

32-bit RISC Microprocessor – TMPM330 Family**TMPM330FDFG****TMPM330FYFG****TMPM330FWFG****1. Overview and Features**

TMPM330 is a 32-bit RISC microprocessor with an ARM Cortex-M3 processor core.

Features of TMPM330 are as follows:

1.1 Features**(1) ARM Cortex-M3 processor core****1) Improved code efficiency has been realized through the use of Thumb2 instruction**

- New 16-bit Thumb instructions for improved program flow
- New 32-bit Thumb instructions for improved performance
- Auto-switching between 32-bit instruction and 16-bit instruction is executed by compiler.

2) Both high performance and low power consumption have been achieved.**-High performance**

- A 32-bit multiplication ($32 \times 32 = 32$ bit) can be executed with one clock.
- Division takes between 2 and 12 cycles depending on dividend and divisor

-Low power consumption

- Optimized design using a low power consumption library
- Standby function that stops the operation of the processor core

3) High-speed interrupt response suitable for real-time control

- An interruptible long instruction.
- Stack push automatically handled by hardware.

(2) On Chip program memory and data memory

Product name	On chip Flash ROM	On chip RAM
TMPM330FDFG	512Kbyte	32Kbyte
TMPM330FYFG	256Kbyte	16Kbyte
TMPM330FWFG	128Kbyte	8Kbyte

- (3) 16-bit timer : 10 channels
- 16-bit interval timer mode
 - 16-bit event counter mode
 - 16-bit PPG output
 - Input capture function
- (4) Real time clock (RTC) : 1 channel
- Clock (hour, minute and second)
 - Calendar (Month, week, date and leap year)
 - Time correction + or – 30 seconds (by software)
 - Alarm (Alarm output)
 - Alarm interrupt
- (5) Watchdog timer : 1 channel
- 26 cycles of binary counter
 - Watchdog timer out
- (6) General-purpose serial interface : 3 channels
- Either UART mode or synchronous mode can be selected (4byte FIFO equipped)
- (7) Serial bus interface : 3 channels
- Either I²C bus mode or synchronous mode can be selected.
- (8) CEC : 1 channel
- Transmission and reception per 1 byte.
- (9) Remote control signal preprocessor : 2 channels
- Can receive up to 72bit data at a time
- (10) 10-bit A/D converter : 12 channels
- Start by an internal or external timer trigger
 - Fixed channel/scan mode
 - Single/repeat mode
 - AD monitoring 2ch
 - Conversion speed 1.44μsec(@fsys = 32MHz)
- (11) Interrupt source
- Internal: 42 factors...The order of precedence can be set over 7 levels (except the watchdog timer interrupt).
 - External: 8 factors...The order of precedence can be set over 7 levels.
- (12) Input/ output ports
- 79 pins

- (13) Standby mode
 - Standby modes :IDLE, SLOW, SLEEP, STOP
 - Sub clock operation(32.768kHz) :SLOW, SLEEP
- (14) Clock generator
 - On-chip PLL (quadrupled)
 - Clock gear function: The high-speed clock can be divided into 8/8, 4/8, 2/8 or 1/8.
- (15) Endian
 - Little endian
- (16) Maximum operating frequency
 - 32MHz
- (17) Operating voltage range
 - 2.7V~3.6V (with on-chip regulator)
- (18) Temperature range
 - -20~85 degrees (except during Flash writing/ erasing)
 - 0~70 degrees (during Flash writing/ erasing)
- (19) Package
 - P-LQFP100-P-1414-0.5D (14mm × 14mm, 0.5mm pitch)

1.2 Block Diagram

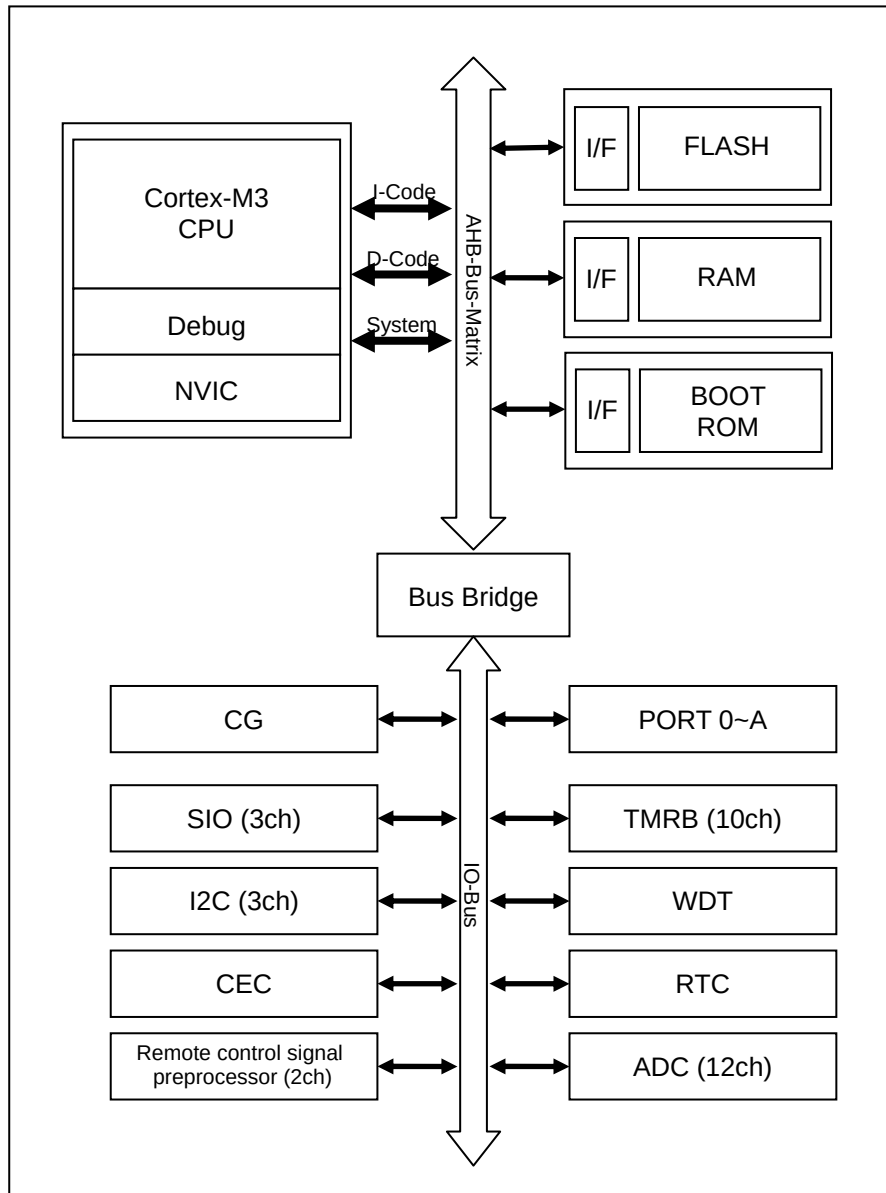


Fig. 1.1 TMPM330 Block Diagram

2. Pin Layout and Pin Functions

This chapter describes the pin layout, pin names and pin functions of TMPM330FDFG/TMPM330FYFG/ TMPM330FWFG.

2.1 Pin Layout (Top view)

Fig. 2-1 shows the pin layout of TMPM330FDFG/TMPM330FYFG/TMPM330FWF.

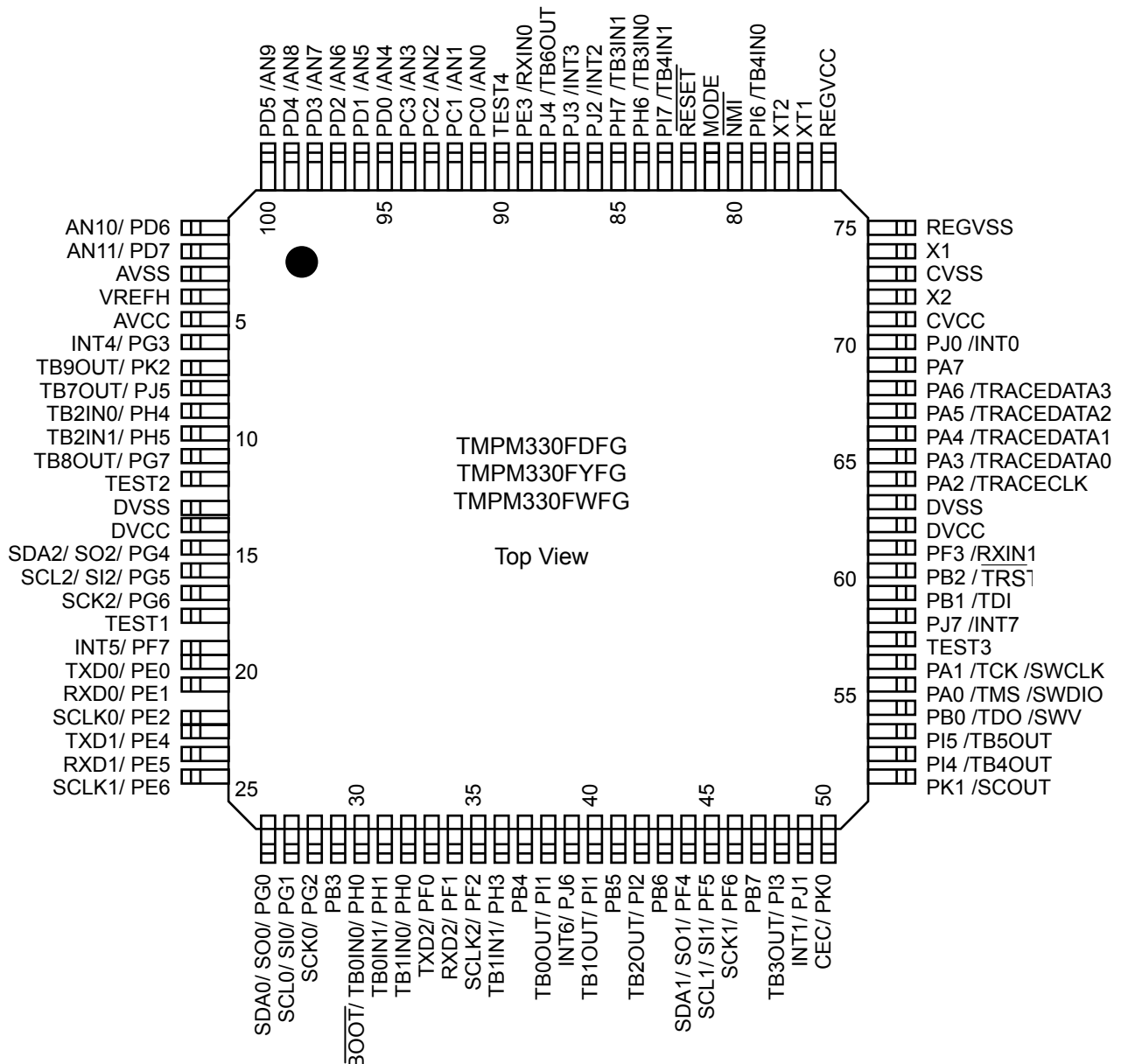


Fig. 2.1 Pin Layout (LQFP100)

Table 2.1 Pin Numbers and Names (1/2)

Pin No.	Pin name	Pin No.	Pin name
1	PD6, AN10	26	PG0, SO0, SDA0
2	PD7, AN11	27	PG1, SI0, SCL0
3	AVSS	28	PG2, SCK0
4	VREFH	29	PB3
5	AVCC	30	PH0, TB0IN0, $\overline{BOO1}$
6	PG3, INT4	31	PH1, TB0IN0
7	PK2, TB9OUT	32	PH2, TB1IN0
8	PJ5, TB7OUT	33	PF0, TXD2
9	PH4, TB2IN0	34	PF1, RXD2
10	PH5, TB2IN1	35	PF2, SCLK2
11	PG7, TB8OUT	36	PH3, TB1IN1
12	TEST2	37	PB4
13	DVSS	38	PI0, TB0OUT
14	DVCC	39	PJ6, INT6
15	PG4, SO2, SDA2	40	PI1, TB1OUT
16	PG5, SI2, SCL2	41	PB5
17	PG6, SCK2	42	PI2, TB2OUT
18	TEST1	43	PB6
19	PF7, INT5	44	PF4, SO1, SDA1
20	PE0, TXD0	45	PF5, SI1, SCL1
21	PE1, RXD0	46	PF6, SCK1
22	PE2, SCLK0	47	PB7
23	PE4, TXD1	48	PI3, TB3OUT
24	PE5, RXD1	49	PJ1, INT1
25	PE6, SCLK1	50	PK0, CEC

Table 2.1 Pin Numbers and Names (2/2)

Pin No.	Pin name	Pin No.	Pin name
51	PK1, SCOUT	76	REGVCC
52	PI4, TB4OUT	77	XT1
53	PI5, TB5OUT	78	XT2
54	PB0, TDO, SWV	79	PI6, TB4IN0
55	PA0, TMS, SWDIO	80	NMI
56	PA1, TCK, SWCLK	81	MODE
57	TEST3	82	RESE
58	PJ7, INT7	83	PI7, TB4IN1
59	PB1, TDI	84	PH6, TB3IN0
60	PB2, TRS ¹	85	PH7, TB3IN1
61	PF3, RXIN1	86	PJ2, INT2
62	DVCC	87	PJ3, INT3
63	DVSS	88	PJ4, TB6OUT
64	PA2, TRACECLK	89	PE3, RXIN0
65	PA3, TRACEDATA0	90	TEST4
66	PA4, TRACEDATA1	91	PC0, AN0
67	PA5, TRACEDATA2	92	PC1, AN1
68	PA6, TRACEDATA3	93	PC2, AN2
69	PA7	94	PC3, AN3
70	PJ0, INT0	95	PD0, AN4
71	CVCC	96	PD1, AN5
72	X2	97	PD2, AN6
73	CVSS	98	PD3, AN7
74	X1	99	PD4, AN8
75	REGVSS	100	PD5, AN9

2.2 Pin names and Functions

Table 2.2 lists the input and output pins of the TMPM330FDFG/TMPM330FYFG/TMPM330FWFG, including alternate pin names and functions for multi-function pins.

Table 2.2 Pin Names and Functions (1/5)

Type	# of Pins	Pin Name	Input/Output	Function	Programmable Pull up/ Pull down	Schmitt trigger	Programmable Open Drain Output
Function	1	PD6 AN10	I I	Input port Analog input	Pull up	—	—
	2	PD7 AN11	I I	Input port Analog input	Pull up	—	—
PS	3	AVSS	I	A/D converter: GND pin (0V) Connect it to a power supply even if the A/D converter is not used.	—	—	—
	4	VREFH	I	Pin for supplying the A/D converter with a reference power supply. Connect this pin to power supply if the A/D converter is not used.	—	—	—
	5	AVCC	I	Pin for supplying the A/D converter with a power supply. Connect it to a power supply even if the A/D converter is not used.	—	—	—
Function	6	PG3 INT4	I/O I	I/O port Interrupt request pin	Pull up	○ w/ noise filter	○
	7	PK2 TB9OUT	I/O O	I/O port Timer B output	Pull up	—	—
	8	PJ5 TB7OUT	I/O I	I/O port Timer B output	Pull up	—	—
	9	PH4 TB2IN0	I/O I	I/O port For inputting the capture trigger of timer B	Pull up	○	—
	10	PH5 TB2IN1	I/O I	I/O port Pin for inputting the capture trigger of timer B	Pull up	○	—
	11	PG7 TB8OUT	I/O O	I/O port Timer B output	Pull up	—	○
Test	12	TEST2	—	TEST pin: Set to OPEN.	—	—	—
PS	13	DVSS	—	GND pin	—	—	—
	14	DVCC	—	Power supply pin	—	—	—
Function	15	PG4 SO2/SD A2	I/O I/O	I/O port Pin for sending and receiving data if the serial bus interface operates in the I2C mode.	Pull up	○	○
	16	PG5 SI2/SCL 2	I/O I/O	I/O port Pin for inputting a clock if the serial bus interface operates in the I2C mode.	Pull up	○	○
	17	PG6 SCK2	I/O I/O	I/O port Pin for inputting and outputting a clock if the serial bus interface operates in the SIO mode.	Pull up	○	○
Test	18	TEST1	—	TEST pin: Set to OPEN.	—	—	—
Function	19	PF7 INT5	I/O I	I/O port Interrupt request pin	Pull up	○ w/ noise filter	○

Table 2.2 Pin Names and Functions (2/5)

Type	# of Pins	Pin Name	Input/Output	Function	Programmable Pull-up/Pull down	Schmitt trigger	Programmable Open Drain Output
Function	20	PE0 TXD0	I/O O	I/O port Sending serial data	Pull up	—	○
	21	PE1 RXD0	I/O I	I/O port Receiving serial data	Pull up	○	○
	22	PE2 SCLK0 CTS0	I/O I I	I/O port Serial clock input/ output Handshake input pin	Pull up	○	○
	24	PE5 RXD1	I/O I	I/O port Receiving serial data	Pull up	○	○
	25	PE6 SCLK1 CTS1	I/O I I	I/O port Serial clock input/ output Handshake input pin	Pull up	○	○
	26	PG0 S00/ SDA0	I/O I/O	I/O port Pin for sending and receiving data if the serial bus interface operates in the I2C mode.	Pull up	○	○
	27	PG1 S10/ SCL0	I/O I/O	I/O port Pin for inputting a clock if the serial bus interface operates in the I2C mode.	Pull up	○	○
	28	PG2 SCK0	I/O I/O	I/O port Pin for inputting and outputting a clock if the serial bus interface operates in the SIO mode.	Pull up	○	○
	29	PB3	I/O	I/O port	Pull up	—	—
Function/ BOOT	30	PH0 TB0IN0	I/O I	I/O port Pin for inputting the capture trigger of timer B	Pull up	○	—
		$\overline{\text{BOOT}}$	I	Pin for setting a single boot mode: This pin goes into single boot mode by sampling "L" at the rise of a reset signal.			
Function	31	PH1 TB0IN1	I/O I	I/O port Pin for inputting the capture trigger of timer B	Pull up	○	—
	32	PH2 TB1IN0	I/O I	I/O port Pin for inputting the capture trigger of timer B	Pull up	○	—
	33	PF0 TXD2	I/O O	I/O port Sending serial data	Pull up	—	○
	34	PF1 RXD2	I/O I	I/O port Receiving serial data	Pull up	○	○
	35	PF2 SCLK2 CTS2	I/O I I	I/O port Serial clock input/ output Handshake input pin	Pull up	○	○
	36	PH3 TB1IN1	I/O I	I/O port Pin for inputting the capture trigger of timer B	Pull up	○	—
	37	PB4	I/O	I/O port	Pull up	—	—
	38	PI0 TB0OUT	I/O O	I/O port Timer B output	Pull up	—	—
	39	PJ6 INT6	I/O I	I/O port Interrupt request pin	Pull up	○ w/ noise filter	○
	40	PI1 TB1OUT	I/O O	I/O port Timer B output	Pull up	—	—
	41	PB5	I/O	I/O port	Pull up	—	—

Table 2.2 Pin Names and Functions (3/5)

Type	# of Pins	Pin Name	Input/Output	Function	Programmable Pull-up/ Pull down	Schmitt trigger	Programmable Open Drain Output
Function	42	PI2 TB2OUT	I/O O	I/O port Timer B output	Pull up	—	—
	43	PB6	I/O	I/O port	Pull up	—	—
	44	PF4 SO1/ SDA1	I/O I/O	I/O port Pin for sending and receiving data if the serial bus interface operates in the I2C mode.	Pull up	○	○
	45	PF5 SI1/ SCL1	I/O I/O	I/O port Pin for inputting a clock if the serial bus interface operates in the I2C mode.	Pull up	○	○
	46	PF6 SCK1	I/O I/O	I/O port Pin for inputting and outputting a clock if the serial bus interface operates in the SIO mode.	Pull up	○	○
	47	PB7	I/O	I/O port	Pull up	—	—
	48	PI3 TB3OUT	I/O O	I/O port Timer B output	Pull up	—	—
	49	PJ1 INT1	I/O I	I/O port Interrupt request pin	Pull up	○ w/ noise filter	○
	50	PK0 CEC	I/O I/O	I/O port CEC pin	—	○	● (Note 4)
	51	PK1 SCOUT	I/O O	I/O port System clock output	Pull up	—	—
	52	PI4 TB4OUT	I/O O	I/O port Timer B output	Pull up	—	—
	53	PI5 TB5OUT	I/O O	I/O port Timer B output	Pull up	—	—
Function/ Debug	54	PB0 TDO/SWV	I/O O	I/O port Debug pin	Pull up	—	—
	55	PA0 TMS/SWDIO	I/O I/O	I/O port Debug pin	Pull up	○	—
	56	PA1 TCK/ SWCLK	I/O I	I/O port Debug pin	Pull up	—	—
Test	57	TEST3	—	TEST pin: Set to OPEN.	—	—	—
Function	58	PJ7 INT7	I/O I	I/O port Interrupt request pin	Pull up	○ w/ noise filter	○
Function/ Debug	59	PB1 TDI	I/O I	I/O port Debug pin	Pull up	—	—
	60	PB2 TRST	I/O I	I/O port Debug pin	Pull up	○	—
Function	61	PF3 RXIN1	I/O I	I/O port Pin for inputting signal to remote controller	Pull up	○	○
PS	62	DVCC	—	Power supply pin	—	—	—
	63	DVSS	—	GND pin	—	—	—

Table 2.2 Pin Names and Functions (4/5)

Type	# of Pins	Pin Name	Input/Output	Function	Programmable Pull-up/Pull down	Schmitt trigger	Programmable Open Drain Output
Function/Debug	64	PA2 TRACECLK	I/O O	I/O port Debug pin	Pull up	—	—
	65	PA3 TRACEDATA0	I/O O	I/O port Debug pin	Pull up	—	—
	66	PA4 TRACEDATA1	I/O O	I/O port Debug pin	Pull up	—	—
	67	PA5 TRACEDATA2	I/O O	I/O port Debug pin	Pull up	—	—
	68	PA6 TRACEDATA3	I/O O	I/O port Debug pin	Pull up	—	—
Function	69	PA7	I/O	I/O port	Pull up		
	70	PJ0 INT0	I/O I	I/O port Interrupt request pin	Pull up	○ w/ noise filter	○
PS	71	CVCC	—	Power supply pin	—	—	—
Clock	72	X2	O	Pin for connecting a high-speed oscillator.	—	—	—
PS	73	CVSS	—	GND pin	—	—	—
Clock	74	X1	I	Pin for connecting a high-speed oscillator.	—	○	—
PS	75	REGVSS	—	GND pin	—	—	—
	76	REGVCC	—	Power supply pin	—	—	—
Clock	77	XT1	I	Pin for connecting a low-speed oscillator.	—	○	—
	78	XT2	O	Pin for connecting a low-speed oscillator.	—	—	—
Function	79	PI6 TB4IN0	I/O I	I/O port Pin for inputting the capture trigger of timer B	Pull up	○	—
	80	NMI	I	Non-maskable interrupt	—	○ w/ noise filter	—
Test	81	MODE	I	Mode pin: Set to DVSS	—	○	—
Function	82	RESET	I	Reset input pin	Set to Pull up	○ w/ noise filter	—
	83	PI7 TB4IN0	I/O I	I/O port Pin for inputting the capture trigger of timer B	Pull up	○	—
	84	PH6 TB3IN0	I/O I	I/O port Pin for inputting the capture trigger of timer B	Pull up	○	—
	85	PH7 TB3IN1	I/O I	I/O port Pin for inputting the capture trigger of timer B	Pull up	○	—
	86	PJ2 INT2	I/O I	I/O port Interrupt request pin	Pull up	○ w/ noise filter	○
	87	PJ3 INT3	I/O I	I/O port Interrupt request pin	Pull up	○ w/ noise filter	○
	88	PJ4 TB6OUT	I/O O	I/O port Timer B output	Pull up	—	—
	89	PE3 RXIN0	I/O I	I/O port For inputting signal to remote controller	Pull up	○	○
Test	90	TEST4	—	TEST pin: Set to OPEN.	—	—	—

Table 2.2 Pin Names and Functions (5/5)

Type	# of Pins	Pin Name	Input/Output	Function	Programmable Pull-up/ Pull down	Schmitt trigger	Programmable Open Drain Output
Function	91	PC0 AN0	I	Input port Analog input	Pull up	—	—
	92	PC1 AN1	I	Input port Analog input	Pull up	—	—
	93	PC2 AN2	I	Input port Analog input	Pull up	—	—
	94	PC3 AN3	I	Input port Analog input	Pull up	—	—
	95	PD0 AN4	I	Input port Analog input	Pull up	—	—
	96	PD1 AN5	I	Input port Analog input	Pull up	—	—
	97	PD2 AN6	I	Input port Analog input	Pull up	—	—
	98	PD3 AN7	I	Input port Analog input	Pull up	—	—
	99	PD4 AN8	I	Input port Analog input	Pull up	—	—
	100	PD5 AN9	I	Input port Analog input	Pull up	—	—

(Note 1) Be sure to set TEST1 through 4 to OPEN.

(Note 2) Be sure to set MODE pin to DVSS.

(Note 3) VREFH and AVSS pins must be connected to GND pin even if the A/D converter is not used.

(Note 4) Nch open drain port.

2.3 Pin Names and Power Supply Pins

Table 2.3 Pin Names and Power Supplies

Pin name	Power supply
PA	DVCC3
PB	DVCC3
PC	DVCC3
PD	DVCC3
PE	DVCC3
PF	AVCC3
PG	DVCC3
PH	DVCC3
PI	DVCC3
PJ	DVCC3
PK	DVCC3
X1, X2	CVCC
XT1, XT2	DVCC3
RESE	DVCC3
NMI	DVCC3
MODE	DVCC3

2.4 Pin Numbers and Power Supply Pins

Table 2.4 Pin Numbers and Power Supplies

Power supply	Pin number	Voltage range
DVCC	14, 62	2.7V~3.6V
AVCC	5	
REGVCC	76	
CVCC	71	

3. Processor Core

The TMPM330 family has a high-performance 32-bit processor core (ARM Cortex-M3 processor core). For information on the operations of this processor core, please refer to the "Cortex-M3 Technical Reference Manual". This chapter describes the functions unique to the TMPM330 series that are not explained in that document.

3.1 Reset Operation

3.1.1 Initial state

The internal circuits, register settings and pin status of the TMPM330 are undefined right after the power-on. The state continues until the $\overline{\text{RESET}}$ pin receives low level input after all the power supply voltage is applied.

3.1.2 Operation

As the precondition, ensure that an internal high-frequency oscillator provides stable oscillation while power supply voltage is in the operating range. To reset the TMPM330, input $\overline{\text{RESET}}$ signal at low level for a minimum duration of 12 system clocks (1.5ms with external 8MHz oscillator).

3.1.3 Cancellation

When the reset is canceled, the system control register and the internal I/O register of the Cortex-M3 processor core are initialized. Note that the PLL multiplication circuit stops after canceling the reset. Therefore, set the PLLSEL register to use PLL multiplication circuit once again.

After the reset exception handling is executed, the program branches off to the interrupt service routine. The address with which the interrupt service routine starts is stored in 0x0000_0004H

(Note 1)	Set the $\overline{\text{RESET}}$ pin to "0" before turning the power on. Cancel the reset after the power supply voltage has stabilized sufficiently within the operating range.
(Note 2)	The reset operation may alter the internal RAM state, but does not alter data in the backup RAM.

4. Memory map

4.1 Memory map

Fig. 4-1 shows the memory map of the TMPM330FDFG.

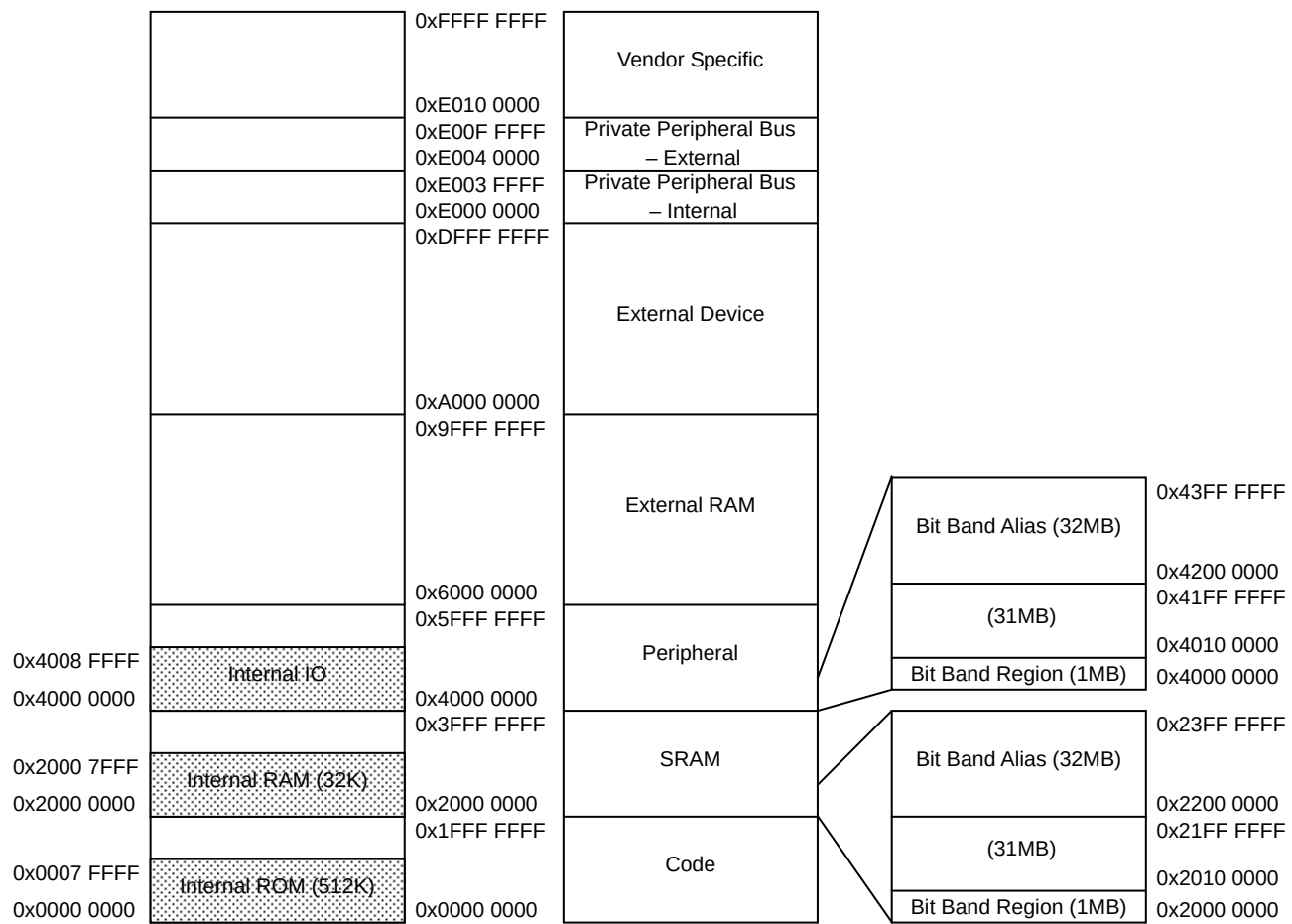


Fig 4.1 Memory map (TMPM330FDFG)

Fig. 4-2 shows the memory map of the TMPM330FYFG.

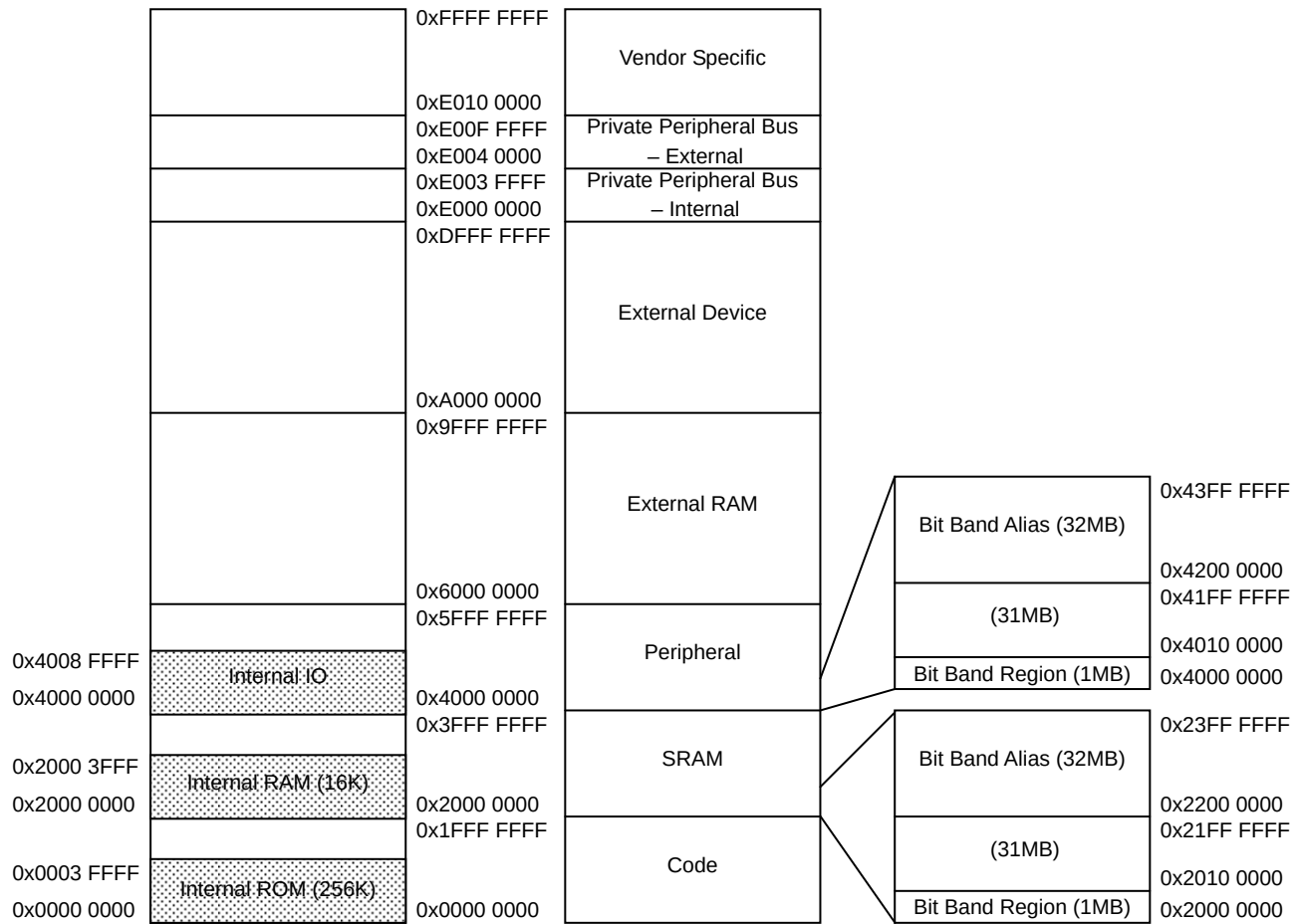


Fig. 4.2 Memory map (TMPM330FYFG)

Fig. 4-3 shows the memory map of the TMPM330FWFG.

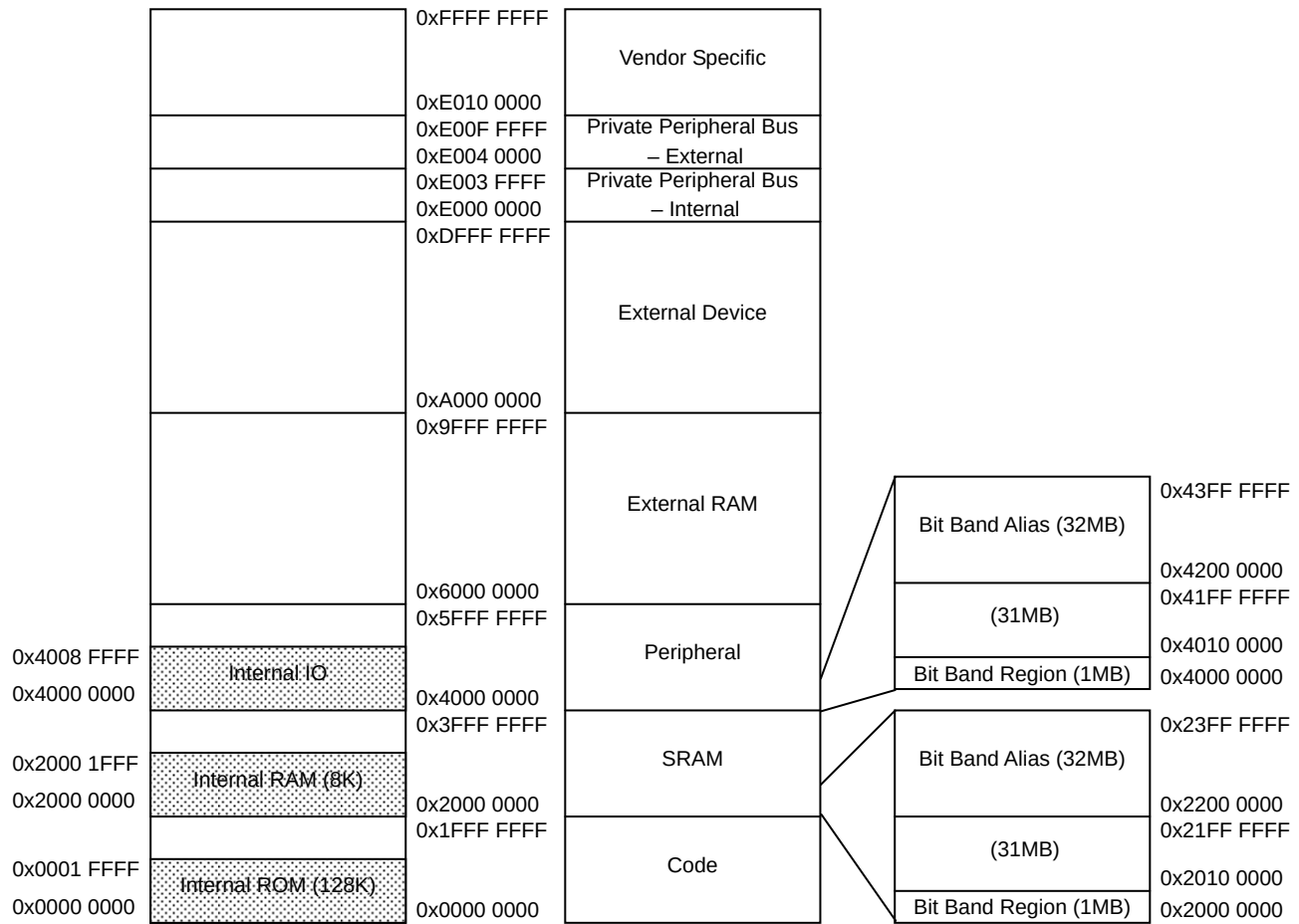


Fig. 4.3 Memory map (TMPM330FWFG)

5. Clock/ Mode Control

5.1 Features

The clock/ mode control block enables to select clock gear, prescaler clock and warm-up of the PLL (including clock multiplication circuit) and oscillator.

The low power consumption mode can reduce power consumption.

5.2 Functions

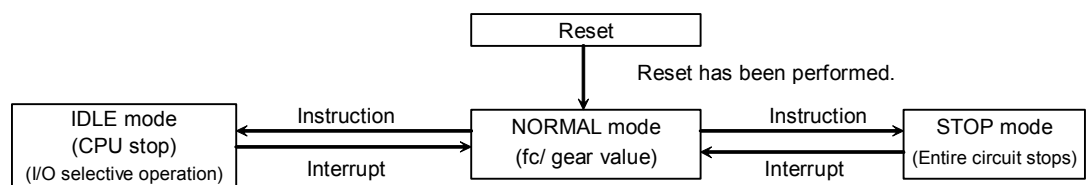
5.2.1 Modes and Mode Transition

The single clock mode is to use only the high-speed clock. The dual clock mode is to use the high-speed and low-speed clocks.

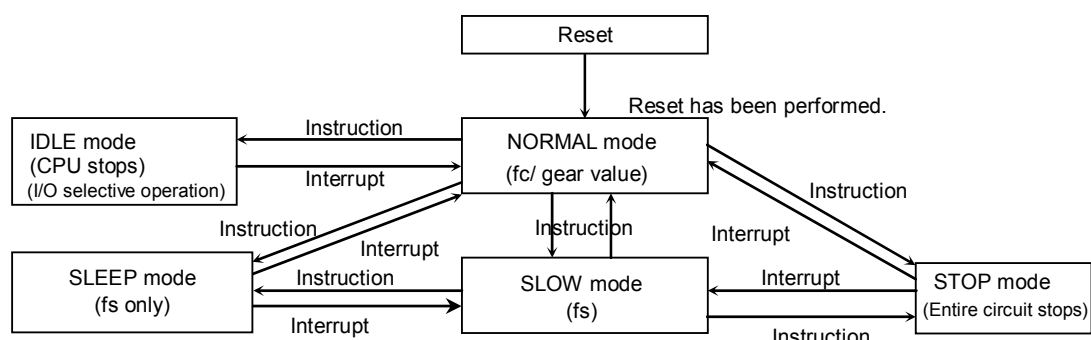
The NORMAL mode and the SLOW mode use the high-speed and low-speed clocks for system clock respectively.

The IDLE, SLEEP and STOP modes can be used as the low power consumption mode that enables to reduce power consumption by halting processor core operation.

Fig. 5-1 shows diagrams of mode transition.



(a) Mode Transition Diagram of Single Clock Mode



(b) Mode Transition Diagram of Dual Clock Mode

Fig. 5-1 Mode Transition

5.2.2 Clock System Block Diagram

Fig. 5-2 shows the clock system diagram.

- High-speed clock gear: f_c , $f_c/2$, $f_c/4$, $f_c/8$
- Prescaler clock: f_{periph} , $f_{periph}/2$, $f_{periph}/4$, $f_{periph}/8$, $f_{periph}/16$, $f_{periph}/32$

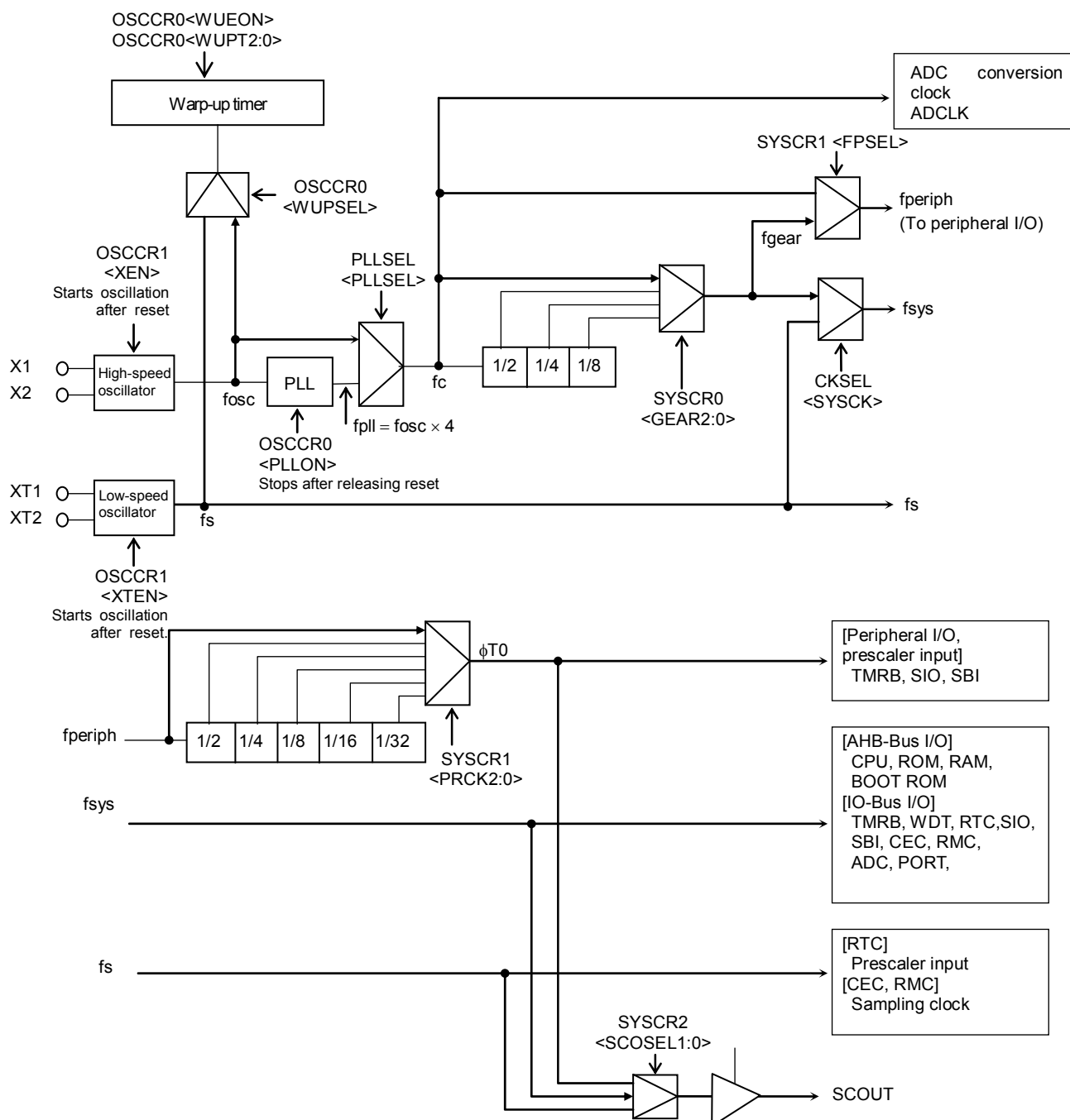


Fig. 5-3 Clock Block Diagram

(Note) The input clocks to selector shown with an arrow are set as default after reset.

5.3 Registers

5.3.1 Register List

Table 5-1 shows registers and addresses of the clock generator.

Table 5-1 Registers of Clock Generator

Register name		Address
System control register 0	SYSCR0	4004_0200H
System control register 1	SYSCR1	4004_0201H
System control register 2	SYSCR2	4004_0202H
Oscillation control register 0	OSCCR0	4004_0204H
Oscillation control register 1	OSCCR1	4004_0205H
Standby control register 0	STBYCR0	4004_0208H
Standby control register 1	STBYCR1	4004_0209H
Standby control register 2	STBYCR2	4004_020AH
PLL selection register	PLLSEL	4004_020CH
System clock selection register	CKSEL	4004_0210H

5.3.2 Detailed Description of Registers

5.3.2.1 System Control Register

		7	6	5	4	3	2	1	0
SYSCR0	Bit symbol	-	-	-	-	-	GEAR2	GEAR1	GEAR0
	Read/Write	R					R/W	R/W	R/W
	After reset	0	0	0	0	0	0	0	0
	Function	"0" is read.					High-speed clock (fc) gear 000: fc 100: fc/2 001: reserved 101: fc/4 010: reserved 110: fc/8 011: reserved 111: reserved		
		15	14	13	12	11	10	9	8
SYSCR1	Bit symbol	-	-	-	FPSEL	-	PRCK2	PRCK1	PRCK0
	Read/Write	R			R/W	R	R/W	R/W	R/W
	After reset	0	0	0	0	0	0	0	0
	Function	"0" is read.			fperiph 0:fgear 1:fc	"0" is read.	Prescaler clock 000: fperiph 100: fperiph/16 001: fperiph/2 101: fperiph/32 010: fperiph/4 110: Reserved 011: fperiph/8 111: Reserved		
		23	22	21	20	19	18	17	16
SYSCR2	Bit symbol	-	-	-	-	-	--	SCOSEL1	SCOSEL0
	Read/Write	R						R/W	R/W
	After reset	0	0	0	0	0	0	0	1
	Function	"0" is read.						SCOUT output 00: fs 01: Reserved 10: fsys 11: φT0	
		31	30	29	28	27	26	25	24
	Bit symbol	-	-	-	-	-	-	-	-
	Read/Write	R							
	After reset	0	0	0	0	0	0	0	0
	Function	"0" is read.							

<Bit 2:0><GEAR 2:0> : Specifies the high-speed clock (fc) gear.

<Bit 10:8><PRCK 2:0> : Specifies the prescaler clock to peripheral I/O.

<Bit 12><FPSEL> : Specifies the source clock to fperiph.

<Bit 17:16><SCOSEL1:0> : Enables to output the specified clock from SCOUT pin.

5.3.2.2 Oscillation Control Register

OSCCR		7	6	5	4	3	2	1	0
	Bit symbol	-	WUPT2	WUPT1	WUPT0	WUPSEL	PLLON	WUEF	WUEON
	Read/Write	R	R/W	R/W	R/W	R/W	R/W	R	W
	After reset	0	0	0	1	0	0	0	0
OSCCR	Function	"0" is read.	Selection of warm-up time for oscillator		Warm-up counter		PLL operation	Status of Warm-up timer for oscillator	Operation of warm-up timer for oscillator
			X1	XT1					
			000: No warm-up	000: No warm-up		0: X1	0: Stop	0: warm-up completed	0: don't care
			001: 2 ¹⁰ /input freq.	001: 2 ⁹ /input freq.		1: XT1	1: Oscillation	1: Warm-up in operation	1: starting warm-up
OSCCR		15	14	13	12	11	10	9	8
	Bit symbol	-	-	-	-	-	-	XTEN	XEN
	Read/Write	R		R/W		R		R/W	R/W
	After reset	0	0	0	0	0	0	1	1
OSCCR	Function	"0" is read.		Write "0".		"0" is read.		Low-speed oscillator	High-speed oscillator
								0: Stop	0: Stop
								1: Oscillation	1: Oscillation
OSCCR		23	22	21	20	19	18	17	16
	Bit symbol	-	-	-	-	-	-	-	-
	Read/Write	R							
	After reset	0	0	0	0	0	0	0	0
OSCCR	Function	"0" is read.							
OSCCR		31	30	29	28	27	26	25	24
	Bit symbol	-	-	-	-	-	-	-	-
	Read/Write	R							
	After reset	0	0	0	0	0	0	0	0
OSCCR	Function	"0" is read.							

<Bit 0><WUEON> : Enables to start the warm-up timer.

<Bit 1><WUEF> : Enables to monitor the status of the warm-up timer.

<Bit 2><PLLON> : Specifies operation of the PLL.
It stops after reset. Resetting the bit is required.

<Bit 3><WUPSEL> : Specifies the oscillator to warm-up.

<Bit 6:4><WUPT2:0> : Specifies time of the warm-up timer for oscillator.

<Bit 8><XEN> : Specifies operation of the high-speed oscillator.

<Bit 9><XTEN> : Specifies operation of the low-speed oscillator.

5.3.2.3 Standby Control Register

STBYCR0		7	6	5	4	3	2	1	0
	Bit symbol	-	-	-	-	-	STBY2	STBY1	STBY0
	Read/Write	R					R/W	R/W	R/W
	After reset	0	0	0	0	0	0	1	1
	Function	"0" is read.					Low power consumption mode 000: Reserved 001: STOP 010: SLEEP 011: IDLE 100: Reserved 101: Reserved 110: Reserved 111: Reserved		
STBYCR1		15	14	13	12	11	10	9	8
	Bit symbol	-	-	-	-	-	-	RXTEN	RXEN
	Read/Write	R						R/W	R/W
	After reset	0	0	0	0	0	0	0	1
	Function	"0" is read.						Low-speed oscillator after releasing STOP mode 0: Stop 1: Oscillation	High-speed oscillator after releasing STOP mode 0: Stop 1: Oscillation
STBYCR2		23	22	21	20	19	18	17	16
	Bit symbol	-	-	-	-	-	-	-	DRVE
	Read/Write	R						R/W	R/W
	After reset	0	0	0	0	0	0	0	0
	Function	"0" is read.						Write "0".	Pin status in STOP mode 0: Active 1: Inactive
		31	30	29	28	27	26	25	24
	Bit symbol	-	-	-	-	-	-	-	-
	Read/Write	R							
	After reset	0	0	0	0	0	0	0	0
	Function	"0" is read.							

<Bit 2:0><STBY2:0> : Specifies the low power consumption mode.

<Bit 8><RXEN> : Specifies the high-speed oscillator operation after releasing the STOP mode.

<Bit 9><RXTEN> : Specifies the low-speed oscillator operation after releasing the STOP mode.

<Bit 16><DRVE> : Specifies the pin status in the STOP mode.

5.3.2.4 PLL Selection Register

PLLSEL		7	6	5	4	3	2	1	0
	Bit symbol	-	-	-	-	-	-	-	PLLSEL
	Read/Write	R							R/W
	After reset	0	0	0	0	0	0	0	0
	Function	"0" is read.							Use of PLL 0: Disuse. X1 selected 1: Use
		15	14	13	12	11	10	9	8
	Bit symbol	-	-	-	-	-	-	-	-
	Read/Write	R							
	After reset	0	0	0	0	0	0	0	0
	Function	"0" is read.							
		23	22	21	20	19	18	17	16
	Bit symbol	-	-	-	-	-	-	-	-
	Read/Write	R							
	After reset	0	0	0	0	0	0	0	0
	Function	"0" is read.							
		31	30	29	28	27	26	25	24
	Bit symbol	-	-	-	-	-	-	-	-
	Read/Write	R							
	After reset	0	0	0	0	0	0	0	0
	Function	"0" is read.							

<Bit 0><PLLSEL> : Specifies use or disuse of the clock multiplied by the PLL.

"X1" is automatically set after reset. Resetting is required when using the PLL.

5.3.2.5 System Clock Selection Register

CKSEL		7	6	5	4	3	2	1	0
	Bit symbol	-	-	-	-	-	-	SYSCK	SYSCK FLG
	Read/Write	R						R/W	R
	After reset	0	0	0	0	0	0	0	0
	Function	"0" is read.						System clock 0: High-speed (fc) 1: Low-speed (fs)	System clock status 0: High-speed (fc) 1: Low-speed (fs) Stable oscillation identical with <SYSCK> value.
		15	14	13	12	11	10	9	8
	Bit symbol	-	-	-	-	-	-	-	-
	Read/Write	R							
	After reset	0	0	0	0	0	0	0	0
	Function	"0" is read.							
		23	22	21	20	19	18	17	16
	Bit symbol	-	-	-	-	-	-	-	-
	Read/Write	R							
	After reset	0	0	0	0	0	0	0	0
	Function	"0" is read.							
		31	30	29	28	27	26	25	24
	Bit symbol	-	-	-	-	-	-	-	-
	Read/Write	R							
	After reset	0	0	0	0	0	0	0	0
	Function	"0" is read.							

<Bit 0><SYSCKFLG>

: Shows the status of the system clock.
Switching the oscillator with <SYSCK> generates time lag to complete. If the output of the oscillator specified in <SYSCK> is read out by <SYSCKFLG>, the switching has been completed.

<Bit 1><SYSCK>

: Enables to specify the system clock. Setting OSCCR1<XEN> and <XTEN> to "1" in advance is required.

5.4 System Clock Control

5.4.1 Initial Values after Reset

Reset initializes the system clock controller as follows.

High-speed oscillator	: ON (oscillating)
Low-speed oscillator	: ON (oscillating)
PLL (phase locked loop circuit)	: OFF (stop)
High-speed clock gear	: fc (no frequency dividing)

For example, when a 10-MHz oscillator is connected to the X1 or X2 pin, fsys becomes 10MHz after reset.

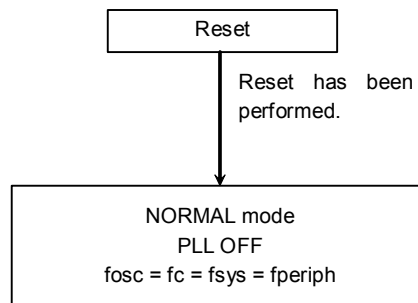


Fig. 5-4 Default State of System Clock

Fosc	: Clock frequency to be input via the X1 and X2 pins
Fc	: High clock frequency
Fsys	: System clock frequency
Fperiph	: Clock frequency selected by SYSCR1<FPSEL2:0> (Clock to be input to the peripheral I/O prescaler)

5.4.2 Main System Clock

- Allows for oscillator connection or external clock input.
- Clock gear: 1/1, 1/2, 1/4, 1/8 (after reset: 1/1)
- High input frequency: 8MHz~10MHz

Table 5-2 Range of High-frequency

External oscillator	Min. operating freq.	Max. operating freq.	After reset (PLL=OFF, CG=1/1)	Clock gear (CG) @PLL=ON				Clock gear (CG) @PLL=OFF			
				1/1	1/2	1/4	1/8	1/1	1/2	1/4	1/8
8MHz	1MHz	40MHz	8	32	16	8	4	8	4	2	1
10MHz			10	40	20	10	5	10	5	2.5	1.25

* PLL=ON/OFF setting: available in OSCCR0<PLLON>

Clock gear setting: available in SYSCR0<GEAR2:0>

- Low input frequency

Table 5-3 Range of Low Frequency

Input Frequency Range	Maximum Operating Frequency	Minimum Operating Frequency
30 ~ 34(kHz)	34 kHz	30 kHz

(Note 1) Switching of clock gear is executed when a value is written to the SYSCR0<GEAR2:0> register. There are cases where switching does not occur immediately after the change in the register setting but the original clock gear is used for execution of instructions. If it is necessary to use the new clock for execution of the instructions following to the clock gear switching instruction, insert a dummy instruction (to execute a write cycle) and DSB instruction.

(Note 2) To use the clock gear, ensure that you make the time setting such that ϕT_n of the prescaler output from each block in the peripheral I/O is calibrated to $\phi T_n < f_{sys}/2$ (ϕT_n becomes slower than $f_{sys}/2$). Do not switch the clock gear during operation of the timer counter or other peripheral I/O.

5.4.3 Oscillation Stabilization Time (Warm-up Function)

The warm-up timer is provided to confirm the oscillation stability of the oscillator when it is connected to the oscillator connection pin. The warm-up time can be selected by setting the OSCCR0<WUPT2:0>. The OSCCR0<WUEON><WUEF> is used to confirm the start and completion of warm-up through software (instruction). After the completion of warm-up is confirmed, switch the system clock by setting the CKSEL<SYSCK>.

When clock switching occurs, the current system clock can be checked by monitoring the CKSEL<SYSCKFLG>.

The warm-up function is used to make the oscillation from the internal oscillator stable in returning from STOP/SLEEP mode. In this case, an interrupt for returning from the low power consumption mode triggers the automatic timer count. After the specified time is reached, the system clock is output and the CPU starts operation.

Table 5-4 shows the warm-up time when switching occurs.

Table 5-4 Warm-up Time (fosc=10 MHz, fs=32.768 kHz)

Warm-up time options OSCCR0<WUPT 2:0>	High-speed clock (fosc) OSCCR0<WUPSEL>="0"		Low-speed clock (fs) OSCCR0<WUPSEL>="1"	
000	—	Without warm-up	—	Without warm-up
001	2 ¹⁰ /input frequency	102.4 (μs)	2 ⁶ /input frequency	1.953 (ms)
010	2 ¹¹ /input frequency	204.8 (μs)	2 ⁷ /input frequency	3.906 (ms)
011	2 ¹² /input frequency	409.6 (μs)	2 ⁸ /input frequency	7.813 (ms)
100	2 ¹³ /input frequency	819.2 (μs)	2 ¹⁵ /input frequency	1.0 (s)
101	2 ¹⁴ /input frequency	1.638 (ms)	2 ¹⁶ /input frequency	2.0 (s)
110	2 ¹⁵ /input frequency	3.277 (ms)	2 ¹⁷ /input frequency	4.0 (s)
111	2 ¹⁶ /input frequency	6.554 (ms)	2 ¹⁸ /input frequency	8.0 (s)

(Note 1) Warm-up is not required when an oscillator module is used for the clock and providing stable oscillation.

(Note 2) The warm-up timer operates according to the oscillation clock, and it may contain errors if there is any fluctuation in the oscillation frequency. Therefore, the warm-up time should be taken as approximate time.

<Example 1> Transition from the NORMAL mode to the SLOW mode

OSCCR0<WUPT2:0>="xx" : Select the warm-up time

OSCCR1<XTEN>="1" : Enable the low-speed oscillation (fs)

OSCCR0<WUEON>="1" : Start the warm-up timer.

OSCCR0<WUEF>Read : Wait until the state becomes "0" (warm-up is finished)

CKSEL<SYSCK>="1" : Switch the system clock to low speed (fs)

CKSEL<SYSCKFLG> Read: Confirm that the current state is "1" (the current system clock is fs)

OSCCR1<XEN>="0" : Disable the high-speed oscillation (fosc)

<Example 2> Transition from the SLOW mode to the NORMAL mode

OSCCR0<WUPT2:0>="xx" : Select the warm-up time

OSCCR1<XEN>="1" : Enable the high-speed oscillation (fosc)

OSCCR0<WUEF>="1" : Start the warm-up timer.

OSCCR0<WUEF> Read : Wait until the state becomes "0" (warm-up is finished).

CKSEL<SYSCK>="0" : Switch the system clock to high speed (fgear)

CKSEL<SYSCKFLG> Read: Confirm that the current state is "0" (the current system clock is fgear)

OSCCR1<XTEN>="0" : Disable the low-speed oscillation (fs)

(Note) When switching the system clock, ensure that the switching has been completed by reading the SYSCR1<SYSCKFLG>.

5.4.4 System Clock Pin Output Function

The TMPM330 enables to output the system clock from a pin. The PK1/SCOUT pin can output the low speed clock f_s , the system clock f_{sys} and the prescaler input clock for peripheral I/O $\phi T0$. By setting the port K registers, the PKCR<PK1C> and PKFR1<PK1F1> to "1", the PK1/SCOUT pin (pin number 51) becomes the SCOUT output pin. The output clock is selected by setting the SYSCR3<SCOSEL1:0>.

Table 5-5 shows the pin states in each mode when the SCOUT pin is set to the SCOUT output.

Table 5-5 Scout Output State in Each Mode

SCOUT selection SYSCR2	Mode	NORMAL	SLOW	Low power consumption mode		
				IDLE	SLEEP	STOP
<SCOSEL1:0> = "00"	Output the fs clock.					
<SCOSEL1:0> = "10"	Output the fsys clock.					
<SCOSEL1:0> = "11"	Output the ΦT0 clock.					
				Fixed to "0" or "1".		

(Note) The phase difference (AC timing) between the system clock output by the SCOUT and the internal clock is not guaranteed.

5.5 Prescaler Clock Control

Each internal I/O (TMRB0-9, SIO0-2 and SBI0-2) has a prescaler for dividing a clock. As the clock $\phi T0$ to be input to each prescaler, the "fperiph" clock specified in the SYSCR1<FPSEL> can be divided according to the setting in the SYSCR1<PRCK2:0>. After the controller is reset, fperiph/1 is selected as $\phi T0$.

5.6 Clock Multiplication Circuit

This circuit, which is included in the PLL, outputs the f_{pll} clock that is quadruple of the high-speed oscillator output clock, f_{osc} . This lowers the oscillator input frequency while increasing the internal clock speed.

5.7 Operation Modes

Two operation modes, NORMAL and SLOW, are available. The features of each mode are described below.

5.7.1 NORMAL Mode

This mode is to operate the CPU core and the peripheral hardware by using the high-speed clock. It is shifted to the NORMAL mode after reset. The dual clock operation that uses the low-speed clock as well is also available.

5.7.2 SLOW Mode

This mode is to operate the CPU core and the peripheral hardware by using the low-speed clock with high-speed clock stopped. This mode allows the I/O port, real time clock (RTC), CEC and RMC (remote control signal preprocessor) functions to operate.

(Note) Be sure to stop the peripheral functions except for the CPU, RTC, I/O port, CEC and RMC before the transition to SLOW mode.

5.8 Low Power Consumption Mode

The TMPM330 has three low power consumption modes: IDLE, SLEEP and STOP. To shift to the low power consumption mode, specify the mode in the system control register STBYCR0<STBY2:0> and execute the WFI (Wait For Interrupt) instruction. In this case, execute reset or generate the interrupt to release the mode. Releasing by the interrupt requires settings in advance. See chapter 6 for details.

(Note) Transition to the low power consumption mode by executing the WFE (Wait For Event) instruction is prohibited. The TMPM330 does not offer any event for releasing the low power consumption mode.

The features of each mode are described below.

5.8.1 IDLE Mode

Only the CPU is stopped in this mode.

The internal I/O has one bit of the ON/OFF setting register for operation in the IDLE mode in the register of each module. This enables operation settings for the IDLE mode. When the internal I/O has been set not to operate in the IDLE mode, it stops operation and holds the state when the system enters the IDLE mode.

Table 5-6 shows a list of the IDLE setting registers.

Table 5-6 Internal I/O setting registers for the IDLE mode

Internal I/O	IDLE mode setting register
TMRB0~9	TBxRUN<I2TBx>
SIO0~2	SCxMOD1<I2Sx>
I2C/SIO (SBI0~2)	SBIBRx<I2SBIx>
CEC	CECEN<I2CEC>
RMC	RMCEN<I2RMC>
A/D converter	ADMOD1<I2AD>
WDT	WDMOD<I2WDT>

5.8.2 SLEEP Mode

The internal low-speed oscillator, real time clock, CEC (only for reception) and RMC operate. By releasing the SLEEP mode, the device returns to the preceding mode of the SLEEP mode and starts operation.

5.8.3 STOP Mode

All the internal circuits including the internal oscillator are brought to a stop.

By releasing the STOP mode, the device returns to the preceding mode of the STOP mode and starts operation.

The STOP mode enables to select the pin status by setting the STBYCR2<DRVE>. Table 5-7 shows the pin status in the STOP mode.

Table 5-7 Pin status of SYSCR2<DRVE> in STOP mode

Pin name		Input/ output	<DRVE>=0	<DRVE>=1
Port	Bit			
PA	0 (PAFR<0>=0) (general-purpose port)	Input enabled Output enabled	— —	Input Output
	0 (PAFR<0>=1) (TMS/SWDIO)	Output enabled (PACR<0>=1)	Output	Output
	1-7	Input enabled (PEIE<x>=1) Output enabled (PECR<x>=1)	— —	Input Output
PB	0 (PBFR<0>=0) (general-purpose port)	Input enabled Output enabled	— —	Input Output
	0 (PBFR<0>=1) (TDO/SWV)	Output enabled (PACR<0>=1)	Output	Output
	1-7	Input enabled (PEIE<x>=1) Output enabled (PECR<x>=1)	— —	Input Output
PC	0-4	Input enabled (PCIE<x>=1)	—	Input
PD	0-7	Input enabled (PDIE<x>=1)	—	Input
PE	0-6	Input enabled (PEIE<x>=1) Output enabled (PECR<x>=1)	— —	Input Output
PF	0-6	Input enabled (PFIE<x>=1) Output enabled (PFCR<x>=1)	— —	Input Output
	7	PFFR<7>=0 (general-purpose port)	—	Input
		PFFR<7>=1 (INT7)	—	Output
PG	0-2,4-7	Input enabled (PGIE<x>=1) Output enabled (PGCR<x>=1)	— —	Input Output
	3	PGFR<3>=0 (general-purpose port)	—	Input
		PGFR<3>=1 (INT4)	—	Output
PH	0-7	Input enabled (PHIE<x>=1) Output enabled (PHCR<x>=1)	— —	Input Output
PI	0-7	Input enabled (PIIE<x>=1) Output enabled (PICR<x>=1)	— —	Input Output
PJ	4-5	Input enabled (PJIE<x>=1) Output enabled (PJCR<x>=1)	— —	Input Output
	0-3, 6,7	PGFR<x>=0 (general-purpose port)	—	Input
		PGFR<x>=1 (INT0-3,6,7)	—	Output
PK	0-2	Input enabled (PKIE<x>=1) Output enabled (PKCR<x>=1)	— —	Input Output
/RESET		Input pin	Input	Input
/NMI		Input pin	Input	Input
X1, XT1		Input pin	—	—
X2, XT2		Output pin	"H" level Output	"H" level Output
MODE		Input pin	Input	Input

—: Indicates that the input or output is disabled.

Input: The input gate is active. To prevent the input pin from floating, fix the input voltage to the "L" or "H" level.

Output: The pin is in the output state.

5.8.4 Low power Consumption Mode Setting

The low power consumption mode is specified by the setting of the standby control register STBYCR<STBY2:0>.

Table 5-8 shows the mode setting in the <STBY2:0>.

Table 5-8 Low power consumption mode setting

Mode	STBYCR0 <STBY2:0>
RESERVED	000
STOP	001
SLEEP	010
IDLE	011

5.8.5 Operational State in Each Mode

Table 5-9 show the operational state in each mode.

Table 5-9 Operational State in Each Mode

Block	NORMAL	SLOW	IDLE	SLEEP	STOP
Processor core	○	○	×	×	×
I/O port	○	○	○	×	×
ADC	○	×	ON/OFF selectable for each module	×	×
SIO	○	×		×	×
I2C	○	×		×	×
TMRB	○	×		×	×
WDT	○	×		×	×
CEC	○	○	○	○	×
RMC	○	○	○	○	×
RTC	○	○	○	○	×
CG	○	○	○	○	×
PLL	○	×	○	×	×
High-speed oscillator (fc)	○	*(Note)	○	×	×
Low-speed oscillator (fs)	○	○	○	○	×

○: Operation ×: Stop

(Note) When the system enters the SLOW mode, the high-speed oscillator must be stopped by setting the OSCCR1<XEN>.

5.8.6 Releasing the Low power Consumption Mode

The low power consumption mode can be released by an interrupt request or by reset. The release source that can be used is determined by the low power consumption mode selected. Details are shown in Table 5-10.

Table 5-10 Release Source in Each Mode

Low power consumption mode			IDLE (programmable)	SLEEP	STOP
Release source	Interrupt	INTWDT	○	×	×
		INT0~7	○	○ (Note 2)	○ (Note 1)
		INTRTC	○	○ (Note 2)	×
		INTTB0~9	○	×	×
		INTRX0~2, INTTX0~2	○	×	×
		INTSBI0~2	○	×	×
		INTCECRX, INTCECTX	○	○ (Note 2)	×
		INTRMCRX0,1	○	○ (Note 2)	×
		INTAD/INTADHP/ INTADM	○	×	×
	RESET	○	○	○	

○: Starts the interrupt handling after the mode is released. (The reset initializes the LSI).

×: Unavailable

(Note 1) The standby mode is released after the warm-up time has elapsed.

(Note 2) To return to the **NORMAL**, release after the warm-up time has elapsed.

(Note 3) To release the low power consumption mode by using the level mode interrupt in the interruptible state, keep the level until the interrupt handling is started. Changing the level before then will prevent the interrupt handling from starting properly.

(Note 4) For shifting to the low power consumption mode, set the CPU to prohibit all the interrupts other than the release source. If not, releasing may be executed by an unspecified interrupt.

- Release by interrupt request

To release the low power consumption mode by an interrupt, the CPU must be set in advance to detect the interrupt. In addition to the setting in the CPU, the clock generator must be set to detect the interrupt to be used to release the SLEEP and STOP modes.

- Release by reset

Any low power consumption modes can be released by reset.

Note that releasing of the STOP mode requires sufficient reset time to allow the oscillator operation to become stable. (Time required for stable oscillation + 500μs and more).

Refer to "Chapter 6. Interrupts" for details.

5.8.7 Warm-up

When the STOP mode is released, the system clock output is started after the elapse of warm-up time at the warm-up counter to allow the internal oscillators to stabilize. The same occurs in the transition to the NORMAL mode after releasing the SLEEP mode.

It is necessary to set the warm-up time in the OSCCR0<WUPT2:0> before executing the instruction to enter the STOP/ SLEEP mode.

Table 5-11 shows whether the warm-up setting of each mode transition is required or not.

Table 5-11 Warm-up setting in mode transition

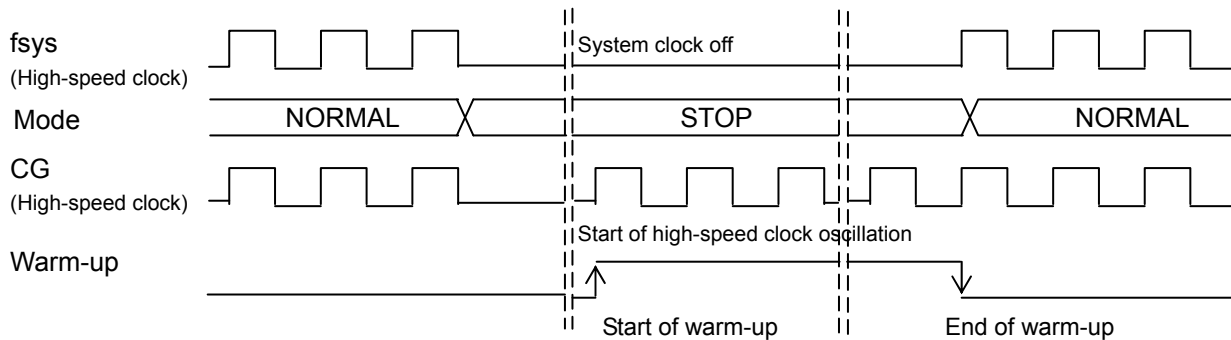
Mode transition	Warm-up setting
NORMAL→IDLE	Not required
NORMAL→SLEEP	Not required
NORMAL→SLOW	Not required
NORMAL→STOP	Not required
IDLE→NORMAL	Not required
SLEEP→NORMAL	Auto-warm-up
SLEEP→SLOW	Not required
SLOW→NORMAL	Required (note)
SLOW→SLEEP	Not required
SLOW→STOP	Not required
STOP→NORMAL	Auto-warm-up
STOP→SLOW	Auto-warm-up

(Note) The warm-up must be activated by software.

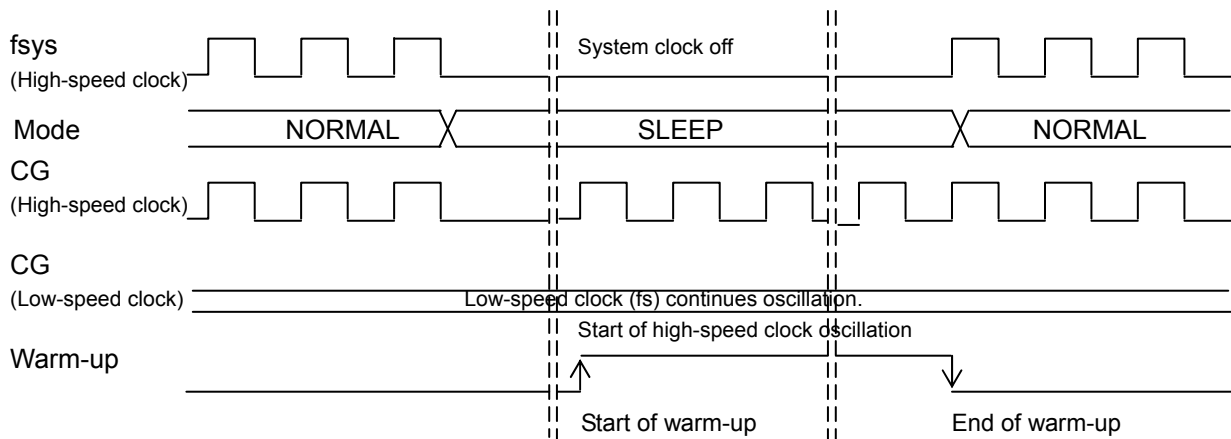
5.8.8 Clock Operations in Mode Transition

The clock operations in mode transition are described in the following sections 5.8.8.1 to 5.8.8.4.

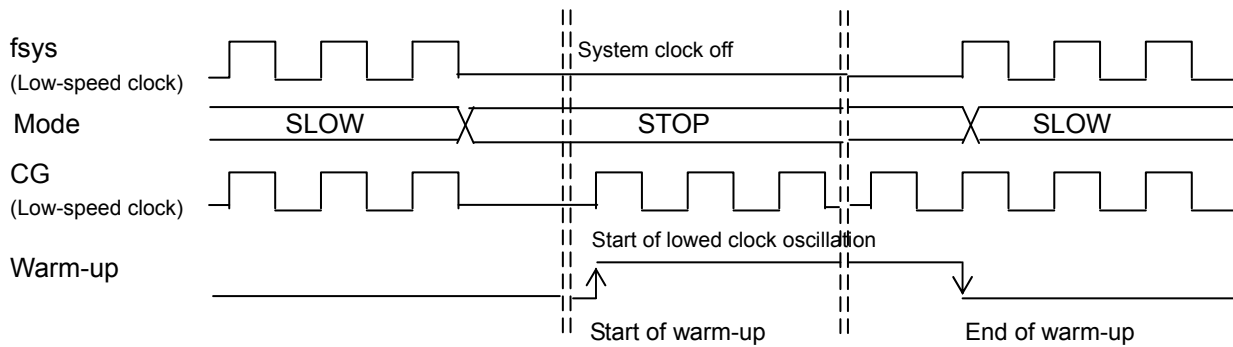
5.8.8.1 Transition of operation modes: NORMAL→STOP→NORMAL



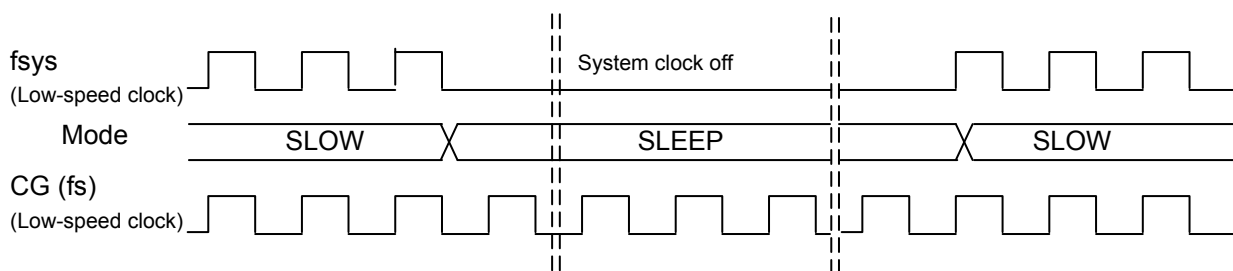
5.8.8.2 Transition of operation modes: NORMAL→SLEEP→NORMAL



5.8.8.3 Transition of operation modes: SLOW→STOP→SLOW



5.8.8.4 Transition of operation modes: SLOW→SLEEP→SLOW



(Note) The low-speed clock (fs) continues oscillation. There is no need to make a warm-up setting.

6 Interrupts

This chapter describes routes, factors and required settings of interrupts.

Interrupts have close relation to the CPU core architecture. Refer to “Cortex-M3 Technical Reference Manual” if needed.

6.1 Overview

The CPU is notified of interrupts by each signal of interrupt factor.

It sets priority on the interrupts and handles an interrupt request with the highest priority.

The CPU is notified of the interrupt factor, which is used for clearing the standby modes, via a clock generator. Therefore setting of the clock generator is required.

6.2 Interrupt Factors

6.2.1 Interrupt route

Fig. 6-1 shows an interrupt request route.

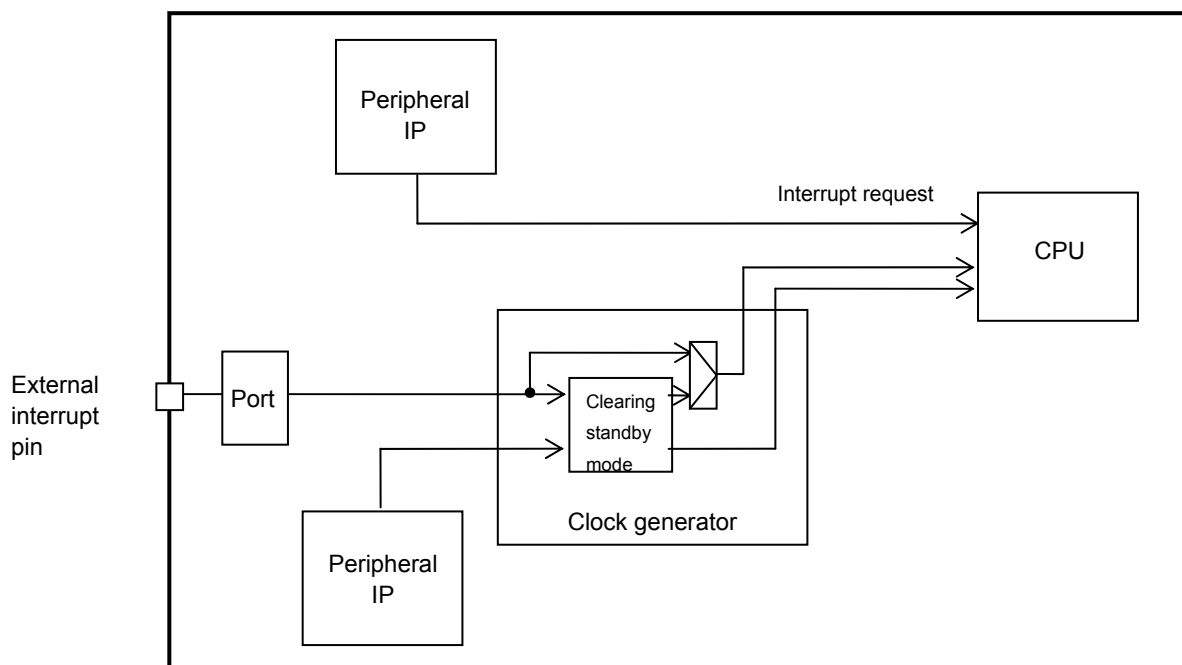


Fig. 6-1 Interrupt Route

6.2.2 Generation

An interrupt signal is sent from an external pin assigned as the interrupt factor or a peripheral IP.

- From external pin

Set the port control register so that the external pin can perform as an interrupt function pin.

- From peripheral IP

Set the peripheral IP to make it possible to output the interrupt.

See chapters of relevant IP for details.

6.2.3 Transmission

An interrupt signal from an external pin or a peripheral IP is directly sent to the CPU unless it is used to clear the standby mode.

An interrupt that can be used as a factor to clear the standby mode is transmitted to the CPU via the clock generator. In this case, set the clock generator in advance. An interrupt from an external pin can be used without setting the clock generator if it does not function as the clearing factor.

6.2.4 List of Interrupt Factors

Table 6-1 shows the list of interrupt factors.

Table 6-1 List of Hardware Interrupt Factors (1/2)

No.	Interrupt Factors		Activation trigger (Clearing standby)	CG interrupt mode control register
0	INT0	Interrupt pin (PJ0/70pin)	Selectable	IMCGA
1	INT1	Interrupt pin (PJ1/49pin)		
2	INT2	Interrupt pin (PJ2/86pin)		
3	INT3	Interrupt pin (PJ3/87pin)		
4	INT4	Interrupt pin (PG3/6pin)		
5	INT5	Interrupt pin (PF7/19pin)		IMCGB
6	INTRX0	Serial transmit (channel.0)		
7	INTTX0	Serial reception (channel.0)		
8	INTRX1	Serial transmit (channel.1)		
9	INTTX1	Serial reception (channel.1)		
10	INTSBI0	Serial bus interface 0		
11	INTSBI1	Serial bus interface 1		
12	INTCECRX	CEC reception	Rising edge	IMCGB
13	INTCECTX	CEC transmission		IMCGD
14	INTAINTRMCRX0	Remote control signal reception (channel.0)		IMCGB
15	INTADHP	Highest priority AD conversion complete interrupt		
16	INTADM0	AD conversion monitoring function interrupt 0		
17	INTADM1	AD conversion monitoring function interrupt 1		
18	INTTB0	16bit TMRB match detection 0		
19	INTTB1	16bit TMRB match detection 1		
20	INTTB2	16bit TMRB match detection 2		
21	INTTB3	16bit TMRB match detection 3		
22	INTTB4	16bit TMRB match detection 4		
23	INTTB5	16bit TMRB match detection 5		
24	INTTB6	16bit TMRB match detection 6		
25	INTRTC	Real time clock timer	Falling edge	IMCGC
26	INTCAP00	16bit TMRB input capture 00		
27	INTCAP01	16bit TMRB input capture 01		
28	INTCAP10	16bit TMRB input capture 10		
29	INTCAP11	16bit TMRB input capture 11		
30	INTCAP50	16bit TMRB input capture 50		
31	INTCAP51	16bit TMRB input capture 51		
32	INTCAP60	16bit TMRB input capture 60		
33	INTCAP61	Input capture 61		
34	INT6	Interrupt pin (PJ6/39pin)	Selectable	IMCGC
35	INT7	Interrupt pin (PJ7/58pin)		
36	INTRX2	Serial transmission (channel.2)		
37	INTTX2	Serial reception (channel.2)		
38	INTSBI2	Serial bus interface 2		
39	INTAINTRMCRX1	Remote control signal reception (channel.1)	Rising edge	IMCGC
40	INTTB7	16bit TMRB match detection 7		
41	INTTB8	16bit TMRB match detection 8		
42	INTTB9	16bit TMRB match detection 9		

Table 6-1 List of Hardware Interrupt Factors (2/2)

No.	Interrupt Factors		Activation trigger (Clearing standby)	Clock Generator
43	INTCAP20	16bit TMRB input capture 20		
44	INTCAP21	16bit TMRB input capture 21		
45	INTCAP30	16bit TMRB input capture 30		
46	INTCAP31	16bit TMRB input capture 31		
47	INTCAP40	16bit TMRB input capture 40		
48	INTCAP41	16bit TMRB input capture 41		
49	INTAD	A/D conversion completion		

6.2.5 Active State

The active state indicates which change in signal of an interrupt factor triggers an interrupt. The CPU considers an interrupt signal as an interrupt factor when it is changed from “L” to “H”. A signal directly sent from the peripheral IP to the CPU is configured to output the “H” as an interrupt request.

The active state of the interrupt to clear the standby mode has the options. As for the signal from the peripheral IP, the trigger is determined to be the rising or falling edges depending on the interrupt factor. If the signal is from the external pin, the “H” level, “L” level, rising or falling edge is selectable.

If the interrupt is used for clearing the standby mode, setting the clock generator register is required. Enable the IMCGx<INTxEN> bit and specify the active state in the IMCGx<EMCG2:0> bits. You must set the activation trigger as shown in Table 6-1.

The interrupt detected by the clock generator is notified to the CPU with a signal in “H” level.

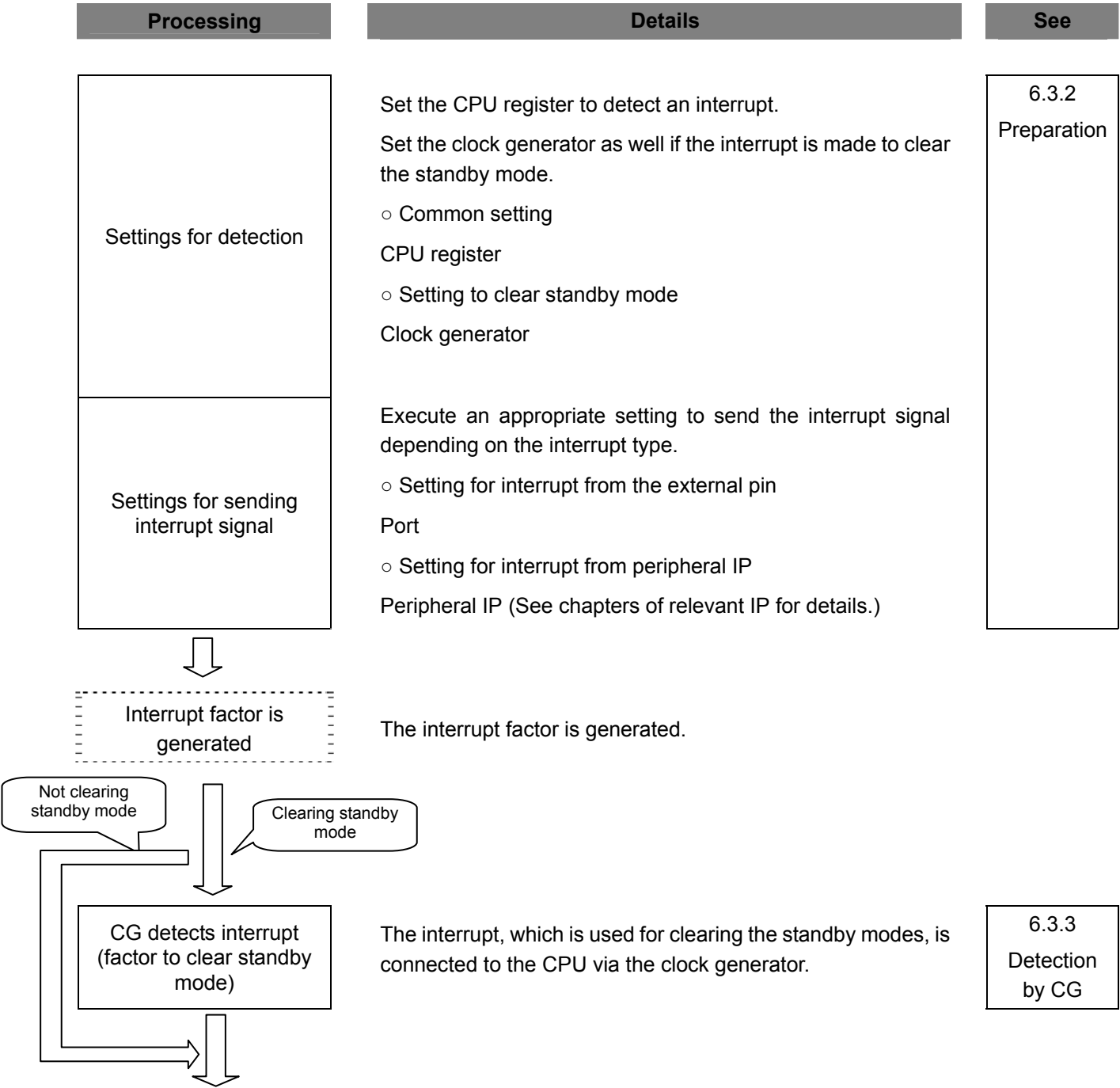
An interrupt from the external pin can be used without setting the clock generator if it does not function as the clearing factor. However, inputting the “H” pulse or the signal in “H” level is required so that the CPU can detect it as an interrupt factor.

6.3 Interrupt handling

6.3.1 Flowchart

The following shows how an interrupt is handled.

indicates hardware handling. indicates software handling.



Processing	Details	See
 Detecting interrupt	The CPU detects the interrupt. The interrupt factor with the highest priority is detected according to the priority order.	6.3.4 Detection by CPU
 Handling interrupt	The CPU handles the interrupt. The CPU pushes register contents to the stack before entering the interrupt service routine	6.3.5 CPU processing
 Executing interrupt service routine	Program for the interrupt service routine. Clear the interrupt factor if needed.	6.3.6 Interrupt service routine
 Returning to preceding program	Configure to return to the preceding program of the interrupt service routine.	

6.3.2 Preparation

When preparing for an interrupt, you need to pay attention to the order of configuration to avoid any unexpected interrupt on the way.

Initiating an interrupt or changing its configuration must be implemented in the following order basically. Disable the interrupt by the CPU. Configure from the farthest route from the CPU. Then enable the interrupt by the CPU.

To configure the clock generator, you must follow the order indicated here not to cause any unexpected interrupt. First, configure the precondition. Secondly, clear the data related to the interrupt in the clock generator and then enable the interrupt.

The following sections are listed in the order of interrupt handling and describe how to configure them.

- (1) Disabling interrupt by CPU
- (2) CPU registers setting
- (3) Preconfiguration 1 (Interrupt from external pin)
- (4) Preconfiguration 2 (interrupt from peripheral IP)
- (5) Configuring the clock generator
- (6) Enabling interrupt by CPU

(1) Disabling interrupt by CPU

To make the CPU for not accepting any interrupt, write "1" to the corresponding bit of the Interrupt Clear-Enable register. Each bit of the register, of which default setting is "0" with interrupt disabled, is assigned to single interrupt factor.

● CPU register		
Interrupt Clear-Enable<m>	←	"1" (interrupt disabled)

(Note) m: corresponding bit

(2) CPU registers setting

You can assign a priority level from 0 to 255 to an interrupt by writing to the eight bit field in an Interrupt Priority Register.

Priority level 0 is the highest priority level.

● CPU register		
Interrupt Priority<m>	←	"priority"

(Note) m: corresponding bit

(3) Preconfiguration 1 (Interrupt from external pin)

Set "1" to the port function register of the corresponding pin. Setting PnFRx[m] allows the pin to be used as the function pin. Setting PnIE[m] allows the pin to be used as the input port.

• Port register		
PnFRx<PnmFRx>	←	"1"
PnIE<PnmIE>	←	"1"

(Note) n: port number
m: corresponding bit
x: function register number

(4) Preconfiguration 2 (interrupt from peripheral IP)

The setting varies depending on the IP to be used. See chapters of relevant IP for details.

(5) Configuring the clock generator

You need to configure active state and enabling interrupt for an interrupt to clear the standby mode with the IMCG register of the clock generator. The IMCG register is capable of configuring each factor.

Before enabling an interrupt, clear the corresponding interrupt request already held. This can avoid unexpected interrupt. To clear corresponding interrupt request, write value corresponding to the interrupt to be used to the ICRCG register. See 6.4.6 INTCG Clear Register for each value.

An interrupt from the external pin can be used without setting the clock generator if it does not function as the clearing factor. However, inputting the "H" pulse or the signal in "H" level is required so that the CPU can detect it as an interrupt factor.

• Clock generator register		
IMCGn<EMCGm>	←	Active state
ICRCG<ICRCG>	←	Value corresponding to the interrupt to be used
IMCGn<INTmEN>	←	"1" (interrupt enabled)

(Note) n: register number
m: number assigned to interrupt factor

(6) Enabling interrupt by CPU

Enable the interrupt by the CPU as shown below.

Clear the suspended interrupt in the Interrupt Clear-Pending register. Enable the intended interrupt with the Interrupt Set-Enable register. Each bit of the register is assigned to single interrupt factor.

Writing “1” to the corresponding bit of the Clear-Pending register clears the suspended interrupt. Writing “1” to the corresponding bit of the Set-Enable register enables the intended interrupt.

●CPU register		
Interrupt Clear-Pending<m>	←	“1”
Interrupt Set-Enable<m>	←	“1”

(Note) m: corresponding bit

6.3.3 Detection by Clock Generator

If the interrupt is used for clearing the standby mode, the interrupt factor is detected by a trigger for an active state specified in the clock generator, and notified to the CPU.

The interrupt factor that enters active state triggered by a rising or falling edge is held in the clock generator after detection. However, if a signal in “H” or “L” level is specified as the trigger to enter the active state, the CPU considers that the interrupt factor is cleared upon exiting from the active state. Therefore, the active state needs to be kept until the interrupt is detected.

The interrupt detected by the clock generator is notified to the CPU with a signal in “H” level. The CPU considers the interrupt signal as an interrupt factor when it is changed from “L” to “H”. To generate an interrupt again, the factor held in the clock generator needs to be cleared with the INTCG clear register.

6.3.4 Detection by CPU

The CPU detects an interrupt factor with the highest priority.

6.3.5 CPU processing

On detecting the interrupt, the CPU pushes the contents of PC, PSR, r0-r3, r12 and LR to the stack before entering the interrupt service routine.

6.3.6 Interrupt Service Routine

Interrupt service routine requires specific programming according to the application to be used. This section describes what is recommended at the service routine programming and how the factor is cleared.

(1) Pushing during interrupt service routine

Common interrupt service routine is accompanied with the interrupt handling and the pushing of the register contents. The Cortex-M3 core automatically pushes the contents of PC, PSR, r0-r3, r12 and LR to the stack. No extra programming is required for them.

Push the contents of other registers if needed.

An interrupt with the higher priority and faults such as NMI are accepted even when the interrupt service routine is being executed. We recommend you to push the contents of the general purpose register that might be rewritten.

(2) Clearing interrupt factor

If an interrupt is used for clearing the standby mode, clear the interrupt request with the INTCG clear register of the clock generator.

If a signal in “H” or “L” level is specified as the trigger to enter the active state, the factor is held unless it is cleared. In this case, the factor needs to be cleared. Clearing the factor causes clearing the interrupt request signal from clock generator.

If a rising or falling edge is specified as the trigger to enter the active state, the factor is cleared by setting the value, which corresponds to the interrupt, to the INTCG clear register. The factor is detected again when the specified edge appears.

6.4 Interrupt-related registers

The clock generator registers and their addresses are as shown below.

6.4.1 Registers

• Clock generator registers		
ICRCG	CG Interrupt Request Clear Register	0x4004_0214
NMIFLG	NMI Flag Register	0x4004_0218
RSTFLG	Reset Flag Register	0x4004_021C
IMCGA	CG Interrupt Mode Control Register A	0x4004_0220
IMCGB	CG Interrupt Mode Control Register B	0x4004_0224
IMCGC	CG Interrupt Mode Control Register C	0x4004_0228
IMCGD	CG Interrupt Mode Control Register D	0x4004_022C

6.4.2 CG Interrupt Mode Control Register A

IMCGA

	7	6	5	4	3	2	1	0
bit Symbol		EMCG02	EMCG01	EMCG00	EMST01	EMST00		INT0EN
Read/Write	R	R/W			R		R	R/W
After reset	0	0	1	0	0	0	0	0
Function	"0" is read.	Active state setting of INT0 standby clear request. (101~111: setting prohibited) 000: "L" level 001: "H" level 010: Falling edge 011: Rising edge 100: Both edges			Active state of INT0 standby clear request 00: — 01: Rising edge 10: Falling edge 11: Both edges		"0" is read.	INT0 clear input 0: Disable 1: Enable
	15	14	13	12	11	10	9	8
bit Symbol		EMCG12	EMCG11	EMCG10	EMST11	EMST10		INT1EN
Read/Write	R	R/W			R		R	R/W
After reset	0	0	1	0	0	0	0	0
Function	"0" is read.	Active state setting of INT1 standby clear request. (101~111: setting prohibited) 000: "L" level 001: "H" level 010: Falling edge 011: Rising edge 100: Both edges			Active state of INT1 standby clear request 00: — 01: Rising edge 10: Falling edge 11: Both edges		"0" is read.	INT1 clear input 0: Disable 1: Enable
	23	22	21	20	19	18	17	16
bit Symbol		EMCG22	EMCG21	EMCG20	EMST21	EMST20		INT2EN
Read/Write	R	R/W			R		R	R/W
After reset	0	0	1	0	0	0	0	0
Function	"0" is read.	Active state setting of INT2 standby clear request. (101~111: setting prohibited) 000: "L" level 001: "H" level 010: Falling edge 011: Rising edge 100: Both edges			Active state of INT2 standby clear request 00: — 01: Rising edge 10: Falling edge 11: Both edges		"0" is read.	INT2 clear input 0: Disable 1: Enable
	31	30	29	28	27	26	25	24
bit Symbol		EMCG32	EMCG31	EMCG30	EMST31	EMST30		INT3EN
Read/Write	R	R/W			R		R	R/W
After reset	0	0	1	0	0	0	0	0
Function	"0" is read.	Active state setting of INT3 standby clear request. (101~111: setting prohibited) 000: "L" level 001: "H" level 010: Falling edge 011: Rising edge 100: Both edges			Active state of INT3 standby clear request 00: — 01: Rising edge 10: Falling edge 11: Both edges		"0" is read.	INT3 clear input 0: Disable 1: Enable

- (Note 1) Refer to EMSTxx bit to know the active condition which is used for clearing standby.
- (Note 2) Please specify the bit for the edge first and then specify the bit for the <INTxEN>. Setting them simultaneously is prohibited.

6.4.3 CG Interrupt Mode Control Register B

IMCGB

	7	6	5	4	3	2	1	0
bit Symbol		EMCG42	EMCG41	EMCG40	EMST41	EMST40		INT4EN
Read/Write	R	R/W			R		R	R/W
After reset	0	0	1	0	0	0	0	0
Function	"0" is read.	Active state setting of INT4 standby clear request (101~111: setting prohibited) 000: "L" level 001: "H" level 010: Falling edge 011: Rising edge 100: Both edges			Active state of INT4 standby clear request 00: — 01: Rising edge 10: Falling edge 11: Both edges		"0" is read.	INT4 clear input 0: Disable 1: Enable
	15	14	13	12	11	10	9	8
bit Symbol		EMCG52	EMCG51	EMCG50	EMST51	EMST50		INT5EN
Read/Write	R	R/W			R		R	R/W
After reset	0	0	1	0	0	0	0	0
Function	"0" is read.	Active state setting of INT5 standby clear request. (101~111: setting prohibited) 000: "L" level 001: "H" level 010: Falling edge 011: Rising edge 100: Both edges			Active state of INT5 standby clear request 00: — 01: Rising edge 10: Falling edge 11: Both edges		"0" is read.	INT5 clear input 0: Disable 1: Enable
	23	22	21	20	19	18	17	16
bit Symbol		EMCG62	EMCG61	EMCG60	EMST61	EMST60		INT6EN
Read/Write	R	R/W			R		R	R/W
After reset	0	0	1	0	0	0	0	0
Function	"0" is read.	Active state setting of INTCECRX standby clear request. Set it as shown below. 011: Rising edge			Active state of INTCECRX standby clear request 00: — 01: Rising edge 10: Falling edge 11: Both edges		"0" is read.	INTCECRX Clear input 0: Disable 1: Enable
	31	30	29	28	27	26	25	24
bit Symbol		EMCG72	EMCG71	EMCG70	EMST71	EMST70		INT7EN
Read/Write	R	R/W			R		R	R/W
After reset	0	0	1	0	0	0	0	0
Function	"0" is read.	Active state setting of INTRMCRX0 standby clear request. Set it as shown below. 011: Rising edge			Active state of INTRMCRX0 standby clear request 00: — 01: Rising edge 10: Falling edge 11: Both edges		"0" is read.	INTRMCRX0 Clear input 0: Disable 1: Enable

(Note 1) Refer to EMSTxx bit to know the active condition which is used for clearing standby.

(Note 2) Please specify the bit for the edge first and then specify the bit for the <INTxEN>. Setting them simultaneously is prohibited.

6.4.4 CG Interrupt Mode Control Register C

IMCGC

	7	6	5	4	3	2	1	0
bit Symbol		EMCG82	EMCG81	EMCG80	EMST81	EMST80		INT8EN
Read/Write	R	R/W			R		R	R/W
After reset	0	0	1	0	0	0	0	0
Function	"0" is read.	Active state setting of INTRTC standby clear request. Set it as shown below. 010: Falling edge			Active state of INTRTC standby clear request. 00: - 01: Rising edge 10: Falling edge 11: Both edges		"0" is read.	INTRTC clear input 0: Disable 1: Enable
	15	14	13	12	11	10	9	8
bit Symbol		EMCG92	EMCG91	EMCG90	EMST91	EMST90		INT9EN
Read/Write	R	R/W			R		R	R/W
After reset	0	0	1	0	0	0	0	0
Function	"0" is read.	Active state setting of INT6 standby clear request. (101~111: setting prohibited) 000: "L" level 001: "H" level 010: Falling edge 011: Rising edge 100: Both edges			Active state of INT6 standby clear request. 00: - 01: Rising edge 10: Falling edge 11: Both edges		"0" is read.	INT6 clear input 0: Disable 1: Enable
	23	22	21	20	19	18	17	16
bit Symbol		EMCGA2	EMCGA1	EMCGA0	EMSTA1	EMSTA0		INTAEN
Read/Write	R	R/W			R		R	R/W
After reset	0	0	1	0	0	0	0	0
Function	"0" is read.	Active state setting of INT7 standby clear request. (101~111: setting prohibited) 000: "L" level 001: "H" level 010: Falling edge 011: Rising edge 100: Both edges			Active state of INT7 standby clear request. 00: - 01: Rising edge 10: Falling edge 11: Both edges		"0" is read.	INTA7 clear input 0: Disable 1: Enable
	31	30	29	28	27	26	25	24
bit Symbol		EMCGB2	EMCGB1	EMCGB0	EMSTB1	EMSTB0		INTBEN
Read/Write	R	R/W			R		R	R/W
After reset	0	0	1	0	0	0	0	0
Function	"0" is read.	Active state setting of INTRMCRX1 standby clear request. Set it as shown below. 011: Rising edge			Active state of INTRMCRX1 standby clear request. 00: - 01: Rising edge 10: Falling edge 11: Both edges		"0" is read.	INTRMC RX1 clear input 0: Disable 1: Enable

(Note 1) Refer to EMSTxx bit to know the active condition which is used for clearing standby.

(Note 2) Please specify the bit for the edge first and then specify the bit for the <INTxEN>. Setting them simultaneously is prohibited.

6.4.5 CG Interrupt Mode Control Register D

IMCGD

	7	6	5	4	3	2	1	0
bit Symbol		EMCG82	EMCG81	EMCG80	EMST81	EMST80		INT8EN
Read/Write	R	R/W			R		R	R/W
After reset	0	0	1	0	0	0	0	0
Function	"0" is read.	Active state setting of INTCECTX standby clear request. Set it as shown below. 010: Falling edge			Active state of INTCECTX standby clear request. 00: - 01: Rising edge 10: Falling edge 11: Both edges		"0" is read.	INTCECTX Clear input 0: Disable 1: Enable
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R	R/W			R			R/W
After reset	0	0	1	0	0	0	0	0
Function	"0" is read.	Write "0".			"0" is read.			Write "0".
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R	R/W			R			R/W
After reset	0	0	1	0	0	0	0	0
Function	"0" is read.	Write "0".			"0" is read.			Write "0".
	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R	R/W			R			R/W
After reset	0	0	1	0	0	0	0	0
Function	"0" is read.	Write "0".			"0" is read.			Write "0".

(Note 1) Refer to EMSTxx bit to know the active condition which is used for clearing standby.

(Note 2) Please specify the bit for the edge first and then specify the bit for the <INTxEN>. Setting them simultaneously is prohibited.

Be sure to set active state of the clear request if interrupt is enabled for clearing the STOP/ IDLE / SLEEP standby modes.

(Note 1) When using interrupts, be sure to follow the sequence of actions shown below:

- 1) If shared with other general ports, enable the port to receive the interrupt.
- 2) Set conditions such as active state upon initialization.
- 3) Clear interrupt requests.
- 4) Enable interrupts.

(Note 2) Settings must be performed while interrupts are disabled.

(Note 3) For clearing the standby modes with TMPM330, 13 interrupt factors (CEC reception/ transmission, remote control signal reception of ch0 and ch1, INT0 to INT7 and RTC timer interrupt) are available. You can use CG for selecting edge/level of active state and judging whether the aforementioned factors are to be used for clearing the standby modes. As for the above interrupt factors excluding INT0 to INT7, set INTxEN bit to “1” and specify the other related bits to the appropriate active state even when they are not used for clearing the standby modes.

(Note 4) Among the above 13 factors to be assigned as the standby mode clear request interrupts, INT0 to INT7 can be used as a normal interrupt without setting CG.

6.4.6 INTCG Clear Register

ICRCG

	7	6	5	4	3	2	1	0
bit Symbol				ICRCG4	ICRCG3	ICRCG2	ICRCG1	ICRCG0
Read/Write	R			W				
After reset	0			0	0	0	0	0
Function	"0" is read.			Clear interrupt requests. 0_0000: INT0 0_0100: INT4 0_1000: INTRTC 0_0001: INT1 0_0101: INT5 0_1001: INT6 0_0010: INT2 0_0110: 0_1010: INT7 0_0011: INT3 INTCECRX 0_1011: 0_0111: INTRMCRX1 IINTRMCRX0 * 0_1100~1_1111: setting prohibited * "0" is read.				
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R							
After reset	0							
Function	"0" is read.							
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write								
After reset								
Function	"0" is read.							
	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R							
After reset	0							
Function	"0" is read.							

6.4.7 NMI Flag Register

NMIFLG

	7	6	5	4	3	2	1	0
bit Symbol							NMIFLG1	NMIFLG0
Read/Write	R							
After reset	0	0	0	0	0	0	0	0
Function	"0" is read.						NMI factor generation flag 0: not applicable 1: generated from NMI pin	NMI factor generation flag 0: not applicable 1: generated from WDT
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R							
After reset	0	0	0	0	0	0	0	0
Function	"0" is read.							
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R							
After reset	0	0	0	0	0	0	0	0
Function	"0" is read.							
	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R							
After reset	0	0	0	0	0	0	0	0
Function	"0" is read.							

(Note) <NMIFLG1:0> are cleared to "0" when they are read.

6.4.8 Reset Flag Register

RSTFLG

	7	6	5	4	3	2	1	0
bit Symbol				DBG_RSTF		WDT_RSTF	PIN_RSTF	PON_RSTF
Read/Write	R			R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	1	1
Function	"0" is read.			Debug reset flag 0: "0" is written 1: Reset from debugger	Write "0".	WDT reset flag 0: "0" is written 1: Reset from WDT	RESET pin flag 0: "0" is written 1: Reset from RESET pin	Power-on reset flag 0: "0" is written 1: Reset from power-on reset
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R							
After reset	0	0	0	0	0	0	0	0
Function	"0" is read.							
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R							
After reset	0	0	0	0	0	0	0	0
Function	"0" is read.							
	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R							
After reset	0	0	0	0	0	0	0	0
Function	"0" is read.							

(Note) The TMPM330 has power-on reset circuit and this register is initialized only by power-on reset. Therefore, "1" is set to the <PON_RSTF> bit in initial reset state right after power-on. Note that this bit is not set by the second and subsequent resets and this register is not cleared automatically. Write "0" to clear the register.

7 Input/Output Ports

7.1 Port registers

Px	: Port register To read/ write port data.
PxCR	: Control register To control input/output * Need to enable the input with PxIE register even when input is set.
PxFRn	: Function register To set functions. An assigned function can be activated by setting "1".
PxOD	: Open drain control register To switch the input of a register that can be set as programmable open drain.
PxPUP	: Pull up control register To control program pull ups.
PxIE	: Input control enable register To control inputs. "0" is set as default to avoid through current. This setting prohibits inputs.

7.1.1 Port A (PA0~PA7)

The port A is a general-purpose, 8-bit input/output port. For this port, inputs and outputs can be specified in units of bits. Besides the general-purpose input/output function, the port A performs the debug communication function and the debug trace output function.

A reset allows PA0 and PA1 to perform debug communication function and the other bits perform as the general-purpose port with input/output disabled. When PA0 functions as the TMS or SWDIO port, input, output and pull-up are enabled. When PA1 functions as the TCK or SWCLK port, input and pull-down are enabled.

Port A register

PA (0x4000_0000)		7	6	5	4	3	2	1	0
	Bit Symbol	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
	Read/Write	R/W							
	After reset	"0"							

Port A control register

PACR (0x4000_0004)		7	6	5	4	3	2	1	0
	Bit Symbol	PA7C	PA6C	PA5C	PA4C	PA3C	PA2C	PA1C	PA0C
	Read/Write	R/W							
	After reset	0	0	0	0	0	0	0	1
	Function	0: Output disabled 1: Output enabled							

Port A function register 1

PAFR1 (0x4000_0008)		7	6	5	4	3	2	1	0
	Bit Symbol	—	PA6F1	PA5F1	PA4F1	PA3F1	PA2F1	PA1F1	PA0F1
	Read/Write	R	R/W						
	After reset	0	0	0	0	0	0	1	1
	Function	"0" is read.	0:PORT 1: TRACE DATA3	0:PORT 1: TRACE DATA2	0:PORT 1: TRACE DATA1	0:PORT 1:TRACE DATA0	0:PORT 1: TRACE CLK	0:PORT 1: TCK/ SWCLK	0:PORT 1: TMS/ SWDIO

Port A pull-up control register

PAPUP (0x4000_002C)		7	6	5	4	3	2	1	0
	Bit Symbol	PA7UP	PA6UP	PA5UP	PA4UP	PA3UP	PA2UP	—	PA0UP
	Read/Write	R/W						R	R/W
	After reset	0	0	0	0	0	0	0	1
	Function	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	"0" is read.	Pull-up 0:off 1:on

Port A pull-down control register

PAPDN (0x4000_0030)		7	6	5	4	3	2	1	0
	Bit Symbol	—	—	—	—	—	—	PA1DN	—
	Read/Write	R						R/W	R
	After reset	0	0	0	0	0	0	1	0
	Function	"0" is read.						Pull-up 0:off 1:on	"0" is read.

Port A input enable control register

PAIE
(0x4000_0038)

	7	6	5	4	3	2	1	0
Bit Symbol	PA7IE	PA6IE	PA5IE	PA4IE	PA3IE	PA2IE	PA1IE	PA0IE
Read/Write	R/W							
After reset	0	0	0	0	0	0	1	1
Function	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled

7.1.2 Port B (PB0~PB7)

The port B is a general-purpose, 8-bit input/output port. For this port, inputs and outputs can be specified in units of bits. Besides the general-purpose input/output function, the port B performs debug communication function.

A reset allows PB0, PB1 and PB2 to perform the debug communication function and the other bits perform as the general-purpose port with input/output disabled. When PB0 functions as the TDO or SWV port, output is enabled. When PB1 functions as the TDI port, input and pull-up are enabled. When PB2 functions as the TRST port, input and pull-up are enabled.

Port B register

PB (0x4000_0040)		7	6	5	4	3	2	1	0
	Bit Symbol	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
	Read/Write	R/W							
	After reset	"0"							

Port B control register

PBCR (0x4000_0044)		7	6	5	4	3	2	1	0
	Bit Symbol	PB7C	PB6C	PB5C	PB4C	PB3C	PB2C	PB1C	PB0C
	Read/Write	R/W							
	After reset	0	0	0	0	0	0	0	1
	Function	0: Output disabled 1: Output enabled							

Port B function register 1

PBF1 (0x4000_0048)		7	6	5	4	3	2	1	0
	Bit Symbol	—	—	—	—	—	PB2F1	PB1F1	PB0F1
	Read/Write	R					R/W		
	After reset	0					1	1	1
	Function	"0" is read.					0:PORT 1: TRST	0:PORT 1: TDI	0:PORT 1: TDO/SWV

Port B pull-up control register

PBPUP (0x4000_006C)		7	6	5	4	3	2	1	0
	Bit Symbol	PB7UP	PB6UP	PB5UP	PB4UP	PB3UP	PB2UP	PB1UP	PB0UP
	Read/Write	R/W							
	After reset	0	0	0	0	0	1	1	0
	Function	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on

Port B input enable control register

PBIE (0x4000_0078)		7	6	5	4	3	2	1	0
	Bit Symbol	PB7IE	PB6IE	PB5IE	PB4IE	PB3IE	PB2IE	PB1IE	PB0IE
	Read/Write	R/W							
	After reset	0	0	0	0	0	1	1	0
	Function	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled

7.1.3 Port C (PC0~PC3)

The port C is a 4-bit input port. Besides the general-purpose input function, the port C functions as the analog input port of the A/D converter.

A reset allows all bits to perform as the general-purpose input port and the port C to be put in input disable mode.

Set the input enable control register when you use the port C as an input port. The setting is not required when you use it as an analog input port of the A/D converter.

Port C register

PC (0x4000_0080)		7	6	5	4	3	2	1	0
	Bit Symbol	—	—	—	—	PC3	PC2	PC1	PC0
	Read/Write	R				R/W			
	After reset	"0" is read.				"0"			

Port C pull-up control register

PCPUP (0x4000_00AC)		7	6	5	4	3	2	1	0
	Bit Symbol	—	—	—	—	PC3UP	PC2UP	PC1UP	PC0UP
	Read/Write	R				R/W			
	After reset	0				0	0	0	0
	Function	"0" is read.				Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on

Port C input enable control register

PCIE (0x4000_00B8)		7	6	5	4	3	2	1	0
	Bit Symbol	—	—	—	—	PC3IE	PC2IE	PC1IE	PC0IE
	Read/Write	R				R/W			
	After reset	0				0	1	1	0
	Function	"0" is read.				Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled

7.1.4 Port D (PD0~PD7)

The port D is an 8-bit input port. Besides the general-purpose input function, the port D receives an analog input of the A/D converter and a 16-bit timer input.

A reset allows all bits to perform as the general-purpose input port and the port D to be put in input disable mode.

Set the input enable control register when you use the port D as an input port. Set the function register 1 and input enable control register when you use the port D as an input port of the 16-bit timer. The setting is not required when you use it as an analog input port of the A/D converter.

Port D register

PD (0x4000_00C0)		7	6	5	4	3	2	1	0
	Bit Symbol	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	Read/Write	R/W							
	After reset	"0"							

Port D function register 1

PDFR1 (0x4000_00C8)		7	6	5	4	3	2	1	0
	Bit Symbol	—	—	—	—	PD3F1	PD2F1	PD1F1	PD0F1
	Read/Write	R				R/W			
	After reset	0				0	0	0	0
	Function	"0" is read.				0:PORT 1: TB6IN1	0:PORT 1: TB6IN0	0:PORT 1: TB5IN1	0:PORT 1: TB5IN0

Port D pull-up control register

PDPUP (0x4000_00EC)		7	6	5	4	3	2	1	0
	Bit Symbol	PD7UP	PD6UP	PD5UP	PD4UP	PD3UP	PD2UP	PD1UP	PD0UP
	Read/Write	R/W							
	After reset	0	0	0	0	0	0	0	0
	Function	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on

Port D input enable control register

PDIE (0x4000_00F8)		7	6	5	4	3	2	1	0
	Bit Symbol	PD7IE	PD6IE	PD5IE	PD4IE	PD3IE	PD2IE	PD1IE	PD0IE
	Read/Write	R/W							
	After reset	0	0	0	0	0	0	0	0
	Function	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled

7.1.5 Port E (PE0~PE6)

The port E is a general-purpose, 7-bit input/output port. For this port, inputs and outputs can be specified in units of bits. Besides the general-purpose port function, the port E performs the serial interface function and the remote control signal preprocessor input function.

A reset allows all bits to perform as the general-purpose port and the port E to be put in input/output disable mode.

Port E register

		7	6	5	4	3	2	1	0
PE (0x4000_0100)	Bit Symbol	—	PE6	PE5	PE4	PE3	PE2	PE1	PE0
	Read/Write	R	R/W						
	After reset	“0”	“0”						

Port E control register

PECR (0x4000_0104)		7	6	5	4	3	2	1	0
	Bit Symbol	—	PE6C	PE5C	PE4C	PE3C	PE2C	PE1C	PE0C
	Read/Write	R	R/W						
	After reset	0	0	0	0	0	0	0	0
	Function	"0" is read.	0: Output disabled 1: Output enabled						

Port E function register 1

PEFR1 (0x4000_0108)		7	6	5	4	3	2	1	0
	Bit Symbol	—	PE6F1	PE5F1	PE4F1	PE3F1	PE2F1	PE1F1	PE0F1
	Read/Write	R	R/W						
	After reset	0	0	0	0	0	0	0	0
	Function	"0" is read.	0:PORT 1:SCLK1	0:PORT 1:RXD1	0:PORT 1:TXD1	0:PORT 1:RXIN0	0:PORT 1:SCLK0	0:PORT 1:RXD0	0:PORT 1:TXD0

Port E function register 2

	7	6	5	4	3	2	1	0	
PEFR2 (0x4000_010C)	Bit Symbol	—	PE6F2	—	—	—	PE2F2	—	—
	Read/Write	R	R/W	R		R/W	R		
	After reset	0	0	0		0	0		
	Function	“0” is read.	0:PORT 1:CTS1	“0” is read.		0:PORT 1:CTS0	“0” is read.		

Port E open drain control register

PEOD (0x4000_0128)		7	6	5	4	3	2	1	0
	Bit Symbol	—	PE6OD	PE5OD	PE4OD	PE3OD	PE2OD	PE1OD	PE0OD
	Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	After reset	0	0	0	0	0	0	0	0
	Function	"0" is read.	0:CMOS 1:Open drain	0:CMOS 1:Open drain	0:CMOS 1:Open drain	0:CMOS 1:Open drain	0:CMOS 1:Open drain	0:CMOS 1:Open drain	0:CMOS 1:Open drain

Port E pull-up control register

PEPUP (0x4000_012C)		7	6	5	4	3	2	1	0
	Bit Symbol	—	PE6UP	PE5UP	PE4UP	PE3UP	PE2UP	PE1UP	PE0UP
	Read/Write	R	R/W						
	After reset	0	0	0	0	0	0	0	0
	Function	"0" is read.	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on

Port E input enable control register

PEIE (0x4000_0138)		7	6	5	4	3	2	1	0
	Bit Symbol	—	PE6IE	PE5IE	PE4IE	PE3IE	PE2IE	PE1IE	PE0IE
	Read/Write	R	R/W						
	After reset	0	0	0	0	0	0	0	0
	Function	"0" is read.	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled

7.1.6 Port F (PF0~PF7)

The port F is a general-purpose, 8-bit input/output port. For this port, inputs and outputs can be specified in units of bits. Besides the general-purpose port function, the port F performs the functions of the serial interface, the remote control signal preprocessor, the serial bus interface and the external interrupt input.

A reset allows all bits to perform as the general-purpose port and the port F to be put in input/ output disable mode.

Port F register

		7	6	5	4	3	2	1	0
PF (0x4000_0140)	Bit Symbol	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
	Read/Write	R/W							
	After reset	"0"							

Port F control register

PFCR (0x4000_0144)		7	6	5	4	3	2	1	0
	Bit Symbol	PF7C	PF6C	PF5C	PF4C	PF3C	PF2C	PF1C	PF0C
	Read/Write	R/W							
	After reset	0	0	0	0	0	0	0	0
	Function	0: Output disabled 1: Output enabled							

Port F function register 1

PFFR1 (0x4000_0148)		7	6	5	4	3	2	1	0
	Bit Symbol	PF7F1	PF6F1	PF5F1	PF4F1	PF3F1	PF2F1	PF1F1	PF0F1
	Read/Write	R/W							
	After reset	0	0	0	0	0	0	0	0
	Function	0:PORT 1: INT5	0:PORT 1: SCK1	0:PORT 1: SI1/ SCL1	0:PORT 1: SO1/ SDA1	0:PORT 1: RXIN1	0:PORT 1: SCLK2	0:PORT 1: RXD2	0:PORT 1: TXD2

Port F function register 2

	7	6	5	4	3	2	1	0
PFFR2 (0x4000_014C)	Bit Symbol	—	—	—	—	PF2F2	—	—
	Read/Write	R				R/W	R	
	After reset	0				0	0	
	Function	"0" is read.				0:PORT 1:CTS2	"0" is read.	

Port F open drain control register

PFOD (0x4000_0168)		7	6	5	4	3	2	1	0
	Bit Symbol	PF7OD	PF6OD	PF5OD	PF4OD	PF3OD	PF2OD	PF1OD	PF0OD
	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	After reset	0	0	0	0	0	0	0	0
	Function	0:CMOS 1:Open drain	0:CMOS 1:Open drain	0:CMOS 1:Open drain	0:CMOS 1:Open drain	0:CMOS 1:Open drain	0:CMOS 1:Open drain	0:CMOS 1:Open drain	0:CMOS 1:Open drain

Port F pull-up control register

PFPUP (0x4000_016C)		7	6	5	4	3	2	1	0
	Bit Symbol	PF7UP	PF6UP	PF5UP	PF4UP	PF3UP	PF2UP	PF1UP	PF0UP
	Read/Write	R/W							
	After reset	0	0	0	0	0	0	0	0
	Function	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on

Port F input enable control register

PFIE (0x4000_0178)		7	6	5	4	3	2	1	0
	Bit Symbol	PF7IE	PF6IE	PF5IE	PF4IE	PF3IE	PF2IE	PF1IE	PF0IE
	Read/Write	R/W							
	After reset	0	0	0	0	0	0	0	0
	Function	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled

7.1.7 Port G (PG0~PG7)

The port G is a general-purpose, 8-bit input/output port. For this port, inputs and outputs can be specified in units of bits. Besides the general-purpose port function, the port G performs the functions of the serial bus interface, the external interrupt input, and the 16-bit timer output.

A reset allows all bits to perform as the general-purpose port and the port G to be put in input/output disable mode.

Port G register

PG (0x4000_0180)		7	6	5	4	3	2	1	0
	Bit Symbol	PG7	PG6	PG5	PG4	PG3	PG2	PG1	PG0
	Read/Write	R/W							
	After reset	"0"							

Port G control register

PGCR (0x4000_0184)		7	6	5	4	3	2	1	0
	Bit Symbol	PG7C	PG6C	PG5C	PG4C	PG3C	PG2C	PG1C	PG0C
	Read/Write	R/W							
	After reset	0	0	0	0	0	0	0	0
	Function	0: Output disabled 1: Output enabled							

Port G function register 1

PGFR1 (0x4000_0188)		7	6	5	4	3	2	1	0
	Bit Symbol	PG7F1	PG6F1	PG5F1	PG4F1	PG3F1	PG2F1	PG1F1	PG0F1
	Read/Write	R/W							
	After reset	0	0	0	0	0	0	0	0
	Function	0:PORT 1: TB8OUT	0:PORT 1: SCK2	0:PORT 1: SI2/ SCL2	0:PORT 1: SO2/ SDA2	0:PORT 1: INT4	0:PORT 1: SCK0	0:PORT 1: SI0/ SCL0	0:PORT 1: SO0/ SDA0

Port G open drain control register

PGOD (0x4000_01A8)		7	6	5	4	3	2	1	0
	Bit Symbol	PG7OD	PG6OD	PG5OD	PG4OD	PG3OD	PG2OD	PG1OD	PG0OD
	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	After reset	0	0	0	0	0	0	0	0
	Function	0:CMOS 1:Open drain	0:CMOS 1:Open drain	0:CMOS 1:Open drain	0:CMOS 1:Open drain	0:CMOS 1:Open drain	0:CMOS 1:Open drain	0:CMOS 1:Open drain	0:CMOS 1:Open drain

Port G pull-up control register

PGPUP (0x4000_01AC)		7	6	5	4	3	2	1	0
	Bit Symbol	PG7UP	PG6UP	PG5UP	PG4UP	PG3UP	PG2UP	PG1UP	PG0UP
	Read/Write	R/W							
	After reset	0	0	0	0	0	0	0	0
	Function	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on

Port G input enable control register

PGIE (0x4000_01B8)		7	6	5	4	3	2	1	0
	Bit Symbol	PG7IE	PG6IE	PG5IE	PG4IE	PG3IE	PG2IE	PG1IE	PG0IE
	Read/Write	R/W							
	After reset	0	0	0	0	0	0	0	0
	Function	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled

7.1.8 Port H (PH0~PH7)

The port H is a general-purpose, 8-bit input/output port. For this port, inputs and outputs can be specified in units of bits. Besides the general-purpose port function, the port H performs the functions of the 16-bit timer input and the operation mode setting.

While a reset signal is in “0” state, the PH0 input is enabled. At the rising edge of the reset signal, if PI0 is “1”, it enters single mode and executes boot from the on-board flash memory. If PI0 is “0”, it enters single boot mode and executes boot from the stored boot program. For details of single boot mode, refer to “Chapter 17 Flash Memory Operation”.

Releasing a reset allows all bits to perform as the general-purpose port and the port H to be put in input/output disable mode.

Port H register

		7	6	5	4	3	2	1	0
	Bit Symbol	PH7	PH6	PH5	PH4	PH3	PH2	PH1	PH0
PH (0x4000_01C0)	Read/Write	R/W							
	After reset	“0”							

Port H control register

		7	6	5	4	3	2	1	0
	Bit Symbol	PH7C	PH6C	PH5C	PH4C	PH3C	PH2C	PH1C	PH0C
PHCR (0x4000_01C4)	Read/Write	R/W							
	After reset	0	0	0	0	0	0	0	0
	Function	0: Output disabled 1: Output enabled							

Port H function register 1

		7	6	5	4	3	2	1	0
	Bit Symbol	PH7F1	PH6F1	PH5F1	PH4F1	PH3F1	PH2F1	PH1F1	PH0F1
PHFR1 (0x4000_01C8)	Read/Write	R/W							
	After reset	0	0	0	0	0	0	0	0
	Function	0:PORT 1: TB3IN1	0:PORT 1: TB3IN0	0:PORT 1: TB2IN1	0:PORT 1: TB2IN0	0:PORT 1: TB1IN1	0:PORT 1: TB1IN0	0:PORT 1: TB0IN1	0:PORT 1: TB0IN0

Port H pull-up control register

		7	6	5	4	3	2	1	0
	Bit Symbol	PH7UP	PH6UP	PH5UP	PH4UP	PH3UP	PH2UP	PH1UP	PH0UP
PHPUP (0x4000_01EC)	Read/Write	R/W							
	After reset	0	0	0	0	0	0	0	1
	Function	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on

Port H input enable control register

		7	6	5	4	3	2	1	0
	Bit Symbol	PH7IE	PH6IE	PH5IE	PH4IE	PH3IE	PH2IE	PH1IE	PH0IE
PHIE (0x4000_01F8)	Read/Write	R/W							
	After reset	0	0	0	0	0	0	0	0
	Function	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled

7.1.9 Port I (PI0~PI7)

The port I is a general-purpose, 8-bit input/output port. For this port, inputs and outputs can be specified in units of bits. Besides the general-purpose port function, the port I performs the 16-bit timer input/output function.

A reset allows all bits to perform as the general-purpose port and the port I to be put in input/output disable mode.

Port I register

	7	6	5	4	3	2	1	0
Bit Symbol	PI7	PI6	PI5	PI4	PI3	PI2	PI1	PI0
Read/Write	R/W							
After reset	"0"							

PI
(0x4000_0200)

Port I control register

	7	6	5	4	3	2	1	0
Bit Symbol	PI7C	PI6C	PI5C	PI4C	PI3C	PI2C	PI1C	PI0C
Read/Write	R/W							
After reset	0	0	0	0	0	0	0	0
Function	0: Output disabled 1: Output enabled							

PICR
(0x4000_0204)

Port I function register 1

	7	6	5	4	3	2	1	0
Bit Symbol	PI7F1	PI6F1	PI5F1	PI4F1	PI3F1	PI2F1	PI1F1	PI0F1
Read/Write	R/W							
After reset	0	0	0	0	0	0	0	0
Function	0:PORT 1: TB4IN1	0:PORT 1: TB4IN0	0:PORT 1: TB5OUT	0:PORT 1: TB4OUT	0:PORT 1: TB3OUT	0:PORT 1: TB2OUT	0:PORT 1: TB1OUT	0:PORT 1: TB0OUT

PIFR1
(0x4000_0208)

Port I pull-up control register

	7	6	5	4	3	2	1	0
Bit Symbol	PI7UP	PI6UP	PI5UP	PI4UP	PI3UP	PI2UP	PI1UP	PI0UP
Read/Write	R/W							
After reset	0	0	0	0	0	0	0	0
Function	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on

PIPUP
(0x4000_022C)

Port I input enable control register

	7	6	5	4	3	2	1	0
Bit Symbol	PI7IE	PI6IE	PI5IE	PI4IE	PI3IE	PI2IE	PI1IE	PI0IE
Read/Write	R/W							
After reset	0	0	0	0	0	0	0	0
Function	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled

PIIE
(0x4000_0238)

7.1.10 Port J (PJ0~PJ7)

The port J is a general-purpose, 8-bit input/output port. For this port, inputs and outputs can be specified in units of bits. Besides the general-purpose port function, the port J performs the functions of the 16-bit timer output and the external interrupt input.

A reset allows all bits to perform as the general-purpose port and the port J to be put in input/output disable mode.

Port J register

	7	6	5	4	3	2	1	0
PJ	PJ7	PJ6	PJ5	PJ4	PJ3	PJ2	PJ1	PJ0
(0x4000_0240)	R/W							
	"0"							

Port J control register

	7	6	5	4	3	2	1	0
PJCR	PJ7C	PJ6C	PJ5C	PJ4C	PJ3C	PJ2C	PJ1C	PJ0C
(0x4000_0244)	R/W							
	0	0	0	0	0	0	0	0
	0: Output disabled 1: Output enabled							

Port J function register 1

	7	6	5	4	3	2	1	0
PJFR1	PJ7F1	PJ6F1	PJ5F1	PJ4F1	PJ3F1	PJ2F1	PJ1F1	PJ0F1
(0x4000_0248)	R/W							
	0	0	0	0	0	0	0	0
	0:PORT 1: INT7	0:PORT 1: INT6	0:PORT 1: TB7OUT	0:PORT 1: TB6OUT	0:PORT 1: INT3	0:PORT 1: INT2	0:PORT 1: INT1	0:PORT 1: INT0

Port J pull-up control register

	7	6	5	4	3	2	1	0
PJPUP	PJ7UP	PJ6UP	PJ5UP	PJ4UP	PJ3UP	PJ2UP	PJ1UP	PJ0UP
(0x4000_026C)	R/W							
	0	0	0	0	0	0	0	0
	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on	Pull-up 0:off 1:on

Port J input enable control register

	7	6	5	4	3	2	1	0
PJIE	PJ7IE	PJ6IE	PJ5IE	PJ4IE	PJ3IE	PJ2IE	PJ1IE	PJ0IE
(0x4000_0278)	R/W							
	0	0	0	0	0	0	0	0
	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled

7.1.11 Port K (PK0~PK2)

The port K is a general-purpose, 3-bit input/output port. For this port, inputs and outputs can be specified in units of bits. Besides the general-purpose port function, the port K performs the functions of the 16-bit timer output, the CEC input, the clock output and the alarm output.

A reset allows all bits to perform as the general-purpose port and the port K to be put in input/output disable mode.

Port K register

	7	6	5	4	3	2	1	0
PK (0x4000_0280)	—	—	—	—	—	PK2	PK1	PK0
Read/Write	R					R/W		
After reset	"0" is read.					"0"		

Port K control register

	7	6	5	4	3	2	1	0
PKCR (0x4000_0284)	—	—	—	—	—	PK2C	PK1C	PK0C
Read/Write	R					R/W		
After reset	0					0	0	0
Function	"0" is read.					0: Output disabled 1: Output enabled		

Port K function register 1

	7	6	5	4	3	2	1	0
PKFR1 (0x4000_0288)	—	—	—	—	—	PK2F1	PK1F1	PK0F1
Read/Write	R					R/W		
After reset	0					0	0	0
Function	"0" is read.					0:PORT 1: TB9OUT	0:PORT 1: SCOUT	0:PORT 1: CEC

Port K function register 2

	7	6	5	4	3	2	1	0
PKFR2 (0x4000_028C)	—	—	—	—	—	—	PK2F1	—
Read/Write	R					—	R/W	R
After reset	0					—	0	0
Function	"0" is read.					—	0:PORT 1: ALARM	"0" is read.

Port K pull-up control register

	7	6	5	4	3	2	1	0
PKPUP (0x4000_02AC)	—	—	—	—	—	PJ2UP	PJ1UP	—
Read/Write	R					R/W		R
After reset	0					0	0	0
Function	"0" is read.					Pull-up 0:off 1:on	Pull-up 0:off 1:on	"0" is read.

Port K input enable control register

PKIE (0x4000_02B8)		7	6	5	4	3	2	1	0
	Bit Symbol	—	—	—	—	—	PK2IE	PK1IE	PK0IE
	Read/Write	R					R/W		
	After reset	0					0	0	0
	Function	“0” is read.					Input 0:disabled 1:enabled	Input 0:disabled 1:enabled	Input 0:disabled 1:enabled

8. 16-bit Timer/Event Counters (TMRBs)

8.1 Outline

Each of the ten channels (TMRB0 through TMRB9) has a multi-functional 16-bit timer/event counter. TMRBs operate in the following four operation modes:

- 16-bit interval timer mode
- 16-bit event counter mode
- 16-bit programmable square-wave output mode (PPG)
- Timer synchronous mode (capable of setting output mode for each 4ch)

The use of the capture function allows TMRBs to perform the following three measurements

- Frequency measurement
- Pulse width measurement
- Time difference measurement

8.2 Differences in the Specifications

Each channel (TMRB0 through TMRB9) functions independently and the channels operate in the same way, except for the differences in their specifications as shown in Table 8-1 and Table 8-2 and the two-phase pulse count function. Therefore, the operational descriptions here are only for TMRB1.

The channels shown below are used as the capture or start trigger.

(1) The flip-flop output of TMRB 7 through TMRB 9 can be used as the capture trigger of other channels.

- TB7OUT => available for TMRB 0 through TMRB 1
- TB8OUT => available for TMRB 2 through TMRB 4
- TB9OUT => available for TMRB 5 through TMRB 6

(2) The start trigger of the timer synchronous mode (with TBxRUN)

- TMRB0 => can start TMRB 0 through TMRB 3 synchronously
- TMRB4 => can start TMRB 4 through TMRB 7 synchronously

Table 8-1 Differences in the Specifications of TMRB Modules (1)

Specification Channel	External pins		Internal signals	
	External clock/ capture trigger input pins	Timer flip-flop output pin	Timer for capture triggers	Timer for synchronous stat triggers
TMRB0	TB0IN0 (shared with PH0) TB0IN1 (shared with PH1)	TB0OUT (shared with PI0)	TB7OUT	TB0PRUN, TB0RUN
TMRB1	TB1IN0 (shared with PH2) TB1IN1 (shared with PH3)	TB1OUT (shared with PI1)	TB7OUT	TB0PRUN, TB0RUN
TMRB2	TB2IN0 (shared with PH4) TB2IN1 (shared with PH5)	TB2OUT (shared with PI2)	TB8OUT	TB0PRUN, TB0RUN
TMRB3	TB3IN0 (shared with PH6) TB3IN1 (shared with PH7)	TB3OUT (shared with PI3)	TB8OUT	TB0PRUN, TB0RUN
TMRB4	TB4IN0 (shared with PI6) TB4IN1 (shared with PI7)	TB4OUT (shared with PI4)	TB8OUT	TB4PRUN, TB4RUN
TMRB5	TB5IN0 (shared with PD0) TB5IN1 (shared with PD1)	TB5OUT (shared with PI5)	TB9OUT	TB4PRUN, TB4RUN
TMRB6	TB6IN0 (shared with PD2) TB6IN1 (shared with PD3)	TB6OUT (shared with PJ4)	TB9OUT	TB4PRUN, TB4RUN
TMRB7	—	TB7OUT (shared with PJ5)	—	TB4PRUN, TB4RUN
TMRB8	—	TB8OUT (shared with PG7)	—	—
TMRB9	—	TB9OUT (shared with PK2)	—	—

Table 8-2 Differences in the Specifications of TMRB Modules (2)

Specification Channel	Interrupt	
	Capture interrupt	TMRB interrupt
TMRB0	INTCA000 INTCAP01	INTTB0
TMRB1	INTCAP10 INTCAP11	INTTB1
TMRB2	INTCAP20 INTCAP21	INTTB2
TMRB3	INTCAP30 INTCAP31	INTTB3
TMRB4	INTCAP40 INTCAP40	INTTB4
TMRB5	INTCAP50 INTCAP51	INTTB5
TMRB6	INTCAP60 INTCAP61	INTTB6
TMRB7	—	—
TMRB8	—	—
TMRB9	—	—

Configuration

Each channel consists of a 16-bit up-counter, two 16-bit timer registers (one of which is double-buffered, except for TBRB0 with one double-buffered 16-bit timer register), two 16-bit capture registers, two comparators, a capture input control, a timer flip-flop and its associated control circuit.

Timer operation modes and the timer flip-flop are controlled by a register.

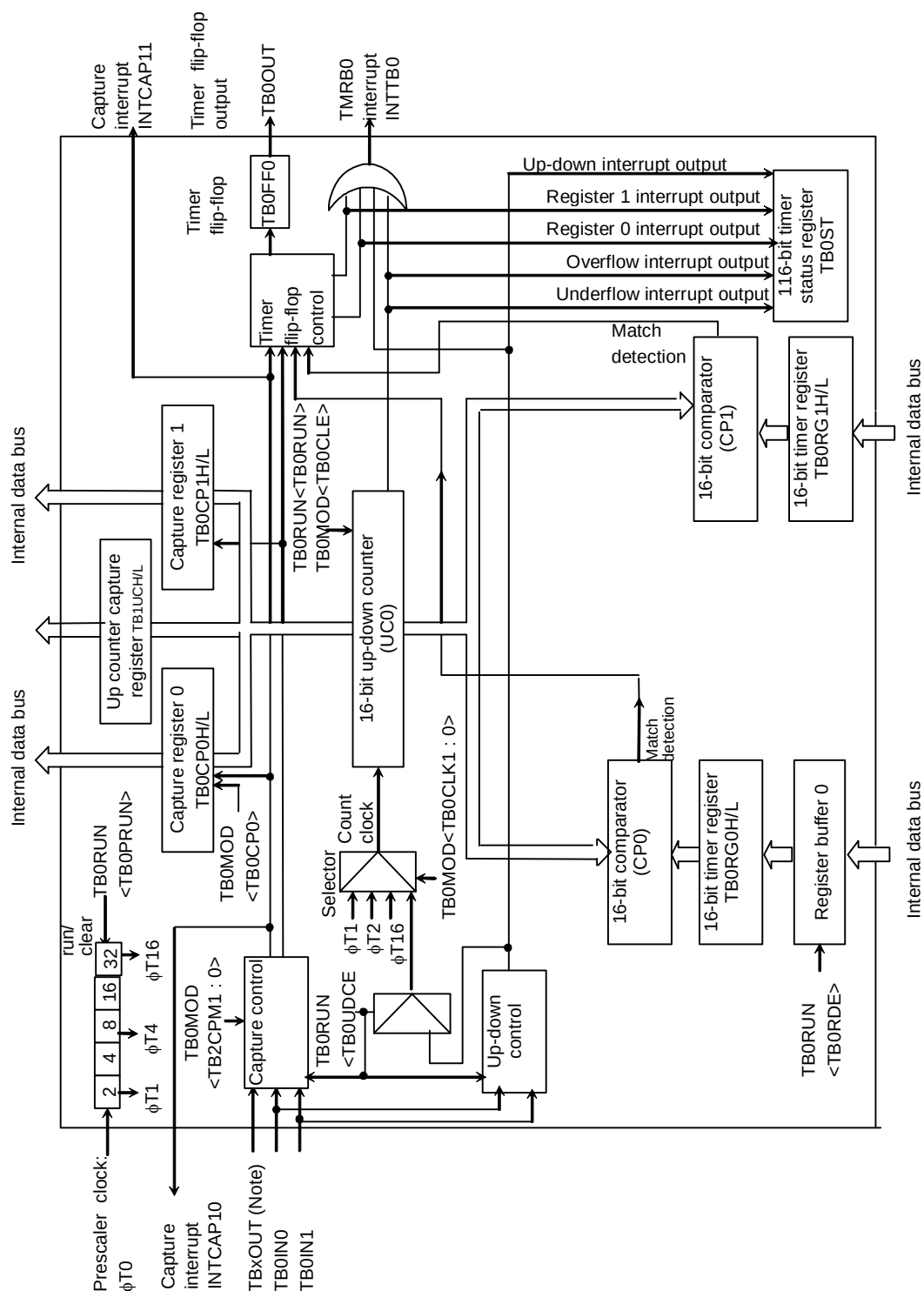


Fig. 8-1 TMRB1 Block Diagram (the same applies to channels 0 and 2 through 9)

(Note) TB7OUT is input to channels 0 and 1. TB8OUT is input to channels 2 through 4. TB9OUT is input to channels 5 and 6. Channels 7 through 9 do not have input pins for TBxOUT, TBnIN0 and TBnIN1. Also, channels 7 through 9 do not have output pins for INTTBn, INTCAPn0 and INTCAPn1.

Registers

8.2.1 TMRB registers

Table 8-3 shows the register names and addresses of each channel.

Table 8-3 TMRB registers (1/2)

Channel		TMRB0		TMRB1		TMRB2		TMRB3	
Specification									
Register names (addresses)	Timer enable register	TB0EN	0x4001_0000	TB1EN	0x4001_0040	TB2EN	0x4001_0080	TB3EN	0x4001_00C0
	Timer RUN register	TB0RUN	0x4001_0004	TB1RUN	0x4001_0044	TB2RUN	0x4001_0084	TB3RUN	0x4001_00C4
	Timer control register	TB0CR	0x4001_0008	TB1CR	0x4001_0048	TB2CR	0x4001_0088	TB3CR	0x4001_00C8
	Timer mode register	TB0MOD	0x4001_000C	TB1MOD	0x4001_004C	TB2MOD	0x4001_008C	TB3MOD	0x4001_00CC
	Timer flip-flop control register	TB0FFCR	0x4001_0010	TB1FFCR	0x4001_0050	TB2FFCR	0x4001_0090	TB3FFCR	0x4001_00D0
	Timer status register	TB0ST	0x4001_0014	TB1ST	0x4001_0054	TB2ST	0x4001_0094	TB3ST	0x4001_00D4
	Interrupt mask register	TB0IM	0x4001_0018	TB1IM	0x4001_0058	TB2IM	0x4001_0098	TB3IM	0x4001_00D8
	Timer up counter register	TB0UC	0x4001_001C	TB1UC	0x4001_005C	TB2UC	0x4001_009C	TB3UC	0x4001_00DC
	Timer register	TB0RG0	0x4001_0020	TB1RG0	0x4001_0060	TB2RG0	0x4001_00A0	TB3RG0	0x4001_00E0
		TB0RG1	0x4001_0024	TB1RG1	0x4001_0064	TB2RG1	0x4001_00A4	TB3RG1	0x4001_00E4
	Capture register	TB0CP0	0x4001_0028	TB1CP0	0x4001_0068	TB2CP0	0x4001_00A8	TB3CP0	0x4001_00E8
		TB0CP1	0x4001_002C	TB1CP1	0x4001_006C	TB2CP1	0x4001_00AC	TB3CP1	0x4001_00EC

Channel		TMRB4		TMRB5		TMRB6		TMRB7	
Specification									
Register names (addresses)	Timer enable register	TB4EN	0x4001_0100	TB5EN	0x4001_0140	TB6EN	0x4001_0180	TB7EN	0x4001_01C0
	Timer RUN register	TB4RUN	0x4001_0104	TB5RUN	0x4001_0144	TB6RUN	0x4001_0184	TB7RUN	0x4001_01C4
	Timer control register	TB4CR	0x4001_0108	TB5CR	0x4001_0148	TB6CR	0x4001_0188	TB7CR	0x4001_01C8
	Timer mode register	TB4MOD	0x4001_010C	TB5MOD	0x4001_014C	TB6MOD	0x4001_018C	TB7MOD	0x4001_01CC
	Timer flip-flop control register	TB4FFCR	0x4001_0110	TB5FFCR	0x4001_0150	TB6FFCR	0x4001_0190	TB7FFCR	0x4001_01D0
	Timer status register	TB4ST	0x4001_0114	TB5ST	0x4001_0154	TB6ST	0x4001_0194	TB7ST	0x4001_01D4
	Interrupt mask register	TB4IM	0x4001_0118	TB5IM	0x4001_0158	TB6IM	0x4001_0198	TB7IM	0x4001_01D8
	Timer up counter register	TB4UC	0x4001_011C	TB5UC	0x4001_015C	TB6UC	0x4001_019C	TB7UC	0x4001_01DC
	Timer register	TB4RG0	0x4001_0120	TB5RG0	0x4001_0160	TB6RG0	0x4001_01A0	TB7RG0	0x4001_01E0
		TB4RG1	0x4001_0124	TB5RG1	0x4001_0164	TB6RG1	0x4001_01A4	TB7RG1	0x4001_01E4
	Capture register	TB4CP0	0x4001_0128	TB5CP0	0x4001_0168	TB6CP0	0x4001_01A8	TB7CP0	0x4001_01E8
		TB4CP1	0x4001_012C	TB5CP1	0x4001_016C	TB6CP1	0x4001_01AC	TB7CP1	0x4001_01EC

Table 8-4 TMRB registers (2/2)

Channel		TMRB8		TMRB9	
Specification					
Register names (addresses)	Timer enable register	TB8EN	0x4001_0200	TB9EN	0x4001_0240
	Timer RUN register	TB8RUN	0x4001_0204	TB9RUN	0x4001_0244
	Timer control register	TB8CR	0x4001_0208	TB9CR	0x4001_0248
	Timer mode register	TB8MOD	0x4001_020C	TB9MOD	0x4001_024C
	Timer flip-flop control register	TB8FFCR	0x4001_0210	TB9FFCR	0x4001_0250
	Timer status register	TB8ST	0x4001_0214	TB9ST	0x4001_0254
	Interrupt mask register	TB8IM	0x4001_0218	TB9IM	0x4001_0258
	Timer up counter register	TB8UC	0x4001_021C	TB9UC	0x4001_025C
	Timer register	TB8RG0	0x4001_0220	TB9RG0	0x4001_0260
		TB8RG1	0x4001_0224	TB9RG1	0x4001_0264
	Capture register	TB8CP0	0x4001_0228	TB9CP0	0x4001_0268
		TB8CP1	0x4001_022C	TB9CP1	0x4001_026C

8.2.1.1 TMRBn enable register (channels 0 through 9)

TBnEN
(0x4001_0xx0)

TMRBn enable register (n=0~9)								
	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
bit Symbol	TBnEN							
Read/Write	R/W	R						
After reset	0	0						
Function	TMRBn operation 0: Disabled 1: Enabled	"0" is read.						

<TBnEN>: Specifies the TMRB operation. When the operation is disabled, no clock is supplied to the other registers in the TMRB module. This can reduce power consumption. (This disables reading from and writing to the other registers.) To use the TMRB, enable the TMRB operation (set to "1") before programming each register in the TMRB module. If the TMRB operation is executed and then disabled, the settings will be maintained in each register.

8.2.1.2 TMRB RUN register (channels 0 through 9)

TMRBn RUN register (n=0~9)

TBnRUN (0x4001_0xx4)		31	30	29	28	27	26	25	24
	bit Symbol								
	Read/Write	R	R	R	R	R	R	R	R
	After reset	0	0	0	0	0	0	0	0
		23	22	21	20	19	18	17	16
	bit Symbol								
	Read/Write	R	R	R	R	R	R	R	R
	After reset	0	0	0	0	0	0	0	0
		15	14	13	12	11	10	9	8
	bit Symbol								
	Read/Write	R	R	R	R	R	R	R	R
	After reset	0	0	0	0	0	0	0	0
		7	6	5	4	3	2	1	0
	bit Symbol						TBnPRUN		TBnRUN
	Read/Write	R					R/W	R	R/W
	After reset	0					0	0	0
	Function	"0" is read.					Timer Run/Stop Control 0: Stop & clear 1: Count * The first bit can be read as "0."		

<TBnRUN> :Controls the TMRB0 count operation.

<TBnPRUN> :Controls the TMRB0 prescaler operation.

8.2.1.3 TMRB control register (channels 0 through 9)

TMRBn control register (n=0~9)

TBnCR
(0x4001_0xx8)

	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
bit Symbol	TBnWBF		TBnSYNC		I2TBn			
Read/Write	R/W	R/W	R/W	R	R/W		R	
After reset	0	0	0	0	0		0	
Function	Double Buffer 0: Disabled 1: Enabled	Write "0".	Timer operation mode 0: individual 1: synchronous	"0" is read.	IDLE 0: Stop 1: Operation	"0" is read.		

<I2TBm>: Controls the operation in the IDLE mode.

<TBnSYNC>: Controls operation mode of timers.

"0": timers operate individually.

"1": timers operate synchronously.

<TBmWBF>: Controls the enabling/disabling of double buffering.

8.2.1.4 TMRB mode register (channels 0 thorough 9)

TMRBn mode register(n=0~9)

TBnMOD
(0x4001_0xxC)

	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
bit Symbol			TBnCP0	TBnCPM1	TBnCPM0	TBnCLE	TBnCLK1	TBnCLK0
Read/Write	R	R/W	W	R/W				
After reset	0	0	1	0	0	0	0	0
Function	"0" is read.	Write "0".	Capture control by software 0: Capture by software 1: Don't care	Capture timing 00: Disable Capture timing 00: Disable 01: TBnIN0 ↑ TBnIN1 ↑ 10: TBnIN0 ↑ TBnIN0 ↓ 11: CAPTRG ↑ CAPTRG ↓		Up-counter control 0: Clear/disable 1: clear/enable	Selects source clock 00: TBnIN0 pin input 01: φT1 10: φT4 11: φT16	

<TBnCLK1:0>:Selects the TMRBn timer count clock.

<TBnCLE>:Clears and controls the TMRBn up-counter.

"0": Disables clearing of the up-counter.

"1": Clears up-counter if there is a match with timer register 1 (TBnRG1).

<TBnCPM1:0>:Specifies TMRBn capture timing.

"00": Capture disable

"01": Takes count values into capture register 0 (TBnCP0) upon rising of TBnIN0 pin input.
Takes count values into capture register 1 (TBnCP1) upon rising of TBnIN1 pin input.

"10": Takes count values into capture register 0 (TBnCP0) upon rising of TBnIN0 pin input.
Takes count values into capture register 1 (TBnCP1) upon falling of TBnIN0 pin input.

"11": Takes count values into capture register 0 (TBnCP0) upon rising of 16-bit timer match output (TBxOUT) and into capture register 1 (TBnCP1) upon falling of TBxOUT (TMRB0 and TMRB1:TB7OUT, TMRB2 through TMRB4:TB8OUT, TMRB5 and TMRB6:TB9OUT).

<TBnCP0>:Captures count values by software and takes them into capture register 0 (TBnCP0).

(Note 1) The value read from bit 5 of TBnMOD is "1".**(Note 2)** Input from TBnIN0 and TBnIN1 is available only for channels TMRB0 thorough 6.

8.2.1.5 TMRB flip-flop control register (channels 0 through 9)

TMRBn flip-flop control register (n=0~9)

TBnFFCR (0x4001_0xx0)		31	30	29	28	27	26	25	24
	bit Symbol								
	Read/Write	R	R	R	R	R	R	R	R
	After reset	0	0	0	0	0	0	0	0
		23	22	21	20	19	18	17	16
	bit Symbol								
	Read/Write	R	R	R	R	R	R	R	R
	After reset	0	0	0	0	0	0	0	0
		15	14	13	12	11	10	9	8
	bit Symbol								
	Read/Write	R	R	R	R	R	R	R	R
	After reset	0	0	0	0	0	0	0	0
		7	6	5	4	3	2	1	0
	bit Symbol			TBnC1T1	TBnC0T1	TBnE1T1	TBnE0T1	TBnFF0C1	TBnFF0C0
	Read/Write	R		R/W				R/W	
	After reset	1	1	0	0	0	0	1	1
	Function	“11” is read.		TBnFF0 reverse trigger				TBnFF0 control 00: Invert 01: Set 10: Clear 11: Don't care * This is always read as “11.”	
				0: Disable trigger					
				1: Enable trigger					
				When the up-counter value is taken into TBnCP1	When the up-counter value is taken into TBnCP0	When the up-counter matches TBnRG1	When the up-counter matches TBnRG0		

<TBnFF0C1:0>:Controls the timer flip-flop.

"00": Reverses the value of TBnFF0 (reverse by using software).

"01": Sets TBnFF0 to "1".

"10": Clears TBnFF0 to "0".

"11":Don't care

<TBnE1:0>:Reverses the timer flip-flop when the up-counter matches the timer register 0,1 (TBnRG0,1).

<TBnC1:0>:Reverses the timer flip-flop when the up-counter value is taken into the capture register 0,1 (TBnCP0,1).

8.2.1.6 TMRB status register (channels 0 through 9)

TMRBn status register (n=0~9)

TnST (0x4001_0xx4)		31	30	29	28	27	26	25	24
	bit Symbol								
	Read/Write	R	R	R	R	R	R	R	R
	After reset	0	0	0	0	0	0	0	0
		23	22	21	20	19	18	17	16
	bit Symbol								
	Read/Write	R	R	R	R	R	R	R	R
	After reset	0	0	0	0	0	0	0	0
		15	14	13	12	11	10	9	8
	bit Symbol								
	Read/Write	R	R	R	R	R	R	R	R
	After reset	0	0	0	0	0	0	0	0
		7	6	5	4	3	2	1	0
	bit Symbol						INTTBOFn	INTTBn1	INTTBn0
	Read/Write	R					R		
	After reset	0					0	0	0
	Function	"0" is read.					0: Interrupt not generated 1: Interrupt generated	0: Interrupt not generated 1: Interrupt generated	0: Interrupt not generated 1: Interrupt generated

<INTTBn0>:Interrupt generated if there is a match with timer register 0 (TBnRG0)

<INTTBn1>:Interrupt generated if there is a match with timer register 1 (TBnRG1)

<INTTBOFn>:Interrupt generated if an up-counter overflow occurs

(Note) If any interrupt is generated, the flag that corresponds to the interrupt is set to TnST and the generation of interrupt is notified to INTC. The flag is cleared by reading the TnST register.

8.2.1.7 TMRB interrupt mask register (channels 0 through 9)

TMRBn interrupt mask register (n=0~9)

TBnIM
(0x4001_0xx8)

	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
bit Symbol						TBIMOFn	TBIMn1	TBIMn0
Read/Write	R					R		
After reset	0					0	0	0
Function	"0" is read					1: Interrupt is masked	1: Interrupt is masked	1: Interrupt is masked

<TBIMn0>: Interrupt is masked if there is a match with timer register 0 (TBnRG0)

<TBIMn1>: Interrupt is masked if there is a match with timer register 1 (TBnRG1).

<TBIMOFn>: Interrupt is masked if an up-and-down counter overflow occurs.

8.2.1.8 TMRB read capture register (channels 0 through 9)

TBnUC read capture register (n=0~9)

TBnUC
(0x4001_0xxC)

	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
bit Symbol	UCn15	UCn14	UCn13	UCn12	UCn11	UCn10	UCn9	UCn8
Read/Write	R/W							
After reset	0							
Function	Data obtained by read capture: 15-8 bit data							
	7	6	5	4	3	2	1	0
bit Symbol	UCn7	UCn6	UCn5	UCn4	UCn3	UCn2	UCn1	UCn0
Read/Write	R/W							
After reset	0							
Function	Data obtained by read capture: 7-0 bit							

8.2.1.9 TMRB timer register (channels 0 through 9)

TBnRG0 timer register (n=0~9)

TBnRG0
(0x4001_0xx0)

	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
bit Symbol	TBnRG01 5	TBnRG01 4	TBnRG01 3	TBnRG01 2	TBnRG01 1	TBnRG01 0	TBnRG09	TBnRG08
Read/Write	R/W							
After reset	0							
Function	Timer count value: 15-8 bit data							
	7	6	5	4	3	2	1	0
bit Symbol	TBnRG07	TBnRG06	TBnRG05	TBnRG04	TBnRG03	TBnRG02	TBnRG01	TBnRG00
Read/Write	R/W							
After reset	0							
Function	Timer count value: 7-0 bit data							

TBnRG1 timer register (n=0~9)

TBnRG1
(0x4001_0xx4)

	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
bit Symbol	TBnRG11 5	TBnRG11 4	TBnRG11 3	TBnRG11 2	TBnRG11 1	TBnRG11 0	TBnRG19	TBnRG18
Read/Write	R/W							
After reset	0							
Function	Timer count value: 15-8 bit data							
	7	6	5	4	3	2	1	0
bit Symbol	TBnRG17	TBnRG16	TBnRG15	TBnRG14	TBnRG13	TBnRG12	TBnRG11	TBnRG10
Read/Write	R/W							
After reset	0							
Function	Timer count value: 7-0 bit data							

8.2.1.10 TMRB capture register (channels 0 through 9)

TBnCP0capture register (n=0~9)

TBnCP0 (0x4001_0xx8)		31	30	29	28	27	26	25	24
	bit Symbol								
	Read/Write	R	R	R	R	R	R	R	R
	After reset	0	0	0	0	0	0	0	0
		23	22	21	20	19	18	17	16
	bit Symbol								
	Read/Write	R	R	R	R	R	R	R	R
	After reset	0	0	0	0	0	0	0	0
		15	14	13	12	11	10	9	8
	bit Symbol	TBnCP01	TBnCP01	TBnCP01	TBnCP01	TBnCP01	TBnCP01	TBnCP09	TBnCP08
		5	4	3	2	1	0		
	Read/Write	R							
	After reset	Undefined							
	Function	Timer capture value: 15-8 bit data							
		7	6	5	4	3	2	1	0
	bit Symbol	TBnCP07	TBnCP06	TBnCP05	TBnCP04	TBnCP03	TBnCP02	TBnCP01	TBnCP00
	Read/Write	R							
	After reset	Undefined							
	Function	Timer capture value: 7-0 bit data							

TBnCP1 capture register (n=0~9)

TBnCP1 (0x4001_0xxC)		31	30	29	28	27	26	25	24
	bit Symbol								
	Read/Write	R	R	R	R	R	R	R	R
	After reset	0	0	0	0	0	0	0	0
		23	22	21	20	19	18	17	16
	bit Symbol								
	Read/Write	R	R	R	R	R	R	R	R
	After reset	0	0	0	0	0	0	0	0
		15	14	13	12	11	10	9	8
	bit Symbol	TBnCP11	TBnCP11	TBnCP11	TBnCP11	TBnCP11	TBnCP11	TBnCP19	TBnCP18
		5	4	3	2	1	0		
	Read/Write	R							
	After reset	Undefined							
	Function	Timer capture value: 15-8 bit data							
		7	6	5	4	3	2	1	0
	bit Symbol	TBnCP17	TBnCP16	TBnCP15	TBnCP14	TBnCP13	TBnCP12	TBnCP11	TBnCP10
	Read/Write	R							
	After reset	Undefined							
	Function	Timer capture value: 7-0 bit data							

8.3 Description of Operations for Each Circuit

The channels operate in the same way, except for the differences in their specifications as shown in Table 8-1 and 8-2. Therefore, the operational descriptions here are only for channel 0.

8.3.1 Prescaler

There is a 4-bit prescaler to generate the source clock for up-counter UC0. The prescaler input clock ϕ_{T0} is $f_{\text{periph}}/1$, $f_{\text{periph}}/2$, $f_{\text{periph}}/4$, $f_{\text{periph}}/8$, $f_{\text{periph}}/16$ or $f_{\text{periph}}/32$ selected by SYSCR1<FPSEL> in the CG. The peripheral clock, f_{periph} , is either f_{gear} , a clock selected by SYSCR1<FPSEL> in the CG, or f_c , which is a clock before it is divided by the clock gear.

The operation or the stoppage of a prescaler is set with TB0RUN<TB0PRUN> where writing “1” starts counting and writing “0” clears and stops counting. Table 8-5 and 8-6 show prescaler output clock resolutions.

Table 8-5 Prescaler Output Clock Resolutions @fc = 40MHz

Release peripheral clock <FPSEL>	Clock gear value <GEAR2:0>	Select prescaler clock <PRCK1:0>	Prescaler output clock resolutions		
			$\phi T1$	$\phi T4$	$\phi T16$
0 (fgear)	000 (fc)	000(fperiph/1)	$fc/2^1(0.05\mu s)$	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$
		001(fperiph/2)	$fc/2^2(0.1\mu s)$	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$
		010(fperiph/4)	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$
		011(fperiph/8)	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$
		100(fperiph/16)	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$
		101(fperiph/32)	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$
	100(fc/2)	000(fperiph/1)	$fc/2^1(0.05\mu s)$	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$
		001(fperiph/2)	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$
		010(fperiph/4)	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$
		011(fperiph/8)	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$
		100(fperiph/16)	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$
		101(fperiph/32)	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$	$fc/2^{11}(51.2\mu s)$
	101(fc/4)	000(fperiph/1)	$fc/2^1(0.05\mu s)$	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$
		001(fperiph/2)	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$
		010(fperiph/4)	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$
		011(fperiph/8)	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$
		100(fperiph/16)	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$	$fc/2^{11}(51.2\mu s)$
		101(fperiph/32)	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$	$fc/2^{12}(102.4\mu s)$
	110(fc/8)	000(fperiph/1)	$fc/2^1(0.05\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$
		001(fperiph/2)	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$
		010(fperiph/4)	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$
		011(fperiph/8)	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$	$fc/2^{11}(51.2\mu s)$
		100(fperiph/16)	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$	$fc/2^{12}(102.4\mu s)$
		101(fperiph/32)	$fc/2^9(12.8\mu s)$	$fc/2^{11}(51.2\mu s)$	$fc/2^{13}(204.8\mu s)$
1 (fc)	000 (fc)	000(fperiph/1)	$fc/2^1(0.05\mu s)$	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$
		001(fperiph/2)	$fc/2^2(0.1\mu s)$	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$
		010(fperiph/4)	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$
		011(fperiph/8)	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$
		100(fperiph/16)	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$
		101(fperiph/32)	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$
	100(fc/2)	000(fperiph/1)	—	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$
		001(fperiph/2)	$fc/2^2(0.1\mu s)$	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$
		010(fperiph/4)	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$
		011(fperiph/8)	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$
		100(fperiph/16)	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$
		101(fperiph/32)	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$
	101(fc/4)	000(fperiph/1)	—	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$
		001(fperiph/2)	—	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$
		010(fperiph/4)	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$
		011(fperiph/8)	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$
		100(fperiph/16)	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$
		101(fperiph/32)	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$
	110(fc/8)	000(fperiph/1)	—	—	$fc/2^5(0.8\mu s)$
		001(fperiph/2)	—	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$
		010(fperiph/4)	—	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$
		011(fperiph/8)	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$
		100(fperiph/16)	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$
		101(fperiph/32)	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$

- (Note 1) The prescaler output clock ϕT_n must be selected so that $\phi T_n < f_{sys}$ is satisfied (so that ϕT_n is slower than f_{sys}).
- (Note 2) Do not change the clock gear while the timer is operating.
- (Note 3) “—” denotes a setting prohibited.

Table 8-6 Prescaler Output Clock Resolutions @ = 32MHz

Release peripheral clock <FPSEL>	Clock gear value <GEAR2:0>	Select prescaler clock <PRCK2:0>	Prescaler output clock resolutions		
			$\phi T1$	$\phi T4$	$\phi T16$
0 (fgear)	000 (fc)	000(fperiph/1)	$fc/2^1(0.0625\mu s)$	$fc/2^3(0.25\mu s)$	$fc/2^5(1.0\mu s)$
		001(fperiph/2)	$fc/2^2(0.125\mu s)$	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$
		010(fperiph/4)	$fc/2^3(0.25\mu s)$	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$
		011(fperiph/8)	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$
		100(fperiph/16)	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$
		101(fperiph/32)	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$
	100(fc/2)	000(fperiph/1)	$fc/2^1(0.0625\mu s)$	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$
		001(fperiph/2)	$fc/2^3(0.25\mu s)$	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$
		010(fperiph/4)	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$
		011(fperiph/8)	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$
		100(fperiph/16)	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$
		101(fperiph/32)	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$	$fc/2^{11}(64.0\mu s)$
	101(fc/4)	000(fperiph/1)	$fc/2^1(0.0625\mu s)$	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$
		001(fperiph/2)	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$
		010(fperiph/4)	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$
		011(fperiph/8)	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$
		100(fperiph/16)	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$	$fc/2^{11}(64.0\mu s)$
		101(fperiph/32)	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$	$fc/2^{12}(128.0\mu s)$
	110(fc/8)	000(fperiph/1)	$fc/2^1(0.0625\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$
		001(fperiph/2)	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$
		010(fperiph/4)	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$
		011(fperiph/8)	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$	$fc/2^{11}(64.0\mu s)$
		100(fperiph/16)	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$	$fc/2^{12}(128.0\mu s)$
		101(fperiph/32)	$fc/2^9(16.0\mu s)$	$fc/2^{11}(64.0\mu s)$	$fc/2^{13}(256.0\mu s)$
1 (fc)	000 (fc)	000(fperiph/1)	$fc/2^1(0.0625\mu s)$	$fc/2^3(0.25\mu s)$	$fc/2^5(1.0\mu s)$
		001(fperiph/2)	$fc/2^2(0.125\mu s)$	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$
		010(fperiph/4)	$fc/2^3(0.25\mu s)$	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$
		011(fperiph/8)	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$
		100(fperiph/16)	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$
		101(fperiph/32)	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$
	100(fc/2)	000(fperiph/1)	—	$fc/2^3(0.25\mu s)$	$fc/2^5(1.0\mu s)$
		001(fperiph/2)	$fc/2^2(0.125\mu s)$	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$
		010(fperiph/4)	$fc/2^3(0.25\mu s)$	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$
		011(fperiph/8)	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$
		100(fperiph/16)	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$
		101(fperiph/32)	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$
	101(fc/4)	000(fperiph/1)	—	$fc/2^3(0.25\mu s)$	$fc/2^5(1.0\mu s)$
		001(fperiph/2)	—	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$
		010(fperiph/4)	$fc/2^3(0.25\mu s)$	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$
		011(fperiph/8)	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$
		100(fperiph/16)	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$
		101(fperiph/32)	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$
	110(fc/8)	000(fperiph/1)	—	—	$fc/2^5(1.0\mu s)$
		001(fperiph/2)	—	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$
		010(fperiph/4)	—	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$
		011(fperiph/8)	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$
		100(fperiph/16)	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$
		101(fperiph/32)	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$

- (Note 1) The prescaler output clock ϕT_n must be selected so that $\phi T_n < f_{sys}$ is satisfied (so that ϕT_n is slower than f_{sys}).
- (Note 2) Do not change the clock gear while the timer is operating.
- (Note 3) “—” denotes a setting prohibited.

8.3.2 Up-counter (UC0)

UC0 is a 16-bit binary counter.

- Source clock

UC0 source clock, specified by TB0MOD<TB0CLK1:0>, can be selected from either three types - $\phi T1$, $\phi T4$ and $\phi T16$ - of prescaler output clock or the external clock of the TB0IN0 pin.

- Count start/ stop

Counter operation is specified by TB0RUN<TB0RUN>. UC0 starts counting if <TB0RUN> = "1", and stops counting and clears counter value if <TB0RUN> = "0".

- Timing to clear UC0

1) When a match is detected

By setting TB0MOD<TB0CLE> = "1", UC0 is cleared if when the comparator detects a match between counter value and the value set in TB0RG1. UC0 operates as a free-running counter if TB0MOD<TB0CLE> = "0".

2) When UC0 stops

UC0 stops counting and clears counter value if TB0RUN <TB0RUN> = "0".

- UC0 overflow

If UC0 overflow occurs, the INTTB0 overflow interrupt is generated.

8.3.3 Timer registers (TB0RG0, TB0RG1)

TB0RG0 and TB0RG1 are registers for setting values to compare with up-counter values and two registers are built into each channel. If the comparator detects a match between a value set in this timer register and that in a UC0 up-counter, it outputs the match detection signal.

- Configuration

TB0RG0 and TB0RG1 of this timer registers are paired with register buffer 0 - a double-buffered configuration. The two registers use TB0CR<TB0WBF> to control the enabling/disabling of double buffering. If <TB0WBF> = "0", double buffering is disabled and if <TB0WBF> = "1", it is enabled. If double buffering is enabled, data is transferred from register buffer 0 to the TB0RG0 and TB0RG1 timer registers when there is a match between UC0 and TB0RG1.

- Default setting

The values of TB0RG0 and TB0RG1 become undefined after a reset. A reset disables the double buffer.

- Register setting

- 1) When not using double-buffering

To write data to the timer registers, either a 2-byte data transfer instruction or a 1-byte data transfer instruction written twice in the order of low-order 8 bits followed by high-order 8 bits can be used.

- 2) When using double-buffering

TB0RG0/ TB0RG1 and the register buffers are assigned to the same address. If <TB0WBF> = "0," the same value is written to TB0RG0, TB0RG1 and each register buffer; if <TB0WBF> = "1," the value is only written to each register buffer. To write an initial value to the timer register, therefore, the register buffers must be set to "disable". Then set <TB0WBF> = "1" and write the following data to the register.

8.3.4 Capture

This is a circuit that controls the timing of latching values from the UC0 up-counter into the TB0CP0 and TB0CP1 capture registers. The timing with which to latch data is specified by TB0MOD <TB0CPM1:0>.

Software can also be used to import values from the UC0 up-counter into the capture register; specifically, UC0 values are taken into the TB0CP0 capture register each time “0” is written to TB0MOD<TB0CP0>. To use this capability, the prescaler must be running (TB0RUN<TB0PRUN> = “1”).

8.3.5 Capture Registers (TB0CP0H/L, TB0CP1H/L)

These are 16-bit registers for latching values from the UC0 up-counter. To read data from the capture register, use a 16-bit data transfer instruction or read in the order of low-order bits followed by high-order bits.

8.3.6 Up-counter capture register (TB0UCH/L)

Other than the capturing functions shown above, the current count value of the UC0 can be captured by reading the TB0UC registers.

8.3.7 Comparators (CP0, CP1)

These are 16-bit comparators for detecting a match by comparing set values of the UC0 up-counter with set values of the TB0RG0 and TB0RG1 timer registers. If a match is detected, INTTB0 is generated.

8.3.8 Timer Flip-flop (TB0FF0)

The timer flip-flop (TB0FF0) is reversed by a match signal from the comparator and a latch signal to the capture registers. It can be enabled or disabled to reverse by setting the TB0FFCR<TB0C1T1, TB0C0T1, TB0E1T1, TB0E0T1>.

The value of TB0FF0 becomes undefined after a reset. The flip-flop can be reversed by writing “00” to TB0FFCR<TB0FF0C1:0>. It can be set to “1” by writing “01,” and can be cleared to “0” by writing “10.”

The value of TB0FF0 can be output to the timer output pin, TB0OUT (shared with PIO). To enable timer output, the port I related registers PICR and PIFR1 must be programmed beforehand.

8.3.9 Capture interrupt (INTCAP00, INTCAP01)

Interrupts INTCAP00 and INTCAP01 can be generated at the timing of latching values from the UC0 up-counter into the TB0CP0 and TB0CP1 capture registers. The interrupt timing is specified by NVIC.

8.4 Description of Operations for Each Mode

8.4.1 16-bit Interval Timer Mode

-Generating interrupts at periodic cycles

To generate the INTTB0 interrupt, specify a time interval in the TB0RG1 timer register.

8.4.2 16-bit Event Counter Mode

It is possible to make it the event counter by using an input clock as an external clock (TB0IN0 pin input).

The up-counter counts up on the rising edge of TB0IN0 pin input. It is possible to read the count value by capturing value using software and reading the captured value.

To use it as an event counter, put the prescaler in a "RUN" state (TB0RUN<TB0PRUN> = "1").

8.4.3 16-bit Programmable Square Wave Output Mode (PPG)

Square waves with any frequency and any duty (programmable square waves) can be output. The output pulse can be either low-active or high-active.

Programmable square waves can be output from the TB0OUT pin by triggering the timer flip-flop (TB0FF) to reverse when the set value of the up-counter (UCO) matches the set values of the timer registers (TB0RG0 and TB0RG1). Note that the set values of TB0RG0 and TB0RG1 must satisfy the following requirement:

$$(\text{Set value of TB0RG0}) < (\text{Set value of TB0RG1})$$

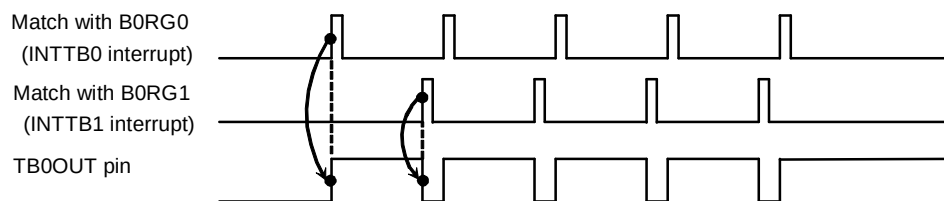


Fig. 8-2 Example of Output of Programmable Square Wave (PPG)

In this mode, by enabling the double buffering of TB0RG0, the value of register buffer 0 is shifted into TB0RG0 when the set value of the up-counter matches the set value of TB0RG1. This facilitates handling of small duties.

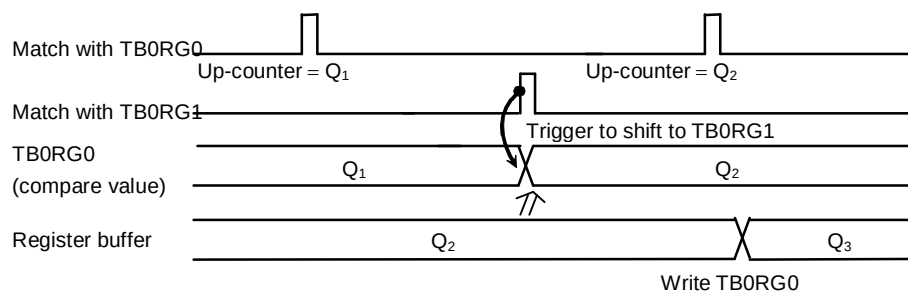


Fig. 8-3 Register Buffer Operation

The block diagram of this mode is shown below.

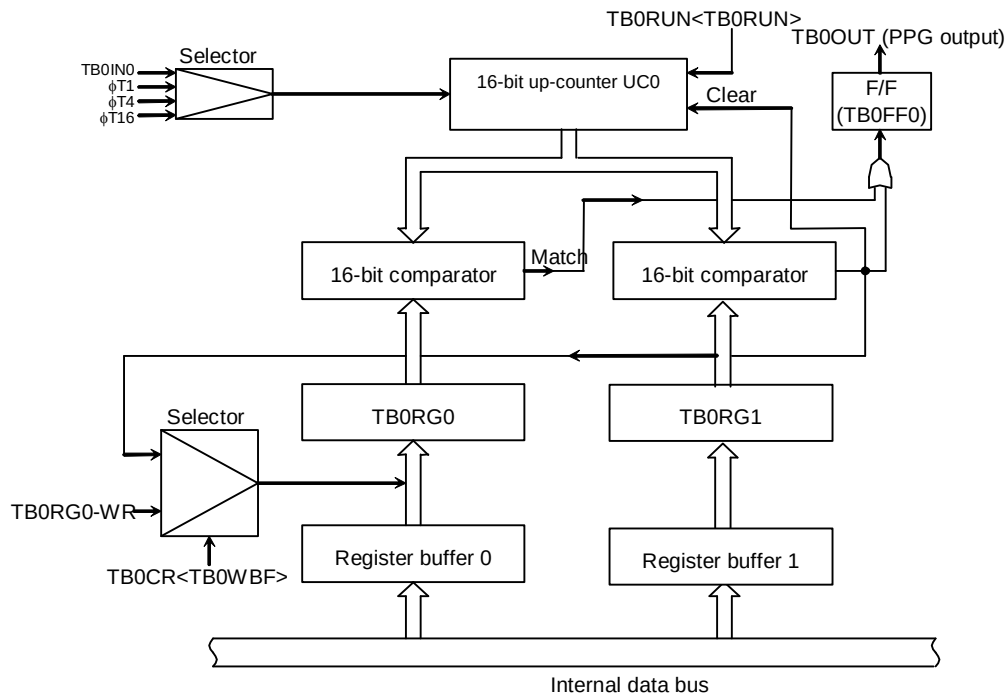


Fig. 8-4 Block Diagram of 16-bit PPG Mode

Each register in the 16-bit PPG output mode must be programmed as listed below.

	7	6	5	4	3	2	1	0	
TB0EN	← 1	X	X	X	X	X	X	X	Starts the TMRB0 module.
TB0RUN	← X	X	X	X	X	0	X	0	Stops the TMRB0 module.
TBORG0	← *	*	*	*	*	*	*	*	Specifies a duty. (16 bits *32-bits for register)
TBORG1	← *	*	*	*	*	*	*	*	Specifies a cycle. (16 bits *32-bits for register)
TB0CR	← 1	0	X	0	0	0	0	0	Enables the TBORG0 double buffering. (Changes the duty/cycle when the INTTB0 interrupt is generated)
TB0FFCR	← X	X	0	0	1	1	1	0	Specifies to trigger TB0FF0 to reverse when a match with TBORG0 or TBORG1 is detected, and sets the initial value of TB0FF0 to "0."
TB0MOD	← 0	0	1	0	0	1	*	*	Designates the prescaler output clock as the input clock, and disables the capture function.
PICR	← -	-	-	-	-	-	-	1	Assigns PI0 to TB0OUT
PIFR1	← -	-	-	-	-	-	-	1	
TB0RUN	← *	*	*	*	*	1	X	1	

X; Don't care -; no change

8.5 Timer synchronous mode

This mode enables the timers to start synchronously.

If the mode is used with PPG output, the output can be applied to drive a motor.

Use of the timer synchronous mode is specified in TBnCR<TBnSYNC>.

<TBnSYNC> = "0": Timers operates individually.

<TBnSYNC> = "1": Timers operate synchronously.

The channels are in two segments: channels TMRB0 through 3 and channels TMRB4 through 7.

If <TBnSYNC> = "1" is set, the start timing is synchronized with TMRB0 and TMRB4. The start timing of each channel, TBmRUN <TBmPRUN, TBmRUN> = "1,1", is ignored.

- (Note 1)** The channels designated for synchronous output must be started by TBmRUN<TBmPRUN, TBmRUN>="1,1" before the start triggered by TMRB0 and TMRB4.
- (Note 2)** Set TBnCR<TBnSYNC> to "0" unless the timer synchronous mode is used. The timer synchronous mode keeps the other channels operation waiting until TMRB0, TMRB4 and TMRB8 start operation.
- (Note 3)** TMRB0 and TMRB4 are the master clocks of the timer synchronous mode. Therefore, their TBnSYNC bit must be set to "0".
- (Note 4)** This mode cannot be applied to TMRB8 and TMRB9.

TBnCR
(0x4001_0xx8)

	7	6	5	4	3	2	1	0
bit Symbol	TBnWBF		TBnSYNC		I2TBn			
Read/Write	R/W	R/W	R/W	R	R/W	R	R	R
After reset	0	0	0	0	0	0	0	0
Function	Double Buffer 0:Disabled 1:Enabled	Write "0".	Timer operation 0:Individual	"0" is read.	IDLE 0:Stop 1:Operation	"0" is read.	"0" is read.	"0" is read.

Set the TBnSYNC bit of the timers, which are used as the slave clocks, to "1".

TBnCR
(0x4001_0xx8)

	7	6	5	4	3	2	1	0
bit Symbol	TBnWBF		TBnSYNC		I2TBn			
Read/Write	R/W	R/W	R/W	R	R/W	R	R	R
After reset	0	0	0	0	0	0	0	0
Function	Double Buffer 0:Disabled 1:Enabled	Write "0".	Timer operation 1:Synchronous	"0" is read.	IDLE 0:Stop 1:Operation	"0" is read.	"0" is read.	"0" is read.

8.6 Applications using the Capture Function

The capture function can be used to develop many applications, including those described below:

- ① One-shot pulse output triggered by an external pulse
- ② Frequency measurement
- ③ Pulse width measurement
- ④ Time difference measurement

① One-shot pulse output triggered by an external pulse

One-shot pulse output triggered by an external pulse is carried out as follows:

The 16-bit up-counter is made to count up by putting it in a free-running state using the prescaler output clock. An external pulse is input through the TB5IN0 pin. A trigger is generated at the rising of the external pulse by using the capture function and the value of the up-counter is taken into the capture registers (TB5CP0).

The NVIC must be programmed so that an interrupt `INTCAP50` is generated at the rising of an external trigger pulse. This interrupt is used to set the timer registers (TB5RG0) to the sum of the TB5CP0 value (c) and the delay time (d), (c + d), and set the timer registers (TB5RG1) to the sum of the TB5RG0 values and the pulse width (p) of one-shot pulse, (c + d + p).

TB5RG1 change must be completed before the next match.

In addition, the timer flip-flop control registers (TB5FFCR<TB5E1T1, TB5E0T1>) must be set to "11." This enables triggering the timer flip-flop (TB5FF0) to reverse when UC5 matches TB5RG0 and TB5RG1. This trigger is disabled by the `INTTB5` interrupt after a one-shot pulse is output.

Symbols (c), (d) and (p) used in the text correspond to symbols c, d and p in Fig. 8-5 One-shot Pulse Output (With Delay)."

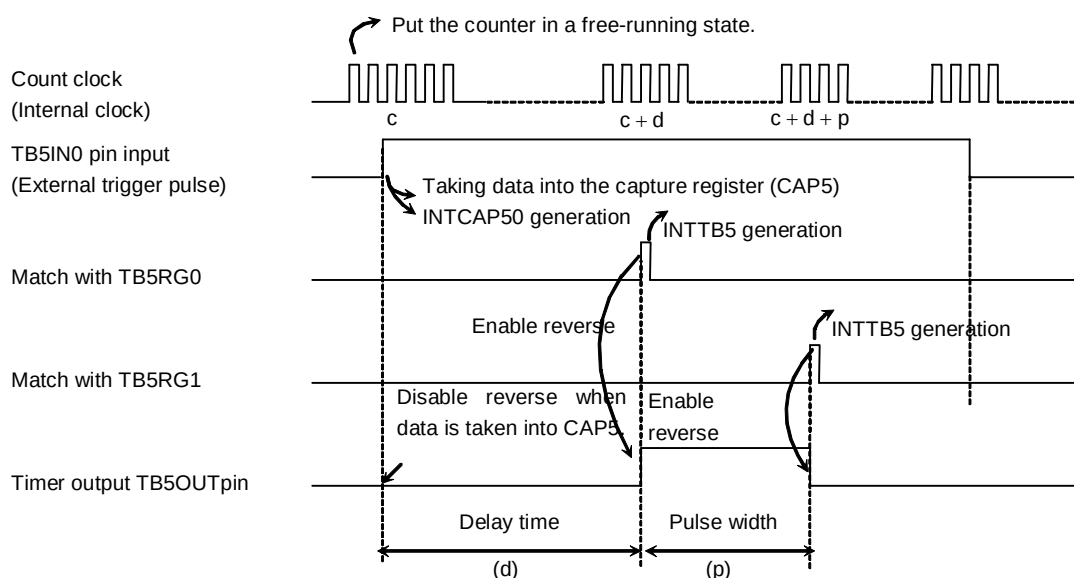


Fig. 8-5 One-shot Pulse Output (With Delay)

If a delay is not required, TB5FF0 is reversed when data is taken into TB5CP0, and TB5RG1 is set to the sum of the TB5CPO value (c) and the one-shot pulse width (p), ($c + p$), by generating the INT0 interrupt. TB5RG1 change must be completed before the next match. TB5FF0 is enabled to reverse when UC5 matches with TB5RG1, and is disabled by generating the INTTB5 interrupt.

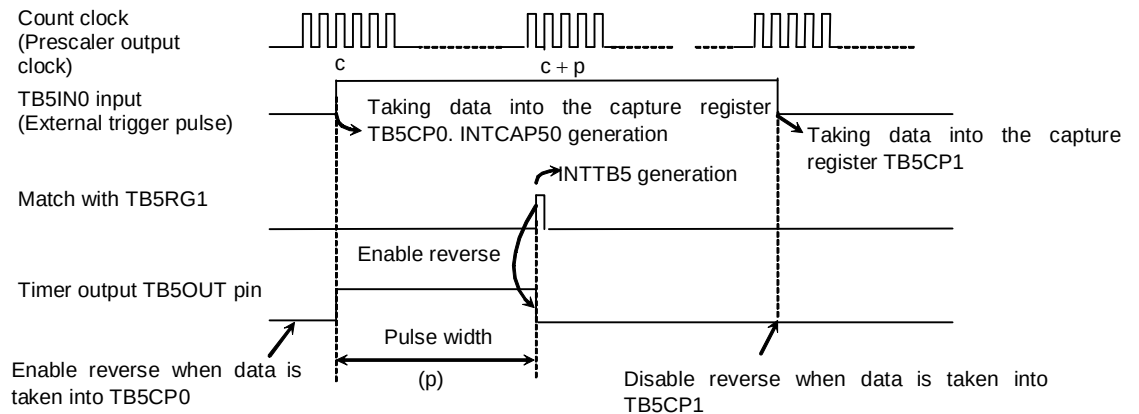


Fig. 8-6 One-shot Pulse Output Triggered by an External Pulse (Without Delay)

② Frequency measurement

The frequency of an external clock can be measured by using the capture function.

To measure frequency, another 16-bit timer (TMRB0) is used in combination with the 16-bit event counter mode (TMRB0 reverses TB0FFCR to specify the measurement time).

The TB3IN0 pin input is selected as the TMRB3 count clock to perform the count operation using an external input clock. TB3MOD<TB3CPM1:0> is set to "11." This setting allows a count value of the 16-bit up-counter UC3 to be taken into the capture register (TB3CP0) upon rising of a timer flip-flop (TB0FFCR) of the 16-bit timer (TMRB0), and an UC3 counter value to be taken into the capture register (TB3CP1) upon falling of TB0FF of the 16-bit timer (TMRB0).

A frequency is then obtained from the difference between TB3CP0 and TB3CP1 based on the measurement, by generating the INTTB0 16-bit timer interrupt.

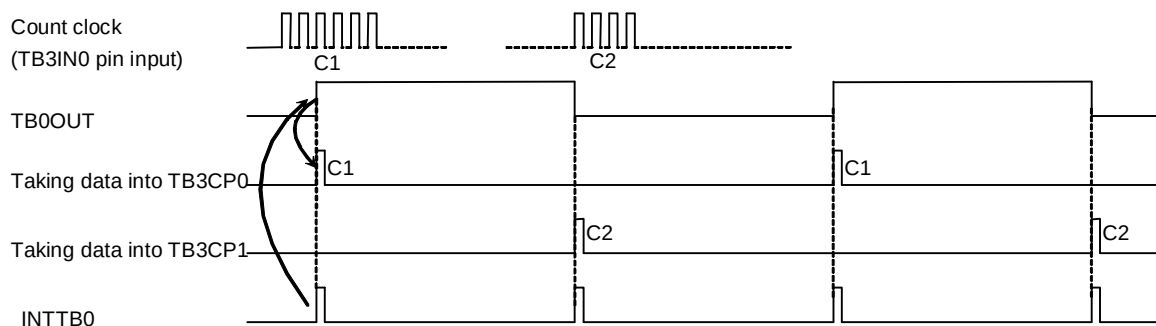


Fig. 8-7 Frequency Measurement

For example, if the set width of TB0FF level "1" of the 16-bit timer is 0.5 s and if the difference between TB3CP0 and TB3CP1 is 100, the frequency is $100 / 0.5 \text{ s} = 200 \text{ Hz}$.

③ Pulse width measurement

By using the capture function, the “H” level width of an external pulse can be measured. Specifically, by putting it in a free-running state using the prescaler output clock, an external pulse is input through the TB5IN0 pin and the up-counter (UC5) is made to count up. A trigger is generated at each rising and falling edge of the external pulse by using the capture function and the value of the up-counter is taken into the capture registers (TB5CP0, TB5CP1). The NVIC must be programmed so that INTCAP51 is generated at the falling edge of an external pulse input through the TB5IN0 pin.

The “H” level pulse width can be calculated by multiplying the difference between TB5CP0 and TB5CP1 by the clock cycle of an internal clock.

For example, if the difference between TB5CP0 and TB5CP1 is 100 and the cycle of the prescaler output clock is $0.5\ \mu\text{s}$, the pulse width is $100 \times 0.5\ \mu\text{s} = 50\ \mu\text{s}$.

Caution must be exercised when measuring pulse widths exceeding the UC5 maximum count time which is dependant upon the source clock used. The measurement of such pulse widths must be made using software.

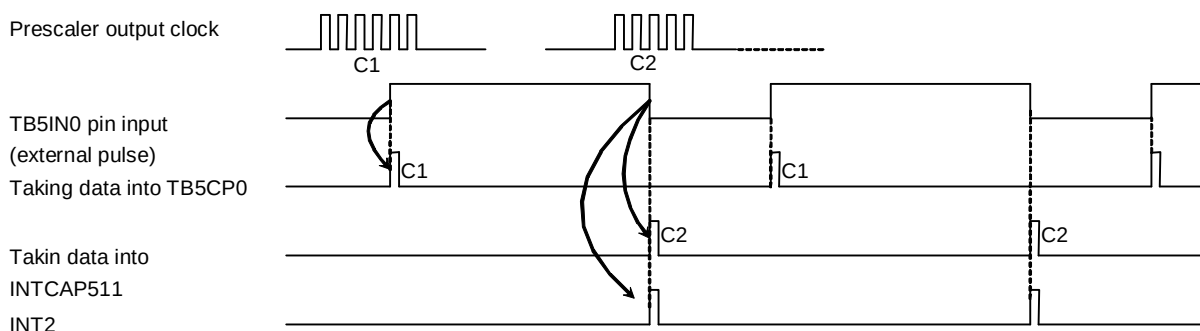


Fig. 8-8 Pulse Width Measurement

The “L” level width of an external pulse can also be measured. In such cases, the difference between C2 generated the first time and C1 generated the second time is initially obtained by performing the second stage of INTCAP51 interrupt processing as shown in “Fig. 8-9 Time Difference Measurement” and this difference is multiplied by the cycle of the prescaler output clock to obtain the “L” level width.

④ Time Difference Measurement

The up-counter (UC5) is made to count up by putting it in a free-running state using the prescaler output clock. The value of UC5 is taken into the capture register (TB5CP0) at the rising edge of the TB5IN0 pin input pulse. The NVIC must be programmed to generate INTCAP50 interrupt at this time.

The value of UC5 is taken into the capture register TB5CP1 at the rising edge of the TB5IN1 pin input pulse. The NVIC must be programmed to generate INTCAP51 interrupt at this time.

The time difference can be calculated by multiplying the difference between TB5CP1 and TB5CP0 by the clock cycle of an internal clock.

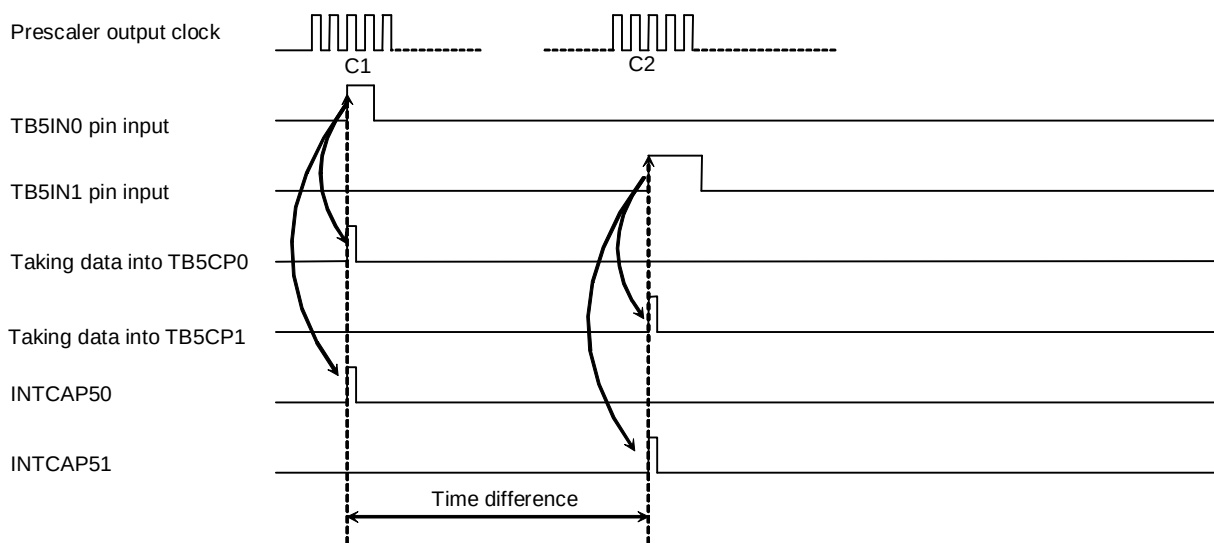
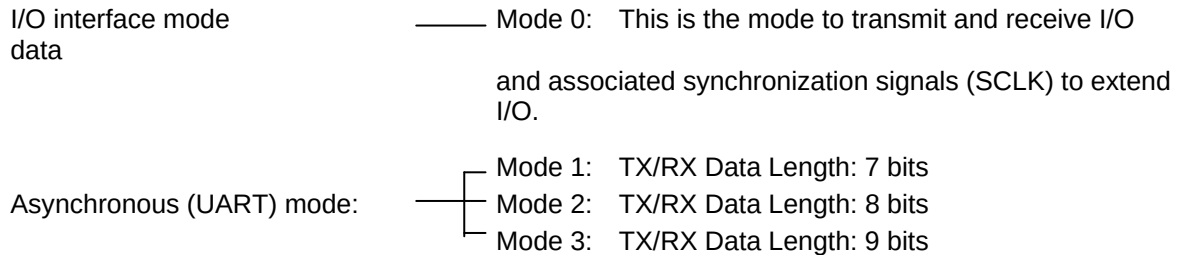


Fig. 8-9 Time Difference Measurement

9 Serial Channel (SIO)

9.1 Features

This device has three serial I/O channels: SIO0 to SIO2. Each channel operates in either the UART mode (asynchronous communication) or the I/O interface mode (synchronous communication) which is selected by the user.



In the above modes 1 and 2, parity bits can be added. The mode 3 has a wakeup function in which the master controller can start up slave controllers via the serial link (multi-controller system). Fig. 9-2 shows the block diagram of SIO0.

Each channel consists of a prescaler, a serial clock generation circuit, a receive buffer, its control circuit, a transmit buffer and its control circuit. Each channel functions independently.

As the SIOs 0 to 2 operate in the same way, only SIO0 is described here.

Table 9-1 Difference in the Specifications of SIO Modules

		Channel 0		Channel 1		Channel 2	
Pin name		TXD0 (PE0) RXD0 (PE1) $\overline{\text{CTS0}}$ / SCLK0 (PE2)		TXD1 (PE4) RXD1 (PE5) $\overline{\text{CTS1}}$ / SCLK1 (PE6)		TXD1 (PF0) RXD1 (PF1) $\overline{\text{CTS1}}$ / SCLK1 (PF2)	
Interrupt		INTRX0 INTTX0		INTRX1 INTTX1		INTRX2 INTTX2	
Register name (address)	Enable register	SC0EN	0x4002_0080	SC1EN	0x4002_00C0	SC2EN	0x4002_0100
	Transmit/ receive buffer register	SC0BUF	0x4002_0084	SC1BUF	0x4002_00C4	SC2BUF	0x4002_0104
	Control register	SC0CR	0x4002_0088	SC1CR	0x4002_00C8	SC2CR	0x4002_0108
	Mode control register 0	SC0MOD0	0x4002_008C	SC1MOD0	0x4002_00CC	SC2MOD0	0x4002_010C
	Baud rate generator control	SC0BRCR	0x4002_0090	SC1BRCR	0x4002_00D0	SC2BRCR	0x4002_0110
	Baud rate generator control 2	SC0BRADD	0x4002_0094	SC1BRADD	0x4002_00D4	SC2BRADD	0x4002_0114
	Mode control register 1	SC0MOD1	0x4002_0098	SC1MOD1	0x4002_00D8	SC2MOD1	0x4002_0118
	Mode control register 2	SC0MOD2	0x4002_009C	SC1MOD2	0x4002_00DC	SC2MOD2	0x4002_011C
	Receive FIFO configuration register	SC0RFC	0x4002_00A0	SC1RFC	0x4002_00E0	SC2RFC	0x4002_0120
	Transmit FIFO configuration register	SC0TFC	0x4002_00A4	SC1TFC	0x4002_00E4	SC2TFC	0x4002_0124
	Receive FIFO status register	SC0RST	0x4002_00A8	SC1RST	0x4002_00E8	SC2RST	0x4002_0128
	Transmit FIFO status register	SC0TST	0x4002_00AC	SC1TST	0x4002_00EC	SC2TST	0x4002_012C
	FIFO configuration register	SC0FCNF	0x4002_00B0	SC1FCNF	0x4002_00F0	SC2FCNF	0x4002_0130

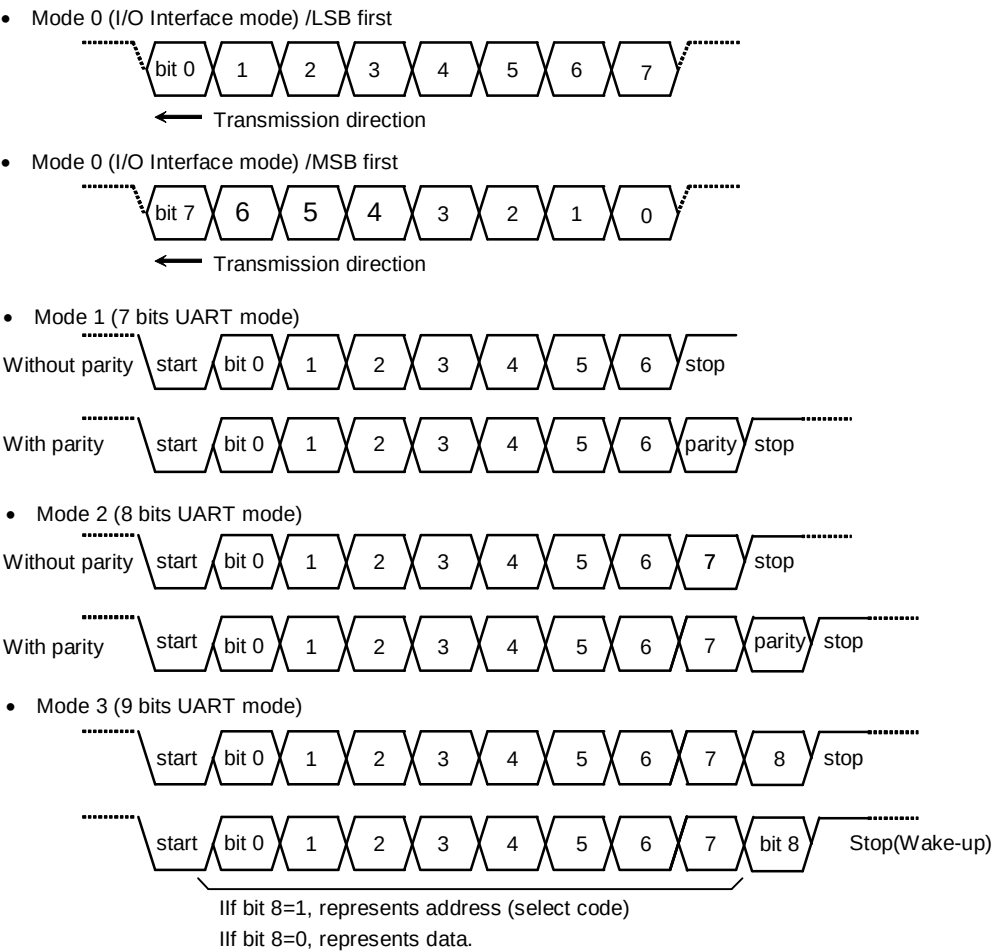


Fig. 9-1 Data Format

9.2 Block Diagram (Channel 0)

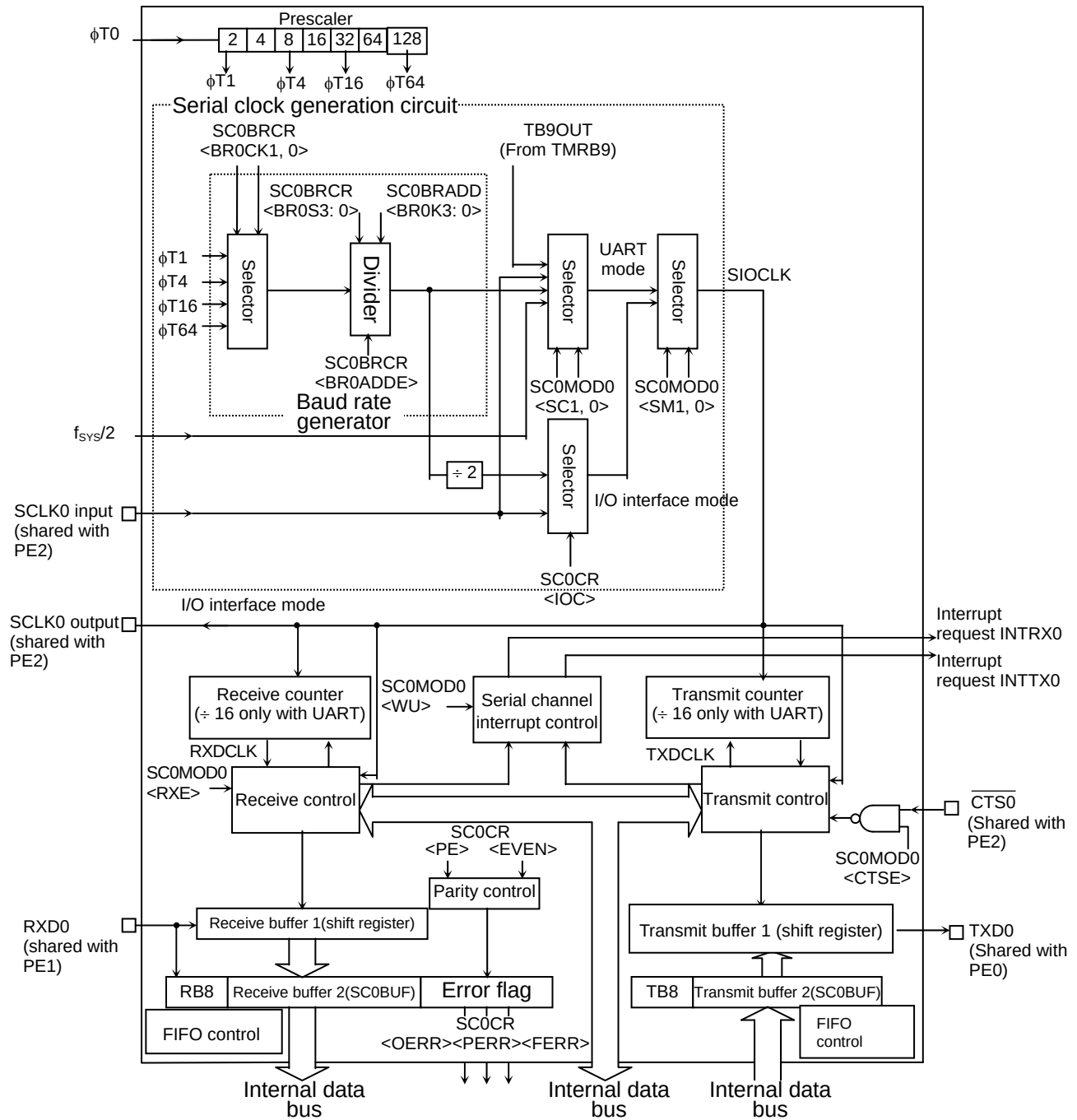


Fig. 9-2 SIO0 Block Diagram

9.3 Operation of Each Circuit (Channel 0)

9.3.1 Prescaler

The device includes a 7-bit prescaler to generate necessary clocks to drive SIO0. The input clock $\phi T0$ to the prescaler is selected by SYSCR1 of CG <PRCK2:0> to provide the frequency of either $f_{\text{periph}}/1$, $f_{\text{periph}}/2$, $f_{\text{periph}}/4$, $f_{\text{periph}}/8$, $f_{\text{periph}}/16$ or $f_{\text{periph}}/32$.

The clock frequency f_{periph} is either the clock “fgear,” to be selected by SYSCR1<FPSEL> of CG, or the clock “fc” before it is divided by the clock gear.

The prescaler becomes active only when the baud rate generator is selected for generating the serial transfer clock. Table 9-2 and 9-3 list the prescaler output clock resolution.

Table 9-2 Clock Resolution to the Baud Rate Generator

@ = 40MHz

Clear peripheral clock <FPSEL>	Clock gear value <GEAR2:0>	Prescaler clock selection <PRCK2:0>	Prescaler output clock resolution			
			$\phi T1$	$\phi T4$	$\phi T16$	$\phi T64$
0 (fgear)	000 (fc)	000(fperiph/1)	$fc/2^1(0.05\mu s)$	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$
		001(fperiph/2)	$fc/2^2(0.1\mu s)$	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$
		010(fperiph/4)	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$
		011(fperiph/8)	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$
		100(fperiph/16)	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$	$fc/2^{11}(51.2\mu s)$
		101(fperiph/32)	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$	$fc/2^{12}(102.4\mu s)$
	100(fc/2)	000(fperiph/1)	$fc/2^2(0.1\mu s)$	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$
		001(fperiph/2)	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$
		010(fperiph/4)	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$
		011(fperiph/8)	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$	$fc/2^{11}(51.2\mu s)$
		100(fperiph/16)	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$	$fc/2^{12}(102.4\mu s)$
		101(fperiph/32)	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$	$fc/2^{11}(51.2\mu s)$	$fc/2^{13}(204.8\mu s)$
	101(fc/4)	000(fperiph/1)	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$
		001(fperiph/2)	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$
		010(fperiph/4)	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$	$fc/2^{11}(51.2\mu s)$
		011(fperiph/8)	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$	$fc/2^{12}(102.4\mu s)$
		100(fperiph/16)	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$	$fc/2^{11}(51.2\mu s)$	$fc/2^{13}(204.8\mu s)$
		101(fperiph/32)	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$	$fc/2^{12}(102.4\mu s)$	$fc/2^{14}(409.6\mu s)$
	110(fc/8)	000(fperiph/1)	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$
		001(fperiph/2)	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$	$fc/2^{11}(51.2\mu s)$
		010(fperiph/4)	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$	$fc/2^{12}(102.4\mu s)$
		011(fperiph/8)	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$	$fc/2^{11}(51.2\mu s)$	$fc/2^{13}(204.8\mu s)$
		100(fperiph/16)	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$	$fc/2^{12}(102.4\mu s)$	$fc/2^{14}(409.6\mu s)$
		101(fperiph/32)	$fc/2^9(12.8\mu s)$	$fc/2^{11}(51.2\mu s)$	$fc/2^{13}(204.8\mu s)$	$fc/2^{15}(819.2\mu s)$
1 (fc)	000 (fc)	000(fperiph/1)	$fc/2^1(0.05\mu s)$	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$
		001(fperiph/2)	$fc/2^2(0.1\mu s)$	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$
		010(fperiph/4)	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$
		011(fperiph/8)	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$
		100(fperiph/16)	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$	$fc/2^{11}(51.2\mu s)$
		101(fperiph/32)	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$	$fc/2^{12}(102.4\mu s)$
	100(fc/2)	000(fperiph/1)	-	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$
		001(fperiph/2)	$fc/2^2(0.1\mu s)$	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$
		010(fperiph/4)	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$
		011(fperiph/8)	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$
		100(fperiph/16)	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$	$fc/2^{11}(51.2\mu s)$
		101(fperiph/32)	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$	$fc/2^{12}(102.4\mu s)$
	101(fc/4)	000(fperiph/1)	-	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$
		001(fperiph/2)	-	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$
		010(fperiph/4)	$fc/2^3(0.2\mu s)$	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$
		011(fperiph/8)	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$
		100(fperiph/16)	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$	$fc/2^{11}(51.2\mu s)$
		101(fperiph/32)	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$	$fc/2^{12}(102.4\mu s)$
	110(fc/8)	000(fperiph/1)	-	-	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$
		001(fperiph/2)	-	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$
		010(fperiph/4)	-	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$
		011(fperiph/8)	$fc/2^4(0.4\mu s)$	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$
		100(fperiph/16)	$fc/2^5(0.8\mu s)$	$fc/2^7(3.2\mu s)$	$fc/2^9(12.8\mu s)$	$fc/2^{11}(51.2\mu s)$
		101(fperiph/32)	$fc/2^6(1.6\mu s)$	$fc/2^8(6.4\mu s)$	$fc/2^{10}(25.6\mu s)$	$fc/2^{12}(102.4\mu s)$

- (Note 1) The prescaler output clock ϕT_n must be selected so that the relationship " $\phi T_n < f_{sys}$ " is satisfied (so that ϕT_n is slower than f_{sys}).
- (Note 2) Do not change the clock gear while SIO is operating.
- (Note 3) The horizontal lines in the above table indicate that the setting is prohibited.

The serial interface baud rate generator uses four different clocks, i.e., ϕT_1 , ϕT_4 , ϕT_{16} and ϕT_{64} , supplied from the prescaler output clock.

Table 9-3 Clock Resolution to the Baud Rate Generator

@ = 32MHz

Clear peripheral clock <FPSEL>	Clock gear value <GEAR2:0>	Prescaler clock selection <PRCK2:0>	Prescaler output clock resolution			
			$\phi T1$	$\phi T4$	$\phi T16$	$\phi T64$
0 (fgear)	000 (fc)	000(fperiph/1)	$fc/2^1(0.0625\mu s)$	$fc/2^3(0.25\mu s)$	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$
		001(fperiph/2)	$fc/2^2(0.125\mu s)$	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$
		010(fperiph/4)	$fc/2^3(0.25\mu s)$	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$
		011(fperiph/8)	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$
		100(fperiph/16)	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$	$fc/2^{11}(64.0\mu s)$
		101(fperiph/32)	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$	$fc/2^{12}(128.0\mu s)$
	100(fc/2)	000(fperiph/1)	$fc/2^1(0.0625\mu s)$	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$
		001(fperiph/2)	$fc/2^3(0.25\mu s)$	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$
		010(fperiph/4)	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$
		011(fperiph/8)	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$	$fc/2^{11}(64.0\mu s)$
		100(fperiph/16)	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$	$fc/2^{12}(128.0\mu s)$
		101(fperiph/32)	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$	$fc/2^{11}(64.0\mu s)$	$fc/2^{13}(256.0\mu s)$
	101(fc/4)	000(fperiph/1)	$fc/2^1(0.0625\mu s)$	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$
		001(fperiph/2)	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$
		010(fperiph/4)	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$	$fc/2^{11}(64.0\mu s)$
		011(fperiph/8)	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$	$fc/2^{12}(128.0\mu s)$
		100(fperiph/16)	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$	$fc/2^{11}(64.0\mu s)$	$fc/2^{13}(256.0\mu s)$
		101(fperiph/32)	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$	$fc/2^{12}(128.0\mu s)$	$fc/2^{14}(512.0\mu s)$
	110(fc/8)	000(fperiph/1)	$fc/2^1(0.0625\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$
		001(fperiph/2)	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$	$fc/2^{11}(64.0\mu s)$
		010(fperiph/4)	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$	$fc/2^{12}(128.0\mu s)$
		011(fperiph/8)	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$	$fc/2^{11}(64.0\mu s)$	$fc/2^{13}(256.0\mu s)$
		100(fperiph/16)	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$	$fc/2^{12}(128.0\mu s)$	$fc/2^{14}(512.0\mu s)$
		101(fperiph/32)	$fc/2^9(16.0\mu s)$	$fc/2^{11}(64.0\mu s)$	$fc/2^{13}(256.0\mu s)$	$fc/2^{15}(1024\mu s)$
1 (fc)	000 (fc)	000(fperiph/1)	$fc/2^1(0.0625\mu s)$	$fc/2^3(0.25\mu s)$	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$
		001(fperiph/2)	$fc/2^2(0.125\mu s)$	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$
		010(fperiph/4)	$fc/2^3(0.25\mu s)$	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$
		011(fperiph/8)	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$
		100(fperiph/16)	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$	$fc/2^{11}(64.0\mu s)$
		101(fperiph/32)	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$	$fc/2^{12}(128.0\mu s)$
	100(fc/2)	000(fperiph/1)	-	$fc/2^3(0.25\mu s)$	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$
		001(fperiph/2)	$fc/2^2(0.125\mu s)$	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$
		010(fperiph/4)	$fc/2^3(0.25\mu s)$	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$
		011(fperiph/8)	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$
		100(fperiph/16)	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$	$fc/2^{11}(64.0\mu s)$
		101(fperiph/32)	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$	$fc/2^{12}(128.0\mu s)$
	101(fc/4)	000(fperiph/1)	-	$fc/2^3(0.25\mu s)$	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$
		001(fperiph/2)	-	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$
		010(fperiph/4)	$fc/2^3(0.25\mu s)$	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$
		011(fperiph/8)	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$
		100(fperiph/16)	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$	$fc/2^{11}(64.0\mu s)$
		101(fperiph/32)	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$	$fc/2^{12}(128.0\mu s)$
	110(fc/8)	000(fperiph/1)	-	-	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$
		001(fperiph/2)	-	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$
		010(fperiph/4)	-	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$
		011(fperiph/8)	$fc/2^4(0.5\mu s)$	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$
		100(fperiph/16)	$fc/2^5(1.0\mu s)$	$fc/2^7(4.0\mu s)$	$fc/2^9(16.0\mu s)$	$fc/2^{11}(64.0\mu s)$
		101(fperiph/32)	$fc/2^6(2.0\mu s)$	$fc/2^8(8.0\mu s)$	$fc/2^{10}(32.0\mu s)$	$fc/2^{12}(128.0\mu s)$

- (Note 1) The prescaler output clock ϕT_n must be selected so that the relationship " $\phi T_n < f_{sys}$ " is satisfied (so that ϕT_n is slower than f_{sys}).
- (Note 2) Do not change the clock gear while SIO is operating.
- (Note 3) The horizontal lines in the above table indicate that the setting is prohibited.

The serial interface baud rate generator uses four different clocks, i.e., ϕT_1 , ϕT_4 , ϕT_{16} and ϕT_{64} , supplied from the prescaler output clock.

Baud Rate Generator

The baud rate generator generates transmit and receive clocks to determine the serial channel transfer rate.

The baud rate generator uses either the $\phi T1$, $\phi T4$, $\phi T16$ or $\phi T64$ clock supplied from the 7-bit prescaler. This input clock selection is made by setting the baud rate generator control register, SC0BRCCR <BR0CK1:0>.

The baud rate generator contains built-in dividers for divide by 1, $N + m/16$ ($N=2\sim 15$, $m=0\sim 15$), and 16. The division is performed according to the settings of the baud rate generator control registers SC0BRCCR <BR0ADDE> <BR0S3:0> and SC0BRADD <BR0K3:0> to determine the resulting transfer rate.

- UART mode

- 1) If SC0BRCCR <BR0ADDE> = 0,

The setting of SC0BRADD <BR0K3:0> is ignored and the counter is divided by N where N is the value set to SC0BRCCR <BR0S3:0>. ($N = 1$ to 16).

- 2) If SC0BRCCR <BR0ADDE> = 1,

The $N + (16 - K)/16$ division function is enabled and the division is made by using the values N (set in SC0BRCCR <BR0S3:0>) and K (set in SC0BRADD <BR0K3:0>). ($N = 2$ to 15, $K = 1$ to 15)

(Note) For the N values of 1 and 16, the above $N+(16-K)/16$ division function is inhibited. So, be sure to set SC0BRCCR<BR0ADDE> to "0."

- I/O interface mode

The $N + (16 - K)/16$ division function cannot be used in the I/O interface mode. Be sure to divide by N, by setting SC0BRCCR <BR0ADDE> to "0".

- Baud rate calculation to use the baud rate generator:

- 1) UART mode

$$\text{Baud rate} = \frac{\text{Baud rated generator input clock}}{\text{Frequency divided by the divide ratio}} / 16$$

The highest baud rate out of the baud rate generator is 1.0Mbps when $\phi T1$ is 16 MHz.

The fsys frequency, which is independent of the baud rate generator, can be used as the serial clock. In this case, the highest baud rate will be 2.0 Mbps when fsys is 32 MHz.

- 2) I/O interface mode

$$\text{Baud rate} = \frac{\text{Baud rated generator input clock}}{\text{Frequency divided by the divide ratio}} / 2$$

The highest baud rate will be generated when $\phi T1$ is 16 MHz. The divide ratio can be set to 1 if double buffer is used and the resulting output baud rate will be 8 Mbps. (If double buffering is not used, the highest baud rate will be 4.0 Mbps applying the divide ratio of "2").

- Example baud rate setting:

1) Division by an integer (divide by N):

Selecting $f_c = 39.321$ MHz for fperiph, setting $\phi T0$ to fperiph/16, using the baud rate generator input clock $\phi T1$, setting the divide ratio N (SC0BRCR<BR0S3:0>) = 4, and setting SC0BRCR<BR0ADDE> = "0," the resulting baud rate in the UART mode is calculated as follows:

* Clocking conditions System clock : High-speed (fc)
 High speed clock gear : x 1 (fc)
 Prescaler clock : $f_{\text{periph}}/16$ ($f_{\text{periph}} = f_{\text{sys}}$)

$$\text{Baud rate} = \frac{f_c/32}{4} / 16$$

$$= 39.321 \times 10^6 \div 32 \div 4 \div 16 \doteq 19200 \text{ (bps)}$$

(Note) The divide by $(N + (16-K)/16)$ function is inhibited and thus SC0BRADD <BR0K3:0> is ignored.

2) For divide by $N + (16-K)/16$ (only for UART mode):

Selecting $f_c = 9.6 \text{ MHz}$ for f_{periph} , setting $\phi T0$ to $f_{\text{periph}}/8$, using the baud rate generator input clock $\phi T1$, setting the divide ratio N ($SC0BRCR\langle BR0S3:0 \rangle = 7$), setting K ($SC0BRADD\langle BR0K3:0 \rangle = 3$), and selecting $SC0BRCR\langle BR0ADDE \rangle = 1$, the resulting baud rate is calculated as follows:

* Clocking conditions

System clock	:	High-speed (fc)
High-speed clock gear	:	x 1 (fc)
Prescaler clock	:	$f_{\text{periph}}/4$ ($f_{\text{periph}} = f_{\text{sys}}$)

$$\begin{aligned}\text{Baud rate} &= \frac{f_c/16}{7 + \frac{(16-3)}{16}} / 16 \\ &= 9.6 \times 10^6 \div 16 \div \left(7 + \frac{13}{16}\right) \div 16 = 4800 \text{ (bps)}\end{aligned}$$

Also, an external clock input may be used as the serial clock. The resulting baud rate calculation is shown below:

- Baud rate calculation for an external clock input:

- 1) UART mode

Baud Rate = external clock input / 16

In this, the period of the external clock input must be equal to or greater than $2/f_{sys}$.

If $f_{sys} = 32$ MHz, the highest baud rate will be $32 \div 2 \div 16 = 1.0$ (Mbps).

- 2) I/O interface mode

Baud Rate = external clock input

When double buffering is used, it is necessary to satisfy the following relationship:

External clock input period $> 6/f_{sys}$

Therefore, when $f_{sys} = 32$ MHz, the baud rate must be set to a rate lower than $32 \div 6 = 5.33$ (Mbps).

When double buffering is not used, it is necessary to satisfy the following relationship:

External clock input period $> 8/f_{sys}$

Therefore, when $f_{sys} = 32$ MHz, the baud rate must be set to a rate lower than $32 \div 8 = 4.0$ (Mbps).

The baud rate examples for the UART mode are shown in Table 9-4 and Table 9-5.

Table 9-4 Selection of UART Baud Rate

(Using the baud rate generator with SC0BRCR <BR0ADDE> = 0)

Unit: (kbps)

fc [MHz]	Input clock Divide ratio N (Set to SC0BRCR <BR0S3 : 0>)	$\phi T1$ (fc/4)	$\phi T4$ (fc/16)	$\phi T16$ (fc/64)	$\phi T64$ (fc/256)
9.830400	2	76.800	19.200	4.800	1.200
↑	4	38.400	9.600	2.400	0.600
↑	8	19.200	4.800	1.200	0.300
↑	0	9.600	2.400	0.600	0.150

(Note) This table shows the case where the system clock is set to fc, the clock gear is set to fc/1, and the prescaler clock is set to $f_{\text{periph}}/2$.

Table 9-5 Selection of UART Baud Rate

(The TMRB9 timer output (internal TB9OUT) is used with the timer input clock set to $\phi T0$.)

Unit: (Kbbs)

TB0RG \ fc	32 MHz	9.8304 MHz	8 MHz
1H	250	76.8	62.5
2H	125	38.4	31.25
3H		25.6	
4H	62.5	19.2	15.625
5H	50	15.36	12.5
6H		12.8	
8H	31.25	9.6	
AH	25	7.68	6.25
10H	15.625	4.8	
14H	12.5	3.84	3.125

Baud rate calculation to use the TMRB9 timer:

$$\text{Transfer rate} = \frac{\text{Clock frequency selected by SYSCR0<PRCK1:0>}}{\text{TB0RG} \times 2 \times 16}$$

↑
(When input clock to the timer TMRB9 is $\phi T0$)

(Note 1) In the I/O interface mode, the TMRB9 timer output signal cannot be used internally as the transfer clock.

(Note 2) This table shows the case where the system clock is set to fc, the clock gear is set to fc, and the prescaler clock is set to $f_{\text{periph}}/4$.

9.3.2 Serial Clock Generation Circuit

This circuit generates basic transmit and receive clocks.

- I/O interface mode

In the SCLK output mode with the SC0CR <IOC> serial control register set to “0,” the output of the previously mentioned baud rate generator is divided by 2 to generate the basic clock.

In the SCLK input mode with SC0CR <IOC> set to “1,” rising and falling edges are detected according to the SC0CR <SCLKS> setting to generate the basic clock.

- Asynchronous (UART) mode :

According to the settings of the serial control mode register SC0MOD0 <SC1:0>, either the clock from the baud rate register, the system clock (f_{sys}), the internal output signal of the TMRB9 timer, or the external clock (SCLKO pin) is selected to generate the basic clock, SIOCLK.

9.3.3 Receive Counter

The receive counter is a 4-bit binary counter used in the asynchronous (UART) mode and is up-counted by SIOCLK. Sixteen SIOCLK clock pulses are used in receiving a single data bit while the data symbol is sampled at the seventh, eighth, and ninth pulses. From these three samples, majority logic is applied to decide the received data.

9.3.4 Receive Control Unit

- I/O interface mode:

In the SCLK output mode with SC0CR <IOC> set to “0,” the RXD0 pin is sampled on the rising edge of the shift clock output to the SCLK0 pin.

In the SCLK input mode with SC0CR <IOC> set to “1,” the serial receive data RXD0 pin is sampled on the rising or falling edge of SCLK input depending on the SC0CR <SCLKS> setting.

- Asynchronous (UART) mode:

The receive control unit has a start bit detection circuit, which is used to initiate receive operation when a normal start bit is detected.

9.3.5 Receive Buffer

The receive buffer is of a dual structure to prevent overrun errors. The first receive buffer (a shift register) stores the received data bit-by-bit. When a complete set of bits have been stored, they are moved to the second receive buffer (SC0BUF). At the same time, the receive buffer full flag (SC0MOD2 “RBFL”) is set to “1” to indicate that valid data is stored in the second receive buffer. However, if the receive FIFO is set enabled, the receive data is moved to the receive FIFO and this flag is immediately cleared.

If the receive FIFO has been disabled (SC0FCNF <CNFG> = 0 and SC0MOD1<FDPX1:0> = 01), the INTRX0 interrupt is generated at the same time. If the receive FIFO has been enabled (SC0FCNF <CNFG> = 1 and SC0MOD1<FDPX1:0> = 01), an interrupt will be generated according to the SC0RFC <RIL1:0> setting.

The CPU will read the data from either the second receive buffer (SC0BUF) or from the receive FIFO (the address is the same as that of the receive buffer). If the receive FIFO has not been enabled, the receive buffer full flag <RBFL> is cleared to "0" by the read operation. The next data received can be stored in the first receive buffer even if the CPU has not read the previous data from the second receive buffer (SC0BUF) or the receive FIFO.

If SCLK is set to generate clock output in the I/O interface mode, the double buffer control bit SC0MOD2 <WBUF> can be programmed to enable or disable the operation of the second receive buffer (SC0BUF).

By disabling the second receive buffer (i.e., the double buffer function) and also disabling the receive FIFO (SC0FCNF <CNFG> = 0 and <FDPX1:0> = 01), handshaking with the other side of communication can be enabled and the SCLK output stops each time one frame of data is transferred. In this setting, the CPU reads data from the first receive buffer. By the read operation of CPU, the SCLK output resumes.

If the second receive buffer (i.e., double buffering) is enabled but the receive FIFO is not enabled, the SCLK output is stopped when the first receive data is moved from the first receive buffer to the second receive buffer and the next data is stored in the first buffer filling both buffers with valid data. When the second receive buffer is read, the data of the first receive buffer is moved to the second receive buffer and the SCLK output is resumed upon generation of the receive interrupt INTRX. Therefore, no buffer overrun error will be caused in the I/O interface SCLK output mode regardless of the setting of the double buffer control bit SC0MOD2 <WBUF>.

If the second receive buffer (double buffering) is enabled and the receive FIFO is also enabled (SC0FCNF <CNFG> = 1 and <FDPX1:0> = 01/11), the SCLK output will be stopped when the receive FIFO is full (according to the setting of SC0FCNF <RFST>) and both the first and second receive buffers contain valid data. Also in this case, if SC0FCNF <RXTXCNT> has been set to "1," the receive control bit RXE will be automatically cleared upon suspension of the SCLK output. If it is set to "0," automatic clearing will not be performed.

(Note) In this mode, the SC0CR <OEER> flag is insignificant and the operation is undefined. Therefore, before switching from the SCLK output mode to another mode, the SC0CR register must be read to initialize this flag.

In other operating modes, the operation of the second receive buffer is always valid, thus improving the performance of continuous data transfer. If the receive FIFO is not enabled, an overrun error occurs when the data in the second receive buffer (SC0BUF) has not been read before the first receive buffer is full with the next receive data. If an overrun error occurs, data in the first receive buffer will be lost while data in the second receive buffer and the contents of SC0CR <RB8> remain intact. If the receive FIFO is enabled, the FIFO must be read before the FIFO is full and the second receive buffer is written by the next data through the first buffer. Otherwise, an overrun error will be generated and the receive FIFO overrun error flag will be set. Even in this case, the data already in the receive FIFO remains intact.

The parity bit to be added in the 8-bit UART mode as well as the most significant bit in the 9-bit UART mode will be stored in SC0CR <RB8>.

In the 9-bit UART mode, the slave controller can be operated in the wake-up mode by setting the wake-up function SC0MOD0 <WU> to "1." In this case, the interrupt INTRX0 will be generated only when SC0CR <RB8> is set to "1."

9.3.6 Receive FIFO Buffer

In addition to the double buffer function already described, data may be stored using the receive FIFO buffer. By setting <CNFG> of the SC0FCNF register and <FDPX1:0> of the SC0MOD1 register, the 4-byte receive buffer can be enabled. Also, in the UART mode or I/O interface mode, data may be stored up to a predefined fill level. When the receive FIFO buffer is to be used, be sure to enable the double buffer function.

If data with parity bit is to be received in the UART mode, parity check must be performed each time a data frame is received.

9.3.7 Receive FIFO Operation

① I/O interface mode with SCLK output:

The following example describes the case a 4-byte data stream is received in the half duplex mode:

SC0RFC<7:6>=01: Clears receive FIFO and sets the condition of interrupt generation.

SC0RFC<1:0>=00: Sets the interrupt to be generated at fill level 4.

SC0FCNF <4:0>=10111: Automatically inhibits continued reception after reaching the fill level. The number of bytes to be used in the receive FIFO is the same as the interrupt generation fill level.

In this condition, 4-byte data reception may be initiated by setting the half duplex transmission mode and writing "1" to the RXE bit. After receiving 4 bytes, the RXE bit is automatically cleared and the receive operation is stopped (SCLK is stopped).

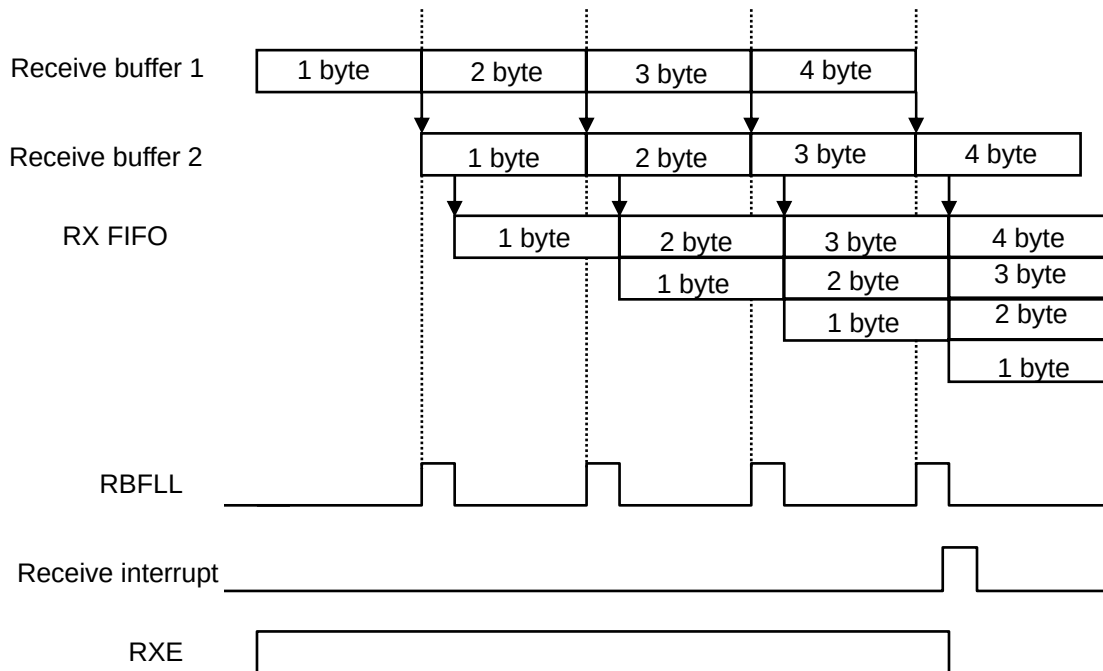


Fig. 9-3 Receive FIFO Operation

② I/O interface mode with SCLK input:

The following example describes the case a 4-byte data stream is received:

SC0RFC <7:6> = 10: Clears receive FIFO and sets the condition of interrupt generation

SC0RFC <1:0> = 00: Sets the interrupt to be generated at fill level 4.

SC0FCNF <1:0> = 10101: Automatically allows continued reception after reaching the fill level.
The number of bytes to be used in the receive FIFO is the maximum allowable number.

In this condition, 4-byte data reception may be initiated by setting the half duplex transmission mode and writing "1" to the RXE bit. After receiving 4 bytes, receive FIFO interrupt is generated. This setting enables the next data reception as well. The next 4 bytes can be received before all the data is read from FIFO.

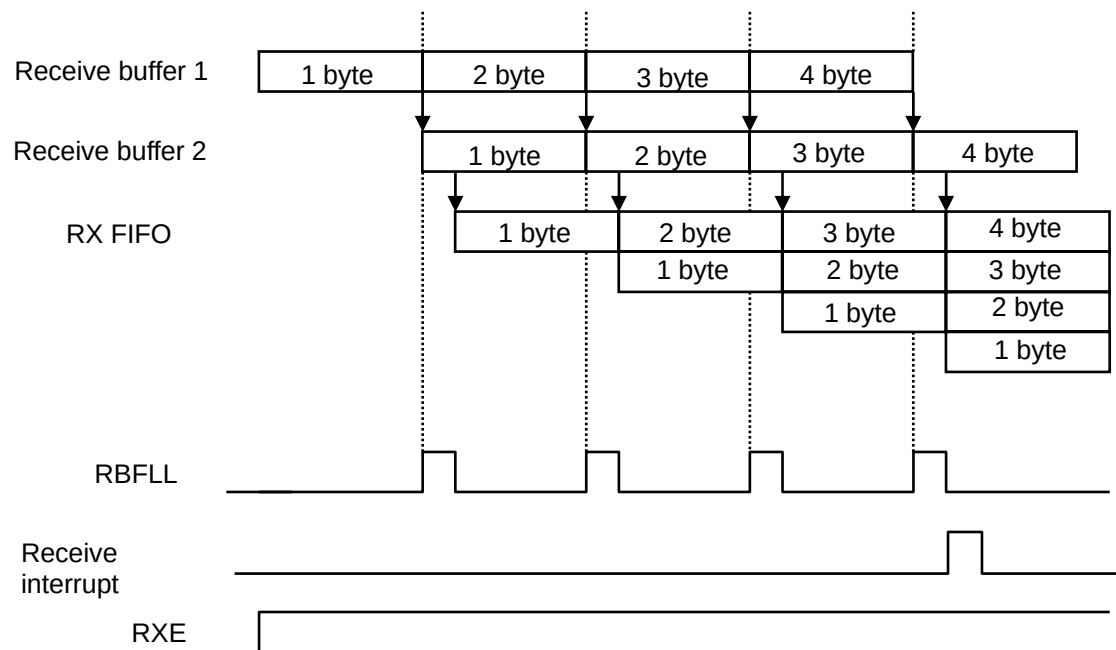


Fig. 9-4 Receive FIFO Operation

9.3.8 Transmit Counter

The transmit counter is a 4-bit binary counter used in the asynchronous communication (UART) mode. It is counted by SIOCLK as in the case of the receive counter and generates a transmit clock (TXDCLK) on every 16th clock pulse.

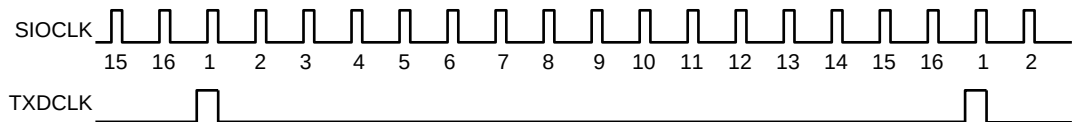


Fig. 9-5 Transmit Clock Generation

9.3.9 Transmit Control Unit

- I/O interface mode:

In the SCLK output mode with SC0CR <IOC> set to "0," each bit of data in the transmit buffer is output to the TXD0 pin on the rising edge of the shift clock output from the SCLK0 pin.

In the SCLK input mode with SC0CR <IOC> set to "1," each bit of data in the transmit buffer is output to the TXD0 pin on the rising or falling edge of the input SCLK signal according to the SC0CR <SCLKS> setting.

- Asynchronous (UART) mode:

When the CPU writes data to the transmit buffer, data transmission is initiated on the rising edge of the next TXDCLK and the transmit shift clock (TXDSFT) is also generated.

- Handshake function

The $\overline{\text{CTS}}$ pin enables frame by frame data transmission so that overrun errors can be prevented. This function can be enabled or disabled by SC0MOD0 <CTSE>.

When the $\overline{\text{CTS0}}$ pin is set to the “H” level, the current data transmission can be completed but the next data transmission is suspended until the $\overline{\text{CTS0}}$ pin returns to the “L” level. However in this case, the INTTX0 interrupt is generated, the next transmit data is requested to the CPU, data is written to the transmit buffer, and it waits until it is ready to transmit data.

Although no $\overline{\text{RTS}}$ pin is provided, a handshake control function can be easily implemented by assigning a port for the $\overline{\text{RTS}}$ function. By setting the port to “H” level upon completion of data reception (in the receive interrupt routine), the transmit side can be requested to suspend data transmission.

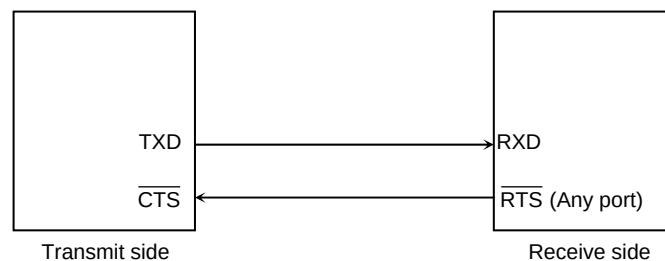


Fig. 9-6 Handshake Function

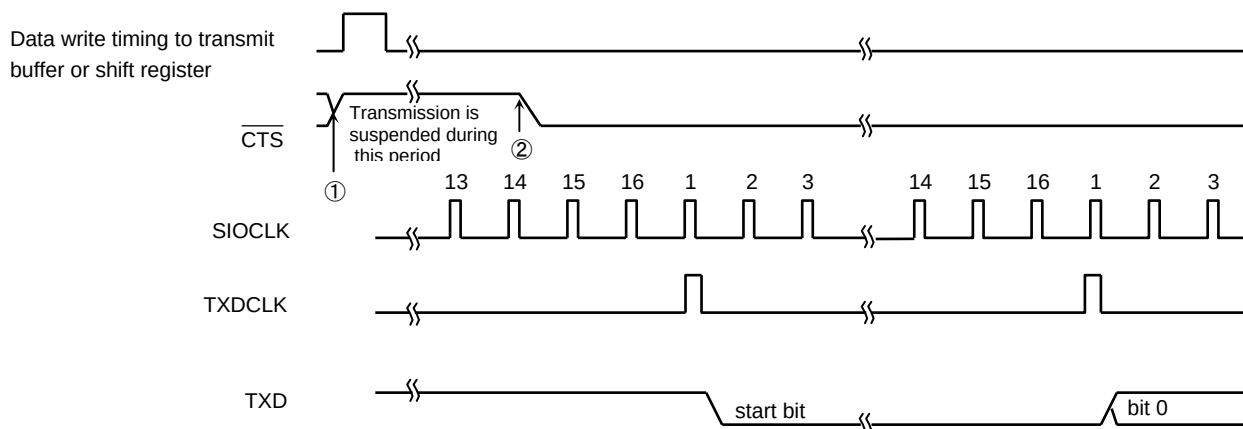


Fig. 9-7 $\overline{\text{CTS}}$ (Clear to Transmit) Signal Timing

(Note 1) If the $\overline{\text{CTS}}$ signal is set to “H” during transmission, the next data transmission is suspended after the current transmission is completed.

(Note 2) Data transmission starts on the first falling edge of the TXDCLK clock after $\overline{\text{CTS}}$ is set to “L.”

Transmit Buffer

The transmit buffer (SC0BUF) is in a dual structure. The double buffering function may be enabled or disabled by setting the double buffer control bit <WBUF> in serial mode control register 2 (SC0MOD2). If double buffering is enabled, data written to Transmit Buffer 2 (SC0BUF) is moved to Transmit Buffer 1 (shift register).

If the transmit FIFO has been disabled (SC0FCNF <CNFG> = 0 or 1 and SC0MOD1<FDPX1:0> = 01), the INTTX0 interrupt is generated at the same time and the transmit buffer empty flag <TBEMP> of SC0MOD2 is set to "1." This flag indicates that Transmit Buffer 2 is now empty and that the next transmit data can be written. When the next data is written to Transmit Buffer 2, the <TBEMP> flag is cleared to "0."

If the transmit FIFO has been enabled (SC0FCNF <CNFG> = 1 and SC0MOD1<FDPX1:0> = 10/11), any data in the transmit FIFO is moved to the Transmit Buffer 2 and <TBEMP> flag is immediately cleared to "0." The CPU writes data to Transmit Buffer 2 or to the transmit FIFO.

If the transmit FIFO is disabled in the I/O interface SCLK input mode and if no data is set in Transmit Buffer 2 before the next frame clock input, which occurs upon completion of data transmission from Transmit Buffer 1, an under-run error occurs and a serial control register (SC0CR) <PERR> parity/under-run flag is set.

If the transmit FIFO is enabled in the I/O interface SCLK input mode, when data transmission from Transmit Buffer 1 is completed, the Transmit Buffer 2 data is moved to Transmit Buffer 1 and any data in transmit FIFO is moved to Transmit Buffer 2 at the same time.

If the transmit FIFO is disabled in the I/O interface SCLK output mode, when data in Transmit Buffer 2 is moved to Transmit Buffer 1 and the data transmission is completed, the SCLK output stops. So, no under-run errors can be generated.

If the transmit FIFO is enabled in the I/O interface SCLK output mode, the SCLK output stops upon completion of data transmission from Transmit Buffer 1 if there is no valid data in the transmit FIFO.

Note) In the I/O interface SCLK output mode, the SC0CR <PEER> flag is insignificant. In this case, the operation is undefined. Therefore, to switch from the SCLK output mode to another mode, SC0CR must be read in advance to initialize the flag.

If double buffering is disabled, the CPU writes data only to Transmit Buffer 1 and the transmit interrupt INTTX0 is generated upon completion of data transmission.

If handshaking with the other side is necessary, set the double buffer control bit <WBUF> to "0" (disable) to disable Transmit Buffer 2; any setting for the transmit FIFO should not be performed.

9.3.10 Transmit FIFO Buffer

In addition to the double buffer function already described, data may be stored using the transmit FIFO buffer. By setting <CNFG> of the SC0FCNF register and <FDPX1:0> of the SC0MOD1 register, the 4-byte transmit buffer can be enabled. In the UART mode or I/O interface mode, up to 4 bytes of data may be stored.

If data is to be transmitted with a parity bit in the UART mode, parity check must be performed on the receive side each time a data frame is received.

9.3.11 Transmit FIFO Operation

- ① I/O interface mode with SCLK output (normal mode):

The following example describes the case a 4-byte data stream is transmitted:

SC0TFC <7:6> = 01: Clears transmit FIFO and sets the condition of interrupt generation

SC0TFC <1:0> = 00: Sets the interrupt to be generated at fill level 0.

SC0FCNF <4:0> = 01011: Inhibits continued transmission after reaching the fill level.

In this condition, data transmission can be initiated by setting the transfer mode to half duplex, writing 4 bytes of data to the transmit FIFO, and setting the <TXE> bit to "1." When the last transmit data is moved to the transmit buffer, the transmit FIFO interrupt is generated. When transmission of the last data is completed, the clock is stopped and the transmission sequence is terminated.

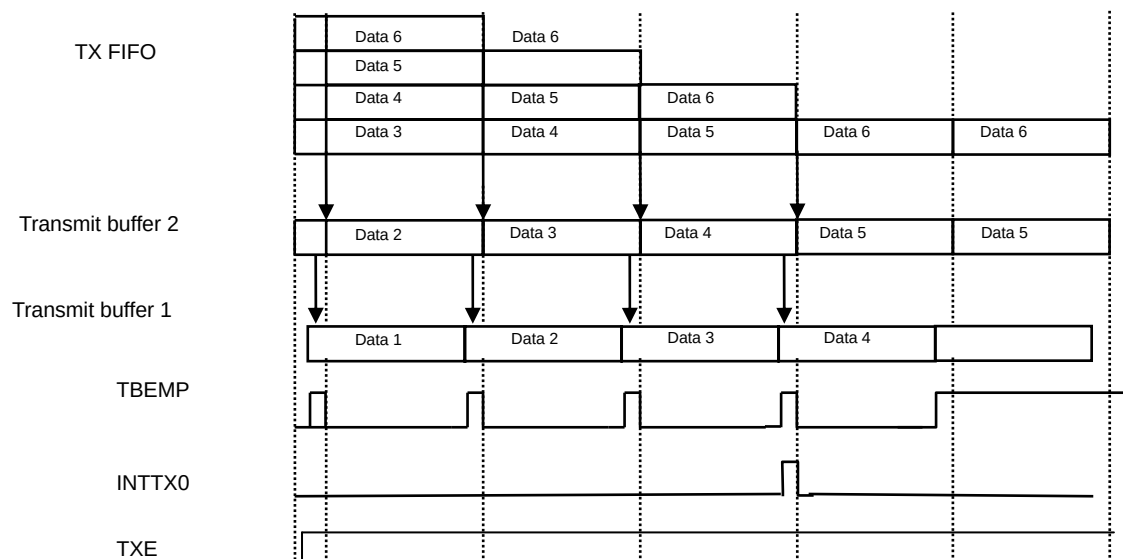


Fig. 9-8 Transmit FIFO Operation

② I/O interface mode with SCLK input (normal mode):

The following example describes the case a 4-byte data stream is transmitted:

SC0TFC <1:0> = 01: Clears the transmit FIFO and sets the condition of interrupt generation.

SC0TFC <7:6> = 000000: Sets the interrupt to be generated at fill level 0.

SC0FCNF <4:0> = 01001: Allows continued transmission after reaching the fill level.

In this condition, data transmission can be initiated along with the input clock by setting the transfer mode to half duplex, writing 4 bytes of data to the transmit FIFO, and setting the <TXE> bit to "1." When the last transmit data is moved to the transmit buffer, the transmit FIFO interrupt is generated.

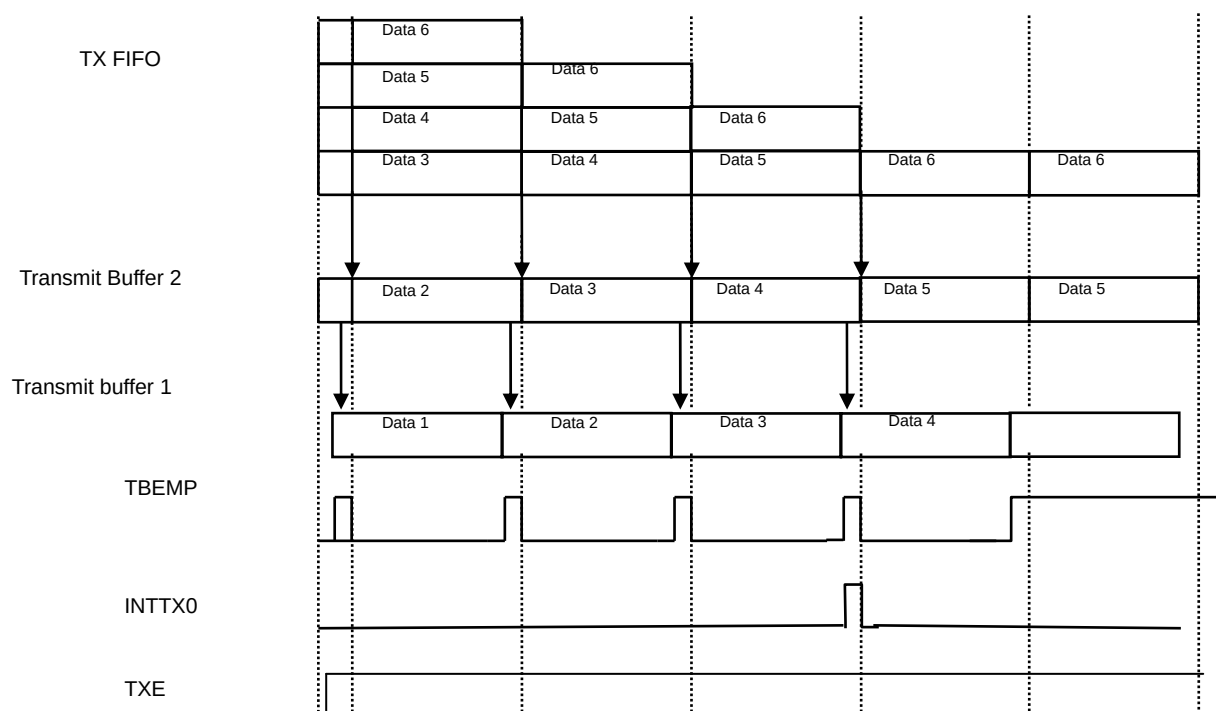


Fig. 9-9 Transmit FIFO Operation

9.3.12 Parity Control Circuit

If the parity addition bit <PE> of the serial control register SC0CR is set to “1,” data is sent with the parity bit. Note that the parity bit may be used only in the 7- or 8-bit UART mode. The <EVEN> bit of SC0CR selects either even or odd parity.

Upon data transmission, the parity control circuit automatically generates the parity with the data written to the transmit buffer (SC0BUF). After data transmission is complete, the parity bit will be stored in SC0BUF bit 7 <TB7> in the 7-bit UART mode and in bit 7 <TB8> in the serial mode control register SC0MOD in the 8-bit UART mode. The <PE> and <EVEN> settings must be completed before data is written to the transmit buffer.

Upon data reception, the parity bit for the received data is automatically generated while the data is shifted to receive buffer 1 and moved to receive buffer 2 (SC0BUF). In the 7-bit UART mode, the parity generated is compared with the parity stored in SC0BUF <RB7>, while in the 8-bit UART mode, it is compared with the bit 7 <RB8> of the SC0CR register. If there is any difference, a parity error occurs and the <PERR> flag of the SC0CR register is set.

In the I/O interface mode, the SC0CR <PERR> flag functions as an under-run error flag, not as a parity flag.

9.3.13 Error Flag

Three error flags are provided to improve the reliability of received data.

1. Overrun error <OERR>: Bit 4 of the serial control register SC0CR

In both UART and I/O interface modes, this bit is set to “1” when an error is generated by completing the reception of the next frame receive data before the receive buffer has been read. If the receive FIFO is enabled, the received data is automatically moved to the receive FIFO and no overrun error will be generated until the receive FIFO is full (or until the usable bytes are fully occupied). This flag is set to “0” when it is read. In the I/O interface SCLK output mode, no overrun error is generated and therefore, this flag is inoperative and the operation is undefined.

2. Parity error/under-run error <PERR>: Bit 3 of the SC0CR register

In the UART mode, this bit is set to “1” when a parity error is generated. A parity error is generated when the parity generated from the received data is different from the parity received. This flag is set to “0” when it is read.

In the I/O interface mode, this bit indicates an under-run error. When the double buffer control bit <WBUF> of the serial mode control register SC0MOD2 is set to “1” in the SCLK input mode, if no data is set to the transmit double buffer before the next data transfer clock after completing the transmission from the transmit shift register, this error flag is set to “1” indicating an under-run error. If the transmit FIFO is enabled, any data content in the transmit FIFO will be moved to the buffer. When the transmit FIFO and the double buffer are both empty, an under-run error will be generated. Because no under-run errors can be generated in the SCLK output mode, this flag is inoperative and the operation is undefined. If Transmit Buffer 2 is disabled, the under-run flag <PERR> will not be set. This flag is set to “0” when it is read.

3. Framing error <FERR>: Bit 2 of the SC0CR register

In the UART mode, this bit is set to “1” when a framing error is generated. This flag is set to “0” when it is read. A framing error is generated if the corresponding stop bit is determined to be “0” by sampling the bit at around the center. Regardless of the <SBLN> (stop bit length) setting of the serial mode control register 2, SC0MOD2, the stop bit status is determined by only 1 bit on the receive side.

Operation mode	Error flag	Function
UART	OERR	Overrun error flag
	PERR	Parity error flag
	FERR	Framing error flag
I/O Interface (SCLK input)	OERR	Overrun error flag
	PERR	Underrun error flag (WBUF = 1)
		Fixed to 0 (WBUF = 0)
	FERR	Fixed to 0
I/O Interface (SCLK output)	OERR	Operation undefined
	PERR	Operation undefined
	FERR	Fixed to 0

9.3.14 Direction of Data Transfer

In the I/O interface mode, the direction of data transfer can be switched between “MSB first” and “LSB first” by the data transfer direction setting bit <DRCHG> of the SC0MOD2 serial mode control register 2. Don't switch the direction when data is being transferred.

9.3.15 Stop Bit Length

In the UART transmission mode, the stop bit length can be set to either 1 or 2 bits by bit 4 <SBLN> of the SC0MOD2 register.

9.3.16 Status Flag

If the double buffer function is enabled (SC0MOD2 <WBUF> = “1”), the bit 6 flag <RBFLL> of the SC0MOD2 register indicates the condition of receive buffer full. When one frame of data has been received and transferred from buffer 1 to buffer 2, this bit is set to “1” to show that buffer 2 is full (data is stored in buffer 2). When the receive buffer is read by CPU/DMAC, it is cleared to “0.” If <WBUF> is set to “0,” this bit is insignificant and must not be used as a status flag. When double buffering is enabled (SC0MOD2 <WBUF> = “1”), the bit 7 flag <TBEMP> of the SC0MOD2 register indicates that Transmit Buffer 2 is empty. When data is moved from Transmit Buffer 2 to Transmit Buffer 1 (shift register), this bit is set to “1” indicating that Transmit Buffer 2 is now empty. When data is set to the transmit buffer by CPU/DMAC, the bit is cleared to “0.” If <WBUF> is set to “0,” this bit is insignificant and must not be used as a status flag.

9.3.17 Configurations of Transmit/Receive Buffer

		<WBUF> = 0	<WBUF> = 1
UART	Transmit buffer	Single	Double
	Receive buffer	Double	Double
I/O Interface (SCLK input)	Transmit buffer	Single	Double
	Receive buffer	Double	Double
I/O Interface (SCLK output)	Transmit buffer	Single	Double
	Receive buffer	Single	Double

9.3.18 Software reset

Software reset is generated by writing the bits 1 and 0 of SC0MOD2 <SWRST1:0> as “10” followed by “01”. As a result, SC0MOD0<RXE>, SC0MOD1<TXE>, SC0MOD2<TBEMP>,<RBFL>,<TXRUN> of mode registers and SC0CR<OERR>, <PERR>, <FERR> of control registers and internal circuit is initialized. Other states are maintained.

9.3.19 Signal Generation Timing

① UART Mode:

Receive Side

Mode	9-bit	8-bit with parity	8-bit, 7-bit, and 7-bit with parity
Interrupt generation timing	Around the center of the 1st stop bit	Around the center of the 1st stop bit	Around the center of the 1st stop bit
Framing error generation timing	Around the center of the stop bit	Around the center of the stop bit	Around the center of the stop bit
Parity error generation timing	—	Around the center of the last (parity) bit	Around the center of the last (parity) bit
Overrun error generation timing	Around the center of the stop bit	Around the center of the stop bit	Around the center of the stop bit

Transmit Side

Mode	9-bit	8-bit with parity	8-bit, 7-bit, and 7-bit with parity
Interrupt generation timing (<WBUF> = 0)	Just before the stop bit is sent	Just before the stop bit is sent	Just before the stop bit is sent
Interrupt generation timing (<WBUF> = 1)	Immediately after data is moved to transmit buffer 1 (just before start bit transmission)	Immediately after data is moved to transmit buffer 1 (just before start bit transmission).	Immediately after data is moved to transmit buffer 1 (just before start bit transmission)

② I/O interface mode:

Receive Side

Interrupt generation timing (<WBUF> = 0)	SCLK output mode	Immediately after the rising edge of the last SCLK
	SCLK input mode	Immediately after the rising or falling edge of the last SCLK (for rising or falling edge mode, respectively).
Interrupt generation timing (<WBUF> = 1)	SCLK output mode	Immediately after the rising edge of the last SCLK (just after data transfer to receive buffer 2) or just after receive buffer 2 is read.
	SCLK input mode	Immediately after the rising edge or falling edge of the last SCLK (right after data is moved to receive buffer 2).
Overrun error generation timing	SCLK input mode	Immediately after the rising or falling edge of the last SCLK (for rising or falling edge mode, respectively)

Transmit Side

Interrupt generation timing (<WBUF> = 0)	SCLK output mode	Immediately after the rising edge of the last SCLK
	SCLK input mode	Immediately after the rising or falling edge of the last SCLK (for rising or falling edge mode, respectively)
Interrupt generation timing (<WBUF> = 1)	SCLK output mode	Immediately after the rising edge of the last SCLK or just after data is moved to Transmit Buffer 1
	SCLK input mode	Immediately after the rising or falling edge of the last SCLK or just after data is moved to Transmit Buffer 1
Under-run error generation timing	SCLK input mode	Immediately after the falling or rising edge of the next SCLK

(Note 1) Do not modify any control register when data is being sent or received (in a state ready to transmit or receive).

(Note 2) Do not stop the receive operation (by setting SC0MOD0 <RXE> = "0") when data is being received.

(Note 3) Do not stop the transmit operation (by setting SC0MOD1 <TXE> = "0") when data is being transmitted.

9.4 Register Description (Only for Channel 0)

The channel 0 registers are described here. Each register for all the channels operates in the same way.

9.4.1 Enable register

	7	6	5	4	3	2	1	0
SC0EN								SIOE
bit Symbol								SIOE
Read/Write	R							R/W
After reset	0							0
Function	"0" is read.							SIO operation 0: disabled 1: enabled

<SIOE>: Specified the SIO operation.
 To use the SIO, enable the SIO operation.
 When the operation is disabled, no clock is supplied to the other registers in the SIO module.
 This can reduce the power consumption.
 If the SIO operation is executed and then disabled, the settings will be maintained in each register.

9.4.2 Buffer register

SC0BUF works as a transmit buffer for WR operation and as a receive buffer for RD operation.

	7	6	5	4	3	2	1	0
SC0BUF	TB7/RB7	TB6/RB6	TB5/RB5	TB4/RB4	TB3/RB3	TB2/RB2	TB1/RB1	TB0/RB0
bit Symbol	TB7/RB7	TB6/RB6	TB5/RB5	TB4/RB4	TB3/RB3	TB2/RB2	TB1/RB1	TB0/RB0
Read/Write	R/W							
After reset	0	0	0	0	0	0	0	0
Function	TB7~0 : Transmit buffer/FIFO RB7~0 : Receive buffer/FIFO							

<TB7:0> Transmit buffer (at WR operation).

<RB7:0> Receive buffer (at RD operation).

9.4.3 Control register

SC0CR		7	6	5	4	3	2	1	0
	bit Symbol	RB8	EVEN	PE	OERR	PERR	FERR	SCLKS	IOC
	Read/Write	R	R/W		R (Cleared to "0" when read)			R/W	
	After reset	0	0	0	0	0	0	0	0
	Function	Receive data bit 8 (For UART)	Parity (For UART) 0: Odd 1: Even	Add parity (For UART) 0: Disabled 1: Enabled	0: Normal operation 1: Error Overrun Parity/underrun Framing			0: SCLK0  1: SCLK0 	(For I/O interface) 0: Baud rate generator 1: SCLK0 pin input

<RB8>: 9th bit of the received data in the 9 bits UART mode.

<EVEN>: Selects even or odd parity.
 "0": odd parity.
 "1": even parity.
 The parity bit may be used only in the 7- or 8-bit UART mode.

<PE>: Controls enabling/ disabling parity.
 The parity bit may be used only in the 7- or 8-bit UART mode.

<OERR>: Error flag (**see note**)

<PERR>: Indicate overrun error, parity error, underrun error and framing error.

<FERR>:

<SCLKS>: Selects edge for data transmission and reception.
 "0": Data transmit/receive at rising edges of SCLK0
 "1": Data transmit/receive at falling edges of SCLK0

<IOC>: Selects input clock in the I/O interface mode.
 "0": baud rate generator
 "1": SCLK0 pin input.

(Note)	Any error flag is cleared when read.

9.4.4 Mode control register 0

	7	6	5	4	3	2	1	0
bit Symbol	TB8	CTSE	RXE	WU	SM1	SM0	SC1	SC0
Read/Write	R/W							
After reset	0	0	0	0	0	0	0	0
Function	Transmit data bit 8	Handshake function control 0: CTS disable 1: CTS enable	Receive control 0: Reception disabled 1: Reception enabled	Wake-up function 0: Reception disabled 1: Reception enabled	Serial transfer mode 00: I/O interface mode 01: 7-bit length UART mode 10: 8-bit length UART mode 11: 9-bit length UART mode		Serial transfer clock (for UART) 00: Timer TB7OUT 01: Baud rate generator 10: Internal clock f _{SYS} 11: External clock (SCLK0 input)	

<TB8>: Writes the 9th bit of transmit data in the 9 bits UART mode.

<CTSE>: Controls handshake function.
Setting "1" enables handshake function using $\overline{\text{CTS}}$ pin.

<RXE>: Controls reception (**see note**).
Set <RXE> after setting each mode register (SC0MOD0, SC0MOD1 and SC0MOD2).

<WU>: Controls wake-up function.
This function is available only at 9-bit UART mode.

	9-bit UART mode	Other modes
0	Interrupt when received	don't care
1	Interrupt only when RB9=1	

<SM1:0>: Specifies transfer mode.

<SC1:0>: Selects the serial transfer clock in the UART mode.
As for the I/O interface mode, the serial transfer clock can be set in the control register SC0CR.

(Note) With <RXE> set to "0," set each mode register (SC0MOD0, SC0MOD1 and SC0MOD2). Then set <RXE> to "1."

9.4.5 Mode control register 1

	7	6	5	4	3	2	1	0
bit Symbol	I2S0	FDPX1	FDPX0	TXE	SINT2	SINT1	SINT0	-
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0
Function	IDLE 0: Stop 1: Start	Transfer mode setting 00: Transfer prohibited 01: Half duplex(RX) 10: Half duplex(TX) 11: Full duplex		Transmit control 0: Disabled 1: Enabled	Interval time of continuous transmission (for I/O interface) 000: None 100: 8SCLK 001: 1SCLK 101: 16SCLK 010: 2SCLK 110: 32SCLK 011: 4SCLK 111: 64SCLK			Write "0".

<I2S0>: Specifies the IDLE mode operation.

<FDPX1:0>: Configures the transfer mode in the I/O interface mode. Also configures the FIFO if it is enabled. In the UART mode, it is used only to specify the FIFO configuration.

<TXE>: This bit enables transmission and is valid for all the transfer modes (**see note**). If disabled while transmission is in progress, transmission is inhibited only after the current frame of data is completed for transmission.

<SINT2:0>: Specifies the interval time of continuous transmission when double buffering or FIFO is enabled in the I/O interface mode. This parameter is valid only for the I/O interface mode when SCLK0 pin input is not selected.

(Note) Specify the mode first and then specify the <TXE> bit.

9.4.6 Mode control register 2

	7	6	5	4	3	2	1	0
bit Symbol	TBEMP	RBFL	TXRUN	SBLN	DRCHG	WBUF	SWRST1	SWRST0
Read/Write	R			R/W				
After reset	1	0	0	0	0	0	0	0
Function	Transmit buffer empty flag 0: full 1: Empty	Receive Buffer full flag 0: Empty 1: full	In transmission flag 0: Stop 1: Start	STOP bit (for UART) 0: 1-bit 1: 2-bit	Setting transfer direction 0: LSB first 1: MSB first	W-buffer 0: Disabled 1: Enabled	SOFT RESET Overwrite "01" on "10" to reset.	

<TBEMP>: This flag shows that the transmit double buffers are empty. When data in the transmit double buffers is moved to the transmit shift register and the double buffers are empty, this bit is set to "1." Writing data again to the double buffers sets this bit to "0." If double buffering is disabled, this flag is insignificant.

<RBFL>: This is a flag to show that the receive double buffers are full. When a receive operation is completed and received data is moved from the receive shift register to the receive double buffers, this bit changes to "1" while reading this bit changes it to "0." If double buffering is disabled, this flag is insignificant.

<TXRUN>: This is a status flag to show that data transmission is in progress.
<TXRUN> and <TBEMP> bits indicate the following status.

<TXRUN>	<TBEMP>	Status
1	-	Transmission in progress
0	1	Transmission completed
	0	Wait state with data in TX buffer

<SBLEN>: This specifies the length of stop bit transmission in the UART mode. On the receive side, the decision is made using only a single bit regardless of the <SBLEN> setting.

<DRCHG>: Specifies the direction of data transfer in the I/O interface mode. In the UART mode, it is fixed to LSB first.

<WBUF>: This parameter enables or disables the transmit/receive buffers to transmit (in both SCLK output/input modes) and receive (in SCLK output mode) data in the I/O interface mode and to transmit data in the UART. When receiving data in the I/O interface mode (I SCLK input) and UART mode, double buffering is enabled in both cases that 0 or 1 is set to <WBUF> bit.

<SWRST1:0>: Overwriting "01" in place of "10" generates a software reset. When this software reset is executed, the following bits and their internal circuits are initialized (**see note 1, 2 and 3**).

Register	Bit
SC0MOD0	RXE
SC0MOD1	TXE
SC0MOD2	TBEMP,RBFLL,TXRUN,
SC0CR	OERR,PERR,FERR

(Note 1) While data transmission is in progress, any software reset operation must be executed twice in succession.

(Note 2) A software reset requires 2 clocks-duration at the time between the end of recognition and the start of execution of software reset instruction.

(Note 3) A software reset initializes other bits. Resetting a mode register and a control register are needed.

9.4.7 Baud rate generator control register(SC0BRCCR) Baud rate generator control register 2(SC0BRADD)

SC0BRCCR		7	6	5	4	3	2	1	0
	bit Symbol	-	BR0ADDE	BR0CK1	BR0CK0	BR0S3	BR0S2	BR0S1	BR0S0
	Read/Write	R/W							
	After reset	0	0	0	0	0	0	0	0
	Function	Write "0".	N + (16 - K)/ 16 divider function 0: disabled 1: enabled	Select input clock to the baud rate generator 00: $\phi T1$ 01: $\phi T4$ 10: $\phi T16$ 11: $\phi T64$		Division ratio "N" 0000: 16 0001: 1 0010: 2 : 1111: 15			

SC0BRADD		7	6	5	4	3	2	1	0
	bit Symbol					BR0K3	BR0K2	BR0K1	BR0K0
	Read/Write	R				R/W			
	After reset	0				0	0	0	0
	Function	"0" is read.				Specify K for the "N + (16 - K)/16" division 0000: Prohibited 0001: K=1 0010: K=2 : 1111: K=15			

<RB0ADDE>: Specifies N + (16-K)/16 division function.
N + (16-K)/16 division function can only be used in the UART mode.

<RB0CK1:0>: Specifies the baud rate generator input clock.

<RB0S3:0>: Specifies division ratio "N".

<RB0K3:0>: Specifies K for the "N+(16-K)/16" division.

The division ratio of the baud rate generator can be specified in the registers shown above.
Table 9-6 lists the settings of baud rate generator division ratio.

Table 9-6 Setting division ratio

	BR0ADDE=0	BR0ADDE=1 (Note 1) (Only UART)
BR0S	Specify "N" (Note 2) (Note 3)	
BR0K	No setting required	Setting "K" (Note 4)
Division ratio	Divide by N	$N + \frac{(16-K)}{16}$ division

- (Note 1) To use the " $N + (16 - K)/16$ " division function, be sure to set BR0K <BR0ADDE> to "1" after setting the K value to BR0K. The " $N + (16 - K)/16$ " division function can only be used in the UART mode.
- (Note 2) The division ratio "1" of the baud rate generator can be specified only when
- the " $N + (16 - K)/16$ " division function is not used in the UART mode.
 - double buffering is used in the I/O interface mode.
- (Note 3) As a division ratio, 1 ("0001") or 16 ("0000") cannot be applied to N when using the " $N + (16 - K)/16$ " division function.
- (Note 4) Specifying "K = 0" is prohibited.

9.4.8 FIFO configuration register

	7	6	5	4	3	2	1	0
bit Symbol	Reserved	Reserved	Reserved	RFST	TFIE	RFIE	RXTXCNT	CNFG
Read/Write	R/W							
After reset	0	0	0	0	0	0	0	0
Function	Be sure to write "000".			Bytes used in RX FIFO 0: Maximum 1: Same as FILL level of RX FIFO	TX interrupt for TX FIFO 0: Disabled 1: Enabled	RX interrupt for RX FIFO 0: Disabled 1: Enabled	Automatic disable of RXE/TXE 0: None 1: Auto disable	FIFO enable 0: Disabled 1: Enabled

- <RFST>: When RX FIFO is enabled, the number of RX FIFO bytes to be used is selected (see note).
0: The maximum number of bytes of the FIFO configured (see also <CNFG>).
1: Same as the fill level for receive interrupt generation specified by SC0RFC <RIL1:0>.
- <TFIE>: When TX FIFO is enabled, transmit interrupts are enabled or disabled by this parameter.
- <RFIE>: When RX FIFO is enabled, receive interrupts are enabled or disabled by this parameter.
- <RXTXCNT>: Controls automatic disabling of transmission and reception.
The mode control register SCOMOD1 <FDPX1:0> is used to set the types of TX/RX.
Setting "1" enables to operate as follows.

Half duplex RX	When the RX FIFO is filled up to the specified number of valid bytes, SC0MOD0<RXE> is automatically set to "0" to inhibit further reception.
Half duplex TX	When the TX FIFO is empty, SC0MOD1<TXE> is automatically set to "0" to inhibit further transmission.
Full duplex	When either of the above two conditions is satisfied, TXE/RXE are automatically set to "0" to inhibit further transmission and reception.

- <CNFG>: Enables FIFO.
If enabled, the SCOMOD1 <FDPX1:0> setting automatically configures FIFO as follows:
(The type of TX/RX can be specified in the mode control register 1 SCOMOD1<FDPX1:0>).

Half duplex RX	RX FIFO 4byte
Half duplex TX	TX FIFO 4byte
Full duplex	RX FIFO 2byte + TX FIFO 2byte

(Note) Regarding TX FIFO, the maximum number of bytes being configured is always available. The available number of bytes is the bytes already written to the TX FIFO.

9.4.9 RX FIFO configuration register

SC0RFC		7	6	5	4	3	2	1	0
	bit Symbol	RFCS	RFIS					RIL1	RIL0
	Read/Write	W	R/W	R				R/W	
	After reset	0	0	0				0	0
	Function	RX FIFO clear 1: Clear "0" is read.	Select interrupt generation condition 0: when the data reaches to the specified fill level. 1: when the data reaches to the specified fill level or the data exceeds the specified fill level at the time data is read.	"0" is read.				FIFO fill level to generate RX interrupts 00:4byte(2 Byte at full duplex) 01:1byte 10:2byte 11:3byte	

<RFCS>: Clears RX FIFO
Setting "1" clears RX FIFO and "0" is always read.

<RFIS>: Specifies the condition of interrupt generation.
0: An interrupt is generated when it reaches to the specified fill level.
An interrupt is generated when it is reaches to the specified fill level or if it exceeds the specified fill level at the time data is read.

<RIL1:0>: Specifies FIFO fill level(see note).

	Other than full duplex	Full duplex
00	4byte	2byte
01	1byte	1byte
10	2byte	2byte
11	3byte	1byte

(Note) RIL1 is ignored when FDPX1:0 = 11 (full duplex)

9.4.10 TX FIFO configuration register

		7	6	5	4	3	2	1	0
SC0TFC	bit Symbol	TFCS	TFIS					TIL1	TIL0
	Read/Write	W	R/W	R				R/W	
	After reset	0	0	0				0	0
	Function	TX FIFO clear 1: Clear Always reads "0".	Select interrupt generation condition 0: when the data reaches to the specified fill level. 1: when the data reaches to the specified fill level or the data cannot reach the specified fill level at the time new data is read.	"0" is read.				FIFO fill level to generate TX interrupts. 00: Empty 01: 1byte 10: 2byte 11: 3byte Note: TIL1 is ignored when FDPX1:0=11 (full duplex).	

<TFCS>: Clears TX FIFO.
Setting "1" clears TX FIFO and "0" is always read.

<TFIS>: Selects interrupt generation condition.
0: An interrupt is generated when the data reaches to the specified fill level.
1: An interrupt is generated when the data reaches to the specified fill level or the data cannot reach the specified fill level at the time new data is read.

<TIL1:0>: Selects FIFO fill level (see note).

	Other than full duplex	Full duplex
00	Empty	Empty
01	1byte	1byte
10	2byte	Empty
11	3byte	1byte

(Note) TIL1 is ignored when FDPX1:0 = 11 (full duplex).

9.4.11 RX FIFO status register

	7	6	5	4	3	2	1	0
bit Symbol	ROR					RLVL2	RLVL1	RLVL0
Read/Write	R	R				R		
After reset	0	0				0	0	0
Function	RX FIFO Overflow 1: Generate d	"0" is read.				Status of RX FIFO fill level 000:Empty 001:1Byte 010:2Byte 011:3Byte 100:4Byte		

<ROR>: Flags for RX FIFO overrun. When the overrun occurs, these bits are set to "1" (see note).

<RLVL2:0>: Shows the fill level of RX FIFO.

(Note) The <ROR> bit is cleared to "0" when receive data is read from the SC0BUF register.

9.4.12 TX FIFO status register

	7	6	5	4	3	2	1	0
bit Symbol	TUR					TLVL2	TLVL1	TLVL0
Read/Write	R	R				R		
After reset	1	0				0	0	0
Function	TX FIFO Under run 1:Generate d Cleared by writing FIFO	"0" is read.				Status of TX FIFO fill level 000:Empty 001:1Byte 010:2Byte 011:3Byte 100:4Byte		

<TUR>: Flags for TX FIFO underrun. When the underrun occurs, these bits are set to "1" (see note).

<TLVL2:0>: Shows the fill level of TX FIFO.

(Note) The <TUR> bit is cleared to "0" when transmit data is written to the SC0BUF register.

9.5 Operation in Each Mode

9.5.1 Mode 0 (I/O interface mode)

Mode 0 consists of two modes, the “SCLK output” mode to output synchronous clock and the “SCLK input” mode to accept synchronous clock from an external source. The following operational descriptions are for the case use of FIFO is disabled. For details of FIFO operation, refer to the previous sections describing receive/transmit FIFO functions.

① Transmitting data

SCLK output mode

In the SCLK output mode, if SC0MOD2<WBUF> is set to “0” and the transmit double buffers are disabled, 8 bits of data are output from the TXD0 pin and the synchronous clock is output from the SCLK0 pin each time the CPU writes data to the transmit buffer. When all data is output, the INTTX0 interrupt is generated.

If SC0MOD2 <WBUF> is set to “1” and the transmit double buffers are enabled, data is moved from Transmit Buffer 2 to Transmit Buffer 1 when the CPU writes data to Transmit Buffer 2 while data transmission is halted or when data transmission from Transmit Buffer 1 (shift register) is completed. When data is moved from Transmit Buffer 2 to Transmit Buffer 1, the transmit buffer empty flag SC0MOD2 <TBEMP> is set to “1,” and the INTTX0 interrupt is generated. If Transmit Buffer 2 has no data to be moved to Transmit Buffer 1, the INTTX0 interrupt is not generated and the SCLK0 output stops.

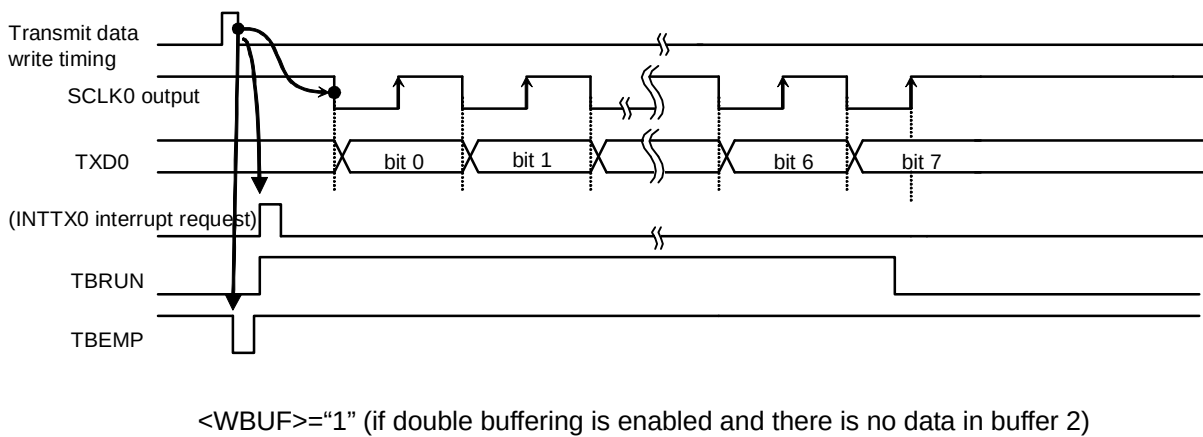
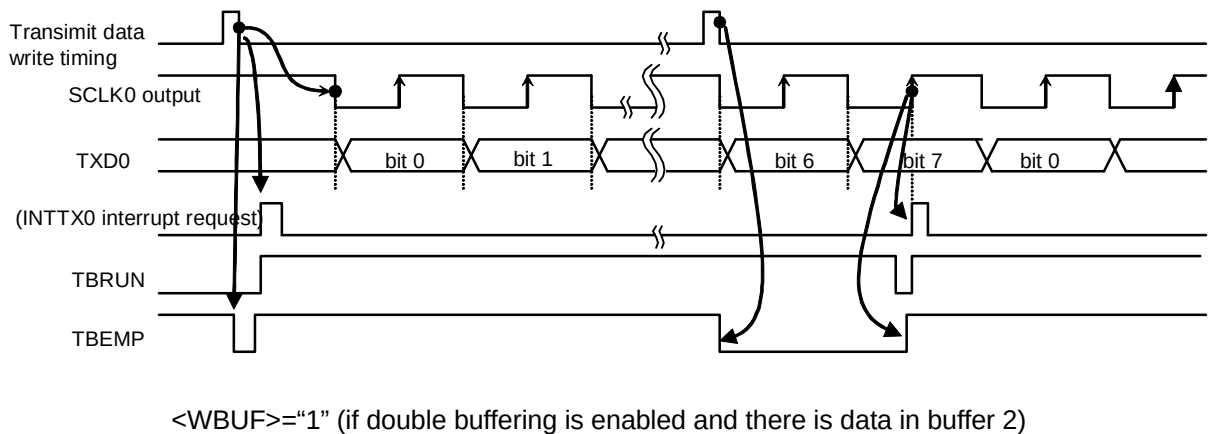
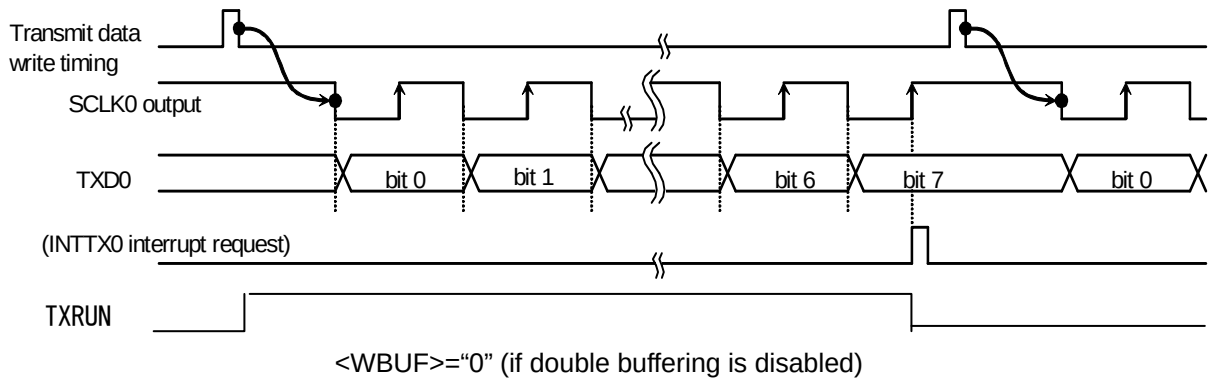
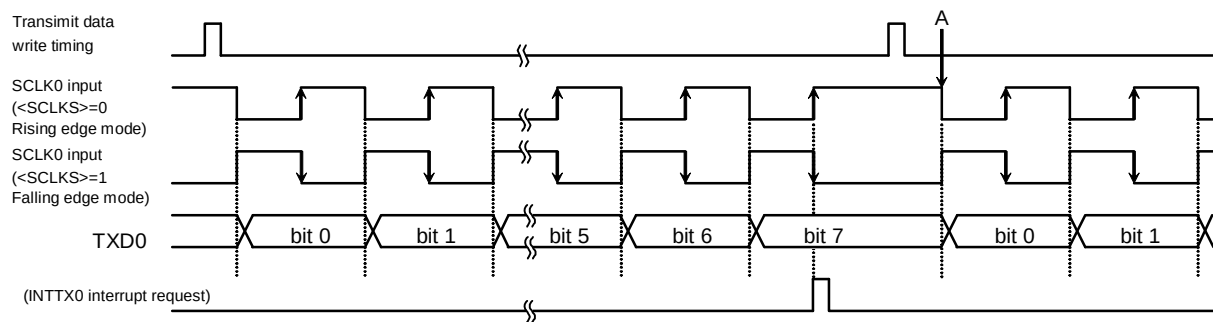


Fig. 9-10 Transmit Operation in the I/O Interface Mode (SCLK0 Output Mode)

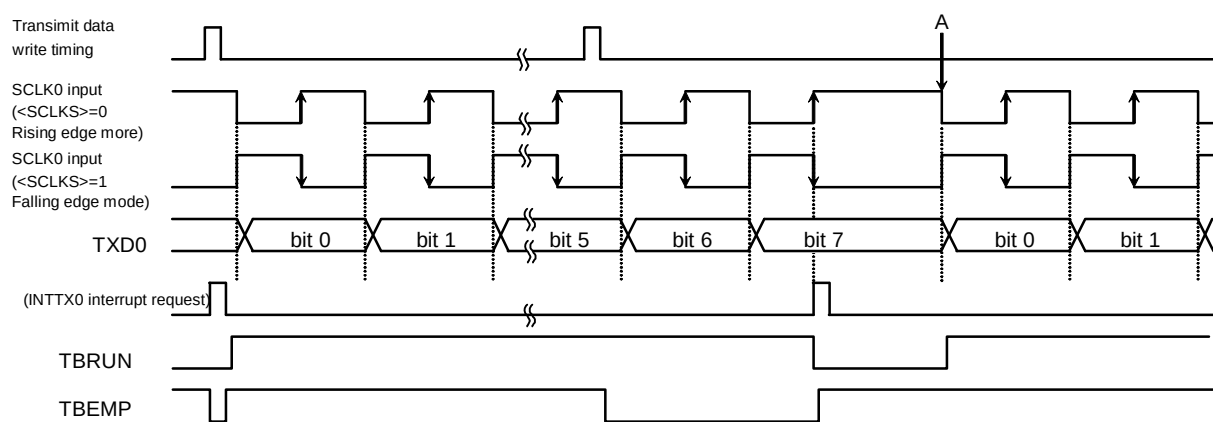
SCLK input mode

In the SCLK input mode, if SC0MOD2 <WBUF> is set to “0” and the transmit double buffers are disabled, 8-bit data that has been written in the transmit buffer is output from the TXD0 pin when the SCLK0 input becomes active. When all 8 bits are sent, the INTTX0 interrupt is generated. The next transmit data must be written before the timing point “A” as shown in Fig. 9-11.

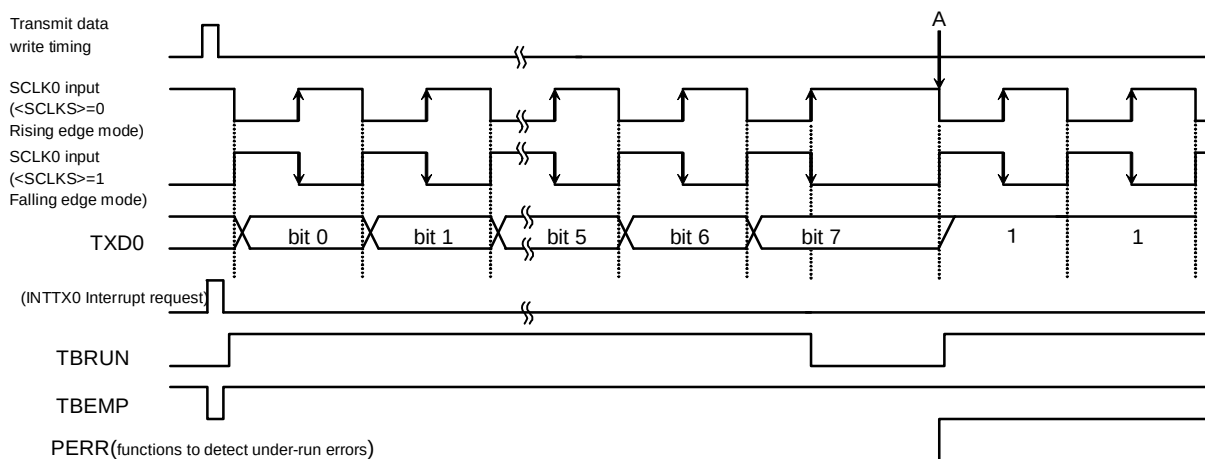
If SC0MOD2 <WBUF> is set to “1” and the transmit double buffers are enabled, data is moved from Transmit Buffer 2 to Transmit Buffer 1 when the CPU writes data to Transmit Buffer 2 before the SCLK0 becomes active or when data transmission from Transmit Buffer 1 (shift register) is completed. As data is moved from Transmit Buffer 2 to Transmit Buffer 1, the transmit buffer empty flag SC0MOD2 <TBEMP> is set to “1” and the INTTX0 interrupt is generated. If the SCLK0 input becomes active while no data is in Transmit Buffer 2, although the internal bit counter is started, an under-run error occurs and 8-bit dummy data (FFh) is sent.



<WBUF>="0" (if double buffering is disabled)



<WBUF>="1" (if double buffering is enabled and there is data in buffer 2)



<WBUF>="1" (if double buffering is enabled and there is no data in buffer 2)

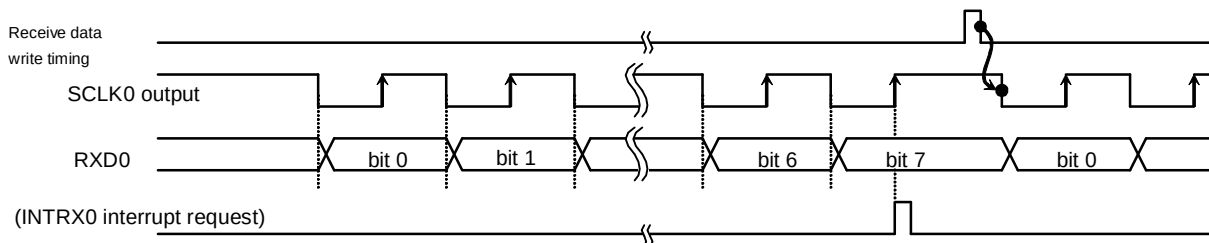
Fig. 9-11 Transmit Operation in the I/O Interface Mode (SCLK0 Input Mode)

② Receiving data
SCLK output mode

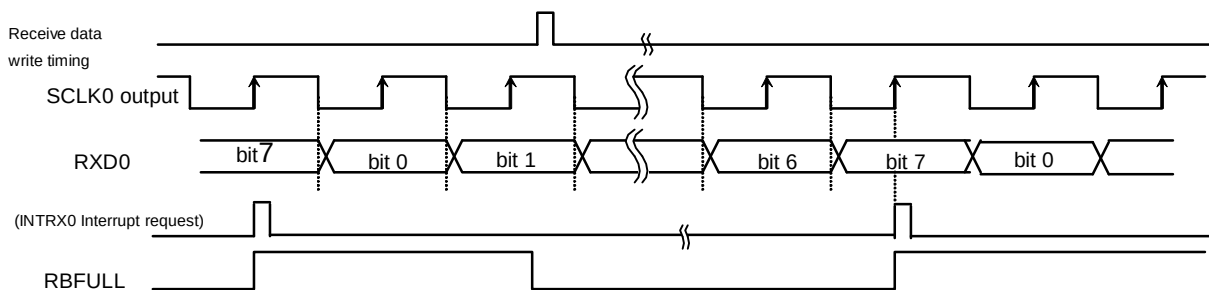
In the SCLK output mode, if SC0MOD2 <WBUF> = "0" and receive double buffering is disabled, a synchronous clock pulse is output from the SCLK0 pin and the next data is shifted into receive buffer 1 each time the CPU reads received data. When all the 8 bits are received, the INTRX0 interrupt is generated.

The first SCLK output can be started by setting the receive enable bit SC0MOD0 <RXE> to "1." If the receive double buffering is enabled with SC0MOD2 <WBUF> set to "1," the first frame received is moved to receive buffer 2 and receive buffer 1 can receive the next frame successively. As data is moved from receive buffer 1 to receive buffer 2, the receive buffer full flag SC0MOD2 <RBFULL> is set to "1" and the INTRX0 interrupt is generated.

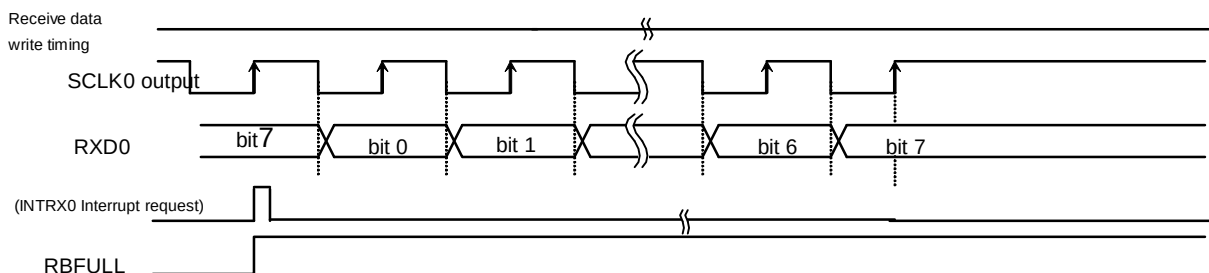
While data is in receive buffer 2, if CPU/DMAC cannot read data from receive buffer 2 before completing reception of the next 8 bits, the INTRX0 interrupt is not generated and the SCLK0 clock stops. In this state, reading data from receive buffer 2 allows data in receive buffer 1 to move to receive buffer 2 and thus the INTRX0 interrupt is generated and data reception resumes.



<WBUF>="0" (if double buffering is disabled)



<WBUF>="1" (if double buffering is enabled and data is read from buffer 2)



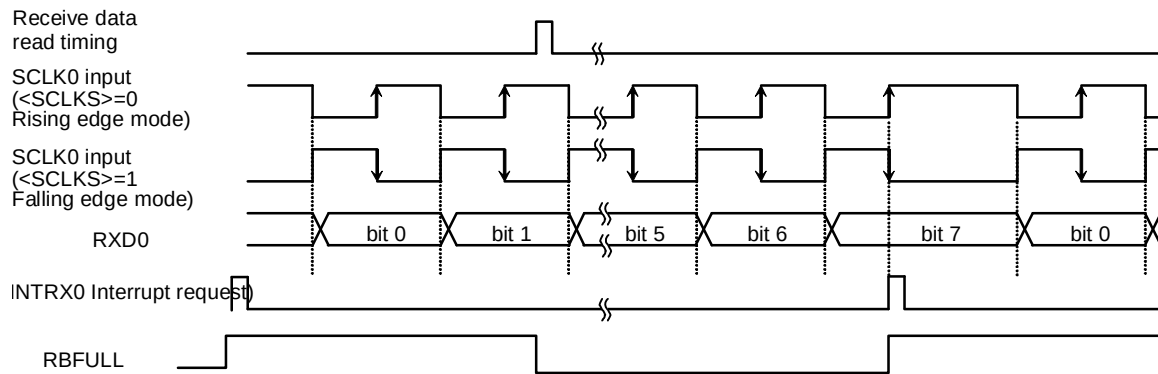
<WBUF>="1" (if double buffering is enabled and data cannot be read from buffer 2)

Fig. 9-12 Receive Operation in the I/O Interface Mode (SCLK0 Output Mode)

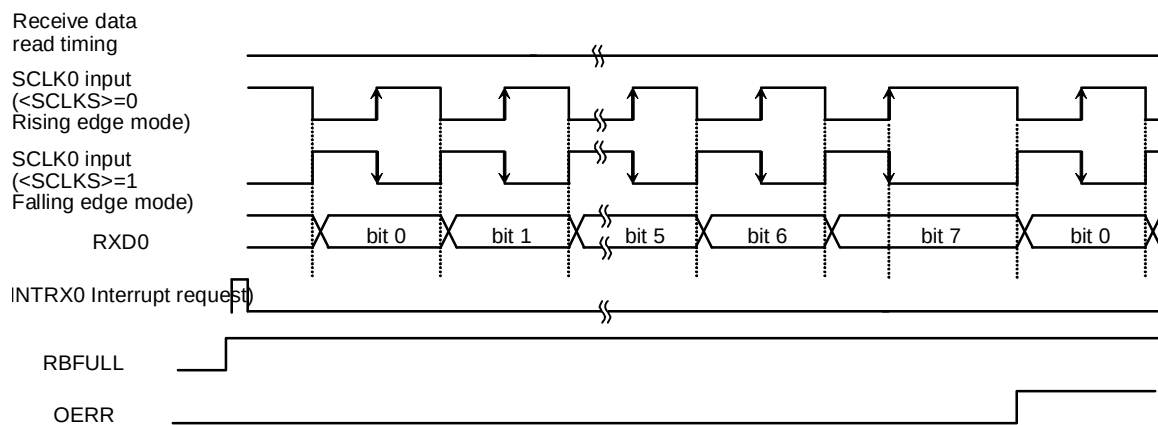
SCLK input mode

In the SCLK input mode, receiving double buffering is always enabled, the received frame can be moved to receive buffer 2 and receive buffer 1 can receive the next frame successively.

The INTRX receive interrupt is generated each time received data is moved to received buffer 2.



If data is read from buffer 2



If data cannot be read from buffer 2

Fig. 9-13 Receive Operation in the I/O Interface Mode (SCLK0 Input Mode)

(Note) To receive data, SC0MOD <RXE> must always be set to "1" (receive enable) in the SCLK output / SCLK input mode.

③ Transmit and receive (full-duplex)

The full-duplex mode is enabled by setting bit 6 <FDPX0> of the serial mode control register 1 (SC0MOD1) to "1".

SCLK output mode

In the SCLK output mode, if SC0MOD2 <WBUF> is set to "0" and both the transmit and receive double buffers are disabled, SCLK is output when the CPU writes data to the transmit buffer. Subsequently, 8 bits of data are shifted into receive buffer 1 and the INTRX0 receive interrupt is generated. Concurrently, 8 bits of data written to the transmit buffer are output from the TXD0 pin, the INTTX0 transmit interrupt is generated when transmission of all data bits has been completed. Then, the SCLK output stops. In this, the next round of data transmission and reception starts when the data is read from the receive buffer and the next transmit data is written to the transmit buffer by the CPU. The order of reading the receive buffer and writing to the transmit buffer can be freely determined. Data transmission is resumed only when both conditions are satisfied.

If SC0MOD2 <WBUF> = "1" and double buffering is enabled for both transmission and reception, SCLK is output when the CPU writes data to the transmit buffer. Subsequently, 8 bits of data are shifted into receive buffer 1, moved to receive buffer 2, and the INTRX0 interrupt is generated. While 8 bits of data is received, 8 bits of transmit data is output from the TXD0 pin. When all data bits are sent out, the INTTX0 interrupt is generated and the next data is moved from the Transmit Buffer 2 to Transmit Buffer 1. If Transmit Buffer 2 has no data to be moved to Transmit Buffer 1 (SC0MOD2 <TBEMP> = 1) or when receive buffer 2 is full (SC0MOD2 <RBFULL> = 1), the SCLK clock is stopped. When both conditions, receive data is read and transmit data is written, are satisfied, the SCLK output is resumed and the next round of data transmission is started.

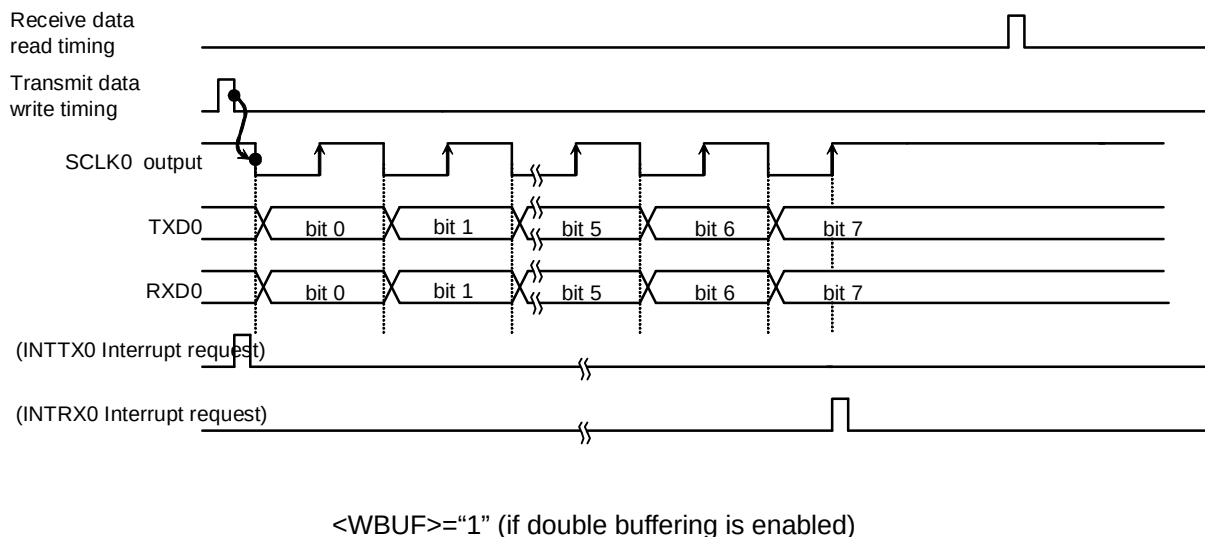
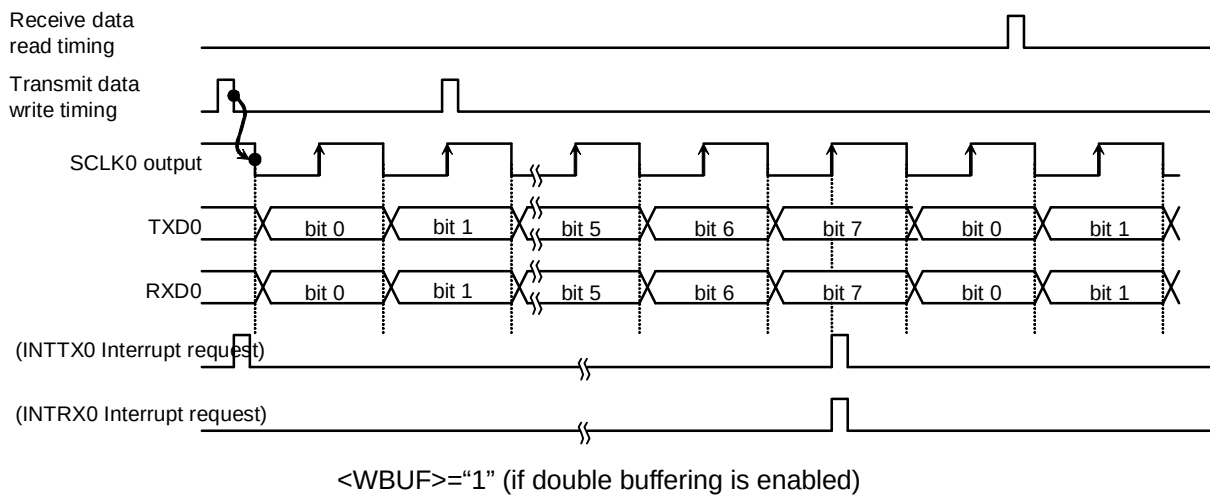
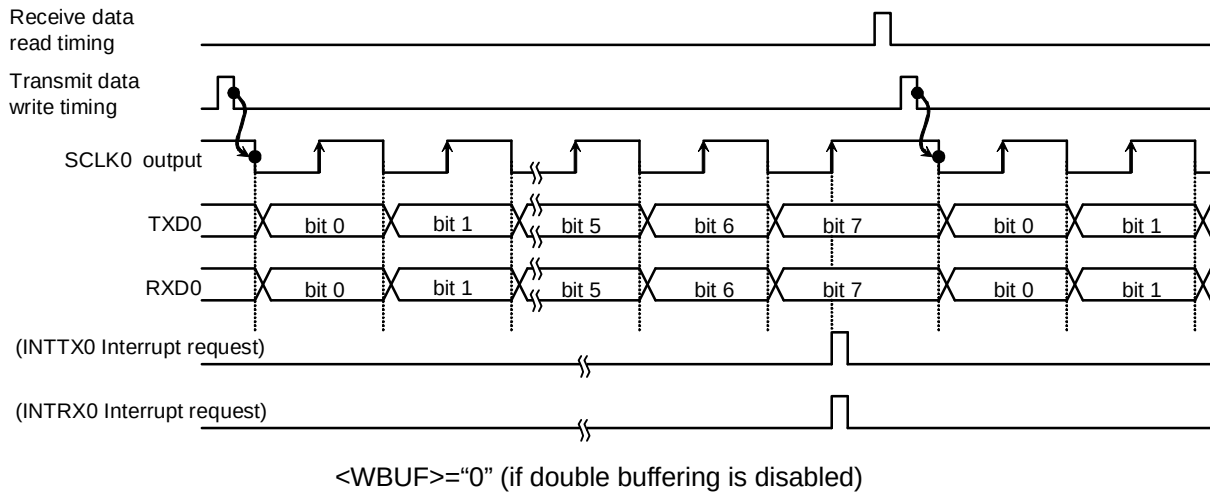


Fig. 9-14 Transmit/Receive Operation in the I/O Interface Mode (SCLK0 Output Mode)

SCLK input mode

In the SCLK input mode with SC0MOD2 <WBUF> set to "0" and the transmit double buffers are disabled (double buffering is always enabled for the receive side), 8-bit data written in the transmit buffer is output from the TXD0 pin and 8 bits of data is shifted into the receive buffer when the SCLK input becomes active. The INTTX0 interrupt is generated upon completion of data transmission and the INTRX0 interrupt is generated at the instant the received data is moved from receive buffer 1 to receive buffer 2. Note that transmit data must be written into the transmit buffer before the SCLK input for the next frame (data must be written before the point A in Fig. 9-15). As double buffering is enabled for data reception, data must be read before completing reception of the next frame data.

If SC0MOD2 <WBUF> = "1" and double buffering is enabled for both transmission and reception, the interrupt INTRX0 is generated at the timing Transmit Buffer 2 data is moved to Transmit Buffer 1 after completing data transmission from Transmit Buffer 1. At the same time, the 8 bits of data received is shifted to buffer 1, it is moved to receive buffer 2, and the INTRX0 interrupt is generated. Upon the SCLK input for the next frame, transmission from Transmit Buffer 1 (in which data has been moved from Transmit Buffer 2) is started while receive data is shifted into receive buffer 1 simultaneously. If data in receive buffer 2 has not been read when the last bit of the frame is received, an overrun error occurs. Similarly, if there is no data written to Transmit Buffer 2 when SCLK for the next frame is input, an under-run error occurs.

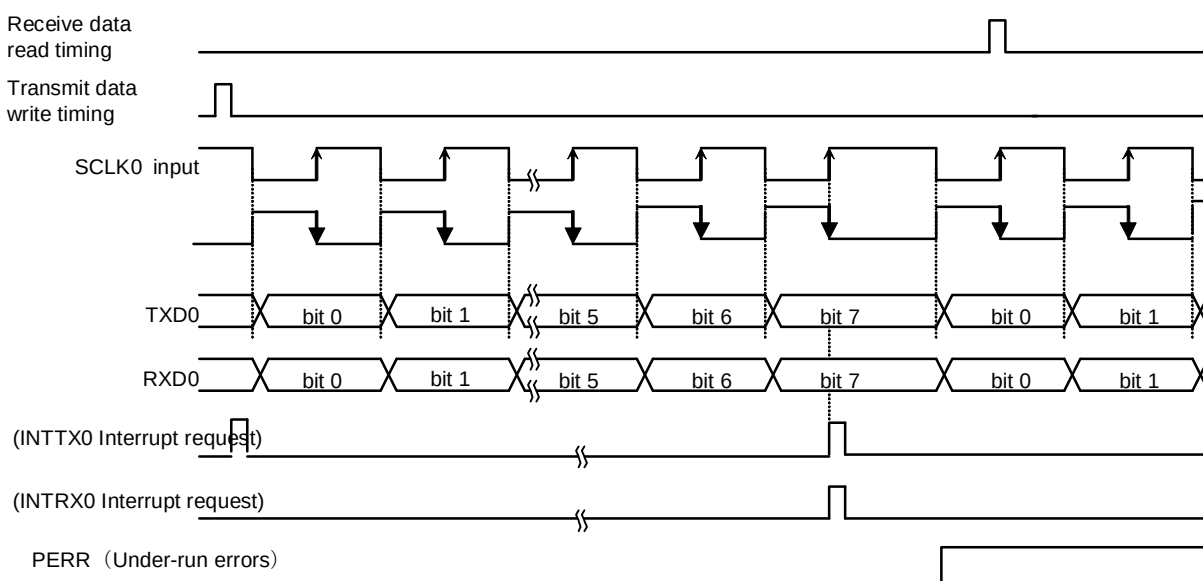
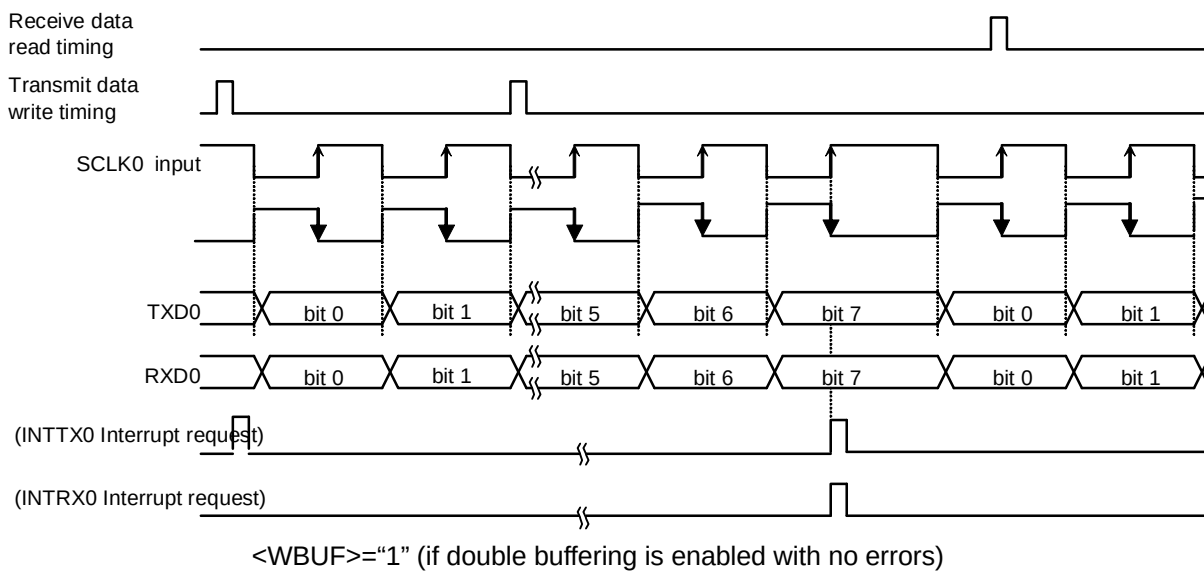
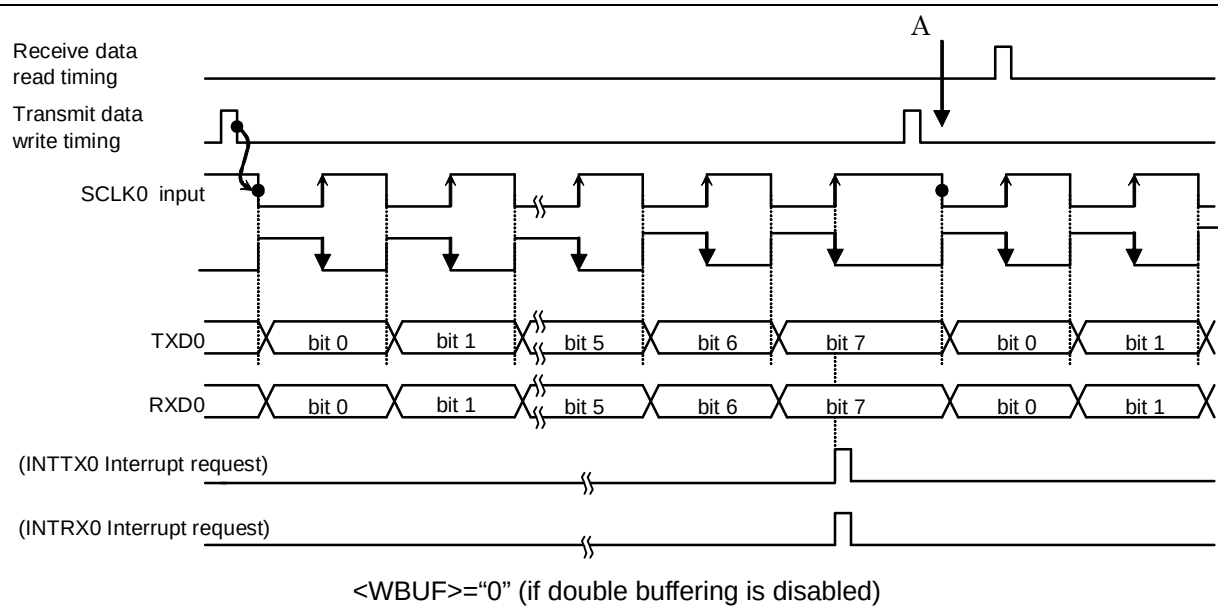


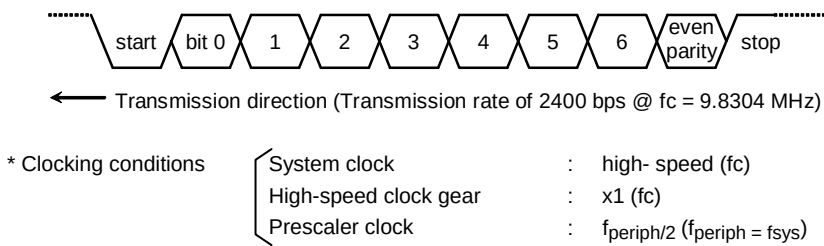
Fig. 9-15 Transmit/Receive Operation in the I/O Interface Mode (SCLK0 Input Mode)

9.5.2 Mode 1 (7-bit UART Mode)

The 7-bit UART mode can be selected by setting the serial mode control register (SC0MOD <SM1, 0>) to “01”.

In this mode, parity bits can be added to the transmit data stream; the serial mode control register (SC0CR <PE>) controls the parity enable/disable setting. When <PE> is set to “1” (enable), either even or odd parity may be selected using the SC0CR <EVEN> bit. The length of the stop bit can be specified using SC0MOD2<SBLEN>.

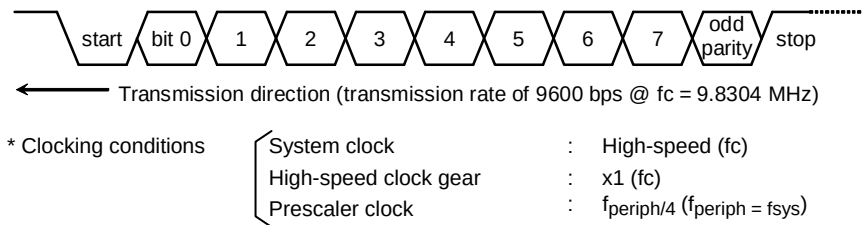
The following table shows the control register settings for transmitting in the following data format.



9.5.3 Mode 2 (8-bit UART Mode)

The 8-bit UART mode can be selected by setting SC0MOD0 <SM1:0> to “10.” In this mode, parity bits can be added and parity enable/disable is controlled using SC0CR <PE>. If <PE> = “1” (enabled), either even or odd parity can be selected using SC0CR <EVEN>.

The control register settings for receiving data in the following format are as follows:



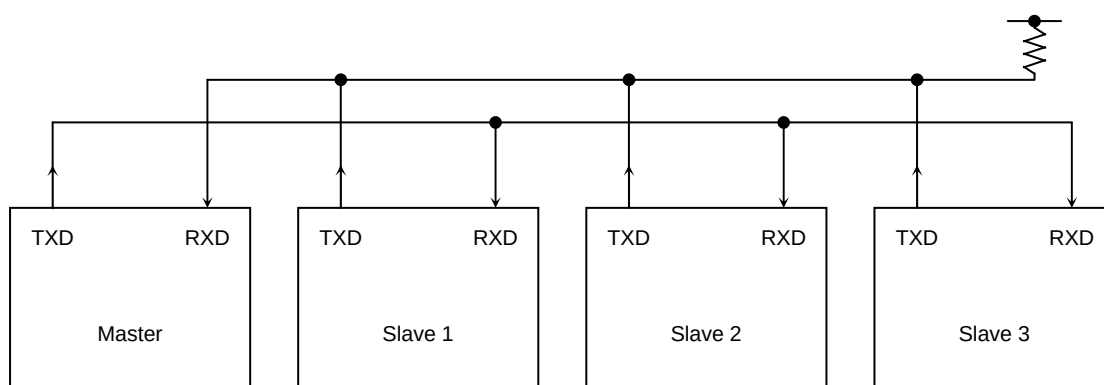
9.5.4 Mode 3 (9-bit UART)

The 9-bit UART mode can be selected by setting SC0MOD0 <SM1:0> to "11." In this mode, parity bits must be disabled (SC0CR <PE> = "0").

The most significant bit (9th bit) is written to bit 7 <TB8> of the serial mode control register 0 (SC0MOD0) for transmitting data. The data is stored in bit 7 <RB8> of the serial control register SC0CR. When writing or reading data to/from the buffers, the most significant bit must be written or read first before writing or reading to/from SC0BUF. The stop bit length can be specified using SC0MOD2 <SBLEN>.

Wakeup function

In the 9-bit UART mode, slave controllers can be operated in the wake-up mode by setting the wake-up function control bit SC0MOD0 <WU> to "1." In this case, the interrupt INTRX0 will be generated only when SC0CR <RB8> is set to "1".

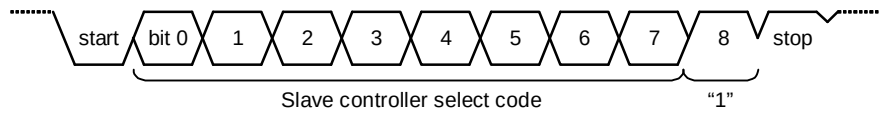


(Note) The TXD pin of the slave controller must be set to the open drain output mode using the ODE register.

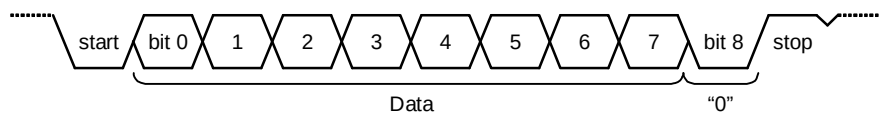
Fig. 9-16 Serial Links to Use Wake-up Function

Protocol

- ① Select the 9-bit UART mode for the master and slave controllers.
- ② Set SC0MOD <WU> to "1" for the slave controllers to make them ready to receive data.
- ③ The master controller is to transmit a single frame of data that includes the slave controller select code (8 bits). In this, the most significant bit (bit 8) <TB8> must be set to "1".

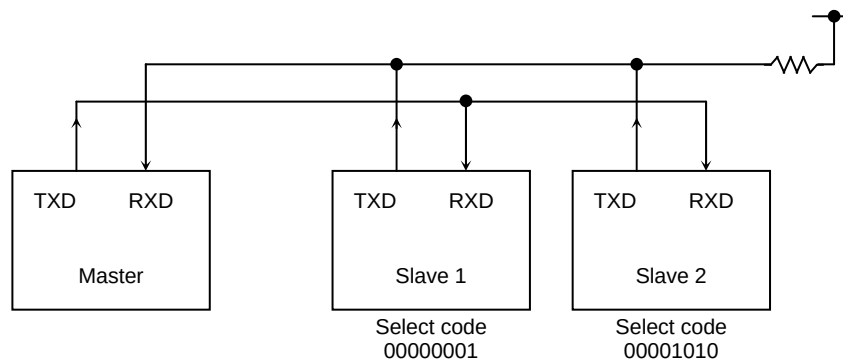


- ④ Each slave controller receives the above data frame; if the code received matches with the controller's own select code, it clears the WU bit to "0".
- ⑤ The master controller transmits data to the designated slave controller (the controller of which SC0MOD <WU> bit is cleared to "0"). In this, the most significant bit (bit 8) <TB8> must be set to "0".



- ⑥ The slave controllers with the <WU> bit set to "1" ignore the receive data because the most significant bit (bit 8) <RB8> is set to "0" and thus no interrupt (INTRX0) is generated. Also, the slave controller with the <WU> bit set to "0" can transmit data to the master controller to inform that the data has been successfully received.

An example: Using the internal clock f_{SYS} as the transfer clock, two slave controllers are serially linked as follows.



10. Serial Bus Interface (SBI)

The TMPM330 contains three Serial Bus Interface (SBI) channels, in which the following two operating modes are included:

- I²C bus mode (with multi-master capability)
- Clock-synchronous 8-bit SIO mode

In the I²C bus mode, the SBI is connected to external devices via SCL and SDA. In the clock-synchronous 8-bit SIO mode, the SBI is connected to external devices via SCK, SI and SO.

The following table shows the programming required to put the SBI in each operating mode.

		Pin name (PIN No.)	Port Function Register	Port Control Register	Port Input Register	Port Open Drain Output
Channel 0	I ² C bus mode	SCL : PG1 (27) SDA : PG0 (26)	PGFR1<1:0> = 11	PGCR<1:0> = 11	PGIE<1:0> = 11	PGOD<1:0> = 11
	Clock-synchronous 8-bit SIO mode	SCK : PG2 (28) SI : PG1 (27) SO : PG0 (26)	PGFR1<2:0> = 111	PGCR<2:0> = 101	PGIE<2:0> = 110	PGOD<2:0> = xxx
Channel 1	I ² C bus mode	SCL : PF5 (45) SDA : PF4 (44)	PFFR1<5:4> = 11	PFCR<5:4> = 11	PFIE<5:4> = 11	PFOD<5:4> = 11
	Clock-synchronous 8-bit SIO mode	SCK : PF6 (46) SI : PF5 (45) SO : PF4 (44)	PFFR1<6:4> = 111	PFCR<6:4> = 101	PFIE<6:4> = 110	PFOD<6:4> = xxx
Channel 2	I ² C bus mode	SCL : PF1 (34) SDA : PF0 (33)	PFFR1<1:0> = 11	PFCR<1:0> = 11	PFIE<1:0> = 11	PFOD<1:0> = 11
	Clock-synchronous 8-bit SIO mode	SCK : PF2 (35) SI : PF1 (34) SO : PF0 (33)	PFFR1<2:0> = 111	PFCR<2:0> = 101	PFIE<2:0> = 110	PFOD<2:0> = xxx

X: Don't care

10.1 Configuration

The configuration is shown in Fig. 10-1.

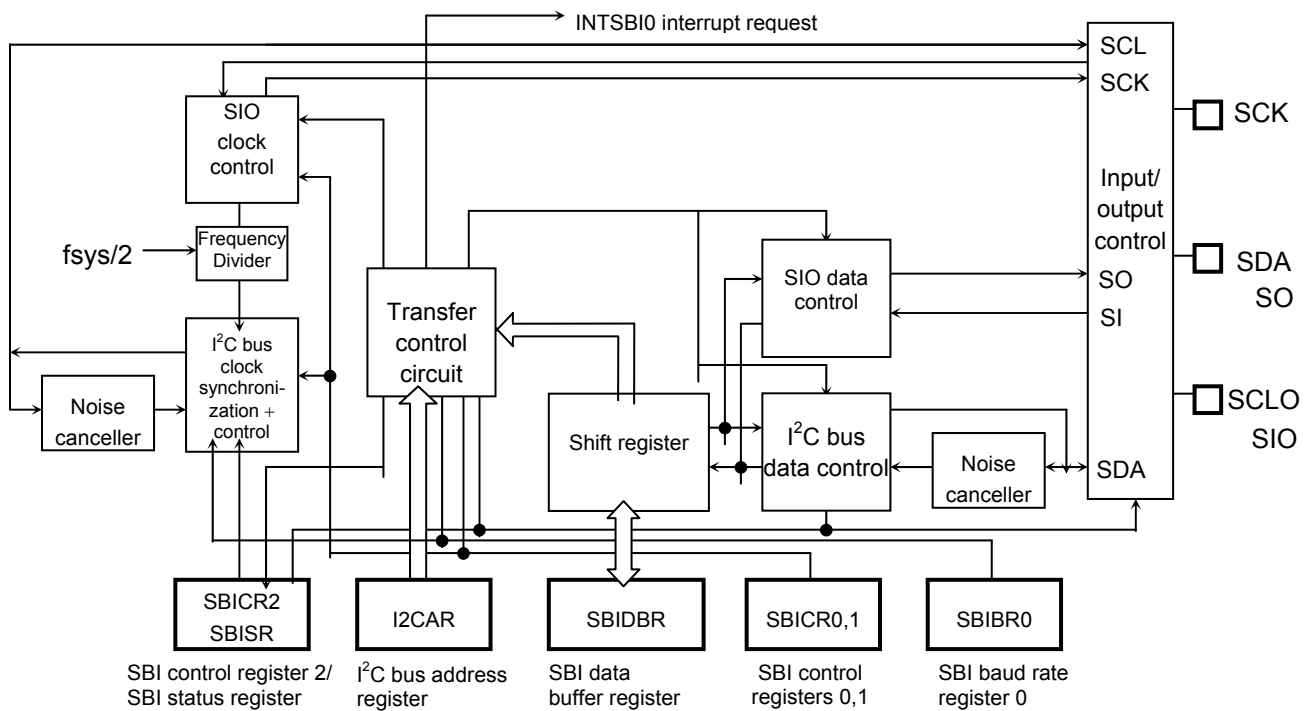


Fig. 10-1 SBI Block Diagram

10.2 Control

The following registers control the serial bus interface and provide its status information for monitoring.

- Serial bus interface control registers 0 (SBIxCR0)
- Serial bus interface control registers 1 (SBIxCR1)
- Serial bus interface control registers 2 (SBIxCR2)
- Serial bus interface buffer registers (SBIxDBR)
- I²C bus address register (SBIxI2CAR)
- Serial bus interface status registers (SBIxSR)
- Serial bus interface baud rate registers 0 (SBIxBR0)

The functions of these registers vary, depending on the mode in which the SBI is operating. For a detailed description of the registers, refer to “10.5 Control in the I2C Bus Mode” and “10.7 Control in the Clock-synchronous 8-bit SIO Mode”.

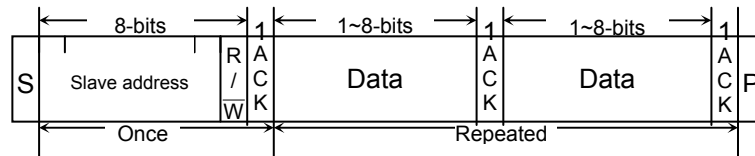
The addresses of each register are shown below.

		Channel 0	Channel 1	Channel 2
Register name (address)	Serial bus interface control register 0	SBI0CR0 0x4002_0000	SBI1CR0 0x4002_0020	SBI2CR0 0x4002_0040
	Serial bus interface control register 1	SBI0CR1 0x4002_0004	SBI1CR1 0x4002_0024	SBI2CR1 0x4002_0044
	Serial bus interface control register 2	SBI0CR2 (reading) 0x4002_0010	SBI1CR2 (reading) 0x4002_0030	SBI2CR2 (reading) 0x4002_0050
	Serial bus interface status register	SBI0SR (writing)	SBI1SR (writing)	SBI2SR (writing)
	Serial bus interface baud rate register 0	SBI0BR0 0x4002_0014	SBI1BR0 0x4002_0034	SBI2BR0 0x4002_0054
	Serial bus interface data buffer register	SBI0DBR 0x4002_0008	SBI1DBR 0x4002_0028	SBI2DBR 0x4002_0048
	I ² C bus address register	SBI0I2CAR 0x4002_000C	SBI1I2CAR 0x4002_002C	SBI2I2CAR 0x4002_004C

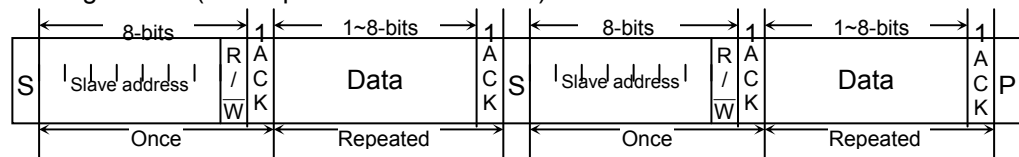
10.3 I²C Bus Mode Data Formats

Fig. 10-2 shows the data formats used in the I²C bus mode.

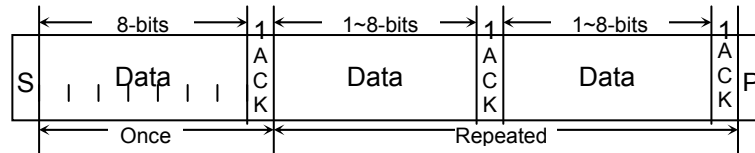
(a) Addressing format



(b) Addressing format (with repeated start condition)



(c) Free data format (master-transmitter to slave-receiver)



Note) S: Start condition
 R/W: Direction bit
 ACK: Acknowledge bit
 P: Stop condition

Fig. 10-2 I²C Bus Mode Data Formats

10.4 Control Registers in the I²C Bus Mode

The following registers control the serial bus interface (SBI) in the I²C bus mode and provide its status information for monitoring.

Serial bus control register 0								
	7	6	5	4	3	2	1	0
SBIxCR0	bit Symbol	SBIEN						
	Read/Write	R/W	R					
	After reset	0	0					
	Function	SBI operation 0: Disable 1: Enable	This can be read as "0."					
	15	14	13	12	11	10	9	8
	bit Symbol							
	Read/Write	R						
	After reset	0						
	Function	This can be read as "0."						
	23	22	21	20	19	18	17	16
	bit Symbol							
	Read/Write	R						
	After reset	0						
	Function	This can be read as "0."						
	31	30	29	28	27	26	25	24
	bit Symbol							
	Read/Write	R						
	After reset	0						
	Function	This can be read as "0."						

<SBIEN>: To use the SBI, enable the SBI operation ("1") before setting each register in the SBI module.

Fig. 10-3 I²C Bus Mode register

SBIXCR1

Serial bus control register 1

	7	6	5	4	3	2	1	0
Bit symbol	BC2	BC1	BC0	ACK		SCK2	SCK1	SCK0/ SWRMON
Read/Write	R/W			R/W	R	R/W		R/W
After reset	0	0	0	0	1	0	0	1
Function	Select the number of bits per transfer (Note 1)			Acknowledgment clock 0: Not generate 1: Generate	This can be read as "1."	Select internal SCL output clock frequency (Note 2) and reset monitor.		
	15	14	13	12	11	10	9	8
Bit symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0."							
	23	22	21	20	19	18	17	16
Bit symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0."							
	31	30	29	28	27	26	25	24
Bit symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0."							

<Bit 2:0><SCK2:0>: Select internal SCL output clock frequency

On writing <SCK2:0>: Select internal SCL output clock frequency

000	n=5	196 kHz	System clock: fsys (=40 MHz) Clock gear : fc/1 Frequency = $\frac{fsys/2}{2^n + 70}$ [Hz]
001	n=6	149 kHz	
010	n=7	101 kHz	
011	n=8	61 kHz	
100	n=9	34 kHz	
101	n=10	18 kHz	reserved
110	n=11	9 kHz	
111		reserved	

<Bit 0>< SWRMON:0>: Software reset status monitor

On reading <SWRMON>: Software reset status monitor

0	Software reset operation is in progress.
1	Software reset operation is not in progress.

<Bit 7:5><BC2:0>: Select the number of bits per transfer

Select the number of bits per transfer

<BC2:0>	When <ACK> = 0		When <ACK> = 1	
	Number of clock cycles	Data length	Number of clock cycles	Data length
000	8	8	9	8
001	1	1	2	1
010	2	2	3	2
011	3	3	4	3
100	4	4	5	4
101	5	5	6	5
110	6	6	7	6
111	7	7	8	7

Fig. 10-4 I²C Bus Mode register

- (Note 1) Clear <BC2:0> to “000” before switching the operation mode to the clock-synchronous 8-bit SIO mode.
- (Note 2) For details on the SCL line clock frequency, refer to “10.5.3 Serial Clock.”
- (Note 3) After a reset, the <SCK0/SWRMON> bit is read as “1.” However, if the SIO mode is selected at the SBIxCR2 register, the initial value of the <SCK0> bit is “0.”

Serial bus control register 2

SBIXCR2

	7	6	5	4	3	2	1	0
bit Symbol	MST	TRX	BB	PIN	SBIM1	SBIM0	SWRST1	SWRST0
Read/Write	W				W (Note 2)		W (Note 1)	
After reset	0	0	0	1	0	0	0	0
Function	Select master/slave 0: Slave 1: Master	Select transmit/receive 0: Receive 1: Transmit	Start/stop condition generation 0: Stop condition generated 1: Start condition generated	Clear INTSBI _n interrupt request 0: – 1: Clear interrupt request	Select serial bus interface operating mode (Note 2) 00: Port mode 01: SIO mode 10: I ² C bus mode 11: (Reserved))		Software reset generation Write “10” followed by “01” to generate a reset.	
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as “0.”							
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as “0.”							
	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as “0.”							

<Bit 1:0><SWRST1:0>: Write “10” followed by “01” to generate a reset.

<Bit 3:2><SBIM1:0> : Select serial bus interface operating mode

Select serial bus interface operating mode (Note 2)

00	Port mode (disables serial bus interface output)
01	Clock –synchronous 8-bit SIO mode
10	I ² C bus mode
11	(Reserved)

<Bit 4><PIN> : Clear INTSBI_n interrupt request

<Bit 5><BB> : Start/stop condition generation

<Bit 6><TRX> : Select transmit/ receive

<Bit 7><MST> : Select master/slave

(Note 1) Reading this register causes it to function as the SBI_nSR register.(Note 2) Ensure that the bus is free before switching the operating mode to the port mode. Ensure that the port is at the “H” level before switching the operating mode from the port mode to the I²C bus or clock-synchronous 8-bit SIO mode.

(Note 3) Ensure that serial transfer is completed before switching the mode.

Fig. 10-5 I²C Bus Mode register

Table 10-1 Base Clock Resolution@f_{sys} = 40 MHz

Clock gear value <GEAR1:0>	Base clock resolution
00 (fc)	$f_{sys}/2^2$ (0.1 μs)
01 (fc/2)	$f_{sys}/2^3$ (0.2 μs)
10 (fc/4)	$f_{sys}/2^4$ (0.4 μs)
11 (fc/8)	$f_{sys}/2^5$ (0.8 μs)

Serial bus interface status register

SBIxSR

	7	6	5	4	3	2	1	0
bit Symbol	MST	TRX	BB	PIN	AL	AAS	ADO	LRB
Read/Write	R							
After reset	0	0	0	1	0	0	0	0
Function	Master/ slave selection monitor 0: Slave 1: Master	Transmit/ receive selection monitor 0: Receive 1: Transmit	I ² C bus state monitor 0: Free 1: Busy	INTSBIIn interrupt request monitor 0: Interrupt request generated 1: Interrupt request cleared	Arbitration lost detection 0: – 1: Detected	Slave address match detection 0: – 1: Detected	General call detection 0: – 1: Detected	Last received bit monitor 0: "0" 1: "1"
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0."							
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0."							
	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0."							

(Note) Writing to this register causes it to function as SBI0CR2.

Fig. 10-6 I²C Bus Mode register

- <Bit 0><LRB> : Last received bit monitor
- <Bit 1><ADO> : General call detection
- <Bit 2><AAS> : Slave address match detection
- <Bit 3><AL> : Arbitration lost detection
- <Bit 4><PIN> : INTSBIIn interrupt request monitor
- <Bit 5><BB> : I²C bus state monitor
- <Bit 6><TRX> : Transmit/ receive selection monitor
- <Bit 7><MST> : Master/ slave selection monitor

Serial bus interface baud rate register 0

SBIxBR0

	7	6	5	4	3	2	1	0
bit Symbol		I2SBI0						
Read/Write	R	R/W	R					R/W
After reset	1	0	1					0
Function	This can be read as "1".	IDLE 0: Stop 1: Operate	This can be read as "1".					Be sure to write "0." (Note)
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							
	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							

<Bit 6><I2SBI0> : Operation at the IDLE mode

(Note) This is read as "1" at the SIO mode.

Serial bus interface data buffer register

SBIxDBR

	7	6	5	4	3	2	1	0
bit Symbol	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Read/Write	R (Receive)/W (Transmit)							
After reset	0							
Function	RX data/ TX data.							
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							
	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							

(Note) The transmission data must be written in to the register from the MSB (bit 7).

SBIxI2CAR

I ² C bus address register								
	7	6	5	4	3	2	1	0
bit Symbol	SA6	SA5	SA4	SA3	SA2	SA1	SA0	ALS
Read/Write	R/W							
After reset	0	0	0	0	0	0	0	0
Function	Set the slave address when the SBI acts as a slave device.							Specify address recognition mode
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							
	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							

<Bit 0><ALS> : Specify address recognition mode

(Note) Please set the bit 0 <ALS> of I²C bus address register SBIxI2CAR to "0", except when you use a free data format. It operates as a free data format when setting it to "1". Selecting the master fixes to transmission. Selecting the slave fixes to reception.

Fig. 10-7 I²C Bus Mode Register

10.5 Control in the I²C Bus Mode

10.5.1 Setting the Acknowledgement Mode

Setting SBIxCR1<ACK> to “1” selects the acknowledge mode. When operating as a master, the SBI adds one clock for acknowledgment signals. As a transmitter, the SBI releases the SDA pin during this clock cycle to receive acknowledgment signals from the receiver. As a receiver, the SBI pulls the SDA pin to the “L” level during this clock cycle and generates acknowledgment signals.

By setting <ACK> to “0”, the non-acknowledgment mode is activated. When operating as a master, the SBI does not generate clock for acknowledgment signals.

10.5.2 Setting the Number of Bits per Transfer

SBIxCR1 <BC2:0> specifies the number of bits of the next data to be transmitted or received.

Under the start condition, <BC2:0> is set to “000,” causing a slave address and the direction bit to be transferred in a packet of eight bits. At other times, <BC2:0> keeps a previously programmed value.

10.5.3 Serial Clock

① Clock source

SBIxCR1 <SCK2:0> specifies the maximum frequency of the serial clock to be output from the SCL pin in the master mode.

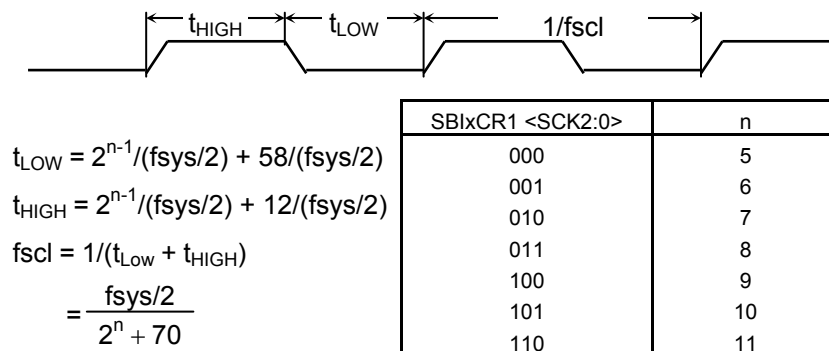


Fig. 10-8 Clock Source

(Note) The highest speeds in the standard and high-speed modes are specified to 100KHz and 400KHz respectively following the communications standards. Note that the internal SCL clock frequency is determined by the f_{sys} used and the calculation formula shown above.

② Clock Synchronization

The I²C bus is driven by using the wired-AND connection due to its pin structure. The first master that pulls its clock line to the “L” level overrides other masters producing the “H” level on their clock lines. This must be detected and responded by the masters producing the “H” level.

Clock synchronization assures correct data transfer on a bus that has two or more master.

For example, the clock synchronization procedure for a bus with two masters is shown below.

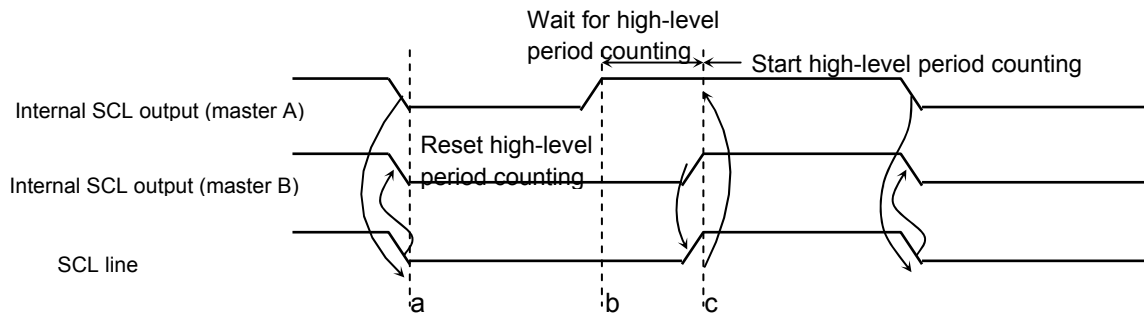


Fig. 10-9 Example of Clock Synchronization

At the point a, Master A pulls its internal SCL output to the “L” level, bringing the SCL bus line to the “L” level. Master B detects this transition, resets its “H” level period counter, and pulls its internal SCL output level to the “L” level.

Master A completes counting of its “L” level period at the point b, and brings its internal SCL output to the “H” level. However, Master B still keeps the SCL bus line at the “L” level, and Master A stops counting of its “H” level period counting. After Master A detects that Master B brings its internal SCL output to the “H” level and brings the SCL bus line to the “H” level at the point c, it starts counting of its “H” level period.

This way, the clock on the bus is determined by the master with the shortest “H” level period and the master with the longest “L” level period among those connected to the bus.

10.5.4 Slave Addressing and Address Recognition Mode

When the SBI is configured to operate as a slave device, the slave address <SA6:0> and <ALS> must be set at SBIxI2CAR. Setting <ALS> to “0” selects the address recognition mode.

10.5.5 Configuring the SBI as a Master or a Slave

Setting SBIxCR2<MST> to “1” configures the SBI to operate as a master device.

Setting <MST> to “0” configures the SBI as a slave device. <MST> is cleared to “0” by the hardware when it detects the stop condition on the bus or the arbitration lost.

10.5.6 Configuring the SBI as a Transmitter or a Receiver

Setting SBIxCR2 <TRX> to “1” configures the SBI as a transmitter. Setting <TRX> to “0” configures the SBI as a receiver.

At the slave mode, the SBI receives the direction bit ($\overline{R/W}$) from the master device on the following occasions:

- when data is transmitted in the addressing format
- when the received slave address matches the value specified at I2CCR
- when a general-call address is received; i.e., the eight bits following the start condition are all zeros

If the value of the direction bit ($\overline{R/W}$) is “1,” <TRX> is set to “1” by the hardware. If the bit is “0,” <TRX> is set to “0”.

As a master device, the SBI receives acknowledgement from a slave device. If the direction bit of “1” is transmitted, <TRX> is set to “0” by the hardware. If the direction bit is “0,” <TRX> changes to “1.” If the SBI does not receive acknowledgement, <TRX> retains the previous value.

<TRX> is cleared to “0” by the hardware when it detects the stop condition on the bus or the arbitration lost.

10.5.7 Generating Start and Stop Conditions

When SBIxSR<BB> is “0,” writing “1” to SBIxCR2 <MST, TRX, BB, PIN> causes the SBI to generate the start condition on the bus and output 8-bit data. <ACK> must be set to “1” in advance.

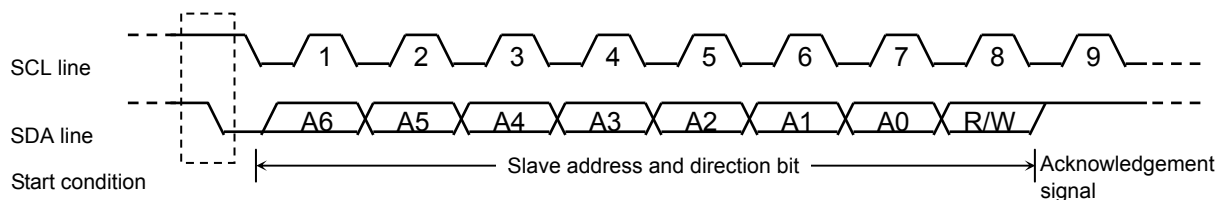


Fig. 10-10 Generating the Start Condition and a Slave Address

When <BB> is “1,” writing “1” to <MST, TRX, PIN> and “0” to <BB> causes the SBI to start a sequence for generating the stop condition on the bus. The contents of <MST, TRX, BB, PIN> should not be altered until the stop condition appears on the bus.

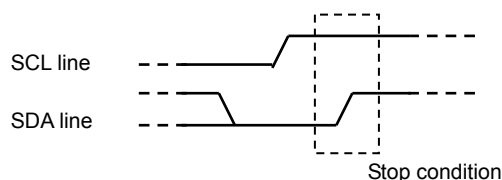


Fig. 10-11 Generating the Stop Condition

SBIxSR<BB> can be read to check the bus state. <BB> is set to “1” when the start condition is detected on the bus (the bus is busy), and set to “0” when the stop condition is detected (the bus is free).

10.5.8 Interrupt Service Request and Release

When a serial bus interface interrupt request (INTSBI0) is generated, SBIxCR2 <PIN> is cleared to "0." While <PIN> is "0," the SBI pulls the SCL line to the "L" level.

After transmission or reception of one data word, <PIN> is cleared to "0." It is set to "1" when data is written to or read from SBIxDBR. It takes a period of t_{LOW} for the SCL line to be released after <PIN> is set to "1."

In the address recognition mode (<ALS> = "0"), <PIN> is cleared to "0" when the received slave address matches the value specified at SBIxI2CAR or when a general-call address is received; i.e., the eight bits following the start condition are all zeros. When the program writes "1" to SBIxCR2<PIN>, it is set to "1." However, writing "0" does clear this bit to "0".

10.5.9 Serial Bus Interface Operating Modes

SBIxCR2 <SBIM1:0> selects an operating mode of the serial bus interface. <SBIM1:0> must be set to "10" to configure the SBI for the I²C bus mode. Make sure that the bus is free before switching the operating mode to the port mode.

10.5.10 Lost-arbitration Detection Monitor

The I²C bus has the multi-master capability (there are two or more masters on a bus), and requires the bus arbitration procedure to ensure correct data transfer.

A master that attempts to generate the start condition while the bus is busy loses bus arbitration, with no start condition occurring on the SDA and SCL lines. The I²C-bus arbitration takes place on the SDA line.

The arbitration procedure for two masters on a bus is shown below. Up until the point a, Master A and Master B output the same data. At the point a, Master A outputs the "L" level and Master B outputs the "H" level. Then Master A pulls the SDA bus line to the "L" level because the line has the wired-AND connection. When the SCL line goes high at the point b, the slave device reads the SDA line data, i.e., data transmitted by Master A. At this time, data transmitted by Master B becomes invalid. This condition of Master B is called "Lost Arbitration". Master B releases its SDA pin, so that it does not affect the data transfer initiated by another master. If two or more masters have transmitted exactly the same first data word, the arbitration procedure continues with the second data word.

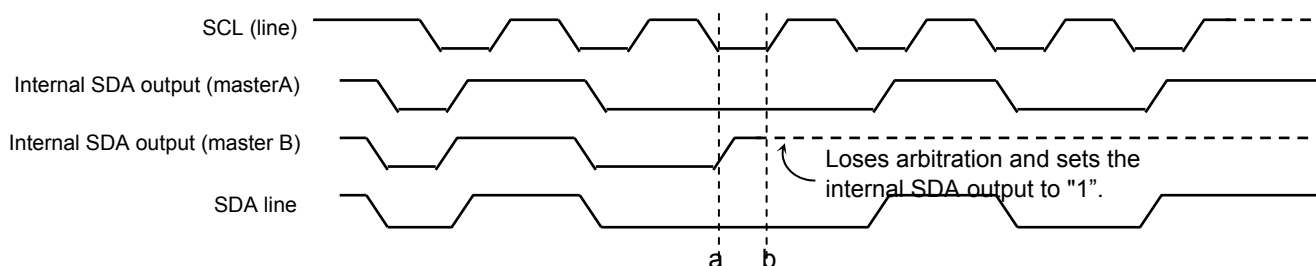


Fig. 10-12 Lost Arbitration

A master compares the SDA bus line level and the internal SDA output level at the rising of the SCL line. If there is a difference between these two values, Arbitration Lost occurs and SBIxSR <AL> is set to "1".

When <AL> is set to "1," SBIxSR <MST, TRX> are cleared to "0," causing the SBI to operate as a slave receiver. <AL> is cleared to "0" when data is written to or read from SBIxDBR or data is written to SBIxCR2.

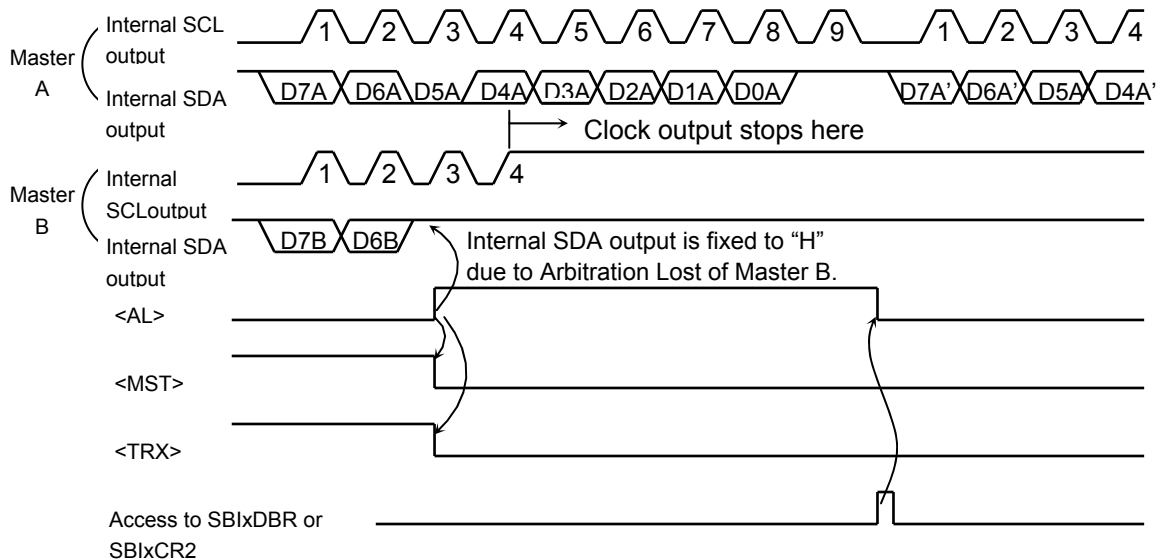


Fig. 10-13 Example of Master B Lost Arbitration (D7A = D7B, D6A = D6B)

10.5.11 Slave Address Match Detection Monitor

When the SBI operates as a slave device in the address recognition mode (SBIxI2CAR <ALS> = "0"), SBIxSR <AAS> is set to "1" on receiving the general-call address or the slave address that matches the value specified at SBIxI2CAR. When <ALS> is "1," <AAS> is set to "1" when the first data word has been received. <AAS> is cleared to "0" when data is written to or read from SBIxDBR.

10.5.12 General-call Detection Monitor

When the SBI operates as a slave device, SBIxSR <AD0> is set to "1" when it receives the general-call address; i.e., the eight bits following the start condition are all zeros. <AD0> is cleared to "0" when the start or stop condition is detected on the bus.

10.5.13 Last Received Bit Monitor

SBIxSR <LRB> is set to the SDA line value that was read at the rising of the SCL line. In the acknowledgment mode, reading SBIxSR <LRB> immediately after generation of the INTSBIx interrupt request causes ACK signal to be read.

10.5.14 Software Reset

If the serial bus interface circuit locks up due to external noise, it can be initialized by using a software reset.

Writing "10" followed by "01" to SBIxCR2 <SWRST1:0> generates a reset signal that initializes the serial bus interface circuit. After a reset, all control registers and status flags are initialized to their reset values. When the serial bus interface is initialized, <SWRST> is automatically cleared to "0".

(Note) A software reset causes the SBI operating mode to switch from the I²C mode to the port mode.

10.5.15 Serial Bus Interface Data Buffer Register (SBIxDBR)

Reading or writing SBIxDBR initiates reading received data or writing transmitted data. When the SBI is acting as a master, setting a slave address and a direction bit to this register generates the start condition.

10.5.16 I²C Bus Address Register (SBIxI2CAR)

When the SBI is configured as a slave device, the SBIxI2CAR<SA6:0> bit is used to specify a slave address. If I2CAR <ALS> is set to "0," the SBI recognizes a slave address transmitted by the master device and receives data in the addressing format. If <ALS> is set to "1," the SBI does not recognize a slave address and receives data in the free data format.

10.5.17 IDLE Setting Register (SBIxBR0)

The SBIxBR0<I2SBI> register determines if the SBI operates or not when it enters the IDLE mode. This register must be programmed before executing an instruction to switch to the standby mode.

10.6 Data Transfer Procedure in the I²C Bus Mode

10.6.1 Device Initialization

First, program SBIxCR1<ACK, SCK2:0> by writing "0" to bits 7 to 5 and bit 3 in SBIxCR1.

Next, program SBIxI2CAR by specifying a slave address at <SA6:0> and an address recognition mode at <ALS>. (<ALS> must be set to "0" when using the addressing format).

Then program SBIxCR2 to initially configure the SBI in the slave receiver mode by writing "0" to <MST, TRX, BB>, "1" to <PIN>, "10" to <SBIM1:0> and "0" to bits 1 and 0.

	7	6	5	4	3	2	1	0	
SBIxCR1	←	0	0	0	X	0	X	X	Specifies ACK and SCL clock.
SBIxI2CAR	←	X	X	X	X	X	X	X	Specifies a slave address and an address recognition mode.
SBIxCR2	←	0	0	0	1	1	0	0	Configures the SBI as a slave receiver.

(Note) X: Don't care

10.6.2 Generating the Start Condition and a Slave Address

① Master mode

In the master mode, the following steps are required to generate the start condition and a slave address.

First, ensure that the bus is free (<BB> = "0"). Then, write "1" to SBIxCR1 <ACK> to select the acknowledgment mode. Write to SBIxDBR a slave address and a direction bit to be transmitted.

When <BB> = "0," writing "1111" to SBIxCR2 <MST, TRX, BB, PIN> generates the start condition on the bus. Following the start condition, the SBI generates nine clocks from the SCL pin. The SBI outputs the slave address and the direction bit specified at SBIxDBR with the first eight clocks, and releases the SDA line in the ninth clock to receive an acknowledgment signal from the slave device.

The INTSBIx interrupt request is generated on the falling of the ninth clock, and <PIN> is cleared to "0." In the master mode, the SBI holds the SCL line at the "L" level while <PIN> is "0." <TRX> changes its value according to the transmitted direction bit at generation of the INTSBIx interrupt request, provided that an acknowledgment signal has been returned from the slave device.

Settings in main routine

	7	6	5	4	3	2	1	0	
→ Reg.	←	S	B	I	S	R			
Reg.	←	Reg. e	0x20						
if Reg.	≠	0x00							Ensures that the bus is free.
Then									
SBIxCR1	←	X	X	X	1	0	X	X	Selects the acknowledgement mode.
SBIxDBR1	←	X	X	X	X	X	X	X	Specifies the desired slave address and direction.
SBIxCR2	←	1	1	1	1	1	0	0	Generates the start condition.

Example of INTSBI0 interrupt routine

Clears the interrupt request.
Processing
End of interrupt

② Slave mode

In the slave mode, the SBI receives the start condition and a slave address.

After receiving the start condition from the master device, the SBI receives a slave address and a direction bit from the master device during the first eight clocks on the SCL line. If the received address matches its slave address specified at SBIXI2CAR or is equal to the general-call address, the SBI pulls the SDA line to the “L” level during the ninth clock and outputs an acknowledgment signal.

The INTSBIX interrupt request is generated on the falling of the ninth clock, and <PIN> is cleared to “0.” In the slave mode, the SBI holds the SCL line at the “L” level while <PIN> is “0”.

(Note) The user can only use a DMA transfer:

- when there is only one master and only one slave and
- continuous transmission or reception is possible.

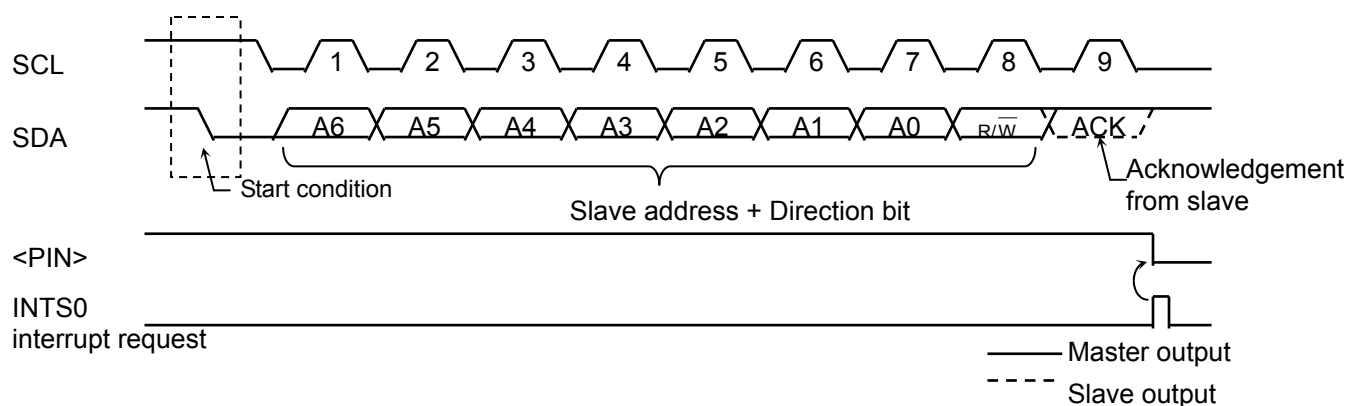


Fig. 10-14 Generation of the Start Condition and a Slave Address

10.6.3 Transferring a Data Word

At the end of a data word transfer, the INTSBIX interrupt is generated to test <MST> to determine whether the SBI is in the master or slave mode.

① Master mode (<MST> = “1”)

Test <TRX> to determine whether the SBI is configured as a transmitter or a receiver.

Transmitter mode (<TRX> = “1”)

Test <LRB>. If <LRB> is “1,” that means the receiver requires no further data. The master then generates the stop condition as described later to stop transmission.

If <LRB> is “0,” that means the receiver requires further data. If the next data to be transmitted has eight bits, the data is written into SBIXDBR. If the data has different length, <BC2:0> and <ACK> are programmed and the transmit data is written into SBIXDBR. Writing the data makes <PIN> to “1,” causing the SCL pin to generate a serial clock for transferring a next data word, and the SDA pin to transfer the data word. After the transfer is completed, the INTSBIX interrupt request is generated, <PIN> is set to “0,” and the SCL pin is pulled to the “L” level. To transmit more data words, test <LRB> again and repeat the above procedure.

INTSB_lx interrupt

if MST = 0

Then go to the slave-mode processing

if TRX = 0

Then go to the receiver-mode processing

if LRB = 0

Then go to processing for generating the stop condition

SBlxCR1 ← X X X X 0 X X X

Specifies the number of bits to be transmitted and specify whether ACK is required.

SBlxDBR ← X X X X X X X X

Writes the transmit data.

End of interrupt processing

(Note) X: Don't care

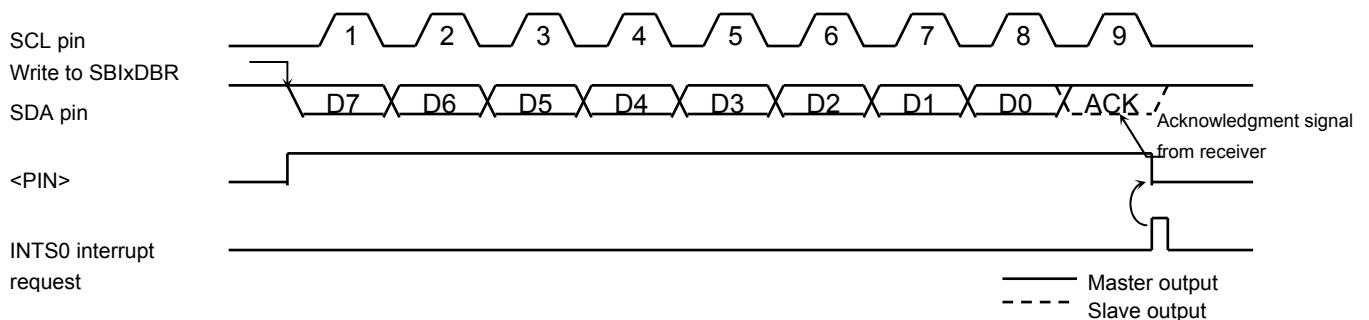


Fig. 10-15 <BC2:0> = "000" and <ACK> = "1" (Transmitter Mode)

Receiver mode (<TRX> = "0")

If the next data to be transmitted has eight bits, the transmit data is written into SBlxDBR. If the data has different length, <BC2:0> and <ACK> are programmed and the received data is read from SBlxDBR to release the SCL line. (The data read immediately after transmission of a slave address is undefined.) On reading the data, <PIN> is set to "1," and the serial clock is output to the SCL pin to transfer the next data word. In the last bit, when the acknowledgment signal becomes the "L" level, "0" is output to the SDA pin.

After that, the INTSB_lx interrupt request is generated, and <PIN> is cleared to "0," pulling the SCL pin to the "L" level. Each time the received data is read from SBlxDBR, one-word transfer clock and an acknowledgement signal are output.

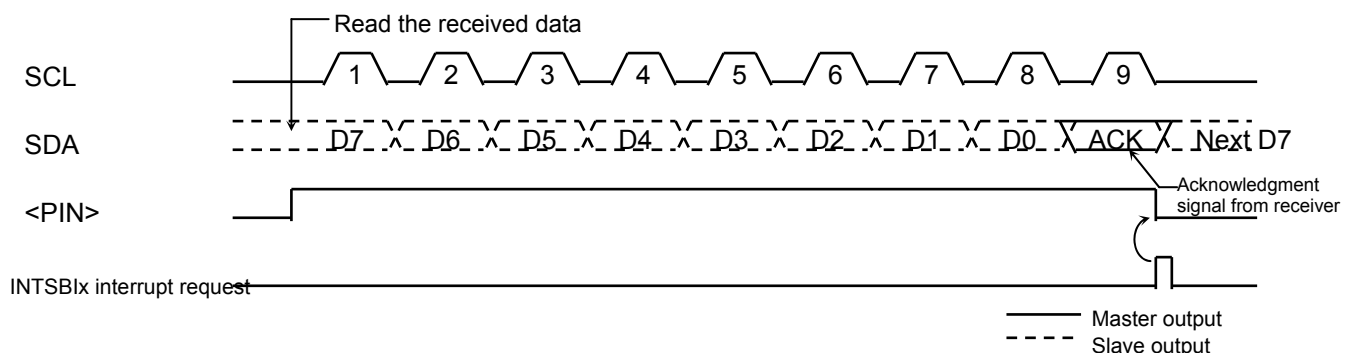


Fig. 10-16 <BC2:0> = "000" and <ACK> = "1" (Receiver Mode)

To terminate the data transmission from the transmitter, <ACK> must be set to "0" immediately before reading the data word second to last. This disables generation of an acknowledgment clock for the last data word. When the transfer is completed, an interrupt request is generated. After the interrupt processing, <BC2:0> must be set to "001" and the data must be read so that a clock is generated for 1-bit transfer. At this time, the master receiver holds the SDA bus line at the "H" level, which signals the end of transfer to the transmitter as an acknowledgment signal.

In the interrupt processing for terminating the reception of 1-bit data, the stop condition is generated to terminate the data transfer.

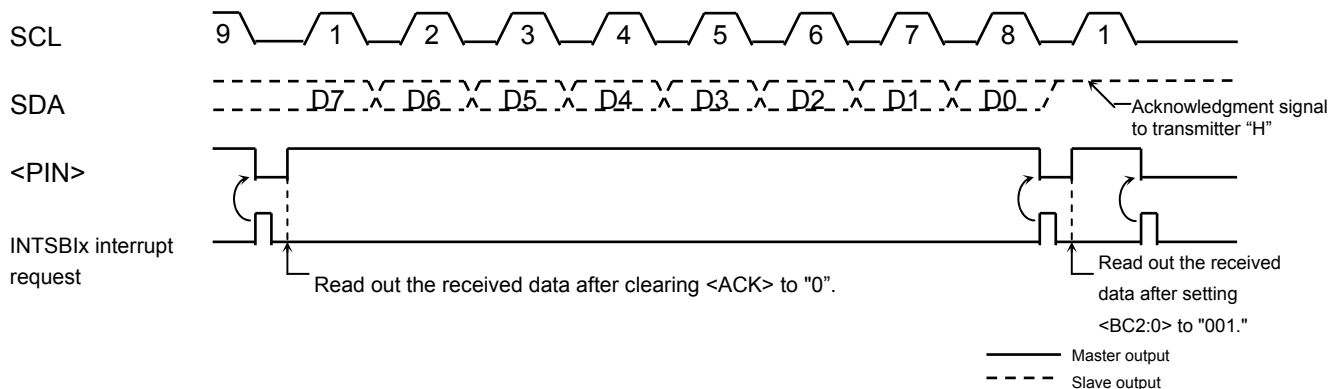


Fig. 10-17 Terminating Data Transmission in the Master Receiver Mode

Example: When receiving N data word

INTSB_{IX} interrupt (after data transmission)

	7	6	5	4	3	2	1	0
SBI _{IX} CR1	←	X	X	X	X	0	X	X
Reg.	←	SBI0DBR						
End of interrupt								

Sets the number of bits of data to be received and specify whether ACK is required.
Reads dummy data.

INTSB_{IX} interrupt (first to (N-2)th data reception)

	7	6	5	4	3	2	1	0
Reg.	←	SBI _{IX} DBR						
End of interrupt								

Reads the first to (N-2)th data words.

INTSB_{IX} interrupt ((N-1)th data reception)

	7	6	5	4	3	2	1	0
SBI _{IX} CR1	←	X	X	X	0	0	X	X
Reg.	←	SBI _{IX} DBR						
End of interrupt								

Disables generation of acknowledgement clock.
Reads the (N-1)th data word.

INTSB_{IX} interrupt (Nth data reception)

	7	6	5	4	3	2	1	0
SBI _{IX} CR1	←	0	0	1	0	0	X	X
Reg.	←	SBI _{IX} DBR						
End of interrupt								

Disables generation of acknowledgement clock.
Reads the Nth data word.

INTSB_{IX} interrupt (after completing data reception)

Processing to generate the stop condition. Terminates the data transmission.
End of interrupt

(Note) X: Don't care

② Slave mode (<MST> = "0")

In the slave mode, the SBI generates the INTSBIx interrupt request on four occasions: 1) when the SBI has received any slave address from the master, 2) when the SBI has received a general-call address, 3) when the received slave address matches its own address, and 4) when a data transfer has been completed in response to a general-call. Also, if the SBI detects Arbitration Lost in the master mode, it switches to the slave mode. Upon the completion of data word transfer in which Arbitration Lost is detected, the INTSBIx interrupt request is generated, <PIN> is cleared to "0," and the SCL pin is pulled to the "L" level. When data is written to or read from SBIxDBR or when <PIN> is set to "1," the SCL pin is released after a period of t_{LOW} .

In the slave mode, the normal slave mode processing or the processing as a result of Arbitration Lost is carried out.

SBIxSR <AL>, <TRX>, <AAS> and <AD0> are tested to determine the processing required. Table 15-19 shows the slave mode states and required processing.

Example: When the received slave address matches the SBI's own address and the direction bit is "1" in the slave receiver mode.

INTSBIx interrupt

```

if TRX = 0
  Then go to other processing
if AL = 1
  Then go to other processing
if AAS = 0
  Then go to other processing
SBIxCR1 ← X X X 1 0 X X X   Sets the number of bits to be transmitted.
SBIxDBR ← X X X X 0 X X X   Sets the transmit data.

```

(Note) X: Don't care

Table 10-2 Processing in Slave Mode

<TRX>	<AL>	<AAS>	<AD0>	State	Processing
1	1	1	0	Arbitration Lost is detected while the slave address was being transmitted and the SBI received a slave address with the direction bit "1" transmitted by another master.	Set the number of bits in a data word to <BC2:0> and write the transmit data into SBIDBR.
	0	1	0	In the slave receiver mode, the SBI received a slave address with the direction bit "1" transmitted by the master.	
		0	0	In the slave transmitter mode, the SBI has completed a transmission of one data word.	Test LRB. If it has been set to "1," that means the receiver does not require further data. Set <PIN> to 1 and reset <TRX> to 0 to release the bus. If <LRB> has been reset to "0," that means the receiver requires further data. Set the number of bits in the data word to <BC2:0> and write the transmit data to the SBIDBR.
0	1	1	1/0	Arbitration Lost is detected while a slave address is being transmitted, and the SBI receives either a slave address with the direction bit "0" or a general-call address transmitted by another master.	Read the SBIDBR (a dummy read) to set <PIN> to 1, or write "1" to <PIN>.
		0	0	Arbitration Lost is detected while a slave address or a data word is being transmitted, and the transfer is terminated.	
	0	1	1/0	In the slave receiver mode, the SBI received either a slave address with the direction bit "0" or a general-call address transmitted by the master.	
		0	1/0	In the slave receiver mode, the SBI has completed a reception of a data word.	Set the number of bits in the data word to <BC2:0> and read the received data from SBIDBR.

10.6.4 Generating the Stop Condition

When SBIxSR <BB> is "1," writing "1" to SBIxCR2 <MST, TRX, PIN> and "0" to <BB> causes the SBI to start a sequence for generating the stop condition on the bus. Do not alter the contents of <MST, TRX, BB, PIN> until the stop condition appears on the bus.

If another device is holding down the SCL bus line, the SBI waits until the SCL line is released. After that, the SDA pin goes high, causing the stop condition to be generated.

7 6 5 4 3 2 1 0
 SBIxCR2 ← 1 1 0 1 1 0 0 0 Generates the stop condition.

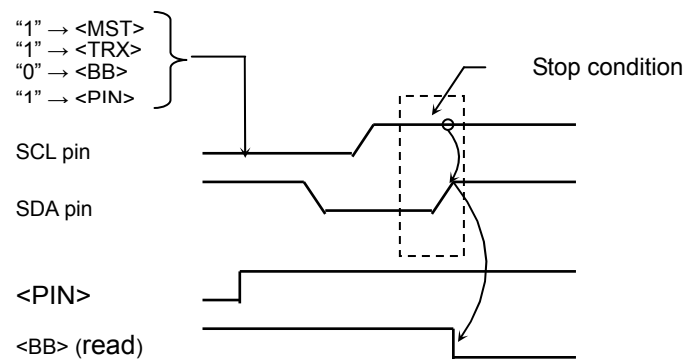


Fig. 10-18 Generating the Stop Condition

10.6.5 Restart Procedure

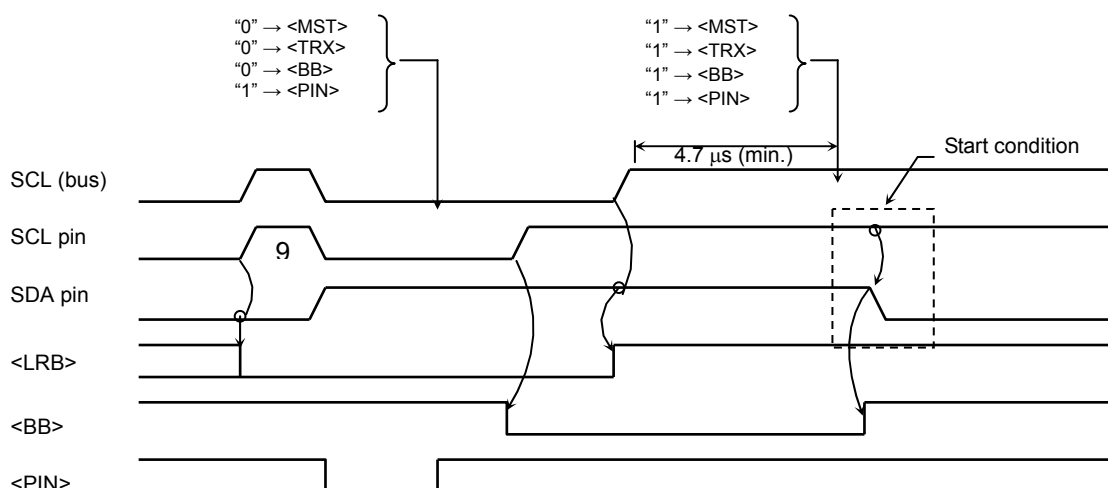
Restart is used when a master device changes the data transfer direction without terminating the transfer to a slave device. The procedure of generating a restart in the master mode is described below.

First, set SBIxCR2 <MST, TRX, BB> to "0" and write "1" to <PIN> to release the bus. At this time, the SDA pin is held at the "H" level and the SCL pin is released. Because no stop condition is generated on the bus, other devices recognize that the bus is busy. Then, test SBIxSR <BB> and wait until it becomes "0" to ensure that the SCL pin is released. Next, test <LRB> and wait until it becomes "1" to ensure that no other device is pulling the SCL bus line to the "L" level. Once the bus is determined to be free this way, use the above-mentioned steps 10.6.2 to generate the start condition.

To satisfy the setup time of restart, at least 4.7- μ s wait period (in the standard mode) must be created by the software after the bus is determined to be free.

	7 6 5 4 3 2 1 0	
SBIxCR2	← 0 0 0 1 1 0 0 0	Releases the bus.
if SBIxSR<BB> ≠ 0		Checks that the SCL pin is released.
Then		
if SBIxSR<LRB> ≠ 1		Checks that no other device is pulling the SCL pin to the "L" level.
Then		
4.7 μ s Wait		
SBIxCR1	← X X X 1 0 X X X	Selects the acknowledgment mode.
SBIxDBR	← X X X X X X X X	Sets the desired slave address and direction.
SBIxCR2	← 1 1 1 1 1 0 0 0	Generates the start condition.

(Note) X: Don't care



(Note) Do not write <MST> to "0" when it is "0." (Restart cannot be initiated.)

Fig. 10-19 Timing Chart of Generating a Restart

10.7 Control in the Clock-synchronous 8-bit SIO Mode

The following registers control the serial bus interface in the clock-synchronous 8-bit SIO mode and provide its status information for monitoring.

Serial bus interface control register 0								
	7	6	5	4	3	2	1	0
SBIxCR0	bit Symbol	SBIEN						
	Read/Write	R/W						
	After reset	0						
	Function	SBI operation 0: Disable 1: Enable						
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0."							
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0."							
	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0."							

<SBIEN>: To use the SBI, enable the SBI operation ("1") before setting each register of SBI module.

Fig. 10-20 SIO Mode Registers

Serial bus interface control register 1

SBIXCR1

	7	6	5	4	3	2	1	0
bit Symbol	SIOS	SIOINH	SIOM1	SIOM0		SCK2	SCK1	SCK0
Read/Write	W				R	W		R/W
After reset	0	0	0	0	1	0	0	0
Function	Start transfer 0: Stop 1: Start	Transfer 0: Continue 1: Forced termination	Select transfer mode 00: Transmit mode 01: (Reserved) 10: Transmit/receive mode 11: Receive mode		This can be read as "1".	Select serial clock frequency		
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							
	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							

On writing <SCK2:0>: Select serial clock frequency

000	n = 3	1.25 MHz	$\left(\begin{array}{ll} \text{System clock} & : f_{\text{sys}} \\ & (=40 \text{ MHz}) \\ \text{Clock gear} & : f_c/1 \\ \text{Frequency} & : \frac{f_{\text{sys}}/4}{2^n} \text{ [Hz]} \end{array} \right)$
001	n = 4	625 kHz	
010	n = 5	313 kHz	
011	n = 6	156 kHz	
100	n = 7	78 kHz	
101	n = 8	39 kHz	
110	n = 9	20 kHz	
111	—	External clock	

(Note) Set <SIOS> to "0" and <SIOINH> to "1" before programming the transfer mode and the serial clock.

Fig. 10-21 SIO Mode Registers

Serial bus interface data buffer register

SBIDBR

	7	6	5	4	3	2	1	0
bit Symbol	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Read/Write	R (Receive)/W (Transmit)							
After reset	Undefined							
Function	RX data/ TX data							
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R							
リセット後	0							
機能	This can be read as "0".							
	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							

Serial bus interface control register 2

SBICR2

	7	6	5	4	3	2	1	0
bit Symbol					SBIM1	SBIM0		
Read/Write	R				W		R	
After reset	1				0	0	1	
Function	This can be read as "1".				Select serial bus interface operating mode 00: Port mode 01: Clock-synchronous 8-bit SIO mode 10: I ² C bus mode 11: (Reserved)		This can be read as "1".	
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							
	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							

Fig. 10-22 SIO Mode Registers

Serial bus interface register

SBIXSR

	7	6	5	4	3	2	1	0
bit Symbol					SIOF	SEF		
Read/Write	R				R		R	
After reset	1				0	0	1	
Function	This can be read as "1".				Serial transfer status monitor 0: Completed 1: In progress	Shift operation status monitor 0: Completed 1: In progress	This can be read as "1".	
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							
	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0".							

Serial bus interface baud rate register 0

SBIXBR0

	7	6	5	4	3	2	1	0
bit Symbol		I2SBI						
Read/Write	R	R/W	R					W
After reset	1	0	1					0
Function	This can be read as "1".	IDLE 0: Stop 1: Operate	This can be read as "1".					Make sure to write "0."
	15	14	13	12	11	10	9	8
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0"							
	23	22	21	20	19	18	17	16
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0"							
	31	30	29	28	27	26	25	24
bit Symbol								
Read/Write	R							
After reset	0							
Function	This can be read as "0"							

Fig. 10-23 SIO Mode Registers

10.7.1 Serial Clock

① Clock source

Internal or external clocks can be selected by programming SBlxCR1 <SCK2:0>.

Internal clocks

In the internal clock mode, one of the seven frequencies can be selected as a serial clock, which is output to the outside through the SCK pin. At the beginning of a transfer, the SCK pin output becomes the “H” level.

If the program cannot keep up with this serial clock rate in writing the transmit data or reading the received data, the SBI automatically enters a wait period. During this period, the serial clock is stopped automatically and the next shift operation is suspended until the processing is completed.

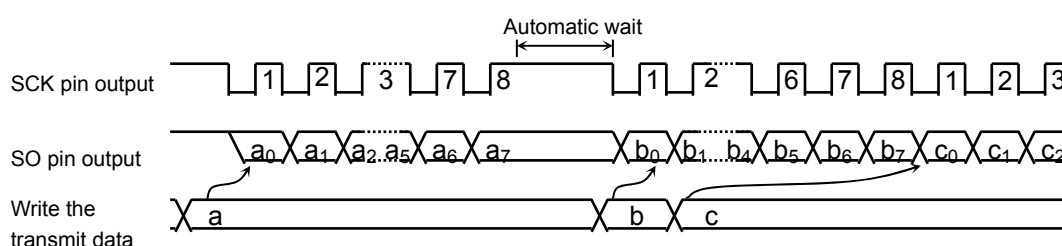


Fig. 10-24 Automatic Wait

External clock (<SCK2:0> = "111")

The SBI uses an external clock supplied from the outside to the SCK pin as a serial clock. For proper shift operations, the serial clock at the “H” and “L” levels must have the pulse widths as shown below.

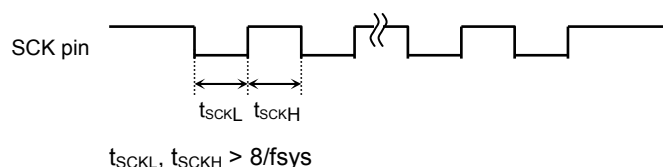


Fig. 10-25 Maximum Transfer Frequency of External Clock Input

② Shift Edge

Leading-edge shift is used in transmission. Trailing-edge shift is used in reception.

Leading-edge shift

Data is shifted at the leading edge of the serial clock (or the falling edge of the SCK pin input/output).

Trailing-edge shift

Data is shifted at the trailing edge of the serial clock (or the rising edge of the SCK pin input/output).

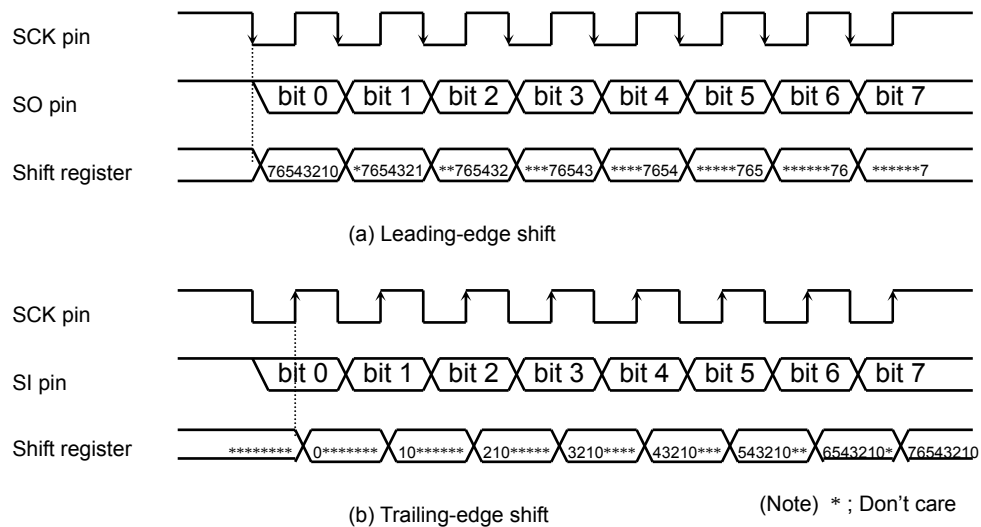


Fig. 10-26 Shift Edge

10.7.2 Transfer Modes

The transmit mode, the receive mode or the transmit/receive mode can be selected by programming SBIXCR1 <SIOM1:0>.

① 8-bit transmit mode

Set the control register to the transmit mode and write the transmit data to SBIXDBR.

After writing the transmit data, writing "1" to SBIXCR1 <SIOS> starts the transmission. The transmit data is moved from SBIXDBR to a shift register and output to the SO pin, with the least-significant bit (LSB) first, in synchronization with the serial clock. Once the transmit data is transferred to the shift register, SBIXDBR becomes empty, and the INTSBIX (buffer-empty) interrupt is generated, requesting the next transmit data.

In the internal clock mode, the serial clock will be stopped and automatically enter the wait state, if next data is not loaded after the 8-bit data has been fully transmitted. The wait state will be cleared when SBIXDBR is loaded with the next transmit data.

In the external clock mode, SBIXDBR must be loaded with data before the next data shift operation is started. Therefore, the data transfer rate varies depending on the maximum latency between when the interrupt request is generated and when SBIXDBR is loaded with data in the interrupt service program.

At the beginning of transmission, the same value as in the last bit of the previously transmitted data is output in a period from setting SBIXSR <SIOF> to "1" to the falling edge of SCK.

Transmission can be terminated by clearing <SIOS> to "0" or setting <SIOINH> to "1" in the INTSBIX interrupt service program. If <SIOS> is cleared, remaining data is output before transmission ends. The program checks SBIXSR <SIOF> to determine whether transmission has come to an end. <SIOF> is cleared to "0" at the end of transmission. If <SIOINH> is set to "1," the transmission is aborted immediately and <SIOF> is cleared to "0".

In the external clock mode, <SIOS> must be set to "0" before the next transmit data shift operation is started. Otherwise, operation will stop after dummy data is transmitted.

	7 6 5 4 3 2 1 0	
SBIXCR1	← 0 1 0 0 0 X X X	Selects the transmit mode.
SBIXDBR	← X X X X X X X X	Writes the transmit data.
SBIXCR1	← 1 0 0 0 0 X X X	Starts transmission.

INTSBIX interrupt

SBIXDBR	← X X X X X X X X	Writes the transmit data.

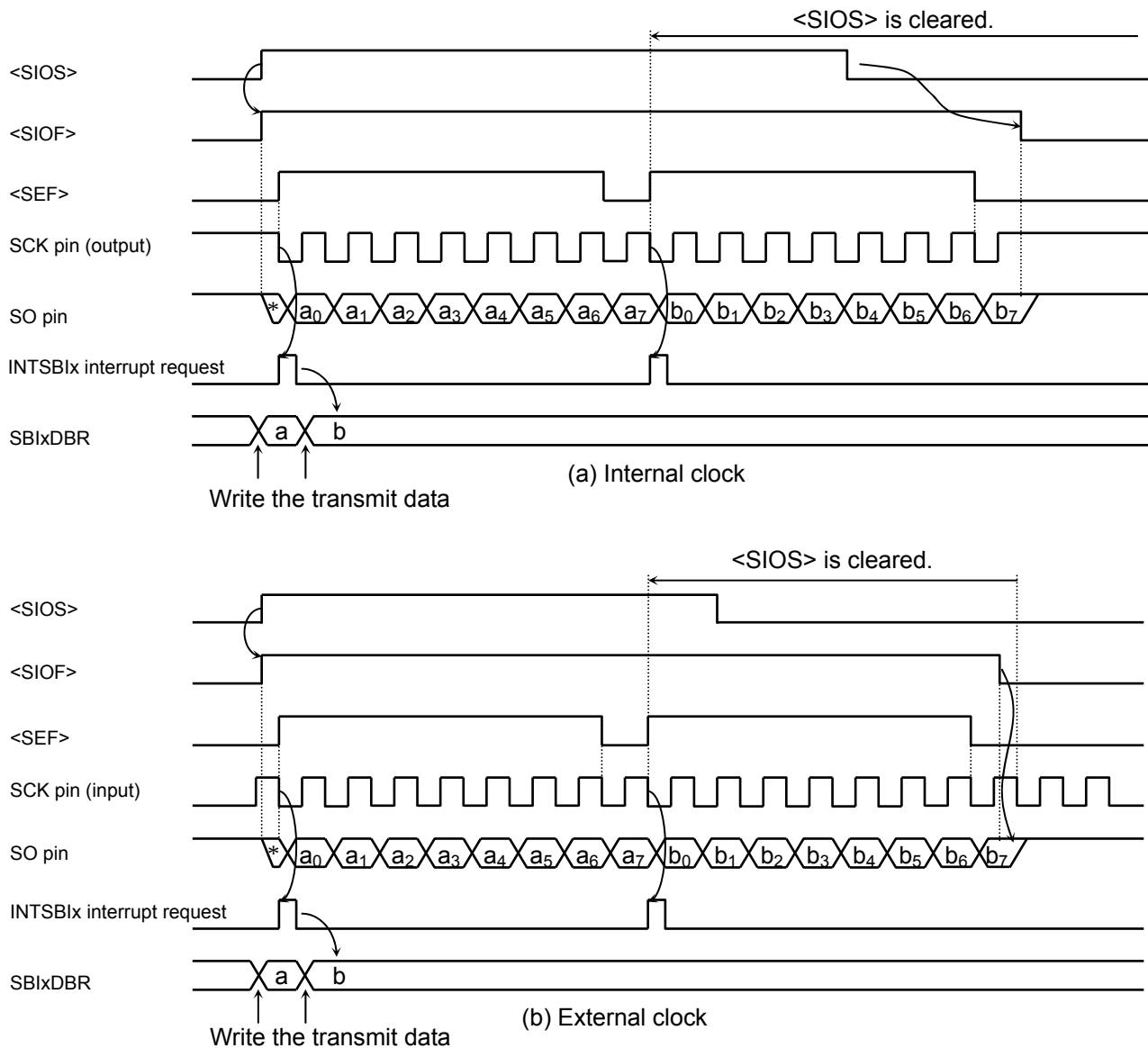


Fig. 10-27 Transmit Mode

Example: Example of programming (external clock) to terminate transmission by <SIO>

	7	6	5	4	3	2	1	0	
→ if SBISR<SIOF> ≠ 0									Recognizes the completion of the transmission.
Then									
→ if SCK ≠ 1									Recognizes "1" is set to the SCK pin by monitoring the port.
Then									
SBICR1 ←	0	0	0	0	0	1	1	1	Completes the transmission by setting <SIOF> = 0.

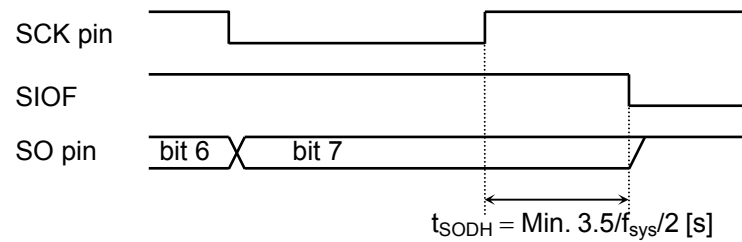


Fig. 10-28 Transmit Data Retention Time at the End of Transmission

② 8-bit receive mode

Set the control register to the receive mode. Then writing “1” to SBIXCR1 <SIOS> enables reception. Data is taken into the shift register from the SI pin, with the least-significant bit (LSB) first, in synchronization with the serial clock. Once the shift register is loaded with the 8-bit data, it transfers the received data to SBIXDBR and the INTSBIX (buffer-full) interrupt request is generated to request reading the received data. The interrupt service program then reads the received data from SBIXDBR.

In the internal clock mode, the serial clock will be stopped and automatically be in the wait state until the received data is read from SBIXDBR.

In the external clock mode, shift operations are executed in synchronization with the external clock. The maximum data transfer rate varies, depending on the maximum latency between generating the interrupt request and reading the received data.

Reception can be terminated by clearing <SIOS> to “0” or setting <SIOINH> to “1” in the INTSBIX interrupt service program. If <SIOS> is cleared, reception continues until all the bits of received data are written to SBIXDBR. The program checks SBIXSR <SIOF> to determine whether reception has come to an end. <SIOF> is cleared to “0” at the end of reception. After confirming the completion of the reception, last received data is read. If <SIOINH> is set to “1,” the reception is aborted immediately and <SIOF> is cleared to “0.” (The received data becomes invalid, and there is no need to read it out.)

(Note) The contents of SBIXDBR will not be retained after the transfer mode is changed. The ongoing reception must be completed by clearing <SIOS> to “0” and the last received data must be read before the transfer mode is changed.

7 6 5 4 3 2 1 0	
SBIXCR1 ← 0 1 1 1 0 X X X	Selects the receive mode.

SBIXCR1 ← 1 0 1 1 0 0 0 0	Starts reception.

INTSBIX interrupt

Reg. ← SBIXDBR	Reads the received data.

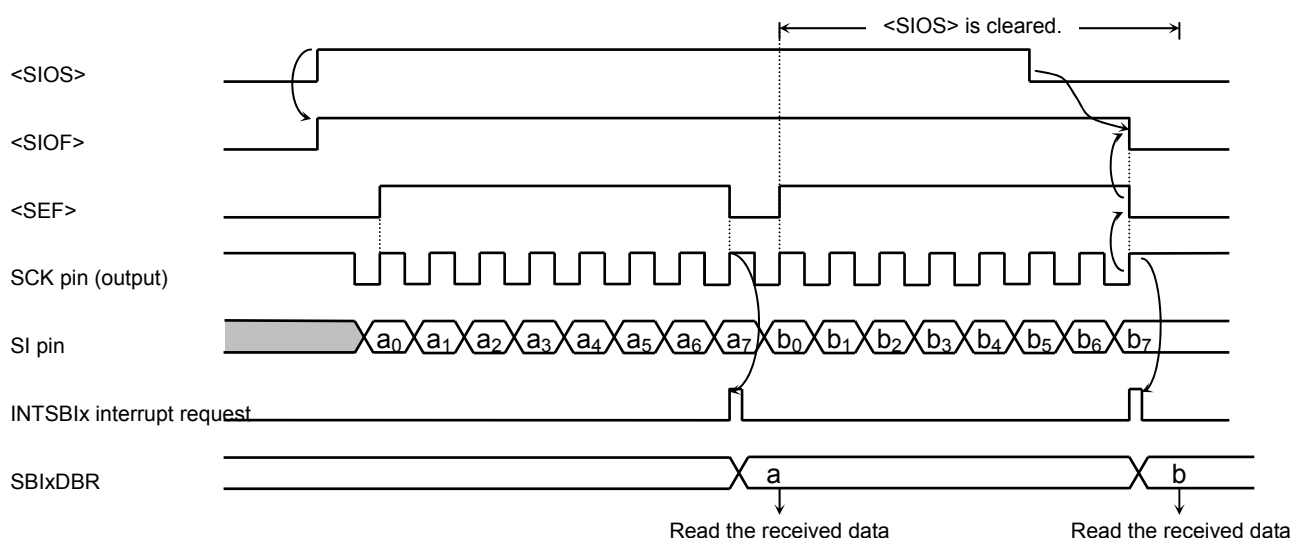


Fig. 10-29 Receive Mode (Example: Internal Clock)

③ 8-bit transmit/receive mode

Set the control register to the transfer/receive mode. Then writing the transmit data to SBIXDBR and setting SBIXCR1 <SIOF> to "1" enables transmission and reception. The transmit data is output through the SO pin at the falling of the serial clock, and the received data is taken in through the SI pin at the rising of the serial clock, with the least-significant bit (LSB) first. Once the shift register is loaded with the 8-bit data, it transfers the received data to SBIXDBR and the INTSBIx interrupt request is generated. The interrupt service program reads the received data from the data buffer register and writes the next transmit data. Because SBIXDBR is shared between transmit and receive operations, the received data must be read before the next transmit data is written.

In the internal clock operation, the serial clock will be automatically in the wait state until the received data is read and the next transmit data is written.

In the external clock mode, shift operations are executed in synchronization with the external serial clock. Therefore, the received data must be read and the next transmit data must be written before the next shift operation is started. The maximum data transfer rate for the external clock operation varies depending on the maximum latency between when the interrupt request is generated and when the transmit data is written.

At the beginning of transmission, the same value as in the last bit of the previously transmitted data is output in a period from setting <SIOF> to "1" to the falling edge of SCK. Transmission and reception can be terminated by clearing <SIOF> to "0" or setting SBIXCR1 <SIOINH> to "1" in the INTSBIx interrupt service program. If <SIOF> is cleared, transmission and reception continue until the received data is fully transferred to SBIXDBR. The program checks SBIXSR <SIOF> to determine whether transmission and reception have come to an end. <SIOF> is cleared to "0" at the end of transmission and reception. If <SIOINH> is set, the transmission and reception are aborted immediately and <SIOF> is cleared to "0."

(Note) The contents of SBIXDBR will not be retained after the transfer mode is changed. The ongoing transmission and reception must be completed by clearing <SIOF> to "0" and the last received data must be read before the transfer mode is changed.

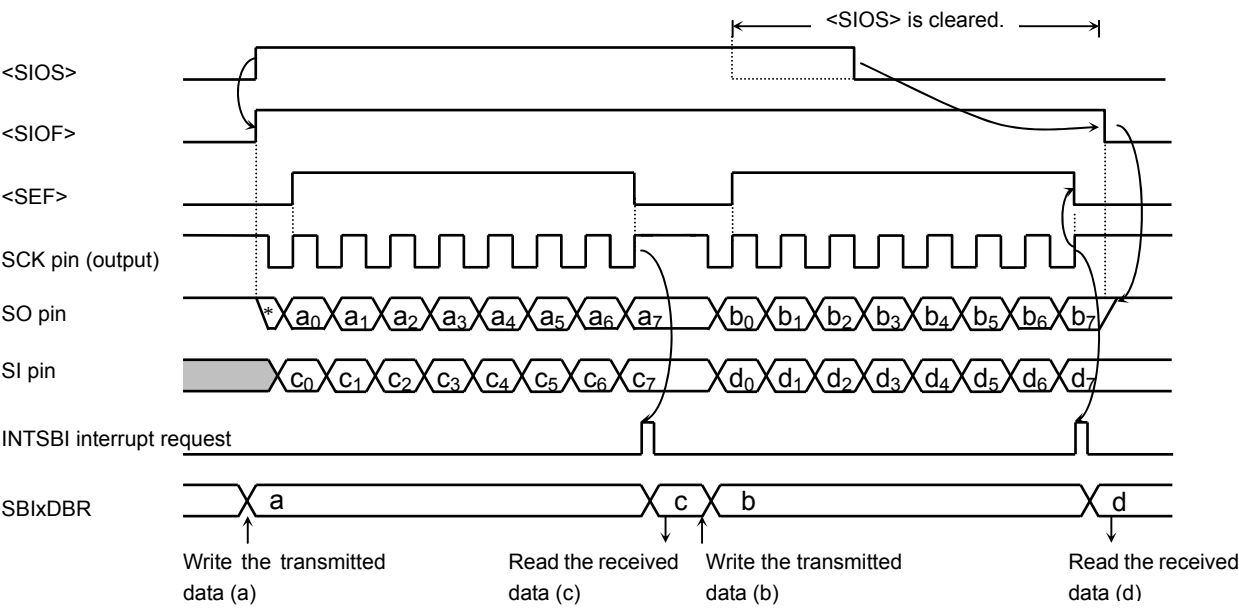


Fig. 10-30 Transmit/Receive Mode (Example: Internal Clock)

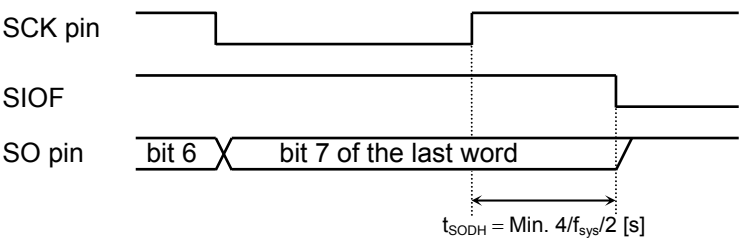


Fig. 10-31 Transmit Data Retention Time at the End of Transmission/Reception (In the Transmit/Receive Mode)

	7	6	5	4	3	2	1	0	
SBIXCR1	←	0	1	1	0	0	X	X	X
									Selects the transmit mode.
SBIXDBR	←	X	X	X	X	X	X	X	X
SBIXCR1	←	1	0	1	0	0	X	X	X
									Writes the transmit data.
									Starts reception/transmission.
INTSBIx interrupt									
Reg.	←	SBIXDBR							Reads the received data.
SBIXDBR	←	X	X	X	X	X	X	X	X
									Writes the transmit data.

11 Consumer Electronics Control (CEC)

11.1 Outline

This IP enables to transmit or receive data that conforms to Consumer Electronics Control (hereafter referred to as CEC) protocol.

11.1.1 Reception

- Clock sampling at 32KHz
 - Adjustable noise canceling time
- Data reception per 1byte
 - Flexible data sampling point
 - Data reception is available even when an address discrepancy is detected.
- Error detection
 - Cycle error (min./ max.)
 - ACK collision
 - Waveform error

11.1.2 Transmission

- Data transmission per 1byte
 - Triggered by auto-detection of bus free state
- Flexible waveform
 - Adjustable rising edge and cycle
- Error detection
 - Arbitration lost
 - ACK response error

11.2 Registers

11.2.1 Control Registers and Addresses

The control registers and address for CEC are as follows.

Registers		Addresses
CEC Enable Register	CECEN	0x4004_0300
Logical Address Register	CECADD	0x4004_0304
Software Reset Register	CECRESET	0x4004_0308
Receive Enable Register	CECREN	0x4004_030C
Receive Buffer Register	CECRBUF	0x4004_0310
Receive Control Register 1	CECR1	0x4004_0314
Receive Control Register 2	CECR2	0x4004_0318
Receive Control Register 3	CECR3	0x4004_031C
Transmit Enable Register	CECTEN	0x4004_0320
Transmit Buffer Register	CECTBUF	0x4004_0324
Transmit Control Register	CECTCR	0x4004_0328
Receive Interrupt Status Register	CECRSTAT	0x4004_032C
Transmit Interrupt Status Register	CECTSTAT	0x4004_0330

11.2.2 Register Map

		31	24	23	16	15	8	7	0
CECEN	0x4004_0300								
CECADD	0x4004_0304								
CECRESET	0x4004_0308								
CECREN	0x4004_030C								
CECRBUF	0x4004_0310								
CECR1	0x4004_0314								
CECR2	0x4004_0318								
CECR3	0x4004_031C								
CECTEN	0x4004_0320								
CECTBUF	0x4004_0324								
CECTCR	0x4004_0328								
CECRSTAT	0x4004_032C								
CECTSTAT	0x4004_0330								

11.2.3 CEC Enable Register [CECEN]

	7	6	5	4	3	2	1	0
bit Symbol	—						I2CEC	CECEN
Read/Write	R						R/W	R/W
After reset	0						0	0
Function	"0" is read.						CEC operation at IDLE 0: Disabled 1: Enabled	CEC operation 0: Disabled 1: Enabled

<I2CEC>: Controls the CEC operation at the IDLE mode.
Set this bit to "1" when using CEC at the IDLE mode.
The <I2CEC> and <CECEN> bits can be set simultaneously.

<CECEN>: Specifies the CEC operation.
Enable CEC before using.
When the CEC operation is disabled, no clocks are supplied to the CEC module except for the enable register. Thus power consumption can be reduced.
When CEC is disabled after it was enabled, each register setting is maintained.

11.2.4 Logical Address Register [CECADD]

	15	14	13	12	11	10	9	8
bit Symbol	CECADD 15	CECADD 14	CECADD 13	CECADD 12	CECADD 11	CECADD 10	CECADD 9	CECADD 8
Read/Write	R/W							
After reset	0							
Function	Logical address 15	Logical address 14	Logical address 13	Logical address 12	Logical address 11	Logical address 10	Logical address 9	Logical address 8

	7	6	5	4	3	2	1	0
bit Symbol	CECADD 7	CECADD 6	CECADD 5	CECADD 4	CECADD 3	CECADD 2	CECADD 1	CECADD 0
Read/Write	R/W							
After reset	0							
Function	Logical address 7	Logical address 6	Logical address 5	Logical address 4	Logical address 3	Logical address 2	Logical address 1	Logical address 0

<CECADD15:0>: Specifies the logical address assigned to CEC.
Multiple addresses can be set simultaneously since each bit corresponds with each address.

(Note) A broadcast message is received regardless of the register setting.
By allocating a logical address of a device to 15, logical "0" is sent as an ACK response to the broadcast message.

11.2.5 Software Reset Register [CECRESET]

	7	6	5	4	3	2	1	0
bit Symbol	—							CEC RESET
Read/Write	R							W
After reset	0							0
Function	"0" is read.							Software reset 0: Disabled 1: Enabled "0" is read.

<CECRESET>: Stops all the CEC operation and initializes the register.
Setting this bit to "1" affects as follows:
Reception: Stops immediately. The received data is discarded.
Transmission (including the CEC line): Stops immediately.
Register: All the registers other than CECEN are initialized.

11.2.6 Receive Enable Register [CECREN]

	7	6	5	4	3	2	1	0
bit Symbol	—							CECREN
Read/Write	R							R/W
After reset	0							0
Function	"0" is read.							Reception control [Write] 0: Disabled 1: Enabled [Read] 0: Stopped 1: in operation

<CECREN>: Controls the reception operation of CEC.
 Writing "0" or "1" to this bit enables or disables data reception. This bit becomes ready for data reception by writing "1".
 The state of the reception circuit is monitored by reading this bit. It enables you to check if what you set has properly been reflected.

(Note 1) Enable the <CECREN> bit after setting the reception control register 1, 2 and 3.

(Note 2) It takes a little time to reflect the setting of the <CECREN> bit to the circuit. Stop transmission and reception before changing the settings or enabling the transmission and reception.

11.2.7 Receive Buffer Register [CECRBUF]

	15	14	13	12	11	10	9	8
bit Symbol	—						CECACK	CECEOM
Read/Write	R						R	R
After reset	0						0	0
Function	"0" is read.						ACK bit	EOM bit
	7	6	5	4	3	2	1	0
bit Symbol	CECRBUF 7	CECRBUF 6	CECRBUF 5	CECRBUF 4	CECRBUF 3	CECRBUF 2	CECRBUF 1	CECRBUF 0
Read/Write	R							
After reset	0							
Function	Received data							

<CECACK>: Reads the received ACK bit.

<CECEOM>: Reads the received EOM bit.

<CECRBUF7:0>: Reads one byte of data received. The bit 7 is the MSB.

(Note 1) Writing to this register is ignored.

(Note 2) Read this register as soon as a receive interrupt is generated. The subsequent reading data may not be ensured.

11.2.8 Receive Control Register 1 [CECRCR1]

	31	30	29	28	27	26	25	24
bit Symbol	—							CECACK DIS
Read/Write	R							R/W
After reset	0							0
Function	"0" is read.							Logical "0" as ACK response 0: send 1: not send
	23	22	21	20	19	18	17	16
bit Symbol	—		CECHNC 1	CECHNC 0	—	CECLNC 2	CECLNC 1	CECLNC 0
Read/Write	R		R/W		R	R/W		
After reset	0		0		0	0		
Function	"0" is read.		Noise cancellation time for detecting "1". 00: 1cycle 01: 2cycles 10: 3cycles 11: 4cycles		"0" is read.	Noise cancellation time for detecting "0". 000: 1cycle 001: 2cycles 010: 3cycles 011: 4cycles 100: 5cycles 101: 6cycles 110: 7cycles 111: 8cycles		
	15	14	13	12	11	10	9	8
bit Symbol	—	CECMIN2	CECMIN1	CECMIN0	—	CECMAX 2	CECMAX 1	CECMAX 0
Read/Write	R	R/W			R	R/W		
After reset	0	0			0	0		
Function	"0" is read.	Time to identify as min. cycle error 000: 2.05ms 001: 2.05ms+1cycle 010: 2.05ms+2cycles 011: 2.05ms+3cycles 100: 2.05ms-1cycle 101: 2.05ms-2cycles 110: 2.05ms-3cycles 111: 2.05ms-4cycles			"0" is read.	Time to identify as max. cycle error 000: 2.75ms 001: 2.75ms+1cycle 010: 2.75ms+2cycles 011: 2.75ms+3cycles 100: 2.75ms-1cycle 101: 2.75ms-2cycles 110: 2.75ms-3cycles 111: 2.75ms-4cycles		
	7	6	5	4	3	2	1	0
bit Symbol	—	CECDAT 2	CECDAT 1	CECDAT 0	CECTOUT		CECRI HLD	CECOTH
Read/Write	R	R/W			R/W		R/W	R/W
After reset	0	0			0		0	0
Function	"0" is read.	Point of determining the data as 0 or 1. 000: 1.05ms 001: 1.05ms+2cycles 010: 1.05ms+4cycles 011: 1.05ms+6cycles 100: 1.05ms-2cycles 101: 1.05ms-4cycles 110: 1.05ms-6cycles 111: Reserved			Cycle to identify timeout 00: 1 bit cycle 01: 2 bit cycles 10: 3 bit cycles 11: Reserved		Error interrupt suspend 0: Yes 1: No	Data reception at logical address discrepan cy 0: Yes 1: No

- <CEACKDIS>: Specifies if logical "0" is sent or not as an ACK response to the data block when destination address corresponds with the address set in the logical address register. (The header block sends logical "0" as an ACK response regardless of the bit setting when detecting the addresses corresponding).
- <CECHNC1:0>: Specifies the time of the noise cancellation for each sampling clock cycle when detecting "1".
It is considered as noise if "1"s of the same number as the specified cycles are not sampled.
- <CECLNC2:0>: Specifies the time of the noise cancellation for each sampling clock cycle when detecting "0".
It is considered as noise if "0"s of the same number as the specified cycles are not sampled.
- <CECMIN2:0>: Specifies the minimum time to identify a valid bit.
Enables to specify it for each sampling clock cycle between the ranges of -4 to +3 cycles from approx. 2.05 ms.
An interrupt is generated and "0" is output to CEC for approx. 3.6 ms when one bit cycle is shorter than the specified time.
- <CECMAX2:0>: Specifies the maximum time to identify a valid bit.
Enables to specify it for each sampling clock cycle between the ranges of -4 to +3 cycles from approx. 2.75 ms.
An interrupt is generated when one bit cycle is longer than the specified time.
- <CECDAT2:0>: Specifies the point of determining the data as 0 or 1.
Enables to specify it per two sampling clock cycles between the ranges of + or - 6 cycles from approx. 1.05 ms.
- <CECTOUT>: Specifies the time to determine a timeout. Enables to specify it between 1 bit and 3 bits for each bit cycle.
This setting is used to detect a timeout occurs when the <CECRIHLD> bit is valid.
- <CECRIHLD>: Specifies if a receive error interrupt (maximum cycle error, buffer overrun and waveform error) is suspended or not.
Setting "1" generates no interrupt at the error detection. If data continues to an ACK bit, an ACK response is executed by a reversed logic. If the subsequent bits are interrupted, it is determined as a timeout, based on the setting in <CECTOUT>.
After the ACK response or the timeout determination, an interrupt is generated.
- <CECOTH>: Specifies if data is received or not when destination address does not correspond with the address set in the logical address register.

- (Note 1) The settings in <CECHNC>, <CECLNC> and <CECDAT> are also used in receiving an ACK response at transmission.
- (Note 2) Changing the configurations during transmission or reception may harm its proper operation. Before the change, set the CECREN <CECREN> bit to disable the reception and read the <CECREN> bit and the CECTEN <CECTEN> bit to ensure that the operation is stopped.
- (Note 3) A broadcast message is received regardless of the <CECOTH> register setting.

11.2.9 Receive Control Register 2 [CECR2]

	15	14	13	12	11	10	9	8
bit Symbol	—	CEC SWAV32	CEC SWAV31	CEC SWAV30	—	CEC SWAV22	CEC SWAV21	CEC SWAV20
Read/Write	R	R/W			R	R/W		
After reset	0	0			0	0		
Function	"0" is read.	Max. cycle to detect start bit 000: 4.7ms 001: 4.7ms+1cycle 010: 4.7ms+2cycles 011: 4.7ms+3cycles 100: 4.7ms+4cycles 101: 4.7ms+5cycles 110: 4.7ms+6cycles 111: 4.7ms+7cycles			"0" is read.	Min. cycle to detect start bit 000: 4.3ms 001: 4.3ms-1cycle 010: 4.3ms-2cycles 011: 4.3ms-3cycles 100: 4.3ms-4cycles 101: 4.3ms-5cycles 110: 4.3ms-6cycles 111: 4.3ms-7cycles		
	7	6	5	4	3	2	1	0
bit Symbol	—	CEC SWAV12	CEC SWAV11	CEC SWAV10	—	CEC SWAV02	CEC SWAV01	CEC SWAV00
Read/Write	R	R/W			R	R/W		
After reset	0	0			0	0		
Function	"0" is read.	Max. time of start bit rising timing. 000: 3.9ms 001: 3.9ms+1cycle 010: 3.9ms+2cycles 011: 3.9ms+3cycles 100: 3.9ms+4cycles 101: 3.9ms+5cycles 110: 3.9ms+6cycles 111: 3.9ms+7cycles			"0" is read.	Min. time of start bit rising timing. 000: 3.5ms 001: 3.5ms-1cycle 010: 3.5ms-2cycles 011: 3.5ms-3cycles 100: 3.5ms-4cycles 101: 3.5ms-5cycles 110: 3.5ms-6cycles 111: 3.5ms-7cycles		

- <CECSWAV3 2:0>: Specifies the cycles to detect a start bit.
 <CECSWAV2 2:0>: <CECSWAV3> is for the maximum cycles. Enables to set it for each sampling clock cycle between the ranges of 0 to +7 cycles from default value (4.7 ms).
 <CECSWAV2> is for the minimum cycles. Enables to set it for each sampling clock cycle between the ranges of 0 to +7 cycles from default value (4.3 ms).
- <CECSWAV1 2:0>: Specifies the rising timing of a start bit in its detection.
 <CECSWAV0 2:0>: <CECSWAV1> is for the maximum time of the rising timing. Enables to set it for each sampling clock cycle between the ranges of 0 to +7 cycles from default value (3.9 ms).
 <CECSWAV0> is for the minimum time of the rising timing. Enables to set it for each sampling clock cycle between the ranges of 0 to -7 cycles from default value (3.5 ms).

(Note) Changing the configurations during transmission or reception may harm its proper operation. Before the change, set CECREN <CECREN> to disable the reception and read the <CECREN> bit to ensure that the operation is stopped.

11.2.10 Receive Control Register 3 [CECR3]

	23	22	21	20	19	18	17	16
bit Symbol	—	CEC WAV32	CEC WAV31	CEC WAV30	—	CEC WAV22	CEC WAV21	CEC WAV20
Read/Write	R	R/W			R	R/W		
After reset	0	0			0	0		
Function	"0" is read.	The latest rising timing of logical "0" determined as proper waveform. 000: 1.7ms 001: 1.7ms+1cycle 010: 1.7ms+2cycles 011: 1.7ms+3cycles 100: 1.7ms+4cycles 101: 1.7ms+5cycles 110: 1.7ms+6cycles 111: 1.7ms+7cycles			"0" is read.	The fastest rising timing of logical "0" determined as proper waveform. 000: 1.3ms 001: 1.3ms-1cycle 010: 1.3ms-2cycles 011: 1.3ms-3cycles 100: 1.3ms-4cycles 101: 1.3ms-5cycles 110: 1.3ms-6cycles 111: 1.3ms-7cycles		
	15	14	13	12	11	10	9	8
bit Symbol	—	CEC WAV12	CEC WAV11	CEC WAV10	—	CEC WAV02	CEC WAV01	CEC WAV00
Read/Write	R	R/W			R	R/W		
After reset	0	0			0	0		
Function	"0" is read.	The latest rising timing of logical "1" determined as proper waveform. 000: 0.8ms 001: 0.8ms+1cycle 010: 0.8ms+2cycles 011: 0.8ms+3cycles 100: 0.8ms+4cycles 101: 0.8ms+5cycles 110: 0.8ms+6cycles 111: 0.8ms+7cycles			"0" is read.	The fastest rising timing of logical "1" determined as proper waveform. 000: 0.4ms 001: 0.4ms-1cycle 010: 0.4ms-2cycles 111: 0.4ms-3cycles 100: 0.4ms-4cycles 101: 0.4ms-5cycles 110: 0.4ms-6cycles 111: 0.4ms-7cycles		
	7	6	5	4	3	2	1	0
bit Symbol	—							CEC WAVEN
Read/Write	R							R/W
After reset	0							0
Function	"0" is read.							Waveform error detection 1: Enabled 0: Disabled

(Note) Changing the configurations during transmission or reception may harm its proper operation. Before the change, configure the <CECREN> bits to disable the reception and read the <CECREN> bit to ensure that the operation is stopped.

- <CECWAV3 2:0>: Specifies the latest rising timing of logical "0" determined as proper waveform. This setting is enabled when the <CECWAVEN> bit is set to "1".
By setting these bits, an error is detected if rising edge of the received waveform comes later than that of proper logical "0". Enables to set it for each sampling clock cycle between the ranges of 0 to 7 cycles from defined maximum tolerance (1.7 ms). The received waveform is considered to be an error if a rising edge is not detected from the start point of the bit to the value specified in <CECWAV3>.
- <CECWAV2 2:0>:
<CECWAV1 2:0>: Specifies the fastest rising timing of logical "0" and the latest rising timing of logical "1" determined as proper waveform. This setting is enabled when the <CECWAVEN> bit is set to "1".
By setting these bits, an error is detected if rising edge of the received waveform comes faster than logical "0" and later than that of proper logical "1".
Enables to set <CECWAV1> for each sampling clock cycle between the ranges of 0 to 7 cycles from defined maximum tolerance (0.8 ms) of logical "1" waveform.
Enables to set <CECWAV2> for each sampling clock cycle between the ranges of 0 to -7 cycles from defined minimum tolerance (1.3 ms) of logical "0" waveform.
The received waveform is considered to be an error if a rising edge is detected between the values specified in <CECWAV2> and <CECWAV1>.
- <CECWAV0 2:0>: Specifies the fastest rising timing of logical "1" determined as proper waveform. This setting is enabled when the <CECWAVEN> bit is set to "1".
By setting these bits, an error is detected if rising edge of the received waveform comes faster than that of proper logical "1". Enables to set <CECWAV0> for each sampling clock cycle between the ranges of 0 to -7 cycles from defined minimum tolerance (0.4 ms).
The received waveform is considered to be an error if a rising edge is not detected from a start point of the bit to the value specified in <CECWAV0>.
- <CECWAVEN>: Detects a received waveform does not identical to the one defined and generates waveform error interrupt.
If enabled, an error is detected according to the setting of <CECWAV0> <CECWAV1> <CECWAV2> <CECWAV3>.

11.2.11 Transmit Enable Register [CECTEN]

	7	6	5	4	3	2	1	0
bit Symbol	—						CEC TRANS	CECTEN
Read/Write	R						R	R/W
After reset	0						0	0
Function	"0" is read.						Transmis sion state 0: not in progress 1: in progress	Transmis sion control [write] 0: Disabled 1: Enabled [read] 0: halted 1: in progress

<CECTrans>: Indicates whether the transmission is in progress or not.
It indicates "1" upon starting the transmission of the start bit. It indicates "0" if transmission is completed or an interrupt is generated.
Writing to this bit is ignored.

<CECTEN>: Controls the CEC transmission.
Writing this bit enables or disables the transmission. Writing "1" to this bit initiates the transmission. This bit is automatically cleared by a transmit completion interrupt or an error interrupt.
The state of transmission circuit can be monitored by reading this bit. It enables you to check if what you set has properly been reflected.

(Note 1) Set <CECTEN> after setting the transmit buffer register and transmit control register.

(Note 2) Stop transmission and reception before changing the settings or enabling the transmission and reception.

11.2.12 Transmit Buffer Register [CECTBUF]

	15	14	13	12	11	10	9	8
bit Symbol	—							CECTEOM
Read/Write	R							R/W
After reset	0							0
Function	"0" is read.							EOM bit
	7	6	5	4	3	2	1	0
bit Symbol	CECTBUF 7	CECTBUF 6	CECTBUF 5	CECTBUF 4	CECTBUF 3	CECTBUF 2	CECTBUF 1	CECTBUF 0
Read/Write	R/W							
After reset	0							
Function	Transmitted data							

<CECTEOM>: Specifies the EOM bit to transmit.

<CECTBUF7:0>: Specifies a byte of data to transmit. The bit 7 is the MSB.

11.2.13 Transmit Control Register [CECTCR]

	23	22	21	20	19	18	17	16
bit Symbol	—	CECSTRS 2	CECSTRS 1	CECSTRS 0	—	CECSPRD 2	CECSPRD 1	CECSPRD 0
Read/Write	R	R/W			R	R/W		
After reset	0	0			0	0		
Function	"0" is read.	Rising timing of start bit 000: Reference value (RV) 001: RV –1cycle 010: RV –2cycle 011: RV –3cycle 100: RV –4cycle 101: RV –5cycle 110: RV –6cycle 111: RV –7cycle			"0" is read.	Start bit cycle 000: RV 001: RV –1cycle 010: RV –2cycle 011: RV –3cycle 100: RV –4cycle 101: RV –5cycle 110: RV –6cycle 111: RV –7cycle		
	15	14	13	12	11	10	9	8
bit Symbol	—	CECDTRS 2	CECDTRS 1	CECDTRS 0	CECDPRD 3	CECDPRD 2	CECDPRD 1	CECDPRD 0
Read/Write	R	R/W			R/W			
After reset	0	0			0			
Function	"0" is read.	Rising timing of data bit 000: RV 001: RV –1cycle 010: RV –2cycle 011: RV –3cycle 100: RV –4cycle 101: RV –5cycle 110: RV –6cycle 111: RV –7cycle			Data bit cycle 0000: RV 1000: RV –8cycle 0001: RV –1cycle 1001: RV –9cycle 0010: RV –2cycle 1010: RV –10cycle 0011: RV –3cycle 1011: RV –11cycle 0100: RV –4cycle 1100: RV –12cycle 0101: RV –5cycle 1101: RV –13cycle 0110: RV –6cycle 1110: RV –14cycle 0111: RV –7cycle 1111: RV –15cycle			
	7	6	5	4	3	2	1	0
bit Symbol	—			CECBRD	CECFREE 3	CECFREE 2	CECFREE 1	CECFREE 0
Read/Write	R			R/W	R/W			
After reset	0			0	0			
Function	"0" is read.			Broadcast transmission 0: No 1: Yes	Time of bus to be free 0000: 1 bit cycle 1000: 9 bit cycle 0001: 2 bit cycle 1001: 10 bit cycle 0010: 3 bit cycle 1010: 11 bit cycle 0011: 4 bit cycle 1011: 12 bit cycle 0100: 5 bit cycle 1100: 13 bit cycle 0101: 6 bit cycle 1101: 14 bit cycle 0110: 7 bit cycle 1110: 15 bit cycle 0111: 8 bit cycle 1111: 16 bit cycle			

- <CECSTRS2:0>: Specifies the rising timing of a start bit.
Enables to set it for each sampling clock cycle between the ranges of 0 to -7 cycles from default value (3.7 ms approx.).
- <CECSPRD2:0>: Specifies a cycle of a start bit.
Enables to set it for each sampling clock cycle between the ranges of 0 to -7 cycles from default value (4.5 ms approx.).
- <CECDTRS2:0>: Specifies the rising timing of a data bit.
Enables to set it for each sampling clock cycle between the ranges of 0 to -7 cycles from default value (logical "1": 0.6 ms approx., logical "0": 1.5 ms approx.).
- <CECDPRD3:0>: Specifies a cycle of a data bit.
Enables to set it for each sampling clock cycle between the ranges of 0 to -15 cycles from default value (2.4 ms approx.).
- <CECBRD>: Set this bit to "1" when transmitting a broadcast message.
- <CECFREE3:0>: Specifies time of a bus to be free that checked before transmission. Start transmission after checking the CEC line kept inactive during the specified cycles.

11.2.14 Receive Interrupt Status Register [CECRSTAT]

	7	6	5	4	3	2	1	0
bit Symbol	—	CECRIWAV	CECRIOR	CECRIACK	CECRIMIN	CECRIMAX	CECRISTA	CECRIEND
Read/Write	R	R	R	R	R	R	R	R
After reset	0	0	0	0	0	0	0	0
Function	"0" is read.	Interrupt flag 1: Wave form error	Interrupt flag 1: Receive buffer overrun	Interrupt flag 1: ACK collision	Interrupt flag 1: Min. cycle error	Interrupt flag 1: Max. cycle error	Interrupt flag 1: Start bit detection	Interrupt flag 1: Completion of 1 byte data reception

<CECRIWAV>: Indicates that waveform error is detected. The error occurs when waveform error detection is enabled in CECRCR3 <CECWAVEN>.

<CECRIOR>: Indicates the receive buffer receives next data before reading the data that had already been set.

<CECRIACK>: Indicates "0" is detected after the specified time to output ACK bit "0".

<CECRIMIN>: Indicates one bit cycle is shorter than the minimum cycle error detection time specified in CECRCR1<CECMIN>.

<CECRIMAX>: Indicates one bit cycle is longer than the maximum cycle error detection time specified in CECRCR1<CECMAX>.

<CECRISTA>: Indicates a start bit is detected.

<CECRIEND>: Indicates 1 byte of data reception is completed.

(Note)	Writing to this bit is ignored.

11.2.15 Transmit Interrupt Status Register [CECTSTAT]

	7	6	5	4	3	2	1	0
bit Symbol	—			CECTIUR	CECTIACK	CECTIAL	CECTIEND	CECTISTA
Read/Write	R			R	R	R	R	R
After reset	0			0	0	0	0	0
Function	"0" is read.			Interrupt flag 1: Transmit buffer underrun	Interrupt flag 1: ACK error detection	Interrupt flag 1: Arbitration lost occurs	Interrupt flag 1: data transmission is completed	Interrupt flag 1: Start transmission

<CECTIUR>: Indicates next data has not set to the transmission buffer within a byte of data transmission.

<CECTIACK>: Indicates one of the following conditions occurs.

- When logical "0" is not detected in transmission to the specific address.
- When logical "1" is not detected in transmission of a broadcast message .

<CECTIAL>: Indicates "0" is detected while transmitting "1".

<CECTIEND>: Indicates data transmission including the EOM bit is completed.

<CECTISTA>: Indicates 1 byte of data transmission is started.

(Note)	Writing to this bit is ignored.

11.3 Operations

11.3.1 Reception

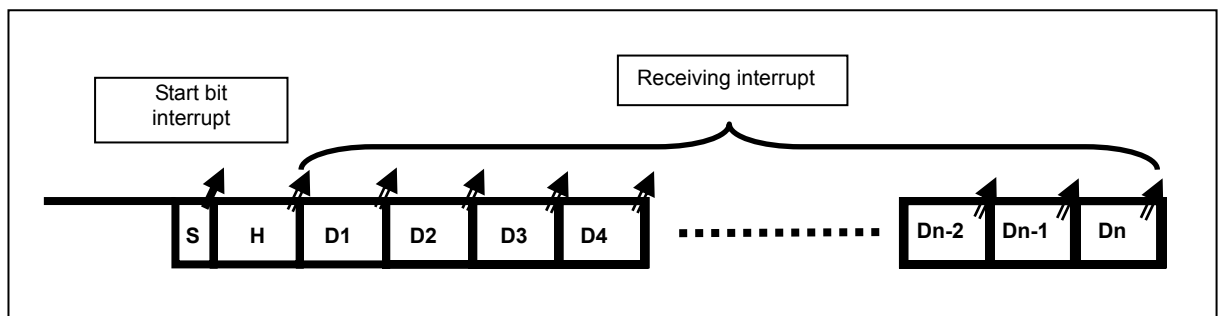
11.3.1.1 Sampling Clock

CEC lines are sampled by a low speed clock.

11.3.1.2 Basic Operation

If reception is enabled, detecting a start bit generates a start bit interrupt and starts receiving data per byte. The interrupt is generated after a byte of data (8 bit), the EOM bit and the ACK bit are received. The data, EOM and ACK bit are stored in a buffer.

Detecting an error during data reception causes an error interrupt, and CEC waits for the next start bit. The received data is discarded.



11.3.1.3 Preconfiguration

Before receiving data, reception settings to the Logical Address Register <CECADD>, the Receive Control Register 1 <CECR1>, the Receive Control Register 2 <CECR2> and the Receive Control Register 3 <CECR3> are required.

(1) Logical Address Configuration

Configure logical address assigned to this product to the CECADD register. Multiple addresses can be set simultaneously since every bit in this register corresponds with each address.

(Note) A broadcast message is received regardless of the CECADD register setting. By allocating a logical address of a device to 15, logical "0" is sent as an ACK response to the broadcast message.

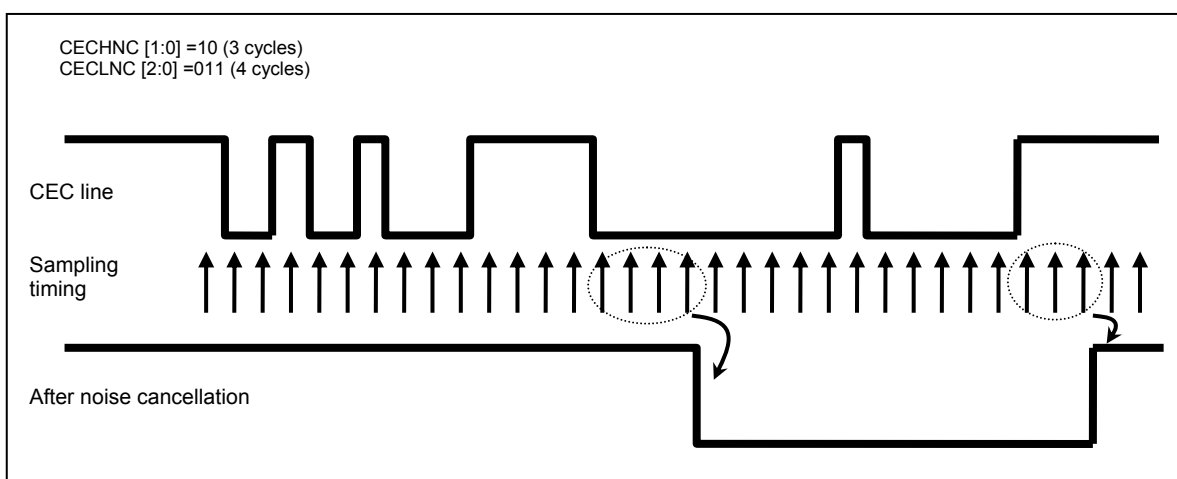
(2) Noise Cancellation Time

The noise cancellation time is configurable with the <CECHNC1:0><CECLNC2:0> bits of the CECR1 register. You can configure the time to detect "1" and "0" respectively.

It is considered as noise if "1"s or "0"s of the same number as the specified value are not sampled.

A CEC line is monitored at each rising edge of a sampling clock. In the case that the CEC line is changed from "1" to "0", the change is fully recognized if "0"s of the same number as specified in the <CECLNC> bit are monitored. In the case that the CEC line is changed from "0" to "1", the change is fully recognized if "1"s of the same number as specified in the <CECHNC> bit are sampled.

The following illustrates the operation of a case that a noise cancelling is configured as <CECHNC 1:0>=10 (3 cycles) and <CECLNC 2:0>=011 (4 cycles). By cancelling the noise, a signal "1" shifts to "0" after "0" is sampled four times. The signal "0" shifts to "1" after "1" is sampled three times.



(3) Cycle Error

Configure the CECRCR1 <CECMIN2:0> <CECMAX2:0> bits to detect a cycle error.

You can specify the time to detect a cycle error for each sampling clock cycle between the ranges of -4 to +3 cycles from the maximum or minimum time set in the CEC standard.

Detecting an error during data reception causes an error interrupt, and CEC waits for the next start bit. The received data is discarded.

(4) Point of Determining Data

Configure the CECRCR1 <CECDAT> bit for the point of determining the data as "0" or "1".

You can specify it per two sampling clock cycles between the ranges of + or - 6 cycles with approx. 1.05 ms from the bit start point.

(5) ACK Response

Configuring the CECRCR1 <CEACKDIS> bit enables you to specify if logical "0" is sent or not as an ACK response to the data block when destination address corresponds with the address set in the logical address register.

The header block sends logical "0" as an ACK response regardless of the bit setting when detecting the addresses corresponding.

(6) Receive Error Interrupt Suspend

Configure the CECRCR1 <CECRIHLD> bit to specify if a receive error interrupt (maximum cycle error, buffer overrun and waveform error) is suspended or not.

Setting "1" generates no interrupt at the error detection. If data continues to the ACK bit, an ACK response is executed by a reversed logic. If the subsequent bits are interrupted, it is determined as a timeout, based on the setting in <CECTOUT> of the CECRCR1 register. After the ACK response or the timeout determination, an interrupt is generated.

(7) Cycles to Identify Timeout

Configure the CECRCR1 <CECTOUT> bit to specify the time to determine a timeout.

This is used when the setting of a receive error interrupt suspension, which is specified in CECRCR1 <CECRIHLD>, is valid.

(8) Data Reception at Logical Address Discrepancy

By setting CECRCR1 <CECOTH>, you can specify if data is received or not when destination address does not correspond with the address set in the logical address register.

In this case, data is received as usual, and an interrupt is generated by detecting an error. However, an ACK response of neither the header block nor the data block is sent.

(Note) A broadcast message is received regardless of the <CECOTH> register setting.

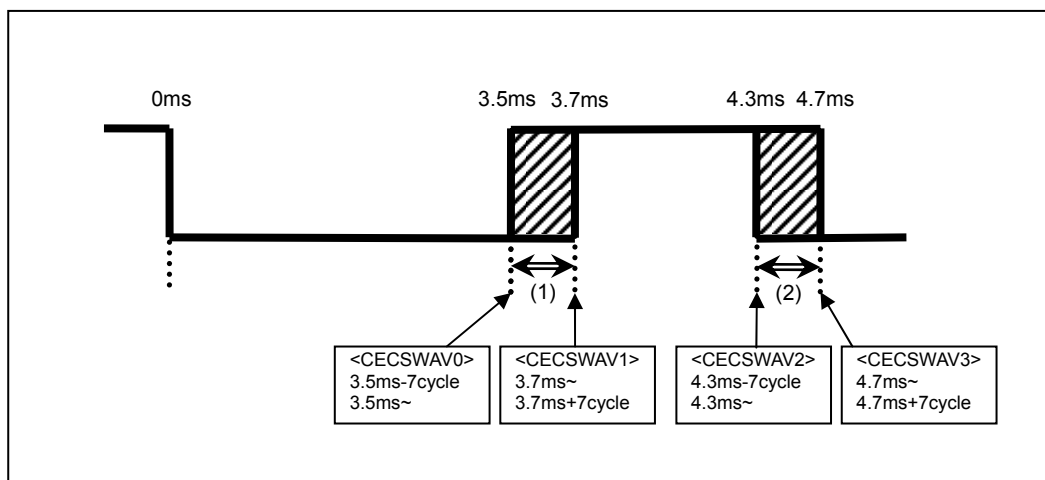
(9) Start Bit Detection

Configuring the CECRCR2 register allows you to specify the rising timing and a cycle of the start bit detection respectively.

<CECSWAV0> is to specify the fastest start bit rising timing. <CECSWAV1> is to specify the latest start bit rising timing ((1) in the figure shown below).

<CECSWAV2> is to specify the minimum cycle of a start bit. <CECSWAV3> is to specify the maximum cycle of a start bit ((2) in the figure shown below).

If a rising edge during the period (1) and a falling edge during the period (2) are detected, the start bit is considered to be valid.



(10) Waveform Error Detection

To detect an error when a received waveform is out of the defined tolerance range, configure the CECRCR3 register.

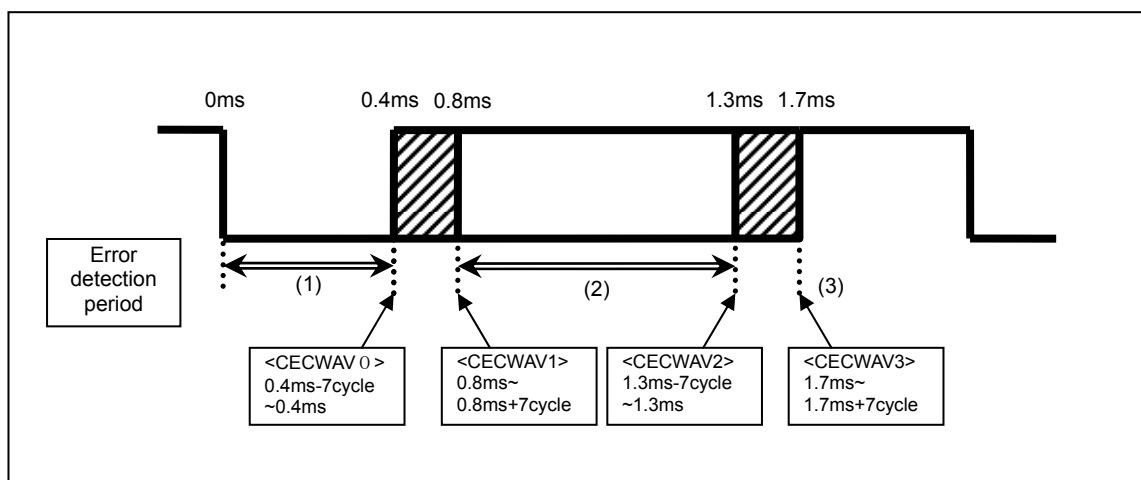
An error is detected when the <CECWAVEN> bit of the CECRCR3 register is enabled. You can specify the detection time in the <CECWAV0> <CECWAV1> <CECWAV2> <CECWAV3> bits.

If the rising edge is detected during the period (1) or (2) shown below, or not detected in the timing described in (3), a waveform error interrupt is generated.

(1) A period between the beginning of a bit and the fastest logical "1" rising timing.

(2) A period between the latest logical "1" rising timing and the fastest logical "0" rising timing.

(3) The latest logical "0" rising timing.



11.3.1.4 Enabling Reception

After configuring the CECADD, CECRCR1, CECRCR2 and CECRCR3 registers, CEC is ready for reception by enabling the CECREN <CECREN> bit. Detecting a start bit initiates the reception.

(Note) Changing the configurations of the CECADD, CECRCR1, CECRCR2 and CECRCR3 registers during transmission or reception may harm its proper operation. Before the change of the registers shown below, set the CECREN <CECREN> bit to disable the reception and read the <CECREN> bit and the CECTEN <CECTEN> bit to ensure that the operation is stopped.

CECRCR1	<CECHNC><CECLNC>	Noise cancellation time
	<CECMIN><CECMAX>	Time to identify cycle error
	<CECOTH>	Data reception at logical address discrepancy
CECRCR2	<CECSWAV0><CECSWAV1> <CECSWAV2><CECSWAV3>	Start bit detection
CECRCR3	<CECWAV0><CECWAV1> <CECWAV2><CECWAV3>	Waveform error detection (when enabled)

11.3.1.5 Reception

After detecting a start bit, a start bit interrupt is generated, and the CECRSTAT <CECRISTA> bit is set.

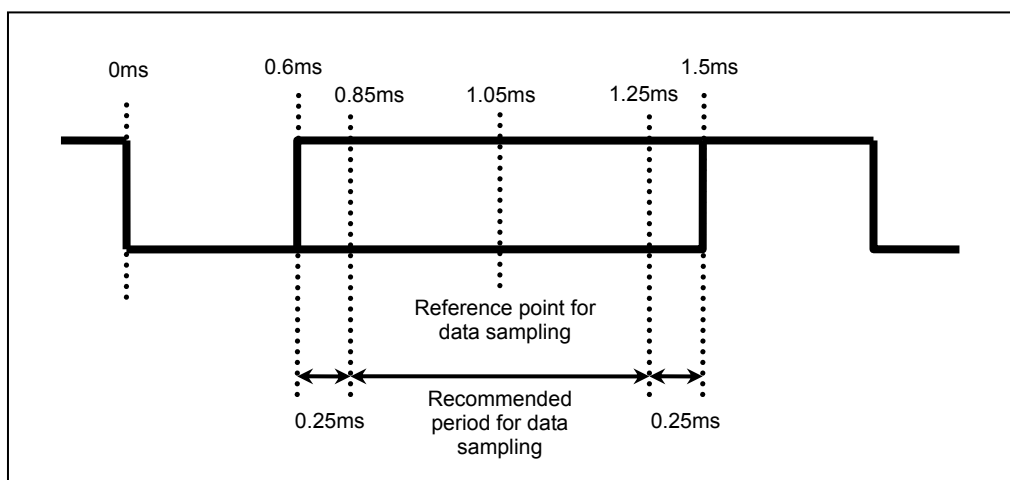
Upon receiving a byte of data, the EOM and ACK bits, they are stored in the CECRBUF register. A receive interrupt is generated and it causes the CECRSTAT <CECRIEND> bit to be set. Same as the other data, the ACK bit that monitored the CEC line is stored instead of the one generated in the CEC circuit.

The reception continues from the first data block until the final data block that has the EOM bit indicating "1". After detecting the final data block, CEC waits for a next start bit.

11.3.1.6 Data Sampling Point

The figure shown below illustrates a data sampling timing.

With the CECRCR1 <CECDAT> bit, you can specify a data sampling point per two sampling clock cycles between the ranges of + or - 6 cycles from a reference point (approx. 1.05 ms).



11.3.1.7 ACK Response

Setting the CECRCR1 <CECACKDIS> bit enables to specify if logical “0” is sent or not as an ACK response to the data block when destination address corresponds with the address set in the logical address register. The header block sends logical “0” as an ACK response regardless of the bit setting when detecting the addresses corresponding.

The following lists the ACK responses.

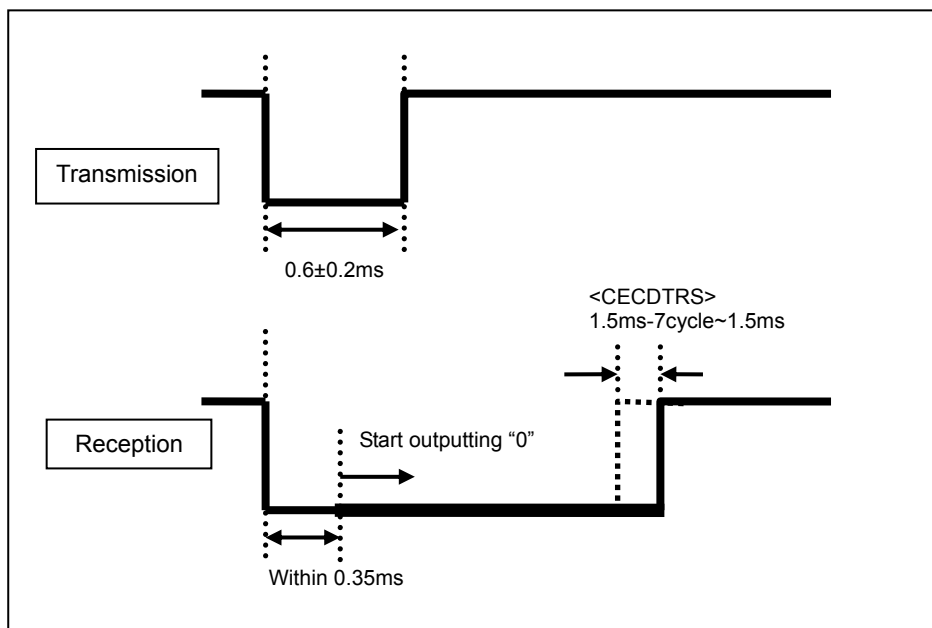
“Yes” indicates that CEC outputs “0” as a response to the ACK signal from a transmission device (ACK bit: logical “0”). “No” indicates that CEC does not output “0” as a response to the ACK signal from a transmission device (ACK bit: logical “1”).

Register setting		Header block address		Data block address	
		Conformity	Discrepancy	Conformity	Discrepancy
CECRCR1 <CECACKDIS>	“0” (responding logical “0”)	Yes	No	Yes	No
	“1” (not responding logical “0”)			No	No

The following describes the ACK response timing.

“0” is output within 0.35 ms from detecting the falling edge of the ACK bit output from the transmission device. The timing to stop output is the same as that of outputting logical “0” for transmission. With the CECTCR <CECDTRS> bit, the timing can be specified between the defined fastest rising timing and the reference value.

(Reference) The configuration of <CECDTRS> is applied for transmission of the data bits and the EOM bit.



11.3.1.8 Detecting Error Interrupt

Detecting an error during data reception causes an error interrupt, and CEC waits for the next start bit. The received data is discarded.

It is possible to suspend a receive error interrupt (maximum cycle error, receive buffer overrun and waveform error), continue reception and send the reversed ACK response.

You can check the interrupt factor by monitoring the bit of the CECRSTAT register corresponding to the interrupt.

11.3.1.9 Details of Receive Error

(1) Cycle Error

Period between the falling edges of the two sequential bits is measured during reception. If the period does not comply with the specified minimum or maximum value, a cycle error interrupt is generated.

The maximum and minimum cycles are specified in the CECRCR1 <CECMIN2:0> <CECMAX2:0> bits. A cycle error can be detected for each sampling clock cycle between the ranges of -4 to +3 cycles from the minimum value (approx. 2.05 ms) or the maximum value (approx. 2.75 ms) defined by the CEC standard.

The CECRSTAT <CECRIMIN> bit or the <CECRIMAX> bit is set if a cycle error interrupt is generated.

The minimum cycle error causes CEC to output "0" for approx. 3.6 ms.

(2) ACK Collision

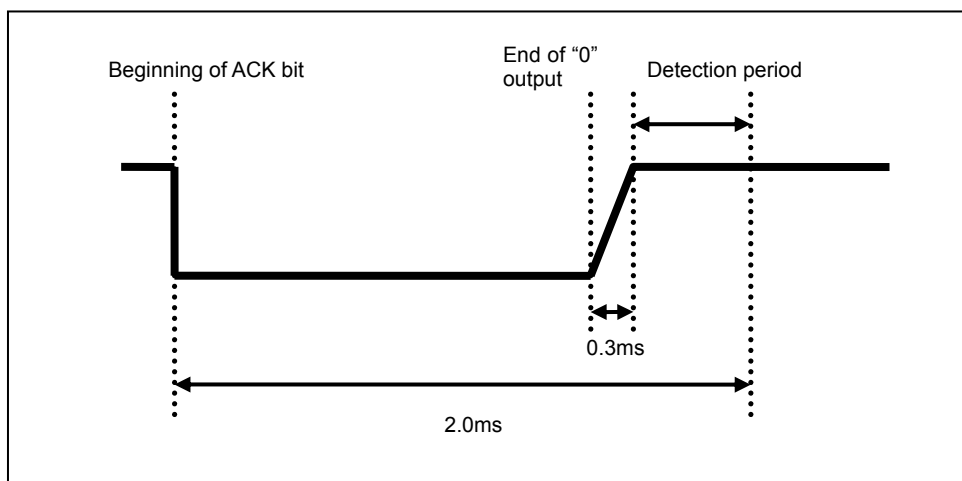
At an ACK response, detecting "0" after the specified period to output generates an ACK collision interrupt or a minimum cycle error interrupt.

The ACK collision interrupt sets the CECRSTAT <CECRIACK> bit. The minimum cycle error interrupt sets the CECRSTAT <CECRIMIN> bit.

The following describes the period and method of detection.

Detection starts approx. 0.3 ms after the end of the period of outputting "0" and ends approx 2.0 ms from the starting point (the falling edge) of the ACK bit.

At 0.3 ms from the end of the period of outputting "0", CEC checks if the CEC line is "0" or not. If it is "0", an ACK collision interrupt is generated. If it is "1", and "0" is detected during the detection period, the minimum cycle error interrupt is generated. The minimum cycle error causes CEC to output "0" for approx. 3.6 ms.



(3) Receive Buffer Overrun

A receive buffer overrun interrupt is generated when the next data reception is completed before reading the data stored in the receive buffer.

The interrupt sets the CECRSTAT <CECRIOR> bit.

(4) Waveform Error

A waveform error occurs when waveform error detection is enabled in CECRCR3. Detecting a waveform, which does not identical to the defined, results in the waveform error. The interrupt is generated.

The interrupt sets the CECRSTAT <CECRIWAV> bit.

(5) Suspending Receive Error Interrupt

You can specify if a maximum cycle error, a buffer overrun and a waveform error are suspended or not without generating an interrupt at error detection. This can be set in the CECRCR1 <CECRIHLD> bit. To enable the setting, a timeout setting with the CECRCR1 <CECTOUT> bit is required.

Under suspend-enable condition, if CEC keeps receiving the next bit and the entire reception including the ACK bit is completed, CEC generates an interrupt after a reversed ACK response is executed. "1" is set to the bits of the CECRSTAT register: the <CECRIEND> bit that indicates the reception completion, and the bits corresponding to the detected errors.

If the reception of the next bit is interrupted, CEC starts to measure the timeout period, and an interrupt is generated after the timeout. "1" is set to the bits of the CECRSTAT register corresponding to the detected error.

The timeout is measured from the end of the last bit received as is the case with wait time of a bus to be free in transmission.

The information that the interrupts are suspended is held until the EOM bit is received or the timeout occurs. Thus, an interrupt is generated in each reception of a byte of data if multiple bytes are received while interrupts are suspended. "1" is set to the bits of the CECRSTAT register: the <CECRIEND> bit that indicates the reception completion, and the bits corresponding to the detected errors. The flags of the suspended interrupts and the reception completion are set to the bits of the CECRSTAT register.

(Note 1)	A minimum cycle error interrupt is generated upon detecting a minimum cycle error in the next received bit while interrupts are suspended. "0" is output to CEC for approx. 3.6 ms. The flags of the suspended interrupts and the minimum cycle error are set to the bits of the CECRSTAT register.
(Note 2)	If an interrupt other than a minimum cycle error interrupt is generated while interrupts are suspended, CEC continues reception by the ACK response or the timeout. All the flags of the detected interrupts are set to the bits of the CECRSTAT register.

11.3.1.10 Stopping Reception

Writing "0" to the CECREN <CECREN> bit disables data reception. The reception is stopped upon disabling the bit during reception. The received data is discarded.

(Note)	If the reception is disabled while "0" is sent as a signal of minimum cycle error, the "0" output is stopped as well.

11.3.2 Transmission

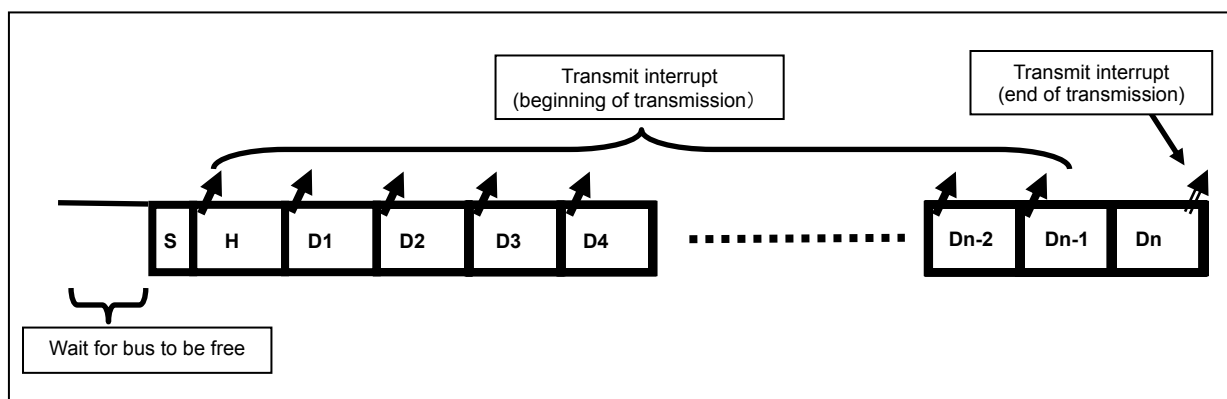
11.3.2.1 Basic Operation

Configure a start bit of transmission after configuring the data buffer. It enables the start bit to be transmitted after time that a bus is free is properly maintained.

Transmitting the first data bit subsequent to the start bit generates a transfer interrupt. It indicates that the next data can be set to a transmit buffer. The ACK bit is sent after a byte of data (8 bit) and the EOM bit are transmitted, and then the ACK response is detected.

The data transmission per byte continues until the data including the EOM bit that indicates "1" is stored in the transmit buffer. If its transmission is completed, a transmit completion interrupt is generated.

If an error is generated during transmission, an error interrupt is generated to stop transmission. Even if reception is enabled, no reception is executed during transmission.



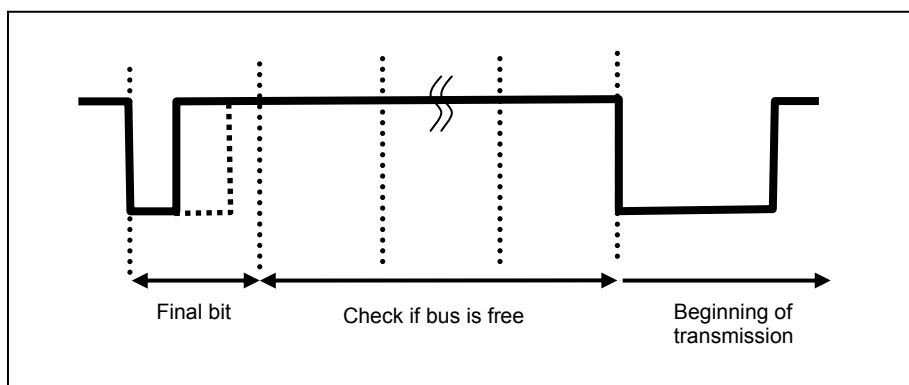
11.3.2.2 Preconfiguration

Before transmitting data, transmission settings to the Transmit Control Register <CECTCR> and the transmit buffer <CECTBUF> are required.

(1) Wait Time for Bus to be Free

Configure the wait time for a bus to be free with the CECTCR <CECFREE> bit. It can be specified for each bit cycle from 1 to 16.

Start point to check if a bus is free is the end of final bit. If a bus is free for specified bit cycles of “1”, transmission starts.



(2) Transmitting Broadcast Message

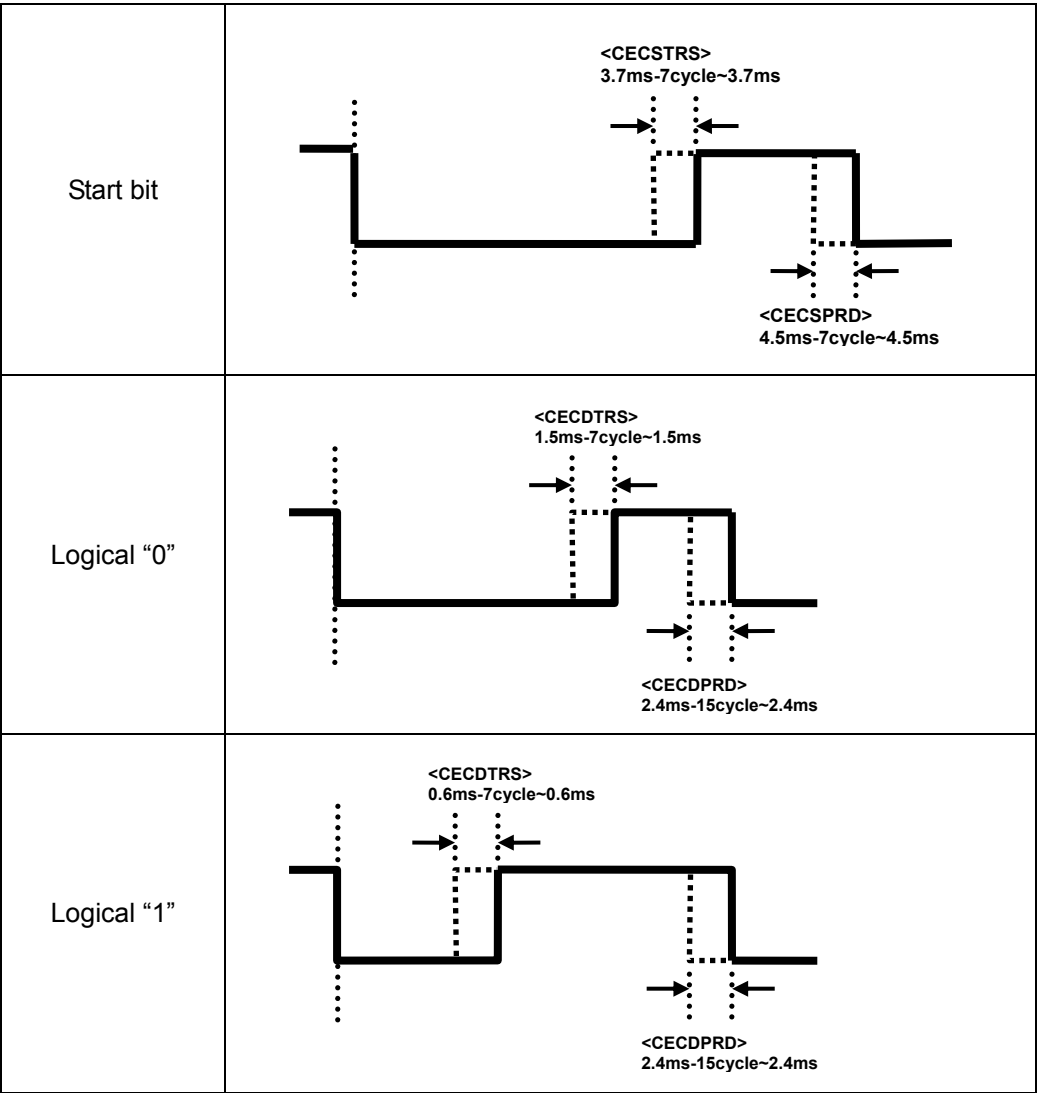
Set the CECTCR <CECBRD> bit when transmitting a broadcast message. If this bit is set, “0” response during an ACK cycle results in an error. If not, “1” response during an ACK cycle results in an error.

(3) Adjusting Transmission Waveform

Both start bit and data bit are capable of adjusting the rising timing and cycle. With the CECTCR <CECSTRS> <CECSPRD> <CECDTRS> <CECDPRD> bits, the timing can be specified between the defined fastest rising/cycle timing and the reference value.

The following figures show how the waveforms differ according to the configurations of the start bit, logical “0” and logical “1”.

(Reference) The configuration of <CECDTRS> is applied for waveform of an ACK response during reception. The ACK response and the logical “0” output show the same waveform.



(4) Preparing Transmission Data

Configure a byte of transmission data and EOM data with the CECTBUF register.

11.3.2.3 Starting Transmission

Transmission is ready by setting the CECTEN <CECTEN> bit to start transmission after setting the CECTCR and CECTBUF registers.

The <CECTEN> bit is never cleared to “0” until a transmit completion interrupt or an error interrupt occurs. Thus you don’t need to set this bit for each a byte of data transmission.

(Note) Changing the configurations of the CECTCR register during transmission or reception may harm its proper operation. Be careful if you change it during transmission.

11.3.2.4 Transmission

Next to the setting for starting transmission, CEC checks if a bus is free. If “1” is sampled as a CEC line for specified bit cycles, start bit is transmitted. The CEC always checks if bus is free. Transmission starts anytime if bus is free for specified bit cycles.

After transmitting a start bit, a byte of data and the EOM data that are set in the buffer are sent to the shift register, and data transmission is started. When CEC starts transmitting the first bit of a byte of data, a transmit interrupt is generated. It sets the CECTSTAT <CECTISTA> bit. Subsequent to the transmit interrupt, a byte of next data can be set to the transmit buffer.

Then the ACK bit is transmitted after 8 bit data and the EOM bit, and the ACK response is checked. This is the end of a byte of data transmission.

Data transfer continues in the above sequence until “1” is set to the EOM bit.

If “1” is set to the EOM, a transmit completion interrupt is generated subsequent to the ACK bit check as described above. Generation of this interrupt, which means the end of a sequence of transmission operation, sets the CECTSTAT <CECTIEND> bit, and clears the CECTEN <CECTEN> bit.

11.3.2.5 ACK Transmission and ACK Error Criterion

A criterion of the ACK error differs depending on the CECTCR <CECBRD> bit.

If this bit is set, broadcast message is transmitted, and the ACK response of logical “0” is determined as an error. If this bit is not set, the ACK response of logical “1” is determined as an error.

11.3.2.6 Detecting Transmission Error

Error detection during transmission generates an interrupt and stops transmission. It clears the CECTEN <CECTEN> bit.

To identify an error factor, the CECTSTAT register has bits that correspond with each interrupt. You can identify the interrupt factor by checking these bits.

(Note) An attempt to stop transmission by an error may cause an improper waveform output to CEC. This is because output is stopped immediately after the error occurs.

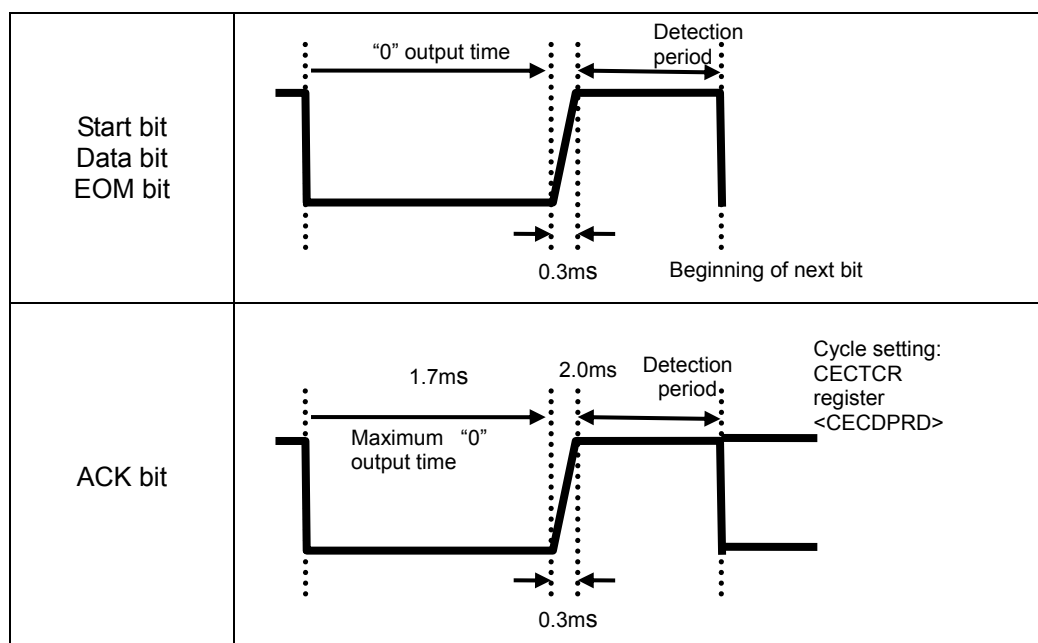
11.3.2.7 Details of Transmission Error

(1) Arbitration Lost

An arbitration lost error occurs when CEC detects "0" on completion of appropriate low duration.

Detecting an arbitration lost error sets the CECTSTAT <CECTIAL> bit.

Two types of the arbitration lost detection periods are shown below.



(2) ACK error

An ACK error interrupt occurs when an ACK response does not conform to the configuration specified in the CECTCR <CECBRD> bit.

When the ACK error interrupt occurs, the CECTSTAT <CECTIACK> bit is set.

The ACK error is detected in the following cases.

Configuration	Determined as an ACK error when
<CECBRD>=0 Broadcast transmission?: No	ACK response is logical "1"
<CECBRD>=1 Broadcast transmission?: Yes	ACK response is logical "0"

(3) Transmit Buffer Underrun

A transmit buffer underrun error is caused by the following sequence.

1. Data in the transmit buffer is transmit to the shift register.
2. An interrupt occurs.
3. A byte of data is transmitted.
4. No data is set to the transmit buffer before starting transmission of a byte of subsequent data.

When an underrun error occurs, the CECTSTAT <CECTIUR> bit is set.

(4) Order of ACK Error and Transmit Buffer Overrun

If interrupt factors of the ACK error and transmit buffer underrun are detected at the end of transmission of a byte of data, the transmit buffer underrun has priority. The transmit buffer underrun interrupt occurs first and then the ACK error interrupt occurs.

11.3.2.8 Stopping Transmission

To stop transmission, send data including the EOM bit that indicates "1". This generates a transmit completion interrupt.

Please note that proper operation is not ensured if the start bit of transmission is set to "0" during transmission.

11.3.2.9 Retransmission

Transmission is stopped by error detection. To retry the transmission, configure the condition and data of starting the transmission.

11.3.3 Software Reset

The entire CEC function can be initialized by software.

Setting “1” to the software reset register CECRESET <CECRESET> bit causes the following operations.

- Reception : Immediately stops. The received data is discarded.
- Transmission : Immediately stops including output to the CEC line.
- Register : All the registers other than CECEN are initialized.

Please note that software reset during transmission may cause the CEC line waveform that does not identical to the defined.

12 Remote control signal preprocessor (RMC)

12.1 Basic operation

Remote control signal preprocessor (hereafter referred to as RMC) receives a remote control signal of which carrier is removed.

12.1.1 Reception of Remote Control Signal

- Sampled by 32KHz clock
- Noise canceller
- Leader detection
- Batch reception up to 72bit of data

12.2 Registers

12.2.1 Register Map

Addresses and names of RMC control registers are shown below.

Register		Address	
		Channel 0	Channel 1
Remote Control Enable Register	RMCxEN	0x4004_0400	0x4004_0440
Remote Control Receive Enable Register	RMCxREN	0x4004_0404	0x4004_0444
Remote Control Receive Data Buffer Register 1	RMCxRBUF1	0x4004_0408	0x4004_0448
Remote Control Receive Data Buffer Register 2	RMCxRBUF2	0x4004_040C	0x4004_044C
Remote Control Receive Data Buffer Register 3	RMCxRBUF3	0x4004_0410	0x4004_0450
Remote Control Receive Control Register 1	RMCxRCR1	0x4004_0414	0x4004_0454
Remote Control Receive Control Register 2	RMCxRCR2	0x4004_0418	0x4004_0458
Remote Control Receive Control Register 3	RMCxRCR3	0x4004_041C	0x4004_045C
Remote Control Receive Control Register 4	RMCxRCR4	0x4004_0420	0x4004_0460
Remote Control Receive Status Register	RMCxRSTAT	0x4004_0424	0x4004_0464

12.2.2 Remote Control Enable Register [RMCEN]

	7	6	5	4	3	2	1	0
bit Symbol	—						I2RMC	RMCEN
Read/Write	R						R/W	R/W
After reset	0						0	0
Function	"0" is read.						RMC in IDLE mode 0: Disabled 1: Enabled	RMC operation 0: Disabled 1: Enabled

<I2RMC>: Controls RMC operation in IDLE mode.
Set "1" to enable RMC in IDLE Mode.
This bit and the <RMCEN> bit can be set simultaneously.

<RMCEN>: Controls RMC operation.
To allow RMC to function, enable the RMCEN bit first. If the operation is disabled, all the clocks for RMC except for the enable register are stopped, and it can reduce power consumption.
If RMC is enabled and then disabled, the settings in each register remain intact.

12.2.3 Remote Control Receive Enable Register [RMCREN]

	7	6	5	4	3	2	1	0
bit Symbol	—							RMCREN
Read/Write	R							R/W
After reset	0							0
Function	"0" is read.							Reception 0: Disabled 1: Enabled

<RMCREN>: Controls reception of RMC.
Setting this bit to "1" enables reception.

(Note) Enable the <RMCREN> bit after setting the RMCxRCR1, RMCxRCR2 and RMCxRCR3.

12.2.4 Remote Control Receive Data Buffer Register 1 [RMCRBUF1]

	31	30	29	28	27	26	25	24
bit Symbol	RMCRBUF 31	RMCRBUF 30	RMCRBUF 29	RMCRBUF 28	RMCRBUF 27	RMCRBUF 26	RMCRBUF 25	RMCRBUF 24
Read/Write	R							
After reset	0							
Function	Received data							
	23	22	21	20	19	18	17	16
bit Symbol	RMCRBUF 23	RMCRBUF 22	RMCRBUF 21	RMCRBUF 20	RMCRBUF 19	RMCRBUF 18	RMCRBUF 17	RMCRBUF 16
Read/Write	R							
After reset	0							
Function	Received data							
	15	14	13	12	11	10	9	8
bit Symbol	RMCRBUF 15	RMCRBUF 14	RMCRBUF 13	RMCRBUF 12	RMCRBUF 11	RMCRBUF 10	RMCRBUF 9	RMCRBUF 8
Read/Write	R							
After reset	0							
Function	Received data							
	7	6	5	4	3	2	1	0
bit Symbol	RMCRBUF 7	RMCRBUF 6	RMCRBUF 5	RMCRBUF 4	RMCRBUF 3	RMCRBUF 2	RMCRBUF 1	RMCRBUF 0
Read/Write	R							
After reset	0							
Function	Received data							

<RMCRBUF31:0>: Reads 4 bytes of received data.

12.2.5 Remote Control Receive Data Buffer Register 2 [RMCRBUF2]

	31	30	29	28	27	26	25	24
bit Symbol	RMCRBUF 63	RMCRBUF 62	RMCRBUF 61	RMCRBUF 60	RMCRBUF 59	RMCRBUF 58	RMCRBUF 57	RMCRBUF 56
Read/Write	R							
After reset	0							
Function	Received data							
	23	22	21	20	19	18	17	16
bit Symbol	RMCRBUF 55	RMCRBUF 54	RMCRBUF 53	RMCRBUF 54	RMCRBUF 53	RMCRBUF 52	RMCRBUF 51	RMCRBUF 50
Read/Write	R							
After reset	0							
Function	Received data							
	15	14	13	12	11	10	9	8
bit Symbol	RMCRBUF 47	RMCRBUF 46	RMCRBUF 45	RMCRBUF 44	RMCRBUF 43	RMCRBUF 42	RMCRBUF 41	RMCRBUF 40
Read/Write	R							
After reset	0							
Function	Received data							
	7	6	5	4	3	2	1	0
bit Symbol	RMCRBUF 39	RMCRBUF 38	RMCRBUF 37	RMCRBUF 36	RMCRBUF 35	RMCRBUF 34	RMCRBUF 33	RMCRBUF 32
Read/Write	R							
After reset	0							
Function	Received data							

<RMCRBUF63:32>: Reads 4 bytes of received data.

12.2.6 Remote Control Receive Data Buffer Register 3 [RMCRBUF3]

	7	6	5	4	3	2	1	0
bit Symbol	RMCRBUF 71	RMCRBUF 70	RMCRBUF 69	RMCRBUF 68	RMCRBUF 67	RMCRBUF 66	RMCRBUF 65	RMCRBUF 64
Read/Write	R							
After reset	0							
Function	Received data							

<RMCRBUF71:64>: Reads a byte of received data.

(Note) Received data is stored from RMCRBUF1 <RMCRBUF0> to RMCRBUF3 <RMCRBUF71> in sequence.

12.2.7 Remote Control Receive Control Register 1 [RMCRCR1]

	31	30	29	28	27	26	25	24
bit Symbol	RMCLC MAX7	RMCLC MAX6	RMCLC MAX5	RMCLC MAX4	RMCLC MAX3	RMCLC MAX2	RMCLC MAX1	RMCLC MAX0
Read/Write	R/W							
After reset	0							
Function	Maximum cycle of leader detection: $RMCLC_{MAX} \times 4 / fs[s]$							
	23	22	21	20	19	18	17	16
bit Symbol	RMCLC MIN7	RMCLC MIN6	RMCLC MIN5	RMCLC MIN4	RMCLC MIN3	RMCLC MIN2	RMCLC MIN1	RMCLC MIN0
Read/Write	R/W							
After reset	0							
Function	Minimum cycle of leader detection: $RMCLC_{MIN} \times 4 / fs[s]$							
	15	14	13	12	11	10	9	8
bit Symbol	RMCLL MAX7	RMCLL MAX6	RMCLL MAX5	RMCLL MAX4	RMCLL MAX3	RMCLL MAX2	RMCLL MAX1	RMCLL MAX0
Read/Write	R/W							
After reset	0							
Function	Maximum low width of leader detection: $RMCLL_{MAX} \times 4 / fs[s]$							
	7	6	5	4	3	2	1	0
bit Symbol	RMCLL MIN7	RMCLL MIN6	RMCLL MIN5	RMCLL MIN4	RMCLL MIN3	RMCLL MIN2	RMCLL MIN1	RMCLL MIN0
Read/Write	R/W							
After reset	0							
Function	Minimum low width of leader detection: $RMCLL_{MIN} \times 4 / fs[s]$							

- <RMCLC_{MAX}7:0>: Specifies a maximum cycle of leader detection.
Calculating formula of the maximum cycle: $RMCLC_{MAX} \times 4 / fs[s]$.
RMC detects the first cycle as a leader if it is within the maximum cycle.
- <RMCLC_{MIN}7:0>: Specifies a minimum cycle of leader detection.
Calculating formula of the minimum cycle: $RMCLC_{MIN} \times 4 / fs[s]$.
RMC detects the first cycle as a leader if it exceeds the minimum cycle.
- <RMCLL_{MAX}7:0>: Specifies a maximum low width of leader detection.
Calculating formula of the maximum low width: $RMCLL_{MAX} \times 4 / fs[s]$.
RMC detects the first cycle as a leader if its low width is within the maximum low width.
- <RMCLL_{MIN}7:0>: Specifies a minimum low width of leader detection.
Calculating formula of the minimum low width: $RMCLL_{MIN} \times 4 / fs[s]$.
RMC detects the first cycle as a leader if its low width exceeds the minimum low width.
If RMCRCR2<RMCLD> = 1, a value less than the specified is determined as data.

(Note)

When you configure the register, you must follow the rule shown below.

Leader	Rules
Low width + high width	<RMCLCMAX7:0> > <RMCLCMIN7:0> <RMCLLMAX7:0> > <RMCLLMIN7:0> <RMCLCMIN7:0> > <RMCLLMAX7:0>
Only with high width	<RMCLCMAX7:0> > <RMCLCMIN7:0> <RMCLLMAX7:0> = 0x00000000 <RMCLLMIN7:0> = don't care
No leader	<RMCLCMAX7:0> = 0x00000000 <RMCLCMIN7:0> = don't care <RMCLLMAX7:0> = don't care <RMCLLMIN7:0> = don't care

12.2.8 Remote Control Receive Control Register 2 [RMCR2]

	31	30	29	28	27	26	25	24
bit Symbol	RMCLIEN	RMCEDIEN	—	—	—	—	RMCLD	RMCPHM
Read/Write	R/W	R/W	R				R/W	R/W
After reset	0	0	0				0	0
Function	Leader detection interrupt 0: Not generated 1: Generated	Remote control input falling edge interrupt 0: Not generated 1: Generated	"0" is read.				Receiving remote control signal with or without leader 0: Disabled 1: Enabled	Receive a remote control signal in phase method? 0: No (receive in cycle method) 1: Yes
	23	22	21	20	19	18	17	16
bit Symbol	—	—	—	—	—	—	—	—
Read/Write	R							
After reset	0							
Function	"0" is read.							
	15	14	13	12	11	10	9	8
bit Symbol	RMCLL7	RMCLL6	RMCLL5	RMCLL4	RMCLL3	RMCLL2	RMCLL1	RMCLL0
Read/Write	R/W							
After reset	1							
Function	Excess low width that triggers reception completion and interrupt generation 00000000~11111110: $RMCLL \times 1/fs[s]$ 11111111: not to use as the trigger							
	7	6	5	4	3	2	1	0
bit Symbol	RMCDMA X7	RMCDMA X6	RMCDMA X5	RMCDMA X4	RMCDMA X3	RMCDMA X2	RMCDMA X1	RMCDMA X0
Read/Write	R/W							
After reset	1							
Function	Maximum data bit cycle that triggers reception completion and interrupt generation 00000000~11111110: $RMCDMAX \times 1/fs[s]$ 11111111: not to use as the trigger							

<RMCLIEN>: Enables to generate a leader detection interrupt by detecting a leader.

<RMCEDIEN>: Enables to generate a remote control input falling edge Interrupt.

<RMCLD>: Enables RMC to receive signals with or without a leader.

<RMCPHM>: Specifies data reception mode of a phase method. If you use the phase method of which signal cycle is fixed, set "1".

<RMCLL7:0>: Specifies an excess low width. If an excess low width is detected, reception is completed and an interrupt is generated. The low width is not detected if <RMCLL7:0> = 11111111b. Calculating formula of an excess low width: $RMCLL \times 1/fs[s]$.

<RMCDMAX7:0>: Specifies a threshold for detecting a maximum data bit cycle. It is detected when a data bit cycle exceeds the threshold. It is not detected when <RMCDMAX7:0> = 11111111b. Calculating formula of the threshold: $RMCDMAX \times 1/fs[s]$.

12.2.9 Remote Control Receive Control Register 3 [RMCR3]

	15	14	13	12	11	10	9	8
bit Symbol	—	RMCDAT H6	RMCDAT H5	RMCDAT H4	RMCDAT H3	RMCDAT H2	RMCDAT H1	RMCDAT H0
Read/Write	R	R/W						
After reset	0	0						
Function	"0" is read.	Larger threshold to determine a signal pattern in a phase method $RMCDATH \times 1/fs[s]$						
	7	6	5	4	3	2	1	0
bit Symbol	—	RMCDAT L6	RMCDAT L5	RMCDAT L4	RMCDAT L3	RMCDAT L2	RMCDAT L1	RMCDAT L0
Read/Write	R	R/W						
After reset	0	0						
Function	"0" is read.	Threshold to determine 0 or 1/ smaller threshold to determine a signal pattern in a phase method $RMCDATL \times 1/fs[s]$						

<RMCDATH6:0>: Specifies a larger threshold (within a range of 1.5T and 2T) to determine a pattern of remote control signal in a phase method. If the measured cycle exceeds the threshold, the bit is determined as "10". If not, the bit is determined as "01". Calculating formula of the threshold: $RMCDATH \times 1/fs[s]$.

<RMCDATL6:0>: Specifies two kinds of thresholds: a threshold to determine whether a data bit is 0 or 1; a smaller threshold (within a range of 1T and 1.5T) to determine a pattern of remote control signal in a phase method.
As for the determination of data bit, if the measured cycle exceeds the threshold, the bit is determined as "1". If not, the bit is determined as "0". Calculating formula of the threshold: $RMCDATL \times 1/fs[s]$.
As for the determination of a remote control signal pattern in a phase method, if the measured cycle exceeds the threshold, the bit is determined as "01". If not, the bit is determined as "00". Calculating formula of the threshold to determine 0 or 1: $RMCDATL \times 1/fs[s]$.

(Note) If the <RMCPHM> bit of the Remote Control Receive Control Register 2 is "0", <RMCDATH6:0> are not enabled. The bits are enabled when <RMCPHM> is "1".

12.2.10 Remote Control Receive Control Register 4 [RMCRCR4]

	7	6	5	4	3	2	1	0
bit Symbol	RMCP0	—	—	—	RMNC3 3	RMNC2 2	RMNC1 1	RMNC0 0
Read/Write	R/W	R			R/W			
After reset	0	0			0			
Function	Remote control input signal 0: Not reversed 1: Reversed	"0" is read.			Noise cancellation time 0000: No cancellation 0001~1111: $RMNC \times 1/f_s[s]$			

<RMCP0>: Specifies whether a remote control input signal is reversed or not.

<RMNC3:0>: Specifies time noises are cancelled by a noise canceller. If <RMNC3:0> = 0000b, noises are not cancelled. Calculating formula of noise cancellation time: $RMNC \times 1/f_s[s]$.

12.2.11 Remote Control Receive Status Register [RMCRSTAT]

	15	14	13	12	11	10	9	8
bit Symbol	RMCRLLIF	RMCLLOIF	RMCDMAXIF	RMCEDIF	—	—	—	—
Read/Write	R	R	R	R	R			
After reset	0	0	0	0	0			
	Leader detection is interrupt factor? 0: No 1: Yes	Low width detection is interrupt factor? 0: No 1: Yes	Maximum data bit cycle detection is interrupt factor? 0: No 1: Yes	Remote control input falling edge interrupt is interrupt factor? 0: No 1: Yes	"0" is read.			
	7	6	5	4	3	2	1	0
bit Symbol	RMCRLLDR	RMCRNUM6	RMCRNUM5	RMCRNUM4	RMCRNUM3	RMCRNUM2	RMCRNUM1	RMCRNUM0
Read/Write	R	R						
After reset	0	0						
Function	Leader detection 0: No 1: Yes	The number of received data bit 0000000: no data bit (only with leader) 0000001~1001000: 1~72bit 1001001~1111111: 73bit and more						

<RMCRLLIF>: Indicates that leader detection is the interrupt factor.

<RMCLLOIF>: Indicates that low width detection is the interrupt factor.

<RMCDMAXIF>: Indicates that maximum data bit cycle detection is the interrupt factor.

<RMCEDIF>: Indicates that a remote control input falling edge interrupt is the interrupt factor.

<RMCRLLDR>: Detects a leader of a received remote control signal

<RMCRNUM6:0>: Indicates the number of bits received as remote control signal data. The number cannot be monitored during reception. On completion of reception, the number is stored.

(Note 1) This register is updated every time an interrupt is generated.

Writing to this register is ignored.

(Note 2) RMC keeps receiving 73 bit or more data unless reception is completed by detecting the maximum data bit cycle or the excess low width. If so, the received data in the data buffer may not be correct.

12.3 Operation Description

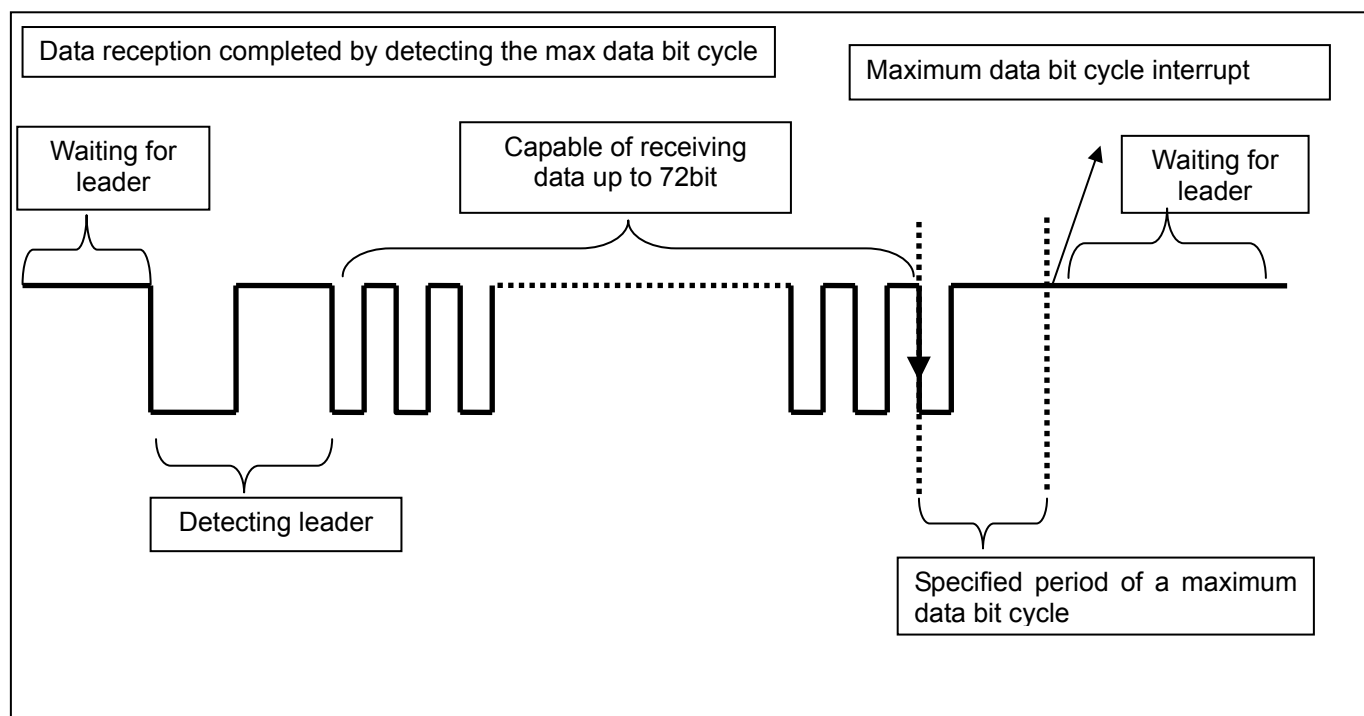
12.3.1 Reception of Remote Control Signal

12.3.1.1 Sampling Clock

A remote control signal is sampled by low-speed clock (fs).

12.3.1.2 Basic Operation

RMC starts to receive a data bit if a leader is detected while RMC is waiting for a leader. Based on a falling edge cycle, the data bit is determined as 0 or 1. By detecting a leader while RMC is waiting for a leader, a leader detection interrupt is generated, and the data bit reception starts. The data bit is determined as 0 or 1 based on a falling edge cycle. RMC is capable of receiving data up to 72bit. Reception is completed by detecting either a maximum data bit cycle or the excess low width. On completion of reception, RMC is waiting for the next leader, and the Remote Control Receive Data Buffer Registers and the Remote Control Receive Status Register are updated.



12.3.1.3 Preparation

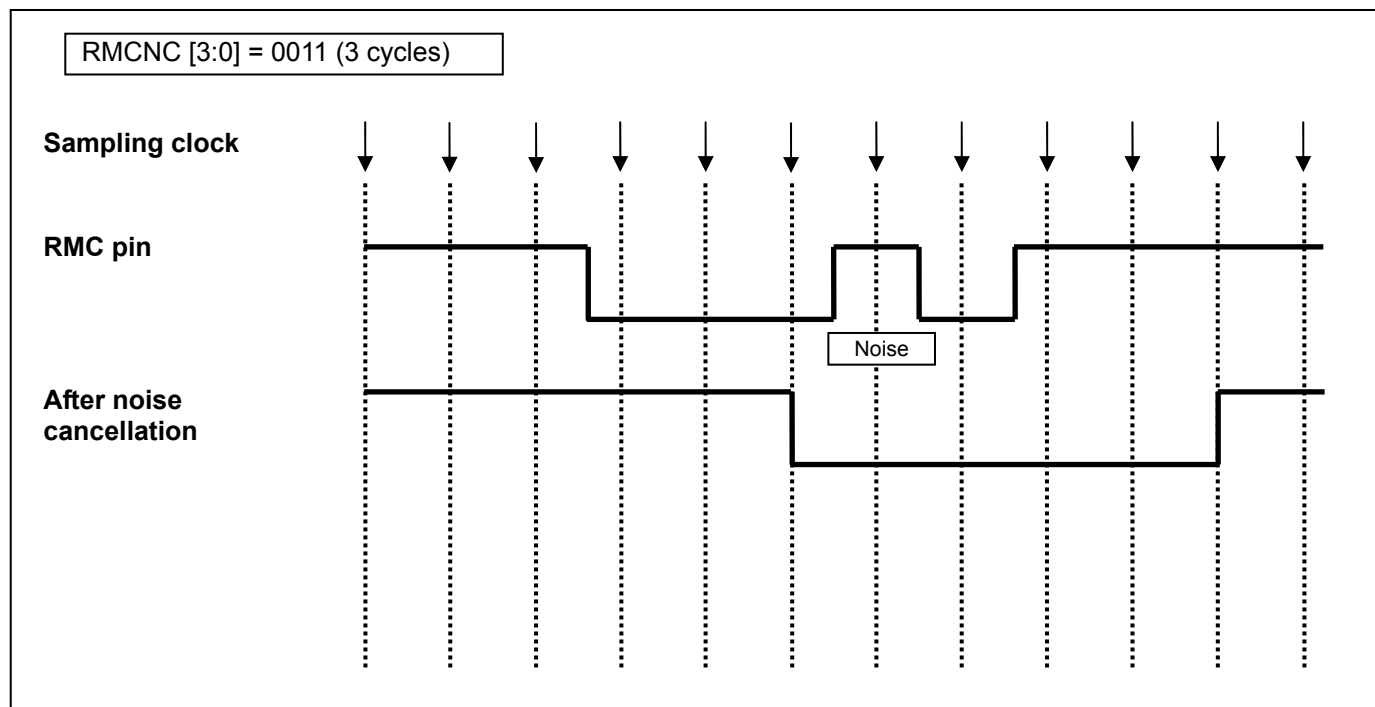
Configure reception operation of a remote control signal with the Remote Control Signal Receive Control Registers (RMCRCR1, RMCRCR 2 and RMCRCR3) before reception.

(1) Settings of Noise Cancelling Time

Configure noise cancelling time with the RMCRCR4 <RMCNC3:0> bit.

RMC monitors a remote control signal in each rising edge of a sampling clock. If “1” is monitored, RMC recognizes that the signal was changed to “0” after monitoring cycles of “0”s specified in RMCNC. If “0” is monitored, RMC recognizes that the signal was changed to “1” after monitoring cycles of “1”s specified in RMCNC.

The following figure shows how RMC operates according to the noise cancel setting of RMCNC [3:0] = 0011 (3 cycles). Subsequent to noise cancellation, the signal is changed from “1” to “0” upon monitoring 3 cycles of “0” s, and the signal is changed from “0” to “1” upon monitoring 3 cycles of “1” s.

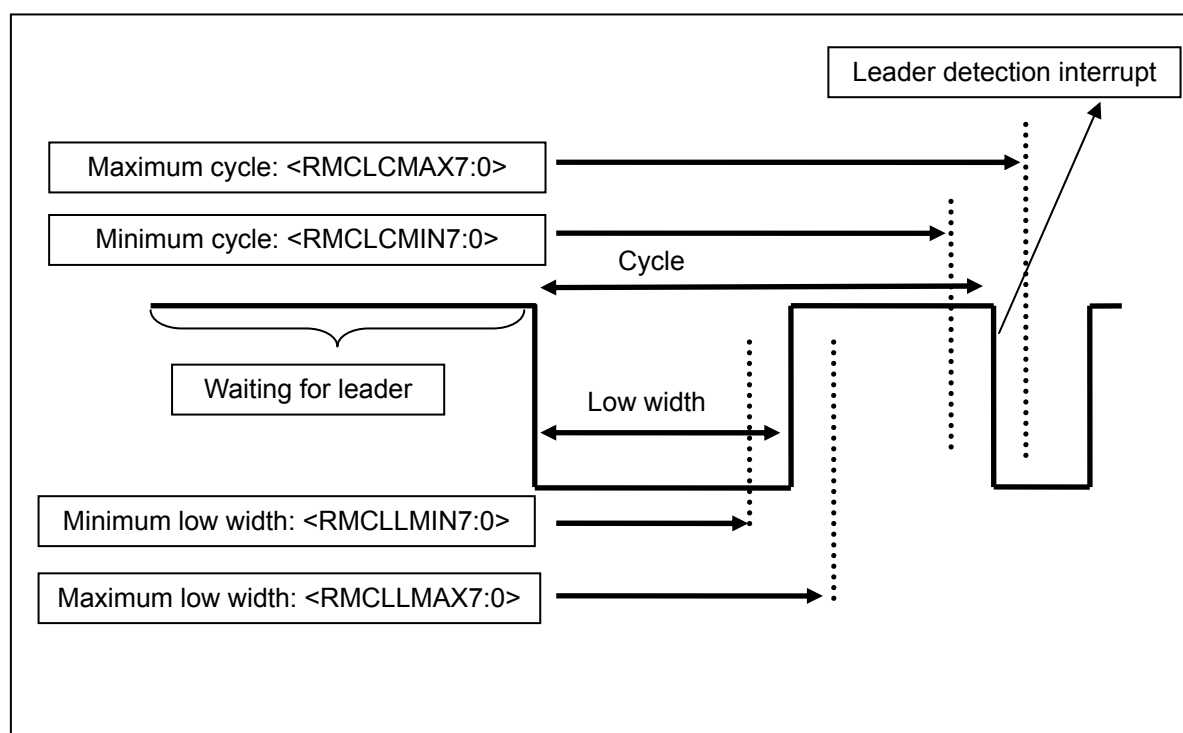


(2) Settings of Detecting Leader

To detect a leader, configure cycle and low width of the leader with the RMCR1 <RMCLLMIN7:0> <RMCLLMAX7:0> <RMCLCMIN7:0> <RMCLCMAX7:0> bits. When you configure the register, you must follow the rule shown below.

Leader	Rules
Low width + High width	<RMCLCMAX7:0> > <RMCLCMIN7:0> <RMCLLMAX7:0> > <RMCLLMIN7:0>
Only with high width	<RMCLCMIN7:0> > <RMCLLMAX7:0> <RMCLCMAX7:0> > <RMCLCMIN7:0> <RMCLLMAX7:0> = 0x00000000 <RMCLLMIN7:0> = don't care
No leader	<RMCLCMAX7:0> = 0x00000000 <RMCLCMIN7:0> = don't care <RMCLLMAX7:0> = don't care <RMCLLMIN7:0> = don't care

The following shows a leader waveform and the RMCR1 register settings.



If you want to generate an interrupt when detecting a leader, configure the RMCR2 <RMCLIEN> bit. A remote control signal without a leader cannot generate a leader detection interrupt.

(3) Settings of Data Bit Determination

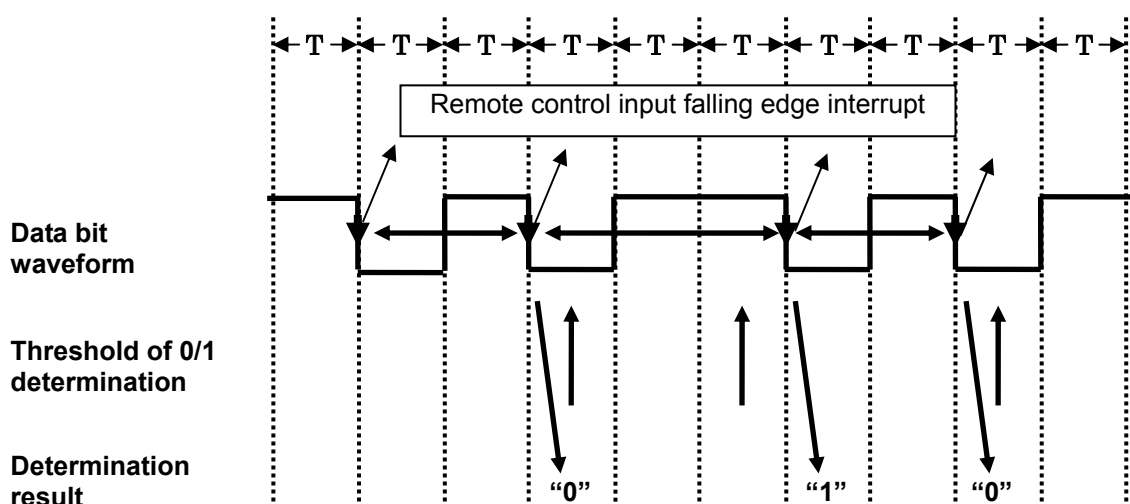
Based on a falling edge cycle, the data bit is determined as 0 or 1.

Configure a threshold of the determination with the RMCRCR3 <RMCDATL6:0> bit. If the cycle exceeds the threshold, the bit is determined as "1". If not, the bit is determined as "0".

By setting "1" to the RMCRCR2 <RMCEDIEN> bit, a remote control signal input falling edge interrupt can be generated in each falling edge of the data bit. Using this interrupt together with a 16-bit timer enables the determination to be done by software.

The following shows how the data bit is determined as "0" or "1".

Example: Threshold of 0/1 determination is set to 2.5T with the <RMCDATL6:0> bit.



As for data bit determination of a remote control signal in a phase method, see 12.3.1.10 "Receiving a Remote Control Signal in a Phase Method".

(4) Settings of Reception Completion

To complete data reception, settings of detecting the maximum data bit cycle and excess low width are required. If multiple factors are specified, reception is completed by the factor detected first. Make sure to configure the reception completion settings.

1) Completed by a maximum data bit cycle

To complete reception by detecting a maximum data bit cycle, you need to configure the RMCRCR2 <RMCDMAX7:0> bits. If the falling edge of the data bit cycle isn't monitored after time specified as threshold in the <RMCDMAX7:0> bits, a maximum data bit cycle is detected. The detection completes reception and generates an interrupt.

2) Completed by excess low width

To complete reception by detecting the low width, you need to configure the RMCRCR2 <RMCLL7:0> bits. After the falling edge of the data bit is detected, if the signal stays low longer than specified, excess low width is detected. The detection completes reception and generates an interrupt.

12.3.1.4 Enabling Reception

By enabling the RMCREN <RMCREN> bit after configuring the RMCRCR1, RMCRCR2 and RMCRCR3 registers, RMC is ready for reception. Detecting a leader initiates reception.

(Note) Changing the configurations of the RMCRCR1, RMCRCR2 and RMCRCR3 registers during reception may harm their proper operation. Be careful if you change them during reception.

12.3.1.5 Reception

Detecting a leader sets the RMCRCRSTAT <RMCRLDR> bit. Simultaneously, a leader detection interrupt is generated if the RMCRCR2 <RMCLIEN> bit is set. When the interrupt is generated, the RMCRCRSTAT <RMCRLIF> bit is set.

Next to the leader detection, each data bit is determined as 0 or 1. The results are stored in the RMCRCRBUF1, RMCRCRBUF 2 and RMCRCRBUF 3 registers up to 72bit. By setting "1" to the RMCRCR2 <RMCEDIEN> bit, a remote control signal input falling edge interrupt can be generated in each falling edge of the data bit. When the interrupt is generated, the RMCRCRSTAT <RMCEDIF> bit is set.

Detecting the maximum data bit cycle or the excess low width completes reception and generates an interrupt.

To check the status of RMC after reception is completed, read the Remote Control Receive Status Register.

On completion of reception, RMC is waiting for the next leader.

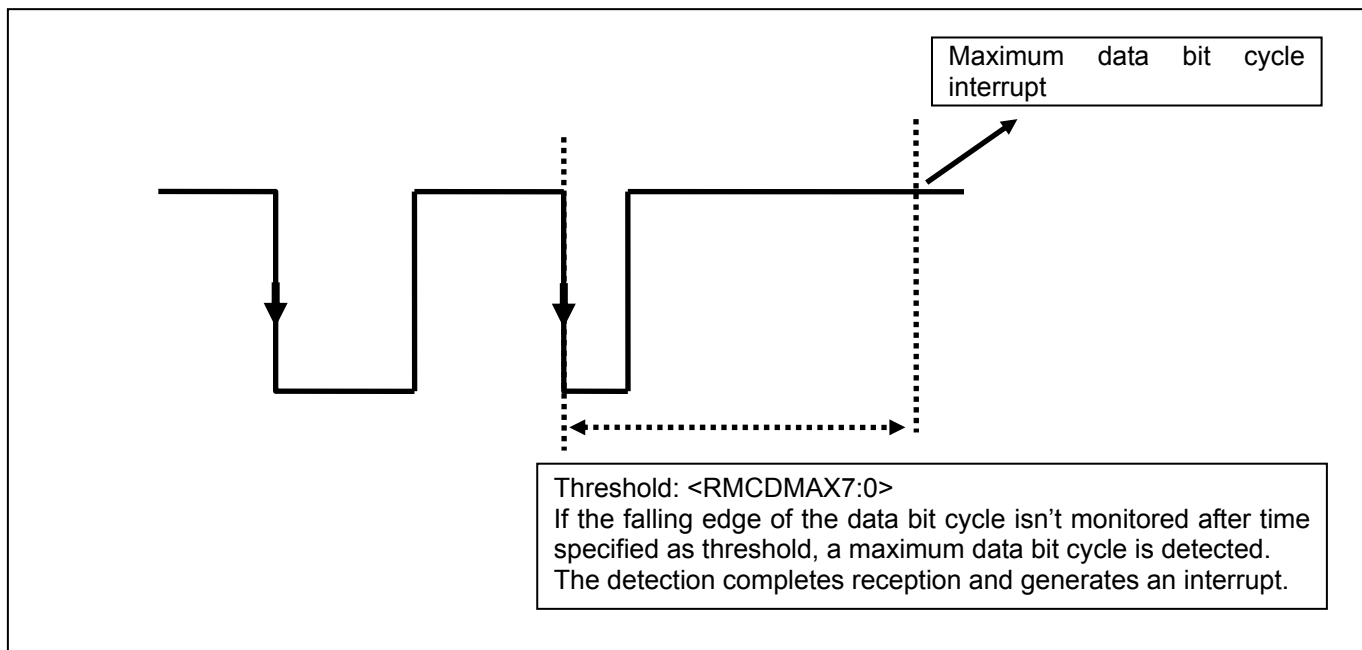
By setting RMC to receive a signal without a leader, RMC recognizes the received data and starts reception without detecting a leader.

If the next data reception is completed before reading the preceding received data, the preceding data is overwritten by the next one.

12.3.1.6 Reception Completion

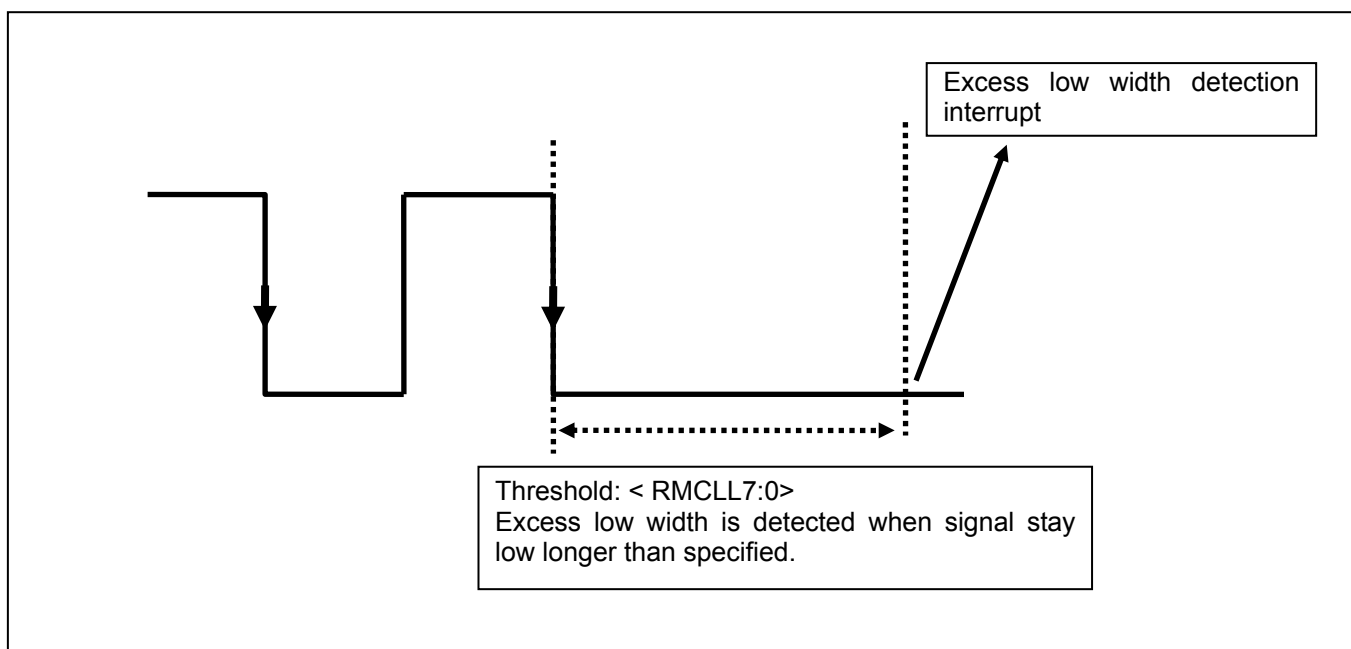
1) Completed by a maximum data bit cycle

Detecting a maximum data bit cycle completes reception and generates an interrupt. After the interrupt is generated, the RMCRSTAT <RMCDMAXIF> bit is set to "1".



2) Completed by excess low width

Detecting excess low width completes reception and generates an interrupt. After the interrupt is generated, the RMCRSTAT <RMCLOIF> bit is set to "1".



RMC keeps receiving 73 bit or more data unless reception is completed by detecting the maximum data bit cycle or the excess low width. If so, the received data in the data buffer may not be correct.

To check the status of RMC after reception is completed, read the Remote Control Receive Status Register. The status of RMC that each bit type indicates is shown below.

<RMCRLDR>	<RMCRNUM6:0>	RMC Status
0	0000001~1001000	Receiving remote control signal without a leader (Data bits: 1~72bit)
0	1001001~1111111	Receiving remote control signal without a leader (Data bits: 73bit and more)
1	0000000	Only with a leader
1	0000001~1001000	Receiving remote control signal with a leader (Data bits: 1~72bit)
1	1001001~1111111	Receiving remote control signal without a leader (Data bits: 73bit and more)

12.3.1.7 Stopping Reception

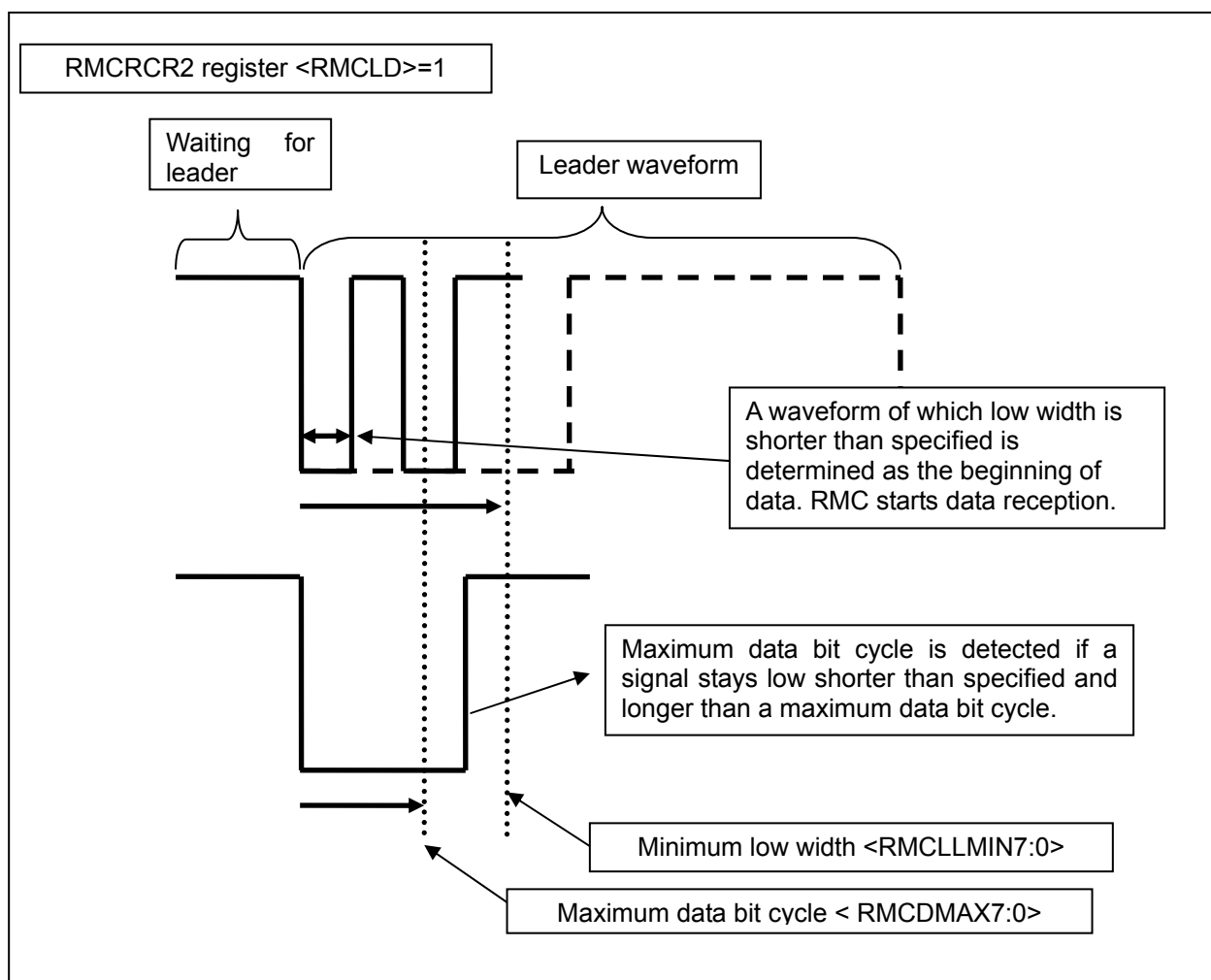
RMC stops reception by clearing the RMCREN <RMCREN> bit to "0" (reception disabled).

Clearing this bit during reception stops reception immediately and the received data is discarded.

12.3.1.8 Receiving Remote Control Signal without Leader

Setting RMCRCR2 <RMCLD> enables RMC to receive signals with or without a leader. By setting RMCRCR2 <RMCLD>, RMC starts receiving data if it recognizes a signal of which low width is shorter than a maximum low width of leader detection specified in the RMCRCR1 <RMCLLMAX7:0> bits. RMC keeps receiving data until the final data bit is received.

If RMCRCR2 <RMCLD> is enabled, the same settings of error detection, reception completion and data bit determination of 0 or 1 are applied regardless of whether a signal has a leader or not. Thus receivable remote control signals are limited.



12.3.1.9 A Leader only with Low Width

The figure shown below illustrates a remote control signal that starts with a leader of which waveform only has low width. This signal starts with a leader that only has low width and a data bit cycle starts from the rising edge. To enable the signal, it must be sent after being reversed by setting the RMCRCR4 <RMCPO> bit to "1". This is because RMC is configured to detect a data bit cycle from the falling edge.

A leader is detected by the low width. When you configure the RMCRCR1 register, you must follow the rule shown below.

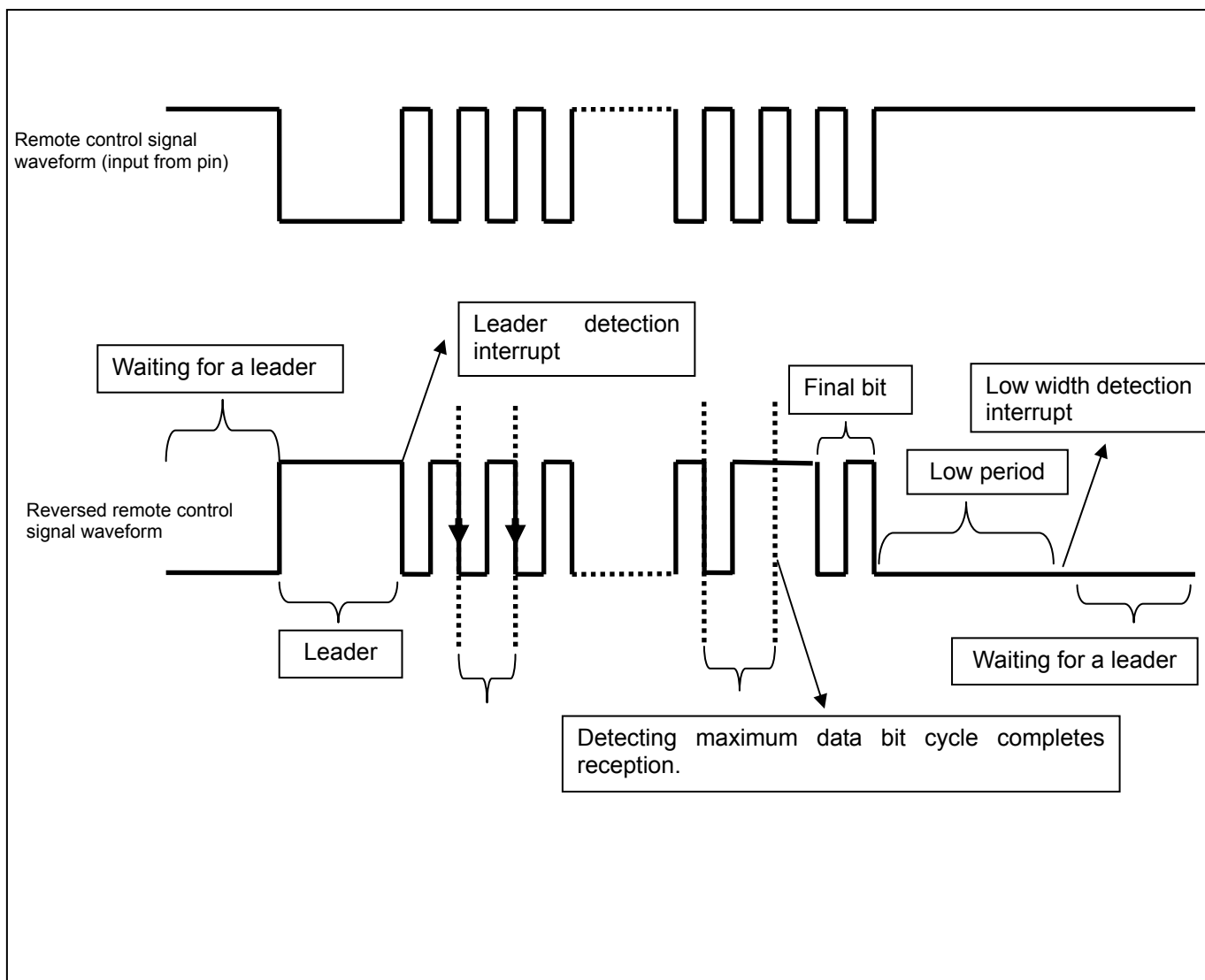
<RMCLLMAX7:0> = 0x00000000
<RMCLHMAX7:0> > <RMCLHMIN7:0>

If the rules are applied, RMC does not care about the value of <RMCLLMIN7:0>.

To determine the data bit as 0 or 1, configure a threshold of the determination with the RMCRCR3 <RMCDATL6:0> bit.

Configure a maximum data bit cycle with the <RMCDMAX7:0> bits of the Remote Control Receive Control Register 2.

To complete reception by detecting the maximum data bit cycle, you need to configure the RMCRCR2 <RMCDMAX7:0> bits. To complete reception by detecting the low width, you need to configure the RMCRCR2 <RMCLL7:0> bits. Detecting the maximum data bit cycle or the excess low width completes reception and generates an interrupt. RMC waits for the next leader.



12.3.1.10 Receiving a Remote Control Signal in a Phase Method

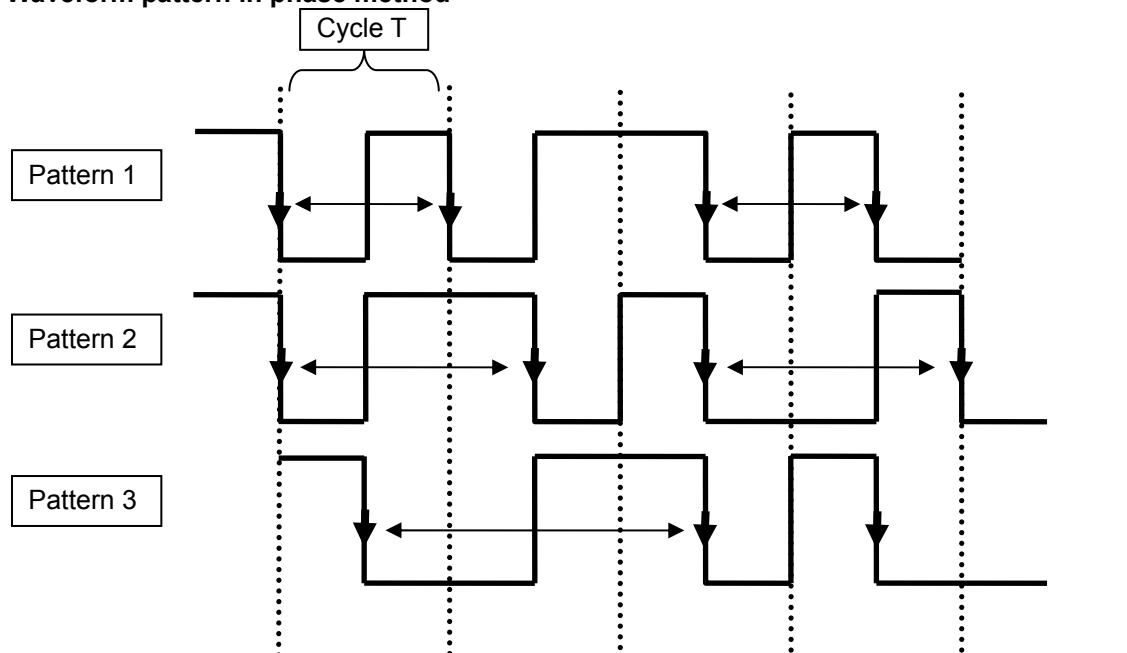
RMC is capable of receiving a remote control signal in a phase method of which signal cycle is fixed. A signal in the phase method has three waveform patterns (see the figure shown below). By setting two thresholds a remote control signal pattern is determined. RMC converts the signal into data "0" or "1". On completion of reception, received data "0" and "1" are stored in the RMCRCR2, RMCRCR3 and RMCRCR4 registers.

By setting $\text{RMCRCR2}[\text{RMCPPH}] = 1$, RMC enables to receive a remote control signal in the phase method. Each threshold can be configured with the $\text{RMCRCR4}[\text{RMCDDL6:0}]$ bits and $\text{RMCRCR4}[\text{RMCDDH6:0}]$ bits. Two thresholds are used to distinguish three waveform patterns. On condition that a cycle between two falling edges is "T", three patterns show cycles of 1T, 1.5T and 2T. Details of the two thresholds are shown below.

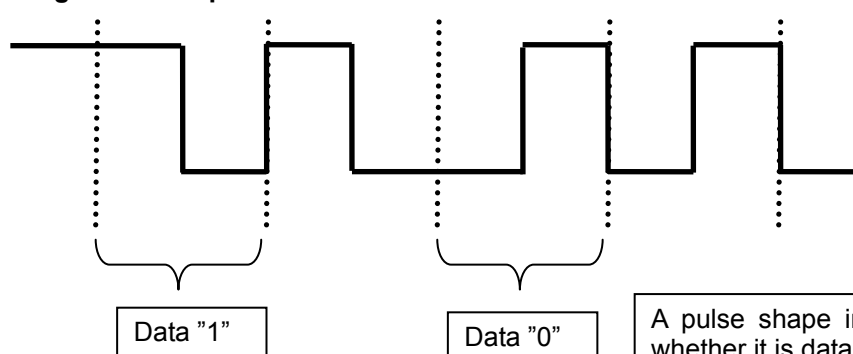
To determine a remote control signal in the phase method, three patterns of data waveform and preceding data are required. In addition, the signal needs to start from data "1".

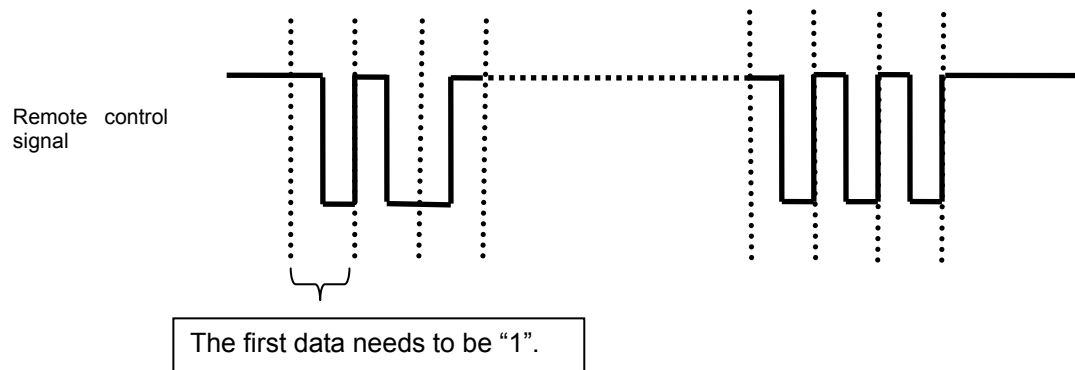
	Determined by	Threshold	Register bits to set
Threshold 1	Pattern 1 & pattern 2	1T~1.5T	RMCRCR2 register <RMCDDL6:0>
Threshold 2	Pattern 2 & pattern 3	1.5T~2T	RMCRCR2 register <RMCDDH6:0>

Waveform pattern in phase method



Remote control signal data in phase method



Remote control signal in phase method

13. Analog/Digital Converter

A 10-bit, sequential-conversion analog/digital converter (A/D converter) is built into the TPM330. This A/D converter is equipped with 12 analog input channels.

Fig. 13-1 shows the block diagram of this A/D converter.

These 12 analog input channels (pins AN0 through AN11) are also used as input/ output ports.

(Note) If it is necessary to reduce a power current by operating the TMPM330 in IDLE or STOP mode and if either case shown below is applicable, you must first stop the A/D converter and then execute the instruction to put the TMPM330 into standby mode:

- 1) The TPM330 must be put into IDLE mode when ADMOD1<I2AD> is "0."
- 2) The TPM330 must be put into STOP mode.

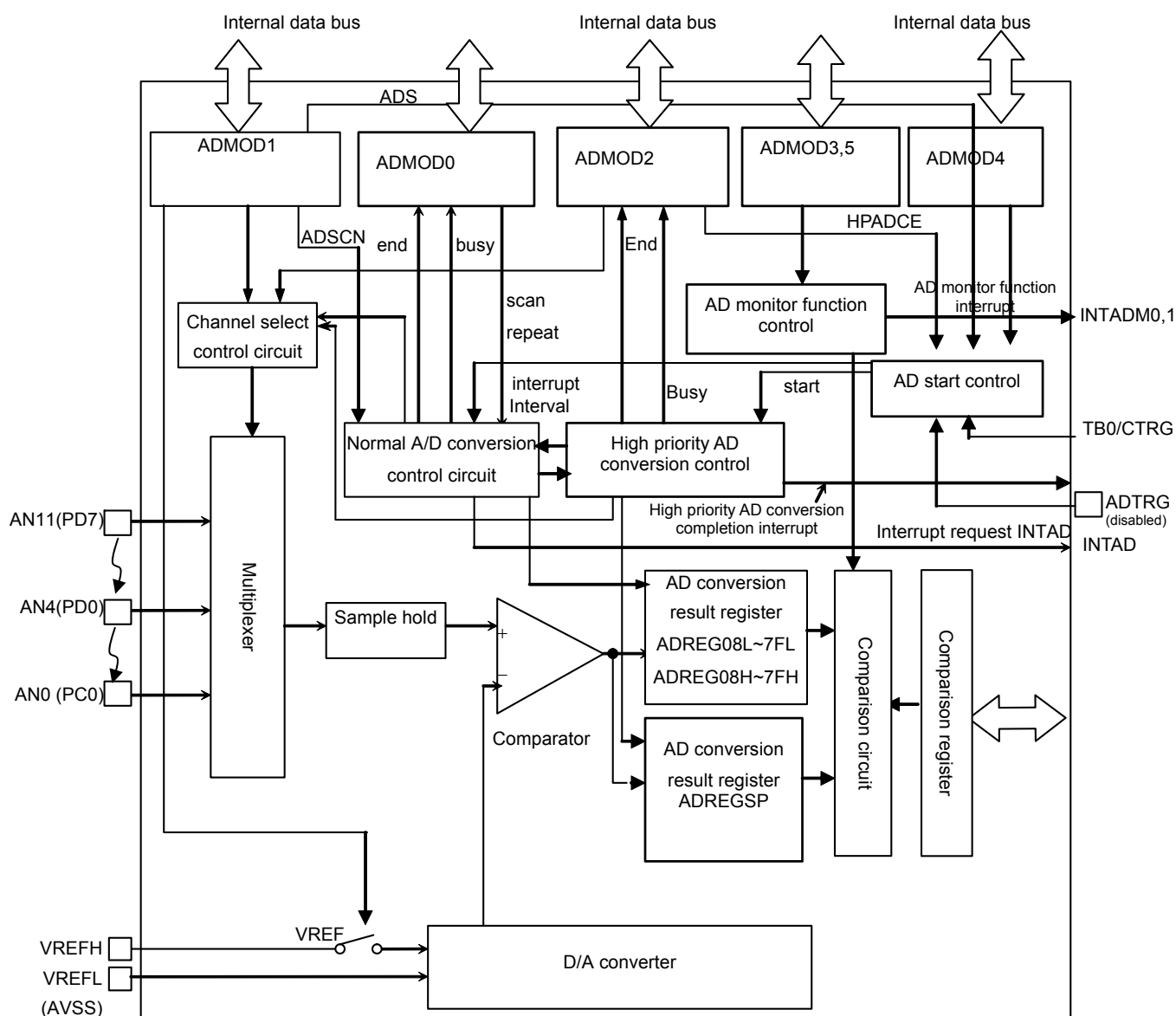


Fig. 13-1 A/D Converter Block Diagram

13.1 Registers

The control registers and addresses of the A/D converter are as follows.

Register name		Address
A/D Conversion Clock Setting Register	ADCLK	0x4003_0000
A/D Mode Control Register 0	ADMOD0	0x4003_0004
A/D Mode Control Register 1	ADMOD1	0x4003_0008
A/D Mode Control Register 2	ADMOD2	0x4003_000C
A/D Mode Control Register 3	ADMOD3	0x4003_0010
A/D Mode Control Register 4	ADMOD4	0x4003_0014
A/D Mode Control Register 5	ADMOD5	0x4003_0018
Lower A/D Conversion Result Register 08L	ADREG08L	0x4003_0030
Upper A/D Conversion Result Register 08H	ADREG08H	0x4003_0031
Lower A/D Conversion Result Register 19L	ADREG19L	0x4003_0034
Upper A/D Conversion Result Register 19H	ADREG19H	0x4003_0035
Lower A/D Conversion Result Register 2AL	ADREG2AL	0x4003_0038
Upper A/D Conversion Result Register 2AH	ADREG2AH	0x4003_0039
Lower A/D Conversion Result Register 3BL	ADREG3BL	0x4003_003C
Upper A/D Conversion Result Register 3BH	ADREG3BH	0x4003_003D
Lower A/D Conversion Result Register 4CL	ADREG4CL	0x4003_0040
Upper A/D Conversion Result Register 4CH	ADREG4CH	0x4003_0041
Lower A/D Conversion Result Register 5DL	ADREG5DL	0x4003_0044
Upper A/D Conversion Result Register 5DH	ADREG5DH	0x4003_0045
Lower A/D Conversion Result Register 6EL	ADREG6EL	0x4003_0048
Upper A/D Conversion Result Register 6EH	ADREG6EH	0x4003_0049
Lower A/D Conversion Result Register 7FL	ADREG7FL	0x4003_004C
Upper A/D Conversion Result Register 7FH	ADREG7FH	0x4003_004D
Lower A/D Conversion Result Register SP	ADREGSPL	0x4003_0050
Upper A/D Conversion Result Register SP	ADREGSPH	0x4003_0051
Lower A/D Conversion Result Comparison Register 0	ADCMP0L	0x4003_0054
Upper A/D Conversion Result Comparison Register 0	ADCMP0H	0x4003_0055
Lower A/D Conversion Result Comparison Register 1	ADCMP1L	0x4003_0058
Upper A/D Conversion Result Comparison Register 1	ADCMP1H	0x4003_0059

13.2 Registers Description

The A/D converter is controlled by A/D mode control registers (ADMOD0, ADMOD1, ADMOD2, ADMOD3, ADMOD4 and ADMOD5). Results of A/D conversion are stored in 16 upper and lower A/D conversion result registers ADREG08H/L through ADREG7FH/L. Results of top-priority conversion are stored in ADREGSPH/L.

Here are the descriptions of the registers.

A/D Mode Control Register 0								
ADMOD0	Bit symbol	EOCFN	ADBFN	ITM1	ITM0	REPEAT	SCAN	ADS
	Read/Write	R		R/W				
	After reset	0	0	0	0	0	0	0
	Function	Normal A/D conversion completion flag 0: Before or during conversion 1: Completion	Normal A/D conversion BUSY flag 0: Conversion stop 1: During conversion	"0" is read. Specify interrupt in fixed channel repeat conversion mode	Specify interrupt in fixed channel repeat conversion mode.	Specify repeat mode 0: Single conversion mode 1: Repeat conversion mode	Specify scan mode 0: Fixed channel mode 1: Channel scan mode	Start A/D conversion 0: Don't care 1: Start conversion read.

Specify A/D conversion interrupt in fixed channel repeat conversion mode

	Fixed channel repeat conversion mode <SCAN> = "0", <REPEAT> = "1"
00	Generate interrupt once every single conversion
01	Generate interrupt once every 4 conversions
10	Generate interrupt once every 8 conversions
11	Setting prohibited

Fig. 13-2 A/D Mode Control Register 0

(Note 1) Please specify the mode first and then specify the <ADS> bit.

A/D Mode Control Register 1

ADMOD1

	7	6	5	4	3	2	1	0
Bit symbol	VREFON	I2AD	ADSCN	—	ADCH3	ADCH2	ADCH1	ADCH0
Read/Write	R/W							
After reset	0	0	0	0	0	0	0	0
Function	VREF application control 0 : OFF 1 : ON	IDLE 0 : Stop 1 : Operation	Specify operation mode for channel scanning 0: 4ch scan 1: 8ch scan	Write "0."	Select analog input channel			

Select analog input channel

<ADCH3.2, 1, 0> <SCAN>	0	1	1
	Fixed channel	Channel scan (ADSCN=0)	Channel scan (ADSCN= 1)
0000	AN0	AN0	AN0
0001	AN1	AN0~AN1	AN0~AN1
0010	AN2	AN0~AN2	AN0~AN2
0011	AN3	AN0~AN3	AN0~AN3
0100	AN4	AN4	AN0~AN4
0101	AN5	AN4~AN5	AN0~AN5
0110	AN6	AN4~AN6	AN0~AN6
0111	AN7	AN4~AN7	AN0~AN7
1000	AN8	AN 8	AN8
1001	AN9	AN8~AN9	AN8~AN9
1010	AN10	AN8~AN10	AN8~AN10
1011	AN11	AN8~AN11	AN8~AN11
1101	Setting prohibited		
1110			
1111			

(Note 1) Before starting AD conversion, write "1" to the <VREFON> bit, wait for 3 μ s during which time the internal reference voltage should stabilize, and then write "1" to the ADMOD0<ADS> bit.

(Note 2) To go into standby mode upon completion of AD conversion, set <VREFON> to "0."

Fig. 13-3 A/D Mode Control Register 1

ADMOD2

A/D Mode Control Register 2								
	7	6	5	4	3	2	1	0
Bit symbol	EOCFHP	ADBFHP	HPADCE	—	HPADCH3	HPADCH2	HPADCH1	HPADCH0
Read/Write	R	R	R/W					
After reset	0	0	0	0	0	0	0	0
Function	Top-priority AD conversion completion flag 0: Before or during conversion 1: Upon completion	Top-priority AD conversion BUSY flag 0: During conversion halts 1: During conversion	Activate top-priority conversion 0: Don't care 1: Start conversion. "0" is always read.	Write "0".	Select analog input channel when activating top-priority conversion.			

	Analog input channel when executing top-priority conversion
<HPADCH4,3,2, 1, 0>	
0000	AN0
0001	AN1
0010	AN2
0011	AN3
0100	AN4
0101	AN5
0110	AN6
0111	AN7
1000	AN8
1001	AN9
1010	AN10
1011	AN11
1100	AN12
1101	Setting prohibited
1110	
1111	

Fig. 13-4 A/D Mode Control Register 2

ADMOD3

A/D Mode Control Register 3								
	7	6	5	4	3	2	1	0
Bit symbol			ADOBIC	REGS3	REGS2	REGS1	REGS0	ADOBSV
Read/Write	R/W	R	R/W					
After reset	0	0	0	0	0	0	0	0
Function	Write "0".	"0" can be read.	Make AD monitor function interrupt setting 0: Smaller than comparison Reg. 1: Larger than comparison Reg.	BIT for selecting the AD conversion result storage Reg. that is to be compared with the comparison Reg. if the AD monitor function is enabled			AD monitor function 0 : Disable 1 : Enable	

<REGS.2, 1, 0>	AD conversion result storage Reg. to be compared
0000	ADREG08
0001	ADREG19
0010	ADREG2A
0011	ADREG3B
0100	ADREG4C
0101	ADREG5D
0110	ADREG6E
0111	ADREG7F
1XXX	ADREGSP

Fig. 13-5 A/D Mode Control Register 3

A/D Mode Control Register 5

	7	6	5	4	3	2	1	0
Bit symbol			ADOBIC	REGS3	REGS2	REGS1	REGS0	ADOBSV
Read/Write	R/W	R	R/W					
After reset	0	0	0	0	0	0	0	0
Function	Write "0".	"0" can be read.	Make AD monitor function interrupt setting Smaller than comparison Reg. Larger than comparison Reg.	BIT for selecting the AD conversion result storage Reg. that is to be compared with the comparison Reg. if the AD monitor function is enabled .				AD monitor function 0 : Disable 1 : Enable

<REGS.2, 1, 0>	AD conversion result storage Reg. to be compared
0000	ADREG08
0001	ADREG19
0010	ADREG2A
0011	ADREG3B
0100	ADREG4C
0101	ADREG5D
0110	ADREG6E
0111	ADREG7F
1XXX	ADREGSP

Fig. 13-6 A/D Mode Control Register 5

		A/D Mode Control Register 4							
		7	6	5	4	3	2	1	0
ADMOD4	Bit symbol	HADHS	HADHTG	ADHS	ADHTG			ADRST1	ADRST0
	Read/Write	RW				R		W	W
	After reset	0	0	0	0	0		—	—
	Function	HW source for activating top-priority A/D conversion 0: External TRG 1: Match with TB2RG0		HW for activating top-priority A/D conversion 0: Disable 1: Enable		HW source for activating normal A/D conversion 0: External TRG 1: Match with TB1RG0		HW for activating normal A/D conversion 0: Disable 1: Enable	
								Overwriting 10 with 01 allows ADC to be software reset.	

(Note 1) If AD conversion is executed with the match triggers <ADHTG> and <HADHTG> of a 16-bit timer set to "1" by using a source for triggering H/W, A/D conversion can be activated at specified intervals by performing three steps shown below when the timer is idle:

1. Select a source for triggering HW: <ADHS>, <HADHS>
2. Enable H/W activation of AD conversion: <ADHTG>, <HADHTG>
3. Start the timer.

(Note 2) Do not make a top-priority AD conversion setting and a normal AD conversion setting simultaneously.

(Note 3) The external trigger cannot be used for H/W activation of AD conversion when it is used for H/W activation of top priority AD conversion.

(Note 4) A software reset initializes other bits. Resetting a mode register is needed.

(Note) The TMPM330 disables the external trigger used for H/W activation. Therefore "0" cannot be set to <HADHS> and <ADHS>.

Fig. 13-7 A/D Mode Control Register 4

Lower A/D Conversion Result Register 08

	7	6	5	4	3	2	1	0
Bit symbol	ADR01	ADR00					OVR0	ADR0RF
Read/Write	R		R				R	R
After reset	0		1				0	0
Function	Store lower 2 bits of A/D conversion result		"1" can be read.				Over RUN flag 0: Not generate 1: Generate	A/D conversion result storage flag 1: Presence of conversion result

Upper A/D Conversion Result Register 08

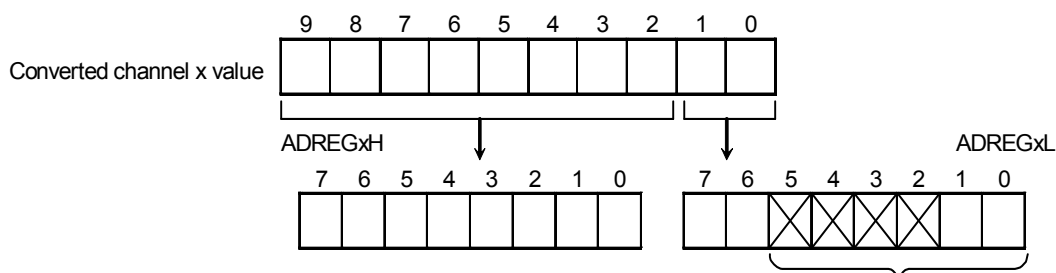
	7	6	5	4	3	2	1	0
Bit symbol	ADR09	ADR08	ADR07	ADR06	ADR05	ADR04	ADR03	ADR02
Read/Write	R							
After reset	0							
Function	Store upper 8 bits of A/D conversion result							

Lower A/D Conversion Result Register 19

	7	6	5	4	3	2	1	0
Bit symbol	ADR11	ADR10					OVR1	ADR1RF
Read/Write	R		R				R	R
After reset	0		1				0	0
Function	Store lower 2 bits of A/D conversion result		"1" can be read.				Over RUN flag 0: Not generate 1: Generate	A/D conversion result storage flag 1: Presence of conversion result

Upper A/D Conversion Result Register 19

	7	6	5	4	3	2	1	0
Bit symbol	ADR19	ADR18	ADR17	ADR16	ADR15	ADR14	ADR13	ADR12
Read/Write	R							
After reset	0							
Function	Store upper 8 bits of A/D conversion result							



- Values of ADREG08L/ADREG19L read from bits 5 through 2 are always "1".
- Bit 0 of ADREG08L/ADREG19L is the A/D conversion result storage flag <ADRxRF>. This bit is set to "1" after an A/D converted value is stored. A read of a lower register (ADREGxL) will set this bit to "0".
- Bit 1 of ADREG08L/ADREG19L is the over RUN flag <OVRx>. This bit is set to "1" if a conversion result is overwritten before both conversion result storage registers (ADREGxH and ADREGxL) are read. A read of a flag will clear this bit to "0".
- To read from conversion result storage registers, first read upper registers and then lower registers.

Fig. 13-8 A/D Conversion Result Register

Lower A/D Conversion Result Register 2A

	7	6	5	4	3	2	1	0
ADREG2AL	ADR21	ADR20					OVR2	ADR2RF
Read/Write	R		R				R	R
After reset	0		1				0	0
Function	Store lower 2 bits of A/D conversion result		"1" is read.				Over RUN flag 0: Not generate 1: Generate	A/D conversion result storage flag 1: Presence of conversion result

Upper A/D Conversion Result Register 2A

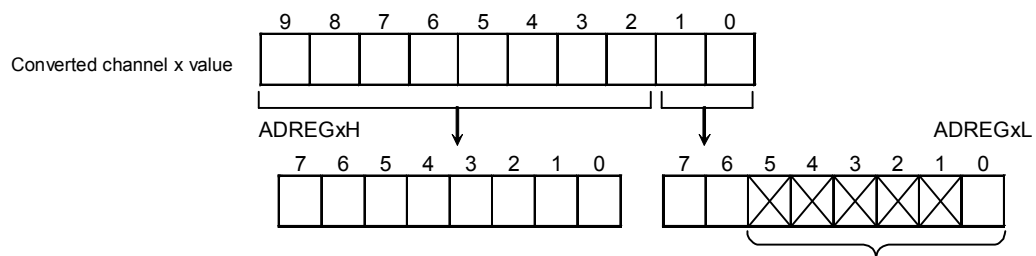
	7	6	5	4	3	2	1	0
ADREG2AH	ADR29	ADR28	ADR27	ADR26	ADR25	ADR24	ADR23	ADR22
Read/Write	R							
After reset	0							
Function	Store upper 8 bits of A/D conversion result							

Lower A/D Conversion Result Register 3B

	7	6	5	4	3	2	1	0
ADREG3BL	ADR31	ADR30					OVR3	ADR3RF
Read/Write	R		R				R	R
After reset	0		1				0	0
Function	Store lower 2 bits of A/D conversion result		"1" is read.				Over RUN flag 0: Not generate 1: Generate	A/D conversion result storage flag 1: Presence of conversion result

Upper A/D Conversion Result Register 3B

	7	6	5	4	3	2	1	0
ADREG3BH	ADR39	ADR38	ADR37	ADR36	ADR35	ADR34	ADR33	ADR32
Read/Write	R							
After reset	0							
Function	Store upper 8 bits of A/D conversion result							



- Values read from bits 5 through 2 of the ADREG2AL/ADREG3BL are always "1".
- Bit 0 of the ADREG2AL/ADREG3BL is the A/D conversion result storage flag <ADR_xRF>. It is set to "1" after an A/D converted value is stored. A read of a lower register (ADREG_xL) will set this bit to "0".
- Bit 1 of the ADREG2AL/ADREG3BL is the over RUN flag <OVR_x>. It is set to "1" if a conversion result is overwritten before both conversion result storage registers (ADREG_xH, ADREG_xL) are read. A read of a flag will clear this bit to "0".
- When reading conversion result storage registers, first read upper registers and then read lower registers.

Fig. 13-9 A/D Conversion Result Register

Lower A/D Conversion Result Register 4C

	7	6	5	4	3	2	1	0
ADREG 4 CL	Bit symbol	ADR41	ADR40				OVR4	ADR4RF
	Read/Write	R		R			R	R
	After reset	0		1			0	0
	Function	Store lower 2 bits of A/D conversion result		"1" is read.			Over RUN flag 0: Not generate 1: Generate	A/D conversion result storage flag 1: Presence of conversion result

Upper A/D Conversion Result Register 4C

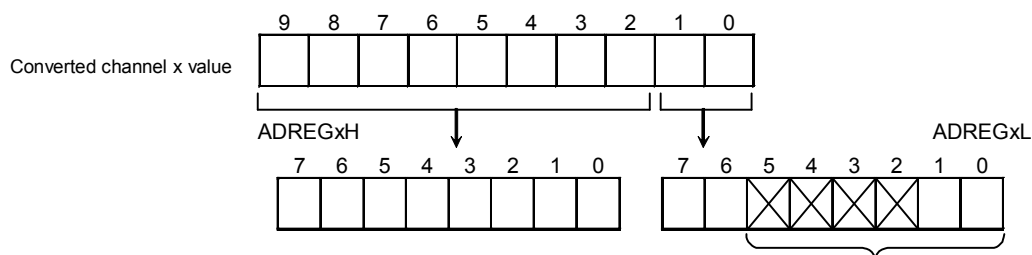
ADREG 4 CL		7	6	5	4	3	2	1	0
	Bit symbol	ADR49	ADR48	ADR47	ADR46	ADR45	ADR44	ADR43	ADR42
	Read/Write	R							
	After reset	0							
	Function	Store upper 8 bits of A/D conversion result							

Lower A/D Conversion Result Register 5D

	7	6	5	4	3	2	1	0
ADREG 4 CL	Bit symbol	ADR51	ADR50				OVR5	ADR5RF
	Read/Write	R		R			R	R
	After reset	0		1			0	0
	Function	Store lower 2 bits of A/D conversion result		"1" is read.			Over RUN flag 0: Not generate 1: Generate	A/D conversion result storage flag 1: Presence of conversion result

Upper A/D Conversion Result Register 5D

ADREG 4 CL		7	6	5	4	3	2	1	0
	Bit symbol	ADR59	ADR58	ADR57	ADR56	ADR55	ADR54	ADR53	ADR52
	Read/Write	R							
	After reset	0							
	Function	Store upper 8 bits of A/D conversion result							



- Values read from bits 5 through 2 of the ADREG4CL/ADREG5DL are always "1".
- Bit 0 of the ADREG4CL/ADREG5DL is the A/D conversion result storage flag <ADRxRF>. It is set to "1" after an A/D converted value is stored. A read of a lower register (ADREGxL) will set this bit to "0".
- Bit 1 of the ADREG4CL/ADREG5DL is the over Run flag <OVRx>. It is set to "1" if a conversion result is overwritten before both conversion result storage registers (ADREGxH and ADREGxL) are read. A read of a flag will clear this bit to "0".
- When reading conversion result storage registers, first read upper registers and then read lower registers.

Fig. 13-10 A/D Conversion Result Register

Lower A/D Conversion Result Register 6E

	7	6	5	4	3	2	1	0
ADREG6EL	ADR61	ADR60					OVR6	ADR6RF
Read/Write	R		R				R	R
After reset	0		1				0	0
Function	Store lower 2 bits of A/D conversion result		"1" is read.				Over RUN flag 0: Not generate 1: Generate	A/D conversion result storage flag 1: Presence of conversion result

Upper A/D Conversion Result Register 6E

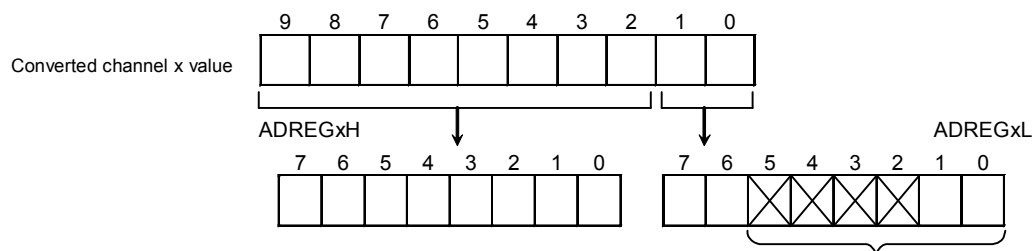
	7	6	5	4	3	2	1	0
ADREG6EL	ADR69	ADR68	ADR67	ADR66	ADR65	ADR64	ADR63	ADR62
Read/Write	R							
After reset	0							
Function	Store upper 8 bits of A/D conversion result							

Lower A/D Conversion Result Register 7F

	7	6	5	4	3	2	1	0
ADREG6EL	ADR71	ADR70					OVR7	ADR7RF
Read/Write	R		R				R	R
After reset	0		1				0	0
Function	Store lower 2 bits of A/D conversion result		"1" is read.				Over RUN flag 0: Not generate 1: Generate	A/D conversion result storage flag 1: Presence of conversion result

Upper A/D Conversion Result Register 7F

	7	6	5	4	3	2	1	0
ADREG6EL	ADR79	ADR78	ADR77	ADR76	ADR75	ADR74	ADR73	ADR72
Read/Write	R							
After reset	0							
Function	Store upper 8 bits of A/D conversion result							



- Values read from bits 5 through 2 of the ADREG6EL/ADREG7FL are always "1".
- Bit 0 of the ADREG6EL/ADREG7FL is the A/D conversion result storage flag <ADRxRF>. It is set to "1" if an A/D converted value is stored. A read of a lower register (ADREGxL) will set this bit to "0".
- Bit 1 of the ADREG6EL/ADREG7FL is the over Run flag <OVRx>. It is set to "1" if a conversion result is overwritten before both conversion result storage registers (ADREGxH and ADREGxL) are read. A read of a flag will clear this bit to "0".
- When reading conversion result storage registers, first read upper registers and then read lower registers.

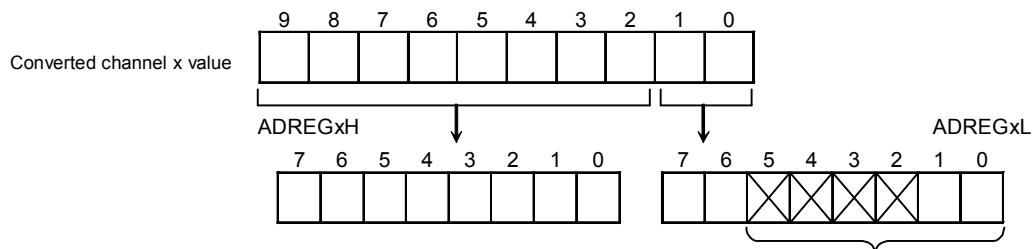
Fig. 13-11 A/D Conversion Result Register

Lower A/D Conversion Result Register SP

	7	6	5	4	3	2	1	0
ADREGSPL	Bit symbol	ADRSP1	ADRSP0				OVRSP	ADRSPRF
	Read/Write	R		R			R	R
	After reset	0		1			0	0
	Function	Store lower 2 bits of A/D conversion result		"1" is read.			Over RUN flag 0: Not generate 1: Generate	A/D conversion result storage flag 1: Presence of conversion result

Upper A/D Conversion Result Register SP

	7	6	5	4	3	2	1	0	
ADREGSPL	Bit symbol	ADRSP9	ADRSP8	ADRSP7	ADRSP6	ADRSP5	ADRSP4	ADRSP3	ADRSP2
	Read/Write	R							
	After reset	0							
	Function	Store upper 8 bits of A/D conversion result							



- Values read from bits 5 through 2 of the ADREGSP are always "1".
- Bit 0 of the ADREGSP is the A/D conversion result storage flag <ADRxRF>. It is set to "1" after an A/D converted value is stored. A read of a lower register (ADREGxL) will set this bit to "0".
- Bit 1 of the ADREGSP is the over RUN flag <OVRx>. It is set to "1" if a conversion result is overwritten before both conversion result storage registers (ADREGxH and ADREGxL) are read. A read of a flag will clear this bit to "0".
- When reading conversion result storage registers, first read upper registers and then read lower registers.

Fig. 13-12 A/D Conversion Result Register

Lower A/D Conversion Result Comparison Register 0

	7	6	5	4	3	2	1	0
ADCMP0L	Bit symbol	ADR021	ADR020					
	Read/Write	R/W		R				
	After reset	0		0				
	Function	Store lower 2 bits of A/D conversion result comparison		"0" is read.				

Upper A/D Conversion Result Comparison Register 0

ADCMP0H		7	6	5	4	3	2	1	0
	Bit symbol	ADR029	ADR028	ADR027	ADR026	ADR025	ADR024	ADR023	ADR022
	Read/Write	R/W							
	After reset	0							
	Function	Store upper 8 bits of A/D conversion result comparison							

Lower A/D Conversion Result Comparison Register 1

	7	6	5	4	3	2	1	0
ADCMP1L	Bit symbol	ADR121	ADR120					
	Read/Write	R/W		R				
	After reset	0		0				
	Function	Store lower 2 bits of A/D conversion result comparison		"0" is read.				

Upper A/D Conversion Result Comparison Register 1

ADCMP1H		7	6	5	4	3	2	1	0
	Bit symbol	ADR129	ADR128	ADR127	ADR126	ADR125	ADR124	ADR123	ADR122
	Read/Write	R/W							
	After reset	0							
	Function	Store upper 8 bits of A/D conversion result comparison							

(Note) To set or change a value in this register, the AD monitor function must be disabled (ADMOD3, 5 <ADOBSVx>="0").

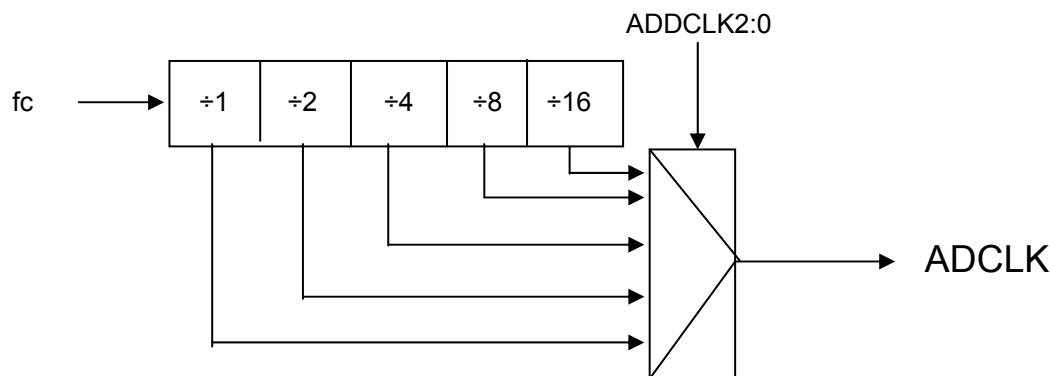
Fig. 13-13 A/D Conversion Result Register

13.3 Conversion Clock

- The conversion time is calculated by the 46 conversion clock.

A/D Conversion Clock Setting Register								
ADCLK	7	6	5	4	3	2	1	0
	Bit symbol	TSH3	tSH2	tSH1	tSH0	ADCLK2	ADCLK1	ADCLK0
	Read/Write	R/W	R/W	R/W	R/W	R	R/W	R/W
	After reset	1	0	0	0	0	0	0
Function	Select the A/D sample hold time 1000: 8 conversion clock 1001: 16 conversion clock 1010: 24 conversion clock 1011: 32 conversion clock 0011: 64 conversion clock 1100: 128 conversion clock 1101: 512 conversion clock The setup other than those above: reserved				"0" is read.	Select the A/D prescaler output 000: fc 001: fc/2 010: fc/4 011: fc/8 100: fc/16 111: reserved		

Fig. 13-14 A/D Conversion Clock Setting Register



Example: If $f_{sys} = f_c = 32 \text{ MHz} / 54 \text{ MHz}$

Prescaler [ADDCLK2:0]	tconv.(conversion time) 40MHz	tconv. (conversion time) 32MHz
1	1.15μs	1.44μs
1/2	2.3μs	2.88μs
1/4	4.6μs	5.75μs

Variable S/H time

Conversion clock	S/H time 40MHz	tconv. (conversion time)
40MHz	Conversion clk*8 (0.2 μs)	1.15μs
	Conversion clk*16 (0.4 μs)	1.35μs
	Conversion clk*24 (0.6 μs)	1.55μs
	Conversion clk*32 (0.8 μs)	1.75μs
	Conversion clk*64 (1.6 μs)	2.55μs
	Conversion clk*128 (3.2 μs)	4.15μs
	Conversion clk*512 (12.8 μs)	13.75μs

(Note) Please do not change the analog to digital conversion clock setting during the analog to digital translation.

Fig. 13-15 A/D Conversion Time

13.4 Description of Operations

13.4.1 Analog Reference Voltage

The "H" level of the analog reference voltage shall be applied to the VREFH pin, and the "L" level shall be applied to the VREFL pin. By writing "0" to the ADMOD1<VREFON> bit, a switched-on state of VREFH-VREFL can be turned into a switched-off state. To start AD conversion, make sure that you first write "1" to the <VREFON> bit, wait for 3 μ s during which time the internal reference voltage should stabilize, and then write "1" to the ADMOD0<ADS> bit.

13.4.2 Selecting the Analog Input Channel

How the analog input channel is selected is different depending on A/D converter operation mode used.

(1) Normal AD conversion mode

- If the analog input channel is used in a fixed state (ADMOD0<SCAN>="0"):

One channel is selected from analog input pins AIN0 through AIN11 by setting ADMOD1<ADCH3 to 0> to an appropriate setting.

- If the analog input channel is used in a scan state (ADMOD0<SCAN>="1"):

One scan mode is selected from 12 scan modes by setting ADMOD1<ADCH3 to 0> and ADSCN to appropriate settings.

(2) Top-priority AD conversion mode

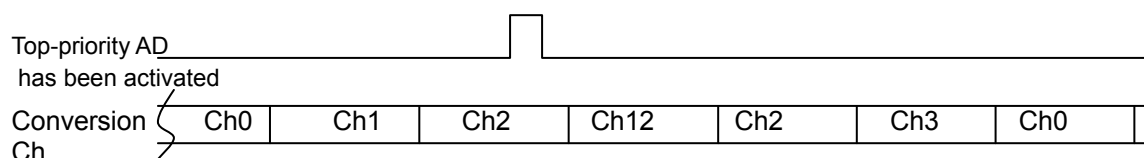
One channel is selected from analog input pins AIN0 through AIN11 by setting ADMOD2<HPADCH3 to 0> to an appropriate setting.

After a reset, ADMOD0<SCAN> is initialized to "0" and ADMOD1<ADCH3:0> is initialized to "0000." This initialization works as a trigger to select a fixed channel input through the AN0 pin. The pins that are not used as analog input channels can be used as ordinary input ports.

If top-priority AD conversion is activated during normal AD conversion, normal AD conversion is discontinued, top-priority AD conversion is executed and completed, and then normal AD conversion is resumed.

Example:

A case in which repeat-scan conversion is ongoing at channels AIN0 through AIN3 with ADMOD0<REPEAT:SCAN> set to "11" and ADMOD1<ADCH3:0> set to 0011, and top-priority AD conversion has been activated at AIN12 with ADMOD2<HPADCH3:0>=1100.



13.4.3 Starting A/D Conversion

Two types of A/D conversion are supported: normal AD conversion and top-priority AD conversion. Normal AD conversion is software activated by setting ADMOD0<ADS> to "1." Top-priority AD conversion is software activated by setting ADMOD2<HPADCE> to "1." 4 operation modes are made available to normal AD conversion. In performing normal AD conversion, one of these operation modes must be selected by setting ADMOD0<2:1> to an appropriate setting. For top-priority AD conversion, only one operation mode can be used: fixed channel single conversion mode. Normal AD conversion can be activated using the HW activation source selected by ADMOD3<ADHS>, and top-priority AD conversion can be activated using the HW activation source selected by ADMOD3<HADHS>. If this bit is "0," normal and top-priority AD conversions are activated in response to the input of a falling edge through the $\overline{\text{ADTRG}}$ pin. If this bit is "1," normal AD conversion is activated in response to TB1RG6 generated by the 16-bit timer 6, and top-priority AD conversion is activated in response to TB2RG5 generated by the 16-bit timer 5. Software activation is still valid even after H/W activation has been authorized.

(Note) When an external trigger is used for the HW activation source of a top priority analog to digital translation, an external trigger cannot usually be set as analog to digital translation HW activation.

(Note) The TMPM330 disables the external trigger used for H/W activation. Therefore "0" cannot be set to <HADHS> and <ADHS>.

When normal A/D conversion starts, the A/D conversion Busy flag (ADMOD0<ADBF>) showing that A/D conversion is under way is set to "1." When top-priority A/D conversion starts, the A/D conversion Busy flag (ADMOD2<ADBFHP>) showing that A/D conversion is under way is set to "1." At that time, the value of the Busy flag for normal A/D conversion before the start of top-priority A/D conversion is retained. The value of the conversion completion flag EOCFN for normal A/D conversion before the start of top-priority A/D conversion can also be retained.

(Note) Normal A/D conversion must not be activated when top-priority A/D conversion is under way. In that case, the top-priority A/D conversion completion flag cannot be set, and the flag for previous normal A/D conversion cannot be cleared.

To reactivate normal A/D conversion while the conversion is under way, a software reset (ADMOD4<ADRST1:0>) must be performed before starting A/D conversion. The HW activation method must not be used to reactivate normal A/D conversion.

If ADMOD2<HPADCE> is set to "1" during normal A/D conversion, ongoing A/D conversion is discontinued and top-priority A/D conversion starts; specifically, A/D conversion (fixed channel single conversion) is executed for a channel designated by ADMOD2<3:0>. After the result of this top-priority A/D conversion is stored in the storage register ADREGSP, normal A/D conversion is resumed.

If HW activation of top-priority A/D conversion is authorized during normal A/D conversion, ongoing A/D conversion is discontinued when requirements for activation using a resource are met, and top-priority A/D conversion (fixed channel single conversion) starts for a channel designated by ADMOD2<3:0>. After the result of this top-priority A/D conversion is stored in the storage register ADREGSP, normal A/D conversion is resumed.

13.4.4 A/D Conversion Modes and A/D Conversion Completion Interrupts

For A/D conversion, the following four operation modes are supported. For normal A/D conversion, an operation mode can be selected by setting ADMOD0<2:1> to an appropriate setting. For top-priority A/D conversion, the fixed channel single conversion mode is automatically selected, irrespective of the ADMOD0<2:1> setting.

- Fixed channel single conversion mode
- Channel scan single conversion mode
- Fixed channel repeat conversion mode
- Channel scan repeat conversion mode

(1) Normal A/D conversion

An operation mode is selected with ADMOD0<REPEAT, SCAN>. As A/D conversion starts, ADMOD0<ADBFN> is set to "1." When specified A/D conversion is completed, the A/D conversion completion interrupt (INTAD) is generated, and ADMOD0<EOCF> showing the completion of A/D conversion is set to "1." If <REPEAT>="0," <ADBFN> returns to "0" concurrently with the setting of EOCF. If <REPEAT> is set to "1," <ADBFN> remains at "1" and A/D conversion continues.

1. Fixed channel single conversion mode

If ADMOD0 <REPEAT, SCAN> is set to "00," A/D conversion is performed in the fixed channel single conversion mode.

In this mode, A/D conversion is performed once for one channel selected. After A/D conversion is completed, ADMOD0<EOCF> is set to "1," ADMOD0<ADBF> is cleared to "0," and the interrupt request INTAD is generated. <EOCF> is cleared to "0" upon read.

2. Channel scan single conversion mode

If ADMOD0 <REPET,SCAN> is set to "01," A/D conversion is performed in the channel scan single conversion mode.

In this mode, A/D conversion is performed once for each scan channel selected. After A/D scan conversion is completed, ADMOD0<EOCF> is set to "1," ADMOD0<ADBF> is cleared to "0," and the interrupt request INTAD is generated. <EOCF> is cleared to "0" upon read.

3. Fixed channel repeat conversion mode

If ADMOD0<REPEAT,SCAN> is set to "10," A/D conversion is performed in fixed channel repeat conversion mode.

In this mode, A/D conversion is performed repeatedly for one channel selected. After A/D conversion is completed, ADMOD <EOCF> is set to "1." ADMOD0 <ADBF> is not cleared to "0." It remains at "1." The timing with which the interrupt request INTAD is generated can be selected by setting ADMOD0 <ITM1:0> to an appropriate setting. <EOCF> is set with the same timing as this interrupt INTAD is generated. I

<EOCF> is cleared to "0" upon read.

With <ITM1:0> set to "00," an interrupt request is generated each time one A/D conversion is completed. In this case, the conversion results are always stored in the storage register ADREG08. After the conversion result is stored, EOCF changes to "1."

With <ITM1:0> set to "01," an interrupt request is generated each time four A/D conversion are completed. In this case, the conversion results are sequentially stored in storage registers ADREG08 through ADREG3B. After the conversion results are stored in ADREG3B, <EOCF> is set to "1," and the storage of subsequent conversion results starts from ADREG08. <EOCF> is cleared to "0" upon read.

With <ITM1:0> set to "10," an interrupt request is generated each time eight A/D conversions are completed. In this case, the conversion results are sequentially stored in storage registers ADREG08 through ADREG7F. After the conversion results are stored in ADREG7F, <EOCF> is set to "1," and the storage of subsequent conversion results starts from ADREG08.

<EOCF> is cleared to "0" upon read.

4. Channel scan repeat conversion mode

If ADMOD0 <REPEAT, SCAN> is set to "11," A/D conversion is performed in the channel scan repeat conversion mode.

In this mode, A/D conversion is performed repeatedly for a scan channel selected. Each time one A/D scan conversion is completed, ADMOD0 <EOCF> is set to "1," and the interrupt request INTAD is generated. ADMOD0 <ADBF> is not cleared to "0." It remains at "1." <EOCF> is cleared to "0" upon read.

To stop the A/D conversion operation in the repeat conversion mode (modes described in 3. and 4. above), write "0" to ADMOD0 <REPEAT>. When ongoing A/D conversion is completed, the repeat conversion mode terminates, and ADMOD0 <ADBF> is set to "0."

Before switching from one mode to standby mode (such standby modes as IDLE, STOP, etc.), check that A/D conversion is not being executed. If A/D conversion is under way, you must stop it or wait until it is completed.

(2) Top-priority A/D conversion

Top-priority A/D conversion is performed only in fixed channel single conversion mode. The ADMOD0<REPEAT, SCAN> setting has no relevance to the top-priority A/D conversion operations or preparations. As activation requirements are met, A/D conversion is performed only once for a channel designated by ADMOD2<HPADCH3:0>. After the A/D conversion is completed, the top-priority A/D conversion completion interrupt is generated, ADMOD2<EOCFHP> is set to "1," and <ADBFHP> returns to "0." The EOCFHP Flag is cleared upon read.

Table 13-1 Relationships among A/D Conversion Modes, Interrupt Generation Timings and Flag Operations

Conversion mode	Interrupt generation timing	EOCF setting timing (see Note)	ADBF (after the interrupt is generated)	ADMOD0		
				ITM1:0	REPEAT	SCAN
Fixed channel single conversion	After conversion is completed	After conversion is completed	0	—	0	0
Fixed channel repeat conversion	Each time one conversion is completed	After one conversion is completed	1	00	1	0
	Each time four conversions are completed	After four conversions are completed	1	01		
	Each time eight conversions are completed	After eight conversions are completed	1	10		
Channel scan single conversion	After scan conversion is completed	After scan conversion is completed	0	—	0	1
Channel scan repeat conversion	Each time one scan conversion is completed	After one scan conversion is completed	1	—	1	1

(Note) EOCF is cleared upon read.

13.4.5 High-priority Conversion Mode

By interrupting ongoing normal A/D conversion, top-priority A/D conversion can be performed. Top-priority A/D conversion can be software activated by setting `ADMOD2<HPADCE>` to "1" or it can be activated using the HW resource by setting `ADMOD3<7:6>` to an appropriate setting. If top-priority A/D conversion has been activated during normal A/D conversion, ongoing normal A/D conversion is interrupted, and single conversion is performed for a channel designated by `ADMOD2<3:0>`. The result of single conversion is stored in `ADREGSP`, and the top-priority A/D conversion interrupt is generated. After top-priority A/D conversion is completed, normal A/D conversion is resumed; the status of normal A/D conversion immediately before being interrupted is maintained. Top-priority A/D conversion activated while top-priority A/D conversion is under way is ignored.

For example, if channel repeat conversion is activated for channels `AN0` through `AN8` and if `<HPADCE>` is set to "1" during `AN3` conversion, `AN3` conversion is suspended, and conversion is performed for a channel designated by `<HPADC3:0>`. After the result of conversion is stored in `ADREGSP`, channel repeat conversion is resumed, starting from `AN3`.

13.4.6 A/D Monitor Function

If `ADCMPx<ADOBSVx>` is set to "1," the A/D monitor function is enabled. If the value of the conversion result storage register specified by `REGS<3:0>` becomes larger or smaller ("larger" or "smaller" to be designated by `ADOBIC`) than the value of a comparison register, the A/D monitor function interrupt is generated. This comparison operation is performed each time a result is stored in a corresponding conversion result storage register, and the interrupt is generated if the conditions are met. Because storage registers assigned to perform the A/D monitor function are usually not read by software, overrun flag `<OVRn>` is always set and the conversion result storage flag `<ADRnRF>` is also set. To use the A/D monitor function, therefore, a flag of a corresponding conversion result storage register must not be used.

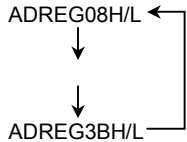
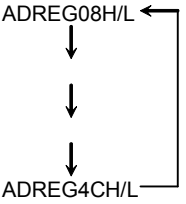
13.4.7 Storing and Reading A/D Conversion Results

A/D conversion results are stored in upper and lower A/D conversion result registers for normal A/D conversion (`ADREG08H/L` through `ADRG7FH/L`).

In fixed channel repeat conversion mode, A/D conversion results are sequentially stored in `ADREG08H/L` through `ADREG7FH/L`. If `<ITM1:0>` is so set as to generate the interrupt each time one A/D conversion is completed, conversion results are stored only in `ADREG08H/L`. If `<ITM1:0>` is so set as to generate the interrupt each time four A/D conversions are completed, conversion results are sequentially stored in `ADREG08H/L` through `ADREG3BH/L`.

Table 13-2 shows analog input channels and related A/D conversion result registers.

Table 13-2 Analog Input Channels and Related A/D Conversion Result Registers

Analog input channel (port A)	A/D conversion result register			
	Conversion modes other than shown to the right	Fixed channel repeat conversion mode (every one conversion)	Fixed channel repeat conversion mode (every four conversions)	Fixed channel repeat conversion mode (every eight conversions)
AN0	ADREG08H/L	ADREG08H/L fixed		
AN1	ADREG19H/L			
AN2	ADREG2AH/L			
AN3	ADREG3BH/L			
AN4	ADREG4CH/L			
AN5	ADREG5DH/L			
AN6	ADREG6EH/L			
AN7	ADREG7FH/L			
AN8	ADREG08H/L			
AN9	ADREG19H/L			
AN10	ADREG2AH/L			
AN11	ADREG3BH/L			
AN12	ADREG4CH/L			

13.4.8 Data Polling

To process A/D conversion results without using interrupts, ADMOD0<EOCF> must be polled. If this flag is set, conversion results are stored in a specified A/D conversion result register. After confirming that this flag is set, read that conversion result storage register. In reading the register, make sure that you first read upper bits and then lower bits to detect an overrun. If OVRn is "0" and ADRnRF is "1" in lower bits, a correct conversion result has been obtained.

14. Watchdog Timer (Runaway Detection Timer)

The TMPM330 has a built-in watchdog timer for detecting runaways.

The watchdog timer (WDT) is for detecting malfunctions (runaways) of the CPU caused by noises or other disturbances and remedying them to return the CPU to normal operation. If the timer detects a runaway, it generates a non-maskable interrupt to notify the CPU and outputs “0” from the output pin of the watchdog timer to notify peripherals.

By connecting the output of the watchdog timer to a reset pin (inside the chip), it is possible to force the watchdog timer to reset itself.

14.1 Configuration

Fig. 14-1 shows the block diagram of the watchdog timer

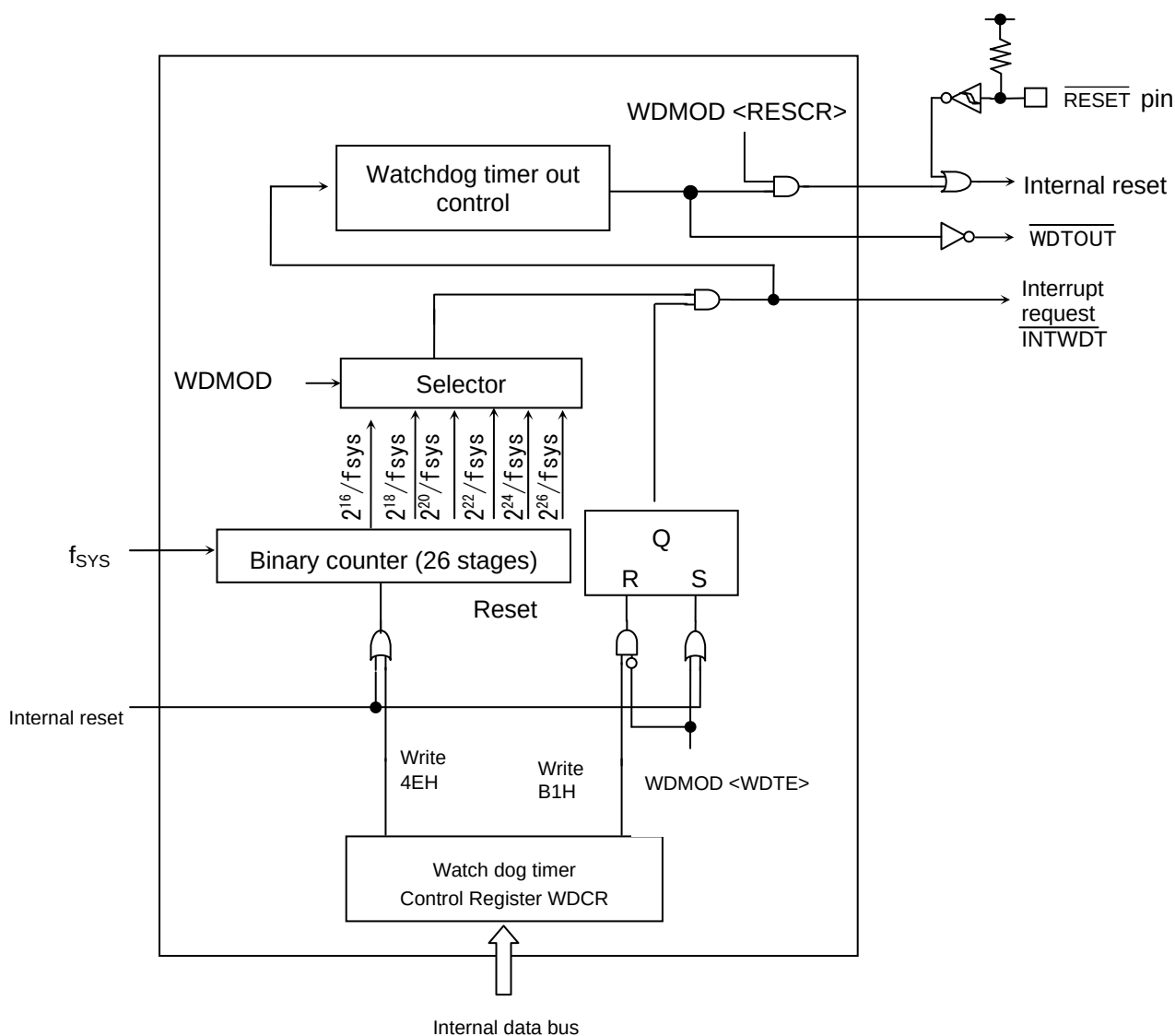


Fig. 14-1 Block Diagram of the Watchdog Timer

14.2 Watchdog Timer Interrupt

The watchdog timer consists of the binary counters that are arranged in 26 stages and work using the $f_{SYS/2}$ system clock as an input clock. The outputs produced by these binary counters are 2^{16} , 2^{18} , 2^{20} , 2^{22} , 2^{24} are 2^{26} . By selecting one of these outputs with WDMOD <WDTP2:0>, a watchdog timer interrupt can be generated and the $\overline{\text{WDTOUT}}$ is output when an overflow occurs, as shown in Fig. 14-2.

Because the watchdog timer interrupt is a non-maskable interrupt factor, NMIFLG <WDT> at the INTC performs a task of identifying it.

The output pin of the watchdog timer can reset the peripherals by outputting "0" caused by an overflow. The output is set to "1" if the watchdog timer is cleared (if the clear code 4EH is written to the WDCR register). The $\overline{\text{WDTOUT}}$ pin outputs "0" at normal mode unless the clear code is written to WDCR register.

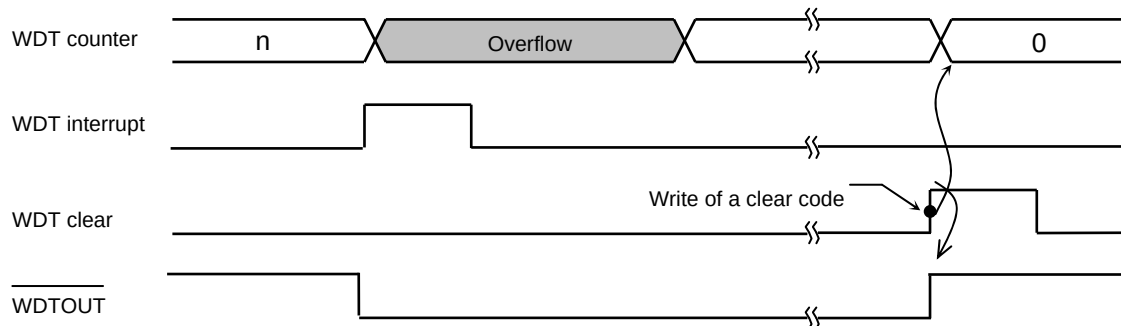


Fig. 14-2 Normal Mode

When an overflow occurs, resetting the chip itself is an option to choose. If the chip is reset, a reset is affected for a 32-state time, as shown in Fig. 14-3. If this reset is affected, the clock f_{SYS} that the clock gear generates by dividing the clock f_C of the high-speed oscillator by 1 is used as an input clock f_{SYS} .

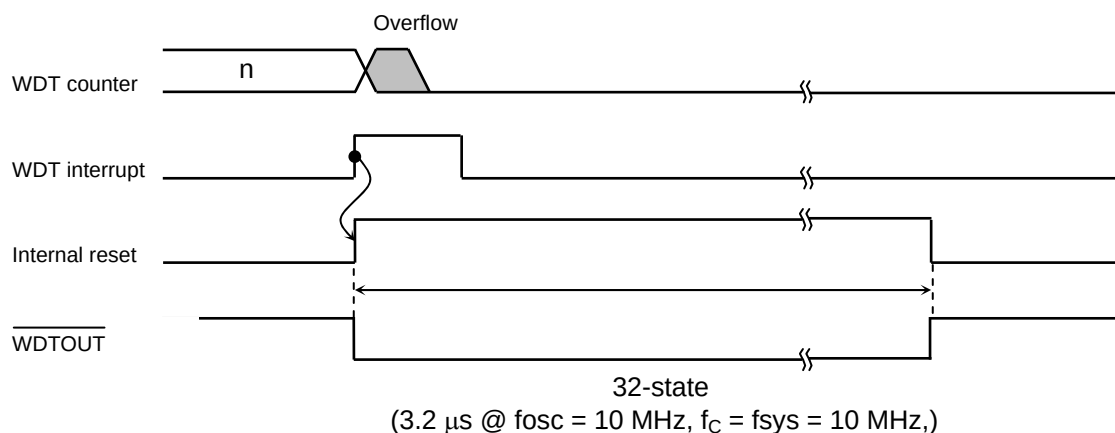


Fig. 14-3 Reset Mode

14.3 Control Registers

The watchdog timer (WDT) is controlled by two control registers WDMOD and WDCR.

14.3.1 Watchdog Timer Mode Register (WDMOD)

1. Specifying the detection time of the watchdog timer <WDTP1: 0>

This is a 2-bit register for specifying the watchdog timer interrupt time for runaway detection. When a reset is effected, this register is initialized to WDMOD <WDTP1, 0> = "00." Fig. 14-4 shows the detection time of the watchdog timer.

2. Enabling/disabling the watchdog timer <WDTE>

When resetting, WDMOD <WDTE> is initialized to "1" and the watchdog timer is enabled.

To disable the watchdog timer, this bit must be set to "0" and, at the same time, the disable code (B1H) must be written to the WDCR register. This dual setting is intended to minimize the probability that the watchdog timer may inadvertently be disabled if a runaway occurs.

To change the status of the watchdog timer from "disable" to "enable," set the <WDTE> bit to "1".

3. Watchdog timer out reset connection <RESCR>

This is a register for specifying whether or not to reset the watchdog timer itself after a runaway is detected. As a reset initializes this setting to WDMOD <RESCR>="0," a reset initiated the output of the watchdog timer is not performed.

14.3.2 Watchdog Timer Control Register (WDCR)

This is a register for disabling the watchdog timer function and controlling the clearing function of the binary counter.

- Disabling control

By writing the disable code (B1H) to this WDCR register after setting WDMOD <WDTE> to "0," the watchdog timer can be disabled.

WDMOD	← 0 - - - - -	Clears WDTE to "0."
WDCR	← 1 0 1 1 0 0 0 1	Writes the disable code (B1H).

- Enabling control

Set WDMOD <WDTE> to "1".

- Watchdog timer clearing control

Writing the clear code (4EH) to the WDCR register clears the binary counter and allows it to resume counting.

WDCR	← 0 1 0 0 1 1 1 0	Writes the clear code (4EH)

(Note) Writing the disable code (BIH) clears the binary counter.

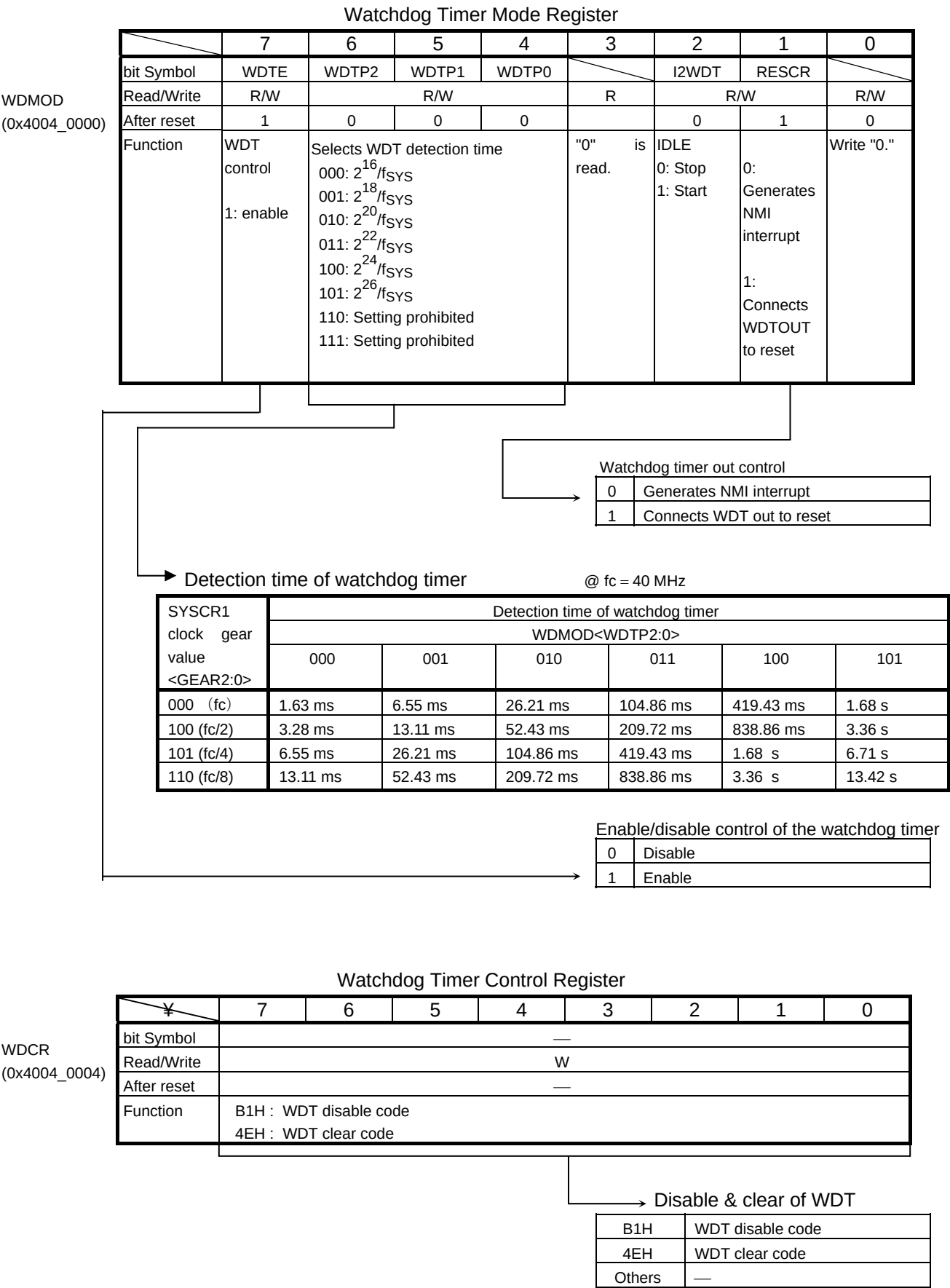


Fig. 14-4 Watchdog Timer Registers

14.4 Control Register

The watchdog timer generates the INTWDT interrupt after a lapse of the detection time specified by the WDMOD <WDTP1, 0> register and outputs a signal at low level from the output pin of the watchdog timer (WDTOUT). Before generating the INTWD interrupt, the binary counter for the watchdog timer must be cleared to "0" using software (instruction). If the CPU malfunctions (runaways) due to noise or other disturbances and cannot execute the instruction to clear the binary counter, the binary counter overflows and the INTWD interrupt is generated. The CPU is able to recognize the occurrence of a malfunction (runaway) by identifying the INTWD interrupt and to restore the faulty condition to normal by using a malfunction (runaway) countermeasure program. Additionally, it is possible to resolve the problem of a malfunction (runaway) of the CPU by connecting the watchdog timer out pin to reset pins of peripheral devices.

The watchdog timer begins operation immediately after a reset is cleared.

In STOP mode, the watchdog timer is reset and in an idle state. When the bus is open ($\overline{\text{BUSAK}} = \text{"L"}), it continues counting. In IDLE mode, its operation depends on the WDMOD <I2WDT> setting. Before putting it in IDLE mode, WDMOD <I2WDT> must be set to an appropriate setting, as required.$

Example:

1. To clear the binary counter

		7	6	5	4	3	2	1	0	
WDCR	←	0	1	0	0	1	1	1	0	Writes the clear code (4EH)

2. To set the detection time of the watchdog timer to $2^{18}/f_{\text{SYS}}$.

		7	6	5	4	3	2	1	0
WDMOD	←	1	0	1	—	—	—	—	—

3. To disable the watchdog timer.

		7	6	5	4	3	2	1	0	
WDMOD	←	0	—	—	—	—	—	—	—	Clears WDTE to "0"
WDCR	←	1	0	1	1	0	0	0	1	Writes the disable code (B1H)

(Note 1) If the watchdog timer is operated when the high-frequency oscillator is idle, the system reset operation initiated by the watchdog timer becomes erratic due to the unstable oscillation of the high-frequency oscillator. Therefore, do not operate the watchdog timer when the high-frequency oscillator is idle.

(Note 2) The counter of the watchdog timer stops at the debug mode.

15 Real Time Clock (RTC)

15.1 Functions

- 1) Clock (hour, minute and second)
- 2) Calendar (month, week, date and leap year)
- 3) Selectable 12 (am/ pm) and 24 hour display
- 4) Time adjustment + or - 30 seconds (by software)
- 5) Alarm (alarm output)
- 6) Alarm interrupt

15.2 Block Diagram

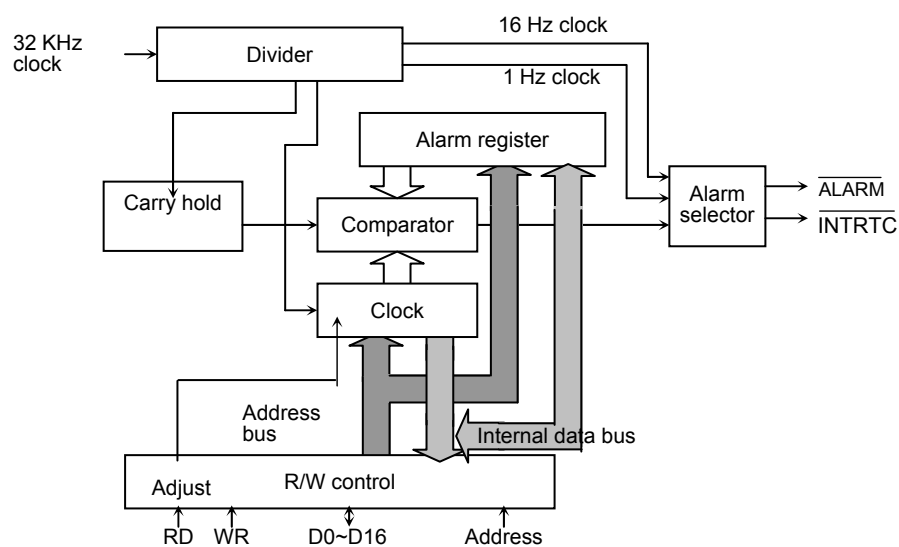


Fig. 15-1 Block Diagram

(Note 1) Western calendar year column:

This product uses only the final two digits of the year. The year following 99 is 00 years. Please take into account the first two digits when handling years in the western calendar.

(Note 2) Leap year:

A leap year is divisible by 4 excluding a year divisible by 100; the year divisible by 100 is not considered to be a leap year. Any year divisible by 400 is a leap year. This product is considered the year divisible by 4 to be a leap year and does not take into account the above exceptions. It needs adjustments for the exceptions.

15.3 Control Register

Table 15-1 PAGE0 (clock function) register

Symbol	Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Function	Read/Write
SECR	0x4004_0100H		40 sec	20 sec	10 sec	8 sec	4 sec	2 sec	1 sec	Second column	R/W
MINR	0x4004_0101H		40 min	20 min	10 min	8 min	4 min	2 min	1 min	Minute column	R/W
HOURL	0x4004_0102H			20 hours /PM/AM	10 hours	8 hours	4 hours	2 hours	1 hours	Hour column	R/W
DAYR	0x4004_0104H						W2	W1	W0	Day of the week column	R/W
DATER	0x4004_0105H			Day 20	Day 10	Day 8	Day 4	Day 2	Day 1	Day column	R/W
MONTHR	0x4004_0106H				Oct.	Aug.	Apr.	Feb.	Jan.	Month column	R/W
YEARR	0x4004_0107H	Year 80	Year 40	Year 20	Year 10	Year 8	Year 4	Year 2	Year 1	Year column (lower two columns)	R/W
PAGER	0x4004_0108H	Interrupt enable			Adjustment function	Clock enable	Alarm enable		PAGE setting	PAGE register	W, R/W
RESTR	0x4004_010CH	1Hz enable	16Hz enable	Clock reset	Alarm reset	Always write "0".				Reset register	W only

(Note) Reading SECR, MINR, HOURL, DAYR, MONTHR, YEARR of PAGE0 captures the current state.

Table 15-2 PAGE1 (alarm function) registers

Symbol	Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Function	Read/Write
SECR	0x4004_0100H										
MINR	0x4004_0101H		40 min	20 min	10 min	8 min	4 min	2 min	1 min	Minute column	R/W
HOURL	0x4004_0102H			20 hours /PM/AM	10 hours	8 hours	4 hours	2 hours	1 hours	Hour column	R/W
DAYR	0x4004_0104H						W2	W1	W0	Day of the week column	R/W
DATER	0x4004_0105H			Day 20	Day 10	Day 8	Day 4	Day 2	Day 1	Day column	R/W
MONTH R	0x4004_0106H								24/12	24-hour clock mode	R/W
YEARR	0x4004_0107H							Leap-year setting		Leap-year mode	R/W
PAGER	0x4004_0108H	Interrupt enable			Adjustme nt function	Clock enable	Alarm enable		PAGE setting	PAGE register	W,R/W
RESTR	0x4004_010CH	1Hz enable	16Hz enable	Clock reset	Alarm reset	Always write "0".				Reset register	W only

(Note 1) Reading SECR, MINR, HOURL, DAYR, MONTHR, YEARR of PAGE0 captures the current state.

(Note 2) SECR, MINR, HOURL, DAYR, MONTHR, YEARR of PAGE0 and YEARR of PAGE1 (for leap year) must be read twice and compare the data captured.

15.4 Control register

The RTC is not initialized by system reset. All registers must be initialized at the beginning of the program.

(1) Second column register (for PAGE0 only)

	7	6	5	4	3	2	1	0
Bit symbol	—	SE6	SE5	SE4	SE3	SE2	SE1	SE0
Read/Write	R	R/W						
After reset	0	Undefined						
Function	"0" is read.	40 sec. column	20 sec. column	10 sec. column	8 sec. column	4 sec. column	2 sec. column	1 sec. column

SECR

0	0	0	0	0	0	0	0 sec
0	0	0	0	0	0	1	1 sec
0	0	0	0	0	1	0	2 sec
0	0	0	0	0	1	1	3 sec
0	0	0	0	1	0	0	4 sec
0	0	0	0	1	0	1	5 sec
0	0	0	0	1	1	0	6 sec
0	0	0	0	1	1	1	7 sec
0	0	0	1	0	0	0	8 sec
0	0	0	1	0	0	1	9 sec
0	0	1	0	0	0	0	10 sec
:							
0	0	1	1	0	0	1	19 sec
0	1	0	0	0	0	0	20 sec
:							
0	1	0	1	0	0	1	29 sec
0	1	1	0	0	0	0	30 sec
:							
0	1	1	1	0	0	1	39 sec
1	0	0	0	0	0	0	40 sec
:							
1	0	0	1	0	0	1	49 sec
1	0	1	0	0	0	0	50 sec
:							
1	0	1	1	0	0	1	59 sec

Note) Do not set data other than as shown above.

(2) Minute column register (for PAGE0/1)

MINR		7	6	5	4	3	2	1	0
	Bit symbol	—	M16	M15	M14	M13	M12	M11	M10
	Read/Write	R	R/W						
	After reset	0	Undefined						
	Function	“0” is read	40 min. column	20 min. column	10 min. column	8 min. column	4 min. column	2 min. column	1 min. column

0	0	0	0	0	0	0	0 min
0	0	0	0	0	0	1	1 min
0	0	0	0	0	1	0	2 min
0	0	0	0	0	1	1	3 min
0	0	0	0	1	0	0	4 min
0	0	0	0	1	0	1	5 min
0	0	0	0	1	1	0	6 min
0	0	0	0	1	1	1	7 min
0	0	0	1	0	0	0	8 min
0	0	0	1	0	0	1	9 min
0	0	1	0	0	0	0	10 min
:							
0	0	1	1	0	0	1	19 min
0	1	0	0	0	0	0	20 min
:							
0	1	0	1	0	0	1	29 min
0	1	1	0	0	0	0	30 min
:							
0	1	1	1	0	0	1	39 min
1	0	0	0	0	0	0	40 min
:							
1	0	0	1	0	0	1	49 min
1	0	1	0	0	0	0	50 min
:							
1	0	1	1	0	0	1	59 min

Note) Do not set data other than as shown above.

(3) Hour column register (for PAGE0/1)

1. 24-hour clock mode (MONTHR<MO0>="1")

	7	6	5	4	3	2	1	0
Bit symbol	—		HO5	HO4	HO3	HO2	HO1	HO0
Read/Write	R		R/W					
After reset	0		Undefined					
Function	"0" is read.		20 hour column	10 hour column	8 hour column	4 hour column	2 hour column	1 hour column

0	0	0	0	0	0	0 o' clock
0	0	0	0	0	1	1 o' clock
0	0	0	0	1	0	2 o' clock

:

0	0	1	0	0	0	8 o' clock
0	0	1	0	0	1	9 o' clock
0	1	0	0	0	0	10 o' clock

:

0	1	1	0	0	1	19 o' clock
1	0	0	0	0	0	20 o' clock

:

1	0	0	0	1	1	23 o' clock

Note) Do not set data other than as shown above.

2. 12-hour clock mode (MONTHR<MO0>="0")

	7	6	5	4	3	2	1	0
Bit symbol	—		HO5	HO4	HO3	HO2	HO1	HO0
Read/Write	R		R/W					
After reset	0		Undefined					
Function	"0" is read.		PM/AM	10 hour column	8 hour column	4 hour column	2 hour column	1 hour column

0	0	0	0	0	0	0 o' clock (AM)
0	0	0	0	0	1	1 o' clock
0	0	0	0	1	0	2 o' clock

:

0	0	1	0	0	1	9 o' clock
0	1	0	0	0	0	10 o' clock
0	1	0	0	0	1	11 o' clock
1	0	0	0	0	0	0 o' clock (PM)
1	0	0	0	0	1	1 o' clock

Note) Do not set data other than as shown above.

(4) Day of the week column register (for PAGE0/1)

	7	6	5	4	3	2	1	0
DAYR	—					WE2	WE1	WE0
Bit symbol	R					R/W		
Read/Write	0					Undefined		
After reset	“0” is read.					W2	W1	W0
Function								

0	0	0	Sunday
0	0	1	Monday
0	1	0	Tuesday
0	1	1	Wednesday
1	0	0	Thursday
1	0	1	Friday
1	1	0	Saturday

Note) Do not set data other than as shown above.

(5) Day column register (PAGE0/1)

	7	6	5	4	3	2	1	0
DATER	—		DA5	DA4	DA3	DA2	DA1	DA0
Bit symbol	R		R/W					
Read/Write	0		Undefined					
After reset	“0” is read.		Day 20	Day 10	Day 8	Day 4	Day 2	Day 1
Function								

0	0	0	0	0	0	0
0	0	0	0	0	1	1st day
0	0	0	0	1	0	2nd day
0	0	0	0	1	1	3rd day
0	0	0	1	0	0	4th day

:

0	0	1	0	0	1	9th day
0	1	0	0	0	0	10th day
0	1	0	0	0	1	11th day

:

0	1	1	0	0	1	19th day
1	0	0	0	0	0	20th day

:

1	0	1	0	0	1	29th day
1	1	0	0	0	0	30th day
1	1	0	0	0	1	31st day

Note 1) Do not set data other than as shown above.

Note 2) Do not set for non-existent days (e.g.: 30th Feb)

(6) Month column register (for PAGE0 only)

	7	6	5	4	3	2	1	0
MONTHR	—			MO4	MO4	MO2	MO1	MO0
Bit symbol	—			R/W				
Read/Write	R			R/W				
After reset	0			Undefined				
Function	"0" is read.			10 months	8 months	4 months	2 months	1 month

0	0	0	0	1	January
0	0	0	1	0	February
0	0	0	1	1	March
0	0	1	0	0	April
0	0	1	0	1	May
0	0	1	1	0	June
0	0	1	1	1	July
0	1	0	0	0	August
0	1	0	0	1	September
1	0	0	0	0	October
1	0	0	0	1	November
1	0	0	1	0	December

Note) Do not set data other than as shown above.

(7) Selection of 24-hour clock or 12-hour clock (for PAGE1 only)

	7	6	5	4	3	2	1	0
MONTHR	—							MO0
Bit symbol	—							MO0
Read/Write	R							R/W
After reset	0							Undefined
Function	"0" is read.							1: 24-hour 0: 12-hour

(Note) Do not change the MONTHR<MO0> bit while the RTC is in operation.

(8) Year column register (for PAGE0 only)

	7	6	5	4	3	2	1	0
Bit symbol	YE7	YE6	YE5	YE4	YE3	YE2	YE1	YE0
Read/Write	R/W							
After reset	Undefined							
Function	80 years	40 years	20 years	10 years	8 years	4 years	2 years	1 years

0	0	0	0	0	0	0	0	00 years
0	0	0	0	0	0	0	1	01 years
0	0	0	0	0	0	1	0	02 years
0	0	0	0	0	0	1	1	03 years
0	0	0	0	0	1	0	0	04 years
0	0	0	0	0	1	0	1	05 years
:								
1	0	0	1	1	0	0	1	99 years

Note) Do not set data other than as shown above.

(9) Leap year register (for PAGE1 only)

	7	6	5	4	3	2	1	0
Bit symbol	—						LEAP1	LEAP0
Read/Write	R						R/W	
After reset	0						Undefined	
Function	"0" is read.						00: leap year	
							01: one year after leap year	
							10: two years after leap year	
							11: three years after leap year	

0	0	Current year is a leap-year.
0	1	Current year is the year following a leap-year.
1	0	Current year is two years after a leap year.
1	1	Current year is three years after a leap year

(10) PAGE register (for PAGE0/1)

	7	6	5	4	3	2	1	0
PAGER	Bit symbol	INTENA	—	ADJUST	ENATMR	ENAALM	—	PAGE
	Read/Write	R/W	R	W	R/W	R/W	R	R/W
	After reset	0	0	Undefined	Undefined	Undefined	0	Undefined
A read-modify-write operation cannot be performed.	Function	INTRTC 0: Disabled 1: Enabled	"0" is read.	0: Don't care 1: Adjust	Clock 0: Disabled 1: Enabled	ALARM 0: Disabled 1: Enabled	"0" is read.	PAGE selection

(Note) Keep the setting order of <ENATMR>, <ENAAML> and <INTENA> as shown in the example below. Ensure an interval of time between Clock/Alarm and interrupt.

Example: Clock setting/Alarm setting

	7	6	5	4	3	2	1	0	
PAGER	←	0	0	0	1	1	0	0	Enables Clock and alarm
PAGER	←	1	0	0	0	1	1	0	Enables interrupt

PAGE	0	Selects Page0
	1	Selects Page1

ADJUST	0	Don't care
	1	Adjusts sec. counter by setting this bit to "1". If it is set when the time elapsed is within 0 and 29, the sec. counter is cleared to "0". If the time elapsed is within 30 and 59, the min. counter is carried and sec. counter is cleared to "0". The ADJUST signal is output during 1 cycle of f_{sys} . After being adjusted once, the ADJUST state is released automatically (PAGE0 only).

(11) Reset register (for PAGE0/1)

		7	6	5	4	3	2	1	0
RESTR	Bit symbol	DIS1HZ	DIS16HZ	RSTTMR	RSTALM	–			
	Read/Write	W							
	After reset	Undefined							
A read-modify-write operation cannot be performed.	Function	1 Hz 0: Enabled 1: Disabled	16 Hz 0: Enabled 1: Disabled	1: Clock reset	1: Alarm reset	Always write “0”.			

RSTALM	0	Unused
	1	Reset alarm register.

RSTTMR	0	Unused
	1	Reset clock register.

<DIS1HZ>	<DIS16HZ>	PAGER<ENAALM>	Interrupt source signal
1	1	1	Alarm
0	1	0	1Hz
1	0	0	16Hz
Others			Outputs "0".

15.5 Operational description

(1) Reading clock data

1. Using 1Hz interrupt

The count-up of the internal data synchronizes with 1Hz interrupt. Data can be read correctly if reading data after 1Hz interrupt occurred.

2. Using pair reading

There is a possibility that the clock data may be read incorrectly if the internal counter operates carry during reading. To ensure correct data reading, read the clock data twice as shown below. A pair of data read successively needs to match.

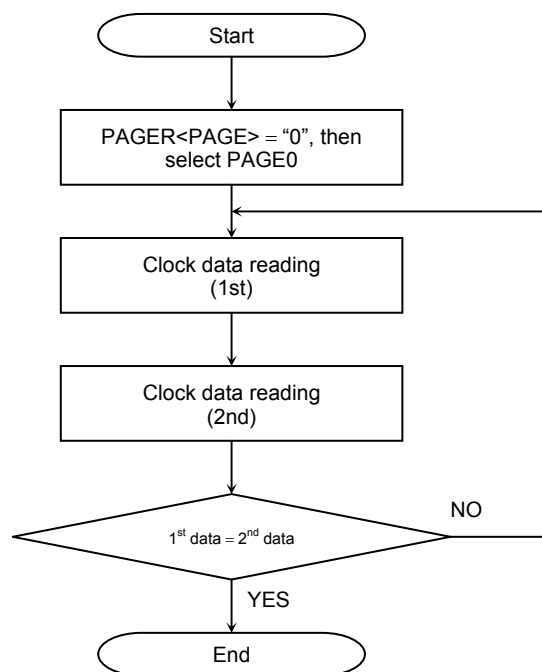


Fig. 15-2 Flowchart of the clock data reading

(2) Writing clock data

A carry during writing ruins correct data writing. The following procedure ensures the correct data writing.

1. Using 1Hz interrupt

The count-up of the internal data synchronizes with 1Hz interrupt. Data can be written correctly if writing data after 1Hz interrupt occurred.

2. Resetting counter

The RTC incorporates 15-stage counter that generates a 1Hz clock from 32,768 KHz. After resetting the counter, the data is written.

If clearing the counter, an interrupt is output only first writing at half of the setting time. To ensure the correct clock counting, enable the 1Hz-interrupt after clearing the counter. And then set the time after the first interrupt (occurs at 0.5Hz) occurs.

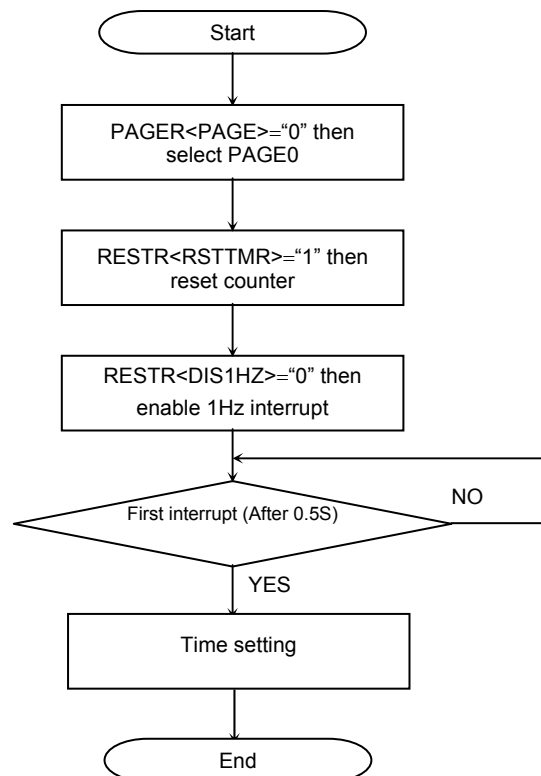


Fig. 15-3 Flowchart of the clock data writing

3. Disabling the clock

Writing "0" to `PAGER<ENATMR>` disables clock operation including a carry. In the meantime, the 1 second carry hold circuit executes time adjustment instead.

While the clock is disabled, the carry hold circuit holds a 1 second carry signal from a divider. When the clock becomes enabled, the carry signal is output to the clock, the time is revised and operation continues. The circuit holds only a 1 second carry signal. Any adjustment that exceeds one second cannot be executed.

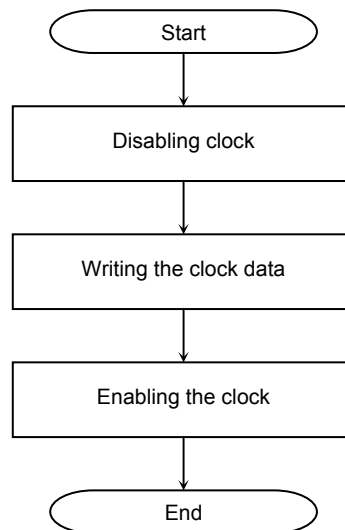


Fig. 15-4 Flowchart of the disabling clock

15.6 Alarm function

By writing "1" to `PAGER<PAGE>`, the alarm function of the `PAGE1` registers is enabled. One of the following three signals is output to the `ALARM` pin.

- (1) "0" (when the alarm register corresponds with the clock)
- (2) 1Hz clock
- (3) 16Hz clock

The `INTRTC` outputs a 1-shot pulse by detecting the falling edge.

The `RTC` is not initialized by reset. Clear the interrupt request flag in the interrupt controller when the clock or alarm function is used.

- (1) "0" (when the alarm register corresponds with the clock)

Provided `PAGER <ENAALM>= "1"`, "0" is output to the `ALARM` pin when the values of the `PAGE0` clock and the `PAGE1` alarm register correspond. In combination with the setting and `PAGER<INTENA>="1"`, the `INTRTC` interrupt is generated and the alarm is triggered.

The alarm settings

To initialize the alarm, write "1" to `RESTR<RSTALM>`. It makes all alarm settings "don't care". In this case, the alarm always corresponds with the value of the clock. The `INTRTC` interrupt request is generated if `PAGER <INENA> <ENAALM>="1"`.

Setting alarm for min., hour, date and day is done by writing data to the relevant `PAGE1` register. Each writing releases the "don't care" state respectively.

When all setting contents correspond with the setting of `PAGER<INTENA> <ENAALM>=1`, the `RTC` generates an `INTRTC` interrupt. However, contents which have not been set up ("don't care" state) are always considered to be corresponding.

Contents which have already been set up cannot be returned independently but all together to the "don't care" state by initializing the alarm.

The following is an example program for outputting an alarm from the `ALARM` pin at noon (PM12:00) every day.

	7	6	5	4	3	2	1	0		
PAGER	←	0	0	0	0	1	0	0	1	Disables alarm, sets PAGE1
RESTR	←	1	1	0	1	0	0	0	0	Initializes alarm
DAYR	←	0	0	0	0	0	0	0	1	W0
DATAR	←	0	0	0	0	0	0	0	1	1 day
HOURR	←	0	0	0	1	0	0	1	0	Sets 12 o'clock
MINR	←	0	0	0	0	0	0	0	0	Sets 00 min.
										Set up time 31 μs (Note)
PAGER	←	0	0	0	0	1	1	0	0	Enables alarm
PAGER	←	1	0	0	0	1	1	0	0	Enables interrupt

When the CPU is operating at high frequency oscillation, it may take a maximum of one clock at 32 kHz (about 30us) for the time register setting to become valid. In the above example, it is necessary to set 31μs of set up time between setting the time register and enabling the alarm register.

(Note) This set up time is unnecessary when you use only internal interruption.

(2) 1Hz clock

The RTC outputs a clock of 1Hz to the ALARM pin by setting `PAGER<ENAALM>= "0"`, `RESTR<DIS1HZ>= "0"` and `<DIS16HZ>= "1"`. It also generates an INTRC interrupt on the falling edge of the clock.

(3) 16Hz clock

The RTC outputs a clock of 16Hz to the ALARM pin by setting `PAGER<ENAALM>= "0"`, `RESTR<DIS1HZ>= "1"` and `<DIS16HZ>= "0"`. It also generates an INTRC interrupt on the falling edge of the clock.

16. Table of Special Function Registers

- [1] Port registers
- [2] 16-bit timer (TMRB)
- [3] Serial bus interface (SBI)
- [4] Serial interface (UART/SIO)
- [5] 10-bit A/D converter (A/DC)
- [6] Watchdog timer (WDT)
- [7] Real time clock (RTC)
- [8] Clock generator (CG)
- [9] CEC
- [10] Remote control signal preprocessor
- [11] Flash

(Note 1) The <R0> areas are read as “0”. Writing the area is ignored.

(Note 2) Access to the <reserved> areas is prohibited.

16.1 Addresses

[1] Port [1/4]

<PORT A>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4000_0000	PA	0x4000_0010		0x4000_0020		0x4000_0030	PAPDN
0x4000_0001	<R0>	0x4000_0011		0x4000_0021		0x4000_0031	<R0>
0x4000_0002	<R0>	0x4000_0012		0x4000_0022		0x4000_0032	<R0>
0x4000_0003	<R0>	0x4000_0013		0x4000_0023		0x4000_0033	<R0>
0x4000_0004	PACR	0x4000_0014		0x4000_0024		0x4000_0034	
0x4000_0005	<R0>	0x4000_0015		0x4000_0025		0x4000_0035	
0x4000_0006	<R0>	0x4000_0016		0x4000_0026		0x4000_0036	
0x4000_0007	<R0>	0x4000_0017		0x4000_0027		0x4000_0037	
0x4000_0008	PAFR1	0x4000_0018		0x4000_0028		0x4000_0038	PAIE
0x4000_0009	<R0>	0x4000_0019		0x4000_0029		0x4000_0039	<R0>
0x4000_000A	<R0>	0x4000_001A		0x4000_002A		0x4000_003A	<R0>
0x4000_000B	<R0>	0x4000_001B		0x4000_002B		0x4000_003B	<R0>
0x4000_000C		0x4000_001C		0x4000_002C	PAPUP	0x4000_003C	
0x4000_000D		0x4000_001D		0x4000_002D	<R0>	0x4000_003D	
0x4000_000E		0x4000_001E		0x4000_002E	<R0>	0x4000_003E	
0x4000_000F		0x4000_001F		0x4000_002F	<R0>	0x4000_003F	

<PORT B>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4000_0040	PB	0x4000_0050		0x4000_0060		0x4000_0070	
0x4000_0041	<R0>	0x4000_0051		0x4000_0061		0x4000_0071	
0x4000_0042	<R0>	0x4000_0052		0x4000_0062		0x4000_0072	
0x4000_0043	<R0>	0x4000_0053		0x4000_0063		0x4000_0073	
0x4000_0044	PBCR	0x4000_0054		0x4000_0064		0x4000_0074	
0x4000_0045	<R0>	0x4000_0055		0x4000_0065		0x4000_0075	
0x4000_0046	<R0>	0x4000_0056		0x4000_0066		0x4000_0076	
0x4000_0047	<R0>	0x4000_0057		0x4000_0067		0x4000_0077	
0x4000_0048	PBFR1	0x4000_0058		0x4000_0068		0x4000_0078	PBIE
0x4000_0049	<R0>	0x4000_0059		0x4000_0069		0x4000_0079	<R0>
0x4000_004A	<R0>	0x4000_005A		0x4000_006A		0x4000_007A	<R0>
0x4000_004B	<R0>	0x4000_005B		0x4000_006B		0x4000_007B	<R0>
0x4000_004C		0x4000_005C		0x4000_006C	PBPUP	0x4000_007C	
0x4000_004D		0x4000_005D		0x4000_006D	<R0>	0x4000_007D	
0x4000_004E		0x4000_005E		0x4000_006E	<R0>	0x4000_007E	
0x4000_004F		0x4000_005F		0x4000_006F	<R0>	0x4000_007F	

<PORT C>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4000_0080	PC	0x4000_0090		0x4000_00A0		0x4000_00B0	
0x4000_0081	<R0>	0x4000_0091		0x4000_00A1		0x4000_00B1	
0x4000_0082	<R0>	0x4000_0092		0x4000_00A2		0x4000_00B2	
0x4000_0083	<R0>	0x4000_0093		0x4000_00A3		0x4000_00B3	
0x4000_0084		0x4000_0094		0x4000_00A4		0x4000_00B4	
0x4000_0085		0x4000_0095		0x4000_00A5		0x4000_00B5	
0x4000_0086		0x4000_0096		0x4000_00A6		0x4000_00B6	
0x4000_0087		0x4000_0097		0x4000_00A7		0x4000_00B7	
0x4000_0088		0x4000_0098		0x4000_00A8		0x4000_00B8	PCIE
0x4000_0089		0x4000_0099		0x4000_00A9		0x4000_00B9	<R0>
0x4000_008A		0x4000_009A		0x4000_00AA		0x4000_00BA	<R0>
0x4000_008B		0x4000_009B		0x4000_00AB		0x4000_00BB	<R0>
0x4000_008C		0x4000_009C		0x4000_00AC	PCPUP	0x4000_00BC	
0x4000_008D		0x4000_009D		0x4000_00AD	<R0>	0x4000_00BD	
0x4000_008E		0x4000_009E		0x4000_00AE	<R0>	0x4000_00BE	
0x4000_008F		0x4000_009F		0x4000_00AF	<R0>	0x4000_00BF	

[1] Port [2/4]

<PORT D>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4000_00C0	PD	0x4000_00D0		0x4000_00E0		0x4000_00F0	
0x4000_00C1	<R0>	0x4000_00D1		0x4000_00E1		0x4000_00F1	
0x4000_00C2	<R0>	0x4000_00D2		0x4000_00E2		0x4000_00F2	
0x4000_00C3	<R0>	0x4000_00D3		0x4000_00E3		0x4000_00F3	
0x4000_00C4		0x4000_00D4		0x4000_00E4		0x4000_00F4	
0x4000_00C5		0x4000_00D5		0x4000_00E5		0x4000_00F5	
0x4000_00C6		0x4000_00D6		0x4000_00E6		0x4000_00F6	
0x4000_00C7		0x4000_00D7		0x4000_00E7		0x4000_00F7	
0x4000_00C8	PDFR1	0x4000_00D8		0x4000_00E8		0x4000_00F8	PDIE
0x4000_00C9	<R0>	0x4000_00D9		0x4000_00E9		0x4000_00F9	<R0>
0x4000_00CA	<R0>	0x4000_00DA		0x4000_00EA		0x4000_00FA	<R0>
0x4000_00CB	<R0>	0x4000_00DB		0x4000_00EB		0x4000_00FB	<R0>
0x4000_00CC		0x4000_00DC		0x4000_00EC	PDPUP	0x4000_00FC	
0x4000_00CD		0x4000_00DD		0x4000_00ED	<R0>	0x4000_00FD	
0x4000_00CE		0x4000_00DE		0x4000_00EE	<R0>	0x4000_00FE	
0x4000_00CF		0x4000_00DF		0x4000_00EF	<R0>	0x4000_00FF	

<PORT E>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4000_0100	PE	0x4000_0110		0x4000_0120		0x4000_0130	
0x4000_0101	<R0>	0x4000_0111		0x4000_0121		0x4000_0131	
0x4000_0102	<R0>	0x4000_0112		0x4000_0122		0x4000_0132	
0x4000_0103	<R0>	0x4000_0113		0x4000_0123		0x4000_0133	
0x4000_0104	PECR	0x4000_0114		0x4000_0124		0x4000_0134	
0x4000_0105	<R0>	0x4000_0115		0x4000_0125		0x4000_0135	
0x4000_0106	<R0>	0x4000_0116		0x4000_0126		0x4000_0136	
0x4000_0107	<R0>	0x4000_0117		0x4000_0127		0x4000_0137	
0x4000_0108	PEFR1	0x4000_0118		0x4000_0128	PEOD	0x4000_0138	PEIE
0x4000_0109	<R0>	0x4000_0119		0x4000_0129	<R0>	0x4000_0139	<R0>
0x4000_010A	<R0>	0x4000_011A		0x4000_012A	<R0>	0x4000_013A	<R0>
0x4000_010B	<R0>	0x4000_011B		0x4000_012B	<R0>	0x4000_013B	<R0>
0x4000_010C	PEFR2	0x4000_011C		0x4000_012C	PEPUP	0x4000_013C	
0x4000_010D	<R0>	0x4000_011D		0x4000_012D	<R0>	0x4000_013D	
0x4000_010E	<R0>	0x4000_011E		0x4000_012E	<R0>	0x4000_013E	
0x4000_010F	<R0>	0x4000_011F		0x4000_012F	<R0>	0x4000_013F	

<PORT F>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4000_0140	PF	0x4000_0150		0x4000_0160		0x4000_0170	
0x4000_0141	<R0>	0x4000_0151		0x4000_0161		0x4000_0171	
0x4000_0142	<R0>	0x4000_0152		0x4000_0162		0x4000_0172	
0x4000_0143	<R0>	0x4000_0153		0x4000_0163		0x4000_0173	
0x4000_0144	PFCR	0x4000_0154		0x4000_0164		0x4000_0174	
0x4000_0145	<R0>	0x4000_0155		0x4000_0165		0x4000_0175	
0x4000_0146	<R0>	0x4000_0156		0x4000_0166		0x4000_0176	
0x4000_0147	<R0>	0x4000_0157		0x4000_0167		0x4000_0177	
0x4000_0148	PFFR1	0x4000_0158		0x4000_0168	PFOD	0x4000_0178	PFIE
0x4000_0149	<R0>	0x4000_0159		0x4000_0169	<R0>	0x4000_0179	<R0>
0x4000_014A	<R0>	0x4000_015A		0x4000_016A	<R0>	0x4000_017A	<R0>
0x4000_014B	<R0>	0x4000_015B		0x4000_016B	<R0>	0x4000_017B	<R0>
0x4000_014C	PFFR2	0x4000_015C		0x4000_016C	PFPUP	0x4000_017C	
0x4000_014D	<R0>	0x4000_015D		0x4000_016D	<R0>	0x4000_017D	
0x4000_014E	<R0>	0x4000_015E		0x4000_016E	<R0>	0x4000_017E	
0x4000_014F	<R0>	0x4000_015F		0x4000_016F	<R0>	0x4000_017F	

[1] Port 3/4]

<PORT G>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4000_0180	PG	0x4000_0190	Reserved	0x4000_01A0		0x4000_01B0	
0x4000_0181	<R0>	0x4000_0191	"	0x4000_01A1		0x4000_01B1	
0x4000_0182	<R0>	0x4000_0192	"	0x4000_01A2		0x4000_01B2	
0x4000_0183	<R0>	0x4000_0193	"	0x4000_01A3		0x4000_01B3	
0x4000_0184	PGCR	0x4000_0194		0x4000_01A4		0x4000_01B4	
0x4000_0185	<R0>	0x4000_0195		0x4000_01A5		0x4000_01B5	
0x4000_0186	<R0>	0x4000_0196		0x4000_01A6		0x4000_01B6	
0x4000_0187	<R0>	0x4000_0197		0x4000_01A7		0x4000_01B7	
0x4000_0188	PGFR1	0x4000_0198		0x4000_01A8	PGOD	0x4000_01B8	PGIE
0x4000_0189	<R0>	0x4000_0199		0x4000_01A9	<R0>	0x4000_01B9	<R0>
0x4000_018A	<R0>	0x4000_019A		0x4000_01AA	<R0>	0x4000_01BA	<R0>
0x4000_018B	<R0>	0x4000_019B		0x4000_01AB	<R0>	0x4000_01BB	<R0>
0x4000_018C		0x4000_019C		0x4000_01AC	PGPUP	0x4000_01BC	
0x4000_018D		0x4000_019D		0x4000_01AD	<R0>	0x4000_01BD	
0x4000_018E		0x4000_019E		0x4000_01AE	<R0>	0x4000_01BE	
0x4000_018F		0x4000_019F		0x4000_01AF	<R0>	0x4000_01BF	

<PORT H>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4000_01C0	PH	0x4000_01D0	Reserved	0x4000_01E0		0x4000_01F0	
0x4000_01C1	<R0>	0x4000_01D1	"	0x4000_01E1		0x4000_01F1	
0x4000_01C2	<R0>	0x4000_01D2	"	0x4000_01E2		0x4000_01F2	
0x4000_01C3	<R0>	0x4000_01D3	"	0x4000_01E3		0x4000_01F3	
0x4000_01C4	PHCR	0x4000_01D4		0x4000_01E4		0x4000_01F4	
0x4000_01C5	<R0>	0x4000_01D5		0x4000_01E5		0x4000_01F5	
0x4000_01C6	<R0>	0x4000_01D6		0x4000_01E6		0x4000_01F6	
0x4000_01C7	<R0>	0x4000_01D7		0x4000_01E7		0x4000_01F7	
0x4000_01C8	PHFR1	0x4000_01D8		0x4000_01E8		0x4000_01F8	PHIE
0x4000_01C9	<R0>	0x4000_01D9		0x4000_01E9		0x4000_01F9	<R0>
0x4000_01CA	<R0>	0x4000_01DA		0x4000_01EA		0x4000_01FA	<R0>
0x4000_01CB	<R0>	0x4000_01DB		0x4000_01EB		0x4000_01FB	<R0>
0x4000_01CC		0x4000_01DC		0x4000_01EC	PHPUP	0x4000_01FC	
0x4000_01CD		0x4000_01DD		0x4000_01ED	<R0>	0x4000_01FD	
0x4000_01CE		0x4000_01DE		0x4000_01EE	<R0>	0x4000_01FE	
0x4000_01CF		0x4000_01DF		0x4000_01EF	<R0>	0x4000_01FF	

<PORT I>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4000_0200	PI	0x4000_0210	Reserved	0x4000_0220		0x4000_0230	
0x4000_0201	<R0>	0x4000_0211	"	0x4000_0221		0x4000_0231	
0x4000_0202	<R0>	0x4000_0212	"	0x4000_0222		0x4000_0232	
0x4000_0203	<R0>	0x4000_0213	"	0x4000_0223		0x4000_0233	
0x4000_0204	PICR	0x4000_0214		0x4000_0224		0x4000_0234	
0x4000_0205	<R0>	0x4000_0215		0x4000_0225		0x4000_0235	
0x4000_0206	<R0>	0x4000_0216		0x4000_0226		0x4000_0236	
0x4000_0207	<R0>	0x4000_0217		0x4000_0227		0x4000_0237	
0x4000_0208	PIFR1	0x4000_0218		0x4000_0228		0x4000_0238	PIIE
0x4000_0209	<R0>	0x4000_0219		0x4000_0229		0x4000_0239	<R0>
0x4000_020A	<R0>	0x4000_021A		0x4000_022A		0x4000_023A	<R0>
0x4000_020B	<R0>	0x4000_021B		0x4000_022B		0x4000_023B	<R0>
0x4000_020C		0x4000_021C		0x4000_022C	PIPUP	0x4000_023C	
0x4000_020D		0x4000_021D		0x4000_022D	<R0>	0x4000_023D	
0x4000_020E		0x4000_021E		0x4000_022E	<R0>	0x4000_023E	
0x4000_020F		0x4000_021F		0x4000_022F	<R0>	0x4000_023F	

[1] Port [4/4]

<PORT J>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4000_0240	PJ	0x4000_0250	Reserved	0x4000_0260		0x4000_0270	
0x4000_0241	<R0>	0x4000_0251	"	0x4000_0261		0x4000_0271	
0x4000_0242	<R0>	0x4000_0252	"	0x4000_0262		0x4000_0272	
0x4000_0243	<R0>	0x4000_0253	"	0x4000_0263		0x4000_0273	
0x4000_0244	PJCR	0x4000_0254		0x4000_0264		0x4000_0274	
0x4000_0245	<R0>	0x4000_0255		0x4000_0265		0x4000_0275	
0x4000_0246	<R0>	0x4000_0256		0x4000_0266		0x4000_0276	
0x4000_0247	<R0>	0x4000_0257		0x4000_0267		0x4000_0277	
0x4000_0248	PJFR1	0x4000_0258		0x4000_0268		0x4000_0278	PJIE
0x4000_0249	<R0>	0x4000_0259		0x4000_0269		0x4000_0279	<R0>
0x4000_024A	<R0>	0x4000_025A		0x4000_026A		0x4000_027A	<R0>
0x4000_024B	<R0>	0x4000_025B		0x4000_026B		0x4000_027B	<R0>
0x4000_024C		0x4000_025C		0x4000_026C	PJPUP	0x4000_027C	
0x4000_024D		0x4000_025D		0x4000_026D	<R0>	0x4000_027D	
0x4000_024E		0x4000_025E		0x4000_026E	<R0>	0x4000_027E	
0x4000_024F		0x4000_025F		0x4000_026F	<R0>	0x4000_027F	

<PORT K>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4000_0280	PK	0x4000_0290		0x4000_02A0		0x4000_02B0	
0x4000_0281	<R0>	0x4000_0291		0x4000_02A1		0x4000_02B1	
0x4000_0282	<R0>	0x4000_0292		0x4000_02A2		0x4000_02B2	
0x4000_0283	<R0>	0x4000_0293		0x4000_02A3		0x4000_02B3	
0x4000_0284	PKCR	0x4000_0294		0x4000_02A4		0x4000_02B4	
0x4000_0285	<R0>	0x4000_0295		0x4000_02A5		0x4000_02B5	
0x4000_0286	<R0>	0x4000_0296		0x4000_02A6		0x4000_02B6	
0x4000_0287	<R0>	0x4000_0297		0x4000_02A7		0x4000_02B7	
0x4000_0288	PKFR1	0x4000_0298		0x4000_02A8		0x4000_02B8	PKIE
0x4000_0289	<R0>	0x4000_0299		0x4000_02A9		0x4000_02B9	<R0>
0x4000_028A	<R0>	0x4000_029A		0x4000_02AA		0x4000_02BA	<R0>
0x4000_028B	<R0>	0x4000_029B		0x4000_02AB		0x4000_02BB	<R0>
0x4000_028C	PKFR2	0x4000_029C		0x4000_02AC	PKPUP	0x4000_02BC	
0x4000_028D	<R0>	0x4000_029D		0x4000_02AD	<R0>	0x4000_02BD	
0x4000_028E	<R0>	0x4000_029E		0x4000_02AE	<R0>	0x4000_02BE	
0x4000_028F	<R0>	0x4000_029F		0x4000_02AF	<R0>	0x4000_02BF	

[2] 16-bit timer [1/4]

<TMRB0>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4001_0000	TB0EN	0x4001_0010	TB0FFCR	0x4001_0020	TB0RG0L	0x4001_0030	
0x4001_0001	<R0>	0x4001_0011	<R0>	0x4001_0021	TB0RG0H	0x4001_0031	
0x4001_0002	<R0>	0x4001_0012	<R0>	0x4001_0022	<R0>	0x4001_0032	
0x4001_0003	<R0>	0x4001_0013	<R0>	0x4001_0023	<R0>	0x4001_0033	
0x4001_0004	TB0RUN	0x4001_0014	TB0ST	0x4001_0024	TB0RG1L	0x4001_0034	
0x4001_0005	<R0>	0x4001_0015	<R0>	0x4001_0025	TB0RG1H	0x4001_0035	
0x4001_0006	<R0>	0x4001_0016	<R0>	0x4001_0026	<R0>	0x4001_0036	
0x4001_0007	<R0>	0x4001_0017	<R0>	0x4001_0027	<R0>	0x4001_0037	
0x4001_0008	TB0CR	0x4001_0018	TB0IM	0x4001_0028	TB0CP0L	0x4001_0038	
0x4001_0009	<R0>	0x4001_0019	<R0>	0x4001_0029	TB0CP0H	0x4001_0039	
0x4001_000A	<R0>	0x4001_001A	<R0>	0x4001_002A	<R0>	0x4001_003A	
0x4001_000B	<R0>	0x4001_001B	<R0>	0x4001_002B	<R0>	0x4001_003B	
0x4001_000C	TB0MOD	0x4001_001C	TB0UCL	0x4001_002C	TB0CP1L	0x4001_003C	
0x4001_000D	<R0>	0x4001_001D	TB0UCH	0x4001_002D	TB0CP1H	0x4001_003D	
0x4001_000E	<R0>	0x4001_001E	<R0>	0x4001_002E	<R0>	0x4001_003E	
0x4001_000F	<R0>	0x4001_001F	<R0>	0x4001_002F	<R0>	0x4001_003F	

<TMRB1>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4001_0040	TB1EN	0x4001_0050	TB1FFCR	0x4001_0060	TB1RG0L	0x4001_0070	
0x4001_0041	<R0>	0x4001_0051	<R0>	0x4001_0061	TB1RG0H	0x4001_0071	
0x4001_0042	<R0>	0x4001_0052	<R0>	0x4001_0062	<R0>	0x4001_0072	
0x4001_0043	<R0>	0x4001_0053	<R0>	0x4001_0063	<R0>	0x4001_0073	
0x4001_0044	TB1RUN	0x4001_0054	TB1ST	0x4001_0064	TB1RG1L	0x4001_0074	
0x4001_0045	<R0>	0x4001_0055	<R0>	0x4001_0065	TB1RG1H	0x4001_0075	
0x4001_0046	<R0>	0x4001_0056	<R0>	0x4001_0066	<R0>	0x4001_0076	
0x4001_0047	<R0>	0x4001_0057	<R0>	0x4001_0067	<R0>	0x4001_0077	
0x4001_0048	TB1CR	0x4001_0058	TB1IM	0x4001_0068	TB1CP0L	0x4001_0078	
0x4001_0049	<R0>	0x4001_0059	<R0>	0x4001_0069	TB1CP0H	0x4001_0079	
0x4001_004A	<R0>	0x4001_005A	<R0>	0x4001_006A	<R0>	0x4001_007A	
0x4001_004B	<R0>	0x4001_005B	<R0>	0x4001_006B	<R0>	0x4001_007B	
0x4001_004C	TB1MOD	0x4001_005C	TB1UCL	0x4001_006C	TB1CP1L	0x4001_007C	
0x4001_004D	<R0>	0x4001_005D	TB1UCH	0x4001_006D	TB1CP1H	0x4001_007D	
0x4001_004E	<R0>	0x4001_005E	<R0>	0x4001_006E	<R0>	0x4001_007E	
0x4001_004F	<R0>	0x4001_005F	<R0>	0x4001_006F	<R0>	0x4001_007F	

<TMRB2>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4001_0080	TB2EN	0x4001_0090	TB2FFCR	0x4001_00A0	TB2RG0L	0x4001_00B0	
0x4001_0081	<R0>	0x4001_0091	<R0>	0x4001_00A1	TB2RG0H	0x4001_00B1	
0x4001_0082	<R0>	0x4001_0092	<R0>	0x4001_00A2	<R0>	0x4001_00B2	
0x4001_0083	<R0>	0x4001_0093	<R0>	0x4001_00A3	<R0>	0x4001_00B3	
0x4001_0084	TB2RUN	0x4001_0094	TB2ST	0x4001_00A4	TB2RG1L	0x4001_00B4	
0x4001_0085	<R0>	0x4001_0095	<R0>	0x4001_00A5	TB2RG1H	0x4001_00B5	
0x4001_0086	<R0>	0x4001_0096	<R0>	0x4001_00A6	<R0>	0x4001_00B6	
0x4001_0087	<R0>	0x4001_0097	<R0>	0x4001_00A7	<R0>	0x4001_00B7	
0x4001_0088	TB2CR	0x4001_0098	TB2IM	0x4001_00A8	TB2CP0L	0x4001_00B8	
0x4001_0089	<R0>	0x4001_0099	<R0>	0x4001_00A9	TB2CP0H	0x4001_00B9	
0x4001_008A	<R0>	0x4001_009A	<R0>	0x4001_00AA	<R0>	0x4001_00BA	
0x4001_008B	<R0>	0x4001_009B	<R0>	0x4001_00AB	<R0>	0x4001_00BB	
0x4001_008C	TB2MOD	0x4001_009C	TB2UCL	0x4001_00AC	TB2CP1L	0x4001_00BC	
0x4001_008D	<R0>	0x4001_009D	TB2UCH	0x4001_00AD	TB2CP1H	0x4001_00BD	
0x4001_008E	<R0>	0x4001_009E	<R0>	0x4001_00AE	<R0>	0x4001_00BE	
0x4001_008F	<R0>	0x4001_009F	<R0>	0x4001_00AF	<R0>	0x4001_00BF	

[2] 16-bit timer [2/4]

<TMRB3>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4001_00C0	TB3EN	0x4001_00D0	TB3FFCR	0x4001_00E0	TB3RG0L	0x4001_00F0	
0x4001_00C1	<R0>	0x4001_00D1	<R0>	0x4001_00E1	TB3RG0H	0x4001_00F1	
0x4001_00C2	<R0>	0x4001_00D2	<R0>	0x4001_00E2	<R0>	0x4001_00F2	
0x4001_00C3	<R0>	0x4001_00D3	<R0>	0x4001_00E3	<R0>	0x4001_00F3	
0x4001_00C4	TB3RUN	0x4001_00D4	TB3ST	0x4001_00E4	TB3RG1L	0x4001_00F4	
0x4001_00C5	<R0>	0x4001_00D5	<R0>	0x4001_00E5	TB3RG1H	0x4001_00F5	
0x4001_00C6	<R0>	0x4001_00D6	<R0>	0x4001_00E6	<R0>	0x4001_00F6	
0x4001_00C7	<R0>	0x4001_00D7	<R0>	0x4001_00E7	<R0>	0x4001_00F7	
0x4001_00C8	TB3CR	0x4001_00D8	TB3IM	0x4001_00E8	TB3CP0L	0x4001_00F8	
0x4001_00C9	<R0>	0x4001_00D9	<R0>	0x4001_00E9	TB3CP0H	0x4001_00F9	
0x4001_00CA	<R0>	0x4001_00DA	<R0>	0x4001_00EA	<R0>	0x4001_00FA	
0x4001_00CB	<R0>	0x4001_00DB	<R0>	0x4001_00EB	<R0>	0x4001_00FB	
0x4001_00CC	TB3MOD	0x4001_00DC	TB3UCL	0x4001_00EC	TB3CP1L	0x4001_00FC	
0x4001_00CD	<R0>	0x4001_00DD	TB3UCH	0x4001_00ED	TB3CP1H	0x4001_00FD	
0x4001_00CE	<R0>	0x4001_00DE	<R0>	0x4001_00EE	<R0>	0x4001_00FE	
0x4001_00CF	<R0>	0x4001_00DF	<R0>	0x4001_00EF	<R0>	0x4001_00FF	

<TMRB4>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4001_0100	TB4EN	0x4001_0110	TB4FFCR	0x4001_0120	TB4RG0L	0x4001_0130	
0x4001_0101	<R0>	0x4001_0111	<R0>	0x4001_0121	TB4RG0H	0x4001_0131	
0x4001_0102	<R0>	0x4001_0112	<R0>	0x4001_0122	<R0>	0x4001_0132	
0x4001_0103	<R0>	0x4001_0113	<R0>	0x4001_0123	<R0>	0x4001_0133	
0x4001_0104	TB4RUN	0x4001_0114	TB4ST	0x4001_0124	TB4RG1L	0x4001_0134	
0x4001_0105	<R0>	0x4001_0115	<R0>	0x4001_0125	TB4RG1H	0x4001_0135	
0x4001_0106	<R0>	0x4001_0116	<R0>	0x4001_0126	<R0>	0x4001_0136	
0x4001_0107	<R0>	0x4001_0117	<R0>	0x4001_0127	<R0>	0x4001_0137	
0x4001_0108	TB4CR	0x4001_0118	TB4IM	0x4001_0128	TB4CP0L	0x4001_0138	
0x4001_0109	<R0>	0x4001_0119	<R0>	0x4001_0129	TB4CP0H	0x4001_0139	
0x4001_010A	<R0>	0x4001_011A	<R0>	0x4001_012A	<R0>	0x4001_013A	
0x4001_010B	<R0>	0x4001_011B	<R0>	0x4001_012B	<R0>	0x4001_013B	
0x4001_010C	TB4MOD	0x4001_011C	TB4UCL	0x4001_012C	TB4CP1L	0x4001_013C	
0x4001_010D	<R0>	0x4001_011D	TB4UCH	0x4001_012D	TB4CP1H	0x4001_013D	
0x4001_010E	<R0>	0x4001_011E	<R0>	0x4001_012E	<R0>	0x4001_013E	
0x4001_010F	<R0>	0x4001_011F	<R0>	0x4001_012F	<R0>	0x4001_013F	

<TMRB5>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4001_0140	TB5EN	0x4001_0150	TB5FFCR	0x4001_0160	TB5RG0L	0x4001_0170	
0x4001_0141	<R0>	0x4001_0151	<R0>	0x4001_0161	TB5RG0H	0x4001_0171	
0x4001_0142	<R0>	0x4001_0152	<R0>	0x4001_0162	<R0>	0x4001_0172	
0x4001_0143	<R0>	0x4001_0153	<R0>	0x4001_0163	<R0>	0x4001_0173	
0x4001_0144	TB5RUN	0x4001_0154	TB5ST	0x4001_0164	TB5RG1L	0x4001_0174	
0x4001_0145	<R0>	0x4001_0155	<R0>	0x4001_0165	TB5RG1H	0x4001_0175	
0x4001_0146	<R0>	0x4001_0156	<R0>	0x4001_0166	<R0>	0x4001_0176	
0x4001_0147	<R0>	0x4001_0157	<R0>	0x4001_0167	<R0>	0x4001_0177	
0x4001_0148	TB5CR	0x4001_0158	TB5IM	0x4001_0168	TB5CP0L	0x4001_0178	
0x4001_0149	<R0>	0x4001_0159	<R0>	0x4001_0169	TB5CP0H	0x4001_0179	
0x4001_014A	<R0>	0x4001_015A	<R0>	0x4001_016A	<R0>	0x4001_017A	
0x4001_014B	<R0>	0x4001_015B	<R0>	0x4001_016B	<R0>	0x4001_017B	
0x4001_014C	TB5MOD	0x4001_015C	TB5UCL	0x4001_016C	TB5CP1L	0x4001_017C	
0x4001_014D	<R0>	0x4001_015D	TB5UCH	0x4001_016D	TB5CP1H	0x4001_017D	
0x4001_014E	<R0>	0x4001_015E	<R0>	0x4001_016E	<R0>	0x4001_017E	
0x4001_014F	<R0>	0x4001_015F	<R0>	0x4001_016F	<R0>	0x4001_017F	

[2] 16-bit timer [3/4]

<TMRB6>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4001_0180	TB6EN	0x4001_0190	TB6FFCR	0x4001_01A0	TB6RG0L	0x4001_01B0	
0x4001_0181	<R0>	0x4001_0191	<R0>	0x4001_01A1	TB6RG0H	0x4001_01B1	
0x4001_0182	<R0>	0x4001_0192	<R0>	0x4001_01A2	<R0>	0x4001_01B2	
0x4001_0183	<R0>	0x4001_0193	<R0>	0x4001_01A3	<R0>	0x4001_01B3	
0x4001_0184	TB6RUN	0x4001_0194	TB6ST	0x4001_01A4	TB6RG1L	0x4001_01B4	
0x4001_0185	<R0>	0x4001_0195	<R0>	0x4001_01A5	TB6RG1H	0x4001_01B5	
0x4001_0186	<R0>	0x4001_0196	<R0>	0x4001_01A6	<R0>	0x4001_01B6	
0x4001_0187	<R0>	0x4001_0197	<R0>	0x4001_01A7	<R0>	0x4001_01B7	
0x4001_0188	TB6CR	0x4001_0198	TB6IM	0x4001_01A8	TB6CP0L	0x4001_01B8	
0x4001_0189	<R0>	0x4001_0199	<R0>	0x4001_01A9	TB6CP0H	0x4001_01B9	
0x4001_018A	<R0>	0x4001_019A	<R0>	0x4001_01AA	<R0>	0x4001_01BA	
0x4001_018B	<R0>	0x4001_019B	<R0>	0x4001_01AB	<R0>	0x4001_01BB	
0x4001_018C	TB6MOD	0x4001_019C	TB6UCL	0x4001_01AC	TB6CP1L	0x4001_01BC	
0x4001_018D	<R0>	0x4001_019D	TB6UCH	0x4001_01AD	TB6CP1H	0x4001_01BD	
0x4001_018E	<R0>	0x4001_019E	<R0>	0x4001_01AE	<R0>	0x4001_01BE	
0x4001_018F	<R0>	0x4001_019F	<R0>	0x4001_01AF	<R0>	0x4001_01BF	

<TMRB7>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4001_01C0	TB7EN	0x4001_01D0	TB7FFCR	0x4001_01E0	TB7RG0L	0x4001_01F0	
0x4001_01C1	<R0>	0x4001_01D1	<R0>	0x4001_01E1	TB7RG0H	0x4001_01F1	
0x4001_01C2	<R0>	0x4001_01D2	<R0>	0x4001_01E2	<R0>	0x4001_01F2	
0x4001_01C3	<R0>	0x4001_01D3	<R0>	0x4001_01E3	<R0>	0x4001_01F3	
0x4001_01C4	TB7RUN	0x4001_01D4	TB7ST	0x4001_01E4	TB7RG1L	0x4001_01F4	
0x4001_01C5	<R0>	0x4001_01D5	<R0>	0x4001_01E5	TB7RG1H	0x4001_01F5	
0x4001_01C6	<R0>	0x4001_01D6	<R0>	0x4001_01E6	<R0>	0x4001_01F6	
0x4001_01C7	<R0>	0x4001_01D7	<R0>	0x4001_01E7	<R0>	0x4001_01F7	
0x4001_01C8	TB7CR	0x4001_01D8	TB7IM	0x4001_01E8	TB7CP0L	0x4001_01F8	
0x4001_01C9	<R0>	0x4001_01D9	<R0>	0x4001_01E9	TB7CP0H	0x4001_01F9	
0x4001_01CA	<R0>	0x4001_01DA	<R0>	0x4001_01EA	<R0>	0x4001_01FA	
0x4001_01CB	<R0>	0x4001_01DB	<R0>	0x4001_01EB	<R0>	0x4001_01FB	
0x4001_01CC	TB7MOD	0x4001_01DC	TB7UCL	0x4001_01EC	TB7CP1L	0x4001_01FC	
0x4001_01CD	<R0>	0x4001_01DD	TB7UCH	0x4001_01ED	TB7CP1H	0x4001_01FD	
0x4001_01CE	<R0>	0x4001_01DE	<R0>	0x4001_01EE	<R0>	0x4001_01FE	
0x4001_01CF	<R0>	0x4001_01DF	<R0>	0x4001_01EF	<R0>	0x4001_01FF	

<TMRB8>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4001_0200	TB8EN	0x4001_0210	TB8FFCR	0x4001_0220	TB8RG0L	0x4001_0230	
0x4001_0201	<R0>	0x4001_0211	<R0>	0x4001_0221	TB8RG0H	0x4001_0231	
0x4001_0202	<R0>	0x4001_0212	<R0>	0x4001_0222	<R0>	0x4001_0232	
0x4001_0203	<R0>	0x4001_0213	<R0>	0x4001_0223	<R0>	0x4001_0233	
0x4001_0204	TB8RUN	0x4001_0214	TB8ST	0x4001_0224	TB8RG1L	0x4001_0234	
0x4001_0205	<R0>	0x4001_0215	<R0>	0x4001_0225	TB8RG1H	0x4001_0235	
0x4001_0206	<R0>	0x4001_0216	<R0>	0x4001_0226	<R0>	0x4001_0236	
0x4001_0207	<R0>	0x4001_0217	<R0>	0x4001_0227	<R0>	0x4001_0237	
0x4001_0208	TB8CR	0x4001_0218	TB8IM	0x4001_0228	TB8CP0L	0x4001_0238	
0x4001_0209	<R0>	0x4001_0219	<R0>	0x4001_0229	TB8CP0H	0x4001_0239	
0x4001_020A	<R0>	0x4001_021A	<R0>	0x4001_022A	<R0>	0x4001_023A	
0x4001_020B	<R0>	0x4001_021B	<R0>	0x4001_022B	<R0>	0x4001_023B	
0x4001_020C	TB8MOD	0x4001_021C	TB8UCL	0x4001_022C	TB8CP1L	0x4001_023C	
0x4001_020D	<R0>	0x4001_021D	TB8UCH	0x4001_022D	TB8CP1H	0x4001_023D	
0x4001_020E	<R0>	0x4001_021E	<R0>	0x4001_022E	<R0>	0x4001_023E	
0x4001_020F	<R0>	0x4001_021F	<R0>	0x4001_022F	<R0>	0x4001_023F	

[2] 16-bit timer [4/4]

<TMRB9>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4001_0240	TB9EN	0x4001_0250	TB9FFCR	0x4001_0260	TB9RG0L	0x4001_0270	
0x4001_0241	<R0>	0x4001_0251	<R0>	0x4001_0261	TB9RG0H	0x4001_0271	
0x4001_0242	<R0>	0x4001_0252	<R0>	0x4001_0262	<R0>	0x4001_0272	
0x4001_0243	<R0>	0x4001_0253	<R0>	0x4001_0263	<R0>	0x4001_0273	
0x4001_0244	TB9RUN	0x4001_0254	TB9ST	0x4001_0264	TB9RG1L	0x4001_0274	
0x4001_0245	<R0>	0x4001_0255	<R0>	0x4001_0265	TB9RG1H	0x4001_0275	
0x4001_0246	<R0>	0x4001_0256	<R0>	0x4001_0266	<R0>	0x4001_0276	
0x4001_0247	<R0>	0x4001_0257	<R0>	0x4001_0267	<R0>	0x4001_0277	
0x4001_0248	TB9CR	0x4001_0258	TB9IM	0x4001_0268	TB9CP0L	0x4001_0278	
0x4001_0249	<R0>	0x4001_0259	<R0>	0x4001_0269	TB9CP0H	0x4001_0279	
0x4001_024A	<R0>	0x4001_025A	<R0>	0x4001_026A	<R0>	0x4001_027A	
0x4001_024B	<R0>	0x4001_025B	<R0>	0x4001_026B	<R0>	0x4001_027B	
0x4001_024C	TB9MOD	0x4001_025C	TB9UCL	0x4001_026C	TB9CP1L	0x4001_027C	
0x4001_024D	<R0>	0x4001_025D	TB9UCH	0x4001_026D	TB9CP1H	0x4001_027D	
0x4001_024E	<R0>	0x4001_025E	<R0>	0x4001_026E	<R0>	0x4001_027E	
0x4001_024F	<R0>	0x4001_025F	<R0>	0x4001_026F	<R0>	0x4001_027F	

[3] Serial bus interface (SBI)

<SBI0>

ADR	Register name
0x4002_0000	SBI0CR0
0x4002_0001	<R0>
0x4002_0002	<R0>
0x4002_0003	<R0>
0x4002_0004	SBI0CR1
0x4002_0005	<R0>
0x4002_0006	<R0>
0x4002_0007	<R0>
0x4002_0008	SBI0DBR
0x4002_0009	<R0>
0x4002_000A	<R0>
0x4002_000B	<R0>
0x4002_000C	SBI0I2CAR
0x4002_000D	<R0>
0x4002_000E	<R0>
0x4002_000F	<R0>

ADR	Register name
0x4002_0010	SBI0CR2/SR
0x4002_0011	<R0>
0x4002_0012	<R0>
0x4002_0013	<R0>
0x4002_0014	SBI0BR0
0x4002_0015	<R0>
0x4002_0016	<R0>
0x4002_0017	<R0>
0x4002_0018	
0x4002_0019	
0x4002_001A	
0x4002_001B	
0x4002_001C	
0x4002_001D	
0x4002_001E	
0x4002_001F	

<SBI1>

ADR	Register name
0x4002_0020	SBI1CR0
0x4002_0021	<R0>
0x4002_0022	<R0>
0x4002_0023	<R0>
0x4002_0024	SBI1CR1
0x4002_0025	<R0>
0x4002_0026	<R0>
0x4002_0027	<R0>
0x4002_0028	SBI1DBR
0x4002_0029	<R0>
0x4002_002A	<R0>
0x4002_002B	<R0>
0x4002_002C	SBI1I2CAR
0x4002_002D	<R0>
0x4002_002E	<R0>
0x4002_002F	<R0>

ADR	Register name
0x4002_0030	SBI1CR2/SR
0x4002_0031	<R0>
0x4002_0032	<R0>
0x4002_0033	<R0>
0x4002_0034	SBI1BR0
0x4002_0035	<R0>
0x4002_0036	<R0>
0x4002_0037	<R0>
0x4002_0038	
0x4002_0039	
0x4002_003A	
0x4002_003B	
0x4002_003C	
0x4002_003D	
0x4002_003E	
0x4002_003F	

<SBI2>

ADR	Register name
0x4002_0040	SBI2CR0
0x4002_0041	<R0>
0x4002_0042	<R0>
0x4002_0043	<R0>
0x4002_0044	SBI2CR1
0x4002_0045	<R0>
0x4002_0046	<R0>
0x4002_0047	<R0>
0x4002_0048	SBI2DBR
0x4002_0049	<R0>
0x4002_004A	<R0>
0x4002_004B	<R0>
0x4002_004C	SBI2I2CAR
0x4002_004D	<R0>
0x4002_004E	<R0>
0x4002_004F	<R0>

ADR	Register name
0x4002_0050	SBI2CR2/SR
0x4002_0051	<R0>
0x4002_0052	<R0>
0x4002_0053	<R0>
0x4002_0054	SBI2BR0
0x4002_0055	<R0>
0x4002_0056	<R0>
0x4002_0057	<R0>
0x4002_0058	
0x4002_0059	
0x4002_005A	
0x4002_005B	
0x4002_005C	
0x4002_005D	
0x4002_005E	
0x4002_005F	

ADR	Register name
0x4002_0060	
0x4002_0061	
0x4002_0062	
0x4002_0063	
0x4002_0064	
0x4002_0065	
0x4002_0066	
0x4002_0067	
0x4002_0068	
0x4002_0069	
0x4002_006A	
0x4002_006B	
0x4002_006C	
0x4002_006D	
0x4002_006E	
0x4002_006F	

ADR	Register name
0x4002_0070	
0x4002_0071	
0x4002_0072	
0x4002_0073	
0x4002_0074	
0x4002_0075	
0x4002_0076	
0x4002_0077	
0x4002_0078	
0x4002_0079	
0x4002_007A	
0x4002_007B	
0x4002_007C	
0x4002_007D	
0x4002_007E	
0x4002_007F	

[4] Serial interface (UART/SIO)

<SIO0>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4002_0080	SC0EN	0x4002_0090	SC0BRCCR	0x4002_00A0	SC0RFC	0x4002_00B0	SC0FCNF
0x4002_0081	<R0>	0x4002_0091	<R0>	0x4002_00A1	<R0>	0x4002_00B1	<R0>
0x4002_0082	<R0>	0x4002_0092	<R0>	0x4002_00A2	<R0>	0x4002_00B2	<R0>
0x4002_0083	<R0>	0x4002_0093	<R0>	0x4002_00A3	<R0>	0x4002_00B3	<R0>
0x4002_0084	SC0BUF	0x4002_0094	SC0BRADD	0x4002_00A4	SC0TFC	0x4002_00B4	
0x4002_0085	<R0>	0x4002_0095	<R0>	0x4002_00A5	<R0>	0x4002_00B5	
0x4002_0086	<R0>	0x4002_0096	<R0>	0x4002_00A6	<R0>	0x4002_00B6	
0x4002_0087	<R0>	0x4002_0097	<R0>	0x4002_00A7	<R0>	0x4002_00B7	
0x4002_0088	SC0CR	0x4002_0098	SC0MOD1	0x4002_00A8	SC0RST	0x4002_00B8	
0x4002_0089	<R0>	0x4002_0099	<R0>	0x4002_00A9	<R0>	0x4002_00B9	
0x4002_008A	<R0>	0x4002_009A	<R0>	0x4002_00AA	<R0>	0x4002_00BA	
0x4002_008B	<R0>	0x4002_009B	<R0>	0x4002_00AB	<R0>	0x4002_00BB	
0x4002_008C	SC0MOD0	0x4002_009C	SC0MOD2	0x4002_00AC	SC0TST	0x4002_00BC	
0x4002_008D	<R0>	0x4002_009D	<R0>	0x4002_00AD	<R0>	0x4002_00BD	
0x4002_008E	<R0>	0x4002_009E	<R0>	0x4002_00AE	<R0>	0x4002_00BE	
0x4002_008F	<R0>	0x4002_009F	<R0>	0x4002_00AF	<R0>	0x4002_00BF	

<SIO1>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4002_00C0	SC1EN	0x4002_00D0	SC1BRCCR	0x4002_00E0	SC1RFC	0x4002_00F0	SC1FCNF
0x4002_00C1	<R0>	0x4002_00D1	<R0>	0x4002_00E1	<R0>	0x4002_00F1	<R0>
0x4002_00C2	<R0>	0x4002_00D2	<R0>	0x4002_00E2	<R0>	0x4002_00F2	<R0>
0x4002_00C3	<R0>	0x4002_00D3	<R0>	0x4002_00E3	<R0>	0x4002_00F3	<R0>
0x4002_00C4	SC1BUF	0x4002_00D4	SC1BRADD	0x4002_00E4	SC1TFC	0x4002_00F4	
0x4002_00C5	<R0>	0x4002_00D5	<R0>	0x4002_00E5	<R0>	0x4002_00F5	
0x4002_00C6	<R0>	0x4002_00D6	<R0>	0x4002_00E6	<R0>	0x4002_00F6	
0x4002_00C7	<R0>	0x4002_00D7	<R0>	0x4002_00E7	<R0>	0x4002_00F7	
0x4002_00C8	SC1CR	0x4002_00D8	SC1MOD1	0x4002_00E8	SC1RST	0x4002_00F8	
0x4002_00C9	<R0>	0x4002_00D9	<R0>	0x4002_00E9	<R0>	0x4002_00F9	
0x4002_00CA	<R0>	0x4002_00DA	<R0>	0x4002_00EA	<R0>	0x4002_00FA	
0x4002_00CB	<R0>	0x4002_00DB	<R0>	0x4002_00EB	<R0>	0x4002_00FB	
0x4002_00CC	SC1MOD0	0x4002_00DC	SC1MOD2	0x4002_00EC	SC1TST	0x4002_00FC	
0x4002_00CD	<R0>	0x4002_00DD	<R0>	0x4002_00ED	<R0>	0x4002_00FD	
0x4002_00CE	<R0>	0x4002_00DE	<R0>	0x4002_00EE	<R0>	0x4002_00FE	
0x4002_00CF	<R0>	0x4002_00DF	<R0>	0x4002_00EF	<R0>	0x4002_00FF	

<SIO2>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4002_0100	SC2EN	0x4002_0110	SC2BRCCR	0x4002_0120	SC2RFC	0x4002_0130	SC2FCNF
0x4002_0101	<R0>	0x4002_0111	<R0>	0x4002_0121	<R0>	0x4002_0131	<R0>
0x4002_0102	<R0>	0x4002_0112	<R0>	0x4002_0122	<R0>	0x4002_0132	<R0>
0x4002_0103	<R0>	0x4002_0113	<R0>	0x4002_0123	<R0>	0x4002_0133	<R0>
0x4002_0104	SC2BUF	0x4002_0114	SC2BRADD	0x4002_0124	SC2TFC	0x4002_0134	
0x4002_0105	<R0>	0x4002_0115	<R0>	0x4002_0125	<R0>	0x4002_0135	
0x4002_0106	<R0>	0x4002_0116	<R0>	0x4002_0126	<R0>	0x4002_0136	
0x4002_0107	<R0>	0x4002_0117	<R0>	0x4002_0127	<R0>	0x4002_0137	
0x4002_0108	SC2CR	0x4002_0118	SC2MOD1	0x4002_0128	SC2RST	0x4002_0138	
0x4002_0109	<R0>	0x4002_0119	<R0>	0x4002_0129	<R0>	0x4002_0139	
0x4002_010A	<R0>	0x4002_011A	<R0>	0x4002_012A	<R0>	0x4002_013A	
0x4002_010B	<R0>	0x4002_011B	<R0>	0x4002_012B	<R0>	0x4002_013B	
0x4002_010C	SC2MOD0	0x4002_011C	SC2MOD2	0x4002_012C	SC2TST	0x4002_013C	
0x4002_010D	<R0>	0x4002_011D	<R0>	0x4002_012D	<R0>	0x4002_013D	
0x4002_010E	<R0>	0x4002_011E	<R0>	0x4002_012E	<R0>	0x4002_013E	
0x4002_010F	<R0>	0x4002_011F	<R0>	0x4002_012F	<R0>	0x4002_013F	

[5] 10-bit A/D converter (A/D)

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4003_0000	ADCLK	0x4003_0010	ADMOD3	0x4003_0020		0x4003_0030	ADREG08L
0x4003_0001	<R0>	0x4003_0011	<R0>	0x4003_0021		0x4003_0031	ADREG08H
0x4003_0002	<R0>	0x4003_0012	<R0>	0x4003_0022		0x4003_0032	<R0>
0x4003_0003	<R0>	0x4003_0013	<R0>	0x4003_0023		0x4003_0033	<R0>
0x4003_0004	ADMOD0	0x4003_0014	ADMOD4	0x4003_0024		0x4003_0034	ADREG19L
0x4003_0005	<R0>	0x4003_0015	<R0>	0x4003_0025		0x4003_0035	ADREG19H
0x4003_0006	<R0>	0x4003_0016	<R0>	0x4003_0026		0x4003_0036	<R0>
0x4003_0007	<R0>	0x4003_0017	<R0>	0x4003_0027		0x4003_0037	<R0>
0x4003_0008	ADMOD1	0x4003_0018	ADMOD5	0x4003_0028		0x4003_0038	ADREG2AL
0x4003_0009	<R0>	0x4003_0019	<R0>	0x4003_0029		0x4003_0039	ADREG2AH
0x4003_000A	<R0>	0x4003_001A	<R0>	0x4003_002A		0x4003_003A	<R0>
0x4003_000B	<R0>	0x4003_001B	<R0>	0x4003_002B		0x4003_003B	<R0>
0x4003_000C	ADMOD2	0x4003_001C		0x4003_002C		0x4003_003C	ADREG3BL
0x4003_000D	<R0>	0x4003_001D		0x4003_002D		0x4003_003D	ADREG3BH
0x4003_000E	<R0>	0x4003_001E		0x4003_002E		0x4003_003E	<R0>
0x4003_000F	<R0>	0x4003_001F		0x4003_002F		0x4003_003F	<R0>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4003_0040	ADREG4CL	0x4003_0050	ADREGSPL	0x4003_0060		0x4003_0070	
0x4003_0041	ADREG4CH	0x4003_0051	ADREGSPH	0x4003_0061		0x4003_0071	
0x4003_0042	<R0>	0x4003_0052	<R0>	0x4003_0062		0x4003_0072	
0x4003_0043	<R0>	0x4003_0053	<R0>	0x4003_0063		0x4003_0073	
0x4003_0044	ADREG5DL	0x4003_0054	ADCMP0L	0x4003_0064		0x4003_0074	
0x4003_0045	ADREG5DH	0x4003_0055	ADCMP0H	0x4003_0065		0x4003_0075	
0x4003_0046	<R0>	0x4003_0056	<R0>	0x4003_0066		0x4003_0076	
0x4003_0047	<R0>	0x4003_0057	<R0>	0x4003_0067		0x4003_0077	
0x4003_0048	ADREG6EL	0x4003_0058	ADCMP1L	0x4003_0068		0x4003_0078	
0x4003_0049	ADREG6EH	0x4003_0059	ADCMP1H	0x4003_0069		0x4003_0079	
0x4003_004A	<R0>	0x4003_005A	<R0>	0x4003_006A		0x4003_007A	
0x4003_004B	<R0>	0x4003_005B	<R0>	0x4003_006B		0x4003_007B	
0x4003_004C	ADREG7FL	0x4003_005C		0x4003_006C		0x4003_007C	
0x4003_004D	ADREG7FH	0x4003_005D		0x4003_006D		0x4003_007D	
0x4003_004E	<R0>	0x4003_005E		0x4003_006E		0x4003_007E	
0x4003_004F	<R0>	0x4003_005F		0x4003_006F		0x4003_007F	

[6] Watchdog timer (WDT)

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4004_0000	WDMOD	0x4004_0010		0x4004_0020		0x4004_0030	
0x4004_0001	<R0>	0x4004_0011		0x4004_0021		0x4004_0031	
0x4004_0002	<R0>	0x4004_0012		0x4004_0022		0x4004_0032	
0x4004_0003	<R0>	0x4004_0013		0x4004_0023		0x4004_0033	
0x4004_0004	WDCR	0x4004_0014		0x4004_0024		0x4004_0034	
0x4004_0005	<R0>	0x4004_0015		0x4004_0025		0x4004_0035	
0x4004_0006	<R0>	0x4004_0016		0x4004_0026		0x4004_0036	
0x4004_0007	<R0>	0x4004_0017		0x4004_0027		0x4004_0037	
0x4004_0008		0x4004_0018		0x4004_0028		0x4004_0038	
0x4004_0009		0x4004_0019		0x4004_0029		0x4004_0039	
0x4004_000A		0x4004_001A		0x4004_002A		0x4004_003A	
0x4004_000B		0x4004_001B		0x4004_002B		0x4004_003B	
0x4004_000C		0x4004_001C		0x4004_002C		0x4004_003C	
0x4004_000D		0x4004_001D		0x4004_002D		0x4004_003D	
0x4004_000E		0x4004_001E		0x4004_002E		0x4004_003E	
0x4004_000F		0x4004_001F		0x4004_002F		0x4004_003F	

[7] Real time clock (RTC)

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4004_0100	SECR	0x4004_0110		0x4004_0120		0x4004_0130	
0x4004_0101	MINR	0x4004_0111		0x4004_0121		0x4004_0131	
0x4004_0102	HOURLR	0x4004_0112		0x4004_0122		0x4004_0132	
0x4004_0103	<R0>	0x4004_0113		0x4004_0123		0x4004_0133	
0x4004_0104	DAYR	0x4004_0114		0x4004_0124		0x4004_0134	
0x4004_0105	DATER	0x4004_0115		0x4004_0125		0x4004_0135	
0x4004_0106	MONTHR	0x4004_0116		0x4004_0126		0x4004_0136	
0x4004_0107	YEARR	0x4004_0117		0x4004_0127		0x4004_0137	
0x4004_0108	PAGER	0x4004_0118		0x4004_0128		0x4004_0138	
0x4004_0109	<R0>	0x4004_0119		0x4004_0129		0x4004_0139	
0x4004_010A	<R0>	0x4004_011A		0x4004_012A		0x4004_013A	
0x4004_010B	<R0>	0x4004_011B		0x4004_012B		0x4004_013B	
0x4004_010C	RESTR	0x4004_011C		0x4004_012C		0x4004_013C	
0x4004_010D	<reserved>	0x4004_011D		0x4004_012D		0x4004_013D	
0x4004_010E	<R0>	0x4004_011E		0x4004_012E		0x4004_013E	
0x4004_010F	<R0>	0x4004_011F		0x4004_012F		0x4004_013F	

[8] Clock generator (CG)

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4004_0200	SYSCR0	0x4004_0210	CKSEL	0x4004_0220	IMCGA0	0x4004_0230	<reserved>
0x4004_0201	SYSCR1	0x4004_0211	<R0>	0x4004_0221	IMCGA1	0x4004_0231	<reserved>
0x4004_0202	SYSCR2	0x4004_0212	<R0>	0x4004_0222	IMCGA2	0x4004_0232	<reserved>
0x4004_0203	<R0>	0x4004_0213	<R0>	0x4004_0223	IMCGA3	0x4004_0233	<reserved>
0x4004_0204	OSCCR0	0x4004_0214	ICRCG	0x4004_0224	IMCGB0	0x4004_0234	<reserved>
0x4004_0205	OSCCR1	0x4004_0215	<R0>	0x4004_0225	IMCGB1	0x4004_0235	<reserved>
0x4004_0206	<R0>	0x4004_0216	<R0>	0x4004_0226	IMCGB2	0x4004_0236	<reserved>
0x4004_0207	<R0>	0x4004_0217	<R0>	0x4004_0227	IMCGB3	0x4004_0237	<reserved>
0x4004_0208	STBYCR0	0x4004_0218	NMIFLG	0x4004_0228	IMCGC0	0x4004_0238	<reserved>
0x4004_0209	STBYCR1	0x4004_0219	<R0>	0x4004_0229	IMCGC1	0x4004_0239	<reserved>
0x4004_020A	STBYCR2	0x4004_021A	<R0>	0x4004_022A	IMCGC2	0x4004_023A	<reserved>
0x4004_020B	<R0>	0x4004_021B	<R0>	0x4004_022B	IMCGC3	0x4004_023B	<reserved>
0x4004_020C	PLLSEL	0x4004_021C	RSTFLG	0x4004_022C	IMCGD0	0x4004_023C	<reserved>
0x4004_020D	<R0>	0x4004_021D	<R0>	0x4004_022D	<reserved>	0x4004_023D	<reserved>
0x4004_020E	<R0>	0x4004_021E	<R0>	0x4004_022E	<reserved>	0x4004_023E	<reserved>
0x4004_020F	<R0>	0x4004_021F	<R0>	0x4004_022F	<reserved>	0x4004_023F	<reserved>

[9] CEC

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4004_0300	CECEN	0x4004_0310	CECRBUF	0x4004_0320	CECTEN	0x4004_0330	CECTSTAT
0x4004_0301	<R0>	0x4004_0311	"	0x4004_0321	<R0>	0x4004_0331	<R0>
0x4004_0302	<R0>	0x4004_0312	<R0>	0x4004_0322	<R0>	0x4004_0332	<R0>
0x4004_0303	<R0>	0x4004_0313	<R0>	0x4004_0323	<R0>	0x4004_0333	<R0>
0x4004_0304	CECADD	0x4004_0314	CECRCR1	0x4004_0324	CECTBUF	0x4004_0334	
0x4004_0305	"	0x4004_0315	"	0x4004_0325	"	0x4004_0335	
0x4004_0306	<R0>	0x4004_0316	"	0x4004_0326	<R0>	0x4004_0336	
0x4004_0307	<R0>	0x4004_0317	"	0x4004_0327	<R0>	0x4004_0337	
0x4004_0308	CECRESET	0x4004_0318	CECRCR2	0x4004_0328	CECTCR	0x4004_0338	
0x4004_0309	<R0>	0x4004_0319	"	0x4004_0329	"	0x4004_0339	
0x4004_030A	<R0>	0x4004_031A	<R0>	0x4004_032A	"	0x4004_033A	
0x4004_030B	<R0>	0x4004_031B	<R0>	0x4004_032B	<R0>	0x4004_033B	
0x4004_030C	CECREN	0x4004_031C	CECRCR3	0x4004_032C	CECRSTAT	0x4004_033C	
0x4004_030D	<R0>	0x4004_031D	"	0x4004_032D	<R0>	0x4004_033D	
0x4004_030E	<R0>	0x4004_031E	"	0x4004_032E	<R0>	0x4004_033E	
0x4004_030F	<R0>	0x4004_031F	<R0>	0x4004_032F	<R0>	0x4004_033F	

[10] Remote control signal preprocessor (RMC)

<RMC0>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4004_0400	RMC0EN	0x4004_0410	RMC0RBUF3	0x4004_0420	RMC0RCR4	0x4004_0430	
0x4004_0401	<R0>	0x4004_0411	<R0>	0x4004_0421	<R0>	0x4004_0431	
0x4004_0402	<R0>	0x4004_0412	<R0>	0x4004_0422	<R0>	0x4004_0432	
0x4004_0403	<R0>	0x4004_0413	<R0>	0x4004_0423	<R0>	0x4004_0433	
0x4004_0404	RMC0REN	0x4004_0414	RMC0RCR1	0x4004_0424	RMC0RSTAT	0x4004_0434	
0x4004_0405	<R0>	0x4004_0415	"	0x4004_0425	"	0x4004_0435	
0x4004_0406	<R0>	0x4004_0416	"	0x4004_0426	<R0>	0x4004_0436	
0x4004_0407	<R0>	0x4004_0417	"	0x4004_0427	<R0>	0x4004_0437	
0x4004_0408	RMC0RBUF1	0x4004_0418	RMC0RCR2	0x4004_0428		0x4004_0438	
0x4004_0409	"	0x4004_0419	"	0x4004_0429		0x4004_0439	
0x4004_040A	"	0x4004_041A	"	0x4004_042A		0x4004_043A	
0x4004_040B	"	0x4004_041B	"	0x4004_042B		0x4004_043B	
0x4004_040C	RMC0RBUF2	0x4004_041C	RMC0RCR3	0x4004_042C		0x4004_043C	
0x4004_040D	"	0x4004_041D	"	0x4004_042D		0x4004_043D	
0x4004_040E	"	0x4004_041E	<R0>	0x4004_042E		0x4004_043E	
0x4004_040F	"	0x4004_041F	<R0>	0x4004_042F		0x4004_043F	

<RMC1>

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4004_0440	RMC1EN	0x4004_0450	RMC1RBUF3	0x4004_0460	RMC1RCR4	0x4004_0470	
0x4004_0441	<R0>	0x4004_0451	<R0>	0x4004_0461	<R0>	0x4004_0471	
0x4004_0442	<R0>	0x4004_0452	<R0>	0x4004_0462	<R0>	0x4004_0472	
0x4004_0443	<R0>	0x4004_0453	<R0>	0x4004_0463	<R0>	0x4004_0473	
0x4004_0444	RMC1REN	0x4004_0454	RMC1RCR1	0x4004_0464	RMC1RSTAT	0x4004_0474	
0x4004_0445	<R0>	0x4004_0455	"	0x4004_0465	"	0x4004_0475	
0x4004_0446	<R0>	0x4004_0456	"	0x4004_0466	<R0>	0x4004_0476	
0x4004_0447	<R0>	0x4004_0457	"	0x4004_0467	<R0>	0x4004_0477	
0x4004_0448	RMC1RBUF1	0x4004_0458	RMC1RCR2	0x4004_0468		0x4004_0478	
0x4004_0449	"	0x4004_0459	"	0x4004_0469		0x4004_0479	
0x4004_044A	"	0x4004_045A	"	0x4004_046A		0x4004_047A	
0x4004_044B	"	0x4004_045B	"	0x4004_046B		0x4004_047B	
0x4004_044C	RMC1RBUF2	0x4004_045C	RMC1RCR3	0x4004_046C		0x4004_047C	
0x4004_044D	"	0x4004_045D	"	0x4004_046D		0x4004_047D	
0x4004_044E	"	0x4004_045E	<R0>	0x4004_046E		0x4004_047E	
0x4004_044F	"	0x4004_045F	<R0>	0x4004_046F		0x4004_047F	

[11]Flash

ADR	Register name	ADR	Register name	ADR	Register name	ADR	Register name
0x4004_0500	SECBIT	0x4004_0510		0x4004_0520	FLCS	0x4004_0530	
0x4004_0501	"	0x4004_0511		0x4004_0521	"	0x4004_0531	
0x4004_0502	"	0x4004_0512		0x4004_0522	"	0x4004_0532	
0x4004_0503	"	0x4004_0513		0x4004_0523	"	0x4004_0533	
0x4004_0504	<reserved>	0x4004_0514		0x4004_0524	<reserved>	0x4004_0534	
0x4004_0505	<reserved>	0x4004_0515		0x4004_0525	<reserved>	0x4004_0535	
0x4004_0506	<reserved>	0x4004_0516		0x4004_0526	<reserved>	0x4004_0536	
0x4004_0507	<reserved>	0x4004_0517		0x4004_0527	<reserved>	0x4004_0537	
0x4004_0508		0x4004_0518		0x4004_0528	<reserved>	0x4004_0538	
0x4004_0509		0x4004_0519		0x4004_0529	<reserved>	0x4004_0539	
0x4004_050A		0x4004_051A		0x4004_052A	<reserved>	0x4004_053A	
0x4004_050B		0x4004_051B		0x4004_052B	<reserved>	0x4004_053B	
0x4004_050C		0x4004_051C		0x4004_052C		0x4004_053C	
0x4004_050D		0x4004_051D		0x4004_052D		0x4004_053D	
0x4004_050E		0x4004_051E		0x4004_052E		0x4004_053E	
0x4004_050F		0x4004_051F		0x4004_052F		0x4004_053F	

17. Flash Memory Operation

This section describes the hardware configuration and operation of the flash memory.

17.1 Flash Memory

17.1.1 Features

1) Memory capacity

The TMPM330FDFG device contains 4M bits (512 kB) of flash memory capacity. The memory area consists of 6 independent memory blocks (128 kB × 3, 64kB × 1 and 32kB × 2) to enable independent write access to each block. When the CPU is to access the internal flash memory, 32-bit data bus width is used.

2) Write/erase time

Write time: 1.28 sec/Chip (Typ) 0.32 sec/128Kbyte (Typ.)

Erase: 0.6 sec/Chip (Typ) 0.1msec/block (Typ.)

(Note) The above values are theoretical values not including data transfer time.

The write time per chip depends on the write method to be used by the user.

3) Programming method

The onboard programming mode is available for the user to program (rewrite) the device while it is mounted on the user's board.

- The onboard programming mode

3-1) User boot mode

The user's original rewriting method can be supported.

3-2) Single boot mode

The rewriting method to use serial data transfer (Toshiba's unique method) can be supported.

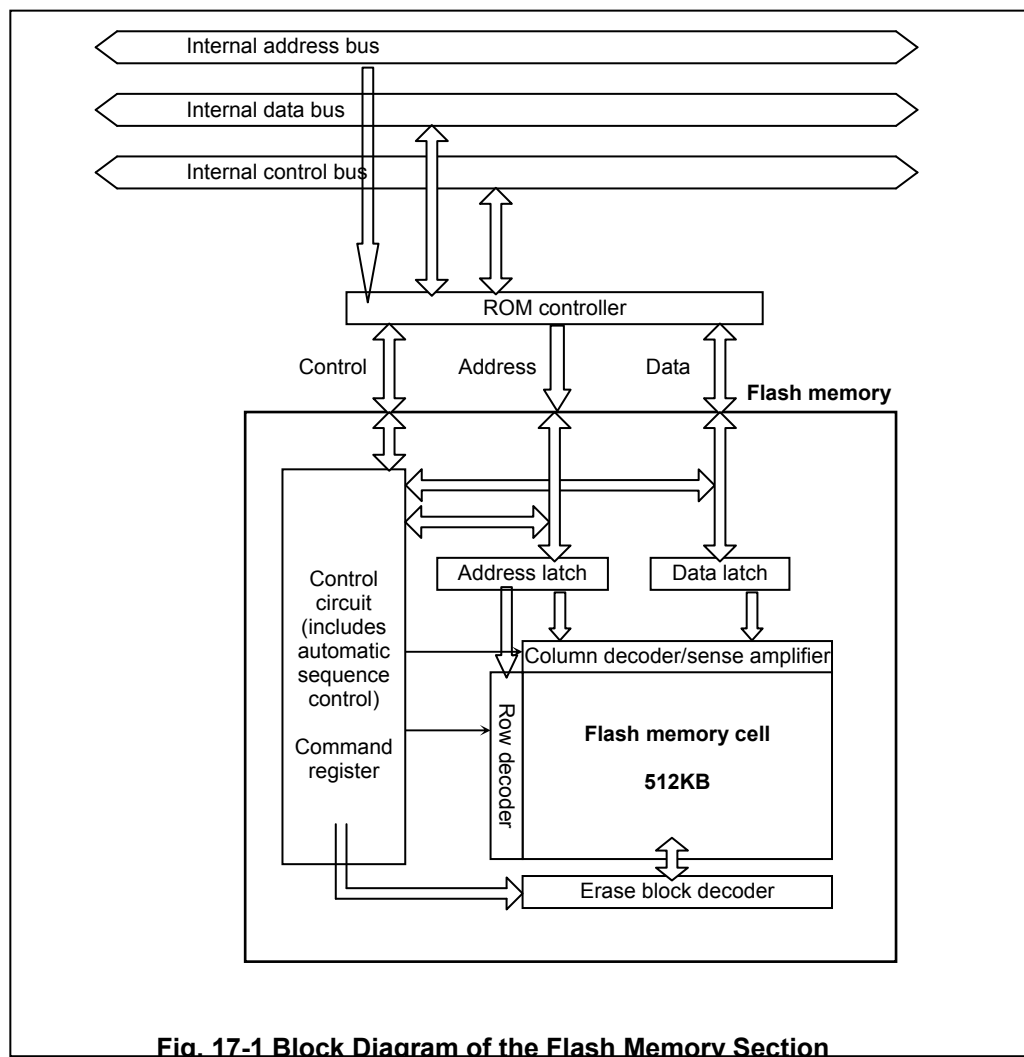
Rewriting method

The flash memory included in this device is generally compliant with the applicable JEDEC standards except for some specific functions. Therefore, if the user is currently using an external flash memory device, it is easy to implement the functions into this device. Furthermore, the user is not required to build his/her own programs to realize complicated write and erase functions because such functions are automatically performed using the circuits already built-in the flash memory chip.

This device is also implemented with a read-protect function to inhibit reading flash memory data from any external writer device. On the other hand, rewrite protection is available only through command-based software programming; any hardware setting method to apply +12VDC is not supported. See chapter 22 for details of ROM protection and security function.

JEDEC compliant functions	Modified, added, or deleted functions
• Automatic programming • Automatic chip erase • Automatic block erase • Data polling/toggle bit	<Modified> Block protect (only software protection is supported) <Deleted> Erase resume - suspend function

17.1.2 Block Diagram of the Flash Memory Section



17.2 Operation Mode

This device has three operation modes including the mode not to use the internal flash memory.

Table 17-1 Operation Modes

Operation mode	Operation details
Single chip mode	After reset is cleared, it starts up from the internal flash memory.
Normal mode	In this operation mode, two different modes, i.e., the mode to execute user application programs and the mode to rewrite the flash memory onboard the user's card, are defined. The former is referred to as "normal mode" and the latter "user boot mode."
User boot mode	The user can uniquely configure the system to switch between these two modes. For example, the user can freely design the system such that the normal mode is selected when the port "A0" is set to "1" and the user boot mode is selected when it is set to "0." The user should prepare a routine as part of the application program to make the decision on the selection of the modes.
Single boot mode	After reset is cleared, it starts up from the internal Boot ROM (Mask ROM). In the Boot ROM, an algorithm to enable flash memory rewriting on the user's set through the serial port of this device is programmed. By connecting to an external host computer through the serial port, the internal flash memory can be programmed by transferring data in accordance with predefined protocols.

Among the flash memory operation modes listed in the above table, the User Boot mode and the Single Boot mode are the programmable modes. These two modes, the User Boot mode and the Single Boot mode, are referred to as "Onboard Programming" modes where onboard rewriting of internal flash memory can be made on the user's card.

Either the Single Chip or Single Boot operation mode can be selected by externally setting the level of the $\overline{\text{BOOT}}$ (PH0) pin while the device is in reset status.

After the level is set, the CPU starts operation in the selected operation mode when the reset condition is removed. Regarding the $\overline{\text{BOOT}}$ (PH0) pin, be sure not to change the levels during operation once the mode is selected.

The mode setting method and the mode transition diagram are shown below:

Table 17-2 Operation Mode Setting

Operation mode	Pin	
	$\overline{\text{RESET}}$	$\overline{\text{BOOT}}$ (PH0)
Single chip mode	0 → 1	1
Single boot mode	0 → 1	0

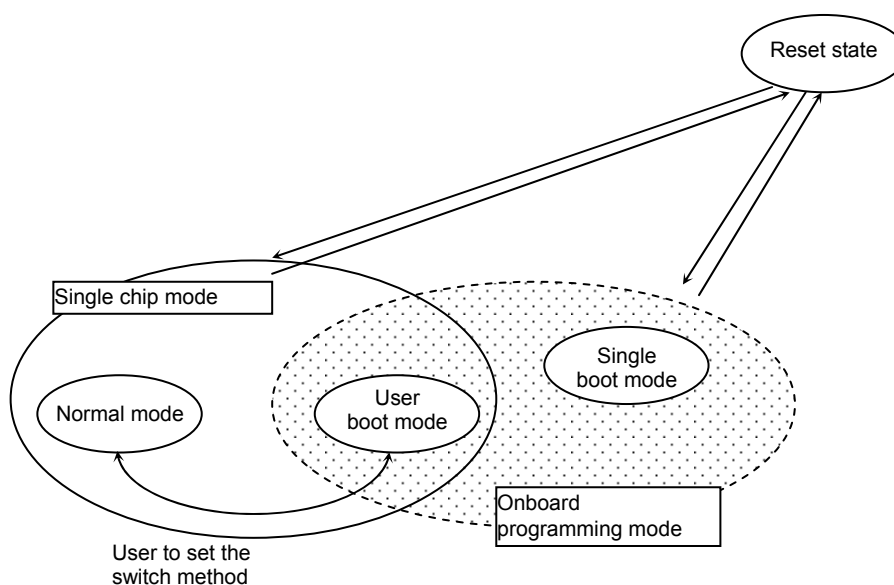


Fig. 17-2 Mode Transition Diagram

17.2.1 Reset Operation

To reset the device, ensure that the power supply voltage is within the operating voltage range, that the internal oscillator has been stabilized, and that the $\overline{\text{RESET}}$ input is held at "0" for a minimum duration of 12 system clocks (1.8 μs with 32MHz operation; the "1/8" clock gear mode is applied after reset).

(Note 1) Regarding power-on reset of devices with internal flash memory;
for devices with internal flash memory, it is necessary to apply "0" to the $\overline{\text{RESET}}$ inputs upon power on for a minimum duration of 500 microseconds regardless of the operating frequency.

(Note 2) While flash auto programming or deletion is in progress, at least 0.5 microseconds of reset period is required regardless of the system clock frequency.

17.2.2 User Boot Mode (ingle chip mode)

User Boot mode is to use flash memory programming routine defined by users. It is used when the data transfer buses for flash memory program code on the old application and for serial I/O are different. It operates at the single chip mode; therefore, a switch from normal mode in which user application is activated at the single chip mode to User Boot Mode for programming flash is required. Specifically, add a mode judgment routine to a reset program in the old application.

The condition to switch the modes needs to be set by using the I/O of M330 in conformity with the user's system setup condition. Also, flash memory programming routine that the user uniquely makes up needs to be set in the new application. This routine is used for programming after being switched to User Boot Mode. The execution of the programming routine must take place while it is stored in the area other than the flash memory since the data in the internal flash memory cannot be read out during delete/ writing mode. Once re-programming is complete, it is recommended to protect relevant flash blocks from accidental corruption during subsequent Single-Chip (Normal mode) operations. All the interruption including a non-maskable are inhibited at User Boot Mode.

(1-A) and (1-B) are the examples of programming with routines in the internal flash memory and in the external memory. For a detailed description of the erase and program sequence, refer to 17.3 On-board Programming of Flash Memory (Rewrite/Erase).

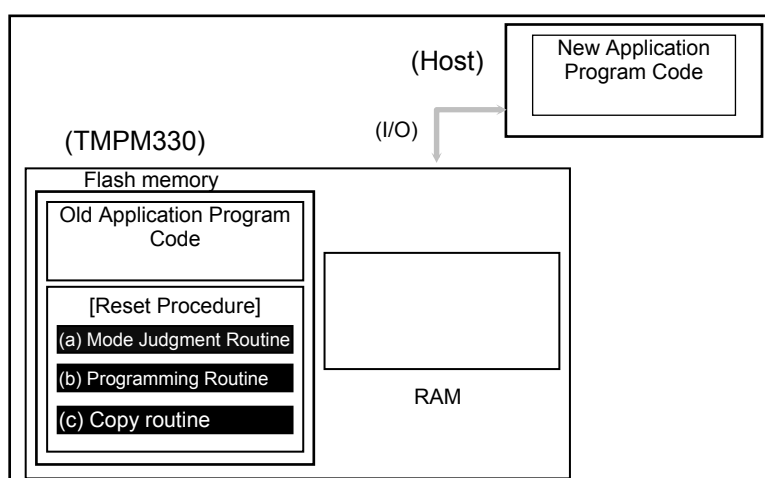
User Boot Mode

(1-A) Method 1: Storing a Programming Routine in the Flash Memory

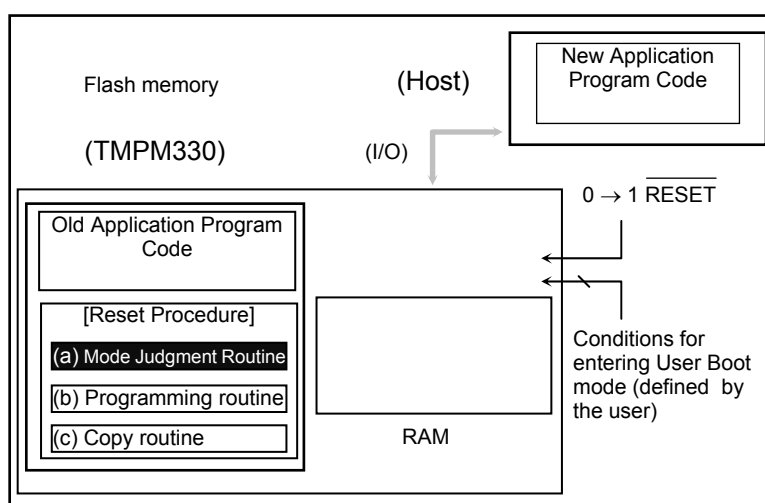
(Step-1)

Determine the conditions (e.g., pin states) required for the flash memory to enter User Boot mode and the I/O bus to be used to transfer new program code. Create hardware and software accordingly. Before installing the TMPM330 on a printed circuit board, write the following program routines into an arbitrary flash block using programming equipment.

- (a) Mode judgment routine: Code to determine whether or not to switch to User Boot mode
- (b) Programming routine: Code to download new program code from a host controller and re-program the flash memory
- (c) Copy routine: Code to copy the data described in (b) from the TMPM330 flash memory to either the TMP19A423 on-chip RAM or external memory device.

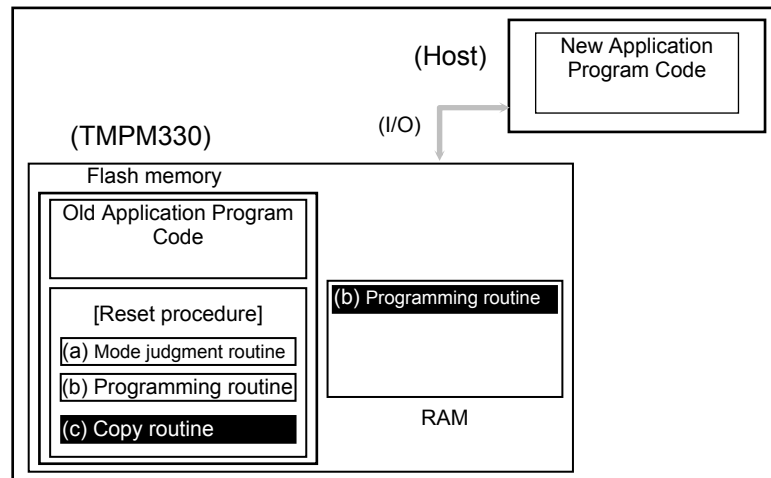
**(Step-2)**

After $\overline{\text{RESET}}$ is released, the reset procedure determines whether to put the TMPM330 flash memory in User Boot mode. If mode switching conditions are met, the flash memory enters User Boot mode. (All interrupts including NMI must be disabled while in User Boot mode.)

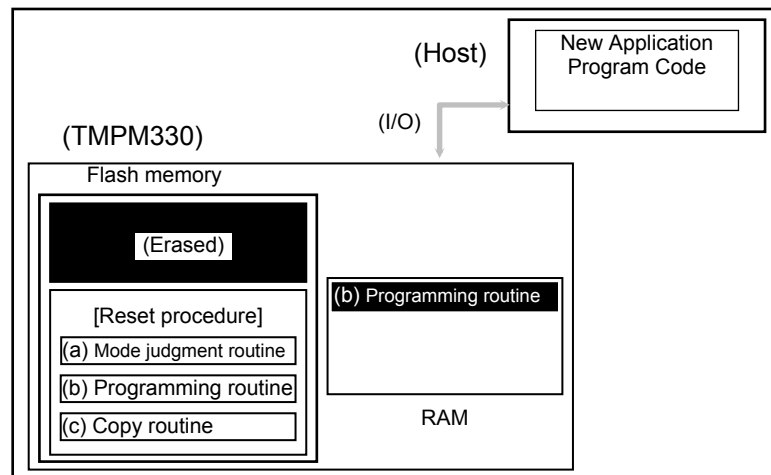


(Step-3)

Once transition to User Boot mode is occurred, execute the copy routine (c) to copy the flash programming routine (b) to the TMPM330 on-chip RAM.

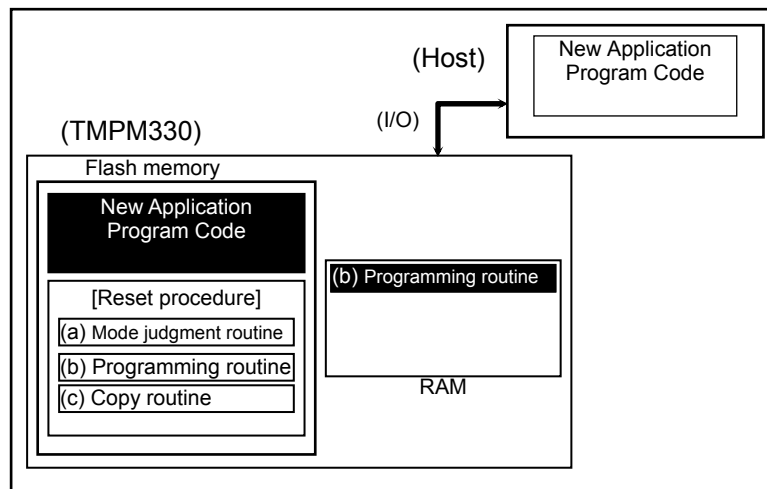
**(Step-4)**

Jump program execution to the flash programming routine in the on-chip RAM to erase a flash block containing the old application program code.

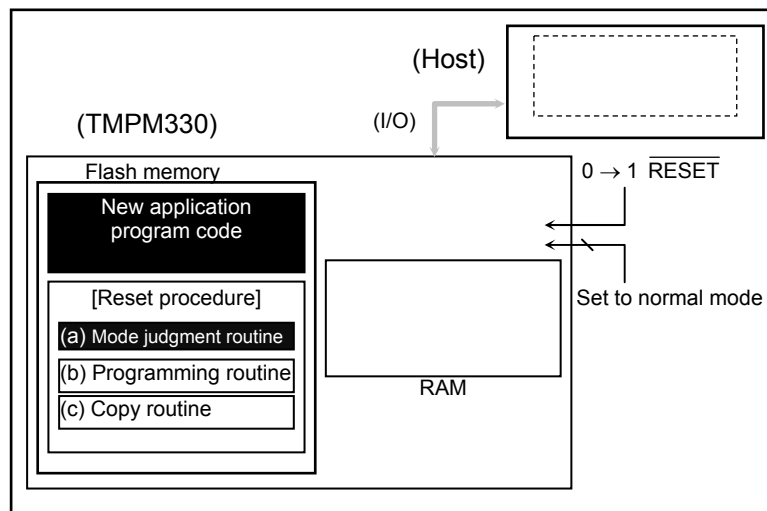


(Step-5)

Continue executing the flash programming routine to download new program code from the host controller and program it into the erased flash block. Once programming is complete, turn on the protection of that flash block.

**(Step-6)**

Set $\overline{\text{RESET}}$ to "0" to reset the TMPM330. Upon reset, the on-chip flash memory is put in Normal mode. After $\overline{\text{RESET}}$ is released, the CPU will start executing the new application program code.



(1-B) Method 2: Transferring a Programming Routine from an External Host

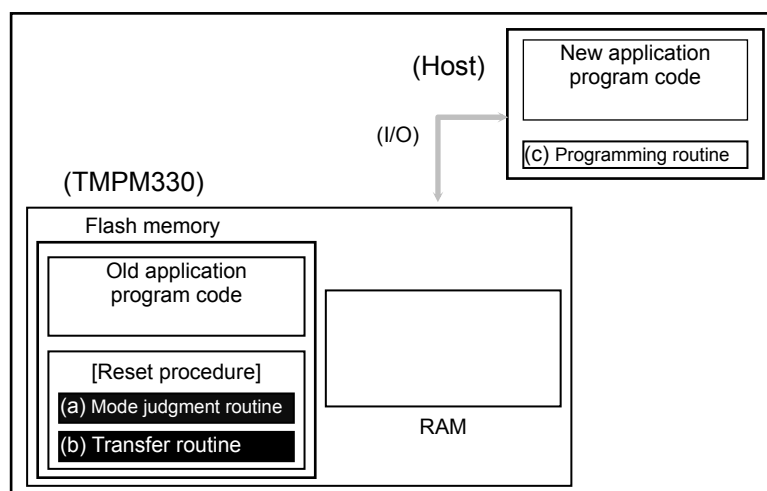
(Step-1)

Determine the conditions (e.g., pin states) required for the flash memory to enter User Boot mode and the I/O bus to be used to transfer new program code. Create hardware and software accordingly. Before installing the TMPM330 on a printed circuit board, write the following program routines into an arbitrary flash block using programming equipment.

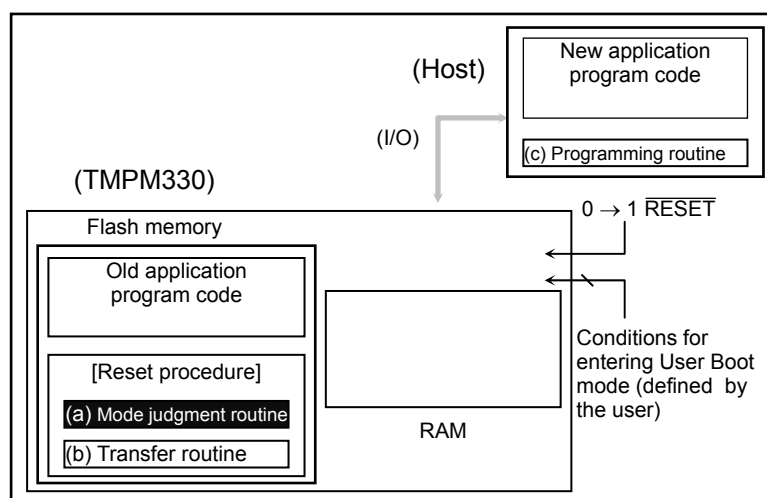
- (a) Mode judgment routine: Code to determine whether or not to switch to User Boot mode
- (b) Transfer routine: Code to download new program code from a host controller

Also, prepare a programming routine shown below on the host controller:

- (c) Programming routine: Code to download new program code from an external host controller and re-program the flash memory

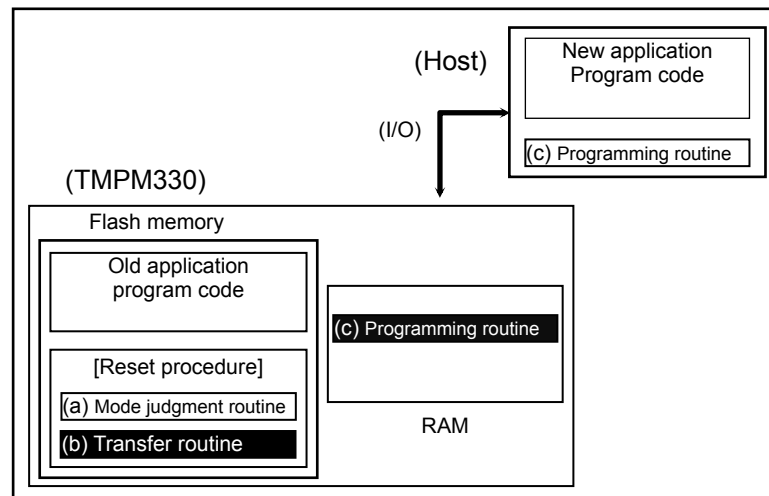
**(Step-2)**

After $\overline{\text{RESET}}$ is released, the reset procedure determines whether to put the TMPM330 flash memory in User Boot mode. If mode switching conditions are met, the flash memory enters User Boot mode. (All interrupts including NMI must be disabled while in User Boot mode).

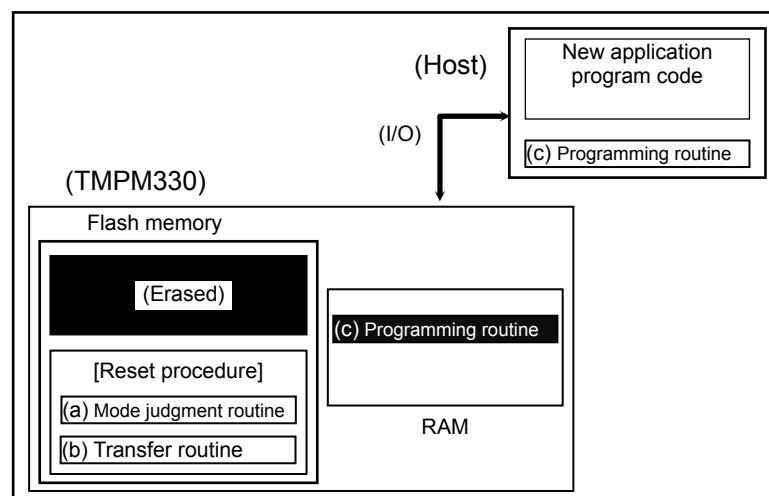


(Step-3)

Once User Boot mode is entered, execute the transfer routine (b) to download the flash programming routine (c) from the host controller to the TMPM330 on-chip RAM.

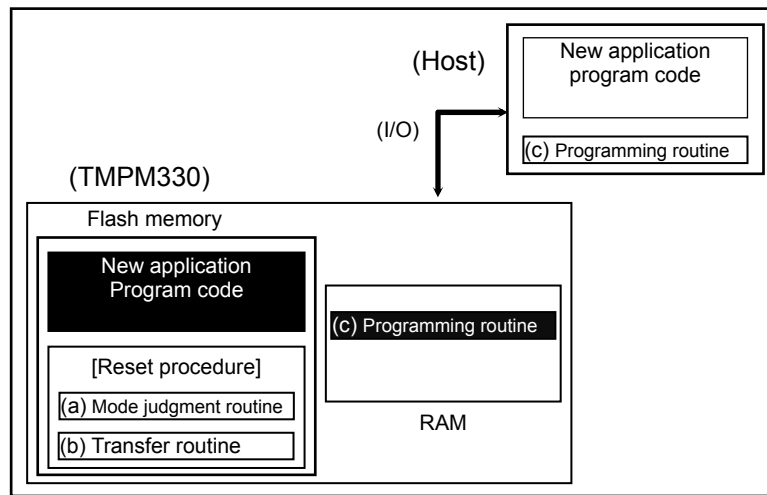
**(Step-4)**

Jump program execution to the flash programming routine in the on-chip RAM to erase a flash block containing the old application program code.

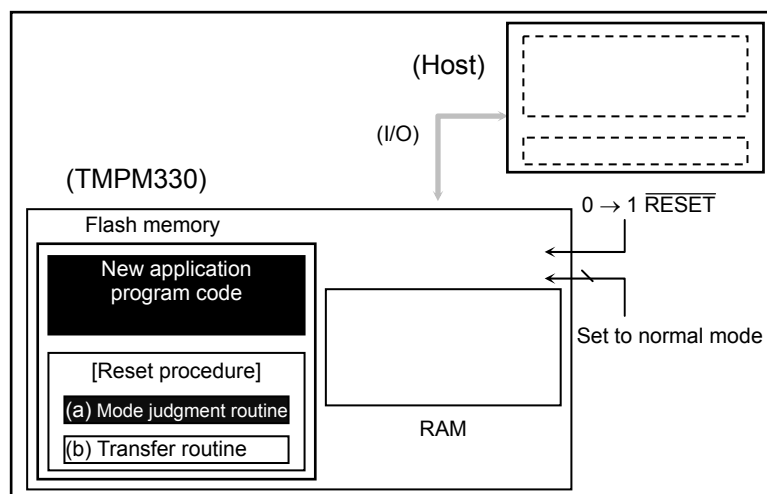


(Step-5)

Continue executing the flash programming routine to download new program code from the host controller and program it into the erased flash block. Once programming is complete, turn on the protection of that flash block.

**(Step-6)**

Set $\overline{\text{RESET}}$ to "0" low to reset the TMPM330. Upon reset, the on-chip flash memory is put in Normal mode. After $\overline{\text{RESET}}$ is released, the CPU will start executing the new application program code.



17.2.3 Single Boot Mode

In Single Boot mode, the flash memory can be re-programmed by using a program contained in the TPM330 on-chip boot ROM. This boot ROM is a masked ROM. When Single Boot mode is selected upon reset, the boot ROM is mapped to the address region including the interrupt vector table while the flash memory is mapped to an address region different from it.

Single Boot mode allows for serial programming of the flash memory. Channel 0 of the SIO (SIO0) of the TPM330 is connected to an external host controller. Via this serial link, a programming routine is downloaded from the host controller to the TPM330 on-chip RAM. Then, the flash memory is re-programmed by executing the programming routine. The host sends out both commands and programming data to re-program the flash memory.

Communications between the SIO0 and the host must follow the protocol described later. To secure the contents of the flash memory, the validity of the application's password is checked before a programming routine is downloaded into the on-chip RAM. If password matching fails, the transfer of a programming routine itself is aborted.

As in the case of User Boot mode, all interrupts including the non-maskable interrupt (NMI) must be disabled in Single Boot mode while the flash memory is being erased or programmed. In Single Boot mode, the boot-ROM programs are executed in Normal mode.

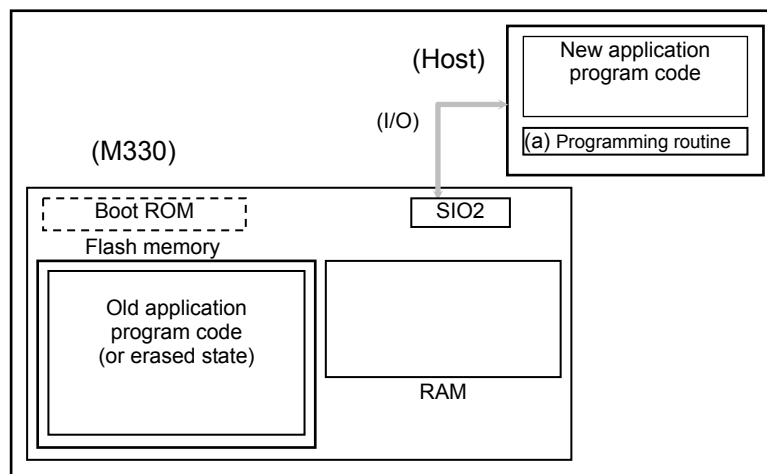
Once re-programming is complete, it is recommended to protect relevant flash blocks from accidental corruption during subsequent Single-Chip (Normal mode) operations.

Single Boot Mode

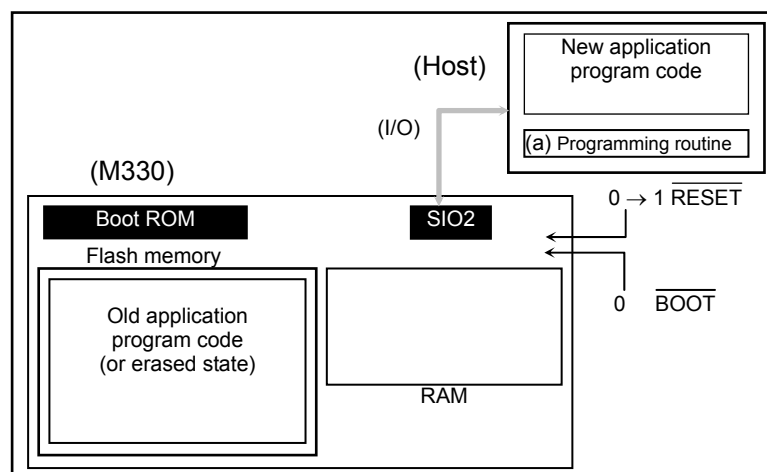
(2-A) Using the Program in the On-Chip Boot ROM

(Step-1)

The flash block containing the older version of the program code need not be erased before executing the programming routine. Since a programming routine and programming data are transferred via the SIO (SIO2), the SIO2 must be connected to a host controller. Prepare a programming routine (a) on the host controller.

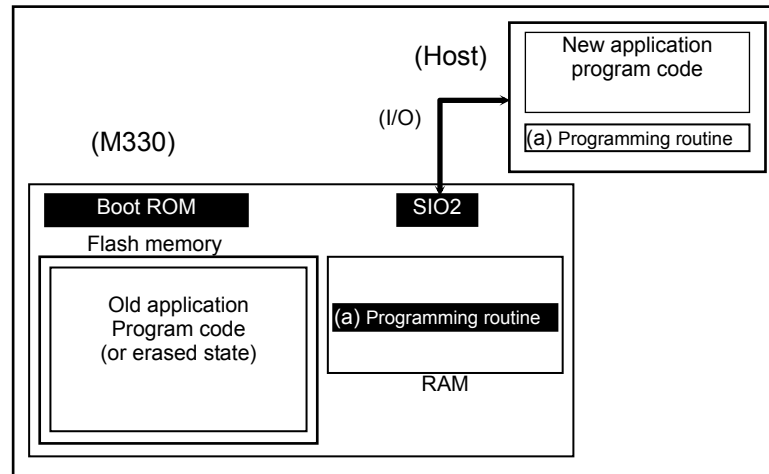
**(Step-2)**

Cancel the reset of the TMPM330 by setting the Single Boot mode pin to "0", so that the CPU re-boots from the on-chip boot ROM. The 12-byte password transferred from the host controller via SIO2 is first compared to the contents of the special flash memory locations. (If the flash block has already been erased, the password is 0xFFFF).

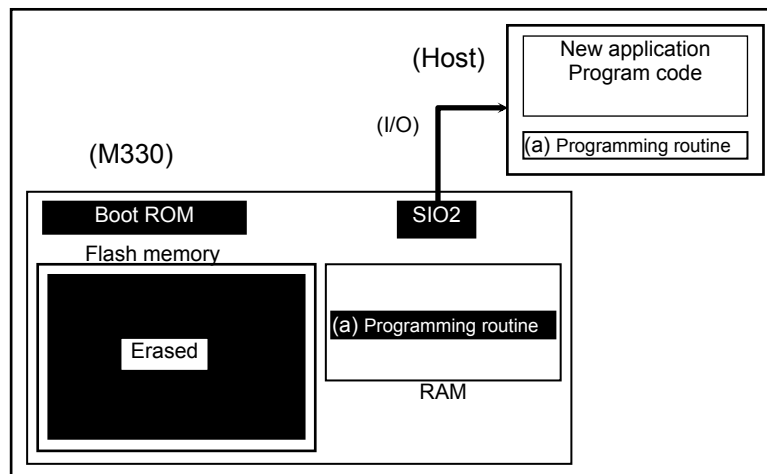


(Step-3)

If the password was correct, the boot program downloads, via the SIO0, the programming routine (a) from the host controller into the on-chip RAM of the TMPM330. The programming routine must be stored in the address range 0x2000_0400 - 0x2000_7FFF.

**(Step-4)**

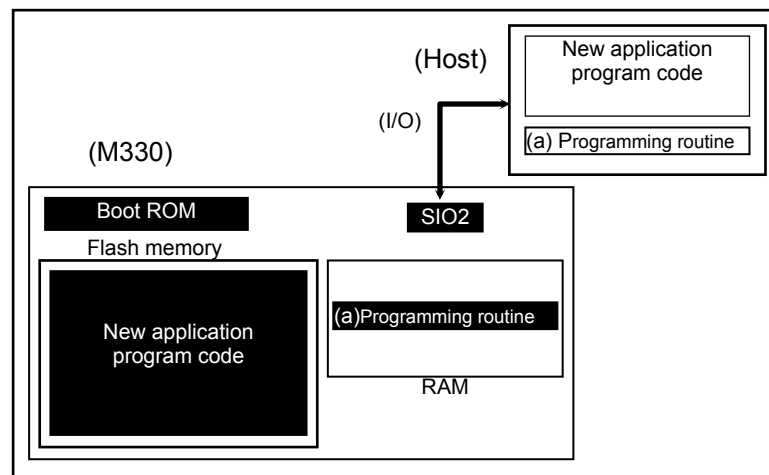
The CPU jumps to the programming routine (a) in the on-chip RAM to erase the flash block containing the old application program code. The Block Erase or Chip Erase command may be used.



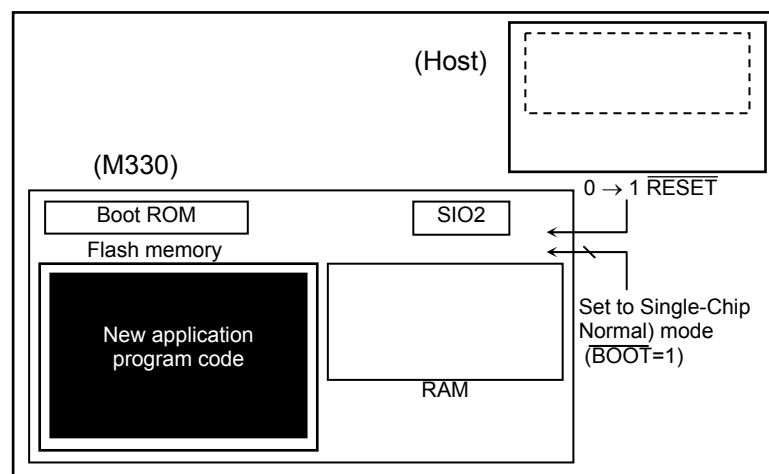
(Step-5)

Next, the programming routine (a) downloads new application program code from the host controller and programs it into the erased flash block. Once programming is complete, protection of that flash block is turned on. It is not allowed to move program control from the programming routine (a) back to the boot ROM.

In the example below, new program code comes from the same host controller via the same SIO0 channel as for the programming routine. However, once the programming routine has begun to execute, it is free to change the transfer path and the source of the transfer. Create board hardware and a programming routine to suit your particular needs.

**(Step-6)**

When programming of the flash memory is complete, power off the board and disconnect the cable leading from the host to the target board. Turn on the power again so that the TMPM330 re-boots in Single-Chip (Normal) mode to execute the new program.



(1) Configuration for Single Boot Mode

To execute the on-board programming, boot the TMPM330 with Single Boot mode following the configuration shown below.

$$\begin{aligned}\overline{\text{BOOT}} \text{ (PH0)} &= 0 \\ \overline{\text{RESET}} &= 0 \rightarrow 1\end{aligned}$$

Set the $\overline{\text{RESET}}$ input to 0, and set the each $\overline{\text{BOOT}}$ (PH0) pins to values shown above, and then release RESET (high).

(2) Memory Map

Fig. 17-3 shows a comparison of the memory maps in Normal and Single Boot modes. In Single Boot mode, the internal flash memory is mapped to 0x3F80_0000 through 0x3F87_FFFF, and the Internal boot ROM (Mask ROM) is mapped to 0x0000_0000 through 0x0000_1FFF.

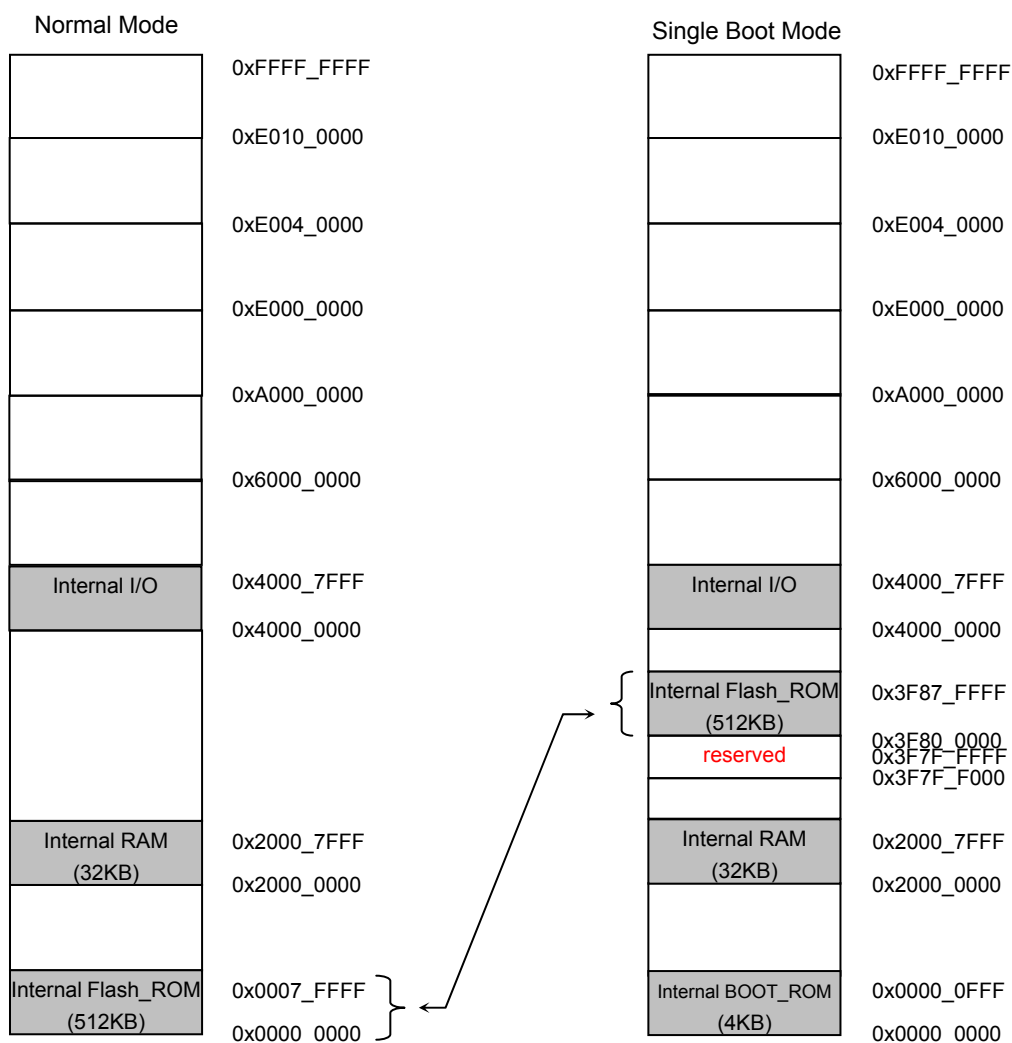


Fig. 17-3 Memory Maps for TMPM330FD

(3) Interface specification

In Single Boot mode, an SIO channel is used for communications with a programming controller. The same configuration is applied to a communication format on a programming controller to execute the on-board programming. Both UART (asynchronous) and I/O Interface (synchronous) modes are supported. The communication formats are shown below.

- UART communication

Communication channel	: SIO channel 0
Serial transfer mode	: UART (asynchronous), half -duplex, LSB fast
Data length	: 8 bit
Parity bits	: None
STOP bits	: 1 bit
Baud rate	: Arbitrary baud rate

- I/O interface mode

Communication channel	: SIO channel 0
Serial transfer mode	: I/O interface mode, full -duplex, LSB fast
Synchronization clock (SCLK0)	: Input mode
Handshaking signal	: PE4 configured as an output mode
Baud rate	: Arbitrary baud rate

Table 17-3 Required Pin Connections

Pins		Interface	
		UART	I/O Interface Mode
Power supply pins	VDD_regu	○	○
	VDD_AD	○	○
	VDD_IO_1,2	○	○
	CVCC	○	○
	VSS_regu	○	○
	VSS_AD	○	○
	VSS_IO_1,2	○	○
	CVSS	○	○
Mode-setting pin	BOOT (PH0)	○	○
Reset pin	RESET	○	○
Communication pins	TXD0(PE0)	○	○
	RXD0(PE1)	○	○
	SCLK0(PE2)	x	○ (Input mode)
	PE4	x	○ (Output mode)

(4) Data Transfer Format

Table 17-4 and Table 17-6 to Table 17-10 illustrate the operation commands and data transfer formats at each operation mode. In conjunction with this section, refer to (6) Operation of Boot Program.

Table 17-4 Single Boot Mode Commands

Code	Command
10H	RAM transfer
20H	Show Flash Memory SUM
30H	Show Product Information
40H	Chip erase

(5) Restrictions on internal memories

Single Boot Mode places restrictions on the internal RAM and ROM as shown in Table 17-5.

Table 17-5 Restrictions in Single Boot Mode

Memory	Details
Internal RAM	BOOT ROM is mapped to 0x2000_0000 to 0x2000_03FF. Store the RAM transfer program in 0x2000_0400~0x2000_7FFF.
Internal ROM	0x3F8F_FFF0 to 0x3F8F_FFFF are assigned for storing software ID information and passwords. Storing program in these addresses is not recommendable.

Table 17-6 Transfer Format for the RAM Transfer Command

	Byte	Data Transferred from the Controller to the TMPM330	Baud rate	Data Transferred from the TMPM330 to the Controller
Boot ROM	1 byte	Serial operation mode and baud rate For UART mode 86H For I/O Interface mode 30H	Desired baud rate (Note 1)	-
	2 byte	-		ACK for the serial operation mode byte For UART mode -Normal acknowledge 86H (The boot program aborts if the baud rate can not be set correctly.) For I/O Interface mode -Normal acknowledge 30H
	3 byte	Command code (10H)		-
	4 byte	-		ACK for the command code byte (Note 2) -Normal acknowledge 10H -Negative acknowledge x 1H -Communication error x 8H
	5 byte - 16 byte	Password sequence (12 bytes) (0x3F8F_FFF4~0x3F8F_FFFF)		-
	17 byte	Check SUM value for bytes 5 - 16		-
	18 byte	-		ACK for the checksum byte (Note 2) -Normal acknowledge 10H -Negative acknowledge x1H -Communication error x8H
	19 byte	RAM storage start address 31 - 24		-
	20 byte	RAM storage start address 23 - 16		-
	21 byte	RAM storage start address 15 - 8		-
	22 byte	RAM storage start address 7 - 0		-
	23 byte	RAM storage byte count 15 - 8		-
	24 byte	RAM storage byte count 7 - 0		-
	25 byte	Check SUM value for bytes 19 - 24		-
	26 byte	-		ACK for the checksum byte (Note 2) -Normal acknowledge 10H -Negative acknowledge x1H -Communication error x8H
	27 byte ~ m byte	RAM storage data		-
	m + byte	Checksum value for bytes 27 - m		-
	m + byte	-		ACK for the checksum byte (Note 2) -Normal acknowledge 10H -Negative acknowledge x1H -Communication error x8H
RAM	m + byte	-		Jump to RAM storage start address

(Note 1) In I/O Interface mode, the baud rate for the transfers of the first and second bytes must be 1/16 of the desired baud rate.

(Note 2) In case of any negative acknowledge, the boot program returns to a state in which it waits for a command code (3rd byte). In I/O Interface mode, if a communication error occurs, a negative acknowledge does not occur.

(Note 3) The 19th to 25th bytes must be within the RAM address range 0x2000_0400~0x2000_7FFF.

Table 17-7 Transfer Format for the Show Flash Memory Sum Command

	Byte	Data Transferred from the Controller to the TMPM330	Baud rate	Data Transferred from the TMPM330 to the Controller
Boot ROM	1 byte	Serial operation mode and baud rate For UART mode 86H For I/O Interface mode 30H	Desired baud rate (Note 1)	-
	2 byte	-		ACK for the serial operation mode byte For UART mode -Normal acknowledge 86H (The boot program aborts if the baud rate can not be set correctly.) For I/O Interface mode -Normal acknowledge 30H
	3 byte	Command code (20H)	-----	-
	4 byte	-		ACK for the command code byte (Note 2) -Normal acknowledge 20H -Negative acknowledge x1H -Communication error x8H
	5 byte	-		SUM (upper byte)
	6 byte	-		SUM (lower byte)
	7 byte	-		Checksum value for bytes 5 and 6
	8 byte	(Wait for the next command code.)		-

Note 1:In I/O Interface mode, the baud rate for the transfers of the first and second bytes must be 1/16 of the desired baud rate.

Note 2:In case of any negative acknowledge, the boot program returns to a state in which it waits for a command code (3rd byte). In I/O Interface mode, if a communication error occurs, a negative acknowledge does not occur.

Table 17-8 Transfer Format for the Show Product Information Command (1/2)

	Byte	Data Transferred from the Controller to the TMPM330	Baud rate	Data Transferred from the TMPM330 to the Controller
Boot ROM	1 byte	Serial operation mode and baud rate For UART mode 86H For I/O Interface mode 30H	Desired baud rate (Note 1)	-
	2 byte	-		ACK for the serial operation mode byte For UART mode -Normal acknowledge 86H (The boot program aborts if the baud rate can not be set correctly.) For I/O Interface mode -Normal acknowledge 30H
	3 byte	Command code (30H)		-
	4 byte	-		ACK for the command code byte (Note 2) -Normal acknowledge 30H -Negative acknowledge × 1H -Communication error × 8H
	5 byte	-		Flash memory data (at address 0x3F87_FF00)
	6 byte	-		Flash memory data (at address 0x3F87_FF01)
	7 byte	-		Flash memory data (at address 0x3F87_FF02)
	8 byte	-		Flash memory data (at address 0x3F87_FF03)
	9 byte - 20 byte	-		Product name (12-byte ASCII code) "TMPM330FD_" from the 9th byte
	21 byte - 24 byte	-		Password comparison start address (4 bytes) F4H, FFH, 87H and 3FH from the 21 st byte
	25 byte - 28 byte	-		RAM start address (4 bytes) 00H, 00H, 00H and 20H from the 25 th byte
	29 byte - 32 byte	-		Dummy data (4 bytes) 00H, 00H, 00H and 00H from the 29 th byte
	33 byte - 36 byte	-		RAM end address (4 bytes) FFH, 7FH, 00H and 20H from the 33 rd byte
	37 byte - 40 byte	-		Dummy data (4 bytes) 00H, 00H, 00H and 00H from the 37 th byte.
	41 byte - 44 byte	-		Dummy data (4 bytes) 00H, 00H, 00H and 00H from the 41 st byte
	45 byte - 46 byte	-		Fuse information (2 bytes) 00H and 00H from the 45 th byte.
	47 byte - 50 byte	-		Flash memory start address (4 bytes) 00H, 00H, 80H and 3FH from the 47 th byte
	51 byte - 54 byte	-		Flash memory end address (4 bytes) FFH, FFH, 87H and 3FH from the 51 st byte
	55 byte - 56 byte	-		Flash memory block count (2 bytes) 06H and 00H from the 55 th byte
	57 byte - 60 byte	-		Start address of a group of the same-size (16K) flash blocks (4 bytes) 00H, 00H, 00H and 00H from 57 th byte

Table 17-9 Transfer Format for the Show Product Information Command (2/2)

	Byte	Data Transferred from the Controller to the TMPM330	Baud rate	Data Transferred from the TMPM330 to the Controller
Boot ROM	61 byte - 64 byte	-		Size (in halfwords) of the same-size (16K) flash blocks (4 bytes) 00H, 20H, 00H and 00H from the 61 st byte
	65 byte	-		Number of flash blocks of the same size (1 byte) 02H
	66 byte - 69 byte	-		Start address of a group of the same-size (32K) flash blocks (4 bytes) 00H, 00H, 80H and 3FH from 66 th byte
	70 byte - 73 byte			Size (in halfwords) of the same-size (32K) flash blocks (4 bytes) 00H, 40H, 00H and 00H from the 70 th byte
	74 byte			Number of flash blocks of the same size (32K) (1 byte) 02H
	75 byte - 78 byte			Start address of a group of the same-size (64K) flash blocks (4 bytes) 00H, 00H, 81H and 3FH from 75 th byte
	79 byte - 82 byte			Size (in halfwords) of the same-size (64K) flash blocks (4 bytes) 00H, 80H, 00H and 00H from the 79 th byte
	83 byte			Number of flash blocks of the same size (64K) (1 byte) 01H
	84 byte - 87 byte			Start address of a group of the same-size (128K) flash blocks (4 bytes) 00H, 00H, 01H and 00H from 84 th byte
	88 byte - 91 byte			Size (in halfwords) of the same-size (128K) flash blocks (4 bytes) 00H, 00H, 01H and 00H from the 88 th byte
	92 byte			Number of flash blocks of the same size (128K) (1 byte) 03H
	93 byte			Checksum value for bytes 5 - 92
	94 byte	(Wait for the next command code.)		-

Note 1:In I/O Interface mode, the baud rate for the transfers of the first and second bytes must be 1/16 of the desired baud rate.

Note 2:In case of any negative acknowledge, the boot program returns to a state in which it waits for a command code (3rd byte). In I/O Interface mode, if a communication error occurs, a negative acknowledge does not occur.

Table 17-10 Transfer Format for the Chip Erase Command

	Byte	Data Transferred from the Controller to the TMPM330	Baud rate	Data Transferred from the TMPM330 to the Controller
Boot ROM	1 byte	Serial operation mode and baud rate For UART mode 86H For I/O Interface mode 30H	Desired baud rate (Note 1)	—
	2 byte	—		ACK for the serial operation mode byte For UART mode -Normal acknowledge 86H For I/O Interface mode -Normal acknowledge 30H (The boot program aborts if the baud rate can not be set correctly.)
	3 byte	Command code (40H)		—
	4 byte	—		ACK for the command code byte (Note 2) -Normal acknowledge 40H -Negative acknowledge × 1H -Communication error × 8H
	5 byte	Chip erase command code (54H)		—
	6 byte	—		ACK for the command code byte (Note 2) -Normal acknowledge 54H -Negative acknowledge × 1H -Communication error × 8H
	7 byte	—		ACK for the chip erase command code byte -Normal acknowledge 4FH -Negative acknowledge 4CH
	8 byte	(Wait for the next command code.)		—

Note 1: In I/O Interface mode, the baud rate for the transfers of the first and second bytes must be 1/16 of the desired baud rate.

Note 2: In case of any negative acknowledge, the boot program returns to a state in which it waits for a command code (3rd byte). In I/O Interface mode, if a communication error occurs, a negative acknowledge does not occur.

(6) Operation of Boot Program

When Single Boot mode is selected, the boot program is automatically executed on startup. The boot program offers these three commands, of which the details are provided on the following subsections. The addresses described in this section are the virtual unless otherwise noted.

1. RAM Transfer command

The RAM Transfer command stores program code transferred from a host controller to the on-chip RAM and executes the program once the transfer is successfully completed. The maximum program size is 31 kbytes. The RAM storage start address must be set within the range.

The RAM Transfer command can be used to download a flash programming routine of your own; this provides the ability to control on-board programming of the flash memory in a unique manner. The programming routine must utilize the flash memory command sequences described in Section 17.3.

Before initiating a transfer, the RAM Transfer command checks a password sequence coming from the controller against that stored in the flash memory.

2. Show Flash Memory Sum command

The Show Flash Memory Sum command adds the contents of the 512 kbytes of the flash memory together. The boot program does not provide a command to read out the contents of the flash memory. Instead, the Flash Memory Sum command can be used for software revision management.

3. Show Product Information command

The Show Product Information command provides the product name, on-chip memory configuration and the like. This command also reads out the contents of the flash memory locations at addresses 0x3F87_FFF0 through 0x3F87_FFF3. In addition to the Show Flash Memory Sum command, these locations can be used for software revision management.

1) RAM Transfer Command (See Table 17-6)

1. The 1st byte specifies which one of the two serial operation modes is used. For a detailed description of how the serial operation mode is determined, see Determination of a Serial Operation Mode described later. If it is determined as UART mode, the boot program then checks if the SIO0 is programmable to the baud rate at which the 1st byte was transferred. During the first-byte interval, the RXE bit in the HSC0MOD register is cleared.

- To communicate in UART mode
Send, from the controller to the target board, 86H in UART data format at the desired baud rate. If the serial operation mode is determined as UART, then the boot program checks if the SIO0 can be programmed to the baud rate at which the first byte was transferred. If that baud rate is not possible, the boot program aborts, disabling any subsequent communications.
- To communicate in I/O Interface mode
Send, from the controller to the target board, 30H in I/O Interface data format at 1/16 of the desired baud rate. Also send the 2nd byte at the same baud rate. Then send all subsequent bytes at a rate equal to the desired baud rate.

In I/O Interface mode, the CPU sees the serial receive pin as if it were a general input port in monitoring its logic transitions. If the baud rate of the incoming data is high or the chip's operating frequency is high, the CPU may not be able to keep up with the speed of logic transitions. To prevent such situations, the 1st and 2nd bytes must be transferred at 1/16 of the desired baud rate; then the boot program calculates 16 times that as the desired baud rate. When the serial operation mode is determined as I/O Interface mode, the SIO0 is configured for SCLK Input mode. Beginning with the third byte, the controller must ensure that its AC timing restrictions are satisfied at the selected baud rate. In the case of I/O Interface mode, the boot program does not check the receive error flag; thus there is no such thing as error acknowledge (bit 3, x8H).

2. The 2nd byte, transmitted from the target board to the controller, is an acknowledge response to the 1st byte. The boot program echoes back the first byte: 86H for UART mode and 30H for I/O Interface mode.

UART mode

If the SIO0 can be programmed to the baud rate at which the 1st byte was transferred, the boot program programs the SC0BRCR and sends back 86H to the controller as an acknowledge. If the SIO0 is not programmable at that baud rate, the boot program simply aborts with no error indication. Following the 1st byte, the controller should allow for a time-out period of five seconds. If it does not receive 86H within the allowed time-out period, the controller should give up the communication. The boot program sets the RXE bit in the SC0MOD0 register to enable reception (1) before loading the SIO transmit buffer with 86H.

- I/O Interface mode
The boot program programs the SC0MOD0 and SC0CR registers to configure the SIO0 in I/O Interface mode (clocked by the rising edge of SCLK0), writes 30H to

the SC0BUF. Then, the SIO0 waits for the SCLK0 signal to come from the controller. Following the transmission of the 1st byte, the controller should send the SCLK clock to the target board after a certain idle time (several microseconds). This must be done at 1/16 the desired baud rate. If the 2nd byte, which is from the target board to the controller, is 30H, then the controller should take it as a go-ahead. The controller must then deliver the 3rd byte to the target board at a rate equal to the desired baud rate. The boot program sets the RXE bit in the SC0MOD register to enable reception before loading the SIO transmit buffer with 30H.

3. The 3rd byte transmitted from the controller to the target board is a command. The code for the RAM Transfer command is 10H.
4. The 4th byte, transmitted from the target board to the controller, is an acknowledge response to the 3rd byte. Before sending back the acknowledge response, the boot program checks for a receive error. If there was a receive error, the boot program transmits x8H (bit 3) and returns to the state in which it waits for a command (the third byte) again. In this case, the upper four bits of the acknowledge response are undefined - they hold the same values as the upper four bits of the previously issued command. When the SIO0 is configured for I/O Interface mode, the boot program does not check for a receive error.

If the 3rd byte is equal to any of the command codes listed in Table 17-4, the boot program echoes it back to the controller. When the RAM Transfer command was received, the boot program echoes back a value of 10H and then branches to the RAM Transfer routine. Once this branch is taken, a password check is done. Password checking is detailed in a later section "Password". If the 3rd byte is not a valid command, the boot program sends back x1H (bit 0) to the controller and returns to the state in which it waits for a command (the third byte) again. In this case, the upper four bits of the acknowledge response are undefined - they hold the same values as the upper four bits of the previously issued command.

5. The 5th to 16th bytes transmitted from the controller to the target board, are a 12-byte password. The 5th byte is compared to the contents of address 0x3F87_FFF4 in the flash memory; the 6th byte is compared to the contents of address 0x3F87_FFF5 in the flash memory; likewise, the 16th byte is compared to the contents of address 0x3F87_FFFF in the flash memory. If the password checking fails, the RAM Transfer routine sets the password error flag.
6. The 17th byte is a checksum value for the password sequence (5th to 16th bytes). To calculate the checksum value for the 12-byte password, add the 12 bytes together, drop the carries and take the two's complement of the total sum. Transmit this checksum value from the controller to the target board. The checksum calculation is described in details in a later section "Checksum Calculation".
7. The 18th byte, transmitted from the target board to the controller, is an acknowledge response to the 5th to 17th bytes. First, the RAM Transfer routine checks for a receive error in the 5th to 17th bytes. If there was a receive error, the boot program sends back 18H (bit 3) and returns to the state in which it waits for a command (i.e., the 3rd byte) again. In this case, the upper four bits of the acknowledge response are the same as

those of the previously issued command (i.e., all 1s). When the SIO0 is configured for I/O Interface mode, the RAM Transfer routine does not check for a receive error.

Next, the RAM Transfer routine performs the checksum operation to ensure data integrity. Adding the series of the 5th to 16th bytes must result in 00H (with the carry dropped). If it is not 00H, one or more bytes of data has been corrupted. In case of a checksum error, the RAM Transfer routine sends back 11H to the controller and returns to the state in which it waits for a command (i.e., the 3rd byte) again.

Finally, the RAM Transfer routine examines the result of the password check. The following two cases are treated as a password error. In these cases, the RAM Transfer routine sends back 11H (bit 0) to the controller and returns to the state in which it waits for a command (i.e., the 3rd byte) again.

- Irrespective of the result of the password comparison, all the 12 bytes of a password in the flash memory are the same value other than FFH. Not the entire password bytes transmitted from the controller matched those contained in the flash memory.

When all the above checks have been successful, the RAM Transfer routine returns a normal acknowledge response (10H) to the controller.

8. The 19th to 22nd bytes, transmitted from the controller to the target board, indicate the start address of the RAM region where subsequent data (e.g., a flash programming routine) should be stored. The 19th byte corresponds to bits 31–24 of the address and the 22nd byte corresponds to bits 7–0 of the address.
9. The 23rd and 24th bytes, transmitted from the controller to the target board, indicate the number of bytes that will be transferred from the controller to be stored in the RAM. The 23rd byte corresponds to bits 15–8 of the number of bytes to be transferred, and the 24th byte corresponds to bits 7–0 of the number of bytes.
10. The 25th byte is a checksum value for the 19th to 24th bytes. To calculate the checksum value, add all these bytes together, drop the carries and take the two's complement of the total sum. Transmit this checksum value from the controller to the target board. The checksum calculation is described in details in a later section "Checksum Calculation".
11. The 26th byte, transmitted from the target board to the controller, is an acknowledge response to the 19th to 25th bytes of data. First, the RAM Transfer routine checks for a receive error in the 19th to 25th bytes. If there was a receive error, the RAM Transfer routine sends back 18H and returns to the command wait state (i.e., the 3rd byte) again. In this case, the upper four bits of the acknowledge response are the same as those of the previously issued command (i.e., all 1s). When the SIO0 is configured for I/O Interface mode, the RAM Transfer routine does not check for a receive error.

Next, the RAM Transfer routine performs the checksum operation to ensure data integrity. Adding the series of the 19th to 25th bytes must result in 00H (with the carry dropped). If it is not 00H, one or more bytes of data has been corrupted. In case of a checksum error, the RAM Transfer routine sends back 11H to the controller and

returns to the state in which it waits for a command (i.e., the 3rd byte) again.

- The RAM storage start address must be within the range 0x2000_04000xFFFD_8000(0xFFFD0x2000_EFFF7FFF.

When the above checks have been successful, the RAM Transfer routine returns a normal acknowledge response (10H) to the controller.

12. The 27th to mth bytes from the controller are stored in the on-chip RAM of the TMPM330. Storage begins at the address specified by the 19th–22nd bytes and continues for the number of bytes specified by the 23rd–24th bytes.
13. The (m+1) th byte is a checksum value. To calculate the checksum value, add the 27th to mth bytes together, drop the carries and take the two's complement of the total sum. Transmit this checksum value from the controller to the target board. The checksum calculation is described in details in a later section "Checksum Calculation".
14. The (m+2) th byte is a acknowledge response to the 27th to (m+1) th bytes.

First, the RAM Transfer routine checks for a receive error in the 27th to (m+1) th bytes. If there was a receive error, the RAM Transfer routine sends back 18H (bit 3) and returns to the state in which it waits for a command (i.e., the 3rd byte) again. In this case, the upper four bits of the acknowledge response are the same as those of the previously issued command (i.e., all 1s). When the SIO0 is configured for I/O Interface mode, the RAM Transfer routine does not check for a receive error.

Next, the RAM Transfer routine performs the checksum operation to ensure data integrity. Adding the series of the 27th to (m+1) th bytes must result in 00H (with the carry dropped). If it is not 00H, one or more bytes of data has been corrupted. In case of a checksum error, the RAM Transfer routine sends back 11H (bit 0) to the controller and returns to the command wait state (i.e., the 3rd byte) again. When the above checks have been successful, the RAM Transfer routine returns a normal acknowledge response (10H) to the controller.

15. If the (m+2) th byte was a normal acknowledge response, a branch is made to the address specified by the 19th to 22nd bytes.

2) Show Flash Memory Sum Command (See Table 17-7)

1. The processing of the 1st and 2nd bytes are the same as for the RAM Transfer command.
2. The 3rd byte, which the target board receives from the controller, is a command. The code for the Show Flash Memory Sum command is 20H.
3. The 4th byte, transmitted from the target board to the controller, is an acknowledge response to the 3rd byte. Before sending back the acknowledge response, the boot program checks for a receive error. If there was a receive error, the boot program transmits x8H (bit 3) and returns to the command wait state again. In this case, the upper four bits of the acknowledge response are undefined - they hold the same values as the upper four bits of the previously issued command. When the SIO0 is configured for I/O Interface mode, the boot program does not check for a receive error.

If the 3rd byte is equal to any of the command codes listed in Table 17-4, the boot program echoes it back to the controller. When the Show Flash Memory Sum command was received, the boot program echoes back a value of 20H and then branches to the Show Flash Memory Sum routine. If the 3rd byte is not a valid command, the boot program sends back x1H (bit 0) to the controller and returns to the command wait state (the third byte) again. In this case, the upper four bits of the acknowledge response are undefined - they hold the same values as the upper four bits of the previously issued command.

4. The Show Flash Memory Sum routine adds all the bytes of the flash memory together. The 5th and 6th bytes, transmitted from the target board to the controller, indicate the upper and lower bytes of this total sum, respectively. For details on sum calculation, see a later section "Calculation of the Show Flash Memory Sum Command".
5. The 7th byte is a checksum value for the 5th and 6th bytes. To calculate the checksum value, add the 5th and 6th bytes together, drop the carry and take the two's complement of the sum. Transmit this checksum value from the controller to the target board.
6. The 8th byte is the next command code.

3) Show Product Information Command (See Table 17-8)

1. The processing of the 1st and 2nd bytes are the same as for the RAM Transfer command.
2. The 3rd byte, which the target board receives from the controller, is a command. The code for the Show Product Information command is 30H.
3. The 4th byte, transmitted from the target board to the controller, is an acknowledge response to the 3rd byte. Before sending back the acknowledge response, the boot program checks for a receive error. If there was a receive error, the boot program transmits x8H (bit 3) and returns to the command wait state again. In this case, the upper four bits of the acknowledge response are undefined - they hold the same values as the upper four bits of the previously issued command. When the SIO0 is configured for I/O Interface mode, the boot program does not check for a receive error.

If the 3rd byte is equal to any of the command codes listed in Table 17-4, the boot program echoes it back to the controller. When the Show Flash Memory Sum command was received, the boot program echoes back a value of 30H and then branches to the Show Flash Memory Sum routine. If the 3rd byte is not a valid command, the boot program sends back x1H (bit 0) to the controller and returns to the state in which it waits for a command (the third byte) again. In this case, the upper four bits of the acknowledge response are undefined - they hold the same values as the upper four bits of the previously issued command.

4. The 5th to 8th bytes, transmitted from the target board to the controller, are the data read from addresses 0x3F87_FFF0-0x3F87_FFF3 in the flash memory. Software version management is possible by storing a software ID in these locations.
5. The 9th to 20th bytes, transmitted from the target board to the controller, indicate the product name, which is "TMPM330FD_ _" in ASCII code (where _ is a space).
6. The 21st to 24th bytes, transmitted from the target board to the controller, indicate the start address of the flash memory area containing the password, i.e., F4H, FFH, 87H, and 3FH.
7. The 25th to 28th bytes, transmitted from the target board to the controller, indicate the start address of the on-chip RAM, i.e., 00H, 00H, FDH, FFH.
8. The 29th to 32nd bytes, transmitted from the target board to the controller, are dummy data (00H, 00H, 00H and 00H).
9. The 33rd to 36th bytes, transmitted from the target board to the controller, indicate the end address of the on-chip RAM, are FFH, 7FH, 00H, 20H.
10. The 37th to 40th bytes, transmitted from the target board to the controller, are 00H, 00H, 00H and 00H. The 41st to 44th bytes, transmitted from the target board to the controller, are FFH, EFH, FDH and FFH.

11. The 45th and 46th bytes transmitted are 01H, 00H.
12. The 47th to 50th bytes, transmitted from the target board to the controller, indicate the start address of the on-chip flash memory, are 00H, 00H, 80H, and 3FH.
13. The 51st to 54th bytes, transmitted from the target board to the controller, indicate the end address of the on-chip flash memory, i.e., FFH, FFH, 87H, and 3FH.
14. The 55th to 56th bytes, transmitted from the target board to the controller, indicate the number of flash blocks available, are 06H, 00H.
15. The 57th to 83rd bytes, transmitted from the target board to the controller, contain information about the flash blocks. Flash blocks of the same size are treated as a group. Information about the flash blocks indicate the start address of a group, the size of the blocks in that group (in halfwords) and the number of the blocks in that group.

The 57th to 65th bytes are the information about the 16-kbyte blocks. The 66th to 74th bytes are the information about the 32-kbyte blocks. The 75th to 83rd bytes are the information about the 64-kbyte blocks. The 84th to 92nd bytes are the information about the 128-kbyte blocks. See Table 17-9 for the values of bytes transmitted.
16. The 66th byte, transmitted from the target board to the controller, is a checksum value for the 5th to 92nd bytes. The checksum value is calculated by adding all these bytes together, dropping the carry and taking the two's complement of the total sum.
17. The 94th byte is the next command code.

4) Chip Erase command (See Table 17-9)

1. The processing of the 1st and 2nd bytes are the same as for the RAM Transfer command.
2. The 3rd byte, which the target board receives from the controller, is a command. The code for the Show Product Information command is 30H.

3. The 4th byte, transmitted from the target board to the controller, is an acknowledge response to the 3rd byte. Before sending back the acknowledge response, the boot program checks for a receive error. If there was a receive error, the boot program transmits x8H (bit 3) and returns to the command wait state again. In this case, the upper four bits of the acknowledge response are undefined - they hold the same values as the upper four bits of the previously issued command.

If the 3rd byte is equal to any of the command codes listed in Table 17-6, the boot program echoes it back to the controller. When the Show Flash Memory Sum command was received, the boot program echoes back a value of 40H. If the 3rd byte is not a valid command, the boot program sends back x1H (bit 0) to the controller and returns to the state in which it waits for a command (the third byte) again. In this case, the upper four bits of the acknowledge response are undefined - they hold the same values as the upper four bits of the previously issued command.

4. The 5th byte, transmitted from the target board to the controller, is the Chip Erase Enable command code (54H).
5. The 6th byte, transmitted from the target board to the controller, is an acknowledge response to the 5th byte.

Before sending back the acknowledge response, the boot program checks for a receive error. If there was a receive error, the boot program transmits x8H (bit 3) and returns to the command wait state again. In this case, the upper four bits of the acknowledge response are undefined - they hold the same values as the upper four bits of the previously issued command.

If the 5th byte is equal to any of the command codes to enable erasing, the boot program echoes it back to the controller. When the Show Flash Memory Sum command was received, the boot program echoes back a value of 54H and then branches to the Chip Erase routine. If the 5th byte is not a valid command, the boot program sends back x1H (bit 0) to the controller and returns to the state in which it waits for a command (the third byte) again. In this case, the upper four bits of the acknowledge response are undefined - they hold the same values as the upper four bits of the previously issued command.

6. The 7th byte indicates whether the Chip Erase command is normally completed or not.
At normal completion, completion code (4FH) is sent.
When an error was detected, error code (4CH) is sent.
7. The 9th byte is the next command code.

5) Acknowledge Responses

The boot program represents processing states with specific codes. Table 17-11 to Table 17-14 show the values of possible acknowledge responses to the received data. The upper four bits of the acknowledge response are equal to those of the command being executed. Bit 3 of the code indicates a receive error. Bit 0 indicates an invalid command error, a checksum error or a password error. Bit 1 and bit 2 are always 0. Receive error checking is not done in I/O Interface mode.

Table 17-11 ACK Response to the Serial Operation Mode Byte

Return Value	Meaning
0x86	The SIO can be configured to operate in UART mode. (See Note)
0x30	The SIO can be configured to operate in I/O Interface mode.

Note: If the serial operation mode is determined as UART, the boot program checks if the SIO can be programmed to the baud rate at which the operation mode byte was transferred. If that baud rate is not possible, the boot program aborts, without sending back any response.

Table 17-12 ACK Response to the Command Byte

Return Value	Meaning
0x?8 (See Note)	A receive error occurred while getting a command code.
0x?1 (See Note)	An undefined command code was received. (Reception was completed normally.)
0x10	The RAM Transfer command was received.
0x20	The Show Flash Memory Sum command was received.
0x30	The Show Product Information command was received.
0x40	The Chip Erase command was received.

Note: The upper four bits of the ACK response are the same as those of the previous command code.

Table 17-13 ACK Response to the Checksum Byte

Return Value	Meaning
0xN8 (See Note)	A receive error occurred.
0xN1 (See Note)	A checksum or password error occurred.
0xN0 (See Note)	The checksum was correct.

Note: The upper four bits of the ACK response are the same as those of the operation command code. It is 1 (N=RAM transfer command data [7:4]) when password error occurs.

Table 17-14 ACK Response to Chip Erase Byte

Return Value	Meaning
54H	The Chip Erase enabling command was received.
4FH	The Chip Erase command was completed.
4CH	The Chip Erase command was abnormally completed.

6) Determination of a Serial Operation Mode

The first byte from the controller determines the serial operation mode. To use UART mode for communications between the controller and the target board, the controller must first send a value of 86H at a desired baud rate to the target board. To use I/O Interface mode, the controller must send a value of 30H at 1/16 the desired baud rate. Figure 17-4 shows the waveforms for the first byte.

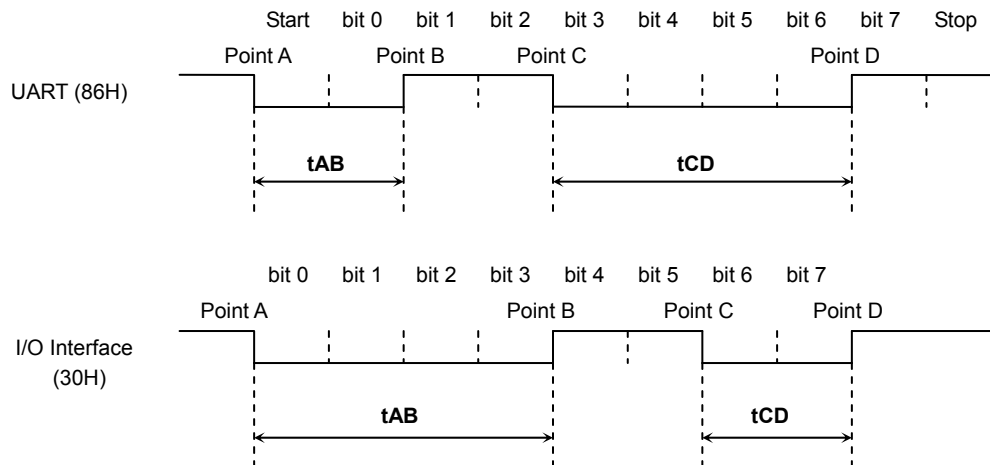


Figure 17-4 Serial Operation Mode Byte

After $\overline{\text{RESET}}$ is released, the boot program monitors the first serial byte from the controller, with the SIO reception disabled, and calculates the intervals of tAB, tAC and tAD. Figure 17-5 shows a flowchart describing the steps to determine the intervals of tAB, tAC and tAD. As shown in the flowchart, the boot program captures timer counts each time a logic transition occurs in the first serial byte. Consequently, the calculated tAB, tAC and tAD intervals are bound to have slight errors. If the transfer goes at a high baud rate, the CPU might not be able to keep up with the speed of logic transitions at the serial receive pin. In particular, I/O Interface mode is more prone to this problem since its baud rate is generally much higher than that for UART mode. To avoid such a situation, the controller should send the first serial byte at 1/16 the desired baud rate.

The flowchart in Figure 17-5 shows how the boot program distinguishes between UART and I/O Interface modes. If the length of tAB is equal to or less than the length of tCD, the serial operation mode is determined as UART mode. If the length of tAB is greater than the length of tCD, the serial operation mode is determined as I/O Interface mode. Bear in mind that if the baud rate is too high or the timer operating frequency is too low, the timer resolution will be coarse, relative to the intervals between logic transitions. This becomes a problem due to inherent errors caused by the way in which timer counts are captured by software; consequently the boot program might not be able to determine the serial operation mode correctly. To prevent this problem, reset UART mode within the programming routine.

For example, the serial operation mode may be determined to be I/O Interface mode when the intended mode is UART mode. To avoid such a situation, when UART mode is utilized, the controller should allow for a time-out period within which it expects to receive an echo-back (86H) from the target board. The controller should give up the communication if it fails to get that echo-back within the allowed time. When I/O Interface mode is utilized, once the first serial byte has been transmitted, the controller should send the SCLK clock after a certain idle time to get an acknowledge response. If the received acknowledge response is not 30H, the controller should give up further communications.

When the intended mode is I/O interface mode, the first byte does not have to be 0x30 as long as t_{AB} is greater than t_{CD} as shown above. 0x91, 0xA1 or 0xB1 can be sent as the first byte code to determine the falling edges of Point A and Point C and the rising edges of Point B and Point D. If t_{AB} is greater than t_{CD} and SIO is selected by the resolution of the operation mode determination, the second byte code is 0x30 even though the transmitted code on the first byte is not 0x30 (The first byte code to determine I/O interface mode is described as 0x30).

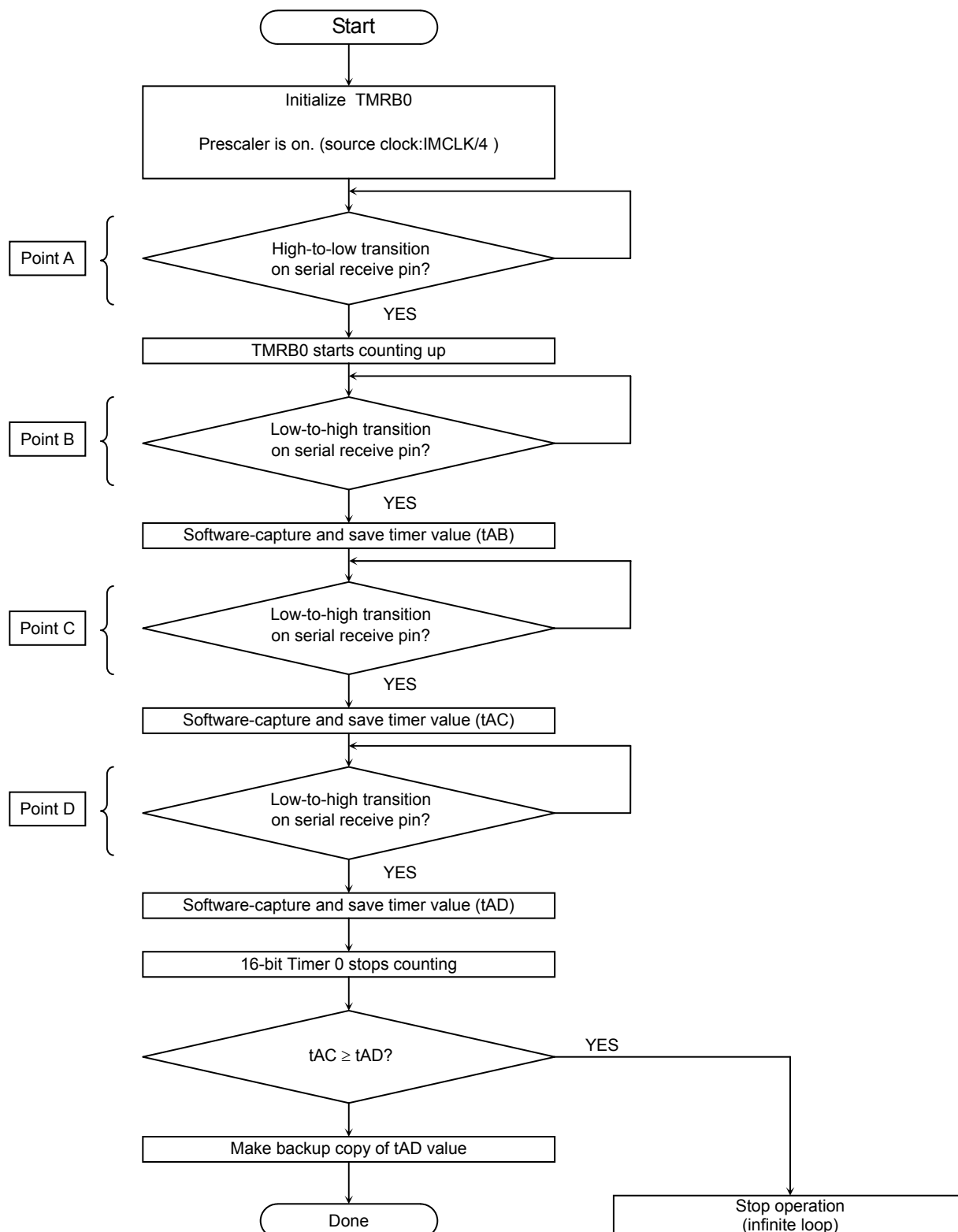


Figure 17-5 Serial Operation Mode Byte Reception Flow

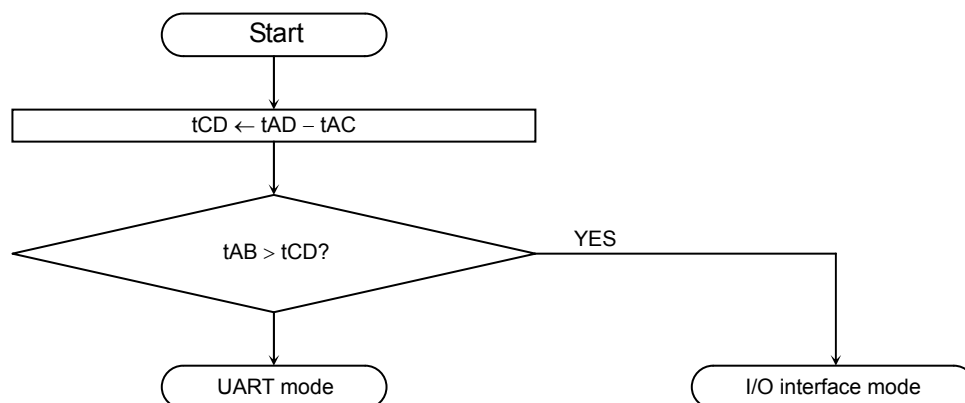


Figure 17-6 Serial Operation Mode Determination Flow

7) Password

The RAM Transfer command (10H) causes the boot program to perform a password check. Following an echo-back of the command code, the boot program checks the contents of the 12-byte password area (0x3F87_FFF4 to 0x3F87_FFFF) within the flash memory. If all these address locations contain the same bytes of data other than FFH, a password area error occurs as shown in Figure 17-7. In this case, the boot program returns an error acknowledge (11H) in response to the checksum byte (the 17th byte), regardless of whether the password sequence sent from the controller is all FFHs.

The password sequence received from the controller (5th to 16th bytes) is compared to the password stored in the flash memory. All of the 12 bytes must match to pass the password check. Otherwise, a password error occurs, which causes the boot program to return an error acknowledge in response to the checksum byte (the 17th byte).

The password check is performed even if the security function is enabled.

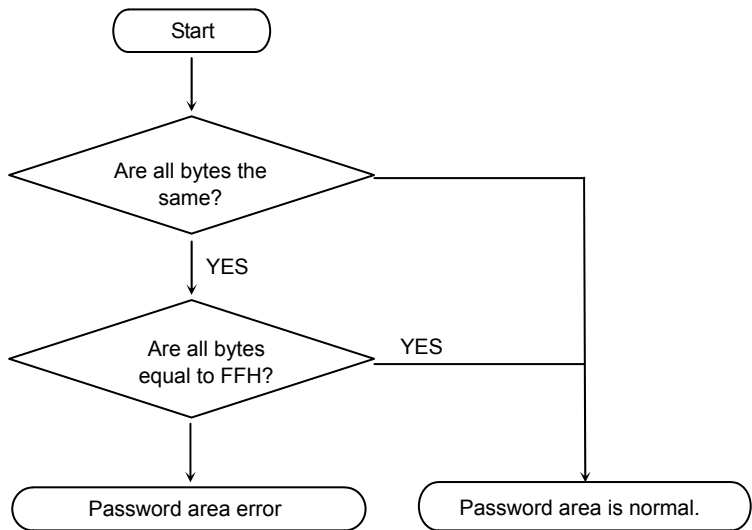


Figure 17-7 Password Area Check Flow

8) Calculation of the Show Flash Memory Sum Command

The result of the sum calculation “byte + byte + byte + ...” is responded by a word quantity. The Show Flash Memory Sum command adds all 512 kbytes of the flash memory together and provides the total sum as a word quantity. The sum is sent to the controller, with the upper eight bits first, followed by the lower eight bits.

Example)

A1H
B2H
C3H
D4H

For the interest of simplicity, assume the depth of the flash memory is four locations. Then the sum of the four bytes is calculated as:

$$A1H + B2H + C3H + D4H = 02EAH$$

Hence, 02H is first sent to the controller, followed by EAH.

9) Checksum Calculation

The checksum byte for a series of bytes of data is calculated by adding the bytes together, dropping the carries, and taking the two's complement of the total sum. The Show Flash Memory Sum command and the Show Product Information command perform the checksum calculation. The controller must perform the same checksum operation in transmitting checksum bytes.

Example) Assume the Show Flash Memory Sum command provides the upper and lower bytes of the sum as E5H and F6H. To calculate the checksum for a series of E5H and F6H:

Add the bytes together

$$E5H + F6H = 1DBH$$

Take the two's complement of the sum, and that is the checksum byte.

$$0 - DBH = 25H$$

(7) General Boot Program Flowchart

Figure 17-8 shows an overall flowchart of the boot program.

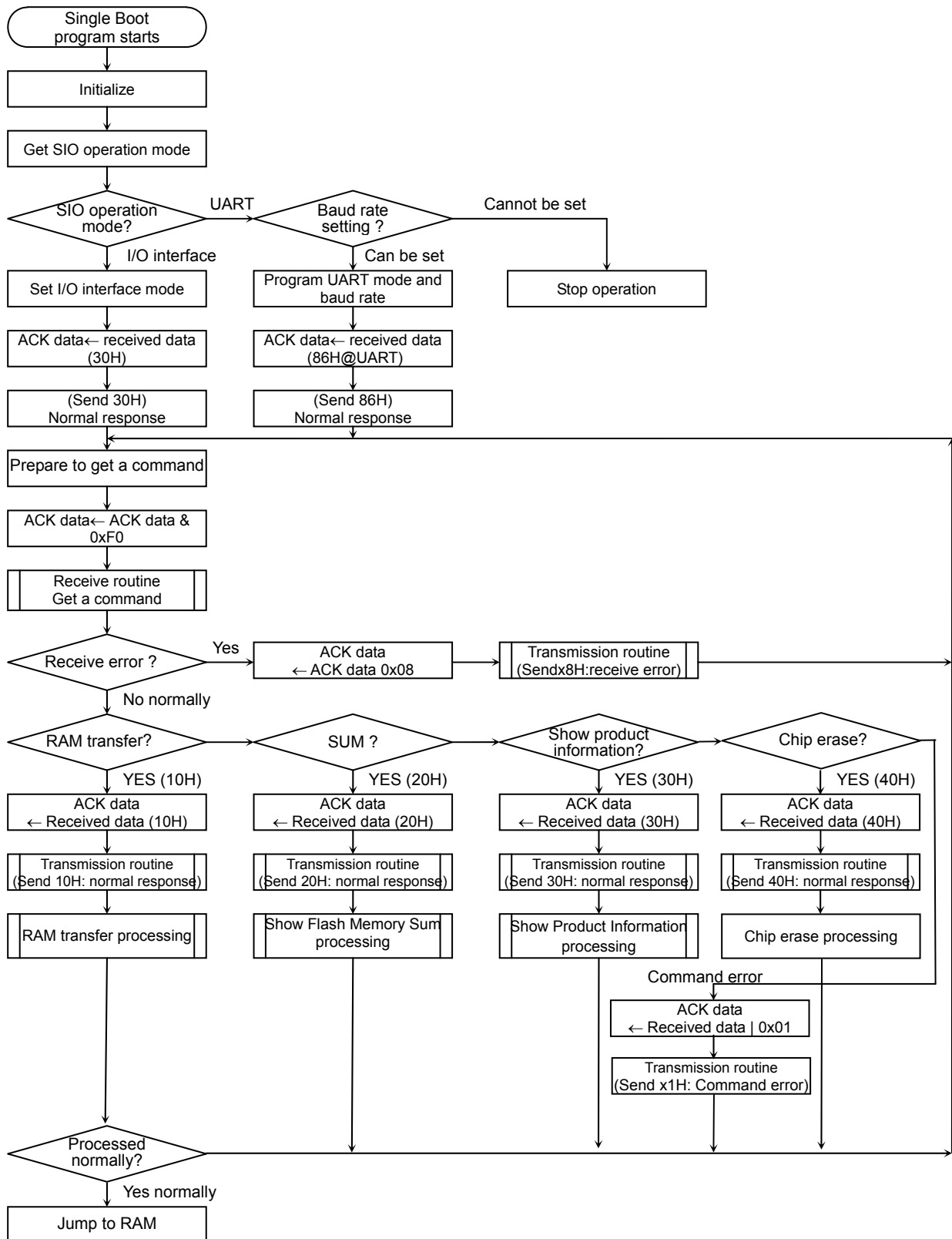


Fig. 17-8 Overall Boot Program Flow

17.3 On-board Programming of Flash Memory (Rewrite/Erase)

In on-board programming, the CPU is to execute software commands for rewriting or erasing the flash memory. The rewrite/erase control program should be prepared by the user beforehand. Because the flash memory content cannot be read while it is being written or erased, it is necessary to run the rewrite/erase program from the internal RAM or from an external memory device after shifting to the user boot mode.

17.3.1 Flash Memory

Except for some functions, writing and erasing flash memory data are in accordance with the standard JEDEC commands. In writing or erasing, use 32-bit data transfer command of the CPU to enter commands to the flash memory. Once the command is entered, the actual write or erase operation is automatically performed internally.

Table 17-15 Flash Memory Functions

Major functions	Description
Automatic page program	Writes data automatically. (128word/1page)
Automatic chip erase	Erases the entire area of the flash memory automatically.
Automatic block erase	Erases a selected block automatically.
Protect function	By writing a 4-bit protection code, the write or erase function can be individually inhibited for each block.

Note that addressing of operation commands is different from the case of standard commands due to the specific interface arrangements with the CPU. Also note that the flash memory is written in 32-bit blocks. So, 32-bit (word) data transfer commands must be used in writing the flash memory.

(1) Block configuration

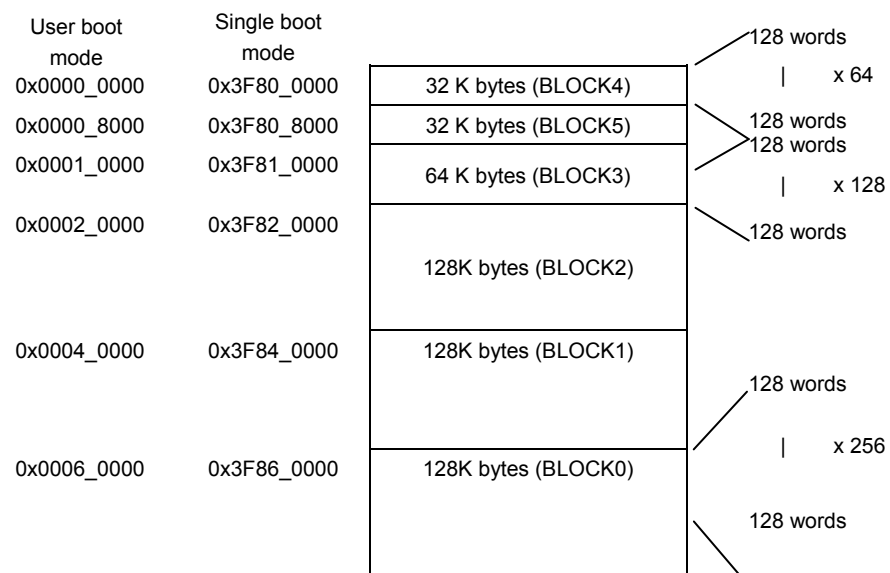


Fig. 17-9 Block Configuration of Flash Memory

(2) Basic operation

Generally speaking, this flash memory device has the following two operation modes:

- The mode to read memory data (Read mode)
- The mode to automatically erase or rewrite memory data (Automatic operation)

Transition to the automatic mode is made by executing a command sequence while it is in the memory read mode. In the automatic operation mode, flash memory data cannot be read and any commands stored in the flash memory cannot be executed. In the automatic operation mode, any interrupt or exception generation cannot set the device to the read mode except when a hardware reset is generated. During automatic operation, be sure not to cause any exceptions other than debug exceptions and reset while a debug port is connected. Any exception generation cannot set the device to the read mode except when a hardware reset is generated.

1) Read

When data is to be read, the flash memory must be set to the read mode. The flash memory will be set to the read mode immediately after power is applied, when CPU reset is removed, or when an automatic operation is normally terminated. In order to return to the read mode from other modes or after an automatic operation has been abnormally terminated, either the Read/reset command (a software command to be described later) or a hardware reset is used. The device must also be in the read mode when any command written on the flash memory is to be executed.

- **Read/reset command and Read command (software reset)**

When ID-Read command is used, the reading operation is terminated instead of automatically returning to the read mode. In this case, the Read/reset command can be used to return the flash memory to the read mode. Also, when a command that has not been completely written has to be canceled, the Read/reset command must be used. The Read command is used to return to the read mode after executing 32-bit data transfer command to write the data "0x0000_00F0" to an arbitrary address of the flash memory.

- **With the Read/reset command, the device is returned to the read mode after completing the third bus write cycle.**

2) Command write

This flash memory uses the command control method. Commands are executed by executing a command sequence to the flash memory. The flash memory executes automatic operation commands according to the address and data combinations applied (refer to Command Sequence).

If it is desired to cancel a command write operation already in progress or when any incorrect command sequence has been entered, the Read/reset command is to be executed. Then, the flash memory will terminate the command execution and return to the read

While commands are generally comprised of several bus cycles, the operation to apply 32-bit data transmit command to the flash memory is called "bus write cycle." The bus write cycles are to be in a specific sequential order and the flash memory will perform an automatic operation when the sequence of the bus write cycle data and address of a command write operation is in accordance with a predefined specific sequence. If any bus write cycle does not follow a

predefined command write sequence, the flash memory will terminate the command execution and return to the read mode.

(Note 1) Command sequences are executed from outside the flash memory area.

(Note 2) Each bus write cycle must be sequentially executed by 32-bit data transmit command. While a command sequence is being executed, access to the flash memory is prohibited. Also, don't generate any interrupt (except debug exceptions when a DSU probe is connected). If such an operation is made, it can result in an unexpected read access to the flash memory and the command sequencer may not be able to correctly recognize the command. While it could cause an abnormal termination of the command sequence, it is also possible that the written command is incorrectly recognized.

(Note 3) For the command sequencer to recognize a command, the device must be in the read mode prior to executing the command. Be sure to check before the first bus write cycle that the FLCS RDY/BSY bit is set to "1." It is recommended to subsequently execute a Read command.

(Note 4) Upon issuing a command, if any address or data is incorrectly written, be sure to perform a software reset to return to the read mode again.

(3) Reset

Hardware reset

A hardware reset is used to cancel the operational mode set by the command write operation when forcibly termination during auto programming/ erasing or abnormal termination during auto operations occurs. The flash memory has a reset input as the memory block and it is connected to the CPU reset signal. Therefore, when the RESET input pin of this device is set to V_{IL} or when the CPU is reset due to any overflow of the watch dog timer, the flash memory will return to the read mode terminating any automatic operation that may be in progress. It should also be noted that applying a hardware reset during an automatic operation can result in incorrect rewriting of data. In such a case, be sure to perform the rewriting again.

Refer to Section 17.2.1 "Reset Operation" for CPU reset operations. After a given reset input, the CPU will read the reset vector data from the flash memory and starts operation after the reset is removed.

(4) Commands

1) Automatic Page Programming

Writing to a flash memory device is to make "1" data cells to "0" data cells. Any "0" data cell cannot be changed to a "1" data cell. For making "0" data cells to "1" data cells, it is necessary to perform an erase operation.

The automatic page programming function of this device writes data in 128 word blocks. A 128 word block is defined by a same [31:9] address and it starts from the address [8:0] = 0 and ends at the address [8:0] = 0x1FF. This programming unit is hereafter referred to as a "page."

Writing to data cells is automatically performed by an internal sequencer and no external control by the CPU is required. The state of automatic page programming (whether it is in writing operation or not) can be checked by the FLCS [0] <RDY/BSY> register.

Also, any new command sequence is not accepted while it is in the automatic page programming mode. If it is desired to interrupt the automatic page programming, use the hardware reset function. If the operation is stopped by a hardware reset operation, it is necessary to once erase the page and then perform the automatic page programming again because writing to the page has not been normally terminated.

The automatic page programming operation is allowed only once for a page already erased. No programming can be performed twice or more times irrespective of the data cell value whether it is "1" or "0." Note that rewriting to a page that has been once written requires execution of the automatic block erase or automatic chip erase command before executing the automatic page programming command again. Note that an attempt to rewrite a page two or more times without erasing the content can cause damages to the device.

No automatic verify operation is performed internally to the device. So, be sure to read the data programmed to confirm that it has been correctly written.

The automatic page programming operation starts when the third bus write cycle of the command cycle is completed. On and after the fifth bus write cycle, data will be written sequentially starting from the next address of the address specified in the fourth bus write cycle (in the fourth bus write cycle, the page top address will be command written) (32 bits of data is input at a time). Be sure to use the 32-bit data transfer command in writing commands on and after the fourth bus cycle. In this, any 32-bit data transfer commands shall not be placed across word boundary. On and after the fifth bus write cycle, data is command written to the same page area. Even if it is desired to write the page only partially, it is required to perform the automatic page programming for the entire page. In this case, the address input for the fourth bus write cycle shall be set to the top address of the page. Be sure to perform command write operation with the input data set to "1" for the data cells not to be set to "0." For example, if the top address of a page is not to be written, set the input data of the fourth bus write cycle to 0xFFFFFFFF to command write the data.

Once the fourth bus cycle is executed, it is in the automatic programming operation. This condition can be checked by monitoring the register bit FLCS [0] <RDY/BSY> (See Table 21-26). Any new command sequence is not accepted while it is in automatic page programming mode. If it is desired to stop operation, use the hardware reset function. Be careful in doing so because data cannot be written normally if the operation is interrupted. When a single page has been command written normally terminating the automatic page writing process, the FLCS [0] <RDY/BSY> bit is set to "1" and it returns to the read mode.

When multiple pages are to be written, it is necessary to execute the page programming command for each page because the number of pages to be written by a single execution of the automatic page program command is limited to only one page. It is not allowed for automatic page programming to process input data across pages.

Data cannot be written to a protected block. When automatic programming is finished, it automatically returns to the read mode. This condition can be checked by monitoring FLCS [0] <RDY/BSY> (See Table 17-16). If automatic programming has failed, the flash memory is locked in the mode and will not return to the read mode. For returning to the read mode, it is necessary to execute hardware reset to reset the flash memory or the device. In this case, while writing to the address has failed, it is recommended not to use the device or not to use the block that includes the failed address.

Note: Software reset becomes ineffective in bus write cycles on and after the fourth bus write cycle of the automatic page programming command.

2) Automatic chip erase

The automatic chip erase operation starts when the sixth bus write cycle of the command cycle is completed.

This condition can be checked by monitoring FLCS [0] <RDY/BSY> (See Table 17-26). While no automatic verify operation is performed internally to the device, be sure to read the data to confirm that data has been correctly erased. Any new command sequence is not accepted while it is in an automatic chip erase operation. If it is desired to stop operation, use the hardware reset function. If the operation is forced to stop, it is necessary to perform the automatic chip erase operation again because the data erasing operation has not been normally terminated.

Also, any protected blocks cannot be erased. If all the blocks are protected, the automatic chip erase operation will not be performed and it returns to the read mode after completing the sixth bus read cycle of the command sequence. When an automatic chip erase operation is normally terminated, it automatically returns to the read mode. If an automatic chip erase operation has failed, the flash memory is locked in the mode and will not return to the read mode.

For returning to the read mode, it is necessary to execute hardware reset to reset the device. In this case, the failed block cannot be detected. It is recommended not to use the device anymore or to identify the failed block by using the block erase function for not to use the identified block anymore.

3) Automatic block erase (for each block)

The automatic block erase operation starts when the sixth bus write cycle of the command cycle is completed.

This status of the automatic block erase operation can be checked by monitoring FLCS <RDY/BSY> (See Table 17-26). While no automatic verify operation is performed internally to the device, be sure to read the data to confirm that data has been correctly erased. Any new command sequence is not accepted while it is in an automatic block erase operation. If it is desired to stop operation, use the hardware reset function. In this case, it is necessary to perform the automatic block erase operation again because the data erasing operation has not been normally terminated.

Also, any protected blocks cannot be erased. If an automatic block erase operation has failed, the flash memory is locked in the mode and will not return to the read mode. In this case, execute hardware reset to reset the device.

4) Automatic programming of protection bits (for each block)

This device is implemented with six protection bits. This protection can be set for each block. See Table 17-19 and 17-20 for table of protection bit addresses. This device assigns 1 bit to 1 block as a protection bit. The applicable protection bit is specified by PBA in the seventh bus write cycle. By automatically programming the protection bits, write and/or erase functions can be inhibited (for protection) individually for each block. The protection status of each block can be checked by the FLCS <BLPRO 5:0> register to be described later. This status of the automatic programming operation to set protection bits can be checked by monitoring FLCS <RDY/BSY> (See Table 17-26). Any new command sequence is not accepted while automatic programming is in progress to program the protection bits. If it is desired to stop the programming operation, use the hardware reset function. In this case, it is necessary to perform the programming operation again because the protection bits may not have been correctly programmed. If all the protection bits have been programmed, the FLCS <BLPRO 5:0> bits are set to "0 x 3F" indicating that it is in the protected state (See Table 17-26). After this, no command writing to any blocks can be performed.

Note: Software reset is ineffective in the seventh bus write cycle of the automatic protection bit programming command. The FLCS <RDY/BSY> bit turns to "0" after entering the seventh bus write cycle.

5) Automatic erasing of protection bits

Different results will be obtained when the automatic protection bit erase command is executed depending on the status of the protection bits and the security bits. It depends on the status of FLCS <BLPRO 5:0> whether it is set to "0 x 3F" or to any other values if SECBIT<SECBIT> is 0x1. Be sure to check the value of FLCS <BLPRO 5:0> before executing the automatic protection bit erase command. See chapter 22 for details.

• When FLCS <BLPRO 5:0> is set to "0 x 3F" (all the protection bits are programmed):

When the automatic protection bit erase command is command written, the flash memory is automatically initialized within the device. When the seventh bus write cycle is completed, the entire area of the flash memory data cells is erased and then the protection bits are erased. This operation can be checked by monitoring FLCS <RDY/BSY>. If the automatic operation to erase protection bits is normally terminated, FLCS will be set to "0x00000001." While no automatic verify operation is performed internally to the device, be sure to read the data to confirm that it has been correctly erased. For returning to the read mode while the automatic operation after the seventh bus cycle is in progress, it is necessary to use the hardware reset to reset the device. If this is done, it is necessary to check the status of protection bits by FLCS <BLPRO 5:0> after returning to the read mode and perform either the automatic protection bit erase, automatic chip erase, or automatic block erase operation, as appropriate.

· **When FLCS <BLPRO 5:0> is other than "0 x 3F" (not all the protection bits are programmed):**

The protection condition can be canceled by the automatic protection bit erase operation. With this device, protection bits set by an individual block can be erased handling all the blocks at a time as shown in Table 17-22. The target bits are specified in the seventh bus write cycle and when the command is completed, the device is in a condition all the blocks are erased. The protection status of each block can be checked by FLCS <BLPRO 5:0> to be described later. This status of the programming operation for automatic protection bits can be checked by monitoring FLCS <RDY/BSY>. When the automatic operation to erase protection bits is normally terminated, the protection bits of FLCS <BLPRO 5:0> selected for erasure are set to "0."

In any case, any new command sequence is not accepted while it is in an automatic operation to erase protection bits. If it is desired to stop the operation, use the hardware reset function. When the automatic operation to erase protection bits is normally terminated, it returns to the read mode.

The FLCS <RDY/BSY> bit is "0" while in automatic operation and it turns to "1" when the automatic operation is terminated.

6) ID-Read

Using the ID-Read command, you can obtain the type and other information on the flash memory contained in the device. The data to be loaded will be different depending on the address [15:14] of the fourth and subsequent bus write cycles (recommended input data is 0x00). On and after the fourth bus write cycle, when an arbitrary flash memory area is read, the ID value will be loaded. Once the fourth bus write cycle of an ID-Read command has passed, the device will not automatically return to the read mode. In this condition, the set of the fourth bus write cycle and ID-Read commands can be repetitively executed. For returning to the read mode, use the Read/reset command or hardware reset command.

(5) Flash control/ status register

This register is used to monitor the status of the flash memory and to indicate the protection status of each block.

Table 17-16 Flash Control Register

FLCS
0x4004_0520

	31	30	29	28	27	26	25	24
bit Symbol	-	-	-	-	-	-	-	-
Read/Write	R							
After reset	0							
Function	"0" is read.							
	23	22	21	20	19	18	17	16
bit Symbol	-	-	BLPRO5	BLPRO4	BLPRO3	BLPRO2	BLPRO1	BLPRO0
Read/Write	R		R	R	R	R	R	R
After reset	0		0	0	0	0	0	0
Function	"0" is read.		Protection for Block 5 0: disabled 1: enabled	Protection for Block 4 0: disabled 1: enabled	Protection for Block 3 0: disabled 1: enabled	Protection for Block 2 0: disabled 1: enabled	Protection for Block 1 0: disabled 1: enabled	Protection for Block 0 0: disabled 1: enabled
	15	14	13	12	11	10	9	8
bit Symbol	-	-	-	-	-	-	-	-
Read/Write	R							
After reset	0							
Function	"0" is read.							
	7	6	5	4	3	2	1	0
bit Symbol	-	-	-	-	-	-	-	RDY/BSY
Read/Write	R							R
After reset	0							1
Function	"0" is read.							Ready/Busy 0: Auto operating 1: Auto operation terminated

Bit 0: Ready/Busy flag bit

The RDY/BSY output is provided as a means to monitor the status of automatic operation. This bit is a function bit for the CPU to monitor the function. When the flash memory is in automatic operation, it outputs "0" to indicate that it is busy. When the automatic operation is terminated, it returns to the ready state and outputs "1" to accept the next command. If the automatic operation has failed, this bit maintains the "0" output. By applying a hardware reset, it returns to "1."

Bit [21:16]: Protection status bits

Each of the protection bits (6 bits) represents the protection status of the corresponding block. When a bit is set to "1," it indicates that the block corresponding to the bit is protected. When the block is protected, data cannot be written to it.

(Note) Please issue it after confirming the command issue is always a ready state. A normal command not only is sent when the command is issued to a busy inside but also there is a possibility that the command after that cannot be input. In that case, please return by system reset or the reset command.

Table 17-17 Security bit register

SECBIT
0x4004_0500

	31	30	29	28	27	26	25	24
bit Symbol	-	-	-	-	-	-	-	-
Read/Write	R							
After reset	0							
Function	"0" is read							
	23	22	21	20	19	18	17	16
bit Symbol	-	-	-	-	-	-	-	-
Read/Write	R							
After reset	0							
Function	"0" is read							
	15	14	13	12	11	10	9	8
bit Symbol	-	-	-	-	-	-	-	-
Read/Write	R							
After reset	0							
Function	"0" is read							
	7	6	5	4	3	2	1	0
bit Symbol	-	-	-	-	-	-	-	SECBIT
Read/Write	R							R/W
After reset	0							1
Function	"0" is read							Security bits 0:disabled 1:enabled

(Note) This register is initialized only by power-on reset.

(6) List of Command Sequences

Table 17-18 Flash Memory Access from the Internal CPU

Command sequence	First bus cycle	Second bus cycle	Third bus cycle	Fourth bus cycle	Fifth bus cycle	Sixth bus cycle	Seventh bus cycle
	Addr.	Addr.	Addr.	Addr.	Addr.	Addr.	Addr.
	Data	Data	Data	Data	Data	Data	Data
Read	0xXX	—	—	—	—	—	—
	0xF0	—	—	—	—	—	—
Read/Reset	0x54XX	0xAAXX	0x54XX	RA	—	—	—
	0xAA	0x55	0xF0	RD	—	—	—
ID-Read	0x54XX	0xAAXX	0x54XX	IA	0xXX	—	—
	0xAA	0x55	0x90	0x00	ID	—	—
Automatic page programming (note)	0x54XX	0xAAXX	0x54XX	PA	PA	PA	PA
	0xAA	0x55	0xA0	PD0	PD1	PD2	PD3
Automatic chip erase	0x54XX	0xAAXX	0x54XX	0x54XX	0xAAXX	0x54XX	—
	0xAA	0x55	0x80	0xAA	0x55	0x10	—
Auto Block erase (note)	0x54XX	0xAAXX	0x54XX	0x54XX	0xAAXX	BA	—
	0xAA	0x55	0x80	0xAA	0x55	0x30	—
Protection bit programming	0x54XX	0xAAXX	0x54XX	0x54XX	0xAAXX	0x54XX	PBA
	0xAA	0x55	0x9A	0xAA	0x55	0x9A	0x9A
Protection bit erase	0x54XX	0xAAXX	0x54XX	0x54XX	0xAAXX	0x54XX	PBA
	0xAA	0x55	0x6A	0xAA	0x55	0x6A	0x6A

Supplementary explanation

- RA: Read address
- **RD: Read data**
- IA: ID address
- **ID: ID data**
- PA: Program page address
PD: Program data (32 bit data)
After the fourth bus cycle, enter data in the order of the address for a page.
- BA: Block address
- PBA: Protection bit address
- 0x54xx: Substitutable by 0x55xx

(Note 1) Always set "0" to the address bits [1:0] in the entire bus cycle. (Recommendable setting values to bits [7:2] are "0".)

(Note 2) Bus cycles are "bus write cycles" except for the second bus cycle of the Read command, the fourth bus cycle of the Read/reset command, and the fifth bus cycle of the ID-Read command. Bus write cycles are executed by 32-bit data transfer commands. The address [31:16] in each bus write cycle should be the target flash memory address [31:16] of the command sequence. Use "Addr." in the table for the address [15:0].

(7) Address bit configuration for bus write cycles

Table 17-19 Address Bit Configuration for Bus Write Cycles

Address	Addr [31:19]	Addr [18:17]	Addr [16]	Addr [15]	Addr [14]	Addr [13:11]	Addr [10:9]	Addr [8]	Addr [7:0]
Normal comm nds	Normal bus write cycle address configuration								
	Flash area	"0" is recommended.			Command				Addr[1:0]=0 (fixed) Others:0 (recommended)
Block erase	BA: Block address (Set the sixth bus write cycle address for block erase operation)								
	Flash area	Block selection			Addr[1:0]=0 (fixed), Others: 0 (recommended)				
Auto page progra mming	PA: Program page address (Set the fourth bus write cycle address for page programming operation)								
	Flash area	Block selection			Page selection				Addr[1:0]=0 (fixed), Others: 0 (recommended)
ID-REA D	IA: ID address (Set the fourth bus write cycle address for ID-Read operation)								
	Flash area	"0" is recommended		ID address		Addr[1:0]=0 (fixed), Others: 0 (recommended)			
Protec tion bit progra mming	PBA: Protection bit address (Set the seventh bus write cycle address for protection bit programming)								
	Flash area	Protection bit selection	"0" is recommended				Protection bit selection	Addr[1:0]=0 (fixed), Others: 0 (recommended)	
Protec tion bit erase	PBA: Protection bit address (Set the seventh bus erase cycle address for protection bit erasure)								
	Flash area	Protection bit selection	Addr[1:0]=0 (fixed), Others: 0 (recommended)						

(Note 1) Table 17-18 "Flash Memory Access from the Internal CPU" can also be used.

(Note 2) Address setting can be performed according to the "Normal bus write cycle address configuration" from the first bus cycle.

(Note 3) "0" is recommended" can be changed as necessary.

Table 17-20 Block Erase Address Table

Block	Address (User boot mode)	Address (Single boot mode)	Size
Block4	0x0000_0000-0x0000_7FFF	0x3F80_0000-0x3F80_7FFF	32K byte
Block5	0x0000_8000-0x0000_FFFF	0x3F80_0000-0x3F80_FFFF	32K byte
Block3	0x0001_0000-0x0001_FFFF	0x3F81_0000-0x3F81_FFFF	64K byte
Block2	0x0002_0000-0x0003_FFFF	0x3F82_0000-0x3F83_FFFF	128K byte
Block1	0x0004_0000-0x0005_FFFF	0x3F84_0000-0x3F85_FFFF	128K byte
Block0	0x0006_0000-0x0007_FFFF	0x3F86_8000-0x3F87_FFFF	128K byte

Example: When BA0 is to be erased, any single address in the range 0x0000_0000 to 0x0000_7FFF or 0x3F80_0000 to 0x3F80_7FFF may be entered.

(Note) As for the addresses from the first to the fifth bus cycles, specify the upper 4 bit with the corresponding flash memory addresses of the blocks to be erased.

Table 17-21 Protection Bit Programming Address Table

Block	Protection bit	The seventh bus write cycle address [18:17] [10:9]			
		Address [18]	Address [17]	Address [10]	Address [9]
Block0	BLPRO0	0	0	0	0
Block1	BLPRO1	0	0	0	1
Block2	BLPRO2	0	0	1	0
Block3	BLPRO3	0	0	1	1
Block4	BLPRO4	0	1	0	0
Block5	BLPRO5	0	1	0	1

Table 17-22 Protection Bit Erase Address Table

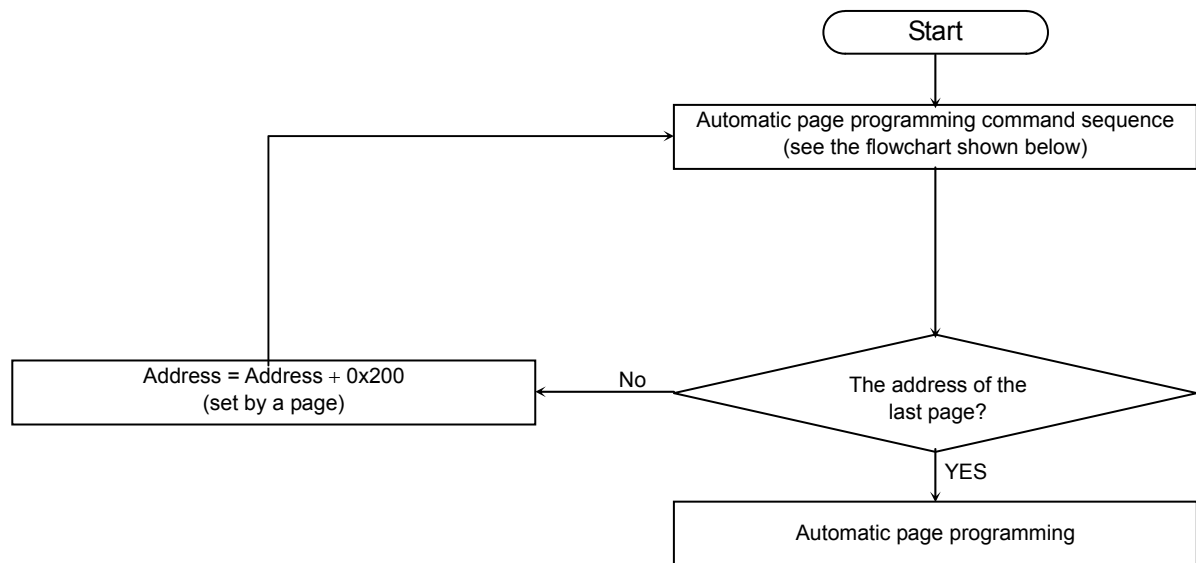
Block	Protection bit	The seventh bus write cycle address [18:17]	
		Address [18]	Address [17]
Block0~3	BLPRO0~3	0	0
Block4~5	BLPRO4~5	0	1

The protection bit erase command cannot erase by individual block.

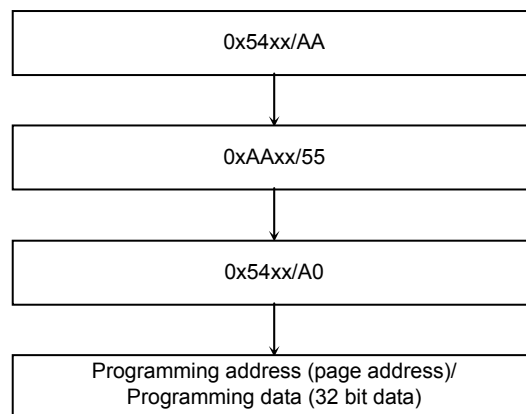
Table 17-23 The ID-Read command's fourth bus write cycle ID address (IA) and the data to be read by the following 32-bit data transfer command (ID)

IA [15:14]	ID [7: 0]	Code
00b	0x98	Manufacturer code
01b	0x5A	Device code
10b	Reserved	---
11b	0x12	Macro code

(8) Flowchart

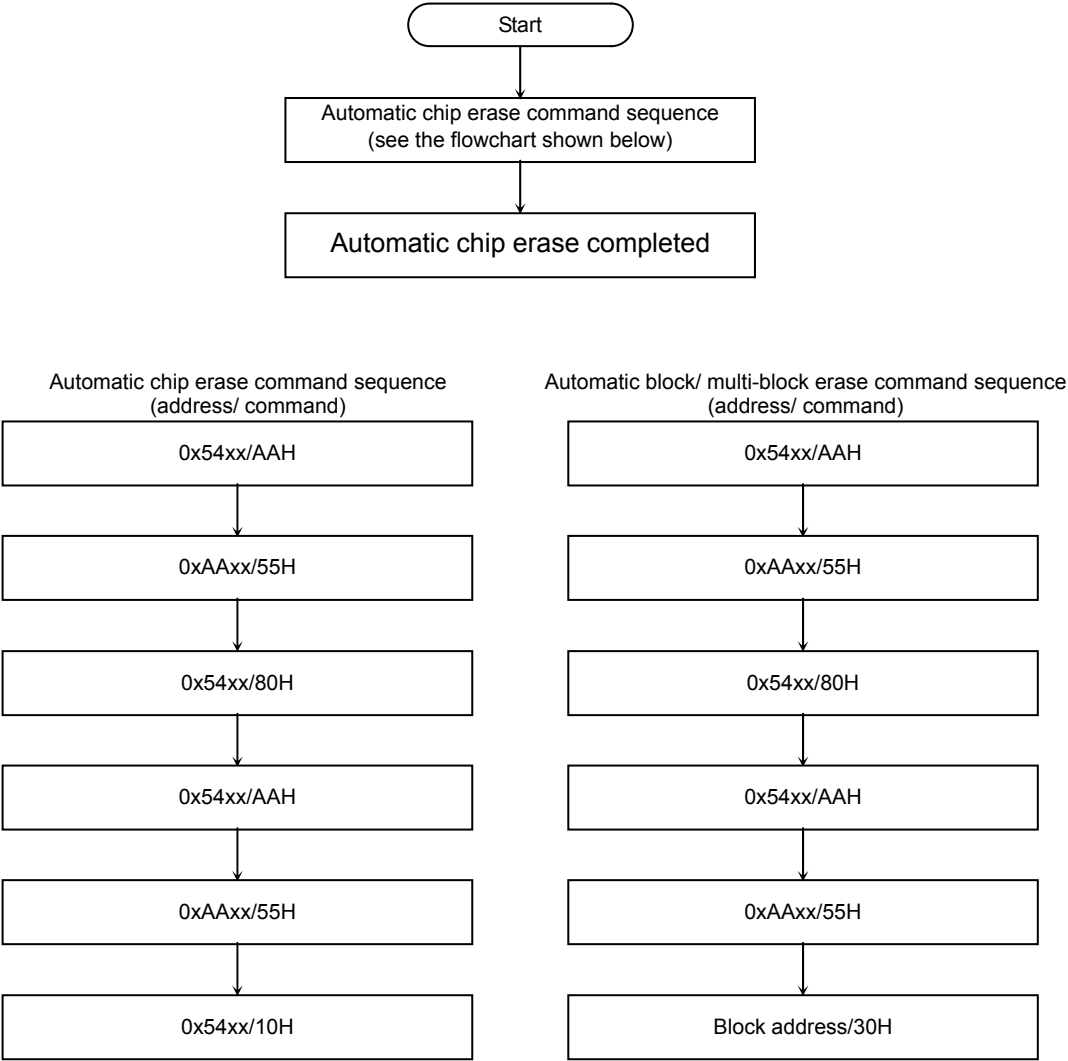


Automatic Page Programming Command Sequence (Address/ Command)



(Note) Command sequence is executed by 0x54xx or 0x55xx.

Fig. 17-10 Automatic Programming



(Note) Command sequence is executed by 0x54xx or 0x55xx.

Fig. 17-11 Automatic Erase

18. ROM protection

18.1 Outline

The TMPM330 offers two kinds of ROM protection/ security functions. One is a write/ erase-protection function for the internal flash ROM data. The other is a security function that restricts internal flash ROM data readout and debugging.

18.2 Features

18.2.1 Write/ erase-protection function

The write/ erase-protection function enables the internal flash to prohibit the writing and erasing operation for each block.

This function is available with a single chip mode, single boot mode and writer mode. To activate the function, write “1” to the corresponding bits to a block to protect. Writing “0” to the bits cancels the protection. The protection settings of the bits can be monitored by FLCS register <BLPRO5:0>. See chapter 21 for programming details.

18.2.2 Security function

The security function restricts flash ROM data readout and debugging.
This function is available under the conditions shown below.

- 1) Register SECBIT <SECBIT> bit is set to "1".
- 2) All the protection bits used for the write/erase-protection function are set to "1".

Note) Register SECBIT <SECBIT> bit is set to "1" at a power-on reset right after power-on.

Table 18-1 shows details of the restrictions by the security function.

Table 18-1 Restrictions by the security function

Item	Details
1) ROM data readout	Data in the ROM area cannot be read out when writer mode is set. By executing readout, the company code 0x0098 is read. The ROM reading operation is available with a single chip mode and single boot mode.
2) Debug port	Communication of JTAG/SW and trace are prohibited.
3) Command for flash memory	Writing a command to the flash memory is prohibited. An attempt to erase the contents in the bits used for the write/erase-protection erases all the protection bits.

18.3 Register

The flash control register shows the status of the flash memory operation and the protection of each block.

Table 18-2 Flash control register

FLCS
0x4004_0520

	31	30	29	28	27	26	25	24
bit Symbol	-	-	-	-	-	-	-	-
Read/Write	R							
After reset	0							
Function	'0'is read							
	23	22	21	20	19	18	17	16
bit Symbol	-	-	BLPRO5	BLPRO4	BLPRO3	BLPRO2	BLPRO1	BLPRO0
Read/Write	R		R	R	R	R	R	R
After reset	0		0	0	0	0	0	0
Function	'0'is read		Protection for Block 5 0: disabled 1: enabled	Protection for Block 4 0: disabled 1: enabled	Protection for Block 3 0: disabled 1: enabled	Protection for Block 2 0: disabled 1: enabled	Protection for Block 1 0: disabled 1: enabled	Protection for Block 0 0: disabled 1: enabled
	15	14	13	12	11	10	9	8
bit Symbol	-	-	-	-	-	-	-	-
Read/Write	R							
After reset	0							
Function	'0'is read							
	7	6	5	4	3	2	1	0
bit Symbol	-	-	-	-	-	-	-	RDY/BSY
Read/Write	R							R
After reset	0							1
Function	'0'is read							Ready/Bus y 0:Auto operating 1:Auto operation terminated

Bit 0: Ready/Busy flag bit

The RDY/BSY output is provided as a means to monitor the status of automatic operation. This bit is a function bit for the CPU to monitor the function. When the flash memory is in automatic operation, it outputs "0" to indicate that it is busy. When the automatic operation is terminated, it returns to the ready state and outputs "1" to accept the next command. If the automatic operation has failed, this bit maintains the "0" output. By applying a hardware reset, it returns to "1."

Bit [21:16]: Protection status bits

Each of the protection bits (6 bits) represents the protection status of the corresponding block. When a bit is set to "1," it indicates that the block corresponding to the bit is protected. When the block is protected, data cannot be written to it.

(Note) Please issue it after confirming the command issue is always a ready state. A normal command not only is sent when the command is issued to a busy inside but also there is a possibility that the command after that cannot be input. In that case, please return by system reset or the reset command.

Table 18-3 Security bit register

SECBIT
0x4004_0500

	31	30	29	28	27	26	25	24
bit Symbol	-	-	-	-	-	-	-	-
Read/Write	R							
After reset	0							
Function	'0' is read							
	23	22	21	20	19	18	17	16
bit Symbol	-	-	-	-	-	-	-	-
Read/Write	R							
After reset	0							
Function	'0' is read							
	15	14	13	12	11	10	9	8
bit Symbol	-	-	-	-	-	-	-	-
Read/Write	R							
After reset	0							
Function	'0' is read							
	7	6	5	4	3	2	1	0
bit Symbol	-	-	-	-	-	-	-	SECBIT
Read/Write	R							R/W
After reset	0							1
Function	'0' is read							Security bits 0:disabled 1:enabled

18.4 Writing and erasing

18.4.1 Protection bits

Writing and erasing protection bits are available with a single chip mode, single boot mode and writer mode.

Writing to the protection bits is done on block-by-block basis.

Erasing of the protection bits is done by two groups of the blocks: block 0 through 3 and block 4 through 5. When the settings for all the blocks are "1", erasing must be done after setting register SECBIT <SECBIT> bit to "0". Setting "1" at that situation erases all the protection bits. To write and erase the protection bits, command sequence is used.

See chapter 21 for details.

18.4.2 Security bit

Rewriting of the security bits is available with a single chip mode and single boot mode. Register SECBIT <SECBIT> bit that activates security function is set to "1" at a power-on reset right after power-on. The bit is rewritten by the following procedure.

- 1) Write the code 0xa74a9d23 to SECBIT register.
- 2) Write data within 16 clocks from the above.

Note) The above procedure is enabled only when using 32-bit data transfer command.