TOSHIBA Original CMOS 32-Bit Microcontroller

TLCS-900/H1 Series

TMP92FD23AFG
TMP92FD23ADFG

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”.

CMOS 32-Bit Microcontrollers

TMP92FD23AFG/ TMP92FD23ADFG

1. Outline and Device Characteristics

The TMP92FD23A is a high-speed advanced 32-bit Microcontroller developed for controlling equipment which processes mass data.

The TMP92FD23A has a high-performance CPU (900/H1 CPU) and various built-in I/Os.

The TMP92FD23AFG and TMP92FD23ADFG are housed in a 100-pin flat package.

Device characteristics are as follows:

(1) CPU: 32-bit CPU (900/H1 CPU)

- Compatible with 900/L1 instruction code
- 16 Mbytes of linear address space
- General-purpose register and register banks
- Micro DMA: 8 channels (250 ns/4 bytes at f_{SYS} = 20 MHz, best case)

(2) Minimum instruction execution time: 50 ns (at f_{SYS} = 20 MHz)

(3) Internal memory

- Internal RAM: 32-Kbytes
- Internal ROM: 512-Kbytes Flash memory
  4Kbytes mask ROM (used for booting)

(4) External memory expansion

- Expandable up to 16 Mbytes (Shared program/data area)
- Can simultaneously support 8- or 16-bit width external data bus
  ･･･Dynamic data bus sizing
- Separate bus system

(5) Memory controller

- Chip select output: 4 channels

(6) 8-bit timers: 6 channels

(7) 16-bit timers: 2 channels

(8) General-purpose serial interface: 3 channels

- UART/synchronous mode: 3 channels (channel 0 , 1 and 2)
- IrDA ver.1.0 (115 kbps) mode selectable: 3 channels (channel 0 , 1 and 2)

(9) Serial bus interface: 2 channels

- I^{2}C bus mode
- Clock synchronous mode

(10) High Speed serial interface: 1 channels

(11) 10-bit AD converter: 12 channels

(12) Watchdog timer

(13) Special timer for CLOCK

(14) Key-on wake up (only for HALT release):8 channels

(15) Program patch logic: 8 banks

(16) Interrupts: 51 interrupts

- 9 CPU interrupts: Software interrupt instruction and illegal instruction
- 33 internal interrupts: Seven selectable priority levels
- 9 external interrupts (INT0 to INT7 and $\overline{\text{NMI}}$): Seven selectable priority levels
  (INT0 to INT7 selectable edge or level interrupt)

(17) Input/output ports: 84 pins

(18) Standby function

- Three HALT modes: IDLE2 (Programmable), IDLE1, STOP

(19) Clock controller

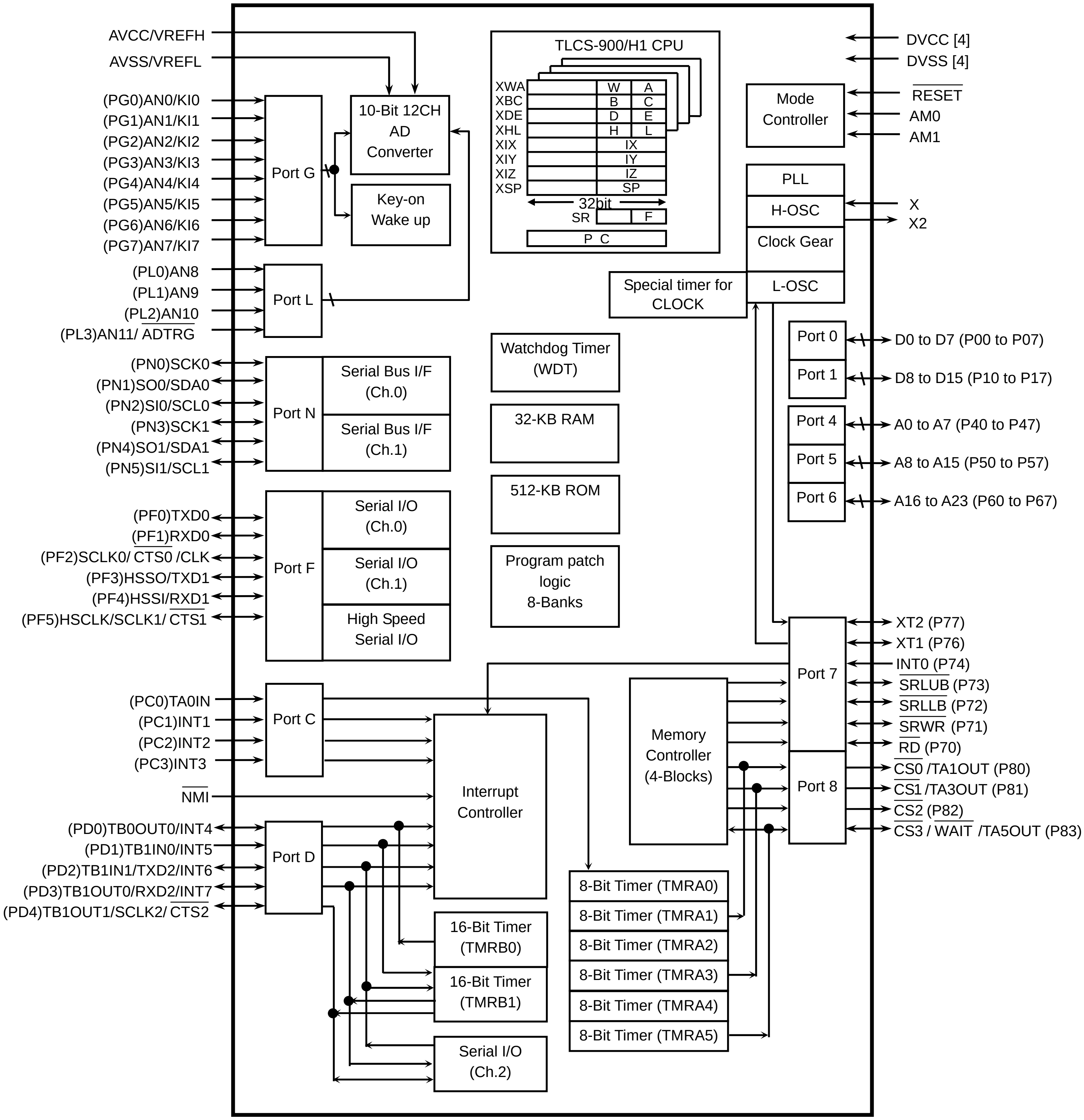
- Clock doubler (PLL)
- Clock gear function: Select high-frequency clock fc to fc/16
- Special timer for CLOCK (fs = 32.768 kHz)

(20) Operating voltage

- V_{CC} = 3.0 V to 3.6 V (fc max = 40 MHz)

(21) Package

- 100-pin QFP: LQFP100-P-1414-0.50F (TMP92FD23AFG)
  QFP100-P-1420-0.65A (TMP92FD23ADFG)



( ): Initial function after reset

Figure 1.1 TMP92FD23A Block Diagram

2. Pin Assignment and Functions

The assignment of input/output pins for the TMP92FD23A, their names and functions are as follows:

2.1 Pin Assignment Diagram

Figure 2.1.1 shows the pin assignment of the TMP92FD23AFG.

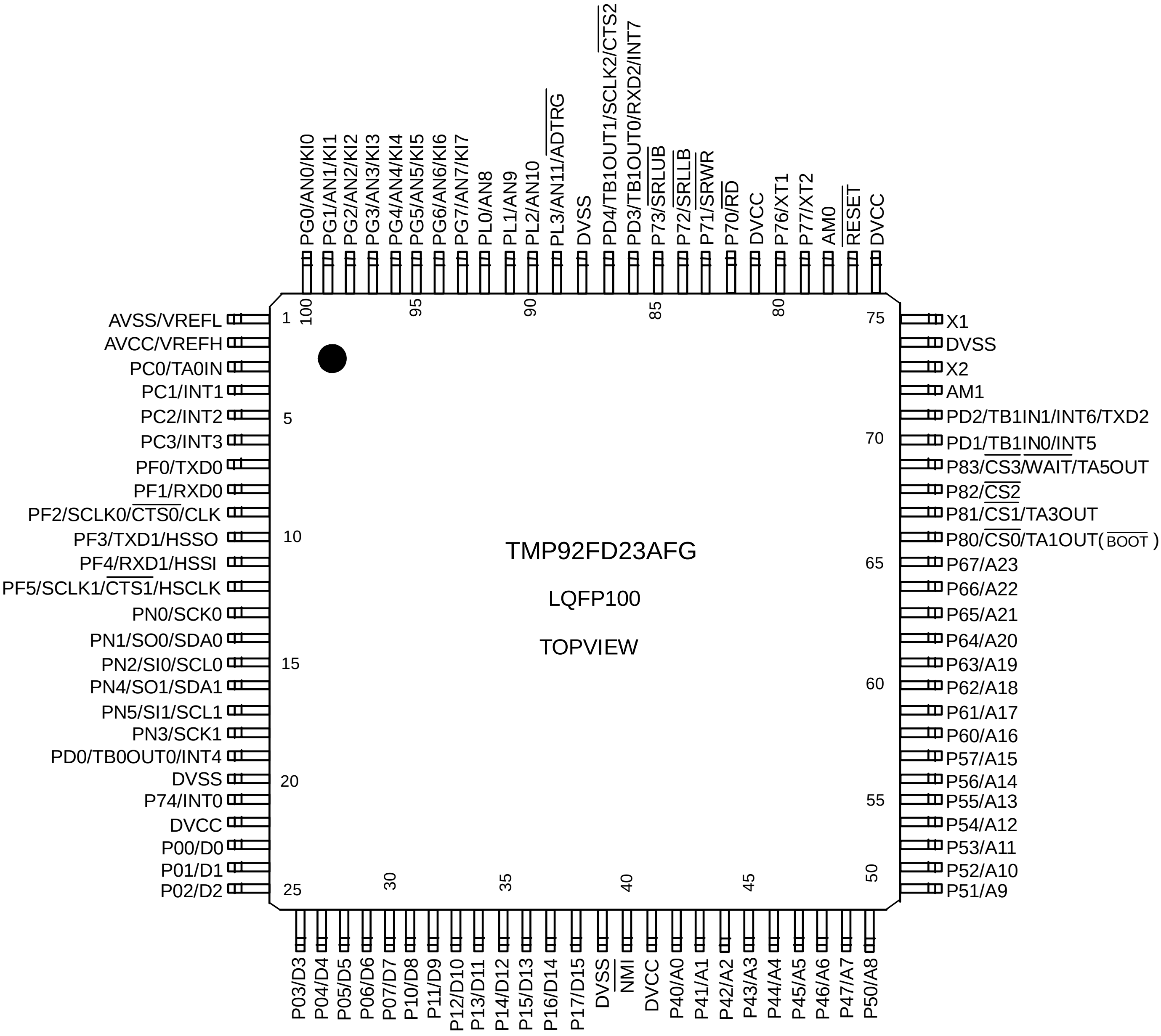


Figure 2.1.1 Pin Assignment Diagram (100-pin LQFP)

Figure 2.1.2 shows the pin assignment of the TMP92FD23ADFG.

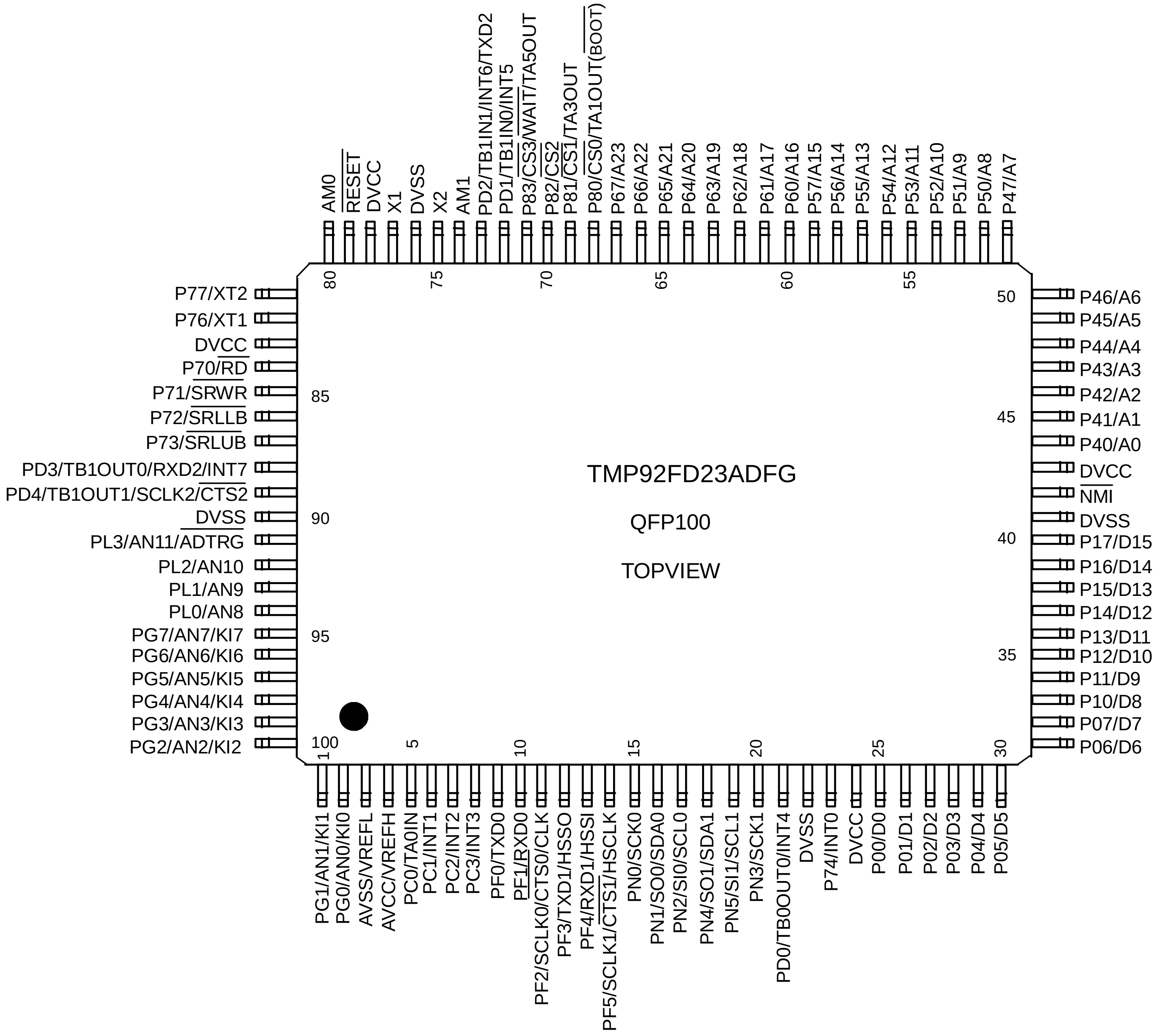


Figure 2.1.2 Pin Assignment Diagram (100-pin QFP)

2.2 Pin Names and Functions

The following table shows the names and functions of the input/output pins

Table 2.2.1 Pin Names and Functions (1/3)

Pin name	Number of Pin	I/O	Function
P00 to P07 D0 to D7	8	I/O I/O	Port 0: I/O port Input or output specifiable in units of bits Data: Data bus 0 to 7
P10 to P17 D8 to D15	8	I/O I/O	Port 1: I/O port Input or output specifiable in units of bits Data: Data bus 8 to 15
P40 to P47 A0 to A7	8	I/O Output	Port 4: I/O port Input or output specifiable in units of bits Address: Address bus 0 to 7
P50 to P57 A8 to A15	8	I/O Output	Port 5: I/O port Input or output specifiable in units of bits Address: Address bus 8 to 15
P60 to P67 A16 to A23	8	I/O Output	Port 6: I/O port Input or output specifiable in units of bits Address: Address bus 16 to 23
P70 $\overline{\text{RD}}$	1	I/O Output	Port 70: I/O port (Schmitt input, with pull-up resistor) Read: Outputs strobe signal for read external memory.
P71 $\overline{\text{SRWR}}$	1	I/O Output	Port 71: I/O port (Schmitt input, with pull-up resistor) Write enable for SRAM: Strobe signal for wiritng data.
P72 $\overline{\text{SRLLB}}$	1	I/O Output	Port 72: I/O port (Schmitt input, with pull-up resistor) Data enable for SRAM on pins D0 to D7
P73 $\overline{\text{SRLUB}}$	1	I/O Output	Port 73: I/O port (Schmitt input, with pull-up resistor) Data enable for SRAM on pins D8 to D15
P74 INT0	1	Input Input	Port 74: Input port (Schmitt input) Interrupt request pin 0 : Interrupt request pin with programmable level/rising/falling edge
P76 XT1	1	I/O Input	Port 76: I/O port (Open drain output) Low-frequency oscillator connection Input pins
P77 XT2	1	I/O Output	Port 77: I/O port (Open drain output) Low-frequency oscillator connection Output pins
P80 $\overline{\text{CS0}}$ TA1OUT ($\overline{\text{BOOT}}$ Note)	1	Output Output Output Input	Port 80: Output port Chip select 0: Outputs “Low” when address is within specified address area 8-bit timer 1 Output: Output pin of 8-bit timer TMRA0 or TMRA1 This pin sets single boot mode (only during reset). (Note) The function of TMP92FD23A.
P81 $\overline{\text{CS1}}$ TA3OUT	1	Output Output Output	Port 81: Output port Chip select 1: Outputs “Low” when address is within specified address area 8-bit timer 3 Output: Output pin of 8-bit timer TMRA2 or TMRA3
P82 $\overline{\text{CS2}}$	1	Output Output	Port 82: Output port Chip select 2: Outputs “Low” when address is within specified address area
P83 $\overline{\text{CS3}}$ TA5OUT $\overline{\text{WAIT}}$	1	I/O Output Output Input	Port 83: I/O port (Schmitt input) Chip select 3: Outputs “Low” when address is within specified address area 8-bit timer 5 Output: Output pin of 8-bit timer TMRA4 or TMRA5 Wait: Signal used to request CPU bus wait
PC0 TA0IN	1	Input Input	Port C0: Input port (Schmitt input) 8-bit timer 0 input: Input pin of 8-bit timer TMRA0
PC1 INT1	1	Input Input	Port C1: Input port (Schmitt input) Interrupt request pin 1 : Interrupt request pin with programmable level/rising/falling edge
PC2 INT2	1	Input Input	Port C2: Input port (Schmitt input) Interrupt request pin 2 : Interrupt request pin with programmable level/rising/falling edge
PC3 INT3	1	Input Input	Port C3: Input port (Schmitt input) Interrupt request pin 3 : Interrupt request pin with programmable level/rising/falling edge

Table 2.2.2 Pin Names and Functions (2/3)

Pin name	Number of Pin	I/O	Function
PD0 TB0OUT0 INT4	1	I/O Output Input	Port D0: I/O port (Schmitt input) 16-bit timer 0 output 0: Output pin of 16-bit timer TMRB0 Interrupt request pin 4 : Interrupt request pin with programmable level/rising/falling edge
PD1 TB1IN0 INT5	1	Input Input Input	Port D1: Input port (Schmitt input) 16-bit timer 1 input 0: Input of count/capture trigger in 16-bit timer TMRB1 Interrupt request pin 5 : Interrupt request pin with programmable level/rising/falling edge
PD2 TB1IN1 TXD2 INT6	1	I/O Input Output Input	Port D2: I/O port (Schmitt input) 16-bit timer 1 input 1: Input of count/capture trigger in 16-bit timer TMRB1 Serial 2 send data: Open drain output programmable Interrupt request pin 6 : Interrupt request pin with programmable level/rising/falling edge
PD3 TB1OUT0 RXD2 INT7	1	I/O Output Input Input	Port D3: I/O port (Schmitt input) 16-bit timer 1 output 0: Output pin of 16-bit timer TMRB1 Serial 2 receive data Interrupt request pin 7 : Interrupt request pin with programmable level/rising/falling edge
PD4 TB1OUT1 SCLK2 $\overline{\text{CTS2}}$	1	I/O Output I/O Input	Port D4: I/O port (Schmitt input) 16-bit timer 1 output 1: Output pin of 16-bit timer TMRB1 Serial 2 clock I/O Serial 2 data send enable (Clear to send)
PF0 TXD0	1	I/O Output	Port F0: I/O port (Schmitt input) Serial 0 send data: Open drain output programmable
PF1 RXD0	1	I/O Input	Port F1: I/O port (Schmitt input) Serial 0 receive data
PF2 SCLK0 $\overline{\text{CTS0}}$ CLK	1	I/O I/O Input Output	Port F2: I/O port (Schmitt input) Serial 0 clock I/O Serial 0 data send enable (Clear to send) Clock: System Clock output
PF3 TXD1 HSSO	1	I/O Output Output	Port F3: I/O port (Schmitt input) Serial 1 send data: Open drain output programmable High speed Serial send data
PF4 RXD1 HSSI	1	I/O Input Input	Port F4: I/O port (Schmitt input) Serial 1 receive data High speed Serial receive data
PF5 SCLK1 $\overline{\text{CTS1}}$ HSCLK	1	I/O I/O Input Output	Port F5: I/O port (Schmitt input) Serial 1 clock I/O Serial 1 data send enable (Clear to send) High speed Serial clock output
PG0 to PG7 AN0 to AN7 KI0 to KI7	8	Input	Port G: Input port (Schmitt input) Analog input 0 to 7: Pin used to input to AD conveter Key input 0 to 7: Pin used of key-on wakeup 0 to 7
PL0 to PL3 AN8 to AN11 $\overline{\text{ADTRG}}$	4	Input	Port L: Input port (Schmitt input) Analog input 8 to 11: Pin used to input to A/D conveter A/D trigger: Signal used for request A/D start (Shared with PL3)

Table 2.2.3 Pin Names and Functions (3/3)

Pin name	Number of Pin	I/O	Function
PN0 SCK0	1	I/O I/O	Port N0: I/O port (Schmitt input) Serial bus interface 0 clock I/O data at SIO mode
PN1 SO0 SDA0	1	I/O Output I/O	Port N1: I/O port (Schmitt input, Open drain output) Serial bus interface 0 send data at SIO mode Serial bus interface 0 send/receive data at I^2C mode
PN2 SI0 SCL0	1	I/O Input I/O	Port N2: I/O port (Schmitt input, Open drain output) Serial bus interface 0 receive data at SIO mode Serial bus interface 0 clock I/O data at I^2C mode
PN3 SCK1	1	I/O I/O	Port N3: I/O port (Schmitt input) Serial bus interface 1 clock I/O data at SIO mode
PN4 SO1 SDA1	1	I/O Output I/O	Port N4: I/O port (Schmitt input, Open drain output) Serial bus interface 1 send data at SIO mode Serial bus interface 1 send/receive data at I^2C mode
PN5 SI1 SCL1	1	I/O Input I/O	Port N5: I/O port (Schmitt input, Open drain output) Serial bus interface 1 receive data at SIO mode Serial bus interface 1 clock I/O data at I^2C mode
$\overline{\text{NMI}}$	1	Input	Non-maskable interrupt request pin: Interrupt request pin with programmable falling edge level or with both edge levels programmable (Schmitt input)
AM0, AM1	2	Input	Operation mode: Fixed to AM1 = “1” and AM0 = “1”
X1 / X2	2	I/O	High-frequency oscillator connection I/O pins
$\overline{\text{RESET}}$	1	Input	Reset: Intializes TMP92FD23A (Schmitt input, with pull-up resistor)
AVCC / VREFH	1	Input	Pin used to both power supply pin for AD converter and standard power supply for AD converter (H)
AVSS / VREFL	1	Input	Pin used to both GND pin for AD converter (0 V) and standard power supply pin for AD converter (L)
DVCC	4	–	Power supply pin (All DVCC pins should be connected with the power supply pin)
DVSS	4	–	GND pins (0 V) (All DVSS pins shold be connected with GND(0V))

3. Operation

This section describes the basic components, functions and operation of the TMP92FD23A.

3.1 CPU

The TMP92FD23A contains an advanced high-speed 32-bit CPU (TLCS-900/H1 CPU)

3.1.1 CPU Outline

The TLCS-900/H1 CPU is a high-speed, high-performance CPU based on the TLCS-900/L1 CPU. The TLCS-900/H1 CPU has an expanded 32-bit internal data bus to process instructions more quickly.

The following is an outline of the CPU:

Table 3.1.1 TMP92FD23A Outline

Parameter	TMP92FD23A
Width of CPU address bus	24 bits
Width of CPU data bus	32 bits
Internal operating frequency	Max 20 MHz
Minimum bus cycle	1-clock access (50 ns at f_{SYS} = 20MHz)
Internal RAM	32-bit 1-clock access
Internal ROM	32-bit interleave 2-1-1-1-clock access
Internal I/O	8-bit 2-clock access
External SRAM, Masked ROM	8- or 16-bit 2-clock access (waits can be inserted)
Minimum instruction execution cycle	1-clock (50 ns at f_{SYS} =20MHz)
Conditional jump	2-clock (100 ns at f_{SYS} =20MHz)
Instruction queue buffer	12 bytes
Instruction set	Compatible with TLCS-900/L1 (LDX instruction is deleted)
CPU mode	Maximum mode only
Micro DMA	8 channels

3.1.2 Reset Operation

When resetting the TMP92FD23A, ensure that 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 low for at least 20 system clocks (64 μs at fc = 10 MHz).

At reset, since the clock doubler (PLL) is bypassed and the clock-gear is set to 1/16, the system clock operates at 312.5 KHz (fc = 10 MHz).

When the reset has been accepted, the CPU performs the following:

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

PC<7:0>	←	data in location FFFF00H
PC<15:8>	←	data in location FFFF01H
PC<23:16>	←	data in location FFFF02H

- Sets the stack pointer (XSP) to 00000000H.
- Sets bits <IFF2:0> of the status register (SR) to 111 (thereby setting the interrupt level mask register to level 7).
- Clears bits <RFP1:0> of the status register to 00 (there by selecting register bank 0).

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

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.

A $\overline{\text{RESET}}$ input terminal becomes "High", if reset release is carried out, a built-in FlashROM warm-up circuit (notes) will start operation, and internal reset will be canceled after the end of the circuit of operation.

The operation of memory controller cannot be insured until power supply becomes stable after power-on reset. The external RAM data provided before turning on the TMP92FD23A may be spoiled because the control signals are unstable until power supply becomes stable after power-on reset.

Note: The warm-up time of build-in FlashROM into becomes it as follows.

at f_{OSCH} = 10 MHz
409.6μs (2^{12} / f_{OSCH})

Figure 3.1.1 shows the example of operating the reset timing of TMP92FD23A.

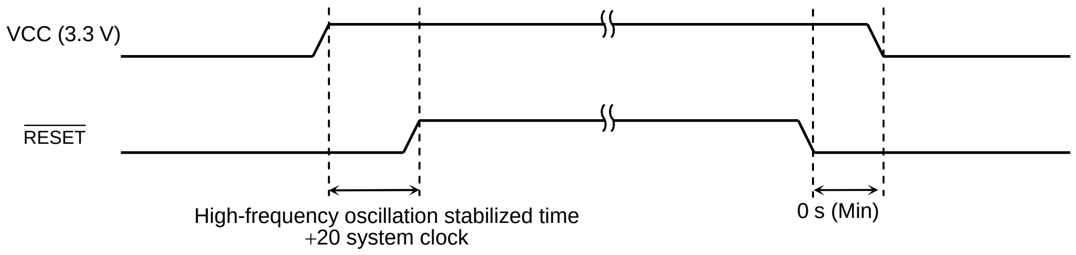

VCC (3.3 V)
$\overline{\text{RESET}}$
High-frequency oscillation stabilized time
+20 system clock
0 s (Min)

Figure 3.1.1 Power on Reset Timing Example

3.1.3 Setting of AM0 and AM1

Set AM1 and AM0 pins as shown in Table 3.1.2 according to system usage.

Table 3.1.2 Operation Mode Setup Table

Operation Mode	Mode Setup Input Pin			
	$\overline{\text{RESET}}$	AM1	AM0	$\overline{\text{BOOT}}$
Internal ROM starting	↗	1	1	1
Single-boot mode		1	1	0

3.2 Memory Map

Figure 3.2.1 is a memory map of the TMP92FD23A.

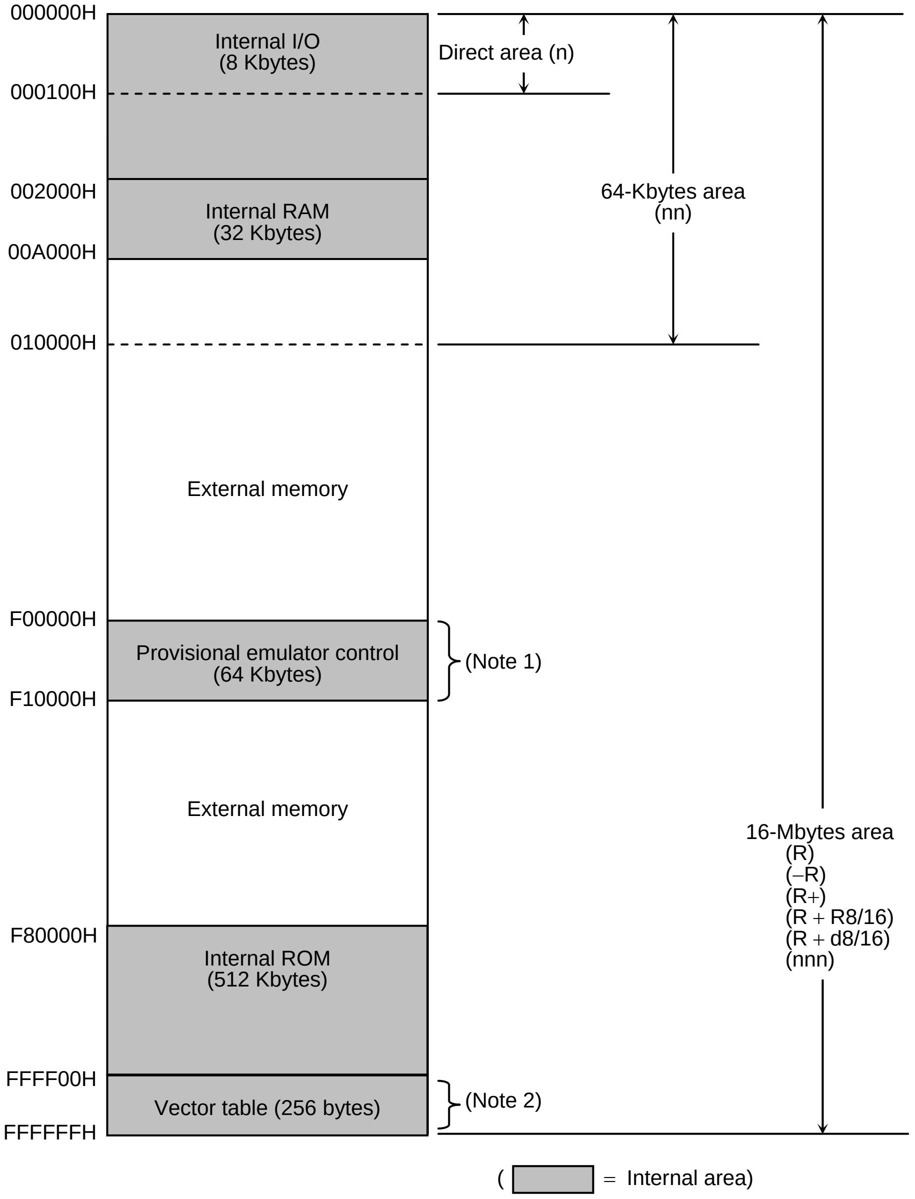


Figure 3.2.1 Memory Map

Note 1: The Provisional emulator control area, mapped F00000H to F0FFFFH after reset, is for emulator use and so is not available. When emulator $\overline{\text{SRWR}}$ signal and $\overline{\text{RD}}$ signal are asserted, this area is accessed. Ensure external memory is used.

Note 2: Do not use the last 16-byte area (FFFFF0H to FFFFFFH). This area is reserved for an emulator.

3.3 Clock Function and Stand-by Function

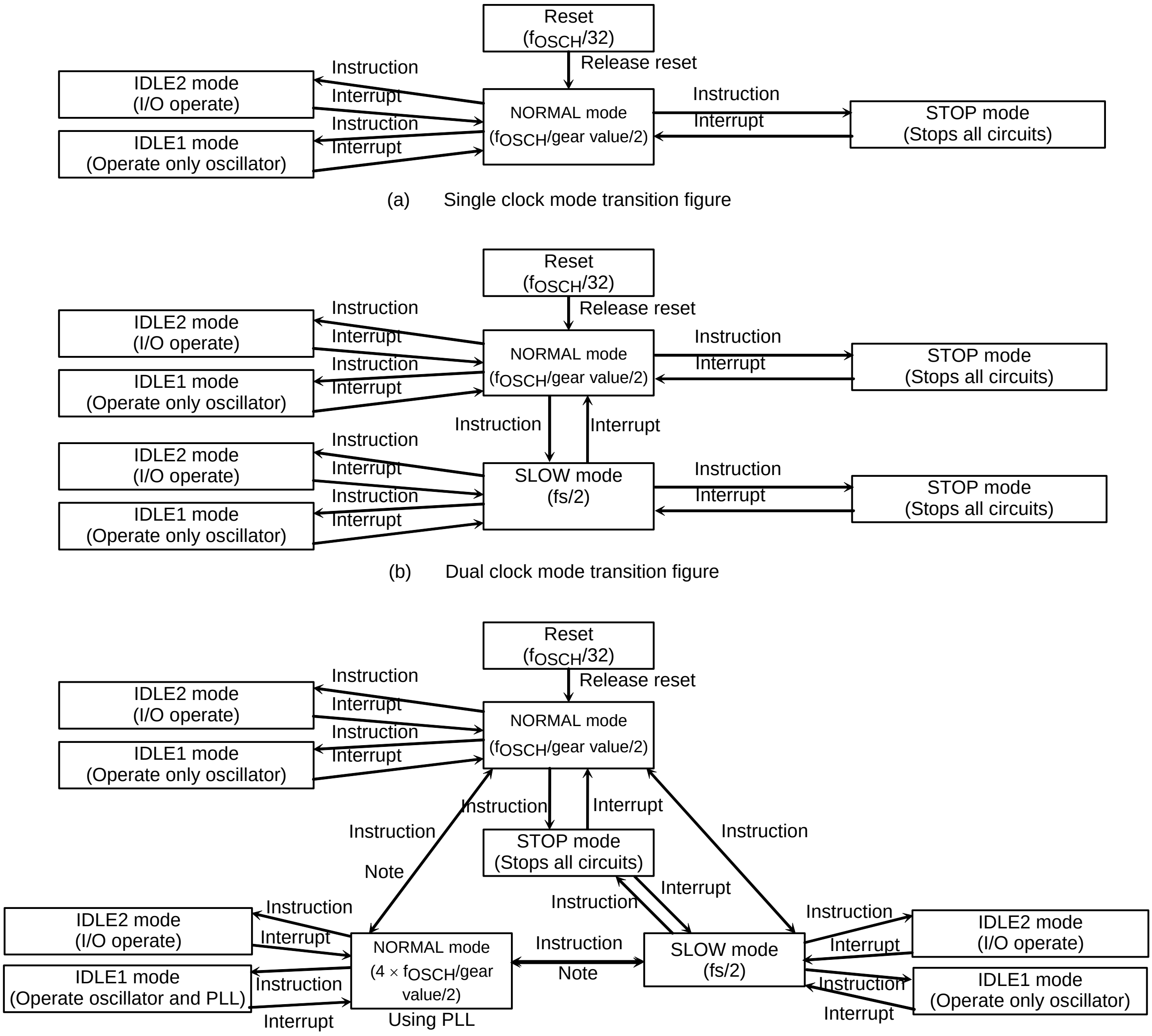
The TMP92FD23A contains (1) clock gear, (2) clock doubler (PLL), (3) stand-by controller and (4) noise reduction circuits. They are used for low power, low noise systems.

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 Clock doubler (PLL)
3.3.5 Noise reduction circuits
3.3.6 Stand-by controller

The clock operating modes are as follows: (a) single clock mode (X1, X2 pins only), (b) dual clock mode (X1, X2, XT1 and XT2 pins) and (c) triple clock mode (X1, X2, XT1 and XT2 pins and PLL).

Figure 3.3.1 shows a transition figure.



(a) Single clock mode transition figure

(b) Dual clock mode transition figure

(c) Triple clock mode transition figure

Note 1: It is not possible to control PLL in SLOW mode when shifting from SLOW mode to NORMAL mode with use of PLL. (PLL start up/stop/change write to PLLCR0<PLLON>, PLLCR1<FCSEL> register)

Note 2: When shifting from NORMAL mode with use of PLL to NORMAL mode, execute the following setting in the same order.

1) Change CPU clock (PLLCR0<FCSEL> ← “0”)

2) Stop PLL circuit (PLLCR1<PLLON> ← “0”)

Note 3: It is not possible to shift from NORMAL mode with use of PLL to STOP mode directly. NORMAL mode should be set once before shifting to STOP mode. (Stop the high-frequency oscillator after stopping PLL.)

Figure 3.3.1 System Clock Block Diagram

The clock frequency input from the X1 and X2 pins is called f_{OSCH} and the clock frequency input from the XT1 and XT2 pins is called fs. The clock frequency selected by SYSCR1<SYSCK> is called the clock f_{FPH}. The system clock f_{SYS} is defined as the divided clock of f_{FPH}, and one cycle of f_{SYS} is defined as one state.

3.3.1 Block Diagram of System Clock

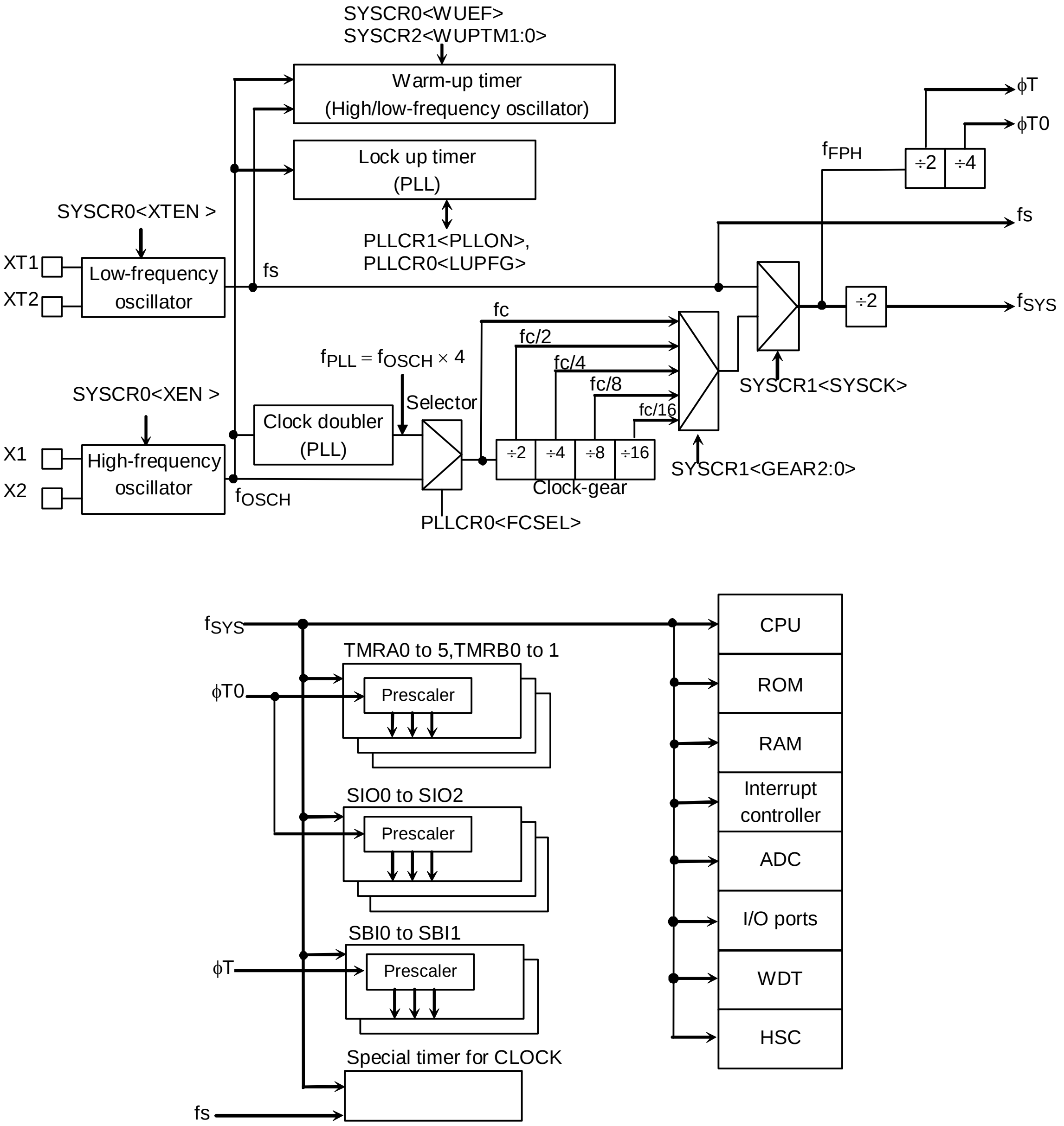


Figure 3.3.2 Block Diagram of System Clock

3.3.2 SFR

		7	6	5	4	3	2	1	0
SYSCR0 (10E0H)	Bit symbol	XEN	XTEN				WUEF		
	Read/Write	R/W					R/W		
	Reset State	1	0				0		
	Function	High-frequency oscillator (f_{OSCH}) 0: Stop 1: Oscillation	Low-frequency oscillator (fs) 0: Stop 1: Oscillation				Warm-up timer 0: Write don’t care 1: Write start timer 0: Read end warm-up 1: Read do not end warm-up		

		7	6	5	4	3	2	1	0
SYSCR1 (10E1H)	Bit symbol					SYSCK	GEAR2	GEAR1	GEAR0
	Read/Write					R/W			
	Reset State					0	1	0	0
	Function					Select system clock 0: fc 1: fs	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		

		7	6	5	4	3	2	1	0
SYSCR2 (10E2H)	Bit symbol	–		WUPTM1	WUPTM0	HALTM1	HALTM0		DRVE
	Read/Write	R/W		R/W					R/W
	Reset State	0		1	0	1	1		0
	Function	Always write “0”		Warm-up timer 00: Reserved 01: 2^{8}/input frequency 10: 2^{14}/input frequency 11: 2^{16}/input frequency		HALT mode 00: Reserved 01: STOP mode 10: IDLE1 mode 11: IDLE2 mode			1: The inside of STOP mode also drives a pin

Note 1: The unassigned registers, SYSCR0<bit5:3>, SYSCR0<bit1:0>, SYSCR1<bit7:4>, and SYSCR2<bit7:6,1> are read as undefined value.

Note 2: Low-frequency oscillator is enabled on reset.

Figure 3.3.3 SFR for System Clock

		7	6	5	4	3	2	1	0
EMCCR0 (10E3H)	Bit symbol	PROTECT					–	–	DRVOSCL
	Read/Write	R					R/W		
	Reset State	0					0	1	1
	Function	Protect flag 0: OFF 1: ON					Always write “0”	Always write “1”	fs oscillator driver ability 1: Normal 0: Weak
EMCCR1 (10E4H)	Bit symbol	Switch the protect ON/OFF by writing the following to 1st-KEY, 2nd-KEY 1st-KEY: write in sequence EMCCR1 = 5AH, EMCCR2 = A5H 2nd-KEY: write in sequence EMCCR1 = A5H, EMCCR2 = 5AH							
	Read/Write								
	Reset State								
	Function								
EMCCR2 (10E5H)	Bit symbol								
	Read/Write								
	Reset State								
	Function								

Note: When restarting the oscillator from the stop oscillation state (e.g. restarting the oscillator in STOP mode), set EMCCR0<DRVOSCH>, <DRVOSCL>= “1”.

Figure 3.3.4 SFR for System Clock

		7	6	5	4	3	2	1	0
PLLCR0 (10E8H)	Bit symbol		FCSEL	LUPFG					
	Read/Write		R/W	R					
	Reset State		0	0					
	Function		Select fc clock 0: f_{OSCH} 1: f_{PLL}	Lock up timer status flag 0: Not end 1: End					

Note: Ensure that the logic of PLLCR0<LUPFG> is different from 900/L1’s DFM.

		7	6	5	4	3	2	1	0
PLLCR1 (10E9H)	Bit symbol	PLLON							
	Read/Write	R/W							
	Reset State	0							
	Function	Control on/off 0: OFF 1: ON							

Figure 3.3.5 SFR for PLL

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 two oscillation circuits and a clock gear circuit for high-frequency (fc) operation. The register SYSCR1<SYSCK> changes the system clock to either fc or fs, SYSCR0<XEN> and SYSCR0<XTEN> control enabling and disabling of each oscillator, and SYSCR1<GEAR2:0> sets the high-frequency clock gear to either 1, 2, 4, 8 or 16 (fc, fc/2, fc/4, fc/8 or fc/16). These functions can reduce the power consumption of the equipment in which the device is installed.

The combination of settings <XEN> = 1, <SYSCK> = 0 and <GEAR2:0> = 100 will cause the system clock (f_{SYS}) to be set to fc/32 (fc/16 × 1/2) after reset.

For example, f_{SYS} is set to 0.3125 MHz when the 10 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<WUPTM1:0>.

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 Times

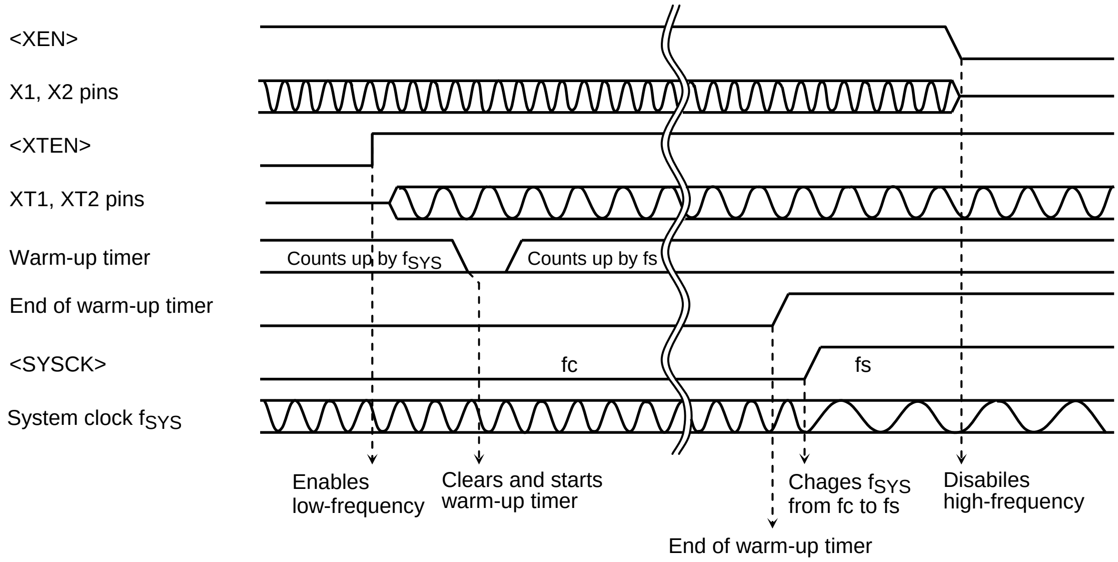
at f_{OSCH} = 10 MHz, fs = 32.768 kHz

Warm-up Time SYSCR2 <WUPTM1:0>	Change to Normal Mode	Change to Slow Mode
01 (2^8/frequency)	25.6 (μs)	7.8 (ms)
10 (2^{14}/frequency)	1.638 (ms)	500 (ms)
11 (2^{16}/frequency)	6.554 (ms)	2000 (ms)

Example 1: Setting the clock
Changing from high-frequency (fc) to low-frequency (fs).

```
SYSCR0   EQU    10E0H
SYSCR1   EQU    10E1H
SYSCR2   EQU    10E2H
         LD     (SYSCR2),  0 X 1 1 – – X – B  ;   Sets warm-up time to 2^16/fs.
         SET    6, (SYSCR0)                   ;   Enables low-frequency oscillation.
         SET    2, (SYSCR0)                   ;   Clears and starts warm-up timer.
WUP:     BIT    2, (SYSCR0)                   ; }
         JR     NZ, WUP                       ; } Detects stopping of warm-up timer.
         SET    3, (SYSCR1)                   ;   Changes fSYS from fc to fs.
         RES    7, (SYSCR0)                   ;   Disables high-frequency oscillation.
```

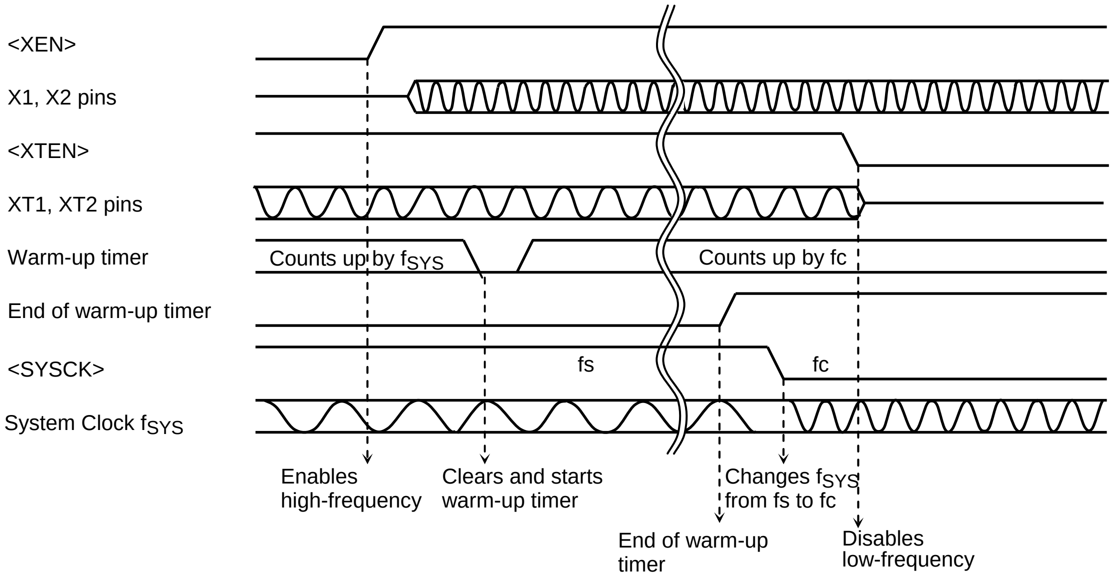
X: Don’t care, –: No change



Example 2: Setting the clock
Changing from low-frequency (fs) to high-frequency (fc).

```
SYSCR0      EQU     10E0H
SYSCR1      EQU     10E1H
SYSCR2      EQU     10E2H
            LD     (SYSCR2),  0 X 1 0 − − X − B  ;    Sets warm-up time to 2^14/fc.
            SET     7, (SYSCR0)                  ;    Enables high-frequency oscillation.
            SET     2, (SYSCR0)                  ;    Clears and starts warm-up timer.
WUP:        BIT     2, (SYSCR0)                  ; }
            JR      NZ, WUP                      ; }  Detects stopping of warm-up timer.
            RES     3, (SYSCR1)                  ;    Changes fSYS from fs to fc.
            RES     6, (SYSCR0)                  ;    Disables low-frequency oscillation.
```

X: Don't care, −: No change



(2) Clock gear controller

f_{FPH} is set according to the contents of the clock gear select register SYSCR1<GEAR2:0> to either fc, fc/2, fc/4, fc/8 or fc/16. Using the clock gear to select a lower value of f_{FPH} reduces power consumption.

Example 3: Changing to a high-frequency gear

```
SYSCR1    EQU    10E1H

          LD     (SYSCR1), XXXX0001B    ;    Changes fSYS to fc/2.
     X: Don't care
```

(High-speed clock gear changing)

To change the clock gear, write the register value to the SYSCR1<GEAR2:0> register.It is necessary for the warm-up time to elapse before the change occurs after writing the register value.

There is the possibility that the instruction following the clock gear changing instruction is executed by the clock gear before changing.To execute the instruction following the clock gear switching instruction by the clock gear after changing, input the dummy instruction as follows (instruction to execute the write cycle).

Example:

```
SYSCR1    EQU    10E1H
          LD     (SYSCR1), XXXX0010B    ;    Changes fSYS to fc/4.
          LD     (DUMMY), 00H           ;    Dummy instruction
        | Instruction to be executed after clock gear has changed |
```

3.3.4 Clock Doubler (PLL)

PLL outputs the f_{PLL} clock signal, which is four times as fast as f_{OSCH}. A low-speed-frequency oscillator can be used, even though the internal clock is high-frequency.

A reset initializes PLL to stop status, so setting to PLLCR0, PLLCR1 register is needed before use.

As with an oscillator, this circuit requires time to stabilize. This is called the lock up time and it is measured by a 16-stage binary counter. Lock up time is about 1.6 ms at f_{OSCH} = 10 MHz.

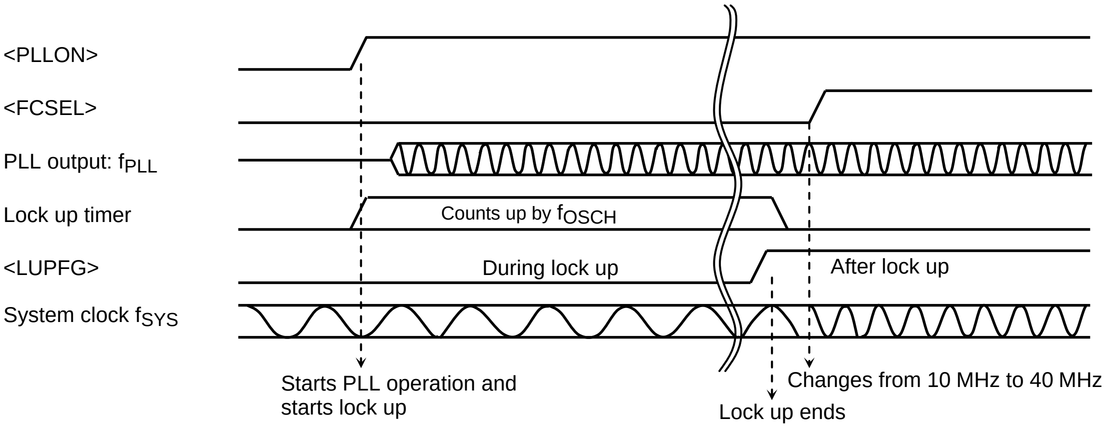
Note 1: Input frequency range for PLL
The input frequency range (High-frequency oscillation) for PLL is as follows:
f_{OSCH} = 6 to 10 MHz (V_{CC} = 3.0 to 3.6 V)

Note 2: PLLCR0<LUPFG>
The logic of PLLCR0<LUPFG> is different from 900/L1’s DFM.
Exercise care in determining the end of lock up time.

The following is an example of settings for PLL starting and PLL stopping.

Example 1: PLL starting

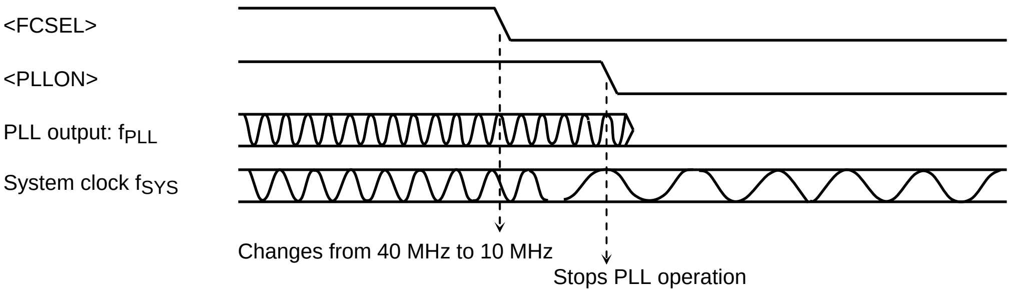
```
PLLCR0      EQU     10E8H
PLLCR1      EQU     10E9H
            LD      (PLLCR1),   1 X X X X X X X B  ;    Enables PLL operation and starts lock up.
LUP:        BIT     5, (PLLCR0)                    ;  }
            JR      Z, LUP                         ;  }  Detects end of lock up.
            LD      (PLLCR0),   X 1 X X X X X X B  ;    Changes fc from 10 MHz to 40 MHz.
X: Don’t care
```



Example 2: PLL stopping

```
PLLCR0      EQU     10E8H
PLLCR1      EQU     10E9H
            LD       (PLLCR0), X0XXXXXXB     ;   Changes fc from 40 MHz to10 MHz.
            LD       (PLLCR1), 0XXXXXXXB     ;   Stop PLL.
```

X: Don’t care



Limitations on the use of PLL

1. It is not possible to execute PLL enable/disable control in the SLOW mode (fs) (writing to PLLCR0 and PLLCR1).
   PLL should be controlled in the NORMAL mode.

2. When stopping PLL operation during PLL use, execute the following settings in the same order.

```
LD      (PLLCR0), 00H          ;   Change the clock fPLL to fOSCH
LD      (PLLCR1), 00H          ;   PLL stop
```

3. When stopping the high-frequency oscillator during PLL use, stop PLL before stopping the high-frequency oscillator.

Examples of settings are shown below:

(1) Start up/change control

(OK) Low-frequency oscillator operation mode (fs) (high-frequency oscillator STOP) → High-frequency oscillator start up → High-frequency oscillator operation mode (f_{OSCH}) → PLL start up → PLL use mode (f_{PLL})

```
      LD     (SYSCR0),   1 1 – – – 1 – – B ;  High-frequency oscillator start/warm-up start
WUP:  BIT    2, (SYSCR0)                   ; }
      JR     NZ, WUP                       ; } Check for warm-up end flag
      LD     (SYSCR1),   – – – – 0 – – – B ;  Change the system clock fs to fOSCH
      LD     (PLLCR1),   1 – – – – – – – B ;  PLL start-up/lock up start
LUP:  BIT    5, (PLLCR0)                   ; }
      JR     Z, LUP                        ; } Check for lock up end flag
      LD     (PLLCR0),   – 1 – – – – – – B ;  Change the system clock fOSCH to fPLL
```

(OK) Low-frequency oscillator operation mode (fs) (high-frequency oscillator Operate) → High-frequency oscillator operation mode (f_{OSCH}) → PLL start up → PLL use mode (f_{PLL})

```
      LD     (SYSCR1),   – – – – 0 – – – B ;  Change the system clock fs to fOSCH
      LD     (PLLCR1),   1 – – – – – – – B ;  PLL start-up/lock up start
LUP:  BIT    5, (PLLCR0)                   ; }
      JR     Z, LUP                        ; } Check for lock up end flag
      LD     (PLLCR0),   – 1 – – – – – – B ;  Change the system clock fOSCH to fPLL
```

(Error) Low-frequency oscillator operation mode (fs) (high-frequency oscillator STOP) → High-frequency oscillator start up → PLL start up → PLL use mode (f_{PLL})

```
      LD     (SYSCR0),   1 1 – – – 1 – – B ;  High-frequency oscillator start/warm-up start
WUP:  BIT    2, (SYSCR0)                   ; }
      JR     NZ, WUP                       ; } Check for warm-up end flag
      LD     (PLLCR1),   1 – – – – – – – B ;  PLL start-up/lock up start
LUP:  BIT    5, (PLLCR0)                   ; }
      JR     Z, LUP                        ; } Check for lock up end flag
      LD     (PLLCR0),   – 1 – – – – – – B ;  Change the internal clock fOSCH to fPLL
      LD     (SYSCR1),   – – – – 0 – – – B ;  Change the system clock fs to fPLL
```

(2) Change/stop control

(OK) PLL use mode (f_{PLL}) → High-frequency oscillator operation mode (f_{OSCH}) → PLL Stop → Low-frequency oscillator operation mode (fs) → High-frequency oscillator stop

```
LD     (PLLCR0),      – 0 – – – – – – B ;  Change the system clock fPLL to fOSCH
LD     (PLLCR1),      0 – – – – – – – B ;  PLL stop
LD     (SYSCR1),      – – – – 1 – – – B ;  Change the system clock fOSCH to fs
LD     (SYSCR0),      0 – – – – – – – B ;  High-frequency oscillator stop
```

(Error) PLL use mode (f_{PLL}) → Low-frequency oscillator operation mode (fs) → PLL stop → High-frequency oscillator stop

```
LD     (SYSCR1),      – – – – 1 – – – B ;  Change the system clock fPLL to fs
LD     (PLLCR0),      – 0 – – – – – – B ;  Change the internal clock (fc) fPLL to fOSCH
LD     (PLLCR1),      0 – – – – – – – B ;  PLL stop
LD     (SYSCR0),      0 – – – – – – – B ;  High-frequency oscillator stop
```

(OK) PLL use mode (f_{PLL}) → Set the STOP mode → High-frequency oscillator operation mode (f_{OSCH}) → PLL stop → Halt (High-frequency oscillator stop)

```
LD     (SYSCR2),      – – – – 0 1 – – B ;  Set the STOP mode
                                            (This command can be executed before use of PLL)
LD     (PLLCR0),      – 0 – – – – – – B ;  Change the system clock fPLL to fOSCH
LD     (PLLCR1),      0 – – – – – – – B ;  PLL stop
HALT                                    ;  Shift to STOP mode
```

(Error) PLL use mode (f_{PLL}) → Set the STOP mode → Halt (High-frequency oscillator stop)

```
LD     (SYSCR2),      – – – – 0 1 – – B ;  Set the STOP mode
                                            (This command can execute before use of PLL)
HALT                                    ;  Shift to STOP mode
```

3.3.5 Noise Reduction Circuits

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

(1) Reduced drivability for low-frequency oscillator

(2) SFR protection of register contents

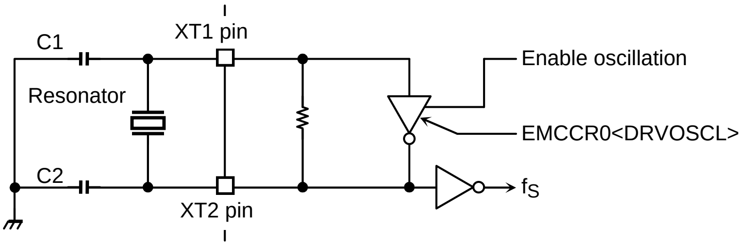
These functions need a setup by EMCCR0, EMCCR1, and EMCCR2 register.

(1) Reduced drivability for low-frequency oscillator

(Purpose)

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

(Block diagram)



(Setting method)

The drive ability of the oscillator is reduced by writing 0 to the EMCCR0<DRVOSCL> register. At reset, <DRVOSCL> is initialized to “1”.

(2) Runaway prevention using SFR protection register

(Purpose)

Prevention of program runaway caused by introduction of noise.

Write operations to a specified SFR are prohibited so that the program is protected from runaway caused by stopping of the clock or by changes to the memory control register (memory controller) which prevent fetch operations.

Runaway error handling is also facilitated by INTP0 interruption.

Specified SFR list

1. Memory controller
   B0CSL/H, B1CSL/H, B2CSL/H, B3CSL/H, BEXCSL/H
   MSAR0, MSAR1, MSAR2, MSAR3,
   MAMR0, MAMR1, MAMR2, MAMR3, PMEMCR
2. Clock gear
   SYSCR0, SYSCR1, SYSCR2, EMCCR0
4. PLL
   PLLCR0, PLLCR1

(Operation explanation)

Execute and release of protection (write operation to specified SFR) becomes possible by setting up a double key to EMCCR1 and EMCCR2 registers.

(Double key)

1st KEY: writes in sequence, 5AH at EMCCR1 and A5H at EMCCR2

2nd KEY: writes in sequence, A5H at EMCCR1 and 5AH at EMCCR2

Protection state can be confirmed by reading EMCCR0<PROTECT>.

At reset, protection becomes OFF.

INTP0 interruption also occurs when a write operation to the specified SFR is executed with protection in the ON state.

3.3.6 Stand-by Controller

(1) HALT modes and port drive register

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: only the CPU halts.

   The internal I/O is available to select operation during IDLE2 mode by setting the following register.

   Table 3.3.2 shows the register setting operation during IDLE2 mode.

Table 3.3.2 SFR Setting Operation during IDLE2 Mode

Internal I/O	SFR
TMRA01	TA01RUN<I2TA01>
TMRA23	TA23RUN<I2TA23>
TMRA45	TA45RUN<I2TA45>
TMRB0	TB0RUN<I2TB0>
TMRB1	TB1RUN<I2TB1>
SIO0	SC0MOD1<I2S0>
SIO1	SC1MOD1<I2S1>
SIO2	SC2MOD1<I2S2>
AD converter	ADMOD1<I2AD>
WDT	WDMOD<I2WDT>
SBI0	SBI0BR0<I2SBI0>
SBI1	SBI1BR0<I2SBI1>

2. IDLE1: Only the oscillator and the Special timer for CLOCK continue to operate.
3. STOP: All internal circuits stop operating.

The operation of each of the different HALT modes is described in Table 3.3.3.

Table 3.3.3 I/O Operation during HALT Modes

<table>
<tr><th colspan="2">HALT Mode</th><th>IDLE2</th><th>IDLE1</th><th>STOP</th></tr>
<tr><td colspan="2">SYSCR2<HALTM1:0></td><td>11</td><td>10</td><td>01</td></tr>
<tr><td rowspan="9">Block</td><td>CPU</td><td colspan="3">Stop</td></tr>
<tr><td>I/O ports</td><td>The state at the time of "HALT" instruction execution is held.</td><td colspan="2">Table 3.3.7 and Table 3.3.8 references</td></tr>
<tr><td>TMRA, TMRB</td><td rowspan="4">Available to select operation block</td><td colspan="2" rowspan="6">Stop</td></tr>
<tr><td>SIO, SBI</td></tr>
<tr><td>AD converter</td></tr>
<tr><td>WDT</td></tr>
<tr><td>Interrupt controller</td><td rowspan="3">Operate</td></tr>
<tr><td>HSC</td></tr>
<tr><td>Special timer for CLOCK</td><td>Operate</td><td></td></tr>
</table>

(2) How to release the HALT mode

These halt states can be released by resetting or requesting an interrupt. The halt release sources are determined by the combination between of the states of the interrupt mask register <IFF2:0> and the HALT modes. The details for releasing the halt status are shown in Table 3.3.4.

Release by interrupt requesting

The HALT mode release method depends on the status of the enabled interrupt .When the interrupt request level set before executing the HALT instruction exceeds the value of the interrupt mask register, the interrupt is processed depending on its status after the HALT mode is released, and the CPU status executing the instruction that follows the HALT instruction. When the interrupt request level set before executing the HALT instruction is less than the value of the interrupt mask register, HALT mode release is not executed. (in non-maskable interrupts, interrupt processing is processed after releasing the HALT mode regardless of the value of the mask register.) However only for INT0 to INT7, INTRTC interrupts, even if the interrupt request level set before executing the halt instruction is less than the value of the interrupt mask register, HALT mode release is executed. In this case, the interrupt is processed, and the CPU starts executing the instruction following the HALT instruction, but the interrupt request flag is held at “1”.

Release by resetting

Release of all halt statuses is executed by resetting.

When the STOP mode is released by RESET, it is necessary to allow enough resetting time (see Table 3.3.5) for operation of the oscillator to stabilize.

When releasing the HALT mode by resetting, the internal RAM data keeps the state before the HALT instruction is executed. However the other settings contents are initialized. (Releasing due to interrupts keeps the state before the HALT instruction is executed.)

Table 3.3.4 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, INT7 (Note 1)	♦	♦	♦*1	○	○	○*1
		INT5,INT6 (PORT) (Note 1)	♦	♦	♦*1	○	○	○*1
		INT5,INT6 (TMRB1)	♦	×	×	×	×	×
		INTTA0 to INTTA5	♦	×	×	×	×	×
		INTB00, INTTB01, INTTB10, INTTB11, INTTBO0, INTTBO1	♦	×	×	×	×	×
		INTRX0 to INTRX2, INTTX0 to INTTX2	♦	×	×	×	×	×
		INTAD	♦	×	×	×	×	×
		KWI	♦	♦	♦*1	△	△	△
		INTRTC	♦	♦	×	○	○	×
		INTSBE0 to INTSBE1	♦	×	×	×	×	×
		INTHSC	♦	×	×	×	×	×
	RESET		Initialize LSI					

♦: After clearing the HALT mode, CPU starts interrupt processing.

○: After clearing the HALT mode, CPU resumes executing starting from the instruction following the HALT instruction.

×: Cannot be used to release the HALT mode.

–: The priority level (interrupt request level) of non-maskable interrupts is fixed to 7, the highest priority level. This combination is not available.

△: Since KWI does not have a function as interruption, this combination does not exist.

*1: Release of the HALT mode is executed after warm-up time has elapsed.

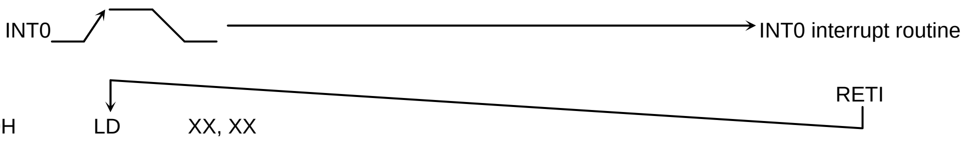
Note 1: When the HALT mode is cleared by an INT0 to 7 interrupt of the level mode in the interrupt enabled status, hold level “H” until starting interrupt processing. If level “L” is set before holding level “L”, interrupt processing is correctly started.

Note 2: Although a KWI can cancel all HALT mode states, the function as interruption does not have it.

Note 3: Specify the HSCSEL register when selecting INTTX1 or INTHSC interrupt with the same interrupt factor.

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

```
Address
8200H      LD       (P7FC), 10H          ;   Sets P74 to INT0 interrupt.
8203H      LD       (IIMC3), 00H         ;   Selects INT0 interrupt rising edge.
8206H      LD       (IIMC2), 00H         ;   Selects INT0 interrupt edge
8209H      LD       (INTE01), 06H            Sets INT0 interrupt level to 6.
820BH      EI      5                     ;   Sets interrupt level to 5 for CPU.
820EH      LD       (SYSCR2), 28H        ;   Sets HALT mode to IDLE1 mode.
820FH      HALT                          ;   Halts CPU.
```



```
8210H      LD      XX, XX
```

(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 3.3.6 illustrates an example of the timing for clearance of the IDLE2 mode halt state by an interrupt.

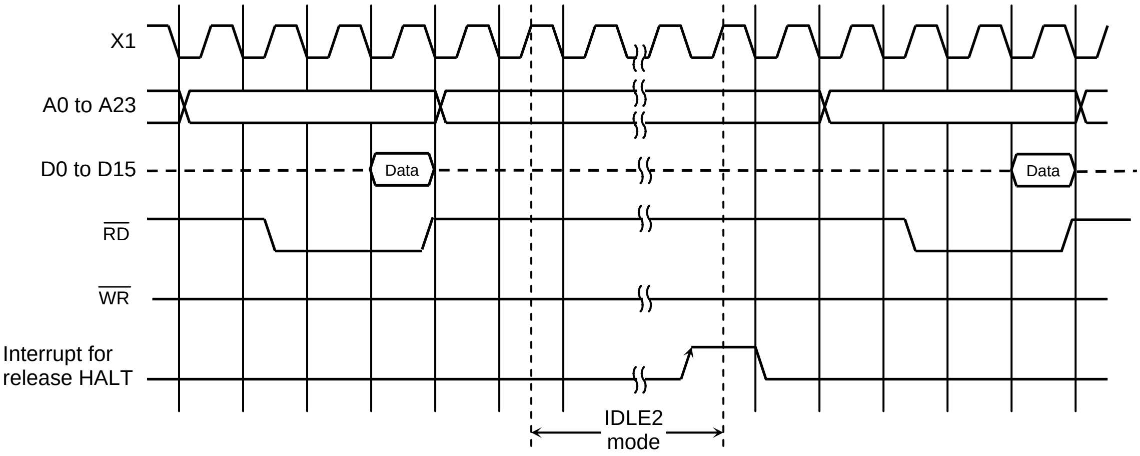


Figure 3.3.6 Timing Chart for IDLE2 Mode Halt State Cleared by Interrupt

2. IDLE1 mode

In IDLE1 mode, only the internal oscillator and Special timer for Clock continue to operate. The system clock 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 3.3.7 illustrates the timing for clearance of the IDLE1 mode halt state by an interrupt.

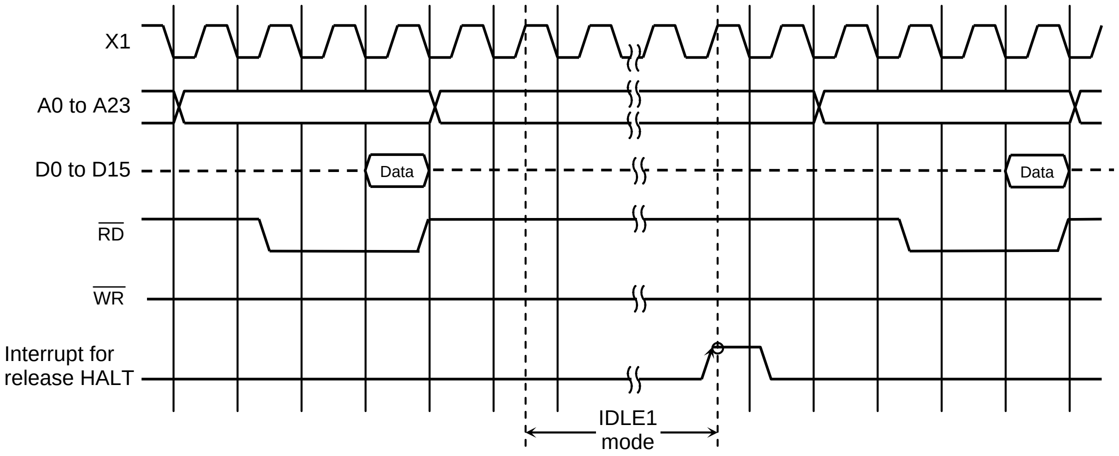


Figure 3.3.7 Timing Chart for IDLE1 Mode Halt State Cleared by Interrupt

3. STOP mode

When STOP mode is selected, all internal circuits stop, including the internal oscillator.

After STOP mode has been cleared system clock output starts when the warm-up time by the counter for a warm-up of internal oscillator and built-in FlashROM warm-up time.

The example of a setting of the Warm-up time at the time of STOP mode release is shown in Table 3.3.5. The warm-up time of built-in FlashROM is shown in Table 3.3.6.

Figure 3.3.8 illustrates the timing for clearance of the STOP mode halt state by an interrupt.

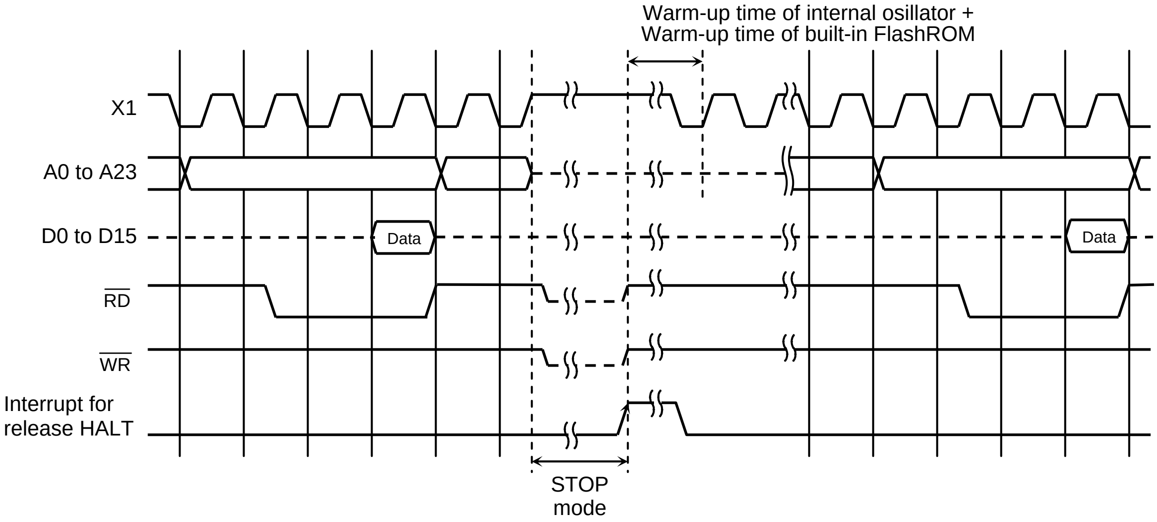


Figure 3.3.8 Timing Chart for STOP Mode Halt State Cleared by Interrupt

Table 3.3.5 Example of Warm-up Time after Releasing STOP Mode

at f_{OSCH} = 10 MHz, fs = 32.768 kHz

SYSCR1 <SYSCK>	SYSCR2<WUPTM1:0>		
	01 (2^8)	10 (2^{14})	11 (2^{16})
0 (fc)	25.6 μs	1.638 ms	6.554 ms
1 (fs)	7.8 ms	500 ms	2000 ms

Table 3.3.6 Example of Warm-up Time after Built-in FlashROM (at the time of STOP mode release)

at f_{OSCH} = 10 MHz, fs = 32.768 kHz

0 (fc)	409.6 μs (2^{12}/f_{OSCH})
1 (fs)	125 ms (2^{12}/fs)

Table 3.3.7 Input Buffer State Table

Port Name	Input Function Name		Input Buffer State								
			During Reset	When the CPU is operating		In HALT mode (IDLE1/2)		In HALT mode (STOP)			
								DRVE = 1		DRVE = 0	
				When used as Function pin	When used as Input pin	When used as Function pin	When used as Input pin	When used as Function pin	When used as Input pin	When used as Function pin	When used as Input pin
P00-P07	D0-D7		OFF	ON upon external read (*1)	ON	OFF	OFF	OFF	OFF	OFF	OFF
P10-P17	D8-D15										
P40-P47	–			OFF							
P50-P57	–										
P60-P67	–										
P70(*2)	–										
P71-P73 (*2)	–		ON								
P74	INT0			ON		ON		ON		ON	
P76	XT1	Oscillator	OFF		OFF			OFF		OFF	
		Port		OFF	ON	OFF					
P77	–			–		–		–		–	
P83	WAIT		ON	ON		OFF		OFF		OFF	
PC0	TA0IN					ON		ON			
PC1	INT1									ON	
PC2	INT2										
PC3	INT3										
PD0	INT4										
PD1	INT5										
	TB1IN0									OFF	
PD2	INT6									ON	
	TB1IN1									OFF	
PD3	INT7									ON	
	RXD2									OFF	
PD4	SCLK2, CTS2										
PF0	–			OFF		OFF					
PF1	RXD0			ON		ON					
PF2	SCLK0, CTS0										
PF3	–			OFF		OFF					
PF4	RXD1, HSSI			ON		ON					
PF5	SCLK1, CTS1										
PG0-PG7	AN0-AN7(*3)		OFF	OFF		OFF					
	KI0-KI7			ON		ON		ON		ON	
PL0-PL2	AN8-AN10(*3)			OFF		OFF		OFF		OFF	
PL3	AN11(*3)										
	ADTRG			ON		ON		ON			
PN0	SCK0		ON								
PN1	SDA0										
PN2	SI0, SCL0										
PN3	SCK1										
PN4	SDA1										
PN5	SI1, SCL1										
NMI	–				–		–		–	ON	–
AM0,AM1	–										
X1	–							OFF		OFF	
RESET	–							ON		ON	

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

*1: ON upon external read.

*2: Port having a pull-up/pull-down resistor.

*3: AIN input does not cause a current to flow through the buffer.

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 (IDLE1/2)		In HALT mode (STOP)			
								DRVE=1		DRVE=0	
				When used as Function pin	When used as Output pin	When used as Function pin	When used as Output pin	When used as Function pin	When used as Output pin	When used as Function pin	When used as Output pin
P00-P07	D0-D7		OFF	ON upon external write (*1)	ON	OFF	ON	OFF	ON	OFF	OFF
P10-P17	D8-D15										
P40-P47	A0-DA7		ON	ON		ON		ON			
P50-P57	A8-A15										
P60-P67	A16-A23										
P70(*2)	$\overline{RD}$										
P71(*2)	$\overline{SRWR}$		OFF								
P72(*2)	$\overline{SRLLB}$										
P73(*2)	$\overline{SRLUB}$										
P76	–		OFF	–	ON(*3)	–	ON(*3)	–	ON(*3)	–	
P77	XT2	Oscillator		ON	OFF	ON	OFF	OFF	OFF	OFF	
		Port		OFF	ON(*3)	OFF	ON(*3)		ON(*3)		
P80	$\overline{CS0}$, TA1OUT		ON		ON	ON	ON	ON	ON		
P81	$\overline{CS1}$, TA3OUT										
P82	$\overline{CS2}$										
P83	$\overline{CS3}$, TA5OUT										
PD0	TB0OUT0		OFF								
PD2	TXD2										
PD3	TB1OUT0										
PD4	TB1OUT1, SCLK2										
PF0	TXD0										
PF1	–			–		–		–		–	
PF2	SCLK0, CLK			ON		ON		ON		OFF	
PF3	TXD1, HSSO										
PF4	–			–		–		–		–	
PF5	SCLK1, HSCLK			ON		ON		ON		OFF	
PN0	SCK0										
PN1(*3)	SO0, SDA0										
PN2(*3)	SCL0										
PN3	SCK1										
PN4(*3)	SO1, SDA1										
PN5(*3)	SCL1										
X2	–		ON		–		–	OFF	–		–

ON: The buffer is always turned on. When the bus is released, however, output buffers for some pins are turned off.

OFF: The buffer is always turned off.

–: Not applicable

*1: ON upon external write.

*2: Port having a pull-up resistor (programmable)

*3: Open-Drain output pin.

3.4 Interrupts

Interrupts are controlled by the CPU Interrupt mask register <IFF2:0> and by the built-in interrupt controller.

The TMP92FD23A has a total of 51 interrupts divided into the following five types:

Interrupts generated by CPU: 9 sources
- Software interrupts: 8 sources
- Illegal instruction interrupt: 1 source

Internal interrupts: 33 sources
- Internal I/O interrupts: 25 sources
- Micro DMA transfer end interrupts: 8 sources

External interrupts: 9 sources
- Interrupts on external pins (INT0 to INT7, $\overline{\text{NMI}}$)

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

Any one of six 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 sends the priority of that interrupt to the CPU. When more than one interrupt is 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 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 (EI num sets <IFF2:0> to num). 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).

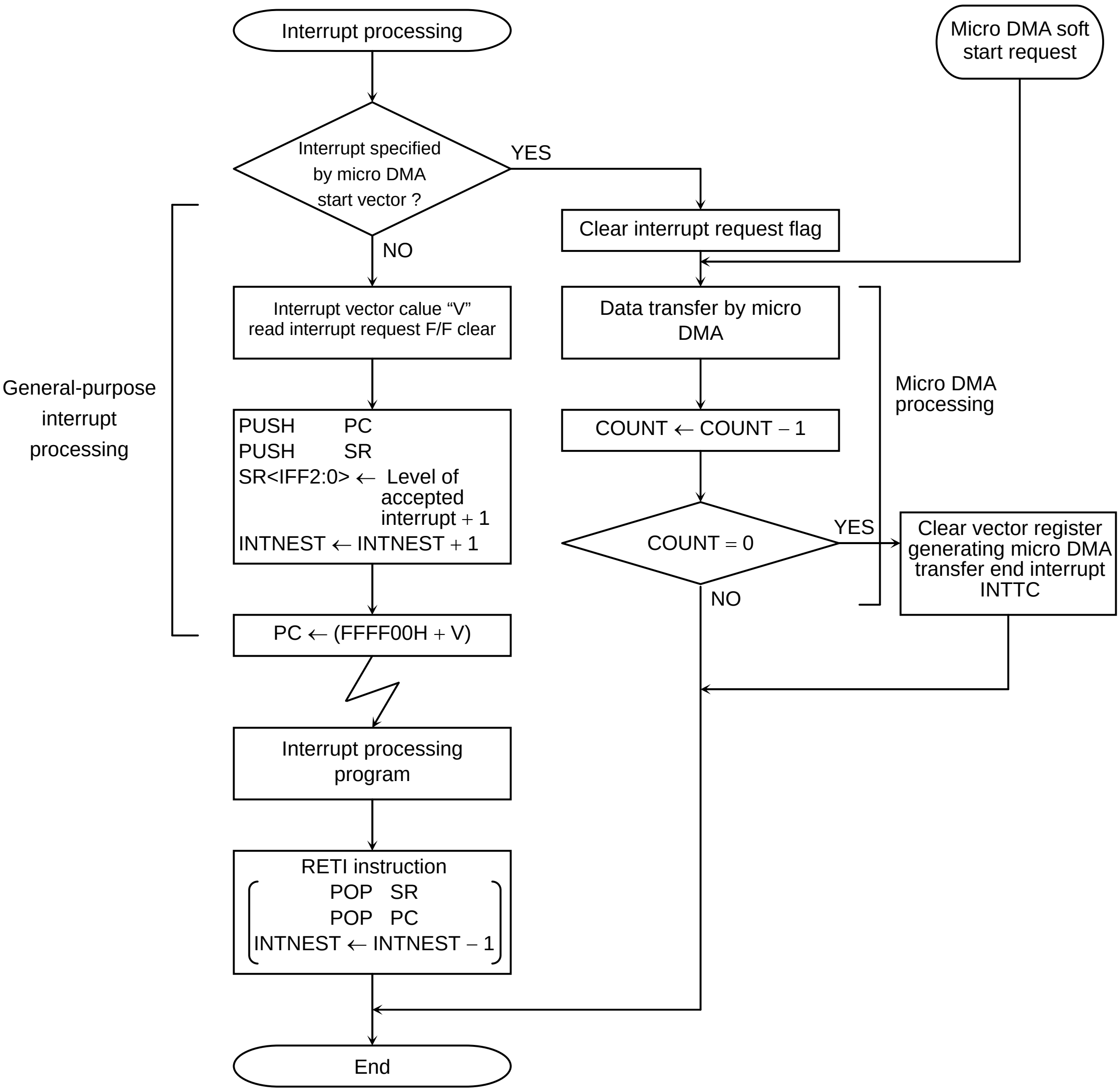
The DI instruction (sets <IFF2:0> to 7) is exactly equivalent to the EI 7 instruction. The DI instruction is used to disable all maskable interrupts (since the priority level for maskable interrupts ranges from 1 to 6). The EI instruction takes effect as soon as it is executed.

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 one-byte, two-byte or four-byte blocks; this mode allows high-speed data transfer to and from internal and external memory and internal I/O ports.

In addition, the TMP92FD23A also has a software start function in which micro DMA processing is requested in software rather than by an interrupt.

Figure 3.4.1 is a flowchart showing overall interrupt processing.



Interrupt processing
Micro DMA soft start request
Interrupt specified by micro DMA start vector ?
YES
NO
Clear interrupt request flag
Interrupt vector calue “V” read interrupt request F/F clear
Data transfer by micro DMA
General-purpose interrupt processing
Micro DMA processing
PUSH PC
PUSH SR
SR<IFF2:0> ← Level of accepted interrupt + 1
INTNEST ← INTNEST + 1
COUNT ← COUNT − 1
COUNT = 0
YES
NO
Clear vector register generating micro DMA transfer end interrupt INTTC
PC ← (FFFF00H + V)
Interrupt processing program
RETI instruction
POP SR
POP PC
INTNEST ← INTNEST − 1
End

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 has 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 + 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 interrupt mask register <IFF2:0>, the CPU will accept the interrupt. The CPU interrupt mask register <IFF2:0> is then set to the value of the priority level for the accepted interrupt plus 1.

If during interrupt processing, an interrupt is generated with a higher priority than the interrupt currently being processed, or if, during the processing of a non-maskable interrupt processing, 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 new interrupt will be sampled immediately after execution of the first instruction of its interrupt processing routine. Specifying DI as the start instruction disables nesting of maskable interrupts.

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

Table 3.4.1 shows the TMP92FD23A interrupt vectors and micro DMA start vectors. FFFF00H to FFFFFFH (256 bytes) is designated as the interrupt vector area.

Table 3.4.1 TMP92FD23A Interrupt Vectors and Micro DMA Start Vectors

Default Priority	Type	Interrupt Source and Source of Micro DMA Request	Vector Value	Address Refer to Vector	Micro DMA Start Vector
1	Non-maskable	Reset or [SWI0] instruction	0000H	FFFF00H	
2		[SWI1] instruction	0004H	FFFF04H	
3		Illegal instruction or [SWI2] instruction	0008H	FFFF08H	
4		[SWI3] instruction	000CH	FFFF0CH	
5		[SWI4] instruction	0010H	FFFF10H	
6		[SWI5] instruction	0014H	FFFF14H	
7		[SWI6] instruction	0018H	FFFF18H	
8		[SWI7] instruction	001CH	FFFF1CH	
9		$\overline{\text{NMI}}$: External interrupt input pin	0020H	FFFF20H	
10		INTWD: Watchdog Timer	0024H	FFFF24H	
–	Maskable	Micro DMA	–	–	– (Note1)
11		INT0: INT0 pin input	0028H	FFFF28H	0AH (Note 2)
12		INT1: INT1 pin input	002CH	FFFF2CH	0BH (Note 2)
13		INT2: INT2 pin input	0030H	FFFF30H	0CH (Note 2)
14		INT3: INT3 pin input	0034H	FFFF34H	0DH (Note 2)
15		INT4: INT4 pin input	0038H	FFFF38H	0EH (Note 2)
16		INT5: INT5 pin input	003CH	FFFF3CH	0FH (Note 2)
17		INT6: INT6 pin input	0040H	FFFF40H	10H (Note 2)
18		INT7: INT7 pin input	0044H	FFFF44H	11H (Note 2)
19		INTTA0: 8-bit timer 0	0048H	FFFF48H	12H
20		INTTA1: 8-bit timer 1	004CH	FFFF4CH	13H
21		INTTA2: 8-bit timer 2	0050H	FFFF50H	14H
22		INTTA3: 8-bit timer 3	0054H	FFFF54H	15H
23		INTTA4: 8-bit timer 4	0058H	FFFF58H	16H
24		INTTA5: 8-bit timer 5	005CH	FFFF5CH	17H
25		(Reserved)	0060H	FFFF60H	18H
26		(Reserved)	0064H	FFFF64H	19H
27		INTRX0: Serial receive (Channel 0)	0068H	FFFF68H	1AH (Note 2)
28		INTTX0: Serial transmission (Channel 0)	006CH	FFFF6CH	1BH
29		INTRX1: Serial receive (Channel 1)	0070H	FFFF70H	1CH (Note 2)
30		INTTX1: Serial transmission (Channel 1) INTHSC: High speed serial	0074H	FFFF74H	1DH
31		INTRX2: Serial receive (Channel 2)	0078H	FFFF78H	1EH (Note 2)
32		INTTX2: Serial transmission (Channel 2)	007CH	FFFF7CH	1FH
33		(Reserved)	0080H	FFFF80H	20H
34		(Reserved)	0084H	FFFF84H	21H
35		INTNSBE0: SBI0 I2Cbus transfer end	0088H	FFFF88H	22H
36		(Reserved)	008CH	FFFF8CH	23H
37		INTNSBE1: SBI1 I2Cbus transfer end	0090H	FFFF90H	24H
38		(Reserved)	0094H	FFFF94H	25H
39		(Reserved)	0098H	FFFF98H	26H
40		(Reserved)	009CH	FFFF9CH	27H
41		(Reserved)	00A0H	FFFFA0H	28H
42		(Reserved)	00A4H	FFFFA4H	29H
43		INTTB00: 16-bit timer 0	00A8H	FFFFA8H	2AH
44		INTTB01: 16-bit timer 0	00ACH	FFFFACH	2BH
45		INTTBO0: 16-bit timer 0 (Overflow)	00B0H	FFFFB0H	2CH
46		INTTB10: 16-bit timer 1	00B4H	FFFFB4H	2DH
47		INTTB11: 16-bit timer 1	00B8H	FFFFB8H	2EH
48		INTTBO1: 16-bit timer 1 (Overflow)	00BCH	FFFFBCH	2FH
49		INTAD: AD conversion end	00C0H	FFFFC0H	30H

Default Priority	Type	Interrupt Source and Source of Micro DMA Request	Vector Value	Address Refer to Vector	Micro DMA Start Vector
50	Maskable	INTP0: Protect 0 (Write to SFR)	00C4H	FFFFC4H	31H
51		INTRTC: Special timer for CLOCK	00C8H	FFFFC8H	32H
52		(Reserved)	00CCH	FFFFCCH	33H
53		INTTC0: Micro DMA end (Channel 0)	00D0H	FFFFD0H	34H
54		INTTC1: Micro DMA end (Channel 1)	00D4H	FFFFD4H	35H
55		INTTC2: Micro DMA end (Channel 2)	00D8H	FFFFD8H	36H
56		INTTC3: Micro DMA end (Channel 3)	00DCH	FFFFDCH	37H
57		INTTC4: Micro DMA end (Channel 4)	00E0H	FFFFE0H	38H
58		INTTC5: Micro DMA end (Channel 5)	00E4H	FFFFE4H	39H
59		INTTC6: Micro DMA end (Channel 6)	00E8H	FFFFE8H	3AH
60		INTTC7: Micro DMA end (Channel 7)	00ECH	FFFFECH	3BH
– to –		(Reserved)	00F0H : 00FCH	FFFFF0H : FFFFFCH	– to –

Note 1: When initiating micro DMA, set at edge detect mode.

Note 2: Micro DMA default priority.
Micro DMA initiation takes priority over other maskable interrupts.

Note 3: Specify the HSCSEL register when selecting INTTX1 or INTHSC that have the same interrupt factor in the default priority 30.

3.4.2 Micro DMA Processing

In addition to general purpose interrupt processing, the TMP92FD23A 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 is implemented through the CPU, when the CPU is placed in a stand-by state by a Halt instruction, the requirements of the micro DMA will be ignored (pending).

Micro DMA supports 8 channels and can be transferred continuously by specifying the micro DMA burst function as below.

(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 eight micro DMA channels allow micro DMA processing to be set for up to eight types of interrupt at once.

When micro DMA is accepted, the interrupt request flip-flop assigned to that channel is cleared. Data in one-byte, two-byte or four-byte blocks, is automatically transferred at once 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 INTTC7) 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 7 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 (i.e., interrupt requests should be disabled).

If using micro DMA and general-purpose interrupts together, first set the level of the interrupt used to start micro DMA processing lower than all the other interrupt levels. (Note) In this case, the cause of general interrupt is limited to the edge interrupt.

The priority of the micro DMA transfer end interrupt (INTTC0 to INTTC3) is defined by the interrupt level and the default priority as the same as the other maskable interrupt.

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 Figure 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 cheking 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

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. In this case, edge triggered interrupts are the only kinds of general interrupts which can be accepted.

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.

Three micro DMA transfer modes are supported: one-byte transfers, two-byte transfer and four-byte transfer. 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 memory to memory, 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 40 different interrupts – the 39 interrupts shown in the micro DMA start vectors in Table 3.4.1 and 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: this cycle is based on an external 8-bit bus, 0 waits, source/transfer destination addresses both even-numbered values.)

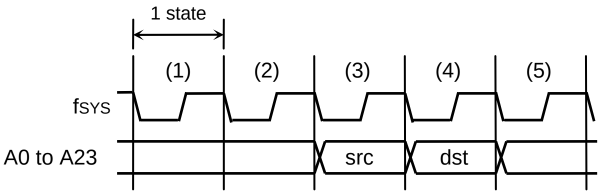


Figure 3.4.2 Timing for Micro DMA Cycle

State (1), (2):	Instruction fetch cycle (Prefetches the next instruction code) If the instruction queue buffer is FULL, this cycle becomes a dummy cycle.
State (3):	Micro DMA read cycle
State (4):	Micro DMA write cycle
State (5):	(The same as in state (1), (2))

(2) Soft start function

The TMP92FD23A 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.

Only one channel can be set for DMA request at once. (Do not write 1 to plural bits)

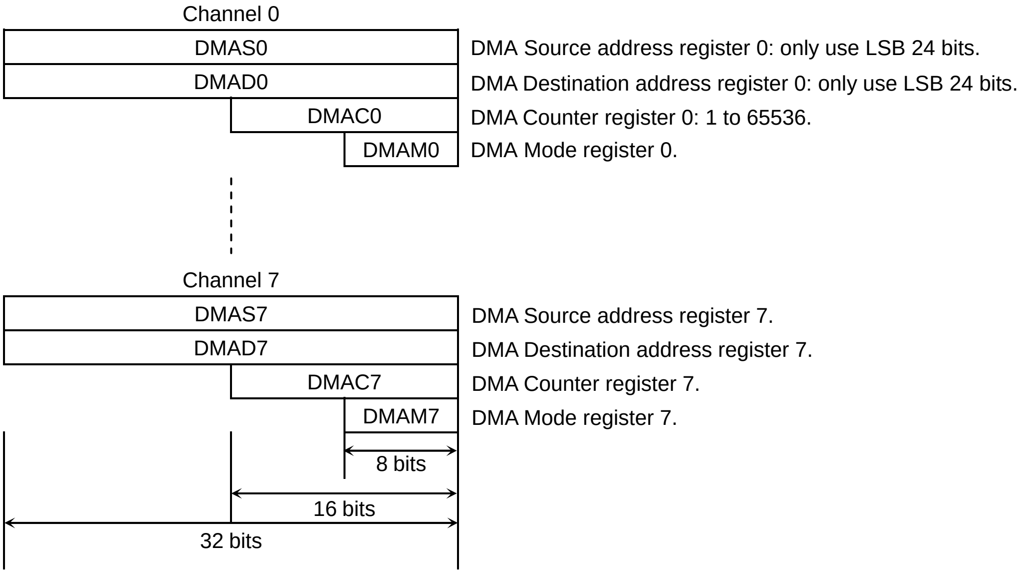
When writing again 1 to the DMAR register, check whether the bit is 0 before writing 1. If read "1", micro DMA transfer isn't started yet.

When a burst is specified by the register DMAB, data is transferred continuously from the initiation of micro DMA 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.

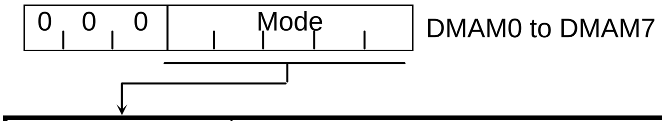
Symbol	Name	Address	7	6	5	4	3	2	1	0
DMAR	DMA Request	109H (Prohibit RMW)	DREQ7	DREQ6	DREQ5	DREQ4	DREQ3	DREQ2	DREQ1	DREQ0
			R/W							
			0	0	0	0	0	0	0	0
			1: DMA request in software							

(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.


Channel 0
DMAS0
DMAD0
DMAC0
DMAM0
DMA Source address register 0: only use LSB 24 bits.
DMA Destination address register 0: only use LSB 24 bits.
DMA Counter register 0: 1 to 65536.
DMA Mode register 0.
Channel 7
DMAS7
DMAD7
DMAC7
DMAM7
DMA Source address register 7.
DMA Destination address register 7.
DMA Counter register 7.
DMA Mode register 7.
8 bits
16 bits
32 bits

(4) Detailed description of the transfer mode register



DMAMn[4:0]	Mode Description	Execution State Number

0 0 0 z z	Destination INC mode
	(DMADn+) ← (DMASn)
	DMACn ← DMACn – 1
	if DMACn = 0 then INTTCn
		5 states

0 0 1 z z	Destination DEC mode
	(DMADn–) ← (DMASn)
	DMACn ← DMACn – 1
	if DMACn = 0 then INTTCn
		5 states

0 1 0 z z	Source INC mode
	(DMADn) ← (DMASn+)
	DMACn ← DMACn – 1
	if DMACn = 0 then INTTCn
		5 states

0 1 1 z z	Source DEC mode
	(DMADn) ← (DMASn–)
	DMACn ← DMACn – 1
	if DMACn = 0 then INTTCn
		5 states

1 0 0 z z	Source and destination INC mode (DMADn+) ← (DMASn+) DMACn ← DMACn – 1 If DMACn = 0 then INTTCn	6 states
1 0 1 z z	Source and destination DEC mode (DMADn–) ← (DMASn–) DMACn ← DMACn – 1 If DMACn = 0 then INTTCn	6 states
1 1 0 z z	Source and destination Fixed mode (DMADn) ← (DMASn) DMACn ← DMACn – 1 If DMACn = 0 then INTTCn	5 states

1 1 1 0 0	Counter mode
	DMASn ← DMASn + 1
	DMACn ← DMACn – 1
	if DMACn = 0 then INTTCn
		5 states

ZZ:	
	00 = 1-byte transfer
	01 = 2-byte transfer
	10 = 4-byte transfer
	11 = (Reserved)

Note1: The execution state number shows number of best case (1-state memory access). 1state = 50ns at f_{SYS} = 20MHz

Note2: N stands for the micro DMA channel number (0 to 7)
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 decremented memory addresses.

Note3: 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 50 interrupts 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 writing a micro DMA start vector to the INTCLR register).

An interrupt priority can be set independently for each interrupt source by writing the priority to the interrupt priority setting register (e.g., INTEPAD or INTE01). 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 interrupt (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 bit 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 eight 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 interrupts 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.

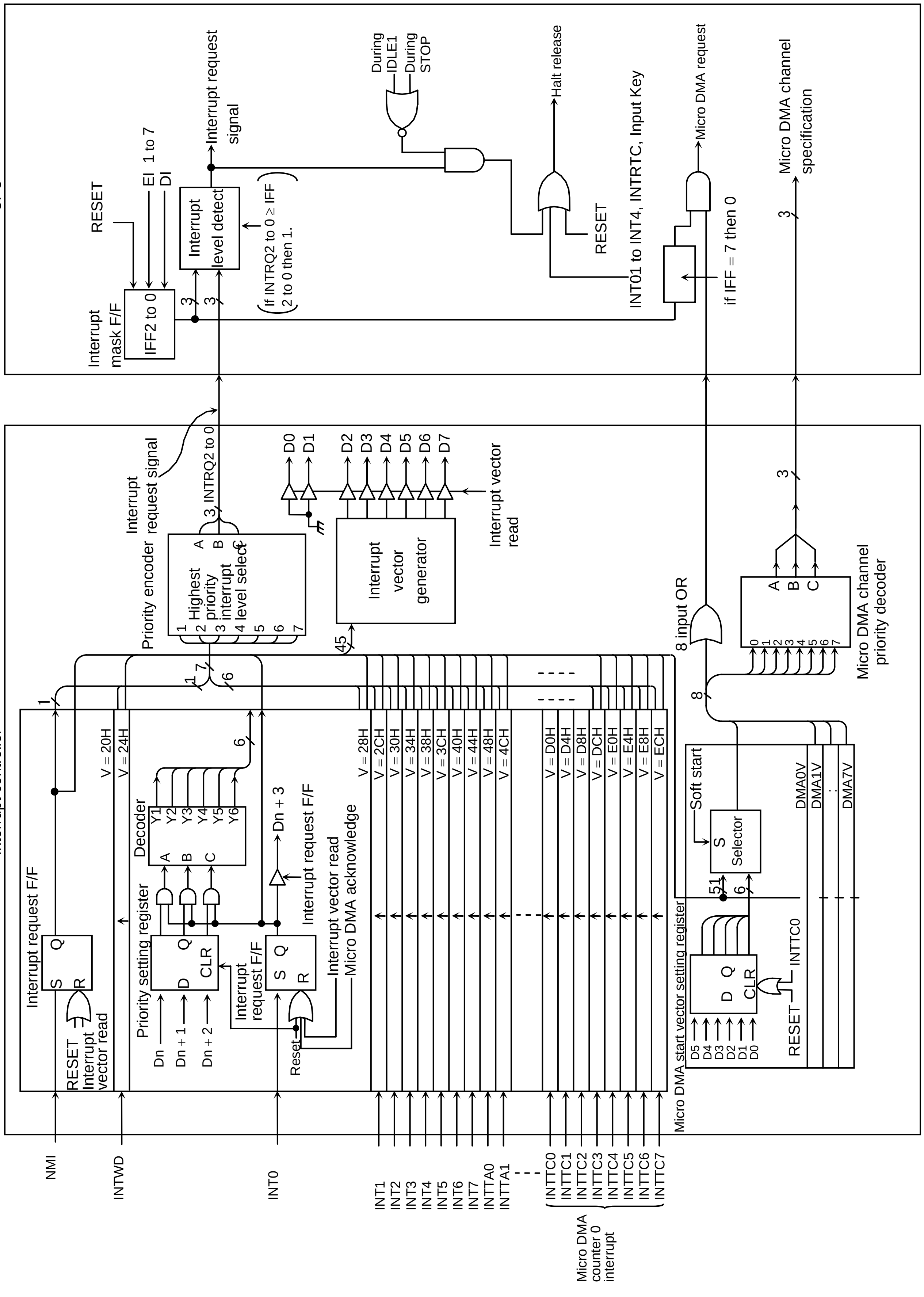


Figure 3.4.3 Block Diagram of Interrupt Controller

(1) Interrupt level setting registers

Symbol	Name	Address	7	6	5	4	3	2	1	0
INTE01	INT0 & INT1 Enable	00D0H	INT1				INT0			
			I1C	I1M2	I1M1	I1M0	I0C	I0M2	I0M1	I0M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1:INT1	Interrupt request level			1:INT0	Interrupt request level		
INTE23	INT2& INT3 Enable	00D1H	INT3				INT2			
			I3C	I3M2	I3M1	I3M0	I2C	I2M2	I2M1	I2M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1:INT3	Interrupt request level			1:INT2	Interrupt request level		
INTE45	INT4& INT5 Enable	00D2H	INT5				INT4			
			I5C	I5M2	I5M1	I5M0	I4C	I4M2	I4M1	I4M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1:INT5	Interrupt request level			1:INT4	Interrupt request level		
INTE67	INT6& INT7 Enable	00D3H	INT7				INT6			
			I7C	I7M2	I7M1	I7M0	I6C	I6M2	I6M1	I6M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1:INT7	Interrupt request level			1:INT6	Interrupt request level		
INTETA01	INTTA0 & INTTA1 Enable	00D4H	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	0
			1: INTTA1	Interrupt request level			1:INTTA0	Interrupt request level		
INTETA23	INTTA2 & INTTA3 Enable	00D5H	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	0
			1: INTTA3	Interrupt request level			1:INTTA2	Interrupt request level		
INTETA45	INTTA4 & INTTA5 Enable	00D6H	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	0
			1: INTTA5	Interrupt request level			1: INTTA4	Interrupt request level		

Interrupt request flag

IxxM2	IxxM1	IxxM0	Function (Write)
0	0	0	Disables interrupt requests
0	0	1	Sets interrupt priority level to 1
0	1	0	Sets interrupt priority level to 2
0	1	1	Sets interrupt priority level to 3
1	0	0	Sets interrupt priority level to 4
1	0	1	Sets interrupt priority level to 5
1	1	0	Sets interrupt priority level to 6
1	1	1	Disables interrupt requests

Symbol	Name	Address	7	6	5	4	3	2	1	0
INTES0	INTRX0 & INTTX0 Enable	00D8H	INTTX0				INTRX0			
			ITX0C	ITX0M2	ITX0M1	ITX0M0	IRX0C	IRX0M2	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		
INTES1HSC	INTRX1 & INTTX1/ INTHSC Enable	00D9H	INTTX1/INTHSC				INTRX1			
			ITX1C	ITX1M2	ITX1M1	ITX1M0	IRX1C	IRX1M2	IRX1M1	IRX1M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1:INTTX1	Interrupt request level			1:INTRX1	Interrupt request level		
INTES2	INTRX2 & INTTX2 Enable	00DAH	INTTX2				INTRX2			
			ITX2C	ITX2M2	ITX2M1	ITX2M0	IRX2C	IRX2M2	IRX2M1	IRX2M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1:INTTX2	Interrupt request level			1:INTRX2	Interrupt request level		
INTESB0	INTSBE0 Enable	00DCH	–				INTSBE0			
			–	–	–	–	ISBE0C	ISBE0M2	ISBE0M1	ISBE0M0
			–	–			R	R/W		
			–	–	–	–	0	0	0	0
			Always write 0				1:INTSBE0	Interrupt request level		
INTESB1	INTSBE1 Enable	00DDH	–				INTSBE1			
			–	–	–	–	ISBE1C	ISBE1M2	ISBE1M1	ISBE1M0
			–	–			R	R/W		
			–	–	–	–	0	0	0	0
			Always write 0				1:INTSBE1	Interrupt request level		
INTETB0	INTTB00 & INTTB01 Enable	00E0H	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		
INTETBO0	INTTBO0 (Overflow) Enable	00E1H	–				INTTBO0(TMRB0)			
			–	–	–	–	ITBO0C	ITBO0M2	ITBO0M1	ITBO0M0
			–	–			R	R/W		
			–	–	–	–	0	0	0	0
			Always write 0				1:INTTBO0	Interrupt request level		
INTETB1	INTTB10 & INTTB11 Enable	00E2H	INTTB11(TMRB1)				INTTB10(TMRB1)			
			ITB11C	ITB11M2	ITB11M1	ITB11M0	ITB10C	ITB10M2	ITB10M1	ITB10M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1:INTTB11	Interrupt request level			1:INTTB10	Interrupt request level		

Interrupt request flag

IxxM2	IxxM1	IxxM0	Function (Write)
0	0	0	Disables interrupt requests
0	0	1	Sets interrupt priority level to 1
0	1	0	Sets interrupt priority level to 2
0	1	1	Sets interrupt priority level to 3
1	0	0	Sets interrupt priority level to 4
1	0	1	Sets interrupt priority level to 5
1	1	0	Sets interrupt priority level to 6
1	1	1	Disables interrupt requests

Symbol	Name	Address	7	6	5	4	3	2	1	0
INTETBO1	INTTBO1 (Overflow) Enable	00E3H	–				INTTBO1(TMRB1)			
			–	–	–	–	ITBO1C	ITBO1M2	ITBO1M1	ITBO1M0
			–	–			R	R/W		
			–	–	–	–	0	0	0	0
			Always write 0				1:INTTBO1	Interrupt request level		
INTEPAD	INTP0 & INTAD Enable	00E4H	INTP0				INTAD			
			IP0C	IP0M2	IP0M1	IP0M0	IADC	IADM2	IADM1	IADM0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1:INTP0	Interrupt request level			1:INTAD	Interrupt request level		
INTERTC	INTRTC Enable	00E5H	–				INTRTC			
			–	–	–	–	IRC	IRM2	IRM1	IRM0
			–	–			R	R/W		
			–	–	–	–	0	0	0	0
			Always write 0				1:INTRTC	Interrupt request level		
INTNMWDT	NMI & INTWDT Enable	00EFH	NMI				INTWDT			
			INCNM	–	–	–	INCWD	–	–	–
			R	–			R	–		
			0	–	–	–	0	–	–	–
			1: NMI	Always write 0			1:INTWDT	Always write 0		
INTETC01	INTTC0 & INTTC1 Enable	00F0H	INTTC1(DMA1)				INTTC0(DMA0)			
			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	00F1H	INTTC3(DMA3)				INTTC2(DMA2)			
			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		
INTETC45	INTTC4 & INTTC5 Enable	00F2H	INTTC5(DMA5)				INTTC4(DMA4)			
			ITC5C	ITC5M2	ITC5M1	ITC5M0	ITC4C	ITC4M2	ITC4M1	ITC4M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1:INTTC5	Interrupt request level			1:INTTC4	Interrupt request level		
INTETC67	INTTC6 & INTTC7 Enable	00F3H	INTTC7(DMA7)				INTTC6(DMA6)			
			ITC7C	ITC7M2	ITC7M1	ITC7M0	ITC6C	ITC6M2	ITC6M1	ITC6M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1:INTTC7	Interrupt request level			1:INTTC6	Interrupt request level		

Interrupt request flag

IxxM2	IxxM1	IxxM0	Function (Write)
0	0	0	Disables interrupt requests
0	0	1	Sets interrupt priority level to 1
0	1	0	Sets interrupt priority level to 2
0	1	1	Sets interrupt priority level to 3
1	0	0	Sets interrupt priority level to 4
1	0	1	Sets interrupt priority level to 5
1	1	0	Sets interrupt priority level to 6
1	1	1	Disables interrupt requests

(2) External interrupt control

Symbol	Name	Address	7	6	5	4	3	2	1	0
IIMC	Interrupt Input mode Control	00F6H (Prohibit RMW)								NMIREE
										W
										0
										$\overline{\text{NMI}}$ 0:Falling 1:Falling and Rising
IIMC2	Interrupt Input mode Control2	00FAH (Prohibit RMW)	I7LE	I6LE	I5LE	I4LE	I3LE	I2LE	I1LE	I0LE
			W							
			0	0	0	0	0	0	0	0
			INT7 0:Edge 1:Level	INT6 0:Edge 1:Level	INT5 0:Edge 1:Level	INT4 0:Edge 1:Level	INT3 0:Edge 1:Level	INT2 0:Edge 1:Level	INT1 0:Edge 1:Level	INT0 0:Edge 1:Level
IIMC3	Interrupt Input mode Control3	00FBH (Prohibit RMW)	I7EDGE	I6EDGE	I5EDGE	I4EDGE	I3EDGE	I2EDGE	I1EDGE	I0EDGE
			W							
			0	0	0	0	0	0	0	0
			INT7 0: Rising /High 1: Falling /Low	INT6 0: Rising /High 1: Falling /Low	INT5 0: Rising /High 1: Falling /Low	INT4 0: Rising /High 1: Falling /Low	INT3 0: Rising /High 1: Falling /Low	INT2 0: Rising /High 1: Falling /Low	INT1 0: Rising /High 1: Falling /Low	INT0 0: Rising /High 1: Falling /Low
INTCLR	Interrupt Clear Control	00F8H (Prohibit RMW)	CLRV7	CLRV6	CLRV5	CLRV4	CLRV3	CLRV2	CLRV1	CLRV0
			W							
			0	0	0	0	0	0	0	0
			Clear the interrupt request flag by the writing of a micro DMA starting vector							

Note 1: Disable INT0 to INT7 requests before changing INT0 to INT7 pins mode from level sense to edge sense.

Setting example for case of INT0:

```
    DI
    LD (IIMC2)  ,XXXXXX0-B        ;   Change from “level” to “edge”.
    LD (INTCLR), 0AH              ;   Clear interrupt request flag.
    NOP                           ;   Wait EI execution.
    NOP
    NOP
    EI
```

X: Don’t care, −: No change

Note 2: See electrical characteristics in section 4 for external interrupt input pulse width.

Note 3: In a setup of a port, when choosing a 16-bit timer input and performing capture control, INT5 and INT6 operate not according to a setup of IIMC2 and IIMC3 register but according to a setup of TB1MOD<TB1CPM1:0>.

Table 3.4.2 Settings of External Interrupt Pin Function

Interrupt Pin	Shared Pin	Mode	Setting Method
INT0	P74	Rising edge	IIMC2<I0LE> = 0, IIMC3<I0EDGE > = 0
		Falling edge	IIMC2<I0LE> = 0, IIMC3<I0EDGE > = 1
		High level	IIMC2<I0LE> = 1, IIMC3<I0EDGE > = 0
		Low level	IIMC2<I0LE> = 1, IIMC3<I0EDGE > = 1
INT1	PC1	Rising edge	IIMC2<I1LE> = 0, IIMC3<I1EDGE > = 0
		Falling edge	IIMC2<I1LE> = 0, IIMC3<I1EDGE > = 1
		High level	IIMC2<I1LE> = 1, IIMC3<I1EDGE > = 0
		Low level	IIMC2<I1LE> = 1, IIMC3<I1EDGE > = 1
INT2	PC2	Rising edge	IIMC2<I2LE> = 0, IIMC3<I2EDGE > = 0
		Falling edge	IIMC2<I2LE> = 0, IIMC3<I2EDGE > = 1
		High level	IIMC2<I2LE> = 1, IIMC3<I2EDGE > = 0
		Low level	IIMC2<I2LE> = 1, IIMC3<I2EDGE > = 1
INT3	PC3	Rising edge	IIMC2<I3LE> = 0, IIMC3<I3EDGE > = 0
		Falling edge	IIMC2<I3LE> = 0, IIMC3<I3EDGE > = 1
		High level	IIMC2<I3LE> = 1, IIMC3<I3EDGE > = 0
		Low level	IIMC2<I3LE> = 1, IIMC3<I3EDGE > = 1
INT4	PD0	Rising edge	IIMC2<I4LE> = 0, IIMC3<I4EDGE > = 0
		Falling edge	IIMC2<I4LE> = 0, IIMC3<I4EDGE > = 1
		High level	IIMC2<I4LE> = 1, IIMC3<I4EDGE > = 0
		Low level	IIMC2<I4LE> = 1, IIMC3<I4EDGE > = 1
INT5	PD1	Rising edge	IIMC2<I5LE> = 0, IIMC3<I5EDGE > = 0
		Falling edge	IIMC2<I5LE> = 0, IIMC3<I5EDGE > = 1
		High level	IIMC2<I5LE> = 1, IIMC3<I5EDGE > = 0
		Low level	IIMC2<I5LE> = 1, IIMC3<I5EDGE > = 1
INT6	PD2	Rising edge	IIMC2<I6LE> = 0, IIMC3<I6EDGE > = 0
		Falling edge	IIMC2<I6LE> = 0, IIMC3<I6EDGE > = 1
		High level	IIMC2<I6LE> = 1, IIMC3<I6EDGE > = 0
		Low level	IIMC2<I6LE> = 1, IIMC3<I6EDGE > = 1
INT7	PD3	Rising edge	IIMC2<I7LE> = 0, IIMC3<I7EDGE > = 0
		Falling edge	IIMC2<I7LE> = 0, IIMC3<I7EDGE > = 1
		High level	IIMC2<I7LE> = 1, IIMC3<I7EDGE > = 0
		Low level	IIMC2<I7LE> = 1, IIMC3<I7EDGE > = 1

(3) SIO receive interrupt control

Symbol	Name	Address	7	6	5	4	3	2	1	0
SIMC	SIO interrupt mode control	F5H (Prohibit RMW)	–					IR2LE	IR1LE	IR0LE
			W					W		
			0					1	1	1
			Always write "1" (Note)					0: INTRX2 edge mode 1: INTRX2 level mode	0: INTRX1 edge mode 1: INTRX1 level mode	0: INTRX0 edge mode 1: INTRX0 level mode

Note: When you use interruption, be sure to set "1" as the bit 7 of a SIMC register.

INTRX2 level enable

0	Edge detect INTRX2
1	"H" level INTRX2

INTRX1 level enable

0	Edge detect INTRX1
1	"H" level INTRX1

INTRX0 rising edge enable

0	Edge detect INTRX0
1	"H" level INTRX0

(4) 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 Clears interrupt request flag INT0.

Symbol	Name	Address	7	6	5	4	3	2	1	0
INTCLR	Interrupt clear control	F8H (Prohibit RMW)	CLRV7	CLRV6	CLRV5	CLRV4	CLRV3	CLRV2	CLRV1	CLRV0
			W							
			0	0	0	0	0	0	0	0
			Interrupt vector							

(5) Micro DMA start vector registers

These registers assign micro DMA processing to 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 0, 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	100H			DMA0V5	DMA0V4	DMA0V3	DMA0V2	DMA0V1	DMA0V0
					R/W					
					0	0	0	0	0	0
					DMA0 start vector					
DMA1V	DMA1 start vector	101H			DMA1V5	DMA1V4	DMA1V3	DMA1V2	DMA1V1	DMA1V0
					R/W					
					0	0	0	0	0	0
					DMA1 start vector					
DMA2V	DMA2 start vector	102H			DMA2V5	DMA2V4	DMA2V3	DMA2V2	DMA2V1	DMA2V0
					R/W					
					0	0	0	0	0	0
					DMA2 start vector					
DMA3V	DMA3 start vector	103H			DMA3V5	DMA3V4	DMA3V3	DMA3V2	DMA3V1	DMA3V0
					R/W					
					0	0	0	0	0	0
					DMA3 start vector					
DMA4V	DMA4 start vector	104H			DMA4V5	DMA4V4	DMA4V3	DMA4V2	DMA4V1	DMA4V0
					R/W					
					0	0	0	0	0	0
					DMA4 start vector					
DMA5V	DMA5 start vector	105H			DMA5V5	DMA5V4	DMA5V3	DMA5V2	DMA5V1	DMA5V0
					R/W					
					0	0	0	0	0	0
					DMA5 start vector					
DMA6V	DMA6 start vector	106H			DMA6V5	DMA6V4	DMA6V3	DMA6V2	DMA6V1	DMA6V0
					R/W					
					0	0	0	0	0	0
					DMA6 start vector					
DMA7V	DMA7 start vector	107H			DMA7V5	DMA7V4	DMA7V3	DMA7V2	DMA7V1	DMA7V0
					R/W					
					0	0	0	0	0	0
					DMA7 start vector					

(6) 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
DMAB	DMA burst	108H	DBST7	DBST6	DBST5	DBST4	DBST3	DBST2	DBST1	DBST0
			R/W							
			0	0	0	0	0	0	0	0
			1: DMA burst request							

(7) Notes

The instruction execution unit and the bus interface unit in this CPU operate independently. Therefore, immediately before an interrupt is generated, if 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 this, an instruction which clears an interrupt request flag should always be placed 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 3–instructions (e.g., "NOP" × 3 times).

If it placed EI instruction without waiting NOP instruction after execution of clearing instruction, interrupt will be enabled 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 to INT7 level mode	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.
	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 to INT7 are 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.) 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. DI LD (IIMC2), 00H ; Switches from level to edge. LD (INTCLR), 0AH ; Clears interrupt request flag. NOP ; Wait EI execution NOP NOP EI
INTRX0 to INTRX2	In level mode (the register SIMC<IRxLE> set to "0"), 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 to INT7: 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. ("H" → "L")

INTRX: Instructions which read the receive buffer.

INTRX0 to INTRX2: Instructions which read the receive buffer.

3.5 Function of Ports

TMP92FD23A has I/O port pins that are shown in Table 3.5.1 in addition to functioning as general-purpose I/O ports, these pins are also used by internal CPU and I/O functions. Table 3.5.2 to Table 3.5.4 list I/O registers and their specifications.

Table 3.5.1 Port Functions

(R: PU = with programmable pull-up resistor, U = with pull-up resistor)

Port Name	Pin Name	Number of Pins	I/O	R	I/O Setting	Pin Name for Built-in Function
Port 0	P00 to P07	8	I/O	–	Bit	D0 to D7
Port 1	P10 to P17	8	I/O	–	Bit	D8 to D15
Port 4	P40 to P47	8	I/O	–	Bit	A0 to A7
Port 5	P50 to P57	8	I/O	–	Bit	A8 to A15
Port 6	P60 to P67	8	I/O	–	Bit	A16 to A23
Port 7	P70	1	I/O	PU	Bit	$\overline{\text{RD}}$
	P71	1	I/O	PU	Bit	$\overline{\text{SRWR}}$
	P72	1	I/O	PU	Bit	$\overline{\text{SRLLB}}$
	P73	1	I/O	PU	Bit	$\overline{\text{SRLUB}}$
	P74	1	Input	–	(Fixed)	INT0
	P76	1	I/O	–	Bit	XT1
	P77	1	I/O	–	Bit	XT2
Port 8	P80	1	Output	–	(Fixed)	$\overline{\text{CS0}}$, TA1OUT
	P81	1	Output	–	(Fixed)	$\overline{\text{CS1}}$, TA3OUT
	P82	1	Output	–	(Fixed)	$\overline{\text{CS2}}$
	P83	1	I/O	–	Bit	$\overline{\text{CS3}}$, $\overline{\text{WAIT}}$, TA5OUT
Port C	PC0	1	Input	–	(Fixed)	TA0IN
	PC1	1	Input	–	(Fixed)	INT1
	PC2	1	Input	–	(Fixed)	INT2
	PC3	1	Input	–	(Fixed)	INT3
Port D	PD0	1	I/O	–	Bit	INT4,TB0OUT0
	PD1	1	Input	–	(Fixed)	INT5,TB1IN0
	PD2	1	I/O	–	Bit	INT6,TB1IN1,TXD2
	PD3	1	I/O	–	Bit	INT7,TB1OUT0,RXD2
	PD4	1	I/O	–	Bit	TB1OUT1,SCLK2, $\overline{\text{CTS2}}$
Port F	PF0	1	I/O	–	Bit	TXD0
	PF1	1	I/O	–	Bit	RXD0
	PF2	1	I/O	–	Bit	SCLK0, $\overline{\text{CTS0}}$, CLK
	PF3	1	I/O	–	Bit	TXD1, HSSO
	PF4	1	I/O	–	Bit	RXD1, HSSI
	PF5	1	I/O	–	Bit	SCLK1, $\overline{\text{CTS1}}$, HSCLK
Port G	PG0 to PG7	8	Input	–	(Fixed)	AN0 to AN7,KI0 to KI7
Port L	PL0 to PL3	4	Input	–	(Fixed)	AN8 to AN11, $\overline{\text{ADTRG}}$ (PL3)
Port N	PN0	1	I/O	–	Bit	SCK0
	PN1	1	I/O	–	Bit	SO0,SDA0
	PN2	1	I/O	–	Bit	SI0,SCL0
	PN3	1	I/O	–	Bit	SCK1
	PN4	1	I/O	–	Bit	SO1,SDA1
	PN5	1	I/O	–	Bit	SI1,SCL1

Table 3.5.2 I/O Registers and Specifications (1/3)

X: Don't care

Port	Pin Name	Specification	I/O Register				
			Pn	PnCR	PnFC	PnFC2	PnODE
Port 0	P00 to P07	Input port	X	0	0	None	None
		Output port	X	1			
		D0 to D7 bus	X	X	1		
Port 1	P10 to P17	Input port	X	0	0	None	None
		Output port	X	1			
		D8 to D15 bus	X	X	1		
Port 4	P40 to P47	Input port	X	0	0	None	None
		Output port	X	1			
		A0 to A7 output	X	X	1		
Port 5	P50 to P57	Input port	X	0	0	None	None
		Output port	X	1			
		A8 to A15 output	X	X	1		
Port 6	P60 to P67	Input port	X	0	0	None	None
		Output port	X	1			
		A16 to A23 output	X	X	1		
Port 7	P70	Input port (Without pull-up)	0	0	0	None	None
		Input port (With pull-up)	1	0	0		
		Output port	X	1	0		
		$\overline{\text{RD}}$ output	X	X	1		
	P71	Input port (Without pull-up)	0	0	0		
		Input port (With pull-up)	1	0	0		
		Output port	X	1	0		
		$\overline{\text{SRWR}}$	X	X	1		
	P72	Input port (Without pull-up)	0	0	0		
		Input port (With pull-up)	1	0	0		
		Output port	X	1	0		
		$\overline{\text{SRLLB}}$	X	X	1		
	P73	Input port (Without pull-up)	0	0	0		
		Input port (With pull-up)	1	0	0		
		Output port	X	1	0		
		$\overline{\text{SRLUB}}$	X	X	1		
	P74	Input port	X	0	0		
		INT0	X	0	1		
	P76	Input port	X	0	None		
		Output port ("0" output)	0	1			
		Output port ("HZ" output)	1	1			
		XT1 input	X	X			
	P77	Input port	X	0	None		
		Output port ("0" output)	0	1			
		Output port ("HZ" output)	1	1			
		XT2 output	X	X			

Table 3.5.3 I/O Registers and Specifications (2/3)

X: Don’t care

Port	Pin Name	Specification	I/O Register				
			Pn	PnCR	PnFC	PnFC2	PnODE
Port 8	P80 to P81	Output port	X	None	0	0	None
	P80	$\overline{\text{CS0}}$ output	X		1	0	
		TA1OUT	X		X	1	
	P81	$\overline{\text{CS1}}$ output	X		1	0	
		TA3OUT	X		X	1	
	P82	Output port	X		0	None	
		$\overline{\text{CS2}}$ output	X		1		
	P83	Input port	X	0	0	0	
		Output port	X	1	0	0	
		$\overline{\text{WAIT}}$ input	X	0	1	0	
		$\overline{\text{CS3}}$ output	X	1	1	0	
		TA5OUT	X	1	0	1	
Port C	PC0	Input port	X	None	0	None	None
		TA0IN input	X		1		
	PC1	Input port	X		0		
		INT1 input	X		1		
	PC2	Input port	X		0		
		INT2 input	X		1		
	PC3	Input port	X		0		
		INT3 input	X		1		
Port D	PD0	Input port	X	0	0	None	None
		Output port	X	1	0		
		INT4 input	X	0	1		
		TB0OUT0	X	1	1		
	PD1	Input port	X	None	0	0	
		INT5Input	X		0	1	
		TB0IN0	X		1	0	
	PD2	Input port	X	0	0	0	
		Output port	X	1	0	0	
		INT6 input	X	0	0	1	
		TB0IN1 input	X	0	1	0	
		TXD2 output (3-state)	X	1	1	0	
		TXD2 (Open drain)output	X	1	1	1	
	PD3	Input port	X	0	0	0	
		Output port	X	1	0	0	
		INT7 input	X	0	0	1	
		RXD2 input	X	0	1	0	
		TB1OUT0 output	X	1	1	0	
	PD4	Input port	X	0	0	0	
		Output port	X	1	0	0	
		SCLK2 input , $\overline{\text{CTS2}}$ input	X	0	0	1	
		SCLK2 output	X	1	0	1	
		TB1OUT1	X	1	1	0	

Table 3.5.4 I/O Registers and Specifications (3/3)

X: Don’t care

Port	Pin Name	Specification	I/O Register					
			Pn	PnCR	PnFC	PnFC2	SIOCNT	PnODE
Port F	PF0	Input port	X	0	0	None	None	None
		Output port	X	1	0			
		TXD0 output (Open drain output)	X	0	1			
		TXD0 output (3-state)	X	1	1			
	PF1	Input port	X	0	0	None		
		Output port	X	1	0			
		RXD0 input	X	0	1			
	PF2	Input port	X	0	0	0		
		Output port	X	1	0	0		
		SCLK0 input , $\overline{\text{CTS0}}$ input	X	0	1	0		
		SCLK0 output	X	1	1	0		
		CLK output	X	1	0	1		
	PF3	Input port	X	0	0	None	0	
		Output port	X	1	0		0	
		TXD1 output (Open drain output)	X	0	1		0	
		TXD1 output (3-state)	X	1	1		0	
		HSSO output (3-state)	X	1	1		1	
	PF4	Input port	X	0	0	None	0	
		Output port	X	1	0		0	
		RXD1 input	X	0	1		0	
		HSSI input	X	0	1		1	
	PF5	Input port	X	0	0	None	0	
		Output port	X	1	0		0	
		SCLK1 input , $\overline{\text{CTS1}}$ input	X	0	1		0	
		SCLK1 output	X	1	1		0	
		HSCLK output	X	1	1		1	
Port G	PG0 to PG7	Input port	X	None	0	None	None	None
		AN0 to AN7 input	X		1			
		KI0 to KI7 input	X		X			
Port L	PL0 to PL3	Input port	X	None	0	None	None	None
		AN8 to AN11 input	X		1			
	PL3	$\overline{\text{ADTRG}}$	X		0			
Port N	PN0 ~ PN5	Input port	X	0	0	None	None	None
		Output port	X	1	0			
	PN0	SCK0 input	X	0	1			
		SCK0 output	X	1	1			
	PN1	SO0 output	X	0	1			
		SDA0 input/output	X	1	1			
	PN2	SI0 input	X	0	1			
		SCL0 input/output	X	1	1			
	PN3	SCK1 input	X	0	1			
		SCK1 output	X	1	1			
	PN4	SO1 output	X	0	1			
		SDA1 input/output	X	1	1			
	PN5	SI1 input	X	0	1			
		SCL1 Input/output	X	1	1			

3.5.1 Port 0 (P00 to P07)

Port 0 is an 8-bit general-purpose I/O port. Bits can be individually set as either inputs or outputs by control register P0CR and function register P0FC.

In addition to functioning as a general-purpose I/O port, port0 can also function as a data bus (D0 to D7).

Moreover, after reset release, since a device is set as an input port, when using it as a data bus (D0 to D7), it needs to set it as P0CR and P0FC.

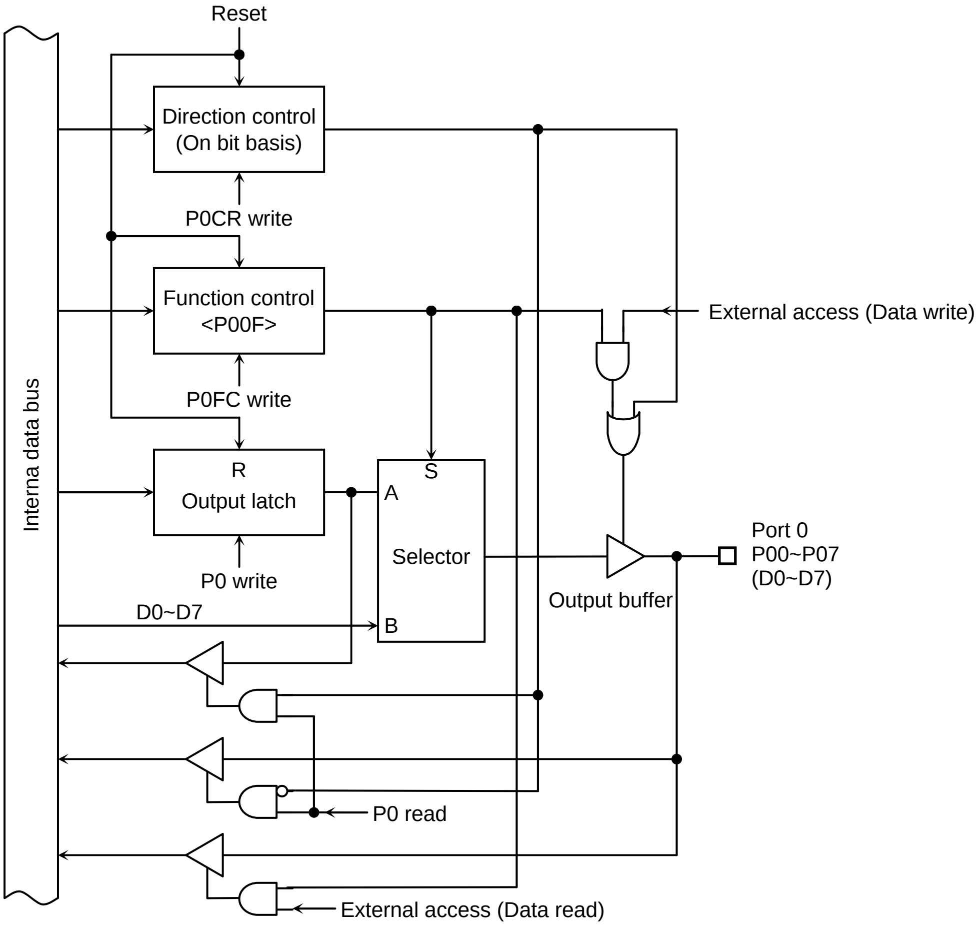


Figure 3.5.1 Port 1

Port 0 register

		7	6	5	4	3	2	1	0
P0 (0000H)	Bit symbol	P07	P06	P05	P04	P03	P02	P01	P00
	Read/Write	R/W							
	Reset State	Data from external port (Output latch register is cleared to “0”)							

Port 0 Control register

		7	6	5	4	3	2	1	0
P0CR (0002H)	Bit symbol	P07C	P06C	P05C	P04C	P03C	P02C	P01C	P00C
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Refer to following table							

Port 0 Function register

		7	6	5	4	3	2	1	0
P0FC (0003H)	Bit symbol								P00F
	Read/Write								W
	Reset State								0
	Function								Refer to following table

Port 0 function setting

P0FC<P00F> P0CR<P0xC>	0	1
0	Input port	Data bus (D0~D7)
1	Output port	

Note1: Read-modify-write is prohibited for P0CR and P0FC.

Note2: <P0xC> is bit x of P0CR register.

Figure 3.5.2 Register for Port 0

3.5.2 Port 1 (P10 to P17)

Port 1 is an 8-bit general-purpose I/O port. Bits can be individually set as either inputs or outputs by control register P1CR and function register P1FC.

In addition to functioning as a general-purpose I/O port, port1 can also function as a data bus (D8 to D15).

Moreover, after reset release, since a device is set as an input port, when using it as a data bus (D8 to D15), it needs to set it as P1CR and P1FC.

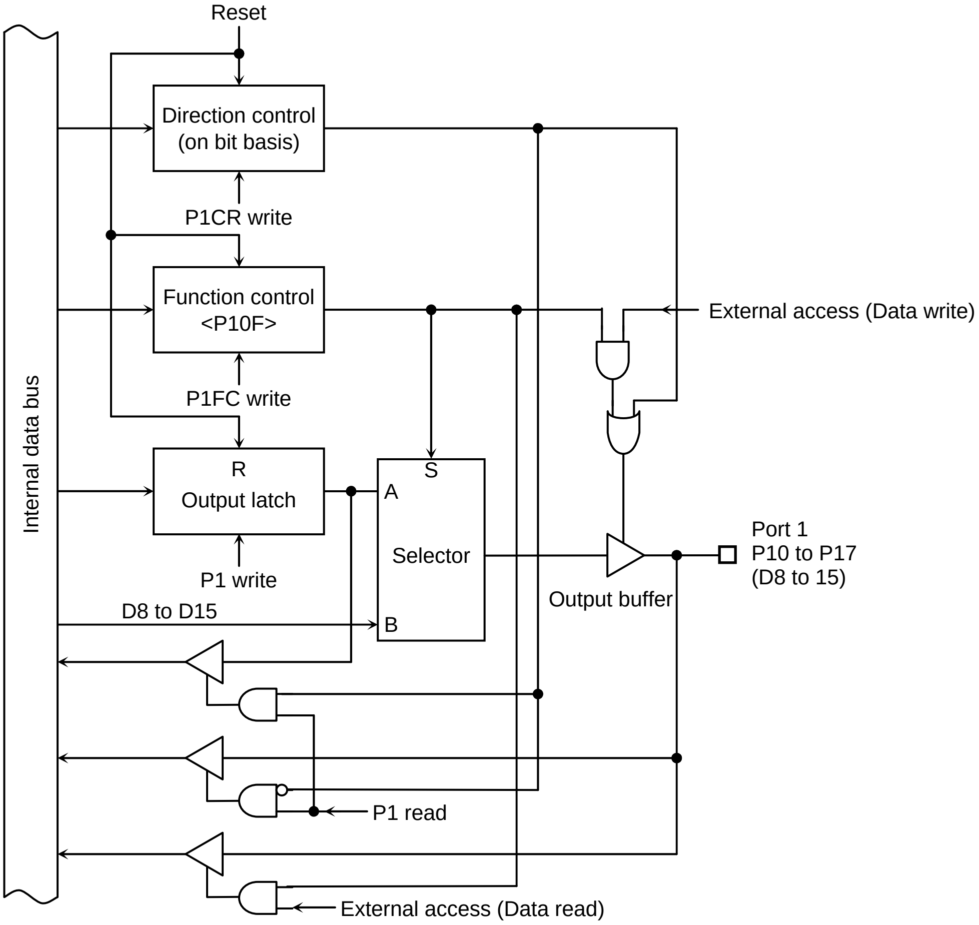


Figure 3.5.3 Port 1

Port 1 register

		7	6	5	4	3	2	1	0
P1 (0004H)	Bit symbol	P17	P16	P15	P14	P13	P12	P11	P10
	Read/Write	R/W							
	Reset State	Data from external port (Output latch register is cleared to “0”)							

Port 1 Control register

		7	6	5	4	3	2	1	0
P1CR (0006H)	Bit symbol	P17C	P16C	P15C	P14C	P13C	P12C	P11C	P10C
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Refer to following table							

Port 1 Function register

		7	6	5	4	3	2	1	0
P1FC (0007H)	Bit symbol								P10F
	Read/Write								W
	Reset State								0
	Function								Refer to following table

Port 1 function setting

P1FC<P10F> P1CR<P1xC>	0	1
0	Input port	Data bus (D8 to D15)
1	Output port	

Note1: Read-modify-write is prohibited for P1CR and P1FC.

Note2: <P1xC> is bit x of P1CR register.

Figure 3.5.4 Register for Port 1

3.5.3 Port 4 (P40 to P47)

Port4 is 8-bit general-purpose I/O ports. Bits can be individually set as either inputs or outputs by control register P4CR and function register P4FC. In addition to functioning as a general-purpose I/O port, port4 can also function as an address bus (A0 to A7).

Moreover, after reset release, since a device is set as an input port, when using it as an address bus (A0 to A7), it needs to set it as P4CR and P4FC.

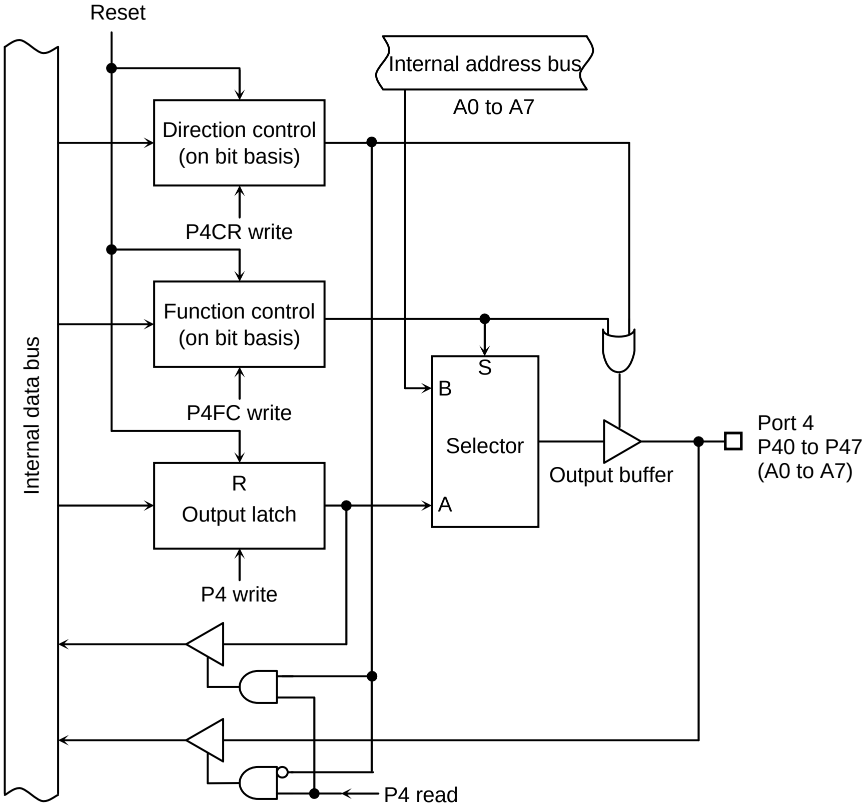


Figure 3.5.5 Port 4

Port 4 register

		7	6	5	4	3	2	1	0
P4 (0010H)	Bit symbol	P47	P46	P45	P44	P43	P42	P41	P40
	Read/Write	R/W							
	Reset State	Data from external port (Output latch register is cleared to "0")							

Port 4 Control register

		7	6	5	4	3	2	1	0
P4 (0012H)	Bit symbol	P47C	P46C	P45C	P44C	P43C	P42C	P41C	P40C
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	0: Input 1: Output							

Port 4 Function register

		7	6	5	4	3	2	1	0
P4FC (0013H)	Bit symbol	P47F	P46F	P45F	P44F	P43F	P42F	P41F	P40F
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	0: Port 1: Address bus (A0 to A7)							

Note1: Read-modify-write is prohibited for P4CR and P4FC.

Note2: When set to address bus A0 to A7, set P4FC after set P4CR.

Figure 3.5.6 Register for Port 4

3.5.4 Port 5 (P40 to P47)

Port4 is 8-bit general-purpose I/O ports. Bits can be individually set as either inputs or outputs by control register P5CR and function register P5FC. In addition to functioning as a general-purpose I/O port, port5 can also function as an address bus (A8 to A15).

Moreover, after reset release, since a device is set as an input port, when using it as an address bus (A8 to A15), it needs to set it as P5CR and P5FC.

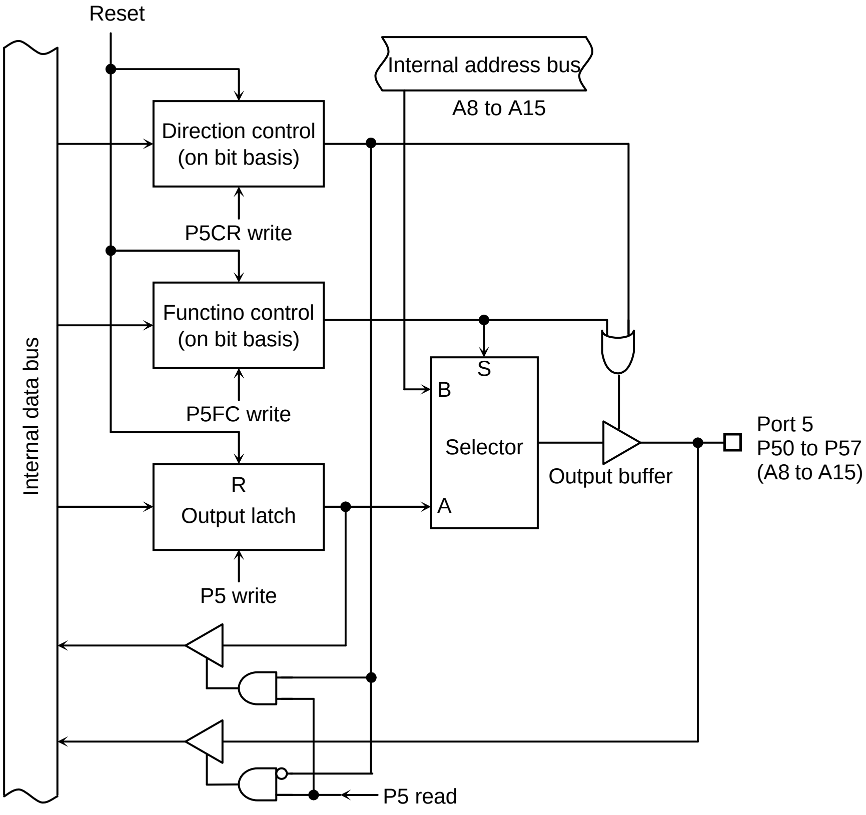


Figure 3.5.7 Port 5

Port 5 register

		7	6	5	4	3	2	1	0
P5	Bit symbol	P57	P56	P55	P54	P53	P52	P51	P50
(0014H)	Read/Write	R/W							
	Reset State	Data from external port (Output latch register is cleared to “0”)							

Port 5 Control register

		7	6	5	4	3	2	1	0
P5	Bit symbol	P57C	P56C	P55C	P54C	P53C	P52C	P51C	P50C
(0016H)	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	0: Input 1: Output							

Port 5 Function register

		7	6	5	4	3	2	1	0
P5FC	Bit symbol	P57F	P56F	P55F	P54F	P53F	P52F	P51F	P50F
(0017H)	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	0: Port 1: Address bus (A8 to A15)							

Note1: Read-modify-write is prohibited for P5CR and P5FC.

Note2: When set to address bus A8 to A15, set P5FC after set P5CR.

Figure 3.5.8 Register for Port 5

3.5.5 Port 6 (P60 to P67)

Port 6 is an 8-bit general-purpose I/O port. Bits can be individually set as either inputs or outputs by control register P6CR and function register P6FC.

In addition to functioning as a general-purpose I/O port, port 6 can also function as an address bus (A16 to A23).

Moreover, after reset release, since a device is set as an input port, when using it as a address bus (A16 to A23), it needs to set it as P6CR and P6FC.

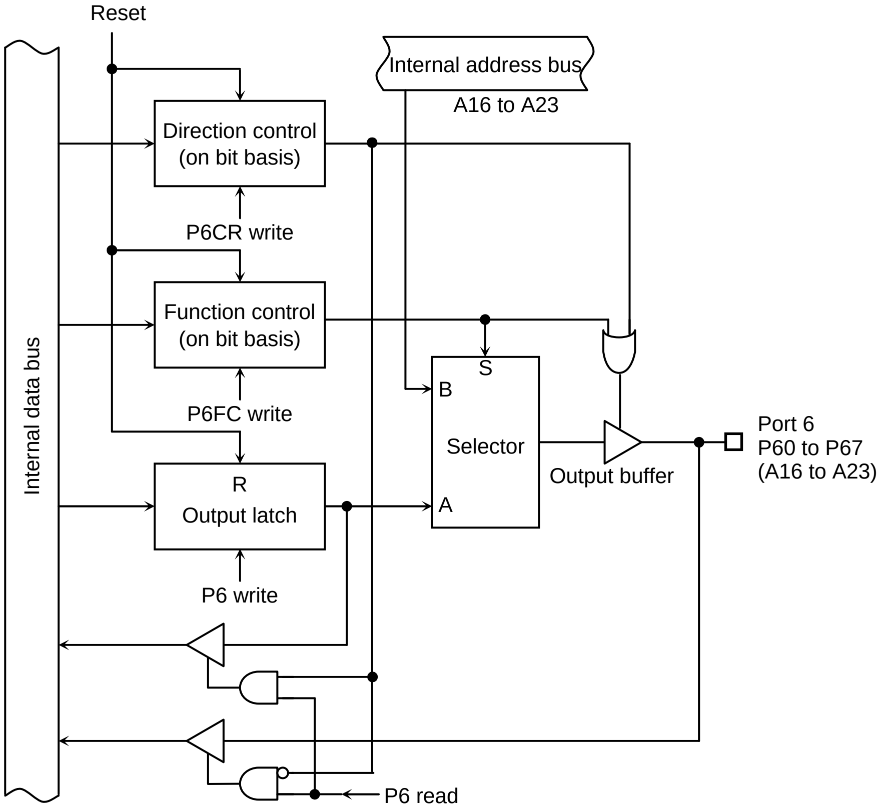


Figure 3.5.9 Port 6

Port 6 register

		7	6	5	4	3	2	1	0
P6 (0018H)	Bit symbol	P67	P66	P65	P64	P63	P62	P61	P60
	Read/Write	R/W							
	Reset State	Data from external port (Output latch register is cleared to “0”)							

Port 6 Control register

		7	6	5	4	3	2	1	0
P6CR (001AH)	Bit symbol	P67C	P66C	P65C	P64C	P63C	P62C	P61C	P60C
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	0: Input 1: Output							

Port 6 Function register

		7	6	5	4	3	2	1	0
P6FC (001BH)	Bit symbol	P67F	P66F	P65F	P64F	P63F	P62F	P61F	P60F
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	0: Port 1: Address bus (A16 to A23)							

Note1: Read-modify-write is prohibited for P6CR and P6FC.

Note2: When set to address bus A16 to A23, set P6FC after set P6CR.

Figure 3.5.10 Register for Port 6

3.5.6 Port 7 (P70 to P74, P76, P77)

As for a port7, P70 to P73, and P76 and P77 are general-purpose I/O ports, and P74 is a port only for inputs.

P76 and P77 become an open drain output, when it is set as an output port. Moreover, P70 to P73 are ports with pull-up resistance. Bits can be individually set as either inputs or outputs by control register P7CR and function register P7FC.

In addition to functioning as a general-purpose I/O port, port7 can also function as a CPU's control. P70 to P73 has the function of RD strobe signal output as an object for external memory connection, and the output for SRAM control ($\overline{\text{SRWR}}$, $\overline{\text{SRLLB}}$ and $\overline{\text{SRLUB}}$). P74 has the function of an external interrupt input (INT0). P76 and P77 have the function of a low-frequency resonator connection (XT1, XT2). These setups become effective by setting "1" as the applicable bit of P7CR and a P7FC register. The edge of the external interruption INT0 and level selection are set up in IIMC2 and IIMC3 register in an interruption controller. P70 to P74 become input mode by the reset action, and P76 and P77 become output mode (high impedance output).

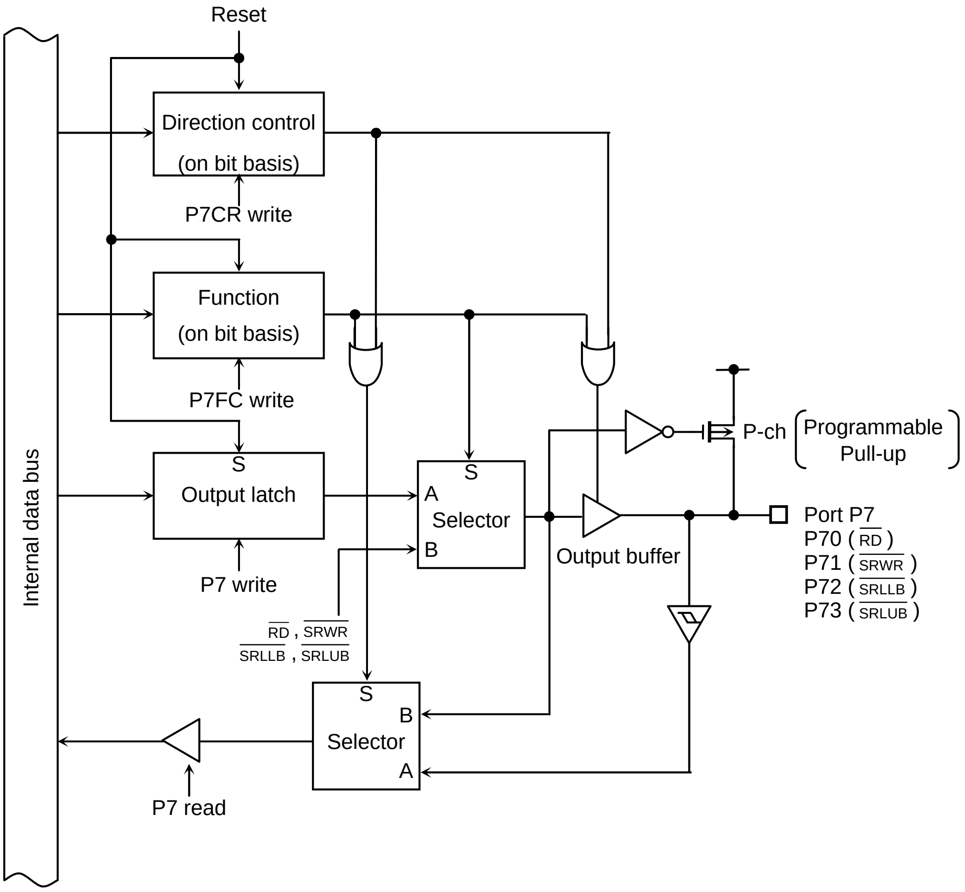

Reset
Direction control
(on bit basis)
P7CR write
Function
(on bit basis)
P7FC write
S
Output latch
P7 write
S
A
Selector
B
$\overline{\text{RD}}$, $\overline{\text{SRWR}}$
$\overline{\text{SRLLB}}$, $\overline{\text{SRLUB}}$
P-ch
Programmable Pull-up
Output buffer
Port P7
P70 ($\overline{\text{RD}}$)
P71 ($\overline{\text{SRWR}}$)
P72 ($\overline{\text{SRLLB}}$)
P73 ($\overline{\text{SRLUB}}$)
S
B
Selector
A
P7 read
Internal data bus

Figure 3.5.11 Port 7 (P70 to P73)

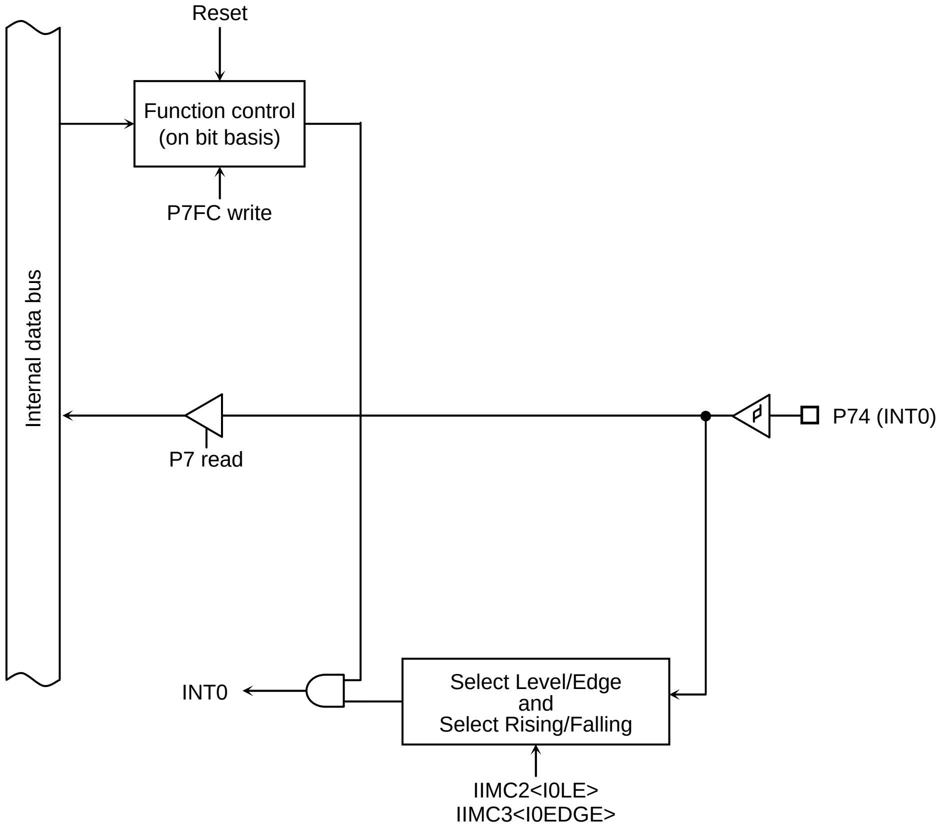


Figure 3.5.12 Port 7(P74)

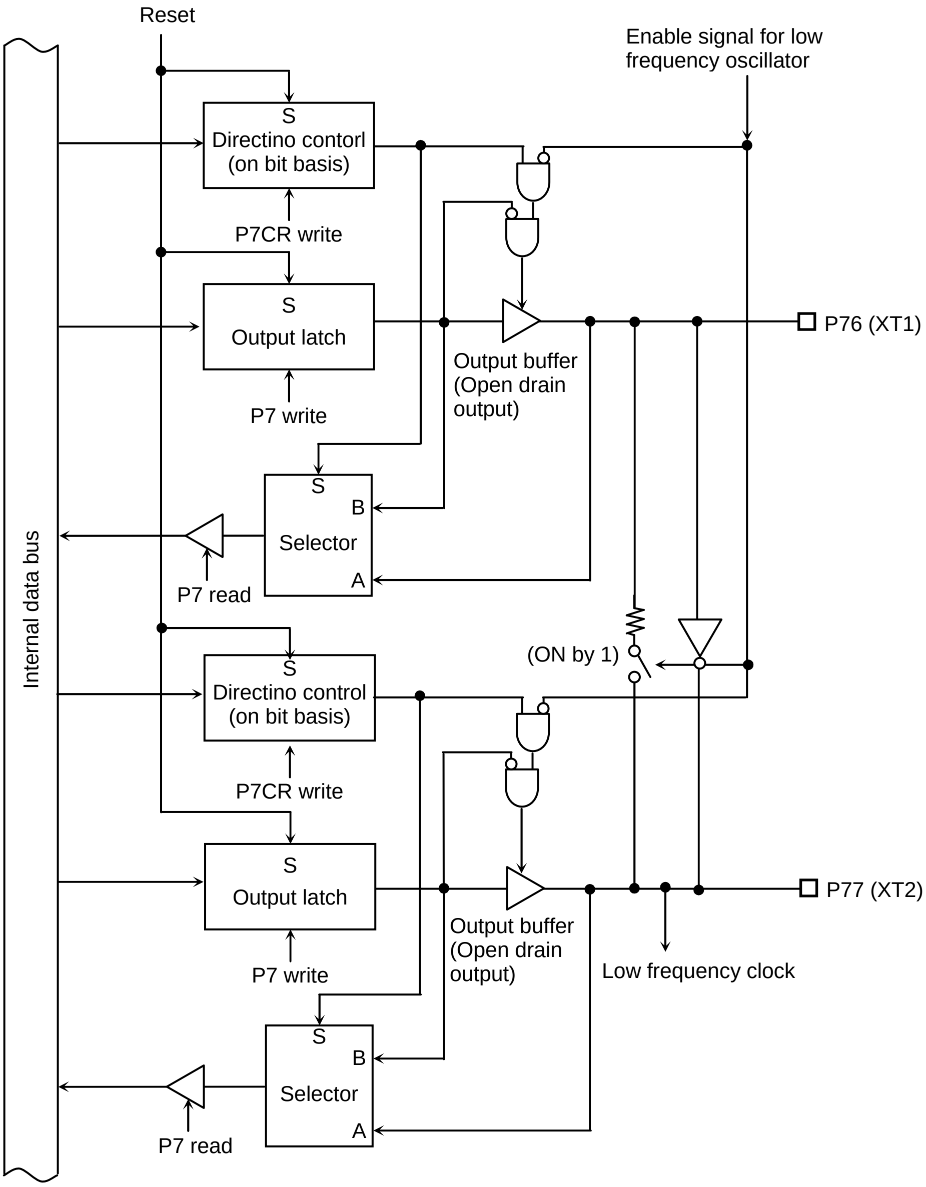

Reset
Enable signal for low frequency oscillator
S
Directino contorl (on bit basis)
P7CR write
S
Output latch
P7 write
Output buffer (Open drain output)
P76 (XT1)
S
B
Selector
A
P7 read
Internal data bus
(ON by 1)
S
Directino control (on bit basis)
P7CR write
S
Output latch
P7 write
Output buffer (Open drain output)
P77 (XT2)
Low frequency clock
S
B
Selector
A
P7 read

Figure 3.5.13 Port7 (P76, P77)

Port 7 register

		7	6	5	4	3	2	1	0
P7 (001CH)	Bit symbol	P77	P76		P74	P73	P72	P71	P70
	Read/Write	R/W			R	R/W			
	Reset State	Data from external port (Output latch register is set to "1")			Data from external port	Data from external port (Output latch register is set to "1")			
	Function	－	－		－	0(Output latch register): Pull-up resistor OFF 1(Output latch register): Pull-up resistor ON			

Port 7 Control register

		7	6	5	4	3	2	1	0
P7CR (001EH)	Bit symbol	P77C	P76C			P73C	P72C	P71C	P70C
	Read/Write	W				W			
	Reset State	1	1			0	0	0	0
	Function	0: Input 1: Output				0: Input 1: Output			

Port 7 Function register

		7	6	5	4	3	2	1	0
P7FC (001FH)	Bit symbol				P74F	P73F	P72F	P71F	P70F
	Read/Write				W				
	Reset State				0	0	0	0	0
	Function				0: Port 1: INT0	0: Port 1: $\overline{\text{SRLUB}}$	0: Port 1: $\overline{\text{SRLLB}}$	0: Port 1: $\overline{\text{SRWR}}$	0: Port 1: $\overline{\text{RD}}$

Note 1: When port P70 to P73 is used in the input mode, P7 register controls the built-in pull-up resistor. Read-modify-write is prohibited in the input mode or the I/O mode. Setting the built-in pull-up resistor may be depended on the states of the input pin.

Note 2: Read-modify-write prohibited for register P7CR, P7FC.

Note 3: On using low-frequency resonator to P76, P77, it is necessary to set the following procedures to reduce the consumption power supply.

・connecting to a resonator

P7CR <P76C,P77C> = "11", P7 <P76,P77> = "00"

・connecting an oscillator

P7CR <P76C,P77C> = "11", P7 <P76,P77> = "10"

Figure 3.5.14 Register for Port 7

3.5.7 Port 8 (P80 to P83)

Port 80 to 82 are 3-bit output ports, and Port 83 is 1-bit I/O port.

In addition to an output and an I/O port function, as for P80 and P81, a standard chip select signal output ($\overline{\text{CS0}}$, $\overline{\text{CS1}}$) and a 8-bit timer output (TA1OUT, TA3OUT), and P82 have a standard chip select signal output ($\overline{\text{CS2}}$), and P83 has the function of a standard chip select signal output ($\overline{\text{CS3}}$), a 8-bit timer output (TA5OUT), and a wait input ($\overline{\text{WAIT}}$).

These functions operate by setting the bit concerned of P8CR, P8FC, and P8FC2 register as “1”. All the bits of P8FC and P8FC2 are cleared to "0" by the reset action, and P80 to P83 becomes an output port. Moreover, the output latch of P82 is cleared to “0” and the output latch of P80 to P81 and P83 is set to “1”.

(1) P80 ($\overline{\text{CS0}}$, TA1OUT), P81 ($\overline{\text{CS1}}$, TA3OUT)

In addition to an output port function, ports P80 and P81 function as a standard chip select signal output ($\overline{\text{CS0}}$, $\overline{\text{CS1}}$) and a 8-bit timer output (TA1OUT, TA3OUT).

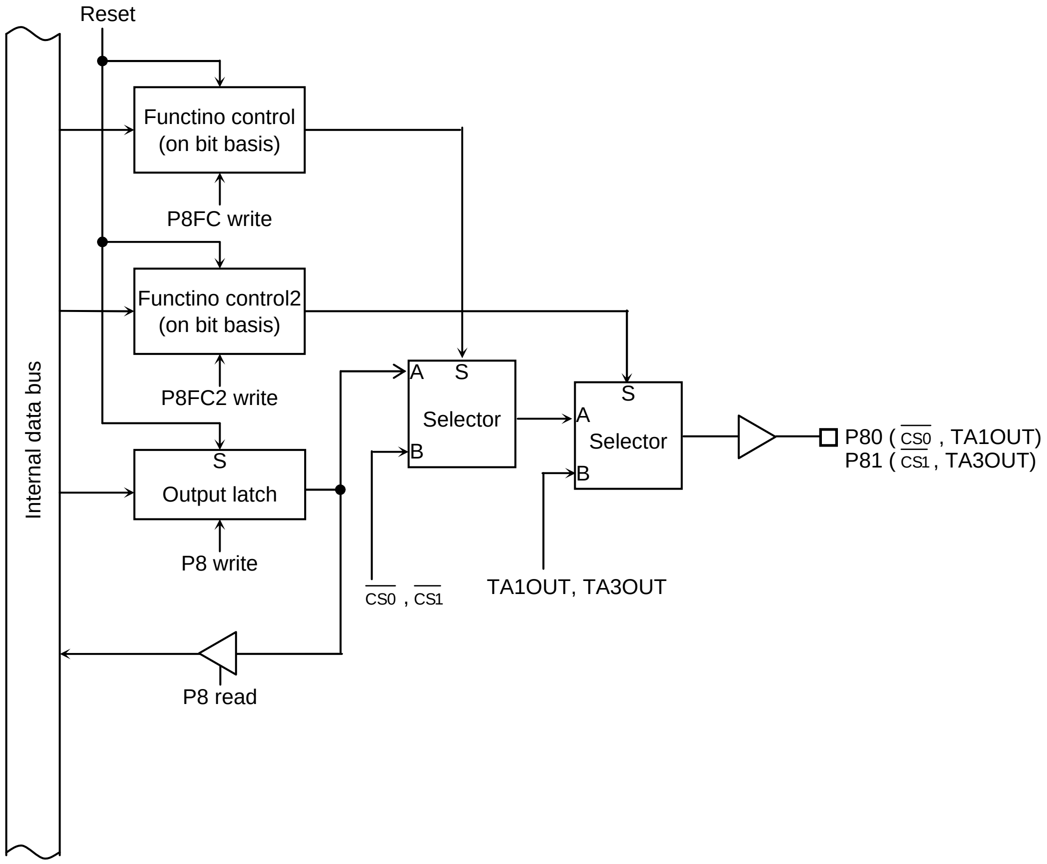


Figure 3.5.15 Port 8 (P80, P81)

(2) P82 ($\overline{\text{CS2}}$)

In addition to an output port function, a port P82 functions as a standard chip select signal output ($\overline{\text{CS2}}$).

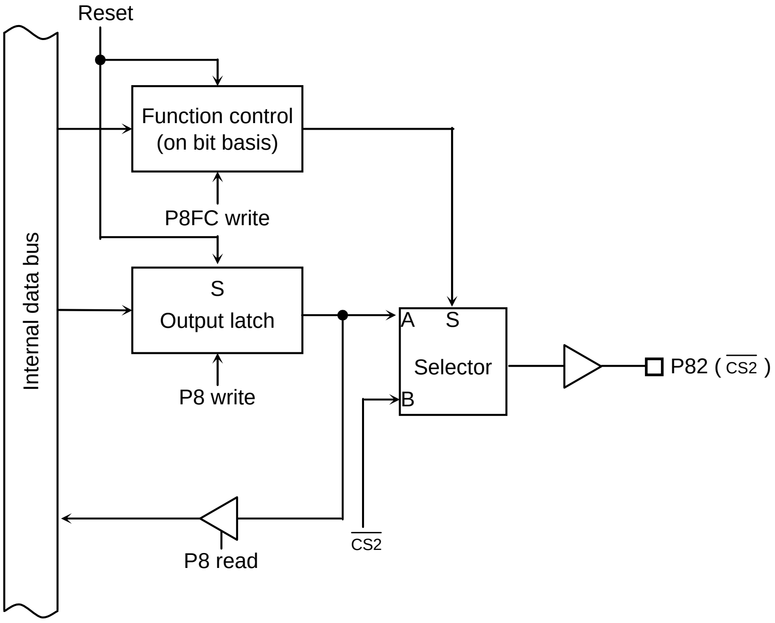


Figure 3.5.16 Port 8 (P82)

(3) P83($\overline{\text{CS3}}$, $\overline{\text{WAIT}}$, TA5OUT)

In addition to an I/O port function, a port P83 functions as a standard chip select signal output ($\overline{\text{CS3}}$) and an 8-bit timer output (TA5OUT), and a wait input ($\overline{\text{WAIT}}$).

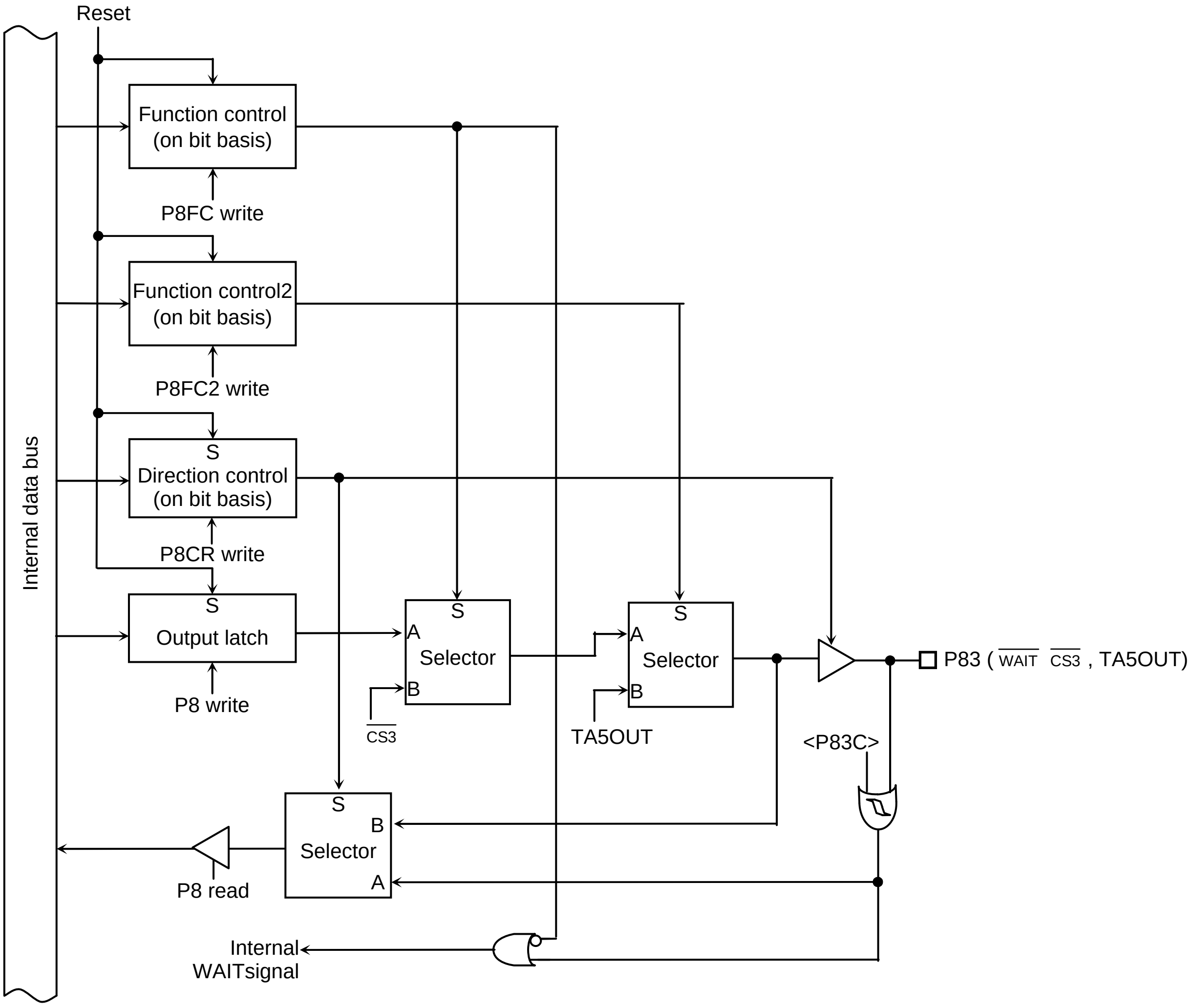


Figure 3.5.17 Port 8 (P83)

Port 8 Register

		7	6	5	4	3	2	1	0
P8 (0020H)	Bit symbol					P83	P82	P81	P80
	Read/Write					R/W			
	Reset State					Data from external port (Note1)	0	1	1

Port 8 Control Register

		7	6	5	4	3	2	1	0
P8CR (0022H)	Bit symbol					P83C			
	Read/Write					W			
	Reset State					1			
						0: Input 1: Output			

Port 8 Function Register

		7	6	5	4	3	2	1	0
P8FC (0023H)	Bit symbol					P83F	P82F	P81F	P80F
	Read/Write					W			
	Reset State					0	0	0	0
	Function					0: Port 1: $\overline{\text{WAIT}}$, $\overline{\text{CS3}}$	0: Port 1: $\overline{\text{CS2}}$	0: Port 1: $\overline{\text{CS1}}$	0: Port 1: $\overline{\text{CS0}}$

Port 8 Function Register 2

		7	6	5	4	3	2	1	0
P8FC2 (0021H)	Bit symbol					P83F2		P81F2	P80F2
	Read/Write					W		W	
	Reset State					0		0	0
	Function					0: <P83F> 1: TA5OUT		0: <P81F> 1: TA3OUT	0: <P80F> 1: TA1OUT

$\overline{\text{WAIT}}$, $\overline{\text{CS3}}$,TA5OUT setting

<P83F:P83F2>		<P83C> 0	1
0	0	Input port	Output port
0	1	Reserved	TA5OUT
1	0	$\overline{\text{WAIT}}$	$\overline{\text{CS3}}$
1	1	Reserved	Reserved

Note 1: Output latch register is set to “1”.

Note 2: Read-modify-write instructions are prohibited for P8CR, P8FC and P8FC2.

Note 3: When using P83 as a $\overline{\text{WAIT}}$ input, while setting it as P8CR <P83C> = “0”, P8FC<P83F> = “1”, it is necessary to set memory control register BxCSL <BxWW2:0> or <BxWR2:0> as “011”.

Note 4: When setting a standard chip select signal ($\overline{\text{CS0}}$ to $\overline{\text{CS3}}$) as an output, P8CR is set up after setting up P8FC.

Figure 3.5.18 Register for Port 8

3.5.8 Port C (PC0 to PC3)

Port C is a 4-bit input port.

In addition to the input port function, Port C has the input function (TA0IN) of a 8-bit timer, and an external interrupt input function (INT1 to INT3). These functions operate by setting the bit concerned of PCFC register as “1”. Edge selection of external interrupt is set up in IIMC2 and IIMC3 register in an interrupt controller. All the bits of PCFC are cleared to "0" by the reset action, and all bits serve as an input port.

(1) PC0 (TA0IN)

In addition to an I/O port function, a port PC0 has a function as a TA0IN input of the timer channel 0.

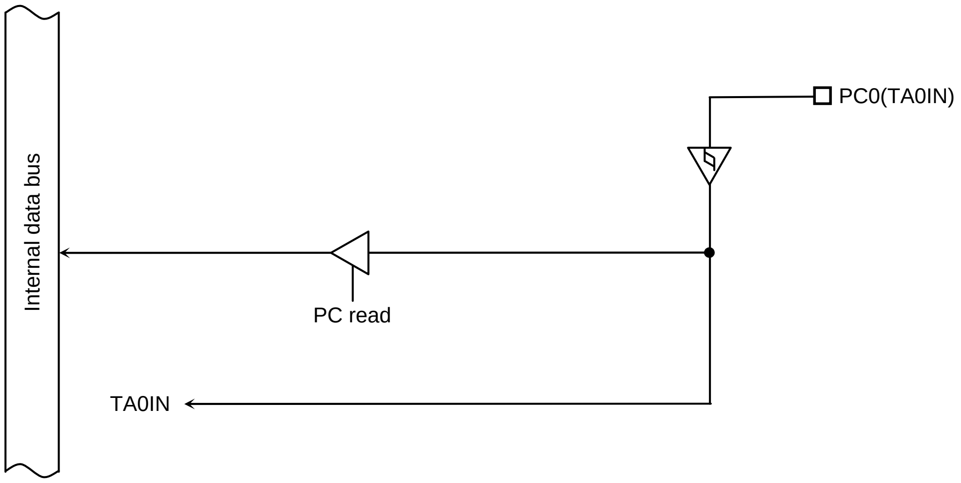


Figure 3.5.19 Port C (PC0)

(2) PC1 (INT1), PC2 (INT2), PC3 (INT3)

In addition to an Input port function, port PC1 to PC3 has a function as an external interrupt input (INT1 to INT3).

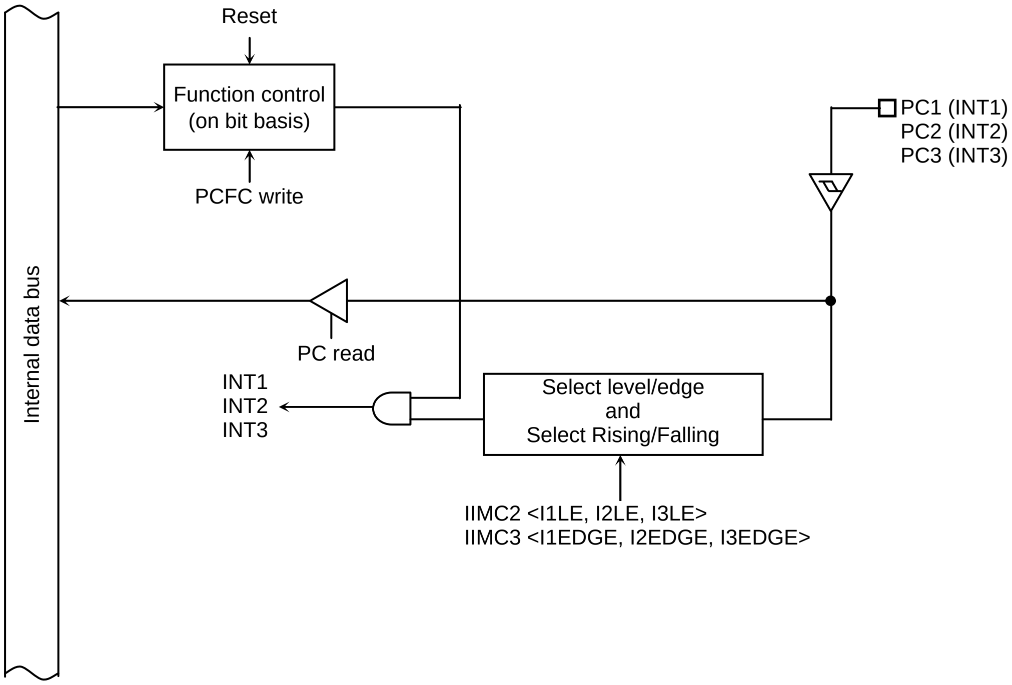


Figure 3.5.20 Port C (PC1, PC2 and PC3)

Port C Register

		7	6	5	4	3	2	1	0
PC	Bit symbol					PC3	PC2	PC1	PC0
(0030H)	Read/Write					R			
	Reset State					Data from external port			

Port C Function Register

		7	6	5	4	3	2	1	0
PCFC	Bit symbol					PC3F	PC2F	PC1F	PC0F
(0033H)	Read/Write					W			
	Reset State					0	0	0	0
	Function					0: Port 1: INT3	0: Port 1: INT2	0: Port 1: INT1	0: Port 1: TA0IN

Note1: Read-modify-write instructions are prohibited for PCFC.

Note2: PC0 is not based on a functional setup of a port, but is inputted into TA0IN of a 8-bit timer (TMRA0).

Figure 3.5.21 Register for Port C

3.5.9 Port D (PD0 to PD4)

Port D is 4-bit I/O port (PD0, PD2 to PD4) and 1-bit input port (PD1).

There are I/O of the serial channel 2, I/O of a 16-bit timer (TMRB0, TMRB1), and an external interrupt input (INT4 to INT7) function in addition to an I/O port function. These functions operate by setting the bit concerned of PDCR, PDFC and PDFC2 register as “1”. Edge selection of external interrupt is set up in IIMC2 and IIMC3 register in an interrupt controller. All the bits of PDCR, PDFC and PDFC2 are cleared to “0” by the reset action, and all bits serve as an input port.

(1) PD0 (INT4, TB0OUT0)

In addition to an I/O port function, a port PD0 has a function as a 16-bit timer output (TB0OUT0) and an external interrupt input (INT4).

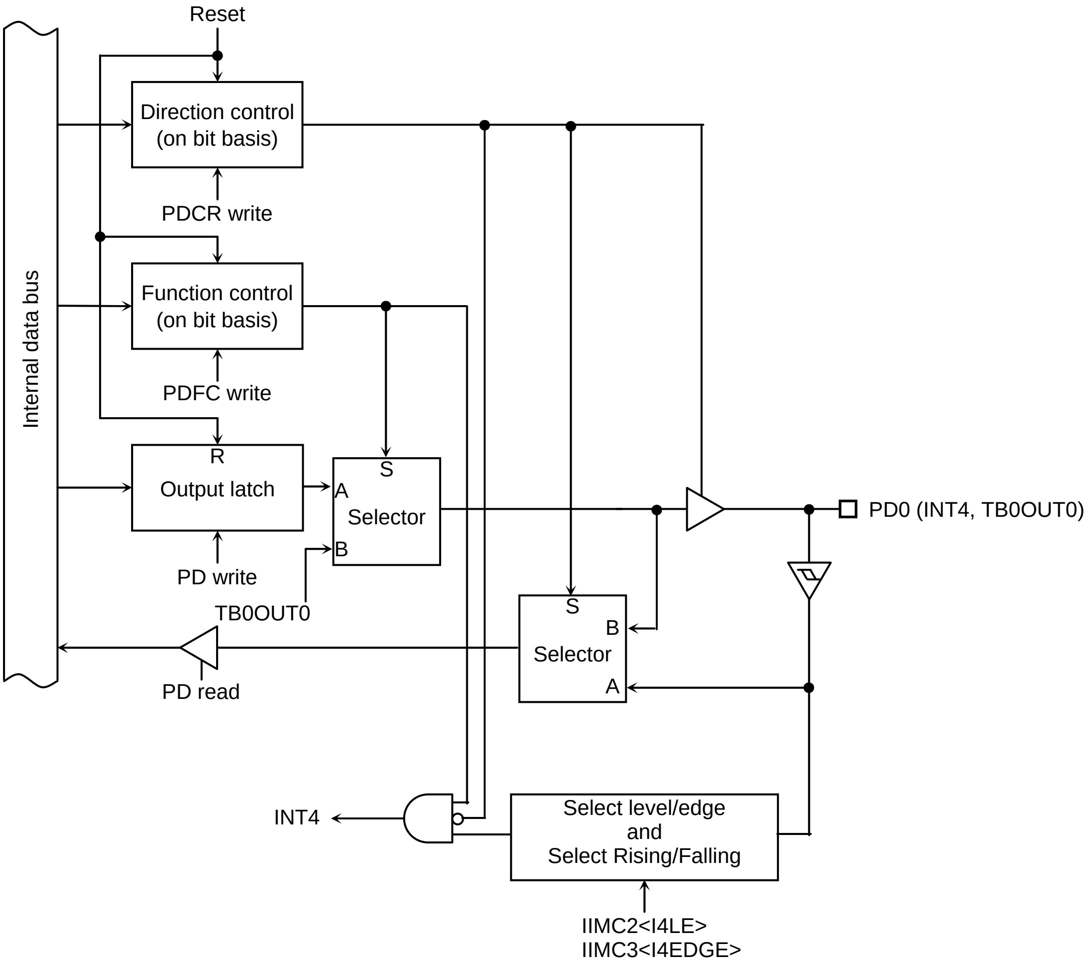


Figure 3.5.22 Register for Port D (PD0)

(2) PD1 (INT5,TB1IN0)

In addition to the input port function, the port PD1 has a function as a 16-bit timer input (TB1IN0) and an external interrupt input (INT5). In a port setup, when choosing a 16-bit timer input and performing capture control, INT5 disregards a setup of IIMC2 and IIMC3 register, and operates according to a setup of TB1MOD <TB1CPM1:0>.

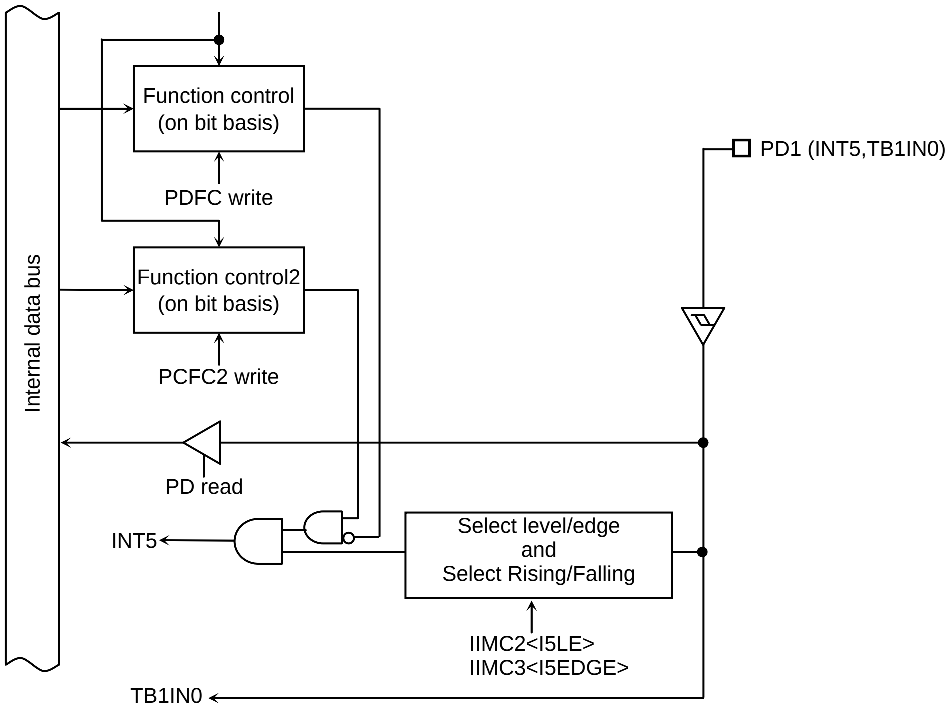


Figure 3.5.23 Port D (PD1)

(3) PD2 (INT6, TB1IN1, TXD2)

In addition to the I/O port, PD2 has a function as a 16-bit timer input (TB1IN1), an external interrupt input (INT6), and a TXD output (TXD2) of the serial channel 2. When using this port as TXD output (TXD2), it can be set as open drain.

In a port setup, when choosing a 16-bit timer input and performing capture control, INT6 disregards a setup of IIMC2 and IIMC3 register, and operates according to a setup of TB1MOD <TB1CPM1:0>.

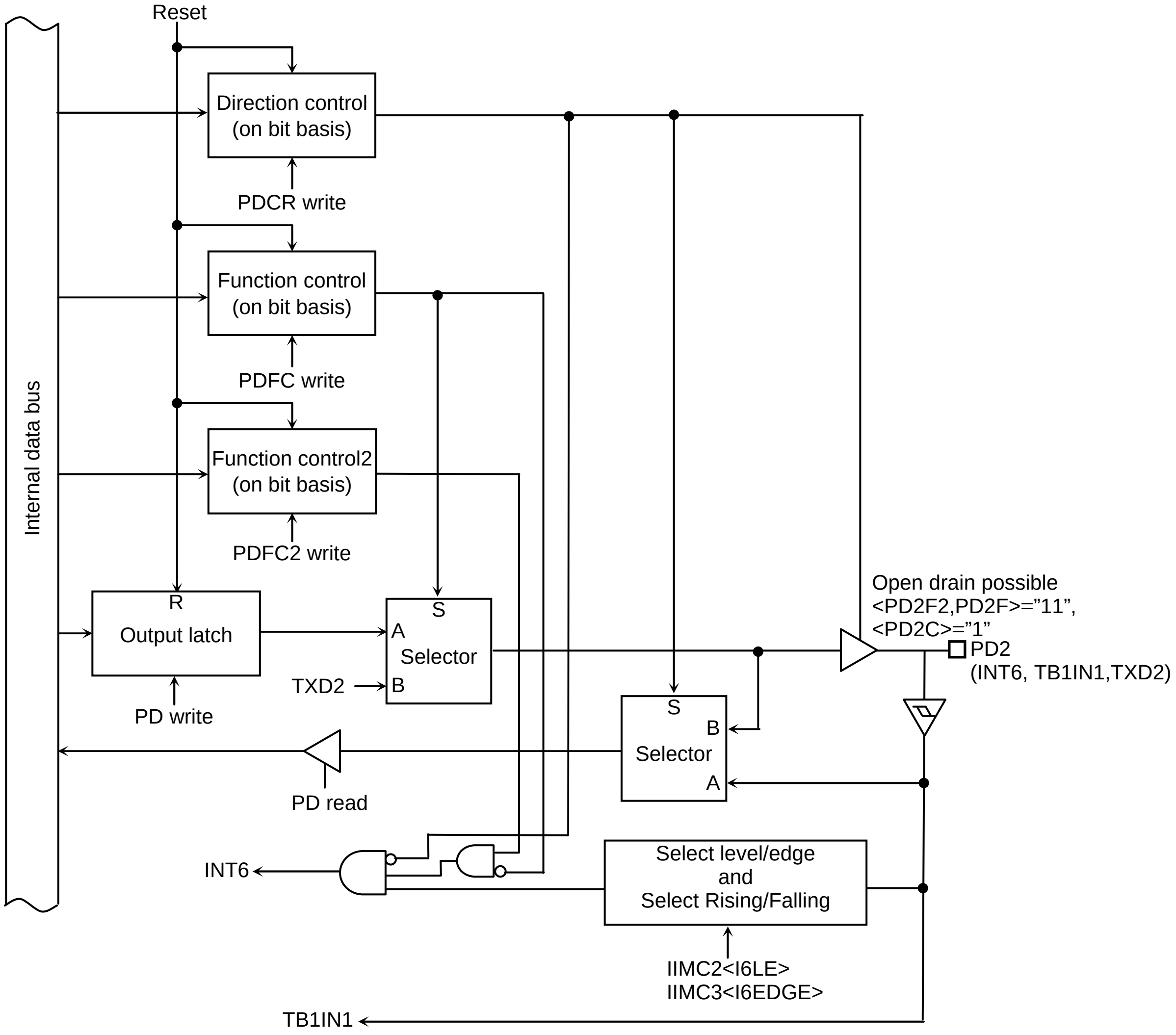


Figure 3.5.24 Port D (PD2)

(4) PD3 (INT7, TB1OUT0, RXD2)

In addition to the I/O port function, the portD3 has a function as a 16-bit timer output (TB1OUT0), an external interrupt input (INT7), and a RXD input (RXD2) of the serial channel 2.

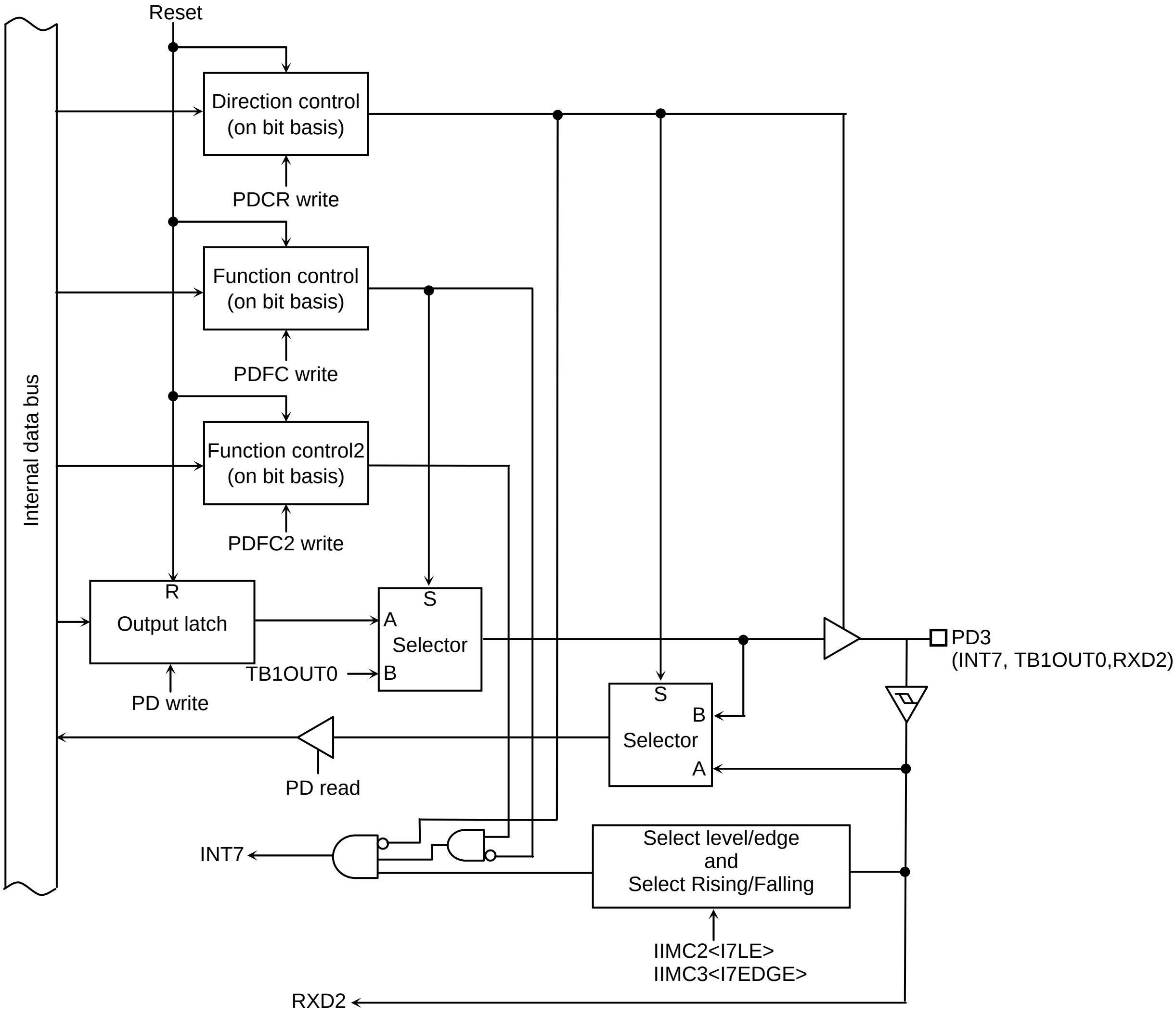


Figure 3.5.25 Port D (PD3)

(5) PD4 (TB1OUT1, SCLK2, $\overline{\text{CTS2}}$)

In addition to the I/O port function, PD4 has a function as a 16-bit timer output (TB1OUT1), SCLK I/O (SCLK2) of the serial channel 2, or a CTS input ($\overline{\text{CTS2}}$).

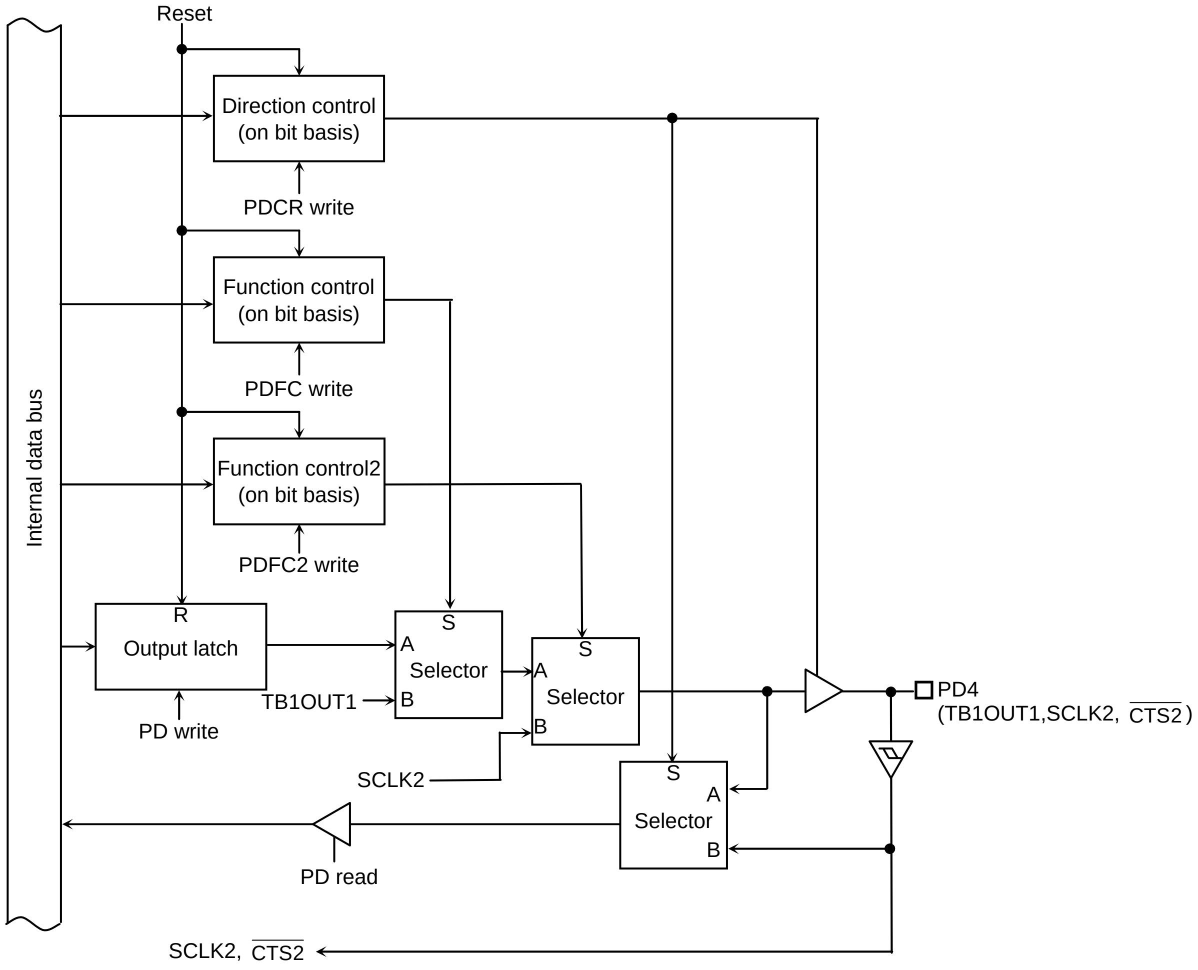

Reset
Internal data bus
Direction control (on bit basis)
PDCR write
Function control (on bit basis)
PDFC write
Function control2 (on bit basis)
PDFC2 write
R
Output latch
PD write
S
A
Selector
B
TB1OUT1
S
A
Selector
B
SCLK2
S
A
Selector
B
PD read
PD4
(TB1OUT1,SCLK2, $\overline{\text{CTS2}}$)
SCLK2, $\overline{\text{CTS2}}$

Figure 3.5.26 Port D (PD4)

Port D Register

		7	6	5	4	3	2	1	0
PD (0034H)	Bit symbol				PD4	PD3	PD2	PD1	PD0
	Read/Write				R/W			R	R/W
	Reset State				Data from external port (Note1)			Data from external port	Data from external port (Note1)

Port D Control Register

		7	6	5	4	3	2	1	0
PDCR (0036H)	Bit symbol				PD4C	PD3C	PD2C		PD0C
	Read/Write				W				
	Reset State				0	0	0		0
	Function				0: Input 1: Output				0: Input 1: Output

Port D Function Register

		7	6	5	4	3	2	1	0
PDFC (0037H)	Bit symbol				PD4F	PD3F	PD2F	PD1F	PD0F
	Read/Write				W				
	Reset State				0	0	0	0	0
	Function				Refer to following table				

Port D Function Register 2

		7	6	5	4	3	2	1	0
PDFC2 (0035H)	Bit symbol				PD4F2	PD3F2	PD2F2	PD1F2	
	Read/Write				W				
	Reset State				0	0	0	0	
	Function				Refer to following table				

PD4 to PD0 function setting

<PDxF2, PDxF, PDxC>	PD4	PD3	PD2	PD1 (Note 3)	PD0 (Note 4)
0 , 0 , 0	Input port	Input port	Input port	Input port	Input port
0 , 0 , 1	Output port	Output port	Output port		Output port
0 , 1 , 0	Reserved	RXD2	TB1IN1	TB1IN0	INT4
0 , 1 , 1	TB1OUT1	TB1OUT0	TXD2(3-state)		TB0OUT0
1 , 0 , 0	SCLK2, $\overline{\text{CTS2}}$ input	INT7	INT6	INT5	
1 , 0 , 1	SCLK2 output	Reserved	Reserved		
1 , 1 , 0	Reserved	Reserved	Reserved	Reserved	
1 , 1 , 1	Reserved	Reserved	TXD2(O.D)		

Note : <PDxF2>,<PDxF> and <PDxC> are the bits x of PDFC2,PDFC and PDCR registers.

Note 1: Output latch register is cleared to “0”.

Note 2: There is no output latch register in PD1.

Note 3: Read-modify-write instructions are prohibited for PDCR, PDFC and PDFC2.

Note 4: TB1IN0 and TB1IN1 input is inputted into the 16-bit timer TMRB1 irrespective of a functional setup of a port.

Note 5: RXD2, SCLK2 input, and $\overline{\text{CTS2}}$ input are inputted into the serial channel 2 irrespective of a functional setup of a port.

Note 6: PD2 does not have a register for 3-state/open drain setup. Moreover, there is no open drain function at the time of an output port.

Figure 3.5.27 Register for Port D

3.5.10 Port F (PF0 to PF5)

Port F is a 6-bit general-purpose I/O ports.

All the bits of PFCR, PFFC and PFFC2 are cleared to "0" by the reset action, and all bits serve as an input port.

In addition to an I/O port, there are I/O of the serial channels 0 and 1, high speed serial channel and an internal clock output function. These functions operate by setting the bit concerned of PFCR, PFFC, PFFC2, HSCSEL register as "1". All the bits of PFCR, PFFC, PFFC2 and HSCSEL are cleared to "0" by the reset action, and all bits serve as an input port.

(1) Port F0 (TXD0)

In addition to an I/O port function, PF0 have a function as an output (TXD0) of the serial channels 0 .

Moreover, when using it as a TXD output terminal, the output buffer has the open drain function in which a program is possible. An open drain function can be set up by the PFFC <PF0F>, PFCR <PF0C> register.

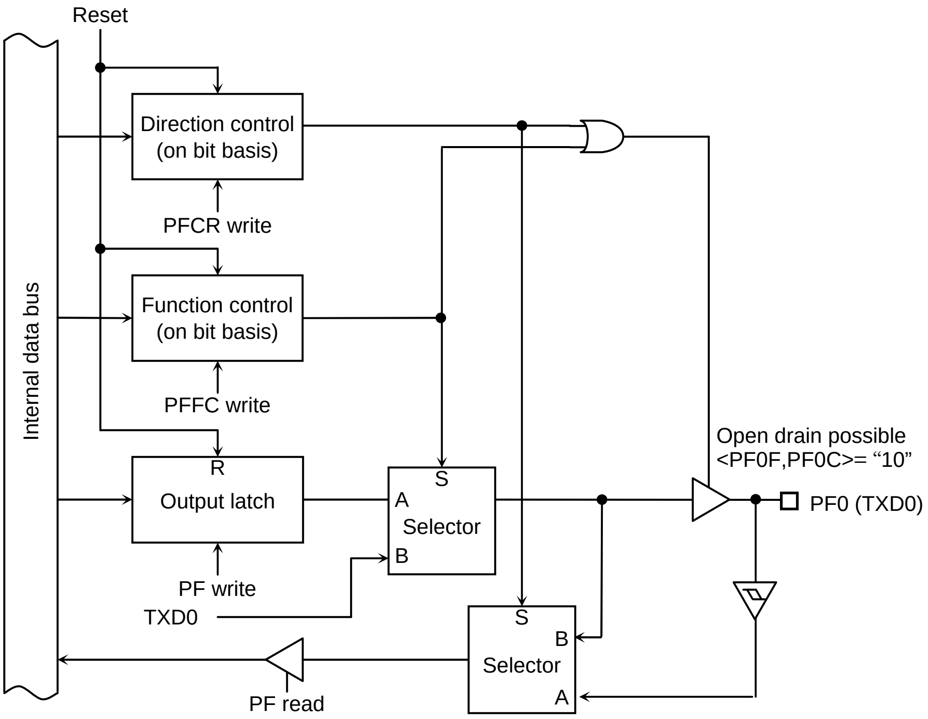


Figure 3.5.28 Port F (PF0)

(2) PF1(RXD0)

In addition to the I/O port, PF1 have a function as an input (RXD0) of the serial channels 0 .

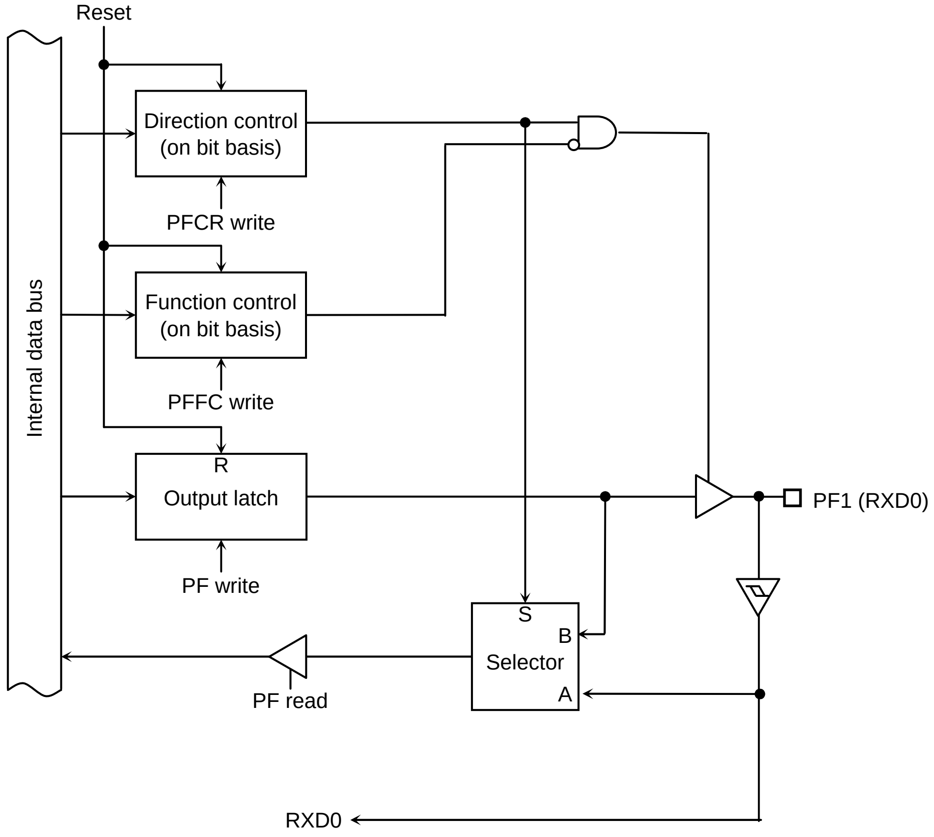


Figure 3.5.29 Port F (PF1)

(3) PF2 ($\overline{\text{CTS0}}$, SCLK0, CLK)

In addition to the I/O port, PF2 has a function as the CTS input ($\overline{\text{CTS0}}$), SCLK I/O (SCLK0), and the internal clock output (CLK) of the serial channel 0.

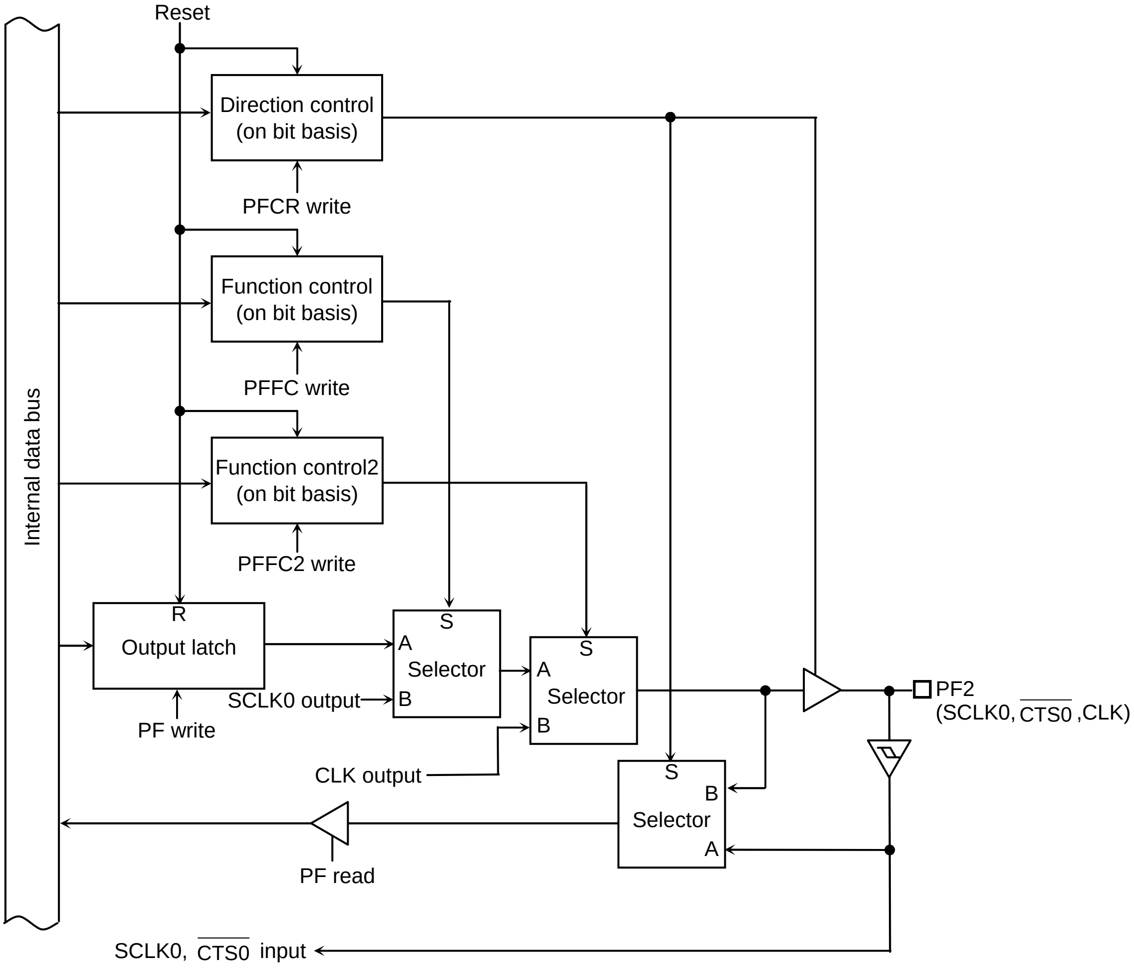


Figure 3.5.30 Port F (PF2)

(4) Port F3 (TXD1, HSSO)

In addition to an I/O port function, PF3 have a function as an output (TXD1) of the serial channels 1 and output (HSSO) of the high speed serial channels.

Moreover, when using it as a TXD output terminal, the output buffer has the open drain function in which a program is possible. An open drain function can be set up by the PFFC <PF3F>, PFCR <PF3C> register.

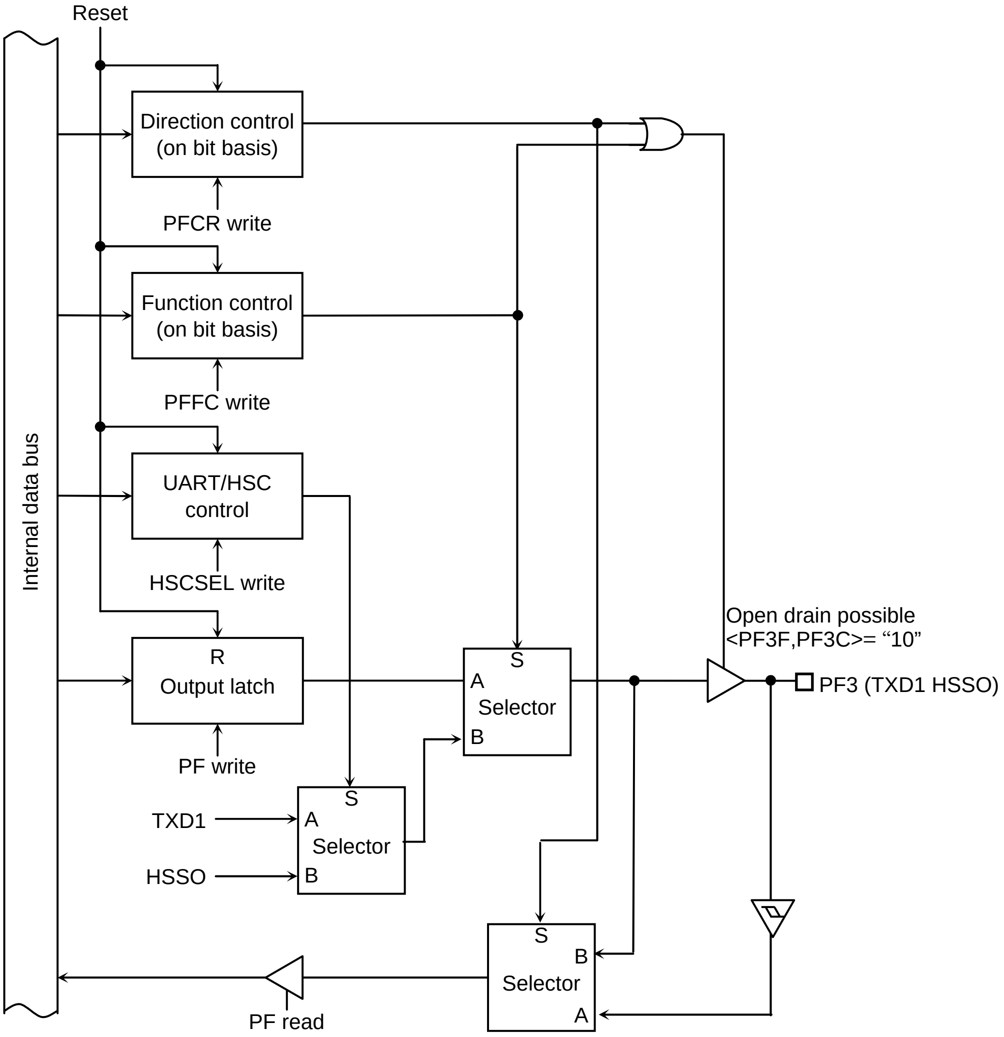


Figure 3.5.31 Port F (PF3)

(5) PF4(RXD1, HSSI)

In addition to the I/O port, PF4 have a function as an input (RXD1) of the serial channels 0 and input (HSSI) of high speed serial channels.

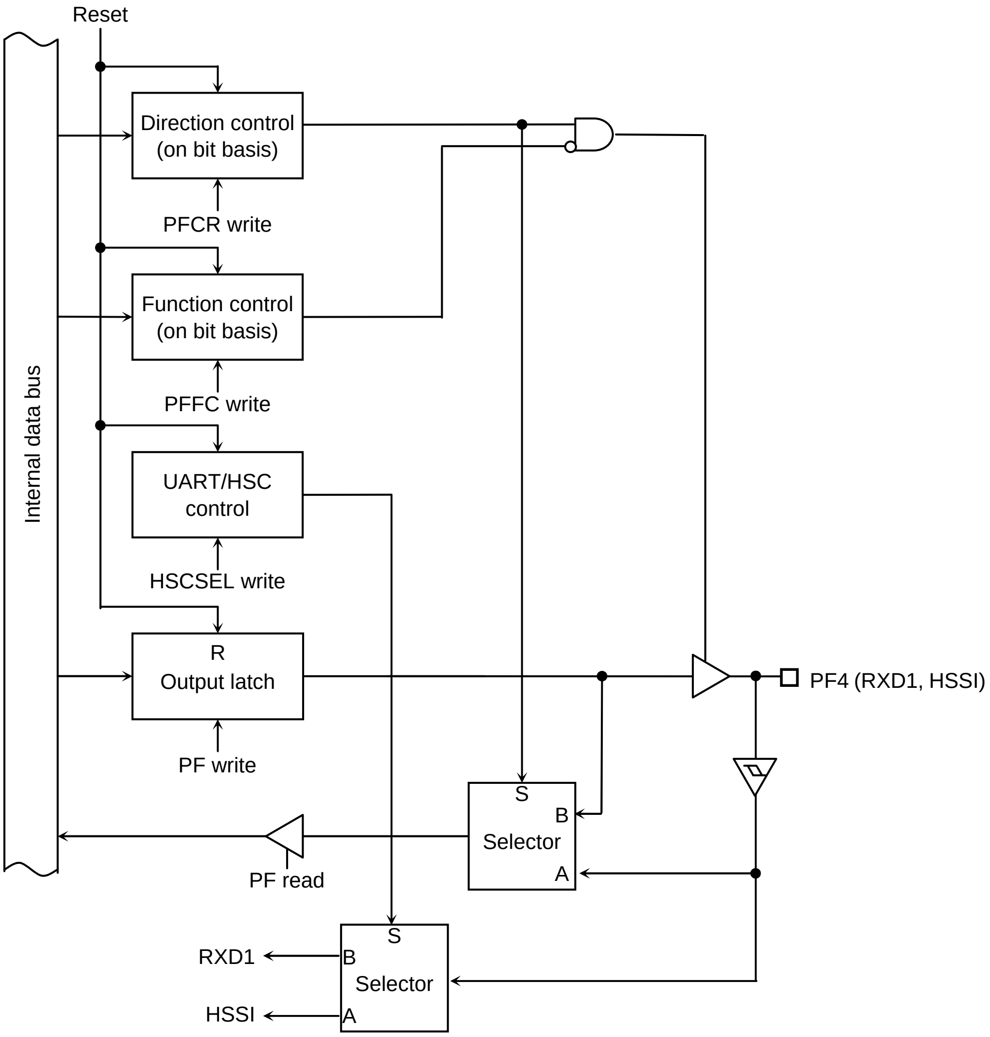


Figure 3.5.32 Port F (PF4)

(6) PF5 ($\overline{\text{CTS1}}$, SCLK1, HSCLK)

In addition to the I/O port function, PF5 has a function as the input ($\overline{\text{CTS1}}$) or I/O (SCLK1) of the serial channel 1 and output (HSCLK) of high speed serial channels.

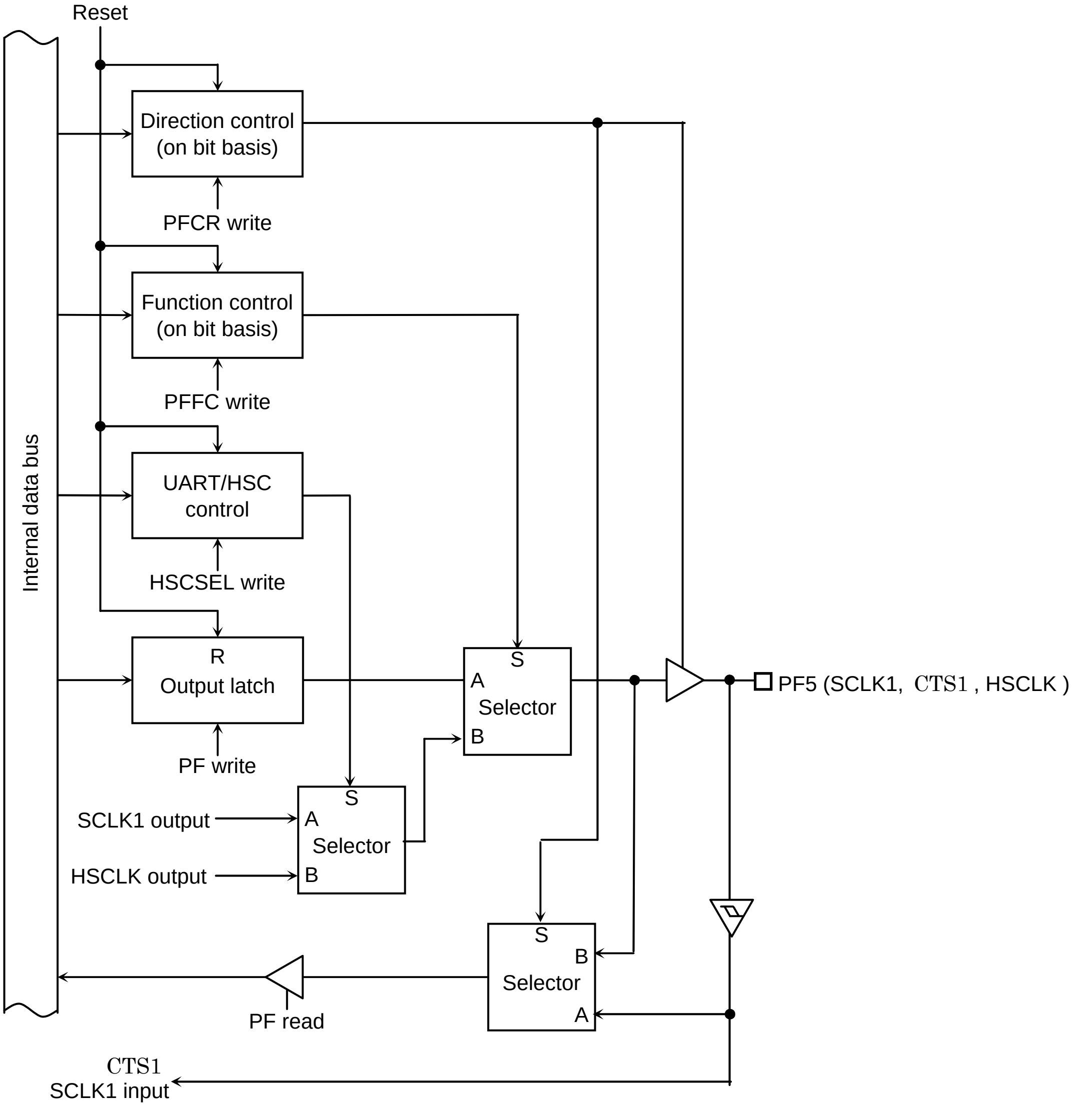


Figure 3.5.33 Port F (PF5)

Port F Register

PF (003CH)

	7	6	5	4	3	2	1	0
Bit symbol			PF5	PF4	PF3	PF2	PF1	PF0
Read/Write			R/W					
Reset State			Data from external port (Output latch register is cleared to “0”)					

Port F Control Register

PFCR (003EH)

	7	6	5	4	3	2	1	0
Bit symbol			PF5C	PF4C	PF3C	PF2C	PF1C	PF0C
Read/Write			W					
Reset State			0	0	0	0	0	0
Function			0: Input 1: Output					

Port F Functon Register

PFFC (003FH)

	7	6	5	4	3	2	1	0
Bit symbol			PF5F	PF4F	PF3F	PF2F	PF1F	PF0F
Read/Write			W					
Reset State			0	0	0	0	0	0
Function			0: Port 1: SCLK1 $\overline{\text{CTS1}}$	0: Port 1: RXD1	0: Port 1: TXD1	0: Port 1: SCLK0 $\overline{\text{CTS0}}$	0: Port 1: RXD0	0: Port 1: TXD0

Port F Functon Register 2

PFFC2 (003DH)

	7	6	5	4	3	2	1	0
Bit symbol						PF2F2		
Read/Write						W		
Reset State						0		
Function						0: <PF2F> 1: CLK		

SIO1/ HSC Control Register

HSCSEL (00F4H)

	7	6	5	4	3	2	1	0
Bit symbol	–	–	–	–	–	–	–	SIOCNT
Read/Write	R							R/W
Reset State	0	0	0	0	0	0	0	0
Function								0: SIO1 1: HSC

PF5 to PF0 function setting

<PFxF2, PFxF, PFxC>	PF2	PF1	PF0
0 , 0 , 0	Input port	Input port	Input port
0 , 0 , 1	Output port	Output port	Output port
0 , 1 , 0	SCLK0, $\overline{\text{CTS0}}$ input	RXD0 input	TXD0 (O.D output)
0 , 1 , 1	SCLK0 output	Reserved	TXD0 (3-state)
1 , 0 , 0	Reserved		
1 , 0 , 1	CLK output		
1 , 1 , 0	Reserved		
1 , 1 , 1	Reserved		
<SIOCNT, PFxF, PFxC>	PF5	PF4	PF3
0 , 0 , 0	Input port	Input port	Input port
0 , 0 , 1	Output port	Output port	Output port
0 , 1 , 0	SCLK1, $\overline{\text{CTS1}}$ input	RXD1 input	TXD1 (O.D output)
0 , 1 , 1	SCLK1 output	Reserved	TXD1 (3-state)
1 , 0 , 0	Reserved	Reserved	Reserved
1 , 0 , 1	Reserved	Reserved	Reserved
1 , 1 , 0	Reserved	HSSI 入力	Reserved
1 , 1 , 1	HSCLK output	Reserved	HSSO (3-state)

Note : <PFxF2>,<PFxF> and <PFxC> are the bits x of PFFC2,PFFC and PFCR registers.

Note 1: Read-modify-write instructions are prohibited for PDCR, PDFC and PDFC2.

Note 2: PF0 and PF3 does not have a register for 3-state/open drain setup. Moreover, there is no open drain function at the time of an output port.

Figure 3.5.34 Register for Port F

3.5.11 Port G (PG0 to PG7)

Port G is 8-bit general-purpose input ports. In addition to an input port function, there are an analog input for AD converters (AN0 to AN7) and a key input (KI0 to KI7) function for a Key on wake up. These functions operate by setting the bit concerned of PGFC, KIEN register as "1". Moreover, edge selection of a key input is set up by the KICR register.

By the reset action, all the bits of PGFC are set to "1", and all the bits of KIEN are cleared to "0", and it becomes all bit analog input ports (port input disable).

A key input is enabled by the KIEN register, and when the edge chosen in the KICR register is detected, the Key on wake up input KWI occurs. Although a Key on wake up input can release all HALT mode states, there is no function as interrupt.

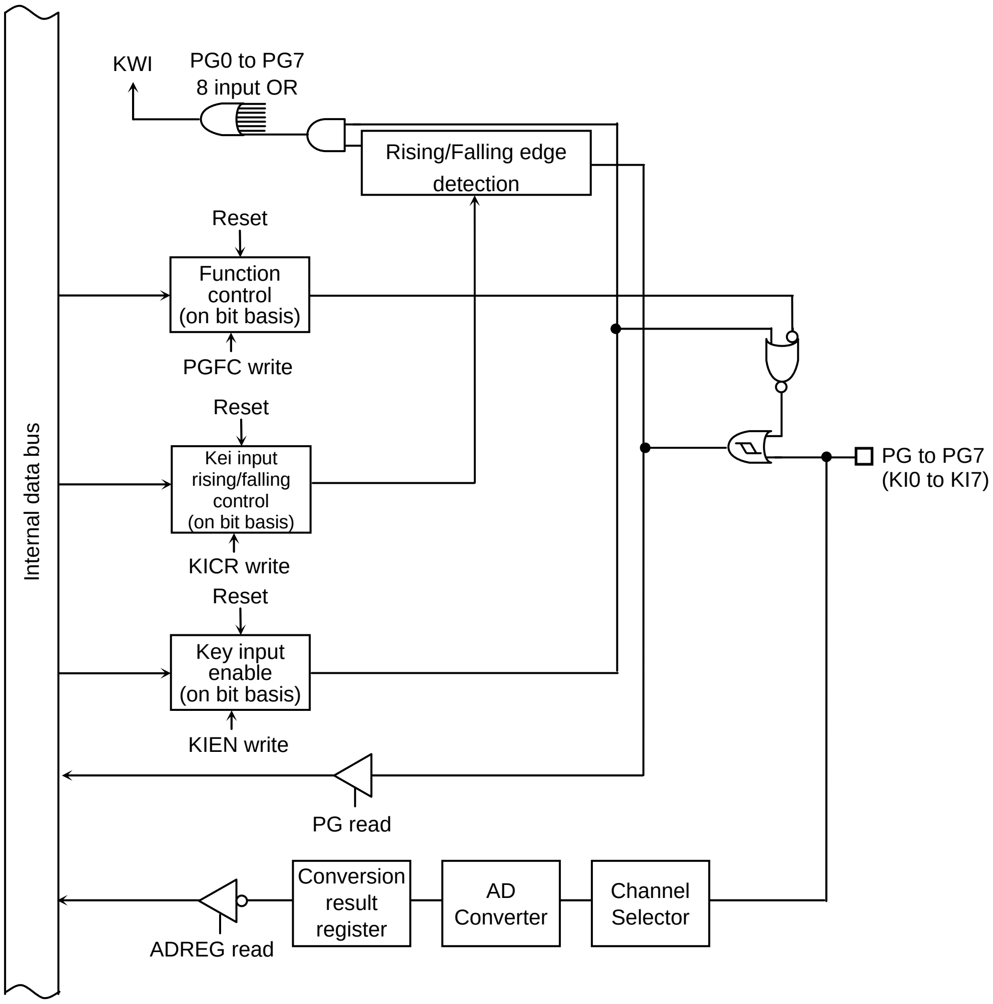

KWI
PG0 to PG7
8 input OR
Rising/Falling edge detection
Reset
Function control (on bit basis)
PGFC write
Reset
Kei input rising/falling control (on bit basis)
KICR write
Reset
Key input enable (on bit basis)
KIEN write
Internal data bus
PG to PG7 (KI0 to KI7)
PG read
Conversion result register
AD Converter
Channel Selector
ADREG read

Figure 3.5.35 Port G

Port G Register

PG
(0040H)

	7	6	5	4	3	2	1	0
Bit symbol	PG7	PG6	PG5	PG4	PG3	PG2	PG1	PG0
Read/Write	R							
Reset State	Data from external port (Note1)							

Port G Function Register

PGFC
(0043H)

	7	6	5	4	3	2	1	0
Bit symbol	PG7F	PG6F	PG5F	PG4F	PG3F	PG2F	PG1F	PG0F
Read/Write	W							
Reset State	1	1	1	1	1	1	1	1
Function	0: Analog input 1: Input port/Key input							

Key input Enable Register

KIEN
(13A0H)

	7	6	5	4	3	2	1	0
Bit symbol	KI7EN	KI6EN	KI5EN	KI4EN	KI3EN	KI2EN	KI1EN	KI0EN
Read/Write	W							
Reset State	0	0	0	0	0	0	0	0
Function	KI7 input 0: Disable 1: Enable	KI6 input 0: Disable 1: Enable	KI5 input 0: Disable 1: Enable	KI4 input 0: Disable 1: Enable	KI3 input 0: Disable 1: Enable	KI2 input 0: Disable 1: Enable	KI1 input 0: Disable 1: Enable	KI0 input 0: Disable 1: Enable

Key input Control Register

KICR
(13A1H)

	7	6	5	4	3	2	1	0
Bit symbol	KI7EDGE	KI6EDGE	KI5EDGE	KI4EDGE	KI3EDGE	KI2EDGE	KI1EDGE	KI0EDGE
Read/Write	W							
Reset State	0	0	0	0	0	0	0	0
Function	KI7 edge 0: Rising 1: Falling	KI6 edge 0: Rising 1: Falling	KI5 edge 0: Rising 1: Falling	KI4 edge 0: Rising 1: Falling	KI3 edge 0: Rising 1: Falling	KI2 edge 0: Rising 1: Falling	KI1 edge 0: Rising 1: Falling	KI0 edge 0: Rising 1: Falling

PG7 to PG0 function setting

<PGxF> <KIxEN>	0	1
0	Input port	Analog input
1	Key input	Reserved

Note : <PGxF> and <KIxEN> are the bits x of PGFC and KIEN registers.

Note 1: It operates as an analog input port (Input port disable).

Note 2: Read-modify-write instructions are prohibited for PGFC,KIEN and KICR.

Note 3: Input channel selection of an AD conberter is set up by AD mode control register ADMOD1.

Figure 3.5.36 Register for Port G

3.5.12 Port L (PL0 to PL3)

Port L is a 4-bit input port. In addition to an input port function, Port L has the analog input function of an AD converter. Moreover, PL3 has the $\overline{\text{ADTRG}}$ function of an AD converter. When you use PL3 as an $\overline{\text{ADTRG}}$, set PLFC <PL3F> as "0". All the bits of a PLFC register are set to "1" by the reset action, and Port L become analog input port (port input disable).

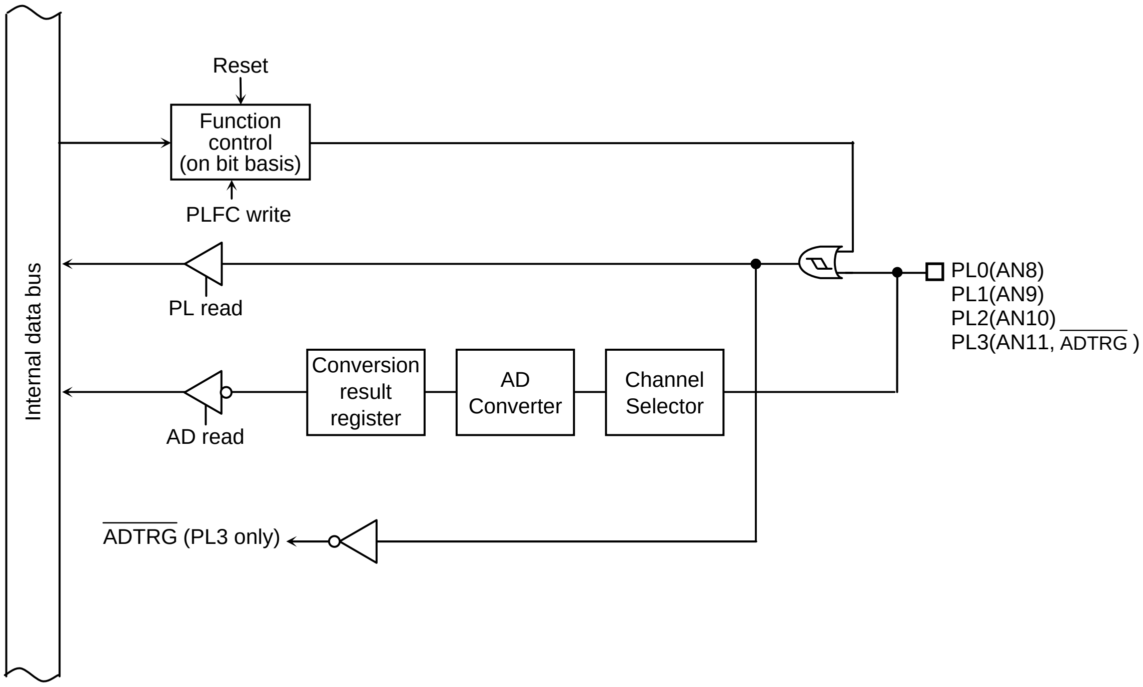

Internal data bus
Reset
Function control (on bit basis)
PLFC write
PL read
Conversion result register
AD Converter
Channel Selector
AD read
$\overline{\text{ADTRG}}$ (PL3 only)
PL0(AN8)
PL1(AN9)
PL2(AN10)
PL3(AN11, $\overline{\text{ADTRG}}$)

Figure 3.5.37 Port L

Port L Register

		7	6	5	4	3	2	1	0
PL	Bit symbol					PL3	PL2	PL1	PL0
(0054H)	Read/Write					R			
	Reset State					Data from external port (Note1)			

Port L Function Register

		7	6	5	4	3	2	1	0
PLFC	Bit symbol					PL3F	PL2F	PL1F	PL0F
(0057H)	Read/Write					W			
	Reset State					1	1	1	1
	Function					0: Analog input 1:Input port (Note3)			

Note 1: It operates as an analog input port (Input port disable).

Note 2: Read-modify-write instructions are prohibited for PLFC.

Note 3: Input channel selectino of an AD converter is set up by AD mode control register ADMOD1<ADCH3:0>. Moreover, a set up of AD trigger ($\overline{\text{ADTRG}}$) input oermission is set up by ADMOD2<ADTRGE>.

Figure 3.5.38 Register for Port L

3.5.13 Port N (PN0 to PN5)

Port N is 6-bit general-purpose I/O ports. Moreover, PN1, PN2, PN4, and PN5 serve as an open drain output, when it is set as an output.

There are the following functions in addition to an I/O port.

- The I/O function of the serial bus interface 0 (SCK0, SO0/SDA0, SI0/SCL0)
- The I/O function of the serial bus interface 1 (SCK1, SO1/SDA1, SI1/SCL1)

These functions operate by setting the bit concerned of PNCR, PNFC register as “1”. All the bits of PNCR and PNFC are cleared to "0" by the reset action, and all bits serve as an input port. Moreover, all the bits of an output latch are set to “1”.

(1) PN0 (SCK0), PN3 (SCK1)

PN0 and PN3 are general-purpose I/O ports. It is also used as a SCK (clock I/O signal in SIO mode).

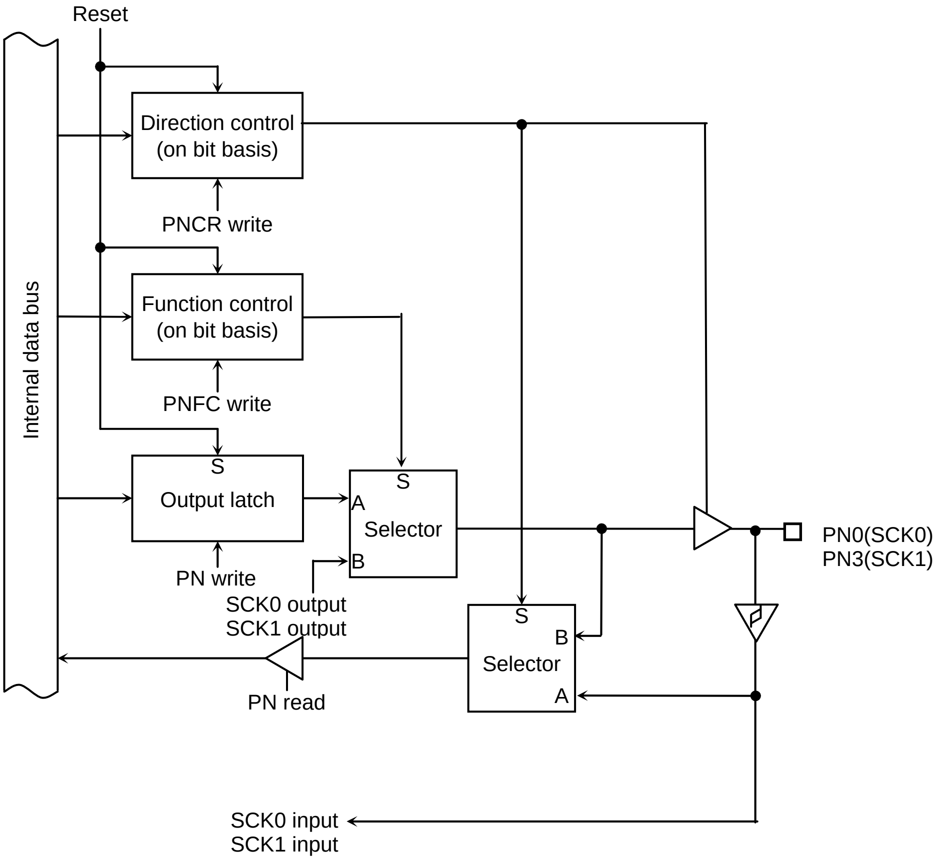


Figure 3.5.39 Port N (PN0, PN3)

(2) PN1 (SDA0/SO0), PN4 (SDA1/SO1)

PN1 and PN4 are general-purpose I/O ports. It is also used as a SO (data output signal in SIO mode), and SDA (data signal in I2CBUS mode). Moreover, these ports serve as an open drain output.

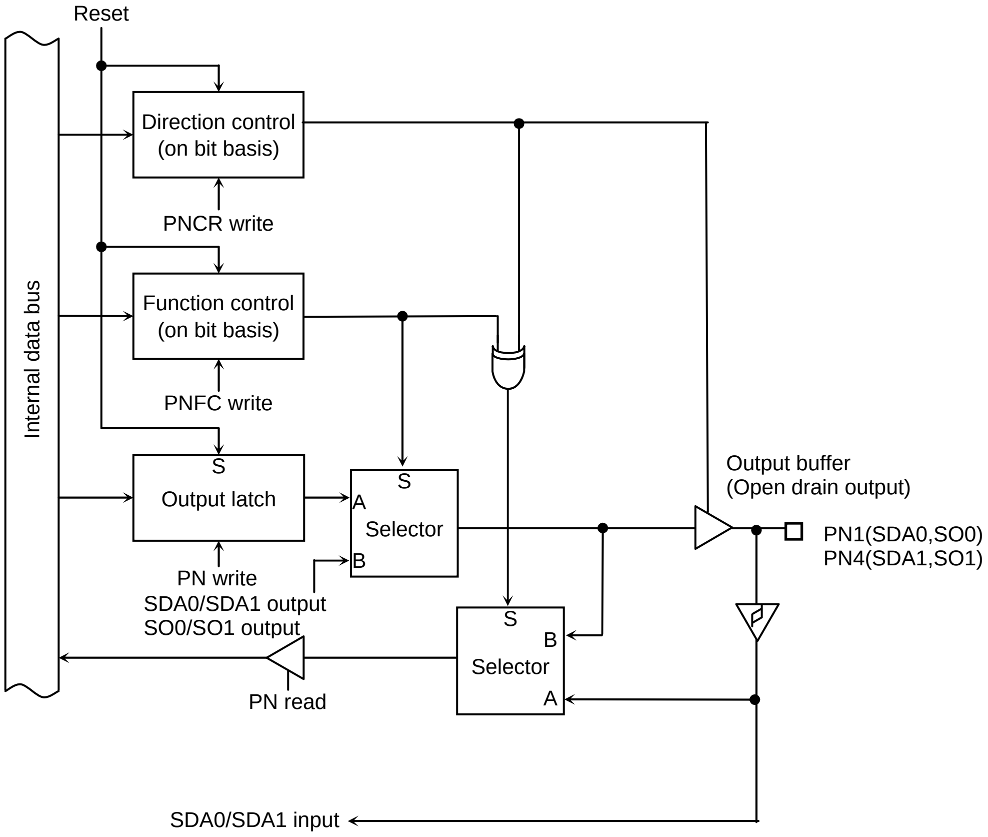


Figure 3.5.40 Port N (PN1, PN4)

(3) PN2 (SCL0/SI0), PN5 (SCL1/SI1)

PN2 and PN5 are general-purpose I/O ports. It is also used as a SI (data input signal in SIO mode), and SCL (clock signal in I²CBUS mode). Moreover, these ports serve as an open drain output.

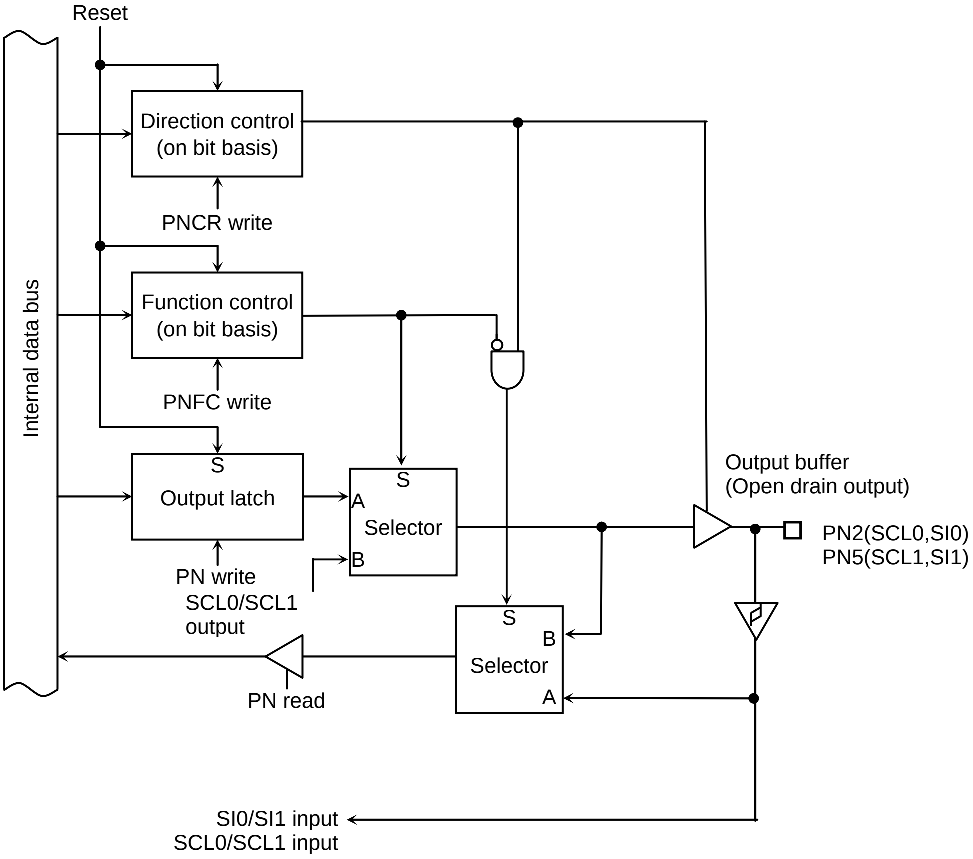


Figure 3.5.41 Port N (PN2, PN5)

Port N Register

		7	6	5	4	3	2	1	0
PN	Bit symbol			PN5	PN4	PN3	PN2	PN1	PN0
(005CH)	Read/Write			R/W					
	Reset State			Data from external port (Output latch register is set to “1”)					

Port N Control Register

		7	6	5	4	3	2	1	0
PNCR	Bit symbol			PN5C	PN4C	PN3C	PN2C	PN1C	PN0C
(005EH)	Read/Write			W					
	Reset State			0	0	0	0	0	0
	Function			0: Input 1: Output					

Port N Function Register

		7	6	5	4	3	2	1	0
PNFC	Bit symbol			PN5F	PN4F	PN3F	PN2F	PN1F	PN0F
(005FH)	Read/Write			W					
	Reset State			0	0	0	0	0	0
	Function			0: Port 1: SI1, SCL1	0: Port 1: SO1,SDA1	0: Port 1: SCK1	0: Port 1: SI0, SCL0	0: Port 1: SO0 SDA0	0: Port 1: SCK0

PN5 to PN0 function setting

<PNxF, PNxC>	PN5	PN4	PN3	PN2	PN1	PN0
0 , 0 , 0	Input port	Input port	Input port	Input port	Input port	Input port
0 , 0 , 1	Output port	Output port	Output port	Output port	Output port	Output port
0 , 1 , 0	SI1 input	SO1 output	SCK1 input	SI0 input	SO0 output	SCK0 input
0 , 1 , 1	SCL1 input/output	SDA1 input/output	SCK1 output	SCL0 input/output	SDA0 input/output	SCK0 output

Note : <PNxF> and <PNxC> are the bits x of PNFC and PNCR registers.

Note 1: Read-modify-write instructions are prohibited for PNFC and PNCR.

Figure 3.5.42 Register for Port N

3.6 Memory Controller

3.6.1 Functions

TMP92FD23A has a memory controller with a variable 4-block address area that controls as follows.

(1) 4-block address area support

Specifies a start address and a block size for 4-block address area.

(2) Connecting memory specifications

Specifies SRAM and ROM as memories to connect with the selected address areas.

(3) Data bus size selection

Whether 8 bits, 16 bits is selected as the data bus size of the respective block address areas.

(4) Wait control

Wait specification bit in the control register and $\overline{\text{WAIT}}$ input pin control the number of waits in the external bus cycle. Read cycle and write cycle can specify the number of waits individually. The number of waits is controlled in 6 modes mentioned below.

0 waits, 1 wait,
2 waits, 3 waits, 4 waits
N waits (control with $\overline{\text{WAIT}}$ pin)

3.6.2 Control Register and Operation after Reset Release

This section describes the registers to control the memory controller, the state after reset release and necessary settings.

(1) Control register

The control registers of the memory controller are as follows.

- Control register: BnCSH/BnCSL (n = 0 to 3, EX)
  Sets the basic functions of the memory controller, that is the connecting memory type, the number of waits to be read and written.
- Memory start address register: MSARn (n = 0 to 3)
  Sets a start address in the selected address areas.
- Memory address mask register: MAMR (n = 0 to 3)
  Sets a block size in the selected address areas.
- Page ROM control register: PMEMCR
  Sets to executed ROM page mode accessing.

(2) Operation after reset release

After reset, only control register (B2CSH/B2CSL) of the block address area 2 is automatically valid (B2CSH<B2E> is set to “1” by reset).

Since the bus width specification bit of the control register of the block address area 2 becomes unfixed, please be sure to set up before accessing the external block address area 2.

The block address area 2 is set to address from 000000H to FFFFFFH after reset (B2CSH<B2M> is cleared to “0”).

After reset release, the block address areas are specified by the memory start address register (MSARn) and the memory address mask register (MAMRn). Then the control register (BnCSH/L) is set.

Set the enable bit (BnCSH<BnE>) of the control register to “1” to enable the setting

3.6.3 Basic Functions and Register Setting

In this section, setting of the block address area, the connecting memory, and the number of waits out of the memory controller’s functions are described.

(1) Block address area specification

The block address area is specified by two registers.

The memory start address register (MSARn) sets the start address of the block address areas. The memory controller compares between the register value and the address every bus cycles. The address bit which is masked by the memory address mask register (MAMRn) is not compared by the memory controller. The block address area size is determined by setting the memory address mask register. The set value in the register is compared with the block address area on the bus. If the compared result is a match, the memory controller sets the chip select signal ($\overline{\text{CSn}}$) to “low”.

(i) Setting memory start address register

The <MS23:MS16> bits of the memory start address register respectively correspond with addresses from A23 to A16. The lower start addresses from A15 to A0 are always set to address 0000H. Therefore the start addresses of the block address area are set to addresses from 000000H to FF0000H every 64 Kbytes.

(ii) Setting memory address mask registers

The memory address mask register sets whether an address bit is compared or not. Set the register to “0” to compare, or to “1” not to compare.

The address bit to be set is depended on the block address area.

Block address area 0: A20 to A8

Block address area 1: A21 to A8

Block address area 2 to 3: A22 to A15

The above-mentioned bits are always compared. The block address area size is determined by the compared result.

The size to be set depending on the block address area is as follows.

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

Note: After reset release, only the control register of the block address area 2 is valid. The control register of the block address area 2 has <B2M> bit. Setting <B2M> bit to “0” sets the block address area 2 to addresses from 000000H to FFFFFFH. Setting <B2M> bit to “1” specifies the start address and the address area size as it is in the other block address area.

(iii) Example of register setting

To set the block address area from 1 to 512 bytes from address 110000H, set the register as follows.

MSAR1 Register

	7	6	5	4	3	2	1	0
Bit symbol	M1S23	M1S22	M1S21	M1S20	M1S19	M1S18	M1S17	M1S16
Specified value	0	0	0	1	0	0	0	1

<M1S23:16> bits of the memory start address register MSAR1 correspond with address from A23 to A16.

From A15 to A0 are cleared to “0”. Therefore setting MSAR1 to the above-mentioned value specifies the start address of the block address area to address 110000H.

MAMR1 Register

	7	6	5	4	3	2	1	0
Bit symbol	M1V21	M1V20	M1V19	M1V18	M1V17	M1V16	M1V15 to M1V9	M1V8
Specified value	0	0	0	0	0	0	0	1

<M1V21:M1V16> and <M1V8> bits of the memory address mask register MAMR1 set whether address from A21 to A16 and A8 are compared or not. Set the register to “0” to compare, or to “1” not to compare. From M1V15 to M1V9 bits set whether address from A15 to A9 are compared or not with 1 bit. A23 and A22 are always compared.

If set like above setting, from A23 to A9 compare with the values set as the start addresses. Therefore 512 bytes of addresses 110000H to 1101FFH are set as the block address area 1. If compared with the addresses on the bus, the chip select signal $\overline{\text{CS1}}$ is set to “LOW”.

A23 to A21 are always compared in the block address area 0. Whether from A20 to A8 are compared or not is set to register.

Similarly, A23 is always compared in block address areas 2 to 3. Whether A22 to A15 are compared or not is set to register.

Note: When the set block address area overlaps with the built-in memory area, or both two address areas overlap, the block address area is processed according to priority as follows.

Built-in I/O > Built-in memory > Block address area 0 > 1 > 2 > 3

Also that any accessed areas outside the address areas set by $\overline{\text{CS0}}$ to $\overline{\text{CS3}}$ are processed as the $\overline{\text{CSEX}}$ space. Therefore, settings of $\overline{\text{CSEX}}$ (BEXCSH/L) apply for the control of wait cycles, data bus width, etc.

(2) Connection memory specification

Setting the <BnOM1:0> bit of the control register (BnCSH) specifies the memory type to be connected with the block address areas. The interface signal is output according to the set memory as follows.

<BnOM1:0> Bit (BnCSH register)

BnOM1	BnOM0	Function
0	0	SRAM/ROM (Default)
0	1	(Reserved)
1	0	(Reserved)
1	1	(Reserved)

(3) Data bus width specification

The data bus width is set for every block address area. The bus size is set by the <BnBUS1:0> bits of the control register (BnCSH) as follows.

<BnBUS1:0> Bit (BnCSH register)

BnBUS1	BnBUS0	Function
0	0	8-bit bus mode (Note 2)
0	1	16-bit bus mode
1	0	(Reserved)
1	1	(Reserved)

This way of changing the data bus size depending on the address being accessed is called “dynamic bus sizing”. The part where the data is output to is depended on the data size, the bus width and the start address.

Note1: Since there is a possibility of abnormal writing/reading of the data if two memories with different bus width are put in consecutive address, do not execute a access to placed on both memories with one command.

Note2: Since after reset becomes unfixed, please be sure to set up bus width specification bit B2CSH <B2BUS1:0> of the control register of the block address area 2 before accessing the external block address area 2.

Operand Data Size (Bit)	Operand Start Address	Memory Data Size (Bit)	CPU Address	CPU Data			
				D32 to D24	D23 to D16	D15 to D8	D7 to D0
8	4n + 0	8/16	4n + 0	xxxxx	xxxxx	xxxxx	b7 to b0
	4n + 1	8	4n + 1	xxxxx	xxxxx	xxxxx	b7 to b0
		16	4n + 1	xxxxx	xxxxx	b7 to b0	xxxxx
	4n + 2	8/16	4n + 2	xxxxx	xxxxx	xxxxx	b7 to b0
	4n + 3	8	4n + 3	xxxxx	xxxxx	xxxxx	b7 to b0
		16	4n + 3	xxxxx	xxxxx	b7 to b0	xxxxx
16	4n + 0	8	(1) 4n + 0	xxxxx	xxxxx	xxxxx	b7 to b0
			(2) 4n + 1	xxxxx	xxxxx	xxxxx	b15 to b8
		16	4n + 0	xxxxx	xxxxx	b15 to b8	b7 to b0
	4n + 1	8	(1) 4n + 1	xxxxx	xxxxx	xxxxx	b7 to b0
			(2) 4n + 2	xxxxx	xxxxx	xxxxx	b15 to b8
		16	(1) 4n + 1	xxxxx	xxxxx	b7 to b0	xxxxx
			(2) 4n + 2	xxxxx	xxxxx	xxxxx	b15 to b8
	4n + 2	8	(1) 4n + 2	xxxxx	xxxxx	xxxxx	b7 to b0
			(2) 4n + 1	xxxxx	xxxxx	xxxxx	b15 to b8
		16	4n + 2	xxxxx	xxxxx	b15 to b8	b7 to b0
	4n + 3	8	(1) 4n + 3	xxxxx	xxxxx	xxxxx	b7 to b0
			(2) 4n + 4	xxxxx	xxxxx	xxxxx	b15 to b8
		16	(1) 4n + 3	xxxxx	xxxxx	b7 to b0	xxxxx
			(2) 4n + 4	xxxxx	xxxxx	xxxxx	b15 to b8
32	4n + 0	8	(1) 4n + 0	xxxxx	xxxxx	xxxxx	b7 to b0
			(2) 4n + 1	xxxxx	xxxxx	xxxxx	b15 to b8
			(3) 4n + 2	xxxxx	xxxxx	xxxxx	b23 to b16
			(4) 4n + 3	xxxxx	xxxxx	xxxxx	b31 to b24
		16	(1) 4n + 0	xxxxx	xxxxx	b15 to b8	b7 to b0
			(2) 4n + 2	xxxxx	xxxxx	b31 to b24	b23 to b16
	4n + 1	8	(1) 4n + 0	xxxxx	xxxxx	xxxxx	b7 to b0
			(2) 4n + 1	xxxxx	xxxxx	xxxxx	b15 to b8
			(3) 4n + 2	xxxxx	xxxxx	xxxxx	b23 to b16
			(4) 4n + 3	xxxxx	xxxxx	xxxxx	b31 to b24
		16	(1) 4n + 1	xxxxx	xxxxx	b7 to b0	xxxxx
			(2) 4n + 2	xxxxx	xxxxx	b23 to b16	b15 to b8
			(3) 4n + 4	xxxxx	xxxxx	xxxxx	b31 to b24
	4n + 2	8	(1) 4n + 2	xxxxx	xxxxx	xxxxx	b7 to b0
			(2) 4n + 3	xxxxx	xxxxx	xxxxx	b15 to b8
			(3) 4n + 4	xxxxx	xxxxx	xxxxx	b23 to b16
			(4) 4n + 5	xxxxx	xxxxx	xxxxx	b31 to b24
		16	(1) 4n + 2	xxxxx	xxxxx	b15 to b8	b7 to b0
			(2) 4n + 4	xxxxx	xxxxx	b31 to b24	b23 to b16
	4n + 3	8	(1) 4n + 3	xxxxx	xxxxx	xxxxx	b7 to b0
			(2) 4n + 4	xxxxx	xxxxx	xxxxx	b15 to b8
			(3) 4n + 5	xxxxx	xxxxx	xxxxx	b23 to b16
			(4) 4n + 6	xxxxx	xxxxx	xxxxx	b31 to b24
		16	(1) 4n + 3	xxxxx	xxxxx	b7 to b0	xxxxx
			(2) 4n + 4	xxxxx	xxxxx	b23 to b16	b15 to b8
			(3) 4n + 6	xxxxx	xxxxx	xxxxx	b31 to b24

xxxxx: During a read, data input to the bus is ignored. At write, the bus is at high impedance and the write strobe signal remains to non active.

(4) Wait control

The external bus cycle completes a wait of two states at least (100 ns at f_{SYS} = 20 MHz).

Setting the <BnWW2:0> and <BnWR2:0> of BnCSL specifies the number of waits in the read cycle and the write cycle. <BnWW2:0> is set with the same method as <BnWR2:0>. A setup is performed as follows.

<BnWW2:0>/<BnWR2:0> Bit (BnCSL Register)

BnWW2	BnWW1	BnWW0	Function
BnWR2	BnWR1	BnWR0	
0	0	1	2states (0 waits) access fixed mode
0	1	0	3states (1 wait) access fixed mode (Default)
1	0	1	4states (2 waits) access fixed mode
1	1	0	5states (3 waits) access fixed mode
1	1	1	6states (4 waits) access fixed mode
0	1	1	$\overline{\text{WAIT}}$ pin input mode
Others			(Reserved)

(i) Waits number fixed mode

The bus cycle is completed with the set states. The number of states is selected from 2 states (0 waits) to 6 states (4 waits).

(ii) $\overline{\text{WAIT}}$ pin input mode

This mode samples the $\overline{\text{WAIT}}$ input pins. It continuously samples the $\overline{\text{WAIT}}$ pin state and inserts a wait if the pin is active. The bus cycle is minimum 2 states. The bus cycle is completed when the wait signal is non-active ("High" level) at 2 states. The bus cycle extends if the wait signal is active at 2 states and more.

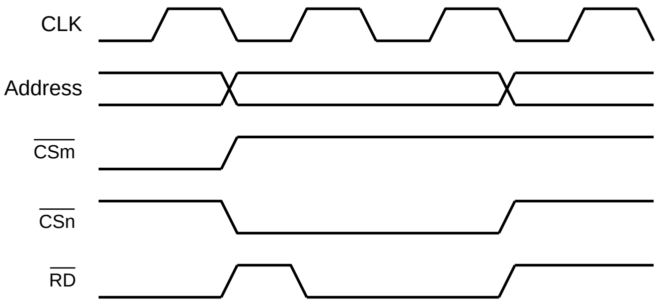
(5) Insert recovery cycle

If a lot of connected pertain ROM and etc. (Much data-output-floating-time (t_{DF})), each other's data-bus-output-recovery-time is trouble. However, by setting <BnREC> of control register (BnCSH), can to insert dummy cycle of 1-state just before first bus cycle of starting access another block address

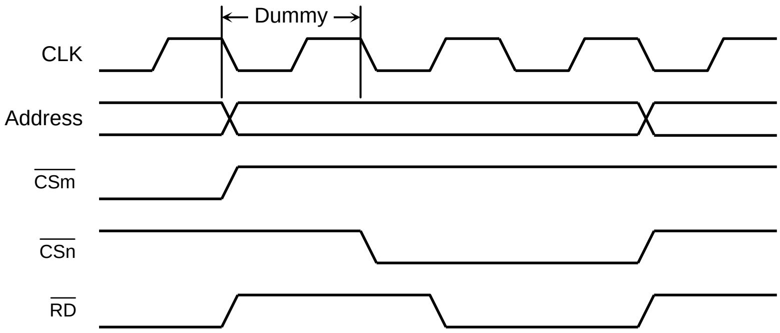
<BnREC> Bit (BnCSH register)

0	No dummy cycle is inserted (Default).
1	Dummy cycle is inserted.

- When not inserting a dummy cycle (0 waits)

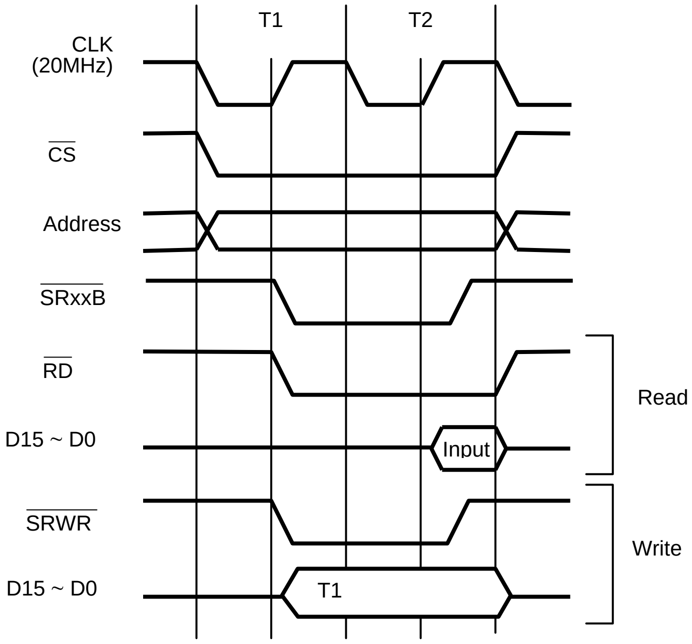


- When inserting a dummy cycle (0 waits)

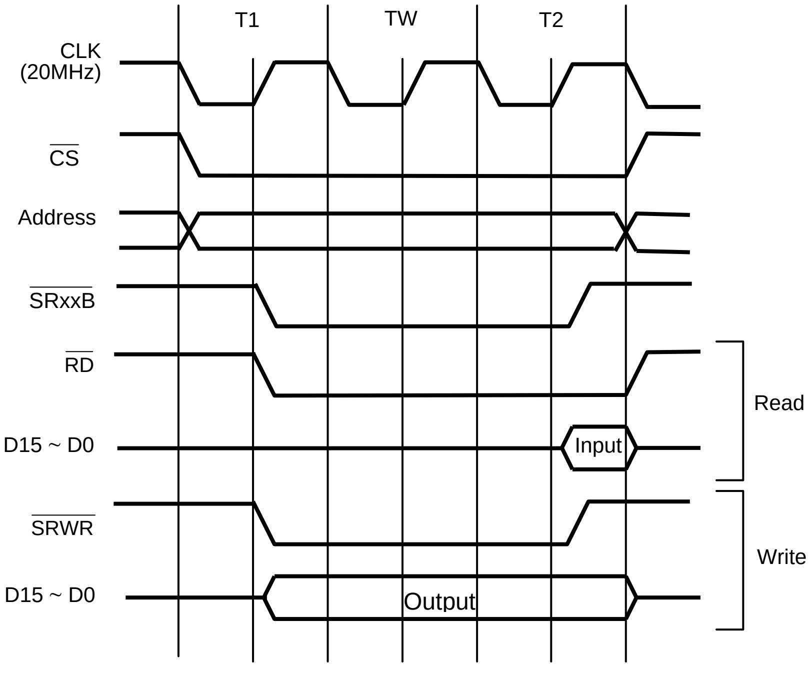


(6) Basic bus timing

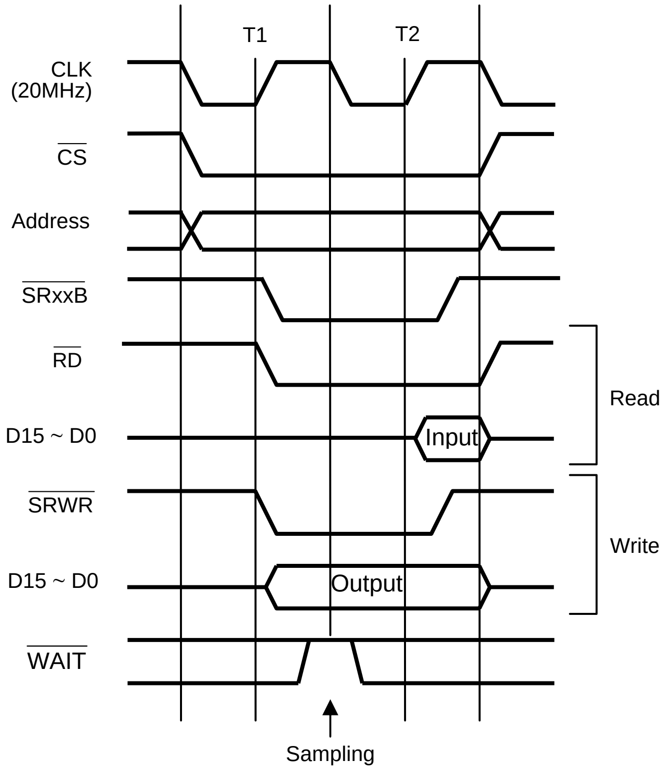
- External read/write bus cycle (0 waits)



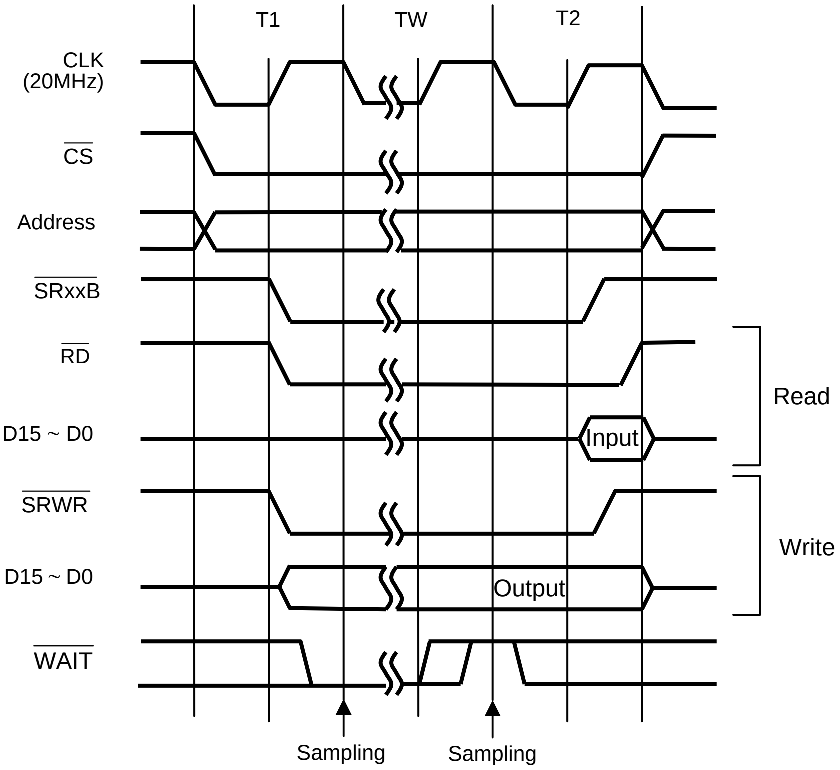
- External read/write bus cycle (1 wait)



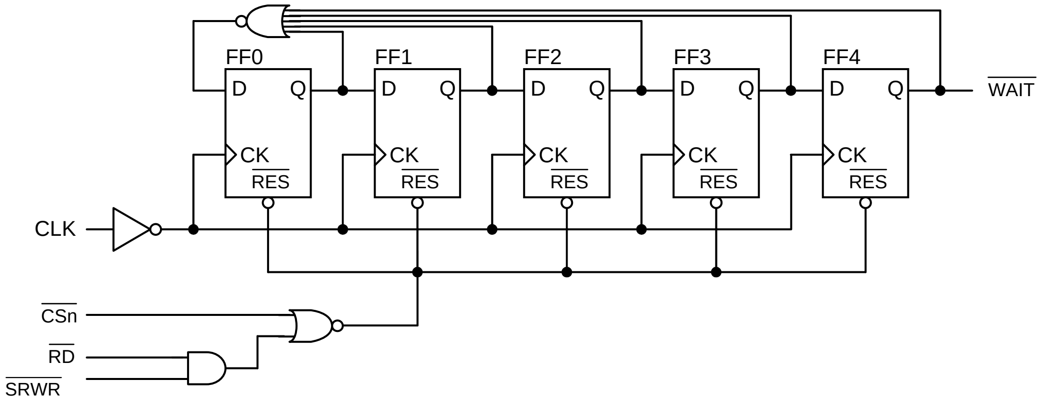
- External read/write bus cycle (0 waits at $\overline{\text{WAIT}}$ pin input mode)

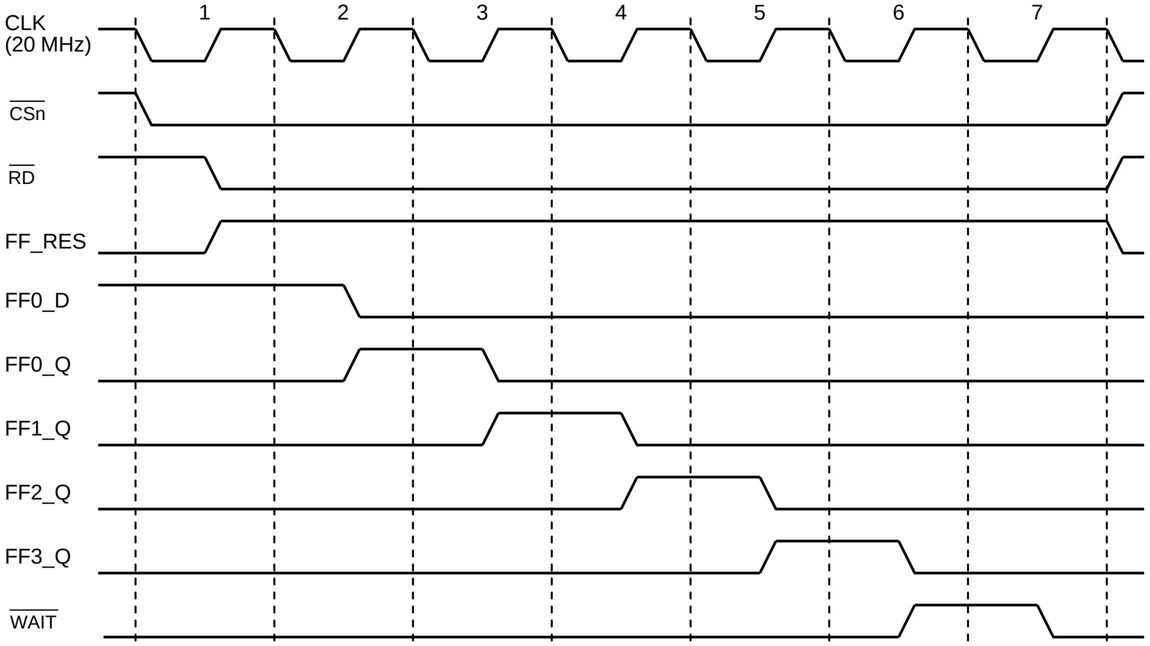


- External read/write bus cycle (n waits at $\overline{\text{WAIT}}$ pin input mode)



- Example of $\overline{\text{WAIT}}$ input cycle (5 waits)





3.6.4 ROM Control (Page mode)

This section describes ROM page mode accessing and how to set registers. ROM page mode is set by the page ROM control register.

(1) Operation and how to set the registers

TMP92FD23A supports ROM access of the page mode. The ROM access of the page mode is specified only in the block address area 2.

ROM page mode is set by the page ROM control register (PMEMCR).

Setting <OPGE> bit of the PMEMCR register to "1" sets the memory access of the block address area to ROM page mode access.

The number of read cycles is set by the <OPWR1:0> bits of the PMEMCR register.

<OPWR1:0> Bit (PMEMCR register)

OPWR1	OPWR0	Number of Cycle in a Page
0	0	1 state (n-1-1-1 mode) (n ≥ 2)
0	1	2 state (n-2-2-2 mode) (n ≥ 3)
1	0	3 state (n-3-3-3 mode) (n ≥ 4)
1	1	(Reserved)

Note: Set the number of waits "n" to the control register (B2CSL) in block address area 2.

The page size (the number of bytes) of ROM in the CPU size is set to the <PR1:0> bit of the PMEMCR register. When data is read out until a border of the set page, the controller completes the page reading operation. The start data of the next page is read in the normal cycle. The following data is set to page read again.

<PR1:0> Bit (PMEMCR register)

PR1	PR0	ROM Page Size
0	0	64 bytes
0	1	32 bytes
1	0	16 bytes
1	1	8 bytes

(2) Signal timing pulse

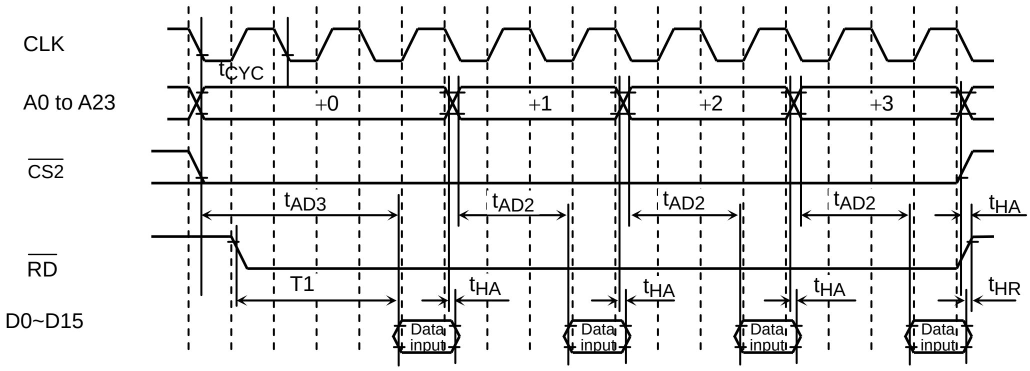


Figure 3.6.1 Timing Pulse diagram (8-bit setting)

3.6.5 List of Registers

The memory control registers and the settings are described as follows. For the addresses of the registers, see Section 5 “Table of Special Function Registers (SFRs)”.

(1) Control registers

The control register is a pair of BnCSL and BnCSH. (“n” is a number of the block address area.) BnCSL has the same configuration regardless of the block address areas. In BnCSH, only B2CSH which is corresponded to the block address area 2 has a different configuration from the others.

BnCSL

	7	6	5	4	3	2	1	0
Bit symbol		BnWW2	BnWW1	BnWW0		BnWR2	BnWR1	BnWR0
Read/Write		W				W		
After reset		0	1	0		0	1	0

<BnWW2:0> Specifies the number of write waits.

001 = 2 states (0 waits) access   010 = 3 states (1 wait) access
101 = 4 states (2 waits) access   110 = 5 states (3 waits) access
111 = 6 states (4 waits) access   011 = $\overline{\text{WAIT}}$ pin input mode
Others = (Reserved)

<BnWR2:0> Specifies the number of read waits.

001 = 2 states (0 waits) access   010 = 3 states (1 wait) access
101 = 4 states (2 waits) access   110 = 5 states (3 waits) access
111 = 6 states (4 waits) access   011 = $\overline{\text{WAIT}}$ pin input mode
Others = (Reserved)

B2CSH

	7	6	5	4	3	2	1	0
Bit symbol	B2E	B2M	−	B2REC	B2OM1	B2OM0	B2BUS1	B2BUS0
Read/Write	W							
After reset	1	0	0	0	0	0	Undefined	Undefined

<B2E>: Enable bit

0 = No chip select signal output.
1 = Chip select signal output (Default).
Note: After reset release, only the enable bit <B2E> of B2CS register is valid (“1”).

<B2M>: Block address area specification

0 = Sets the block address area of CS2 to addresses 000000H to FFFFFFH (Default).
1 = Sets the block address area of CS2 to programmable.
Note: After reset release, the block address area 2 is set to addresses 000000H to FFFFFFH.

<B2REC>: Sets the dummy cycle for data output recovery time.

0 = Not insert a dummy cycle (Default).
1 = Insert a dummy cycle.

<B2OM1:0>

00 = SRAM or ROM (Default)
Others = (Reserved)

<B2BUS1:0> Sets the data bus width.

00 = 8 bits
01 = 16 bits
10 = (Reserved)
11 = (Reserved)
Note: The value of <B2BUS> bit is set according to the state of AM<1:0> pin after reset release.

BnCSH (n = 0, 1, 3)

	7	6	5	4	3	2	1	0
Bit symbol	BnE			BnREC	BnOM1	BnOM0	BnBUS1	BnBUS0
Read/Write	W			W				
After reset	0			0	0	0	0	0

<BnE>: Enable bit

0 = No chip select signal output (Default).
1 = Chip select signal output.
Note: After reset release, only the enable bit B2E of B2CS register is valid (“1”).

<BnREC>: Sets the dummy cycle for data output.

0 = Not insert a dummy cycle (Default).
1 = Insert a dummy cycle.

<BnOM1:0>

00 = SRAM or ROM (Default)
01 = (Reserved)
10 = (Reserved)
11 = (Reserved)

<BnBUS1:0> Sets the data bus width.

00 = 8 bits (Default)
01 = 16 bits
10 = (Reserved)
11 = (Reserved)

BEXCSL

	7	6	5	4	3	2	1	0
Bit symbol		BEXWW2	BEXWW1	BEXWW0		BEXWR2	BEXWR1	BEXWR0
Read/Write		W				W		
After reset		0	1	0		0	1	0

<BEXWW2:0> Specifies the number of write waits.

001 = 2 states (0 waits) access   010 = 3 states (1 wait) access
101 = 4 states (2 waits) access   110 = 5 states (3 waits) access
111 = 6 states (4 waits) access   011 = $\overline{\text{WAIT}}$ pin input mode
Others = (Reserved)

<BEXWR2:0> Specifies the number of read waits.

001 = 2 states (0 waits) access   010 = 3 states (1 wait) access
101 = 4 states (2 waits) access   110 = 5 states (3 waits) access
111 = 6 states (4 waits) access   011 = $\overline{\text{WAIT}}$ pin input mode
Others = (Reserved)

BEXCSH

	7	6	5	4	3	2	1	0
Bit symbol				BEXREC	BEXOM1	BEXOM0	BEXBUS1	BEXBUS0
Read/Write				W				
After reset				0	0	0	0	0

<BEXOM1:0>

00 = SRAM or ROM (Default)
01 = (Reserved)
10 = (Reserved)
11 = (Reserved)

<BEXBUS1:0>

00 = 8 bits (Default)
01 = 16 bits
10 = (Reserved)
11 = (Reserved)

(2) Block address register

A start address and an address area of the block address are specified by the memory start address register (MSARn) and the memory address mask register (MAMRn). The memory start address register sets all start address similarly regardless of the block address areas.

The bit to be set by the memory address mask register is depended on the block address area.

MSARn (n = 0 to 3)

	7	6	5	4	3	2	1	0
Bit symbol	MnS23	MnS22	MnS21	MnS20	MnS19	MnS18	MnS17	MnS16
Read/Write	R/W							
After reset	1	1	1	1	1	1	1	1

<MnS23:16> Sets a start address.

Sets the start address of the block address areas. <MnS23:16> are corresponding to the address from A23 to A16.

MAMR0

	7	6	5	4	3	2	1	0
Bit symbol	M0V20	M0V19	M0V18	M0V17	M0V16	M0V15	M0V14 to M0V9	M0V8
Read/Write	R/W							
After reset	1	1	1	1	1	1	1	1

<M0V20:8>

Enables or masks comparison of the addresses. <M0V20:8> are corresponding to addresses from A20 to A8. <M0V14:9> are corresponding to address from A14 to A9 by 1 bit. If “0” is set, the comparison between the value of the address bus and the start address is enabled. If “1” is set, the comparison is masked.

MAMR1

	7	6	5	4	3	2	1	0
Bit symbol	M1V21	M1V20	M1V19	M1V18	M1V17	M1V16	M1V15 to M1V9	M1V8
Read/Write	R/W							
After reset	1	1	1	1	1	1	1	1

<M1V21:8>

Enables or masks comparison of the addresses. <M1V21:8> are corresponding to addresses from A21 to A8. <M1V15:9> are corresponding to address from A15 to A9 by 1 bit. If “0” is set, the comparison between the value of the address bus and the start address is enabled. If “1” is set, the comparison is masked.

MAMRn (n = 2 to 3)

	7	6	5	4	3	2	1	0
Bit symbol	MnV22	MnV21	MnV20	MnV19	MnV18	MnV17	MnV16	MnV15
Read/Write	R/W							
After reset	1	1	1	1	1	1	1	1

<MnV22:15>

Enables or masks comparison of the addresses. <MnV22:15> are corresponding to addresses from A22 to A15. If “0” is set, the comparison between the value of the address bus and the start address is enabled. If “1” is set, the comparison is masked.

After a reset, MASR0 to MSAR3 and MSAR0 to MAMR3 are set to “FFH”. B0CSH<B0E>, B1CSH<B1E>, and B3CSH<B3E> are reset to “0”. This disabling the CS0, CS1, and CS3 areas. However, B2CSH<B2M> is reset to “0” and B2CSH<B2E> to “1”, and CS2 is enabled 000000H to FFFFFFH. Also the bus width and number of waits specified in BEXCSH/L are used for accessing address except the specified CS0 to CS3 area.

(3) Page ROM control register (PMEMCR)

The page ROM control register sets page ROM accessing. ROM page accessing is executed only in block address area 2.

PMEMCR

	7	6	5	4	3	2	1	0
Bit symbol				OPGE	OPWR1	OPWR0	PR1	PR0
Read/Write				R/W				
After reset				0	0	0	1	0

<OPGE> enable bit

0 = No ROM page mode accessing (Default)

1 = ROM page mode accessing

<OPWR1:0> Specifies the number of waits.

00 = 1 state (n-1-1-1 mode) (n ≥ 2) (Default)

01 = 2 states (n-2-2-2 mode) (n ≥ 3)

10 = 3 states (n-3-3-3 mode) (n ≥ 4)

11 = (Reserved)

Note: Set the number of waits “n” to the control register (BnCSL) in block address area.

<PR1:0> ROM page size

00 = 64 bytes

01 = 32 bytes

10 = 16 bytes (Default)

11 = 8 bytes

Table 3.6.1 Control Register (1/2)

		7	6	5	4	3	2	1	0
B0CSL	Bit symbol		B0WW2	B0WW1	B0WW0		B0WR2	B0WR1	B0WR0
(0140H)	Read/Write		W				W		
	After reset		0	1	0		0	1	0
B0CSH	Bit symbol	B0E	−	−	B0REC	B0OM1	B0OM0	B0BUS1	B0BUS0
(0141H)	Read/Write	W							
	After reset	0	0 (Note1)	0 (Note1)	0	0	0	0	0
MAMR0	Bit symbol	M0V20	M0V19	M0V18	M0V17	M0V16	M0V15	M0V14-V9	M0V8
(0142H)	Read/Write	R/W							
	After reset	1	1	1	1	1	1	1	1
MSAR0	Bit symbol	M0S23	M0S22	M0S21	M0S20	M0S19	M0S18	M0S17	M0S16
(0143H)	Read/Write	R/W							
	After reset	1	1	1	1	1	1	1	1
B1CSL	Bit symbol		B1WW2	B1WW1	B1WW0		B1WR2	B1WR1	B1WR0
(0144H)	Read/Write		W				W		
	After reset		0	1	0		0	1	0
B1CSH	Bit symbol	B1E	−	−	B1REC	B1OM1	B1OM0	B1BUS1	B1BUS0
(0145H)	Read/Write	W							
	After reset	0	0 (Note)	0 (Note)	0	0	0	0	0
MAMR1	Bit symbol	M1V21	M1V20	M1V19	M1V18	M1V17	M1V16	M1V15-V9	M1V8
(0146H)	Read/Write	R/W							
	After reset	1	1	1	1	1	1	1	1
MSAR1	Bit symbol	M1S23	M1S22	M1S21	M1S20	M1S19	M1S18	M1S17	M1S16
(0147H)	Read/Write	R/W							
	After reset	1	1	1	1	1	1	1	1
B2CSL	Bit symbol		B2WW2	B2WW1	B2WW0		B2WR2	B2WR1	B2WR0
(0148H)	Read/Write		W				W		
	After reset		0	1	0		0	1	0
B2CSH	Bit symbol	B2E	B2M	−	B2REC	B2OM1	B2OM0	B2BUS1	B2BUS0
(0149H)	Read/Write	W							
	After reset	1	0	0 (Note1)	0	0	0	Note3	Note3
MAMR2	Bit symbol	M2V22	M2V21	M2V20	M2V19	M2V18	M2V17	M2V16	M2V15
(014AH)	Read/Write	R/W							
	After reset	1	1	1	1	1	1	1	1
MSAR2	Bit symbol	M2S23	M2S22	M2S21	M2S20	M2S19	M2S18	M2S17	M2S16
(014BH)	Read/Write	R/W							
	After reset	1	1	1	1	1	1	1	1
B3CSL	Bit symbol		B3WW2	B3WW1	B3WW0		B3WR2	B3WR1	B3WR0
(014CH)	Read/Write		W				W		
	After reset		0	1	0		0	1	0
B3CSH	Bit symbol	B3E	−	−	B3REC	B3OM1	B3OM0	B3BUS1	B3BUS0
(014DH)	Read/Write	W							
	After reset	0	0 (Note)	0 (Note)	0	0	0	0	0
MAMR3	Bit symbol	M3V22	M3V21	M3V20	M3V19	M3V18	M3V17	M3V16	M3V15
(014EH)	Read/Write	R/W							
	After reset	1	1	1	1	1	1	1	1
MSAR3	Bit symbol	M3S23	M3S22	M3S21	M3S20	M3S19	M3S18	M3S17	M3S16
(014FH)	Read/Write	R/W							
	After reset	1	1	1	1	1	1	1	1

Table 3.6.2 Control Register (1/2)

		7	6	5	4	3	2	1	0
BEXCSH	Bit symbol				BEXREC	BEXOM1	BEXOM0	BEXBUS1	BEXBUS0
(0159H)	Read/Write				W				
	After reset				0	0	0	0	0
BEXCSL	Bit symbol		BEXWW2	BEXWW1	BEXWW0		BEXWR2	BEXWR1	BEXWR0
(0158H)	Read/Write		W				W		
	After reset		0	1	0		0	1	0
PMEMCR	Bit symbol				OPGE	OPWR1	OPWR0	PR1	PR0
(0166H)	Read/Write				R/W				
	After reset				0	0	0	1	0

Note 1: Always write “0”.

Note 2: Read-modify-write instruction is prohibited for BnCSL, BnCSH registers (n=0 to 3, EX).

Note3: Since after reset release becomes undefined, please be sure to set up before accessing the block address space 2.

3.6.6 Cautions

(1) The notes of the timing between $\overline{\text{CS}}$ and $\overline{\text{RD}}$.

If the parasitic capacitance of the read signal (Output enable signal) is greater than that of the chip select signal, it is possible that an unintended read cycle occurs due to a delay in the read signal. Such an unintended read cycle may cause a trouble as in the case of (a) in Figure 3.6.2.

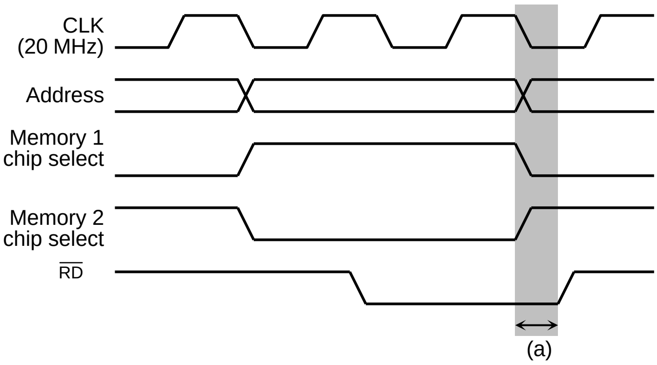


Figure 3.6.2 Read Signal Delay Read Cycle

Example: When using an externally connected flash EEPROM which users JEDEC standard commands, note that the toggle bit may not be read out correctly. If the read signal in the cycle immediately preceding the access to the flash EEPROM does not go “high” in time, as shown in Figure 3.6.3 an unintended read cycle like the one shown in (b) may occur.

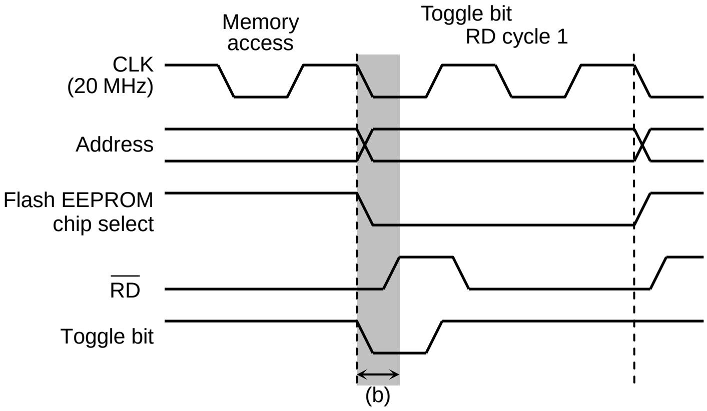


Figure 3.6.3 Flash EEPROM Toggle Bit Read Cycle

When the toggle bit reverse with this unexpected read cycle, TMP92FD23A always reads same value of the toggle bit, and cannot read the toggle bit correctly. To avoid this phenomenon, the data polling control recommended.

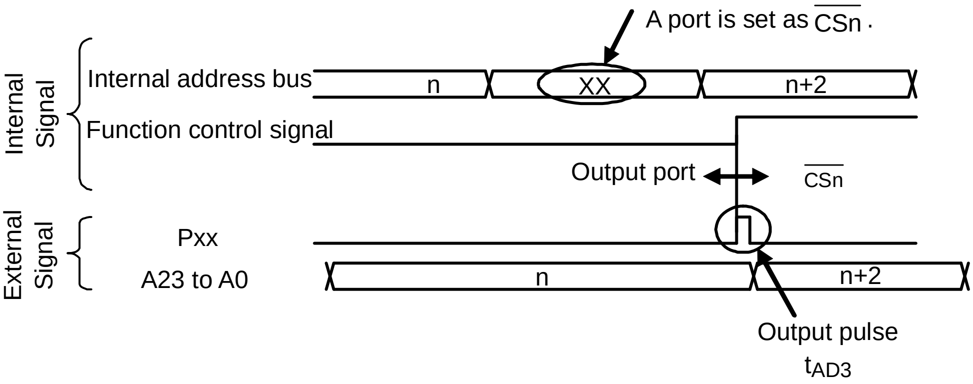
(2) The cautions at the time of the functional change of a $\overline{\text{CSn}}$.

A chip select signal output has the case of a combination terminal with a general-purpose port function. In this case, an output latch register and a function control register are initialized by the reset action, and an object terminal is initialized by the port output ("1" or "0") by it.

Functional change

Although an object terminal is changed from a port to a chip select signal output by setting up a function control register (PnFC register), the short pulse for several ns may be outputted to the changing timing. Although it does not become especially a problem when using the usual memory, it may become a problem when using a special memory.

* XX is a function register address.(When an output port is initialized by "0")

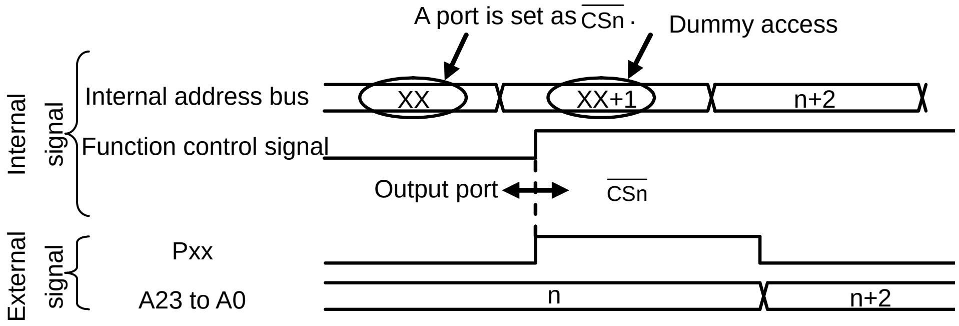


The measure by software

The countermeasures in S/W for avoiding this phenomenon are explained.

Since CS signal decodes the address of the access area and is generated, an unnecessary pulse is outputted by access to the object CS area immediately after setting it as a CSn function. Then, if internal area is accessed also immediately after setting a port as CS function, an unnecessary pulse will not output.

1. Prohibition of use of an NMI function
2. The ban on interruption under functional change (DI command)
3. A dummy command is added in order to carry out continuous internal access.
4. (Access to a functional change register is corresponded by 16-bit command. (LDW command))



3.7 8-Bit Timers (TMRA)

The TMP92FD23A features 6 built-in 8-bit timers.
These timers are paired into three modules: TMRA01, TMRA23 and TMRA45. Each module consists of two channels and can operate in any of the following four operating modes.

- 8-bit interval timer mode
- 16-bit interval timer mode
- 8-bit programmable square wave pulse generation output mode
  (PPG: Variable duty cycle with variable period)
- 8-bit pulse width modulation output mode
  (PWM: Variable duty cycle with constant period)

Figure 3.7.1 to Figure 3.7.3 show block diagrams for TMRA01, TMRA23 and TMRA45.
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 controls SFR (special function registers).
Each of the three modules (TMRA01, TMRA23 and TMRA45) can be operated independently. All modules operate in the same manner; hence only the operation of TMRA01 is explained here.

Table 3.7.1 Registers and Pins for Each Module

Module Specification		TMRA01	TMRA23	TMRA45
External pin	Input pin for external clock	TA0IN (Shared with PC0)	None	None
	Output pin for timer flip-flop	TA1OUT (Shared with P80)	TA3OUT (Shared with P81)	TA5OUT (Shared with P83)
SFR (Address)	Timer RUN register	TA01RUN (1100H)	TA23RUN (1108H)	TA45RUN (1110H)
	Timer register	TA0REG (1102H) TA1REG (1103H)	TA2REG (110AH) TA3REG (110BH)	TA4REG (1112H) TA5REG (1113H)
	Timer mode register	TA01MOD(1104H)	TA23MOD(110CH)	TA45MOD(1114H)
	Timer flip-flop control register	TA1FFCR(1105H)	TA3FFCR(110DH)	TA5FFCR(1115H)

3.7.1 Block Diagrams

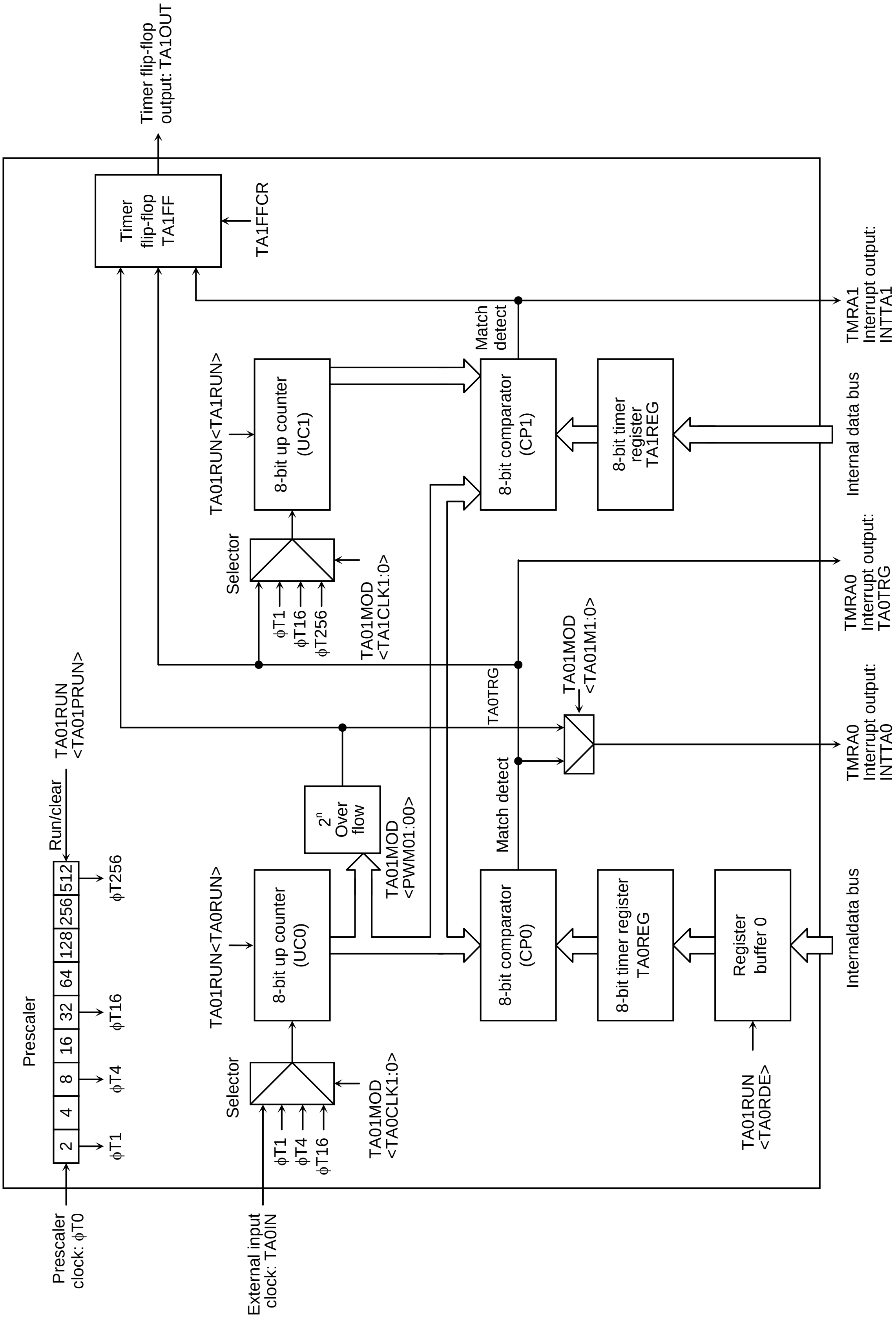


Figure 3.7.1 TMRA01 Block Diagram

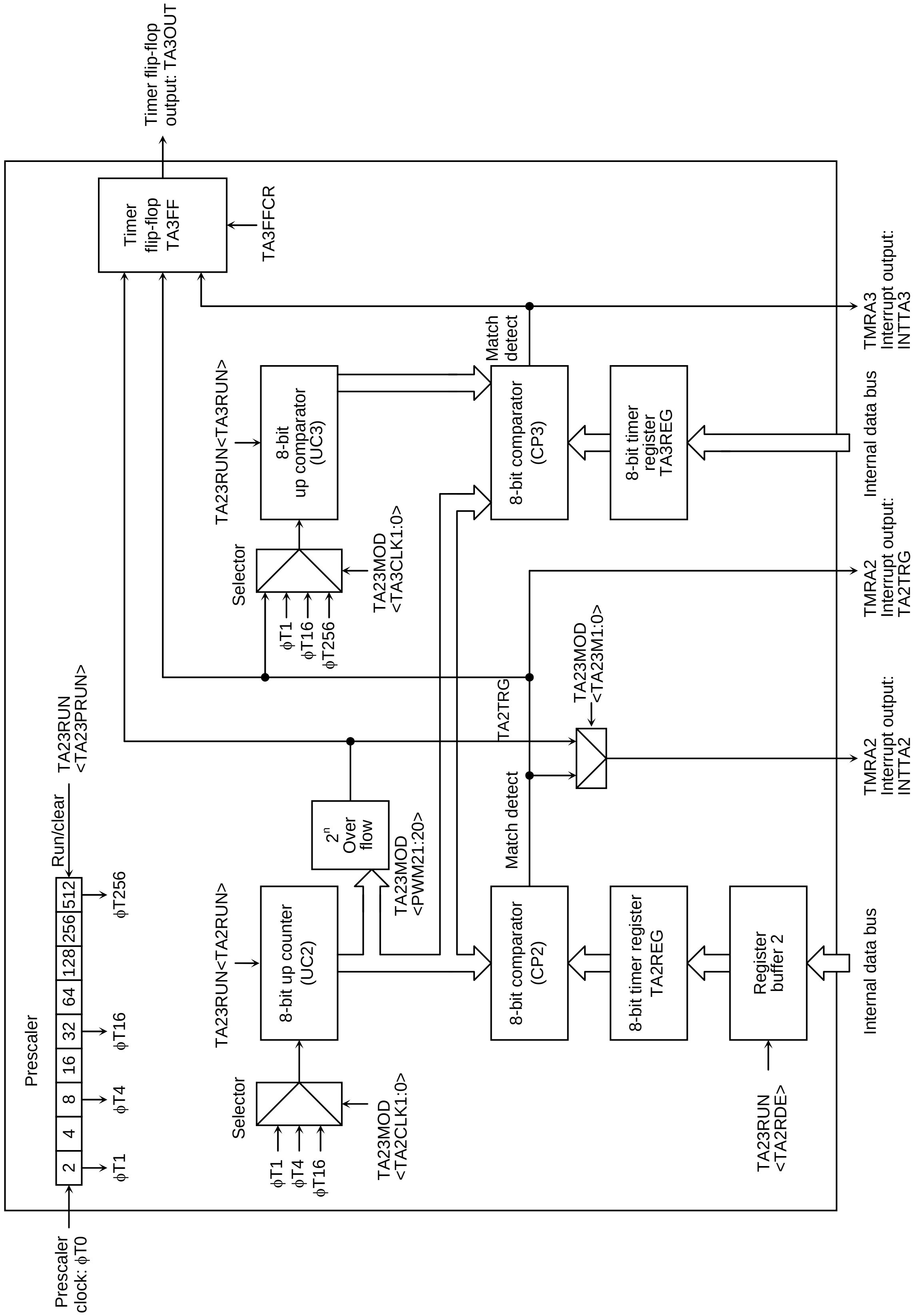


Figure 3.7.2 TMRA23 Block Diagram

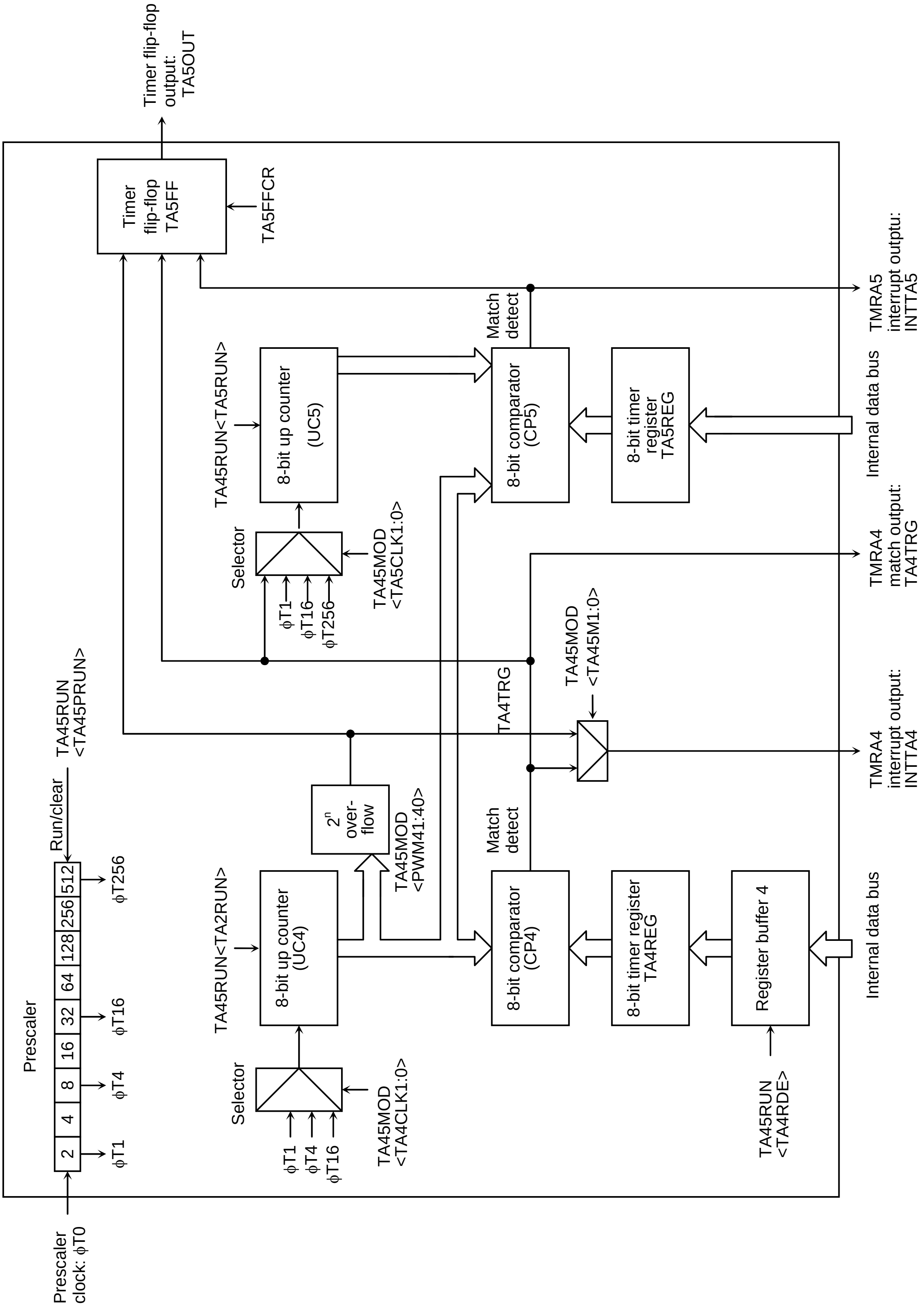


Figure 3.7.3 TMRA45 Block Diagram

3.7.2 Operation of Each Circuit

(1) Prescalers

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

The clock ϕT0 is divided by 4 the CPU clock f_{FPH} and input to this prescaler.

The prescaler's operation can be controlled using TA01RUN <TA0PRUN> in the timer control register. Setting <TA0PRUN> to "1" starts the count; setting <TA0PRUN> 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

Clock Value SYSCR1 <GEAR2:0>	System clock SYSCR1 <SYSCK>	–	Timer counter input clock TMRA prescaler TAxMOD<TAxCLK1:0>			
			ϕT1(1/2)	ϕT4(1/8)	ϕT16(1/32)	ϕT256(1/512)
–	1 (fs)	1/4	fs/8	fs/32	fs/128	fs/2048
000 (1/1)	0 (fc)		fc/8	fc/32	fc/128	fc/2048
001 (1/2)			fc/16	fc/64	fc/256	fc/4096
010 (1/4)			fc/32	fc/128	fc/512	fc/8192
011 (1/8)			fc/64	fc/256	fc/1024	fc/16384
100 (1/16)			fc/128	fc/512	fc/2048	fc/32768

(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 ϕT1, ϕT4 or ϕT16. The clock setting is specified by the value set in TA01MOD<TA0CLK1:0>.

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 ϕT1, ϕT16 or ϕT256, 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 2^n 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 0, set <TA0RDE> to "1", and write the following data to the register buffer. Figure 3.7.4 show the configuration of TA0REG.

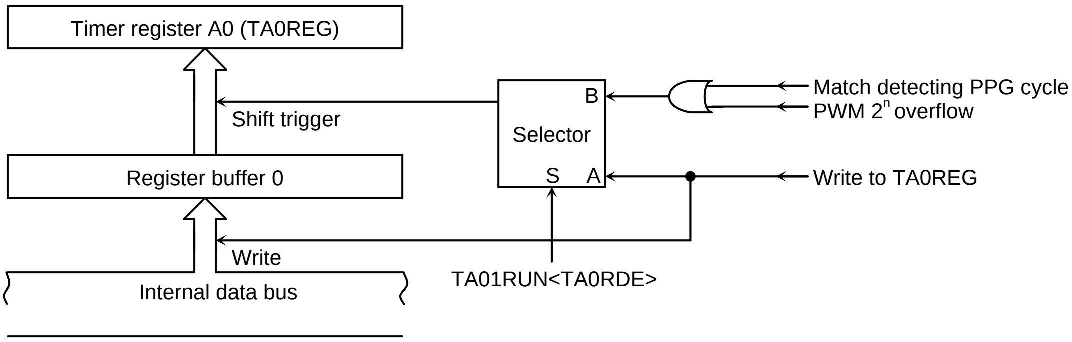


Figure 3.7.4 Configuration of TA0REG

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: 001102H	TA1REG: 001103H
TA2REG: 00110AH	TA3REG: 00110BH
TA4REG: 001112H	TA5REG: 001113H

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 detects 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-flops control register a reset clears the value of TA1FF to “0”. Writing “01” or “10” to TA1FFCR<TA1FFC1:0> sets 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 P80).

When this pin is used as the timer output, the timer flip-flop should be set beforehand using the port 8 function register P8CR and P8FC.

3.7.3 SFR

TMRA01 Run Register

		7	6	5	4	3	2	1	0
TA01RUN	Bit symbol	TA0RDE				I2TA01	TA01PRUN	TA1RUN	TA0RUN
(1100H)	Read/Write	R/W				R/W			
	Reset State	0				0	0	0	0
	Function	Double buffer 0: Disable 1: Enable				IDLE2 0: Stop 1: Operate	TMRA01 prescaler	UC1	UC0
							0: Stop and clear 1: Run (Count up)		

TA0REG double buffer control

0	Disable
1	Enable

Timer run/stop control

0	Stop and clear
1	Run (Count up)

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	Bit symbol	TA2RDE				I2TA23	TA23PRUN	TA3RUN	TA2RUN
(1108H)	Read/Write	R/W				R/W			
	Reset State	0				0	0	0	0
	Function	Double buffer 0: Disable 1: Enable				IDLE2 0: Stop 1: Operate	TMRA23 prescaler	UC3	UC2
							0: Stop and clear 1: Run (Count up)		

TA2REG double buffer control

0	Disable
1	Enable

Timer run/stop control

0	Stop and clear
1	Run (Count up)

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

Figure 3.7.5 Register for TMRA

TMRA45 Run Register

TA45RUN (1110H)		7	6	5	4	3	2	1	0
	Bit symbol	TA4RDE				I2TA45	TA45PRUN	TA5RUN	TA4RUN
	Read/Write	R/W				R/W			
	Reset State	0				0	0	0	0
	Function	Double buffer 0: Disable 1: Enable				IDLE4 0: Stop 1: Operate	TMRA45 prescaler	UC5	UC4
							0: Stop and clear 1: Run (Count up)		

TA4REG double buffer control

0	Disable
1	Enable

Timer run/stop control

0	Stop and clear
1	Run (Count up)

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

Figure 3.7.6 Register for TMRA

TMRA01 Mode Register

		7	6	5	4	3	2	1	0
TA01MOD (1104H)	Bit symbol	TA01M1	TA01M0	PWM01	PWM00	TA1CLK1	TA1CLK0	TA0CLK1	TA0CLK0
	Read/Write	R/W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Operation mode 00: 8-bit timer mode 01: 16-bit timer mode 10: 8-bit PPG mode 11: 8-bit PWM mode		PWM cycle 00: Reserved 01: 2^6 10: 2^7 11: 2^8		Source clock for TMRA1 00: TA0TRG 01: ϕT1 10: ϕT16 11: ϕT256		Source clock for TMRA0 00: TA0IN pin input (Note) 01: ϕT1 10: ϕT4 11: ϕT16	

TMRA0 input clock

<TA0CLK1:0>	00	TA0IN (External input)
	01	ϕT1
	10	ϕT4
	11	ϕT16

TMRA1 input clock

		TA01MOD<TA01M1:0>≠01	TA01MOD<TA01M1:0>=01
<TA1CLK1:0>	00	Matching output for TMRA0	Overflow output from TMRA0 (16-bit timer mode)
	01	ϕT1	
	10	ϕT16	
	11	ϕT256	

PWM cycle selection

<PWM01:00>	00	Reserved
	01	2^6 ×Clock source
	10	2^7 ×Clock source
	11	2^8 × Clock source

TMRA01 operation mode selection

<TA01MA1:0>	00	8-bit timer × 2ch
	01	16-bit timer
	10	8-bit PPG
	11	8-bit PWM (TMRA0), 8-bit timer (TMRA1)

Note: When set TA0IN, set TA01MOD after set port C0.

Figure 3.7.7 Register for TMRA

TMRA23 Mode Register

		7	6	5	4	3	2	1	0
TA23MOD	Bit symbol	TA23M1	TA23M0	PWM21	PWM20	TA3CLK1	TA3CLK0	TA2CLK1	TA2CLK0
(110CH)	Read/Write	R/W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Operation mode 00: 8-bit timer mode 01: 16-bit timer mode 10: 8-bit PPG mode 11: 8-bit PWM mode		PWM cycle 00: Reserved 01: 2^6 10: 2^7 11: 2^8		Source clock for TMRA3 00: TA2TRG 01: ϕT1 10: ϕT16 11: ϕT256		Source clock for TMRA2 00: Reserved 01: ϕT1 10: ϕT4 11: ϕT16	

TMRA2 input clock

<TA2CLK1:0>	00	Reserved
	01	ϕT1
	10	ϕT4
	11	ϕT16

TMRA3 input clock

		TA23MOD<TA23M1:0>≠01	TA23MOD<TA23M1:0>=01
<TA3CLK1:0>	00	Matching output for TMRA2	Overflow output from TMRA2 (16-bit timer mode)
	01	ϕT1	
	10	ϕT16	
	11	ϕT256	

PWM cycle selection

<PWM23:00>	00	Reserved
	01	2^6 × Clock source
	10	2^7 × Clock source
	11	2^8 × Clock source

TMRA23 operation mode selection

<TA23MA1:0>	00	8-bit timer × 2ch
	01	16-bit timer
	10	8-bit PPG
	11	8-bit PWM (TMRA2), 8-bit timer (TMRA3)

Figure 3.7.8 Register for TMRA

TMRA45 Mode Register

		7	6	5	4	3	2	1	0
TA45MOD	Bit symbol	TA45M1	TA45M0	PWM41	PWM40	TA5CLK1	TA5CLK0	TA4CLK1	TA4CLK0
(1114H)	Read/Write	R/W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Operation mode 00: 8-bit timer mode 01: 16-bit timer mode 10: 8-bit PPG mode 11: 8-bit PWM mode		PWM cycle 00: Reserved 01: 2^6 10: 2^7 11: 2^8		Source clock for TMRA5 00: TA4TRG 01: ϕT1 10: ϕT16 11: ϕT256		Source clock for TMRA4 00: Reserved 01: ϕT1 10: ϕT4 11: ϕT16	

TMRA4 input clock

<TA4CLK1:0>	00	Reserved
	01	ϕT1
	10	ϕT4
	11	ϕT16

TMRA5 input clock

		TA45MOD<TA45M1:0>≠01	TA45MOD<TA45M1:0>=01
<TA5CLK1:0>	00	Matching output for TMRA4	Overflow output from TMRA4 (16-bit timer mode)
	01	ϕT1	
	10	ϕT16	
	11	ϕT256	

PWM cycle selection

<PWM45:00>	00	Reserved
	01	2^6 ×Clock source
	10	2^7 × Clock source
	11	2^8 ×Clock source

TMRA45 operation mode selection

<TA45MA1:0>	00	8-bit timer × 2ch
	01	16-bit timer
	10	8-bit PPG
	11	8-bit PWM (TMRA4), 8-bit timer (TMRA5)

Figure 3.7.9 Register for TMRA

TMRA1 Flip-Flop Control Register

		7	6	5	4	3	2	1	0
TA1FFCR (1105H)	Bit symbol					TA1FFC1	TA1FFC0	TA1FFIE	TA1FFIS
	Read/Write					R/W			
	Reset State					1	1	0	0
Read modify write instruction is prohibited.	Function					00: Invert TA1FF 01: Set TA1FF 10: Clear TA1FF 11: Don't care		TA1FF control for inversion 0: Disable 1: Enable	TA1FF inversion select 0: TMRA0 1: TMRA1

Inversion signal for timer flip-flop 1 (TA1FF)
(Don't care except in 8-bit timer mode)

TA1FFIS	0	Inversion by TMRA0
	1	Inversion by TMRA1

Inversion of TA1FF

TA1FFIE	0	Disabled
	1	Enabled

Control of TA1FF

<TA1FFC1:0>	00	Inverts the value of TA1FF (Software inversion)
	01	Sets TA1FF to “1”
	10	Clears TA1FF to “0”
	11	Don't care

Note: The values of bits4 to 6 of TA1FFCR are undefined when read.

Figure 3.7.10 Register for TMRA

TMRA3 Flip-Flop Control Register

		7	6	5	4	3	2	1	0
TA3FFCR	Bit symbol					TA3FFC1	TA3FFC0	TA3FFIE	TA3FFIS
(110DH)	Read/Write					R/W			
	Reset State					1	1	0	0
Read modify write instruction is prohibited.	Function					00: Invert TA3FF 01: Set TA3FF 10: Clear TA3FF 11: Don’t care		TA3FF control for inversion 0: Disable 1: Enable	TA3FF inversion select 0: TMRA2 1: TMRA3

Inversion signal for timer flip-flop 3 (TA3FF)
(Don’t care except in 8-bit timer mode)

TA3FFIS	0	Inversion by TMRA2
	1	Inversion by TMRA3

Inversion of TA3FF

TA3FFIE	0	Disabled
	1	Enabled

Control of TA3FF

<TA3FFC1:0>	00	Inverts the value of TA3FF (Software inversion)
	01	Sets TA3FF to “1”
	10	Clears TA3FF to “0”
	11	Don’t care

Note: The values of bits4 to 6 of TA3FFCR are undefined when read.

Figure 3.7.11 Register for TMRA

TMRA5 Flip-Flop Control Register

		7	6	5	4	3	2	1	0
TA5FFCR (1115H)	Bit symbol					TA5FFC1	TA5FFC0	TA5FFIE	TA5FFIS
	Read/Write					R/W			
	Reset State					1	1	0	0
Read modify write instruction is prohibited.	Function					00: Invert TA5FF 01: Set TA5FF 10: Clear TA5FF 11: Don’t care		TA5FF control for inversion 0: Disable 1: Enable	TA5FF inversion select 0: TMRA4 1: TMRA5

Inversion signal for timer flip-flop 5 (TA5FF)
(Don’t care except in 8-bit timer mode)

TA5FFIS	0	Inversion by TMRA4
	1	Inversion by TMRA5

Inversion of TA5FF

TA5FFIE	0	Disabled
	1	Enabled

Control of TA5FF

<TA5FFC1:0>	00	Inverts the value of TA5FF (Software inversion)
	01	Sets TA5FF to “1”
	10	Clears TA5FF to “0”
	11	Don’t care

Note: The values of bits4 to 6 of TA5FFCR are undefined when read.

Figure 3.7.12 Register for TMRA

TMRA Register

		7	6	5	4	3	2	1	0
TA0REG	Bit symbol	–							
(1102H)	Read/Write	W							
	Reset State	Undefined							
TA1REG	Bit symbol	–							
(1103H)	Read/Write	W							
	Reset State	Undefined							
TA2REG	Bit symbol	–							
(110AH)	Read/Write	W							
	Reset State	Undefined							
TA3REG	Bit symbol	–							
(110BH)	Read/Write	W							
	Reset State	Undefined							
TA4REG	Bit symbol	–							
(1112H)	Read/Write	W							
	Reset State	Undefined							
TA5REG	Bit symbol	–							
(1113H)	Read/Write	W							
	Reset State	Undefined							

Note: Read-modify-write instruction is prohibited for above registers.

Figure 3.7.13 Register for TMRA

3.7.4 Operation in Each Mode

(1) 8-bit timer mode

Both TMRA0 and TMRA1 can be used independently as 8-bit interval timers. When set function and count data, TMRA0 and TMRA1 should be stopped.

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 40 μs at f_C = 40 MHz, set each register as follows:

*Clock state: Clock gear : 1/1(fc)

		MSB							LSB	
		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 ϕT1 (= (8/fc)s @f_C = 40 MHz) as the input clock.
TA1REG	←	1	1	0	0	1	0	0	0	Set 40 μs ÷ ϕT1 = 200 = C8H to TAREG.
INTETA01	←	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 3.7.3.

Table 3.7.3 Selecting Interrupt Interval and the Input Clock Using 8-Bit Timer

Input Clock	Interrupt Interval (at f_C = 40 MHz)	Resolution
ϕT1 (8/fC)	0.2 μs to 51.2 μs	0.2 μs
ϕT4 (32/fC)	0.8 μs to 204.8 μs	0.8 μs
ϕT16 (128/fC)	3.2 μs to 819.2μs	3.2 μs
ϕT256 (2048/fC)	51.2 μs to 13.11 ms	51.2 μs

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

TMRA0: Uses TMRA0 input (TA0IN) and can be selected from ϕT1, ϕT4 or ϕT16

TMRA1: Match output of TMRA0 (TA0TRG) and can be selected from ϕT1, ϕT16, ϕT256

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 1.2-μs square wave pulse from the TA1OUT pin at f_C = 40 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: Clock gear : 1/1(fc)

		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 ϕT1 (= (8/fc)s @ f_C = 40 MHz) as the input clock.
TA1REG	←	0	0	0	0	0	0	1	1	Set the timer register to 1.2 μs ÷ ϕT1 ÷ 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.
P8FC	←	X	X	X	X	–	X	–	1	Set P80 to function as the TA1OUT pin.
TA01RUN	←	–	X	X	X	–	1	1	–	Start TMRA1 counting.

X: Don’t care, –: No change

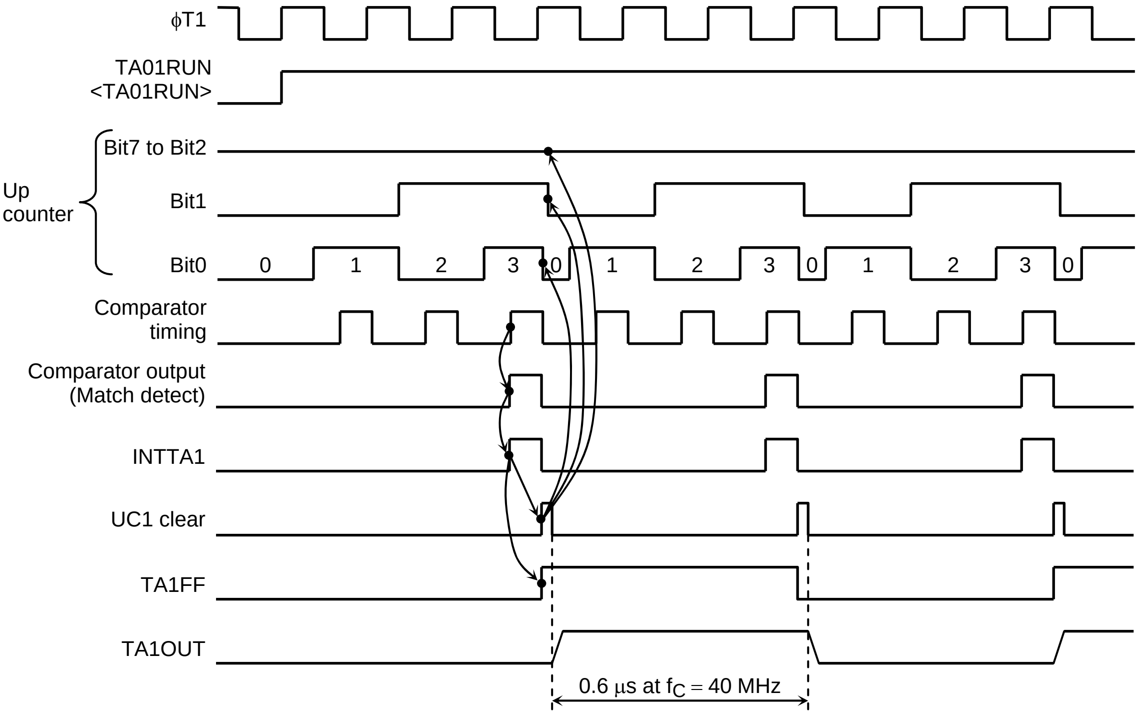


Figure 3.7.14 Square Wave Output Timing Chart (50 % Duty)

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.

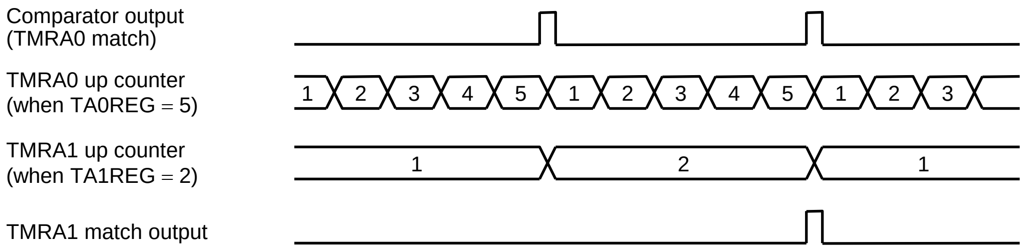


Figure 3.7.15 TMRA1 Count Up on Signal from TMRA0

(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<TA01M1:0> 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<TA01CLK1:0>. Table 3.7.2 shows the relationship between the timer (interrupt) cycle and the input clock selection.

To set the timer interrupt interval, set the lower eight bits in timer register TA0REG and the upper eight bits in TA1REG. Be sure to set TA0REG first (as entering data in TA0REG temporarily disables the compare, while entering data in TA1REG starts the compare).

Setting example: To generate an INTTA1 interrupt every 0.2 s at f_C = 40 MHz, set the timer registers TA0REG and TA1REG as follows:
*Clock state: Clock gear : 1/1(fc)

If ϕT16 (=(128/fc)s @f_C = 40 MHz) is used as the input clock for counting, set the following value in the registers:

0.2 s ÷ (128/fc)s = 62500 = 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, though 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 comparator TMRA0 and TMRA1, the up counters 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 TA1FF is inverted.

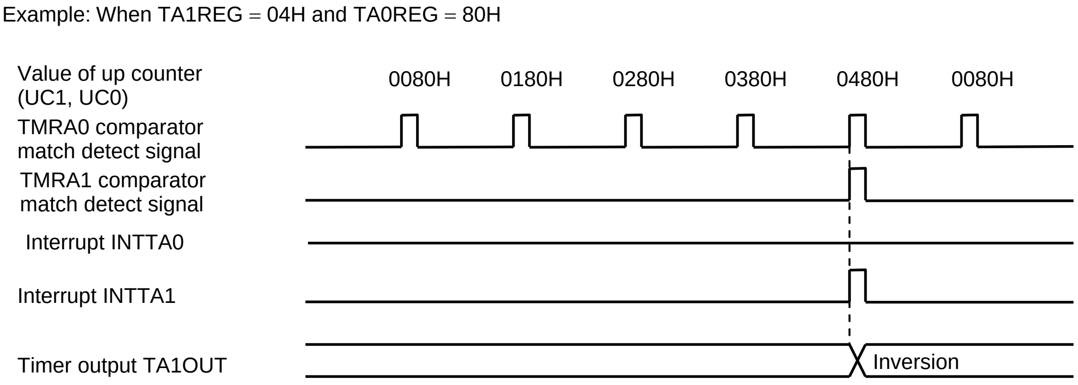

Example: When TA1REG = 04H and TA0REG = 80H
Value of up counter (UC1, UC0)
0080H
0180H
0280H
0380H
0480H
0080H
TMRA0 comparator match detect signal
TMRA1 comparator match detect signal
Interrupt INTTA0
Interrupt INTTA1
Timer output TA1OUT
Inversion

Figure 3.7.16 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 P80).

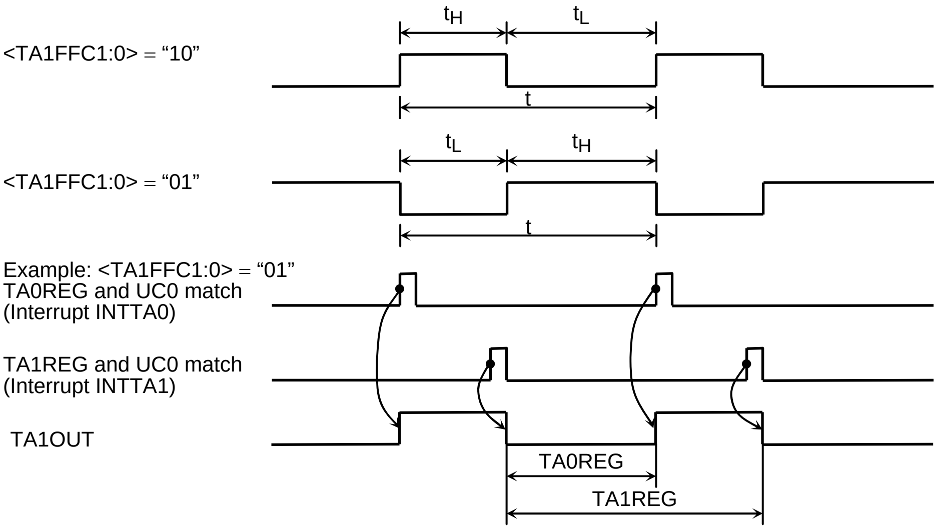

<TA1FFC1:0> = "10"
tH
tL
t
<TA1FFC1:0> = "01"
tL
tH
t
Example: <TA1FFC1:0> = "01"
TA0REG and UC0 match (Interrupt INTTA0)
TA1REG and UC0 match (Interrupt INTTA1)
TA1OUT
TA0REG
TA1REG

Figure 3.7.17 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 3.7.18 shows a block diagram representing this mode.

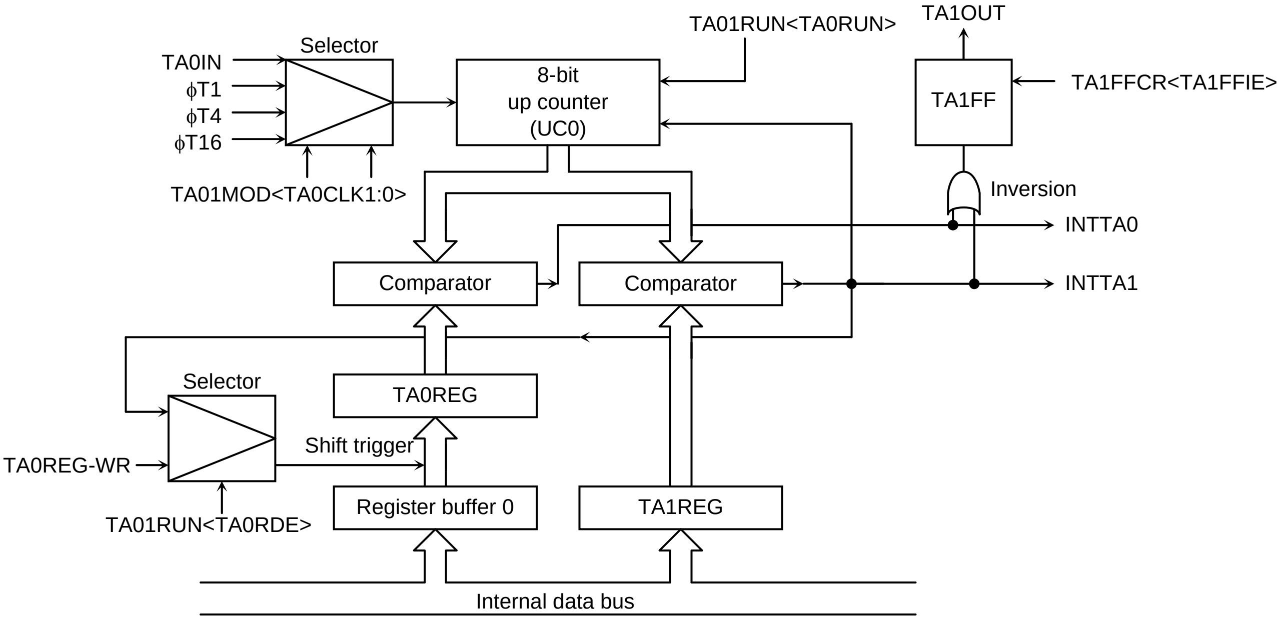


Figure 3.7.18 Block Diagram of 8-Bit PPG Output Mode

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).

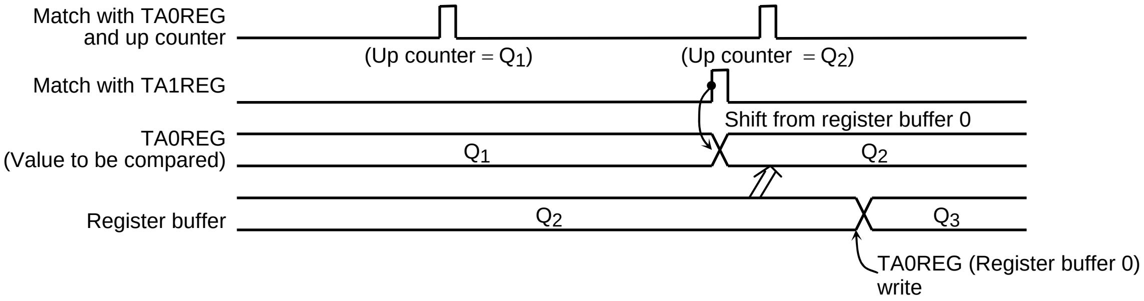
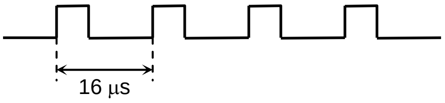


Figure 3.7.19 Operation of Register Buffer 0

Example: To generate 1/4 duty 62.5 kHz pulses (at f_C = 40 MHz)



*Clock state: Clock gear : 1/1(fc)

Calculate the value that should be set in the timer register.
To obtain a frequency of 62.5 kHz, the pulse cycle t should be:

t = 1/62.5 kHz = 16 μs

ϕT1 (= (8/fc)s @f_C = 40 MHz);

16 μs ÷ (8/fc)s = 80

Therefore set TA1REG = 80 = 50H

The duty is to be set to 1/4: t × 1/4 = 16 μs × 1/4 = 4 μs

4 μs ÷ (8/fc)s = 20

Therefore, set TA0REG = 20 = 14H

		7	6	5	4	3	2	1	0	
TA01RUN	←	0	X	X	X	–	0	0	0	Stop TMRA0 and TMRA1 and clear it to “0”.
TA01MOD	←	1	0	X	X	X	X	0	1	Set the 8-bit PPG mode, and select ϕT1 as input clock.
TA0REG	←	0	0	0	1	0	1	0	0	Write 14H.
TA1REG	←	0	1	0	1	0	0	0	0	Write 50H.
TA1FFCR	←	X	X	X	X	0	1	1	X	Set TA1FF, enabling both inversion and the double buffer. 10 generate a negative logic pulse.
P8FC2	←	X	X	X	X	–	X	–	1	Set P80 as the TA1OUT pin.
TA01RUN	←	1	X	X	X	–	1	1	1	Start TMRA0 and TMRA1 counting.

X: Don't care, –: No change

(4) 8-bit PWM 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 P80). TMRA1 can also be used as an 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 ≠ 0

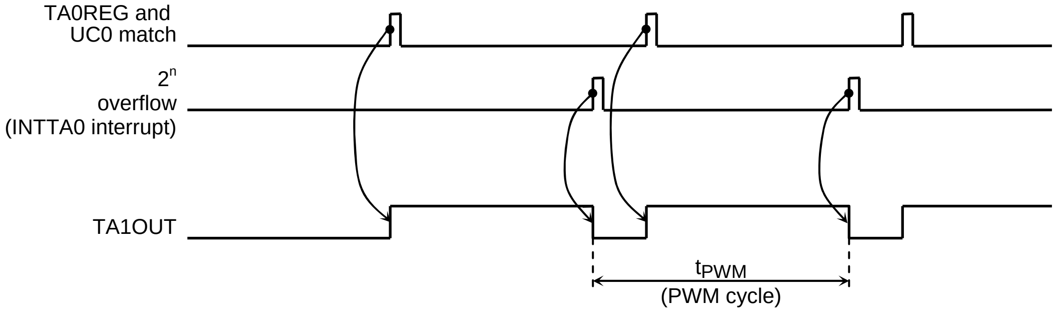


Figure 3.7.20 8-Bit PWM Waveforms

Figure 3.7.21 shows a block diagram representing this mode.

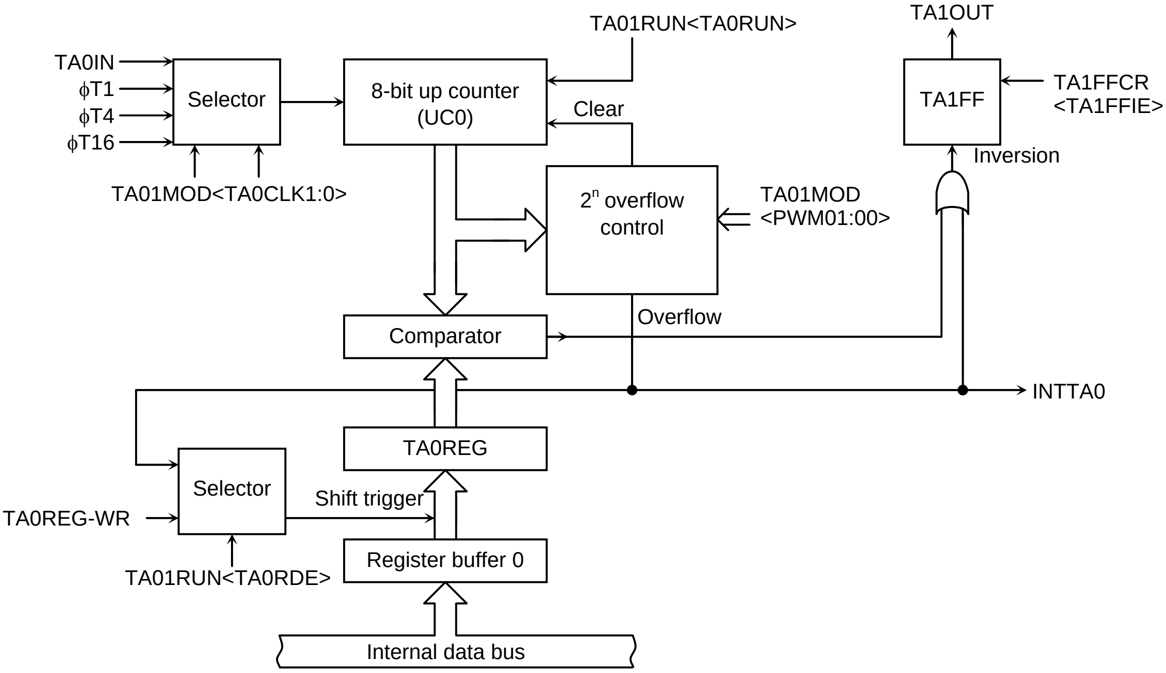


Figure 3.7.21 Block Diagram of 8-Bit PWM Mode

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.

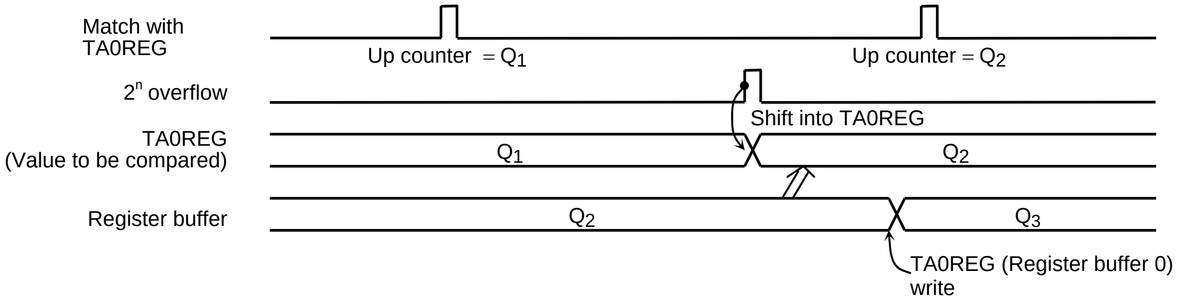
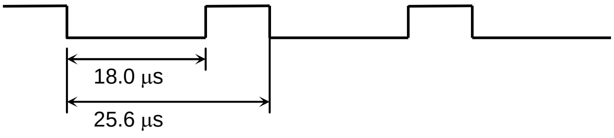


Figure 3.7.22 Register Buffer Operation

Example: To output the following PWM waves on the TA1OUT pin (at f_C = 40 MHz).



*Clock state: Clock gear : 1/1(fc)

To achieve a 25.6-μs PWM cycle by setting ϕT1 (= (8/fc)s @f_C = 40 MHz):
25.6 μs ÷ (8/fc)s = 128 = 2^n
Therefore n should be set to 7.
Since the low level period is 18.0 μs when ϕT1 = (8/fc)s,
set the following value for TREG0:

18.0 μs ÷ (8/fc)s = 90 = 5AH

		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	1	Select 8-bit PWM mode (cycle: 27) and select ϕT1 as the input clock.
TA0REG	←	0	1	0	1	1	0	1	0	Write 5AH.
TA1FFCR	←	X	X	X	X	1	0	1	X	Clear TA1FF to 0, enable the inversion and double buffer.
P8FC2	←	–	–	–	–	–	–	–	1	Set P80 as the TA1OUT pin.
TA01RUN	←	1	X	X	X	–	1	–	1	Start TMRA0 counting.

X: Don't care, –: No change

Table 3.7.4 PWM Cycle

Clock gear value SYSCR1 <GEAR2:0>	System clock SYSCR0 <SYSCK>	–	PWM cycle TAxxMOD<PWMx1:0>								
			2^6 (x64)			2^7(x128)			2^8(x256)		
			TAxxMOD<TAxCLK1:0>			TAxxMOD<TAxCLK1:0>			TAxxMOD<TAxCLK1:0>		
			φT1(x2)	φT4(x8)	φT16(x32)	φT1(x2)	φT4(x8)	φT16(x32)	φT1(x2)	φT4(x8)	φT16(x32)
–	1(fs)	×4	512/fs	2048/fs	8192/fs	1024/fs	4096/fs	16384/fs	2048/fs	8192/fs	32768/fs
000(x1)	0(fc)		512/fc	2048/fc	8192/fc	1024/fc	4096/fc	16384/fc	2048/fc	8192/fc	32768/fc
001(x2)			1024/fc	4096/fc	16384/fc	2048/fc	8192/fc	32768/fc	4096/fc	16384/fc	65536/fc
010(x4)			2048/fc	8192/fc	32768/fc	4096/fc	16384/fc	65536/fc	8192/fc	32768/fc	131072/fc
011(x8)			4096/fc	16384/fc	65536/fc	8192/fc	32768/fc	131072/fc	16384/fc	65536/fc	262144/fc
100(x16)			8192/fc	32768/fc	131072/fc	16384/fc	65536/fc	262144/fc	32768/fc	131072/fc	524288/fc

(5) Settings for each mode

Table 3.7.5 shows the SFR settings for each mode.

Table 3.7.5 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 × 2 channels	00	–	Lower timer match, φT1, φT16, φT256 (00, 01, 10, 11)	External clock, φT1, φT4, φT16 (00, 01, 10, 11)	0: Lower timer output 1: Upper timer output
16-bit timer mode	01	–	–	External clock, φT1, φT4, φT16 (00, 01, 10, 11)	–
8-bit PPG × 1 channel	10	–	–	External clock, φT1, φT4, φT16 (00, 01, 10, 11)	–
8-bit PWM × 1 channel	11	2^6, 2^7, 2^8 (01, 10, 11)	–	External clock, φT1, φT4, φT16 (00, 01, 10, 11)	–
8-bit timer × 1 channel	11	–	φT1, φT16, φT256 (01, 10, 11)	–	Output disabled

–: Don’t care

3.8 16-Bit Timer/Event Counters (TMRB0)

The TMP92FD23A incorporates two multifunctional 16-bit timer/event counter (TMRB0 and TMRB1) which has the following operation modes:

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

Can be used following operation modes by capture function.

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

Figure 3.8.1 to Figure 3.8.2 show block diagram of TMRB0 and TMRB1. Timer/event counter 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.

Timer/event counter is controlled by an 11-byte control SFR.Each channel(TMRB0,TMRB1) operate independently.In this section, the explanation describes only for TMRB1 because each channel is identical operation except for the difference as follows;

Table 3.8.1 Pins and SFR of TMRB

Channel Spec		TMRB0	TMRB1
External pin	External clock/ Caputre triggr input pin	None	TB1IN0 (Share with PD1) TB1IN1 (Share with PD2)
	Timer flip-flop output pin	TB0OUT0 (Share with PD0)	TB1OUT0 (Share with PD3) TB1OUT1 (Share with PD4)
SFR (Address)	Timre run register	TB0RUN (1180H)	TB1RUN (1190H)
	Timrer mode register	TB0MOD (1182H)	TB1MOD (1192H)
	Timre flip-flop control register	TB0FFCR (1183H)	TB1FFCR (1193H)
	Timer register	TB0RG0L (1188H)	TB1RG0L (1198H)
		TB0RG0H (1189H)	TB1RG0H (1199H)
		TB0RG1L (118AH)	TB1RG1L (119AH)
		TB0RG1H (118BH)	TB1RG1H (119BH)
	Capture register	TB0CP0L (118CH)	TB1CP0L (119CH)
		TB0CP0H (118DH)	TB1CP0H (119DH)
		TB0CP1L (118EH)	TB1CP1L (119EH)
		TB0CP1H (118FH)	TB1CP1H (119FH)

3.8.1 Block Diagrams

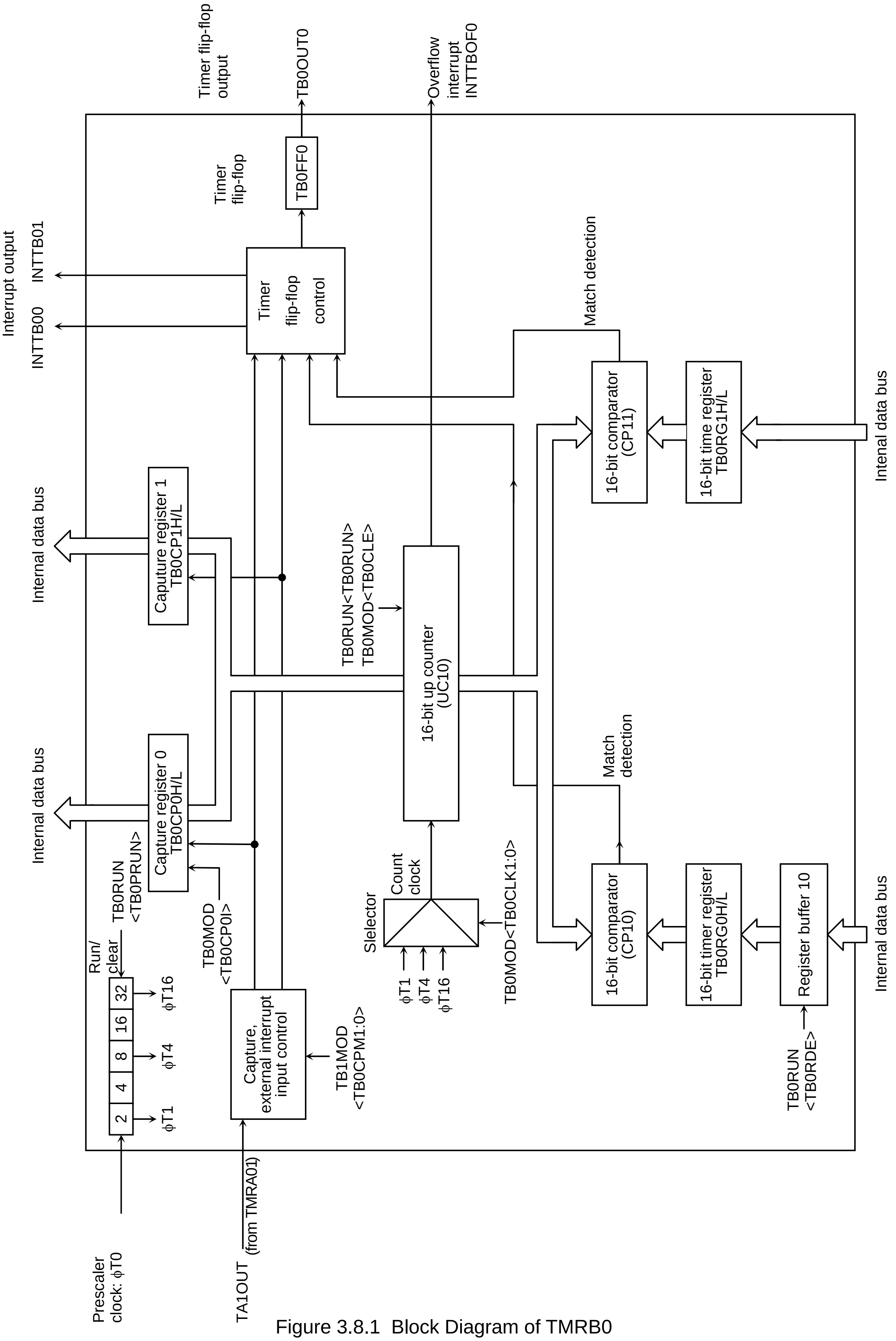


Figure 3.8.1 Block Diagram of TMRB0

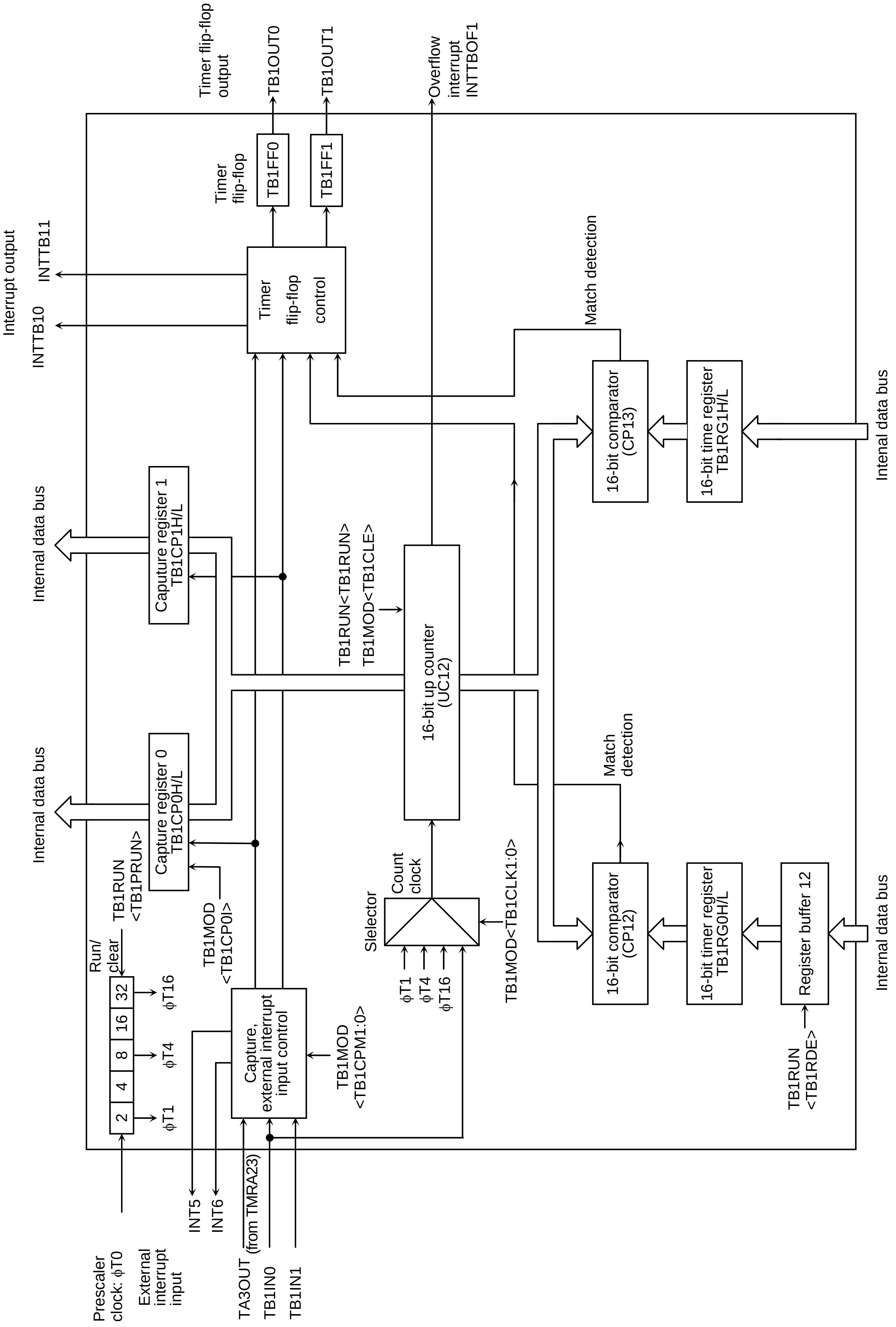


Figure 3.8.2 Block Diagram of TMRB1

3.8.2 Operation of Each Block

(1) Prescaler

The 5-bit prescaler generates the source clock for TMRB1. The prescaler clock (ϕT0) is divided clock (Divided by 4) from selected clock by the register SYSCR1<SYSCK> of clock gear.

This prescaler can be started or stopped using TB1RUN<TB1PRUN>. Counting starts when <TB0PRUN> is set to 1; the prescaler is cleared to 0 and stops operation when <TB0PRUN> is cleared to 0.

Table 3.8.2 Prescaler Clock Resolution

Gear Value SYSCR1 <GEAR2:0>	System clock SYSCR1 <SYSCK>	–	Timer counter input clock TMRB prescaler TBxMOD<TBxCLK1:0>		
			ϕT1(1/2)	ϕT4(1/8)	ϕT16(1/32)
–	1 (fs)	1/4	fs/8	fs/32	fs/128
000 (1/1)	0 (fc)		fc/8	fc/32	fc/128
001 (1/2)			fc/16	fc/64	fc/256
010 (1/4)			fc/64	fc/128	fc/512
011 (1/8)			fc/64	fc/256	fc/1024
100 (1/16)			fc/128	fc/512	fc/2048

(2) Up counter (UC12)

UC12 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 ϕT1, ϕT4 and ϕT16 can be selected as the input clock. Counting or stopping and clearing of the counter is controlled by TB1RUN<TB1RUN>. TMRB0 cannot choose an external clock as an input clock (there is no external clock input terminal).

When clearing is enabled, the up counter UC12 will be cleared to 0 each time its value matches the value in the timer register TB1RG1H/L. If clearing is disabled, the counter operates as a free-running counter. Clearing can be enabled or disabled using TB1MOD<TB1CLE>.

A timer overflow interrupt (INTTBOF1) is generated when UC12 overflow occurs.

(3) Timer registers (TB1RG0H/L and TB1RG1H/L)

These 16-bit registers are used to set the interval time. When the value in the up counter UC12 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 is always needed. For example, either 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 TB1RG0 timer register has a double-buffer structure, which is paired with register buffer. The value set in TB1RUN<TB1RDE> determines whether the double-buffer structure is enabled or disabled: it is disabled when <TB1RDE> = 0, and enabled when <TB1RDE> = 1.

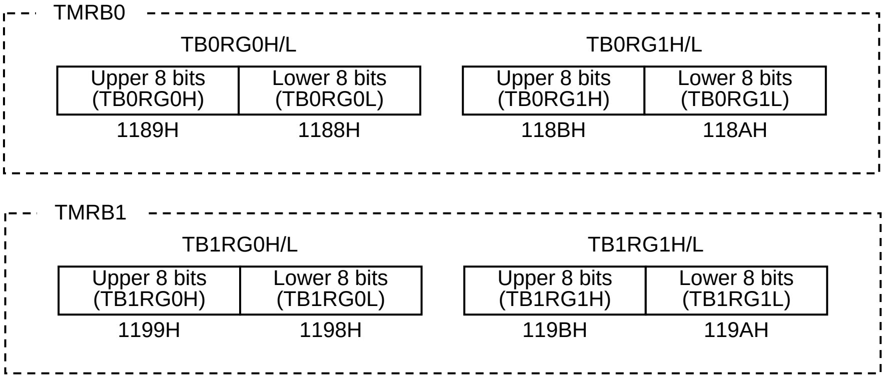
When the double buffer is enabled, data is transferred from the register buffer to the timer register when the values in the up counter (UC12) and the timer register TB1RG1 match.

After a reset, TB1RG0H/L and TB1RG1H/L are undefined. If the 16-bit timer is to be used after a reset, data should be written to it beforehand.

On a reset <TB1RDE> is initialized to 0, disabling the double buffer. To use the double buffer, write data to the timer register, set <TB1RDE> to 1, then write data to the register buffer as shown below.

TB1RG0H/L and the register buffer both have the same memory addresses (1188H and 1189H) allocated to them. If <TB1RDE> = 0, the value is written to both the timer register and the register buffer. If <TB1RDE> = 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 (TB1CP0H/L and TB1CP1H/L)

These 16-bit registers are used to latch the values in the up counters UC12.

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 twice for lower 8 bits and upper 8 bits in order.

The addresses of the capture registers are as follows:

TMRB0

TB0RG0H/L		TB0RG1H/L	
Upper 8 bits (TB0RG0H)	Lower 8 bits (TB0RG0L)	Upper 8 bits (TB0RG1H)	Lower 8 bits (TB0RG1L)
1189H	1188H	118BH	118AH

TMRB1

TB1RG0H/L		TB1RG1H/L	
Upper 8 bits (TB1RG0H)	Lower 8 bits (TB1RG0L)	Upper 8 bits (TB1RG1H)	Lower 8 bits (TB1RG1L)
1199H	1198H	119BH	119AH

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

(5) Capture input control

This circuit controls the timing to latch the value of up counter UC12 into TB1CP0H/L and TB1CP1H/L.

Interrupt timing of capture register and selection edge of external interrupt are set by TB1MOD<TB1CPM1:0>. (TMRB0 does not include the selection edge of external interrupt.)

The value in the up counter can be loaded into a capture register by software. Whenever 0 is programmed to TB1MOD<TB1CP0I>, the current value in the up counter is loaded into capture register TB1CP0H/L. It is necessary to keep the prescaler in run mode (e.g., TB1RUN<TB1PRUN> must be held at a value of 1).

(6) Comparators (CP12, CP13)

CP12 is 16-bit comparators which compare the value in the up counter UC12 with the value set in TB1RG0H/L or TB1RG1H/L respectively, in order to detect a match. If a match is detected, the comparator generates an interrupt (INTTB10 or INTTB11 respectively).

(7) Timer flip-flops (TB1FF0 and TB1FF1)

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 TB1FFCR<TB1C0T1, TB1E1T1 and TB1E0T1>.

After a reset the value of TB1FF0 is undefined. If "00" is programmed to TB1FFCR <TB1FF0C1:0> or <TB1FF1C1:0>, TB1FF0 will be inverted. If "01" is programmed to the capture registers, the value of TB1FF0 will be set to "1". If "10" is programmed to the capture registers, the value of TB1FF0 will be cleared to "0".

The values of TB1FF0 and TB1FF1 can be output via the timer output pins TB1OUT0 (which is shared with PD3), TB1OUT1 (which is shard with PD4). The timer output pin of TMRB0 is one pin (TB0OUT0: which is shard with PD0). Timer output should be specified using the port D function register.

3.8.3 SFRs

TMRB0 Run Register

		7	6	5	4	3	2	1	0
TB0RUN (1180H)	Bit symbol	TB0RDE	–			I2TB0	TB0PRUN		TB0RUN
	Read/Write	R/W				R/W			R/W
	Reset State	0	0			0	0		0
	Function	Double buffer 0: Disable 1: Enable	Always write “0”			IDLE2 0: Stop 1: Operate	TMRB0 prescaler		Up counter (UC10)
							0: Stop and clear 1: Run (Count up)		

Count operation

<TB0PRUN>, <TB0RUN>	0	Stop and clear
	1	Count up

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

TMRB1 Run Register

		7	6	5	4	3	2	1	0
TB1RUN (1190H)	Bit symbol	TB1RDE	–			I2TB1	TB1PRUN		TB1RUN
	Read/Write	R/W				R/W			R/W
	Reset State	0	0			0	0		0
	Function	Double buffer 0: Disable 1: Enable	Always write “0”			IDLE2 0: Stop 1: Operate	TMRB1 prescaler		Up counter (UC12)
							0: Stop and clear 1: Run (Count up)		

Count operation

<TB1PRUN>, <TB1RUN>	0	Stop and clear
	1	Count up

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

Figure 3.8.3 The Registers for TMRB

TMRB0 Mode Register

		7	6	5	4	3	2	1	0
TB0MOD	Bit symbol	–	–	TB0CP0I	TB0CPM1	TB0CPM0	TB0CLE	TB0CLK1	TB0CLK0
(1182H)	Read/Write	R/W		W*	R/W				
	Reset State	0	0	1	0	0	0	0	0
Read-modify-write instruction is prohibited.	Function	Always write “0”		Software capture control 0: Software capture 1: Undefined	Capture timing 00: Disable 01: Reserved 10: Reserved 11: TA1OUT↑ TA1OUT↓		Up counter control 0: Disable 1: Enable	TMRB0 source clock 00: Reserved 01: ϕT1 10: ϕT4 11: ϕT16	

TMRB0 source clock

<TB0CLK1:0>	00	Reserved
	01	ϕT1
	10	ϕT4
	11	ϕT16

Control clearing for up counter (UC10)

<TB0CLE>	0	Disable
	1	Enable clearing by match with TB0RG1H/L

Capture timing

		Capture control
<TB0CPM1:0>	00	Disable
	01	Reserved
	10	Reserved
	11	Capture to TB0CP0H/L at rising edge of TA1OUT Capture to TB0CP1H/L at falling edge of TA1OUT

Software capture

<TB0CP0I>	0	The value of up counter is captured to TB0CP0H/L
	1	Undefined

Figure 3.8.4 The Registers for TMRB0

TMRB0 Mode Register

		7	6	5	4	3	2	1	0
TB1MOD (1192H)	Bit symbol	TB1CT1	TB1ET1	TB1CP0I	TB1CPM1	TB1CPM0	TB1CLE	TB1CLK1	TB1CLK0
	Read/Write	R/W		W*	R/W				
	Reset State	0	0	1	0	0	0	0	0
Read-modify -write instruction is prohibited.	Function	TB1FF1 Inversion trigger 0: Trigger disable 1: Trigger enable		Software capture control 0: Software capture 1: Undefined	Capture timing 00: Disable INT5 is rising edge 01: TB1IN0 ↑ TB1IN1 ↑ INT5 is rising edge 10: TB1IN0 ↑ TB1IN0 ↓ INT5 is falling edge 11: TA3OUT ↑ TA3OUT ↓ INT5 is rising edge		Up counter clear control 0: Disable 1:Enable	TMRB1 source clock 00: TB1IN0 pin input 01: φT1 10: φT4 11: φT16	
		Invert when the UC10 value is loaded in to TB1CP1H/L	Invert when match UC10 with TB1RG1H/L						

TMRB1 source clock

<TB1CLK1:0>	00	TB1IN0 pin input
	01	φT1
	10	φT4
	11	φT16

Control clearing for up counter (UC12)

<TB1CLE>	0	Disable
	1	Enable clearing by match with TB1RG1H/L

Capture/interrupt timing

		Capture control	INT5 control
<TB0CPM1:0>	00	Disable	INT5 occurs at the rising edge of TB1IN0 ↑
	01	Capture to TB1CP0H/L at rising edge of TB1IN0 Capture to TB1CP1H/L at rising edge of TB1IN1	
	10	Capture to TB1CP0H/L at rising edge of TB1IN0 Capture to TB1CP1H/L at falling edge of TB1IN0	INT5 occurs at the rising edge of TB1IN0 ↓
	11	Capture to TB1CP0H/L at rising edge of TA3OUT Capture to TB1CP1H/L at falling edge of TA3OUT	INT5 occurs at the rising edge of TB1IN0 ↑

Software capture

<TB1CP0I>	0	The value of up counter is captured to TB1CP0H/L
	1	Undefined

TB1FF1 control
Inverted when UC12 value matches the valued in TB1RG1H/L

<TB1ET1>	0	Disable inversion
	1	Enable inversion

TB1FF1 control
Inverted when UC10 value is captured into TB1CP1H/L

<TB1CT1>	0	Disable inversion
	1	Enable inversion

Note:When controlling capture by using TB1MOD<TB1CPM1:0>, control capture after setting SYSCR2<DRVE> to ”0”.

Figure 3.8.5 The Registers for TMRB0

TMRB0 Flip-Flop Control Register

		7	6	5	4	3	2	1	0
TB0FFCR	Bit symbol	–	–	TB0C1T1	TB0C0T1	TB0E1T1	TB0E0T1	TB0FF0C1	TB0FF0C0
(1183H)	Read/Write	W*		R/W				W*	
	Reset State	1	1	0	0	0	0	1	1
Read-modify -write instruction is prohibited.	Function	Always write “11”.		TB0FF0 inversion trigger 0: Disable trigger 1: Enable trigger				Control TB0FF0 00: Invert 01: Set 10: Clear 11: Don’t care * Always read as 11.	
				Invert when the UC value is loaded in to TB0CP1H/L	Invert when the UC value is loaded in to TB0CP0H/L	Invert when the UC value matches the value in TB0RG1H/L.	Invert when the UC value matches the value in TB0RG0H/L.		

Timer flip-flop control (TB0FF0)

<TB0FFC1:0>	00	Invert
	01	Set to “11”
	10	Clear to “00”
	11	Don’t care

TB0FF0 control

Inverted when UC10 value matches the valued in TB0RG0H/L

<TB0E0T1>	0	Disable inversion
	1	Enable inversion

TB0FF0 control

Inverted when UC10 value matches the valued in TB0RG1H/L

<TB0E1T1>	0	Disable inversion
	1	Enable inversion

TB0FF0 control

Inverted when UC10 value is captured into TB0CP0H/L

<TB0C0T1>	0	Disable inversion
	1	Enable inversion

TB0FF0 control

Inverted when UC10 value is captured into TB0CP1H/L

<TB0C1T1>	0	Disable inversion
	1	Enable inversion

Figure 3.8.6 The Registers for TMRB

TMRB1 Flip-Flop Control Register

TB1FFCR
(1193H)

Read-modify
-write
instruction is
prohibited.

	7	6	5	4	3	2	1	0
Bit symbol	TB1FF1C1	TB1FF1C0	TB1C1T1	TB1C0T1	TB1E1T1	TB1E0T1	TB1FFC1	TB1FFC0
Read/Write	W*		R/W				W*	
Reset State	1	1	0	0	0	0	1	1
Function	TB1FF1 control 00: Invert 01: Set 10: Clear 11: Don’t care * Always read as “11”.		TB0FF0 inversion trigger 0: Disable trigger 1: Enable trigger				Control TB1FF0 00: Invert 01: Set 10: Clear 11: Don’t care * Always read as 11.	
			Invert when the UC value is loaded in to TB1CP1H/L	Invert when the UC value is loaded in to TB1CP0H/L	Invert when the UC value matches the value in TB1RG1H/L.	Invert when the UC value matches the value in TB1RG0H/L.		

Timer flip-flop control(TB1FF0)

<TB1FFC1:0>	00	Invert
	01	Set to “11”
	10	Clear to “00”
	11	Don’t care

TB1FF0 control
Inverted when UC12 value matches the valued in TB1RG0H/L

<TB1E0T1>	0	Disable inversion
	1	Enable inversion

TB1FF0 control
Inverted when UC12 value matches the valued in TB1RG1H/L

<TB1E1T1>	0	Disable inversion
	1	Enable inversion

TB1FF0 control
Inverted when UC12 value is captured into TB1CP0H/L

<TB1C0T1>	0	Disable inversion
	1	Enable inversion

TB1FF0 control
Inverted when UC12 value is captured into TB1CP1H/L

<TB1C1T1>	0	Disable inversion
	1	Enable inversion

TB1FF1 control

<TB1FF1C1:0>	00	Invert value of TB1FF1
	01	Set TB1FF1 to “1”
	10	Set TB1FF1 to “0”
	11	Don’t care

Figure 3.8.7 The Registers for TMRB

Timer register

		7	6	5	4	3	2	1	0
TB0RG0L	bit Symbol	−							
(1188H)	Read/Write	W							
	Reset State	Undefined							
TB0RG0H	bit Symbol	−							
(1189H)	Read/Write	W							
	Reset State	Undefined							
TB0RG1L	bit Symbol	−							
(118AH)	Read/Write	W							
	Reset State	Undefined							
TB0RG1H	bit Symbol	−							
(118BH)	Read/Write	W							
	Reset State	Undefined							
TB1RG0L	bit Symbol	−							
(1198H)	Read/Write	W							
	Reset State	Undefined							
TB1RG0H	bit Symbol	−							
(1199H)	Read/Write	W							
	Reset State	Undefined							
TB1RG1L	bit Symbol	−							
(119AH)	Read/Write	W							
	Reset State	Undefined							
TB1RG1H	bit Symbol	−							
(119BH)	Read/Write	W							
	Reset State	Undefined							

Note: Read-modify-write instructuio is prohibited.

Figure 3.8.8 The Registers for TMRB

Capture register

		7	6	5	4	3	2	1	0
TB0CP0L (118CH)	bit Symbol	–							
	Read/Write	R							
	Reset State	Undefined							
TB0CP0H (118DH)	bit Symbol	–							
	Read/Write	R							
	Reset State	Undefined							
TB0CP1L (118EH)	bit Symbol	–							
	Read/Write	R							
	Reset State	Undefined							
TB0CP1H (118FH)	bit Symbol	–							
	Read/Write	R							
	Reset State	Undefined							
TB1CP0L (119CH)	bit Symbol	–							
	Read/Write	W							
	Reset State	Undefined							
TB1CP0H (119DH)	bit Symbol	–							
	Read/Write	R							
	Reset State	Undefined							
TB1CP1L (119EH)	bit Symbol	–							
	Read/Write	R							
	Reset State	Undefined							
TB1CP1H (119FH)	bit Symbol	–							
	Read/Write	R							
	Reset State	Undefined							

Note: Read-modify-write instructuio is prohibited.

Figure 3.8.9 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 interval time is set the timer register TB1RG1H/L to generate the interrupt INTTB11.

		7	6	5	4	3	2	1	0	
TB1RUN	←	0	0	X	X	–	0	X	0	Stop TMRB1.
INTETB1	←	X	1	0	0	X	0	0	0	Enable INTTB11 and set interrupt level 4. Disable INTTB10.
TB1FFCR	←	1	1	0	0	0	0	1	1	Disable the trigger.
TB1MOD	←	0	0	1	0	0	1	*	*	Select internal clock for input and disable the capture function.
				(** = 01, 10, 11)						
TB1RG1H/L	←	*	*	*	*	*	*	*	*	Set the interval time (16 bits).
		*	*	*	*	*	*	*	*	
TB1RUN	←	0	0	X	X	–	1	X	1	Start TMRB1.

X: Don't care, –: No change

(2) 16-bit event counter mode

In 16-bit timer mode as described in above, the timer can be used as an event counter by selecting the external clock (TB1IN0 pin input) as the input clock.

Up counter counting up by rising edge of TB1IN0 pin input. And execution software capture and reading capture value enable reading count value.

		7	6	5	4	3	2	1	0	
TB1RUN	←	0	0	X	X	–	0	X	0	Stop TMRB1.
PDCR	←	X	X	X	X	–	–	0	–	Set PD1 to TB1IN0 input mode.
PDFC2	←	X	X	X	X	–	–	0	X	
PDFC	←	X	X	X	X	–	–	1	–	
INTETB1	←	X	1	0	0	X	0	0	0	Set INTTB11 to enable (Interrupt level4). Set INTTB10 to disable.
TB1FFCR	←	1	1	0	0	0	0	1	1	Set trigger to disable.
TB1MOD	←	0	0	1	0	0	1	0	0	Set input clock to TB1IN0 pin input.
TB1RG1H/L	←	*	*	*	*	*	*	*	*	Set number of count. (16 bits)
		*	*	*	*	*	*	*	*	
TB1RUN	←	0	0	X	X	–	1	X	1	Start TMRB1.

X: Don’t care, –: No change

Note: When used as an event counter, set the prescaler to “RUN” (TB1RUN<TB1PRUN> = “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 TB1FF0 that is to be enabled by the match of the up counter UC12 with timer register TB1RG0H/L or TB1RG1H/L and to be output to TB1OUT0. In this mode the following conditions must be satisfied.

(Value set in TB1RG0H/L) < (Value set in TB1RG1H/L)

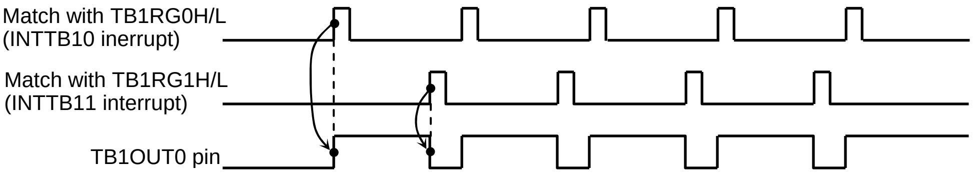


Figure 3.8.10 Programmable Pulse Generation (PPG) Output Waveforms

When the TB1RG0H/L double buffer is enabled in this mode, the value of register buffer 12 will be shifted into TB1RG0H/L at match with TB1RG1H/L. This feature facilitates the handling of low duty waves.

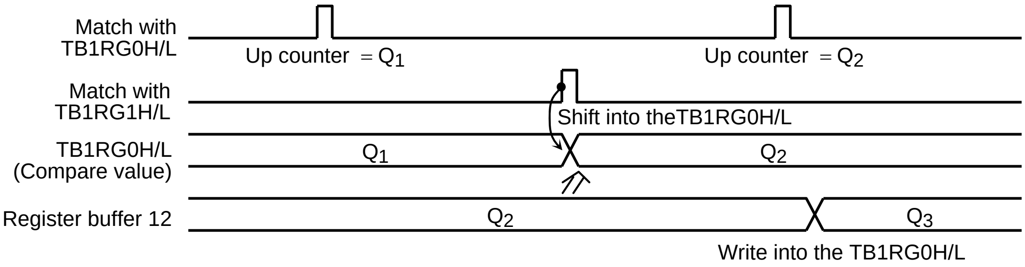


Figure 3.8.11 Operation of Register Buffer

The following block diagram illustrates this mode.

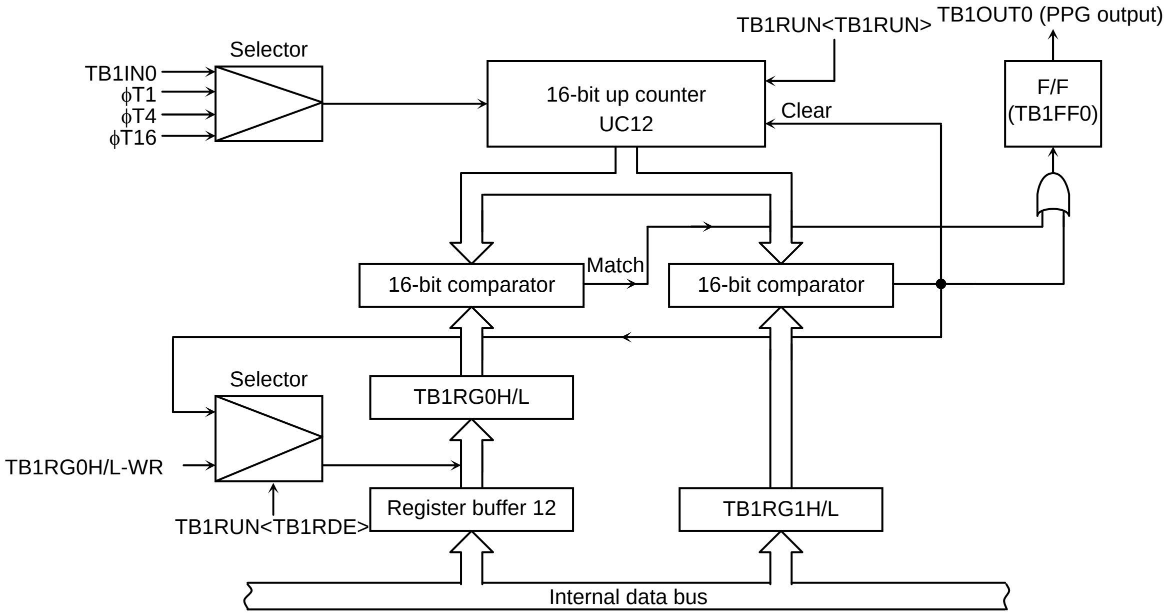


Figure 3.8.12 Block Diagram of 16-Bit Mode

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

		7	6	5	4	3	2	1	0	
TB1RUN	←	0	0	X	X	–	0	X	0	Disable the TB1RG0H/L double buffer and stop TMRB0.
TB1RG0H/L	←	*	*	*	*	*	*	*	*	Set the duty ratio (16 bits).
		*	*	*	*	*	*	*	*	
TB1RG1H/L	←	*	*	*	*	*	*	*	*	Set the frequency (16 bits).
		*	*	*	*	*	*	*	*	
TB1RUN	←	1	0	X	X	–	0	X	0	Enable the TB1RG0H/L double buffer.
										(The duty and frequency are changed on an INTTB11 interrupt.)
TB1FFCR	←	1	1	0	0	1	1	1	0	Set the mode to invert TB0FF0 at the match with TB1RG0H/L,
										TB1RG1H/L. Clear TB1FF0H/L to 0.
TB1MOD	←	0	0	1	0	0	1	*	*	Select the internal clock as the input clock and disable
				(** = 01, 10, 11)						the capture function.
PDFC2	←	X	X	X	–	0	–	–	X	
PDFC		X	X	X	–	1	–	–	–	Set PD3 to function as TB1OUT0.
PDCR	←	X	X	X	–	1	–	X	–	
TB1RUN	←	1	0	X	X	–	1	X	1	Start TMRB1.

X: Don't care, –: No change

(4) Capture function examples

Used capture function, they can be applicable in many ways, for example:

1. One-shot pulse output from external trigger pulse
2. Frequency measurement
3. Pulse width measurement
4. Measurement of difference time

1. One-shot pulse output from external trigger pulse

Set the up counter UC12 in free-running mode with the internal input clock, input the external trigger pulse from TB1IN0 pin, and load the value of up counter into capture register TB1CP0H/L at the rise edge of external trigger pulse.

When the interrupt INT5 is generated at the rise edge of external trigger pulse, set the TB1CP0H/L value (c) plus a delay time (d) to TB1RG0H/L (= c + d), and set the above set value (c + d) plus a one-shot width (p) to TB1RG1H/L (= c + d + p). And, set "11" to timer flip-flop control register TB1FFCR<TB1E1T1, TB1E0T1>. Set to trigger enable for be inverted timer flip-flop TB1FF0 by UC0 matching with TB1RG0H/L and with TB1RG1H/L. When interrupt INTTB11 occurs, this inversion will be disabled after one-shot pulse is output.

The (c), (d), and (p) correspond to c, d, and p in Figure 3.8.13.

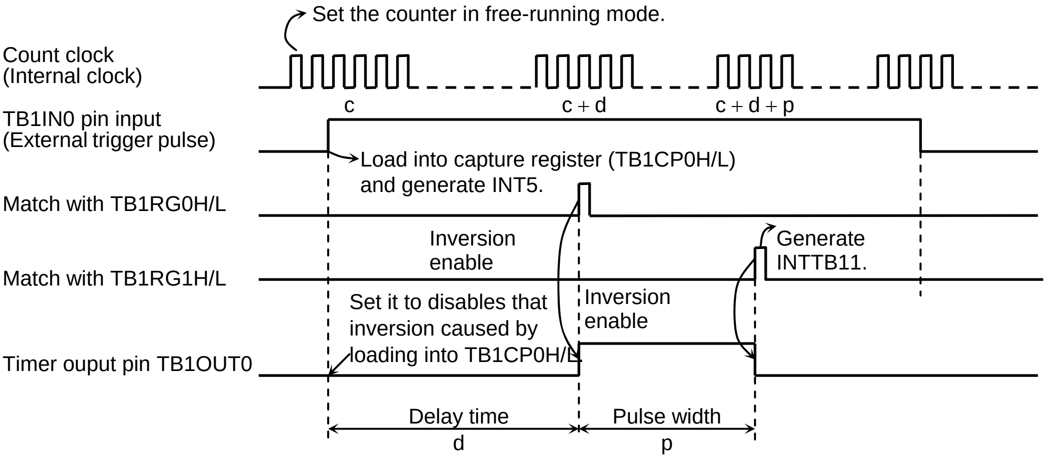


Figure 3.8.13 One-shot Pulse Output (with delay)

Example: To output a 2 [ms] one-shot pulse with a 3 [ms] delay to the external trigger pulse via the TB1IN0 pin.

```
                          * Clock state
                            System clock:            High frequency  (fc)
                            High speed clock gear:   1/1 (fc)
Setting in Main
                                                     → Set free running.
                                                     → Count using ϕT1.
TB1MOD     ←  X  X  1  0  1  0  0  1
                                                     → Load into TB1CP0 by rising edge of TB1IN0 pin input.
TB1FFCR    ←  X  X  0  0  0  0  1  0
                                                     → Clear TB1FF0 to 0.
                                                     → Disable inversion of TB1FF0.
PDCR       ←  X  X  X  –  1  –  X  –   ⎫
PDFC       ←  X  X  X  –  1  –  –  –   ⎬ Set PD3 to function as the TB1OUT0 pin.
PDFC2      ←  X  X  X  –  0  –  –  X   ⎭

INTE45     ←  X  1  0  0  X  –  –  –
                                         Enable INT5. Disable INTTB10 and INTTB11.
INTETB1    ←  X  0  0  0  X  0  0  0
TB1RUN     ←  –  0  X  X  –  1  X  1     Start TMRB1.

Setting in INT5
TB1RG0H/L  ←  TB1CP0H/L + 3 ms/ϕT1
TB1RG1H/L  ←  TB1RG0H/L + 2 ms/ϕT1
TB1FFCR    ←  X  X  –  –  1  1  –  –
                                                     → Enable inversion of TB1FF0 when match with
                                                       TB1RG0H/L or TB1RG1H/L.
INTETB1    ←  X  1  0  0  X  –  –  –                   Set INTTB11 to enable.

Setting in INTTB11
TB1FFCR    ←  X  X  –  –  0  0  –  –
                                                     → Disable inversion of TB1FF0 when match with
                                                       TB1RG0H/L or TB1RG1H/L.
INTETB1    ←  X  0  0  0  X  –  –  –                   Disable INTTB11.
X : Don’t care, – : No change
```

When delay time is unnecessary, invert timer flip-flop TB1FF0 when up counter value is loaded into capture register (TB1CP0H/L), and set the TB1CP0H/L value (c) plus the one-shot pulse width (p) to TB0RG1H/L when the interrupt INT5 occurs. The TB1FF0 inversion should be enable when the up counter (UC12) value matches TB1RG1H/L, and disabled when generating the interrupt INTTB11.

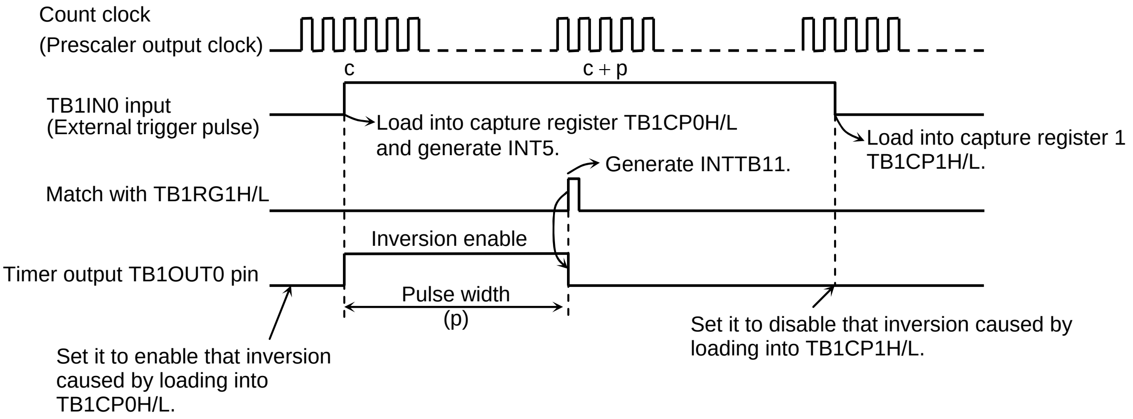

Count clock
(Prescaler output clock)
c
c + p
TB1IN0 input
(External trigger pulse)
Load into capture register TB1CP0H/L and generate INT5.
Load into capture register 1 TB1CP1H/L.
Generate INTTB11.
Match with TB1RG1H/L
Inversion enable
Timer output TB1OUT0 pin
Pulse width
(p)
Set it to enable that inversion caused by loading into TB1CP0H/L.
Set it to disable that inversion caused by loading into TB1CP1H/L.

Figure 3.8.14 One-shot Pulse Output (without delay)

2. Frequency measurement

The frequency of the external clock can be measured in this mode. Frequency is measured by the 8-bit timers TMRA23 and the 16-bit timer/event counter.

TMRA23 is used to setting of measurement time by inversion TA3FF.

Counter clock in TMRB1 select TB1IN0 pin input, and count by external clock input. Set to TB1MOD<TB1CPM1:0> = "11". The value of the up counter (UC12) is loaded into the capture register TB1CP0H/L at the rise edge of the timer flip-flop TA3FF of 8-bit timers (TMRA23), and into TB0CP1H/L at its fall edge.

The frequency is calculated by difference between the loaded values in TB1CP0H/L and TB1CP1H/L when the interrupt (INTTA2 or INTTA3) is generates by either 8-bit timer.

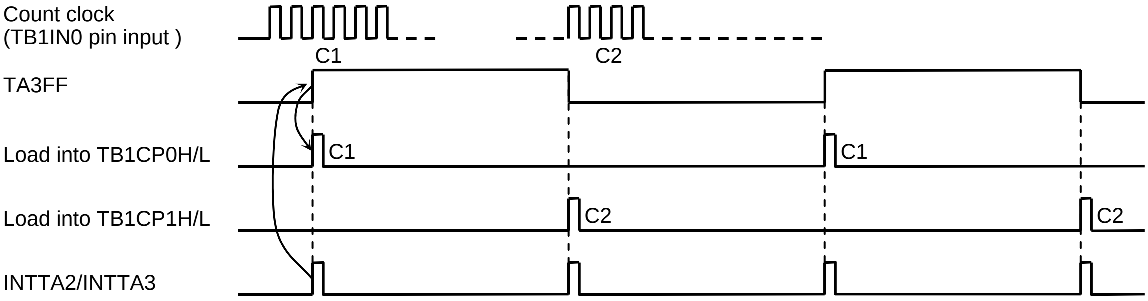


Figure 3.8.15 Frequency Measurement

For example, if the value for the level 1 width of TA3FF of the 8-bit timer is set to 0.5 s and the difference between the values in TB1CP0H/L and TB1CP1H/L is 100, the frequency is 100 ÷ 0.5 s = 200 Hz.

3. Pulse width measurement

This mode allows measuring the high level width of an external pulse. While keeping the 16-bit timer/event counter counting (Free running) with the prescaler output clock input, external pulse is input through the TB1IN0 pin. Then the capture function is used to load the UC12 values into TB1CP0H/L and TB1CP1H/L at the rising edge and falling edge of the external trigger pulse respectively. The interrupt INT5 occurs at the falling edge of TB1IN0.

The pulse width is obtained from the difference between the values of TB1CP0H/L and TB1CP1H/L and the internal clock cycle.

For example, if the prescaler output clock is 0.8 μs and the difference between TB1CP0H/L and TB1CP1H/L is 100, the pulse width will be 100 × 0.8 μs = 80 μs.

Additionally, the pulse width that is over the UC12 maximum count time specified by the clock source can be measured by changing software.

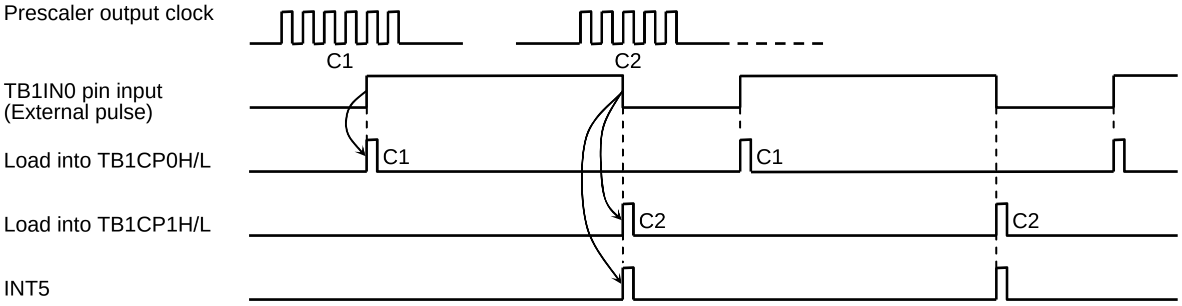


Figure 3.8.16 Pulse Width Measurement

Note: Pulse Width measure by setting “10” to TB1MOD<TB1CPM1:0>. The external interrupt INT5 is generated in timing of falling edge of TB1IN0 input. In other modes, it is generated in timing of rising edge of TB1IN0 input.

The width of low level can be measured from the difference between the first C2 and the second C1 at the second INT5 interrupt.

4. Measurement of difference time

This mode is used to measure the difference in time between the rising edges of external pulses input through TB1IN0 and TB1IN1.

Keep the 16-bit timer/event counter (TMRB1) counting (Free running) with the prescaler output clock, and load the UC12 value into TB1CP0H/L at the rising edge of the input pulse to TB1IN0. Then the interrupt INT5 is generated.

Similarly, the UC12 value is loaded into TB1CP1H/L at the rising edge of the input pulse to TB1IN1, generating the interrupt INT6.

The time difference between these pulses can be obtained by multiplying the value subtracted TB1CP0H/L from TB1CP1H/L and the internal clock cycle together at which loading the UC12 value into TB1CP0H/L and TB1CP1H/L has been done.

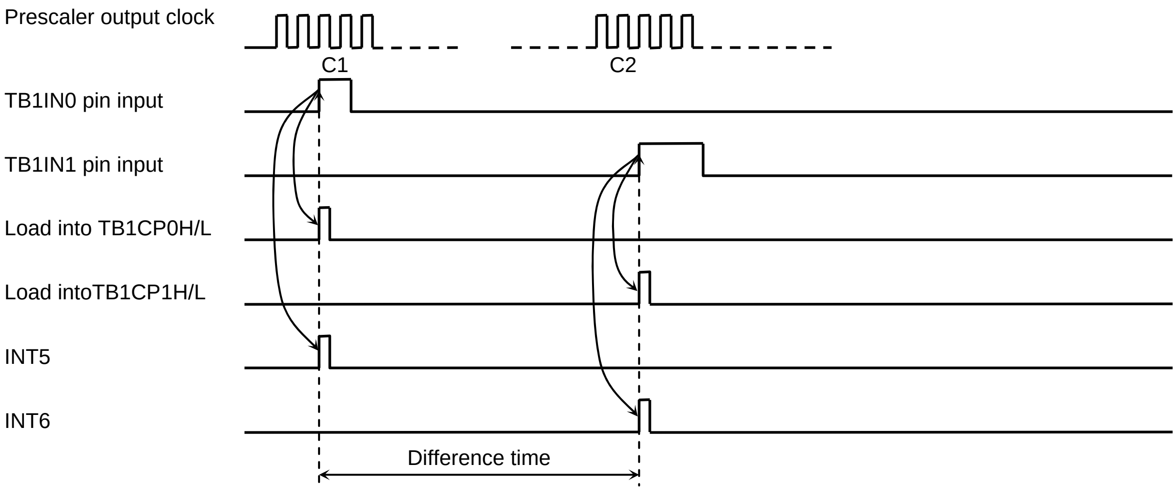


Figure 3.8.17 Measurement of Difference Time

3.9 Serial Channels

TMP92FD23A includes 3 serial I/O channels. Each channel is called SIO0, SIO1 and SIO2. For each channels either UART mode (asynchronous transmission) or I/O interface mode (synchronous transmission) can be selected.

I/O interface mode —— Mode 0: For transmitting and receiving I/O data using the synchronizing signal SCLK for extending I/O.

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, Figure 3.9.3 and Figure 3.9.4 are block diagrams for each channel.

Each channel can be used independently.

Each channel operates in the same function except for the following points; hence only the operation of channel 0 is explained below.

Table 3.9.1 Differences between Channels 0 to 1

	Channel 0	Channel 1	Channel 2
Pin name	TXD0 (PF0) RXD0 (PF1) $\overline{\text{CTS0}}$ /SCLK0 (PF2)	TXD1 (PF3) RXD1 (PF4) $\overline{\text{CTS1}}$/SCLK1 (PF5)	TXD2 (PD2) RXD2 (PD3) $\overline{\text{CTS2}}$ /SCLK2 (PD4)
IrDA mode	Yes	Yes	Yes

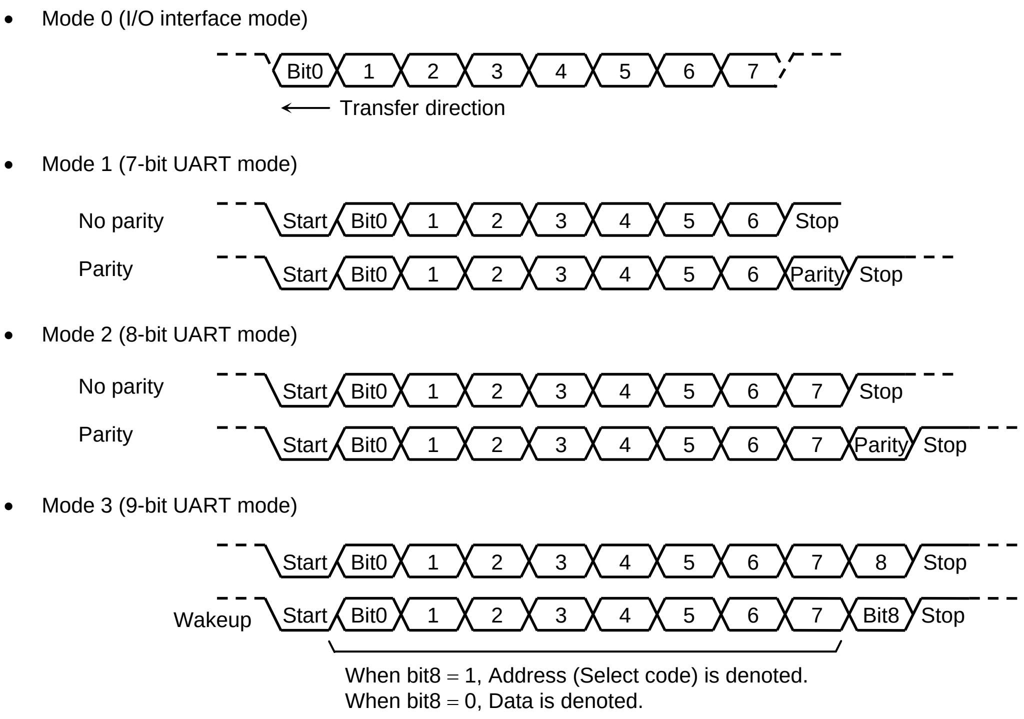


Figure 3.9.1 Data Formats

3.9.1 Block Diagrams

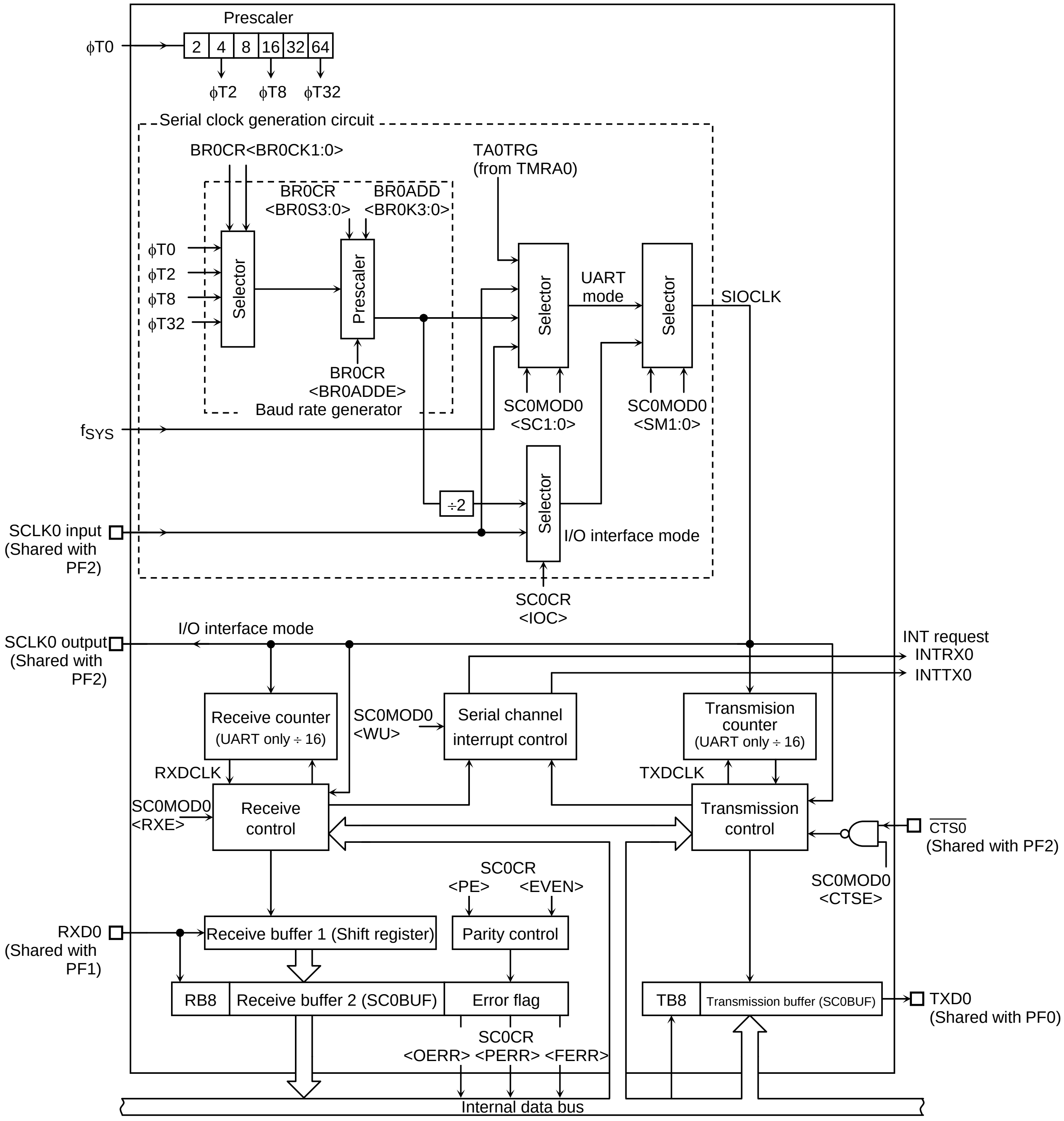


Figure 3.9.2 Block Diagram of the Serial Channel 0

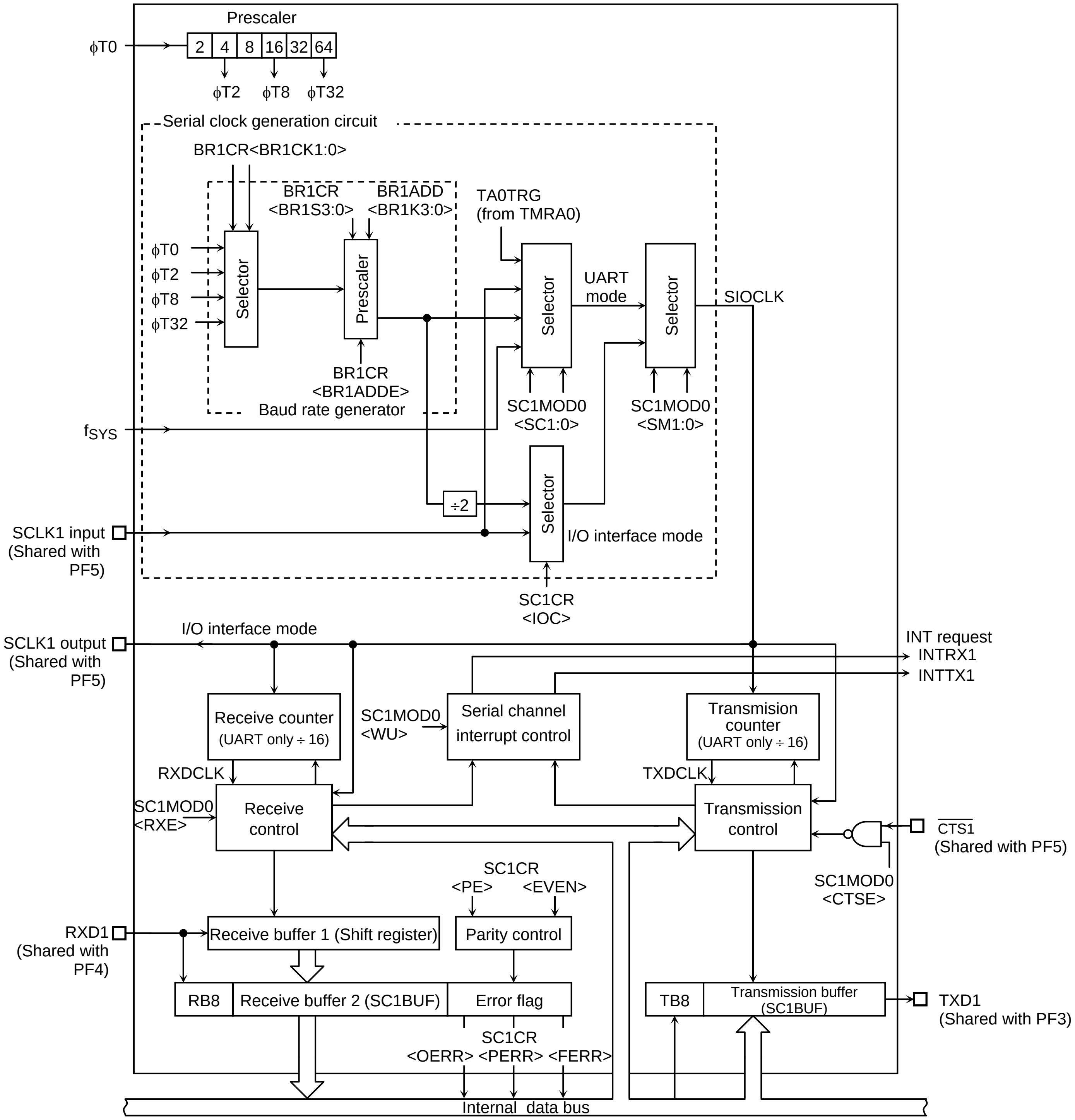


Figure 3.9.3 Block Diagram of the Serial Channel 1

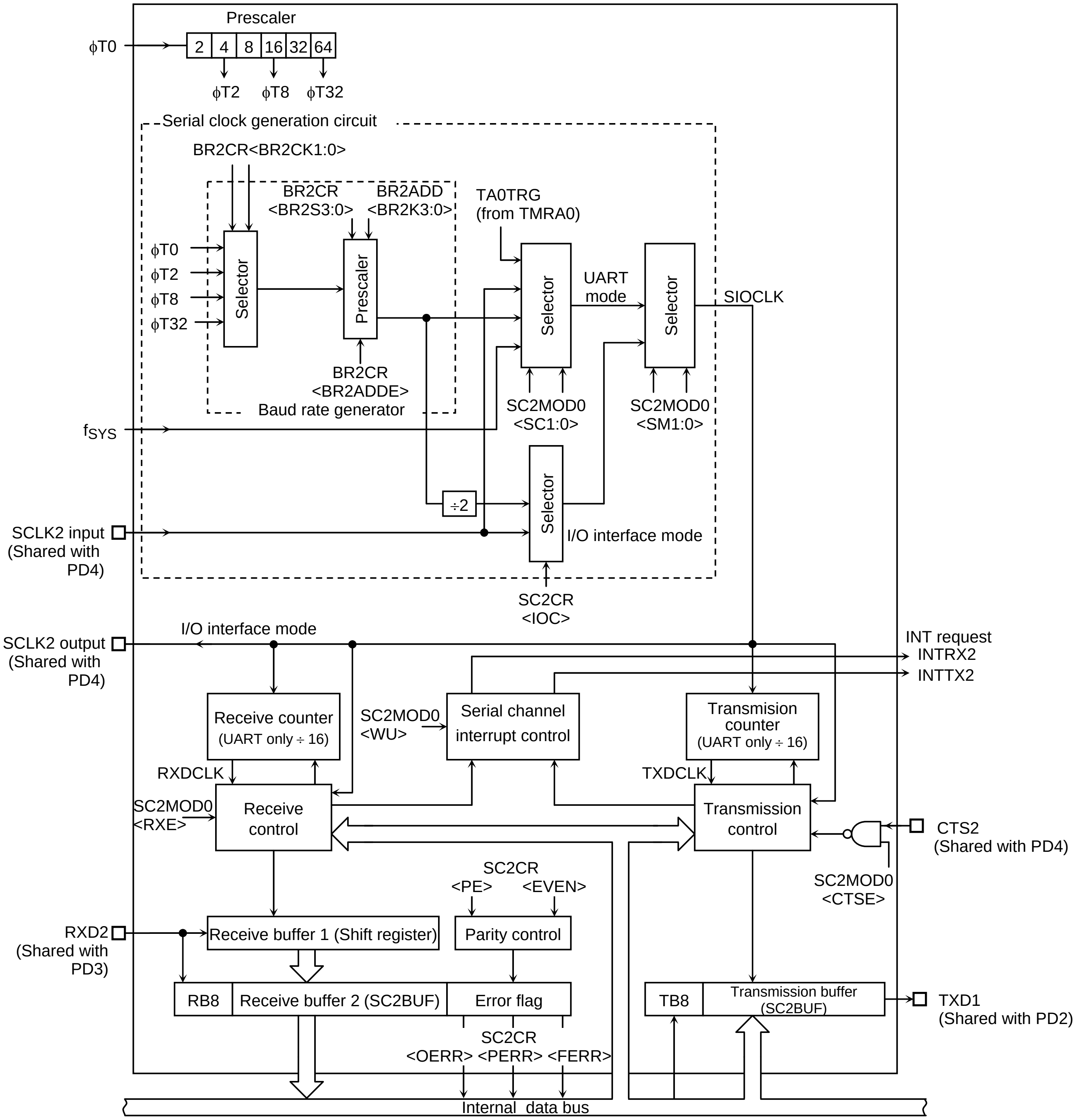


Figure 3.9.4 Block Diagram of the Serial Channel 2

3.9.2 Operation for Each Circuit

(1) Prescaler

There is a 6-bit prescaler for generating a clock to SIO0. The prescaler can be run only case of selecting the baud rate generator as the serial transfer 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

System clock SYSCR1 <SYSCK>	Clock Gear SYSCR1 <GEAR2:0>	–	Clock Resolution BR0CR<BR0CK1:0>			
			ϕT0	ϕT2(1/4)	ϕT8(1/16)	ϕT32(1/64)
1(fs)	–	1/4	fs/4	fs/16	fs/64	fs/256
0 (fc)	000(1/1)		fc/4	fc/16	fc/64	fc/256
	001(1/2)		fc/8	fc/32	fc/128	fc/512
	010(1/4)		fc/16	fc/64	fc/256	fc/1024
	011(1/8)		fc/32	fc/128	fc/512	fc/2048
	100(1/16)		fc/64	fc/256	fc/1024	fc/4096

The baud rate generator selects between 4 clock inputs: ϕT0, ϕT2, ϕT8, and ϕT32 among the prescaler outputs.

(2) Baud rate generator

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

The input clock to the baud rate generator, ϕT0, ϕT2, ϕT8 or ϕT32, is generated by the 6-bit SIO 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 + (16 – K)/16 or 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 …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…15) and the value of K set in BR0ADD<BR0K3:0> (K = 1, 2, 3…15)

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

- In I/O interface mode

The N + (16 – K)/16 division function is not available in I/O interface mode. Clear BR0CR<BR0ADDE> to 0 before dividing by N.

The method for calculating the transfer rate when the baud rate generator is used is explained below.

- In UART mode

$$\text{Baud rate} = \frac{\text{Input clock of baud rate generator}}{\text{Frequency divider for baud rate generator}} \div 16$$

- In I/O interface mode

$$\text{Baud rate} = \frac{\text{Input clock of baud rate generator}}{\text{Frequency divider for baud rate generator}} \div 2$$

- Integer divider (N divider)

  For example, when the source clock frequency (fC) is 12.288 MHz, the input clock is ϕT2 (fC/16), the frequency divider N (BR0CR<BR0S3:0>) = 5, and BR0CR<BR0ADDE> = 0, the baud rate in UART mode is as follows:

  * Clock state [High speed Clock gear : 1/1 (fc)

$$\text{Baud rate} = \frac{\text{Input clock of baud rate generator}}{\text{Frequency divider for baud rate generator}} \div 16$$

$$= \frac{f_C/16}{5} \div 16$$

$$= 12.288 \times 10^6 \div 16 \div 5 \div 16 = 9600 \text{ (bps)}$$

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

- N + (16 – K)/16 divider (UART mode only)

  Accordingly, when the source clock frequency (f_C) = 4.8 MHz, the input clock is ϕT0 (f_C/4), the frequency divider N (BR0CR<BR0S3:0>) = 3, K (BR0ADD<BR0K3:0>) = 7, and BR0CR<BR0ADDE> = 1, the baud rate in UART mode is as follows:

  * Clock state [High speed Clock gear : 1/1 (fc)

$$\text{Baud rate} = \frac{\text{Input clock of baud rate generator}}{\text{Frequency divider for baud rate generator}} \div 16$$

$$= \frac{f_C/4}{7 + \frac{(16-3)}{16}} \div 16$$

$$= 4.8 \times 10^6 \div 4 \div (7 + \frac{13}{16}) \div 16 = 9600 \text{ (bps)}$$

Table 3.9.3 show examples of UART mode transfer rates.

Additionally, the external clock input is available in the serial clock. (Serial channels 0, 1 and 2). 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) ≥ 4/f_C

- In I/O interface mode

  Baud rate = external clock input frequency

  It is necessary to satisfy (External clock input cycle) ≥ 16/f_C

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

Unit (Kbps)

f_C [MHz]	Input Clock Frequency Divider	ϕT0 (f_C/4)	ϕT2 (f_C/16)	ϕT8 (f_C/64)	ϕT32 (f_C/256)
9.8304	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
↑	10	9.600	2.400	0.600	0.150
12.2880	5	38.400	9.600	2.400	0.600
↑	A	19.200	4.800	1.200	0.300
14.7456	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
24.5760	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: Transfer rates in I/O interface mode are eight times faster than the values given above.

In UART mode, TMRA match detect signal (TA0TRG) can be used for serial transfer clock.

Method for calculating the timer output frequency which is needed when outputting trigger of timer

TA0TRG frequency = Baud rate × 16

Note: The TMRA0 match detect signal cannot be used as the transfer clock in I/O Interface mode.

(3) Serial clock generation circuit

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

- In I/O interface mode

  In SCLK output mode with the setting SC0CR<IOC> = 0, the basic clock is generated by dividing the output of the baud rate generator by 2, as described previously.

  In SCLK input mode with the setting SC0CR<IOC> = 1, the rising edge or falling edge will be detected according to the setting of the SC0CR<SCLKS> register to generate the basic clock.

- In UART mode

  The SC0MOD0<SC1:0> setting determines whether the baud rate generator clock, the internal clock f_{SYS}, the match detect signal from 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 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 I/O interface mode

  In SCLK output mode with the setting SC0CR<IOC> = 0, the RXD0 signal is sampled on the rising edge or falling of the shift clock which is output on the SCLK0 pin, according to the SC0CR<SCLKS> setting.

  In SCLK input mode with the setting SC0CR<IOC> = 1, the RXD0 signal is sampled on the rising or falling edge of the SCLK0 input, according to the SC0CR<SCLKS> setting.

- 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 one 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 cause an INTRX0 interrupt to be generated. The CPU only reads receiving buffer 2 (SC0BUF). Even before the CPU reads receiving buffer 2 (SC0BUF), the received data can be stored in receiving buffer 1. However, unless receiving buffer 2 (SC0BUF) is read before all bits of the next data 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.

SIO interrupt mode is selectable by the register SIMC.

(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.

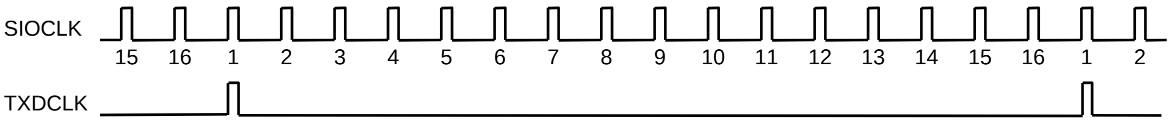


Figure 3.9.5 Generation of the Transmission Clock

(8) Transmission controller

- In I/O interface mode

In SCLK output mode with the setting SC0CR<IOC> = 0, the data in the transmission buffer is output one bit at a time to the TXD0 pin on the rising or falling edge of the shift clock which is output on the SCLK0 pin, according to the SC0CR<SCLKS> setting.

In SCLK input mode with the setting SC0CR<IOC> = 1, the data in the transmission buffer is output one bit at a time on the TXD0 pin on the rising or falling edge of the SCLK0 input, according to the SC0CR<SCLKS> setting.

- 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 $\overline{\text{CTS}}$ 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 $\overline{\text{CTS0}}$ pin goes high on completion of the current data send, data transmission is halted until the $\overline{\text{CTS0}}$ pin goes 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.

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

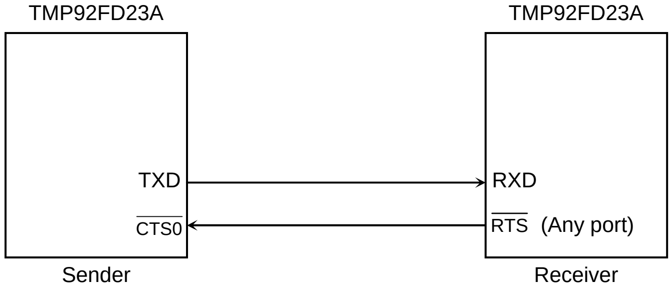
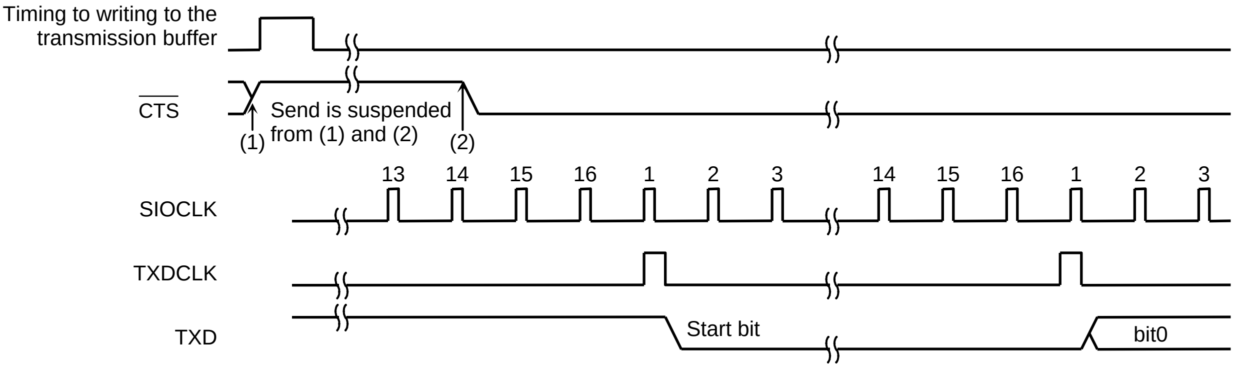


Figure 3.9.6 Handshake Function


Timing to writing to the transmission buffer
$\overline{\text{CTS}}$
Send is suspended from (1) and (2)
(1)
(2)
13
14
15
16
1
2
3
14
15
16
1
2
3
SIOCLK
TXDCLK
TXD
Start bit
bit0

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 3.9.7 $\overline{\text{CTS}}$ (Clear to send) Timing

(9) Transmission buffer

The transmission buffer (SC0BUF) shifts out and sends the transmission data written from the CPU form the least significant bit (LSB) in order. When all the bits are shifted out, the transmission buffer becomes empty and 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.

   The below is a recommended flow when the overrun-error is generated.

   (INTRX interrupt routine)

   1) Read receiving buffer

   2) Read error flag

   3) If <OERR> = 1

      then

      a) Set to disable receiving (Write "0" to SC0MOD0<RXE>)

      b) Wait to terminate current frame

      c) Read receiving buffer

      d) Read error flag

      e) Set to enable receiving (Write "1" to SC0MOD0<RXE>)

      f) Request to transmit again

   4) Other

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 Bits (Note)	8 Bits + Parity (Note)	8 Bits, 7 Bits + Parity, 7 Bits
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

Note1: In 9-bit and 8-bit parity modes, interrupts coincide with the ninth bit pulse.
Thus, when servicing the interrupt, it is necessary to wait for a 1-bit period (to allow the stop bit to be transferred) to allow checking for a framing error.

Note2: The higher the transfer rate, the later than the middle receive interrupts and errors occur.

Transmitting

Mode	9 Bits	8 Bits + Parity	8 Bits, 7 Bits + Parity, 7 Bits
Interrupt Timing	Just before stop bit is transmitted	Just before stop bit is transmitted	Just before stop bit is transmitted

2. I/O interface

Transmission Interrupt Timing	SCLK output mode	Immediately after last bit data. (See Figure 3.9.25.)
	SCLK input mode	Immediately after rise of last SCLK signal rising mode, or immediately after fall in falling mode. (See Figure 3.9.26.)
Receiving Interrupt Timing	SCLK output mode	Timing used to transfer received to data receive buffer 2 (SC0BUF) (e.g. immediately after last SCLK). (See Figure 3.9.27.)
	SCLK input mode	Timing used to transfer received data to receive buffer 2 (SC0BUF) (e.g. immediately after last SCLK). (See Figure 3.9.28.)

3.9.3 SFR

		7	6	5	4	3	2	1	0
SC0MOD0 (1202H)	Bit symbol	TB8	CTSE	RXE	WU	SM1	SM0	SC1	SC0
	Read/Write	R/W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Transfer data bit8	Hand shake 0: CTS disable 1: CTS enable	Receive function 0: Receive disable 1: Receive enable	Wakeup function 0: Disable 1: Enable	Serial transmission mode 00: I/O interface mode 01: 7-bit UART mode 10: 8-bit UART mode 11: 9-bit UART mode		Serial transmission clock (UART) 00: TMRA0 trigger 01: Baud rate generator 10: Internal clock f_{SYS} 11: External clcok (SCLK0 input)	

Serial transmission clock source (UART)

00	TMRA0 match detect signal
01	Baud rate generator
10	Internal clock f_{SYS}
11	External clock (SCLK0 input)

Note: The clock selection for the I/O interface mode is controlled by the serial control register (SC0CR).

Serial transmission mode

00	I/O interface mode	
01	UART mode	7-bit mode
10		8-bit mode
11		9-bit mode

Wakeup function

	9-bit UART	Other modes
0	Interrupt generated when data is received	Don't care
1	Interrupt generated only when SC0CR<RB8> = 1	

Receiving function

0	Receive disabled
1	Receive enabled

Handshake function ($\overline{\text{CTS}}$ pin)

0	Disabled (always transferable)
1	Enabled

Transmission data bit8

Figure 3.9.8 Serial Mode Control Register (for SIO0)

SC1MOD0
(120AH)

	7	6	5	4	3	2	1	0
Bit symbol	TB8	CTSE	RXE	WU	SM1	SM0	SC1	SC0
Read/Write	R/W							
Reset State	0	0	0	0	0	0	0	0
Function	Transfer data bit8	Hand shake 0: CTS disable 1: CTS enable	Receive function 0: Receive disable 1: Receive enable	Wakeup function 0: Disable 1: Enable	Serial transmission mode 00: I/O interface mode 01: 7-bit UART mode 10: 8-bit UART mode 11: 9-bit UART mode		Serial transmission clock (UART) 00: TMRA0 trigger 01: Baud rate generator 10: Internal clock f_{SYS} 11: External clock (SCLK1 input)	

→Serial transmission clock source (for UART)

00	TMRA0 match detect signal
01	Baud rate generator
10	Internal clock f_{SYS}
11	External clock (SCLK1 input)

Note: The clock selection for the I/O interface mode is controlled by the serial control register (SC1CR).

→Serial transmission mode

00	I/O Interface mode	
01	UART mode	7-bit mode
10		8-bit mode
11		9-bit mode

→Wakeup function

	9-bit UART	Other modes
0	Interrupt generated when data is received	Don't care
1	Interrupt generated only when SC1CR<RB8> = 1	

→Receiving function

0	Receive disabled
1	Receive enabled

→Handshake function ($\overline{\text{CTS}}$ pin)

0	Disabled (always transferable)
1	Enabled

→Transmission data bit8

Figure 3.9.9 Serial Mode Control Register (for SIO1)

		7	6	5	4	3	2	1	0
SC2MOD0 (1212H)	Bit symbol	TB8	CTSE	RXE	WU	SM1	SM0	SC1	SC0
	Read/Write	R/W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Transfer data bit8	Hand shake 0: CTS disable 1: CTS enable	Receive function 0: Receive disable 1: Receive enable	Wakeup function 0: Disable 1: Enable	Serial transmission mode 00: I/O interface mode 01: 7-bit UART mode 10: 8-bit UART mode 11: 9-bit UART mode		Serial transmission clock (UART) 00: TMRA0 trigger 01: Baud rate generator 10: Internal clock f_{SYS} 11: External clock (SCLK2 input)	

Serial transmission clock source (for UART)

00	TMRA0 match detect signal
01	Baud rate generator
10	Internal clock f_{SYS}
11	External clock (SCLK2 input)

Note: The clock selection for the I/O interface mode is controlled by the serial control register (SC2CR).

Serial transmission mode

00	I/O Interface mode	
01	UART mode	7-bit mode
10		8-bit mode
11		9-bit mode

Wakeup function

	9-bit UART	Other modes
0	Interrupt generated when data is received	Don’t care
1	Interrupt generated only when SC2CR<RB8> = 1	

Receiving function

0	Receive disabled
1	Receive enabled

Handshake function ($\overline{\text{CTS}}$ pin)

0	Disabled (always transferable)
1	Enabled

Transmission data bit8

Figure 3.9.10 Serial Mode Control Register (for SIO2)

SC0CR
(1201H)

	7	6	5	4	3	2	1	0
Bit symbol	RB8	EVEN	PE	OERR	PERR	FERR	SCLKS	IOC
Read/Write	R	R/W		R (Cleared to 0 when read)			R/W	
Reset State	Undefined	0	0	0	0	0	0	0
Function	Received data bit8	Parity 0: Odd 1: Even	Parity addition 0: Disable 1: Enable	1: Error Overrun	Parity	Framing	0: SCLK0 ↑ 1: SCLK0 ↓	0: Baud rate generator 1: SCLK0 pin input

I/O interface input clock selection

0	Baud rate generator
1	SCLK0 pin input

Edge selection for SCLK pin (I/O mode)

0	Transmits and receivers data on rising edge of SCLK0.	↑
1	Transmits and receivers data on falling edge SCLK0.	↓

Framing error flag
Parity error flag
Overrun error flag
} Cleared to 0 when read

Parity additions enable

0	Disabled
1	Enabled

Even parity addition/check

0	Odd parity
1	Even parity

Received data bit8

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

Figure 3.9.11 Serial Control Register (for SIO0)

SC1CR
(1209H)

	7	6	5	4	3	2	1	0
Bit symbol	RB8	EVEN	PE	OERR	PERR	FERR	SCLKS	IOC
Read/Write	R	R/W		R (cleared to 0 when read)			R/W	
Reset State	Undefined	0	0	0	0	0	0	0
Function	Received data bit8	Parity 0: Odd 1: Even	Parity addition 0: Disable 1: Enable	1: Error			0: SCLK1 ↑ 1: SCLK1 ↓	0: Baud rate generator 1: SCLK1 pin input
				Overrun	Parity	Framing		

I/O interface input clock select

0	Baud rate generator
1	SCLK1 pin input

Edge selection for SCLK pin (Input/Output mode)

0	Transmits and receives data on rising edge of SCLK1. ↑
1	Transmits and receives data on falling edge of SCLK1. ↓

Framing error flag
Parity error flag
Overrun error flag

Cleared to 0 when read

Parity additions enable

0	Disabled
1	Enabled

Even parity addition/check

0	Odd parity
1	Even parity

Received data bit8

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

Figure 3.9.12 Serial Control Register (for SIO1)

SC2CR
(1211H)

	7	6	5	4	3	2	1	0
Bit symbol	RB8	EVEN	PE	OERR	PERR	FERR	SCLKS	IOC
Read/Write	R	R/W		R (cleared to 0 when read)			R/W	
Reset State	Undefined	0	0	0	0	0	0	0
Function	Received data bit8	Parity 0: Odd 1: Even	Parity addition 0: Disable 1: Enable	1: Error			0: SCLK2 ↑ 1: SCLK2 ↓	0: Baud rate generator 1: SCLK2 pin input
				Overrun	Parity	Framing		

I/O interface input clock select

0	Baud rate generator
1	SCLK2 pin input

Edge selection for SCLK pin (Input/Output mode)

0	Transmits and receives data on rising edge of SCLK2. ↑
1	Transmits and receives data on falling edge of SCLK2. ↓

Framing error flag
Parity error flag
Overrun error flag
} Cleared to 0 when read

Parity additions enable

0	Disabled
1	Enabled

Even parity addition/check

0	Odd parity
1	Even parity

Received data bit8

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

Figure 3.9.13 Serial Control Register (for SIO2)

BR0CR (1203H)

	7	6	5	4	3	2	1	0
Bit symbol	–	BR0ADDE	BR0CK1	BR0CK0	BR0S3	BR0S2	BR0S1	BR0S0
Read/Write	R/W							
Reset State	0	0	0	0	0	0	0	0
Function	Always write “0”.	+(16 − K)/16 division 0: Disable 1: Enable	00: ϕT0 01: ϕT2 10: ϕT8 11: ϕT32		Divided frequency setting			

+(16 − K)/16 division enable

0	Disable
1	Enable

Setting the input clock of baud rate generator

00	Internal clock ϕT0
01	Internal clock ϕT2
10	Internal clock ϕT8
11	Internal clock ϕT32

BR0ADD (1204H)

	7	6	5	4	3	2	1	0
Bit symbol					BR0K3	BR0K2	BR0K1	BR0K0
Read/Write					R/W			
Reset State					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)	0001 (N = 1) (Only UART) to 1111 (N = 15) 0000 (N = 16)
0000	Disable	Disable	Divided by N
0001 (K = 1) to 1111 (K = 15)	Disable	Divided by N + (16 − K)/16	

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

N	UART mode	I/O mode
2 to 15	○	×
1 , 16	×	×

The baud rate generator can be set “1” in UART mode only when the +(16-K)/16 division function is not used. Do not use in I/O interface mode.

Note2:Set BR0CR <BR0ADDE> to 1 after setting K (K = 1 to 15) to BR0ADD<BR0K3:0> when the +(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 3.9.14 Baud Rate Generator Control (for SIO0)

BR1CR (120BH)

	7	6	5	4	3	2	1	0
Bit symbol	–	BR1ADDE	BR1CK1	BR1CK0	BR1S3	BR1S2	BR1S1	BR1S0
Read/Write	R/W							
Reset State	0	0	0	0	0	0	0	0
Function	Always write “0”.	+ (16 − K)/16 division 0: Disable 1: Enable	00: ϕT0 01: ϕT2 10: ϕT8 11: ϕT32		Divided frequency setting			

+ (16 − K)/16 division enable

0	Disabled
1	Enabled

Input clock selection for baud rate generator

00	Internal clock ϕT0
01	Internal clock ϕT2
10	Internal clock ϕT8
11	Internal clock ϕT32

BR1ADD (120CH)

	7	6	5	4	3	2	1	0
Bit symbol					BR1K3	BR1K2	BR1K1	BR1K0
Read/Write					R/W			
Reset State					0	0	0	0
Function					Set frequency divisor K (divided by N + (16 − K)/16).			

Baud rate generator frequency divisor setting

	BR1CR<BR1ADDE> = 1		BR1CR<BR1ADDE> = 0
BR1CR <BR1S3:0> BR1ADD <BR1K3:0>	0000 (N = 16) or 0001 (N = 1)	0010 (N = 2) to 1111 (N = 15)	0001 (N = 1) (Only UART) to 1111 (N = 15) 0000 (N = 16)
0000	Disable	Disable	Divided by N
0001 (K = 1) to 1111 (K = 15)	Disable	Divided by N + (16 − K)/16	

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

N	UART mode	I/O mode
2 to 15	○	×
1 , 16	×	×

The baud rate generator can be set “1” in UART mode only when the +(16-K)/16 division function is not used. Do not use in I/O interface mode.

Note2:Set BR1CR <BR1ADDE> to 1 after setting K (K = 1 to 15) to BR1ADD<BR1K3:0> when the +(16-K)/16 division function is used. Writes to unused bits in the BR1ADD register do not affect operation, and undefined data is read from these unused bits.

Figure 3.9.15 Baud Rate Generator Control (for SIO1)

BR2CR (1213H)

	7	6	5	4	3	2	1	0
Bit symbol	–	BR2ADDE	BR2CK1	BR2CK0	BR2S3	BR2S2	BR2S1	BR2S0
Read/Write	R/W							
Reset State	0	0	0	0	0	0	0	0
Function	Always write "0".	+(16 − K)/16 division 0: Disable 1: Enable	00: ϕT0 01: ϕT2 10: ϕT8 11: ϕT32		Divided frequency setting			

+(16 − K)/16 division enable

0	Disable
1	Enable

Setting the input clock of baud rate generator

00	Internal clock ϕT0
01	Internal clock ϕT2
10	Internal clock ϕT8
11	Internal clock ϕT32

BR2ADD (1214H)

	7	6	5	4	3	2	1	0
Bit symbol					BR2K3	BR2K2	BR2K1	BR2K0
Read/Write					R/W			
Reset State					0	0	0	0
Function					Sets frequency divisor "K" (divided by N + (16 − K)/16).			

Sets baud rate generator frequency divisor

	BR2CR<BR2ADDE> = 1		BR2CR<BR2ADDE> = 0
BR2CR <BR2S3:0> BR2ADD <BR2K3:0>	0000 (N = 16) or 0001 (N = 1)	0010 (N = 2) to 1111 (N = 15)	0001 (N = 1) (Only UART) to 1111 (N = 15) 0000 (N = 16)
0000	Disable	Disable	Divided by N
0001 (K = 1) to 1111 (K = 15)	Disable	Divided by N + (16 − K)/16	

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

N	UART mode	I/O mode
2 to 15	○	×
1 , 16	×	×

The baud rate generator can be set "1" in UART mode only when the +(16-K)/16 division function is not used. Do not use in I/O interface mode.

Note2:Set BR2CR <BR2ADDE> to 1 after setting K (K = 1 to 15) to BR2ADD<BR2K3:0> when the +(16-K)/16 division function is used. Writes to unused bits in the BR2ADD register do not affect operation, and undefined data is read from these unused bits.

Figure 3.9.16 Baud Rate Generator Control (for SIO2)

SC0BUF
(1200H)

7	6	5	4	3	2	1	0	
TB7	TB6	TB5	TB4	TB3	TB2	TB1	TB0	(Transmission)

7	6	5	4	3	2	1	0	
RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	(Receiving)

Note: Prohibit read-modify-write for SC0BUF.

Figure 3.9.17 Serial Transmission/Receiving Buffer Registers (for SIO0)

SC0MOD1
(1205H)

	7	6	5	4	3	2	1	0
Bit symbol	I2S0	FDPX0						
Read/Write	R/W							
Reset State	0	0						
Function	IDLE2 0: Stop 1: Run	Duplex 0: Half 1: Full						

Figure 3.9.18 Serial Mode Control Register 1 (for SIO0)

SC1BUF
(1208H)

7	6	5	4	3	2	1	0	
TB7	TB6	TB5	TB4	TB3	TB2	TB1	TB0	(Transmission)

7	6	5	4	3	2	1	0	
RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	(Receiving)

Note: Prohibit read-modify-write for SC1BUF.

Figure 3.9.19 Serial Transmission/Receiving Buffer Registers (for SIO1)

SC1MOD1
(120DH)

	7	6	5	4	3	2	1	0
Bit symbol	I2S1	FDPX1						
Read/Write	R/W							
Reset State	0	0						
Function	IDLE2 0: Stop 1: Run	Duplex 0: Half 1: Full						

Figure 3.9.20 Serial Mode Control Register 1 (for SIO1)

SC2BUF
(1210H)

7	6	5	4	3	2	1	0	
TB7	TB6	TB5	TB4	TB3	TB2	TB1	TB0	(Transmission)

7	6	5	4	3	2	1	0	
RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	(Receiving)

Note: Prohibit read-modify-write for SC2BUF.

Figure 3.9.21 Serial Transmission/Receiving Buffer Registers (for SIO2)

SC2MOD1
(1215H)

	7	6	5	4	3	2	1	0
Bit symbol	I2S2	FDPX2						
Read/Write	R/W							
Reset State	0	0						
Function	IDLE2 0: Stop 1: Run	Duplex 0: Half 1: Full						

Figure 3.9.22 Serial Mode Control Register 1 (for SIO2)

3.9.4 Operation in Each Mode

(1) Mode 0 (I/O interface mode)

This mode allows an increase in the number of I/O pins available for transmitting data to or receiving data from an external shift register.

This mode includes the SCLK output mode to output synchronous clock SCLK and SCLK input mode to input external synchronous clock SCLK.

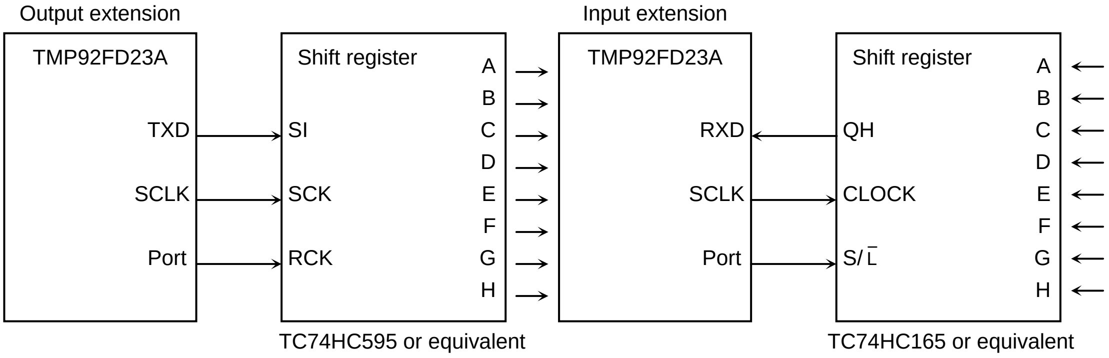


Figure 3.9.23 SCLK Output Mode Connection Example

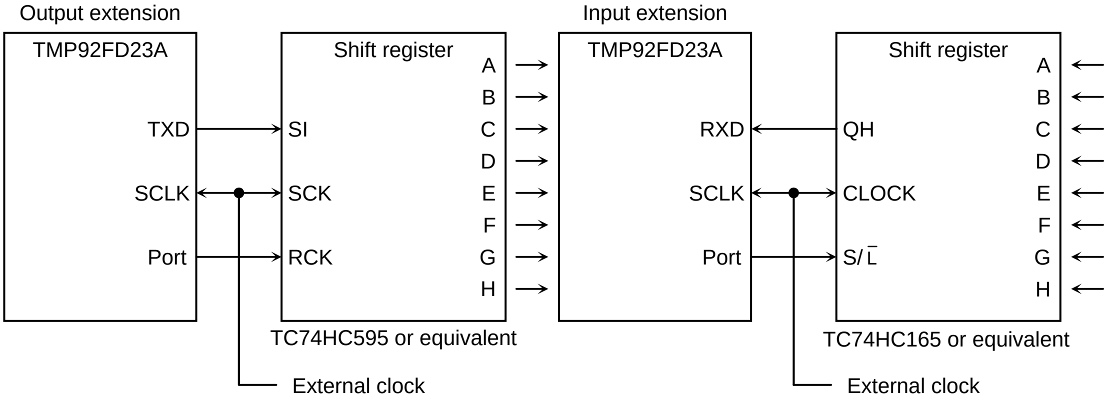


Figure 3.9.24 Example of SCLK Input Mode Connection

1. Transmission

In SCLK output mode 8-bit data and a synchronous clock are output on the TXD0 and SCLK0 pins respectively each time the CPU writes the data to the transmission buffer. When all data is output, INTES0<ITX0C> will be set to generate the INTTX0 interrupt.

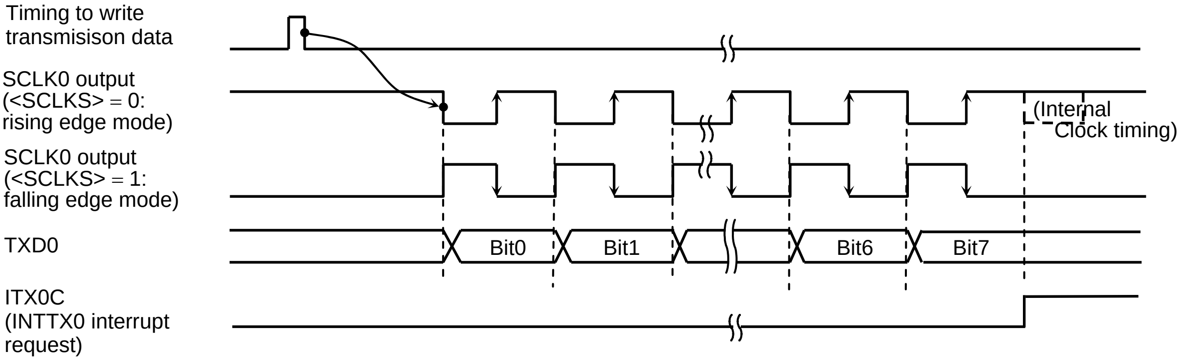

Timing to write transmisison data
SCLK0 output (<SCLKS> = 0: rising edge mode)
SCLK0 output (<SCLKS> = 1: falling edge mode)
TXD0
ITX0C (INTTX0 interrupt request)
(Internal Clock timing)
Bit0
Bit1
Bit6
Bit7

Figure 3.9.25 Transmitting Operation in I/O Interface Mode (SCLK0 output mode)

In SCLK input mode, 8-bit data is output on the TXD0 pin when the SCLK0 input becomes active after the data has been written to the transmission buffer by the CPU.

When all data is output, INTES0<ITX0C> will be set to generate INTTX0 interrupt.

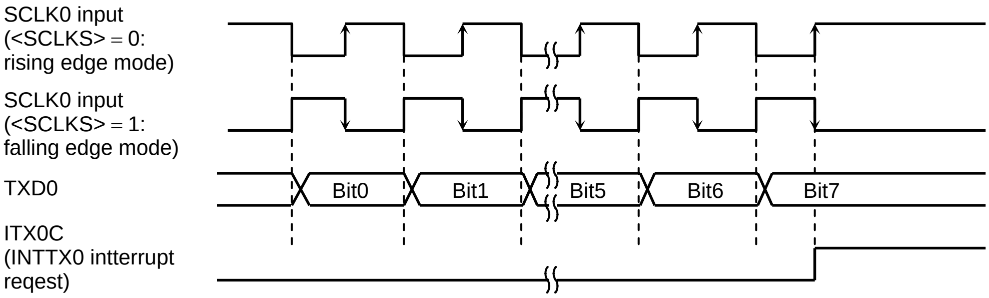

SCLK0 input (<SCLKS> = 0: rising edge mode)
SCLK0 input (<SCLKS> = 1: falling edge mode)
TXD0
ITX0C (INTTX0 intterrupt reqest)
Bit0
Bit1
Bit5
Bit6
Bit7

Figure 3.9.26 Transmitting Operation in I/O Interface Mode (SCLK0 input mode)

2. Receiving

In SCLK output mode the synchronous clock is output on the SCLK0 pin and the data is shifted to receiving buffer 1. This is initiated when the receive interrupt flag INTES0<IRX0C> is cleared as the received data is read. When 8-bit data is received, the data is transferred to receiving buffer 2 (SC0BUF) following the timing shown below and INTES0<IRX0C> is set to 1 again, causing an INTRX0 interrupt to be generated.

Setting SC0MOD0<RXE> to 1 initiates SCLK0 output.

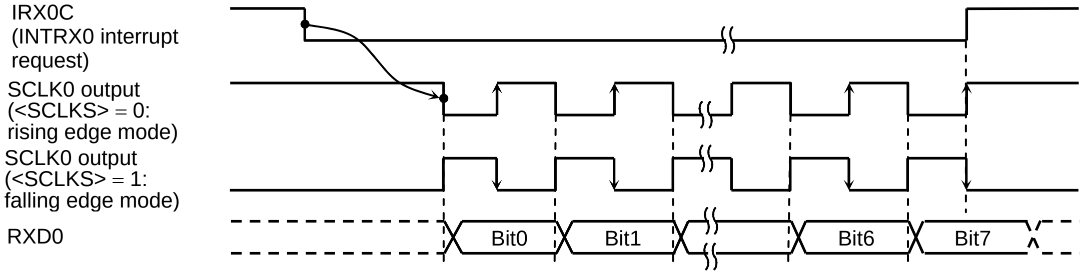

IRX0C
(INTRX0 interrupt request)
SCLK0 output
(<SCLKS> = 0: rising edge mode)
SCLK0 output
(<SCLKS> = 1: falling edge mode)
RXD0
Bit0
Bit1
Bit6
Bit7

Figure 3.9.27 Receiving Operation in I/O Interface Mode (SCLK0 output mode)

In SCLK input mode the data is shifted to receiving buffer 1 when the SCLK input goes active. The SCLK input goes active when the receive interrupt flag INTES0<IRX0C> is cleared as the received data is read. When 8-bit data is received, the data is shifted to receiving buffer 2 (SC0BUF) following the timing shown below and INTES0<IRX0C> is set to 1 again, causing an INTRX0 interrupt to be generated.

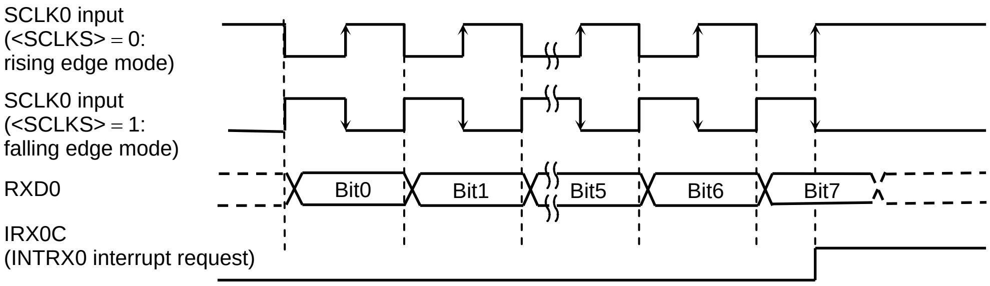

SCLK0 input
(<SCLKS> = 0: rising edge mode)
SCLK0 input
(<SCLKS> = 1: falling edge mode)
RXD0
Bit0
Bit1
Bit5
Bit6
Bit7
IRX0C
(INTRX0 interrupt request)

Figure 3.9.28 Receiving Operation in I/O Interface Mode (SCLK0 input mode)

Note: The system must be put in the receive enable state (SC0MOD0<RXE> = 1) before data can be received.

3. Transmission and receiving (Full duplex mode)

When full duplex mode is used, set the receive interrupt level to 0 and set enable the level of transmit interrupt. Ensure that the program which transmits the interrupt reads the receiving buffer before setting the next transmit data.

The following is an example of this:

Example: Channel 0, SCLK output
Baud rate = 9600 bps
fc = 14.7456 MHz

Main routine

	7	6	5	4	3	2	1	0	
INTES0	X	0	0	1	X	0	0	0	Set the INTTX0 level to 1. Set the INTRX0 level to 0.
PFCR	–	–	–	–	–	1	0	1	Set PF0, PF1 and PF2 to function as the TXD0,
PFFC	–	–	–	–	–	1	1	1	RXD0 and SCLK0 pins respectively.
SC0MOD0	0	0	0	0	0	0	0	0	Select I/O interface mode.
SC0MOD1	1	1	0	0	0	0	0	0	Select full duplex mode.
SC0CR	0	0	0	0	0	0	0	0	Set the SCLK output, transmit on negative edge, and receive on positive edge.
BR0CR	0	0	1	1	0	0	1	1	Set to 9600 bps.
SC0MOD0	0	0	1	0	0	0	0	0	Set receive to enable.
SC0BUF	*	*	*	*	*	*	*	*	Set the transmit data and start.

INTTX0 interrupt routine

ACC	← SC0BUF	Read the receiving buffer.
SC0BUF	* * * * * * * *	Set the next transmit data.

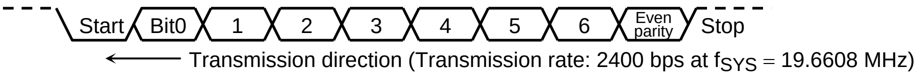
X: Don't care, –: No change

(2) 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.


Start
Bit0
1
2
3
4
5
6
Even parity
Stop
Transmission direction (Transmission rate: 2400 bps at f_{SYS} = 19.6608 MHz)

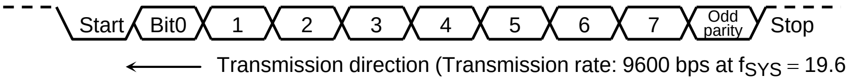
		7	6	5	4	3	2	1	0	
PFCR	←	–	–	–	–	–	–	–	1	Set PF0 to function as the TXD0 pin.
PFFC	←	–	–	–	–	–	–	–	1	
SC0MOD0	←	X	0	–	X	0	1	0	1	Select 7-bit UART mode.
SC0CR	←	X	1	1	X	X	X	0	0	Add even parity.
BR0CR	←	0	0	1	0	1	0	0	0	Set to 2400 bps.
INTES0	←	X	1	0	0	–	–	–	–	Set INTTX0 interrupt to enable and set to level 4.
SC0BUF	←	*	*	*	*	*	*	*	*	Set the transmit data.

X: Don't care, –: No change

(3) 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_{SYS} = 19.6608 MHz)

Main settings

		7	6	5	4	3	2	1	0	
PFCR	←	–	–	–	–	–	–	0	–	Set PF1 to function as the RXD0 pin.
PFFC	←	–	–	–	–	–	–	1	–	
SC0MOD0	←	–	0	1	X	1	0	0	1	Enable receiving in 8-bit UART mode.
SC0CR	←	X	0	1	X	X	X	0	0	Add odd parity.
BR0CR	←	0	0	0	1	1	0	0	0	Set to 9600 bps.
INTES0	←	–	–	–	–	X	1	0	0	Set INTTX0 interrupt to enable and set to level 4.

Interrupt processing

A_{CC} ← SC0CR AND 00011100 } Check for errors
if A_{CC} ≠ 0 then ERROR }

A_{CC} ← SC0BUF Read the received data

X: Don't care, –: No change

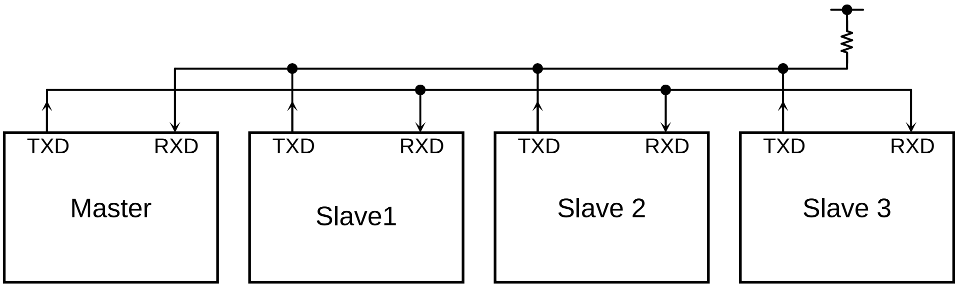
(4) Mode 3 (9-bit UART mode)

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 <TB8>, <RB8> 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.

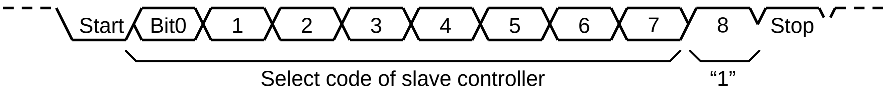


Note: The TXD pin of each slave controller must be in open-drain output mode.

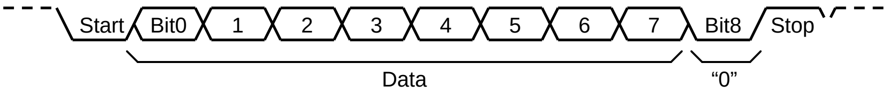
Figure 3.9.29 Serial Link Using Wakeup Function

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.

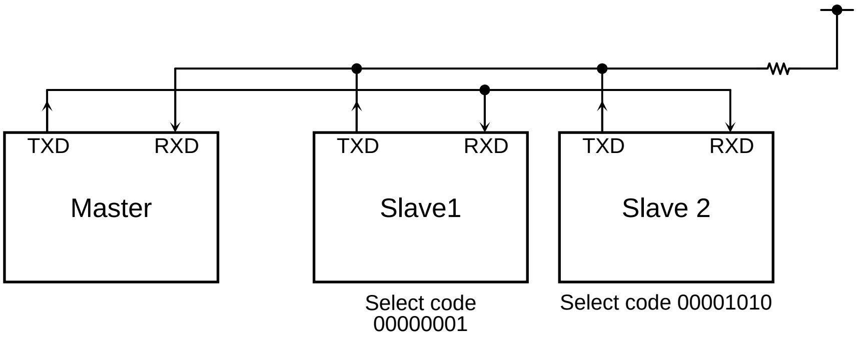


4. Each slave controller receives the above frame. Each controller checks the above select code against its own select code. The controller whose code matches clears its <WU> bit to 0.
5. The master controller transmits data to the specified slave controller (the controller whose SC0MOD0<WU> bit has 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 set 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.



- Setting the master controller

Main

PFCR	←	– – – – – – 0 1	Set PF0 and PF1 to function as the TXD0 and RXD0 pins respectively.
PFFC	←	– – – – – – 1 1	
INTES0	←	X 1 0 0 X 1 0 1	Set INTTX0 to enable, and set interrupt level to level 4. Set INTRX0 to enable, and set interrupt level to 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 – – – – – – –	Set TB8 to 0.
SC0BUF	←	* * * * * * * *	Set the transmission data.

- Setting the slave controller

Main

PFCR	←	– – – – – – 0 1	Set PF1 and PF0 to function as the RXD0 and TXD0 pins respectively.
PFFC	←	– – – – – – 1 1	
INTES0	←	X 1 0 0 X 1 0 1	Set INTRX0 to enable, and set interrupt level to level 4. Set INTRX0 to enable, and set interrupt level to level 5
SC0MOD0	←	0 0 1 1 1 1 1 0	Set to <WU> = “1” in 9-bit UART mode transfer clock f_{SYS}.
INTRX0 interrupt			
A_{CC}	←	SC0BUF	
if A_{CC} =		select code	
Then SC0MOD0	←	– – – 0 – – – –	Clear <WU> to 0

3.9.5 Support for IrDA

SIO0, SIO1 and SIO2 include support for the IrDA 1.0 infrared data communication specification.

Figure 3.9.30 shows the block diagram.

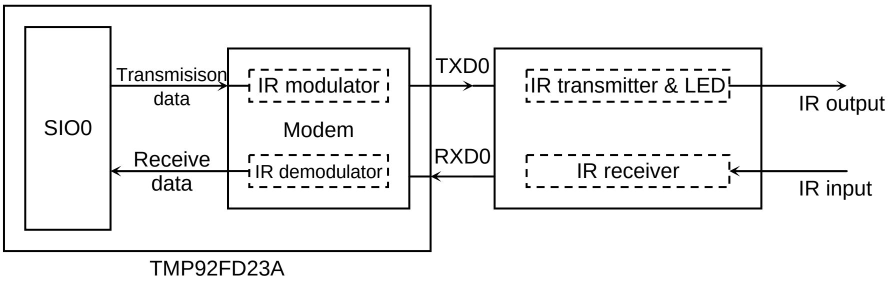


Figure 3.9.30 Block Diagram

(1) Modulation of the transmission data

When the transmit data is 0, the modem outputs 1 to TXD0 pin with either 3/16 or 1/16 times for width of baud rate. The pulse width is selected by the SIR0CR<PLSEL>.

When the transmit data is 1, the modem outputs 0.

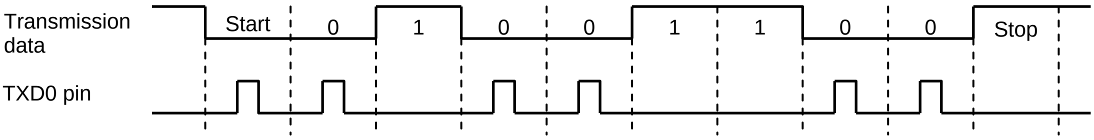


Figure 3.9.31 Transmission Example (SIO0)

(2) Modulation of the receive data

When the receive data is the effective width of pulse “1”, the modem outputs “0” to SIO0. Otherwise the modem outputs “1” to SIO0. The effective pulse width is selected by SIR0CR<SIR0WD3:0>.

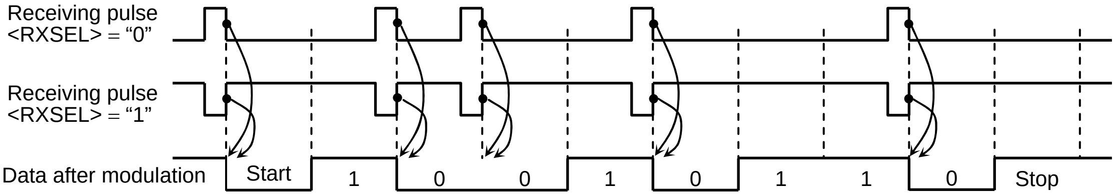


Figure 3.9.32 Receiving Example (SIO0)

(3) Data format

The data format is fixed as follows:

- Data length: 8 bits
- Parity bits: none
- Stop bits: 1 bit

(4) SFR

Figure 3.9.33, Figure 3.9.34 and Figure 3.9.35 shows the control register SIR0CR, SIR1CR and SIR2CR. Set the data SIRxCR during SIOx is stopping. The following example describes how to set this register:

```
1)  SIO setting                    ;  Set the SIO to UART mode.
    ↓
2)  LD       (SIR0CR), 07H         ;  Set the receive data pulse width to 16×+100ns.
3)  LD       (SIR0CR), 37H         ;  TXEN, RXEN Enable the transmission and receiving.
    ↓
4)  Start transmission             ;  The modem operates as follows:
    and receiving for SIO0                    • SIO0 starts transmitting.
                                              • IR receiver starts receiving.
```

(5) Notes

1. Baud rate for IrDA

   When IrDA is operated, set 01 to SC0MOD0<SC1:0> to generate baud rate.

   The setting except above (TA0TRG, f_{IO} and SCLK0 input) can not be used.

2. The pulse width for transmission

   The IrDA 1.0 specification is defined in Table 3.9.4.

Table 3.9.4 Baud Rate and Pulse Width Specifications

Baud Rate	Modulation	Rate Tolerance (% of rate)	Pulse Width (min)	Pulse Width (typ.)	Pulse Width (max)
2.4 Kbps	RZI	±0.87	1.41 μs	78.13 μs	88.55 μs
9.6 Kbps	RZI	±0.87	1.41 μs	19.53 μs	22.13 μs
19.2 Kbps	RZI	±0.87	1.41 μs	9.77 μs	11.07 μs
38.4 Kbps	RZI	±0.87	1.41 μs	4.88 μs	5.96 μs
57.6 Kbps	RZI	±0.87	1.41 μs	3.26 μs	4.34 μs
115.2 Kbps	RZI	±0.87	1.41 μs	1.63 μs	2.23 μs

The pulse width is defined either baud rate T × 3/16 or 1.6 μs (1.6 μs is equal to 3/16 pulse width when baud rate is 115.2 Kbps).

The TMP92FD23A has the function selects the pulse width of transmission either 3/16 or 1/16. But 1/16 pulse width can be selected when the baud rate is equal or less than 38.4 Kbps.

As the same reason, + (16 – K)/16 division function in the baud rate generator of SIO0 can not be used to generate 115.2 Kbps baud rate.

Also when the 38.4 Kbps and 1/16 pulse width, + (16 – K)/16 divisions function cannot be used.

Table 3.9.5 Baud Rate and Pulse Width for (16 – K)/16 Division Function

Pulse Width	Baud Rate					
	115.2 Kbps	57.6 Kbps	38.4 Kbps	19.2 Kbps	9.6 Kbps	2.4 Kbps
T × 3/16	×	○	○	○	○	○
T × 1/16	–	–	×	○	○	○

○: Can be used (16 – K)/16 division function.
×: Cannot be used (16 – K)/16 division function.
–: Cannot be set to 1/16 pulse width.

		7	6	5	4	3	2	1	0
SIR0CR (1207H)	Bit symbol	PLSEL	RXSEL	TXEN	RXEN	SIR0WD3	SIR0WD2	SIR0WD1	SIR0WD0
	Read/Write	R/W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Select transmit pulse width 0: 3/16 1: 1/16	Receive data 0: “H” pulse 1: “L” pulse	Transmit 0: Disable 1: Enable	Receive 0: Disable 1: Enable	Select receive pulse width Set effective pulse width for equal or more than 2x × (value + 1) + 100 ns Can be set: 1 to 14 Can not be set: 0, 15			

Select receive pulse width
Formula: Effective pulse width ≥ 2x × (value + 1) + 100 ns
$x = 1/f_{FPH}$

0000	Cannot be set
0001	Equal or more than 4x + 100 ns
to	
1110	Equal or more than 30x + 100 ns
1111	Can not be set

Receive operation

0	Disable (Received input is ignored)
1	Enable

Transmit operation

0	Disable (Input from SIO is ignored)
1	Enable

Select transmit pulse width

0	3/16
1	1/16

Note: If a pulse width complying with IrDA1.0 standard (1.6 μs min.) can be guaranteed with a low baud rate, setting this bit activetion, resulting in reduced power dissipation.

Figure 3.9.33 IrDA Control Register (for SIO0)

SIR1CR
(120FH)

	7	6	5	4	3	2	1	0
Bit symbol	PLSEL	RXSEL	TXEN	RXEN	SIR1WD3	SIR1WD2	SIR1WD1	SIR1WD0
Read/Write	R/W							
Reset State	0	0	0	0	0	0	0	0
Function	Select transmit pulse width 0: 3/16 1: 1/16	Receive data 0: "H" pulse 1: "L" pulse	Transmit 0: Disable 1: Enable	Receive 0: Disable 1: Enable	Select receive pulse width Set effective pulse width for equal or more than 2x × (value + 1) + 100 ns Can be set: 1 to 14 Can not be set: 0, 15			

Select receive pulse width
Formula: Effective pulse width ≥ 2x × (value +1) +100 ns
$x = 1/f_{FPH}$

0000	Cannot be set
0001	Equal or more than 4x + 100 ns
to	
1110	Equal or more than 30x + 100 ns
1111	Can not be set

Receive operation

0	Disable (Received input is ignored)
1	Enable

Transmit operation

0	Disable (Input from SIO is ignored)
1	Enable

Select transmit pulse width

0	3/16
1	1/16

Note: If a pulse width complying with IrDA1.0 standard (1.6 μs min.) can be guaranteed with a low baud rate, setting this bit activetion, resulting in reduced power dissipation.

Figure 3.9.34 IrDA Control Register 1 (for SIO1)

SIR2CR
(1217H)

	7	6	5	4	3	2	1	0
Bit symbol	PLSEL	RXSEL	TXEN	RXEN	SIR2WD3	SIR2WD2	SIR2WD1	SIR2WD0
Read/Write	R/W							
Reset State	0	0	0	0	0	0	0	0
Function	Select transmit pulse width 0: 3/16 1: 1/16	Receive data 0: "H" pulse 1: "L" pulse	Transmit 0: Disable 1: Enable	Receive 0: Disable 1: Enable	Select receive pulse width Set effective pulse width for equal or more than 2x × (value + 1) + 100 ns Can be set: 1 to 14 Can not be set: 0, 15			

Select receive pulse width
Formula: Effective pulse width ≥ 2x × (value + 1) + 100 ns
$x = 1/f_{SYS}$

0000	Cannot be set
0001	Equal or more than 4x + 100 ns
to	
1110	Equal or more than 30x + 100 ns
1111	Can not be set

Receive operation

0	Disable (Received input is ignored)
1	Enable

Transmit operation

0	Disable (Input from SIO is ignored)
1	Enable

Select transmit pulse width

0	3/16
1	1/16

Note: If a pulse width complying with IrDA1.0 standard (1.6 μs min.) can be guaranteed with a low baud rate, setting this bit activetion, resulting in reduced power dissipation.

Figure 3.9.35 IrDA Control Register 2 (for SIO2)

3.10 Serial Bus Interface (SBI)

The TMP92FD23A has 2-channel serial bus interface which employs a clocked-synchronous 8-bit SIO mode and an I²C bus mode. They are called SBI0 and SBI1.

The serial bus interface is connected to an external device through PN1 (SDA0) and PN2 (SCL0), PN4 (SDA1) and PN5 (SCL1) in the I²C bus mode; and through PN0 (SCK0), PN1 (SO0), PN2 (SI0), PN3 (SCK1), PN4 (SO1) and PN5 (SI1) in the clocked-synchronous 8-bit SIO mode.

Each of the channels can be operated independently. Since both SBI0 and SBI1 channels operate in the same manner, a channel explains only the case of SBI0.

Each pin is specified as follows: (SBI0)

	PNCR<PN2C, PN1C, PN0C>	PNFC<PN2F, PN1F, PN0F>
I^{2}C Bus Mode	11X	11X
Clocked Synchronous 8-Bit SIO Mode	011 010	X11

Each pin is specified as follows: (SBI1)

	PNCR<PN5C, PN4C, PN3C>	PNFC<PN5F, PN4F, PN3F>
I^{2}C Bus Mode	11X	11X
Clocked Synchronous 8-Bit SIO Mode	011 010	X11

X: Don’t care

3.10.1 Configuration

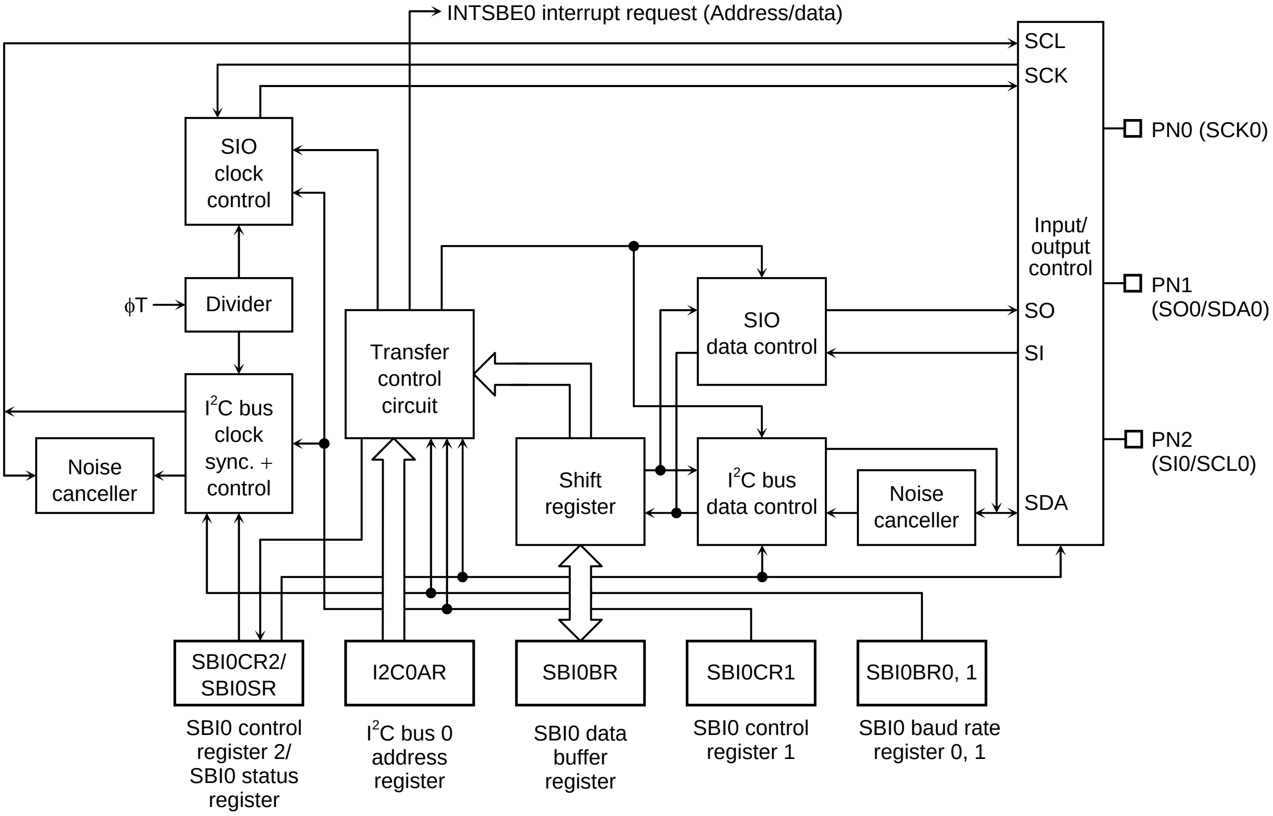


Figure 3.10.1 Serial Bus Interface 0 (SBI0)

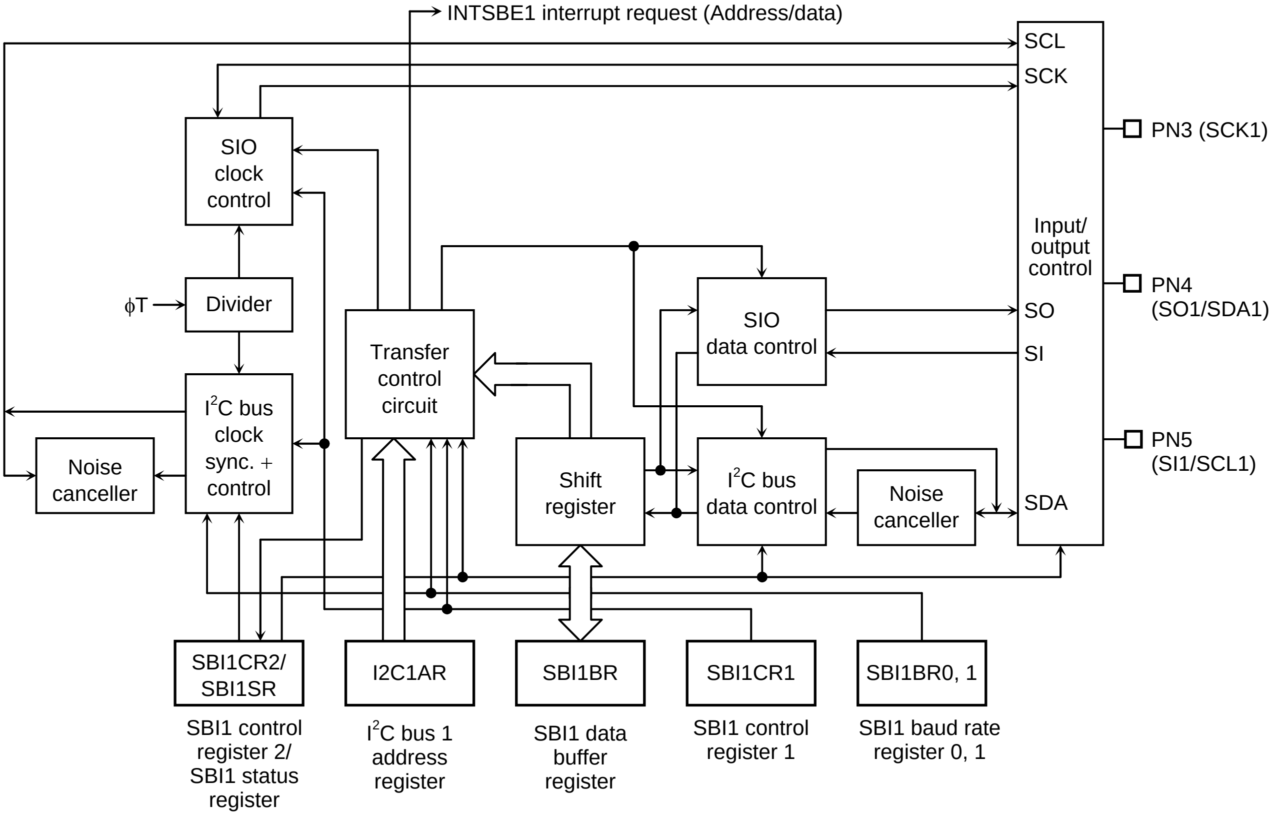


Figure 3.10.2 Serial Bus Interface 0 (SBI1)

3.10.2 Serial Bus Interface (SBI) Control

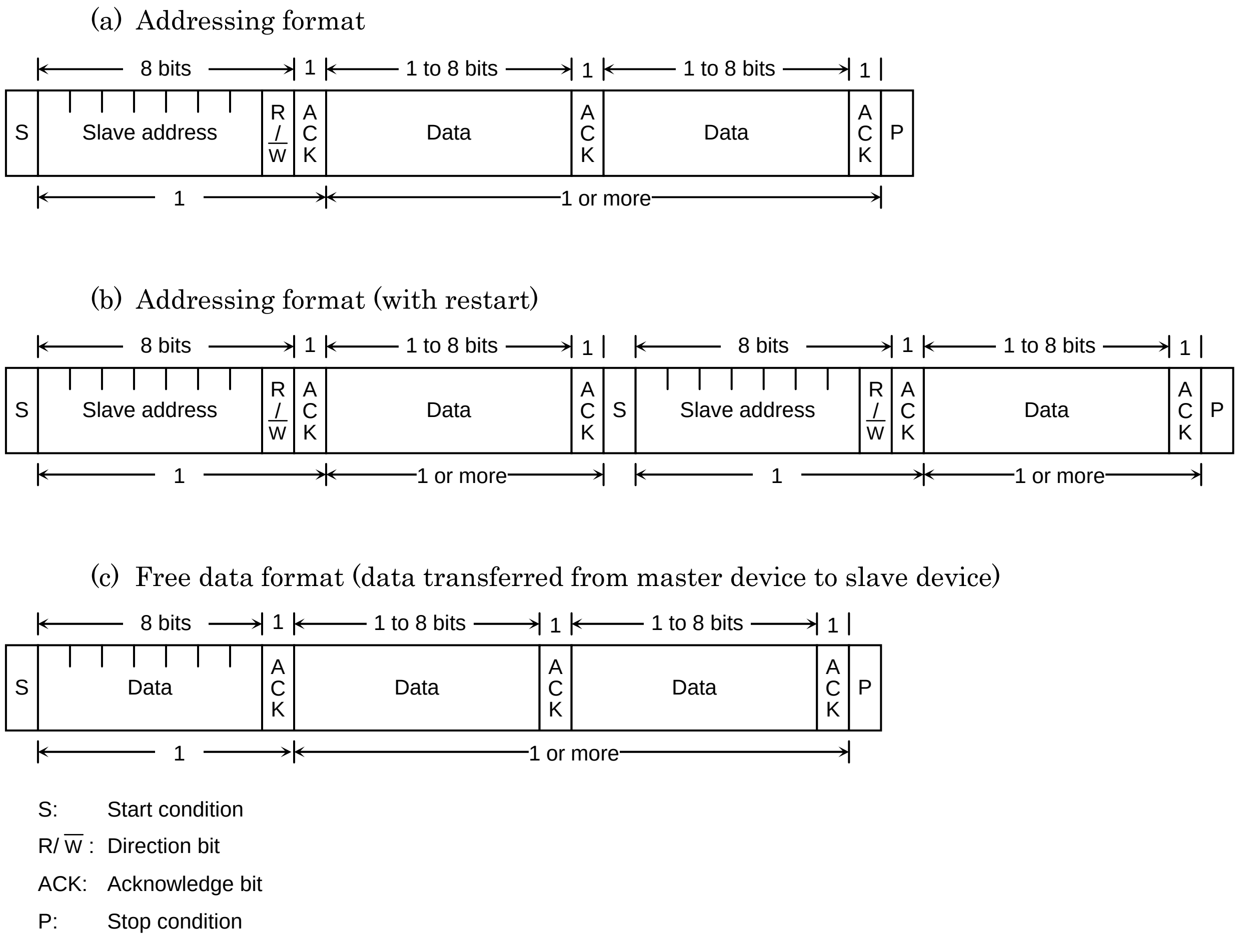
The following registers are used to control the serial bus interface and monitor the operation status.

- Serial bus interface 0 control register 1 (SBI0CR1), (SBI1CR1)
- Serial bus interface 0 control register 2 (SBI0CR2), (SBI1CR2)
- Serial bus interface 0 data buffer register (SBI0DBR), (SBI1DBR)
- I²C bus 0 address register (I2C0AR), (I2C1AR)
- Serial bus interface 0 status register (SBI0SR), (SBI1SR)
- Serial bus interface 0 baud rate register 0 (SBI0BR0), (SBI1BR0)
- Serial bus interface 0 baud rate register 1 (SBI0BR1), (SBI1BR1)

The above registers differ depending on a mode to be used. Refer to section 3.10.4 "I²C Bus Mode Control Register" 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)

Figure 3.10.3 Data Format in the I²C Bus Mode

3.10.4 I^{2}C Bus Mode Control Register

The following registers are used to control and monitor the operation status when using the serial bus interface (SBI0, SBI1) in the I^{2}C bus mode.

Serial Bus Interface 0 Control Register 1

		7	6	5	4	3	2	1	0
SBI0CR1 (1240H)	Bit symbol	BC2	BC1	BC0	ACK		SCK2	SCK1	SCK0/ SWRMON
	Read/Write	W			R/W		W		R/W
Prohibit read-modify-write	Reset State	0	0	0	0		0	0	0/1 (Note 3)
	Function	Number of transferred bits (Note 1)			Acknowledge edge mode specification 0: Not generate 1:Generate		Internal serial clock selection and software reset monitor (Note 2)		

Internal serial clock selection <SCK2:0> at write

000	n = 5	– (Note 4)	
001	n = 6	– (Note 4)	System clock: fSYS
010	n = 7	– (Note 4)	f_{SYS} = 20 MHz
011	n = 8	– (Note 4)	(internal SCL output)
100	n = 9	76.9 kHz	
101	n = 10	38.8 kHz	$f_{SCL} = \dfrac{f_{SYS} \times 2}{2^n + 8}$ [Hz]
110	n = 11	19.5 kHz	
111	Reserved	(Reserved)	

Software reset state monitor <SWRMON> at Read

0	During software reset
1	Initial data

Acknowledge mode specification

0	Not generate clock pulse for acknowledge signal
1	Generate clock pulse for acknowledge signal

Number of bits transferred

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

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

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

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

Note 4: This I^{2}C bus circuit does not support Fast mode, it supports standard mode only. Although the I^{2}C bus circuit itself allows the setting of a baud rate over 100 kbps, the compliance with the I^{2}C specification is not guaranteed in that case.

Figure 3.10.4 Registers for the I^{2}C Bus Mode (SBI0)

Serial Bus Interface 1 Control Register 1

		7	6	5	4	3	2	1	0
SBI1CR1 (1248H)	Bit symbol	BC2	BC1	BC0	ACK		SCK2	SCK1	SCK0/ SWRMON
	Read/Write	W			R/W		W		R/W
Prohibit read-modify- write	Reset State	0	0	0	0		0	0	0/1 (Note 3)
	Function	Number of transferred bits (Note 1)			Acknowledge edge mode specification 0: Not generate 1: Generate		Internal serial clock selection and software reset monitor (Note 2)		

Internal serial clock selection <SCK2:0> at write

000	n = 5	– (Note 4)	System clock: f_{SYS} f_{SYS} = 20 MHz (internal SCL output) $f_{SCL} = \frac{f_{SYS}\times 2}{2^n + 8}$ [Hz]
001	n = 6	– (Note 4)	
010	n = 7	– (Note 4)	
011	n = 8	– (Note 4)	
100	n = 9	76.9 kHz	
101	n = 10	38.8 kHz	
110	n = 11	19.5 kHz	
111	Reserved	(Reserved)	

Software reset state monitor <SWRMON> at Read

0	During software reset
1	Initial data

Acknowledge mode specification

0	Not generate clock pulse for acknowledge signal
1	Generate clock pulse for acknowledge signal

Number of bits transferred

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

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

Note 2: For the frequency of the SCL pin clock, see 3.10.5 (3) “Serial clock”.

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

Note 4: This I^2C bus circuit does not support Fast mode, it supports standard mode only. Although the I^2C bus circuit itself allows the setting of a baud rate over 100 kbps, the compliance with the I^2C specification is not guaranteed in that case.

Figure 3.10.5 Registers for the I^2C Bus Mode (SBI1)

Serial Bus Interface 0 Control Register 2

SBI0CR2
(1243H)

Read-modify-write instruction is prohibited.

	7	6	5	4	3	2	1	0
Bit symbol	MST	TRX	BB	PIN	SBIM1	SBIM0	SWRST1	SWRST0
Read/Write	W				W (Note 1)		W (Note 1)	
Reset State	0	0	0	1	0	0	0	0
Function	Master/ slave selection	Transmitter /receiver selection	Start/stop condition generation	Cancel INTSBE0 interrupt request	Serial bus interface operating mode selection (Note 2) 00: Port mode 01: SIO mode 10: I^2C bus mode 11: (Reserved)		Software reset generate write “10” and “01”, then an internal software reset signal is generated.	

Serial bus interface operating mode selection (Note 2)

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

INTSBE0 interrupt request

0	–
1	Cancel interrupt request

Start/stop condition generation

0	Generates the stop condition
1	Generates the start condition

Transmitter/receiver selection

0	Receiver
1	Transmitter

Master/slave selection

0	Slave
1	Master

Note 1: Reading this register function as SBI0SR register.

Note 2: Switch a mode to port mode after confirming that the bus is free.

Switch a mode between I^2C bus mode and clocked-synchronous 8-bit SIO mode after confirming that input signals via port are high level.

Figure 3.10.6 Registers for the I^2C Bus Mode (SBI0)

Serial Bus Interface 1 Control Register 2

		7	6	5	4	3	2	1	0
SBI1CR2	Bit symbol	MST	TRX	BB	PIN	SBIM1	SBIM0	SWRST1	SWRST0
(124BH)	Read/Write	W				W (Note 1)		W (Note 1)	
	Reset State	0	0	0	1	0	0	0	0
Read-modify-write instruction is prohibited.	Function	Master/ slave selection	Transmitter /receiver selection	Start/stop condition generation	Cancel INTSBE1 interrupt request	Serial bus interface operating mode selection (Note 2) 00: Port mode 01: SIO mode 10: I^2C bus mode 11: (Reserved)		Software reset generate write “10” and “01”, then an internal software reset signal is generated.	

Serial bus interface operating mode selection (Note 2)

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

INTSBE1 interrupt request

0	–
1	Cancel interrupt request

Start/stop condition generation

0	Generates the stop condition
1	Generates the start condition

Transmitter/receiver selection

0	Receiver
1	Transmitter

Master/slave selection

0	Slave
1	Master

Note 1: Reading this register function as SBI1SR register.

Note 2: Switch a mode to port mode after confirming that the bus is free.

Switch a mode between I^2C bus mode and clocked-synchronous 8-bit SIO mode after confirming that input signals via port are high level.

Figure 3.10.7 Registers for the I^2C Bus Mode (SBI1)

Serial Bus Interface 0 Status Register

		7	6	5	4	3	2	1	0
SBI0SR	Bit symbol	MST	TRX	BB	PIN	AL	AAS	AD0	LRB
(1243H)	Read/Write	R							
	Reset State	0	0	0	1	0	0	0	0
Read-modify-write instruction is prohibited.	Function	Master/ slave status selection monitor	Transmitter /receiver status selection monitor	I²C bus status monitor	INTSBE0 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”

Last received bit monitor

0	Last received bit was “0”
1	Last received bit was “1”

GENERAL CALL detection monitor

0	Undetected
1	GENERAL CALL detected

Slave address match detection monitor

0	Undetected
1	Slave address match or GENERAL CALL detected

Arbitration lost detection monitor

0	–
1	Arbitration lost

INTSBE0 interrupt request monitor

0	Interrupt requested
1	Interrupt canceled

I²C bus status monitor

0	Free
1	Busy

Transmitter/receiver status monitor

0	Receiver
1	Transmitter

Master/slave status monitor

0	Slave
1	Master

Note: Writing in this register functions as SBI0CR2.

Figure 3.10.8 Registers for the I²C Bus Mode (SBI0)

Serial Bus Interface 1 Status Register

SBI1SR
(124BH)

Read-modify-write instruction is prohibited.

	7	6	5	4	3	2	1	0
Bit symbol	MST	TRX	BB	PIN	AL	AAS	AD0	LRB
Read/Write	R							
Reset State	0	0	0	1	0	0	0	0
Function	Master/ slave status selection monitor	Transmitter /receiver status selection monitor	I2C bus status monitor	INTSBE1 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”

Last received bit monitor

0	Last received bit was “0”
1	Last received bit was “1”

GENERAL CALL detection monitor

0	Undetected
1	GENERAL CALL detected

Slave address match detection monitor

0	Undetected
1	Slave address match or GENERAL CALL detected

Arbitration lost detection monitor

0	–
1	Arbitration lost

INTSBE1 interrupt request monitor

0	Interrupt requested
1	Interrupt canceled

I^2C bus status monitor

0	Free
1	Busy

Transmitter/receiver status monitor

0	Receiver
1	Transmitter

Master/slave status monitor

0	Slave
1	Master

Note: Writing in this register functions as SBI1CR2.

Figure 3.10.9 Registers for the I^2C Bus Mode (SBI1)

Serial Bus Interface 0 Baud Rate Register 0

SBI0BR0 (1244H)

Read-modify-write instruction is prohibited.

	7	6	5	4	3	2	1	0
Bit symbol	–	I2SBI0						
Read/Write	W	R/W						
Reset State	0	0						
Function	Always write “0”.	IDLE2 0: Stop 1: Run						

Operation during IDLE2 mode

0	Stop
1	Operation

Serial Bus Interface 0 Baud Rate Register 1

SBI0BR1 (1245H)

Read-modify-write instruction is prohibited.

	7	6	5	4	3	2	1	0
Bit symbol	P4EN	–						
Read/Write	W							
Reset State	0	0						
Function	Internal clock 0: Stop 1: Run	Always write “0”.						

Baud rate clock control

0	Stop
1	Operate

Serial Bus Interface 0 Data Buffer Register

SBI0DBR (1241H)

	7	6	5	4	3	2	1	0
Bit symbol	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Read/Write	R (Receiving)/W (Transmission)							
Reset State	Undefined							

Read-modify-write instruction is prohibited.

Note 1: When writing transmission data, start from the MSB (Bit7). Receiving data is placed from LSB (Bit0).

Note 2: SBI0DBR can’t be read the written data. Therefore read-modify-write instruction (e.g., “BIT” instruction) is prohibited.

Note 3: Written data in SBI0DBR is cleared by INTSBE0 signal.

I^{2}C Bus Address Register

I2C0AR (1242H)

Read-modify-write instruction is prohibited.

	7	6	5	4	3	2	1	0
Bit symbol	SA6	SA5	SA4	SA3	SA2	SA1	SA0	ALS
Read/Write	W							
Reset State	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

Figure3.10.10 Registers for the I^{2}C Bus Mode (SBI0)

Serial Bus Interface 1 Baud Rate Register 0

SBI1BR0
(124CH)
Read-modify-write instruction is prohibited.

	7	6	5	4	3	2	1	0
Bit symbol	–	I2SBI0						
Read/Write	W	R/W						
Reset State	0	0						
Function	Always write “0”.	IDLE2 0: Stop 1: Run						

Operation during IDLE2 mode

0	Stop
1	Operation

Serial Bus Interface 1 Baud Rate Register 1

SBI1BR1
(124DH)
Read-modify-write instruction is prohibited.

	7	6	5	4	3	2	1	0
Bit symbol	P4EN	–						
Read/Write	W							
Reset State	0	0						
Function	Internal clock 0: Stop 1: Run	Always write “0”.						

Baud rate clock control

0	Stop
1	Operate

Serial Bus Interface 1 Data Buffer Register

SBI1DBR
(1249H)

	7	6	5	4	3	2	1	0
Bit symbol	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Read/Write	R (Receiving)/W (Transmission)							
Reset State	Undefined							

Read-modify-write instruction is prohibited.

Note 1: When writing transmission data, start from the MSB (Bit7). Receiving data is placed from LSB (Bit0).

Note 2: SBI1DBR can’t be read the written data. Therefore read-modify-write instruction (e.g., “BIT” instruction) is prohibited.

Note 3: Written data in SBI1DBR is cleared by INTSBE1 signal.

I^{2}C Bus Address Register

I2C1AR
(124AH)
Read-modify-write instruction is prohibited.

	7	6	5	4	3	2	1	0
Bit symbol	SA6	SA5	SA4	SA3	SA2	SA1	SA0	ALS
Read/Write	W							
Reset State	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

Figure 3.10.11 Registers for the I^{2}C Bus Mode (SBI1)

3.10.5 Control in I^{2}C Bus Mode

(1) Acknowledge mode specification

Set the SBI0CR1<ACK> to "1" for operation in the acknowledge mode. The TMP92FD23A 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 TMP92FD23A does not generate a clock pulse for the acknowledge signal when operating in the master mode.

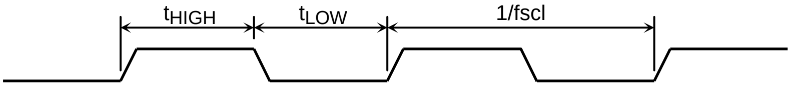
(2) Number of transfer bits

Since the SBI0CR1<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

The SBI0CR1<SCK2:0> is used to specify the maximum transfer frequency for output on the SCL pin in the master mode. Set the baud rates, which have been calculated according to the formula below, to meet the specifications of the I^{2}C bus, such as the smallest pulse width of t_{LOW}.



$$t_{LOW} = 2^{n-1}/f_{SBI}$$
$$t_{HIGH} = 2^{n-1}/f_{SBI} + 8/f_{SBI}$$
$$fscl = 1/(t_{LOW} + t_{HIGH})$$
$$= \frac{f_{SBI}}{2^n + 8}$$

SBI0CR1<SCK2:0>	n
000	5
001	6
010	7
011	8
100	9
101	10
110	11

Note1: f_{SBI} shows f_{SYS}.

Note2: In a setup of prescaler of SYSCR0, the fc/16 mode cannot be used at the time of SBI circuit use.

Figure 3.10.12 Clock Source

2. Clock synchronization

In the I^2C 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 TMP92FD23A has a clock synchronization function for normal data transfer even when more than one master exists on the bus.

The example explains the clock synchronization procedures when two masters simultaneously exist on a bus.

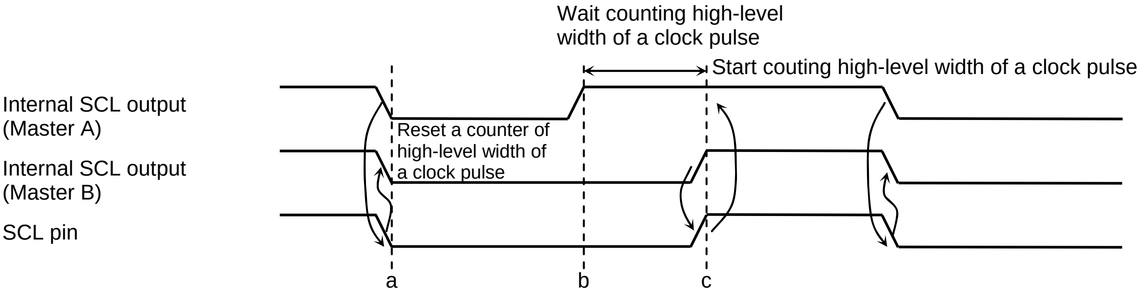

Wait counting high-level width of a clock pulse
Start couting high-level width of a clock pulse
Internal SCL output (Master A)
Internal SCL output (Master B)
SCL pin
Reset a counter of high-level width of a clock pulse
a
b
c

Figure 3.10.13 Clock Synchronization

As master A pulls down the internal SCL output to the low level at point “a”, the SCL line of the bus becomes the low level. After detecting this situation, master B resets a counter of high-level width of an own clock pulse and sets the internal SCL output to the low level.

Master A finishes counting low-level width of an own clock pulse at point “b” and sets the internal SCL output to the high level. Since master B holds the SCL line of the bus at the low level, master A waits for counting high-level width of an own clock pulse. After master B finishes 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 this device is to be used as a slave device, set the slave address <SA6:0> and <ALS> in I2C0AR.

Clear the <ALS> to “0” for the address recognition mode.

(5) Master/slave selection

Set the SBI0CR2<MST> to “1” for operating the TMP92FD23A as a master device. Clear the SBI0CR2<MST> to “0” for operation as a slave device. The <MST> is cleared to “0” by the hardware after a stop condition on the bus is detected or arbitration is lost.

(6) Transmitter/receiver selection

Set the SBI0CR2<TRX> to “1” for operating the TMP92FD23A as a transmitter. Clear the <TRX> to “0” for operation as a receiver. In slave mode, when transfer data in addressing format, when received slave address is same value with setting value to I2C0AR, or GENERAL CALL is received (All 8-bit data are “0” after a start condition), the <TRX> is set to “1” by the hardware if the direction bit (R/$\overline{\mathrm{W}}$) sent from the master device is “1”, and <TRX> is cleared to “0” by the hardware if the bit is “0”.

In the master mode, after an acknowledge signal is returned from the slave device, the <TRX> is cleared to “0” by the hardware if a transmitted direction bit is “1”, and is set to “1” by the hardware if it is “0”. When an acknowledge signal is not returned, the current condition is maintained.

The <TRX> is cleared to “0” by the hardware after a stop condition on the bus is detected or arbitration is lost.

(7) Start/stop condition generation

When the SBI0SR<BB> = “0”, slave address and direction bit which are set to SBI0DBR is output on the bus after generating a start condition by writing “1111” to the SBI0CR2<MST, TRX, BB, PIN>. It is necessary to set transmitted data to the data buffer register (SBI0DBR) and set “1” to the <ACK> beforehand.

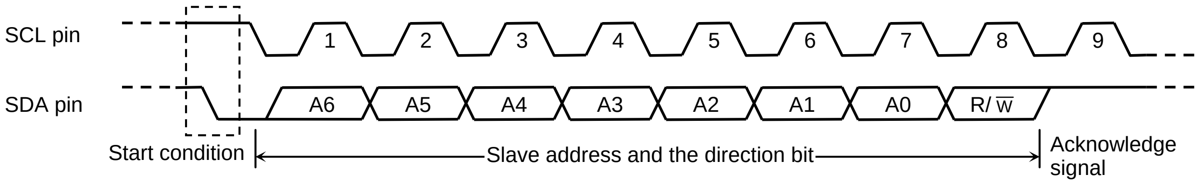


Figure 3.10.14 Start Condition Generation and Slave Address Generation

When the SBI0SR<BB> = “1”, the sequence for generating a stop condition can be initiated by writing “111” to the SBI0CR2<MST, TRX, PIN> and writing “0” to the SBI0CR2<BB>. Do not modify the contents of the SBI0CR2<MST, TRX, BB, PIN> until a stop condition has been generated on the bus.

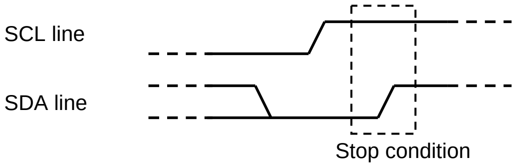


Figure 3.10.15 Stop Condition Generation

The state of the bus can be ascertained by reading the contents of SBI0SR<BB>. SBI0SR<BB> will be set to 1 (Bus busy status) if a start condition has been detected on the bus, and will be cleared to 0 if a stop condition has been detected (Bus free status).

In addition, since there is a restrictions matter about stop condition generating in master mode, please refer to 3.10.6. (4) “Stop condition generation”.

(8) Interrupt service requests and interrupt cancellation

When a serial bus interface interrupt request 0 (INTSBE0) occurs, the SBI0SR2 <PIN> is cleared to “0”. During the time that the SBI0SR2<PIN> is “0”, the SCL line is pulled down to the low level.

The <PIN> is cleared to “0” when end of transmission or receiving 1 word of data. And when writing data to SBI0DBR or reading data from SBI0DBR, <PIN> is set to “1”.

The time from the <PIN> being set to “1” until the SCL line is released takes t_{LOW}.

In the address recognition mode (<ALS> = “0”), <PIN> is cleared to “0” when the received slave address is the same as the value set at the I2C0AR or when a GENERAL CALL is received (All 8-bit data are “0” after a start condition). Although SBI0CR2<PIN> can be set to “1” by the program, the <PIN> is not clear it to “0” when it is programmed “0”.

(9) Serial bus interface operation mode selection

The SBI0CR2<SBIM1:0> is used to specify the serial bus interface operation mode.

Set the SBI0CR2<SBIM1:0> to “10” when the device is to be used in I^{2}C 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 I^{2}C bus mode, a bus arbitration procedure has been implemented in order to guarantee the integrity of transferred data.

Data on the SDA pin is used for I^{2}C 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 pin of the bus is wire-AND and the SDA pin is pulled down to the low level by master A. When the SCL pin of the bus is pulled up at point “b”, the slave device reads the data on the SDA pin, that is, data in master A. Data transmitted from master B becomes invalid. The master B state is known as “ARBITRATION LOST”. 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.

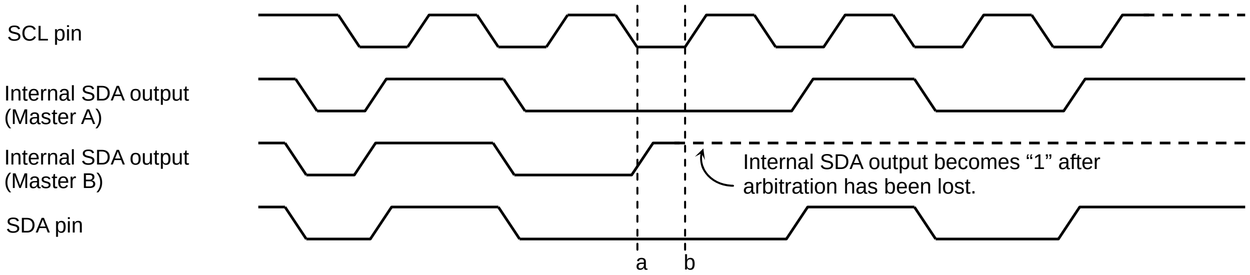

SCL pin
Internal SDA output (Master A)
Internal SDA output (Master B)
SDA pin
Internal SDA output becomes “1” after arbitration has been lost.
a
b

Figure 3.10.16 Arbitration Lost

The TMP92FD23A 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 SBI0SR<AL> is set to "1".

When SBI0SR<AL> is set to "1", SBI0SR<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".

SBI0SR <AL> is cleared to "0" when data is written to or read from SBI0DBR or when data is written to SBI0CR2.

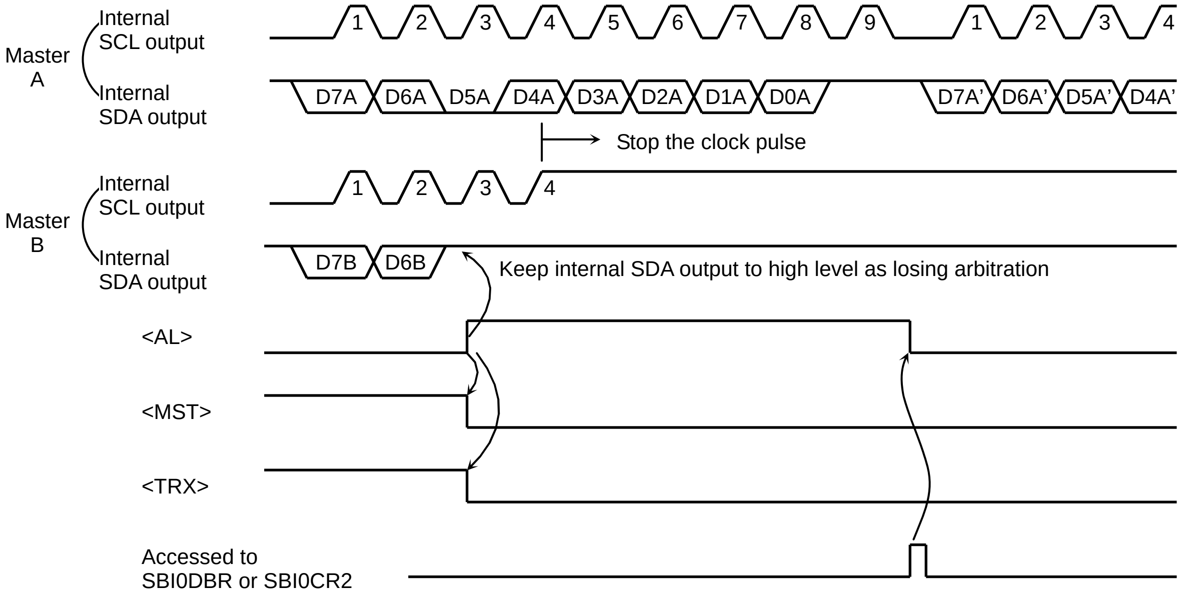


Figure3.10.17 Example of a Master Device B (D7A = D7B, D6A = D6B)

(11) Slave address match detection monitor

SBI0SR<AAS> operates following in during slave mode; In address recognition mode (e.g., when I2C0AR<ALS> = "0"), when received GENERAL CALL or same slave address with value set to I2C0AR, SBI0SR<AAS> is set to "1". When <ALS> = "1", SBI0SR<AAS> is set to "1" after the first word of data has been received. SBI0SR<AAS> is cleared to "0" when data is written to SBI0DBR or read from SBI0DBR.

(12) GENERAL CALL detection monitor

SBI0SR<AD0> operates following in during slave mode; when received GENERAL CALL (all 8-bit data is "0", after a start condition), SBI0SR<AD0> is set to "1". And SBI0SR<AD0> is cleared to "0" when a start condition or stop condition on the bus is detected.

(13) Last received bit monitor

The value on the SDA line detected on the rising edge of the SCL line is stored in the SBI0SR<LRB>. In the acknowledge mode, immediately after an INTSBE0 interrupt request has been generated, an acknowledge signal is read by reading the contents of the SBI0SR<LRB>.

(14) Software reset function

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

When write first “10” next “01” to SBI0CR2<SWRST1:0>, reset signal is inputted to serial bus interface circuit, and circuit is initialized. All command registers except SBI0CR2<SBIM1:0> and status flag except SBI0CR2<SBIM1:0> are initialized to value of just after reset. SBI0CR1<SWRMON> is set to “1” automatically when completed initialization of serial bus interface.

(15) Serial bus interface data buffer register (SBI0DBR)

The received data can be read and transmission data can be written by reading or writing SBI0DBR.

In the master mode, after the slave address and the direction bit are set in this register, the start condition is generated.

(16) I^{2}C bus address register (I2C0AR)

I2C0AR<SA6:0> is used to set the slave address when the TMP92FD23A functions as a slave device.

The slave address outputted from the master device is recognized by setting the I2C0AR<ALS> to “0”. And, the data format becomes the addressing format. When set <ALS> to “1”, the slave address is not recognized, the data format becomes the free data format.

(17) Baud rate register (SBI0BR1)

Write “1” to baud rate circuit control register SBI0BR1<P4EN> before using I^{2}C bus.

(18) Setting register for IDLE2 mode operation (SBI0BR0)

SBI0BR0<I2SBI0> is the register setting operation/stop during IDLE2 mode. Therefore, setting <I2SBI0> is necessary before the HALT instruction is executed.

3.10.6 Data Transfer in I^{2}C Bus Mode

(1) Device initialization

In first, set the SBI0BR1<P4EN>, SBI0CR1<ACK, SCK2:0>. Set SBI0BR1<P4EN> to "1" and clear bits 7 to 5 and 3 in the SBI0CR1 to "0".

Next, set a slave address <SA6:0> and the <ALS> (<ALS> = "0" when an addressing format) to the I2C0AR.

And, write "000" to SBI0CR2<MST, TRX, BB>, "1" to <PIN>, "10" to <SBIM1:0> and "00" to <SWRST1:0>. Set initialization status to slave receiver mode by this setting.

(2) Start condition generation and slave address generation

1. Master mode

In the master mode, the start condition and the slave address are generated as follows.

In first, check a bus free status (when SBI0SR<BB> = "0").
Set the SBI0CR1<ACK> to "1" (Acknowledge mode) and specify a slave address and a direction bit to be transmitted to the SBI0DBR.

When SBI0SR<BB> = "0", the start condition are generated by writing "1111" to SBI0CR2<MST, TRX, BB, PIN>. Subsequently to the start condition, nine clocks are output from the SCL pin. While eight clocks are output, the slave address and the direction bit which are set to the SBI0DBR. At the 9th clock, the SDA line is released and the acknowledge signal is received from the slave device.

An INTSBE0 interrupt request generate at the falling edge of the 9th clock. The <PIN> is cleared to "0". In the master mode, the SCL pin is pulled down to the low level while <PIN> is "0". When an interrupt request is generated, the <TRX> is changed according to the direction bit only when an acknowledge signal is returned from the slave device.

2. Slave mode

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

After the start condition is received from the master device, while eight clocks are output from the SCL pin, the slave address and the direction bit that are output from the master device are received.

When a GENERAL CALL or the same address as the slave address set in I2C0AR is received, the SDA line is pulled down to the low level at the 9th clock, and the acknowledge signal is output.

An INTSBE0 interrupt request is generated on the falling edge of the 9th clock. The <PIN> is cleared to "0". In slave mode the SCL line is pulled down to the low level while the <PIN> = "0".

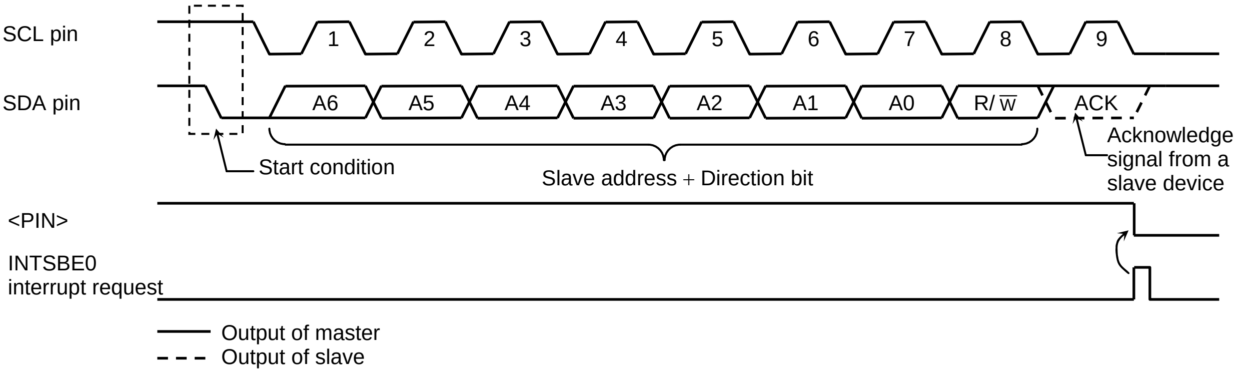


Figure3.10.18 Start Condition Generation and Slave Address Transfer

(3) 1-word data transfer

Check the <MST> by the INTSBE0 interrupt process after the 1-word data transfer is completed, and determine whether the mode is a master or slave.

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

Check the <TRX> and determine whether the mode is a transmitter or receiver.

When the <TRX> = "1" (Transmitter mode)

Check the <LRB>. When <LRB> is "1", a receiver does not request data. Implement the process to generate a stop condition (Refer to (4)) and terminate data transfer.

When the <LRB> is "0", the receiver is requests new data. When the next transmitted data is 8 bits, write the transmitted data to SBI0DBR. When the next transmitted data is other than 8 bits, set the <BC2:0> <ACK> and write the transmitted data to SBI0DBR. After written the data, <PIN> becomes "1", a serial clock pulse is generated for transferring a new 1-word of data from the SCL0 pin, and then the 1-word data is transmitted. After the data is transmitted, an INTSBE0 interrupt request generates. The <PIN> becomes "0" and the SCL0 line is pulled down to the low level. If the data to be transferred is more than one word in length, repeat the procedure from the <LRB> checking above.

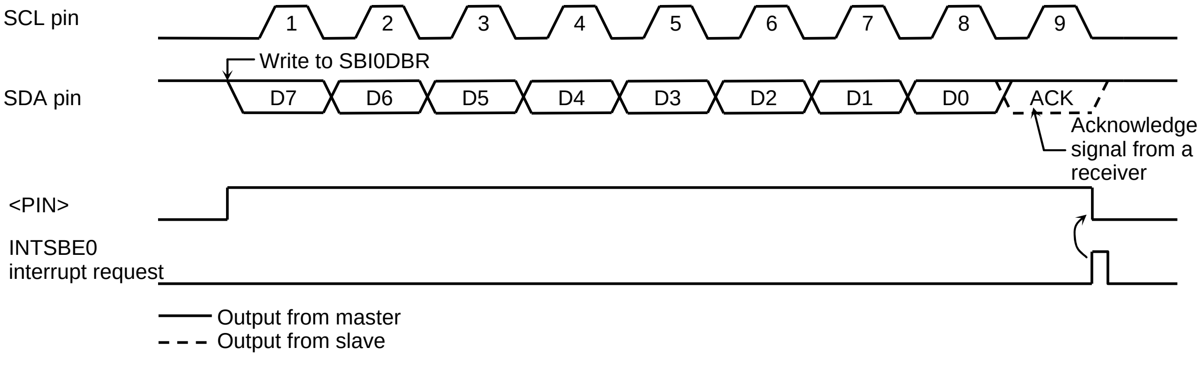


Figure3.10.19 Example in which <BC2:0> = "000" and <ACK> = "1" in Transmitter Mode

When the <TRX> is “0” (Receiver mode)

When the next transmitted data is other than 8 bits, set <BC2:0> <ACK> and read the received data from SBI0DBR to release the SCL0 line (Data which is read immediately after a slave address is sent is undefined). After the data is read, <PIN> becomes “1”. Serial clock pulse for transferring new 1 word of data is defined SCL and outputs “L” level from SDA0 pin with acknowledge timing.

An INTSBE0 interrupt request then generates and the <PIN> becomes “0”, Then the TMP92FD23A pulls down the SCL pin to the low level. The TMP92FD23A outputs a clock pulse for 1 word of data transfer and the acknowledge signal each time that received data is read from the SBI0DBR.

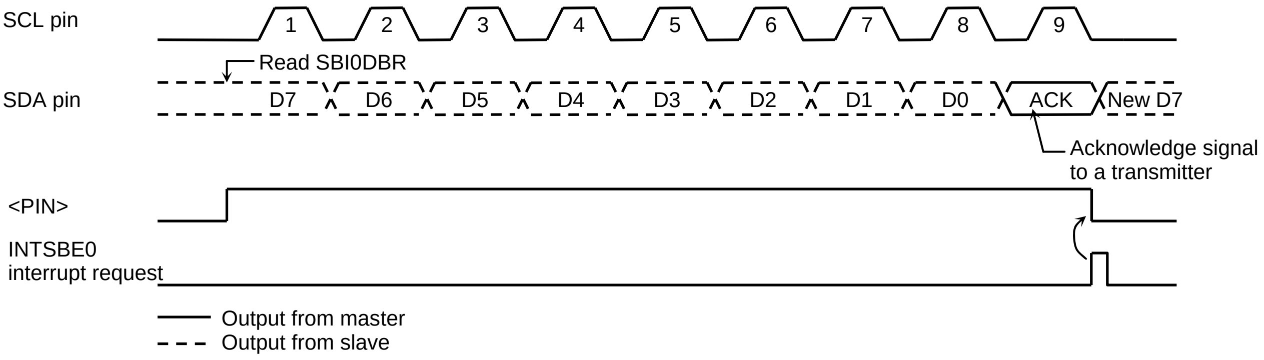


Figure3.10.20 Example of when <BC2:0> = “000”, <ACK> = “1” in Receiver Mode

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 TMP92FD23A generates a clock pulse for a 1-bit data transfer. Since the master device is a receiver, the SDA0 line on the bus remains high. The transmitter receives the high signal as an ACK signal. The receiver indicates to the transmitter that the data transfer is completed.

After the one data bit has been received and an interrupt request has been generated, the TMP92FD23A generates a stop condition (See section (4)) and terminates data transfer.

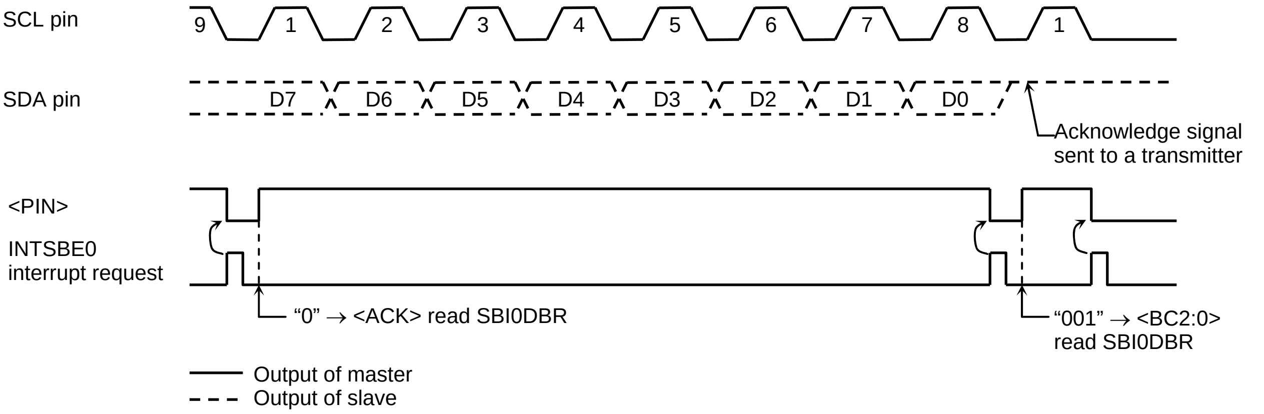


Figure3.10.21 Termination of Data Transfer in Master Receiver Mode

2. When the <MST> is "0" (Slave mode)

   In the slave mode the TMP92FD23A operates either in normal slave mode or in slave mode after losing arbitration.

   In the slave mode, an INTSBE0 interrupt request generate when the TMP92FD23A receives a slave address or a GENERAL CALL from the master device, or when a GENERAL CALL is received and data transfer is completed, or after matching received address. In the master mode, the TMP92FD23A operates in a slave mode if it losing arbitration. An INTSBE0 interrupt request is generated when a word data transfer terminates after losing arbitration. When an INTSBE0 interrupt request is generated the <PIN> is cleared to "0" and the SCL pin is pulled down to the low level. Either reading/writing from/to the SBI0DBR or setting the <PIN> to "1" will release the SCL pin after taking tLOW time.

   Check the SBI0SR<AL>, <TRX>, <AAS>, and <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 TMP92FD23A detects arbitration lost 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 of single word to <BC2:0>, and write the transmit data to SBI0DBR.
	0	1	0	In slave receiver mode, the TMP92FD23A receives a slave address for which the value of the direction bit sent from the master is “1”.	
		0	0	In salve transmitter mode, transmission of data of single word is terminated.	Check the <LRB>, If <LRB> is set to “1”, set <PIN> to “1”, reset “0” to <TRX> and release the bus for the receiver no request next data. If <LRB> was cleared to “0”, set bit number of single word to <BC2:0> and write the transmit data to SBI0DBR for the receiver requests next data.
0	1	1	1/0	The TMP92FD23A detects arbitration lost 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 SBI0DBR for setting the <PIN> to “1” (Reading dummy data) or set the <PIN> to “1”.
		0	0	The TMP92FD23A detects arbitration lost when transmitting a slave address or data, and transfer of word terminates.	
	0	1	1/0	In slave receiver mode the TMP92FD23A receives a slave address or GENERAL CALL for which the value of the direction bit sent from the master is “0”.	
		0	1/0	In slave receiver mode the TMP92FD23A terminates receiving word data.	Set bit number of single word to <BC2:0>, and read the receiving data from SBI0DBR.

(4) Stop condition generation

When SBI0SR<BB> = “1”, the sequence for generating a stop condition is started by writing “111” to SBI0CR2<MST, TRX, PIN> and “0” to SBI0CR2<BB>. Do not modify the contents of SBI0CR2<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 TMP92FD23A generates a stop condition when the other device has released the SCL line and SDA0 pin rising.

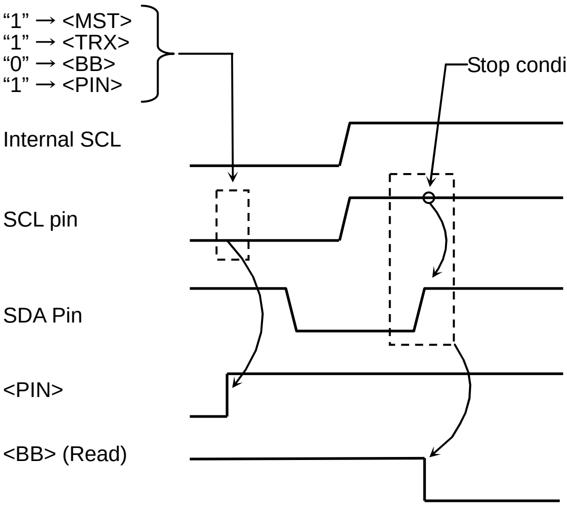


Figure3.10.22 Stop Condition Generation (Single master)

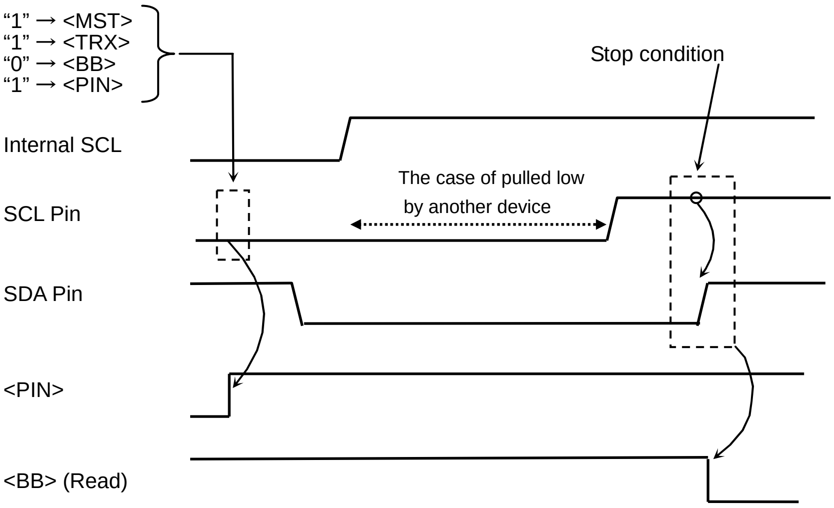


Figure3.10.23 Stop Condition Generation (Multi master)

(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 this device is in the master mode.

Clear the SBI0CR2<MST, TRX, BB> to "000" and set the SBI0CR2<PIN> to "1" to release the bus. The SDA0 line remains the high level and the SCL0 pin is released. Since a stop condition is not generated on the bus, other devices assume the bus to be in a busy state. Check the SBI0SR<BB> until it becomes "0" to check that the SCL0 pin of this device is released. 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 stays in a free state, generate a start condition with procedure described in (2).

In order to meet setup time 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.

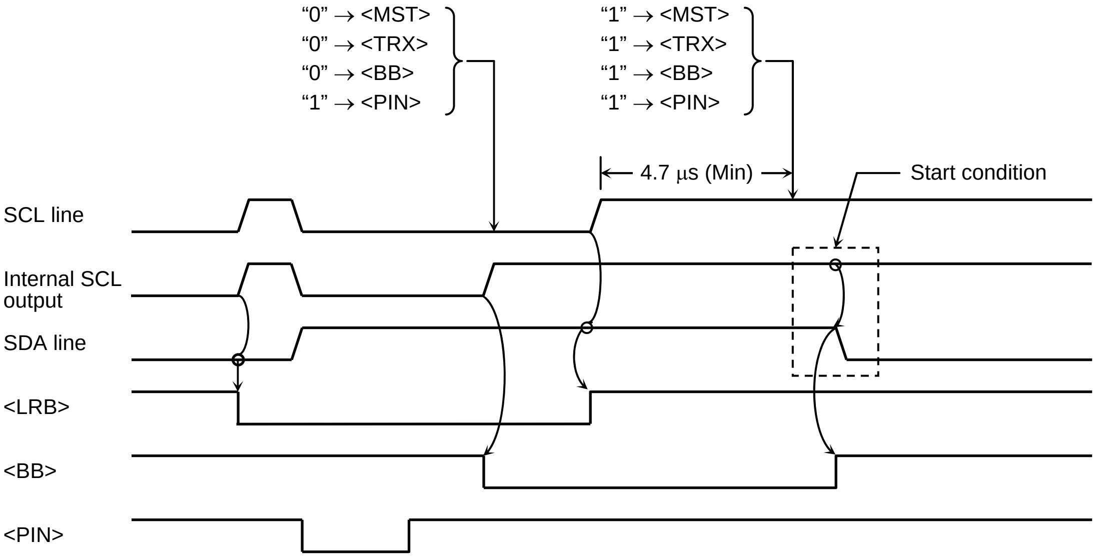


Figure 3.10.24 Timing Diagram when Restarting

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.

Serial Bus Interface 0 Control Register 1

SBI0CR1 (1240H)

Read-modify-write instruction is prohibited.

	7	6	5	4	3	2	1	0
Bit symbol	SIOS	SIOINH	SIOM1	SIOM0		SCK2	SCK1	SCK0
Read/Write	W					W		
Reset State	0	0	0	0		0	0	0
Function	Transfer start 0: Stop 1: Start	Continue/ abort transfer 0: Continue transfer 1:Abort transfer	Transfer mode select 00: Transmit mode 01: (Reserved) 10: Transmit/receive mode 11: Receive mode			Serial clock selection and reset monitor		

Internal serial clock selection <SCK2:0> at write

000	n = 4	2.5 MHz	System clock: f_{SYS} f_{SYS} = 20 MHz (internal SCL output) $f_{SCL} = \frac{f_{SYS} \times 2}{2^n}$ [Hz]
001	n = 5	1.25 MHz	
010	n = 6	625.0 kHz	
011	n = 7	312.5 kHz	
100	n = 8	156.3 kHz	
101	n = 9	78.1 kHz	
110	n = 10	39.1 kHz	
111	–	External clock: SCK0	

Transfer mode selection

00	8-bit transmit mode
01	(Reserved)
10	8-bit transmit/receive mode
11	8-bit receive mode

Continue/abort transfer

0	Continue transfer
1	Abort transfer (Automatically cleared after transfer aborted)

Indicate transfer start/stop

0	Stop
1	Start

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

Serial Bus Interface 0 Data Buffer Register

SBI0DBR (1241H)

Read-modify-write instruction is prohibited.

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

Figure 3.10.25 Register for the SIO Mode (SBI0)

Serial Bus Interface 1 Control Register 1

SBI1CR1
(1248H)

Read-modify-write instruction is prohibited.

	7	6	5	4	3	2	1	0
Bit symbol	SIOS	SIOINH	SIOM1	SIOM0		SCK2	SCK1	SCK0
Read/Write	W					W		
Reset State	0	0	0	0		0	0	0
Function	Transfer start 0: Stop 1: Start	Continue/ abort transfer 0: Continue transfer 1:Abort transfer	Transfer mode select 00: Transmit mode 01: (Reserved) 10: Transmit/receive mode 11: Receive mode			Serial clock selection and reset monitor		

Internal serial clock selection <SCK2:0> at write

000	n = 4	2.5 MHz	System clock: fSYS f_{SYS} = 20 MHz (internal SCL output) $f_{SCL} = \frac{f_{SYS}\times 2}{2n}$ [Hz]
001	n = 5	1.25 MHz	
010	n = 6	625.0 kHz	
011	n = 7	312.5 kHz	
100	n = 8	156.3 kHz	
101	n = 9	78.1 kHz	
110	n = 10	39.1 kHz	
111	–	External clock: SCK1	

Transfer mode selection

00	8-bit transmit mode
01	(Reserved)
10	8-bit transmit/receive mode
11	8-bit receive mode

Continue/abort transfer

0	Continue transfer
1	Abort transfer (Automatically cleared after transfer aborted)

Indicate transfer start/stop

0	Stop
1	Start

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

Serial Bus Interface 0 Data Buffer Register

SBI1DBR
(1248H)
Read-modify-write instruction is prohibited.

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

Figure 3.10.26 Register for the SIO Mode (SBI1)

Serial Bus Interface 0 Control Register 2

SBI0CR2
(1243H)

Read-modify-write instruction is prohibited.

	7	6	5	4	3	2	1	0
Bit symbol					SBIM1	SBIM0	–	–
Read/Write					W			
Reset State					0	0	0	0
Function					Serial bus interface operation mode selection 00: Port mode 01: SIO mode 10: I^2C bus mode 11: (Reserved)		Always write “0”.	Always write “0”.

Note 1: Set the SBI0CR1<BC2:0> “000” before switching to a clocked-synchronous 8-bit SIO mode.

Note 2: Please always write “00” to SBICR2<1:0>.

Serial bus interface operation mode selection

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

Serial Bus Interface 0 Status Register

SBI0SR
(1243H)

	7	6	5	4	3	2	1	0
Bit symbol					SIOF	SEF		
Read/Write					R			
Reset State					0	0		
Function					Serial transfer operation status monitor	Shift operation status monitor		

Serial transfer operating status monitor

0	Transfer terminated
1	Transfer in progress

Shift operation status monitor

0	Shift operation terminated
1	Shift operation in progress

Serial Bus Interface 0 Baud Rate Register 0

SBI0BR0
(1244H)

Read-modify-write instruction is prohibited.

	7	6	5	4	3	2	1	0
Bit symbol	–	I2SBI0						
Read/Write	W	R/W						
Reset State	0	0						
Function	Always write “0”.	IDLE2 0: Stop 1: Operate						

Operation in IDLE mode

0	Stop
1	Operate

Note: Clocked-synchronous mode cannot operate in IDLE2 mode.

Serial Bus Interface 0 Baud Rate Register 1

SBI0BR1
(1245H)

Prohibit read-modify-write

	7	6	5	4	3	2	1	0
Bit symbol	P4EN	–						
Read/Write	W							
Reset State	0	0						
Function	Internal clock 0: Stop 1: Operate	Always write “0”.						

Baud rate clock control

0	Stop
1	Operate

Figure 3.10.27 Registers for the SIO Mode (SBI1)

Serial Bus Interface 1 Control Register 2

SBI1CR2
(124BH)

Read-modify-write instruction is prohibited.

	7	6	5	4	3	2	1	0
Bit symbol					SBIM1	SBIM0	–	–
Read/Write					W			
Reset State					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)		Always write “0”.	Always write “0”.

Note 1: Set the SBI1CR1<BC2:0> “000” before switching to a clocked-synchronous 8-bit SIO mode.

Note 2: Please always write “00” to SBICR2<1:0>.

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)

Serial Bus Interface 1 Status Register

SBI1SR
(124BH)

	7	6	5	4	3	2	1	0
Bit symbol					SIOF	SEF		
Read/Write					R			
Reset State					0	0		
Function					Serial transfer operation status monitor	Shift operation status monitor		

Serial transfer operating status monitor

0	Transfer terminated
1	Transfer in progress

Shift operation status monitor

0	Shift operation terminated
1	Shift operation in progress

Serial Bus Interface 1 Baud Rate Register 0

SBI1BR0
(124CH)

Read-modify-write instruction is prohibited.

	7	6	5	4	3	2	1	0
Bit symbol	–	I2SBI1						
Read/Write	W	R/W						
Reset State	0	0						
Function	Always write “0”.	IDLE2 0: Stop 1: Operate						

Operation in IDLE mode

0	Stop
1	Operate

Note: Clocked-synchronous mode cannot operate in IDLE2 mode.

Serial Bus Interface 1 Baud Rate Register 1

SBI1BR1
(124DH)

Read-modify-write instruction is prohibited.

	7	6	5	4	3	2	1	0
Bit symbol	P4EN	–						
Read/Write	W							
Reset State	0	0						
Function	Internal clock 0: Stop 1: Operate	Always write “0”.						

Baud rate clock control

0	Stop
1	Operate

Figure 3.10.28 Registers for the SIO Mode (SBI1)

(1) Serial clock

1. Clock source

SBI0CR1<SCK2:0> is used to select the following functions:

Internal clock

In an internal clock mode, any of seven frequencies can be selected. The serial clock is output to the outside on the SCK pin.

When the device is writing (in the transmit mode) or reading (in the receive mode) data cannot follow the serial clock rate, an automatic wait function is executed to stop the serial clock automatically and holds the next shift operation until reading or writing is complete.

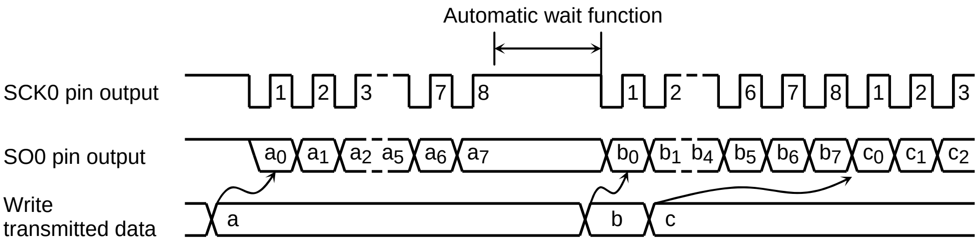
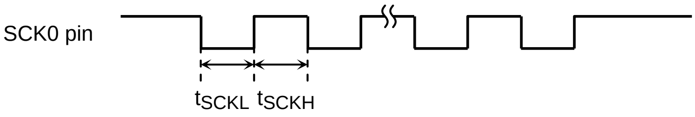


Figure 3.10.29 Automatic Wait Function

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 2.5 MHz (when f_{SYS} = 20 MHz).



t_{SCKL}, t_{SCKH} > $4/f_{SYS}$

Figure 3.10.30 Maximum Data Transfer Frequency when External Clock Input

2. Shift edge

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

(a) Leading edge shift

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

(b) Trailing edge shift

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

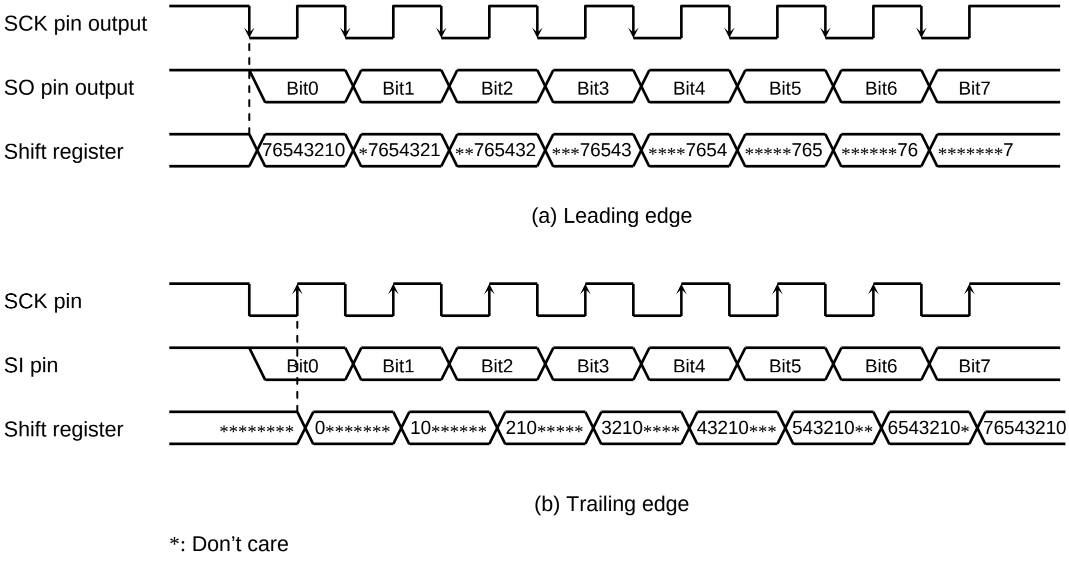


Figure 3.10.31 Shift Edge

(2) Transfer modes

The SBI0CR1<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 the SBI0DBR.

After the transmit data has been written, set the SBI0CR1<SIOS> to “1” to start data transfer. The transmitted data is transferred from the SBI0DBR 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 SBI0DBR becomes empty. The INTSBE0 (Buffer empty) interrupt request is generated to request new data.

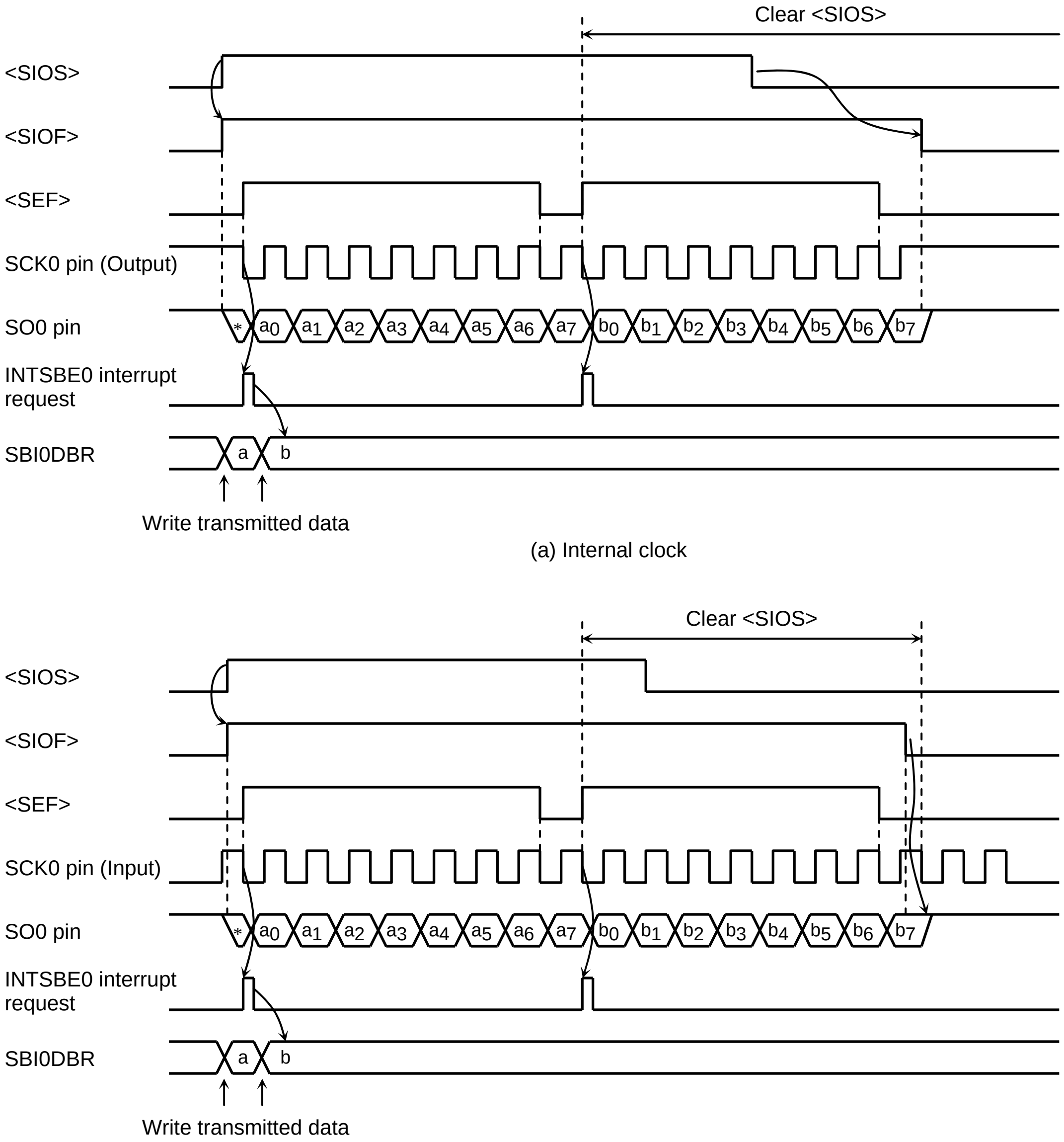
When the internal clock is used, the serial clock will stop and the automatic wait function will be initiated if new data is not loaded to the data buffer register after the specified 8-bit data is transmitted. When new transmission data is written, the automatic wait function is canceled.

When the external clock is used, data should be written to the SBI0DBR before new data is shifted. The transfer speed is determined by the maximum delay time between the time when an interrupt request is generated and the time when data is written to the SBI0DBR by the interrupt service program.

When the transmit is started, after the SBI0SR<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 the <SIOS> is cleared to “0” by the INTSBE0 interrupt service program or when the <SIOINH> is set to “1”. When the <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, the <SIOF> (Bit3 of the SBI0SR) to be sensed. The SBI0SR<SIOF> is cleared to “0” when transmission has been completed. When the <SIOINH> is set to “1”, transmitting data stops. The <SIOF> turns “0”.

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



(a) Internal clock

(b) External clock

Figure 3.10.32 Transfer Mode

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

```
STEST1:     BIT       2, (SBI0SR)               ;    If <SEF> = 1 then loop
            JR        NZ, STEST1
STEST2:     BIT       0, (PN)                   ;    If SCK0 = 0 then loop
            JR        Z, STEST2
            LD        (SBI0CR1), 00000111B      ;    <SIOS> ← 0
```

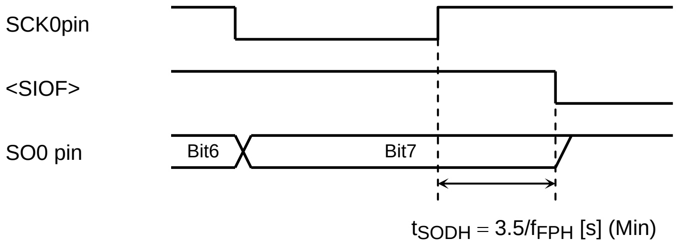


Figure 3.10.33 Transmitted Data Hold Time at End of Transmission

2. 8-bit receive mode

   Set the control register to receive mode and set the SBI0CR1<SIOS> 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 the 8-bit data is received, the data is transferred from the shift register to the SBI0DBR. The INTSBE0 (Buffer full) interrupt request is generated to request that the received data be read. The data is then read from the SBI0DBR by the interrupt service program.

   When the internal clock is used, the serial clock will stop and the automatic wait function will be in effect until the received data is read from the SBI0DBR.

   When the external clock is used, since shift operation is synchronized with an external clock pulse, the received data should be read from the SBI0DBR before the next serial clock pulse is input. If the received data is not read, further data 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 the <SIOS> is cleared to “0” by the INTSBE0 interrupt service program or when the <SIOINH> is set to “1”. If <SIOS> is cleared to “0”, received data is transferred to the SBI0DBR 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, the SBI0SR<SIOF> to be sensed. The <SIOF> is cleared to “0” when receiving is complete. When it is confirmed that receiving has been completed, the last data is read. When the <SIOINH> is set to “1”, data receiving stops. The <SIOF> 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 the SBI0DBR will be lost. If the mode must be changed, conclude data receiving by clearing the <SIOS> to “0”, read the last data, then change the mode.

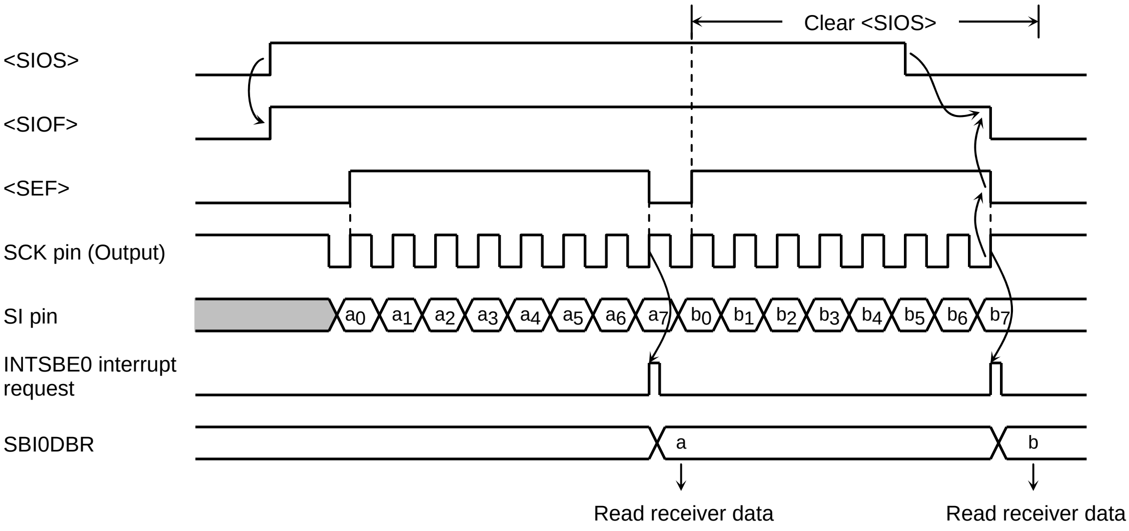


Figure 3.10.34 Receiver Mode (Example: Internal clock)

3. 8-bit transmit/receive mode

Set a control register to a transmit/receive mode and write data to the SBI0DBR. After the data is written, set the SBI0CR<SIOS> to “1” to start transmitting/receiving. When data is transmitted, the data is output from the SO0 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 the SBI0DBR and the INTSBE0 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. The SBI0DBR is used for both transmitting and receiving. Transmitted data should always be written after received data is read.

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

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

When the transmit is started, after the SBI0SR<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 the <SIOS> is cleared to “0” by the INTSBE0 interrupt service program or when the SBI0CR1<SIOINH> is set to “1”. When the <SIOS> is cleared to “0”, received data is transferred to the SBI0DBR 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 the SBI0SR to be sensed. The <SIOF> is set to “0” when transmitting/receiving is completed. When the <SIOINH> is set to “1”, data transmitting/receiving stops. The <SIOF> is then cleared to “0”.

Note: When the transfer mode is changed, the contents of the SBI0DBR will be lost. If the mode must be changed, conclude data transmitting/receiving by clearing the <SIOS> to “0”, read the last data, then change the transfer mode.

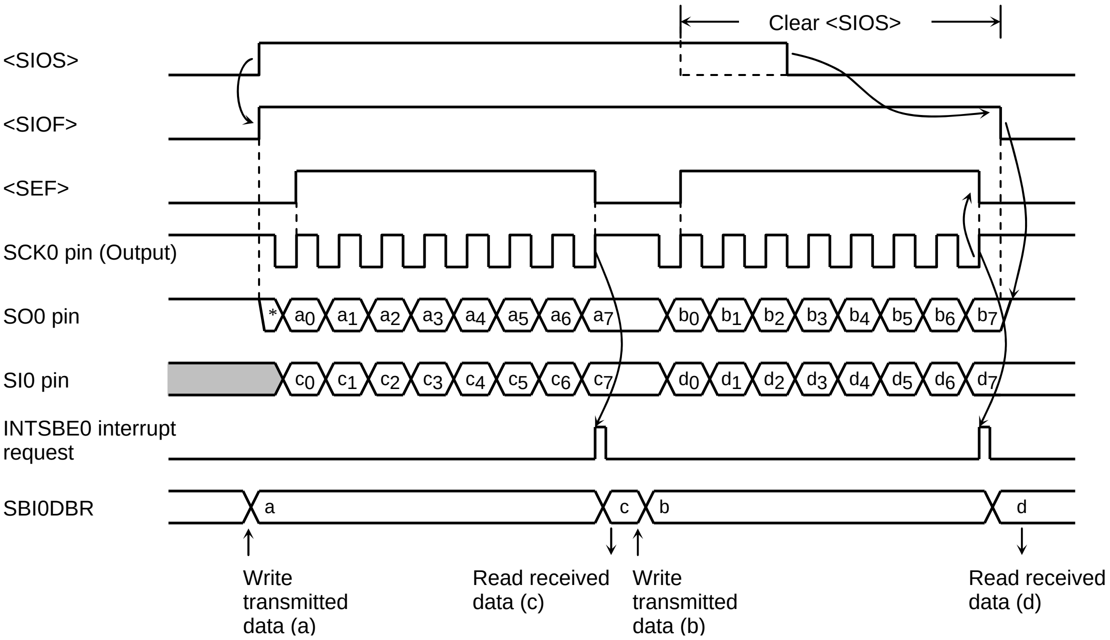

Clear <SIOS>
<SIOS>
<SIOF>
<SEF>
SCK0 pin (Output)
SO0 pin
* a0 a1 a2 a3 a4 a5 a6 a7 b0 b1 b2 b3 b4 b5 b6 b7
SI0 pin
c0 c1 c2 c3 c4 c5 c6 c7 d0 d1 d2 d3 d4 d5 d6 d7
INTSBE0 interrupt request
SBI0DBR
a c b d
Write transmitted data (a)
Read received data (c)
Write transmitted data (b)
Read received data (d)

Figure 3.10.35 Transmit/Received Mode (Example: Internal clock)

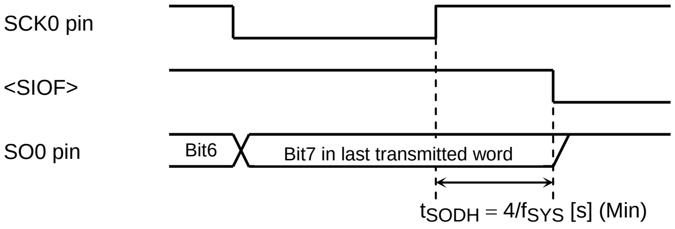

SCK0 pin
<SIOF>
SO0 pin
Bit6
Bit7 in last transmitted word
t_{SODH} = 4/f_{SYS} [s] (Min)

Figure 3.10.36 Transmitted Data Hold Time at End of Transmit/Receive

3.11 High Speed SIO (HSC)

Multifunction High Speed SIO (HSC) for 1 channel is contained. HSC supports only the master mode in I/O interface mode (synchronous transmission).

Its features are summarized as follows:

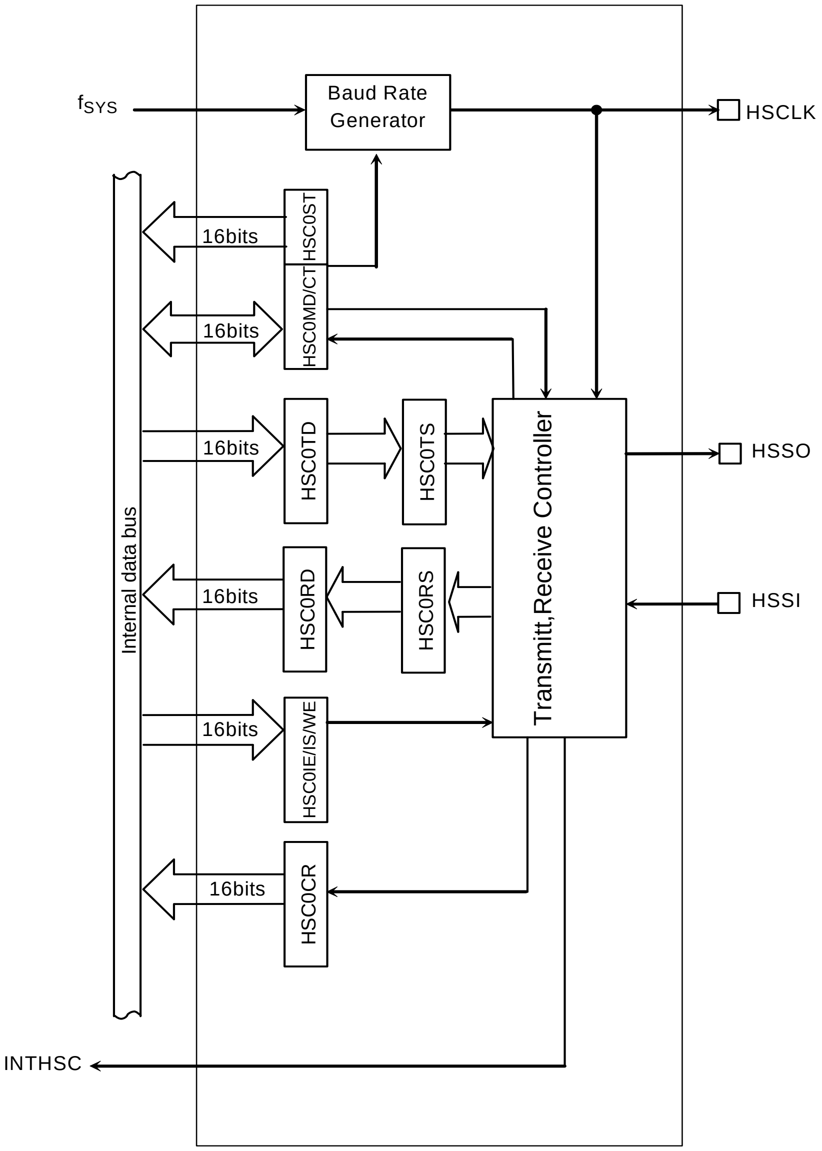
1) Double buffer (Transmit/Receive)
2) Generates the CRC-7 and CRC-16 values for transmission and reception
3) Baud Rate : 10Mbps (max)
4) Selects the MSB/LSB-first
5) Selects the 8/16-bit data length
6) Selects the Clock Rising/Falling edge
7) One types of interrupt: INTHSC
   Select Read/Mask/Clear interrupt/ Clear enable for 4 interrupts:
   RFR0 (Receive buffer of HSC0RD: Full),
   RFW0 (Transmission buffer of HSC0TD: Empty),
   REND0 (Receive buffer of HSC0RS: Full),
   TEND0 (Transmission buffer of HSC0TS: Empty).
   RFR0,RFW0 can be processed data at high-speeed by using micro DMA.

Table 3.11.1 Registers and Pins for HSC

	HSC
Pin name	HSSO (PF3) HSSI (PF4) HSCLK (PF5)
SFR (address)	HSC0MD (C00H/C01H) HSC0CT (C02H/C03H) HSC0ST (C04H/C05H) HSC0CR (C06H/C07H) HSC0IS (C08H/C09H) HSC0WE (C0AH/C0BH) HSC0IE (C0CH/C0DH) HSC0IR (C0EH/C0FH) HSC0TD (C10H/C11H) HSC0RD (C12H/C13H) HSC0TS (C14H/C15H) HSC0RS (C16H/C17H)

3.11.1 Block diagram

Figure 3.11.1 shows a block diagram of the HSC.



Note : The HSSO, HSSI, HSCLK pins are set to configured as input ports (Ports PF3, PF4 and PF5) by upon reset. Thus, these pins require pull-up resistors to fix their voltage levels.

Figure 3.11.1 HSC Block diagram

3.11.2 SFR

This section describes the SFRs of the HSC are as follows. These area connected to the CPU with 16 bit data buses.

(1) Mode setting register

The HSC0MD register specifies the operating mode, clock operation, etc.

HSC0MD Register

		7	6	5	4	3	2	1	0
HSC0MD	bit Symbol		XEN0				CLKSEL02	CLKSEL01	CLKSEL00
(0C00H)	Read/Write		R/W				R/W		
	Reset State		0				1	0	0
	Function		SYSCK 0: Disable 1: Enable				Select baud rate 000: Reserved 100: f_{SYS}/16 001: f_{SYS}/2 101: f_{SYS}/32 010: f_{SYS}/4 111: f_{SYS}/64 011: f_{SYS}/8 111:Reserved		
		15	14	13	12	11	10	9	8
(0C01H)	bit Symbol	LOOPBACK0	MSB1ST0	DOSTAT0		TCPOL0	RCPOL0	TDINV0	RDINV0
	Read/Write	R/W				R/W			
	Reset State	0	1	1		0	0	0	0
	Function	LOOPBACK test Mode 0:Disbale 1:Enable	Start Bit for Transmission /Reception 0:LSB 1:MSB	HSSO0 Pin When Not Transmitting 0: Fixed to "0" 1: Fixed to "1"		Synchroniza-tion Clock Edge Select For Transmission 0: Falling edge 1: Rising edge	Synchroniza-tion Clock Edge Select for Reception 0: fall 1: rise	Data Inversion for Transmission 0: Disable 1: Enable	Data Inversion for Reception 0: Disable 1: Enable

Figure 3.11.2 HSC0MD Register

(a) <LOOPBACK0>

The internal HSSO output to be internally connected to the HSSI input. This setup can be used for testing.

Also, a clock signal is generated from the HSCLK pin, regardless of whether data transmission or reception is in progress when setting the XEN0 and LOOPBACK0 bits to 1 enables.

Data transmission or reception must not be performed while changing the state of this bit.

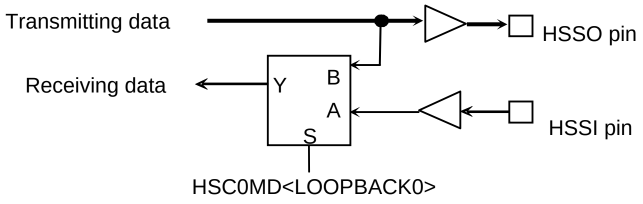


Figure 3.11.3 <LOOPBACK0> Register Function

(b) <MSB1ST0>

This bit specifies whether to transmit/receive byte with the MSB first or with the LSB first. Data transmission or reception must not be performed while changing the state of this bit.

(c) <DOSTAT0>

This bit specifies the status of the HSSO pin of when data transmission is not performed (i.e., after completing data transmission or during data reception). Data transmission or reception must not be performed while changing the state of this bit.

(d) <TCPOL0>

This bit specifies the polarity of the active edge of the synchronization clock for data transmission.

The XEN0 bit should be cleared to 0 for changing the state of this bit. At the same time, RCPOL0 should also be cleared to 0.

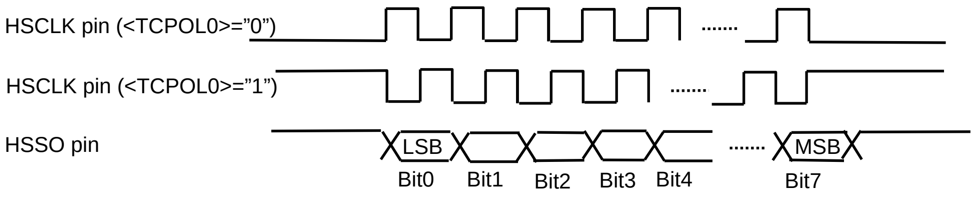


Figure 3.11.4 <TCPOL0> Register function

(e) <RCPOL0>

This bit specifies the polarity of the active edge of the synchronization clock during for data reception.

The <XEN0> bit should be cleared to 0 for changing the state of this bit. TCPOL0 should also be cleared to 0.

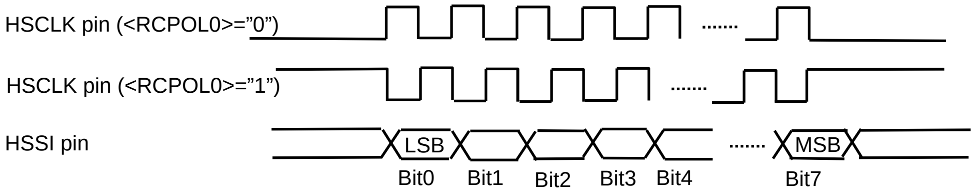


Figure 3.11.5 <RCPOL0> Register function

(f) <TDINV0>

This bit specifies whether to logically invert the data transmitted from the HSSO pin or not. Data transmission or reception must not be performed while changing the state of this bit.

Data which is inputted to CRC calculation circuit is transmission data which is written to HSC0TD. This input data is not corresponded to <TDINV0>.

<TDINV0> is not corresponded to <DOSTAT0>: it set condition of HSSO pin when it is not transferred.

(g) <RDINV0>

This bit specifies whether to logically invert the data received from the HSSI pin or not. Data transmission or reception must not be performed while changing the state of this bit.

Data which is inputted to CRC calculation circuit is selected by <RDINV0>.

(h) <XEN0>

This bit enables or disables the internal clock signal.

(i) <CLKSEL02:00>

This bit selects the baud rate. The baud rate is generated using the system clock f_{SYS} and is programmable as shown below according to the system clock settings.

Data transmission or reception must not be performed while changing the state of these bits

Table 3.11.2 Example of baud rate

<CLKSEL02:00>	Baud rate [Mbps]		
	f_{SYS} =12MHz	f_{SYS} =16MHz	f_{SYS} =20MHz
f_{SYS}/2	6	8	10
f_{SYS}/4	3	4	5
f_{SYS}/8	1.5	2	2.5
f_{SYS}/16	0.75	1	1.25
f_{SYS}/32	0.375	0.5	0.625
f_{SYS}/64	0.1875	0.25	0.3125

(2) Control Register

The HSC0CT register specifies data length, CRC, etc.

HSC0CT Register

		7	6	5	4	3	2	1	0
HSC0CT	bit Symbol	–	–	UNIT160			ALGNEN0	RXWEN0	RXUEN0
(0C02H)	Read/Write	R/W					R/W		
	Reset State	0	1	0			0	0	0
	Function	Always write “0”.	Always write “1”.	Data Length 0: 8 bits 1: 16 bits			Full Duplex Alignment 0: Disable 1: Enable	Sequential Reception0: Disable 1: Enable	Receive UNIT 0: Disable 1: Enable
		15	14	13	12	11	10	9	8
(0C03H)	bit Symbol	CRC16_7_B0	CRCRX_TX_B0	CRCRESET_B0				DMAERFW0	DMAERFR0
	Read/Write	R/W						R/W	R/W
	Reset State	0	0	0				0	0
	Function	CRC Select 0: CRC7 1: CRC16	CRC Data 0: Transmit 1: Receive	CRC Calculation Register 0:Reset 1: Reset Release				Micro DMA 0: Disable 1: Enable	Micro DMA 0: Disable 1: Enable

Figure 3.11.6 HSC0CT Register

(a) <CRC16_7_B0>

This bit selects the CRC calculation algorithm from the CRC7 and CRC16.

(b) <CRCRX_TX_B0>

This bit selects the data to be sent to the CRC generator.

(c) <CRCRESET_B0>

This bit is used to initialize the CRC calculation register.

This section describes how to calculate the CRC16 of the transmit data and to append the calculated CRC value at the end of the transmit data. Figure 3.11.7 below illustrates the flow chart of the CRC calculation procedures.

a. Program the HSC0CT<CRC16_7_B> bit to select the CRC algorithm from CRC7 and CRC16. Then, also program the CRCRX_TX_B bit to specify the data on which the CRC calculation is performed.
b. To reset the HSC0CR register, write a 0 to the CRCRESET_B bit and then write a 1 to the same bit.
c. Load the HSC0TD register with the transmit data, and wait until transmission of all data is completed.
d. Read the HSC0CR register and obtain the result of the CRC calculation.
e. Transmit the CRC obtained in step (d) in the same way as step (c).

The CRC calculation on the receive data can be performed in the same procedures.

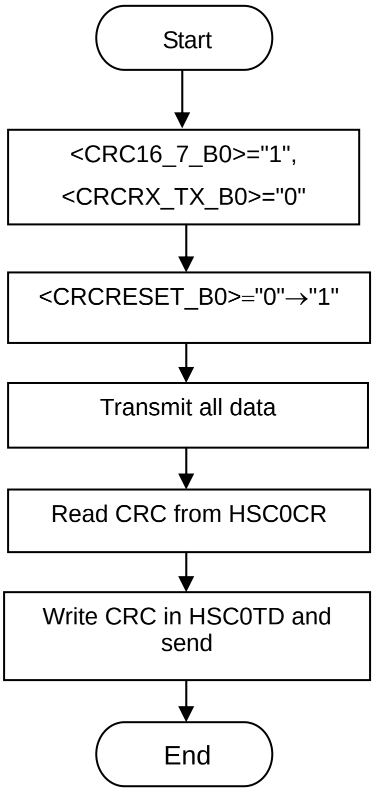


Figure 3.11.7 Flow Chart of the CRC Calculation Procedures

(d) <DMAERFW0>

This bit sets the interrupt clearing using to unnecessary because be supported RFW0 interrupt to Micro DMA. If this bit is set to "1", it is set to one-shot interrupt, clearing interrupt by HSC0WE register become to unnecessary. HSC0ST<RFW0> flag generate 1-shot interrupt when change from "0" to "1"(Rising).

(e) <DMAERFR0>

This bit sets the interrupt clearing using CPU to unnecessary because be supported RFR0 interrupt to Micro DMA. If this bit is set to "1", it is set to one-shot interrupt, clearing interrupt by HSC0WE register become to unnecessary. HSC0ST<RFR0> flag generate 1-shot interrupt when change from "0" to "1"(Rising).

(f) <UNIT160>

This bit selects the data length for transmission and reception. The data length is hereafter referred to as the UNIT. Data transmission or reception must not be performed while changing the state of this bit

(g) <ALGNEN0>

This bit should be set to 1 when performing the full-duplex communication. This bit specifies whether to align the transmit and receive data on the UNIT-size boundaries.

Data transmission or reception must not be performed while changing the state of this bit.

(h) <RXWEN0>

This bit enables or disables the Sequential mode reception.

(i) <RXUEN0>

This bit enables or disables the Unit mode reception.

For <RXWEN0> = “1”, this bit is disabled. Data transmission or reception must not be performed while changing the state of this bit.

[Data Transmission/Reception Modes]

This HSC Controller supports six operating modes as listed below.

These are specified by the <ALGNEN0>, <RXWEN0>, <RXUEN0> bits.

Table 3.11.3 transmit/receive operation mode

Operation mode	Bit Settings			Description
	<ALGNEN0>	<RXWEN0>	<RXUEN0>	
(1) UNIT transmission	0	0	0	Transmit written data per UNIT
(2) Sequential transmission	0	0	0	Transmit written data sequentially
(3) UNIT reception	0	0	1	Receive only one UNIT-size data
(4) Sequential reception	0	1	0	Automatically receive data if buffer has any empty space
(5) UNIT transmission and reception	1	0	1	Transmit/receive one UNIT-size data with the addresses of transmit/receive data aligned on UNIT-size boundaries
(6)Sequential transmission and reception	1	1	0	Transmit/receive data sequentially with the addresses of transmit/receive data aligned on UNIT-size boundaries

Difference between the UNIT-mode and Sequential-mode transmission

UNIT mode transmission transmits one-UNIT by writing data after confirming HSC0ST<TEND0>=1.

In the Sequential-mode transmission, transmit data written into the HSC0TD is loaded sequentially.

In hard ware, this mode of transmission keeps transmitting data as long as the transmit data exists.. This mode of transmission keeps transmitting data as long as the transmit data exists. Therefore, the Sequential-mode transmission continues as long as the next data is written to it when HSC0ST<REND0>=1.

Unit-mode transmission and Sequential-mode transmission depend on the way of using. Hardware doesn't depend on.

Figure 3.11.8 show Flow chart of UNIT-mode transmission and Sequential-mode transmission.

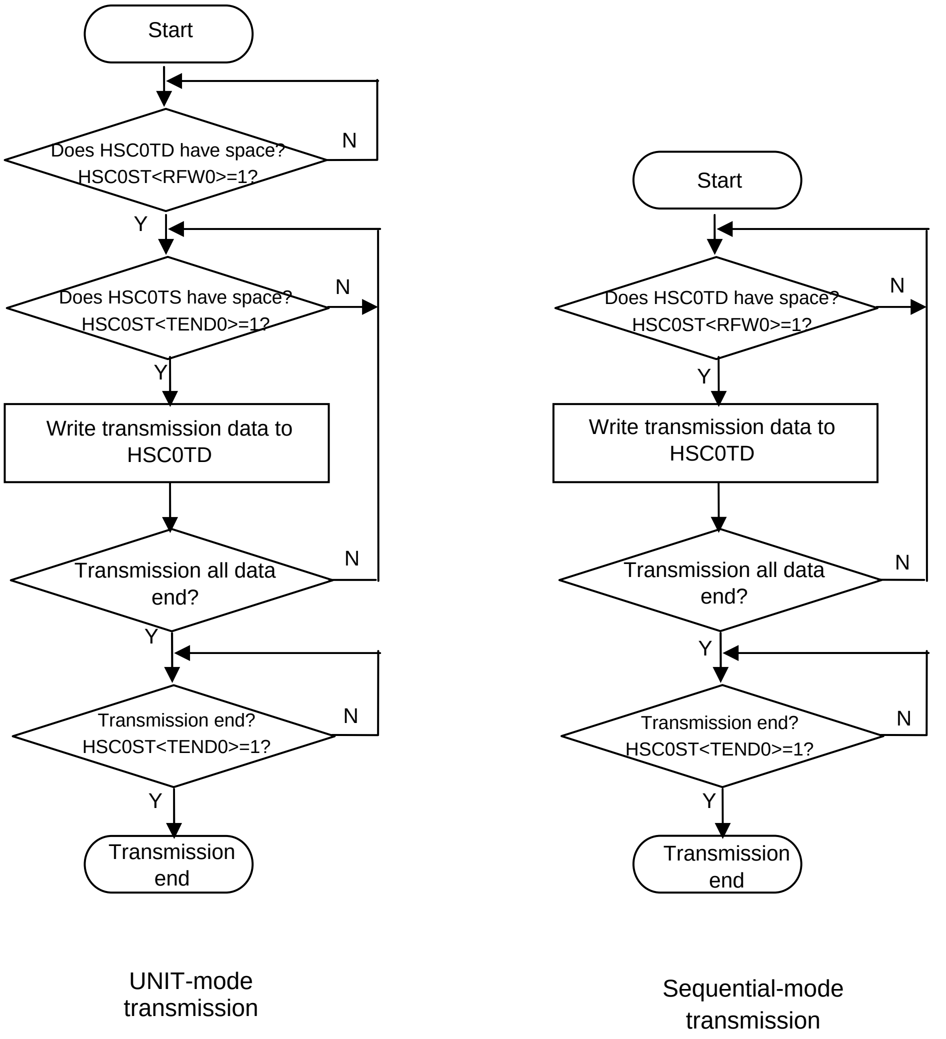


Figure 3.11.8 Flow chart of UNIT-mode transmission and Sequential-mode transmission

Differences Between the UNIT-mode and Sequential-mode Receptions

The UNIT-mode reception receives only one UNIT-size data.

Writing a 1 to the HSC0CT<RXUEN0> bit initiates a receive operation of one UNIT data. Then, it is stored the received data into the receive data register (HSC0RD).

Reading the HSC0RD register after writing a 0 to the HSC0CT<RXUEN0> bit.

If the HSC0RD register is read again when the HSC0CT<RXUEN0> bit is set to1, one-UNIT data is additionally received.

In hardware, this mode receives sequentially by Single buffer.

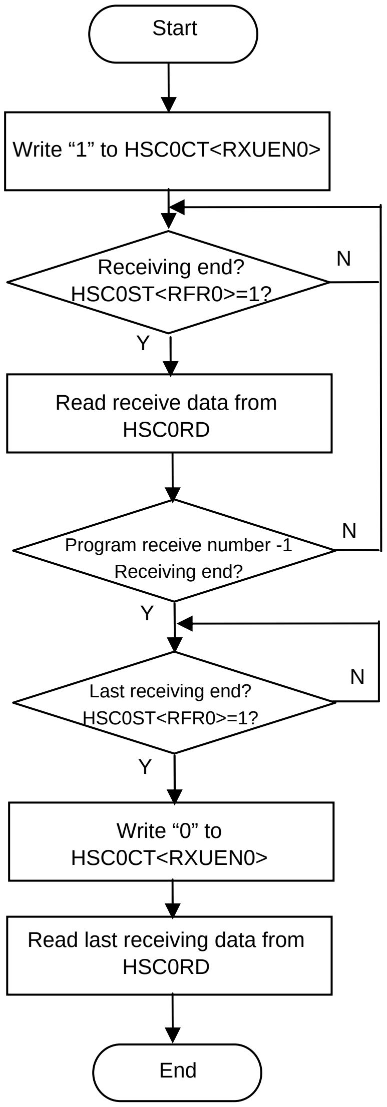
HSC0ST<REND0> is changed during UNIT receiving.

The Sequential-mode reception automatically receives the data as long as the receive Buffer has any empty space.

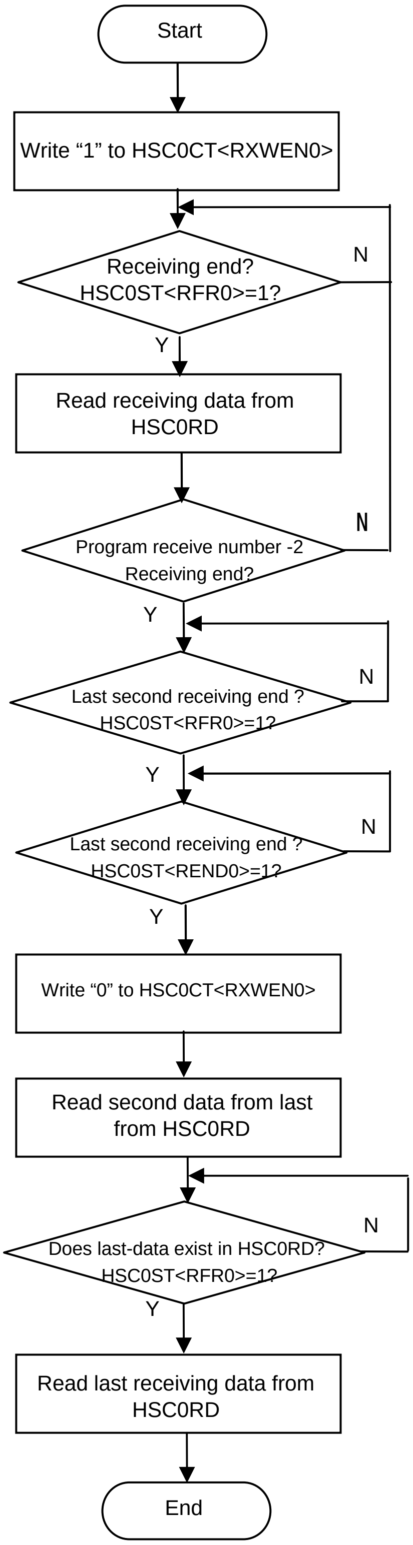
This mode of reception keeps receiving the next data automatically unless the data receive Buffer becomes full. Therefore, the reception continues sequentially without stopping at every UNIT-sized reception by reading it after data is loaded in HSC0RD.

In hardware, this mode receives sequentially by Double buffer.

Figure 3.11.9 show Flow chart of UNIT-reception and Sequential-mode reception.



UNIT-mode reception


Start
Write “1” to HSC0CT<RXWEN0>
Receiving end?
HSC0ST<RFR0>=1?
N
Y
Read receiving data from
HSC0RD
Program receive number -2
Receiving end?
N
Y
Last second receiving end ?
HSC0ST<RFR0>=1?
N
Y
Last second receiving end ?
HSC0ST<REND0>=1?
N
Y
Write “0” to HSC0CT<RXWEN0>
Read second data from last
from HSC0RD
Does last-data exist in HSC0RD?
HSC0ST<RFR0>=1?
N
Y
Read last receiving data from
HSC0RD
End

Sequential-mode reception

Figure 3.11.9 Flow chart of UNIT-mode reception and Sequential-mode reception

(3) Interrupt , Status register

Read of condition, Mask of condition, Clear interrupt and Clear enable can control each 4 interrupts; RFR0 (HSC0RD receiving buffer is full), RFW0 (HSC0TD transmission buffer is empty), REND0 (HSC0RS receiving buffer is full), TEND0 (HSC0TS transmission buffer is empty).

RFR0, RFW0 can high-speed transaction by micro DMA.

Following is description of Interrupt・status (example RFW0).

Status register HSC0ST<RFW0> show RFW0 (internal signal that show whether transmission data register exist or not). This register is "0" when transmission data exist. This register is "1" when transmission data doesn't exist. It can read internal signal directly. Therefore, it can confirm transmission data at any time.

Interrupt status register HSC0IS<RFWIS0> is set by rising edge of RFW0. This register keeps that condition until write "1" to this register and reset when HSC0WE<RFWWE0> is "1".

RFW0 interrupt generate when interrupt enable register HSC0IE<RFWIE0> is "1". When it is "0", interrupt is not generated.

Interrupt request register HSC0IR<RFWIR0> show whether interrupt is generating or not.

Interrupt status write enable register HSC0WE<RFWWE0> set that enables reset for reset interrupts status register by mistake.

Circuit config of transmission data shift register (HSC0TS), receiving register (HSC0RD), receiving data shift register (HSC0RS) are same with above register.

Control register HSC0CT<DMAERFW0>, HSC0CT<DMAERFR0> is register for using micro DMA. When micro DMA transfer is executed by using RFW0 interrupt, set "1" to <DMAERFW0>, and when it is executed by using RFR0 interrupt, set "1" to <DMAERFR0>, and prohibit other interrupt.

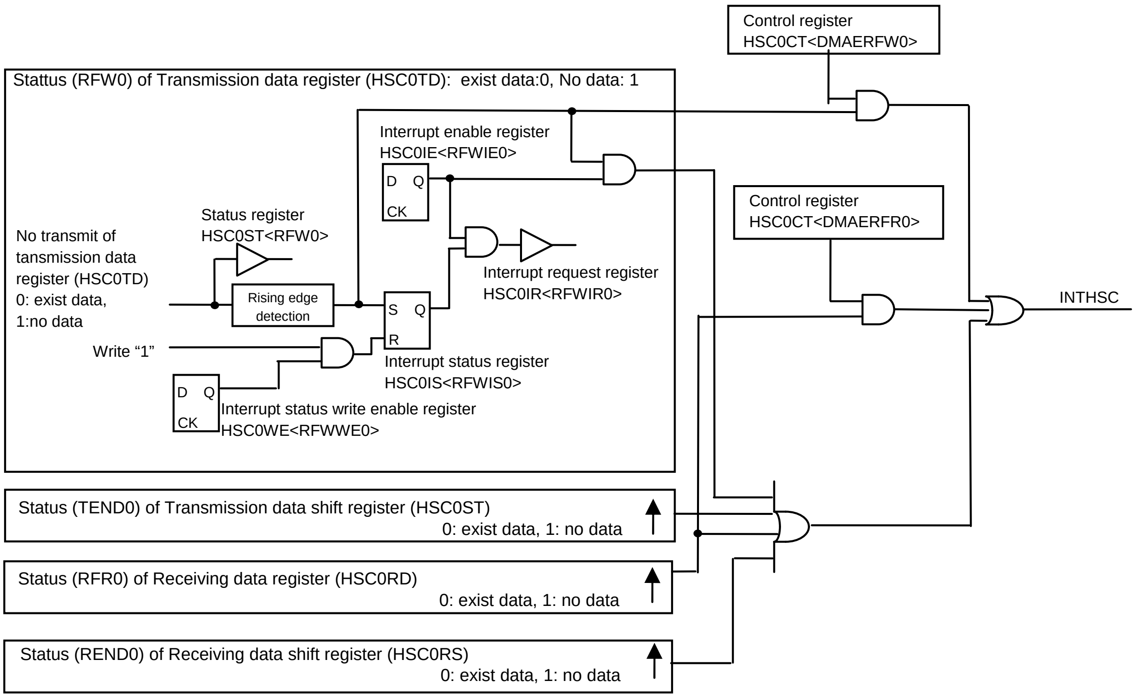

Control register
HSC0CT<DMAERFW0>
Stattus (RFW0) of Transmission data register (HSC0TD): exist data:0, No data: 1
Interrupt enable register
HSC0IE<RFWIE0>
D Q
CK
Status register
HSC0ST<RFW0>
No transmit of
tansmission data
register (HSC0TD)
0: exist data,
1:no data
Rising edge detection
S Q
R
Interrupt request register
HSC0IR<RFWIR0>
Control register
HSC0CT<DMAERFR0>
INTHSC
Write "1"
Interrupt status register
HSC0IS<RFWIS0>
D Q
CK
Interrupt status write enable register
HSC0WE<RFWWE0>
Status (TEND0) of Transmission data shift register (HSC0ST)
0: exist data, 1: no data
Status (RFR0) of Receiving data register (HSC0RD)
0: exist data, 1: no data
Status (REND0) of Receiving data shift register (HSC0RS)
0: exist data, 1: no data

Figure 3.11.10 Figurer for interrupt, status

(3-1) Status register

This register contains four bits that indicates the status of data communication.

HSC0ST Register

		7	6	5	4	3	2	1	0
HSC0ST	bit Symbol					TEND0	REND0	RFW0	RFR0
(0C04H)	Read/Write					R			
	Reset State					1	0	1	0
	Function					Receiving 0:operation 1: no operation	Receive Shift register 0: no data 1: exist data	Transmit buffer 0:untransm-itted data exist 1: no Untrans-mitted data	Receive buffer 0:no valid data 1: valid data exist
		15	14	13	12	11	10	9	8
(0C05H)	bit Symbol								
	Read/Write								
	Reset State								
	Function								

Figure 3.11.11 HSC0ST Register

(a) <TEND0>

This bit is cleared to 0 when the transmit register (HSC0TS) contains valid data; otherwise, it is set to 1.

(b) <REND0>

This bit is set to 1 when completing the data reception and valid data is stored into the receive data register (if there is any valid data). This bit is cleared to 0 when the receive register (HSC0RS) contains no valid data, or when the reception is in progress.

It is cleared to “0”, when CPU read the data and shift to receive read register.

(c) <RFW0>

After wrote the received data to receive data write register, shift the data to receive data shift register. This bit keeps “0”until all valid data has moved. And this bit is set to “1” when it can accept the next data and contains no valid data.

(d) <RFR0>

This bit is set to “1” when received data is shifted from received data shift register to received data read register and there is any valid data. It is set to “0” when the data is read and contains no valid data.

(3-2) Interrupt status register

This register is used for reading four interrupts status and clearing interrupts.

This register is cleared to “0” by writing “1” to applicable bit. Status of this register show interrupt source state. This register can confirm changing of interrupt condition, even if interrupt enable register is masked.

HSC0IS Register

		7	6	5	4	3	2	1	0
HSC0IS	bit Symbol					TENDIS0	RENDIS0	RFWIS0	RFRIS0
(0C08H)	Read/Write					R/W			
	Reset State					0	0	0	0
	Function					Read 0:no interrupt 1:interrupt Write 0:Don’t care 1:clear	Read 0:no interrupt 1:interrupt Write 0:Don’t care 1:clear	Read 0:no interrupt 1:interrupt Write 0:Don’t care 1:clear	Read 0:nointerrupt 1:interrupt Write 0:Don’t care 1:clear
		15	14	13	12	11	10	9	8
(0C09H)	bit Symbol								
	Read/Write								
	Reset State								
	Function								

Figure 3.11.12 HSC0IS Register

(a) <TENDIS0>

This bit is used for reading the status of TEND interrupt and clearing interrupt.

If writing this bit, set “1” to HSC0WE<TENDWE0>.

(b) <REMDIS0>

This bit is used for reading the status of REND interrupt and clearing interrupt.

If writing this bit, set “1” to HSC0WE<RENDWE0>.

(c) <RFWDIS0>

This bit is used for reading the status of RFW interrupt and clearing interrupt.

If writing this bit, set “1” to HSC0WE<RFWWE0>.

(d) <RFRIS0>

This bit is used for reading the status of RFR interrupt and clearing interrupt.

If writing this bit, set “1” to HSC0WE<RFRWE0>.

(3-3) Interrupt status write enable register

This register enables or disables the clearing status bit of four types of interrupts.

HSC0WE Register

		7	6	5	4	3	2	1	0
HSC0WE	bit Symbol					TENDWE0	RENDWE0	RFWWE0	RFRWE0
(0C0AH)	Read/Write					R/W			
	Reset State					0	0	0	0
	Function					Clear HSC0IS <TENDIS0> 0: Disable 1: Enable	Clear HSC0IS <RENDIS0> 0: Disable 1: Enable	Clear HSC0IS <TFWIS0> 0: Disable 1: Enable	Clear HSC0IS <RFRIS0> 0: Disable 1: Enable
		15	14	13	12	11	10	9	8
(0C0BH)	bit Symbol								
	Read/Write								
	Reset State								
	Function								

Figure 3.11.13 HSC0WE Register

(a) <TENDWE0>

This bit enables or disables clearing the HSC0IS<TENDIS0>.

(b) <RENDWE0>

This bit enables or disables clearing the HSC0IS<RENDIS0>.

(c) <RFWWE0>

This bit enables or disables clearing the HSC0IS<RFWIS0>.

(d) <RFRWE0>

This bit enables or disables clearing the HSC0IS<RFRIS0>.

(3-4) Interrupt enable register

This register enables or disables the generation of four types of interrupts.

HSC0IE Register

		7	6	5	4	3	2	1	0
HSC0IE	bit Symbol					TENDIE0	RENDIE0	RFWIE0	RFRIE0
(0C0CH)	Read/Write					R/W			
	Reset State					0	0	0	0
	Function					TEND0 interrupt 0: Disable 1: Enable	REND0 interrupt 0: Disable 1: Enable	RFW0 interrupt 0: Disable 1: Enable	RFR0 interrupt 0: Disable 1: Enable
		15	14	13	12	11	10	9	8
(0C0DH)	bit Symbol								
	Read/Write								
	Reset State								
	Function								

Figure 3.11.14 HSC0IE Register

(a) <TENDIE0>

This bit enables or disables the TEND0 interrupt.

(b) <RENDIE0>

This bit enables or disables the REND0 interrupt.

(c) <RFWIE0>

This bit enables or disables the RFW0 interrupt.

(d) <RFRIE0>

This bit enables or disables the RFR0 interrupt.

(3-5) Interrupt request register

This register is used for showing generation condition for 4 interrupts.

This register is set to the reading “0” (interrupt doesn’t generate) always when Interrupt enable register is masked.

HSC0IR Register

		7	6	5	4	3	2	1	0
HSC0IR	bit Symbol					TENDIR0	RENDIR0	RFWIR0	RFRIR0
(0C0EH)	Read/Write					R			
	Reset State					0	0	0	0
	Function					TEND0 interrupt 0: None 1:Generate	REND0 interrupt 0: None 1:Generate	RFW0 interrupt 0: None 1:Generate	RFR0 interrupt 0: None 1:Generate
		15	14	13	12	11	10	9	8
(0C0FH)	bit Symbol								
	Read/Write								
	Reset State								
	Function								

Figure 3.11.15 HSC0IR Register

(a) <TENDIR0>

This bit is used for showing the condition of TEND0 interrupt generation.

(b) <TENDIR0>

This bit is used for showing the condition of REND0 interrupt generation.

(c) <RFWIR0>

This bit is used for showing the condition of RFW0 interrupt generation.

(d) <RFRIR0>

This bit is used for showing the condition of RFR0 interrupt generation.

(4) HSC0CR (HSC0 CRC register)

This register contains the CRC calculation result for transmit/receive data.

HSC0CR register

		7	6	5	4	3	2	1	0
HSC0CR (0C06H)	bit Symbol	CRCD007	CRCD006	CRCD005	CRCD004	CRCD003	CRCD002	CRCD001	CRCD000
	Read/Write	R							
	Reset State	0	0	0	0	0	0	0	0
	Function	CRC calculation result load register [7:0]							
		15	14	13	12	11	10	9	8
(0C07H)	bit Symbol	CRCD015	CRCD014	CRCD013	CRCD012	CRCD011	CRCD010	CRCD009	CRCD008
	Read/Write	R							
	Reset State	0	0	0	0	0	0	0	0
	Function	CRC calculation result load register [15:8]							

Figure 3.11.16 HSC0CR register

(a) <CRCD015:000>

The CRC result which is calculated according to the settings of the CRC16_7_b0, CRCRX_TX_B0 and CRCRESET_B0 bits in the HSC0CT register are loaded into this register. When using the CRC16 algorithm, all the bits participate in the CRC generation. When using the CRC7 algorithm, only the lower seven bits participates in the CRC generation. The following describes the steps required to calculate the CRC16 for the transmit data.

First, initialize the CRC calculation register by writing a 1 to the CRCRESET_B0 bit after programming three bits as follows: CRC16_7_b0 =1, CRCRX_TX_B0 = 0, and CRCRESET_B0 = 0.

Then, by writing the transmit data into the HSC0TD register, complete the transmission of all bits, for which the CRC should be calculated.

The HSC0ST<TEND0> bit should be checked to confirm whether the reception is completed.

By reading the HSC0CR register after the transmission is completed, the CRC16 for the transmit data can be obtained.

(5) Transmit Data Register

This register is used for writing the transmit data.

HSC0TD Register

		7	6	5	4	3	2	1	0
HSC0TD	bit Symbol	TXD007	TXD006	TXD005	TXD004	TXD003	TXD002	TXD001	TXD000
(0C10H)	Read/Write	R/W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Transmit data bits [7:0]							
		15	14	13	12	11	10	9	8
(0C11H)	bit Symbol	TXD015	TXD014	TXD013	TXD012	TXD011	TXD010	TXD009	TXD008
	Read/Write	R/W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Transmit data bits [15:8]							

Figure 3.11.17 HSC0TD Register

(a) <TXD015:000>

This register is used for writing the transmit data. When this register is read, the last-written data is read out.

This register is overwritten if the next data is written with this register being full.

Please check the state of the RFW0 bit before starting a write operation.

HSC0CT<UNIT160>= “1”, all bits are valid.

HSC0CT<UNIT160>= “0”, lower 7 bits are valid.

(6) Receive Data Register

This register is used for reading the received data.

HSC0RD Register

		7	6	5	4	3	2	1	0
HSC0RD	bit Symbol	RXD007	RXD006	RXD005	RXD004	RXD003	RXD002	RXD001	RXD000
(0C12H)	Read/Write	R							
	Reset State	0	0	0	0	0	0	0	0
	Function	Receive data register [7:0]							
		15	14	13	12	11	10	9	8
(0C13H)	bit Symbol	RXD015	RXD014	RXD013	RXD012	RXD011	RXD010	RXD009	RXD008
	Read/Write	R							
	Reset State	0	0	0	0	0	0	0	0
	Function	Receive data register [15:8]							

Figure 3.11.18 HSC0RD Register

(a) <RXD015:000>

The HSC0RD register is used for reading the received data. Please check the state of the RFR0 bit before starting a read operation.

HSC0CT<UNIT160> = “1”, all bits are valid.

HSC0CT<UNIT160> = “0”, lower 7 bits are valid.

(7) Transmit data shift register

This register is used for changing the transmission data to serial. This register is used for confirming the changing condition when LSI test.

HSC0TS Register

		7	6	5	4	3	2	1	0
HSC0TS (0C14H)	bit Symbol	TSD007	TSD006	TSD005	TSD004	TSD003	TSD002	TSD001	TSD000
	Read/Write	R							
	Reset State	0	0	0	0	0	0	0	0
	Function	Transmit data shift register [7:0]							
		15	14	13	12	11	10	9	8
(0C15H)	bit Symbol	TSD015	TSD014	TSD013	TSD012	TSD011	TSD010	TSD009	TSD008
	Read/Write	R							
	Reset State	0	0	0	0	0	0	0	0
	Function	Transmit data shift register [15:8]							

Figure 3.11.19 HSC0TS Register

(a) <TSD015:000>

This register is used for reading the status of transmission data shift register.

HSC0CT<UNIT160>= “1”, all bits are valid.

HSC0CT<UNIT160>= “0”, lower 7 bits are valid.

(8) Receive data shift register

This register is used for reading the receive data shift register.

HSC0RS Register

		7	6	5	4	3	2	1	0
HSC0RS	bit Symbol	RSD007	RSD006	RSD005	RSD004	RSD003	RSD002	RSD001	RSD000
(0C16H)	Read/Write	R							
	Reset State	0	0	0	0	0	0	0	0
	Function	Receive data shift register [7:0]							
		15	14	13	12	11	10	9	8
(0C17H)	bit Symbol	RSD015	RSD014	RSD013	RSD012	RSD011	RSD010	RSD009	RSD008
	Read/Write	R							
	Reset State	0	0	0	0	0	0	0	0
	function	Receive data shift register [15:8]							

Figure 3.11.20 HSC0RS Register

(a) <RSD015:000>

This register is used for reading the status of receive data shift register.

HSC0CT<UNIT160>= “1”, all bits are valid.

HSC0CT<UNIT160>=“0”, lower 7 bits are valid.

3.11.3 Operation timing

Following examples show operation timing.

- Setting condition 1:
  Transmission in UNIT=8bit, LSB first

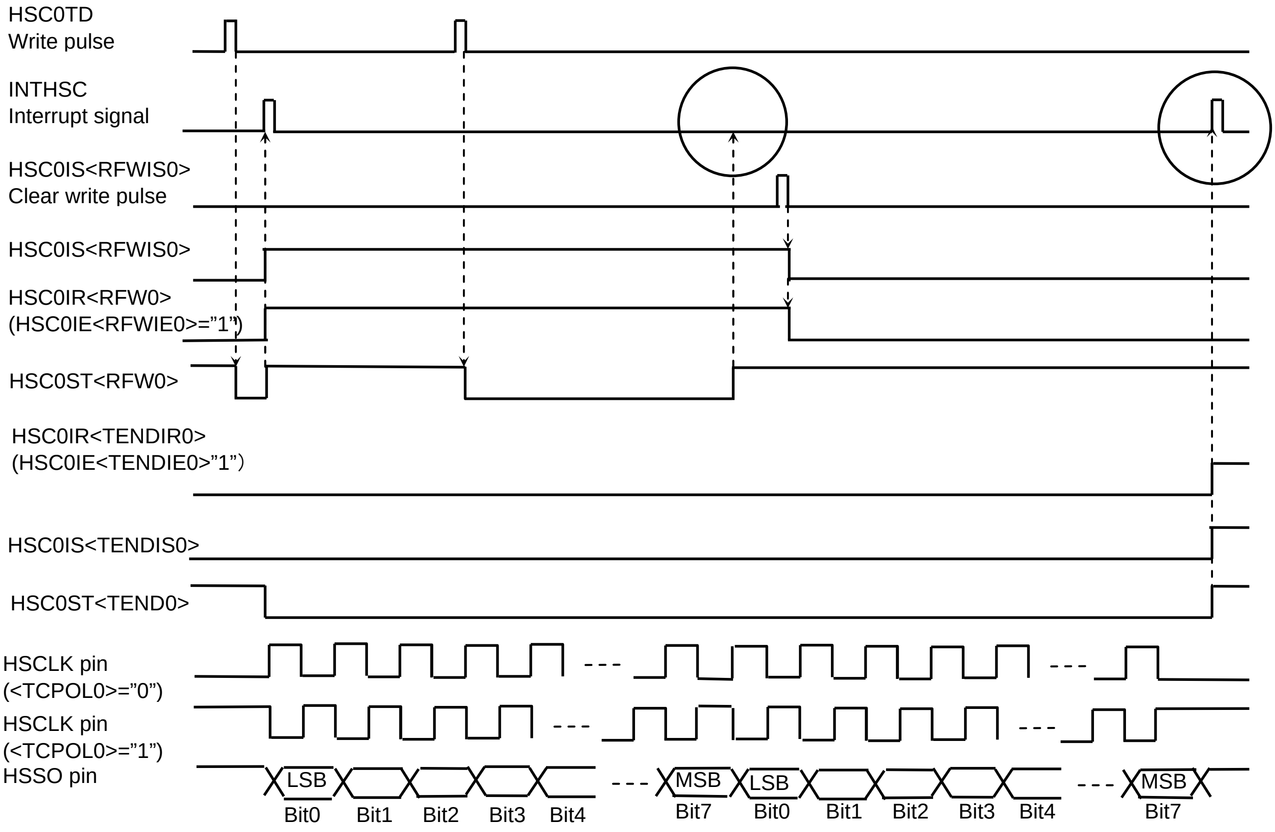


Figure 3.11.21 Transmission timing

In above condition, HSC0ST<RFW0> flag is set to "0" just after wrote transmission data. When data of HSC0TD register finish shifting to transmission register (HSC0TS), HSC0ST<RFW0> is set to "1", it is informed that can write next transmission data, start transmission clock and data from HSCLK pin and HSSO pin at same time with inform.

In this case, HSC0IS, HSC0IR change and INTHSC interrupt generate by synchronization to rising of HSC0ST<RFW0> flag. When HSC0IR register is setting to "1", interrupt is not generated even if HSC0ST<RFW0> was set to "1".

When finish transmission and lose data that must to transmit to HSC0TD register and HSC0TS register, transmission data and clock are stopped by setting "1" to HSC0ST<TEND0>, and INTHSC interrupt is generated at same time. In this case, if HSC0ST<TEND0> is set to "1" at different interrupt source, INTHSC is not generated. Therefore must to clear HSC0IS<RFW0> to "0".

- Setting condition 2:
  UNIT transmission in UNIT=8bit, LSB first

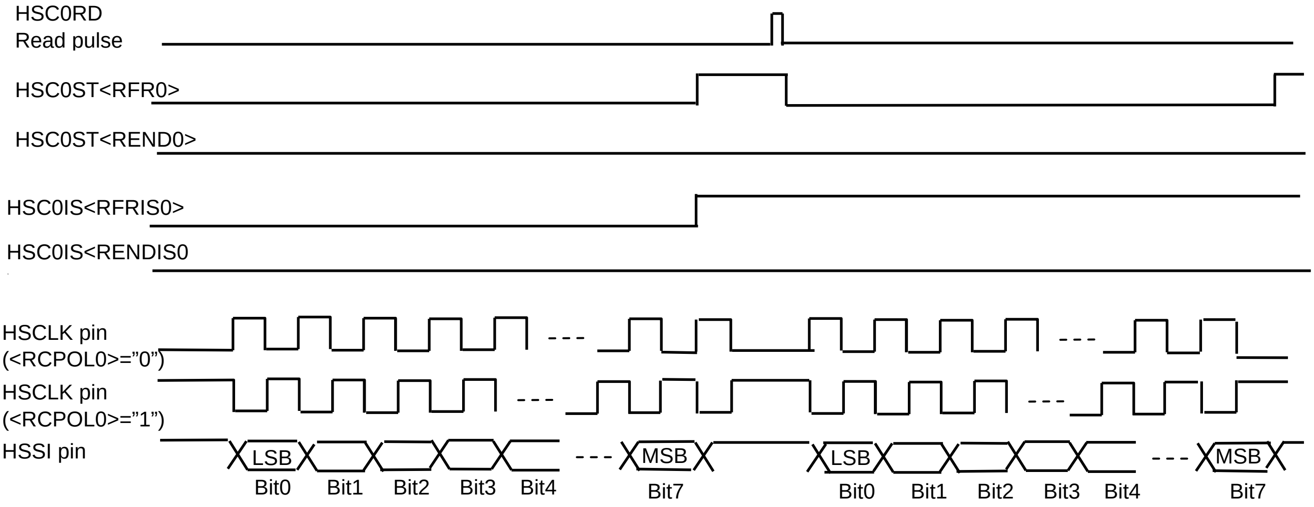


Figure 3.11.22 UNIT receiving (HSC0CT<RXUEN0>=1)

If set HSC0CT<RXUEN0> to "1" without valid receiving data to HSC0RD register (HSC0ST<RFR0>=0), UNIT receiving is started. When receiving is finished and stored receiving data to HSC0RD register, HSC0ST<RFR0> flag is set to "1", and inform that can read receiving data. Just after read HSC0RD register, HSC0ST<RFR0> flag is cleared to "0" and it start receiving next data automatically.

If be finished UNIT receiving, set HSC0CT<RXUEN0> to "0" after confirmed that HSC0ST<RFR0> was set to "1".

- Setting condition 3:

  Sequential receiving in UNIT=8 bit, LSB first

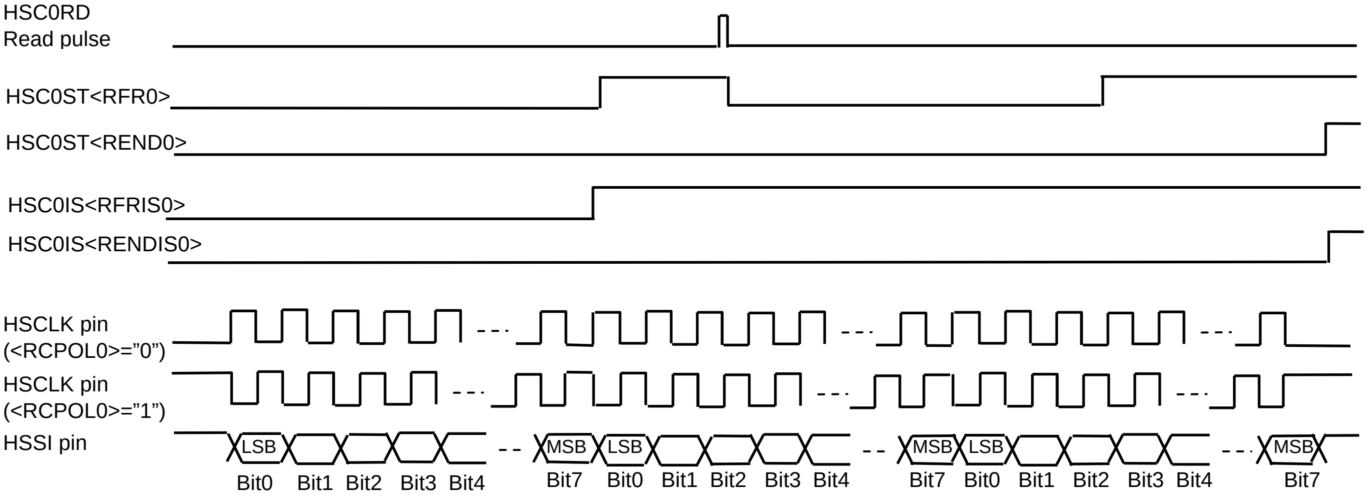


Figure 3.11.23 continuous receiving (HSC0CT<RXWEN0>=1)

If set HSC0CT<RXWEN0> to "1" without valid receiving data in HSC0RD register (HSC0ST<RFR0>=0), sequential receiving is started. When first receiving is finished and stored receiving data to HSC0RD register, HSC0ST<RFR0> flag is set to "1", and inform that can read receiving data. Sequential receiving is received until receiving data is stored to HSC0RD and HSC0RS registers If finished sequential receiving, set HSC0CT<RXWEN0> to "0" after confirmed that HSC0ST<REND0> was set to "1".

- Setting condition 4:

  Transmission by using micro DMA in UNIT=8bit, LSB first

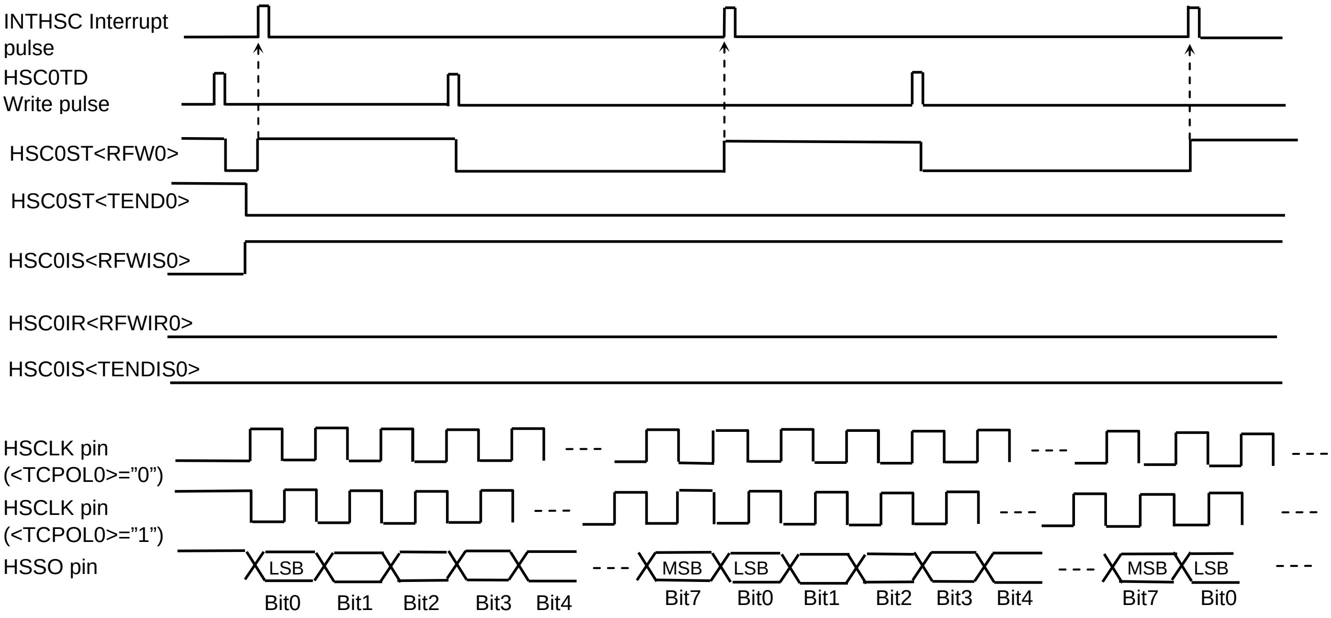


Figure 3.11.24 Micro DMA transmission (transmission)

If all bits of HSC0IE register are "0" and HSC0CT<DMAERFW0> is "1", transmission is started by writing transmission data to HSC0TD register.

If data of HSC0TD register is shifted to HSC0TS register and HSC0ST<RFW0> is set to "1" and can write next transmission data, INTHSC interrupt (RFW0 interrupt) is generated. By starting Micro DMA at this interrupt, can transmit sequential data automatically.

However, If transmit it at Micro DMA, set Micro DMA beforehand.

- Setting condition 5:

  Receiving by using micro DMA in UNIT=8bit, LSB first

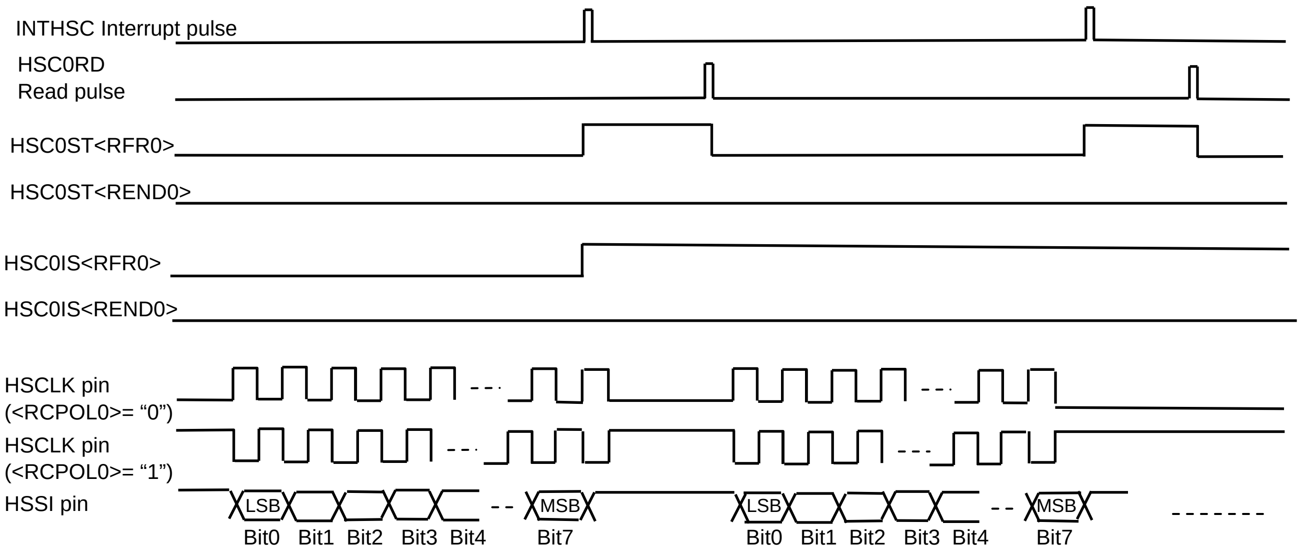


Figure 3.11.25 Micro DMA transmission (UNIT receiving (HSC0CT<RFUEN0>=1))

If all bits of HSC0IE register is "0" and HSC0CT<DMAERFR0> is "1", UNIT receiving is started by setting HSC0CT<RXUEN0> to "1". If receiving data is stored to HSC0RD register and can read receiving data, INTHSC interrupt (RFR0 interrupt) is generated. By starting Micro DMA at this interrupt, it can be received sequential data automatically.

However, If receive it at Micro DMA, set Micro DMA beforehand.

3.11.4 Example

Following is discription of HSC setting method.

(1) UNIT transmission

This example shows the case of transmission is executed by following setting, and it is generated INTHSC interrupt by finish transmission.

UNIT: 8bit
LSB first
Baud rate : f_{SYS}/8
Synchronous clock edge: Rising

Setting expample

```
    ld    (pffc), 0x38              ; Port setting PF3: HSSO, PF4: HSSI, PF5: HSCLK
    ld    (pfcr), 0x28              ; port setting PF3: HSSO, PF4: HSSI, PF5: HSCLK
    ld    (hscsel), 0x01            ; port setting PF3: HSSO, PF4: HSSI, PF5: HSCLK

    ldw   (hsc0ct), 0x0040          ; Set data length to 8bit
    ldw   (hsc0md), 0x2c43          ; System clock enable, baud rate selection: fSYS/8
                                    ; LSB first, synchronous clock edge setting: set to Rising

    ld    (hsc0ie), 0x08            ; Set to TEND0 interrupt enable
    ld    (intes1hsc), 0x10         ; Set INTHSC interrupt level to 1
    ei                              ; Interrupt enable (iff=0)

loop                                ; Confirm that transmission data register doesn't have no transmission data
    bit    1, (hsc0st)              ; <RFW0>=1 ?
    jr    z, loop

    ld    (hsc0td), 0x3a            ; Write Transmission data and Start transmission
    .
    .
    .
```

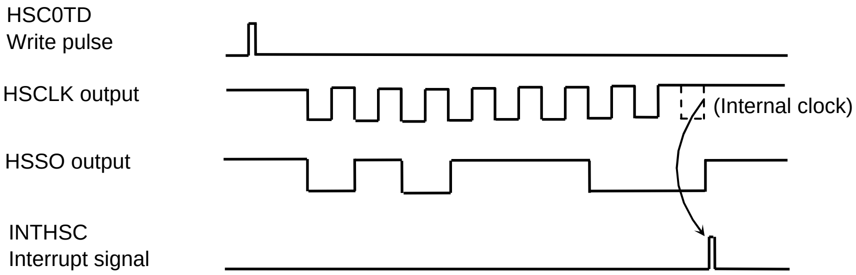


Figure 3.11.26 Example of UNIT transmission

(2) UNIT receiving

This example shows case of receiving is executed by following setting, and it is generated INTHSC interrupt by finish receiving.

UNIT: 8bit
LSB first
Baud rate selection : f_{SYS}/8
Synchronous clock edge: Rising

Setting example

```
ld     (pffc), 0x38            ; Port setting PF3: HSSO, PF4: HSSI, PF5: HSCLK
ld     (pfcr), 0x28            ; port setting PF3: HSSO, PF4: HSSI, PF5: HSCLK
ld     (hscsel), 0x01          ; port setting PF3: HSSO, PF4: HSSI, PF5: HSCLK

ldw    (hsc0ct), 0x0040        ; Set data length to 8bit
ldw    (hsc0md), 0x2c43        ; System clock enable, baud rate selection: fSYS/8
                               ; LSB first, synchronous clock edge setting: set to Rising

ld     (hsc0ie), 0x01          ; Set to RFR0 interrupt enable
ld     (intes1hsc), 0x10       ; Set INTHSC interrupt level to 1
ei                             ; Interrupt enable (iff=0)

set    0x0, (hsc0ct)           ; Start UNIT receiving
.
.
.
```

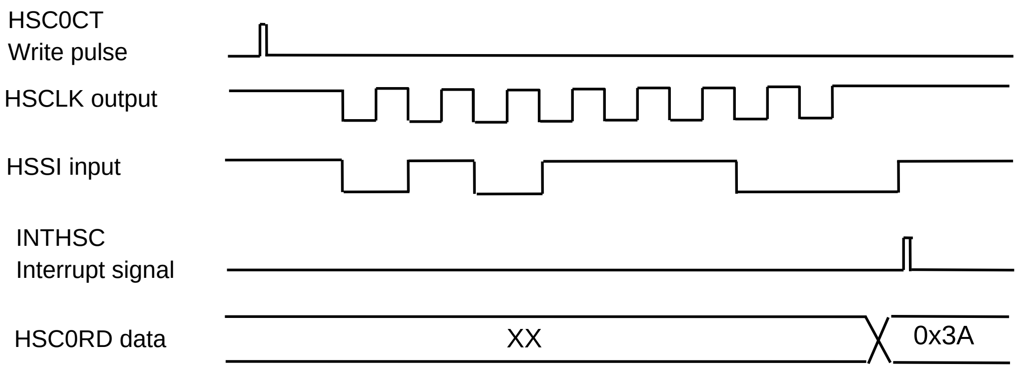


Figure 3.11.27 Example of UNIT receiving

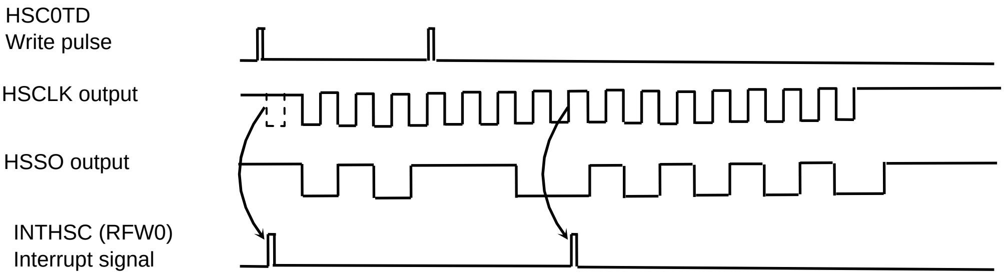
(3) Sequential transmission

This example shows case of transmission is executed by following setting, and it is executed 2byte sequential transmission.

UNIT: 8bit
LSB first
Baud rate selection: $f_{SYS}/8$
Synchronous clock edge: Rising

Setting example

```
    ld    (pffc), 0x38              ; Port setting PF3: HSSO, PF4: HSSI, PF5: HSCLK
    ld    (pfcr), 0x28              ; port setting PF3: HSSO, PF4: HSSI, PF5: HSCLK
    ld    (hscsel), 0x01            ; port setting PF3: HSSO, PF4: HSSI, PF5: HSCLK

    ldw    (hsc0ct), 0x0040         ; Set data length to 8bit
    ldw    (hsc0md), 0x2c43         ; System clock enable, baud rate selection: fSYS/8
                                    ; LSB first, synchronous clock edge setting: set to Rising

loop1:                    ; Confirm that transmission data register doesn't have no transmission data
    bit    1, (hsc0st)              ; <RFW0>=1 ?
    jr     z, loop1

    ld    (hsc0td), 0x3a            ; Write transmission data of first byte and start transmission

loop2                     ; Confirm that transmission data register doesn't have no-transmission data
    bit    1, (hsc0st)              ; <RFW0>=1 ?
    jr     z, loop2

    ld    (hsc0td), 0x55            ; Write transmission data of second byte

loop3:                    ; Confirm that transmission data register doesn't have no-transmission data
    bit    3, (hsc0st)              ; <TEND0>=1 ?
    jr     z, loop3
    .                               ; Finish transmission
    .
```



Note: Timing of this figure is an example. There is also that transmission interbal between first byte and sescond byte generate. (High baud rate etc.)

Figure 3.11.28 Example of sequential transmission

(4) Sequential receiving

This example shows case of receiving is executed by following setting, and it is executed 2byte sequential receiving.

UNIT: 8bit
LSB first
Baud rate selection: $f_{SYS}/8$
Synchronous clock edge: Rising

Setting example

```
    ld    (pffc), 0x38              ; Port setting PF3: HSSO, PF4: HSSI, PF5: HSCLK
    ld    (pfcr), 0x28              ; Port setting PF3: HSSO, PF4: HSSI, PF5: HSCLK
    ld    (hscsel), 0x01            ; Port setting PF3: HSSO, PF4: HSSI, PF5: HSCLK

    ldw    (hsc0ct), 0x0040         ; Set data length to 8bit
    ldw    (hsc0md), 0x2c43         ; System clock enable, baud rate selection: fSYS/8
                                    ; LSB first, synchronous clock edge setting: set to Rising
    set    0x01,(hsc0ct)            ; Start sequential receiving

loop1:                   ; Confirm that receiving data register has receiving data of first byte
    bit    0, (hsc0st)              ; <RFR0>=1 ?
    jr    z, loop1

loop2:                   ; Confirm that receiving data register has receiving data of second byte
    bit    2, (hsc0st)              ; <REND0>=1 ?
    jr    z, loop2

    res    0x01, (hsc0ct)           ; Sequential receiving disable

    ld    a, (hsc0rd)               ; Read receiving data of first byte

loop3:                   ; Confirm that receiving data of second byte is shifted from receiving data
                           shift register to receiving data register
    bit    0, (hsc0st)              ; <RFR0>=1 ?
    jr    z, loop3
    ld    w, (hsc0rd)               ; Read receiving data of second byte
```

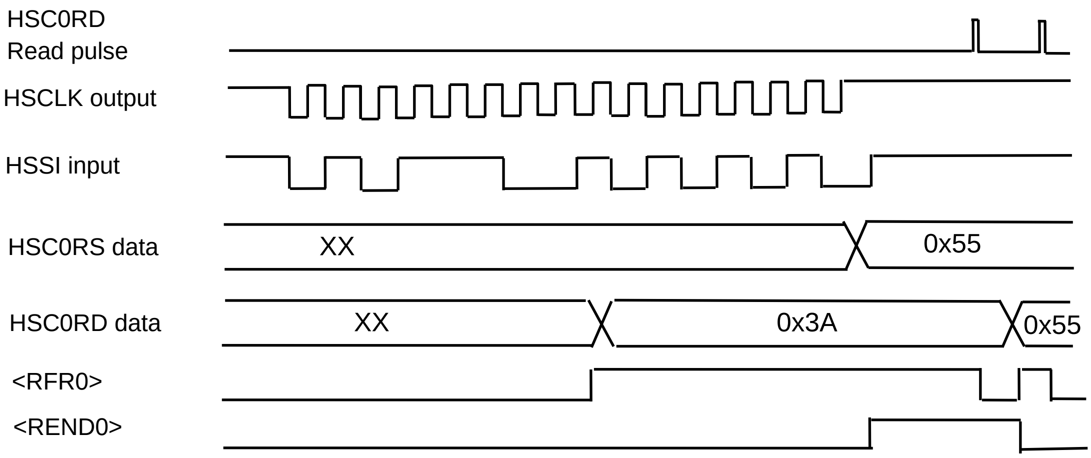


Figure 3.11.29 Example of sequential receiving

(5) Sequeintial Transmission by using micro DMA

This example shows case of sequential transmission of 4byte is executed at using micro DMA by following setting.

UNIT: 8bit
LSB first
Baud rate : $f_{SYS}/8$
Synchronous clock edge: Rising

Setting example

```
Main routine
;-- micro DMA setting --
    ld    (dma0v), 0x1D          ; Set micro DMA0 to INTHSC
    ld    wa, 0x0003             ; Set number of micro DMA transmission  to that number -1 (third time)
    ldc   dmac0, wa
    ld    a, 0x08                ; micro DMA mode setting: source INC mode, 1 byte transfer
    ldc   dmam0, a

    ld    xwa, 0x806000          ; Set source address
    ldc   dmas0, xwa
    ld    xwa, 0xC10             ; Set source address to HSC0TD register
    ldc   dmad0, xwa

;-- HSC setting --
    ld    (pffc), 0x38           ; Port setting PF3: HSSO, PF4: HSSI, PF5: HSCLK
    ld    (pfcr), 0x28           ; Port setting PF3: HSSO, PF4: HSSI, PF5: HSCLK
    ld    (hscsel), 0x01         ; Port setting PF3: HSSO, PF4: HSSI, PF5: HSCLK

    ldw   (hsc0ct), 0x0040       ; Set data length to 8bit
    ldw   (hsc0md), 0x2c43       ; System clock enable, baud rate selection: fSYS/8
                                 ; LSB first, synchronous clock edge setting: set to Rising

    ld    (hsc0ie), 0x00         ; Set to interrupt disable
    set   1, (hsc0ct+1)          ; Set micro DMA operation by RFW0 to enable
    ld    (intetc01), 0x01       ; Set INTTC0 interrupt level to 1
    ei                           ; Interrupt enable (iff=0)

loop1:                     ; Confirm that transmission data register doesn't have no transmission data
    bit   1, (hsc0st)            ; <RFW0>=1 ?
    jr    z, loop1

    ld    (hsc0td), 0x3a         ; Write Transmission data and Start transmission

Interrupt routine (INTTC0)
loop2:
    bit   1, (hsc0st)            ; <RFW0> = 1 ?
    jr    z, loop2
    bit   3, (hsc0st)            ; <TEND0> = 1 ?
    jr    z, loop2
    nop
```

(6) UNIT receiving by using micro DMA

This example shows case of UNIT receiving sequentially 4byte is executed at using micro DMA by following setting.

UNIT: 8bit
LSB first
Baud rate : $f_{SYS}/8$
Synchronous clock edge: Rising

Setting example

Main routine

```
;-- micro DMA setting --
    ld     (dma0v), 0x1D             ; Set micro DMA0 to INTHSC
    ld     wa, 0x0003                ; Set number of micro DMA transmission to that number -1 (third time)
    ldc    dmac0, wa
    ld     a, 0x00                   ; micro DMA mode setting: source INC mode, 1 byte transfer
    ldc    dmam0, a

    ld     xwa, 0xC12                ; Set source address to HSC0RD register
    ldc    dmas0, xwa
    ld     xwa, 0x807000             ; Set source address
    ldc    dmad0, xwa

;-- HSC setting --
    ld     (pffc), 0x38              ; Port setting PF3: HSSO, PF4: HSSI, PF5: HSCLK
    ld     (pfcr), 0x28              ; Port setting PF3: HSSO, PF4: HSSI, PF5: HSCLK
    ld     (hscsel), 0x01            ; Port setting PF3: HSSO, PF4: HSSI, PF5: HSCLK

    ldw    (hsc0ct), 0x0040          ; Set data length to 8bit
    ldw    (hsc0md), 0x2c43          ; System clock enable, baud rate selection: fSYS/8
                                     ; LSB first, synchronous clock edge setting: set to Rising

    ld     (hsc0ie), 0x00            ; Set to interrupt disable
    set    0, (hsc0ct+1)             ; Set micro DMA operation by RFR0 to enable
    ld     (intetc01), 0x01          ; Set INTTC0 interrupt level to 1
    ei                               ; Interrupt enable (iff=0)

    set    0x0, (hsc0ct)             ; Start UNIT receiving
```

Interrupt routine (INTTC0)

```
loop2:                               ; Wait receiving finish case of UNIT receiving
    bit    0, (hsc0st)               ; <RFR0> = 1 ?
    jr     z, loop2
    res    0, (hsc0ct)               ; UNIT receiving disable
    ld     a, (hsc0rd)               ; Read last receiving data
    Nop
```

3.12 Analog/Digital Converter

The TMP92FD23A incorporates a 10-bit successive approximation type analog/digital converter (AD converter) with 12-channel analog input.

Figure 3.12.1 is a block diagram of the AD converter. The 12-channel analog input pins (AN0 to AN11) are shared with the input only port (Port G and Port L) so they can be used as an input port.

Note: When IDLE2, IDLE1 or STOP mode is selected, as to reduce the power, with some timings the system may enter a stand-by 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.

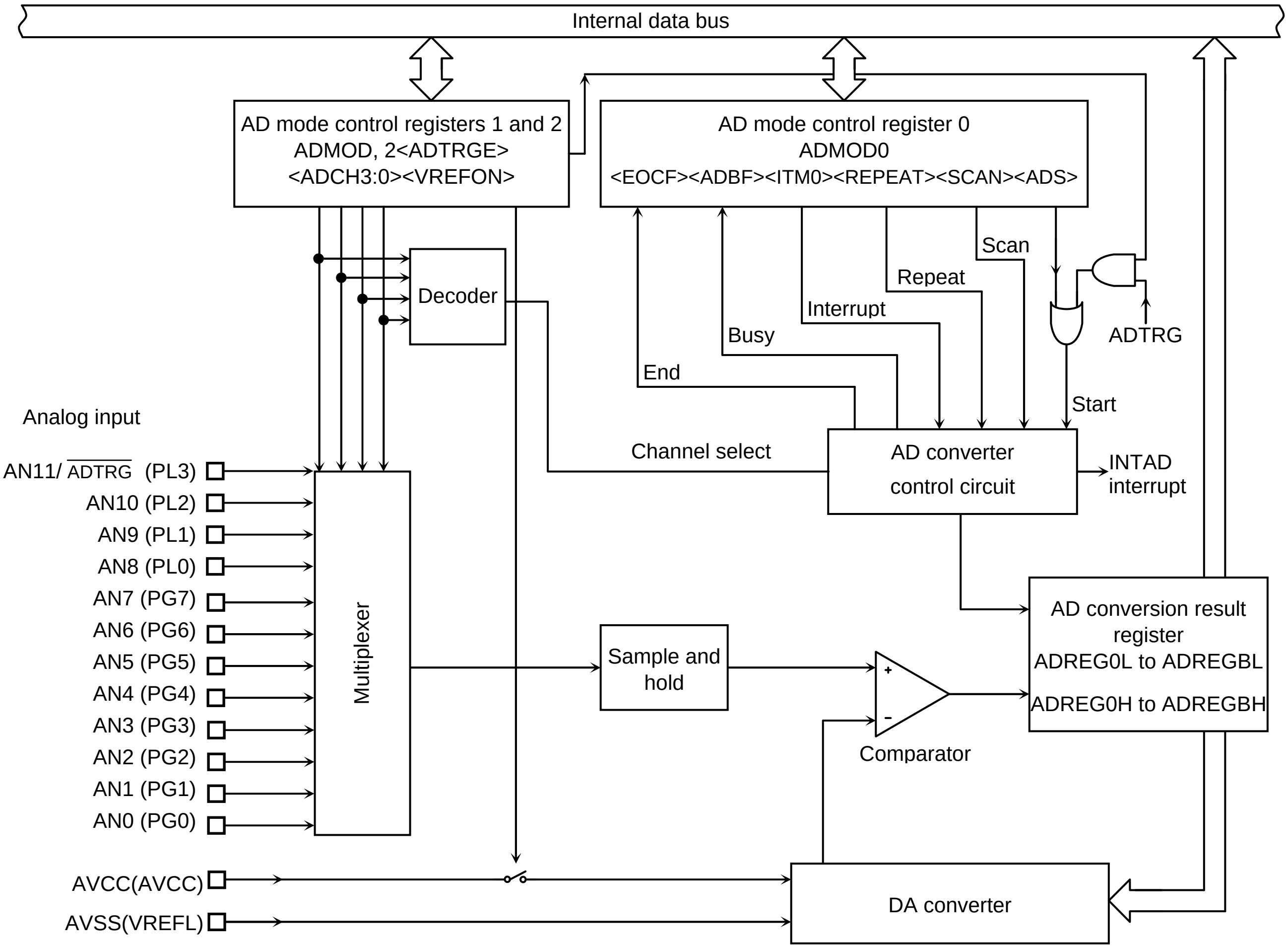


Figure 3.12.1 Block Diagram of AD Converter

3.12.1 Analog/Digital Converter Registers

The AD converter is controlled by the three AD mode control registers: ADMOD0, ADMOD1 and ADMOD2. The 24 AD conversion data result registers (ADREG0H/L to ADREGBH/L) store the results of AD conversion.

Figure 3.12.2 to Figure 3.12.10 show the registers related to the AD converter.

AD Mode Control Register 0

		7	6	5	4	3	2	1	0
ADMOD0	Bit symbol	EOCF	ADBF	–	–	ITM0	REPEAT	SCAN	ADS
(12B8H)	Read/Write	R		R/W					
	Reset State	0	0	0	0	0	0	0	0
	Function	AD conversion end flag 0:Conversion in progress 1:Conversion complete	AD conversion busy flag 0:Conversion stopped 1:Conversion in progress	Always write "0"	Always write "0"	Interrupt specification in conversion channel fixed repeat mode 0:Every conversion 1:Every fourth conversion	Repeat mode specification 0:Single conversion 1:Repeat conversion mode	Scan mode specification 0:Conversion channel fixed mode 1:Conversion channel scan mode	AD conversion start 0: Don't care 1:Start conversion Always "0" when read

AD conversion start

0	Don't care
1	Start AD conversion

Note: Always read as "0".

AD scan mode setting

0	AD conversion channel fixed mode
1	AD conversion channel scan mode

AD repeat mode setting

0	AD single conversion mode
1	AD repeat conversion mode

Specify AD conversion interrupt for channel fixed repeat conversion mode

	Channel fixed repeat conversion mode <SCAN> = "0", <REPEAT> = "1"
0	Generates interrupt every conversion.
1	Generates interrupt every fourth conversion.

AD conversion busy flag

0	AD conversion stopped
1	AD conversion in progress

AD conversion end flag

0	Before or during AD conversion
1	AD conversion complete

Figure 3.12.2 Register for AD Converter

AD Mode Control Register 1

ADMOD1
(12B9H)

	7	6	5	4	3	2	1	0
Bit symbol	VREFON	I2AD	–	–	ADCH3	ADCH2	ADCH1	ADCH0
Read/Write	R/W							
Reset State	0	0	0	0	0	0	0	0
Function	VREF application control 0: Off 1: On	IDLE2 0: Stop 1: Operate	Always write "0"	Always write "0"	Analog input channel selection			

Analog input channel selection

<SCAN> <ADCH3:0>	0 (Channel Fixed)	1 (Channel Scanned)
0000	AN0	AN0
0001	AN1	AN0 → AN1
0010	AN2	AN0 → AN1 → AN2
0011	AN3	AN0 → AN1 → AN2 → AN3
0100	AN4	AN0 → AN1 → AN2 → AN3 → AN4
0101	AN5	AN0 → AN1 → AN2 → AN3 → AN4 → AN5
0110	AN6	AN0 → AN1 → AN2 → AN3 → AN4 → AN5 → AN6
0111	AN7	AN0 → AN1 → AN2 → AN3 → AN4 → AN5 → AN6 → AN7
1000	AN8	AN0 → AN1 → AN2 → AN3 → AN4 → AN5 → AN6 → AN7 → AN8
1001	AN9	AN0 → AN1 → AN2 → AN3 → AN4 → AN5 → AN6 → AN7 → AN8→ AN9
1010	AN10	AN0 → AN1 → AN2 → AN3 → AN4 → AN5 → AN6 → AN7 → AN8→ AN9→ AN10
1011	AN11	AN0 → AN1 → AN2 → AN3 → AN4 → AN5 → AN6 → AN7 → AN8→ AN9→ AN10 → AN11
1100~1111	Please do not set up.	

IDLE2 control

0	Stopped
1	In operation

Control of application of reference voltage to AD converter

0	Off
1	On

Note: As pin AN11 also functions as the $\overline{\text{ADTRG}}$ input pin, do not set ADMOD1<ADCH3:0> = "1011" when using $\overline{\text{ADTRG}}$ with ADMOD2<ADTRGE> set to "1".

Figure 3.12.3 Register for AD Converter

AD Mode Control Register 2

ADMOD2 (12BAH)		7	6	5	4	3	2	1	0
	Bit symbol	–	–	–	–	–	–	–	ADTRGE
	Read/Write	R/W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Always write “0”	Always write “0”	Always write “0”	Always write “0”	Always write “0”	Always write “0”	Always write “0”	AD external trigger start control 0: Disable 1: Enable

AD conversion start control by external trigger ($\overline{\text{ADTRG}}$ input)

0	Disabled
1	Enabled

Figure 3.12.4 Register for AD Converter

AD Conversion Result Register 0 Low

ADREG0L
(12A0H)

	7	6	5	4	3	2	1	0
Bit symbol	ADR01	ADR00						ADR0RF
Read/Write	R							R
Reset State	Undefined							0
Function	Stores lower 2 bits of AD conversion result.							AD conversion data storage flag 1:Conversion result stored

AD Conversion Result Register 0 High

ADREG0H
(12A1H)

	7	6	5	4	3	2	1	0
Bit symbol	ADR09	ADR08	ADR07	ADR06	ADR05	ADR04	ADR03	ADR02
Read/Write	R							
Reset State	Undefined							
Function	Stores upper 8 bits of AD conversion result.							

AD Conversion Result Register 1 Low

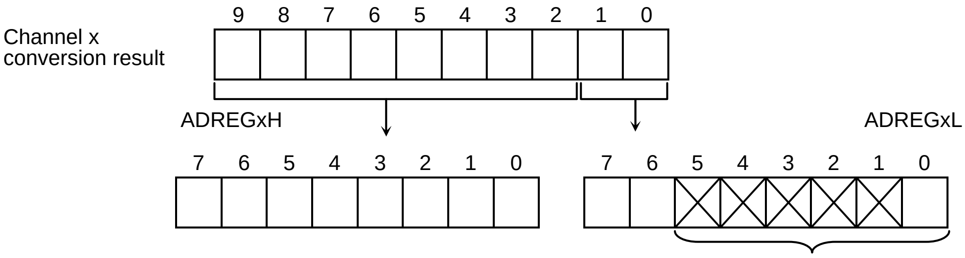
ADREG1L
(12A2H)

	7	6	5	4	3	2	1	0
Bit symbol	ADR11	ADR10						ADR1RF
Read/Write	R							R
Reset State	Undefined							0
Function	Stores lower 2 bits of AD conversion result.							AD conversion data storage flag 1:Conversion result stored

AD Conversion Result Register 1 High

ADREG1H
(12A3H)

	7	6	5	4	3	2	1	0
Bit symbol	ADR19	ADR18	ADR17	ADR16	ADR15	ADR14	ADR13	ADR12
Read/Write	R							
Reset State	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 3.12.5 Register for AD Converter

AD Conversion Result Register 2 Low

ADREG2L
(12A4H)

	7	6	5	4	3	2	1	0
Bit symbol	ADR21	ADR20						ADR2RF
Read/Write	R							R
Reset State	Undefined							0
Function	Stores lower 2 bits of AD conversion result.							AD conversion data storage flag 1:Conversion result stored

AD Conversion Result Register 2 High

ADREG2H
(12A5H)

	7	6	5	4	3	2	1	0
Bit symbol	ADR29	ADR28	ADR27	ADR26	ADR25	ADR24	ADR23	ADR22
Read/Write	R							
Reset State	Undefined							
Function	Stores upper 8 bits of AD conversion result.							

AD Conversion Result Register 3 Low

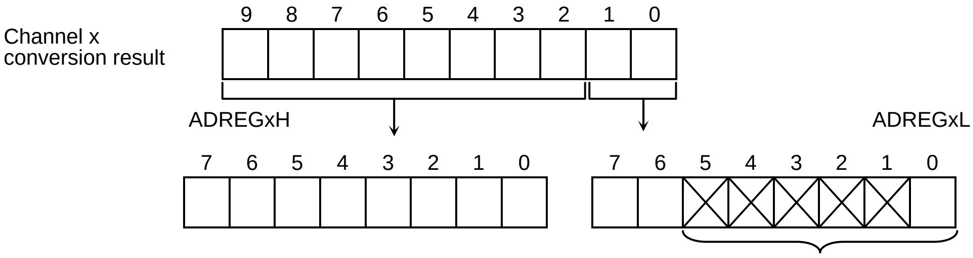
ADREG3L
(12A6H)

	7	6	5	4	3	2	1	0
Bit symbol	ADR31	ADR30						ADR3RF
Read/Write	R							R
Reset State	Undefined							0
Function	Stores lower 2 bits of AD conversion result.							AD conversion data storage flag 1: Conversion result stored

AD Conversion Result Register 3 High

ADREG3H
(12A7H)

	7	6	5	4	3	2	1	0
Bit symbol	ADR39	ADR38	ADR37	ADR36	ADR35	ADR34	ADR33	ADR32
Read/Write	R							
Reset State	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 3.12.6 Register for AD Converter

AD Conversion Result Register 4 Low

ADREG4L (12A8H)

	7	6	5	4	3	2	1	0
Bit symbol	ADR41	ADR40						ADR0RF
Read/Write	R							R
Reset State	Undefined							0
Function	Stores lower 2 bits of AD conversion result.							AD conversion data storage flag 1:Conversion result stored

AD Conversion Result Register 4 High

ADREG4H (12A9H)

	7	6	5	4	3	2	1	0
Bit symbol	ADR49	ADR48	ADR47	ADR46	ADR45	ADR44	ADR43	ADR42
Read/Write	R							
Reset State	Undefined							
Function	Stores upper 8 bits of AD conversion result.							

AD Conversion Result Register 5 Low

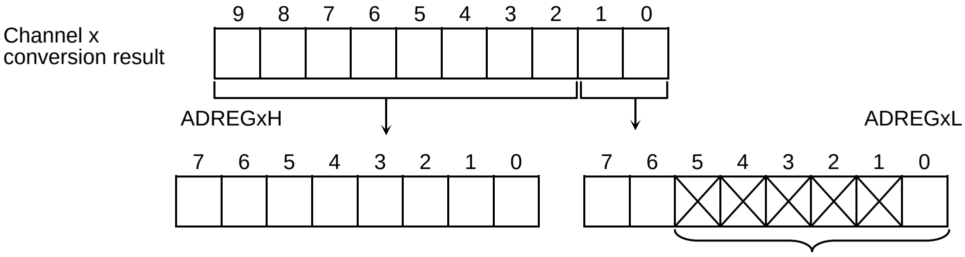
ADREG5L (12AAH)

	7	6	5	4	3	2	1	0
Bit symbol	ADR51	ADR50						ADR5RF
Read/Write	R							R
Reset State	Undefined							0
Function	Stores lower 2 bits of AD conversion result.							AD conversion storage flag 1:Conversion result stored

AD Conversion Result Register 5 High

ADREG5H (12ABH)

	7	6	5	4	3	2	1	0
Bit symbol	ADR59	ADR58	ADR57	ADR56	ADR55	ADR54	ADR53	ADR52
Read/Write	R							
Reset State	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 3.12.7 Register for AD Converter

AD Conversion Result Register 6 Low

		7	6	5	4	3	2	1	0
ADREG6L (12ACH)	Bit symbol	ADR61	ADR60						ADR6RF
	Read/Write	R							R
	Reset State	Undefined							0
	Function	Stores lower 2 bits of AD conversion result.							AD conversion data storage flag 1: Conversion result stored

AD Conversion Result Register 6 High

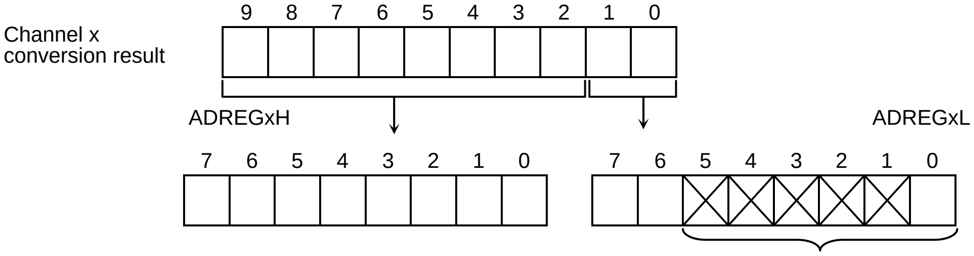
		7	6	5	4	3	2	1	0
ADREG6H (12ADH)	Bit symbol	ADR69	ADR68	ADR67	ADR66	ADR65	ADR64	ADR63	ADR62
	Read/Write	R							
	Reset State	Undefined							
	Function	Stores upper 8 bits of AD conversion result.							

AD Conversion Result Register 7 Low

		7	6	5	4	3	2	1	0
ADREG7L (12AEH)	Bit symbol	ADR71	ADR70						ADR7RF
	Read/Write	R							R
	Reset State	Undefined							0
	Function	Stores lower 2 bits of AD conversion result.							AD conversion data storage flag 1: Conversion result stored

AD Conversion Result Register 7 High

		7	6	5	4	3	2	1	0
ADREG7H (12AFH)	Bit symbol	ADR79	ADR78	ADR77	ADR76	ADR75	ADR74	ADR73	ADR72
	Read/Write	R							
	Reset State	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 3.12.8 Register for AD Converter

AD Conversion Result Register 8 Low

ADREG8L (12B0H)		7	6	5	4	3	2	1	0
	Bit symbol	ADR81	ADR80						ADR8RF
	Read/Write	R							R
	Reset State	Undefined							0
	Function	Stores lower 2 bits of AD conversion result.							AD conversion data storage flag 1: Conversion result stored

AD Conversion Result Register 8 High

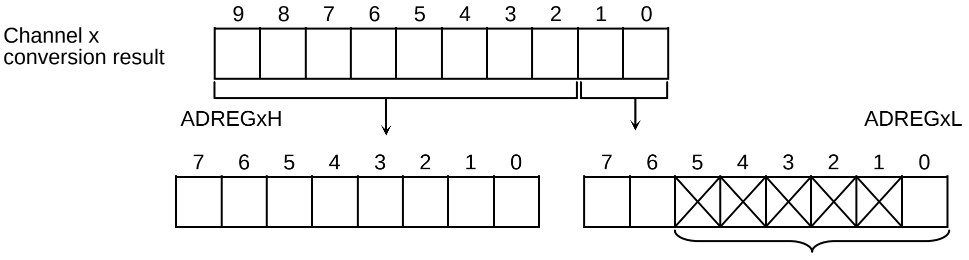
ADREG8H (12B1H)		7	6	5	4	3	2	1	0
	Bit symbol	ADR89	ADR88	ADR87	ADR86	ADR85	ADR84	ADR83	ADR82
	Read/Write	R							
	Reset State	Undefined							
	Function	Stores upper 8 bits of AD conversion result.							

AD Conversion Result Register 9 Low

ADREG9L (12B2H)		7	6	5	4	3	2	1	0
	Bit symbol	ADR91	ADR90						ADR9RF
	Read/Write	R							R
	Reset State	Undefined							0
	Function	Stores lower 2 bits of AD conversion result.							AD conversion storage flag 1: Conversion result stored

AD Conversion Result Register 9 High

ADREG9H (12B3H)		7	6	5	4	3	2	1	0
	Bit symbol	ADR99	ADR98	ADR97	ADR96	ADR95	ADR94	ADR93	ADR92
	Read/Write	R							
	Reset State	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 3.12.9 Register for AD Converter

AD Conversion Result Register A Low

ADREGAL
(12B4H)

	7	6	5	4	3	2	1	0
Bit symbol	ADRA1	ADRA0						ADRARF
Read/Write	R							R
Reset State	Undefined							0
Function	Stores lower 2 bits of AD conversion result.							AD conversion data storage flag 1: Conversion result stored

AD Conversion Result Register A High

ADREGAH
(12B5H)

	7	6	5	4	3	2	1	0
Bit symbol	ADRA9	ADRA8	ADRA7	ADRA6	ADRA5	ADRA4	ADRA3	ADRA2
Read/Write	R							
Reset State	Undefined							
Function	Stores upper 8 bits of AD conversion result.							

AD Conversion Result Register B Low

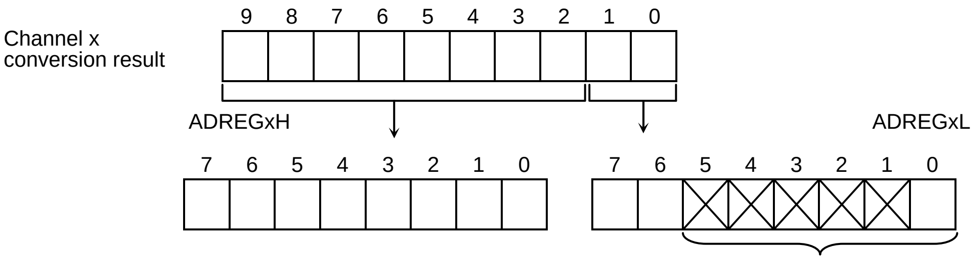
ADREGBL
(12B6H)

	7	6	5	4	3	2	1	0
Bit symbol	ADRB1	ADRB0						ADRBRF
Read/Write	R							R
Reset State	Undefined							0
Function	Stores lower 2 bits of AD conversion result.							AD conversion data storage flag 1: Conversion result stored

AD Conversion Result Register B High

ADREGBH
(12B7H)

	7	6	5	4	3	2	1	0
Bit symbol	ADRB9	ADRB8	ADRB7	ADRB6	ADRB5	ADRB4	ADRB3	ADRB2
Read/Write	R							
Reset State	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 3.12.10 Register for AD Converter

3.12.2 Description of Operation

(1) Analog reference voltage

A high level analog reference voltage is applied to the AVCC pin; a low level analog reference voltage is applied to the AVSS pin. To perform AD conversion, the reference voltage, the difference between AVCC and AVSS, is 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 AVCC and AVSS, write 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 3 μs until the internal reference voltage stabilizes (this is not related to fc), 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<ADCH1:0> selects one of the input pins AN0 to AN3 as the input channel.
- In analog input channel scan mode (ADMOD0<SCAN> = "1")
  Setting ADMOD1<ADCH1:0> selects one of the four scan modes.

Table 3.12.1 illustrates analog input channel selection in each operation mode.

On a reset, ADMOD0<SCAN> is set to 0 and ADMOD1<ADCH3:0> is initialized to "00". 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 3.12.1 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	AN0 → AN1 → AN2 → AN3 → AN4
0101	AN5	AN0 → AN1 → AN2 → AN3 → AN4 → AN5
0110	AN6	AN0 → AN1 → AN2 → AN3 → AN4 → AN5 → AN6
0111	AN7	AN0 → AN1 → AN2 → AN3 → AN4 → AN5 → AN6 → AN7
1000	AN8	AN0 → AN1 → AN2 → AN3 → AN4 → AN5 → AN6 → AN7 → AN8
1001	AN9	AN0 → AN1 → AN2 → AN3 → AN4 → AN5 → AN6 → AN7 → AN8→ AN9
1010	AN10	AN0 → AN1 → AN2 → AN3 → AN4 → AN5 → AN6 → AN7 → AN8→ AN9→ AN10
1011	AN11	AN0 → AN1 → AN2 → AN3 → AN4 → AN5 → AN6 → AN7 → AN8→ AN9→ AN10→ AN11

(3) Starting AD conversion

To start AD conversion, write a “1” to ADMOD0<ADS> in AD mode control register “0” or ADMOD2<ADTRGE> in AD mode control register 2, and input falling edge on $\overline{\text{ADTRG}}$ pin. When AD conversion starts, the AD conversion busy flag ADMOD0<ADBF> will be set to “1”, indicating that AD conversion is in progress.

During AD conversion, a falling edge input on the $\overline{\text{ADTRG}}$ 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.), write “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 3.12.2 shows the relationship between the AD conversion modes and interrupt requests.

Table 3.12.2 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 forth 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 (4.2μs at f_{SYS} = 20 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 (ADREG0H/L to ADREGBH/L) store the results of AD conversion. (ADREG0H/L to ADREGBH/L are read-only registers.)

In channel fixed repeat conversion mode, the conversion results are stored successively in registers from ADREG0H/L to ADREGBH/L. In other modes from AN0 to AN11 conversion results are stored in from ADREG0H/L to ADREGBH/L respectively.

Table 3.12.3 shows the correspondence between the analog input channels and the registers which are used to hold the results of AD conversion.

Table 3.12.3 Correspondence between Analog Input Channels and AD Conversion Result Registers

Analog Input Channel (Port G/Port L)	AD Conversion Result Register	
	Conversion Modes Other than at Right	Channel Fixed Repeat Conversion Mode ADMOD0<ITM0> = “1”
AN0	ADREG0H/L	
AN1	ADREG1H/L	
AN2	ADREG2H/L	
AN3	ADREG3H/L	ADREG0H/L
AN4	ADREG4H/L	↓
AN5	ADREG5H/L	ADREG1H/L
AN6	ADREG6H/L	↓
AN7	ADREG7H/L	ADREG2H/L
AN8	ADREG8H/L	↓
AN9	ADREG9H/L	ADREG3H/L
AN10	ADREGAH/L	
AN11	ADREGBH/L	

<ADRxRF>, 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<EOCF> to “0”.

Setting example:

1. Convert the analog input voltage on the AN3 pin and write the result, to memory address 2800H using the AD interrupt (INTAD) processing routine.

Main routine:

		7	6	5	4	3	2	1	0	
INTEPAD	←	X	−	−	−	X	1	0	0	Enable INTAD and set it to interrupt level 4.
ADMOD1	←	1	1	0	0	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	←	ADREG3H/L	Read value of ADREG3L and ADREG3H into 16-bits general-purpose register WA.
WA	←	> > 6	Shift contents read into WA six times to right and zero fill upper bits.
(2800H)	←	WA	Write contents of WA to memory address 2800H.

2. This example repeatedly converts the analog input voltages on the three pins AN0, AN1 and AN2, using channel scan repeat conversion mode.

INTEPAD	←	X	−	−	−	X	0	0	0	Disable INTAD.
ADMOD1	←	1	1	0	0	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

3.13 Watchdog Timer (Runaway detection timer)

The TMP92FD23A contains a watchdog timer of runaway detecting.

The watchdog timer (WDT) is used to return the CPU to the 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.)

3.13.1 Configuration

Figure 3.13.1 is a block diagram of the watchdog timer (WDT).

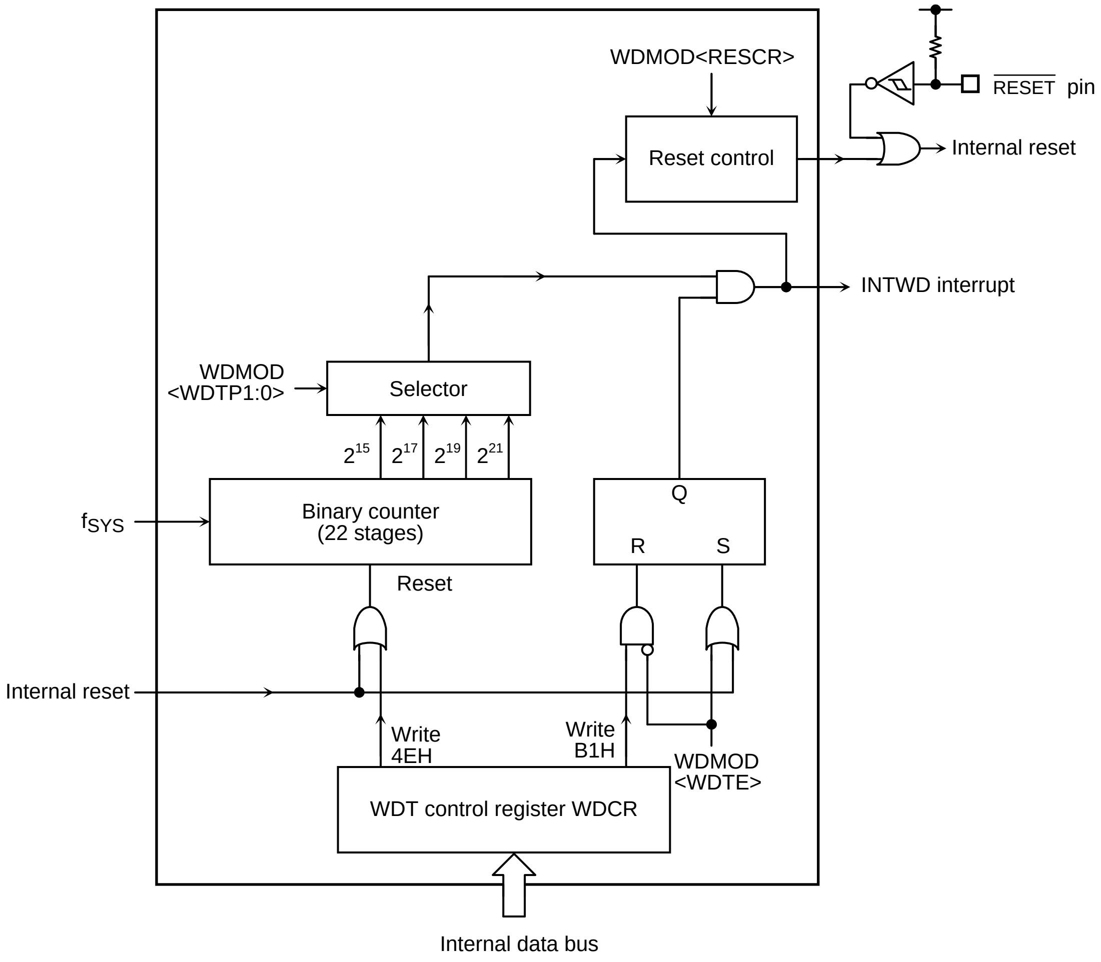


Figure 3.13.1 Block Diagram of Watchdog Timer

Note: It needs to care designing the total machine set, because Watchdog timer can’t operate completely by external noise.

3.13.2 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 cleared "0" 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-malfunction program.

The watchdog timer begins operating immediately on release of the watchdog timer reset.

The watchdog timer is halted in IDLE1 or STOP mode.

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.

The watchdog timer consists of a 22-stage binary counter which uses the clock f_{SYS} as the input clock. The binary counter can output $2^{15}/f_{SYS}$, $2^{17}/f_{SYS}$, $2^{19}/f_{SYS}$ and $2^{21}/f_{SYS}$.

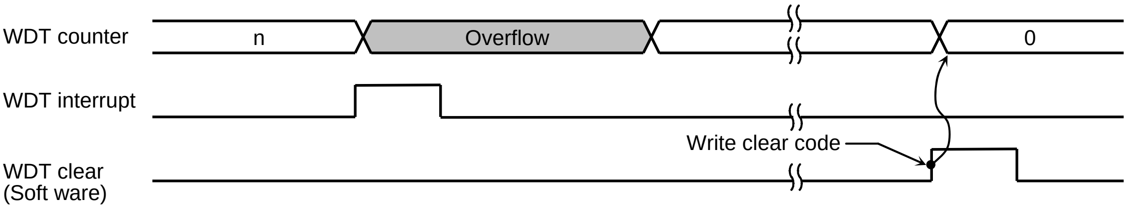


Figure 3.13.2 Normal Mode

The runaway detection result can also be connected to the reset pin internally.

In this case, the reset time will be between 22 and 29 system clocks (70.4 to 92.8 μs at f_{OSCH} = 10 MHz) as shown in Figure 3.13.3. After a reset, the f_{SYS} clock is $f_{FPH}/2$, where f_{FPH} is generated by dividing the high-speed oscillator clock (f_{OSCH}) by sixteen through the clock gear function

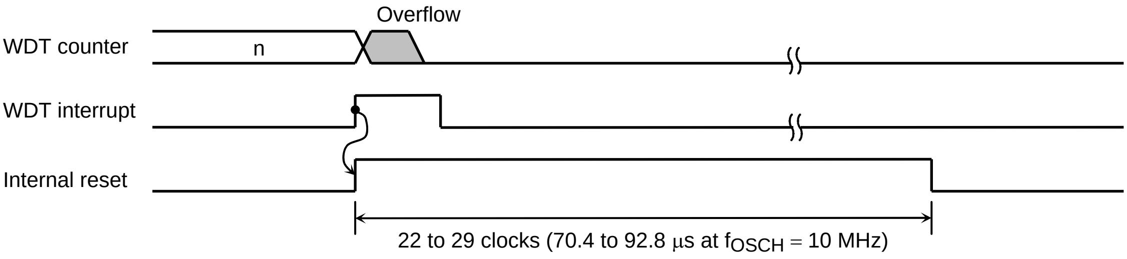


Figure 3.13.3 Reset Mode

3.13.3 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 <WDTP1:0>

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 time for WDT is $2^{15}/f_{SYS}$ [s].

2. Watchdog timer enable/disable control register <WDTE>

At reset, the WDMOD<WDTE> is initialized to 1, enabling the watchdog timer.

To disable the watchdog timer, it is necessary to set 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" at 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.

WDCR	←	0	1	0	0	1	1	1	0	Write the clear code (4EH).
WDMOD	←	0	−	−	X	0	−	−	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.

WDMOD
(1300H)

	7	6	5	4	3	2	1	0
Bit symbol	WDTE	WDTP1	WDTP0		–	I2WDT	RESCR	–
Read/Write	R/W				R/W			
Reset State	1	0	0		0	0	0	0
Function	WDT control 1: Enable	Select detecting time 00: $2^{15}/f_{SYS}$ 01: $2^{17}/f_{SYS}$ 10: $2^{19}/f_{SYS}$ 11: $2^{21}/f_{SYS}$			Always write “0”	IDLE2 0: Stop 1: Operate	1: Internally connects WDT out to the reset pin	Always write “0”

Watchdog timer out control

0	–
1	Connects WDT out to a reset

IDLE2 control

0	Stop
1	Operation

Watchdog timer detection time

00	$2^{15}/f_{SYS}$ (Approximately 1.64 ms at f_{SYS} = 20 MHz)
01	$2^{17}/f_{SYS}$ (Approximately 6.55 ms at f_{SYS} = 20 MHz)
10	$2^{19}/f_{SYS}$ (Approximately 26.2 ms at f_{SYS} = 20 MHz)
11	$2^{21}/f_{SYS}$ (Approximately 104.9 ms at f_{SYS} = 20 MHz)

Watchdog timer enable/disable control

0	Disabled
1	Enabled

Figure 3.13.4 Watchdog Timer Mode Register

WDCR
(1301H)
Read
-modify
-write
instruction is
prohibited

	7	6	5	4	3	2	1	0
Bit symbol	–							
Read/Write	W							
Reset State	–							
Function	B1H: WDT disable code 4EH: WDT clear code							

WDT disable/clear control

B1H	Disable code
4EH	Clear code
Others	Don’t care

Figure 3.13.5 Watchdog Timer Control Register

3.14 Special timer for CLOCK

The TMP92FD23A includes a timer which is used for a clock operation.

An interrupt (INTRTC) can be generated each 0.0625[s] or 0.125[s] or 0.25[s] or 0.50[s] by using a low-frequency clock of 32.768 kHz. A clock function can be easily used.

Special timer for Clock can operate in all modes in which a low-frequency oscillation is operated. In addition, INTRTC can return from each standby mode except STOP mode.

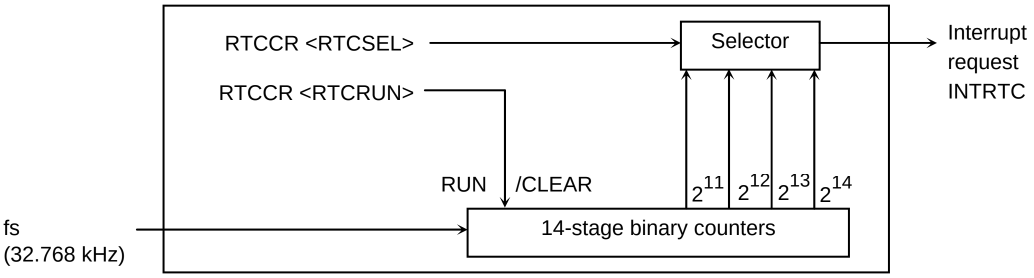


Figure 3.14.1 Block Diagram for Special timer for CLOCK

The Special timer for CLOCK is controlled by Special timer for CLOCK control register (RTCCR).

Figure 3.14.2 shows the timer for real time clock control register.

		7	6	5	4	3	2	1	0
RTCCR (1310H)	Bit Symbol	–					RTCSEL1	RTCSEL0	RTCRUN
	Read/Write	R/W					R/W		
	Reset State	0					0	0	0
	Function	Always write “0”					00 : 2^{14}/fs 01 : 2^{13}/fs 10 : 2^{12}/fs 11 : 2^{11}/fs		0 : Stop & Clear 1 : Run

Counting operation

0	Stop & clear
1	Count

Interrupt generation cycle
(fs = 32.768 kHz)

00	0.50s
11	0.25s
10	0.125s
11	0.0625s

Figure 3.14.2 Register for Special timer for CLOCK

3.15 Program patch logic

The TMP92FD23A has a program patch logic, which enables the user to fix the program code in the Internal ROM. Patch program must be read into Internal RAM from external memory during the startup routine.

Up to eight 4-byte sequences or banks (32-bytes in total) can be replaced with patch code. More significant code correction can be performed by replacing program code with 1-byte instruction code which generates a software interrupt (SWI) to make a branch to a specified location in the Internal RAM area.

The program patch logic only compares addresses in the Internal ROM area; it cannot fix the program code in the Internal peripheral, Internal RAM and external ROM areas.

Each of eight banks is independently programmable, and functionally equivalent. In the following sections, any references to bank0 also apply to other banks.

3.15.1 Block Diagram

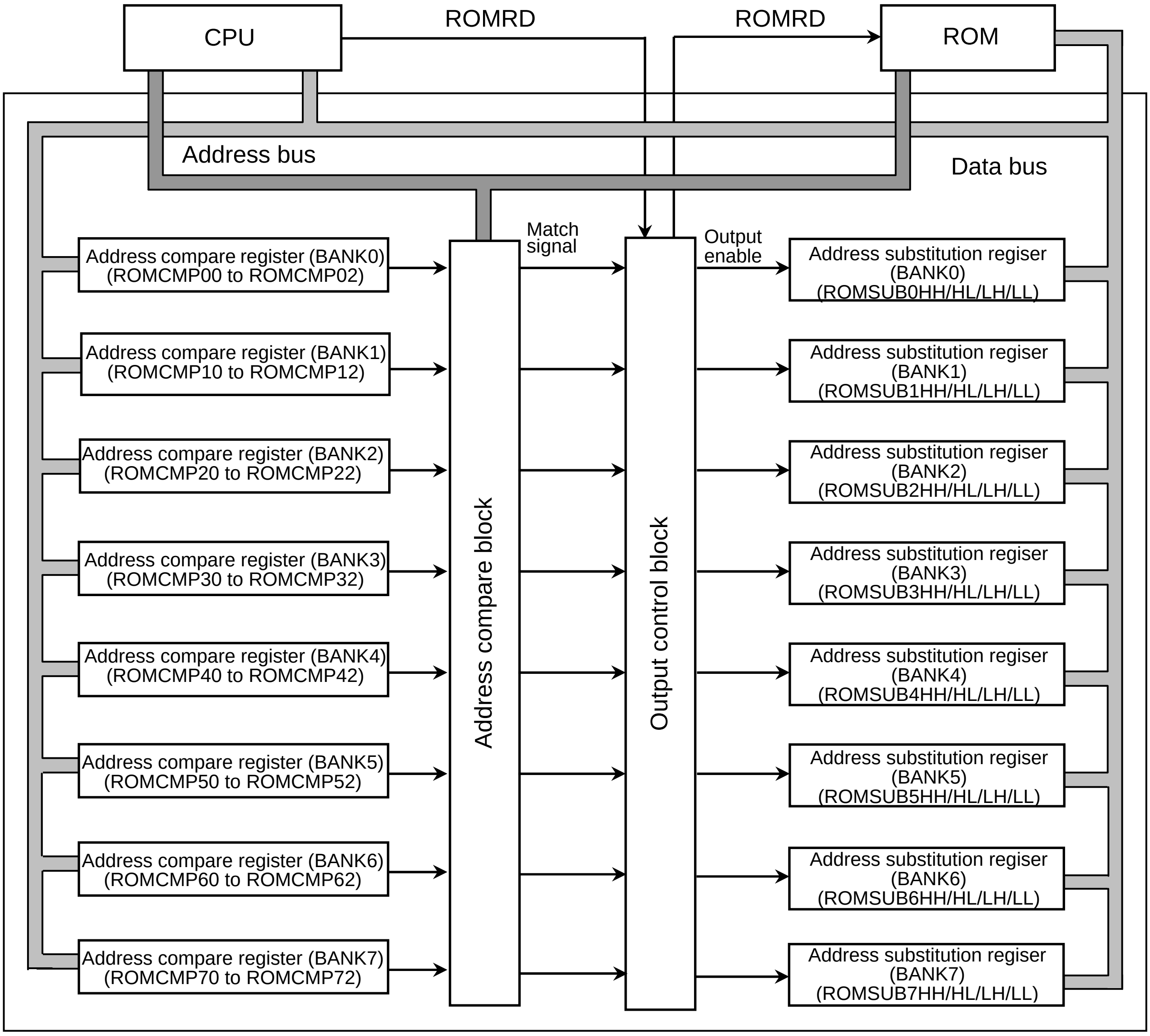

CPU
ROMRD
ROMRD
ROM
Address bus
Data bus
Match signal
Output enable
Address compare register (BANK0) (ROMCMP00 to ROMCMP02)
Address compare register (BANK1) (ROMCMP10 to ROMCMP12)
Address compare register (BANK2) (ROMCMP20 to ROMCMP22)
Address compare register (BANK3) (ROMCMP30 to ROMCMP32)
Address compare register (BANK4) (ROMCMP40 to ROMCMP42)
Address compare register (BANK5) (ROMCMP50 to ROMCMP52)
Address compare register (BANK6) (ROMCMP60 to ROMCMP62)
Address compare register (BANK7) (ROMCMP70 to ROMCMP72)
Address compare block
Output control block
Address substitution regiser (BANK0) (ROMSUB0HH/HL/LH/LL)
Address substitution regiser (BANK1) (ROMSUB1HH/HL/LH/LL)
Address substitution regiser (BANK2) (ROMSUB2HH/HL/LH/LL)
Address substitution regiser (BANK3) (ROMSUB3HH/HL/LH/LL)
Address substitution regiser (BANK4) (ROMSUB4HH/HL/LH/LL)
Address substitution regiser (BANK5) (ROMSUB5HH/HL/LH/LL)
Address substitution regiser (BANK6) (ROMSUB6HH/HL/LH/LL)
Address substitution regiser (BANK7) (ROMSUB7HH/HL/LH/LL)

Figure 3.15.1 Program Patch Logic Diagram

Note: Don't set the same value to an address compare register (Bank0 to 7).

3.15.2 SFR Descriptions

The program patch logic consists of eight banks (0 to 7). Each bank is provided with 3-bytes of address compare registers (ROMCMP00 to ROMCMP72) and 4-bytes of address substitution registers (ROMSUBLL, ROMSUBLH, ROMSUBHL and ROMSUBHH).

BANK0 Address Compare Register 0

		7	6	5	4	3	2	1	0
ROMCMP00 (1400H)	Bit symbol	ROMC07	ROMC06	ROMC05	ROMC04	ROMC03	ROMC02		
	Read/Write	W							
	Reset State	0	0	0	0	0	0		
	Function	Target ROM address (Lower 6 bits)							

BANK0 Address Compare Register 1

		7	6	5	4	3	2	1	0
ROMCMP01 (1401H)	Bit symbol	ROMC15	ROMC14	ROMC13	ROMC12	ROMC11	ROMC10	ROMC09	ROMC08
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Target ROM address (Middle 8 bits)							

BANK0 Address Compare Register 2

		7	6	5	4	3	2	1	0
ROMCMP02 (1402H)	Bit symbol	ROMC23	ROMC22	ROMC21	ROMC20	ROMC19	ROMC18	ROMC17	ROMC16
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Target ROM address (Upper 8 bits)							

Note 1: The ROMCMP00, ROMCMP01, and ROMCMP02 registers do not support read-modify-write operation.

Note 2: Bit0 and Bit1 of the ROMCMP00 is read as undefined.

Figure 3.15.2 Address Compare Registers (Bank0)

BANK1 Address Compare Register 0

		7	6	5	4	3	2	1	0
ROMCMP10	Bit symbol	ROMC07	ROMC06	ROMC05	ROMC04	ROMC03	ROMC02		
(1408H)	Read/Write	W							
	Reset State	0	0	0	0	0	0		
	Function	Target ROM address (Lower 6 bits)							

BANK1 Address Compare Register 1

		7	6	5	4	3	2	1	0
ROMCMP11	Bit symbol	ROMC15	ROMC14	ROMC13	ROMC12	ROMC11	ROMC10	ROMC09	ROMC08
(1409H)	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Target ROM address (Middle 8 bits)							

BANK1 Address Compare Register 2

		7	6	5	4	3	2	1	0
ROMCMP12	Bit symbol	ROMC23	ROMC22	ROMC21	ROMC20	ROMC19	ROMC18	ROMC17	ROMC16
(140AH)	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Target ROM address (Upper 8 bits)							

Note 1: The ROMCMP10, ROMCMP11, and ROMCMP12 registers do not support read-modify-write operation.

Note 2: Bit0 and Bit1 of the ROMCMP10 is read as undefined.

Figure 3.15.3 Address Compare Registers (Bank1)

BANK2 Address Compare Register 0

		7	6	5	4	3	2	1	0
ROMCMP20 (1410H)	Bit symbol	ROMC07	ROMC06	ROMC05	ROMC04	ROMC03	ROMC02		
	Read/Write	W							
	Reset State	0	0	0	0	0	0		
	Function	Target ROM address (Lower 6 bits)							

BANK2 Address Compare Register 1

		7	6	5	4	3	2	1	0
ROMCMP21 (1411H)	Bit symbol	ROMC15	ROMC14	ROMC13	ROMC12	ROMC11	ROMC10	ROMC09	ROMC08
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Target ROM address (Middle 8 bits)							

BANK2 Address Compare Register 2

		7	6	5	4	3	2	1	0
ROMCMP22 (1412H)	Bit symbol	ROMC23	ROMC22	ROMC21	ROMC20	ROMC19	ROMC18	ROMC17	ROMC16
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Target ROM address (Upper 8 bits)							

Note 1: The ROMCMP20, ROMCMP21, and ROMCMP22 registers do not support read-modify-write operation.

Note 2: Bit0 and Bit1 of the ROMCMP20 is read as undefined.

Figure 3.15.4 Address Compare Registers (Bank2)

BANK3 Address Compare Register 0

		7	6	5	4	3	2	1	0
ROMCMP30 (1418H)	Bit symbol	ROMC07	ROMC06	ROMC05	ROMC04	ROMC03	ROMC02		
	Read/Write	W							
	Reset State	0	0	0	0	0	0		
	Function	Target ROM address (Lower 6 bits)							

BANK3 Address Compare Register 1

		7	6	5	4	3	2	1	0
ROMCMP31 (1419H)	Bit symbol	ROMC15	ROMC14	ROMC13	ROMC12	ROMC11	ROMC10	ROMC09	ROMC08
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Target ROM address (Middle 8 bits)							

BANK3 Address Compare Register 2

		7	6	5	4	3	2	1	0
ROMCMP32 (141AH)	Bit symbol	ROMC23	ROMC22	ROMC21	ROMC20	ROMC19	ROMC18	ROMC17	ROMC16
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Target ROM address (Upper 8 bits)							

Note 1: The ROMCMP30, ROMCMP31, and ROMCMP32 registers do not support read-modify-write operation.

Note 2: Bit0 and Bit1 of the ROMCMP30 is read as undefined.

Figure 3.15.5 Address Compare Registers (Bank3)

BANK4 Address Compare Register 0

		7	6	5	4	3	2	1	0
ROMCMP40	Bit symbol	ROMC07	ROMC06	ROMC05	ROMC04	ROMC03	ROMC02		
(1420H)	Read/Write	W							
	Reset State	0	0	0	0	0	0		
	Function	Target ROM address (Lower 6 bits)							

BANK4 Address Compare Register 1

		7	6	5	4	3	2	1	0
ROMCMP41	Bit symbol	ROMC15	ROMC14	ROMC13	ROMC12	ROMC11	ROMC10	ROMC09	ROMC08
(1421H)	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Target ROM address (Middle 8 bits)							

BANK4 Address Compare Register 2

		7	6	5	4	3	2	1	0
ROMCMP42	Bit symbol	ROMC23	ROMC22	ROMC21	ROMC20	ROMC19	ROMC18	ROMC17	ROMC16
(1422H)	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Target ROM address (Upper 8 bits)							

Note 1: The ROMCMP40, ROMCMP41, and ROMCMP42 registers do not support read-modify-write operation.

Note 2: Bit0 and Bit1 of the ROMCMP40 is read as undefined.

Figure 3.15.6 Address Compare Registers (Bank4)

BANK5 Address Compare Register 0

		7	6	5	4	3	2	1	0
ROMCMP50	Bit symbol	ROMC07	ROMC06	ROMC05	ROMC04	ROMC03	ROMC02		
(1428H)	Read/Write	W							
	Reset State	0	0	0	0	0	0		
	Function	Target ROM address (Lower 6 bits)							

BANK5 Address Compare Register 1

		7	6	5	4	3	2	1	0
ROMCMP51	Bit symbol	ROMC15	ROMC14	ROMC13	ROMC12	ROMC11	ROMC10	ROMC09	ROMC08
(1429H)	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Target ROM address (Middle 8 bits)							

BANK5 Address Compare Register 2

		7	6	5	4	3	2	1	0
ROMCMP52	Bit symbol	ROMC23	ROMC22	ROMC21	ROMC20	ROMC19	ROMC18	ROMC17	ROMC16
(142AH)	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Target ROM address (Upper 8 bits)							

Note 1: The ROMCMP50, ROMCMP51, and ROMCMP52 registers do not support read-modify-write operation.

Note 2: Bit0 and Bit1 of the ROMCMP50 is read as undefined.

Figure 3.15.7 Address Compare Registers (Bank5)

BANK6 Address Compare Register 0

		7	6	5	4	3	2	1	0
ROMCMP60	Bit symbol	ROMC07	ROMC06	ROMC05	ROMC04	ROMC03	ROMC02		
(1430H)	Read/Write	W							
	Reset State	0	0	0	0	0	0		
	Function	Target ROM address (Lower 6 bits)							

BANK6 Address Compare Register 1

		7	6	5	4	3	2	1	0
ROMCMP61	Bit symbol	ROMC15	ROMC14	ROMC13	ROMC12	ROMC11	ROMC10	ROMC09	ROMC08
(1431H)	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Target ROM address (Middle 8 bits)							

BANK6 Address Compare Register 2

		7	6	5	4	3	2	1	0
ROMCMP62	Bit symbol	ROMC23	ROMC22	ROMC21	ROMC20	ROMC19	ROMC18	ROMC17	ROMC16
(1432H)	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Target ROM address (Upper 8 bits)							

Note 1: The ROMCMP60, ROMCMP61, and ROMCMP62 registers do not support read-modify-write operation.

Note 2: Bit0 and Bit1 of the ROMCMP60 is read as undefined.

Figure 3.15.8 Address Compare Registers (Bank6)

BANK7 Address Compare Register 0

		7	6	5	4	3	2	1	0
ROMCMP70 (1438H)	Bit symbol	ROMC07	ROMC06	ROMC05	ROMC04	ROMC03	ROMC02		
	Read/Write	W							
	Reset State	0	0	0	0	0	0		
	Function	Target ROM address (Lower 6 bits)							

BANK7 Address Compare Register 1

		7	6	5	4	3	2	1	0
ROMCMP71 (1439H)	Bit symbol	ROMC15	ROMC14	ROMC13	ROMC12	ROMC11	ROMC10	ROMC09	ROMC08
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Target ROM address (Middle 8 bits)							

BANK7 Address Compare Register 2

		7	6	5	4	3	2	1	0
ROMCMP72 (143AH)	Bit symbol	ROMC23	ROMC22	ROMC21	ROMC20	ROMC19	ROMC18	ROMC17	ROMC16
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Target ROM address (Upper 8 bits)							

Note 1: The ROMCMP70, ROMCMP71, and ROMCMP72 registers do not support read-modify-write operation.

Note 2: Bit0 and Bit1 of the ROMCMP70 is read as undefined.

Figure 3.15.9 Address Compare Registers (Bank7)

BANK0 Address substitution Register LL

ROMSUB0LL (1404H)		7	6	5	4	3	2	1	0
	Bit symbol	ROMS07	ROMS06	ROMS05	ROMS04	ROMS03	ROMS02	ROMS01	ROMS00
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Lower 8　bits)							

BANK0 Address substitution Register LH

ROMSUB0LH (1405H)		7	6	5	4	3	2	1	0
	Bit symbol	ROMS15	ROMS14	ROMS13	ROMS12	ROMS11	ROMS10	ROMS09	ROMS08
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Upper 8 bits)							

BANK0 Address substitution Register HL

ROMSUB0HL (1406H)		7	6	5	4	3	2	1	0
	Bit symbol	ROMS23	ROMS22	ROMS21	ROMS20	ROMS19	ROMS18	ROMS17	ROMS16
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Lower 8 bits)							

BANK0 Address substitution Register HH

ROMSUB0HH (1407H)		7	6	5	4	3	2	1	0
	Bit symbol	ROMS31	ROMS30	ROMS29	ROMS28	ROMS27	ROMS26	ROMS25	ROMS24
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Upper 8 bits)							

Note: The ROMSUB0LL, ROMSUB0LH, ROMSUB0HL, and ROMSUB0HH registers do not support read-modify-write operation.

Figure 3.15.10 Address Substitution Registers (Bank 0)

BANK1 Address substitution Register LL

		7	6	5	4	3	2	1	0
ROMSUB1LL (140CH)	Bit symbol	ROMS07	ROMS06	ROMS05	ROMS04	ROMS03	ROMS02	ROMS01	ROMS00
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Lower 8 bits)							

BANK1 Address substitution Register LH

		7	6	5	4	3	2	1	0
ROMSUB1LH (140DH)	Bit symbol	ROMS15	ROMS14	ROMS13	ROMS12	ROMS11	ROMS10	ROMS09	ROMS08
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Upper 8 bits)							

BANK1 Address substitution Register HL

		7	6	5	4	3	2	1	0
ROMSUB1HL (140EH)	Bit symbol	ROMS23	ROMS22	ROMS21	ROMS20	ROMS19	ROMS18	ROMS17	ROMS16
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Lower 8 bits)							

BANK1 Address substitution Register HH

		7	6	5	4	3	2	1	0
ROMSUB1HH (140FH)	Bit symbol	ROMS31	ROMS30	ROMS29	ROMS28	ROMS27	ROMS26	ROMS25	ROMS24
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Upper 8 bits)							

Note: The ROMSUB1LL, ROMSUB1LH, ROMSUB1HL, and ROMSUB1HH registers do not support read-modify-write operation.

Figure 3.15.11 Address Substitution Registers (Bank 1)

BANK2 Address substitution Register LL

ROMSUB2LL
(1414H)

	7	6	5	4	3	2	1	0
Bit symbol	ROMS07	ROMS06	ROMS05	ROMS04	ROMS03	ROMS02	ROMS01	ROMS00
Read/Write	W							
Reset State	0	0	0	0	0	0	0	0
Function	Patch code (Lower 8 bits)							

BANK2 Address substitution Register LH

ROMSUB2LH
(1415H)

	7	6	5	4	3	2	1	0
Bit symbol	ROMS15	ROMS14	ROMS13	ROMS12	ROMS11	ROMS10	ROMS09	ROMS08
Read/Write	W							
Reset State	0	0	0	0	0	0	0	0
Function	Patch code (Upper 8 bits)							

BANK2 Address substitution Register HL

ROMSUB2HL
(1416H)

	7	6	5	4	3	2	1	0
Bit symbol	ROMS23	ROMS22	ROMS21	ROMS20	ROMS19	ROMS18	ROMS17	ROMS16
Read/Write	W							
Reset State	0	0	0	0	0	0	0	0
Function	Patch code (Lower 8 bits)							

BANK2 Address substitution Register HH

ROMSUB2HH
(1417H)

	7	6	5	4	3	2	1	0
Bit symbol	ROMS31	ROMS30	ROMS29	ROMS28	ROMS27	ROMS26	ROMS25	ROMS24
Read/Write	W							
Reset State	0	0	0	0	0	0	0	0
Function	Patch code (Upper 8 bits)							

Note: The ROMSUB2LL, ROMSUB2LH, ROMSUB2HL, and ROMSUB2HH registers do not support read-modify-write operation.

Figure 3.15.12 Address Substitution Registers (Banks 2)

BANK3 Address substitution Register LL

ROMSUB3LL (141CH)		7	6	5	4	3	2	1	0
	Bit symbol	ROMS07	ROMS06	ROMS05	ROMS04	ROMS03	ROMS02	ROMS01	ROMS00
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Lower 8 bits)							

BANK3 Address substitution Register LH

ROMSUB3LH (141DH)		7	6	5	4	3	2	1	0
	Bit symbol	ROMS15	ROMS14	ROMS13	ROMS12	ROMS11	ROMS10	ROMS09	ROMS08
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Upper 8 bits)							

BANK3 Address substitution Register HL

ROMSUB3HL (141EH)		7	6	5	4	3	2	1	0
	Bit symbol	ROMS23	ROMS22	ROMS21	ROMS20	ROMS19	ROMS18	ROMS17	ROMS16
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Lower 8 bits)							

BANK3 Address substitution Register HH

ROMSUB3HH (141FH)		7	6	5	4	3	2	1	0
	Bit symbol	ROMS31	ROMS30	ROMS29	ROMS28	ROMS27	ROMS26	ROMS25	ROMS24
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Upper 8 bits)							

Note: The ROMSUB3LL, ROMSUB3LH, ROMSUB3HL, and ROMSUB3HH registers do not support read-modify-write operation.

Figure 3.15.13 Address Substitution Registers (Banks 3)

BANK4 Address substitution Register LL

		7	6	5	4	3	2	1	0
ROMSUB4LL (1424H)	Bit symbol	ROMS07	ROMS06	ROMS05	ROMS04	ROMS03	ROMS02	ROMS01	ROMS00
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Lower 8 bits)							

BANK4 Address substitution Register LH

		7	6	5	4	3	2	1	0
ROMSUB4LH (1425H)	Bit symbol	ROMS15	ROMS14	ROMS13	ROMS12	ROMS11	ROMS10	ROMS09	ROMS08
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Upper 8 bits)							

BANK4 Address substitution Register HL

		7	6	5	4	3	2	1	0
ROMSUB4HL (1426H)	Bit symbol	ROMS23	ROMS22	ROMS21	ROMS20	ROMS19	ROMS18	ROMS17	ROMS16
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Lower 8 bits)							

BANK4 Address substitution Register HH

		7	6	5	4	3	2	1	0
ROMSUB4HH (1427H)	Bit symbol	ROMS31	ROMS30	ROMS29	ROMS28	ROMS27	ROMS26	ROMS25	ROMS24
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Upper 8 bits)							

Note: The ROMSUB4LL, ROMSUB4LH, ROMSUB4HL, and ROMSUB4HH registers do not support read-modify-write operation.

Figure 3.15.14 Address Substitution Registers (Banks 4)

BANK5 Address substitution Register LL

ROMSUB5LL (142CH)		7	6	5	4	3	2	1	0
	Bit symbol	ROMS07	ROMS06	ROMS05	ROMS04	ROMS03	ROMS02	ROMS01	ROMS00
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Lower 8 bits)							

BANK5 Address substitution Register LH

ROMSUB5LH (142DH)		7	6	5	4	3	2	1	0
	Bit symbol	ROMS15	ROMS14	ROMS13	ROMS12	ROMS11	ROMS10	ROMS09	ROMS08
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Upper 8 bits)							

BANK5 Address substitution Register HL

ROMSUB5HL (142EH)		7	6	5	4	3	2	1	0
	Bit symbol	ROMS23	ROMS22	ROMS21	ROMS20	ROMS19	ROMS18	ROMS17	ROMS16
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Lower 8 bits)							

BANK5 Address substitution Register HH

ROMSUB5HH (142FH)		7	6	5	4	3	2	1	0
	Bit symbol	ROMS31	ROMS30	ROMS29	ROMS28	ROMS27	ROMS26	ROMS25	ROMS24
	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Upper 8 bits)							

Note: The ROMSUB5LL, ROMSUB5LH, ROMSUB5HL, and ROMSUB5HH registers do not support read-modify-write operation.

Figure 3.15.15 Address Substitution Registers (Banks 5)

BANK6 Address substitution Register LL

		7	6	5	4	3	2	1	0
ROMSUB6LL	Bit symbol	ROMS07	ROMS06	ROMS05	ROMS04	ROMS03	ROMS02	ROMS01	ROMS00
(1434H)	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Lower 8 bits)							

BANK6 Address substitution Register LH

		7	6	5	4	3	2	1	0
ROMSUB6LH	Bit symbol	ROMS15	ROMS14	ROMS13	ROMS12	ROMS11	ROMS10	ROMS09	ROMS08
(1435H)	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Upper 8 bits)							

BANK6 Address substitution Register HL

		7	6	5	4	3	2	1	0
ROMSUB6HL	Bit symbol	ROMS23	ROMS22	ROMS21	ROMS20	ROMS19	ROMS18	ROMS17	ROMS16
(1436H)	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Lower 8 bits)							

BANK6 Address substitution Register HH

		7	6	5	4	3	2	1	0
ROMSUB6HH	Bit symbol	ROMS31	ROMS30	ROMS29	ROMS28	ROMS27	ROMS26	ROMS25	ROMS24
(1437H)	Read/Write	W							
	Reset State	0	0	0	0	0	0	0	0
	Function	Patch code (Upper 8 bits)							

Note: The ROMSUB6LL, ROMSUB6LH, ROMSUB6HL, and ROMSUB6HH registers do not support read-modify-write operation.

Figure 3.15.16 Address Substitution Registers (Banks 6)

BANK7 Address substitution Register LL

ROMSUB7LL
(143CH)

	7	6	5	4	3	2	1	0
Bit symbol	ROMS07	ROMS06	ROMS05	ROMS04	ROMS03	ROMS02	ROMS01	ROMS00
Read/Write	W							
Reset State	0	0	0	0	0	0	0	0
Function	Patch code (Lower 8 bits)							

BANK7 Address substitution Register LH

ROMSUB7LH
(143DH)

	7	6	5	4	3	2	1	0
Bit symbol	ROMS15	ROMS14	ROMS13	ROMS12	ROMS11	ROMS10	ROMS09	ROMS08
Read/Write	W							
Reset State	0	0	0	0	0	0	0	0
Function	Patch code (Upper 8 bits)							

BANK7 Address substitution Register HL

ROMSUB7HL
(143EH)

	7	6	5	4	3	2	1	0
Bit symbol	ROMS23	ROMS22	ROMS21	ROMS20	ROMS19	ROMS18	ROMS17	ROMS16
Read/Write	W							
Reset State	0	0	0	0	0	0	0	0
Function	Patch code (Lower 8 bits)							

BANK7 Address substitution Register HH

ROMSUB7HH
(143FH)

	7	6	5	4	3	2	1	0
Bit symbol	ROMS31	ROMS30	ROMS29	ROMS28	ROMS27	ROMS26	ROMS25	ROMS24
Read/Write	W							
Reset State	0	0	0	0	0	0	0	0
Function	Patch code (Upper 8 bits)							

Note: The ROMSUB7LL, ROMSUB7LH, ROMSUB7HL, and ROMSUB7HH registers do not support read-modify-write operation.

Figure 3.15.17 Address Substitution Registers (Banks 7)

3.15.3 Operation

(1) Replacing data

Correction procedure:

Load the address compare registers ROMCMPx0 to ROMCMPx2 (banks No. x = 0 to 7) with the target address where ROM data need be replaced. Store 4-byte patch code in the ROMSUBxLL, ROMSUBxLH, ROMSUBxHL and ROMSUBxHH (banks No. x = 0 to 7) registers.

After each register store , when the CPU address matches the value stored in the ROMCMPx0 to ROMCMPx2 (banks No. x = 0 to 7) registers, the program patch logic disables RD output to the internal ROM and drives out the code stored in the ROMSUBxLL to ROMSUBxHH (banks No. x = 0 to 7) to the internal bus. The CPU thus fetches the patch code.

The following shows some examples:

Examples:

a. Replacing 00H at address FF1230H with AAH

		7	6	5	4	3	2	1	0	
ROMCMP00	←	0	0	1	1	0	0	0	0	Stores 30H in address compare register 0 for bank0.
ROMCMP01	←	0	0	0	1	0	0	1	0	Stores 12H in address compare register 1 for bank0.
ROMCMP02	←	1	1	1	1	1	1	1	1	Stores FFH in address compare register 2 for bank0.
ROMSUB0LL	←	1	0	1	0	1	0	1	0	Store AAH in address substitution register LL for bank0.
ROMSUB0LH	←	0	0	0	1	0	0	0	1	Store 11H in address substitution register LH for bank0.
ROMSUB0HL	←	0	0	1	0	0	0	1	0	Store 22H in address substitution register HL for bank0.
ROMSUB0HH	←	0	0	1	1	0	0	1	1	Store 33H in address substitution register HH for bank0.

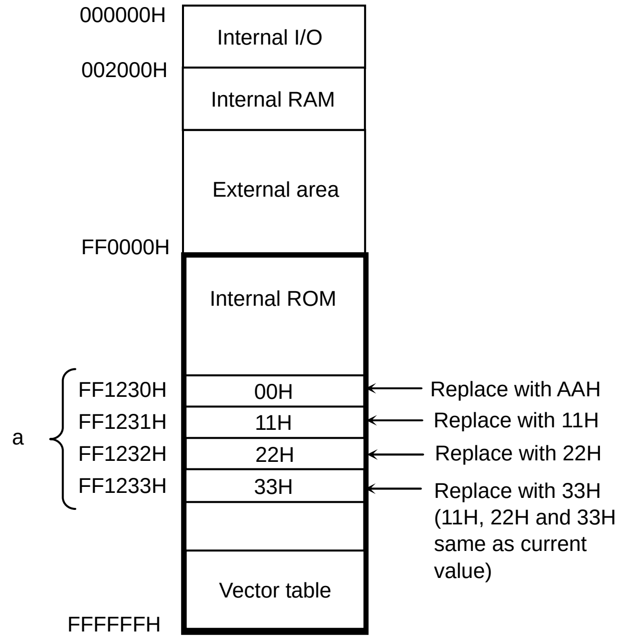


Figure 3.15.18 Example Patch Code Implementation

b. Replacing 33H at address FF1233H with BBH

		7	6	5	4	3	2	1	0	
ROMCMP00	←	0	0	1	1	0	0	0	0	Stores 30H in address compare register 0 for bank0.
ROMCMP01	←	0	0	0	1	0	0	1	0	Stores 12H in address compare register 1 for bank0.
ROMCMP02	←	1	1	1	1	1	1	1	1	Stores FFH in address compare register 2 for bank0.
ROMSUB0LL	←	0	0	0	0	0	0	0	0	Store 00H in address substitution register LL for bank0
ROMSUB0LH	←	0	0	0	1	0	0	0	1	Store 11H in address substitution register LH for bank0
ROMSUB0HL	←	0	0	1	0	0	0	1	0	Store 22H in address substitution register HL for bank0.
ROMSUB0HH	←	1	0	1	1	1	0	1	1	Store BBH in address substitution register HH for bank0.

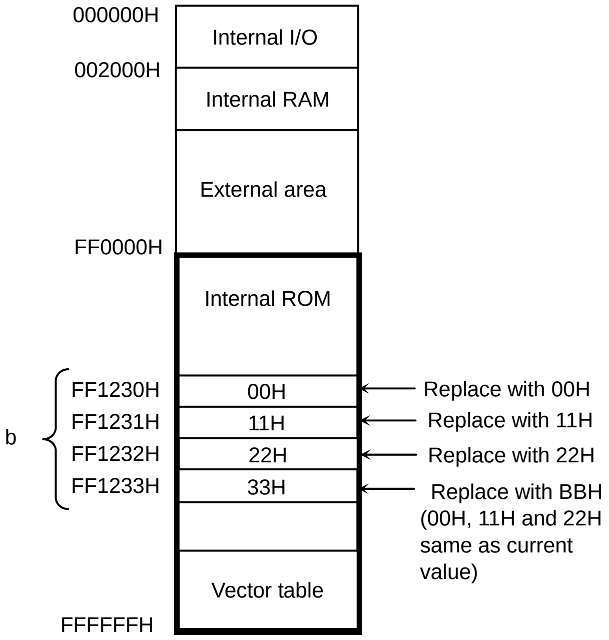


Figure 3.15.19 Example Patch Code Implementation

c. Replacing 00H at address FF1230H with AAH, 11H at address FF1231H with BBH, 22H at address FF1232H with CCH and 33H at address FF1233H with DDH

		7	6	5	4	3	2	1	0	
ROMCMP00	←	0	0	1	1	0	0	0	0	Stores 30H in address compare register 0 for bank0.
ROMCMP01	←	0	0	0	1	0	0	1	0	Stores 12H in address compare register 1 for bank0.
ROMCMP02	←	1	1	1	1	1	1	1	1	Stores FFH in address compare register 2 for bank0.
ROMSUB0LL	←	1	0	1	0	1	0	1	0	Store AAH in address substitution register LL for bank0
ROMSUB0LH	←	1	0	1	1	1	0	1	1	Store BBH in address substitution register LH for bank0.
ROMSUB0HL	←	1	1	0	0	1	1	0	0	Store CCH in address substitution register HL for bank0.
ROMSUB0HH	←	1	1	0	1	1	1	0	1	Store DDH in address substitution register HH for bank0.

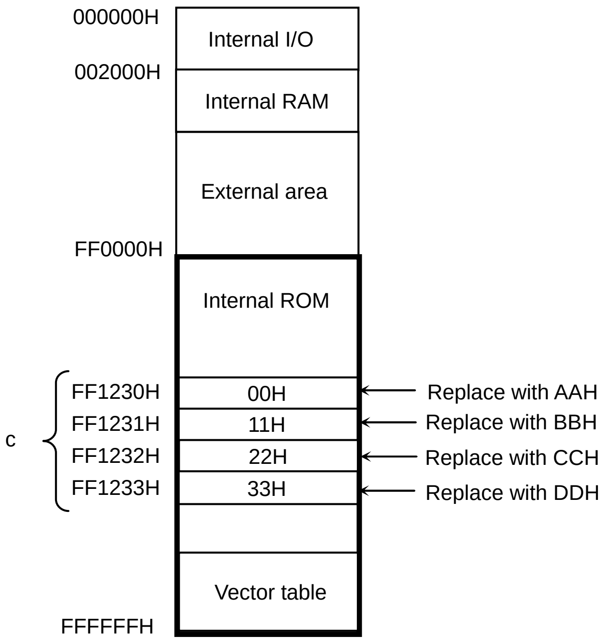


Figure 3.15.20 Example Patch Code Implementation

d. Replacing 11H at address FF1231H with AAH, 22H at address FF1232H with BBH, 33H at address FF1233H with CCH and 44H at address FF1234H with DDH (Requiring two banks)

		7	6	5	4	3	2	1	0	
ROMCMP00	←	0	0	1	1	0	0	0	0	Stores 30H in address compare register 0 for bank0.
ROMCMP01	←	0	0	0	1	0	0	1	0	Stores 12H in address compare register 1 for bank0.
ROMCMP02	←	1	1	1	1	1	1	1	1	Stores FFH in address compare register 2 for bank0.
ROMSUB0LL	←	0	0	0	0	0	0	0	0	Store 00H in address substitution register LL for bank0
ROMSUB0LH	←	1	0	1	0	1	0	1	0	Store AAH in address substitution register LH for bank0.
ROMSUB0HL	←	1	0	1	1	1	0	1	1	Store BBH in address substitution register HL for bank0
ROMSUB0HH	←	1	1	0	0	1	1	0	0	Store CCH in address substitution register HH for bank0
ROMCMP10	←	0	0	1	1	0	1	0	0	Stores 34H in address compare register 0 for bank1.
ROMCMP11	←	0	0	0	1	0	0	1	0	Stores 12H in address compare register 1 for bank1.
ROMCMP12	←	1	1	1	1	1	1	1	1	Stores FFH in address compare register 2 for bank1.
ROMSUB1LL	←	1	1	0	1	1	1	0	1	Store DDH in address substitution register LL for bank1
ROMSUB1LH	←	0	1	0	1	0	1	0	1	Store 55H in address substitution register LH for bank1
ROMSUB1HL	←	0	1	1	0	0	1	1	0	Store 66H in address substitution register HL for bank1.
ROMSUB1HH	←	0	1	1	1	0	1	1	1	Store 77H in address substitution register HH for bank1.

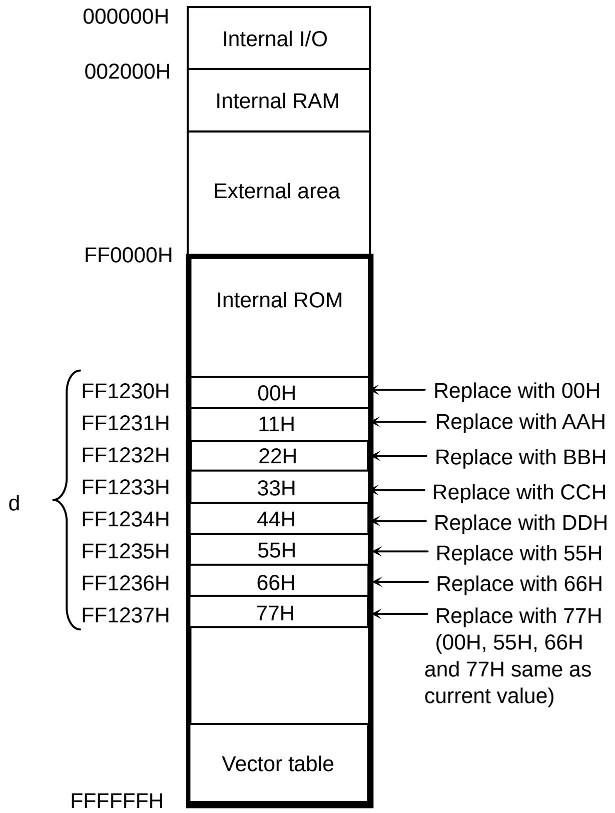


Figure 3.15.21 Example Patch Code Implementation

(2) Using an interrupt to cause a branch

A wider range of program code can also be fixed using a software interrupt (SWI). With a patch code loaded into on-chip RAM, the program patch logic can be used to replace program code at a specified address with a single-byte SWI instruction, which causes a branch to the patch program.

Note that this method can only be used if the original ROM data has been developed with on-chip RAM addresses specified as SWI vector addresses.

Correction procedure:

Load the address compare registers ROMCMPx0 to ROMCMPx2 (x = bank No. 0 to 7) with the start address of the program code that is to be fixed. If it is an even address, store an SWI instruction code (e.g., SWI: F9H) in ROMSUBxLL or ROMSUBxHL. If the start address is an odd address, store an SWI instruction code in ROMSUBxLH or ROMSUBxHH. When the data for the purpose of substitution is required only for 1 to 3 bytes, please set the same data as original ROM data to the remaining data.

When the CPU address matches the value stored in the ROMCMPx0 to ROMCMPx2 registers, the program patch logic disables RD output to the internal ROM and drives out the SWI instruction code to the internal bus. Upon fetching the SWI code, the CPU makes a branch to the internal RAM area to execute the preloaded code.

At the end of the patch program executed from the internal RAM, the CPU directly rewrites the saved PC value so that it points to the address following the patch code, and then executes a RETI.

The following shows an example:

Example: Fixing a program within the range from FF5000H to FF507FH

Before developing the original ROM data, set the SWI1 vector reference address to 002500H (on-chip RAM area).

Use the startup routine to load the patch code to on-chip RAM (002500H to 0025EFH). Store the start address (FF5000H) of the ROM area to be fixed in the ROMCMP00 to ROMCMP02. Store the SWI1 instruction code (F9H) in the ROMSUB0LL and the current data at FF5001H (AAH) in the ROMSUB0LH and the current data at FF5002H (BBH) in the ROMSUB0HL and the current data at FF5003 (CCH) in the ROMSUB0HH. When the CPU address matches the value stored in ROMCMP00 to ROMCMP02, the program patch logic replaces the ROM-based code at FF5000H with F9H. The CPU then executes the SWI1 instruction, which causes a branch to 002500H in the on-chip RAM area. After executing the patch program the CPU finally rewrites the saved PC value to FF5080H and executes a RETI.

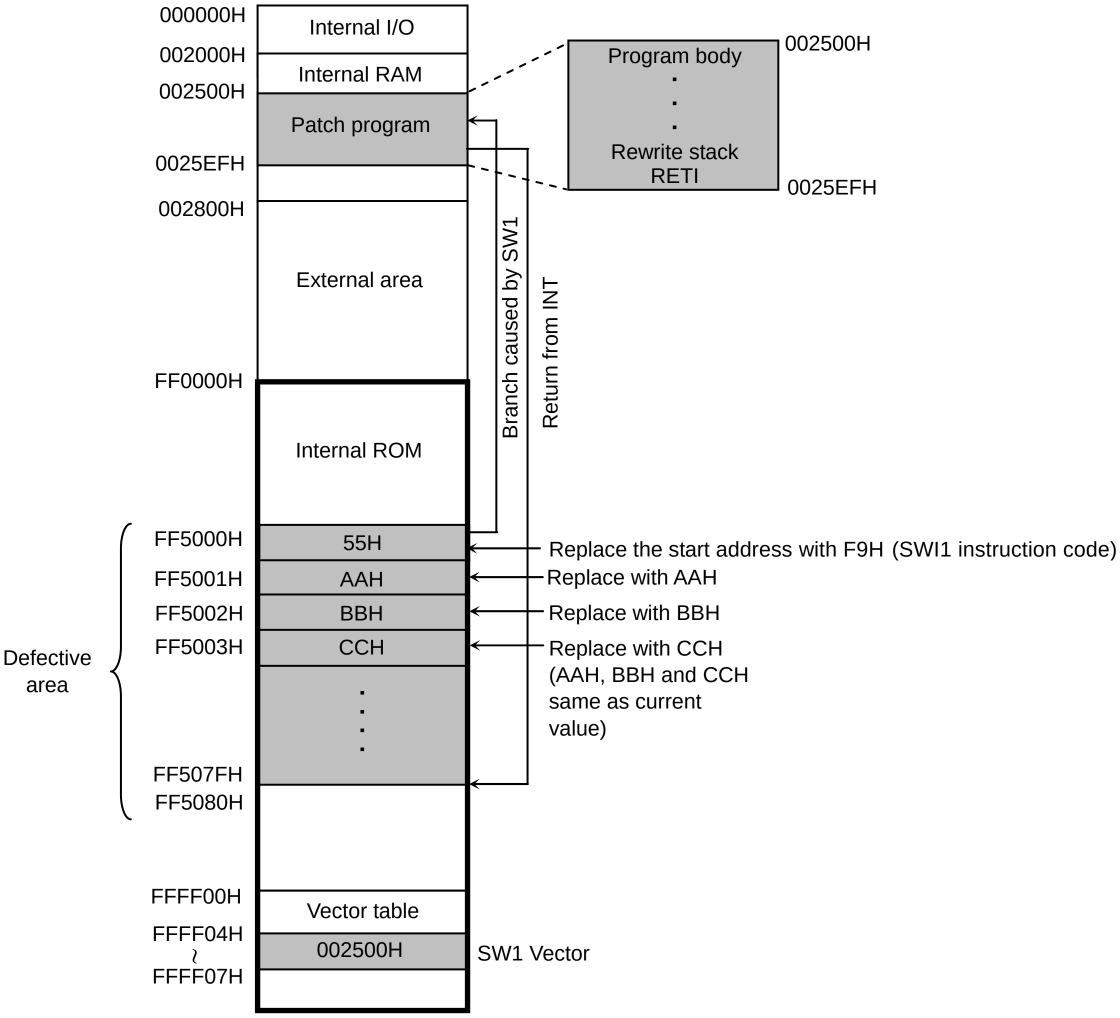


Figure 3.15.22 Example Patch Code Implementation

3.16 Flash Memory

The TMP92FD23A incorporates flash memory that can be electrically erased and programmed using a single 3V power supply.

The flash memory is programmed and erased using JEDEC-standard commands. After a program or erase command is input, the corresponding operation is automatically performed internally. Erase operations can be performed by the entire chip (chip erase) or on a sector basis (sector erase).

The configuration and operations of the flash memory are described below.

3.16.1 Features

- Power supply voltage for program/erase operations
  Vcc = 3.0 V to 3.6 V (-10 °C to 40 °C)
- Configuration
  128 K × 32 bits (512 Kbytes)
- Functions
  Single-long word programming
  Chip erase
  Sector erase
  Data polling/Toggle bit
- Sector size
  4 Kbytes × 128
- Mode control
  JEDEC-standard commands
- Programming method
  On-board programming
  Parallel programmer
- Security
  Write protection
  Read protection

3.16.2 Block Diagram

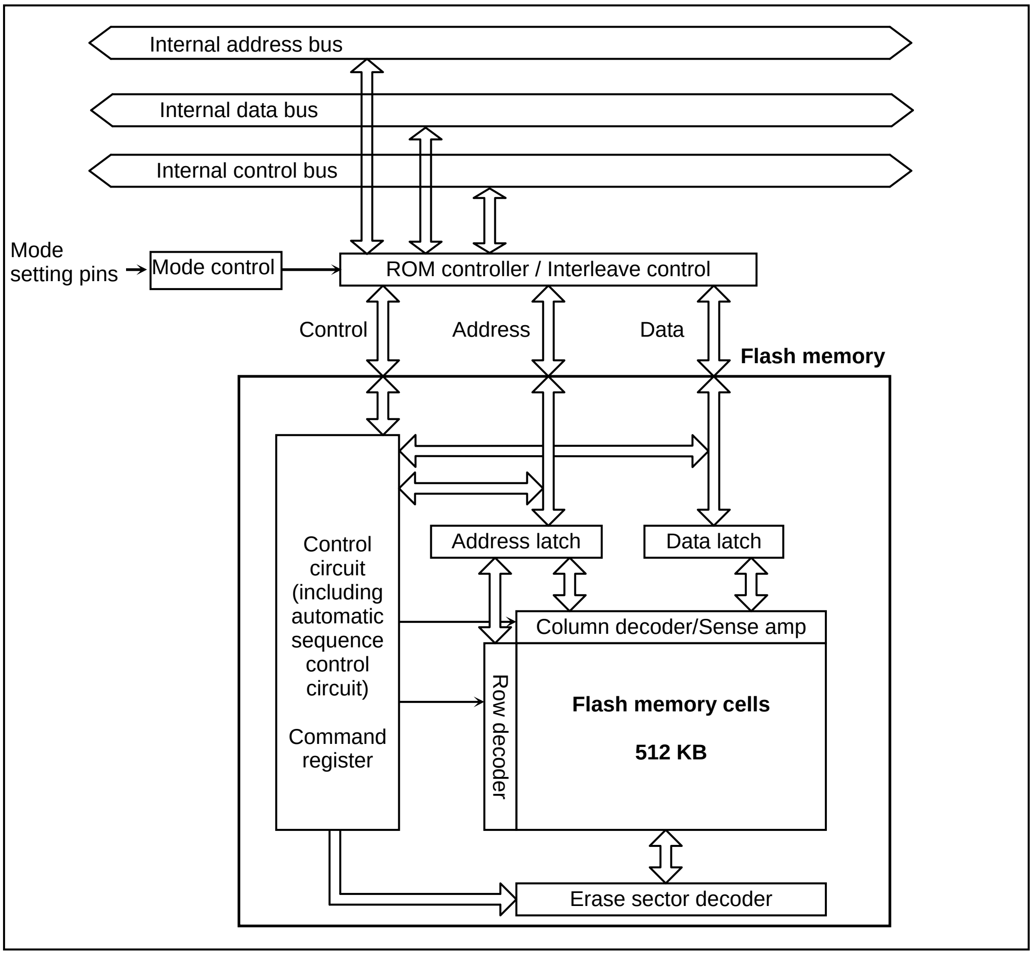


Figure 3.16.1 Block Diagram of Flash Memory Unit

3.16.3 Operation Modes

3.16.3.1 Overview

The following three types of operation modes are available to control program/erase operations on the flash memory.

Table 3.16.1 Description of Operation Modes

Operation Mode Name	Description
Single Chip mode	After reset release, the device starts up from the internal flash memory. Single Chip mode is further divided into two modes: “Normal mode” is a mode in which user application programs are executed, and “User Boot mode” is used to program the flash memory on-board. The means of switching between these two modes can be set by the user as desired. For example, it can be set so that Port 00 = “1” selects Normal mode and Port 00 = “0” selects User Boot mode. The user must include a routine to handle mode switching in a user application program.
Normal mode	In this mode, the device starts up from a user application program.
User Boot mode	In this mode, the flash memory can be programmed by a user-specified method.
Single Boot mode	After reset release, the device starts up from the internal boot ROM (mask ROM). The boot ROM includes an algorithm which allows a program for programming/erasing the flash memory on-board via a serial port to be transferred to the device’s internal RAM. The transferred program is then executed in the internal RAM so that the flash memory can be programmed/erased by receiving data from an external host and issuing program/erase commands.
Programmer mode	This mode enables the internal flash memory to be programmed/erased using a general-purpose programmer. For programmers that can be used, please contact your local Toshiba sales representative.

Of the modes listed in Table 3.16.1, the internal flash memory can be programmed in User Boot mode, Single Boot mode and Programmer mode.

The mode in which the flash memory can be programmed/erased while mounted on the user board is defined as the on-board programming mode. Of the modes listed above, Single Boot mode and User Boot mode are classified as on-board programming modes. Single Boot mode supports Toshiba’s proprietary programming/erase method using serial I/O. User Boot mode (within Single Chip mode) allows the flash memory to be programmed/erased by a user-specified method.

Programmer mode is provided with a read protect function which prohibits reading of ROM data. By enabling the read protect function upon completion of programming, the user can protect ROM data from being read by third parties.

The operation mode — Single Chip mode, Single Boot mode or Programmer mode — is determined during reset by externally setting the input levels on the AM0, AM1 and $\overline{\text{BOOT}}$ (P80) pins.

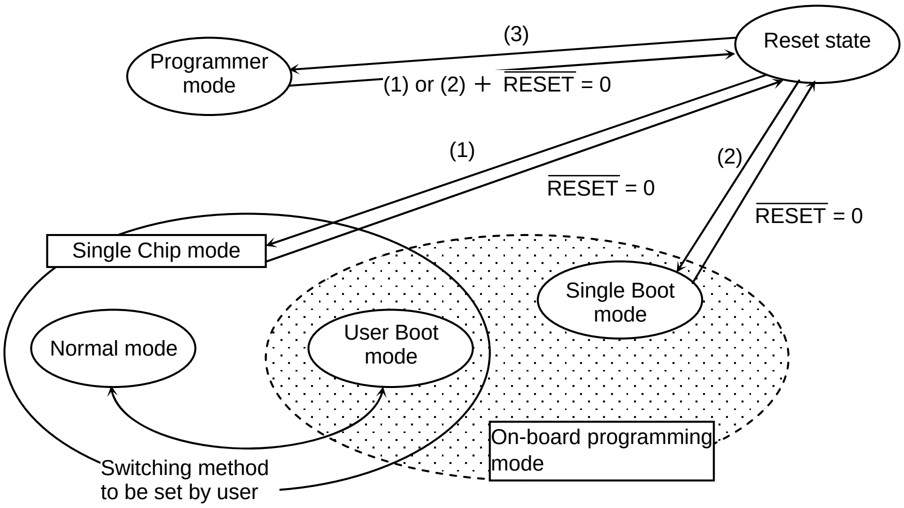
Except in Programmer mode which is entered with $\overline{\text{RESET}}$ held at "0", the CPU will start operating in the selected mode after the reset state is released. Once the operation mode has been set, make sure that the input levels on the mode setting pins are not changed during operation. Table 3.16.2 shows how to set each operation mode, and Figure 3.16.2 shows a mode transition diagram.

Table 3.16.2 Operation Mode Pin Settings

	Operation Mode	Input Pins			
		$\overline{\text{RESET}}$	$\overline{\text{BOOT}}$ (P80)	AM1	AM0
(1)	Single Chip mode (Normal or User Boot mode)	↑	1	1	1
(2)	Single Boot mode		0	1	1
(3)	Programmer mode	0	−	1	0

Although P80 is an output port, it becomes an input port with pull-up resistor only during a reset. After a reset, P80 operates as follows depending on the operation mode.

- Single chip mode: Output port (Without pull-up resistor)
- Single boot mode: Pull-up (Input gate is invalid, and output gate is in high impedance.)


(3)
Reset state
Programmer mode
(1) or (2) + $\overline{\text{RESET}}$ = 0
(1)
$\overline{\text{RESET}}$ = 0
(2)
$\overline{\text{RESET}}$ = 0
Single Chip mode
Single Boot mode
Normal mode
User Boot mode
On-board programming mode
Switching method to be set by user

Numbers in ( ) correspond to the operation mode pin settings shown in Table 3.16.2.

Figure 3.16.2 Mode Transition Diagram

3.16.3.2 Reset Operation

To reset the device, hold the $\overline{\text{RESET}}$ input at "0" for at least 20 system clocks while the power supply voltage is within the rated operating voltage range and the internal high-frequency oscillator is oscillating stably. For details, refer to 3.1.2 "Reset Operation."

3.16.3.3 Memory Map for Each Operation Mode

In this product, the memory map varies with operation mode. The memory map and sector address ranges for each operation mode are shown below.

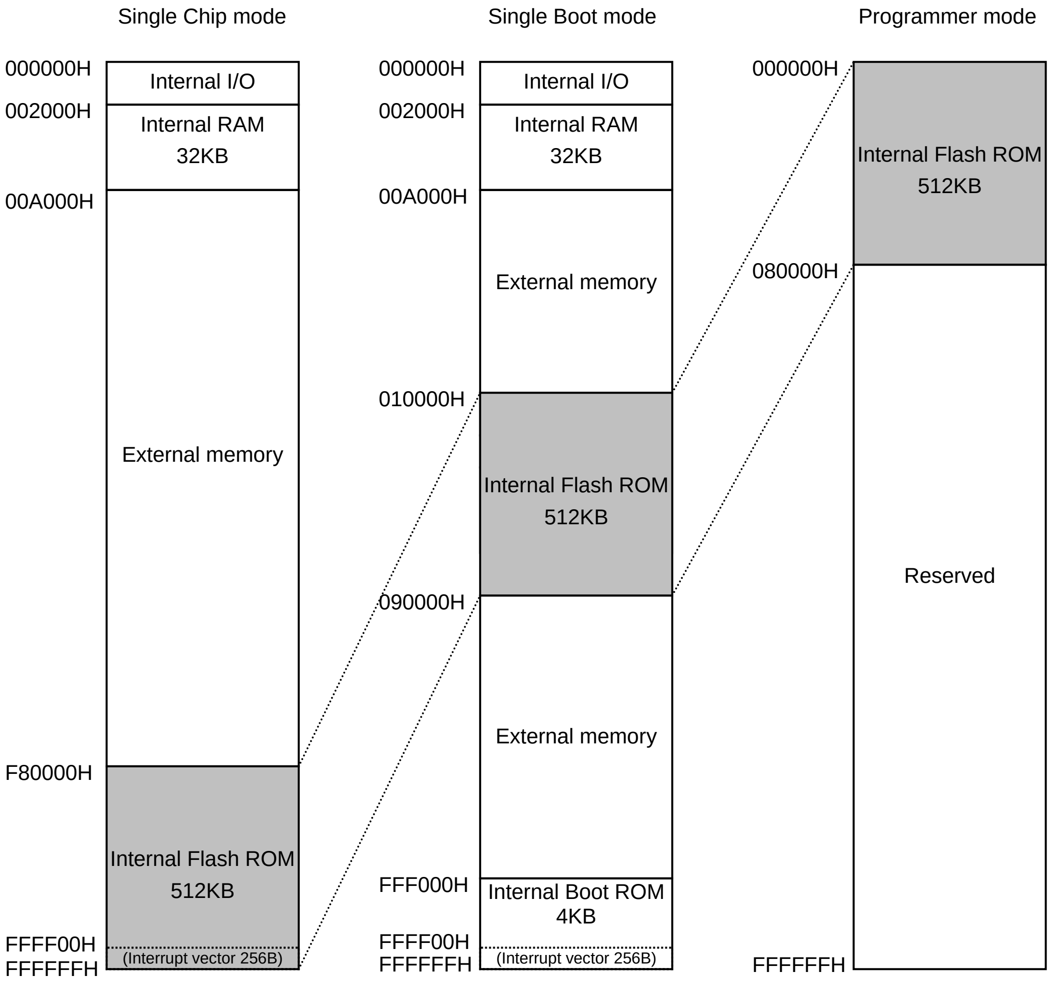


Figure 3.16.3 TMP92FD23A Memory Map for Each Operation Mode

Table 3.16.3 Sector Address Ranges for Each Operation Mode

	Single Chip Mode	Single Boot Mode
Sector-0	F80000H to F80FFFH	10000H to 10FFFH
Sector-1	F81000H to F81FFFH	11000H to 11FFFH
Sector-2	F82000H to F82FFFH	12000H to 12FFFH
Sector-3	F83000H to F83FFFH	13000H to 13FFFH
Sector-4	F84000H to F84FFFH	14000H to 14FFFH
Sector-5	F85000H to F85FFFH	15000H to 15FFFH
Sector-6	F86000H to F86FFFH	16000H to 16FFFH
Sector-7	F87000H to F87FFFH	17000H to 17FFFH
Sector-8	F88000H to F88FFFH	18000H to 18FFFH
Sector-9	F89000H to F89FFFH	19000H to 19FFFH
Sector-10	F8A000H to F8AFFFH	1A000H to 1AFFFH
Sector-11	F8B000H to F8BFFFH	1B000H to 1BFFFH
Sector-12	F8C000H to F8CFFFH	1C000H to 1CFFFH
Sector-13	F8D000H to F8DFFFH	1D000H to 1DFFFH
Sector-14	F8E000H to F8EFFFH	1E000H to 1EFFFH
Sector-15	F8F000H to F8FFFFH	1F000H to 1FFFFH
Sector-16	F90000H to F90FFFH	20000H to 20FFFH
.	.	.
.	.	.
.	.	.
.	.	.
Sector-111	FEF000H to FEFFFFH	7F000H to 7FFFFH
Sector-112	FF0000H to FF0FFFH	80000H to 80FFFH
Sector-113	FF1000H to FF1FFFH	81000H to 81FFFH
Sector-114	FF2000H to FF2FFFH	82000H to 82FFFH
Sector-115	FF3000H to FF3FFFH	83000H to 83FFFH
Sector-116	FF4000H to FF4FFFH	84000H to 84FFFH
Sector-117	FF5000H to FF5FFFH	85000H to 85FFFH
Sector-118	FF6000H to FF6FFFH	86000H to 86FFFH
Sector-119	FF7000H to FF7FFFH	87000H to 87FFFH
Sector-120	FF8000H to FF8FFFH	88000H to 88FFFH
Sector-121	FF9000H to FF9FFFH	89000H to 89FFFH
Sector-122	FFA000H to FFAFFFH	8A000H to 8AFFFH
Sector-123	FFB000H to FFBFFFH	8B000H to 8BFFFH
Sector-124	FFC000H to FFCFFFH	8C000H to 8CFFFH
Sector-125	FFD000H to FFDFFFH	8D000H to 8DFFFH
Sector-126	FFE000H to FFEFFFH	8E000H to 8EFFFH
Sector-127	FFF000H to FFFFFFH	8F000H to 8FFFFH

3.16.4 Single Boot Mode

In Single Boot mode, the internal boot ROM (mask ROM) is activated to transfer a program/erase routine (user-created boot program) from an external source into the internal RAM. This program/erase routine is then used to program/erase the flash memory. In this mode, the internal boot ROM is mapped into an area containing the interrupt vector table, in which the boot ROM program is executed. The flash memory is mapped into an address space different from the one into which the boot ROM is mapped (see Figure 3.16.3).

The device's SIO (SIO1) and the controller are connected to transfer the program/erase routine from the controller to the device's internal RAM. This program/erase routine is then executed to program/erase the flash memory.

The program/erase routine is executed by sending commands and write data from the controller. The communications protocol between the device and the controller is described later in this manual. Before the program/erase routine can be transferred to the RAM, user password verification is performed to ensure the security of user ROM data. If the password is not verified correctly, the RAM transfer operation cannot be performed. In Single Boot mode, disable interrupts and use the interrupt request flags to check for an interrupt request.

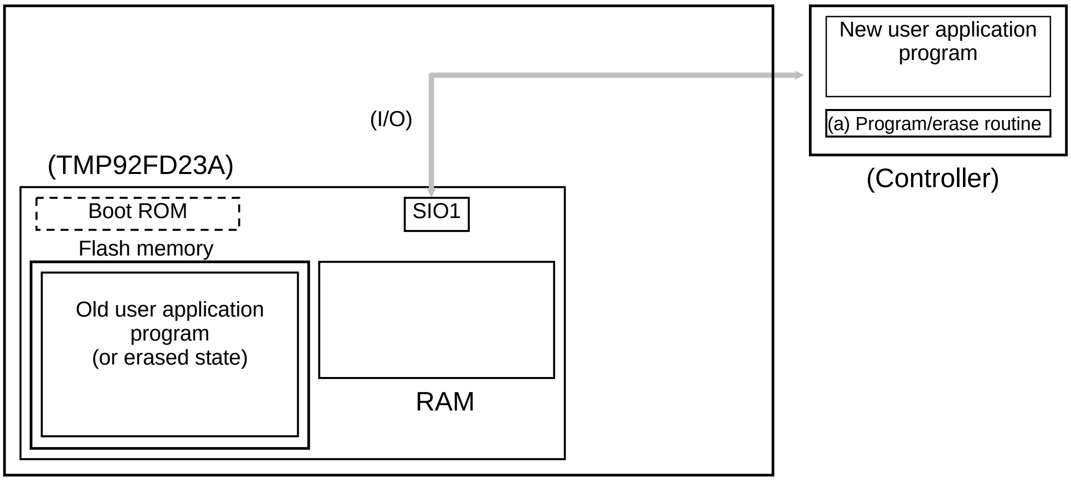
Note1: In Single Boot mode, the boot-ROM programs are executed in Normal mode. Do not change to another operation mode in the program/erase routine.

Note2: In the initial routine of the boot-ROM program, after changing the clock gear from fc/16 to fc, PLL is active. Therefore, fc is set to four times as fast as f_{OSCH}.

3.16.4.1 Using the program/erase algorithm in the internal boot ROM

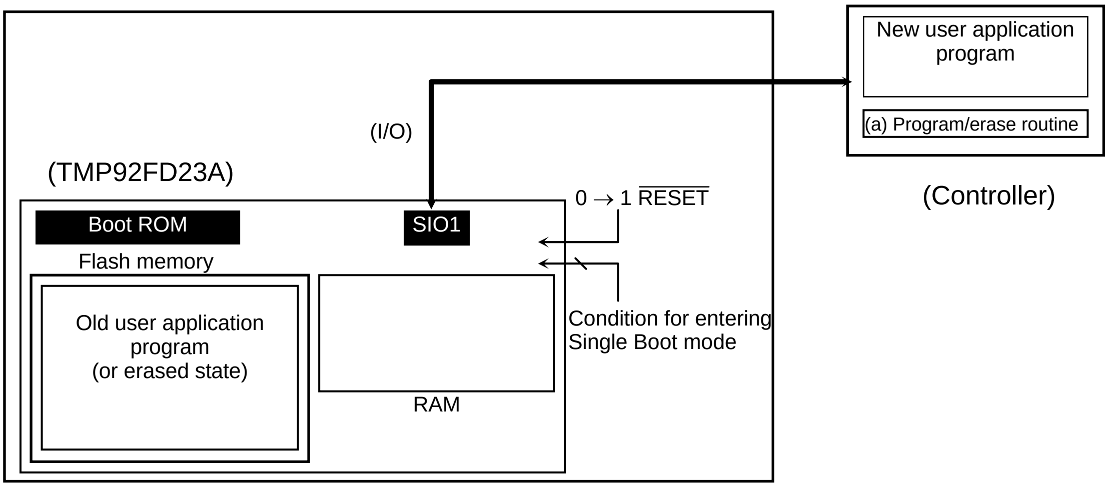
(Step-1) Environment setup

Since the program/erase routine and write data are transferred via SIO (SIO1), connect the device's SIO (SIO1) and the controller on the board. The user must prepare the program/erase routine (a) on the controller.


New user application program
(a) Program/erase routine
(Controller)
(I/O)
(TMP92FD23A)
Boot ROM
SIO1
Flash memory
Old user application program (or erased state)
RAM

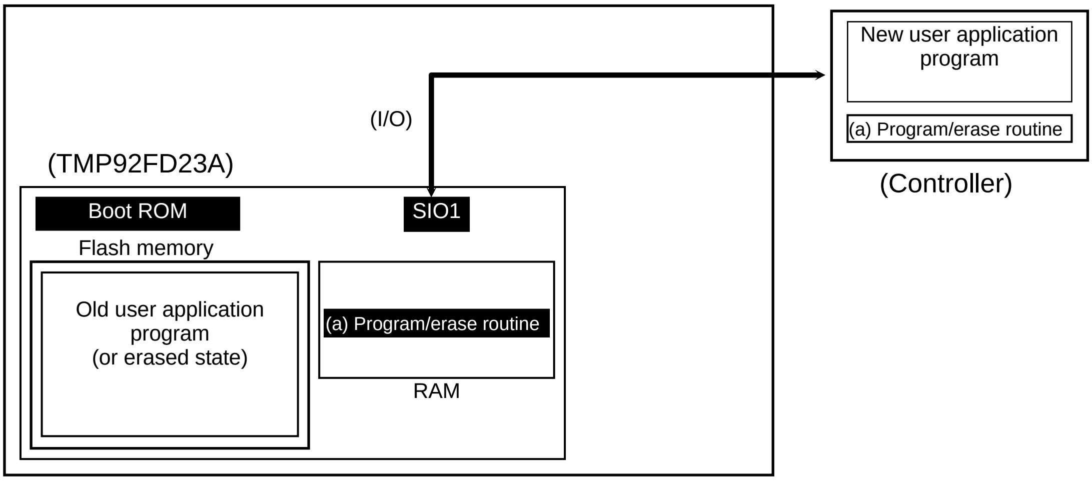
(Step-2) Starting up the internal boot ROM

Release the reset with the relevant input pins set for entering Single Boot mode. When the internal boot ROM starts up, the program/erase routine (a) is transferred from the controller to the internal RAM via SIO according to the communications procedure for Single Boot mode. Before this can be carried out, the password entered by the user is verified against the password written in the user application program. (If the flash memory has been erased, 12 bytes of "0xFF" are used as the password.)


New user application program
(a) Program/erase routine
(Controller)
(I/O)
(TMP92FD23A)
0 → 1 RESET
Boot ROM
SIO1
Flash memory
Old user application program (or erased state)
RAM
Condition for entering Single Boot mode

(Step-3) Copying the program/erase routine to the RAM

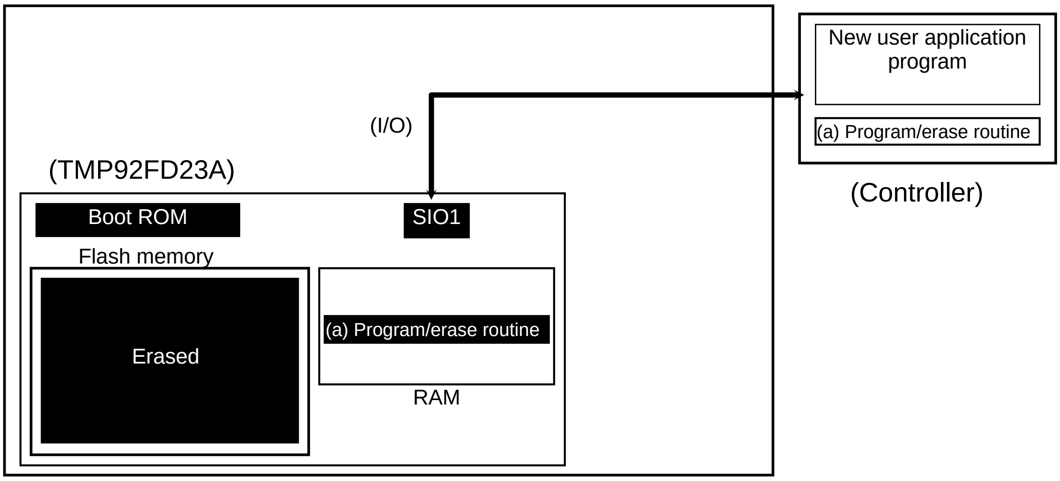
After password verification is completed, the boot ROM copies the program/erase routine (a) from the controller to the RAM using serial communications. The program/erase routine must be stored within the RAM address range of 002000H to 009DFFH.


New user application program
(a) Program/erase routine
(Controller)
(I/O)
(TMP92FD23A)
Boot ROM
SIO1
Flash memory
Old user application program (or erased state)
(a) Program/erase routine
RAM

(Step-4) Executing the program/erase routine in the RAM

Control jumps to the program/erase routine (a) in the RAM. If necessary, the old user application program is erased (sector erase or chip erase).

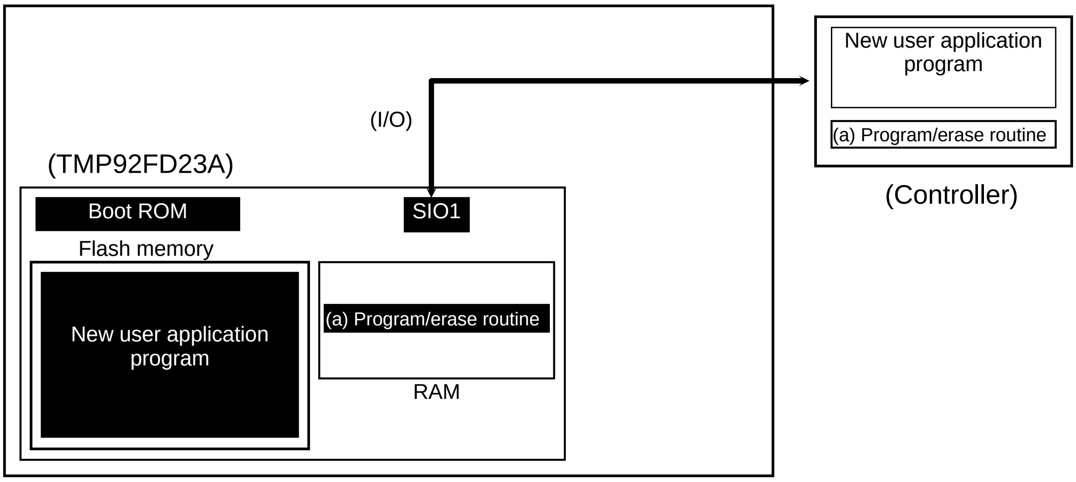
Note: The boot ROM is provided with an erase command, which enables the entire chip to be erased from the controller without using the program/erase routine. If it is necessary to erase data on a sector basis, incorporate the necessary code in the program/erase routine.


New user application program
(a) Program/erase routine
(Controller)
(I/O)
(TMP92FD23A)
Boot ROM
SIO1
Flash memory
Erased
(a) Program/erase routine
RAM

(Step-5) Copying the new user application program

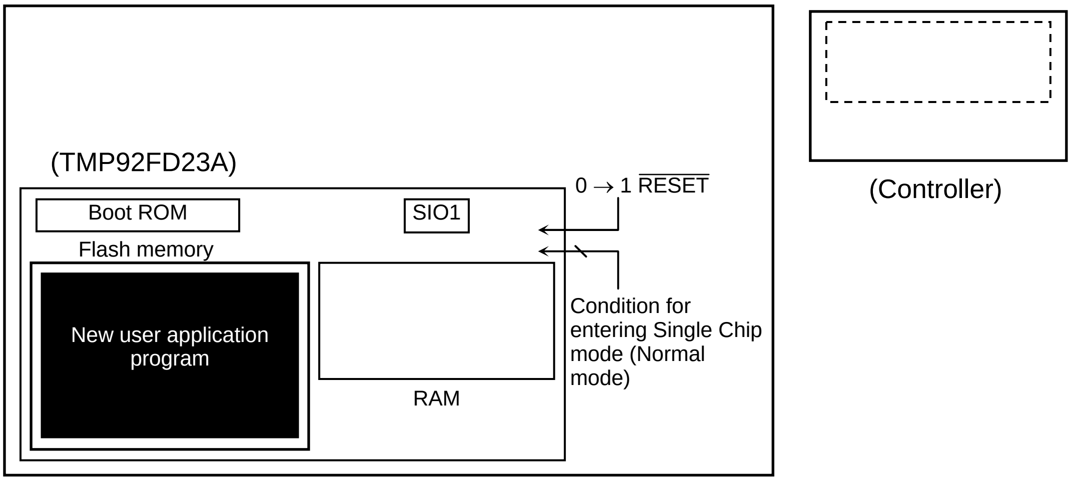
The program/erase routine (a) loads the new user application program from the controller into the erased area of the flash memory.

In the example below, the new user application program is transferred under the same communications conditions as those used for transferring the program/erase routine. However, after the program/erase routine has been transferred, this routine can be used to change the transfer settings (data bus and transfer source). Configure the board hardware and program/erase routine as desired.


(I/O)
(TMP92FD23A)
Boot ROM
SIO1
Flash memory
New user application program
(a) Program/erase routine
RAM
New user application program
(a) Program/erase routine
(Controller)

(Step-6) Executing the new user application program

After the programming operation has been completed, turn off the power to the board and remove the cable connecting the device and the controller. Then, turn on the power again and start up the device in Single Chip mode to execute the new user application program.


(TMP92FD23A)
0 → 1 $\overline{\text{RESET}}$
Boot ROM
SIO1
Flash memory
New user application program
RAM
Condition for entering Single Chip mode (Normal mode)
(Controller)

3.16.4.2 Connection Examples for Single Boot Mode

In Single Boot mode the flash memory is programmed by serial transfer. Therefore, on-board programming is performed by connecting the device's SIO (SIO1) and the controller (programming tool) and sending commands from the controller to the device. Figure 3.16.4 shows an example of connection between the target board and a programming controller. Figure 3.16.5 shows an example of connection between the target board and an RS232C board.

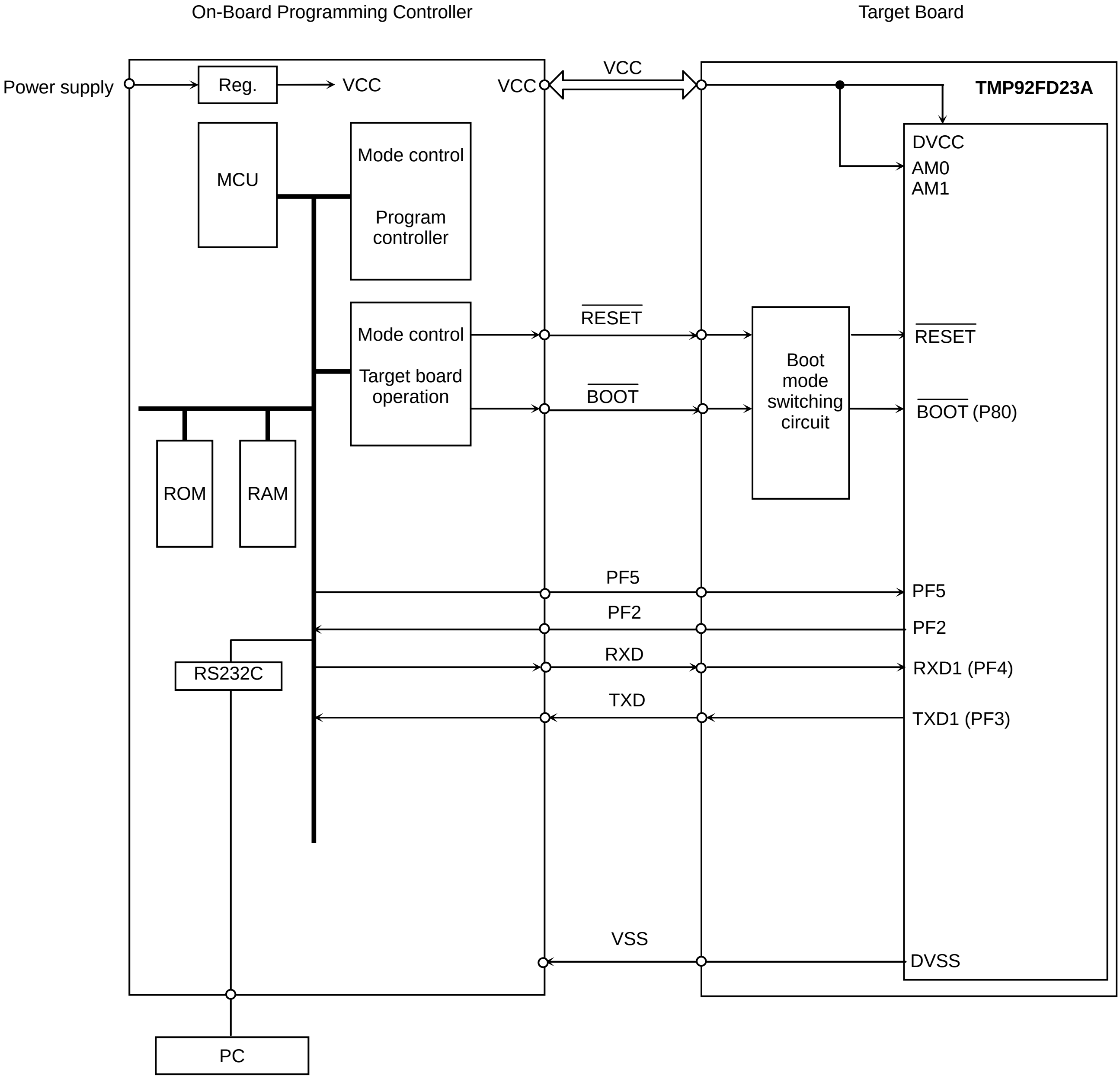


Figure 3.16.4 Example of Connection with an External Controller in Single Boot Mode

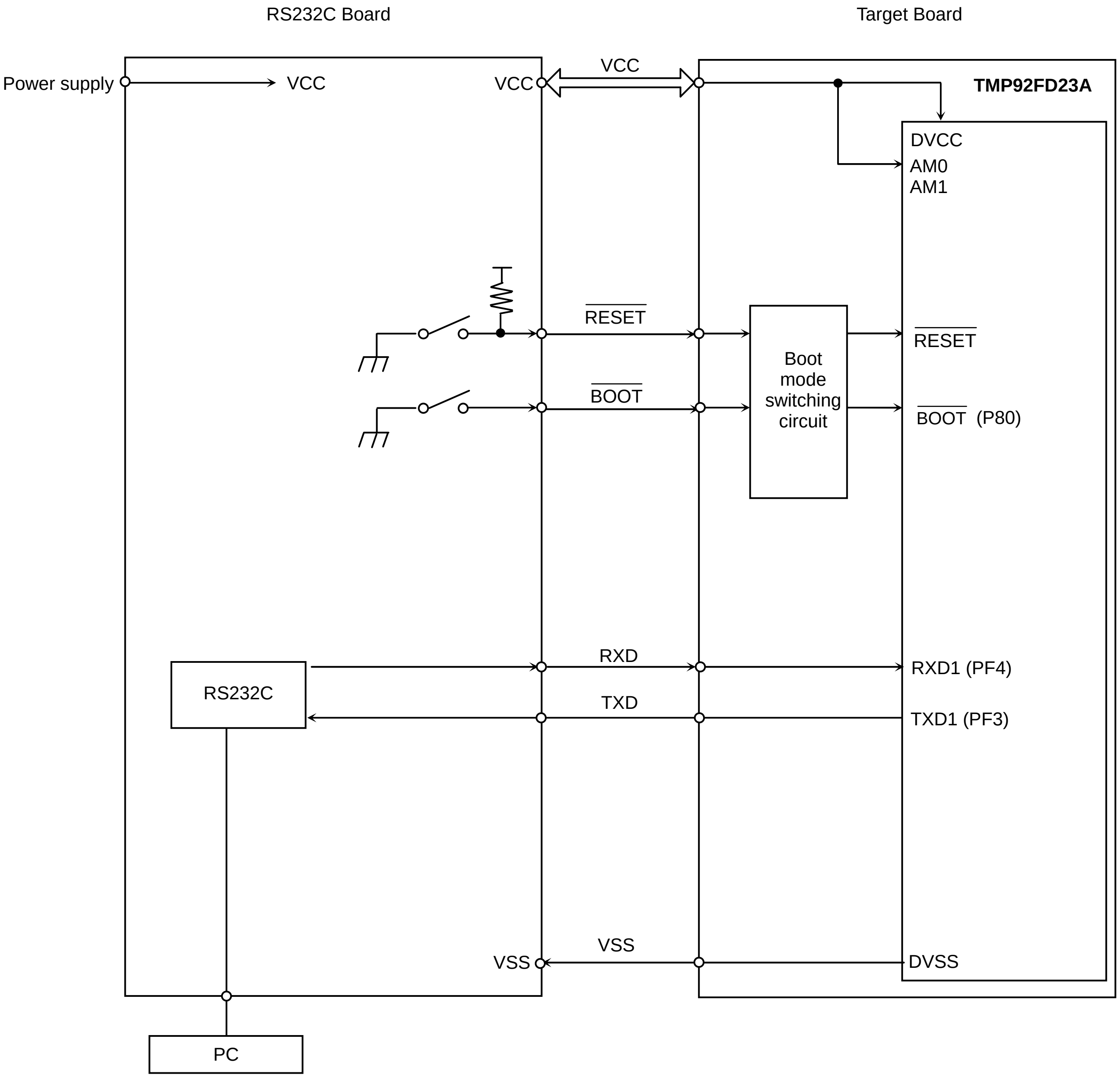


Figure 3.16.5 Example of Connection with an RS232C Board in Single Boot Mode

3.16.4.3 Mode Setting

To perform on-board programming, the device must be started up in Single Boot mode by setting the input pins as shown below.

- AM0,AM1 = 1
- $\overline{\text{BOOT}}$ = 0
- $\overline{\text{RESET}}$ = 0 → 1

Set the AM0, AM1, and $\overline{\text{BOOT}}$ pins as shown above with the $\overline{\text{RESET}}$ pin held at "0". Then, setting the $\overline{\text{RESET}}$ pin to "1" will start up the device in Single Boot mode.

3.16.4.4 Memory Maps

Figure 3.16.6 shows a comparison of the memory map for Normal mode (in Single Chip mode) and the memory map for Single Boot mode. In Single Boot mode, the flash memory is mapped to addresses 10000H to 8FFFFH (physical addresses) and the boot ROM (mask ROM) is mapped to addresses FFF000H to FFFFFFH.

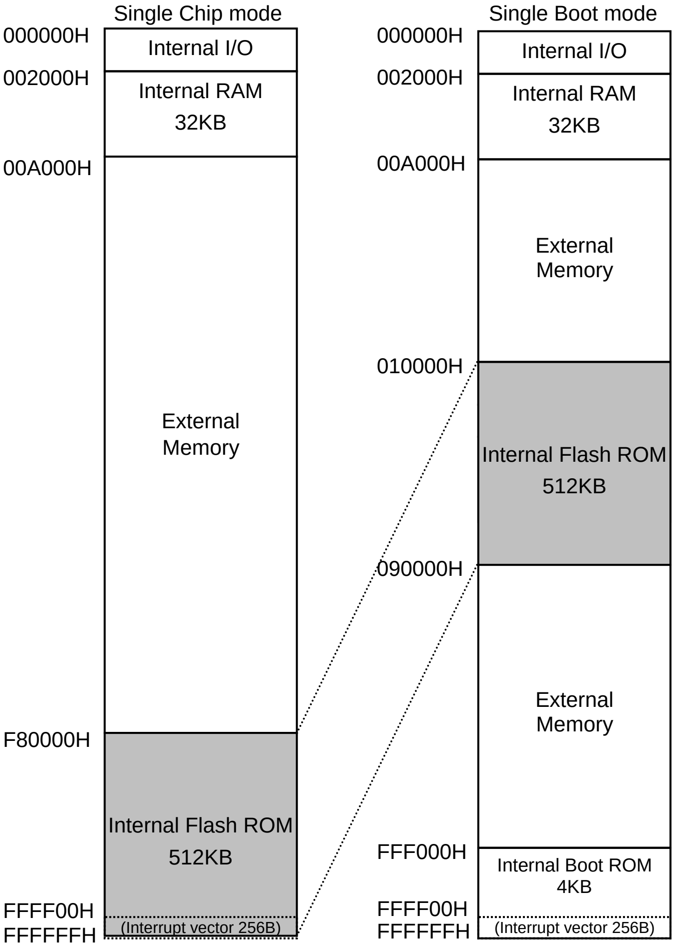


Figure 3.16.6 Comparison of Memory Maps

3.16.4.5 Interface Specifications

The SIO communications format in Single Boot mode is shown below. The device supports the UART (asynchronous communications) serial operation mode.

To perform on-board programming, the same communications format must also be set on the programming controller's side.

● UART (asynchronous) communications

· Communications channel	: SIO channel 1 (For the pins to be used, see Table 3.16.4.)
· Serial transfer mode	: UART (asynchronous communications) mode
· Data length	: 8 bits
· Parity bit	: None
· Stop bit	: 1 bit
· Baud rate	: See Table 3.16.5 and Table 3.16.6.

Table 3.16.4 Pin Connections

Pins		UART
Power supply pins	DVCC	○
	DVSS	○
Mode setting pins	AM1,AM0, $\overline{\text{BOOT}}$	○
Reset pin	$\overline{\text{RESET}}$	○
Communications pins	TXD1	○
	RXD1	○

Note: Unused pins are in the initial state after reset release.

Table 3.16.5 Baud Rate Table

SIO	Transfer Rate (bps)				
UART	115200	57600	38400	19200	9600

Table 3.16.6 Correspondence between Operating Frequency and Baud Rate in Single Boot Mode

Reference Baud Rate (bps)		9600		19200		38400		57600		115200	
Reference Frequency (MHz)	Supported Range (MHz)	Baud Rate (bps)	Error (%)	(bps)	(%)	(bps)	(%)	(bps)	(%)	(bps)	(%)
6~10	—	(Note1)	—	—	—	—	—	—	—	—	—
8	7.83~8.14	9615	+0.16	19231	+0.16	38462	+0.16	—	—	—	—
9.2160	9.04~9.40	9600	0	19200	0	38400	0	57600	0	115200	0
9.8304	9.64~10.0	9600	0	19200	0	38400	0	—	—	—	—
10	9.94~10.0	9766	+1.73	19531	+1.73	39063	+1.73	56818	–1.36	—	—

Reference frequency: The frequency of the high-speed oscillation circuit that can be used in Single Boot mode.
To program the flash memory using Single Boot mode, one of the reference frequencies must be selected as a high-speed clock.

Supported Range: The range of clock frequencies that are detected as each reference frequency. It may not be possible to perform Single Boot operations at clock frequencies outside of the supported range.

Note1: To automatically detect the reference frequency (microcontroller clock frequency), the transfer baud rate error of the flash memory programming controller and the oscillation frequency error must be within ±2% in total.

Note 2: The single boot mode can be used in all the operation frequencies (X1=6 to 10MHz) when the baud rate on the flash memory programming controller side is 9600bps. Please change to the baud rate of the desire by executing the program (user-created boot program) on RAM by using RAM Transfer command when the baud rate is changed after an initial communication is executed by 9600bps. For the TOSHIBA flash programmer, the baud rate change from 9600 to 115200bps is supported.

3.16.4.6 Data Transfer Formats

Table 3.16.7 to Table 3.16.13 show the operation command data and the data transfer format for each operation mode.

Table 3.16.7 Operation Command Data

Operation Command Data	Operation Mode
10H	RAM Transfer
20H	Flash Memory SUM
30H	Product Information Read
40H	Flash Memory Chip Erase
60H	Flash Memory Protect Set

Table 3.16.8 Transfer Format of Single Boot Program [RAM Transfer]

	Transfer Byte Number	Transfer Data from Controller to Device	Baud Rate	Transfer Data from Device to Controller
Boot ROM	1st byte	Baud rate setting UART 86H	Desired baud rate (Note 1)	—
	2nd byte	—		ACK response to baud rate setting Normal (baud rate OK) ･UART 86H (If the desired baud rate cannot be set, operation is terminated.)
	3rd byte	Operation command data (10H)		—
	4th byte	—		ACK response to operation command (Note 2) Normal 10H Error x1H Protection applied (Note 4) x6H Communications error x8H
	5th byte to 16th byte	Password data (12 bytes) (08FEF4H to 08FEFFH)		—
	17th byte	CHECKSUM value for 5th to 16th bytes		—
	18th byte	—		ACK response to CHECKSUM value (Note 2) Normal 10H Error 11H Communications error 18H
	19th byte	RAM storage start address 31 to 24 (Note 3)		—
	20th byte	RAM storage start address 23 to 16 (Note 3)		—
	21st byte	RAM storage start address 15 to 8 (Note 3)		—
	22nd byte	RAM storage start address 7 to 0 (Note 3)		—
	23rd byte	RAM storage byte count 15 to 8 (Note 3)		—
	24th byte	RAM storage byte count 7 to 0 (Note 3)		—
	25th byte	CHECKSUM value for 19th to 24th bytes (Note 3)		—
	26th byte	—		ACK response to CHECKSUM value (Note 2) Normal 10H Error 11H Communications error 18H
	27th byte to m’th byte	RAM storage data		—
	(m + 1)th byte	CHECKSUM value for 27th to m’th bytes		—
	(m + 2)th byte	—		ACK response to CHECKSUM value (Note 2) Normal 10H Error 11H Communications error 18H
RAM	(m + 3)th byte	—		Jump to RAM storage start address

Note 1: **For the desired baud rate setting, see Table 3.16.6.**

Note 2: **After sending an error response, the device waits for operation command data (3rd byte).**

Note 3: **The data to be transferred in the 19th to 25th bytes should be programmed within the RAM address range of 002000H to 009DFFH (32.256 Kbytes).**

Note 4: **When read protection or write protection is applied, the device aborts the received operation command and waits for the next operation command data (3rd byte).**

Table 3.16.9 Transfer Format of Single Boot Program [Flash Memory SUM]

	Transfer Byte Number	Transfer Data from Controller to Device	Baud Rate	Transfer Data from Device to Controller
Boot ROM	1st byte	Baud rate setting UART 86H	Desired baud rate (Note1)	—
	2nd byte	—		ACK response to baud rate setting Normal (baud rate OK) ・UART 86H (If the desired baud rate cannot be set, operation is terminated.)
	3rd byte	Operation command data (20H)		—
	4th byte	—		ACK response to operation command (Note 2) Normal 20H Error x1H Communications error x8H
	5th byte	—		SUM (upper)
	6th byte	—		SUM (lower)
	7th byte	—		CHECKSUM value for 5th and 6th bytes
	8th byte	(Wait for the next operation command data)		—

Note 1: For the desired baud rate setting, see Table 3.16.6.
Note 2: After sending an error response, the device waits for operation command data (3rd byte).

Table 3.16.10 Transfer Format of Single Boot Program [Product Information Read] (1/2)

	Transfer Byte Number	Transfer Data from Controller to Device	Baud Rate	Transfer Data from Device to Controller
Boot ROM	1st byte	Baud rate setting UART 86H	Desired baud rate (Note 1)	—
	2nd byte	—		ACK response to baud rate setting Normal (baud rate OK) ・UART 86H (If the desired baud rate cannot be set, operation is terminated.)
	3rd byte	Operation command data (30H)		—
	4th byte	—		ACK response to operation command (Note 2) Normal 30H Error x1H Communications error x8H
	5th byte	—		Flash memory data (address 08FEF0H)
	6th byte	—		Flash memory data (address 08FEF1H)
	7th byte	—		Flash memory data (address 08FEF2H)
	8th byte	—		Flash memory data (address 08FEF3H)
	9th byte to 20th byte	—		Part number (ASCII code, 12 bytes) ‘TMP92FD23_ _ _ ’ (from 9th byte)
	21st byte to 24th byte	—		Password comparison start address (4 bytes) F4H, FEH, 08H, 00H (from 21st byte)
	25th byte to 28th byte	—		RAM start address (4 bytes) 00H, 20H, 00H, 00H (from 25th byte)
	29th byte to 32nd byte	—		RAM (user area) end address (4 bytes) FFH, 9DH, 00H, 00H (from 29th byte)
	33rd byte to 36th byte	—		RAM end address (4 bytes) FFH, 9FH, 00H, 00H (from 33rd byte)
	37th byte to 40th byte	—		Dummy data (4 bytes) 00H,00H,00H,00H (from 37th byte)
	41st byte to 44th byte	—		Dummy data (4 bytes) 00H, 00H, 00H, 00H (from 41st byte)
	45th byte to 46th byte	—		FUSE information (2 bytes from 45th byte) Read protection/Write protection 1) Applied/Applied : 00H, 00H 2) Not applied/Applied : 01H, 00H 3) Applied/Not applied : 02H, 00H 4) Not applied/Not applied : 03H, 00H
	47th byte to 50th byte	—		Flash memory start address (4 bytes) 00H, 00H, 01H, 00H (from 47th byte)
	51st byte to 54th byte	—		Flash memory end address (4 bytes) FFH, FFH, 08H, 00H (from 51st byte)
	55th byte to 56th byte	—		Number of sectors in flash memory (2 bytes) 80H, 00H (from 55th byte)
	57th byte to 60th byte	—		Start address of flash memory sectors of the same size (4 bytes) 00H, 00H, 01H, 00H (from 57th byte)

Table 3.16.11 Transfer Format of Single Boot Program [Product Information Read] (2/2)

	Transfer Byte Number	Transfer Data from Controller to Device	Baud rate	Transfer Data from Device to Controller
Boot ROM	61st byte to 64th byte	—		Size (in half words) of flash memory sectors of the same size (4 bytes) 00H, 08H, 00H, 00H (from 61st byte)
	65th byte	—		Number of flash memory sectors of the same size (1 byte) 80H
	66th byte	—		CHECKSUM value for 5th to 65th bytes
	67th byte	(Wait for the next operation command data)		—

Note 1: For the desired baud rate setting, see Table 3.16.6.
Note 2: After sending an error response, the device waits for operation command data (3rd byte).

Table 3.16.12 Transfer Format of Single Boot Program [Flash Memory Chip Erase]

	Transfer Byte Number	Transfer Data from Controller to Device	Baud Rate	Transfer Data from Device to Controller
Boot ROM	1st byte	Baud rate setting UART 86H	Desired baud rate (Note 1)	—
	2nd byte	—		ACK response to baud rate setting Normal (baud rate OK) ・UART 86H (If the desired baud rate cannot be set, operation is terminated.)
	3rd byte	Operation command data (40H)		—
	4th byte	—		ACK response to operation command (Note2) Normal 40H Error x1H Communications error x8H
	5th byte	Erase Enable command data (54H)		—
	6th byte	—		ACK response to operation command (Note 2) Normal 54H Error x1H Communications error x8H
	7th byte	—		ACK response to Erase command Normal 4FH Error 4CH
	8th byte	—		ACK response Normal 5DH Error 60H
	9th byte	(Wait for the next operation command data)		—

Note 1: For the desired baud rate setting, see Table 3.16.6.
Note 2: After sending an error response, the device waits for operation command data (3rd byte).

Table 3.16.13 Transfer Format of Single Boot Program [Flash Memory Protect Set]

	Transfer Byte Number	Transfer Data from Controller to Device	Baud Rate	Transfer Data from Device to Controller
Boot ROM	1st byte	Baud rate setting UART 86H	Desired baud rate (Note 1)	—
	2nd byte	—		ACK response to baud rate setting Normal (baud rate OK) ・UART 86H (If the desired baud rate cannot be set, operation is terminated.)
	3rd byte	Operation command data (60H)		—
	4th byte	—		ACK response to operation command (Note2) Normal 60H Error x1H Communications error x8H
	5th byte to 16th byte	Password data (12 bytes) (08FEF4H to 08FEFFH)		—
	17th byte	CHECKSUM value for 5th to 16th bytes		—
	18th byte	—		ACK response to checksum value (Note 2) Normal 60H Error 61H Communications error 68H
	19th byte	—		ACK response to Protect Set command Normal 6FH Error 6CH
	20th byte	—		ACK response Normal 31H Error 34H
	21st byte	(Wait for the next operation command data)		—

Note 1: For the desired baud rate setting, see Table 3.16.6.
Note 2: After sending an error response, the device waits for operation command data (3rd byte).

3.16.4.7 Boot Program

When the device starts up in Single Boot mode, the boot program is activated.

The following explains the commands that are used in the boot program to communicate with the controller when the device starts up in Single Boot mode. Use this information for creating a controller for using Single Boot mode or for building a user boot environment.

1. RAM Transfer command

   In RAM transfer, data is transferred from the controller and stored in the device's internal RAM. When the transfer completes normally, the boot program will start running the transferred user program. Up to 32.256 Kbytes of data can be transferred as a user program. (This limit is implemented in the boot program to protect the stack pointer area.) The user program starts executing from the RAM storage start address.

   This RAM transfer function enables a user-created program/erase routine to be executed, allowing the user to implement their own on-board programming method. To perform on-board programming with a user program, the flash memory command sequences (see section 3.16.6) must be used. After the RAM Transfer command has been completed, the entire internal RAM area can be used.

   If read protection or write protection is applied on the device or a password error occurs, this command will not be executed.

2. Flash Memory SUM command

   This command calculates the SUM of 512 Kbytes of data in the flash memory and returns the result. There is no operation command available to the boot program for reading data from the entire area of the flash memory. Instead, this Flash Memory SUM command can be used. Reading the SUM value enables revision management of the application program.

3. Product Information Read command

   This command returns the information about the device including its part number and memory details stored in the flash memory at addresses 08FEF0H to 08FEF3H. This command can also be used for revision management of the application program.

4. Flash Memory Chip Erase command

   This command erases all the sectors in the flash memory. If read protection or write protection is applied on the device, all the sectors in the flash memory are erased and the read protection or write protection is cleared.

   Since this command is also used to restore the operation of the boot program when the password is forgotten, it does not include password verification.

5. Flash Memory Protect Set command

   This command sets both read protection and write protection on the device. However, if a password error occurs, this command will not be executed.

   When read protection is set, the flash memory cannot be read in Programmer mode. When write protection is set, the flash memory cannot be written in Programmer mode.

3.16.4.8 RAM Transfer Command (See Table 3.16.8)

1. From the controller to the device

   The data in the 1st byte is used to determine the baud rate. The 1st byte is transferred with receive operation disabled (SC1MOD0<RXE> = 0). (The baud rate is determined using an internal timer.)

   - To communicate in UART mode

     Send the value 86H from the controller to the target board using UART settings at the desired baud rate. If the serial operation mode is determined as UART, the device checks to see whether or not the desired baud rate can be set. If the device determines that the desired baud rate cannot be set, operation is terminated and no communications can be established.

2. From the device to the controller

   The data in the 2nd byte is the ACK response returned by the device for the serial operation mode setting data sent in the 1st byte. If the data in the 1st byte is found to signify UART and the desired baud rate can be set, the device returns 86H.

   - Baud rate determination

     The device determines whether or not the desired baud rate can be set. If it is found that the baud rate can be set, the boot program rewrites the BR1CR and BR1ADD values and returns 86H. If it is found that the desired baud rate cannot be set, operation is terminated and no data is returned. The controller sets a time-out time (5 seconds) after it has finished sending the 1st byte. If the controller does not receive the response (86H) normally within the time-out time, it should be considered that the device is unable to communicate. Receive operation is enabled (SC1MOD0<RXE> = 1) before 86H is written to the transmission buffer.

3. From the controller to the device

   The data in the 3rd byte is operation command data. In this case, the RAM Transfer command data (10H) is sent from the controller to the device.

4. From the device to the controller

   The data in the 4th byte is the ACK response to the operation command data in the 3rd byte. First, the device checks to see if the received data in the 3rd byte contains any error. If a receive error is found, the device returns the ACK response data for communications error (bit 3) x8H and waits for the next operation command data (3rd byte). The upper four bits of the ACK response data are undefined (They are the upper four bits of the immediately preceding operation command data).

   Next, if the data received in the 3rd byte corresponds to one of the operation commands given in Table 3.16.7, the device echoes back the received data (ACK response for normal reception). In the case of the RAM Transfer command, if read or write protection is not applied, 10H is echoed back and then execution branches to the RAM transfer processing routine. If protection is applied, the device returns the corresponding ACK response data (bit 2/1) x6H and waits for the next operation command data (3rd byte). The upper four bits of the ACK response data are undefined. (They are the upper four bits of the immediately preceding operation command data.)

   After branching to the RAM transfer processing routine, the device checks the data in the password area. For details, see 3.16.4.15 "Password".

   If the data in the 3rd byte does not correspond to any operation command, the device returns the ACK response data for operation command error (bit0) x1H and waits for the next operation command data (3rd byte). The upper four bits of the ACK response data are undefined. (They are the upper four bits of the immediately preceding operation command data.)

5. From the controller to the device

   The 5th to 16th bytes contain password data (12 bytes). The data in the 5th to 16th bytes is verified against the data at addresses 08FEF4H to 08FEFFH in the flash memory, respectively.

6. From the controller to the device

   The 17th byte contains CHECKSUM data. The CHECKSUM data sent by the controller is the two's complement of the lower 8-bit value obtained by summing the data in the 5th to 16th bytes by unsigned 8-bit addition (ignoring any overflow). For details on CHECKSUM, see 3.16.4.17 "How to Calculate CHECKSUM."

7. From the device to the controller

   The data in the 18th byte is the ACK response data to the 5th to 17th bytes (ACK response to the CHECKSUM value). The device first checks to see whether the data received in the 5th to 17th bytes contains any error. If a receive error is found, the device returns the ACK response data for communications error (bit 3) 18H and waits for the next operation command data (3rd byte). The upper four bits of the ACK response data are the upper four bits of the immediately preceding operation command data, so the value of these bits is “1”.

   Next, the device checks the CHECKSUM data in the 17th byte. This check is made to see if the lower 8-bit value obtained by summing the data in the 5th to 17th bytes by unsigned 8-bit addition (ignoring any overflow) is 00H. If the value is not 00H, the device returns the ACK response data for CHECKSUM error (bit 0) 11H and waits for the next operation command data (3rd byte).

   Finally, the device examines the result of password verification. If all the data in the 5th to 16th bytes is not verified correctly, the device returns the ACK response data for password error (bit 0) 11H and waits for the next operation command data (3rd byte).

   If no error is found in all the above checks, the device returns the ACK response data for normal reception 10 H.

8. From the controller to the device

   The data in the 19th to 22nd bytes indicates the RAM start address for storing block transfer data. The 19th byte corresponds to address bits 31 to 24, the 20th byte to address bits 23 to 16, the 21st byte to address bits 15 to 8, and the 22nd byte to address bits 7 to 0.

9. From the controller to the device

   The data in the 23rd and 24th bytes indicates the number of bytes to be transferred. The 23rd byte corresponds to bits 15 to 8 of the transfer byte count and the 24th byte corresponds to bits 7 to 0.

10. From the controller to the device

    The data in the 25th byte is CHECKSUM data. The CHECKSUM data sent by the controller is the two’s complement of the lower 8-bit value obtained by summing the data in the 19th to 24th bytes by unsigned 8-bit addition (ignoring any overflow). For details on CHECKSUM, see 3.16.4.17 “How to Calculate CHECKSUM .”

Note: The data in the 19th to 25th bytes should be placed within addresses 002000H to 009DFFH (32.256 Kbytes) in the internal RAM.

11. From the device to the controller

    The data in the 26th byte is the ACK response data to the data in the 19th to 25th bytes (ACK response to the CHECKSUM value).

    The device first checks to see whether the data received in the 19th to 25th bytes contains any error. If a receive error is found, the device returns the ACK response data for communications error (bit 3) 18H and waits for the next operation command (3rd byte). The upper four bits of the ACK response data are the upper four bits of the immediately preceding operation command data, so the value of these bits is “1”.

    Next, the device checks the CHECKSUM data in the 25th byte. This check is made to see if the lower 8-bit value obtained by summing the data in the 19th to 25th bytes by unsigned 8-bit addition (ignoring any overflow) is 00H. If the value is not 00H, the device returns the ACK response data for CHECKSUM error (bit 0) 11H and waits for the next operation command data (3rd byte).

12. From the controller to the device

    The data in the 27th to m’th bytes is the data to be stored in the RAM. This data is written to the RAM starting at the address specified in the 19th to 22nd bytes. The number of bytes to be written is specified in the 23rd and 24th bytes.

13. From the controller to the device

    The data in the (m+1)th byte is CHECKSUM data. The CHECKSUM data sent by the controller is the two’s complement of the lower 8-bit value obtained by summing the data in the 27th to m’th bytes by unsigned 8-bit addition (ignoring any overflow). For details on CHECKSUM, see 3.16.4.17 ”How to Calculate CHECKSUM.”

14. From the device to the controller

    The data in the (m + 2)th byte is the ACK response data to the 27th to (m+1)th bytes (ACK response to the CHECKSUM value).

    The device first checks to see whether the data in the 27th to (m+1)th byte contains any error. If a receive error is found, the device returns the ACK response data for communications error (bit 3) 18H and waits for the next operation command (3rd byte). The upper four bits of the ACK response are the upper four bits of the immediately preceding operation command data, so the value of these bits is “1”.

    Next, the device checks the CHECKSUM data in the (m+1)th byte. This check is made to see if the lower 8-bit value obtained by summing the data in the 27th to (m+1)th bytes by unsigned 8-bit addition (ignoring any overflow) is 00H. If the value is not 00H, the device returns the ACK response data for CHECKSUM error (bit 0) 11H and waits for the next operation command data (3rd byte).

    If no error is found in all the above checks, the device returns the ACK response data for normal reception 10H.

15. From the device to the controller

    If the ACK response data in the (m + 2)th byte is 10H (normal reception), the boot program then jumps to the RAM start address specified in the 19th to 22nd bytes.

3.16.4.9 Flash Memory SUM command (See Table 3.16.9)

1. The data in the 1st and 2nd bytes is the same as in the case of the RAM Transfer command.

2. From the controller to the device

   The data in the 3rd byte is operation command data. The Flash Memory SUM command data (20H) is sent here.

3. From the device to the controller

   The data in the 4th byte is the ACK response data to the operation command data in the 3rd byte.

   The device first checks to see if the data in the 3rd byte contains any error. If a receive error is found, the device returns the ACK response data for communications error (bit 3) x8H and waits for the next operation command data (3rd byte). The upper four bits of the ACK response data are undefined. (They are the upper four bits of the immediately preceding operation command data.)

   Then, if the data in the 3rd byte corresponds to one of the operation command values given in Table 3.16.7, the device echoes back the received data (ACK response for normal reception). In this case, 20H is echoed back and execution then branches to the flash memory SUM processing routine. If the data in the 3rd byte does not correspond to any operation command, the device returns the ACK response data for operation command error (bit 0) x1H and waits for the next operation command data (3rd byte). The upper four bits of the ACK response data are undefined. (They are the upper four bits of the immediately preceding operation command data.)

4. From the device to the controller

   The data in the 5th and 6th bytes is the upper and lower data of the SUM value, respectively. For details on SUM, see 3.16.4.16 “How to Calculate SUM .”

5. From the device to the controller

   The data in the 7th byte is CHECKSUM data. This is the two’s complement of the lower 8-bit value obtained by summing the data in the 5th and 6th bytes by unsigned 8-bit addition (ignoring any overflow).

6. From the controller to the device

   The data in the 8th byte is the next operation command data.

3.16.4.10 Product Information Read command (See Table 3.16.10 and Table 3.16.11)

1. The data in the 1st and 2nd bytes is the same as in the case of the RAM Transfer command.

2. From the controller to the device

   The data in the 3rd byte is operation command data. The Product Information Read command data (30H) is sent here.

3. From the device to the controller

   The data in the 4th byte is the ACK response data to the operation command data in the 3rd byte.

   The device first checks to see if the data in the 3rd byte contains any error. If a receive error is found, the device returns the ACK response data for communications error (bit 3) x8H and waits for the next operation command data (3rd byte). The upper four bits of the ACK response data are undefined. (They are the upper four bits of the immediately preceding operation command data.)

   Then, if the data in the 3rd byte corresponds to one of the operation command values given in Table 3.16.7, the device echoes back the received data (ACK response for normal reception). In this case, 30H is returned and execution then branches to the product information read processing routine. If the data in the 3rd byte does not correspond to any operation command, the device returns the ACK response data for operation command error (bit 0) x1H and waits for the next operation command data (3rd byte). The upper four bits of the ACK response data are undefined. (They are the upper four bits of the immediately preceding operation command data.)

4. From the device to the controller

   The data in the 5th to 8th bytes is the data stored at addresses 08FEF0H to 08FEF3H in the flash memory. By writing the ID information of software at these addresses, the version of the software can be managed. (For example, 0002H can indicate that the software is now in version 2.)

5. From the device to the controller

   The data in the 9th to 20th bytes denotes the part number of the device. ‘TMP92FD23_ _ _’ is sent in ASCII code starting from the 9th byte.

Note: An underscore (‘_’) indicates a space.

6. From the device to the controller

   The data in the 21st to 24th bytes is the password comparison start address. F4H, FEH, 08H and 00H are sent starting from the 21st byte.

7. From the device to the controller

   The data in the 25th to 28th bytes is the RAM start address. 00H, 20H, 00H and 00H are sent starting from the 25th byte.

8. From the device to the controller

   The data in the 29th to 32nd bytes is the RAM (user area) end address. FFH, 9DH, 00H and 00H are sent starting from the 29th byte.

9. From the device to the controller

   The data in the 33rd to 36th bytes is the RAM end address. FFH, 9FH, 00H and 00H are sent starting from the 33rd byte.

10. From the device to the controller

    The data in the 37th to 44th bytes is dummy data.

11. From the device to the controller

    The data in the 45th and 46th bytes contains the protection status and sector division information of the flash memory.

    ●Bit 0 indicates the read protection status.
    - 0: Read protection is applied.
    - 1: Read protection is not applied.

    ●Bit 1 indicates the write protection status.
    - 0: Write protection is applied.
    - 1: Write protection is not applied.

    ●Bit 2 indicates whether or not the flash memory is divided into sectors.
    - 0: The flash memory is divided into sectors.
    - 1: The flash memory is not divided into sectors.

    ●Bits 3 to 15 are sent as “0”.

12. From the device to the controller

    The data in the 47th to 50th bytes is the flash memory start address. 00H, 00H, 01H and 00H are sent starting from the 47th byte.

13. From the device to the controller

    The data in the 51st to 54th bytes is the flash memory end address. FFH, FFH, 08H and 00H are sent starting from the 51st byte.

14. From the device to the controller

    The data in the 55th and 56th bytes indicates the number of sectors in the flash memory. 80H and 00H are sent starting from the 55th byte.

15. From the device to the controller

    The data in the 57th to 65th bytes contains sector information of the flash memory. Sector information is comprised of the start address (starting from the flash memory start address), sector size and number of consecutive sectors of the same size. Note that the sector size is represented in word units.

    The data in the 57th to 65th bytes indicates 4 Kbytes of sectors (sector 0 to sector 127).

    For the data to be transferred, see Table 3.16.10 and Table 3.16.11.

16. From the device to the controller

    The data in the 66th byte is CHECKSUM data. This is the two’s complement of the lower 8-bit value obtained by summing the data in the 5th to 65th bytes by unsigned 8-bit addition (ignoring any overflow).

17. From the controller to the device

    The data in the 67th byte is the next operation command data.

3.16.4.11 Flash Memory Chip Erase Command (See Table 3.16.12)

1. The data in the 1st and 2nd bytes is the same as in the case of the RAM Transfer command.

2. From the controller to the device

   The data in the 3rd byte is operation command data. The Flash Memory Chip Erase command data (40H) is sent here.

3. From the device to the controller

   The data in the 4th byte is the ACK response data to the operation command data in the 3rd byte.

   The device first checks to see if the data in the 3rd byte contains any error. If a receive error is found, the device returns the ACK response data for communications error (bit 3) x8H and waits for the next operation command data (3rd byte). The upper four bits of the ACK response data are undefined. (They are the upper four bits of the immediately preceding operation command data.)

   Then, if the data in the 3rd byte corresponds to one of the operation command values given in Table 3.16.7, the device echoes back the received data (ACK response for normal reception). In this case, 40H is echoed back. If the data in the 3rd byte does not correspond to any operation command, the device returns the ACK response data for operation command error (bit 0) x1H and waits for the next operation command data (3rd byte). The upper four bits of the ACK response data are undefined. (They are the upper four bits of the immediately preceding operation command data.)

4. From the controller to the device

   The data in the 5th byte is Erase Enable command data (54H).

5. From the device to the controller

   The data in the 6th byte is the ACK response data to the Erase Enable command data in the 5th byte.

   The device first checks to see if the data in the 5th byte contains any error. If a receive error is found, the device returns the ACK response data for communications error (bit 3) x8H and waits for the next operation command data (3rd byte). The upper four bits of the ACK response data are undefined (They are the upper four bits of the immediately preceding operation command data.)

   Then, if the data in the 5th byte corresponds to the Erase Enable command data, the device echoes back the received data (ACK response for normal reception). In this case, 54H is echoed back and execution jumps to the flash memory chip erase processing routine. If the data in the 5th byte does not correspond to the Erase Enable command data, the device returns the ACK response data for operation command error (bit 0) x1H and waits for the next operation command (3rd byte). The upper four bits of the ACK response data are undefined. (They are the upper four bits of the immediately preceding operation command data.)

6. From the device to the controller

   The data in the 7th byte indicates whether or not the erase operation has completed successfully. If the erase operation has completed successfully, the device returns the end code (4FH). If an erase error has occurred, the device returns the error code (4CH).

7. From the device to the controller

   The data in the 8th byte is ACK response data. If the erase operation has completed successfully, the device returns the ACK response for erase completion (5DH). If an erase error has occurred, the device returns the ACK response for erase error (60H).

8. From the controller to the device

   The data in the 9th byte is the next operation command data.

3.16.4.12 Flash Memory Protect Set command (See Table 3.16.13)

1. The data in the 1st and 2nd bytes is the same as in the case of the RAM Transfer command.

2. From the controller to the device

   The data in the 3rd byte is operation command data. The Flash Memory Protect Set command data (60H) is sent here.

3. From the device to the controller

   The data in the 4th byte is the ACK response data to the operation command data in the 3rd byte.

   The device first checks to see if the data in the 3rd byte contains any error. If a receive error is found, the device returns the ACK response data for communications error (bit 3) x8H and waits for the next operation command data. The upper four bits of the ACK response data are undefined. (They are the upper four bits of the immediately preceding operation command data.)

   Then, if the data in the 3rd byte corresponds to one of the operation command data values given in Table 3.16.7, the device echoes back the received data (ACK response for normal reception). In this case, 60H is echoed back and execution branches to the flash memory protect set processing routine.

   After branching to this routine, the data in the password area is checked. For details, see 3.16.4.15 ”Password.”

   If the data in the 3rd byte does not correspond to any operation command, the device returns the ACK response data for operation command error (bit 0) x1H and waits for the next operation command data (3rd byte). The upper four bits of the ACK response data are undefined. (They are the upper four bits of the immediately preceding operation command data.)

4. From the controller to the device

   The data in the 5th to 16th bytes is password data (12 bytes). The data in the 5th byte is verified against the data at address 08FEF4H in the flash memory and the data in the 6th byte against the data at address 08FEF5H. In this manner, the received data is verified consecutively against the data at the specified address in the flash memory. The data in the 16th byte is verified against the data at address 08FEFFH in the flash memory.

5. From the controller to the device

   The data in the 17th byte is CHECKSUM data. The CHECKSUM data sent by the controller is the two’s complement of the lower 8-bit value obtained by summing the data in 5th to 16th bytes by unsigned 8-bit addition (ignoring any overflow). For details on CHECKSUM, see 3.16.4.17 ”How to Calculate CHECKSUM.”

6. From the device to the controller

   The data in the 18th byte is the ACK response data to the data in the 5th to 17th bytes (ACK response to the CHECKSUM value).

   The device first checks to see whether the data in the 5th to 17th bytes contains any error. If a receive error is found, the device returns the ACK response data for communications error (bit 3) 68H and waits for the next operation command data (3rd byte). The upper four bits of the ACK response data are the upper four bits of the immediately preceding operation command data, so the value of these bits is "6".

   Then, the device checks the CHECKSUM data in the 17th byte. This check is made to see if the lower 8 bits of the value obtained by summing the data in the 5th to 17th bytes by unsigned 8-bit addition (ignoring any overflow) is 00H. If the value is not 00H, the device returns the ACK response data for CHECKSUM error (bit 0) 61H and waits for the next operation command data (3rd byte).

   Finally, the device examines the result of password verification. If all the data in the 5th to 16th bytes is not verified correctly, the device returns the ACK response data for password error (bit 0) 61H and waits for the next operation command data (3rd byte).

   If no error is found in the above checks, the device returns the ACK response data for normal reception 60H.

7. From the device to the controller

   The data in the 19th byte indicates whether or not the protect set operation has completed successfully. If the operation has completed successfully, the device returns the end code (6FH). If an error has occurred, the device returns the error code (6CH).

8. From the device to the controller

   The data in the 20th byte is ACK response data. If the protect set operation has completed successfully, the device returns the ACK response data for normal completion (31H). If an error has occurred, the device returns the ACK response data for error (34H).

9. From the device to the controller

   The data in the 21st byte is the next operation command data.

3.16.4.13 ACK Response Data

The boot program notifies the controller of its processing status by sending various response data. Table 3.16.14 to Table3.16.19 show the ACK response data returned for each type of received data. The upper four bits of ACK response data are a direct reflection of the upper four bits of the immediately preceding operation command data. Bit 3 indicates a receive error and bit 0 indicates an operation command error, CHECKSUM error or password error.

Table 3.16.14 ACK Response Data to Serial Operation Mode Setting Data

Transfer Data	Meaning
86H	The device can communicate in UART mode. (Note)

Note: If the desired baud rate cannot be set, the device returns no data and terminates operation.

Table 3.16.15 ACK Response Data to Operation Command Data

Transfer Data	Meaning
x8H (Note)	A receive error occurred in the operation command data.
x6H (Note)	Terminated receive operation due to protection setting.
x1H (Note)	Undefined operation command data was received normally.
10H	Received the RAM Transfer command.
20H	Received the Flash Memory SUM command.
30H	Received the Product Information Read command.
40H	Received the Flash Memory Chip Erase command.
60H	Received the Flash Memory Protect Set command.

Note: The upper four bits are a direct reflection of the upper four bits of the immediately preceding operation command data.

Table 3.16.16 ACK Response data to CHECKSUM Data for RAM Transfer Command

Transfer Data	Meaning
18H	A receive error occurred.
11H	A CHECKSUM error or password error occurred.
10H	Received the correct CHECKSUM value.

Table 3.16.17 ACK Response Data to Flash Memory Chip Erase Operation

Transfer Data	Meaning
54H	Received the Erase Enable command.
4FH	Completed erase operation.
4CH	An erase error occurred.
5DH (Note)	Reconfirmation of erase operation
60H (Note)	Reconfirmation of erase error

Note: These codes are returned for reconfirmation of communications.

Table 3.16.18 ACK Response Data to CHECKSUM Data for Flash Memory Protect Set Command

Transfer Data	Meaning
68H	A receive error occurred.
61H	A CHECKSUM or password error occurred.
60H	Received the correct CHECKSUM value.

Table3.16.19 ACK Response Data to Flash Memory Protect Set Operation

Transfer Data	Meaning
6FH	Completed the protect (read/write) set operation.
6CH	A protect (read/write) set error occurred.
31H (Note)	Reconfirmation of protect (read/write) set operation
34H (Note)	Reconfirmation of protect (read/write) set error

Note: These codes are returned for reconfirmation of communications.

3.16.4.14 Determining Serial Operation Mode

To communicate in UART mode, the controller should transmit the data value 86H as the first byte at the desired baud rate. Figure 3.16.7 shows the waveform of this operation.

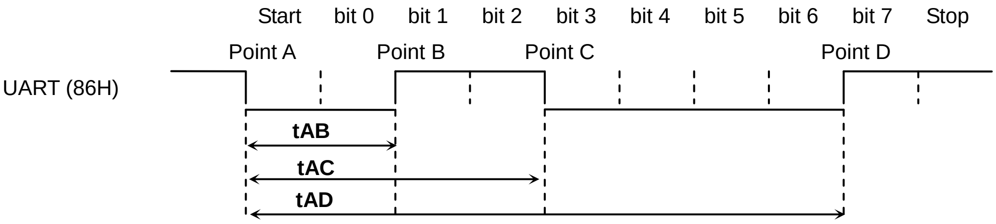


Figure 3.16.7 Data for Determining Serial Operation Mode

The boot program receives the first byte (86H) after reset release not as serial communications data. Instead, the boot program uses the first byte to determine the baud rate. The baud rate is determined by the output periods of tAB, tAC and tAD as shown in Figure 3.16.7 using the procedure shown in Figure 3.16.8.

The CPU monitors the level of the receive pin. Upon detecting a level change, the CPU captures the timer value to determine the baud rate.

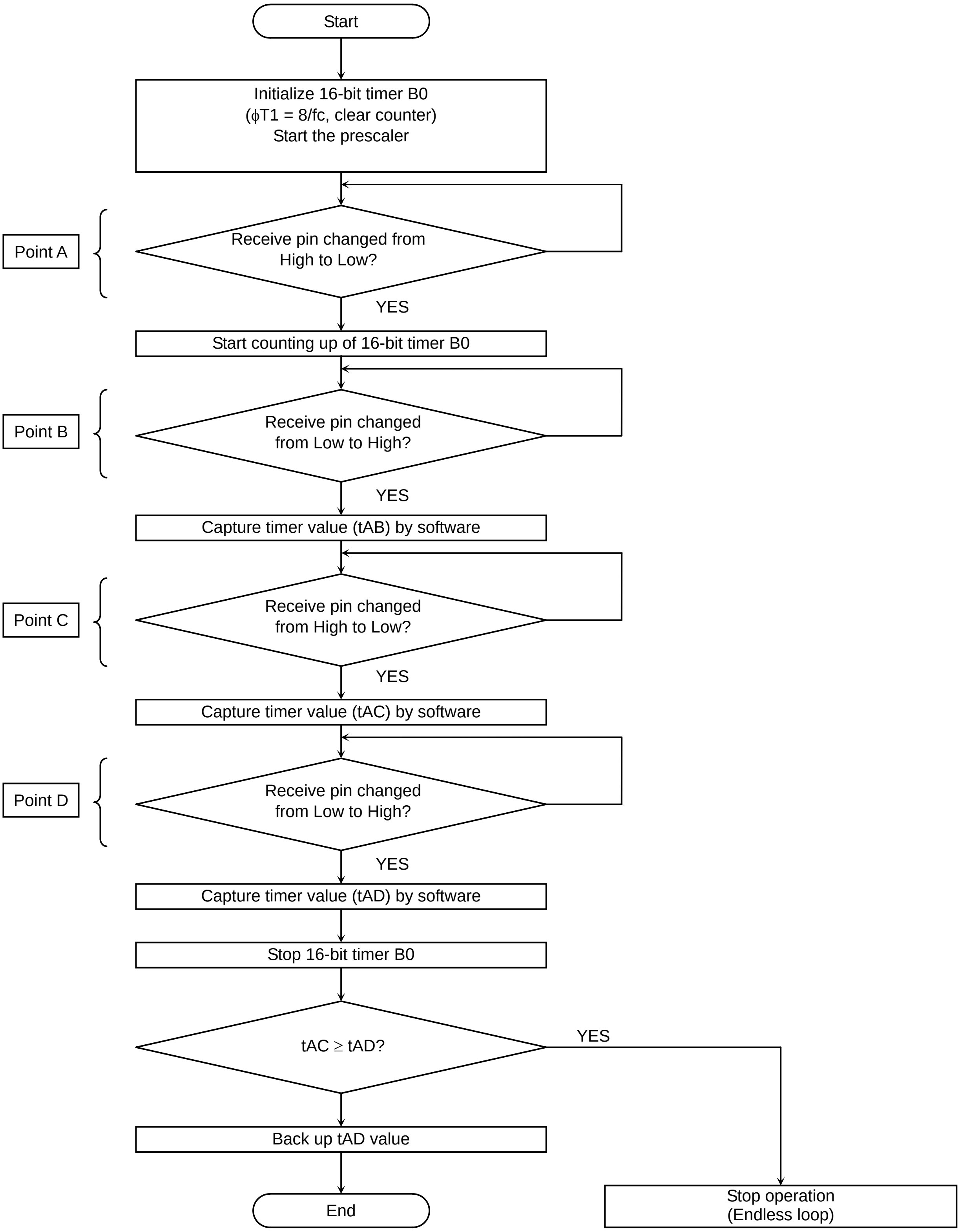

Start
Initialize 16-bit timer B0
(ϕT1 = 8/fc, clear counter)
Start the prescaler
Point A
Receive pin changed from High to Low?
YES
Start counting up of 16-bit timer B0
Point B
Receive pin changed from Low to High?
YES
Capture timer value (tAB) by software
Point C
Receive pin changed from High to Low?
YES
Capture timer value (tAC) by software
Point D
Receive pin changed from Low to High?
YES
Capture timer value (tAD) by software
Stop 16-bit timer B0
tAC ≥ tAD?
YES
Back up tAD value
End
Stop operation
(Endless loop)

Figure 3.16.8 Flowchart for Serial Operation Mode Receive Operation

3.16.4.15 Password

When the RAM Transfer command (10H) or the Flash Memory Protect Set command (60H) is received as operation command data, password verification is performed. First, the device echoes back the operation command data (10H to 60H) and checks the data (12 bytes) in the password area (addresses 08FEF4H to 08FEFFH).

Then, the device verifies the password data received in the 5th to 16th bytes against the data in the password area as shown in Table 3.16.20.

Unless all the 12 bytes are verified correctly, a password error will occur.

A password error will also occur if all the 12 bytes of password data contain the same value. Only exception is when all the 12 bytes are “FFH” and verified correctly and the reset vector area (addresses 08FF00H to 08FF02H) is all “FFH”. In this case, a blank device will be assumed and no password error will occur.

If a password error has occurred, the device returns the ACK response data for password error in the 18th byte.

Table 3.16.20 Password Verification Table

Receive data	Data to be verified against
5th byte	Data at address 08FEF4H
6th byte	Data at address 08FEF5H
7th byte	Data at address 08FEF6H
8th byte	Data at address 08FEF7H
9th byte	Data at address 08FEF8H
10th byte	Data at address 08FEF9H
11th byte	Data at address 08FEFAH
12th byte	Data at address 08FEFBH
13th byte	Data at address 08FEFCH
14th byte	Data at address 08FEFDH
15th byte	Data at address08FEFEH
16th byte	Data at address 08FEFFH

Example of data that cannot be specified as a password

For blank products (Note)

・The password of a blank product must be all “FFH” (FFH, FFH, FFH, FFH, FFH, FFH, FFH, FFH, FFH, FFH, FFH, FFH).

Note: A blank product is a product in which all the bytes in the password area (addresses 08FEF4H to 08FEFFH) and the reset vector area (addresses 08FF00H to 08FF02H) are “FFH”.

For programmed products

・The same 12 consecutive bytes cannot be specified as a password.

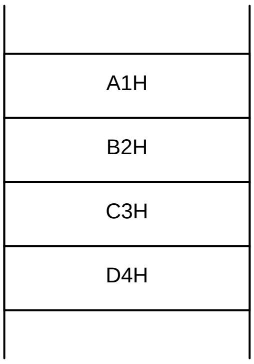
The table below shows password error examples.

Programmed product	1	2	3	4	5	6	7	8	9	10	11	12	Note
Error example 1	FFH	FFH	FFH	FFH	FFH	FFH	FFH	FFH	FFH	FFH	FFH	FFH	All ”FF”
Error example 2	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	All ”00”
Error example 3	5AH	5AH	5AH	5AH	5AH	5AH	5AH	5AH	5AH	5AH	5AH	5AH	All ”5A”

3.16.4.16 How to Calculate SUM

SUM is calculated by summing the values of all data read from the flash memory by unsigned 8-bit addition and is returned as a word (16-bit) value. The resulting SUM value is sent to the controller in order of upper 8 bits and lower 8 bits. All the 512 Kbytes of data in the flash memory are included in the calculation of SUM. When the Flash Memory SUM command is executed, SUM is calculated in this way.

Example:



When SUM is calculated from the four data entries shown to the left, the result is as follows:

A1H + B2H + C3H + D4H = 02EAH
SUM upper 8 bits: 02H
SUM lower 8 bits: EAH

Thus, the SUM value is sent to the controller in order of 02H and EAH.

3.16.4.17 How to Calculate CHECKSUM

CHECKSUM is calculated by taking the two’s complement of the lower 8-bit value obtained by summing the values of received data by unsigned 8-bit addition (ignoring any overflow). When the Flash Memory SUM command or the Product Information Read command is executed, CHECKSUM is calculated in this way. The controller should also use this CHECKSUM calculation method for sending CHECKSUM values.

Example: Calculating CHECKSUM for the Flash Memory SUM command

When the upper 8-bit data of SUM is E5H and the lower 8-bit data is F6H, CHECKSUM is calculated as shown below.

First, the upper 8 bits and lower 8 bits of the SUM value are added by unsigned operation.

E5H + F6H = 1DBH

Then, the two’s complement of the lower 8 bits of this result is obtained as shown below. The resulting CHECKSUM value (25H) is sent to the controller.

0 – DBH = 25H

3.16.5 User Boot Mode (in Single Chip Mode)

User Boot mode, which is a sub mode of Single Chip mode, enables a user-created flash memory program/erase routine to be used. To do so, the operation mode of Single Chip mode must be changed from Normal mode for executing a user application program to User Boot mode for programming/erasing the flash memory.

For example, the reset processing routine of a user application program may include a routine for selecting Normal mode or User Boot mode upon entering Single Chip mode. Any mode-selecting condition may be set using the device's I/O to suit the user system.

To program/erase the flash memory in User Boot mode, a program/erase routine must be incorporated in the user application program in advance. Since the processor cannot read data from the internal flash memory while it is being programmed or erased, the program/erase routine must be executed from the outside of the flash memory. While the flash memory is being programmed/erased in User Boot mode, interrupts must be disabled.

The pages that follow explain the procedure for programming the flash memory using two example cases. In one case the program/erase routine is stored in the internal flash memory (1-A); in the other the program/erase routine is transferred from an external source (1-B).

3.16.5.1 (1-A) Program/Erase Procedure Example 1

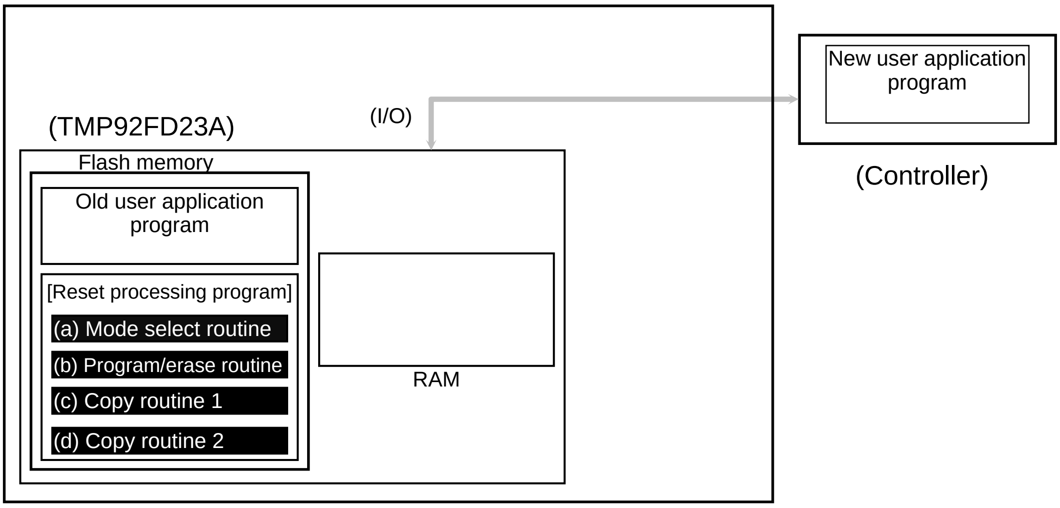
When the program/erase routine is stored in the internal flash memory

(Step-1) Environment setup

First, the condition (e.g. pin status) for entering User Boot mode must be set and the I/O bus for transferring data must be determined. Then, the device's peripheral circuitry must be designed and a corresponding program must be written. Before mounting the device on the board, it is necessary to write the following four routines into one of the sectors in the flash memory.

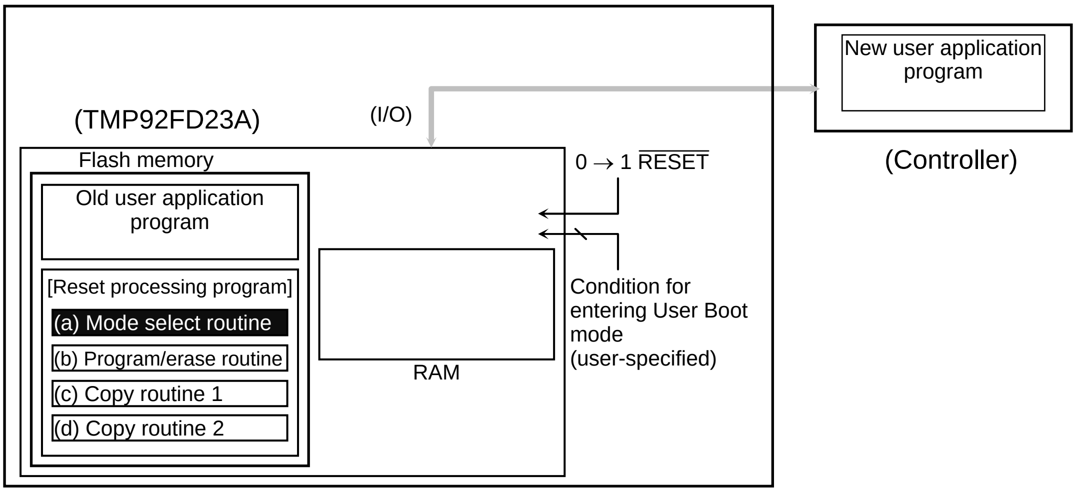
(a) Mode select routine	: Selects Normal mode or User Boot mode.
(b) Program/erase routine	: Loads program/erase data from an external source and programs/erases the flash memory.
(c) Copy routine 1	: Copies routines (a) to (d) into the internal RAM or external memory.
(d) Copy routine 2	:Copies routines (a) to (d) from the internal RAM or external memory into the flash memory.

Note: The above (d) is a routine for reconstructing the program/erase routine on the flash memory. If the entire flash memory is always programmed and the program/erase routine is included in the new user application program, this copy routine is not needed.


(TMP92FD23A)
(I/O)
Flash memory
Old user application program
[Reset processing program]
(a) Mode select routine
(b) Program/erase routine
(c) Copy routine 1
(d) Copy routine 2
RAM
New user application program
(Controller)

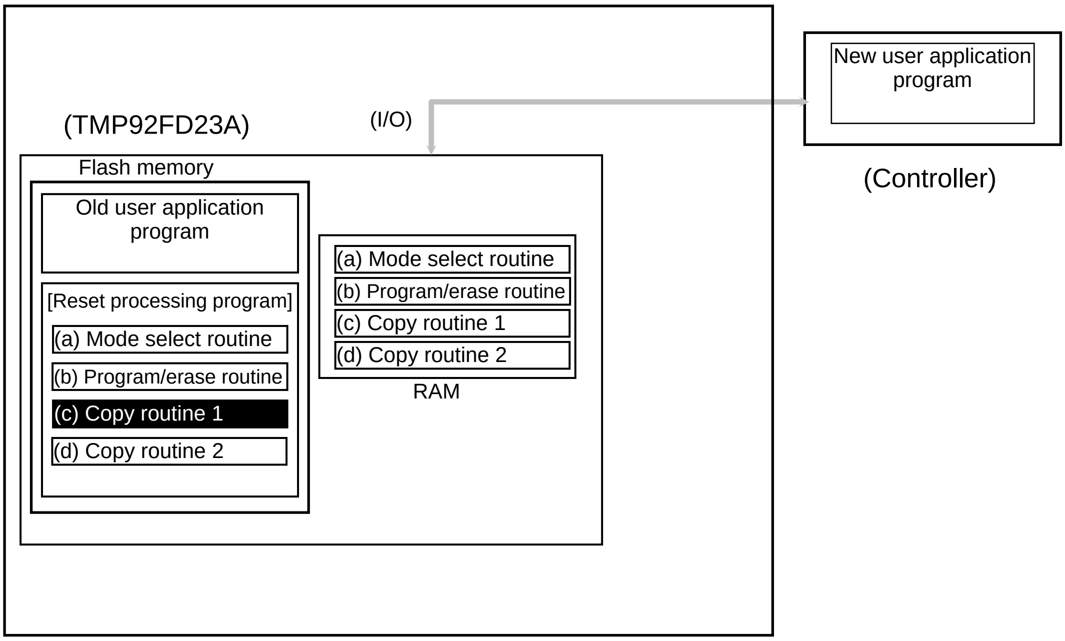
(Step-2) Entering User Boot mode (using the reset processing)

After reset release, the reset processing program determines whether or not the device should enter User Boot mode. If the condition for entering User Boot mode is true, User Boot mode is entered to program/erase the flash memory.


(TMP92FD23A)
(I/O)
Flash memory
Old user application program
[Reset processing program]
(a) Mode select routine
(b) Program/erase routine
(c) Copy routine 1
(d) Copy routine 2
RAM
0 → 1 RESET
Condition for entering User Boot mode (user-specified)
New user application program
(Controller)

(Step-3) Copying the program/erase routine

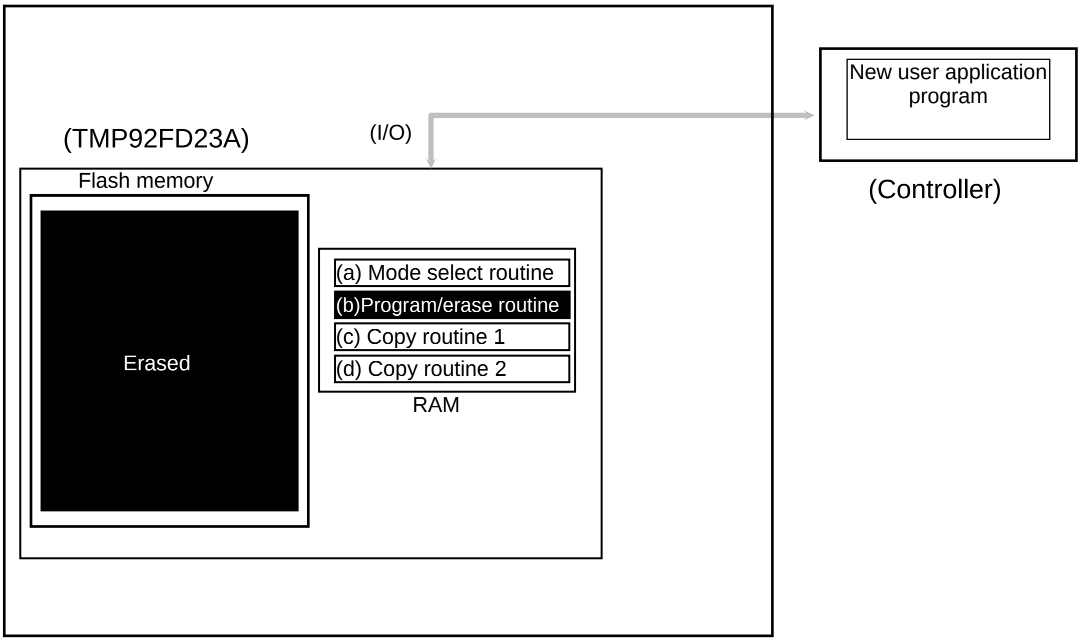
After the device has entered User Boot mode, the copy routine 1 (c) copies the routines (a) to (d) into the internal RAM or external memory (The routines are copied into the internal RAM here.)



(Step-4) Erasing the flash memory by the program/erase routine

Control jumps to the program/erase routine in the RAM and the old user program area is erased (sector erase or chip erase). (In this case, the flash memory erase command is issued from the RAM.)

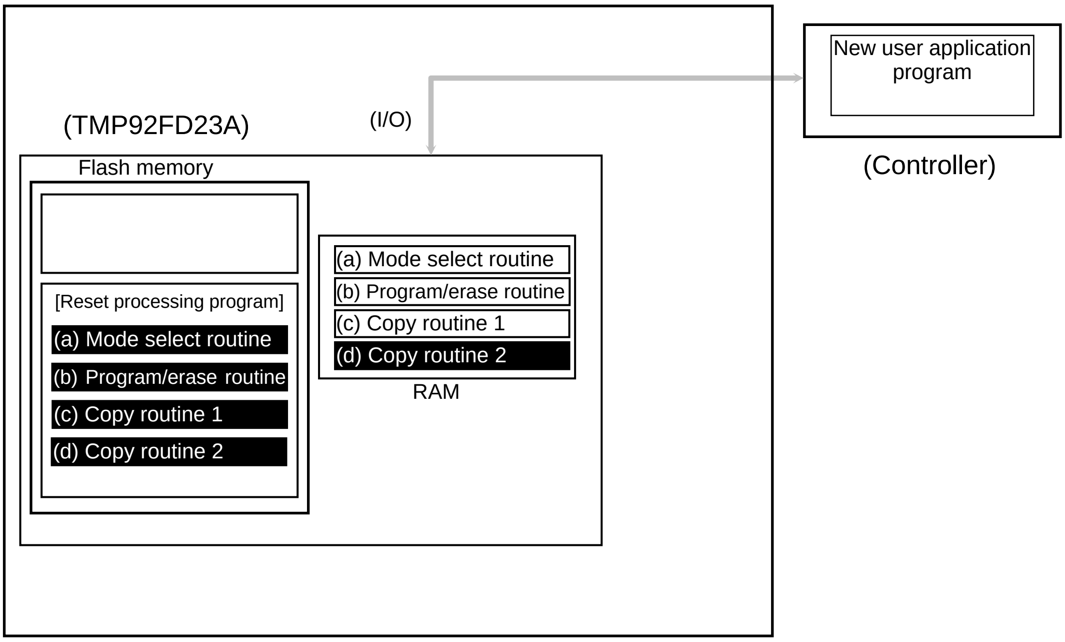
Note: If data is erased on a sector basis and the routines (a) to (d) are left in the flash memory, only the program/erase routine (b) need be copied into the RAM.



(Step-5) Restoring the user boot program in the flash memory

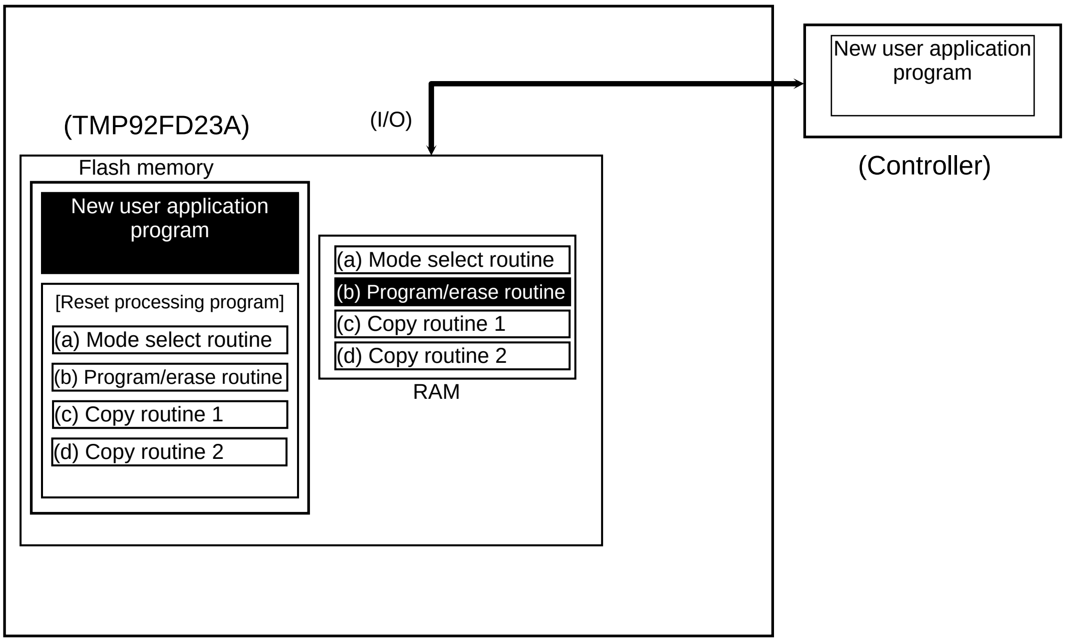
The copy routine 2 (d) in the RAM copies the routines (a) to (d) into the flash memory.

Note: If data is erased on a sector basis and the routines (a) to (d) are left in the flash memory, step 5 is not needed.



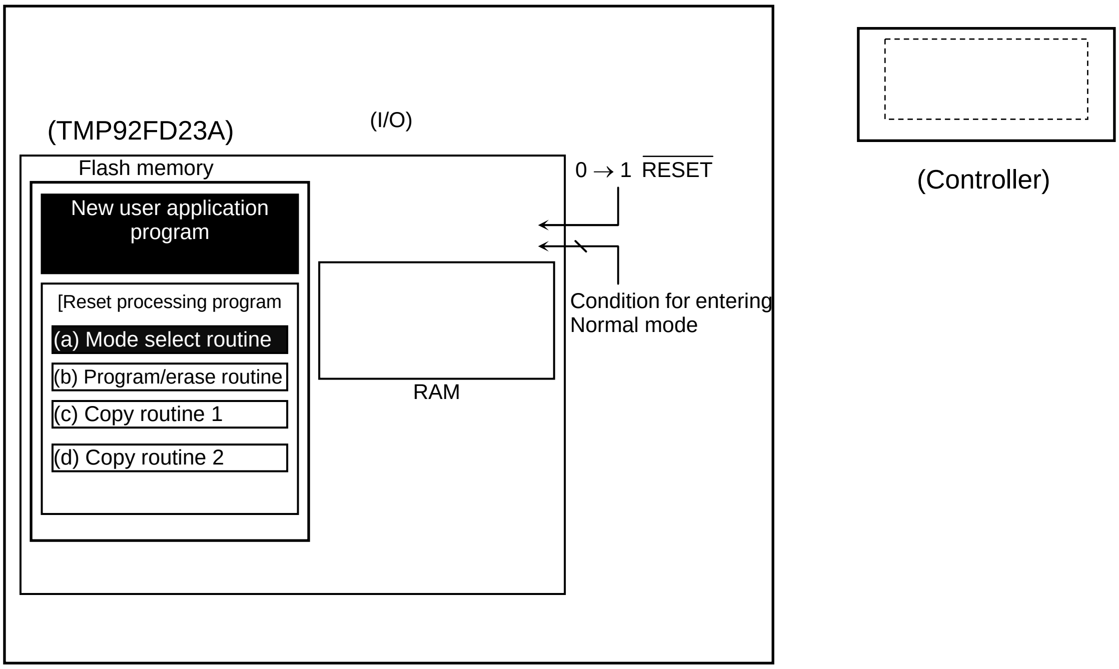
(Step-6) Writing the new user application program to the flash memory

The program/erase routine in the RAM is executed to load the new user application program from the controller into the erased area of the flash memory.



(Step-7) Executing the new user application program

The $\overline{\text{RESET}}$ input pin is driven Low (“0”) to reset the device. The mode setting condition is set for Normal mode. After reset release, the device will start executing the new user application program.



3.16.5.2 (1-B) Program/Erase Procedure Example 2

In this example, only the boot program (minimum requirement) is stored in the flash memory and other necessary routines are supplied from the controller.

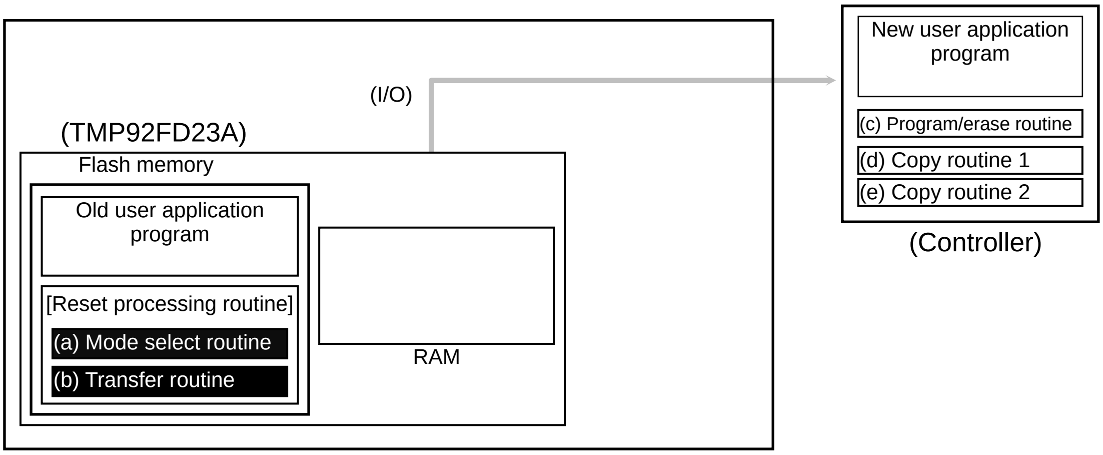
(Step-1) Environment setup

First, the condition (e.g. pin status) for entering User Boot mode must be set and the I/O bus for transferring data must be determined. Then, the device's peripheral circuitry must be designed and a corresponding program must be written. Before mounting the device on the board, it is necessary to write the following two routines into one on the sectors in the flash memory.

(a) Mode select routine : Selects Normal mode or User Boot mode.
(b) Transfer routine : Loads the program/erase routine from an external source.

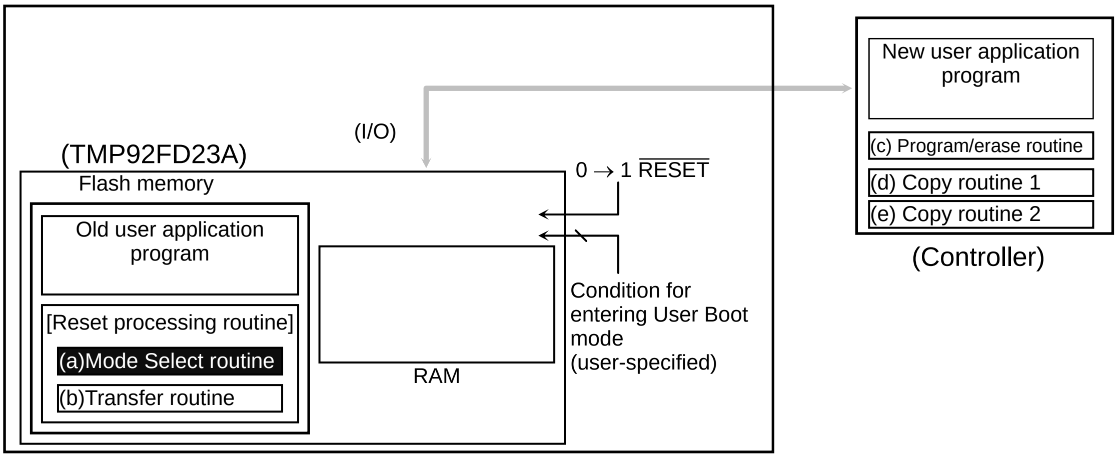
The following routines are prepared on the controller.

(c) Program/erase routine : Programs/erases the flash memory.
(d) Copy routine 1 : Copies routines (a) and (b) into the internal RAM or external memory.
(e) Copy routine 2 : Copies routines (a) and (b) from the internal RAM or external memory into the flash memory.



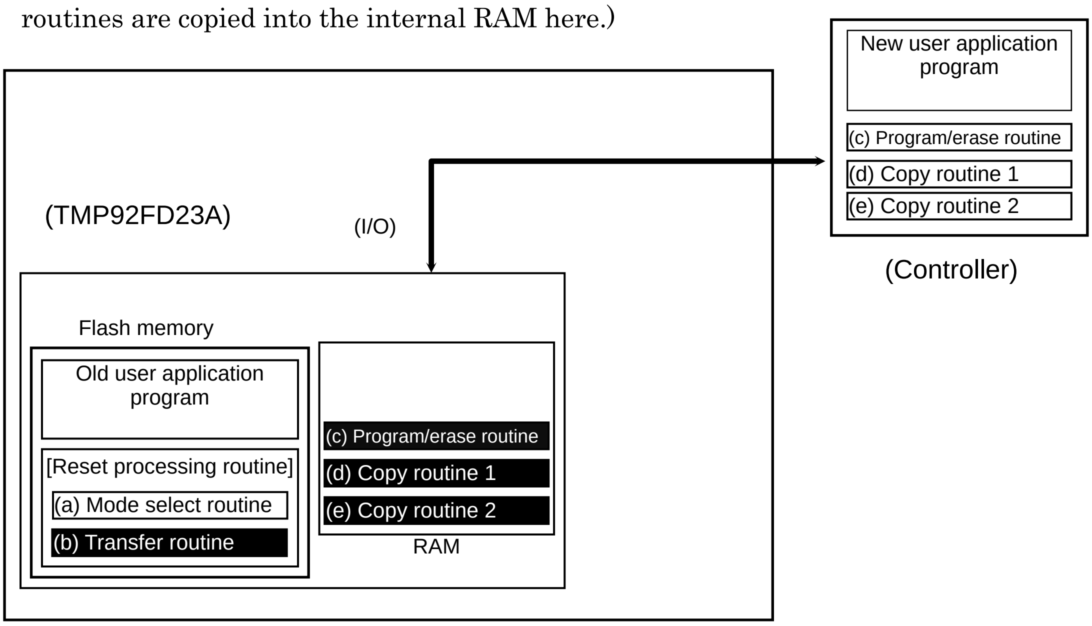
(Step-2) Entering User Boot mode (using the reset processing)

The following explanation assumes that these routines are incorporated in the reset processing program. After reset release, the reset processing program first determines whether or not the device should enter User Boot mode. If the condition for entering User Boot mode is true, User Boot mode is entered to program/erase the flash memory.


New user application program
(c) Program/erase routine
(d) Copy routine 1
(e) Copy routine 2
(Controller)
(I/O)
(TMP92FD23A)
Flash memory
0 → 1 $\overline{\text{RESET}}$
Old user application program
[Reset processing routine]
(a)Mode Select routine
(b)Transfer routine
RAM
Condition for entering User Boot mode (user-specified)

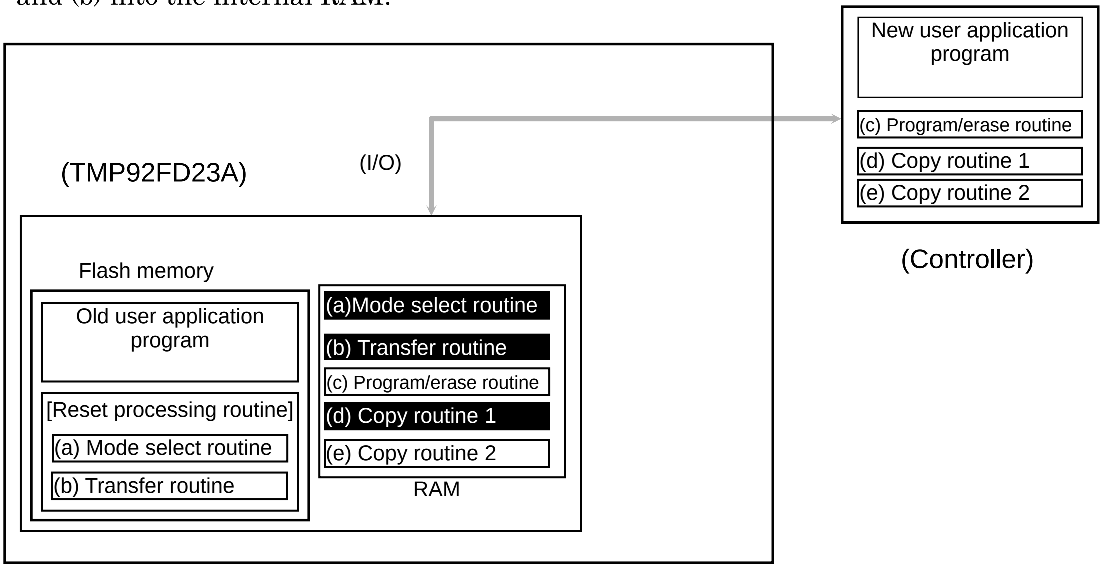
(Step-3) Copying the program/erase routine to the internal RAM

After the device has entered User Boot mode, the transfer routine (b) transfers the routines (c) to (e) from the controller to the internal RAM (or external memory). (The routines are copied into the internal RAM here.)



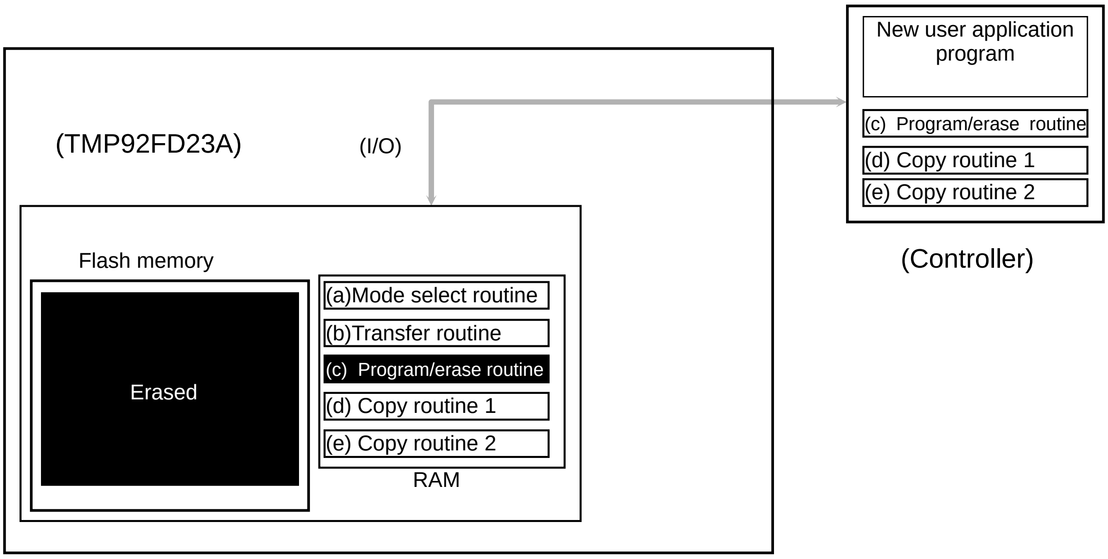
(Step-4) Executing the copy routine 1 in the internal RAM

Control jumps to the internal RAM and the copy routine 1 (d) copies the routines (a) and (b) into the internal RAM.



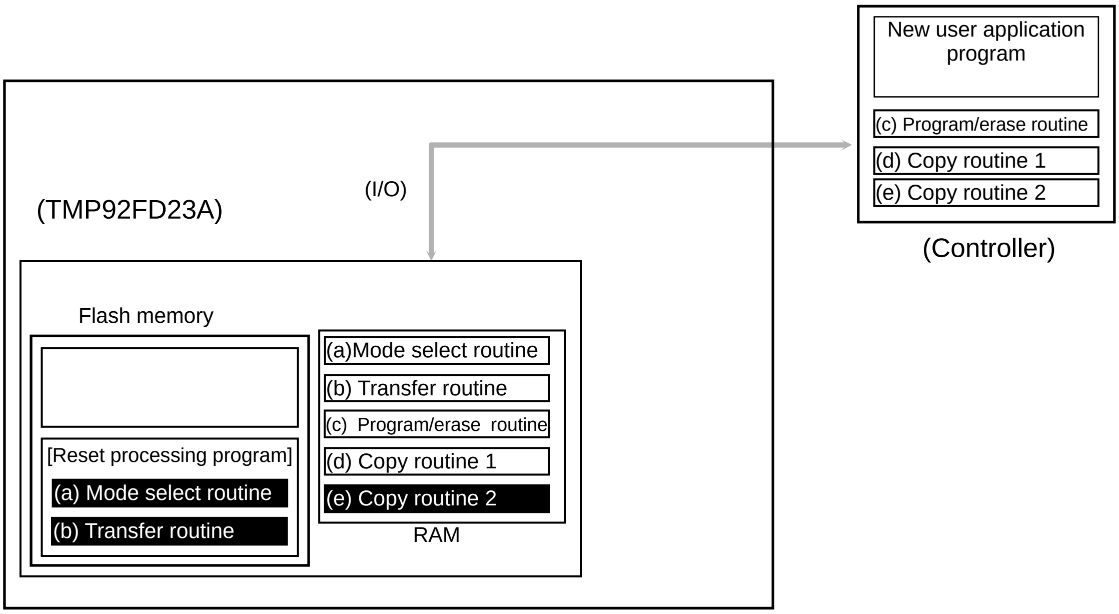
(Step-5) Erasing the flash memory by the program/erase routine

The program/erase routine (c) erases the old user program area.



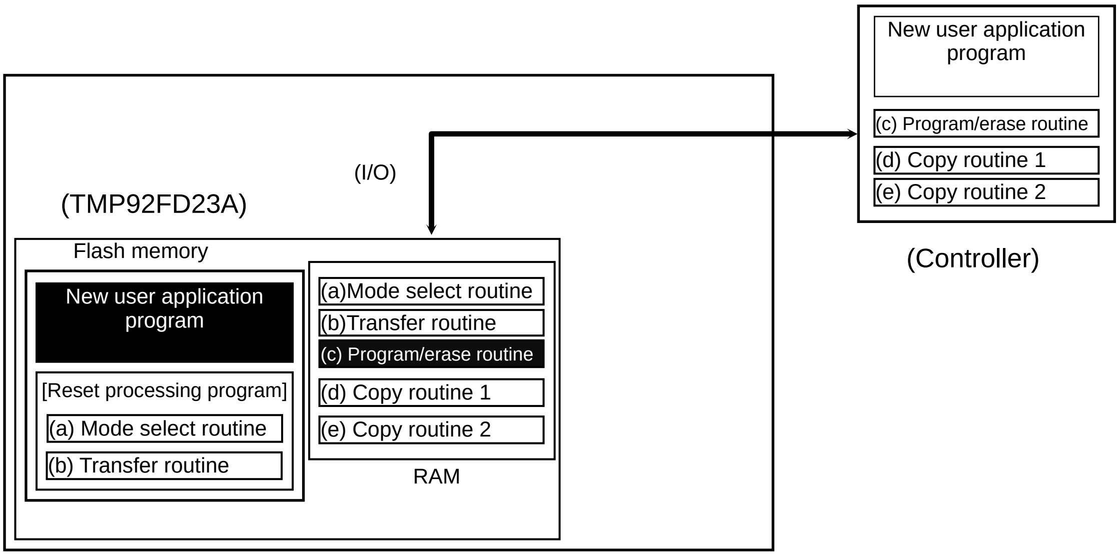
(Step-6) Restoring the user boot program in the flash memory

The copy routine (e) copies the routines (a) and (b) from the internal RAM into the flash memory.



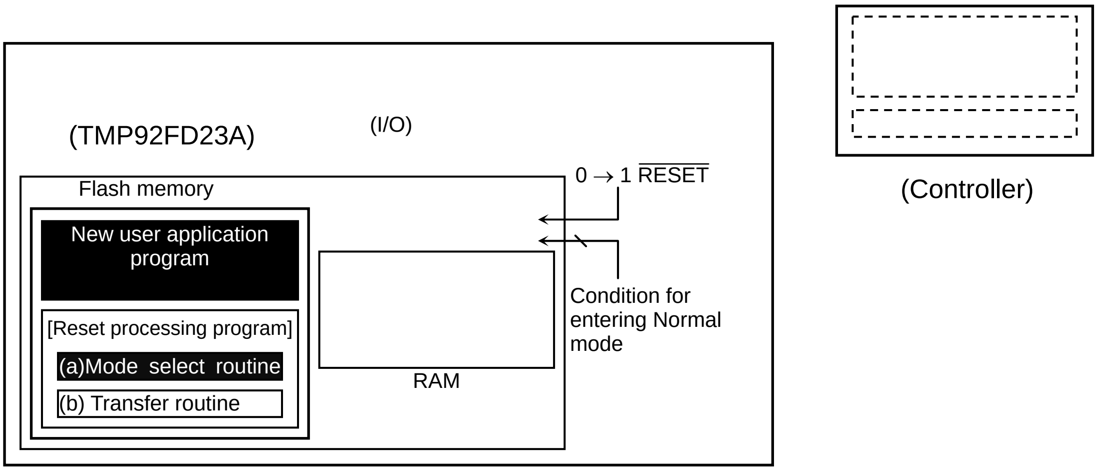
(Step-7) Writing the new user application program to the flash memory

The program/erase routine (c) in the RAM is executed to load the new user application program from the controller into the erased area of the flash memory.



(Step-8) Executing the new user application program

The $\overline{\text{RESET}}$ input pin is driven Low (“0”) to reset the device. The mode setting condition is set for Normal mode. After reset release, the device will start executing the new user application program.



3.16.6 Flash Memory Command Sequences

The operation of the flash memory is comprised of six commands, as shown in Table 3.16.21. Addresses specified in each command sequence must be in an area where the flash memory is mapped. For details, see Table 3.16.3.

Table 3.16.21 Command Sequences

	Command Sequence	1st Bus Write Cycle		2nd Bus Write Cycle		3rd Bus Write Cycle		4th Bus Write Cycle		5th Bus Write Cycle		6th Bus Write Cycle	
		Addr.	Data	Addr.	Data	Addr.	Data	Addr.	Data	Addr.	Data	Addr.	Data
1	Single Long Word Program	AA8H	AAH	550H	55H	AA8H	A0H	PA (Note 1)	PD (Note 1)				
2	Sector Erase (4-KB Erase)	AA8H	AAH	550H	55H	AA8H	80H	AA8H	AAH	550H	55H	SA (Note 2)	30H
3	Chip Erase (All Erase)	AA8H	AAH	550H	55H	AA8H	80H	AA8H	AAH	550H	55H	AA8H	10H
4	Product ID Entry	AA8H	AAH	550H	55H	AA8H	90H						
5	Product ID Exit	xxH	F0H										
	Product ID Exit	AA8H	AAH	550H	55H	AA8H	F0H						
6	Read Protect Set	AA8H	AAH	550H	55H	AA8H	A5H	778H	F0H (Note3)				
	Write Protect Set	AA8H	AAH	550H	55H	AA8H	A5H	778H	0FH (Note3)				

Note 1: PA = Program Long Word address, PD = Program Long Word data
Set the address and data to be programmed. Program addresses must be specified in multiples of 4.

Note 2: SA = Sector Erase address, Each sector erase range is selected by address A23 to A12.
The A2 of address must be specified to "1".

Note 3: When apply read protect and write protect, be sure to program the data of 00H.

Table 3.16.22 Hardware Sequence Flags

Status		D7	D6
During auto operation	Single Long Word Program	$\overline{D7}$	Toggle
	Sector Erase/Chip Erase	0	Toggle
	Read Protect Set/Write Protect Set	Cannot be used	Toggle

Note: D31 to D8 and D5 to D0 are "don't care".

3.16.6.1 Single Long Word Program

The Single Long Word Program command sequence programs the flash memory on a long word basis. The address and data to be programmed are specified in the 4th bus write cycle. It takes a maximum of 60 μs to program a single long word. Another command sequence cannot be executed until the write operation has completed. This can be checked by reading the same address in the flash memory repeatedly until the same data is read consecutively. While a write operation is in progress, bit 6 of data is toggled each time it is read.

Note: To rewrite data to Flash memory addresses at which data (including FFFF_FFFFH) is already written, make sure to erase the existing data by "sector erase" or "chip erase" before rewriting data.

3.16.6.2 Sector Erase (4-Kbyte Erase)

The Sector Erase command sequence erases 4 Kbytes of data in the flash memory at a time. The flash memory address range to be erased is specified in the 6th bus write cycle. For the address range of each sector, see Table 3.16.3. The A2 of address must be specified to "1". This command sequence cannot be used in Programmer mode.

It takes a maximum of 75 ms to erase 4 Kbytes. Another command sequence cannot be executed until the erase operation has completed. This can be checked by reading the same address in the flash memory repeatedly until the same data is read consecutively. While a erase operation is in progress, bit 6 of data is toggled each time it is read.

3.16.6.3 Chip Erase (All Erase)

The Chip Erase command sequence erases the entire area of the flash memory.

It takes a maximum of 300 ms to erase the entire flash memory. Another command sequence cannot be executed until the erase operation has completed. This can be checked by reading the same address in the flash memory repeatedly until the same data is read consecutively. While a erase operation is in progress, bit 6 of data is toggled each time it is read.

Erase operations clear data to FFH.

3.16.6.4 Product ID Entry

When the Product ID Entry command is executed, Product ID mode is entered. In this mode, the vendor ID, flash macro ID, flash size ID, and read/write protect status can be read from the flash memory. In Product ID mode, the data in the flash memory cannot be read.

3.16.6.5 Product ID Exit

This command sequence is used to exit Product ID mode.

3.16.6.6 Read Protect Set

The Read Protect Set command sequence applies read protection on the flash memory. When read protection is applied, the flash memory cannot be read in Programmer mode and the RAM Transfer and Flash Memory Program commands cannot be executed in Single Boot mode.

To cancel read protection, it is necessary to execute the Chip Erase command sequence. To check whether or not read protection is applied, read xxx778H in Product ID mode. It takes a maximum of 60 μs to set read protection on the flash memory. Another command sequence cannot be executed until the read protection setting has completed. This can be checked by reading the same address in the flash memory repeatedly until the same data can be read consecutively. While a read protect operation is in progress, bit 6 of data is toggled each time it is read.

3.16.6.7 Write Protect Set

The Write Protect Set command sequence applies write protection on the flash memory. When write protection is applied, the flash memory cannot be written to in Programmer mode and the RAM Transfer and Flash Memory Program commands cannot be executed in Single Boot mode.

To cancel write protection, it is necessary to execute the Chip Erase command sequence. To check whether or not write protection is applied, read xxx778H in Product ID mode. It takes a maximum of 60 μs to set write protection. Another command sequence cannot be executed until the write protection setting has completed. This can be checked by reading the same address in the flash memory repeatedly until the same data can be read consecutively. While a write protect operation is in progress, bit 6 of data is toggled each time it is read.

3.16.6.8 Hardware Sequence Flags

The following hardware sequence flags are available to check the auto operation execution status of the flash memory.

1) Data polling (D7)

When data is written to the flash memory, D7 outputs the complement of its programmed data until the write operation has completed. After the write operation has completed, D7 outputs the proper cell data. By reading D7, therefore, the operation status can be checked. While the Sector Erase or Chip Erase command sequence is being executed, D7 outputs "0". After the command sequence is completed, D7 outputs "1" (cell data). Then, the data written to all the bits can be read after waiting for 1 μs.

When read/write protection is applied, the data polling function cannot be used. Instead, use the toggle bit (D6) to check the operation status.

2) Toggle bit (D6)

When the Flash Memory Program, Sector Erase, Chip Erase, Write Protect Set, or Read Protect Set command sequence is executed, bit 6 (D6) of the data read by read operations outputs "0" and "1" alternately each time it is read until the processing of the executed command sequence has completed. The toggle bit (D6) thus provides a software means of checking whether or not the processing of each command sequence has completed. Normally, the same address in the flash memory is read repeatedly until the same data is read successively. The initial read of the toggle bit always returns "1".

Note: The flash memory incorporated in the TMP92FD23A does not have an exceed-time-limit bit (D5). It is therefore necessary to set the data polling time limit and toggle bit polling time limit so that polling can be stopped if the time limit is exceeded.

3.16.6.9 Data Read

Data is read from the flash memory in byte units or word units or long word units. It is not necessary to execute a command sequence to read data from the flash memory.

3.16.6.10 Programming the Flash Memory by the Internal CPU

The internal CPU programs the flash memory by using the command sequences and hardware sequence flags described above. However, since the flash memory cannot be read during auto operation mode, the program/erase routine must be executed outside of the flash memory.

The CPU can program the flash memory either by using Single Boot mode or by using a user-created protocol in Single Chip mode (User Boot).

1) Single Boot:

The microcontroller is started up in Single Boot mode to program the flash memory by the internal boot ROM program. In this mode, the internal boot ROM is mapped to an area including the interrupt vector table, in which the boot ROM program is executed. The flash memory is mapped to an address area different from the boot ROM area. The boot ROM program loads data into the flash memory by serial transfer. In Single Boot mode, interrupts must be disabled including non-maskable interrupts ($\overline{\text{NMI}}$, etc.).

For details, see 3.16.4 “Single Boot Mode”

2) User Boot:

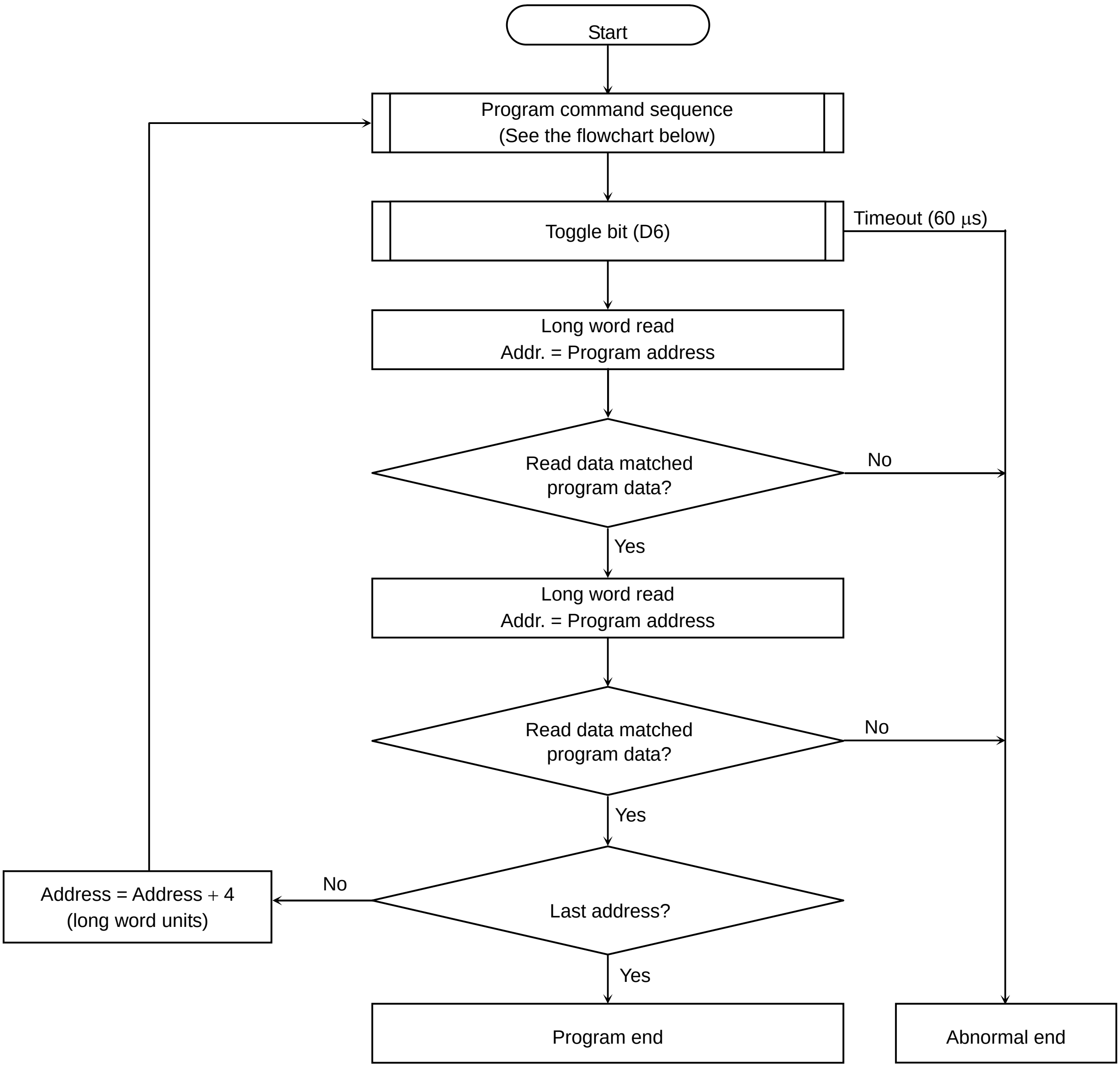
In this method, the flash memory is programmed by executing a user-created routine in Single Chip mode (normal operation mode). In this mode, the user-created program/erase routine must also be executed outside of the flash memory. It is also necessary to disable interrupts including non-maskable interrupts.

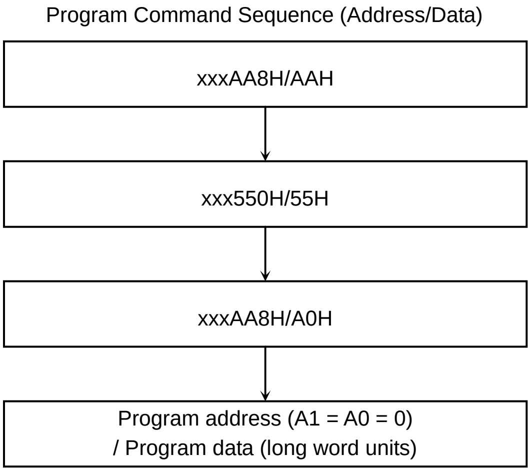
The user should prepare a flash memory program/erase routine (including routines for loading write data and writing the loaded data into the flash memory). In the main program, normal operation is switched to flash memory programming operation to execute the flash memory program/erase routine outside of the flash memory area. For example, the flash memory program/erase routine may be transferred from the flash memory to the internal RAM and executed there or it may be prepared and executed in external memory.

For details, see 3.16.5 “User Boot Mode (in Single Chip Mode)”

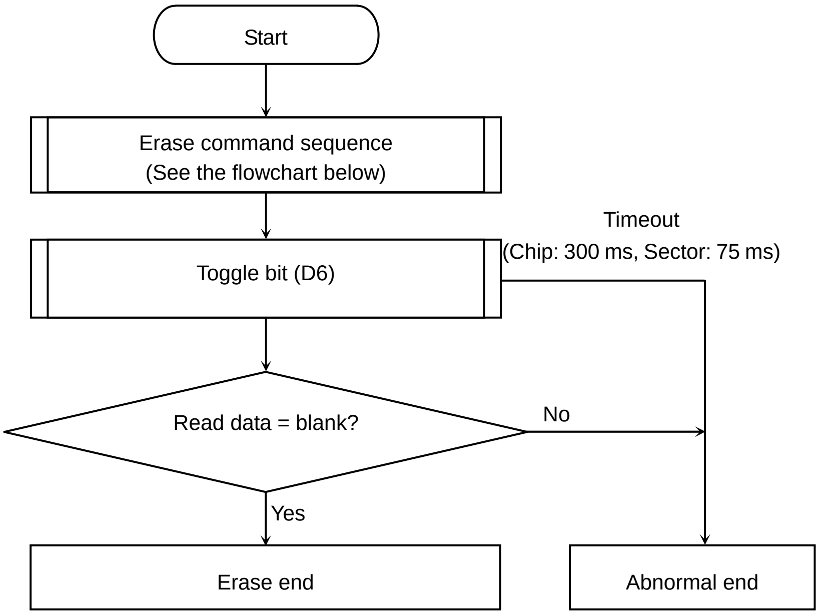
Flowcharts: Flash memory access by the internal CPU

Single Long Word Program

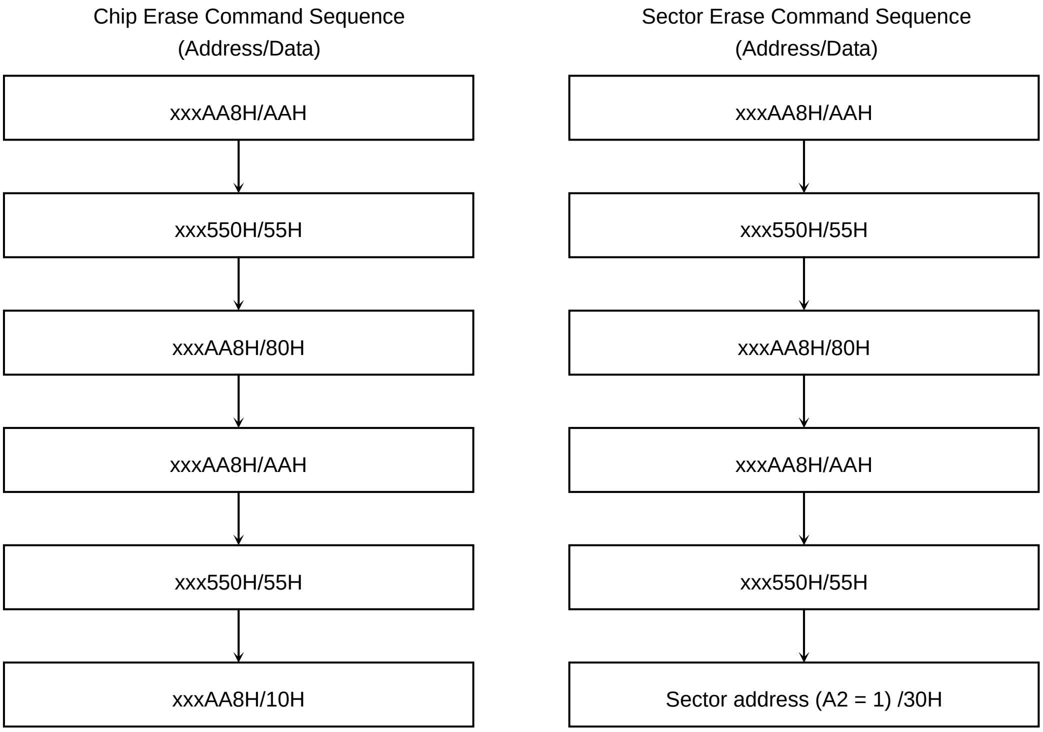



Program Command Sequence (Address/Data)
xxxAA8H/AAH
xxx550H/55H
xxxAA8H/A0H
Program address (A1 = A0 = 0)
/ Program data (long word units)

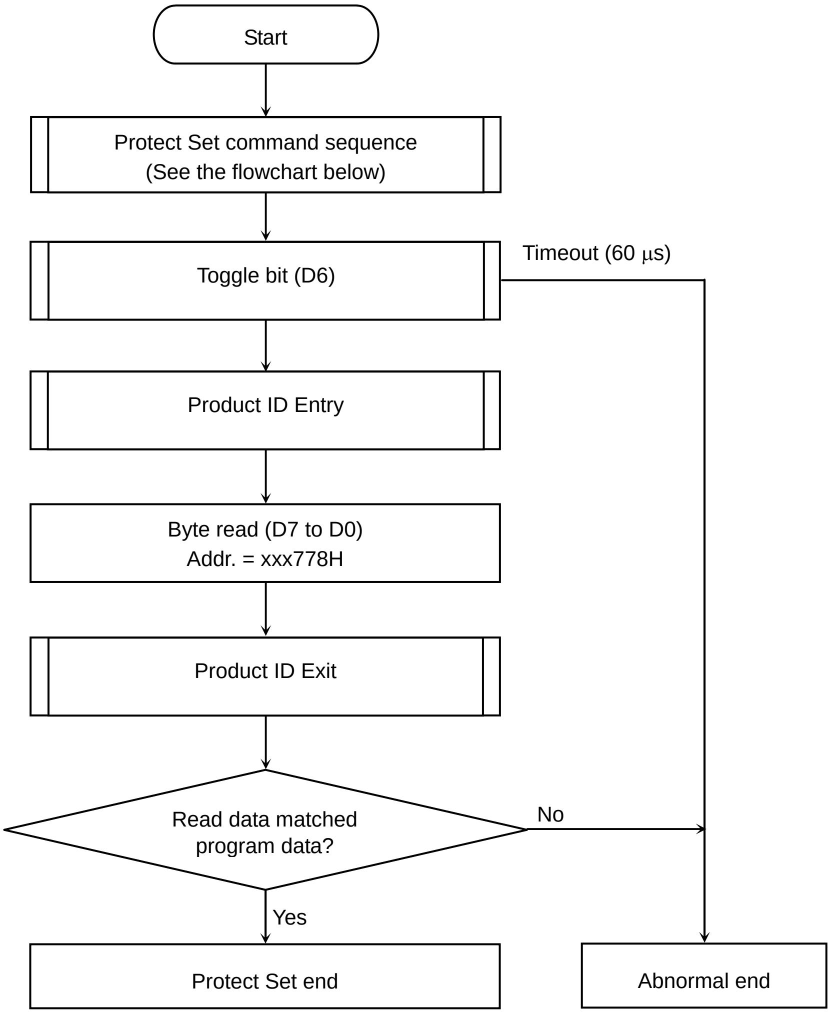
Chip Erase/Sector Erase



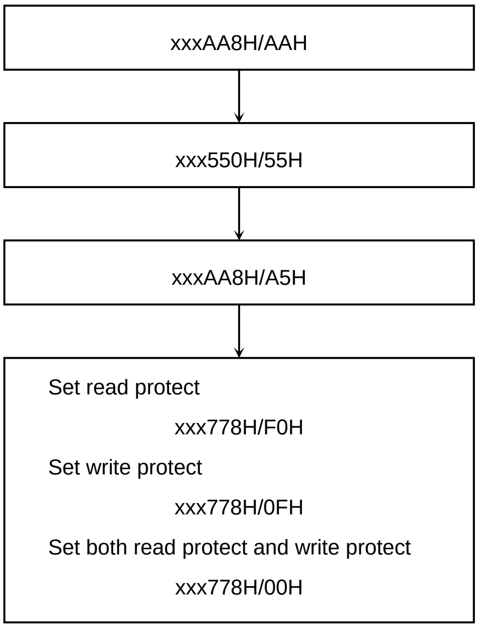
Note: In Chip Erase, whether or not the entire flash memory is blank is checked.
In Sector Erase, whether or not the selected sector is blank is checked.



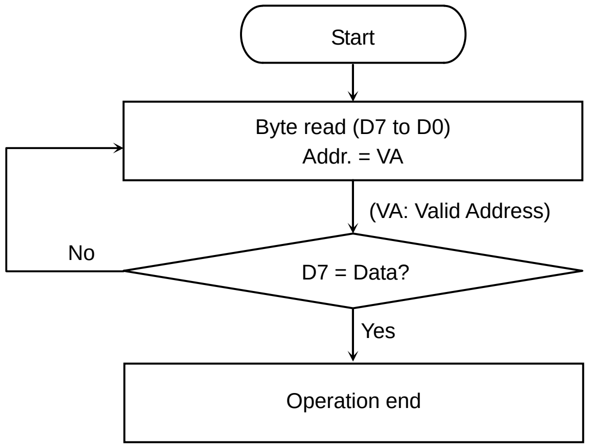
Read/Write Protect Set



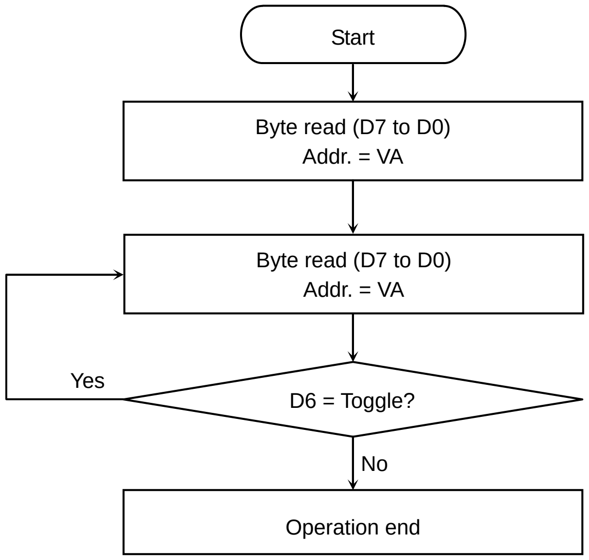
Protect Set Command Sequence
(Address/Data)



Data Polling (D7)



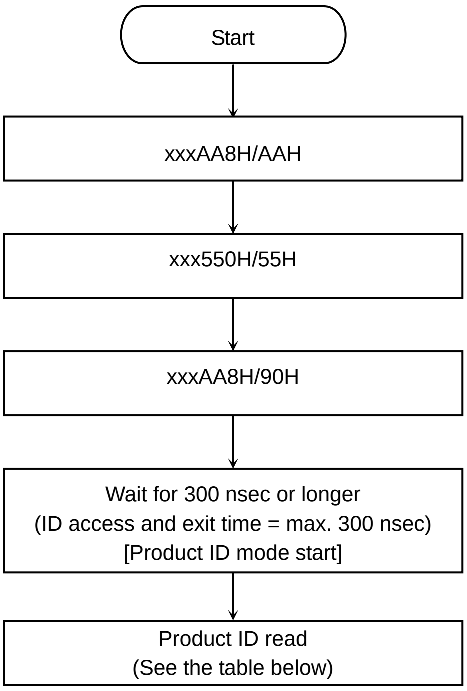
Toggle Bit (D6)



Note: Hardware sequence flags are read from the flash memory in byte units or word units or long word units.

VA: In Single Long Word Program, VA denotes the address to be programmed.
In Sector Erase, VA denotes any address in the selected sector.
In Chip Erase, VA denotes any address in the flash memory.
In Read Protect Set, VA denotes the protect set address (xxx778H).
In Write Protect Set, VA denotes the protect set address (xxx778H).

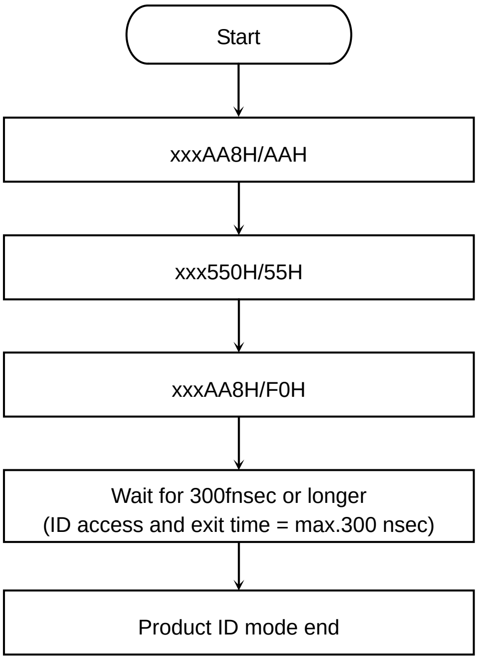
Product ID Entry

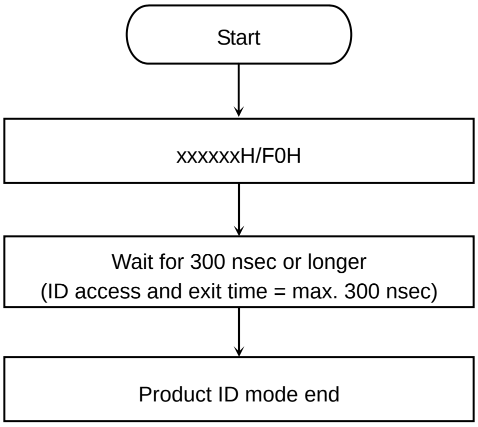


Read Values in Product ID Mode

	Address	Read Value
Vendor ID	xxxxF8H	xxxx_xx98H (D7 to D0)
Flash macro ID	xxxxF8H	xxxx_4BxxH (D15 to D8)
Flash size ID	xxxxF8H	xx7F_xxxxH (D23 to D16)
Read/Write Protect status	xxx778H	Data programmed when protection is set. When protection is not set, FFH. (D7 to D0)

Product ID Exit


Start
xxxAA8H/AAH
xxx550H/55H
xxxAA8H/F0H
Wait for 300fnsec or longer
(ID access and exit time = max.300 nsec)
Product ID mode end


Start
xxxxxxH/F0H
Wait for 300 nsec or longer
(ID access and exit time = max. 300 nsec)
Product ID mode end

(Example: Program to be loaded and executed in RAM)

Erase the flash memory (chip erase) and then write F047_0706H to address F80000H.

```
;#### Flash memory chip erase processing ####
        ld          XIX, 0xF80000                   ; set start address
CHIPERASE:
        ld          (0xF80AA8), 0xAA                ; 1st bus write cycle
        ld          (0xF80550), 0x55                ; 2nd bus write cycle
        ld          (0xF80AA8), 0x80                ; 3rd bus write cycle
        ld          (0xF80AA8), 0xAA                ; 4th bus write cycle
        ld          (0xF80550), 0x55                ; 5th bus write cycle
        ld          (0xF80AA8), 0x10                ; 6th bus write cycle

        cal         TOGGLECHK                       ; check toggle bit

CHIPERASE_LOOP:
        ld          XWA, (XIX+)                     ; read data from flash memory
        cp          XWA, 0xFFFFFFFF                 ; blank data?
        j           ne, CHIPERASE_ERR               ; if not blank data, jump to error processing
        cp          XIX, 0xFFFFFF                   ; end address (0xFFFFFF)?
        j           ULT, CHIPERASE_LOOP             ; check entire memory area and then end loop processing


;#### Flash memory program processing ####
        ld          XIX, 0xF80000                   ; set program address
        ld          XWA, 0xF0470706                 ; set program data
PROGRAM:
        ld          (0xF80AA8), 0xAA                ; 1st bus write cycle
        ld          (0xF80550), 0x55                ; 2nd bus write cycle
        ld          (0xF80AA8), 0xA0                ; 3rd bus write cycle
        ld          (XIX), XWA                      ; 4th bus write cycle

        cal         TOGGLECHK                       ; check toggle bit

        ld          XBC, (XIX)                      ; read data from flash memory
        cp          XWA, XBC.
        j           ne, PROGRAM_ERR                 ; if programmed data cannot be read, error is determined
        ld          XBC, (XIX)                      ; read data from flash memory
        cp          XWA, XBC
        j           ne, PROGRAM_ERR                 ; if programmed data cannot be read, error is determined

PROGRAM_END:
        j           PROGRAM_END                     ; program operation end


;#### Toggle bit (D6) check processing ####
TOGGLECHK:
        ld          L, (XIX)
        and         L, 0y01000000                   ; check toggle bit (D6)
        ld          H, L                            ; save first toggle bit data
TOGGLECHK1:
        ld          L, (XIX)
        and         L, 0y01000000                   ; check toggle bit (D6)
        cp          L, H                            ; toggle bit = toggled?
        j           z, TOGGLECHK2                   ; if not toggled, end processing
        ld          H, L                            ; save current toggle bit state
        j           TOGGLECHK1                      ; recheck toggle bit
TOGGLECHK2:
        ret


;####  Error processing ####
CHIPERASE_ERR:
        j           CHIPERASE_ERR                   ; chip erase error

PROGRAM_ERR:
        j           PROGRAM_ERR                     ; program error
```

(Example: Program to be loaded and executed in RAM)
Erase data at addresses F90000H to F90FFFH (sector erase) and then write F047_0706H to address F90000H.

```
;#### Flash memory sector erase processing ####
        ld      XIX, 0xF90004               ; set sector erase address (A2 = 1)
SECTORERASE:
        ld      (0xF80AA8), 0xAA            ; 1st bus write cycle
        ld      (0xF80550), 0x55            ; 2nd bus write cycle
        ld      (0xF80AA8), 0x80            ; 3rd bus write cycle
        ld      (0xF80AA8), 0xAA            ; 4th bus write cycle
        ld      (0xF80550), 0x55            ; 5th bus write cycle
        ld      (XIX), 0x30                 ; 6th bus write cycle

        cal     TOGGLECHK                   ; check toggle bit

        ld      XIX, 0xF90000               ; set start address
SECTORERASE_LOOP:
        ld      XWA, (XIX+)                 ; read data from flash memory
        cp      XWA, 0xFFFFFFFF             ; blank data?
        j       ne, SECTORERASE_ERR         ; if not blank data, jump to error processing
        cp      XIX, 0xF90FFF               ; end address (0xF90FFF)?
        j       ULT, SECTORERASE_LOOP       ; check erased sector area and then end loop processing


;#### Flash memory program processing ####
        ld      XIX, 0xF90000               ; set program address
        ld      XWA, 0xF0470706             ; set program data
PROGRAM:
        ld      (0xF80AA8), 0xAA            ; 1st bus write cycle
        ld      (0xF80550), 0x55            ; 2nd bus write cycle
        ld      (0xF80AA8), 0xA0            ; 3rd bus write cycle
        ld      (XIX), XWA                  ; 4th bus write cycle

        cal     TOGGLECHK                   ; check toggle bit

        ld      XBC, (XIX)                  ; read data from flash memory
        cp      XWA, XBC
        j       ne, PROGRAM_ERR             ; if programmed data cannot be read, error is determined
        ld      XBC, (XIX)                  ; read data from flash memory
        cp      XWA, XBC
        j       ne, PROGRAM_ERR             ; if programmed data cannot be read, error is determined

PROGRAM_END:
        j       PROGRAM_END                 ; program operation end


;#### Toggle bit (D6) check processing ####
TOGGLECHK:
        ld      L, (XIX)
        and     L, 0y01000000               ; check toggle bit (D6)
        ld      H, L                        ; save first toggle bit data
TOGGLECHK1:
        ld      L, (XIX)
        and     L, 0y01000000               ; check toggle bit (D6)
        cp      L, H                        ; toggle bit = toggled?
        j       z, TOGGLECHK2               ; If not toggled, end processing
        ld      H, L                        ; save current toggle bit state
        j       TOGGLECHK1                  ; Recheck toggle bit
TOGGLECHK2:
        ret


;#### Error processing ####
SECTORERASE_ERR:
        j       SECTORERASE_ERR             ;  sector erase error

PROGRAM_ERR:
        j       PROGRAM_ERR                 ; program error
```

(Example: Program to be loaded and executed in RAM)
Set read protection and write protection on the flash memory.

```
;#### Flash Memory Protect Set processing ####
        ld          XIX, 0xF80778                ; set protect address
PROTECT:
        ld          (0xF80AA8), 0xAA             ; 1st bus write cycle
        ld          (0xF80550), 0x55             ; 2nd bus write cycle
        ld          (0xF80AA8), 0xA5             ; 3rd bus write cycle
        ld          (XIX), 0x00                  ; 4th bus write cycle

        cal         TOGGLECHK                    ; check toggle bit
        cal         PID_ENTRY                    ;
        ld          A, (XIX)                     ; read protected address
        cal         PID_EXIT                     ;
        cp          A, 0x00                      ;(0xF80778)=0x00?
        j           ne, PROTECT_ERR              ; protected?

PROTECT_END:
        j           PROTECT_END                  ; protect set operation completed

PROTECT_ERR:
        j           PROTECT_ERR                  ; protect set error

;#### Product ID Entry processing ####
PID_ENTRY:
        ld          (0xF80AA8), 0xAA             ; 1st bus write cycle
        ld          (0xF80550), 0x55             ; 2nd bus write cycle
        ld          (0xF80AA8), 0x90             ; 3rd bus write cycle
        ; --- wait for 300 nsec or longer (execute NOP instruction [50nsec/@fFPH=40MHz] six times) ---
        nop
        nop
        nop
        nop
        nop
        nop                                      ; wait for 300 nsec
        ret

;#### Product ID Exit processing ####
PID_EXIT:
        ld          (0xF80000), 0xF0             ; 1st bus write cycle
        ; --- wait for 300 nsec or longer (execute NOP instruction [50nsec/@fFPH=40MHz] six times) ---
        nop
        nop
        nop
        nop
        nop
        nop                                      ; wait for 300 nsec
        ret

;####  Toggle bit (D6) check processing ####
TOGGLECHK:
        ld          L, (XIX)
        and         L, 0y01000000                ; check toggle bit (D6)
        ld          H, L                         ; save first toggle bit data
TOGGLECHK1:
        ld          L, (XIX)
        and         L, 0y01000000                ; check toggle bit (D6)
        cp          L, H                         ; toggle bit = toggled?
        j           z, TOGGLECHK2                ; if not toggled, end processing
        ld          H, L                         ; save current toggle bit state
        j           TOGGLECHK1                   ; recheck toggle bit
TOGGLECHK2:
        ret
```

(Example: Program to be loaded and executed in RAM)
Read data from address F80000H.

```
;#### Flash memory read processing ####
READ:
        ld          XWA, (0xF80000)              ; read data from flash memory
```

4. Electrical Characteristics

4.1 Absolute Maximum Ratings

Parameter	Symbol	Rating	Unit
Power Supply Voltage	V_{CC}	−0.5 to 4.0	V
Input Voltage	V_{IN}	−0.5 to VCC + 0.5	V
Output Current (1 pin) Except PN1, PN2, PN4 and PN5	I_{OL}	2	mA
Output Current (1 pin) PN1, PN2, PN4 and PN5	I_{OL2}	3.5	mA
Output Current (1 pin)	I_{OH}	−2	mA
Output Current (Total)	ΣI_{OL}	80	mA
Output Current (Total)	ΣI_{OH}	−80	mA
Power Dissipation (Ta = 85°C)	P_D	600	mW
Soldering Temperature (10 s)	T_{SOLDER}	260	°C
Storage Temperature	T_{STG}	−65 to 150	°C
Operation Temperature	T_{OPR}	−40 to 85	°C

Note: The absolute maximum ratings are rated values which must not be exceeded during operation, even for an instant. Any one of the ratings must not be exceeded. If any absolute maximum rating is exceeded, a device may break down or its performance may be degraded, causing it to catch fire or explode resulting in injury to the user. Thus, when designing products which include this device, ensure that no absolute maximum rating value will ever be exceeded.

Solderability of lead free products

Test parameter	Test condition	Note
Solderability	(1) Use of Sn-37Pb solder Bath Solder bath temperature =230°C, Dipping time = 5 seconds The number of times = one, Use of R-type flux	Pass: solderability rate until forming ≥ 95%
	(2) Use of Sn-3.0Ag-0.5Cu solder bath Solder bath temperature =245°C, Dipping time = 5 seconds The number of times = one, Use of R-type flux (use of lead free)	

4.2 DC Electrical Characteristics (1/2)

V_{CC} = 3.3 ± 0.3V/fc = 6 to 40 MHz/Ta = −40 to 85°C

Parameter	Symbol	Min	Typ.	Max	Unit	Condition
Power Supply Voltage (DVCC = AVCC) (DVSS = AVSS = 0V)	V_{CC}	3.0		3.6	V	X1=6 to 10MHz XT1=30 to 34KHz
Power Supply Voltage (DVCC = AVCC) (DVSS = AVSS = 0V) for erase/program operations of flash memory	V_{CC}	3.0		3.6	V	X1=6 to 10MHz Ta= −10 to 40°C
Input Low Voltage for P00 to P07 (D0~D7) P10 to P17 (D8~D15)	V_{IL0}	−0.3		0.6	V	
Input Low Voltage for P40 to P47 (A0 to A7) P50 to P57 (A8 to A15) P60 to P67 (A16 to A23) P76, P77 P80 to P82	V_{IL1}			0.3 × VCC		
Input Low Voltage for P70 to P73, P83 PC0 to PC3, PD0 to PD4 PF0 to PF5, PG0 to PG7 PL0 to PL3, PN0, PN3	V_{IL2}			0.25 × VCC		
$\overline{\text{RESET}}$, $\overline{\text{NMI}}$, P74(INT0)	V_{IL2a}			0.2 × VCC		
Input Low Voltage for AM0, AM1	V_{IL3}			0.3		
Input Low Voltage for X1, XT1(P76)	V_{IL4}			0.2 × VCC		
Input Low Voltage for PN1, PN2, PN4, PN5	V_{IL5}			0.3 × VCC		
Input High Voltage for P00 to P07 (D0 to D7) P10 to P17 (D8 to D15)	V_{IH0}	2.0		VCC + 0.3	V	
Input High Voltage for P40 to P47 (A0 to A7) P50 to P57 (A8 to A15) P60 to P67 (A16 to A23) P76, P77, P80 to P82	V_{IH1}	0.7 × VCC				
Input High Voltage for P70 to P73, P83 PC0 to PC3, PD0~PD4 PF0 to PF5, PG0~PG7 PL0 to PL3, PN0, PN3	V_{IH2}	0.75 × VCC				
$\overline{\text{RESET}}$, $\overline{\text{NMI}}$, P74(INT0)	V_{IH2a}	0.8 × VCC				
Input High Voltage for AM0, AM1	V_{IH3}	VCC − 0.3				
Input High Voltage for X1, XT1(P76)	V_{IH4}	0.8 × VCC				
Input High Voltage for PN1, PN2, PN4, PN5	V_{IH5}	0.7 × VCC		5.5		

V_{CC} = 3.3 ± 0.3V/fc = 6 to 40 MHz/Ta = –40 to 85°C

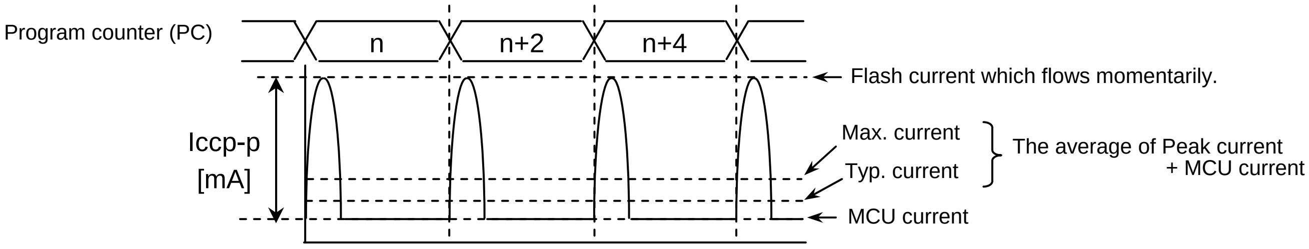
Parameter	Symbol	Min	Typ.	Max	Unit	Condition
Output Low Voltage	V_{OL}			0.45	V	IOL = 1.6 mA
Output Low Voltage for PN1, PN2, PN4, PN5	V_{OL2}			0.4		IOL = 3.0 mA
Output High Voltage	V_{OH}	2.4				IOH = –400 µA
Input Leakage Current	I_{LI}		0.02	±5	µA	0.0 ≦ Vin ≦ VCC
Output Leakage Current	I_{LO}		0.05	±10		0.2 ≦ Vin ≦ VCC − 0.2
Power Down Voltage at STOP (for STOP, RAM back-up)	V_{STOP}	1.8		3.6	V	VIL2 = 0.2 × Vcc, VIH2 = 0.8 × Vcc
Pull-Up Resistor for $\overline{RESET}$	R_{RST}	80		500	KΩ	
Programmable Pull-Up Resistor for P70 to P73	R_{KH}					
Pin Capacitance	C_{IO}			10	pF	fc = 1 MHz
Schmitt Width for P70 to P73, P83 PC0 to PC3, PD0 to PD4 PF0 to PF5, PG0 to PG7 PL0 to PL3, PN0 to PN5 $\overline{RESET}$, P74(INT0)	V_{TH}	0.2			V	
NORMAL (Note 2)	ICC		55	70	mA	f_C = 40 MHz f_{SYS} = 20 MHz
IDLE2 Mode	ICC_{IDLE2}		13	22		
IDLE1 Mode	ICC_{IDLE1}		4	9		
SLOW (Note 2)	ICC		75	120	µA	XT1 = 32.768 KHz (f_{SYS} = 16.384 KHz)
SLOW−IDLE2 Mode	ICC_{IDLE2}		20	90		
SLOW−IDLE1 Mode	ICC_{IDLE1}		10	80		
STOP	ICC_{STOP}		1.5	50		VCC =3.6V
Peak current by intermitt operation	Iccp-p		40		mA	VCC =3.0V~3.6V

Note 1: Typical values are for when Ta = 25°C and VCC = 3.3 V unless otherwise noted.

Note 2: ICC measurement conditions (NORMAL, SLOW):
All functions are operational; output pins are opened and input pins are fixed. CL = 30 pF is loaded to data and address bus.

When the program is operating by the flash memory, or when data reed from the flash memory, the flash memory operate intermittently. Therefore, it outputs a peak current like a following diagram, momentarily. In this case, the power supply current; Icc (NORMAL/SLOW mode) is the sum of average value of a peak current and a MCU current value.

When designing the power supply, set to a circuit which a peak current can be supplied. In SLOW mode, a deference of peak current and average current is large.


Program counter (PC)
n
n+2
n+4
Flash current which flows momentarily.
Iccp-p
[mA]
Max. current
Typ. current
The average of Peak current + MCU current
MCU current

Flash memory intermittent operation

4.3 AC Characteristics

4.3.1 Basic Bus Cycle

Read cycle

V_{CC} = 3.3 ± 0.3V/fc = 6 to 40 MHz/Ta = –40 to 85°C

No.	Parameter	Symbol	Variable		f_{SYS} = 20 MHz (fc = 40 MHz)	f_{SYS} = 13.5MHz (fc = 27 MHz)	Unit
			Min	Max			
1	OSC period (X1/X2)	t_{OSC}	25		25	37.0	ns
2	System clock period (= T)	t_{CYC}	50		50	74.0	ns
3	CLK Low Width	t_{CL}	0.5T – 15		10	22	ns
4	CLK High Width	t_{CH}	0.5T – 15		10	22	ns
5-1	A0 to A23 Valid→ D0 to D15 input at 0 WAIT	t_{AD}		2.0T – 50	50	98	ns
5-2	A0 to A23 Valid → D0 to D15 input at 1 WAIT	t_{AD3}		3.0T – 50	100	172	ns
6-1	$\overline{\text{RD}}$ Falling → D0 to D15 input at 0 WAIT	t_{RD}		1.5T – 45	30	66	ns
6-2	$\overline{\text{RD}}$ Rising → D0 to D15 input at 1 WAIT	t_{RD3}		2.5T – 45	80	140	ns
7-1	$\overline{\text{RD}}$ Low Width at 0 WAIT	t_{RR}	1.5T – 20		55	91	ns
7-2	$\overline{\text{RD}}$ Low Width at 1 WAIT	t_{RR3}	2.5T – 20		105	165	ns
8	A0 to A23 valid → $\overline{\text{RD}}$ Rising	t_{AR}	0.5T – 20		5	17	ns
9	$\overline{\text{RD}}$ Falling → CLK Falling	t_{RK}	0.5T – 20		5	17	ns
10	A0 to A23 valid → D0 to D15 Hold	t_{HA}	0		0	0	ns
11	$\overline{\text{RD}}$ Rising → D0 to D15 Hold	t_{HR}	0		0	0	ns
12	$\overline{\text{WAIT}}$ Set-up Time	t_{TK}	20		20	20	ns
13	$\overline{\text{WAIT}}$ Hold Time	t_{KT}	5		5	5	ns
14	Data Byte Control Access Time for SRAM	t_{SBA}		1.5T – 45	30	66	ns
15	$\overline{\text{RD}}$ High Width	t_{RRH}	0.5T – 15		10	22	ns

Write cycle

V_{CC} = 3.3 ± 0.3V/fc = 6 to 40 MHz/Ta = –40 to 85°C

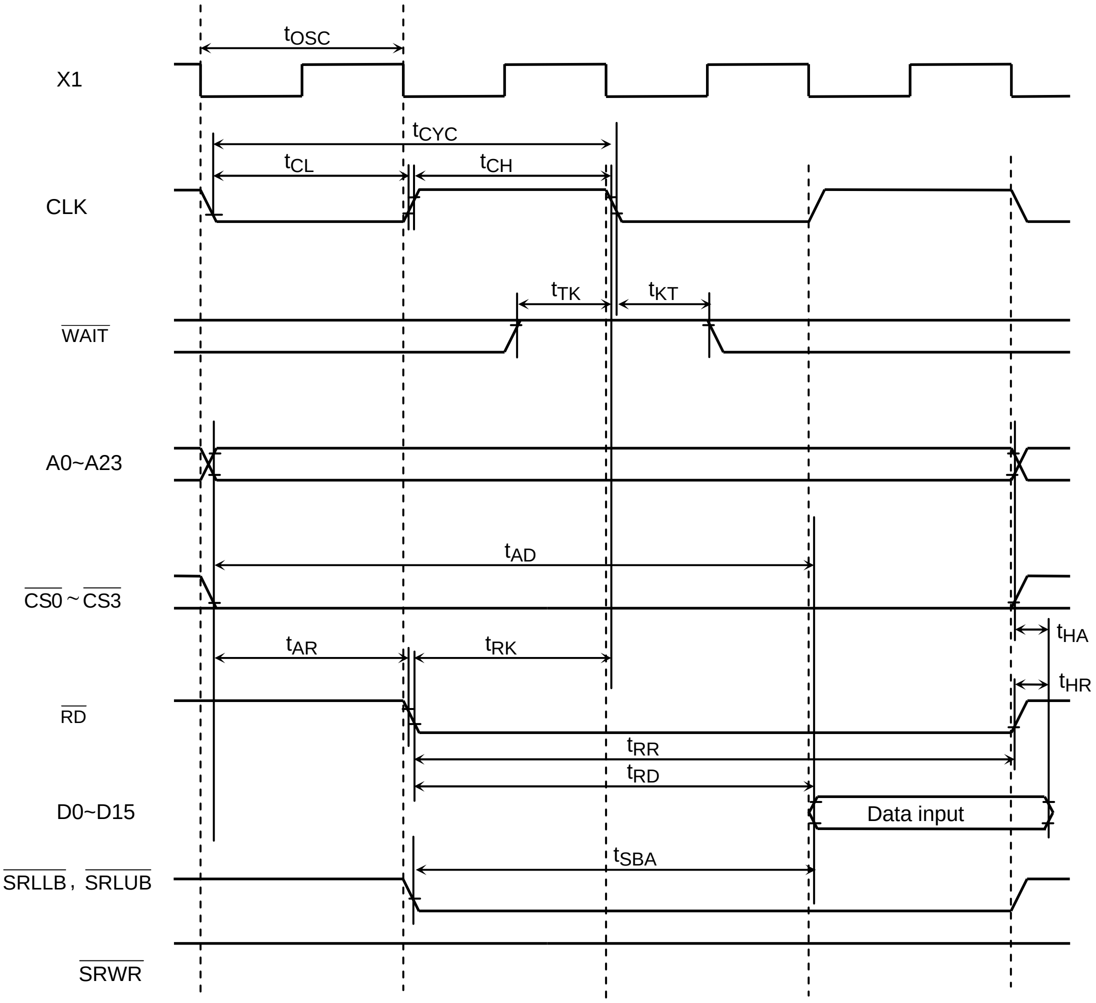
No.	Parameter	Symbol	Variable		f_{SYS} = 20 MHz (fc = 40 MHz)	f_{SYS} = 13.5MHz (fc = 27 MHz)	Unit
			Min	Max			
16	$\overline{\text{SRWR}}$ Falling → CLK Falling	t_{SWK}	0.5T – 20		5	17	ns
17	$\overline{\text{SRWR}}$ Rising → A0 to A23 Hold	t_{SWA}	0.25T – 5		7.5	13.5	ns
18	$\overline{\text{RD}}$ Rising → D0 to D15 Output	t_{RDO}	0.5T – 5		20	32	ns
19	Write Pulse Width for SRAM	t_{SWP}	1.25T – 30		32.5	62.5	ns
20	Data Byte Control to End of Write for SRAM	t_{SBW}	1.25T – 30		32.5	62.5	ns
21	Address Setup Time for SRAM	t_{SAS}	0.5T – 20		5	17	ns
22	Write Recovery Time for SRAM	t_{SWR}	0.25T – 5		7.5	13.5	ns
23	Data Setup Time for SRAM	t_{SDS}	1.25T – 35		27.5	57.5	ns
24	Data Hold Time for SRAM	t_{SDH}	0.25T – 5		7.5	13.5	ns

AC measuring condition

Output: High = 0.7 VCC, Low = 0.3 VCC, C_L = 50 pF

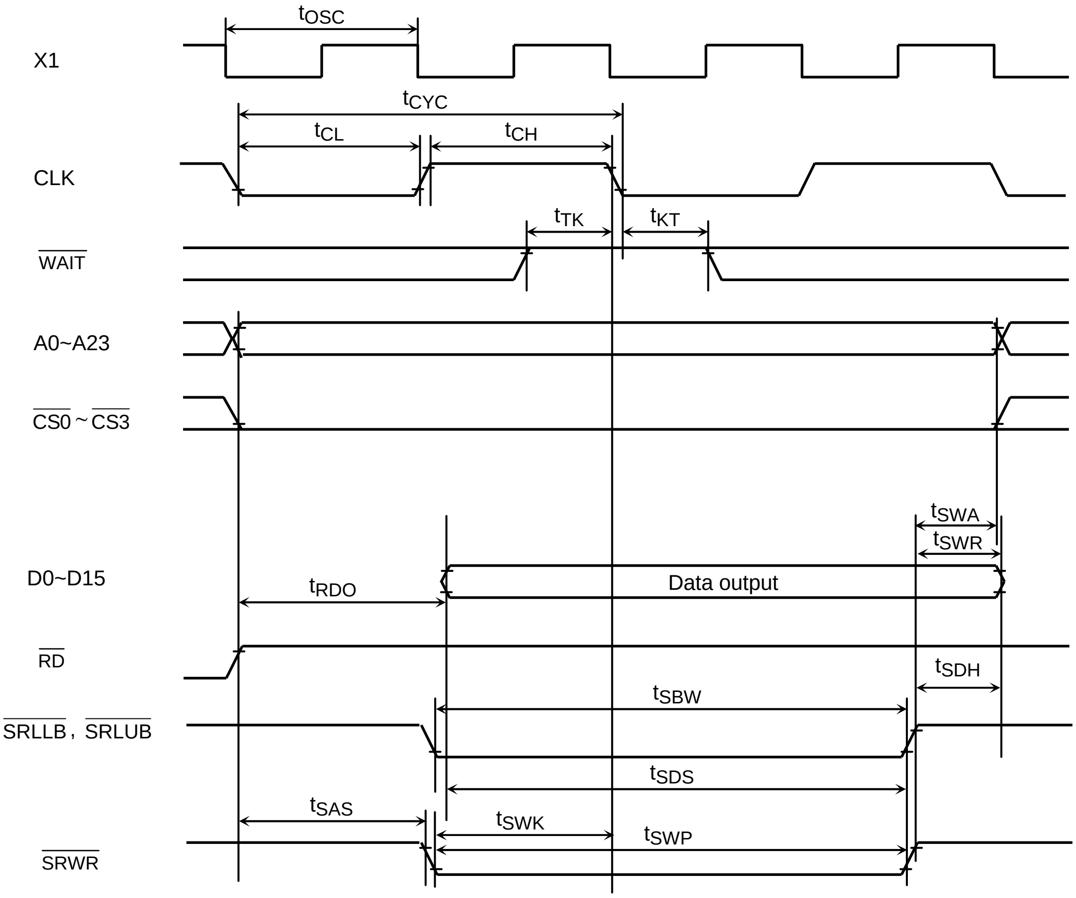
Input: High = 0.9 VCC, Low = 0.1 VCC

(1) Read cycle (0 waits, fc = f_{OSCH}, f_{FPH} = fc/1)



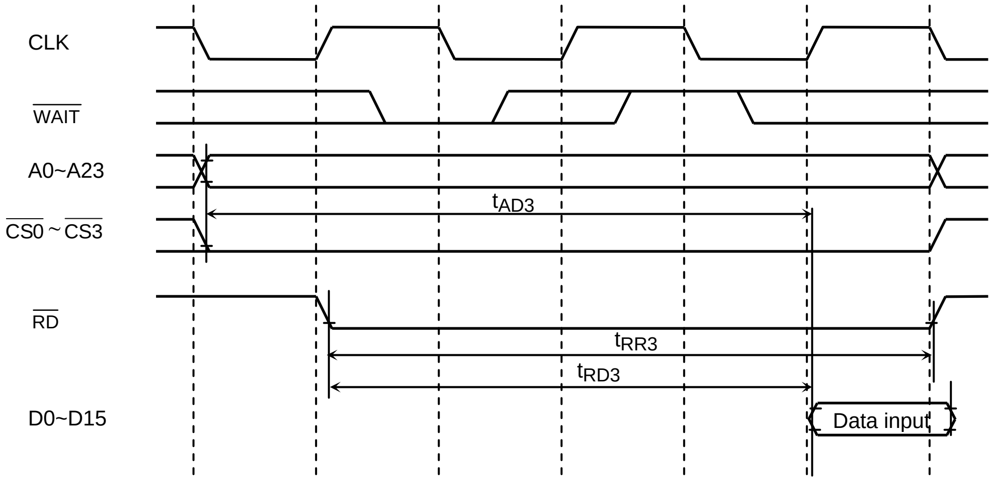
Note: The phase relation between X1 input signal and the other signals is undefined.
The above timing chart is an example.

(2) Write cycle (0 waits, fc = f_{OSCH}, f_{FPH} = fc/1)

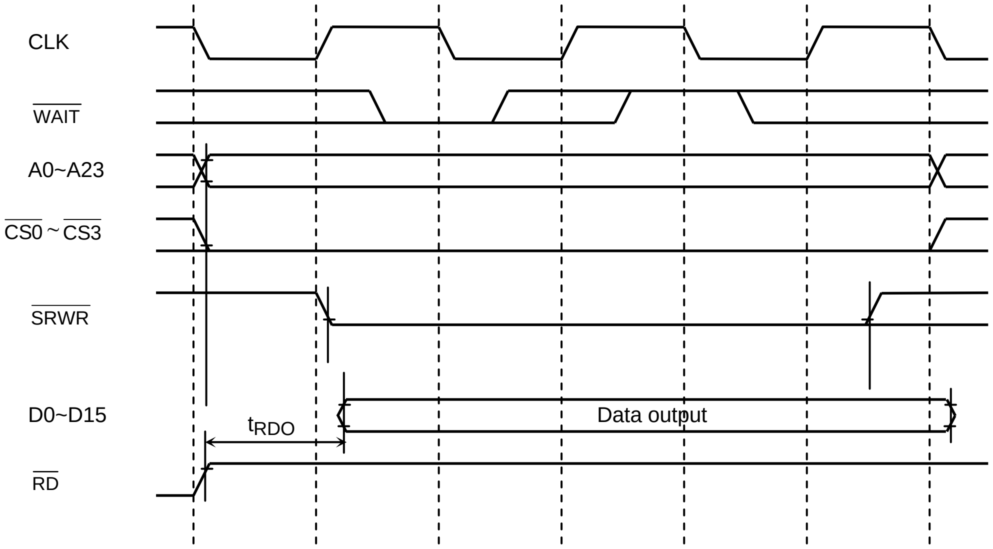


Note: The phase relation between X1 input signal and the other signals is undefined.
The above timing chart is an example.

(3) Read cycle (1 wait, fc = f_{OSCH}, f_{FPH} = fc/1)



(4) Write cycle (1 wait, fc = f_{OSCH}, f_{FPH} = fc/1)



4.3.2 Page ROM Read Cycle

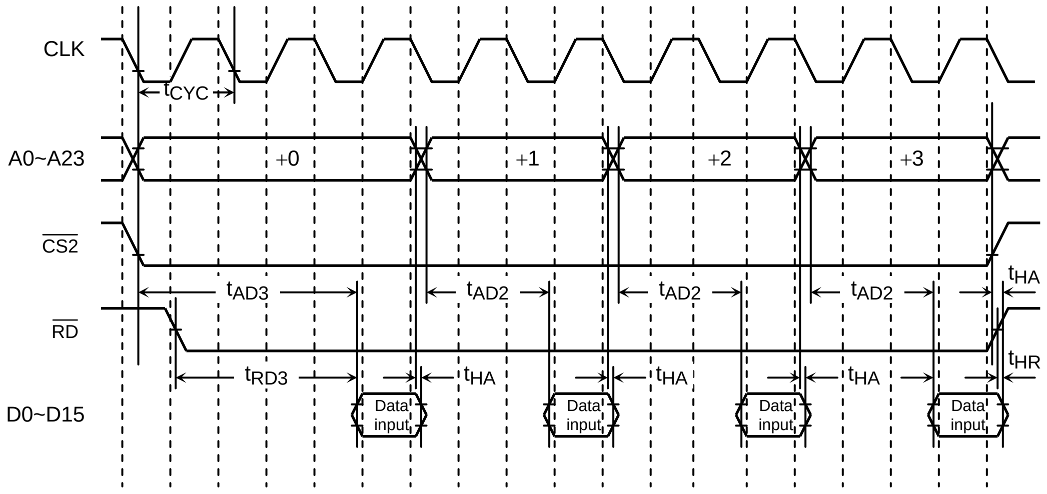
(1) 3-2-2-2 mode

Vcc = 3.3 ± 0.3 V/fc = 6~40 MHz/Ta = –40~85°C

No.	Parameter	Symbol	Variable		f_{SYS} = 20MHz (fc = 40 MHz)	f_{SYS} = 18MHz (fc = 36 MHz)	f_{SYS} = 13.5kHz (fc = 27 MHz)	Unit
			Min	Max				
1	System Clock Period (= T)	t_{CYC}	50		50	55.5	74	ns
2	A0, A1 → D0 to D15 input	t_{AD2}		2.0T – 50	50	61	98	ns
3	A2~A23 → D0 to D15 input	t_{AD3}		3.0T – 50	100	116.5	172	ns
4	$\overline{RD}$ Falling→ D0 to D15 input	t_{RD3}		2.5T – 45	80	93.8	140	ns
5	A0 to A23 valid → D0 to D15 Hold	t_{HA}	0		0	0	0	ns
6	$\overline{RD}$ Rising → D0 to D15 Hold	t_{HR}	0		0	0	0	ns

AC measuring condition

- Output: High = 0.7 VCC, Low = 0.3 VCC, CL = 50 pF
- Input: High = 0.9 VCC, Low = 0.1 VCC



Timing Pulse Diagram (8-byte setting)

4.3.3 Serial Channel Timing

(1) SCLK input mode (I/O interface mode)

Parameter	Symbol	Variable		f_{SYS} = 20 MHz (fc = 40 MHz)		f_{SYS} = 13.5MHz (fc = 27 MHz)		Unit
		Min	Max	Min	Max	Min	Max	
SCLK cycle	t_{SCY}	16X		0.40		0.59		μs
Output data → SCLK Rising/Falling *	t_{OSS}	$t_{SCY}/2 - 4X - 70$		30		78		ns
SCLK Rising/Falling* → Output Data Hold	t_{OHS}	$t_{SCY}/2 + 2X + 0$		250		370		ns
SCLK Rising/Falling* → Input Data Hold	t_{HSR}	3X + 10		85		121		ns
SCLK Rising/Falling* → Input Data Valid	t_{SRD}		$t_{SCY} - 0$		400		592	ns
Input Data Valid → SCLK Rising/Falling*	t_{RDS}	0		0		0		ns

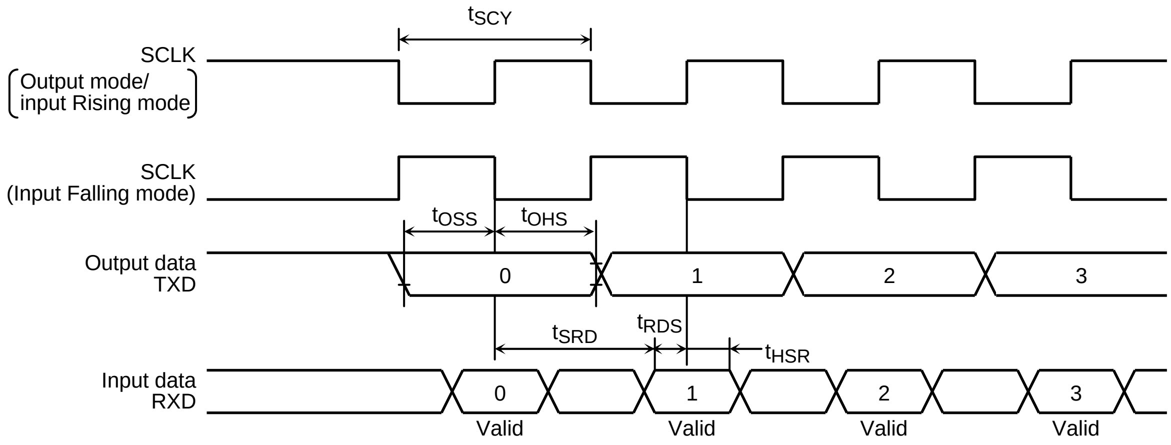
(2) SCLK output mode (I/O Interface mode)

Parameter	Symbol	Variable		f_{SYS} = 20 MHz (fc = 40 MHz)		f_{SYS} = 13.5MHz (fc = 27 MHz)		Unit
		Min	Max	Min	Max	Min	Max	
SCLK cycle	t_{SCY}	16X	8192X	0.40	204	0.59	303	μs
Output data → SCLK Rising/Falling *	t_{OSS}	$t_{SCY}/2 - 40$		160		256		ns
SCLK Rising/Falling* → Output Data Hold	t_{OHS}	$t_{SCY}/2 - 40$		160		256		ns
SCLK Rising/Falling* → Input Data Hold	t_{HSR}	0		0		0		ns
SCLK Rising/Falling* → Input Data Valid	t_{SRD}		$t_{SCY} - 1X - 180$		195		375	ns
Input Data Valid → SCLK Rising/Falling*	t_{RDS}	1X + 180		205		217		ns

*: SCLK rinsing/falling edge: The rising edge is used in SCLK rising mode.
The falling edge is used in SCLK falling mode.

Note 1: t_{SCY} = 16X at f_{SYS} = 20MHz or 13.5MHz

Note 2: 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.

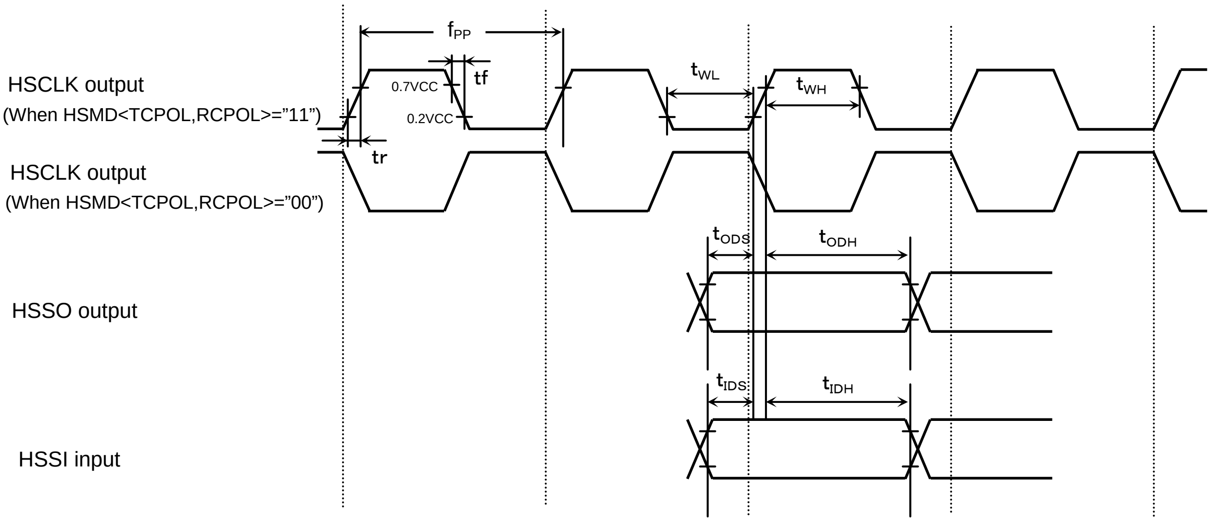


4.3.4 High Speed SIO Timing

Symbol	Parameter	Variable		f_{SYS} = 20MHz (fc = 40 MHz)	f_{SYS} = 18MHz (fc = 36 MHz)	f_{SYS} = 13.5MHz (fc = 27 MHz)	Unit
		Min	Max				
f_{PP}	HSCLK frequency (= 1/X)		10	10	9	6.75	MHz
t_r	HSCLK rising timing		8	8	8	8	ns
t_f	HSCLK falling time		8	8	8	8	
t_{WL}	HSCLK Low pulse width	0.5X-8		42	47	66	
t_{WH}	HSCLK High pulse width	0.5X-16		34	39	58	
t_{ODS1}	Output data valid → HSCLK rise	0.5X-18		32	37	56	
t_{ODS2}	Output data valid → HSCLK fall	0.5X-23		27	32	51	
t_{ODH}	HSCLK rise/fall → Output data hold	0.5X-10		40	45	64	
t_{IDS}	Input data valid → HSCLK rise/fall	0X+20		20	20	20	
t_{IDH}	HSCLK rise/fall → Input data hold	0X+5		5	5	5	

AC measuring conditions

Output level : High = 0.7 VCC, Low = 0.2 VCC, CL = 25 pF
Input level : High = 0.9 VCC, Low = 0.1 VCC



4.3.5 Interrupts

Parameter	Symbol	Variable		$f_{SYS} = 20$ MHz (fc = 40 MHz)		$f_{SYS} = 13.5$MHz (fc = 27 MHz)		Unit
		MIN	MAX	MIN	MAX	MIN	MAX	
$\overline{\text{NMI}}$, INT0 to INT7 Low level Width	T_{INTAL}	4X + 40		140		188		ns
$\overline{\text{NMI}}$, INT0~INT7 High level Width	T_{INTAH}	4X + 40		140		188		

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.

4.3.6 Event Counter (TA0IN, TB1IN0, TB1IN1)

Parameter	Symbol	Variable		$f_{SYS} = 20$ MHz (fc = 40 MHz)		$f_{SYS} = 13.5$MHz (fc = 27 MHz)		Unit
		MIN	MAX	MIN	MAX	MIN	MAX	
Clock period	T_{VCK}	8X + 100		300		396		ns
Clock Low level Width	T_{VCKL}	4X + 40		140		188		ns
Clock High level Width	T_{VCKH}	4X + 40		140		188		ns

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.

4.4 AD Conversion Characteristics

Parameter	Symbol	Min	Typ.	Max	Unit
AD Converter Power Supply Voltage	A_{VCC}	VCC	VCC	VCC	V
AD Converter GND	A_{VSS}	VSS	VSS	VSS	
Analog Input Voltage	A_{VIN}	A_{VSS}		A_{VCC}	
Total error (Quantize error of ± 0.5LSB is included)	E_T		±1.0	±4.0	LSB

Note 1: 1LSB = (VREFH − VREFL) / 1024 [V]

Note 2: Minimum frequency for operation

AD converter operatinon is generated only using fc (high-frequency oscillator). fs is not guaranteed. However, if clock frequency which is selected by clock is over than 4MHz, operation is guaranteed.

Note 3: The value for Icc includes the current which flows through the AV_{CC} pin.

4.5 Flash Characteristics

(1) Rewriting

Parameter	Condition	Min	Typ	Max	Unit
Gurantee on Flash-memory rewriting	Vcc = 3.0V to 3.6V X1 = 6MHz to 10MHz Ta = -10 to 40℃	–	–	100	Times

4.6 Recommended Oscillation Circuit

The TMP92FD23A has been evaluated by the oscillator vender below. Use this information when selecting external parts.

Note: The total load value of the oscillator is the sum of external loads (C1 and C2) and the floating load of the actual assembled board. There is a possibility of operating error when using C1 and C2 values in the table below. When designing the board, design the minimum length pattern around the oscillator. We also recommend that oscillator evaluation be carried out using the actual board.

(1) Connection example

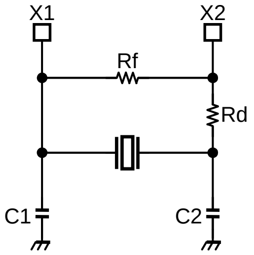


Figure 4.6.1 High-frequency oscillator

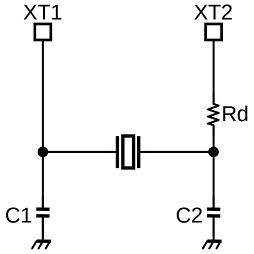


Figure 4.6.2 Low-frequency oscillator

(2) Recommended ceramic oscillator: Murata Manufacturing Co., Ltd.

MCU	Oscillation Frequency [MHZ]	Oscillator Product Number	Item of Oscillator	Parameter of elements				Running Condition	
				C1 [pF]	C2 [pF]	Rf [Ω]	Rd [Ω]	Voltage of Power [V]	Ta [°C]
TMP92CY23/ TMP92FD23A	6.00	SMD	CSTCR6M00G55-R0	(39)	(39)	Open	0	3.0~3.6	−20~80
		Lead	CSTLS6M00G56-B0	(47)	(47)				
	10.00	SMD	CSTCE10M0G55-R0	(33)	(33)				
		Lead	CSTLS10M0G56-B0	(47)	(47)				

Note 1: The figure in parentheses ( ) under C1 and C2 is the built-in condenser type.In CST **type oscillator, capacitance C1 and C2 is built-in.

Note 2: The product numbers and specifications of the oscillators made by Murata Manufacturing Co., Ltd. are subject to change. For up-to-date information, please refer to the following URL:
http:// www.murata.co.jp/

5. Table of Special function registers (SFRs)

The SFRs include the I/O ports and peripheral control registers allocated to the 8-Kbyte address space from 000000H to 001FFFH.

(1) I/O Port

(2) Interrupt control

(3) DMA controller

(4) Memory controller

(5) Clock control/PLL

(6) 8-bit timer

(7) 16-bit timer

(8) High speed serial channel

(9) UART/serial channel

(10) I^{2}CBUS/serial channel

(11) AD converter

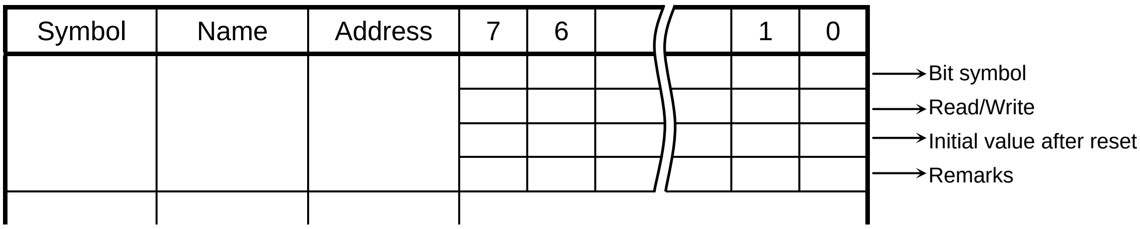
(12) Watchdog timer

(13) Special timer for CLOCK

(14) Key-on wake up

(15) Program patch function

Table layout



Note: "Prohibit RMW" in the table means that you cannot use RMW instructions on these registers.

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 is prohibited when controlling the pull-up resistor.

Table 5.1 I/O Register Address Map

[1] Port

Address	Name
0000H	P0
1H	
2H	P0CR
3H	P0FC
4H	P1
5H	
6H	P1CR
7H	P1FC
8H	
9H	
AH	
BH	
CH	
DH	
EH	
FH	

Address	Name
0010H	P4
1H	
2H	P4CR
3H	P4FC
4H	P5
5H	
6H	P5CR
7H	P5FC
8H	P6
9H	
AH	P6CR
BH	P6FC
CH	P7
DH	
EH	P7CR
FH	P7FC

Address	Name
0020H	P8
1H	P8FC2
2H	P8CR
3H	P8FC
4H	
5H	
6H	
7H	
8H	
9H	
AH	
BH	
CH	
DH	
EH	
FH	

Address	Name
0030H	PC
1H	
2H	PCCR
3H	PCFC
4H	PD
5H	PDFC2
6H	PDCR
7H	PDFC
8H	
9H	
AH	
BH	
CH	PF
DH	PFFC2
EH	PFCR
FH	PFFC

Address	Name
0040H	PG
1H	
2H	
3H	PGFC
4H	
5H	
6H	
7H	
8H	
9H	
AH	
BH	
CH	
DH	
EH	
FH	

Address	Name
0050H	
1H	
2H	
3H	
4H	PL
5H	
6H	
7H	PLFC
8H	
9H	
AH	
BH	
CH	PN
DH	
EH	PNCR
FH	PNFC

Note: Do not access no allocated name address.

[2] INTC

Address	Name
00D0H	INTE01
1H	INTE23
2H	INTE45
3H	INTE67
4H	INTETA01
5H	INTETA23
6H	INTETA45
7H	Reserved
8H	INTES0
9H	INTES1HSC
AH	INTES2
BH	Reserved
CH	INTESB0
DH	INTESB1
EH	Reserved
FH	Reserved

Address	Name
00E0H	INTETB0
1H	INTESTBO0
2H	INTETB1
3H	INTSTBO1
4H	INTEPAD
5H	INTERTC
6H	
7H	
8H	
9H	
AH	
BH	
CH	
DH	
EH	
FH	INTENMWDT

Address	Name
00F0H	INTTC01
1H	INTTC23
2H	INTTC45
3H	INTTC67
4H	HSCSEL
5H	SIMC
6H	IIMC
7H	
8H	INTCLR
9H	Reserved
AH	IIMC2
BH	IIMC3
CH	Reserved
DH	Reserved
EH	Reserved
FH	Reserved

[3] DMA controller

Address	Name
0100H	DMA0V
1H	DMA1V
2H	DMA2V
3H	DMA3V
4H	DMA4V
5H	DMA5V
6H	DMA6V
7H	DMA7V
8H	DMAB
9H	DMAR
AH	Reserved
BH	
CH	
DH	
EH	
FH	

[4] Memory controller

Address	Name
0140H	B0CSL
1H	B0CSH
2H	MAMR0
3H	MSAR0
4H	B1CSL
5H	B1CSH
6H	MAMR1
7H	MSAR1
8H	B2CSL
9H	B2CSH
AH	MAMR2
BH	MSAR2
CH	B3CSL
DH	B3CSH
EH	MAMR3
FH	MSAR3

Address	Name
0150H	Reserved
1H	Reserved
2H	Reserved
3H	Reserved
4H	Reserved
5H	Reserved
6H	Reserved
7H	Reserved
8H	BEXCSL
9H	BEXCSH
AH	Reserved
BH	Reserved
CH	
DH	
EH	
FH	

Address	Name
0160H	Reserved
1H	Reserved
2H	Reserved
3H	
4H	
5H	
6H	PMEMCR
7H	
8H	
9H	
AH	
BH	
CH	Reserved
DH	
EH	
FH	

[5] Clock control/PLL

Address	Name
10E0H	SYSCR0
1H	SYSCR1
2H	SYSCR2
3H	EMCCR0
4H	EMCCR1
5H	EMCCR2
6H	
7H	
8H	PLLCR0
9H	PLLCR1
AH	
BH	
CH	
DH	
EH	
FH	

Note: Do not access no allocated name address.

[6] 8-bit timer

Address	Name
1100H	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	Name
1110H	TA45RUN
1H	
2H	TA4REG
3H	TA5REG
4H	TA45MOD
5H	TA5FFCR
6H	
7H	
8H	
9H	
AH	
BH	
CH	
DH	
EH	
FH	

[7] 16-bit timer

Address	Name
1180H	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

Address	Name
1190H	TB1RUN
1H	
2H	TB1MOD
3H	TB1FFCR
4H	
5H	
6H	
7H	
8H	TB1RG0L
9H	TB1RG0H
AH	TB1RG1L
BH	TB1RG1H
CH	TB1CP0L
DH	TB1CP0H
EH	TB1CP1L
FH	TB1CP1H

[8] High speed serial

Address	Name
0C00H	HSC0MD
1H	HSC0MD
2H	HSC0CT
3H	HSC0CT
4H	HSC0ST
5H	HSC0ST
6H	HSC0CR
7H	HSC0CR
8H	HSC0IS
9H	HSC0IS
AH	HSC0WE
BH	HSC0WE
CH	HSC0IE
DH	HSC0IE
EH	HSC0IR
FH	HSC0IR

Address	Name
0C10H	HSC0TD
1H	HSC0TD
2H	HSC0RD
3H	HSC0RD
4H	HSC0TS
5H	HSC0TS
6H	HSC0RS
7H	HSC0RS
8H	
9H	
AH	
BH	
CH	
DH	
EH	
FH	

[8] UART/SIO

Address	Name
1200H	SC0BUF
1H	SC0CR
2H	SC0MOD0
3H	BR0CR
4H	BR0ADD
5H	SC0MOD1
6H	
7H	SIR0CR
8H	SC1BUF
9H	SC1CR
AH	SC1MOD0
BH	BR1CR
CH	BR1ADD
DH	SC1MOD1
EH	
FH	SIR1CR

Address	Name
1210H	SC2BUF
1H	SC2CR
2H	SC2MOD0
3H	BR2CR
4H	BR2ADD
5H	SC2MOD1
6H	
7H	SIR2CR
8H	
9H	
AH	
BH	
CH	
DH	
EH	
FH	

Note: Do not access no allocated name address.

[9] I^{2}C bus/SIO

Address	Name
1240H	SBI0CR1
1H	SBI0DBR
2H	I2C0AR
3H	SBI0CR2/SBI0SR
4H	SBI0BR0
5H	SBI0BR1
6H	
7H	
8H	SBI1CR1
9H	SBI1DBR
AH	I2C1AR
BH	SBI1CR2/SBI1SR
CH	SBI1BR0
DH	SBI1BR1
EH	
FH	

[10] AD converter

Address	Name
12A0H	ADREG0L
1H	ADREG0H
2H	ADREG1L
3H	ADREG1H
4H	ADREG2L
5H	ADREG2H
6H	ADREG3L
7H	ADREG3H
8H	ADREG4L
9H	ADREG4H
AH	ADREG5L
BH	ADREG5H
CH	ADREG6L
DH	ADREG6H
EH	ADREG7L
FH	ADREG7H

Address	Name
12B0H	ADREG8L
1H	ADREG8H
2H	ADREG9L
3H	ADREG9H
4H	ADREGAL
5H	ADREGAH
6H	ADREGBL
7H	ADREGBH
8H	ADMOD0
9H	ADMOD1
AH	ADMOD2
BH	Reserved
CH	Reserved
DH	
EH	
FH	

[11] Watch dog timer

Address	Name
1300H	WDMOD
1H	WDCR
2H	
3H	
4H	
5H	
6H	
7H	
8H	
9H	
AH	
BH	
CH	
DH	
EH	
FH	

[12] Special timer for CLOCK

Address	Name
1310H	RTCCR
1H	
2H	
3H	
4H	
5H	
6H	
7H	
8H	
9H	
AH	
BH	
CH	
DH	
EH	
FH	

[13] Key-on wake up

Address	Name
13A0H	KIEN
1H	KICR
2H	
3H	
4H	
5H	
6H	
7H	
8H	
9H	
AH	
BH	
CH	
DH	
EH	
FH	

Note: Do not access no allocated name address.

[14] Program patch function

Address	Name
1400H	ROMCMP00
1H	ROMCMP01
2H	ROMCMP02
3H	
4H	ROMSUB0LL
5H	ROMSUB0LH
6H	ROMSUB0HL
7H	ROMSUB0HH
8H	ROMCMP10
9H	ROMCMP11
AH	ROMCMP12
BH	
CH	ROMSUB1LL
DH	ROMSUB1LH
EH	ROMSUB1HL
FH	ROMSUB1HH

Address	Name
1410H	ROMCMP20
1H	ROMCMP21
2H	ROMCMP22
3H	
4H	ROMSUB2LL
5H	ROMSUB2LH
6H	ROMSUB2HL
7H	ROMSUB2HH
8H	ROMCMP30
9H	ROMCMP31
AH	ROMCMP32
BH	
CH	ROMSUB3LL
DH	ROMSUB3LH
EH	ROMSUB3HL
FH	ROMSUB3HH

Address	Name
1420H	ROMCMP40
1H	ROMCMP41
2H	ROMCMP42
3H	
4H	ROMSUB4LL
5H	ROMSUB4LH
6H	ROMSUB4HL
7H	ROMSUB4HH
8H	ROMCMP50
9H	ROMCMP51
AH	ROMCMP52
BH	
CH	ROMSUB5LL
DH	ROMSUB5LH
EH	ROMSUB5HL
FH	ROMSUB5HH

Address	Name
1430H	ROMCMP60
1H	ROMCMP61
2H	ROMCMP62
3H	
4H	ROMSUB6LL
5H	ROMSUB6LH
6H	ROMSUB6HL
7H	ROMSUB6HH
8H	ROMCMP70
9H	ROMCMP71
AH	ROMCMP72
BH	
CH	ROMSUB7LL
DH	ROMSUB7LH
EH	ROMSUB7HL
FH	ROMSUB7HH

Note: Do not access no allocated name address.

(1) I/O ports (1/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
P0	Port 0	0000H	P07	P06	P05	P04	P03	P02	P01	P00
			R/W							
			Data from external port (Output latch register is cleared to “0”)							
P1	Port 1	0004H	P17	P16	P15	P14	P13	P12	P11	P10
			R/W							
			Data from external port (Output latch register is cleared to “0”)							
P4	Port 4	0010H	P47	P46	P45	P44	P43	P42	P41	P40
			R/W							
			Data from external port (Output latch register is cleared to “0”)							
P5	Port 5	0014H	P57	P56	P55	P54	P53	P52	P51	P50
			R/W							
			Data from external port (Output latch register is cleared to “0”)							
P6	Port 6	0018H	P67	P66	P65	P64	P63	P62	P61	P60
			R/W							
			Data from external port (Output latch register is cleared to “0”)							
P7	Port 7	001CH	P77	P76		P74	P73	P72	P71	P70
			R/W			R	R/W			
			Data from external port (Output latch register is set to “1”)			Data from external port	Data from external port (Output latch register is set to “1”)			
			–			–	0 (Output latch register): Pull-up resistor OFF 1 (Output latch register): Pull-up resistor ON			
P8	Port 8	0020H					P83	P82	P81	P80
							R/W			
							Data from external port (Output latch register is set to “1”)	0	1	1
PC	Port C	0030H					PC3	PC2	PC1	PC0
							R			
							Data from external port			
PD	Port D	0034H				PD4	PD3	PD2	PD1	PD0
						R/W			R	R/W
						Data from external port (Note 1)			Data from external port	Data from external port (Note 1)
PF	Port F	003CH			PF5	PF4	PF3	PF2	PF1	PF0
					R/W					
					Data from external port (Output latch register is cleared to “0”)					
PG	Port G	0040H	PG7	PG6	PG5	PG4	PG3	PG2	PG1	PG0
			R							
			Data from external port (Note 2)							
PL	Port L	0054H					PL3	PL2	PL1	PL0
							R			
							Data from external port (Note 2)			
PN	Port N	005CH			PN5	PN4	PN3	PN2	PN1	PN0
					R/W					
					Data from external port (Output latch register is set to “1”)					

Note1: Output latch register is cleared to “0”. (There is no output latch register.)

Note2: It operates as an analog input port.(Input port disable)

I/O ports (2/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
P0CR	Port 0 Control register	0002H (Prohibit RMW)	P07C	P06C	P05C	P04C	P03C	P02C	P01C	P00C
			W							
			0	0	0	0	0	0	0	0
			0: Input 1: Output							
P0FC	Port 0 Function register	0003H (Prohibit RMW)								P00F
										W
										0
										0:Port 1:Data bus (D0 to D7)
P1CR	Port 1 Control register	0006H (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 register	0007H (Prohibit RMW)								P10F
										W
										0
										0:Port 1:Data bus (D8 to D15)
P4CR	Port 4 Control register	0012H (Prohibit RMW)	P47C	P46C	P45C	P44C	P43C	P42C	P41C	P40C
			W							
			0	0	0	0	0	0	0	0
			0: Input 1: Output							
P4FC	Port 4 Function register	0013H (Prohibit RMW)	P47F	P46F	P45F	P44F	P43F	P42F	P41F	P40F
			W							
			0	0	0	0	0	0	0	0
			0: Port 1: Address bus (A0 to A7)							
P5CR	Port 5 Control register	0016H (Prohibit RMW)	P57C	P56C	P55C	P54C	P53C	P52C	P51C	P50C
			W							
			0	0	0	0	0	0	0	0
			0: Input 1: Output							
P5FC	Port 5 Function register	0017H (Prohibit RMW)	P57F	P56F	P55F	P54F	P53F	P52F	P51F	P50F
			W							
			0	0	0	0	0	0	0	0
			0: Port 1: Address bus (A8 to A15)							
P6CR	Port 6 Control register	001AH (Prohibit RMW)	P67C	P66C	P65C	P64C	P63C	P62C	P61C	P60C
			W							
			0	0	0	0	0	0	0	0
			0: Input 1: Output							
P6FC	Port 6 Function register	001BH (Prohibit RMW)	P67F	P66F	P65F	P64F	P63F	P62F	P61F	P60F
			W							
			0	0	0	0	0	0	0	0
			0: Port 1: Address bus (A16 to A23)							

I/O ports (3/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
P7CR	Port 7 Control register	001EH (Prohibit RMW)	P77C	P76C			P73C	P72C	P71C	P70C
			W				W			
			1	1			0	0	0	0
			0: Input 1: Output				0: Input 1: Output			
P7FC	Port 7 Function register	001FH (Prohibit RMW)				P74F	P73F	P72F	P71F	P70F
						W				
						0	0	0	0	0
						0: Port input 1: INT0 input	0: Port 1: $\overline{\text{SRLUB}}$	0: Port 1: $\overline{\text{SRLLB}}$	0: Port 1: $\overline{\text{SRWR}}$	0: Port 1: $\overline{\text{RD}}$
P8FC2	Port 8 Function register 2	0021H (Prohibit RMW)					P83F2		P81F2	P80F2
							W		W	
							0		0	0
							0: <P83F> 1: TA5OUT		0: <P81F> 1: TA3OUT	0: <P80F> 1: TA1OUT
P8CR	Port 8 Control register	0022H (Prohibit RMW)					P83C			
							W			
							1			
							0: Input 1: Output			
P8FC	Port 8 Function register	0023H (Prohibit RMW)					P83F	P82F	P81F	P80F
							W			
							0	0	0	0
							<P83F,P83C> 00: Port input 01: Port output 10: $\overline{\text{WAIT}}$ input 11: $\overline{\text{CS3}}$ output	0: Port 1: $\overline{\text{CS2}}$	0: Port 1: $\overline{\text{CS1}}$	0: Port 1: $\overline{\text{CS0}}$
PCFC	Port C Function register	0033H (Prohibit RMW)					PC3F	PC2F	PC1F	PC0F
							W			
							0	0	0	0
							0: Port 1: INT3	0: Port 1: INT2	0: Port 1: INT1	0: Port 1: TA0IN
PDFC2	Port D Function register 2	0035H (Prohibit RMW)				PD4F2	PD3F2	PD2F2	PD1F2	
						W				
						0	0	0	0	
						<Refer to PDFC>				
PDCR	Port D Control register	0036H (Prohibit RMW)				PD4C	PD3C	PD2C		PD0C
						W				W
						0	0	0		0
						0: Input 1: Output				0: Input 1: Output
PDFC	Port D Function register	0037H (Prohibit RMW)				PD4F	PD3F	PD2F	PD1F	PD0F
						W				
						0	0	0	0	0

<PDxF2,PDxF, PDxC>	PD4	PD3	PD2	PD1	PD0
000	Input port	Input port	Input port	Input port	Input port
001	Output port	Output port	Output port		Output port
010	Reserved	RXD2	TB1IN1	TB1IN0	INT4
011	TB1OUT1	TB1OUT0	TXD2 (3-STATE)		TB0OUT0
100	SCLK2 input $\overline{\text{CTS2}}$ input	INT7	INT6	INT5	
101	SCLK2 output	Reserved	Reserved		
110	Reserved	Reserved	Reserved	Reserved	
111	Reserved	Reserved	TXD2 (Open Drain)		

I/O ports (4/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
PFFC2	Port F Function register 2	003DH (Prohibit RMW)						PF2F2		
								W		
								0		
								0:<PF2F> 1: CLK		
PFCR	Port F Control register	003EH (Prohibit RMW)			PF5C	PF4C	PF3C	PF2C	PF1C	PF0C
					W					
					0	0	0	0	0	0
					0: Input 1: Output					
PFFC	Port F Function register	003FH (Prohibit RMW)			PF5F	PF4F	PF3F	PF2F	PF1F	PF0F
					W					
					0	0	0	0	0	0

<PFxF2,PFxF,PFxC>	PF2	PF1	PF0
000	Input port	Input port	Input port
001	Output port	Output port	Output port
010	SCLK0 input $\overline{\text{CTS0}}$ input	RXD0	TXD0 (Open Drain)
011	SCLK0 output	Reserved	TXD0 (3-STATE)
100	Reserved	Reserved	Reserved
101	CLK output	Reserved	Reserved
110	Reserved	Reserved	Reserved
111	Reserved	Reserved	Reserved
<SIOCNT,PFxF2,PFxF,PFxC>	PF5	PF4	PF3
0000	Input port	Input port	Input port
0001	Output port	Output port	Output port
0010	SCLK1 input $\overline{\text{CTS1}}$ input	RXD1	TXD1 (Open Drain)
0011	SCLK1 output	Reserved	TXD1 (3-STATE)
1000	Reserved	Reserved	Reserved
1001	Reserved	Reserved	Reserved
1010	Reserved	HSSI input	Reserved
1011	HSCLK output	Reserved	HSSO(3-stage)

Symbol	Name	Address	7	6	5	4	3	2	1	0
PGFC	Port G Control register	0043H (Prohibit RMW)	PG7F	PG6F	PG5F	PG4F	PG3F	PG2F	PG1F	PG0F
			W							
			1	1	1	1	1	1	1	1
			0:Port/Key input 1: Analog input							
PLFC	Port L Function register	0057H (Prohibit RMW)					PL3F	PL2F	PL1F	PL0F
							W			
							1	1	1	1
							0: Port input 1: Analog input			
PNCR	Port N Control register	005EH (Prohibit RMW)			PN5C	PN4C	PN3C	PN2C	PN1C	PN0C
					W					
					0	0	0	0	0	0
					0: Input 1: Output					
PNFC	Port N Function register	005FH (Prohibit RMW)			PN5F	PN4F	PN3F	PN2F	PN1F	PN0F
					W					
					0	0	0	0	0	0

<PNxF,PNxC>	PN5	PN4	PN3	PN2	PN1	PN0
00	Input port	Input port	Input port	Input port	Input port	Input port
01	Output port	Output port	Output port	Output port	Output port	Output port
10	SI1 input	SO1 output	SCK1 input	SI0 input	SO0 output	SCK0 input
11	SCL1I/O	SDA1 I/O	SCK1 output	SCL0 I/O	SDA0 I/O	SCK0 output

Note1: When port P70 to P73 is used in the input mode, P7 register controls the built-in pull-up resistor. Read-modify-write is prohibited in the input mode or the I/O mode. Setting the built-in pull-up resistor may be depended on the states of the input pin.

Note 2: Notes on using low-frequency resonator to P76,P77, it is necessary to set the following procedures to reduce the consumption power supply.

・connecting to a resonator

Set P7CR<P76C,P77C>="11",P7<P76,P77>="00".

・connectiion to an oscillator

Set P7CR<P76C,P77C>="11",P7<P76,P77>="10".

Note 3: When using P83 as a $\overline{\text{WAIT}}$ input, while setting it as P8CR<P83C>= "0" and P8FC<P83F> = "1", it is necessary to set memory control register BxCSL<BxWW2:0> or <BxWR2:0> as "011".

Note 4: When setting P80 to P83 as a standard chip select signal ($\overline{\text{CS0}}$ to $\overline{\text{CS3}}$) output, P8CR is set up after setting up P8FC.

Note 5: PC0 is not based on a functional setup of a port, but is inputted into TA0IN of a 8-bit timer (TMRA0)

Note 6: TB1IN0 and TB1IN1 input is inputted into the 16-bit timer TMRB1 irrespective of a functional setup of a port.

Note 7: RXD2, SCLK2 input, and CTS2 input are inputted into the serial channel 2 irrespective of a functional setup of a port.

Note 8: PD2 does not have a register for 3-state / open drain setup. Moreover, there is no open drain function at the time of an output port.

Note 9: PF0 and PF3 does not have a register for 3-state / open drain setup. Moreover, there is no open drain function at the time of an output port.

Note10: Input channel selection of an AD converter in PG0 to PG7 and PL0 to PL3 is set up by AD mode control register ADMOD1 <ADCH3:0>. Moreover, a setup of AD trigger ($\overline{\text{ADTRG}}$) input permission is set up by ADMOD2 <ADTRGE>.

Note11: Specify the HSCSEL<SIOCNT> when selecting TXD1 or HSSO, RXD1 or HSSI and SCLK1 or HSCLK.

(2) Interrupt control (1/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
INTE01	INT0 & INT1 enable	00D0H	INT1				INT0			
			I1C	I1M2	I1M1	I1M0	I0C	I0M2	I0M1	I0M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1: INT1	Interrupt request level			1: INT0	Interrupt request level		
INTE23	INT2 & INT3 enable	00D1H	INT3				INT2			
			I3C	I3M2	I23M1	I3M0	I2C	I2M2	I2M1	I2M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1: INT3	Interrupt request level			1: INT2	Interrupt request level		
INTE45	INT4 & INT5 enable	00D2H	INT5				INT4			
			I5C	I5M2	I5M1	I5M0	I4C	I4M2	I4M1	I4M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1: INT5	Interrupt request level			1: INT4	Interrupt request level		
INTE67	INT6 & INT7 enable	00D3H	INT7				INT6			
			I7C	I7M2	I7M1	I7M0	I6C	I6M2	I6M1	I6M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1: INT7	Interrupt request level			1: INT6	Interrupt request level		
INTETA01	INTTA0 & INTTA1 enable	00D4H	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	0
			1: INTTA1	Interrupt request level			1: INTTA0	Interrupt request level		
INTETA23	INTTA2 & INTTA3 enable	00D5H	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	0
			1: INTTA3	Interrupt request level			1: INTTA2	Interrupt request level		
INTETA45	INTTA4 & INTTA5 enable	00D6H	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	0
			1: INTTA5	Interrupt request level			1: INTTA4	Interrupt request level		
INTES0	INTRX0 & INTTX0 enable	00D8H	INTTX0				INTRX0			
			ITX0C	ITX0M2	ITX0M1	ITX0M0	IRX0C	IRX0M2	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		
INTES1HSC	INTRX1 & INTTX1/ INTHSC enable	00D9H	INTTX1/INTHSC				INTRX1			
			ITX1C	ITX1M2	ITX1M1	ITX1M0	IRX1C	IRX1M2	IRX1M1	IRX1M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1: INTTX1	Interrupt request level			1: INTRX1	Interrupt request level		
INTES2	INTRX2 & INTTX2 enable	00DAH	INTTX2				INTRX2			
			ITX2C	ITX2M2	ITX2M1	ITX2M0	IRX2C	IRX2M2	IRX2M1	IRX2M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1: INTTX2	Interrupt request level			1: INTRX2	Interrupt request level		

Interrupt control (2/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
INTESB0	INTSBE0 enable	00DCH	–				INTSBE0			
			–	–	–	–	ISBE0C	ISBE0M2	ISBE0M1	ISBE0M0
			–	–			R	R/W		
			–	–	–	–	0	0	0	0
			Always write “0”				1: INTSBE0	Interrupt request level		
INTESB1	INTSBE1 enable	00DDH	–				INTSBE1			
			–	–	–	–	ISBE1C	ISBE1M2	ISBE1M1	ISBE1M0
			–	–			R	R/W		
			–	–	–	–	0	0	0	0
			Always write “0”				1: INTSBE1	Interrupt request level		
INTETB0	INTTB00 & INTTB01 enable	00E0H	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		
INTETBO0	INTTBO0 (Overflow) enable	00E1H	–				INTTBO0 (TMRB0)			
			–	–	–	–	ITBO0C	ITBO0M2	ITBO0M1	ITBO0M0
			–	–			R	R/W		
			–	–	–	–	0	0	0	0
			Always write “0”				1: INTTBO0	Interrupt request level		
INTETB1	INTTB10 & INTTB11 enable	00E2H	INTTB11 (TMRB1)				INTTB10 (TMRB1)			
			ITB11C	ITB11M2	ITB11M1	ITB11M0	ITB10C	ITB10M2	ITB10M1	ITB10M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1: INTTB11	Interrupt request level			1: INTTB10	Interrupt request level		
INTETBO1	INTTBO1 (Overflow) enable	00E3H	–				INTTBO1 (TMRB1)			
			–	–	–	–	ITBO1C	ITBO1M2	ITBO1M1	ITBO1M0
			–	–			R	R/W		
			–	–	–	–	0	0	0	0
			Always write “0”				1: INTTBO1	Interrupt request level		
INTEPAD	INTP0& INTAD enable	00E4H	INTP0				INTAD			
			IP0C	IP0M2	IP0M1	IP0M0	IADC	IADM2	IADM1	IADM0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1: INTP0	Interrupt request level			1: INTAD	Interrupt request level		
INTERTC	INTRTC enable	00E5H	–				INTRTC			
			–	–	–	–	IRC	IRM2	IRM1	IRM0
			–	–			R	R/W		
			–	–	–	–	0	0	0	0
			Always write “0”				1: INTRTC	Interrupt request level		
INTNMWDT	NMI & INTWD enable	00EFH	NMI				INTWDT			
			INCNM	–	–	–	INCWD	–	–	–
			R				R			
			0	–	–	–	0	–	–	–
			1: NMI	Always write “0”			1: INTWDT	Always write “0”		

Interrupt control (3/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
INTETC01	INTTC0 & INTTC1 enable	00F0H	INTTC1 (DMA1)				INTTC0 (DMA0)			
			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	00F1H	INTTC3 (DMA3)				INTTC2 (DMA2)			
			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		
INTETC45	NTTC4 & INTTC5 enable	00F2H	INTTC5 (DMA5)				INTTC4 (DMA4)			
			ITC5C	ITC5M2	ITC5M1	ITC5M0	ITC4C	ITC4M2	ITC4M1	ITC4M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1: INTTC5	Interrupt request level			1: INTTC4	Interrupt request level		
INTETC67	NTTC6 & INTTC7 enable	00F3H	INTTC7 (DMA7)				INTTC6 (DMA6)			
			ITC7C	ITC7M2	ITC7M1	ITC7M0	ITC6C	ITC6M2	ITC6M1	ITC6M0
			R	R/W			R	R/W		
			0	0	0	0	0	0	0	0
			1: INTTC7	Interrupt request level			1: INTTC6	Interrupt request level		

Interrupt control (4/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
HSCSEL	HSC Selection register	00F4H	–	–	–	–	–	–	–	SIOCNT
			R							R/W
			0	0	0	0	0	0	0	0
										0: SIO1 1: HSC
SIMC	SIO Interrupt Mode Control register	00F5H (Prohibit RMW)	–					IR2LE	IR1LE	IR0LE
			W					W		
			0					1	1	1
			Always write "1".					INTRX2 0: edge mode 1: level mode	INTRX1 0: edge mode 1: level mode	INTRX0 0: edge mode 1: level mode
IIMC	Interrupt Input Mode Control register	00F6H (Prohibit RMW)								NMIREE
										W
										0
										$\overline{\text{NMI}}$ 0:Falling 1:Falling and Rising
IIMC2	Interrupt Input Mode Control register2	00FAH (Prohibit RMW)	I7LE	I6LE	I5LE	I4LE	I3LE	I2LE	I1LE	I0LE
			W							
			0	0	0	0	0	0	0	0
			INT7 0: Edge 1: Level	INT6 0: Edge 1: Level	INT5 0: Edge 1: Level	INT4 0: Edge 1: Level	INT3 0: Edge 1: Level	INT2 0: Edge 1: Level	INT1 0: Edge 1: Level	INT0 0: Edge 1: Level
IIMC3	Interrupt Input Mode Control register3	00FBH (Prohibit RMW)	I7EDGE	I6EDGE	I5EDGE	I4EDGE	I3EDGE	I2EDGE	I1EDGE	I0EDGE
			W							
			0	0	0	0	0	0	0	0
			INT7 0: Rising /High 1: Falling /Low	INT6 0: Rising /High 1: Falling /Low	INT5 0: Rising /High 1: Falling /Low	INT4 0: Rising /High 1: Falling /Low	INT3 0: Rising /High 1: Falling /Low	INT2 0: Rising /High 1:Falling /Low	INT1 0: Rising /High 1: Falling /Low	INT0 0: Rising /High 1: Falling /Low
INTCLR	Interrupt Clear Control register	00F8H (Prohibit RMW)	CLRV7	CLRV6	CLRV5	CLRV4	CLRV3	CLRV2	CLRV1	CLRV0
			W							
			0	0	0	0	0	0	0	0
			Clear the interrupt request flag by the writing of a micro DMA starting vector							

(3) DMA controller

Symbol	Name	Address	7	6	5	4	3	2	1	0
DMA0V	DMA0 start vector	0100H			DMA0V5	DMA0V4	DMA0V3	DMA0V2	DMA0V1	DMA0V0
					R/W					
					0	0	0	0	0	0
					DMA0 start vector					
DMA1V	DMA1 start vector	0101H			DMA1V5	DMA1V4	DMA1V3	DMA1V2	DMA1V1	DMA1V0
					R/W					
					0	0	0	0	0	0
					DMA1 start vector					
DMA2V	DMA2 start vector	0102H			DMA2V5	DMA2V4	DMA2V3	DMA2V2	DMA2V1	DMA2V0
					R/W					
					0	0	0	0	0	0
					DMA2 start vector					
DMA3V	DMA3 start vector	0103H			DMA3V5	DMA3V4	DMA3V3	DMA3V2	DMA3V1	DMA3V0
					R/W					
					0	0	0	0	0	0
					DMA3 start vector					
DMA4V	DMA4 start vector	0104H			DMA4V5	DMA4V4	DMA4V3	DMA4V2	DMA4V1	DMA4V0
					R/W					
					0	0	0	0	0	0
					DMA4 start vector					
DMA5V	DMA5 start vector	0105H			DMA5V5	DMA5V4	DMA5V3	DMA5V2	DMA5V1	DMA5V0
					R/W					
					0	0	0	0	0	0
					DMA5 start vector					
DMA6V	DMA6 start vector	0106H			DMA6V5	DMA6V4	DMA6V3	DMA6V2	DMA6V1	DMA6V0
					R/W					
					0	0	0	0	0	0
					DMA6 start vector					
DMA7V	DMA7 start vector	0107H			DMA7V5	DMA7V4	DMA7V3	DMA7V2	DMA7V1	DMA7V0
					R/W					
					0	0	0	0	0	0
					DMA7 start vector					
DMAB	DMA burst	0108H	DBST7	DBST6	DBST5	DBST4	DBST3	DBST2	DBST1	DBST0
			R/W							
			0	0	0	0	0	0	0	0
			1: DMA request on burst mode							
DMAR	DMA request	0109H (Prohibit RMW)	DREQ7	DREQ6	DREQ5	DREQ4	DREQ3	DREQ2	DREQ1	DREQ0
			R/W							
			0	0	0	0	0	0	0	0
			1: DMA request in software							

(4) Memory controller (1/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0
B0CSL	Block 0 MEMC Control register Low	0140H (Prohibit RMW)		B0WW2	B0WW1	B0WW0		B0WR2	B0WR1	B0WR0
				W				W		
				0	1	0		0	1	0
				Write waits 001: 0 WAIT 010: 1 WAIT 101: 2 WAIT 110: 3 WAIT 111: 4 WAIT 011: $\overline{\text{WAIT}}$ pin Others: Reserved				Read waits 001: 0 WAIT 010: 1 WAIT 101: 2 WAIT 110: 3 WAIT 111: 4 WAIT 011: $\overline{\text{WAIT}}$ pin Others: Reserved		
B0CSH	Block 0 MEMC Control register High	0141H (Prohibit RMW)	B0E			B0REC	B0OM1	B0OM0	B0BUS1	B0BUS0
			W			W				
			0			0	0	0	0	0
			CS select 0: Disable 1: Enable			0: Not insert a dummy cycle 1: insert a dummy cycle	00: ROM/SRAM 01: Reserved 10: Reserved 11: Reserved		Data Bus width 00: 8-bit 01: 16-ibt 10: Reserved 11: Reserved	
B1CSL	Block 1 MEMC Control register Low	0144H (Prohibit RMW)		B1WW2	B1WW1	B1WW0		B1WR2	B1WR1	B1WR0
				W				W		
				0	1	0		0	1	0
				Write waits 001: 0 WAIT 010: 1 WAIT 101: 2 WAIT 110: 3 WAIT 111: 4 WAIT 011: $\overline{\text{WAIT}}$ pin Others: Reserved				Read waits 001: 0 WAIT 010: 1 WAIT 101: 2 WAIT 110: 3 WAIT 111: 4 WAIT 011: $\overline{\text{WAIT}}$ pin Others: Reserved		
B1CSH	Block 1 MEMC control register High	0145H (Prohibit RMW)	B1E			B1REC	B1OM1	B1OM0	B1BUS1	B1BUS0
			W			W				
			0			0	0	0	0	0
			CS select 0:Disable 1:Enable			0: Not insert a dummy cycle 1: insert a dummy cycle	00: ROM/SRAM 01: Reserved 10: Reserved 11: Reserved		Data Bus width 00: 8-bit 01: 16-ibt 10: Reserved 11: Reserved	
B2CSL	Block 2 MEMC control register Low	0148H (Prohibit RMW)		B2WW2	B2WW1	B2WW0		B2WR2	B2WR1	B2WR0
				W				W		
				0	1	0		0	1	0
				Write waits 001: 0 WAIT 010: 1 WAIT 101: 2 WAIT 110: 3 WAIT 111: 4 WAIT 011: $\overline{\text{WAIT}}$ pin Others: Reserved				Read waits 001: 0 WAIT 010: 1 WAIT 101: 2 WAIT 110: 3 WAIT 111: 4 WAIT 011: $\overline{\text{WAIT}}$ pin Others: Reserved		
B2CSH	Block 2 MEMC control register High	0149H (Prohibit RMW)	B2E	B2M		B2REC	B2OM1	B2OM0	B2BUS1	B2BUS0
			W			W				
			1	0		0	0	0	0/1 (Note)	0/1 (Note)
			CS select 0:Disable 1:Enable	0:16 MB 1: Sets area		0: Not insert a dummy cycle 1: insert a dummy cycle	00: ROM/SRAM 01: Reserved 10: Reserved 11: Reserved		Data Bus width 00: 8-bit 01: 16-ibt 10: Reserved 11: Reserved	

Note: Since after reset becomes unfixed, please be sure to set up bus bit B2CSH<B2BUS1:0> of the control register before accessing the external block address area 2.

Memory controller (2/3)

Symbol	Name	Address	7	6	5	4	3	2	1	0
B3CSL	Block 3 MEMC control register Low	014CH (Prohibit RMW)		B3WW2	B3WW1	B3WW0		B3WR2	B3WR1	B3WR0
				W				W		
				0	1	0		0	1	0
				Write waits 001: 0 WAIT 010: 1 WAIT 101: 2 WAIT 110: 3 WAIT 111: 4 WAIT 011: $\overline{\text{WAIT}}$ pin Others: Reserved				Read waits 001: 0 WAIT 010: 1 WAIT 101: 2 WAIT 110: 3 WAIT 111: 4 WAIT 011: $\overline{\text{WAIT}}$ pin Others: Reserved		
B3CSH	Block 3 MEMC control register High	014DH (Prohibit RMW)	B3E			B3REC	B3OM1	B3OM0	B3BUS1	B3BUS0
			W			W				
			0			0	0	0	0	0
			CS select 0:Disable 1:Enable			0: Not insert a dummy cycle 1: insert a dummy cycle	00: ROM/SRAM 01: Reserved 10: Reserved 11: Reserved		Data Bus width 00: 8-bit 01: 16-ibt 10: Reserved 11: Reserved	
BEXCSL	BLOCK EX MEMC Control register Low	0158H (Prohibit RMW)		BEXWW2	BEXWW1	BEXWW0		BEXWR2	BEXWR1	BEXWR0
				W				W		
				0	1	0		0	1	0
				Write waits 001: 0 WAIT 010: 1 WAIT 101: 2 WAIT 110: 3 WAIT 111: 4 WAIT 011: $\overline{\text{WAIT}}$ pin Others: Reserved				Read waits 001: 0 WAIT 010: 1 WAIT 101: 2 WAIT 110: 3 WAIT 111: 4 WAIT 011: $\overline{\text{WAIT}}$ pin Others: Reserved		
BEXCSH	BLOCK EX MEMC Control register High	0159H (Prohibit RMW)				BEXREC	BEXOM1	BEXOM0	BEXBUS1	BEXBUS0
						W				
						0	0	0	0	0
						0: Not insert a dummy cycle 1: insert a dummy cycle	00: ROM/SRAM 01: Reserved 10: Reserved 11: Reserved		Data Bus width 00: 8-bit 01: 16-ibt 10: Reserved 11: Reserved	
PMEMCR	Page ROM Control register	0166H				OPGE	OPWR1	OPWR0	PR1	PR0
						R/W				
						0	0	0	1	0
						ROM page access 0: Disable 1: Enable	Wait number on page 00:1 state (n-1-1-1 mode) 01:2 state (n-2-2-2 mode) 10:3 state (n-3-3-3 mode) 11:Reserved		Byte number in a page 00:64 byte 01:32 byte 10:16 byte 11:8 byte	

Memory controller (3/3)

Symbol	Name	Address	7	6	5	4	3	2	1	0
MAMR0	Memory Mask register 0	0142H	M0V20	M0V19	M0V18	M0V17	M0V16	M0V15	M0V14-9	M0V8
			R/W							
			1	1	1	1	1	1	1	1
			0: Compare enable 1: Compare disable							
MSAR0	Memory Start Address register 0	0143H	M0S23	M0S22	M0S21	M0S20	M0S19	M0S18	M0S17	M0S16
			R/W							
			1	1	1	1	1	1	1	1
			Set start address A23 to A16							
MAMR1	Memory Mask register 1	0146H	M1V21	M1V20	M1V19	M1V18	M1V17	M1V16	MV15-9	M1V8
			R/W							
			1	1	1	1	1	1	1	1
			0: Compare enable 1: Compare disable							
MSAR1	Memory Start Address register 1	0147H	M1S23	M1S22	M1S21	M1S20	M1S19	M1S18	M1S17	M1S16
			R/W							
			1	1	1	1	1	1	1	1
			Set start address A23 to A16							
MAMR2	Memory Mask register 2	014AH	M2V22	M2V21	M2V20	M2V19	M2V18	M2V17	M2V16	M2V15
			R/W							
			1	1	1	1	1	1	1	1
			0: Compare enable 1: Compare disable							
MSAR2	Memory Start Address register 3	014BH	M2S23	M2S22	M2S21	M2S20	M2S19	M2S18	M2S17	M2S16
			R/W							
			1	1	1	1	1	1	1	1
			Set start address A23 to A16							
MAMR3	Memory Mask register 3	014EH	M3V22	M3V21	M3V20	M3V19	M3V18	M3V17	M3V16	M3V15
			R/W							
			1	1	1	1	1	1	1	1
			0: Compare enable 1: Compare disable							
MSAR3	Memory Start Address register 3	014FH	M3S23	M3S22	M3S21	M3S20	M3S19	M3S18	M3S17	M3S16
			R/W							
			1	1	1	1	1	1	1	1
			Set start address A23 to A16							

(5) Clock control/PLL (1/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0
SYSCR0	System Clock Control register 0	10E0H	XEN	XTEN				WUEF		
			R/W					R/W		
			1	0				0		
			High-frequency oscillator (f_{OSCH}) 0: Stop 1: Oscillation	Low-frequency oscillator (fs) 0: Stop 1: Oscillation				Warm-up timer 0: Write don't care 1: Write start timer 0: Read end warm-up 1: Read do not end warm-up		
SYSCR1	System Clock Control register 1	10E1H					SYSCK	GEAR2	GEAR1	GEAR0
							R/W			
							0	1	0	0
							Select system clock 0: fc 1: fs	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	10E2H	~		WUPTM1	WUPTM0	HALTM1	HALTM0		DRVE
			W		R/W					R/W
			0		1	0	1	1		0
			Always write "0"		Warm-up timer 00: Reserved 01: 2^{8}/input frequency 10: 2^{14}/input frequency 11: 2^{16}/input frequency		HALT mode 00: Reserved 01: STOP mode 10: IDLE1 mode 11: IDLE2 mode			1: The inside of STOP mode also drives a pin
PLLCR0	PLL Control register 0	10E8H		FCSEL	LWUPFG					
				R/W	R					
				0	0					
				Select fc clock 0: f_{OSCH} 1: f_{PLL}	Lock up timer status flag 0: Not end 1: End					
PLLCR1	PLL Control register 1	10E9H	PLLON							
			R/W							
			0							
			Control on/off 0: OFF 1: ON							

Clock control/PLL (2/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0
EMCCR0	EMC Control register 0	10E3H	PROTECT					–	–	DRVOSCL
			R					R/W		
			0					0	1	1
			Protect flag 0: OFF 1: ON					Always write “0”	Always write “1”	fs oscillator driver ability 1: Normal 0: Weak
EMCCR1	EMC Control register 1	10E4H	Switch the protect ON/OFF by writing the following to 1st-KEY, 2nd-KEY 1st-KEY: write in sequence EMCCR1 = 5AH, EMCCR2 = A5H 2nd-KEY: write in sequence EMCCR1 = A5H, EMCCR2 = 5AH							
EMCCR2	EMC Control register 2	10E5H								

(6) 8-bit timer (1/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0
TA01RUN	8-bit timer RUN register	1100H	TA0RDE				I2TA01	TA01PRUN	TA1RUN	TA0RUN
			R/W				R/W			
			0				0	0	0	0
			Double buffer 0: Disable 1: Enable				IDLE2 0: Stop 1: Operate	TMRA01 prescaler	UC1	UC0
								0: Stop and clear 1: Run (Count up)		
TA0REG	8-bit timer register 0	1102H (Prohibit RMW)	–							
			W							
			Undefined							
TA1REG	8-bit timer register 1	1103H (Prohibit RMW)	–							
			W							
			Undefined							
TA01MOD	8-bit timer source CLK & mode register	1104H	TA01M1	TA01M0	PWM01	PWM00	TA1CLK1	TA1CLK0	TA0CLK1	TA0CLK0
			R/W							
			0	0	0	0	0	0	0	0
			Operation mode 00: 8-bit timer mode 01: 16-bit timer mode 10: 8-bit PPG mode 11: 8-bit PWM mode		PWM cycle 00: Reserved 01: 2^6 10: 2^7 11: 2^8		Source clock for TMRA1 00: TA0TRG 01: ϕT1 10: ϕT16 11: ϕT256		Source clock for TMRA0 00: TA0IN pin input 01: ϕT1 10: ϕT4 11: ϕT16	
TA1FFCR	8-bit timer flip-flop control register	1105H (Prohibit RMW)					TA1FFC1	TA1FFC0	TA1FFIE	TA1FFIS
							R/W		R/W	
							1	1	0	0
							00: Invert TA1FF 01: Set TA1FF 10: Clear TA1FF 11: Don't care		TA1FF control for inversion 0: Disable 1: Enable	TA1FF inversion select 0: TMRA0 1: TMRA1
TA23RUN	8-bit timer RUN register	1108H	TA2RDE				I2TA23	TA23PRUN	TA3RUN	TA2RUN
			R/W				R/W			
			0				0	0	0	0
			Double buffer 0: Disable 1: Enable				IDLE2 0: Stop 1: Operate	TMRA23 prescaler	UC3	UC2
								0: Stop and clear 1: Run (Count up)		
TA2REG	8-bit timer register 2	110AH (Prohibit RMW)	–							
			W							
			Undefined							
TA3REG	8-bit timer register 3	110BH (Prohibit RMW)	–							
			W							
			Undefined							
TA23MOD	8-bit timer source CLK & mode register r	110CH	TA23M1	TA23M0	PWM21	PWM20	TA3CLK1	TA3CLK0	TA2CLK1	TA2CLK0
			R/W							
			0	0	0	0	0	0	0	0
			Operation mode 00: 8-bit timer mode 01: 16-bit timer mode 10: 8-bit PPG mode 11: 8-bit PWM mode		PWM cycle 00: Reserved 01: 2^6 10: 2^7 11: 2^8		Source clock for TMRA3 00: TA2TRG 01: ϕT1 10: ϕT16 11: ϕT256		Source clock for TMRA2 00: Reserved 01: ϕT1 10: ϕT4 11: ϕT16	
TA3FFCR	8-bit timer flip-flop control register	110DH (Prohibit RMW)					TA3FFC1	TA3FFC0	TA3FFIE	TA3FFIS
							R/W		R/W	
							1	1	0	0
							00: Invert TA3FF 01: Set TA3FF 10: Clear TA3FF 11: Don't care		TA3FF control for inversion 0: Disable 1: Enable	TA3FF inversion select 0: TMRA2 1: TMRA3

8-bit timer (2/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0
TA45RUN	8-bit timer RUN register	1110H	TA4RDE				I2TA45	TA45PRUN	TA5RUN	TA4RUN
			R/W				R/W			
			0				0	0	0	0
			Double buffer 0: Disable 1: Enable				IDLE4 0: Stop 1: Operate	TMRA45 prescaler	UC5	UC4
								0: Stop and clear 1: Run (Count up)		
TA4REG	8-bit timer register 4	1112H (Prohibit RMW)	–							
			W							
			Undefined							
TA5REG	8-bit timer register 5	1113H (Prohibit RMW)	–							
			W							
			Undefined							
TA45MOD	8-bit timer source CLK & mode register	1114H	TA45M1	TA45M0	PWM41	PWM40	TA5CLK1	TA5CLK0	TA4CLK1	TA4CLK0
			R/W							
			0	0	0	0	0	0	0	0
			Operation mode 00: 8-bit timer mode 01: 16-bit timer mode 10: 8-bit PPG mode 11: 8-bit PWM mode		PWM cycle 00: Reserved 01: 2^6 10: 2^7 11: 2^8		Source clock for TMRA5 00: TA4TRG 01: ϕT1 10: ϕT16 11: ϕT256		Source clock for TMRA4 00: Reserved 01: ϕT1 10: ϕT4 11: ϕT16	
TA5FFCR	8-bit timer flip-flop control register	1115H (Prohibit RMW)					TA5FFC1	TA5FFC0	TA5FFIE	TA5FFIS
							R/W		R/W	
							1	1	0	0
							00: Invert TA5FF 01: Set TA5FF 10: Clear TA5FF 11: Don't care		TA5FF control for inversion 0: Disable 1: Enable	TA5FF inversion select 0: TMRA4 1: TMRA5

(7) 16-bit timer (1/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0
TB0RUN	16-bit timer RUN register	1180H	TB0RDE	–			I2TB0	TB0PRUN		TB0RUN
			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	TMRB0 prescaler		Up counter (UC0)
								0: Stop and clear 1: Run (Count up)		
TB0MOD	16-bit timer source CLK & mode register	1182H (Prohibit RMW)	–	–	TB0CP0I	TB0CPM1	TB0CPM0	TB0CLE	TB0CLK1	TB0CLK0
			R/W		W	R/W				
			0	0	1	0	0	0	0	0
			Always write “0”		Software capture control 0: Software capture 1: Undefined	Capture timing 00: Disable 01: Reserved 10: Reserved 11:TA1OUT↑ TA1OUT↓		Up counter control 0: Disable 1: Enable	TMRB0 source clock 00: Reserved 01: φT1 10: φT4 11: φT16	
TB0FFCR	16-bit timer flip-flop control register	1183H (Prohibit RMW)	–	–	TB0C1T1	TB0C0T1	TB0E1T1	TB0E0T1	TB0FFC1	TB0FFC0
			W*		R/W				W*	
			1	1	0	0	0	0	1	1
			Always write “11”.		TB0FF0 inversion trigger 0: Disable trigger 1: Enable trigger				Control TB0FF0 00: Invert 01: Set 10: Clear 11: Don’t care * Always read as 11.	
					Invert when the UC value is loaded in to TB0CP1H/L	Invert when the UC value is loaded in to TB0CP0H/L	Invert when the UC value matches the value in TB0RG1H/L	Invert when the UC value matches the value in TB0RG0H/L		
TB0RG0L	16-bit timer register 0 Low	1188H (Prohibit RMW)	–							
			W							
			Undefined							
TB0RG0H	16-bit timer register 0 High	1189H (Prohibit RMW)	–							
			W							
			Undefined							
TB0RG1L	16-bit timer register 1 Low	118AH (Prohibit RMW)	–							
			W							
			Undefined							
TB0RG1H	16-bit timer register 1 High	118BH (Prohibit RMW)	–							
			W							
			Undefined							
TB0CP0L	16-bit timer Capture register 0Low	118CH	–							
			R							
			Undefined							
TB0CP0H	16-bittimer Capture register 0 High	118DH	–							
			R							
			Undefined							
TB0CP1L	16-bit timer Capture register 1 Low	118EH	–							
			R							
			Undefined							
TB0CP1H	16-bit timer Capture register 1 High	118FH	–							
			R							
			Undefined							

16-bitTimer (2/2)

Symbol	Name	Address	7	6	5	4	3	2	1	0
TB1RUN	16-bit timer RUN register	1190H	TB1RDE	–			I2TB1	TB1PRUN		TB1RUN
			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	TMRB1 prescaler		Up counter (UC1)
								0: Stop and clear 1: Run (Count up)		
TB1MOD	16-bit timer source CLK & mode register	1192H (Prohibit RMW)	TB1CT1	TB1ET1	TB1CP0I	TB1CPM1	TB1CPM0	TB1CLE	TB1CLK1	TB1CLK0
			R/W		W	R/W				
			0	0	1	0	0	0	0	0
			TB1FF1 Inversion trigger 0: Trigger disable 1: Trigger enable		Software capture control 0: Software capture 1: Undefined	Capture timing 00: Disable INT5 is rising edge 01: TB1N0 ↑ TB1IN1 ↑ INT5 is rising edge 10: TB1IN0 ↑ TB1IN0 ↓ INT5 is falling edge 11: TA3OUT ↑ TA3OUT ↓ INT5 is rising edge		Up counter control 0: Disable 1: Enable	TMRB1 source clock 00: TB1IN0 pin input 01: ϕT1 10: ϕT4 11: ϕT16	
			Invert when capture to capture register 1	Invert when match UC0 with TB1RG1H/L						
TB1FFCR	16-bit timer flip-flop control register	1193H (Prohibit RMW)	TB1FF1C1	TB1FF1C0	TB1C1T1	TB1C0T1	TB1E1T1	TB1E0T1	TB1FFC1	TB1FFC0
			W*		R/W				W*	
			1	1	0	0	0	0	1	1
			TB1FF1 control 00: Invert 01: Set 10: Clear 11: Don’t care * Always read as “11”.		TB0FF0 inversion trigger 0: Disable trigger 1: Enable trigger				Control TB1FF0 00: Invert 01: Set 10: Clear 11: Don’t care * Always read as 11.	
					Invert when the UC value is loaded in to TB1CP1H/L	Invert when the UC value is loaded in to TB1CP0H/L	Invert when the UC value matches the value in TB1RG1H/L.	Invert when the UC value matches the value in TB1RG0H/L.		
TB1RG0L	16-bit timer register 0 Low	1198H (Prohibit RMW)	–							
			W							
			Undefined							
TB1RG0H	16-bit timer register 0 High	1199H (Prohibit RMW)	–							
			W							
			Undefined							
TB1RG1L	16-bit timer register 1 Low	119AH (Prohibit RMW)	–							
			W							
			Undefined							
TB1RG1H	16-bit timer register 1 High	119BH (Prohibit RMW)	–							
			W							
			Undefined							
TB1CP0L	16-bit timer Capture register 0 Low	119CH	–							
			R							
			Undefined							
TB1CP0H	16-bittimer Capture register 0 High	119DH	–							
			R							
			Undefined							
TB1CP1L	16-bit timer Capture register 1 Low	119EH	–							
			R							
			Undefined							
TB1CP1H	16-bit timer Capture register 1 High	119FH	–							
			R							
			Undefined							

(8) High speed serial (1/3)

Symbol	Name	Address	7	6	5	4	3	2	1	0
HSC0MD	High Speed Serial Mode register	0C00H		XEN0				CLKSEL02	CLKSEL01	CLKSEL00
				R/W				R/W		
				0				1	0	0
				SYSCK 0: Disable 1: Enable				Select baud rate 000:Reserved 100:f_{SYS}/16 001: f_{SYS}/2 101: f_{SYS}/32 010: f_{SYS}/4 110: f_{SYS}/64 011: f_{SYS}/8 111:Reserved		
		0C01H	LOOPBACK0	MSB1ST0	DOSTAT0		TCPOL0	RCPOL0	TDINV0	RDINV0
			R/W				R/W			
			0	1	1		0	0	0	0
			LOOPBACK test mode 0: Disable 1: Enable	Start bit for transmit /receive 0:LSB 1:MSB	HSSO pin (no transmit) 0: fixed to ”0” 1:fixed to ”1”		Synchronous clock edge during transmitting 0: fall 1: rise	Synchronous clock edge during receiving 0: fall 1: rise	Invert data During transmitting 0:Disable 1:Enable	Invert data During receiving 0:Disable 1:Enable
HSC0CT	High Speed Serial Control register	0C02H	–	–	UNIT160			ALGNEN0	RXWEN0	RXUEN0
			R/W					R/W		
			0	1	0			0	0	0
			Always write “0”	Always write “1”	Data length 0: 8bit 1: 16bit			Full duplex alignment 0:Disable 1:Enable	Sequential receive 0:Disable 1:Enable	Receive UNIT 0:Disable 1:Enable
		0C03H	CRC16_7_B0	CRCRX_TX_B0	CRCREST_B0				DMAERFW0	DMAERFR0
			R/W						R/W	
			0	0	0				0	0
			CRC select 0:CRC7 1:CRC16	CRC data 0:Transmit 1:Receive	CRC calculate register 0: Reset 1:Release Reset				Micro DMA 0: Disable 1: Enable	Micro DMA 0: Disable 1: Enable
HSC0ST	High Speed Serial Status register	0C04H					TEND0	REND0	RFW0	RFR0
							R			
							1	0	1	0
							Transmitting 0:operation 1: no operation	Receive Shift register 0: no data 1: exist data	Transmit buffer 0: untransmitted data exist 1: no untransmitted data	Receive buffer 0: no valid data 1: valid data exist
		0C05H								
HSC0CR	High Speed Serial CRC register	0C06H	CRCD007	CRCD006	CRCD005	CRCD004	CRCD003	CRCD002	CRCD001	CRCD000
			R							
			0	0	0	0	0	0	0	0
			CRC calculation result load register[7:0]							
		0C07H	CRCD015	CRCD014	CRCD013	CRCD012	CRCD011	CRCD010	CRCD009	CRCD008
			R							
			0	0	0	0	0	0	0	0
			CRC calculation result load register[15:8]							

High speed serial (2/3)

Symbol	Name	Address	7	6	5	4	3	2	1	0
HSC0IS	High Speed Serial Interrupt status register	0C08H					TENDIS0	RENDIS0	RFWIS0	RFRIS0
							R/W			
							0	0	0	0
							Read 0: no interrupt 1: interrupt Write 0: Don’t care 1: clear	Read 0: no interrupt 1: interrupt Write 0: Don’t care 1: clear	Read 0: no interrupt 1: interrupt Write 0: Don’t care 1: clear	Read 0: no interrupt 1: interrupt Write 0: Don’t care 1: clear
		0C09H								
HSC0WE	High Speed Serial interrupt status write enable register	0C0AH					TENDWE0	RENDWE0	RFWWE0	RFRWE0
							R/W			
							0	0	0	0
							Clear HSC0IS <TENDIS0> 0: Disable 1: Enable	Clear HSC0IS <RENDIS0> 0: Disable 1: Enable	Clear HSC0IS <RFWIS0> 0: Disable 1: Enable	Clear HSC0IS <RFRIS0> 0: Disable 1: Enable
		0C0BH								
HSC0IE	High Speed Serial Interrupt enable register	0C0CH					TENDIE0	RENDIE0	RFWIE0	RFRIE0
							R/W			
							0	0	0	0
							TEND0 interrupt 0: Disable 1: Enable	REND0 interrupt 0: Disable 1: Enable	RFW0 interrupt 0: Disable 1: Enable	RFR0 interrupt 0: Disable 1: Enable
		0C0DH								
HSC0IR	High Speed Serial Interrupt request register	0C0EH					TENDIR0	RENDIR0	RFWIR0	RFRIR0
							R			
							0	0	0	0
							TEND0 interrupt 0: None 1: generate	REND0 interrupt 0: None 1: generate	RFW0 interrupt 0: None 1: generate	RFR0 interrupt 0: None 1: generate
		0C0FH								

High speed serial (3/3)

Symbol	Name	Address	7	6	5	4	3	2	1	0
HSC0TD	High Speed Serial transmission data register	0C10H	TXD007	TXD006	TXD005	TXD004	TXD003	TXD002	TXD001	TXD000
			R/W							
			0	0	0	0	0	0	0	0
			Transmission data register [7:0]							
		0C11H	TXD015	TXD014	TXD013	TXD012	TXD011	TXD010	TXD009	TXD008
			R/W							
			0	0	0	0	0	0	0	0
			Transmission data register [15:8]							
HSC0RD	High Speed Serial receiving data register	0C12H	RXD007	RXD006	RXD005	RXD004	RXD003	RXD002	RXD001	RXD000
			R/W							
			0	0	0	0	0	0	0	0
			Receive data register [7:0]							
		0C13H	RXD015	RXD014	RXD013	RXD012	RXD011	RXD010	RXD009	RXD008
			R/W							
			0	0	0	0	0	0	0	0
			Receive data register [15:8]							
HSC0TS	High Speed Serial transmit data shift register	0C14H	TSD007	TSD006	TSD005	TSD004	TSD003	TSD002	TSD001	TSD000
			R/W							
			0	0	0	0	0	0	0	0
			Transmit data shift register [7:0]							
		0C15H	TSD015	TSD014	TSD013	TSD012	TSD011	TSD010	TSD009	TSD008
			R/W							
			0	0	0	0	0	0	0	0
			Transmit data shift register [15:8]							
HSC0RS	High Speed Serial receive data shift register	0C16H	RSD007	RSD006	RSD005	RSD004	RSD003	RSD002	RSD001	RSD000
			R/W							
			0	0	0	0	0	0	0	0
			Receive data shift register [7:0]							
		0C17H	RSD015	RSD014	RSD013	RSD012	RSD011	RSD010	RSD009	RSD008
			R/W							
			0	0	0	0	0	0	0	0
			Receive data shift register [15:8]							

(9) UART/serial channel (1/3)

Symbol	Name	Address	7	6	5	4	3	2	1	0
SC0BUF	Serial channel 0 Buffer register	1200H (Prohibit RMW)	RB7/TB7	RB6/TB6	RB5/TB5	RB4/TB4	RB3/TB3	RB2/TB2	RB1/TB1	RB0/TB0
			R (Receive)/W (Transmission)							
			Undefined							
SC0CR	Serial channel 0 Control register	1201H	RB8	EVEN	PE	OERR	PERR	FERR	SCLKS	IOC
			R	R/W		R (Cleared to 0 when read)			R/W	
			Undefined	0	0	0	0	0	0	0
			Received data bit8	Parity 0: Odd 1: Even	Parity addition 0: Disable 1: Enable	1: Error			0:SCLK0 ↑ 1:SCLK0 ↓	0: Baud rate generator 1: SCLK0 pin input
						Overrun	Parity	Framing		
SC0MOD0	Serial channel 0 Mode0 register	1202H	TB8	CTSE	RXE	WU	SM1	SM0	SC1	SC0
			R/W							
			0	0	0	0	0	0	0	0
			Transfer data bit8	Hand shake 0: CTS disable 1: CTS enable	Receive function 0: Receive disable 1: Receive enable	Wakeup function 0: Disable 1: Enable	Serial transmission mode 00: I/O interface mode 01: 7-bit UART mode 10: 8-bit UART mode 11: 9-bit UART mode		Serial transmission clock (UART) 00: TMRA0 trigger 01: Baud rate generator 10: Internal clock f_{SYS} 11: External clcok (SCLK0 input)	
BR0CR	Serial channel 0 Baud rate Control register	1203H	−	BR0ADDE	BR0CK1	BR0CK0	BR0S3	BR0S2	BR0S1	BR0S0
			R/W							
			0	0	0	0	0	0	0	0
			Always write “0”.	+(16 − K) /16 division 0: Disable 1: Enable	00: φT0 01: φT2 10: φT8 11: φT32		Divided frequency setting			
BR0ADD	Serial channel 0 K setting register	1204H					BR0K3	BR0K2	BR0K1	BR0K0
							R/W			
							0	0	0	0
							Sets frequency divisor “K” (divided by N + (16˜ K)/16).			
SC0MOD1	Serial channel 0 Mode1 register	1205H	I2S0	FDPX0						
			R/W							
			0	0						
			IDLE2 0: Stop 1: Run	Duplex 0: Half 1: Full						
SIR0CR	IrDA control register 0	1207H	PLSEL	RXSEL	TXEN	RXEN	SIR0WD3	SIR0WD2	SIR0WD1	SIR0WD0
			R/W							
			0	0	0	0	0	0	0	0
			Select transmit pulse width 0: 3/16 1: 1/16	Receive data 0: “H” pulse 1: “L” pulse	Transmit 0: Disable 1: Enable	Receive 0: Disable 1: Enable	Select receive pulse width Set effective pulse width for equal or more than 2x × (value + 1) + 100 ns Can be set: 1 to 14 Can not be set: 0, 15			

UART/serial channel (2/3)

Symbol	Name	Address	7	6	5	4	3	2	1	0
SC1BUF	Serial channel 1 Buffer register	1208H (Prohibit RMW)	RB7/TB7	RB6/TB6	RB5/TB5	RB4/TB4	RB3/TB3	RB2/TB2	RB1/TB1	RB0/TB0
			R (Receive)/W (Transmission)							
			Undefined							
SC1CR	Serial channel 1 Control register	1209H	RB8	EVEN	PE	OERR	PERR	FERR	SCLKS	IOC
			R	R/W		R (Cleared to 0 when read)			R/W	
			Undefined	0	0	0	0	0	0	0
			Received data bit8	Parity 0: Odd 1: Even	Parity addition 0: Disable 1: Enable	1: Error			0:SCLK1 ↑ 1:SCLK1 ↓	0: Baud rate generator 1: SCLK1 pin input
						Overrun	Parity	Framing		
SC1MOD0	Serial channel 1 Mode0 register	120AH	TB8	CTSE	RXE	WU	SM1	SM0	SC1	SC0
			R/W							
			0	0	0	0	0	0	0	0
			Transfer data bit8	Hand shake 0: CTS disable 1: CTS enable	Receive function 0: Receive disable 1: Receive enable	Wakeup function 0: Disable 1: Enable	Serial transmission mode 00: I/O interface mode 01: 7-bit UART mode 10: 8-bit UART mode 11: 9-bit UART mode		Serial transmission clock (UART) 00: TMRA0 trigger 01: Baud rate generator 10: Internal clock f_{SYS} 11: External clock (SCLK1 input)	
BR1CR	Serial channel 1 Baud rate Control register	120BH	–	BR1ADDE	BR1CK1	BR1CK0	BR1S3	BR1S2	BR1S1	BR1S0
			R/W							
			0	0	0	0	0	0	0	0
			Always write "0".	+(16 − K)/16 division 0: Disable 1: Enable	00: ϕT0 01: ϕT2 10: ϕT8 11: ϕT32		Divided frequency setting			
BR1ADD	Serial channel 1 K setting register	120CH					BR1K3	BR1K2	BR1K1	BR1K0
							R/W			
							0	0	0	0
							Sets frequency divisor "K" (divided by N + (16 − K)/16).			
SC1MOD1	Serial channel 1 Mode1 register	120DH	I2S1	FDPX1						
			R/W							
			0	0						
			IDLE2 0: Stop 1: Run	Duplex 0: Half 1: Full						
SIR1CR	IrDA control register 1	120FH	PLSEL	RXSEL	TXEN	RXEN	SIR1WD3	SIR1WD2	SIR1WD1	SIR1WD0
			R/W							
			0	0	0	0	0	0	0	0
			Select transmit pulse width 0: 3/16 1: 1/16	Receive data 0: "H" pulse 1: "L" pulse	Transmit 0: Disable 1: Enable	Receive 0: Disable 1: Enable	Select receive pulse width Set effective pulse width for equal or more than 2x × (value + 1) + 100 ns Can be set: 1 to 14 Can not be set: 0, 15			

UART/serial channel (3/3)

Symbol	Name	Address	7	6	5	4	3	2	1	0
SC2BUF	Serial channel 2 Buffer register	1210H (Prohibit RMW)	RB7/TB7	RB6/TB6	RB5/TB5	RB4/TB4	RB3/TB3	RB2/TB2	RB1/TB1	RB0/TB0
			R (Receive)/W (Transmission)							
			Undefined							
SC2CR	Serial channel 2 Control register	1211H	RB8	EVEN	PE	OERR	PERR	FERR	SCLKS	IOC
			R	R/W		R (Cleared to 0 when read)			R/W	
			Undefined	0	0	0	0	0	0	0
			Received data bit8	Parity 0: Odd 1: Even	Parity addition 0: Disable 1: Enable	1: Error			0:SCLK2 ↑ 1:SCLK2 ↓	0: Baud rate generator 1: SCLK2 pin input
						Overrun	Parity	Framing		
SC2MOD0	Serial channel 2 Mode0 register	1212H	TB8	CTSE	RXE	WU	SM1	SM0	SC1	SC0
			R/W							
			0	0	0	0	0	0	0	0
			Transfer data bit8	Hand shake 0: CTS disable 1: CTS enable	Receive function 0: Receive disable 1: Receive enable	Wakeup function 0: Disable 1: Enable	Serial transmission mode 00: I/O interface mode 01: 7-bit UART mode 10: 8-bit UART mode 11: 9-bit UART mode		Serial transmission clock (UART) 00: TMRA0 trigger 01: Baud rate generator 10: Internal clock f_{SYS} 11: External clock (SCLK2 input)	
BR2CR	Serial channel 2 Baud rate Control register	1213H	–	BR2ADDE	BR2CK1	BR2CK0	BR2S3	BR2S2	BR2S1	BR2S0
			R/W							
			0	0	0	0	0	0	0	0
			Always write “0”.	+(16 − K) /16 division 0: Disable 1: Enable	00: φT0 01: φT2 10: φT8 11: φT32		Divided frequency setting			
BR2ADD	Serial channel 2 K setting register	1214H					BR2K3	BR2K2	BR2K1	BR2K0
							R/W			
							0	0	0	0
							Sets frequency divisor “K” (divided by N + (16 − K)/16).			
SC2MOD1	Serial channel 2 Mode1 register	1215H	I2S2	FDPX2						
			R/W							
			0	0						
			IDLE2 0: Stop 1: Run	Duplex 0: Half 1: Full						
SIR2CR	IrDA control register 2	1217H	PLSEL	RXSEL	TXEN	RXEN	SIR2WD3	SIR2WD2	SIR2WD1	SIR2WD0
			R/W							
			0	0	0	0	0	0	0	0
			Select transmit pulse width 0: 3/16 1: 1/16	Receive data 0: “H” pulse 1: “L” pulse	Transmit 0: Disable 1: Enable	Receive 0: Disable 1: Enable	Select receive pulse width Set effective pulse width for equal or more than 2x × (value + 1) + 100 ns Can be set: 1 to 14 Can not be set: 0, 15			

(10) I^{2}C Bus/Serial channel (1/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
SBI0CR1	Serial bus interface 0 control register 1	1240H (I^{2}C bus mode) (Prohibit RMW)	BC2	BC1	BC0	ACK		SCK2	SCK1	SCK0/ SWRMON
			W			R/W		W		R/W
			0	0	0	0		0	0	0/1
			Number of transferred bits 000: 8 001: 1 010: 2 011: 3 100: 4 101: 5 110: 6 111: 7			Acknowledge mode 0: Disable 1: Enable		Setting of the divide value “n” 000: 5 001: 6 010: 7 011: 8 100: 9 101: 10 110: 11 111: Reserved		
		1240H (SIO mode) (Prohibit RMW)	SIOS	SIOINH	SIOM1	SIOM0		SCK2	SCK1	SCK0
			W					W		
			0	0	0	0		0	0	0
			Transfer 0: Stop 1: Start	Transfer 0:Continue 1:Abort	Transfer mode 00: 8-bit transmit 01: Reserved 10: 8-bit transmit/receive 11: 8-bit receive			Setting of the divide value “n” 000: 4 001: 5 010: 6 011: 7 100: 8 101: 9 110: 10 111: External clock SCK0		
SBI0DBR	SBI buffer Register	1241H (Prohibit RMW)	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
			R (Receiving)/W (Transmission)							
			Undefined							
I2C0AR	I^{2}CBUS 0 address Register	1242H (Prohibit RMW)	SA6	SA5	SA4	SA3	SA2	SA1	SA0	ALS
			W							
			0	0	0	0	0	0	0	0
			Setting Slave address							address recognition 0:Enable 1:Disable
SBI0SR when Read	Serial bus interface 0 status Register	1243H (I^{2}C bus mode) (Prohibit RMW)	MST	TRX	BB	PIN	AL/ SBIM1	AAS/ SBIM0	AD0/ SWRST1	LRB/ SWRST0
			R/W							
			0	0	0	1	0	0	0	0
			0:Slave 1:Master	0:Receive 1:Transmit	Bus status monitor 0:Free 1:Busy	INTSBE0 interrupt 0:request 1:Cancel	Arbitration lost detection monitor 1:Detect	Slave address match detection monitor 1:Detect	General call detection 1:Detect	Last receive bit monitor 0: “0” 1: “1”
SBI0CR2 when Write	Serial bus interface 0 control Register 2				Start/stop condition generation 0:Stop 1:Start		Operation mode selection 00: Port mode 10: I^{2}C mode 01: SIO mode 11: Reserved		Software reset generate write “10” and “01”, then an internal reset signal is generated.	
SBI0SR when Read	Serial bus interface 0 status Register	1243H (SIO mode) (Prohibit RMW)					SIOF/ SBIM1	SEF/ SBIM0	–	–
							R/W		W	
							0	0	0	0
							Transfer status 0:Stopped 1:In progress	Shift status 0:Stopped 1:In progress		
SBI0CR2 when Write	Serial bus interface 0 control Register 2						Operation mode selection 00: Port mode 10: I^{2}C mode 01: SIO mode 11: Reserved		Always write “0”.	Always write “0”.

I^{2}C Bus/Serial channel (2/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
SBI0BR0	Serial bus interface 0 baud rate register 0	1244H (Prohibit RMW)	–	I2SBI0						
			W	R/W						
			0	0						
			Always write “0”.	IDLE2 0: Stop 1: Run						
SBI0BR1	Serial bus interface 0 baud rate register 1	1245H (Prohibit RMW)	P4EN	–						
			W							
			0	0						
			Internal clock 0: Stop 1: Run	Always write “0”.						

I^2C Bus/Serial channel (3/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
SBI1CR1	Serial bus interface 1 control register 1	1248H (I^2C bus mode) (Prohibit RMW)	BC2	BC1	BC0	ACK		SCK2	SCK1	SCK0/ SWRMON
			W			R/W		W		R/W
			0	0	0	0		0	0	0/1
			Number of transferred bits 000: 8 001: 1 010: 2 011: 3 100: 4 101: 5 110: 6 111: 7			Acknowledge mode 0: Disable 1: Enable		Setting of the divide value “n” 000: 5 001: 6 010: 7 011: 8 100: 9 101: 10 110: 11 111: Reserved		
		1248H (SIO mode) (Prohibit RMW)	SIOS	SIOINH	SIOM1	SIOM0		SCK2	SCK1	SCK0
			W					W		
			0	0	0	0		0	0	0
			Transfer 0: Stop 1: Start	Transfer 0:Continue 1:Abort	Transfer mode 00: 8-bit transmit 01: Reserved 10: 8-bit transmit/receive 11: 8-bit receive			Setting of the divide value “n” 000: 4 001: 5 010: 6 011: 7 100: 8 101: 9 110: 10 111: External clock SCK1		
SBI1DBR	SBI 1 buffer Register	1249H (Prohibit RMW)	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
			R (Receiving)/W (Transmission)							
			Undefined							
I2C1AR	I^2CBUS 1 address Register	124AH (Prohibit RMW)	SA6	SA5	SA4	SA3	SA2	SA1	SA0	ALS
			W							
			0	0	0	0	0	0	0	0
			Setting Slave address							Address recognition 0:Enable 1:Disable
SBI1SR when Read	Serial bus interface 1 status Register	124BH (I^2C bus mode) (Prohibit RMW)	MST	TRX	BB	PIN	AL/ SBIM1	AAS/ SBIM0	AD0/ SWRST1	LRB/ SWRST0
			R/W							
			0	0	0	1	0	0	0	0
			0:Slave 1:Master	0:Receive 1:Transmit	Bus status monitor 0:Free 1:Busy	INTSBE1 interrupt 0:request 1:Cancel	Arbitration lost detection monitor 1:Detect	Slave address match detection monitor 1:Detect	General call detection 1:Detect	Last receive bit monitor 0: “0” 1: “1”
SBI1CR2 when Write	Serial bus interface 1 control Register 2				Start/stop condition generation 0: Stop 1: Start		Operation mode selection 00: Port mode 10: I^2C mode 01: SIO mode 11: Reserved		Software reset generate write “10” and “01”, then an internal reset signal is generated.	
SBI1SR when Read	Serial bus interface 1 status Register	124BH (SIO mode) (Prohibit RMW)					SIOF/ SBIM1	SEF/ SBIM0	–	–
							R/W		W	
							0	0	0	0
							Transfer status 0:Stopped 1:In progress	Shift status 0:Stopped 1:In progress		
SBI1CR2 when Write	Serial bus interface 1 control Register 2						Operation mode selection 00: Port mode 10: I^2C mode 01: SIO mode 11: Reserved		Always write “0”.	Always write “0”.

I^{2}C Bus/Serial channel (4/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
SBI1BR0	Serial bus interface 1 baud rate register 0	124CH (Prohibit RMW)	–	I2SBI1						
			W	R/W						
			0	0						
			Always write “0”.	IDLE2 0: Stop 1: Run						
SBI1BR1	Serial bus interface 1 baud rate register 1	124DH (Prohibit RMW)	P4EN	–						
			W							
			0	0						
			Internal clock 0: Stop 1: Run	Always write “0”.						

(11) AD converter (1/3)

Symbol	Name	Address	7	6	5	4	3	2	1	0
ADMOD0	AD Mode Control register 0	12B8H	EOCF	ADBF	–	–	ITM0	REPEAT	SCAN	ADS
			R		R/W					
			0	0	0	0	0	0	0	0
			AD conversion end flag 0: Conversion in progress 1: Conversion complete	AD conversion busy flag 0: Conversion stopped 1: Conversion in progress	Always write “0”.	Always write “0”.	Interrupt specification in conversion channel fixed repeat mode 0: Every conversion 1: Every fourth conversion	Repeat mode specification 0: Single conversion 1: Repeat conversion mode	Scan mode specification 0: Conversion channel fixed mode 1: Conversion channel scan mode	AD conversion start 0: Don’t care 1: Start conversion Always “0” when read
ADMOD1	AD Mode Control register 1	12B9H	VREFON	I2AD	–	–	ADCH3	ADCH2	ADCH1	ADCH0
			R/W							
			0	0	0	0	0	0	0	0
			VREF application control 0: OFF 1: ON	IDLE2 0: Stop 1: Operate	Always write “0”.	Always write “0”.	Analog input channel selection 0000: AN0 AN0 0001: AN1 AN0 → AN1 0010: AN2 AN0 → AN1 → AN2 0011: AN3 AN0 → AN1 → AN2 → AN3 0100: AN4 AN0 → AN1 → AN2 → AN3 → AN4 0101: AN5 AN0 → AN1 → AN2 → AN3 → AN4 → AN5 0110: AN6 AN0 → AN1 → AN2 → AN3 → AN4 → AN5 → AN6 0111: AN7 AN0 → AN1 → AN2 → AN3 → AN4 → AN5 → AN6 → AN7 1000: AN8 AN0 → AN1 → AN2 → AN3 → AN4 → AN5 → AN6 → AN7 → AN8 1001: AN9 AN0 → AN1 → AN2 → AN3 → AN4 → AN5 → AN6 → AN7 → AN8 → AN9 1010: AN10 AN0 → AN1 → AN2 → AN3 → AN4 → AN5 → AN6 → AN7 → AN8 → AN9 → AN10 1011: AN11 AN0 → AN1 → AN2 → AN3 → AN4 → AN5 → AN6 → AN7 → AN8 → AN9 → AN10 → AN11 1100 to 1111: Reserved			
ADMOD2	AD Mode Control register 2	12BAH	–	–	–	–	–	–	–	ADTRGE
			R/W							
			0	0	0	0	0	0	0	0
			Always write “0”.	Always write “0”.	Always write “0”.	Always write “0”.	Always write “0”.	Always write “0”.	Always write “0”.	AD conversion trigger start control 0: Disable 1: Enable

AD converter (2/3)

Symbol	Name	Address	7	6	5	4	3	2	1	0
ADREG0L	AD result register 0 low	12A0H	ADR01	ADR00						ADR0RF
			R							R
			Undefined							0
ADREG0H	AD result register 0 High	12A1H	ADR09	ADR08	ADR07	ADR06	ADR05	ADR04	ADR03	ADR02
			R							
			Undefined							
ADREG1L	AD result register 1 low	12A2H	ADR11	ADR10						ADR1RF
			R							R
			Undefined							0
ADREG1H	AD result register 1 High	12A3H	ADR19	ADR18	ADR17	ADR16	ADR15	ADR14	ADR13	ADR12
			R							
			Undefined							
ADREG2L	AD result register 2 low	12A4H	ADR21	ADR20						ADR2RF
			R							R
			Undefined							0
ADREG2H	AD result register 2 High	12A5H	ADR29	ADR28	ADR27	ADR26	ADR25	ADR24	ADR23	ADR22
			R							
			Undefined							
ADREG3L	AD result register 3 low	12A6H	ADR31	ADR30						ADR3RF
			R							R
			Undefined							0
ADREG3H	AD result register 3 High	12A7H	ADR39	ADR38	ADR37	ADR36	ADR35	ADR34	ADR33	ADR32
			R							
			Undefined							
ADREG4L	AD result register 4 low	12A8H	ADR41	ADR40						ADR4RF
			R							R
			Undefined							0
ADREG4H	AD result register 4 High	12A9H	ADR49	ADR48	ADR47	ADR46	ADR45	ADR44	ADR43	ADR42
			R							
			Undefined							
ADREG5L	AD result register 5 low	12AAH	ADR51	ADR50						ADR5RF
			R							R
			Undefined							0
ADREG5H	AD result register 5 High	12ABH	ADR59	ADR58	ADR57	ADR56	ADR55	ADR54	ADR53	ADR52
			R							
			Undefined							
ADREG6L	AD result register 6 low	12ACH	ADR61	ADR60						ADR6RF
			R							R
			Undefined							0
ADREG6H	AD result register 6 High	12ADH	ADR69	ADR68	ADR67	ADR66	ADR65	ADR64	ADR63	ADR62
			R							
			Undefined							
ADREG7L	AD result register 7 low	12AEH	ADR71	ADR70						ADR7RF
			R							R
			Undefined							0
ADREG7H	AD result register 7 High	12AFH	ADR79	ADR78	ADR77	ADR76	ADR75	ADR74	ADR73	ADR72
			R							
			Undefined							

AD converter (3/3)

Symbol	Name	Address	7	6	5	4	3	2	1	0
ADREG8L	AD result register 8 low	12B0H	ADR81	ADR80						ADR8RF
			R							R
			Undefined							0
ADREG8H	AD result register 8 High	12B1H	ADR89	ADR88	ADR87	ADR86	ADR85	ADR84	ADR803	ADR82
			R							
			Undefined							
ADREG9L	AD result register 9 low	12B2H	ADR91	ADR90						ADR9RF
			R							R
			Undefined							0
ADREG9H	AD result register 9 High	12B3H	ADR99	ADR98	ADR97	ADR96	ADR95	ADR94	ADR93	ADR92
			R							
			Undefined							
ADREGAL	AD result register A low	12B4H	ADRA1	ADRA0						ADRARF
			R							R
			Undefined							0
ADREGAH	AD result register A High	12B5H	ADRA9	ADRA8	ADRA7	ADRA6	ADRA5	ADRA4	ADRA3	ADRA2
			R							
			Undefined							
ADREGBL	AD result register B low	12B6H	ADRB1	ADRB0						ADRBRF
			R							R
			Undefined							0
ADREGBH	AD result register B High	12B7H	ADRB9	ADRB8	ADRB7	ADRB6	ADRB5	ADRB4	ADRB3	ADRB2
			R							
			Undefined							

(12) Watch dog timer

Symbol	Name	Address	7	6	5	4	3	2	1	0
WDMOD	WDT Mode register	1300H	WDTE	WDTP1	WDTP0		–	I2WDT	RESCR	–
			R/W				R/W			
			1	0	0		0	0	0	0
			WDT control 1: Enable	WDT detection time 00: $2^{15}/f_{SYS}$ 01: $2^{17}/f_{SYS}$ 10: $2^{19}/f_{SYS}$ 11: $2^{21}/f_{SYS}$			Always write “0”.	IDLE2 0: Stop 1: Operate	1: Internally connects WDT out to the reset pin	Always write “0”
WDCR	WDT Control register	1301H (Prohibit RMW)	–							
			W							
			–							
			B1H: WDT disable code 4E: WDT clear code							

(13) Special timer for CLOCK

Symbol	Name	Address	7	6	5	4	3	2	1	0
RTCCR	RTC control register	1310H	–					RTCSEL1	RTCSEL0	RTCRUN
			R/W					R/W		
			0					0	0	0
			Always write “0”					00: $2^{14}/f_S$ 01: $2^{13}/f_S$ 10: $2^{12}/f_S$ 11: $2^{11}/f_S$		0: Stop & Clear 1: RUN

(14) Key-on wake up

Symbol	Name	Address	7	6	5	4	3	2	1	0
KIEN	KEY input enable setting register	13A0H (Prohibit RMW)	KI7EN	KI6EN	KI5EN	KI4EN	KI3EN	KI2EN	KI1EN	KI0EN
			W							
			0	0	0	0	0	0	0	0
			KI7Input 0: Disable 1: Enable	KI6Input 0: Disable 1: Enable	KI5Input 0: Disable 1: Enable	KI4Input 0: Disable 1: Enable	KI3Input 0: Disable 1: Enable	KI2Input 0: Disable 1: Enable	KI1Input 0: Disable 1: Enable	KI0Input 0: Disable 1: Enable
KICR	KEY input Control register	13A1H (Prohibit RMW)	KI7EDGE	KI6DGE	KI5EDGE	KI4EDGE	KI3EDGE	KI2EDGE	KI1EDGE	KI0EDGE
			W							
			0	0	0	0	0	0	0	0
			KI7 edge 0: Rising 1: Falling	KI6 edge 0: Rising 1: Falling	KI5 edge 0: Rising 1: Falling	KI4 edge 0: Rising 1: Falling	KI3 edge 0: Rising 1: Falling	KI2 edge 0: Rising 1: Falling	KI1 edge 0: Rising 1: Falling	KI0 edge 0: Rising 1: Falling

(15) Program patch function (1/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
ROMCMP00	Address compare register 00	1400H (Prohibit RMW)	ROMC07	ROMC06	ROMC05	ROMC04	ROMC03	ROMC02		
			W							
			0	0	0	0	0	0		
			Target ROM address (Lower 6 bit)							
ROMCMP01	Address compare register 01	1401H (Prohibit RMW)	ROMC15	ROMC14	ROMC13	ROMC12	ROMC11	ROMC10	ROMC09	ROMC08
			W							
			0	0	0	0	0	0	0	0
			Target ROM address (Middle 8 bit)							
ROMCMP02	Address compare register 02	1402H (Prohibit RMW)	ROMC23	ROMC22	ROMC21	ROMC20	ROMC19	ROMC18	ROMC17	ROMC16
			W							
			0	0	0	0	0	0	0	0
			Target ROM address (Upper 8 bit)							
ROMSUB0LL	Address substitution register 0LL	1404H (Prohibit RMW)	ROMS07	ROMS06	ROMS05	ROMS04	ROMS03	ROMS02	ROMS01	ROMS00
			W							
			0	0	0	0	0	0	0	0
			Patch code (Lower 8 bits)							
ROMSUB0LH	Address substitution register 0LH	1405H (Prohibit RMW)	ROMS15	ROMS14	ROMS13	ROMS12	ROMS11	ROMS10	ROMS09	ROMS08
			W							
			0	0	0	0	0	0	0	0
			Patch code (Upper 8 bits)							
ROMSUB0HL	Address substitution register 0HL	1406H (Prohibit RMW)	ROMS23	ROMS22	ROMS21	ROMS20	ROMS19	ROMS18	ROMS17	ROMS16
			W							
			0	0	0	0	0	0	0	0
			Patch code (Lower 8 bits)							
ROMSUB0HH	Address substitution register 0HH	1407H (Prohibit RMW)	ROMS31	ROMS30	ROMS29	ROMS28	ROMS27	ROMS26	ROMS25	ROMS24
			W							
			0	0	0	0	0	0	0	0
			Patch code (Upper 8 bits)							
ROMCMP10	Address compare register 10	1408H (Prohibit RMW)	ROMC07	ROMC06	ROMC05	ROMC04	ROMC03	ROMC02		
			W							
			0	0	0	0	0	0		
			Target ROM address (Lower 6 bit)							
ROMCMP11	Address compare register 11	1409H (Prohibit RMW)	ROMC15	ROMC14	ROMC13	ROMC12	ROMC11	ROMC10	ROMC09	ROMC08
			W							
			0	0	0	0	0	0	0	0
			Target ROM address (Middle 8 bit)							
ROMCMP12	Address compare register 12	140AH (Prohibit RMW)	ROMC23	ROMC22	ROMC21	ROMC20	ROMC19	ROMC18	ROMC17	ROMC16
			W							
			0	0	0	0	0	0	0	0
			Target ROM address (Upper 8 bit)							
ROMSUB1LL	Address substitution register 1LL	140CH (Prohibit RMW)	ROMS07	ROMS06	ROMS05	ROMS04	ROMS03	ROMS02	ROMS01	ROMS00
			W							
			0	0	0	0	0	0	0	0
			Patch code (Lower 8 bits)							
ROMSUB1LH	Address substitution register 1LH	140DH (Prohibit RMW)	ROMS15	ROMS14	ROMS13	ROMS12	ROMS11	ROMS10	ROMS09	ROMS08
			W							
			0	0	0	0	0	0	0	0
			Patch code (Upper 8 bits)							
ROMSUB1HL	Address substitution register 1HL	140EH (Prohibit RMW)	ROMS23	ROMS22	ROMS21	ROMS20	ROMS19	ROMS18	ROMS17	ROMS16
			W							
			0	0	0	0	0	0	0	0
			Patch code (Lower 8 bits)							
ROMSUB1HH	Address substitution register 1HH	140FH (Prohibit RMW)	ROMS31	ROMS30	ROMS29	ROMS28	ROMS27	ROMS26	ROMS25	ROMS24
			W							
			0	0	0	0	0	0	0	0
			Patch code (Upper 8 bits)							

Program patch function (2/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
ROMCMP20	Address compare register 20	1410H (Prohibit RMW)	ROMC07	ROMC06	ROMC05	ROMC04	ROMC03	ROMC02		
			W							
			0	0	0	0	0	0		
			Target ROM address (Lower 6 bit)							
ROMCMP21	Address compare register 21	1411H (Prohibit RMW)	ROMC15	ROMC14	ROMC13	ROMC12	ROMC11	ROMC10	ROMC09	ROMC08
			W							
			0	0	0	0	0	0	0	0
			Target ROM address (Middle 8 bit)							
ROMCMP22	Address compare register 22	1412H (Prohibit RMW)	ROMC23	ROMC22	ROMC21	ROMC20	ROMC19	ROMC18	ROMC17	ROMC16
			W							
			0	0	0	0	0	0	0	0
			Target ROM address (Upper 8 bit)							
ROMSUB2LL	Address substitution register 2LL	1414H (Prohibit RMW)	ROMS07	ROMS06	ROMS05	ROMS04	ROMS03	ROMS02	ROMS01	ROMS00
			W							
			0	0	0	0	0	0	0	0
			Patch code (Lower 8 bits)							
ROMSUB2LH	Address substitution register 2LH	1415H (Prohibit RMW)	ROMS15	ROMS14	ROMS13	ROMS12	ROMS11	ROMS10	ROMS09	ROMS08
			W							
			0	0	0	0	0	0	0	0
			Patch code (Upper 8 bits)							
ROMSUB2HL	Address substitution register 2HL	1416H (Prohibit RMW)	ROMS23	ROMS22	ROMS21	ROMS20	ROMS19	ROMS18	ROMS17	ROMS16
			W							
			0	0	0	0	0	0	0	0
			Patch code (Lower 8 bits)							
ROMSUB2HH	Address substitution register 2HH	1417H (Prohibit RMW)	ROMS31	ROMS30	ROMS29	ROMS28	ROMS27	ROMS26	ROMS25	ROMS24
			W							
			0	0	0	0	0	0	0	0
			Patch code (Upper 8 bits)							
ROMCMP30	Address compare register 30	1418H (Prohibit RMW)	ROMC07	ROMC06	ROMC05	ROMC04	ROMC03	ROMC02		
			W							
			0	0	0	0	0	0		
			Target ROM address (Lower 6 bit)							
ROMCMP31	Address compare register 31	1419H (Prohibit RMW)	ROMC15	ROMC14	ROMC13	ROMC12	ROMC11	ROMC10	ROMC09	ROMC08
			W							
			0	0	0	0	0	0	0	0
			Target ROM address (Middle 8 bit)							
ROMCMP32	Address compare register 32	141AH (Prohibit RMW)	ROMC23	ROMC22	ROMC21	ROMC20	ROMC19	ROMC18	ROMC17	ROMC16
			W							
			0	0	0	0	0	0	0	0
			Target ROM address (Upper 8 bit)							
ROMSUB3LL	Address substitution register 3LL	141CH (Prohibit RMW)	ROMS07	ROMS06	ROMS05	ROMS04	ROMS03	ROMS02	ROMS01	ROMS00
			W							
			0	0	0	0	0	0	0	0
			Patch code (Lower 8 bits)							
ROMSUB3LH	Address substitution register 3LH	141DH (Prohibit RMW)	ROMS15	ROMS14	ROMS13	ROMS12	ROMS11	ROMS10	ROMS09	ROMS08
			W							
			0	0	0	0	0	0	0	0
			Patch code (Upper 8 bits)							
ROMSUB3HL	Address substitution register 3HL	141EH (Prohibit RMW)	ROMS23	ROMS22	ROMS21	ROMS20	ROMS19	ROMS18	ROMS17	ROMS16
			W							
			0	0	0	0	0	0	0	0
			Patch code (Lower 8 bits)							
ROMSUB3HH	Address substitution register 3HH	141FH (Prohibit RMW)	ROMS31	ROMS30	ROMS29	ROMS28	ROMS27	ROMS26	ROMS25	ROMS24
			W							
			0	0	0	0	0	0	0	0
			Patch code (Upper 8 bits)							

Program patch function (3/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
ROMCMP40	Address compare register 40	1420H (Prohibit RMW)	ROMC07	ROMC06	ROMC05	ROMC04	ROMC03	ROMC02		
			W							
			0	0	0	0	0	0		
			Target ROM address (Lower 6 bit)							
ROMCMP41	Address compare register 41	1421H (Prohibit RMW)	ROMC15	ROMC14	ROMC13	ROMC12	ROMC11	ROMC10	ROMC09	ROMC08
			W							
			0	0	0	0	0	0	0	0
			Target ROM address (Middle 8 bit)							
ROMCMP42	Address compare register 42	1422H (Prohibit RMW)	ROMC23	ROMC22	ROMC21	ROMC20	ROMC19	ROMC18	ROMC17	ROMC16
			W							
			0	0	0	0	0	0	0	0
			Target ROM address (Upper 8 bit)							
ROMSUB4LL	Address substitution register 4LL	1424H (Prohibit RMW)	ROMS07	ROMS06	ROMS05	ROMS04	ROMS03	ROMS02	ROMS01	ROMS00
			W							
			0	0	0	0	0	0	0	0
			Patch code (Lower 8 bits)							
ROMSUB4LH	Address substitution register 4LH	1425H (Prohibit RMW)	ROMS15	ROMS14	ROMS13	ROMS12	ROMS11	ROMS10	ROMS09	ROMS08
			W							
			0	0	0	0	0	0	0	0
			Patch code (Upper 8 bits)							
ROMSUB4HL	Address substitution register 4HL	1426H (Prohibit RMW)	ROMS23	ROMS22	ROMS21	ROMS20	ROMS19	ROMS18	ROMS17	ROMS16
			W							
			0	0	0	0	0	0	0	0
			Patch code (Lower 8 bits)							
ROMSUB4HH	Address substitution register 4HH	1427H (Prohibit RMW)	ROMS31	ROMS30	ROMS29	ROMS28	ROMS27	ROMS26	ROMS25	ROMS24
			W							
			0	0	0	0	0	0	0	0
			Patch code (Upper 8 bits)							
ROMCMP50	Address compare register 50	1428H (Prohibit RMW)	ROMC07	ROMC06	ROMC05	ROMC04	ROMC03	ROMC02		
			W							
			0	0	0	0	0	0		
			Target ROM address (Lower 6 bit)							
ROMCMP51	Address compare register 51	1429H (Prohibit RMW)	ROMC15	ROMC14	ROMC13	ROMC12	ROMC11	ROMC10	ROMC09	ROMC08
			W							
			0	0	0	0	0	0	0	0
			Target ROM address (Middle 8 bit)							
ROMCMP52	Address compare register 52	142AH (Prohibit RMW)	ROMC23	ROMC22	ROMC21	ROMC20	ROMC19	ROMC18	ROMC17	ROMC16
			W							
			0	0	0	0	0	0	0	0
			Target ROM address (Upper 8 bit)							
ROMSUB5LL	Address substitution register 5LL	142CH (Prohibit RMW)	ROMS07	ROMS06	ROMS05	ROMS04	ROMS03	ROMS02	ROMS01	ROMS00
			W							
			0	0	0	0	0	0	0	0
			Patch code (Lower 8 bits)							
ROMSUB5LH	Address substitution register 5LH	142DH (Prohibit RMW)	ROMS15	ROMS14	ROMS13	ROMS12	ROMS11	ROMS10	ROMS09	ROMS08
			W							
			0	0	0	0	0	0	0	0
			Patch code (Upper 8 bits)							
ROMSUB5HL	Address substitution register 5HL	142EH (Prohibit RMW)	ROMS23	ROMS22	ROMS21	ROMS20	ROMS19	ROMS18	ROMS17	ROMS16
			W							
			0	0	0	0	0	0	0	0
			Patch code (Lower 8 bits)							
ROMSUB5HH	Address substitution register 5HH	142FH (Prohibit RMW)	ROMS31	ROMS30	ROMS29	ROMS28	ROMS27	ROMS26	ROMS25	ROMS24
			W							
			0	0	0	0	0	0	0	0
			Patch code (Upper 8 bits)							

Program patch function (4/4)

Symbol	Name	Address	7	6	5	4	3	2	1	0
ROMCMP60	Address compare register 60	1430H (Prohibit RMW)	ROMC07	ROMC06	ROMC05	ROMC04	ROMC03	ROMC02		
			W							
			0	0	0	0	0	0		
			Target ROM address (Lower 6 bit)							
ROMCMP61	Address compare register 61	1431H (Prohibit RMW)	ROMC15	ROMC14	ROMC13	ROMC12	ROMC11	ROMC10	ROMC09	ROMC08
			W							
			0	0	0	0	0	0	0	0
			Target ROM address (Middle 8 bit)							
ROMCMP62	Address compare register 62	1432H (Prohibit RMW)	ROMC23	ROMC22	ROMC21	ROMC20	ROMC19	ROMC18	ROMC17	ROMC16
			W							
			0	0	0	0	0	0	0	0
			Target ROM address (Upper 8 bit)							
ROMSUB6LL	Address substitution register 6LL	1434H (Prohibit RMW)	ROMS07	ROMS06	ROMS05	ROMS04	ROMS03	ROMS02	ROMS01	ROMS00
			W							
			0	0	0	0	0	0	0	0
			Patch code (Lower 8 bits)							
ROMSUB6LH	Address substitution register 6LH	1435H (Prohibit RMW)	ROMS15	ROMS14	ROMS13	ROMS12	ROMS11	ROMS10	ROMS09	ROMS08
			W							
			0	0	0	0	0	0	0	0
			Patch code (Upper 8 bits)							
ROMSUB6HL	Address substitution register 6HL	1436H (Prohibit RMW)	ROMS23	ROMS22	ROMS21	ROMS20	ROMS19	ROMS18	ROMS17	ROMS16
			W							
			0	0	0	0	0	0	0	0
			Patch code (Lower 8 bits)							
ROMSUB6HH	Address substitution register 6HH	1437H (Prohibit RMW)	ROMS31	ROMS30	ROMS29	ROMS28	ROMS27	ROMS26	ROMS25	ROMS24
			W							
			0	0	0	0	0	0	0	0
			Patch code (Upper 8 bits)							
ROMCMP70	Address compare register 70	1438H (Prohibit RMW)	ROMC07	ROMC06	ROMC05	ROMC04	ROMC03	ROMC02		
			W							
			0	0	0	0	0	0		
			Target ROM address (Lower 6 bit)							
ROMCMP71	Address compare register 71	1439H (Prohibit RMW)	ROMC15	ROMC14	ROMC13	ROMC12	ROMC11	ROMC10	ROMC09	ROMC08
			W							
			0	0	0	0	0	0	0	0
			Target ROM address (Middle 8 bit)							
ROMCMP72	Address compare register 72	143AH (Prohibit RMW)	ROMC23	ROMC22	ROMC21	ROMC20	ROMC19	ROMC18	ROMC17	ROMC16
			W							
			0	0	0	0	0	0	0	0
			Target ROM address (Upper 8 bit)							
ROMSUB7LL	Address substitution register 7LL	143CH (Prohibit RMW)	ROMS07	ROMS06	ROMS05	ROMS04	ROMS03	ROMS02	ROMS01	ROMS00
			W							
			0	0	0	0	0	0	0	0
			Patch code (Lower 8 bits)							
ROMSUB7LH	Address substitution register 7LH	143DH (Prohibit RMW)	ROMS15	ROMS14	ROMS13	ROMS12	ROMS11	ROMS10	ROMS09	ROMS08
			W							
			0	0	0	0	0	0	0	0
			Patch code (Upper 8 bits)							
ROMSUB7HL	Address substitution register 7HL	143EH (Prohibit RMW)	ROMS23	ROMS22	ROMS21	ROMS20	ROMS19	ROMS18	ROMS17	ROMS16
			W							
			0	0	0	0	0	0	0	0
			Patch code (Lower 8 bits)							
ROMSUB7HH	Address substitution register 7HH	143FH (Prohibit RMW)	ROMS31	ROMS30	ROMS29	ROMS28	ROMS27	ROMS26	ROMS25	ROMS24
			W							
			0	0	0	0	0	0	0	0
			Patch code (Upper 8 bits)							

6. Port Section Equivalent Circuit Diagram

- Reading the circuit diagram

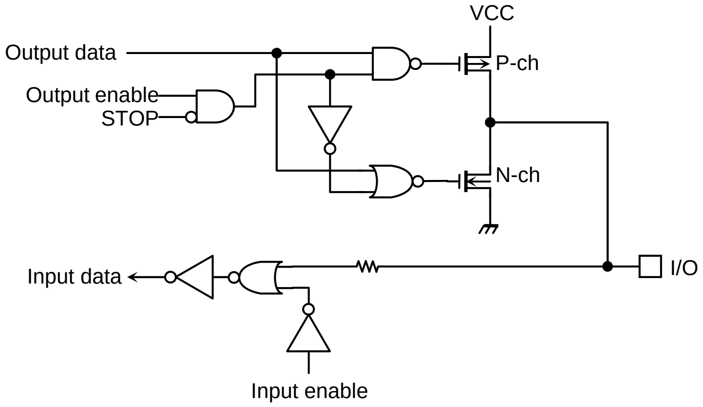
Basically, the gate symbols written are the same as those used for the standard CMOS logic IC [74HCXX] series.

The dedicated signal is described below.

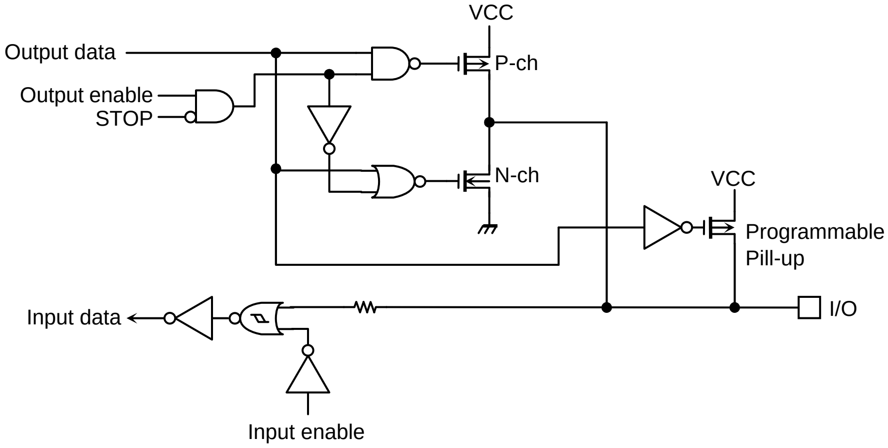
STOP: This signal becomes active “1” when the halt mode setting register is set to the STOP mode and the CPU executes the HALT instruction. When the drive enable bit <DRVE> is set to “1”, however, STOP remains at “0”.

The input protection resistance ranges from several tens of ohms to several hundreds of ohms.

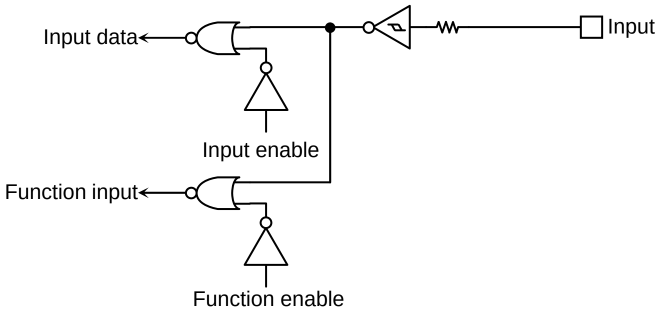
- P0 (D0 to D7), P1 (D8 to D15), P4 (A0 to A7), P5 (A8 to A15), P6 (A16 to A23)



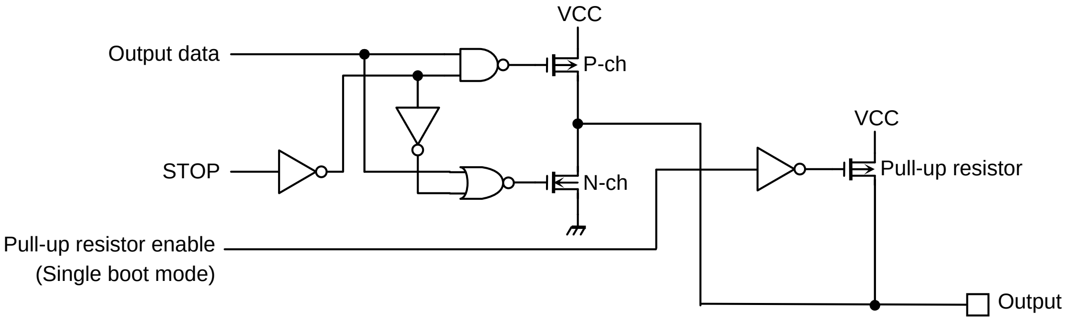
- P70 ($\overline{\text{RD}}$), P71 ($\overline{\text{SRWR}}$), P72 ($\overline{\text{SRLLB}}$), P73 ($\overline{\text{SRLUB}}$)



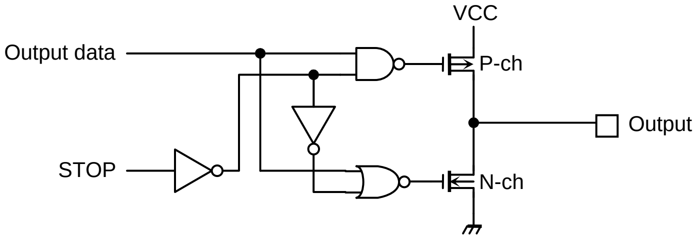
- P74 (INT0), PC1 to PC3 (INT1 to INT3)



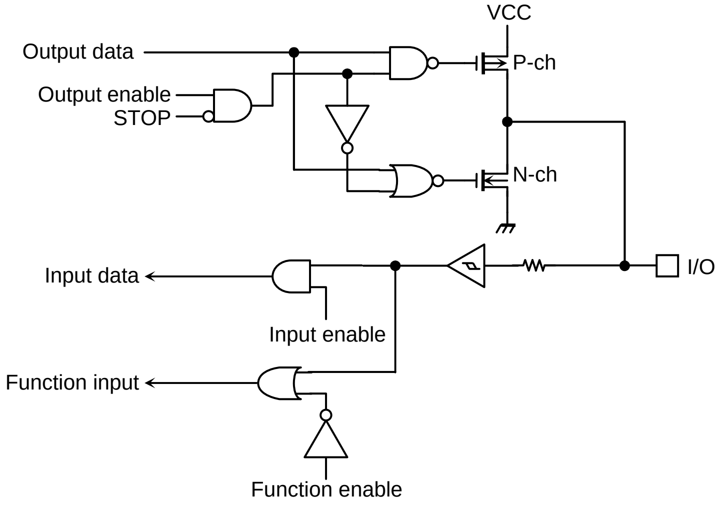
- P80 ($\overline{\text{CS0}}$,TA1OUT, $\overline{\text{BOOT}}$)



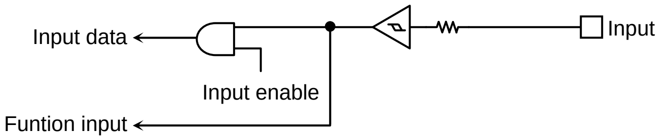
- P81 ($\overline{\text{CS1}}$, TA3OUT), P82 ($\overline{\text{CS2}}$)



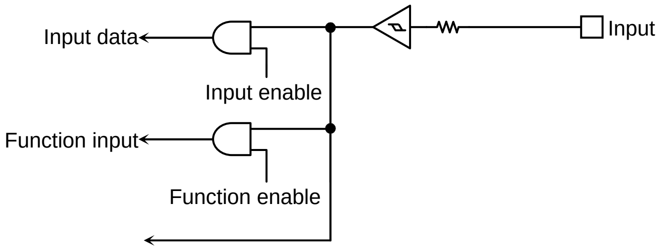
- P83 ($\overline{\text{CS3}}$, $\overline{\text{WAIT}}$,TA5OUT), PD0 (INT4,TB0OUT)



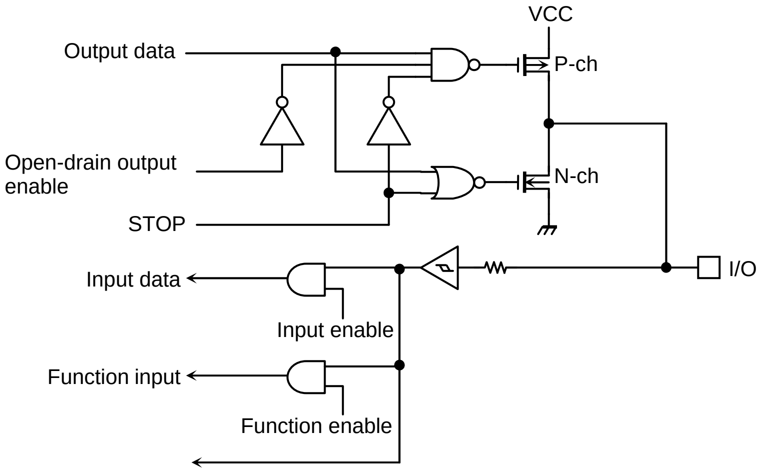
- PC0 (TA0IN)



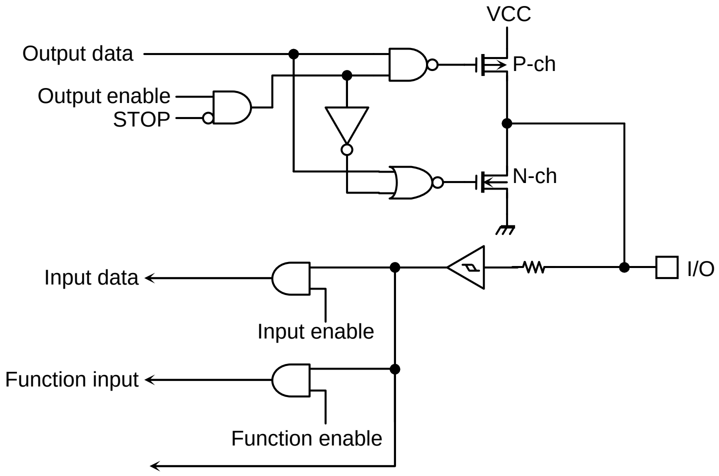
- PD1 (INT5, TB1IN0), PD3 (INT7, TB1OUT0, RXD2)



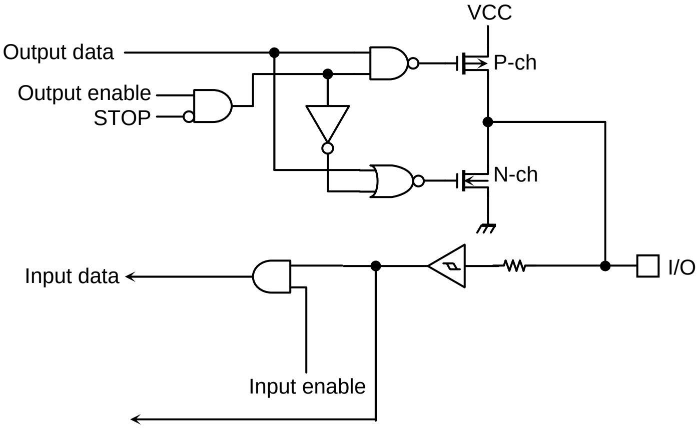
- PD2 (INT6, TB1IN1, TXD2)



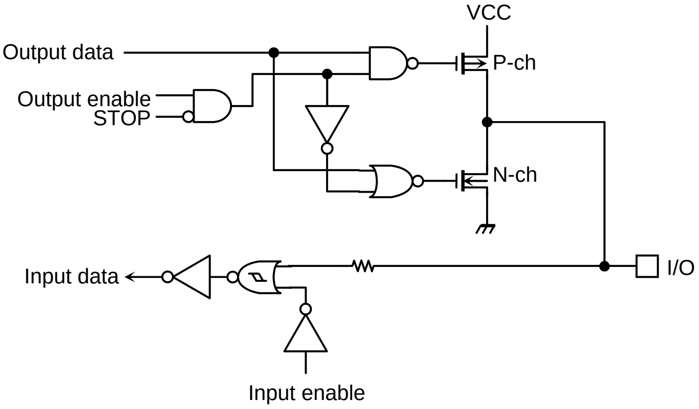
- PD3 (INT7, TB1OUT0, RXD2)



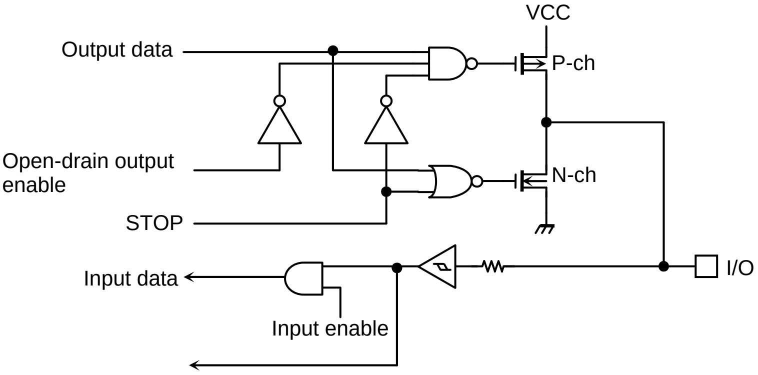
- PD4 (TB1OUT1,SCLK2, $\overline{\text{CTS2}}$), PF1 (RXD0), PF2 (SCLK0, $\overline{\text{CTS0}}$,CLK), PF4 (RXD1, HSSI), PF5 (SCLK1, $\overline{\text{CTS1}}$, HSCLK), PN0 (SCK0), PN3 (SCK1)



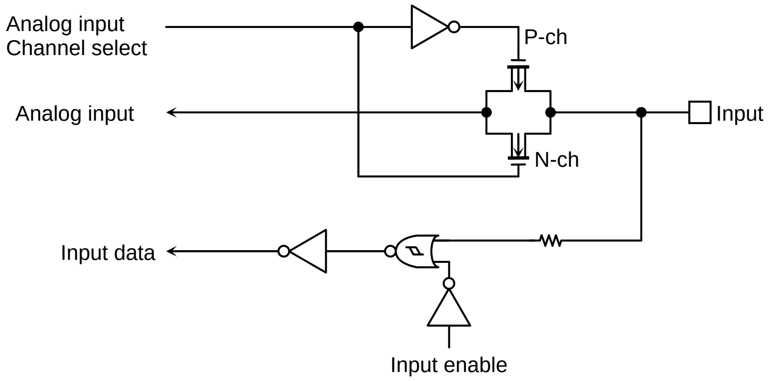
- PF0 (TXD0), PF3 (TXD1, HSSO)



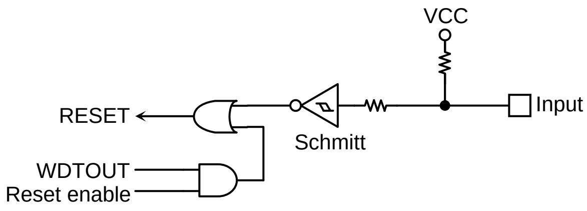
- PN1 (SDA0,SO0), PN2 (SCL0, SI0), PN4 (SDA1, SO1), PN5 (SCL1, SI1)



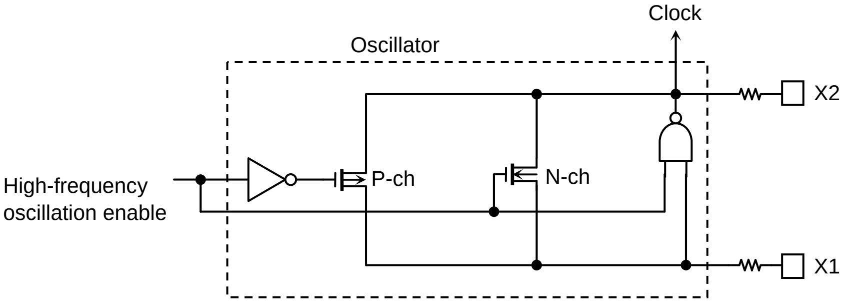
- PG (AN0 to AN7), PL (AN8 to AN11)



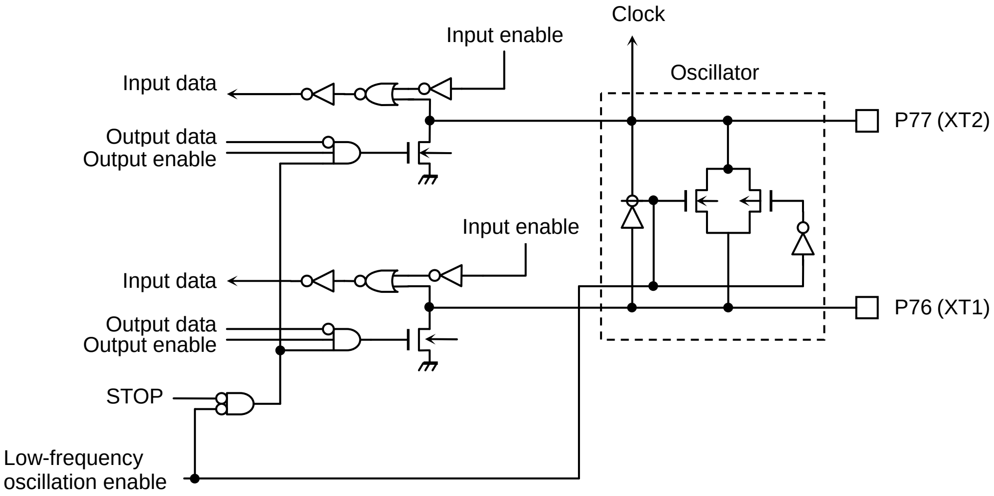
- $\overline{\text{RESET}}$



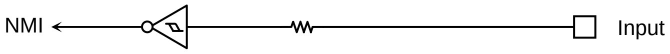
- X1, X2



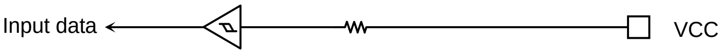
- P76 (XT1), P77 (XT2)



- $\overline{\text{NMI}}$



- AM0 to AM1



7. Points to Note and Restrictions

(1) Notation

a. The notation for built-in/ I/O registers is as follows register symbol <Bit symbol>
(e.g., TA01RUN <TA0RUN> denotes bit TA0RUN of register TA01RUN).

b. 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 bit 3 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)

c. fc, fs, f_{FPH}, f_{SYS} and one state

The clock frequency input on X1 and 2 is called f_{OSCH}. The clock selected by PLLCR0<FCSEL> is called fc.

The clock selected by SYSCR1<SYSCK> is called f_{FPH}. The clock frequency give by f_{FPH} divided by 2 is called f_{SYS}.

One cycle of f_{SYS} is referred to as one state.

(2) Points to note

a. AM0 and AM1 pins

This pin is connected to the V_{CC} or the V_{SS} pin. Do not alter the level when the pin is active.

b. Reserved address areas

The 16-byte area from FFFFF0H to FFFFFFH is reserved as internal area and cannot be used. When using Toshiba's Flash programming service, prepare your ROM data (Hex file) by leaving these 16 bytes blank or setting them all to "FF" and register it with our ROM data entry system.

Moreover, when using an emulator, since it is used for control of an emulator, 64K bytes with arbitrary 16M byte area of use cannot be performed.

c. HALT mode (IDLE1)

When the HALT instruction is executed in IDLE1 mode (in which only the oscillator operates), the internal Special timer for CLOCK operate. When necessary, stop the circuit by setting RTCCR<RTCRUN> to 0, before the HALT instructions is executed.

d. 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.

e. Watchdog timer

The watchdog timer starts operation immediately after a reset is released. When the watchdog timer is not to be used, disable it.

f. 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.

g. 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)).

h. Undefined SFR

The value of an undefined bit in an SFR is undefined when read.

i. POP SR instruction

Please execute the POP SR instruction during DI condition.

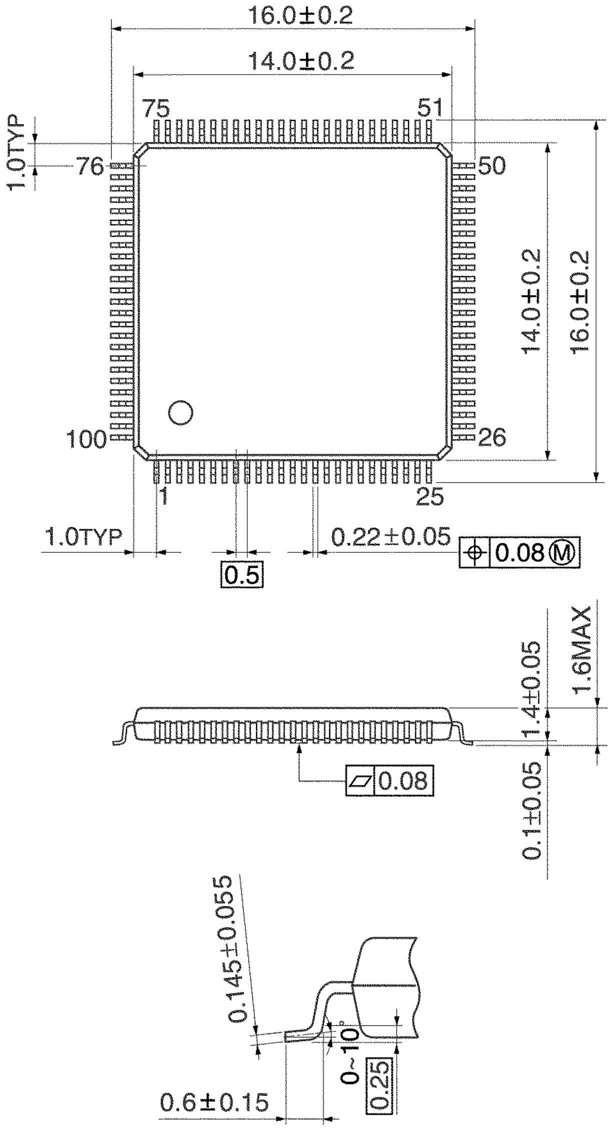
j. Interrupt

When you use interruption, be sure to set "1" as the bit 7 of a SIMC register.

8. Package Dimensions

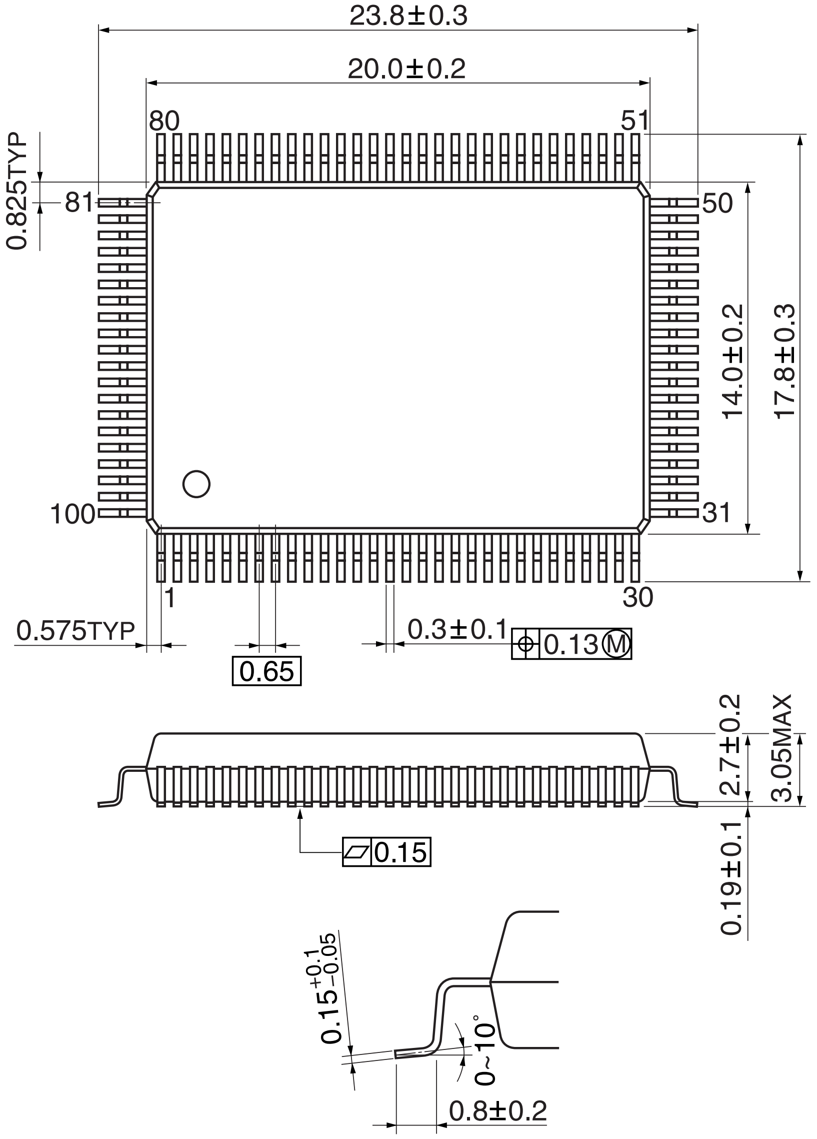
Package Name: LQFP100-P-1414-0.50F

Unit: mm



Package Name: QFP100-P-1420-0.65A

Unit: mm


23.8±0.3
20.0±0.2
80
51
81
50
0.825TYP
14.0±0.2
17.8±0.3
100
31
1
30
0.575TYP
0.65
0.3±0.1
0.13 M
2.7±0.2
3.05MAX
0.15
0.19±0.1
0.15 +0.1 −0.05
0~10°
0.8±0.2