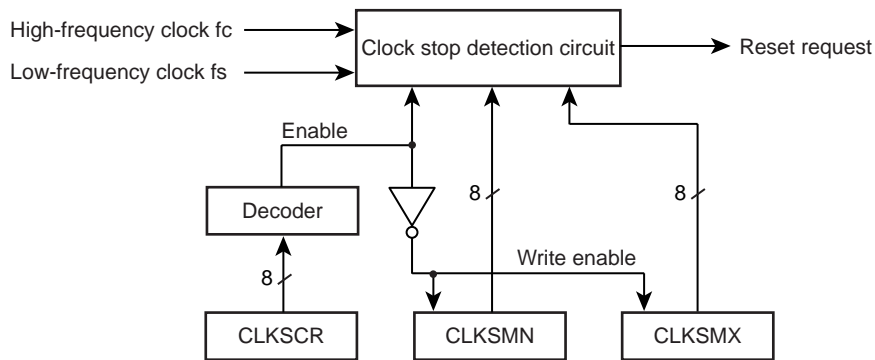
Figure 2-18 Configuration of Clock Stop Detection Circuit

2.4.2 Control

The clock stop detection circuit is controlled by the clock stop detection control register (CLKSCR), the clock stop minimum value compare register (CLKSMN), and the clock stop maximum value compare register (CLKSMX).
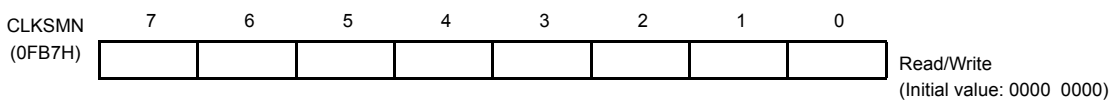
Clock Stop Detection Control Register

CLKSCR (0FB6H)	7	6	5	4	3	2	1	0	(Initial value: 0000 0000)

CLKSCR	Clock stop detection circuit operation control	00H: Disable E4H: Enable Others: Reserved	R/W

Note 1: When SYSCR2<XEN> is set to "0", CLKSCR is automatically cleared to "00H" to disable clock stop detection.

Note 2: When STOP, IDLE0, or SLEEP0 mode is started, CLKSCR is automatically cleared to "00H" to disable clock stop detection.

Note 3: CLKSCR can be set to "E4H" only when both SYSCR2<XEN> and SYSCR2<XTEN> are "1". When either SYSCR2<XEN> or SYSCR2<XTEN> is "0", any attempt to change CLKSCR from "00H" to "E4H" is ineffective.

Note 4: CLKSCR is not initialized by an internal reset. To initialize this register, set the $\overline{\text{RESET}}$ pin (external reset) to "L" level.

Note 5: When SYSCR2<XEN, XTEN> = "1" and CLKSCR = "E4H", the low-frequency clock continues oscillating even if SYSCR2<XTEN> is set to "0" or even if an internal reset is generated. (Dual-clock operation continues during reset operation and after reset release.)

Clock Stop Minimun Value Compare Register

CLKSMN (0FB7H)	7	6	5	4	3	2	1	0	Read/Write (Initial value: 0000 0000)

Clock Stop Maximun Value Compare Register

CLKSMX	7	6	5	4	3	2	1	0	
(0FB8H)									Read/Write

(Initial value: 1111 1111)

Note 1: CLKSMN and CLKSMX cannot be written while the clock stop detection circuit is enabled. If these registers are written while clock stop detection is enabled, the values written to these registers are not reflected.

Note 2: Set CLKSMN and CLKSMX according to the clock frequencies to be used. At this time, make sure that CLKSMN is smaller than CLKSMX. For how to calculate the values to be set in these registers, see "2.4.2.1 Setting the minimum and maximum values for clock stop detection" below.

Note 3: CLKSMN and CLKSMX are not initialized by an internal reset. To initialize these registers, set the $\overline{\text{RESET}}$ pin (external reset) to "L" level.

2.4.2.1 Setting the minimum and maximum values for clock stop detection

The CLKSMN and CLKSMX registers are used to determine a clock stop/unstable condition. The values to be set in CLKSMN and CLKSMX are respectively obtained by calculating the minimum and maximum values of the frequency ratio of the high-frequency clock (fc) to the low-frequency clock (fs) including deviation. Basically, CLKSMN should be set to the minimum value of the frequency ratio divided by four (fractions to be dropped), and CLKSMX should be set to the maximum value of the frequency ratio divided by four (fractions to be rounded up). The equations for obtaining the CLKSMN and CLKSMX values and example settings are shown below.

(Equations)

$$\text{CLKSMN value} = \frac{\text{fc} \times (1 - (\Delta\text{f})/\text{fc})}{\text{fs} \times (1 + (\Delta\text{f})/\text{fs}) \times 4} \qquad \text{(Fractions to be dropped)}$$

$$\text{CLKSMX value} = \frac{\text{fc} \times 1 + (\Delta\text{f})/\text{fc}}{\text{fs} \times (1 - (\Delta\text{f})/\text{fs}) \times 4} \qquad \text{(Fractions to be rounded up)}$$

fc	: High-frequency clock frequency
Δf/fc	: High-frequency clock frequency deviation
fs	: Low-frequency clock frequency
Δf/fs	: Low-frequency clock frequency deviation

(Setting Examples)

Conditions	fc	: 8 MHz
	Δf/fc	: ± 10%
	fs	: 32.768 MHz
	Δf/fs	: ± 1%

$$\text{CLKSMN value} = \frac{8 \times 10^6 \times (1 - 0.1)}{32.768 \times 10^3 \times (1 + 0.01) \times 4}$$

(Nearly Equal) = 54 (36H)

$$\text{CLKSMX value} = \frac{8 \times 10^6 \times (1 + 0.1)}{32.768 \times 10^3 \times (1 - 0.01) \times 4}$$

(Nearly Equal) = 68 (44H)

When the clock stop detection circuit is enabled, even a slight deviation from the clock frequency ratio range set by CLKSMN and CLKSMX will generate a clock stop detection reset. Therefore, be sure to allow sufficient margins when setting the CLKSMN and CLKSMX values.

CLKSMN and CLKSMX cannot be written when the clock stop detection circuit is enabled. These registers must be set before the clock stop detection circuit is enabled. CLKSMN and CLKSMX are not initialized by an internal reset. To initialize these registers, set the $\overline{\text{RESET}}$ pin (external reset) to "L" level.

2.4.2.2　Enabling/Disabling the clock stop detection circuit

The clock stop detection circuit is enabled by writing "E4H" in CLKSCR and it is disabled by writing "00H" in CLKSCR. Only when SYSCR2<XEN> = "1" and SYSCR2<XTEN> = "1" can CLKSCR be written. In addition, make sure that the high-frequency and low-frequency clocks are oscillating stably before writing to CLKSCR.

When STOP, IDLE0, or SLEEP0 mode is started while the clock stop detection circuit is enabled, CLKSCR is cleared to "00H" and the clock stop detection circuit is disabled. To enable the clock stop detection circuit again after STOP, IDLE0, or SLEEP0 mode is exited, write "E4H" in CLKSCR. While the clock stop detection circuit is enabled, setting SYSCR2<XEN> to "0" also clears CLKSCR to "00H".

CLKSCR is not cleared by an internal reset. To initialize this register, set the $\overline{\text{RESET}}$ pin (external reset) to "L" level.

2.4.2.3 Generating and releasing a reset

When a clock stop condition is detected, a clock stop detection reset is generated not immediately but after two rising edges of the low-frequency clock. Therefore, the reset generation timing of the clock stop detection reset differs between the high-frequency clock and the low-frequency clock.

When the high-frequency clock stops and the frequency ratio goes out of the range specified by CLKSMN and CLKSMX, a clock stop detection reset is generated after two rising edges of the low-frequency clock. During the reset, the clock stop detection operation continues. The reset is released when the clock resumes stable operation and the frequency ratio returns to the specified range again.

When a stop condition is detected in the low-frequency clock, a clock stop detection reset is generated only after the low-frequency clock resumes oscillation and the second rising edge of the low-frequency clock is detected. This means that no reset will be generated if the low-frequency clock stops completely.

The clock stop detection reset is followed by a flash reset. The maximum reset period is $24/\text{fc} + 2^{10}/\text{fc}$ [s] (65.5 μs at 16.0 MHz).

Note: If the high-frequency or low-frequency clock temporarily goes out of the specified frequency ratio range due to such causes as noise, a clock stop detection reset is also generated.

· When the high-frequency clock stops
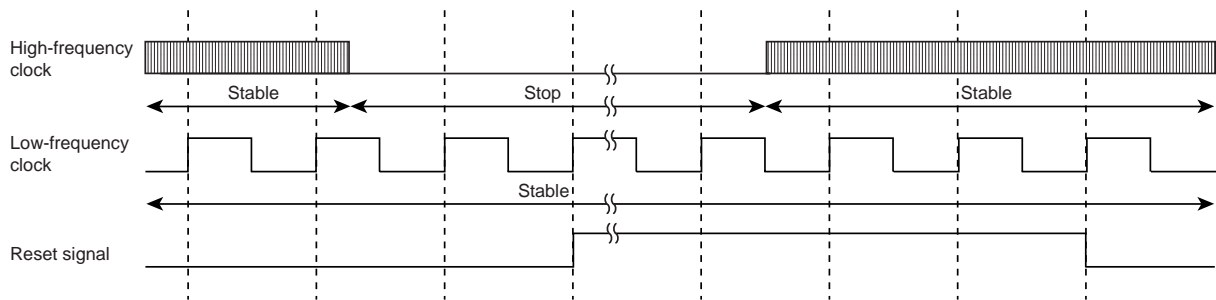
· When the low-frequency clock stops
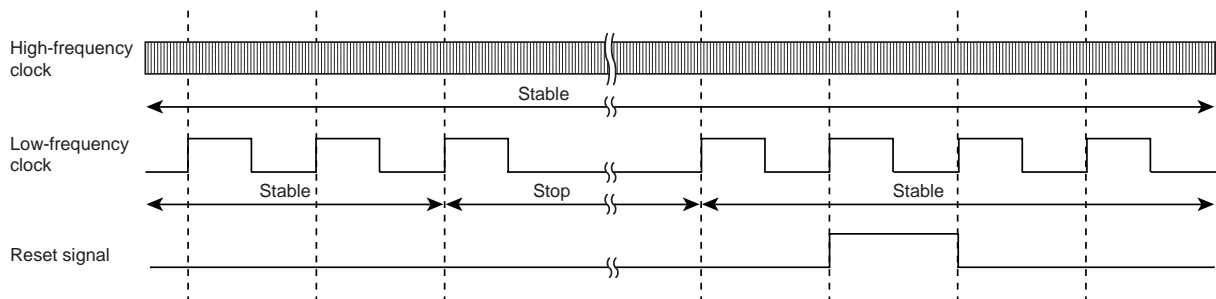
Figure 2-19 Clock Stop Detection Timing

3. Interrupt Control Circuit

The TMP86FM26UG has a total of 21 interrupt sources excluding reset. Interrupts can be nested with priorities. Four of the internal interrupt sources are non-maskable while the rest are maskable.

Interrupt sources are provided with interrupt latches (IL), which hold interrupt requests, and independent vectors. The interrupt latch is set to "1" by the generation of its interrupt request which requests the CPU to accept its interrupts. Interrupts are enabled or disabled by software using the interrupt master enable flag (IMF) and interrupt enable flag (EF). If more than one interrupts are generated simultaneously, interrupts are accepted in order which is dominated by hardware. However, there are no prioritized interrupt factors among non-maskable interrupts.

Interrupt Factors		Enable Condition	Interrupt Latch	Vector Address	Priority
Internal/External	(Reset)	Non-maskable	–	FFFE	1
Internal	INTSWI (Software interrupt)	Non-maskable	–	FFFC	2
Internal	INTUNDEF (Executed the undefined instruction interrupt)	Non-maskable	–	FFFC	2
Internal	INTATRAP (Address trap interrupt)	Non-maskable	IL2	FFFA	2
Internal	INTWDT (Watchdog timer interrupt)	Non-maskable	IL3	FFF8	2
External	$\overline{INT0}$	IMF• EF4 = 1, INT0EN = 1	IL4	FFF6	5
External	INT1	IMF• EF5 = 1	IL5	FFF4	6
Internal	INTTBT	IMF• EF6 = 1	IL6	FFF2	7
External	INT2	IMF• EF7 = 1	IL7	FFF0	8
Internal	INTTC1	IMF• EF8 = 1	IL8	FFEE	9
Internal	INTRXD0	IMF• EF9 = 1	IL9	FFEC	10
Internal	INTTXD0	IMF• EF10 = 1	IL10	FFEA	11
Internal	INTRXD1	IMF• EF11 = 1	IL11	FFE8	12
Internal	INTTXD1	IMF• EF12 = 1	IL12	FFE6	13
External	INT3	IMF• EF13 = 1	IL13	FFE4	14
External	INT4	IMF• EF14 = 1	IL14	FFE2	15
Internal	INTSIO	IMF• EF15 = 1	IL15	FFE0	16
Internal	INTTC3	IMF• EF16 = 1	IL16	FFBE	17
Internal	INTTC4	IMF• EF17 = 1	IL17	FFBC	18
Internal	INTTC5	IMF• EF18 = 1	IL18	FFBA	19
Internal	INTTC6	IMF• EF19 = 1	IL19	FFB8	20
-	Reserved (Note: Refer to Chapter of Real-Time Clock)	IMF• EF20 = 1	IL20	FFB6	21
External	$\overline{INT5}$	IMF• EF21 = 1	IL21	FFB4	22
-	Reserved	IMF• EF22 = 1	IL22	FFB2	23
-	Reserved	IMF• EF23 = 1	IL23	FFB0	24

Note 1: To use the address trap interrupt (INTATRAP), clear WDTCR1<ATOUT> to "0" (It is set for the "reset request" after reset is cancelled). For details, see "Address Trap".

Note 2: To use the watchdog timer interrupt (INTWDT), clear WDTCR1<WDTOUT> to "0" (It is set for the "Reset request" after reset is released). For details, see "Watchdog Timer".

3.1 Interrupt latches (IL21 to IL2)

An interrupt latch is provided for each interrupt source, except for a software interrupt and an executed the undefined instruction interrupt. When interrupt request is generated, the latch is set to "1", and the CPU is requested to accept the interrupt if its interrupt is enabled. The interrupt latch is cleared to "0" immediately after accepting interrupt. All interrupt latches are initialized to "0" during reset.

The interrupt latches are located on address 002EH, 003CH and 003DH in SFR area. Each latch can be cleared to "0" individually by instruction. However, IL2 and IL3 should not be cleared to "0" by software. For clearing the interrupt latch, load instruction should be used and then IL2 and IL3 should be set to "1". If the read-modify-write instructions such as bit manipulation or operation instructions are used, interrupt request would be cleared inadequately if interrupt is requested while such instructions are executed.

Interrupt latches are not set to "1" by an instruction.

Since interrupt latches can be read, the status for interrupt requests can be monitored by software.

Note: In main program, before manipulating the interrupt enable flag (EF) or the interrupt latch (IL), be sure to clear IMF to "0" (Disable interrupt by DI instruction). Then set IMF newly again as required after operating on the EF or IL (Enable interrupt by EI instruction)
In interrupt service routine, because the IMF becomes "0" automatically, clearing IMF need not execute normally on interrupt service routine. However, if using multiple interrupt on interrupt service routine, manipulating EF or IL should be executed before setting IMF="1".

Example 1 :Clears interrupt latches

```
DI                                           ; IMF ← 0
LDW        (ILL), 1110100000111111B          ; IL12, IL10 to IL6 ← 0
EI                                           ; IMF ← 1
```

Example 2 :Reads interrupt latchess

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

Example 3 :Tests interrupt latches

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

3.2   Interrupt enable register (EIR)

The interrupt enable register (EIR) enables and disables the acceptance of interrupts, except for the non-maskable interrupts (Software interrupt, undefined instruction interrupt, address trap interrupt and watchdog interrupt). Non-maskable interrupt is accepted regardless of the contents of the EIR.

The EIR consists of an interrupt master enable flag (IMF) and the individual interrupt enable flags (EF). These registers are located on address 002CH, 003AH and 003BH in SFR area, and they can be read and written by an instructions (Including read-modify-write instructions such as bit manipulation or operation instructions).

3.2.1   Interrupt master enable flag (IMF)

The interrupt enable register (IMF) enables and disables the acceptance of the whole maskable interrupt. While IMF = "0", all maskable interrupts are not accepted regardless of the status on each individual interrupt enable flag (EF). By setting IMF to "1", the interrupt becomes acceptable if the individuals are enabled. When an interrupt is accepted, IMF is cleared to "0" after the latest status on IMF is stacked. Thus the maskable interrupts which follow are disabled. By executing return interrupt instruction [RETI/RETN], the stacked data, which was the status before interrupt acceptance, is loaded on IMF again.

The IMF is located on bit0 in EIRL (Address: 003AH in SFR), and can be read and written by an instruction. The IMF is normally set and cleared by [EI] and [DI] instruction respectively. During reset, the IMF is initialized to "0".

3.2.2 Individual interrupt enable flags (EF21 to EF4)

Each of these flags enables and disables the acceptance of its maskable interrupt. Setting the corresponding bit of an individual interrupt enable flag to "1" enables acceptance of its interrupt, and setting the bit to "0" disables acceptance. During reset, all the individual interrupt enable flags (EF21 to EF4) are initialized to "0" and all maskable interrupts are not accepted until they are set to "1".

Note: In main program, before manipulating the interrupt enable flag (EF) or the interrupt latch (IL), be sure to clear IMF to "0" (Disable interrupt by DI instruction). Then set IMF newly again as required after operating on the EF or IL (Enable interrupt by EI instruction)
In interrupt service routine, because the IMF becomes "0" automatically, clearing IMF need not execute normally on interrupt service routine. However, if using multiple interrupt on interrupt service routine, manipulating EF or IL should be executed before setting IMF="1".

Example 1 : Enables interrupts individually and sets IMF

```
        DI                                  ; IMF ← 0

        LDW        (EIRL), 1110100010100000B ; EF15 to EF13, EF11, EF7, EF5 ← 1
          :                                    Note: IMF should not be set.

          :

        EI                                  ; IMF ← 1
```

Example 2 : C compiler description example

```
        unsigned  int _io (3AH)  EIRL;        /* 3AH shows EIRL address */

        _DI();

        EIRL = 10100000B;

          :

        _EI();
```

Interrupt Latches

(Initial value: 00000000 000000**)

ILH,ILL	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
(003DH, 003CH)	IL15	IL14	IL13	IL12	IL11	IL10	IL9	IL8	IL7	IL6	IL5	IL4	IL3	IL2		

ILH (003DH) ILL (003CH)

(Initial value: 00000000)

ILE						7	6	5	4	3	2	1	0
(002EH)						IL23	IL22	IL21	IL20	IL19	IL18	IL17	IL16
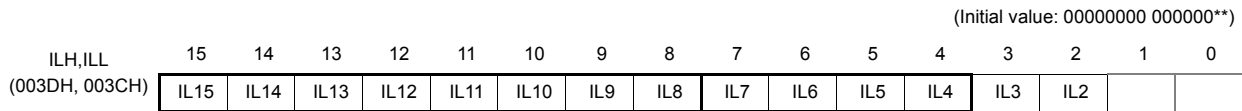
ILE (002EH)

IL21 to IL2	Interrupt latches	at RD 0: No interrupt request 1: Interrupt request	at WR 0: Clears the interrupt request 1: (Interrupt latch is not set.)	R/W

Note 1: To clear any one of bits IL7 to IL4, be sure to write "1" into IL2 and IL3.

Note 2: In main program, before manipulating the interrupt enable flag (EF) or the interrupt latch (IL), be sure to clear IMF to "0" (Disable interrupt by DI instruction). Then set IMF newly again as required after operating on the EF or IL (Enable interrupt by EI instruction)
In interrupt service routine, because the IMF becomes "0" automatically, clearing IMF need not execute normally on interrupt service routine. However, if using multiple interrupt on interrupt service routine, manipulating EF or IL should be executed before setting IMF="1".

Note 3: Do not clear IL with read-modify-write instructions such as bit operations.

Interrupt Enable Registers

(Initial value: 00000000 0000***0)

EIRH,EIRL	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
(003BH, 003AH)	EF15	EF14	EF13	EF12	EF11	EF10	EF9	EF8	EF7	EF6	EF5	EF4				IMF

EIRH (003BH) EIRL (003AH)

(Initial value: 00000000)

EIRE						7	6	5	4	3	2	1	0
(002CH)						EF23	EF22	EF21	EF20	EF19	EF18	EF17	EF16
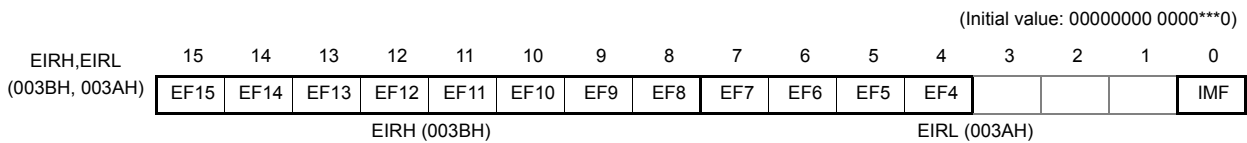
EIRE (002CH)

EF21 to EF4	Individual-interrupt enable flag (Specified for each bit)	0: Disables the acceptance of each maskable interrupt. 1: Enables the acceptance of each maskable interrupt.	R/W
IMF	Interrupt master enable flag	0: Disables the acceptance of all maskable interrupts 1: Enables the acceptance of all maskable interrupts	

Note 1: *: Don't care

Note 2: Do not set IMF and the interrupt enable flag (EF15 to EF4) to "1" at the same time.

Note 3: In main program, before manipulating the interrupt enable flag (EF) or the interrupt latch (IL), be sure to clear IMF to "0" (Disable interrupt by DI instruction). Then set IMF newly again as required after operating on the EF or IL (Enable interrupt by EI instruction)
In interrupt service routine, because the IMF becomes "0" automatically, clearing IMF need not execute normally on interrupt service routine. However, if using multiple interrupt on interrupt service routine, manipulating EF or IL should be executed before setting IMF="1".

3.3 Interrupt Sequence

An interrupt request, which raised interrupt latch, is held, until interrupt is accepted or interrupt latch is cleared to "0" by resetting or an instruction. Interrupt acceptance sequence requires 8 machine cycles (2 μs @16 MHz) after the completion of the current instruction. The interrupt service task terminates upon execution of an interrupt return instruction [RETI] (for maskable interrupts) or [RETN] (for non-maskable interrupts). Figure 3-1 shows the timing chart of interrupt acceptance processing.

3.3.1 Interrupt acceptance processing is packaged as follows.

a. The interrupt master enable flag (IMF) is cleared to "0" in order to disable the acceptance of any following interrupt.

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

c. The contents of the program counter (PC) and the program status word, including the interrupt master enable flag (IMF), are saved (Pushed) on the stack in sequence of PSW + IMF, PCH, PCL. Meanwhile, the stack pointer (SP) is decremented by 3.

d. The entry address (Interrupt vector) of the corresponding interrupt service program, loaded on the vector table, is transferred to the program counter.

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

Note: When the contents of PSW are saved on the stack, the contents of IMF are also saved.
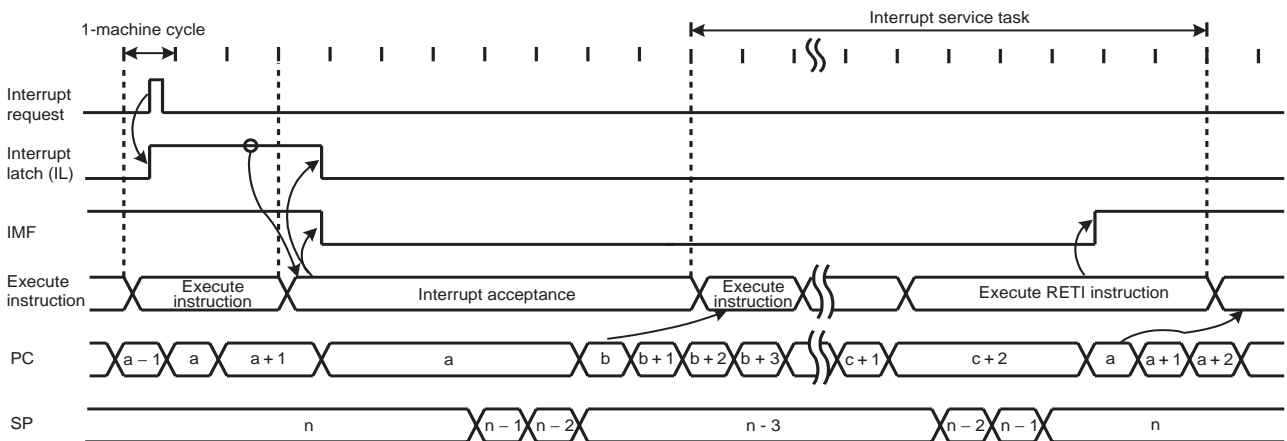
Note 1: a: Return address entry address, b: Entry address, c: Address which RETI instruction is stored

Note 2: On condition that interrupt is enabled, it takes 38/fc [s] or 38/fs [s] at maximum (If the interrupt latch is set at the first machine cycle on 10 cycle instruction) to start interrupt acceptance processing since its interrupt latch is set.

Figure 3-1 Timing Chart of Interrupt Acceptance/Return Interrupt Instruction

Example: Correspondence between vector table address for INTTBT and the entry address of the interrupt service program
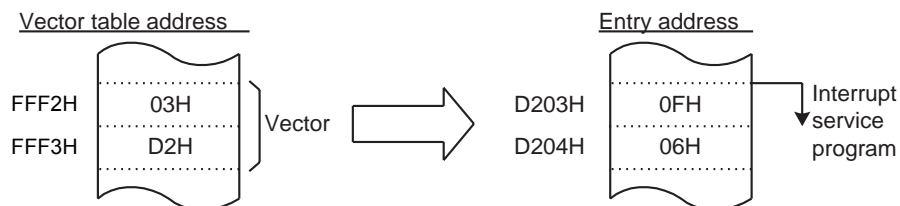
Figure 3-2 Vector table address,Entry address

A maskable interrupt is not accepted until the IMF is set to "1" even if the maskable interrupt higher than the level of current servicing interrupt is requested.

In order to utilize nested interrupt service, 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.

To avoid overloaded nesting, clear the individual interrupt enable flag whose interrupt is currently serviced, before setting IMF to "1". As for non-maskable interrupt, keep interrupt service shorten compared with length between interrupt requests; otherwise the status cannot be recovered as non-maskable interrupt would simply nested.

3.3.2 Saving/restoring general-purpose registers

During interrupt acceptance processing, the program counter (PC) and the program status word (PSW, includes IMF) are automatically saved on the stack, but the accumulator and others are not. These registers are saved by software if necessary. When multiple interrupt services are nested, it is also necessary to avoid using the same data memory area for saving registers. The following methods are used to save/restore the general-purpose registers.

3.3.2.1 Using PUSH and POP instructions

If only a specific register is saved or interrupts of the same source are nested, general-purpose registers can be saved/restored using the PUSH/POP instructions.

Example :Save/store register using PUSH and POP instructions

```
PINTxx:         PUSH       WA                 ; Save WA register
                (interrupt processing)
                POP        WA                 ; Restore WA register
                RETI                          ; RETURN
```
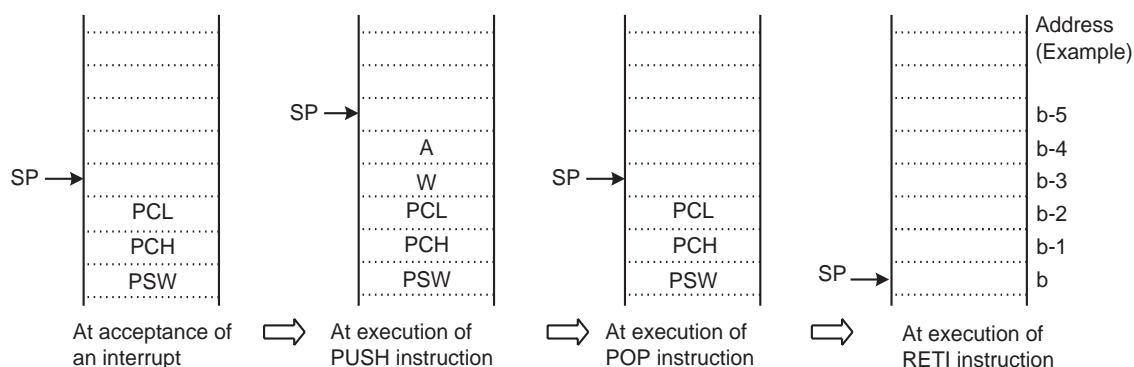

Figure 3-3 Save/store register using PUSH and POP instructions

3.3.2.2 Using data transfer instructions

To save only a specific register without nested interrupts, data transfer instructions are available.

Example :Save/store register using data transfer instructions

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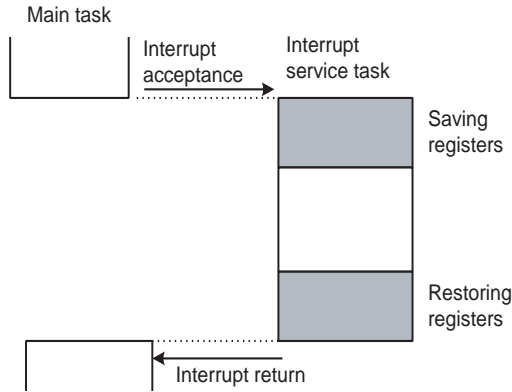

Saving/Restoring general-purpose registers using PUSH/POP data transfer instruction

Figure 3-4 Saving/Restoring General-purpose Registers under Interrupt Processing

3.3.3   Interrupt return

Interrupt return instructions [RETI]/[RETN] perform as follows.

[RETI]/[RETN] Interrupt Return
1. Program counter (PC) and program status word (PSW, includes IMF) are restored from the stack. 2. Stack pointer (SP) is incremented by 3.

As for address trap interrupt (INTATRAP), it is required to alter stacked data for program counter (PC) to restarting address, during interrupt service program.

Note: If [RETN] is executed with the above data unaltered, the program returns to the address trap area and INTATRAP occurs again.When interrupt acceptance processing has completed, stacked data for PCL and PCH are located on address (SP + 1) and (SP + 2) respectively.

Example 1 :Returning from address trap interrupt (INTATRAP) service program

```
PINTxx:        POP          WA                      ; Recover SP by 2
               LD           WA, Return Address      ;
               PUSH         WA                      ; Alter stacked data
               (interrupt processing)
               RETN                                 ; RETURN
```

Example 2 :Restarting without returning interrupt
(In this case, PSW (Includes IMF) before interrupt acceptance is discarded.)

```
PINTxx:         INC        SP                          ; Recover SP by 3

                INC        SP                          ;

                INC        SP                          ;

                (interrupt processing)

                LD         EIRL, data                  ; Set IMF to "1" or clear it to "0"

                JP         Restart Address             ; Jump into restarting address
```

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 1: It is recommended that stack pointer be return to rate before INTATRAP (Increment 3 times), if return interrupt instruction [RETN] is not utilized during interrupt service program under INTATRAP (such as Example 2).

Note 2: When the interrupt processing time is longer than the interrupt request generation time, the interrupt service task is performed but not the main task.

3.4   Software Interrupt (INTSW)

Executing the SWI instruction generates a software interrupt and immediately starts interrupt processing (INTSW is highest prioritized interrupt).

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

3.4.1   Address error detection

FFH is read if for some cause such as noise the CPU attempts to fetch an instruction from a non-existent memory address during single chip mode. Code FFH 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 FFH to unused areas of the program memory. Address trap reset is generated in case that an instruction is fetched from RAM, DBR or SFR areas.

3.4.2   Debugging

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

3.5   Undefined Instruction Interrupt (INTUNDEF)

Taking code which is not defined as authorized instruction for instruction causes INTUNDEF. INTUNDEF is generated when the CPU fetches such a code and tries to execute it. INTUNDEF is accepted even if non-maskable interrupt is in process. Contemporary process is broken and INTUNDEF interrupt process starts, soon after it is requested.

Note: The undefined instruction interrupt (INTUNDEF) forces CPU to jump into vector address, as software interrupt (SWI) does.

3.6   Address Trap Interrupt (INTATRAP)

Fetching instruction from unauthorized area for instructions (Address trapped area) causes reset output or address trap interrupt (INTATRAP). INTATRAP is accepted even if non-maskable interrupt is in process. Contemporary process is broken and INTATRAP interrupt process starts, soon after it is requested.

Note: The operating mode under address trapped, whether to be reset output or interrupt processing, is selected on watchdog timer control register (WDTCR).

3.7 External Interrupts

The TMP86FM26UG has 6 external interrupt inputs. These inputs are equipped with digital noise reject circuits (Pulse inputs of less than a certain time are eliminated as noise).

Edge selection is also possible with INT1 to INT4. The $\overline{INT0}$/P60 pin can be configured as either an external interrupt input pin or an input/output port, and is configured as an input port during reset.

Edge selection, noise reject control and $\overline{INT0}$/P60 pin function selection are performed by the external interrupt control register (EINTCR).

Source	Pin	Enable Conditions	Release Edge (level)	Digital Noise Reject
INT0	$\overline{INT0}$	IMF • EF4 • INT0EN=1	Falling edge	Pulses of less than 2/fc [s] are eliminated as noise. Pulses of 7/fc [s] or more are considered to be signals. In the SLOW or the SLEEP mode, pulses of less than 1/fs [s] are eliminated as noise. Pulses of 3.5/fs [s] or more are considered to be signals.
INT1	INT1	IMF • EF5 = 1	Falling edge or Rising edge	Pulses of less than 15/fc or 63/fc [s] are eliminated as noise. Pulses of 49/fc or 193/fc [s] or more are considered to be signals. In the SLOW or the SLEEP mode, pulses of less than 1/fs [s] are eliminated as noise. Pulses of 3.5/fs [s] or more are considered to be signals.
INT2	INT2	IMF • EF7 = 1	Falling edge or Rising edge	Pulses of less than 7/fc [s] are eliminated as noise. Pulses of 25/fc [s] or more are considered to be signals. In the SLOW or the SLEEP mode, pulses of less than 1/fs [s] are eliminated as noise. Pulses of 3.5/fs [s] or more are considered to be signals.
INT3	INT3	IMF • EF13 = 1	Falling edge or Rising edge	Pulses of less than 7/fc [s] are eliminated as noise. Pulses of 25/fc [s] or more are considered to be signals. In the SLOW or the SLEEP mode, pulses of less than 1/fs [s] are eliminated as noise. Pulses of 3.5/fs [s] or more are considered to be signals.
INT4	INT4	IMF • EF14 = 1	Falling edge, Rising edge, Falling and Rising edge or H level	Pulses of less than 7/fc [s] are eliminated as noise. Pulses of 25/fc [s] or more are considered to be signals. In the SLOW or the SLEEP mode, pulses of less than 1/fs [s] are eliminated as noise. Pulses of 3.5/fs [s] or more are considered to be signals.
INT5	$\overline{INT5}$	IMF • EF21 = 1	Falling edge	Pulses of less than 2/fc [s] are eliminated as noise. Pulses of 7/fc [s] or more are considered to be signals. In the SLOW or the SLEEP mode, pulses of less than 1/fs [s] are eliminated as noise. Pulses of 3.5/fs [s] or more are considered to be signals.

Note 1: In NORMAL1/2 or IDLE1/2 mode, if a signal with no noise is input on an external interrupt pin, it takes a maximum of "signal establishment time + 6/fs[s]" from the input signal's edge to set the interrupt latch.

Note 2: When INT0EN = "0", IL4 is not set even if a falling edge is detected on the $\overline{INT0}$ pin input.

Note 3: When a pin with more than one function is used as an output and a change occurs in data or input/output status, an interrupt request signal is generated in a pseudo manner. In this case, it is necessary to perform appropriate processing such as disabling the interrupt enable flag.

External Interrupt Control Register

EINTCR	7	6	5	4	3	2	1	0	
(0037H)	INT1NC	INT0EN	INT4ES		INT3ES	INT2ES	INT1ES		(Initial value: 0000 000*)
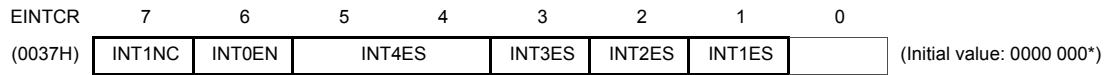
INT1NC	Noise reject time select	0: Pulses of less than 63/fc [s] are eliminated as noise 1: Pulses of less than 15/fc [s] are eliminated as noise	R/W
INT0EN	P60/$\overline{INT0}$ pin configuration	0: P60 input/output port 1: $\overline{INT0}$ pin (Port P60 should be set to an input mode)	R/W
INT4 ES	INT4 edge select	00: Rising edge 01: Falling edge 10: Rising edge and Falling edge 11: H level	R/W
INT3 ES	INT3 edge select	0: Rising edge 1: Falling edge	R/W
INT2 ES	INT2 edge select	0: Rising edge 1: Falling edge	R/W
INT1 ES	INT1 edge select	0: Rising edge 1: Falling edge	R/W

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

Note 2: When the system clock frequency is switched between high and low or when the external interrupt control register (EINTCR) is overwritten, the noise canceller may not operate normally. It is recommended that external interrupts are disabled using the interrupt enable register (EIR).

Note 3: The maximum time from modifying INT1NC until a noise reject time is changed is 2^6/fc.

Note 4: In case $\overline{RESET}$ pin is released while the state of INT4 pin keeps "H" level, the external interrupt 4 request is not generated even if the INT4 edge select is specified as "H" level. The rising edge is needed after $\overline{RESET}$ pin is released.

4. Special Function Register (SFR)

The TMP86FM26UG adopts the memory mapped I/O system, and all peripheral control and data transfers are performed through the special function register (SFR) or the data buffer register (DBR). The SFR is mapped on address 0000H to 003FH, DBR is mapped on address 0F80H to 0FFFH.

This chapter shows the arrangement of the special function register (SFR) and data buffer register (DBR) for TMP86FM26UG.

4.1 SFR

Address	Read	Write
0000H	Reserved	
0001H	P1DR	
0002H	P2DR	
0003H	P3DR	
0004H	P3OUTCR	
0005H	P5DR	
0006H	P6DR	
0007H	P7DR	
0008H	P1CR	
0009H	P2PRD	-
000AH	P3PRD	-
000BH	P5CR	
000CH	P6OUTCR	
000DH	P7CR	
000EH	UART1SR	UART1CR1
000FH	-	UART1CR2
0010H	TREG1AL	
0011H	TREG1AM	
0012H	TREG1AH	
0013H	TREG1B	
0014H	TC1CR1	
0015H	TC1CR2	
0016H	TC1SR	-
0017H	Reserved	
0018H	TC3CR	
0019H	TC4CR	
001AH	TC5CR	
001BH	TC6CR	
001CH	TTREG3	
001DH	TTREG4	
001EH	TTREG5	
001FH	TTREG6	
0020H	PWREG3	
0021H	PWREG4	
0022H	PWREG5	
0023H	PWREG6	
0024H	P6PRD	-
0025H	UART0SR	UART0CR1

Address	Read	Write
0026H	-	UART0CR2
0027H	P2OUTCR	
0028H	LCDCR	
0029H	P1LCR	
002AH	P5LCR	
002BH	P7LCR	
002CH	EIRE	
002DH	Reserved	
002EH	ILE	
002FH	Reserved	
0030H	Reserved	
0031H	Reserved	
0032H	Reserved	
0033H	Reserved	
0034H	-	WDTCR1
0035H	-	WDTCR2
0036H	TBTCR	
0037H	EINTCR	
0038H	SYSCR1	
0039H	SYSCR2	
003AH	EIRL	
003BH	EIRH	
003CH	ILL	
003DH	ILH	
003EH	Reserved	
003FH	PSW	

Note 1: Do not access reserved areas by the program.

Note 2: − ; Cannot be accessed.

Note 3: Write-only registers and interrupt latches cannot use the read-modify-write instructions (Bit manipulation instructions such as SET, CLR, etc. and logical operation instructions such as AND, OR, etc.).

4.2 DBR

Address	Read	Write
0F80H	SEG1/0	
0F81H	SEG3/2	
0F82H	SEG5/4	
0F83H	SEG7/6	
0F84H	SEG9/8	
0F85H	SEG11/10	
0F86H	SEG13/12	
0F87H	SEG15/14	
0F88H	SEG17/16	
0F89H	SEG19/18	
0F8AH	SEG21/20	
0F8BH	SEG23/22	
0F8CH	SEG25/24	
0F8DH	SEG27/26	
0F8EH	SEG29/28	
0F8FH	SEG31/30	
0F90H	SIOBR0	
0F91H	SIOBR1	
0F92H	SIOBR2	
0F93H	SIOBR3	
0F94H	SIOBR4	
0F95H	SIOBR5	
0F96H	SIOBR6	
0F97H	SIOBR7	
0F98H	-	SIOCR1
0F99H	SIOSR	SIOCR2
0F9AH	-	STOPCR
0F9BH	RD0BUF	TD0BUF
0F9CH	RD1BUF	TD1BUF
0F9DH	Reserved	
0F9EH	MERGECR	
0F9FH	Reserved	

Address	Read	Write
0FA0H	RTCCR1	
0FA1H	RTCCR2	
0FA2H	RTCSR	-
0FA3H	RTREG1L	
0FA4H	RTREG1M	
0FA5H	RTREG1H	
0FA6H	DIVRG1L	-
0FA7H	DIVRG1M	-
0FA8H	DIVRG1H	-
0FA9H	DIVRG2L	-
0FAAH	DIVRG2M	-
0FABH	DIVRG2H	-
0FACH	SECR	
0FADH	MINR	
0FAEH	HOURR	
0FAFH	WEEKR	
0FB0H	DAYR	
0FB1H	MONTHR	
0FB2H	YEARR	
0FB3H	LEAPR	
0FB4H	Reserved	
0FB5H	IRSCR	
0FB6H	CLKSCR	
0FB7H	CLKSMN	
0FB8H	CLKSMX	
0FB9H	Reserved	
0FBAH	Reserved	
0FBBH	Reserved	
0FBCH	Reserved	
0FBDH	Reserved	
0FBEH	Reserved	
0FBFH	Reserved	

Address	Read	Write
0FC0H	Reserved	
: :	: :	
0FDFH	Reserved	

Address	Read	Write
0FE0H	EEPCR	
0FE1H	EEPSR	-
0FE2H	EEPEVA	
0FE3H	Reserved	
0FE4H	Reserved	
0FE5H	Reserved	
0FE6H	Reserved	
0FE7H	Reserved	
0FE8H	Reserved	
0FE9H	Reserved	
0FEAH	Reserved	
0FEBH	Reserved	
0FECH	Reserved	
0FEDH	Reserved	
0FEEH	Reserved	
0FEFH	Reserved	
0FF0H	Reserved	
0FF1H	Reserved	
0FF2H	Reserved	
0FF3H	Reserved	
0FF4H	Reserved	
0FF5H	Reserved	
0FF6H	Reserved	
0FF7H	Reserved	
0FF8H	Reserved	
0FF9H	Reserved	
0FFAH	Reserved	
0FFBH	Reserved	
0FFCH	Reserved	
0FFDH	Reserved	
0FFEH	Reserved	
0FFFH	Reserved	

Note 1: Do not access reserved areas by the program.

Note 2: − ; Cannot be accessed.

Note 3: Write-only registers and interrupt latches cannot use the read-modify-write instructions (Bit manipulation instructions such as SET, CLR, etc. and logical operation instructions such as AND, OR, etc.).

5. I/O Ports

The TMP86FM26UG has 8 parallel input/output ports (41 pins) as follows.

	Primary Function	Secondary Functions
Port P1	8-bit I/O port	LCD segment output.
Port P2	5-bit I/O port	Low-frequency resonator connections, external interrupt input, STOP mode release signal input, real time clock output and real time clock input.
Port P3	4-bit I/O port	Timer/counter input/output, UART input and divider output.
Port P5	8-bit I/O port	LCD segment output.
Port P6	8-bit I/O port	External interrupt input, Key on Wake up input, Serial interface input/output, UART input/output and serial PROM mode control input.
Port P7	8-bit I/O port	LCD segment output.

Each output port contains a latch, which holds the output data. All input ports do not have latches, so the external input data should be externally held until the input data is read from outside or reading should be performed several timer before processing. Figure 5-1 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 cannot be recognized from outside, so that transient input such as chattering must be processed by the program.

Output data changes in the S2 state of the write cycle during execution of the instruction which writes to an I/O port.
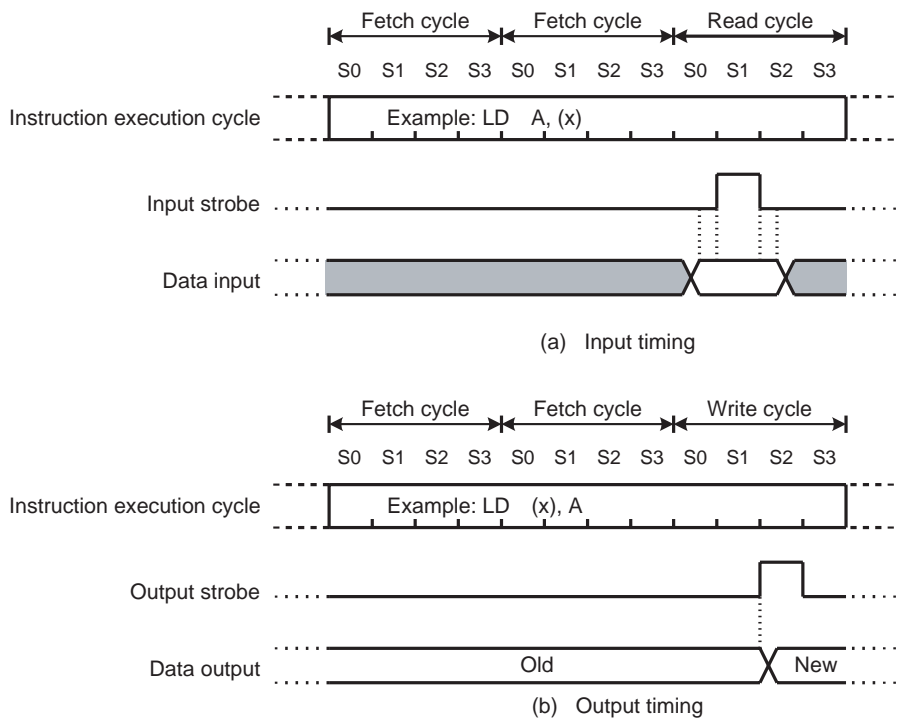
Note: The positions of the read and write cycles may vary, depending on the instruction.

Figure 5-1 Input/Output Timing (Example)

5.1   Port P1 (P17 to P10)

Port P1 is an 8-bit input/output port which can be configured as an input or an output in 1-bit unit. Port P1 is also used as a segment output of LCD. Input/output mode is specified by the P1 control register (P1CR).

When used as an input port, the corresponding bit of P1CR and P1LCR should be cleared to "0".

When used as an output port, the corresponding bit of P1CR should be set to "1", and the respective P1LCR bit should be cleared to "0".

When used as a segment pins of LCD, the respective bit of P1LCR should be set to "1".

During reset, the output latch (P1DR), P1CR and P1LCR are initialized to "0".

When the bit of P1CR and P1LCR is "0", the corresponding bit data by read instruction is a terminal input data.

When the bit of P1CR is "0" and that of P1LCR is "1", the corresponding bit data by read instruction is always "0".

When the bit of P1CR is "1", the corresponding bit data by read instruction is the value of P1DR.

Table 5-1   Register Programming for Multi-function Ports

Function	Programmed Value		
	P1DR	P1CR	P1LCR
Port input	*	"0"	"0"
Port "0" output	"0"	"1"	"0"
Port "1" output	"1"	"1"	"0"
LCD segment output	*	*	"1"

Note: Asterisk (*) indicates "1" or "0" either of which can be selected.

Table 5-2   Values Read from P1DR and Register Programming

Conditions		Values Read from P1DR
P1CR	P1LCR	
"0"	"0"	Terminal input data
"0"	"1"	"0"
"1"	"0"	Output latch contents
	"1"	
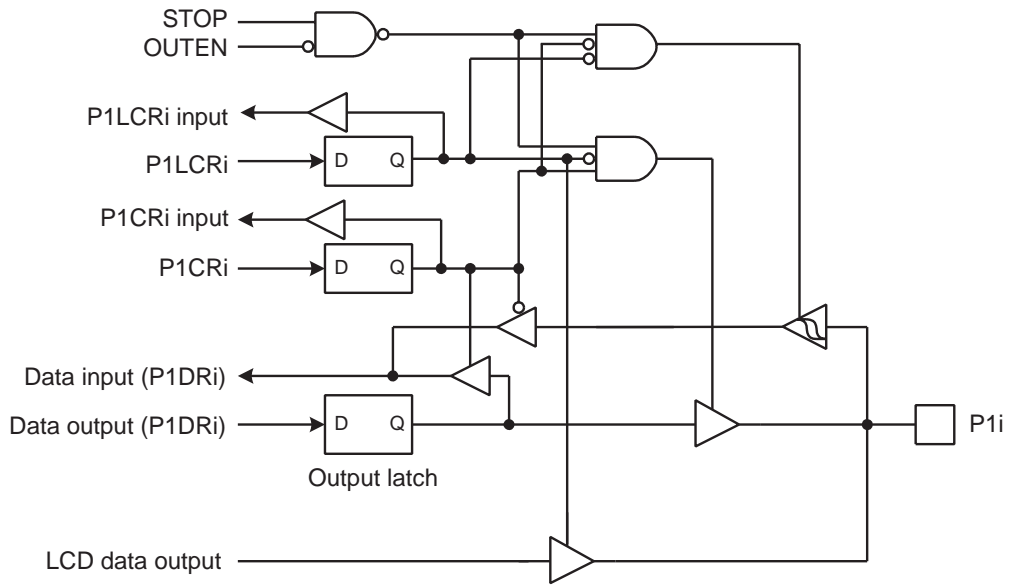
Note: i = 7 to 0

Figure 5-2 Port 1

P1DR (0001H) R/W	7	6	5	4	3	2	1	0	
	P17 SEG24	P16 SEG25	P15 SEG26	P14 SEG27	P13 SEG28	P12 SEG29	P11 SEG30	P10 SEG31	(Initial value: 0000 0000)

P1LCR (0029H)									(Initial value: 0000 0000)

P1LCR	Port P1/segment output control (Set for each bit individually)	0: P1 input/output port 1: LCD segment output	R/W

P1CR (0008H)									(Initial value: 0000 0000)

P1CR	P1 port input/output control (Set for each bit individually)	0: Input mode 1: Output mode	R/W

Note: The port placed in input mode reads the pin input state. Therefore, when the input and output modes are used together, the output latch contents for the port in input mode might be changed by executing a bit manipulation instruction.

5.2 Port P2 (P24 to P20)

Port P2 is a 5-bit input/output port.

It is also used as an external interrupt, a STOP mode release signal input, and low-frequency crystal oscillator connection pins. When used as an input port or a secondary function pins, respective output latch (P2DR) should be set to "1".

During reset, the P2DR is initialized to "1".

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

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

It can be selected whether output circuit of port P2 is C-MOS output or a sink open drain individually, by setting P2OUTCR. When a corresponding bit of P2OUTCR is "0", the output circuit is selected to a sink open drain and when a corresponding bit of P2OUTCR is "1", the output circuit is selected to a C-MOS output.

P2 port output latch (P2DR) and P2 port terminal input (P2PRD) are located on their respective address.

When read the output latch data, the P2DR should be read and when read the terminal input data, the P2PRD register should be read. If a read instruction is executed for P2DR and P2PRD, read data of bits 7 to 5 are unstable and then read data of bits 7 to 5 and 2 to 1 are unstable in case of P2OUTCR.

a) P20

b) P22, P21

c) P24, P23
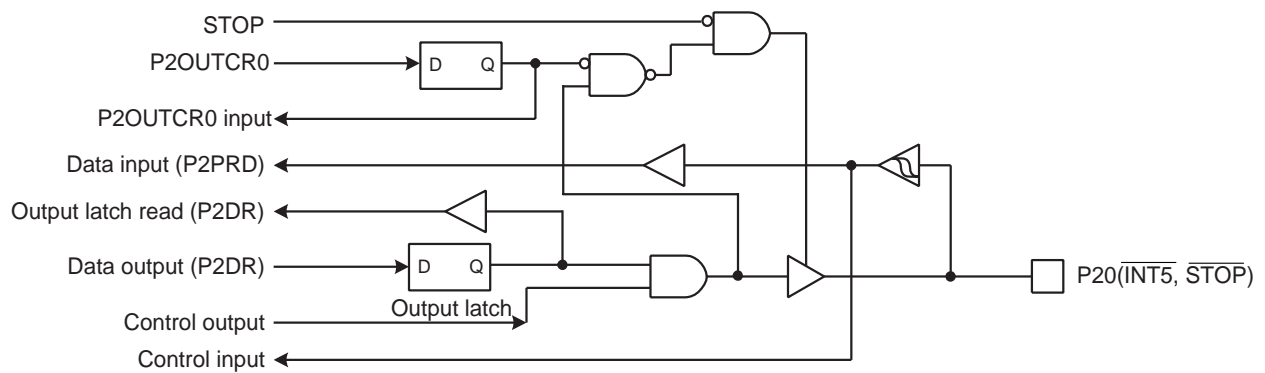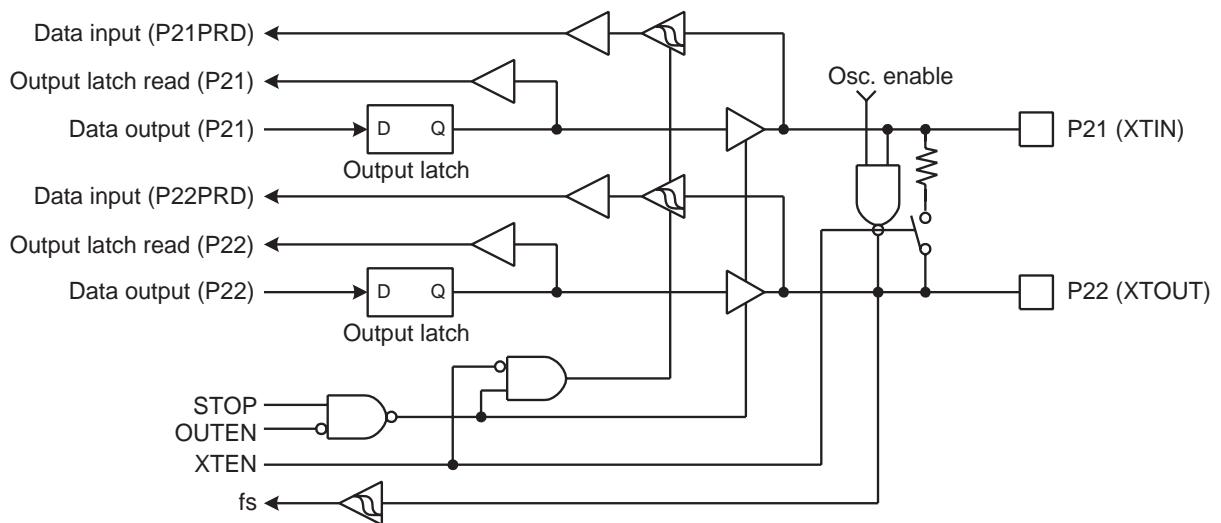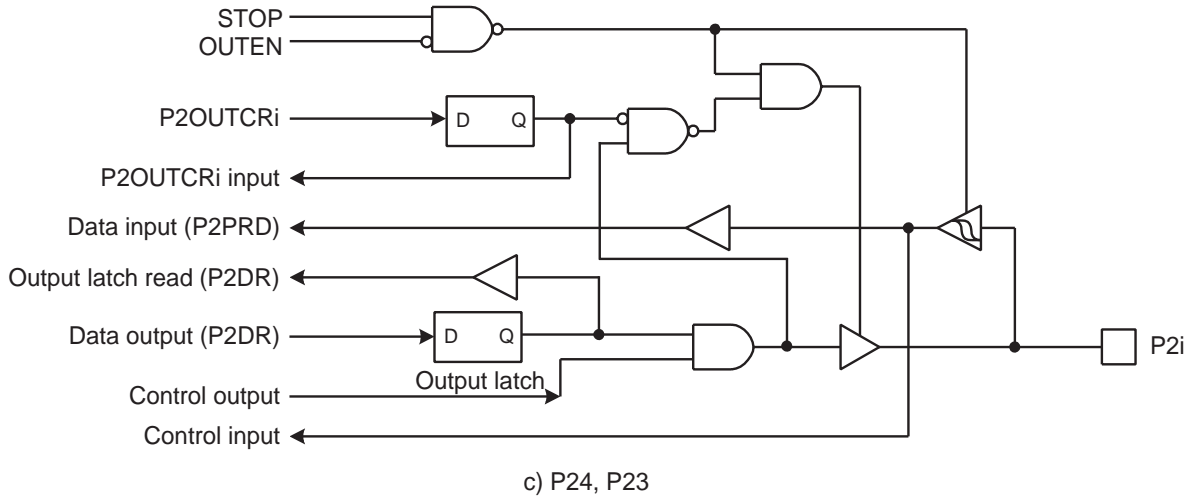
Note: i = 4 and 3

Figure 5-3 Port 2

	7	6	5	4	3	2	1	0	
P2DR (0002H) R/W				P24 RTCOUT	P23 RTCIN	P22 XTOUT	P21 XTIN	P20 $\overline{INT5}$ $\overline{STOP}$	(Initial value: ***1 1111)

	7	6	5	4	3	2	1	0
P2PRD (0009H) Read only				P24	P23	P22	P21	P20

	7	6	5	4	3	2	1	0	
P2OUTCR (0027H) R/W				P24	P23			P20	(Initial value: ***0 0**0)

P2OUTCR	Port P2 output circuit control (Set for each bit individually)	0: Sink open-drain output 1: C-MOS output	R/W

Note: Port P20 is used as $\overline{STOP}$ pin. Therefore, when stop mode is started, OUTEN does not affect to P20, and P20 becomes high-Z mode.

5.3 Port P3 (P33 to P30)

Port P3 is a 8-bit input/output port.

It is also used as a timer/counter input/output, UART input/output and divider output.

When used as a timer/counter input/output, UART input/output and divider output, respective output latch (P3DR) should be set to "1".

It can be selected whether output circuit of port P3 is C-MOS output or a sink open drain individually, by setting P3OUTCR. When a corresponding bit of P3OUTCR is "0", the output circuit is selected to a sink open drain and when a corresponding bit of P3OUTCR is "1", the output circuit is selected to a C-MOS output. When used as an input port, UART input and timer/counter input, respective output control (P3OUTCR) should be set to "0" after P3DR is set to "1". During reset, the P3DR is initialized to "1", and the P3OUTCR is initialized to "0".

P3 port output latch (P3DR) and P3 port terminal input (P3PRD) are located on their respective address.

When read the output latch data, the P3DR should be read and when read the terminal input data, the P3PRD register should be read. If a read instruction is executed for port P3, read data of bits 7 to 4 are unstable.

Table 5-3   Register Programming for Multi-function Ports (P33 to P30)

Function	Programmed Value	
	P3DR	P3OUTCR
Port input, UART input or timer counter input	"1"	"0"
Port "0" output	"0"	Programming for each applications
Port "1" output, UART output or timer counter output	"1"	

a) P33, P31, P30

Note: i = 3, 1, 0

b) P32
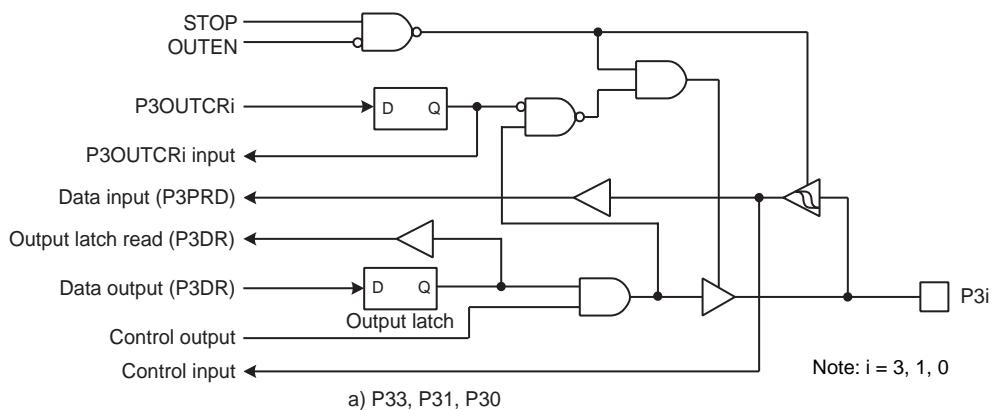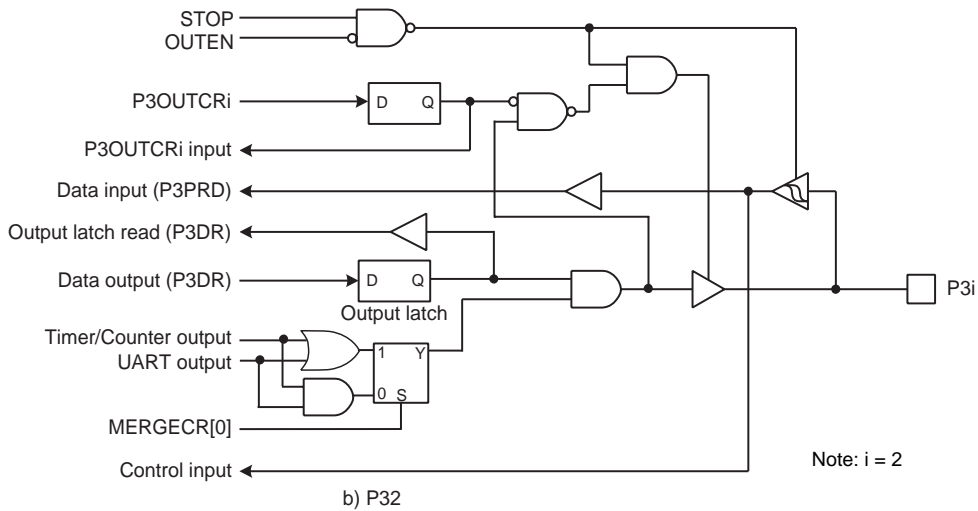
Figure 5-4 Port 3

	7	6	5	4	3	2	1	0	
P3DR (0003H) R/W					P33 PWM6 $\overline{PDO6}$ $\overline{PPG6}$ TC6 RXD1	P32 $\overline{PWM4}$ $\overline{PDO4}$ $\overline{PPG4}$ TC4 TXD1	P31 $\overline{PWM3}$ $\overline{PDO3}$ TC3	P30 $\overline{DVO}$	(Initial value: **** 1111)

P3OUTCR (0004H)									(Initial value: **** 0000)

P3OUTCR	Port P3 output circuit control (Set for each bit individually)	0: Sink open-drain output 1: C-MOS output	R/W

P3PRD (000AH) Read only					P33	P32	P31	P30

Pin P32 can be used to output the logical AND or OR of UART output and timer/counter output when both of these functions are enabled. Whether to output the AND or OR of these outputs is selected by MERGECR<UTSEL>.

To use only UART ouput or timer/counter output, set MERGECR<UTSEL> to "0"and then disable the output of whichever function not being used.

UART/Timer Output Select Register

	7	6	5	4	3	2	1	0	
MERGECR (0F9EH)								UTSEL	(Initial value: **** ***0)

UTSEL	UART/Timer counter output	0: Logical AND 1: Logical OR	R/W

Note: When MERGECR is read, bits 7 to 1 are read as undefined data.
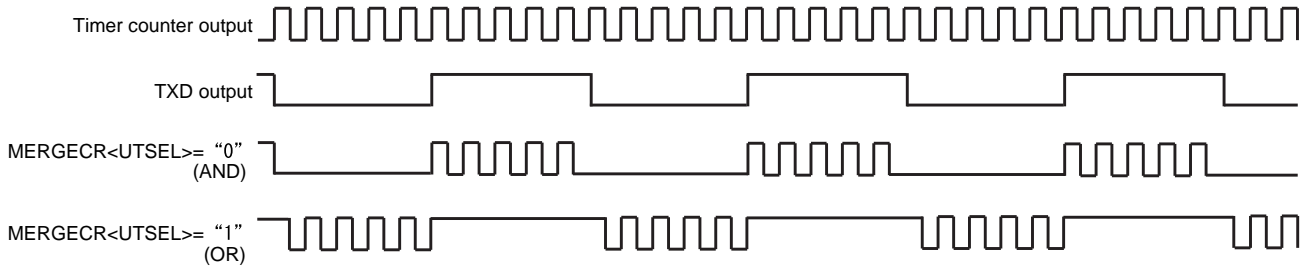
Figure 5-5 Pin P32 Output Waveform

5.4   Port P5 (P57 to P50)

Port P5 is an 8-bit input/output port which can be configured as an input or an output in 1-bit unit. Port P5 is also used as a segment output of LCD. Input/output mode is specified by the P5 control register (P5CR).

When used as an input port, the corresponding bit of P5CR and P5LCR should be cleared to "0".

When used as an output port, the corresponding bit of P5CR should be set to "1", and the respective P5LCR bit should be cleared to "0".

When used as a segment pins of LCD, the respective bit of P5LCR should be set to "1".

During reset, the output latch (P5DR), P5CR and P5LCR are initialized to "0".

When the bit of P5CR and P5LCR is "0", the corresponding bit data by read instruction is a terminal input data.

When the bit of P5CR is "0" and that of P5LCR is "1", the corresponding bit data by read instruction is always "0".

When the bit of P5CR is "1", the corresponding bit data by read instruction is the value of P5DR.

Table 5-4   Register Programming for Multi-function Ports

Function	Programmed Value		
	P5DR	P5CR	P5LCR
Port input	*	"0"	"0"
Port "0" output	"0"	"1"	"0"
Port "1" outpu	"1"	"1"	"0"
LCD segment output	*	*	"1"

Note: Asterisk (*) indicates "1" or "0" either of which can be selected.

Table 5-5   Values Read from P1DR and Register Programming

Conditions		Values Read from P5DR
P5CR	P5LCR	
"0"	"0"	Terminal input data
"0"	"1"	"0"
"1"	"0"	Output latch contents
	"1"	
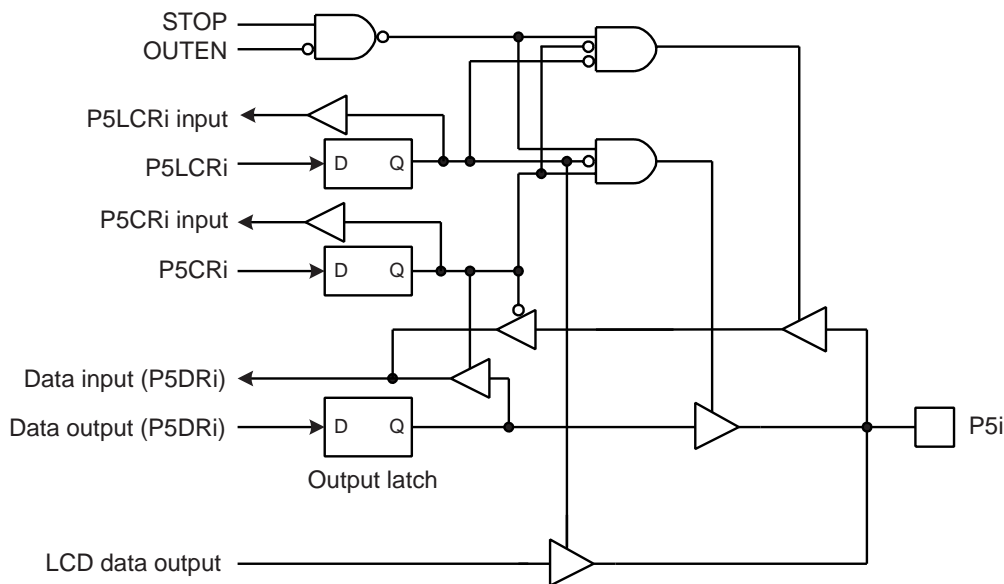
Note: i = 7 to 0

Figure 5-6 Port 5

P5DR	7	6	5	4	3	2	1	0	
(0005H) R/W	P57 SEG16	P56 SEG17	P55 SEG18	P54 SEG19	P53 SEG20	P52 SEG21	P51 SEG22	P50 SEG23	(Initial value: 0000 0000)

P5LCR (002AH)									(Initial value: 0000 0000)

P5LCR	Port P5/segment output control (Set for each bit individually)	0: P5 input/output port 1: LCD segment output	R/W

P5CR (000BH)								

P5CR	P5 port input/output control (Set for each bit individually)	0: Input mode 1: Output mode	R/W

Note: The port placed in input mode reads the pin input state. Therefore, when the input and output modes are used together, the output latch contents for the port in input mode might be changed by executing a bit manipulation instruction.

5.5 Port P6 (P67 to P60)

Port P6 is a 8-bit input/output port.

It is also used as a timer counter input, UART input/output, serial interface input/output, external interrupt input, Key on wake up input and seral PROM mode control input.

When used as a secondary function pins respective output latch (P6DR) should be set to "1".

It can be selected whether output circuit of port P6 is C-MOS output or a sink open drain individually, by setting P6OUTCR. When a corresponding bit of P6OUTCR is "0", the output circuit is selected to a sink open drain and when a corresponding bit of P6OUTCR is "1", the output circuit is selected to a C-MOS output. When used as an input port, UART input, serial interface input, external interrupt input, Key on Wake up input and timer/counter input, respective output control (P6OUTCR) should be set to "0" after P6DR is set to "1". During reset, the P6DR is initialized to "1", and the P6OUTCR is initialized to "0".

P6 port output latch (P6DR) and P6 port terminal input (P6PRD) are located on their respective address.

When read the output latch data, the P6DR should be read and when read the terminal input data, the P6PRD register should be read.

Table 5-6 Register Programming for Multi-function Ports (P67 to P60)

Function	Programmed Value	
	P6DR	P6OUTCR
Port input, UART input, serial interface input, external interrrupt input, Key on Wake up input or timer counter input	"1"	"0"
Port "0" output	"0"	Programming for each applications
Port "1" output, UART output or timer counter output	"1"	
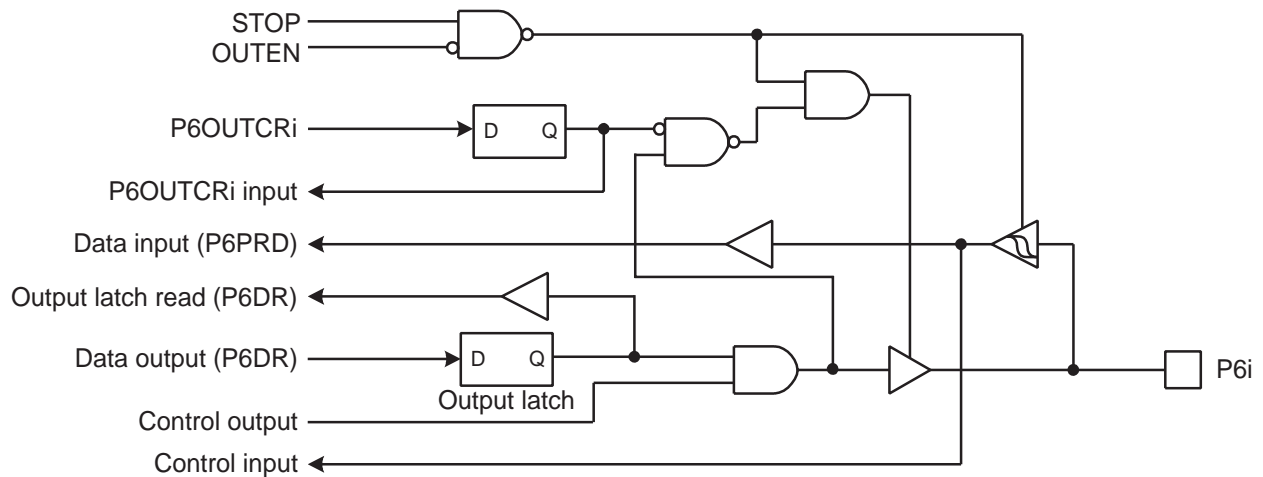
Note: i = 7 to 0

Figure 5-7 Port 6

	7	6	5	4	3	2	1	0	
P6DR (0006H) R/W	P67 STOP5	P66 STOP4 SO TXD0	P65 STOP3 SI RXD0 BOOT	P64 STOP2 $\overline{SCK}$ INT4	P63 INT3	P62 INT2 ECNT	P61 INT1 ECIN	P60 $\overline{INT0}$	(Initial value: 1111 1111)

P6OUTCR (000CH)									(Initial value: 0000 0000)

P6OUTCR	Port P6 output circuit control (Set for each bit individually)	0: Sink open-drain output 1: C-MOS output	R/W

P6PRD (0024H) Read only	P67	P66	P65	P64	P63	P62	P61	P60

5.6 Port P7(P77 to P70)

Port P7 is an 8-bit input/output port which can be configured as an input or an output in 1-bit unit. Port P7 is also used as a segment output of LCD. Input/output mode is specified by the P7 control register (P7CR).

When used as an input port, the corresponding bit of P7CR and P7LCR should be cleared to "0".

When used as an output port, the corresponding bit of P7CR should be set to "1", and the respective P7LCR bit should be cleared to "0".

When used as a segment pins of LCD, the respective bit of P7LCR should be set to "1".

During reset, the output latch (P7DR), P7CR and P7LCR are initialized to "0".

When the bit of P7CR and P7LCR is "0", the corresponding P7bit data by read instruction is a terminal input data.

When the bit of P7CR is "0" and that of P7LCR is "1", the corresponding bit data by read instruction is always "0".

When the bit of P7CR is "1", the corresponding bit data by read instruction is the value of P7DR.

Table 5-7 Register Programming for Multi-function Ports

Function	Programmed Value		
	P7DR	P7CR	P7LCR
Port input	*	"0"	"0"
Port "0" output	"0"	"1"	"0"
Port "1" output	"1"	"1"	"0"
LCD segment output	*	*	"1"

Note: Asterisk (*) indicates "1" or "0" either of which can be selected.

Table 5-8 Values Read from P7DR and Register Programming

Conditions		Values Read from P7DR
P7CR	P7LCR	
"0"	"0"	Terminal input data
"0"	"1"	"0"
"1"	"0"	Output latch contents
	"1"	
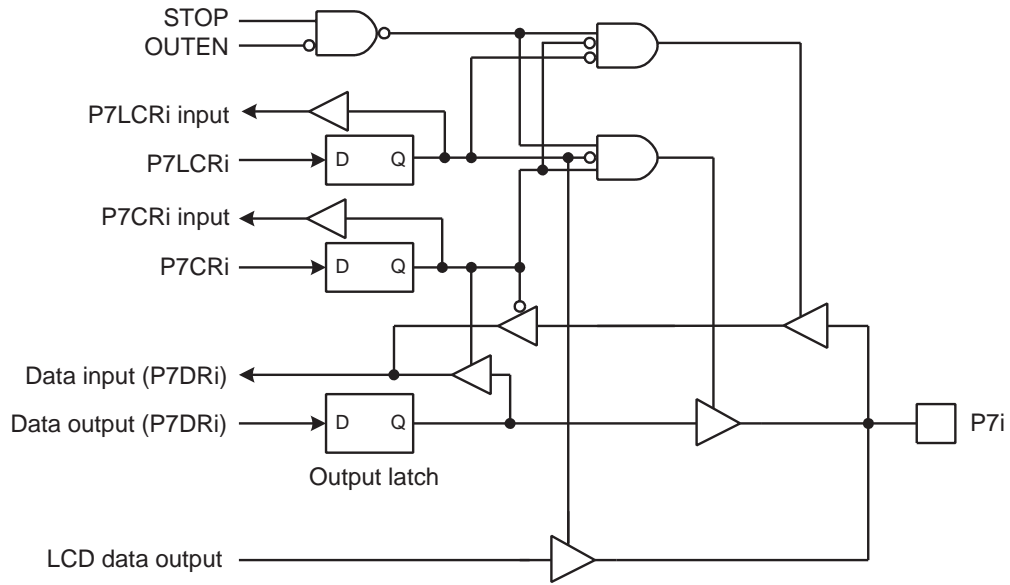
Note: i = 7 to 0

Figure 5-8 Port 7

P7DR	7	6	5	4	3	2	1	0	
(0007H) R/W	P77 SEG8	P76 SEG9	P75 SEG10	P74 SEG11	P73 SEG12	P72 SEG13	P71 SEG14	P70 SEG15	(Initial value: 0000 0000)

P7LCR (002BH)									(Initial value: 0000 0000)

P7LCR	Port P7/segment output control (Set for each bit individually)	0: P7 input/output port 1: LCD segment output	R/W

P7CR (000DH)									(Initial value: 0000 0000)

P7CR	P7 port input/output control (Set for each bit individually)	0: Input mode 1: Output mode	R/W

Note: The port placed in input mode reads the pin input state. Therefore, when the input and output modes are used together, the output latch contents for the port in input mode might be changed by executing a bit manipulation instruction.

6. Watchdog Timer (WDT)

The watchdog timer is a fail-safe system to detect rapidly the CPU malfunctions such as endless loops due to spurious noises or the deadlock conditions, and return the CPU to a system recovery routine.

The watchdog timer signal for detecting malfunctions can be programmed only once as "reset request" or "interrupt request". Upon the reset release, this signal is initialized to "reset request".

When the watchdog timer is not used to detect malfunctions, it can be used as the timer to provide a periodic interrupt.

Note: Care must be taken in system design since the watchdog timer functions are not be operated completely due to effect of disturbing noise.

6.1 Watchdog Timer Configuration
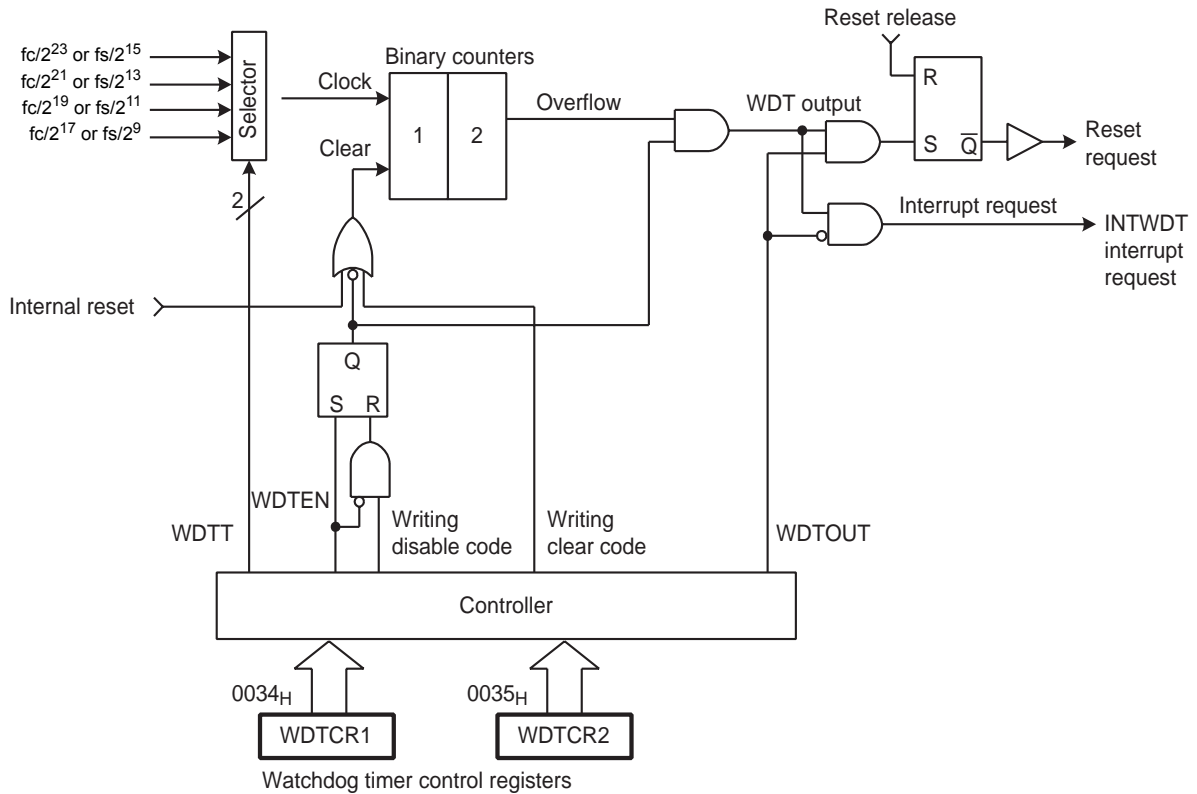
Figure 6-1 Watchdog Timer Configuration

6.2 Watchdog Timer Control

The watchdog timer is controlled by the watchdog timer control registers (WDTCR1 and WDTCR2). The watchdog timer is automatically enabled after the reset release.

6.2.1 Malfunction Detection Methods Using the Watchdog Timer

The CPU malfunction is detected, as shown below.

1. Set the detection time, select the output, and clear the binary counter.
2. Clear the binary counter repeatedly within the specified detection time.

If the CPU malfunctions such as endless loops or the deadlock conditions occur for some reason, the watchdog timer output is activated by the binary-counter overflow unless the binary counters are cleared. When WDTCR1<WDTOUT> is set to "1" at this time, the reset request is generated and then internal hardware is initialized. When WDTCR1<WDTOUT> is set to "0", a watchdog timer interrupt (INTWDT) is generated.

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

Note: The watchdog timer consists of an internal divider and a two-stage binary counter. When the clear code 4EH is written, only the binary counter is cleared, but not the internal divider. The minimum binary-counter overflow time, that depends on the timing at which the clear code (4EH) is written to the WDTCR2 register, may be 3/4 of the time set in WDTCR1<WDTT>. Therefore, write the clear code using a cycle shorter than 3/4 of the time set to WDTCR1<WDTT>.

Example : Setting the watchdog timer detection time to $2^{21}/fc$ [s], and resetting the CPU malfunction detection

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

              LD        (WDTCR1), 00001101B    : WDTT ← 10, WDTOUT ← 1

              LD        (WDTCR2), 4EH          : Clears the binary counters (always clears immediately before and
                                                 after changing WDTT).
Within 3/4 of WDT
              :
detection time
              :

              LD        (WDTCR2), 4EH          : Clears the binary counters.


Within 3/4 of WDT
              :
detection time
              :

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

Watchdog Timer Control Register 1

WDTCR1 (0034H)	7	6	5 (ATAS)	4 (ATOUT)	3 WDTEN	2 WDTT	1	0 WDTOUT	(Initial value: **11 1001)
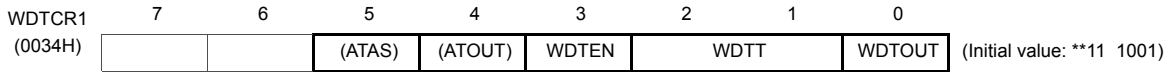
WDTEN	Watchdog timer enable/disable	0: Disable (Writing the disable code to WDTCR2 is required.) 1: Enable			Write only
WDTT	Watchdog timer detection time [s]		NORMAL1/2 mode		SLOW1/2 mode	Write only
			DV7CK = 0	DV7CK = 1		
		00	$2^{25}/fc$	$2^{17}/fs$	$2^{17}/fs$	
		01	$2^{23}/fc$	$2^{15}/fs$	$2^{15}fs$	
		10	$2^{21}fc$	$2^{13}/fs$	$2^{13}fs$	
		11	$2^{19}/fc$	$2^{11}/fs$	$2^{11}/fs$	
WDTOUT	Watchdog timer output select	0: Interrupt request 1: Reset request			Write only

Note 1: After clearing WDTOUT to "0", the program cannot set it to "1".

Note 2: fc: High-frequency clock [Hz], fs: Low-frequency clock [Hz], *: Don't care

Note 3: WDTCR1 is a write-only register and must not be used with any of read-modify-write instructions. If WDTCR1 is read, a don't care is read.

Note 4: To activate the STOP mode, disable the watchdog timer or clear the counter immediately before entering the STOP mode. After clearing the counter, clear the counter again immediately after the STOP mode is inactivated.

Note 5: To clear WDTEN, set the register in accordance with the procedures shown in "6.2.3 Watchdog Timer Disable".

Watchdog Timer Control Register 2

WDTCR2 (0035H)	7	6	5	4	3	2	1	0	(Initial value: **** ****)
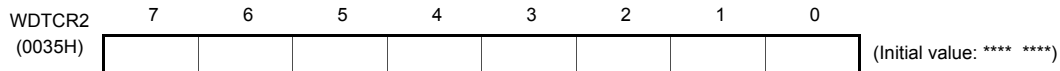
WDTCR2	Write Watchdog timer control code	4EH: Clear the watchdog timer binary counter (Clear code) B1H: Disable the watchdog timer (Disable code) D2H: Enable assigning address trap area Others: Invalid	Write only

Note 1: The disable code is valid only when WDTCR1<WDTEN> = 0.

Note 2: *: Don't care

Note 3: The binary counter of the watchdog timer must not be cleared by the interrupt task.

Note 4: Write the clear code 4EH using a cycle shorter than 3/4 of the time set in WDTCR1<WDTT>.

6.2.2 Watchdog Timer Enable

Setting WDTCR1<WDTEN> to "1" enables the watchdog timer. Since WDTCR1<WDTEN> is initialized to "1" during reset, the watchdog timer is enabled automatically after the reset release.

6.2.3 Watchdog Timer Disable

To disable the watchdog timer, set the register in accordance with the following procedures. Setting the register in other procedures causes a malfunction of the microcontroller.

1. Set the interrupt master flag (IMF) to "0".
2. Set WDTCR2 to the clear code (4EH).
3. Set WDTCR1<WDTEN> to "0".
4. Set WDTCR2 to the disable code (B1H).

Note: While the watchdog timer is disabled, the binary counters of the watchdog timer are cleared.

Example :Disabling the watchdog timer

DI		: IMF ← 0
LD	(WDTCR2), 04EH	: Clears the binary counter
LDW	(WDTCR1), 0B101H	: WDTEN ← 0, WDTCR2 ← Disable code

Table 6-1   Watchdog Timer Detection Time (Example: fc = 16.0 MHz, fs = 32.768 kHz)

WDTT	Watchdog Timer Detection Time[s]		
	NORMAL1/2 mode		SLOW mode
	DV7CK = 0	DV7CK = 1	
00	2.097	4	4
01	524.288 m	1	1
10	131.072 m	250 m	250 m
11	32.768 m	62.5 m	62.5 m

6.2.4 Watchdog Timer Interrupt (INTWDT)

When WDTCR1<WDTOUT> is cleared to "0", a watchdog timer interrupt request (INTWDT) is generated by the binary-counter overflow.

A watchdog timer interrupt is the non-maskable interrupt which can be accepted regardless of the interrupt master flag (IMF).

When a watchdog timer interrupt is generated while the other interrupt including a watchdog timer interrupt is already accepted, the new watchdog timer interrupt is processed immediately and the previous interrupt is held pending. Therefore, if watchdog timer interrupts are generated continuously without execution of the RETN instruction, too many levels of nesting may cause a malfunction of the microcontroller.

To generate a watchdog timer interrupt, set the stack pointer before setting WDTCR1<WDTOUT>.

Example :Setting watchdog timer interrupt

LD	SP, 043FH	: Sets the stack pointer
LD	(WDTCR1), 00001000B	: WDTOUT ← 0

6.2.5   Watchdog Timer Reset

When a binary-counter overflow occurs while WDTCR1<WDTOUT> is set to "1", a watchdog timer reset request is generated. When a watchdog timer reset request is generated, the internal hardware is reset. The reset time is maximum $24/fc$ [s] (1.5 μs @ fc = 16.0 MHz). After reset, the CPU enters a wait state until the power supply of the flash memory control circuit is stable. The wait time is $2^{10}/fc$ [s] (62.5 μs @ fc = 16.0 MHz). The CPU halts and remains in a wait state, and restarts operation after the wait time.

Note: When a watchdog timer reset is generated in the SLOW1 mode, the reset time is maximum $24/fc$ (high-frequency clock) since the high-frequency clock oscillator is restarted. However, when crystals have inaccuracies upon start of the high-frequency clock oscillator, the reset time should be considered as an approximate value because it has slight errors.
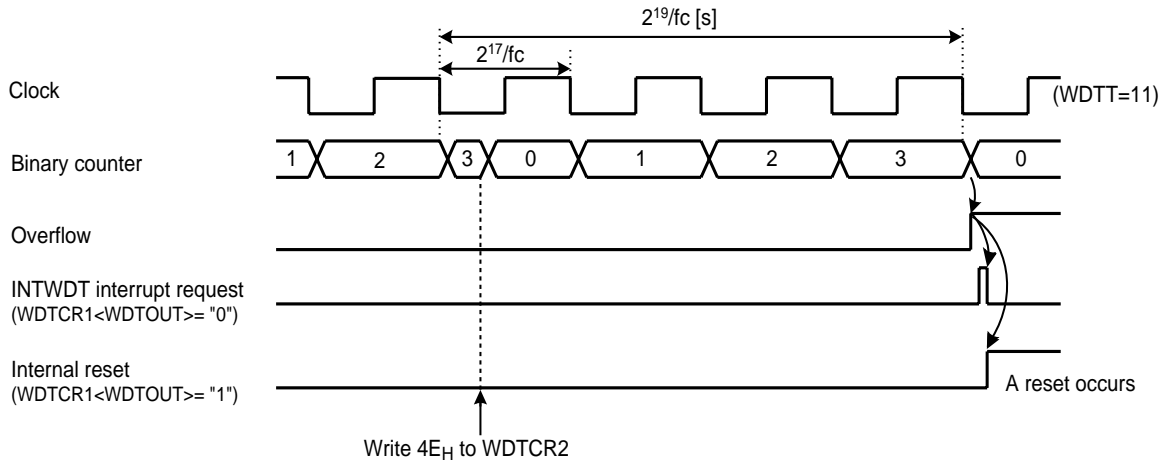
Figure 6-2 Watchdog Timer Interrupt

6.3 Address Trap

The Watchdog Timer Control Register 1 and 2 share the addresses with the control registers to generate address traps.

Watchdog Timer Control Register 1

WDTCR1 (0034H)	7	6	5	4	3	2	1	0	
			ATAS	ATOUT	(WDTEN)	(WDTT)		(WDTOUT)	(Initial value: **11 1001)
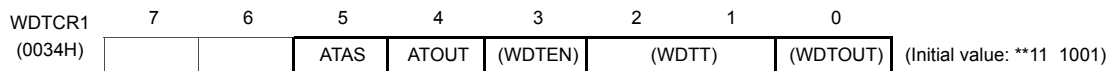
			Write only
ATAS	Select address trap generation in the internal RAM area	0: Generate no address trap 1: Generate address traps (After setting ATAS to "1", writing the control code D2H to WDTCR2 is required)	
ATOUT	Select operation at address trap	0: Interrupt request 1: Reset request	

Watchdog Timer Control Register 2

WDTCR2 (0035H)	7	6	5	4	3	2	1	0	
									(Initial value: **** ****)
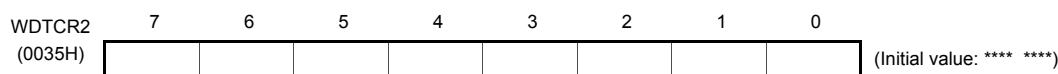
			Write only
WDTCR2	Write Watchdog timer control code and address trap area control code	D2H: Enable address trap area selection (ATRAP control code) 4EH: Clear the watchdog timer binary counter (WDT clear code) B1H: Disable the watchdog timer (WDT disable code) Others: Invalid	

6.3.1 Selection of Address Trap in Internal RAM (ATAS)

WDTCR1<ATAS> specifies whether or not to generate address traps in the internal RAM area. To execute an instruction in the internal RAM area, clear WDTCR1<ATAS> to "0". To enable the WDTCR1<ATAS> setting, set WDTCR1<ATAS> and then write D2H to WDTCR2.

Executing an instruction in the SFR or DBR area generates an address trap unconditionally regardless of the setting in WDTCR1<ATAS>.

6.3.2 Selection of Operation at Address Trap (ATOUT)

When an address trap is generated, either the interrupt request or the reset request can be selected by WDTCR1<ATOUT>.

6.3.3 Address Trap Interrupt (INTATRAP)

While WDTCR1<ATOUT> is "0", if the CPU should start looping for some cause such as noise and an attempt be made to fetch an instruction from the on-chip RAM (while WDTCR1<ATAS> is "1"), DBR or the SFR area, address trap interrupt (INTATRAP) will be generated.

An address trap interrupt is a non-maskable interrupt which can be accepted regardless of the interrupt master flag (IMF).

When an address trap interrupt is generated while the other interrupt including an address trap interrupt is already accepted, the new address trap is processed immediately and the previous interrupt is held pending. Therefore, if address trap interrupts are generated continuously without execution of the RETN instruction, too many levels of nesting may cause a malfunction of the microcontroller.

To generate address trap interrupts, set the stack pointer beforehand.

6.3.4 Address Trap Reset

While WDTCR1<ATOUT> is "1", if the CPU should start looping for some cause such as noise and an attempt be made to fetch an instruction from the on-chip RAM (while WDTCR1<ATAS> is "1"), DBR or the SFR area, address trap reset will be generated.

When an address trap reset request is generated, the internal hardware is reset. The reset time is maximum $24/fc$ [s] (1.5 μs @ fc = 16.0 MHz). After reset, the CPU enters a wait state until the power supply of the flash memory control circuit is stable. The wait time is $2^{10}/fc$ [s] (62.5 μs @ fc = 16.0 MHz). The CPU halts and remains in a wait state, and restarts operation after the wait time.

Note: When an address trap reset is generated in the SLOW1 mode, the reset time is maximum 24/fc (high-frequency clock) since the high-frequency clock oscillator is restarted. However, when crystals have inaccuracies upon start of the high-frequency clock oscillator, the reset time should be considered as an approximate value because it has slight errors.

7. 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).

7.1 Time Base Timer

7.1.1 Configuration
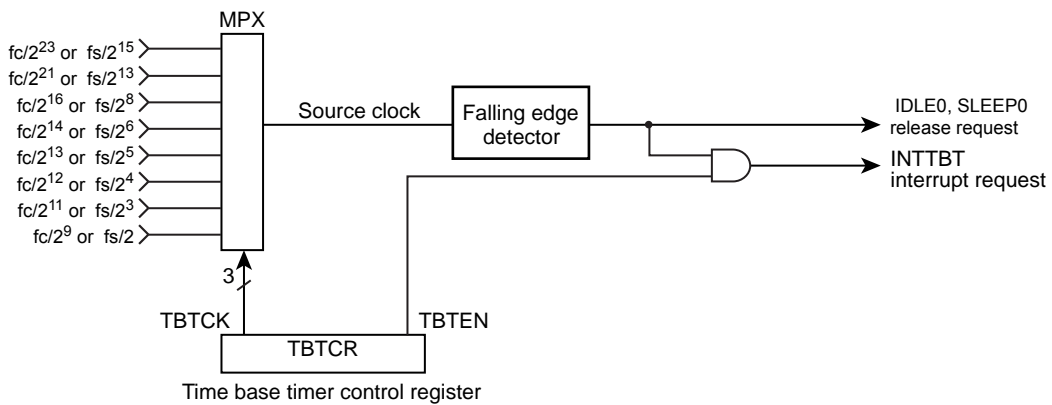
Figure 7-1 Time Base Timer configuration

7.1.2 Control

Time Base Timer is controlled by Time Base Timer control register (TBTCR).

Time Base Timer Control Register

	7	6	5	4	3	2	1	0	
TBTCR (0036H)	(DVOEN)	(DVOCK)		(DV7CK)	TBTEN	TBTCK			(Initial Value: 0000 0000)

				NORMAL1/2, IDLE1/2 Mode		SLOW1/2 SLEEP1/2 Mode	
TBTEN	Time Base Timer enable / disable	0: Disable 1: Enable					
TBTCK	Time Base Timer interrupt Frequency select : [Hz]			DV7CK = 0	DV7CK = 1		R/W
			000	$fc/2^{23}$	$fs/2^{15}$	$fs/2^{15}$	
			001	$fc/2^{21}$	$fs/2^{13}$	$fs/2^{13}$	
			010	$fc/2^{16}$	$fs/2^{8}$	–	
			011	$fc/2^{14}$	$fs/2^{6}$	–	
			100	$fc/2^{13}$	$fs/2^{5}$	–	
			101	$fc/2^{12}$	$fs/2^{4}$	–	
			110	$fc/2^{11}$	$fs/2^{3}$	–	
			111	$fc/2^{9}$	$fs/2$	–	

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

Note 2: The interrupt frequency (TBTCK) must be selected with the time base timer disabled (TBTEN="0"). (The interrupt frequency must not be changed with the disable from the enable state.) Both frequency selection and enabling can be performed simultaneously.

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

LD	(TBTCR) , 00000010B	; TBTCK $\leftarrow$ 010
LD	(TBTCR) , 00001010B	; TBTEN $\leftarrow$ 1
DI		; IMF $\leftarrow$ 0
SET	(EIRL) . 6	

Table 7-1   Time Base Timer Interrupt Frequency (Example : fc = 16.0 MHz, fs = 32.768 kHz)

TBTCK	Time Base Timer Interrupt Frequency [Hz]		
	NORMAL1/2, IDLE1/2 Mode	NORMAL1/2, IDLE1/2 Mode	SLOW1/2, SLEEP1/2 Mode
	DV7CK = 0	DV7CK = 1	
000	1.91	1	1
001	7.63	4	4
010	244.14	128	–
011	976.56	512	–
100	1953.13	1024	–
101	3906.25	2048	–
110	7812.5	4096	–
111	31250	16384	–

7.1.3   Function

An INTTBT (Time Base Timer Interrupt) is generated on the first falling edge of source clock (The divider output of the timing generator which is selected by TBTCK.) after 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 (Figure 7-2).
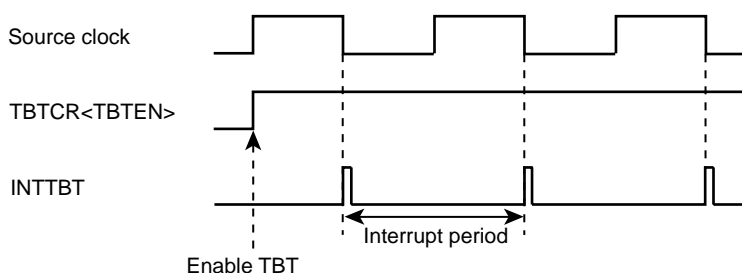
Figure 7-2 Time Base Timer Interrupt

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

Approximately 50% duty pulse can be output using the divider output circuit, which is useful for piezoelectric buzzer drive. Divider output is from $\overline{\text{DVO}}$ pin.

7.2.1 Configuration

(a) configuration   (b) Timing chart
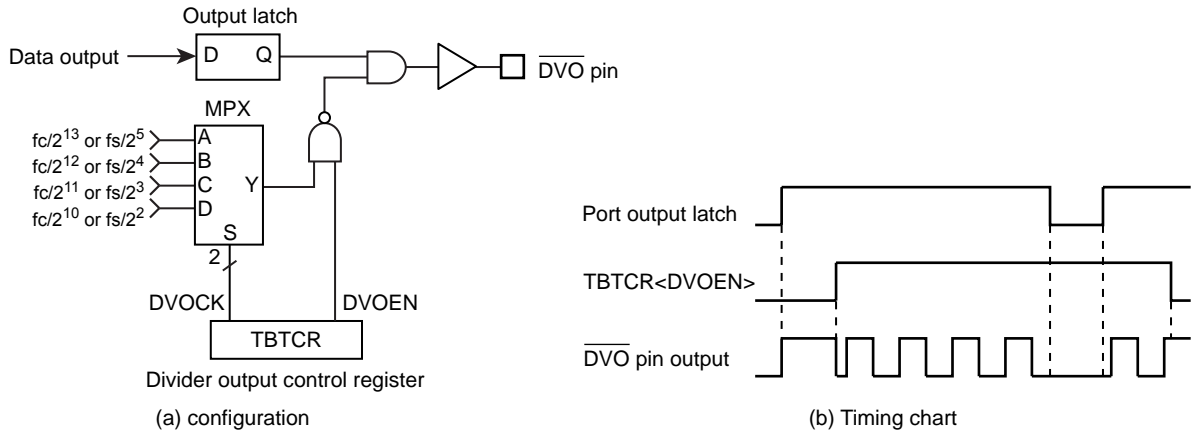
Figure 7-3 Divider Output

7.2.2 Control

The Divider Output is controlled by the Time Base Timer Control Register.

Time Base Timer Control Register

	7	6	5	4	3	2	1	0	
TBTCR (0036H)	DVOEN	DVOCK		(DV7CK)	(TBTEN)	(TBTCK)			(Initial value: 0000 0000)
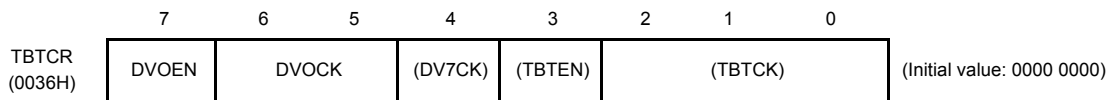
DVOEN	Divider output enable / disable	0: Disable 1: Enable					R/W
DVOCK	Divider Output ($\overline{\text{DVO}}$) frequency selection: [Hz]		NORMAL1/2, IDLE1/2 Mode		SLOW1/2 SLEEP1/2 Mode		R/W
			DV7CK = 0	DV7CK = 1			
		00	$fc/2^{13}$	$fs/2^5$	$fs/2^5$		
		01	$fc/2^{12}$	$fs/2^4$	$fs/2^4$		
		10	$fc/2^{11}$	$fs/2^3$	$fs/2^3$		
		11	$fc/2^{10}$	$fs/2^2$	$fs/2^2$		

Note: Selection of divider output frequency (DVOCK) must be made while divider output is disabled (DVOEN="0"). Also, in other words, when changing the state of the divider output frequency from enabled (DVOEN="1") to disable(DVOEN="0"), do not change the setting of the divider output frequency.

Example :1.95 kHz pulse output (fc = 16.0 MHz)

```
LD          (TBTCR) , 00000000B              ; DVOCK ← "00"

LD          (TBTCR) , 10000000B              ; DVOEN ← "1"
```

Table 7-2   Divider Output Frequency (Example : fc = 16.0 MHz, fs = 32.768 kHz)

DVOCK	Divider Output Frequency [Hz]		SLOW1/2, SLEEP1/2 Mode
	NORMAL1/2, IDLE1/2 Mode		
	DV7CK = 0	DV7CK = 1	
00	1.953 k	1.024 k	1.024 k
01	3.906 k	2.048 k	2.048 k
10	7.813 k	4.096 k	4.096 k
11	15.625 k	8.192 k	8.192 k

8. 18-Bit Timer/Counter (TC1)

8.1 Configuration
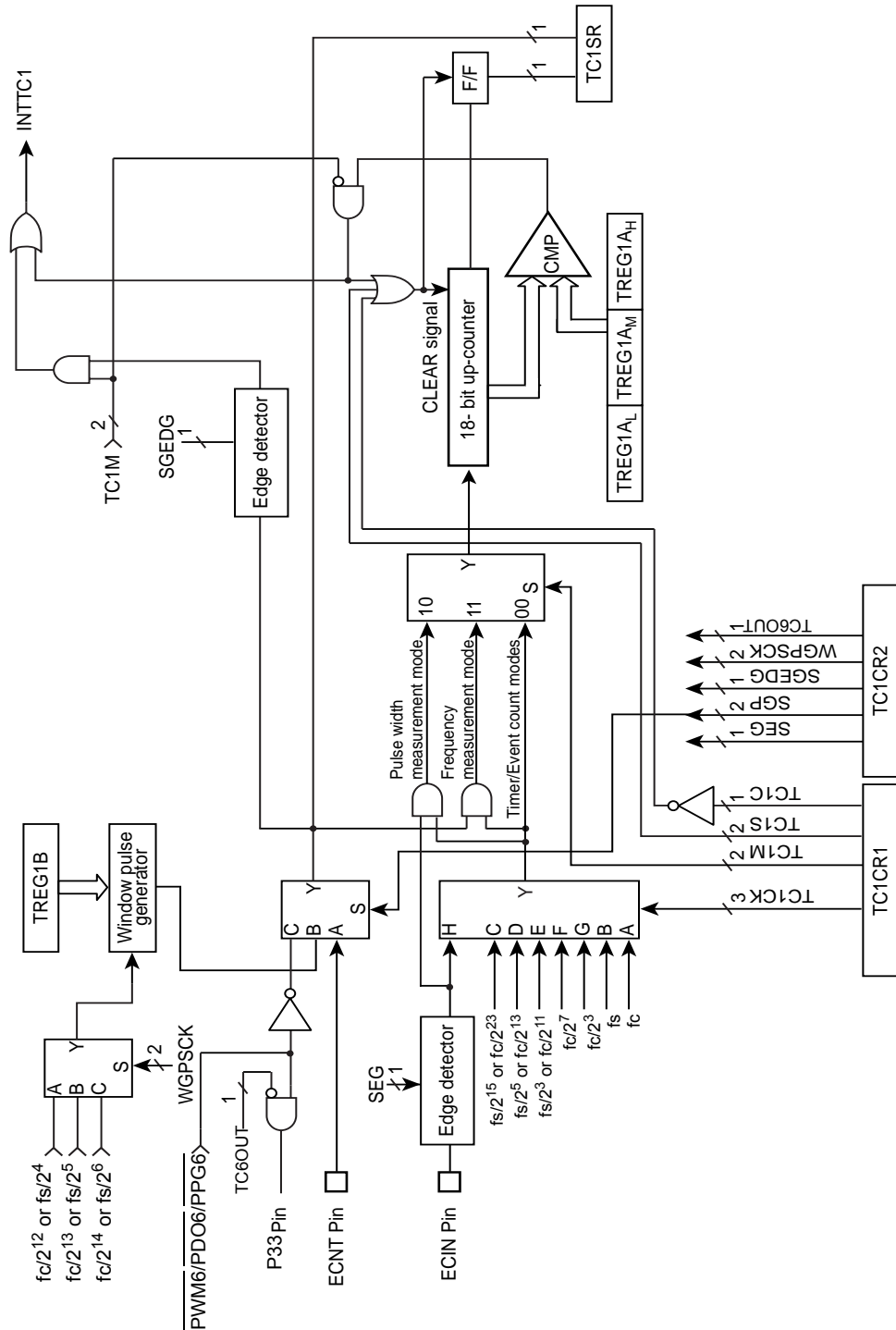
Figure 8-1 Timer/Counter1

8.2 Control

The Timer/counter 1 is controlled by timer/counter 1 control registers (TC1CR1/TC1CR2), an 18-bit timer register (TREG1A), and an 8-bit internal window gate pulse setting register (TREG1B).

Timer register

	7	6	5	4	3	2	1	0	
TREG1AH (0012H) R/W	–	–	–	–	–	–	TREG1AH		(Initial value: **** **00)

	7	6	5	4	3	2	1	0	
TREG1AM (0011H) R/W	TREG1AM								(Initial value: 0000 0000)

	7	6	5	4	3	2	1	0	
TREG1AL (0010H) R/W	TREG1AL								(Initial value: 0000 0000)

	7	6	5	4	3	2	1	0	
TREG1B (0013H)	Ta				Tb				(Initial value: 0000 0000)

		WGPSCK	NORMAL1/2, IDLE1/2 modes		SLOW1/2, SLEEP1/2 modes	
			DV7CK=0	DV7CK=1		
Ta	Setting "H" level period of the window gate pulse	00 01 10	$(16 - Ta) \times 2^{12}/fc$ $(16 - Ta) \times 2^{13}/fc$ $(16 - Ta) \times 2^{14}/fc$	$(16 - Ta) \times 2^{4}/fs$ $(16 - Ta) \times 2^{5}/fs$ $(16 - Ta) \times 2^{6}/fs$	$(16 - Ta) \times 2^{4}/fs$ $(16 - Ta) \times 2^{5}/fs$ $(16 - Ta) \times 2^{6}/fs$	R/W
Tb	Setting "L" level period of the window gate pulse	00 01 10	$(16 - Tb) \times 2^{12}/fc$ $(16 - Tb) \times 2^{13}/fc$ $(16 - Tb) \times 2^{14}/fc$	$(16 - Tb) \times 2^{4}/fs$ $(16 - Tb) \times 2^{5}/fs$ $(16 - Tb) \times 2^{6}/fs$	$(16 - Tb) \times 2^{4}/fs$ $(16 - Tb) \times 2^{5}/fs$ $(16 - Tb) \times 2^{6}/fs$	

Timer/counter 1 control register 1

	7	6	5	4	3	2	1	0	
TC1CR1 (0014H)	TC1C	TC1S		TC1CK			TC1M		(Initial value: 1000 1000)
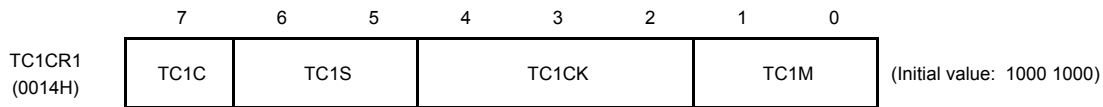
<table>
<tr>
<td rowspan="2">TC1C</td>
<td rowspan="2">Counter/overfow flag controll</td>
<td>0:</td>
<td colspan="4">Clear Counter/overflow flag ("1" is automatically set after clearing.)</td>
<td rowspan="2">R/W</td>
</tr>
<tr>
<td>1:</td>
<td colspan="4">Not clear Counter/overflow flag</td>
</tr>
<tr>
<td rowspan="3">TC1S</td>
<td rowspan="3">TC1 start control</td>
<td>00:</td>
<td colspan="4">Stop and counter clear and overflow flag clear</td>
<td rowspan="3">R/W</td>
</tr>
<tr>
<td>10:</td>
<td colspan="4">Start</td>
</tr>
<tr>
<td>*1:</td>
<td colspan="4">Reserved</td>
</tr>
<tr>
<td rowspan="11">TC1CK</td>
<td rowspan="11">TC1 source clock select</td>
<td></td>
<td colspan="2">NORMAL1/2,IDLE1/2 modes</td>
<td>SLOW1/2
mode</td>
<td>SLEEP1/2
mode</td>
<td rowspan="10">R/W</td>
</tr>
<tr>
<td></td>
<td>DV7CK="0"</td>
<td>DV7CK="1"</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000:</td>
<td>fc</td>
<td>fc</td>
<td>fc</td>
<td>fc</td>
</tr>
<tr>
<td>001:</td>
<td>fs</td>
<td>fs</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>010:</td>
<td>$fc/2^{23}$</td>
<td>$fs/2^{15}$</td>
<td>$fs/2^{15}$</td>
<td>$fs/2^{15}$</td>
</tr>
<tr>
<td>011:</td>
<td>$fc/2^{13}$</td>
<td>$fs/2^5$</td>
<td>$fs/2^5$</td>
<td>$fs/2^5$</td>
</tr>
<tr>
<td>100:</td>
<td>$fc/2^{11}$</td>
<td>$fs/2^3$</td>
<td>$fs/2^3$</td>
<td>$fs/2^3$</td>
</tr>
<tr>
<td>101:</td>
<td>$fc/2^7$</td>
<td>$fc/2^7$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>110:</td>
<td>$fc/2^3$</td>
<td>$fc/2^3$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>111:</td>
<td colspan="4">External clock (ECIN pin input)</td>
</tr>
<tr>
<td rowspan="4">TC1M</td>
<td rowspan="4">TC1 mode select</td>
<td>00:</td>
<td colspan="4">Timer/Event counter mode</td>
<td rowspan="4">R/W</td>
</tr>
<tr>
<td>01:</td>
<td colspan="4">Reserved</td>
</tr>
<tr>
<td>10:</td>
<td colspan="4">Pulse width measurement mode</td>
</tr>
<tr>
<td>11:</td>
<td colspan="4">Frequency measurement mode</td>
</tr>
</table>

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

Note 2: Writing to the low-byte of the timer register 1A (TREG1AL, TREG1AM), the compare function is inhibited until the high-byte (TREG1AH) is written.

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

Note 4: "fc" can be selected as the source clock only in the timer mode during SLOW mode and in the pulse width measurement mode during NORMAL 1/2 or IDLE 1/2 mode.

Note 5: When a read instruction is executed to the timer register (TREG1A), the counter immediate value, not the register set value, is read out. Therefore it is impossible to read out the written value of TREG1A. To read the counter value, the read instruction should be executed when the counter stops to avoid reading unstable value.

Note 6: Set the timer register (TREG1A) to ≥ 1.

Note 7: When using the timer mode and pulse width measurement mode, set TC1CK (TC1 source clock select) to internal clock.

Note 8: When using the event counter mode, set TC1CK (TC1 source clock select) to external clock.

Note 9: Because the read value is different from the written value, do not use read-modify-write instructions to TREG1A.

Note 10:$fc/2^7$, $fc/2^3$ can not be used as source clock in SLOW/SLEEP mode.

Note 11:The read data of bits 7 to 2 in TREG1AH are always "0". (Data "1" can not be written.)

Timer/Counter 1 control register 2

	7	6	5	4	3	2	1	0	
TC1CR2 (0015H)	SEG	SGP		SGEDG	WGPSCK		TC6OUT	"0"	(Initial value: 0000 000*)
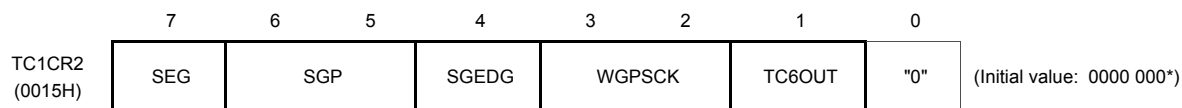
SEG	External input clock (ECIN) edge select	0: 1:	Counts at the falling edge Counts at the both (falling/rising) edges				R/W
SGP	Window gate pulse select	00: 01: 10: 11:	ECNT input Internal window gate pulse (TREG1B) $\overline{PWM6}/\overline{PDO6}/\overline{PPG6}$ (TC6)output Reserved				R/W
SGEDG	Window gate pulse interrupt edge select	0: 1:	Interrupts at the falling edge Interrupts at the falling/rising edges				R/W
WGPSCK	Window gate pulse source clock select		NORMAL1/2,IDLE1/2 modes		SLOW1/2 mode	SLEEP1/2 mode	R/W
			DV7CK="0"	DV7CK="1"			
		00: 01: 10: 11:	2^{12}/fc 2^{13}/fc 2^{14}/fc Reserved	2^4/fs 2^5/fs 2^6/fs Reserved	2^4/fs 2^5/fs 2^6/fs Reserved	2^4/fs 2^5/fs 2^6/fs Reserved	
TC6OUT	TC6 output ($\overline{PWM6}/\overline{PDO6}/\overline{PPG6}$) external output select	0: 1:	Output to P33 No output to P33				R/W

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

Note 2: Set the mode, source clock, and edge (selection) when the TC1 stops (TC1S = 00).

Note 3: If there is no need to use $\overline{PWM6}/\overline{PDO6}/\overline{PPG6}$ as window gate pulse of TC1 always write "0" to TC6OUT.

Note 4: Make sure to write TC1CR2 "0" to bit 0 in TC1CR2.

Note 5: When using the event counter mode or pulse width measurement mode, set SEG to "0".

TC1 status register

	7	6	5	4	3	2	1	0	
TC1SR (0016H)	HECF	HEOVF	"0"	"0"	"0"	"0"	"0"	"0"	(Initial value: 0000 0000)
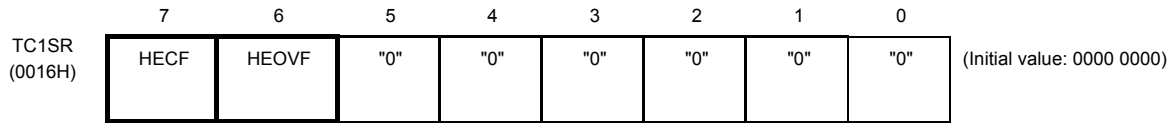
HECF	Operating Status monitor	0: Stop (during Tb) or disable 1: Under counting (during Ta)	Read only
HEOVF	Counter overflow monitor	0: No overflow 1: Overflow status	

8.3 Function

TC1 has four operating modes. The timer mode of the TC1 is used at warm-up when switching form SLOW mode to NORMAL2 mode.

8.3.1 Timer mode

In this mode, counting up is performed using the internal clock. The contents of TREGIA are compared with the contents of up-counter. If a match is found, an INTTC1 interrupt is generated, and the counter is cleared. Counting up resumes after the counter is cleared.

Table 8-1   Source clock (internal clock) of Timer/Counter 1

Source Clock				Resolution		Maximum Time Setting	
NORMAL1/2, IDLE1/2 Mode		SLOW Mode	SLEEP Mode	$fc = 16$ MHz	$fs = 32.768$ kHz	$fc = 16$ MHz	$fs = 32.768$ kHz
DV7CK = 0	DV7CK = 1						
$fc/2^{23}$ [Hz]	$fs/2^{15}$ [Hz]	$fs/2^{15}$ [Hz]	$fs/2^{15}$ [Hz]	0.52 s	1 s	38.2 h	72.8 h
$fc/2^{13}$	$fs/2^5$	$fs/2^5$	$fs/2^5$	512 ms	0.98 ms	2.2 min	4.3 min
$fc/2^{11}$	$fs/2^3$	$fs/2^3$	$fs/2^3$	128 ms	244 ms	0.6 min	1.07 min
$fc/2^7$	$fc/2^7$	-	-	8 ms	-	2.1 s	-
$fc/2^3$	$fc/2^3$	-	-	0.5 ms	-	131.1 ms	-
fc	fc	fc (Note)	-	62.5 ns	-	16.4 ms	-
fs	fs	-	-	-	30.5 ms	-	8 s

Note: When fc is selected for the source clock in SLOW mode, the lower bits 11 of TREG1A is invalid, and a match of the upper bits 7 makes interrupts.
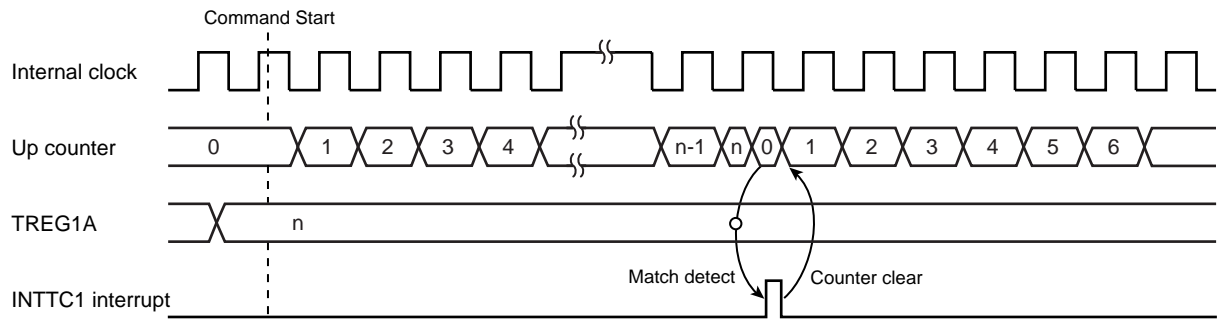
Figure 8-2 Timing chart for timer mode

8.3.2 Event Counter mode

It is a mode to count up at the falling edge of the ECIN pin input. When using this mode, set TC1CR1<TC1CK> to the external clock and then set TC1CR2<SEG> to "0" (Both edges can not be used).

The countents 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. Counting up resumes for ECIN pin input edge each after the counter is cleared.

The maximum applied frequency is $fc/2^4$ [Hz] in NORMAL 1/2 or IDLE 1/2 mode and $fs/2^4$[Hz] in SLOW or SLEEP mode . Two or more machine cycles are required for both the "H" and "L" levels of the pulse width.
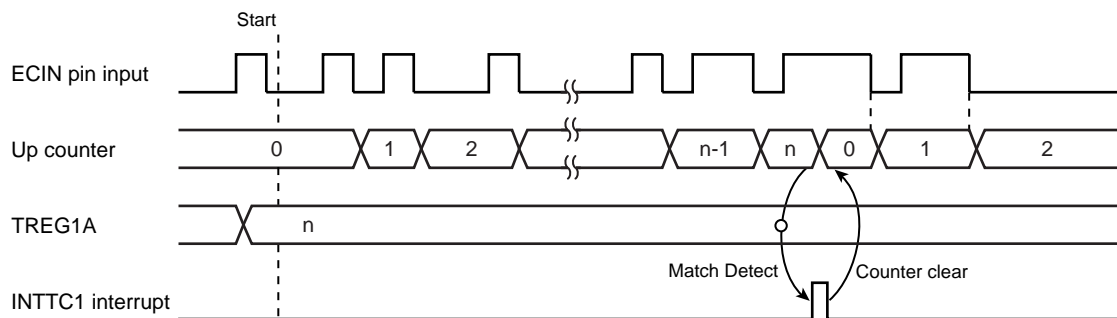
Figure 8-3 Event counter mode timing chart

8.3.3 Pulse Width Measurement mode

In this mode, pulse widths are counted on the falling edge of logical AND-ed pulse between ECIN pin input (window pulse) and the internal clock. When using this mode, set TC1CR1<TC1CK> to suitable internal clock and then set TC1CR2<SEG> to "0" (Both edges can not be used).

An INTTC1 interrupt is generated when the ECIN input detects the falling edge of the window pulse or both rising and falling edges of the window pulse, that can be selected by TC1CR2<SGEDG>.

The contents of TREG1A should be read while the count is stopped (ECIN pin is low), then clear the counter using TC1CR1<TC1C> (Normally, execute these process in the interrupt program).

When the counter is not cleared by TC1CR1<TC1C>, counting-up resumes from previous stopping value. When up counter is counted up from 3FFFFH to 00000H, an overflow occurs. At that time, TC1SR<HEOVF> is set to "1". TC1SR<HEOVF> remains the previous data until the counter is required to be cleared by TC1CR1<TC1C>.

Note: In pulse width measurement mode, if TC1CR1<TC1S> is written to "00" while ECIN input is "1", INTTC1 interrupt occurs. According to the following step, when timer counter is stopped, INTTC1 interrupt latch should be cleared to "0".

Example :

TC1STOP :		
⋮	⋮	
DI		; Clear IMF
CLR	(EIRH). 0	; Clear bit0 of EIRH
LD	(TC1CR1), 00011010B	; Stop timer couter 1
LD	(ILH), 11111110B	; Clear bit0 of ILH
SET	(EIRH). 0	; Set bit0 of EIRH
EI		; Set IMF
⋮	⋮	

Note 1: When SGEDG (window gate pulse interrupt edge select) is set to both edges and ECIN pin input is "1" in the pulse width measurement mode, an INTTC1 interrupt is generated by setting TC1S (TC1 start control) to "10" (start).

Note 2: In the pulse width measurement mode, HECF (operating status monitor) cannot used.

Note 3: Because the up counter is counted on the falling edge of logical AND-ed pulse (between ECIN pin input and the internal clock), if ECIN input becomes falling edge while internal source clock is "H" level, the up counter stops plus "1".
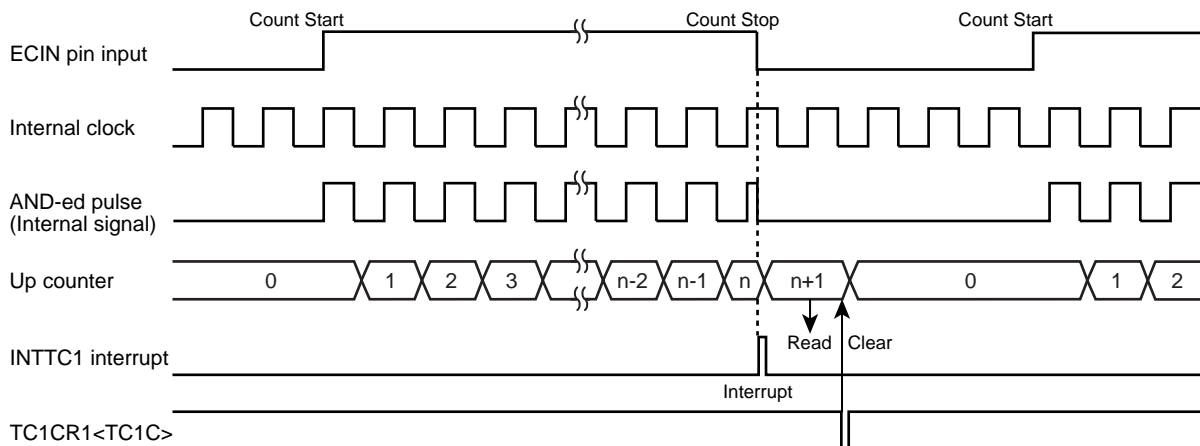
Figure 8-4 Pulse width measurement mode timing chart

8.3.4 Frequency Measurement mode

In this mode, the frequency of ECIN pin input pulse is measured. When using this mode, set TC1CR1<TC1CK> to the external clock.

The edge of the ECIN input pulse is counted during "H" level of the window gate pulse selected by TC1CR2<SGP>. To use ECNT input as a window gate pulse, TC1CR2<SGP> should be set to "00".

An INTTC1 interrupt is generated on the falling edge or both the rising/falling edges of the window gate pulse, that can be selected by TC1CR2<SGEDG>. In the interrupt service program, read the contents of TREG1A while the count is stopped (window gate pulse is low), then clear the counter using TC1CR1<TC1C>. When the counter is not cleared, counting up resumes from previous stopping value.

The window pulse status can be monitored by TC1SR<HECF>.

When up counter is counted up from 3FFFFH to 00000H, an overflow occurs. At that time, TC1SR<HEOVF> is set to "1". TC1SR<HEOVF> remains the previous data until the counter is required to be cleared by TC1CR1<TC1C>.

Using TC6 output ($\overline{PWM6}/\overline{PDO6}/\overline{PPG6}$) for the window gate pulse, external output of $\overline{PWM6}/\overline{PDO6}/\overline{PPG6}$ to P33 can be controlled using TC1CR2<TC6OUT>. Zero-clearing TC1CR2<TC6OUT> outputs $\overline{PWM6}/\overline{PDO6}/\overline{PPG6}$ to P33; setting 1 in TC1CR2<TC6OUT> does not output $\overline{PWM6}/\overline{PDO6}/\overline{PPG6}$ to P33. (TC1CR2<TC6OUT> is used to control output to P33 only. Thus, use the timer counter 6 control register to operate/stop $\overline{PWM6}/\overline{PDO6}/\overline{PPG6}$.)

When the internal window gate pulse is selected, the window gate pulse is set as follows.

Table 8-2 Internal window gate pulse setting time

| | | WGPSCK | NORMAL1/2,IDLE1/2 modes | | SLOW1/2, SLEEP1/2 modes | |
			DV7CK=0	DV7CK=1		
Ta	Setting "H" level period of the window gate pulse	00 01 10	$(16 - Ta) \times 2^{12}/fc$ $(16 - Ta) \times 2^{13}/fc$ $(16 - Ta) \times 2^{14}/fc$	$(16 - Ta) \times 2^4/fs$ $(16 - Ta) \times 2^5/fs$ $(16 - Ta) \times 2^6/fs$	$(16 - Ta) \times 2^4/fs$ $(16 - Ta) \times 2^5/fs$ $(16 - Ta) \times 2^6/fs$	R/W
Tb	Setting "L" level period of the window gate pulse	00 01 10	$(16 - Tb) \times 2^{12}/fc$ $(16 - Tb) \times 2^{13}/fc$ $(16 - Tb) \times 2^{14}/fc$	$(16 - Tb) \times 2^4/fs$ $(16 - Tb) \times 2^5/fs$ $(16 - Tb) \times 2^6/fs$	$(16 - Tb) \times 2^4/fs$ $(16 - Tb) \times 2^5/fs$ $(16 - Tb) \times 2^6/fs$	

The internal window gate pulse consists of "H" level period (Ta) that is counting time and "L" level period (Tb) that is counting stop time. Ta or Tb can be individually set by TREG1B. One cycle contains Ta + Tb.

Note 1: Because the internal window gate pulse is generated in synchronization with the internal divider, it may be delayed for a maximum of one cycle of the source clock (WGPSCK) immediately after start of the timer.

Note 2: Set the internal window gate pulse when the timer counter is not operating or during the Tb period. When Tb is overwritten during the Tb period, the update is valid from the next Tb period.

Note 3: In case of TC1CR2<SEG> = "1", if window gate pulse becomes falling edge, the up counter stops plus "1" regardless of ECIN input level. Therefore, if ECIN is always "H" or "L" level, count value becomes "1".

Note 4: In case of TC1CR2<SEG> = "0", because the up counter is counted on the falling edge of logical AND-ed pulse (between ECIN pin input and window gate pulse), if window gate pulse becomes falling edge while ECIN input is "H" level, the up counter stops plus "1". Therefore, if ECIN input is always "H" level, count value becomes "1".

Table 8-3   Table Setting Ta and Tb (WGPSCK = 10, fc = 16 MHz)

Setting Value	Setting time	Setting Value	Setting time
0	16.38ms	8	8.19ms
1	15.36ms	9	7.17ms
2	14.34ms	A	6.14ms
3	13.31ms	B	5.12ms
4	12.29ms	C	4.10ms
5	11.26ms	D	3.07ms
6	10.24ms	E	2.05ms
7	9.22ms	F	1.02ms

Table 8-4   Table Setting Ta and Tb (WGPSCK = 10, fs = 32.768 kHz)

Setting Valuen	Setting time	Setting Value	Setting time
0	31.25ms	8	15.63ms
1	29.30ms	9	13.67ms
2	27.34ms	A	11.72ms
3	25.39ms	B	9.77ms
4	23.44ms	C	7.81ms
5	21.48ms	D	5.86ms
6	19.53ms	E	3.91ms
7	17.58ms	F	1.95ms
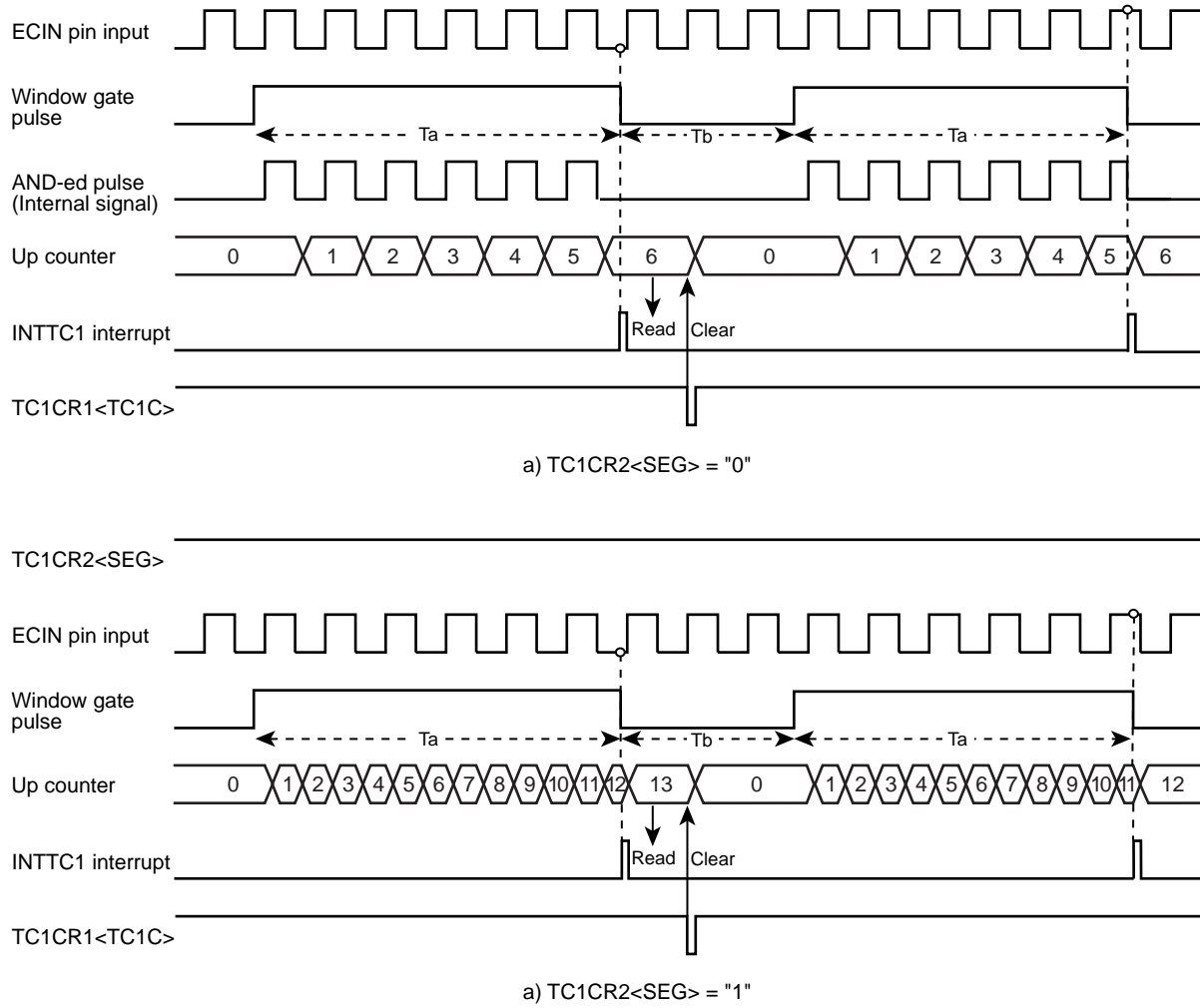
Figure 8-5 Timing chart for the frequency measurement mode (Window gate pulse falling interrupt)

9. 8-Bit TimerCounter (TC3, TC4)

9.1 Configuration
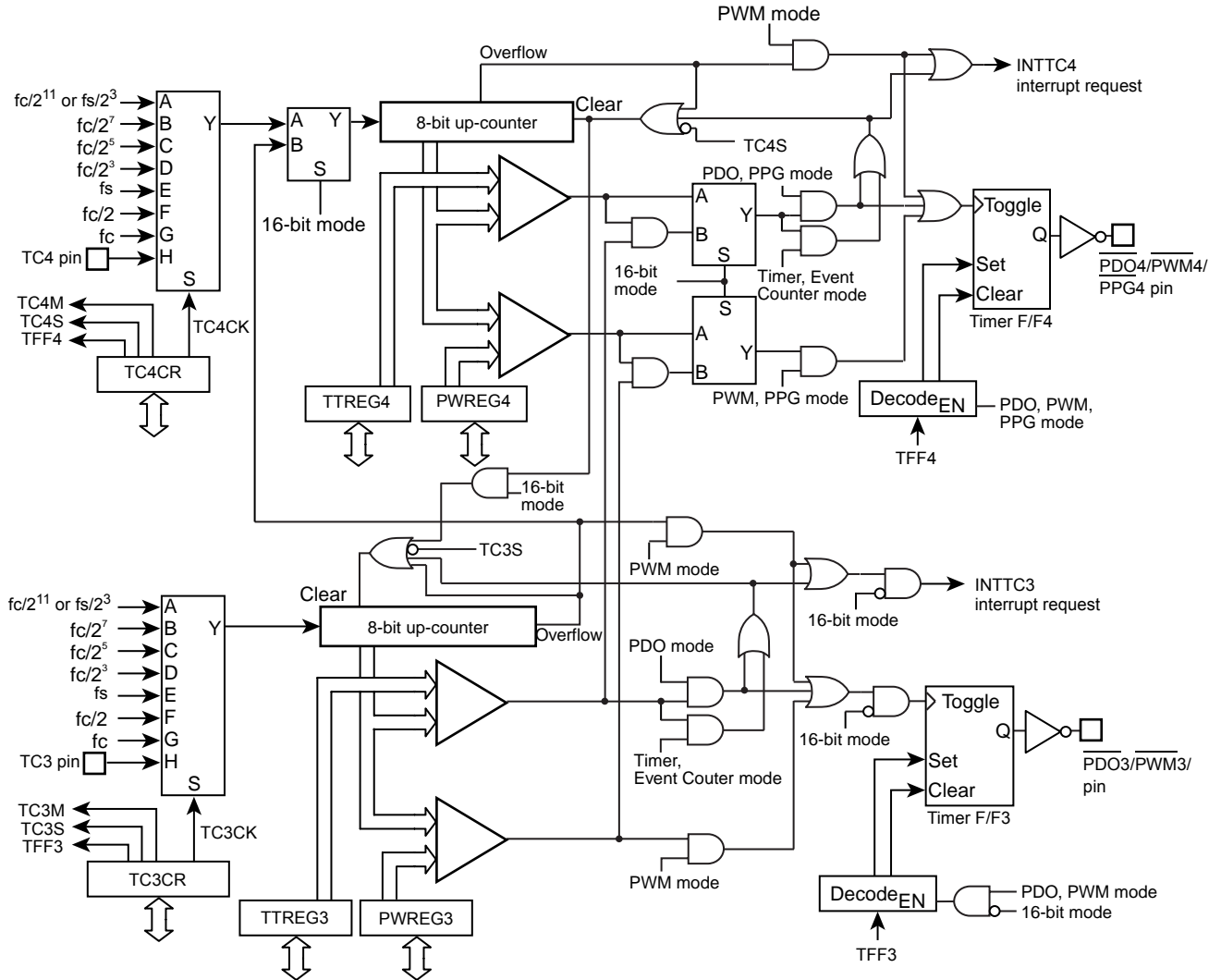
Figure 9-1 8-Bit TimerCounter 3, 4

9.2 TimerCounter Control

The TimerCounter 3 is controlled by the TimerCounter 3 control register (TC3CR) and two 8-bit timer registers (TTREG3, PWREG3).

TimerCounter 3 Timer Register

TTREG3	7	6	5	4	3	2	1	0	
(001CH) R/W									(Initial value: 1111 1111)

PWREG3	7	6	5	4	3	2	1	0	
(0020H) R/W									(Initial value: 1111 1111)
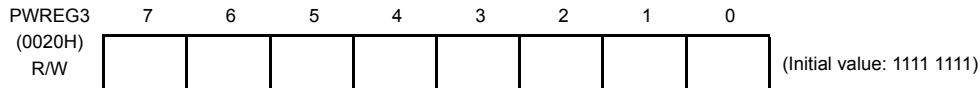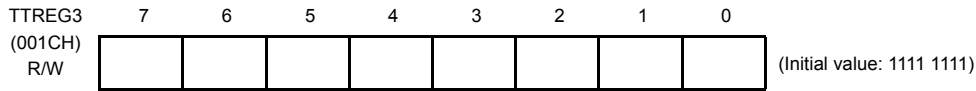
Note 1: Do not change the timer register (TTREG3) setting while the timer is running.

Note 2: Do not change the timer register (PWREG3) setting in the operating mode except the 8-bit and 16-bit PWM modes while the timer is running.

TimerCounter 3 Control Register

TC3CR	7	6	5	4	3	2	1	0	
(0018H)	TFF3	TC3CK			TC3S	TC3M			(Initial value: 0000 0000)
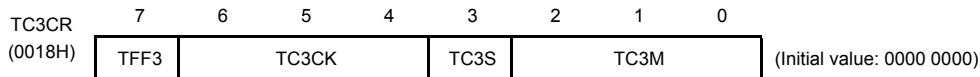
TFF3	Time F/F3 control	0: Clear 1: Set			R/W
			NORMAL1/2, IDLE1/2 mode		SLOW1/2 SLEEP1/2 mode	
			DV7CK = 0	DV7CK = 1		
TC3CK	Operating clock selection [Hz]	000	$fc/2^{11}$	$fs/2^3$	$fs/2^3$	R/W
		001	$fc/2^7$	$fc/2^7$	–	
		010	$fc/2^5$	$fc/2^5$	–	
		011	$fc/2^3$	$fc/2^3$	–	
		100	fs	fs	fs	
		101	fc/2	fc/2	–	
		110	fc	fc	fc (Note 8)	
		111	TC3 pin input			
TC3S	TC3 start control	0: Operation stop and counter clear 1: Operation start			R/W
TC3M	TC3M operating mode select	000: 8-bit timer/event counter mode 001: 8-bit programmable divider output (PDO) mode 010: 8-bit pulse width modulation (PWM) output mode 011: 16-bit mode (Each mode is selectable with TC4M.) 1**: Reserved			R/W

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

Note 2: Do not change the TC3M, TC3CK and TFF3 settings while the timer is running.

Note 3: To stop the timer operation (TC3S= 1 → 0), do not change the TC3M, TC3CK and TFF3 settings. To start the timer operation (TC3S= 0 → 1), TC3M, TC3CK and TFF3 can be programmed.

Note 4: To use the TimerCounter in the 16-bit mode, set the operating mode by programming TC4CR<TC4M>, where TC3M must be fixed to 011.

Note 5: To use the TimerCounter in the 16-bit mode, select the source clock by programming TC3CK. Set the timer start control and timer F/F control by programming TC4CR<TC4S> and TC4CR<TFF4>, respectively.

Note 6: The operating clock settings are limited depending on the timer operating mode. For the detailed descriptions, see Table 9-1 and Table 9-2.

Note 7: The timer register settings are limited depending on the timer operating mode. For the detailed descriptions, see Table 9-3.

Note 8: The operating clock fc in the SLOW or SLEEP mode can be used only as the high-frequency warm-up mode.

The TimerCounter 4 is controlled by the TimerCounter 4 control register (TC4CR) and two 8-bit timer registers (TTREG4 and PWREG4).

TimerCounter 4 Timer Register

TTREG4	7	6	5	4	3	2	1	0	
(001DH) R/W									(Initial value: 1111 1111)

PWREG4	7	6	5	4	3	2	1	0	
(0021H) R/W									(Initial value: 1111 1111)
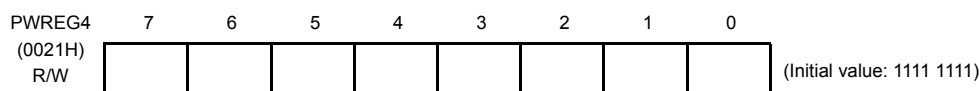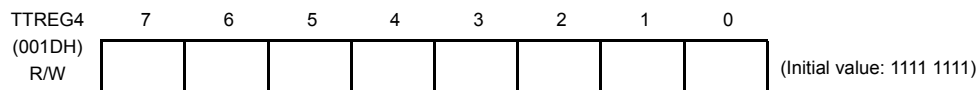
Note 1: Do not change the timer register (TTREG4) setting while the timer is running.

Note 2: Do not change the timer register (PWREG4) setting in the operating mode except the 8-bit and 16-bit PWM modes while the timer is running.

TimerCounter 4 Control Register

TC4CR	7	6	5	4	3	2	1	0	
(0019H)	TFF4	TC4CK			TC4S	TC4M			(Initial value: 0000 0000)
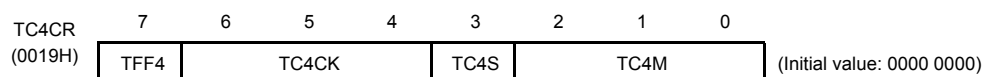
TFF4	Timer F/F4 control	0: Clear 1: Set				R/W
TC4CK	Operating clock selection [Hz]		NORMAL1/2, IDLE1/2 mode		SLOW1/2 SLEEP1/2 mode	R/W
			DV7CK = 0	DV7CK = 1		
		000	$fc/2^{11}$	$fs/2^3$	$fs/2^3$	
		001	$fc/2^7$	$fc/2^7$	–	
		010	$fc/2^5$	$fc/2^5$	–	
		011	$fc/2^3$	$fc/2^3$	–	
		100	fs	fs	fs	
		101	fc/2	fc/2	–	
		110	fc	fc	–	
		111	TC4 pin input			
TC4S	TC4 start control	0: Operation stop and counter clear 1: Operation start				R/W
TC4M	TC4M operating mode select	000: 8-bit timer/event counter mode 001: 8-bit programmable divider output (PDO) mode 010: 8-bit pulse width modulation (PWM) output mode 011: Reserved 100: 16-bit timer/event counter mode 101: Warm-up counter mode 110: 16-bit pulse width modulation (PWM) output mode 111: 16-bit PPG mode				R/W

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

Note 2: Do not change the TC4M, TC4CK and TFF4 settings while the timer is running.

Note 3: To stop the timer operation (TC4S= 1 → 0), do not change the TC4M, TC4CK and TFF4 settings.
    To start the timer operation (TC4S= 0 → 1), TC4M, TC4CK and TFF4 can be programmed.

Note 4: When TC4M= 1** (upper byte in the 16-bit mode), the source clock becomes the TC3 overflow signal regardless of the TC4CK setting.

Note 5: To use the TimerCounter in the 16-bit mode, select the operating mode by programming TC4M, where TC3CR<TC3M> must be set to 011.

Note 6: To the TimerCounter in the 16-bit mode, select the source clock by programming TC3CR<TC3CK>. Set the timer start control and timer F/F control by programming TC4S and TFF4, respectively.

Note 7: The operating clock settings are limited depending on the timer operating mode. For the detailed descriptions, see Table 9-1 and Table 9-2.

Note 8: The timer register settings are limited depending on the timer operating mode. For the detailed descriptions, see Table 9-3.

Table 9-1   Operating Mode and Selectable Source Clock (NORMAL1/2 and IDLE1/2 Modes)

Operating mode	$fc/2^{11}$ or $fs/2^3$	$fc/2^7$	$fc/2^5$	$fc/2^3$	fs	fc/2	fc	TC3 pin input	TC4 pin input
8-bit timer	O	O	O	O	–	–	–	–	–
8-bit event counter	–	–	–	–	–	–	–	O	O
8-bit PDO	O	O	O	O	–	–	–	–	–
8-bit PWM	O	O	O	O	O	O	O	–	–
16-bit timer	O	O	O	O	–	–	–	–	–
16-bit event counter	–	–	–	–	–	–	–	O	–
Warm-up counter	–	–	–	–	O	–	–	–	–
16-bit PWM	O	O	O	O	O	O	O	O	–
16-bit PPG	O	O	O	O	–	–	–	O	–

Note 1: For 16-bit operations (16-bit timer/event counter, warm-up counter, 16-bit PWM and 16-bit PPG), set its source clock on lower bit (TC3CK).

Note 2: O : Available source clock

Table 9-2   Operating Mode and Selectable Source Clock (SLOW1/2 and SLEEP1/2 Modes)

Operating mode	$fc/2^{11}$ or $fs/2^3$	$fc/2^7$	$fc/2^5$	$fc/2^3$	fs	fc/2	fc	TC3 pin input	TC4 pin input
8-bit timer	O	–	–	–	–	–	–	–	–
8-bit event counter	–	–	–	–	–	–	–	O	O
8-bit PDO	O	–	–	–	–	–	–	–	–
8-bit PWM	O	–	–	–	O	–	–	–	–
16-bit timer	O	–	–	–	–	–	–	–	–
16-bit event counter	–	–	–	–	–	–	–	O	–
Warm-up counter	–	–	–	–	–	–	O	–	–
16-bit PWM	O	–	–	–	O	–	–	O	–
16-bit PPG	O	–	–	–	–	–	–	O	–

Note1: For 16-bit operations (16-bit timer/event counter, warm-up counter, 16-bit PWM and 16-bit PPG), set its source clock on lower bit (TC3CK).

Note2: O : Available source clock

Table 9-3   Constraints on Register Values Being Compared

Operating mode	Register Value
8-bit timer/event counter	$1 \leq$ (TTREGn) ≤ 255
8-bit PDO	$1 \leq$ (TTREGn) ≤ 255
8-bit PWM	$2 \leq$ (PWREGn) ≤ 254
16-bit timer/event counter	$1 \leq$ (TTREG4, 3) ≤ 65535
Warm-up counter	$256 \leq$ (TTREG4, 3) ≤ 65535
16-bit PWM	$2 \leq$ (PWREG4, 3) ≤ 65534
16-bit PPG	$1 \leq$ (PWREG4, 3) $<$ (TTREG4, 3) ≤ 65535 and (PWREG4, 3) + 1 $<$ (TTREG4, 3)

Note: n = 3 to 4

9.3　Function

The TimerCounter 3 and 4 have the 8-bit timer, 8-bit event counter, 8-bit programmable divider output (PDO), 8-bit pulse width modulation (PWM) output modes. The TimerCounter 3 and 4 (TC3, 4) are cascadable to form a 16-bit timer. The 16-bit timer has the operating modes such as the 16-bit timer, 16-bit event counter, warm-up counter, 16-bit pulse width modulation (PWM) output and 16-bit programmable pulse generation (PPG) modes.

9.3.1　8-Bit Timer Mode (TC3 and 4)

In the timer mode, the up-counter counts up using the internal clock. When a match between the up-counter and the timer register j (TTREGj) value is detected, an INTTCj interrupt is generated and the up-counter is cleared. After being cleared, the up-counter restarts counting.

Note 1: In the timer mode, fix TCjCR<TFFj> to 0. If not fixed, the $\overline{\text{PDOj}}$, $\overline{\text{PWMj}}$ and $\overline{\text{PPGj}}$ pins may output pulses.

Note 2: In the timer mode, do not change the TTREGj setting while the timer is running. Since TTREGj is not in the shift register configuration in the timer mode, the new value programmed in TTREGj is in effect immediately after the programming. Therefore, if TTREGi is changed while the timer is running, an expected operation may not be obtained.

Note 3: j = 3, 4

Table 9-4　Source Clock for TimerCounter 3, 4 (Internal Clock)

Source Clock			Resolution		Maximum Time Setting	
NORMAL1/2, IDLE1/2 mode		SLOW1/2, SLEEP1/2 mode	fc = 16 MHz	fs = 32.768 kHz	fc = 16 MHz	fs = 32.768 kHz
DV7CK = 0	DV7CK = 1					
$fc/2^{11}$ [Hz]	$fs/2^3$ [Hz]	$fs/2^3$ [Hz]	128 μs	244.14 μs	32.6 ms	62.3 ms
$fc/2^7$	$fc/2^7$	–	8 μs	–	2.0 ms	–
$fc/2^5$	$fc/2^5$	–	2 μs	–	510 μs	–
$fc/2^3$	$fc/2^3$	–	500 ns	–	127.5 μs	–

Example : Setting the timer mode with source clock $fc/2^7$ Hz and generating an interrupt 80 μs later
(TimerCounter4, fc = 16.0 MHz)

LD	(TTREG4), 0AH	: Sets the timer register (80 μs$\div2^7$/fc = 0AH).
DI		
SET	(EIRE). 1	: Enables INTTC4 interrupt.
EI		
LD	(TC4CR), 00010000B	: Sets the operating clock to $fc/2^7$, and 8-bit timer mode.
LD	(TC4CR), 00011000B	: Starts TC4.
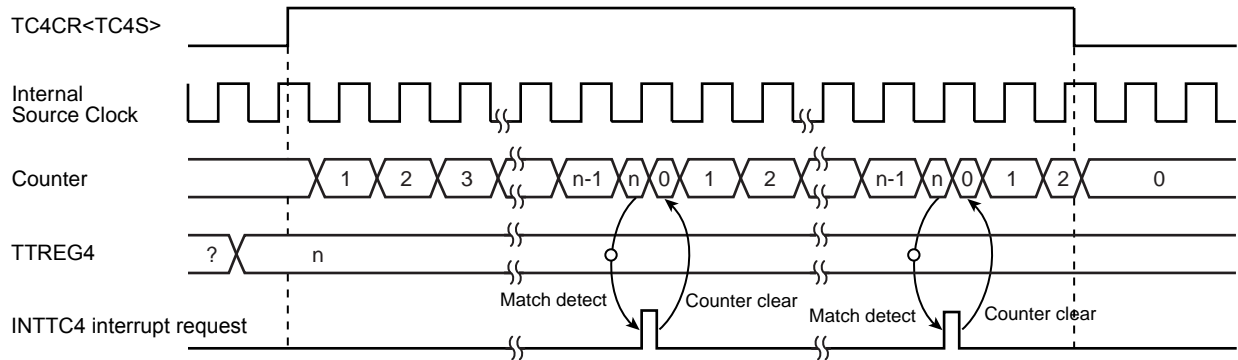
Figure 9-2 8-Bit Timer Mode Timing Chart (TC4)

9.3.2   8-Bit Event Counter Mode (TC3, 4)

In the 8-bit event counter mode, the up-counter counts up at the falling edge of the input pulse to the TCj pin. When a match between the up-counter and the TTREGj value is detected, an INTTCj interrupt is generated and the up-counter is cleared. After being cleared, the up-counter restarts counting at the falling edge of the input pulse to the TCj pin. Two machine cycles are required for the low- or high-level pulse input to the TCj pin. Therefore, a maximum frequency to be supplied is $fc/2^4$ Hz in the NORMAL1/2 or IDLE1/2 mode, and $fs/2^4$ Hz in the SLOW1/2 or SLEEP1/2 mode.

> Note 1: In the event counter mode, fix TCjCR<TFFj> to 0. If not fixed, the $\overline{PDOj}$, $\overline{PWMj}$ and $\overline{PPGj}$ pins may output pulses.
>
> Note 2: In the event counter mode, do not change the TTREGj setting while the timer is running. Since TTREGj is not in the shift register configuration in the event counter mode, the new value programmed in TTREGj is in effect immediately after the programming. Therefore, if TTREGi is changed while the timer is running, an expected operation may not be obtained.
>
> Note 3: j = 3, 4

Figure 9-3 8-Bit Event Counter Mode Timing Chart (TC4)

9.3.3   8-Bit Programmable Divider Output (PDO) Mode (TC3, 4)

This mode is used to generate a pulse with a 50% duty cycle from the $\overline{PDOj}$ pin.

In the PDO mode, the up-counter counts up using the internal clock. When a match between the up-counter and the TTREGj value is detected, the logic level output from the $\overline{PDOj}$ pin is switched to the opposite state and the up-counter is cleared. The INTTCj interrupt request is generated at the time. The logic state opposite to the timer F/Fj logic level is output from the $\overline{PDOj}$ pin. An arbitrary value can be set to the timer F/Fj by TCjCR<TFFj>. Upon reset, the timer F/Fj value is initialized to 0.

To use the programmable divider output, set the output latch of the I/O port to 1.

Example :Generating 1024 Hz pulse using TC4 (fc = 16.0 MHz)

Setting port

LD   (TTREG4), 3DH   : $1/1024 \div 2^7 / fc \div 2$ = 3DH

LD   (TC4CR), 00010001B   : Sets the operating clock to $fc/2^7$, and 8-bit PDO mode.

LD   (TC4CR), 00011001B   : Starts TC4.

Note 1: In the programmable divider output mode, do not change the TTREGj setting while the timer is running. Since TTREGj is not in the shift register configuration in the programmable divider output mode, the new value programmed in TTREGj is in effect immediately after programming. Therefore, if TTREGi is changed while the timer is running, an expected operation may not be obtained.

Note 2: When the timer is stopped during PDO output, the $\overline{PDOj}$ pin holds the output status when the timer is stopped. To change the output status, program TCjCR<TFFj> after the timer is stopped. Do not change the TCjCR<TFFj> setting upon stopping of the timer.
Example: Fixing the $\overline{PDOj}$ pin to the high level when the TimerCounter is stopped
CLR (TCjCR).3: Stops the timer.
CLR (TCjCR).7: Sets the $\overline{PDOj}$ pin to the high level.

Note 3: j = 3, 4

Figure 9-4 8-Bit PDO Mode Timing Chart (TC4)

9.3.4   8-Bit Pulse Width Modulation (PWM) Output Mode (TC3, 4)

This mode is used to generate a pulse-width modulated (PWM) signals with up to 8 bits of resolution. The up-counter counts up using the internal clock.

When a match between the up-counter and the PWREGj value is detected, the logic level output from the timer F/Fj is switched to the opposite state. The counter continues counting. The logic level output from the timer F/Fj is switched to the opposite state again by the up-counter overflow, and the counter is cleared. The INTTCj interrupt request is generated at this time.

Since the initial value can be set to the timer F/Fj by TCjCR<TFFj>, positive and negative pulses can be generated. Upon reset, the timer F/Fj is cleared to 0.

(The logic level output from the $\overline{PWMj}$ pin is the opposite to the timer F/Fj logic level.)

Since PWREGj in the PWM mode is serially connected to the shift register, the value set to PWREGj can be changed while the timer is running. The value set to PWREGj during a run of the timer is shifted by the INTTCj interrupt request and loaded into PWREGj. While the timer is stopped, the value is shifted immediately after the programming of PWREGj. If executing the read instruction to PWREGj during PWM output, the value in the shift register is read, but not the value set in PWREGj. Therefore, after writing to PWREGj, the reading data of PWREGj is previous value until INTTCj is generated.

For the pin used for PWM output, the output latch of the I/O port must be set to 1.

Note 1: In the PWM mode, program the timer register PWREGj immediately after the INTTCj interrupt request is generated (normally in the INTTCj interrupt service routine.) If the programming of PWREGj and the interrupt request occur at the same time, an unstable value is shifted, that may result in generation of the pulse different from the programmed value until the next INTTCj interrupt request is generated.

Note 2: When the timer is stopped during PWM output, the $\overline{PWMj}$ pin holds the output status when the timer is stopped. To change the output status, program TCjCR<TFFj> after the timer is stopped. Do not change the TCjCR<TFFj> upon stopping of the timer.
Example: Fixing the $\overline{PWMj}$ pin to the high level when the TimerCounter is stopped
CLR (TCjCR).3: Stops the timer.
CLR (TCjCR).7: Sets the $\overline{PWMj}$ pin to the high level.

Note 3: To enter the STOP mode during PWM output, stop the timer and then enter the STOP mode. If the STOP mode is entered without stopping the timer when fc, fc/2 or fs is selected as the source clock, a pulse is output from the $\overline{PWMj}$ pin during the warm-up period time after exiting the STOP mode.

Note 4: j = 3, 4

Table 9-5   PWM Output Mode

Source Clock			Resolution		Repeated Cycle	
NORMAL1/2, IDLE1/2 mode		SLOW1/2, SLEEP1/2 mode	fc = 16 MHz	fs = 32.768 kHz	fc = 16 MHz	fs = 32.768 kHz
DV7CK = 0	DV7CK = 1					
$fc/2^{11}$ [Hz]	$fs/2^3$ [Hz]	$fs/2^3$ [Hz]	128 μs	244.14 μs	32.8 ms	62.5 ms
$fc/2^7$	$fc/2^7$	–	8 μs	–	2.05 ms	–
$fc/2^5$	$fc/2^5$	–	2 μs	–	512 μs	–
$fc/2^3$	$fc/2^3$	–	500 ns	–	128 μs	–
fs	fs	fs	30.5 μs	30.5 μs	7.81 ms	7.81 ms
fc/2	fc/2	–	125 ns	–	32 μs	–
fc	fc	–	62.5 ns	–	16 μs	–

Figure 9-5 8-Bit PWM Mode Timing Chart (TC4)

9.3.5   16-Bit Timer Mode (TC3 and 4)

In the timer mode, the up-counter counts up using the internal clock. The TimerCounter 3 and 4 are cascadable to form a 16-bit timer.

When a match between the up-counter and the timer register (TTREG3, TTREG4) value is detected after the timer is started by setting TC4CR<TC4S> to 1, an INTTC4 interrupt is generated and the up-counter is cleared. After being cleared, the up-counter continues counting. Program the lower byte and upper byte in this order in the timer register. (Programming only the upper or lower byte should not be attempted.)

Note 1: In the timer mode, fix TCjCR<TFFj> to 0. If not fixed, the $\overline{PDOj}$, $\overline{PWMj}$, and $\overline{PPGj}$ pins may output a pulse.

Note 2: In the timer mode, do not change the TTREGj setting while the timer is running. Since TTREGj is not in the shift register configuration in the timer mode, the new value programmed in TTREGj is in effect immediately after programming of TTREGj. Therefore, if TTREGj is changed while the timer is running, an expected operation may not be obtained.
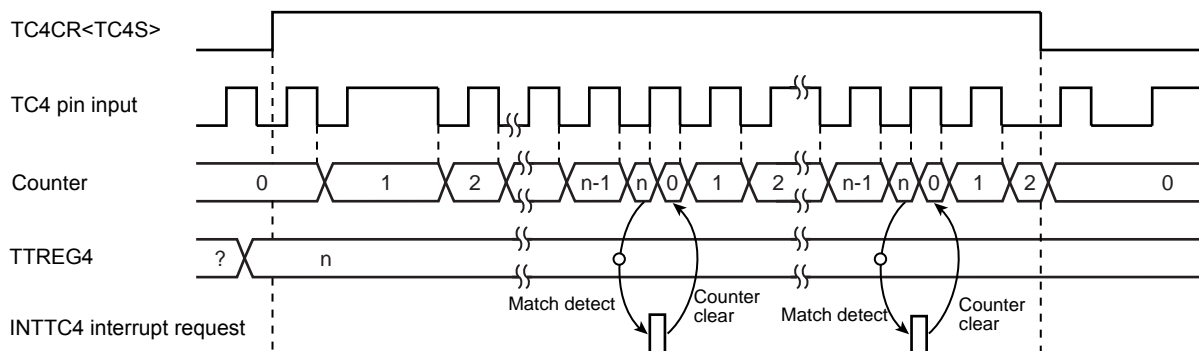
Note 3: j = 3, 4

Table 9-6   Source Clock for 16-Bit Timer Mode

Source Clock			Resolution		Maximum Time Setting	
NORMAL1/2, IDLE1/2 mode		SLOW1/2, SLEEP1/2 mode	fc = 16 MHz	fs = 32.768 kHz	fc = 16 MHz	fs = 32.768 kHz
DV7CK = 0	DV7CK = 1					
$fc/2^{11}$	$fs/2^3$	$fs/2^3$	128 μs	244.14 μs	8.39 s	16 s
$fc/2^7$	$fc/2^7$	–	8 μs	–	524.3 ms	–
$fc/2^5$	$fc/2^5$	–	2 μs	–	131.1 ms	–
$fc/2^3$	$fc/2^3$	–	500 ns	–	32.8 ms	–

Example : Setting the timer mode with source clock $fc/2^7$ Hz, and generating an interrupt 300 ms later
(fc = 16.0 MHz)

LDW	(TTREG3), 927CH	: Sets the timer register (300 ms÷2^7/fc = 927CH).
DI		
SET	(EIRE). 1	: Enables INTTC4 interrupt.
EI		
LD	(TC3CR), 13H	:Sets the operating clock to $fc/2^7$, and 16-bit timer mode (lower byte).
LD	(TC4CR), 04H	: Sets the 16-bit timer mode (upper byte).
LD	(TC4CR), 0CH	: Starts the timer.
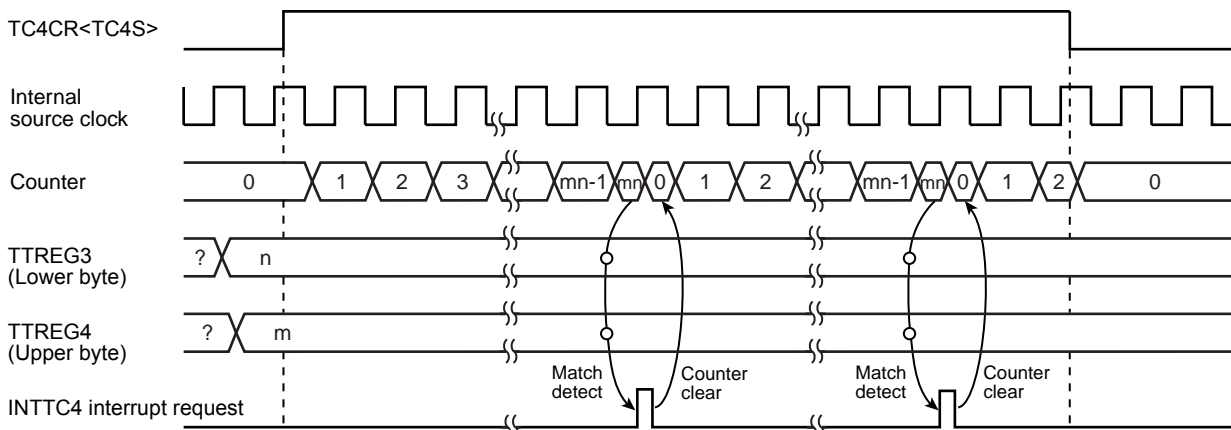
Figure 9-6 16-Bit Timer Mode Timing Chart (TC3 and TC4)

9.3.6 16-Bit Event Counter Mode (TC3 and 4)

In the event counter mode, the up-counter counts up at the falling edge to the TC3 pin. The TimerCounter 3 and 4 are cascadable to form a 16-bit event counter.

When a match between the up-counter and the timer register (TTREG3, TTREG4) value is detected after the timer is started by setting TC4CR<TC4S> to 1, an INTTC4 interrupt is generated and the up-counter is cleared.

After being cleared, the up-counter restarts counting at the falling edge of the input pulse to the TC3 pin. Two machine cycles are required for the low- or high-level pulse input to the TC3 pin.

Therefore, a maximum frequency to be supplied is $fc/2^4$ Hz in the NORMAL1/2 or IDLE1/2 mode, and $fs/2^4$ in the SLOW1/2 or SLEEP1/2 mode. Program the lower byte (TTREG3), and upper byte (TTREG4) in this order in the timer register. (Programming only the upper or lower byte should not be attempted.)

Note 1: In the event counter mode, fix TCjCR<TFFj> to 0. If not fixed, the $\overline{PDOj}$, $\overline{PWMj}$ and $\overline{PPGj}$ pins may output pulses.
Note 2: In the event counter mode, do not change the TTREGj setting while the timer is running. Since TTREGj is not in the shift register configuration in the event counter mode, the new value programmed in TTREGj is in effect immediately after the programming. Therefore, if TTREGj is changed while the timer is running, an expected operation may not be obtained.
Note 3: j = 3, 4

9.3.7 16-Bit Pulse Width Modulation (PWM) Output Mode (TC3 and 4)

This mode is used to generate a pulse-width modulated (PWM) signals with up to 16 bits of resolution. The TimerCounter 3 and 4 are cascadable to form the 16-bit PWM signal generator.

The counter counts up using the internal clock or external clock.

When a match between the up-counter and the timer register (PWREG3, PWREG4) value is detected, the logic level output from the timer F/F4 is switched to the opposite state. The counter continues counting. The logic level output from the timer F/F4 is switched to the opposite state again by the counter overflow, and the counter is cleared. The INTTC4 interrupt is generated at this time.

Two machine cycles are required for the high- or low-level pulse input to the TC3 pin. Therefore, a maximum frequency to be supplied is $fc/2^4$ Hz in the NORMAL1/2 or IDLE1/2 mode, and $fs/2^4$ to in the SLOW1/2 or SLEEP1/2 mode.

Since the initial value can be set to the timer F/F4 by TC4CR<TFF4>, positive and negative pulses can be generated. Upon reset, the timer F/F4 is cleared to 0.

(The logic level output from the $\overline{PWM}4$ pin is the opposite to the timer F/F4 logic level.)

Since PWREG4 and 3 in the PWM mode are serially connected to the shift register, the values set to PWREG4 and 3 can be changed while the timer is running. The values set to PWREG4 and 3 during a run of the timer are shifted by the INTTCj interrupt request and loaded into PWREG4 and 3. While the timer is stopped, the values are shifted immediately after the programming of PWREG4 and 3. Set the lower byte (PWREG3) and upper byte (PWREG4) in this order to program PWREG4 and 3. (Programming only the lower or upper byte of the register should not be attempted.)

If executing the read instruction to PWREG4 and 3 during PWM output, the values set in the shift register is read, but not the values set in PWREG4 and 3. Therefore, after writing to the PWREG4 and 3, reading data of PWREG4 and 3 is previous value until INTTC4 is generated.

For the pin used for PWM output, the output latch of the I/O port must be set to 1.

Note 1: In the PWM mode, program the timer register PWREG4 and 3 immediately after the INTTC4 interrupt request is generated (normally in the INTTC4 interrupt service routine.) If the programming of PWREGj and the interrupt request occur at the same time, an unstable value is shifted, that may result in generation of pulse different from the programmed value until the next INTTC4 interrupt request is generated.
Note 2: When the timer is stopped during PWM output, the $\overline{PWM}4$ pin holds the output status when the timer is stopped. To change the output status, program TC4CR<TFF4> after the timer is stopped. Do not program TC4CR<TFF4> upon stopping of the timer.
Example: Fixing the $\overline{PWM}4$ pin to the high level when the TimerCounter is stopped

CLR (TC4CR).3: Stops the timer.

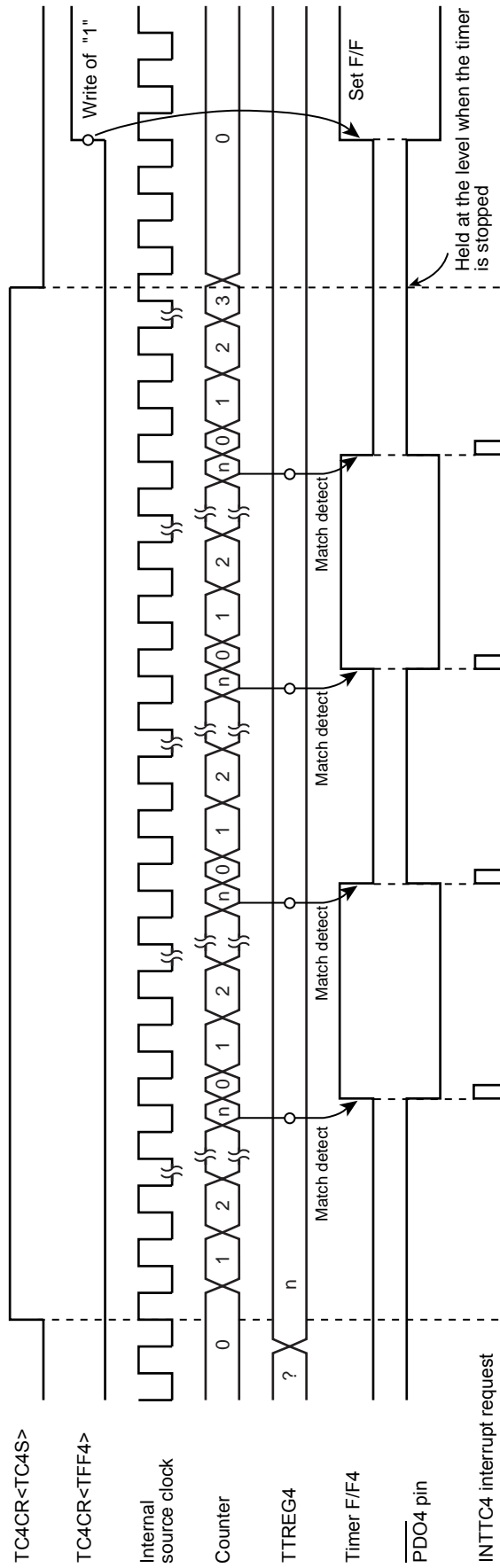CLR (TC4CR).7 : Sets the $\overline{\text{PWM}}4$ pin to the high level.

Note 3: To enter the STOP mode, stop the timer and then enter the STOP mode. If the STOP mode is entered without stopping of the timer when fc, fc/2 or fs is selected as the source clock, a pulse is output from the $\overline{\text{PWM}}4$ pin during the warm-up period time after exiting the STOP mode.

Table 9-7   16-Bit PWM Output Mode

Source Clock			Resolution		Repeated Cycle	
NORMAL1/2, IDLE1/2 mode		SLOW1/2, SLEEP1/2 mode	fc = 16 MHz	fs = 32.768 kHz	fc = 16 MHz	fs = 32.768 kHz
DV7CK = 0	DV7CK = 1					
$fc/2^{11}$	$fs/2^3$ [Hz]	$fs/2^3$ [Hz]	128 µs	244.14 µs	8.39 s	16 s
$fc/2^7$	$fc/2^7$	–	8 µs	–	524.3 ms	–
$fc/2^5$	$fc/2^5$	–	2 µs	–	131.1 ms	–
$fc/2^3$	$fc/2^3$	–	500 ns	–	32.8 ms	–
fs	fs	fs	30.5 µs	30.5 µs	2 s	2 s
fc/2	fc/2	–	125 ns	–	8.2 ms	–
fc	fc	–	62.5 ns	–	4.1 ms	–

Example :Generating a pulse with 1-ms high-level width and a period of 32.768 ms (fc = 16.0 MHz)

Setting ports

LDW	(PWREG3), 07D0H	: Sets the pulse width.
LD	(TC3CR), 33H	: Sets the operating clock to $fc/2^3$, and 16-bit PWM output mode (lower byte).
LD	(TC4CR), 056H	: Sets TFF4 to the initial value 0, and 16-bit PWM signal generation mode (upper byte).
LD	(TC4CR), 05EH	: Starts the timer.
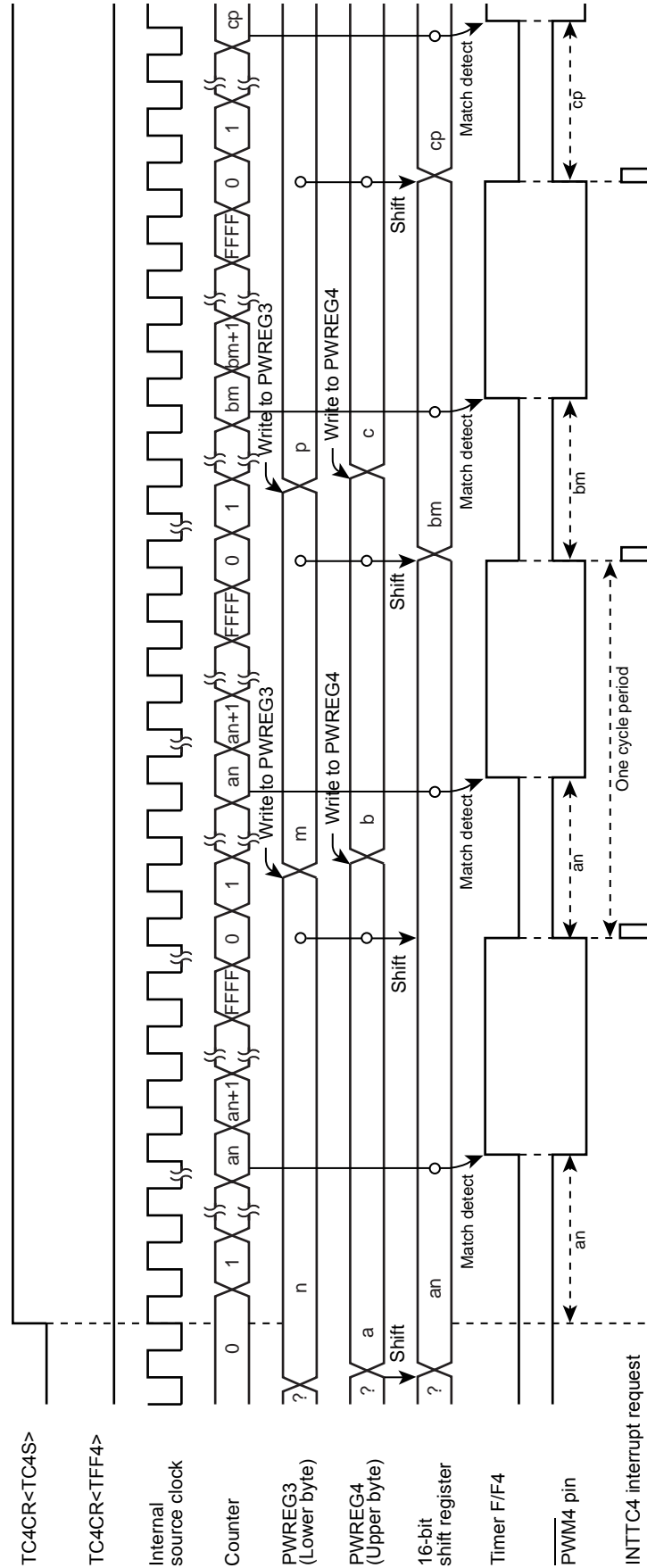
Figure 9-7 16-Bit PWM Mode Timing Chart (TC3 and TC4)

9.3.8 16-Bit Programmable Pulse Generate (PPG) Output Mode (TC3 and 4)

This mode is used to generate pulses with up to 16-bits of resolution. The timer counter 3 and 4 are cascadable to enter the 16-bit PPG mode.

The counter counts up using the internal clock or external clock. When a match between the up-counter and the timer register (PWREG3, PWREG4) value is detected, the logic level output from the timer F/F4 is switched to the opposite state. The counter continues counting. The logic level output from the timer F/F4 is switched to the opposite state again when a match between the up-counter and the timer register (TTREG3, TTREG4) value is detected, and the counter is cleared. The INTTC4 interrupt is generated at this time.

Two machine cycles are required for the high- or low-level pulse input to the TC3 pin. Therefore, a maximum frequency to be supplied is $fc/2^4$ Hz in the NORMAL1/2 or IDLE1/2 mode, and $fs/2^4$ to in the SLOW1/2 or SLEEP1/2 mode.

Since the initial value can be set to the timer F/F4 by TC4CR<TFF4>, positive and negative pulses can be generated. Upon reset, the timer F/F4 is cleared to 0.

(The logic level output from the $\overline{PPG}4$ pin is the opposite to the timer F/F4.)

Set the lower byte and upper byte in this order to program the timer register. (TTREG3 → TTREG4, PWREG3 → PWREG4) (Programming only the upper or lower byte should not be attempted.)

For PPG output, set the output latch of the I/O port to 1.

Example :Generating a pulse with 1-ms high-level width and a period of 16.385 ms (fc = 16.0 MHz)

Setting ports

LDW	(PWREG3), 07D0H	: Sets the pulse width.
LDW	(TTREG3), 8002H	: Sets the cycle period.
LD	(TC3CR), 33H	: Sets the operating clock to $fc/2^3$, and16-bit PPG mode (lower byte).
LD	(TC4CR), 057H	: Sets TFF4 to the initial value 0, and 16-bit PPG mode (upper byte).
LD	(TC4CR), 05FH	: Starts the timer.

Note 1: In the PPG mode, do not change the PWREGi and TTREGi settings while the timer is running. Since PWREGi and TTREGi are not in the shift register configuration in the PPG mode, the new values programmed in PWREGi and TTREGi are in effect immediately after programming PWREGi and TTREGi. Therefore, if PWREGi and TTREGi are changed while the timer is running, an expected operation may not be obtained.

Note 2: When the timer is stopped during PPG output, the $\overline{PPG}4$ pin holds the output status when the timer is stopped. To change the output status, program TC4CR<TFF4> after the timer is stopped. Do not change TC4CR<TFF4> upon stopping of the timer.
Example: Fixing the $\overline{PPG}4$ pin to the high level when the TimerCounter is stopped
CLR (TC4CR).3: Stops the timer
CLR (TC4CR).7: Sets the $\overline{PPG}4$ pin to the high level

Note 3: i = 3, 4
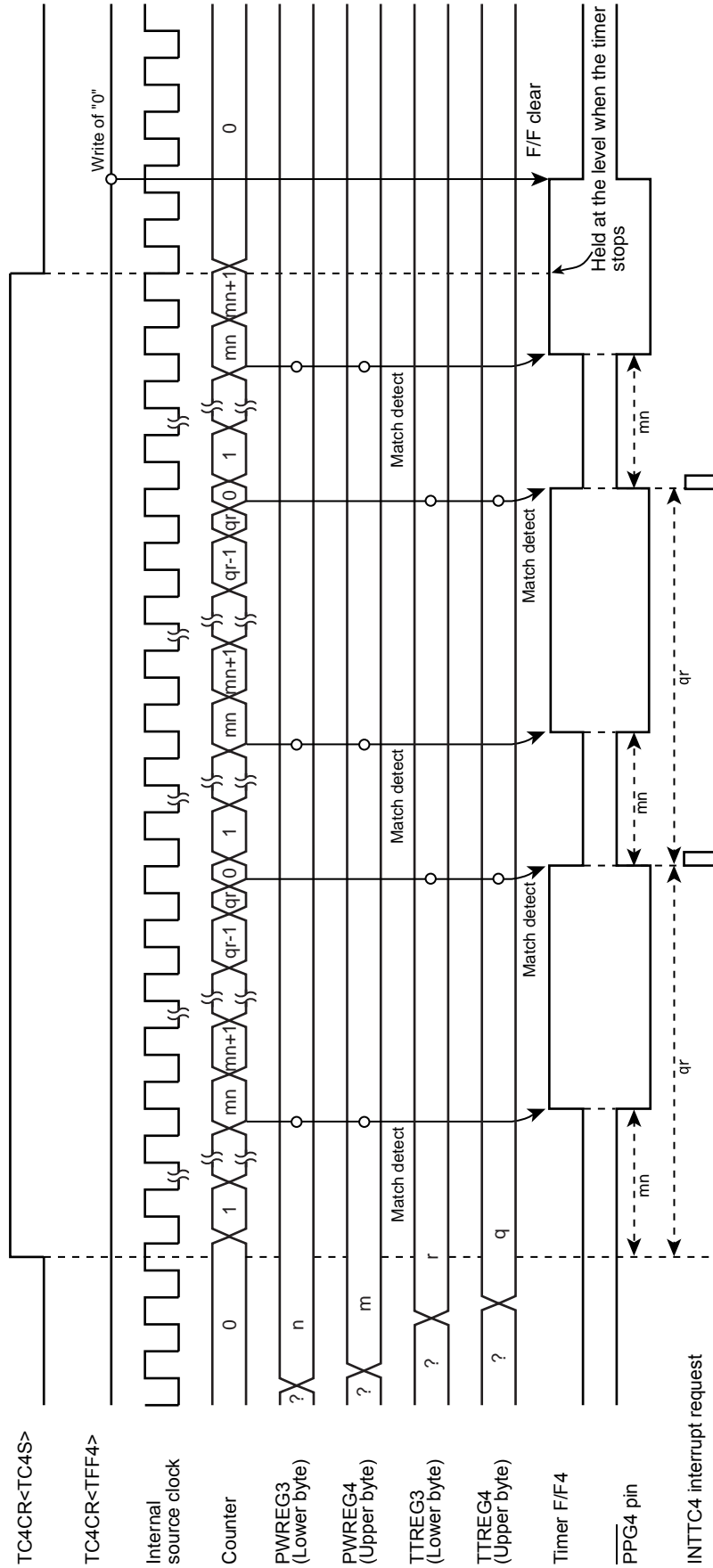
Figure 9-8 16-Bit PPG Mode Timing Chart (TC3 and TC4)

9.3.9 Warm-Up Counter Mode

In this mode, the warm-up period time is obtained to assure oscillation stability when the system clocking is switched between the high-frequency and low-frequency. The timer counter 3 and 4 are cascadable to form a 16-bit TimerCounter. The warm-up counter mode has two types of mode; switching from the high-frequency to low-frequency, and vice-versa.

Note 1: In the warm-up counter mode, fix TCiCR<TFFi> to 0. If not fixed, the $\overline{PDOi}$, $\overline{PWMi}$ and $\overline{PPGi}$ pins may output pulses.

Note 2: In the warm-up counter mode, only upper 8 bits of the timer register TTREG4 and 3 are used for match detection and lower 8 bits are not used.

Note 3: i = 3, 4

9.3.9.1 Low-Frequency Warm-up Counter Mode
(NORMAL1 $\rightarrow$ NORMAL2 $\rightarrow$ SLOW2 $\rightarrow$ SLOW1)

In this mode, the warm-up period time from a stop of the low-frequency clock fs to oscillation stability is obtained. Before starting the timer, set SYSCR2<XTEN> to 1 to oscillate the low-frequency clock. When a match between the up-counter and the timer register (TTREG4, 3) value is detected after the timer is started by setting TC4CR<TC4S> to 1, the counter is cleared by generating the INTTC4 interrupt request. After stopping the timer in the INTTC4 interrupt service routine, set SYSCR2<SYSCK> to 1 to switch the system clock from the high-frequency to low-frequency, and then clear of SYSCR2<XEN> to 0 to stop the high-frequency clock.

Table 9-8   Setting Time of Low-Frequency Warm-Up Counter Mode (fs = 32.768 kHz)

Minimum Time Setting (TTREG4, 3 = 0100H)	Maximum Time Setting (TTREG4, 3 = FF00H)
7.81 ms	1.99 s

Example :After checking low-frequency clock oscillation stability with TC4 and 3, switching to the SLOW1 mode

```
            SET        (SYSCR2).6        : SYSCR2<XTEN> ← 1

            LD         (TC3CR), 43H      : Sets TFF3=0, source clock fs, and 16-bit mode.

            LD         (TC4CR), 05H      : Sets TFF4=0, and warm-up counter mode.

            LD         (TTREG3), 8000H   : Sets the warm-up time.
                                            (The warm-up time depends on the oscillator characteristic.)

            DI                           : IMF ← 0

            SET        (EIRE). 1         : Enables the INTTC4.

            EI                           : IMF ← 1

            SET        (TC4CR).3         : Starts TC4 and 3.

              :          :
PINTTC4:    CLR        (TC4CR).3         : Stops TC4 and 3.

            SET        (SYSCR2).5        : SYSCR2<SYSCK> ← 1
                                            (Switches the system clock to the low-frequency clock.)

            CLR        (SYSCR2).7        : SYSCR2<XEN> ← 0 (Stops the high-frequency clock.)

            RETI

              :          :
VINTTC4:    DW         PINTTC4           : INTTC4 vector table
```

9.3.9.2 High-Frequency Warm-Up Counter Mode
(SLOW1 → SLOW2 → NORMAL2 → NORMAL1)

In this mode, the warm-up period time from a stop of the high-frequency clock fc to the oscillation stability is obtained. Before starting the timer, set SYSCR2<XEN> to 1 to oscillate the high-frequency clock. When a match between the up-counter and the timer register (TTREG4, 3) value is detected after the timer is started by setting TC4CR<TC4S> to 1, the counter is cleared by generating the INTTC4 interrupt request. After stopping the timer in the INTTC4 interrupt service routine, clear SYSCR2<SYSCK> to 0 to switch the system clock from the low-frequency to high-frequency, and then SYSCR2<XTEN> to 0 to stop the low-frequency clock.

Table 9-9   Setting Time in High-Frequency Warm-Up Counter Mode

Minimum time Setting (TTREG4, 3 = 0100H)	Maximum time Setting (TTREG4, 3 = FF00H)
16 μs	4.08 ms

Example :After checking high-frequency clock oscillation stability with TC4 and 3, switching to the NORMAL1 mode

```
            SET     (SYSCR2).7          : SYSCR2<XEN> ← 1

            LD      (TC3CR), 63H        : Sets TFF3=0, source clock fc, and 16-bit mode.

            LD      (TC4CR), 05H        : Sets TFF4=0, and warm-up counter mode.

            LD      (TTREG3), 0F800H    : Sets the warm-up time.
                                          (The warm-up time depends on the oscillator characteristic.)

            DI                          : IMF ← 0

            SET     (EIRE). 1           : Enables the INTTC4.

            EI                          : IMF ← 1

            SET     (TC4CR).3           : Starts the TC4 and 3.

             :        :

PINTTC4:    CLR     (TC4CR).3           : Stops the TC4 and 3.

            CLR     (SYSCR2).5          : SYSCR2<SYSCK> ← 0
                                          (Switches the system clock to the high-frequency clock.)

            CLR     (SYSCR2).6          : SYSCR2<XTEN> ← 0
                                          (Stops the low-frequency clock.)

            RETI

             :        :

VINTTC4:    DW      PINTTC4             : INTTC4 vector table
```

TOSHIBA

10. 8-Bit TimerCounter (TC5, TC6)

10.1 Configuration
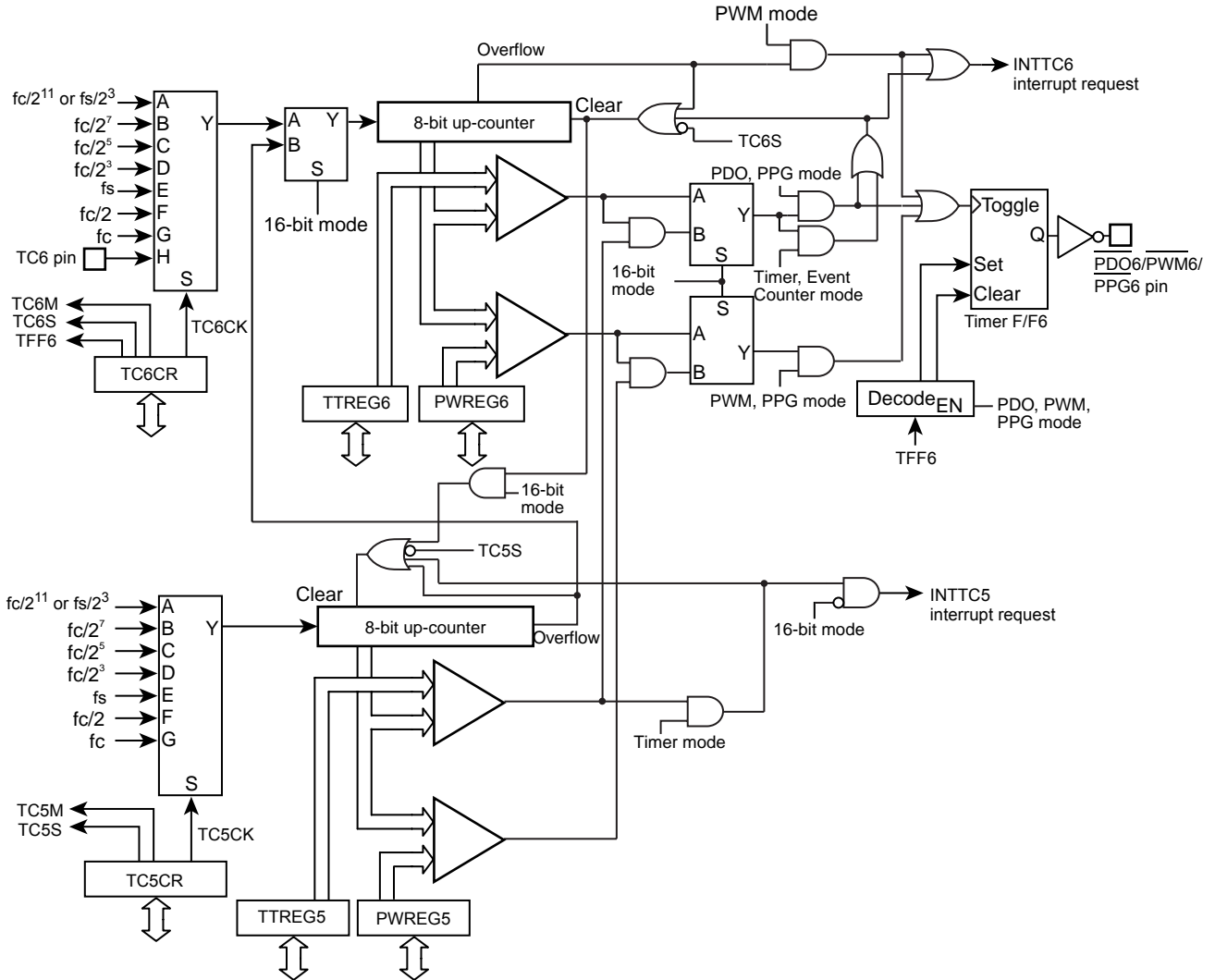
Figure 10-1 8-Bit TimerCounter 5, 6

10.2 TimerCounter Control

The TimerCounter 5 is controlled by the TimerCounter 5 control register (TC5CR) and two 8-bit timer registers (TTREG5, PWREG5).

TimerCounter 5 Timer Register

TTREG5 (001EH) R/W	7	6	5	4	3	2	1	0	(Initial value: 1111 1111)

PWREG5 (0022H) R/W	7	6	5	4	3	2	1	0	(Initial value: 1111 1111)
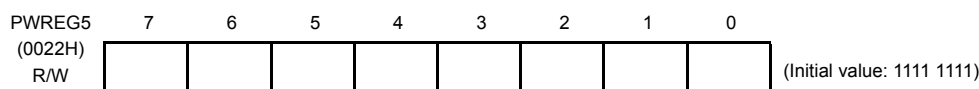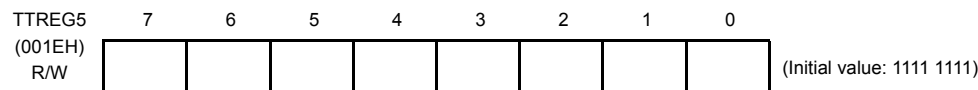
Note 1: Do not change the timer register (TTREG5) setting while the timer is running.

Note 2: Do not change the timer register (PWREG5) setting in the operating mode except the 8-bit and 16-bit PWM modes while the timer is running.

TimerCounter 5 Control Register

TC5CR (001AH)	7	6	5	4	3	2	1	0	(Initial value: *000 0000)
	-	TC5CK			TC5S	TC5M			

				NORMAL1/2, IDLE1/2 mode		SLOW1/2 SLEEP1/2 mode	
				DV7CK = 0	DV7CK = 1		
TC5CK	Operating clock selection [Hz]	000		$fc/2^{11}$	$fs/2^3$	$fs/2^3$	R/W
		001		$fc/2^7$	$fc/2^7$	–	
		010		$fc/2^5$	$fc/2^5$	–	
		011		$fc/2^3$	$fc/2^3$	–	
		100		fs	fs	fs	
		101		fc/2	fc/2	–	
		110		fc	fc	fc (Note 8)	
		111		Reserved			
TC5S	TC5 start control	0: Operation stop and counter clear 1: Operation start					R/W
TC5M	TC5M operating mode select	000: 8-bit timer 001: Reserved 010: Reserved 011: 16-bit mode (Each mode is selectable with TC6M.) 1**: Reserved					R/W

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

Note 2: Do not change the TC5M, TC5CK and TFF5 settings while the timer is running.

Note 3: To stop the timer operation (TC5S= 1 → 0), do not change the TC5M and TC5CK settings. To start the timer operation (TC5S= 0 → 1), TC5M and TC5CK can be programmed.

Note 4: To use the TimerCounter in the 16-bit mode, set the operating mode by programming TC6CR<TC6M>, where TC5M must be fixed to 011.

Note 5: To use the TimerCounter in the 16-bit mode, select the source clock by programming TC5CK. Set the timer start control and timer F/F control by programming TC6CR<TC6S> and TC6CR<TFF6>, respectively.

Note 6: The operating clock settings are limited depending on the timer operating mode. For the detailed descriptions, see Table 10-1 and Table 10-2.

Note 7: The timer register settings are limited depending on the timer operating mode. For the detailed descriptions, see Table 10-3.

Note 8: The operating clock fc in the SLOW or SLEEP mode can be used only as the high-frequency warm-up mode.

The TimerCounter 6 is controlled by the TimerCounter 6 control register (TC6CR) and two 8-bit timer registers (TTREG6 and PWREG6).

TimerCounter 6 Timer Register

TTREG6 (001FH) R/W	7	6	5	4	3	2	1	0	(Initial value: 1111 1111)

PWREG6 (0023H) R/W	7	6	5	4	3	2	1	0	(Initial value: 1111 1111)
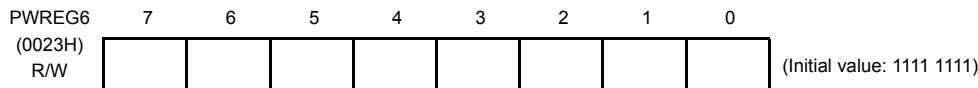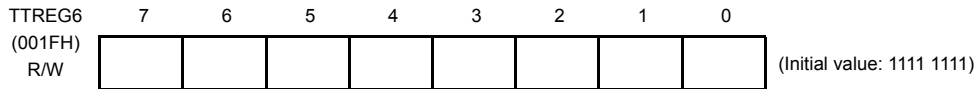
Note 1: Do not change the timer register (TTREG6) setting while the timer is running.

Note 2: Do not change the timer register (PWREG6) setting in the operating mode except the 8-bit and 16-bit PWM modes while the timer is running.

TimerCounter 6 Control Register

TC6CR (001BH)	7	6	5	4	3	2	1	0	(Initial value: 0000 0000)
	TFF6	TC6CK			TC6S	TC6M			
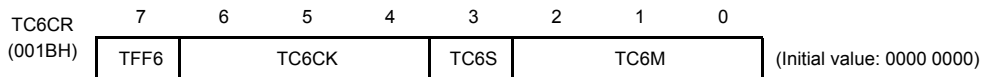
TFF6	Timer F/F6 control	0: Clear 1: Set				R/W
TC6CK	Operating clock selection [Hz]		NORMAL1/2, IDLE1/2 mode		SLOW1/2 SLEEP1/2 mode	R/W
			DV7CK = 0	DV7CK = 1		
		000	$fc/2^{11}$	$fs/2^3$	$fs/2^3$	
		001	$fc/2^7$	$fc/2^7$	–	
		010	$fc/2^5$	$fc/2^5$	–	
		011	$fc/2^3$	$fc/2^3$	–	
		100	fs	fs	fs	
		101	fc/2	fc/2	–	
		110	fc	fc	–	
		111	TC6 pin input			
TC6S	TC6 start control	0: Operation stop and counter clear 1: Operation start				R/W
TC6M	TC6M operating mode select	000: 8-bit timer/event counter mode 001: 8-bit programmable divider output (PDO) mode 010: 8-bit pulse width modulation (PWM) output mode 011: Reserved 100: 16-bit timer/event counter mode 101: Warm-up counter mode 110: 16-bit pulse width modulation (PWM) output mode 111: 16-bit PPG mode				R/W

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

Note 2: Do not change the TC6M, TC6CK and TFF6 settings while the timer is running.

Note 3: To stop the timer operation (TC6S= 1 → 0), do not change the TC6M, TC6CK and TFF6 settings.
To start the timer operation (TC6S= 0 → 1), TC6M, TC6CK and TFF6 can be programmed.

Note 4: When TC6M= 1** (upper byte in the 16-bit mode), the source clock becomes the TC5 overflow signal regardless of the TC6CK setting.

Note 5: To use the TimerCounter in the 16-bit mode, select the operating mode by programming TC6M, where TC5CR<TC5M> must be set to 011.

Note 6: To the TimerCounter in the 16-bit mode, select the source clock by programming TC5CR<TC5CK>. Set the timer start control and timer F/F control by programming TC6S and TFF6, respectively.

Note 7: The operating clock settings are limited depending on the timer operating mode. For the detailed descriptions, see Table 10-1 and Table 10-2.

Note 8: The timer register settings are limited depending on the timer operating mode. For the detailed descriptions, see Table 10-3.

Note 9: To use the PDO, PWM or PPG mode, a pulse is not output from the timer output pin when TC1CR2<TC6OUT> is set to 1. To output a pulse from the timer output pin, clear TC1CR2<TC6OUT> to 0.

Table 10-1 Operating Mode and Selectable Source Clock (NORMAL1/2 and IDLE1/2 Modes)

Operating mode	$fc/2^{11}$ or $fs/2^3$	$fc/2^7$	$fc/2^5$	$fc/2^3$	fs	fc/2	fc	TC5 pin input	TC6 pin input
8-bit timer	O	O	O	O	–	–	–	–	–
8-bit event counter	–	–	–	–	–	–	–	–	O
8-bit PDO	O	O	O	O	–	–	–	–	–
8-bit PWM	O	O	O	O	O	O	O	–	–
16-bit timer	O	O	O	O	–	–	–	–	–
Warm-up counter	–	–	–	–	O	–	–	–	–
16-bit PWM	O	O	O	O	O	O	O	–	–
16-bit PPG	O	O	O	O	–	–	–	–	–

Note 1: For 16-bit operations (16-bit timer, warm-up counter, 16-bit PWM and 16-bit PPG), set its source clock on lower bit (TC5CK).

Note 2: O : Available source clock

Table 10-2 Operating Mode and Selectable Source Clock (SLOW1/2 and SLEEP1/2 Modes)

Operating mode	$fc/2^{11}$ or $fs/2^3$	$fc/2^7$	$fc/2^5$	$fc/2^3$	fs	fc/2	fc	TC5 pin input	TC6 pin input
8-bit timer	O	–	–	–	–	–	–	–	–
8-bit event counter	–	–	–	–	–	–	–	–	O
8-bit PDO	O	–	–	–	–	–	–	–	–
8-bit PWM	O	–	–	–	O	–	–	–	–
16-bit timer	O	–	–	–	–	–	–	–	–
Warm-up counter	–	–	–	–	–	–	O	–	–
16-bit PWM	O	–	–	–	O	–	–	–	–
16-bit PPG	O	–	–	–	–	–	–	–	–

Note1: For 16-bit operations (16-bit timer, warm-up counter, 16-bit PWM and 16-bit PPG), set its source clock on lower bit (TC5CK).

Note2: O : Available source clock

Table 10-3 Constraints on Register Values Being Compared

Operating mode	Register Value
8-bit timer/event counter	$1 \leq$ (TTREGn) ≤ 255
8-bit PDO	$1 \leq$ (TTREGn) ≤ 255
8-bit PWM	$2 \leq$ (PWREGn) ≤ 254
16-bit timer	$1 \leq$ (TTREG6, 5) ≤ 65535
Warm-up counter	$256 \leq$ (TTREG6, 5) ≤ 65535
16-bit PWM	$2 \leq$ (PWREG6, 5) ≤ 65534
16-bit PPG	$1 \leq$ (PWREG6, 5) $<$ (TTREG6, 5) ≤ 65535 and (PWREG6, 5) + 1 $<$ (TTREG6, 5)

Note: n = 5 to 6

10.3 Function

The TimerCounter 6 have the 8-bit timer, 8-bit event counter, 8-bit programmable divider output (PDO), 8-bit pulse width modulation (PWM) output modes. The TimerCounter 5 and 6 (TC5, 6) are cascadable to form a 16-bit timer. The 16-bit timer has the operating modes such as the 16-bit timer, 16-bit event counter, warm-up counter, 16-bit pulse width modulation (PWM) output and 16-bit programmable pulse generation (PPG) modes.

10.3.1 8-Bit Timer Mode (TC5 and 6)

In the timer mode, the up-counter counts up using the internal clock. When a match between the up-counter and the timer register j (TTREGj) value is detected, an INTTCj interrupt is generated and the up-counter is cleared. After being cleared, the up-counter restarts counting.

Note 1: In the timer mode, fix TCjCR<TFFj> to 0. If not fixed, the $\overline{PDOj}$, $\overline{PWMj}$ and $\overline{PPGj}$ pins may output pulses.

Note 2: In the timer mode, do not change the TTREGj setting while the timer is running. Since TTREGj is not in the shift register configuration in the timer mode, the new value programmed in TTREGj is in effect immediately after the programming. Therefore, if TTREGi is changed while the timer is running, an expected operation may not be obtained.

Note 3: j = 5, 6

Table 10-4 Source Clock for TimerCounter 5, 6 (Internal Clock)

Source Clock			Resolution		Maximum Time Setting	
NORMAL1/2, IDLE1/2 mode		SLOW1/2, SLEEP1/2 mode	fc = 16 MHz	fs = 32.768 kHz	fc = 16 MHz	fs = 32.768 kHz
DV7CK = 0	DV7CK = 1					
$fc/2^{11}$ [Hz]	$fs/2^3$ [Hz]	$fs/2^3$ [Hz]	128 μs	244.14 μs	32.6 ms	62.3 ms
$fc/2^7$	$fc/2^7$	–	8 μs	–	2.0 ms	–
$fc/2^5$	$fc/2^5$	–	2 μs	–	510 μs	–
$fc/2^3$	$fc/2^3$	–	500 ns	–	127.5 μs	–

Example : Setting the timer mode with source clock $fc/2^7$ Hz and generating an interrupt 80 μs later
(TimerCounter6, fc = 16.0 MHz)

```
LD      (TTREG6), 0AH          : Sets the timer register (80 μs÷2⁷/fc = 0AH).
DI
SET     (EIRE). 3              : Enables INTTC6 interrupt.
EI
LD      (TC6CR), 00010000B     : Sets the operating clock to fc/2⁷, and 8-bit timer mode.
LD      (TC6CR), 00011000B     : Starts TC6.
```
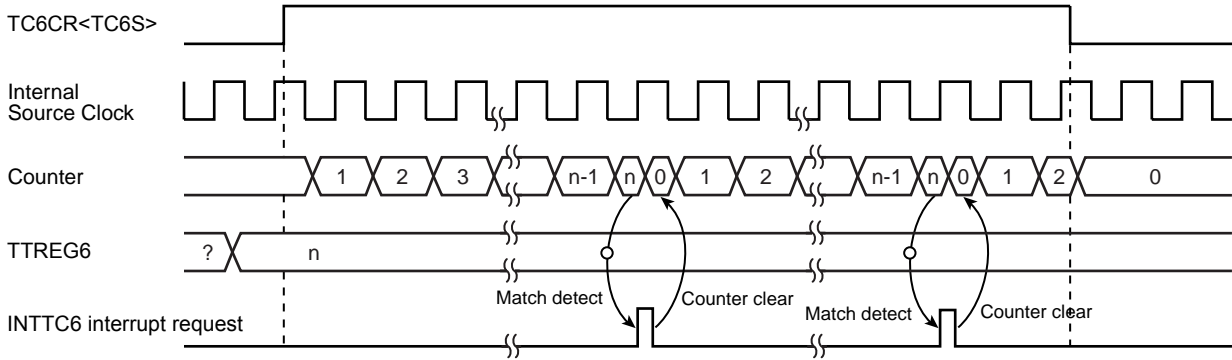
Figure 10-2 8-Bit Timer Mode Timing Chart (TC6)

10.3.2 8-Bit Event Counter Mode (TC6)

In the 8-bit event counter mode, the up-counter counts up at the falling edge of the input pulse to the TCj pin. When a match between the up-counter and the TTREGj value is detected, an INTTCj interrupt is generated and the up-counter is cleared. After being cleared, the up-counter restarts counting at the falling edge of the input pulse to the TCj pin. Two machine cycles are required for the low- or high-level pulse input to the TCj pin. Therefore, a maximum frequency to be supplied is $fc/2^4$ Hz in the NORMAL1/2 or IDLE1/2 mode, and $fs/2^4$ Hz in the SLOW1/2 or SLEEP1/2 mode.

Note 1: In the event counter mode, fix TCjCR<TFFj> to 0. If not fixed, the $\overline{PDOj}$, $\overline{PWMj}$ and $\overline{PPGj}$ pins may output pulses.

Note 2: In the event counter mode, do not change the TTREGj setting while the timer is running. Since TTREGj is not in the shift register configuration in the event counter mode, the new value programmed in TTREGj is in effect immediately after the programming. Therefore, if TTREGi is changed while the timer is running, an expected operation may not be obtained.

Note 3: j = 6

Figure 10-3 8-Bit Event Counter Mode Timing Chart (TC6)

10.3.3 8-Bit Programmable Divider Output (PDO) Mode (TC6)

This mode is used to generate a pulse with a 50% duty cycle from the $\overline{PDOj}$ pin.

In the PDO mode, the up-counter counts up using the internal clock. When a match between the up-counter and the TTREGj value is detected, the logic level output from the $\overline{PDOj}$ pin is switched to the opposite state and the up-counter is cleared. The INTTCj interrupt request is generated at the time. The logic state opposite to the timer F/Fj logic level is output from the $\overline{PDOj}$ pin. An arbitrary value can be set to the timer F/Fj by TCjCR<TFFj>. Upon reset, the timer F/Fj value is initialized to 0.

To use the programmable divider output, set the output latch of the I/O port to 1.

Example :Generating 1024 Hz pulse using TC6 (fc = 16.0 MHz)

Setting port

LD	(TTREG6), 3DH	: $1/1024 \div 2^7/\text{fc} \div 2 = 3DH$
LD	(TC6CR), 00010001B	: Sets the operating clock to $\text{fc}/2^7$, and 8-bit PDO mode.
LD	(TC6CR), 00011001B	: Starts TC6.

Note 1: In the programmable divider output mode, do not change the TTREGj setting while the timer is running. Since TTREGj is not in the shift register configuration in the programmable divider output mode, the new value programmed in TTREGj is in effect immediately after programming. Therefore, if TTREGi is changed while the timer is running, an expected operation may not be obtained.

Note 2: When the timer is stopped during PDO output, the $\overline{\text{PDOj}}$ pin holds the output status when the timer is stopped. To change the output status, program TCjCR<TFFj> after the timer is stopped. Do not change the TCjCR<TFFj> setting upon stopping of the timer.
Example: Fixing the $\overline{\text{PDOj}}$ pin to the high level when the TimerCounter is stopped
CLR (TCjCR).3: Stops the timer.
CLR (TCjCR).7: Sets the $\overline{\text{PDOj}}$ pin to the high level.

Note 3: j = 6

Figure 10-4 8-Bit PDO Mode Timing Chart (TC6)

10.3.4 8-Bit Pulse Width Modulation (PWM) Output Mode (TC6)

This mode is used to generate a pulse-width modulated (PWM) signals with up to 8 bits of resolution. The up-counter counts up using the internal clock.

When a match between the up-counter and the PWREGj value is detected, the logic level output from the timer F/Fj is switched to the opposite state. The counter continues counting. The logic level output from the timer F/Fj is switched to the opposite state again by the up-counter overflow, and the counter is cleared. The INTTCj interrupt request is generated at this time.

Since the initial value can be set to the timer F/Fj by TCjCR<TFFj>, positive and negative pulses can be generated. Upon reset, the timer F/Fj is cleared to 0.

(The logic level output from the $\overline{\text{PWMj}}$ pin is the opposite to the timer F/Fj logic level.)

Since PWREGj in the PWM mode is serially connected to the shift register, the value set to PWREGj can be changed while the timer is running. The value set to PWREGj during a run of the timer is shifted by the INTTCj interrupt request and loaded into PWREGj. While the timer is stopped, the value is shifted immediately after the programming of PWREGj. If executing the read instruction to PWREGj during PWM output, the value in the shift register is read, but not the value set in PWREGj. Therefore, after writing to PWREGj, the reading data of PWREGj is previous value until INTTCj is generated.

For the pin used for PWM output, the output latch of the I/O port must be set to 1.

Note 1: In the PWM mode, program the timer register PWREGj immediately after the INTTCj interrupt request is generated (normally in the INTTCj interrupt service routine.) If the programming of PWREGj and the interrupt request occur at the same time, an unstable value is shifted, that may result in generation of the pulse different from the programmed value until the next INTTCj interrupt request is generated.

Note 2: When the timer is stopped during PWM output, the $\overline{\text{PWMj}}$ pin holds the output status when the timer is stopped. To change the output status, program TCjCR<TFFj> after the timer is stopped. Do not change the TCjCR<TFFj> upon stopping of the timer.
Example: Fixing the $\overline{\text{PWMj}}$ pin to the high level when the TimerCounter is stopped
CLR (TCjCR).3: Stops the timer.
CLR (TCjCR).7: Sets the $\overline{\text{PWMj}}$ pin to the high level.

Note 3: To enter the STOP mode during PWM output, stop the timer and then enter the STOP mode. If the STOP mode is entered without stopping the timer when fc, fc/2 or fs is selected as the source clock, a pulse is output from the $\overline{\text{PWMj}}$ pin during the warm-up period time after exiting the STOP mode.

Note 4: j = 6

Table 10-5 PWM Output Mode

Source Clock			Resolution		Repeated Cycle	
NORMAL1/2, IDLE1/2 mode		SLOW1/2, SLEEP1/2 mode	fc = 16 MHz	fs = 32.768 kHz	fc = 16 MHz	fs = 32.768 kHz
DV7CK = 0	DV7CK = 1					
$fc/2^{11}$ [Hz]	$fs/2^3$ [Hz]	$fs/2^3$ [Hz]	128 μs	244.14 μs	32.8 ms	62.5 ms
$fc/2^7$	$fc/2^7$	–	8 μs	–	2.05 ms	–
$fc/2^5$	$fc/2^5$	–	2 μs	–	512 μs	–
$fc/2^3$	$fc/2^3$	–	500 ns	–	128 μs	–
fs	fs	fs	30.5 μs	30.5 μs	7.81 ms	7.81 ms
fc/2	fc/2	–	125 ns	–	32 μs	–
fc	fc	–	62.5 ns	–	16 μs	–

Figure 10-5 8-Bit PWM Mode Timing Chart (TC6)

10.3.5 16-Bit Timer Mode (TC5 and 6)

In the timer mode, the up-counter counts up using the internal clock. The TimerCounter 5 and 6 are cascadable to form a 16-bit timer.

When a match between the up-counter and the timer register (TTREG5, TTREG6) value is detected after the timer is started by setting TC6CR<TC6S> to 1, an INTTC6 interrupt is generated and the up-counter is cleared. After being cleared, the up-counter continues counting. Program the lower byte and upper byte in this order in the timer register. (Programming only the upper or lower byte should not be attempted.)

Note 1: In the timer mode, fix TCjCR<TFFj> to 0. If not fixed, the $\overline{PDOj}$, $\overline{PWMj}$, and $\overline{PPGj}$ pins may output a pulse.

Note 2: In the timer mode, do not change the TTREGj setting while the timer is running. Since TTREGj is not in the shift register configuration in the timer mode, the new value programmed in TTREGj is in effect immediately after programming of TTREGj. Therefore, if TTREGj is changed while the timer is running, an expected operation may not be obtained.

Note 3: j = 5, 6

Table 10-6 Source Clock for 16-Bit Timer Mode

Source Clock			Resolution		Maximum Time Setting	
NORMAL1/2, IDLE1/2 mode		SLOW1/2, SLEEP1/2 mode	fc = 16 MHz	fs = 32.768 kHz	fc = 16 MHz	fs = 32.768 kHz
DV7CK = 0	DV7CK = 1					
$fc/2^{11}$	$fs/2^3$	$fs/2^3$	128 μs	244.14 μs	8.39 s	16 s
$fc/2^7$	$fc/2^7$	–	8 μs	–	524.3 ms	–
$fc/2^5$	$fc/2^5$	–	2 μs	–	131.1 ms	–
$fc/2^3$	$fc/2^3$	–	500 ns	–	32.8 ms	–

Example : Setting the timer mode with source clock $fc/2^7$ Hz, and generating an interrupt 300 ms later
(fc = 16.0 MHz)

LDW	(TTREG5), 927CH	: Sets the timer register (300 ms÷2^7/fc = 927CH).
DI		
SET	(EIRE). 3	: Enables INTTC6 interrupt.
EI		
LD	(TC5CR), 13H	:Sets the operating clock to $fc/2^7$, and 16-bit timer mode (lower byte).
LD	(TC6CR), 04H	: Sets the 16-bit timer mode (upper byte).
LD	(TC6CR), 0CH	: Starts the timer.
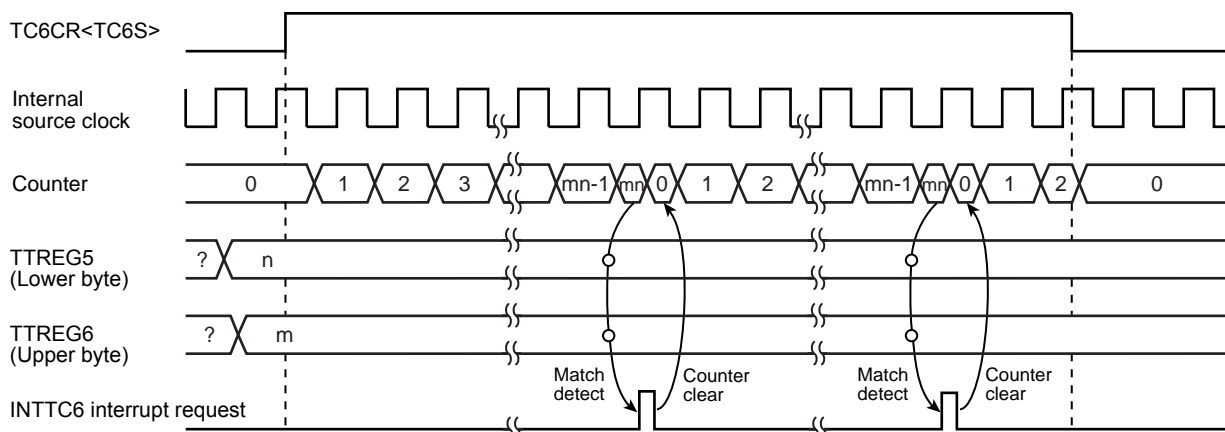
Figure 10-6 16-Bit Timer Mode Timing Chart (TC5 and TC6)

10.3.6 16-Bit Pulse Width Modulation (PWM) Output Mode (TC5 and 6)

This mode is used to generate a pulse-width modulated (PWM) signals with up to 16 bits of resolution. The TimerCounter 5 and 6 are cascadable to form the 16-bit PWM signal generator.

The counter counts up using the internal clock.

When a match between the up-counter and the timer register (PWREG5, PWREG6) value is detected, the logic level output from the timer F/F6 is switched to the opposite state. The counter continues counting. The logic level output from the timer F/F6 is switched to the opposite state again by the counter overflow, and the counter is cleared. The INTTC6 interrupt is generated at this time.

Two machine cycles are required for the high- or low-level pulse input to the TC5 pin. Therefore, a maximum frequency to be supplied is $fc/2^4$ Hz in the NORMAL1/2 or IDLE1/2 mode, and $fs/2^4$ to in the SLOW1/2 or SLEEP1/2 mode.

Since the initial value can be set to the timer F/F6 by TC6CR<TFF6>, positive and negative pulses can be generated. Upon reset, the timer F/F6 is cleared to 0.

(The logic level output from the $\overline{\text{PWM}}$6 pin is the opposite to the timer F/F6 logic level.)

Since PWREG6 and 5 in the PWM mode are serially connected to the shift register, the values set to PWREG6 and 5 can be changed while the timer is running. The values set to PWREG6 and 5 during a run of the timer are shifted by the INTTCj interrupt request and loaded into PWREG6 and 5. While the timer is stopped, the values are shifted immediately after the programming of PWREG6 and 5. Set the lower byte (PWREG5) and upper byte (PWREG6) in this order to program PWREG6 and 5. (Programming only the lower or upper byte of the register should not be attempted.)

If executing the read instruction to PWREG6 and 5 during PWM output, the values set in the shift register is read, but not the values set in PWREG6 and 5. Therefore, after writing to the PWREG6 and 5, reading data of PWREG6 and 5 is previous value until INTTC6 is generated.

For the pin used for PWM output, the output latch of the I/O port must be set to 1.

Note 1: In the PWM mode, program the timer register PWREG6 and 5 immediately after the INTTC6 interrupt request is generated (normally in the INTTC6 interrupt service routine.) If the programming of PWREGj and the interrupt request occur at the same time, an unstable value is shifted, that may result in generation of pulse different from the programmed value until the next INTTC6 interrupt request is generated.

Note 2: When the timer is stopped during PWM output, the $\overline{\text{PWM}}$6 pin holds the output status when the timer is stopped. To change the output status, program TC6CR<TFF6> after the timer is stopped. Do not program TC6CR<TFF6> upon stopping of the timer.
Example: Fixing the $\overline{\text{PWM}}$6 pin to the high level when the TimerCounter is stopped
CLR (TC6CR).3: Stops the timer.
CLR (TC6CR).7 : Sets the $\overline{\text{PWM}}$6 pin to the high level.

Note 3: To enter the STOP mode, stop the timer and then enter the STOP mode. If the STOP mode is entered without stopping of the timer when fc, fc/2 or fs is selected as the source clock, a pulse is output from the $\overline{\text{PWM}}$6 pin during the warm-up period time after exiting the STOP mode.

Table 10-7 16-Bit PWM Output Mode

Source Clock			Resolution		Repeated Cycle	
NORMAL1/2, IDLE1/2 mode		SLOW1/2, SLEEP1/2 mode	fc = 16 MHz	fs = 32.768 kHz	fc = 16 MHz	fs = 32.768 kHz
DV7CK = 0	DV7CK = 1					
$fc/2^{11}$	$fs/2^3$ [Hz]	$fs/2^3$ [Hz]	128 μs	244.14 μs	8.39 s	16 s
$fc/2^7$	$fc/2^7$	–	8 μs	–	524.3 ms	–
$fc/2^5$	$fc/2^5$	–	2 μs	–	131.1 ms	–
$fc/2^3$	$fc/2^3$	–	500 ns	–	32.8 ms	–
fs	fs	fs	30.5 μs	30.5 μs	2 s	2 s
fc/2	fc/2	–	125 ns	–	8.2 ms	–
fc	fc	–	62.5 ns	–	4.1 ms	–

Example :Generating a pulse with 1-ms high-level width and a period of 32.768 ms (fc = 16.0 MHz)

	Setting ports	
LDW	(PWREG5), 07D0H	: Sets the pulse width.
LD	(TC5CR), 33H	: Sets the operating clock to fc/2^3, and 16-bit PWM output mode (lower byte).
LD	(TC6CR), 056H	: Sets TFF6 to the initial value 0, and 16-bit PWM signal generation mode (upper byte).
LD	(TC6CR), 05EH	: Starts the timer.
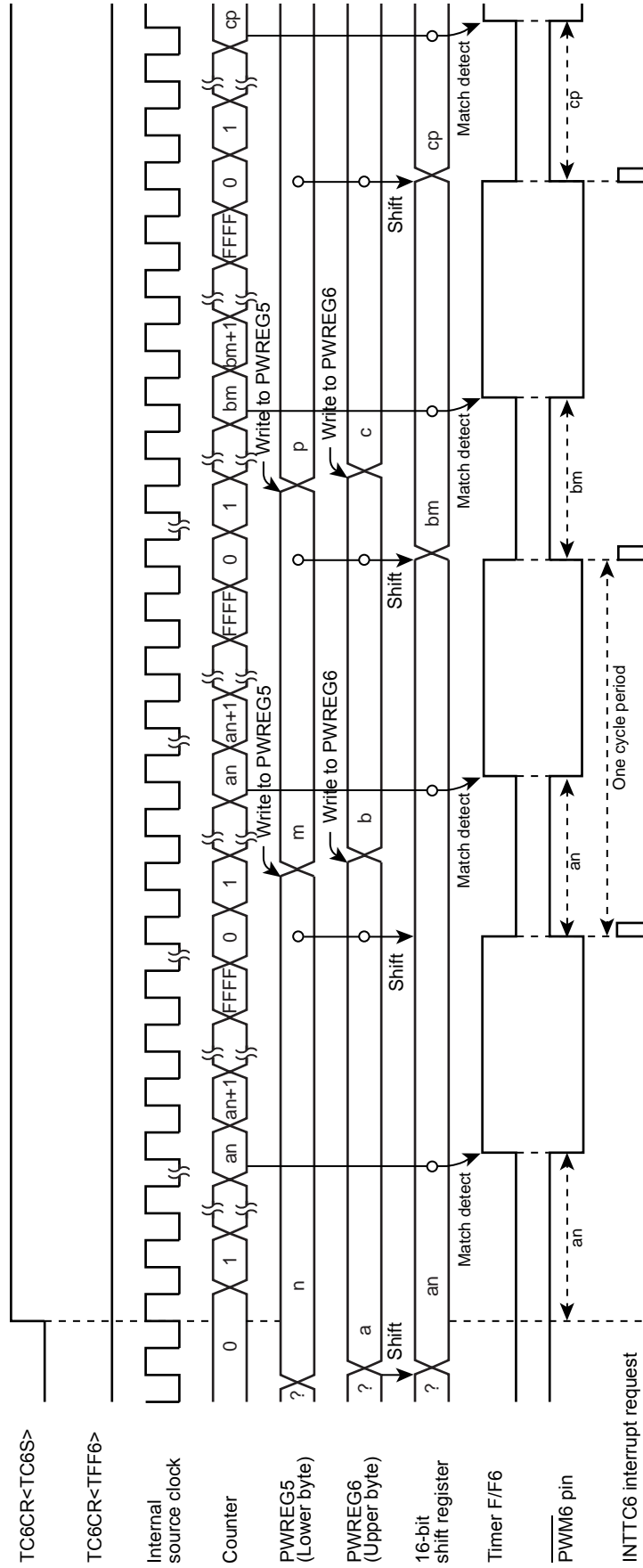
Figure 10-7 16-Bit PWM Mode Timing Chart (TC5 and TC6)

10.3.7 16-Bit Programmable Pulse Generate (PPG) Output Mode (TC5 and 6)

This mode is used to generate pulses with up to 16-bits of resolution. The timer counter 5 and 6 are cascadable to enter the 16-bit PPG mode.

The counter counts up using the internal clock. When a match between the up-counter and the timer register (PWREG5, PWREG6) value is detected, the logic level output from the timer F/F6 is switched to the opposite state. The counter continues counting. The logic level output from the timer F/F6 is switched to the opposite state again when a match between the up-counter and the timer register (TTREG5, TTREG6) value is detected, and the counter is cleared. The INTTC6 interrupt is generated at this time.

Since the initial value can be set to the timer F/F6 by TC6CR<TFF6>, positive and negative pulses can be generated. Upon reset, the timer F/F6 is cleared to 0.

(The logic level output from the $\overline{PPG6}$ pin is the opposite to the timer F/F6.)

Set the lower byte and upper byte in this order to program the timer register. (TTREG5 $\rightarrow$ TTREG6, PWREG5 $\rightarrow$ PWREG6) (Programming only the upper or lower byte should not be attempted.)

For PPG output, set the output latch of the I/O port to 1.

Example : Generating a pulse with 1-ms high-level width and a period of 16.385 ms (fc = 16.0 MHz)

	Setting ports	
LDW	(PWREG5), 07D0H	: Sets the pulse width.
LDW	(TTREG5), 8002H	: Sets the cycle period.
LD	(TC5CR), 33H	: Sets the operating clock to $fc/2^3$, and 16-bit PPG mode (lower byte).
LD	(TC6CR), 057H	: Sets TFF6 to the initial value 0, and 16-bit PPG mode (upper byte).
LD	(TC6CR), 05FH	: Starts the timer.

Note 1: In the PPG mode, do not change the PWREGi and TTREGi settings while the timer is running. Since PWREGi and TTREGi are not in the shift register configuration in the PPG mode, the new values programmed in PWREGi and TTREGi are in effect immediately after programming PWREGi and TTREGi. Therefore, if PWREGi and TTREGi are changed while the timer is running, an expected operation may not be obtained.

Note 2: When the timer is stopped during PPG output, the $\overline{PPG6}$ pin holds the output status when the timer is stopped. To change the output status, program TC6CR<TFF6> after the timer is stopped. Do not change TC6CR<TFF6> upon stopping of the timer.
Example: Fixing the $\overline{PPG6}$ pin to the high level when the TimerCounter is stopped
CLR (TC6CR).3: Stops the timer
CLR (TC6CR).7: Sets the $\overline{PPG6}$ pin to the high level

Note 3: i = 5, 6
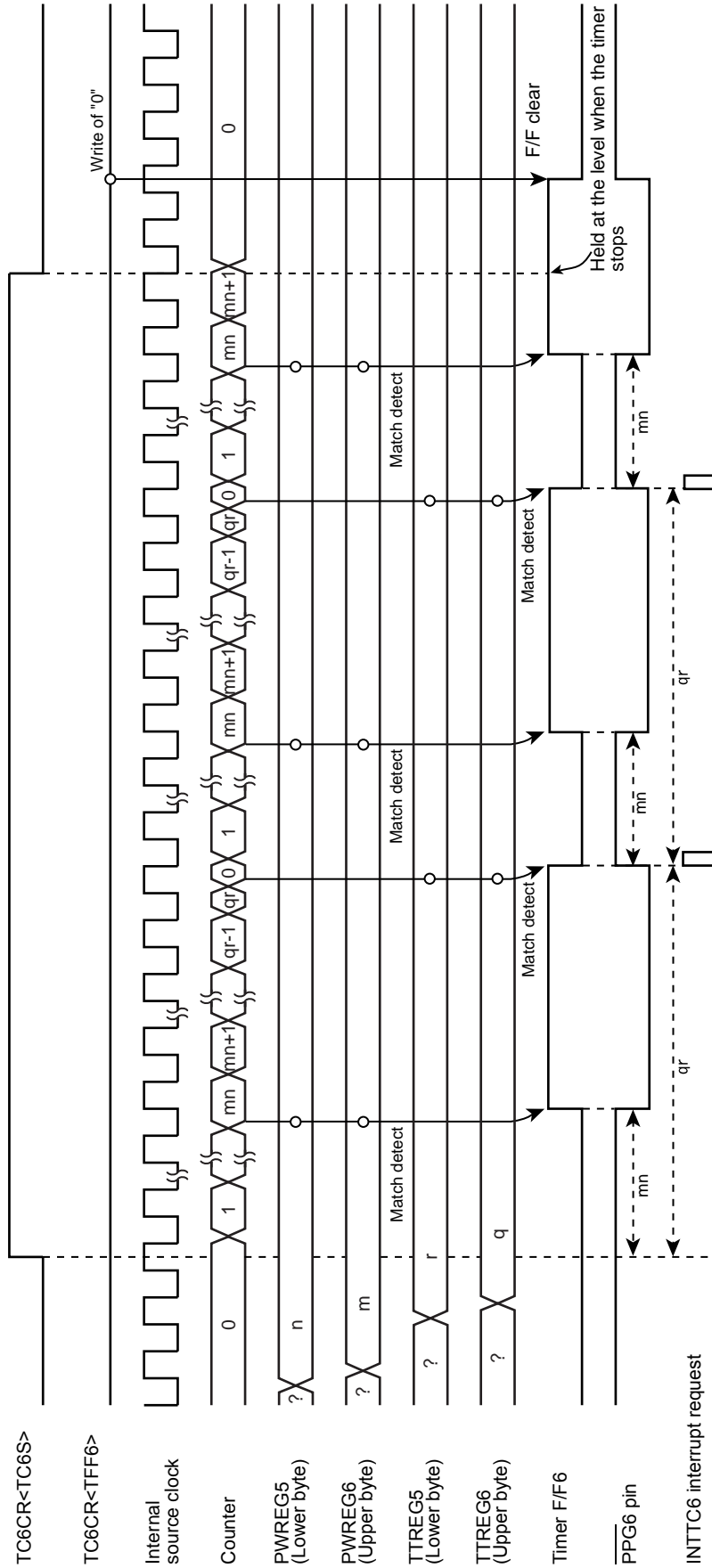
Figure 10-8   16-Bit PPG Mode Timing Chart (TC5 and TC6)

10.3.8 Warm-Up Counter Mode

In this mode, the warm-up period time is obtained to assure oscillation stability when the system clocking is switched between the high-frequency and low-frequency. The timer counter 5 and 6 are cascadable to form a 16-bit TimerCounter. The warm-up counter mode has two types of mode; switching from the high-frequency to low-frequency, and vice-versa.

Note 1: In the warm-up counter mode, fix TCiCR<TFFi> to 0. If not fixed, the $\overline{PDOi}$, $\overline{PWMi}$ and $\overline{PPGi}$ pins may output pulses.

Note 2: In the warm-up counter mode, only upper 8 bits of the timer register TTREG6 and 5 are used for match detection and lower 8 bits are not used.

Note 3: i = 5, 6

10.3.8.1 Low-Frequency Warm-up Counter Mode (NORMAL1 → NORMAL2 → SLOW2 → SLOW1)

In this mode, the warm-up period time from a stop of the low-frequency clock fs to oscillation stability is obtained. Before starting the timer, set SYSCR2<XTEN> to 1 to oscillate the low-frequency clock. When a match between the up-counter and the timer register (TTREG6, 5) value is detected after the timer is started by setting TC6CR<TC6S> to 1, the counter is cleared by generating the INTTC6 interrupt request. After stopping the timer in the INTTC6 interrupt service routine, set SYSCR2<SYSCK> to 1 to switch the system clock from the high-frequency to low-frequency, and then clear of SYSCR2<XEN> to 0 to stop the high-frequency clock.

Table 10-8 Setting Time of Low-Frequency Warm-Up Counter Mode (fs = 32.768 kHz)

Minimum Time Setting (TTREG6, 5 = 0100H)	Maximum Time Setting (TTREG6, 5 = FF00H)
7.81 ms	1.99 s

Example : After checking low-frequency clock oscillation stability with TC6 and 5, switching to the SLOW1 mode

	SET	(SYSCR2).6	: SYSCR2<XTEN> ← 1
	LD	(TC5CR), 43H	: Sets TFF5=0, source clock fs, and 16-bit mode.
	LD	(TC6CR), 05H	: Sets TFF6=0, and warm-up counter mode.
	LD	(TTREG5), 8000H	: Sets the warm-up time. (The warm-up time depends on the oscillator characteristic.)
	DI		: IMF ← 0
	SET	(EIRE). 3	: Enables the INTTC6.
	EI		: IMF ← 1
	SET	(TC6CR).3	: Starts TC6 and 5.
	:	:	
PINTTC6:	CLR	(TC6CR).3	: Stops TC6 and 5.
	SET	(SYSCR2).5	: SYSCR2<SYSCK> ← 1 (Switches the system clock to the low-frequency clock.)
	CLR	(SYSCR2).7	: SYSCR2<XEN> ← 0 (Stops the high-frequency clock.)
	RETI		
	:	:	
VINTTC6:	DW	PINTTC6	: INTTC6 vector table

10.3.8.2 High-Frequency Warm-Up Counter Mode
(SLOW1 → SLOW2 → NORMAL2 → NORMAL1)

In this mode, the warm-up period time from a stop of the high-frequency clock fc to the oscillation stability is obtained. Before starting the timer, set SYSCR2<XEN> to 1 to oscillate the high-frequency clock. When a match between the up-counter and the timer register (TTREG6, 5) value is detected after the timer is started by setting TC6CR<TC6S> to 1, the counter is cleared by generating the INTTC6 interrupt request. After stopping the timer in the INTTC6 interrupt service routine, clear SYSCR2<SYSCK> to 0 to switch the system clock from the low-frequency to high-frequency, and then SYSCR2<XTEN> to 0 to stop the low-frequency clock.

Table 10-9 Setting Time in High-Frequency Warm-Up Counter Mode

Minimum time Setting (TTREG6, 5 = 0100H)	Maximum time Setting (TTREG6, 5 = FF00H)
16 μs	4.08 ms

Example : After checking high-frequency clock oscillation stability with TC6 and 5, switching to the NORMAL1 mode

	SET	(SYSCR2).7	: SYSCR2<XEN> ← 1
	LD	(TC5CR), 63H	: Sets TFF5=0, source clock fc, and 16-bit mode.
	LD	(TC6CR), 05H	: Sets TFF6=0, and warm-up counter mode.
	LD	(TTREG5), 0F800H	: Sets the warm-up time. (The warm-up time depends on the oscillator characteristic.)
	DI		: IMF ← 0
	SET	(EIRE). 3	: Enables the INTTC6.
	EI		: IMF ← 1
	SET	(TC6CR).3	: Starts the TC6 and 5.
	:	:	
PINTTC6:	CLR	(TC6CR).3	: Stops the TC6 and 5.
	CLR	(SYSCR2).5	: SYSCR2<SYSCK> ← 0 (Switches the system clock to the high-frequency clock.)
	CLR	(SYSCR2).6	: SYSCR2<XTEN> ← 0 (Stops the low-frequency clock.)
	RETI		
	:	:	
VINTTC6:	DW	PINTTC6	: INTTC6 vector table

11. Real-Time Clock (RTC)

The TMP86FM26UG incorporates a real-time clock (RTC) with the auto calendar function. Clock and calendar counting can be performed using a high-frequency or low-frequency clock as the reference clock. The RTC of the TMP86FM26UG has the following features.

Note: Interrupt latch IL20 (bit 4 of ILE register) may be set to "1" while Real-Time Clock is operated. Therefore, the interrupt enable register EF20 (bit 4 of EIRE register) should be always set to "0" to prevent the unexpected interrupt request. Also, if ILE register is read, value of bit 4 should be not used by program.

- Clock (hours, minutes, seconds) and calendar (leap year, year, month, day, day of the week) counting function (BCD code)

- 2 channels of 20-bit counters
  The RTC comprises RTC counter 1 (high-frequency or low-frequency) and RTC counter 2 (low-frequency only).

- 1Hz pulse output function
  The RTCOUT pin can output pulses equivalent to 1Hz.

- Software capture of 20-bit RTC counters (RTC counters 1, 2)
  Each RTC counter value can be read while the RTC is running. When the system is operating in the dual-clock mode and the low-frequency clock is used as the source clock, RTC counter 2 can be used for detecting a stop condition in the low-frequency oscillator (deadlock detection).

- Leap year function
  Leap years are realized by using a leap year register (which counts from 0 to 3) independent of the year counter.

- Clock error correction function
  A 20-bit register is provided to correct the number of clock counts per second. This function makes it possible to reduce clock errors caused by the variations inherent in each external resonator or the effects of load capacitance. Error correction can be performed on a basis of 1 ppm for high-frequency and 30.5 ppm for low-frequency.

- Clock error measurement function
  In the error measurement mode, a 20-bit timer counter can be run in the free run mode using the precise reference clock (on a second basis) input to the RTCIN pin as the window pulse. By using this function, the value to be set in the compare register can easily be obtained. The output of RTC counter 1 based on the high-frequency source clock can be fed back to the input of RTC counter 2. This allows the user to adjust RTC counter 1 (high-frequency error) in one second and then to adjust RTC counter 2 (low-frequency error) on a stand-alone basis using 16 seconds in the pre-shipment adjustment process.

- A wide range of operating frequencies
  Because the number of clock counts per second is adjustable, 30.000 kHz to 34.000 kHz can be used as the low-frequency clock and 1 MHz to 16 MHz can be used as the high-frequency clock.

- Clock/calendar value preserving function
  The RTC counters and clock/calendar counters are not initialized by external or internal reset and the count values can be preserved. (The RTC is stopped by reset.)

11.1 Configuration
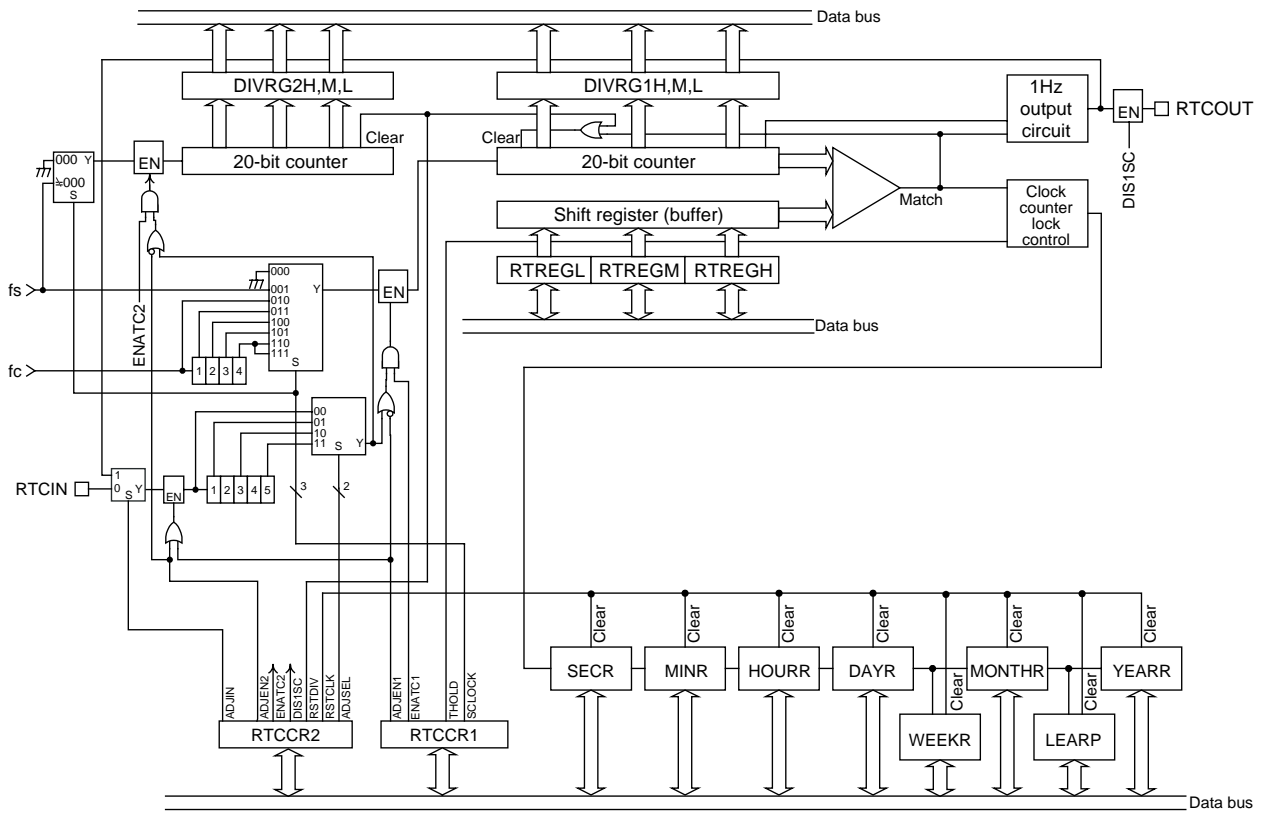
Figure 11-1   RTC Block Diagram

11.2 Control

The RTC is controlled by the RTC control register 1 (RTCCR1), RTC control register 2 (RTCCR2), RTC status register (RTCSR), clock counter registers (SECR, MINR, HOURR, DAYR, MONTHR, YEARR, WEEKR, LEAPR), RTC counter compare register (RTREG1L/M/H), and RTC counter 1/2 monitor registers (DIVRG1L/M/H, DIVRG2L/M/H).

RTC Control Register 1

RTCCR1 (0FA0H)	7	6	5	4	3	2	1	0	Read/Write
	ENATC1	ADJEN1	-	-	SCLOCK			THOLD	(Initial value: 0000 0000)

ENATC1	RTC counter 1 start control	0: Timer stop 1: Timer start (The source clock must be selected by SCLOCK before starting the timer.)	
ADJEN1	RTC counter 1 operation mode control	0: Clock mode 1: Error measurement mode	R/W
SCLOCK	RTC counter source clock control [Hz]		
			RTC counter 1	RTC counter 2
		000:	Stop	Stop
		001:	fs	fs
		010:	fc	
		011:	fc/2	
		100:	fc/4	
		101:	fc/8	
		110:	fc/16	
		111:		
THOLD	Clock counter lock control	0: Count up by a match of RTC counter 1 1: Do not count up by a match of RTC counter 1	

RTC Control Register 2

RTCCR2 (0FA1H)	7 ENATC2	6 ADJEN2	5 RSTDIV	4 RST-CLK	3 DIS1SC	2 ADJIN	1	0 ADJSEL	Read/Write (Initial value: 0000 0000)

ENATC2	RTC counter 2 start control	0: Timer stop 1: Timer start (The source clock must be selected by SCLOCK before starting the timer.)	
ADJEN2	RTC counter 2 operation mode control	0: Free run mode 1: Error measurement mode	
RSTDIV	RTC counter 1/2 initialization control	0: – 1: Initialize RTC counters 1, 2 (This bit is automatically cleared to "0" after being set to "1".)	
RSTCLK	Clock counter registers initialization control	0: – 1: Initialize the clock counter registers (This bit is automatically cleared to "0" after being set to "1".)	R/W
DIS1SC	RTCOUT pin output control	0: No output 1: Output pulse equivalent to 1 second from RTCOUT pin	
ADJIN	Reference clock input control	0: Input external input pulse on RTCIN pin 1: Input internal output pulse on RTCOUT pin	
ADJSEL	Window pulse width select	00: H level period of reference clock (Reference clock = Window pulse) 01: Reference clock divided by one (1 clock) 10: Reference clock divided by three (4 clocks) 11: Reference clock divided by five (16 clocks)	

Note 1: RTCCR2<RSTDIV, RSTCLK> are automatically cleared to "0" after being set to "1".

Note 2: Before writing to RTCCR1<CYCINT, SCLOCK>, make sure that the RTC has stopped completely. This state can be confirmed by RTCSR<RTCF1, RTCF2> = "0" after setting RTCCR1<ENATC1> and RTCCR2<ENATC2> to "0".

Note 3: Before writing to or reading from the clock counter registers, set RTCCR1<THOLD> to "1" and then check that RTCSR<TLOCK> has become "1". After writing to or reading from the clock counter registers, set RTCCR1<THOLD> to "0" again. If the clock counter registers are written while RTCCR1<THOLD> = "0", written values will not be reflected.

Note 4: Changing RTCCR1<THOLD> from "1" to "0" should be done while RTCSR<TLOCK> = "1".

Note 5: To start the error measurement mode, first set RTCCR2<RSTDIV, RSTCLK, ADJIN> to "1" and then set RTCCR1<ENATC1, ADJEN1> simultaneously to "1" (RTCCR2<ENATC2, ADJEN2> in the case of RTC counter 2). If these steps are not observed, measurement cannot be performed correctly.

Note 6: When RTCCR1<ENATC1> and RTCCR2<ENATC2> are "1", RTC counters 1 and 2 stop counting if break processing is performed with a debugger. After the break is released, each counter will resume counting from where it stopped.

RTC Status Register

RTCSR (0FA2H)	7	6	5	4	3	2	1	0	Read Only
				ADJF2	ADJF1	RTCF2	RTCF1	TLOCK	(Initial value: ***0 0000)

ADJF2	RTC counter 2 operation mode monitor	0:	Free run mode active		Read only
		1:	Error measurement mode active		
ADJF1	RTC counter 1 operation mode monitor	0:	Clock mode active		
		1:	Error measurement mode active		
RTCF2	RTC counter 2 operation status monitor		Free run mode	Error measurement mode	
		0:	RTC counter 2 inactive	Measuremnt completed	
		1:	RTC counter 2 active	Before or during measurement	
RTCF1	RTC counter 1 operation status monitor		Clock mode	Error measurement mode	
		0:	RTC counter 1 inactive	Measuremnt completed	
		1:	RTC counter 1 active	Before or during measurement	
TLOCK	Clock counter lock status monitor	0:	Time counter count up enabled		
		1:	Time counter count up disabled (Time counter register read/write enabled)		

Note 1: The value set in RTCCR2<ADJEN2> is reflected in RTCSR<ADJF2> after 4/fs [s] at the maximum.

Note 2: The value set in RTCCR1<ADJEN1> is reflected in RTCSR<ADJF1> after 4/n [s] (n = clock selected by RTCCR1<SCLOCK>) at the maximum.

Note 3: The value set in RTCCR2<ENATC2> is reflected in RTCSR<RTCF2> after 4/fs [s] at the maximum.

Note 4: The value set in RTCCR1<ENATC1> is reflected in RTCSR<RTCF1> after 4/n [s] (n = clock selected by RTCCR1<SCLOCK>) at the maximum.

Note 5: The value set in RTCCR1<THOLD> is reflected in RTCSR<TLOCK> after 4/n [s] (n = clock selected by RTCCR1<SCLOCK>) at the maximum.

Note 6: When RTCCR1<ENATC1, ADJEN1> and RTCCR2<ENATC2, ADJEN2> are changed from "1" to "0", do not change the values in RTCCR1<CYCINT, SCLOCK, THOLD>, RTCCR2<RSTDIV, RSTCLK, DIS1SC, ADJIN, ADJSEL> and the clock registers until RTCSR<RTCF1,2> and <ADJF1,2> become "0".

RTC Counter 1 Monitor Register

DIVRG1L	7	6	5	4	3	2	1	0	Read Only
(0FA6H)									

DIVRG1M	7	6	5	4	3	2	1	0	Read Only
(0FA7H)									

DIVRG1H	7	6	5	4	3	2	1	0	Read Only
(0FA8H)									
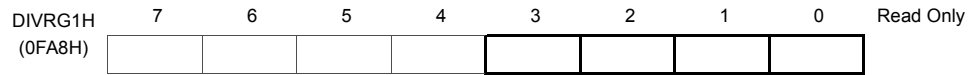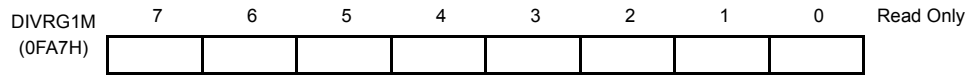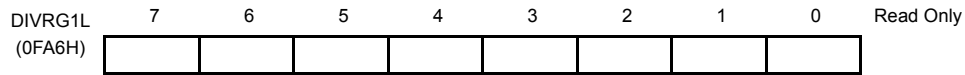
Note: When DIVRG1H is read, bits 7 to 4 are read as "0".

RTC Counter 1 Compare Register

RTREG1L	7	6	5	4	3	2	1	0	Read/Write
(0FA3H)									(Initial value: 1111 1111)

RTREG1M	7	6	5	4	3	2	1	0	Read/Write
(0FA4H)									(Initial value: 1111 1111)

RTREG1H	7	6	5	4	3	2	1	0	Read/Write
(0FA5H)									(Initial value: 0000 1111)
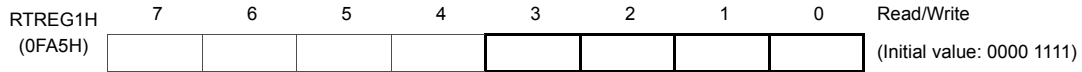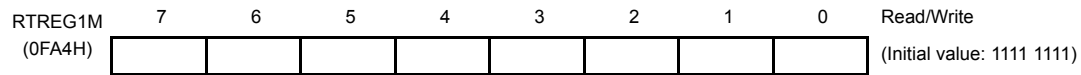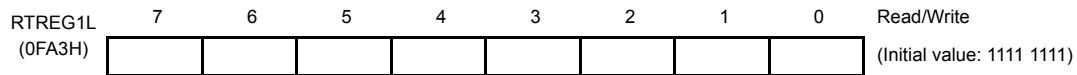
Note: When RTREG1H is read, bits 7 to 4 are read as "0".

RTC Counter 2 Monitor Register

DIVRG2L (0FA9H)	7	6	5	4	3	2	1	0	Read Only

DIVRG2M (0FAAH)	7	6	5	4	3	2	1	0	Read Only

DIVRG2H (0FABH)	7	6	5	4	3	2	1	0	Read Only
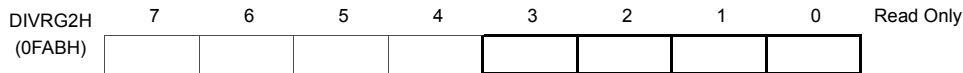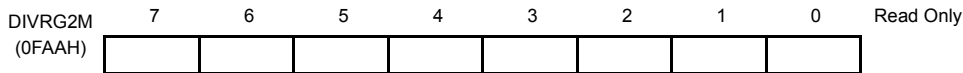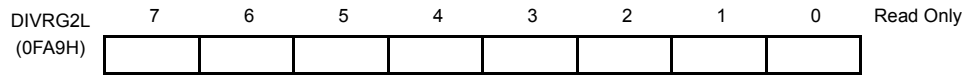
Note: When DIVRG2H is read, bits 7 to 4 are read as "0".

Leap Year Register

LEAPR (0FB3H)	7	6	5	4	3	2	1	0	Read/Write
									(Initial value: 0000 0000)
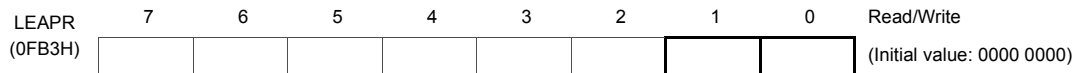
LEAP	Leap year counter setting	00: Leap year 01: Leap year + 1 year 10: Leap year + 2 years 11: Leap year + 3 years	R/W

Day-of-week Register
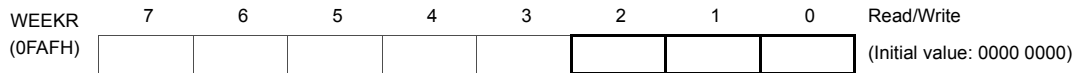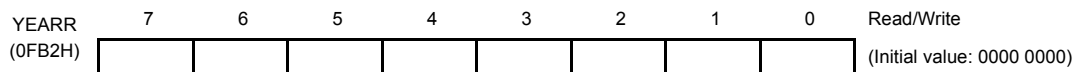
WEEKR (0FAFH)	7	6	5	4	3	2	1	0	Read/Write
									(Initial value: 0000 0000)

Year Register

YEARR (0FB2H)	7	6	5	4	3	2	1	0	Read/Write
									(Initial value: 0000 0000)

Month Register

MONTHR (0FB1H)	7	6	5	4	3	2	1	0	Read/Write
									(Initial value: 0000 0001)

Day Register

DAYR (0FB0H)	7	6	5	4	3	2	1	0	Read/Write
									(Initial value: 0000 0001)

Hours Register

HOURR (0FAEH)	7	6	5	4	3	2	1	0	Read/Write
									(Initial value: 0000 0000)

Minutes Register

MINR (0FADH)	7	6	5	4	3	2	1	0	Read/Write (Initial value: 0000 0000)

Seconds Register

SECR (0FACH)	7	6	5	4	3	2	1	0	Read/Write (Initial value: 0000 0000)
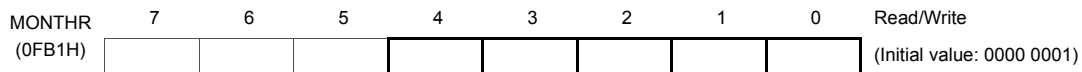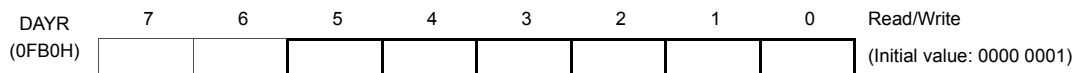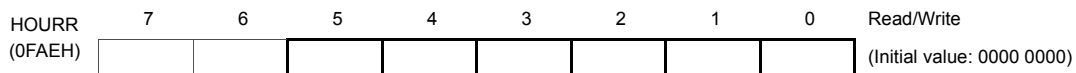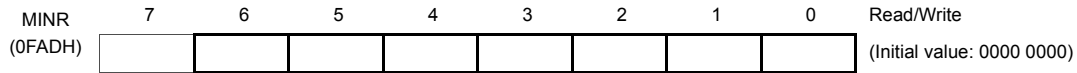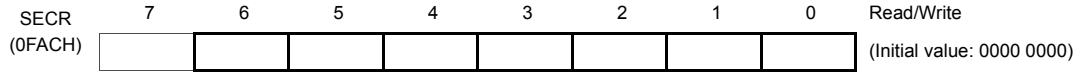
Note 1: Each register should be set in BCD format in the following range. If a hexadecimal value or a value outside the specified range is set, counting cannot be performed correctly.

SECR : 00-59   DAYR : 01-28, 29, 30, 31 (Note 2)   WEEKR : 00-06
MIN : 00-59   MONTHR : 01-12   LEAPR : 00-03
HOURR : 00-23   YEARR : 00-99

Note 2: The range of values that can be set or counted in the DAYR register depends on the MONTHR and LEAPR registers. Set the DAYR register as shown below according to the year and month settings. If a value outside the specified range is set, counting cannot be performed correctly.

DAYR   01-28   When MONTHR = 02H LEAPR ≠ 00H
01-29   When MONTHR = 02H LEAPR = 00H
01-30   When MONTHR = 04H, 06H, 09H, 11H
01-31   Others

Note 3: Writing to empty bits in these registers has no effect, and empty bits are read as "0".

11.2.1 RTC control registers

The first step in using the RTC is to select the source clock in RTCCR1<SCLOCK>. Then, RTC counter 1 can be started by setting RTCCR1<ENATC1> to "1". All the clock counters start counting by a match between the RTC counter 1 value and the RTC counter 1 compare register value.

RTCCR2<ENATC2> is the start control register for RTC counter 2. This counter can be used to measure the error of the low-frequency clock (stand-alone operation) or to detect a stop condition in the low-frequency oscillator when the system is operating in the dual-clock mode.

When STOP, SLEEP0, or IDLE0 mode is started, RTCCR1<SCLOCK>, RTCCR1<ENATC1>, and RTCCR2<ENATC2> are initialized and the RTC stops. To start the RTC again after exiting each mode, it is necessary to set RTCCR1<SCLOCK>, RTCCR1<ENATC1>, and RTCCR2<ENATC2> again.

11.2.2 Clock counter registers

There are eight clock counter registers: seconds register (SECR), minutes register (MINR), hours register (HOURR), day register (DAYR), month register (MONTHR), year register (YEARR), day-of-week register (WEEKR), and leap year register (LEAPR). All of these registers except the WEEKR and LEAPR registers count in BCD format.

All the clock counter registers are not initialized by a hardware reset (external or internal) in order to preserve the current time. Therefore, be sure to initialize these registers upon power-on. To prevent inadvertent clearing, the clock counter registers cannot be initialized while the RTC is running. To initialize these registers, set RTCCR2<RSTCLK> to "1" after the RTC has stopped completely. (This state can be confirmed by RTCSR<RTCF1> = "0" after clearing RTCCR1<ENATC1> to "0".) When RTCSR<RTCF1> = "1", the registers cannot be initialized even if RTCCR2<RSTCLK> is set to "1". The clock counter registers are initialized to year 00, January 1, 00 hours, 00 minutes, 00 seconds, LEAPR = "0", and WEEKR = "0".

When the leap year counter (LEAPR) is "0", the month counter (MONTHR) and the day counter (DAYR) operate to accommodate a leap year. Thus, the LEAPR register should be set to the remainder obtained by dividing the YEARR value by four. For example, if YEARR is "08H" (leap year), set LEAPR to "00H". If YEARR is "09H" (leap year + 1 year), set LEAPR to "01H".

For the week counter (WEEKR), assign the values 0 through 6 to the days of the week by programming.

The maximum value of the day counter (DAYR) varies depending on the month counter (MONTHR) and leap year counter (LEAPR).

The clock counter registers must be written while the RTC is not running (RTCSR<RTCF1> = "0") or the RTC count lock is enabled (RTCSR<TLOCK> = "1"). Write operations are ineffective in any other conditions. If non-BCD data is written to the counters using BCD format, counting cannot be performed correctly.

11.2.3 RTC counters 1 and 2

RTC counters 1 and 2 are 20-bit up counters. The source clock for RTC counter 1 can be selected by RTCCR1<SCLOCK>. The source clock for RTC counter 2 is fixed to low-frequency.

RTC counter 1 is compared with the RTC counter 1 compare register (RTREG1L/M/H) for a match. When a match occurs, one second is counted up in the clock ounter. The oscillator frequency can be corrected by changing the value set in the compare register. This compare function makes it possible to count each second more precisely. The frequency can be corrected on a basis of $1/fs = 30.5$ ppm ($fs = 32.768$ kHz) for the low-frequency source clock and on a basis of $1/fc = 1$ ppm ($fc = 1$ MHz) for the high-frequency source clock. It is also possible to enhance the precision of frequency correction further by changing the compare register value at fixed intervals by using INTRTC interrupts.

RTC counters 1 and 2 are not initialized by a hardware reset (internal or external). To prevent inadvertent clearing, RTC counters 1 and 2 cannot be initialized while the RTC is running. To initialize these counters, set RTCCR2<RSTDIV> to "1" after the RTC has stopped completely. (This state can be confirmed by

RTCSR<RTCF1/2> = "0" after clearing RTCCR1<ENATC1> and RTCCR2 <ENATC2> to "0".) When RTCSR<RTCF1/2> = "1", the RTC counter registers are not initialized even if RTCCR2<RSTDIV> is set to "1". When the initialization is completed, RTC counters 1 and 2 are cleared to "00000H".

11.2.4 RTC counter monitor registers

The values in RTC counters 1 and 2 can be read out by using the RTC counter 1 and RTC counter 2 monitor registers (DIVRG1L/M/H, DIVRG2L/M/H). The RTC counter monitor registers have the capture function to prevent the counter values from counting up while they are being read. When DIVRG1L is read, the low-order, middle-order, and high-order digits of the count value are captured all at once to DIVRG1L/M/H. Thus, be sure to read DIVRG1L, DIVRG1M, and DIVRG1H in this order. (The values in DIVRG1L, DIVRG1M, and DIVRG1H are updated only when DIVRG1L is read.) Likewise, the high-order, middle-order, and low-order digits of the count value in RTC counter 2 are also captured all at once when DIVRG2L is read. If the RTC counter counts up while it is being captured, the correct value cannot be obtained. Therefore, to obtain the correct value, read the monitor registers twice and check that the same value is read by two consecutive reads.

Note: When the system clock is high-frequency, RTC counter 1 using the high-frequency source clock counts up faster than the instruction execution cycle. This means that the same value cannot be read from the low-order 8 bits by two consecutive reads. Therefore, only the high-order and middle-order 12 bits should be used to determine whether or not the same value is read by two consecutive reads. This also applies when the system clock is low-frequency and RTC counters 1 and 2 using the low-frequency source clock are to be read. For details, see Example 1 below.

Example 1 : This program shows how to read RTC counter 1 correctly.
(When the system clock and source clock are both high-frequency, or when the system clock and source clock are both low-frequency)

```
LOOP:    LD    A, (DIVRG1L)    ; Read the low-order 8 bits of RTC counter 1 (first time).

         LD    L, (DIVRG1M)    ; Read the middle-order 8 bits of RTC counter 1 (first time).

         LD    H, (DIVRG1H)    ; Read the high-order 4 bits of RTC counter 1 (first time).

         LD    A, (DIVRG1L)    ; Read the low-order 8 bits of RTC counter 1 (second time).

         CMP   L, (DIVRG1M)    ; Compare the middle-order 8 bits of RTC counter 1 with the
                                 first read data.

         JR    NZ, LOOP        ; If not match, read DIVRG1L again.

         CMP   H, (DIVRG1H)    ; Compare the high-order 4 bits of RTC counter 1 with the first
                                 read data.

         JR    NZ, LOOP        ; If not match read DIVRG1L again.
```

Example 2 : This program shows how to read RTC counter 2 correctly.
(When the system clock is high-frequency and the source clock is low-frequency)

```
LOOP:    LD    A, (DIVRG2L)    ; Read the low-order 8 bits of RTC counter 2 (first time).

         LD    L, (DIVRG2M)    ; Read the middle-order 8 bits of RTC counter 2 (first time).

         LD    H, (DIVRG2H)    ; Read the high-order 4 bits of RTC counter 2 (first time).

         CMP   A, (DIVRG2L)    ; Compare the low-order 8 bits of RTC counter 2 with the first
                                 read data.

         JR    NZ, LOOP        ; If not match, read DIVRG2L again.

         CMP   L, (DIVRG2M)    ; Compare the middle-order 8 bits of RTC counter 2 with the
                                 first read data.

         JR    NZ, LOOP        ; If not match, read DIVRG2L again.

         CMP   H, (DIVRG2H)    ; Compare the high-order 4 bits of counter 2 with the first read
                                 data.

         JR    NZ, LOOP        ; If not match, read DIVRG2L again.
```
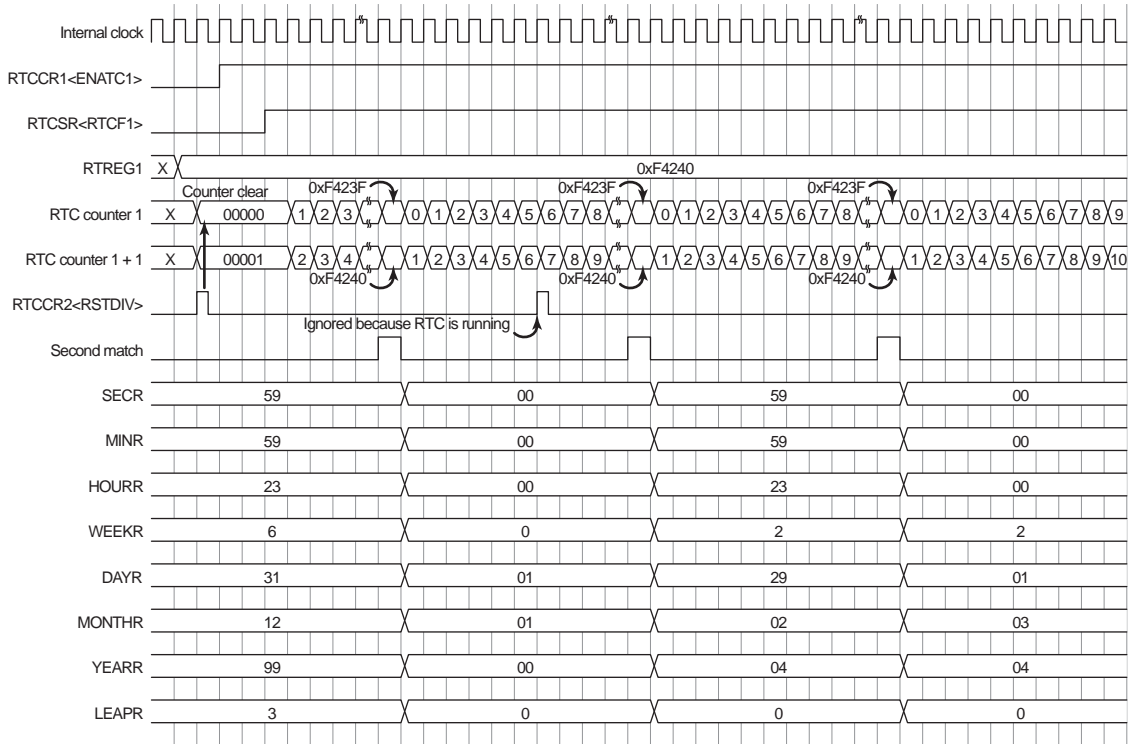
Figure 11-2 RTC Operation Timing

11.2.5 RTC counter 1 compare register

The RTC counter 1 compare register (RTREG1L/M/H) has a 2-stage structure. The value written to each compare register is reflected when a match is detected between the RTC counter 1 value and the compare register value written the last time around (when one second is counted up).

If a write to RTREG1H/M/L and a match detection occurs at the same time, the unstable value being written is shifted and next match detection may deviate from the specified value. Therefore, each compare register must be written while the RTC is inactive or immediately after the one-second match is detected. While the RTC is inactive, the value written to the compare register is reflected immediately.

Since the compare register comprises 3 bytes, if a match with RTC counter 1 occurs before 3 bytes have all been written, the desired value cannot be set for the next one second. For example, let us assume a case in which the current RTREG1 value is F8240H and this value should be set to F8180H for the next one second. If a one-second match is detected immediately after "80H" is written to RTREG1L, the RTREG1 value becomes F8280H and the intended one second cannot be realized (Figure 11-3).

The RTC counter 1 compare register is initialized to the full count (FFFFFH) by a hardware reset.
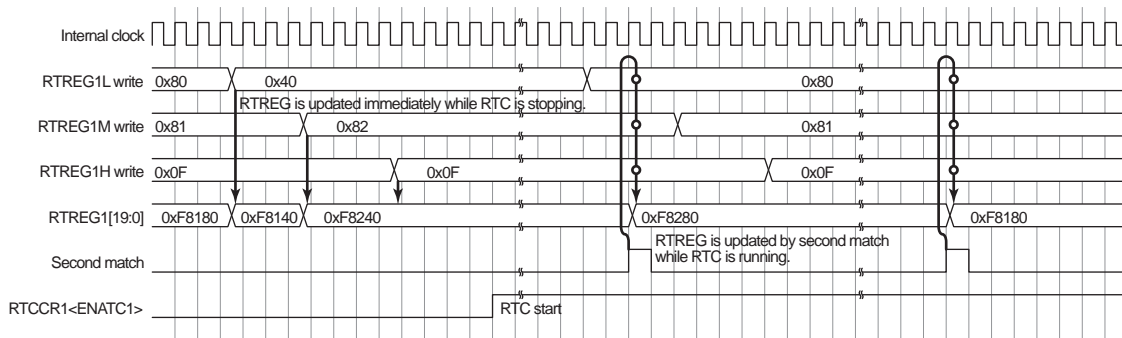
Figure 11-3 RTC Counter 1 Compare Register Update Timing

11.3 Function

11.3.1 Error measurement mode (RTCCR1<ADJEN1, ADJEN2>)

The error measurement mode is the function of measuring one second for determining the value to be set in the RTC counter compare register (RTREG1L/M/H). An external reference clock (normally 1 Hz) capable of counting seconds precisely is input from the RTCIN pin as the window pulse, and the width of this pulse is counted with RTC counter 1. The input of an external reference clock can be enabled by setting RTCCR2<ADJIN> to "0".

The window pulse width can be selected from four types in RTCCR2<ADJSEL>. The reference clock of 1 Hz can be divided to generate the window pulse of 1 second, 4 seconds, or 16 seconds. RTC counter 1 counts up while the window pulse is high. Error measurement can be performed more precisely as the window pulse becomes longer (although the measurement period also becomes longer.) The precision of error measurement can also be enhanced by increasing the source clock frequency. However, rotate operation is needed in this case. When the low-frequency source clock is used, it is recommended that error measurement be performed for 16 seconds for optimum results.

The output of RTC counter 1 based on the high-frequency source clock can be fed back to the input of RTC counter 2. This feature allows the user to adjust RTC counter 1 (high-frequency error) in one second and then to adjust RTC counter 2 (low-frequency error) on a stand-alone basis using 16 seconds. In this way, the time needed for low-frequency error adjustment can be shortened in the pre-shipment adjustment process.

When the window pulse goes low and error measurement finishes, RTCSR<RTCFn> is set to "0" and then RTC counter n stops. After RTC counter stopped, clear RTCCR1<ENATCn> to "0", read the RTC counter monitor registers (DIVRGnL/M/H). (n=1,2) If the monitor registers overflowed in 16-second measurement, the clock counter registers should also be read. Then, calculate the clock count per second and set this value in the RTC counter 1 compare register (RTREG1L/M/H).

Note 1: To measure the error of RTC counter 2 based the output of RTC counter 1, set RTCCR2<ENATC2, ADJEN2> to "1". Then, after confirming RTCSR<ADJF2> = "1", start RTC counter 1. For details, see Example 2.

Note 2: In the error measurement mode, an error of 1 source clock is generated at the rising and falling edges of the window pulse respectively (an error of 2 source clocks in total). For details, see Table 11-1 and Table 11-2.

Note 3: When RTCCR2<ADJSEL> (window pulse width select) is set to "00", set RTCCR1<ENATC1, ADJEN1> to "1" and then check RTCSR<ADJF1> = "1" before inputting "H" level on the RTCIN pin. If "H" level is input before RTCSR<ADJF1> has become "1", the window pulse width may become shorter than intended.
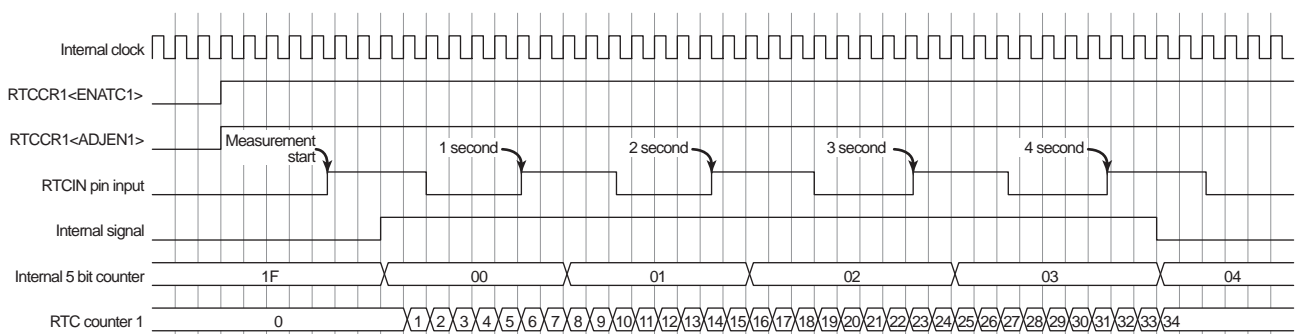
Figure 11-4 Timing Example of 4-Clock Measurement

Table 11-1 Measurement Error in the Error Measurement Mode (High-Frequency Source Clock)

Frequency [MHz]	Measurement Period [seconds]	Source Clock	Measurement Error [ppm]	RTREG1 Set Value (at measurement)	Count Loop Times
16	1	fc/16	2	FFFFFH	1
		fc/8	1	F4240H	2
		fc/4	0.5		4
		fc/2	0.25		8
		fc	0.125		16 (Example 3)
8		fc/8	2	FFFFFH	1
		fc/4	1	F4240H	2
		fc/2	0.5		4
		fc	0.25		8
4		fc/4	2	FFFFFH	1
		fc/2	1	F4240H	2
		fc	0.5		4
2		fc/2	2	FFFFFH	1
		fc	1	F4240H	2
1		fc	2	FFFFFH	1 (Example 1)
8	4	fc/2	0.125	F4240H	16
4		fc/2	0.25		8
		fc	0.125		16
2		fc/2	0.5		4
		fc	0.25		8
1		fc/2	1		2
		fc	0.5		4
2	16	fc/2	0.125	F4240H	16
1		fc/2	0.25		8
		fc	0.125		16

Table 11-2 Measurement Error in the Error Measurement Mode (Low-Frequency Source Clock)

Frequency [kHz]	Measurement Period [seconds]	Source Clock	Measurement Error [ppm]	RTREG1/2 Set Value (at measurement)	Count Loop Times
32.768	1	fs	61.04	FFFFFH	1
	4		15.26		4
	16		3.81		16 (Example 2)

Note 1: The value to be set in the RTC counter compare register is obtained by dividing the measurement value under each condition (RTC counter monitor register value) by the count loop times. If the source clock is high-frequency and the count loop times is "2" or larger, include the seconds register (SECR) in the calculation. For details, see Example 3.

Note 2: The total error per second is calculated by "(1 source clock period / 2) + measurement error [ppm]". Thus, the maximum total error per second is "0.5 + measurement error [ppm]" for the high-frequency source clock and "15.25 + measurement error [ppm]" for the low-frequency source clock. The error in the "1 source clock period / 2" portion can be decreased in software processing of the clock mode by using the bits dropped by rotate operation in the error measurement mode.

Example 1 : This program runs RTC counter 1 in the error measurement mode based on the RTCIN pin input. After measurement is done, the measurement data is stored in the RTC counter 1 compare register. Then, RTC counter 1 is run in the clock mode.

(fc = 1 MHz, RTC counter 1 source clock = fc, precise 1 Hz external input from RTCIN pin, measurement period = 1 second)

```
                LD          HL, RTREG1L
                LD          DE, RTREG1H
                LD          (HL), 0FFFFH        ; RTREG1M,L = FFFFH
                LD          (DE), 0FH           ; RTREG1H = 0FH
                LD          (RTCCR1), 0000_0100B  ; Set the source clock to fc.
                LD          (RTCCR2), 0011_0001B  ; Initialize RTC counters, set for RTCIN pin input, and set the window pulse
                                                    to 1 second.
                LD          (RTCCR1), 1100_0100B  ; Start RTC counter 1 in the error measurement mode.
LOOP1:
                TEST        (RTCSR). 1
                JRS         F, LOOP1            ; Loop until RTC counter 1 stops.

                LD          (RTCCR1), 0000_0100B  ; Stop RTC counter 1.
                LD          WA, (DIVRG1L)       ; Read the counter monitor register and store value in the compare register.
                LD          (RTREG1L), WA
                LD          A, (DIVRG1H)        ;Read the counter monitor register and store value in the compare register.
                LD          (RTREG1H), A

                LD          (RTCCR1), 1000_0100B  ; Start RTC counter 1 in the clock mode.
```

Example 2 : This program runs RTC counter 2 in the error measurement mode based on RTC counter 1. When measurement is done, the measurement data is stored at addresses 40 to 42 in RAM.
(fc = 16 MHz, fs = 32.768 kHz, RTC counter 1 source clock = fc/16, measurement period = 16 seconds)

```
            LD          HL, RTREG1L
            LD          DE, RTREG1H
            LD          (HL), 4240H              ; RTREG1M, L = 4240H (measurement value = 0F4240H)
            LD          (DE),  0FH               ; RTREG1H = 0FH
            LD          (RTCCR1), 0000_1100B     ; Set the source clock to fc/16.
            LD          (RTCCR2), 0010_0111B     ; Initialize RTC counters, set for RTCOUT pin internal output
                                                 pulse,and set the window pulse width to 16 seconds.
            LD          (RTCCR2), 1110_0111B     ; Start RTC counter 2 in the error measurement mode.
LOOP1:
            TEST        (RTCSR). 4               ; Wait for RTC counter 2 to enter the error measurement
                                                 mode.
            JRS         T, LOOP1
            LD          (RTCCR1), 1000_1100B     ; Start RTC counter 1 in the clock mode.
LOOP2:
            TEST        (RTCSR). 2
            JRS         F, LOOP2                 ; Loop until RTC counter 2 stops.

            LD          (RTCCR2), 0000_0111B     ; Stop RTC counter 2.
            LD          WA, (DIVRG2L)
            LD          (40H), WA
            LD          A, (DIVRG2H)
            LD          (42H), A
```

Example 3 : This program runs RTC counter 1 in the error measurement mode based on the RTCIN pin input. When measurement is done, the measurement data is stored in the RTC counter 1 compare register. Then, RTC 1 counter is run in the clock mode.
(fc = 16 MHz, RTC counter 1 source clock = fc, precise 1 Hz external input from RTCIN pin, measurement period = 1 second)

```
            LD      HL, RTREG1L
            LD      DE, RTREG1H
            LD      (HL), 4240H         ; RTREG1M, L = 4240H
            LD      (DE), 0FH           ; RTREG1H = 0FH
            LD      (RTCCR1), 0000_0100B ; Set source clock to fc.
            LD      (RTCCR2), 0011_0001B ; Initialize RTC counters, set for RTCIN pin input, and set the
                                          window pulse width to 1 second.
            LD      (RTCCR1), 1100_0100B ; Start RTC counter 1 in the error measurement mode.
LOOP1:
            TEST    (RTCSR). 1
            JRS     F, LOOP1            ; Loop until RTC counter 1 stops.

            LD      (RTCCR1), 0000_0100B ; Stop RTC counter 1.
sShift:                                 ; Shift 4 bits to the right (rotate operation).
            LD      WA, (DIVRG1L)
            LD      (RTREG1L), A        ; Store data in the compare register (low-order).
            LD      A, W
            RORD    A, (RTREG1L)
            LD      (RTREG1M), A        ; Store data in compare register (middle-order).
            LD      A, (DIVRG1H)
            RORD    A, (RTREG1M)
            LD      WA, (RTREG1L)
            CMP     (SECR), 10H
            JR      Z, s16sec           ; If SECR = 16 seconds, go to s16sec.
            CMP     (SECR), 0FH
            JR      Z, s15sec           ; If SECR = 15 seconds, go to s15sec.
            JR      sError              ; In other cases, go to sError.
s16sec:
            ADD     WA, 4240H
            JR      sSecend
s15sec:
            LD      HL, 0F424H
            SUB     HL, WA
            LD      WA, 4240H
            SUB     WA, HL
sSecend:
            LD      (RTREG1L), WA
            LD      A, 0FH
            LD      (RTREG1H), A        ; Store data in the compare register (high-order).

            LD      (RTCCR1), 1000_1100B ; Start RTC counter 1 in the clock mode (fc/16).
            :       :               :
sError:                                 ; Go to error processing.
```

11.3.2 Clock counter lock control (RTCCR1<THOLD>)

If the counting up of one second occurs while the clock counter register is being accessed, the correct time may not be obtained or written. To avoid this phenomenon, the counting up of the clock counters can be suspended.

When RTCCR1<THOLD> is set to "1", the count up operation is suspended after RTCSR<TLOCK> becomes "1". If the counting up of one second occurs while the count up operation is suspended, this one second will be counted up as soon as the count up operation is enabled again.

Note: If is only one second that can be held by this function. If the counting up of one second occurs multiple times while the count up operation is suspended, the clock counter will advance only one second after the count up operation is enabled again.
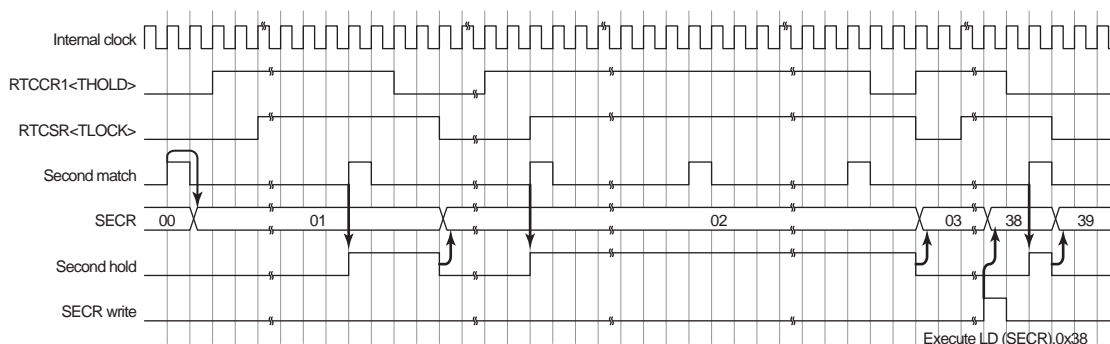
Figure 11-5 Count Up Control Timing Diagram

Example 1 : This program suspends the count up operation and writes to the clock counter registers while RTC counter 1 is running.
(The seconds and minutes registers are at addresses 40H and 41H in RAM, respectively.)

```
            LD      HL, RTCCR1
            LD      DE, RTCSR
            SET     (HL). 0             ; Set THOLD to "1".
    LOOP:
            TEST    (DE). 0             ; Wait until TLOCK = "1".
            JRS     T, LOOP

            LD      IX, SECR
            LD      A, (40H)
            LD      (IX), A             ; Set the seconds register.

            LD      IX, MINR
            LD      A, (41H)
            LD      (IX), A             ; Set the minutes register.

            CLR     (HL). 0             ; Clear THOLD to "0".
```

Example 2 :This program suspends the count up operation and reads the clock counter registers while RTC counter 1 is running.
(The seconds and minutes registers are stored at addresses 40H and 41H in RAM, respectively.)

```
                    LD          HL, RTCCR1

                    LD          DE, RTCSR

                    SET         (HL). 0            ; Set THOLD to "1".

        LOOP:

                    TEST        (DE). 0            ; Wait until TLOCK = "1".

                    JRS         T, LOOP


                    LD          A, (SECR)

                    LD          (40H), A           ; Read the seconds register and store data at address 40H.


                    LD          A, (MINR)

                    LD          (41H), A           ; Read the minutes register and store data at address 41H.


                    CLR         (HL). 0            ; Clear THOLD to "0".
```

Example 3 :This program reads the clock counter registers while RTC counter 1 is running (without using
RTCCR1<THOLD>).
(The seconds and minutes registers are stored at addresses 40H and 41H in RAM, respectively.)

```
                    LD          IX, DIVRG1L

        LOOP1:

                    LD          W, (IX)            ; Read the low-order 8 bits of RTC counter 1.

                    LD          HL, (IX+1)         ; Read the high-order 4 bits and middle-order 8 bits of RTC
                                                     counter1.

                    LD          W, (IX)            ; Read the low-order 8 bits of RTC counter 1.

                    CMP         HL, (IX+1)         ; Read the high-order 4 bits and middle-order 8 bits of RTC
                                                     counter1.

                    JR          NZ, LOOP1          ; If data could not be read correctly, return to LOOP1.


                    LD          A, (SECR)

                    LD          (40H), A           ; Read the seconds register and store data at address 40H.


                    LD          A, (MINR)

                    LD          (41H), A           ; Read the minutes register and store data at address 41H.

        LOOP2:

                    LD          W, (IX)            ; Read the low-order 8 bits of RTC counter 1.

                    LD          DE, (IX+1)         ; Read the high-order 4 bits and middle-order 8 bits of RTC
                                                     counter 1.

                    LD          W, (IX)            ; Read the low-order 8 bits of RTC counter 1.

                    CMP         DE, (IX+1)         ; Read the high-order 4 bits and middle-order 8 bits of RTC
                                                     counter 1.

                    JR          NZ, LOOP2          ; If data could not be read correctly, return to LOOP2.


                    CMP         DE, HL

                    JR          LT, LOOP1          ; If count up occurred, return to LOOP1.
```

11.3.3 RTCOUT pin output

While the RTC is running, pulses equivalent to 1 Hz can be output from the RTCOUT pin.

Note 1: To use RTCOUT output, set RTCCR2<DIS1SC> to "1" and then start the RTC (RTCCR1<ENATC1> = "1").
If these steps are not performed in this order, the pulse for the first one second cannot be precisely 1 Hz.

Note 2: If the RTC is stopped while output is being made from the RTCOUT pin, the RTCOUT pin will be fixed at "H" level.

Note 3: The falling edge of RTCOUT output is triggered by a match between the RTC counter 1 value and the RTC counter 1 compare register value. The rising edge of RTCOUT is triggered by a match between the RTC counter 1 value and the RTC counter 1 compare register value divided by two. Thus, if the value set in the RTC counter 1 compare register is not an even number, the pulse width cannot be precisely 0.5 seconds. (The "L" level width becomes one source clock shorter than the "H" level width.)

Table 11-3 Duty Cycles of the RTCOUT Output Waveform

RTCCR1 <SCLOCK>		RTREG1H/M/L	32.768 kHz	1 MHz	2 MHz	4 MHz	8 MHz	16 MHz
000	L width	08000H	0.5	–	–	–	–	–
	H width		0.5	–	–	–	–	–
001	L width	F4240H	–	0.5	0.25	0.125	0.0625	0.03125
	H width		–	0.5	0.25	0.125	0.0625	0.03125
010	L width		–	1.0	0.5	0.25	0.125	0.0625
	H width		–	1.0	0.5	0.25	0.125	0.0625
011	L width		–	2.0	1.0	0.5	0.25	0.125
	H width		–	2.0	1.0	0.5	0.25	0.125
100	L width		–	4.0	2.0	1.0	0.5	0.25
	H width		–	4.0	2.0	1.0	0.5	0.25
101	L width		–	8.0	4.0	2.0	1.0	0.5
	H width		–	8.0	4.0	2.0	1.0	0.5
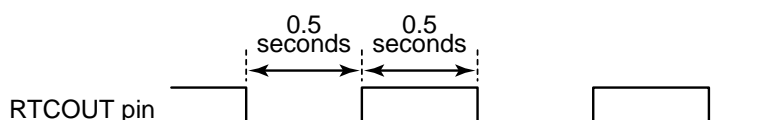
Figure 11-6 RTCOUT Output Waveform

12. Synchronous Serial Interface (SIO)

The TMP86FM26UG has a clocked-synchronous 8-bit serial interface. Serial interface has an 8-byte transmit and receive data buffer that can automatically and continuously transfer up to 64 bits of data.

Serial interface is connected to outside peripherl devices via SO, SI, SCK port.

12.1 Configuration

SIO control / status register
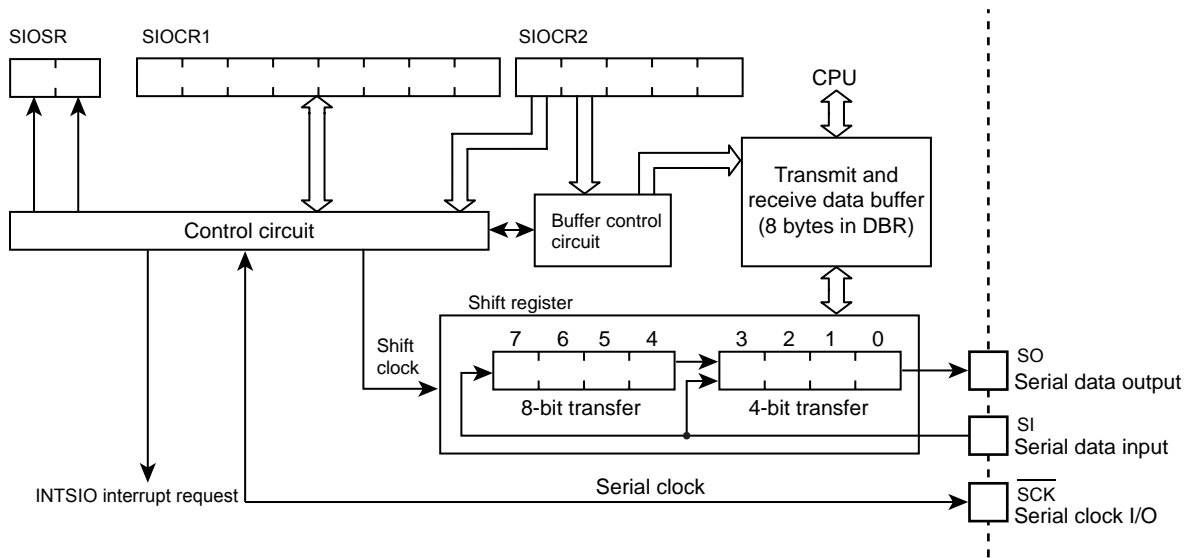
Figure 12-1 Serial Interface

12.2 Control

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

The transmit and receive data buffer is controlled by the SIOCR2<BUF>. The data buffer is assigned to address 0F90H to 0F97H for SIO in the DBR area, and can continuously transfer up to 8 words (bytes or nibbles) at one time. When the specified number of words has been transferred, a buffer empty (in the transmit mode) or a buffer full (in the receive mode or transmit/receive mode) interrupt (INTSIO) is generated.

When the internal clock is used as the serial clock in the 8-bit receive mode and the 8-bit transmit/receive mode, a fixed interval wait can be applied to the serial clock for each word transferred. Four different wait times can be selected with SIOCR2<WAIT>.

SIO Control Register 1

SIOCR1	7	6	5	4	3	2	1	0	
(0F98H)	SIOS	SIOINH		SIOM			SCK		(Initial value: 0000 0000)
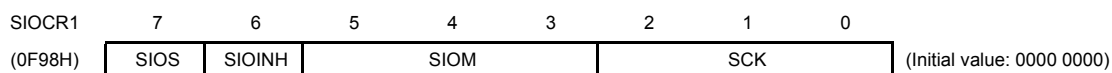
SIOS	Indicate transfer start / stop	0: Stop 1: Start				Write only
SIOINH	Continue / abort transfer	0: Continuously transfer 1: Abort transfer (Automatically cleared after abort)				
SIOM	Transfer mode select	000: 8-bit transmit mode 010: 4-bit transmit mode 100: 8-bit transmit / receive mode 101: 8-bit receive mode 110: 4-bit receive mode Except the above: Reserved				
SCK	Serial clock select		NORMAL1/2, IDLE1/2 mode		SLOW1/2 SLEEP1/2 mode	Write only
			DV7CK = 0	DV7CK = 1		
		000	$fc/2^{13}$	$fs/2^5$	$fs/2^5$	
		001	$fc/2^8$	$fc/2^8$	-	
		010	$fc/2^7$	$fc/2^7$	-	
		011	$fc/2^6$	$fc/2^6$	-	
		100	$fc/2^5$	$fc/2^5$	-	
		101	$fc/2^4$	$fc/2^4$	-	
		110	Reserved			
		111	External clock (Input from SCK pin)			

Note 1: fc; High-frequency clock [Hz], fs; Low-frequency clock [Hz]
Note 2: Set SIOS to "0" and SIOINH to "1" when setting the transfer mode or serial clock.
Note 3: SIOCR1 is write-only register, which cannot access any of in read-modify-write instruction such as bit operate, etc.

SIO Control Register 2

SIOCR2	7	6	5	4	3	2	1	0	
(0F99H)				WAIT		BUF			(Initial value: ***0 0000)

WAIT	Wait control	Always sets "00" except 8-bit transmit / receive mode. 00: $T_f = T_D$(Non wait) 01: $T_f = 2T_D$(Wait) 10: $T_f = 4T_D$(Wait) 11: $T_f = 8T_D$ (Wait)	Write only
BUF	Number of transfer words (Buffer address in use)	000: 1 word transfer 0F90H 001: 2 words transfer 0F90H ~ 0F91H 010: 3 words transfer 0F90H ~ 0F92H 011: 4 words transfer 0F90H ~ 0F93H 100: 5 words transfer 0F90H ~ 0F94H 101: 6 words transfer 0F90H ~ 0F95H 110: 7 words transfer 0F90H ~ 0F96H 111: 8 words transfer 0F90H ~ 0F97H	

Note 1: The lower 4 bits of each buffer are used during 4-bit transfers. Zeros (0) are stored to the upper 4bits when receiving.

Note 2: Transmitting starts at the lowest address. Received data are also stored starting from the lowest address to the highest address. (The first buffer address transmitted is 0F90H).

Note 3: The value to be loaded to BUF is held after transfer is completed.

Note 4: SIOCR2 must be set when the serial interface is stopped (SIOF = 0).

Note 5: *: Don't care

Note 6: SIOCR2 is write-only register, which cannot access any of in read-modify-write instruction such as bit operate, etc.

SIO Status Register

SIOSR	7	6	5	4	3	2	1	0
(0F99H)	SIOF	SEF						
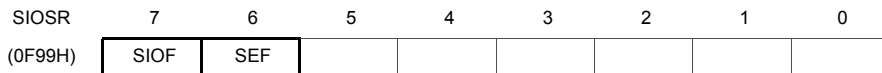
SIOF	Serial transfer operating status monitor	0: Transfer terminated 1: Transfer in process	Read only
SEF	Shift operating status monitor	0: Shift operation terminated 1: Shift operation in process	

Note 1: T_f; Frame time, T_D; Data transfer time

Note 2: After SIOS is cleared to "0", SIOF is cleared to "0" at the termination of transfer or the setting of SIOINH to "1".
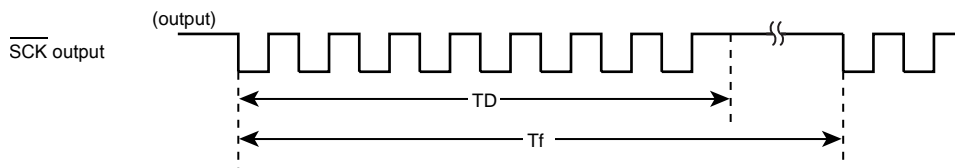
Figure 12-2 Frame time (T_f) and Data transfer time (T_D)

12.3 Serial clock

12.3.1 Clock source

Internal clock or external clock for the source clock is selected by SIOCR1<SCK>.

12.3.1.1 Internal clock

Any of six frequencies can be selected. The serial clock is output to the outside on the SCK pin. The SCK pin goes high when transfer starts.

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

Table 12-1 Serial Clock Rate

	NORMAL1/2, IDLE1/2 mode				SLOW1/2, SLEEP1/2 mode	
	DV7CK = 0		DV7CK = 1			
SCK	Clock	Baud Rate	Clock	Baud Rate	Clock	Baud Rate
000	$fc/2^{13}$	1.91 Kbps	$fs/2^5$	1024 bps	$fs/2^5$	1024 bps
001	$fc/2^8$	61.04 Kbps	$fc/2^8$	61.04 Kbps	-	-
010	$fc/2^7$	122.07 Kbps	$fc/2^7$	122.07 Kbps	-	-
011	$fc/2^6$	244.14 Kbps	$fc/2^6$	244.14 Kbps	-	-
100	$fc/2^5$	488.28 Kbps	$fc/2^5$	488.28 Kbps	-	-
101	$fc/2^4$	976.56 Kbps	$fc/2^4$	976.56 Kbps	-	-
110	-	-	-	-	-	-
111	External	External	External	External	External	External

Note: 1 Kbit = 1024 bit (fc = 16 MHz, fs = 32.768 kHz)
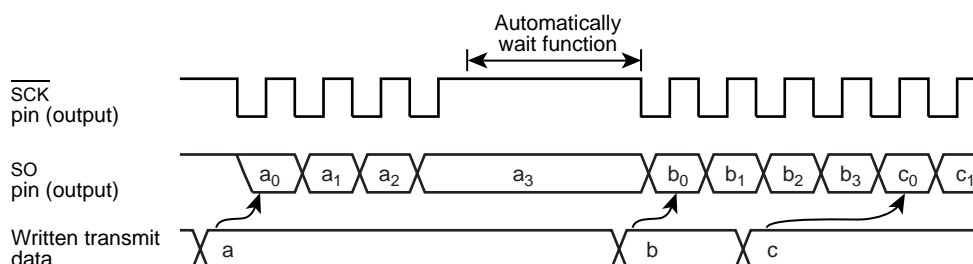
Figure 12-3 Automatic Wait Function (at 4-bit transmit mode)

12.3.1.2 External clock

An external clock connected to the $\overline{SCK}$ pin is used as the serial clock. In this case, output latch of this port should be set to "1". To ensure shifting, a pulse width of at least 4 machine cycles is required. This pulse is needed for the shift operation to execute certainly. Actually, there is necessary processing time for interrupting, writing, and reading. The minimum pulse is determined by setting the mode and the program. Therfore, maximum transfer frequency will be 488.3K bit/sec (at fc=16MHz).

$t_{cyc} = 4/fc$ (In the NORMAL1/2, IDLE1/2 modes)
$4/fs$ (In the SLOW1/2, SLEEP1/2 modes)
$t_{SCKL}, t_{SCKH} > 4t_{cyc}$
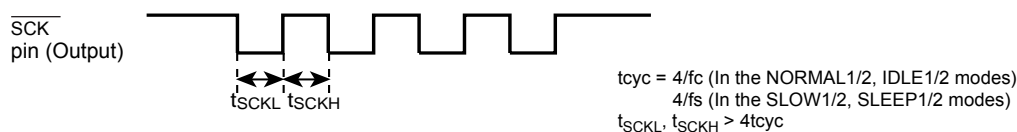
Figure 12-4 External clock pulse width

12.3.2 Shift edge

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

12.3.2.1 Leading edge

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

12.3.2.2 Trailing edge

Received data are shifted on the trailing edge of the serial clock (rising edge of the $\overline{SCK}$ pin input/output).
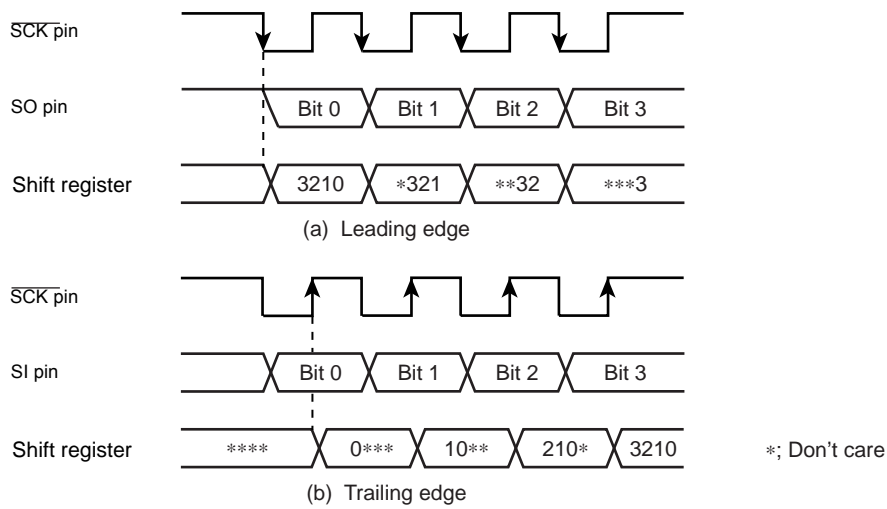
Figure 12-5 Shift edge

12.4 Number of bits to transfer

Either 4-bit or 8-bit serial transfer can be selected. When 4-bit serial transfer is selected, only the lower 4 bits of the transmit/receive data buffer register are used. The upper 4 bits are cleared to "0" when receiving.
The data is transferred in sequence starting at the least significant bit (LSB).

12.5 Number of words to transfer

Up to 8 words consisting of 4 bits of data (4-bit serial transfer) or 8 bits (8-bit serial transfer) of data can be transferred continuously. The number of words to be transferred can be selected by SIOCR2<BUF>.

An INTSIO interrupt is generated when the specified number of words has been transferred. If the number of words is to be changed during transfer, the serial interface must be stopped before making the change. The number of words can be changed during automatic-wait operation of an internal clock. In this case, the serial interface is not required to be stopped.
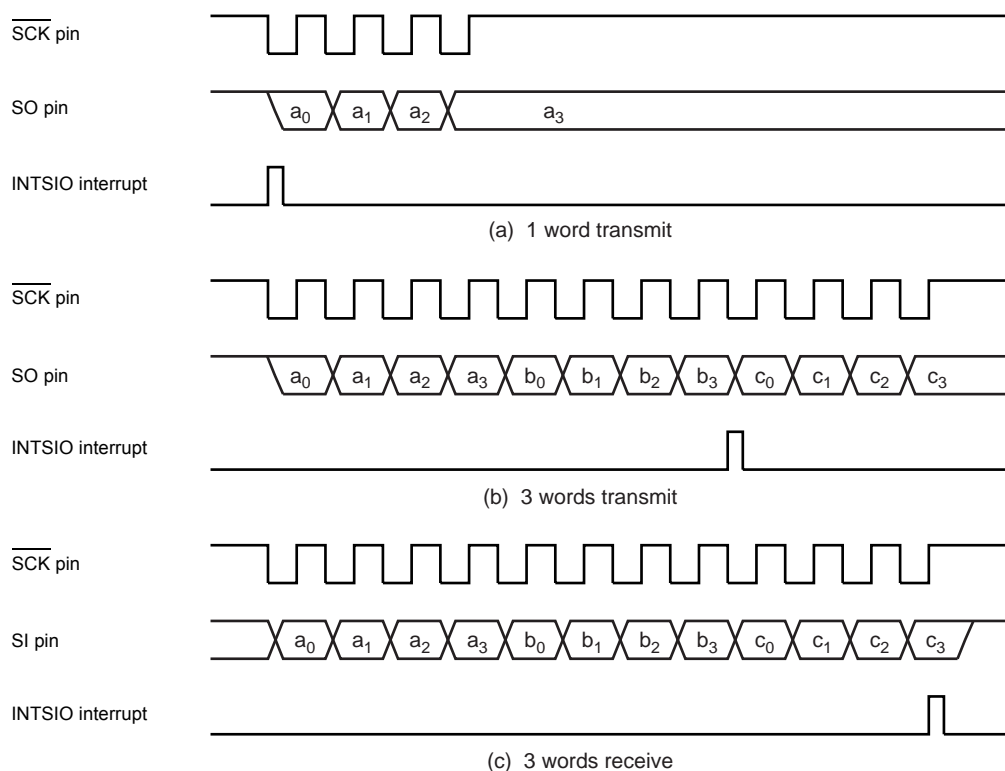
Figure 12-6 Number of words to transfer (Example: 1word = 4bit)

12.6 Transfer Mode

SIOCR1<SIOM> is used to select the transmit, receive, or transmit/receive mode.

12.6.1 4-bit and 8-bit transfer modes

In these modes, firstly set the SIO control register to the transmit mode, and then write first transmit data (number of transfer words to be transferred) to the data buffer registers (DBR).

After the data are written, the transmission is started by setting SIOCR1<SIOS> to "1". The data are then output sequentially to the SO pin in synchronous with the serial clock, starting with the least significant bit (LSB). As soon as the LSB has been output, the data are transferred from the data buffer register to the shift register. When the final data bit has been transferred and the data buffer register is empty, an INTSIO (Buffer empty) interrupt is generated to request the next transmitted data.

When the internal clock is used, the serial clock will stop and an automatic-wait will be initiated if the next transmitted data are not loaded to the data buffer register by the time the number of data words specified with the SIOCR2<BUF> has been transmitted. Writing even one word of data cancels the automatic-wait; therefore, when transmitting two or more words, always write the next word before transmission of the previous word is completed.

Note: Automatic waits are also canceled by writing to a DBR not being used as a transmit data buffer register; therefore, during SIO do not use such DBR for other applications. For example, when 3 words are transmitted, do not use the DBR of the remained 5 words.

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

The transmission is ended by clearing SIOCR1<SIOS> to "0" or setting SIOCR1<SIOINH> to "1" in buffer empty interrupt service program.

SIOCR1<SIOS> is cleared, the operation will end after all bits of words are transmitted.

That the transmission has ended can be determined from the status of SIOSR<SIOF> because SIOSR<SIOF> is cleared to "0" when a transfer is completed.

When SIOCR1<SIOINH> is set, the transmission is immediately ended and SIOSR<SIOF> is cleared to "0".

When an external clock is used, it is also necessary to clear SIOCR1<SIOS> to "0" before shifting the next data; If SIOCR1<SIOS> is not cleared before shift out, dummy data will be transmitted and the operation will end.

If it is necessary to change the number of words, SIOCR1<SIOS> should be cleared to "0", then SIOCR2<BUF> must be rewritten after confirming that SIOSR<SIOF> has been cleared to "0".

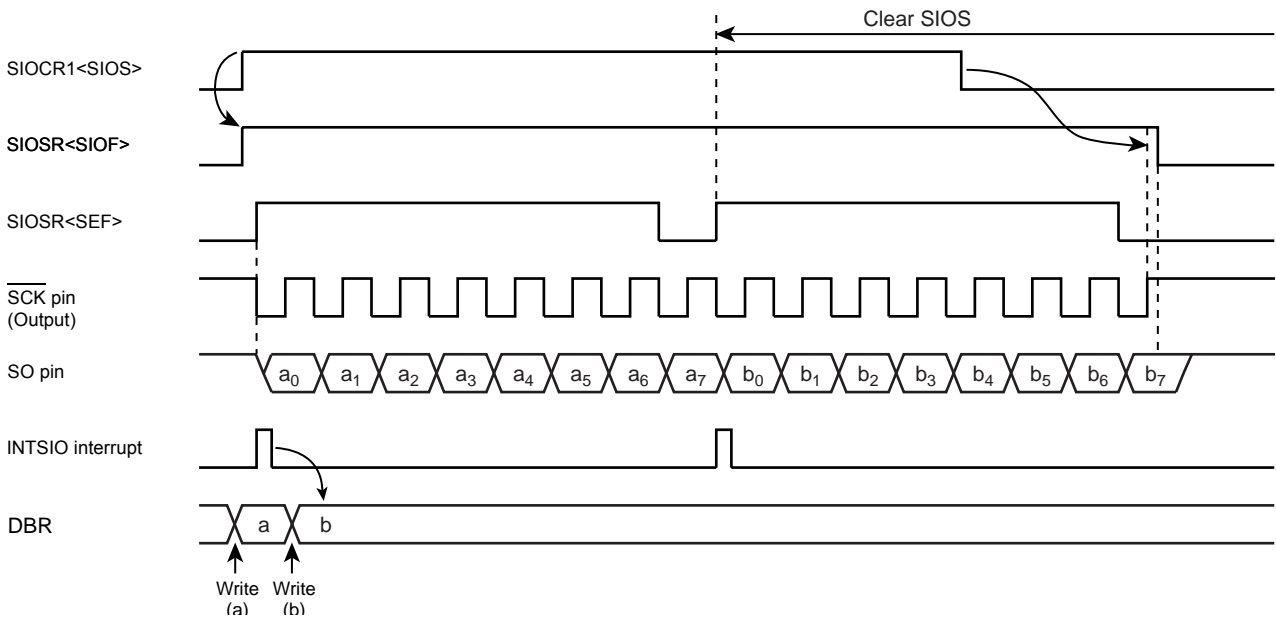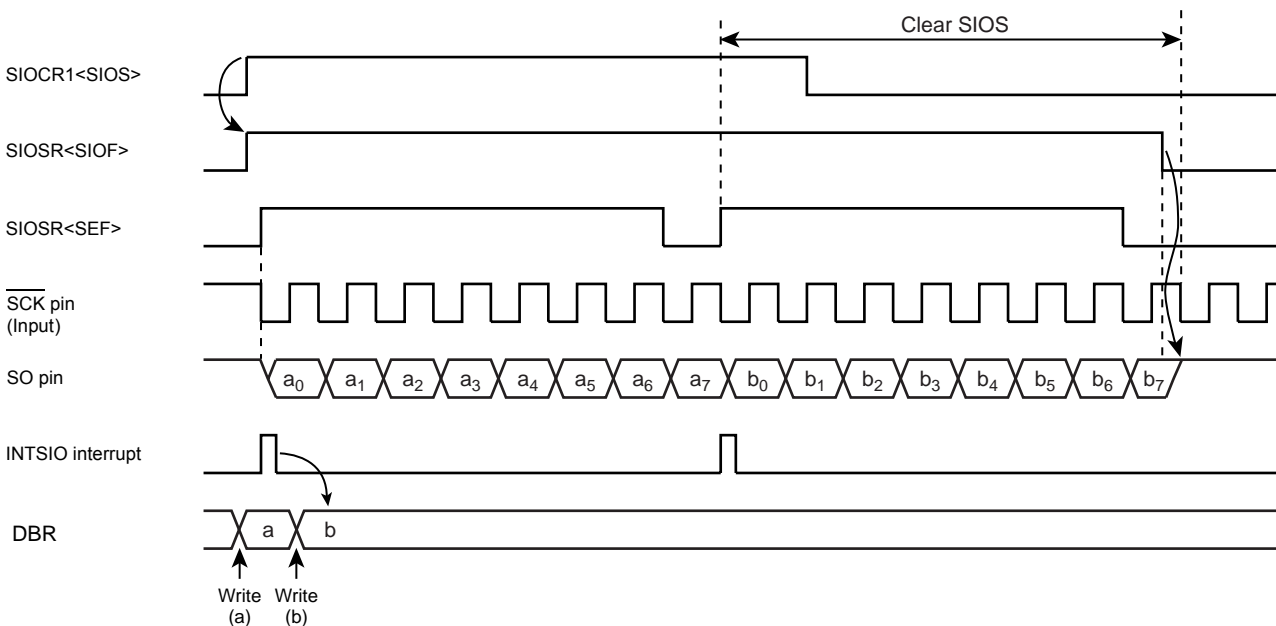Figure 12-7 Transfer Mode (Example: 8bit, 1word transfer, Internal clock)

Figure 12-8 Transfer Mode (Example: 8bit, 1word transfer, External clock)

t_{SODH} = min 3.5/fc [s] (In the NORMAL1/2, IDLE1/2 modes)
t_{SODH} = min 3.5/fs [s] (In the SLOW1/2, SLEEP1/2 modes)
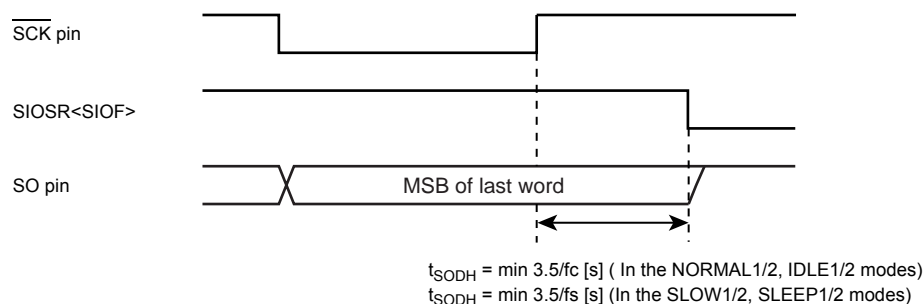
Figure 12-9 Transmiiied Data Hold Time at End of Transfer

12.6.2 4-bit and 8-bit receive modes

After setting the control registers to the receive mode, set SIOCR1<SIOS> to "1" to enable receiving. The data are then transferred to the shift register via the SI pin in synchronous with the serial clock. When one word of data has been received, it is transferred from the shift register to the data buffer register (DBR). When the number of words specified with the SIOCR2<BUF> has been received, an INTSIO (Buffer full) interrupt is generated to request that these data be read out. The data are then read from the data buffer registers by the interrupt service program.

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

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

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

The receiving is ended by clearing SIOCR1<SIOS> to "0" or setting SIOCR1<SIOINH> to "1" in buffer full interrupt service program.
When SIOCR1<SIOS> is cleared, the current data are transferred to the buffer. After SIOCR1<SIOS> cleared, the receiving is ended at the time that the final bit of the data has been received. That the receiving has ended can be determined from the status of SIOSR<SIOF>. SIOSR<SIOF> is cleared to "0" when the receiving is ended. After confirmed the receiving termination, the final receiving data is read. When SIOCR1<SIO-INH> is set, the receiving is immediately ended and SIOSR<SIOF> is cleared to "0". (The received data is ignored, and it is not required to be read out.)

If it is necessary to change the number of words in external clock operation, SIOCR1<SIOS> should be cleared to "0" then SIOCR2<BUF> must be rewritten after confirming that SIOSR<SIOF> has been cleared to "0". If it is necessary to change the number of words in internal clock, during automatic-wait operation which occurs after completion of data receiving, SIOCR2<BUF> must be rewritten before the received data is read out.

Note: The buffer contents are lost when the transfer mode is switched. If it should become necessary to switch the transfer mode, end receiving by clearing SIOCR1<SIOS> to "0", read the last data and then switch the transfer mode.
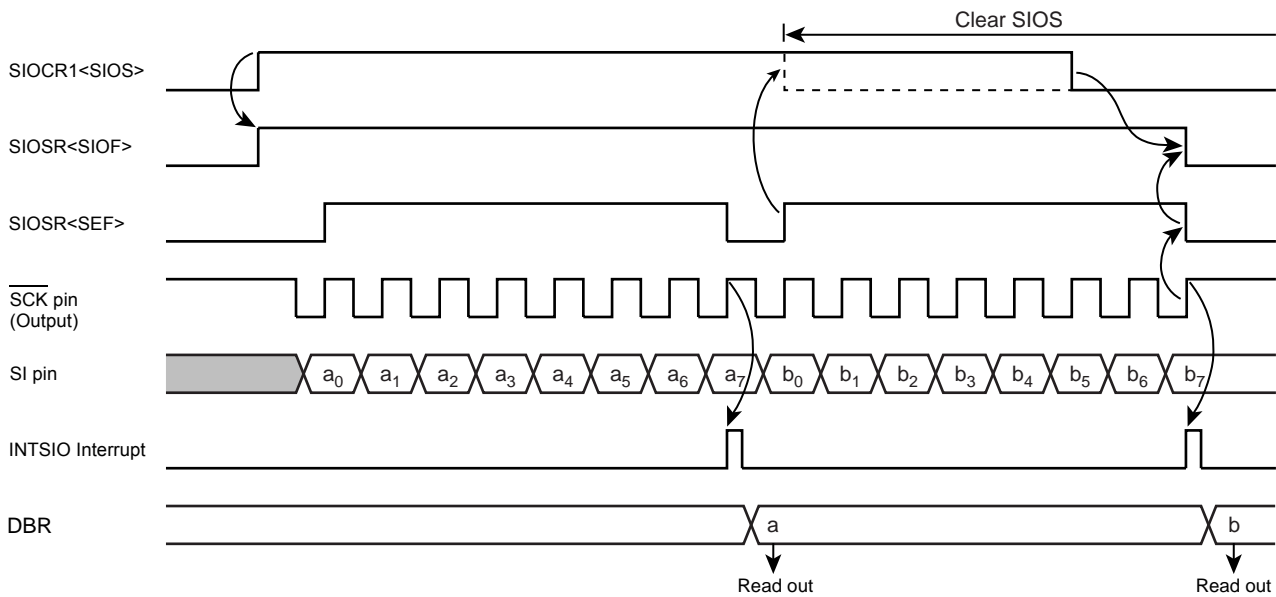
Figure 12-10 Receive Mode (Example: 8bit, 1word transfer, Internal clock)

12.6.3 8-bit transfer / receive mode

After setting the SIO control register to the 8-bit transmit/receive mode, write the data to be transmitted first to the data buffer registers (DBR). After that, enable the transmit/receive by setting SIOCR1<SIOS> to "1". When transmitting, the data are output from the SO pin at leading edges of the serial clock. When receiving, the data are input to the SI pin at the trailing edges of the serial clock. When the all receive is enabled, 8-bit data are transferred from the shift register to the data buffer register. An INTSIO interrupt is generated when the number of data words specified with the SIOCR2<BUF> has been transferred. Usually, read the receive data from the buffer register in the interrupt service. The data buffer register is used for both transmitting and receiving; therefore, always write the data to be transmitted after reading the all received data.

When the internal clock is used, a wait is initiated until the received data are read and the next transfer data are written. A wait will not be initiated if even one transfer data word has been written.

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

The transmit/receive operation is ended by clearing SIOCR1<SIOS> to "0" or setting SIOCR1<SIOINH> to "1" in INTSIO interrupt service program.

When SIOCR1<SIOS> is cleared, the current data are transferred to the buffer. After SIOCR1<SIOS> cleared, the transmitting/receiving is ended at the time that the final bit of the data has been transmitted. That the transmitting/receiving has ended can be determined from the status of SIOSR<SIOF>. SIOSR<SIOF> is cleared to "0" when the transmitting/receiving is ended.

When SIOCR1<SIOINH> is set, the transmit/receive operation is immediately ended and SIOSR<SIOF> is cleared to "0".

If it is necessary to change the number of words in external clock operation, SIOCR1<SIOS> should be cleared to "0", then SIOCR2<BUF> must be rewritten after confirming that SIOSR<SIOF> has been cleared to "0".

If it is necessary to change the number of words in internal clock, during automatic-wait operation which occurs after completion of transmit/receive operation, SIOCR2<BUF> must be rewritten before reading and writing of the receive/transmit data.

Note: The buffer contents are lost when the transfer mode is switched. If it should become necessary to switch the transfer mode, end receiving by clearing SIOCR1<SIOS> to "0", read the last data and then switch the transfer mode.
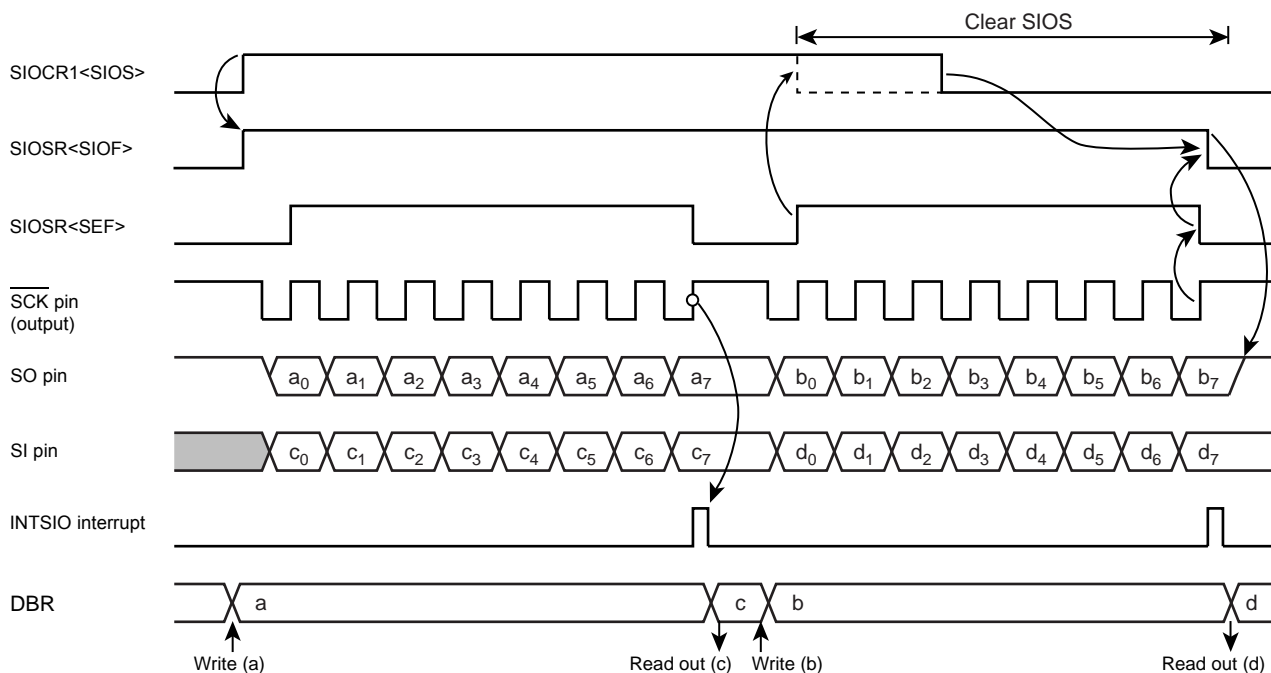
Figure 12-11 Transfer / Receive Mode (Example: 8bit, 1word transfer, Internal clock)

t_{SODH} = min 4/fc [s] (In the NORMAL1/2, IDLE1/2 modes)
t_{SODH} = min 4/fs [s] (In the SLOW1/2, SLEEP1/2 modes)
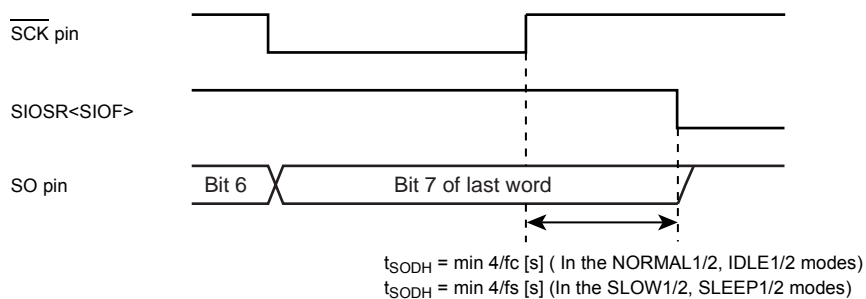
Figure 12-12 Transmitted Data Hold Time at End of Transfer / Receive

13. Asynchronous Serial interface (UART0)

13.1 Configuration
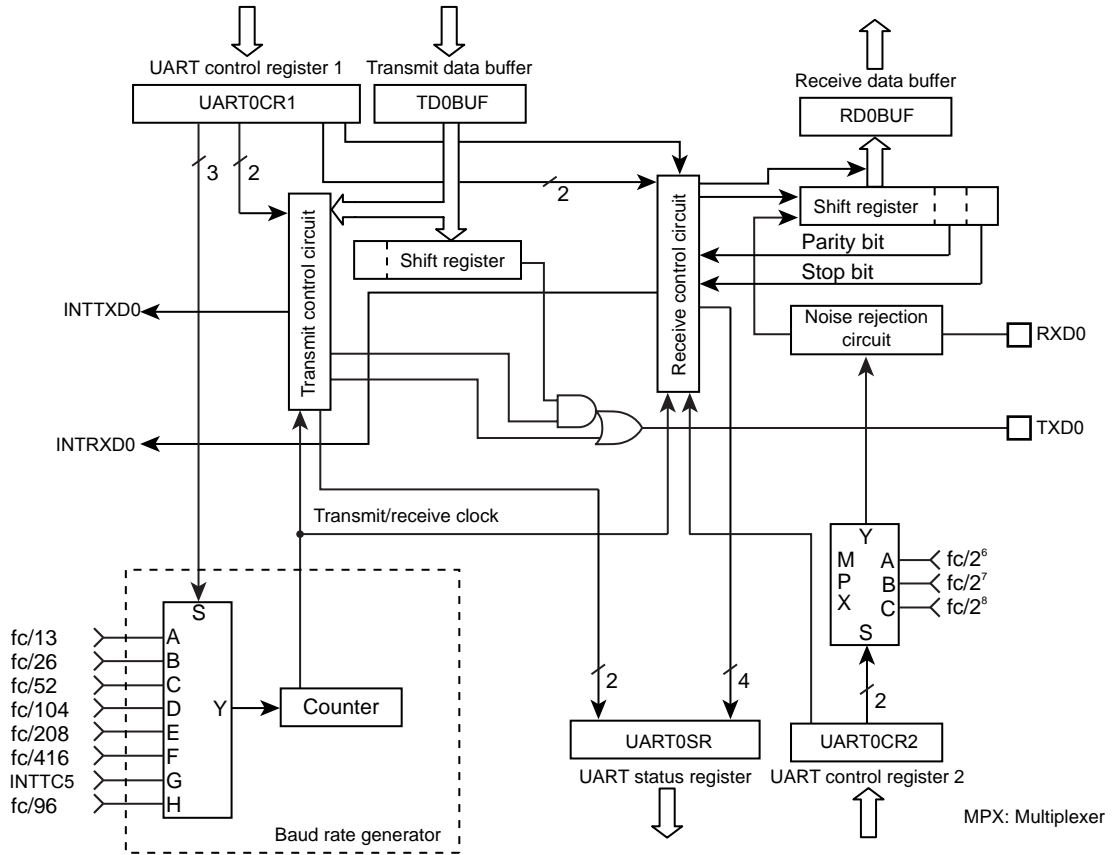
Figure 13-1 UART0 (Asynchronous Serial Interface)

13.2 Control

UART0 is controlled by the UART0 Control Registers (UART0CR1, UART0CR2). The operating status can be monitored using the UART status register (UART0SR).

UART0 Control Register1

UART0CR1 (0025H)	7	6	5	4	3	2	1	0	
	TXE	RXE	STBT	EVEN	PE	BRG			(Initial value: 0000 0000)

TXE	Transfer operation	0: Disable 1: Enable	Write only
RXE	Receive operation	0: Disable 1: Enable	
STBT	Transmit stop bit length	0: 1 bit 1: 2 bits	
EVEN	Even-numbered parity	0: Odd-numbered parity 1: Even-numbered parity	
PE	Parity addition	0: No parity 1: Parity	
BRG	Transmit clock select	000: fc/13 [Hz] 001: fc/26 010: fc/52 011: fc/104 100: fc/208 101: fc/416 110: TC5 (Input INTTC5) 111: fc/96	

Note 1: When operations are disabled by setting TXE and RXE bit to "0", the setting becomes valid when data transmit or receive complete. When the transmit data is stored in the transmit data buffer, the data are not transmitted. Even if data transmit is enabled, until new data are written to the transmit data buffer, the current data are not transmitted.

Note 2: The transmit clock and the parity are common to transmit and receive.

Note 3: UART0CR1<RXE> and UART0CR1<TXE> should be set to "0" before UART0CR1<BRG> is changed.

UART0 Control Register2

UART0CR2 (0026H)	7	6	5	4	3	2	1	0	
						RXDNC	STOPBR		(Initial value: **** *000)

RXDNC	Selection of RXD input noise rejection time	00: No noise rejection (Hysteresis input) 01: Rejects pulses shorter than 31/fc [s] as noise 10: Rejects pulses shorter than 63/fc [s] as noise 11: Rejects pulses shorter than 127/fc [s] as noise	Write only
STOPBR	Receive stop bit length	0: 1 bit 1: 2 bits	

Note: When UART0CR2<RXDNC> = "01", pulses longer than 96/fc [s] are always regarded as signals; when UART0CR2<RXDNC> = "10", longer than 192/fc [s]; and when UART0CR2<RXDNC> = "11", longer than 384/fc [s].

UART0 Status Register

UART0SR (0025H)	7	6	5	4	3	2	1	0	
	PERR	FERR	OERR	RBFL	TEND	TBEP			(Initial value: 0000 11**)

PERR	Parity error flag	0: No parity error 1: Parity error	Read only
FERR	Framing error flag	0: No framing error 1: Framing error	
OERR	Overrun error flag	0: No overrun error 1: Overrun error	
RBFL	Receive data buffer full flag	0: Receive data buffer empty 1: Receive data buffer full	
TEND	Transmit end flag	0: On transmitting 1: Transmit end	
TBEP	Transmit data buffer empty flag	0: Transmit data buffer full (Transmit data writing is finished) 1: Transmit data buffer empty	

Note: When an INTTXD is generated, TBEP flag is set to "1" automatically.

UART0 Receive Data Buffer

RD0BUF (0F9BH)	7	6	5	4	3	2	1	0	Read only
									(Initial value: 0000 0000)

UART0 Transmit Data Buffer

TD0BUF (0F9BH)	7	6	5	4	3	2	1	0	Write only
									(Initial value: 0000 0000)

13.3 Transfer Data Format

In UART0, an one-bit start bit (Low level), stop bit (Bit length selectable at high level, by UART0CR1<STBT>), and parity (Select parity in UART0CR1<PE>; even- or odd-numbered parity by UART0CR1<EVEN>) are added to the transfer data. The transfer data formats are shown as follows.

PE	STBT	Frame Length									
		1	2	3	8	9	10	11	12		
0	0	Start	Bit 0	Bit 1	– – – –	Bit 6	Bit 7	Stop 1			
0	1	Start	Bit 0	Bit 1	– – – –	Bit 6	Bit 7	Stop 1	Stop 2		
1	0	Start	Bit 0	Bit 1	– – – –	Bit 6	Bit 7	Parity	Stop 1		
1	1	Start	Bit 0	Bit 1	– – – –	Bit 6	Bit 7	Parity	Stop 1	Stop 2	

Figure 13-2 Transfer Data Format

Without parity / 1 STOP bit

With parity / 1 STOP bit

Without parity / 2 STOP bit

With parity / 2 STOP bit
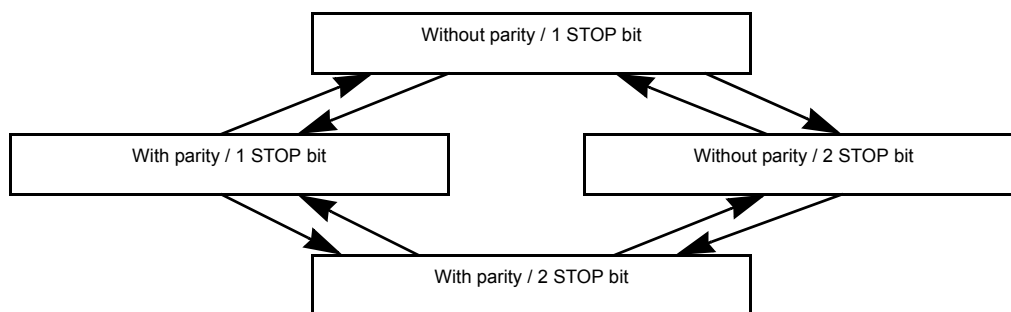
Figure 13-3 Caution on Changing Transfer Data Format

Note: In order to switch the transfer data format, perform transmit operations in the above Figure 13-3 sequence except for the initial setting.

13.4 Transfer Rate

The baud rate of UART0 is set of UART0CR1<BRG>. The example of the baud rate are shown as follows.

Table 13-1 Transfer Rate (Example)

BRG	Source Clock		
	16 MHz	8 MHz	4 MHz
000	76800 [baud]	38400 [baud]	19200 [baud]
001	38400	19200	9600
010	19200	9600	4800
011	9600	4800	2400
100	4800	2400	1200
101	2400	1200	600

When TC5 is used as the UART0 transfer rate (when UART0CR1<BRG> = "110"), the transfer clock and transfer rate are determined as follows:

Transfer clock [Hz] = TC5 source clock [Hz] / TTREG5 setting value

Transfer Rate [baud] = Transfer clock [Hz] / 16

13.5 Data Sampling Method

The UART0 receiver keeps sampling input using the clock selected by UART0CR1<BRG> until a start bit is detected in RXD0 pin input. RT clock starts detecting "L" level of the RXD0 pin. Once a start bit is detected, the start bit, data bits, stop bit(s), and parity bit are sampled at three times of RT7, RT8, and RT9 during one receiver clock interval (RT clock). (RT0 is the position where the bit supposedly starts.) Bit is determined according to majority rule (The data are the same twice or more out of three samplings).

(a) Without noise rejection circuit

(b) With noise rejection circuit
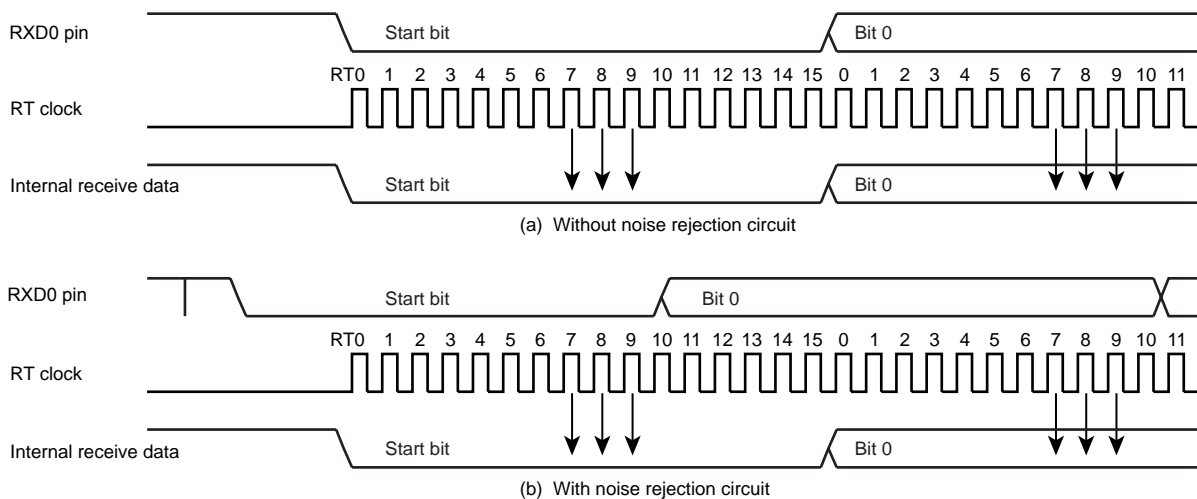
Figure 13-4 Data Sampling Method

13.6 STOP Bit Length

Select a transmit stop bit length (1 bit or 2 bits) by UART0CR1<STBT>.

13.7 Parity

Set parity / no parity by UART0CR1<PE> and set parity type (Odd- or Even-numbered) by UART0CR1<EVEN>.

13.8 Transmit/Receive Operation

13.8.1 Data Transmit Operation

Set UART0CR1<TXE> to "1". Read UART0SR to check UART0SR<TBEP> = "1", then write data in TD0BUF (Transmit data buffer). Writing data in TD0BUF zero-clears UART0SR<TBEP>, transfers the data to the transmit shift register and the data are sequentially output from the TXD0 pin. The data output include a one-bit start bit, stop bits whose number is specified in UART0CR1<STBT> and a parity bit if parity addition is specified. Select the data transfer baud rate using UART0CR1<BRG>. When data transmit starts, transmit buffer empty flag UART0SR<TBEP> is set to "1" and an INTTXD0 interrupt is generated.

While UART0CR1<TXE> = "0" and from when "1" is written to UART0CR1<TXE> to when send data are written to TD0BUF, the TXD0 pin is fixed at high level.
When transmitting data, first read UART0SR, then write data in TD0BUF. Otherwise, UART0SR<TBEP> is not zero-cleared and transmit does not start.

13.8.2 Data Receive Operation

Set UART0CR1<RXE> to "1". When data are received via the RXD0 pin, the receive data are transferred to RD0BUF (Receive data buffer). At this time, the data transmitted includes a start bit and stop bit(s) and a parity bit if parity addition is specified. When stop bit(s) are received, data only are extracted and transferred to RD0BUF (Receive data buffer). Then the receive buffer full flag UART0SR<RBFL> is set and an INTRXD0 interrupt is generated. Select the data transfer baud rate using UART0CR1<BRG>.

If an overrun error (OERR) occurs when data are received, the data are not transferred to RD0BUF (Receive data buffer) but discarded; data in the RD0BUF are not affected.

Note: When a receive operation is disabled by setting UART0CR1<RXE> bit to "0", the setting becomes valid when data receive is completed. However, if a framing error occurs in data receive, the receive-disabling setting may not become valid. If a framing error occurs, be sure to perform a re-receive operation.

13.9 Status Flag

13.9.1 Parity Error

When parity determined using the receive data bits differs from the received parity bit, the parity error flag UART0SR<PERR> is set to "1". The UART0SR<PERR> is cleared to "0" when the RD0BUF is read after reading the UART0SR.

RXD0 pin		Parity	Stop	
Shift register	xxxx0**	pxxxx0*	1pxxxx0
UART0SR<PERR>			After reading UART0SR then RD0BUF clears PERR.
INTRXD0 interrupt			
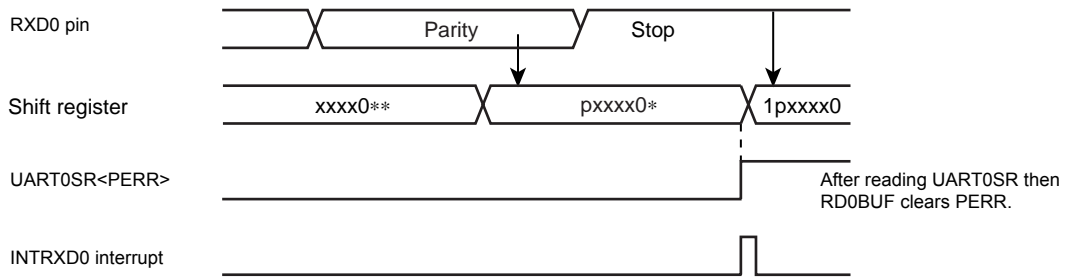
Figure 13-5 Generation of Parity Error

13.9.2 Framing Error

When "0" is sampled as the stop bit in the receive data, framing error flag UART0SR<FERR> is set to "1". The UART0SR<FERR> is cleared to "0" when the RD0BUF is read after reading the UART0SR.

RXD0 pin		Final bit	Stop	
Shift register	xxx0**	xxxx0*	0xxxx0
UART0SR<FERR>			After reading UART0SR then RD0BUF clears FERR.
INTRXD0 interrupt			
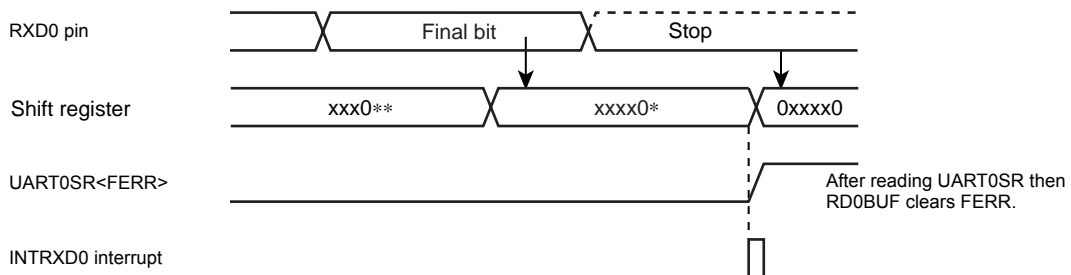
Figure 13-6 Generation of Framing Error

13.9.3 Overrun Error

When all bits in the next data are received while unread data are still in RD0BUF, overrun error flag UART0SR<OERR> is set to "1". In this case, the receive data is discarded; data in RD0BUF are not affected. The UART0SR<OERR> is cleared to "0" when the RD0BUF is read after reading the UART0SR.
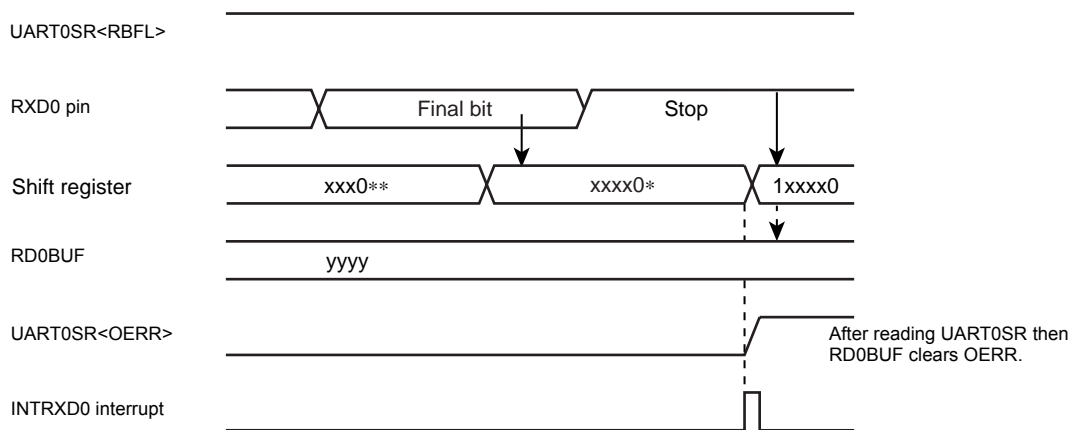
Figure 13-7 Generation of Overrun Error

Note: Receive operations are disabled until the overrun error flag UART0SR<OERR> is cleared.

13.9.4 Receive Data Buffer Full

Loading the received data in RD0BUF sets receive data buffer full flag UART0SR<RBFL> to "1". The UART0SR<RBFL> is cleared to "0" when the RD0BUF is read after reading the UART0SR.
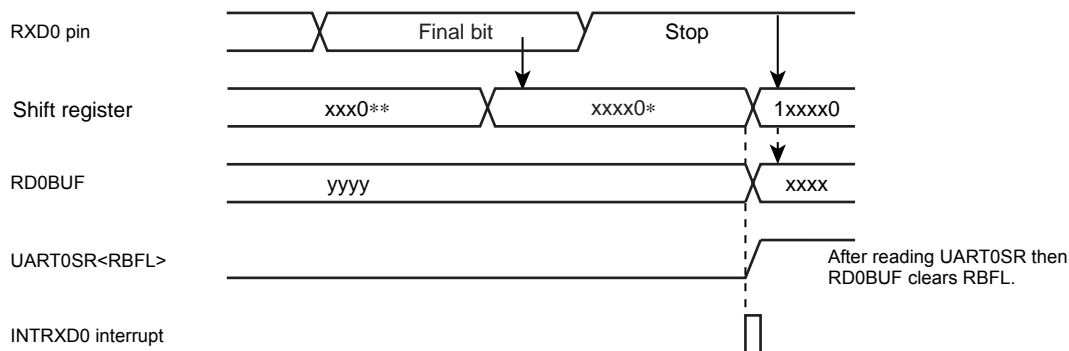
Figure 13-8 Generation of Receive Data Buffer Full

Note: If the overrun error flag UART0SR<OERR> is set during the period between reading the UART0SR and reading the RD0BUF, it cannot be cleared by only reading the RD0BUF. Therefore, after reading the RD0BUF, read the UART0SR again to check whether or not the overrun error flag which should have been cleared still remains set.

13.9.5 Transmit Data Buffer Empty

When no data is in the transmit buffer TD0BUF, that is, when data in TD0BUF are transferred to the transmit shift register and data transmit starts, transmit data buffer empty flag UART0SR<TBEP> is set to "1". The UART0SR<TBEP> is cleared to "0" when the TD0BUF is written after reading the UART0SR.
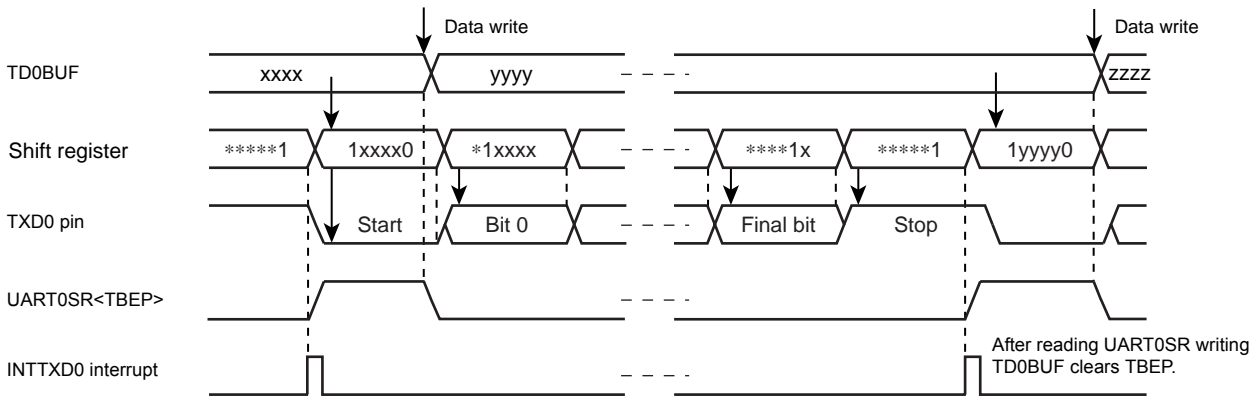
Figure 13-9 Generation of Transmit Data Buffer Empty

13.9.6 Transmit End Flag

When data are transmitted and no data is in TD0BUF (UART0SR<TBEP> = "1"), transmit end flag UART0SR<TEND> is set to "1". The UART0SR<TEND> is cleared to "0" when the data transmit is started after writing the TD0BUF.
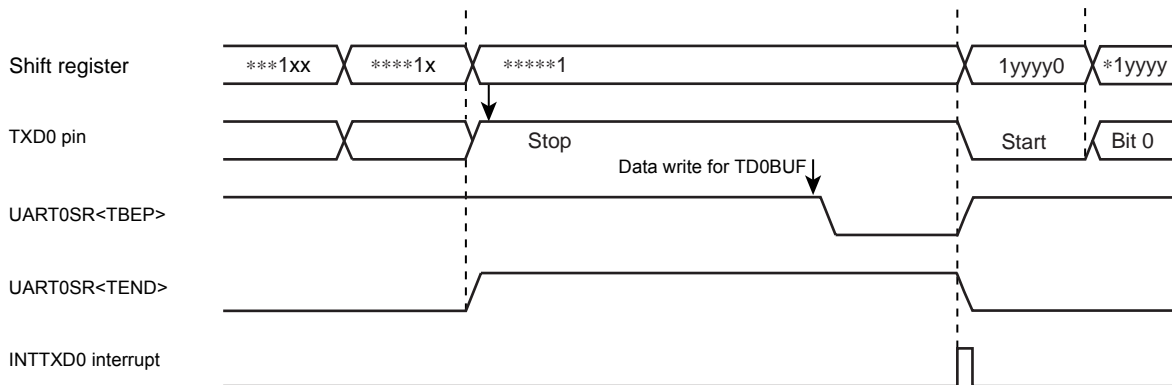
Figure 13-10 Generation of Transmit End Flag and Transmit Data Buffer Empty

14. Asynchronous Serial interface (UART1)

14.1 Configuration
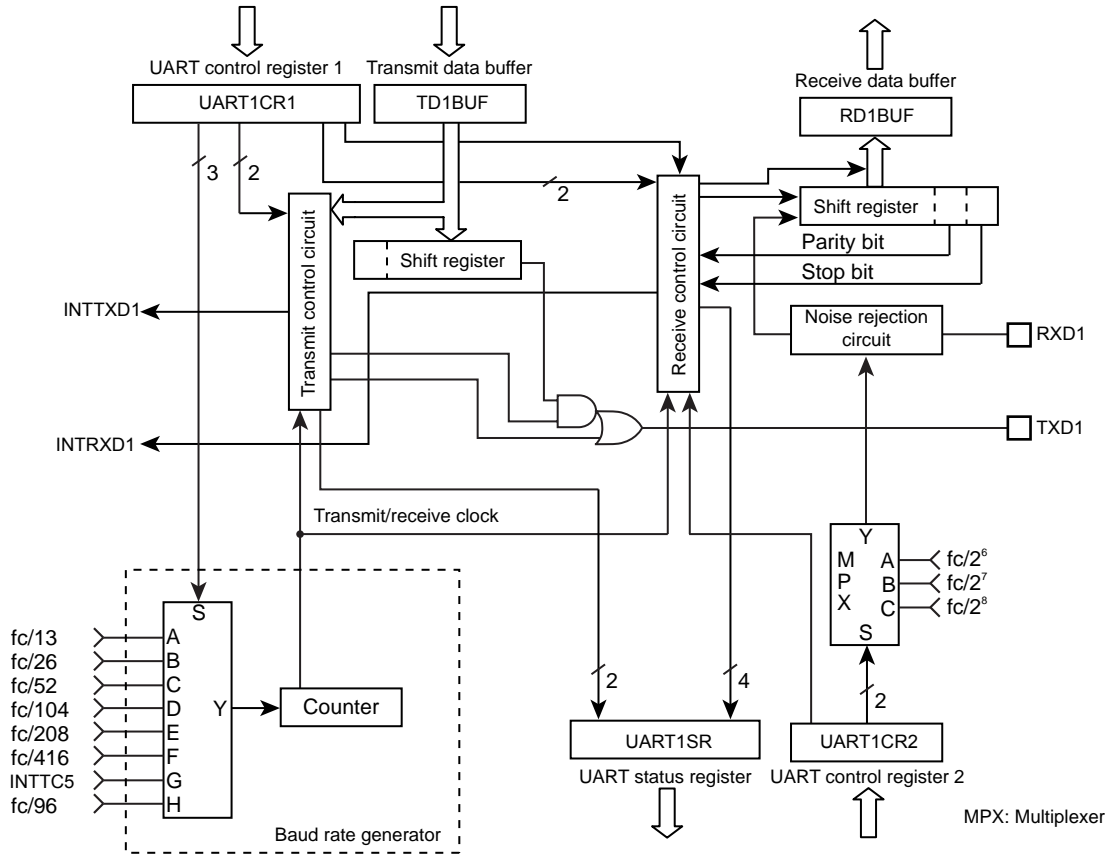
Figure 14-1 UART1 (Asynchronous Serial Interface)

14.2 Control

UART1 is controlled by the UART1 Control Registers (UART1CR1, UART1CR2). The operating status can be monitored using the UART status register (UART1SR).

UART1 Control Register1

UART1CR1	7	6	5	4	3	2	1	0	
(000EH)	TXE	RXE	STBT	EVEN	PE		BRG		(Initial value: 0000 0000)

TXE	Transfer operation	0: Disable 1: Enable	Write only
RXE	Receive operation	0: Disable 1: Enable	
STBT	Transmit stop bit length	0: 1 bit 1: 2 bits	
EVEN	Even-numbered parity	0: Odd-numbered parity 1: Even-numbered parity	
PE	Parity addition	0: No parity 1: Parity	
BRG	Transmit clock select	000: fc/13 [Hz] 001: fc/26 010: fc/52 011: fc/104 100: fc/208 101: fc/416 110: TC5 (Input INTTC5) 111: fc/96	

Note 1: When operations are disabled by setting TXE and RXE bit to "0", the setting becomes valid when data transmit or receive complete. When the transmit data is stored in the transmit data buffer, the data are not transmitted. Even if data transmit is enabled, until new data are written to the transmit data buffer, the current data are not transmitted.

Note 2: The transmit clock and the parity are common to transmit and receive.

Note 3: UART1CR1<RXE> and UART1CR1<TXE> should be set to "0" before UART1CR1<BRG> is changed.

UART1 Control Register2

UART1CR2	7	6	5	4	3	2	1	0	
(000FH)						RXDNC		STOPBR	(Initial value: **** *000)

RXDNC	Selection of RXD input noise rejection time	00: No noise rejection (Hysteresis input) 01: Rejects pulses shorter than 31/fc [s] as noise 10: Rejects pulses shorter than 63/fc [s] as noise 11: Rejects pulses shorter than 127/fc [s] as noise	Write only
STOPBR	Receive stop bit length	0: 1 bit 1: 2 bits	

Note: When UART1CR2<RXDNC> = "01", pulses longer than 96/fc [s] are always regarded as signals; when UART1CR2<RXDNC> = "10", longer than 192/fc [s]; and when UART1CR2<RXDNC> = "11", longer than 384/fc [s].

UART1 Status Register

UART1SR	7	6	5	4	3	2	1	0	
(000EH)	PERR	FERR	OERR	RBFL	TEND	TBEP			(Initial value: 0000 11**)

PERR	Parity error flag	0:	No parity error	Read only
		1:	Parity error	
FERR	Framing error flag	0:	No framing error	
		1:	Framing error	
OERR	Overrun error flag	0:	No overrun error	
		1:	Overrun error	
RBFL	Receive data buffer full flag	0:	Receive data buffer empty	
		1:	Receive data buffer full	
TEND	Transmit end flag	0:	On transmitting	
		1:	Transmit end	
TBEP	Transmit data buffer empty flag	0:	Transmit data buffer full (Transmit data writing is finished)	
		1:	Transmit data buffer empty	

Note: When an INTTXD is generated, TBEP flag is set to "1" automatically.

UART1 Receive Data Buffer

RD1BUF	7	6	5	4	3	2	1	0	Read only
(0F9CH)									(Initial value: 0000 0000)

UART1 Transmit Data Buffer

TD1BUF	7	6	5	4	3	2	1	0	Write only
(0F9CH)									(Initial value: 0000 0000)

14.3 Transfer Data Format

In UART1, an one-bit start bit (Low level), stop bit (Bit length selectable at high level, by UART1CR1<STBT>), and parity (Select parity in UART1CR1<PE>; even- or odd-numbered parity by UART1CR1<EVEN>) are added to the transfer data. The transfer data formats are shown as follows.

PE	STBT	Frame Length									
		1	2	3		8	9	10	11	12	
0	0	Start	Bit 0	Bit 1	– – – -	Bit 6	Bit 7	Stop 1			
0	1	Start	Bit 0	Bit 1	– – – -	Bit 6	Bit 7	Stop 1	Stop 2		
1	0	Start	Bit 0	Bit 1	– – – -	Bit 6	Bit 7	Parity	Stop 1		
1	1	Start	Bit 0	Bit 1	– – – -	Bit 6	Bit 7	Parity	Stop 1	Stop 2	

Figure 14-2 Transfer Data Format
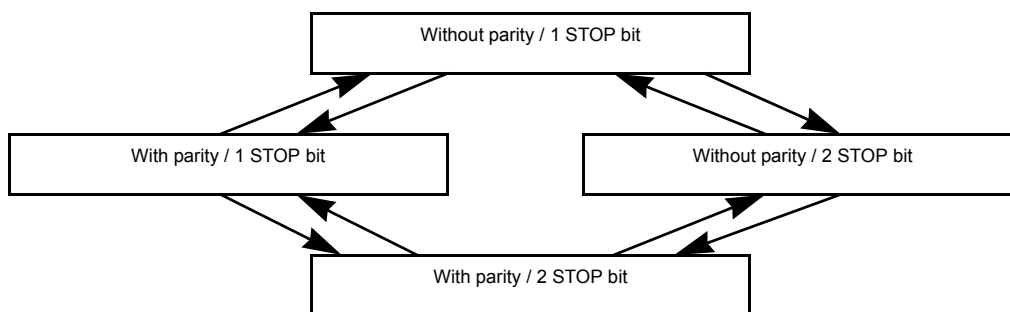
Figure 14-3 Caution on Changing Transfer Data Format

Note: In order to switch the transfer data format, perform transmit operations in the above Figure 14-3 sequence except for the initial setting.

14.4 Transfer Rate

The baud rate of UART1 is set of UART1CR1<BRG>. The example of the baud rate are shown as follows.

Table 14-1 Transfer Rate (Example)

BRG	Source Clock		
	16 MHz	8 MHz	4 MHz
000	76800 [baud]	38400 [baud]	19200 [baud]
001	38400	19200	9600
010	19200	9600	4800
011	9600	4800	2400
100	4800	2400	1200
101	2400	1200	600

When TC5 is used as the UART1 transfer rate (when UART1CR1<BRG> = "110"), the transfer clock and transfer rate are determined as follows:

Transfer clock [Hz] = TC5 source clock [Hz] / TTREG5 setting value

Transfer Rate [baud] = Transfer clock [Hz] / 16

14.5 Data Sampling Method

The UART1 receiver keeps sampling input using the clock selected by UART1CR1<BRG> until a start bit is detected in RXD1 pin input. RT clock starts detecting "L" level of the RXD1 pin. Once a start bit is detected, the start bit, data bits, stop bit(s), and parity bit are sampled at three times of RT7, RT8, and RT9 during one receiver clock interval (RT clock). (RT0 is the position where the bit supposedly starts.) Bit is determined according to majority rule (The data are the same twice or more out of three samplings).
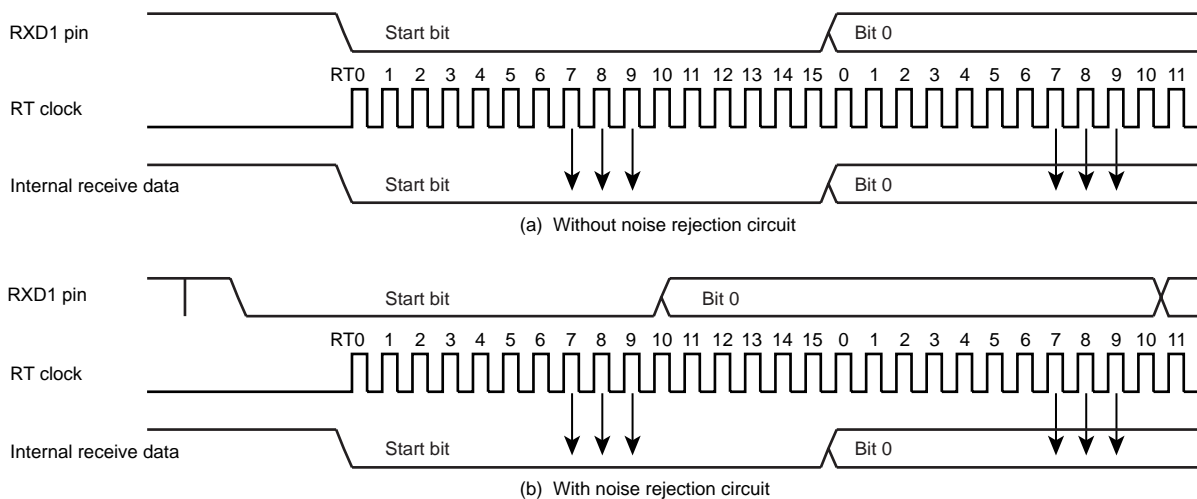
Figure 14-4 Data Sampling Method

14.6 STOP Bit Length

Select a transmit stop bit length (1 bit or 2 bits) by UART1CR1<STBT>.

14.7 Parity

Set parity / no parity by UART1CR1<PE> and set parity type (Odd- or Even-numbered) by UART1CR1<EVEN>.

14.8 Transmit/Receive Operation

14.8.1 Data Transmit Operation

Set UART1CR1<TXE> to "1". Read UART1SR to check UART1SR<TBEP> = "1", then write data in TD1BUF (Transmit data buffer). Writing data in TD1BUF zero-clears UART1SR<TBEP>, transfers the data to the transmit shift register and the data are sequentially output from the TXD1 pin. The data output include a one-bit start bit, stop bits whose number is specified in UART1CR1<STBT> and a parity bit if parity addition is specified. Select the data transfer baud rate using UART1CR1<BRG>. When data transmit starts, transmit buffer empty flag UART1SR<TBEP> is set to "1" and an INTTXD1 interrupt is generated.

While UART1CR1<TXE> = "0" and from when "1" is written to UART1CR1<TXE> to when send data are written to TD1BUF, the TXD1 pin is fixed at high level.
When transmitting data, first read UART1SR, then write data in TD1BUF. Otherwise, UART1SR<TBEP> is not zero-cleared and transmit does not start.

14.8.2 Data Receive Operation

Set UART1CR1<RXE> to "1". When data are received via the RXD1 pin, the receive data are transferred to RD1BUF (Receive data buffer). At this time, the data transmitted includes a start bit and stop bit(s) and a parity bit if parity addition is specified. When stop bit(s) are received, data only are extracted and transferred to RD1BUF (Receive data buffer). Then the receive buffer full flag UART1SR<RBFL> is set and an INTRXD1 interrupt is generated. Select the data transfer baud rate using UART1CR1<BRG>.

If an overrun error (OERR) occurs when data are received, the data are not transferred to RD1BUF (Receive data buffer) but discarded; data in the RD1BUF are not affected.

Note: When a receive operation is disabled by setting UART1CR1<RXE> bit to "0", the setting becomes valid when data receive is completed. However, if a framing error occurs in data receive, the receive-disabling setting may not become valid. If a framing error occurs, be sure to perform a re-receive operation.

14.9 Status Flag

14.9.1 Parity Error

When parity determined using the receive data bits differs from the received parity bit, the parity error flag UART1SR<PERR> is set to "1". The UART1SR<PERR> is cleared to "0" when the RD1BUF is read after reading the UART1SR.
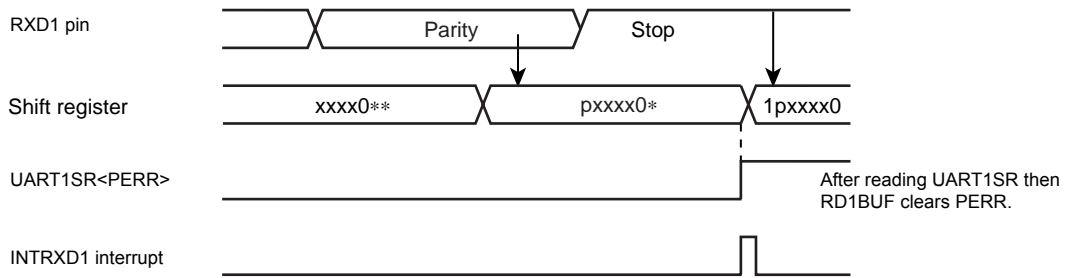
Figure 14-5 Generation of Parity Error

14.9.2 Framing Error

When "0" is sampled as the stop bit in the receive data, framing error flag UART1SR<FERR> is set to "1". The UART1SR<FERR> is cleared to "0" when the RD1BUF is read after reading the UART1SR.
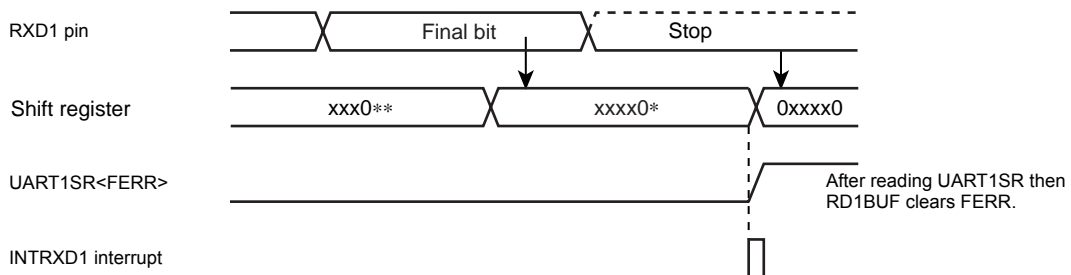
Figure 14-6 Generation of Framing Error

14.9.3 Overrun Error

When all bits in the next data are received while unread data are still in RD1BUF, overrun error flag UART1SR<OERR> is set to "1". In this case, the receive data is discarded; data in RD1BUF are not affected. The UART1SR<OERR> is cleared to "0" when the RD1BUF is read after reading the UART1SR.
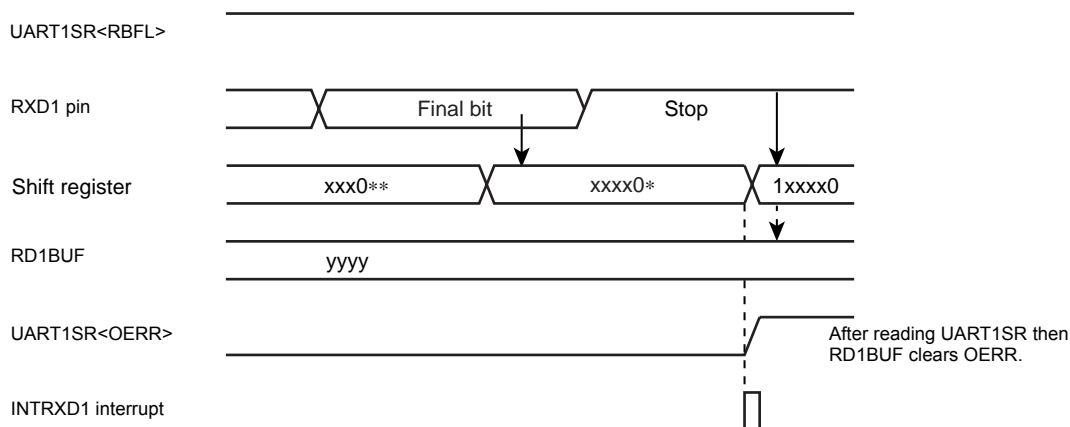
Figure 14-7 Generation of Overrun Error

Note: Receive operations are disabled until the overrun error flag UART1SR<OERR> is cleared.

14.9.4 Receive Data Buffer Full

Loading the received data in RD1BUF sets receive data buffer full flag UART1SR<RBFL> to "1". The UART1SR<RBFL> is cleared to "0" when the RD1BUF is read after reading the UART1SR.
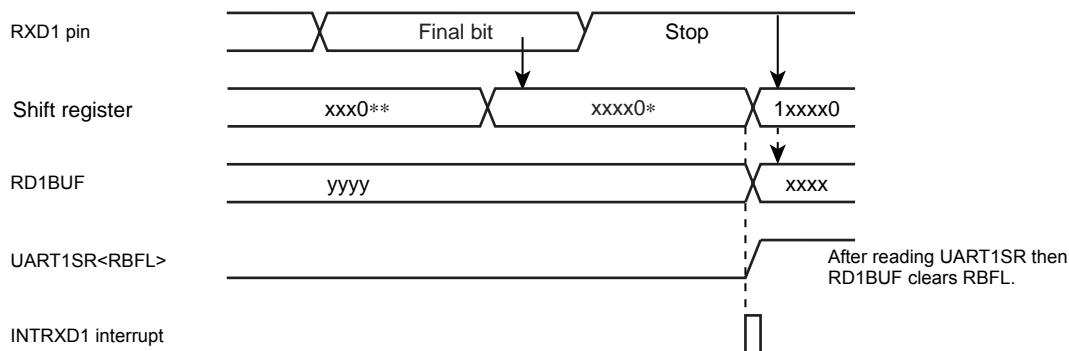
Figure 14-8 Generation of Receive Data Buffer Full

Note: If the overrun error flag UART1SR<OERR> is set during the period between reading the UART1SR and reading the RD1BUF, it cannot be cleared by only reading the RD1BUF. Therefore, after reading the RD1BUF, read the UART1SR again to check whether or not the overrun error flag which should have been cleared still remains set.

14.9.5 Transmit Data Buffer Empty

When no data is in the transmit buffer TD1BUF, that is, when data in TD1BUF are transferred to the transmit shift register and data transmit starts, transmit data buffer empty flag UART1SR<TBEP> is set to "1". The UART1SR<TBEP> is cleared to "0" when the TD1BUF is written after reading the UART1SR.
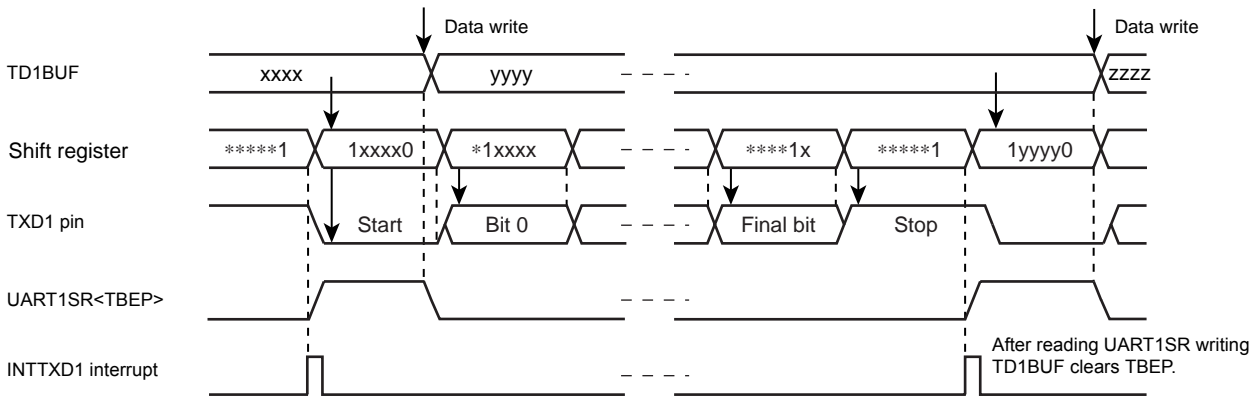
Figure 14-9 Generation of Transmit Data Buffer Empty

14.9.6 Transmit End Flag

When data are transmitted and no data is in TD1BUF (UART1SR<TBEP> = "1"), transmit end flag UART1SR<TEND> is set to "1". The UART1SR<TEND> is cleared to "0" when the data transmit is started after writing the TD1BUF.
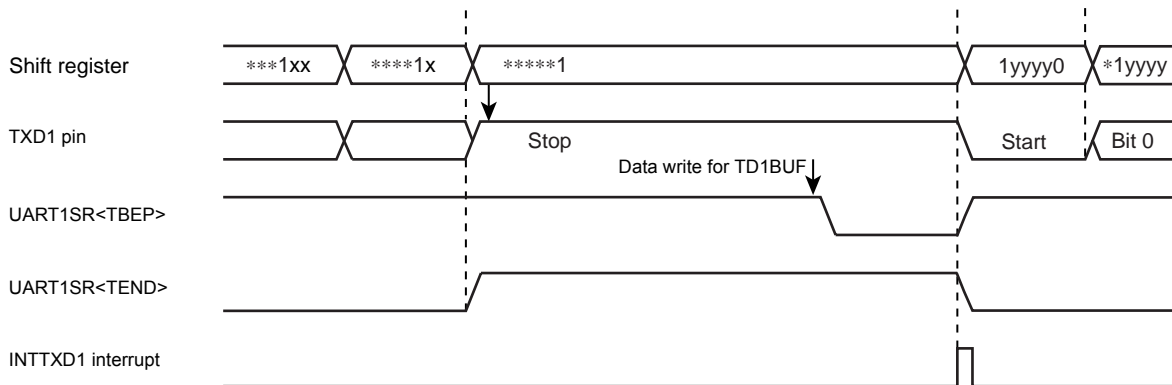
Figure 14-10 Generation of Transmit End Flag and Transmit Data Buffer Empty

15. Key-on Wakeup (KWU)

In the TMP86FM26UG, the STOP mode is released by not only P20($\overline{\text{INT5}}$/$\overline{\text{STOP}}$) pin but also four (STOP2 to STOP5) pins.

When the STOP mode is released by STOP2 to STOP5 pins, the $\overline{\text{STOP}}$ pin needs to be used.
In details, refer to the following section " 15.2 Control ".

15.1 Configuration
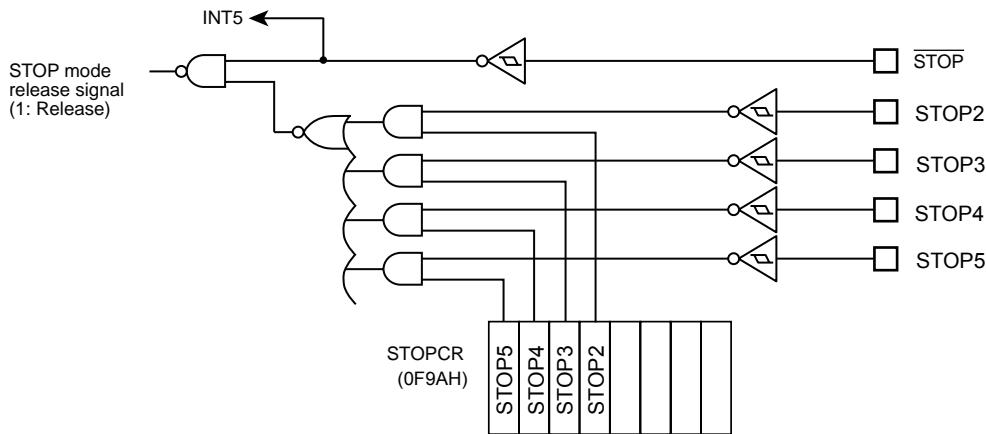
Figure 15-1 Key-on Wakeup Circuit

15.2 Control

STOP2 to STOP5 pins can controlled by Key-on Wakeup Control Register (STOPCR). It can be configured as enable/disable in 1-bit unit. When those pins are used for STOP mode release, configure corresponding I/O pins to input mode by I/O port register beforehand.

Key-on Wakeup Control Register

STOPCR	7	6	5	4	3	2	1	0	
(0F9AH)	STOP5	STOP4	STOP3	STOP2					(Initial value: 0000 ****)

STOP5	STOP mode released by STOP5	0:Disable 1:Enable	Write only
STOP4	STOP mode released by STOP4	0:Disable 1:Enable	Write only
STOP3	STOP mode released by STOP3	0:Disable 1:Enable	Write only
STOP2	STOP mode released by STOP2	0:Disable 1:Enable	Write only

15.3 Function

Stop mode can be entered by setting up the System Control Register (SYSCR1), and can be exited by detecting the "L" level on STOP2 to STOP5 pins, which are enabled by STOPCR, for releasing STOP mode (Note1).

Also, each level of the STOP2 to STOP5 pins can be confirmed by reading corresponding I/O port data register, check all STOP2 to STOP5 pins "H" that is enabled by STOPCR before the STOP mode is started (Note2).

Note 1: When the STOP mode released by the edge release mode (SYSCR1<RELM> = "0"), inhibit input from STOP2 to STOP5 pins by Key-on Wakeup Control Register (STOPCR) or must be set "H" level into STOP2 to STOP5 pins that are available input during STOP mode.

Note 2: When the $\overline{STOP}$ pin input is high or STOP2 to STOP5 pins input which is enabled by STOPCR is low, executing an instruction which starts STOP mode will not place in STOP mode but instead will immediately start the release sequence (Warm up).

Note 3: $\overline{STOP}$ pin doesn't have the control register such as STOPCR, so when STOP mode is released by STOP2 to STOP5 pins, $\overline{STOP}$ pin also should be used as STOP mode release function.

Note 4: When the STOP mode is released by STOP2 to STOP5 pins, the level of $\overline{STOP}$ pin should hold "L" level (Figure 15-2).

a) $\overline{STOP}$

b) In case of STOP2 to STOP5
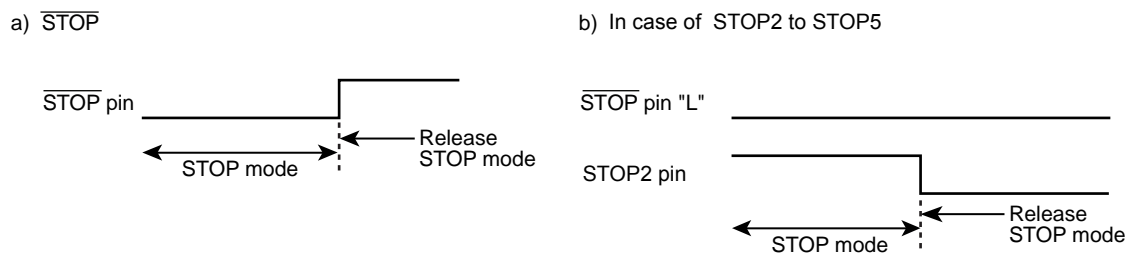
Figure 15-2 Priority of $\overline{STOP}$ pin and STOP2 to STOP5 pins

Table 15-1 Release level (edge) of STOP mode

Pin name	Release level (edge)	
	SYSCR1<RELM>="1" (Note2)	SYSCR1<RELM>="0"
$\overline{STOP}$	"H" level	Rising edge
STOP2	"L" level	Don't use (Note1)
STOP3	"L" level	Don't use (Note1)
STOP4	"L" level	Don't use (Note1)
STOP5	"L" level	Don't use (Note1)

TOSHIBA

16. LCD Driver

The TMP86FM26UG has a driver and control circuit to directly drive the liquid crystal device (LCD). The pins to be connected to LCD are as follows:

1. Segment output port 32 pins (SEG31 to SEG0)
2. Common output port4 pins (COM3 to COM0)

In addition, C0, C1, V1, V2, V3 pin are provided for the LCD driver's booster circuit.

The devices that can be directly driven is selectable from LCD of the following drive methods:

1. 1/4 Duty (1/3 Bias) LCD   Max 128 Segments(8 segments × 16 digits)
2. 1/3 Duty (1/3 Bias) LCD   Max 96 Segments(8 segments × 12 digits)
3. 1/2 Duty (1/2 Bias) LCD   Max 64 Segments(8 segments × 8 digits)
4. Static LCD   Max 32 Segments(8 segments × 4 digits)

16.1 Configuration
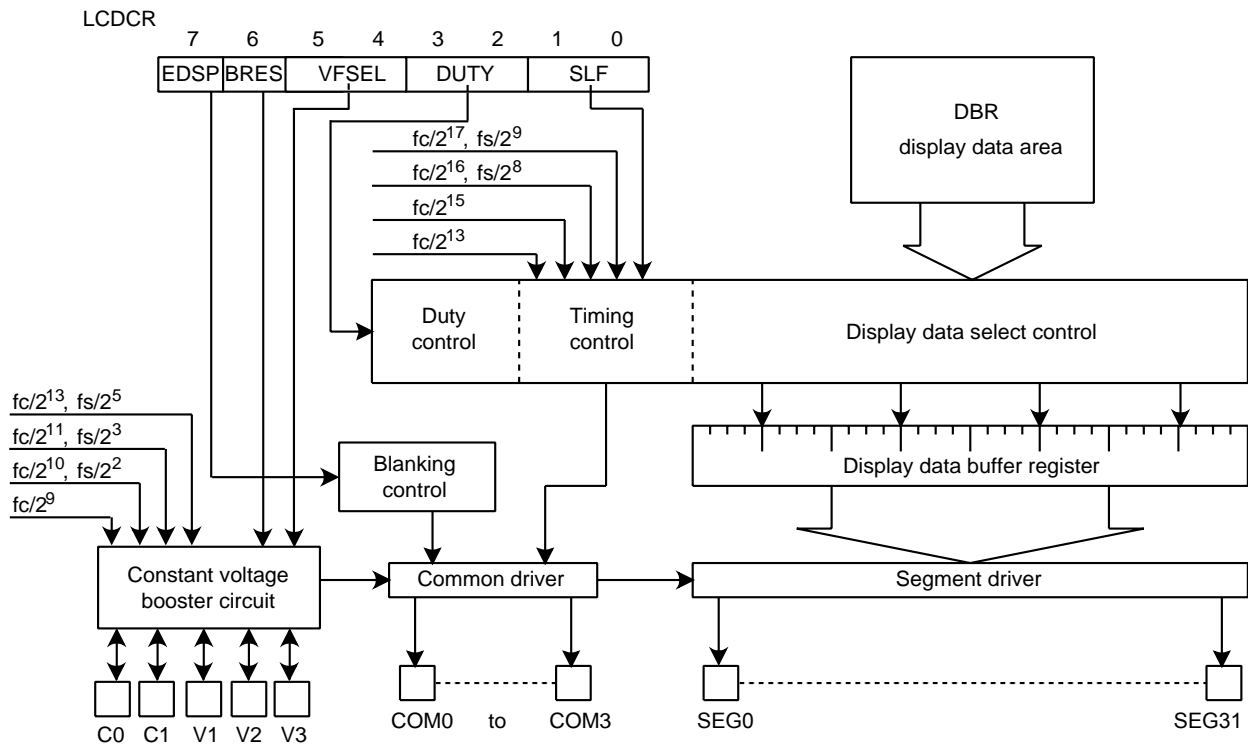
Figure 16-1 LCD Driver

Note: The LCD driver incorporates a dedicated divider circuit. Therefore, the break function of a debugger (development tool) will not stop LCD driver output.

16.2 Control

The LCD driver is controlled using the LCD control register (LCDCR). The LCD driver's display is enabled using the EDSP.

LCD Driver Control Register

LCDCR	7	6	5	4	3	2	1	0
(0028H)	EDSP	BRES	VFSEL		DUTY		SLF	

(Initial value: 0000 0000)

EDSP	LCD Display Control	0: Blanking 1: Enables LCD display (Blanking is released)				R/W
BRES	Booster circuit control	0: Disable (use divider resistance) 1: Enable				
VFSEL	Selection of boost frequency		NORMAL1/2, IDLE1/2 mode		SLOW1/2, SLEEP0/1/2 mode	
			DV7CK = 0	DV7CK = 1		
		00	$fc/2^{13}$	$fs/2^5$	$fs/2^5$	
		01	$fc/2^{11}$	$fs/2^3$	$fs/2^3$	
		10	$fc/2^{10}$	$fs/2^2$	$fs/2^2$	
		11	$fc/2^9$	$fc/2^9$	–	
DUTY	Selection of driving methods	00: 1/4 Duty (1/3 Bias) 01: 1/3 Duty (1/3 Bias) 10: 1/2 Duty (1/2 Bias) 11: Static				
SLF	Selection of LCD frame frequency		NORMAL1/2, IDLE1/2 mode		SLOW1/2, SLEEP0/1/2 mode	
			DV7CK = 0	DV7CK = 1		
		00	$fc/2^{17}$	$fs/2^9$	$fs/2^9$	
		01	$fc/2^{16}$	$fs/28$	$fs/2^8$	
		10	$fc/2^{15}$	$fc/2^{15}$	–	
		11	$fc/2^{13}$	$fc/2^{13}$	–	

Note 1: When <BRES>(Booster circuit control) is set to "0", $V_{DD} \geq V3 \geq V2 \geq V1 \geq V_{SS}$ should be satisfied.
When <BRES> is set to "1", 3.6 [V] $\geq V3 \geq V_{DD}$ should be satisfied.
If these conditions are not satisfied, it not only affects the quality of LCD display but also may damage the device due to over voltage of the port.

Note 2: When used as the booster circuit, bias should be composed to 1/3. Therefore, do not set LCDCR<DUTY> to "10" or "11" when the booster circuit is enable.

Note 3: Do not set SLF to "10" or "11" in SLOW1/2 modes.

Note 4: Do not set VFSEL to "11" SLOW1/2 modes.

16.2.1 LCD driving methods

As for LCD driving method, 4 types can be selected by LCDCR<DUTY>. The driving method is initialized in the initial program according to the LCD used.

(a) 1/4 Duty (1/3 Bias)

(b) 1/3 Duty (1/3 Bias)

(c) 1/2 Duty (1/2 Bias)

(d) Static
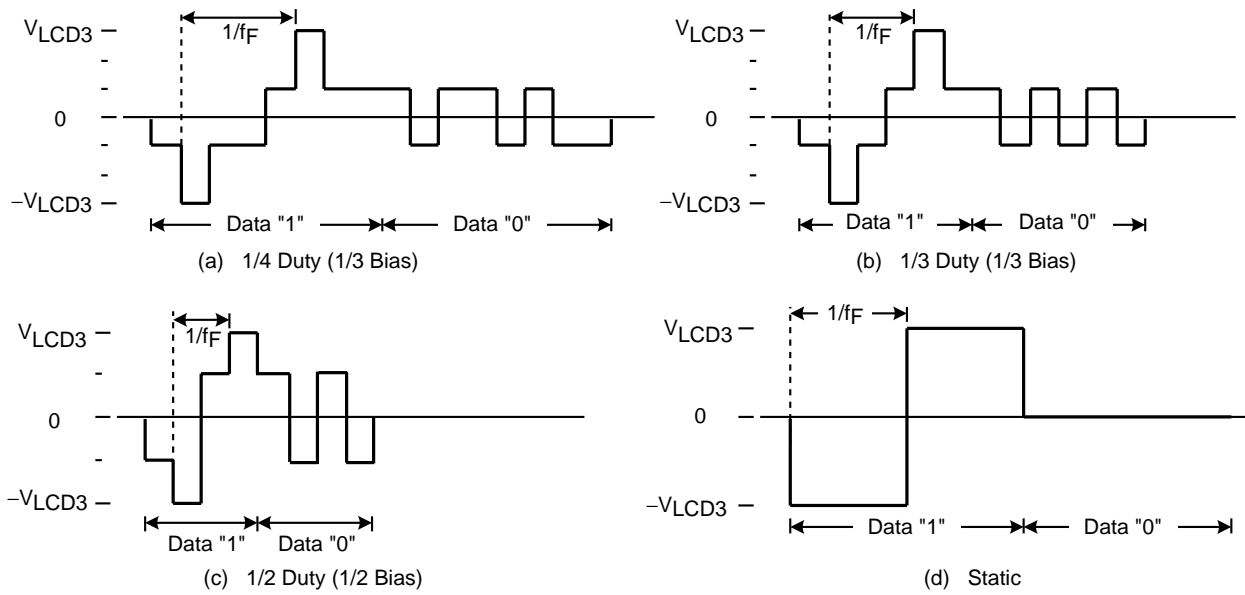
Note 1: f_F: Frame frequency

Note 2: V_{LCD3}: LCD drive voltage

Figure 16-2 LCD Drive Waveform (COM-SEG pins)

16.2.2 Frame frequency

Frame frequency (f_F) is set according to driving method and base frequency as shown in the following Table 16-1. The base frequency is selected by LCDCR<SLF> according to the frequency fc and fs of the basic clock to be used.

Table 16-1 Setting of LCD Frame Frequency

(a) At the single clock mode. At the dual clock mode (DV7CK = 0).

SLF	Base frequency [Hz]	Frame frequency [Hz]			
		1/4 Duty	1/3 Duty	1/2 Duty	Static
00	$\dfrac{fc}{2^{17}}$	$\dfrac{fc}{2^{17}}$	$\dfrac{4}{3} \cdot \dfrac{fc}{2^{17}}$	$\dfrac{4}{2} \cdot \dfrac{fc}{2^{17}}$	$\dfrac{fc}{2^{17}}$
	(fc = 16 MHz)	122	163	244	122
	(fc = 8 MHz)	61	81	122	61
01	$\dfrac{fc}{2^{16}}$	$\dfrac{fc}{2^{16}}$	$\dfrac{4}{3} \cdot \dfrac{fc}{2^{16}}$	$\dfrac{4}{2} \cdot \dfrac{fc}{2^{16}}$	$\dfrac{fc}{2^{16}}$
	(fc = 8 MHz)	122	163	244	122
	(fc = 4 MHz)	61	81	122	61
10	$\dfrac{fc}{2^{15}}$	$\dfrac{fc}{2^{15}}$	$\dfrac{4}{3} \cdot \dfrac{fc}{2^{15}}$	$\dfrac{4}{2} \cdot \dfrac{fc}{2^{15}}$	$\dfrac{fc}{2^{15}}$
	(fc = 4 MHz)	122	163	244	122
	(fc = 2 MHz)	61	81	122	61
11	$\dfrac{fc}{2^{13}}$	$\dfrac{fc}{2^{13}}$	$\dfrac{4}{3} \cdot \dfrac{fc}{2^{13}}$	$\dfrac{4}{2} \cdot \dfrac{fc}{2^{13}}$	$\dfrac{fc}{2^{13}}$
	(fc = 1 MHz)	122	163	244	122

Note: fc: High-frequency clock [Hz]

Table 16-2

(b) At the dual clock mode (DV7CK = 1 or SYSCK = 1)

SLF	Base frequency [Hz]	Frame frequency [Hz]			
		1/4 Duty	1/3 Duty	1/2 Duty	Static
00	$\dfrac{fs}{2^{9}}$	$\dfrac{fs}{2^{9}}$	$\dfrac{4}{3} \cdot \dfrac{fs}{2^{9}}$	$\dfrac{4}{2} \cdot \dfrac{fs}{2^{9}}$	$\dfrac{fs}{2^{9}}$
	(fs = 32.768 kHz)	64	85	128	64
01	$\dfrac{fs}{2^{8}}$	$\dfrac{fs}{2^{8}}$	$\dfrac{4}{3} \cdot \dfrac{fs}{2^{8}}$	$\dfrac{4}{2} \cdot \dfrac{fs}{2^{8}}$	$\dfrac{fs}{2^{8}}$
	(fs = 32.768 kHz)	128	171	256	128

Note: fs: Low-frequency clock [Hz]

16.2.3 Driving method for LCD driver

In the TMP86FM26UG, LCD driving voltages can be generated using either an internal booster circuit or an external resistor divider. This selection is made in LCDCR<BRES>.

16.2.3.1 When using the booster circuit (LCDCR<BRES>="1")

When the reference voltage is connected to the V1 pin, the booster circuit boosts the reference voltage twofold (V2) or threefold (V3) to generate the output voltages for segment/common signals. When the reference voltage is connected to the V2 pin, it is reduced to 1/2 (V1) or boosted to 3/2 (V3). When the reference voltage is connected to the V3 pin, it is reduced to 1/3 (V1) or 2/3 (V2).

LCDCR<VFSEL> is used to select the reference frequency in the booster circuit. The faster the boosting frequency, the higher the segment/common drive capability, but power consumption is increased. Conversely, the slower the boosting frequency, the lower the segment/common drive capability, but power consumption is reduced. If the drive capability is insufficient, the LCD may not be displayed clearly. Therefore, select an optimum boosting frequency for the LCD panel to be used.

Table 16-3 shows the V3 pin current capacity and boosting frequency.

Note: When used as the booster circuit, bias should be composed to 1/3. Therefore, do not set LCDCR<DUTY> to "10" or "11" when the booster circuit is enable (LCDCR<BRES>="1").

Keep the following condition.

$V_3 \geq V_{DD}$
$V_1 = 1/3 \times V3$
$C = 0.1$ to $0.47\ \mu F$

VDD
V3
V2
V1
C1
C0
VSS
C
C
C
Reference voltage

a) Reference pin = V1

Keep the following condition.

$V_3 \geq V_{DD}$
$V_2 = 2/3 \times V3$
$C = 0.1$ to $0.47\ \mu F$

VDD
V3
V2
V1
C1
C0
VSS
C
C
C
C
Reference voltage
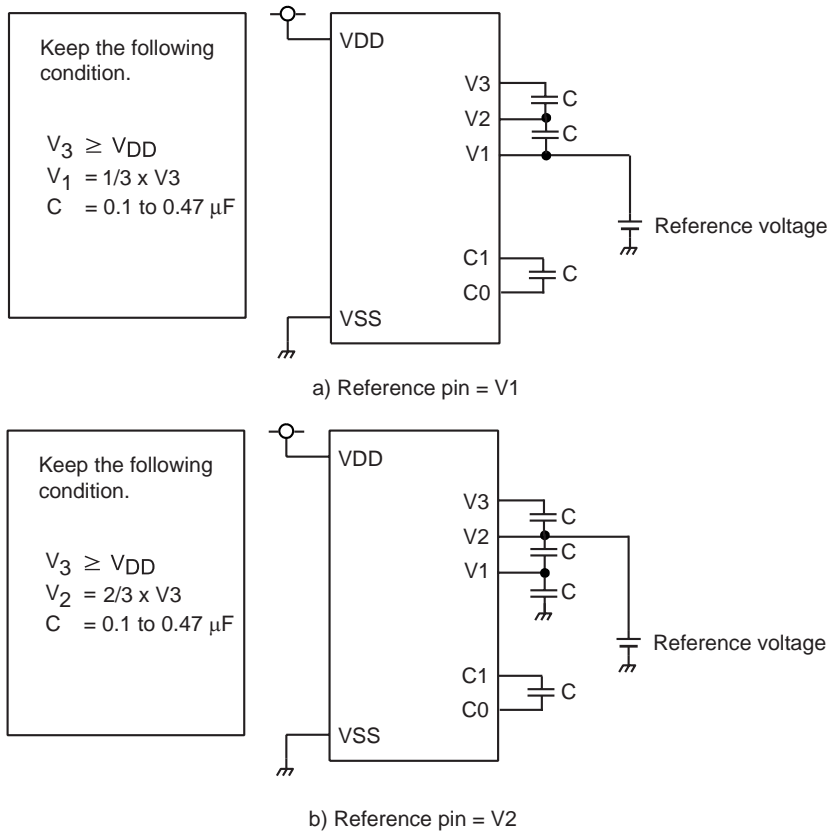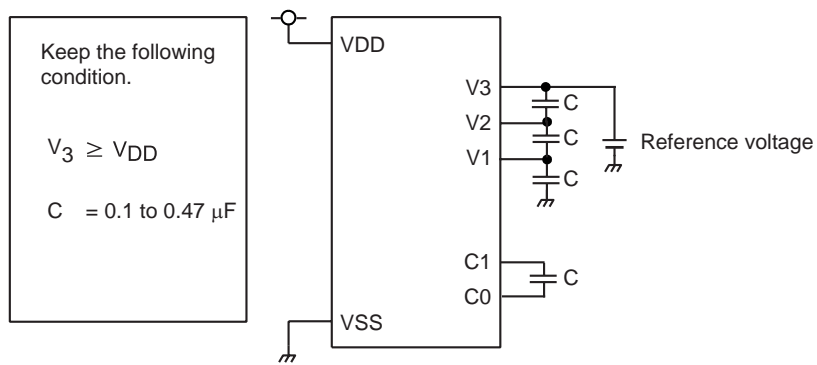
b) Reference pin = V2

c) Reference pin = V3

d) Reference pin = V3

Note 1: When the TMP86FM26UG uses the booster circuit to drive the LCD, the power supply and capacitor for the booster circuit should be connected as shown above.

Note 2: When the reference voltage is connected to a pin other than V1, add a capacitor between V1 and GND.

Figure 16-3 Connection Examples When Using the Booster Circuit (LCDCR<BRES> = "1")

Table 16-3 V3 Pin Current Capacity and Boosting Frequency (typ.)

VFSEL	Boosting frequency	fc = 16 MHz	fc = 8 MHz	fc = 4 MHz	fc = 32.768 MHz
00	$fc/2^{13}$ or $fs/2^5$	−37 mV/ μA	−80 mV/ μA	−138 mV/ μA	−76 mV/ μA
01	$fc/2^{11}$ or $fs/2^3$	−19 mV/ μA	−24 mV/ μA	−37 mV/ μA	−23 mV/ μA
10	$fc/2^{10}$ or $fs/2^2$	−17 mV/ μA	−19 mV/ μA	−24 mV/ μA	−18 mV/ μA
11	$fc/2^9$	−16 mV/ μA	−17 mV/ μA	−19 mV/ μA	−

Note 1: The current capacity is the amount of voltage that falls per 1μA.

Note 2: The boosting frequency should be selected depending on your LCD panel.

Note 3: For the reference pin V1 or V2, a current capacity ten times larger than the above is recommended to ensure stable operation.

For example, when the boosting frequency is $fc/2^9$ (at fc = 8 MHz), −1.7 mV/ μA or more is recommended for the current capacity of the reference pin V1.

16.2.3.2 When using an external resistor divider (LCDCR<BRES>="0")

When an external resistor divider is used, the voltage of an external power supply is divided and input on V1, V2, and V3 to generate the output voltages for segment/common signals.

The smaller the external resistor value, the higher the segment/common drive capability, but power consumption is increased. Conversely, the larger the external resistor value, the lower the segment/common drive capability, but power consumption is reduced. If the drive capability is insufficient, the LCD may not be displayed clearly. Therefore, select an optimum resistor value for the LCD panel to be used.
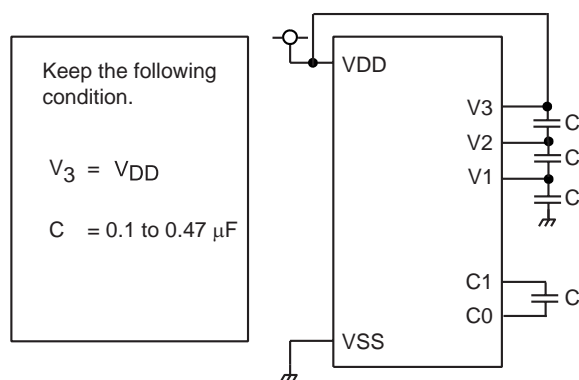

Keep the following conditon.
$$V_{DD} \geq V_3 \geq V_2 \geq V_1 \geq V_{SS}$$
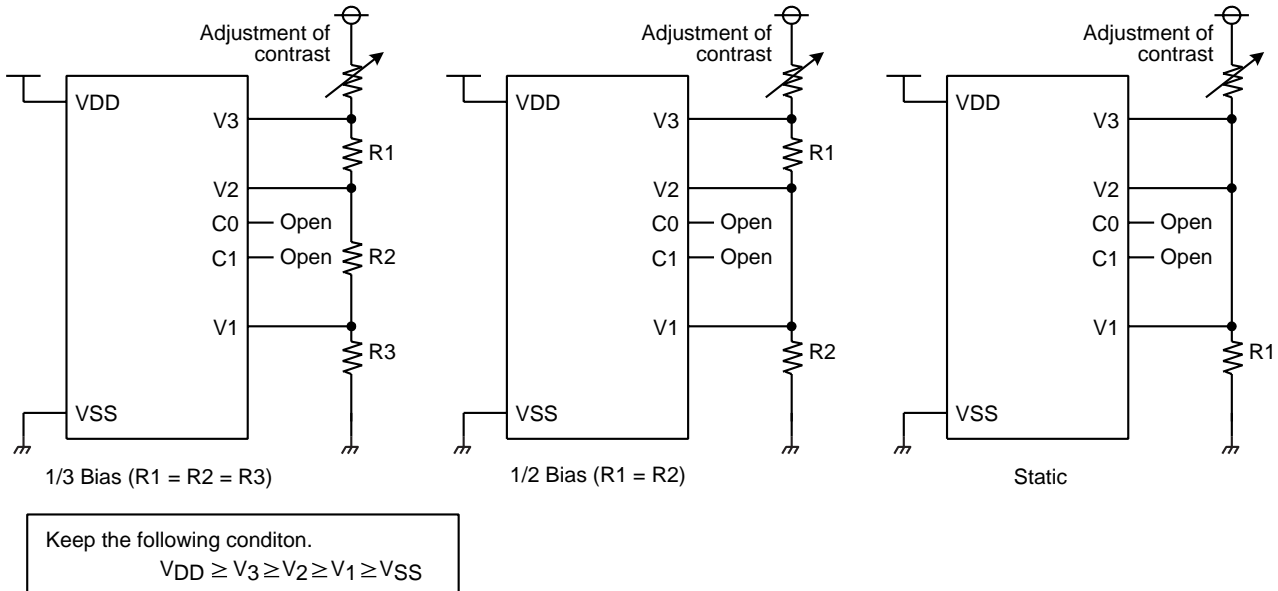
Figure 16-4 Connection Examples When Using an External Resistor Divider
(LCDCR<BRES> = "0")

16.3 LCD Display Operation

16.3.1 Display data setting

Display data is stored to the display data area (assigned to address 0F80H to 0F8FH, 16bytes) in the DBR. The display data which are stored in the display data area is automatically read out and sent to the LCD driver by the hardware. The LCD driver generates the segment signal and common signal according to the display data and driving method. Therefore, display patterns can be changed by only over writing the contents of display data area by the program. Table 16-5 shows the correspondence between the display data area and SEG/COM pins.

LCD light when display data is "1" and turn off when "0". According to the driving method of LCD, the number of pixels which can be driven becomes different, and the number of bits in the display data area which is used to store display data also becomes different.

Therefore, the bits which are not used to store display data as well as the data buffer which corresponds to the addresses not connected to LCD can be used to store general user process data (see Table 16-4).

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

Table 16-4 Driving Method and Bit for Display Data

Driving methods	Bit 7/3	Bit 6/2	Bit 5/1	Bit 4/0
1/4 Duty	COM3	COM2	COM1	COM0
1/3 Duty	–	COM2	COM1	COM0
1/2 Duty	–	–	COM1	COM0
Static	–	–	–	COM0

Note: –: This bit is not used for display data

Table 16-5 LCD Display Data Area (DBR)

Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0F80H		SEG1				SEG0		
0F81H		SEG3				SEG2		
0F82H		SEG5				SEG4		
0F83H		SEG7				SEG6		
0F84H		SEG9				SEG8		
0F85H		SEG11				SEG10		
0F86H		SEG13				SEG12		
0F87H		SEG15				SEG14		
0F88H		SEG17				SEG16		
0F89H		SEG19				SEG18		
0F8AH		SEG21				SEG20		
0F8BH		SEG23				SEG22		
0F8CH		SEG25				SEG24		
0F8DH		SEG27				SEG26		
0F8EH		SEG29				SEG28		
0F8FH		SEG31				SEG30		
	COM3	COM2	COM1	COM0	COM3	COM2	COM1	COM0

16.3.2 Blanking

Blanking is enabled when EDSP is cleared to "0".

Blanking turns off LCD through outputting a GND level to SEG/COM pin.

When in STOP mode, EDSP is cleared to "0" and automatically blanked. To redisplay ICD after exiting STOP mode, it is necessary to set EDSP back to "1".

Note: During reset, the LCD segment outputs and LCD common outputs are fixed "0" level. But the multiplex terminal of input/output port and LCD segment output becomes high impedance. Therefore, when the reset input is long remarkably, ghost problem may appear in LCD display.

TOSHIBA

16.4 Control Method of LCD Driver

16.4.1 Initial setting

Figure 16-5 shows the flowchart of initialization.

Example : To operate a 1/4 duty LCD of 32 segments × 4 com-mons at frame frequency $fc/2^{16}$ [Hz], and booster frequency $fc/2^{13}$ [Hz]

```
LD          (LCDCR), 01000001B        ; Sets LCD driving method and frame frequency. Boost frequency

LD          (P*LCR), 0FFH             ; Sets segment output control register. (*; Port No.)

:           :

:           :                         ; Sets the initial value of display data.

LD          (LCDCR), 11000001B        ; Display enable
```
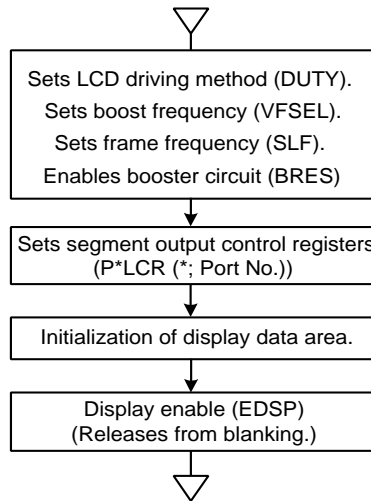

Figure 16-5 Initial Setting of LCD Driver

16.4.2 Store of display data

Generally, display data are prepared as fixed data in program memory (ROM) and stored in display data area by load command.

Example :To display using 1/4 duty LCD a numerical value which corresponds to the LCD data stored in data memory at address 80H (when pins COM and SEG are connected to LCD as in Figure 16-6), display data become as shown in Table 16-6.

```
        LD          A, (80H)
        ADD         A, TABLE-$-7
        LD          HL, 0F80H
        LD          W, (PC + A)
        LD          (HL), W
        RET
TABLE:  DB          11011111B, 00000110B,
                    11100011B, 10100111B,
                    00110110B, 10110101B,
                    11110101B, 00010111B,
                    11110111B, 10110111B
```

Note: DB is a byte data difinition instruction.

Figure 16-6 Example of COM, SEG Pin Connection (1/4 Duty)

Table 16-6 Example of Display Data (1/4 Duty)

No.	display	Display data	No.	display	Display data
0		11011111	5		10110101
1		00000110	6		11110101
2		11100011	7		00000111
3		10100111	8		11110111
4		00110110	9		10110111

Example 2: Table 16-6 shows an example of display data which are displayed using 1/2 duty LCD in the same way as Table 16-7. The connection between pins COM and SEG are the same as shown in Figure 16-7.
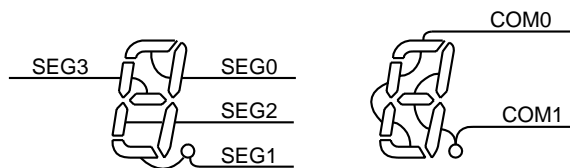
Figure 16-7 Example of COM, SEG Pin Connection

Table 16-7 Example of Display Data (1/2 Duty)

Number	Display data		Number	Display data	
	High order address	Low order address		High order address	Low order address
0	**01**11	**01**11	5	**11**10	**01**01
1	**00**10	**00**10	6	**11**11	**01**01
2	**10**01	**01**11	7	**01**10	**00**11
3	**10**10	**01**11	8	**11**11	**01**11
4	**11**10	**00**10	9	**11**10	**01**11

Note: *: Don't care

16.4.3 Example of LCD drive output
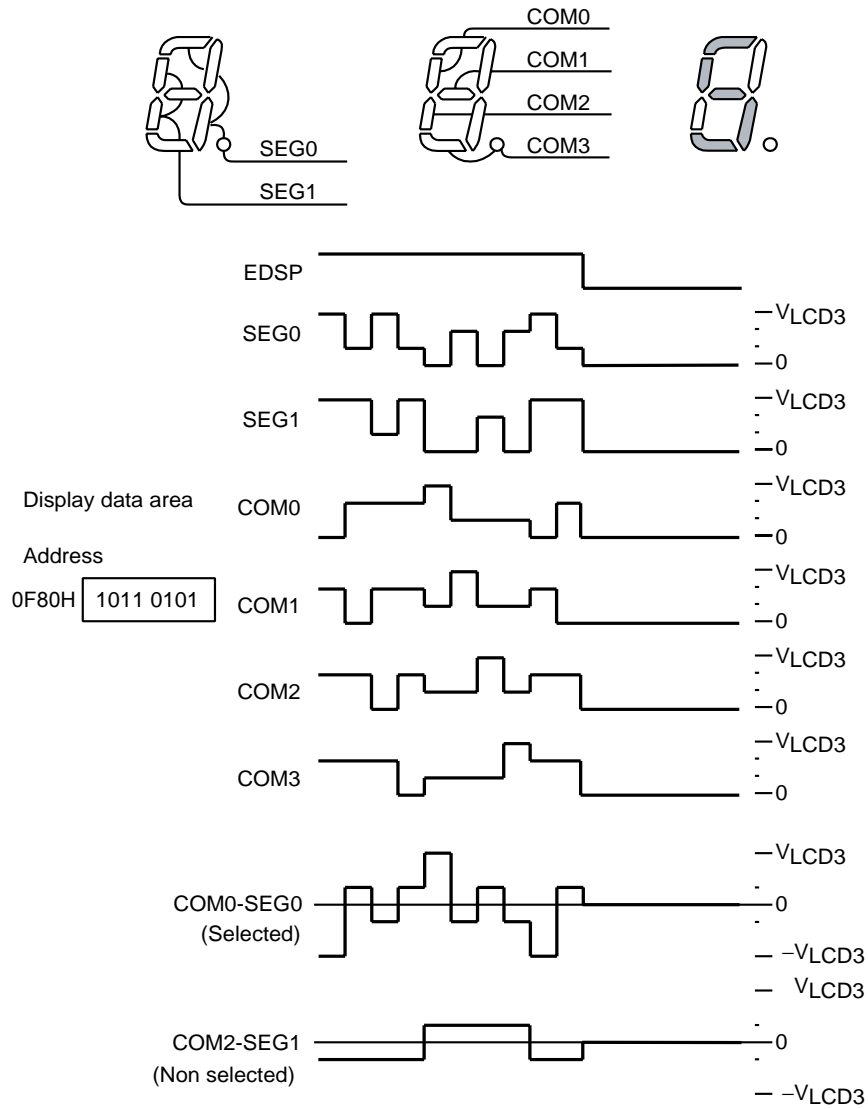
Figure 16-8 1/4 Duty (1/3 bias) Drive
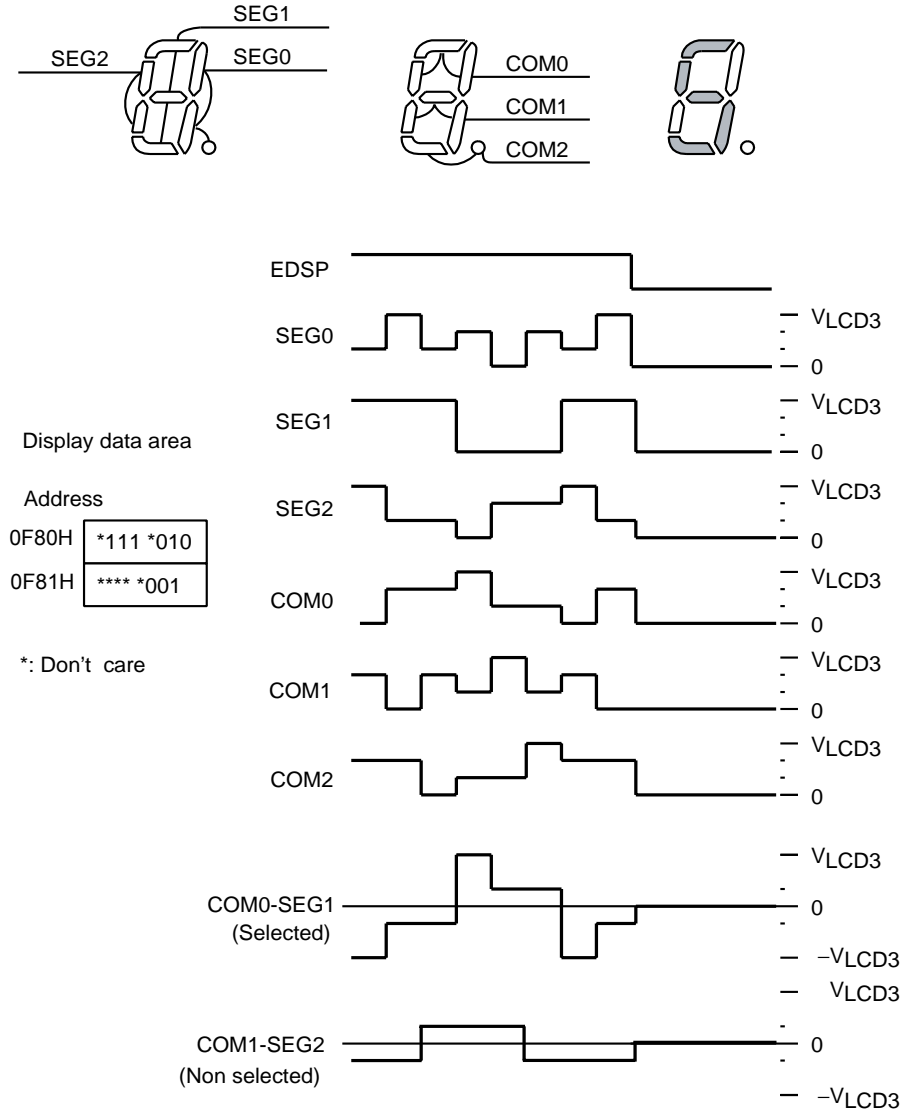
Figure 16-9 1/3 Duty (1/3 bias) Drive
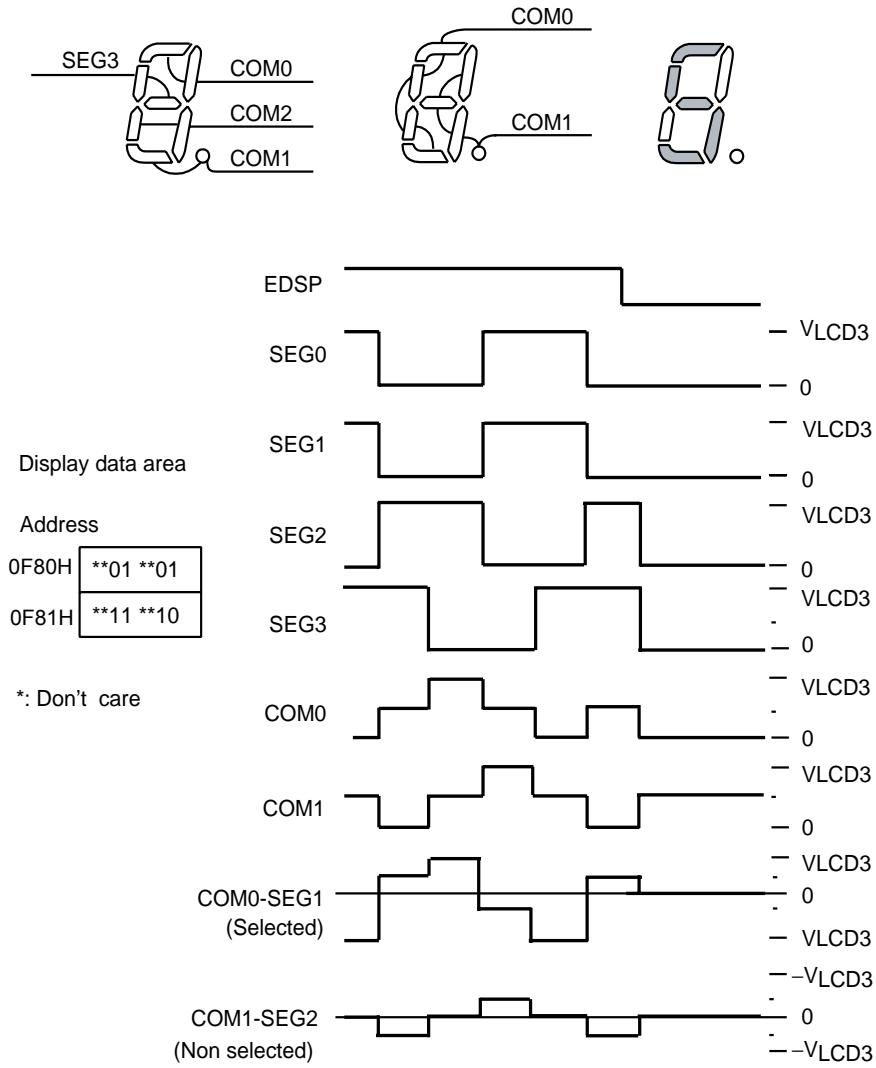
Figure 16-10 1/2 Duty (1/2 bias) Drive

TOSHIBA

SEG0
SEG5 SEG1
SEG6
SEG4 SEG2
SEG3 SEG7

COM0

Display data area

Address

0F80H	***0 ***1
0F81H	***1 ***1
0F82H	***1 ***0
0F83H	***0 ***1

*: Don't care

EDSP

SEG0 — V_{LCD3} / 0

SEG4 — V_{LCD3} / 0

SEG7 — V_{LCD3} / 0

COM0 — V_{LCD3} / 0

COM0-SEG0
(Selected) — V_{LCD3} / 0 / $-V_{LCD3}$

COM0-SEG4
(Non selected) — V_{LCD3} / 0 / $-V_{LCD3}$
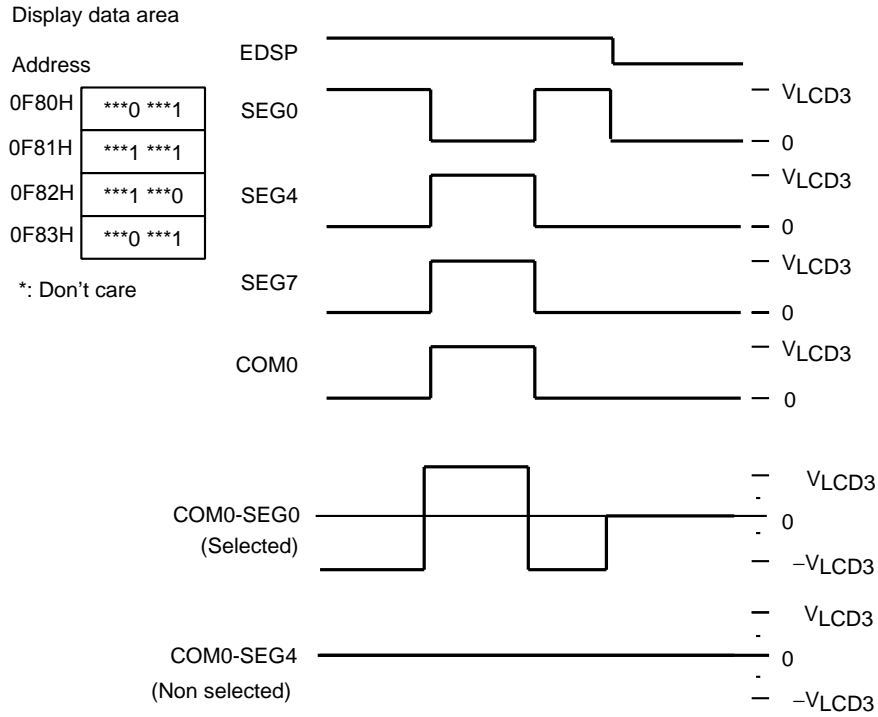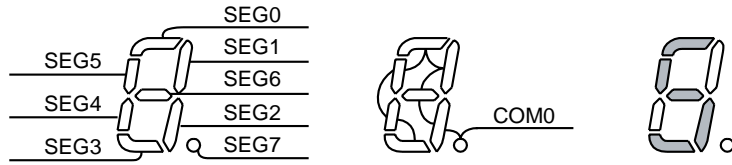
Figure 16-11 Static Drive

17. FLASH Memory

17.1 Outline

The TMP86FM26UG incorporates 32768 bytes of FLASH memory (Address 8000H to FFFFH). Of these bytes, 512 bytes (Address 8000H to 81FFH) can be used as data memory. When these 512 bytes (Address 8000H to 81FFH) are used as data memory, the 32256 bytes (Address 8200H to FFFFH) can be used as program memory. The writing to FLASH memory is controlled by FLASH control register (EEPCR), FLASH status register (EEPSR) and FLASH write emulate time control register (EEPEVA).

The FLASH memory of the TMP86FM26UG features:

- The FLASH memory is constructed of 512 pages FLASH memory and one page size is 64 bytes (512 pages × 64 bytes = 32768 bytes).
- The TMP86FM26UG incorporates a 64-byte temporary data buffer. The data written to FLASH memory is temporarily stored in this data buffer. After 64 bytes data have been written to the temporary data buffer, the writing to FLASH memory automatically starts by page writing (The 64 bytes data are written to specified page of FLASH simultaneously). At the same time, page-by-page erasing occurs automatically. So, it is unnecessary to erase individual pages in advance.
- The FLASH control circuit incorporates an oscillator dedicated to the FLASH. So FLASH writing time is independent of the system clock frequency (fc). In addition, because an FLASH control circuit controls writing time for each FLASH memory cell, the writing time varies in each page (Typically 4 ms per page).
- Controlling the power for the FLASH control circuit (regulator and voltage step-up circuit) achieves low power consumption if the FLASH is not in use (Example: When the program is executed in RAM area).

17.2 Conditions for Accessing the FLASH Areas

The conditions for accessing the FLASH areas vary depending on each operation mode. The following tables shows FLASH are access conditions.

Table 17-1 FLASH Area Access Conditions

	Area	Operation Mode	
		MCU mode (Note1)	Serial PROM mode (Note 2)
Data Memory	8000H to 81FFH	Write/read/fetch [Note 3] supported	
Program Memory	8200H to FFFFH	Read/fetch only	Write/read/fetch supported

Note 1: "MCU mode" shows NORMAL1/2 and SLOW1/2 modes.

Note 2: Serial PROM mode" shows the FLASH controlling mode. For details, refer to "2.19 Serial PROM mode"

Note 3: "Fetch" means reading operation of FLASH data as an instruction by CPU.

17.3 Differences among Product Series

The specifications of the FLASH product (TMP86FM26UG) are different from those of the emulation chip (TMP86FM26UG) as listed below. See " 17.5.2 Control " for explanations about the control registers.

		FLASH Product (TMP86FM26UG)	Emulation Chip (TMP86FM26UG)
Rewriting the EEPCR register<EEPMD, EEPRS, MNPWDW>		It is possible to rewrite the EEPCR register only when the program execution area in use is RAM/BOOT-ROM	
Accessing the EEPEVA register		It is possible only to write- and read-access the EEPEVA register. The writing to this register does not affect the function.	The time required to emulate FLASH writing is put under control.
FLASH write time (The emulation chip is written to emulation memory instead of FLASH)		Typically 4 ms (Independent of the system clock)	The FLASH write time is set up using the EEPEVA register (Dependent on the system clock).
Executing a read instruction/fetch to the 8000H to FFFFH area when EEPSR<BFBUSY> = "1"		If EEPSR<BFBUSY> = "1", executing a read instruction/fetch to the FLASH area causes FFH to be read regardless of what the current ROM data is. Fetching FFH results in a software interrupt occurring.	
			The debugger memory window always displays ROM data.
Executing a write instruction to the 8000H to 81FFH area when EEPCR<EEPMD> = "0011", EEPSR<EWUPEN> = "1" and EEPSR<BFBUSY> = "0".	MCU mode	The EEPSR<BFBUSY> is set to "1" (Write enabled).	
	SerialPROM mode		
Executing a write instruction to the 8200H to FFFFH area when EEPCR<EEPMD> = "0011", EEPSR<EWUPEN> = "1" and EEPSR<BFBUSY> = "0".	MCU mode	The EEPSR<BFBUSY> stays at "0" (Write disabled).	
			In the debugger memory window, it is possible to rewrite the 8200H to FFFFH area (The EEPSR<BFBUSY> remains unchanged).
	SerialPROM mode	The EEPSR<BFBUSY> is set to "1" (Write enabled).	
Data memory (8000H to 81FFH)		512 bytes of FLASH are included in the 8000H to 81FFH area.	512 bytes of emulation memory are included in the 8000H to 81FFH area. (Turning off the power for the emulation chip erases data in the emulation memory.)
BOOT-ROM		2 Kbytes are included in the 3800H to 3FFFH area.	
Operating voltage		VDD = 1.8 to 3.6 V	VDD = 1.8 to 3.3 V

17.4 FLASH Memory Configuration

64 consecutive bytes in the FLASH area are treated as one group, which is defined as a page. The TMP86FM26UG incorporates a one-page temporary data buffer. Writing data to FLASH is temporarily stored in this 64-byte data buffer. After 64 bytes data have been written to the temporary data buffer, these data are written to specified page of FLASH at a time. However, data can be read from any address byte by byte.

17.4.1 Page Configuration

The FLASH area has a page configuration of 64 bytes/page as shown below. The total number of bytes in it is 512 pages × 64 bytes (= 32768 bytes). The writeable area is 8000H to FFFFH in Serial PROM mode.

Note: The program memory (8200H to FFFFH) can be written only in the Serial PROM mode. For details of the Serial PROM mode, refer to "2.19 Serial PROM mode".

Figure 17-1 Page Configuration

17.5 Data Memory of FLASH(address 8000H to 81FFH)

The TMP86FM26UG incorporates 512 bytes (8000H to 81FFH) of data memory of FLASH, which features:

- In the MCU mode, user-created programs can rewrite the data memory of FLASH in page (64 bytes) units. (It can be used to save application last keys and preset data.)
- In the serial PROM mode, it is possible to perform serial writing to the data memory of FLASH in the same manner as for the program memory. So, initial values can be factory-set in the data memory of FLASH.
- Using support programs incorporated in the BOOT-ROM makes it easy to write to the FLASH.
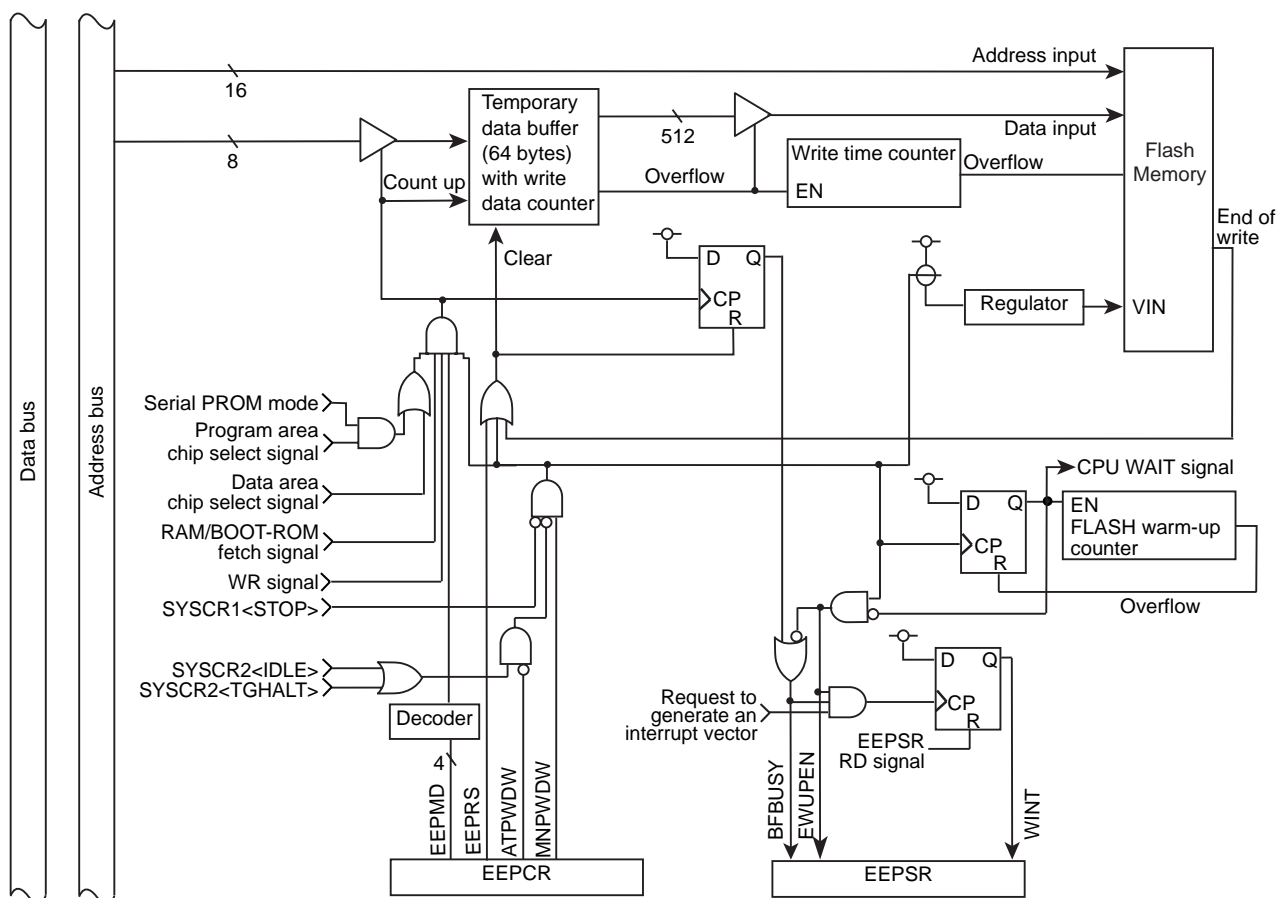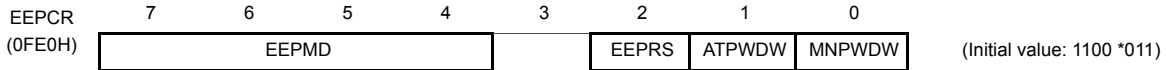
17.5.1 Configuration

Figure 17-2 Data Memory

17.5.2 Control

The FLASH is controlled by FLASH control register (EEPCR), FLASH status register (EEPSR) and FLASH write emulate time control register (EEPEVA).

FLASH Control Register

EEPCR	7	6	5	4	3	2	1	0	
(0FE0H)			EEPMD			EEPRS	ATPWDW	MNPWDW	(Initial value: 1100 *011)

				Program Execution Area	
				RAM/ BOOT	FLASH
EEPMD	FLASH write enable control (Write protect)	1100: FLASH write disable 0011: FLASH write enable Other: values: Reserved		R/W	Read only
EEPRS	FLASH write forcible stop	0: - 1: FLASH writing is forced to stop. (The write data counter is initialized.) * After writing "1" to EEPRS, it is automatically cleared to "0".			Read only
ATPWDW	Automatic power control for the FLASH control circuit in the IDLE0/1/2, SLEEP0/1/2 modes. (This bit is available only when MNPWDW is set to "1".)	0: Automatic power shut down is executed in IDLE0/1/2 and SLEEP0/1/2 modes. 1: Automatic power shut down is not executed in IDLE0/1/2 and SLEEP0/1/2 modes. (The power is always supplied in these modes.			R/W
MNPWDW	Software-based power control for the FLASH control circuit	0: The power for the FLASH control circuit is turned off. 1: The power for the FLASH control circuit is turned on			Read only

Note 1: The EEPMD, EEPRS, and MNPWDW can be rewritten only when a program fetch is taking place in the RAM or BOOT-ROM area. If an attempt is made to rewrite the EEPCR register when a program is being executed in the FLASH area, the EEPMD, EEPRS, and MNPWDW keep holding the previous data; they are not rewritten.

Note 2: To write to the FLASH, set the EEPMD with "0011B" in advance when a program fetch is taking place in the RAM area. However, this processing is not required if a support program in the BOOT-ROM is used.

Note 3: To forcibly stop writing of FLASH, set the EEPRS to "1" when a program fetch is taking place in the RAM area.

Note 4: The ATPWDW functions only if the MNPWDW is "1". If the MNPWDW is "0", the power for the FLASH control circuit is kept turned off regardless of the setting of the ATPWDW.

Note 5: When a STOP mode is executed, the power for the FLASH control circuit is turned off regardless of the setting of the ATP-WDW. If the MNPWDW is "0", entering/exiting the STOP mode allows the power for the FLASH control circuit to be kept turned off.

Note 6: Executing a read instruction to the EEPCR register results in bit3 being read as undefined. Bit2 is always read as "0".

Note 7: The following attention is necessary when the MNPWDW is set or cleared.

When the MNPWDW is changed from "1" to "0"	Clear the interrupt master enable flag (IMF) to "0" in advance to disable an interrupt. After that, do not set IMF to "1" during EEPSR<EWUPEN> = "0". If a watchdog timer is used as interrupt request, clear the binary counter for the watchdog timer just before MNPWDW is changed from "1" to "0".
When the MNPWDW is changed from "0" to "1"	When write to or read from the Flash memory, make sure that the EEPSR<EWUPEN> is "1" by software. Once the MNPWDW is rewritten from "0" to "1" by software, keep performing software-based polling until the EEPSR<EWUPEN> becomes "1".

FLASH Status Register

EEPSR (0FE1H)	7	6	5	4	3	2	1	0	(Initial value: **** *010)
						WINT	EWUPEN	BFBUSY	

WINT	Interrupt detection during a write to the FLASH		0: Not detected 1: Detected (Interrupt occurred) * WINT is automatically cleared to "0" when read instruction is executed to EEPSR.			Read only
EWUPEN	FLASH control circuit status monitor	Control circuit status	Operating (Power on)		Halt (power off) or warm-up	
		FLASH status	Temporary data buffer empty	Writing	Disable	
			1	1	0	
BFBUSY	FLASH write busy flag		0	1	1	

Note 1: If a nonmaskable interrupt occurs during a write to the FLASH, the WINT is set to "1" and the writing is discontinued, and then warm-up (CPU wait) for the control circuit of Flash memory is executed. (The write data counter is initialized.) If WINT = "1" is detected in the nonmaskable interrupt service routine, a write is not completed successfully. So, it is necessary to try a write again. The content of the page to which a write is taking place may be changed to an unexpected value depending on the timing when the WINT becomes "1".

Note 2: Even if a nonmaskable interrupt occurs during an FLASH warm-up, the CPU stays at a halt until the warm-up is finished.

Note 3: The WINT is automatically cleared to "0" when a read instruction is executed to the EEPSR register.

Note 4: When MNPWDW is changed from "0" to "1", EWUPEN becomes "1" after taking 2^{10}/fc [s] (if SYSCK = "0") or 2^3/fs [s] (if SYSCK = "1"). Before accessing the FLASH, make sure that the EWUPEN is "1" in the RAM area.

Note 5: If the BFBUSY is "1", executing a read instruction or fetch to the FLASH area causes FFH to be read. Fetching FFH results in a software interrupt occurring.

FLASH Write Emulation Time Control Register (Setting of this register functions only in emulation chip (TMP86FM26UG).)

EEPEVA (0FE2H)	7	6	5	4	3	2	1	0	(Initial value: **** *000)
						EEPSUCR			

				Emulation Chip (TMP86FM26UG)		FLASH Product (TMP86FM26UG)	
				NORMAL1/2 IDEL1/2 mode	SLOW1/2 SLEEP1/2 mode	All operation modes	
EEPSUCR	Controlling the FLASH write emulation time [s] for emulation chip		000 001 010 011 100 101 110 111	2^{16}/fc 2^{15}/fc 2^{14}/fc 2^{13}/fc 2^{12}/fc Reserved Reserved Reserved	2^7/fc 2^6/fc 2^5/fc 2^4/fc 2^3/fc Reserved Reserved Reserved	Typ. 4 ms (regardless of register settings and the sysytem clock)	R/W

Note 1: Only in the emulation chip, the EEPSUCR functions. In the FLASH product, it is possible only to write- and read-access the register. It does not actually function. Because the FLASH product incorporates a dedicated oscillator, its write time is independent of the system clock.

Note 2: Executing a read instruction to the EEPEVA register results in bit7 to 3 being read as undefined.

Note 3: The following table lists the write emulation time specified by the setting of the EEPSUCR. Select an appropriate value according to the operating frequency used. The shading indicates recommended settings.

	EEPSUCR Setting	NORMAL1/2 Mode					SLOW1/2 Mode
		fc=16 MHz	fc=8 MHz	fc=4 MHz	fc=2 MHz	fc=1 MHz	fs=32.768kHz
Write Time [ms]	000	4.10	8.19	16.38	32.77	65.54	3.91
	001	2.05	4.10	8.19	16.38	32.77	1.95
	010	1.02	2.05	4.10	8.19	16.38	0.98
	011	0.51	1.02	2.05	4.10	8.19	0.49
	100	0.26	0.51	1.02	2.05	4.10	0.24

17.5.3 FLASH Write Enable Control (EEPCR<EEPMD>)

In the FLASH product, the control register can be used to disable a write to the FLASH (Write protect) in order to prevent a write to the FLASH from occurring by mistake because of a program error or microcontroller malfunction. To enable a write to the FLASH, set the EEPCR<EEPMD> with 0011B. To disable a write to the FLASH, set the EEPCR<EEPMD> with 1100B. A reset initializes the EEPCR<EEPMD> to 1100B to disable a write to the FLASH. Usually, set the EEPCR<EEPMD> with 1100B, except when it is necessary to write to the FLASH.

Note: The EEPCR<EEPMD> can be rewritten only when a program is being executed in the RAM area. Executing a write instruction to the EEPCR<EEPMD> in the FLASH area does not change its setting.

17.5.4 FLASH Write Forcible Stop (EEPCR<EEPRS>)

To forcibly stop a write to the FLASH, set the EEPCR<EEPRS> to "1". Setting the EEPCR<EEPRS> to "1" initializes the write data counter of data buffer and forcibly stops a write, and then a warm-up (CPU wait) for the control circuit of Flash memory is executed. After warm-up period, the EEPSR<BFBUSY> is cleared to "0". The warm-up period is 2^{10}/fc (SYSCK = "0") or 2^3/fs (SYSCK = "1"). After this, if writing to FLASH starts again, data is stored as the first byte of the temporary data buffer and sets the EEPSR<BFBUSY> to "1". Therefore, it is necessary to write 64 bytes data to the temporary data buffer.

After 1 to 63 bytes are saved to the temporary data buffer, if the EEPCR<EEPRS> is set to "1" the specified page of FLASH is not written. (It keeps previous data.)

Note 1: After 64 bytes are written to the temporary data buffer, the setting the EEPCR<EEPRS> to "1" may cause the writing the page of FLASH to an unexpected value.

Note 2: The EEPCR<EEPRS> can be rewritten only when a program is being executed in the RAM area. In the FLASH area, executing a write instruction to the EEPCR<EEPRS> does not affect its setting.

Note 3: During the warm-up period for Flash memory (CPU wait), the peripheral circuits continue operating, but the CPU stays at a halt until the warm-up is finished. Even if an interrupt latch is set to "1" by generating of interrupt request, an interrupt sequence doesn't start till the end of warm-up. If interrupts occur during a warm-up period with IMF = "1", the interrupt sequence which depends on interrupt priority will start after warm-up period.

Note 4: When the EEPCR<EEPRS> is set to "1" with EEPSR<BFBUSY> = "0", a warm-up is not executed.

Note 5: If executed a write or read instruction to the Flash area immediately after setting EEPCR<EEPRS>, insert one or more machine cycle instructions after setting EEPCR<EEPRS>.

Example :Reads the Flash memory data immediately after setting EEPCR<EEPRS> to "1"

```
LD      HL,8000H

LD      (EEPCR), 3FH    ; Sets EEPCR<EEPRS>to"1"

NOP                     ; NOP(Do not execute write or read instruction immediately after setting
                          EEPCR<EEPRS>,)

LD      A,(HL)          ; Reads the data of address 8000H (Write or read instruction to the
                          Flash memory)
```
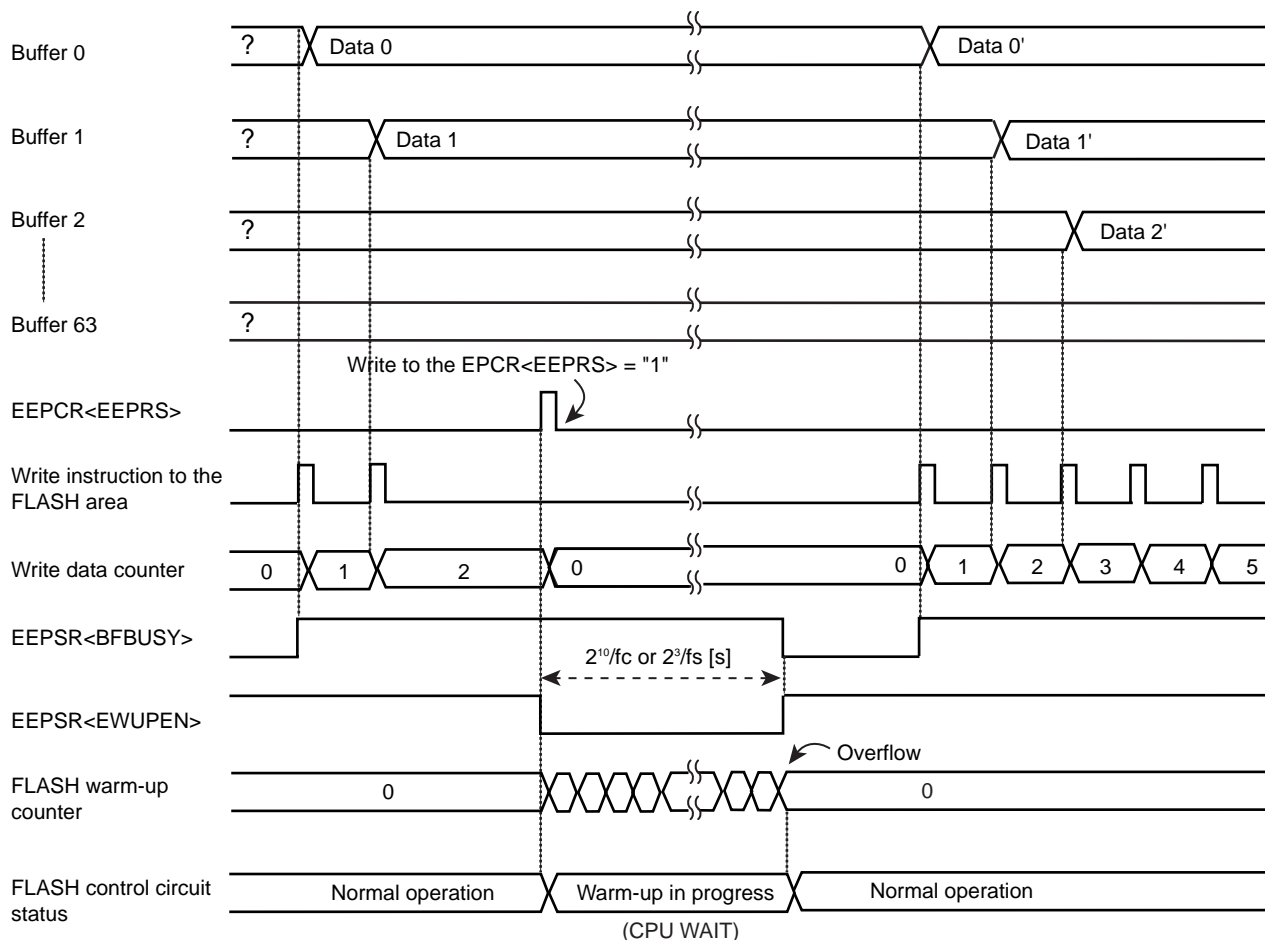
Figure 17-3 Write Data Counter Initialization and Write Forcible Stop

17.5.5 Power Control for the FLASH Control Circuit

For the FLASH product, it is possible to turn off the power for FLASH control circuit (such as a regulator) to suppress power consumption if the FLASH area is not accessed. For the emulation chip (TMP86FM26UG), the register setting and the CPU wait functions behave in the same manner as for the FLASH product to maintain compatibility; however, power consumption is not suppressed.

The EEPCR<MNPWDW> and EEPCR<ATPWDW> are used to control the power for the FLASH control circuit. If the power for the FLASH control circuit is turned off according to the setting of these registers, starting to use the circuits again needs to allow warm-up time for the power supply.

Table 17-2 Power Supply Warm-up Time (CPU wait) for the FLASH Control Circuit

NORMAL1/2 IDLE0/1/2 Mode	SLOW1/2 SLEEP0/1/2 Mode	STOP Mode (when EEPCR<MNPWDW> = "1")	
		To Return to a NORMAL Mode	To Return to a SLOW Mode
$2^{10}/fc$ [s] (64 ms @16 MHz)	$2^3/fs$ [s] (244 ms @32.768 kHz)	STOP warm-up time + $2^{10}/fc$ [s]	STOP warm-up time + $2^3/fs$ [s]

17.5.5.1 Software-based Power Control for the FLASH Control Circuit (EEPCR<MNPWDW>)

The EEPCR<MNPWDW> is a software-based power control bit for the FLASH control circuit. When a program is being executed in the RAM area, setting this bit enables software-based power control. Clearing the EEPCR<MNPWDW> to "0" immediately turns off the power for the FLASH control circuit. Once the EEPCR<MNPWDW> is switched from "0" to "1", before attempting a read or fetch from the FLASH

area, it is necessary to insert a warm up period by software until the power supply is stabilized. In this case, because the CPU wait is not executed, any other instructions except accessing to Flash (write or read) are available. When MNPWDW is changed from "0" to "1", EWUPEN becomes "1" after taking 2^{10}/fc [s] (SYSCK = "0") or 2^3/fs [s] (SYSCK = "1"). Usually software-based polling should be performed until the EEPSR<EWUPEN> becomes "1". An example of setting is given below.

Example of controlling the EEPCR<NNPWDW>

1. Transfer a program for controlling the EEPCR<MNPWDW> to the RAM area.
2. Release an address trap in the RAM area (Setup the WDTCR1 and WDTCR2 registers).
3. Jump to the control program transferred to the RAM area.
4. Clear the interrupt master enable flag (IMF = "0").
5. Clear the binary counter if the watchdog timer is in use.
6. To turn off the power for the FLASH control circuit, clear the EEPCR<MNPWDW> to "0".
7. Perform CPU processing as required.
8. To access the FLASH area again, set the EEPCR<MNPWDW> to "1".
9. Keep program polling until the EEPSR<EWUPEN> becomes "1".
   (Upon completion of an FLASH warming-up, the EEPSR<EWUPEN> is set to "1". It takes 2^{10}/ fc (SYSCK = "0") or 2^3/fs (SYSCK = "1") until EWUPEN becomes "1".)

This procedure enables the FLASH area to be accessed.

If the EEPCR<MNPWDW> is "1", entering a STOP mode forcibley turns off the power for the FLASH control circuit. When the STOP mode is released, a STOP mode oscillation warm-up is carried out, and then the CPU wait period (Warm-up for stabilizing of FLSH power supply circuit) is automatically performed. If the EEPCR<MNPWDW> is "0", entering / exiting the STOP mode keeps the power for the FLASH control circuit turned off.

Note 1: If the EEPSR<EWUPEN> is "0", do not access (Fetch, read, or write) the FLASH area. Executing a read instruction or fetch to the FLASH area causes FFH to be read. Fetching FFH results in a software interrupt occurring.

Note 2: To clear the EEPCR<MNPWDW> to "0", clear the interrupt master enable flag (IMF) to "0" in advance to disable an interrupt. After that, do not set IMF to "1" during EEPSR<EWUPEN> = "0".

Note 3: If the EEPCR<MNPWDW> is "0", generating a nonmaskable interrupt automatically rewrites the MNPWDW to "1" to warm up the FLASH control circuit (CPU wait). That time, the peripheral circuits continue operating, but the CPU stays at a halt until the warm-up is finished.

Note 4: The EEPCR<MNPWDW> can be rewritten only when a program is being executed in the RAM area. In the FLASH area, executing a write instruction to the EEPCR<MNPWDW> does not affect its setting.

Note 5: If a watchdog timer is used as interrupt request, clear the binary counter for the watchdog timer just before MNPWDW is changed from "1" to "0".

Note 6: During the warm-up period with a software polling of EEPSR<EWUPEN>, if a nonmaskable interrupt occurs, the CPU stays at a halt until the warm-up is finished.
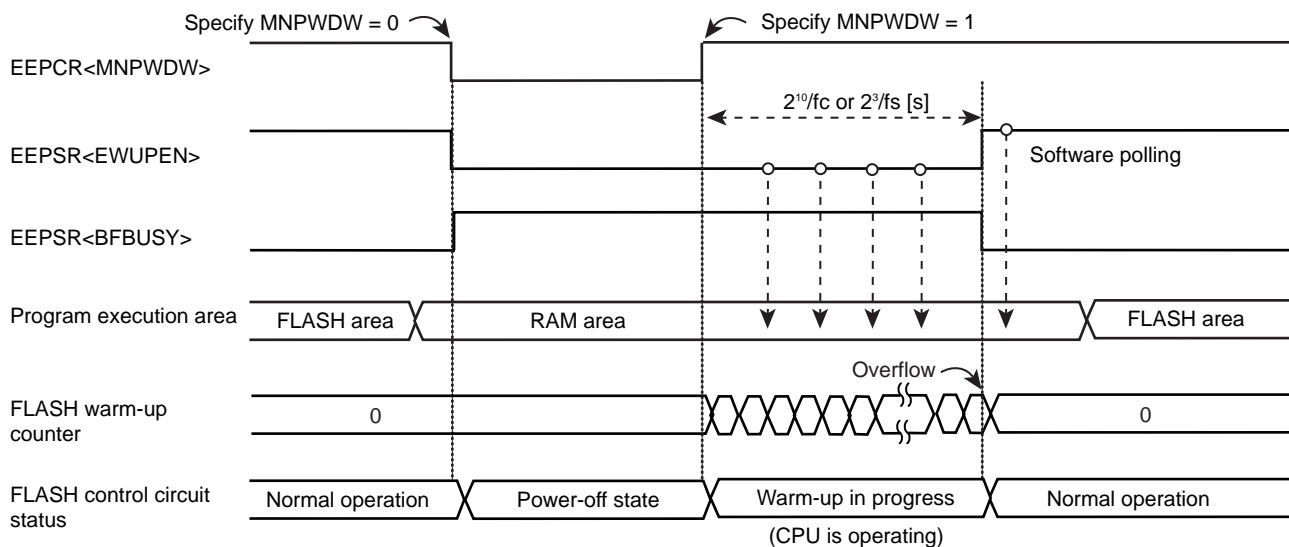
Figure 17-4 Software-base Power Control for the FLASH Control Circuit
(EEPCR<MNPWDW>)

Example :Performing software-based power control for the FLASH control circuit

```
          sRAMAREA:
                    DI                                      ; Disable an interrupt (IMF = "0")
                    LD          (WDTCR2),4EH                ; Clear the binary counter if the watchdog timer is in use
                    CLR         (EEPCR).0                   ; Clear the EEPCR<MNPWDW> to "0".
                    :           :
                    :           :
                    SET         (EEPCR).0                   ; Set the EEPCR<MNPWDW> to "1"
          sLOOP1:   TEST        (EEPSR).1                   ; Monitor the EEPSR<EWUPEN> register.
                    JRS         T,sLOOP1                    ; Jump to sLOOP1 if EEPSR<EWUPEN> = "0".
                    JP          MAIN                        ; Jump to the FLASH area.
```

17.5.5.2 Automatic Power Control for the FLASH Control Circuit (EEPCR<ATPWDW>)

The EEPCR<ATPWDW> is an automatic power control bit for the FLASH control circuit. It is possible to suppress power consumption by automatically shutting down the power for the FLASH control circuit when an operation mode is changed to IDLE0/1/2 and SLEEP0/1/2 modes. This bit can be specified regardless of the area in which a program is being executed.

After the EEPCR<ATPWDW> is cleared to "0", entering an operation mode (IDLE0/1/2 or SLEEP0/1/2) where the CPU is at a halt automatically turns off the power for the FLASH control circuit. Once the operation mode is released, the warm-up time (CPU wait) is automatically counted to resume normal processing. The CPU wait period is either 2^{10}/fc (SYSCK = "0") or 2^3/fs (SYSCK = "1"). If the EEPCR<ATPWDW> is "1", releasing the operation mode does not cause the CPU wait.

If EEPCR<MNPWDW> = "1", executing a STOP mode forcibly turns off the power for the FLASH control circuit regardless of the setting of the EEPCR<ATPWDW>. When the STOP mode is released, a STOP mode oscillation warm-up is carried out, and then an FLASH control circuit warm-up (CPU wait) is automatically performed. If the EEPCR<MNPWDW> is "0", entering/exiting a STOP mode allows the power for the FLASH control circuit to be kept turned off.

Note 1: The EEPCR<ATPWDW> functions only if the EEPCR<MNPWDW> is "1". If the EEPCR<MNPWDW> is "0", the power for the FLASH control circuit is kept turned off when an operation mode is executed or released.

Note 2: During an FLASH warm-up (CPU wait), the peripheral circuits continue operating, but the CPU stays at a halt. Even if an interrupt latch is set under this condition, no interrupt process occurs until the CPU wait is completed. If the IMF is "1" when the interrupt latch is set, interrupt process takes place according to the interrupt priority after the CPU has started operating.
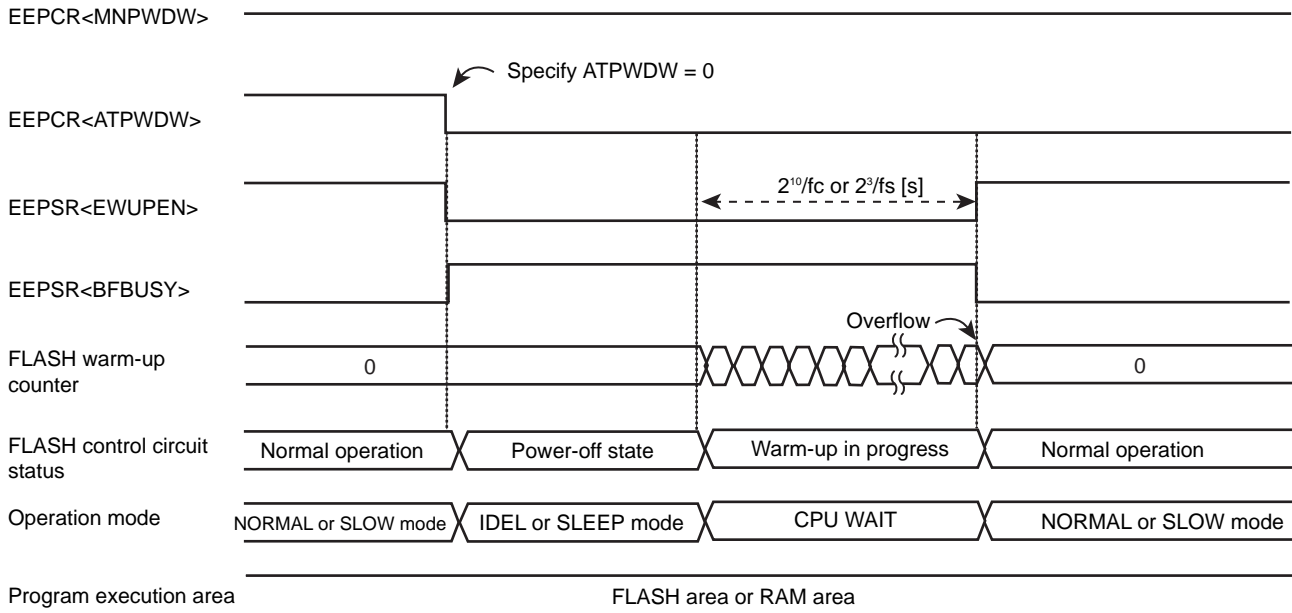
Figure 17-5 Automatic Power Control for the FLASH Control Circuit (EEPCR<ATPWDW>)

17.5.6 Accessing the FLASH Data Memory Area

During the writing to the data memory of FLASH area, neither a read nor fetch can be performed for the 8000H to FFFFH area. Therefore, to write the data memory of FLASH, the program being executed should be jumped to RAM area or should be jumped to the support program in BOOT-ROM. For details about the support program in BOOT-ROM, refer to "" 17.5.6.2 Method of Using Support Programs in the BOOT-ROM "".

An LD instruction can be used to read data from the data memory of FLASH area byte by byte. The support program incorporated in the BOOT-ROM can also be used to read data from the data memory of FLASH area.

If a nonmaskable interrupt occurs during a write to the FLASH (EEPSR<BFBUSY> = "1"), the WINT is set to "1" and the writing is discontinued, and then the warm-up (CPU wait) for control circuit of Flash memory is executed (The write data counter is also initialized). If WINT = "1" is detected in the nonmaskable interrupt service routine, a write is not completed successfully. So, it is necessary to try a write again. The warm-up period is 2^{10}/fc (SYSCK = "0") or 2^3/fs (SYSCK = "1"). After 1 to 63 bytes are saved to the temporary data buffer, if an interrupt generates, the specified page of FLASH is not written. (It keeps previous data.)

Note 1: After 64 bytes are written to the temporary data buffer, the generating of an interrupt may cause the writing the page of FLASH to an unexpected value.

Note 2: During the warm-up period for Flash memory (CPU wait), the peripheral circuits continue operating, but the CPU stays at a halt until the warm-up is finished. Even if an interrupt latch is set to "1" by generating of interrupt request, an interrupt sequence doesn't start till the end of warm-up. If interrupts occur during a warm-up period with IMF = "1", the interrupt sequence which depends on interrupt priority will start after warm-up period.

Note 3: When write the data to Flash memory from RAM area, disable all the nonmaskable interrupt by clearing interrupt master enable flag (IMF) to "0" beforehand. However, in support program of BOOT-ROM, there is no need to clear the IMF because BOOT-ROM already has a DI (Disable Interrupt) instruction.

17.5.6.1 Method of Developing the Control Program in the RAM Area

To develop the program in RAM, the write control program should be stored in FLASH beforehand or should load from external device by using peripheral function (Example: UART, SIO etc). Given below is an example of developing the control program in the RAM area.

(1) Example of developing and writing the control program to the RAM area

1. For the emulation chip, set the EEPEVA register with an optimum time value according to the operating frequency.
2. Transfer the write control program to the RAM area.
3. Release an address trap in the RAM area (Set up the WDTCR1 and WDTCR2 registers).
4. Jump to the RAM area.
5. Monitor the EEPSR<EWUPEN>. If it is "0", set the EEPCR<MNPWDW> to "1", and then start and keep polling until the EEPSR<EWUPEN> becomes "1".
6. Clear the interrupt master enable flag (IMF = "0").
7. Set the EEPCR with "3BH" (to enable a write to the FLASH).
8. Execute a write instruction for 64 bytes to the FLASH area.
9. Start and keep polling by software until the EEPSR<BFBUSY> becomes "0".
   (Upon completion of an erase and write to the FLASH cells, the EEPSR<BFBUSY> is set to "1". For the FLASH product, the required write time is typically 4 ms. For the emulation chip, it is the value specified in the EEPEVA register.)
10. Set the EEPCR with "CBH" (to disable a write to the FLASH).
11. Jump to the FLASH area (Main program).

Note: See " (2) Method of specifying an address for a write to the FLASH " for a description about the FLASH address to be specified at step 8 above.

(2) Method of specifying an address for a write to the FLASH

The FLASH page to be written is specified by the 10 high-order bits of the address of the first-byte data. The first-byte data is stored at the first address of the temporary data buffer. If the data to be written is, for example, 8040H, page 1 is selected, and the data is stored at the first address of the temporary data buffer. Even if the 6 low-order bits of the specified address is not 000000B, the first-byte data is always stored at the first address of the data buffer.

Any address can be specified as the second and subsequent address within FLASH area (for the MCU mode, 8000 to 81FFH and, for the serial PROM mode, 8000H to FFFFH). The write data bytes are stored in the temporary data buffer in the sequence they are written, regardless of what address is specified. Usually, the address that is the same as the first-byte is specified for the second and subsequent address. A 16-bit transfer instruction (LDW) can also be used for writing to the temporary data buffer.

Example :Data bytes 00H to 3FH are written to page 1

```
                DI                                        ; Disable an interrupt (IMF = "0")

                LD       C,00H

                LD       HL,EEPCR                         ; Specify the EEPCR register address.

                LD       IX,8040H                         ; Specify a write address.

                LD       (HL),3BH                         ; Specify the EEPCR

         sLOOP1:

                LD       (IX),C                           ; Store data to the temporary data buffer. (A write page is selected
                                                            when the first byte is written.)

                INC      C                                ; C = C + 1

                CMP      C,40H                            ; Jump to sLOOP1 if C is not 40H

                JR       NZ,sLOOP1

         sLOOP2:

                TEST     (EEPSR).0

                JRS      F,sLOOP2                          ; Jump to sLOOP2 if EEPSR<BFBUSY> = "1".

                LD       (HL),0CBH                         ; Specify the EEPCR
```

Note: If the BFBUSY is "1", executing a read instruction or fetch to the FLASH area causes "FFH" to be
read. Fetching "FFH" results in a software interrupt occurring.
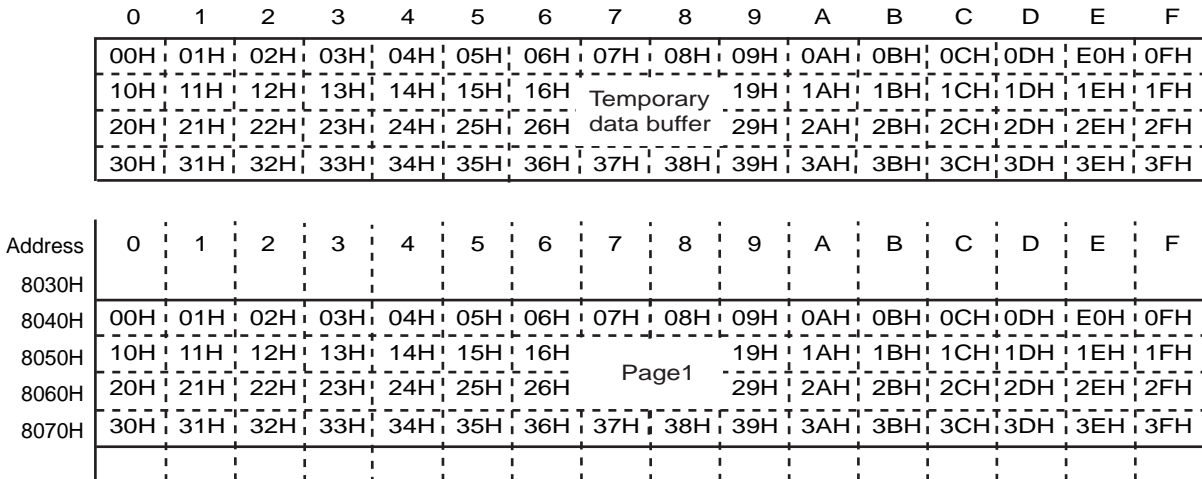
Figure 17-6 Data Buffer and Write Page (Example)
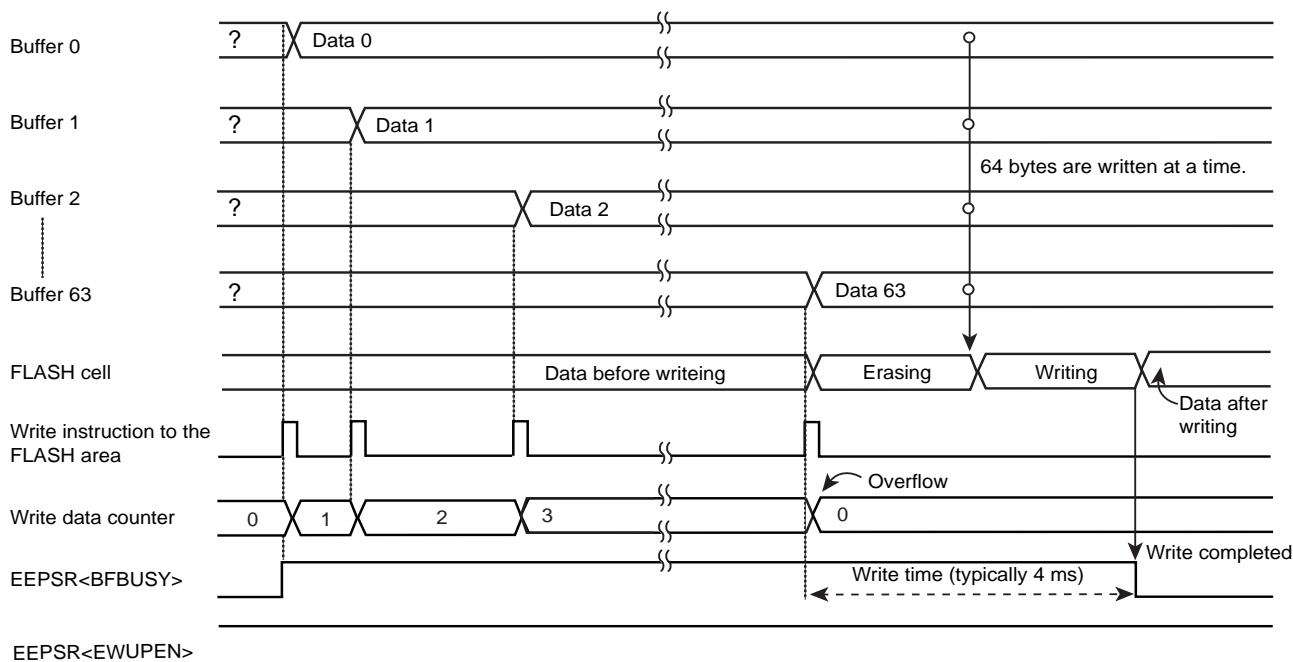
Figure 17-7 Write to the FLASH Data Memory Area (In case of FLASH product)
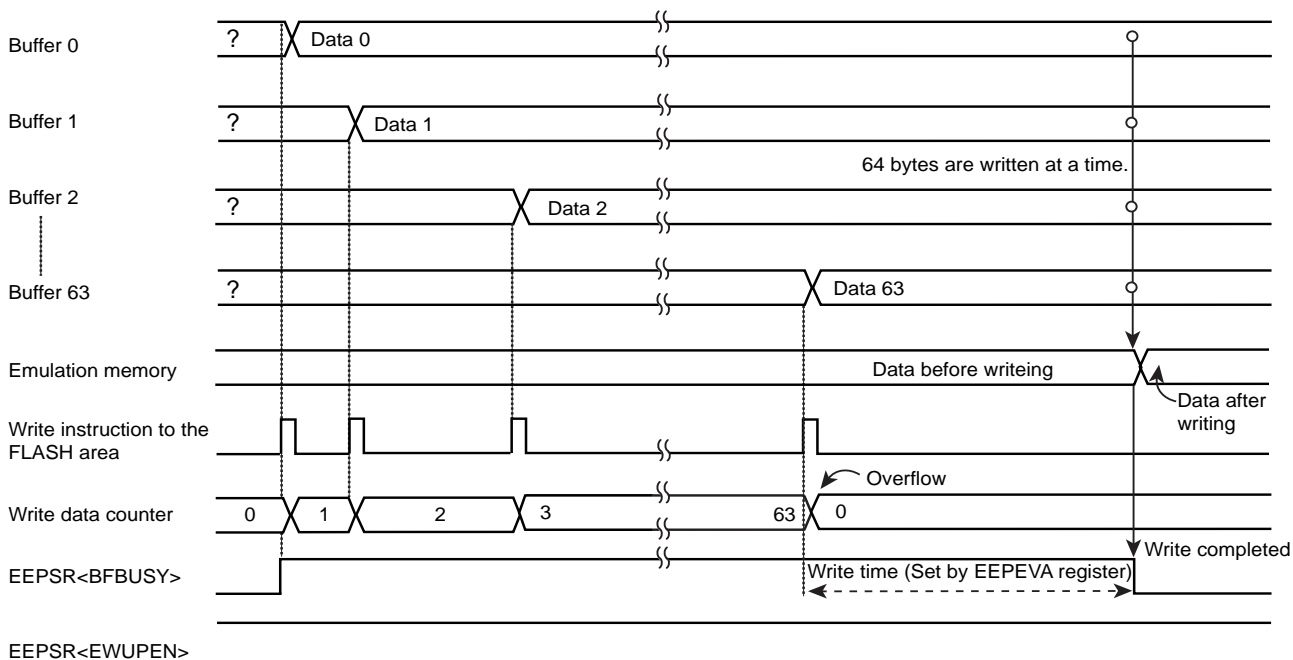
Figure 17-8 Write to the FLASH Data Memory Area (In case of emulation chip)

Note 1: The emulation chip is written to emulation memory instead of FLASH cell.

Note 2: In case of emulation chip,the data stacked to data buffer is written to emulation memory just before EEPSR<BFBUSY>is changed from "1" to "0". therefore, if the writing of FLASH is stopped forcibly after the write data counter becomes overflow, the memory value on the page subjected to a write may be different from FLASH product.

17.5.6.2 Method of Using Support Programs in the BOOT-ROM

The BOOT-ROM of TMP86FM26UG has Support Program to simplify writing/reading of FLASH. This program supports three subroutines.

1. Writing to data FLASH from RAM
2. Reading from data FLASH to RAM
3. Reading from program FLASH to RAM

In addition to a program for controlling a write in the serial PROM mode, the BOOT-ROM incorporates support programs for simplifying a write to the data memory of FLASH in the MCU mode. The support programs take the form of a subroutine. After setting general-purpose registers with the necessary data, just execute a CALL instruction for a support program. It enables a write to and a read from the FLASH. There are two subroutines in BOOT-ROM. The Table 17-3 shows the function of these subroutines.

Table 17-3 Support Program (Subroutines) in BOOT-ROM

Program	CALL address	Function
Support program 1	3E00H	Writing to data memory of FLASH (8000H to 81FFH) from RAM area is available. 64-byte data can be written at a time.
Support program 2	3E2CH	Reading from FLASH memory (8000H to FFFFH) into RAM area is available. 64-byte data can be read at a time.

When using the support program, it is unnecessary to prepare an FLASH write program in advance or develop it in the RAM area.

Support program 1 enables 64 consecutive data bytes to be transferred from the RAM area to any data memory of FLASH page in block. (Only data memory of FLASH is available.) Support program 2 enables data to be transferred from any FLASH memory page (Both data memory of FLASH and program memory are available.) to a specified 64-byte consecutive RAM area in block.

How to use the support programs in the BOOT-ROM is explained below. See " (3) Support program 1 (Block transfer from the RAM area to the FLASH data area) " and " (4) Support program 2 (Block transfer from the FLASH area to the RAM area) " for the source code of the support programs.

(1) Example of using support program 1 to write data to the FLASH data area

(Block transfer from the RAM area to the FLASH data area to the RAM area)

1. For the emulation chip, set the EEPEVA register with the optimum time according to the operating frequency.
2. Set data in the transfer-source RAM area.
3. Set the RAM area start address (Transfer source) in the HL register.
4. Set the FLASH data area start address (Transfer destination) in the DE register.
5. Set "1FH" in the B register. (Be sure to set 1FH (Half of the number of bytes to be written.))
6. Clear the binary counter if the watchdog timer is in use.
7. Execute a CALL instruction to "3E00H".
8. Data is transferred from the RAM area to the FLASH data area in block. After several milliseconds, program control is returned to the main routine.

Note 1: Steps 1 to 6 above are executed in the FLASH area.
Note 2: Support program 1 rewrites the HL, DE, B, and WA registers. If the existing data in them are necessary, save it in advance.
Note 3: If the EEPCR<MNPWDW> is "0", executing support program 1 rewrites it to "1" before performing a block transfer.
Note 4: Executing a CALL instruction for support program 1 consumes two bytes of stack.

Note 5: If the watchdog timer is in use, be sure to clear the binary counter for it before executing a CALL instruction to 3E00H.

Note 6: Do not specify the address from 8200H to FFFFH as a transfer destination address in MCU mode.

Example : Setting up HL = 0050H, DE = 8100H, and B = 1FH, and executing a CALL instruction for support program 1 (3E00H)

Figure 17-9 Example of Using Support Program 1 to Write Data to the FLASH Data Area

(2) Using support program 2 to read data from the FLASH area

(Block transfer from the FLASH area to the RAM area)

1. Set the RAM area start address (Transfer destination) in the HL register.
2. Set the FLASH area start address (Transfer source) in the DE register.
3. Set "1FH" in the B register. (Be sure to set 1FH (Half of the number of bytes to be read.))
4. Execute a CALL instruction to "3E2CH".
5. Data is transferred from the FLASH area to the RAM area in block. Upon completion of processing, program control is returned to the main routine.

Note 1: A LD instruction can be used to read data from the FLASH area in byte units without using support program 2.

Note 2: Steps 1 to 4 above are executed in the FLASH area.

Note 3: Support program 2 rewrites the HL, DE, B, and WA registers. If the existing data in them are necessary, save it in advance.

Note 4: Executing a CALL instruction for support program 2 consumes two bytes of stack.

Example : Setting up HL = 0050H, DE = 8100H, and B = 1FH, and executing a CALL instruction for support program 2 (3E2CH)

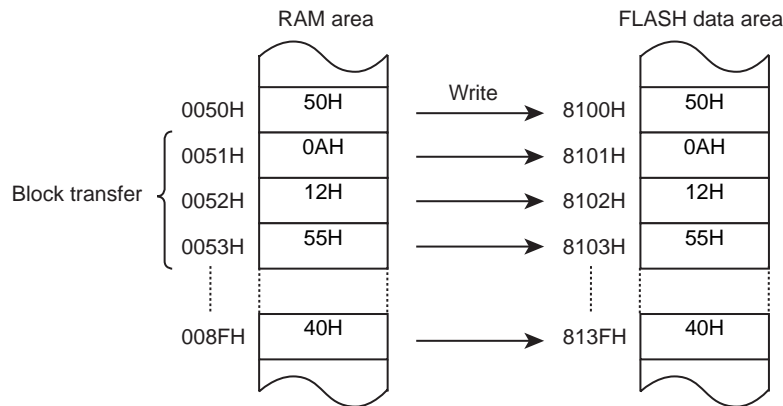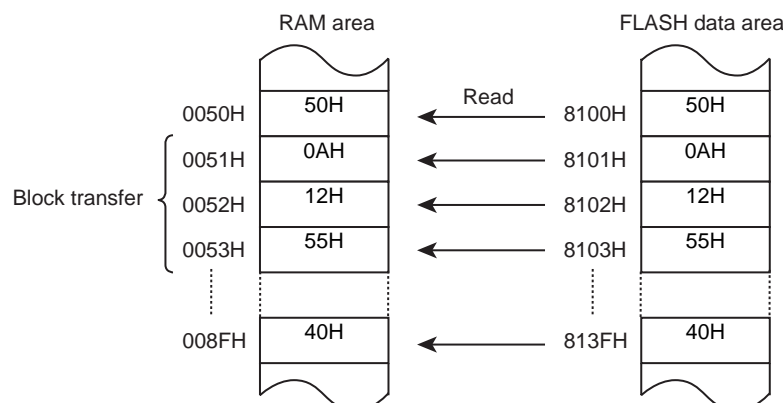**Figure 17-10 Example of Using Support Program 2 to Read Data from the FLASH area**

(3)   Support program 1 (Block transfer from the RAM area to the FLASH data area)

Shown below is the support program source code for writing data to the FLASH.

Example :

```
USER_SUB_WRITE section code abs = 3E00H
sUSER_main1:
                TEST        (EEPSR).1
                JRS         f,sRAM_to_EEP        ; Jump sRAM_to_EEP if the EEPSR<EWUPEN> is "1".
sEEP_warmingup:
                SET         (EEPCR).0            ; Set the EEPCR<MNPWDW> to "1".
                TEST        (EEPSR).1            ; Wait until a warm-up is completed.
                JRS         t,sEEP_warmingup
sRAM_to_EEP:
                DI                               ; Disable an interrupt.
                AND         DE,0FFC0H            ; Mask the 6 low-order bits.
                LD          (EEPCR),3BH          ; Enable a write to the FLASH.
sBFBUSY_loop:
                LD          WA,(HL)              ; Read data from the RAM.
                LD          (DE),WA              ; Writeh data to the FLASH.
                INC         HL
                INC         HL
                DEC         B
                JRS         F,sBFBUSY_loop
sEEP_write_end:
                TEST        (EEPSR).0            ; Perform polling on the BFBUSY flag.
                JRS         F,sEEP_write_end
                LD          (EEPCR),0CBH         ; Disable a write to hte FLASH
                RET
```

(4)   Support program 2 (Block transfer from the FLASH area to the RAM area)

Shown below is the support program source code for reading data from the FLASH.

Example :

```
USER_SUB_READ section code abs = 3E2CH
sUSER_main2:
                AND         DE,0FFC0            ; Mask the 6 low-order bits.
                LD          (EEPCR),0CBH       ; Disable a write to the FLASH.
sEEP_read_loop:
                LD          WA,(DE)
                LD          (DE),WA
                INC         HL
                INC         HL
                INC         DE
                INC         DE
                DEC         B
                JRS         F,sEEP_read_loop
                RET
```

17.6 FLASH Program Memory

The TMP86FM26UG incorporates 32256 bytes (8200H to FFFFH) of program memory. If the data memory of FLASH is not in use, the TMP86FM26UG can be used as an FLASH product with 32768 full bytes. To write data to the program memory (Data memory of FLASH), execute the serial PROM mode.

17.6.1 Configuration

The program memory has the same configuration as for the data memory of FLASH. See Section " 17.5.1 Configuration ".

17.6.2 Control

The program memory is controlled in the same manner as for the data memory of FLASH. See Section " 17.5.2 Control ".

17.6.3 FLASH Write Enable Control (EEPCR<EEPMD>)

The FLASH write enable control register for the program memory behaves in the same manner as for the data memory of FLASH. See Section " 17.5.3 FLASH Write Enable Control (EEPCR<EEPMD>) ".

17.6.4 FLASH Write Forcible Stop (EEPCR<EEPRS>)

The FLASH write forcible stop register for the program memory behaves in the same manner as for the data memory of FLASH. See Section " 17.5.4 FLASH Write Forcible Stop (EEPCR<EEPRS>) ".

17.6.5 Power Control for the FLASH Control Circuit

The power for the program memory control circuit is controlled in the same manner as for the data memory of FLASH. See Section " 17.5.5 Power Control for the FLASH Control Circuit ".

17.6.6 Accessing the FLASH Program Memory Area

Basically, a write to the program memory area is carried out using UART communication after the serial PROM mode is entered. For explanations about what control is performed in the serial PROM mode, see the following descriptions.

18. Input/Output Circuit

18.1 Control pins

The input/output circuitries of the TMP86FM26UG control pins are shown below.

Control Pin	I/O	Input/Output Circuitry	Remarks
XIN XOUT	Input Output		Resonator connecting pins (High frequency) R_f = 3 MΩ (typ.) R_O = 1 kΩ (typ.)
XTIN XTOUT	Input Output		Resonator connecting pins (Low frequency) R_f = 20 MΩ (typ.) R_O = 220 kΩ (typ.)
$\overline{\text{RESET}}$	Input		Hysteresis input Pull-up resistor R_{IN} = 220 kΩ (typ.) R = 100 kΩ (typ.)
TEST	Input		Pull-down resistor R_{IN} = 70 kΩ (typ.) R = 100 kΩ (typ.)

18.2 Input/Output Ports

Port	I/O	Input/Output Circuitry	Remarks
P1	I/O		Tri-state I/O Hysteresis input High current outout (N-ch) LCD segment output R = 100 Ω (typ.)
P5 P7	I/O		Tri-state I/O LCD segment output R = 100 Ω (typ.)
P20	I/O		Sink open drain output or C-MOS output Hysteresis input High current output (N-ch) R = 100 Ω (typ.)
P21 P22	I/O		Sink open drain output Hysteresis input R = 100 Ω (typ.)
P23 P24 P3 P6	I/O		Sink open drain output or C-MOS output Hysteresis input High current output (N-ch) R = 100 Ω (typ.)

Note: Although a voltage exceeding V_{DD} can be output from pins P1, P5 and P7 when used as LCD segment output, the absolute maximum rating for input voltage on these pins is "−0.3 to V_{DD} + 0.3 "[V].

19. Electrical Characteristics

19.1 Absolute Maximum Ratings

The absolute maximum ratings are rated values which must not be exceeded during operation, even for an instant. Any one of the ratings must not be exceeded. If any absolute maximum rating is exceeded, a device may break down or its performance may be degraded, causing it to catch fire or explode resulting in injury to the user. Thus, when designing products which include this device, ensure that no absolute maximum rating value will ever be exceeded.

(Vss = 0 V)

Parameter	Symbol	Pins	Ratings	Unit
Supply voltage	V_{DD}	VDD	−0.3 to 4.0	V
	V_{LCD}	V3 pin	−0.3 to 4.0	V
Input voltage	V_{IN}		−0.3 to V_{DD} + 0.3	V
Output voltage	V_{OUT}		−0.3 to V_{DD} + 0.3	V
Output current(Per 1 pin)	I_{OUT1}	P1, P20, P23, P24, P3, P5, P6, P7 ports	−1.8	mA
	I_{OUT2}	P21, P22, P5, P7 ports	3.2	
	I_{OUT3}	P1, P20, P23, P24, P3, P6 ports	30	
Output current(Total)	ΣI_{OUT1}	P1, P20, P23, P24, P3, P5, P6, P7 ports	−30	mA
	ΣI_{OUT2}	P21, P22, P5, P7 ports	60	
	ΣI_{OUT3}	P1, P20, P23, P24, P3, P6 ports	120	
Power dissipation[Topr = 85 °C]	PD		350	mW
Soldering temperature(time)	Tsld		260 (10s)	°C
Storage temperature	Tstg		−55 to 125	°C
Operating temperature	Topr		−40 to 85	°C

19.2 Operating Conditions

The Operating Conditions show the conditions under which the device be used in order for it to operate normally while maintaining its quality. If the device is used outside the range of Operating Conditions (power supply voltage, operating temperature range, or AC/DC rated values), it may operate erratically. Therefore, when designing your application equipment, always make sure its intended working conditions will not exceed the range of Operating Conditions.

19.2.1 MCU mode

(Vss = 0 V, Topr = –40 to 85 °C)

Parameter	Symbol	Pins	Condition		Min	Max	Unit
Supply voltage	V_{DD}		fc = 16 MHz	NORMAL1,2 modes	2.7	3.6	V
				IDLE0,1,2 modes			
			fc = 8 MHz (In case of connecting the resonator)	NORMAL1,2 modes	1.8		
				IDLE0,1,2 modes			
			fc = 4.2 MHz (In case of external clock input)	NORMAL1,2 modes	1.8		
				IDLE0,1,2 modes			
			fs = 32.768 kHz	SLOW1,2 modes			
				SLEEP0,1,2 modes			
				STOP mode			
Input high voltage	V_{IH1}	Except hysteresis input	$V_{DD} \geq 2.7$ V		$V_{DD} \times 0.70$	V_{DD}	V
	V_{IH2}	Hysteresis input			$V_{DD} \times 0.75$		
	V_{IH3}		$V_{DD} < 2.7$ V		$V_{DD} \times 0.90$		
Input low voltage	V_{IL1}	Except hysteresis input	$V_{DD} \geq 2.7$ V			$V_{DD} \times 0.30$	V
	V_{IL2}	Hysteresis input			0	$V_{DD} \times 0.25$	
	V_{IL3}		$V_{DD} < 2.7$ V			$V_{DD} \times 0.10$	
Clock frequency (In case of connecting the resonator)	fc	XIN,XOUT	V_{DD} = 1.8 to 3.6 V		1.0	8.0	MHz
			V_{DD} = 2.7 to 3.6 V			16.0	
	fs	XTIN,XTOUT			30.0	34.0	kHz
Clock frequency (In case of external clock input)	fc	XIN,XOUT	V_{DD} = 1.8 to 3.6 V		1.0	4.2	MHz
			V_{DD} = 2.7 to 3.6 V			16.0	
	fs	XTIN,XTOUT			30.0	34.0	kHz

19.2.2 Serial PROM mode

(Vss = 0 V, Topr = 25°C ± 5 °C)

Parameter	Symbol	Pins	Condition	Min	Max	Unit
Supply voltage	V_{DD}		2 MHz ≤ fc ≤ 16MHz	2.7	3.6	V
Clock frequency	fc	XIN, XOUT	V_{DD} = 2.7 to 3.6 V	2.0	16.0	MHz

Note: The operating temperature area of serial PROM mode is 25°C ± 5 °C and the operating area of high frequency of seria PROM mode is different from MCU mode.

19.3 DC Characteristics

(Vss = 0 V, Topr = –40 to 85 °C)

Parameter	Symbol	Pins			Condition		Min	Typ.	Max	Unit
Hysteresis voltage	V_{HS}	Hysteresis input			V_{DD} = 3.3 V		-	0.4	-	V
Input current	I_{IN1}	TEST			V_{DD} = 3.6 V, V_{IN} = 0 V		-	-	–5	μA
	I_{IN2}	Sink Open Drain, Tri–state Port			V_{DD} = 3.6 V, V_{IN} = 3.6/0 V		-	-	±5	
	I_{IN3}	$\overline{RESET}$			V_{DD} = 3.6 V, V_{IN} = 3.6 V		-	-	+5	
Input resistance	R_{IN1}	TEST Pull–Down			V_{DD} = 3.6 V, V_{IN} = 3.6 V		-	70	-	kΩ
	R_{IN2}	$\overline{RESET}$ Pull–Up			V_{DD} = 3.6 V, V_{IN} = 0 V		100	220	450	
High frequency feedback resistor	R_{fx}	XIN–XOUT			V_{DD} = 3.6 V		-	3	-	MΩ
Low frequency feedback resistor	R_{fxt}	XTIN–XTOUT			V_{DD} = 3.6 V		-	20	-	
Output leakage current	I_{LO1}	Sink Open Drain Port			V_{DD} = 3.6 V, V_{OUT} = 3.6 V		-	-	+10	μA
	I_{LO2}	Tri–state Port			V_{DD} = 3.6 V, V_{OUT} = 3.6 V / 0 V		-	-	±10	
Output high voltage	V_{OH}	Tri–state Port,CMOS Port			V_{DD} = 3.6 V, I_{OH} = -0.6 mA		3.2	-	-	V
Output low voltage	V_{OL}	Except XOUT, Nch high current port			V_{DD} = 3.6 V, I_{OL} = 0.9 mA		-	-	0.4	V
Output low current	I_{OL}	Nch High current port			V_{DD} = 3.6 V, V_{OL} = 1.0 V		-	20	-	mA
Supply current in NORMAL1, 2 modes		Fetch area	Flash area		V_{DD} = 3.6 V V_{IN} =3.4 V/0.2 V fc = 16 MHz fs = 32.768 kHz	MNP = "1"	-	5.3	7.3	mA
			RAM area			MNP = "0"	-	3.4	5.0	
Supply current in IDLE0, 1, 2 modes						MNP&ATP = "1"	-	3.1	5.0	
						MNP&ATP = "0"	-	2.2	4.0	
Supply current in SLOW1 mode		Fetch area	Flash area		V_{DD} = 3.0 V V_{IN} = 2.8 V/0.2V fs = 32.768 kHz	MNP = "1"	-	850	1200	μA
			RAM area			MNP = "0"	-	7.0	18	
Supply current in SLEEP1 mode						MNP&ATP = "1"	-	850	1200	
						MNP&ATP = "0"	-	5.5	16	
Supply current in SLEEP0 mode						MNP&ATP = "1"	-	850	1200	
						MNP&ATP = "0"	-	4.5	14	
Supply current in STOP mode					V_{DD} = 3.6 V V_{IN} =3.4 V/0.2		-	0.5	10	

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

Note 2: Input current (I_{IN1}, I_{IN3}): The current through pull-up or pull-down resistor is not included.

Note 3: The supply currents of SLOW2 and SLEEP2 modes are equivalent to IDLE0, 1, 2

Note 4: MNP (MNPWDW) shows bit0 in EEPCR register and ATP (ATPWDW) shows bit1 in EEPCR register.

Note 5: "Fetch" means reading operation of FLASH data as an instruction by CPU.

19.4 Timer Counter 1 input (ECIN) Characteristics

(Vss = 0 V, Topr = –40 to 85 °C)

Parameter	Symbol	Condition		Min	Typ.	Max	Unit
TC1 input (ECIN input)	t_{TC1}	Frequency measurement mode V_{DD} = 2.7 to 3.6 V	Single edge count	–	–	16	MHz
			Both edge count				
		Frequency measurement mode V_{DD} = 1.8 to 2.7 V	Single edge count	–	–	8	
			Both edge count				

19.5 LCD Characteristics

(Vss = 0 V, Topr = –40 to 85 °C)

Parameter	Symbol	Pins	Condition	Min	Typ.	Max	Unit
LCD output voltage (Lcd booster is enable)	V2-3OUT	V2 pin	V3 ≥ VDD Reference supply pin: V1 SEG/COM pin: No load	–	V1×2	–	V
		V3 pin		–	V1×3	–	
	V1-3OUT	V1 pin	V3 ≥ VDD Reference supply pin: V2 SEG/COM pin: No load	–	V2×1/2	–	
		V3 pin		–	V2×3/2	–	
	V1-2OUT	V1 pin	V3 ≥ VDD Reference supply pin: V3 SEG/COM pin: No load	–	V2×1/3	–	
		V2 pin		–	V2×2/3	–	
LCD reference voltage	$V1_{IN}$	V1 pin	LCD booster circuit is enable (V3 ≥ V_{DD})	0.8	–	1.2	V
	$V2_{IN}$	V2 pin		1.6	–	2.4	
	$V3_{IN}$	V3 pin		2.4	–	3.6	
Capacity for LCD booster circuit	C_{LCD}			0.1	–	0.47	μF

19.6 AC Characteristics

(Vss = 0.0 V, 2.7 V $\leq$ V$_{DD}$ < 3.6 V, Topr = –40 to 85 °C)

Parameter	Symbol	Condition		Min	Typ.	Max	Unit
Machine cycle time	tcyc	NORMAL1, 2 modes		0.25	–	4	μs
		IDLE1, 2 modes					
		SLOW1, 2 modes		117.6	–	133.3	
		SLEEP1, 2 modes					
High-level clock pulse width	t$_{WCH}$	For external clock operation(XIN input) fc = 16 MHz		–	31.25	–	ns
Low-level clock pulse width	t$_{WCL}$						
High-level clock pulse width	t$_{WSH}$	For external clock operation(XTIN input) fs = 32.768 kHz		–	15.26	–	μs
Low-level clock pulse width	t$_{WSL}$						

(Vss = 0.0 V, 1.8 V $\leq$ V$_{DD}$ < 3.6 V, Topr = –40 to 85 °C)

Parameter	Symbol	Condition		Min	Typ.	Max	Unit
Machine cycle time	tcyc	NORMAL1, 2 modes		0.5	–	4	μs
		IDLE1, 2 modes					
		SLOW1, 2 modes		117.6	–	133.3	
		SLEEP1, 2 modes					
High-level clock pulse width	t$_{WCH}$	For external clock operation(XIN input) fc = 4.2 MHz		–	119.04	–	ns
Low-level clock pulse width	t$_{WCL}$						
High-level clock pulse width	t$_{WSH}$	For external clock operation(XTIN input) fs = 32.768 kHz		–	15.26	–	μs
Low-level clock pulse width	t$_{WSL}$						

19.7 Flash Characteristics

(Vss = 0 V)

Parameter	Condition	Min	Typ.	Max	Unit
Number of guaranteed writes (page writing) to Flash memory in serial PROM mode	V$_{DD}$ = 2.7 to 3.6 V, 2 MHz $\leq$ fc $\leq$ 16 MHz (Topr = 25 °C ± 5 °C)	–	–	10^5	Times
Number of guaranteed writes (page writing) to Flash memory in MCU mode	V$_{DD}$ = 1.8 to 3.6 V at fc = 8 MHz V$_{DD}$ = 2.7 to 3.6 V at fc = 16MHz (Topr = –40 to 85 °C)	–	–	10^5	
Writing time to Flash data memory for one page (64 bytes) in MCU mode		–	4	6	ms

19.8 Oscillating Conditions

(1)High-frequency Oscillation   (2)Low-frequency Oscillation
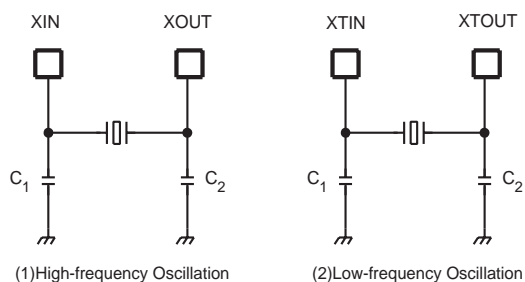
Note 1: A quartz resonator can be used for high-frequency oscillation only when VDD is 2.7 V or above. If VDD is below 2.7V, use a ceramic resonator.

Note 2: To ensure stable oscillation, the resonator position, load capacitance, etc. must be appropriate. Because these factors are greatly affected by board patterns, please be sure to evaluate operation on the board on which the device will actually be mounted.

Note 3: The product numbers and specifications of the resonators by Murata Manufacturing Co., Ltd. are subject to change. For up-to-date information, please refer to the following
URL: http://www.murata.com

19.9 Handling Precaution

- The solderability test conditions for lead-free products (indicated by the suffix G in product name) are shownbelow.

  1. When using the Sn-37Pb solder bath
     Solder bath temperature = 230 °C
     Dipping time = 5 seconds
     Number of times = once
     R-type flux used
  2. When using the Sn-3.0Ag-0.5Cu solder bath
     Solder bath temperature = 245 °C
     Dipping time = 5 seconds
     Number of times = once
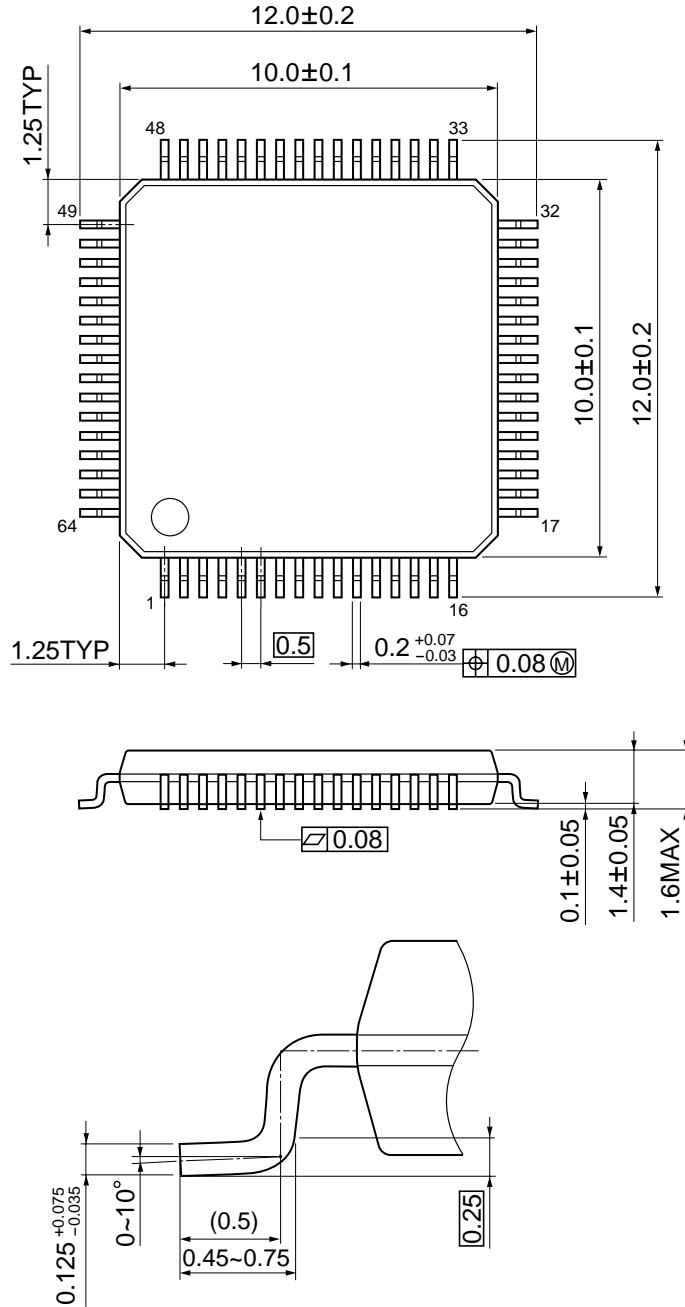
     R-type flux used

     Note: The pass criteron of the above test is as follows: Solderability rate until forming $\geq$ 95 %

- When using the device (oscillator) in places exposed to high electric fields such as cathode-ray tubes, we recommend electrically shielding the package in order to maintain normal operating condition.

20. Package Dimensions

LQFP64-P-1010-0.50E   Rev 02

Unit: mm

12.0±0.2

10.0±0.1

1.25TYP

48   33

49   32

10.0±0.1

12.0±0.2

64   17

1   16

1.25TYP   0.5   0.2 $^{+0.07}_{-0.03}$ ⊕ 0.08 Ⓜ

⌿ 0.08

0.1±0.05

1.4±0.05

1.6MAX

0.125 $^{+0.075}_{-0.035}$

0~10°

(0.5)

0.45~0.75

0.25

Page 230

This is a technical document that describes the operating functions and electrical specifications of the 8-bit microcontroller series TLCS-870/C (LSI).

Toshiba provides a variety of development tools and basic software to enable efficient software development.

These development tools have specifications that support advances in microcomputer hardware (LSI) and can be used extensively. Both the hardware and software are supported continuously with version updates.

The recent advances in CMOS LSI production technology have been phenomenal and microcomputer systems for LSI design are constantly being improved. The products described in this document may also be revised in the future. Be sure to check the latest specifications before using.

Toshiba is developing highly integrated, high-performance microcomputers using advanced MOS production technology and especially well proven CMOS technology.

We are prepared to meet the requests for custom packaging for a variety of application areas.
We are confident that our products can satisfy your application needs now and in the future.