

TOSHIBA

TOSHIBA Original CMOS 16-Bit Microcontroller

TLCS-900/L1 Series

TMP91CW18A

TOSHIBA CORPORATION

Semiconductor Company

Preface

Thank you very much for making use of Toshiba microcomputer LSIs.
Before use this LSI, refer the section, "Points of Note and Restrictions".
Especially, take care below cautions.

****CAUTION****

How to release the HALT mode

Usually, interrupts can release all halts status. However, the interrupts = ($\overline{\text{NMI}}$, INT0 to INT4), which can release the HALT mode may not be able to do so if they are input during the period CPU is shifting to the HALT mode (for about 5 clocks of f_{FPH}) with IDLE1 or STOP mode (IDLE2 is not applicable to this case). (In this case, an interrupt request is kept on hold internally.)

If another interrupt is generated after it has shifted to HALT mode completely, halt status can be released without difficulty. The priority of this interrupt is compare with that of the interrupt kept on hold internally, and the interrupt with higher priority is handled first followed by the other interrupt.

CMOS 16-Bit Microcontroller TMP91CW18AF

1. Outline and Features

TMP91CW18A is a high-speed 16-bit microcontroller designed for the control of various mid-to large-scale equipment.

TMP91CW18AF comes in a 80-pin flat package.

Listed below are the features.

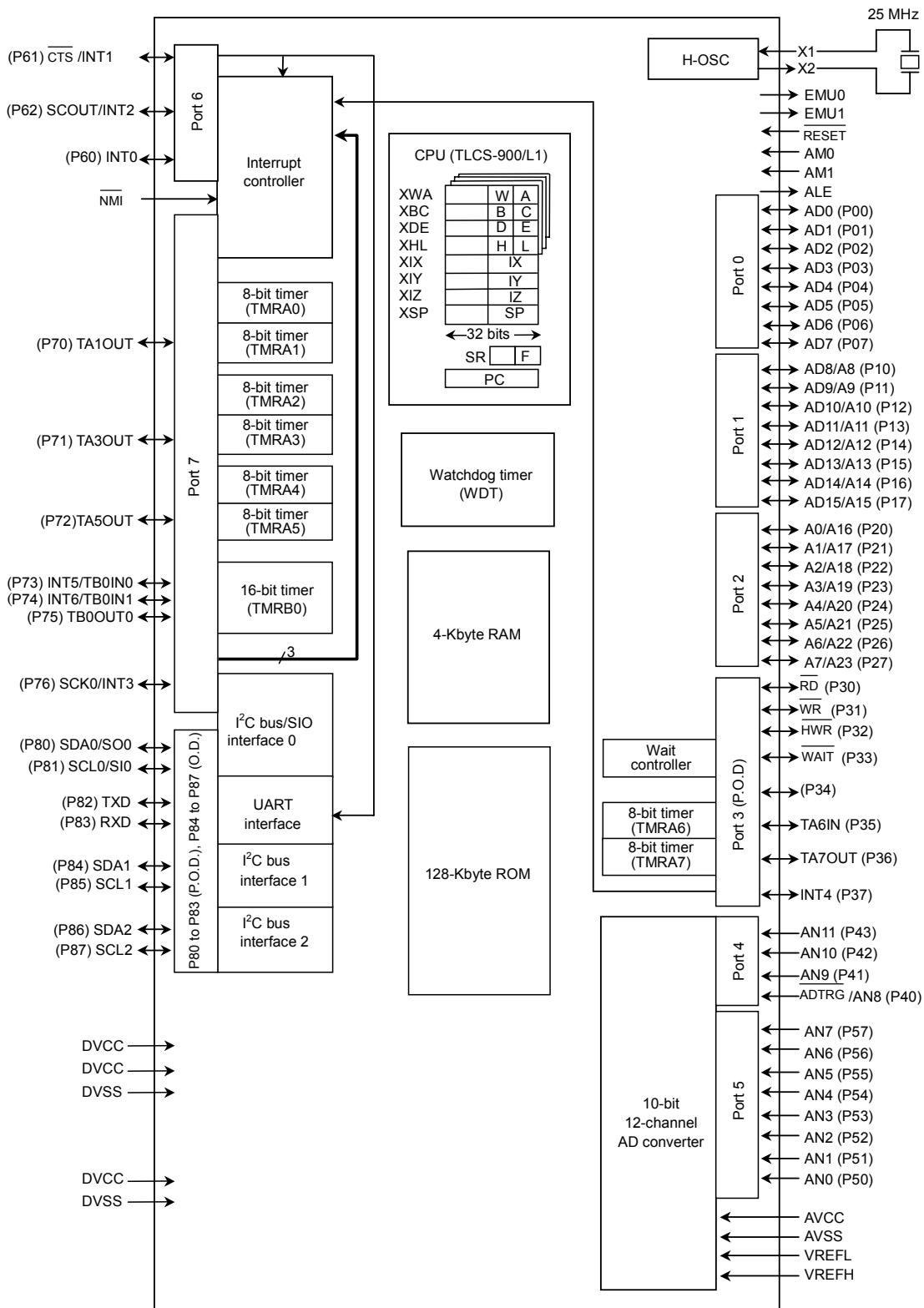
- (1) High-speed 16-bit CPU (900/L1 CPU)
 - Instruction mnemonics are upward-compatible with TLCS-90/900/900H
 - 16 Mbytes of linear address space
 - General-purpose registers and register banks
 - 16-bit multiplication and division instructions: Bit transfer and arithmetic instructions
 - Micro DMA: 4 channels (640 ns /2 bytes at 25 MHz)
- (2) Minimum instruction execution time: 160 ns (at 25 MHz)
- (3) Built-in RAM: 4 Kbytes
Built-in ROM: 128 Kbytes
- (4) External memory expansion
 - Expandable up to 16 Mbytes (Shared program/data area)

RESTRICTIONS ON PRODUCT USE

030619EBP

- The information contained herein is subject to change without notice.
- The information contained herein is presented only as a guide for the applications of our products. No responsibility is assumed by TOSHIBA for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of TOSHIBA or others.
- TOSHIBA is continually working to improve the quality and reliability of its products. Nevertheless, semiconductor devices in general can malfunction or fail due to their inherent electrical sensitivity and vulnerability to physical stress. It is the responsibility of the buyer, when utilizing TOSHIBA products, to comply with the standards of safety in making a safe design for the entire system, and to avoid situations in which a malfunction or failure of such TOSHIBA products could cause loss of human life, bodily injury or damage to property.
In developing your designs, please ensure that TOSHIBA products are used within specified operating ranges as set forth in the most recent TOSHIBA products specifications. Also, please keep in mind the precautions and conditions set forth in the "Handling Guide for Semiconductor Devices," or "TOSHIBA Semiconductor Reliability Handbook" etc..
- The TOSHIBA products listed in this document are intended for usage in general electronics applications (computer, personal equipment, office equipment, measuring equipment, industrial robotics, domestic appliances, etc.). These TOSHIBA products are neither intended nor warranted for usage in equipment that requires extraordinarily high quality and/or reliability or a malfunction or failure of which may cause loss of human life or bodily injury ("Unintended Usage"). Unintended Usage include atomic energy control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, combustion control instruments, medical instruments, all types of safety devices, etc.. Unintended Usage of TOSHIBA products listed in this document shall be made at the customer's own risk.
- The products described in this document are subject to the foreign exchange and foreign trade laws.
- TOSHIBA products should not be embedded to the downstream products which are prohibited to be produced and sold, under any law and regulations.
- For a discussion of how the reliability of microcontrollers can be predicted, please refer to Section 1.3 of the chapter entitled Quality and Reliability Assurance/Handling Precautions.

- (5) Wait controller: 1 channel
- (6) 8-bit timer: 8 channels
- (7) 16-bit timer: 1 channel
- (8) General-purpose serial interface (UART): 1 channel
- (9) Serial bus interface (I²C/Select of synchronous): 1 channel
- (10) Serial bus interface (I2C): 2 channels
- (11) 10-bit AD converter (S/H): 12 channels
 - Conversion time: 84 states (6.72 μ s at $f_{PPH} = 25$ MHz)
- (12) Watchdog timer
- (13) Interrupts function
 - 9 CPU interrupts: Software interrupt instruction and illegal instruction
 - 21 internal interrupts: Seven selectable priority levels
 - 8 external interrupts:
- (14) Input/Output ports: 62 pins
 - I/O: 50 pins (Programmable open drain: 12 pins)
 - Input: 12 pins
- (15) Standby mode
 - Three HALT modes: Programmable IDLE2, IDLE1, STOP
- (16) Clock controller
 - Clock gear: changes high-frequency clock f_c to $f_c/16$
- (17) Open voltage
 - $V_{cc} = 4.5$ V to 5.5 V (f_c max = 25 MHz)
- (18) Package
 - P-QFP80-1420-0.80B



() : Initial function after reset

Figure 1.1 TMP91CW18A Block Diagram

2. Pin Assignment and Pin Functions

The assignment of input/output pins for the TMP91CW18A, their names and functions are as follows.

2.1 Pin Assignment Diagram

Figure 2.1.1 shows the pin assignment of the TMP91CW18AF.

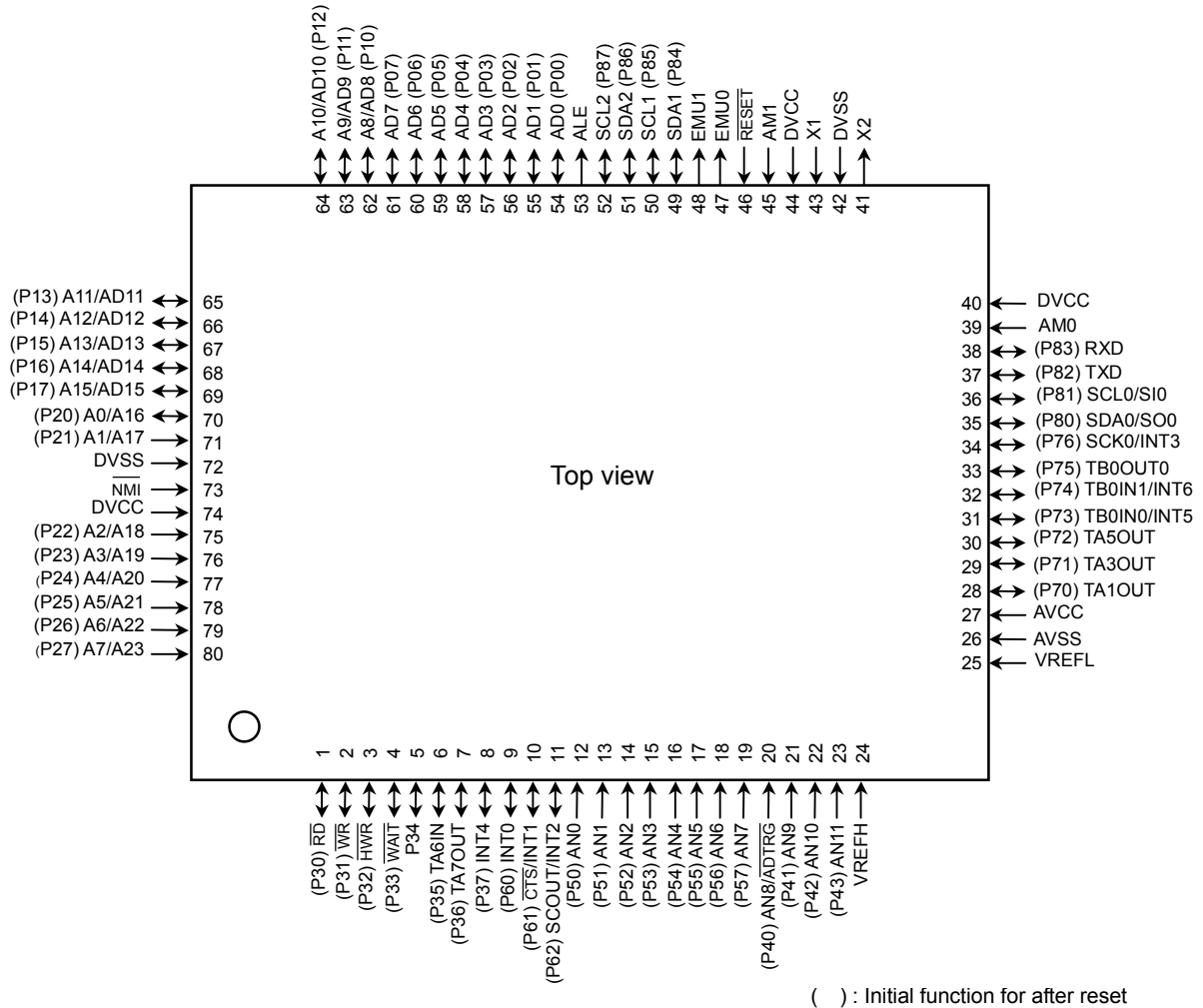


Figure 2.1.1 Pin Assignment Diagram (80-pin QFP)

2.2 Pin Names and Functions

The names of the input/output pins and their functions are described below.

Table 2.2.1 Pin Names and Functions (1/3)

Pin Name	Number of Pins	I/O	Functions
P00 to P07 AD0 to AD7	8	I/O Tri-state	Port 0: I/O port that allows I/O to be selected at the bit level Address and data (Lower): Bits 0 to 7 of address and data bus
P10 to P17 AD8 to AD15 A8 to A15	8	I/O Tri-state Output	Port 1: I/O port that allows I/O to be selected at the bit level Address and data (Upper): Bits 8 to 15 for address and data bus Address: Bits 8 to 15 of address bus
P20 to P27 A0 to A7 A16 to A23	8	I/O Output Output	Port 2: I/O port that allows I/O to be selected at the bit level Address: Bits 0 to 7 of address bus Address: Bits 16 to 23 of address bus
P30 \overline{RD}	1	I/O Output	Port 30: I/O port By setting (P3<P30> = 0, P3FC<P30FC> = 1), \overline{RD} signal is generated during reading internal areas. Read: Strobe signal for reading external memory Open-drain output pin by programmable
P31 \overline{WR}	1	I/O Output	Port 31: I/O port Write: Strobe signal for writing data to pins AD0 to AD7 Open-drain output pin by programmable
P32 \overline{HWR}	1	I/O Output	Port 32: I/O port (with pull-up resistor) High write: Strobe signal for writing data to pins AD8 to AD15 Open-drain output pin by programmable
P33 \overline{WAIT}	1	I/O Input	Port 33: I/O port (with pull-up resistor) Wait: Pin used to request CPU bus wait Open-drain output pin by programmable
P34	1	I/O	Port 34: I/O port Open-drain output pin by programmable
P35 TA6IN	1	I/O Input	Port 35: I/O port Timer A6 input Open-drain output pin by programmable
P36 TA7OUT	1	I/O Output	Port 36: I/O port Timer A7 output Open-drain output pin by programmable
P37 INT4	1	I/O Input	Port 37: I/O port Interrupt request pin 4: Interrupt request pin with programmable rising edge/falling edge levels Open-drain output pin by programmable
P40 to P43 AN8 to AN11 \overline{ADTRG}	4	Input Input Input	Port 40: Pin used to input port Analog input: Pin used to input to AD converter AD Trigger: Signal used to request start of AD conversion
P50 to P57 AN0 to AN7	8	Input Input	Port 5: Pin used to input port Analog input: Pin used to input to AD converter
P60 INT0	1	I/O Input	Port 60: I/O port Interrupt Request pin 0: Interrupt request pin with programmable rising edge/falling edge levels
P61 \overline{CTS} INT1	1	I/O Input Input	Port 61: I/O port Serial data send enable (Clear to send) Interrupt request pin 1: Interrupt request pin with programmable rising edge/falling edge levels
P62 SCOUT INT2	1	I/O Output Input	Port 62: I/O port System clock output: Outputs f_{PPH} or f_s clock Interrupt request pin 2: Interrupt request pin with programmable rising edge/falling edge levels

Table 2.2.2 Pin Names and Functions (2/3)

Pin Name	Number of Pins	I/O	Functions
P70 TA1OUT	1	I/O Output	Port 70: I/O port Timer A1 output
P71 TA3OUT	1	I/O Output	Port 71: I/O port Timer A3 output
P72 TA5OUT	1	I/O Output	Port 72: I/O port Timer A5 output
P73 TB0IN0 INT5	1	I/O Input Input	Port 73: I/O port Timer B0 input 0 Interrupt request pin 5: Interrupt request pin with programmable rising edge/falling edge levels
P74 TB0IN1 INT6	1	I/O Input Input	Port 74: I/O port Timer B0 input 1 Interrupt request pin 6: Interrupt request pin with rising edge levels
P75 TB0OUT0	1	I/O Output	Port 75: I/O port Timer B0 output 0
P76 SCK0 INT3	1	I/O I/O Input	Port 76: I/O port Serial clock I/O 0 Interrupt request pin 3: Interrupt request pin with programmable rising edge/falling edge levels
P80 SO0 SDA0	1	I/O Output I/O	Port 80: I/O port Serial bus interface send data at SIO mode 0. Serial bus interface send/receive data at I ² C mode 0. Open-drain output pin by programmable
P81 SI0 SCL0	1	I/O Input I/O	Port 81: I/O port Serial bus interface receive data at SIO mode 0. Serial bus interface clock I/O data at I ² C mode 0. Open-drain output pin by programmable
P82 TXD	1	I/O Output	Port 82: I/O port Serial send data (UART) Open-drain output pin by programmable
P83 RXD	1	I/O Input	Port 83: I/O port Serial receive data (UART) Open-drain output pin by programmable
P84 SDA1	1	I/O I/O	Port 84: I/O port Serial bus interface send/receive data at I ² C mode 1 N-ch FET open-drain output
P85 SCL1	1	I/O I/O	Port 85: I/O port Serial bus interface clock I/O data at I ² C mode 1 N-ch FET open-drain output
P86 SDA2	1	I/O I/O	Port 86: I/O port Serial bus interface send/receive data at I ² C mode 2 N-ch FET open-drain output
P87 SCL2	1	I/O I/O	Port 87: I/O port Serial bus interface clock I/O data at I ² C mode 2 N-ch FET open-drain output

Table 2.2.3 Pin Names and Functions (3/3)

Pin Name	Number of Pins	I/O	Functions
ALE	1	Output	Address latch enable Can be disabled to reduce noise.
$\overline{\text{NMI}}$	1	Input	Non-maskable interrupt request pin: Interrupt request pin with programmable falling edge level or with both edge levels programmable
AM0 to AM1	2	Input	Address mode: The Vcc pin should be connected.
EMU0, EMU1	1	Output	Test pins: Open pins
$\overline{\text{RESET}}$	1	Input	Reset: Initializes TMP91CW18A (with pull-up resistor).
VREFH	1	Input	Pin for reference voltage input to AD converter (H)
VREFL	1	Input	Pin for reference voltage input to AD converter (L)
AVCC	1		Power supply pin for AD converter
AVSS	1		GND pin for AD converter (0 V)
X1/X2	2	I/O	High-frequency oscillator connection pins
DVCC	3		Power supply pins (All Vcc pins should be connected with the power supply pin.)
DVSS	2		GND pins (All pins should be connected with GND (0V).)

3. Operation

This section describes the basic components, functions and operation of the TMP91CW18A.

Notes and restrictions which apply to the various items described here are outlined in Section 7. Precautions and restrictions at the end of this databook.

3.1 CPU

The TMP91CW18A incorporates a high-performance 16-bit CPU (the 900/L1 CPU). For a description of this CPU's operation, please refer to the section of this databook which describes the TLCS-900/L1 CPU.

The following sub-sections describe functions peculiar to the CPU used in the TMP91CW18A. These functions are not covered in the section devoted to the TLCS-900/L1 CPU.

3.1.1 Reset

When resetting the TMP91CW18A microcontroller, ensure that the power supply voltage is within the operating voltage range and that the internal high-frequency oscillator has stabilized. Then set the $\overline{\text{RESET}}$ input to low level at least for 10 system clocks (13 μs at 25 MHz).

Thus, when turn on the switch, be set to the power supply voltage is within the operating voltage range, and that the internal high-frequency oscillator has stabilized. Then hold the $\overline{\text{RESET}}$ input to low level at least for 10 system clocks.

Clock gear is initialized 1/16 mode by reset operation. It means that the system clock mode f_{SYS} is set to $f_c/32$ ($= f_c/16 \times 1/2$).

When the reset is accept, the CPU:

- Sets the program counter (PC) as follows in accordance with the reset vector stored at address FFFF00H to FFFF02H:

PC<0:7>	←	Data in location FFFF00H
PC<8:15>	←	Data in location FFFF01H
PC<16:23>	←	Data in location FFFF02H
- Sets the stack pointer (XSP) to 100H.
- Sets bits <IFF0:2> of the status register (SR) to 111 (thereby setting the interrupt level mask register to level 7).
- Sets the <MAX> bit of the status register to 1 (MAX mode).

Note: As this product does not support MIN mode, do not write a 0 to the <MAX> bit.

- Clears bits <RFP0:2> of the status register to 000 (thereby selecting register bank 0).

When the reset is cleared, the CPU starts executing instructions according to the program counter settings. CPU internal registers not mentioned above do not change when the reset is cleared.

When the reset is accepted, the CPU sets internal I/O, ports and other pins as follows.

- Initializes the internal I/O registers.
- Sets the port pins including the pins that also act as internal I/O, to general-purpose input or output port mode.
- Sets the ALE pin to High-Z.

Note: By resetting, register in CPU except program counter (PC), status register (SR), and stack

Pointer (XSP) and the data in internal RAM are not changed.

Figure 3.1.1 shows the timing of a reset for the TMP91CW18A.

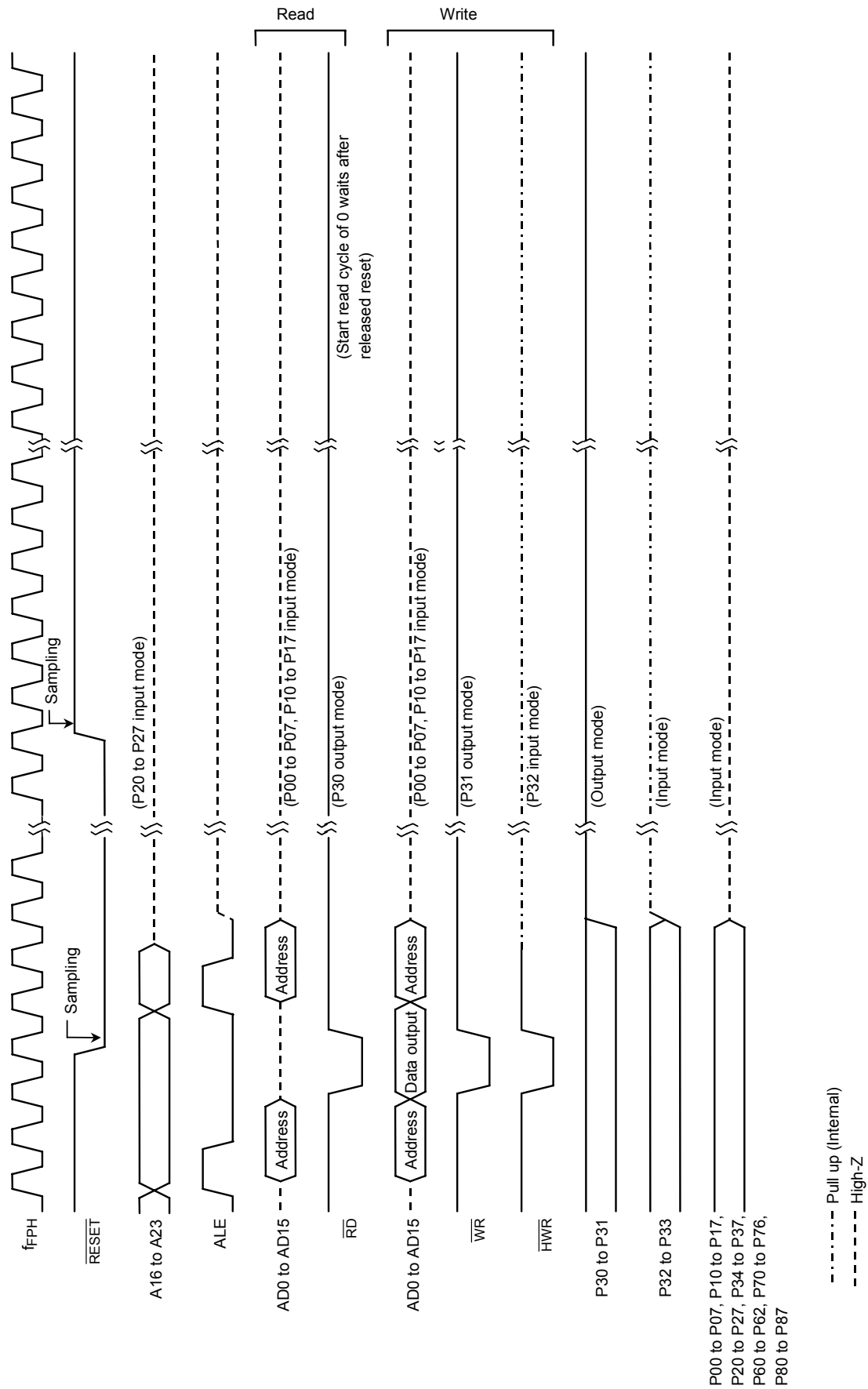


Figure 3.1.1 TMP91CW18A Reset Timing Example

3.2 Memory Map

Figure 3.2.1 is a memory map of the TMP91CW18A.

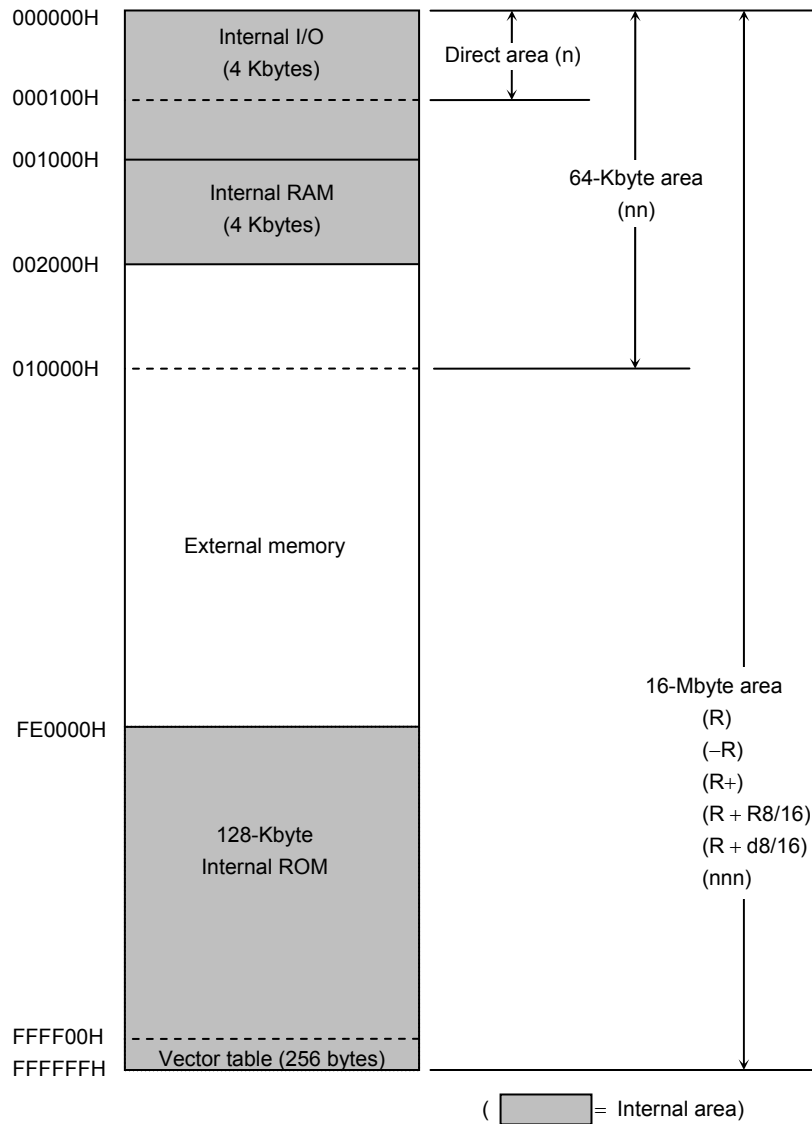


Figure 3.2.1 Memory Map

3.3 Standby Function/Noise Reducing Circuit

The TMP91CW18A contains (1) a clock gearing system, (2) a standby controller and (3) a noise reduction circuit. It is used for low-power and low-noise systems.

This chapter is organized as follows.

3.3.1 Block Diagram of System Clock

3.3.2 SFR

3.3.3 System Clock Controller

3.3.4 Prescaler Clock Controller

3.3.5 Noise Reduction Circuits

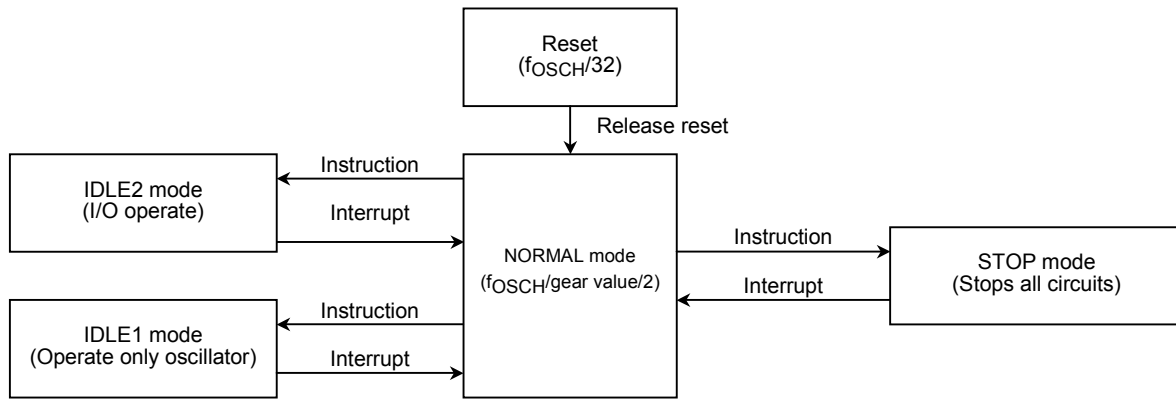
3.3.6 Standby Controller

3.3.1 Block Diagram of System Clock

The clock operating modes are as follows: Single clock mode (X1, X2 pins only)

Figure 3.3.1 shows a transition figure.

The clock frequency input from the X1 and X2 pins is called f_c . The clock frequency selected by SYSCR1<SYSCK> is called the system clock f_{PPH} . The system clock f_{SYS} is defined as the divided clock of f_{PPH} and one cycle of f_{SYS} is regard to as one state.



(a) Single clock mode transition figure

Figure 3.3.1 System Clock Block Diagram

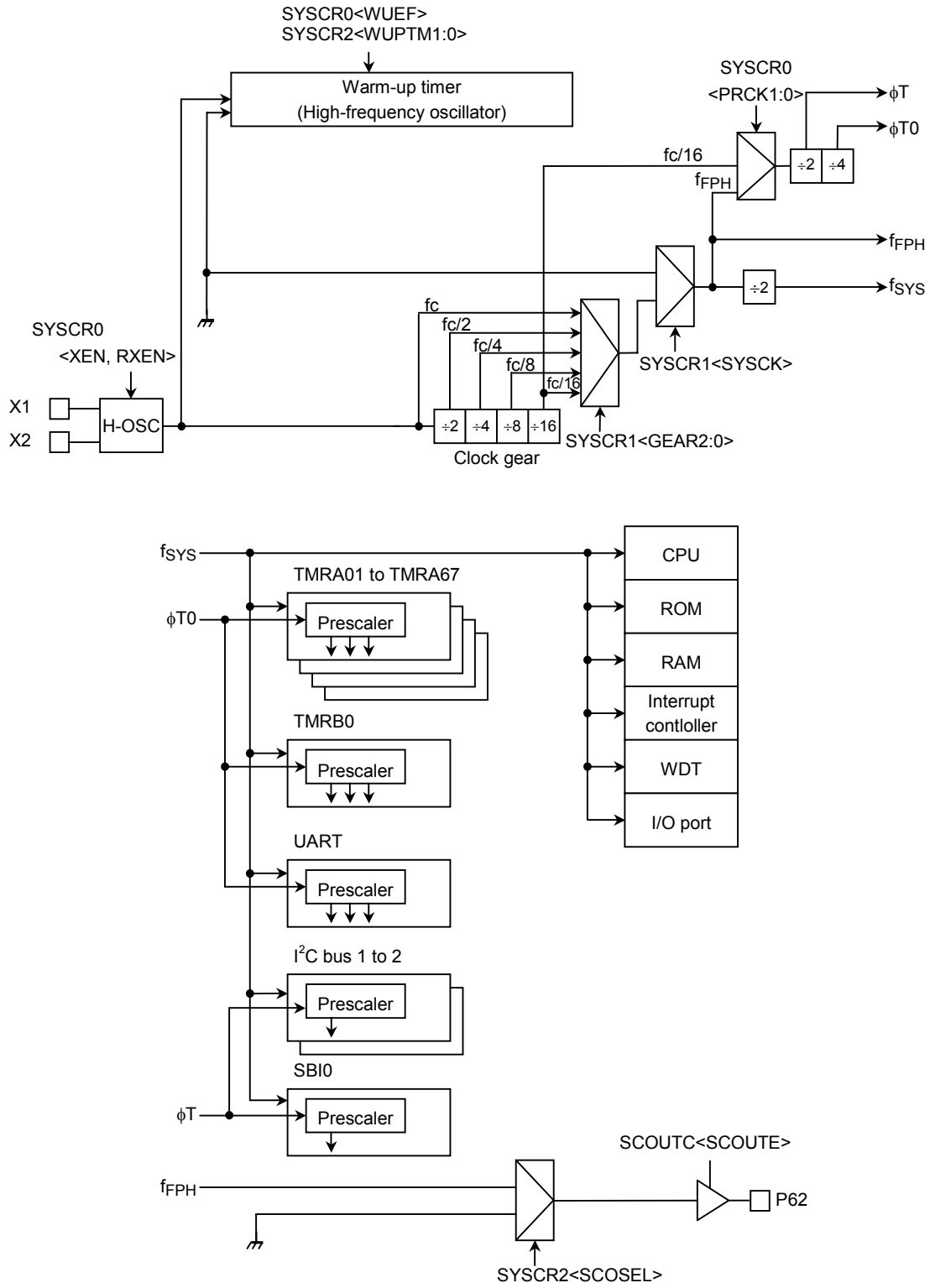


Figure 3.3.2 Block Diagram of System Clock

3.3.2 SFR

	7	6	5	4	3	2	1	0		
SYSCR0 (00E0H)	Bit symbol	XEN	–	RXEN	–	–	WUEF	PRCK1	PRCK0	
	Read/Write	R/W								
	After reset	1	0	1	0	0	0	0	0	
	Function	High-frequency oscillator (fc) 0: Stop 1: Oscillation	Always write 0	High-frequency oscillator (fc) after release of Stop mode 0: Stop 1: Oscillation	Always write 0	Always write 0	Warm-up timer Write 0: Don't care Write 1: Start timer Read 0: End warm-up Read 1: Do not end warm-up	Select prescaler clock 00: f _{FPH} 01: Reserved 10: fc/16 11: Reserved		
SYSCR1 (00E1H)	Bit symbol	–	–	–	–	–	GEAR2	GEAR1	GEAR0	
	Read/Write	–							R/W	
	After reset	–	–	–	–	0	1	0	0	
	Function	–	–	–	–	Always write 0	Select gear value of high frequency (fc) 000: fc 001: fc/2 010: fc/4 011: fc/8 100: fc/16 101: (Reserved) 110: (Reserved) 111: (Reserved)			
SYSCR2 (00E2H)	Bit symbol	–	SCOSEL	WUPTM1	WUPTM0	HALTM1	HALTM0	–	DRVE	
	Read/Write	–	R/W	R/W	R/W	R/W	R/W	–	R/W	
	After reset	–	0	1	0	1	1	–	0	
	Function	–	0: fs 1: f _{FPH}	Warm-up timer 00: Reserved 01: 2 ⁸ /inputted frequency 10: 2 ¹⁴ 11: 2 ¹⁶		HALT mode 00: Reserved 01: STOP mode 10: IDLE1 mode 11: IDLE2 mode		–	Pin state control in STOP mode 0: I/O off 1: Remains the state before halt	

Note 1: SYSCR1<bit7:4> and SYSCR2<bit7, 1> are read as undefined-value.

Note 2: When using the built-in I²C bus or the I²C bus/SIO, set the select prescaler clock register SYSCR0<PRCK1:0> to 00 (f_{FPH}).

Figure 3.3.3 SFR for System Clock

	7	6	5	4	3	2	1	0	
EMCCR0 (00E3H)	Bit symbol	PROTECT	-	-	-	ALEEN	EXTIN	DRVOSCH	-
	Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	After reset	0	0	1	0	0	0	1	1
	Function	Protect flag 0: OFF 1: ON	Always write 0	Always write 1	Always write 0	ALE pin output control 0: High-Z output 1: ALE output	1: External clock	fc oscillator driving ability 1: Normal 0: Weak	Always write 1
EMCCR1 (00E4H)	Bit symbol	Writing 1FH turns protections off. Writing any value other than 1FH turns protection on.							
	Read/Write								
	After reset								
	Function								

Note: In case restarting the oscillator in the stop oscillation state (e.g. Restart the oscillator in STOP mode), set EMCCR0<DRVOSCH>, <DRVOSCL>="1".

Figure 3.3.4 SFR for Noise Reduction

3.3.3 System Clock Controller

The system clock controller generates the system clock signal (f_{SYS}) for the CPU core and internal I/O. It contains gear circuit for high-frequency (f_c) operation. The register SYSCR0<XEN> control enabling and disabling of each oscillator and SYSCR1<GEAR0:2> sets the high-frequency clock gear to either 1, 2, 4, 8 or 16 (f_c , $f_c/2$, $f_c/4$, $f_c/8$ or $f_c/16$). These functions can reduce the power consumption of the equipment in which the device is installed.

The combination of settings <XEN> = 1, <XTEN> = 0, <SYSCK> = 0 and <GEAR0:2> = 100 will cause the system clock (f_{SYS}) to be set to $f_c/32$ ($f_c/16 \times 1/2$) after a reset.

For example, f_{SYS} is set to 0.78 MHz when the 25-MHz oscillator is connected to the X1 and X2 pins.

(1) Switching from NORMAL mode to SLOW mode

When the resonator is connected to the X1 and X2 pins, or to the XT1 and XT2 pins, the warm-up timer can be used to change the operation frequency after stable oscillation has been attained.

The warm-up time can be selected using SYSCR2<WUPTM0:1>. This warm-up timer can be programmed to start and stop as shown in the following examples 1 and 2.

Table 3.3.1 shows the warm-up time.

Note 1: When using an oscillator (other than a resonator) with stable oscillation, a warm-up timer is not needed.

Note 2: The warm-up timer is operated by an oscillation clock. Hence, there may be some variation in warm-up time.

Table 3.3.1 Warm-up Time

Warm-up Time SYSCR2<WUPTM1:0>	Change to NORMAL Mode
01 ($2^0/\text{frequency}$)	10 (μs)
10 ($2^4/\text{frequency}$)	0.655 (ms)
11 ($2^6/\text{frequency}$)	2.621 (ms)

at $f_{OSCH} = 25 \text{ MHz}$

(2) Clock gear controller

When the high-frequency clock f_c is selected by setting $\text{SYSCR1}\langle\text{SYSCK}\rangle = 0$, f_{FPH} is set according to the contents of the clock gear select register $\text{SYSCR1}\langle\text{GEAR0:2}\rangle$ to either f_c , $f_c/2$, $f_c/4$, $f_c/8$ or $f_c/16$. Using the clock gear to select a lower value of f_{FPH} reduces power consumption.

Example:

Changing to a high-frequency gear

```
SYSCR1 EQU 00E1H
LD (SYSCR1), XXXX0000B ; Changes fSYS to fc/2.
LD (SYSCR1), XXXX0100B ; Changes fSYS to fc/32.
```

X: Don't care

(Changing to high-frequency clock gear)

To change the clock gear, write the appropriate value to the $\text{SYSCR1}\langle\text{GEAR0:2}\rangle$ register. The value of f_{FPH} will not change until a period of time equal to the warm-up time has elapsed from the point at which the register is written to.

There is a possibility that the instruction immediately following the instruction which changes the clock gear will be executed before the new clock setting comes into effect. To ensure that this does not happen, insert a dummy instruction (to execute a write cycle) as follows.

Example:

```
SYSCR1 EQU 00E1H
LD (SYSCR1), XXXX0001B ; Changes fSYS to fc/4.
LD (DUMMY), 00H ; Dummy instruction
```

Instruction to be executed after clock gear has changed

(3) Internal clock pin output function

The P62/SCOUT/INT2 pin outputs an internal clock: f_{FPH} or f_s .

The following combination of settings – port 6 control register $\text{P6CR}\langle\text{P62C}\rangle = 1$ and $\text{P6FC}\langle\text{P62F}\rangle = 1$ – specifies that a clock signal will be output on the P62/SCOUT/INT2 pin. The setting of $\text{SYSCR2}\langle\text{SCOSEL}\rangle$ determines which clock is output.

Table 3.3.2 shows the pin state of the P62/SCOUT/INT2 pin when it is selected for clock output in the different operation modes.

Table 3.3.2 SCOUT Pin States in Different Operation Modes

Operation Mode SCOUT Select	NORMAL, SLOW	HALT mode		
		IDLE2	IDLE1	STOP
$\langle\text{SCOSEL}\rangle = 0$	Outputs low.			
$\langle\text{SCOSEL}\rangle = 1$	Outputs f_{FPH} clock.		Fixed to 0 or 1.	

3.3.4 Prescaler Clock Controller

For the internal I/O (TMRA01 to TMRA67, TMRB0, TMRB1, SIO0, SIO1 and SBI) there is a prescaler which can divide the clock. The ϕT clock input to the prescaler is either the clock f_{PPH} divided by 2 or the clock $f_c/16$ divided by 2. The setting of the SYSCR0 <PRCK0:1> register determines which clock signal is input.

The $\phi T0$ clock input to the prescaler is either the clock f_{PPH} divided by 4 or the clock $f_c/16$ divided by 4. The setting of the SYSCR0 <PRCK0:1> register determines which clock signal is input.

3.3.5 Noise Reduction Circuits

Noise reduction circuits are built in, allowing implementation of the following features.

- (1) Reduced driveability for high-frequency oscillator
- (2) Single drive for high-frequency oscillator
- (3) Disabling of ALE pin output
- (4) Protection of register contents

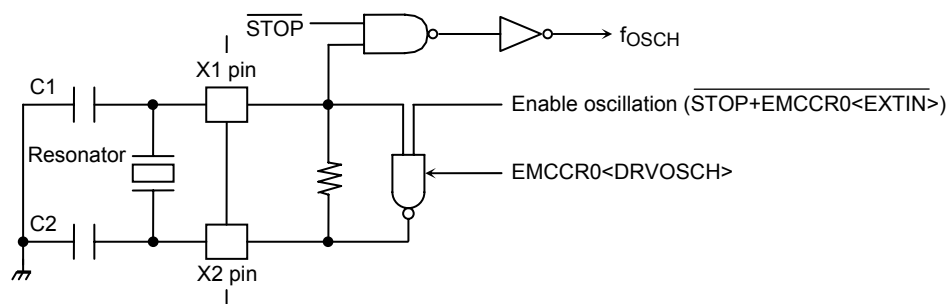
The above functions are performed by making the appropriate settings in the EMCCR0 and EMCCR1 registers.

- (1) Reduced driveability for high-frequency oscillator

(Purpose)

Reduces noise and power for oscillator when a resonator is used.

(Block diagram)



(Setting method)

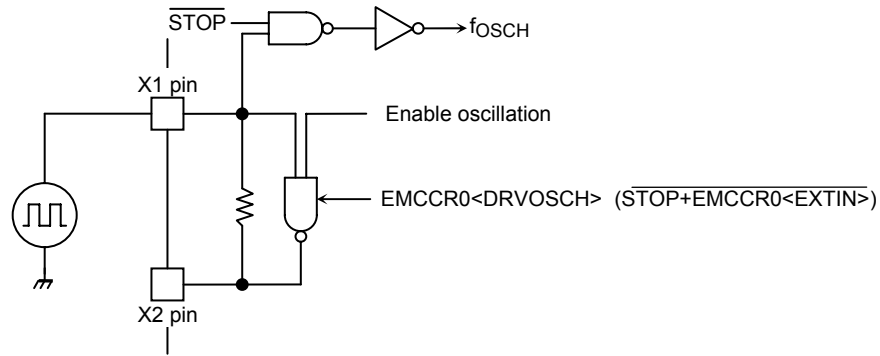
The driveability of the oscillator is reduced by writing 0 to the EMCCR0 <DRVOSCH> register. On a reset, <DRVOSCH> is initialized to 1 and the oscillator starts oscillation by normal driveability when the power supply is on.

(2) Single drive for high-frequency oscillator

(Purpose)

Not need twin-drive and protect mistake-operation by inputted noise to X2 pin when the external oscillator is used.

(Block diagram)



(Setting method)

When a 1 is written to the EMCCR0<EXTIN>, the oscillator is disabled and is operated as a buffer. The X2 pin always outputs a 1.

<EXTIN> is initialized to 0 by a reset.

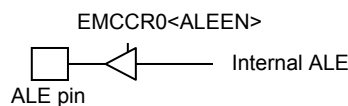
Note: Do not write EMCCR0<EXTIN> = "1" when using external resonator.

(3) Disabling ALE pin output

(Purpose)

If the CPU does not access any external area, output of the ALE pulse can be disabled, thereby reduction noise.

(Block diagram)



(Setting method)

Writing 0 to the EMCCR0<ALEEN> register sets the ALE pin to high-impedance. <ALEEN> is initialized to 0 by a reset.

If the CPU needs to access an external area, 1 must first be written to <ALEEN>.

(4) Protection of register contents

(Purpose)

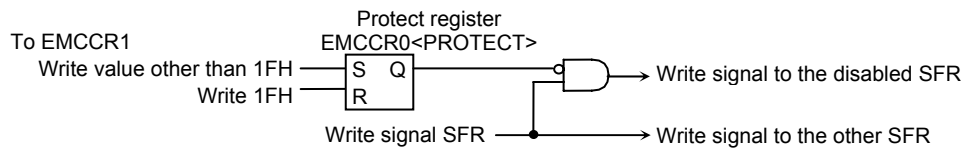
An item for mistake operation by inputted noise.

To execute the program certainty which is occurred mistake operation, the protect register can be disabled write-operation for the specific SFR.

Write disabled SFRs

- | |
|--|
| <p>1. CS/WAIT controller
 B0CS, B1CS, B2CS, B3CS, BEXCS,
 MSAR0, MSAR1, MSAR2, MSAR3,
 MAMR0, MAMR1, MAMR2, MAMR3</p> <p>2. Clock gear
 SYSCR0, SYSCR1, SYSCR2, EMCCR0</p> |
|--|

(Block diagram)



(Setting method)

Writing any value other than 1FH to the EMCCR1 register turns on protection, thereby preventing the CPU from writing to the specific SFR.

Writing 1FH to EMCCR1 turns off protection.

The protection status is set in EMCCR0<PROTECT>.

Resetting initializes the protection status to OFF.

3.3.6 Standby Controller

(1) HALT modes

When the HALT instruction is executed, the operating mode switches to IDLE2, IDLE1 or STOP mode, depending on the contents of the SYSCR2<HALTM1:0> register.

The subsequent actions performed in each mode are as follows.

1. IDLE2: The CPU only is halted.

In IDLE2 mode internal I/O operations can be performed by setting the following registers.

Table 3.3.3 shows the registers of setting operation during IDLE2 mode.

Table 3.3.3 Registers of Setting Operation during IDLE2 Mode

Internal I/O	SFR
TMRA01	TA01RUN<I2TA01>
TMRA23	TA23RUN<I2TA23>
TMRA45	TA45RUN<I2TA45>
TMRA67	TA67RUN<I2TA67>
TMRB0	TB0RUN<I2TB0>
SIO0 (I ² C bus/SIO)	SBI0BR00<I2SBI0>
SIO1 (I ² C bus)	SBI0BR01<I2SBI0>
SIO2 (I ² C bus)	SBI0BR02<I2SBI0>
UART	SC0MOD1<I2SO>
AD converter	ADM0D1<I2AD>
WDT	WDM0D<I2WDT>

2. IDLE1: Only the oscillator continue to operate.

3. STOP: All internal circuits stop operating.

The operation of each of the different HALT modes is described in Table 3.3.4.

Table 3.3.4 I/O Operation during HALT Modes

HALT Mode		IDLE2	IDLE1	STOP
SYSCR2<HALTM1:0>		11	10	01
Block	CPU	Stop		
	I/O ports	Keep the state when the HALT instruction was executed.		See Table 3.3.7, Table 3.3.8
	TMRA, TMRB	Available to select operation block.		Stop
	SIO, SBI			
	AD converter			
	WDT			
Interrupt controller	Operation			

(2) How to clear a HALT mode

The halt state can be cleared by a reset or by an interrupt request. The combination of the value in <IFF0:2> of the interrupt mask register and the current HALT mode determine in which ways the HALT mode may be cleared. The details associated with each type of halt state clearance are shown in Table 3.3.5.

- Clearance by interrupt request

Whether or not the HALT mode is cleared and subsequent operation depends on the status of the generated interrupt. If the interrupt request level set before execution of the HALT instruction is greater than or equal to the value in the interrupt mask register, the following sequence takes place. The HALT mode is cleared, the interrupt is then processed and the CPU then resumes execution starting from the instruction following the HALT instruction. If the interrupt request level set before execution of the HALT instruction is less than the value in the interrupt mask register, the HALT mode is not cleared. (If a non-maskable interrupt is generated, the HALT mode is cleared and the interrupt processed, regardless of the value in the interrupt mask register.)

However, for $\overline{\text{NMI}}$ and INT0 to INT4 interrupts only, even if the interrupt request level set before execution of the HALT instruction is less than the value in the interrupt mask register, the HALT mode is cleared. In this case, the interrupt is not processed and the CPU resumes execution starting from the instruction following the HALT instruction. The interrupt request flag remains set to 1.

Note: Usually, interrupts can release all halts status. However, the interrupts = ($\overline{\text{NMI}}$, INT0 to INT4) which can release the HALT mode may not be able to do so if they are input during the period CPU is shifting to the HALT mode (for about 5 clocks of f_{PPH}) with IDLE1 or STOP mode (IDLE2 is not applicable to this case). (In this case, an interrupt request is kept on hold internally.)

If another interrupt is generated after it has shifted to HALT mode completely, halt status can be released without difficulty. The priority of this interrupt is compare with that of the interrupt kept on hold internally, and the interrupt with higher priority is handled first followed by the other interrupt.

- Clearance by reset

Any halt state can be cleared by a reset.

When STOP mode is cleared by a RESET signal, sufficient time (at least 3 ms) must be allowed after the reset for the operation of the oscillator to stabilize.

When a HALT mode is cleared by resetting, the contents of the internal RAM remain the same as they were before execution of the HALT instruction. However, all other settings are re-initialized. (Clearance by an interrupt affects neither the RAM contents nor any other settings – the state which existed before the HALT instruction was executed is retained.)

Table 3.3.5 Source of Halt State Clearance and Halt Clearance Operation

Status of Received Interrupt		Interrupt Enabled (Interrupt level) ≥ (Interrupt mask)			Interrupt Disabled (Interrupt level) < (Interrupt mask)			
		HALT Mode	IDLE2	IDLE1	STOP	IDLE2	IDLE1	STOP
Source of Halt State Clearance	Interrupt	NMI	◆	◆	◆*1	—	—	—
		INTWDT	◆	×	×	—	—	—
		INT0 to INT4 (Note 1)	◆	◆	◆*1	○	○	○*1
		INT5, INT6	◆ (Note 2)	×	×	×	×	×
		INTTA0 to INTTA7	◆	×	×	×	×	×
		INTTB00, 01, 10, 11, OF0, OF1	◆	×	×	×	×	×
		INTRX0, INTRX1, TX0, TX1	◆	×	×	×	×	×
		INTS2	◆	×	×	×	×	×
		INTAD	◆	×	×	×	×	×
	RESET	Reset initializes the LSI						

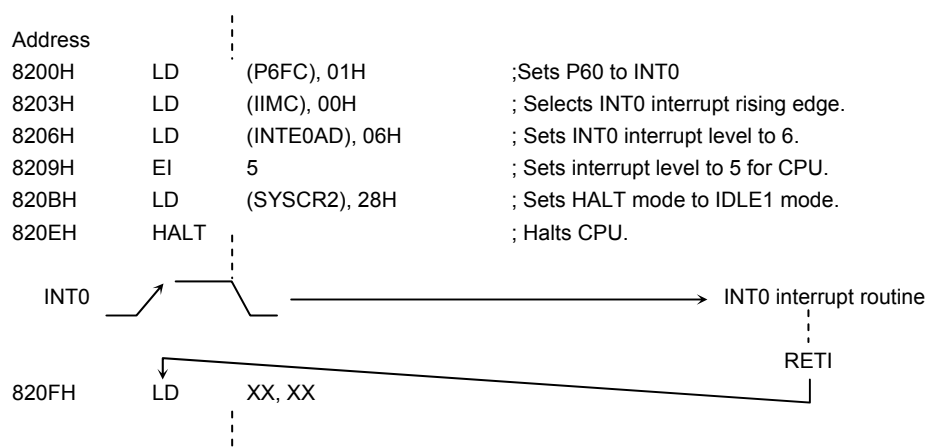
- ◆: After clearing the HALT mode, CPU starts interrupt processing. (RESET initializes the microcontroller.)
- : After clearing the HALT mode, CPU resumes executing starting from instruction following the HALT instruction.
- ×: Cannot be used to clear the HALT mode.
- : The priority level (interrupt request level) of non-maskable interrupts is fixed to 7, the highest priority level. There is not this combination type.
- *1: The HALT mode is cleared when the warm-up time has elapsed.

Note 1: When the HALT mode is cleared by an INT0 to INT4 interrupt of the level mode in the interrupt enabled status, hold this level until starting interrupt processing. Changing level before holding level, interrupt processing is correctly started.

Note 2: If one of the external interrupts INT5 and INT6 are generated in IDLE2 mode, TB0RUN<I2TB0> is set to 1.

(Example – clearing IDLE1 mode)

An INT0 interrupt clears the halt state when the device is in IDLE1 mode.



(3) Operation

1. IDLE2 mode

In IDLE2 mode only specific internal I/O operations, as designated by the IDLE2 setting register, can take place. Instruction execution by the CPU stops.

Figure 3.3.5 illustrates an example of the timing for clearance of the IDLE2 mode halt state by an interrupt.

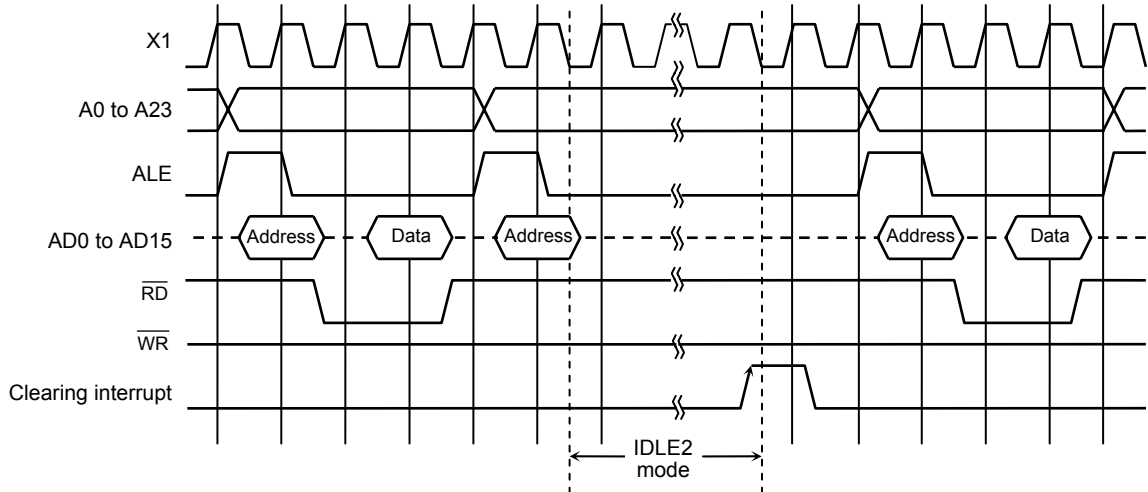


Figure 3.3.5 Timing Chart for IDLE2 Mode Halt State Cleared by Interrupt

2. IDLE1 mode

In IDLE1 mode, only the internal oscillator and the RTC continue to operate. The system clock in the MCU stops.

In the halt state, the interrupt request is sampled asynchronously with the system clock; however, clearance of the halt state (e.g., restart of operation) is synchronous with it.

Figure 3.3.6 illustrates the timing for clearance of the IDLE1 mode halt state by an interrupt.

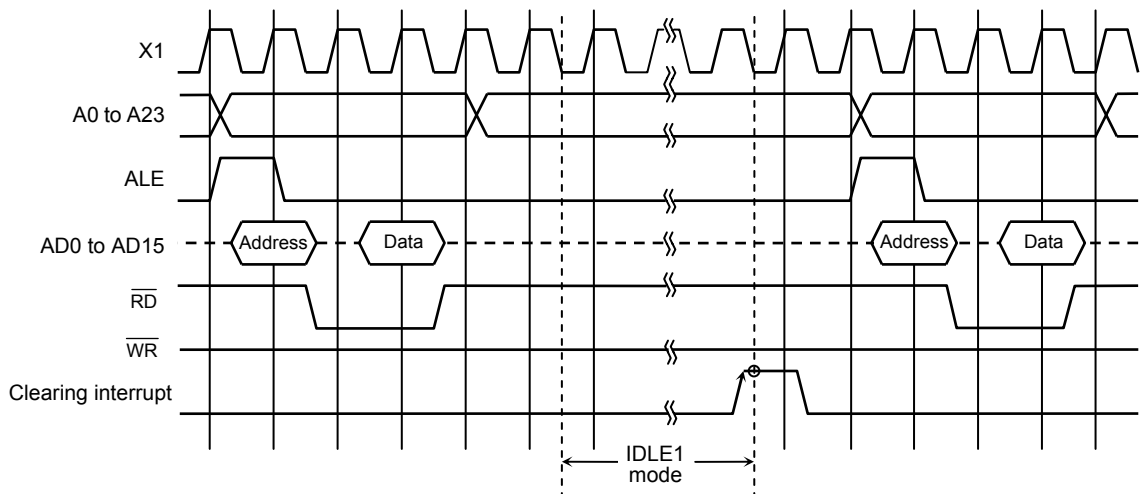


Figure 3.3.6 Timing Chart for IDLE1 Mode Halt State Cleared by Interrupt

3. STOP mode

When STOP mode is selected, all internal circuits stop, including the internal oscillator pin status in STOP mode depends on the settings in the SYSCR2<DRVE> register. Table 3.3.7 and Table 3.3.8 summarizes the state of these pins in STOP mode.

After STOP mode has been cleared system clock output starts when the warm-up time has elapsed, in order to allow oscillation to stabilize. After STOP mode has been cleared, either NORMAL mode or SLOW mode can be selected using the SYSCR0<RSYSCK> register. Therefore, <RSYSCK>, <RXEN> and <RXTEN> must be set. See the sample warm-up time in Table 3.3.6.

Figure 3.3.7 illustrates the timing for clearance of the STOP mode halt state by an interrupt.

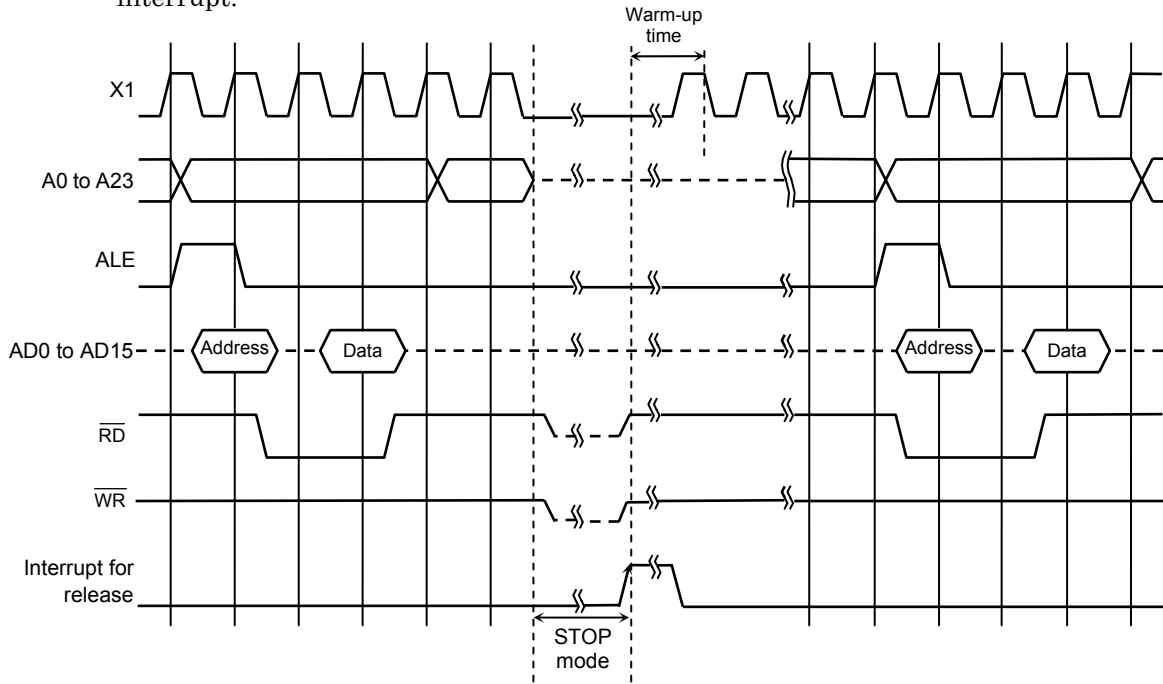


Figure 3.3.7 Timing Chart for STOP Mode Halt State Cleared by Interrupt

Table 3.3.6 Sample Warm-up Time after Clearance of STOP Mode

at f_{OSCH} = 25 MHz

SYSCR0 <RSYSCK>	SYSCR2<WUPTM1:0>		
	01 (2 ⁸)	10 (2 ¹⁴)	11 (2 ¹⁶)
0 (fc)	10 μs	0.655 ms	2.621 ms

Table 3.3.7 Input Buffer State Table

Port Name	Input Function Name	During Reset	Input Buffer State							
			When the CPU is Operating		In HALT Mode (IDLE1/IDLE2)		In HALT Mode (STOP)			
			When Used as Function Pin	When Used as Input Port	When Used as Function Pin	When Used as Input Port	<DRVE> = 0 When Used as Function Pin	<DRVE> = 0 When Used as Input Port	<DRVE> = 1 When Used as Function Pin	<DRVE> = 1 When Used as Input Port
P00 to P07	AD0 to AD7		–	OFF	–	OFF	–	OFF	–	OFF
P10 to P17	AD8 to AD15		–	OFF	–	OFF	–	OFF	–	OFF
P20 to P27	AD16 to AD23		–	OFF	–	OFF	–	OFF	–	OFF
P30	–		–	ON	–	OFF	–	OFF	–	OFF
P31	–		–	ON	–	OFF	–	OFF	–	OFF
P32 (Note 1)	–		–	ON	–	OFF	–	OFF	–	OFF
P33 (Note 1)	WAIT		ON	ON	OFF	OFF	OFF	OFF	OFF	OFF
P34	–		–	ON	–	ON	–	OFF	–	ON
P35	TA6IN		ON	ON	ON	ON	OFF	OFF	ON	ON
P36	–		–	ON	–	ON	OFF	OFF	–	ON
P37	INT4		ON	ON	ON	ON	ON	OFF	ON	ON
P40 to P42 (Note 2)	AN8 to AN10		*	ON upon port read	*	OFF	*	OFF	*	OFF
P43 (Note 2)	AN11 ADTRG		* ON	ON upon port read	* ON	OFF	* ON	OFF	* ON	OFF
P50 to P57 (Note 2)	AN0 to AN7		*	ON upon port read	*	OFF	OFF	OFF	ON	OFF
P60	INT0		ON	ON	ON	ON	* ON	ON	* ON	ON
P61	INT1 CTS		ON ON	ON	ON	ON	ON	ON	ON	ON
P62	INT2		ON	ON	ON	ON	ON	ON	ON	ON
P70	–		–	ON	–	ON	–	OFF	–	ON
P71	–		–	ON	–	ON	–	OFF	–	ON
P72	–		–	ON	–	ON	–	OFF	–	ON
P73	TB0IN0 INT5		ON	ON	ON	ON	ON	OFF	ON	ON
P74	TB0IN1 INT6		ON	ON	ON	ON	ON	OFF	ON	ON
P75	–		–	ON	–	ON	ON	OFF	–	ON
P76	INT3		ON	ON	ON	ON	ON	ON	ON	ON
P80	SDA0		ON	ON	ON	ON	OFF	OFF	ON	ON
P81	SIO SCL0		ON	ON	ON	ON	OFF	OFF	ON	ON
P82	–		ON	ON	ON	ON	OFF	OFF	ON	ON
P83	RXD0		ON	ON	ON	ON	OFF	OFF	ON	ON
P84	SDA1		ON	ON	ON	ON	OFF	OFF	ON	ON
P85	SCL1		ON	ON	ON	ON	OFF	OFF	ON	ON
P86	SDA2		ON	ON	ON	ON	OFF	OFF	ON	ON
P87	SCL2		ON	ON	ON	ON	OFF	OFF	ON	ON
NMI	–	ON	ON	–	ON	–	ON	–	ON	–
RESET	–	ON	ON	–	ON	–	ON	–	ON	–
AM0, AM1	–	ON	ON	–	ON	–	ON	–	ON	–
X1	–	ON	ON	–	ON	–	OFF	–	OFF	–

ON: The buffer is always turned on. A current flows through the input buffer if the input pin is not driven.

OFF: The buffer is always turned off.

–: Not applicable

Note 1: Port having a pull-up/pull-down resistor.

Note 2: AIN input does not cause a current to flow through the buffer.

*: AIN input is always enable.

Table 3.3.8 Output Buffer State Table

Port Name	Output Function Name	Output Buffer State								
		During reset	When the CPU is Operating		In HALT Mode (IDLE1/IDLE2)		In HALT Mode (STOP)			
			When Used as Function Pin	When Used as Output Port	When Used as Function Pin	When Used as Output Port	<DRVE> = 0 When Used as Function Pin	<DRVE> = 0 When Used as Output Port	<DRVE> = 1 When Used as Function Pin	<DRVE> = 1 When Used as Output Port
P00 to P07	AD0 to AD7	OFF	ON upon external write	ON	OFF	ON	OFF	OFF	OFF	ON
P10 to P17	AD8 to AD15	OFF		ON	OFF	ON	OFF	OFF	OFF	ON
	A8 to A15	OFF	ON	ON	ON	ON	OFF	OFF	ON	ON
P20 to P27	A0 to A7	OFF	ON	ON	ON	ON	OFF	OFF	ON	ON
	A16 to A23	OFF	ON	ON	ON	ON	OFF	OFF	ON	ON
P30	RD		ON	ON	ON	ON	OFF	OFF	ON	ON
P31	WR		ON	ON	ON	ON	OFF	OFF	ON	ON
P32 (Note 1)	HWR		ON	ON	ON	ON	OFF	OFF	ON	ON
P33 (Note 1)	–		–	ON	–	ON	–	OFF	–	ON
P34	–		–	ON	–	ON	–	OFF	–	ON
P35	–		–	ON	–	ON	–	OFF	–	ON
P36	TA7OUT		ON	ON	ON	ON	OFF	OFF	ON	ON
P37	–		–	ON	–	ON	–	OFF	–	ON
P40 to P7	–		–	–	–	–	–	–	–	–
P43	–		–	–	–	–	–	–	–	–
P50 to P7	–		–	–	–	–	–	–	–	–
P60	–		ON	ON	ON	ON	ON	OFF	ON	ON
P61	–		–	ON	–	ON	–	OFF	–	ON
P62	SCOUT		ON	ON	ON	ON	ON	OFF	ON	ON
P70	TA1OUT		ON	ON	ON	ON	ON	OFF	ON	ON
P71	TA3OUT		ON	ON	ON	ON	ON	OFF	ON	ON
P72	TA5OUT		ON	ON	ON	ON	ON	OFF	ON	ON
P73	–		–	ON	–	ON	–	OFF	–	ON
P74	–		–	ON	–	ON	–	OFF	–	ON
P75	TB0OUT0		ON	ON	ON	ON	ON	OFF	ON	ON
P76	SCK0		ON	ON	ON	ON	ON	OFF	ON	ON
P80	SDA0		ON	ON	ON	ON	ON	OFF	ON	ON
P81	SO0		ON	ON	ON	ON	ON	OFF	ON	ON
	SCL0		ON	ON	ON	ON	ON	OFF	ON	ON
P82	TXD0		ON	ON	ON	ON	ON	OFF	ON	ON
P83	–		ON	ON	ON	ON	ON	OFF	ON	ON
P84	SDA1		ON	ON	ON	ON	ON	OFF	ON	ON
P85	SCL1		ON	ON	ON	ON	ON	OFF	ON	ON
P86	SDA2		ON	ON	ON	ON	ON	OFF	ON	ON
P87	SCL2		ON	ON	ON	ON	ON	OFF	ON	ON
ALE	–	OFF	ON	–	ON	–	ON	–	ON	–
X2	–	ON	ON	–	ON	–	“H” level output	–	“H” level output	–

ON: A current flows through the output buffer since the buffer is always turned on.

OFF: The buffer is always turned off.

–: Not applicable

Note 1: Port having a pull-up/pull-down resistor.

3.4 Interrupts

Interrupts are controlled by the CPU's interrupt mask register <IFF2:0> (Bits 12 to 14 of the status register) and by the built-in interrupt controller.

The TMP91CW18A has a total of 38 interrupts divided into the following five types:

- Interrupts generated by CPU: 9 sources
(Software interrupts, illegal instruction interrupt)
- Internal I/O interrupts: 21 sources
- External interrupts: 8 sources
Interrupts on external pins ($\overline{\text{NMI}}$ and INT0 to INT6)

A fixed individual interrupt vector number is assigned to each interrupt source.

Any one of 6 levels of priority can also be assigned to each maskable interrupt. Non-maskable interrupts have a fixed priority level of 7, the highest level.

When an interrupt is generated, the interrupt controller transmits the interrupt source's priority value to the CPU. When more than one interrupt are generated simultaneously, the interrupt controller sends the priority value of the interrupt with the highest priority to the CPU. (The highest priority level is 7, the level used for non-maskable interrupts.)

The CPU compares the interrupt priority level which it receives with the value held in the CPU's interrupt mask register <IFF2:0>. If the priority level of the interrupt is greater than or equal to the value in the interrupt mask register, the CPU accepts the interrupt.

However, software interrupts and illegal instruction interrupts generated by the CPU are processed irrespective of the value in <IFF2:0>.

The value in the interrupt mask register <IFF2:0> can be changed using the EI instruction; the command EI n sets the contents of <IFF2:0> to n. For example, the command EI 3 enables the acceptance of all non-maskable interrupts and of maskable interrupts whose priority level, as set in the interrupt controller, is 3 or higher. The commands EI and EI 0 enable the acceptance of all non-maskable interrupts and of maskable interrupts with a priority level of 1 or above (hence both are equivalent to the command EI 1).

The DI instruction (which sets <IFF2:0> to 7) is exactly equivalent to the EI 7 instruction. The DI instruction is used to disable all maskable interrupts (since the priority level for maskable interrupts ranges from 1 to 6). The EI instruction takes effect as soon as it is executed. (On the TLCS-90, the EI instruction takes effect after the execution of the instruction which follows it.)

In addition to the general-purpose interrupt processing mode described above, there is also a micro DMA processing mode.

In micro DMA mode the CPU automatically transfers data in 1-byte, 2-byte or 4-byte blocks, this mode allows high-speed data transfer to and from internal and external memory and internal I/O ports.

In addition, the TMP91CW18A also has a soft start function in which micro DMA processing is requested in software rather than by an interrupt.

Figure 3.4.1 is a flowchart showing an overview of interrupt processing.

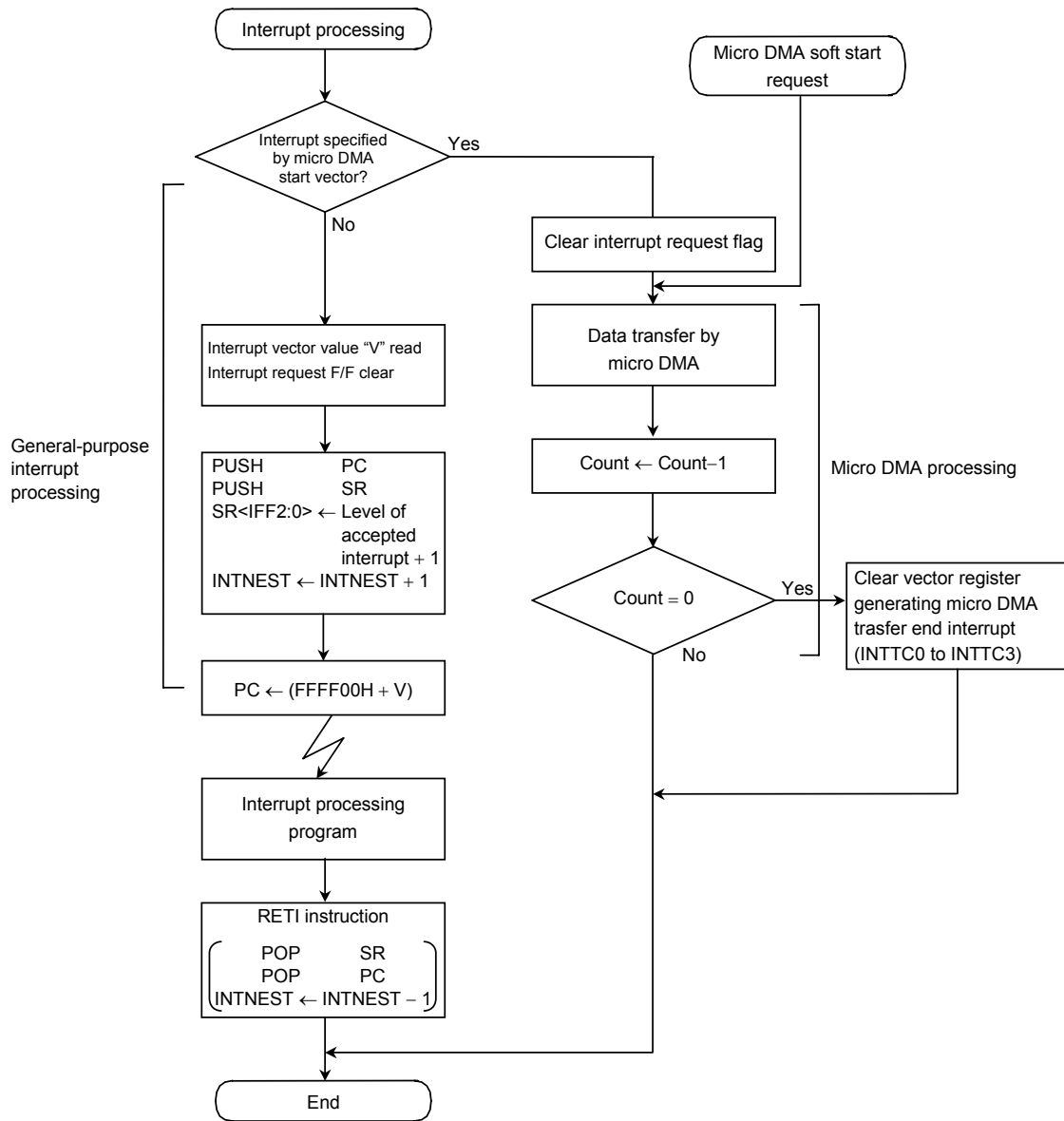


Figure 3.4.1 Interrupt and Micro DMA Processing Sequence

3.4.1 General-purpose Interrupt Processing

When the CPU accepts an interrupt, it usually performs the following sequence of operations. However, in the case of software interrupts and illegal instruction interrupts generated by the CPU, the CPU skips steps (1) and (3) and executes only steps (2), (4) and (5).

- (1) The CPU reads the interrupt vector from the interrupt controller.

When more than one interrupt with the same priority level have been generated simultaneously, the interrupt controller generates an interrupt vector in accordance with the default priority and clears the interrupt requests.

(The default priority is determined as follows: The smaller the vector value, the higher the priority.)

- (2) The CPU pushes the program counter (PC) and status register (SR) onto the top of the stack (Pointed to by XSP).
- (3) The CPU sets the value of the CPU's interrupt mask register <IFF2:0> to the priority level for the accepted interrupt plus 1. However, if the priority level for the accepted interrupt is 7, the register's value is set to 7.
- (4) The CPU increments the interrupt nesting counter INTNEST by 1.
- (5) The CPU jumps to the address given by adding the contents of address FFFF00H to the interrupt vector, then starts the interrupt processing routine.

On completion of interrupt processing, the RETI instruction is used to return control to the main routine. RETI restores the contents of the program counter and the status register from the stack and decrements the interrupt nesting counter INTNEST by 1.

Non-maskable interrupts cannot be disabled by a user program. Maskable interrupts, however, can be enabled or disabled by a user program. A program can set the priority level for each interrupt source. (A priority level setting of 0 or 7 will disable an interrupt request.)

If an interrupt request is received for an interrupt with a priority level equal to or greater than the value set in the CPU's interrupt mask register <IFF2:0>, the CPU will accept the interrupt. The CPU's interrupt mask register <IFF2:0> is then set to the value of the priority level for the accepted interrupt plus 1.

Thus, if during interrupt processing another interrupt is generated with a higher priority than the interrupt currently being processed, or if during the processing of a non-maskable interrupt, a non-maskable interrupt request is generated from another source, the CPU will suspend the routine which it is currently executing and accept the new interrupt. When processing of the new interrupt has been completed, the CPU will resume processing of the suspended interrupt.

If the CPU receives another interrupt request while performing processing steps (1) to (5), the second interrupt will be sampled immediately after execution of the first instruction of its interrupt processing routine. Specifying DI as the first instruction disables nesting of maskable interrupts. (Note: On the TLCS-900 and 900/L, sampling is performed before execution of the first instruction.)

A reset initializes the interrupt mask register <IFF2:0> to 111, disabling all maskable interrupts.

Table 3.4.1 shows the TMP91CW18A interrupt vectors and micro DMA start vectors. FFFF00H to FFFFFFFH (256 bytes) is designated as the interrupt vector area.

Table 3.4.1 TMP91CW18A Interrupt Vectors and Micro DMA Start Vectors

Default Priority	Type	Interrupt Source or Source of Micro DMA Request	Vector Value	Vector Reference Address	Micro DMA Start Vector	
1	Non-maskable	Reset or instruction "SWI0"	0000H	FFFF00H	–	
2		Instruction "SWI1"	0004H	FFFF04H	–	
3		Illegal instruction or instruction "SWI2"	0008H	FFFF08H	–	
4		Instruction "SWI3"	000CH	FFFF0CH	–	
5		Instruction "SWI4"	0010H	FFFF10H	–	
6		Instruction "SWI5"	0014H	FFFF14H	–	
7		Instruction "SWI6"	0018H	FFFF18H	–	
8		Instruction "SWI7"	001CH	FFFF1CH	–	
9		$\overline{\text{NMI}}$: NMI pin input	0020H	FFFF20H	–	
10		INTWD: Watchdog timer	0024H	FFFF24H	–	
–	–	Micro DMA	–	–	–	
11	Maskable	INT0: INT0 pin input	0028H	FFFF28H	0AH	
12		INT1: INT1 pin input	002CH	FFFF2CH	0BH	
13		INT2: INT2 pin input	0030H	FFFF30H	0CH	
14		INT3: INT3 pin input	0034H	FFFF34H	0DH	
15		INT4: INT4 pin input	0038H	FFFF38H	0EH	
16		INT5: INT5 pin input	003CH	FFFF3CH	0FH	
17		INT6: INT6 pin input	0040H	FFFF40H	10H	
–		–	–	–	–	–
–		–	–	–	–	–
20		INTTA0: 8-bit timer 0	004CH	FFFF4CH	13H	
21		INTTA1: 8-bit timer 1	0050H	FFFF50H	14H	
22		INTTA2: 8-bit timer 2	0054H	FFFF54H	15H	
23		INTTA3: 8-bit timer 3	0058H	FFFF58H	16H	
24		INTTA4: 8-bit timer 4	005CH	FFFF5CH	17H	
25		INTTA5: 8-bit timer 5	0060H	FFFF60H	18H	
26		INTTA6: 8-bit timer 6	0064H	FFFF64H	19H	
27		INTTA7: 8-bit timer 7	0068H	FFFF68H	1AH	
28		INTTB00: 16-bit timer 0 (TB0RG0)	006CH	FFFF6CH	1BH	
29		INTTB01: 16-bit timer 0 (TB0RG1)	0070H	FFFF70H	1CH	
–		–	–	–	–	–
–		–	–	–	–	–
32		INTTBOF0: 16-bit timer 0 (Overflow)	007CH	FFFF7CH	1FH	
–		–	–	–	–	–
34		INTRX0: Serial receive (Channel 0)	0084H	FFFF84H	21H	
35		INTTX0: Serial transmission (Channel 0)	0088H	FFFF88H	22H	
36		INTI2C2: I ² C bus interface interrupt	008CH	FFFF8CH	23H	
37		INTI2C1: I ² C bus interface interrupt	0090H	FFFF90H	24H	
38		INTSBI0: Serial bus interface interrupt	0094H	FFFF94H	25H	
–		–	–	–	–	–
40		INTAD: AD conversion end	009CH	FFFF9CH	27H	
41		INTTC0: Micro DMA end (Channel 0)	00A0H	FFFA0H	28H	
42		INTTC1: Micro DMA end (Channel 1)	00A4H	FFFA4H	29H	
43	INTTC2: Micro DMA end (Channel 2)	00A8H	FFFA8H	2AH		
44	INTTC3: Micro DMA end (Channel 3)	00ACH	FFFACH	2BH		
–	–	00B0H	FFFB0H	–		
to	(Reserved)	to	to	to		
–	–	00FCH	FFFFFCH	–		

3.4.2 Micro DMA Processing

In addition to general-purpose interrupt processing, the TMP91CW18A also includes a micro DMA function. Micro DMA processing for interrupt requests set by micro DMA is performed at the highest priority level for maskable interrupts (Level 6), regardless of the priority level of the interrupt source.

Because the micro DMA function has been implemented with the cooperative operation of CPU, when CPU is a state of standby mode (STOP, IDLE1 and IDLE2) by HALT instruction, the requirement of micro DMA will be ignored (Pending) and DMA transfer is started after release HALT.

(1) Micro DMA operation

When an interrupt request is generated by an interrupt source specified by the micro DMA start vector register, the micro DMA triggers a micro DMA request to the CPU at interrupt priority level 6 and starts processing the request. The four micro DMA channels allow micro DMA processing to be set for up to four types of interrupt at once.

When micro DMA is accepted, the interrupt request flip-flop assigned to that channel is cleared. Data is automatically transferred from the transfer source address to the transfer destination address set in the control register, and the transfer counter is decremented by 1. If the value of the counter after it has been decremented is not 0, DMA processing ends with no change in the value of the micro DMA start vector register. If the value of the decremented counter is 0, a micro DMA transfer end interrupt (INTTC0 to INTTC3) is sent from the CPU to the interrupt controller. In addition, the micro DMA start vector register is cleared to 0, the next micro DMA operation is disabled and micro DMA processing terminates.

If micro DMA requests are set simultaneously for more than one channel, priority is not based on the interrupt priority level but on the channel number. The lower the channel number, the higher the priority (Channel 0 thus has the highest priority and channel 3 the lowest).

If an interrupt request is triggered for the interrupt source in use during the interval between the time at which the micro DMA start vector is cleared and the next setting, general-purpose interrupt processing is performed at the interrupt level set. Therefore, if the interrupt is only being used to initiate micro DMA (and not as a general-purpose interrupt), the interrupt level should first be set to 0 (e.g., interrupt requests should be disabled).

If micro DMA and general-purpose interrupts are being used together as described above, the level of the interrupt which is being used to initiate micro DMA processing should first be set to a lower value than all the other interrupt levels. (Note) In this case, edge-triggered interrupts are the only kinds of general interrupts which can be accepted.

Note: If the priority level of micro DMA is set higher than that of other interrupts, CPU operates as follows.
 In case INTxxx interrupt is generated first and then INTyyy interrupt is generated between checking "Interrupt specified by micro DMA start vector" (in the Table 3.4.1) and reading interrupt vector with setting below. The vector shifts to that of INTyyy at the time.
 This is because the priority level of INTyyy is higher than that of INTxxx.
 In the interrupt routine, CPU reads the vector of INTyyy because checking of micro DMA has finished.
 And INTyyy is generated regardless of transfer counter of micro DMA.
 INTxxx: level 1 without micro DMA
 INTyyy: level 6 with micro DMA

Although the control registers used for setting the transfer source and transfer destination addresses are 32 bits wide, this type of register can only output 24-bit addresses. Accordingly, micro DMA can only access 16 Mbytes (The upper 8 bits of a 32-bit address are not valid).

Three micro DMA transfer modes are supported: 1-byte transfers, 2-byte (One-word) transfers and 4-byte transfers. After a transfer in any mode, the transfer source and transfer destination addresses will either be incremented or decremented, or will remain unchanged. This simplifies the transfer of data from I/O to memory, from memory to I/O, and from I/O to I/O. For details of the various transfer modes, see section 3.4.2 (4) “Detailed description of the transfer mode register”.

Since a transfer counter is a 16-bit counter, up to 65536 micro DMA processing operations can be performed per interrupt source (Provided that the transfer counter for the source is initially set to 0000H).

Micro DMA processing can be initiated by any one of 35 different interrupts – the 34 interrupts shown in the micro DMA start vectors in Table 3.4.1 or a micro DMA soft start.

Figure 3.4.2 shows a 2-byte transfer carried out using a micro DMA cycle in transfer destination address INC mode. (Micro DMA transfers are the same in every mode except counter mode.) (The conditions for this cycle are as follows: External 16-bit bus, 0 waits, and even-numbered transfer source and transfer destination addresses.)

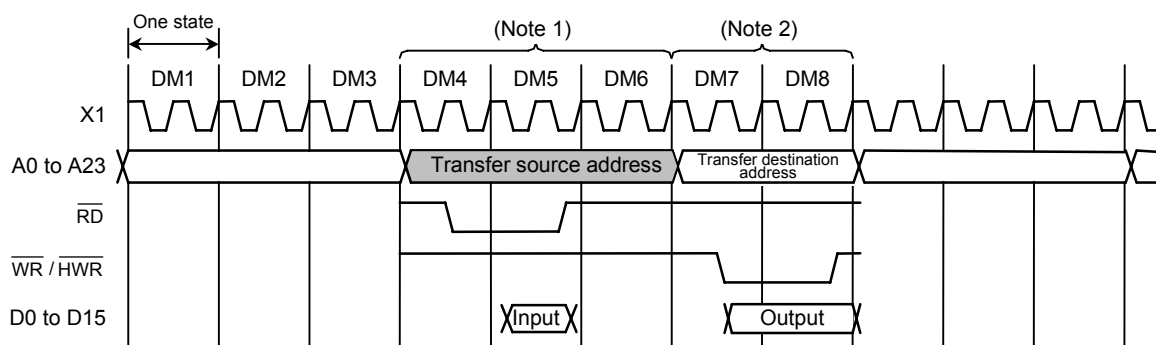


Figure 3.4.2 Timing for Micro DMA Cycle

States 1, 3: Instruction fetch cycle (Prefetch the next instruction)

Once 3 or more bytes of code have been fetched into the instruction queue, dummy cycle is inserted into instruction fetch cycle.

States 4, 5: Micro DMA read cycle

State 6: Dummy cycle (The address bus remains unchanged from state 5)

States 7, 8: Micro DMA write cycle

Note 1: If the source address area is an 8-bit bus, it is incremented by two states.

If the source address area is a 16-bit bus and the address starts from an odd number, it is incremented by two states.

Note 2: If the destination address area is an 8-bit bus, it is incremented by two states.

If the destination address area is a 16-bit bus and the address starts from an odd number, it is incremented by two states.

(2) Soft start function

The TMP91CW18A can initiate micro DMA either with an interrupt or by using the micro DMA soft start function, in which micro DMA is initiated by a write cycle which writes to the register DMAR.

Writing 1 to any bit of the register DMAR causes micro DMA to be performed once (If write “0” to each bit, micro DMA doesn’t operate). On completion of the transfer, the bits of DMAR which support the end channel are automatically cleared to 0.

DMA can only be requested for one channel at once. (Therefore, do not write 1 to plural bits.)

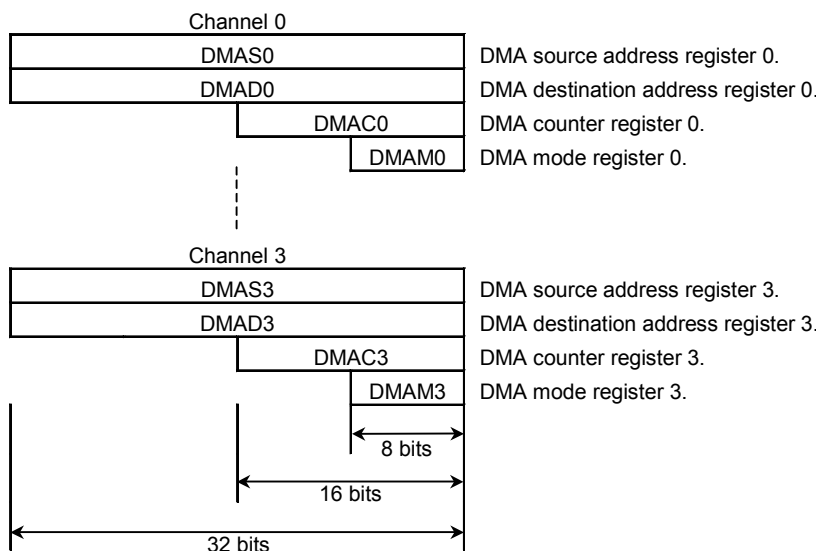
When writing again 1 to the DMAR register, check whether the bit is 0 before writing 1. If read 1, micro DMA transfer isn’t started yet.

When a burst is specified by the register DMAB, data is transferred continuously until the value in the micro DMA transfer counter is 0 after start up of the micro DMA. If execute soft start during micro DMA transfer by interrupt source, micro DMA transfer counter doesn’t change. Don’t use Read-modify write instruction to avoid writing to other bits by mistake.

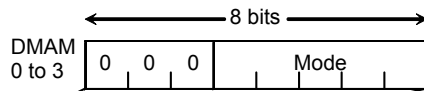
Symbol	Name	Address	7	6	5	4	3	2	1	0
DMAR	DMA request register	89H (Prohibit RMW)					DMA request			
							DMAR3	DMAR2	DMAR1	DMAR0
							R/W			
							0	0	0	0

(3) Transfer control registers

The transfer source address and the transfer destination address are set in the following registers. An instruction of the form “LDC cr, r” can be used to set these registers.



(4) Detailed description of the transfer mode register



Note: Only values whose upper 3 bits are 000 should be set in this register.

		Number of Transfer Bytes	Mode Description	Number of Execution States (*)	Minimum Execution Time at $f_c = 25 \text{ MHz}$
000 (Fixed)	000	00	Byte transfer Transfer destination address INC mode I/O to memory	8 states	640 ns
		01	Word transfer (DMADn+) ← (DMASn) DMACn ← DMACn - 1	12 states	960 ns
		10	4-byte transfer If DMACn = 0, then INTTCn is generated.		
	001	00	Byte transfer Transfer destination address DEC mode I/O to memory	8 states	640 ns
		01	Word transfer (DMADn-) ← (DMASn) DMACn ← DMACn - 1	12 states	960 ns
		10	4-byte transfer If DMACn = 0, then INTTCn is generated.		
	010	00	Byte transfer Transfer source address INC mode Memory to I/O	8 states	640 ns
		01	Word transfer (DMADn) ← (DMASn+) DMACn ← DMACn - 1	12 states	960 ns
		10	4-byte transfer If DMACn = 0, then INTTCn is generated.		
	011	00	Byte transfer Transfer source address DEC mode Memory to I/O	8 states	640 ns
		01	Word transfer (DMADn) ← (DMASn-) DMACn ← DMACn - 1	12 states	960 ns
		10	4-byte transfer If DMACn = 0, then INTTCn is generated.		
100	00	Byte transfer Fixed address mode I/O to I/O	8 states	640 ns	
	01	Word transfer (DMADn) ← (DMASn-) DMACn ← DMACn - 1	12 states	960 ns	
	10	4-byte transfer If DMACn = 0, then INTTCn is generated.			
101	00	Counter mode For counting number of times interrupt is generated DMASn ← DMASn + 1 DMACn ← DMACn - 1 If DMACn = 0, then INTTCn is generated.	5 states	400 ns	

*: External 16-bit bus, 0 waits, word transfer mode or 4-byte transfer mode, even-numbered transfer source and transfer destination addresses.

Note: n stands for the micro DMA channel number (0 to 3)

DMADn+/DMASn+: Post-increment (Register value is incremented after transfer)

DMADn-/DMASn-: Post-decrement (Register value is decremented after transfer)

“I/O” signifies fixed memory addresses; “memory” signifies incremented or memory addresses.

The transfer mode register should not be set to any value other than those listed above.

3.4.3 Interrupt Controller Operation

The block diagram in Figure 3.4.3 shows the interrupt circuits. The left-hand side of the diagram shows the interrupt controller circuit. The right-hand side shows the CPU interrupt request signal circuit and the halt release circuit.

For each of the 38 interrupt channels there is an interrupt request flag (Consisting of a flip-flop), an interrupt priority setting register and a micro DMA start vector register. The interrupt request flag latches interrupt requests from the peripherals. The flag is cleared to 0 in the following cases: When a reset occurs, when the CPU reads the channel vector of an interrupt it has received, when the CPU receives a micro DMA request (when micro DMA is set), when a micro DMA burst transfer is terminated, and when an instruction that clears the interrupt for that channel is executed (by a 0 written to the clear bit in the interrupt priority setting register).

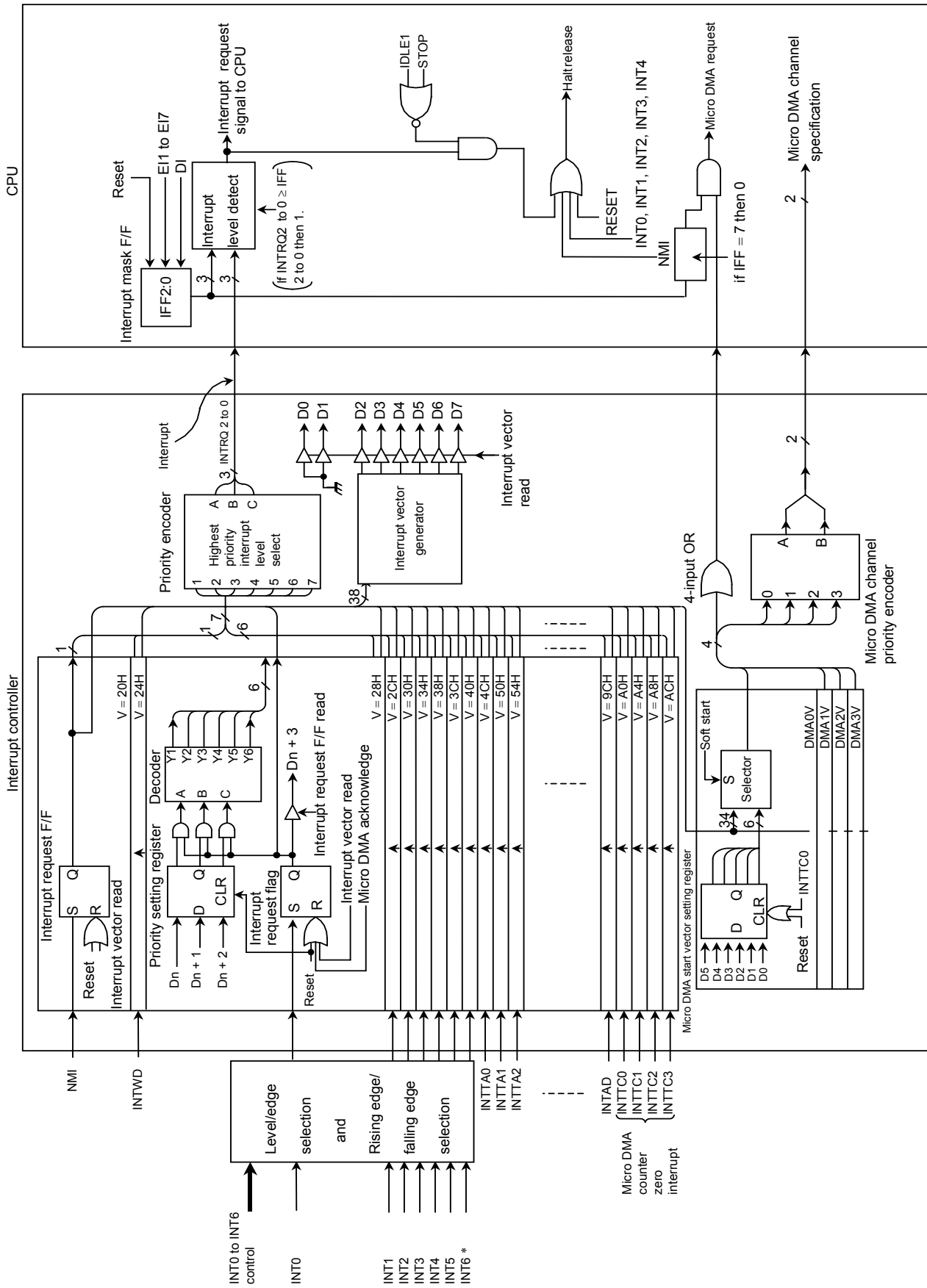
An interrupt priority can be set independently for each interrupt source by writing the priority to the interrupt priority setting register (e.g., INTE0AD or INTE12). 6 interrupt priorities levels (1 to 6) are provided. Setting an interrupt source's priority level to 0 (or 7) disables interrupt requests from that source. The priority of non-maskable interrupts (NMI pin interrupts and watchdog timer interrupts) is fixed at 7. If more than one interrupt request with a given priority level are generated simultaneously, the default priority (The interrupt with the lowest priority or, in other words, the interrupt with the lowest vector value) is used to determine which interrupt request is accepted first.

The 3rd and 7th bits of the interrupt priority setting register indicate the state of the interrupt request flag and thus whether an interrupt request for a given channel has occurred.

If several interrupts are generated simultaneously, the interrupt controller sends the interrupt request for the interrupt with the highest priority and the interrupt's vector address to the CPU. The CPU compares the mask value set in <IFF2:0> of the status register (SR) with the priority level of the requested interrupt; if the latter is higher, the interrupt is accepted. Then, the CPU sets SR<IFF2:0> to the priority level of the accepted interrupt + 1. Hence, during processing of the accepted interrupt, new interrupt requests with a priority value equal to or higher than the value set in SR <IFF2:0> (e.g., interrupts with a priority higher than the interrupt being processed) will be accepted.

When interrupt processing has been completed (e.g., after execution of a RETI instruction), the CPU restores to SR<IFF2:0> the priority value which was saved on the stack before the interrupt was generated.

The interrupt controller also includes four registers which are used to store the micro DMA start vector. Writing the start vector of the interrupt source for the micro DMA processing (See Table 3.4.1), enables the corresponding interrupt to be processed by micro DMA processing. The values must be set in the micro DMA parameter registers (e.g., DMAS and DMAD) prior to micro DMA processing.

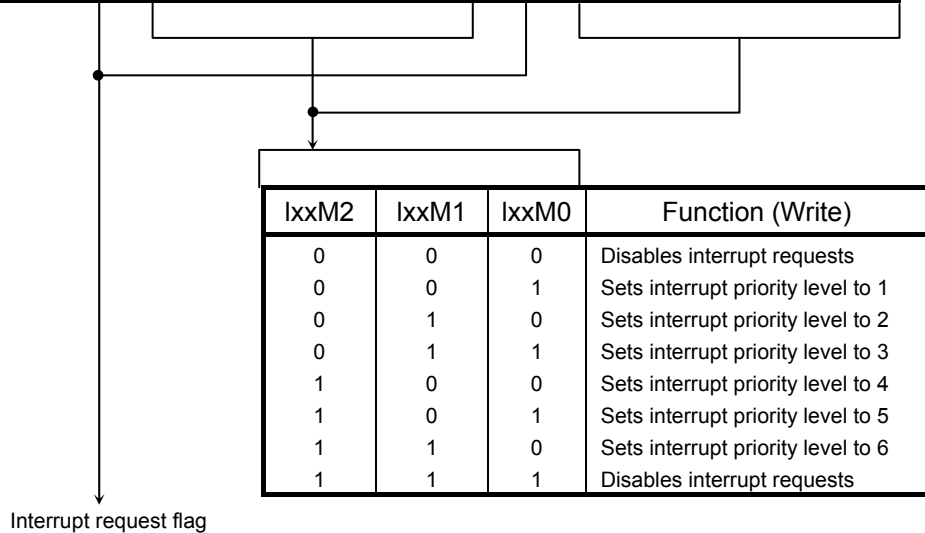


*: Only rising edge

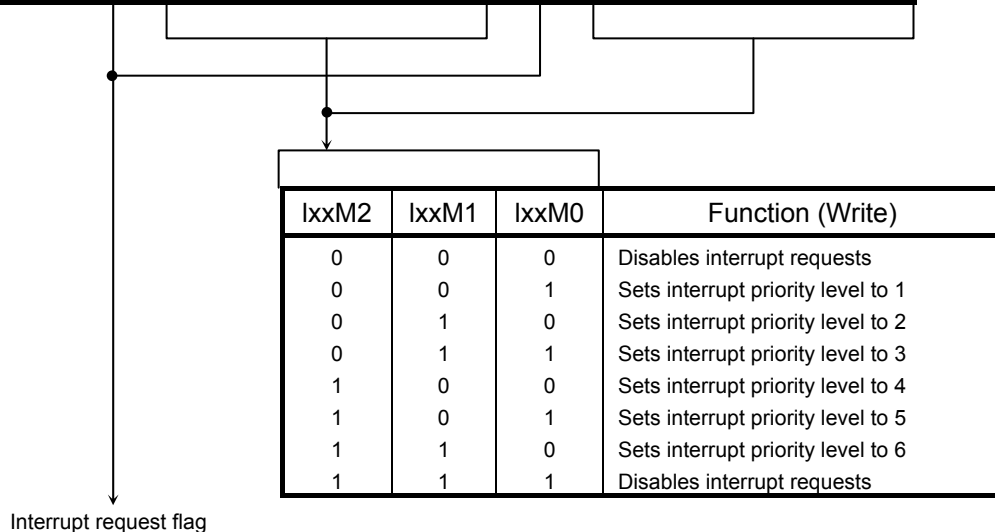
Figure 3.4.3 Block Diagram of Interrupt Controller

(1) Interrupt priority setting registers

Symbol	Name	Address	7	6	5	4	3	2	1	0
INTE0AD	INT0 & INTAD enable	90H	INTAD				INT0			
			IADC	IADM2	IADM1	IADM0	IOC	IOM2	IOM1	IOM0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	
INTE12	INT1 & INT2 enable	91H	INT2				INT1			
			I2C	I2M2	I2M1	I2M0	I1C	I1M2	I1M1	I1M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	
INTE34	INT3 & INT4 enable	92H	INT4				INT3			
			I4C	I4M2	I4M1	I4M0	I3C	I3M2	I3M1	I3M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	
INTE56	INT5 & INT6 enable	93H	INT6				INT5			
			I6C	I6M2	I6M1	I6M0	I5C	I5M2	I5M1	I5M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	
INTEA01	INTTA0 & INTTA1 enable	95H	INTTA1 (TMRA1)				INTTA0 (TMRA0)			
			ITA1C	ITA1M2	ITA1M1	ITA1M0	ITA0C	ITA0M2	ITA0M1	ITA0M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	
INTEA23	INTTA2 & INTTA3 enable	96H	INTTA3 (TMRA3)				INTTA2 (TMRA2)			
			ITA3C	ITA3M2	ITA3M1	ITA3M0	ITA2C	ITA2M2	ITA2M1	ITA2M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	
INTEA45	INTTA4 & INTTA5 enable	97H	INTTA5 (TMRA5)				INTTA4 (TMRA4)			
			ITA5C	ITA5M2	ITA5M1	ITA5M0	ITA4C	ITA4M2	ITA4M1	ITA4M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	
INTEA67	INTTA6 & INTTA7 enable	98H	INTTA7 (TMRA7)				INTTA6 (TMRA6)			
			ITA7C	ITA7M2	ITA7M1	ITA7M0	ITA6C	ITA6M2	ITA6M1	ITA6M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	



Symbol	Name	Address	7	6	5	4	3	2	1	0
INTETB0	Interrupt enable TMRB0	99H	INTTB01 (TMRB0)				INTTB00 (TMRB0)			
			ITB01C	ITB01M2	ITB01M1	ITB01M0	ITB00C	ITB00M2	ITB00M1	ITB00M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
INTETB0 OV	Interrupt enable TMRB0 (Over flow)	9BH	(Reserved)				INTTBOF0 (TMRB0)			
			/				ITF0C	ITF0M2	ITF0M1	ITF0M0
			/				R	R/W		
			/				0	0	0	0
INTE UART	Interrupt enable UART	9CH	INTTX0				INTRX0			
			ITX0C	ITX0M2	ITX0M1	ITX0M0	IRX0C	IRX0M2	IRX0M1	IRX0M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
INTES2	Interrupt enable I ² C2	9DH	(Reserved)				INTI2C2			
			/				INTI2C2C	I2C2M2	I2C2M1	I2C2M0
			/				R	R/W		
			/				0	0	0	0
INTES1	Interrupt enable I ² C1 & SBI	9EH	INTI2C1				INTSBI0			
			INTI2C1C	I2C1M2	I2C1M1	I2C1M0	IS0C	IS0M2	IS0M1	IS0M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
INTETC 01 & INTTC1 enable	INTTC0 & INTTC1 enable	A0H	INTTC1				INTTC0			
			ITC1C	ITC1M2	ITC1M1	ITC1M0	ITC0C	ITC0M2	ITC0M1	ITC0M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
INTETC 23 & INTTC3 enable	INTTC2 & INTTC3 enable	A1H	INTTC3				INTTC2			
			ITC3C	ITC3M2	ITC3M1	ITC3M0	ITC2C	ITC2M2	ITC2M1	ITC2M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0



(2) External interrupt control

Symbol	Name	Address	7	6	5	4	3	2	1	0		
IIMC	Interrupt input mode control	8CH (Prohibit RMW)	-	I4EDGE	I3EDGE	I2EDGE	I1EDGE	I0EDGE	I0LE	NMIREE		
			W									
			0	0	0	0	0	0	0	0		
			Always write 0	INT4EDGE 0: Rising 1: Falling	INT3EDGE 0: Rising 1: Falling	INT2EDGE 0: Rising 1: Falling	INT1EDGE 0: Rising 1: Falling	INT0EDGE 0: Rising 1: Falling	0: INT0 edge mode 1: INT0 level mode	1: Operates even on rising + falling edge of NMI		

















INT0 level enable

0	Edge detect INT
1	High level INT

NMI rising edge enable

0	INT request generation at falling edge
1	INT request generation at rising/falling edge

Setting for external interruption inputs

Interrupt Request Pin	Pin Name	Mode	Condition
NMI	-	 Falling edge	IIMC<NMIREE> = 0
		 Falling/rising edge	IIMC<NMIREE> = 1
INT0	P60	 Rising edge	IIMC<I0LE> = 0, <I0EDGE> = 0
		 Falling edge	IIMC<I0LE> = 0, <I0EDGE> = 1
		 High level	IIMC<I0LE> = 1
INT1	P61	 Rising edge	IIMC<I1EDGE> = 0
		 Falling edge	IIMC<I1EDGE> = 1
INT2	P62	 Rising edge	IIMC<I2EDGE> = 0
		 Falling edge	IIMC<I2EDGE> = 1
INT3	P76	 Rising edge	IIMC<I3EDGE> = 0
		 Falling edge	IIMC<I3EDGE> = 1
INT4	P37	 Rising edge	IIMC<I4EDGE> = 0
		 Falling edge	IIMC<I4EDGE> = 1
INT5	P73	 Rising edge	TB0MOD<TB0CPM1, 0> = 0, 0 or 0, 1 or 1, 1
		 Falling edge	TB0MOD<TB0CPM1, 0> = 1, 0
INT6	P74	 Rising edge	-

(3) Interrupt request flag clear register

The interrupt request flag is cleared by writing the appropriate micro DMA start vector, as given in Table 3.4.1, to the register INTCLR.

For example, to clear the interrupt flag INT0, perform the following register operation after execution of the DI instruction.

INTCLR ← 0AH INT0 Clears interrupt request flag.

Symbol	Name	Address	7	6	5	4	3	2	1	0		
INTCLR	Interrupt clear control	88H (Prohibit RMW)			CLR5	CLR4	CLR3	CLR2	CLR1	CLR0		
			W									
					0	0	0	0	0	0		
			Interrupt vector									

(4) Micro DMA start vector registers

These registers assign micro DMA processing to an interrupt sets which source corresponds to DMA. The interrupt source whose micro DMA start vector value matches the vector set in one of these registers is designated as the micro DMA start source.

When the micro DMA transfer counter value reaches zero, the micro DMA transfer end interrupt corresponding to the channel is sent to the interrupt controller, the micro DMA start vector register is cleared, and the micro DMA start source for the channel is cleared. Therefore, in order for micro DMA processing to continue, the micro DMA start vector register must be set again during processing of the micro DMA transfer end interrupt.

If the same vector is set in the micro DMA start vector registers of more than one channel, the lowest numbered channel takes priority.

Accordingly, if the same vector is set in the micro DMA start vector registers for two different channels, the interrupt generated on the lower-numbered channel is executed until micro DMA transfer is complete. If the micro DMA start vector for this channel has not been set in the channel's micro DMA start vector register again, micro DMA transfer for the higher-numbered channel will be commenced. (This process is known as micro DMA chaining.)

Symbol	Name	Address	7	6	5	4	3	2	1	0				
DMA0V	DMA0 start vector	80H	DMA0 start vector											
			DMA0V5		DMA0V4		DMA0V3		DMA0V2		DMA0V1		DMA0V0	
			R/W											
			0		0		0		0		0		0	
DMA1V	DMA1 start vector	81H	DMA1 start vector											
			DMA1V5		DMA1V4		DMA1V3		DMA0V2		DMA1V1		DMA1V0	
			R/W											
			0		0		0		0		0		0	
DMA2V	DMA2 start vector	82H	DMA2 start vector											
			DMA2V5		DMA2V4		DMA2V3		DMA2V2		DMA2V1		DMA2V0	
			R/W											
			0		0		0		0		0		0	
DMA3V	DMA3 start vector	83H	DMA3 start vector											
			DMA3V5		DMA3V4		DMA3V3		DMA3V2		DMA3V1		DMA3V0	
			R/W											
			0		0		0		0		0		0	

(5) Specification of a micro DMA burst

Specifying the micro DMA burst function causes micro DMA transfer, once started, to continue until the value in the transfer counter register reaches 0. Setting any of the bits in the register DMAB which correspond to a micro DMA channel (as shown below) to 1 specifies that any micro DMA transfer on that channel will be a burst transfer.

Symbol	Name	Address	7	6	5	4	3	2	1	0
DMAR	DMA software request register	89H (Prohibit RMW)	DMAR3		DMAR2		DMAR1		DMAR0	
			R/W		R/W		R/W		R/W	
			0		0		0		0	
			1: DMA software request							
DMAB	DMA burst register	8AH	DMAB3		DMAB2		DMAB1		DMAB0	
			R/W							
			0		0		0		0	
			1: DMA burst request							

(6) Notes

The instruction execution unit and the bus interface unit in this CPU operate independently. Therefore, if immediately before an interrupt is generated, the CPU fetches an instruction which clears the corresponding interrupt request flag, the CPU may execute this instruction in between accepting the interrupt and reading the interrupt vector. In this case, the CPU will read the default vector 0004H and jump to interrupt vector address FFFF04H.

To avoid the above program, place instructions that clear interrupt request flags after a DI instruction. And in the case of setting an interrupt enable again by EI instruction after the execution of clearing instruction, execute EI instruction after clearing and more than 1-instructions (e.g., "NOP" × 1 times). If placed EI instruction without waiting NOP instruction after execution of clearing instruction, interrupt will be enable before request flag is cleared.

In the case of changing the value of the interrupt mask register <IFF2:0> by execution of POP SR instruction, disable an interrupt by DI instruction before execution of POP SR instruction.

In addition, please note that the following two circuits are exceptional and demand special attention.

INT0 level mode	<p>In level mode INT0 is not an edge-triggered interrupt, hence in level mode the interrupt request flip-flop for INT0 does not function. The peripheral interrupt request passes through the S input of the flip-flop and becomes the Q output. If the interrupt input mode is changed from edge mode to level mode, the interrupt request flag is cleared automatically.</p> <p>If the CPU enters the interrupt response sequence as a result of INT0 going from 0 to 1, INT0 must then be held at 1 until the interrupt response sequence has been completed. If INT0 is set to level mode so as to release a halt state, INT0 must be held at 1 from the time INT0 changes from 0 to 1 until the halt state is released. (Hence, it is necessary to ensure that input noise is not interpreted as a 0, causing INT0 to revert to 0 before the halt state has been released.)</p> <p>When the mode changes from level mode to edge mode, interrupt request flags which were set in level mode will not be cleared. Interrupt request flags must be cleared using the following sequence.</p> <pre> DI LD (IIMC), 00H ; Switches interrupt input mode from level mode to ; edge mode. LD (INTCLR), 0AH ; Clears interrupt request flag. NOP ; Wait EI instruction EI </pre>
INTRX	The interrupt request flip-flop can only be cleared by a reset or by reading the serial channel receive buffer. It cannot be cleared by writing INTCLR register.

Note: The following instructions or pin input state changes are equivalent to instructions which clear the interrupt request flag.

INT0: Instructions which switch to level mode after an interrupt request has been generated in edge mode.

The pin input changes from high to low after an interrupt request has been generated in level mode. (High → Low)

INTRX: Instructions which read the receive buffer.

3.5 Port Functions

The TMP91CW18A features 62-bit settings which relate to the various I/O ports.

As well as general-purpose I/O port functionality, the port pins also have I/O functions which relate to the built-in CPU and internal I/Os. Table 3.5.1 lists the functions of each port pin.

Table 3.5.2 lists I/O registers and their specifications.

Table 3.5.1 Port Functions

(OD: Δ = Open drain)

(R: \uparrow = with programmable pull-up resistor)

(POD: \circ = Programmable open drain)

Port Name	Pin Name	Number of Pins	Direction	R	POD	OD	Direction Setting Unit	Pin Name for Internal Function
Port 0	P00 to P07	8	I/O	–			Bit	AD0 to AD7
Port 1	P10 to P17	8	I/O	–			Bit	AD8 to AD15/A8 to A15
Port 2	P20 to P27	8	I/O	–			Bit	A16 to A23/A0 to A7
Port 3	P30	1	I/O	–	\circ		Bit	\overline{RD}
	P31	1	I/O	–	\circ		Bit	\overline{WR}
	P32	1	I/O	\uparrow	\circ		Bit	\overline{HWR}
	P33	1	I/O	\uparrow	\circ		Bit	\overline{WAIT}
	P34	1	I/O	–	\circ		Bit	TA6IN
	P35	1	I/O	–	\circ		Bit	TA7OUT
	P36	1	I/O	–	\circ		Bit	INT4
Port 4	P40 to P43	4	Input	–			(Fixed)	AN8 to AN11, \overline{ADTRG} (P40)
Port 5	P50 to P57	8	Input	–			(Fixed)	AN0 to AN7
Port 6	P60	1	I/O	–			Bit	INT0
	P61	1	I/O	–			Bit	INT1/ \overline{CTS}
	P62	1	I/O	–			Bit	INT2/SCOUT
Port 7	P70	1	I/O	–			Bit	TA1OUT
	P71	1	I/O	–			Bit	TA3OUT
	P72	1	I/O	–			Bit	TA5OUT
	P73	1	I/O	–			Bit	TB0IN0/INT5
	P74	1	I/O	–			Bit	TB0IN1/INT6
	P75	1	I/O	–			Bit	TB0OUT0
	P76	1	I/O	–			Bit	INT3/SCK0
Port 8	P80	1	I/O	–	\circ		Bit	SDA0/SO0
	P81	1	I/O	–	\circ		Bit	SCL0/SIO
	P82	1	I/O	–	\circ		Bit	TXD
	P83	1	I/O	–	\circ		Bit	RXD
	P84	1	I/O	–		Δ	Bit	SDA1
	P85	1	I/O	–		Δ	Bit	SCL1
	P86	1	I/O	–		Δ	Bit	SDA2
	P87	1	I/O	–		Δ	Bit	SCL2

Table 3.5.2 I/O Registers and Their Specifications (1/2)

Port	Name	Specification	I/O registers			
			Pn	PnCR	PnFC	
Port 0	P00 to P07	Input port	x	0	None	
		Output port	x	1		
		AD bus (AD0 to AD7)	x	x		
Port 1	P10 to P17	Input port	x	0	0	
		Output port	x	1	0	
		AD bus (AD8 to AD15)	x	0	1	
		A output (A8 to A15)	x	1	1	
Port 2	P20 to P27	Input port	x	0	0	
		Output port	x	1	0	
		A output (A0 to A7)	x	0	1	
		A output (A16 to A23)	x	1	1	
Port 3	P30	Input port	x	0	0	
		Output port	x	1	0	
		Outputs \overline{RD} only when accessing an external area	1	None	0	
		Always output \overline{RD}	0		1	
	P31	Input port	x	0	0	
		Output port	x	1	0	
		Outputs \overline{WR} only when accessing an external area	x	None	1	
	P32	Input port (without pull up)	0	0	0	
		Input port (with pull up)	1	0	0	
		Output port	x	1	0	
		H \overline{WR} output	x	1	1	
	P33	WAIT $\overline{}$ input (without pull up)	0	0	None	
		WAIT $\overline{}$ input (with pull up)	1	0		
		Output port	x	1		
	P34	Input port	x	0	None	
		Output port	x	1		
	P35	Input port	x	0	None	
		Output port	x	1		
		TA6IN input	x	0		
	P36	Input port	x	0	0	
		Output port	x	1	0	
		TA7OUT output	x	1	1	
	P37	Input port	x	0	0	
		Output port	x	1	0	
INT4 input		x	0	1		
Port 4	P40 to P43	Input port	x	None		
		AN input (AN8 to AN11) (Note 1)	x			
	P40	ADTRG $\overline{}$ input (Note 2)	x			
Port 5	P50 to P57	Input port	x	None		
		AN input (AN0 to AN7) (Note 1)	x			
Port 6	P60	Input port	x	0	0	
		Output port	x	1	0	
		INT0 input	x	0	1	
	P61	Input port	x	0	0	
		Output port	x	1	0	
		INT1 input	x	0	1	
		\overline{CTS} input	x	0	0	
	P62 (Note 4)	Input port	(SCOUT<SCOUTE:0>)	x	0	0
		Output port	(SCOUT<SCOUTE:0>)	x	1	0
		INT2 input	(SCOUT<SCOUTE:0>)	x	0	1
SCOUT output		(SCOUT<SCOUTE:1>)	x	1	0	

X: Don't care

Table 3.5.3 I/O Registers and Their Specifications (2/2)

Port	Name	Specification		I/O registers		
				Pn	PnCR	PnFC
Port 7	P70	Input port		x	0	0
		Output port		x	1	0
		TA1OUT output		x	1	1
	P71	Input port		x	0	0
		Output port		x	1	0
		TA3OUT output		x	1	1
	P72	Input port		x	0	0
		Output port		x	1	0
		TA5OUT output		x	1	1
	P73	Input port	(IIEC<INT5E:0>)	x	0	None
		Output port	(IIEC<INT5E:0>)	x	1	
		INT5/TB0IN0 input	(IIEC<INT5E:1>)	x	0	
	P74	Input port	(IIEC<INT6E:0>)	x	0	None
		Output port	(IIEC<INT6E:0>)	x	1	
		INT6/TB0IN1 input	(IIEC<INT6E:1>)	x	0	
	P75	Input port		x	0	0
		Output port		x	1	0
		TB0OUT0 output		x	1	1
	P76	Input port	(IIEC<INT3E:0>)	x	0	0
		Output port	(IIEC<INT3E:0>)	x	1	0
		SCK0 input/output	(IIEC<INT3E:0>)	x	1	1
INT3 input		(IIEC<INT3E:1>)	x	0	1	
Port 8 (Note 5)	P80	Input port		x	0	0
		Output port		x	1	0
		SDA0 input		x	1	1
		SO0 output		x	1	1
	P81	Input port		x	0	0
		Output port		x	1	0
		SCL0 input/output		x	1	1
		SI0 input		x	1	1
	P82	Input port		x	0	0
		Output port		x	1	0
		TXD output		x	1	1
	P83	Input port		x	0	None
		Output port		x	1	
		RXD input		x	0	
	P84	Input port		x	0	0
		Output port		x	1	0
		SDA1 input/output		x	1	1
	P85	Input port		x	0	0
		Output port		x	1	0
		SCL1 input/output		x	1	1
	P86	Input port		x	0	0
		Output port		x	1	0
		SDA2 input/output		x	1	1
	P87	Input port		x	0	0
		Output port		x	1	0
		SCL2 input/output		x	1	1

X: Don't care

Note 1: When P50 to P57 are used as AD converter input channels, a 3-bit field in the AD mode control register ADMOD1<ADCH3:0> is used to select the channel.

Note 2: When P40 is used as the $\overline{\text{ADTRG}}$ input, ADMOD1<ADTRGE> is used to enable external trigger input.

Note 3: When P30 to P37 are used as open-drain outputs, P3ODE<P37ODE:P30ODE> are used to set open-drain output mode.

Note 4: When P62 is used SCOUT, SCOUT<SCOUTE> is used to set SCOUT.

Note 5: When P80 to P83 are used as open-drain outputs, P8ODE<P83ODE:P80ODE> are used to set open-drain output mode.

Note 6: When P73, P74, P76 are used as INT5/TB0IN0, INT6/TB0IN1 and INT3 are used to set IIEC<INT5E, INT6E, INT3E>.

After a reset the port pins listed below function as general-purpose I/O port pins.
A reset sets I/O pins which can be programmed for either input port pins.

Setting the port pins for internal function use must be done in software.

3.5.1 Port 0 (P00 to P07)

Port 0 is an 8-bit general-purpose I/O port each bit can be individually for input or output using the control register P0CR. Resetting resets all bits of P0CR to 0 and sets port 0 to input mode.

In addition to functioning as a general-purpose I/O port, port 0 can also function as an address data bus (AD0 to AD7), allowing access to external memory. In this case, all bits in the control register P0CR are cleared to 0.

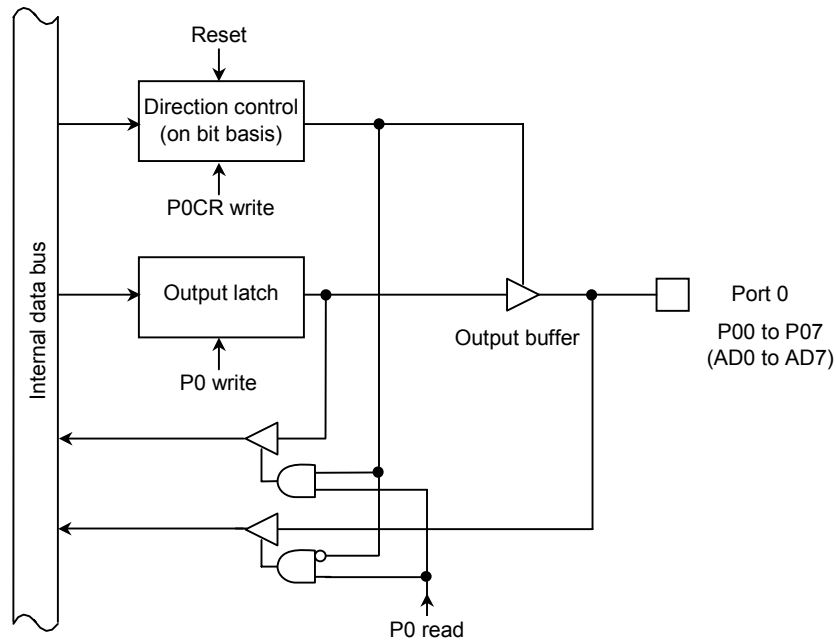


Figure 3.5.1 Port 0

3.5.2 Port 1 (P10 to P17)

Port 1 is an 8-bit general-purpose I/O port. Each bit can be set individually for input or output using the control register P1CR and the function register P1FC. Resetting resets all bits of the output latch P1, the control register P1CR and the function register P1FC to 0 and sets port 1 to input mode.

In addition to functioning as a general-purpose I/O port, port 1 can also function as an address data bus (AD8 to AD15) or an address bus (A8 to A15).

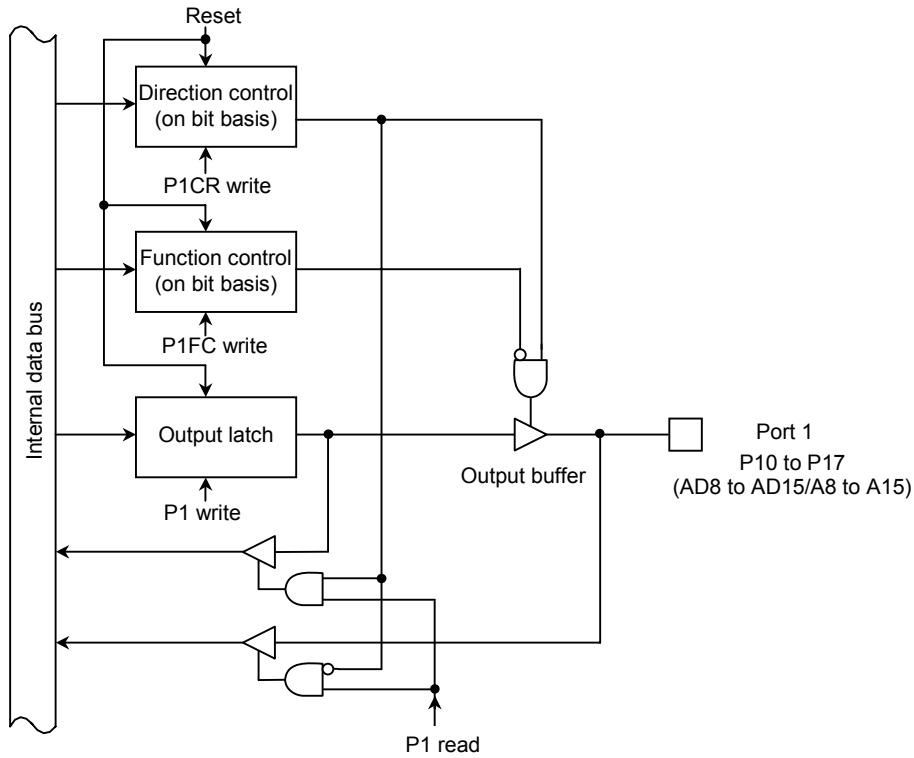


Figure 3.5.2 Port 1

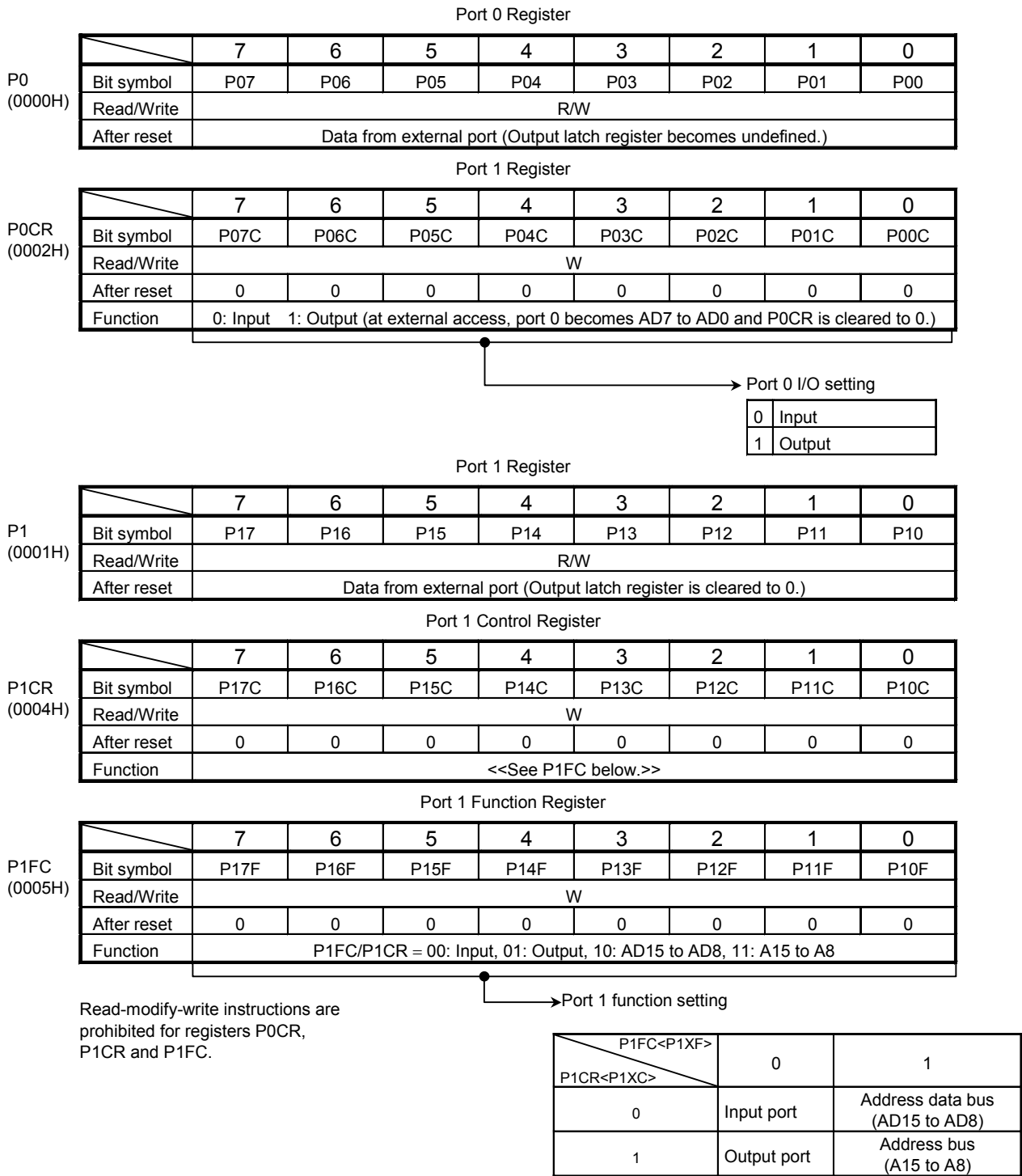


Figure 3.5.3 Registers for Ports 0 and 1

3.5.3 Port 2 (P20 to P27)

Port 2 is an 8-bit general-purpose I/O port. Each bit can be set individually for input or output using the control register P2CR and the function register P2FC. Resetting set all bits of the output latch P2 to “1”, the control register P2CR and the function register P2FC to 0 and sets port 2 to input mode.

In addition to functioning as a general-purpose I/O port, port 2 can also function as an address bus (A0 to A7) or (A16 to A23).

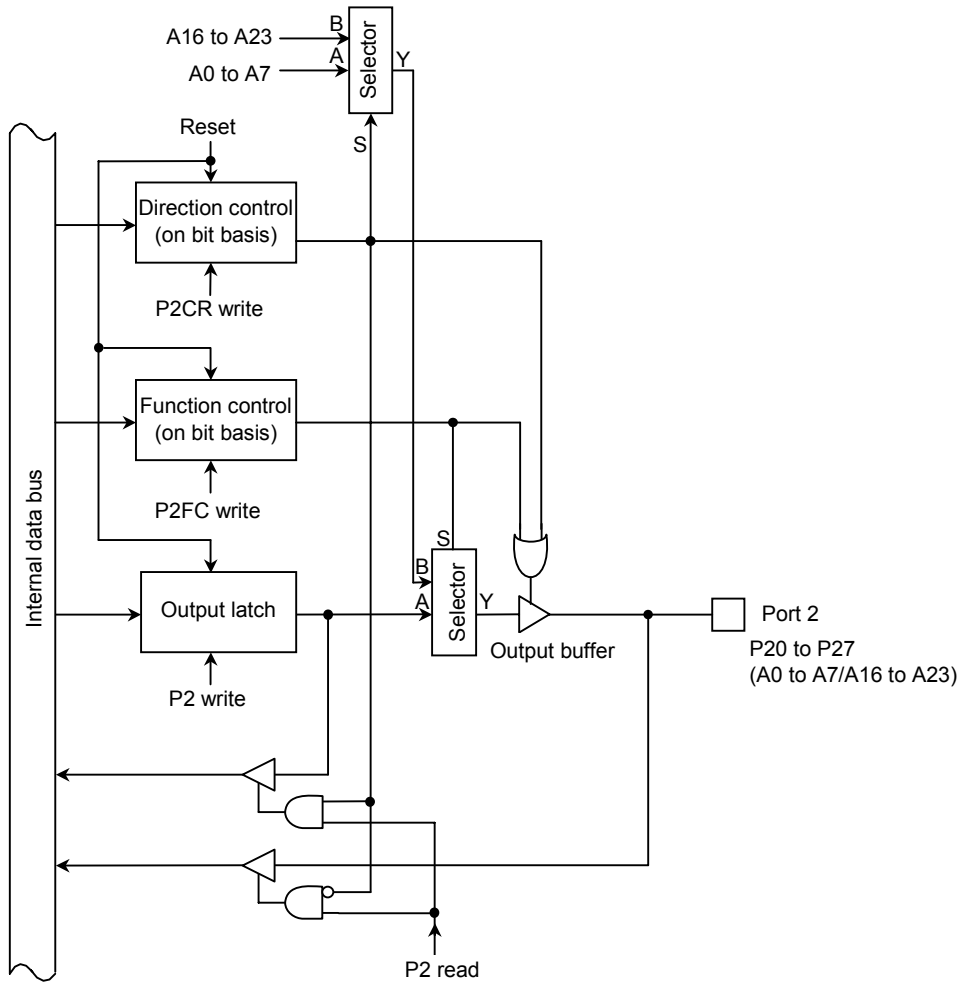


Figure 3.5.4 Port 2

Port 2 Register

	7	6	5	4	3	2	1	0	
P2 (0006H)	Bit symbol	P27	P26	P25	P24	P23	P22	P21	P20
	Read/Write	R/W							
	After reset	Data from external port (Output latch register is set to 1.)							

Port 2 Control Register

	7	6	5	4	3	2	1	0	
P2CR (0008H)	Bit symbol	P27C	P26C	P25C	P24C	P23C	P22C	P21C	P20C
	Read/Write	W							
	After reset	0	0	0	0	0	0	0	0
	Function	<<See P2FC below>>							

Port 2 Function Register

	7	6	5	4	3	2	1	0	
P2FC (0009H)	Bit symbol	P27F	P26F	P25F	P24F	P23F	P22F	P21F	P20F
	Read/Write	W							
	After reset	0	0	0	0	0	0	0	0
	Function	P2FC/P2CR = 00: Input, 01: Output, 10: A7 to A0, 11: A23 to A16							

Note: Read-modify-write instructions are prohibited for P2CR and P2FC.

Port 2 function setting

	P2FC<P2XF>	0	1
P2CR<P2XC>	0	Input port	Address bus (A7 to A0)
	1	Output port	Address bus (A23 to A16)

Note: <P2XF> is bit X in register P2FC; <P2XC> is bit X in register P2CR.
When setting port 2 to function as the address bus A23 to A16, first set P2CR, then set P2FC.

Figure 3.5.5 Registers for Port 2

3.5.4 Port 3 (P30 to P37)

Port 3 is an 8-bit general-purpose I/O port. Each bit can be set individually for input or output. I/O is set using the control register P3CR and the function register P3FC. Resetting sets all bits of the output latch P3 and bit 0 and bit 1 of control register P3CR to 1. Bit 2 to bit 7 of P3CR are set to 0. All bits of the control register P3CR (P30C, P31C sets to 1, P32C to P37C resets to 0) and the function register P3FC (of which bits 3, 4 and 5 are unused) are cleared to 0. Resetting also causes P30 and P31 to output 1, sets P32 to P33 to input mode and turns on the pull-up resistor.

And also, when output port is set, each bit is able to be set as open-drain port by P3ODE.

In addition to functioning as a general-purpose I/O port, port 3 can also function as the I/O for the CPU's control/status signal.

When the P30 pin is set for \overline{RD} signal output mode (<P30F> = 1), clearing the output latch register <P30> to 0 causes the \overline{RD} strobe signal (used for the pseudo-static RAM) to be output from the P30 pin even while the internal address area is being accessed.

If the output latch register <P30> remains set to 1, the \overline{RD} strobe signal is output only while the external address area is being accessed.

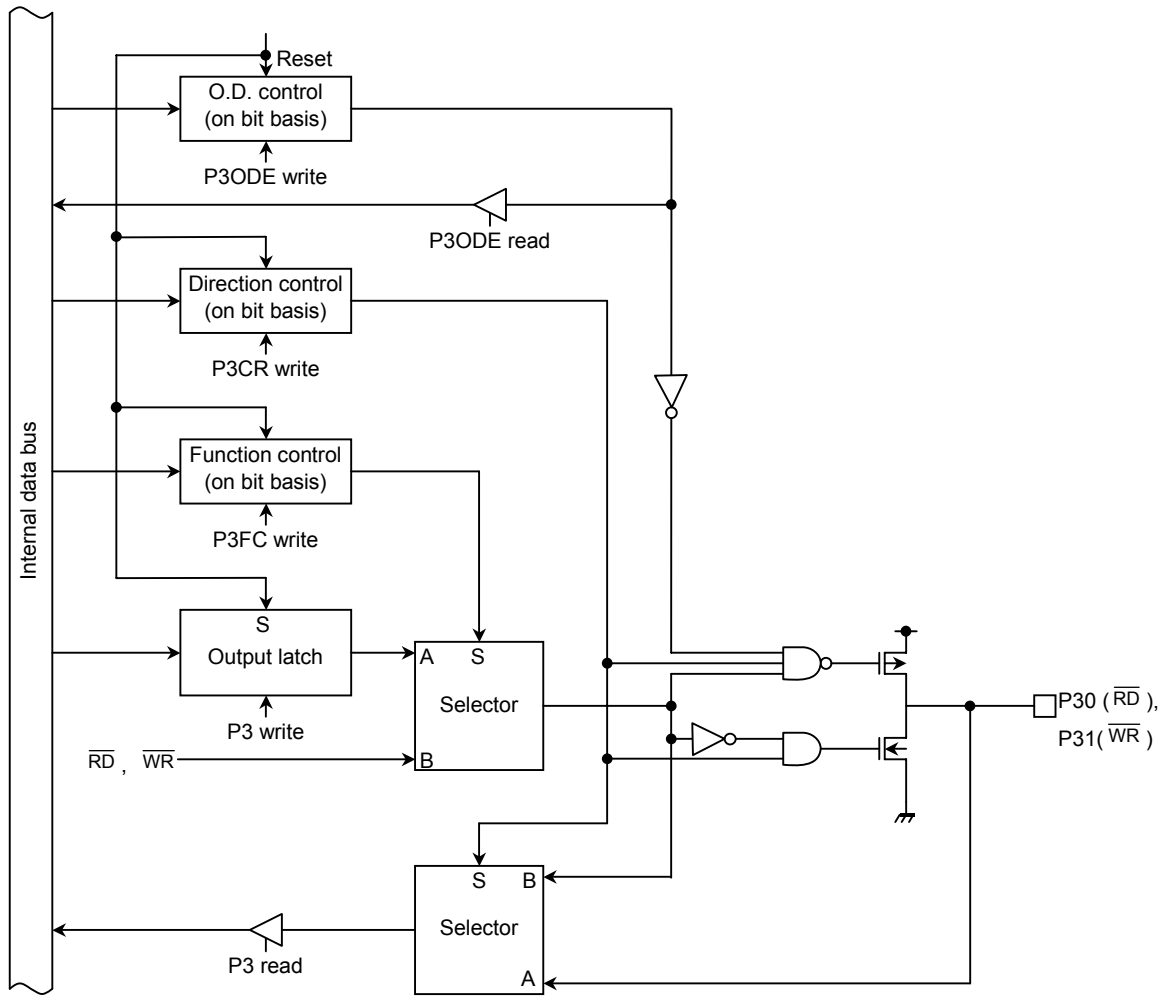


Figure 3.5.6 Port 3 (P30, P31)

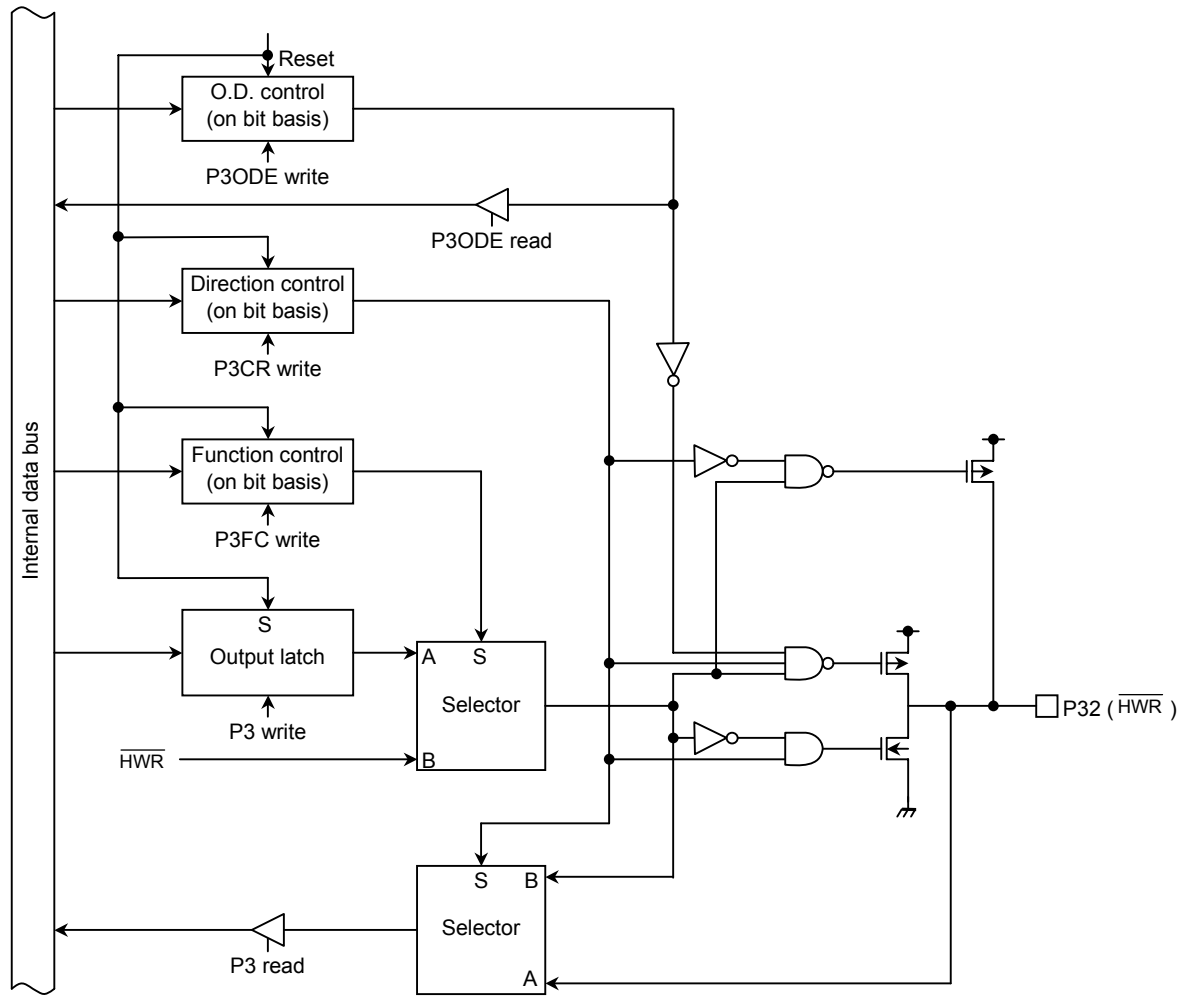


Figure 3.5.7 Port 3 (P32)

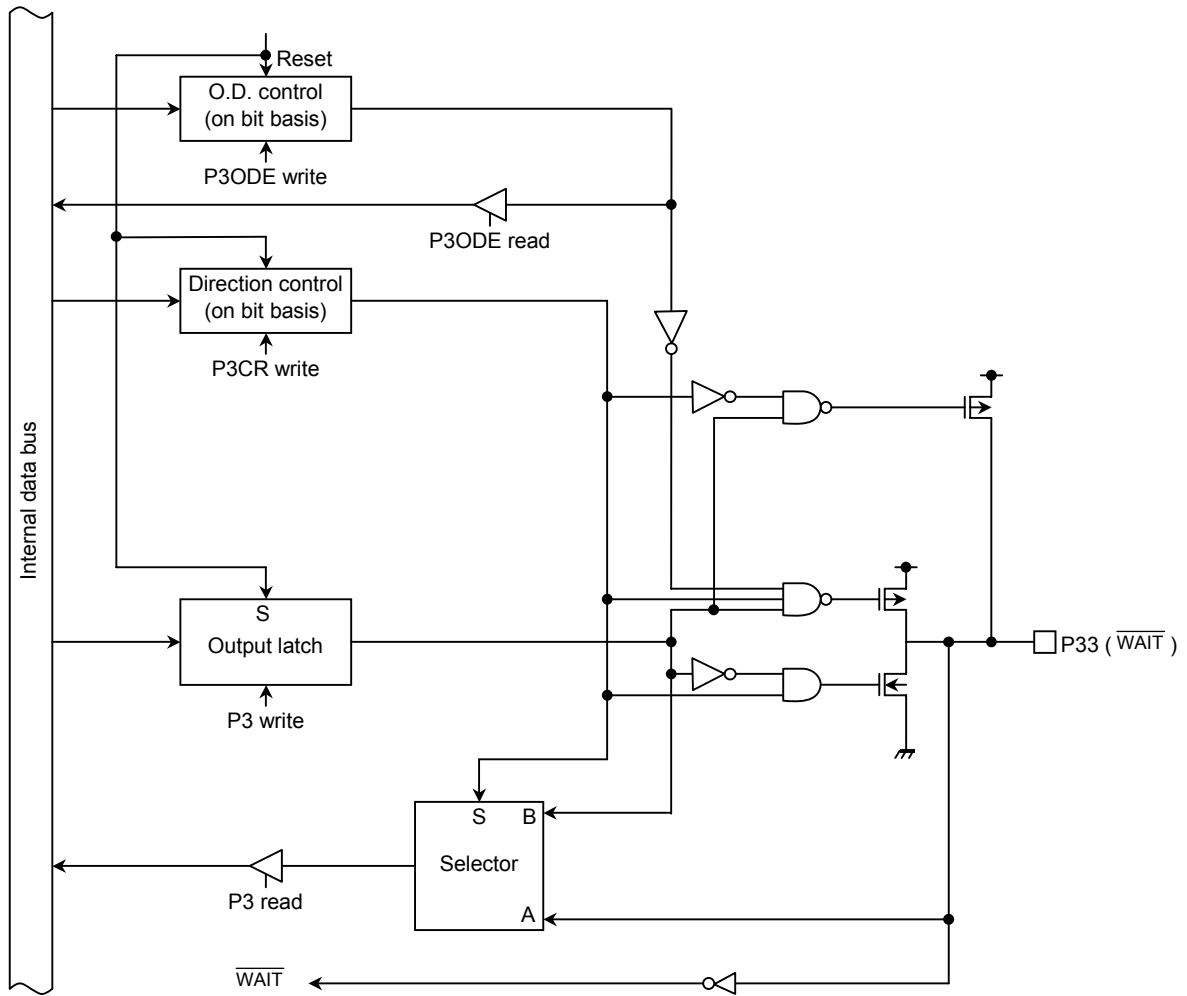


Figure 3.5.8 Port 3 (P33)

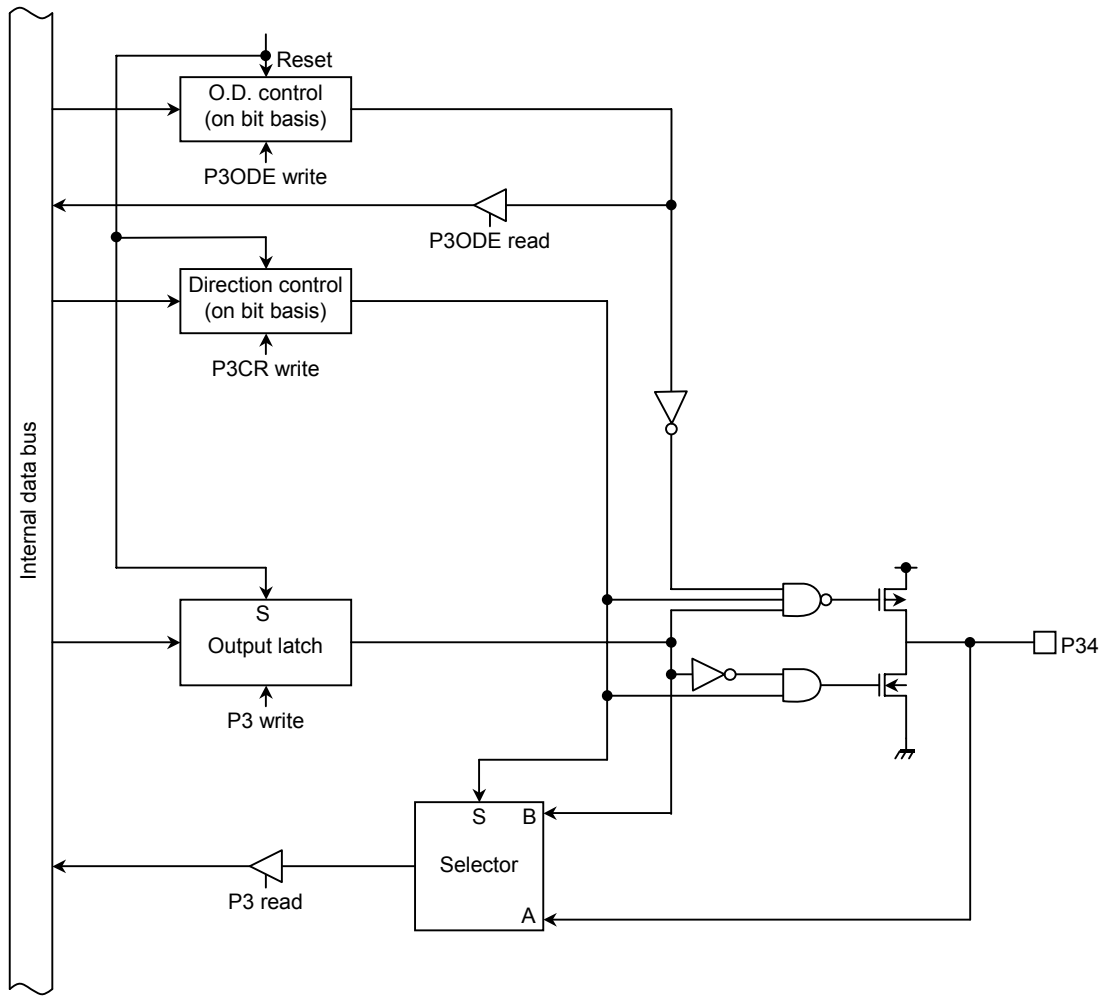


Figure 3.5.9 Port 3 (P34)

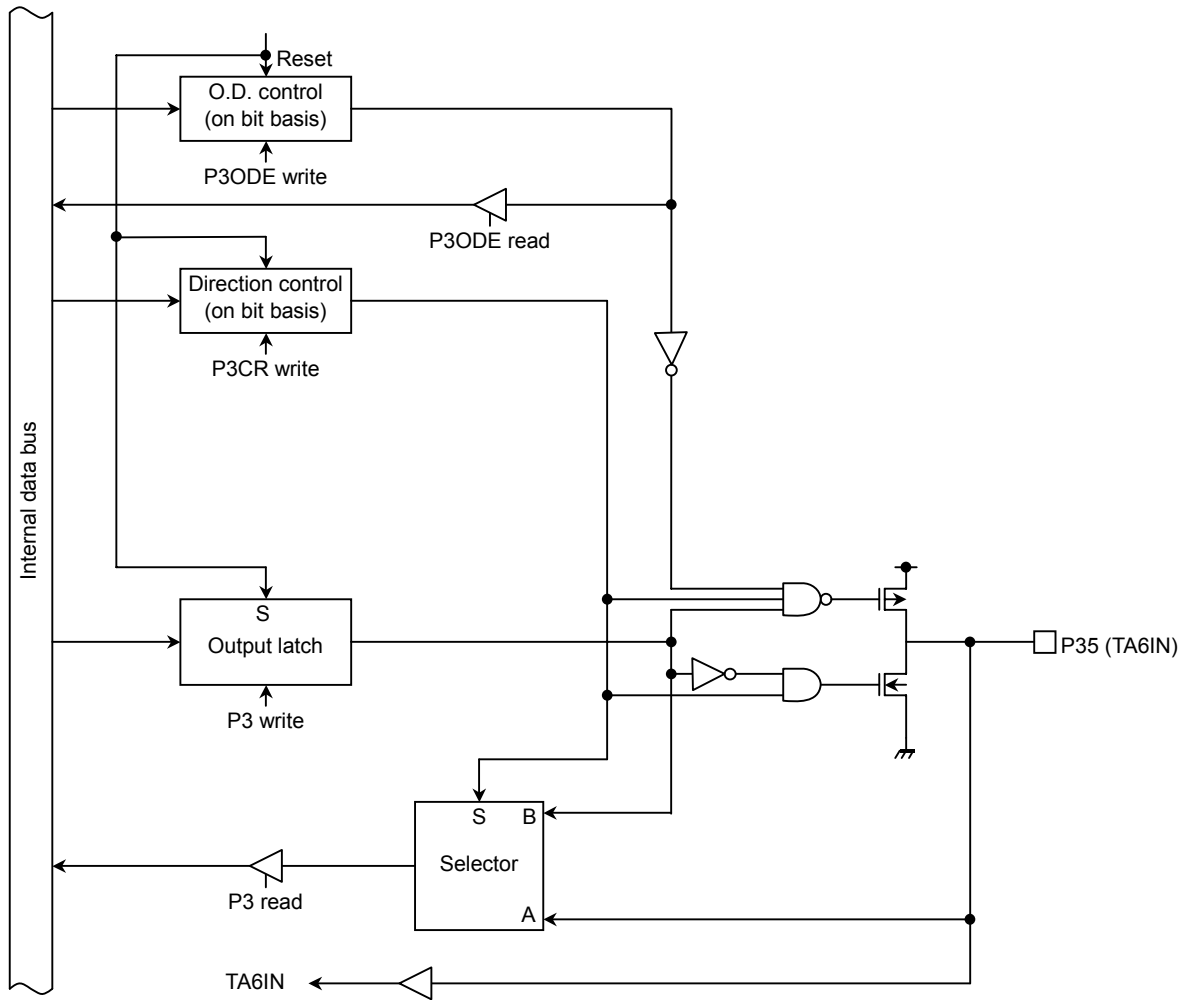


Figure 3.5.10 Port 3 (P35)

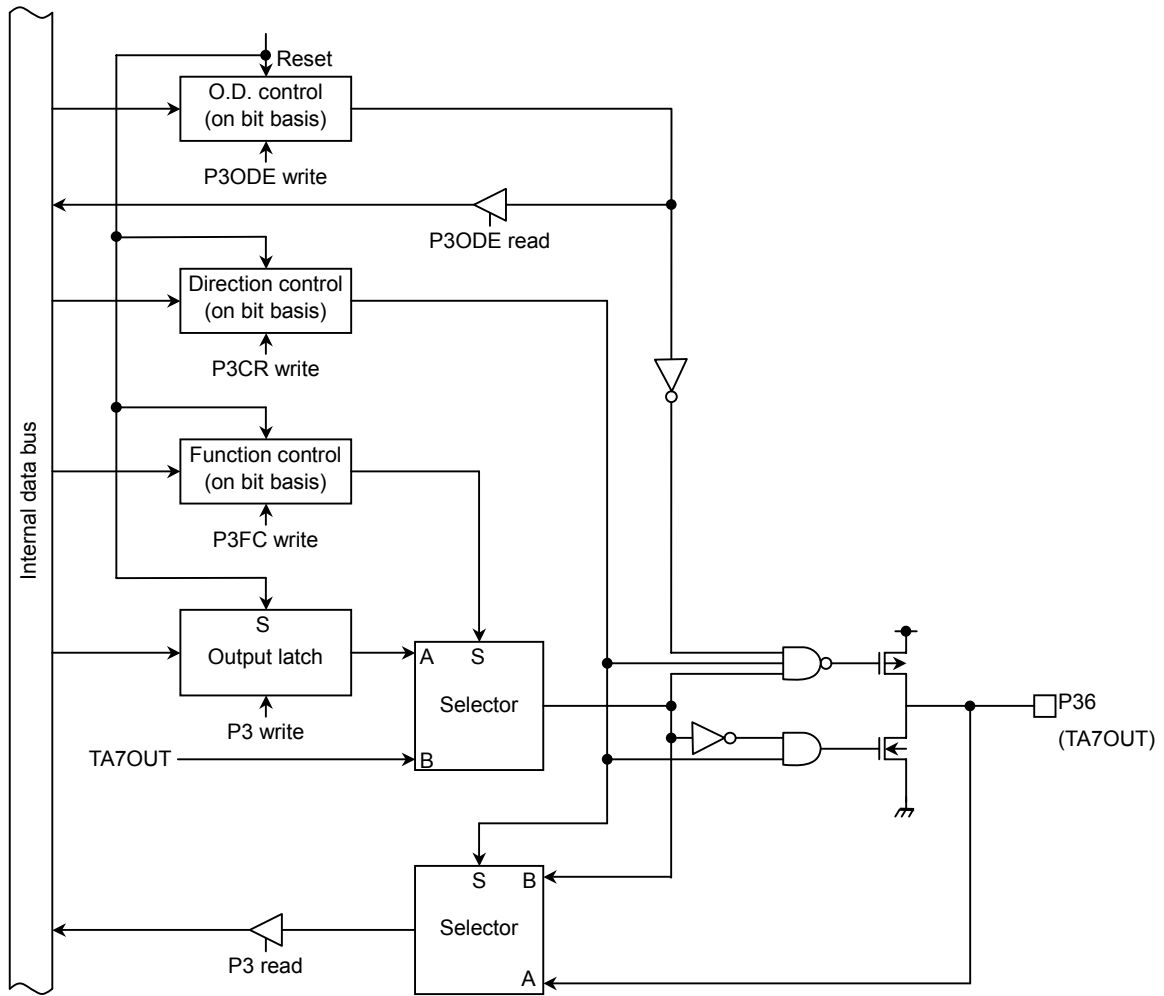


Figure 3.5.11 Port 3 (P36)

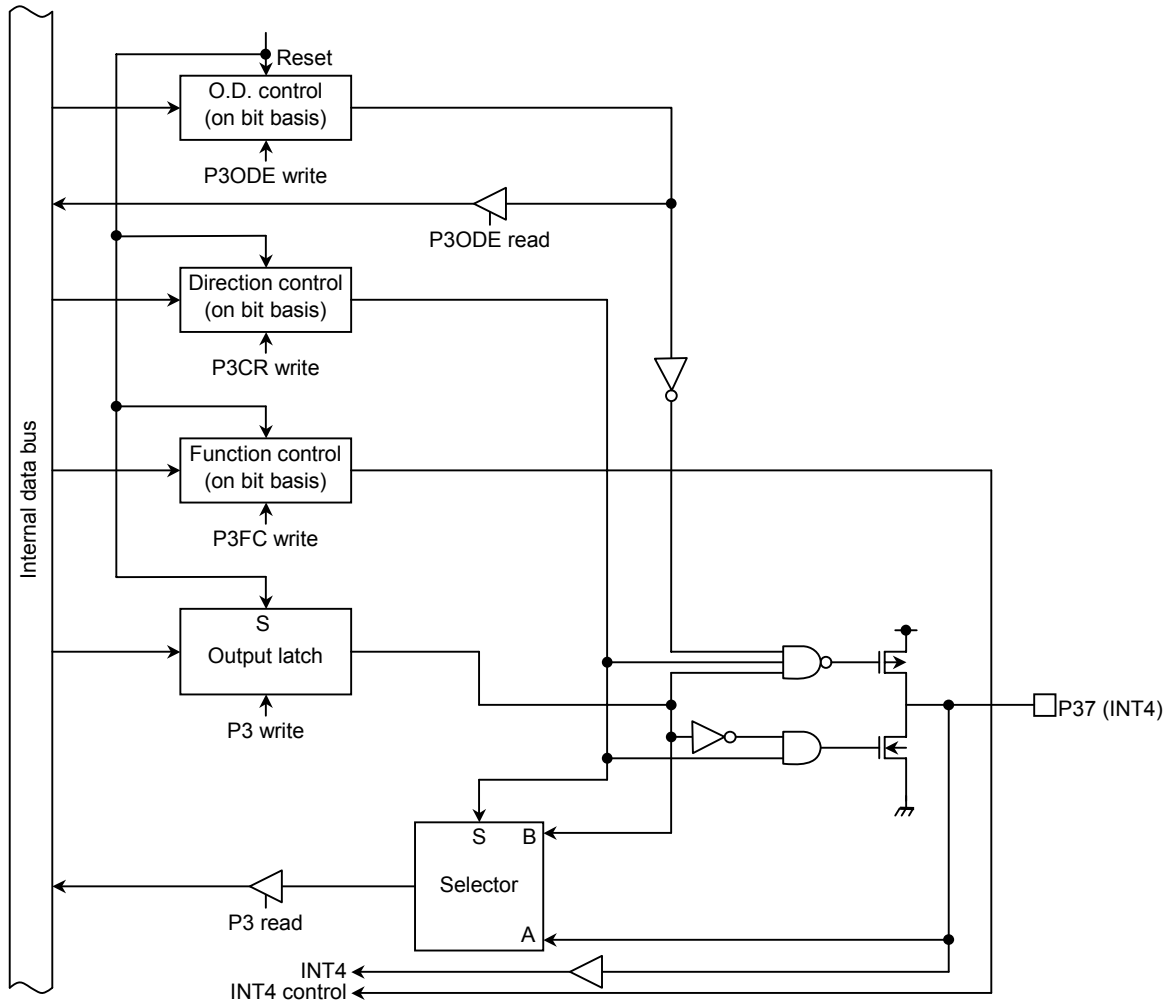


Figure 3.5.12 Port 3 (P37)

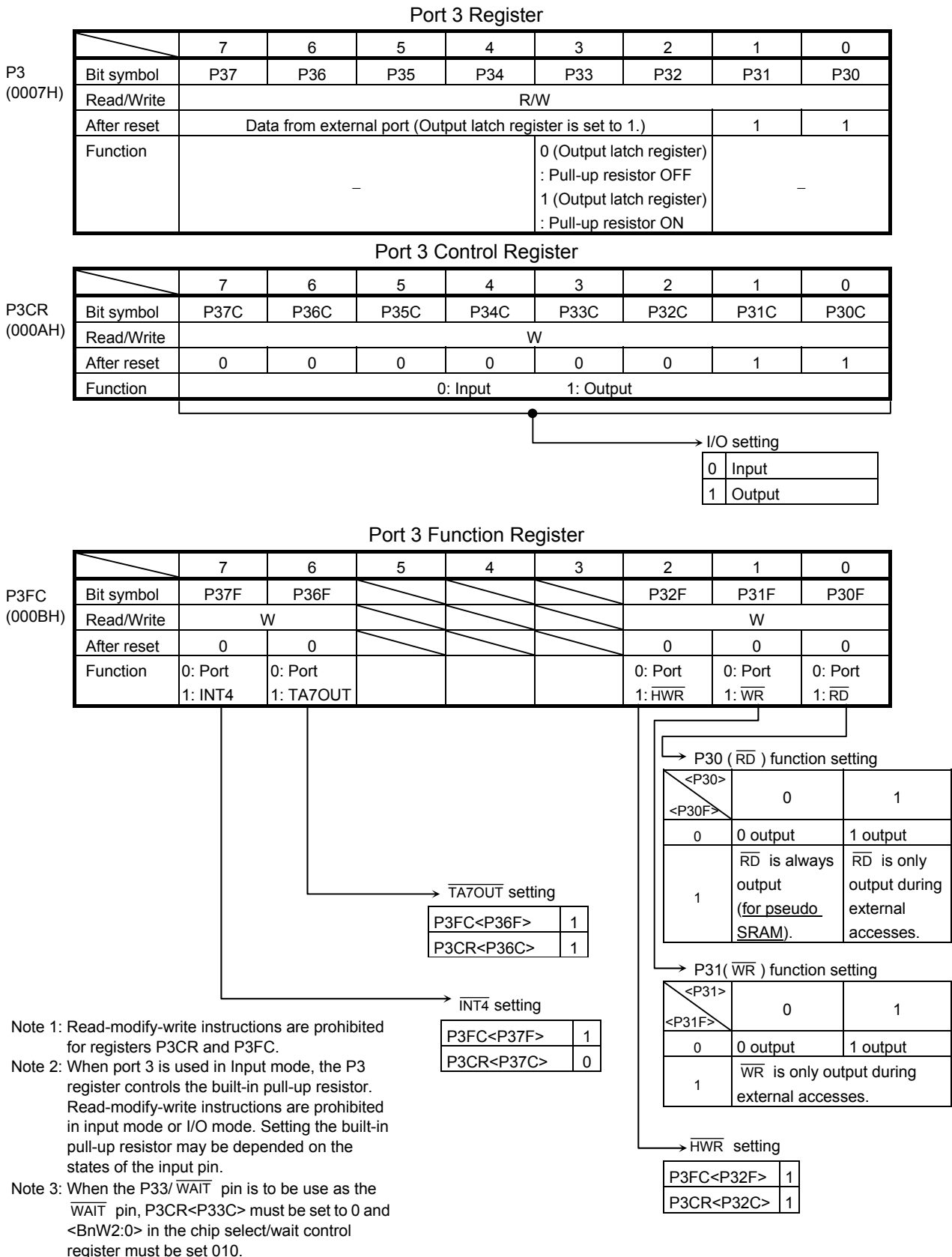


Figure 3.5.13 Register for Port 3 (1/2)

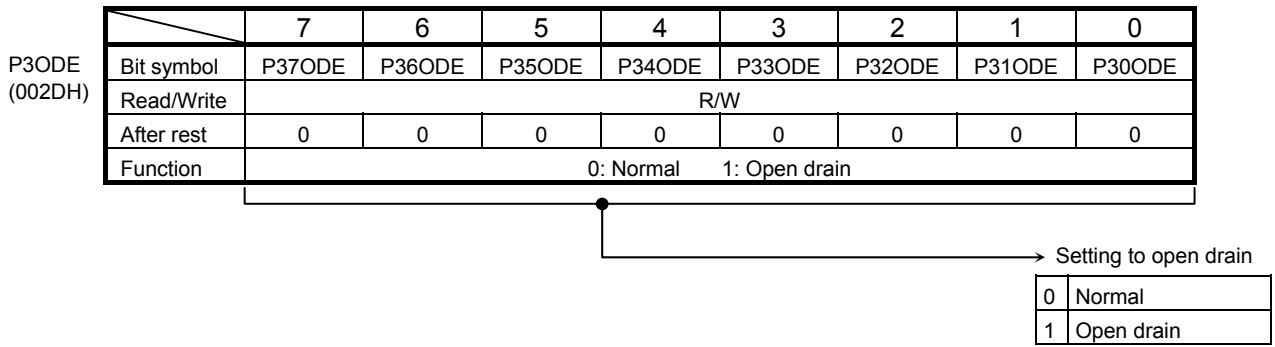


Figure 3.5.14 Register for Port 3 (2/2)

3.5.5 Port 4 (P40 to P43)

Port 4 is a 4-bit input port and can also be used as the analog input pins for the internal AD converter. P40 is also used as the AD trigger input pin of AD converter.

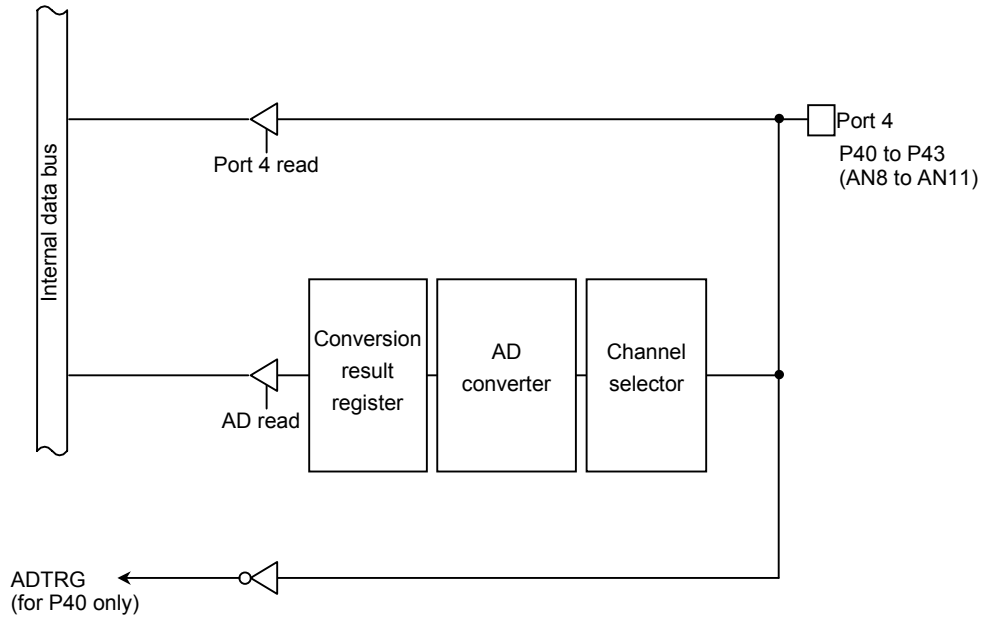


Figure 3.5.15 Port 4

	7	6	5	4	3	2	1	0
P4 (000CH)					P43	P42	P41	P40
Bit symbol								
Read/Write					R			
After reset					Data from external port			

Note: The input channel selection of AD converter and the permission of ADTRG (P40) input are set by AD converter mode register ADMOD1.

Figure 3.5.16 Register for Port 4

3.5.6 Port 5 (P50 to P57)

Port 5 is an 8-bit input port and can also be used as the analog input pins for the internal AD converter.

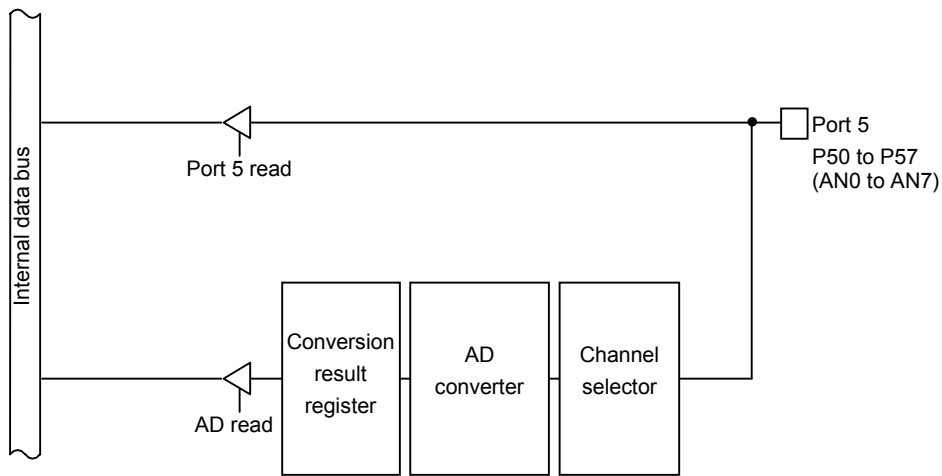


Figure 3.5.17 Port 5

	7	6	5	4	3	2	1	0	
P5 (000DH)	Bit symbol	P57	P56	P55	P54	P53	P52	P51	P50
	Read/Write	R							
	After reset	Data from external port							

Figure 3.5.18 Register for Port 5

3.5.7 Port 6 (P60 to P62)

Port pins 60 to 62 constitute a 3-bit general-purpose I/O port. Each bit can be set individually for input or output. Resetting sets port 6 to be an input port. It also sets all bits of the output latch to 1. In addition to functioning as a general-purpose I/O port, port pins 60 to 62 can also function the external interrupt INT0 to INT2 input, \overline{CTS} input, SCOUT output function. The various functions can each be enabled by writing a 1 to the corresponding bit of the Port 6 function register (P6FC) or SCOUT control register (SCOUTC).

Resetting resets all bits of the registers P6CR and P6FC to 0 and sets all bits to be input port pins.

(1) Port pin 60 (INT0)

Port pin 60 is a general-purpose I/O port pin. It can also be used as the external interrupt INT0 input pin.

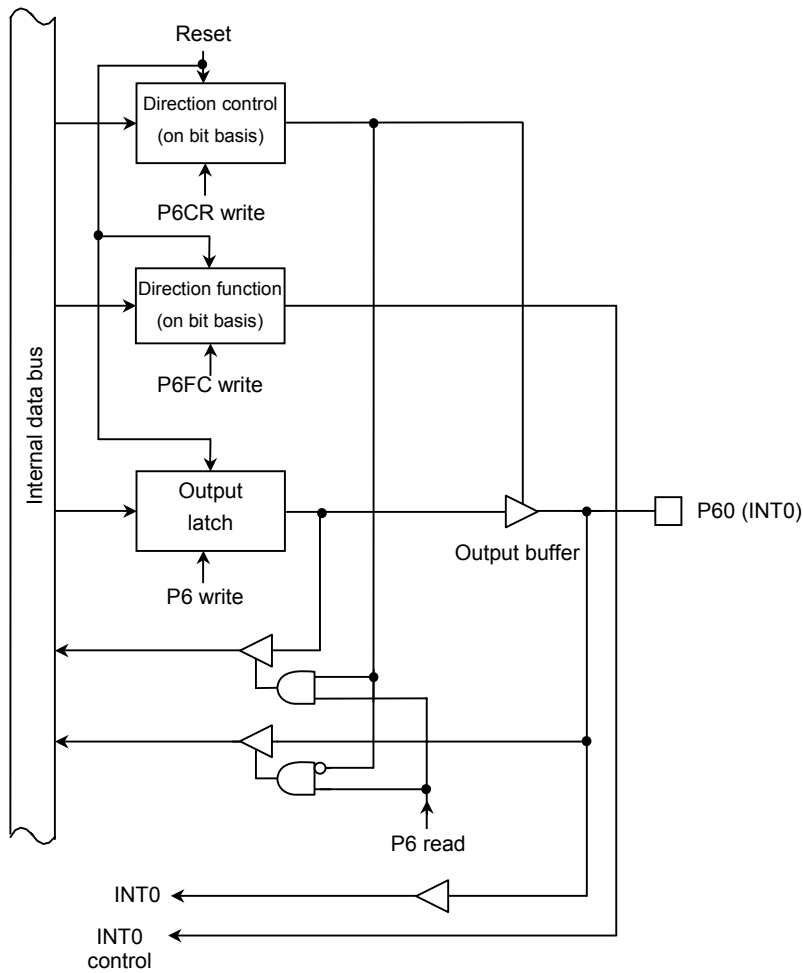


Figure 3.5.19 Port 60

(2) Port pin 61 ($\overline{\text{CTS}}$ /INT1)

Port pin 61 is a general-purpose I/O port pin. It can also be used as the external INT1 input pin or as the $\overline{\text{CST}}$ input pin (in UART mode).

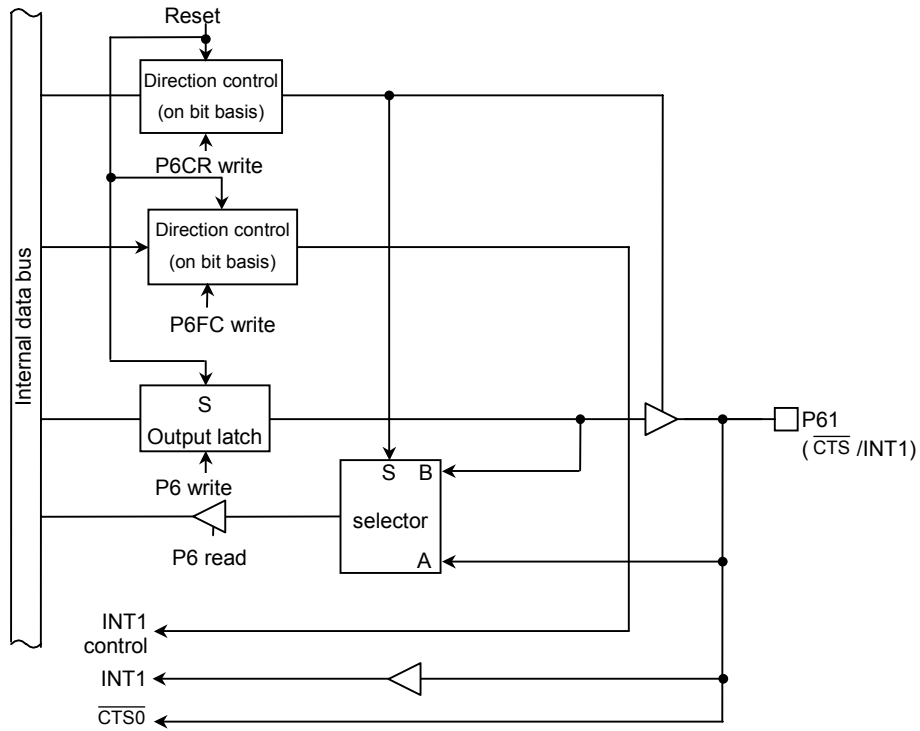


Figure 3.5.20 Port 61

(3) Port pin 62 (INT2/SCOUT)

Port pin 62 is a general-purpose I/O port pin. It can also be used as the external interrupt INT2 input pin or SCOUT output pin function.

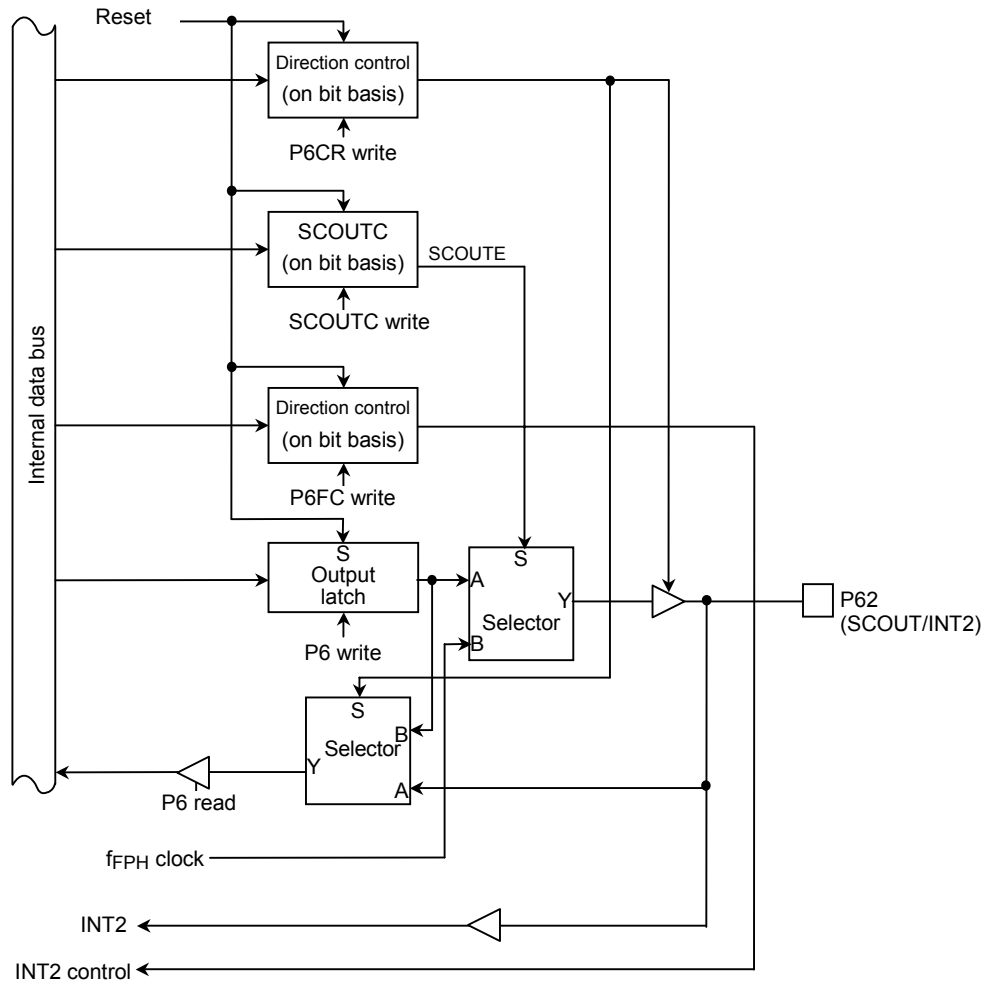


Figure 3.5.21 Port 62

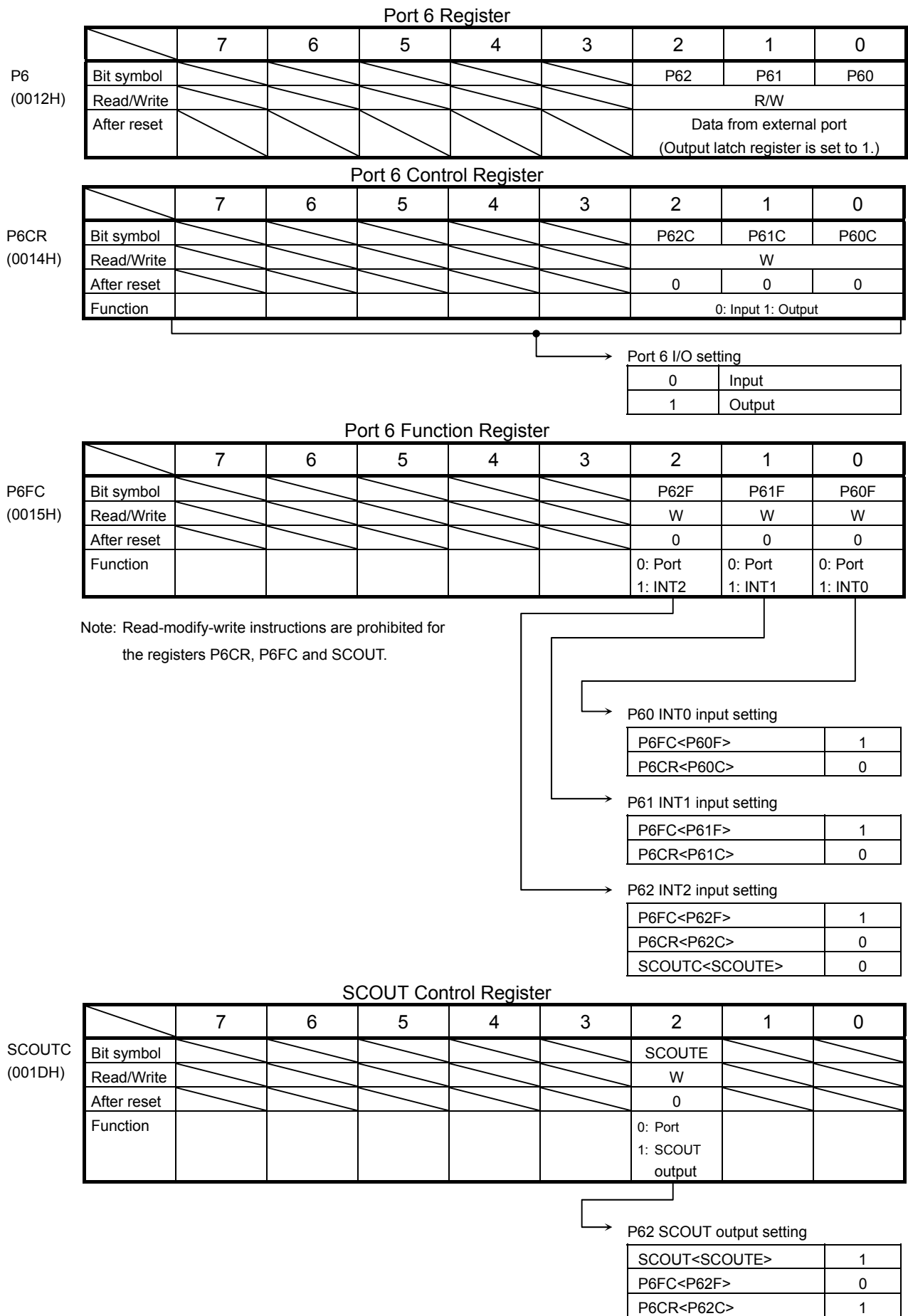


Figure 3.5.22 Registers for Port 6

3.5.8 Port 7 (P70 to P76)

Port 7 is a 7-bit general-purpose I/O port. Each bit can be set individually for input or output. Resetting sets port 7 to be an input port. And the output latch register P7 (All bit) set to 1. In addition to functioning as a general-purpose I/O port, P70, P71 and P72 also functions as an 1, 3, 5 output (TA1OUT, TA3OUT and TA5OUT) of the 8-bit timer A, and port pins 73 and 74 can function as the 16-bit timer clock input INT5 and INT6 input, TB0IN0/INT5 and TB0IN1/INT6. P75 as 16-bit timer output (TB0OUT0), P76 as I/O function of the serial interface 0 (SCK0). For each of the output pins, timer output can be enabled by writing a 1 to the corresponding bit in the port 7 function register (P7FC).

SCK0 output function become available when a proper bit of port 7 function register P7FC is 1 and a proper bit of interrupt control register IIEC is 0.

To use TB0IN0/INT5, TB0IN1/INT6 and SCK0/INT3 pin as external interrupt input pins, a proper bit of interrupt enable register IIEC must be set 1.

By reset, a value of P7CR, P7FC and IIEC become 0 and all bits become input mode.

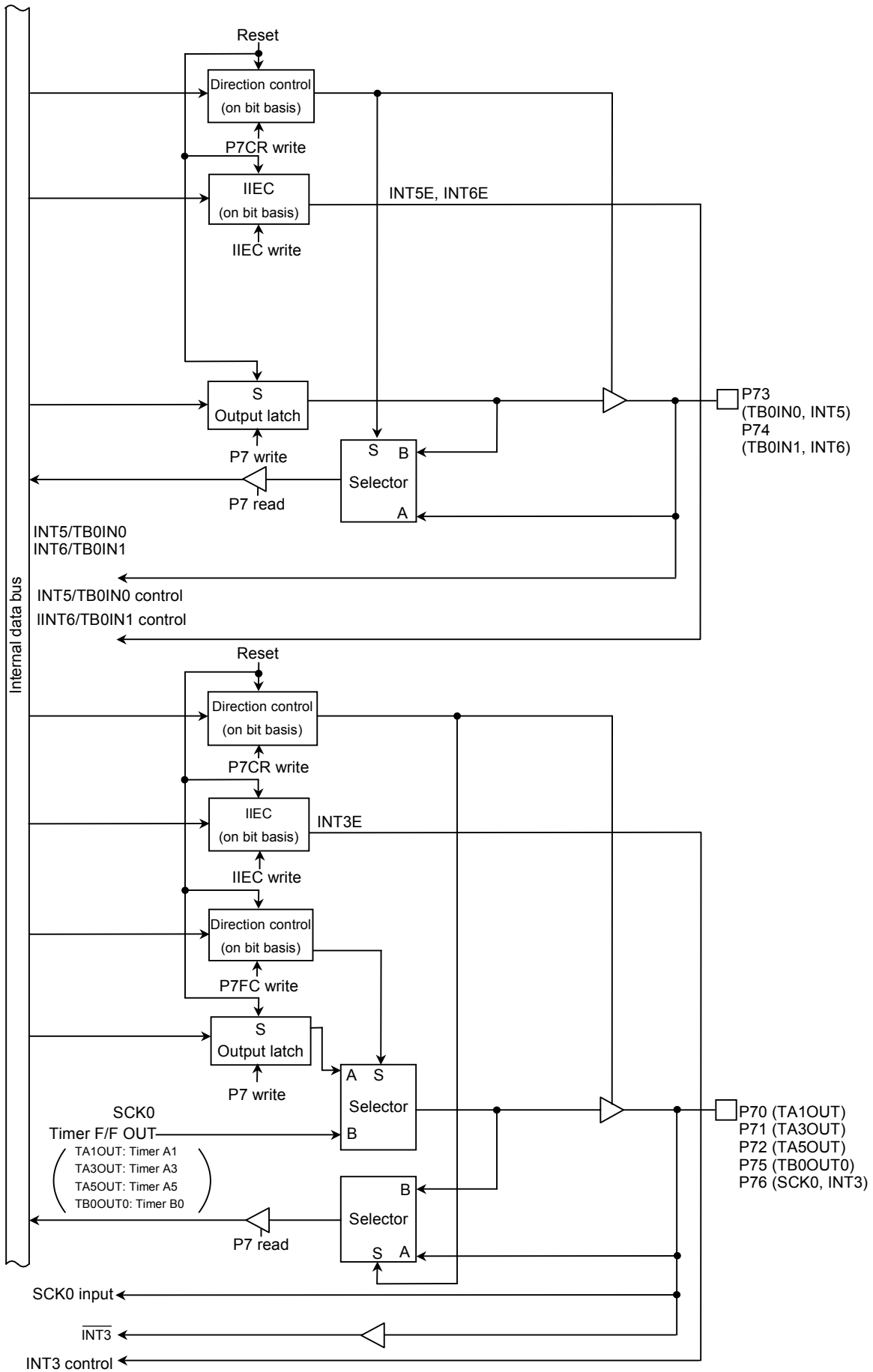


Figure 3.5.23 Port 7

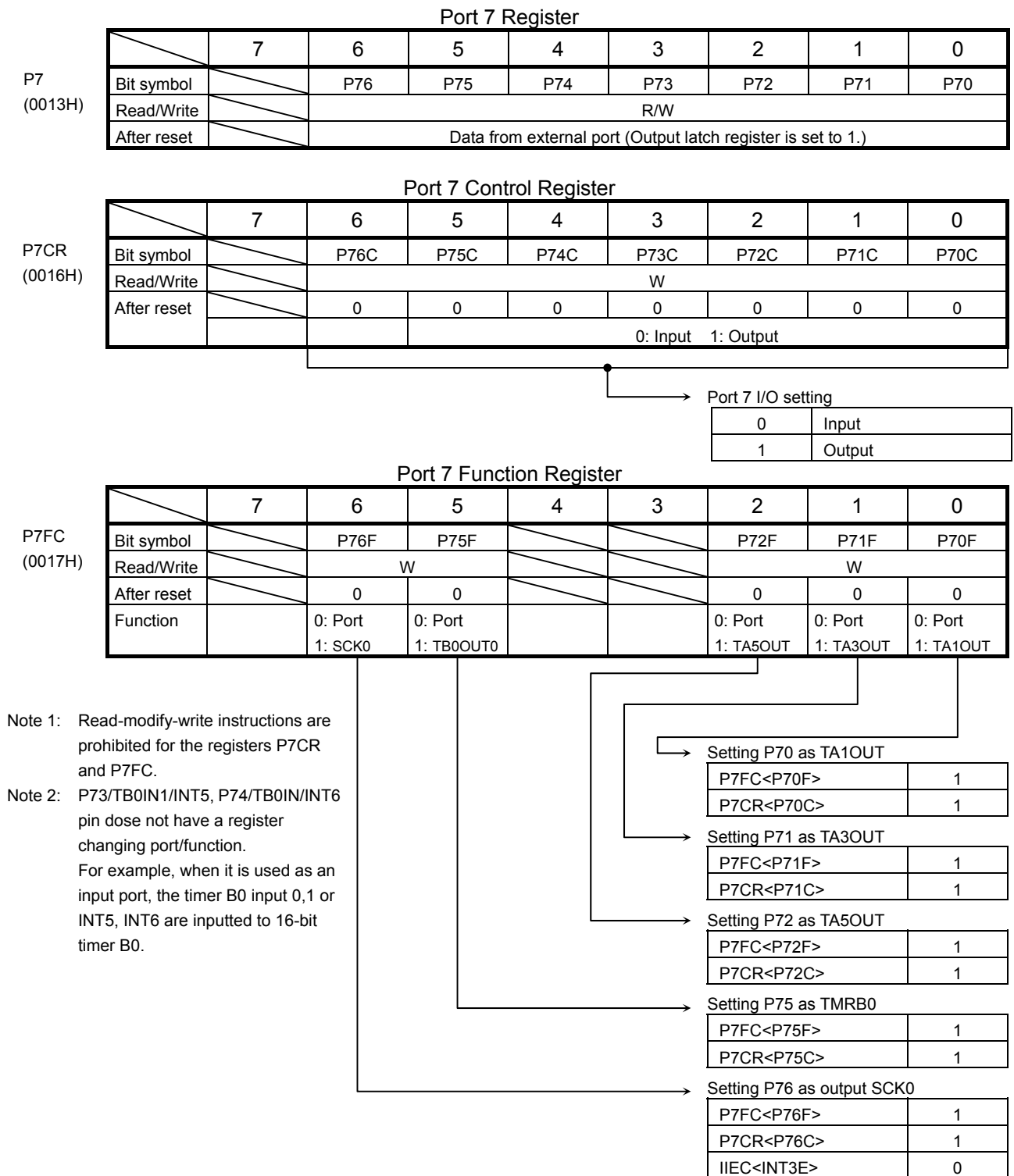


Figure 3.5.24 Port 7 Registers (1/2)

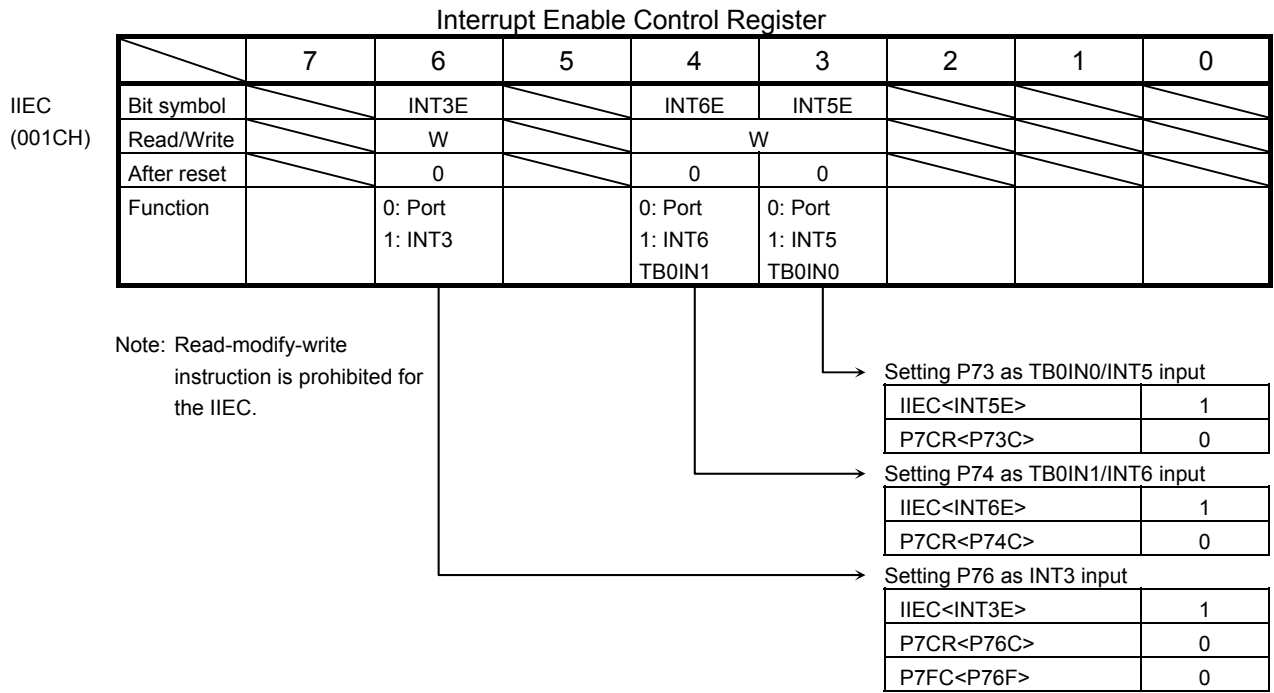


Figure 3.5.25 Port 7 Registers (2/2)

3.5.9 Port 8 (P80 to P87)

Port 8 is an 8-bit general-purpose I/O port. Each bit can be set individually for input or output. Resetting sets port 8 to be an input port. It also sets all bits in the output latch register P8 to P1. Besides I/O function, each port can be used both as another function port as follows P80, P81 are used both as I/O pin SDA0/SO0, SCL0/SIO of I²C bus/SIO.

P82, P83 are used both as I/O pin TXD, RXD of UART. P84 and P85 are used both as I/O pin of SDA1, SCL1 of I²C bus 1. P86, P87 are used both as I/O pin SDA2, SCL2 of I²C bus 2.

These functions can be enabled by writing a 1 to the corresponding bits in the port 8 function register (P8FC). And also, when the output is set for each bit, open-drain is selectable by P8ODE.

Resetting resets all bits of the registers P8CR and P8FC to 0, and sets all bits to be input port pins.

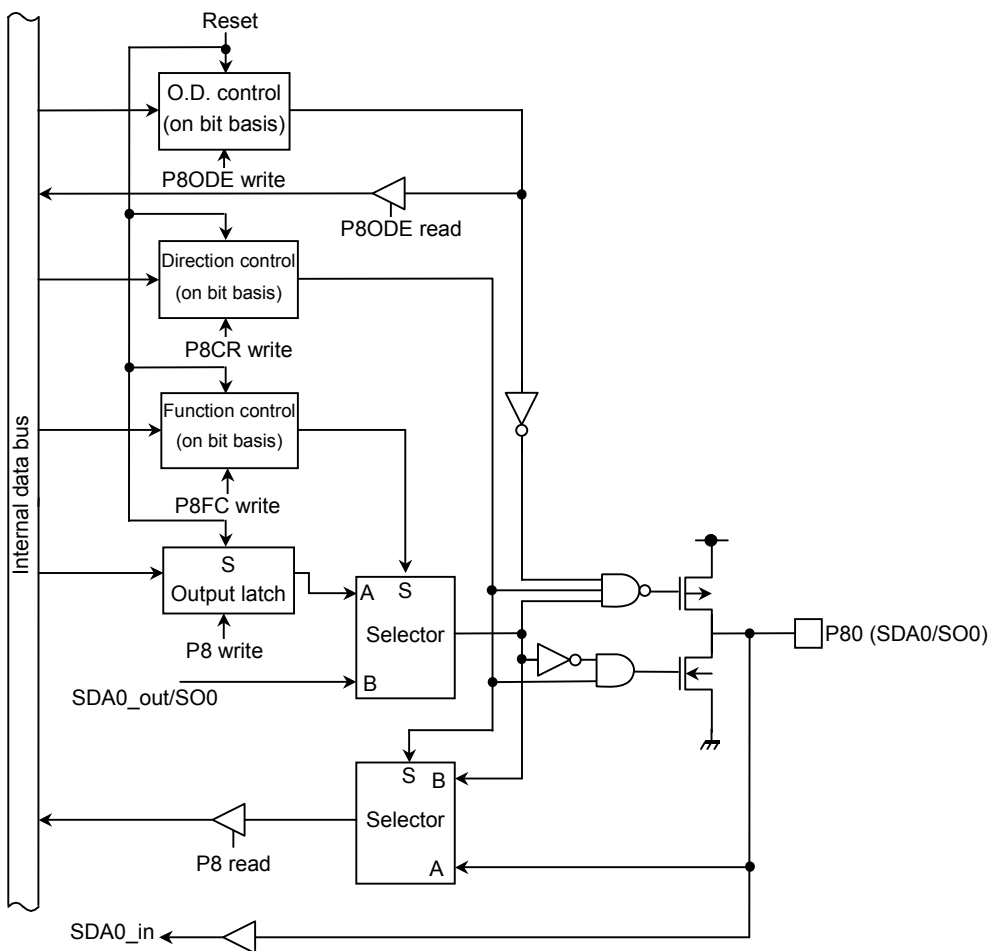


Figure 3.5.26 Port 8 (P80)

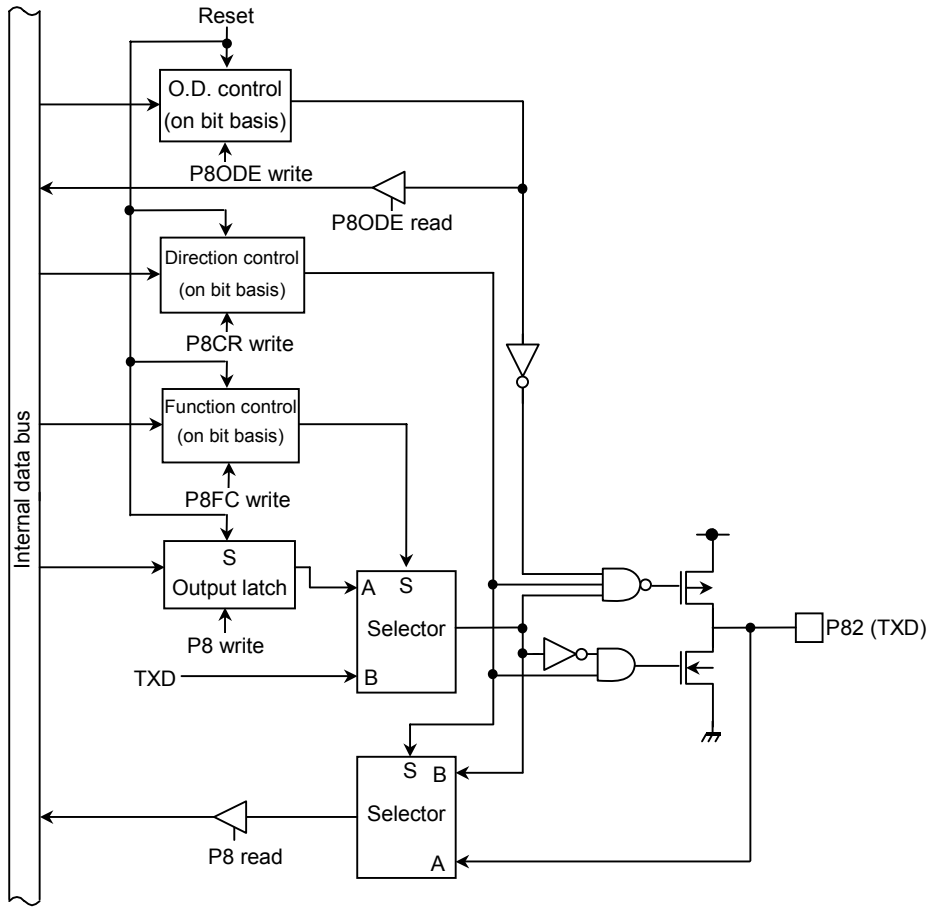


Figure 3.5.28 Port 8 (P82)

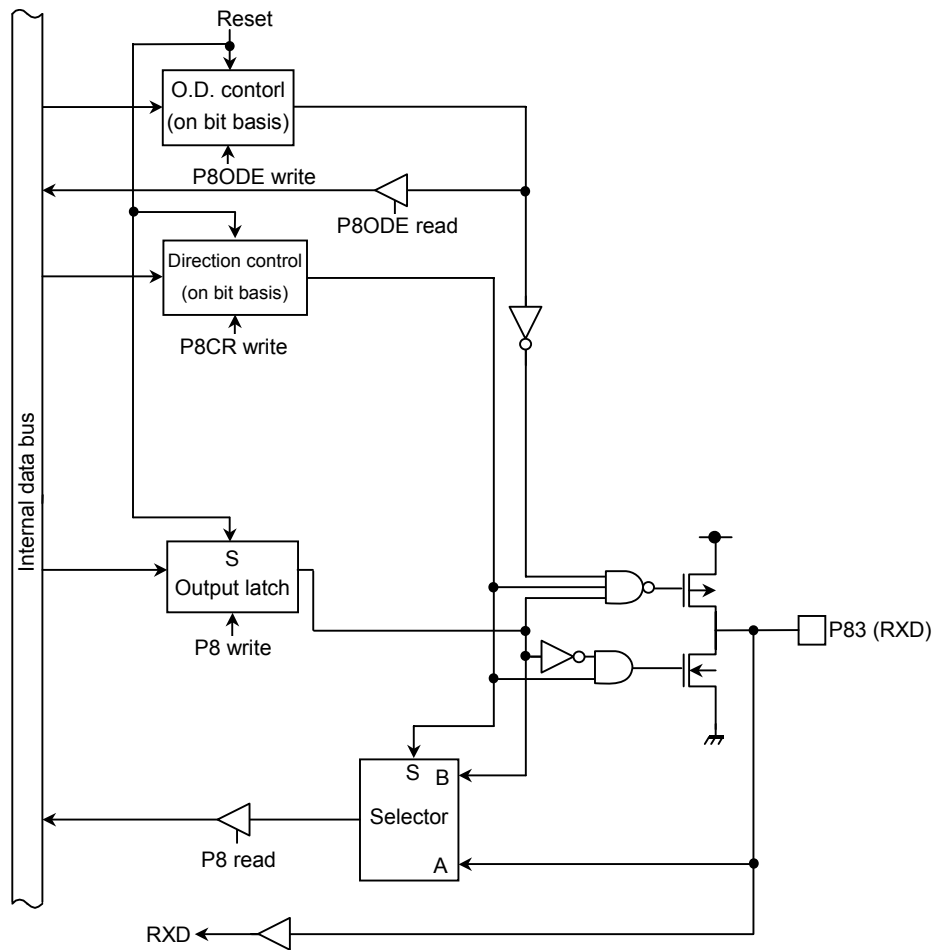


Figure 3.5.29 Port 8 (P83)

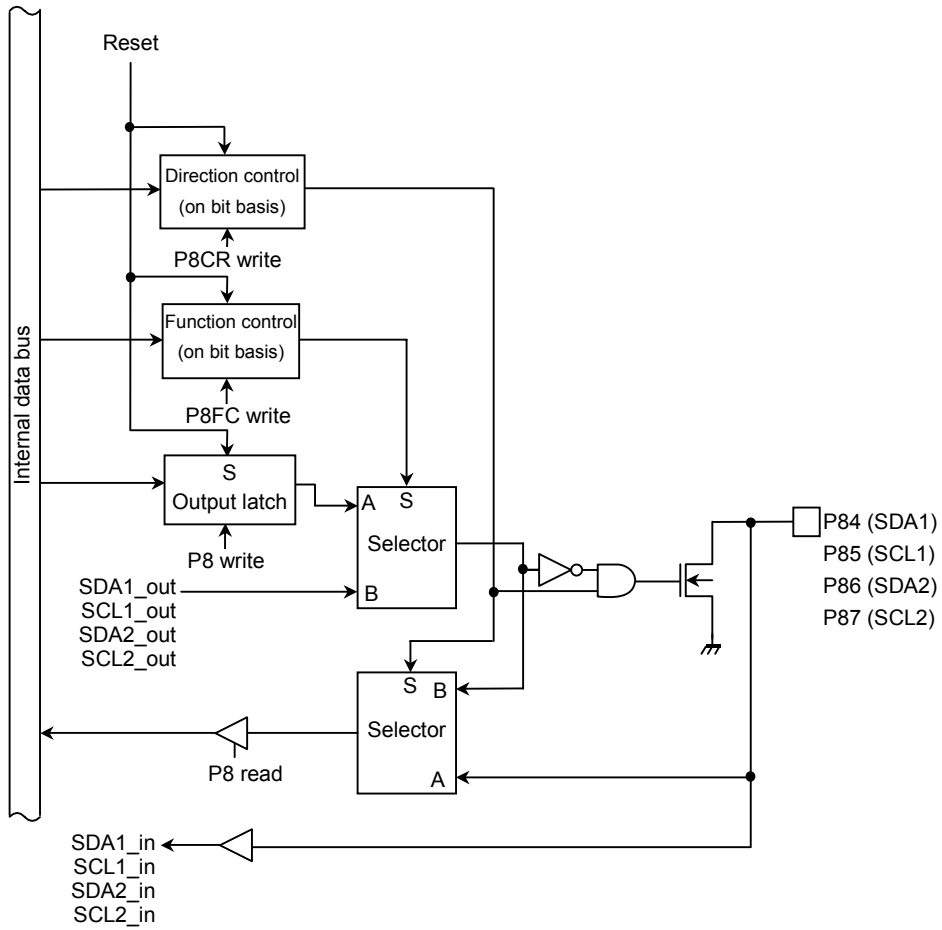


Figure 3.5.30 Port 8 (P84 to P87)

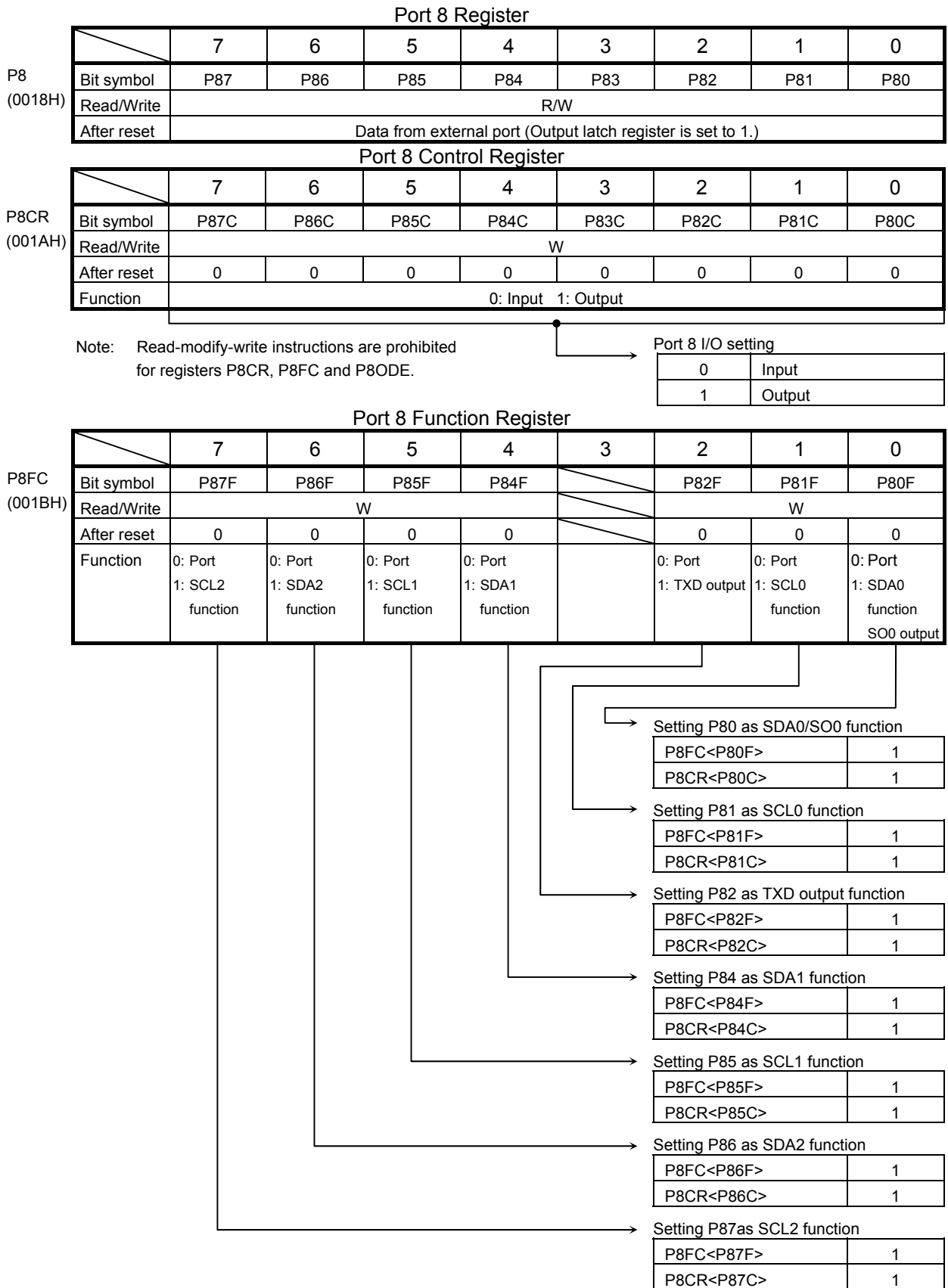


Figure 3.5.31 Port 8 Registers (1/2)

		7	6	5	4	3	2	1	0
P8ODE (002FH)	Bit symbol	/				P83ODE	P82ODE	P81ODE	P80ODE
	Read/Write	/				R/W			
	After reset	/				0	0	0	0
	Function	/				0: Normal 1: Open drain			

Setting to open drain	
0	Normal
1	Open drain

Figure 3.5.32 Port 8 Registers (2/2)

3.6 Wait Controller

On the TMP91CW18A, four user-specifiable address areas (CS0 to CS3) can be set. The data bus width and the number of waits can be set independently for each address area (CS0 to CS3 plus any other).

TMP91CW18A does not have the chip select signal for the specified address area.

In using TMP91CW18A, if chip select signal is needed for each memory space, user have to generate chip select signals by making a external circuit (Address decoder circuit) in order to access external ROM/RAM. 4 blocks address areas are defined by memory start address register MSAR0 to MSAR3 and memory address mask register MAMR0 to MAMR3.

The wait control registers B0CS to B3CS and BEXCS should be used to specify the master enable/disable status the data bus width and the number of waits for each address area.

The input pin which controls these states is the bus wait request pin ($\overline{\text{WAIT}}$).

(After this chapter, 4 block address spaces are described as CS0 space, CS1 space, CS2 space and CS3 space.)

3.6.1 Specifying an Address Area

The address areas CS0 to CS3 are specified using the memory start address registers (MSAR0 to MSAR3) and the memory address mask registers (MAMR0 to MAMR3).

During each bus cycle, a compare operation is performed to determine whether or not the address specified on the bus corresponds to a location in one of the areas CS0 to CS3. If the result of the comparison is a match, it indicates that the corresponding CS area is to be accessed. If so, the bus cycle proceeds according to the settings in the corresponding B0CS to B3CS wait control register. (See 3.6.2 “Wait Control Registers”.)

(1) Memory start address registers

Figure 3.6.1 shows the memory start address registers. The memory start address registers MSAR0 to MSAR3 determine the start addresses for the memory areas CS0 to CS3 respectively. The 8 most significant bits (A23 to A16) of the start address should be set in <S23:16>. The 16 least significant bits of the start address (A15 to A0) are fixed to 0. Thus the start address can only be set to lie on a 64-Kbyte boundary, starting from 000000H. Figure 3.6.2 shows the relationship between the value set in the start address register and the start address.

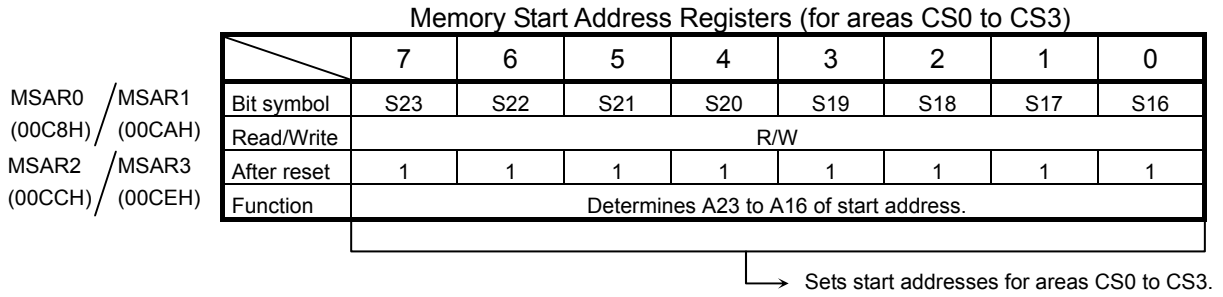


Figure 3.6.1 Memory Start Address Register

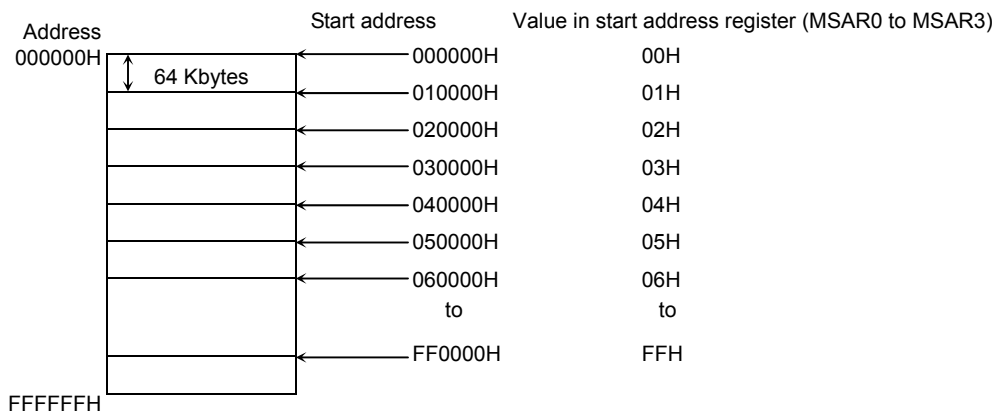


Figure 3.6.2 Relationship between Start Address and Start Address Register Value

(2) Memory address mask registers

Figure 3.6.3 shows the memory address mask registers. The size of each of the areas CS0 to CS3 can be set by specifying a mask in the corresponding memory address mask register (MAMR0 to MAMR3). Each bit in a memory address mask register (MAMR0 to MAMR3) which is set to 1 masks the corresponding bit of the start address which has been set in the corresponding memory start address register (MSAR0 to MSAR3). The compare operation used to determine whether or not a bus address is in one of the areas CS0 to CS3 only compares address bits for which a 0 has been set in the corresponding bit position in the corresponding memory address mask register.

Also, the address bits which each memory address mask register can mask vary from register to register; hence, the possible size settings for the areas CS0 to CS3 differ accordingly.

		Memory Address Mask Register (for CS0 area)							
		7	6	5	4	3	2	1	0
MAMR0 (00C9H)	Bit symbol	V20	V19	V18	V17	V16	V15	V14 to V9	V8
	Read/Write	R/W							
	After reset	1	1	1	1	1	1	1	1
	Function	Sets size of CS0 area 0: Used for address compare							
Range of possible settings for CS0 area size: 256 bytes to 2 Mbytes									
		Memory Address Mask Register (CS1)							
		7	6	5	4	3	2	1	0
MAMR1 (00CBH)	Bit symbol	V21	V20	V19	V18	V17	V16	V15 to V9	V8
	Read/Write	R/W							
	After reset	1	1	1	1	1	1	1	1
	Function	Sets size of CS1 area 0: Used for address compare							
Range of possible settings for CS1 area size: 256 bytes to 4 Mbytes.									
		Memory Address Mask Register (CS2, CS3)							
		7	6	5	4	3	2	1	0
MAMR2 / MAMR3 (00CDH) / (00CFH)	Bit symbol	V22	V21	V20	V19	V18	V17	V16	V15
	Read/Write	R/W							
	After reset	1	1	1	1	1	1	1	1
	Function	Sets size of CS2 or CS3 area 0: Used for address compare							
Range of possible settings for CS2 and CS3 area sizes: 32 Kbytes to 8 Mbytes.									

Figure 3.6.3 Memory Address Mask Registers

(3) Setting memory start addresses and address areas

Figure 3.6.4 shows an example in which CS0 is specified to be a 64-Kbyte address area starting at 010000H.

First, MSAR0<S23:16>, the 8 most significant bits of the start address register and which correspond to the memory start address, are set to 01H. Next, based on the desired CS0 area size, the difference between the start address and the end address (01FFFFH) is calculated. Bits 20 to 8 of this result constitute the mask value for the desired CS0 area size. Setting this value in MAMR0<V20:8> (Bits 20 to 8 of the memory address mask register) sets the desired area size for CS0. In this example 07H is set in MAMR0, specifying an area size of 64 Kbytes.

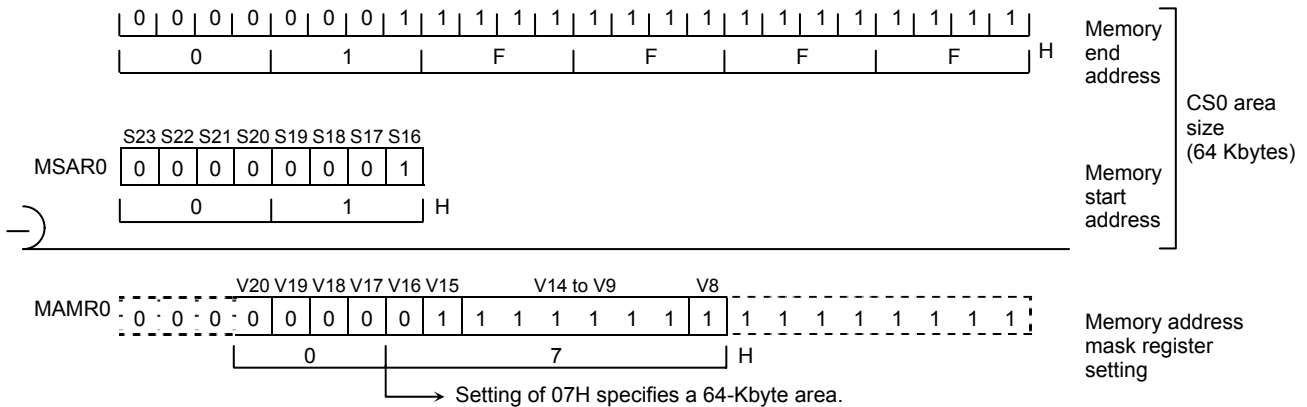


Figure 3.6.4 Example Showing How to Set the CS0 Area

A reset sets MSAR0 to MSAR3 and MAMR0 to MAMR3 to FFH. In addition, B0CS<B0E>, B1CS<B1E> and B3CS<B3E> are cleared to 0, disabling the CS0, CS1 and CS3 areas. However, since a reset clears B2CS<B2M> to 0 and sets B2CS<B2E> to 1, CS2 is enabled with the address range 002000H to FDFFFFH. When addresses outside the areas specified as CS0 to CS3 are accessed, the bus width and number of waits specified in BEXCS are used. (See 3.6.2 “Wait Control Registers”.)

(4) Address area size specification

Table 3.6.1 shows the valid area sizes for each CS area and indicates which method can be used to make the size setting. A “Δ” indicates that it is not possible to set the area size in question using the memory start address register and memory address mask register. If an area size for a CS area marked “Δ” in the table is to be set, the start address must either be set to 000000H or to a value that is greater than 000000H by an integer multiple of the desired area size.

If the CS2 area is set to 16 Mbytes or if two or more areas overlap, the lowest-numbered CS area has highest priority (e.g., CS0 has a higher priority than any other area).

Example: To set the area size for CS0 to 128 Kbytes:

(1) Valid start addresses

000000H)	128 Kbytes	
020000H)	128 Kbytes	Any of these addresses may be set as the start address.
040000H)	128 Kbytes	
060000H)	128 Kbytes	
⋮			

(2) Invalid start addresses

000000H)	64 Kbytes	← This is not an integer multiple of the desired area size setting. Hence, none of these addresses can be set as the start address.
010000H)	128 Kbytes	
030000H)	128 Kbytes	
050000H)		
⋮			

Table 3.6.1 Valid Area Sizes for Each CS Area

CS Area \ Size (Bytes)	256	512	32 K	64 K	128 K	256 K	512 K	1 M	2 M	4 M	8 M
CS0	○	○	○	○	Δ	Δ	Δ	Δ	Δ		
CS1	○	○		○	Δ	Δ	Δ	Δ	Δ	Δ	
CS2			○	○	Δ	Δ	Δ	Δ	Δ	Δ	Δ
CS3			○	○	Δ	Δ	Δ	Δ	Δ	Δ	Δ

3.6.2 Wait Control Registers

Figure 3.6.5 lists the wait control registers.

The master enable/disable, data bus width and number of wait states for each address area (CS0 to CS3 plus any other) are set in the respective wait control registers, B0CS to B3CS or BEXCS.

		7	6	5	4	3	2	1	0
B0CS (00C0H) Read-modify-write instructions are prohibited.	Bit symbol	B0E	 	-	-	B0BUS	B0W2	B0W1	B0W0
	Read/Write	W	 	W					
	After reset	0	 	0	0	0	0	0	0
	Function	0: Disable 1: Enable	 	Always write 0		Data bus width 0: 16 bits 1: 8 bits	Number of waits 000: 2 waits 100 001: 1 wait 101 010: (1 + N) waits 110 011: 0 waits 111		
							} Invalid settings		
B1CS (00C1H) Read-modify-write instructions are prohibited.	Bit symbol	B1E	 	-	-	B1BUS	B1W2	B1W1	B1W0
	Read/Write	W	 	W					
	After reset	0	 	0	0	0	0	0	0
	Function	0: Disable 1: Enable	 	Always write 0		Data bus width 0: 16 bits 1: 8 bits	Number of waits 000: 2 waits 100 001: 1 wait 101 010: (1 + N) waits 110 011: 0 waits 111		
							} Invalid settings		
B2CS (00C2H) Read-modify-write instructions are prohibited.	Bit symbol	B2E	B2M	-	-	B2BUS	B2W2	B2W1	B2W0
	Read/Write	W							
	After reset	1	0	0	0	0	0	0	0
	Functions	0: Disable 1: Enable	CS2 area selection 0: 16-Mbyte area 1: CS area	Always write 0		Data bus width 0: 16 bits 1: 8 bits	Number of waits 000: 2 waits 100 001: 1 wait 101 010: (1 + N) waits 110 011: 0 waits 111		
							} Invalid settings		
B3CS (00C3H) Read-modify-write instructions are prohibited.	Bit symbol	B3E	 	-	-	B3BUS	B3W2	B3W1	B3W0
	Read/Write	W	 	W					
	After reset	0	 	0	0	0	0	0	0
	Functions	0: Disable 1: Enable	 	Always write 0		Data bus width 0: 16 bits 1: 8 bits	Number of waits 000: 2 waits 100 001: 1 wait 101 010: (1 + N) waits 110 011: 0 waits 111		
							} Invalid settings		
BEXCS (00C7H) Read modify write instructions are prohibited.	Bit symbol	 	 	 	 	BEXBUS	BEXW2	BEXW1	BEXW0
	Read/Write	 	 	 	 	W			
	After reset	 	 	 	 	0	0	0	0
	Functions	 	 	 	 	Data bus width 0: 16 bits 1: 8 bits	Number of waits 000: 2 waits 100 001: 1 wait 101 010: (1 + N) waits 110 011: 0 waits 111		
							} Invalid settings		

Master enable bit

0	16-Mbyte area
1	CS area enable

CS2 area selection

0	16-Mbyte area
1	Specified address area

Number of address area waits
(See 3.6.2 (3) "Wait control".)

Data bus width selection

0	16-bit data bus
1	8-bit data bus

Figure 3.6.5 Chip Select/Wait Control Registers

(1) Master enable bits

Bit7 (<B0E>, <B1E>, <B2E> or <B3E>) of the wait control register is the master bit which is used to enable or disable settings for the corresponding address area. Writing 1 to this bit enables the settings. A reset disables <B0E>, <B1E> and <B3E> (e.g., clears them to 0) and enables <B2E> (e.g., sets it to 1). Hence after a reset only the CS2 area is enabled.

(2) Data bus width selection

Bit3 (<B0BUS>, <B1BUS>, <B2BUS>, <B3BUS> or <BEXBUS>) of a wait control register specifies the width of the data bus. This bit should be clear to 0 when memory is to be accessed using a 16-bit data bus, and to 1 when an 8-bit data bus is to be used.

This process of changing the data bus width according to the address being accessed is known as dynamic bus sizing. For details of this bus operation, see Figure 3.6.2.

Table 3.6.2 Dynamic Bus Sizing

Operand Data Bus Width	Operand Start Address	Memory Data Bus Width	CPU Address	CPU Data	
				D15 to D8	D7 to D0
8 bits	2n + 0 (Even number)	8 bits	2n + 0	xxxxx	b7 to b0
		16 bits	2n + 0	xxxxx	b7 to b0
	2n + 1 (Odd number)	8 bits	2n + 1	xxxxx	b7 to b0
		16 bits	2n + 1	b7 to b0	xxxxx
16 bits	2n + 0 (Even number)	8 bits	2n + 0	xxxxx	b7 to b0
			2n + 1	xxxxx	b15 to b8
	16 bits	2n + 0	b15 to b8	b7 to b0	
	2n + 1 (Odd number)	8 bits	2n + 1	xxxxx	b7 to b0
			2n + 2	xxxxx	b15 to b8
	16 bits	2n + 1	b7 to b0	xxxxx	
	2n + 2	xxxxx	b15 to b8		
32 bits	2n + 0 (Even number)	8 bits	2n + 0	xxxxx	b7 to b0
			2n + 1	xxxxx	b15 to b8
		2n + 2	xxxxx	b23 to b16	
		2n + 3	xxxxx	b31 to b24	
	16 bits		2n + 0	b15 to b8	b7 to b0
			2n + 2	b31 to b24	b23 to b16
	2n + 1 (Odd number)	8 bits	2n + 1	xxxxx	b7 to b0
			2n + 2	xxxxx	b15 to b8
		2n + 3	xxxxx	b23 to b16	
		2n + 4	xxxxx	b31 to b24	
16 bits			2n + 1	b7 to b0	xxxxx
			2n + 2	b23 to b16	b15 to b8
		2n + 4	xxxxx	b31 to b24	

Input data in bit positions marked xxxxx is ignored during a read. During a write, the bus lines corresponding to these bit positions go high-impedance and the write strobe signal for the bus remains inactive.

(3) Wait control

Bits 0 to 2 (<B0W0:2>, <B1W0:2>, <B2W0:2>, <B3W0:2> or <BEXW0:2>) of a wait control register specify the number of waits that are to be inserted when the corresponding memory area is accessed.

The following types of wait operation can be specified using these bits. Bit settings other than those listed in the table should not be made.

Table 3.6.3 Wait Operation Settings

<BxW2:0>	Number of Waits	Wait Operation
000	2 waits	Inserts a wait of two states, irrespective of the $\overline{\text{WAIT}}$ pin state.
001	1 wait	Inserts a wait of one state, irrespective of the $\overline{\text{WAIT}}$ pin state.
010	(1 + N) waits	Inserts one wait state, then continuously samples the state of the $\overline{\text{WAIT}}$ pin. While the $\overline{\text{WAIT}}$ pin remains low, the wait continues; the bus cycle is prolonged until the pin goes high.
011	0 waits	Ends the bus cycle without a wait, regardless of the $\overline{\text{WAIT}}$ pin state.

A Reset sets these bits to 000 (2 waits).

(4) Bus width and wait control for an area other than CS0 to CS3

The wait control register BEXCS controls the bus width and number of waits when memory locations which are not in one of the four user-specified address areas (CS0 to CS3) are accessed. The BEXCS register settings are always enabled for areas other than CS0 to CS3.

(5) Selecting 16-Mbyte area/specified address area

Clearing B2CS<B2M> (Bit6 of the wait control register for CS2) to 0 designates the 16-Mbyte area 002000H to FDFFFFH as the CS2 area. Setting B2CS<B2M> to 1 designates the address area specified by the start address register MSAR2 and the address mask register MAMR2 as CS2 (e.g., if B2CS<B2M> = 1, CS2 is specified in the same manner as CS0, CS1 and CS3 are).

A reset clears this bit to 0, specifying CS2 as a 16-Mbyte address area.

(6) Procedure for setting wait control

When using the wait control function, set the registers in the following order.

- (1) Set the memory start address registers MSAR0 to MSAR3.

Set the start addresses for CS0 to CS3.

- (2) Set the memory address mask registers MAMR0 to MAMR3.

Set the sizes of CS0 to CS3.

- (3) Set the wait control registers B0CS to B3CS.

Set the data bus width, number of waits and master enable/disable status for CS0 to CS3.

If a CS0 to CS3 address is specified which is actually an internal I/O, RAM or ROM area address, the CPU accesses the internal address area.

Setting example:

In this example CS0 is set to be the 64-Kbyte area 010000H to 01FFFFH. The bus width is set to 16 bits and the number of waits is set to 0.

MSAR0 = 01HStart address: 010000H

MAMR0 = 07HAddress area: 64 Kbytes

BOCS = 83HROM/SRAM, 16-bit data bus, 0 waits, CS0 area settings
enabled

3.6.3 Connecting External Memory

Figure 3.6.6 shows an example of how to connect external memory to the TMP91CW18A.

In this example the ROM is connected using a 16-bit bus. The RAM and I/O are connected using an 8-bit bus.

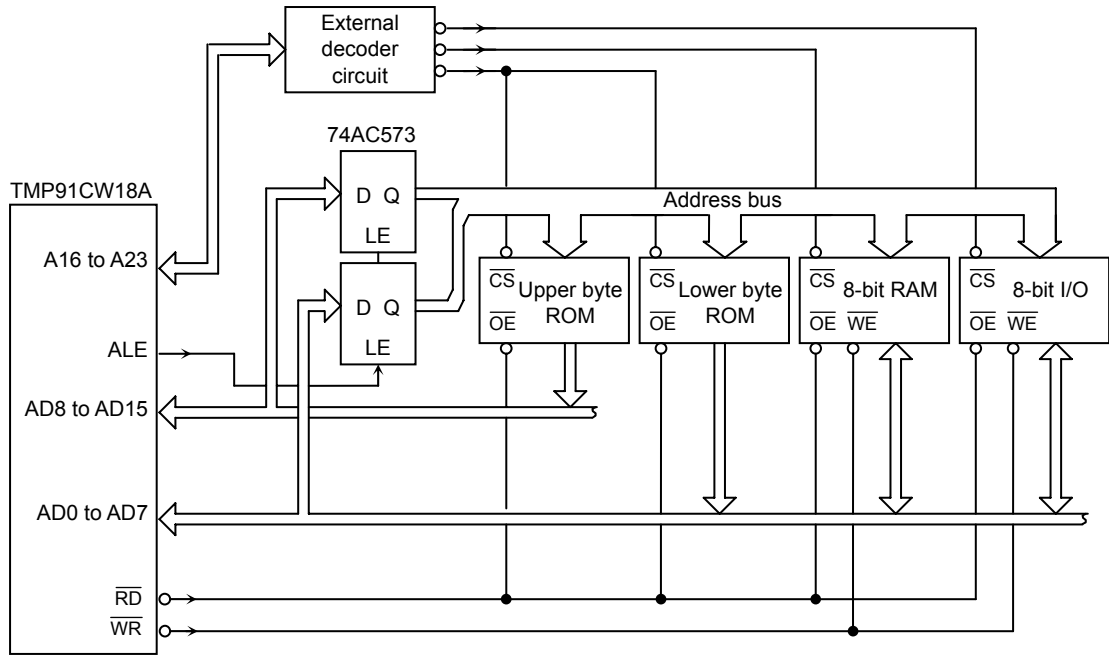


Figure 3.6.6 Example of External Memory Connection
(ROM uses 16-bit bus: RAM and I/O use 8-bit bus.)

Since the MCU has no CS pins, user have to make chip select signals by making a external address decoder circuit in order to control external memory area which is decided by memory start address register and memory address mask register.

3.7 8-Bit Timer (TMRA)

The TMP91CW18A features eight built-in 8-bit timer.

These timers are paired into four modules: TMRA01, TMRA23, TMRA45 and TMRA67. Each module consists of two channels and can operate in any of the following four operating modes.

- 8-bit interval timer mode (4 timers)
- 16-bit interval timer mode (2 timers)
- 8-bit programmable square wave pulse generation output mode (PPG – variable duty cycle with variable period) (1 timer)
- 8-bit pulse width modulation output mode (PWM – variable duty cycle with constant period) (1 timer)

Figure 3.7.1 to Figure 3.7.4 show block diagrams for TMRA01, TMRA23, TMRA45 and TMRA67.

Each channel consists of an 8-bit up counter, an 8-bit comparator and an 8-bit timer register. In addition, a timer flip-flop and a prescaler are provided for each pair of channels.

The operation mode and timer flip-flops are controlled by five control SFRs (Special function registers).

Each of the four modules (TMRA01, TMRA23, TMRA45 and TMRA67) can be operated independently. All modules operate in the same manner; hence only the operation of TMRA01 is explained here.

The contents of this chapter is as follows.

3.7.1 Block Diagrams

3.7.2 Operation of Each Circuit

3.7.3 SFRs

3.7.4 Operation in Each Mode

- (1) 8-bit timer mode
- (2) 16-bit timer mode
- (3) 8-bit PPG (Programmable pulse generation) output mode
- (4) 8-bit PWM (Pulse width modulation) output mode
- (5) Setting for each mode

Table 3.7.1 Registers and Pins for Each Module

Module		TMRA01	TMRA23	TMRA45	TMRA67
External pin	Input pin for external clock				TA6IN (Shared with P35)
	Output pin for timer flip-flop	TA1OUT (Shared with P70)	TA3OUT (Shared with P71)	TA5OUT (Shared with P72)	TA7OUT (Shared with P36)
SFR (Address)	Timer run register	TA01RUN (0100H)	TA23RUN (0108H)	TA45RUN (0110H)	TA67RUN (0118H)
	Timer register	TA0REG (0102H) TA1REG (0103H)	TA2REG (010AH) TA3REG (010BH)	TA4REG (0112H) TA5REG (0113H)	TA6REG (011AH) TA7REG (011BH)
	Timer mode register	TA01MOD (0104H)	TA23MOD (010CH)	TA45MOD (0114H)	TA67MOD (011CH)
	Timer flip-flop control register	TA1FFCR (0105H)	TA3FFCR (010DH)	TA5FFCR (0115H)	TA7FFCR (011DH)

3.7.1 Block Diagrams

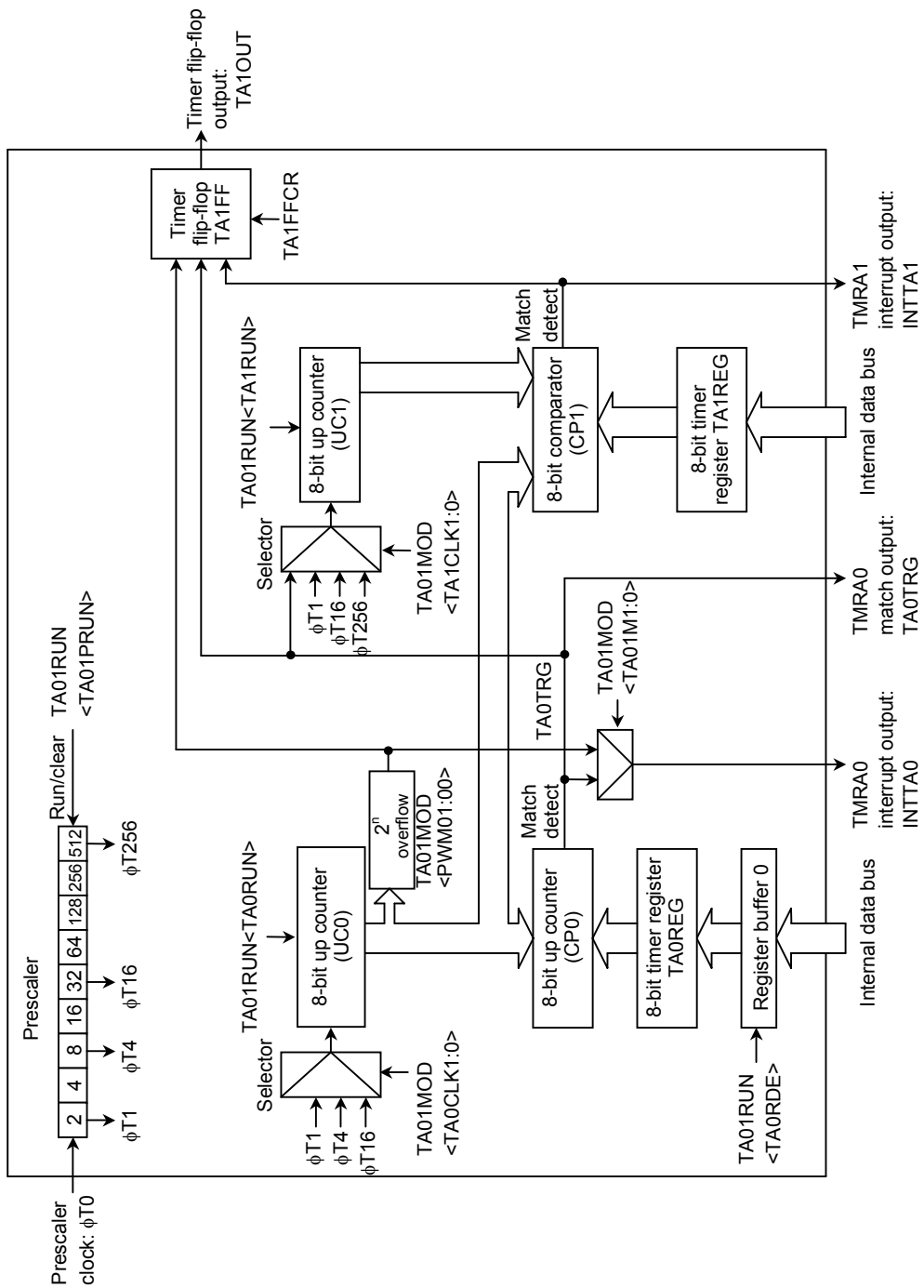


Figure 3.7.1 TMRA01 Block Diagram

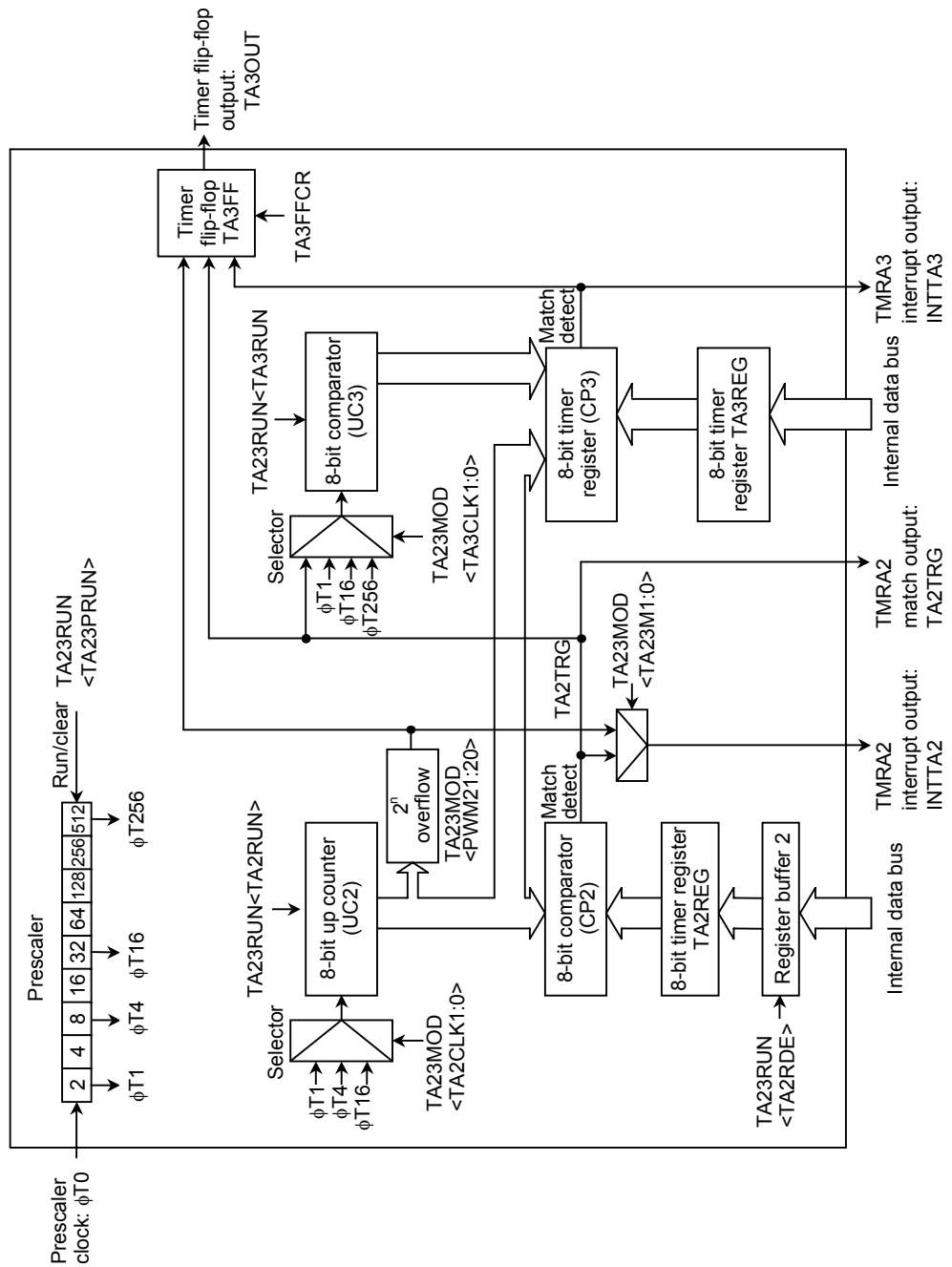


Figure 3.7.2 TMRA23 Block Diagram

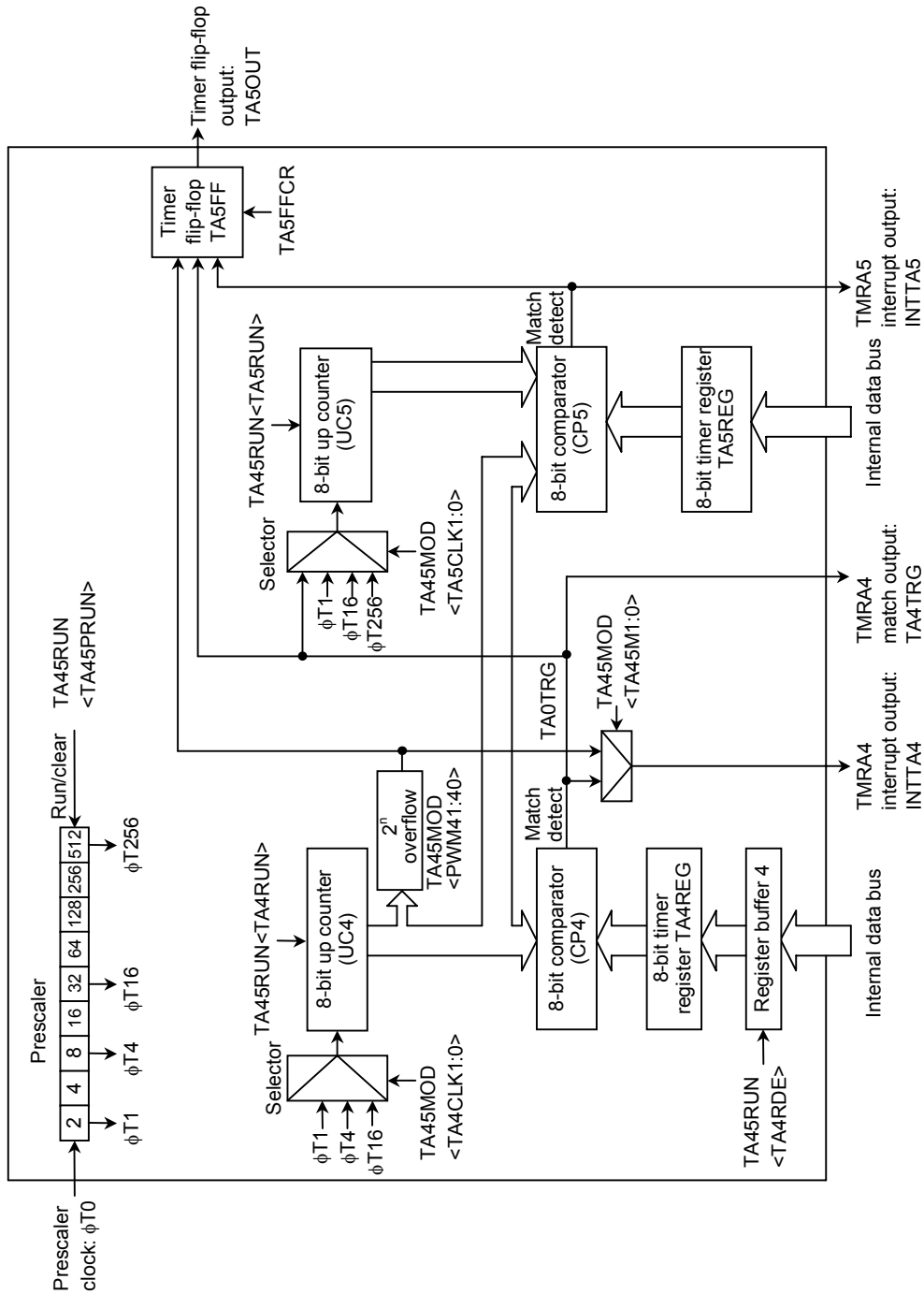


Figure 3.7.3 TMRA45 Block Diagram

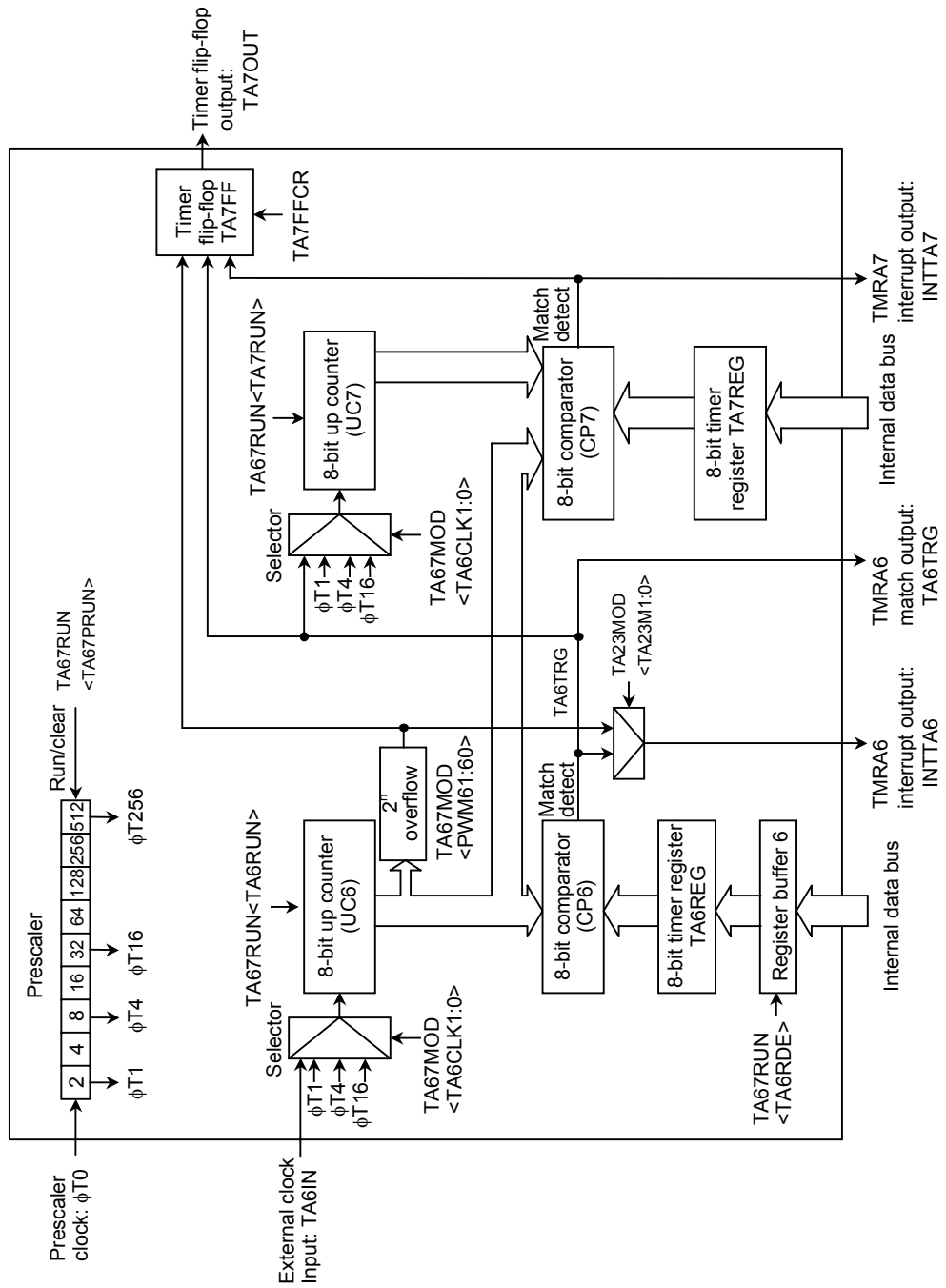


Figure 3.7.4 TMRA67 Block Diagram

3.7.2 Operation of Each Circuit

(1) Prescalers

A 9-bit prescaler generates the input clock to TMRA01.

The clock $\phi T0$ is divided by 4 and input to this prescaler. $\phi T0$ can be either f_{FPH} or $f_c/16$ and is selected using the prescaler clock selection register $SYSCR0<PRCK1:0>$.

The prescaler's operation can be controlled using $TA01RUN<TA01PRUN>$ in the timer control register. Setting $<TA01PRUN>$ to 1 starts the count. Setting $<TA01PRUN>$ to 0 clears the prescaler to 0 and stops operation. Table 3.7.2 shows the various prescaler output clock resolutions.

Table 3.7.2 Prescaler Output Clock Resolution

at $f_c = 25 \text{ MHz}$

System Clock Selection <SYSCK>	Prescaler Clock Selection <PRCK1:0>	Gear Value <GEAR2:0>	Prescaler Output Clock Resolution			
			$\phi T1$	$\phi T4$	$\phi T16$	$\phi T256$
0 (f_c)	00 (f_{FPH})	000 (f_c)	$2^3/f_c$ (0.3 μs)	$2^5/f_c$ (1.3 μs)	$2^7/f_c$ (5.1 μs)	$2^{11}/f_c$ (62.5 μs)
		001 ($f_c/2$)	$2^4/f_c$ (0.6 μs)	$2^6/f_c$ (2.6 μs)	$2^8/f_c$ (10.2 μs)	$2^{12}/f_c$ (163.8 μs)
		010 ($f_c/4$)	$2^5/f_c$ (1.3 μs)	$2^7/f_c$ (5.1 μs)	$2^9/f_c$ (20.5 μs)	$2^{13}/f_c$ (327.7 μs)
		011 ($f_c/8$)	$2^6/f_c$ (2.6 μs)	$2^8/f_c$ (10.2 μs)	$2^{10}/f_c$ (41.0 μs)	$2^{14}/f_c$ (655.4 μs)
		100 ($f_c/16$)	$2^7/f_c$ (5.1 μs)	$2^9/f_c$ (20.5 μs)	$2^{11}/f_c$ (81.9 μs)	$2^{15}/f_c$ (1311 μs)
	10 ($f_c/16$ clock)	XXX	$2^7/f_c$ (5.1 μs)	$2^9/f_c$ (20.5 μs)	$2^{11}/f_c$ (81.9 μs)	$2^{15}/f_c$ (1311 μs)

xxx: Don't care

(2) Up counters (UC0 and UC1)

These are 8-bit binary counters which count up the input clock pulses for the clock specified by $TA01MOD$.

The input clock for UC0 is selectable and can be either the external clock input via the $TA0IN$ pin or one of the three internal clocks $\phi T1$, $\phi T4$ or $\phi T16$. The clock setting is specified by the value set in $TA01MOD<TA01CLK1:0>$.

The input clock for UC1 depends on the operation mode. In 16-bit timer mode, the overflow output from UC0 is used as the input clock. In any mode other than 16-bit timer mode, the input clock is selectable and can either be one of the internal clocks $\phi T1$, $\phi T16$ or $\phi T256$, or the comparator output (the match detection signal) from TMRA0.

For each interval timer, the timer operation control register bits $TA01RUN<TA0RUN>$ and $TA01RUN<TA1RUN>$ can be used to stop and clear the up counters and to control their count. A reset clears both up counters, stopping the timers.

(3) Timer registers (TA0REG and TA1REG)

These are 8-bit registers which can be used to set a time interval. When the value set in the timer register TA0REG or TA1REG matches the value in the corresponding up counter, the comparator match detect signal goes active. If the value set in the timer register is 00H, the signal goes active when the up counter overflows.

The TA0REG are double buffer structure, each of which makes a pair with register buffer.

The setting of the bit TA01RUN<TA0RDE> determines whether TA0REG's double buffer structure is enabled or disabled. It is disabled if <TA0RDE> = 0 and enabled if <TA0RDE> = 1.

When the double buffer is enabled, data is transferred from the register buffer to the timer register when a 2ⁿ overflow occurs in PWM mode or at the start of the PPG cycle in PPG mode. Hence the double buffer cannot be used in timer mode.

A reset initializes <TA0RDE> to 0, disabling the double buffer. To use the double buffer, write data to the timer register, set <TA0RDE> to 1 and write the following data to the register buffer. Figure 3.7.5 shows the configuration of TA0REG.

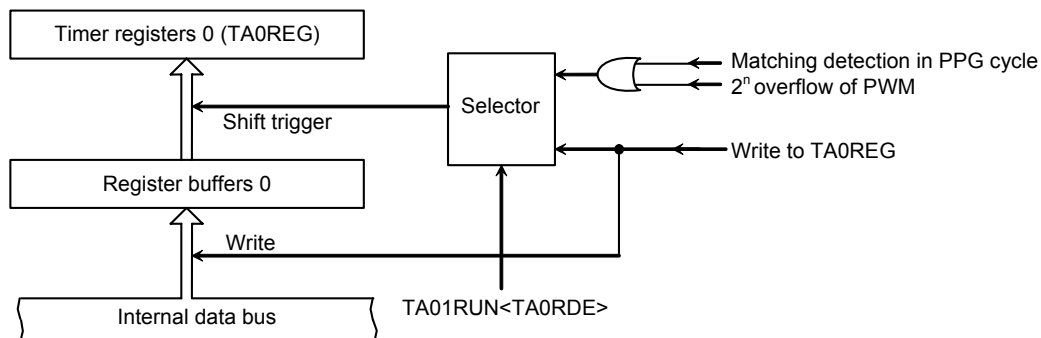


Figure 3.7.5 Configuration of TA0REG

Note: The same memory address is allocated to the timer register and the register buffer. When <TA0RDE> = 0, the same value is written to the register buffer and the timer register, when <TA0RDE> = 1, only the register buffer is written to.

The address of each timer register is as follows.

TA0REG: 000102H	TA1REG: 000103H
TA2REG: 00010AH	TA3REG: 00010BH
TA4REG: 000112H	TA5REG: 000113H
TA6REG: 00011AH	TA7REG: 00011BH

All these registers are write only and cannot be read.

(4) Comparator (CP0, CP1)

The comparator compares the value in an up counter with the value set in a timer register. If they match, the up counter is cleared to 0 and an interrupt signal (INTTA0 or INTTA1) is generated. If timer flip-flop inversion is enabled, the timer flip-flop is inverted at the same time.

(5) Timer flip-flop (TA1FF)

The timer flip-flop (TA1FF) is a flip-flop inverted by the match detect signal (8-bit comparator output) of each interval timer.

Whether inversion is enabled or disabled is determined by the setting of the bit TA1FFCR<TA1FFIE> in the timer flip-flop control register.

A reset clears the value of TA1FF to 0. Writing 01 or 10 to TA1FFCR<TA1FFC1:0> programs TA1FF to 0 or 1. Writing 00 to these bits inverts the value of TA1FF. (This is known as software inversion.)

The TA1FF signal is output via the TA1OUT pin (which can also be used as P71). When this pin is used as the timer output, the timer flip-flop should be set beforehand using the port 7 function register (P7FC).

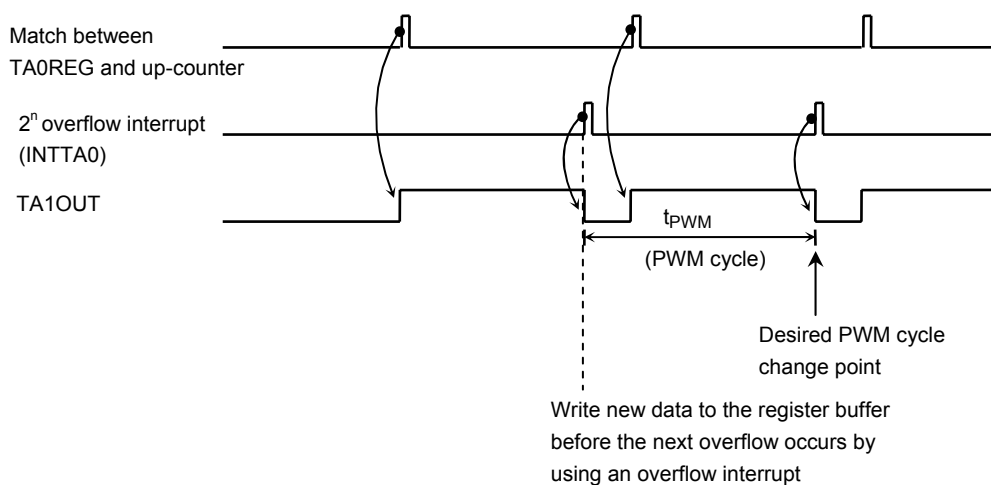
Note: When the double buffer is enabled for an 8-bit timer in PWM or PPG mode, caution is required as explained below.

If new data is written to the register buffer immediately before an overflow occurs by a match between the timer register value and the up-counter value, the timer flip-flop may output an unexpected value.

For this reason, make sure that in PWM mode new data is written to the register buffer by six cycles ($f_{SYS} \times 6$) before the next overflow occurs by using an overflow interrupt.

In the case of using PPG mode, make sure that new data is written to the register buffer by six cycles before the next cycle compare match occurs by using a cycle compare match interrupt.

Example when using PWM mode



3.7.3 SFRs

TMRA01 Run Register

		7	6	5	4	3	2	1	0
TA01RUN (0100H)	Bit symbol	TA0RDE				I2TA01	TA01PRUN	TA1RUN	TA0RUN
	Read/Write	R/W				R/W			
	After reset	0				0	0	0	0
	Function	Double buffer 0: Disable 1: Enable				IDLE2 0: Stop 1: Operate	8-bit timer run/stop control 0: Stop and clear 1: Run (Count up)		

Timer run/stop control

0	Stop and clear
1	Run (Count up)

TA0REG double buffer control

0	Disable
1	Enable

I2TA01: Operation in IDLE2 mode
 TA01PRUN: Run prescaler
 TA1RUN: Run timer 1
 TA0RUN: Run timer 0

Note: The values of bits 4 to 6 of TA01RUN are undefined when read.

TMRA23 Run Register

		7	6	5	4	3	2	1	0
TA23RUN (0108H)	Bit symbol	TA2RDE				I2TA23	TA23PRUN	TA3RUN	TA2RUN
	Read/Write	R/W				R/W			
	After reset	0				0	0	0	0
	Function	Double buffer 0: Disable 1: Enable				IDLE2 0: Stop 1: Run	8-bit timer run/stop control 0: Stop and clear 1: Run (Count up)		

Timer run/stop control

0	Stop and clear
1	Run (Count up)

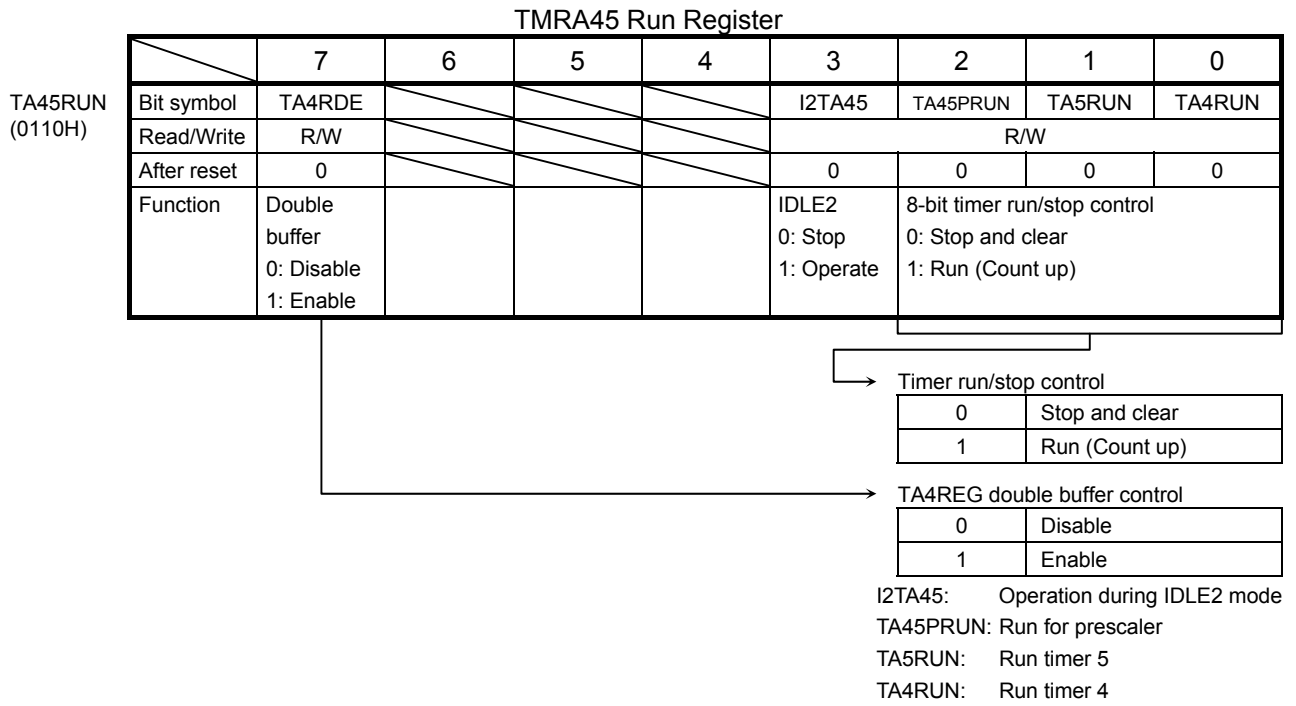
TA2REG double buffer control

0	Disable
1	Enable

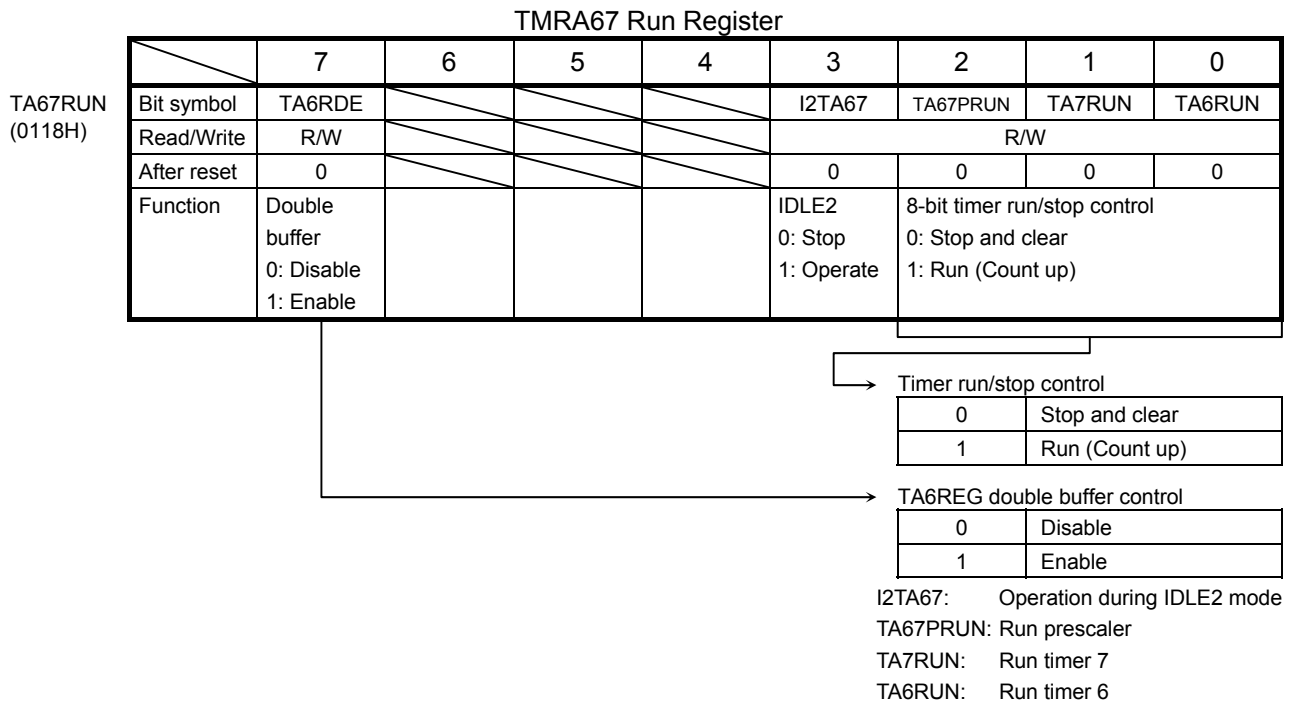
I2TA23: Operation in IDLE2 mode
 TA23PRUN: Run prescaler
 TA3RUN: Run timer 3
 TA2RUN: Run timer 2

Note: The values of bits 4 to 6 of TA23RUN are undefined when read.

Figure 3.7.6 Registers for TMRA



Note: The values of bits 4 to 6 of TA45RUN are undefined when read.



Note: The values of bits 4 to 6 TA67RUN are undefined when read.

Figure 3.7.7 TMRA Registers

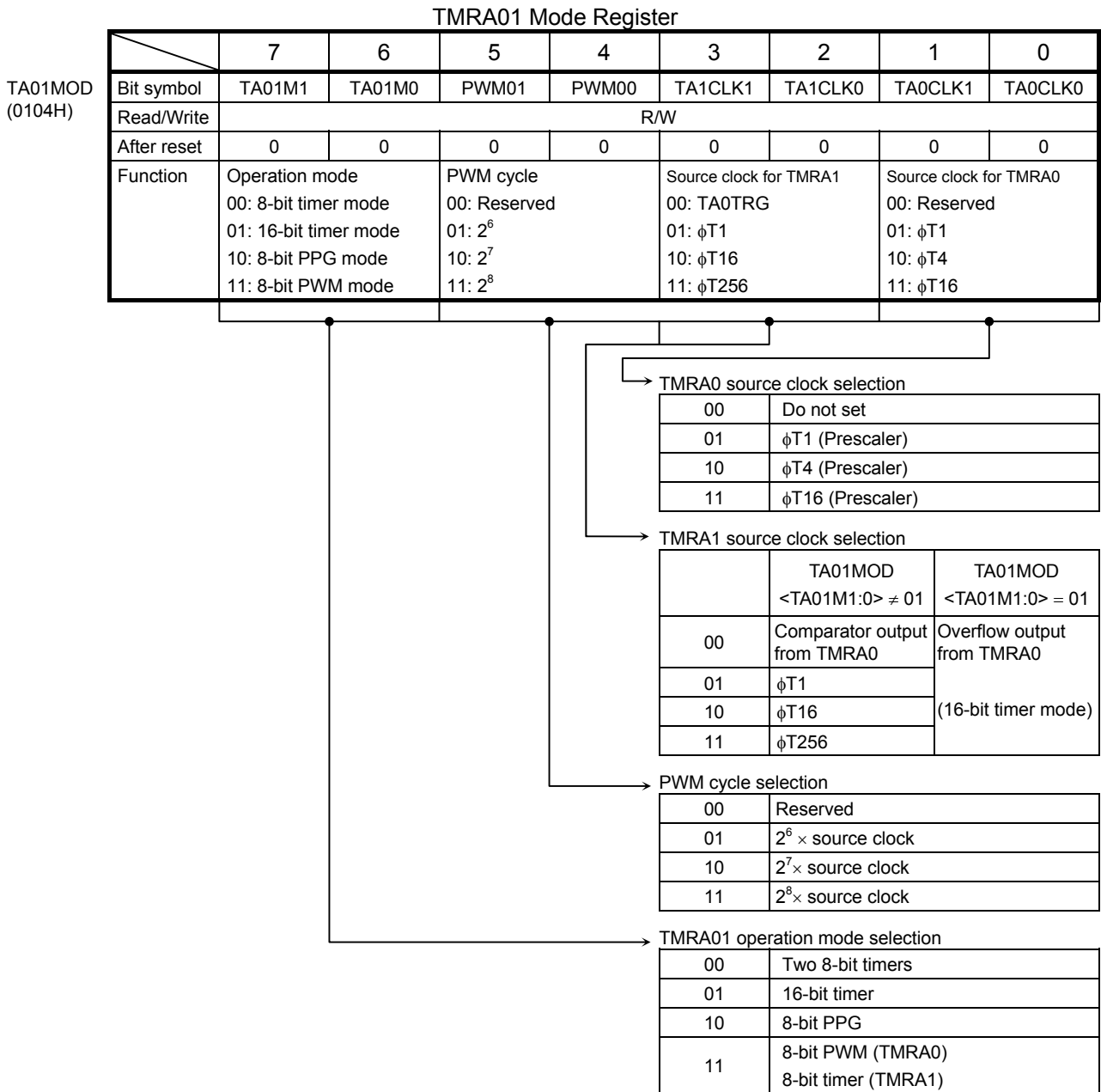


Figure 3.7.8 TMRA Registers

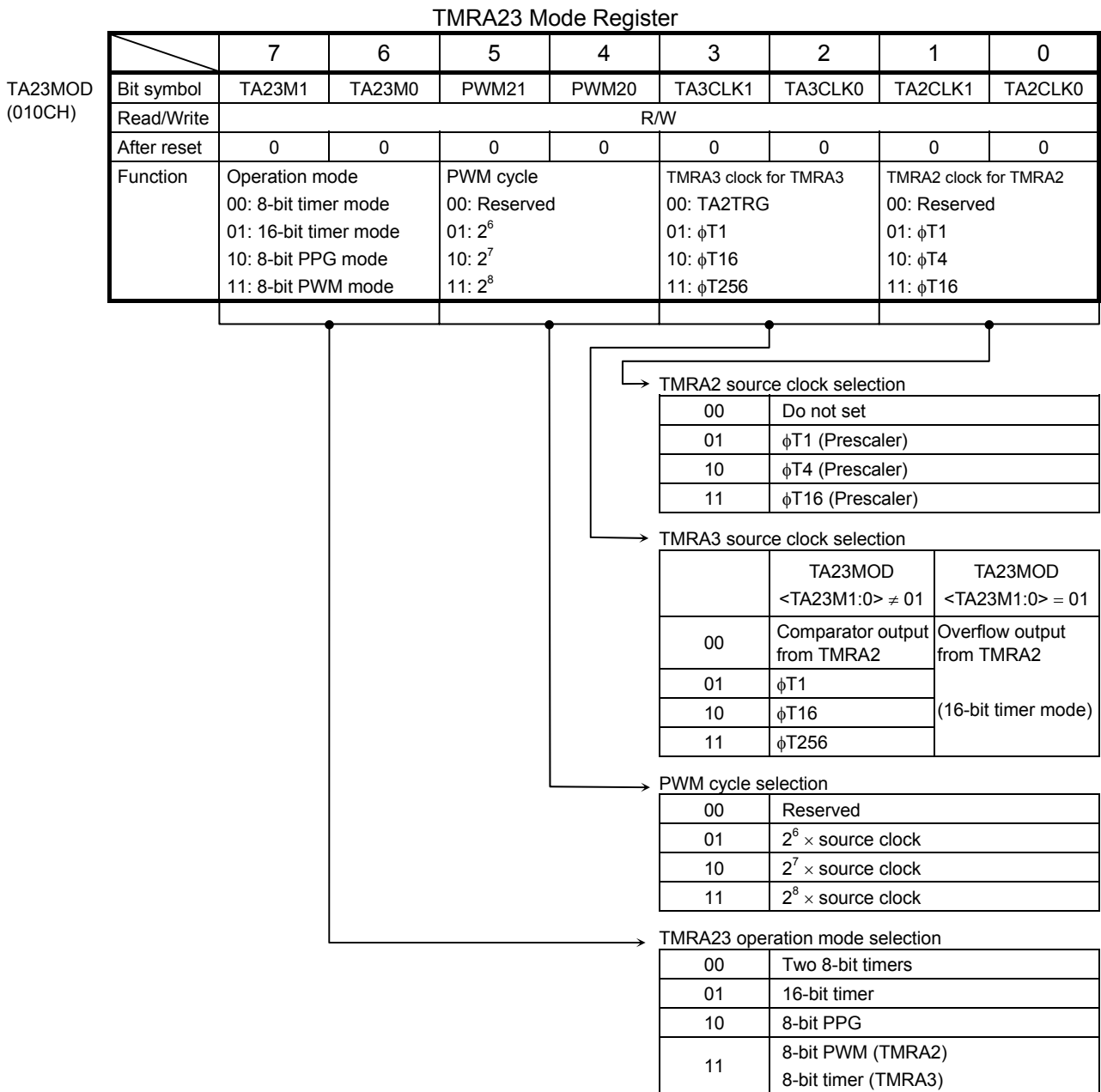


Figure 3.7.9 TMRA Registers

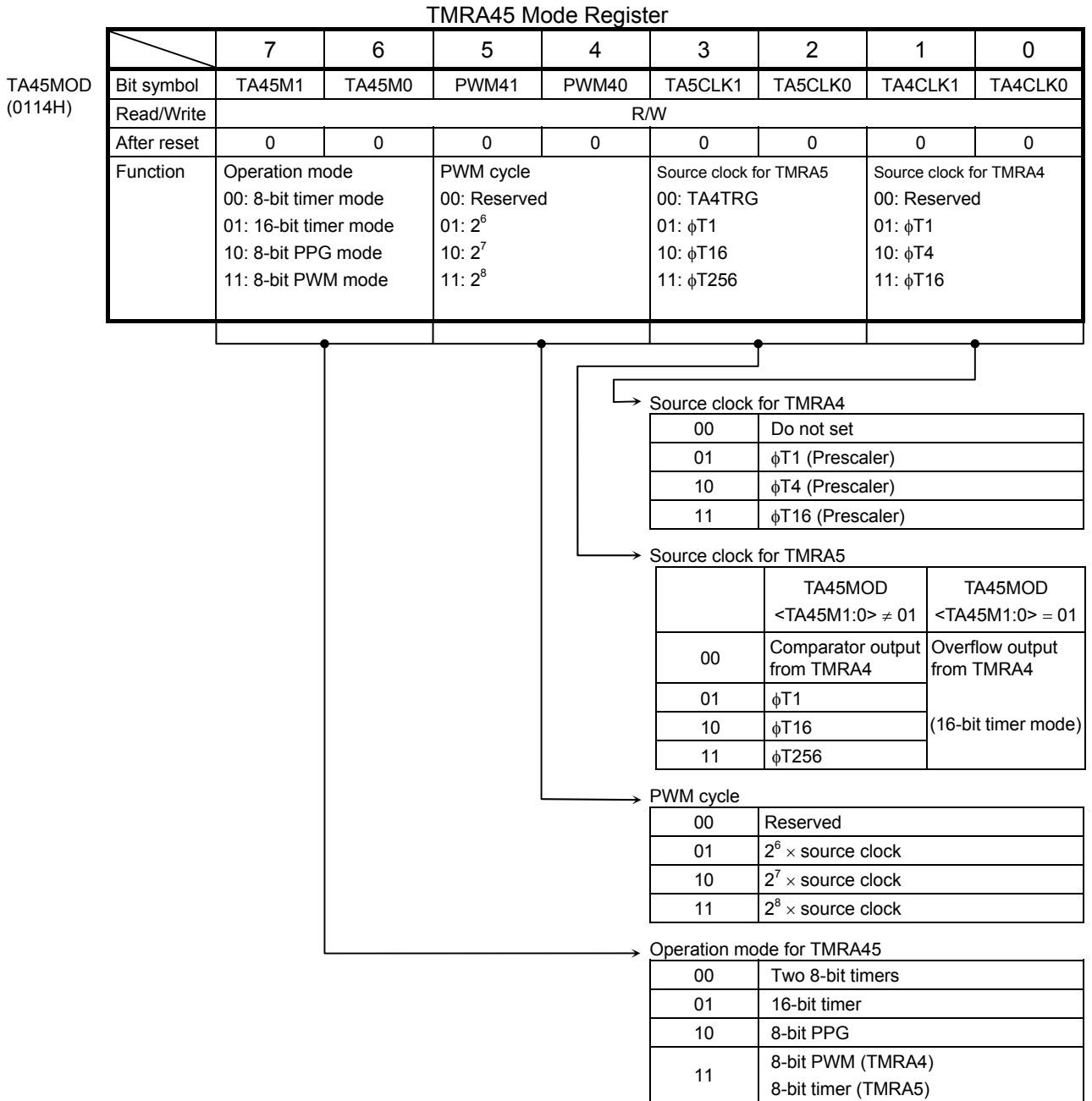


Figure 3.7.10 Registers for TMRA

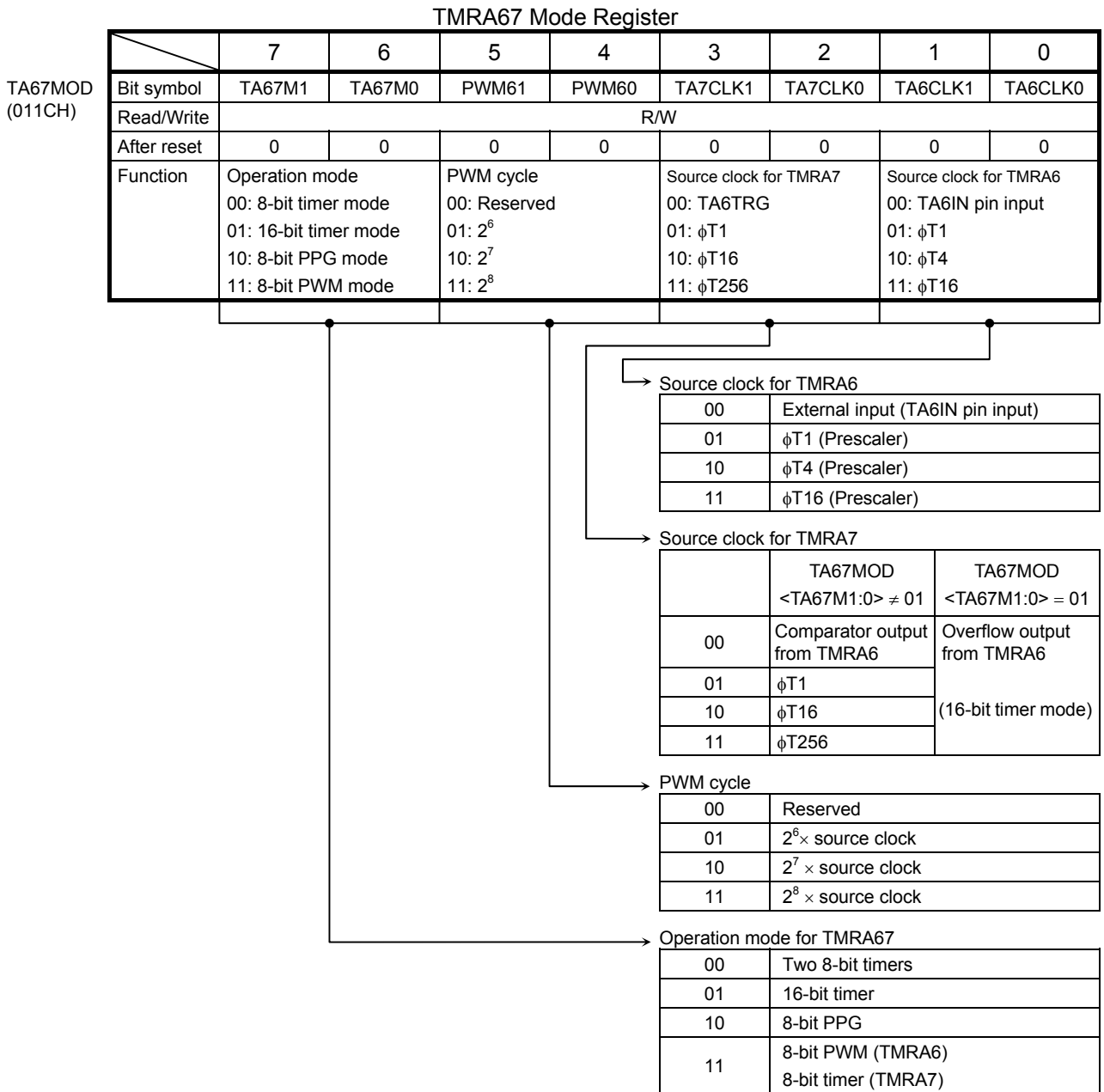
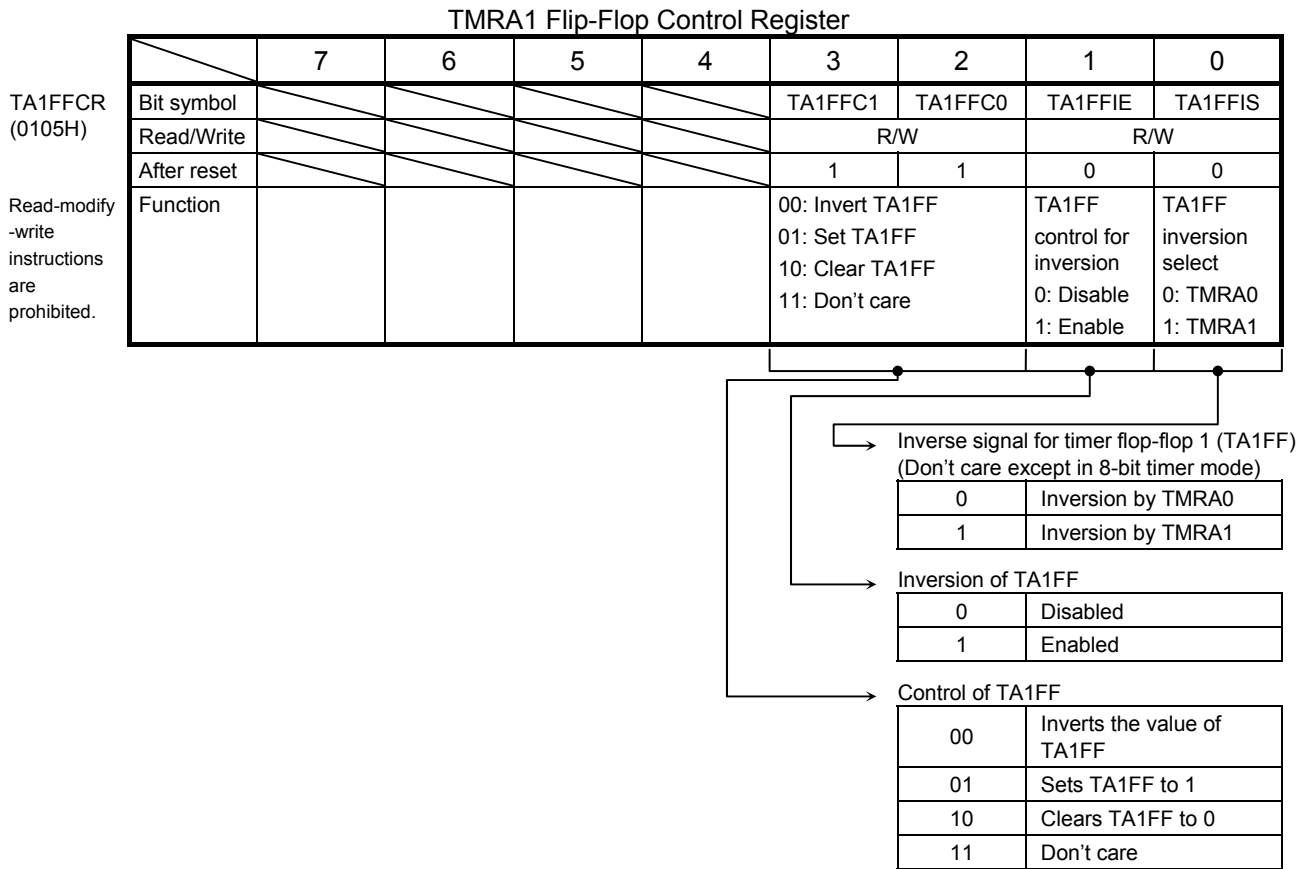
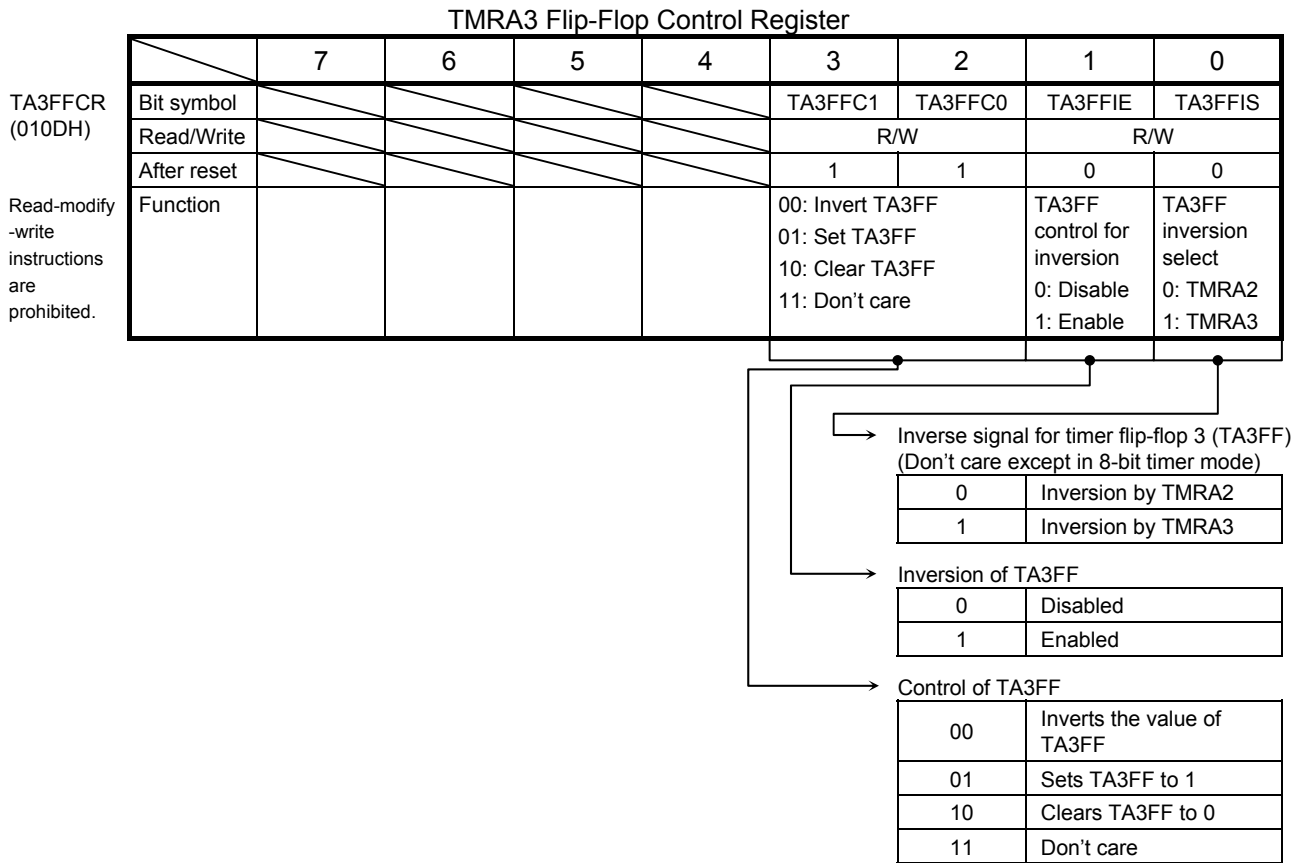


Figure 3.7.11 TMRA Registers



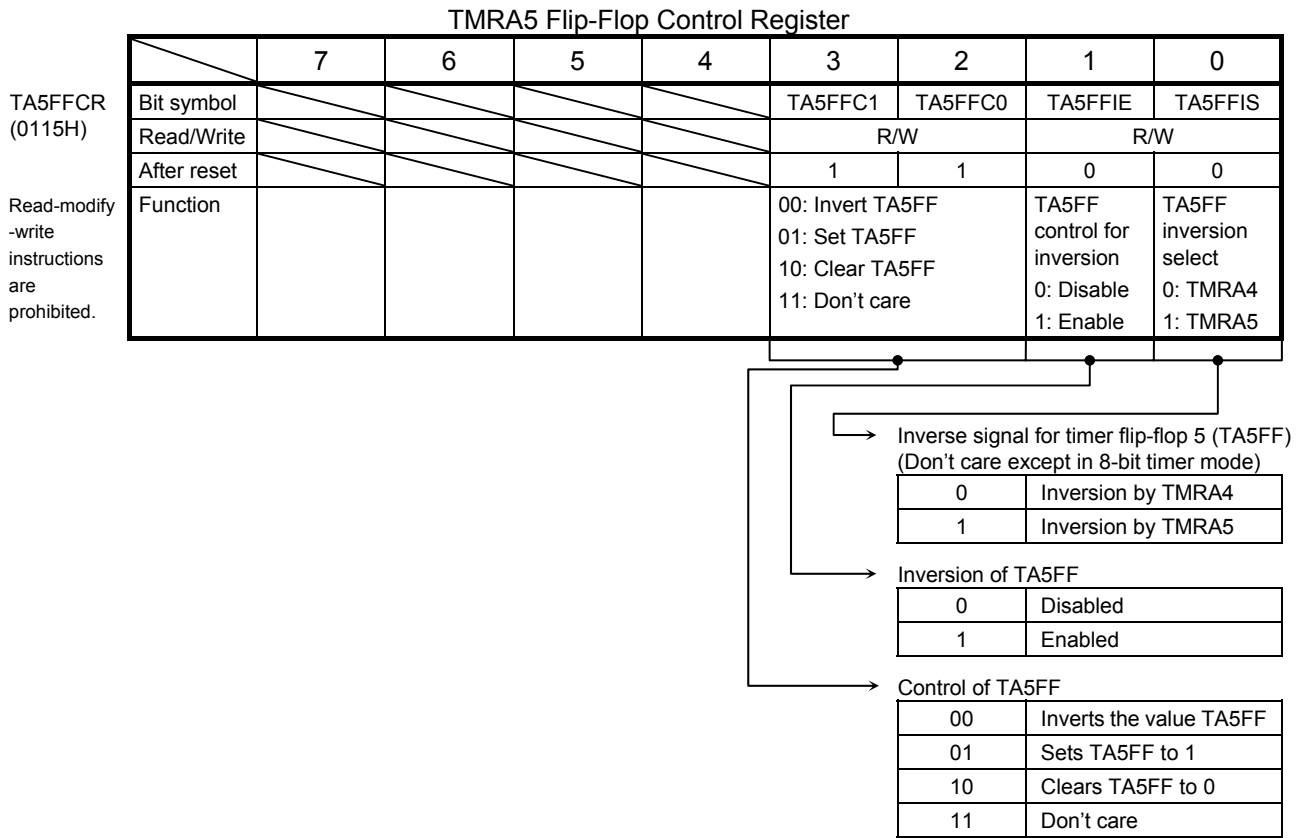
Note: TA1FFCR<TA1FFC1:0> is read "1" when read.

Figure 3.7.12 TMRA Registers



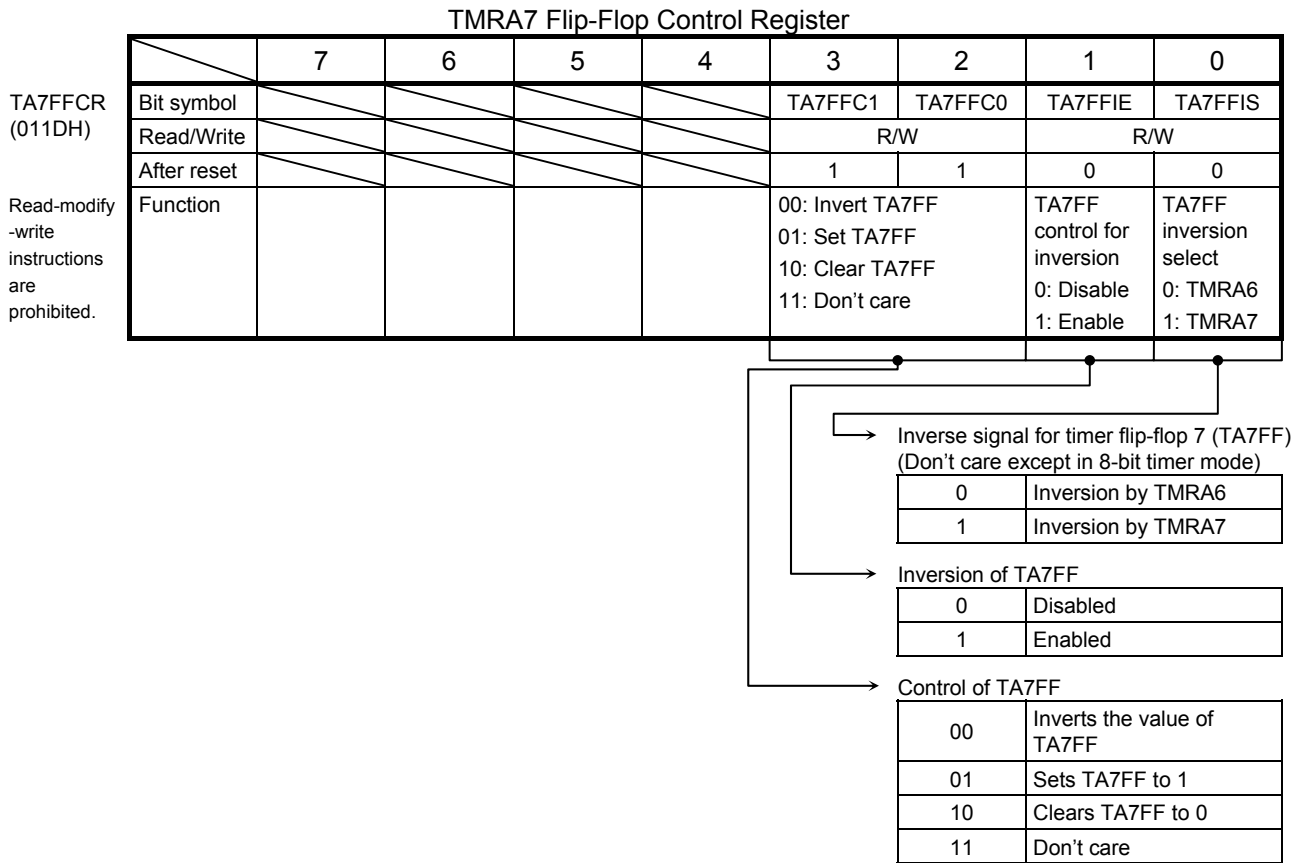
Note: TA3FFCR<TA3FFC1:0> is read "1" when read.

Figure 3.7.13 TMRA Registers



Note: TA5FFCR<TA5FFC1:0> is read "1" when read.

Figure 3.7.14 Registers for TMRA



Note: TA7FFCR<TA7FFC1:0> is read "1" when read.

Figure 3.7.15 TMRA Registers

		Timer register							
		7	6	5	4	3	2	1	0
TA0REG (0102H)	bit Symbol	—							
	Read/Write	W							
	After reset	Undefined							
TA1REG (0103H)	bit Symbol	—							
	Read/Write	W							
	After reset	Undefined							
TA2REG (010AH)	bit Symbol	—							
	Read/Write	W							
	After reset	Undefined							
TA3REG (010BH)	bit Symbol	—							
	Read/Write	W							
	After reset	Undefined							
TA4REG (0112H)	bit Symbol	—							
	Read/Write	W							
	After reset	Undefined							
TA5REG (0113H)	bit Symbol	—							
	Read/Write	W							
	After reset	Undefined							
TA6REG (011AH)	bit Symbol	—							
	Read/Write	W							
	After reset	Undefined							
TA7REG (011BH)	bit Symbol	—							
	Read/Write	W							
	After reset	Undefined							

Note: The above register are prohibited read-modify write instruction.

Figure 3.7.16 TMRA Registers

3.7.4 Operation in Each Mode

(1) 8-bit timer mode

Both TMRA0 and TMRA1 can be used independently as 8-bit interval timer.

1. Generating interrupts at a fixed interval (Using TMRA1)

To generate interrupts at constant intervals using TMRA1 (INTTA1), first stop TMRA1 then set the operation mode, input clock and a cycle to TA01MOD and TA1REG register respectively. Then, enable the interrupt INTTA1 and start TMRA1 counting.

Example: To generate an INTTA1 interrupt every 13 μs at $f_c = 25$ MHz, set each register as follows.

* Clock state $\left\{ \begin{array}{l} \text{System clock: High frequency (} f_c \text{)} \\ \text{Prescaler clock: } f_{PPH} \end{array} \right.$

	MSB	7	6	5	4	3	2	1	0	LSB	
TA01RUN	←	-	-	X	X	-	-	0	-		Stop TMRA1 and clear it to 0.
TA01MOD	←	0	0	X	X	0	1	X	X		Select 8-bit timer mode and select $\phi T1$ (0.3 μs at $f_c = 25$ MHz) as the input clock.
TA1REG	←	0	0	1	0	1	0	0	0		Set TA1REG to $13 \mu s \div \phi T1 = 40 = 28H$.
INTEA01	←	X	1	0	1	-	-	-	-		Enable INTTA1 and set it to level 5.
TA01RUN	←	-	X	X	X	-	1	1	-		Start TMRA1 counting.

X: Don't care, -: No change

Select the input clock using Table 3.7.2

Note: The input clocks for TMRA0 and TMRA1 differ as follows.

TMRA0: Uses can be selected from $\phi T1$, $\phi T4$ or $\phi T16$

TMRA1: Match output of TMRA0 and can be selected from $\phi T1$, $\phi T16$, $\phi T256$

2. Generating a 50 % duty ratio square wave pulse

The state of the timer flip-flop (TA1FF) is inverted at constant intervals and its status output via the timer output pin (TA1OUT).

Example: To output a 1.9 μs square wave pulse from the TA1OUT pin at $f_c = 25$ MHz, use the following procedure to make the appropriate register settings. This example uses TMRA1; however either TMRA0 or TMRA1 may be used.

* Clock state { System clock: High frequency (f_c)
 Clock gear: 1 (f_c)
 Prescaler clock: f_{FPH}

	7	6	5	4	3	2	1	0	
TA01RUN	← -	X	X	X	-	-	0	-	Stop TMRA1 and clear it to 0.
TA01MOD	← 0	0	X	X	0	1	-	-	Select 8-bit timer mode and select $\phi T1$ ($0.3 \mu s$ at $f_c = 25$ MHz) as the input clock.
TA1REG	← 0	0	0	0	0	0	1	1	Set the timer register to $1.9 \mu s \div \phi T1 \div 2 = 3$.
TA1FFCR	← X	X	X	X	1	0	1	1	Clear TA1FF to 0 and set it to invert on the match detect signal from TMRA1.
P7CR	← X	-	-	-	-	-	-	1	} Set P70 to function as the TA1OUT pin.
P7FC	← X	-	-	X	X	-	-	1	
TA01RUN	← -	X	X	X	-	1	1	-	Start TMRA1 counting.

X: Don't care, -: No change

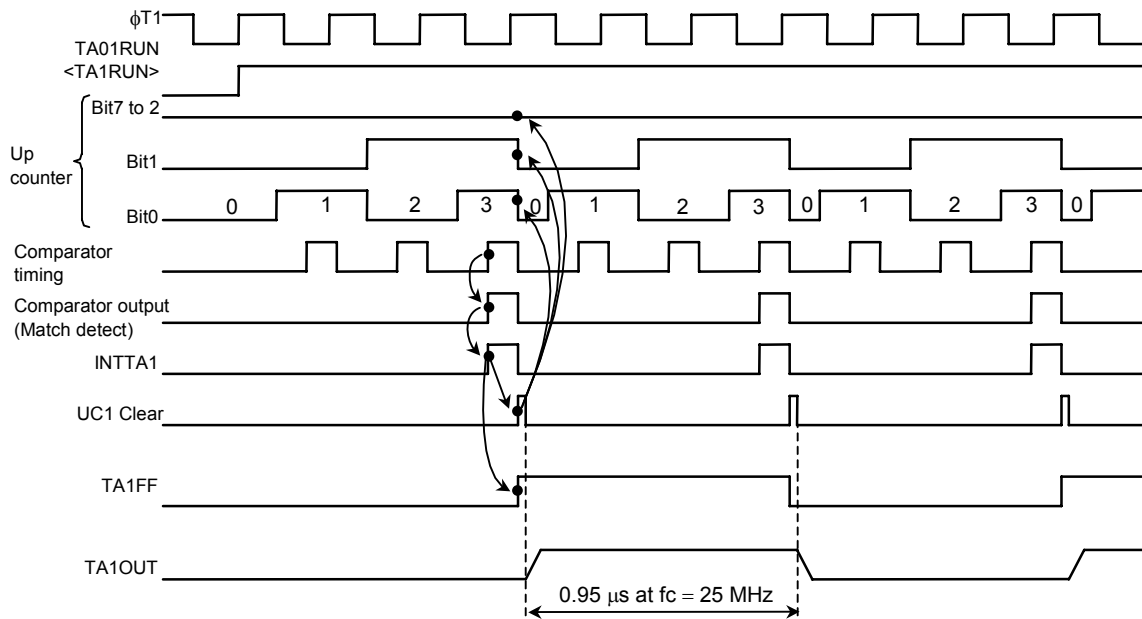


Figure 3.7.17 Square Wave Output Timing Chart (50 % duty)

3. Making TMRA1 count up on the match signal from the TMRA0 comparator

Select 8-bit timer mode and set the comparator output from TMRA0 to be the input clock to TMRA1.

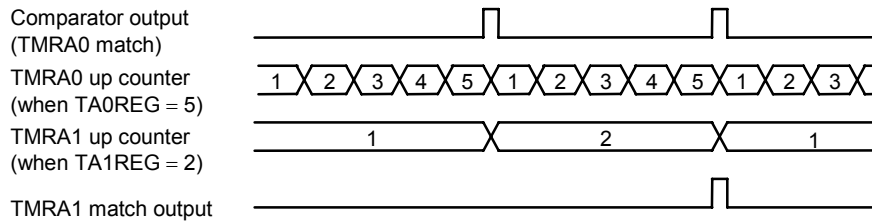


Figure 3.7.18 TMRA1 Count Up on Signal from TMRA0

(2) 16-bit timer mode

A 16-bit interval timer is configured by pairing the two 8-bit timers TMRA0 and TMRA1.

To make a 16-bit interval timer in which TMRA0 and TMRA1 are cascaded together, set $TA01MOD\langle TA01M1:0 \rangle$ to 01.

In 16-bit timer mode, the overflow output from TMRA0 is used as the input clock for TMRA1, regardless of the value set in $TA01MOD\langle TA01CLK1:0 \rangle$. Table 3.7.2 shows the relationship between the timer (Interrupt) cycle and the input clock selection.

Setting example: To generate an INTTA1 interrupt every 0.32 s at $f_c = 25$ MHz, set the timer registers TA0REG and TA1REG as follows.

* Clock state

{	System clock: High frequency (f_c)
	Clock gear: 1 (f_c)
	Prescaler clock: f_{PPH}

If $\phi T16 (= (2^7/f_c) \text{ s @ } 25 \text{ MHz})$ is used as the input clock for counting, set the following value in the registers:

$$0.32 \text{ s} \div (2^7/f_c) \text{ s} = 62500 = \text{F424H}$$

(e.g., set TA1REG to F4H and TA0REG to 24H).

The comparator match signal is output from TMRA0 each time the up counter UC0 matches TA0REG, where the up counter UC0 is not be cleared.

In the case of the TMRA1 comparator, the match detect signal is output on each comparator pulse on which the values in the up counter UC1 and TA1REG match.

When the match detect signal is output simultaneously from both the comparators TMRA0 and TMRA1, the up counter UC0 and UC1 are cleared to 0 and the interrupt INTTA1 is generated. Also, if inversion is enabled, the value of the timer flip-flop 1 TA1FF is inverted.

Example: When TA1REG = 04H and TA0REG = 80H.

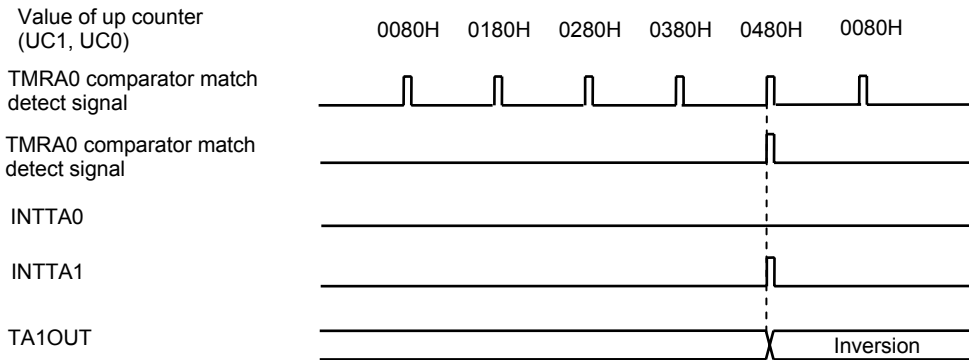


Figure 3.7.19 Timer Output by 16-Bit Timer Mode

(3) 8-bit PPG (Programmable pulse generation) output mode

Square wave pulses can be generated at any frequency and duty ratio by TMRA0. The output pulses may be active low or active high. In this mode TMRA1 cannot be used.

TMRA0 outputs pulses on the TA1OUT pin (which can also be used as P70).

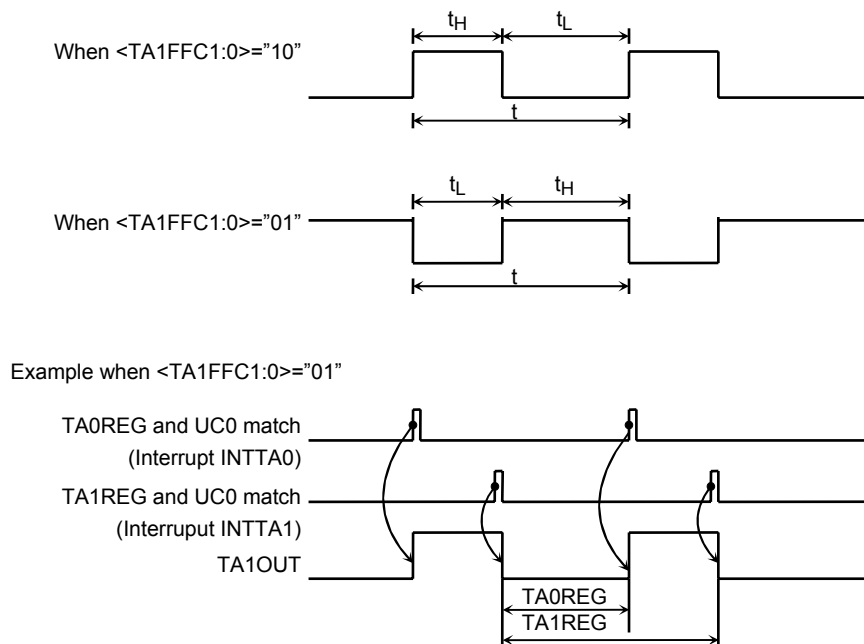


Figure 3.7.20 8-Bit PPG Output Waveforms

In this mode a programmable square wave is generated by inverting the timer output each time the 8-bit up counter (UC0) matches the value in one of the timer registers TA0REG or TA1REG.

The value set in TA0REG must be smaller than the value set in TA1REG.

Although the up counter for TMRA1 (UC1) is not used in this mode, TA01RUN<TA1RUN> should be set to 1 so that UC1 is set for counting.

Figure 3.7.21 shows a block diagram representing this mode.

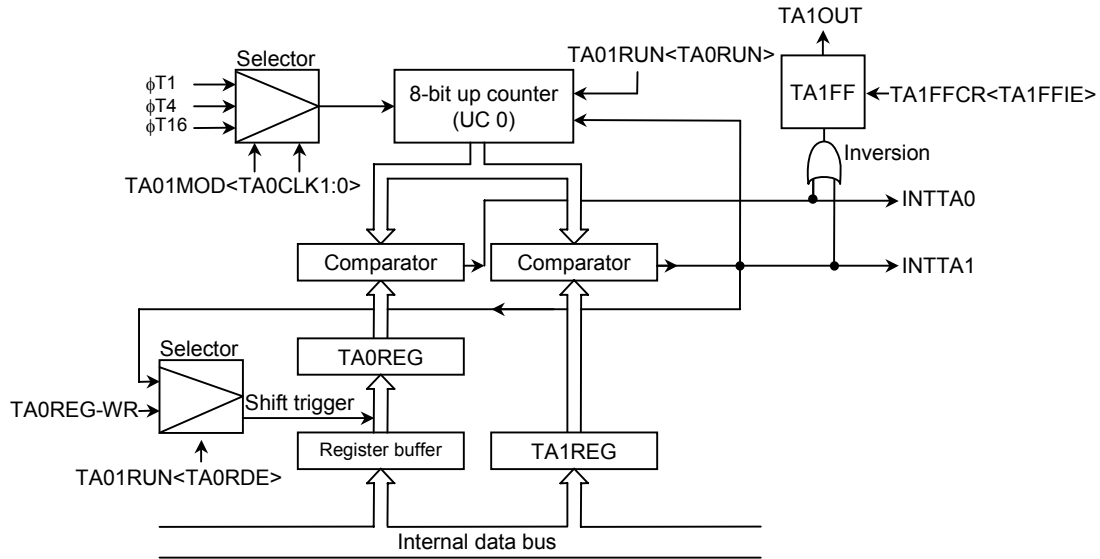


Figure 3.7.21 Block Diagram of 8-Bit PPG Output Mode

If the TA0REG double buffer is enabled in this mode, the value of the register buffer will be shifted into TA0REG each time TA1REG matches UC0.

Use of the double buffer facilitates the handling of low duty waves (when duty is varied).

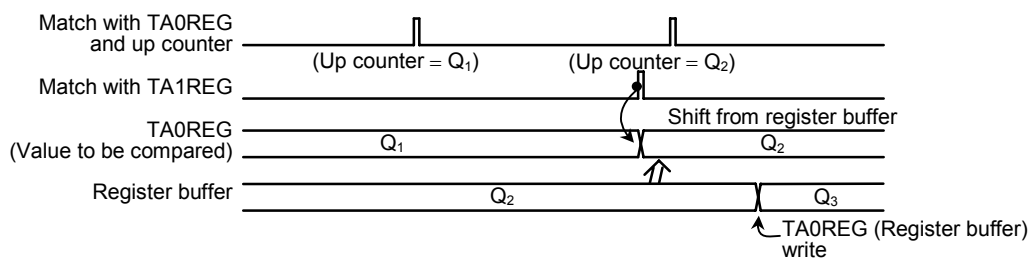
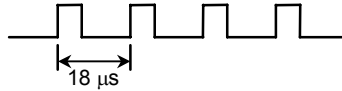


Figure 3.7.22 Operation of Register Buffer

Example: To generate 1/4 duty 55 kHz pulses (at $f_c = 25$ MHz).



* Clock state $\left\{ \begin{array}{l} \text{System clock: High frequency (} f_c \text{)} \\ \text{Clock gear: 1 (} f_c \text{)} \\ \text{Prescaler clock: } f_{FPH} \end{array} \right.$

Calculate the value which should be set in the timer register.

To obtain a frequency of 55 kHz, the pulse cycle t should be: $t = 1/55 \text{ kHz} = 18 \mu\text{s}$

$\phi T1 = (2^3/f_c)s$ (at 25 MHz);

$$18 \mu\text{s} \div (2^3/f_c)s \approx 60$$

Therefore set TA1REG to 60 (3CH)

The duty is to be set to 1/4: $t \times 1/4 = 18 \mu\text{s} \times 1/4 = 4.5 \mu\text{s}$

$$4.5 \mu\text{s} \div (2^3/f_c)s \approx 15$$

Therefore, set TA0REG = 15 = 0FH.

	7	6	5	4	3	2	1	0	
TA01RUN	← 0	X	X	X	-	0	0	0	Stop TMRA0 and TMRA01, and clear it to 0.
TA01MOD	← 1	0	X	X	X	X	0	1	Set the 8-bit PPG mode, and select $\phi T1$ as input clock.
TA0REG	← 0	0	0	0	1	1	1	1	Write 0FH.
TA1REG	← 0	0	1	1	1	1	0	0	Write 3CH.
TA1FFCR	← X	X	X	X	0	1	1	X	Set TA1FF, enabling both inversion and the double buffer.
									10 generates a negative logic pulse.
P7CR	← X	-	-	-	-	-	-	1	} Set P70 as the TA1OUT pin.
P7FC	← X	-	-	X	X	-	-	1	
TA01RUN	← 1	X	X	X	-	1	1	1	Start TMRA0 and TMRA01 counting.

X: Don't care, -: No change

(4) 8-bit PWM (Pulse width modulation) output mode

This mode is only valid for TMRA0. In this mode, a PWM pulse with the maximum resolution of 8 bits can be output.

When TMRA0 is used the PWM pulse is output on the TA1OUT pin (which is also used as P71). TMRA1 can also be used as 8-bit timer.

The timer output is inverted when the up counter (UC0) matches the value set in the timer register TA0REG or when 2^n counter overflow occurs ($n = 6, 7$ or 8 as specified by TA01MOD<PWM01:00>). The up counter UC0 is cleared when 2^n counter overflow occurs.

The following conditions must be satisfied before this PWM mode can be used.

Value set in TA0REG < Value set for 2^n counter overflow

Value set in TA0REG $\neq 0$

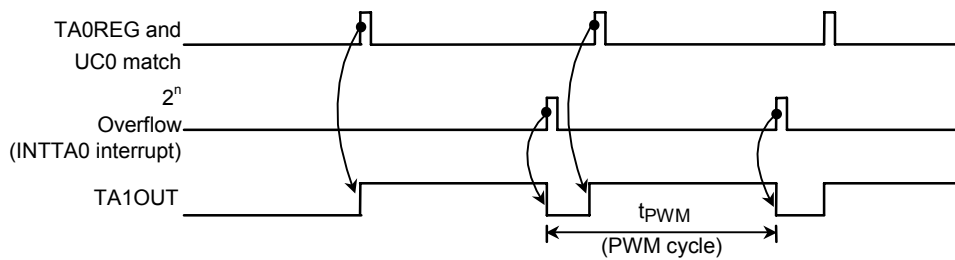


Figure 3.7.23 8-Bit PWM Waveforms

Figure 3.7.24 shows a block diagram representing this mode.

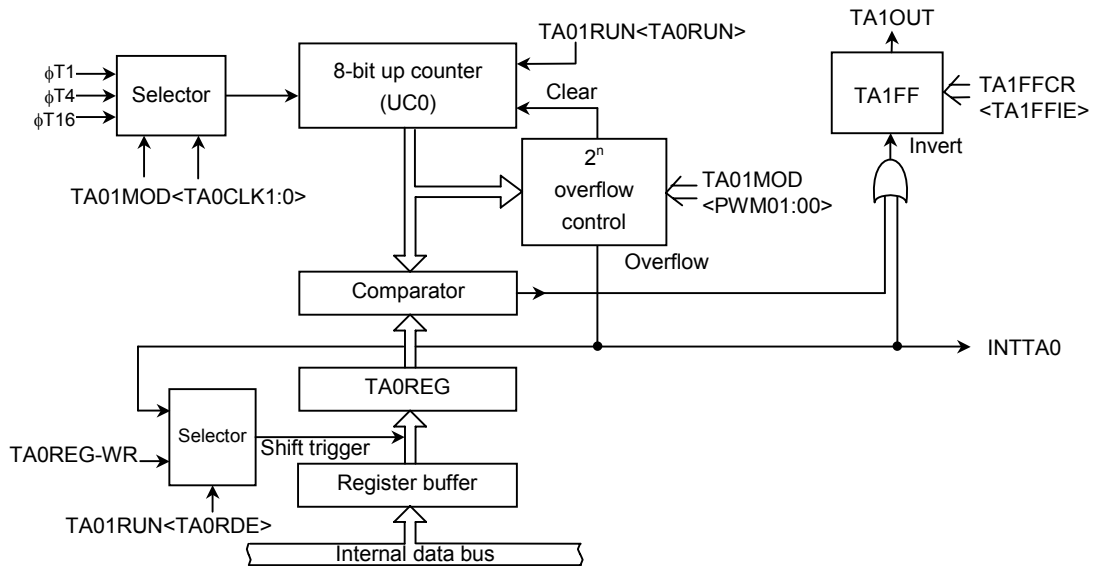


Figure 3.7.24 Block Diagram of 8-Bit PWM Mode

In this mode, the value of the register buffer will be shifted into TA0REG if 2^n overflow is detected when the TA0REG double buffer is enabled.

Use of the double buffer facilitates the handling of low duty ratio waves.

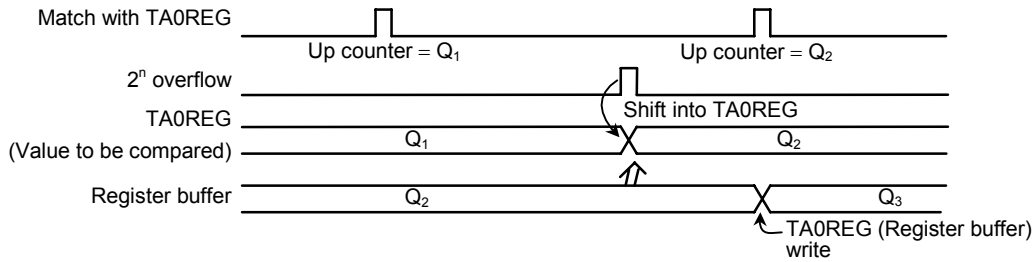
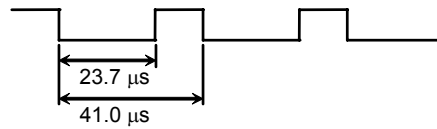


Figure 3.7.25 Register Buffer Operation

Example: To output the following PWM waves on the TA1OUT pin at $f_c = 25$ MHz:



* Clock state $\left\{ \begin{array}{l} \text{System clock: High frequency } (f_c) \\ \text{Clock gear: 1 } (f_c) \\ \text{Prescaler clock: } f_{PPH} \end{array} \right.$

To achieve a $41.0 \mu\text{s}$ PWM cycle by setting $\phi T1 = (2^3/f_c)\text{s}$ (at $f_c = 25$ MHz):

$$41.0 \mu\text{s} \div (2^3/f_c)\text{s} \approx 128$$

$$2^n = 128$$

Therefore n should be set to 7.

Since the low-level period is $23.7 \mu\text{s}$ when $\phi T1 = (2^3/f_c)\text{s}$,

set the following value for TA0REG:

$$23.7 \mu\text{s} \div (2^3/f_c)\text{s} \approx 74 = 4\text{AH}$$

	MSB				LSB					
	7	6	5	4	3	2	1	0		
TA01RUN	←	-	X	X	X	-	-	0	Stop TMRA0 and clear it to 0.	
TA01MOD	←	1	1	1	0	-	-	0	Select 8-bit PWM mode (cycle: 2^7) and select $\phi T1$ as the input clock.	
TA0REG	←	0	1	0	0	1	0	1	Write 4AH.	
TA1FFCR	←	X	X	X	X	1	0	1	X	Clear TA1FF to 0, enable the inversion and double buffer.
P7CR	←	X	-	-	-	-	-	1	} Set P70 and the TA1OUT pin.	
P7FC	←	X	-	-	X	X	-	1		
TA01RUN	←	1	X	X	X	-	1	-		Start TMRA0 counting.

X: Don't care, -: No change

Table 3.7.3 PWM Cycle

at $f_c = 25$ MHz

Select System Clock <SYSCK>	Select Prescaler Clock <PRCK1:0>	Gear Value <GEAR2:0>	PWM Cycle								
			2^6			2^7			2^8		
			$\phi T1$	$\phi T4$	$\phi T16$	$\phi T1$	$\phi T4$	$\phi T16$	$\phi T1$	$\phi T4$	$\phi T16$
0(f_c)	00 (f_{FPH})	000(f_c)	20.5 μ s	82 μ s	328 μ s	41 μ s	164 μ s	655 μ s	82 μ s	328 μ s	1311 μ s
		001($f_c/2$)	41.0 μ s	164 μ s	655 μ s	82 μ s	328 μ s	1311 μ s	164 μ s	655 μ s	2621 μ s
		10($f_c/4$)	81.9 μ s	328 μ s	1311 μ s	164 μ s	655 μ s	2621 μ s	328 μ s	1311 μ s	5243 μ s
		011($f_c/8$)	163.8 μ s	655 μ s	2621 μ s	328 μ s	1311 μ s	5243 μ s	655 μ s	2621 μ s	10486 μ s
		00($f_c/16$)	327.7 μ s	1311 μ s	5243 μ s	655 μ s	2621 μ s	10486 μ s	1311 μ s	5243 μ s	20972 μ s
	10 ($f_c/16$ clock)	XXX	327.7 μ s	1311 μ s	5243 μ s	655 μ s	2621 μ s	10486 μ s	1311 μ s	5243 μ s	20972 μ s

XXX: Don't care

(5) Settings for each mode

Table 3.7.4 shows the SFR settings for each mode.

Table 3.7.4 Timer Mode Setting Registers

Register name	TA01MOD				TA1FFCR
<Bit symbol>	<TA01M1:0>	<PWM01:00>	<TA1CLK1:0>	<TA0CLK1:0>	TA1FFIS
Function	Timer mode	PWM cycle	Upper timer input clock	Lower timer input clock	Timer f/f invert signal select
8-bit timer \times 2 channels	00	–	Lower timer match $\phi T1, \phi T16, \phi T256$ (00, 01, 10, 11)	External clock $\phi T1, \phi T4, \phi T16$ (01, 10, 11)	0: Lower timer output 1: Upper timer output
16-bit timer mode	01	–	–	External clock $\phi T1, \phi T4, \phi T16$ (01, 10, 11)	–
8-bit PPG \times 1 channel	10	–	–	External clock $\phi T1, \phi T4, \phi T16$ (01, 10, 11)	–
8-bit PWM \times 1 channel	11	$2^6, 2^7, 2^8$ (01, 10, 11)	–	External clock $\phi T1, \phi T4, \phi T16$ (01, 10, 11)	–
8-bit PWM \times 1 channel	11	–	$\phi T1, \phi T16, \phi T256$ (01, 10, 11)	–	Output disabled

–: Don't care

3.8 16-Bit Timer/Event Counters (TMRB)

The TMP91CW18A incorporates two multifunctional 16-bit timer/event counters (TMRB0) which have the following operation modes:

- 16-bit interval timer mode
- 16-bit event counter mode
- 16-bit programmable pulse generation (PPG) mode

Can be used following operation modes by capture function:

- Frequency measurement mode
- Pulse width measurement mode
- Time differential measurement mode

Each timer/event counter channel consists of a 16-bit up counter, two 16-bit timer registers (One of them with a double-buffer structure), two 16-bit capture registers, two comparators, a capture input controller, a timer flip-flop and a control circuit.

Each timer/event counter is controlled by an 11-byte control SFR.

3.8.1 Block Diagram

3.8.2 Operation of Each Block

3.8.3 SFRs

3.8.4 Operation in Each Mode

- (1) 16-bit interval timer mode
- (2) 16-bit event counter mode
- (3) 16-bit programmable pulse generation (PPG) output mode
- (4) Capture function examples
 1. One-shot pulse output from external trigger pulse
 2. Frequency measurement
 3. Pulse width measurement
 4. Time differential measurement

Table 3.8.1 Differences between TMRB0 and TMRB1

Spec		Channel	TMRB0	
External pins	External clock/capture trigger input pins		TB0IN0 (Also used as P73) TB0IN1 (Also used as P74)	
	Timer flip-flop output pins		TB0OUT0 (Also used as P75) TB0OUT1 (Also used as P83)	
SFR (Address)	Timer run register		TB0RUN (0180H)	
	Timer mode register		TB0MOD (0182H)	
	Timer flip-flop control register		TB0FFCR (0183H)	
	Timer register			TB0RG0L (0188H) TB0RG0H (0189H) TB0RG1L (018AH) TB0RG1H (018BH)
		Capture register		

3.8.1 Block Diagram

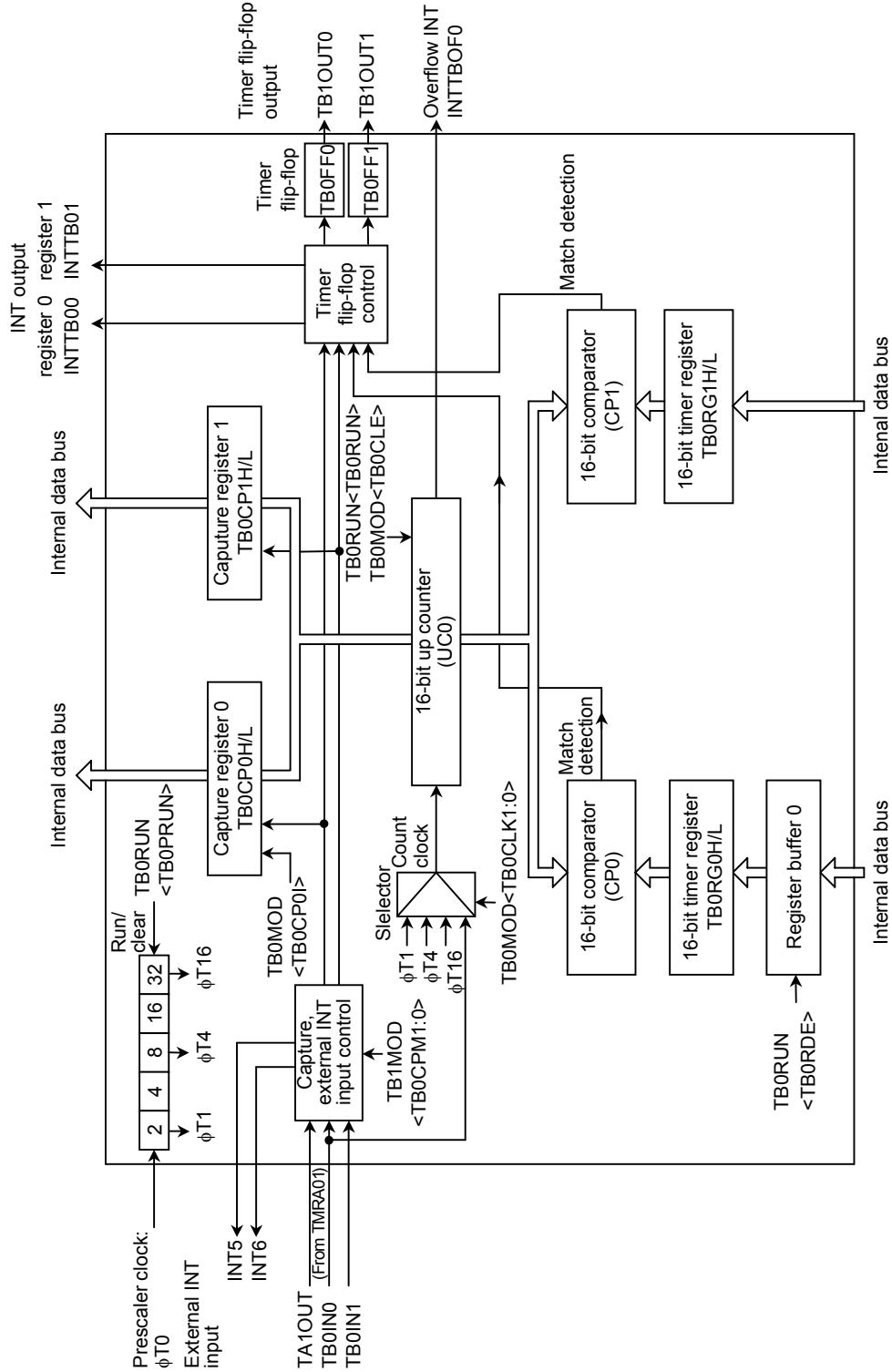


Figure 3.8.1 Block Diagram of TMRB0

3.8.2 Operation of Each Block

(1) Prescaler

The 5-bit prescaler generates the source clock for TMRB0. The prescaler clock ($\phi T0$) is divided clock (Divided by 4) from selected clock by the register SYSCR0<PRCK1:0> of clock gear.

This prescaler can be started or stopped using TB0RUN<TB0PRUN>. Counting starts when <TB0RUN> is set to 1; the prescaler is cleared to 0 and stops operation when <TB0RUN> is cleared to 0.

Table 3.8.2. Prescaler Clock Resolution

at $f_c = 25 \text{ MHz}$

System Clock Selection <SYSCK>	Prescaler Clock Selection <PRCK1:0>	Clock Gear Value <GEAR2:0>	Prescaler Clock Resolution		
			$\phi T1$	$\phi T4$	$\phi T16$
0 (f_c)	00 (f_{FPH})	000 (f_c)	$2^3/f_c$ (0.3 μs)	2^5 (1.3 μs)	$2^7/f_c$ (5.1 μs)
		001 ($f_c/2$)	$2^4/f_c$ (0.6 μs)	2^6 (2.6 μs)	$2^8/f_c$ (10.2 μs)
		010 ($f_c/4$)	$2^5/f_c$ (1.3 μs)	$2^7/f_c$ (5.1 μs)	$2^9/f_c$ (20.5 μs)
		011 ($f_c/8$)	$2^6/f_c$ (2.6 μs)	$2^8/f_c$ (10.2 μs)	$2^{10}/f_c$ (41.0 μs)
		100 ($f_c/16$)	$2^7/f_c$ (5.1 μs)	$2^9/f_c$ (20.5 μs)	$2^{11}/f_c$ (81.9 μs)
	10 (Note: $f_c/16$ clock)	XXX	$2^7/f_c$ (5.1 μs)	$2^9/f_c$ (20.5 μs)	$2^{11}/f_c$ (81.9 μs)

xxx: Don't care

(2) Up counter (UC0)

UC0 is a 16-bit binary counter which counts up pulses input from the clock specified by TB0MOD<TB0CLK1:0>.

Any one of the prescaler internal clocks $\phi T1$, $\phi TB0$, $\phi T16$ or an clock input via the TB0IN0 pin can be selected as the input clock. Counting or stopping and clearing of the counter is controlled by TB0RUN<TB0RUN>.

When clearing is enabled, the up counter UC0 will be cleared to 0 each time its value matches the value in the timer register TBORG1H/L. Clearing can be enabled or disabled using TB0MOD<TB0CLE>.

If clearing is disabled, the counter operates as a free-running counter.

A timer overflow interrupt (INTTBOF0) is generated when UC0 overflow occurs.

(3) Timer registers (TBORG0H/L and TBORG1H/L)

These two 16-bit registers are used to set the interval time. When the value in the up counter UC0 matches the value set in this timer register, the comparator match detect signal will go active.

Setting data for both upper and lower timer registers are always needed. For example, either by using 2-byte data transfer instruction or using 1-byte data transfer instruction twice for lower 8 bits and upper 8 bits in order.

The TBORG0 timer register has a double-buffer structure, which is paired with register buffer. The value set in TBORUN<TBORDE> determines whether the double-buffer structure is enabled or disabled: it is disabled when <TBORDE> = 0, and enabled when <TBORDE> = 1.

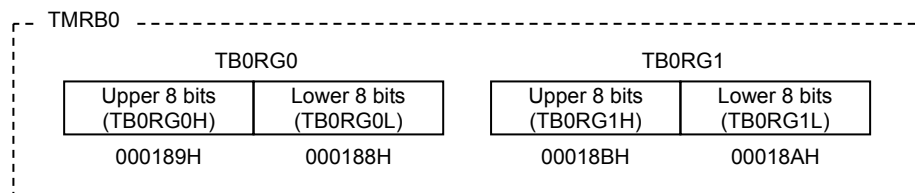
When the double buffer is enabled, data is transferred from the register buffer to the timer register when the values in the up counter (UC0) and the timer register TBORG1 match.

After a reset, TBORG0 and TBORG1 are undefined. If the 16-bit timer is to be used after a reset, data should be written to it beforehand.

On a reset <TBORDE> is initialized to 0, disabling the double buffer. To use the double buffer, write data to the timer register, set <TBORDE> to 1, then write data to the register buffer as shown below.

TBORG0 and the register buffer both have the same memory addresses (000188H and 000189H) allocated to them. If <TBORDE> = 0, the value is written to both the timer register and the register buffer. If <TBORDE> = 1, the value is written to the register buffer only.

The addresses of the timer registers are as follows.



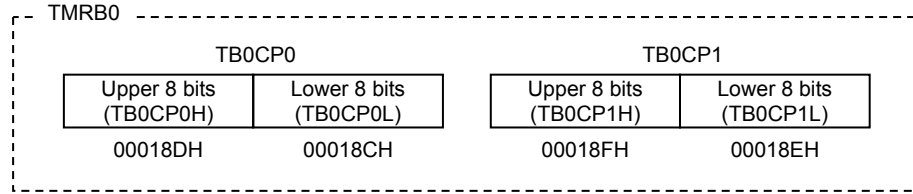
The timer registers are write only registers and thus cannot be read.

(4) Capture registers (TB0CP0H/L and TB0CP1H/L)

These 16-bit register are used to latch the values in the up counter.

Data in the capture registers should be read all 16 bits. For example, using a 2-byte data load instruction or two 1-byte data load instructions. The least significant byte is read first, followed by the most significant byte.

The addresses of the capture registers are as follows.



The capture registers are read only registers and thus cannot be written to.

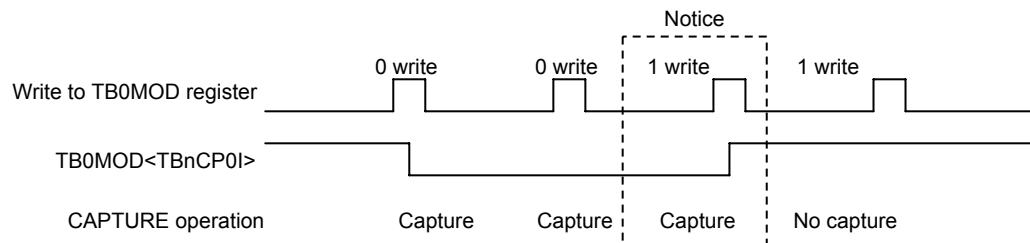
(5) Capture input control and external interrupt control (INT5 and INT6)

This circuit controls the timing to latch the value of up counter UC0 into TB0CP0, TB0CP1 and the generation of external interrupts. The latch timing for the capture register and selection of edge for external interrupt is determined by TB0MOD<TB0CPM1:0>.

The edge of external interrupt INT6 is fixed to rise edge.

In addition, the value in the up counter can be loaded into a capture register by software. Whenever 0 is written to TB0MOD<TB0CP0I>, the current value in the up counter is loaded into capture register TB0CP0. It is necessary to keep the prescaler in run mode (e.g., TB0RUN<TB0PRUN> must be held at a value of 1).

Note: As described above, whenever 0 is written to TB0MOD<TB0CP0I>, the current value in the up counter is loaded into capture register TB0CP0. However, note that the current value in the up counter is also loaded into capture register TB0CP0 when 1 is written to TB0MOD<TB0CP0I> while this bit is holding 0.



(6) Comparators (CP0 and CP1)

CP0 and CP1 are 16-bit comparator which compare the value in the up counter UC0 with the value set in TB0RG0 or TB0RG1 respectively, in order to detect a match. If a match is detected, the comparator generates an interrupt (INTTB00 or INTTB01 respectively).

(7) Timer flip-flops (TB0FF0 and TB0FF1)

These flip-flops are inverted by the match detect signals from the comparators and the latch signals to the capture registers. Inversion can be enabled and disabled for each element using TB0FFCR<TB0C1T1, TB0C0T1, TB0E1T1, TB0E0T1>. After a reset the value of TB0FF0 is undefined. If 00 is written to TB0FFCR<TB0FF0C1:0> or <TB0FF1C1:0>, TB0FF0 will be inverted. If 01 is written to the capture registers, the value of TB0FF0 will be set to 1. If 10 is written to the capture registers, the value of TB0FF0 will be cleared to 0. The values of TB0FF0 and TB0FF1 can be output via the timer output pins TB0OUT0 (which is shared with P75). Timer output should be specified using the port 7 function registers.

3.8.3 SFRs

TMRB0 Run Register

		7	6	5	4	3	2	1	0
TB0RUN (0180H)	Bit symbol	TB0RDE	-			I2TB0	TB0PRUN		TB0RUN
	Read/Write	R/W	R/W			R/W	R/W		R/W
	After reset	0	0			0	0		0
	Function	Double buffer 0: Disable 1: Enable	Always write 0			IDLE2 0: Stop 1: Operate	16-bit timer run/stop control 0: Stop and clear 1: Run (Count up)		

Count operation

0	Stop and clear
1	Count up

I2TB0: Operation during IDLE2 mode
 TB0PRUN: Operation of prescaler
 TB0RUN: Operation of TMRB0

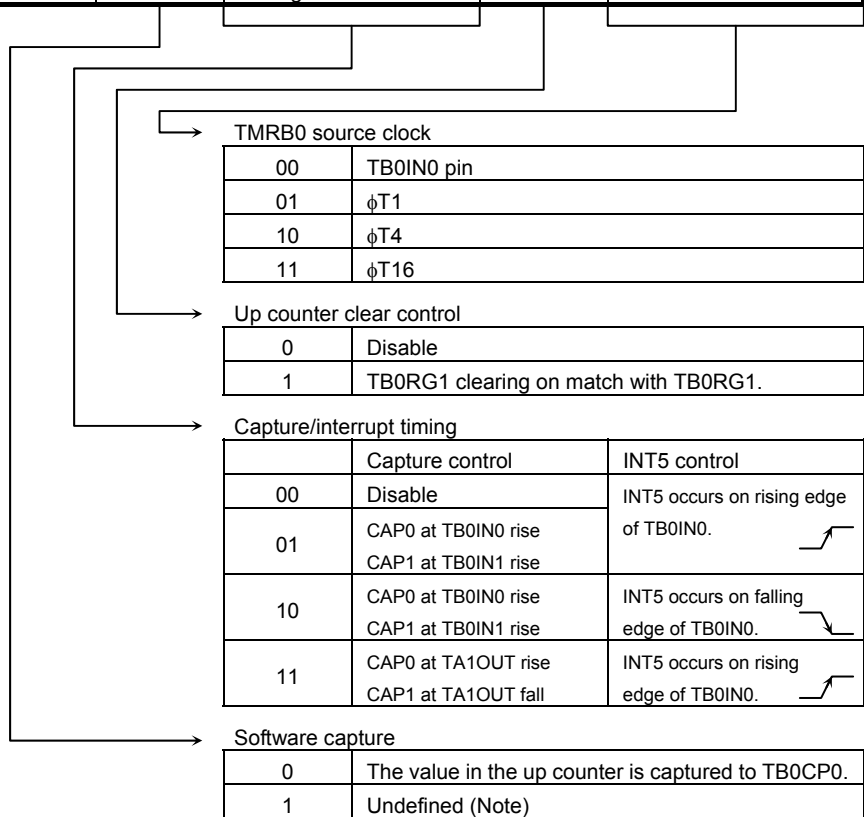
Note: The 1, 4 and 5 of TB0RUN are read as undefined value.

Figure 3.8.1 The Registers for TMRB

TMRB0 Mode Register

	7	6	5	4	3	2	1	0
Bit symbol	TB0CT1	TB0ET1	TB0CP0I	TB0CPM1	TB0CPM0	TB0CLE	TB0CLK1	TB0CLK0
Read/Write	R/W		W*	R/W				
After reset	0	0	1	0	0	0	0	0
Function	TBOFF1 inversion 0: Disable trigger 1: Enable trigger Invert when the UC value is captured to TB0CP1.		Execute software capture 0: Execute 1: Undefined	Capture timing 00: Disable INT5 occurs on rising edge. 01: TB0IN0 ↑ TB0IN1 ↑ INT5 occurs on rising edge. 10: TB0IN0 ↑ TB0IN1 ↓ INT5 occurs on falling edge. 11: TA1OUT ↑ TA1OUT ↓ INT5 occurs on rising edge.		Control up counter 0: Disable clearing 1: Enable clearing		TMRB0 input clock 00: TB0IN0 pin 01: φT1 10: φT4 11: φT16

Read-modify-write instructions are prohibited.



Note: Whenever writing 0 to TB0MOD<TB0CP0I> bit, present value of up counter is received to capture register TB0CP0.
 But, write 1 to TB0MOD<TB0CP0I> in condition of written 0 to TB0MOD<TB0CP0I> bit, present value of up counter is received to capture register TB0CP0.
 Therefore you must to regard.

Figure 3.8.2 The Registers for TMRB

TMRB0 Flip-Flop Control Register

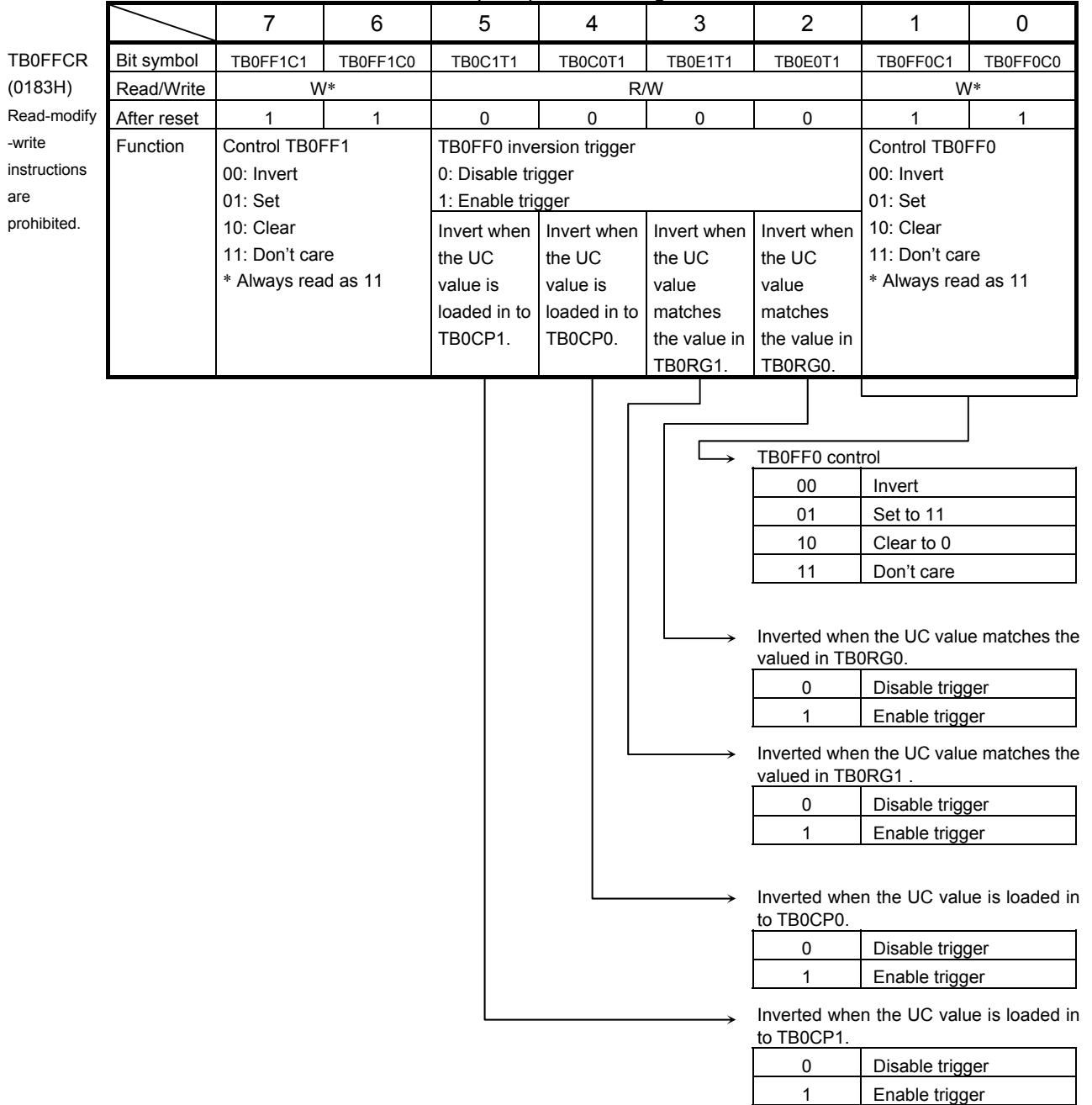


Figure 3.8.3 The Registers for TMRB

		7	6	5	4	3	2	1	0
TB0RG0L (0188H)	bit Symbol	—							
	Read/Write	W							
	After reset	Undefined							
TB0RG0H (0189H)	bit Symbol	—							
	Read/Write	W							
	After reset	Undefined							
TB0RG1L (018AH)	bit Symbol	—							
	Read/Write	W							
	After reset	Undefined							
TB0RG1H (018BH)	bit Symbol	—							
	Read/Write	W							
	After reset	Undefined							

Note: The above registers are prohibited read-modify-write instruction.

Figure 3.8.4 The Registers for TMRB

3.8.4 Operation in Each Mode

(1) 16-bit interval timer mode

Generating interrupts at fixed intervals.

In this example, the interrupt INTTB01 is set to be generated at fixed intervals. The interval time is set in the timer register TBORG1.

	7	6	5	4	3	2	1	0		
TBORUN	←	0	0	X	X	-	0	X	0	Stop TMRB0.
INTEB0	←	X	1	0	0	X	0	0	0	Enable INTTB01 and set interrupt level 4. Disable INTTB00.
TB0FFCR	←	1	1	0	0	0	0	1	1	Disable the trigger.
TB0MOD	←	0	0	1	0	0	1	*	*	Select internal clock for input and disable the capture function.
					(** = 01, 10, 11)					
TBORG1	←	*	*	*	*	*	*	*	*	Set the interval time (16 bits).
TBORUN	←	0	0	X	X	-	1	X	1	Start TMRB0.

X: Don't care, -: No change

(2) 16-bit event counter mode

As described above, in 16-bit timer mode, if the external clock (TB0IN0 pin input) is selected as the input clock, the timer can be used as an event counter. To read the value of the counter, first perform software capture once, then read the captured value.

	7	6	5	4	3	2	1	0		
TBORUN	←	0	0	X	X	-	0	X	0	Stop TMRB0.
P7CR	←	-	-	-	-	0	-	-	-	Set P73 to input mode.
INTEB0	←	X	1	0	0	X	0	0	0	Enable INTTB01 and set interrupt level 4. Disable INTTB00.
TB0FFCR	←	1	1	0	0	0	0	1	1	Disable the trigger.
TB0MOD	←	0	0	1	0	0	1	0	0	Select TB0IN0 as the input clock.
TBORG1	←	*	*	*	*	*	*	*	*	Set the number of counts. (16 bits)
TBORUN	←	0	0	X	X	-	1	X	1	Start TMRB0.

X: Don't care, -: No change

When the timer is used as an event counter, set the prescaler in run mode (e.g., with TBORUN<TB0PRUN> = 1).

(3) 16-bit programmable pulse generation (PPG) output mode

Square wave pulses can be generated at any frequency and duty ratio. The output pulse may be either low-active or high-active.

The PPG mode is obtained by inversion of the timer flip-flop TB0FF0 that is to be enabled by the match of the up counter UC0 with timer register TB0RG0 or TB0RG1 and to be output to TB0OUT0. In this mode the following conditions must be satisfied.

$$(\text{Value set in TB0RG0}) < (\text{Value set in TB0RG1})$$

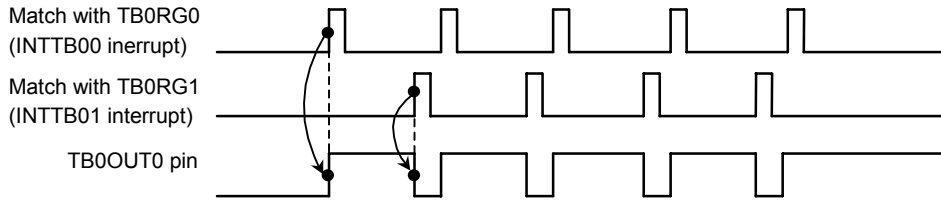


Figure 3.8.5 Programmable Pulse Generation (PPG) Output Waveforms

When the TB0RG0 double buffer is enabled in this mode, the value of register buffer 0 will be shifted into TB0RG0 at match with TB0RG1. This feature facilitates the handling of low-duty waves.

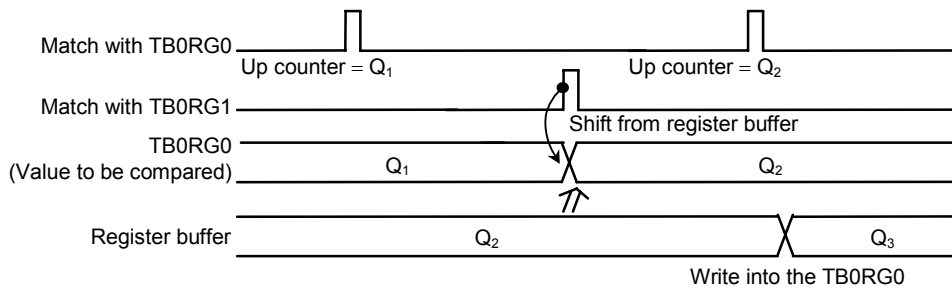


Figure 3.8.6 Operation of Register Buffer

The following block diagram illustrates this mode.

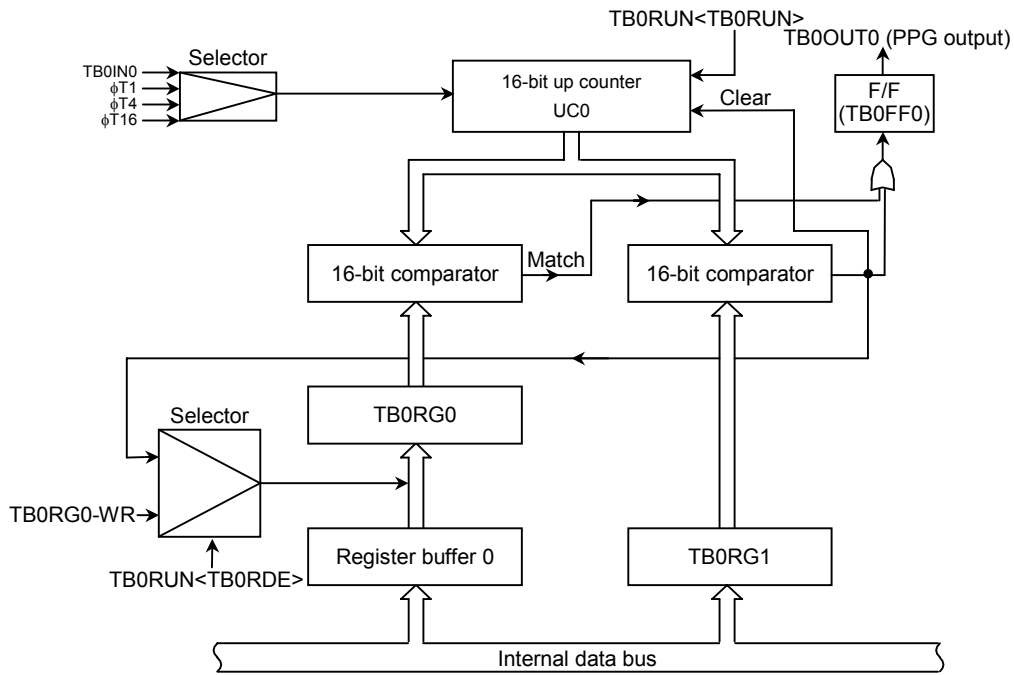


Figure 3.8.7 Block Diagram of 16-Bit Mode

The following example shows how to set 16-bit PPG output mode.

	7	6	5	4	3	2	1	0	
TB0RUN	← 0	0	X	X	-	0	X	0	Disable the TBORG0 double buffer and stop TMRB0.
TBORG0	← *	*	*	*	*	*	*	*	Set the duty ratio. (16 bits)
TBORG1	← *	*	*	*	*	*	*	*	Set the frequency. (16 bits)
TB0RUN	← 1	0	X	X	-	0	X	0	Enable the TBORG0 double buffer. (The duty and frequency are changed on an INTTB01 interrupt.)
TB0FFCR	← X	X	0	0	1	1	1	0	Set the mode to invert TB0FF0 at the match with TBORG0/TBORG1. Clear TB0FF0 to 0.
TB0MOD	← 0	0	1	0	0	1	*	*	Select the internal clock as the input clock and disable the capture function.
					(** = 01, 10, 11)				
P7CR	← X	-	1	-	-	-	-	-	} Set P75 to function as TB0OUT0.
P7FC	← X	-	1	X	X	-	-	-	
TB0RUN	← 1	0	X	X	-	1	X	1	Start TMRB0.

X: Don't care, -: No change

(4) Capture function examples

The capture function can be used in many ways. The following are examples:

1. As a one-shot pulse output from external trigger pulse
2. For frequency measurement
3. For pulse width measurement
4. For time difference measurement

1. One-shot pulse output from external trigger pulse

Set the up counter UC0 to free-running mode with the internal input clock, input an external trigger pulse via the TB0IN0 pin, and load the value of the up counter into the capture register TB0CP0 on the rising edge of the TB0IN0 input signal.

When the interrupt INT5 is generated on the rising edge of the TB0IN0 input, set the TB0CP0 value (c) plus a delay time (d) in TB0RG0 and set this value (c + d) plus the one-shot pulse width (p) in TB0RG1. (Thus $TB0RG0 = c + d$ and $TB0RG1 = c + d + p$.) When the interrupt INT5 occurs, $TB0FFCR < TB0E1T1, TB0E0T1 >$ should be set to 11 and that the TB0FF0 inversion is enabled only when the up counter value matches TB0RG0 or TB0RG1. When an INTTB01 interrupt occurs, a one-shot pulse will be output and inversion will be disabled.

(c), (d) and (p) correspond to c, d and p in Figure 3.8.8.

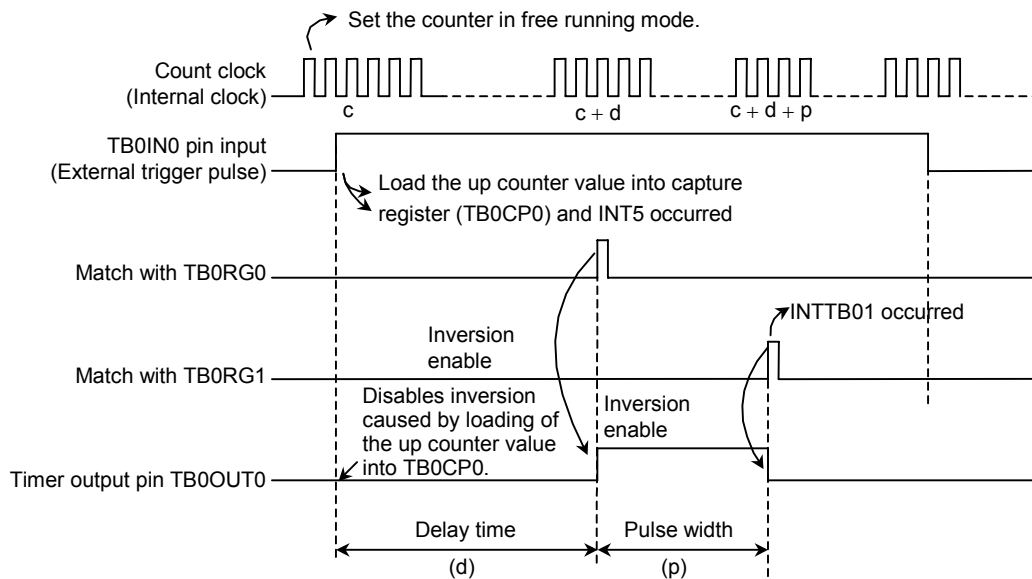


Figure 3.8.8 One-shot Pulse Output (with delay)

2. Frequency measurement

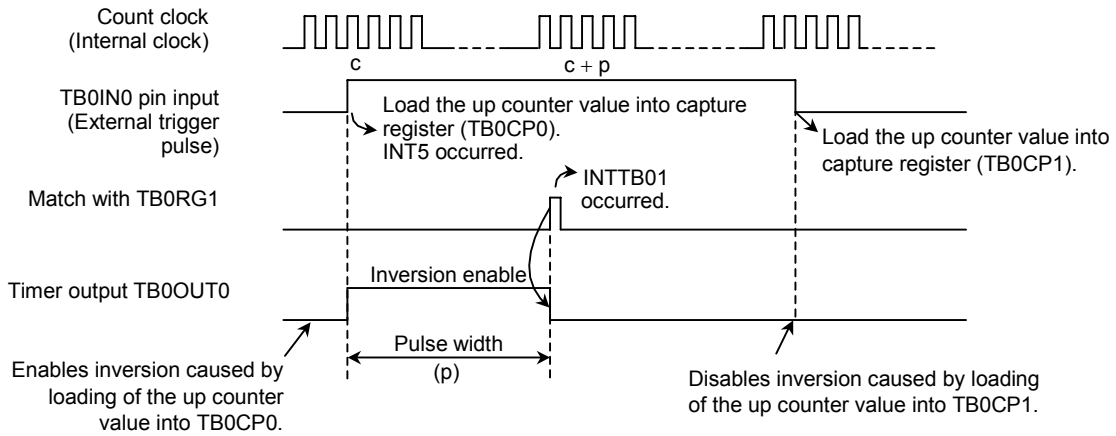


Figure 3.8.9 One-shot Pulse Output (without delay)

The frequency of the external clock can be measured in this mode. The clock is input via the TB0IN0 pin and its frequency is measured using the two 8-bit timers of TMRA01 and the 16-bit timer/event counter TMRB0.

The TB0IN0 pin input should be selected as the clock input to TMRB0. Set TB0MOD<TB0CPM1:0> to 11. The value of the up counter is loaded into the capture register TB0CP0 on the rising edge of the TA1FF signal from the timer flip-flop for the two 8-bit timers (TMRA01), and loaded into TB0CP1 on the falling edge of the TA1FF signal.

The frequency is calculated using the difference between the values loaded into TB0CP0 and TB0CP1 when the interrupt (INTTA0 or INTTA1) is generated by either one of the 8-bit timers.

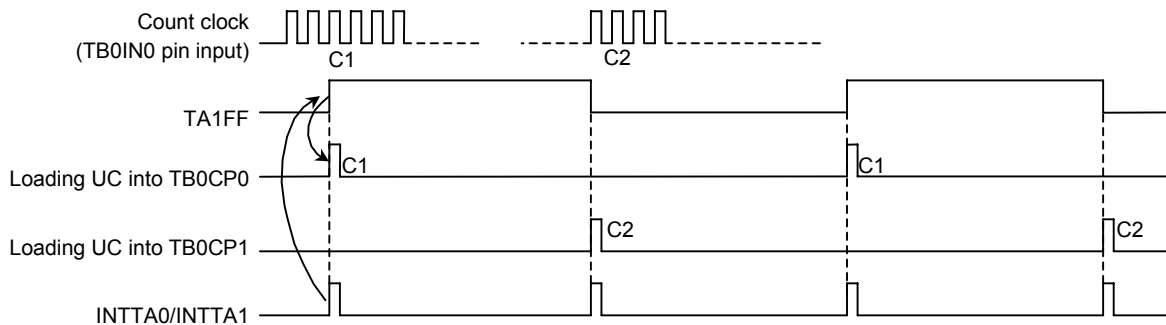


Figure 3.8.10 Frequency Measurement

For example, if the value for the level 1 width of TA1FF of the 8-bit timer is set to 0.5 s and the difference between the values in TB0CP0 and TB0CP1 is 100, the frequency is $100 \div 0.5 \text{ s} = 200 \text{ Hz}$.

3. Pulse width measurement

This mode allows the H-level width of an external pulse to be measured. With the 16-bit timer/event counter operating as a free-running counter counting the pulses from the internal clock input, the external pulse is input via the TB0IN0 pin. Then, the capture function is used to load values from UC0 into TB0CP0 and TB0CP1 on the rising and falling edges of the external trigger pulse respectively. The interrupt INT4 occurs on the falling edge of TB0IN0.

The pulse width is obtained from the difference between the values in TB0CP0 and TB0CP1 and the period of the internal clock.

For example, if the period of the internal clock is $0.8 \mu\text{s}$ and the difference between the values in TB0CP0 and TB0CP1 is 100, the pulse width is $100 \times 0.8 \mu\text{s} = 80 \mu\text{s}$.

In addition, the pulse width which is over the UC0 maximum count time specified by the clock source can be measured by changing software.

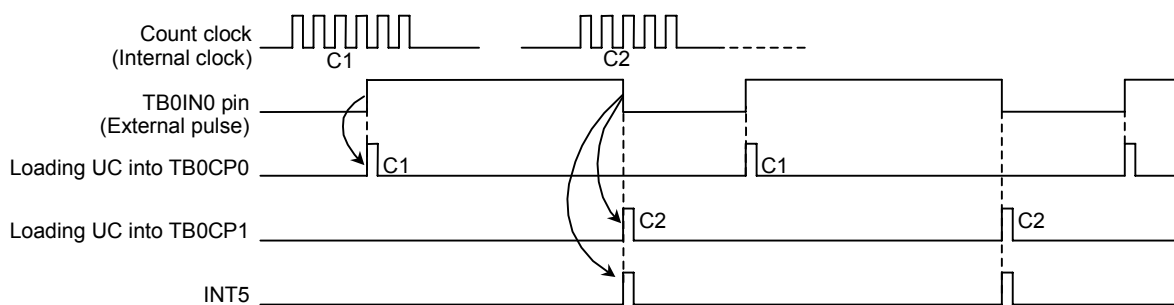


Figure 3.8.11 Pulse Width Measurement

Note: In pulse width measuring mode only (e.g., when $\text{TB0MOD} < \text{TB0CPM1:0} > = 10$), the external interrupt INT5 occurs on the falling edge of the signal input to the TB0IN0 pin. In other modes it occurs on the rising edge.

The width of the L level can be calculated from the difference between the first C1 pulse and the second C0 pulse at the second INT5 interrupt.

The width of the L level is obtained by multiplying the difference between the first C1 and the second C0 at the second INT5 interrupt by the period of the internal clock.

4. Time difference measurement

This mode is used to measure the time difference between the rising edges of the external pulses input via TB0IN0 and TB0IN1.

With the 16-bit timer/event counter (TMRB0) operating as a free-running counter counting the pulses from the internal clock input, load the UC0 value into TB0CP0 on the rising edge of the signal input via TB0IN0. This generates the interrupt INT5.

Similarly, the UC0 value is loaded into TB0CP1 on the rising edge of the signal input via TB0IN1, generating the interrupt INT6.

The time difference between these pulses can be obtained from the difference between the time counts at which loading the up counter value into TB0CP0 and TB0CP1 has been done.

The time difference between these pulses can be obtained by multiplying the value subtracted TB0CP0 from TB0CP1 and the internal clock cycle together at which loading the up counter value into TB0CP0 and TB0CP1 has been done.

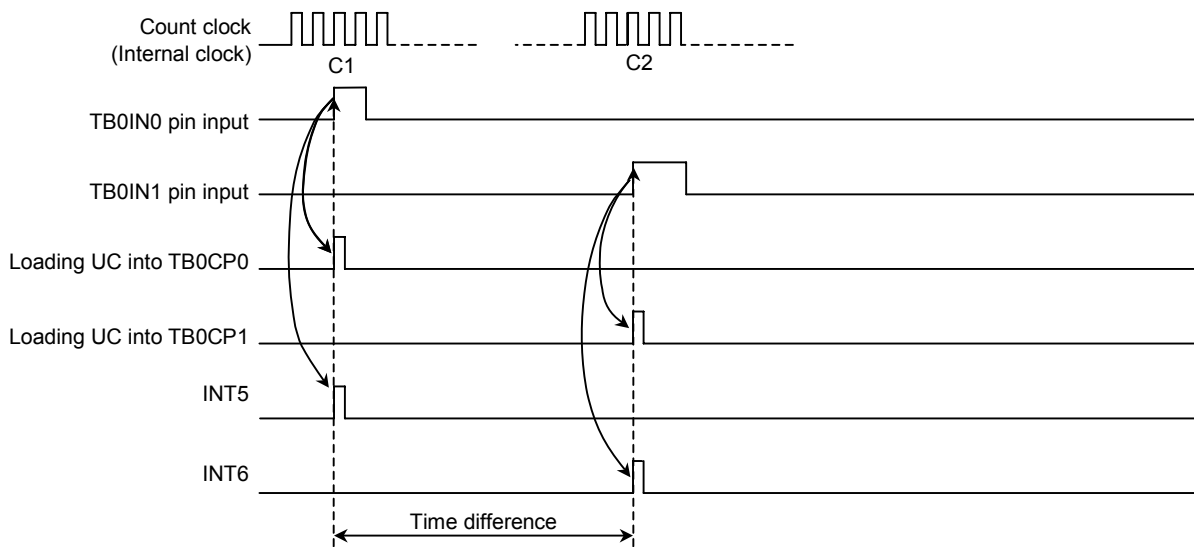


Figure 3.8.12 Time Difference Measurement

3.9 UART

UART mode (Asynchronous transmission) can be selected.

- UART mode
 - Mode 1: 7-bit data
 - Mode 2: 8-bit data
 - Mode 3: 9-bit data

In mode 1 and mode 2, a parity bit can be added. Mode 3 has a wakeup function for making the master controller start slave controllers via a serial link (A multi controller system).

Figure 3.9 2 shows the block diagram of UART.

Both channels operate in the same fashion except for the following points, hence only the operation of channel 0 is explained below.

Table 3.9.1 Differences between UART

	UART
Pin Name	TXD (P82) RXD (P83) $\overline{\text{CTS}}$ (P61)

This chapter contains the following sections:

- 3.9.1 Block Diagram
- 3.9.2 Operation of Each Circuit
- 3.9.3 SFRs
- 3.9.4 Operation in Each Mode

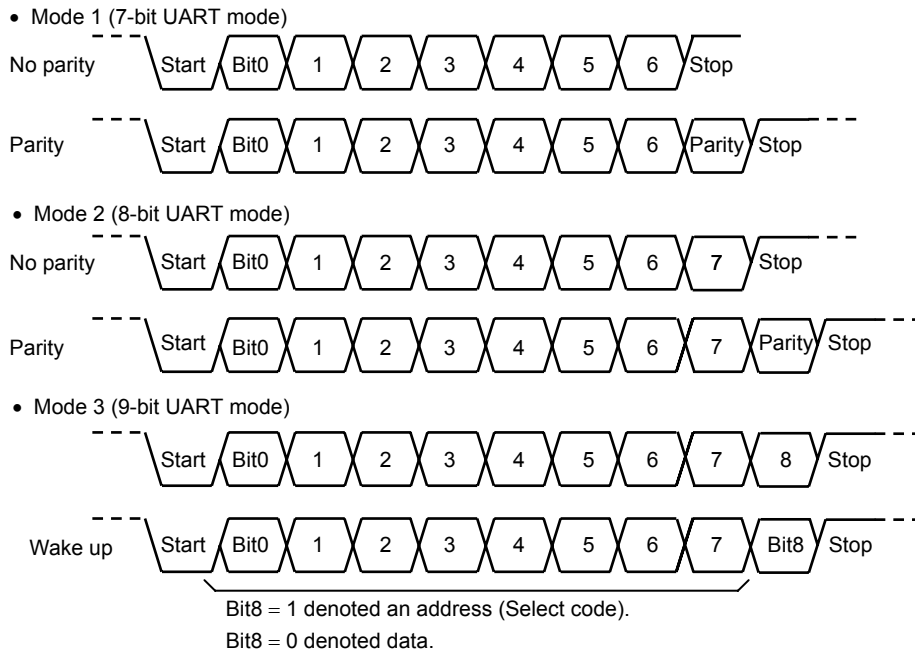


Figure 3.9.1 Data Formats

3.9.1 Block Diagram

Figure 3.9.2 is a block diagram representing serial channel.

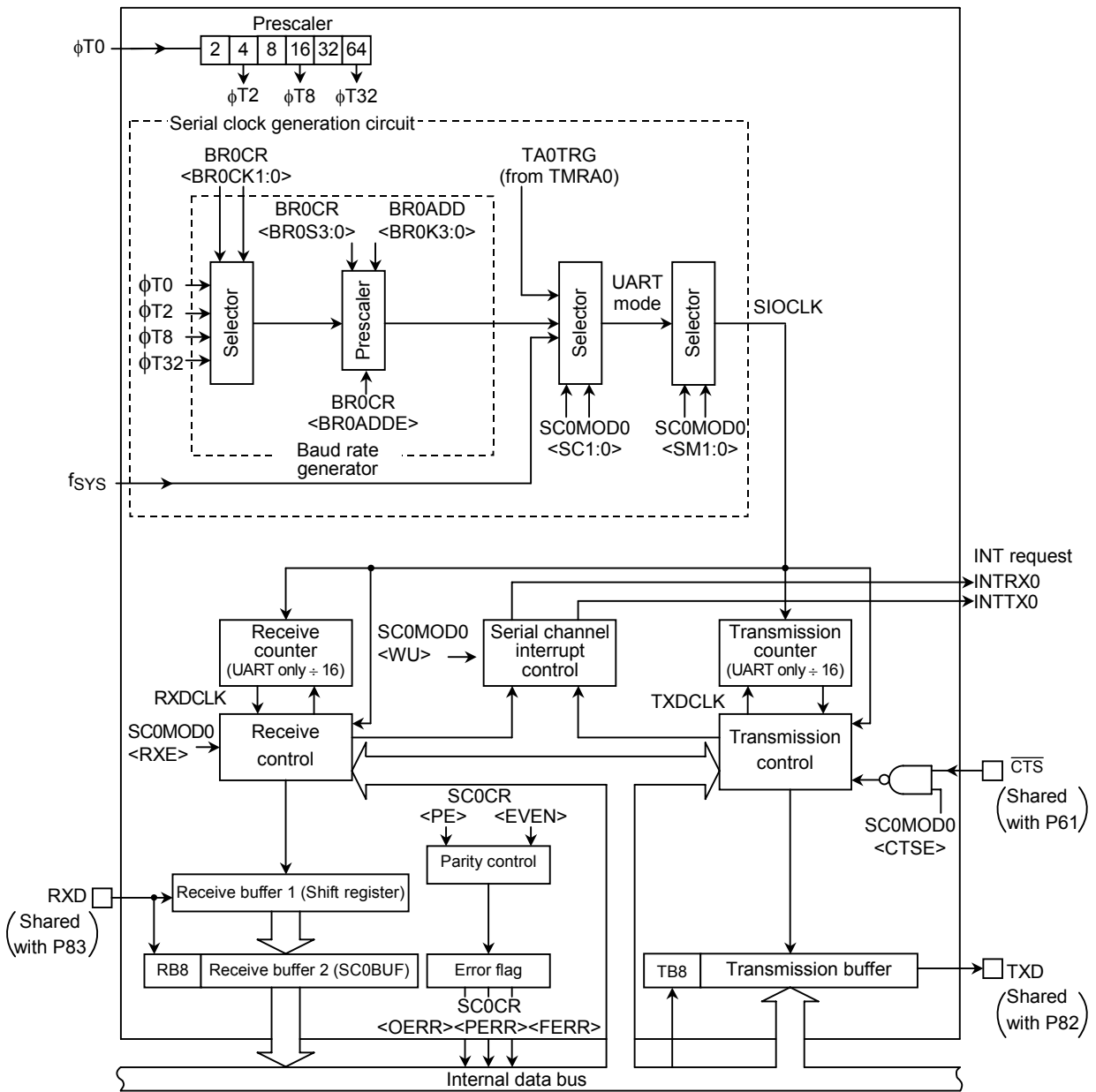


Figure 3.9.2 Block Diagram of the UART

3.9.2 Operation of Each Circuit

(1) Prescaler, prescaler clock select

There is a 6-bit prescaler for waking UART. The clock selected using SYSCR<PRCK1:0> is divided by 4 and input to the prescaler as $\phi T0$. The prescaler can be run by selecting the baud rate generator as the waking serial clock.

Table 3.9.2 shows prescaler clock resolution into the baud rate generator.

Table 3.9.2 Prescaler Clock Resolution to Baud Rate Generator

Select System Clock <SYSCK>	Select Prescaler Clock <PRCK1:0>	Gear Value <GEAR2:0>	Prescaler Output Clock Resolution			
			$\phi T0$	$\phi T2$	$\phi T8$	$\phi T32$
0 (fc)	00 (f_{FPH})	000 (fc)	$2^2/fc$	$2^4/fc$	$2^6/fc$	$2^8/fc$
		001 (fc/2)	$2^3/fc$	$2^5/fc$	$2^7/fc$	$2^9/fc$
		010 (fc/4)	$2^4/fc$	$2^6/fc$	$2^8/fc$	$2^{10}/fc$
		011 (fc/8)	$2^5/fc$	$2^7/fc$	$2^9/fc$	$2^{11}/fc$
		100 (fc/16)	$2^6/fc$	$2^8/fc$	$2^{10}/fc$	$2^{12}/fc$
	10 (fc/16 clock)	XXX	—	$2^8/fc$	$2^{10}/fc$	$2^{12}/fc$

X: Don't care, —: Cannot be used

The baud rate generator selects between 4 clock inputs: $\phi T0$, $\phi T2$, $\phi T8$, and $\phi T32$ among the prescaler outputs.

(2) Baud rate generator

The baud rate generator is a circuit which generates transmitting and receiving clocks which determine the transfer rate of the serial channels.

The input clock to the baud rate generator, $\phi T0$, $\phi T2$, $\phi T8$ or $\phi T32$ is generated by the 6-bit prescaler which is shared by the timers. One of these input clocks is selected using the $BR0CR<BR0CK1:0>$ field in the baud rate generator control register.

The baud rate generator includes a frequency divider, which divides the frequency by 1 or $n + m/16$ ($n = 2$ to 15, $m = 0$ to 15) to 16 values, determining the transfer rate.

The transfer rate is determined by the settings of $BR0CR<BR0ADDE, BR0S3:0>$ and $BR0ADD<BR0K3:0>$.

- In UART mode

(1) When $BR0CR<BR0ADDE> = 0$

The settings $BR0ADD<BR0K3:0>$ are ignored. The baud rate generator divides the selected prescaler clock by N , which is set in $BR0CK<BR0S3:0>$. ($N = 1, 2, 3 \dots 16$)

(2) When $BR0CR<BR0ADDE> = 1$

The $N + (16 - K)/16$ division function is enabled. The baud rate generator divides the selected prescaler clock by $N + (16 - K)/16$ using the value of N set in $BR0CR<BR0S3:0>$ ($N = 2, 3 \dots 15$) and the value of K set in $BR0ADD<BR0K3:0>$ ($K = 1, 2, 3 \dots 15$)

Note: If $N = 1$ or $N = 16$, the $N + (16 - K)/16$ division function is disabled. Clear $BR0CR<BR0ADDE>$ to 0.

- Integer divider (N divider)

For example, when the source clock frequency (f_c) = 12.288 MHz, the input clock frequency = $\phi T2$ ($f_c/16$), the frequency divider N ($BR0CR<BR0S3:0>$) = 5, and $BR0CR<BR0ADDE> = 0$, the baud rate in UART mode is as follows:

$$* \text{ Clock state } \left\{ \begin{array}{l} \text{System clock: High frequency (} f_c \text{)} \\ \text{Clock gear: 1 (} f_c \text{)} \\ \text{Prescaler clock: System clock} \end{array} \right.$$

$$\begin{aligned} \text{Baud rate} &= \frac{f_c/16}{5} \div 16 \\ &= 12.288 \times 10^6 \div 16 \div 5 \div 16 = 9600 \text{ (bps)} \end{aligned}$$

Note: The $N + (16 - K)/16$ division function is disabled and setting $BR0ADD<BR0K3:0>$ is invalid.

- $N + (16 - K)/16$ divider (UART mode)

Accordingly, when the source clock frequency (f_c) = 4.8 MHz, the input clock frequency = $\phi T0$, the frequency divider N ($BR0CR<BR0S3:0>$) = 7, K ($BR0ADD<BR0K3:0>$) = 3, and $BR0CR<BR0ADDE> = 1$, the baud rate in UART mode is as follows:

$$* \text{ Clock state } \left\{ \begin{array}{l} \text{System clock: High frequency (} f_c \text{)} \\ \text{Clock gear: 1 (} f_c \text{)} \\ \text{Prescaler clock: System clock} \end{array} \right.$$

$$\begin{aligned} \text{Baud rate} &= \frac{f_c/4}{7 + (16 - 3)/16} \div 16 \\ &= 4.8 \times 10^6 \div 4 \div (7 + 13/16) \div 16 = 9600 \text{ (bps)} \end{aligned}$$

Table 3.9.3 show examples of UART mode transfer rates.

Additionally, the external clock input is available in the serial clock. The method for calculating the baud rate is explained below.

- In UART mode

Baud rate = External clock input frequency ÷ 16

It is necessary to satisfy (External clock input cycle) $\geq 4/f_c$.

Table 3.9.3 Transfer Rate Selection
(when baud rate generator is used and BR0CR<BR0ADDE> = 0)

fc [MHz]	Input Clock		φT0	φT2	φT8	φT32
	Frequency Divider					
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
12.288000	5		38.400	9.600	2.400	0.600
↑	A		19.200	4.800	1.200	0.300
14.745600	2		115.200	28.800	7.200	1.800
↑	3		76.800	19.200	4.800	1.200
↑	6		38.400	9.600	2.400	0.600
↑	C		19.200	4.800	1.200	0.300
19.6608	1		307.200	76.800	19.200	4.800
↑	2		153.600	38.400	9.600	2.400
↑	4		76.800	19.200	4.800	1.200
↑	8		38.400	9.600	2.400	0.600
↑	10		19.200	4.800	1.200	0.300
22.1184	3		115.200	28.800	7.200	1.800
24.576	1		384.000	96.000	24.000	6.000
↑	2		192.000	48.000	12.000	3.000
↑	4		96.000	24.000	6.000	1.500
↑	5		76.800	19.200	4.800	1.200
↑	8		48.000	12.000	3.000	0.750
↑	A		38.400	9.600	2.400	0.600
↑	10		24.000	6.000	1.500	0.375

Note: The values in this table are calculated for when fc is selected as the system clock, the clock gear is set for fc and the system clock is the prescaler clock input.

Timer out clock (TA0TRG) can be used for source clock of UART mode only.

Calculation method the frequency of TA0TRG

Frequency of TA0TRG = Baud rate × 16

(3) Serial clock generation circuit

This circuit generates the basic clock for transmitting and receiving data.

- In UART mode

The SC0MOD0<SC1:0> setting determines whether the baud rate generator clock, the internal system clock f_{SYS}, the match detect signal from timer TMRA0 or the external clock (SCLK0) is used to generate the basic clock SIOCLK.

(4) Receiving counter

The receiving counter is a 4-bit binary counter used in UART mode which counts up the pulses of the SIOCLK clock. It takes 16 SIOCLK clock pulses to receive 1 bit of data, each data bit is sampled three times – on the 7th, 8th and 9th clock cycles.

The value of the data bit is determined from these three samples using the majority rule.

For example, if the data bit is sampled respectively as 1, 0 and 1 on 7th, 8th and 9th clock cycles, the received data bit is taken to be 1. A data bit sampled as 0, 0 and 1 is taken to be 0.

(5) Receiving control

- In UART mode

The receiving control block has a circuit which detects a start bit using the majority rule. Received bits are sampled three times; when two or more out of three samples are 0, the bit is recognized as the start bit and the receiving operation commences.

The values of the data bits that are received are also determined using the majority rule.

(6) The receiving buffers

To prevent overrun errors, the receiving buffers are arranged in a double-buffer structure.

Received data is stored 1 bit at a time in receiving buffer 1 (which is a shift register). When 7 or 8 bits of data have been stored in receiving buffer 1, the stored data is transferred to receiving buffer 2 (SC0BUF); this causes an INTRX0 interrupt to be generated. The CPU only reads receiving buffer 2 (SC0BUF). Even before the CPU has finished reading the contents of receiving buffer 2 (SC0BUF), more data can be received and stored in receiving buffer 1. However, if receiving buffer 2 (SC0BUF) has not been read completely before all the bits of the next data item are received by receiving buffer 1, an overrun error occurs. If an overrun error occurs, the contents of receiving buffer 1 will be lost, although the contents of receiving buffer 2 and SC0CR<RB8> will be preserved.

SC0CR<RB8> is used to store either the parity bit – added in 8-bit UART mode – or the most significant bit (MSB) – in 9-bit UART mode.

In 9-bit UART mode the wakeup function for the slave controller is enabled by setting SC0MOD0<WU> to 1. In this mode INTRX0 interrupts occur only when the value of SC0CR<RB8> is 1.

(7) Transmission counter

The transmission counter is a 4-bit binary counter which is used in UART mode and which like the receiving counter, counts the SIOCLK clock pulses, a TXDCLK pulse is generated every 16 SIOCLK clock pulses.

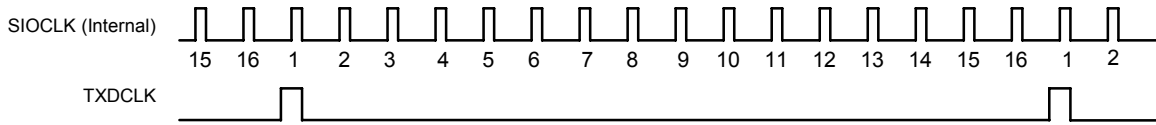


Figure 3.9.3 Generation of the Transmission Clock

(8) Transmission controller

- In UART mode

When transmission data sent from the CPU is written to the transmission buffer, transmission starts on the rising edge of the next TXDCLK, generating a transmission shift clock TXDSFT.

Handshake function

Use of \overline{CTS} pin allows data can be sent in units of one frame; thus, overrun errors can be avoided. The handshake functions is enabled or disabled by the SC0MOD<CTSE> setting.

When the \overline{CTS} pin foes high on completion of the current data send, data transmission is halted until the \overline{CTS} pin foes low again. However, the INTTX0 interrupt is generated, it requests the next data send to the CPU. The next data is written in the transmission buffer and data sending is halted.

Although there is no \overline{RTS} pin, a handshake function can easily be configured by assigning any port to perform the \overline{RTS} function. The \overline{RTS} should be output high to request send data halt after data receive is completed by software in the RXD interrupt routine.

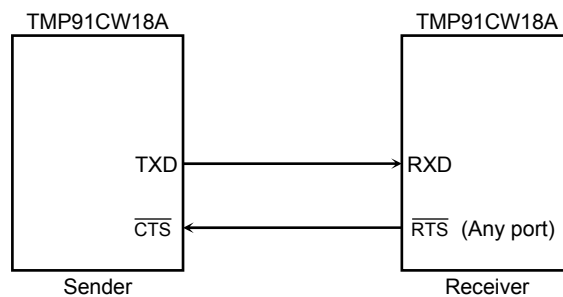
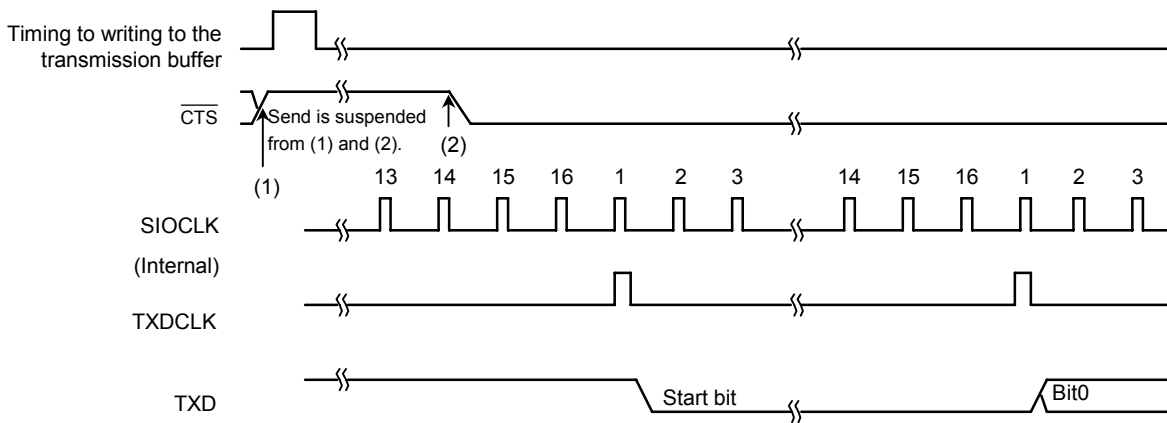


Figure 3.9.4 Handshake Function



Note 1: If the $\overline{\text{CTS}}$ signal goes high during transmission, no more data will be sent after completion of the current transmission.

Note 2: Transmission starts on the first falling edge of the TXDCLK clock after the $\overline{\text{CTS}}$ signal has fallen.

Figure 3.9.5 $\overline{\text{CTS}}$ (Clear to send) Timing

(9) Transmission buffer

The transmission buffer (SC0BUF) shifts out and sends the transmission data written from the CPU, in order one bit at a time starting with the least significant bit (LSB) and finishing with the most significant bit (MSB). When all the bits have been shifted out, the empty transmission buffer generates an INTTX0 interrupt.

(10) Parity control circuit

When SC0CR<PE> in the serial channel control register is set to 1, it is possible to transmit and receive data with parity. However, parity can be added only in 7-bit UART mode or 8-bit UART mode. The SC0CR<EVEN> field in the serial channel control register allows either even or odd parity to be selected.

In the case of transmission, parity is automatically generated when data is written to the transmission buffer SC0BUF. The data is transmitted after the parity bit has been stored in SC0BUF<TB7> in 7-bit UART mode or in SC0MOD0<TB8> in 8-bit UART mode. SC0CR<PE> and SC0CR<EVEN> must be set before the transmission data is written to the transmission buffer.

In the case of receiving, data is shifted into receiving buffer 1, and the parity is added after the data has been transferred to receiving buffer 2 (SC0BUF), and then compared with SC0BUF<RB7> in 7-bit UART mode or with SC0CR<RB8> in 8-bit UART mode. If they are not equal, a parity error is generated and the SC0CR<PERR> flag is set.

(11) Error flags

Three error flags are provided to increase the reliability of data reception.

1. Overrun error <OERR>

If all the bits of the next data item have been received in receiving buffer 1 while valid data still remains stored in receiving buffer 2 (SC0BUF), an overrun error is generated.

Following show over run generating process flow example.

(Receiving interrupts routine)

(1) Read of receiving buffer

(2) Read of error flag

(3) If <OERR> = "1"

Then

a) Set to receiving enable write 0 to <RXE>

b) Wait for end of now frame

c) Read of receiving buffer

d) Read of error flag

e) Set to receiving enable write 1 to <RXE>

f) Request transmission again

(4) Other process

2. Parity error <PERR>

The parity generated for the data shifted into receiving buffer 2 (SC0BUF) is compared with the parity bit received via the RXD pin. If they are not equal, a parity error is generated.

3. Framing error <FERR>

The stop bit for the received data is sampled three times around the center. If the majority of the samples are 0, a framing error is generated.

(12) Timing generation

1. In UART mode

Receiving

Mode	9-Bit (Note)	8-Bit + Parity (Note)	8-Bit, 7-Bit + Parity, 7-Bit
Interrupt timing	Center of last bit (Bit8)	Center of last bit (Parity bit)	Center of stop bit
Framing error timing	Center of stop bit	Center of stop bit	Center of stop bit
Parity error timing	—	Center of last bit (Parity bit)	Center of stop bit
Overrun error timing	Center of last bit (Bit8)	Center of last bit (Parity bit)	Center of stop bit

Note: In 9-bit mode and 8-bit + Parity mode, interrupts coincide with the 9th bit pulse.

Thus, when servicing the interrupt, it is necessary to allow a 1-bit period to elapse (so that the stop bit can be transferred) in order to allow proper framing error checking.

Transmitting

Mode	9-Bit	8-Bit + Parity	8-Bit, 7-Bit + Parity, 7-Bit
Interrupt timing	Just before stop bit is transmitted.	Just before last data bit is transmitted.	Just before last data bit is transmitted.

3.9.3 SFRs

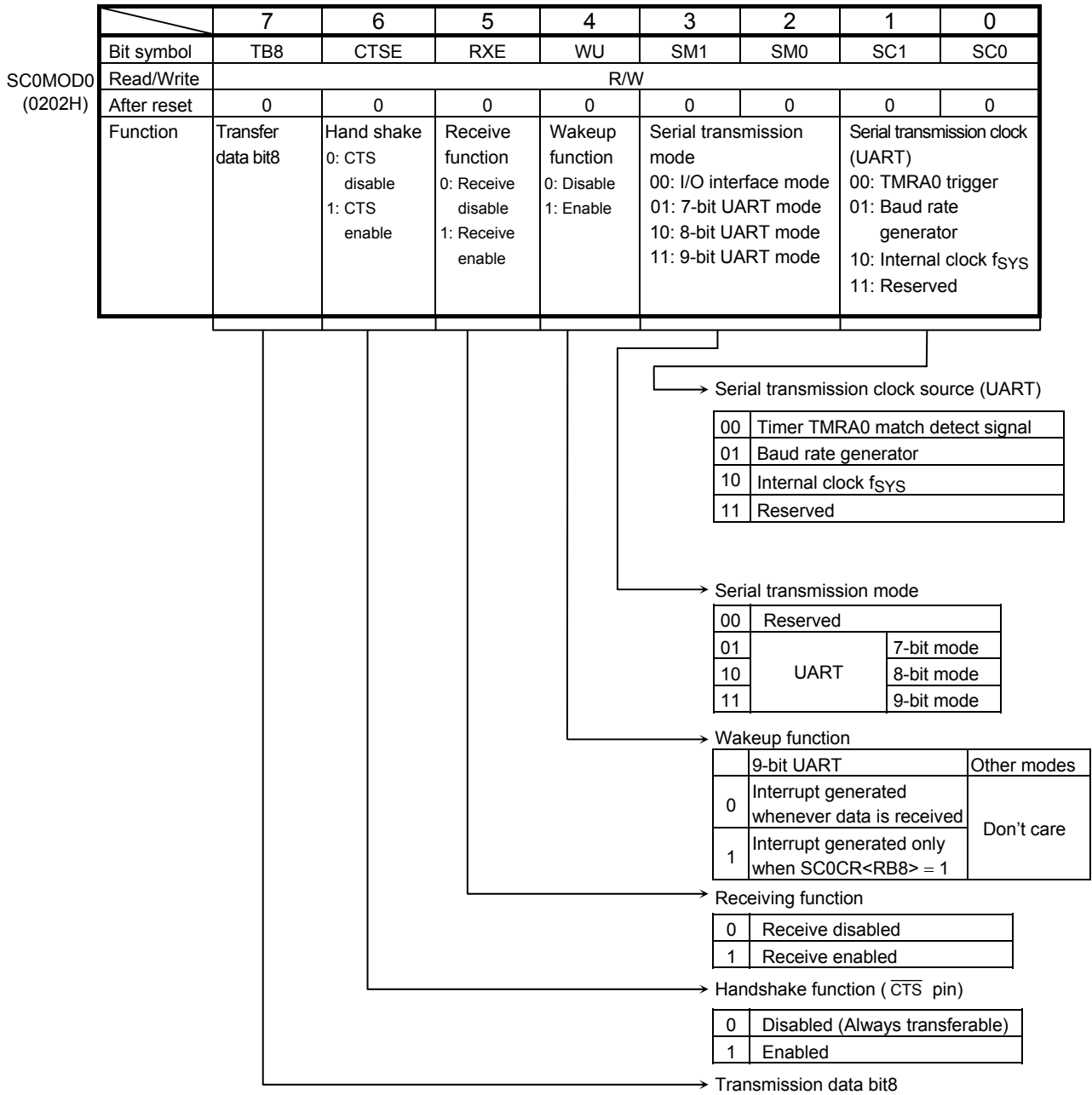
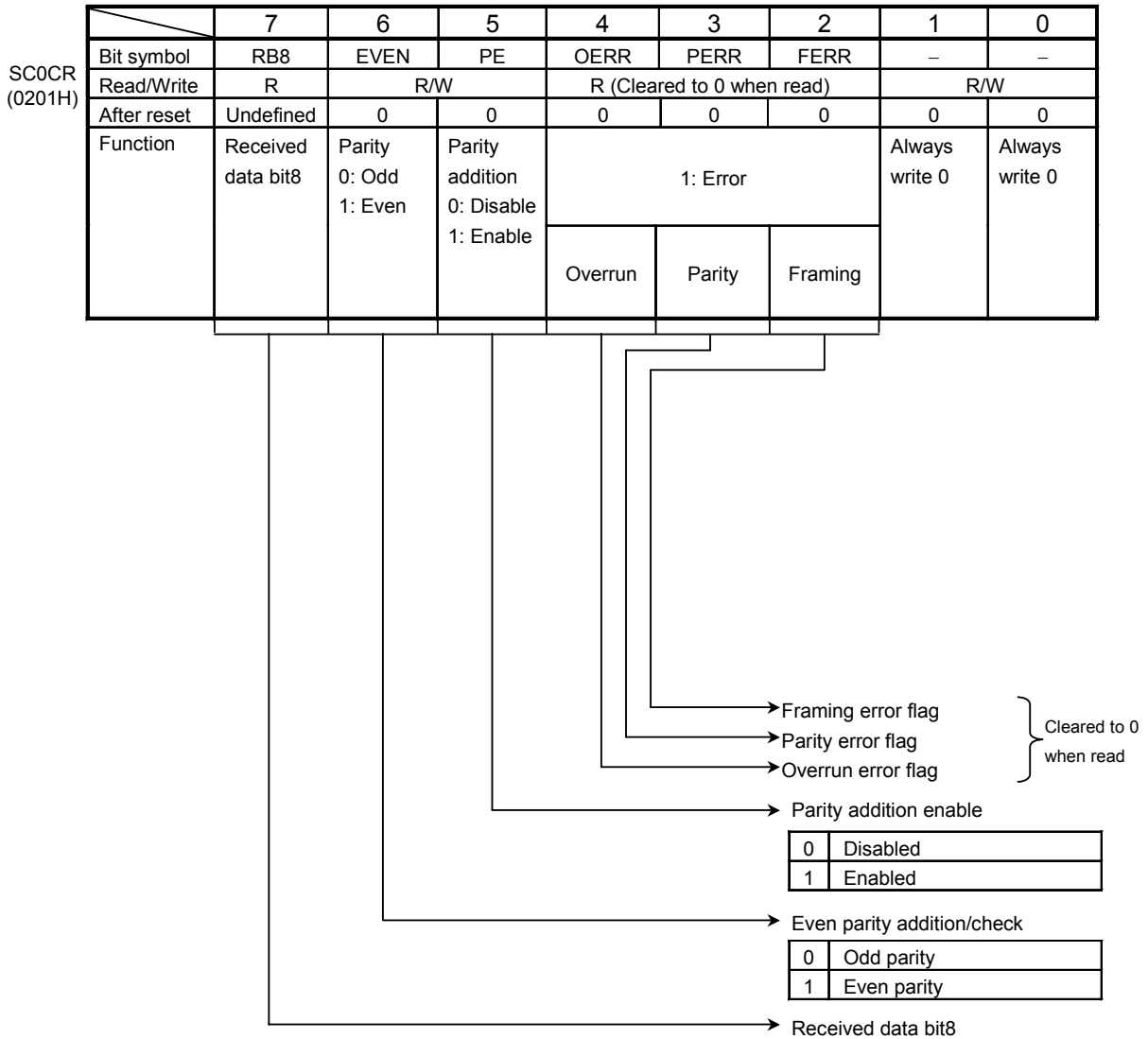


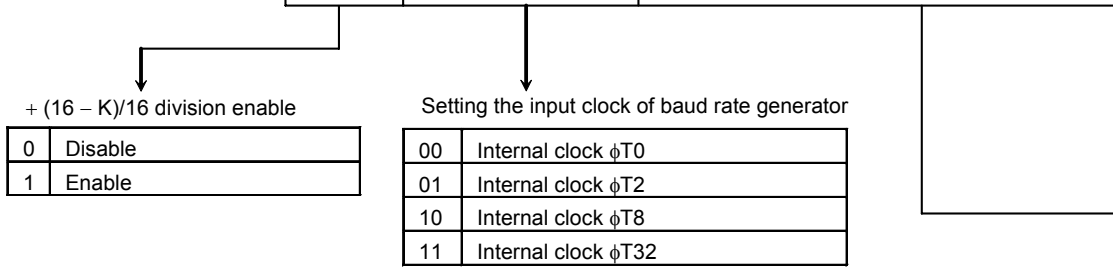
Figure 3.9.6 Serial Mode Control Register 0 (SC0MOD0)



Note: As all error flags are cleared after reading, do not test only a single bit with a bit testing instruction.

Figure 3.9.7 Serial Control Register 0 (SC0CR)

	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	Always write 0	+ (16 - K)/16 division 0: Disable 1: Enable	00: ϕ T0 01: ϕ T2 10: ϕ T8 11: ϕ T32	Setting of the divided frequency				



+ (16 - K)/16 division enable

0	Disable
1	Enable

Setting the input clock of baud rate generator

00	Internal clock ϕ T0
01	Internal clock ϕ T2
10	Internal clock ϕ T8
11	Internal clock ϕ T32

	7	6	5	4	3	2	1	0
Bit symbol					BR0K3	BR0K2	BR0K1	BR0K0
Read/Write					R/W			
After reset					0	0	0	0
Function					Sets frequency divisor "K" (Divided by $N = (16 - K)/16$)			

Sets baud rate generator frequency divisor ←

	BR0CR<BR0ADDE> = 1	BR0CR<BR0ADDE> = 0
BR0CR <BR0S3:0> BR0ADD <BR0K3:0>	0000 (N = 16) or 0001 (N = 1)	0010 (N = 2) to 1111 (N = 15)
0000	Disable	Disable
0001 (K = 1) to 1111 (K = 15)	Disable	Divided by $N + (16 - K)/16$
		Divided by N

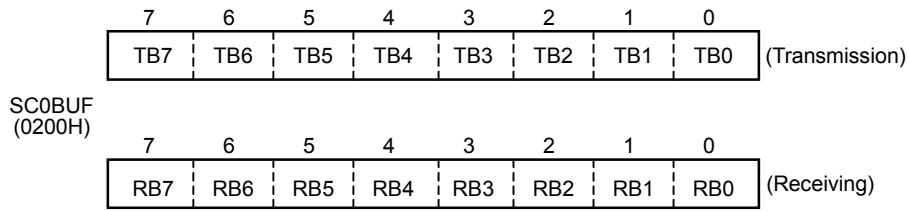
Note1: Availability of +(16-K)/16 division function

N	UART Mode
2 to 15	○
1, 16	×

The baud rate generator can be set "1" in UART mode and disable +(16-K)/16 division function. Don't use in I/O interface mode.

Note2: Set BR0CR <BR0ADDE> to 1 after setting K (K = 1 to 15) to BR0ADD<BR0K3:0> when +(16-K)/16 division function is used. Writes to unused bits in the BR0ADD register do not affect operation, and undefined data is read from these unused bits.

Figure 3.9.8 Baud Rate Generator Control (BR0CR, BR0ADD)



Note: Prohibit read-modify-write for SC0BUF.

Figure 3.9.9 Serial Transmission/Receiving Buffer Registers (Channel 0, SC0BUF)

	7	6	5	4	3	2	1	0
SC0MOD1 (0205H)	Bit symbol	I2S0	—	/	/	/	/	/
	Read/Write	R/W	R/W	/	/	/	/	/
	After reset	0	0	/	/	/	/	/
	Function	IDLE2 0: Stop 1: Run	Always write "0".					

Figure 3.9.10 Serial Mode Control Register 1 (Channel 0, SC0MOD1)

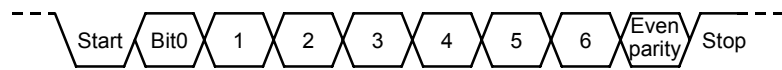
3.9.4 Operation in Each Mode

(1) Mode 1 (7-bit UART mode)

7-bit UART mode is selected by setting the serial channel mode register SC0MOD0<SM1:0> field to 01.

In this mode, a parity bit can be added. Use of a parity bit is enabled or disabled by the setting of the serial channel control register SC0CR<PE> bit; whether even parity or odd parity will be used is determined by the SC0CR<EVEN> setting when SC0CR<PE> is set to 1 (Enabled).

Setting example: When transmitting data of the following format, the control registers should be set as described below. This explanation applies to channel 0.



← Transmission direction (Transmission rate: 2400 bps at $f_c = 12.288$ MHz)

* Clock state { System clock: High frequency (f_c)
 Clock gear: 1 (f_c)
 Prescaler clock: System clock

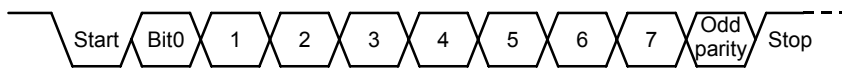
	7 6 5 4 3 2 1 0	
P8CR	← - - - - - 1 - -	} Set P82 to function as the TXD pin.
P8FC	← - - - - - 1 - -	
SC0MOD	← X 0 - X 0 1 0 1	Select 7-bit UART mode.
SC0CR	← X 1 1 X X X 0 0	Add even parity.
BR0CR	← 0 0 1 0 0 1 0 1	Set the transfer rate to 2400 bps.
INTEVART	← X 1 0 0 - - - -	Enable the INTTX0 interrupt and set it to interrupt level 4.
SC0BUF	← * * * * * * * *	Set data for transmission.

X: Don't care, -: No change

(2) Mode 2 (8-bit UART mode)

8-bit UART mode is selected by setting SC0MOD0<SM1:0> to 10. In this mode a parity bit can be added (Use of a parity bit is enabled or disabled by the setting of SC0CR<PE>), whether even parity or odd parity will be used is determined by the SC0CR<EVEN> setting when SC0CR<PE> is set to 1 (Enabled).

Setting example: When receiving data of the following format, the control registers should be set as described below.



← Transmission direction (Transmission rate: 9600 bps at $f_c = 12.288$ MHz)

* Clock state { System clock: High frequency (f_c)
 Clock gear: 1 (f_c)
 Prescaler clock: System clock

Main settings

	7 6 5 4 3 2 1 0	
P8CR	← - - - - 0 - - -	Set P83 to function as the RXD pin.
SC0MOD	← - 0 1 X 1 0 0 1	Enable receiving in 8-bit UART mode.
SC0CR	← X 0 1 X X X 0 0	Add even parity.
BR0CR	← 0 0 0 1 0 1 0 1	Set the transfer rate to 9600 bps.
INTEVART	← - - - - X 1 0 0	Enable the INTRX0 interrupt and set it to interrupt level 4.

Interrupt processing

Acc	← SC0CR AND 00011100	} Check for errors.
if Acc	≠ 0 then ERROR	
Acc	← SC0BUF	Read the received data.

X: Don't care, -: No change

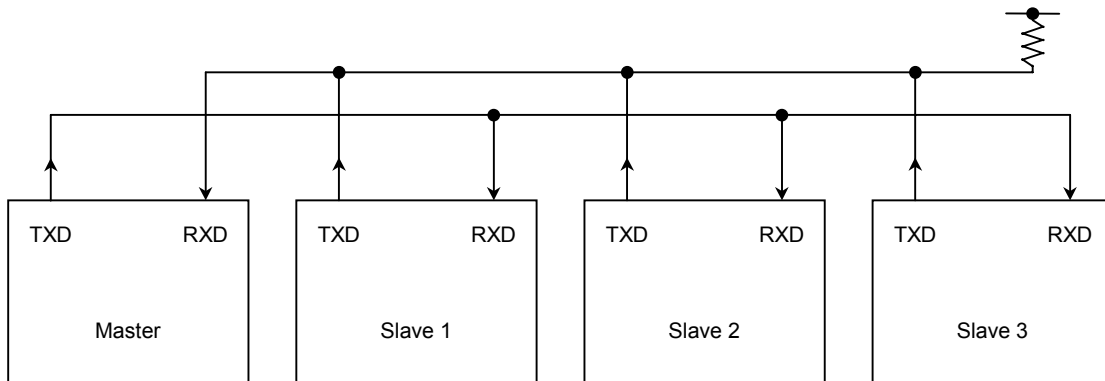
(3) Mode 3 (9-bit UART mode)

9-bit UART mode is selected by setting SC0MOD0<SM1:0> to 11. In this mode parity bit cannot be added.

In the case of transmission the MSB (9th bit) is written to SC0MOD0<TB8>. In the case of receiving it is stored in SC0CR<RB8>. When the buffer is written and read, the MSB is read or written first, before the rest of the SC0BUF data.

Wakeup function

In 9-bit UART mode, the wakeup function for slave controllers is enabled by setting SC0MOD0<WU> to 1. The interrupt INTRX0 can only be generated when <RB8> = 1.

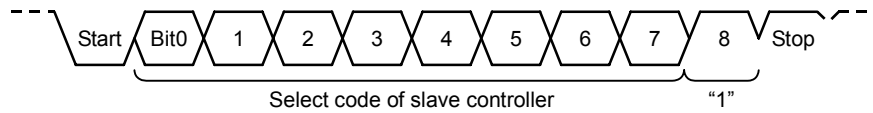


Note: The TXD pin of each slave controller must be in open-drain output mode.

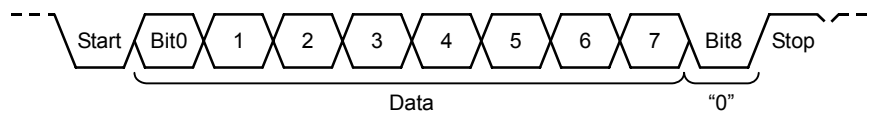
Figure 3.9.11 Serial Link Using Wakeup Function

Protocol

1. Select 9-bit UART mode on the master and slave controllers.
2. Set the SC0MOD0<WU> bit on each slave controller to 1 to enable data receiving.
3. The master controller transmits data one frame at a time. Each frame includes an 8-bit select code which identifies a slave controller. The MSB (Bit8) of the data (<TB8>) is set to 1.

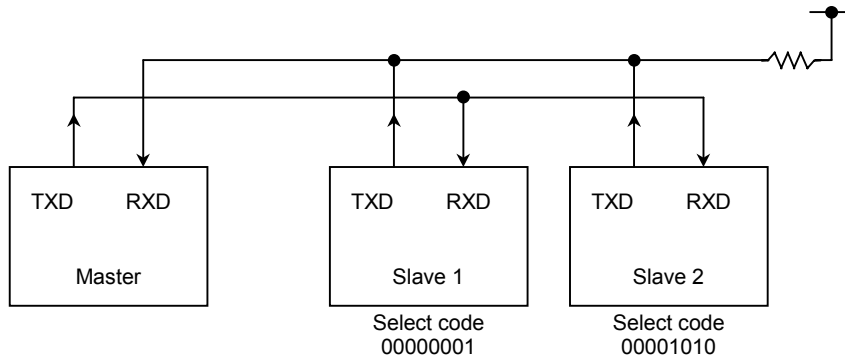


4. Each slave controller receives the above frame. Each controller checks the above select code against its own select code. The controller whose code matches clears its WU bit to 0.
5. The master controller transmits data to the specified slave controller. (The controller whose SC0MOD<WU> bit has been cleared to 0.) The MSB (Bit8) of the data (<TB8>) is cleared to 0.



6. The other slave controllers (whose <WU> bits remain at 1) ignore the received data because their MSBs (Bit8 or <RB8>) are cleared to 0, disabling INTRX0 interrupts. The slave controller whose WU bit = 0 can also transmit to the master controller. In this way, it can signal the master controller that the data transmission from the master controller has been completed.

Setting example: To link two slave controllers serially with the master controller using the internal clock f_{SYS} as the transfer clock.



Since serial channels 0 and 1 operate in exactly the same way, channel 0 only is used for the purposes of this explanation.

- Setting the master controller

Main

P8CR	← - - - - 0 1 - 1	} Set P82 and P83 to function as the TXD and RXD pins respectively.
P8FC	← - - - - X 1 - -	
INTEUART	← X 1 0 0 X 1 0 1	Enable the INTTX0 interrupt and set it to interrupt level 4.
		Enable the INTRX0 interrupt and set it to interrupt level 5.
SC0MOD0	← 1 0 1 0 1 1 1 0	Set f _{SYS} as the transmission clock for 9-bit UART mode.
SC0BUF	← 0 0 0 0 0 0 0 1	Set the select code for slave controller 1.

INTTX0 interrupt

SC0MOD0	← 0 - - - - - - -	Clear TB8 to TB0.
SC0BUF	← * * * * * * * *	Set data for transmission.

- Setting the slave controller

Main

P8CR	← - - - - - 0 1 0 -	} Select P82 and P83 to function as the TXD and RXD pins respectively (Open-drain output).
P8FC	← - - - - - X 1 - -	
P8ODE	← - - - - - 1 - -	Enable INTRX0 and INTTX0.
INTEUART	← X 1 0 1 X 1 1 0	Set <WU> to 1 in 9-bit UART transmission mode using f _{SYS} as the transfer clock.
SC0MOD0	← 0 0 1 1 1 1 1 0	

INTRX0 interrupt

```

Acc ← SC0BUF
if Acc = select code
then SC0MOD0 ← - - - - 0 - - - - Clear <WU> to 0.
    
```

3.10 Serial Bus Interface 0 (SBI: SIO/I²C Bus)

The TMP91CW18A has a one-channel serial bus interface which employs a clocked-synchronous 8-bit SIO mode and an I²C bus mode.

The serial bus interface is connected to an external device through P80 (SDA0) and P81 (SCL0) in the I²C bus mode, and through P76 (SCK0), P80 (SO0) and P81 (SI0) in the clocked synchronous 8-bit SIO mode.

Each pin is specified as follows.

	XXODE	XXCR		XXFC	
	P80DE<P820DE, P800DE>	P8CR<P81C, P80C>	P7CR<P76C>	P8FC<P81F, P80F>	P7FC<P76F>
I ² C bus mode	11	11	X	11	0
Clocked-synchronous 8-bit SIO mode	XX	01	1	11	1

X: Don't care

3.10.1 Configuration

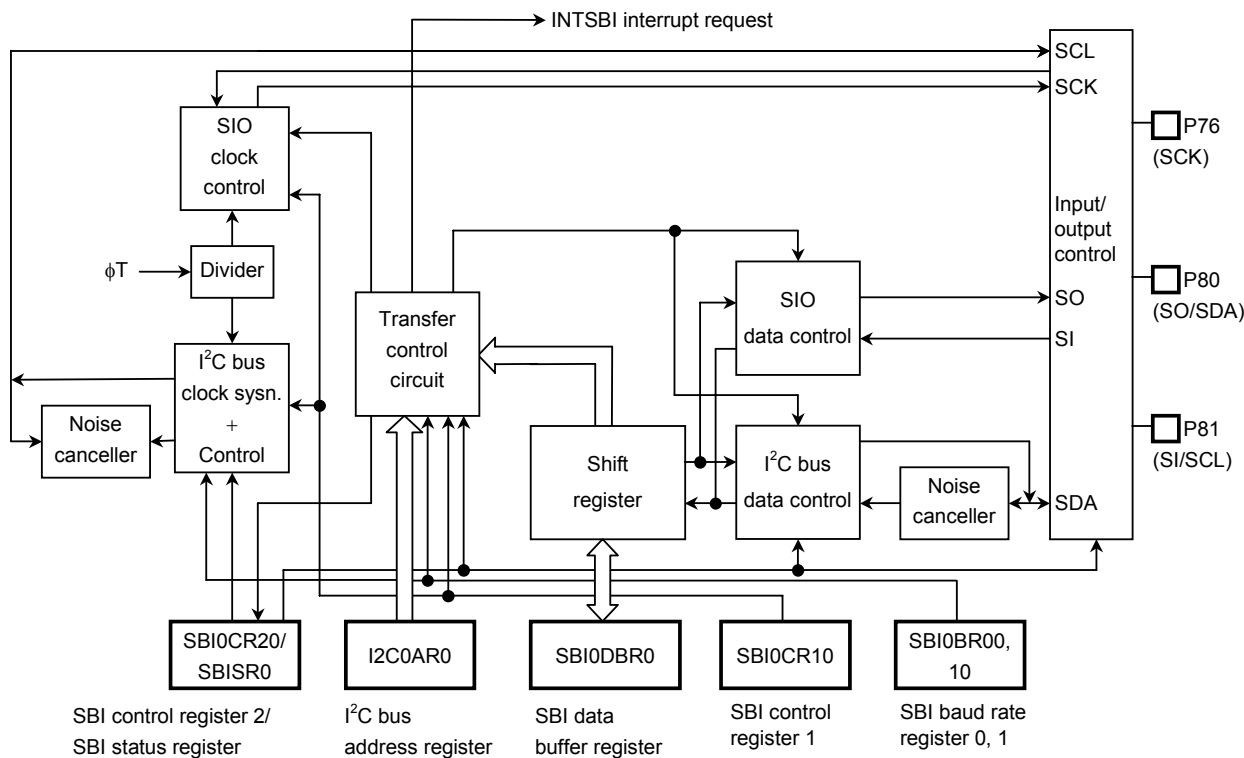


Figure 3.10.1 Serial Bus Interface 0 (SBI)

3.10.2 Serial Bus Interface (SBI) Control

The following registers are used to control the serial bus interface and monitor the operation status.

- Serial bus interface control register 1 (SBI0CR10)
- Serial bus interface control register 2 (SBI0CR20)
- Serial bus interface data buffer register (SBI0DBR0)
- I²C bus address register (I2C0AR0)
- Serial bus interface status register (SBISR0)
- Serial bus interface baud rate register 0 (SBI0BR00)
- Serial bus interface baud rate register 1 (SBI0BR10)

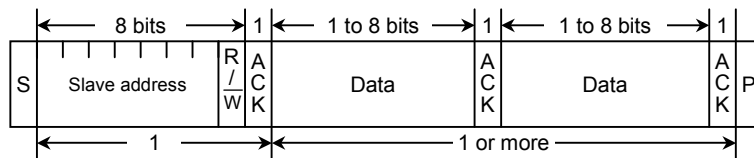
The above registers differ depending on a mode to be used.

Refer to section 3.10.4 “I²C Bus Mode Control” and 3.10.7 “Clocked-synchronous 8-Bit SIO Mode Control”.

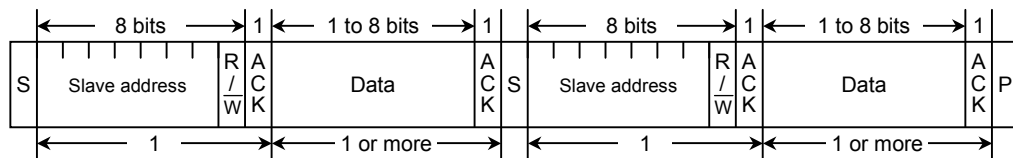
3.10.3 The Data Formats in the I²C Bus Mode

The data formats in the I²C bus mode are shown below.

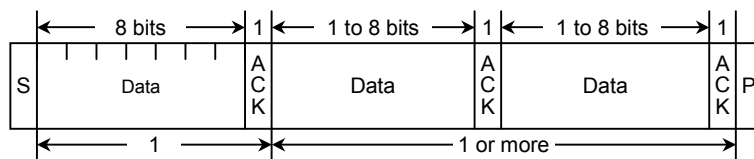
(a) Addressing format



(b) Addressing format (with restart)



(c) Free data format (Data transferred from master device to slave device)

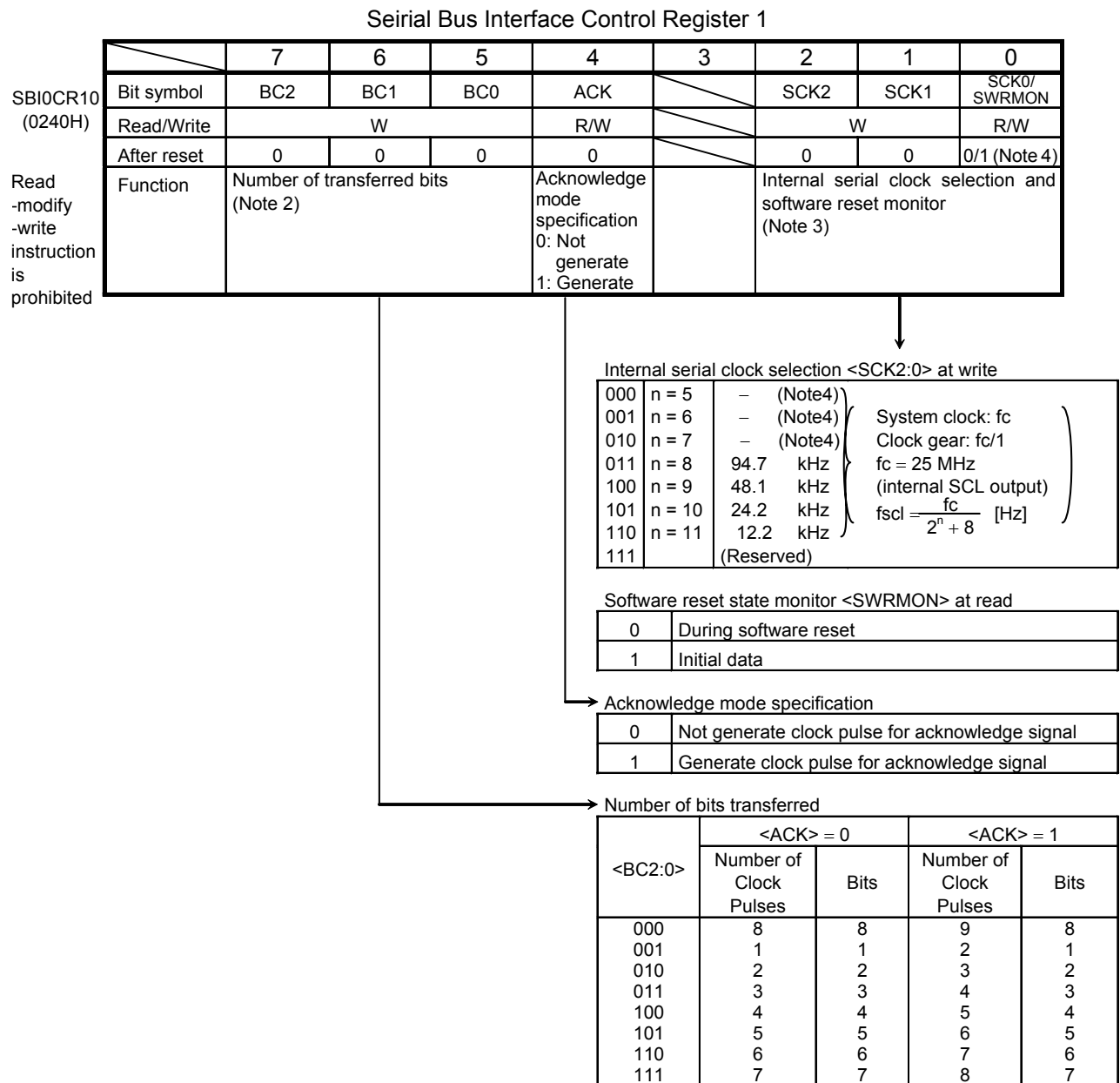


- S: Start condition
- R/W: Direction bit
- ACK: Acknowledge bit
- P: Stop condition

Figure 3.10.2 Data Format in the I²C Bus Mode

3.10.4 I²C Bus Mode Control

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



Note 1: Set the <BC2:0> to "000" before switching to a clock synchronous 8-bit SIO mode.

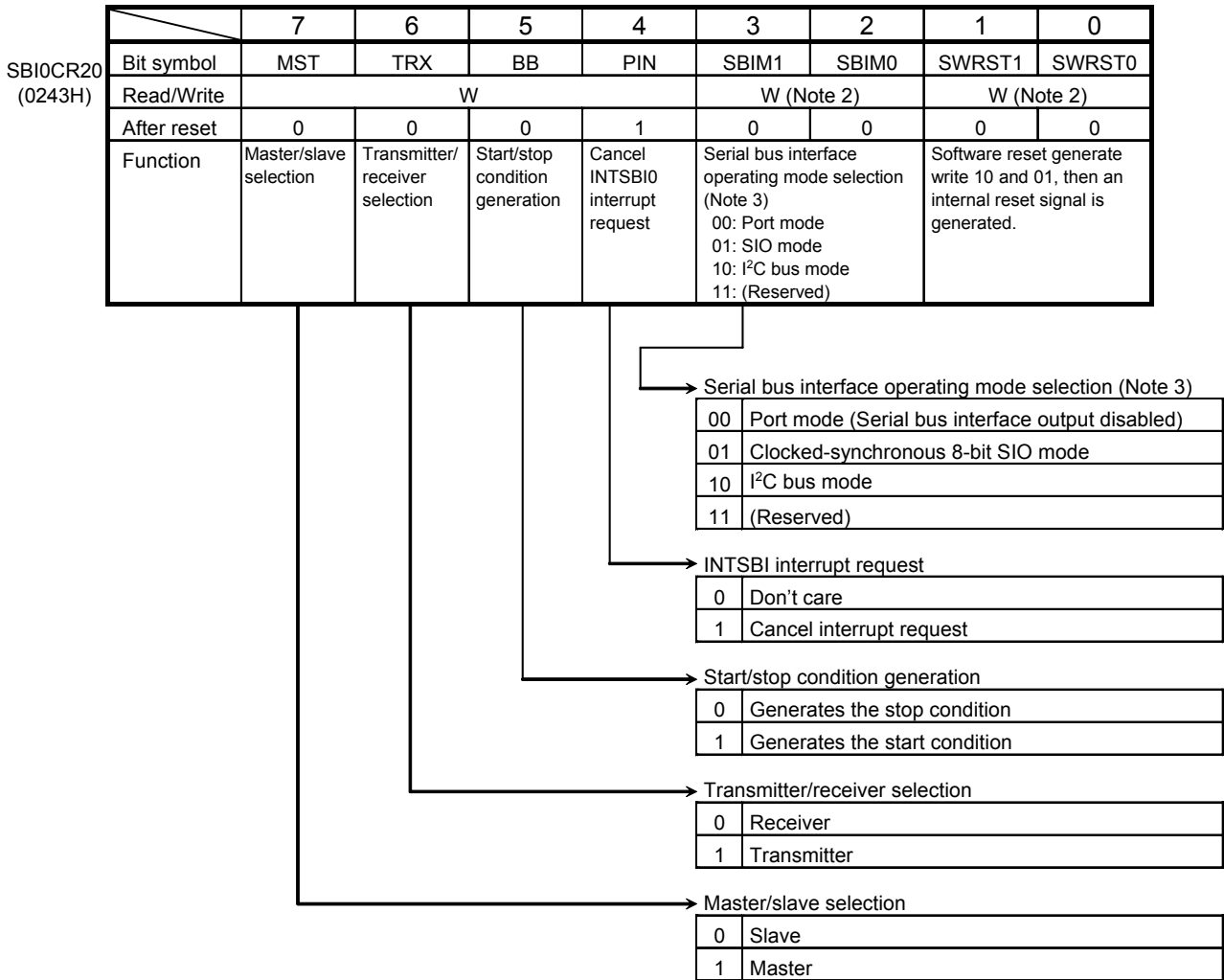
Note 2: For the frequency of the SCL line clock, see 3.10.5 (3) "Serial clock".

Note 3: Initial data of SCK0 is 0, SWRMON is 1.

Note 4: This I²C bus circuit does not support Fast mode, it supports standard mode only. Although the I²C bus circuit itself allows the setting of a baud rate over 100 kbps, the compliance with the I²C specification is not guaranteed in that case.

Figure 3.10.3 Registers for the I²C Bus Mode (1/4)

Serial Bus Interface Control Register 2



Note 1: Read-modify-write instruction is prohibited for SBI0CR20.

Note 2: Reading this register function as SBISR0 register.

Note 3: Switch a mode to port mode after confirming that the bus is free.

Switch a mode between I²C bus mode and clock-synchronous 8-bit SIO mode after confirming that input signals via port are high level.

Figure 3.10.4 Registers for the I²C Bus Mode (2/4)

Serial Bus Interface Status Register

SBISR0
(0243H)

	7	6	5	4	3	2	1	0
Bit symbol	MST	TRX	BB	PIN	AL	AAS	AD0	LRB
Read/Write	R							
After reset	0	0	0	1	0	0	0	0
Function	Master/ slave status monitor	Transmitter/ receiver status monitor	I ² C bus status monitor	INTSBI0 interrupt request monitor	Arbitration lost detection monitor 0: – 1: Detected	Slave address match detection monitor 0: Undetected 1: Detected	GENERAL CALL detection monitor 0: Undetected 1: Detected	Last received bit monitor 0: 0 1: 1



Note 1: Read-modify-write instruction is prohibited for SBISR0.

Note 2: Writing in this register functions as SBI0CR20.

Figure 3.10.5 Registers for the I²C Bus Mode (3/4)

Serial Bus Interface Baud Rate Register 0

	7	6	5	4	3	2	1	0
SBI0BR00 (0244H)	Bit symbol	–	I2SBI0					
	Read/Write	W	R/W					
	After reset	0	0					
Read-modify-write instructions are prohibited.	Function	Always write 0	IDLE2 0: Stop 1: Run					

Operation during IDLE 2 mode

0	Stop
1	Operation

Serial Bus Interface Baud Rate Register 1

	7	6	5	4	3	2	1	0
SBI0BR10 (0245H)	Bit symbol	P4EN	–					
	Read/Write	W	W					
	After reset	0	0					
Read-modify-write instructions are prohibited.	Function	Internal clock 0: Stop 1: Operate	Always write 0					

Baud rate clock control

0	Stop
1	Operation

Serial Bus Interface Data Buffer Register

	7	6	5	4	3	2	1	0	
SBI0DBR0 (0241H)	Bit symbol	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	Read/Write	R (Received)/W (Transfer)							
	After reset	Undefined							

- Note 1: Read-modify-write instruction is prohibited for SBI0DBR0.
- Note 2: When writing transmitted data, start from the MSB (Bit7). Receiving data is placed from LSB (Bit0).
- Note 3: SBI0DBR0 can't be read the written data. Therefore read-modify-write instruction (e.g., "BIT" instruction) is prohibited.
- Note 4: Written data in SBI0DBR0 is cleared by INTSBI0 signal.

I²C Bus Address Register

	7	6	5	4	3	2	1	0	
I2C0AR0 (0242H)	Bit symbol	SA6	SA5	SA4	SA3	SA2	SA1	SA0	ALS
	Read/Write	W							
	After reset	0	0	0	0	0	0	0	0
	Function	Slave address selection for when device is operating as slave device.							Address recognition mode specification.

Note: Read-modify-write instruction is prohibited for I2C0AR0.

Address recognition mode specification

0	Slave address recognition
1	Non slave address recognition

Figure 3.10.6 Registers for the I²C Bus Mode (4/4)

3.10.5 Control in I²C Bus Mode

(1) Specifying acknowledge mode

Set the SBI0CR10<ACK> to 1 for operation in the acknowledge mode. The TMP91CW18A generates an additional clock pulse for an acknowledge signal when operating in master mode. In the transmitter mode during the clock pulse cycle, the SDA pin is released in order to receive the acknowledge signal from the receiver. In the receiver mode during the clock pulse cycle, the SDA pin is set to the low in order to generate the acknowledge signal.

Clear the <ACK> to 0 for operation in the non-acknowledge mode, the TMP91CW18A does not generate a clock pulse for the acknowledge signal when operating in the master mode.

(2) Number of transfer bits

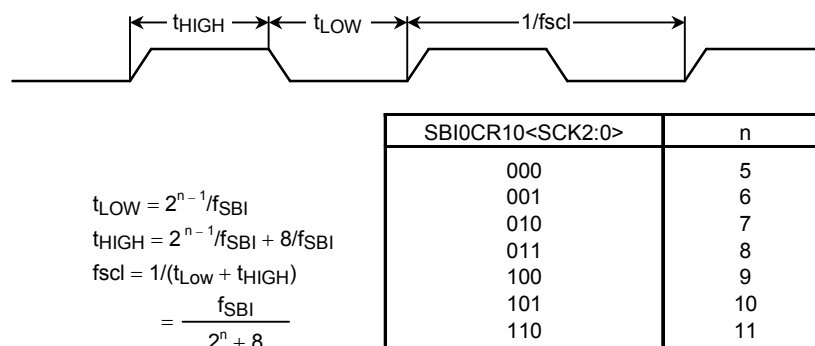
The SBI0CR10<BC2:0> is used to select a number of bits for next transmitting and receiving data.

Since SBI0CR10<BC2:0> is cleared to 000 on start-up, a slave address and direction bit transmissions are executed in 8 bits. Other than these, the <BC2:0> retains a specified value.

(3) Serial clock

1. Clock source

SBI0CR10<SCK2:0> is used to specify the maximum transfer frequency for output on the SCL pin in master mode. Set a communication baud rate that meets the I²C bus specification, such as the shortest pulse width of t_{LOW}, based on the equations shown below.



Note 1: f_{SBI} is the clock f_{FPH}.

Note 2: f_{SBI} is the clock either fc/16 or f_{FPH} which is selected SYSCR0<PRCK1:0>.

Figure 3.10.7 Clock Source

2. Clock synchronization

In the I²C bus mode, in order to wired-AND a bus, a master device which pulls down a clock line to low level, in the first place, invalidate a clock pulse of another master device which generates a high-level clock pulse. The master device with a high-level clock pulse needs to detect the situation and implement the following procedure.

The TMP91CW18A has a clock synchronization function which allows normal data transfer even when more than one master exists on the bus.

The following example explains the clock synchronization procedures used when there are two masters present on the bus.

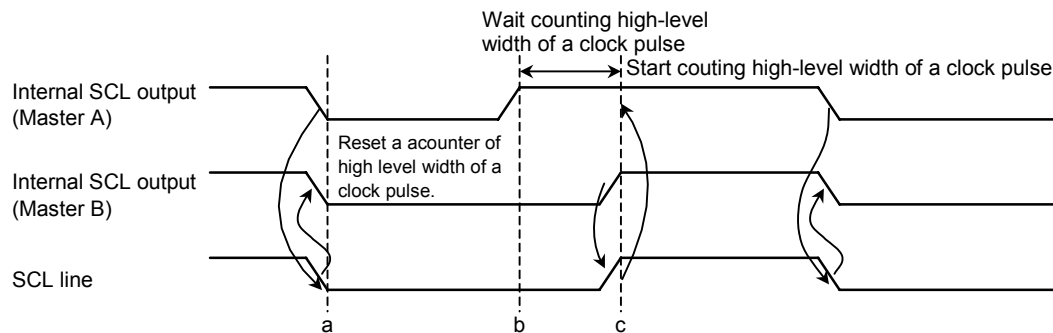


Figure 3.10.8 Clock Synchronization

When master A pulls the internal SCL output low at point a, the bus's SCL line goes low. After detecting this, master B resets a counter of high level width of an own clock pulse and sets the internal SCL output low.

Master A finishes counting low-level width of an own clock pulse at point b and sets the internal SCL output high. Since master B is holding the bus's SCL line low, master A waits for counting high-level width of an own clock pulse. After master B has finished counting low-level width of an own clock pulse at point c and master A detects the SCL line of the bus at the high level, and starts counting high level of an own clock pulse.

The clock pulse on the bus is determined by the master device with the shortest high-level width and the master device with the longest low-level width from among those master devices connected to the bus.

(4) Slave address and address recognition mode specification

When the TMP91CW18A is to be used as a slave device, set the slave address <SA6:0> and <ALS> in I2C0AR0. Clear <ALS> to 0 for the address recognition mode.

(5) Master/slave selection

To operate the TMP91CW18A as a master device set SBI0CR20<MST> to 1. To operate it as a slave device clear SBI0CR20<MST> to 0. SBI0CR20<MST> is cleared to 0 in hardware when a stop condition is detected on the bus or arbitration is lost.

(6) Transmitter/receiver selection

To operate the TMP91CW18A as a transmitter set SBI0CR20<TRX> to 1. To operate it as a receiver clear SBI0CR20<TRX> to 0. When data with an addressing format is transferred in slave mode, when a slave address with the same value that an I2C0AR0 or a GENERAL CALL is received (All 8-bit data are 0 after a start condition), SBI0CR20<TRX> is set to 1 in hardware if the direction bit (R/ \bar{W}) sent from the master device is 1 and is cleared to 0 in hardware if the bit is 0. In master mode, when an acknowledge signal is returned from the slave device, SBI0CR20<TRX> is cleared to 0 in hardware if the value of the transmitted direction bit is 1, and is set to 1 in hardware if the value of the bit is 0. If an acknowledge signal is not returned, the current state is maintained.

SBI0CR20<TRX> is cleared to 0 in hardware when a stop condition is detected on the I²C bus or when arbitration is lost.

(7) Start/stop condition generation

When SBISR0<BB> = 0, slave address and direction bit which are set in SBI0DBR0 are output on the bus after generating a start condition by writing 1 to the SBI0CR20<MST, TRX, BB and PIN>. It is necessary to set transmitted data to the data buffer register (SBI0DBR0) and set 1 to <ACK> beforehand.

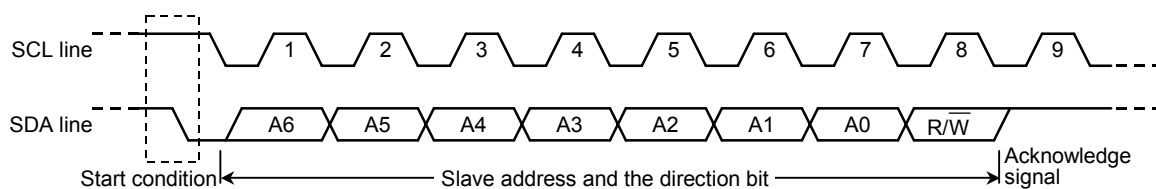


Figure 3.10.9 Start Condition Generation and Slave Address Generation

When SBISR0<BB> = 1, the sequence for generating a stop condition can be initiated by writing 1s to SBISR0<MST, TRX and PIN> and writing 0 to SBISR0<BB>. Do not modify the contents of SBISR0<MST, TRX, BB and PIN> until a stop condition has been generated on the bus.

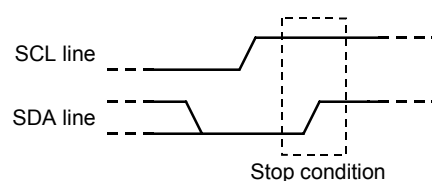


Figure 3.10.10 Stop Condition Generation

The state of the bus can be ascertained by reading the contents of SBISR0<BB>. SBISR0<BB> will be set to 1 if a start condition has been detected on the bus, and will be cleared to 0 if a stop condition has been detected.

Refer to chapter 3.10.6 (4) “Stop condition generation”, where the restriction for the stop condition in the master mode is explained.

(8) Interrupt service requests and interrupt cancellation

When a serial bus interface interrupt request INTSBI0 is generated, SBI0CR20<PIN> is cleared to 0. The SCL line is pulled low while SBI0CR20<PIN> = 0.

SBI0CR20<PIN> is cleared to 0 when a single word of data is transmitted or received. Either writing data to or reading data from SBI0DBR0 sets SBI0CR20<PIN> to 1.

The time from SBI0CR20<PIN> being set to 1 until the release of the SCL line is t_{LOW}.

In address recognition mode (e.g., when SBI0CR20<ALS> = 0), SBI0CR20<PIN> is cleared to 0 when the slave address matches the value set in I2C0AR0 or when a GENERAL CALL is received (All 8-bit data are 0 after a start condition). Although SBI0CR20<PIN> can be set to 1 by a program, writing 0 to SBI0CR20<PIN> does not clear it to 0.

(9) Serial bus interface operation mode selection

SBI0CR20<SBIM1:0> is used to specify the serial bus interface operation mode. Set SBI0CR20<SBIM1:0> to 10 when the device is to be used in I²C bus mode after confirming pin condition of serial bus interface to “H”.

Switch a mode to port after confirming a bus is free.

(10) Arbitration lost detection monitor

Since more than one master device can exist simultaneously on the bus in I²C bus mode, a bus arbitration procedure has been implemented in order to guarantee the integrity of transferred data.

Data on the SDA line is used for I²C bus arbitration.

The following example illustrates the bus arbitration procedure when there are two master devices on the bus. Master A and master B output the same data until point a. After master A outputs L and master B, H, the SDA line of the bus is wire-AND and the SDA line is pulled low by master A. When the bus's SCL line is pulled up at point b, the slave device reads the data on the SDA line, that is, data in master A. Data transmitted from master B becomes invalid. The master B state is known as arbitration lost. A master B device which loses arbitration releases the internal SDA output in order not to affect data transmitted from other masters with arbitration. When more than one master sends the same data at the first word, arbitration occurs continuously after the second word.

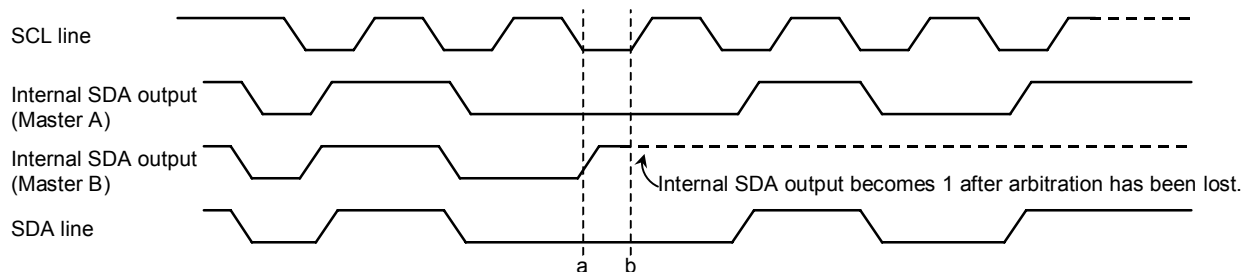


Figure 3.10.11 Arbitration Lost

The TMP91CW18A compares the levels on the bus's SDA line with those of the internal SDA output on the rising edge of the SCL line. If the levels do not match, arbitration is lost and SBISR0<AL> is set to 1.

When SBISR0<AL> is set to 1, SBISR0<MST, TRX> are cleared to 00 and the mode is switched to slave receiver mode. Thus, clock output is stopped in data transfer after setting <AL> = 1.

SBISR0<AL> is cleared to 0 when data is written to or read from SBI0DBR0 or when data is written to SBI0CR20.

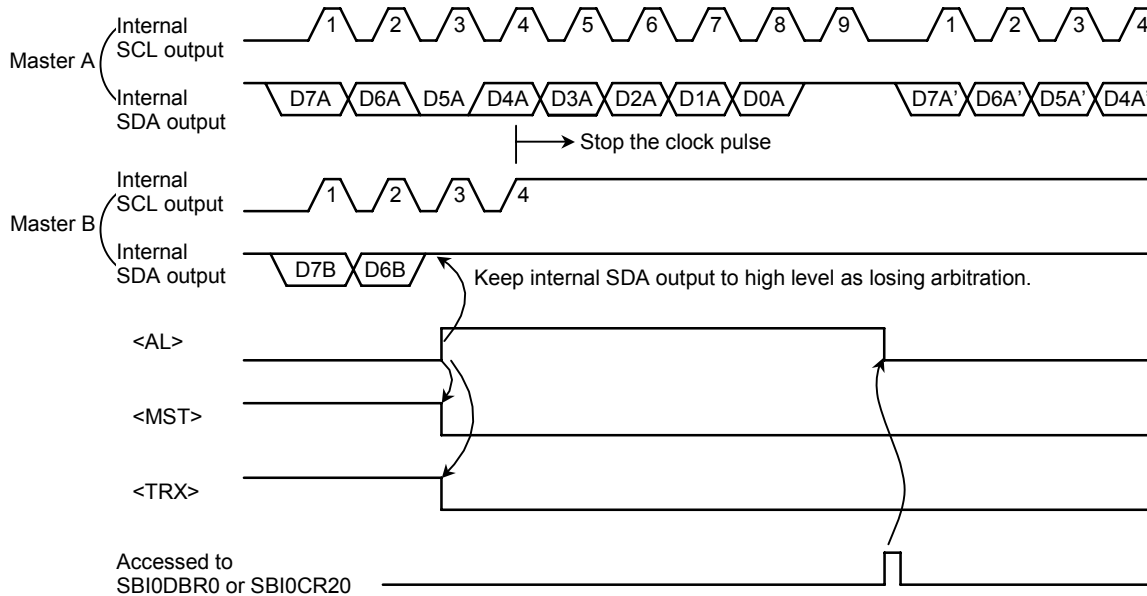


Figure 3.10.12 Example of when TMP91CW18A is a Master Device B (D7A = D7B, D6A = D6B)

(11) Slave address match detection monitor

SBISR0<AAS> is set to 1 in slave mode, in address recognition mode (e.g., when I2C0AR0<ALS> = 0), when a GENERAL CALL is received or when a slave address matches the value set in I2C0AR0. When I2C0AR0<ALS> = 1, SBISR0<AAS> is set to 1 after the first word of data has been received. SBISR0<AAS> is cleared to 0 when data is written to or read from the data buffer register SBI0DBR0.

(12) GENERAL CALL detection monitor

SBISR0<AD0> is set to 1 in slave mode when a GENERAL CALL is received (All 8-bit received data is 0 after a start condition). SBISR0<AD0> is cleared to 0 when a start condition or stop condition is detected on the bus.

(13) Last received bit monitor

The value on the SDA line detected on the rising edge of the SCL line is stored in SBISR0<LRB>. In acknowledge mode, immediately after an INTSBI0 interrupt request has been generated, an acknowledge signal is read by reading the contents of SBISR0<LRB>.

(14) Software reset function

The software reset function is used to initialize the SBI circuit, when SBI is locked by external noises, etc.

An internal reset signal pulse can be generated by setting SBI0CR20<SWRST1:0> to 10 and 01. This initializes the SBI circuit internally. All command except SBI0CR20<SBIM1:0> registers and status registers are initialized as well.

SBI0CR10<SWRMON> is automatically set to 1 after the SBI circuit has been initialized.

(15) Serial bus interface data buffer register (SBI0DBR0)

Received data can be read by reading SBI0DBR0 and transferred data can be written by writing to SBI0DBR0.

When the start condition has been generated in master mode, the slave address and the direction bit are set in this register.

(16) I²C bus address register (I2C0AR0)

I2C0AR0<SA6:0> is used to set the slave address when the TMP91CW18A functions as a slave device.

If the slave address output from the master device is recognized as matching the TMP91CW18A's slave address, I2C0AR0<ALS> is cleared to 0. The data format is the addressing format. When the slave address output from the master does not match the TMP91CW18A's slave address, I2C0AR0<ALS> is set to 1 and the data format is the free data format.

(17) Baud rate register (SBI0BR10)

Write 1 to SBI0BR10<P4EN> before operation commences.

(18) Setting register for IDLE2 mode operation (SBI0BR00)

The setting of SBI0BR00<I2SBI0> determines whether the device is operating or is stopped in IDLE2 mode. Hence, <I2SBI0> must be set before a HALT instruction is executed.

3.10.6 Data Transfer in I²C Bus Mode

(1) Device initialization

Set SBI0BR10<P4EN> and SBI0CR10<ACK, SCK2:0> to 1. Set SBI0BR10 to 1 and clear bits 7, 6, 5 and 3 of SBI0CR10 to 0.

Set a slave address in I2C0AR0<SA6:0> and I2C0AR0<ALS> (I2C0AR0<ALS> = 0 when an addressing format).

For specifying the default setting to a slave receiver mode, clear 0 to the <MST, TRX and BB> and set 1 to the <PIN>, 10 to the <SBIM1:0>.

(2) Start condition generation and slave address generation

1. Master mode

In master mode the start condition and the slave address are generated as follows. Check a bus free status (when <BB> = 0).

Set SBI0CR10<ACK> to 1 (Acknowledge mode) and specify a slave address and a direction bit to be transmitted to the SBI0DBR0.

If SBI0CR20<BB> = 0, the start condition is generated by writing 1111 to SBI0CR20<MST, TRX, BB and PIN>. Subsequently to the start condition, 9 clocks are output from the SCL pin. While 8 clocks are output, the slave address and the direction bit which are set to the SBI0DBR0. On the 9th clock pulse the SDA line is released and the acknowledge signal is received from the slave device.

An INTSBI0 interrupt request occurs on the falling edge of the 9th clock pulse. SBI0CR20<PIN> is cleared to 0. In master mode the SCL pin is pulled low while SBI0CR20<PIN> is 0. When an interrupt request occurs, the value of SBI0CR20<TRX> is changed according to the direction bit setting only if the slave device returns an acknowledge signal.

2. Slave mode

In slave mode the start condition and the slave address are received.

After the start condition has been received from the master device, while 8 clocks are output from the SCL pin, the slave address and the direction bit which are output from the master device are received.

When a GENERAL CALL or an address matching the slave address set in I2C0AR0 is received, the SDA line is pulled down low on the 9th clock pulse and an acknowledge signal is output.

An INTSBI0 interrupt request occurs on the falling edge of the 9th clock pulse. SBI0CR20<PIN> is cleared to 0. In slave mode the SCL line is pulled low while SBI0CR20<PIN> = 0. When an interrupt request occurs, the value of SBI0CR20<TRX> is changed according to the direction bit setting only if the slave device returns an acknowledge signal.

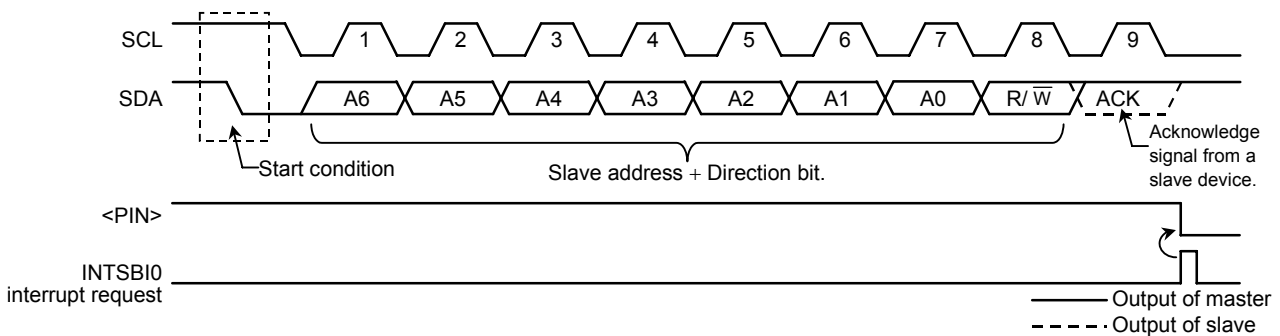


Figure 3.10.13 Start Condition Generation and Slave Address Transfer

(3) Single-word data transfer

Check the <MST> setting using an INTSBI0 interrupt process after the transfer of each word of data is completed and determine whether the device is in master mode or slave mode.

1. If <MST> = 1 (Master mode)

Check the <TRX> setting and determine whether the device is in transmitter mode or receiver mode.

If <TRX> = 1 (Transmitter mode)

Check the <LRB> setting. If <LRB> = 1, there is no receiver requesting data. Implement the process for generating a stop condition (See section 3.10.6 (4)) and terminate data transfer.

If <LRB> = 0, the receiver is requesting new data. When the next transmitted data is 8 bits, write the transmitted data to SBI0DBR0. When the next transmitted data is other than 8 bits, set <BC2:0> to 1, set <ACK> to 1 and write the transmitted data to SBI0DBR0. After the data has been written, <PIN> is set to 1, a serial clock pulse is generated to trigger transfer of the next word of data via the SCL pin, and the word is transmitted. After the data has been transmitted, an INTSBI0 interrupt request is generated. <PIN> is cleared to 0 and the SCL line is pulled low. If the length of the data to be transferred is greater than one word, repeat the latter steps of the procedure, starting from the check of the <LRB> setting.

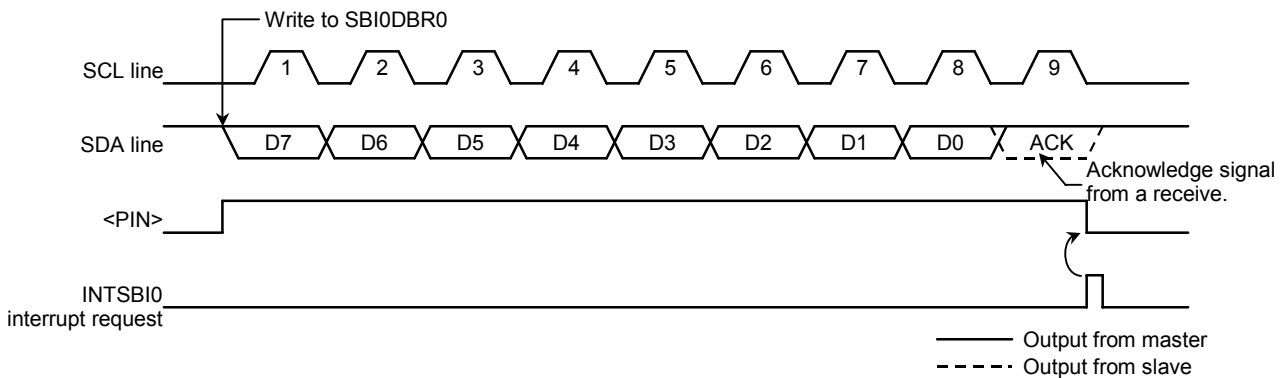


Figure 3.10.14 Example in which <BC2:0> = 000 and <ACK> = 1 in Transmitter Mode

If <TRX> = 0 (Receiver mode)

When the next transmitted data is other than 8 bits, set <BC2:0> again. Set <ACK> to 1 and read the received data from SBI0DBR0 so as to release the SCL line (the value of data which is read immediately after a slave address is sent is undefined). After the data has been read, <PIN> is set to 1.

Serial clock pulse for transferring new 1 word of data is defined SCL and outputs “L” level from SDA pin with acknowledge timing.

An INTSBI0 interrupt request is then generated and <PIN> is cleared to 0. The TMP91CW18A then pulls the SCL pin low. From then on the TMP91CW18A outputs a clock pulse, so as to transfer a data word, and an acknowledge signal each time received data is read from SBI0DBR0.

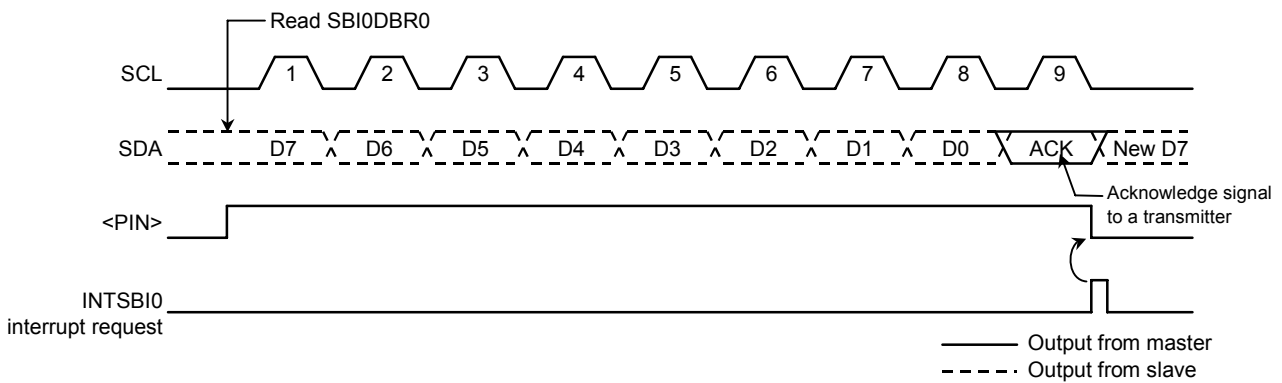


Figure 3.10.15 Example of when <BC2:0> = “000”, <ACK> = “1” in Receiver Mode

In order to terminate the transmission of data to a transmitter, clear <ACK> to 0 before reading data which is 1 word before the last data to be received. The last data word does not generate a clock pulse as the acknowledge signal. After the data has been transmitted and an interrupt request has been generated, set <BC2:0> to 001 and read the data. The TMP91CW18A generates a clock pulse for a 1-bit data transfer. Since the master device is a receiver, the SDA line on the bus remains high. The transmitter interprets the high signal as an ACK signal. The receiver indicates to the transmitter that data transfer is complete.

After the one data bit has been received and an interrupt request been generated, the TMP91CW18A generates a stop condition (See section 3.10.6 (4)) and terminates data transfer.

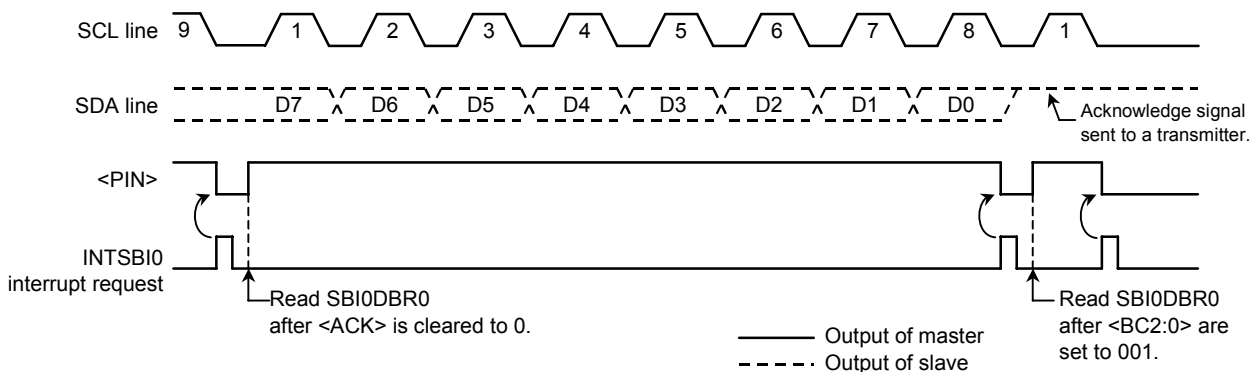


Figure 3.10.16 Termination of Data Transfer in Master Receiver Mode

2. If <MST> = 0 (Slave mode)

In slave mode the TMP91CW18A operates either in normal slave mode or in slave mode after losing arbitration.

In slave mode an INTSBI0 interrupt request is generated when the TMP91CW18A receives a slave address or a GENERAL CALL from the master device, or when a GENERAL CALL is received and data transfer is complete or when a matching slave address is received. The TMP91CW18A will enter slave mode from master mode if it loses arbitration. An INTSBI0 interrupt request is generated when a word data transfer terminates after arbitration has been lost. When an INTSBI0 interrupt request is generated, <PIN> is cleared to 0 and the SCL pin is pulled low. Either reading data to or writing data from SBI0DBR0, or setting <PIN> to 1 will release the SCL pin after tLOW.

Check the SBISR0<AL>, <TRX>, <AAS>, <AD0> and implements processes according to conditions listed in the next table.

Table 3.10.1 Operation in the Slave Mode

<TRX>	<AL>	<AAS>	<AD0>	Conditions	Process
1	1	1	0	The TMP91CW18A loses arbitration when transmitting a slave address and receives a slave address for which the value of the direction bit sent from another master is 1.	Set the number of bits a word in <BC2:0> and write the transmitted data to SBI0DBR0.
		0	0	In slave receiver mode the TMP91CW18A receives a slave address for which the value of the direction bit sent from the master is 1.	
	0	0	In slave transmitter mode a single word of is transmitted. Set <BC2:0> to the number of bits in a word.	Check the <LRB> setting. If <LRB> is set to 1, set <PIN> to 1 since the receiver win no request the data which follows. Then, clear <TRX> to 0 to release the bus. If <LRB> is cleared to 0 of and write the transmitted data to SBI0DBR0 since the receiver requests next data.	
0	1	1	1/0	The TMP91CW18A loses arbitration when transmitting a slave address and receives a slave address or GENERAL CALL for which the value of the direction bit sent from another master is 0.	Read the SBI0DBR0 for setting the <PIN> to 1 (Reading dummy data) or set the <PIN> to 1.
		0	0	The TMP91CW18A loses arbitration when transmitting a slave address or data and terminates word data transfer.	
	0	1	1/0	In slave receiver mode the TMP91CW18A receives a slave address or GENERAL CALL for which the value of the direction bit sent from the master is 0.	Set <BC2:0> to the number of bits in a word and read the received data from SBI0DBR0.
		0	1/0	In slave receiver mode the TMP91CW18A terminates receiving word data.	

(4) Stop condition generation

When SBISR0<BB> = 1, the sequence for generating a stop condition start by writing “1” to SBI0CR20<MST, TRX, PIN> and “0” to SBI0CR20<BB>. Do not modify the contents of SBI0CR20<MST, TRX, PIN, BB> until a stop condition has been generated on the bus. When the bus’s SCL line has been pulled low by another device, the TMP91CW18A generates a stop condition when the other device has released the SCL line and SDA pin rising.

When SBI0CR20<MST, TRX, PIN> are written 1 and <BB> is written 0 (generate stop condition in master mode), <BB> changes to 0 by internal SCL changes to 1, without waiting stop condition. To check whether SCL and SDA pin are 1 by sensing their ports is needed to detect bus free condition.

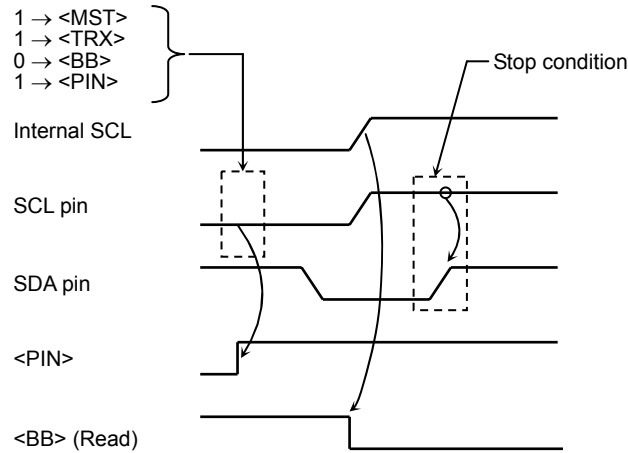


Figure 3.10.17 Stop Condition Generation (Single master)

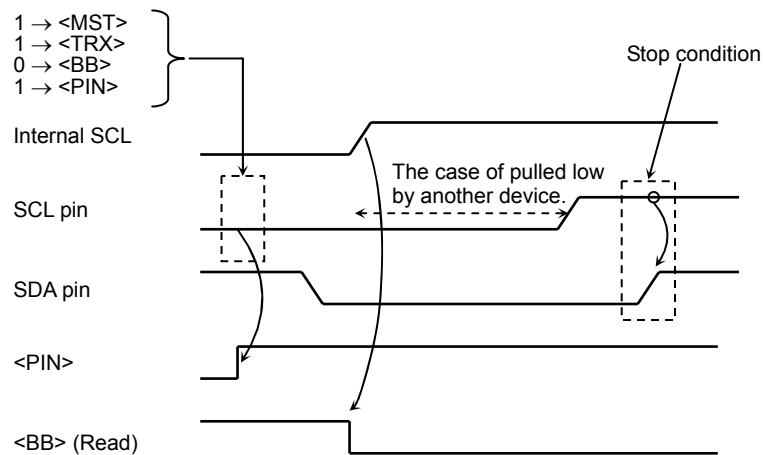


Figure 3.10.18 Stop Condition Generation (Multi master)

(5) Restart

Restart is used during data transfer between a master device and a slave device to change the data transfer direction.

The following description explains how to restart when the TMP91CW18A is in master mode.

Clear SBI0CR20<MST, TRX, BB> to 0 and set SBI0CR20<PIN> to 1 to release the bus. The SDA line remains high and the SCL pin is released. Since a stop condition has not been generated on the bus, other devices assume the bus to be in busy state.

And confirm SCL pin, that SCL pin is released and become bus-free state by SBISR0<BB> = 0 or signal level 1 of SCL pin in port mode. Check the <LRB> until it becomes 1 to check that the SCL line on a bus is not pulled down to the low level by other devices. After confirming that the bus remains in a free state, generate a start condition using the procedure described in (2).

In order to satisfy the setup time requirements when restarting, take at least 4.7 μs of waiting time by software from the time of restarting to confirm that the bus is free until the time to generate the start condition.

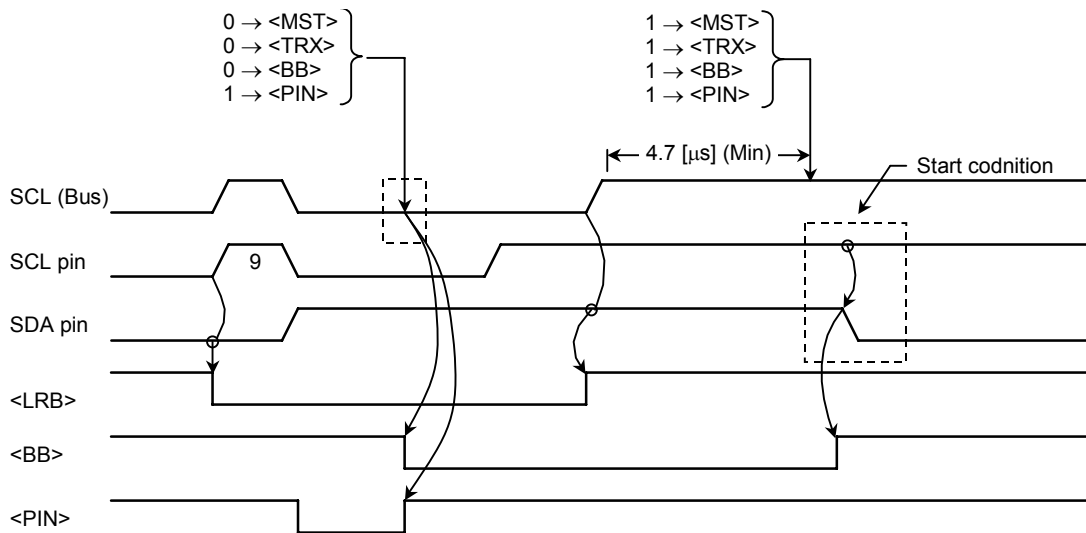
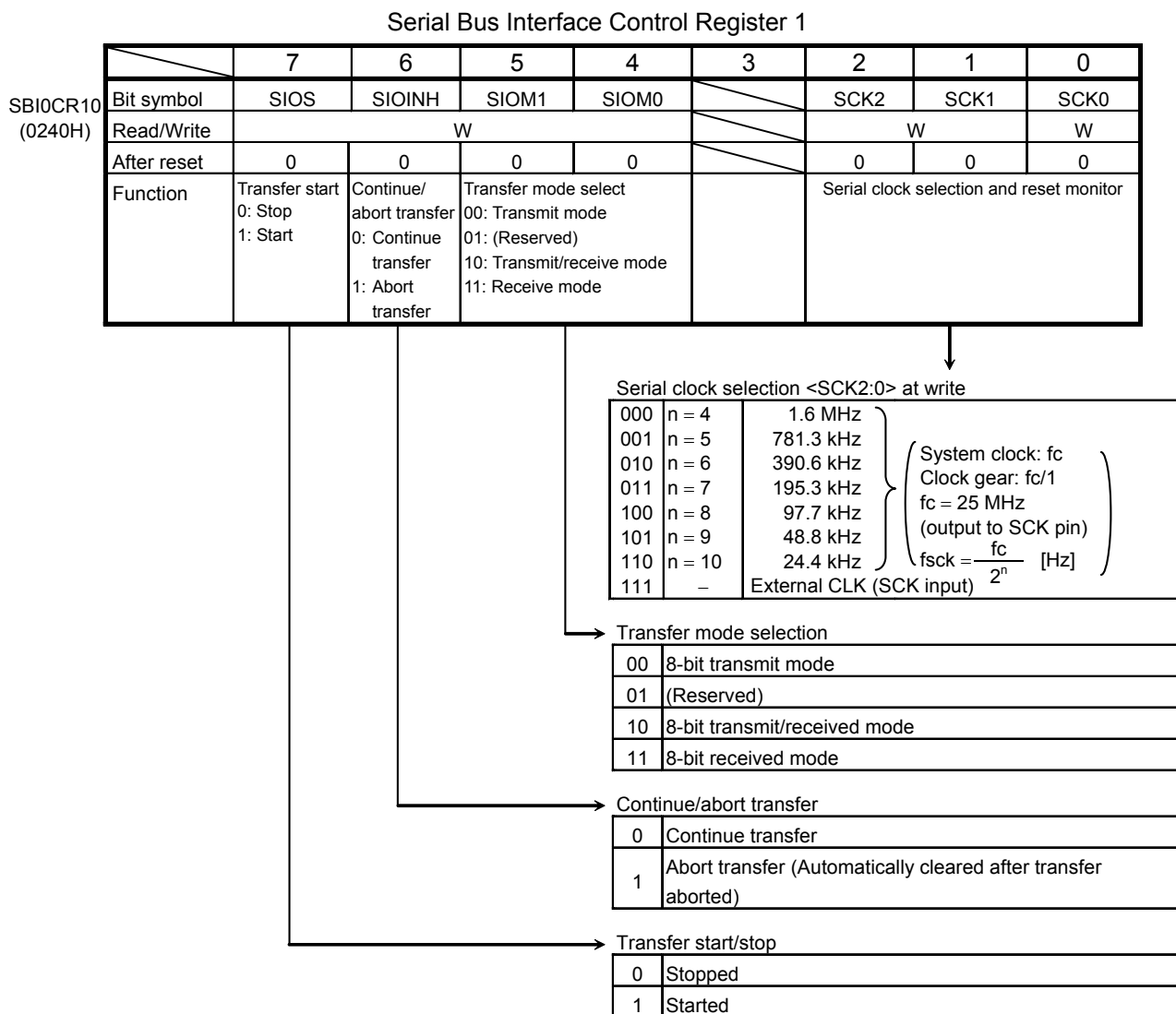


Figure 3.10.19 Timing Diagram for TMP91CW18A Restart

3.10.7 Clocked-synchronous 8-Bit SIO Mode Control

The following registers are used to control and monitor the operation status when the serial bus interface (SBI) is being operated in clocked-synchronous 8-bit SIO mode.



Note 1: Read-modify-write instruction is prohibited for SBI0CR10.

Note 2: Set the transfer mode and the serial clock after programming <SIOS> to 0 and <SIOINH> to 1.

Serial Bus Interface Data Buffer Register

		7	6	5	4	3	2	1	0
SBI0DBR0 (0241H)	Bit symbol	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	Read/Write	R (Receiver)/W(Transfer)							
	After reset	Undefined							

Note: Read-modify-write instruction is prohibited for SBI0DBR0.

Figure 3.10.20 Register for the SIO Mode

Serial Bus Interface Control Register 2

	7	6	5	4	3	2	1	0
SBI0CR20 (0243H)	Bit symbol				SBIM1	SBIM0	–	–
	Read/Write				W		W	W
	After reset				0	0	0	0
	Function				Serial bus interface operation mode selection 00: Port mode 01: SIO mode 10: I ² C bus mode 11: (Reserved)		(Note 3)	(Note 3)

Serial bus interface operation mode selection

00	Port mode (Serial bus interface output disabled)
01	Clocked-synchronous 8-bit SIO mode
10	I ² C bus mode
11	(Reserved)

Note 1: Read-modify-write instruction is prohibited for SBI0CR20.

Note 2: Set the SBI0CR10<BC2:0> to “000” before switching to a clocked-synchronous 8-bit SIO mode.

Note 3: Please always write SBICR2<1:0> to 00.

Serial Bus Interface Status Register

	7	6	5	4	3	2	1	0
SBISR0 (0243H)	Bit symbol				SIOF	SEF		
	Read/Write				R			
	After reset				0	0		
	Function				Serial transfer operation status monitor	Shift operation status monitor		

Shift operation status monitor

0	Shift operation terminated
1	Shift operation in progress

Serial transfer operating status monitor

0	Transfer terminated
1	Transfer in progress

Figure 3.10.21 Registers for the SIO Mode (1/2)

Serial Bus Interface Baud Rate Register 0

	7	6	5	4	3	2	1	0
SBI0BR00 (0244H)								
Bit symbol	-	I2SBI0						
Read/Write	W	R/W						
After reset	0	0						
Function	Always write 0	IDLE2 0: Stop 1: Run						

Read-modify-write instructions are prohibited.

Operation in IDLE2 mode

0	Stop
1	Operation

Serial Bus Interface Baud Rate Register 1

	7	6	5	4	3	2	1	0
SBI0BR10 (0245H)								
Bit symbol	P4EN	-						
Read/Write	W	W						
After reset	0	0						
Function	Internal clock 0: Stop 1: Operate	Always write 0						

Read-modify-write instructions are prohibited.

Baud rate clock control

0	Stop
1	Operation

Figure 3.10.22 Registers for the SIO Mode (2/2)

(1) Serial clock

1. Clock source

SBI0CR10<SCK2:0> is used to select the following functions.

Internal clock

In internal clock mode one of seven frequencies can be selected. The serial clock signal is output to the outside on the SCK pin. When the device is writing (in transmit mode) or reading (in receive mode), data cannot follow the serial clock rate, so an automatic wait function is executed which automatically stops the serial clock and holds the next shift operation until reading or writing has been completed.

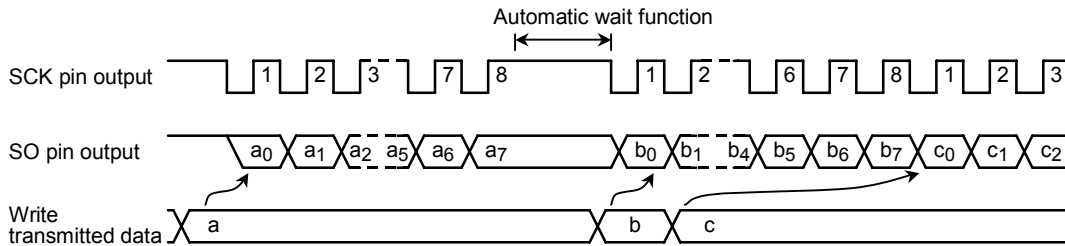


Figure 3.10.23 Automatic Wait Function

External clock (<SCK2:0> = 111)

An external clock input via the SCK pin is used as the serial clock. In order to ensure the integrity of shift operations, both the high and low level serial clock pulse widths shown below must be maintained. The maximum data transfer frequency is 1.6 MHz (when $f_c = 25$ MHz).

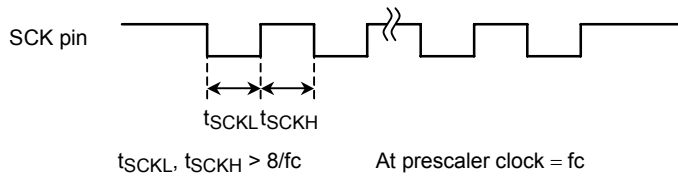


Figure 3.10.24 Maximum Data Transfer Frequency when External Clock Input Used

2. Shift edge

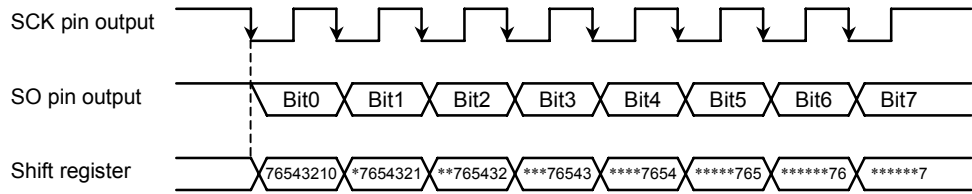
Data is transmitted on the leading edge of the clock and received on the trailing edge.

Leading edge shift

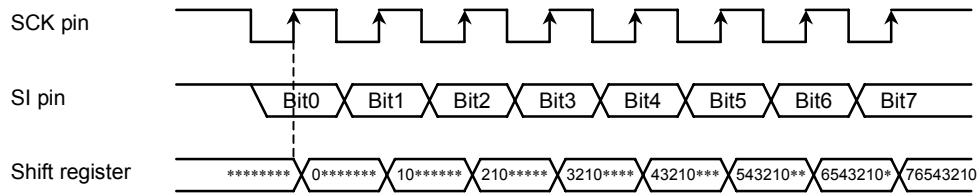
Data is shifted on the leading edge of the serial clock (on the falling edge of the SCK pin input/output).

Trailing edge shift

Data is shifted on the trailing edge of the serial clock (on the rising edge of the SCK pin input/output).



(a) Leading edge



*: Don't care

(b) Trailing edge

Figure 3.10.25 Shift Edge

(2) Transfer modes

SBI0CR10<SIOM1:0> is used to select a transmit, receive or transmit/receive mode.

1. 8-bit transmit mode

Set a control register to a transmit mode and write transmission data to SBI0DBR0.

After the transmit data has been written, set SBI0CR10<SIOS> to 1 to start data transfer. The transmitted data is transferred from SBI0DBR0 to the shift register and output, starting with the least significant bit (LSB), via the SO pin and synchronized with the serial clock. When the transmission data has been transferred to the shift register, the SBI0DBR0 becomes empty. An INTSBI0 (Buffer empty) interrupt request is generated to request new data.

When the internal clock is used, the serial clock will stop and the automatic wait function will be initiated if new data is not loaded into the data buffer register after the specified 8-bit data has been transmitted. When new transmission data is written, the automatic-wait function is canceled.

When an external clock is used, data should be written to SBI0DBR0 before new data is shifted. The transfer speed is determined by the maximum delay time between the time when an interrupt request is generated and a data is written to SBI0DBR0 by the interrupt service program.

When the transmission is started, after the SBISR0<SIOF> goes 1 output from the SO pin holds final bit of the last data until falling edge of the SCK.

Data transmission ends when <SIOS> is cleared to 0 by the buffer empty interrupt service program or when <SIOINH> is set to 1. When <SIOS> is cleared to 0, the transmitted mode ends when all data is output. In order to confirm whether data is being transmitted properly by the program, set SBISR0<SIOF> (Bit3 of SBISR0) to be sensed. SBISR0<SIOF> is cleared to 0 when transmission has been completed. When <SIOINH> is set to 1, data transmission stops. SBISR0<SIOF> is then cleared to 0.

When an external clock is used, it is also necessary to clear SBISR0<SIOS> to 0 before new data is shifted; otherwise, dummy data will be transmitted and operation ends.

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

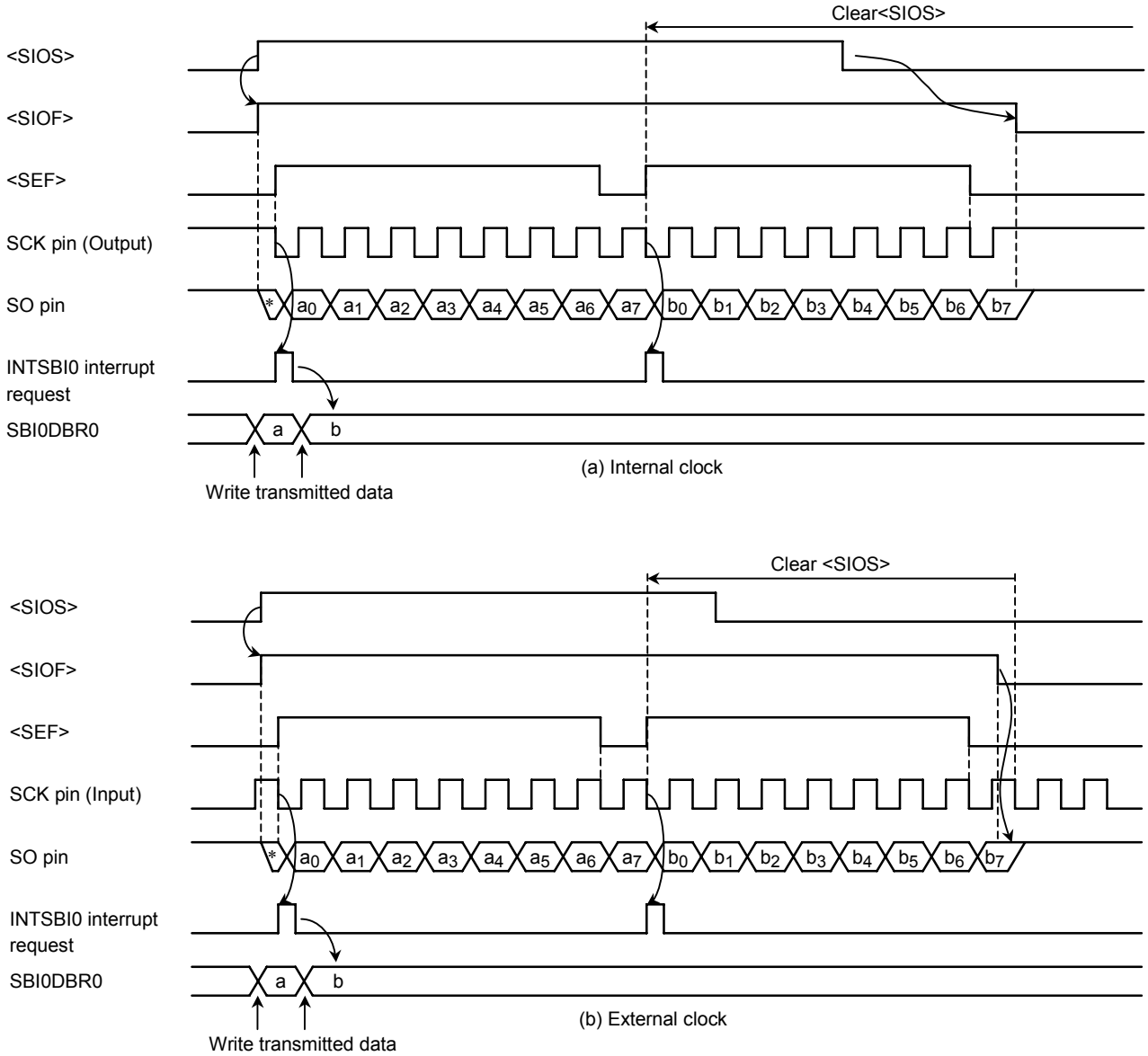


Figure 3.10.26 Transfer Mode

```

STEST1: BIT    2, (SBISR0)           ; If <SEF> = 1 then loop.
        JR     NZ, STEST1
STEST2: BIT    0, (P7)              ; If SCK = 0 then loop.
        JR     Z, STEST2
        LD    (SBI0CR10), 00000111B ; <SIOS> ← 0.
    
```

2. 8-bit receive mode

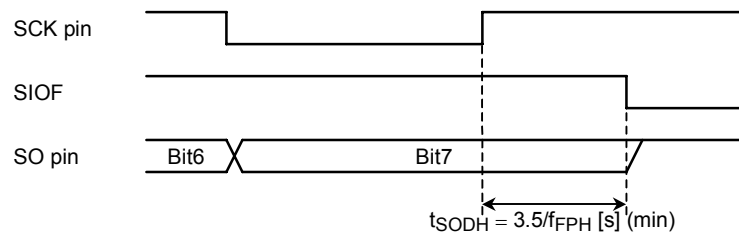


Figure 3.10.27 Transmitted Data Hold Time at End of Transmission

Set the control register to receive mode and set $SBI0CR10\langle SIOS \rangle$ to 1 for switching to receive mode. Data is received into the shift register via the SI pin and synchronized with the serial clock, starting from the least significant bit (LSB). When 8-bit data is received, the data is transferred from the shift register to $SBI0DBR0$. An $INTSBI$ (Buffer full) interrupt request is generated to request that the received data be read. The data is then read from $SBI0DBR0$ by the interrupt service program.

When an internal clock is used, the serial clock will stop and the automatic wait function will be in effect until the received data has been read from $SBI0DBR0$. When an external clock is used, since shift operation is synchronized with an external clock pulse, the received data should be read from $SBI0DBR0$ before the next serial clock pulse is input. If the received data is not read, any further data which is to be received is canceled. The maximum transfer speed when an external clock is used is determined by the delay time between the time when an interrupt request is generated and the time when the received data is read.

Receiving of data ends when $\langle SIOS \rangle$ is cleared to 0 by the buffer full interrupt service program or $\langle SIOINH \rangle$ is set to 1. If $\langle SIOS \rangle$ is cleared to 0, received data is transferred to $SBI0DBR0$ in complete blocks. The received mode ends when the transfer is complete. In order to confirm whether data is being received properly by the program, set $SBISR0\langle SIOF \rangle$ to be sensed. $\langle SIOF \rangle$ is cleared to 0 when receiving has been completed. When it is confirmed that receiving has been completed, the last data is read. When $\langle SIOINH \rangle$ is set to 1, data receiving stops. $\langle SIOF \rangle$ is cleared to 0. (The received data becomes invalid, therefore no need to read it.)

Note: When the transfer mode is changed, the contents of $SBI0DBR0$ will be lost. If the mode must be changed, conclude data receiving by clearing $\langle SIOS \rangle$ to 0, read the last data, then change the mode.

3. 8-bit transmit/receive mode

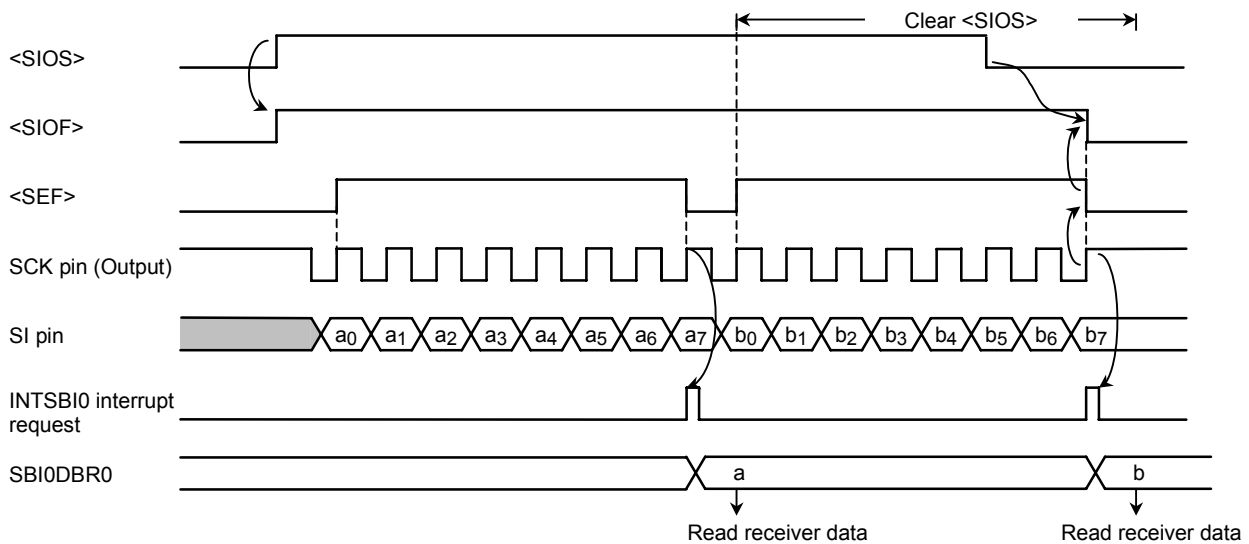


Figure 3.10.28 Receiver Mode (Example: Internal clock)

Set a control register to a transmit/receive mode and write data to SBI0DBR0. After the data has been written, set SBI0CR10<SIOS> to 1 to start transmitting/receiving. When data is transmitted, the data is output via the SO pin, starting from the least significant bit (LSB) and synchronized with the leading edge of the serial clock signal. When data is received, the data is input via the SI pin on the trailing edge of the serial clock signal. 8-bit data is transferred from the shift register to SBI0DBR0 and an INTSBI0 interrupt request is generated. The interrupt service program reads the received data from the data buffer register and writes the data which is to be transmitted. SBI0DBR0 is used for both transmitting and receiving. Transmitted data should always be written after received data has been read.

When an internal clock is used, the automatic wait function will be in effect until the received data has been read and the next data has been written.

When an external clock is used since the shift operation is synchronized with the external clock, received data is read and transmitted data is written before a new shift operation is executed. The maximum transfer speed when an external clock is used is determined by the delay time between the time when an interrupt request is generated and the time at which received data is read and transmitted data is written.

When the transmit is started after the SBISR0<SIOF> goes 1 output from the SO pin holds final bit of the last data until falling edge of the SCK.

Transmitting/receiving data ends when <SIOS> is cleared to 0 by the INTSBI0 interrupt service program or SBI0CR10<SIOINH> is set to 1. When <SIOS> is cleared to 0, received data is transferred to SBI0DBR0 in complete blocks. The transmit/receive mode ends when the transfer is complete. In order to confirm whether data is being transmitted/received properly by the program, set SBISR0 to be sensed. <SIOF> is set to 0 when transmitting/receiving has been completed. When <SIOINH> is set to 1, data transmitting/receiving stops. <SIOF> is then cleared to 0.

Note: When the transfer mode is changed, the contents of SBI0DBR0 will be lost. If the mode must be changed, conclude data transmitting/receiving by clearing <SIOS> to 0, read the last data, then change the transfer mode.

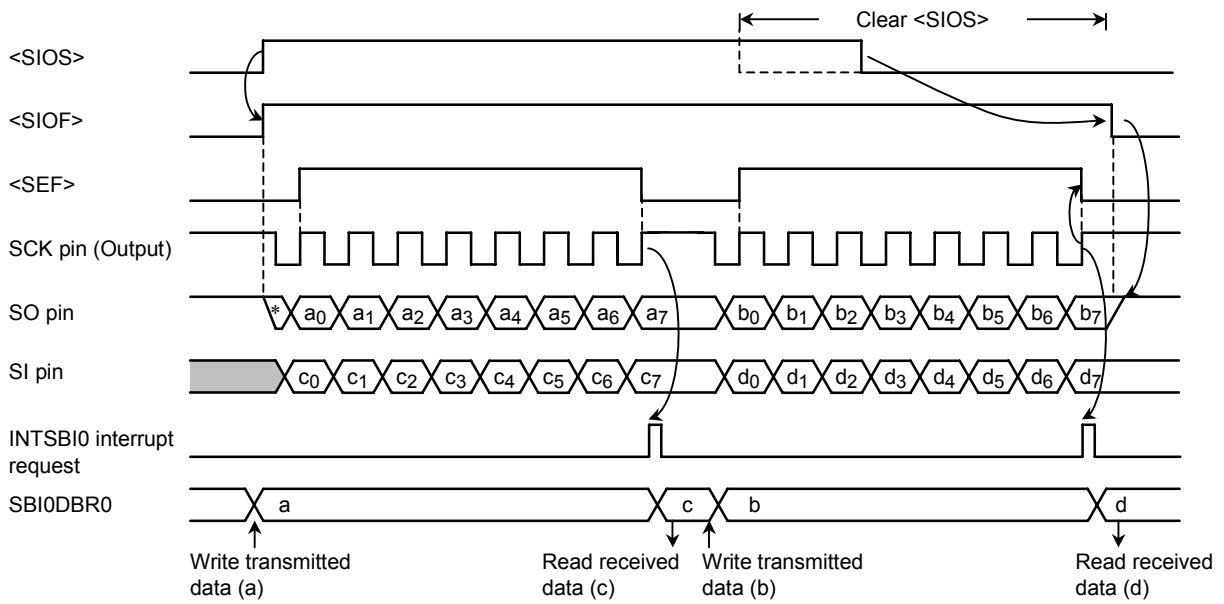


Figure 3.10.29 Transmit/Received Mode (Example: Using internal clock)

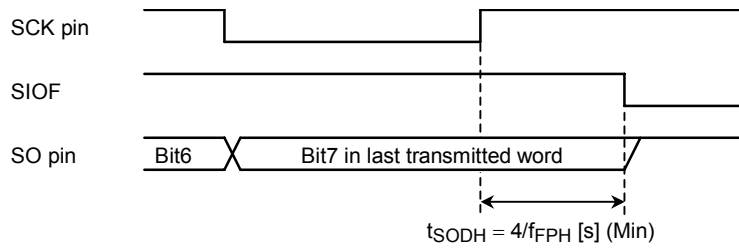


Figure 3.10.30 Transmitted Data Hold Time at End of Transmit/Receive

3.11 Serial Bus Interface 1 (I²C bus)

The serial bus interface is connected to an external device through P84 (SDA) and P85 (SCL) in the I²C bus mode. Each pin is specified as follows.

	P8CR<P85C, P84C>	P8FC<P85F, P84F>
I ² C bus mode	11	11

3.11.1 Configuration

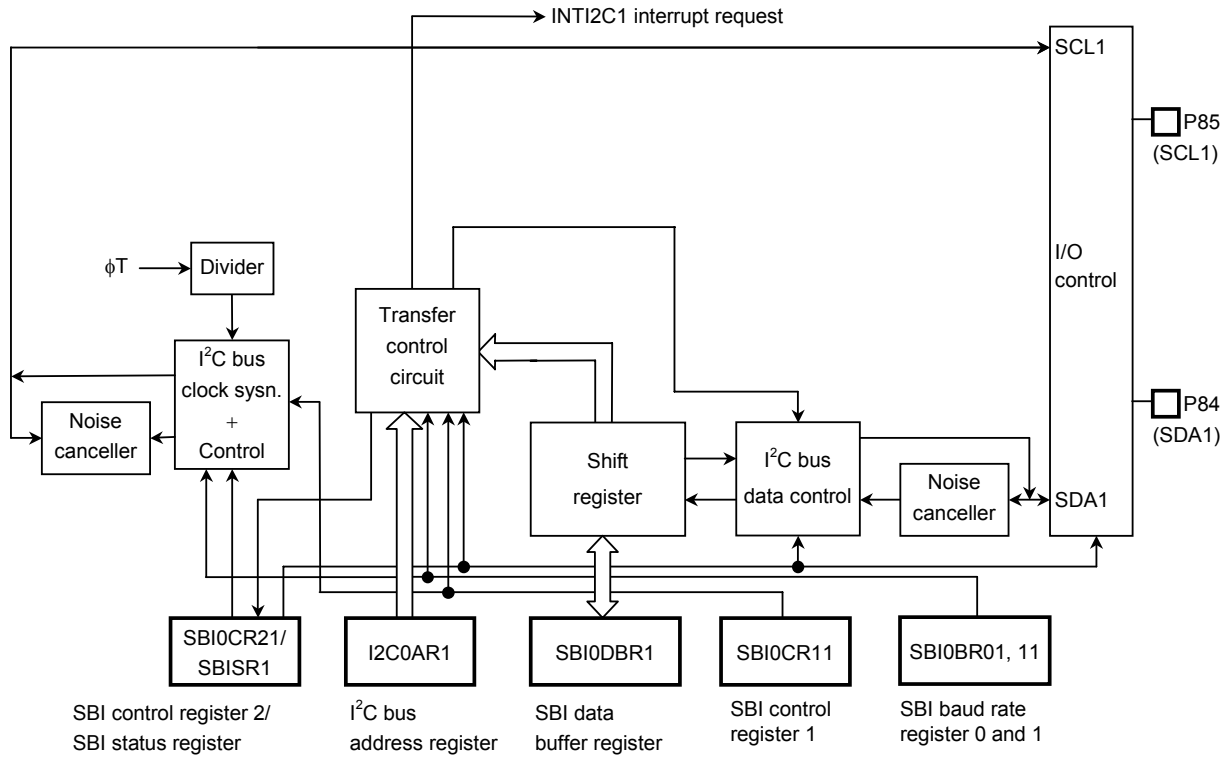


Figure 3.11.1 Serial Bus Interface (I²C bus)

3.11.2 Serial Bus Interface (SBI) Control

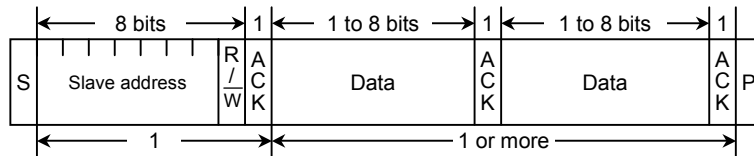
The following registers are used to control the serial bus interface and monitor the operation status.

- Serial bus interface control register 1 (SBI0CR11)
- Serial bus interface control register 2 (SBI0CR21)
- Serial bus interface data buffer register (SBI0DBR1)
- I²C bus address register (I2C0AR1)
- Serial bus interface status register (SBISR1)
- Serial bus interface baud rate register 0 (SBI0BR01)
- Serial bus interface baud rate register 1 (SBI0BR11)

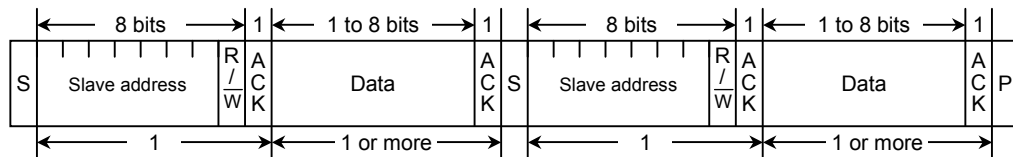
3.11.3 The Data Formats in The I²C Bus Mode

The data formats in the I²C bus mode are shown below.

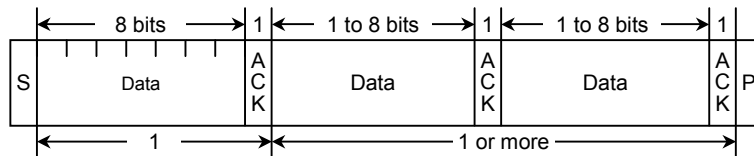
(a) Addressing format



(b) Addressing format (with restart)



(c) Free data format (Data transferred from master device to slave device)

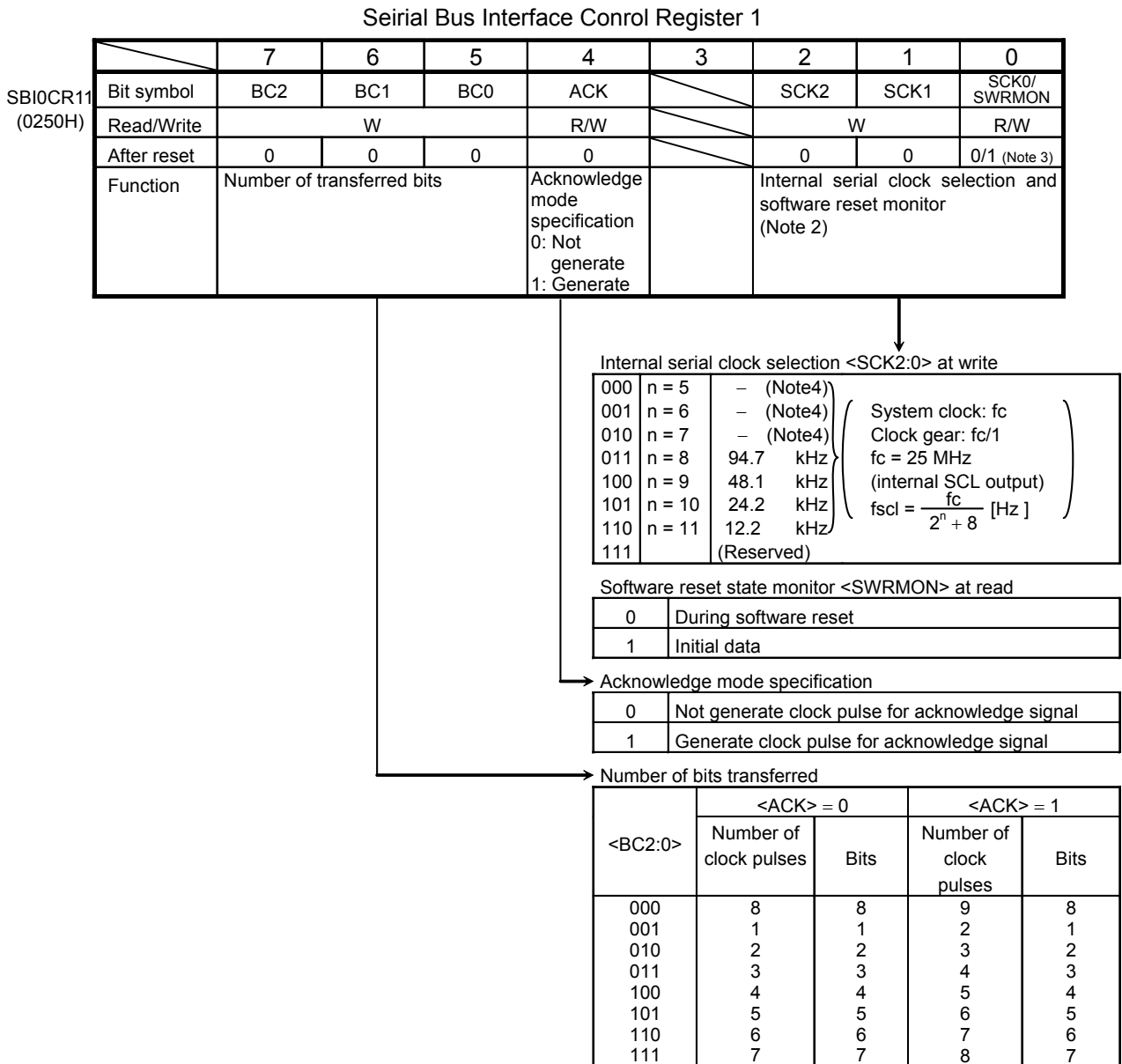


- S: Start condition
- R/W: Direction bit
- ACK: Acknowledge bit
- P: Stop condition

Figure 3.11.2 Data Format in the I²C Bus Mode

3.11.4 I²C Bus Mode Control

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



Note 1: Read-modify-write instruction is prohibited for SBI0CR11.

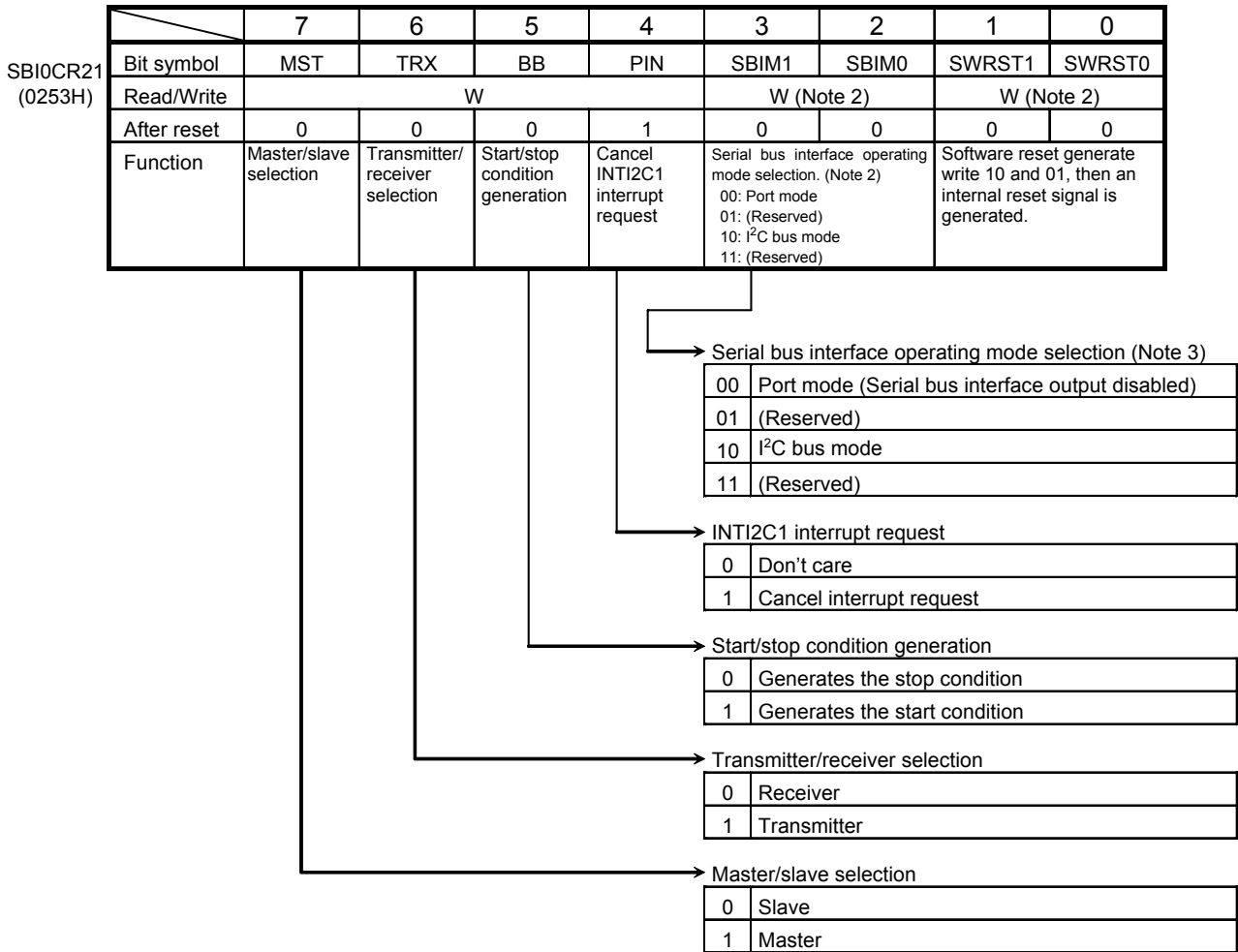
Note 2: For the frequency of the SCL line clock, See 3.11.5 (3) Serial clock.

Note 3: Initial data of SCK0 is 0, SWRMON is 1.

Note 4: This I²C bus circuit does not support Fast mode, it supports standard mode only. Although the I²C bus circuit itself allows the setting of a baud rate over 100 kbps, the compliance with the I²C specification is not guaranteed in that case.

Figure 3.11.3 Registers for the I²C Bus Mode

Serial Bus Interface Control Register 21



Note 1: Read-modify-write instruction is prohibited for SBI0CR21.

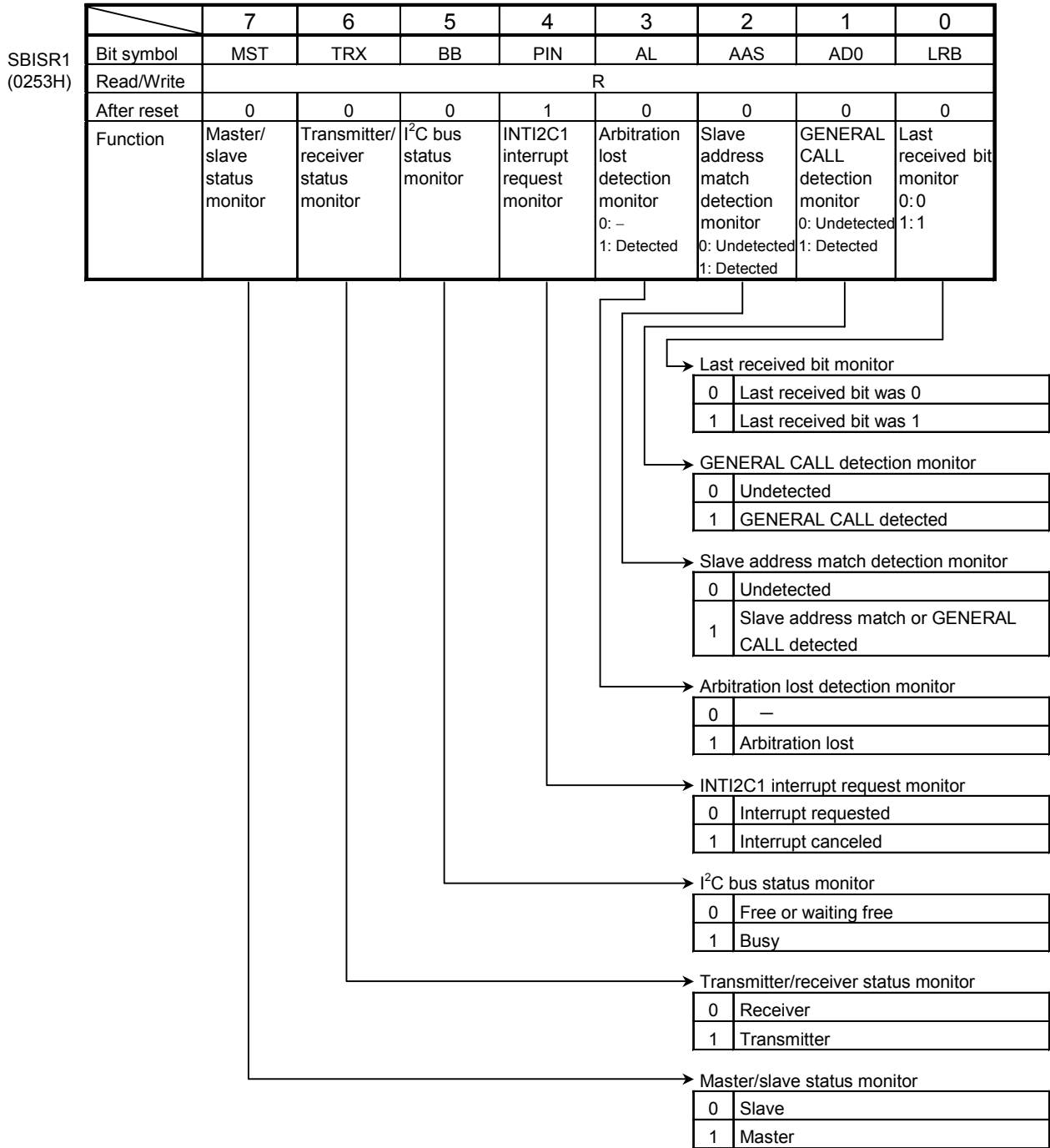
Note 2: Reading this register function as SBISR1.

Note 3: Switch a mode to port mode after confirming that the bus is free.

Switch a mode between I²C bus mode and clock synchronous 8-bit SIO mode after confirming that input signals via port are high level.

Figure 3.11.4 Registers for the I²C Bus Mode

Serial Bus Interface Status Register 1



Note 1: Read-modify-write instruction is prohibited for SBISR1.

Note 2: Writing in this register functions as SBI0CR21.

Figure 3.11.5 Registers for the I²C Bus Mode

Serial Bus Interface Baud Rate Register 0

		7	6	5	4	3	2	1	0
SBI0BR01 (0254H)	Bit symbol	–	I2SBI0						
	Read/Write	W	R/W						
Read-modify-write instructions are prohibited.	After reset	0	0						
	Function	Always write 0	IDLE2 0: Stop 1: Run						

Operation during IDLE2 mode

0	Stop
1	Operation

Serial Bus Interface Baud Rate Register 1

		7	6	5	4	3	2	1	0
SBI0BR11 (0255H)	Bit symbol	P4EN	–						
	Read/Write	W	W						
Read-modify-write instructions are prohibited.	After reset	0	0						
	Function	Internal clock 0: Stop 1: Operate	Always write 0						

Baud rate clock control

0	Stop
1	Operation

Serial Bus Interface Data Buffer Register

		7	6	5	4	3	2	1	0
SBI0DBR1 (0251H)	Bit symbol	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	Read/Write	R (Received)/W (Transfer)							
	After reset	Undefined							

Note 1: Read-modify-write instruction is prohibited for SBI0DBR1.

Note 2: When writing transmitted data, start from the MSB (Bit7). Receiving data is placed from LSB (Bit0).

Note 3: SBI0DBR1 can't be read the written data. Therefore read-modify-write instruction (e.g., "BIT" instruction) is prohibited.

Note 4: Written data in SBI0DBR1 is cleared by INTI2C1 signal.

I²C Bus Address Register

		7	6	5	4	3	2	1	0
I2C0AR1 (0252H)	Bit symbol	SA6	SA5	SA4	SA3	SA2	SA1	SA0	ALS
	Read/Write	W							
	After reset	0	0	0	0	0	0	0	0
	Function	Slave address selection for when device is operating as slave device.							
		Address recognition mode specification							

Address recognition mode specification

0	Slave address recognition
1	Non slave address recognition

Note: Read-modify-write instruction is prohibited for I2C0AR1.

Figure 3.11.6 Registers for the I²C Bus Mode

3.11.5 Control in I²C Bus Mode

(1) Specifying acknowledge mode

Set the SBI0CR11<ACK> to 1 for operation in the acknowledge mode. The TMP91CW18A generates an additional clock pulse for an acknowledge signal when operating in master mode. In the transmitter mode during the clock pulse cycle, the SDA pin is released in order to receive the acknowledge signal from the receiver. In the receiver mode during the clock pulse cycle, the SDA pin is set to the low in order to generate the acknowledge signal.

Clear the <ACK> to 0 for operation in the non-acknowledge mode, the TMP91CW18A does not generate a clock pulse for the acknowledge signal when operating in the master mode.

(2) Number of transfer bits

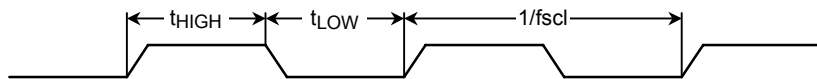
The SBI0CR11<BC2:0> is used to select a number of bits for next transmitting and receiving data.

Since SBI0CR11<BC2:0> is cleared to 000 on start-up, a slave address and direction bit transmissions are executed in 8 bits. Other than these, the <BC2:0> retains a specified value.

(3) Serial clock

1. Clock source

SBI0CR11<SCK2:0> is used to specify the maximum transfer frequency for output on the SCL pin in master mode. Set a communication baud rate that meets the I²C bus specification, such as the shortest pulse width of t_{LOW}, based on the equations shown below.



SBI0CR11<SCK2:0>	n
000	5
001	6
010	7
011	8
100	9
101	10
110	11

$$t_{LOW} = 2^{n-1} SBI$$

$$t_{HIGH} = 2^{n-1} / f_{SBI} + 8 / f_{SBI}$$

$$f_{scL} = 1 / (t_{LOW} + t_{HIGH})$$

$$= \frac{f_{SBI}}{2^n + 8}$$

Note 1: f_{SBI} is the clock f_{FPH}.
 Note 2: f_{SBI} is the clock either fc/16 or f_{FPH} which is selected SYSCR0<PRCK1:0>

Figure 3.11.7 Clock Source

2. Clock synchronization

In the I²C bus mode in order to wired-AND a bus, a master device which pulls down a clock line to low level, in the first place, invalidate a clock pulse of another master device which generates a high-level clock pulse. The master device with a high-level clock pulse needs to detect the situation and implement the following procedure.

The TMP91CW18A has a clock synchronization function which allows normal data transfer even when more than one master exists on the bus.

The following example explains the clock synchronization procedures used when there are two masters present on the bus.

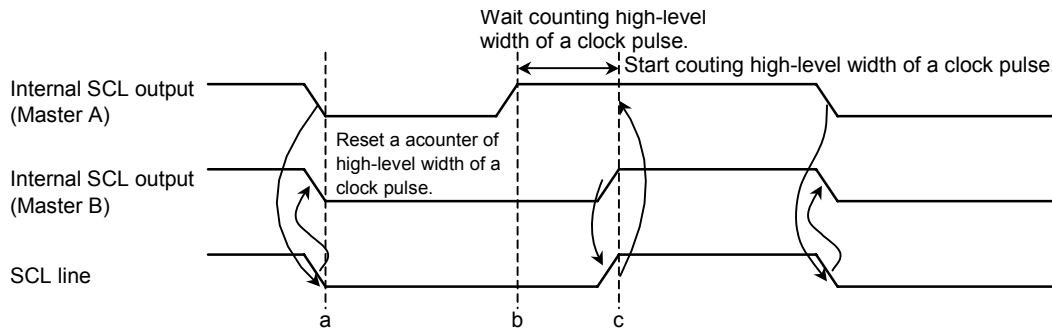


Figure 3.11.8 Clock Synchronization

When master A pulls the internal SCL output low at point a, the bus's SCL line goes low. After detecting this, master B resets a counter of high-level width of an own clock pulse and sets the internal SCL output low.

Master A finishes counting low level width of an own clock pulse at point b and sets the internal SCL output high. Since master B is holding the bus's SCL line low, master A waits for counting high level width of an own clock pulse. After master B has finished counting low level width of an own clock pulse at point c and master A detects the SCL line of the bus at the high level, and starts counting high level of an own clock pulse.

The clock pulse on the bus is determined by the master device with the shortest high level width and the master device with the longest low-level width from among those master devices connected to the bus.

(4) Slave address and address recognition mode specification

When the TMP91CW18A is to be used as a slave device, set the slave address <SA6:0> and <ALS> in I2C0AR1. Clear <ALS> to 0 for the address recognition mode.

(5) Master/Slave selection

To operate the TMP91CW18A as a master device set SBI0CR21<MST> to 1. To operate it as a slave device clear SBI0CR21<MST> to 0. SBI0CR21<MST> is cleared to 0 in hardware when a stop condition is detected on the bus or when arbitration is lost.

(6) Transmitter/receiver selection

To operate the TMP91CW18A as a transmitter set SBI0CR21<TRX> to 1. To operate it as a receiver clear SBI0CR21<TRX> to 0. When data with an addressing format is transferred in slave mode, when a slave address with the same value that an I2C0AR1 or a GENERAL CALL is received (All 8-bit data are 0 after a start condition), SBI0CR21<TRX> is set to 1 in hardware if the direction bit (R/ \bar{W}) sent from the master device is 1, and is cleared to 0 in hardware if the bit is 0. In master mode, when an acknowledge signal is returned from the slave device, SBI0CR21<TRX> is cleared to 0 in hardware if the value of the transmitted direction bit is 1, and is set to 1 in hardware if the value of the bit is 0. If an acknowledge signal is not returned, the current state is maintained.

SBI0CR21<TRX> is cleared to 0 in hardware when a stop condition is detected on the I²C bus or when arbitration is lost.

(7) Start/stop condition generation

When SBISR1<BB> = 0, slave address and direction bit which are set in SBI0DBR1 are output on the bus after generating a start condition by writing 1 to the SBI0CR21<MST, TRX, BB and PIN>. It is necessary to set transmitted data to the data buffer register (SBI0DBR1) and set 1 to <ACK> beforehand.

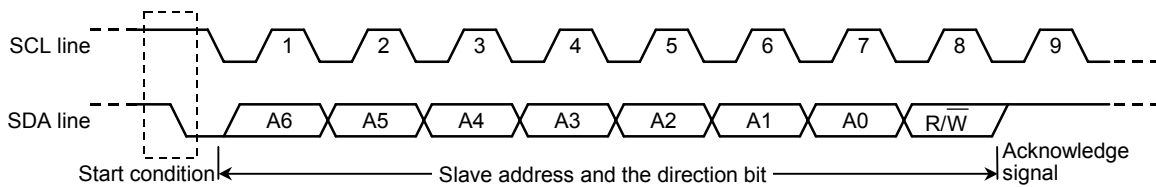


Figure 3.11.9 Start Condition Generation and Slave Address Generation

When SBISR1<BB> = 1, the sequence for generating a stop condition can be initiated by writing 1s to SBISR1<MST, TRX and PIN> and writing 0 to SBISR1<BB>. Do not modify the contents of SBISR1<MST, TRX, BB and PIN> until a stop condition has been generated on the bus.

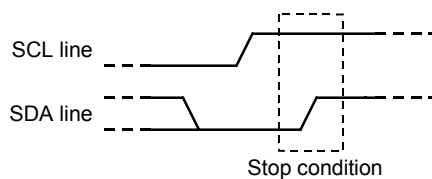


Figure 3.11.10 Stop Condition Generation

The state of the bus can be ascertained by reading the contents of SBISR1<BB>. SBISR1<BB> will be set to 1 if a start condition has been detected on the bus, and will be cleared to 0 if a stop condition has been detected.

Refere to chapter 3.11.6 (4) “Stop condition generation”, where the restriction for the stop-condition in the master-mode is explained.

(8) Interrupt service requests and interrupt cancellation

When a serial bus interface interrupt request (INTI2C1) is generated,

SBI0CR21<PIN> is cleared to 0. The SCL line is pulled low while SBI0CR21<PIN> = 0. SBI0CR21<PIN> is cleared to 0 when a single word of data is transmitted or received. Either writing data to or reading data from SBI0DBR1 sets SBI0CR21<PIN> to 1.

The time from SBI0CR21<PIN> being set to 1 until the release of the SCL line is t_{LOW} .

In address recognition mode (e.g., when SBI0CR21<ALS> = 0), SBI0CR21<PIN> is cleared to 0 when the slave address matches the value set in I2C0AR1 or when a GENERAL CALL is received (All 8-bit data are 0 after a start condition). Although SBI0CR21<PIN> can be set to 1 by a program, writing 0 to SBI0CR21<PIN> does not clear it to 0.

(9) Serial bus interface operation mode selection

SBI0CR21<SBIM1:0> is used to specify the serial bus interface operation mode. Set SBI0CR21<SBIM1:0> to 10 when the device is to be used in I²C bus mode after confirming pin condition of serial bus interface to H. Switch a mode to port after confirming a bus is free.

(10) Arbitration lost detection monitor

Since more than one master device can exist simultaneously on the bus in I²C bus mode, a bus arbitration procedure has been implemented in order to guarantee the integrity of transferred data.

Data on the SDA line is used for I²C bus arbitration.

The following example illustrates the bus arbitration procedure when there are two master devices on the bus. Master A and master B output the same data until point a. After master A outputs L and master B, H, the SDA line of the bus is wire-AND and the SDA line is pulled low by master A. When the bus's SCL line is pulled up at point b, the slave device reads the data on the SDA line, that is, data in master A. Data transmitted from master B becomes invalid. The master B state is known as arbitration lost. A master B device which loses arbitration releases the internal SDA output in order not to affect data transmitted from other masters with arbitration. When more than one master sends the same data at the first word, arbitration occurs continuously after the second word.

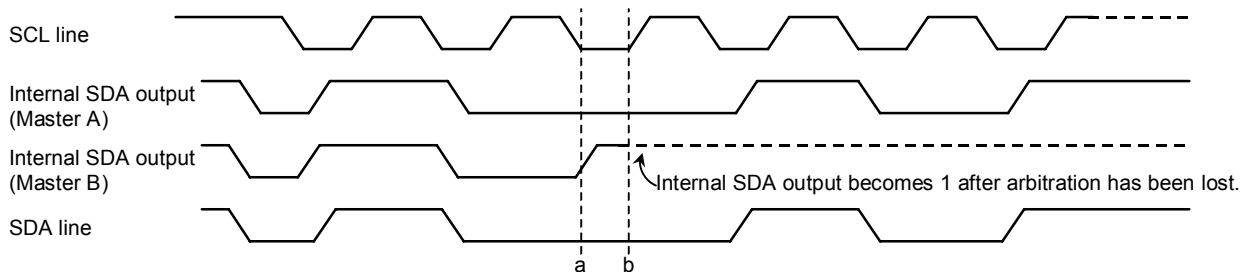


Figure 3.11.11 Arbitration Lost

The TMP91CW18A compares the levels on the bus's SDA line with those of the internal SDA output on the rising edge of the SCL line. If the levels do not match, arbitration is lost and SBISR1<AL> is set to 1.

When SBISR1<AL> is set to 1, SBISR1<MST, TRX> are cleared to 00 and the mode is switched to slave receiver mode. Thus, clock output is stopped in data transfer after setting <AL> = 1.

SBISR1<AL> is cleared to 0 when data is written to or read from SBI0DBR1, when data is written to SBI0CR21.

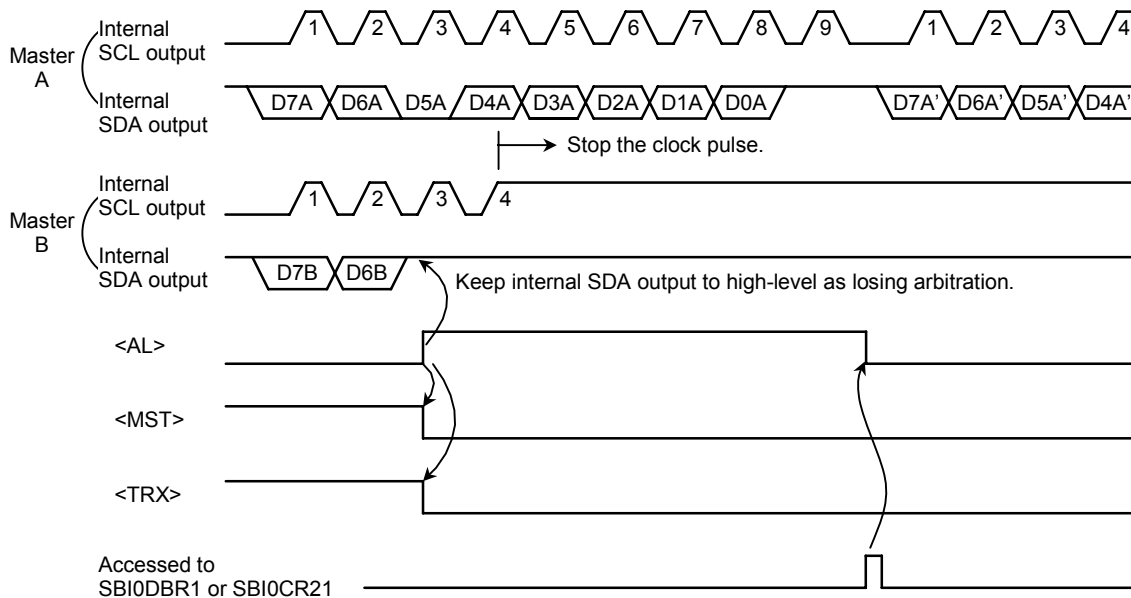


Figure 3.11.12 Example of when TMP91CW18A is a Master Device B (D7A = D7B, D6A = D6B)

(11) Slave address match detection monitor

SBISR1<AAS> is set to 1 in slave mode, in address recognition mode (e.g., when I2C0AR1<ALS> = 0), when a GENERAL CALL is received or a slave address matches the value set in I2C0AR1. When I2C0AR1<ALS> = 1, SBISR1<AAS> is set to 1 after the first word of data has been received. SBISR1<AAS> is cleared to 0 when data is written to or read from the data buffer register SBI0DBR1.

(12) GENERAL CALL detection monitor

SBISR1<AD0> is set to 1 in slave mode, when a GENERAL CALL is received (All 8-bit received data is 0 after a start condition.) SBISR1<AD0> is cleared to 0 when a start condition or stop condition is detected on the bus.

(13) Last received bit monitor

The value on the SDA line detected on the rising edge of the SCL line is stored in SBISR1<LRB>. In acknowledge mode, immediately after an INTI2C1 interrupt request has been generated, an acknowledge signal is read by reading the contents of SBISR1<LRB>.

(14) Software reset function

The software reset function is used to initialize the SBI circuit, when SBI is locked by external noises, etc.

An internal reset signal pulse can be generated by setting SBI0CR21<SWRST1:0> to 10 and 01. This initializes the SBI circuit internally. All command except SBI0CR21<SBIM1:0> registers and status registers are initialized as well.

SBI0CR11<SWRMON> is automatically set to “1” after the SBI circuit has been initialized.

(15) Serial bus interface data buffer register (SBI0DBR1)

Received data can be read by reading SBI0DBR1 and transferred data can be written by writing to SBI0DBR1.

When the start condition has been generated in master mode, the slave address and the direction bit are set in this register.

(16) I²C bus address register (I2C0AR1)

I2C0AR1<SA6:0> is used to set the slave address when the TMP91CW18A functions as a slave device.

If the slave address output from the master device is recognized as matching the TMP91CW18A's slave address, I2C0AR1<ALS> is cleared to 0. The data format is the addressing format. When the slave address output from the master does not match the TMP91CW18A's slave address, I2C0AR1<ALS> is set to 1 and the data format is the free data format.

(17) Baud rate register (SBI0BR11)

Write 1 to SBI0BR11<P4EN> before operation commences.

(18) Setting register for IDLE2 mode operation (SBI0BR01)

The setting of SBI0BR01<I2SBI0> determines whether the device is operating or is stopped in IDLE2 mode. Hence, <I2SBI0> must be set before a HALT instruction is executed.

3.11.6 Data Transfer in I²C Bus Mode

(1) Device initialization

Set SBI0BR11<P4EN> and SBI0CR11<ACK, SCK2:0> to 1. Set SBI0BR11 to 1 and clear bits 7, 6, 5 and 3 of SBI0CR1 to 0.

Set a slave address in I2C0AR1<SA6:0> and I2C0AR<ALS>. (I2C0AR<ALS> = 0 when an addressing format.)

For specifying the default setting to a slave receiver mode, clear the <MST, TRX, BB> to 0 and set the <PIN> to 1, the <SBIM1:0> to 10.

(2) Start condition generation and slave address generation

1. Master mode

In master mode, the start condition and the slave address are generated as follows. Check a bus free status (when <BB> = 0).

Set SBI0CR11<ACK> to 1 (Acknowledge mode) and specify a slave address and a direction bit to be transmitted to the SBI0DBR1.

If SBI0CR11<BB> = 0, the start condition is generated by writing 1111 to SBI0CR21<MST, TRX, BB and PIN>. Subsequently to the start condition, 9 clocks are output from the SCL pin. While 8 clocks are output, the slave address and the direction bit which are set to the SBI0DBR1. On the 9th clock pulse the SDA line is released and the acknowledge signal is received from the slave device.

An INTI2C1 interrupt request occurs on the falling edge of the 9th clock pulse. SBI0CR21<PIN> is cleared to 0. In master mode the SCL pin is pulled low while SBI0CR21<PIN> is 0. When an interrupt request occurs, the value of SBI0CR21<TRX> is changed according to the direction bit setting only if the slave device returns an acknowledge signal.

2. Slave mode

In slave mode the start condition and the slave address are received.

After the start condition has been received from the master device, while 8 clocks are output from the SCL pin, the slave address and the direction bit which are output from the master device are received.

When a GENERAL CALL or an address matching the slave address set in I2C0AR1 is received, the SDA line is pulled down low on the 9th clock pulse and an acknowledge signal is output.

An INTI2C1 interrupt request occurs on the falling edge of the 9th clock pulse. SBI0CR21<PIN> is cleared to 0. In slave mode the SCL line is pulled low while SBI0CR21<PIN> = 0. When an interrupt request occurs, the value of SBI0CR21<TRX> is changed according to the direction bit setting only if the slave device returns an acknowledge signal.

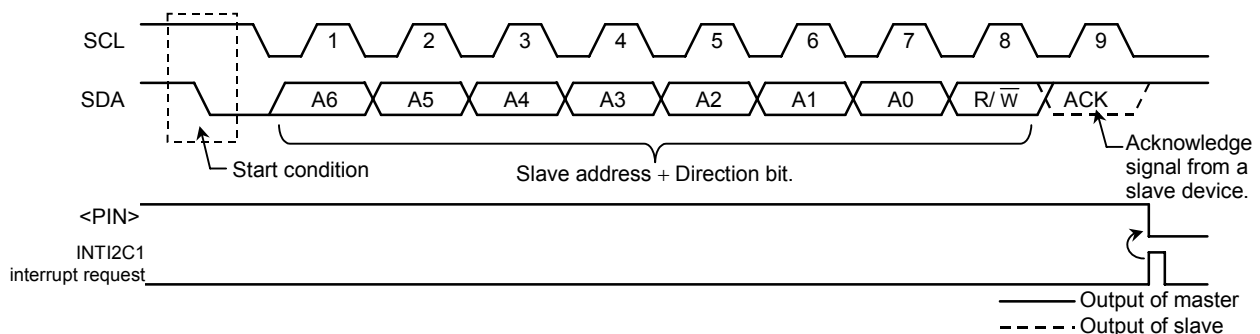


Figure 3.11.13 Start Condition Generation and Slave Address Transfer

(3) Single word data transfer

Check the <MST> setting using an INTI2C1 interrupt process after the transfer of each word of data is completed and determine whether the device is in master mode or slave mode.

1. If <MST> = 1 (Master mode)

Check the <TRX> setting and determine whether the device is in transmitter mode or receiver mode.

If <TRX> = 1 (Transmitter mode)

Check the <LRB> setting. If <LRB> = 1, there is no receiver requesting data. Implement the process for generating a stop condition (See section 3.10.6 (4)) and terminate data transfer.

If <LRB> = 0, the receiver is requesting new data. When the next transmitted data is 8 bits, write the transmitted data to SBI0DBR1. When the next transmitted data is other than 8 bits, set <BC2:0> to 1, set <ACK> to 1 and write the transmitted data to SBI0DBR1. After the data has been written, <PIN> is set to 1, a serial clock pulse is generated to trigger transfer of the next word of data via the SCL pin, and the word is transmitted. After the data has been transmitted, INTI2C1 interrupt request is generated. <PIN> is cleared to 0 and the SCL line is pulled low. If the length of the data to be transferred is greater than one word, repeat the latter steps of the procedure, starting from the check of the <LRB> setting.

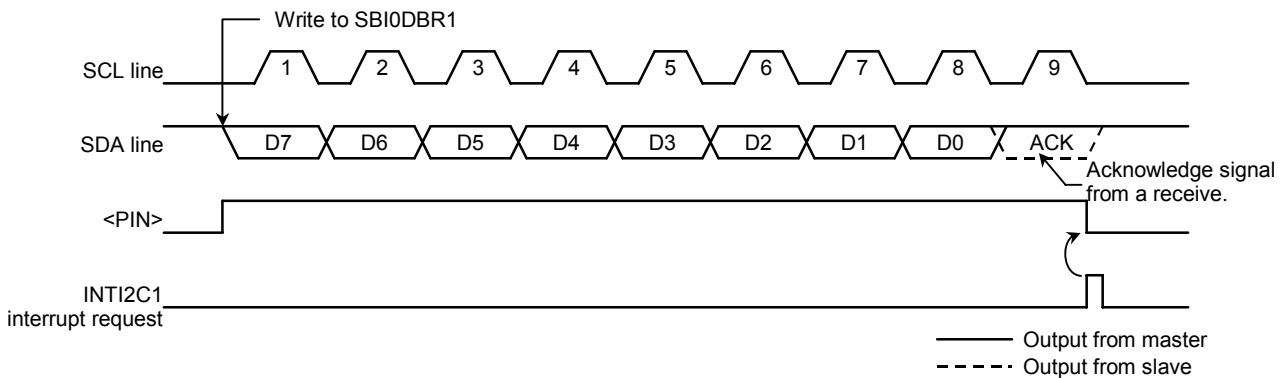


Figure 3.11.14 Example in which <BC2:0> = 000 and <ACK> = 1 in Transmitter Mode

If <TRX> = 0 (Receiver mode)

When the next transmitted data is other than 8 bits, set <BC2:0> again. Set <ACK> to 1 and read the received data from SBI0DBR1 so as to release the SCL line (the value of data which is read immediately after a slave address is sent is undefined). After the data has been read, <PIN> is set to 1.

Serial clock pulse for transferring new 1 word of data is defined SCL and outputs “L” level from SDA pin with acknowledge timing.

An INTI2C1 interrupt request is then generated and <PIN> is cleared to 0. The TMP91CW18A then pulls the SCL pin low. From then on the TMP91CW18A outputs a clock pulse, so as to transfer a data word, and an acknowledge signal each time received data is read from SBI0DBR1.

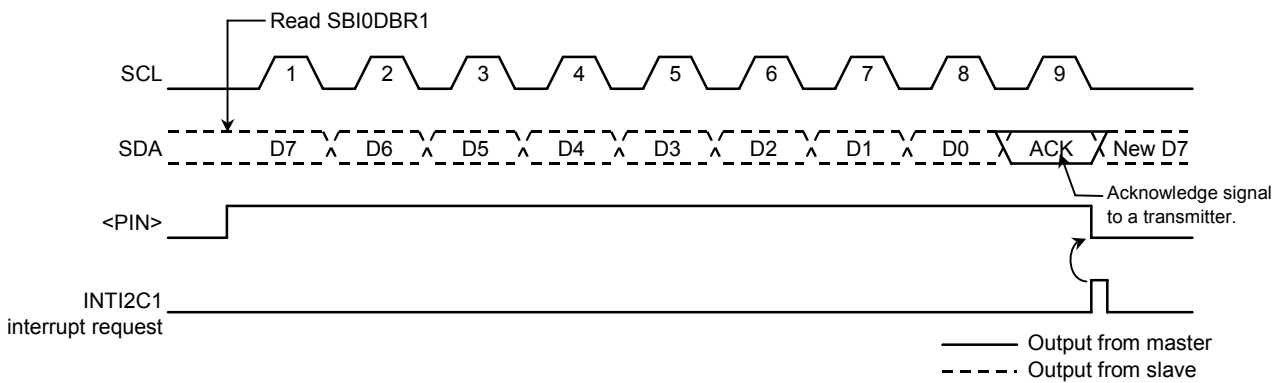


Figure 3.11.15 Example of when <BC2:0> = 000, <ACK> = 1 in Receiver Mode

In order to terminate the transmission of data to a transmitter, clear <ACK> to 0 before reading data which is 1 word before the last data to be received. The last data word does not generate a clock pulse as the acknowledge signal. After the data has been transmitted and an interrupt request has been generated, set <BC2:0> to 001 and read the data. The TMP91CW18A generates a clock pulse for a 1-bit data transfer. Since the master device is a receiver, the SDA line on the bus remains high. The transmitter interprets the high signal as an ACK signal. The receiver indicates to the transmitter that data transfer is complete.

After the 1 data bit has been received and an interrupt request been generated, the TMP91CW18A generates a stop condition (See section 3.10.6 (4)) and terminates data transfer.

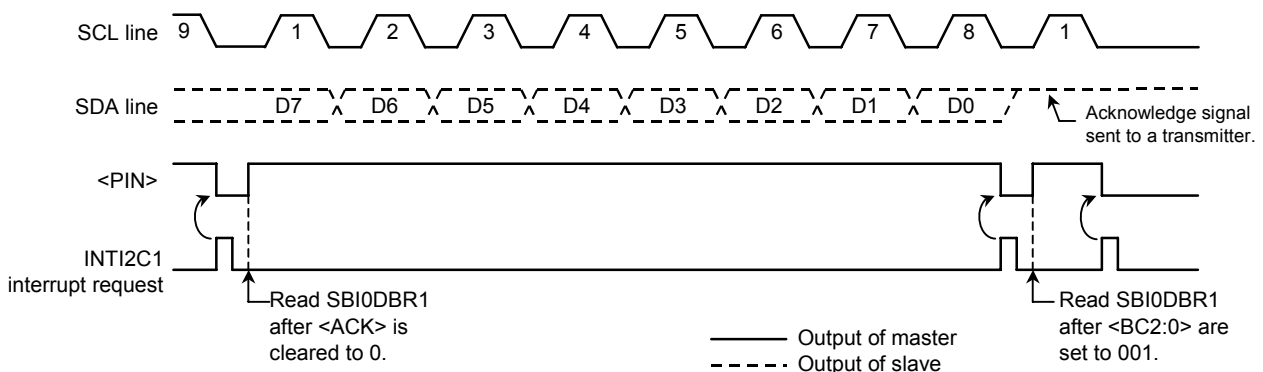


Figure 3.11.16 Termination of Data Transfer in Master Receiver Mode

2. If <MST> = 0 (Slave mode)

In slave mode the TMP91CW18A operates either in normal slave mode or in slave mode after losing arbitration.

In slave mode an INTI2C1 interrupt request is generated when the TMP91CW18A receives a slave address or a GENERAL CALL from the master device, or when a GENERAL CALL is received and data transfer is complete, or when a matching slave address is received. The TMP91CW18A will enter slave mode from master mode if it loses arbitration. An INTI2C1 interrupt request is generated when a word data transfer terminates after arbitration has been lost. When an INTI2C1 interrupt request is generated, <PIN> is cleared to 0 and the SCL pin is pulled low. Either reading data to or writing data from SBI0DBR1, or setting <PIN> to 1 will release the SCL pin after tLOW.

Check the SBISR1<AL>, <TRX>, <AAS> and <AD0> and implements processes according to conditions listed in the next table.

Table 3.11.1 Operation in the Slave Mode

<TRX>	<AL>	<AAS>	<AD0>	Conditions	Process
1	1	1	0	The TMP91CW18A loses arbitration when transmitting a slave address and receives a slave address for which the value of the direction bit sent from another master is 1.	Set the number of bits a word in <BC2:0> and write the transmitted data to SBI0DBR1.
		0	0	In slave receiver mode the TMP91CW18A receives a slave address for which the value of the direction bit sent from the master is 1.	
	0	0	In slave transmitter mode a single word of is transmitted. Set <BC2:0> to the number of bits in a word.	Check the <LRB> setting. If <LRB> is set to 1, set <PIN> to 1 since the receiver win no request the data which follows. Then, clear <TRX> to 0 to release the bus. If <LRB> is cleared to 0 of and write the transmitted data to SBI0DBR1 since the receiver requests next data.	
0	1	1	1/0	The TMP91CW18A loses arbitration when transmitting a slave address and receives a slave address or GENERAL CALL for which the value of the direction bit sent from another master is 0.	Read the SBI0DBR1 for setting the <PIN> to 1 (Reading dummy data) or set the <PIN> to 1.
		0	0	The TMP91CW18A loses arbitration when transmitting a slave address or data and terminates word data transfer.	
	0	1	1/0	In slave receiver mode the TMP91CW18A receives a slave address or GENERAL CALL for which the value of the direction bit sent from the master is 0.	Set <BC2:0> to the number of bits in a word and read the received data from SBI0DBR1.
		0	1/0	In slave receiver mode the TMP91CW18A terminates receiving word data.	

(4) Stop condition generation

When SBISR1<BB> = 1, the sequence for generating a stop condition start by writing 1 to SBI0CR21<MST, TRX, PIN> and “0” to SBI0CR21<BB>. Do not modify the contents of SBI0CR21<MST, TRX, PIN, BB> until a stop condition has been generated on the bus. When the bus’s SCL line has been pulled low by another device, the TMP91CW18A generates a stop condition when the other device has released the SCL line and SDA pin rising.

When SBI0CR21<MST, TRX, PIN> are written 1 and <BB> is written 0 (Generate stop condition in master mode), <BB> changes to 0 by internal SCL changes to 1, without waiting stop condition. To check whether SCL and SDA pin are 1 by sensing their ports is needed to detect bus free condition.

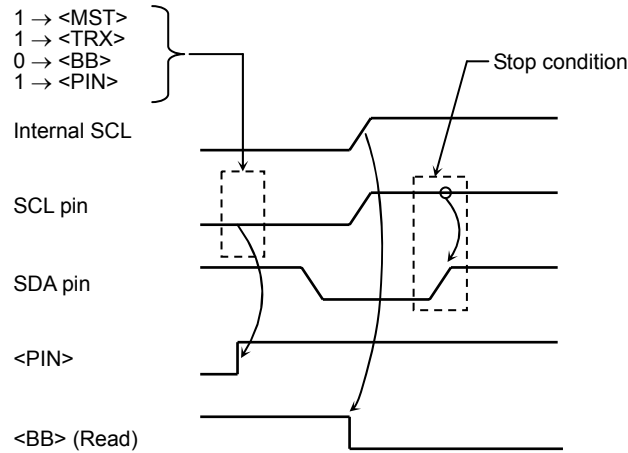


Figure 3.11.17 Stop Condition Generation (Single master)

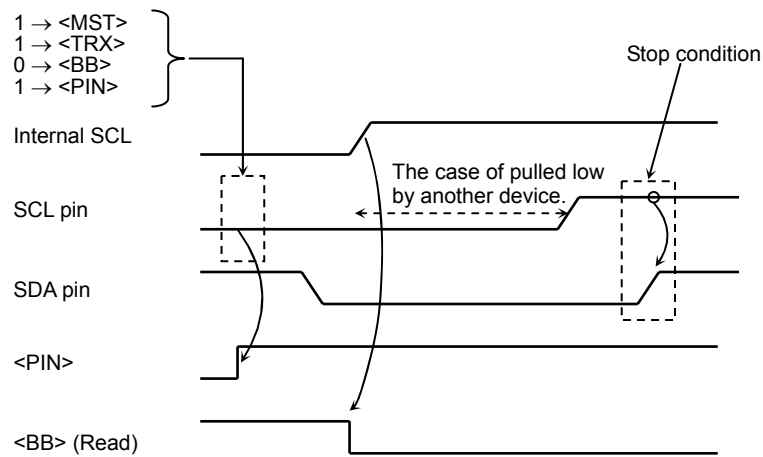


Figure 3.11.18 Stop Condition Generation (Multi master)

(5) Restart

Restart is used during data transfer between a master device and a slave device to change the data transfer direction.

The following description explains how to restart when the TMP91CW18A is in master mode.

Clear SBI0CR21<MST, TRX, BB> to 0 and set SBI0CR21<PIN> to 1 to release the bus. The SDA line remains high and the SCL pin is released. Since a stop condition has not been generated on the bus, other devices assume the bus to be in busy state.

And confirm SCL pin, that SCL pin is released and become bus-free state by SBISR1<BB> = 0 or signal level 1 of SCL pin in port mode. Check the <LRB> until it becomes 1 to check that the SCL line on a bus is not pulled down to the low-level by other devices. After confirming that the bus remains in a free state, generate a start condition using the procedure described in (2).

In order to satisfy the setup time requirements when restarting, take at least 4.7 μ s of waiting time by software from the time of restarting to confirm that the bus is free until the time to generate the start condition.

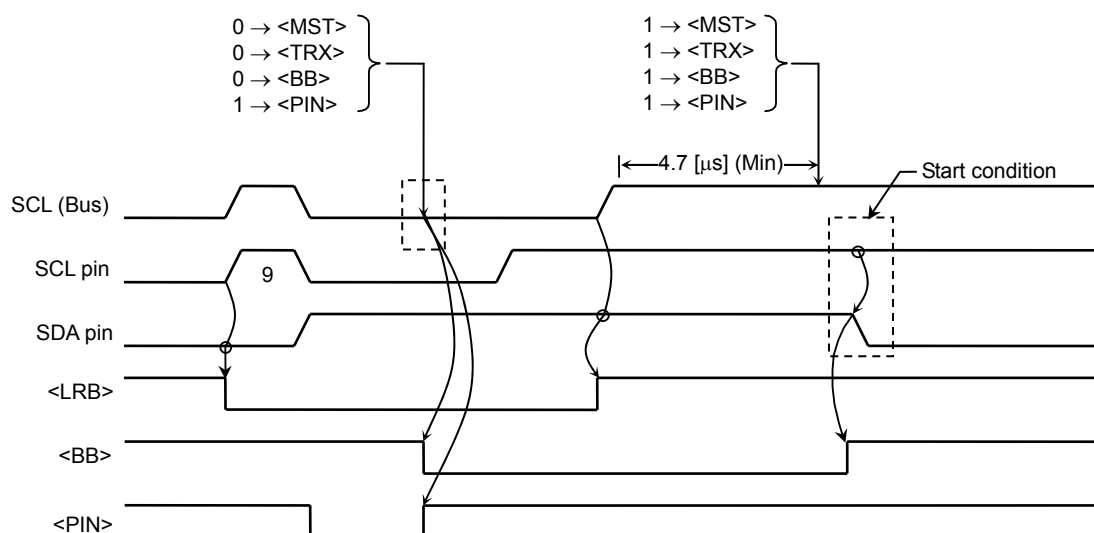


Figure 3.11.19 Timing Diagram for TMP91CW18A Restart

3.12 Serial Bus Interface 2 (I²C bus 2)

The serial bus interface is connected to an external device through P86 (SDA) and P87 (SCL) in the I²C bus mode. Each pin is specified as follows.

	P8CR<P87C, P86C>	P8FC<P87F, P86F>
I ² C bus mode	11	11

3.12.1 Configuration

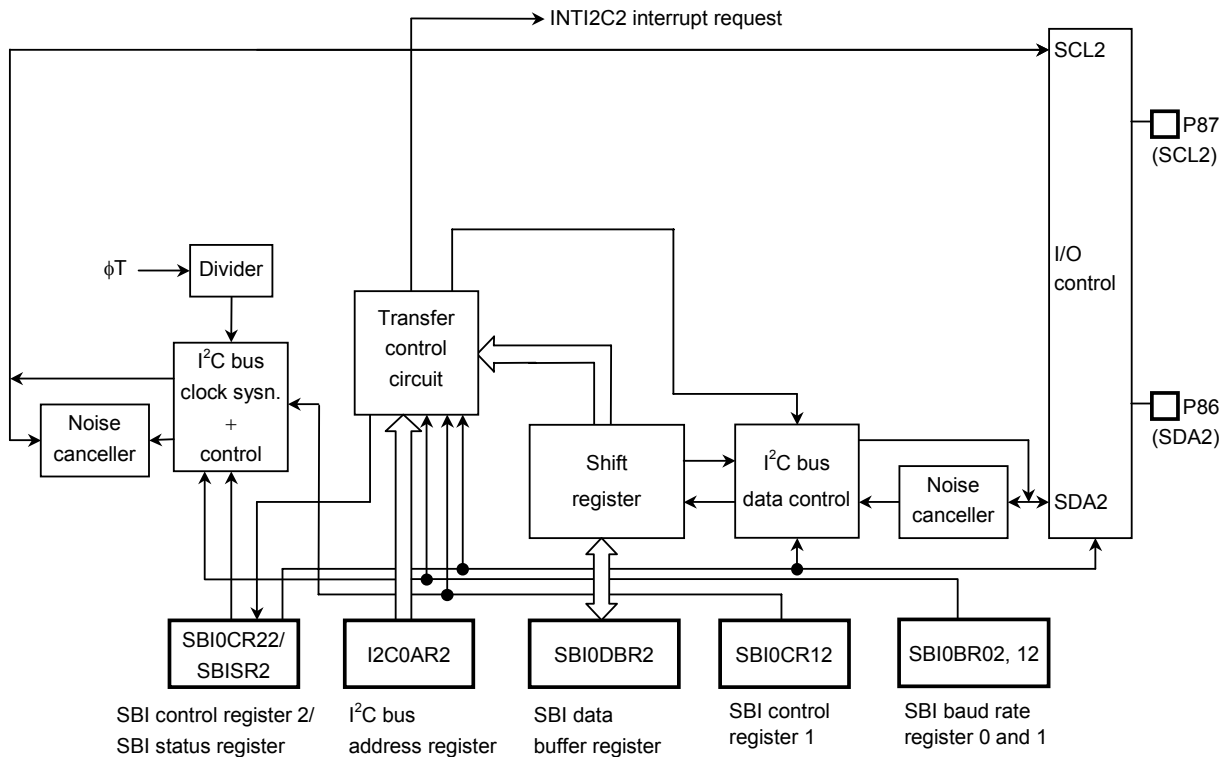


Figure 3.12.1 Serial Bus Interface (I²C bus)

3.12.2 Serial Bus Interface (SBI) Control

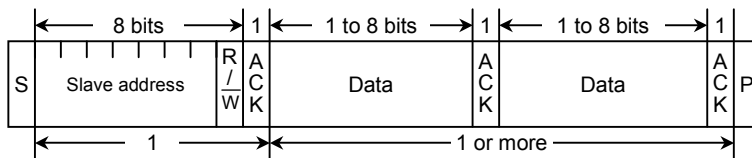
The following registers are used to control the serial bus interface and monitor the operation status.

- Serial bus interface control register 1 (SBI0CR12)
- Serial bus interface control register 2 (SBI0CR22)
- Serial bus interface data buffer register (SBI0DBR2)
- I²C bus address register (I2C0AR2)
- Serial bus interface status register (SBISR2)
- Serial bus interface baud rate register 0 (SBI0BR02)
- Serial bus interface baud rate register 1 (SBI0BR12)

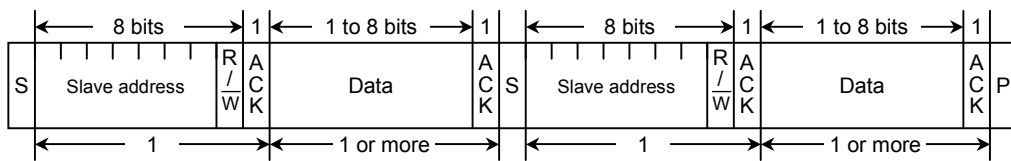
3.12.3 The Data Formats in the I²C Bus Mode

The data formats in the I²C bus mode are shown below.

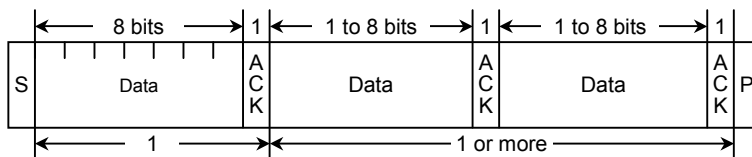
(a) Addressing format



(b) Addressing format (with restart)



(c) Free data format (Data transferred from master device to slave device)

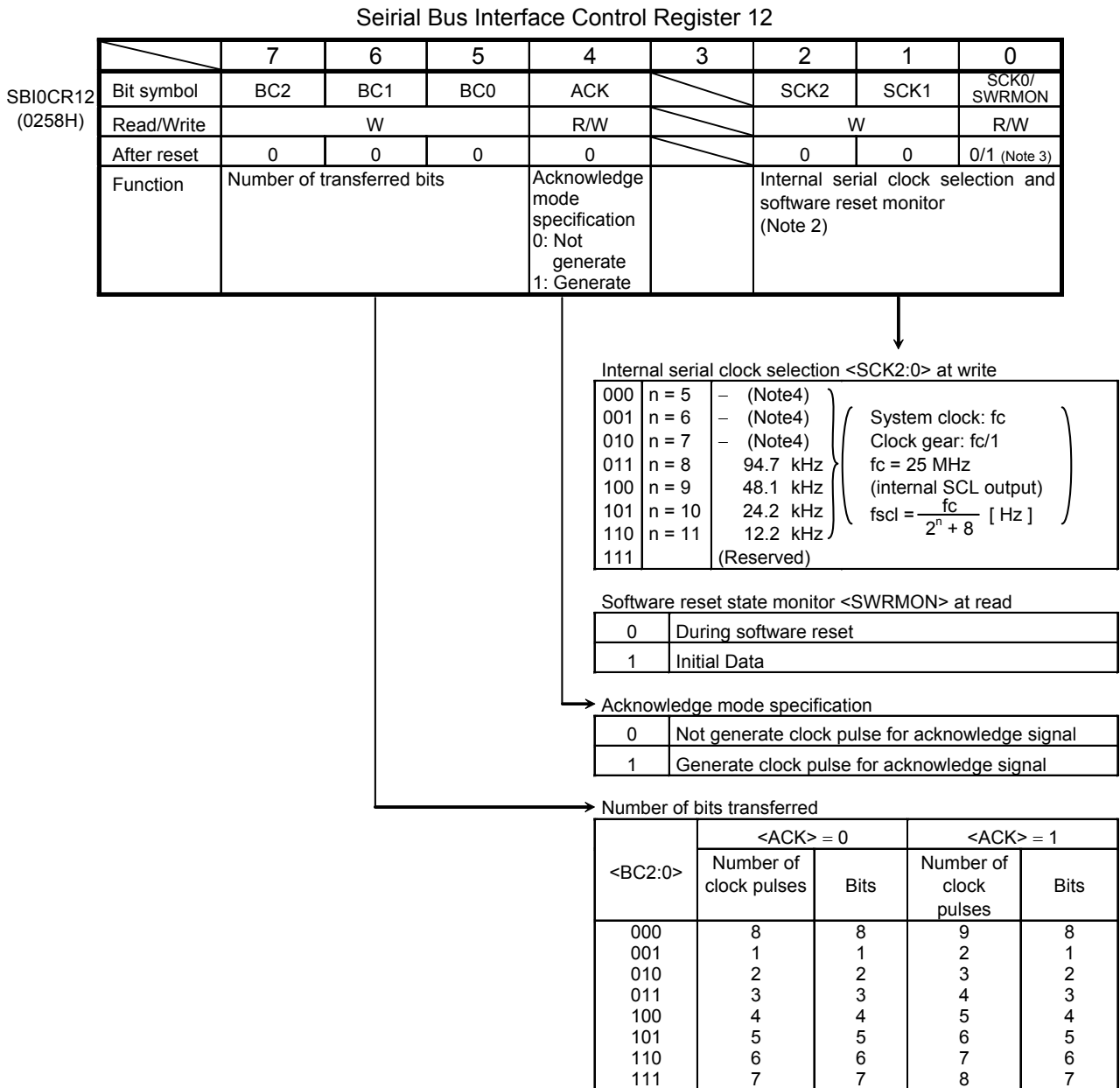


- S: Start condition
- R/W: Direction bit
- ACK: Acknowledge bit
- P: Stop condition

Figure 3.12.2 Data Format in the I²C Bus Mode

3.12.4 I²C Bus Mode Control

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



Note 1: Read-modify-write instruction is prohibited for SBI0CR12.

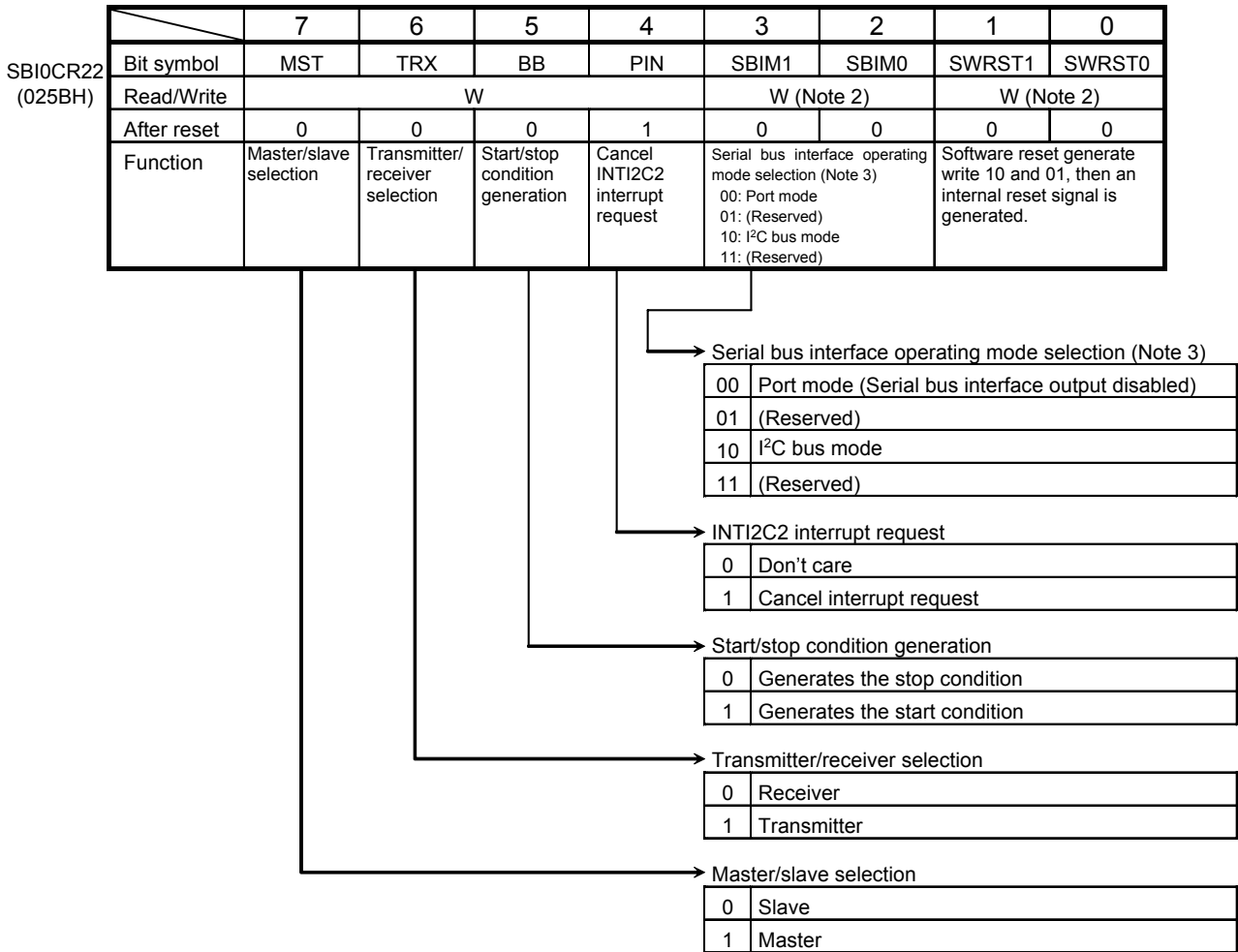
Note 2: For the frequency of the SCL line clock, See 3.12.5 (3) Serial clock.

Note 3: Initial data of SCK0 is 0, SWRMON is 1.

Note 4: This I²C bus circuit does not support Fast mode, it supports standard mode only. Although the I²C bus circuit itself allows the setting of a baud rate over 100 kbps, the compliance with the I²C specification is not guaranteed in that case.

Figure 3.12.3 Registers for the I²C Bus Mode (1/4)

Serial Bus Interface Control Register 2



Note 1: Read-modify-write instruction is prohibited for SBI0CR22.

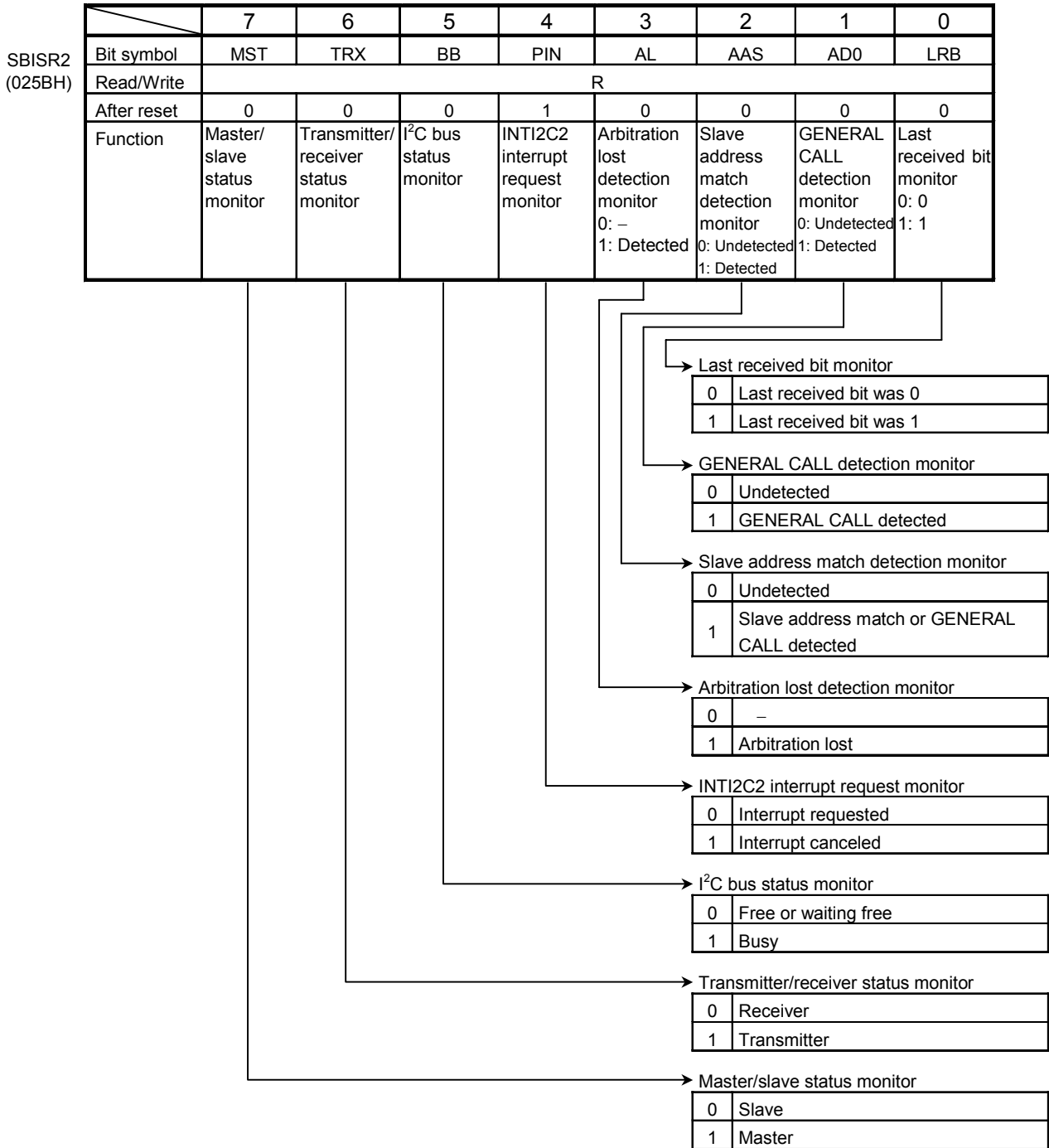
Note 2: Reading this register function as SBISR2 register.

Note 3: Switch a mode to port mode after confirming that the bus is free.

Switch a mode between I²C bus mode and clock synchronous 8-bit SIO mode after confirming that input signals via port are high level.

Figure 3.12.4 Registers for the I²C Bus Mode (2/4)

Serial Bus Interface Status Register



Note 1: Read-modify-write instruction is prohibited for SBISR2.

Note 2: Writing in this register functions as SBI0CR22.

Figure 3.12.5 Registers for the I²C Bus Mode (3/4)

Serial Bus Interface Baud Rate Register 02

	7	6	5	4	3	2	1	0
SBI0BR02 (025CH)	Bit symbol	–	I2SBI0					
	Read/Write	W	R/W					
Read-modify-write instructions are prohibited.	After reset	0	0					
	Function	Always write 0	IDLE2 0: Stop 1: Run					

Operation during IDLE2 mode

0	Stop
1	Run

Serial Bus Interface Baud Rate Register 12

	7	6	5	4	3	2	1	0
SBI0BR12 (025DH)	Bit symbol	P4EN	–					
	Read/Write	W	W					
Read-modify-write instructions are prohibited.	After reset	0	0					
	Function	Internal clock 0: Stop 1: Run	Always write 0					

Baud rate clock control

0	Stop
1	Run

Serial Bus Interface Data Buffer Register 2

	7	6	5	4	3	2	1	0	
SBI0DBR2 (0259H)	Bit symbol	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	Read/Write	R (Received)/W (Transfer)							
Read-modify-write instruction is prohibited.	After reset	Undefined							

- Note 1: When writing transmitted data, start from the MSB (Bit7). Receiving data is placed from LSB (Bit0).
- Note 2: SBI0DBR2 can't be read the written data. Therefore read-modify-write instruction (e.g., "BIT" instruction) is prohibited.
- Note 3: Written data in SBI0DBR2 is cleared by INTI2C2 signal.

I²C Bus Address Register 2

	7	6	5	4	3	2	1	0	
I2C0AR2 (0254H)	Bit symbol	SA6	SA5	SA4	SA3	SA2	SA1	SA0	ALS
	Read/Write	W							
Read-modify-write instruction is prohibited.	After reset	0	0	0	0	0	0	0	0
	Function	Slave address selection for when dePvice is operating as slave device.							Address recognition mode specification

Address recognition mode specification

0	Slave address recognition
1	Non-slave address recognition

Figure 3.12.6 Registers for the I²C Bus Mode (4/4)

3.12.5 Control in I²C Bus Mode

(1) Specifying acknowledge mode

Set the SBI0CR12<ACK> to 1 for operation in the acknowledge mode. The TMP91CW18A generates an additional clock pulse for an acknowledge signal when operating in master mode. In the transmitter mode during the clock pulse cycle, the SDA pin is released in order to receive the acknowledge signal from the receiver. In the receiver mode during the clock pulse cycle, the SDA pin is set to the low in order to generate the acknowledge signal.

Clear the <ACK> to 0 for operation in the non-acknowledge mode, the TMP91CW18A does not generate a clock pulse for the acknowledge signal when operating in the master mode.

(2) Number of transfer bits

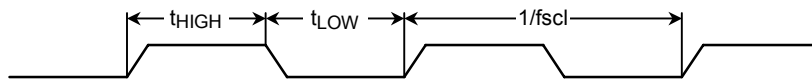
SBI0CR12<BC2:0> is used to select a number of bits for next transmitting and receiving data.

Since SBI0CR12<BC2:0> is cleared to 000 on start-up, a slave address and direction bit transmissions are executed in 8 bits. Other than these, the <BC2:0> retains a specified value.

(3) Serial clock

1. Clock source

SBI0CR12<SCK2:0> is used to specify the maximum transfer frequency for output on the SCL pin in master mode. Set a communication baud rate that meets the I²C bus specification, such as the shortest pulse width of t_{LOW}, based on the equations shown below.



$$t_{LOW} = 2^{n-1}/f_{SBI}$$

$$t_{HIGH} = 2^{n-1}/f_{SBI} + 8/f_{SBI}$$

$$f_{scl} = 1/(t_{LOW} + t_{HIGH})$$

$$= \frac{f_{SBI}}{2^n + 8}$$

SBI0CR12<SCK2:0>	n
000	5
001	6
010	7
011	8
100	9
101	10
110	11

Note 1: f_{SBI} is the clock f_{FPH}.

Note 2: f_{SBI} is the clock either fc/16 or f_{FPH} which is selected SYSCR0<PRCK1:0>.

Figure 3.12.7 Clock Source

2. Clock synchronization

In the I²C bus mode, in order to wired-AND a bus, a master device which pulls down a clock line to low level, in the first place, invalidate a clock pulse of another master device which generates a high-level clock pulse. The master device with a high-level clock pulse needs to detect the situation and implement the following procedure.

The TMP91CW18A has a clock synchronization function which allows normal data transfer even when more than one master exists on the bus.

The following example explains the clock synchronization procedures used when there are two masters present on the bus.

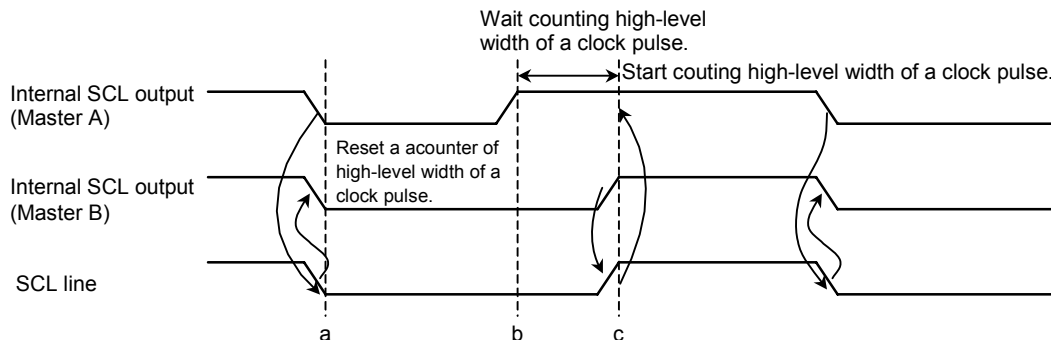


Figure 3.12.8 Clock Synchronization

When master A pulls the internal SCL output low at point a, the bus's SCL line goes low. After detecting this, master B resets a counter of high-level width of an own clock pulse and sets the internal SCL output low.

Master A finishes counting low-level width of an own clock pulse at point b and sets the internal SCL output high. Since master B is holding the bus's SCL line low, master A waits for counting high-level width of an own clock pulse. After master B has finished counting low-level width of an own clock pulse at point c and master A detects the SCL line of the bus at the high level, and starts counting high level of an own clock pulse.

The clock pulse on the bus is determined by the master device with the shortest high-level width and the master device with the longest low-level width from among those master devices connected to the bus.

(4) Slave address and address recognition mode specification

When the TMP91CW18A is to be used as a slave device, set the slave address <SA6:0> and <ALS> in I2C0AR2. Clear <ALS> to 0 for the address recognition mode.

(5) Master/slave selection

To operate the TMP91CW18A as a master device set SBI0CR22<MST> to 1. To operate it as a slave device clear SBI0CR22<MST> to 0. SBI0CR22<MST> is cleared to 0 in hardware when a stop condition is detected on the bus or when arbitration is lost.

(6) Transmitter/receiver selection

To operate the TMP91CW18A as a transmitter set SBI0CR22<TRX> to 1. To operate it as a receiver clear SBI0CR22<TRX> to 0. When data with an addressing format is transferred in slave mode, when a slave address with the same value that an I2C0AR2 or a GENERAL CALL is received (All 8-bit data are 0 after a start condition), SBI0CR22<TRX> is set to 1 in hardware if the direction bit (R/ \bar{W}) sent from the master device is 1, and is cleared to 0 in hardware if the bit is 0. In master mode, when an acknowledge signal is returned from the slave device, SBI0CR22<TRX> is cleared to 0 in hardware if the value of the transmitted direction bit is 1, and is set to 1 in hardware if the value of the bit is 0. If an acknowledge signal is not returned, the current state is maintained.

SBI0CR22<TRX> is cleared to 0 in hardware when a stop condition is detected on the I²C bus or when arbitration is lost.

(7) Start/stop condition generation

When SBISR2<BB> = 0, slave address and direction bit which are set in SBI0DBR2 are output on the bus after generating a start condition by programming 1 to the SBI0CR22<MST, TRX, BB and PIN>. It is necessary to set transmitted data to the data buffer register (SBI0DBR2) and set 1 to <ACK> beforehand.

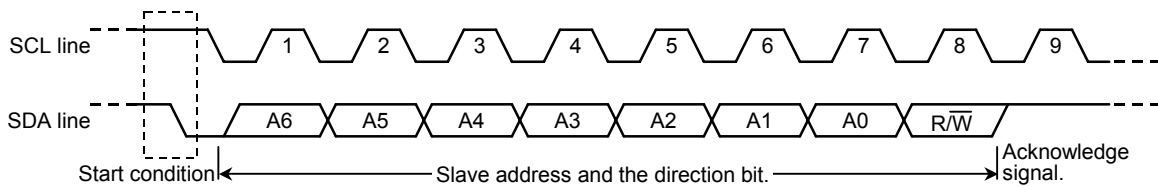


Figure 3.12.9 Start Condition Generation and Slave Address Generation

When SBISR2<BB> = 1, the sequence for generating a stop condition can be initiated by writing 1s to SBI0SR22<MST, TRX and PIN> and programming 0 to SBISR2<BB>. Do not modify the contents of SBI0SR22<MST, TRX, BB and PIN> until a stop condition has been generated on the bus.

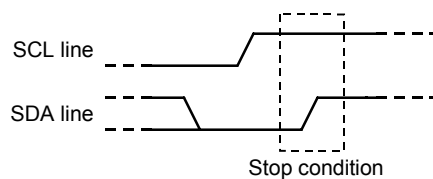


Figure 3.12.10 Stop Condition Generation

The state of the bus can be ascertained by reading the contents of SBISR2<BB>. SBISR2<BB> will be set to 1 if a start condition has been detected on the bus, and will be cleared to 0 if a stop condition has been detected.

Refer to chapter 3.12.6 (4) “Stop condition generation”, where the restriction for the stop condition in the master mode is explained.

(8) Interrupt service requests and interrupt cancellation

When a serial bus interface interrupt request (INTI2C2) is generated, SBI0CR22<PIN> is cleared to 0. The SCL line is pulled low while SBI0CR22<PIN> = 0.

SBI0CR22<PIN> is cleared to 0 when a single word of data is transmitted or received. Either writing data to or reading data from SBI0DBR2 sets SBI0CR22<PIN> to 1.

The time from SBI0CR22<PIN> being set to 1 until the release of the SCL line is t_{LOW} .

In address recognition mode (e.g., when SBI0CR22<ALS> = 0), SBI0CR22<PIN> is cleared to 0 when the slave address matches the value set in I2COAR2 or a GENERAL CALL is received (All 8-bit data are 0 after a start condition). Although SBI0CR22<PIN> can be set to 1 by a program, programming 0 to SBI0CR22<PIN> does not clear it to 0.

(9) Serial bus interface operation mode selection

SBI0CR22<SBIM1:0> is used to specify the serial bus interface operation mode. Set SBI0CR22<SBIM1:0> to 10 when the device is to be used in I²C bus mode. After confirming pin condition of serial bus interface to high.

Switch a mode to port after confirming a bus is free.

(10) Arbitration lost detection monitor

Since more than one master device can exist simultaneously on the bus in I²C bus mode, a bus arbitration procedure has been implemented in order to guarantee the integrity of transferred data.

Data on the SDA line is used for I²C bus arbitration.

The following example illustrates the bus arbitration procedure when there are 2 master devices on the bus. Master A and master B output the same data until point a. After master A outputs L and master B, H, the SDA line of the bus is wire-AND and the SDA line is pulled low by master A. When the bus's SCL line is pulled up at point b, the slave device reads the data on the SDA line, that is, data in master A. Data transmitted from master B becomes invalid. The master B state is known as arbitration lost. A master B device which loses arbitration releases the internal SDA output in order not to affect data transmitted from other masters with arbitration. When more than one master sends the same data at the 1st word, arbitration occurs continuously after the 2nd word.

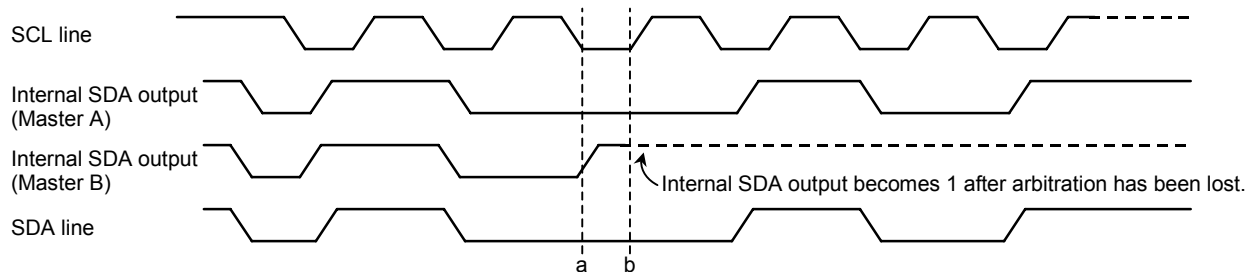


Figure 3.12.11 Arbitration Lost

The TMP91CW18A compares the levels on the bus's SDA line with those of the internal SDA output on the rising edge of the SCL line. If the levels do not match, arbitration is lost and SBISR2<AL> is set to 1.

When SBISR2<AL> is set to 1, SBISR2<MST, TRX> are cleared to 00 and the mode is switched to slave receiver mode. Thus, clock output is stopped in data transfer after setting <AL> = "1".

SBISR2<AL> is cleared to 0 when data is written to or read from SBI0DBR2 or when data is written to SBI0CR22.

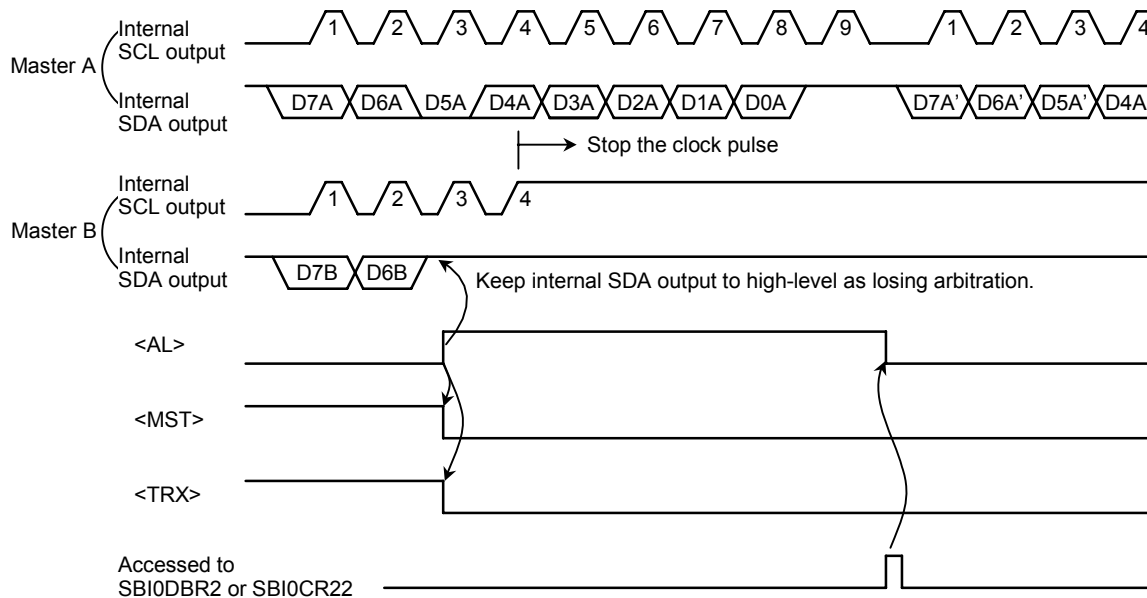


Figure 3.12.12 Example of when TMP91CW18A is a Master Device B (D7A = D7B, D6A = D6B)

(11) Slave address match detection monitor

SBISR2<AAS> is set to 1 in slave mode, in address recognition mode (e.g., when I2C0AR2<ALS> = 0), when a GENERAL CALL is received or a slave address matches the value set in I2C0AR2. When I2C0AR2<ALS> = 1, SBISR2<AAS> is set to 1 after 1st word of data has been received. SBISR2<AAS> is cleared to 0 when data is written to or read from the data buffer register SBI0DBR2.

(12) GENERAL CALL detection monitor

SBISR2<AD0> is set to 1 in slave mode when a GENERAL CALL is received (All 8-bit received data is 0 after a start condition). SBISR2<AD0> is cleared to 0 when a start condition or stop condition is detected on the bus.

(13) Last received bit monitor

The value on the SDA line detected on the rising edge of the SCL line is stored in SBISR2<LRB>. In acknowledge mode, immediately after an INTI2C2 interrupt request has been generated, an acknowledge signal is read by reading the contents of SBISR2<LRB>.

(14) Software reset function

The software reset function is used to initialize the SBI circuit when SBI is locked by external noises, etc.

An internal reset signal pulse can be generated by setting SBI0CR22<SWRST1:0> to 10 and 01. This initializes the SBI circuit internally. All command except SBI0CR22<SBIM1:0> registers and status registers are initialized as well.

SBI0CR12<SWRMON> is automatically set to 1 after the SBI circuit has been initialized.

(15) Serial bus interface data buffer register (SBI0DBR2)

Received data can be read by reading SBI0DBR2 and transferred data can be written by writing to SBI0DBR2.

When the start condition has been generated in master mode, the slave address and the direction bit are set in this register.

(16) I²C bus address register (I2C0AR2)

I2C0AR2<SA6:0> is used to set the slave address when the TMP91CW18A functions as a slave device.

If the slave address output from the master device is recognized as matching the TMP91CW18A's slave address, I2C0AR2<ALS> is cleared to 0. The data format is the addressing format. When the slave address output from the master does not match the TMP91CW18A's slave address, I2C0AR2<ALS> is set to 1 and the data format is the free data format.

(17) Baud rate register (SBI0BR12)

Write 1 to SBI0BR12<P4EN> before operation commences.

(18) Setting register for IDLE2 mode operation (SBI0BR02)

The setting of SBI0BR02<I2SBI0> determines whether the device is operating or is stopped in IDLE2 mode. Hence, <I2SBI0> must be set before a HALT instruction is executed.

3.12.6 Data Transfer in I²C Bus Mode

(1) Device initialization

Set SBI0BR12<P4EN> and SBI0CR12<ACK, SCK2:0> to 1. Set SBI0BR12<P4EN> to 1 and clear bits 7, 6, 5 and 3 of SBI0CR12.

Set a slave address in I2C0AR2<SA6:0> and I2C0AR2<ALS> (I2C0AR2<ALS> = 0 when an addressing format).

For specifying the default setting to a slave receiver mode, clear the <MST, TRX, BB> to 0 and set the <PIN> to 1, the <SBIM1:0> to 10.

(2) Start condition generation and slave address generation

1. Master mode

In master mode, the start condition and the slave address are generated as follows.

Check a bus free status (when <BB> = 0).

Set SBI0CR12<ACK> to 1 (Acknowledge mode) and specify a slave address and a direction bit to be transmitted to the SBI0DBR2.

If SBI0CR22<BB> = 0, the start condition is generated by writing 1111 to SBI0CR22<MST, TRX, BB and PIN>. Subsequently to the start condition, 9 clocks are output from the SCL pin. While 8 clocks are output, the slave address and the direction bit which are set to the SBI0DBR2. On the 9th clock pulse the SDA line is released and the acknowledge signal is received from the slave device.

An INTI2C2 interrupt request occurs on the falling edge of the 9th clock pulse. SBI0CR22<PIN> is cleared to 0. In master mode the SCL pin is pulled low while SBI0CR22<PIN> is 0. When an interrupt request occurs, the value of SBI0CR22<TRX> is changed according to the direction bit setting only if the slave device returns an acknowledge signal.

2. Slave mode

In slave mode the start condition and the slave address are received.

After the start condition has been received from the master device, while 8 clocks are output from the SCL pin, the slave address and the direction bit which are output from the master device are received.

When a GENERAL CALL or an address matching the slave address set in I2C0AR2 is received, the SDA line is pulled down low on the 9th clock pulse and an acknowledge signal is output.

An INTI2C2 interrupt request occurs on the falling edge of the 9th clock pulse. SBI0CR22<PIN> is cleared to 0. In slave mode the SCL line is pulled low while SBI0CR22<PIN> = 0. When an interrupt request occurs, the value of SBI0CR22<TRX> is changed according to the direction bit setting only if the slave device returns an acknowledge signal.

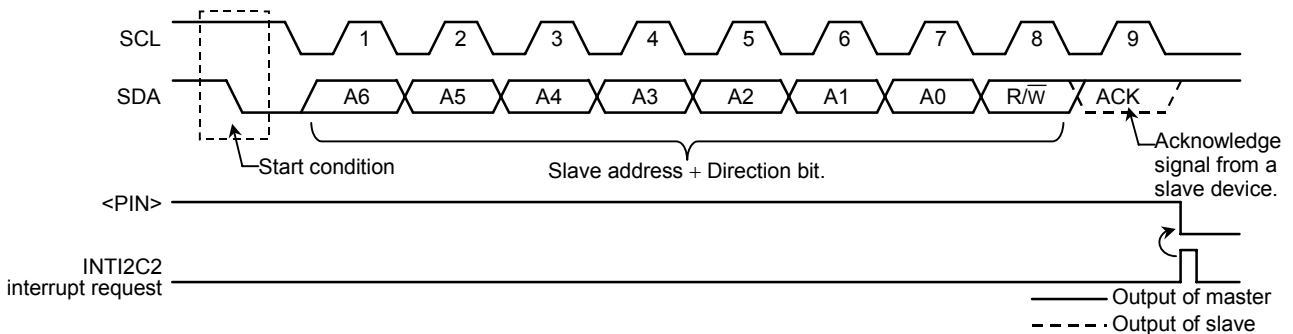


Figure 3.12.13 Start Condition Generation and Slave Address Transfer

(3) Single-word data transfer

Check the <MST> setting using an INTI2C2 interrupt process after the transfer of each word of data is completed and determine whether the device is in master mode or slave mode.

1. If <MST> = 1 (Master mode)

Check the <TRX> setting and determine whether the device is in transmitter mode or receiver mode.

If <TRX> = 1 (Transmitter mode)

Check the <LRB> setting. If <LRB> = 1, there is no receiver requesting data. Implement the process for generating a stop condition (See section 3.10.6 (4)) and terminate data transfer.

If <LRB> = 0, the receiver is requesting new data. When the next transmitted data is 8 bits, write the transmitted data to SBI0DBR2. When the next transmitted data is other than 8 bits, set <BC2:0> to 1, set <ACK> to 1 and write the transmitted data to SBI0DBR2. After the data has been written, <PIN> is set to 1, a serial clock pulse is generated to trigger transfer of the next word of data via the SCL pin, and the word is transmitted. After the data has been transmitted, an INTI2C2 interrupt request is generated. <PIN> is cleared to 0 and the SCL line is pulled low. If the length of the data to be transferred is greater than one word, repeat the latter steps of the procedure, starting from the check of the <LRB> setting.

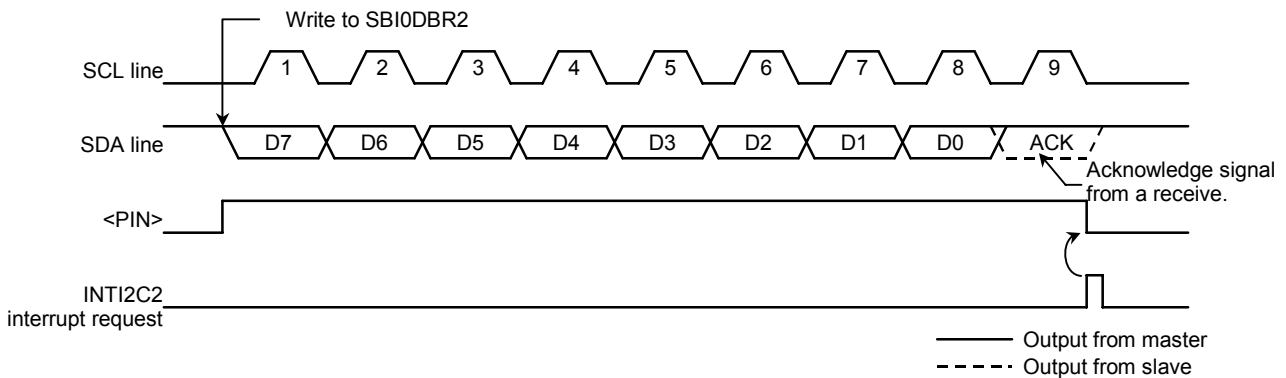


Figure 3.12.14 Example in which <BC2:0> = 000 and <ACK> = 1 in Transmitter Mode

If <TRX> = 0 (Receiver mode)

When the next transmitted data is other than 8 bits, set <BC2:0> again. Set <ACK> to 1 and read the received data from SBI0DBR2 so as to release the SCL line. (The value of data which is read immediately after a slave address is sent is undefined.) After the data has been read, <PIN> is set to 1.

Serial clock pulse for transferring new 1 word of data is defined SCL and outputs “L” level from SDA pin with acknowledge timing.

An INTI2C2 interrupt request is then generated and <PIN> is cleared to 0. The TMP91CW18A then pulls the SCL pin low. From then on the TMP91CW18A outputs a clock pulse, so as to transfer a data word, and an acknowledge signal each time received data is read from SBI0DBR2.

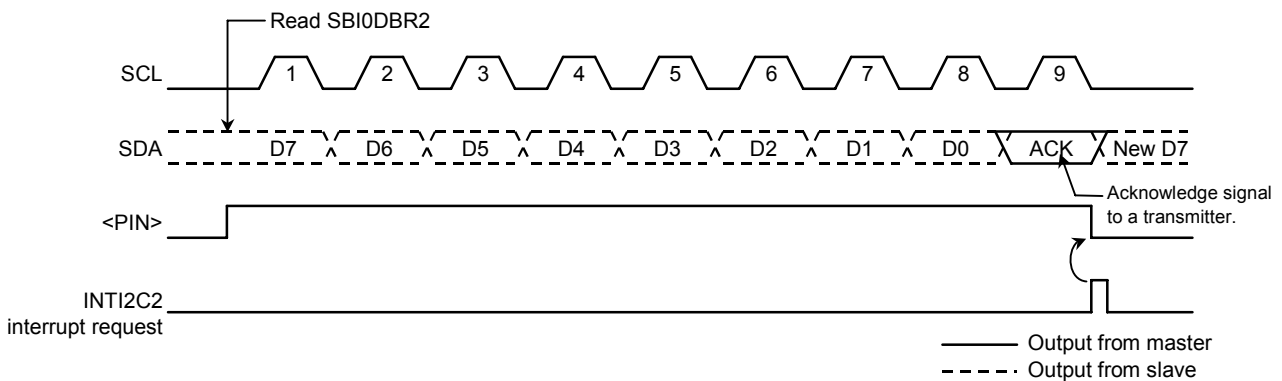


Figure 3.12.15 Example of when <BC2:0> = “000”, <ACK> = “1” in Receiver Mode

In order to terminate the transmission of data to a transmitter, clear <ACK> to 0 before reading data which is 1 word before the last data to be received. The last data word does not generate a clock pulse as the acknowledge signal. After the data has been transmitted and an interrupt request has been generated, set <BC2:0> to 001 and read the data. The TMP91CW18A generates a clock pulse for a 1-bit data transfer. Since the master device is a receiver, the SDA line on the bus remains high. The transmitter interprets the high signal as an ACK signal. The receiver indicates to the transmitter that data transfer is completed.

After the 1 data bit has been received and an interrupt request been generated, the TMP91CW18A generates a stop condition (See section 3.10.6 (4)) and terminates data transfer.

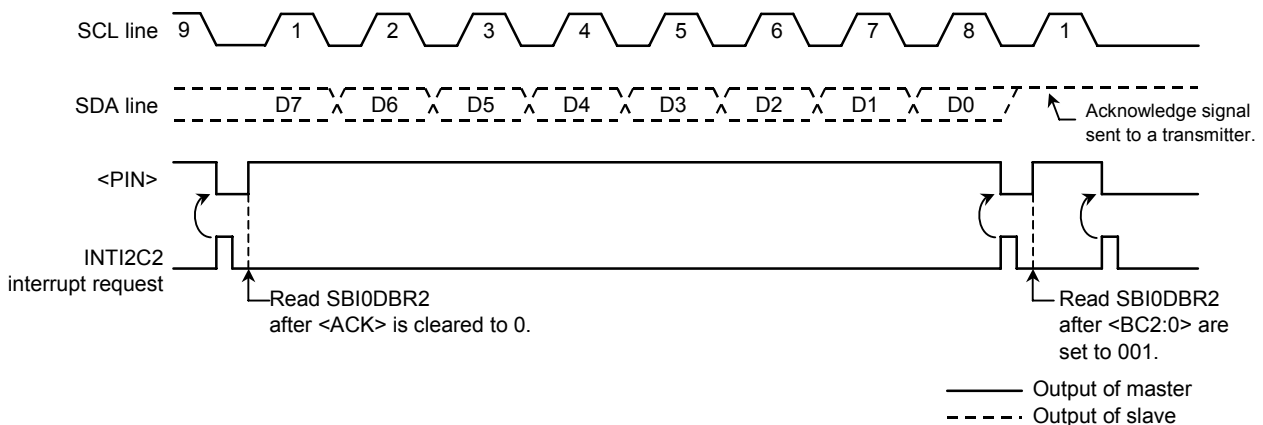


Figure 3.12.16 Termination of Data Transfer in Master Receiver Mode

2. If <MST> = 0 (Slave mode)

In slave mode the TMP91CW18A operates either in normal slave mode or in slave mode after losing arbitration.

In slave mode an INTI2C2 interrupt request is generated when the TMP91CW18A receives a slave address or a GENERAL CALL from the master device, or when a GENERAL CALL is received and data transfer is complete, or when a matching slave address is received. The TMP91CW18A will enter slave mode from master mode if it loses arbitration. An INTI2C2 interrupt request is generated when a word data transfer terminates after arbitration has been lost. When an INTI2C2 interrupt request is generated, <PIN> is cleared to 0 and the SCL pin is pulled low. Either reading data to or writing data from SBI0DBR2, or setting <PIN> to 1 will release the SCL pin after t_{LOW}.

Check the SBISR2<AL>, <TRX>, <AAS> and <AD0> and implements processes according to conditions listed in the next table.

Table 3.12.1 Operation in the Slave Mode

<TRX>	<AL>	<AAS>	<AD0>	Conditions	Process
1	1	1	0	The TMP91CW18A loses arbitration when transmitting a slave address and receives a slave address for which the value of the direction bit sent from another master is 1.	Set the number of bits a word in <BC2:0> and write the transmitted data to SBI0DBR2.
		0	0	In slave receiver mode the TMP91CW18A receives a slave address for which the value of the direction bit sent from the master is 1.	
	0	0	0	In slave transmitter mode a single word of is transmitted. Set <BC2:0> to the number of bits in a word.	Check the <LRB> setting. If <LRB> is set to 1, set <PIN> to 1 since the receiver win no request the data which follows. Then, clear <TRX> to 0 to release the bus. If <LRB> is cleared to 0 of and write the transmitted data to SBI0DBR2 since the receiver requests next data.
0	1	1	1/0	The TMP91CW18A loses arbitration when transmitting a slave address and receives a slave address or GENERAL CALL for which the value of the direction bit sent from another master is 0.	Read the SBI0DBR2 for setting the <PIN> to 1 (Reading dummy data) or set the <PIN> to 1.
		0	0	The TMP91CW18A loses arbitration when transmitting a slave address or data and terminates word data transfer.	
	0	1	1/0	In slave receiver mode the TMP91CW18A receives a slave address or GENERAL CALL for which the value of the direction bit sent from the master is 0.	Set <BC2:0> to the number of bits in a word and read the received data from SBI0DBR2.
		0	1/0	In slave receiver mode the TMP91CW18A terminates receiving word data.	

(4) Stop condition generation

When $SBISR2<BB> = 1$, the sequence for generating a stop condition start by writing “1” to $SBI0CR22<MST, TRX, PIN>$ and “0” to $SBI0CR22<BB>$. Do not modify the contents of $SBI0CR22<MST, TRX, PIN, BB>$ until a stop condition has been generated on the bus. When the bus’s SCL line has been pulled low by another device, the TMP91CW18A generates a stop condition when the other device has released the SCL line and SDA pin rising.

When $SBI0CR22<MST, TRX, PIN>$ are written 1 and $<BB>$ is written 0 (Generate stop condition in master mode), $<BB>$ changes to 0 by internal SCL changes to 1, without waiting stop condition. To check whether SCL and SDA pin are 1 by sensing their ports is needed to detect bus free condition.

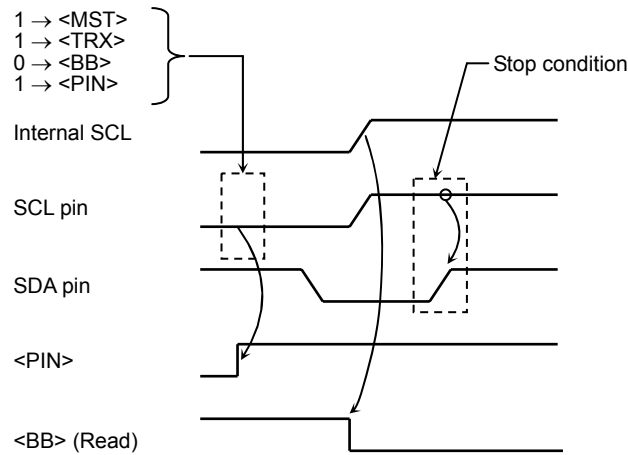


Figure 3.12.17 Stop Condition Generation (Single master)

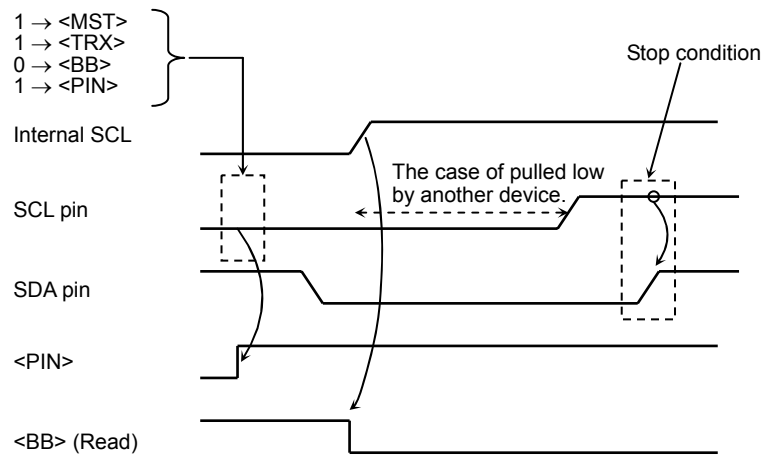


Figure 3.12.18 Stop Condition Generation (Multi master)

(5) Restart

Restart is used during data transfer between a master device and a slave device to change the data transfer direction.

The following description explains how to restart when the TMP91CW18A is in master mode.

Clear SBI0CR22<MST, TRX, BB> to 0 and set SBI0CR22<PIN> to 1 to release the bus. The SDA line remains high and the SCL pin is released. Since a stop condition has not been generated on the bus, other devices assume the bus to be in busy state.

And confirm SCL pin, that SCL pin is released and become bus-free state by SBISR2<BB> = 0 or signal level 1 of SCL pin in port mode. Check the <LRB> until it becomes 1 to check that the SCL line on a bus is not pulled down to the low-level by other devices. After confirming that the bus remains in a free state, generate a start condition using the procedure described in (2).

In order to satisfy the setup time requirements when restarting, take at least 4.7 μ s of waiting time by software from the time of restarting to confirm that the bus is free until the time to generate the start condition.

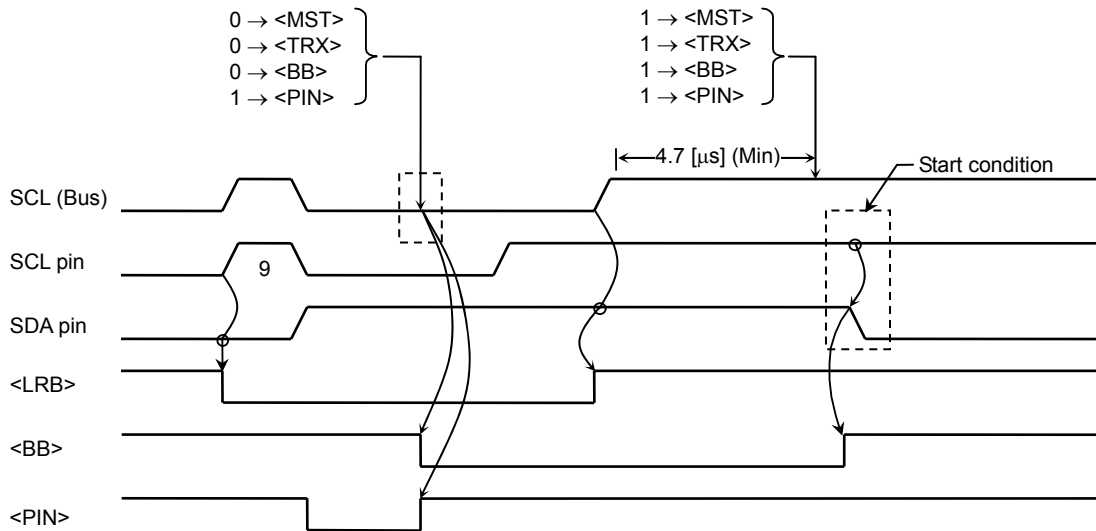


Figure 3.12.19 Timing Diagram for TMP91CW18A Restart

3.13 Analog/Digital Converter

The TMP91CW18A incorporates a 10-bit successive approximation-type analog/digital converter (AD converter) with 12-channel analog input.

Figure 3.13.1 is a block diagram of the AD converter. The 12-channel analog input pins (AN0 to AN7) are shared with the input-only port 5 (AN8 to AN11) are shared with the input-only port 4 can thus be used as general-purpose input port.

Note: When IDLE2, IDLE1 or STOP mode is selected, so as to reduce the power, with some timing the system may enter a standby mode even though the internal comparator is still enabled. Therefore be sure to check that AD converter operations are halted before a HALT instruction is executed.

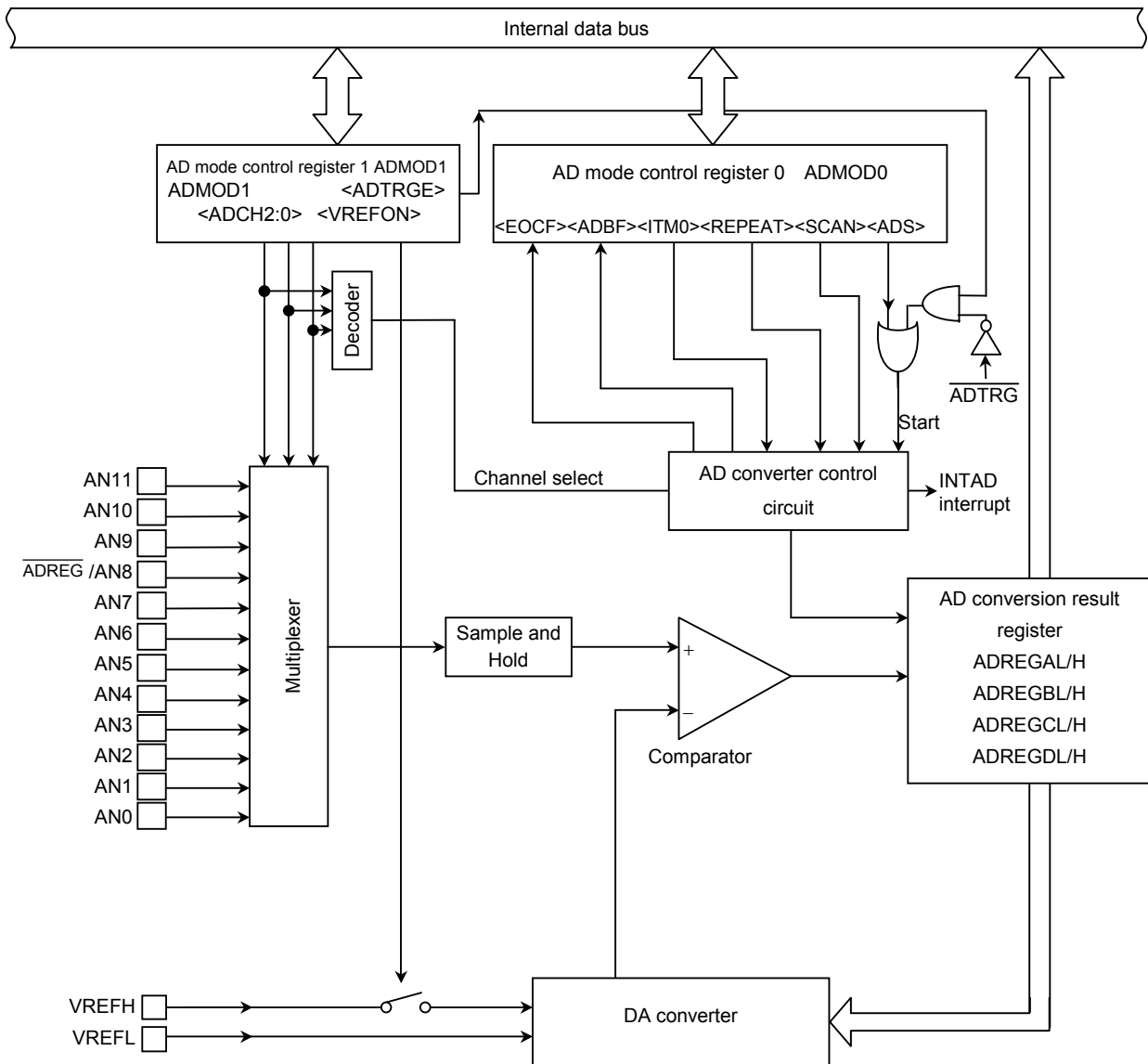


Figure 3.13.1 Block Diagram of AD Converter

3.13.1 Analog/Digital Converter Registers

The AD converter is controlled by the two AD mode control registers; ADMOD0 and ADMOD1. The eight AD conversion data upper and lower registers (ADREGAH/L, ADREGBH/L, ADREGCH/L and ADREGDH/L) store the results of AD conversion.

Figure 3.13.2 shows the registers related to the AD converter.

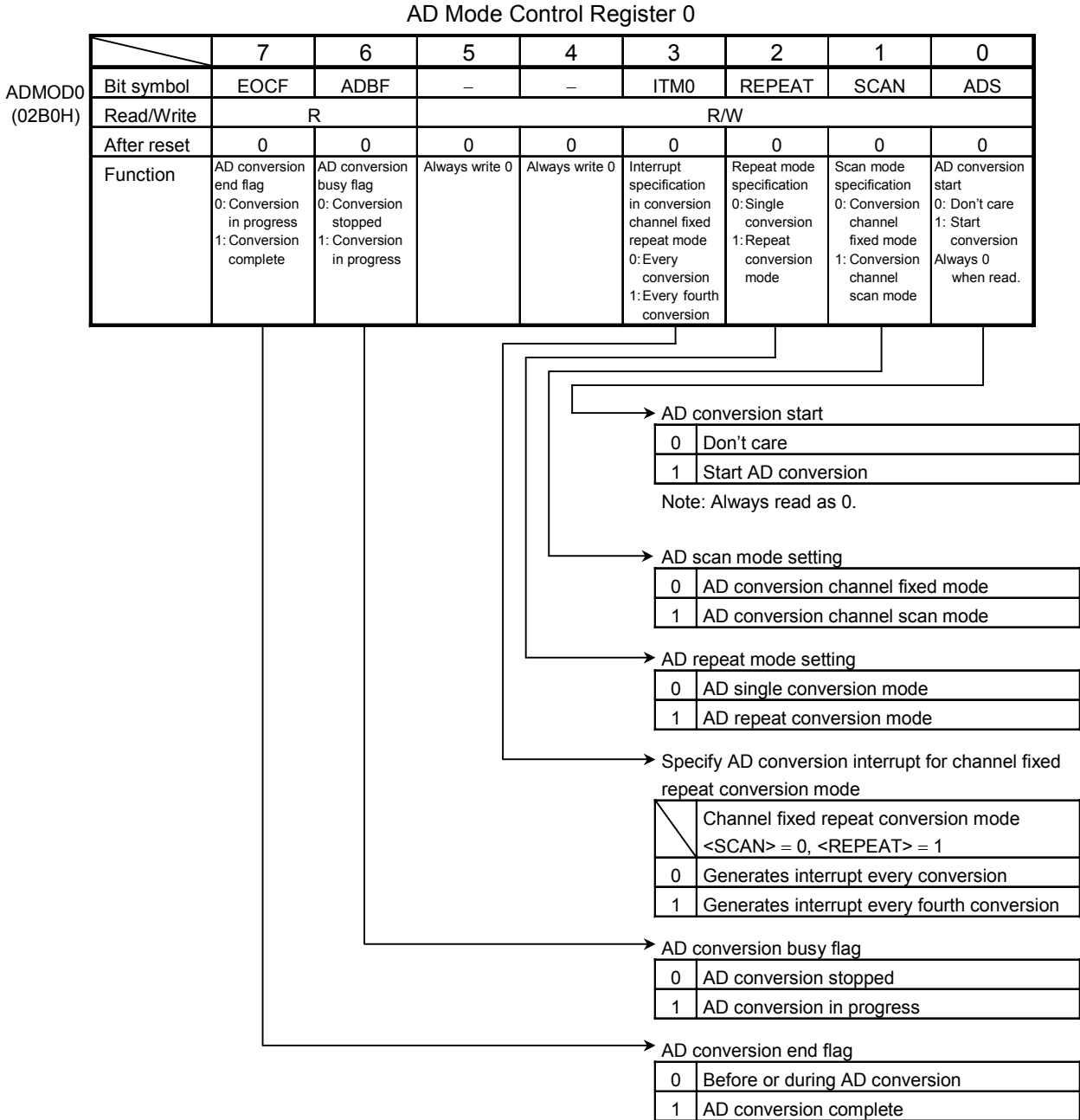
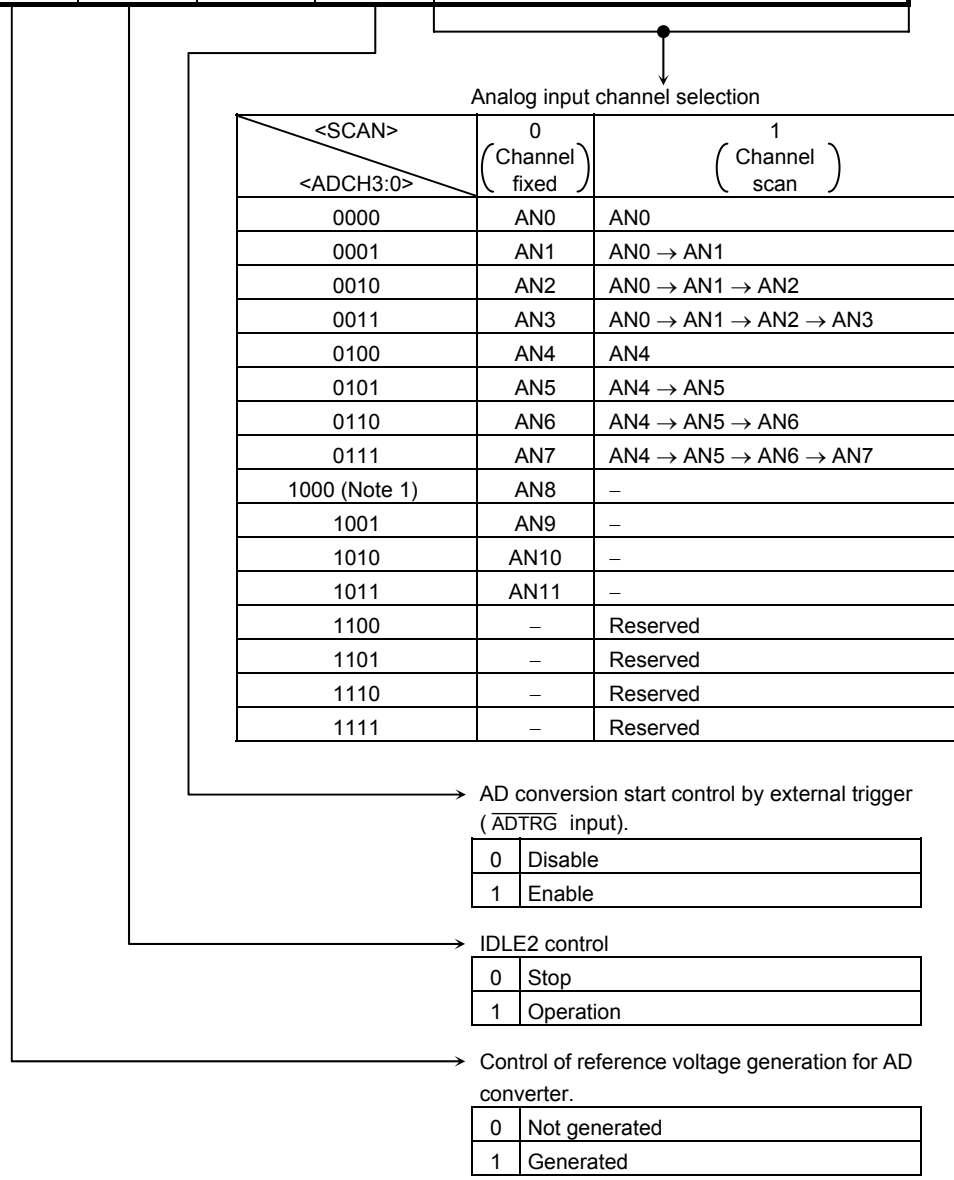


Figure 3.13.2 AD Converter Related Register

AD Mode Control Register 1

	7	6	5	4	3	2	1	0
Bit symbol	VREFON	I2AD		ADTRGE	ADCH3	ADCH2	ADCH1	ADCH0
Read/Write	R/W	R/W		R/W				
After reset	0	0		0	0	0	0	0
Function	VREF generation control 0: Not generated 1: Generated	IDLE2 0: Stop 1: Operation		AD external trigger start control. 0: Disable 1: Enable	Analog input channel selection.			



Before starting conversion (before writing 1 to ADMOD0<ADS>), set the <VREFON> bit to 1.

- Note 1: As pin AN8 also functions as the $\overline{\text{ADTRG}}$ input pin, do not set <ADCH3:0> = 1000, when using $\overline{\text{ADTRG}}$ with <ADTRGE> set to 1.
- Note 2: AN8 to AN11 dose not use scan mode. When using <ADCH3> set to 1, ADMODO<SCAN> clear to 0.

Figure 3.13.3 AD Converter Related Register

AD Conversion Data Low Register 0/4/8

	7	6	5	4	3	2	1	0
ADREGAL (02A0H)	ADRA1	ADRA0						ADRARF
Read/Write	R							R
After reset	Undefined							0
Function	Stores lower 2 bits of AD conversion result.							AD conversion data storage flag. 1: Conversion result stored

AD Conversion Data Upper Register 0/4/8

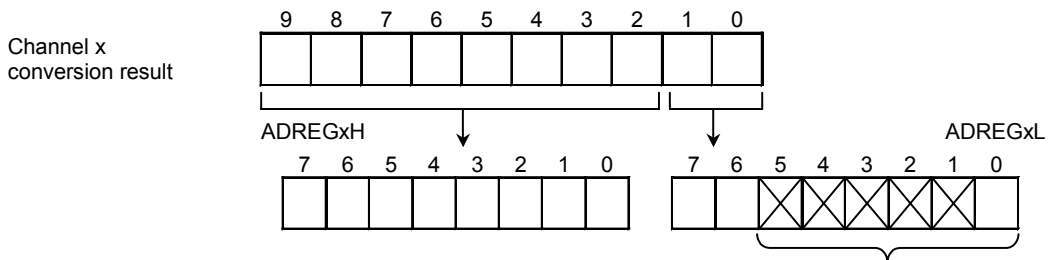
	7	6	5	4	3	2	1	0
ADREGAH (02A1H)	ADRA9	ADRA8	ADRA7	ADRA6	ADRA5	ADRA4	ADRA3	ADRA2
Read/Write	R							
After reset	Undefined							
Function	Stores upper 8 bits AD conversion result.							

AD Conversion Data Lower Register 1/5/9

	7	6	5	4	3	2	1	0
ADREGBL (02A2H)	ADRB1	ADRB0						ADBRF
Read/Write	R							R
After reset	Undefined							0
Function	Stores lower 2 bits of AD conversion result.							AD conversion result flag. 1: Conversion result stored

AD Conversion Data Upper Register 1/5/9

	7	6	5	4	3	2	1	0
ADREGBH (02A3H)	ADRB9	ADRB8	ADRB7	ADRB6	ADRB5	ADRB4	ADRB3	ADRB2
Read/Write	R							
After reset	Undefined							
Function	Stores upper 8 bits of AD conversion result.							



- Bits 5 to 1 are always read as 1.
- Bit0 is the AD conversion data storage flag <ADRxRF>. When the AD conversion result is stored, the flag is set to 1. When either of the registers (ADREGxH, ADREGxL) is read, the flag is cleared to 0.

Figure 3.13.4 AD Converter Related Registers

AD Conversion Result Lower Register 2/6/10

	7	6	5	4	3	2	1	0
ADREGCL (02A4H)	Bit symbol	ADRC1	ADRC0					ADRCRF
	Read/Write	R						R
	After reset	Undefined						0
	Function	Stores lower 2 bits of AD conversion result.						AD conversion data storage flag. 1: Conversion result stored

AD Conversion Data Upper Register 2/6/10

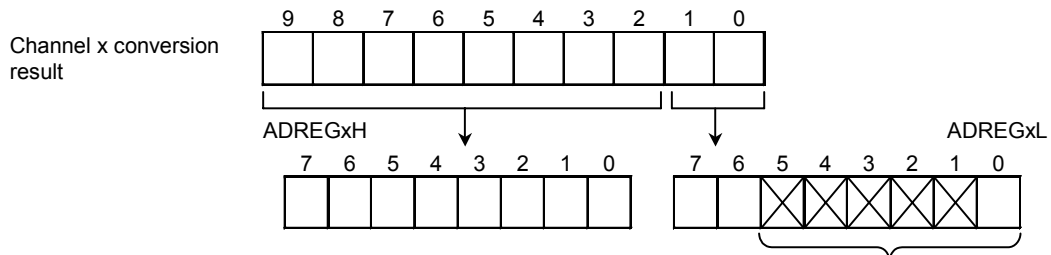
	7	6	5	4	3	2	1	0	
ADREGCH (02A5H)	Bit symbol	ADRC9	ADRC8	ADRC7	ADRC6	ADRC5	ADRC4	ADRC3	ADRC2
	Read/Write	R							
	After reset	Undefined							
	Function	Stores upper 8 bits of AD conversion result.							

AD Conversion Data Lower Register 3/7/11

	7	6	5	4	3	2	1	0
ADREGDL (02A6H)	Bit symbol	ARD1	ARD0					ARDRF
	Read/Write	R						R
	After reset	Undefined						0
	Function	Stores lower 2 bits of AD conversion result.						AD conversion data storage flag. 1: Conversion result stored

AD Conversion Result Upper Register 3/7/11

	7	6	5	4	3	2	1	0	
ADREGDH (02A7H)	Bit symbol	ARD9	ARD8	ARD7	ARD6	ARD5	ARD4	ARD3	ARD2
	Read/Write	R							
	After reset	Undefined							
	Function	Stores upper 8 bits of AD conversion result.							



- Bits 5 to 1 are always read as 1.
- Bit0 is the AD conversion data storage flag <ADRxRF>. When the AD conversion result is stored, the flag is set to 1. When either of the registers (ADREGxH, ADREGxL) is read, the flag is cleared to 0.

Figure 3.13.5 AD Converter Related Registers

3.13.2 Description of Operation

(1) Analog reference voltage

A high-level analog reference voltage is applied to the VREFH pin, a low-level analog reference voltage is applied to the VREFL pin. To perform AD conversion, the reference voltage, the difference between VREFH and VREFL are divided by 1024 using string resistance. The result of the division is then compared with the analog input voltage.

To turn off the switch between VREFH and VREFL, program a 0 to ADMOD1<VREFON> in AD mode control register 1. To start AD conversion in the OFF state, first write a 1 to ADMOD1<VREFON>, wait for 3 μ s until the internal reference voltage stabilizes (This is not related to f_c), then set ADMOD0<ADS> to 1.

(2) Analog input channel selection

The analog input channel selection varies depends on the operation mode of the AD converter.

- In analog input channel fixed mode (ADMOD0<SCAN> = 0)
Setting ADMOD1<ADCH3:0> selects one of the input pins AN0 to AN11 as the input channel.
- In analog input channel scan mode (ADMOD0<SCAN> = 1)
Setting ADMOD1<ADCH3:0> selects one of the eight scan modes.

Table 3.13.1 illustrates analog input channel selection in each operation mode.

On a reset, ADMOD0<SCAN> is cleared to 0 and ADMOD1<ADCH3:0> is initialized to 0000. Thus pin AN0 is selected as the fixed input channel. Pins not used as analog input channels can be used as standard input port pins.

Table 3.13.1 Analog Input Channel Selection

<ADCH3:0>	Channel Fixed <SCAN> = 0	Channel Scan <SCAN> = 1
0000	AN0	AN0
0001	AN1	AN0 → AN1
0010	AN2	AN0 → AN1 → AN2
0011	AN3	AN0 → AN1 → AN2 → AN3
0100	AN4	AN4
0101	AN5	AN4 → AN5
0110	AN6	AN4 → AN5 → AN6
0111	AN7	AN4 → AN5 → AN6 → AN7
1000	AN8	–
1001	AN9	–
1010	AN10	–
1011	AN11	–

(3) Starting AD conversion

To start AD conversion, write a 1 to ADMOD0<ADS> in AD mode control register 0 or ADMOD1<ADTRGE> in AD mode control register 1, pull the $\overline{\text{ADREG}}$ pin input from high to low. When AD conversion starts, the AD conversion busy flag ADMOD0<ADBF> will be set to 1, indicating that AD conversion is in progress.

Writing a 1 to ADMOD0<ADS> during AD conversion restarts conversion. At that time, to determine whether the AD conversion results have been preserved, check the value of the conversion data storage flag ADREGxL<ADR_xRF>.

During AD conversion, a falling edge input on the $\overline{\text{ADREG}}$ pin will be ignored.

(4) AD conversion modes and the AD conversion end interrupt

The four AD conversion modes are:

- Channel fixed single conversion mode
- Channel scan single conversion mode
- Channel fixed repeat conversion mode
- Channel scan repeat conversion mode

The ADMOD0<REPEAT> and ADMOD0<SCAN> settings in AD mode control register 0 determine the AD mode setting.

Completion of AD conversion triggers an INTAD AD conversion end interrupt request. Also, ADMOD0<EOCF> will be set to 1 to indicate that AD conversion has been completed.

1. Channel fixed single conversion mode

Setting ADMOD0<REPEAT> and ADMOD0<SCAN> to 00 selects conversion channel fixed single conversion mode.

In this mode data on one specified channel is converted once only. When the conversion has been completed, the ADMOD0<EOCF> flag is set to 1, ADMOD0<ADBF> is cleared to 0, and an INTAD interrupt request is generated.

2. Channel scan single conversion mode

Setting ADMOD0<REPEAT> and ADMOD0<SCAN> to 01 selects conversion channel scan single conversion mode.

In this mode data on the specified scan channels is converted once only. When scan conversion has been completed, ADMOD0<EOCF> is set to 1, ADMOD0<ADBF> is cleared to 0, and an INTAD interrupt request is generated.

3. Channel fixed repeat conversion mode

Setting ADMOD0<REPEAT> and ADMOD0<SCAN> to 10 selects conversion channel fixed repeat conversion mode.

In this mode data on one specified channel is converted repeatedly. When conversion has been completed, ADMOD0<EOCF> is set to 1 and ADMOD0<ADBF> is not cleared to 0 but held at 1. INTAD interrupt request generation timing is determined by the setting of ADMOD0<ITM0>.

Clearing <ITM0> to 0 generates an interrupt request every time an AD conversion is completed.

Setting <ITM0> to 1 generates an interrupt request on completion of every fourth conversion.

4. Channel scan repeat conversion mode

Setting ADMOD0<REPEAT> and ADMOD0<SCAN> to 11 selects conversion channel scan repeat conversion mode.

In this mode data on the specified scan channels is converted repeatedly. When each scan conversion has been completed, ADMOD0<EOCF> is set to 1 and an INTAD interrupt request is generated. ADMOD0<ADBF> is not cleared to 0 but held at 1.

To stop conversion in a repeat conversion mode (e.g., in cases 3. and 4.), program a 0 to ADMOD0<REPEAT>. After the current conversion has been completed, the repeat conversion mode terminates and ADMOD0<ADBF> is cleared to 0.

Switching to a halt state (IDLE2 mode with ADMOD1<I2AD> cleared to 0, IDLE1 mode or STOP mode) immediately stops operation of the AD converter even when AD conversion is still in progress. In repeat conversion modes (e.g., in cases 3. and 4.), when the halt is released, conversion restarts from the beginning. In single conversion modes (e.g., in cases 1. and 2.), conversion does not restart when the halt is released. (The converter remains stopped.)

Table 3.13.2 shows the relationship between the AD conversion modes and interrupt requests.

Table 3.13.2 Relationship between AD Conversion Modes and Interrupt Requests

Mode	Interrupt Request Generation	ADMOD0		
		<ITM0>	<REPEAT>	<SCAN>
Channel fixed single conversion mode	After completion of conversion	X	0	0
Channel scan single conversion mode	After completion of scan conversion	X	0	1
Channel fixed repeat conversion mode	Every conversion	0	1	0
	Every fourth conversion	1		
Channel scan repeat conversion mode	After completion of every scan conversion	X	1	1

X: Don't care

(5) AD conversion time

84 states (6.7 μ s at $f_{PPH} = 25$ MHz) are required for the AD conversion of one channel.

(6) Storing and reading the results of AD conversion

The AD conversion data upper and lower registers (ADREGAH/L to ADREGDH/L) store the results of AD conversion. (ADREGAH/L to ADREGDH/L are read only registers.)

In channel fixed repeat conversion mode, the conversion results are stored successively in registers ADREGAH/L to ADREGDH/L. In other modes the AN0 and AN4, AN5 and AN9, AN2, and AN6, AN10 and AN10, AN7 and AN11 conversion results are stored in ADREGAH/L, ADREGBH/L, ADREGCH/L and ADREGDH/L respectively.

Table 3.13.3 shows the correspondence between the analog input channels and the registers which are used to hold the results of AD conversion.

Table 3.13.3 Correspondence between Analog Input Channels and AD Conversion Result Registers

Analog Input Channel (Port A)	AD Conversion Result Register	
	Conversion Modes Other than at Right	Channel Fixed Repeat Conversion Mode ($\langle ITM0 \rangle = 1$)
AN0	ADREGAH/L	
AN1	ADREGBH/L	
AN2	ADREGCH/L	
AN3	ADREGDH/L	
AN4	ADREGAH/L	
AN5	ADREGBH/L	
AN6	ADREGCH/L	
AN7	ADREGDH/L	
AN8	ADREGAH/L	
AN9	ADREGBH/L	
AN10	ADREGCH/L	
AN11	ADREGDH/L	

$\langle ADRxRF \rangle$, bit0 of the AD conversion data lower register is used as the AD conversion data storage flag. The storage flag indicates whether the AD conversion result register has been read or not. When a conversion result is stored in the AD conversion result register, the flag is set to 1. When either of the AD conversion result registers (ADREGxH or ADREGxL) is read, the flag is cleared to 0.

Reading the AD conversion result also clears the AD conversion end flag $ADMOD0 \langle EOCF \rangle$ to 0.

Setting example:

- Convert the analog input voltage on the AN3 pin and write the result to memory address 0800H using the AD interrupt (INTAD) processing routine.

Main routine:

	7	6	5	4	3	2	1	0		
INTE0AD	←	X	1	0	0	-	-	-	Enable INTAD and set it to interrupt level 4.	
ADMOD1	←	1	1	X	X	0	0	1	1	Set pin AN3 to be the analog input channel.
ADMOD0	←	X	X	0	0	0	0	0	1	Start conversion in channel fixed single conversion mode.

Interrupt routine processing example:

WA	←	ADREGD	Read value of ADREGDL and ADREGDH into 16-bit general-purpose register WA.
WA	>>	6	Shift contents read into WA 6 times to right and 0-fill upper bits.
(0800H)	←	WA	Write contents of WA to memory address 0800H.

- This example repeatedly converts the analog input voltages on the three pins AN0, AN1 and AN2, using channel scan repeat conversion mode.

INTE0AD	←	X	0	0	0	-	-	-	Disable INTAD.	
ADMOD1	←	1	1	X	X	0	0	1	0	Set pins AN0 to AN2 to be the analog input channels.
ADMOD0	←	X	X	0	0	0	1	1	1	Start conversion in channel scan repeat conversion mode.

X: Don't care, -: No change

3.14 Watchdog Timer (Runaway detection timer)

The TMP91CW18A features a watchdog timer for detecting runaway.

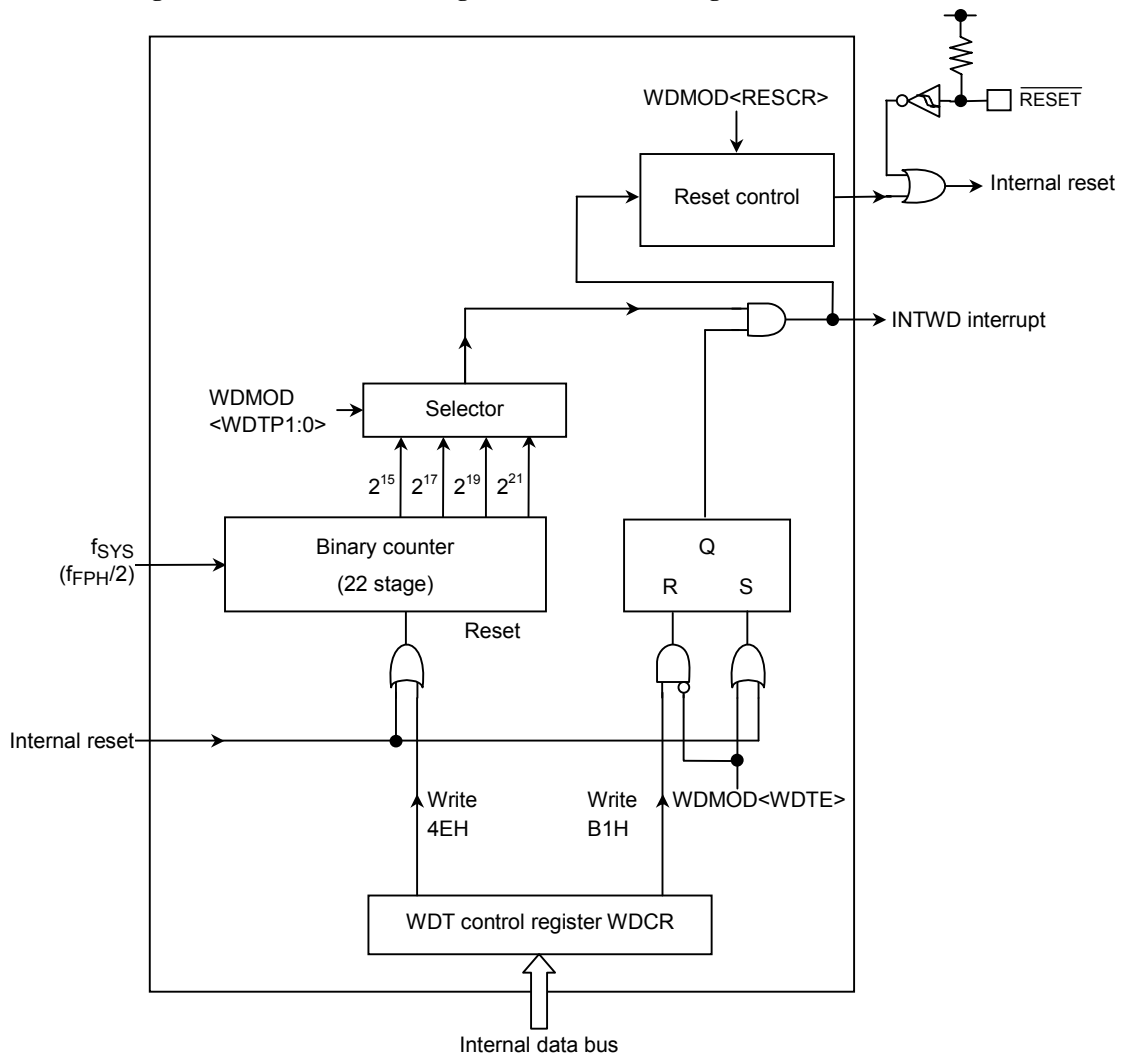
The watchdog timer (WDT) is used to return the CPU to normal state when it detects that the CPU has started to malfunction (Runaway) due to causes such as noise. When the watchdog timer detects a malfunction, it generates a non-maskable interrupt INTWD to notify the CPU of the malfunction.

Connecting the watchdog timer output to the reset pin internally forces a reset. (The level of external $\overline{\text{RESET}}$ pin is not changed.)

3.14.1 Configuration

Note: It needs to care designing the total machine set, because watchdog timer can't operate completely by external noise.

Figure 3.14.1 is a block diagram of the watchdog timer (WDT).



Note: It needs to care designing the total machine set, because watchdog timer can't operate completely by external noise.

Figure 3.14.1 Block Diagram of Watchdog Timer

The watchdog timer consists of a 22-stage binary counter which uses the system clock (f_{SYS}) as the input clock. The binary counter can output $f_{SYS}/2^{15}$, $f_{SYS}/2^{17}$, $f_{SYS}/2^{19}$ and $f_{SYS}/2^{21}$.

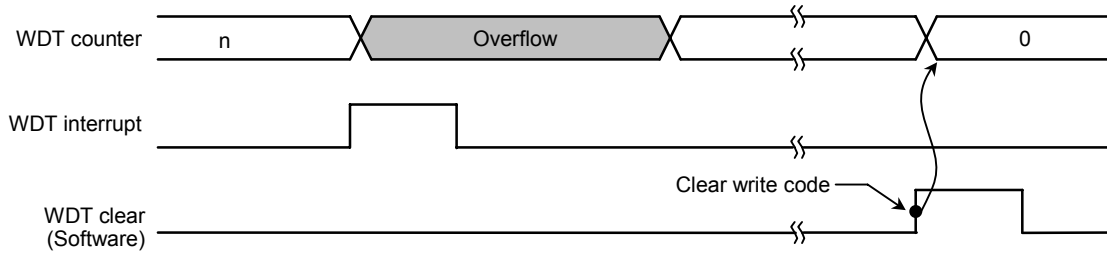


Figure 3.14.2 Normal Mode

The runaway is detected when an overflow occurs, and the watchdog timer can reset device. In this case, the reset time will be between 22 and 29 states (28.2 to 37.1 μs at $f_{PPH} = 25MHz$, $f_{OSCH} = 1 \text{ state}$) is $f_{PPH}/2$, where f_{PPH} is generated by dividing the high-speed oscillator clock (f_{OSCH}) by sixteen through the clock gear function.

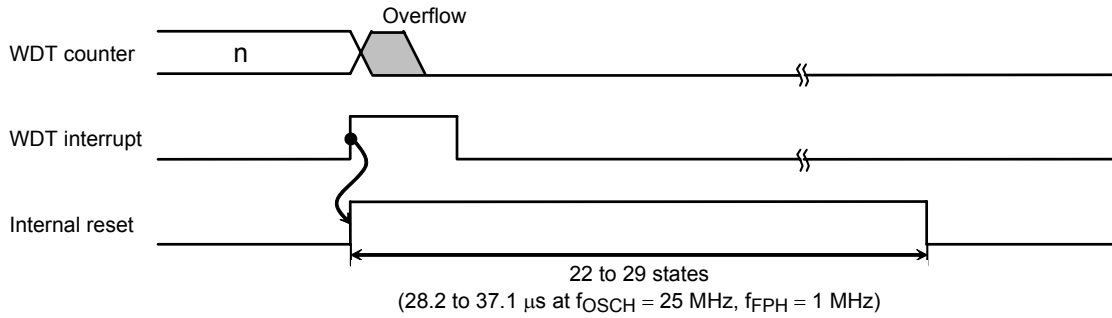


Figure 3.14.3 Reset Mode

3.14.2 Control Registers

The watchdog timer WDT is controlled by two control registers WDMOD and WDCR.

(1) Watchdog timer mode register (WDMOD)

1. Setting the detection time for the watchdog timer in <WDTP>

This 2-bit register is used for setting the watchdog timer interrupt time used when detecting runaway. On a reset this register is initialized to WDMOD<WDTP1:0> = 00.

The detection times for WDT are shown in Figure 3.14.4.

2. Watchdog timer enable/disable control register <WDTE>

On a reset WDMOD<WDTE> is initialized to 1, enabling the watchdog timer.

To disable the watchdog timer, it is necessary to clear this bit to 0 and to write the disable code (B1H) to the Watchdog timer control register WDCR. This makes it difficult for the watchdog timer to be disabled by runaway.

However, it is possible to return the watchdog timer from the disabled state to the enabled state merely by setting <WDTE> to 1.

3. Watchdog timer out reset connection <RESCR>

This register is used to connect the output of the watchdog timer with the RESET terminal internally. Since WDMOD<RESCR> is initialized to 0 on a reset, a reset by the watchdog timer will not be performed.

(2) Watchdog timer control register (WDCR)

This register is used to disable and clear the binary counter for the watchdog timer.

Disable control the watchdog timer can be disabled by clearing WDMOD<WDTE> to 0 and then writing the disable code (B1H) to the WDCR register.

```
WDMOD    ← 0 - - - - -      Clear WDMOD<WDTE> to 0.
WDCR     ← 1 0 1 1 0 0 0 1  Write the disable code (B1H).
```

- Enable control

Set WDMOD<WDTE> to 1.

- Watchdog timer clear control

To clear the binary counter and cause counting to resume, write the clear code (4EH) to the WDCR register.

```
WDCR     ← 0 1 0 0 1 1 1 0  Write the clear code (4EH).
```

Note1: If it is used disable control, set the disable code (B1H) to WDCR after write the clear code (4EH) once. (Please refer to setting example.)

Note2: If it is changed Watchdog timer setting, change setting after set to disable condition once.

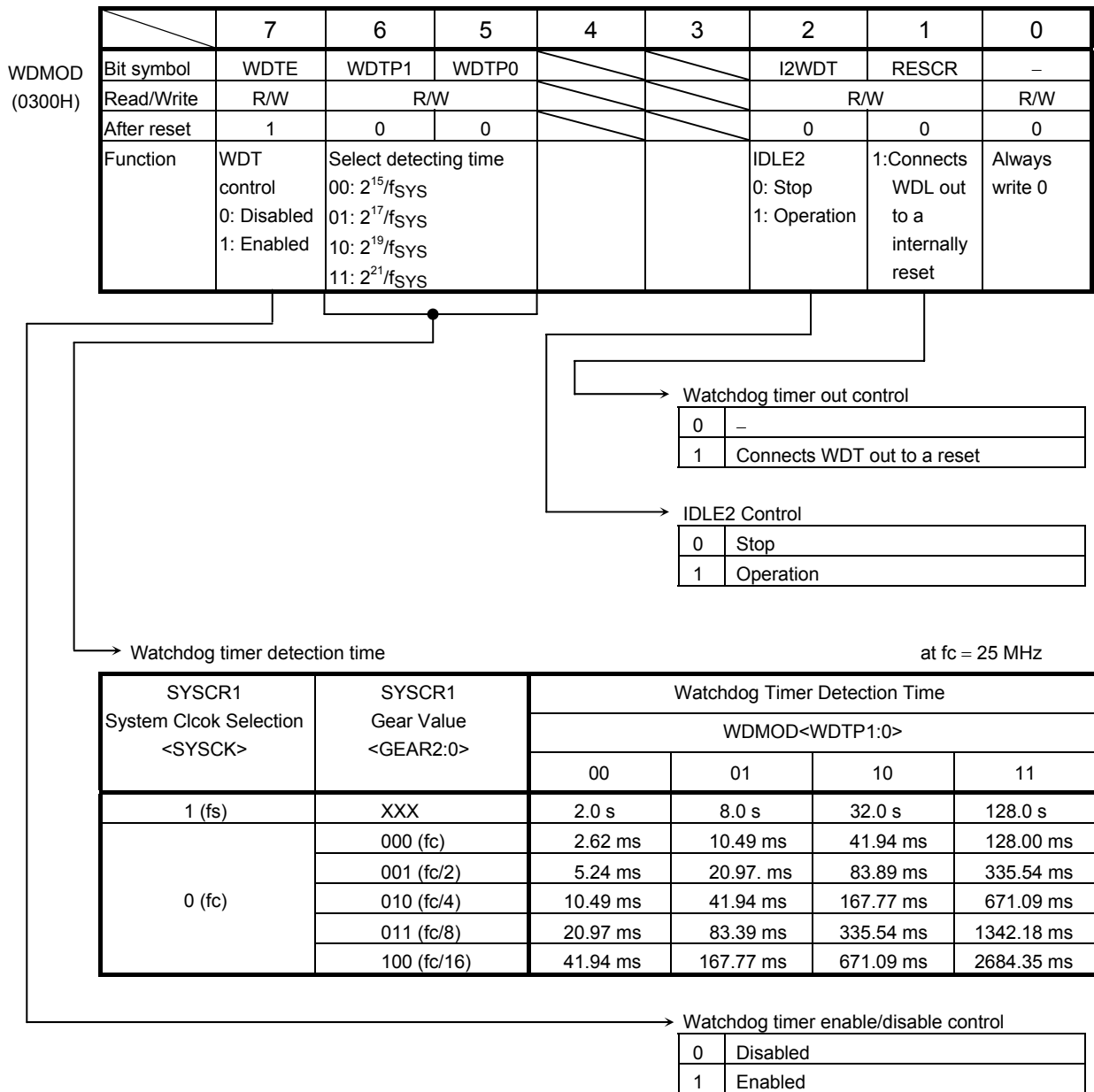


Figure 3.14.4 Watchdog Timer Mode Register

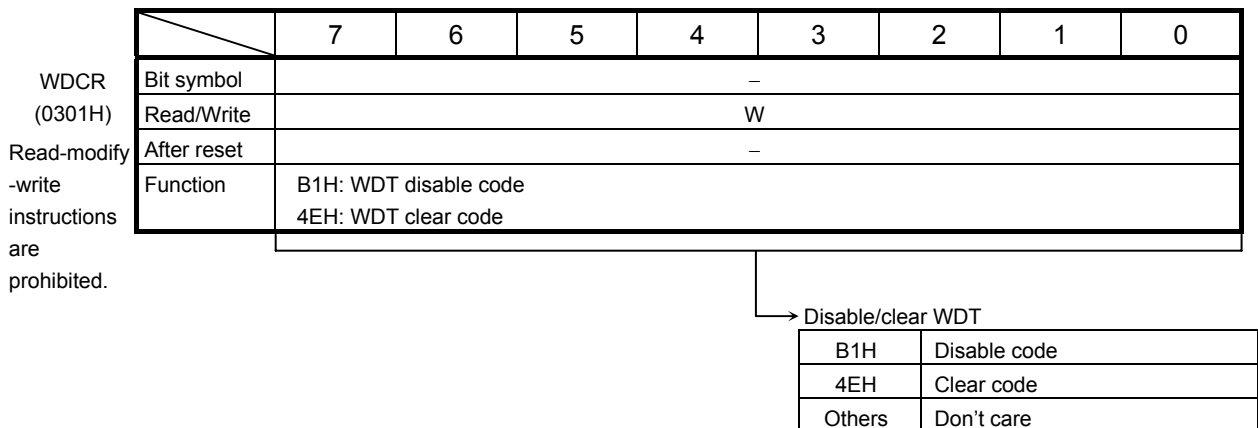


Figure 3.14.5 Watchdog Timer Control Register

3.14.3 Operation

The watchdog timer generates an INTWD interrupt when the detection time set in the WDMOD<WDTP1:0> has elapsed. The watchdog timer must be zero-cleared in software before an INTWD interrupt will be generated. If the CPU malfunctions (e.g., if runaway occurs) due to causes such as noise, but does not execute the instruction used to clear the binary counter, the binary counter will overflow and an INTWD interrupt will be generated. The CPU will detect malfunction (Runaway) due to the INTWD interrupt and in this case it is possible to return to the CPU to normal operation by means of an anti-mulfunction program. By connecting the watchdog timer out pin to a peripheral device's reset input, the occurrence of a CPU malfunction can also be relayed to other devices.

The watchdog timer does not operate in IDLE1 or STOP mode, as the binary counter continues counting during bus release (when $\overline{\text{BUSAK}}$ goes low).

When the device is in IDLE2 mode, the operation of WDT depends on the WDMOD<I2WDT> setting. Ensure that WDMOD<I2WDT> is set before the device enters IDLE2 mode.

Example: 1. Clear the binary counter.

WDCR ← 0 1 0 0 1 1 1 0 Write the clear code (4EH).

2. Set the watchdog timer detection time to $2^{17}/f_{\text{SYS}}$.

WDMOD ← 1 0 1 - - - - -

3. Disable the watchdog timer.

WDMOD ← 0 - - - - - Clear WDTE to 0.

WDCR ← 1 0 1 1 0 0 0 1 Write the disable code (B1H).

Note: The watchdog timer cannot operate by disturbance noise in some case. Take care when design the device.

4. Electrical Characteristics

4.1 Maximum Ratings

Parameter	Symbol	Rating	Unit
Power supply voltage	V_{CC}	-0.5 to 6.5	V
Input voltage	V_{IN}	-0.5 to $V_{CC} + 0.5$	
Output current	IOL	2	mA
Output current	IOH	-2	
Output current (Total)	ΣIOL	80	
Output current (Total)	ΣIOH	-80	
Power dissipation ($T_a = 70^\circ\text{C}$)	PD	600	
Soldering temperature (10 s)	TSOLDER	260	°C
Storage temperature	TSTG	-65 to 150	
Operating temperature	TOPR	-30 to 70	

Note: The maximum ratings are rated values which must not be exceeded during operation, even for an instant. Any one of the ratings must not be exceeded. If any maximum rating is exceeded, a device may break down or its performance may be degraded, causing it to catch fire or explode resulting in injury to the user. Thus, when designing products which include this device, ensure that no maximum rating value will ever be exceeded.

4.2 DC Characteristics (1/2)

Parameter	Symbol	Condition	Min	Typ. (Note1)	Max	Unit
Power supply voltage ($AV_{CC} = DV_{CC}$ $AV_{SS} = DV_{SS} = 0\text{ V}$)	V_{CC}	$f_c = 8$ to 25 MHz	4.5		5.5	V
Input low voltage	P00 to P17 (AD0 to AD15)	V_{IL}	$V_{CC} \geq 4.5\text{ V}$	-0.3	0.8	V
	P20 to P87	V_{IL1}			$0.3 V_{CC}$	
	RESET, NMI	V_{IL2}			$0.25 V_{CC}$	
	AM0, AM 1	V_{IL3}			0.3	
	X1	V_{IL4}			$0.2 V_{CC}$	
Input high voltage	P00 to P17 (AD0 to AD15)	V_{IH}	$V_{CC} = 4.5$ to 5.5 V		$0.7 V_{CC}$	$V_{CC} + 0.3$
	P20 to P87	V_{IH1}			$0.7 V_{CC}$	
	RESET, NMI	V_{IH2}			$0.75 V_{CC}$	
	AM0 to AM 1	V_{IH3}			$V_{CC} - 0.3$	
	X1	V_{IH4}			$0.8 V_{CC}$	
Output low voltage	VOL	IOL = 1.6 mA ($V_{CC} = 4.5$ to 5.5 V)			0.45	V
Output high voltage	VOH	IOH = -400 μA ($V_{CC} = 5.0\text{ V} \pm 10\%$)	$0.8V_{CC}$			

Note: Typical values are for when $T_a = 25^\circ\text{C}$ and $V_{CC} = 5.0\text{ V}$ unless otherwise noted.

DC Characteristics (2/2)

Parameter	Symbol	Condition	Min	Typ. (Note 1)	Max	Unit
Input leakage current	ILI	$0.0 \leq V_{IN} \leq V_{CC}$		0.02	± 1	μA
Output leakage current	ILO	$0.2 \leq V_{IN} \leq V_{CC} - 0.2$		0.05	± 10	
Power down voltage (at STOP, RAM back up)	VSTOP	$V_{IL2} = 0.2 V_{CC}$, $V_{IH2} = 0.8 V_{CC}$	2.0		5.5	V
RESET pull-up resistor	RRST	$V_{CC} = 5 \text{ V} \pm 10\%$	40		200	$\text{k}\Omega$
Pin capacitance	CIO	$f_c = 1 \text{ MHz}$			10	pF
Schmitt width $\overline{\text{RESET}}$, $\overline{\text{NMI}}$	VTH		0.4	1.0		V
Programmable pull-up resistor	RKH	$V_{CC} = 5 \text{ V} \pm 10\%$	40		200	$\text{k}\Omega$
NORMAL (Note 2)	I_{CC}	$V_{CC} = 5 \text{ V} \pm 10\%$ $f_c = 25 \text{ MHz}$ (Typ. $V_{CC} = 5.0 \text{ V}$)		22.5	35.0	mA
IDLE2				8.6	13.0	
IDLE1				3.5	7.0	
STOP			$T_a \leq 70^\circ\text{C}$	$V_{CC} = 4.5$ to 5.5 V		0.2

Note 1: Typical values are for when $T_a = 25^\circ\text{C}$ and $V_{CC} = 5.0 \text{ V}$ unless otherwise noted.

Note 2: I_{CC} measurement conditions (NORMAL):

All functions are operational. Output pins are open and input pins are fixed.

4.3 AC Characteristics

(1) $V_{CC} = 5.0 \text{ V} \pm 10 \%$

No.	Parameter	Symbol	Variable		$f_{\text{FPH}} = 25 \text{ MHz}$		Unit
			Min	Max	Min	Max	
1	f_{FPH} period (= x)	t_{FPH}	40	31250	40		ns
2	A0 to A15 valid \rightarrow ALE fall	t_{AL}	$0.5x - 15$		5		ns
3	ALE fall \rightarrow A0 to A15 hold	t_{LA}	$0.5x - 15$		5		ns
4	ALE high width	t_{LL}	$x - 20$		20		ns
5	ALE fall \rightarrow $\overline{\text{RD}}$ / $\overline{\text{WR}}$ fall	t_{LC}	$0.5x - 20$		0		ns
6	$\overline{\text{RD}}$ rise \rightarrow ALE rise	t_{CLR}	$0.5x - 15$		5		ns
7	$\overline{\text{WR}}$ rise \rightarrow ALE rise	t_{CLW}	$x - 15$		25		ns
8	A0 to A15 valid \rightarrow $\overline{\text{RD}}$ / $\overline{\text{WR}}$ fall	t_{ACL}	$x - 25$		15		ns
9	A0 to A23 valid \rightarrow $\overline{\text{RD}}$ / $\overline{\text{WR}}$ fall	t_{ACH}	$1.5x - 50$		10		ns
10	$\overline{\text{RD}}$ rise \rightarrow A0 to A23 hold	t_{CAR}	$0.5x - 20$		0		ns
11	$\overline{\text{WR}}$ rise \rightarrow A0 to A23 hold	t_{CAW}	$x - 20$		20		ns
12	A0 to A15 valid \rightarrow D0 to D15 input	t_{ADL}		$3.0x - 45$		75	ns
13	A0 to A23 valid \rightarrow D0 to D15 input	t_{ADH}		$3.5x - 35$		105	ns
14	$\overline{\text{RD}}$ fall \rightarrow D0 to D15 input	t_{RD}		$2.0x - 40$		40	ns
15	$\overline{\text{RD}}$ low width	t_{RR}	$2.0x - 20$		60		ns
16	$\overline{\text{RD}}$ rise \rightarrow D0 to D15 hold	t_{HR}	0		0		ns
17	$\overline{\text{RD}}$ rise \rightarrow A0 to A15 output	t_{RAE}	$x - 15$		25		ns
18	$\overline{\text{WR}}$ low width	t_{WW}	$1.5x - 20$		40		ns
19	D0 to D15 Valid \rightarrow $\overline{\text{WR}}$ rise	t_{DW}	$1.5x - 50$		10		ns
20	$\overline{\text{WR}}$ rise \rightarrow D0 to D15 hold	t_{WD}	$x - 15$		25		ns
21	A0 to A23 valid \rightarrow $\overline{\text{WAIT}}$ input $\left[\begin{smallmatrix} (1+N) \text{ WAIT} \\ \text{mode} \end{smallmatrix} \right]$	t_{AWH}		$3.5x - 90$		50	ns
22	A0 to A15 valid \rightarrow $\overline{\text{WAIT}}$ input $\left[\begin{smallmatrix} (1+N) \text{ WAIT} \\ \text{mode} \end{smallmatrix} \right]$	t_{AWL}		$3.0x - 80$		40	ns
23	$\overline{\text{RD}}$ / $\overline{\text{WR}}$ fall \rightarrow $\overline{\text{WAIT}}$ hold $\left[\begin{smallmatrix} (1+N) \text{ WAIT} \\ \text{mode} \end{smallmatrix} \right]$	t_{CW}	$2.0x + 0$		80		ns
24	A0 to A23 valid \rightarrow Port input	t_{APH}		$3.5x - 120$		20	ns
25	A0 to A23 valid \rightarrow Port hold	t_{APH2}	$3.5x$		140		ns
26	A0 to A23 valid \rightarrow Port valid	t_{AP}		$3.5x + 100$		319	ns

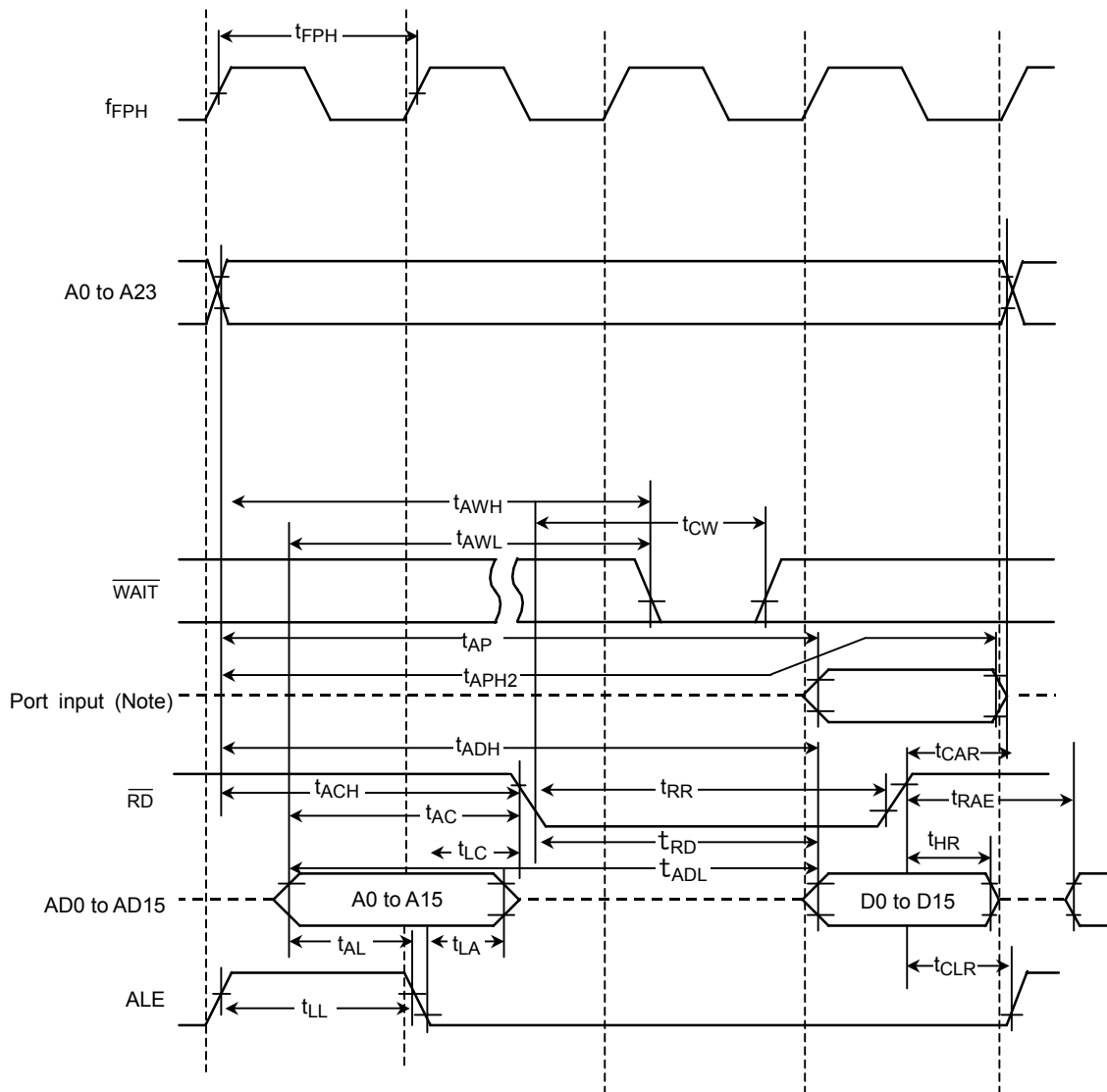
AC measuring conditions

- Output level: High = $2.2 V_{CC}$ / Low = $0.8 V_{CC}$, $CL = 50 \text{ pF}$
- Input level: High = $2.4 V_{CC}$ / Low = $0.45 V_{CC}$ (AD0 to AD15)
High = $0.8 V_{CC}$ / Low = $0.2 V_{CC}$ (except AD0 to AD15)

Note: Symbol x in the above table means the period of clock f_{FPH} , it's half period of the system clock f_{SYS} for CPU core. The period of f_{FPH} depends on the clock gear setting or the selection of high/low oscillator frequency.

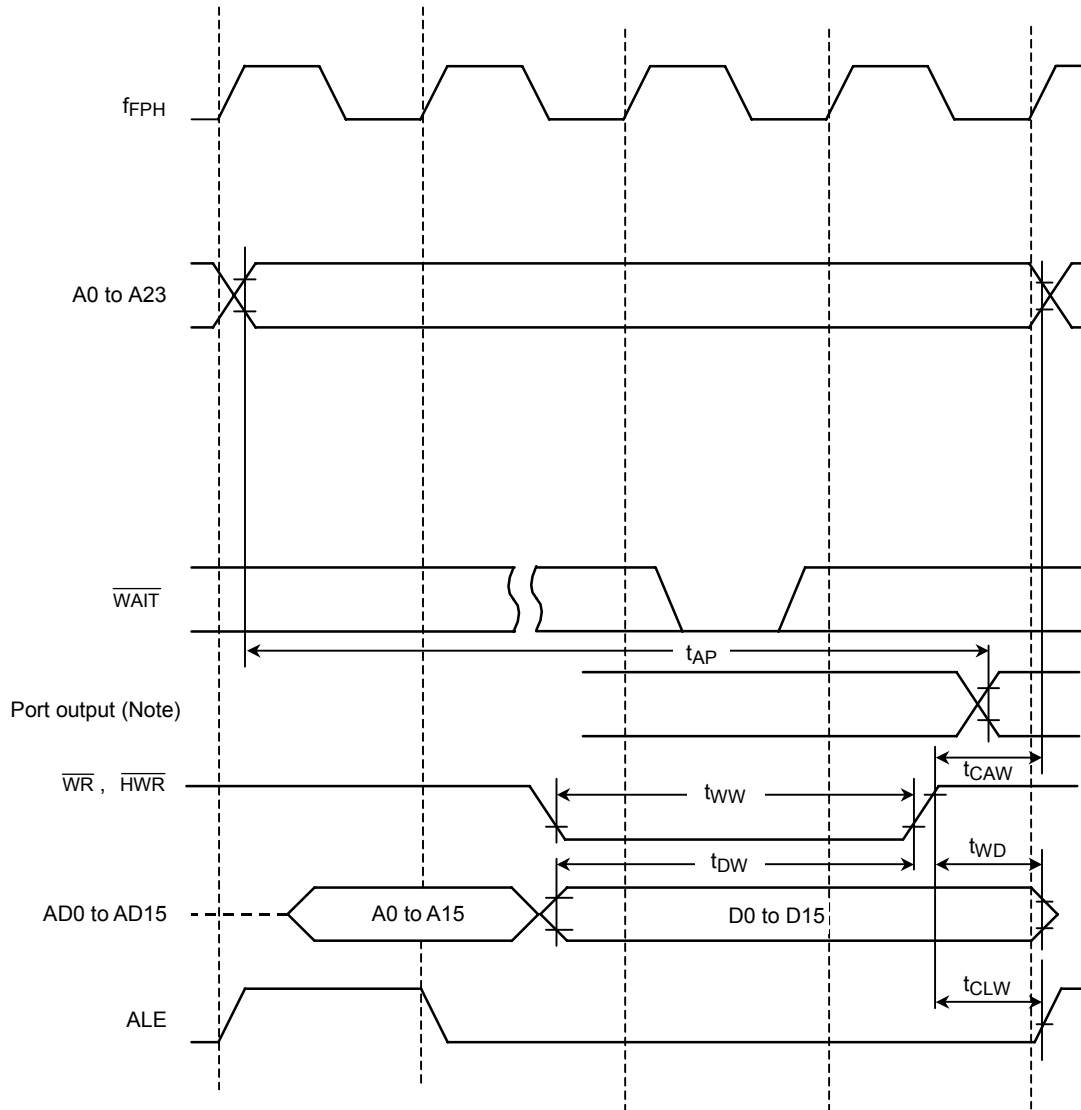
(2) $V_{CC} = 5.0 \text{ V} \pm 10 \%$

1. Read cycle



Note: Since the CPU accesses the internal area to read data from a port, the control signals of external pins such as \overline{RD} are not enabled. Therefore, the above waveform diagram should be regarded as depicting internal operation. Please also note that the timing and AC characteristics of port input/output shown above are typical representation. For details, contact your local Toshiba sales representative.

2. Write cycle



Note: Since the CPU accesses the internal area to write data to a port, the control signals of external pins such as \overline{WR} are not enabled. Therefore, the above waveform diagram should be regarded as depicting internal operation. Please also note that the timing and AC characteristics of port input/output shown above are typical representation. For details, contact your local Toshiba sales representative.

4.4 AD Conversion Characteristics

$$AV_{CC} = V_{CC}, AV_{SS} = V_{SS}$$

Parameter	Symbol	Condition	Min	Typ.	Max	Unit
Analog input voltage range (+)	VREFH	$V_{CC} = 5\text{ V} \pm 10\%$	$V_{CC} - 1.5\text{ V}$	V_{CC}	V_{CC}	V
Analog input voltage range (-)	VREFL	$V_{CC} = 5\text{ V} \pm 10\%$	V_{SS}	V_{SS}	$V_{SS} + 0.2\text{ V}$	
Analog input voltage range	VAIN		VREFL		VREFH	
Analog input voltage range <VREFON> = 1	IREF (VREFL = 0 V)	$V_{CC} = 5\text{ V} \pm 10\%$		1.44	2.00	mA
<VREFON> = 0		$V_{CC} = 5\text{ V} \pm 10\%$		0.02	5.0	μA
Error (Not including quantizing errors)	-	$V_{CC} = 5\text{ V} \pm 10\%$		± 1.0	± 4.0	LSB

Note 1: $1\text{ LSB} = (V_{REFH} - V_{REFL})/1024\text{ [V]}$

Note 2: The operation above is guaranteed for $f_{FPH} \geq 4\text{ MHz}$.

Note 3: The value for I_{CC} includes the current which flows through the AV_{CC} pin.

4.5 Event Counter (TA0IN, TA4IN, TB0IN0, TB0IN1, TB1IN0, TB1IN1)

Parameter	Symbol	Variable		25 MHz		Unit
		Min	Max	Min	Max	
Clock period	t_{VCK}	$8X + 100$		420		ns
Clock low level width	t_{VCKL}	$4X + 40$		200		ns
Clock high level width	t_{VCKH}	$4X + 40$		200		ns

4.6 Interrupt, Capture

(1) $\overline{\text{NMI}}$, INT0 to INT4 interrupts

Parameter	Symbol	Variable		25 MHz		Unit
		Min	Max	Min	Max	
$\overline{\text{NMI}}$, INT0 to INT4 low-level width	t_{INTAL}	$4X + 40$		200		ns
$\overline{\text{NMI}}$, INT0 to INT4 high-level width	t_{INTAH}	$4X + 40$		200		ns

(2) INT5 to INT6 interrupts, capture

The INT5 to INT6 input width depends on the system clock and prescaler clock settings.

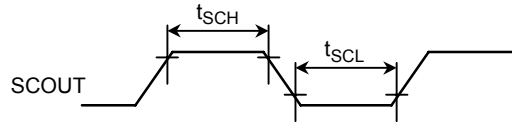
System Clock Selected <SYSCK>	Prescaler Clock Selected <PRCK1:0>	t_{INTBL} (INT5 to INT6 low-level width)		t_{INTBH} (INT5 to INT6 high-level width)		Unit
		Variable	$f_{FPH} = 25\text{ MHz}$	Variable	$f_{FPH} = 25\text{ MHz}$	
		Min	Max	Min	Max	
0 (fc)	00 (f_{FPH})	$8X + 100$	420	$8X + 100$	420	ns
	10 ($fc/16$)	$128Xc + 0.1$	5.22	$128Xc + 0.1$	5.22	μs

Xc: Period of clock fc

4.7 SCOUT Pin AC Characteristics

Parameter	Symbol	Variable		25 MHz		Condition	Unit
		Min	Max	Min	Max		
Low-level width	t_{SCH}	$0.5T - 15$		5		$V_{CC} = 5 V \pm 10\%$	ns
High-level width	t_{SCL}	$0.5T - 15$		5		$V_{CC} = 5 V \pm 10\%$	ns

T: Period of SCOUT



5. Table of SFRs

(SFR: Special function register)

The SFRs include the I/O ports and peripheral control registers allocated to the 4-Kbyte address space from 000000H to 000FFFH.

- (1) I/O port
- (2) I/O port control
- (3) Interrupt control
- (4) Wait control
- (5) Clock gear
- (6) 8-bit timer
- (7) 16-bit timer
- (8) UART
- (9) I²C bus/SIO
- (10) I²C bus 1
- (11) I²C bus 2
- (12) AD converter
- (13) Watchdog timer

Table layout

Symbol	Name	Address	7	6			1	0

→ Bit symbol

→ Read/write

→ Initial value after reset

→ Remarks

Note: "Prohibit RMW" in the table means that you cannot use RMW instructions on these register.

Example: When setting bit0 only of the register PxCR, the instruction "SET 0, (PxCR)" cannot be used. The LD (Transfer) instruction must be used to write all eight bits.

Read/write

- R/W: Both read and write are possible.
- R: Only read is possible.
- W: Only write is possible.
- W*: Both read and write are possible (when this bit is read as1)
- Prohibit RMW: Read-modify-write instructions are prohibited. (The EX, ADD, ADC, BUS, SBC, INC, DEC, AND, OR, XOR, STCF, RES, SET, CHG, TSET, RLC, RRC, RL, RR, SLA, SRA, SLL, SRL, RLD and RRD instruction are read-modify-write instructions.)
- R/W*: Read-modify-write instruction is prohibited when controlling the pull-up resistor.

I/O Register Address Map

[1] PORT

Address	Register	Address	Register	Address	Register
0000H	P0	0010H		0020H	
1H	P1	1H		1H	
2H	P0CR	2H	P6	2H	
3H		3H	P7	3H	
4H	P1CR	4H	P6CR	4H	
5H	P1FC	5H	P6FC	5H	
6H	P2	6H	P7CR	6H	
7H	P3	7H	P7FC	7H	
8H	P2CR	8H	P8	8H	
9H	P2FC	9H		9H	
AH	P3CR	AH	P8CR	AH	
BH	P3FC	BH	P8FC	BH	
CH	P4	CH	IIEC	CH	
DH	P5	DH	SCOUTC	DH	P3ODE
EH		EH		EH	
FH		FH		FH	P8ODE

[2] INTC

Address	Register	Address	Register	Address	Register
0080H	DMA0V	0090H	INTE0AD	00A0H	INTETC01
1H	DMA1V	1H	INTE12	1H	INTETC23
2H	DMA2V	2H	INTE34	2H	
3H	DMA3V	3H	INTE56	3H	
4H		4H		4H	
5H		5H	INTEA01	5H	
6H		6H	INTEA23	6H	
7H		7H	INTEA45	7H	
8H	INTCLR	8H	INTEA67	8H	
9H	DMAR	9H	INTETB0	9H	
AH	DMAB	AH		AH	
BH	(Reserved)	BH	INTETB0OV	BH	
CH	IIMC	CH	INTEUART	CH	
DH		DH	INTES2	DH	
EH		EH	INTES1	EH	
FH		FH		FH	

[3] CS/WAIT

Address	Register
00C0H	B0CS
1H	B1CS
2H	B2CS
3H	B3CS
4H	
5H	
6H	
7H	BEXCS
8H	MSAR0
9H	MAMR0
AH	MSAR1
BH	MAMR1
CH	MSAR2
DH	MAMR2
EH	MSAR3
FH	MAMR3

[4] CGEAR

Address	Register
00E0H	SYSCR0
1H	SYSCR1
2H	SYSCR2
3H	EMCCR0
4H	EMCCR1
5H	
6H	
7H	
8H	
9H	(Reserved)
AH	
BH	
CH	
DH	
EH	
FH	

[5] TMR8

Address	Register
0100H	TA01RUN
1H	
2H	TA0REG
3H	TA1REG
4H	TA01MOD
5H	TA1FFCR
6H	
7H	
8H	TA23RUN
9H	
AH	TA2REG
BH	TA3REG
CH	TA23MOD
DH	TA3FFCR
EH	
FH	

Address	Register
0110H	TA45RUN
1H	
2H	TA4REG
3H	TA5REG
4H	TA45MOD
5H	TA5FFCR
6H	
7H	
8H	TA67RUN
9H	
AH	TA6REG
BH	TA7REG
CH	TA67MOD
DH	TA7FFCR
EH	
FH	

[6] TMR16

Address	Register
0180H	TB0RUN
1H	
2H	TB0MOD
3H	TB0FFCR
4H	
5H	
6H	
7H	
8H	TB0RG0L
9H	TB0RG0H
AH	TB0RG1L
BH	TB0RG1H
CH	TB0CP0L
DH	TB0CP0H
EH	TB0CP1L
FH	TB0CP1H

[7] UART

Address	Register
0200H	SC0BUF
1H	SC0CR
2H	SC0MOD0
3H	BR0CR
4H	BR0ADD
5H	SC0MOD1
6H	
7H	
8H	
9H	
AH	
BH	
CH	
DH	
EH	
FH	

[8] I²C bus/SIO

Address	Register
0240H	SBI0CR10
1H	SBI0DBR0
2H	I2C0AR0
3H	SBI0CR20/SBISR0
4H	SBI0BR00
5H	SBI0BR10
6H	
7H	
8H	
9H	
AH	
BH	
CH	
DH	
EH	
FH	

[9] I²C bus

Address	Register
0250H	SBI0CR11
1H	SBI0DBR1
2H	I2C0AR1
3H	SBI0CR21/SBISR1
4H	SBI0BR01
5H	SBI0BR11
6H	
7H	
8H	SBI0CR12
9H	SBI0DBR2
AH	I2C0AR2
BH	SBI0CR22/SBISR2
CH	SBI0BR02
DH	SBI0BR12
EH	
FH	

[10] 12-bit ADC

Address	Register
02A0H	ADREGAL
1H	ADREGAH
2H	ADREGBL
3H	ADREGBH
4H	ADREGCL
5H	ADREGCH
6H	ADREGDL
7H	ADREGDH
8H	
9H	
AH	
BH	
CH	
DH	
EH	
FH	

Address	Register
02B0H	ADM0D0
1H	ADM0D1
2H	(Reserved)
3H	
4H	
5H	
6H	
7H	
8H	
9H	
AH	
BH	
CH	
DH	
EH	
FH	

[11] WDT

Address	Register
0300H	WDMOD
1H	WDCR
2H	
3H	
4H	
5H	
6H	
7H	
8H	
9H	
AH	
BH	
CH	
DH	
EH	
FH	

(1) I/O port

Symbol	Name	Address	7	6	5	4	3	2	1	0		
P0	Port 0	00H	P07	P06	P05	P04	P03	P02	P01	P00		
			R/W									
			Data from external port (Output latch register becomes undefined)									
P1	Port 1	01H	P17	P16	P15	P14	P13	P12	P11	P10		
			R/W									
			Data from external port (Output latch register is cleared to 0)									
P2	Port 2	06H	P27	P26	P25	P24	P23	P22	P21	P20		
			R/W									
			Data from external port (Output latch register is set to 1)									
P3	Port 3	07H	P37	P36	P35	P34	P33	P32	P31	P30		
			R/W									
			Data from external port (Output latch register is set to 1)								1	1
			-						0 (Output latch register) : Pull-up resistor OFF 1 (Output latch register) : Pull-up resistor ON		-	
P4	Port 4	0CH					P43	P42	P41	P40		
			R									
			Data from external port									
P5	Port 5	0DH	P57	P56	P55	P54	P53	P52	P51	P50		
			R									
			Data from external port									
P6	Port 6	12H						P62	P61	P60		
			R/W									
			Data from external port (Output latch register is set to 1)									
P7	Port 7	13H		P76	P75	P74	P73	P72	P71	P70		
			R/W									
			Data from external port (Output latch register is set to 1)									
P8	Port 8	18H	P87	P86	P85	P84	P83	P82	P81	P80		
			R/W									
			Data from external port (Output latch register is set to 1)									

(2) I/O port control (1/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0			
P0CR	Port 0 control	02H (Prohibit RMW)	P07C	P06C	P05C	P04C	P03C	P02C	P01C	P00C			
			W										
			0	0	0	0	0	0	0	0			
			0: Input 1: Output (When access to external, it is cleared to "0")										
P1CR	Port 1 control	04H (Prohibit RMW)	P17C	P16C	P15C	P14C	P13C	P12C	P11C	P10C			
			W										
			0	0	0	0	0	0	0	0			
			0: Input 1: Output										
P1FC	Port 1 function	05H (Prohibit RMW)	P17F	P16F	P15F	P14F	P13F	P12F	P11F	P10F			
			W										
			0	0	0	0	0	0	0	0			
			P1CR/P1FC = 00: Input port, 01: D8 to D15, 10: AD8 to AD15, 11: A8 to A15										
P3CR	Port 3 control	0AH (Prohibit RMW)	P37C	P36C	P35C	P34C	P33C	P32C	P31C	P30C			
			W										
			0	0	0	0	0	0	1	1			
			0: Input 1: Output										
P3FC	Port 3 function	0BH (Prohibit RMW)	P37F	P36F				P32F	P31F	P30F			
			W	W				W					
			0	0				0	0	0			
			0: Port 1: INT4	0: Port 1: TA7OUT				0: Port 1: HWR	0: Port 1: WR	0: Port 1: RD			
P3ODE	Port 3 open-drain enable	2DH (Prohibit RMW)	P37ODE	P36ODE	P35ODE	P34ODE	P33ODE	P32ODE	P31ODE	P30ODE			
			R/W										
			0	0	0	0	0	0	0	0			
			0: Normal 1: Opendrain										
P6CR	Port6 control	14H (Prohibit RMW)						P62C	P61C	P60C			
											W		
											0	0	0
			0: Input 1: Output										
P6FC	Port6 function	15H (Prohibit RMW)						P62F	P61F	P60F			
											W		
											0	0	0
											0: Port 1: INT2	0: Port 1: INT1	0: Port 1: INT0
SCOUTC	SCOUT control	1DH (Prohibit RMW)						SCOUTE					
											W		
											0		
											0: Port 1: SCOUT		
P7CR	Port 7 control	16H (Prohibit RMW)		P76C	P75C	P74C	P73C	P72C	P71C	P70C			
											W		
											0	0	0
			0: Input 1: Output										
P7FC	Port 7 function	17H (Prohibit RMW)		P76F	P75F			P72F	P71F	P70F			
											W		
											0	0	0
											0: Port 1: SCK0	0: Port 1: TB0OUT0	0: Port 1: TA5OUT
IIEC	Interrupt input enable control	1CH (Prohibit RMW)		INT3E		INT6E	INT5E						
											W		
											0	0	0
											0: Port 1: INT3	0: Port 1: INT6 TB0IN1	0: Port 1: INT5 TB0IN0

I/O port control (2/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0
P8CR	Port 8 control	1AH (Prohibit RMW)	P87C	P86C	P85C	P84C	P83C	P82C	P81C	P80C
			W							
			0	0	0	0	0	0	0	0
			0: Input 1: Output							
P8FC	Port 8 function	1BH (Prohibit RMW)	P87F	P86F	P85F	P84F		P82F	P81F	P80F
			W					W		
			0	0	0	0		0	0	0
			0: Port 1: SCL2	0: Port 1: SDA2	0: Port 1: SCL1	0: Port 1: SDA1		0: Port 1: TXD0	0: Port 1: SCL0	0: Port 1: SDA0 1: SO0
P8ODE	Port 8 open-drain enable	2FH					P83ODE	P82ODE	P81ODE	P80ODE
			R/W							
							0	0	0	0
			0: Normal 1: Opendrain							

(3) Interrupt control (1/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
INTE0AD	INTAD enable	90H	INTAD				INT0			
			IADC	IADM2	IADM1	IADM0	I0C	I0M2	I0M1	I0M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1: INTAD	Interrupt request level			1: INT0	Interrupt request level		
INTE12	INT1 INT2 enable	91H	INT2				INT1			
			I2C	I2M2	I2M1	I2M0	I1C	I1M2	I1M1	I1M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1: INT2	Interrupt request level			1: INT1	Interrupt request level		
INTE34	INT3 INT4 enable	92H	INT4				INT3			
			I4C	I4M2	I4M1	I4M0	I3C	I3M2	I3M1	I3M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1: INT4	Interrupt request level			1: INT3	Interrupt request level		
INTE56	INT5 INT6 enable	93H	INT6				INT5			
			I6C	I6M2	I6M1	I6M0	I5C	I5M2	I5M1	I5M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1: INT6	Interrupt request level			1: INT5	Interrupt request level		

Interrupt control (2/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
INTETA01	INTTA0 INTTA1 enable	95H	INTT1 (TMRA1)				INTT0 (TMRA0)			
			IT1C	IT1M2	IT1M1	IT1M0	IT0C	IT0M2	IT0M1	IT0M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
1: INTT1			Interrupt request level			1: INTT0			Interrupt request level	
INTETA23	INTTA2 INTTA3 enable	96H	INTT3 (TMRA3)				INTT2 (TMRA2)			
			IT3C	IT3M2	IT3M1	IT3M0	IT2C	IT2M2	IT2M1	IT2M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
1: INTT3			Interrupt request level			1: INTT2			Interrupt request level	
INTETA45	INTTA4 INTTA5 enable	97H	INTT5 (TMRA5)				INTT4 (TMRA4)			
			IT5C	IT5M2	IT5M1	IT5M0	IT4C	IT4M2	IT4M1	IT4M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
1: INTT5			Interrupt request level			1: INTT4			Interrupt request level	
INTETA67	INTTA6 INTTA7 enable	98H	INTT7 (TMRA7)				INTT6 (TMRA6)			
			IT7C	IT7M2	IT7M1	IT7M0	IT6C	IT6M2	IT6M1	IT6M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
1: INTT7			Interrupt request level			1: INTT6			Interrupt request level	
INTETB0	TMRB0 enable	99H	INTTB01 (TMRB0)				INTTB00 (TMRB0)			
			ITB01C	ITB01M2	ITB01M1	ITB01M0	ITB00C	ITB00M2	ITB00M1	ITB00M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
1: INTTB01			Interrupt request level			1: INTTB00			Interrupt request level	
INTETB0OV	TMRB0 enable (Overflow)	9BH	(Reserved)				INTTBOF0 (TMRB0 over flow)			
							ITF0C	ITF0M2	ITF0M1	ITF0M0
							R	R/W		
							0	0	0	0
1: INTT08			Interrupt request level							
INTUART	UART enable	9CH	INTTX0				INTRX0			
			ITX0C	ITX0M2	ITX0M1	ITX0M0	IRX0C	IRX0N2	IRX0M1	IRX0M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
1: INTTX0			Interrupt request level			1: INTRX0			Interrupt request level	
INTES2	INTI2C2 enable	9DH	(Reserved)				INTI2C2			
							INTI2C2C	I2C2M2	I2C2M1	I2C2M0
							R	R/W		
							0	0	0	0
1: INTI2C2			Interrupt request level							

Interrupt control (3/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
INTES1	INTI2C1 SBI0 enable	9EH	INTI2C1				INTSBI0			
			INTI2C1C	I2C1M2	I2C1M1	I2C1M0	IS0C	IS0M2	IS0M1	IS0M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
		1: INTI2C1	Interrupt request level			1: INTSBI	Interrupt request level			
INTETC01	INTTC0 INTTC1 enable	A0H	INTTC1				INTTC0			
			ITC1C	ITC1M2	ITC1M1	ITC1M0	ITC0C	ITC0M2	ITC0M1	ITC0M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
		1: INTTC1	Interrupt request level			1: INTTC0	Interrupt request level			
INTETC23	INTTC2 INTTC3 enable	A1H	INTTC3				INTTC2			
			ITC3C	ITC3M2	ITC3M1	ITC3M0	ITC2C	ITC2M2	ITC2M1	ITC2M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
		1: INTTC3	Interrupt request level			1: INTTC2	Interrupt request level			

Interrupt control (4/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0				
DMA0V	DMA0 request vector	80H	/	/	DMA0V5	DMA0V4	DMA0V3	DMA0V2	DMA0V1	DMA0V0				
					R/W									
					0	0	0	0	0	0				
					DMA0 start vector									
DMA1V	DMA1 request vector	81H	/	/	DMA1V5	DMA1V4	DMA1V3	DMA1V2	DMA1V1	DMA1V0				
					R/W									
					0	0	0	0	0	0				
					DMA1 start vector									
DMA2V	DMA2 request vector	82H	/	/	DMA2V5	DMA2V4	DMA2V3	DMA2V2	DMA2V1	DMA2V0				
					R/W									
					0	0	0	0	0	0				
					DMA2 start vector									
DMA3V	DMA3 request vector	83H	/	/	DMA3V5	DMA3V4	DMA3V3	DMA3V2	DMA3V1	DMA3V0				
					R/W									
					0	0	0	0	0	0				
					DMA3 start vector									
INTCLR	Interrupt clear control (Prohibit RMW)	88H	/	/	CLRV5	CLRV4	CLRV3	CLRV2	CLRV1	CLRV0				
					W									
					0	0	0	0	0	0				
					Interrupt request clear by DMA start vector									
DMAR	DMA software request register	89H	/	/	/	/	DMAR3	DMAR2	DMAR1	DMAR0				
							R/W							
							0	0	0	0				
							1: Soft request of DMA							
DMAB	DMA burst request register	8AH	/	/	/	/	DMAB3	DMAB2	DMAB1	DMAB0				
							R/W							
							0	0	0	0				
							1: Burst transfer of DMA							
IIMC	Input mode control (Prohibit RMW)	8CH	/	-	I4EDGE	I3EDGE	I2EDGE	I1EDGE	I0EDGE	I0LE	NMIREE			
				W										
				0	0	0	0	0	0	0	0			
				Always write 0	INT4 edge 0: Rising 1: Falling	INT3 edge 0: Rising 1: Falling	INT2 edge 0: Rising 1: Falling	INT1 edge 0: Rising 1: Falling	INT0 edge 0: Rising 1: Falling	INT0 edge 0: Edge 1: Level	1: $\overline{\text{NMI}}$ edge rising operation			

(4) Wait control (1/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0	
B0CS	Block 0 WAIT control register	C0H (Prohibit RMW)	B0E		-	-	B0BUS	B0W2	B0W1	B0W0	
			W		W						
			0		0	0	0	0	0	0	
			0: Disable 1: Enable		Always write 0	Data bus width selection 0: 16 bits 1: 8 bits	000: 2 waits 001: 1 wait 010: (1 + N) waits 011: 0 waits Others : Reserved				
B1CS	Block 1 WAIT control register	C1H (Prohibit RMW)	B1E		-	-	B1BUS	B1W2	B1W1	B1W0	
			W		W	W	W	W	W	W	
			0		0	0	0	0	0	0	
			0: Disable 1: Enable		Always write 0	Data bus width selection 0: 16 bits 1: 8 bits	000: 2 waits 001: 1 wait 010: (1 + N) waits 011: 0 waits Others : Reserved				
B2CS	Block 2 WAIT control register	C2H (Prohibit RMW)	B2E	B2M	-	-	B2BUS	B2W2	B2W1	B2W0	
			W	W	W	W	W	W	W	W	
			1	0	0	0	0	0	0	0	
			0: Disable 1: Enable	0: 16 M -area 1: Area setting	Always write 0	Data bus width selection 0: 16 bits 1: 8 bits	000: 2 waits 001: 1 wait 010: (1 + N) waits 011: 0 waits Others : Reserved				
B3CS	Block 3 WAIT control register	C3H (Prohibit RMW)	B3E		-	-	B3BUS	B3W2	B3W1	B3W0	
			W		W	W	W	W	W	W	
			0		0	0	0	0	0	0	
			0: Disable 1: Enable		Always write 0	Data bus width selection 0: 16 bits 1: 8 bits	000: 2 waits 001: 1 wait 010: (1 + N) waits 011: 0 waits Others : Reserved				
BEXCS	External WAIT control register	C7H (Prohibit RMW)					BEXBUS	BEXW2	BEXW1	BEXW0	
							W	W	W	W	
							0	0	0	0	
							Data bus width selection 0: 16 bits 1: 8 bits	000: 2 waits 001: 1 wait 010: (1 + N) waits 011: 0 waits Others : Reserved			

Chip select/wait control (2/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0
MSAR0	Memory start address register 0	C8H	S23	S22	S21	S20	S19	S18	S17	S16
			R/W							
			1	1	1	1	1	1	1	1
			Determines A23 to A16 of start address							
MAMR0	Memory address mask register 0	C9H	V20	V19	V18	V17	V16	V15	V14 to 9	V8
			R/W							
			1	1	1	1	1	1	1	1
			Set size of CS0 0: used for address compare							
MSAR1	Memory start address register 1	CAH	S23	S22	S21	S20	S19	S18	S17	S16
			R/W							
			1	1	1	1	1	1	1	1
			Determines A23 to A16 of start address							
MAMR1	Memory address mask register 1	CBH	V21	V20	V19	V18	V17	V16	V15 to 9	V8
			R/W							
			1	1	1	1	1	1	1	1
			Set size of CS1 0: used for address compare							
MSAR2	Memory start address register 2	CCH	S23	S22	S21	S20	S19	S18	S17	S16
			R/W							
			1	1	1	1	1	1	1	1
			Determines A23 to A16 of start address							
MAMR2	Memory address mask register 2	CDH	V22	V21	V20	V19	V18	V17	V16	V15
			R/W							
			1	1	1	1	1	1	1	1
			Set size of CS2 0: used for address compare							
MSAR3	Memory start address register 3	CEH	S23	S22	S21	S20	S19	S18	S17	S16
			R/W							
			1	1	1	1	1	1	1	1
			Determines A23 to A16 of start address							
MAMR3	Memory address mask register 3	CFH	V22	V21	V20	V19	V18	V17	V16	V15
			R/W							
			1	1	1	1	1	1	1	1
			Set size of CS3 0: used for address compare							

(5) Clock gear

Symbol	Name	Address	7	6	5	4	3	2	1	0		
SYSCR0	System clock control register 0	E0H	XEN	–	RXEN	–	–	WUEF	PRCK1	PRCK0		
			R/W									
			1	0	1	0	0	0	0	0		
			High-frequency oscillator (fc) 0: Stop 1: Oscillation	Always write 0	High-frequency oscillator (fc) after release of Stop mode 0: Stop 1: Oscillation	Always write 0	Always write 0	Warm-up timer Write 0: Don't care Write 1: start timer Read 0: end warm up Read 1: do not end warm up	Select prescaler clock 00: f _{FPH} 01: fc 10: fc/16 11: Reserved			
SYSCR1	System clock control register 1	E1H	7	6	5	4	–	GEAR2	GEAR1	GEAR0		
			R/W									
			1	0	1	0	0	0	0	0		
			High-frequency oscillator (fc) 0: Stop 1: Oscillation	Always write 0	High-frequency oscillator (fc) after release of Stop mode 0: Stop 1: Oscillation	Always write 0	Always write 0	Select gear value of high frequency (fc) 000: fc 001: fc/2 010: fc/4 011: fc/8 100: fc/16 101: (Reserved) 110: (Reserved) 111: (Reserved)				
SYSCR2	System clock control register 2	E2H	7	SCOSEL	WUPTM1	WUPTM0	HALTM1	HALTM0	1	DRVE		
			6	R/W	R/W	R/W	R/W	R/W	0	R/W		
			5	0	1	0	1	1	1	0		
			High-frequency oscillator (fc) 0: Stop 1: Oscillation	0: fs 1: f _{FPH}	Warm-up timer 00: Reserved 01: 2 ⁹ /inputted frequency 10: 2 ¹⁴ 11: 2 ¹⁶	HALT mode 00: Reserved 01: STOP mode 10: IDLE1 mode 11: IDLE2 mode			Pin state control in STOP mode 0: I/O off 1: Remain the state before HALT			
EMCCR0	EMC control register 0	E3H	PROTECT	–	–	–	ALEEN	EXTIN	DRVOSCH	–		
			R	R/W	R/W	R/W	R/W	R/W	R/W	R/W		
			0	0	1	0	0	0	1	1		
			1: Protected	Always write 0	Always write 1	Always write 0	ALE output enable	fc external clock	fc OSC 1: Normal 0: Weak	Always write 1		
EMCCR1	EMC control register 1	E4H	Protection is turned OFF by writing 1FH. Protection is turned ON by writing any value other than 1FH.									

(6) 8-bit timer (1/4)

(6-1) TMRA01

Symbol	Name	Address	7	6	5	4	3	2	1	0
TA01RUN	Timer RUN	100H	TA0RDE	 	 	 	I2TA01	TA01PRUN	TA1RUN	TA0RUN
			R/W	 	 	 	R/W	R/W	R/W	R/W
			0	 	 	 	0	0	0	0
			Double buffer 0: Disable 1: Enable				IDLE2 0: Stop 1: Operate	8-bit timer run/stop control 0: Stop and clear 1: Run (Count up)		
TA0REG	8-bit timer register 0	102H (Prohibit RMW)	-							
			W							
			Undefined							
TA1REG	8-bit timer register 1	103H (Prohibit RMW)	-							
			W							
			Undefined							
TA01MOD	8-bit timer source CLK and mode	104H	TA01M1	TA01M0	PWM01	PWM00	TA1CLK1	TA1CLK0	TA0CLK1	TA0CLK0
			R/W							
			0	0	0	0	0	0	0	0
			00: 8-bit timer 01: 16-bit timer 10: 8-bit PPG 11: 8-bit PWM		00: Reserved 01: 2 ⁶ PWM cycle 10: 2 ⁷ 11: 2 ⁸		00: TA0TRG 01: φT1 10: φT16 11: φT256		00: Reserved 01: φT1 10: φT4 11: φT16	
TA1FFCR	8-bit timer flip-flop control	105H (Prohibit RMW)	 	 	 	 	TA1FFC1	TA1FFC0	TA1FFIE	TA1FFIS
			R/W							
			 	 	 	 	1	1	0	0
							00: Invert TA1FF 01: Set TA1FF 10: Clear TA1FF 11: Don't care		1: TA1FF invert enable	TA1FF inversion select 0: TMRA0 1: TMRA1

8-bit timer (2/4)

(6-2) TMRA23

Symbol	Name	Address	7	6	5	4	3	2	1	0
TA23RUN	Timer RUN	108H	TA2RDE	 	 	 	I2TA23	TA23PRUN	TA3RUN	TA2RUN
			R/W	 	 	 	R/W	R/W	R/W	R/W
			0	 	 	 	0	0	0	0
			Double buffer 0: Disable 1: Enable				IDLE2 0: Stop 1: Operate	8-bit timer run/stop control 0: Stop and clear 1: Run (Count up)		
TA2REG	8-bit timer register 0	10AH (Prohibit RMW)	—							
			W							
			Undefined							
TA2REG	8-bit timer register 1	10BH (Prohibit RMW)	—							
			W							
			Undefined							
TA23MOD	8-bit timer source CLK and mode	10CH	TA23M1	TA23M0	PWM21	PWM20	TA3CLK1	TA3CLK0	TA2CLK1	TA2CLK0
			R/W							
			0	0	0	0	0	0	0	0
			00: 8-bit timer 01: 16-bit timer 10: 8-bit PPG 11: 8-bit PWM	00: Reserved 01: 2 ⁶ PWM cycle 10: 2 ⁷ 11: 2 ⁸	00: TA2TRG 01: φT1 10: φT16 11: φT256	00: Reserved 01: φT1 10: φT4 11: φT16				
TA3FFCR	8-bit timer flip-flop control	10DH (Prohibit RMW)	 	 	 	 	TA3FFC1	TA3FFC0	TA3FFIE	TA3FFIS
			R/W							
			 	 	 	 	1	1	0	0
							00: Invert TA3FF 01: Set TA3FF 10: Clear TA3FF 11: Don't care	1: TA3FF invert enable	TA1FF inversion select 0: TMRA2 1: TMRA3	

8-bit timer (3/4)

(6-3) TMRA45

Symbol	Name	Address	7	6	5	4	3	2	1	0
TA45RUN	Timer RUN	110H	TA4RDE	 	 	 	I2TA45	TA45PRUN	TA5RUN	TA4RUN
			R/W	 	 	 	R/W	R/W	R/W	R/W
			0	 	 	 	0	0	0	0
			Double buffer 0: Disable 1: Enable				IDLE2 0: Stop 1: Operate	8-bit timer run/stop control 0: Stop and clear 1: Run (Count up)		
TA4REG	8-bit timer register 0	112H (Prohibit RMW)	-							
			W							
			Undefined							
TA5REG	8-bit timer register 1	113H (Prohibit RMW)	-							
			W							
			Undefined							
TA45MOD	8-bit timer source CLK and mode	114H	TA45M1	TA45M0	PWM41	PWM40	TA5CLK1	TA5CLK0	TA4CLK1	TA4CLK0
			R/W							
			0	0	0	0	0	0	0	0
			00: 8-bit timer 01: 16-bit timer 10: 8-bit PPG 11: 8-bit PWM		00: Reserved 01: 2 ⁶ PWM cycle 10: 2 ⁷ 11: 2 ⁸		00: TA4TRG 01: φT1 10: φT16 11: φT256		00: Reserved 01: φT1 10: φT4 11: φT16	
TA5FFCR	8-bit timer flip-flop control	115H (Prohibit RMW)	 	 	 	 	TA5FFC1	TA5FFC0	TA5FFIE	TA5FFIS
			 	 	 	 	R/W			
			 	 	 	 	1	1	0	0
							00: Invert TA5FF 01: Set TA5FF 10: Clear TA5FF 11: Don't care		1: TA5FF invert enable	TA5FF inversion select 0: TMRA4 1: TMRA5

8-bit timer (4/4)

(6-4) TMRA67

Symbol	Name	Address	7	6	5	4	3	2	1	0
TA67RUN	Timer RUN	118H	TA6RDE	 	 	 	I2TA67	TA67PRUN	TA7RUN	TA6RUN
			R/W	 	 	 	R/W	R/W	R/W	R/W
			0	 	 	 	0	0	0	0
			Double buffer 0: Disable 1: Enable				IDLE2 0: Stop 1: Operate	8-bit timer run/stop control 0: Stop and clear 1: Run (Count up)		
TA6REG	8-bit timer register 0	11AH (Prohibit RMW)	—							
			W							
			Undefined							
TA7REG	8-bit timer register 1	11BH (Prohibit RMW)	—							
			W							
			Undefined							
TA67MOD	8-bit timer source CLK and mode	11CH	TA67M1	TA67M0	PWM61	PWM60	TA7CLK1	TA7CLK0	TA6CLK1	TA6CLK0
			R/W							
			0	0	0	0	0	0	0	0
			00: 8-bit timer 01: 16-bit timer 10: 8-bit PPG 11: 8-bit PWM	00: Reserved 01: 2 ⁶ PWM cycle 10: 2 ⁷ 11: 2 ⁸	00: TA6TRG 01: φT1 10: φT16 11: φT256	00: External input (TA6IN) 01: φT1 10: φT4 11: φT16				
TA7FFCR	8-bit timer flip-flop control	11DH (Prohibit RMW)	 	 	 	 	TA7FFC1	TA7FFC0	TA7FFIE	TA7FFIS
			 	 	 	 	R/W			
			 	 	 	 	1	1	0	0
							00: Invert TA7FF 01: set TA7FF 10: Clear TA7FF 11: Don't care	1: TA7FF invert enable	TA7FF inversion select 0: TMRA6 1: TMRA7	

(7) 16-bit timer

(7-1) TMRB0

Symbol	Name	Address	7	6	5	4	3	2	1	0
TB0RUN	Timer control	180H	TBORDE	—			I2TB0	TB0PRUN		TB0RUN
			R/W	R/W			R/W	R/W		R/W
			0	0			0	0		0
			Double buffer 0: Disable 1: Enable	Always write 0			IDLE2 0: Stop 1: Operate	16-bit timer run/stop control 0: Stop and clear 1: Run (Count up)		
TB0MOD	16-bit timer source CLK and mode	182H (Prohibit RMW)	TB0CT1	TB0ET1	TB0CP0I	TB0CPM1	TB0CPM0	TB0CLE	TB0CLK1	TB0CLK0
			R/W		W*	R/W				
			0	0	1	0	0	0	0	0
			TB0FF1 INV TRG 0: TRG disable 1: TRG enable		0: Soft-capture 1: Undefined	Capture timing (TB0IN0, TB0IN1) 00: Disable 01: ↑, ↓ 10: ↑, ↓ 11: ↑, ↓ (TA1OUT)		1: TB0UC clear enable	Source clock 00: TB0IN0 input 01: φT1 10: φT4 11: φT16	
	CAP1	TREG1 matching								
TB0FFCR	16-bit timer flip-flop control	183H (Prohibit RMW)	TB0FF1C1	TB0FF1C0	TB0C1T1	TB0C0T1	TB0E1T1	TB0E0T1	TB0FF0C1	TB0FF0C0
			W*		R/W				W*	
			1	1	0	0	0	0	1	1
			00: Invert TB0FF1 01: Set 10: Clear 11: Don't care "Always read as 0"		TB0FF0 inversion trigger 0: Disable trigger 1: Enable trigger Invert when the UC value is loaded in to TB0CP1.				Invert when the UC value is loaded in to TB0CP0.	Invert when the UC value matches the value in TB0RG1.
TB0RG0L	16-bit timer register 0 L	188H (Prohibit RMW)	—							
			W							
			Undefined							
TB0RG0H	16-bit timer register 0 H	189H (Prohibit RMW)	—							
			W							
			Undefined							
TB0RG1L	16-bit timer register 1 L	18AH (Prohibit RMW)	—							
			W							
			Undefined							
TB0RG1H	16-bit timer register 1 H	18BH (Prohibit RMW)	—							
			W							
			Undefined							
TB0CP0L	Capture register 0 L	18CH	—							
			R							
			Undefined							
TB0CP0H	Capture register 0 H	18DH	—							
			R							
			Undefined							
TB0CP1L	Capture register 1 L	18EH	—							
			R							
			Undefined							
TB0CP1H	Capture register 1 H	18FH	—							
			R							
			Undefined							

(8) UART

(8-1) UART

Symbol	Name	Address	7	6	5	4	3	2	1	0		
SC0BUF	Serial channel 0 buffer	200H (Prohibit RMW)	RB7/TB7	RB6/TB6	RB5/TB5	RB4/TB4	RB3/TB3	RB2/TB2	RB1/TB1	RB0/TB0		
			R (Receiving)/W (Transmission)									
			Undefined									
SC0CR	Serial channel 0 control	201H	RB8	EVEN	PE	OERR	PERR	FERR	–	–		
			R	R/W			R (Cleared to 0 by reading)		R/W	R/W		
			Undefined	0	0	0	0	0	0	0		
			Receive data bit	Parity 0: Odd 1: Even	1: Parity enable	1: Error Overrun Parity Framing			Always write 0	Always write 0		
SC0MOD0	Serial channel 0 mode 0	202H	TB8	CTSE	RXE	WU	SM1	SM0	SC1	SC0		
			R/W									
			0	0	0	0	0	0	0	0		
			Send data bit8	Hand shake 0: CTS disable 1: CTS enable	1: Receive enable	1: Wakeup enable	00: Reserved 01: UART 7 bits 10: UART 8 bits 11: UART 9 bits		00: TA0OUT trigger 01: Baud rate generator 10: Internal clock f_{SYS} 11: Reserved			
BR0CR	Baud rate control	203H	–	BROADDE	BR0CK1	BR0CK0	BR0S3	BR0S2	BR0S1	BR0S0		
			R/W									
			0	0	0	0	0	0	0	0		
			Always write 0	$(16 - K)/16$ divided frequency 0: Disable 1: Enable	00: $\phi T0$ (4/fc) 01: $\phi T2$ (16/fc) 10: $\phi T8$ (64/fc) 11: $\phi T32$ (256/fc)	Setting the divided frequency "N" (0 to F)						
BR0ADD	Serial channel 0 K setting register	204H	 	 	 	 	BRA0K3	BRA0K2	BRA0K1	BRA0K0		
			R/W									
			 	 	 	 	0	0	0	0		
			Sets the frequency divisor "K" (Divided by N + (16 - K)/16)									
SC0MOD1	Serial channel 0 mode 1	205H	I2S0	–	 	 	 	 	 	 		
			R/W	R/W	 	 	 	 	 	 		
			0	0	 	 	 	 	 	 		
			IDLE2 0: Stop 1: Run	Always write 0	 	 	 	 	 	 		

(9) I²C bus/SIO (1/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0	
SBI0CR10	Serial bus interface control register 10	240H (I ² C bus mode) (Prohibit RMW)	BC2	BC1	BC0	ACK		SCK2	SCK1	SCK0/SWRMON	
			W			R/W		W	W	R/W	
			0	0	0	0		0	0	0/1	
		Number of transferred bit 000: 8, 100: 4 001: 1, 101: 5 010: 2, 110: 6 011: 3, 111: 7					Acknowledge mode 0: Disable 1: Enable	Selection of Serial clock frequency 000: 5 100: 9 001: 6 101: 10 010: 7 110: 11 011: 8 111: Reserved			
		240H (SIO mode) (Prohibit RMW)	SIOS	SIOINH	SIOM1	SIOM0		SCK2	SCK1	SCK0	
			W	W	W	W		W	W	W	
0	0		0	0	0	0		0			
Transfer start 0: End 1: Start		Continue/aboard transfer 0: Continue 1: Stop	Transfer mode select 00: 8-bit transmit 10: 8-bit transmit/receive 11: 8-bit receive			Selection of serial clock frequency 000: 4 100: 8 001: 5 101: 9 010: 6 110: 10 011: 7 111: Reserved					
SBI0DBR0	SBI buffer register 0	241H (Prohibit RMW)	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
			R (Receiving)/W (Transmission)								
			Undefined								
I2C0AR0	I ² C Bus address register 0	242H (Prohibit RMW)	SA6	SA5	SA4	SA3	SA2	SA1	SA0	ALS	
			W	W	W	W	W	W	W	W	
			0	0	0	0	0	0	0	0	
			Set of slave address								Address recognition mode 1: No 0: Slave address
Write SBI0CR20	Serial bus interface control register 20	243H (I ² C bus mode) (Prohibit RMW)	MST	TRX	BB	PIN	SBIM1	SBIM0	SWRST1	SWRST0	
			W	W	W	W	W	W	W	W	
			0	0	0	1	0	0	0	0	
			Master/Slave selection 0: Slave 1: Master	Transfer/receiver selection 0: Receive 1: Transfer	Start/stop condition generation 0: Stop 1: Start	INTSBIO interrupt request cancel 0: Don't care 1: Cancel	Serial bus interface operating mode selection 00: Port mode 01: SIO mode 10: I ² C bus mode 11: Reserved	Software reset generate write 10 and 01, then an internal reset signal is generated			
Read SBISR0	Serial bus interface status register 0	243H (I ² C bus mode) (Prohibit RMW)	MST	TRX	BB	PIN	AL	AAS	AD0	LRB	
			R	R	R	R	R	R	R	R	
			0	0	0	1	0	0	0	0	
			0: Slave 1: Master	0: Receive 1: Transfer	I ² C bus status monitor 0: Free 1: Busy	INTSBIO request monitor 0: Request 1: Cancel	Arbitration lost detection monitor 1: Detect	Slave address match detection monitor 1: Detect	General call detection monitor 1: Detect	End bit monitor 0: "0" 1: "1"	

I²C bus/SIO (2/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0
Write SBI0CR20	Serial bus interface control register 2	243H (SIO mode) (Prohibit RMW)	7	6	5	4	SBIM1	SBIM0	-	-
			6	5	4	3	W		W	W
			5	4	3	2	1	1	0	0
			4	3	2	1	Serial bus interface operating mode selection 00: Port mode 01: SIO mode 10: I ² C bus mode 11: Reserved		Always write 0	Always write 0
Read SBISR0	Serial bus interface status register 00	243H (SIO mode) (Prohibit RMW)	7	6	5	4	SIOF	SEF	1	0
			6	5	4	3	R	R	1	0
			5	4	3	2	0	0	1	0
			4	3	2	1	Transfer monitor 0: End 1: Transfer	Shift operate monitor 0: End 1: Shift	1	0
SBI0BR00	Serial bus interface baud rate register 00	244H (Prohibit RMW)	-	I2SBI0	5	4	3	2	1	0
			W	R/W	5	4	3	2	1	0
			0	0	5	4	3	2	1	0
			Always write 0	IDLE2 0: Stop 1: Operate	5	4	3	2	1	0
SBI0BR10	Serial bus interface baud rate register 10	245H (Prohibit RMW)	P4EN	-	5	4	3	2	1	0
			W	W	5	4	3	2	1	0
			0	0	5	4	3	2	1	0
			Internal clock 0: Disable 1: Enable	Always write 0	5	4	3	2	1	0

(10) I²C bus 1 (1/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0	
SBI0CR11	Serial bus interface control register 11	250H (Prohibit RMW)	BC2	BC1	BC0	ACK	/	SCK2	SCK1	SCK0/ SWRMON	
			W			R/W		W	W	R/W	
			0	0	0	0		0	0	0/1	
			Number of transferred bit				Acknowledge mode 0: Disable 1: Enable	Selection of serial clock frequency			
			000: 8	100: 4		000: 5		100: 9			
001: 1	101: 5		001: 6	101: 10							
010: 2	110: 6		010: 7	110: 11							
011: 3	111: 7				011: 8	111: Reserved					

I²C bus 1 (2/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0	
SBI0DBR1	SBI buffer register 1	251H (Prohibit RMW)	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
			R (Receiving)/W (Transmission)								
			Undefined								
I2C0AR1	I ² C Bus address register 1	252H (Prohibit RMW)	SA6	SA5	SA4	SA3	SA2	SA1	SA0	ALS	
			W	W	W	W	W	W	W	W	
			0	0	0	0	0	0	0	0	
			Set of slave address								Address recognition mode 1: No 0: Slave address
Write SBI0CR21	Serial bus interface control register 21	253H (I ² C bus mode) (Prohibit RMW)	MST	TRX	BB	PIN	SBIM1	SBIM0	SWRST1	SWRST0	
			W	W	W	W	W	W	W	W	
			0	0	0	1	0	0	0	0	
			Master/slave selection 0: Slave 1: Master	Transfer/receiver selection 0: Receive 1: Transfer	Start/stop condition generation 0: Stop 1: Start	INTI2C1 interrupt request cancel 0: Don't care 1: Cancel	Serial bus interface operating mode selection 00: Port mode 01: Reserved 10: I ² C bus mode 11: Reserved	Software reset generate write 10 and 01, then an internal reset signal is generated			
Read SBISR1	Serial bus interface status register 1	253H (I ² C bus mode) (Prohibit RMW)	MST	TRX	BB	PIN	AL	AAS	AD0	LRB	
			R	R	R	R	R	R	R	R	
			0	0	0	1	0	0	0	0	
			0: Slave 1: Master	0: Receive 1: Transfer	I ² C bus status monitor 0: Free 1: Busy	INTI2C1 request monitor 0: Request 1: Cancel	Arbitration lost detection monitor 1: Detect	Slave address match detection monitor 1: Detect	General call detection monitor 1: Detect	End bit monitor 0: "0" 1: "1"	
SBI0BR01	Serial bus interface baud rate register 01	254H (Prohibit RMW)	–	I2SBI0							
			W	R/W							
			0	0							
			Always write 0	IDLE2 0: Stop 1: Operate							
SBI0BR11	Serial bus interface baud rate register 11	255H (Prohibit RMW)	P4EN	–							
			W	W							
			0	0							
			Internal clock 0: Disable 1: Enable	Always write 0							

(11) I²C bus 2 (1/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0
SBI0CR12	Serial bus interface control register 12	258H (Prohibit RMW)	BC2	BC1	BC0	ACK	/	SCK2	SCK1	SCK0/ SWRMON
			W			R/W		W	W	R/W
			0	0	0	0		0	0	0
			Number of transferred bit					Acknowledge mode	Selection of serial clock frequency	
000: 8			100: 4		0: Disable		000: 5		100: 9	
001: 1			101: 5		1: Enable		001: 6		101: 10	
010: 2			110: 6							
011: 3			111: 7				010: 7		110: 11	
								011: 8		111: Reserved

I²C bus 2 (2/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0		
SBI0DBR2	SBI buffer register 2	259H (Prohibit RMW)	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
			R (Receiving)/W (Transmission)									
			Undefined									
I2C0AR2	I ² C Bus address register 2	25AH (Prohibit RMW)	SA6	SA5	SA4	SA3	SA2	SA1	SA0	ALS		
			W	W	W	W	W	W	W	W	W	
			0	0	0	0	0	0	0	0	0	
Set of slave address									Address recognition mode 1: No 0: Slave address			
Write SBI0CR22	Serial bus interface control register 22	25BH (I ² C bus mode) (Prohibit RMW)	MST	TRX	BB	PIN	SBIM1	SBIM0	SWRST1	SWRST0		
			W	W	W	W	W	W	W	W		
			0	0	0	1	0	0	0	0		
			Master/slave selection 0: Slave 1: Master	Transfer/receiver selection 0: Receive 1: Transfer	Start/Stop condition generation 0: Stop 1: Start	INTI2C2 interrupt request cancel 0: Don't care 1: Cancel	Serial bus interface operating mode selection 00: Port mode 01: Reserved 10: I ² C bus mode 11: Reserved	Software reset generate write "10" and "01", then an internal reset signal is generated				
Read SBISR2	Serial bus interface status register 2	25BH (I ² C bus mode) (Prohibit RMW)	MST	TRX	BB	PIN	AL	AAS	AD0	LRB		
			R	R	R	R	R	R	R	R		
			0	0	0	1	0	0	0	0		
			0: Slave 1: Master	0: Receive 1: Transfer	I ² C bus status monitor 0: Free 1: Busy	INTI2C2 request monitor 0: Request 1: Cancel	Arbitration lost detection monitor 1: Detect	Slave address match detection monitor 1: Detect	General call detection monitor 1: Detect	End bit monitor 0: "0" 1: "1"		
SBI0BR02	Serial bus interface baud rate register 02	25CH (Prohibit RMW)	-	I2SBI0								
			W	R/W								
			0	0								
			Always write 0	IDLE2 0: Stop 1: Operate								
SBI0BR12	Serial bus interface baud rate register 12	25DH (Prohibit RMW)	P4EN	-								
			W	W								
			0	0								
			Internal clock 0: Disable 1: Enable	Always write 0								

(12) AD converter (1/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0
ADMOD0	AD mode register 0	2B0H	EOCF	ADBF	–	–	ITM0	REPEAT	SCAN	ADS
			R		R/W	R/W	R/W	R/W	R/W	R/W
			0	0	0	0	0	0	0	0
			AD conversion end flag 1: End	AD conversion bust flag 1: Busy	Always write 0		Interrupt in repeat mode	Repeat mode specific -ation 1: Repeat	Scan mode Specific -ation 1: Scan	AD conversion start 1: Start
ADMOD1	AD mode register 1	2B1H	VREFON	I2AD		ADTRGE	ADCH3	ADCH2	ADCH1	ADCH0
			R/W	R/W		R/W	R/W			
			0	0		0	0	0	0	0
			VREF control 1: VREF on	IDLE2 0: Abort 1: Operate		AD control for external start 1: Enable	Input channel 0000: AN0 AN0 0001: AN1 AN0 → AN1 0010: AN2 AN0 → AN1 → AN2 0011: AN3 AN0 → AN1 → AN2 → AN3 0100: AN4 AN4 0101: AN5 AN4 → AN5 0110: AN6 AN4 → AN5 → AN6 0111: AN7 AN4 → AN5 → AN6 1000: AN8 AN8 1001: AN9 AN9 1010: AN10 AN10 1011: AN11 AN11 1100: Reserved 1101: Reserved 1110: Reserved 1111: Reserved			

AD converter (2/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0	
ADREGAL	AD result register A low	2A0H	ADRA1	ADRA0						ADRARF	
			R							R	
			Undefined								0
			Stores lower 2 bits of AD conversion result								1: Conversion result stored
ADREGAH	AD result register A high	2A1H	ADRA9	ADRA8	ADRA7	ADRA6	ADRA5	ADRA4	ADRA3	ADRA2	
			R								
			Undefined								
			Stores upper 8 bits of AD conversion result								
ADREGBL	AD result register B low	2A2H	ADRB1	ADRB0						ADBRF	
			R							R	
			Undefined								0
			Stores lower 2 bits of AD conversion result								1: Conversion result stored
ADREGBH	AD result register B high	2A3H	ADRB9	ADRB8	ADRB7	ADRB6	ADRB5	ADRB4	ADRB3	ADRB2	
			R								
			Undefined								
			Stores upper 8 bits of AD conversion result								
ADREGCL	AD result register C low	2A4H	ADRC1	ADRC0						ADRCRF	
			R							R	
			Undefined								0
			Stores lower 2 bits of AD conversion result								1: Conversion result stored
ADREGCH	AD result register C high	2A5H	ADRC9	ADRC8	ADRC7	ADRC6	ADRC5	ADRC4	ADRC3	ADRC2	
			R								
			Undefined								
			Stores upper 8 bits of AD conversion result								
ADREGDL	AD result register D low	2A6H	ADRD1	ADRD0						ADDRF	
			R							R	
			Undefined								0
			Stores lower 2 bits of AD conversion result								1: Conversion result stored
ADREGDH	AD result register D high	2A7H	ADRD9	ADRD8	ADRD7	ADRD6	ADRD5	ADRD4	ADRD3	ADRD2	
			R								
			Undefined								
			Stores upper 8 bits of AD conversion result								

(13) Watchdog timer

Symbol	Name	Address	7	6	5	4	3	2	1	0
WDMOD	WDT mode register	300H	WDTE	WDTP1	WDTP0	 	 	I2WDT	RESCR	–
			R/W	R/W	R/W	 	 	R/W	R/W	R/W
			1	0	0	 	 	0	0	0
			1: WDT enable	00: $2^{15}/f_{SYS}$ 01: $2^{17}/f_{SYS}$ 10: $2^{19}/f_{SYS}$ 11: $2^{21}/f_{SYS}$			IDLE2 0: Abort 1: Operate	1: RESET connect internally WDT out to reset pin	Always write 0	
WDCR	WD control	301H (Prohibit RMW)	–							
			W							
			–							
			B1H: WDT disable 4EH: WDT clear							

6. Port Section Equivalent Circuit Diagrams

- Reading the circuit diagrams

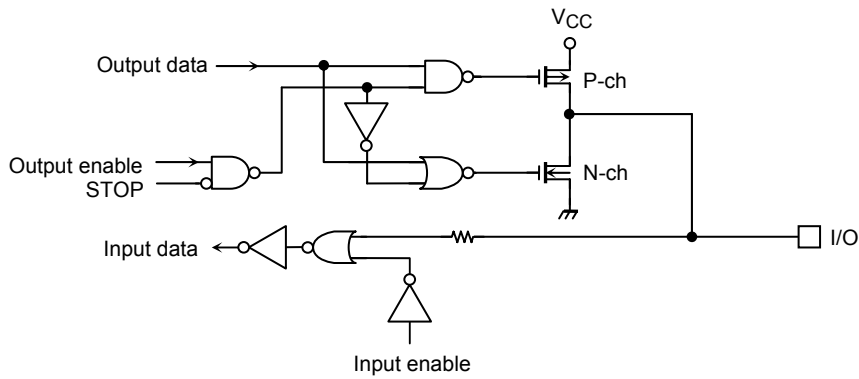
The gate symbols used are essentially the same as those used for the standard CMOS logic IC [74HCXX] series.

The dedicated signal is described below.

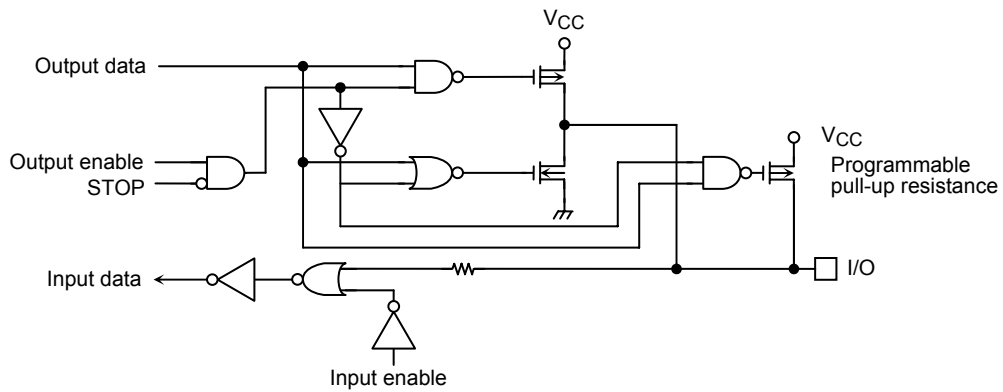
STOP: This signal becomes active (1) when the HALT mode setting register is set to STOP mode (e.g., when SYSCR2<HALTM1:0> = 0, 1) and the CPU executes the HALT instruction. When the drive enable bit SYSCR2<DRVE> is set to 1, however, STOP will remain at 0.

- The input protection resistances range from several tens of ohms to several hundreds of ohms.

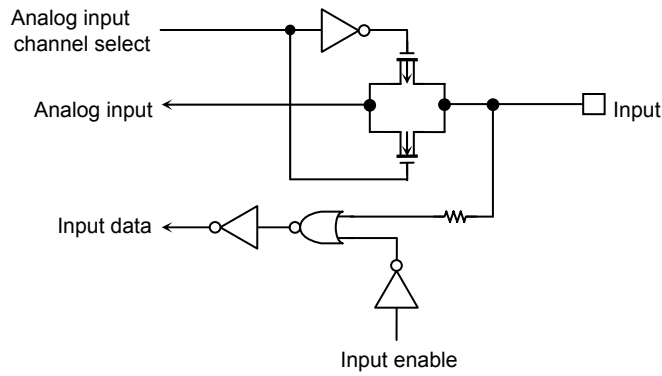
- P0 (AD0 to AD7), P1 (AD8 to AD15, A8 to A15), P2 (A16 to A23, A0 to A7), P6 and P7



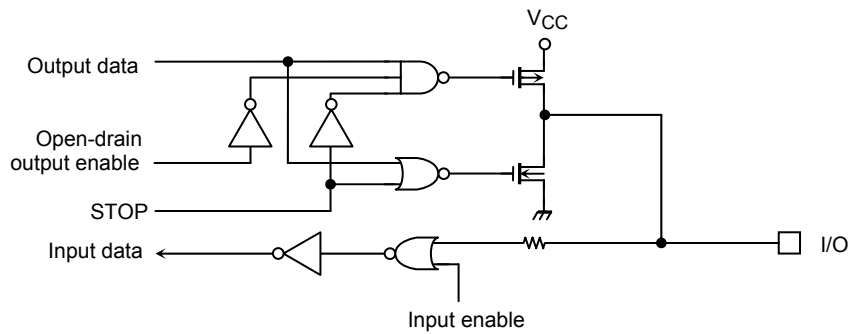
- P32 and P33



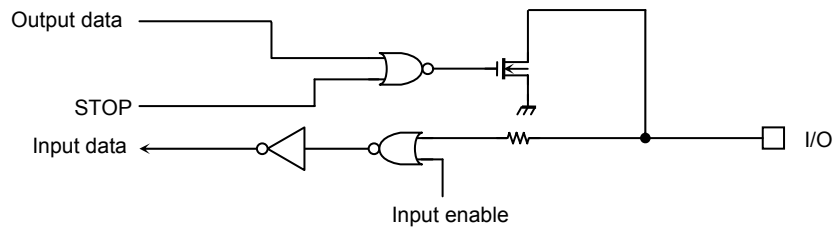
■ P4 (AN8 to AN11) and P5 (AN0 to AN7)



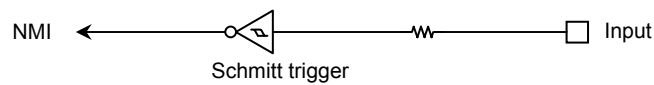
■ P80 (SDA0/SO0), P81 (SCL0/SI0), P82 (TXD), P83 (RXD), P30 (RD), P31 (WR), P34, P35 (TA6IN), P36 (TA7OUT) and P37 (INT4)



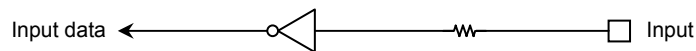
■ P84 (SDA1), P85 (SCL1), P86 (SDA2) and P87 (SCL2)



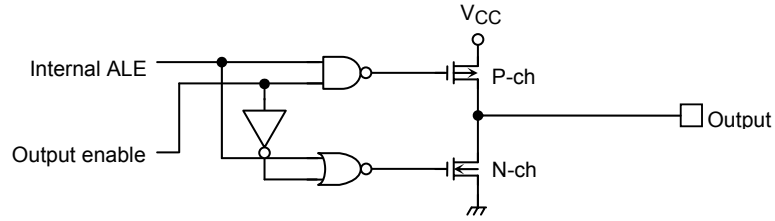
■ $\overline{\text{NMI}}$



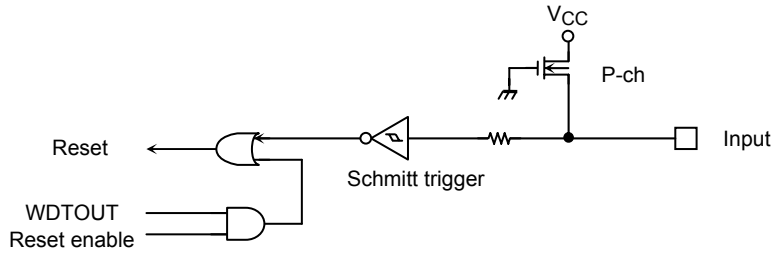
■ AM0 to AM1



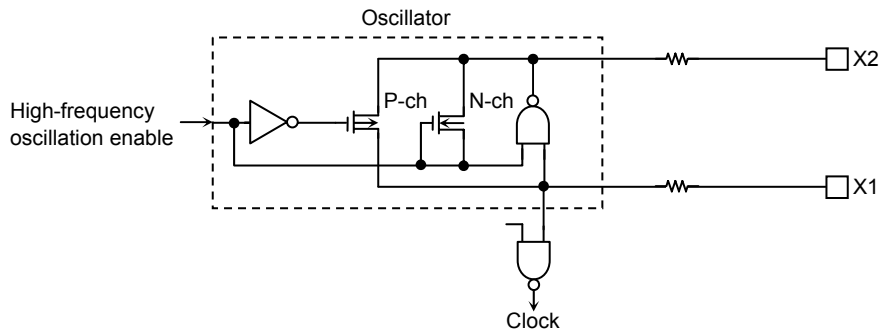
■ ALE



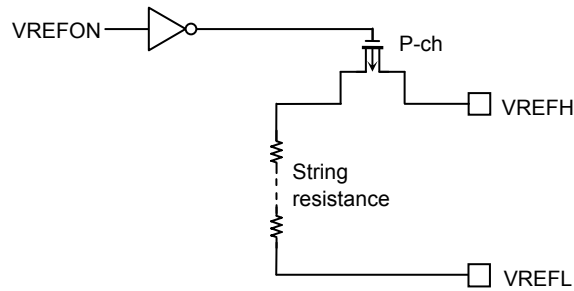
■ $\overline{\text{RESET}}$



■ X1 and X2



■ VREFH and VREFL



7. Points of Note and Restrictions

(1) Notation

1. The notation for built-in I/O registers is as follows. Register symbol <Bit symbol>

Example: TA01RUN<TA0RUN> denotes bit TA0RUN of register TA01RUN.

2. Read-modify-write instructions

An instruction in which the CPU reads data from memory and writes the data to the same memory location in one instruction.

Example 1: SET 3, (TA01RUN) ... Set bit3 of TA01RUN.

Example 2: INC 1, (100H) ... Increment the data at 100H.

- Examples of read-modify-write instructions on the TLCS-900

Exchange instruction

EX (mem), R

Arithmetic operations

ADD (mem), R/# ADC (mem), R/#

SUB (mem), R/# SBC (mem), R/#

INC #3, (mem) DEC #3, (mem)

Logic operations

AND (mem), R/# OR (mem), R/#

XOR (mem), R/#

Bit manipulation operations

STCF #3/A, (mem) RES #3, (mem)

SET #3, (mem) CHG #3, (mem)

TSET #3, (mem)

Rotate and shift operations

RLC (mem) RRC (mem)

RL (mem) RR (mem)

SLA (mem) SRA (mem)

SLL (mem) SRL (mem)

RLD (mem) RRD (mem)

3. fc, fFPH, fSYS and one state

The clock frequency input on ins X1 and 2 is called fOSCH. The clock selected by DFMCRO<ACT1:0> is called fc.

The clock selected by SYSCR1<SYSCK> is called fFPH. The clock frequency give by fFPH divided by 2 is called fSYS.

One cycle of fSYS is referred to as one state.

(2) Points of note and restrictions

1. AM0 and AM1 pins

Fix these pins to V_{CC} unless changing voltage.

2. EMU0 and EMU1

Open pins.

3. Reserved address areas

The TMP91CW18A does not have any reserved areas.

4. Warm-up counter

The warm-up counter operates when STOP mode is released, even if the system is using an external oscillator. As a result a time equivalent to the warm-up time elapses between input of the release request and output of the system clock.

5. Programmable pull-up resistance

The programmable pull-up resistor can be turned ON/OFF by a program when the ports are set for use as input ports. When the ports are set for use as output ports, they cannot be turned ON/OFF by a program.

The data registers (e.g., P3) are used to turn the pull-up/pull-down resistors ON/OFF. Consequently read-modify-write instructions are prohibited.

6. Watchdog timer

The watchdog timer starts operation immediately after a reset is released. When the watchdog timer is not to be used, disable it.

When the bus is released, neither internal memory nor internal I/O can be accessed. However, the internal I/O continues to operate. Hence the watchdog timer continues to run. Therefore be careful about the bus releasing time and set the detection timer of watchdog timer.

7. AD converter

The string resistor between the VREFH and VREFL pins can be cut by a program so as to reduce power consumption. When STOP mode is used, disable the resistor using the program before the HALT instruction is executed.

8. CPU (Micro DMA)

Only the LDC cr, r and LDC r, cr instructions can be used to access the control registers in the CPU (e.g., the transfer source address register (DMASn)).

9. Undefined SFR

The value of an undefined bit in an SFR is undefined when read.

10. POP SR instruction

Please execute the POP SR instruction during DI condition.

11. Releasing the HALT mode by requesting an interruption

Usually, interrupts can release all halts status. However, the interrupts = ($\overline{\text{NMI}}$, INT0 to INT4) which can release the HALT mode may not be able to do so if they are input during the period CPU is shifting to the HALT mode (for about 5 clocks of f_{FPH}) with IDLE1 or STOP mode (IDLE2 is applicable to this case.) (In this case, an interrupt request is kept on hold internally.)

If another interrupt is generated after it has shifted to HALT mode completely, halt status can be released without difficulty. The priority of this interrupt is compare with that of the interrupt kept on hold internally and the interrupt with higher priority is handled first followed by the other interrupt.

8. Package Dimensions

P-QFP80-1420-0.80B

Unit: mm

