

## Description

---

### Description

The M30201 group of single-chip microcomputers are built using the high-performance silicon gate CMOS process using a M16C/60 Series CPU core. M30201 group is packaged in a 52-pin plastic molded SDIP, or 56-pin plastic molded QFP. These single-chip microcomputers operate using sophisticated instructions featuring a high level of instruction efficiency. With 1M bytes of address space, they are capable of executing instructions at high speed.

The M30201 group includes a wide range of products with different internal memory types and sizes and various package types.

### Features

- Basic machine instructions ..... Compatible with the M16C/60 series
- Memory capacity ..... ROM/RAM (See figure 1.4. ROM expansion.)
- Shortest instruction execution time ..... 100ns ( $f(X_{IN})=10\text{MHz}$ )
- Supply voltage ..... 4.0 to 5.5V ( $f(X_{IN})=10\text{MHz}$ ) :mask ROM version  
2.7 to 5.5V ( $f(X_{IN})=7\text{MHz}$  with software one-wait):mask ROM version  
4.0 to 5.5V ( $f(X_{IN})=10\text{MHz}$ ) :flash memory version
- Interrupts ..... 9 internal and 3 external interrupt sources, 4 software (including key input interrupt)
- Multifunction 16-bit timer ..... Timer A x 1, timer B x 2, timer X x 3
- Clock output
- Serial I/O ..... 1 channel for UART or clock synchronous, 1 for UART
- A-D converter ..... 10 bits X 8 channels (Expandable up to 13 channels)
- Watchdog timer ..... 1 line
- Programmable I/O ..... 43 lines
- LED drive ports ..... 8 ports
- Clock generating circuit ..... 2 built-in clock generation circuits (built-in feedback resistor, and external ceramic or quartz oscillator)

### Applications

Home appliances, Audio, office equipment, Automobiles

Specifications written in this manual are believed to be accurate, but are not guaranteed to be entirely free of error.  
Specifications in this manual may be changed for functional or performance improvements. Please make sure your manual is the latest edition.

## -----Table of Contents-----

Central Processing Unit (CPU) .....	12	Timer .....	37
Reset .....	15	Serial I/O .....	64
Clock Generating Circuit .....	19	A-D Converter .....	78
Protection .....	26	Programmable I/O Ports .....	88
Interrupts .....	27	Electric Characteristics .....	95
Watchdog Timer .....	35	Flash Memory version .....	126

Description

**Pin Configuration**

Figures 1.1 to 1.2 show the pin configurations (top view).

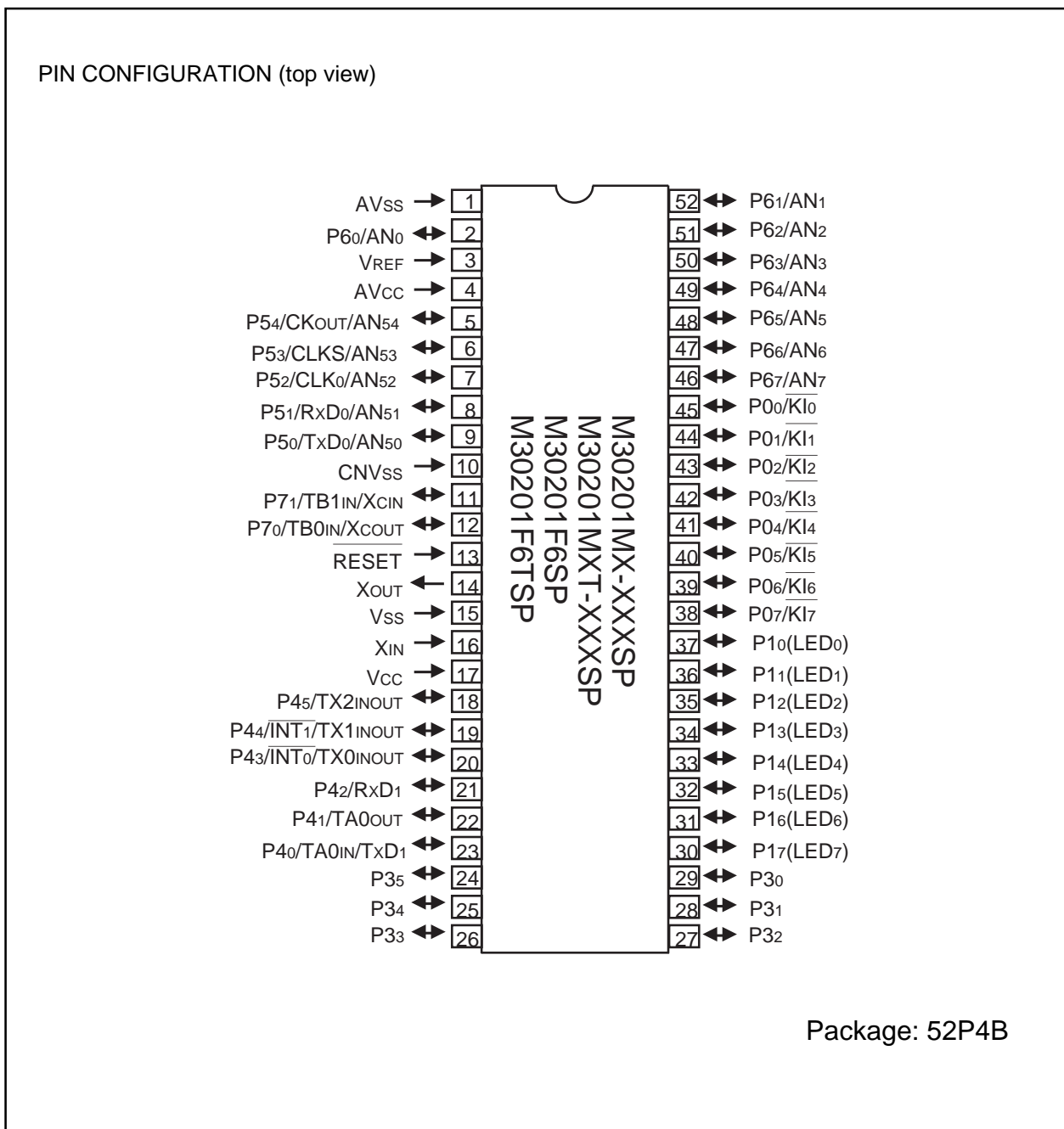


Figure 1.1. Pin configuration for the M30201 group (shrink DIP product) (top view)

Description

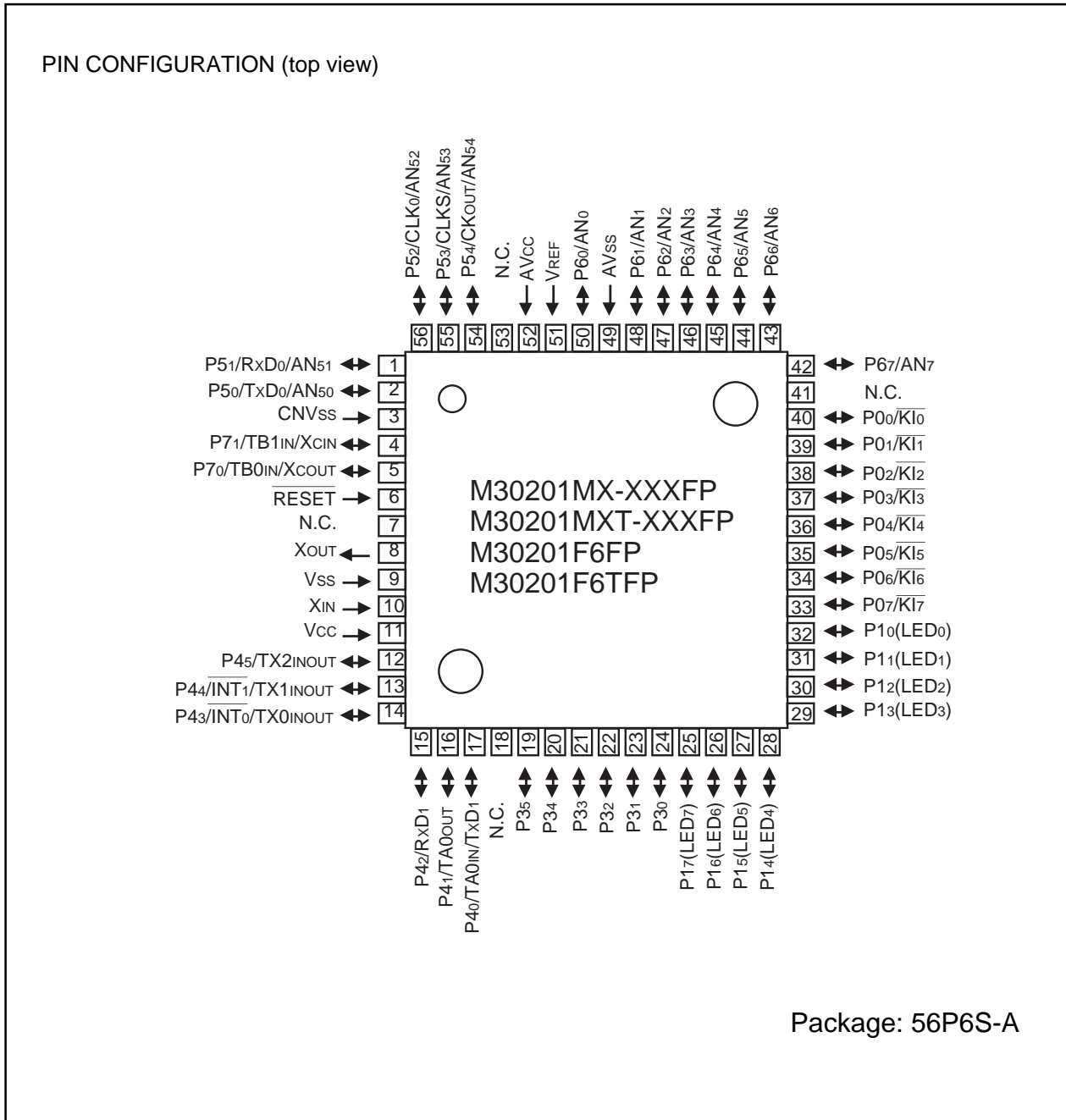
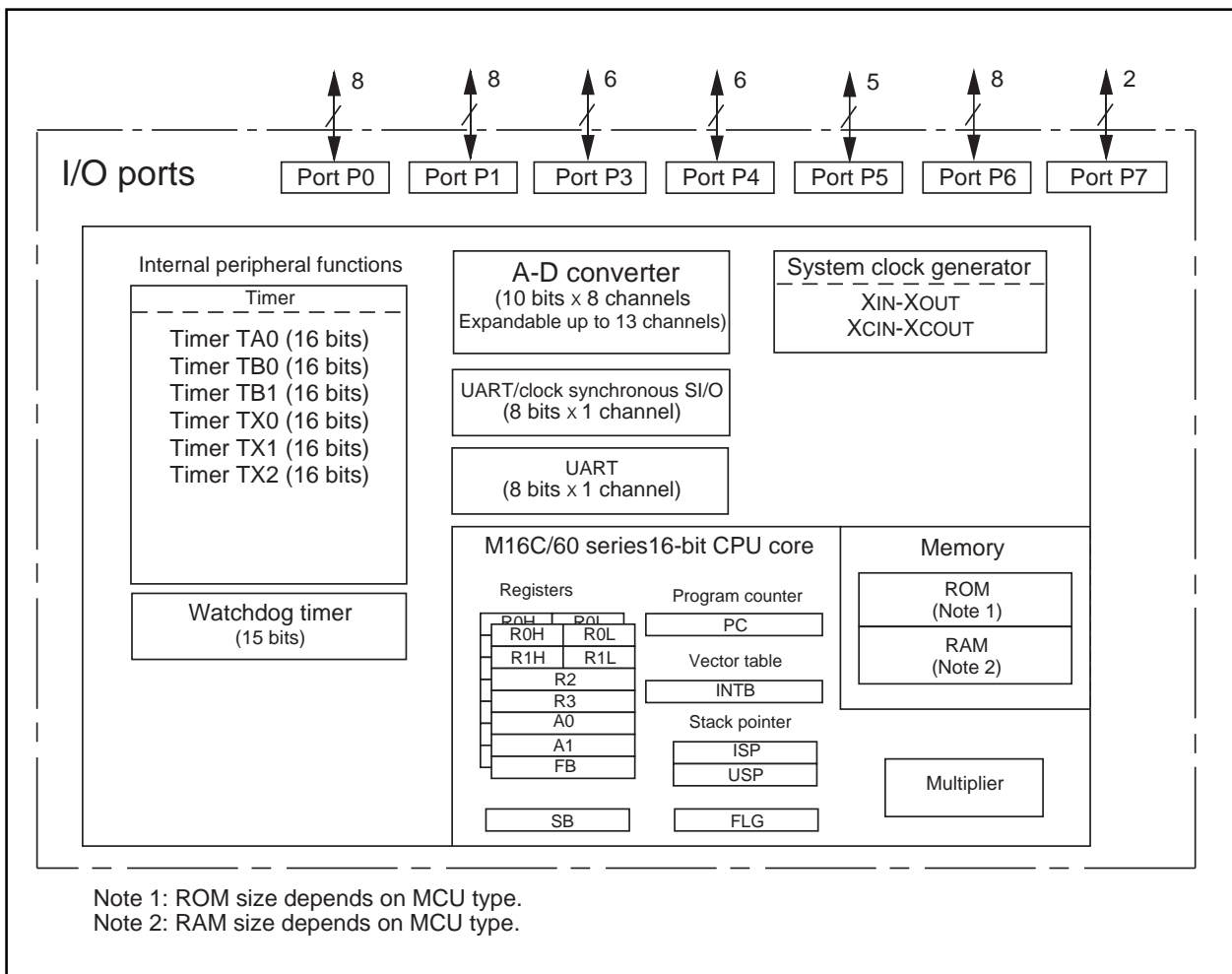


Figure 1.2. Pin configuration for the M30201 group (QFP product) (top view)

Description

**Block Diagram**

Figure 1.3 is a block diagram of the M30201 group.



**Figure 1.3. Block diagram for the M30201 group**

**Description****Performance Outline**

Table 1.1 is performance outline of M30201 group.

**Table 1.1. Performance outline of M30201 group**

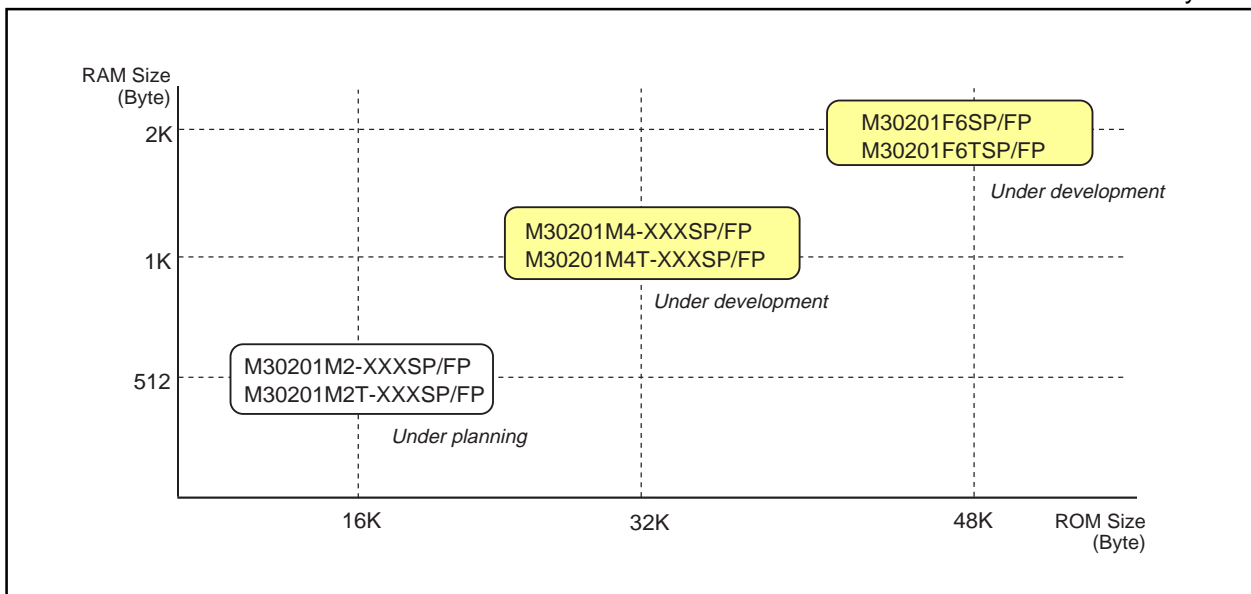
Item		Performance
Number of basic instructions		91 instructions
Shortest instruction execution time		100ns (f(XIN)=10MHz)
Memory capacity	ROM	(See figure 4. ROM expansion.)
	RAM	(See figure 4. ROM expansion.)
I/O port	P0 to P7	43 lines
Multifunction timer	TA0	16 bits x 1
	TB0, TB1	16 bits x 2
	TX0, TX1, TX2	16 bits x 3
Serial I/O	UART0	(UART or clock synchronous) x 1
	UART1	UART x 1
A-D converter		10 bits x 8 channels (Expandable up to 13 channels)
Watchdog timer		15 bits x 1 (with prescaler)
Interrupt		9 internal and 3 external sources, 4 software sources
Clock generating circuit		2 built-in clock generation circuits (built-in feedback resistor, and external ceramic or quartz oscillator)
Supply voltage		4.0 to 5.5V (f(XIN)=10MHz) :mask ROM version 2.7 to 5.5V (f(XIN)=7MHz with software one-wait) :mask ROM version 4.0 to 5.5V (f(XIN)=10MHz) :flash memory version
Power consumption		18mW (f(XIN)=7MHz with software one-wait, Vcc=3V) :mask ROM version 95mW (f(XIN)=10MHz no wait, Vcc=5V) :flash memory version
I/O characteristics	I/O withstand voltage	5V
	Output current	5mA (15mA:LED drive port)
Device configuration		CMOS silicon gate
Package		52-pin plastic mold SDIP 56-pin plastic mold QFP

**Description**

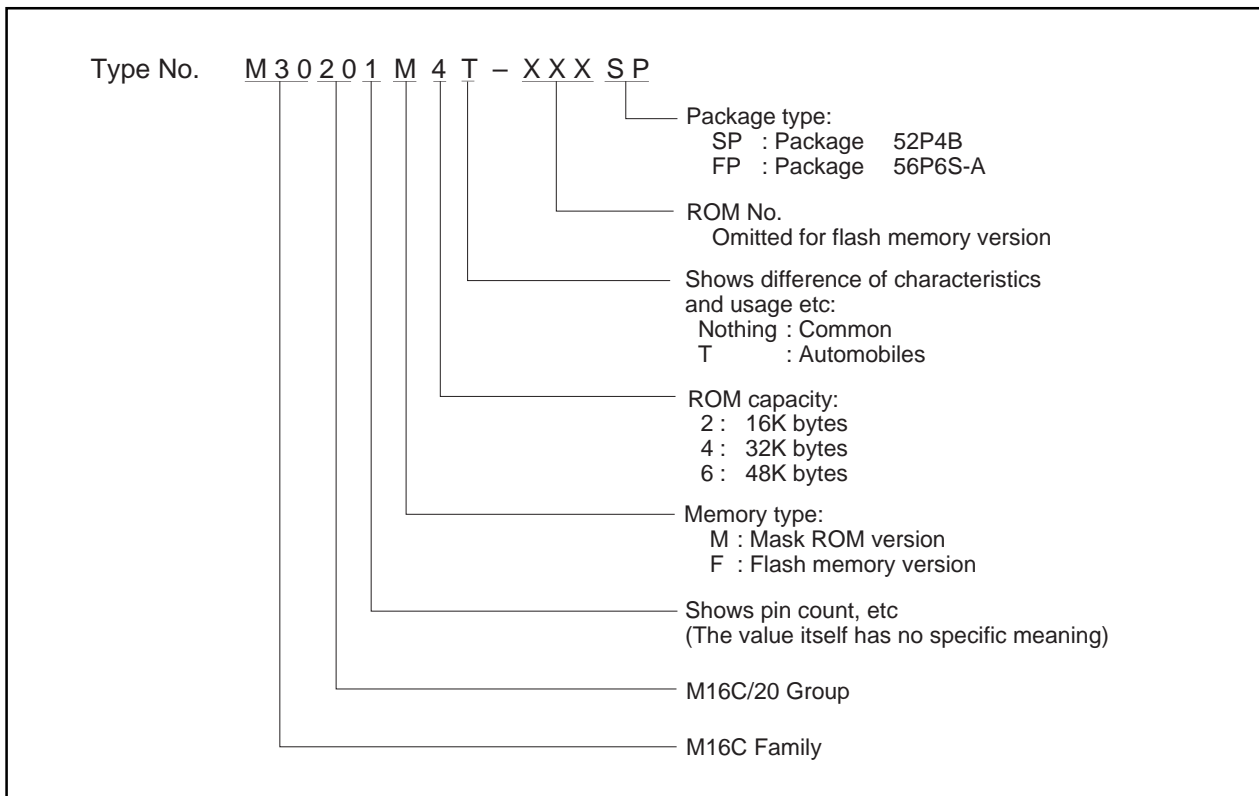
Mitsubishi plans to release the following products in the M30201 group:

- (1) Support for mask ROM version and flash memory version
- (2) ROM capacity
- (3) Package
  - 52P4B : Plastic molded SDIP (mask ROM version and flash memory version)
  - 56P6S-A : Plastic molded QFP (mask ROM version and flash memory version)

July 1998



**Figure 1.4. ROM expansion**



**Figure 1.5. Type No., memory size, and package**

**Pin Description****Pin Description**

Pin name	Signal name	I/O type	Function
Vcc, Vss	Power supply input		Supply 2.7 to 5.5 V to the Vcc pin. Supply 0 V to the Vss pin.
CNVss	CNVss	Input	Connect it to the Vss pin.
RESET	Reset input	Input	A "L" on this input resets the microcomputer.
XIN XOUT	Clock input Clock output	Input Output	These pins are provided for the main clock generating circuit. Connect a ceramic resonator or crystal between the XIN and the XOUT pins. To use an externally derived clock, input it to the XIN pin and leave the XOUT pin open.
AVCC	Analog power supply input		This pin is a power supply input for the A-D converter. Connect it to Vcc.
AVss	Analog power supply input		This pin is a power supply input for the A-D converter. Connect it to Vss.
VREF	Reference voltage input	Input	This pin is a reference voltage input for the A-D converter.
P00 to P07	I/O port P0	Input/output	This is an 8-bit CMOS I/O port. It has an input/output port direction register that allows the user to set each pin for input or output individually. When set for input, the user can specify in units of four bits via software whether or not they are tied to a pull-up resistor.
P10 to P17	I/O port P1	Input/output	This is an 8-bit I/O port equivalent to P0.
P30 to P35	I/O port P3	Input/output	This is a 6-bit I/O port equivalent to P0.
P40 to P45	I/O port P4	Input/output	This is a 6-bit I/O port equivalent to P0. The P40 pin is shared with timer A0 input and serial I/O output TxD1. The P41 pin is shared with timer A0 output. The P42 pin is shared with serial I/O input RxD1. The P43 pin is shared with external interrupt INT0 and timer X0 input/output TX0INOUT. The P44 pin is shared with external interrupt INT1 and timer X1 input/output TX1INOUT. The P45 pin is shared with timer X2 input/output TX2INOUT.
P50 to P54	I/O port P5	Input/output	This is a 5-bit I/O port equivalent to P0. The P50, P51, P52, and P53 pins are shared with serial I/O pins TxD0, RxD0, CLK0, and CLKS. The P54 pin is shared with clock output CLKOUT. Also, these pins are shared with analog input pins AN50 through AN54.
P60 to P67	I/O port P6	Input/output	This is an 8-bit I/O port equivalent to P0. These pins are shared with analog input pins AN0 through AN7.
P70 to P71	I/O port P7	Input/output	This is a 2-bit I/O port equivalent to P0. These pins are used for input/output to and from the oscillator circuit for the clock. Connect a crystal oscillator between the XCIN and the XCOU pins.

## Memory

### Operation of Functional Blocks

The M30201 accommodates certain units in a single chip. These units include ROM and RAM to store instructions and data and the central processing unit (CPU) to execute arithmetic/logic operations. Also included are peripheral units such as timers, serial I/O, A-D converter, and I/O ports.

The following explains each unit.

### Memory

Figure 1.6 is a memory map of the M30201. The address space extends the 1M bytes from address 00000<sub>16</sub> to FFFFF<sub>16</sub>. From FFFFF<sub>16</sub> down is ROM. For example, in the M30201M4-XXXFP, there is 32K bytes of internal ROM from F8000<sub>16</sub> to FFFFF<sub>16</sub>. The vector table for fixed interrupts such as the reset are mapped to FFFDC<sub>16</sub> to FFFFF<sub>16</sub>. The starting address of the interrupt routine is stored here. The address of the vector table for timer interrupts, etc., can be set as desired using the internal register (INTB). See the section on interrupts for details.

From 00400<sub>16</sub> up is RAM. For example, in the M30201M4-XXXFP, there is 1K byte of internal RAM from 00400<sub>16</sub> to 007FF<sub>16</sub>. In addition to storing data, the RAM also stores the stack used when calling subroutines and when interrupts are generated.

The SFR area is mapped to 00000<sub>16</sub> to 003FF<sub>16</sub>. This area accommodates the control registers for peripheral devices such as I/O ports, A-D converter, serial I/O, and timers, etc. Any part of the SFR area that is not occupied is reserved and cannot be used for other purposes.

The special page vector table is mapped to FFE00<sub>16</sub> to FFFDB<sub>16</sub>. If the starting addresses of subroutines or the destination addresses of jumps are stored here, subroutine call instructions and jump instructions can be used as 2-byte instructions, reducing the number of program steps.

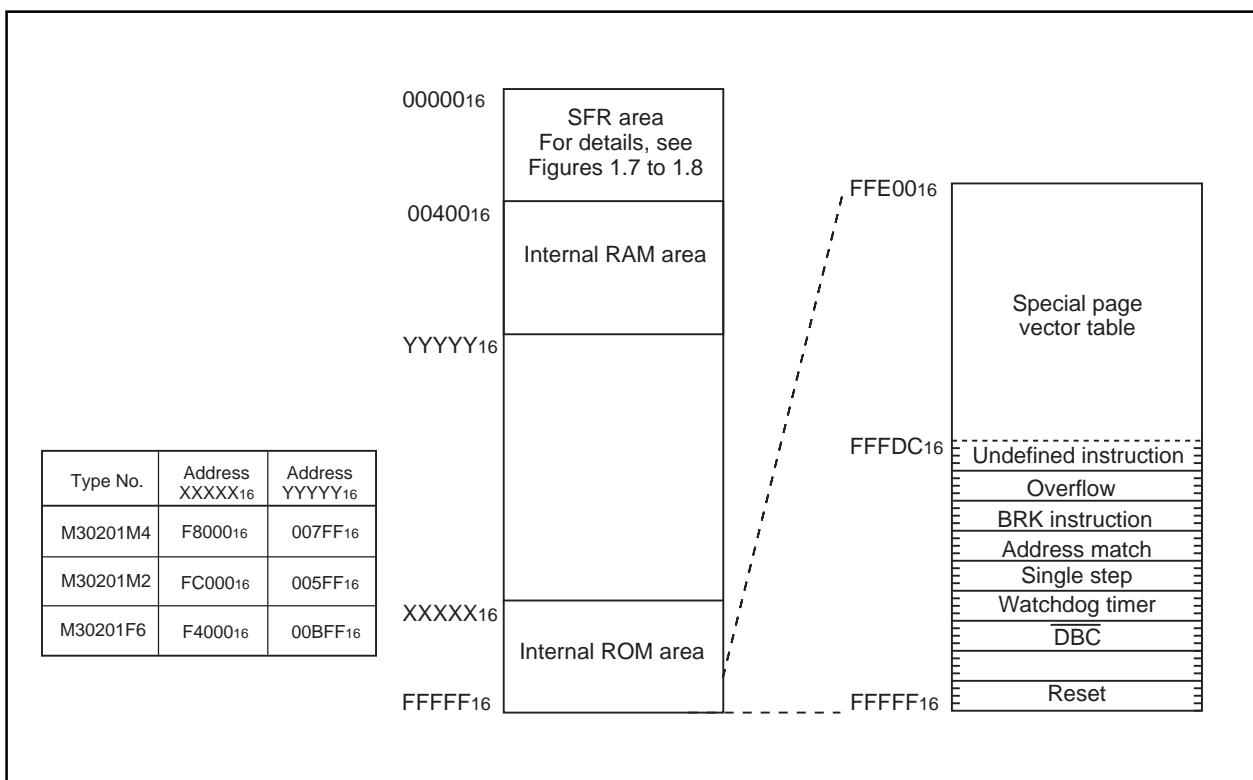


Figure 1.6. Memory map



0000 <sub>16</sub>		0040 <sub>16</sub>	
0001 <sub>16</sub>		0041 <sub>16</sub>	
0002 <sub>16</sub>		0042 <sub>16</sub>	
0003 <sub>16</sub>		0043 <sub>16</sub>	
0004 <sub>16</sub>	Processor mode register 0 (PM0)	0044 <sub>16</sub>	
0005 <sub>16</sub>	Processor mode register 1 (PM1)	0045 <sub>16</sub>	
0006 <sub>16</sub>	System clock control register 0 (CM0)	0046 <sub>16</sub>	
0007 <sub>16</sub>	System clock control register 1 (CM1)	0047 <sub>16</sub>	
0008 <sub>16</sub>		0048 <sub>16</sub>	
0009 <sub>16</sub>	Address match interrupt enable register (AIER)	0049 <sub>16</sub>	
000A <sub>16</sub>	Protect register (PRCR)		
000B <sub>16</sub>		004A <sub>16</sub>	
000C <sub>16</sub>		004B <sub>16</sub>	
000D <sub>16</sub>		004C <sub>16</sub>	
000E <sub>16</sub>	Watchdog timer start register (WDTS)	004D <sub>16</sub>	Key input interrupt control register (KUPIC)
000F <sub>16</sub>	Watchdog timer control register (WDC)	004E <sub>16</sub>	A-D conversion interrupt control register (ADIC)
0010 <sub>16</sub>		004F <sub>16</sub>	
0011 <sub>16</sub>	Address match interrupt register 0 (RMAD0)	0050 <sub>16</sub>	
0012 <sub>16</sub>		0051 <sub>16</sub>	UART0 transmit interrupt control register (S0TIC)
0013 <sub>16</sub>		0052 <sub>16</sub>	UART0 receive interrupt control register (S0RIC)
0014 <sub>16</sub>		0053 <sub>16</sub>	UART1 transmit interrupt control register (S1TIC)
0015 <sub>16</sub>	Address match interrupt register 1 (RMAD1)	0054 <sub>16</sub>	UART1 receive interrupt control register (S1RIC)
0016 <sub>16</sub>		0055 <sub>16</sub>	Timer A0 interrupt control register (TA0IC)
0017 <sub>16</sub>		0056 <sub>16</sub>	Timer X0 interrupt control register (TX0IC)
0018 <sub>16</sub>		0057 <sub>16</sub>	Timer X1 interrupt control register (TX1IC)
0019 <sub>16</sub>		0058 <sub>16</sub>	Timer X2 interrupt control register (TX2IC)
001A <sub>16</sub>		0059 <sub>16</sub>	
001B <sub>16</sub>		005A <sub>16</sub>	Timer B0 interrupt control register (TB0IC)
001C <sub>16</sub>		005B <sub>16</sub>	Timer B1 interrupt control register (TB1IC)
001D <sub>16</sub>		005C <sub>16</sub>	
001E <sub>16</sub>		005D <sub>16</sub>	INT0 interrupt control register (INT0IC)
001F <sub>16</sub>		005E <sub>16</sub>	INT1 interrupt control register (INT1IC)
0020 <sub>16</sub>		005F <sub>16</sub>	
0021 <sub>16</sub>			
0022 <sub>16</sub>			
0023 <sub>16</sub>			
0024 <sub>16</sub>			
0025 <sub>16</sub>			
0026 <sub>16</sub>			
0027 <sub>16</sub>			
0028 <sub>16</sub>			
0029 <sub>16</sub>			
002A <sub>16</sub>			
002B <sub>16</sub>			
002C <sub>16</sub>			
002D <sub>16</sub>			
002E <sub>16</sub>			
002F <sub>16</sub>			
0030 <sub>16</sub>			
0031 <sub>16</sub>			
0032 <sub>16</sub>			
0033 <sub>16</sub>			
0034 <sub>16</sub>			
0035 <sub>16</sub>			
0036 <sub>16</sub>			
0037 <sub>16</sub>			
0038 <sub>16</sub>			
0039 <sub>16</sub>			
003A <sub>16</sub>			
003B <sub>16</sub>			
003C <sub>16</sub>			
003D <sub>16</sub>			
003E <sub>16</sub>			
003F <sub>16</sub>			

Figure 1.7. Location of peripheral unit control registers (1)

0380 <sup>16</sup>	Count start flag (TABSR)	03C0 <sup>16</sup>	A-D register 0 (AD0)
0381 <sup>16</sup>	Clock prescaler reset flag (CPSRF)	03C1 <sup>16</sup>	
0382 <sup>16</sup>	One-shot start flag (ONSF)	03C2 <sup>16</sup>	A-D register 1 (AD1)
0383 <sup>16</sup>	Trigger select register (TRGSR)	03C3 <sup>16</sup>	
0384 <sup>16</sup>	Up-down flag (UDF)	03C4 <sup>16</sup>	A-D register 2 (AD2)
0385 <sup>16</sup>		03C5 <sup>16</sup>	
0386 <sup>16</sup>	Timer A0 (TA0)	03C6 <sup>16</sup>	A-D register 3 (AD3)
0387 <sup>16</sup>		03C7 <sup>16</sup>	
0388 <sup>16</sup>	Timer X0 (TX0)	03C8 <sup>16</sup>	A-D register 4 (AD4)
0389 <sup>16</sup>		03C9 <sup>16</sup>	
038A <sup>16</sup>	Timer X1 (TX1)	03CA <sup>16</sup>	A-D register 5 (AD5)
038B <sup>16</sup>		03CB <sup>16</sup>	
038C <sup>16</sup>	Timer X2 (TX2)	03CC <sup>16</sup>	A-D register 6 (AD6)
038D <sup>16</sup>		03CD <sup>16</sup>	
038E <sup>16</sup>	Clock divided counter (CDC)	03CE <sup>16</sup>	A-D register 7 (AD7)
038F <sup>16</sup>		03CF <sup>16</sup>	
0390 <sup>16</sup>	Timer B0 (TB0)	03D0 <sup>16</sup>	
0391 <sup>16</sup>		03D1 <sup>16</sup>	
0392 <sup>16</sup>	Timer B1 (TB1)	03D2 <sup>16</sup>	
0393 <sup>16</sup>		03D3 <sup>16</sup>	
0394 <sup>16</sup>		03D4 <sup>16</sup>	A-D control register 2 (ADCON2)
0395 <sup>16</sup>		03D5 <sup>16</sup>	
0396 <sup>16</sup>	Timer A0 mode register (TA0MR)	03D6 <sup>16</sup>	A-D control register 0 (ADCON0)
0397 <sup>16</sup>	Timer X0 mode register (TX0MR)	03D7 <sup>16</sup>	A-D control register 1 (ADCON1)
0398 <sup>16</sup>	Timer X1 mode register (TX1MR)	03D8 <sup>16</sup>	
0399 <sup>16</sup>	Timer X2 mode register (TX2MR)	03D9 <sup>16</sup>	
039A <sup>16</sup>		03DA <sup>16</sup>	
039B <sup>16</sup>	Timer B0 mode register (TB0MR)	03DB <sup>16</sup>	
039C <sup>16</sup>	Timer B1 mode register (TB1MR)	03DC <sup>16</sup>	
039D <sup>16</sup>		03DD <sup>16</sup>	
039E <sup>16</sup>		03DE <sup>16</sup>	
039F <sup>16</sup>		03DF <sup>16</sup>	
03A0 <sup>16</sup>	UART0 transmit/receive mode register (U0MR)	03E0 <sup>16</sup>	Port P0 (P0)
03A1 <sup>16</sup>	UART0 bit rate generator (U0BRG)	03E1 <sup>16</sup>	Port P1 (P1)
03A2 <sup>16</sup>	UART0 transmit buffer register (U0TB)	03E2 <sup>16</sup>	Port P0 direction register (PD0)
03A3 <sup>16</sup>		03E3 <sup>16</sup>	Port P1 direction register (PD1)
03A4 <sup>16</sup>	UART0 transmit/receive control register 0 (U0C0)	03E4 <sup>16</sup>	Port P2 (P2) (Reserved)
03A5 <sup>16</sup>	UART0 transmit/receive control register 1 (U0C1)	03E5 <sup>16</sup>	Port P3 (P3)
03A6 <sup>16</sup>	UART0 receive buffer register (U0RB)	03E6 <sup>16</sup>	Port P2 direction register (PD2) (Reserved)
03A7 <sup>16</sup>		03E7 <sup>16</sup>	Port P3 direction register (PD3)
03A8 <sup>16</sup>	UART1 transmit/receive mode register (U1MR)	03E8 <sup>16</sup>	Port P4 (P4)
03A9 <sup>16</sup>	UART1 bit rate generator (U1BRG)	03E9 <sup>16</sup>	Port P5 (P5)
03AA <sup>16</sup>	UART1 transmit buffer register (U1TB)	03EA <sup>16</sup>	Port P4 direction register (PD4)
03AB <sup>16</sup>		03EB <sup>16</sup>	Port P5 direction register (PD5)
03AC <sup>16</sup>	UART1 transmit/receive control register 0 (U1C0)	03EC <sup>16</sup>	Port P6 (P6)
03AD <sup>16</sup>	UART1 transmit/receive control register 1 (U1C1)	03ED <sup>16</sup>	Port P7 (P7)
03AE <sup>16</sup>	UART1 receive buffer register (U1RB)	03EE <sup>16</sup>	Port P6 direction register (PD6)
03AF <sup>16</sup>		03EF <sup>16</sup>	Port P7 direction register (PD7)
03B0 <sup>16</sup>	UART transmit/receive control register 2 (UCON)	03F0 <sup>16</sup>	
03B1 <sup>16</sup>		03F1 <sup>16</sup>	
03B2 <sup>16</sup>		03F2 <sup>16</sup>	
03B3 <sup>16</sup>		03F3 <sup>16</sup>	
03B4 <sup>16</sup>	Flash memory control register 0 (FCON0) (Note)	03F4 <sup>16</sup>	
03B5 <sup>16</sup>	Flash memory control register 1 (FCON1) (Note)	03F5 <sup>16</sup>	
03B6 <sup>16</sup>	Flash command register (FCMD) (Note)	03F6 <sup>16</sup>	
03B7 <sup>16</sup>		03F7 <sup>16</sup>	
03B8 <sup>16</sup>		03F8 <sup>16</sup>	
03B9 <sup>16</sup>		03F9 <sup>16</sup>	
03BA <sup>16</sup>		03FA <sup>16</sup>	
03BB <sup>16</sup>		03FB <sup>16</sup>	
03BC <sup>16</sup>		03FC <sup>16</sup>	Pull-up control register 0 (PUR0)
03BD <sup>16</sup>		03FD <sup>16</sup>	Pull-up control register 1 (PUR1)
03BE <sup>16</sup>		03FE <sup>16</sup>	Port P1 drive control register (DRR)
03BF <sup>16</sup>		03FF <sup>16</sup>	

Note: This register is only exist in flash memory version.

Figure 1.8. Location of peripheral unit control registers (2)

## Central Processing Unit (CPU)

The CPU has a total of 13 registers shown in Figure 1.9. Seven of these registers (R0, R1, R2, R3, A0, A1, and FB) come in two sets; therefore, these have two register banks.

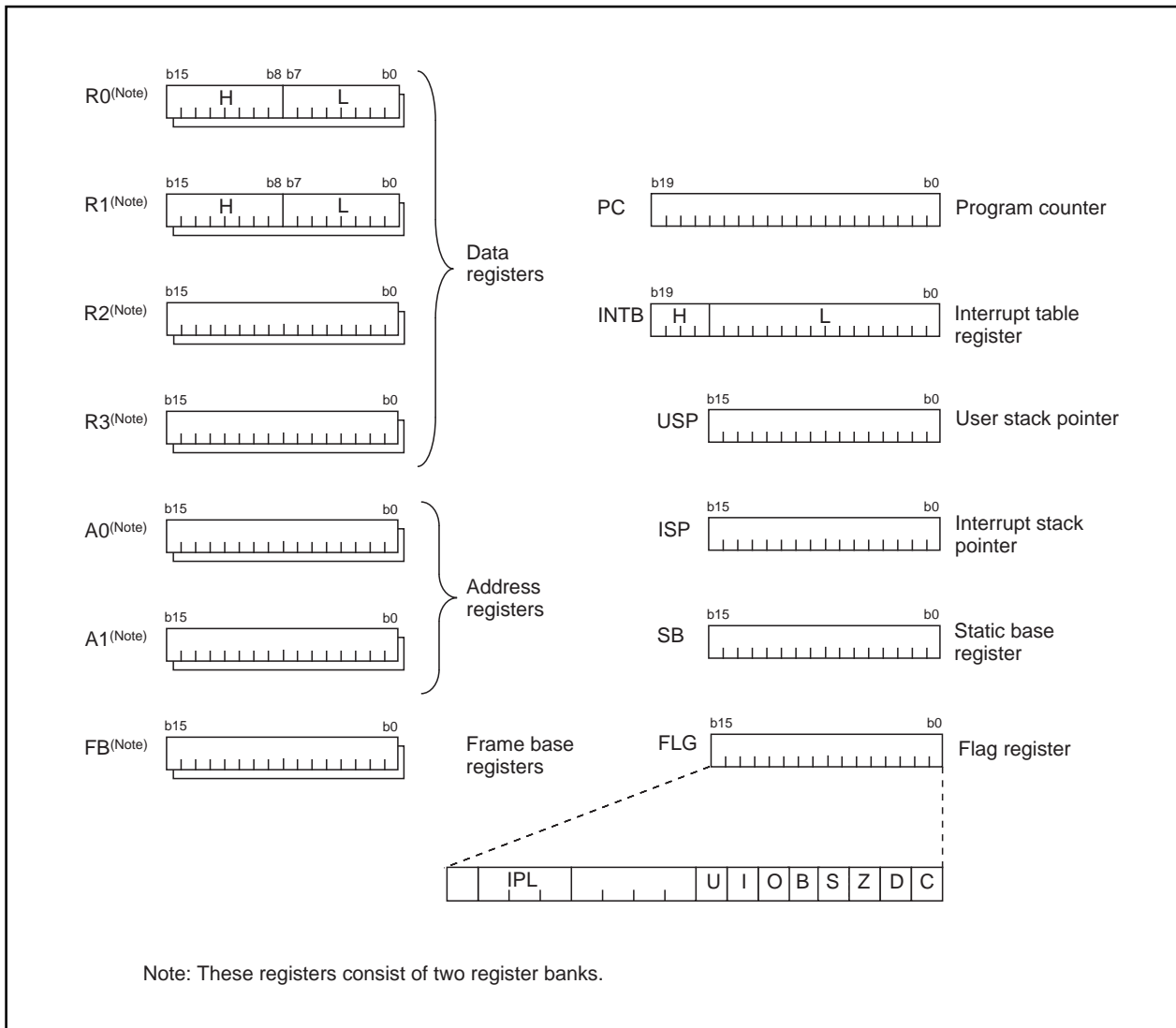


Figure 1.9. Central processing unit register

### (1) Data registers (R0, R0H, R0L, R1, R1H, R1L, R2, and R3)

Data registers (R0, R1, R2, and R3) are configured with 16 bits, and are used primarily for transfer and arithmetic/logic operations.

Registers R0 and R1 each can be used as separate 8-bit data registers, high-order bits as (R0H, R1H), and low-order bits as (R0L, R1L). In some instructions, registers R2 and R0, as well as R3 and R1 can use as 32-bit data registers (R2R0, R3R1).

### (2) Address registers (A0 and A1)

Address registers (A0 and A1) are configured with 16 bits, and have functions equivalent to those of data registers. These registers can also be used for address register indirect addressing and address register relative addressing.

In some instructions, registers A1 and A0 can be combined for use as a 32-bit address register (A1A0).

### (3) Frame base register (FB)

Frame base register (FB) is configured with 16 bits, and is used for FB relative addressing.

### (4) Program counter (PC)

Program counter (PC) is configured with 20 bits, indicating the address of an instruction to be executed.

### (5) Interrupt table register (INTB)

Interrupt table register (INTB) is configured with 20 bits, indicating the start address of an interrupt vector table.

### (6) Stack pointer (USP/ISP)

Stack pointer comes in two types: user stack pointer (USP) and interrupt stack pointer (ISP), each configured with 16 bits.

Your desired type of stack pointer (USP or ISP) can be selected by a stack pointer select flag (U flag).

This flag is located at the position of bit 7 in the flag register (FLG).

### (7) Static base register (SB)

Static base register (SB) is configured with 16 bits, and is used for SB relative addressing.

### (8) Flag register (FLG)

Flag register (FLG) is configured with 11 bits, each bit is used as a flag. Figure 1.10 shows the flag register (FLG). The following explains the function of each flag:

- **Bit 0: Carry flag (C flag)**

This flag retains a carry, borrow, or shift-out bit that has occurred in the arithmetic/logic unit.

- **Bit 1: Debug flag (D flag)**

This flag enables a single-step interrupt.

When this flag is "1", a single-step interrupt is generated after instruction execution. This flag is cleared to "0" when the interrupt is acknowledged.

- **Bit 2: Zero flag (Z flag)**

This flag is set to "1" when an arithmetic operation resulted in 0; otherwise, cleared to "0".

- **Bit 3: Sign flag (S flag)**

This flag is set to "1" when an arithmetic operation resulted in a negative value; otherwise, cleared to "0".

- **Bit 4: Register bank select flag (B flag)**

This flag chooses a register bank. Register bank 0 is selected when this flag is "0"; register bank 1 is selected when this flag is "1".

- **Bit 5: Overflow flag (O flag)**

This flag is set to "1" when an arithmetic operation resulted in overflow; otherwise, cleared to "0".

- **Bit 6: Interrupt enable flag (I flag)**

This flag enables a maskable interrupt.

An interrupt is disabled when this flag is "0", and is enabled when this flag is "1". This flag is cleared to "0" when the interrupt is acknowledged.

• **Bit 7: Stack pointer select flag (U flag)**

Interrupt stack pointer (ISP) is selected when this flag is “0” ; user stack pointer (USP) is selected when this flag is “1”.

This flag is cleared to “0” when a hardware interrupt is acknowledged or an INT instruction of software interrupt Nos. 0 to 31 is executed.

• **Bits 8 to 11: Reserved area**

• **Bits 12 to 14: Processor interrupt priority level (IPL)**

Processor interrupt priority level (IPL) is configured with three bits, for specification of up to eight processor interrupt priority levels from level 0 to level 7.

If a requested interrupt has priority greater than the processor interrupt priority level (IPL), the interrupt is enabled.

• **Bit 15: Reserved area**

The C, Z, S, and O flags are changed when instructions are executed. See the software manual for details.

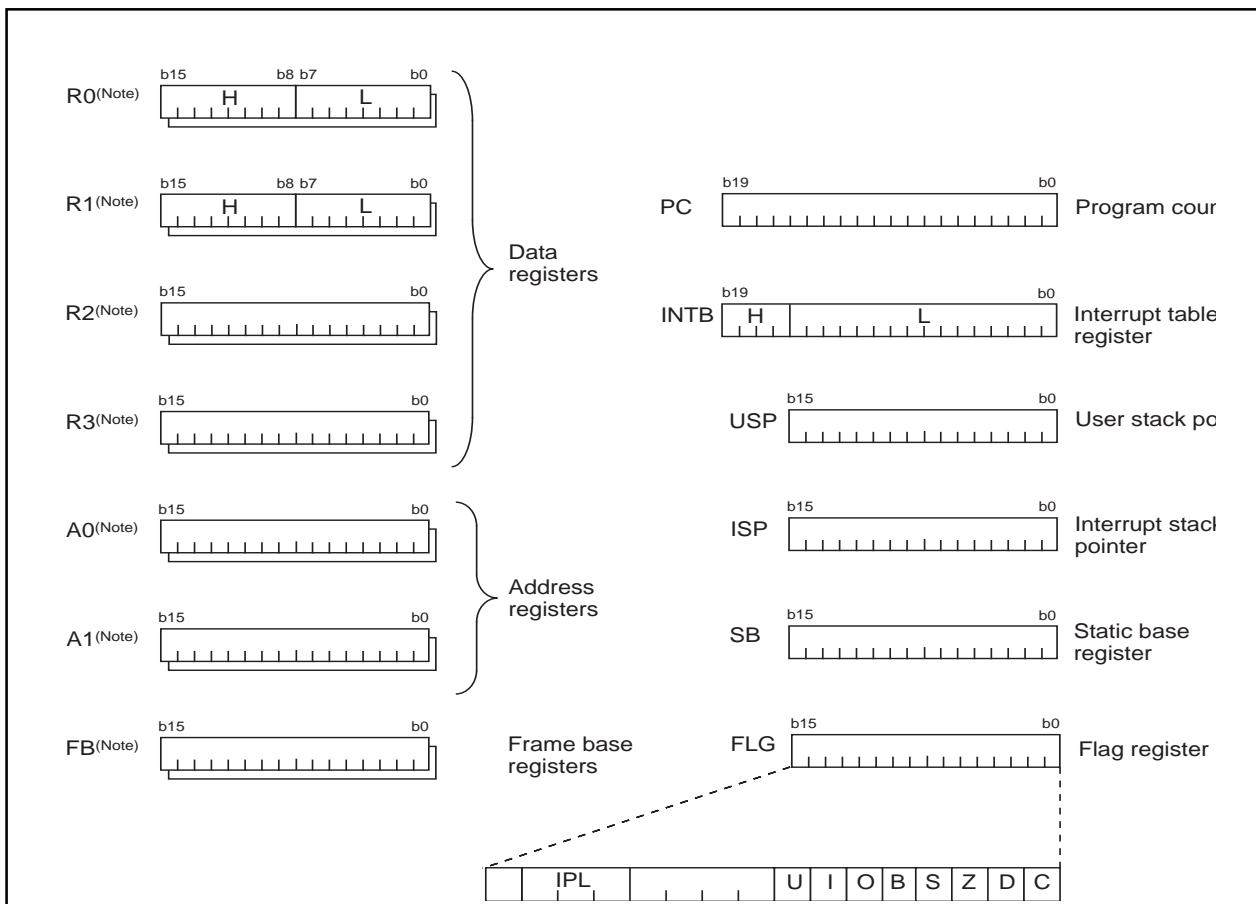


Figure 1.10. Flag register (FLG)

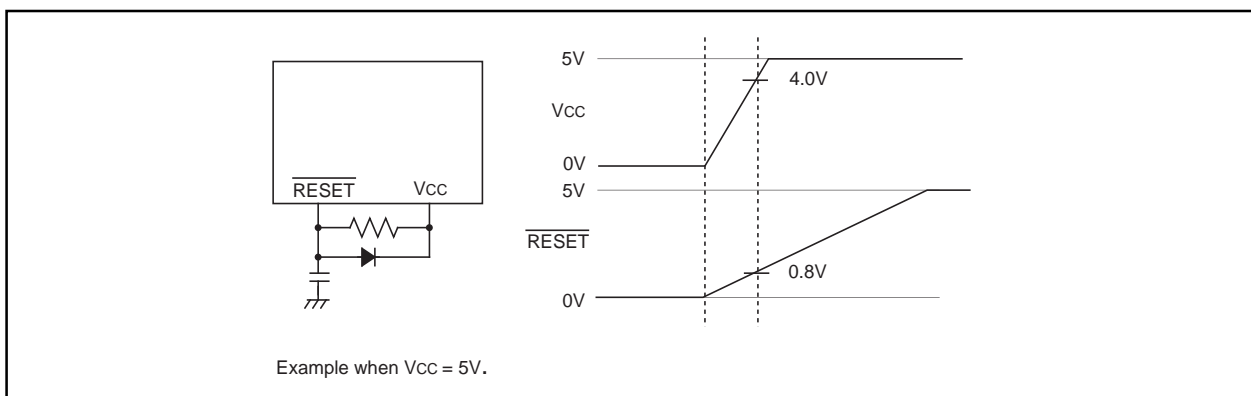
## Reset

### Reset

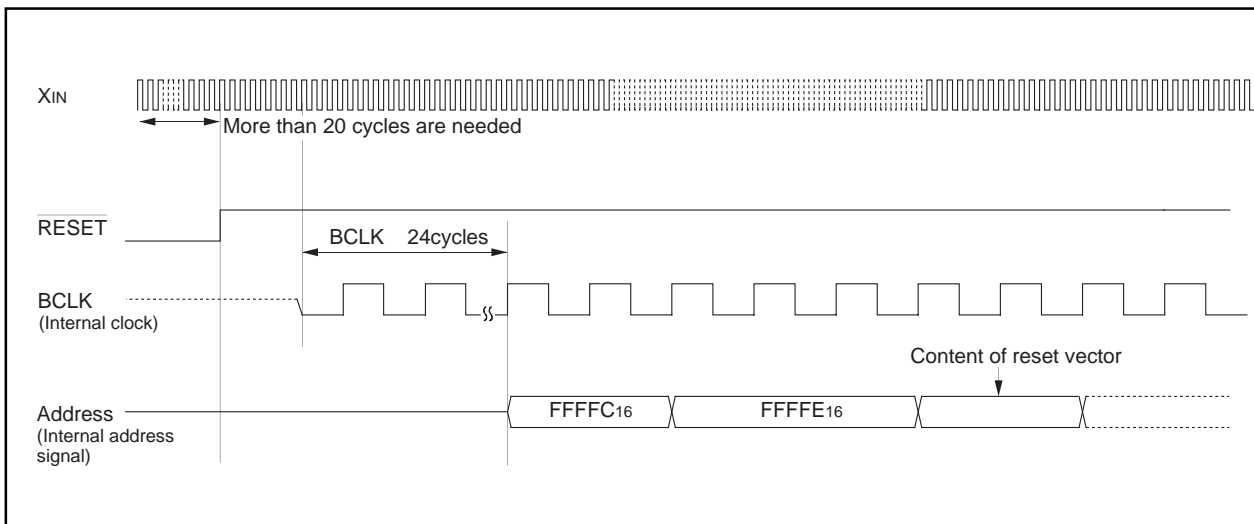
There are two kinds of resets; hardware and software. In both cases, operation is the same after the reset. (See "Software Reset" for details of software resets.) This section explains on hardware resets.

When the supply voltage is in the range where operation is guaranteed, a reset is effected by holding the reset pin level "L" (0.2V<sub>CC</sub> max.) for at least 20 cycles. When the reset pin level is then returned to the "H" level while main clock is stable, the reset status is cancelled and program execution resumes from the address in the reset vector table.

Figure 1.11 shows the example reset circuit. Figure 1.12 shows the reset sequence.



**Figure 1.11. Example reset circuit**



**Figure 1.12. Reset sequence**

(1) Processor mode register 0	(0004 <sub>16</sub> )...		(33) Timer B0 mode register	(039B <sub>16</sub> )...	
(2) Processor mode register 1	(0005 <sub>16</sub> )...		(34) Timer B1 mode register	(039C <sub>16</sub> )...	
(3) System clock control register 0	(0006 <sub>16</sub> )...		(35) UART0 transmit/receive mode register	(03A0 <sub>16</sub> )...	
(4) System clock control register 1	(0007 <sub>16</sub> )...		(36) UART0 transmit/receive control register 0	(03A4 <sub>16</sub> )...	
(5) Address match interrupt enable register	(0009 <sub>16</sub> )...		(37) UART0 transmit/receive control register 1	(03A5 <sub>16</sub> )...	
(6) Protect register	(000A <sub>16</sub> )...		(38) UART1 transmit/receive mode register	(03A8 <sub>16</sub> )...	
(7) Watchdog timer control register	(000F <sub>16</sub> )...		(39) UART1 transmit/receive control register 0	(03AC <sub>16</sub> )...	
(8) Address match interrupt register 0	(0010 <sub>16</sub> )...		(40) UART1 transmit/receive control register 1	(03AD <sub>16</sub> )...	
	(0011 <sub>16</sub> )...		(41) UART transmit/receive control register 2	(03B0 <sub>16</sub> )...	
	(0012 <sub>16</sub> )...		(42) Flash memory control register 0 (Note)	(03B4 <sub>16</sub> )...	
(9) Address match interrupt register 1	(0014 <sub>16</sub> )...		(43) Flash memory control register 1 (Note)	(03B5 <sub>16</sub> )...	
	(0015 <sub>16</sub> )...		(44) Flash command register	(03B6 <sub>16</sub> )...	
	(0016 <sub>16</sub> )...		(45) A-D control register 2	(03D4 <sub>16</sub> )...	
(10) Key input interrupt control register	(004D <sub>16</sub> )...		(46) A-D control register 0	(03D6 <sub>16</sub> )...	
(11) A-D conversion interrupt control register	(004E <sub>16</sub> )...		(47) A-D control register 1	(03D7 <sub>16</sub> )...	
(12) UART0 transmit interrupt control register	(0051 <sub>16</sub> )...		(48) Port P0 direction register	(03E2 <sub>16</sub> )...	
(13) UART0 receive interrupt control register	(0052 <sub>16</sub> )...		(49) Port P1 direction register	(03E3 <sub>16</sub> )...	
(14) UART1 transmit interrupt control register	(0053 <sub>16</sub> )...		(50) Port P2 direction register	(03E6 <sub>16</sub> )...	
(15) UART1 receive interrupt control register	(0054 <sub>16</sub> )...		(51) Port P3 direction register	(03E7 <sub>16</sub> )...	
(16) Timer A0 interrupt control register	(0055 <sub>16</sub> )...		(52) Port P4 direction register	(03EA <sub>16</sub> )...	
(17) Timer X0 interrupt control register	(0056 <sub>16</sub> )...		(53) Port P5 direction register	(03EB <sub>16</sub> )...	
(18) Timer X1 interrupt control register	(0057 <sub>16</sub> )...		(54) Port P6 direction register	(03EE <sub>16</sub> )...	
(19) Timer X2 interrupt control register	(0058 <sub>16</sub> )...		(55) Port P7 direction register	(03EF <sub>16</sub> )...	
(20) Timer B0 interrupt control register	(005A <sub>16</sub> )...		(56) Pull-up control register 0	(03FC <sub>16</sub> )...	
(21) Timer B1 interrupt control register	(005B <sub>16</sub> )...		(57) Pull-up control register 1	(03FD <sub>16</sub> )...	
(22) INT0 interrupt control register	(005D <sub>16</sub> )...		(58) Port P1 drive capacity control register	(03FE <sub>16</sub> )...	
(23) INT1 interrupt control register	(005E <sub>16</sub> )...		(59) Data registers (R0/R1/R2/R3)		
(24) Count start flag	(0380 <sub>16</sub> )...		(60) Address registers (A0/A1)		
(25) Clock prescaler reset flag	(0381 <sub>16</sub> )...		(61) Frame base register (FB)		
(26) One-shot start flag	(0382 <sub>16</sub> )...		(62) Interrupt table register (INTB)		
(27) Trigger select flag	(0383 <sub>16</sub> )...		(63) User stack pointer (USP)		
(28) Up-down flag	(0384 <sub>16</sub> )...		(64) Interrupt stack pointer (ISP)		
(29) Timer A0 mode register	(0396 <sub>16</sub> )...		(65) Static base register (SB)		
(30) Timer X0 mode register	(0397 <sub>16</sub> )...		(66) Flag register (FLG)		
(31) Timer X1 mode register	(0398 <sub>16</sub> )...				
(32) Timer X2 mode register	(0399 <sub>16</sub> )...				

x : Nothing is mapped to this bit  
? : Undefined

The content of other registers and RAM is undefined when the microcomputer is reset. The initial values must therefore be set.

Note: This register is only exist in flash memory version.

Figure 1.13. Device's internal status after a reset is cleared

## Software Reset

### Software Reset

Writing "1" to bit 3 of the processor mode register 0 (address 000416) applies a (software) reset to the microcomputer. A software reset has almost the same effect as a hardware reset. The contents of internal RAM are preserved.

Figure 1.14 shows the processor mode register 0 and 1.

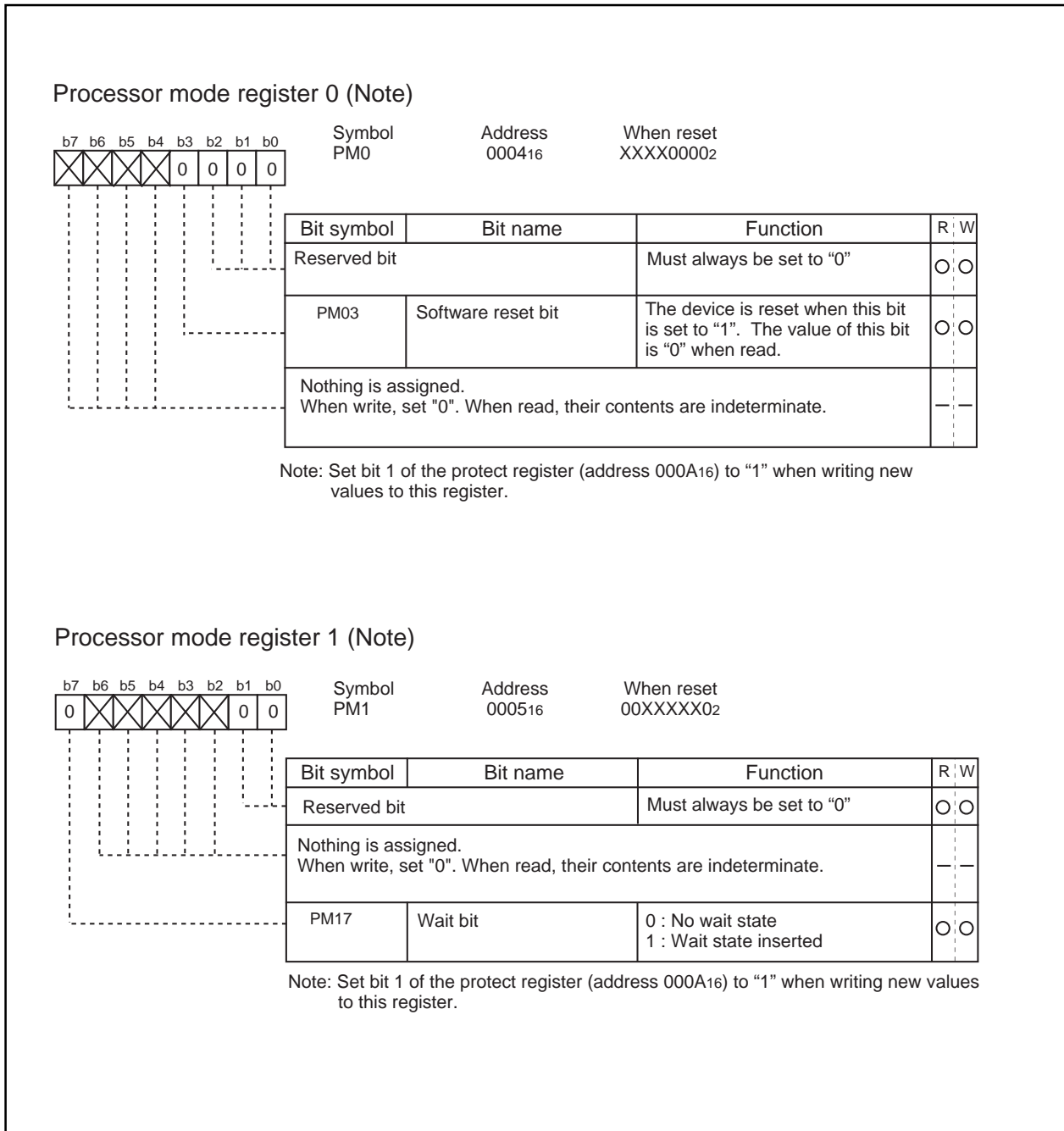


Figure 1.14. Processor mode register 0 and 1.



## Software Wait

---

### Software wait

The wait bit (bit 7) of the processor mode register 1 (address 0005<sub>16</sub>)(note) allows you to insert software wait states for the internal ROM/RAM areas. If this bit is 0, the bus cycle is executed in one BCLK (internal clock) period; if the bit is 1, the bus cycle is executed in two BCLK periods. This bit is cleared to 0 after a reset.

The SFR area is unaffected by this control bit; it is always accessed in two BCLK periods.

Table 1.2 shows the relationship between software wait states and bus cycles.

Note: Before attempting to change the contents of the processor mode register 1, set bit 1 of the protect register (address 000A<sub>16</sub>) to "1".

**Table 1.2. Software waits and bus cycles**

Area	Wait bit	Bus cycle
SFR	Invalid	2 BCLK cycles
Internal ROM/RAM	0	1 BCLK cycle
	1	2 BCLK cycles

## Clock Generating Circuit

### Clock Generating Circuit

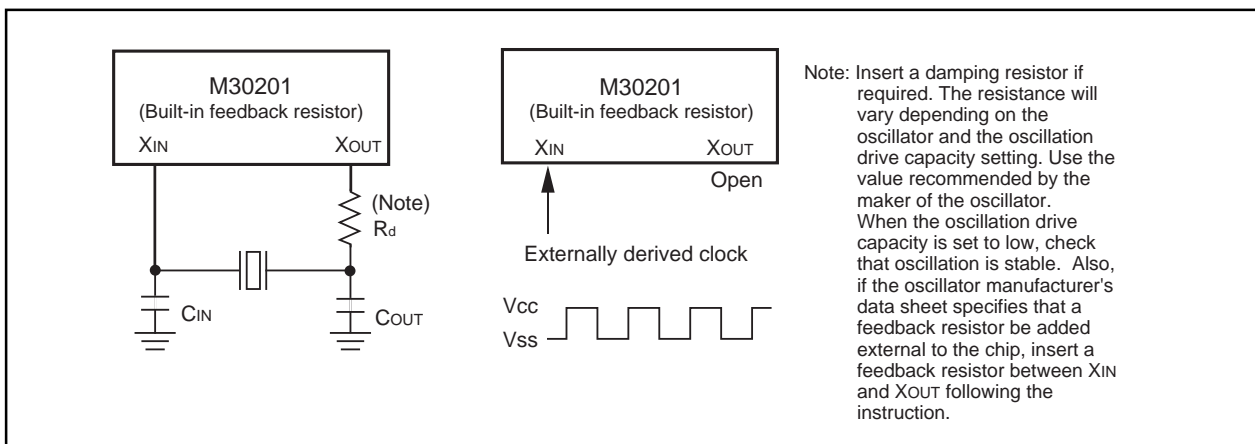
The clock generating circuit contains two oscillator circuits that supply the operating clock sources to the CPU and internal peripheral units.

**Table 1.3. Main clock and sub-clock generating circuits**

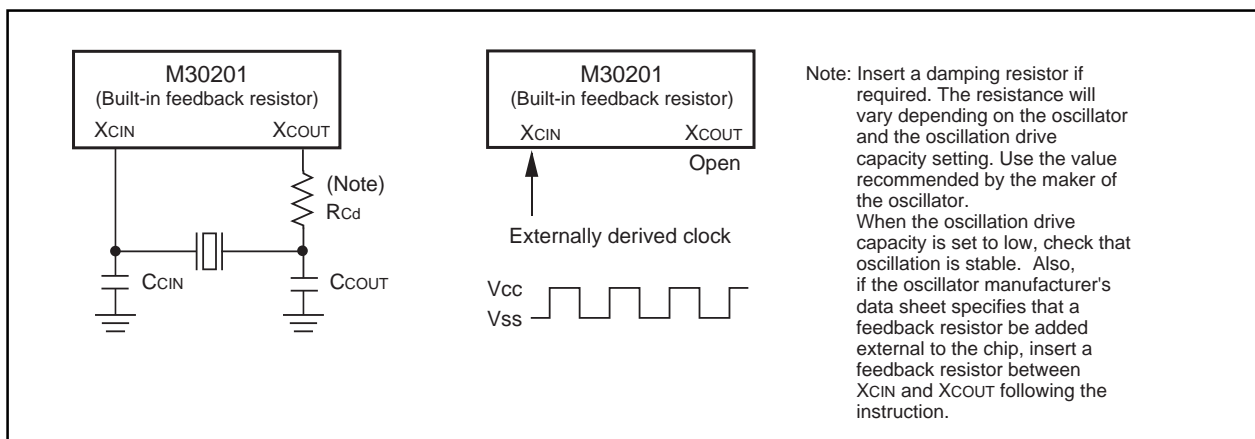
	Main clock generating circuit	Sub clock generating circuit
Use of clock	<ul style="list-style-type: none"> <li>• CPU's operating clock source</li> <li>• Internal peripheral units' operating clock source</li> </ul>	<ul style="list-style-type: none"> <li>• CPU's operating clock source</li> <li>• Timer A/B/X's count clock source</li> </ul>
Usable oscillator	Ceramic or crystal oscillator	Crystal oscillator
Pins to connect oscillator	XIN, XOUT	XCIN, XCOU
Oscillation stop/restart function	Available	Available
Oscillator status immediately after reset	Oscillating	Stopped
Other	Externally derived clock can be input	

### Example of oscillator circuit

Figure 1.15 shows some examples of the main clock circuit, one using an oscillator connected to the circuit, and the other one using an externally derived clock for input. Figure 1.16 shows some examples of sub-clock circuits, one using an oscillator connected to the circuit, and the other one using an externally derived clock for input. Circuit constants in Figures 15 and 16 vary with each oscillator used. Use the values recommended by the manufacturer of your oscillator.



**Figure 1.15. Examples of main clock**



**Figure 1.16. Examples of sub-clock**

## Clock Generating Circuit

### Clock Control

Figure 1.17 shows the block diagram of the clock generating circuit.

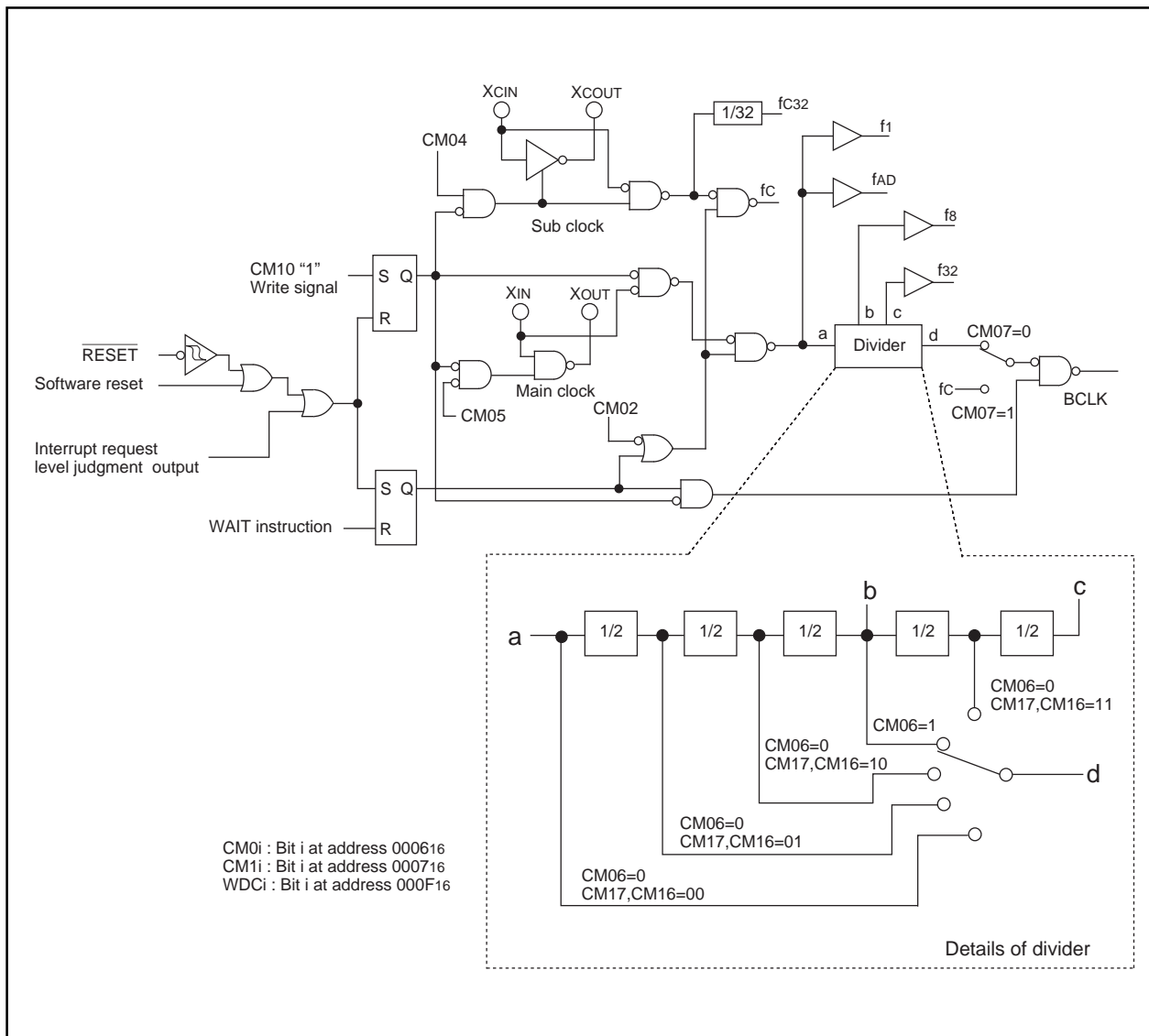


Figure 1.17. Clock generating circuit

## Clock Generating Circuit

---

The following paragraphs describes the clocks generated by the clock generating circuit.

### (1) Main clock

The main clock is generated by the main clock oscillation circuit. After a reset, the clock is divided by 8 to BCLK. The clock can be stopped using the main clock stop bit (bit 5 at address 0006<sub>16</sub>). Stopping the clock, after switching the operating clock source of CPU to the sub-clock, reduces the power dissipation. After the oscillation of the main clock oscillation circuit has stabilized, the drive capacity of the main clock oscillation circuit can be reduced using the XIN-XOUT drive capacity select bit (bit 5 at address 0007<sub>16</sub>). Reducing the drive capacity of the main clock oscillation circuit reduces the power dissipation. This bit changes to "1" when shifting from high-speed/medium-speed mode to stop mode and at a reset. When shifting from low-speed/low power dissipation mode to stop mode, the value before stop mode is retained.

### (2) Sub-clock

The sub-clock is generated by the sub-clock oscillation circuit. No sub-clock is generated after a reset. After oscillation is started using the port Xc select bit (bit 4 at address 0006<sub>16</sub>), the sub-clock can be selected as BCLK by using the system clock select bit (bit 7 at address 0006<sub>16</sub>). However, be sure that the sub-clock oscillation has fully stabilized before switching.

After the oscillation of the sub-clock oscillation circuit has stabilized, the drive capacity of the sub-clock oscillation circuit can be reduced using the XCIN-XCOUT drive capacity select bit (bit 3 at address 0006<sub>16</sub>). Reducing the drive capacity of the sub-clock oscillation circuit reduces the power dissipation. This bit changes to "1" when shifting to stop mode and at a reset.

### (3) BCLK

The BCLK is the clock that drives the CPU, and is fc or the clock is derived by dividing the main clock by 1, 2, 4, 8, or 16. The BCLK is derived by dividing the main clock by 8 after a reset.

The main clock division select bit 0(bit 6 at address 0006<sub>16</sub>) changes to "1" when shifting from high-speed/medium-speed to stop mode and at reset. When shifting from low-speed/low power dissipation mode to stop mode, the value before stop mode is retained.

### (4) Peripheral function clock (f<sub>1</sub>, f<sub>8</sub>, f<sub>32</sub>, f<sub>AD</sub>)

The clock for the peripheral devices is derived from the main clock or by dividing it by 8 or 32. The peripheral function clock is stopped by stopping the main clock or by setting the WAIT peripheral function clock stop bit (bit 2 at 0006<sub>16</sub>) to "1" and then executing a WAIT instruction.

### (5) fc<sub>32</sub>

This clock is derived by dividing the sub-clock by 32. It is used for the timer A, timer B and timer X counts.

### (6) fc

This clock has the same frequency as the sub-clock. It is used for BCLK and for the watchdog timer.

Clock Generating Circuit

Figure 1.18 shows the system clock control registers 0 and 1.

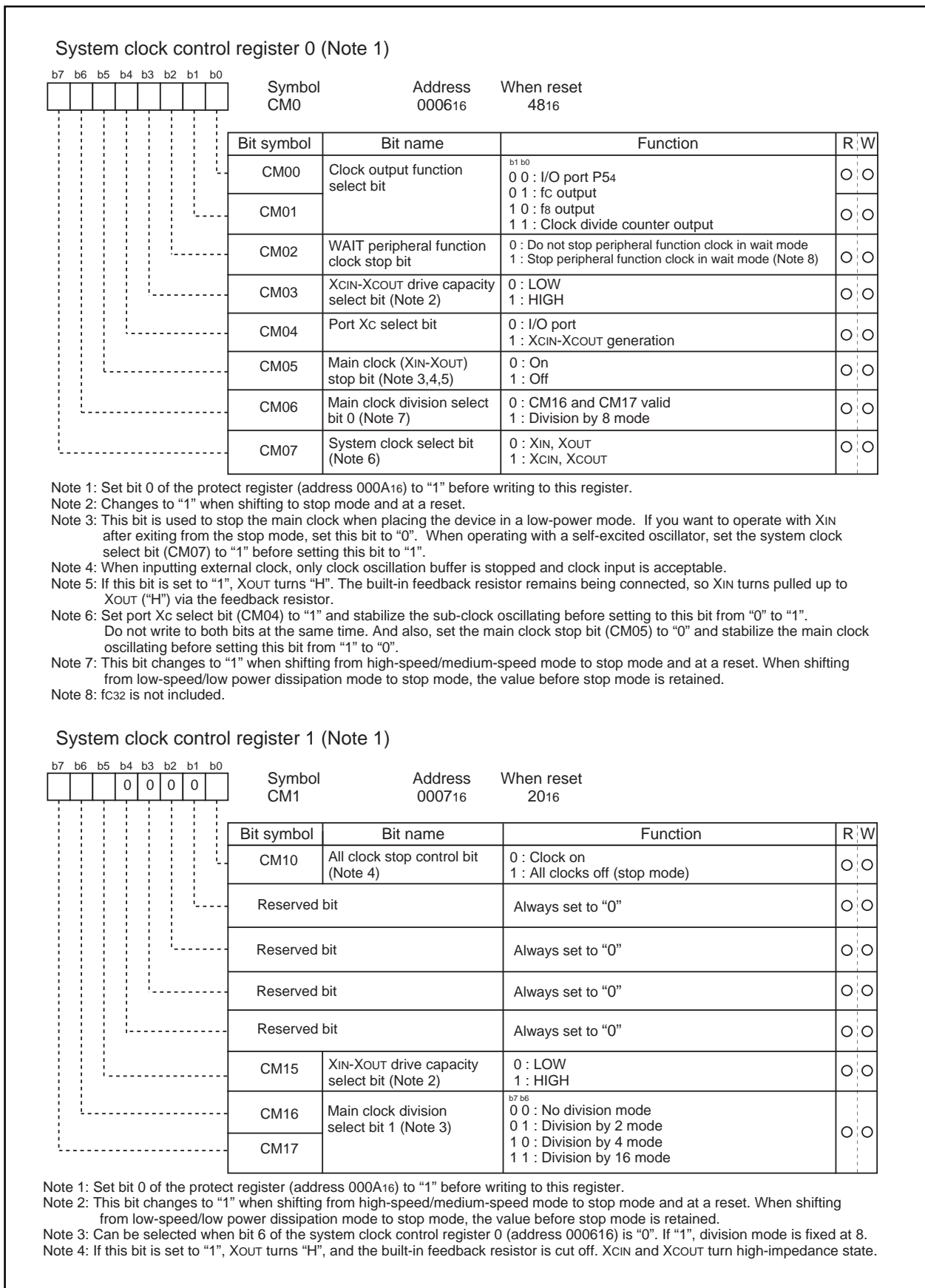


Figure 1.18. Clock control registers 0 and 1

## Clock Generating Circuit

### Clock Output

The clock output function select bit allows you to choose the clock from f8, fc, or a divide-by-n clock that is output from the P54/CKOUT pin. The clock divide counter is an 8-bit counter whose count source is f32, and its divide ratio can be set in the range of 0016 to FF16. Figure 1.19 shows a block diagram of clock output.

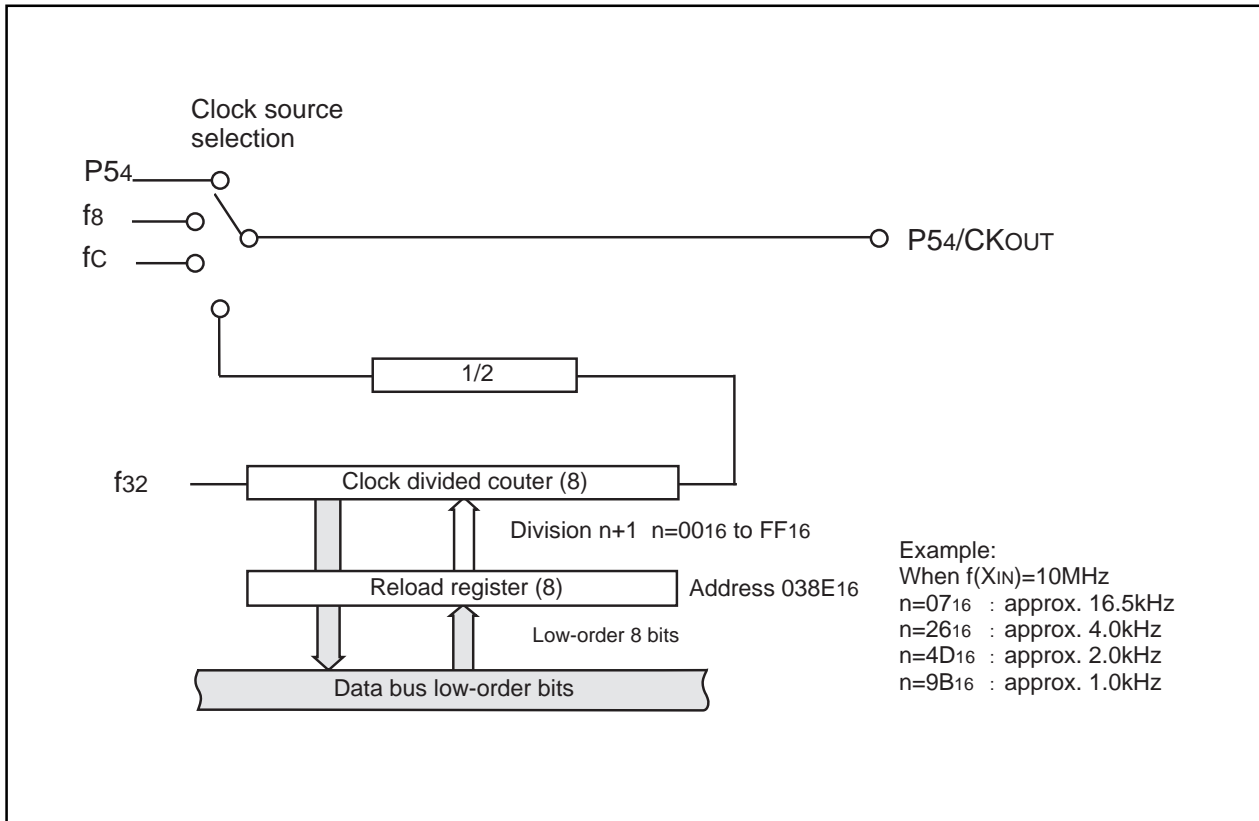


Figure 1.19. Block diagram of clock output

## Wait Mode

**Stop Mode**

Writing "1" to the all-clock stop control bit (bit 0 at address 0007<sub>16</sub>) stops all oscillation and the microcomputer enters stop mode. In stop mode, the content of the internal RAM is retained provided that VCC remains above 2V.

Because the oscillation of BCLK, f<sub>1</sub> to f<sub>32</sub>, f<sub>c</sub>, f<sub>c32</sub>, and f<sub>AD</sub> stops in stop mode, peripheral functions such as the A-D converter and watchdog timer do not function. However, timer A, timer B and timer X operate provided that the event counter mode is set to an external pulse, and UART0 functions provided an external clock is selected. Table 1.4 shows the status of the ports in stop mode.

Stop mode is cancelled by a hardware reset or an interrupt. If an interrupt is to be used to cancel stop mode, that interrupt must first have been enabled. If returning by an interrupt, that interrupt routine is executed.

When shifting from high-speed/medium-speed mode to stop mode and at a reset, the main clock division select bit 0 (bit 6 at address 0006<sub>16</sub>) is set to "1". When shifting from low-speed/low power dissipation mode to stop mode, the value before stop mode is retained.

**Table 1.4. Port status during stop mode**

Pin		States
Port		Retains status before stop mode
CLKOUT	When f <sub>c</sub> selected	"H"
	When f <sub>8</sub> , clock divided counter output selected	Retains status before stop mode

**Wait Mode**

When a WAIT instruction is executed, BCLK stops and the microcomputer enters the wait mode. In this mode, oscillation continues but BCLK and watchdog timer stop. Writing "1" to the WAIT peripheral function clock stop bit and executing a WAIT instruction stops the clock being supplied to the internal peripheral functions, allowing power dissipation to be reduced. Table 1.5 shows the status of the ports in wait mode.

Wait mode is cancelled by a hardware reset or interrupt. If an interrupt is used to cancel wait mode, the microcomputer restarts from the interrupt routine using as BCLK, the clock that had been selected when the WAIT instruction was executed.

**Table 1.5. Port status during wait mode**

Pin		States
Port		Retains status before wait mode
CLKOUT	When f <sub>c</sub> selected	Does not stop
	When f <sub>8</sub> , clock divided counter output selected	Does not stop when the WAIT peripheral function clock stop bit is "0". When the WAIT peripheral function clock stop bit is "1", the status immediately prior to entering wait mode is maintained.

## Status Transition of BCLK

### Status Transition of BCLK

Power dissipation can be reduced and low-voltage operation achieved by changing the count source for BCLK. Table 1.6 shows the operating modes corresponding to the settings of system clock control registers 0 and 1.

When reset, the device starts in division by 8 mode. The main clock division select bit 0 (bit 6 at address 000616) changes to "1" when shifting from high-speed/medium-speed to stop mode and at a reset. When shifting from low-speed/low power dissipation mode to stop mode, the value before stop mode is retained. The following shows the operational modes of BCLK.

#### (1) Division by 2 mode

The main clock is divided by 2 to obtain the BCLK.

#### (2) Division by 4 mode

The main clock is divided by 4 to obtain the BCLK.

#### (3) Division by 8 mode

The main clock is divided by 8 to obtain the BCLK. When reset, the device starts operating from this mode. Before the user can go from this mode to no division mode, division by 2 mode, or division by 4 mode, the main clock must be oscillating stably. When going to low-speed or lower power consumption mode, make sure the sub-clock is oscillating stably.

#### (4) Division by 16 mode

The main clock is divided by 16 to obtain the BCLK.

#### (5) No-division mode

The main clock is divided by 1 to obtain the BCLK.

#### (6) Low-speed mode

fc is used as BCLK. Note that oscillation of both the main and sub-clocks must have stabilized before transferring from this mode to another or vice versa. At least 2 to 3 seconds are required after the sub-clock starts. Therefore, the program must be written to wait until this clock has stabilized immediately after powering up and after stop mode is cancelled.

#### (7) Low power dissipation mode

fc is the BCLK and the main clock is stopped.

Note : Before the count source for BCLK can be changed from XIN to XCIN or vice versa, the clock to which the count source is going to be switched must be oscillating stably. Allow a wait time in software for the oscillation to stabilize before switching over the clock.

**Table 1.6. Operating modes dictated by settings of system clock control registers 0 and 1**

CM17	CM16	CM07	CM06	CM05	CM04	Operating mode of BCLK
0	1	0	0	0	Invalid	Division by 2 mode
1	0	0	0	0	Invalid	Division by 4 mode
Invalid	Invalid	0	1	0	Invalid	Division by 8 mode
1	1	0	0	0	Invalid	Division by 16 mode
0	0	0	0	0	Invalid	No-division mode
Invalid	Invalid	1	Invalid	0	1	Low-speed mode
Invalid	Invalid	1	Invalid	1	1	Low power dissipation mode



## Power Saving

---

### Power Saving

There are three power save modes.

#### (1) Normal operating mode

- **High-speed mode**

In this mode, one main clock cycle forms BCLK. The CPU operates on the BCLK. The peripheral functions operate on the clocks specified for each respective function.

- **Medium-speed mode**

In this mode, the main clock is divided into 2, 4, 8, or 16 to form BCLK. The CPU operates on the BCLK. The peripheral functions operated on the clocks specified for each respective function.

- **Low-speed mode**

In this mode, fc forms BCLK. The CPU operates on the fc clock. fc is the clock supplied by the subclock. The peripheral functions operate on the clocks specified for each respective function.

- **Low power-dissipation mode**

This mode is selected when the main clock is stopped from low-speed mode. The CPU operates on the fc clock. fc is the clock supplied by the subclock. Only the peripheral functions for which the subclock was selected as the count source continue to run.

#### (2) Wait mode

CPU operation is halted in this mode. The oscillator continues to run.

#### (3) Stop mode

All oscillators stop in this mode. The CPU and internal peripheral functions all stop. Of all 3 power saving modes, power savings are greatest in this mode.

Figure 1.20 shows the transition between each of the three modes, (1), (2), and (3).

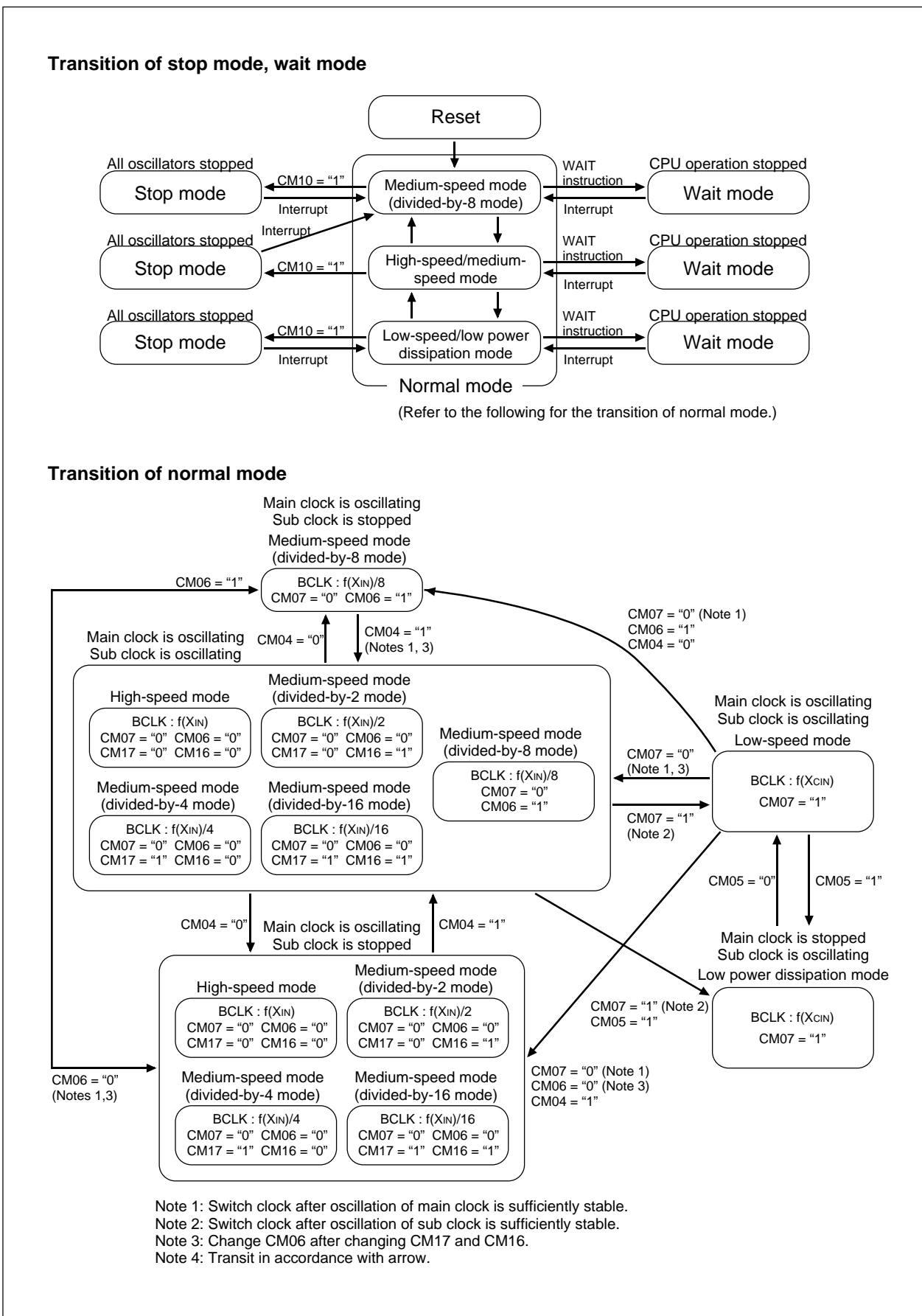


Figure 1.20. Clock transition

## Protection

## Protection

The protection function is provided so that the values in important registers cannot be changed in the event that the program runs out of control. Figure 1.21 shows the protect register. The values in the processor mode register 0 (address 0004<sub>16</sub>), processor mode register 1 (address 0005<sub>16</sub>), system clock control register 0 (address 0006<sub>16</sub>), system clock control register 1 (address 0007<sub>16</sub>) and port P4 direction register (address 03EA<sub>16</sub>) can only be changed when the respective bit in the protect register is set to "1". Therefore, important outputs can be allocated to port P4.

If, after "1" (write-enabled) has been written to the port P4 direction register write-enable bit (bit 2 at address 000A<sub>16</sub>), a value is written to any address, the bit automatically reverts to "0" (write-inhibited). However, the system clock control registers 0 and 1 write-enable bit (bit 0 at 000A<sub>16</sub>) and processor mode register 0 and 1 write-enable bit (bit 1 at 000A<sub>16</sub>) do not automatically return to "0" after a value has been written to an address. The program must therefore be written to return these bits to "0".

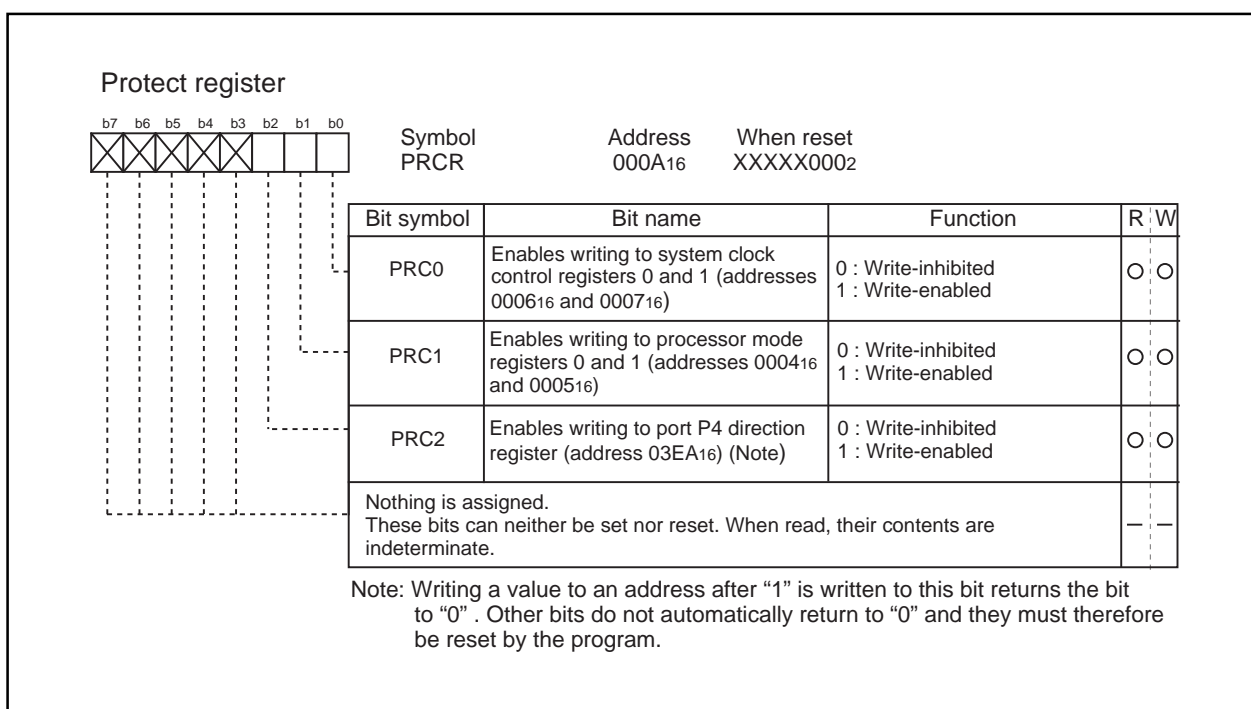


Figure 1.21. Protect register

## Interrupts

## Overview of Interrupt

## Type of Interrupts

Figure 1.22 lists the types of interrupts.

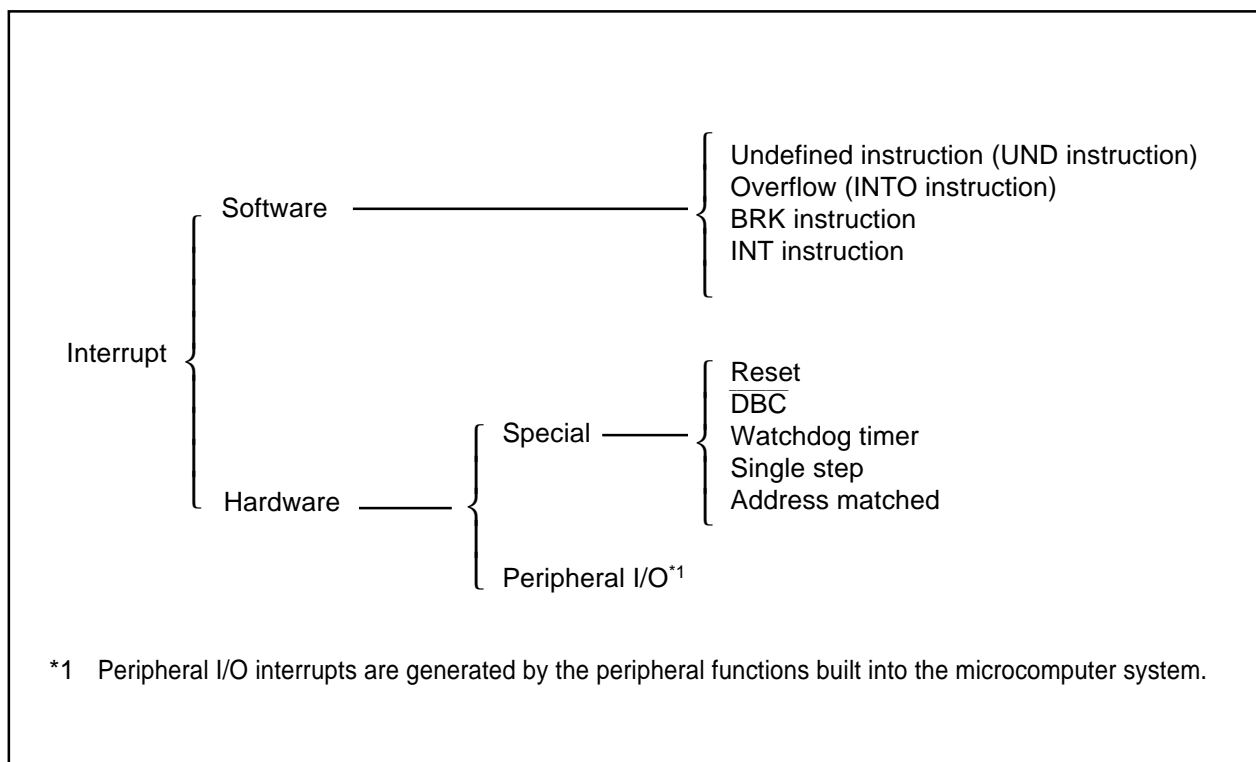


Figure 1.22. Classification of interrupts

- Maskable interrupt : An interrupt which can be enabled (disabled) by the interrupt enable flag (I flag) or whose interrupt priority can be changed by priority level.
- Non-maskable interrupt : An interrupt which cannot be enabled (disabled) by the interrupt enable flag (I flag) or whose interrupt priority cannot be changed by priority level.

## Interrupts

---

### Software Interrupts

A software interrupt occurs when executing certain instructions. Software interrupts are non-maskable interrupts.

- **Undefined instruction interrupt**

An undefined instruction interrupt occurs when executing the UND instruction.

- **Overflow interrupt**

An overflow interrupt occurs when executing the INTO instruction with the overflow flag (O flag) set to "1".

The following are instructions whose O flag changes by arithmetic:

ABS, ADC, ADCF, ADD, CMP, DIV, DIVU, DIVX, NEG, RMPA, SBB, SHA, SUB

- **BRK interrupt**

A BRK interrupt occurs when executing the BRK instruction.

- **INT interrupt**

An INT interrupt occurs when assigning one of software interrupt numbers 0 through 63 and executing the INT instruction. Software interrupt numbers 0 through 31 are assigned to peripheral I/O interrupts, so executing the INT instruction allows executing the same interrupt routine that a peripheral I/O interrupt does.

The stack pointer (SP) used for the INT interrupt is dependent on which software interrupt number is involved.

So far as software interrupt numbers 0 through 31 are concerned, the microcomputer saves the stack pointer assignment flag (U flag) when it accepts an interrupt request. It changes the U flag to "0" and selects the interrupt stack pointer (ISP), and then executes an interrupt sequence. When returning from the interrupt routine, the U flag is returned to the state it was before the acceptance of interrupt request. So far as software numbers 32 through 63 are concerned, the stack pointer does not make a shift.

## Hardware Interrupts

Hardware interrupts are classified into two types — special interrupts and peripheral I/O interrupts.

### (1) Special interrupts

Special interrupts are non-maskable interrupts.

- **Reset**

Reset occurs if an “L” is input to the RESET pin.

- **DBC interrupt**

This interrupt is exclusively for the debugger, do not use it in other circumstances.

- **Watchdog timer interrupt**

Generated by the watchdog timer.

- **Single-step interrupt**

This interrupt is exclusively for the debugger, do not use it in other circumstances. With the debug flag (D flag) set to “1”, a single-step interrupt occurs after one instruction is executed.

- **Address match interrupt**

An address match interrupt occurs immediately before the instruction held in the address indicated by the address match interrupt register is executed with the address match interrupt enable bit set to “1”. If an address other than the first address of the instruction in the address match interrupt register is set, no address match interrupt occurs.

### (2) Peripheral I/O interrupts

A peripheral I/O interrupt is generated by one of built-in peripheral functions. The interrupt vector table is the same as the one for software interrupt numbers 0 through 31 the INT instruction uses. Peripheral I/O interrupts are maskable interrupts.

- **Key-input interrupt**

A key-input interrupt occurs if an “L” is input to the  $\overline{KI}$  pin.

- **A-D conversion interrupt**

This is an interrupt that the A-D converter generates.

- **UART0 and UART1 transmission interrupt**

These are interrupts that the serial I/O transmission generates.

- **UART0 and UART1 reception interrupt**

These are interrupts that the serial I/O reception generates.

- **Timer A0 interrupt**

This is an interrupts that timer A0 generates.

- **Timer B0 and timer B2 interrupt**

These are interrupts that timer B generates.

- **Timer X0 to timer X2 interrupt**

These are interrupts that timer X generates.

- **$\overline{INT0}$  and  $\overline{INT1}$  interrupt**

An  $\overline{INT}$  interrupt occurs if either a rising edge or a falling edge is input to the  $\overline{INT}$  pin.

## Interrupts

### Interrupts and Interrupt Vector Tables

If an interrupt request is accepted, a program branches to the interrupt routine set in the interrupt vector table. Set the first address of the interrupt routine in each vector table. Figure 1.23 shows format for specifying interrupt vector addresses.

Two types of interrupt vector tables are available — fixed vector table in which addresses are fixed and variable vector table in which addresses can be varied by the setting.

	MSB	LSB
Vector address + 0	Low address	
Vector address + 1	Mid address	
Vector address + 2	0 0 0 0	High address
Vector address + 3	0 0 0 0	0 0 0 0

Figure 1.23. Format for specifying interrupt vector addresses

#### • Fixed vector tables

The fixed vector table is a table in which addresses are fixed. The vector tables are located in an area extending from FFFDC<sub>16</sub> to FFFFF<sub>16</sub>. One vector table comprises four bytes. Set the first address of interrupt routine in each vector table. Table 1.7 shows the interrupts assigned to the fixed vector tables and addresses of vector tables.

Table 1.7. Interrupt and fixed vector address

Interrupt source	Vector table addresses Address (L) to address (H)	Remarks
Undefined instruction	FFFD <sub>16</sub> to FFFD <sub>16</sub>	Interrupt on UND instruction
Overflow	FFFE <sub>0</sub> <sub>16</sub> to FFFE <sub>3</sub> <sub>16</sub>	Interrupt on INTO instruction
BRK instruction	FFFE <sub>4</sub> <sub>16</sub> to FFFE <sub>7</sub> <sub>16</sub>	If the vector is filled with FF <sub>16</sub> , program execution starts from the address shown by the vector in the variable vector table
Address match	FFFE <sub>8</sub> <sub>16</sub> to FFFE <sub>B</sub> <sub>16</sub>	There is an address-matching interrupt enable bit
Single step (Note)	FFFE <sub>C</sub> <sub>16</sub> to FFFE <sub>F</sub> <sub>16</sub>	Do not use
Watchdog timer	FFFF <sub>0</sub> <sub>16</sub> to FFFF <sub>3</sub> <sub>16</sub>	
DBC (Note)	FFFF <sub>4</sub> <sub>16</sub> to FFFF <sub>7</sub> <sub>16</sub>	Do not use
-	FFFF <sub>8</sub> <sub>16</sub> to FFFF <sub>B</sub> <sub>16</sub>	-
Reset	FFFF <sub>C</sub> <sub>16</sub> to FFFF <sub>F</sub> <sub>16</sub>	

Note: Interrupts used for debugging purposes only.

## Interrupts

### • Variable vector tables

The addresses in the variable vector table can be modified, according to the user's settings. Indicate the first address using the interrupt table register (INTB). The 256-byte area subsequent to the address the INTB indicates becomes the area for the variable vector tables. One vector table comprises four bytes. Set the first address of the interrupt routine in each vector table. Table 1.8 shows the interrupts assigned to the variable vector tables and addresses of vector tables.

**Table 1.8. Interrupt causes (variable interrupt vector addresses)**

Software interrupt number	Vector table address Address (L) to address (H)	Interrupt source	Remarks
Software interrupt number 0	+0 to +3 (Note)	BRK instruction	Cannot be masked by I flag
—		—	
Software interrupt number 11	+44 to +47 (Note)	—	
Software interrupt number 12	+48 to +51 (Note)	—	
Software interrupt number 13	+52 to +55 (Note)	Key input interrupt	
Software interrupt number 14	+56 to +59 (Note)	A-D	
—		—	
Software interrupt number 17	+68 to +71 (Note)	UART0 transmit	
Software interrupt number 18	+72 to +75 (Note)	UART0 receive	
Software interrupt number 19	+76 to +79 (Note)	UART1 transmit	
Software interrupt number 20	+80 to +83 (Note)	UART1 receive	
Software interrupt number 21	+84 to +87 (Note)	Timer A0	
Software interrupt number 22	+88 to +91 (Note)	Timer X0	
Software interrupt number 23	+92 to +95 (Note)	Timer X1	
Software interrupt number 24	+96 to +99 (Note)	Timer X2	
Software interrupt number 25	+100 to +103 (Note)	—	
Software interrupt number 26	+104 to +107 (Note)	Timer B0	
Software interrupt number 27	+108 to +111 (Note)	Timer B1	
Software interrupt number 28	+112 to +115 (Note)	—	
Software interrupt number 29	+116 to +119 (Note)	$\overline{\text{INT0}}$	
Software interrupt number 30	+120 to +123 (Note)	$\overline{\text{INT1}}$	
Software interrupt number 31	+124 to +127 (Note)	—	
Software interrupt number 32 to Software interrupt number 63	+128 to +131 (Note) to +252 to +255 (Note)	Software interrupt	Cannot be masked by I flag

Note : Address relative to address in interrupt table register (INTB).



## Interrupts

---

### Interrupt Control

Descriptions are given here regarding how to enable or disable maskable interrupts and how to set the priority to be accepted. What is described here does not apply to non-maskable interrupts.

Enable or disable a maskable interrupt using the interrupt enable flag (I flag), interrupt priority level select bit, and processor interrupt priority level (IPL). Whether an interrupt request is present or absent is indicated by the interrupt request bit. The interrupt request bit and the interrupt priority level selection bit are located in the interrupt control register of each interrupt. Also, the interrupt enable flag (I flag) and the IPL are located in the flag register (FLG).

Figure 1.24 shows the interrupt control registers.

Interrupts

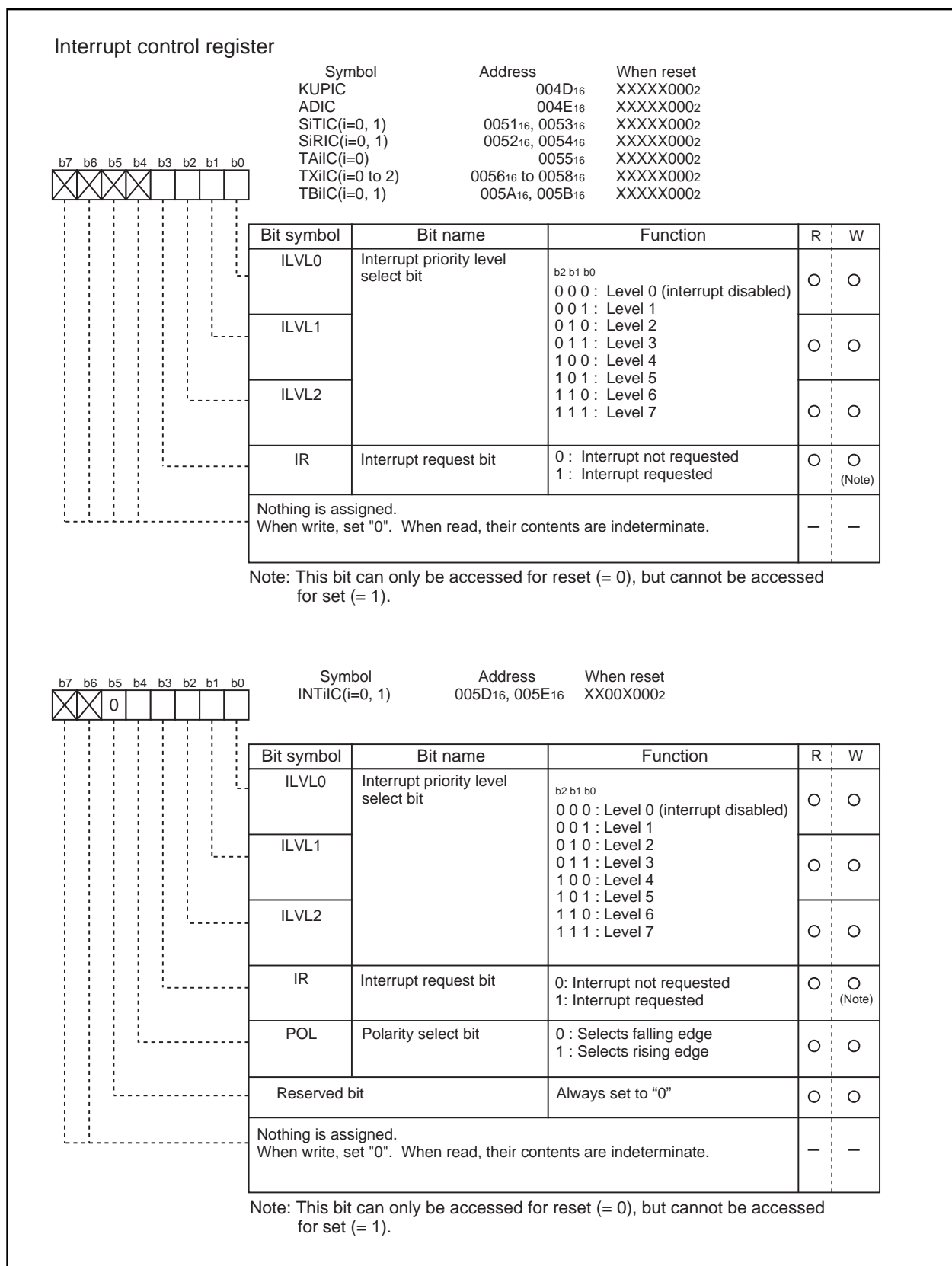


Figure 1.24. Interrupt control register

## Interrupts

### Interrupt Enable Flag

The interrupt enable flag (I flag) controls the enabling and disabling of maskable interrupts. Setting this flag to "1" enables all maskable interrupts; setting it to "0" disables all maskable interrupts. This flag is set to "0" after reset.

### Interrupt Request Bit

The interrupt request bit is set to "1" by hardware when an interrupt is requested. After the interrupt is accepted and jumps to the corresponding interrupt vector, the request bit is set to "0" by hardware. The interrupt request bit can also be set to "0" by software. (Do not set this bit to "1").

### Interrupt Priority Level Select Bit and Processor Interrupt Priority Level (IPL)

Set the interrupt priority level using the interrupt priority level select bit, which is one of the component bits of the interrupt control register. When an interrupt request occurs, the interrupt priority level is compared with the IPL. The interrupt is enabled only when the priority level of the interrupt is higher than the IPL. Therefore, setting the interrupt priority level to "0" disables the interrupt.

Table 1.9 shows the settings of interrupt priority levels and Table 1.10 shows the interrupt levels enabled, according to the consist of the IPL.

The following are conditions under which an interrupt is accepted:

- interrupt enable flag (I flag) = 1
- interrupt request bit = 1
- interrupt priority level > IPL

The interrupt enable flag (I flag), the interrupt request bit, the interrupt priority select bit, and the IPL are independent, and they are not affected by one another.

**Table 1.9. Settings of interrupt priority levels**

Interrupt priority level select bit	Interrupt priority level	Priority order
b2 b1 b0 0 0 0	Level 0 (interrupt disabled)	———
0 0 1	Level 1	Low ↓ High
0 1 0	Level 2	
0 1 1	Level 3	
1 0 0	Level 4	
1 0 1	Level 5	
1 1 0	Level 6	
1 1 1	Level 7	

**Table 1.10. Interrupt levels enabled according to the contents of the IPL**

IPL	Enabled interrupt priority levels
IPL <sub>2</sub> IPL <sub>1</sub> IPL <sub>0</sub> 0 0 0	Interrupt levels 1 and above are enabled
0 0 1	Interrupt levels 2 and above are enabled
0 1 0	Interrupt levels 3 and above are enabled
0 1 1	Interrupt levels 4 and above are enabled
1 0 0	Interrupt levels 5 and above are enabled
1 0 1	Interrupt levels 6 and above are enabled
1 1 0	Interrupt levels 7 and above are enabled
1 1 1	All maskable interrupts are disabled

## Interrupts

---

### Changing the Interrupt Control Register

#### < Program examples >

The program examples are described as follow:

Example 1:

```
INT_SWITCH1:
  FCLR   I           ; Disable interrupts.
  AND.B  #00h, 0055h ; Clear TA0IC int. priority level and int. request bit.
  NOP                    ; Four NOP instructions are required when using HOLD function.
  NOP
  FSET   I           ; Enable interrupts.
```

Example 2:

```
INT_SWITCH2:
  FCLR   I           ; Disable interrupts.
  AND.B  #00h, 0055h ; Clear TA0IC int. priority level and int. request bit.
  MOV.W  MEM, R0     ; Dummy read.
  FSET   I           ; Enable interrupts.
```

Example 3:

```
INT_SWITCH3:
  PUSHC  FLG         ; Push Flag register onto stack
  FCLR   I           ; Disable interrupts.
  AND.B  #00h, 0055h ; Clear TA0IC int. priority level and int. request bit.
  POPC   FLG         ; Enable interrupts.
```

The reason why two NOP instructions or dummy read are inserted before FSET I in Examples 1 and 2 is to prevent the interrupt enable flag I from being set before the interrupt control register is rewritten due to effects of the instruction queue.

If changing the interrupt control register using an instruction other than the instructions listed here, and if an interrupt occurs associated with this register during execution of the instruction, there can be instances in which the interrupt request bit is not set. To avoid this problem, use one of the instructions given below to change the register.

Following instructions: AND, OR, BCLR or BSET

## Interrupts

### Interrupt Sequence

An interrupt sequence — what are performed over a period from the instant an interrupt is accepted to the instant the interrupt routine is executed — is described here.

If an interrupt occurs during execution of an instruction, the processor determines its priority when the execution of the instruction is completed, and transfers control to the interrupt sequence from the next cycle. If an interrupt occurs during execution of either the SMOVB, SMOVF, SSTR or RMPA instruction, the processor temporarily suspends the instruction being executed, and transfers control to the interrupt sequence.

In the interrupt sequence, the processor carries out the following in sequence given:

- (1) CPU gets the interrupt information (the interrupt number and interrupt request level) by reading address 0000016. After this, the corresponding interrupt request bit becomes "0".
- (2) Saves the content of the flag register (FLG) as it was immediately before the start of interrupt sequence in the temporary register (Note) within the CPU.
- (3) Sets the interrupt enable flag (I flag), the debug flag (D flag), and the stack pointer select flag (U flag) to "0" (the U flag, however, does not change if the INT instruction, in software interrupt numbers 32 through 63, is executed).
- (4) Saves the content of the temporary register (Note) within the CPU in the stack area.
- (5) Saves the content of the program counter (PC) in the stack area.
- (6) Sets the interrupt priority level of the accepted instruction in the IPL.

After the interrupt sequence is completed, the processor resumes executing instructions from the first address of the interrupt routine.

Note: This register cannot be utilized by the user.

### Interrupt Response Time

'Interrupt response time' is the period between the instant an interrupt occurs and the instant the first instruction within the interrupt routine has been executed. This time comprises the period from the occurrence of an interrupt to the completion of the instruction under execution at that moment (a) and the time required for executing the interrupt sequence (b). Figure 1.25 shows the interrupt response time.

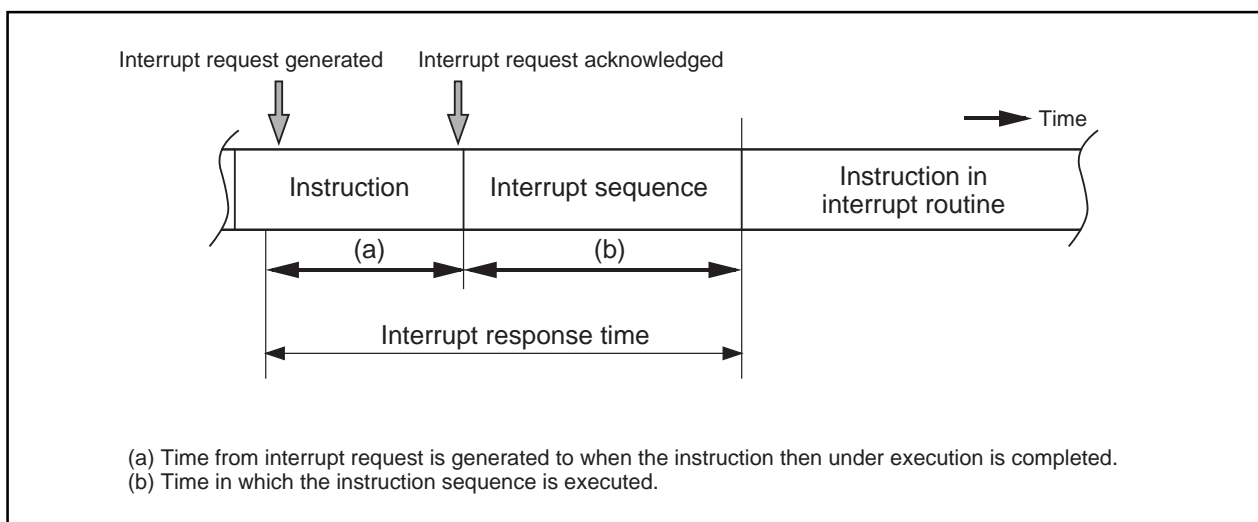


Figure 1.25. Interrupt response time

Interrupts

Time (a) is dependent on the instruction under execution. Thirty cycles is the maximum required for the DIVX instruction (without wait).

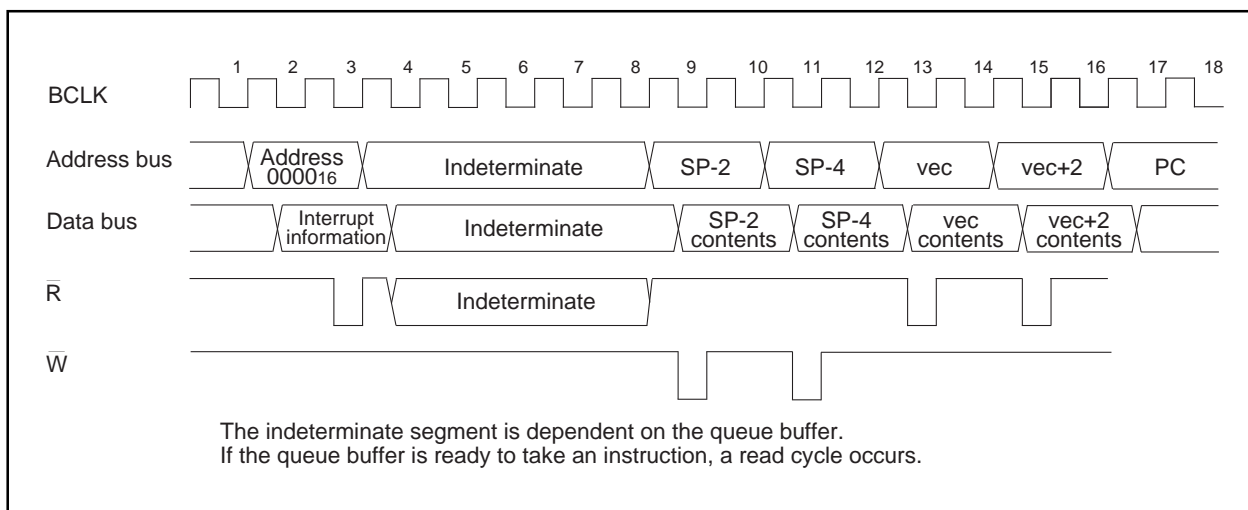
Time (b) is as shown in Table 1.11.

**Table 1.11. Time required for executing the interrupt sequence**

Interrupt vector address	Stack pointer (SP) value	16-bit bus, without wait	8-bit bus, without wait
Even	Even	18 cycles (Note 1)	20 cycles (Note 1)
Even	Odd	19 cycles (Note 1)	20 cycles (Note 1)
Odd (Note 2)	Even	19 cycles (Note 1)	20 cycles (Note 1)
Odd (Note 2)	Odd	20 cycles (Note 1)	20 cycles (Note 1)

Note 1: Add 2 cycles in the case of a  $\overline{DBC}$  interrupt; add 1 cycle in the case either of an address match interrupt or of a single-step interrupt.

Note 2: Locate an interrupt vector address in an even address, if possible.



**Figure 1.26. Time required for executing the interrupt sequence**

**Variation of IPL when Interrupt Request is Accepted**

If an interrupt request is accepted, the interrupt priority level of the accepted interrupt is set in the IPL.

If an interrupt request, that does not have an interrupt priority level, is accepted, one of the values shown in Table 1.12 is set in the IPL.

**Table 1.12. Relationship between interrupts without interrupt priority levels and IPL**

Interrupt sources without priority levels	Value set in the IPL
Watchdog timer	7
Reset	0
Other	Not changed

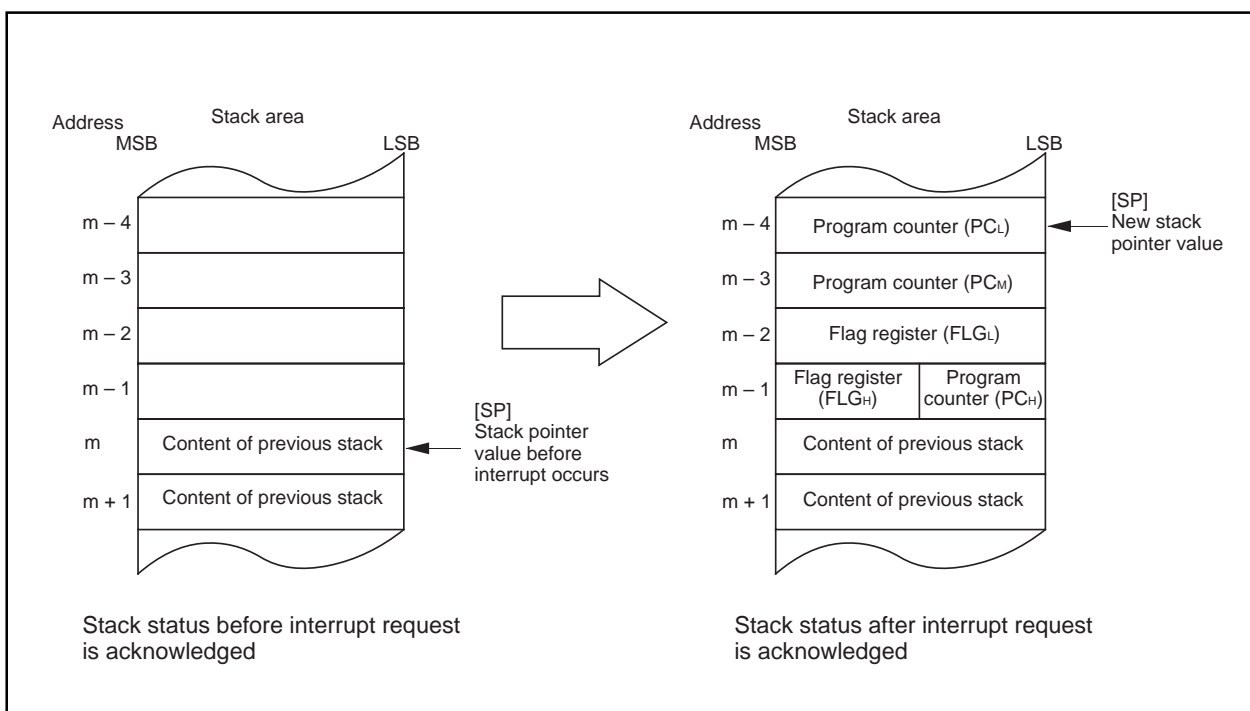
## Interrupts

### Saving Registers

In the interrupt sequence, only the contents of the flag register (FLG) and that of the program counter (PC) are saved in the stack area.

First, the processor saves the 4 high-order bits of the program counter, and 4 high-order bits and 8 low-order bits of the FLG register, 16 bits in total, in the stack area, then saves 16 low-order bits of the program counter. Figure 1.27 shows the state of the stack as it was before the acceptance of the interrupt request, and the state the stack after the acceptance of the interrupt request.

Save other necessary registers at the beginning of the interrupt routine using software. Using the PUSHM instruction alone can save all the registers except the stack pointer (SP).

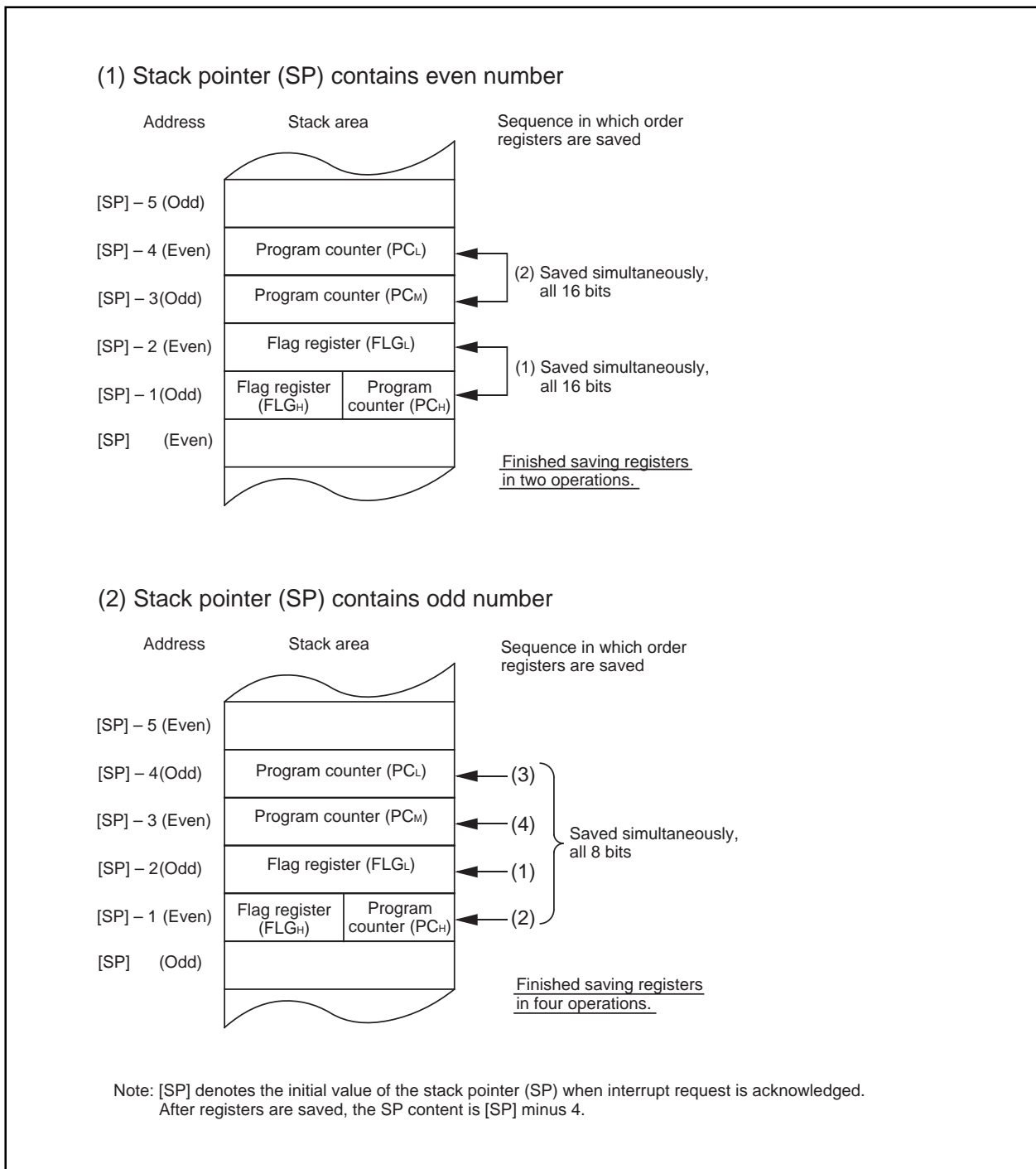


**Figure 1.27. State of stack before and after acceptance of interrupt request**

**Interrupts**

The operation of saving registers carried out in the interrupt sequence is dependent on whether the content of the stack pointer (Note), at the time of acceptance of an interrupt request, is even or odd. If the content of the stack pointer (Note) is even, the content of the flag register (FLG) and the content of the program counter (PC) are saved, 16 bits at a time. If odd, their contents are saved in two steps, 8 bits at a time. Figure 1.28 shows the operation of the saving registers.

Note: Stack pointer indicated by U flag.



**Figure 1.28. Operation of saving registers**



### Returning from an Interrupt Routine

Executing the REIT instruction at the end of an interrupt routine returns the contents of the flag register (FLG) as it was immediately before the start of interrupt sequence and the contents of the program counter (PC), both of which have been saved in the stack area. Then control returns to the program that was being executed before the acceptance of the interrupt request, so that the suspended process resumes.

Return the other registers saved by software within the interrupt routine using the POPM or similar instruction before executing the REIT instruction.

### Interrupt Priority

If there are two or more interrupt requests occurring at a point in time within a single sampling (checking whether interrupt requests are made), the interrupt assigned a higher priority is accepted.

Assign an arbitrary priority to maskable interrupts (peripheral I/O interrupts) using the interrupt priority level select bit. If the same interrupt priority level is assigned, however, the interrupt assigned a higher hardware priority is accepted.

Priorities of the special interrupts, such as Reset (dealt with as an interrupt assigned the highest priority), watchdog timer interrupt, etc. are regulated by hardware.

Figure 1.29 shows the priorities of hardware interrupts.

Software interrupts are not affected by the interrupt priority. If an instruction is executed, control branches invariably to the interrupt routine.

### Interrupt Priority Level Judge Circuit

This circuit selects the interrupt with the highest priority level when two or more interrupts are generated simultaneously.

Figure 1.30 shows the interrupt resolution circuit.

Interrupts

Reset >  $\overline{DBC}$  > Watchdog timer > Peripheral I/O > Single step > Address match

Figure 1.29. Hardware interrupts priorities

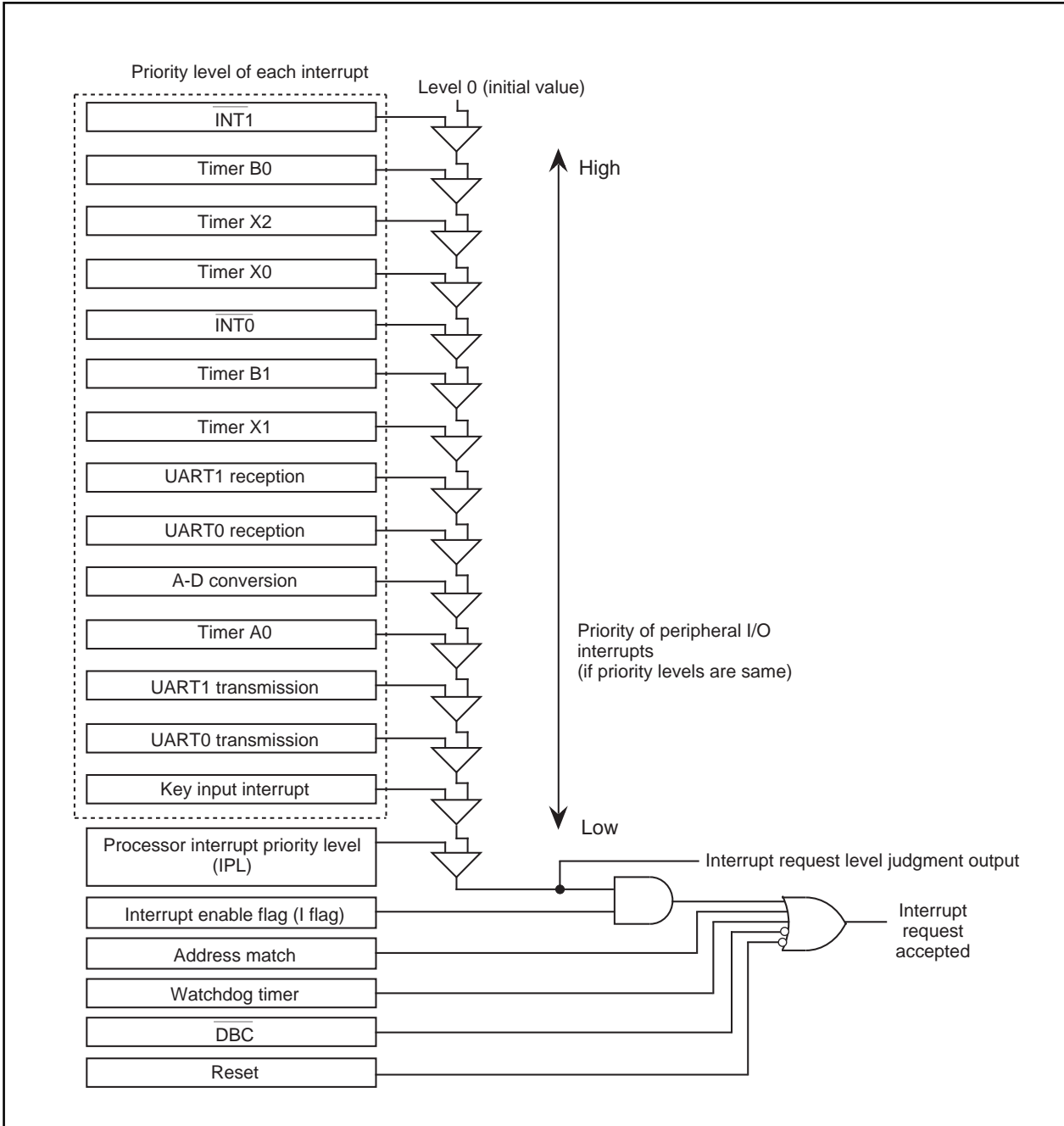


Figure 1.30. Interrupt resolution circuit

## Key Input Interrupt

### Key Input Interrupt

If the direction register of any of P00 to P07 is set for input and a falling edge is input to that port, a key input interrupt is generated. A key input interrupt can also be used as a key-on wakeup function for cancelling the wait mode or stop mode. Figure 1.31 shows the block diagram of the key input interrupt. Note that if an "L" level is input to any pin that has not been disabled for input, inputs to the other pins are not detected as an interrupt.

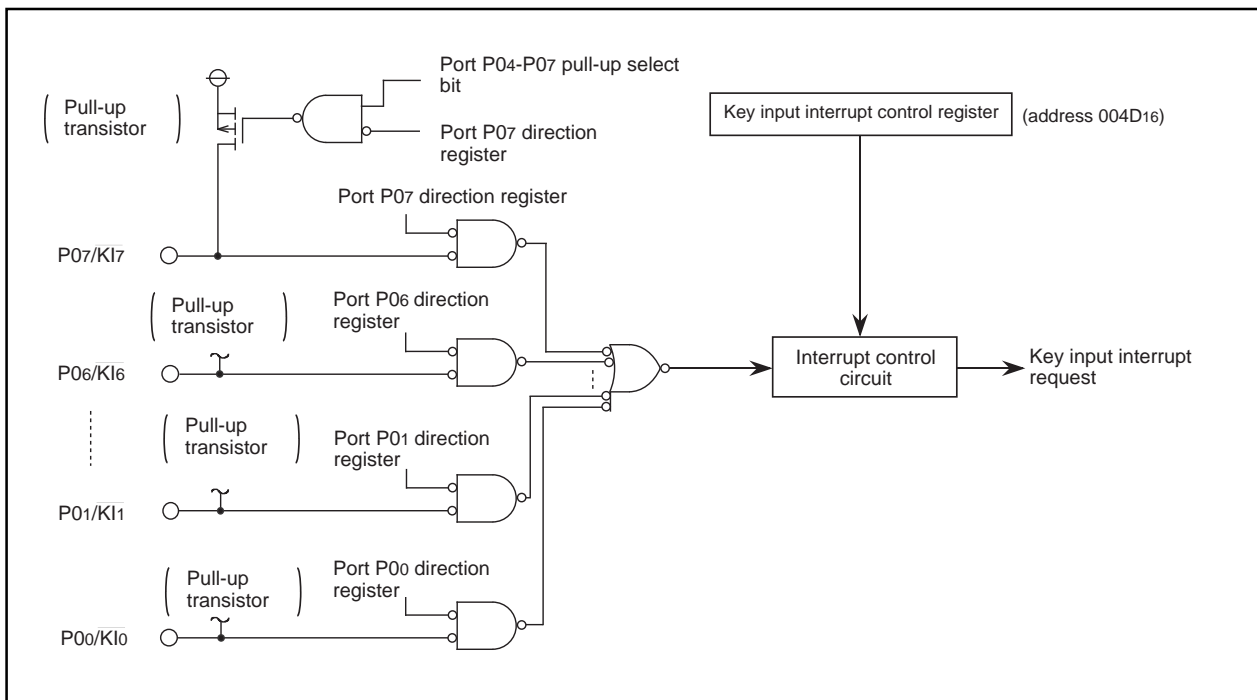


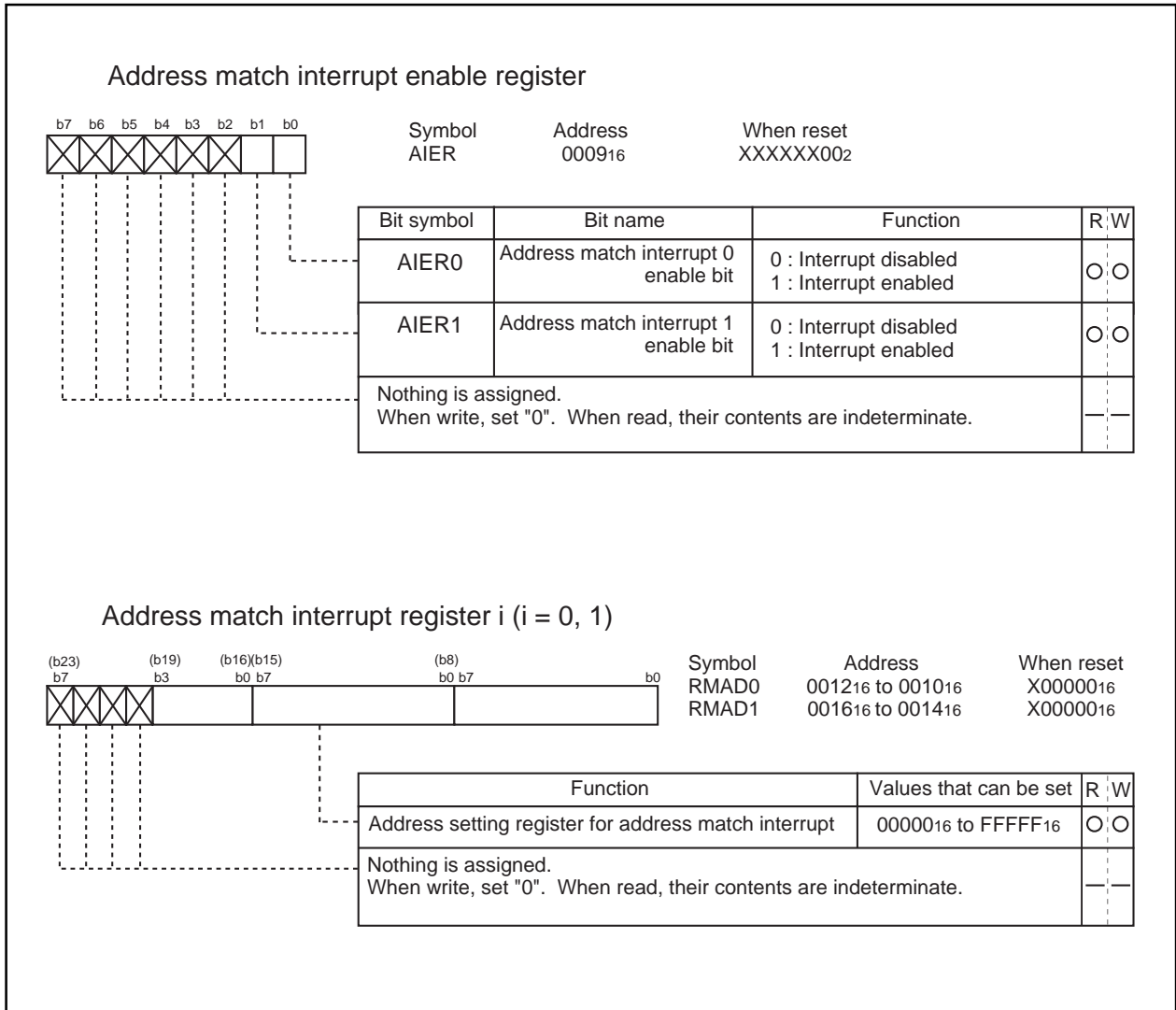
Figure 1.31. Block diagram of key input interrupt

## Address Match Interrupt

### Address Match Interrupt

An address match interrupt is generated when the address match interrupt address register contents match the program counter value. Two address match interrupts can be set, each of which can be enabled and disabled by an address match interrupt enable bit. Address match interrupts are not affected by the interrupt enable flag (I flag) and processor interrupt priority level (IPL).

Figure 1.32 shows the address match interrupt-related registers.



**Figure 1.32. Address match interrupt-related registers**

## Interrupts

**Precautions for Interrupts****(1) Reading address 00000<sub>16</sub>**

- When maskable interrupt is occurred, CPU read the interrupt information (the interrupt number and interrupt request level) in the interrupt sequence.

The interrupt request bit of the certain interrupt written in address 00000<sub>16</sub> will then be set to "0".

Reading address 00000<sub>16</sub> by software sets enabled highest priority interrupt source request bit to "0".

Though the interrupt is generated, the interrupt routine may not be executed.

Do not read address 00000<sub>16</sub> by software.

**(2) Setting the stack pointer**

- The value of the stack pointer immediately after reset is initialized to 0000<sub>16</sub>. Accepting an interrupt before setting a value in the stack pointer may become a factor of runaway. Be sure to set a value in the stack pointer before accepting an interrupt. Concerning the first instruction immediately after reset, generating any interrupts is prohibited.

**(3) External interrupt**

- Either an "L" level or an "H" level of at least 250 ns width is necessary for the signal input to pins  $\overline{\text{INT0}}$  and  $\overline{\text{INT1}}$  regardless of the CPU operation clock.
- When changing a polarity of pins  $\overline{\text{INT0}}$  and  $\overline{\text{INT1}}$ , the interrupt request bit may become "1". Clear the interrupt request bit after changing the polarity. Figure 1.33 shows the switching condition of  $\overline{\text{INT}}$  interrupt request.

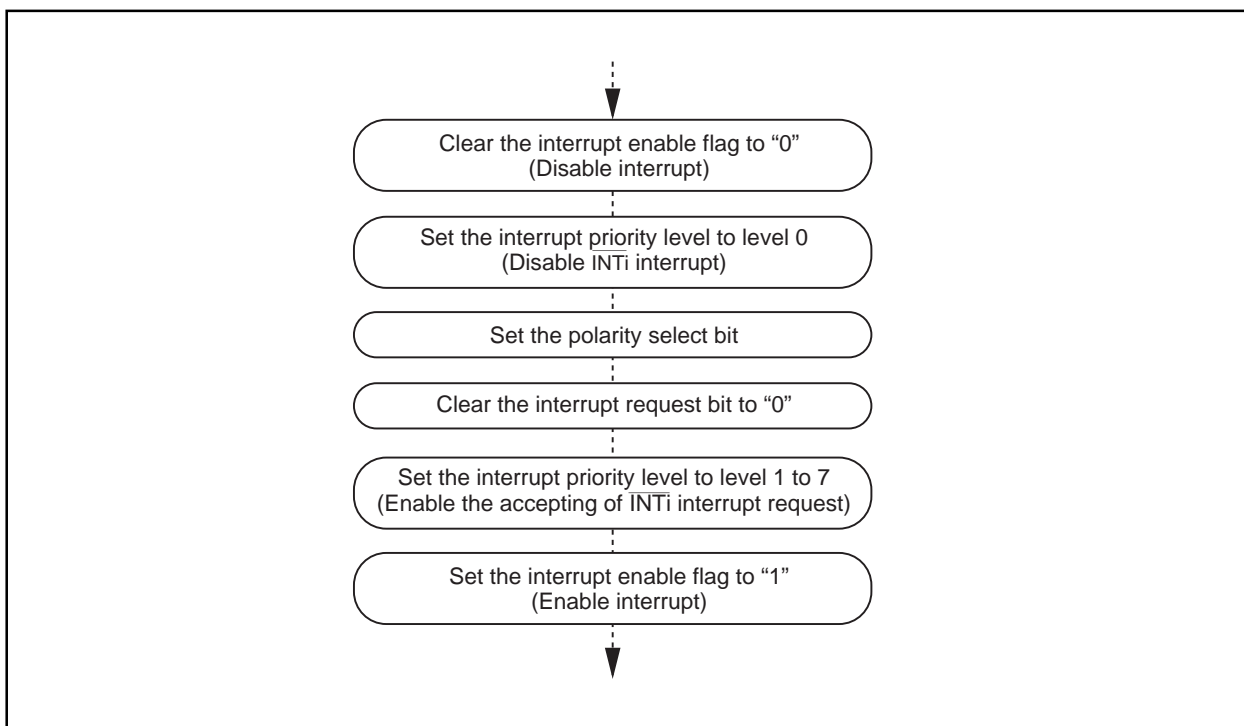


Figure 1.33. Switching condition of  $\overline{\text{INT}}$  interrupt request

**(4) Changing interrupt control register**

See "Changing Interrupt Control Register".

## Watchdog Timer

### Watchdog Timer

The watchdog timer has the function of detecting when the program is out of control. The watchdog timer is a 15-bit counter which down-counts the clock derived by dividing the BCLK using the prescaler. A watchdog timer interrupt is generated when an underflow occurs in the watchdog timer. When XIN is selected for the BCLK, bit 7 of the watchdog timer control register (address 000F<sub>16</sub>) selects the prescaler division ratio (by 16 or by 128). When XCIN is selected as the BCLK, the prescaler is set for division by 2 regardless of bit 7 of the watchdog timer control register (address 000F<sub>16</sub>).

When XIN is selected in BCLK

$$\text{Watchdog timer cycle} = \frac{\text{Prescaler division ratio (16 or 128)} \times \text{watchdog timer count (32768)}}{\text{BCLK}}$$

When XCIN is selected in BCLK

$$\text{Watchdog timer cycle} = \frac{\text{Prescaler division ratio (2)} \times \text{watchdog timer count (32768)}}{\text{BCLK}}$$

For example, when BCLK is 10MHz and the prescaler division ratio is set to 16, the watchdog timer cycle is approximately 52.4 ms.

The watchdog timer is initialized by writing to the watchdog timer start register (address 000E<sub>16</sub>) and when a watchdog timer interrupt request is generated. The prescaler is initialized only when the microcomputer is reset. After a reset is cancelled, the watchdog timer and prescaler are both stopped. The count is started by writing to the watchdog timer start register (address 000E<sub>16</sub>).

Figure 1.34 shows the block diagram of the watchdog timer. Figure 1.35 shows the watchdog timer-related registers.

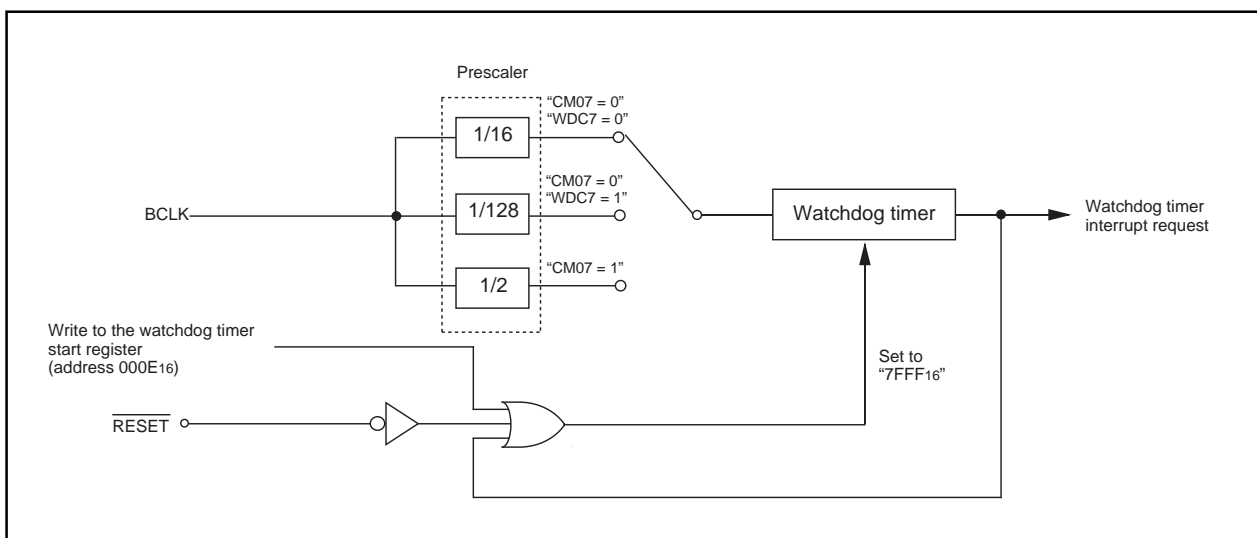


Figure 1.34. Block diagram of watchdog timer

## Watchdog Timer

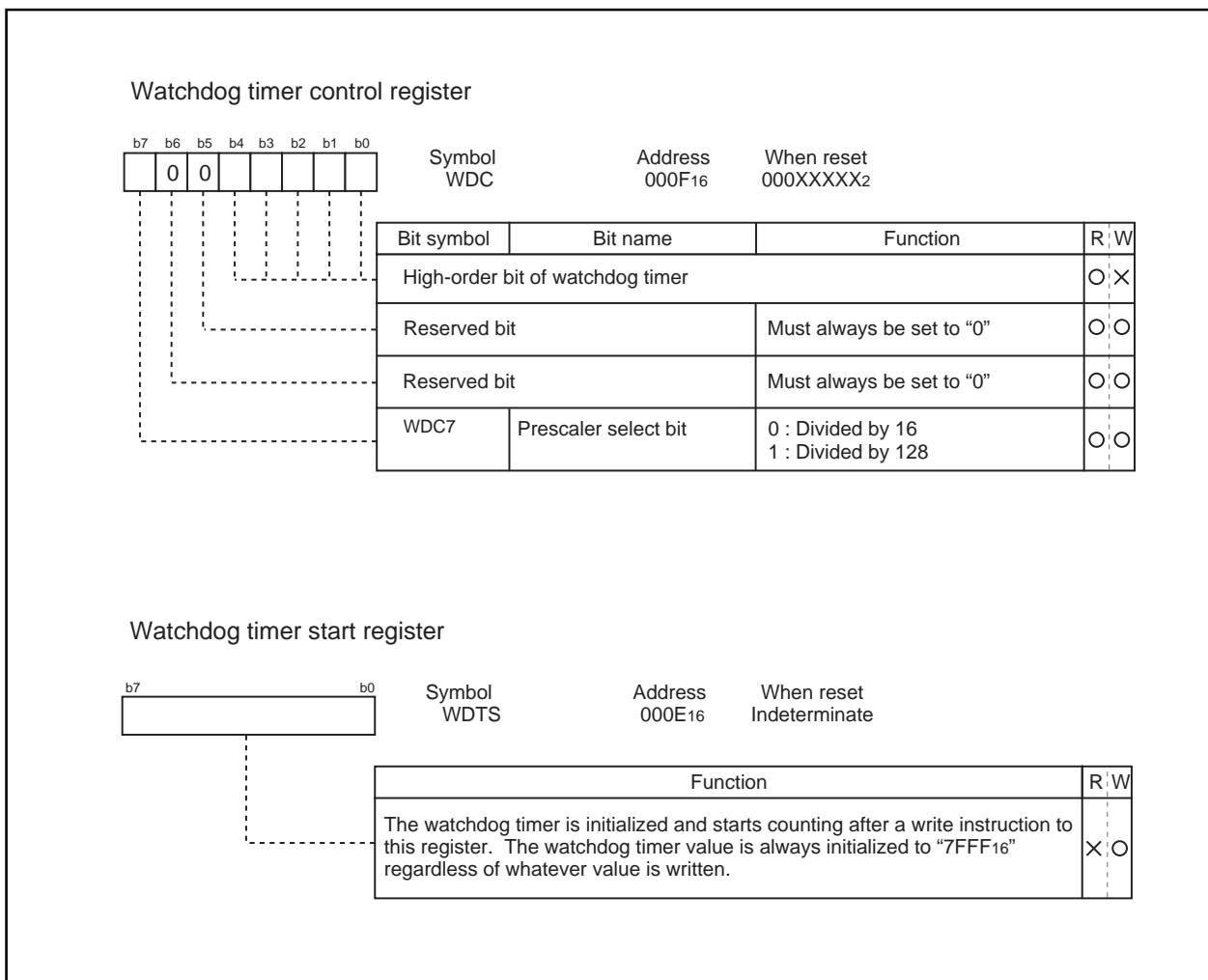


Figure 1.35. Watchdog timer control and start registers

Timer

Timer

There are six 16-bit timers. These timers can be classified by function into timer A (one), timers B (two) and timers X (three). All these timers function independently. Figure 1.36 show the block diagram of timers.

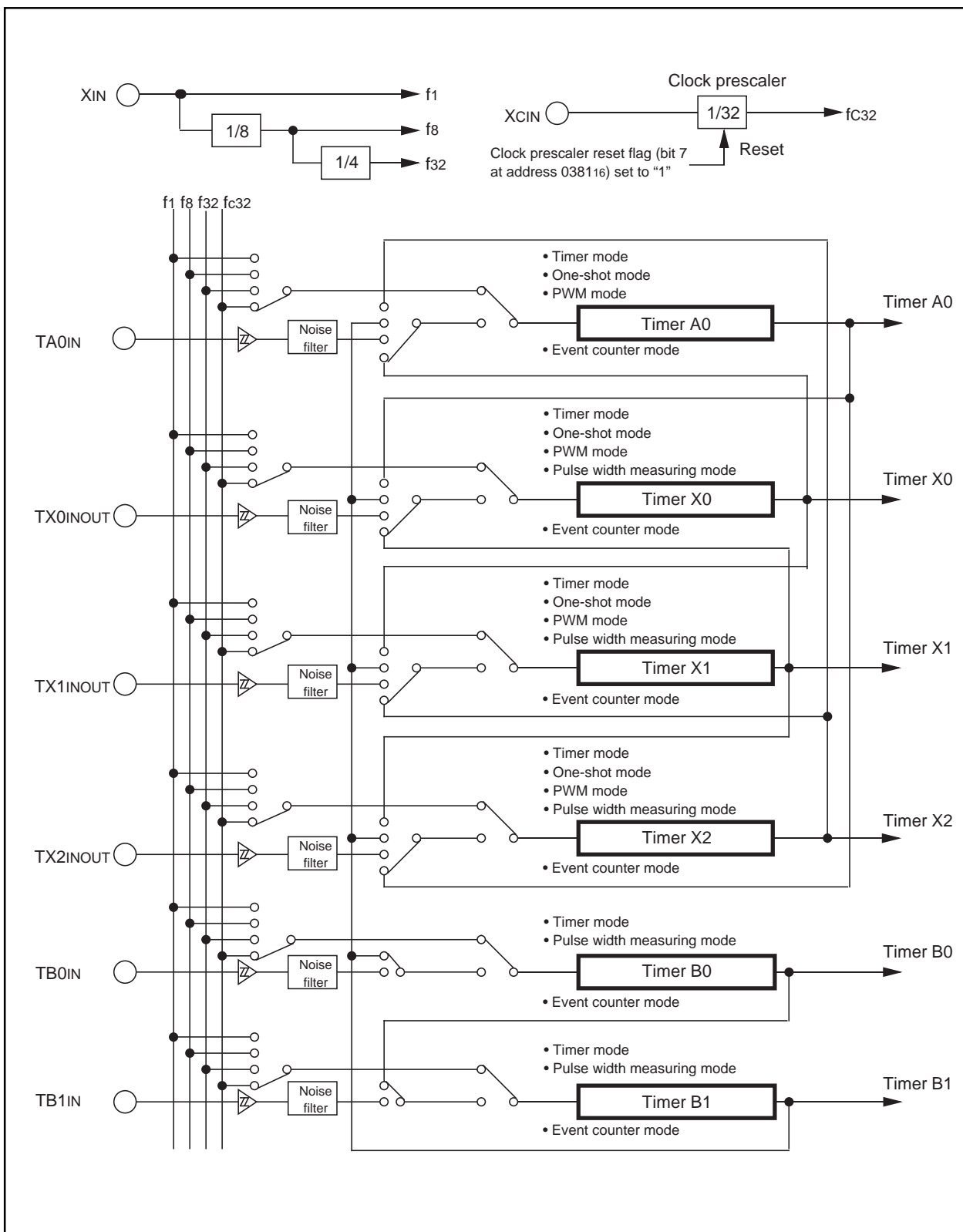


Figure 1.36. Timer block diagram



# Timer A

## Timer A

Figure 1.37 shows the block diagram of timer A. Figures 1.38 to 1.40 show the timer A-related registers. Use the timer A0 mode register bits 0 and 1 to choose the desired mode.

Timer A has the four operation modes listed as follows:

- Timer mode: The timer counts an internal count source.
- Event counter mode: The timer counts pulses from an external source or a timer overflow.
- One-shot timer mode: The timer stops counting when the count reaches "000016".
- Pulse width modulation (PWM) mode: The timer outputs pulses of a given width.

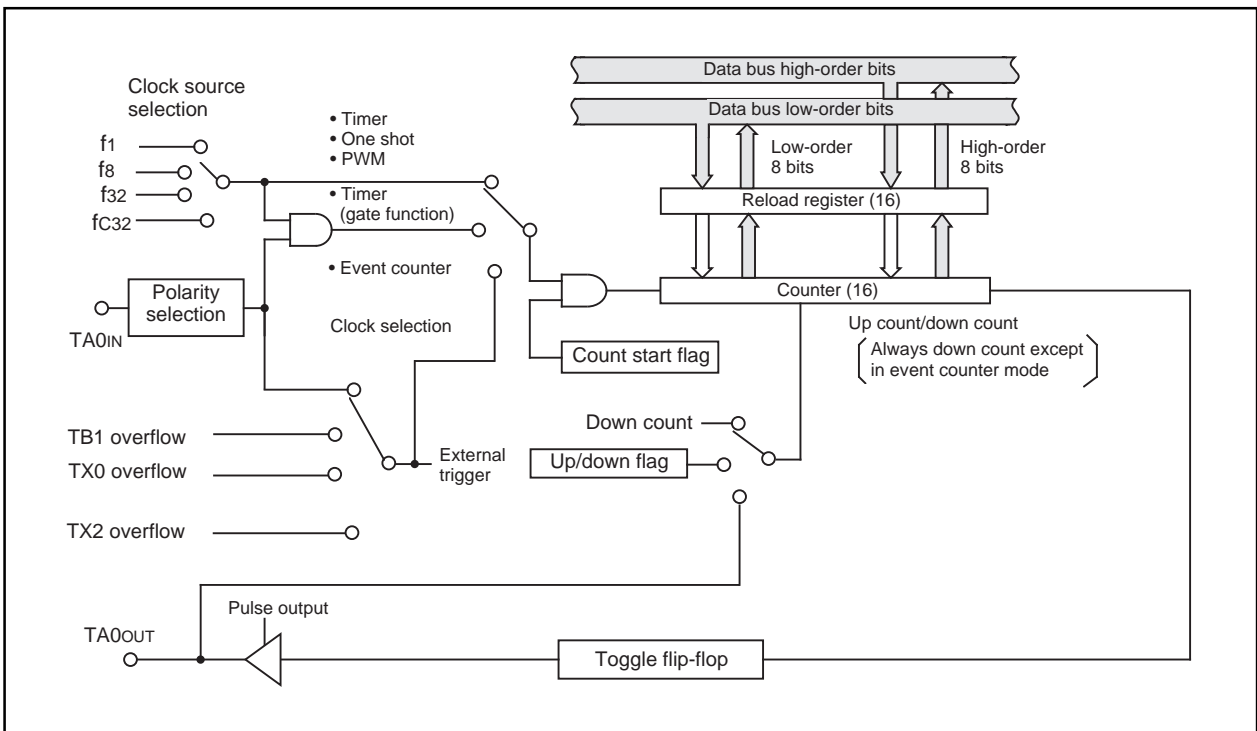


Figure 1.37. Block diagram of timer A

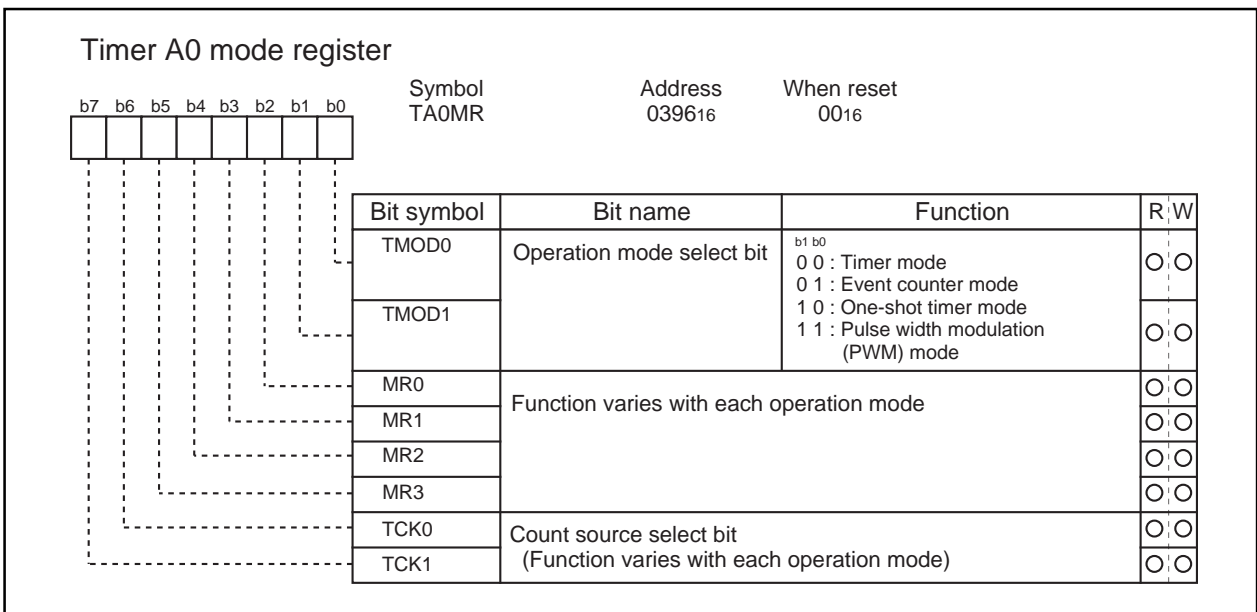


Figure 1.38. Timer A-related registers (1)

Timer A

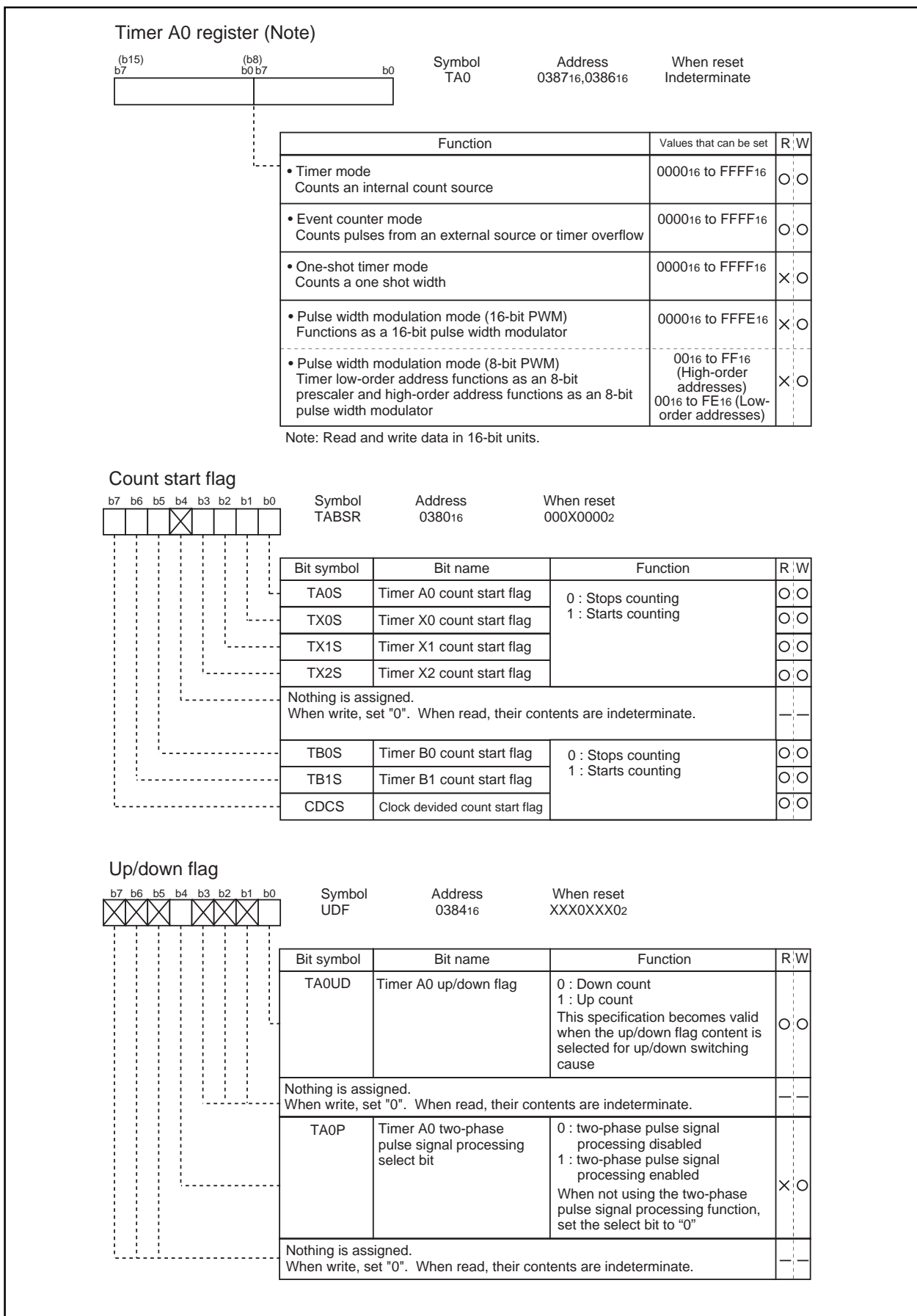


Figure 1.39. Timer A-related registers (2)

Timer A

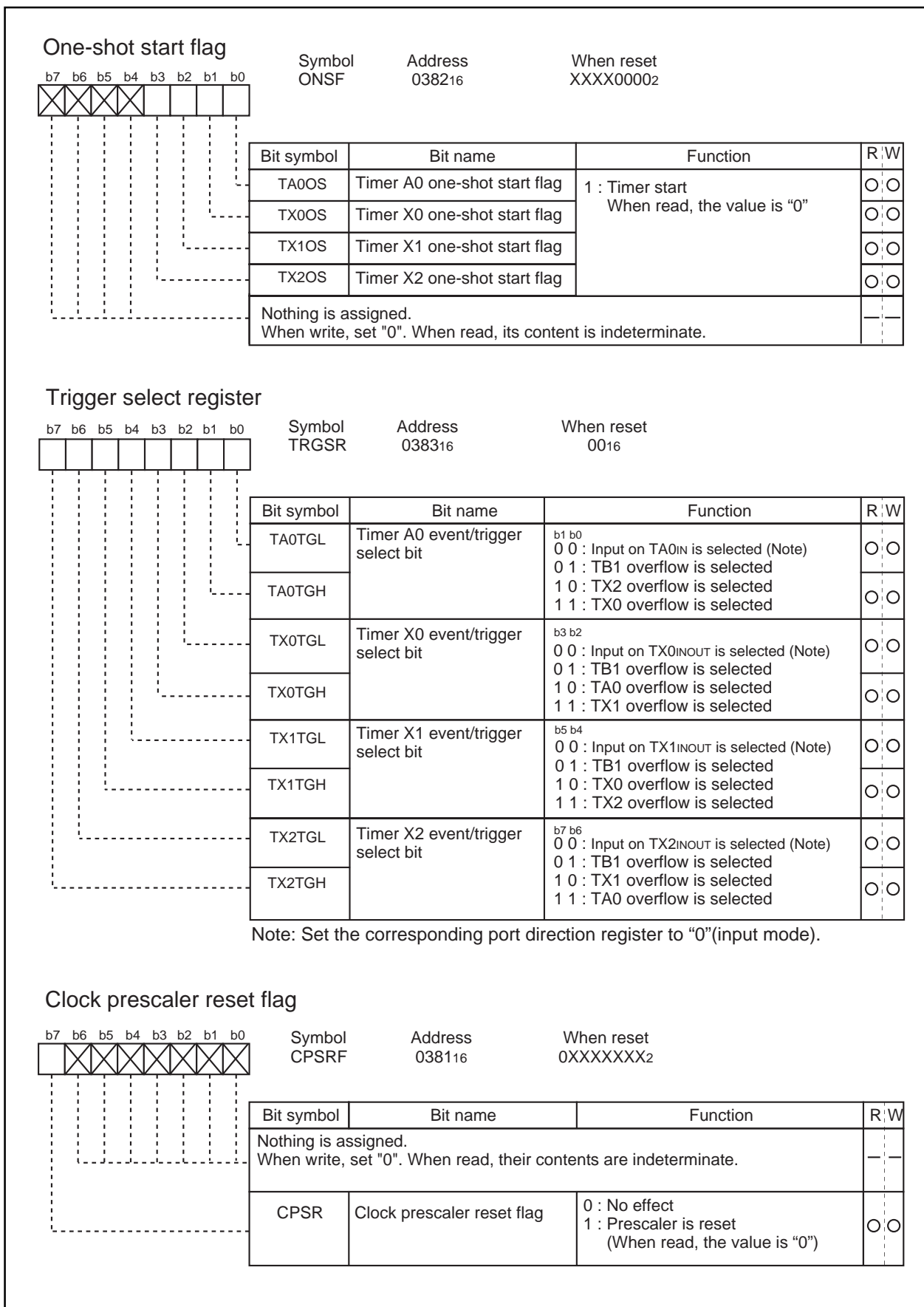


Figure 1.40. Timer A-related registers (3)

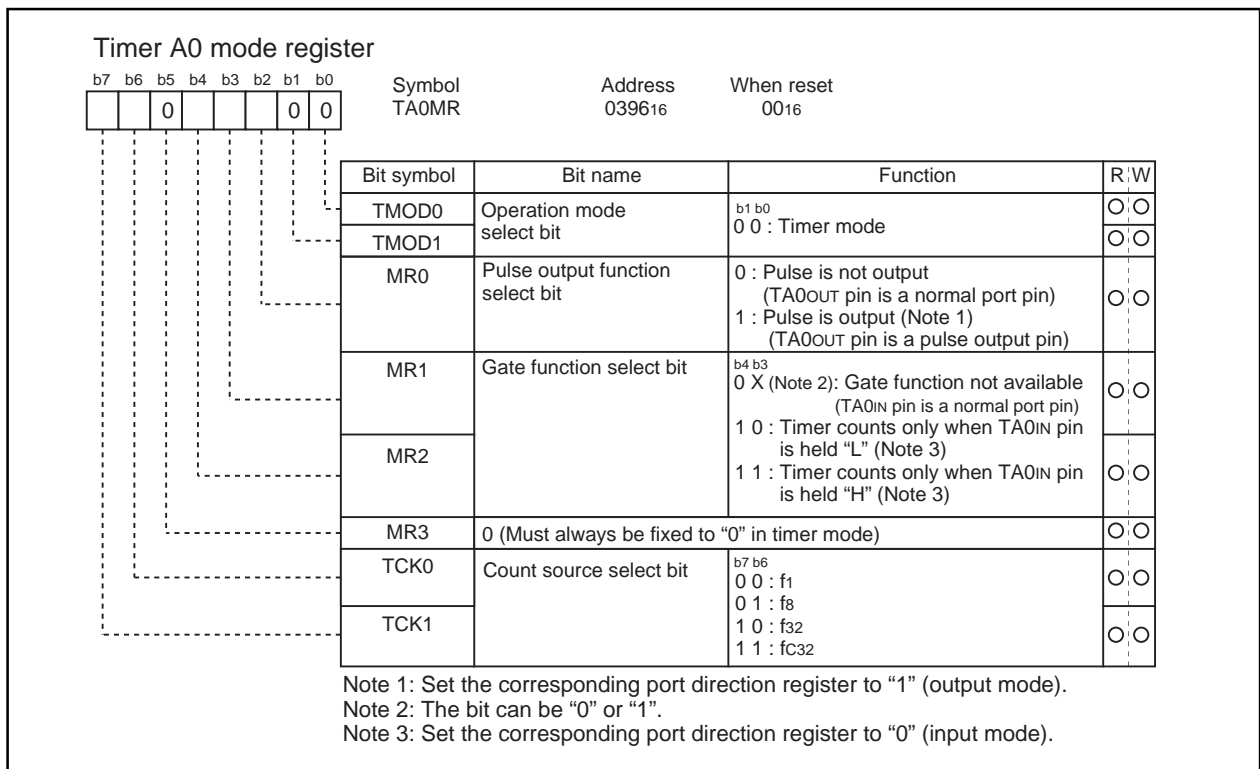
Timer A

**(1) Timer mode**

In this mode, the timer counts an internally generated count source. (See Table 1.13.) Figure 1.41 shows the timer A0 mode register in timer mode.

**Table 1.13. Specifications of timer mode**

Item	Specification
Count source	f1, f8, f32, fc32
Count operation	<ul style="list-style-type: none"> <li>• Down count</li> <li>• When the timer underflows, it reloads the reload register contents before continuing counting</li> </ul>
Divide ratio	1/(n+1) n : Set value
Count start condition	Count start flag is set (= 1)
Count stop condition	Count start flag is reset (= 0)
Interrupt request generation timing	When the timer underflows
TA0IN pin function	Programmable I/O port or gate input
TA0OUT pin function	Programmable I/O port or pulse output
Read from timer	Count value can be read out by reading timer A0 register
Write to timer	<ul style="list-style-type: none"> <li>• When counting stopped When a value is written to timer A0 register, it is written to both reload register and counter</li> <li>• When counting in progress When a value is written to timer A0 register, it is written to only reload register (Transferred to counter at next reload time)</li> </ul>
Select function	<ul style="list-style-type: none"> <li>• Gate function Counting can be started and stopped by the TA0IN pin's input signal</li> <li>• Pulse output function Each time the timer underflows, the TA0OUT pin's polarity is reversed</li> </ul>



**Figure 1.41. Timer A0 mode register in timer mode**

## Timer A

**(2) Event counter mode**

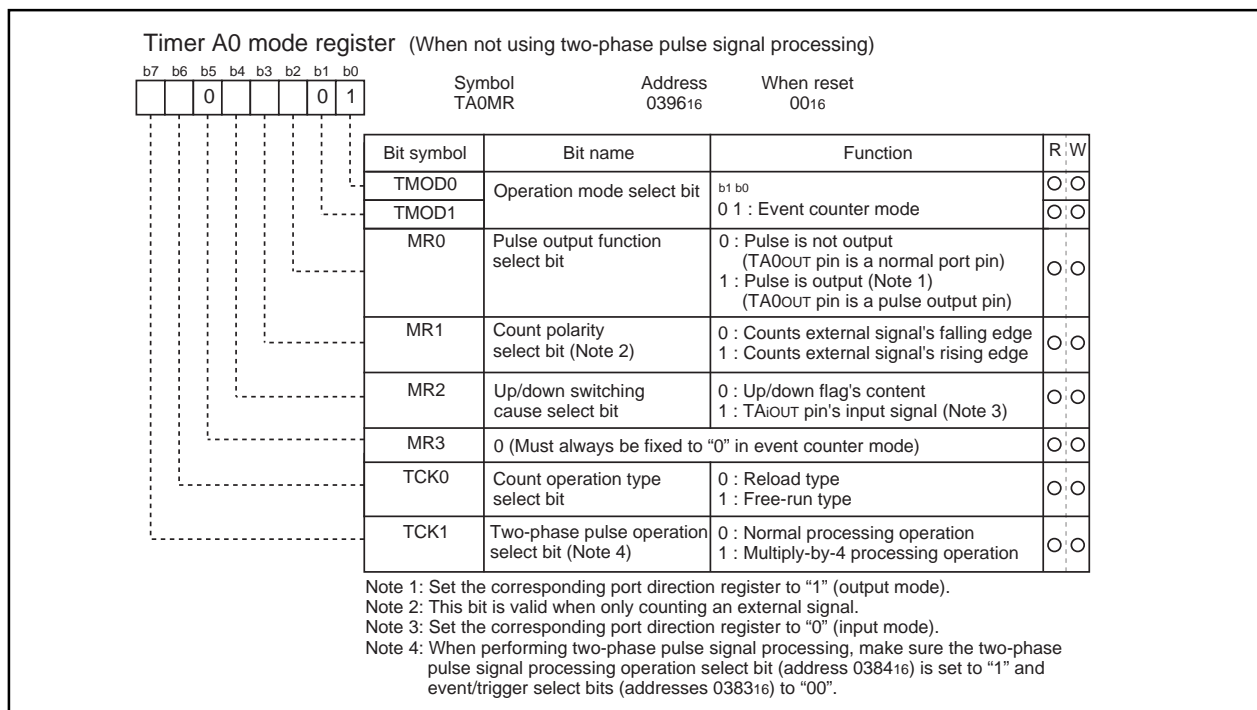
In this mode, the timer counts an external signal or an internal timer's overflow. Timer A0 can count a single-phase and a two-phase external signal. Table 1.14 lists timer specifications when counting a single-phase external signal. Figure 1.42 shows the timer A0 mode register in event counter mode.

Table 1.15 lists timer specifications when counting a two-phase external signal. Figure 1.43 shows the timer A0 mode register in event counter mode.

**Table 1.14. Timer specifications in event counter mode (when not processing two-phase pulse signal)**

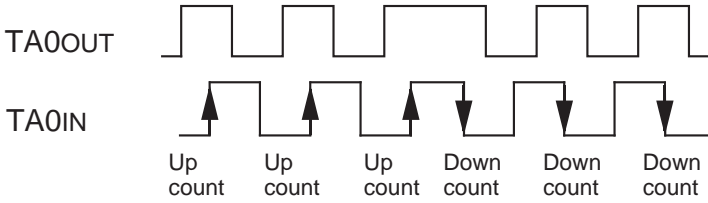
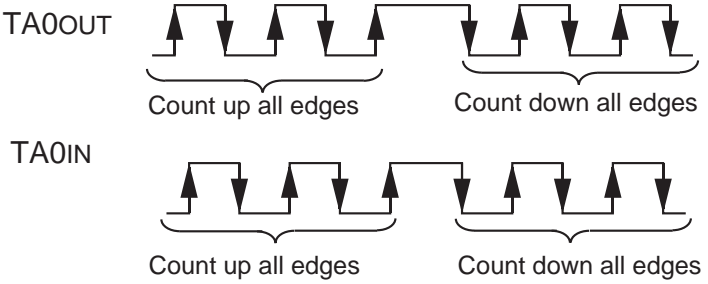
Item	Specification
Count source	<ul style="list-style-type: none"> <li>External signals input to TA0IN pin (effective edge can be selected by software)</li> <li>TB1 overflow, TX0 overflow, TX2 overflow</li> </ul>
Count operation	<ul style="list-style-type: none"> <li>Up count or down count can be selected by external signal or software</li> <li>When the timer overflows or underflows, it reloads the reload register contents before continuing counting (Note)</li> </ul>
Divide ratio	$1 / (FFFF_{16} - n + 1)$ for up count $1 / (n + 1)$ for down count      n : Set value
Count start condition	Count start flag is set (= 1)
Count stop condition	Count start flag is reset (= 0)
Interrupt request generation timing	The timer overflows or underflows
TA0IN pin function	Programmable I/O port or count source input
TA0OUT pin function	Programmable I/O port, pulse output, or up/down count select input
Read from timer	Count value can be read out by reading timer A0 register
Write to timer	<ul style="list-style-type: none"> <li>When counting stopped When a value is written to timer A0 register, it is written to both reload register and counter</li> <li>When counting in progress When a value is written to timer A0 register, it is written to only reload register (Transferred to counter at next reload time)</li> </ul>
Select function	<ul style="list-style-type: none"> <li>Free-run count function Even when the timer overflows or underflows, the reload register content is not reloaded to it</li> <li>Pulse output function Each time the timer overflows or underflows, the TA0OUT pin's polarity is reversed</li> </ul>

Note: This does not apply when the free-run function is selected.

**Figure 1.42. Timer A0 mode register in event counter mode**

Timer A

**Table 1.15. Timer specifications in event counter mode (when processing two-phase pulse signal)**

Item	Specification
Count source	<ul style="list-style-type: none"> <li>Two-phase pulse signals input to TA0IN or TA0OUT pin</li> </ul>
Count operation	<ul style="list-style-type: none"> <li>Up count or down count can be selected by two-phase pulse signal</li> <li>When the timer overflows or underflows, the reload register content is reloaded and the timer starts over again (Note)</li> </ul>
Divide ratio	<ul style="list-style-type: none"> <li><math>1 / (FFFF_{16} - n + 1)</math> for up count</li> <li><math>1 / (n + 1)</math> for down count</li> </ul> <p style="text-align: right;">n : Set value</p>
Count start condition	Count start flag is set (= 1)
Count stop condition	Count start flag is reset (= 0)
Interrupt request generation timing	Timer overflows or underflows
TA0IN pin function	Two-phase pulse input
TA0OUT pin function	Two-phase pulse input
Read from timer	Count value can be read out by reading timer A0 register
Write to timer	<ul style="list-style-type: none"> <li>When counting stopped When a value is written to timer A0 register, it is written to both reload register and counter</li> <li>When counting in progress When a value is written to timer A0 register, it is written to only reload register. (Transferred to counter at next reload time.)</li> </ul>
Select function	<ul style="list-style-type: none"> <li>Normal processing operation The timer counts up rising edges or counts down falling edges on the TA0IN pin when input signal on the TA0OUT pin is "H"</li> </ul> <div style="text-align: center;">  </div> <ul style="list-style-type: none"> <li>Multiply-by-4 processing operation If the phase relationship is such that the TA0IN pin goes "H" when the input signal on the TA0OUT pin is "H", the timer counts up rising and falling edges on the TA0OUT and TA0IN pins. If the phase relationship is such that the TA0IN pin goes "L" when the input signal on the TA0OUT pin is "H", the timer counts down rising and falling edges on the TA0OUT and TA0IN pins.</li> </ul> <div style="text-align: center;">  </div>

Note: This does not apply when the free-run function is selected.

Timer A

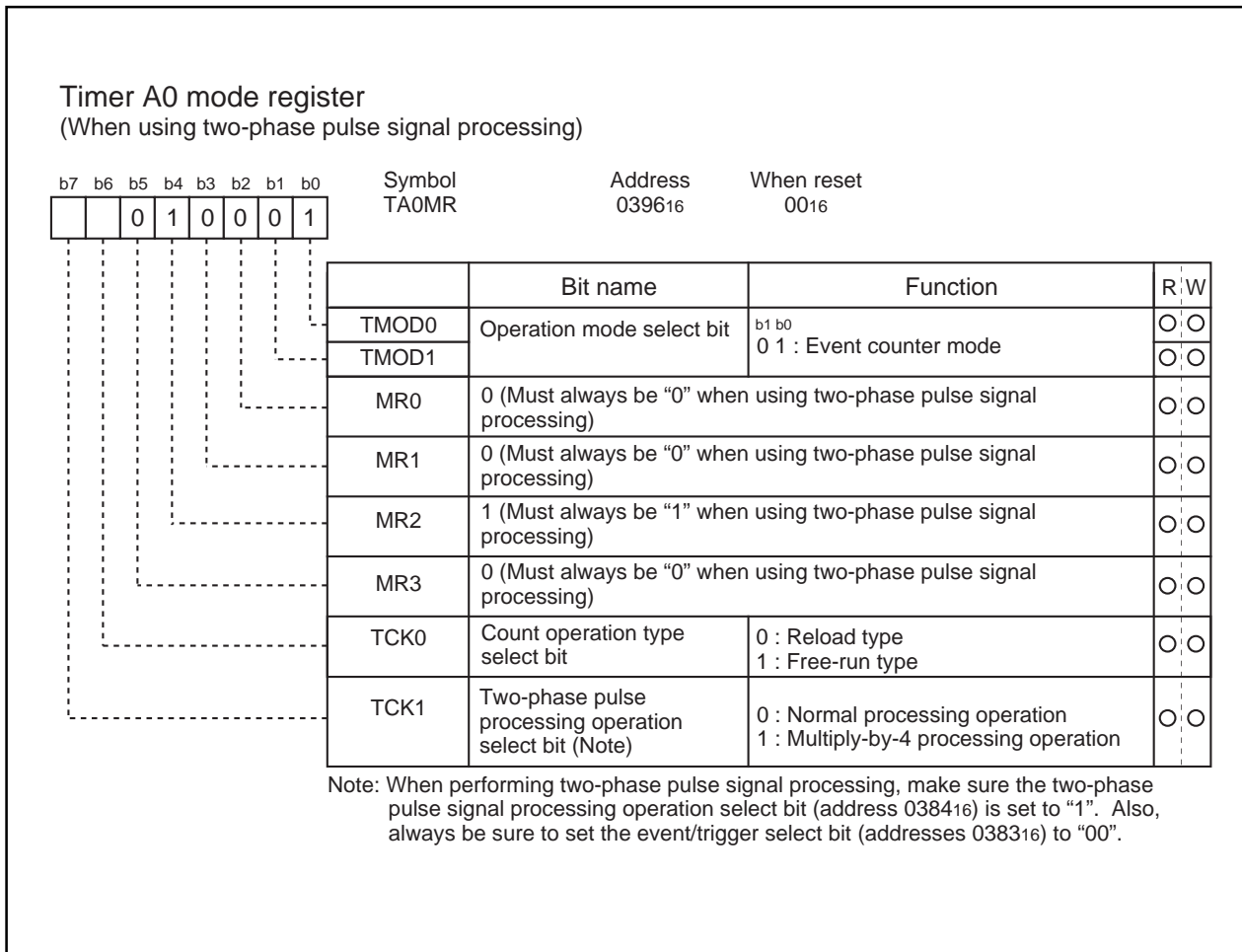


Figure 1.43. Timer A0 mode register in event counter mode

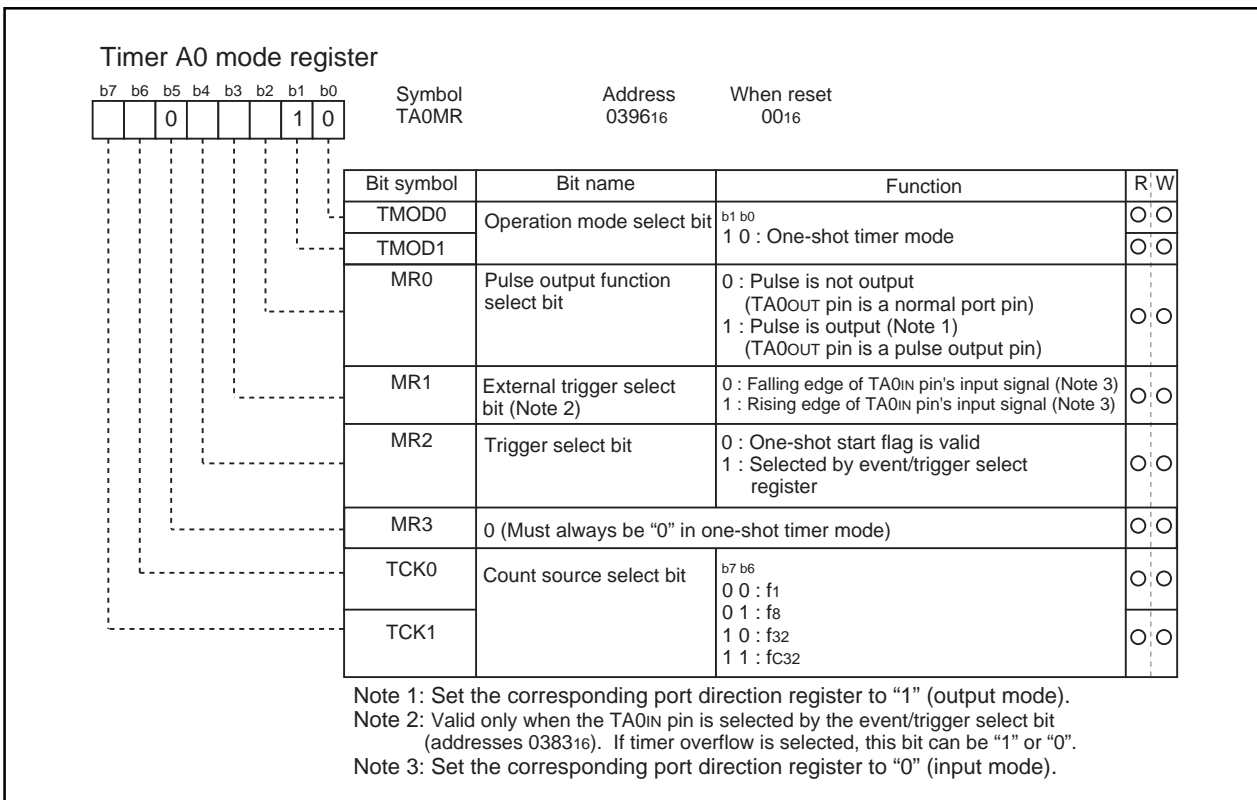
Timer A

**(3) One-shot timer mode**

In this mode, the timer operates only once. (See Table 1.16.) When a trigger occurs, the timer starts up and continues operating for a given period. Figure 1.44 shows the timer A0 mode register in one-shot timer mode.

**Table 1.16. Timer specifications in one-shot timer mode**

Item	Specification
Count source	f1, f8, f32, fc32
Count operation	<ul style="list-style-type: none"> <li>The timer counts down</li> <li>When the count reaches 0000<sub>16</sub>, the timer stops counting after reloading a new count</li> <li>If a trigger occurs when counting, the timer reloads a new count and restarts counting</li> </ul>
Divide ratio	1/n    n : Set value
Count start condition	<ul style="list-style-type: none"> <li>An external trigger is input</li> <li>The timer overflows</li> <li>The one-shot start flag is set (= 1)</li> </ul>
Count stop condition	<ul style="list-style-type: none"> <li>A new count is reloaded after the count has reached 0000<sub>16</sub></li> <li>The count start flag is reset (= 0)</li> </ul>
Interrupt request generation timing	The count reaches 0000 <sub>16</sub>
TA0IN pin function	Programmable I/O port or trigger input
TA0OUT pin function	Programmable I/O port or pulse output
Read from timer	When timer A0 register is read, it indicates an indeterminate value
Write to timer	<ul style="list-style-type: none"> <li>When counting stopped When a value is written to timer A0 register, it is written to both reload register and counter</li> <li>When counting in progress When a value is written to timer A0 register, it is written to only reload register (Transferred to counter at next reload time)</li> </ul>



**Figure 1.44. Timer A0 mode register in one-shot timer mode**



Timer A

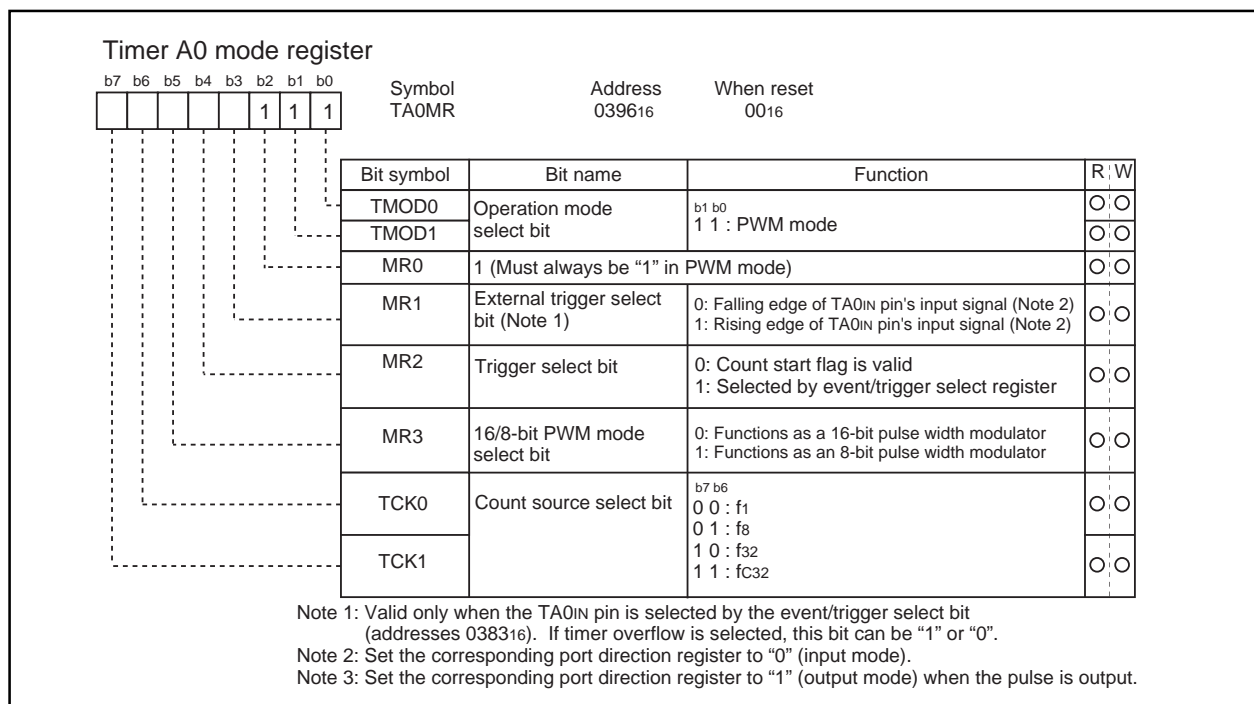
**(4) Pulse width modulation (PWM) mode**

In this mode, the timer outputs pulses of a given width in succession. (See Table 1.17.) In this mode, the counter functions as either a 16-bit pulse width modulator or an 8-bit pulse width modulator. Figure 1.45 shows the timer A0 mode register in pulse width modulation mode. Figure 1.46 shows the example of how a 16-bit pulse width modulator operates. Figure 1.47 shows the example of how an 8-bit pulse width modulator operates.

**Table 1.17. Timer specifications in pulse width modulation mode**

Item		Specification
Count source		f1, f8, f32, fc32
Count operation		<ul style="list-style-type: none"> <li>The timer counts down (operating as an 8-bit or a 16-bit pulse width modulator)</li> <li>The timer reloads a new count at a rising edge of PWM pulse and continues counting</li> <li>The timer is not affected by a trigger that occurs when counting</li> </ul>
16-bit PWM		<ul style="list-style-type: none"> <li>High level width <math>n / f_i</math> n : Set value</li> <li>Cycle time <math>(2^{16}-1) / f_i</math> fixed</li> </ul>
8-bit PWM		<ul style="list-style-type: none"> <li>High level width <math>n \times (m+1) / f_i</math> n : values set to timer A0 register's high-order address</li> <li>Cycle time <math>(2^8-1) \times (m+1) / f_i</math> m : values set to timer A0 register's low-order address</li> </ul>
Count start condition		<ul style="list-style-type: none"> <li>External trigger is input</li> <li>The timer overflows</li> <li>The count start flag is set (= 1)</li> </ul>
Count stop condition		<ul style="list-style-type: none"> <li>The count start flag is reset (= 0)</li> </ul>
Interrupt request generation timing	8 bits PWM	<ul style="list-style-type: none"> <li>Set value of "H" level width is except FF<sub>16</sub>, 00<sub>16</sub> : PWM pulse goes "L"</li> <li>Set value of "H" level width is FF<sub>16</sub>, 00<sub>16</sub> : Timing that count value goes to 01<sub>16</sub></li> </ul>
	16 bits PWM	<ul style="list-style-type: none"> <li>Set value of "H" level width is except FFFF<sub>16</sub>, 0000<sub>16</sub> : PWM pulse goes "L"</li> <li>Set value of "H" level width is FFFF<sub>16</sub>, 0000<sub>16</sub> : Timing that count value goes to 0001<sub>16</sub></li> </ul>
TA0IN pin function		Programmable I/O port or trigger input
TA0OUT pin function		Pulse output
Read from timer		When timer A0 register is read, it indicates an indeterminate value
Write to timer		<ul style="list-style-type: none"> <li>When counting stopped : When a value is written to timer A0 register, it is written to both reload register and counter</li> <li>When counting in progress : When a value is written to timer A0 register, it is written to only reload register (Transferred to counter at next reload time)</li> </ul>

Note: When set value of "H" level width is 00<sub>16</sub> or 0000<sub>16</sub>, pulse outputs "L" level and inversion value, FF<sub>16</sub> or FFFF<sub>16</sub> is set to timer.



**Figure 1.45. Timer A0 mode register in pulse width modulation mode**

Timer A

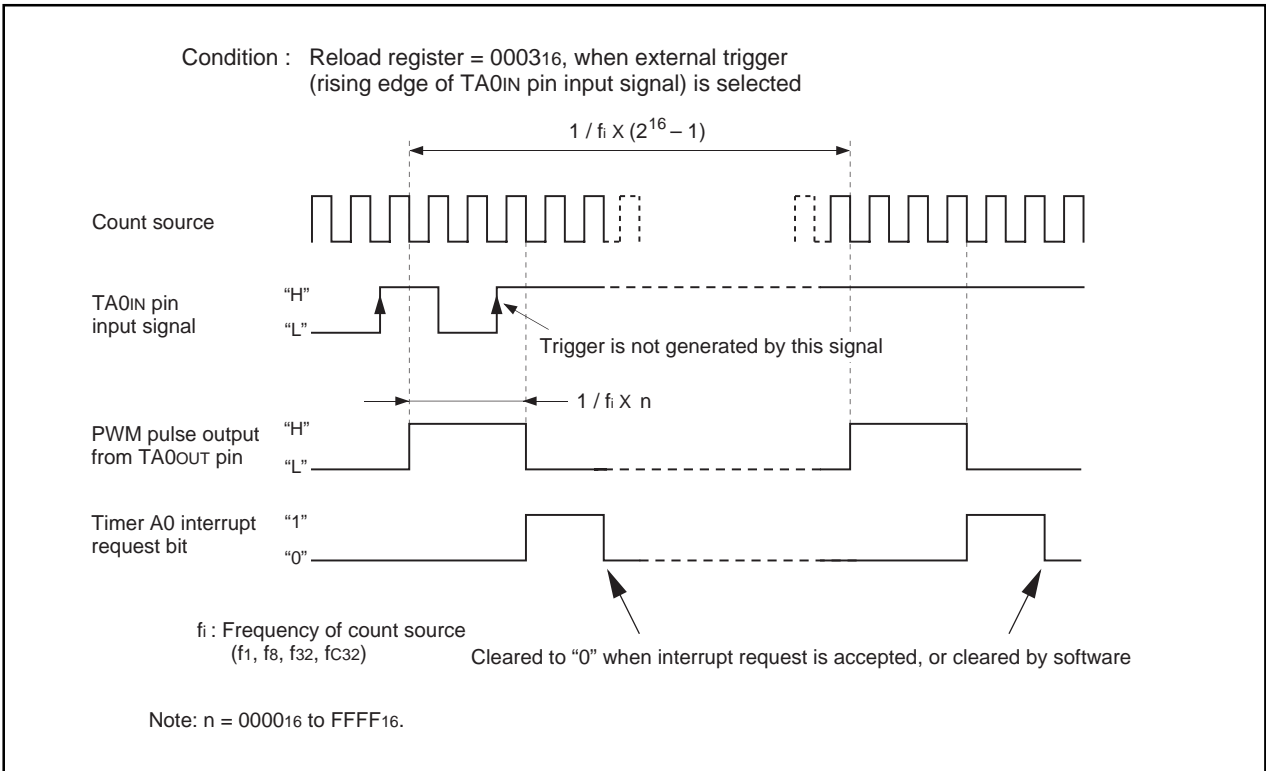


Figure 1.46. Example of how a 16-bit pulse width modulator operates

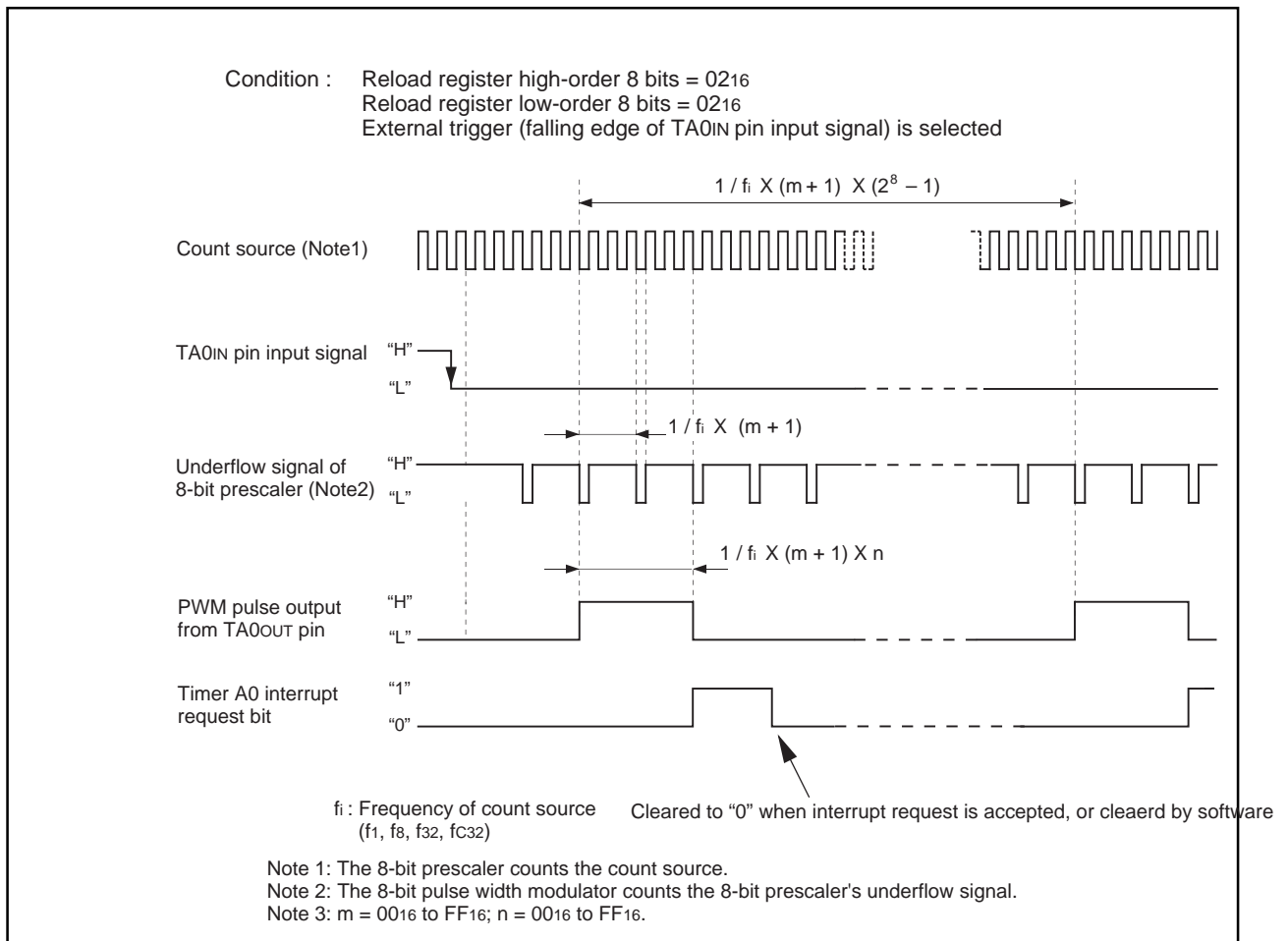


Figure 1.47. Example of how an 8-bit pulse width modulator operates

## Timer B

### Timer B

Figure 1.48 shows the block diagram of timer B. Figures 1.49 and 1.50 show the timer B-related registers. Use the timer Bi mode register (i = 0, 1) bits 0 and 1 to choose the desired mode.

Timer B has three operation modes listed as follows:

- Timer mode : The timer counts an internal count source.
- Event counter mode : The timer counts pulses from an external source or a timer overflow.
- Pulse period/pulse width measuring mode : The timer measures an external signal's pulse period or pulse width.

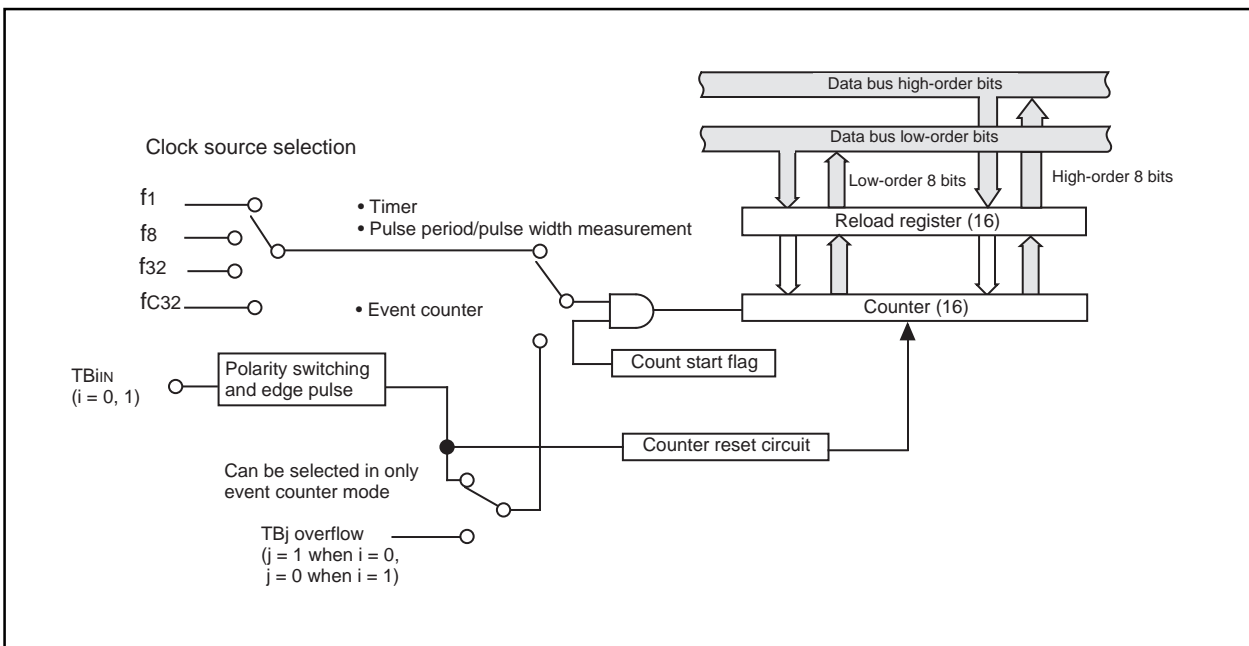


Figure 1.48. Block diagram of timer B

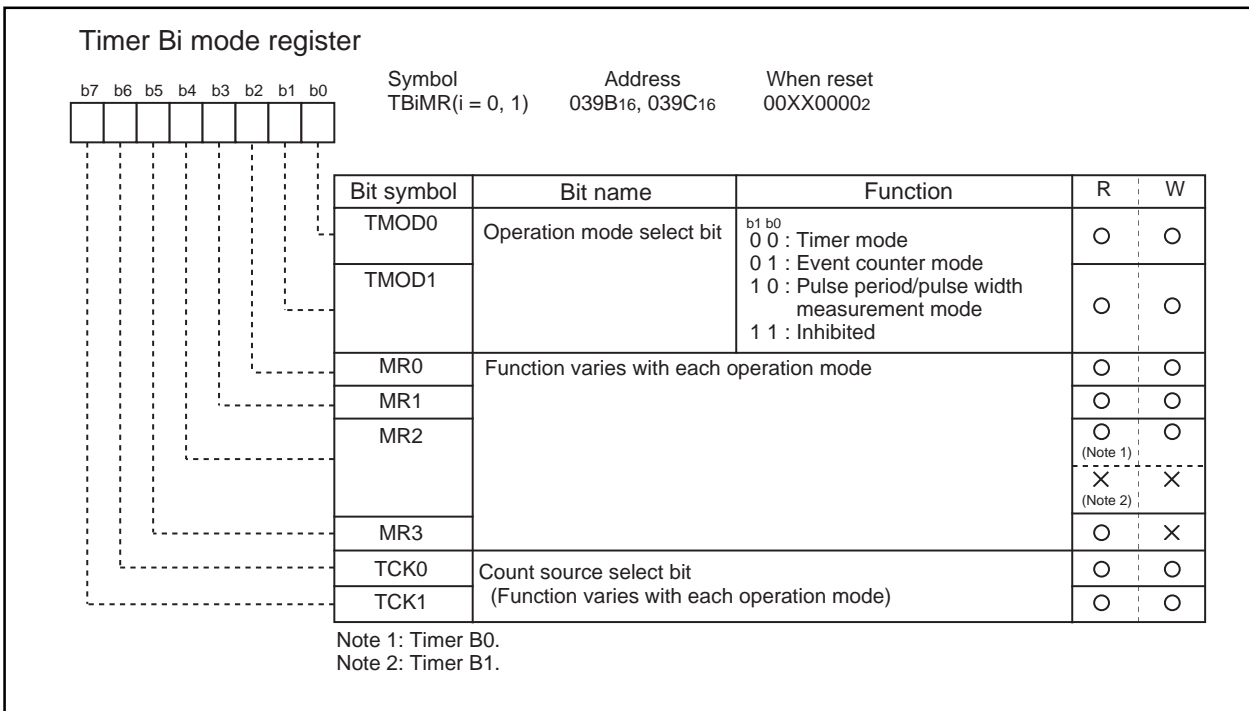


Figure 1.49. Timer B-related registers (1)

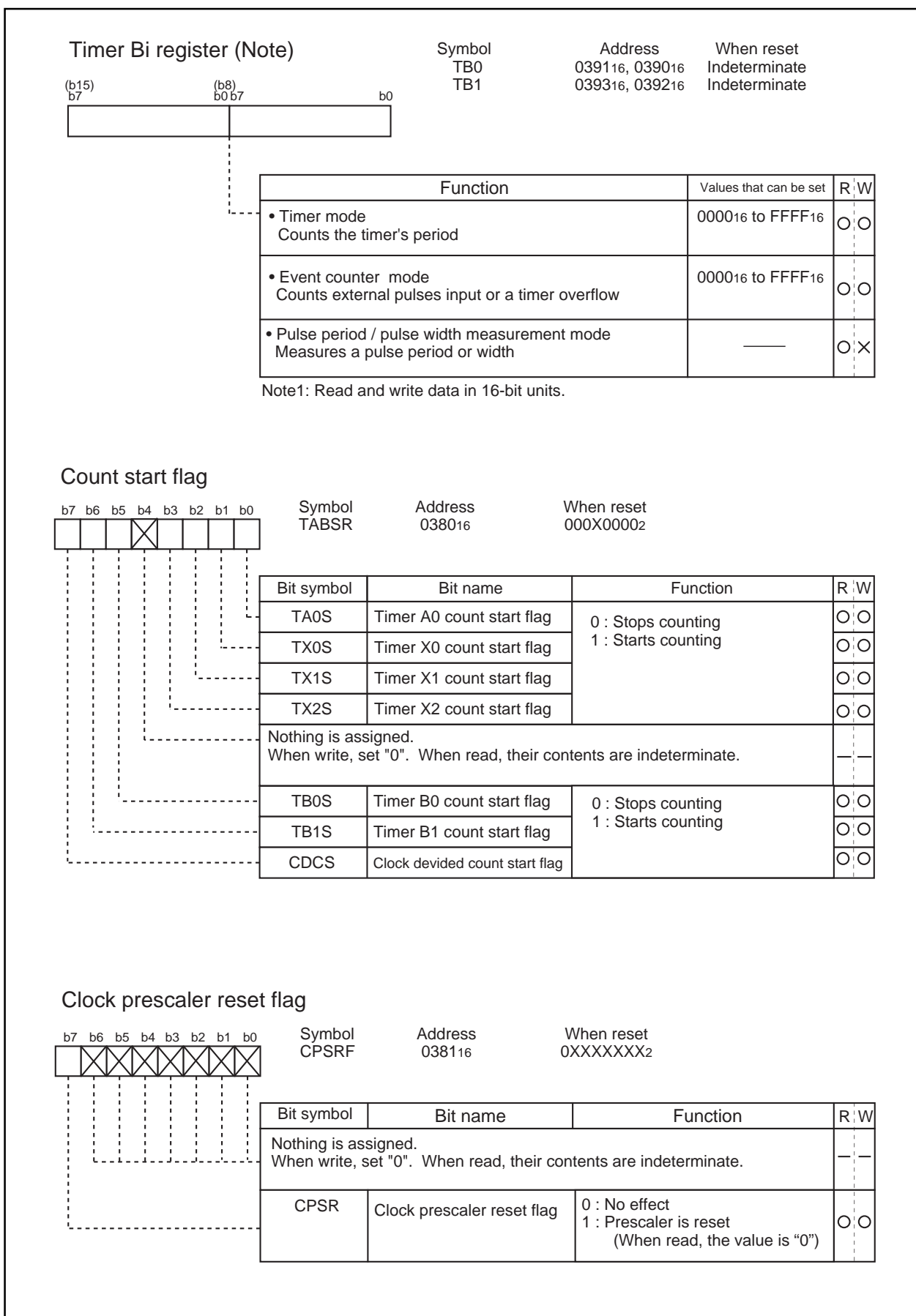


Figure 1.50. Timer B-related registers (2)

Timer B

(1) Timer mode

In this mode, the timer counts an internally generated count source. (See Table 1.18.) Figure 1.51 shows the timer Bi mode register in timer mode.

Table 1.18. Timer specifications in timer mode

Item	Specification
Count source	f1, f8, f32, fC32
Count operation	<ul style="list-style-type: none"> <li>Counts down</li> <li>When the timer underflows, it reloads the reload register contents before continuing counting</li> </ul>
Divide ratio	1/(n+1) n : Set value
Count start condition	Count start flag is set (= 1)
Count stop condition	Count start flag is reset (= 0)
Interrupt request generation timing	The timer underflows
TBiIN pin function	Programmable I/O port
Read from timer	Count value is read out by reading timer Bi register
Write to timer	<ul style="list-style-type: none"> <li>When counting stopped When a value is written to timer Bi register, it is written to both reload register and counter</li> <li>When counting in progress When a value is written to timer Bi register, it is written to only reload register (Transferred to counter at next reload time)</li> </ul>

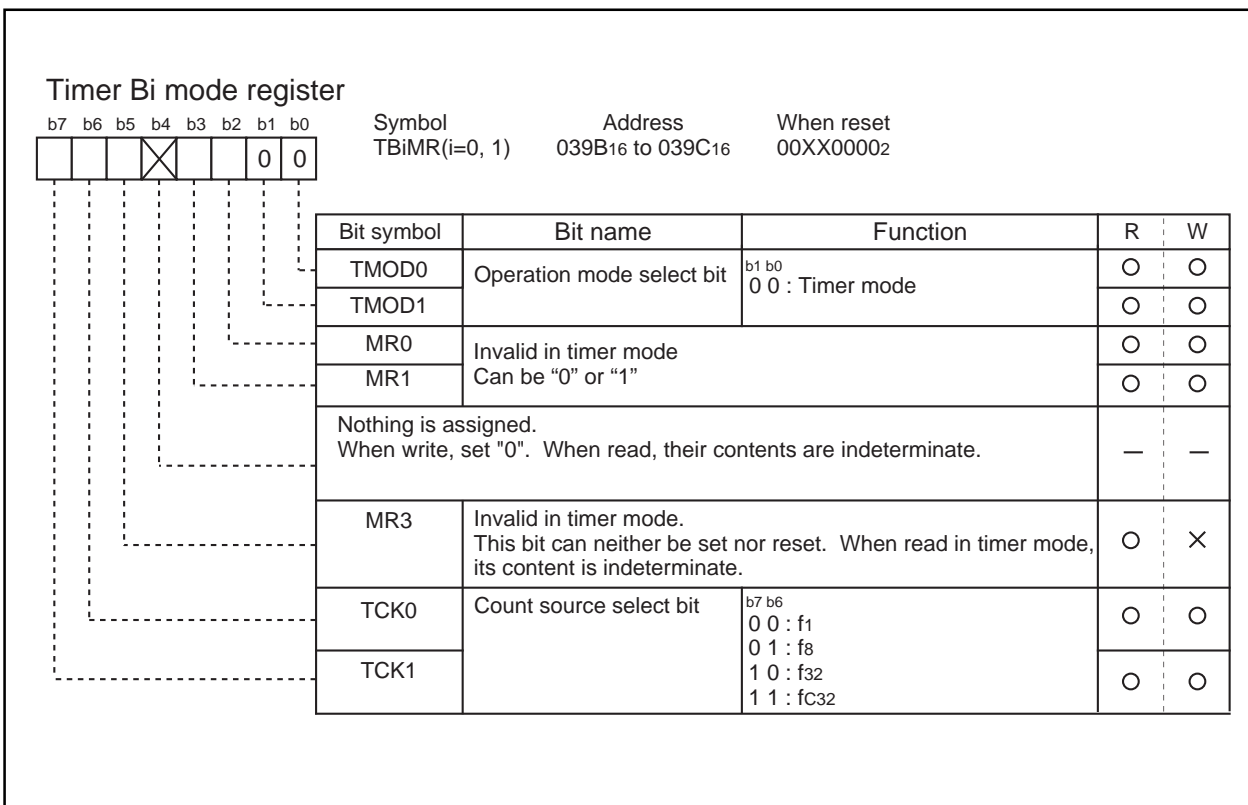


Figure 1.51. Timer Bi mode register in timer mode

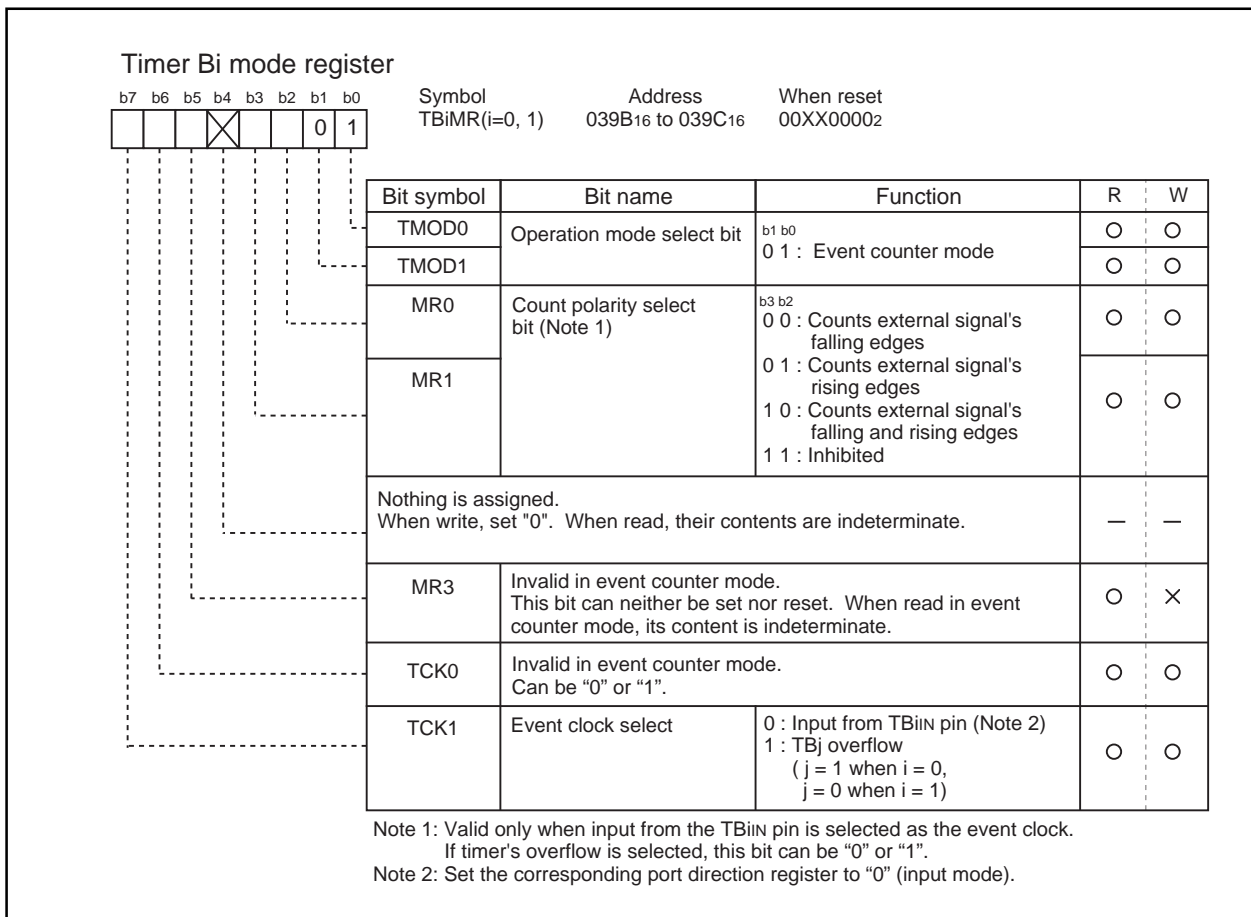
Timer B

**(2) Event counter mode**

In this mode, the timer counts an external signal or an internal timer's overflow. (See Table 1.19.) Figure 1.52 shows the timer Bi mode register in event counter mode.

**Table 1.19. Timer specifications in event counter mode**

Item	Specification
Count source	<ul style="list-style-type: none"> <li>External signals input to TBIiN pin</li> <li>Effective edge of count source can be a rising edge, a falling edge, or falling and rising edges as selected by software</li> </ul>
Count operation	<ul style="list-style-type: none"> <li>Counts down</li> <li>When the timer underflows, it reloads the reload register contents before continuing counting</li> </ul>
Divide ratio	$1/(n+1)$ n : Set value
Count start condition	Count start flag is set (= 1)
Count stop condition	Count start flag is reset (= 0)
Interrupt request generation timing	The timer underflows
TBIiN pin function	Count source input
Read from timer	Count value can be read out by reading timer Bi register
Write to timer	<ul style="list-style-type: none"> <li>When counting stopped When a value is written to timer Bi register, it is written to both reload register and counter</li> <li>When counting in progress When a value is written to timer Bi register, it is written to only reload register (Transferred to counter at next reload time)</li> </ul>



**Figure 1.52. Timer Bi mode register in event counter mode**

## Timer B

**(3) Pulse period/pulse width measurement mode**

In this mode, the timer measures the pulse period or pulse width of an external signal. (See Table 1.20.)

Figure 1.53 shows the timer Bi mode register in pulse period/pulse width measurement mode. Figure

1.54 shows the operation timing when measuring a pulse period. Figure 1.55 shows the operation timing

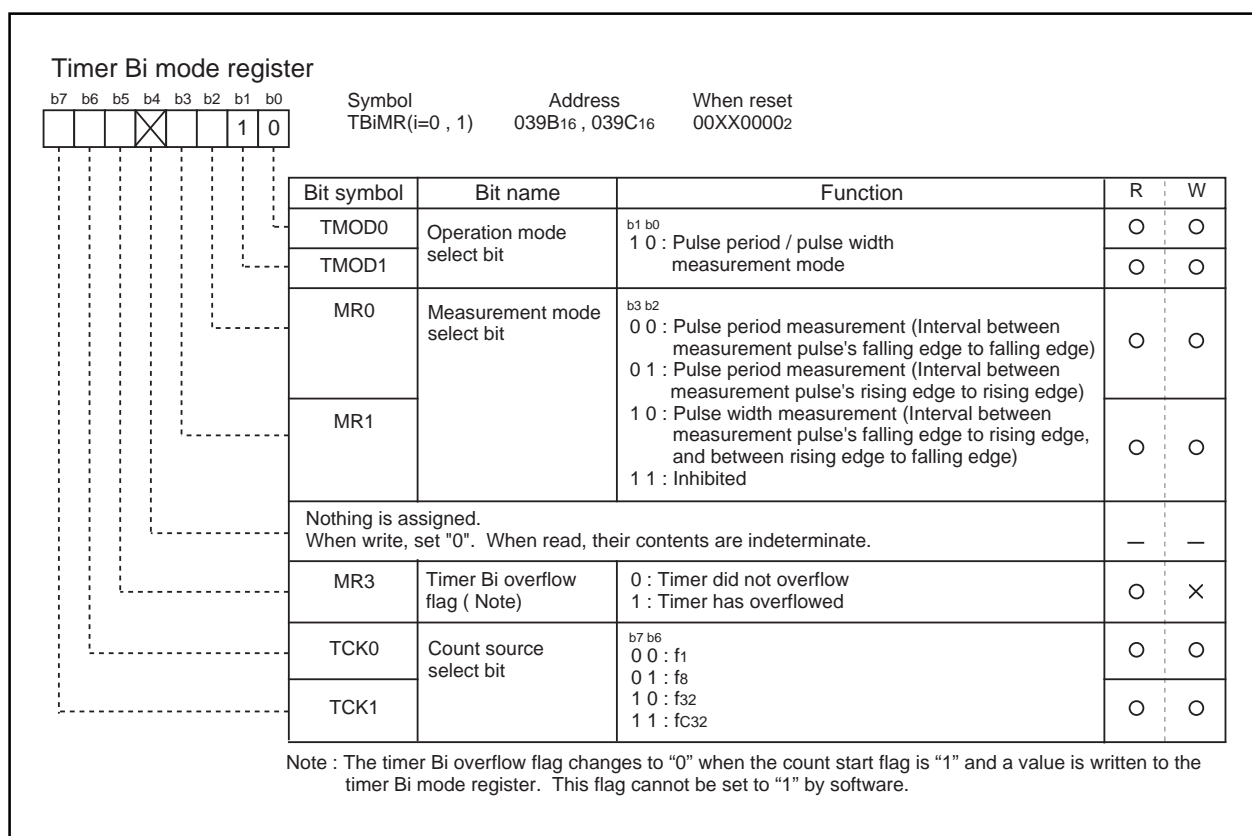
when measuring a pulse width.

**Table 1.20. Timer specifications in pulse period/pulse width measurement mode**

Item	Specification
Count source	f1, f8, f32, fc32
Count operation	<ul style="list-style-type: none"> <li>• Up count</li> <li>• Counter value "000016" is transferred to reload register at measurement pulse's effective edge and the timer continues counting</li> </ul>
Count start condition	Count start flag is set (= 1)
Count stop condition	Count start flag is reset (= 0)
Interrupt request generation timing	<ul style="list-style-type: none"> <li>• When measurement pulse's effective edge is input (Note 1)</li> <li>• When an overflow occurs. (Simultaneously, the timer Bi overflow flag changes to "1". The timer Bi overflow flag changes to "0" when the count start flag is "1" and a value is written to the timer Bi mode register.)</li> </ul>
TBiIN pin function	Measurement pulse input
Read from timer	When timer Bi register is read, it indicates the reload register's content (measurement result) (Note 2)
Write to timer	Cannot be written to

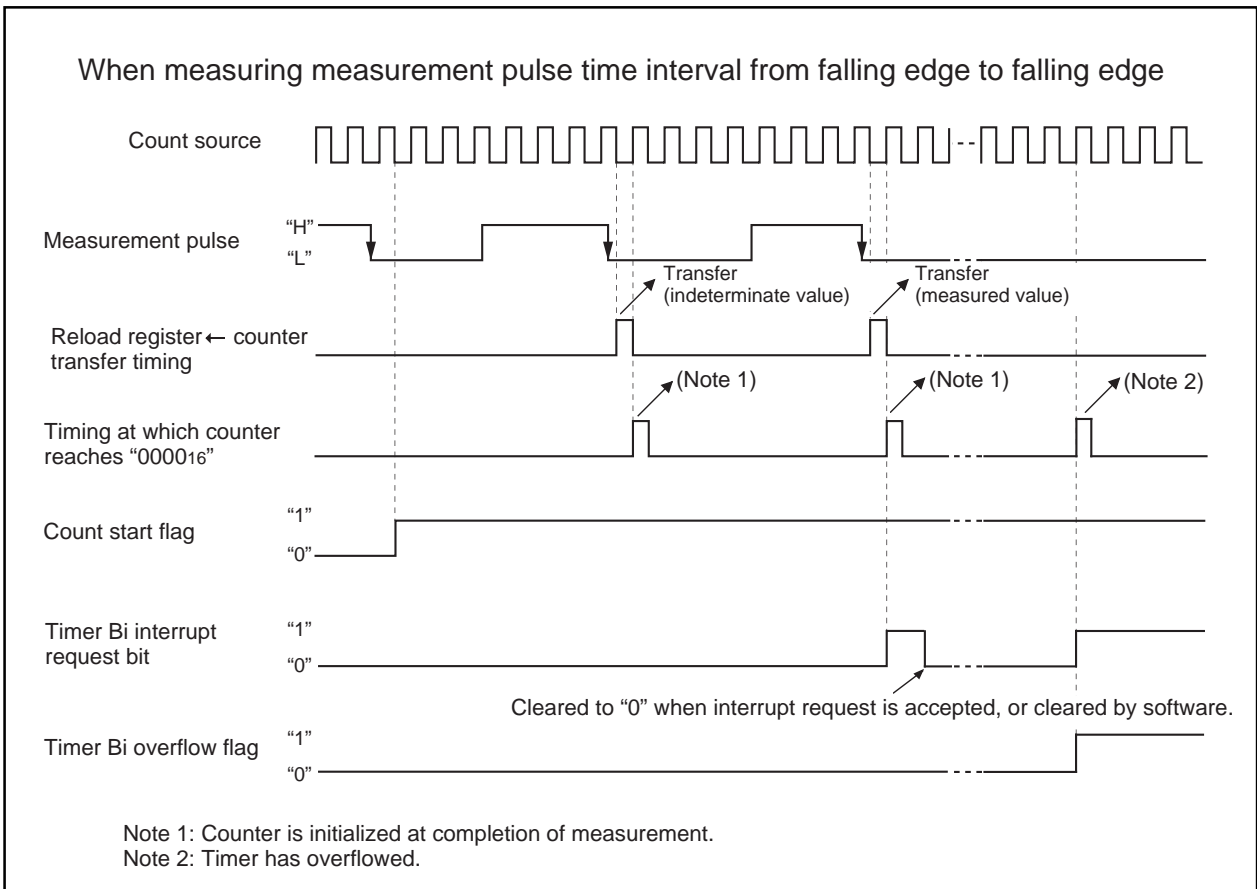
Note 1: An interrupt request is not generated when the first effective edge is input after the timer has started counting.

Note 2: The value read out from the timer Bi register is indeterminate until the second effective edge is input after the timer.

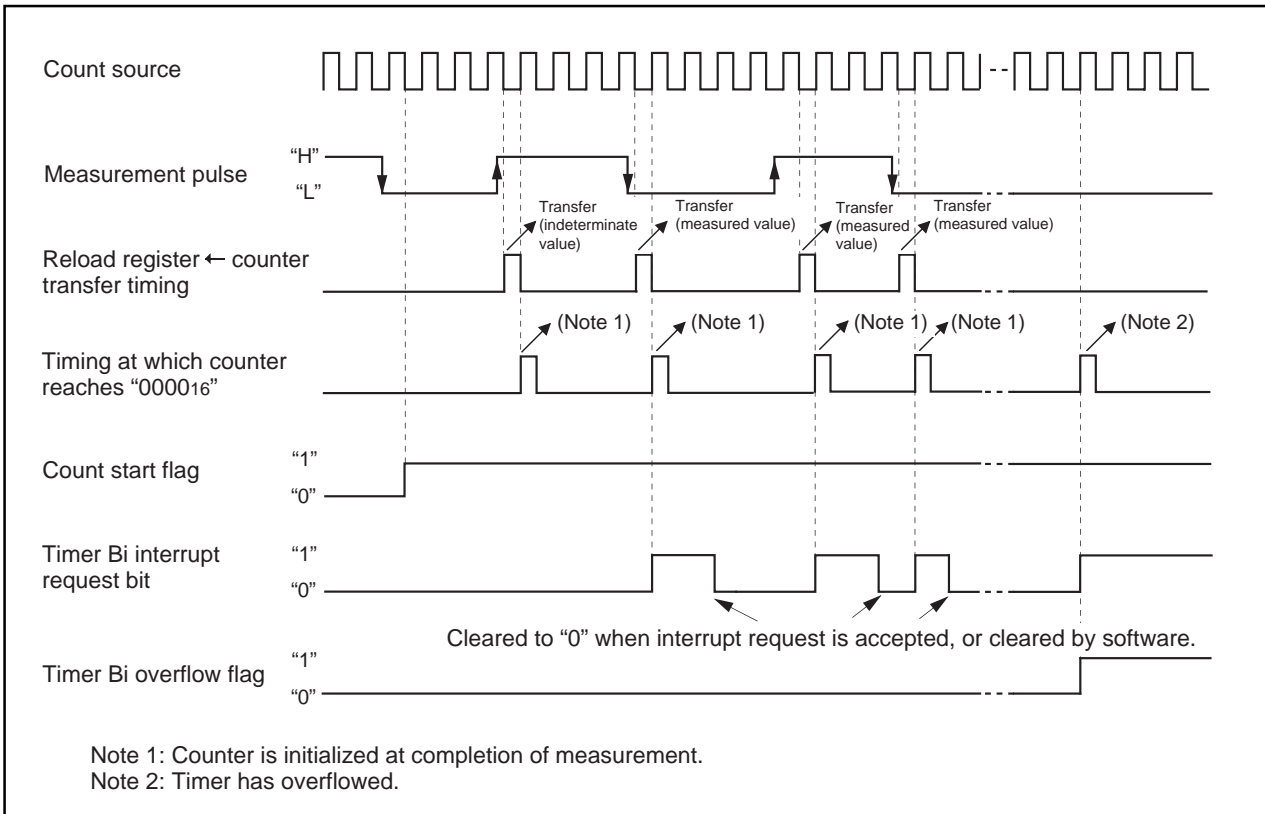


**Figure 1.53. Timer Bi mode register in pulse period/pulse width measurement mode**

**Timer B**



**Figure 1.54. Operation timing when measuring a pulse period**



**Figure 1.55. Operation timing when measuring a pulse width**



Timer X

Timer X

Figure 1.56 shows the block diagram of timer X. Figures 1.57 to 1.59 show the timer X-related registers. Use the timer Xi mode register bits 0 and 1 to choose the desired mode.

Timer X has the five operation modes listed as follows:

- Timer mode : The timer counts an internal count source.
- Event counter mode : The timer counts pulses from an external source or a timer overflow.
- One-shot timer mode : The timer stops counting when the count reaches "0000<sub>16</sub>".
- Pulse period/pulse width measuring mode : The timer measures an external signal's pulse period or pulse width.
- Pulse width modulation (PWM) mode : The timer outputs pulses of a given width.

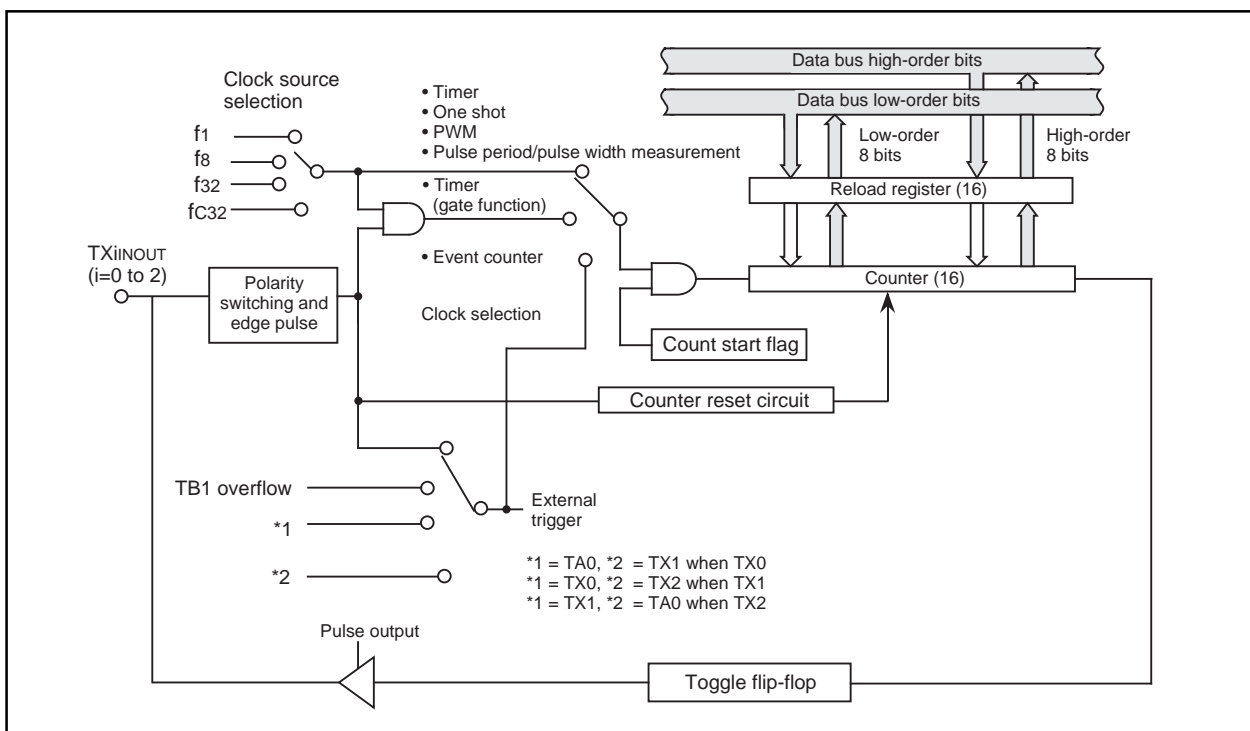


Figure 1.56. Block diagram of timer X

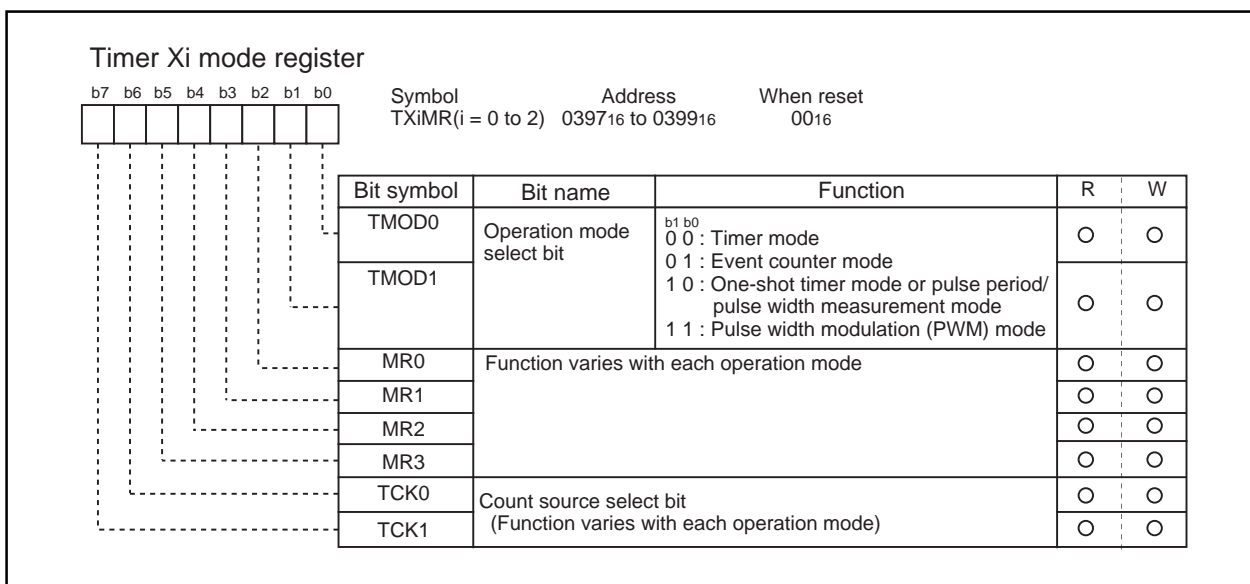


Figure 1.57. Timer X-related registers (1)

Timer X

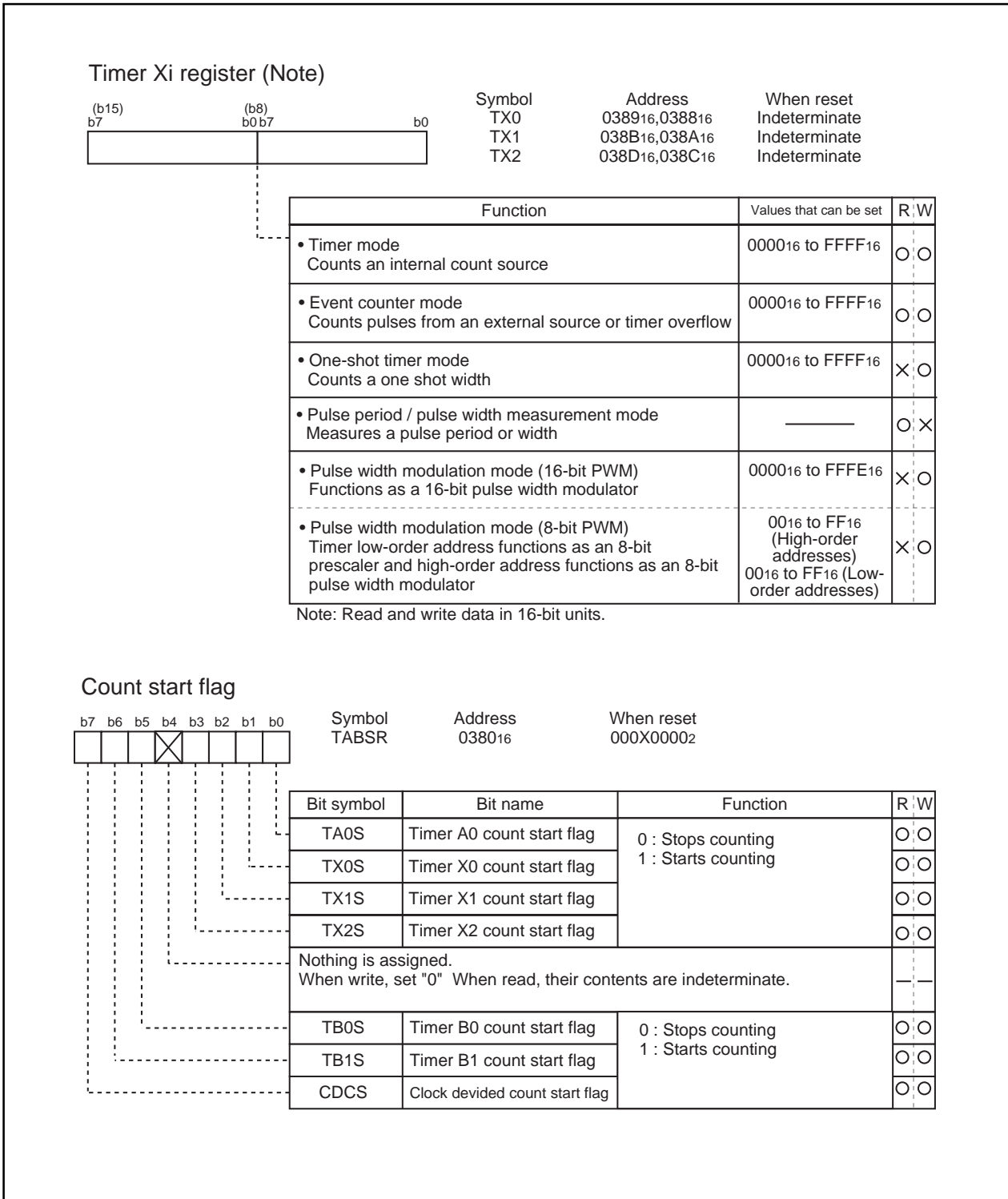


Figure 1.58. Timer X-related registers (2)

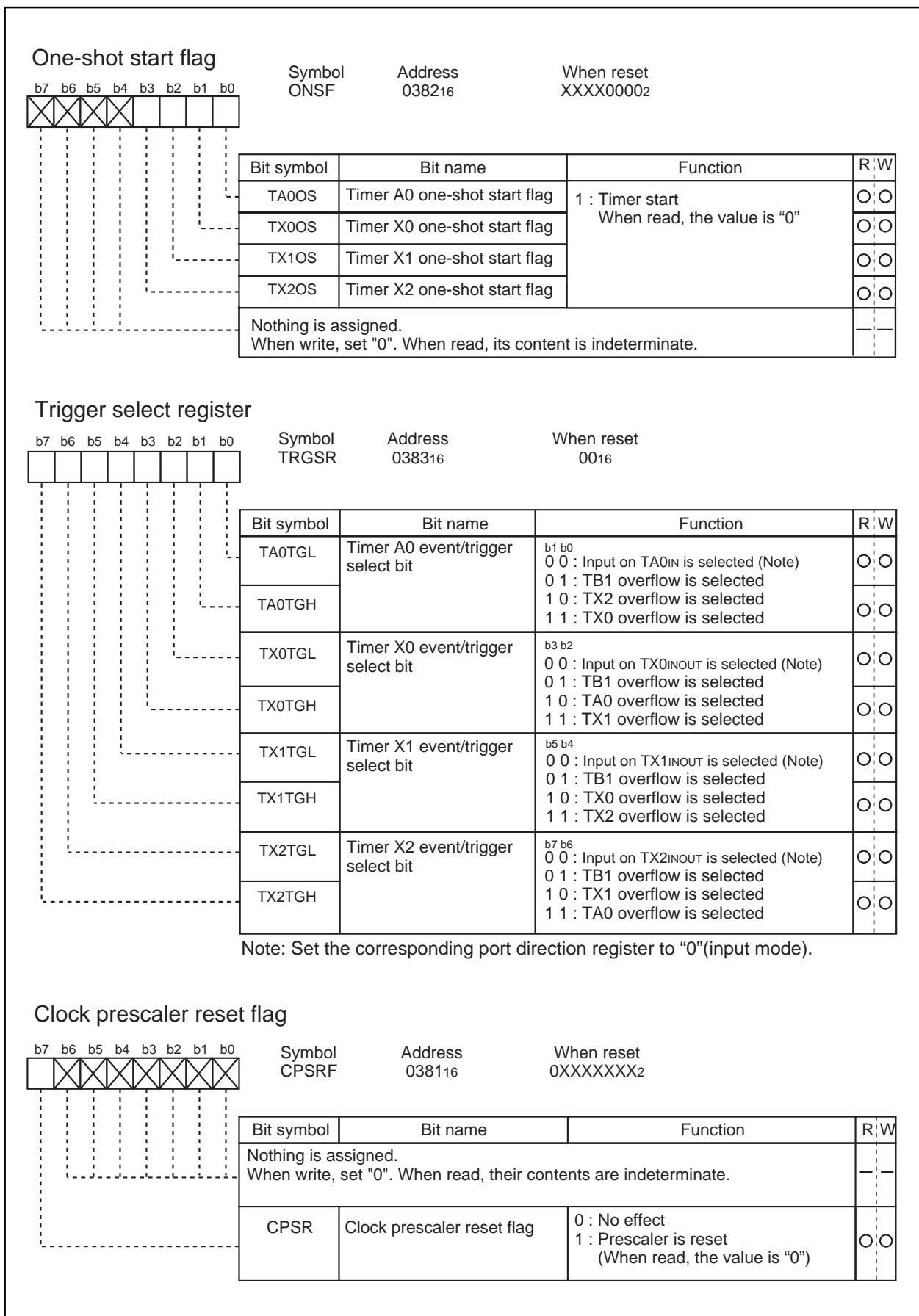


Figure 1.59. Timer X-related registers (3)

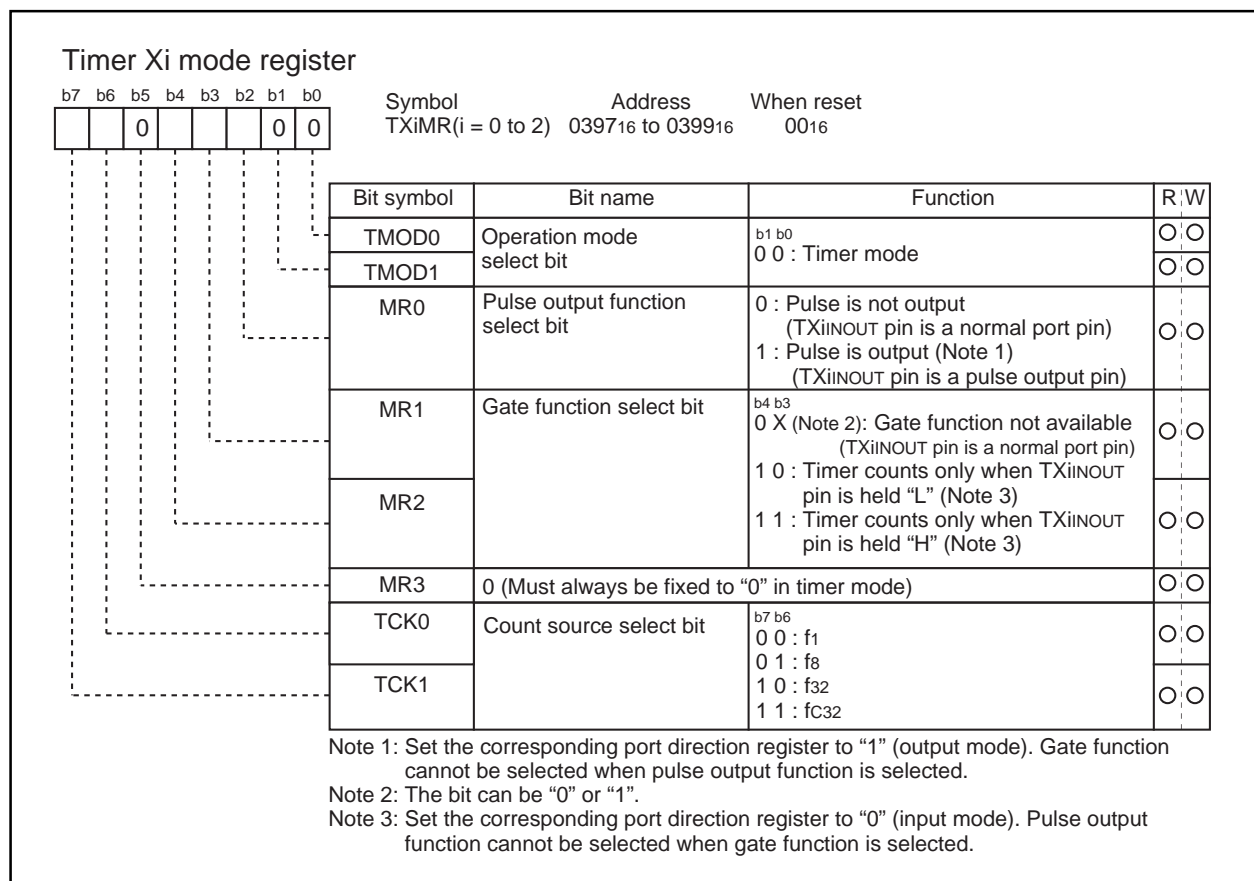
## Timer X

**(1) Timer mode**

In this mode, the timer counts an internally generated count source. (See Table 1.21.) Figure 1.60 shows the timer Xi mode register in timer mode.

**Table 1.21. Specifications of timer mode**

Item	Specification
Count source	f1, f8, f32, fC32
Count operation	<ul style="list-style-type: none"> <li>• Down count</li> <li>• When the timer underflows, it reloads the reload register contents before continuing counting</li> </ul>
Divide ratio	1/(n+1) n : Set value
Count start condition	Count start flag is set (= 1)
Count stop condition	Count start flag is reset (= 0)
Interrupt request generation timing	When the timer underflows
TXiINOUT pin function	Programmable I/O port, gate input or pulse output
Read from timer	Count value can be read out by reading timer Xi register
Write to timer	<ul style="list-style-type: none"> <li>• When counting stopped When a value is written to timer Xi register, it is written to both reload register and counter</li> <li>• When counting in progress When a value is written to timer Xi register, it is written to only reload register (Transferred to counter at next reload time)</li> </ul>
Select function	<ul style="list-style-type: none"> <li>• Gate function Counting can be started and stopped by the TXiINOUT pin's input signal</li> <li>• Pulse output function Each time the timer underflows, the TXiINOUT pin's polarity is reversed</li> </ul>

**Figure 1.60. Timer Xi mode register in timer mode**

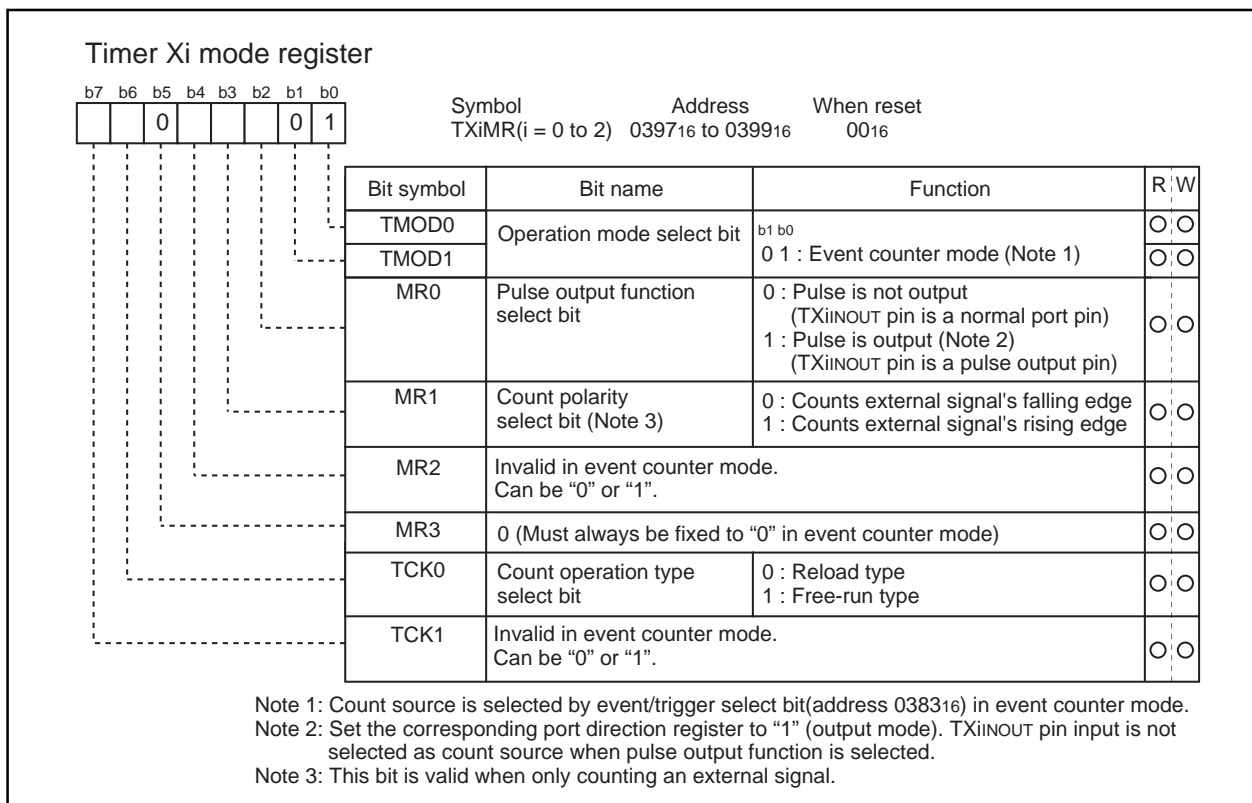
## (2) Event counter mode

In this mode, the timer counts an external signal or an internal timer's overflow. (See Table 1.22.) Figure 1.61 shows the timer Xi mode register in event counter mode.

**Table 1.22. Timer specifications in event counter mode (when not processing two-phase pulse signal)**

Item	Specification
Count source	<ul style="list-style-type: none"> <li>External signals input to TXiINOUT pin (effective edge can be selected by software)</li> <li>TB1 overflow, TA0 overflow, TXi overflow</li> </ul>
Count operation	<ul style="list-style-type: none"> <li>Down count</li> <li>When the timer underflows, it reloads the reload register contents before continuing counting (Note)</li> </ul>
Divide ratio	$1/(n + 1)$ n : Set value
Count start condition	Count start flag is set (= 1)
Count stop condition	Count start flag is reset (= 0)
Interrupt request generation timing	The timer underflows
TXiINOUT pin function	Programmable I/O port, count source input or pulse output
Read from timer	Count value can be read out by reading timer Xi register
Write to timer	<ul style="list-style-type: none"> <li>When counting stopped When a value is written to timer Xi register, it is written to both reload register and counter</li> <li>When counting in progress When a value is written to timer Xi register, it is written to only reload register (Transferred to counter at next reload time)</li> </ul>
Select function	<ul style="list-style-type: none"> <li>Free-run count function Even when the timer underflows, the reload register content is not reloaded to it</li> <li>Pulse output function Each time the timer underflows, the TXiINOUT pin's polarity is reversed</li> </ul>

Note: This does not apply when the free-run function is selected.



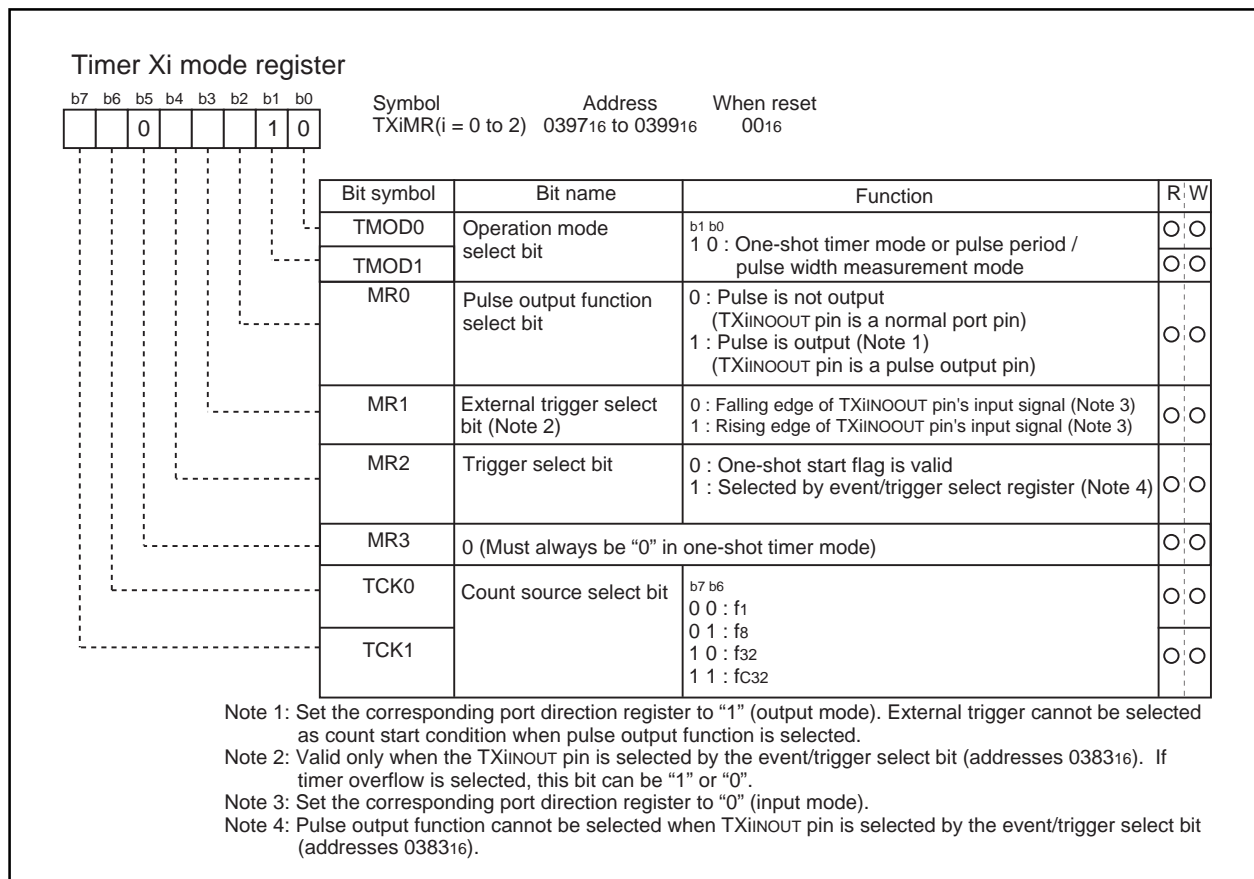
**Figure 1.61. Timer Xi mode register in event counter mode**

### (3) One-shot timer mode

In this mode, the timer operates only once. (See Table 1.23.) When a trigger occurs, the timer starts up and continues operating for a given period. Figure 1.62 shows the timer Xi mode register in one-shot timer mode.

**Table 1.23. Timer specifications in one-shot timer mode**

Item	Specification
Count source	f1, f8, f32, fC32
Count operation	<ul style="list-style-type: none"> <li>The timer counts down</li> <li>When the count reaches 0000<sub>16</sub>, the timer stops counting after reloading a new count</li> <li>If a trigger occurs when counting, the timer reloads a new count and restarts counting</li> </ul>
Divide ratio	1/n    n : Set value
Count start condition	<ul style="list-style-type: none"> <li>An external trigger is input</li> <li>The timer overflows</li> <li>The one-shot start flag is set (= 1)</li> </ul>
Count stop condition	<ul style="list-style-type: none"> <li>A new count is reloaded after the count has reached 0000<sub>16</sub></li> <li>The count start flag is reset (= 0)</li> </ul>
Interrupt request generation timing	The count reaches 0000 <sub>16</sub>
TXiINOUT pin function	Programmable I/O port, trigger input or pulse output
Read from timer	When timer Xi register is read, it indicates an indeterminate value
Write to timer	<ul style="list-style-type: none"> <li>When counting stopped When a value is written to timer Xi register, it is written to both reload register and counter</li> <li>When counting in progress When a value is written to timer Xi register, it is written to only reload register (Transferred to counter at next reload time)</li> </ul>



**Figure 1.62. Timer Xi mode register in one-shot timer mode**

#### (4) Pulse period/pulse width measurement mode

In this mode, the timer measures the pulse period or pulse width of an external signal. (See Table 1.24.)

Figure 1.63 shows the timer Xi mode register in pulse period/pulse width measurement mode. Figure

1.64 shows the operation timing when measuring a pulse period. Figure 1.65 shows the operation timing

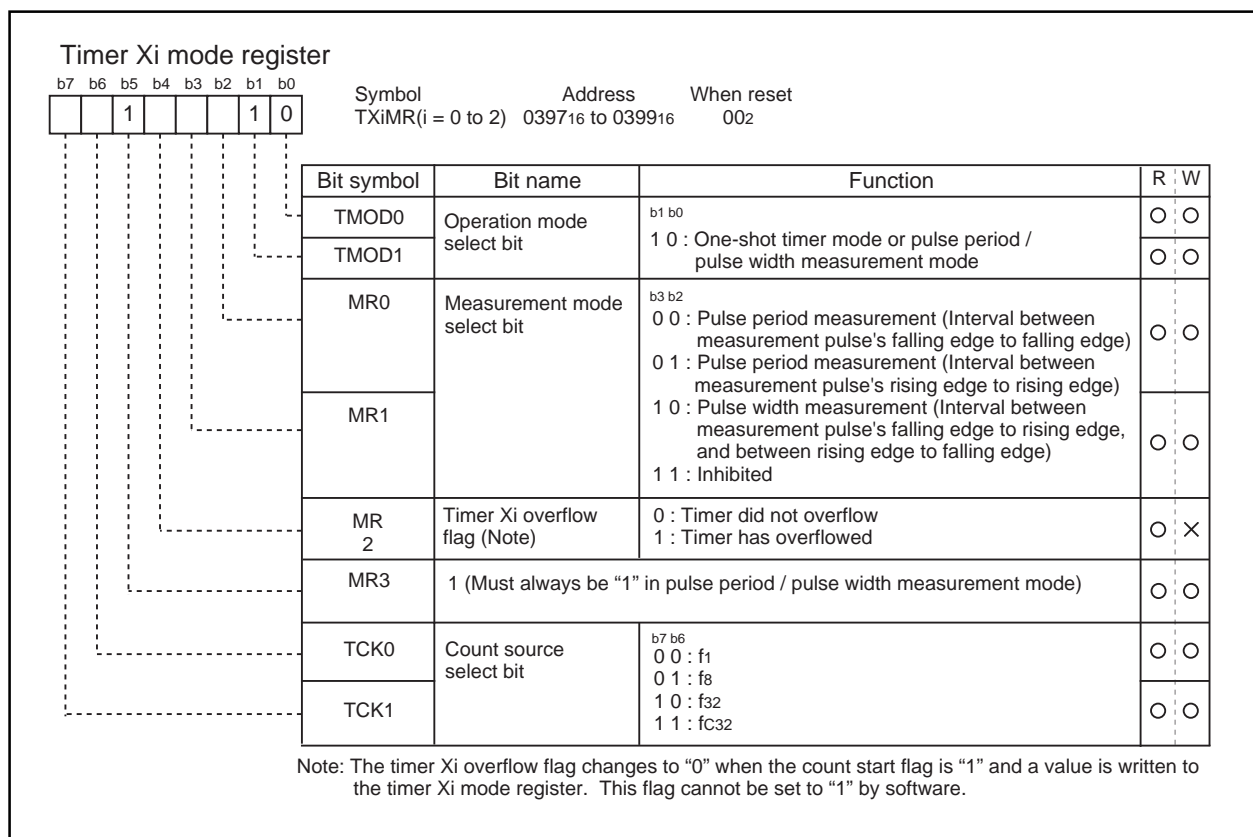
when measuring a pulse width.

**Table 1.24. Timer specifications in pulse period/pulse width measurement mode**

Item	Specification
Count source	f1, f8, f32, fc32
Count operation	<ul style="list-style-type: none"> <li>• Up count</li> <li>• Counter value "000016" is transferred to reload register at measurement pulse's effective edge and the timer continues counting</li> </ul>
Count start condition	Count start flag is set (= 1)
Count stop condition	Count start flag is reset (= 0)
Interrupt request generation timing	<ul style="list-style-type: none"> <li>• When measurement pulse's effective edge is input (Note 1)</li> <li>• When an overflow occurs. (Simultaneously, the timer Xi overflow flag changes to "1". The timer Xi overflow flag changes to "0" when the count start flag is "1" and a value is written to the timer Xi mode register.)</li> </ul>
TXiINOUT pin function	Measurement pulse input
Read from timer	When timer Xi register is read, it indicates the reload register's content (measurement result) (Note 2)
Write to timer	Cannot be written to

Note 1: An interrupt request is not generated when the first effective edge is input after the timer has started counting.

Note 2: The value read out from the timer Xi register is indeterminate until the second effective edge is input after the timer.



**Figure 1.63. Timer Xi mode register in pulse period/pulse width measurement mode**

Timer X

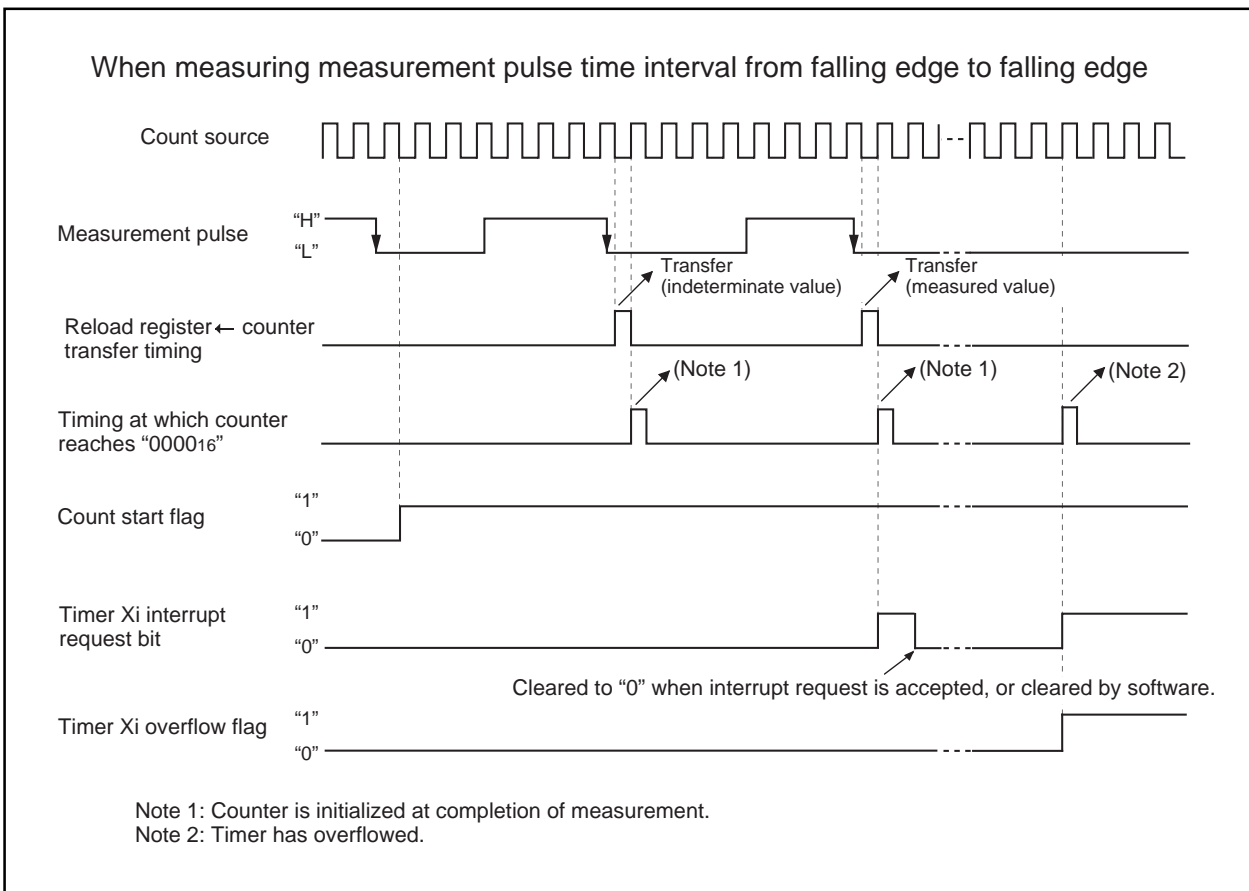


Figure 1.64. Operation timing when measuring a pulse period

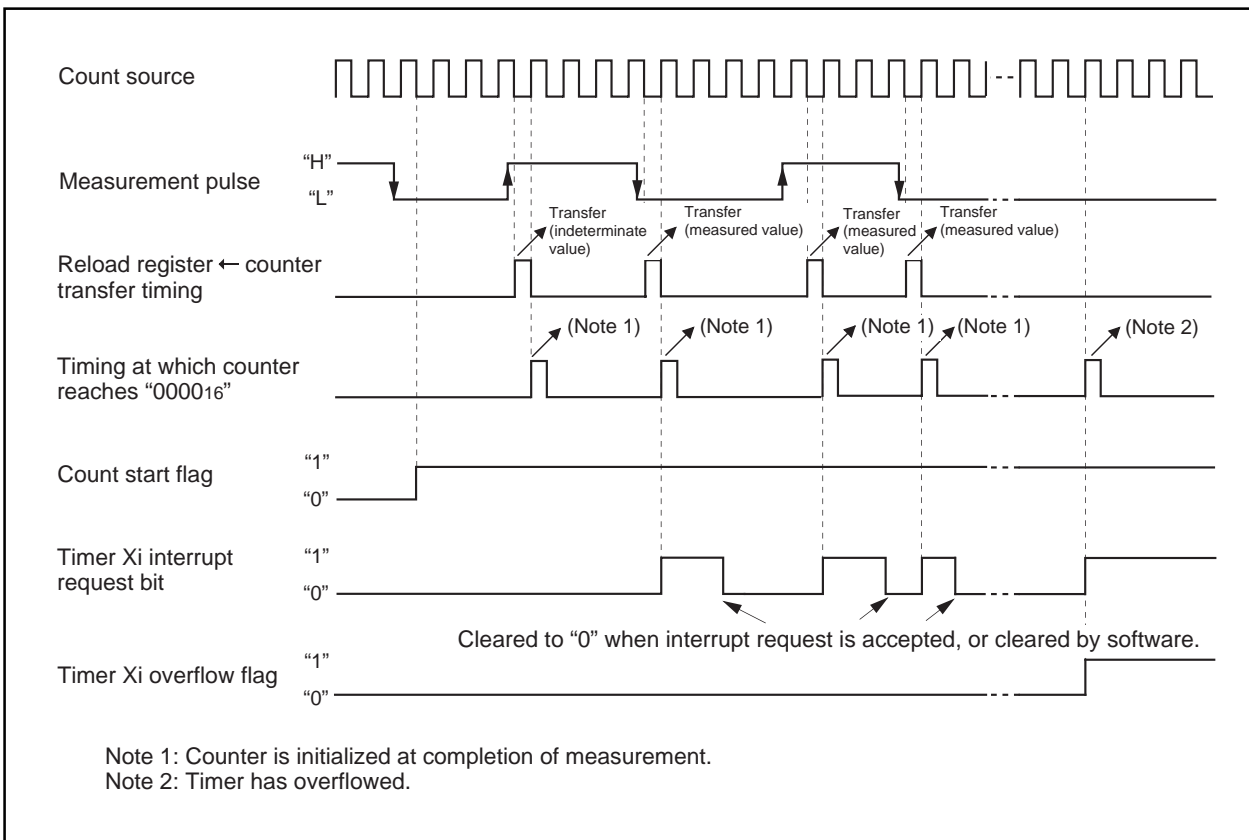


Figure 1.65. Operation timing when measuring a pulse width



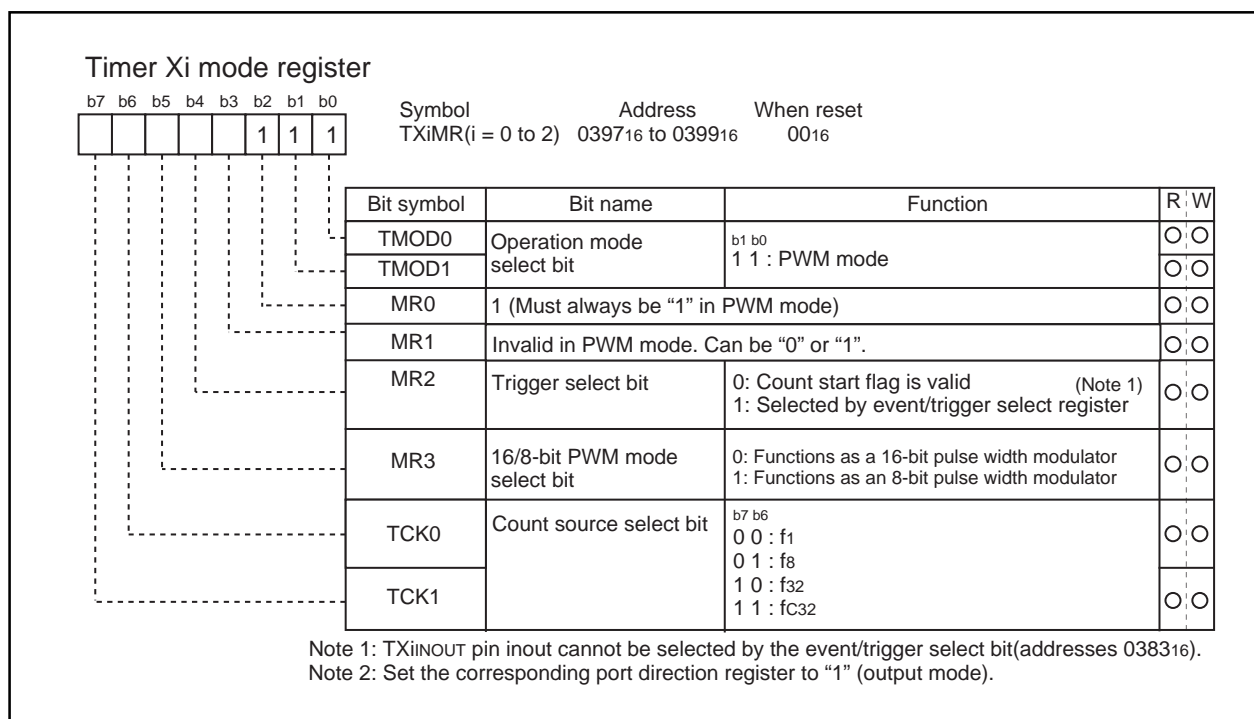
### (5) Pulse width modulation (PWM) mode

In this mode, the timer outputs pulses of a given width in succession. (See Table 1.25.) In this mode, the counter functions as either a 16-bit pulse width modulator or an 8-bit pulse width modulator. Figure 1.66 shows the timer Xi mode register in pulse width modulation mode. Figure 1.67 shows the example of how a 16-bit pulse width modulator operates. Figure 1.68 shows the example of how an 8-bit pulse width modulator operates.

**Table 1.25. Timer specifications in pulse width modulation mode**

Item		Specification
Count source		f1, f8, f32, fc32
Count operation		<ul style="list-style-type: none"> <li>Down counts (operating as an 8-bit or a 16-bit pulse width modulator)</li> <li>The timer reloads a new count at a rising edge of PWM pulse and continues counting</li> <li>The timer is not affected by a trigger that occurs when counting</li> </ul>
16-bit PWM		<ul style="list-style-type: none"> <li>"H" level width <math>n / f_i</math> <span style="float:right">n : Set value</span></li> <li>Cycle time <math>(2^{16}-1) / f_i</math> fixed</li> </ul>
8-bit PWM		<ul style="list-style-type: none"> <li>"H" level width <math>n \times (m+1) / f_i</math> n: values set to timer Xi register's high-order address</li> <li>Cycle time <math>(2^8-1) \times (m+1) / f_i</math> m : values set to timer Xi register's low-order address</li> </ul>
Count start condition		<ul style="list-style-type: none"> <li>The timer overflows</li> <li>The count start flag is set (= 1)</li> </ul>
Count stop condition		<ul style="list-style-type: none"> <li>The count start flag is reset (= 0)</li> </ul>
Interrupt request generation timing	8 bits PWM	<ul style="list-style-type: none"> <li>Set value of "H" level width is except FF<sub>16</sub>, 00<sub>16</sub> : PWM pulse goes "L"</li> <li>Set value of "H" level width is FF<sub>16</sub>, 00<sub>16</sub> : Timing that count value goes to 01<sub>16</sub></li> </ul>
	16 bits PWM	<ul style="list-style-type: none"> <li>Set value of "H" level width is except FFFF<sub>16</sub>, 0000<sub>16</sub> : PWM pulse goes "L"</li> <li>Set value of "H" level width is FFFF<sub>16</sub>, 0000<sub>16</sub> : Timing that count value goes to 0001<sub>16</sub></li> </ul>
TXiINOUT pin function		Pulse output
Read from timer		When timer Xi register is read, it indicates an indeterminate value
Write to timer		<ul style="list-style-type: none"> <li>When counting stopped When a value is written to timer Xi register, it is written to both reload register and counter</li> <li>When counting in progress When a value is written to timer Xi register, it is written to only reload register (Transferred to counter at next reload time)</li> </ul>

Note: When set value of "H" level width is 00<sub>16</sub> or 0000<sub>16</sub>, pulse outputs "L" level and inversion value, FF<sub>16</sub> or FFFF<sub>16</sub> is set to timer.



**Figure 1.66. Timer Xi mode register in pulse width modulation mode**

Timer X

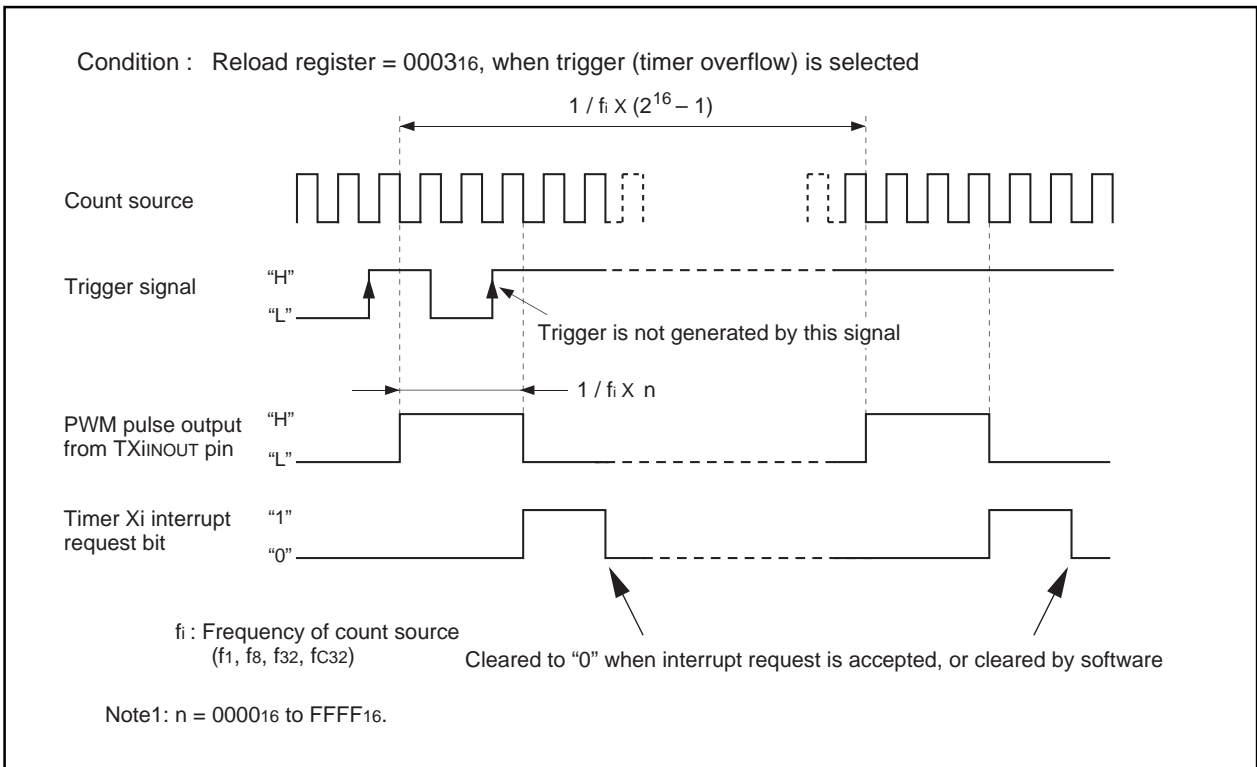


Figure 1.67. Example of how a 16-bit pulse width modulator operates

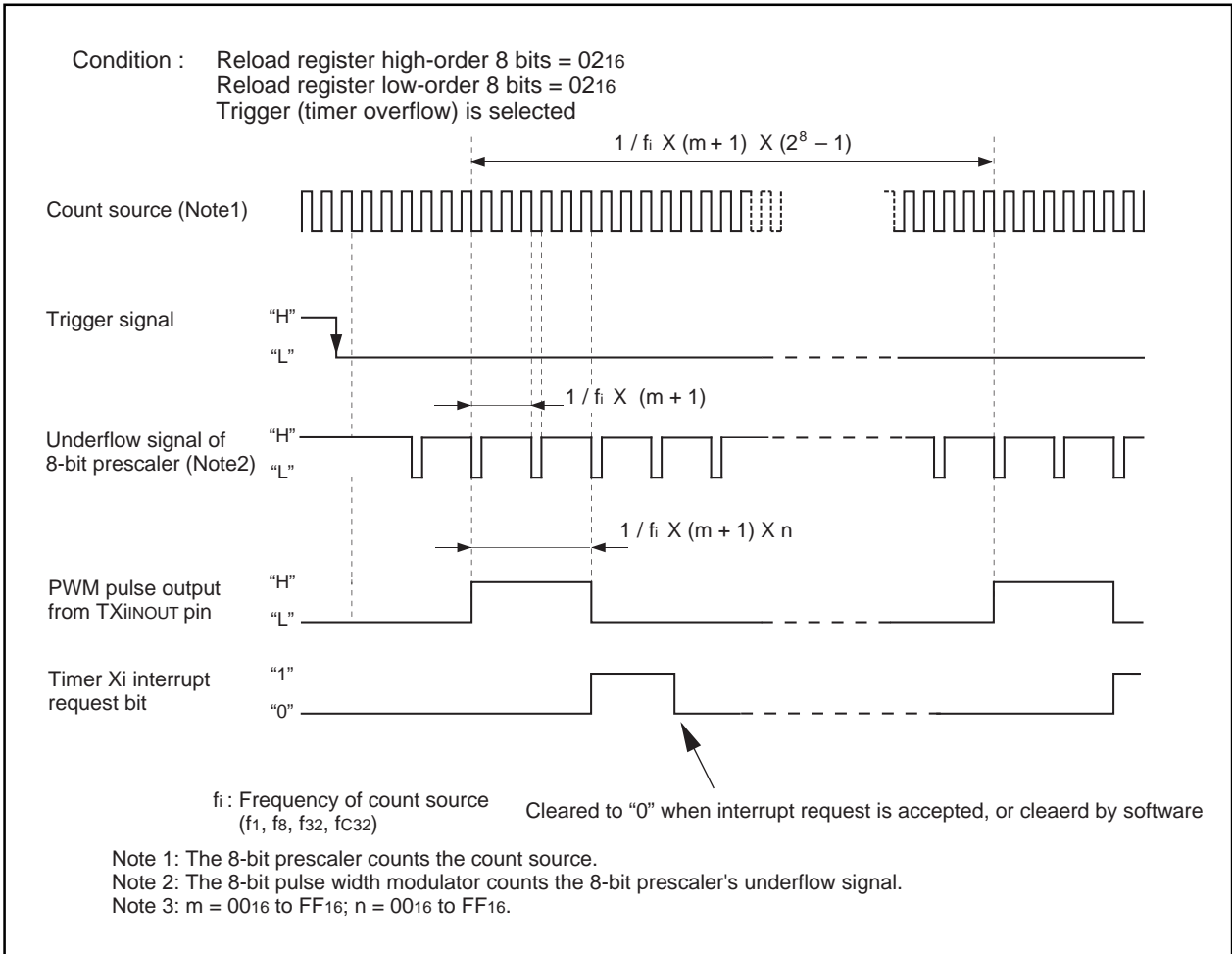


Figure 1.68. Example of how an 8-bit pulse width modulator operates

## Serial I/O

Serial I/O is configured as two channels: UART0 and UART1.

UART0 and UART1 each have an exclusive timer to generate a transfer clock, so they operate independently of each other.

Figure 1.69 shows the block diagram of UART0 and UART1. Figure 1.70 shows the block diagram of the transmit/receive unit.

UART0 has two operation modes: a clock synchronous serial I/O mode and a clock asynchronous serial I/O mode (UART mode). The contents of the serial I/O mode select bits (bits 0 to 2 at addresses 03A016 and 03A816) determine whether UART0 is used as a clock synchronous serial I/O or as a UART.

UART1 is used as a UART only.

Figures 1.71 through 1.73 show the registers related to UARTi.

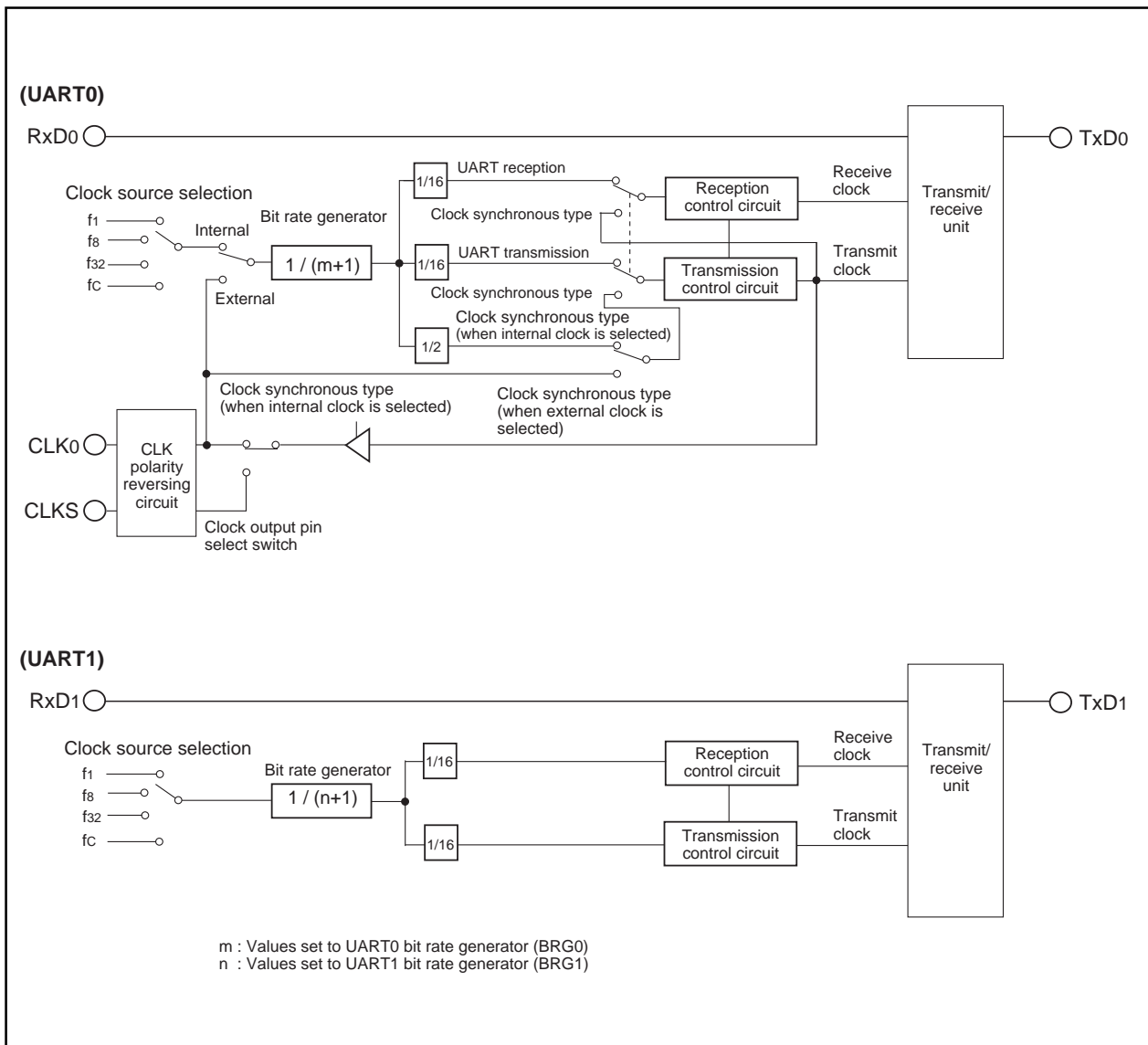


Figure 1.69. Block diagram of UARTi (i = 0, 1)

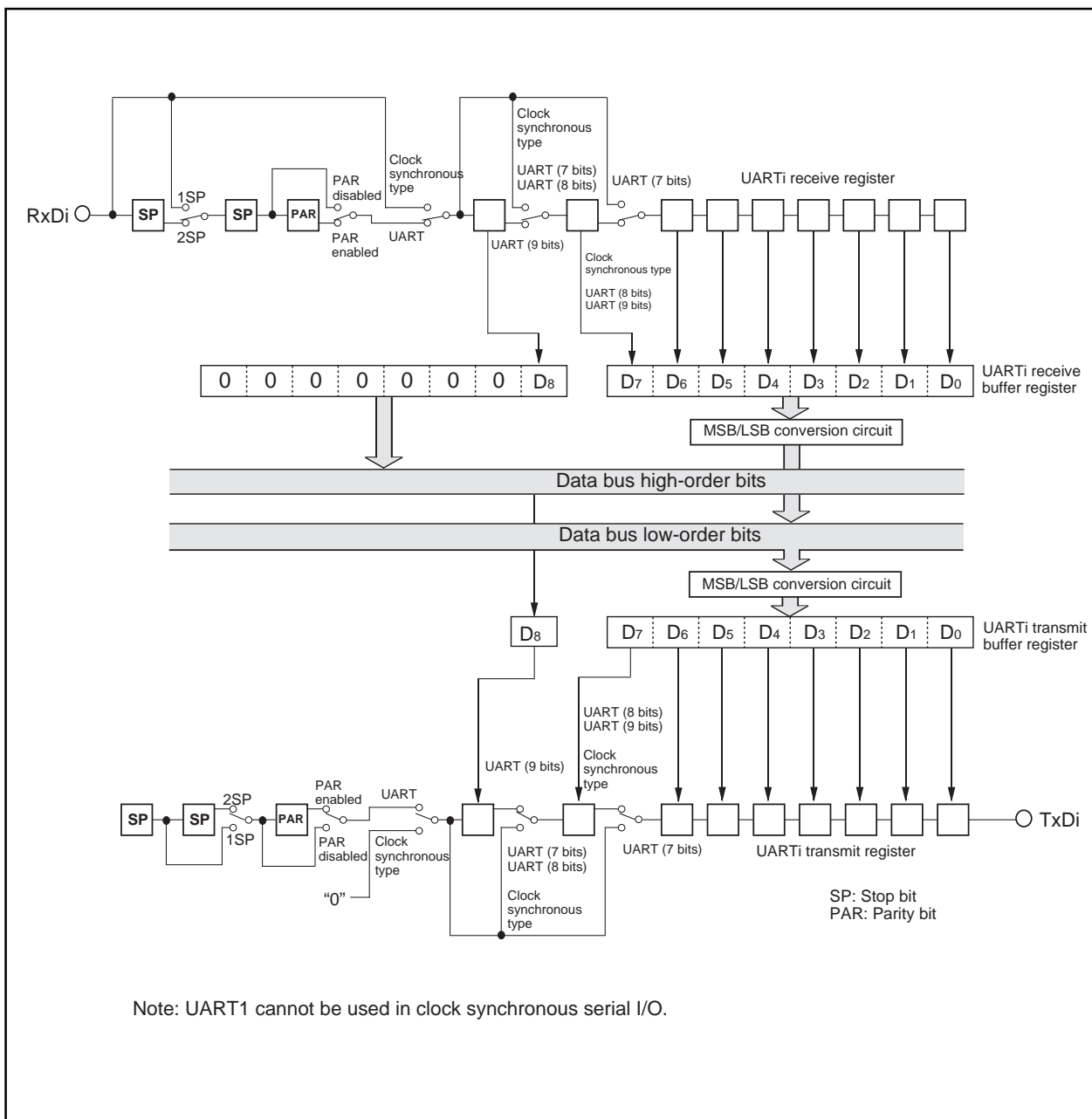


Figure 1.70. Block diagram of transmit/receive unit

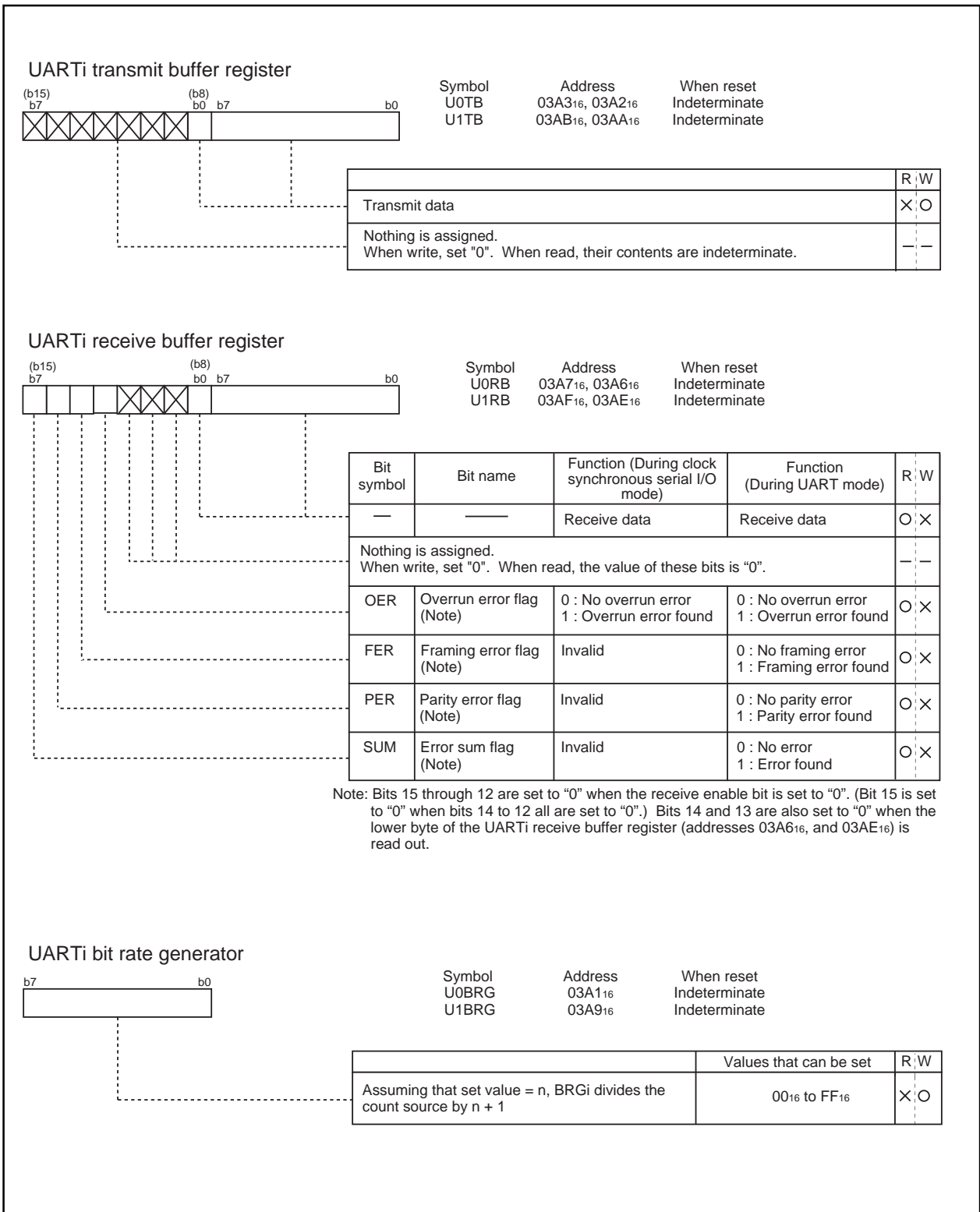


Figure 1.71. Serial I/O-related registers (1)

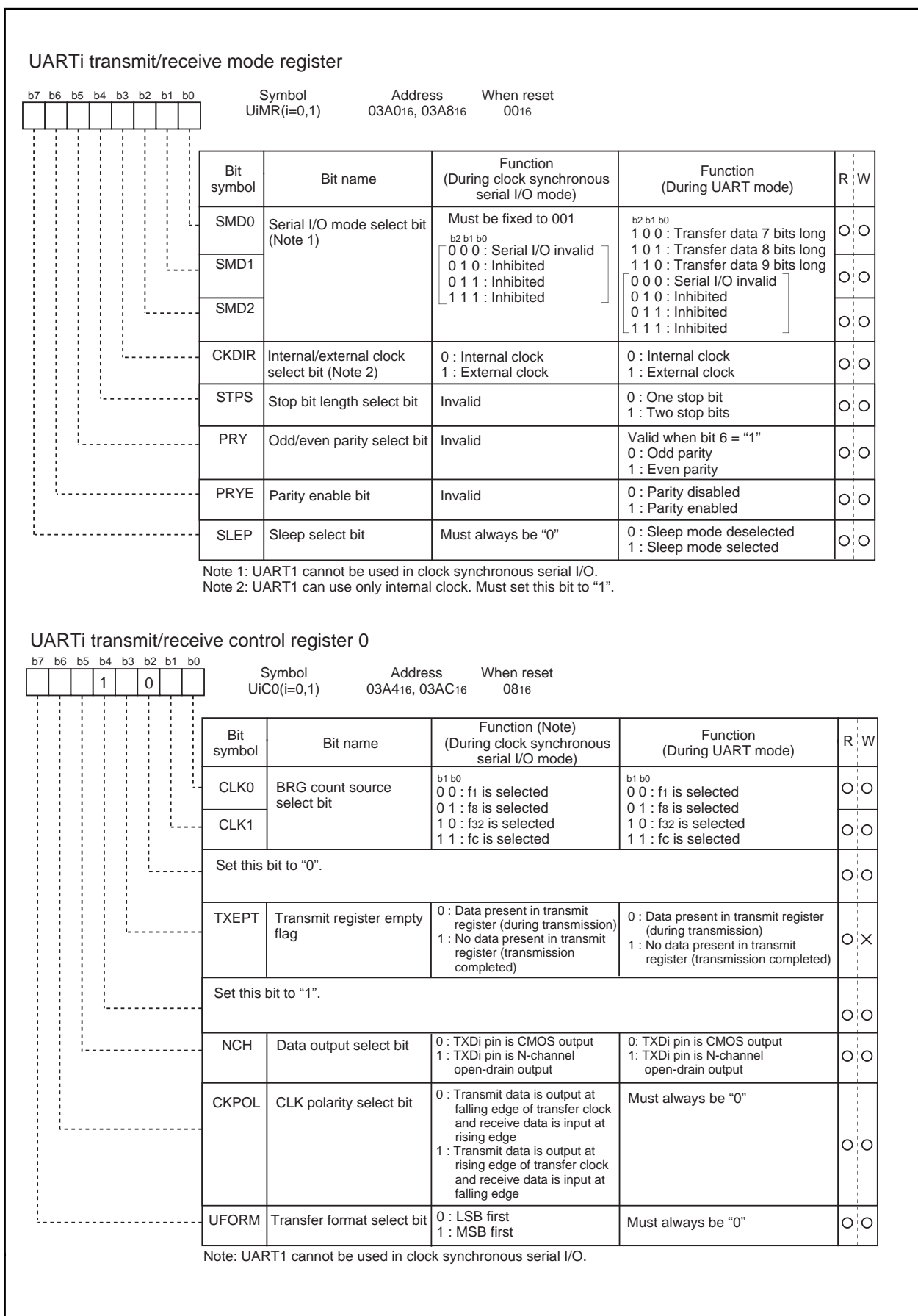


Figure 1.72. Serial I/O-related registers (2)

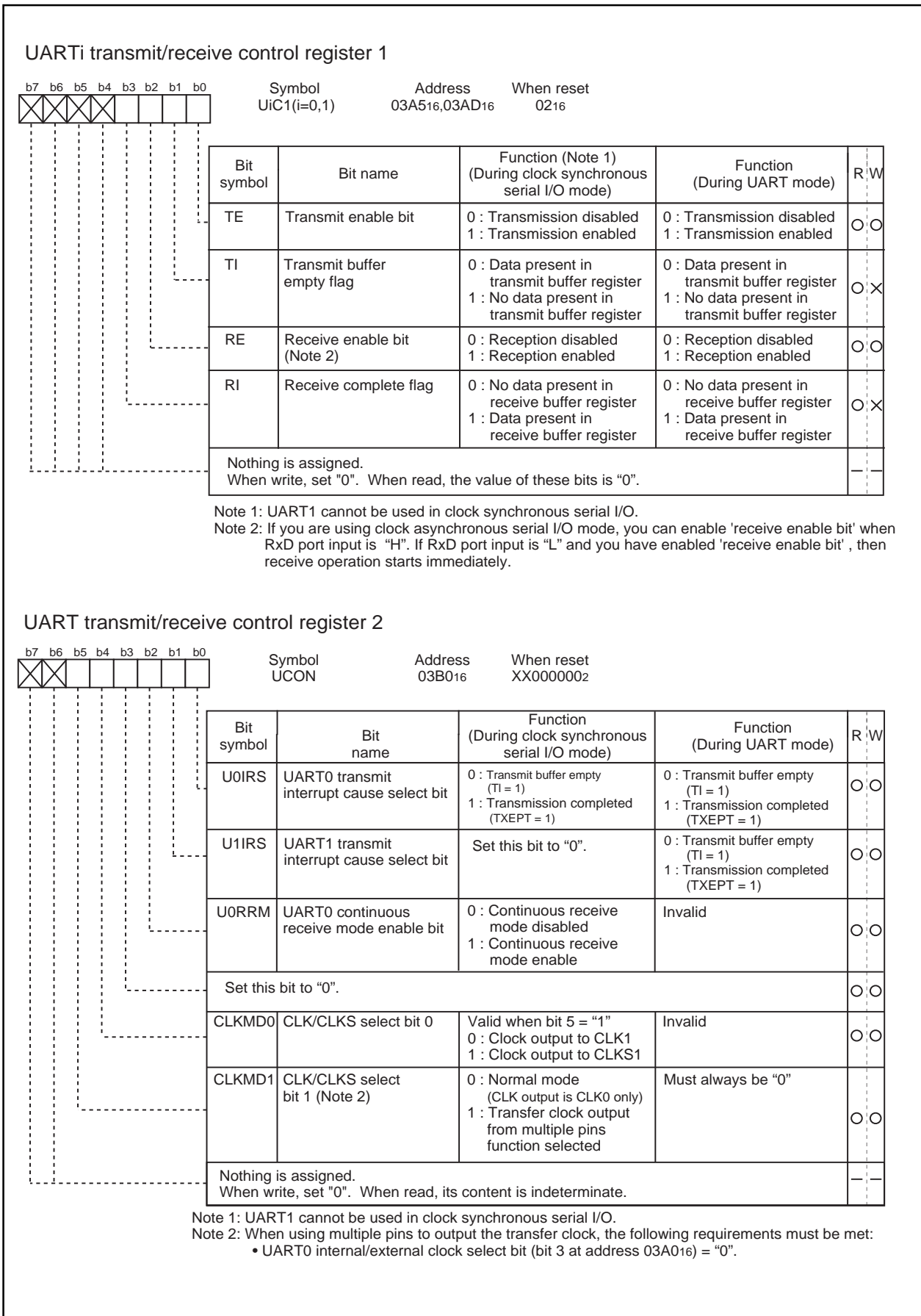


Figure 1.73. Serial I/O-related registers (3)

## Clock synchronous serial I/O mode

**(1) Clock synchronous serial I/O mode**

The clock synchronous serial I/O mode uses a transfer clock to transmit and receive data. (See Table 1.26.) Figure 1.65 shows the UART0 transmit/receive mode register.

**Table 1.26. Specifications of clock synchronous serial I/O mode**

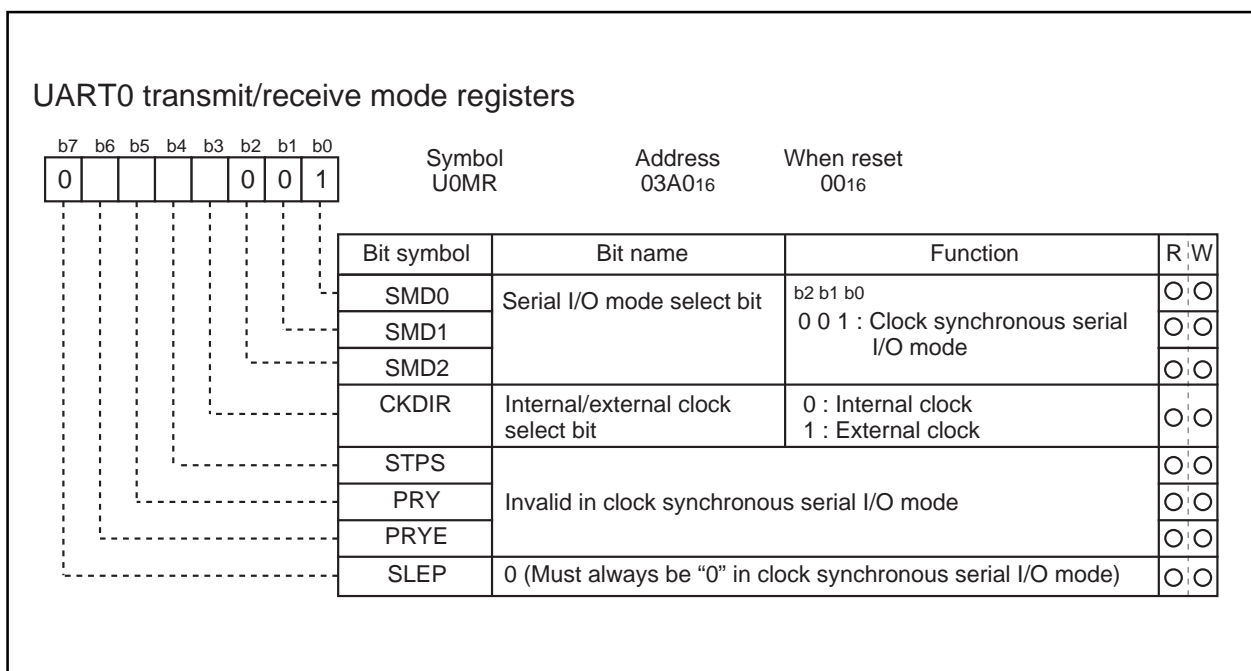
Item	Specification
Transfer data format	<ul style="list-style-type: none"> <li>• Transfer data length: 8 bits</li> </ul>
Transfer clock	<ul style="list-style-type: none"> <li>• When internal clock is selected (bit 3 at address 03A0<sub>16</sub> = "0") : <math>f_i / 2(n+1)</math> (Note 1)  <math>f_i = f_1, f_8, f_{32}, f_c</math></li> <li>• When external clock is selected (bit 3 at address 03A0<sub>16</sub> = "1") : Input from CLK0 pin</li> </ul>
Transmission start condition	<ul style="list-style-type: none"> <li>• To start transmission, the following requirements must be met: <ul style="list-style-type: none"> <li>– Transmit enable bit (bit 0 at address 03A5<sub>16</sub>) = "1"</li> <li>– Transmit buffer empty flag (bit 1 at addresses 03A5<sub>16</sub>) = "0"</li> </ul> </li> <li>• Furthermore, if external clock is selected, the following requirements must also be met: <ul style="list-style-type: none"> <li>– CLK0 polarity select bit (bit 6 at address 03A4<sub>16</sub>) = "0": CLK0 input level = "H"</li> <li>– CLK0 polarity select bit (bit 6 at address 03A4<sub>16</sub>) = "1": CLK0 input level = "L"</li> </ul> </li> </ul>
Reception start condition	<ul style="list-style-type: none"> <li>• To start reception, the following requirements must be met: <ul style="list-style-type: none"> <li>– Receive enable bit (bit 2 at address 03A5<sub>16</sub>) = "1"</li> <li>– Transmit enable bit (bit 0 at address 03A5<sub>16</sub>) = "1"</li> <li>– Transmit buffer empty flag (bit 1 at address 03A5<sub>16</sub>) = "0"</li> </ul> </li> <li>• Furthermore, if external clock is selected, the following requirements must also be met: <ul style="list-style-type: none"> <li>– CLK0 polarity select bit (bit 6 at address 03A4<sub>16</sub>) = "0": CLK0 input level = "H"</li> <li>– CLK0 polarity select bit (bit 6 at address 03A4<sub>16</sub>) = "1": CLK0 input level = "L"</li> </ul> </li> </ul>
Interrupt request generation timing	<ul style="list-style-type: none"> <li>• When transmitting <ul style="list-style-type: none"> <li>– Transmit interrupt cause select bit (bit 0 at address 03B0<sub>16</sub>) = "0": Interrupts requested when data transfer from UART0 transfer buffer register to UART0 transmit register is completed</li> <li>– Transmit interrupt cause select bit (bit 0 at address 03B0<sub>16</sub>) = "1": Interrupts requested when data transmission from UART0 transfer register is completed</li> </ul> </li> <li>• When receiving <ul style="list-style-type: none"> <li>– Interrupts requested when data transfer from UART0 receive register to UART0 receive buffer register is completed</li> </ul> </li> </ul>
Error detection	<ul style="list-style-type: none"> <li>• Overrun error (Note 2)  This error occurs when the next data is ready before contents of UART0 receive buffer register are read out</li> </ul>
Select function	<ul style="list-style-type: none"> <li>• CLK polarity selection  Whether transmit data is output/input at the rising edge or falling edge of the transfer clock can be selected</li> <li>• LSB first/MSB first selection  Whether transmission/reception begins with bit 0 or bit 7 can be selected</li> <li>• Continuous receive mode selection  Reception is enabled simultaneously by a read from the receive buffer register</li> <li>• Transfer clock output from multiple pins selection  UART0 transfer clock can be chosen by software to be output from one of the two pins set</li> </ul>

Note 1: "n" denotes the value 00<sub>16</sub> to FF<sub>16</sub> that is set to the UART bit rate generator.

Note 2: If an overrun error occurs, the UART0 receive buffer will have the next data written in. Note also that the UART0 receive interrupt request bit is not set to "1".



**Clock synchronous serial I/O mode**



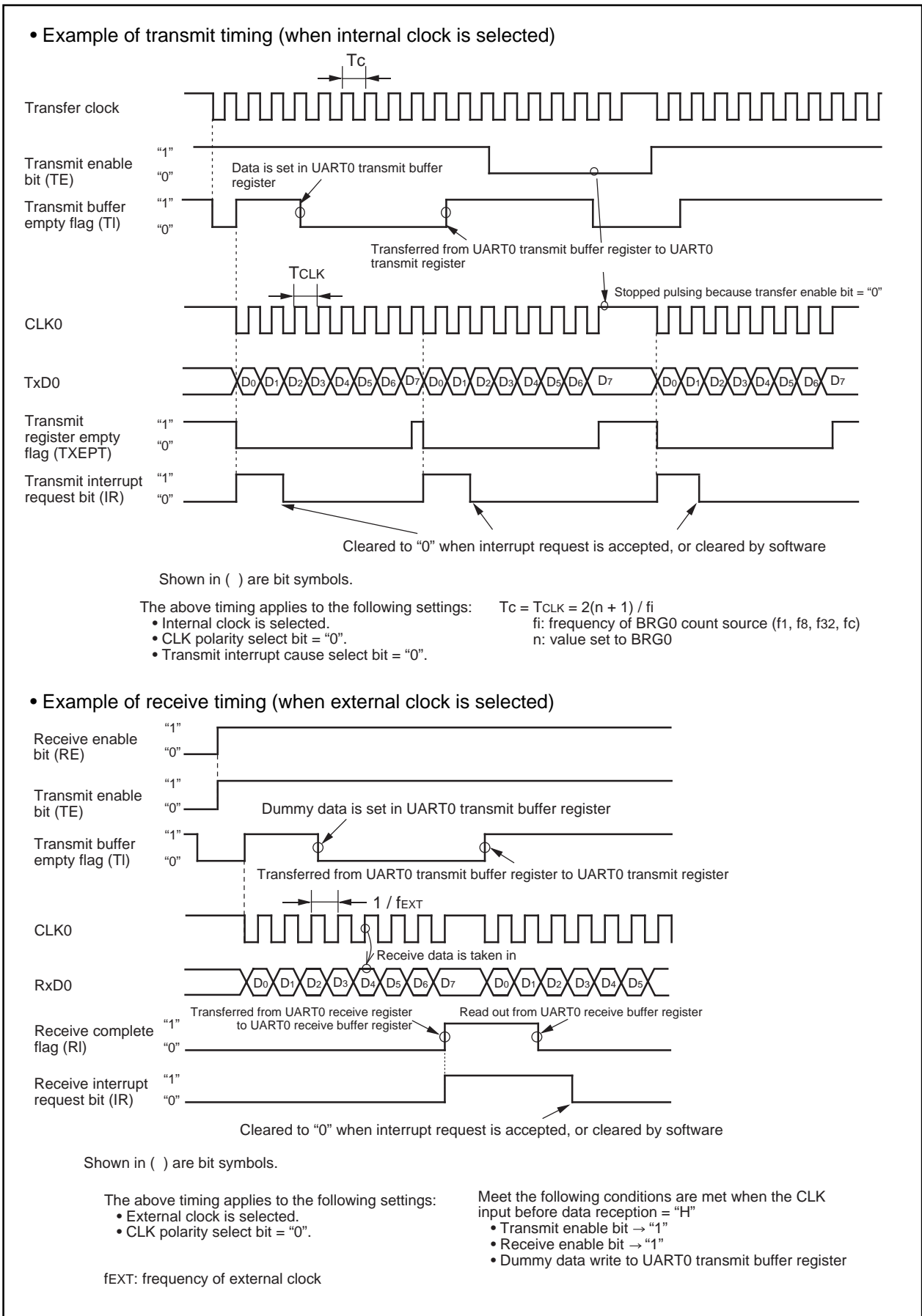
**Figure 1.74. UART0 transmit/receive mode register in clock synchronous serial I/O mode**

Table 1.27 lists the functions of the input/output pins during clock synchronous serial I/O mode. Note that for a period from when the UART0 operation mode is selected to when transfer starts, the TxD0 pin outputs a "H". (If the N-channel open-drain is selected, this pin is in floating state.)

**Table 1.27. Input/output pin functions in clock synchronous serial I/O mode**

Pin name	Function	Method of selection
TxD0 (P50)	Serial data output	Port P50 direction register (bit 0 at address 03EB <sub>16</sub> ) = "1" (Outputs dummy data when performing reception only)
RxD0 (P51)	Serial data input	Port P51 direction register (bit 1 at address 03EB <sub>16</sub> ) = "0" (Can be used as an input port when performing transmission only)
CLK0 (P52)	Transfer clock output	Internal/external clock select bit (bit 3 at address 03A0 <sub>16</sub> ) = "0"
	Transfer clock input	Internal/external clock select bit (bit 3 at address 03A0 <sub>16</sub> ) = "1" Port P52 direction register (bit 2 at address 03EB <sub>16</sub> ) = "0"

**Clock synchronous serial I/O mode**

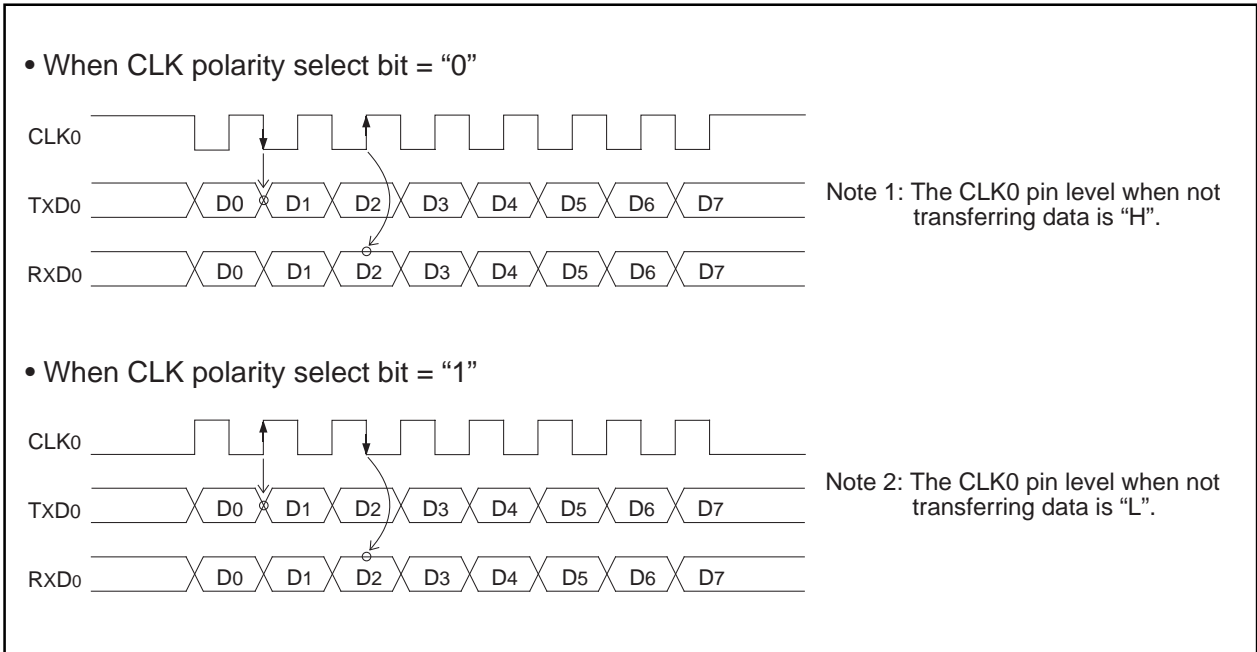


**Figure 1.75. Typical transmit/receive timings in clock synchronous serial I/O mode**

**Clock synchronous serial I/O mode**

**(a) Polarity select function**

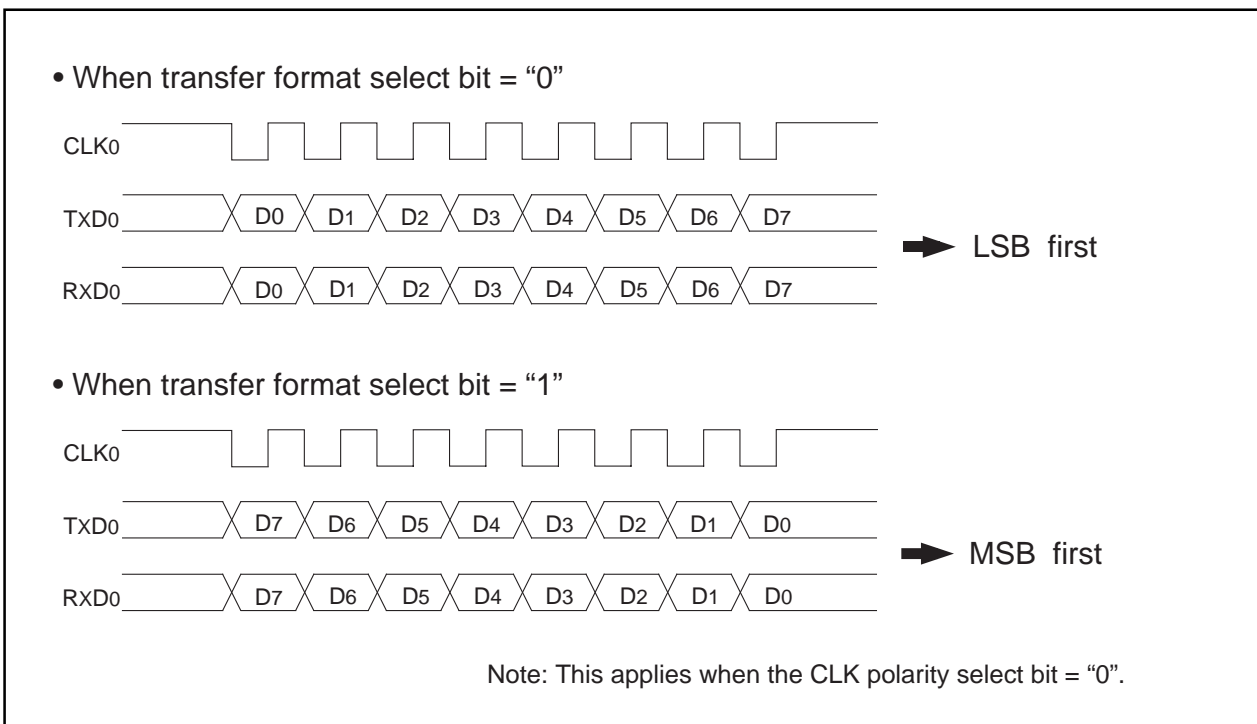
As shown in Figure 1.76, the CLK polarity select bit (bit 6 at addresses 03A4<sub>16</sub>) allows selection of the polarity of the transfer clock.



**Figure 1.76. Polarity of transfer clock**

**(b) LSB first/MSB first select function**

As shown in Figure 1.77, when the transfer format select bit (bit 7 at addresses 03A4<sub>16</sub>) = "0", the transfer format is "LSB first"; when the bit = "1", the transfer format is "MSB first".



**Figure 1.77. Transfer format**

## Clock synchronous serial I/O mode

### (c) Transfer clock output from multiple pins function

This function allows the setting two transfer clock output pins and choosing one of the two to output a clock by using the CLK and CLKS select bit (bits 4 and 5 at address 03B016). (See Figure 1.78.) The multiple pins function is valid only when the internal clock is selected for UART0.

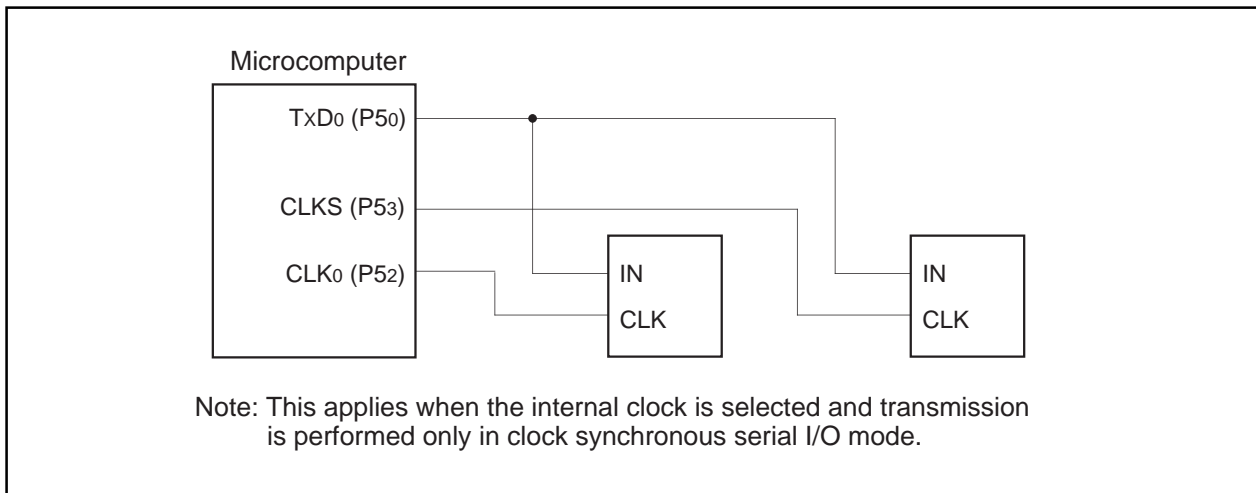


Figure 1.78. The transfer clock output from the multiple pins function usage

### (d) Continuous receive mode

If the continuous receive mode enable bit (bits 2 and 3 at address 03B016) is set to "1", the unit is placed in continuous receive mode. In this mode, when the receive buffer register is read out, the unit simultaneously goes to a receive enable state without having to set dummy data to the transmit buffer register back again.

## Clock asynchronous serial I/O (UART) mode

**(2) Clock asynchronous serial I/O (UART) mode**

The UART mode allows transmitting and receiving data after setting the desired transfer rate and transfer data format. (See Table 1.28.) Figure 1.79 shows the UARTi transmit/receive mode register.

**Table 1.28. Specifications of UART Mode**

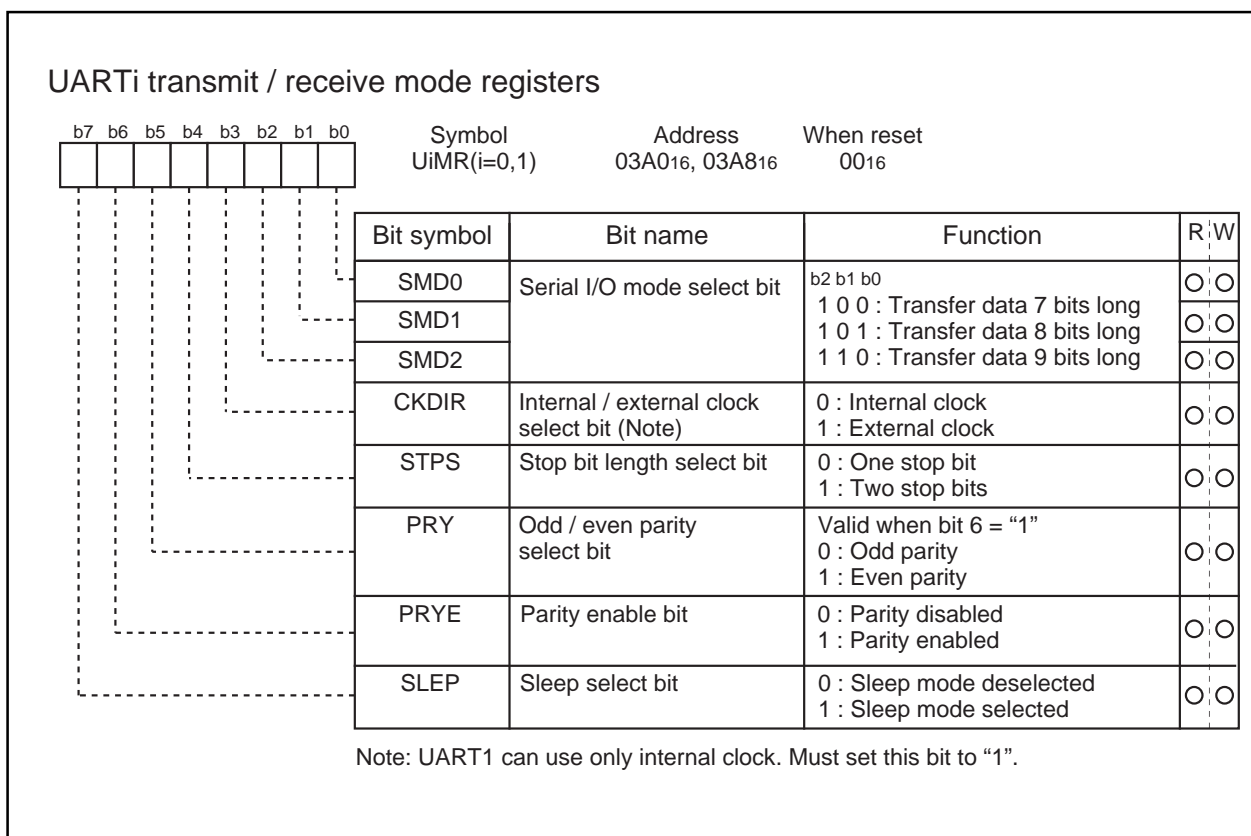
Item	Specification
Transfer data format	<ul style="list-style-type: none"> <li>• Character bit (transfer data): 7 bits, 8 bits, or 9 bits as selected</li> <li>• Start bit: 1 bit</li> <li>• Parity bit: Odd, even, or nothing as selected</li> <li>• Stop bit: 1 bit or 2 bits as selected</li> </ul>
Transfer clock	<ul style="list-style-type: none"> <li>• When internal clock is selected (bit 3 at addresses 03A0<sub>16</sub>, 03A8<sub>16</sub> = "0") : f<sub>i</sub>/16(n+1) (Note 1) f<sub>i</sub> = f<sub>1</sub>, f<sub>8</sub>, f<sub>32</sub>, f<sub>c</sub></li> <li>• When external clock is selected (bit 3 at addresses 03A0<sub>16</sub>="1") : f<sub>EXT</sub>/16(n+1) (Note 1) (Note 2)</li> </ul>
Transmission start condition	<ul style="list-style-type: none"> <li>• To start transmission, the following requirements must be met: <ul style="list-style-type: none"> <li>- Transmit enable bit (bit 0 at addresses 03A5<sub>16</sub>, 03AD<sub>16</sub>) = "1"</li> <li>- Transmit buffer empty flag (bit 1 at addresses 03A5<sub>16</sub>, 03AD<sub>16</sub>) = "0"</li> </ul> </li> </ul>
Reception start condition	<ul style="list-style-type: none"> <li>• To start reception, the following requirements must be met: <ul style="list-style-type: none"> <li>- Receive enable bit (bit 2 at addresses 03A5<sub>16</sub>, 03AD<sub>16</sub>) = "1"</li> <li>- Start bit detection</li> </ul> </li> </ul>
Interrupt request generation timing	<ul style="list-style-type: none"> <li>• When transmitting <ul style="list-style-type: none"> <li>- Transmit interrupt cause select bits (bits 0,1 at address 03B0<sub>16</sub>) = "0": Interrupts requested when data transfer from UARTi transfer buffer register to UARTi transmit register is completed</li> <li>- Transmit interrupt cause select bits (bits 0, 1 at address 03B0<sub>16</sub>) = "1": Interrupts requested when data transmission from UARTi transfer register is completed</li> </ul> </li> <li>• When receiving <ul style="list-style-type: none"> <li>- Interrupts requested when data transfer from UARTi receive register to UARTi receive buffer register is completed</li> </ul> </li> </ul>
Error detection	<ul style="list-style-type: none"> <li>• Overrun error (Note 3) This error occurs when the next data is ready before contents of UARTi receive buffer register are read out</li> <li>• Framing error This error occurs when the number of stop bits set is not detected</li> <li>• Parity error This error occurs when if parity is enabled, the number of 1's in parity and character bits does not match the number of 1's set</li> <li>• Error sum flag This flag is set (= 1) when any of the overrun, framing, and parity errors is encountered</li> </ul>
Select function	<ul style="list-style-type: none"> <li>• Sleep mode selection This mode is used to transfer data to and from one of multiple slave micro-computers</li> </ul>

Note 1: 'n' denotes the value 00<sub>16</sub> to FF<sub>16</sub> that is set to the UART bit rate generator.

Note 2: f<sub>EXT</sub> is input from the CLK0 pin. Since UART1 does not have this pin, cannot select external clock.

Note 3: If an overrun error occurs, the UARTi receive buffer will have the next data written in. Note also that the UARTi receive interrupt request bit is not set to "1".

**Clock asynchronous serial I/O (UART) mode**



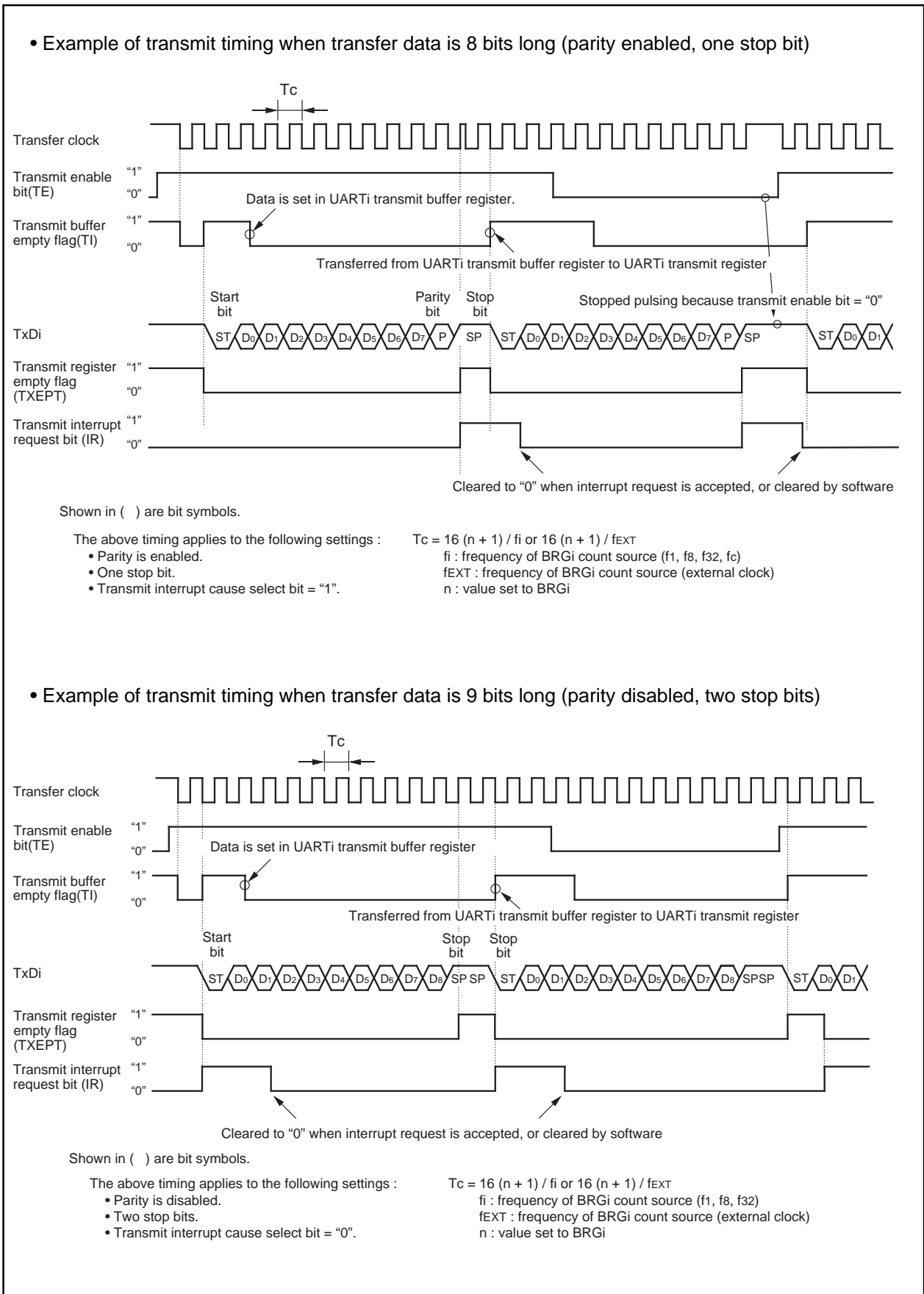
**Figure 1.79. UART<sub>i</sub> transmit/receive mode register in UART mode**

Table 1.29 lists the functions of the input/output pins during UART mode. Note that for a period from when the UART<sub>i</sub> operation mode is selected to when transfer starts, the Tx<sub>Di</sub> pin outputs a "H". (If the N-channel open-drain is selected, this pin is in floating state.)

**Table 1.29. Input/output pin functions in UART mode**

Pin name	Function	Method of selection
TxD <sub>i</sub> (P5 <sub>0</sub> , P4 <sub>0</sub> )	Serial data output	Port P5 <sub>1</sub> and P4 <sub>2</sub> direction register (bit 0 at address 03EB <sub>16</sub> , bit 0 at address 03EA <sub>16</sub> ) = "1" (Can be used as an input port when performing reception only)
RxD <sub>i</sub> (P5 <sub>1</sub> , P4 <sub>2</sub> )	Serial data input	Port P5 <sub>1</sub> and P4 <sub>2</sub> direction register (bit 1 at address 03EB <sub>16</sub> , bit 2 at address 03EA <sub>16</sub> ) = "0" (Can be used as an input port when performing transmission only)
CLK0 (P5 <sub>2</sub> )	Programmable I/O port	Internal/external clock select bit (bit 3 at address 03A0 <sub>16</sub> ) = "0"
	Transfer clock input	Internal/external clock select bit (bit 3 at address 03A0 <sub>16</sub> ) = "1"

**Clock asynchronous serial I/O (UART) mode**



**Figure 1.80. Typical transmit timings in UART mode**

Clock asynchronous serial I/O (UART) mode

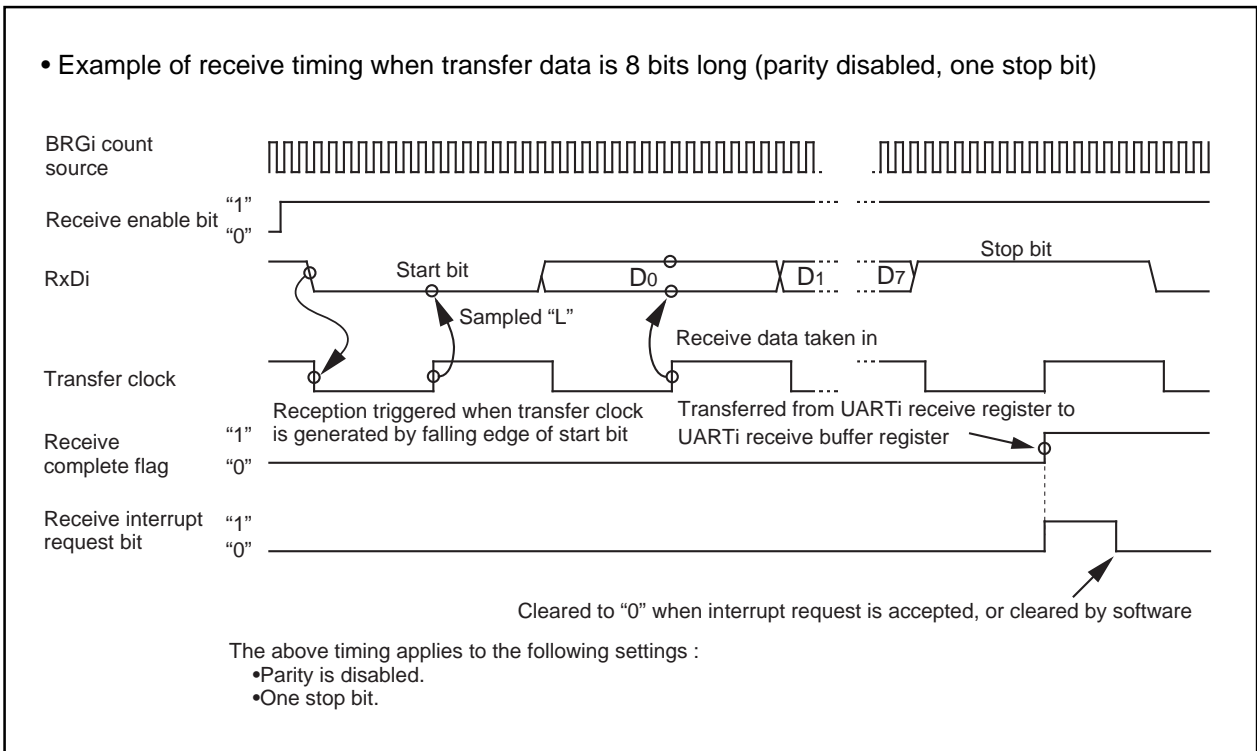


Figure 1.81. Typical receive timing in UART mode

**(a) Sleep mode**

This mode is used to transfer data between specific microcomputers among multiple microcomputers connected using UARTi. The sleep mode is selected when the sleep select bit (bit 7 at addresses 03A016, 03A816) is set to "1" during reception. In this mode, the unit performs receive operation when the MSB of the received data = "1" and does not perform receive operation when the MSB = "0".



## A-D Converter

## A-D Converter

The A-D converter consists of one 10-bit successive approximation A-D converter circuit with a capacitive coupling amplifier. Pins P60 to P67, and P50 to P54 also function as the analog signal input pins. The direction registers of these pins for A-D conversion must therefore be set to input. The Vref connect bit (bit 5 at address 03D716) can be used to isolate the resistance ladder of the A-D converter from the reference voltage input pin (VREF) when the A-D converter is not used. Doing so stops any current flowing into the resistance ladder from VREF, reducing the power dissipation. When using the A-D converter, start A-D conversion only after setting bit 5 of 03D716 to connect VREF.

The result of A-D conversion is stored in the A-D registers of the selected pins. When set to 10-bit precision, the low 8 bits are stored in the even addresses and the high 2 bits in the odd addresses. When set to 8-bit precision, the low 8 bits are stored in the even addresses.

Table 1.30 shows the performance of the A-D converter. Figure 1.82 shows the block diagram of the A-D converter, and Figures 1.83 and 1.84 show the A-D converter-related registers.

Table 1.30. Performance of A-D converter

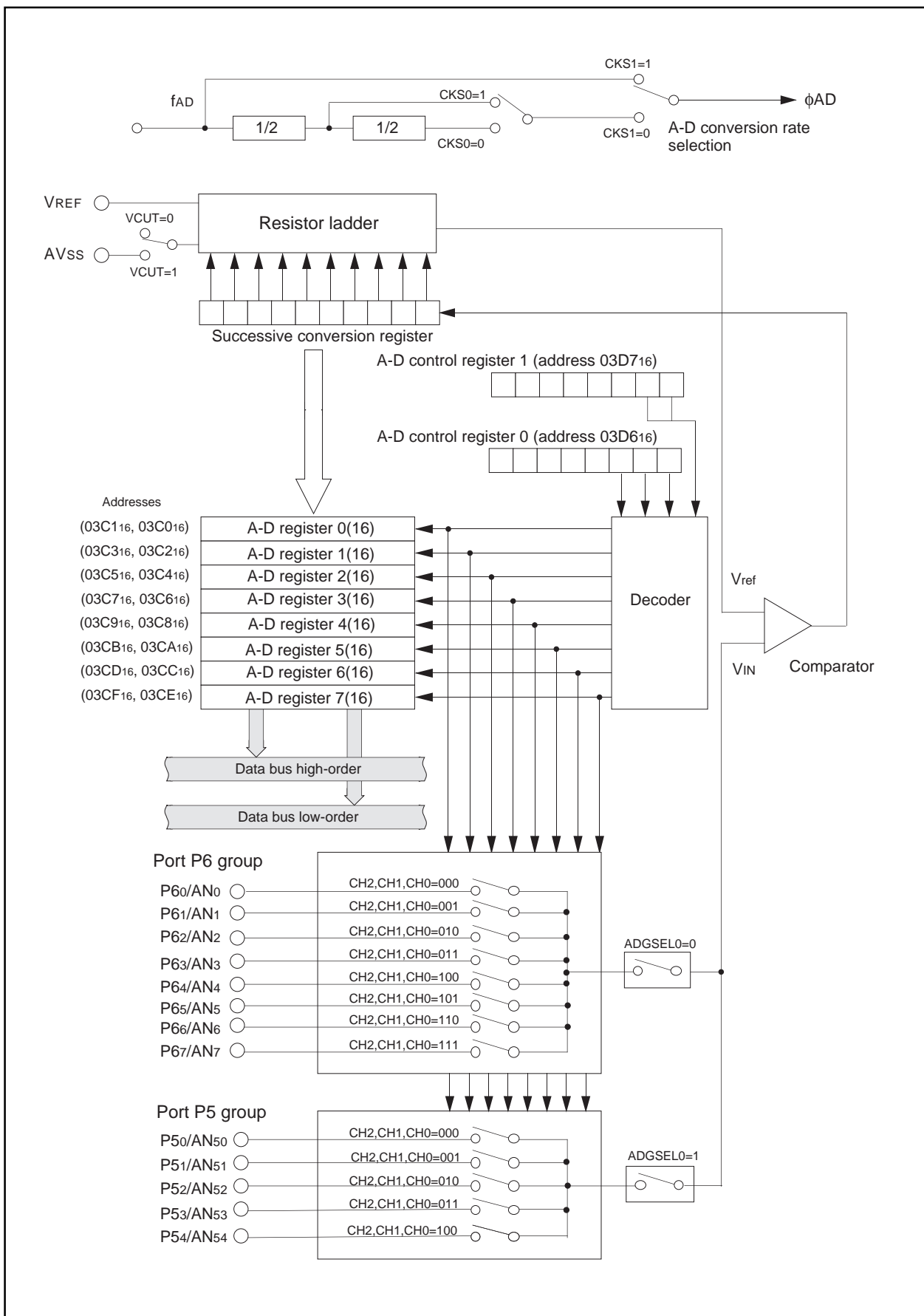
Item	Performance
Method of A-D conversion	Successive approximation (capacitive coupling amplifier)
Analog input voltage (Note 1)	0V to AVCC (VCC)
Operating clock $\phi_{AD}$ (Note 2)	VCC = 5V fAD, divide-by-2 of fAD, divide-by-4 of fAD, fAD=f(XIN) VCC = 3V divide-by-2 of fAD, divide-by-4 of fAD, fAD=f(XIN)
Resolution	8-bit or 10-bit (selectable)
Absolute precision	VCC = 5V <ul style="list-style-type: none"> <li>• Without sample and hold function <math>\pm 3</math>LSB</li> <li>• With sample and hold function (8-bit resolution) <math>\pm 2</math>LSB</li> <li>• With sample and hold function (10-bit resolution) <math>\pm 3</math>LSB</li> </ul> VCC = 3V <ul style="list-style-type: none"> <li>• Without sample and hold function (8-bit resolution) <math>\pm 2</math>LSB</li> </ul>
Operating modes	One-shot mode, repeat mode, single sweep mode, repeat sweep mode 0, and repeat sweep mode 1
Analog input pins	8 pins (AN0 to AN7) + 5 pins (AN50 to AN54)
A-D conversion start condition	• Software trigger A-D conversion starts when the A-D conversion start flag changes to "1"
Conversion speed per pin	• Without sample and hold function 8-bit resolution: 49 $\phi_{AD}$ cycles, 10-bit resolution: 59 $\phi_{AD}$ cycles • With sample and hold function 8-bit resolution: 28 $\phi_{AD}$ cycles, 10-bit resolution: 33 $\phi_{AD}$ cycles

Note 1: Does not depend on use of sample and hold function.

Note 2: Without sample and hold function, set the  $\phi_{AD}$  frequency to 250kHz min.

With the sample and hold function, set the  $\phi_{AD}$  frequency to 1MHz min.

**A-D Converter**



**Figure 1.82. Block diagram of A-D converter**

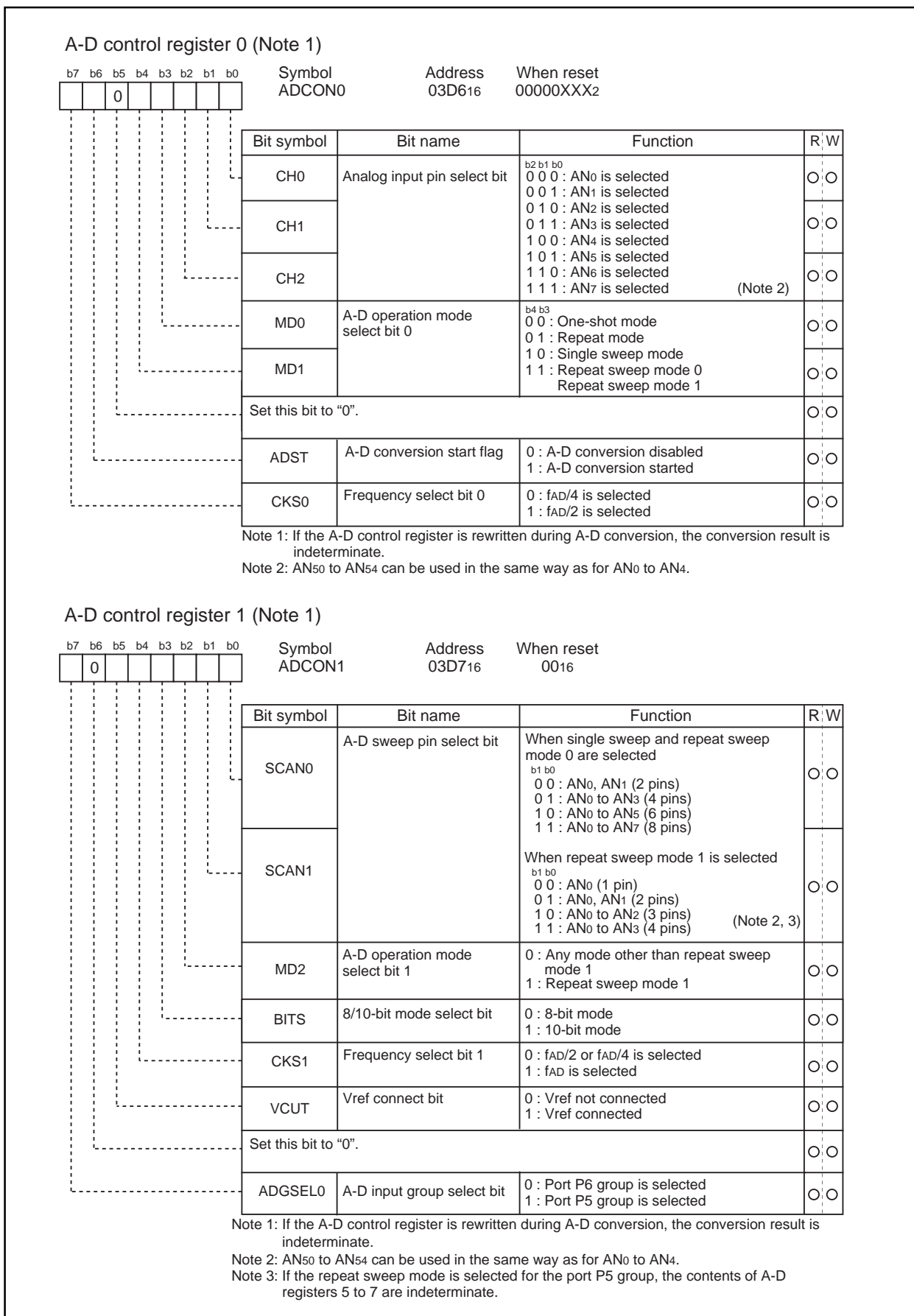
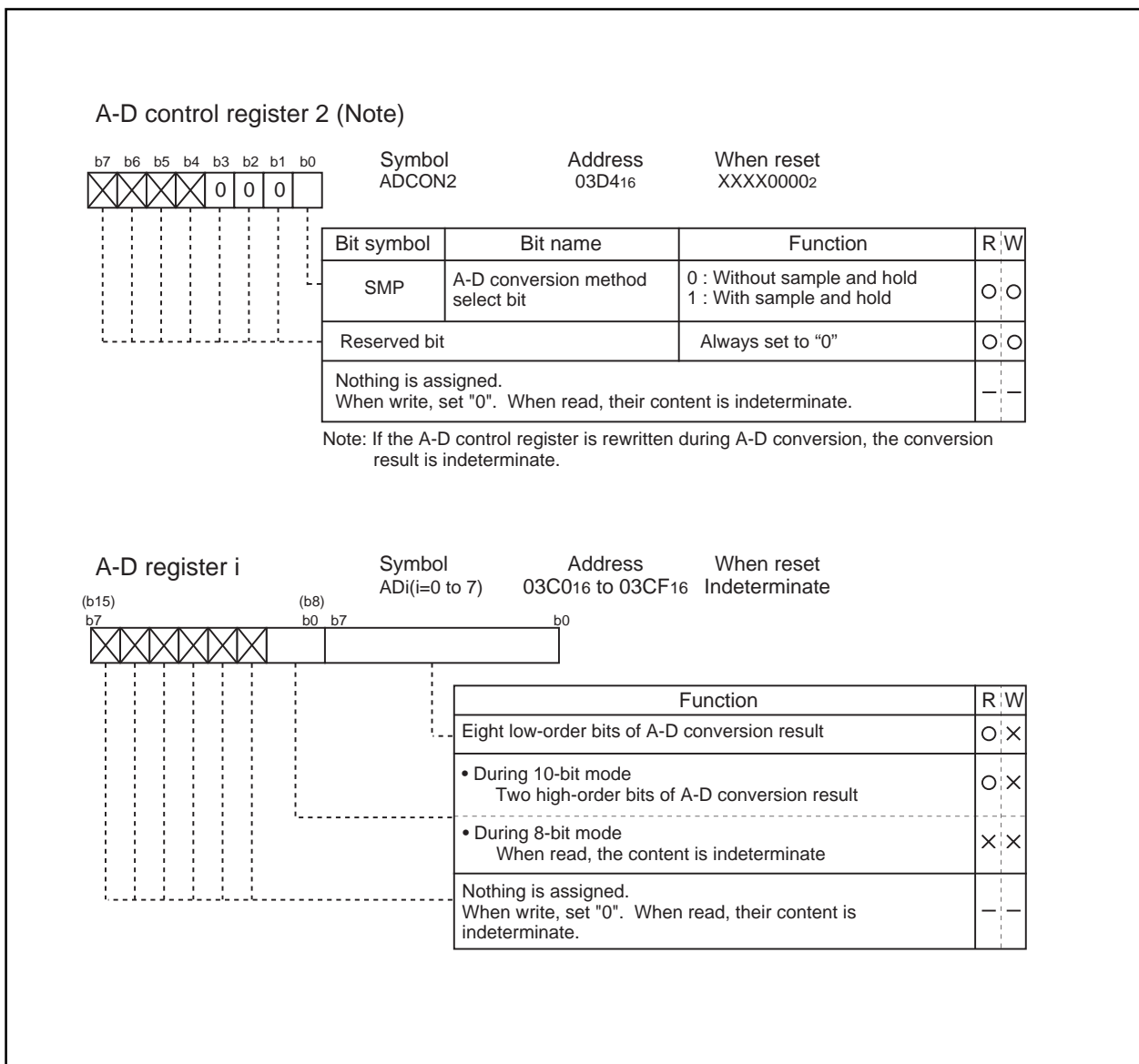


Figure 1.83. A-D converter-related registers (1)

**A-D Converter**



**Figure 1.84. A-D converter-related registers (2)**

A-D Converter

(1) One-shot mode

In one-shot mode, the pin selected using the analog input pin select bit is used for one-shot A-D conversion. (See Table 1.31.) Figure 1.85 shows the A-D control register in one-shot mode.

Table 1.31. One-shot mode specifications

Item	Specification
Function	The pin selected by the analog input pin select bit is used for one A-D conversion
Start condition	Writing "1" to A-D conversion start flag
Stop condition	<ul style="list-style-type: none"> <li>• End of A-D conversion (A-D conversion start flag changes to "0")</li> <li>• Writing "0" to A-D conversion start flag</li> </ul>
Interrupt request generation timing	End of A-D conversion
Input pin	One of AN <sub>0</sub> to AN <sub>7</sub> , as selected (Note)
Reading of result of A-D converter	Read A-D register corresponding to selected pin

Note : AN<sub>50</sub> to AN<sub>54</sub> can be used in the same way as for AN<sub>0</sub> to AN<sub>4</sub>.

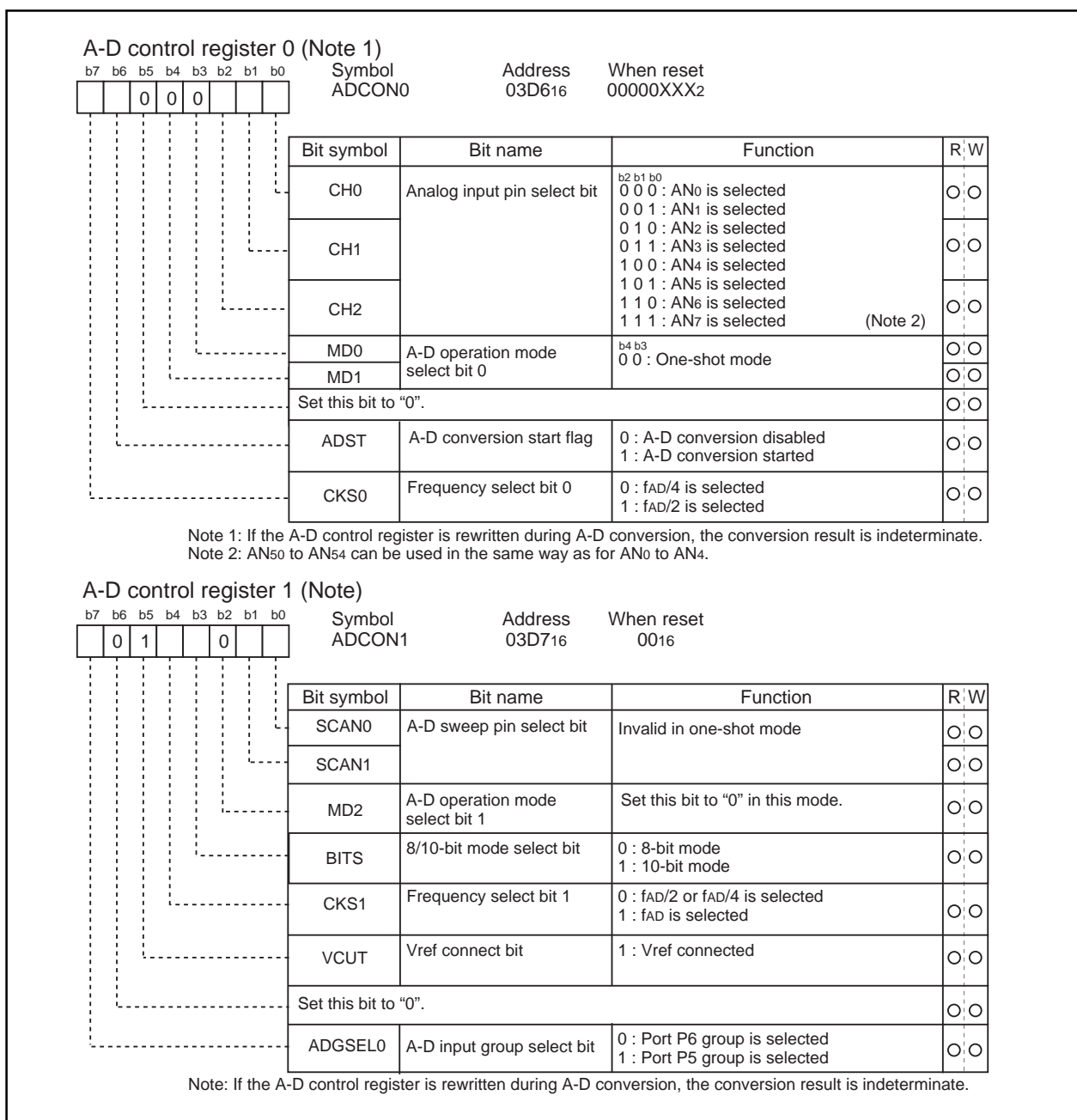


Figure 1.85. A-D conversion register in one-shot mode

A-D Converter

(2) Repeat mode

In repeat mode, the pin selected using the analog input pin select bit is used for repeated A-D conversion. (See Table 1.32.) Figure 1.86 shows the A-D control register in repeat mode.

Table 1.32. Repeat mode specifications

Item	Specification
Function	The pin selected by the analog input pin select bit is used for repeated A-D conversion
Start condition	Writing "1" to A-D conversion start flag
Stop condition	Writing "0" to A-D conversion start flag
Interrupt request generation timing	None generated
Input pin	One of AN0 to AN7, as selected (Note)
Reading of result of A-D converter	Read A-D register corresponding to selected pin

Note : AN50 to AN54 can be used in the same way as for AN0 to AN4.

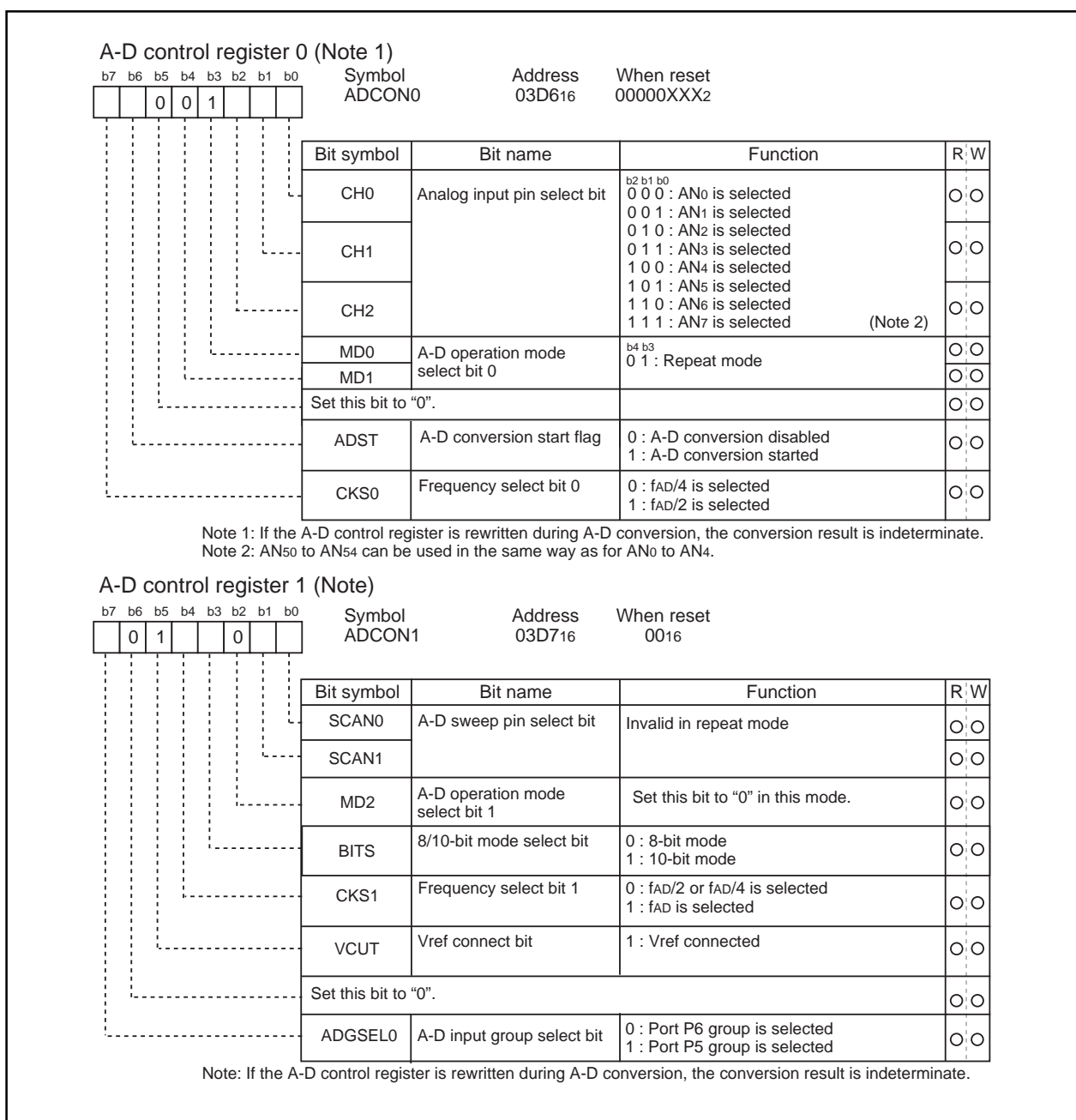


Figure 1.86. A-D conversion register in repeat mode

**A-D Converter**

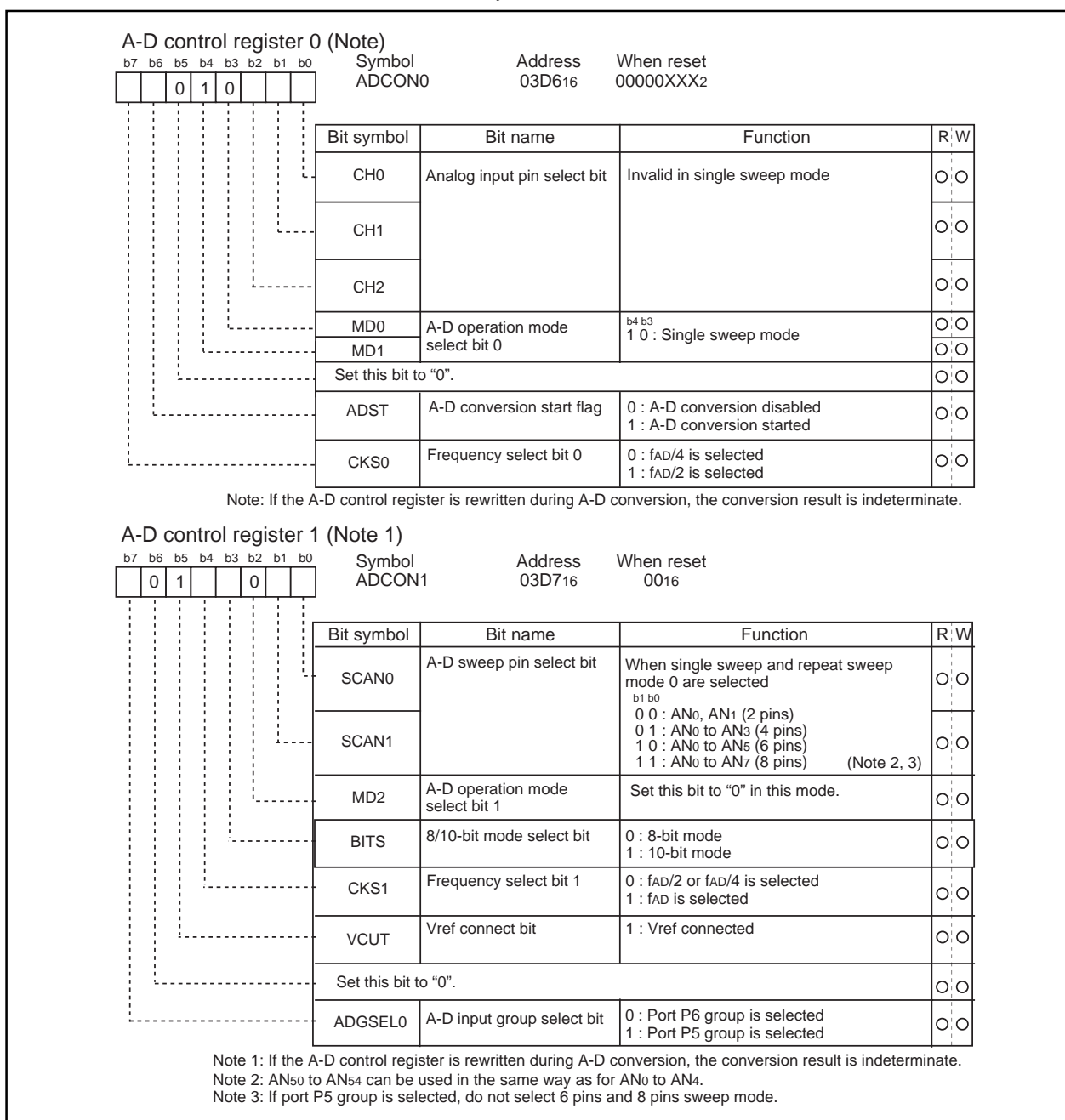
**(3) Single sweep mode**

In single sweep mode, the pins selected using the A-D sweep pin select bit are used for one-by-one A-D conversion. (See Table 1.33.) Figure 1.87 shows the A-D control register in single sweep mode.

**Table 1.33. Single sweep mode specifications**

Item	Specification
Function	The pins selected by the A-D sweep pin select bit are used for one-by-one A-D conversion
Start condition	Writing "1" to A-D converter start flag
Stop condition	<ul style="list-style-type: none"> <li>• End of A-D conversion (A-D conversion start flag changes to "0".)</li> <li>• Writing "0" to A-D conversion start flag</li> </ul>
Interrupt request generation timing	End of A-D conversion
Input pin	AN0 and AN1 (2 pins), AN0 to AN3 (4 pins), AN0 to AN5 (6 pins), or AN0 to AN7 (8 pins)(Note)
Reading of result of A-D converter	Read A-D register corresponding to selected pin

Note : AN50 to AN54 can be used in the same way as for AN0 to AN4.



**Figure 1.87. A-D conversion register in single sweep mode**

A-D Converter

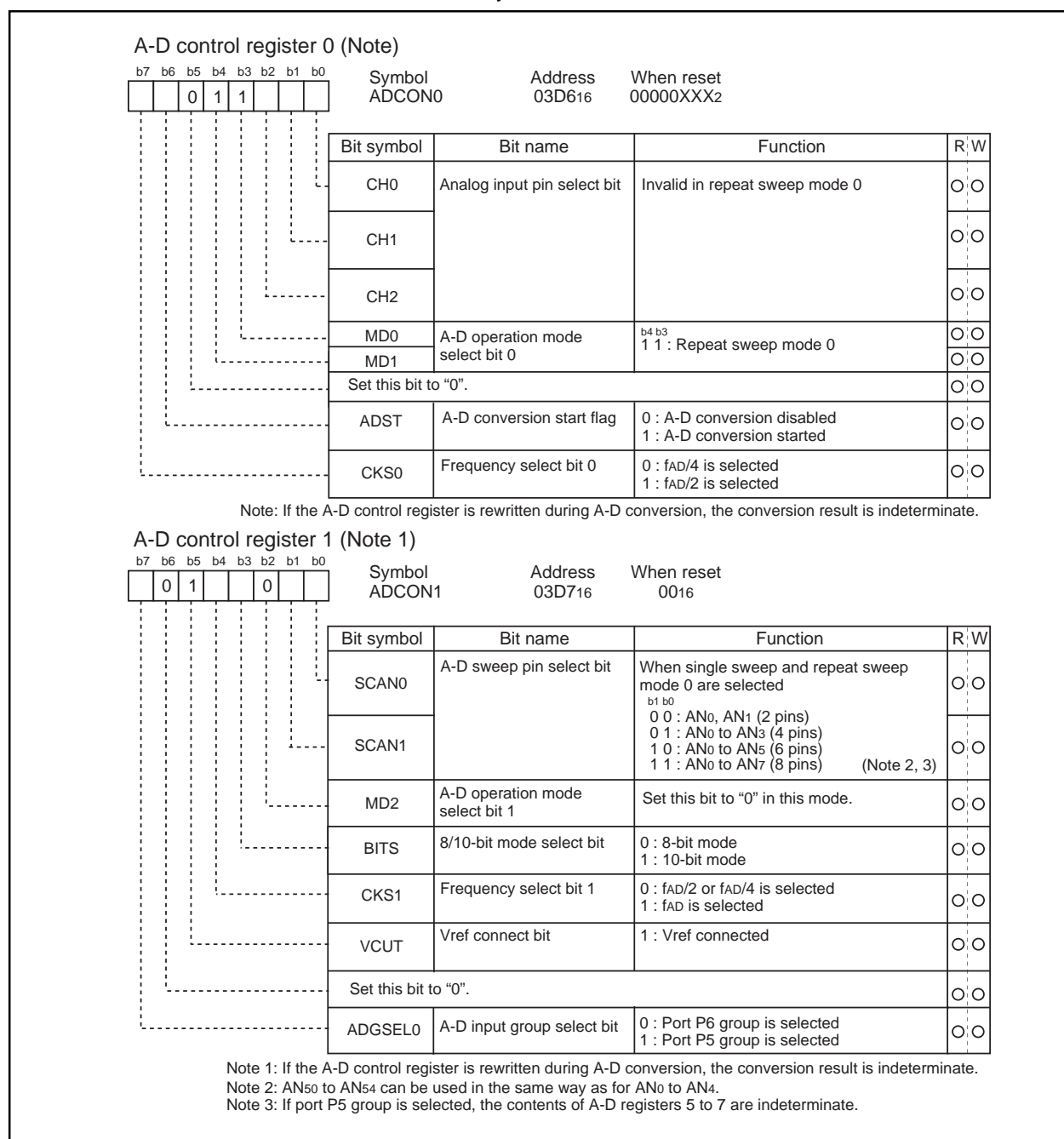
**(4) Repeat sweep mode 0**

In repeat sweep mode 0, the pins selected using the A-D sweep pin select bit are used for repeat sweep A-D conversion. (See Table 1.34.) Figure 1.88 shows the A-D control register in repeat sweep mode 0.

**Table 1.34. Repeat sweep mode 0 specifications**

Item	Specification
Function	The pins selected by the A-D sweep pin select bit are used for repeat sweep A-D conversion
Start condition	Writing "1" to A-D conversion start flag
Stop condition	Writing "0" to A-D conversion start flag
Interrupt request generation timing	None generated
Input pin	AN0 and AN1 (2 pins), AN0 to AN3 (4 pins), AN0 to AN5 (6 pins), or AN0 to AN7 (8 pins)(Note)
Reading of result of A-D converter	Read A-D register corresponding to selected pin (at any time)

Note : AN50 to AN54 can be used in the same way as for AN0 to AN4.



**Figure 1.88. A-D conversion register in repeat sweep mode 0**



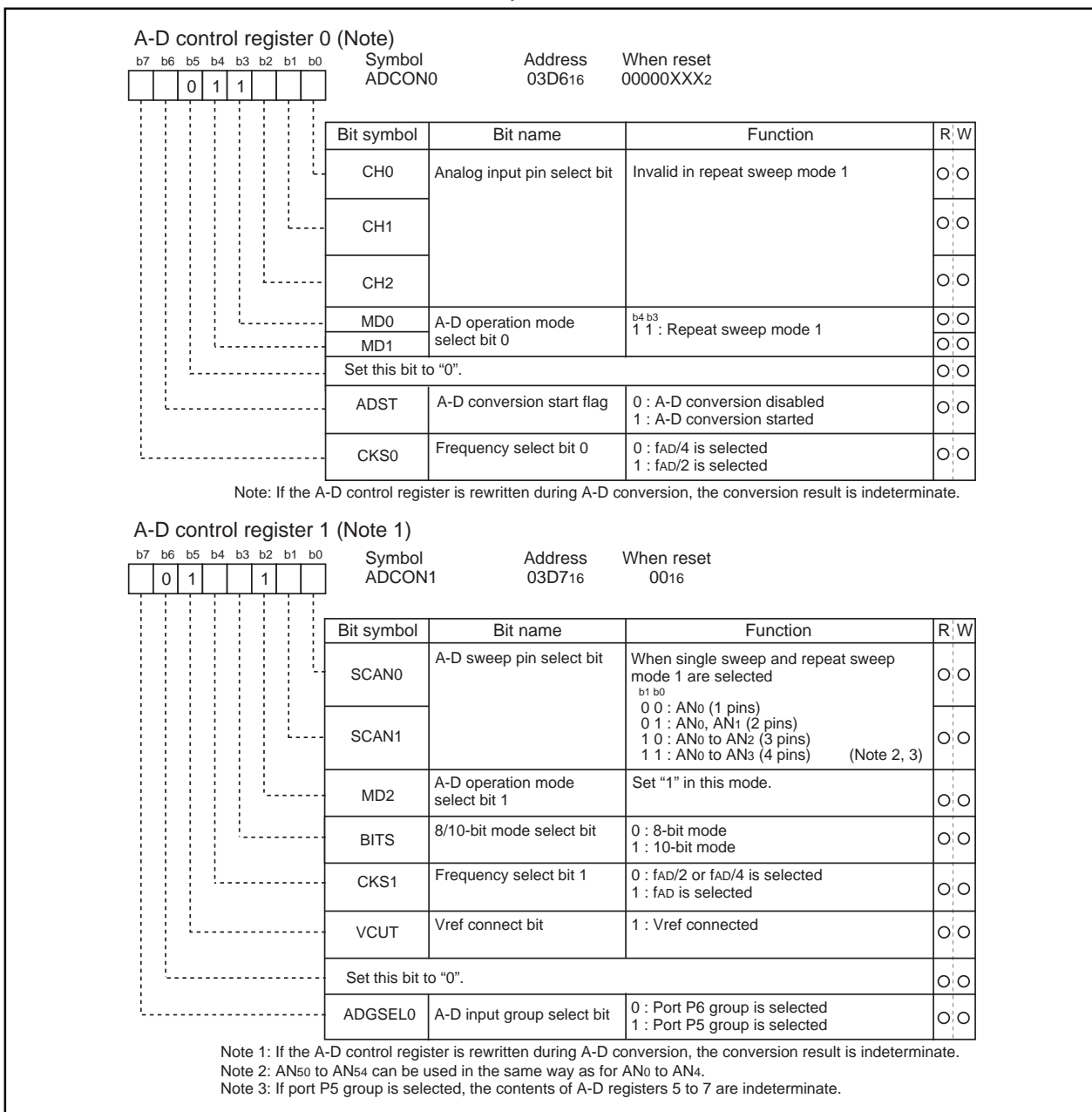
### (5) Repeat sweep mode 1

In repeat sweep mode 1, all pins are used for A-D conversion with emphasis on the pin or pins selected using the A-D sweep pin select bit. (See Table 1.35.) Figure 1.89 shows the A-D control register in repeat sweep mode

**Table 1.35. Repeat sweep mode 1 specifications**

Item	Specification
Function	All pins perform repeat sweep A-D conversion, with emphasis on the pin or pins selected by the A-D sweep pin select bit Example : AN <sub>0</sub> selected AN <sub>0</sub> → AN <sub>1</sub> → AN <sub>0</sub> → AN <sub>2</sub> → AN <sub>0</sub> → AN <sub>3</sub> , etc
Start condition	Writing "1" to A-D conversion start flag
Stop condition	Writing "0" to A-D conversion start flag
Interrupt request generation timing	None generated
Input pin	AN <sub>0</sub> (1 pin), AN <sub>0</sub> and AN <sub>1</sub> (2 pins), AN <sub>0</sub> to AN <sub>2</sub> (3 pins), AN <sub>0</sub> to AN <sub>3</sub> (4 pins) (Note)
Reading of result of A-D converter	Read A-D register corresponding to selected pin (at any time)

Note : AN<sub>50</sub> to AN<sub>54</sub> can be used in the same way as for AN<sub>0</sub> to AN<sub>4</sub>.



**Figure 1.89. A-D conversion register in repeat sweep mode 1**

## A-D Converter

---

- **Sample and hold**

Sample and hold is selected by setting bit 0 of the A-D control register 2 (address 03D416) to "1". When sample and hold is selected, the rate of conversion of each pin increases. As a result, a 28  $\phi_{AD}$  cycle is achieved with 8-bit resolution and 33  $\phi_{AD}$  with 10-bit resolution. Sample and hold can be selected in all modes. However, in all modes, be sure to specify before starting A-D conversion whether sample and hold is to be used.

## Programmable I/O Port

---

### Programmable I/O Ports

There are 43 programmable I/O ports: P0 to P7. Each port can be set independently for input or output using the direction register. A pull-up resistance for each block of 4 ports can be set. The port P1 allows the drive capacity of its N-channel output transistor to be set as necessary.

Figures 1.90 to 1.92 show the programmable I/O ports.

Each pin functions as a programmable I/O port and as the I/O for the built-in peripheral devices.

To use the pins as the inputs for the built-in peripheral devices, set the direction register of each pin to input mode. When the pins are used as the outputs for the built-in peripheral devices, they function as outputs regardless of the contents of the direction registers. See the descriptions of the respective functions for how to set up the built-in peripheral devices.

#### (1) Direction registers

Figure 1.93 shows the direction registers.

These registers are used to choose the direction of the programmable I/O ports. Each bit in these registers corresponds one for one to each I/O pin.

#### (2) Port registers

Figure 1.94 shows the port registers.

These registers are used to write and read data for input and output to and from an external device. A port register consists of a port latch to hold output data and a circuit to read the status of a pin. Each bit in port registers corresponds one for one to each I/O pin.

#### (3) Pull-up control registers

Figure 1.95 shows the pull-up control registers.

The pull-up control register can be set to apply a pull-up resistance to each block of 4 ports. When ports are set to have a pull-up resistance, the pull-up resistance is connected only when the direction register is set for input.

#### (4) Port P1 drive capacity control register

Figure 1.95 shows a structure of the port P1 drive capacity control register.

This register is used to control the drive capacity of the port P1's N-channel output transistor. Each bit in this register corresponds one for one to the port pins.

Programmable I/O Port

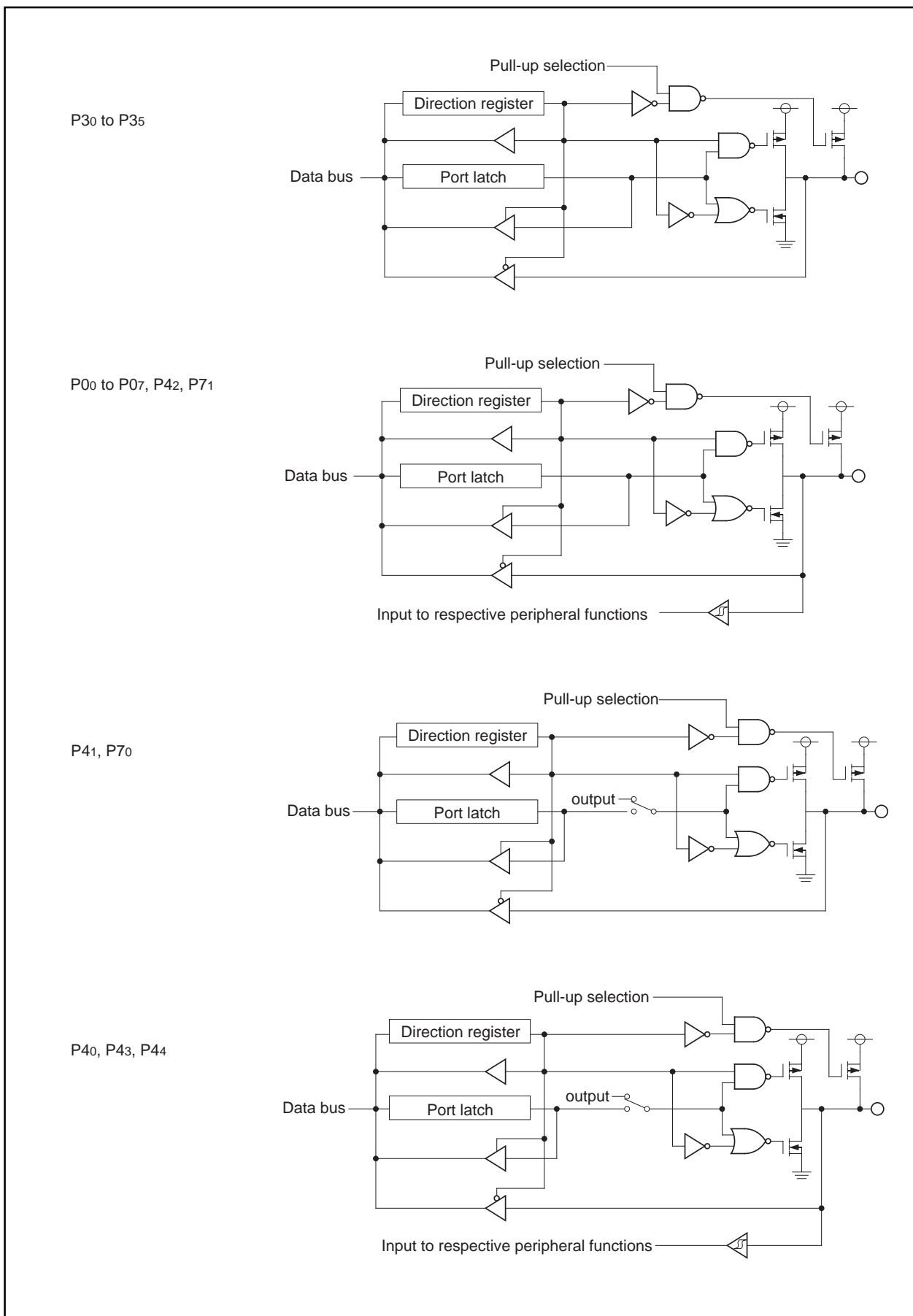


Figure 1.90. Programmable I/O ports (1)

Programmable I/O Port

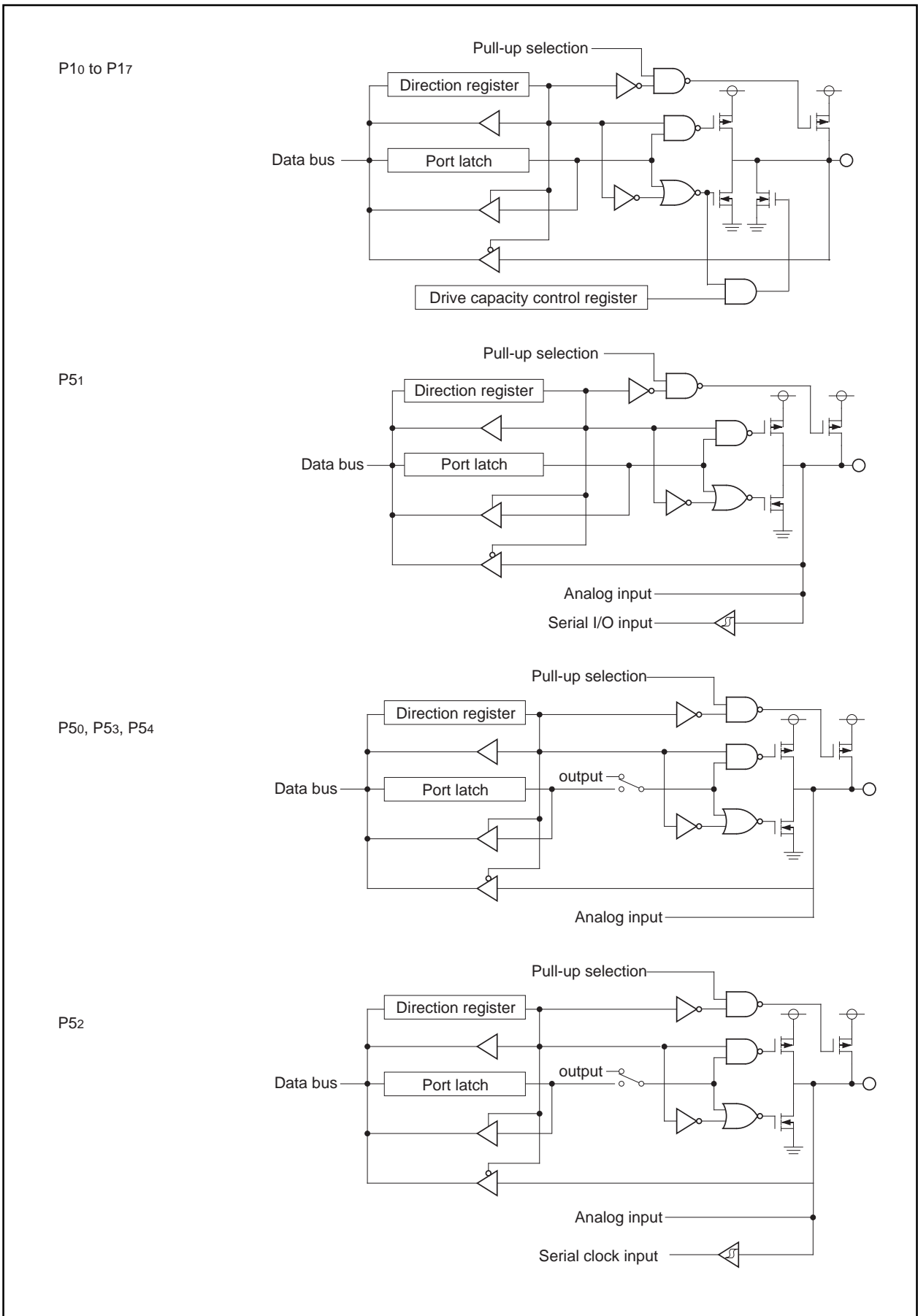


Figure 1.91. Programmable I/O ports (2)

# Programmable I/O Port

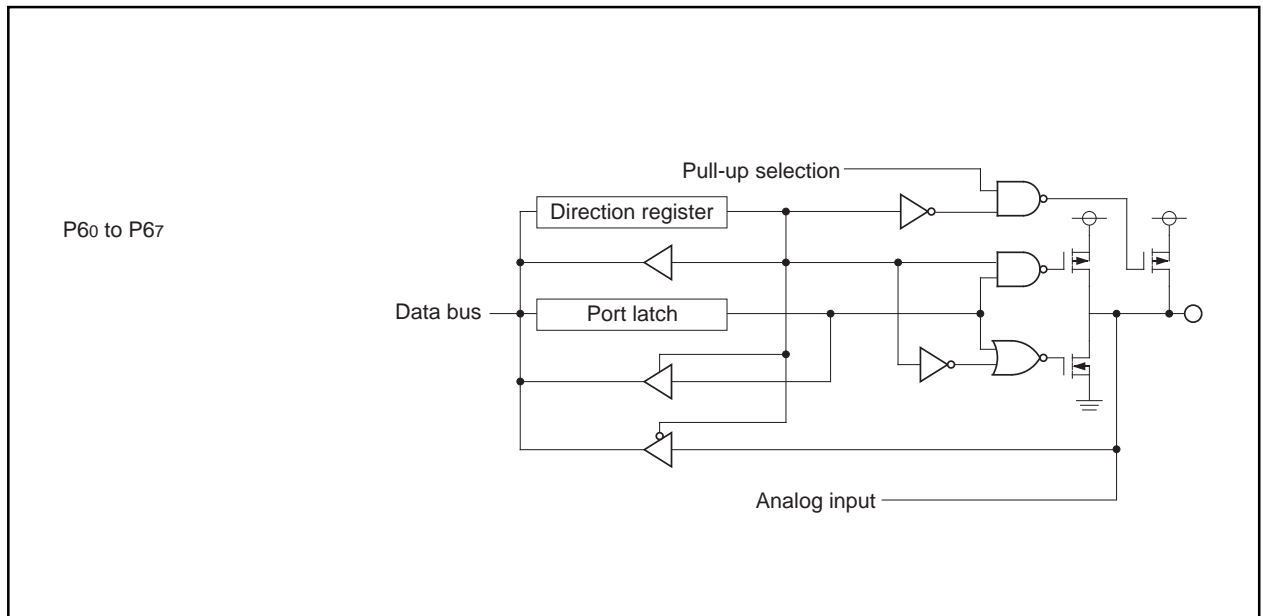


Figure 1.92. Programmable I/O ports (3)

Programmable I/O Port

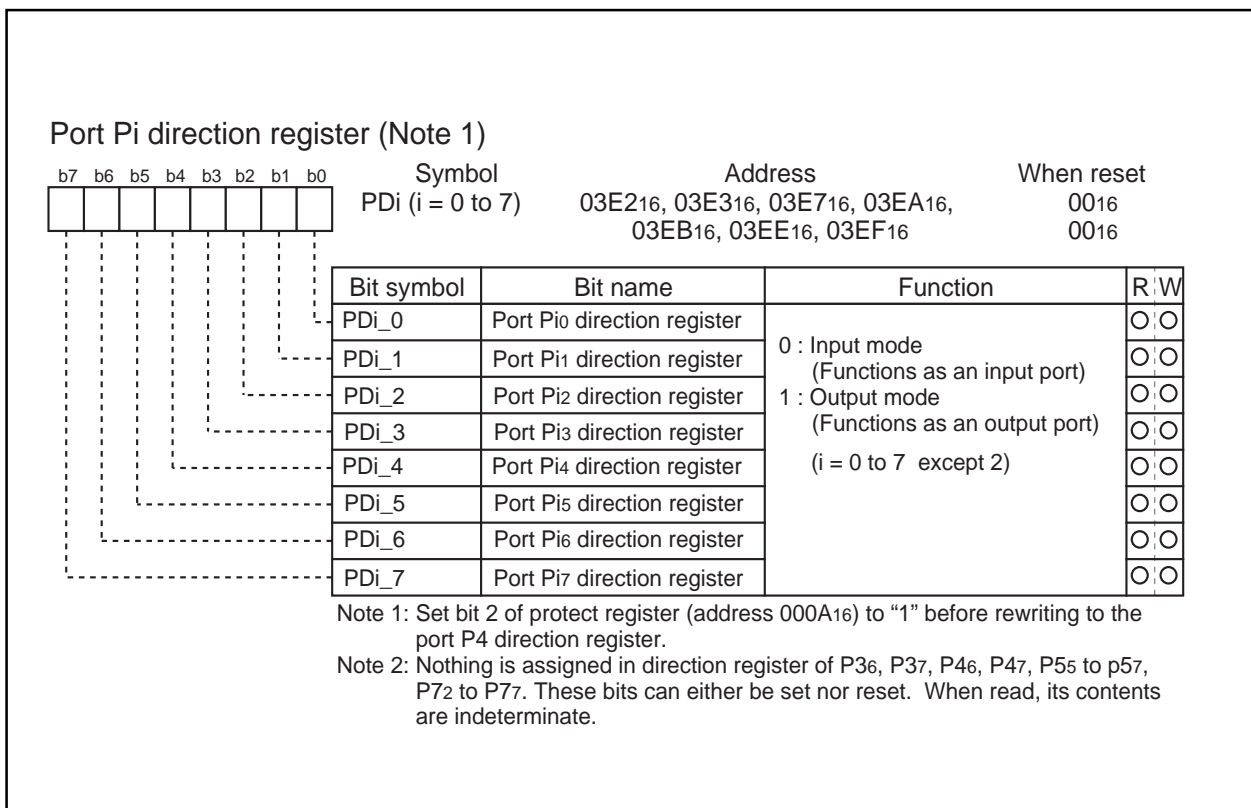


Figure 1.93. Direction register

Programmable I/O Port

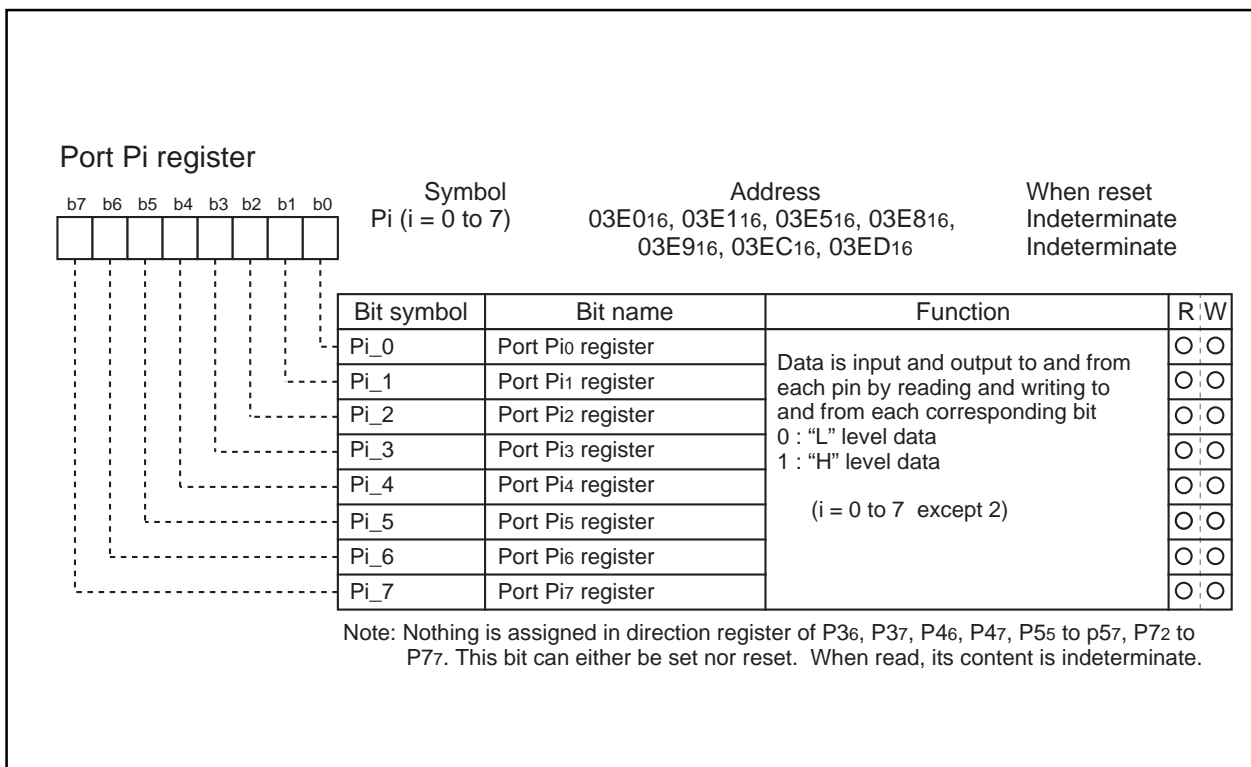


Figure 1.94. Port register



Programmable I/O Port

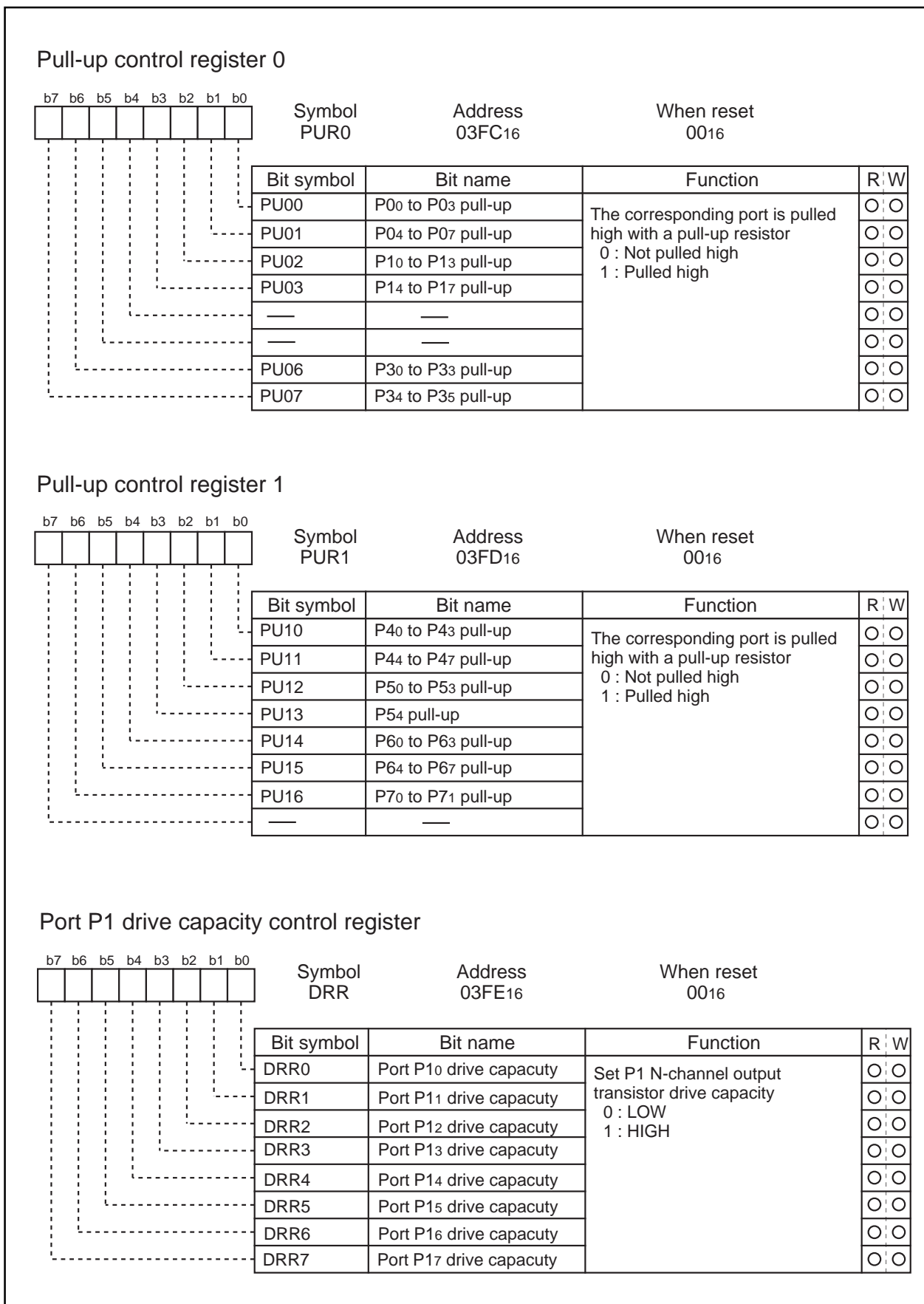


Figure 1.95. Pull-up control register

## Example connection of unused pins

**Table 1.36. Example connection of unused pins**

Pin name	Connection
Ports P0, P1, P3 to P7	After setting for input mode, connect every pin to Vss (pull-down); or after setting for output mode, leave these pins open.
XOUT (Note)	Open
AVcc	Connect to Vcc
AVSS, VREF	Connect to Vss

Note: With external clock input to XIN pin.

## Usage precaution

---

### Usage Precaution

#### Timer A (timer mode)

- (1) Reading the timer A0 register while a count is in progress allows reading, with arbitrary timing, the value of the counter. Reading the timer A0 register with the reload timing gets "FFFF16". Reading the timer A0 register after setting a value in the timer A0 register with a count halted but before the counter starts counting gets a proper value.

#### Timer A (event counter mode)

- (1) Reading the timer A0 register while a count is in progress allows reading, with arbitrary timing, the value of the counter. Reading the timer A0 register with the reload timing gets "FFFF16" by under-flow or "000016" by overflow. Reading the timer A0 register after setting a value in the timer A0 register with a count halted but before the counter starts counting gets a proper value.
- (2) When stop counting in free run type, set timer again.

#### Timer A (one-shot timer mode)

- (1) Setting the count start flag to "0" while a count is in progress causes as follows:
  - The counter stops counting and a content of reload register is reloaded.
  - The TA0OUT pin outputs "L" level.
  - The interrupt request generated and the timer A0 interrupt request bit goes to "1".
- (2) The timer A0 interrupt request bit goes to "1" if the timer's operation mode is set using any of the following procedures:
  - Selecting one-shot timer mode after reset.
  - Changing operation mode from timer mode to one-shot timer mode.
  - Changing operation mode from event counter mode to one-shot timer mode.

Therefore, to use timer A0 interrupt (interrupt request bit), set timer A0 interrupt request bit to "0" after the above listed changes have been made.

#### Timer A (pulse width modulation mode)

- (1) The timer A0 interrupt request bit becomes "1" if setting operation mode of the timer in compliance with any of the following procedures:
  - Selecting PWM mode after reset.
  - Changing operation mode from timer mode to PWM mode.
  - Changing operation mode from event counter mode to PWM mode.

Therefore, to use timer A0 interrupt (interrupt request bit), set timer A0 interrupt request bit to "0" after the above listed changes have been made.

- (2) Setting the count start flag to "0" while PWM pulses are being output causes the counter to stop counting. If the TA0OUT pin is outputting an "H" level in this instance, the output level goes to "L", and the timer A0 interrupt request bit goes to "1". If the TA0OUT pin is outputting an "L" level in this instance, the level does not change, and the timer A0 interrupt request bit does not becomes "1".

## Usage precaution

---

### Timer B (timer mode, event counter mode)

- (1) Reading the timer Bi register while a count is in progress allows reading, with arbitrary timing, the value of the counter. Reading the timer Bi register with the reload timing gets "FFFF16". Reading the timer Bi register after setting a value in the timer Bi register with a count halted but before the counter starts counting gets a proper value.

### Timer B (pulse period/pulse width measurement mode)

- (1) If changing the measurement mode select bit is set after a count is started, the timer Bi interrupt request bit goes to "1".
- (2) When the first effective edge is input after a count is started, an indeterminate value is transferred to the reload register. At this time, timer Bi interrupt request is not generated.

### Timer X (timer mode)

- (1) Reading the timer Xi register while a count is in progress allows reading, with arbitrary timing, the value of the counter. Reading the timer Xi register with the reload timing gets "FFFF16". Reading the timer A0 register after setting a value in the timer Xi register with a count halted but before the counter starts counting gets a proper value.

### Timer X (event counter mode)

- (1) Reading the timer Xi register while a count is in progress allows reading, with arbitrary timing, the value of the counter. Reading the timer Xi register with the reload timing gets "FFFF16" by underflow or "000016" by overflow. Reading the timer Xi register after setting a value in the timer Xi register with a count halted but before the counter starts counting gets a proper value.
- (2) When stop counting in free run type, set timer again.

### Timer X (one-shot timer mode)

- (1) Setting the count start flag to "0" while a count is in progress causes as follows:
  - The counter stops counting and a content of reload register is reloaded.
  - The TXiINOUT pin outputs "L" level.
  - The interrupt request generated and the timer Xi interrupt request bit goes to "1".
- (2) The timer Xi interrupt request bit goes to "1" if the timer's operation mode is set using any of the following procedures:
  - Selecting one-shot timer mode after reset.
  - Changing operation mode from timer mode to one-shot timer mode.
  - Changing operation mode from event counter mode to one-shot timer mode.

Therefore, to use timer Xi interrupt (interrupt request bit), set timer Xi interrupt request bit to "0" after the above listed changes have been made.

## Usage precaution

---

### Timer X (pulse width modulation mode)

(1) The timer Xi interrupt request bit becomes “1” if setting operation mode of the timer in compliance with any of the following procedures:

- Selecting PWM mode after reset.
- Changing operation mode from timer mode to PWM mode.
- Changing operation mode from event counter mode to PWM mode.

Therefore, to use timer Xi interrupt (interrupt request bit), set timer Xi interrupt request bit to “0” after the above listed changes have been made.

(2) Setting the count start flag to “0” while PWM pulses are being output causes the counter to stop counting. If the TXiINOUT pin is outputting an “H” level in this instance, the output level goes to “L”, and the timer Xi interrupt request bit goes to “1”. If the TXiINOUT pin is outputting an “L” level in this instance, the level does not change, and the timer Xi interrupt request bit does not becomes “1”.

### Timer X (pulse period/pulse width measurement mode)

- (1) If changing the measurement mode select bit is set after a count is started, the timer Xi interrupt request bit goes to “1”.
- (2) When the first effective edge is input after a count is started, an indeterminate value is transferred to the reload register. At this time, timer Xi interrupt request is not generated.

### A-D Converter

(1) Write to each bit (except bit 6) of A-D control register 0, to each bit of A-D control register 1, and to bit 0 of A-D control register 2 when A-D conversion is stopped (before a trigger occurs).

In particular, when the Vref connection bit is changed from “0” to “1”, start A-D conversion after an elapse of 1  $\mu$ s or longer.

(2) When changing A-D operation mode, select analog input pin again.

(3) Using one-shot mode or single sweep mode

Read the correspondence A-D register after confirming A-D conversion is finished. (It is known by A-D conversion interrupt request bit.)

(4) Using repeat mode, repeat sweep mode 0 or repeat sweep mode 1

Use the undivided main clock as the internal CPU clock.

### Stop Mode and Wait Mode

(1) When returning from stop mode by hardware reset,  $\overline{\text{RESET}}$  pin must be set to “L” level until main clock oscillation is stabilized.

(2) When shifting to WAIT mode or STOP mode, the program stops after reading 8 bytes from the WAIT instruction and the instruction that sets all clock stop bits to “1” in the instruction queue. Therefore, insert a minimum of 8 NOPs after the WAIT instruction and the instruction that sets all clock stop bits to “1”.

## Usage precaution

---

### Interrupts

#### (1) Reading address 00000<sub>16</sub>

- When maskable interrupt is occurred, CPU read the interrupt information (the interrupt number and interrupt request level) in the interrupt sequence.

The interrupt request bit of the certain interrupt written in address 00000<sub>16</sub> will then be set to "0".

Reading address 00000<sub>16</sub> by software sets enabled highest priority interrupt source request bit to "0".

Though the interrupt is generated, the interrupt routine may not be executed.

Do not read address 00000<sub>16</sub> by software.

#### (2) Setting the stack pointer

- The value of the stack pointer immediately after reset is initialized to 0000<sub>16</sub>. Accepting an interrupt before setting a value in the stack pointer may become a factor of runaway. Be sure to set a value in the stack pointer before accepting an interrupt.

Concerning the first instruction immediately after reset, generating any interrupt is prohibited.

#### (3) External interrupt

- When changing a polarity of pins  $\overline{\text{INT0}}$  and  $\overline{\text{INT1}}$ , the interrupt request bit may become "1". Clear the interrupt request bit after changing the polarity.

#### (4) Changing interrupt control register

See "Changing Interrupt Control Register".

## Electrical characteristics

## Electrical characteristics

Table 1.37. Absolute maximum ratings

Symbol	Parameter	Condition	Rated value	Unit
V <sub>cc</sub>	Supply voltage		- 0.3 to 7	V
AV <sub>cc</sub>	Analog supply voltage		- 0.3 to 7	V
V <sub>I</sub>	Input voltage RESET, CNV <sub>ss</sub> , P0 <sub>0</sub> to P0 <sub>7</sub> , P1 <sub>0</sub> to P1 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>5</sub> , P4 <sub>0</sub> to P4 <sub>5</sub> , P5 <sub>0</sub> to P5 <sub>4</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub> , VREF, XIN		- 0.3 to V <sub>cc</sub> + 0.3 (Note 1)	V
V <sub>O</sub>	Output voltage P0 <sub>0</sub> to P0 <sub>7</sub> , P1 <sub>0</sub> to P1 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>5</sub> , P4 <sub>0</sub> to P4 <sub>5</sub> , P5 <sub>0</sub> to P5 <sub>4</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub> , VREF, XIN		- 0.3 to V <sub>cc</sub> + 0.3	V
P <sub>d</sub>	Power dissipation	T <sub>a</sub> = 25 °C	1000 (Note 2)	mW
T <sub>opr</sub>	Operating ambient temperature		- 20 to 85 (Note 3)	°C
T <sub>stg</sub>	Storage temperature		- 40 to 150 (Note 4)	°C

Note 1: When writing to frash MCU, CNV<sub>ss</sub> is -0.3 to 13 (V) .

Note 2: Flat package (56P6S-A) is 300 mW.

Note 3: Extended operating temperature version: -40 to 85 °C.

Note 4: Extended operating temperature version: -65 to 150 °C.

Electrical characteristics (V<sub>cc</sub> = 5V)

V<sub>CC</sub> = 5V

**Table 1.38. Recommended operating conditions (Note 1)**

Symbol	Parameter		Standard			Unit	
			Min	Typ.	Max.		
V <sub>cc</sub>	Supply voltage (Note 2)					V	
		Mask ROM version	2.7	5.0	5.5		
		Flash memory version	4.0	5.0	5.5		
AV <sub>cc</sub>	Analog supply voltage			V <sub>cc</sub>		V	
V <sub>ss</sub>	Supply voltage			0		V	
AV <sub>ss</sub>	Analog supply voltage			0		V	
V <sub>IH</sub>	HIGH input voltage	P00 to P07, P10 to P17, P30 to P35, P40 to P45, P50 to P54, P60 to P67, P70, P71, X <sub>IN</sub> , RESET, CNV <sub>SS</sub> ,	0.8V <sub>cc</sub>		V <sub>cc</sub>	V	
V <sub>IL</sub>	LOW input voltage	P00 to P07, P10 to P17, P30 to P35, P40 to P45, P50 to P54, P60 to P67, P70, P71, X <sub>IN</sub> , RESET, CNV <sub>SS</sub>	0		0.2V <sub>cc</sub>	V	
I <sub>OH (peak)</sub>	HIGH peak output current	P00 to P07, P10 to P17, P30 to P35, P40 to P45, P50 to P54, P60 to P67, P70, P71			- 10.0	mA	
I <sub>OL (peak)</sub>	LOW peak output current	P00 to P07, P30 to P35, P40 to P45, P50 to P54, P60 to P67, P70, P71			10.0	mA	
I <sub>OL (peak)</sub>	LOW peak output current	P10 to P17			30.0 10.0	mA	
						HIGHPOWER LOWPOWER	
I <sub>OH (avg)</sub>	HIGH average output current	P00 to P07, P10 to P17, P30 to P35, P40 to P45, P50 to P54, P60 to P67, P70, P71			- 5.0	mA	
I <sub>OL (avg)</sub>	LOW average output current	P00 to P07, P30 to P35, P40 to P45, P50 to P54, P60 to P67, P70, P71			5.0	mA	
I <sub>OL (avg)</sub>	LOW average output current	P10 to P17			15.0 5.0	mA	
						HIGHPOWER LOWPOWER	
f (X <sub>IN</sub> )	Main clock input oscillation frequency	Without wait	Mask ROM version	V <sub>cc</sub> =4.0V to 5.5V	0	10	MHz
				V <sub>cc</sub> =2.7V to 4.0V	0	5 x V <sub>cc</sub> - 10.000	MHz
			Flash memory version	V <sub>cc</sub> =4.0V to 5.5V	0	10	MHz
		With wait	Mask ROM version	V <sub>cc</sub> =4.0V to 5.5V	0	10	MHz
				V <sub>cc</sub> =2.7V to 4.0V	0	2.31 x V <sub>cc</sub> + 0.760	MHz
			Flash memory version	V <sub>cc</sub> =4.0V to 5.5V	0	10	MHz
f (X <sub>CIN</sub> )	Subclock oscillation frequency			32.768	50	kHz	

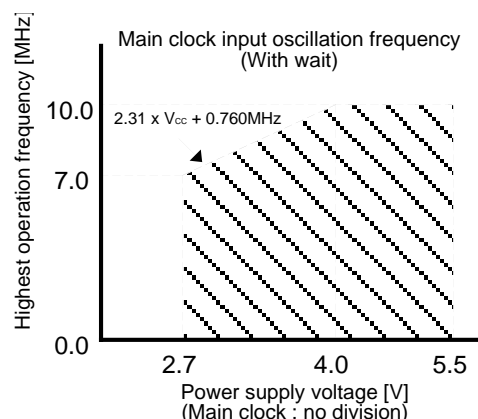
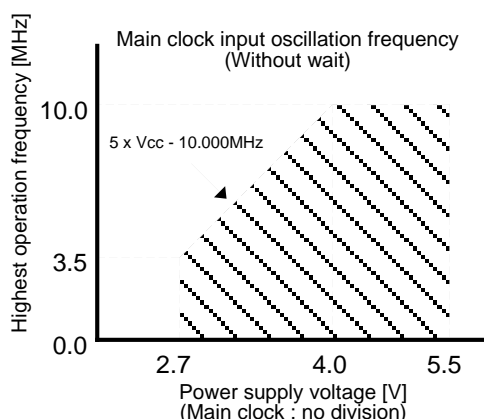
Note 1: Unless otherwise noted: V<sub>cc</sub> = 2.7V to 5.5V, V<sub>ss</sub> = 0V, T<sub>a</sub> = - 20 to 85°C (Extended operating temperature version:- 40 to 85°C). Flash version: V<sub>cc</sub> = 4.0V to 5.5V, V<sub>ss</sub> = 0V, T<sub>a</sub> = - 20 to 85°C (Extended operating temperature version:- 40 to 85°C.)

Note 2: Flash version: V<sub>cc</sub> = 4.0V to 5.5V

Note 3: The average output current is an average value measured over 100ms.

Note 4: Keep output current as follows:

The sum of port P3 and P4 I<sub>OL</sub> (peak) is under 40 mA. The sum of port P1 I<sub>OL</sub> (peak) is under 60 mA. The sum of port P1, P3 and P4 I<sub>OH</sub> (peak) is under 40 mA. The sum of port P0, P5, P6 and P7 I<sub>OL</sub> (peak) is under 80 mA. The sum of port P0, P5, P6 and P7 I<sub>OH</sub> (peak) is under 80 mA.





Electrical characteristics (V<sub>CC</sub> = 5V)V<sub>CC</sub> = 5V

Table 1.39. Electrical characteristics (Note1)

Symbol	Parameter		Measuring condition	Standard			Unit	
				Min.	Typ.	Max.		
V <sub>OH</sub>	HIGH output voltage	P0 <sub>0</sub> to P0 <sub>7</sub> , P1 <sub>0</sub> to P1 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>5</sub> , P4 <sub>0</sub> to P4 <sub>5</sub> , P5 <sub>0</sub> to P5 <sub>4</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub>	I <sub>OH</sub> = - 5 mA	3.0			V	
V <sub>OH</sub>	HIGH output voltage	P0 <sub>0</sub> to P0 <sub>7</sub> , P1 <sub>0</sub> to P1 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>5</sub> , P4 <sub>0</sub> to P4 <sub>5</sub> , P5 <sub>0</sub> to P5 <sub>4</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub>	I <sub>OH</sub> = - 200 μA	4.7			V	
V <sub>OH</sub>	HIGH output voltage	X <sub>OUT</sub>	HIGHPOWER	I <sub>OH</sub> = - 1 mA	3.0		V	
			LOWPOWER	I <sub>OH</sub> = - 0.5 mA	3.0		V	
V <sub>OH</sub>	HIGH output voltage	X <sub>COU</sub> T	HIGHPOWER	No load		3.0	V	
			LOWPOWER	No load		1.6	V	
V <sub>OL</sub>	LOW output voltage	P0 <sub>0</sub> to P0 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>5</sub> , P4 <sub>0</sub> to P4 <sub>5</sub> P5 <sub>0</sub> to P5 <sub>4</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub>	I <sub>OL</sub> = 5 mA			2.0	V	
V <sub>OL</sub>	LOW output voltage	P0 <sub>0</sub> to P0 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>5</sub> , P4 <sub>0</sub> to P4 <sub>5</sub> P5 <sub>0</sub> to P5 <sub>4</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub>	I <sub>OL</sub> = 200 μA			0.45	V	
V <sub>OL</sub>	LOW output voltage	P1 <sub>0</sub> to P1 <sub>7</sub>	HIGHPOWER	I <sub>OL</sub> = 15mA		2.0	V	
			LOWPOWER	I <sub>OL</sub> = 5 mA		2.0	V	
V <sub>OL</sub>	LOW output voltage	P1 <sub>0</sub> to P1 <sub>7</sub>	HIGHPOWER	I <sub>OL</sub> = 200 μA		0.3	V	
			LOWPOWER	I <sub>OL</sub> = 200 μA		0.45	V	
V <sub>OL</sub>	LOW output voltage	X <sub>OUT</sub>	HIGHPOWER	I <sub>OH</sub> = 1 mA		2.0	V	
			LOWPOWER	I <sub>OH</sub> = 0.5 mA		2.0	V	
V <sub>OL</sub>	LOW output voltage	X <sub>OUT</sub>	HIGHPOWER	No load		0	V	
			LOWPOWER	No load		0	V	
V <sub>T+</sub> -V <sub>T-</sub>	Hysteresis	TA0 <sub>IN</sub> , TX0 <sub>INOUT</sub> , TX1 <sub>INOUT</sub> , TX2 <sub>INOUT</sub> TB0 <sub>IN</sub> , TB1 <sub>IN</sub> INT <sub>0</sub> , INT <sub>1</sub> , CLK <sub>0</sub> , K <sub>I0</sub> to K <sub>I7</sub> RxD <sub>0</sub> , RxD <sub>1</sub>		0.2		0.8	V	
V <sub>T+</sub> -V <sub>T-</sub>	Hysteresis	RESET		0.2		1.8	V	
I <sub>IH</sub>	HIGH input current	P0 <sub>0</sub> to P0 <sub>7</sub> , P1 <sub>0</sub> to P1 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>5</sub> , P4 <sub>0</sub> to P4 <sub>5</sub> , P5 <sub>0</sub> to P5 <sub>4</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub> , RESET, CNV <sub>SS</sub>	V <sub>I</sub> = 5V			5.0	μA	
I <sub>IL</sub>	LOW input current	P0 <sub>0</sub> to P0 <sub>7</sub> , P1 <sub>0</sub> to P1 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>5</sub> , P4 <sub>0</sub> to P4 <sub>5</sub> , P5 <sub>0</sub> to P5 <sub>4</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub> , RESET, CNV <sub>SS</sub>	V <sub>I</sub> = 0V			-5.0	μA	
R <sub>PULLUP</sub>	Pull-up resistor	P0 <sub>0</sub> to P0 <sub>7</sub> , P1 <sub>0</sub> to P1 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>5</sub> , P4 <sub>0</sub> to P4 <sub>5</sub> , P5 <sub>0</sub> to P5 <sub>4</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub>	V <sub>I</sub> = 0V	30.0	50.0	167.0	kΩ	
R <sub>XIN</sub>	Feedback resistor	X <sub>IN</sub>			1.0		MΩ	
R <sub>XCIN</sub>	Feedback resistor	X <sub>CIN</sub>			6.0		MΩ	
V <sub>RAM</sub>	RAM retention voltage		When clock is stopped	2.0			V	
I <sub>CC</sub>	Power supply current		I/O pin has no load	f(X <sub>IN</sub> )=10MHz Square wave, no division		19.0	38.0	mA
				f(X <sub>CIN</sub> )=32kHz Square wave		90.0		μA
				f(X <sub>CIN</sub> )=32kHz With wait(Note2)		4.0		μA
				Ta=25°C when clock is stopped			1.0	μA
				Ta=85°C when clock is stopped			20.0	μA

Note 1: Unless otherwise noted: V<sub>CC</sub> = 5V, V<sub>SS</sub> = 0V at Ta = 25°C, f(X<sub>IN</sub>) = 10MHz)

Note 2: With one timer operated using fc32.

Electrical characteristics ( $V_{CC} = 5V$ ) $V_{CC} = 5V$ 

Table 1.40. A-D conversion characteristics

Symbol	Parameter		Measuring condition	Standard			Unit
				Min.	Typ.	Max.	
–	Resolution		$V_{REF} = V_{CC}$			10	Bits
–	Absolute accuracy	Sample & hold function not available	$V_{REF} = V_{CC} = 5V$			$\pm 3$	LSB
		Sample & hold function available(10bit)	$V_{REF} = V_{CC} = 5V$			$\pm 3$	LSB
		Sample & hold function available(8bit)	$V_{REF} = V_{CC} = 5V$			$\pm 2$	LSB
$R_{LADDER}$	Ladder resistance		$V_{REF} = V_{CC}$	10		40	kohm
$t_{CONV}$	Conversion time(10bit)			3.3			$\mu s$
$t_{CONV}$	Conversion time(8bit)			2.8			$\mu s$
$t_{SAMP}$	Sampling time			0.3			$\mu s$
$V_{REF}$	Reference voltage			2		$V_{CC}$	V
$V_{IA}$	Analog input voltage			0		$V_{REF}$	V

Electrical characteristics ( $V_{CC} = 5V$ ) $V_{CC} = 5V$ Timing requirements (referenced to  $V_{CC} = 5V$ ,  $V_{SS} = 0V$  at  $T_a = 25^\circ C$  unless otherwise specified)**Table 1.41. External clock input**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_c$	External clock input cycle time	100		ns
$t_w(H)$	External clock input HIGH pulse width	40		ns
$t_w(L)$	External clock input LOW pulse width	40		ns
$t_r$	External clock rise time		15	ns
$t_f$	External clock fall time		15	ns

**Table 1.42. Timer A input (counter input in event counter mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_c(TA)$	TA0IN input cycle time	100		ns
$t_w(TAH)$	TA0IN input HIGH pulse width	40		ns
$t_w(TAL)$	TA0IN input LOW pulse width	40		ns

**Table 1.43. Timer A input (gating input in timer mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_c(TA)$	TA0IN input cycle time	400		ns
$t_w(TAH)$	TA0IN input HIGH pulse width	200		ns
$t_w(TAL)$	TA0IN input LOW pulse width	200		ns

**Table 1.44. Timer A input (external trigger input in one-shot timer mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_c(TA)$	TA0IN input cycle time	200		ns
$t_w(TAH)$	TA0IN input HIGH pulse width	100		ns
$t_w(TAL)$	TA0IN input LOW pulse width	100		ns

**Table 1.45. Timer A input (external trigger input in pulse width modulation mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_w(TAH)$	TA0IN input HIGH pulse width	100		ns
$t_w(TAL)$	TA0IN input LOW pulse width	100		ns

**Table 1.46. Timer A input (up/down input in event counter mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_c(UP)$	TA0OUT input cycle time	2000		ns
$t_w(UPH)$	TA0OUT input HIGH pulse width	1000		ns
$t_w(UPL)$	TA0OUT input LOW pulse width	1000		ns
$t_{su}(UP-TIN)$	TA0OUT input setup time	400		ns
$t_h(TIN-UP)$	TA0OUT input hold time	400		ns

Electrical characteristics ( $V_{CC} = 5V$ ) $V_{CC} = 5V$ Timing requirements (referenced to  $V_{CC} = 5V$ ,  $V_{SS} = 0V$  at  $T_a = 25^\circ C$  unless otherwise specified)**Table 1.47. Timer B input (counter input in event counter mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TB)}$	TBiIN input cycle time (counted on one edge)	100		ns
$t_{w(TBH)}$	TBiIN input HIGH pulse width (counted on one edge)	40		ns
$t_{w(TBL)}$	TBiIN input LOW pulse width (counted on one edge)	40		ns
$t_{c(TB)}$	TBiIN input cycle time (counted on both edges)	200		ns
$t_{w(TBH)}$	TBiIN input HIGH pulse width (counted on both edges)	80		ns
$t_{w(TBL)}$	TBiIN input LOW pulse width (counted on both edges)	80		ns

**Table 1.48. Timer B input (pulse period measurement mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TB)}$	TBiIN input cycle time	400		ns
$t_{w(TBH)}$	TBiIN input HIGH pulse width	200		ns
$t_{w(TBL)}$	TBiIN input LOW pulse width	200		ns

**Table 1.49. Timer B input (pulse width measurement mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TB)}$	TBiIN input cycle time	400		ns
$t_{w(TBH)}$	TBiIN input HIGH pulse width	200		ns
$t_{w(TBL)}$	TBiIN input LOW pulse width	200		ns

**Table 1.50. Timer X input (counter input in event counter mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TX)}$	TXiINOUT input cycle time	100		ns
$t_{w(TXH)}$	TXiINOUT input HIGH pulse width	40		ns
$t_{w(TXL)}$	TXiINOUT input LOW pulse width	40		ns

**Table 1.51. Timer X input (gate input in timer mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TX)}$	TXiINOUT input cycle time	400		ns
$t_{w(TXH)}$	TXiINOUT input HIGH pulse width	200		ns
$t_{w(TXL)}$	TXiINOUT input LOW pulse width	200		ns

**Table 1.52. Timer X input (external trigger input in one-shot timer mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TX)}$	TXiINOUT input cycle time	200		ns
$t_{w(TXH)}$	TXiINOUT input HIGH pulse width	100		ns
$t_{w(TXL)}$	TXiINOUT input LOW pulse width	100		ns

Electrical characteristics ( $V_{CC} = 5V$ ) $V_{CC} = 5V$ Timing requirements (referenced to  $V_{CC} = 5V$ ,  $V_{SS} = 0V$  at  $T_a = 25^\circ C$  unless otherwise specified)

Table 1.53. Timer X input (pulse period measurement mode)

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_c(TX)$	TXiINOUT input cycle time	400		ns
$t_w(TXH)$	TXiINOUT input HIGH pulse width	200		ns
$t_w(TXL)$	TXiINOUT input LOW pulse width	200		ns

Table 1.54. Timer X input (pulse width measurement mode)

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_c(TX)$	TXiINOUT input cycle time	400		ns
$t_w(TXH)$	TXiINOUT input HIGH pulse width	200		ns
$t_w(TXL)$	TXiINOUT input LOW pulse width	200		ns

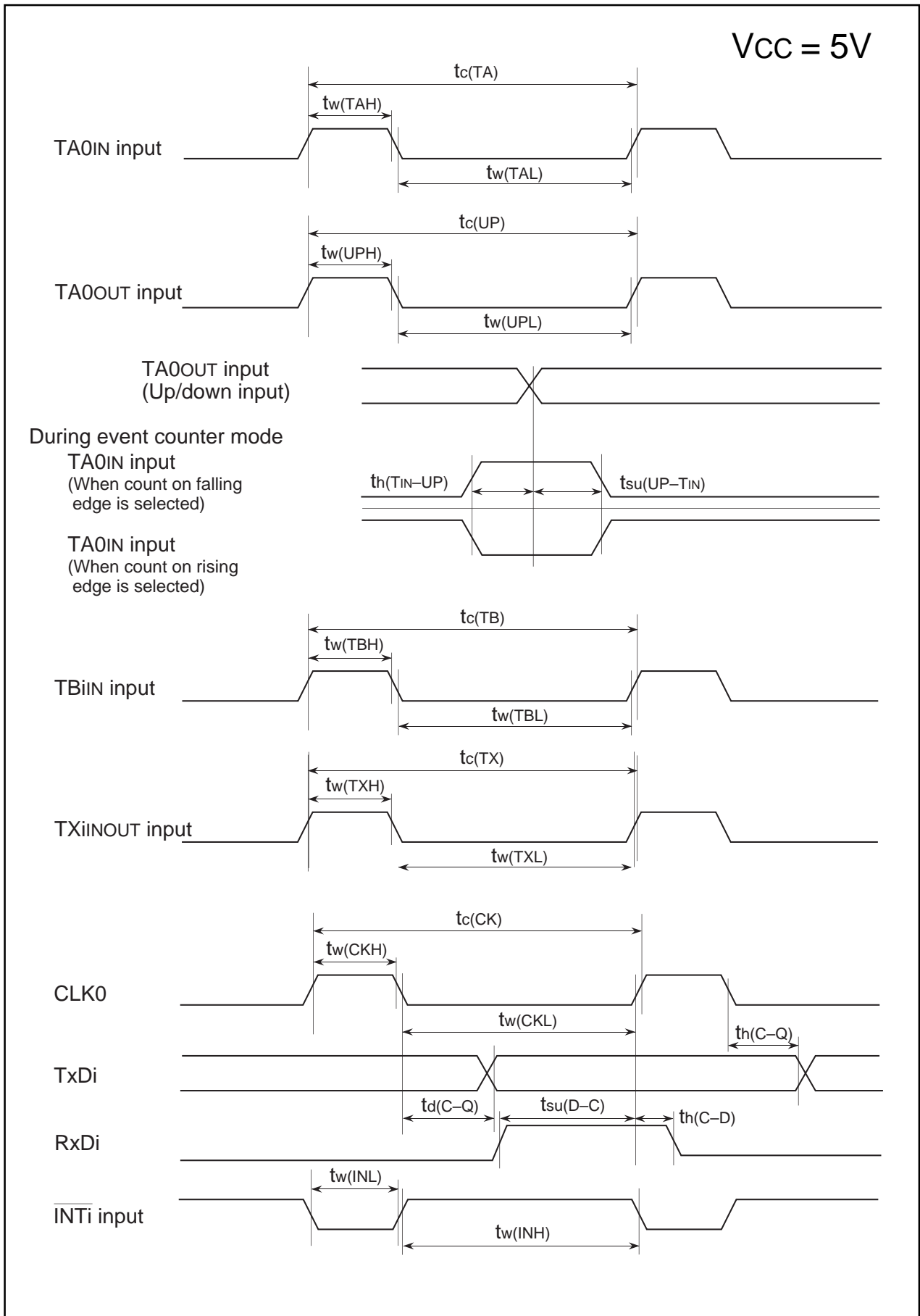
Table 1.55. Serial I/O

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_c(CK)$	CLK0 input cycle time	200		ns
$t_w(CKH)$	CLK0 input HIGH pulse width	100		ns
$t_w(CKL)$	CLK0 input LOW pulse width	100		ns
$t_d(C-Q)$	TxDi output delay time		80	ns
$t_h(C-Q)$	TxDi hold time	0		ns
$t_{su}(D-C)$	RxDi input setup time	30		ns
$t_h(C-D)$	RxDi input hold time	90		ns

Table 1.56. External interrupt  $\overline{INT}_i$  inputs

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_w(INH)$	$\overline{INT}_i$ input HIGH pulse width	250		ns
$t_w(INL)$	$\overline{INT}_i$ input LOW pulse width	250		ns

Electrical characteristics ( $V_{CC} = 5V$ )



Electrical characteristics (V<sub>CC</sub> = 3V)V<sub>CC</sub> = 3V

Table 1.57. Electrical characteristics (Note 1)

Symbol	Parameter	Measuring condition	Standard			Unit	
			Min.	Typ.	Max.		
V <sub>OH</sub>	HIGH output voltage P0 <sub>0</sub> to P0 <sub>7</sub> , P1 <sub>0</sub> to P1 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>5</sub> , P4 <sub>0</sub> to P4 <sub>5</sub> , P5 <sub>0</sub> to P5 <sub>4</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub>	I <sub>OH</sub> = - 1 mA	2.5			V	
V <sub>OH</sub>	HIGH output voltage X <sub>OUT</sub>	HIGHPOWER I <sub>OH</sub> = - 1 mA	2.5			V	
		LOWPOWER I <sub>OH</sub> = - 50 μA	2.5				
V <sub>OH</sub>	HIGH output voltage X <sub>COUT</sub>	HIGHPOWER No load		3.0		V	
		LOWPOWER No load		1.6			
V <sub>OL</sub>	LOW output voltage P0 <sub>0</sub> to P0 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>5</sub> , P4 <sub>0</sub> to P4 <sub>5</sub> , P5 <sub>0</sub> to P5 <sub>4</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub>	I <sub>OL</sub> = 1 mA			0.5	V	
V <sub>OL</sub>	LOW output voltage P1 <sub>0</sub> to P1 <sub>7</sub>	HIGHPOWER I <sub>OL</sub> = 3 mA			0.5	V	
		LOWPOWER I <sub>OL</sub> = 1 mA			0.5		
V <sub>OL</sub>	LOW output voltage X <sub>OUT</sub>	HIGHPOWER I <sub>OH</sub> = 0.1 mA			0.5	V	
		LOWPOWER I <sub>OH</sub> = 50 μA			0.5		
V <sub>OL</sub>	LOW output voltage X <sub>OUT</sub>	HIGHPOWER No load		0		V	
		LOWPOWER No load		0			
V <sub>T+</sub> -V <sub>T-</sub>	Hysteresis TA0 <sub>IN</sub> , TX0 <sub>INOUT</sub> , TX1 <sub>INOUT</sub> , TX2 <sub>INOUT</sub> , TB0 <sub>IN</sub> , TB1 <sub>IN</sub> INT <sub>0</sub> , INT <sub>1</sub> , CLK <sub>0</sub> , K <sub>l0</sub> to K <sub>l7</sub> , Rx <sub>D0</sub> , Rx <sub>D1</sub>		0.2		0.8	V	
V <sub>T+</sub> -V <sub>T-</sub>	Hysteresis RESET		0.2		1.8	V	
I <sub>IH</sub>	HIGH input current P0 <sub>0</sub> to P0 <sub>7</sub> , P1 <sub>0</sub> to P1 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>5</sub> , P4 <sub>0</sub> to P4 <sub>5</sub> , P5 <sub>0</sub> to P5 <sub>4</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub> , RESET, CNV <sub>SS</sub>	V <sub>I</sub> = 3V			4.0	μA	
I <sub>IL</sub>	LOW input current P0 <sub>0</sub> to P0 <sub>7</sub> , P1 <sub>0</sub> to P1 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>5</sub> , P4 <sub>0</sub> to P4 <sub>5</sub> , P5 <sub>0</sub> to P5 <sub>4</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub> , RESET, CNV <sub>SS</sub>	V <sub>I</sub> = 0V			-4.0	μA	
R <sub>PULLUP</sub>	Pull-up resistor P0 <sub>0</sub> to P0 <sub>7</sub> , P1 <sub>0</sub> to P1 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>5</sub> , P4 <sub>0</sub> to P4 <sub>5</sub> , P5 <sub>0</sub> to P5 <sub>4</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub>	V <sub>I</sub> = 0V	66.0	120.0	500.0	kΩ	
R <sub>XIN</sub>	Feedback resistor X <sub>IN</sub>			3.0		MΩ	
R <sub>XIN</sub>	Feedback resistor X <sub>IN</sub>			10.0		MΩ	
V <sub>RAM</sub>	RAM retention voltage	When clock is stopped	2.0			V	
I <sub>CC</sub>	Power supply current	I/O pin has no load	f(X <sub>IN</sub> )=7MHz Square wave, no division	6.0	15.0	mA	
			f(X <sub>CIN</sub> )=32kHz Square wave	40.0			μA
			f(X <sub>CIN</sub> )=32kHz With wait. Oscillation capacity HIGH (Note 2)	2.8		μA	
			f(X <sub>CIN</sub> )=32kHz With wait. Oscillation capacity LOW (Note 2)	0.9		μA	
			T <sub>a</sub> =25°C when clock is stopped			1.0	μA
			T <sub>a</sub> =85°C when clock is stopped			20.0	

Note 1: Unless otherwise noted: V<sub>CC</sub> = 3V, V<sub>SS</sub> = 0V at T<sub>a</sub> = 25°C, f(X<sub>IN</sub>) = 7MHz, with wait)

Note 2: With one timer operated using fc32.

Electrical characteristics ( $V_{CC} = 3V$ ) $V_{CC} = 3V$ 

Table 1.58. A-D conversion characteristics

Symbol	Parameter		Measuring condition	Standard			Unit
				Min.	Typ.	Max.	
–	Resolution		$V_{REF} = V_{CC}$			10	Bits
–	Absolute accuracy	Sample & hold function not available (8bit)	$V_{REF} = V_{CC} = 3V$ , $\emptyset_{AD} = f_{AD}/2$			$\pm 2$	LSB
$R_{LADDER}$	Ladder resistance		$V_{REF} = V_{CC}$	10		40	kohm
$t_{CONV}$	Conversion time(8bit)			14.0			$\mu s$
$V_{REF}$	Reference voltage			2.7		$V_{CC}$	V
$V_{IA}$	Analog input voltage			0		$V_{REF}$	V



Electrical characteristics ( $V_{CC} = 3V$ ) $V_{CC} = 3V$ Timing requirements (referenced to  $V_{CC} = 3V$ ,  $V_{SS} = 0V$  at  $T_a = 25^\circ C$  unless otherwise specified)**Table 1.59. External clock input**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_c$	External clock input cycle time	143		ns
$t_{w(H)}$	External clock input HIGH pulse width	60		ns
$t_{w(L)}$	External clock input LOW pulse width	60		ns
$t_r$	External clock rise time		18	ns
$t_f$	External clock fall time		18	ns

**Table 1.60. Timer A input (counter input in event counter mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TA)}$	TA0IN input cycle time	150		ns
$t_{w(TAH)}$	TA0IN input HIGH pulse width	60		ns
$t_{w(TAL)}$	TA0IN input LOW pulse width	60		ns

**Table 1.61. Timer A input (gating input in timer mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TA)}$	TA0IN input cycle time	600		ns
$t_{w(TAH)}$	TA0IN input HIGH pulse width	300		ns
$t_{w(TAL)}$	TA0IN input LOW pulse width	300		ns

**Table 1.62. Timer A input (external trigger input in one-shot timer mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TA)}$	TA0IN input cycle time	300		ns
$t_{w(TAH)}$	TA0IN input HIGH pulse width	150		ns
$t_{w(TAL)}$	TA0IN input LOW pulse width	150		ns

**Table 1.63. Timer A input (external trigger input in pulse width modulation mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{w(TAH)}$	TA0IN input HIGH pulse width	150		ns
$t_{w(TAL)}$	TA0IN input LOW pulse width	150		ns

**Table 1.64. Timer A input (up/down input in event counter mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(UP)}$	TA0OUT input cycle time	3000		ns
$t_{w(UPH)}$	TA0OUT input HIGH pulse width	1500		ns
$t_{w(UPL)}$	TA0OUT input LOW pulse width	1500		ns
$t_{su(UP-TIN)}$	TA0OUT input setup time	600		ns
$t_{h(TIN-UP)}$	TA0OUT input hold time	600		ns

Electrical characteristics ( $V_{CC} = 3V$ ) $V_{CC} = 3V$ Timing requirements (referenced to  $V_{CC} = 3V$ ,  $V_{SS} = 0V$  at  $T_a = 25^\circ C$  unless otherwise specified)**Table 1.65. Timer B input (counter input in event counter mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TB)}$	TBiIN input cycle time (counted on one edge)	150		ns
$t_w(TBH)$	TBiIN input HIGH pulse width (counted on one edge)	60		ns
$t_w(TBL)$	TBiIN input LOW pulse width (counted on one edge)	60		ns
$t_{c(TB)}$	TBiIN input cycle time (counted on both edges)	300		ns
$t_w(TBH)$	TBiIN input HIGH pulse width (counted on both edges)	160		ns
$t_w(TBL)$	TBiIN input LOW pulse width (counted on both edges)	160		ns

**Table 1.66. Timer B input (pulse period measurement mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TB)}$	TBiIN input cycle time	600		ns
$t_w(TBH)$	TBiIN input HIGH pulse width	300		ns
$t_w(TBL)$	TBiIN input LOW pulse width	300		ns

**Table 1.67. Timer B input (pulse width measurement mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TB)}$	TBiIN input cycle time	600		ns
$t_w(TBH)$	TBiIN input HIGH pulse width	300		ns
$t_w(TBL)$	TBiIN input LOW pulse width	300		ns

**Table 1.68. Timer X input (counter input in event counter mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TX)}$	TXiINOUT input cycle time	150		ns
$t_w(TXH)$	TXiINOUT input HIGH pulse width	60		ns
$t_w(TXL)$	TXiINOUT input LOW pulse width	60		ns

**Table 1.69. Timer X input (gate input in timer mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TX)}$	TXiINOUT input cycle time	600		ns
$t_w(TXH)$	TXiINOUT input HIGH pulse width	300		ns
$t_w(TXL)$	TXiINOUT input LOW pulse width	300		ns

**Table 1.70. Timer X input (external trigger input in one-shot timer mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TX)}$	TXiINOUT input cycle time	300		ns
$t_w(TXH)$	TXiINOUT input HIGH pulse width	150		ns
$t_w(TXL)$	TXiINOUT input LOW pulse width	150		ns

Electrical characteristics ( $V_{CC} = 3V$ ) $V_{CC} = 3V$ Timing requirements (referenced to  $V_{CC} = 3V$ ,  $V_{SS} = 0V$  at  $T_a = 25^\circ C$  unless otherwise specified)

Table 1.71. Timer X input (pulse period measurement mode)

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TX)}$	TXiINOUT input cycle time	600		ns
$t_{w(TXH)}$	TXiINOUT input HIGH pulse width	300		ns
$t_{w(TXL)}$	TXiINOUT input LOW pulse width	300		ns

Table 1.72. Timer X input (pulse width measurement mode)

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TX)}$	TXiINOUT input cycle time	600		ns
$t_{w(TXH)}$	TXiINOUT input HIGH pulse width	300		ns
$t_{w(TXL)}$	TXiINOUT input LOW pulse width	300		ns

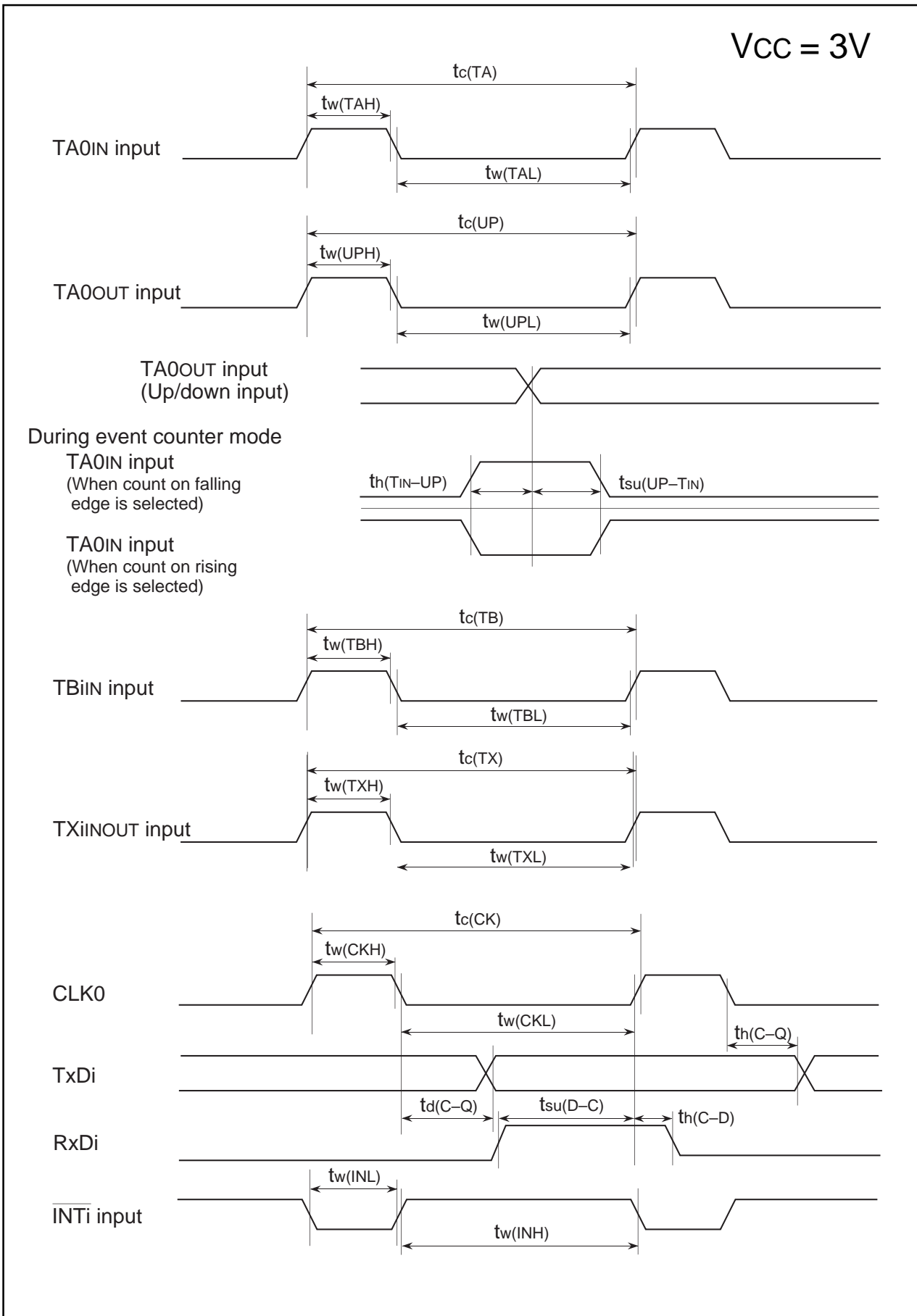
Table 1.73. Serial I/O

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(CK)}$	CLK0 input cycle time	300		ns
$t_{w(CKH)}$	CLK0 input HIGH pulse width	150		ns
$t_{w(CKL)}$	CLK0 input LOW pulse width	150		ns
$t_d(C-Q)$	TxDi output delay time		160	ns
$t_h(C-Q)$	TxDi hold time	0		ns
$t_{su}(D-C)$	RxDi input setup time	50		ns
$t_h(C-D)$	RxDi input hold time	90		ns

Table 1.74. External interrupt  $\overline{INT}_i$  inputs

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{w(INH)}$	$\overline{INT}_i$ input HIGH pulse width	380		ns
$t_{w(INL)}$	$\overline{INT}_i$ input LOW pulse width	380		ns

Electrical characteristics ( $V_{CC} = 3V$ )



## Description

**Outline Performance**

Table AA-1 shows the outline performance of the M30201 (flash memory version).

**Table AA-1. Outline Performance of the M30201 (flash memory version)**

Item		Performance
Power supply voltage		4.0V to 5.5 V (f(XIN)=10MHz)
Program/erase voltage		VPP=12V ± 5% (f(XIN)=10MHz)
		VCC=5V ± 5% (f(XIN)=10MHz)
Flash memory operation mode		Three modes (parallel I/O, standard serial I/O, CPU rewrite)
Erase block division	User ROM area	See Figure 1.AA.3.
	Boot ROM area	One division (4 Kbytes) (Note 1)
Program method		In units of byte
Erase method		Collective erase
Program/erase control method		Program/erase control by software command
Number of commands		6 commands
Program/erase count		100 times
ROM code protect		Parallel I/O mode is supported.

Note: The boot ROM area contains a standard serial I/O mode control program which is stored in it when shipped from the factory. This area can be erased and programmed in only parallel I/O mode.

Description

**Flash Memory**

The M30201 (flash memory version) contains the NOR type of flash memory that requires a high-voltage VPP power supply for program/erase operations, in addition to the VCC power supply for device operation. For this flash memory, three flash memory modes are available in which to read, program, and erase: parallel I/O and standard serial I/O modes in which the flash memory can be manipulated using a programmer and a CPU rewrite mode in which the flash memory can be manipulated by the Central Processing Unit (CPU). Each mode is detailed in the pages to follow.

In addition to the ordinary user ROM area to store a microcomputer operation control program, the flash memory has a boot ROM area that is used to store a program to control rewriting in CPU rewrite and standard serial I/O modes. This boot ROM area has had a standard serial I/O mode control program stored in it when shipped from the factory. However, the user can write a rewrite control program in this area that suits the user's application system. This boot ROM area can be rewritten in only parallel I/O mode.

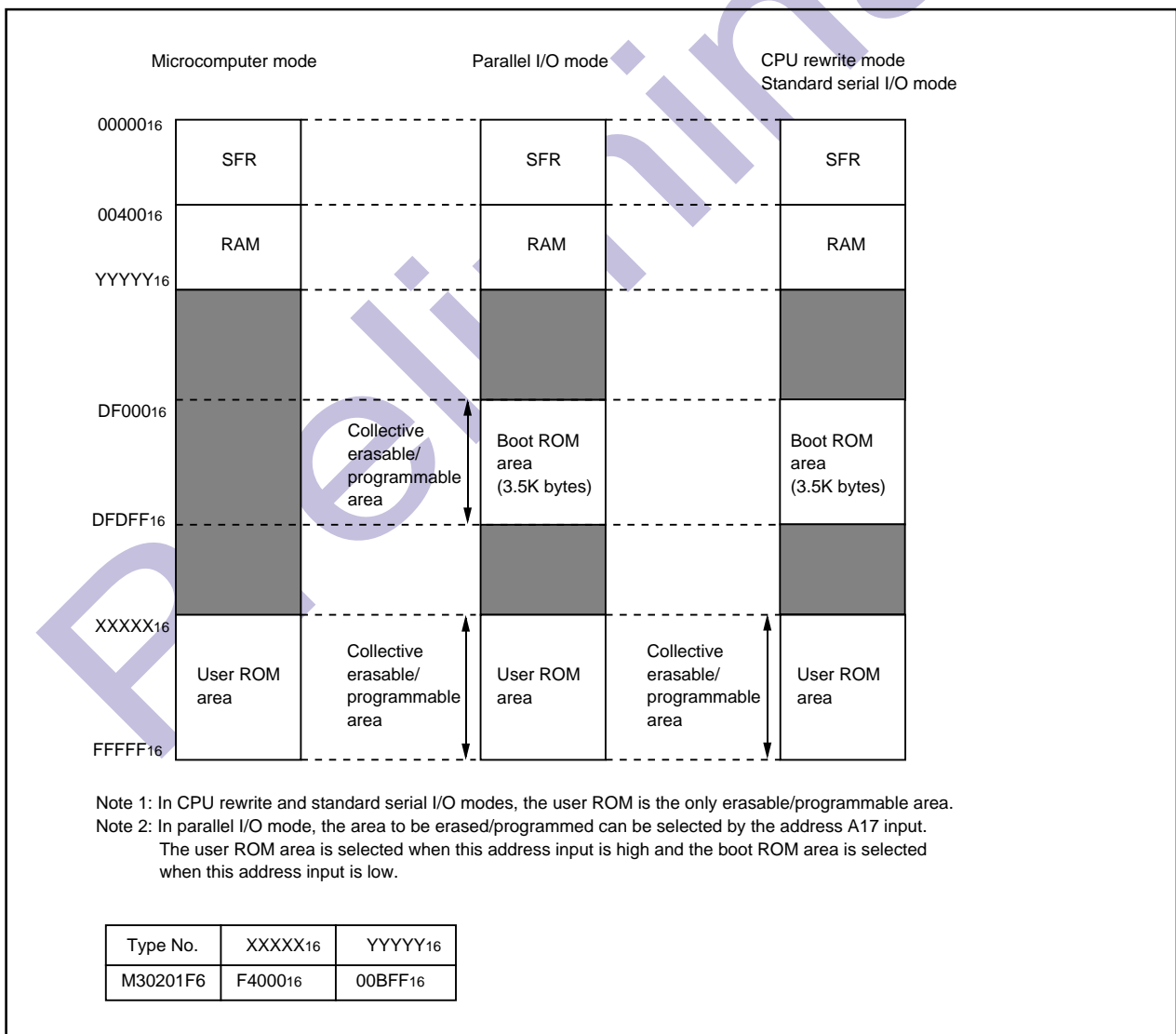


Figure AA-3. Block diagram of flash memory version

## CPU Rewrite Mode

### CPU Rewrite Mode

In CPU rewrite mode, the on-chip flash memory can be operated on (read, program, or erase) under control of the Central Processing Unit (CPU). In CPU rewrite mode, the flash memory can be operated on by reading or writing to the flash memory control register and flash command register. Figure BB-1, Figure BB-2 show the flash memory control register, and flash command register respectively.

Also, in CPU rewrite mode, the CNVSS pin is used as the VPP power supply pin. Apply the power supply voltage, VPPH, from an external source to this pin.

In CPU rewrite mode, only the user ROM area shown in Figure AA-3 can be rewritten; the boot ROM area cannot be rewritten. Make sure the program and block commands are issued for only the user ROM area. The control program for CPU rewrite mode can be stored in either user ROM or boot ROM area. In the CPU rewrite mode, because the flash memory cannot be read from the CPU, the rewrite control program must be transferred to internal RAM before it can be executed.

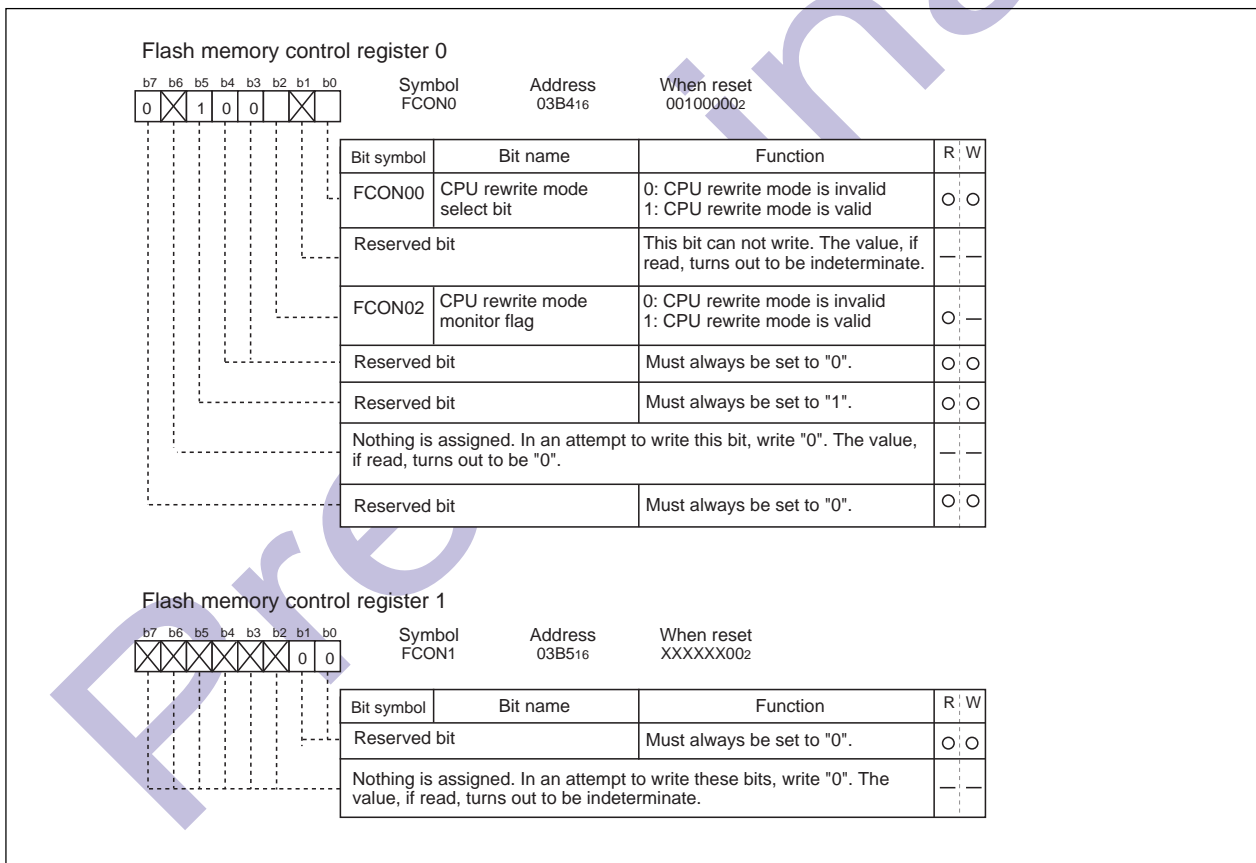


Figure BB-1. Flash memory control register

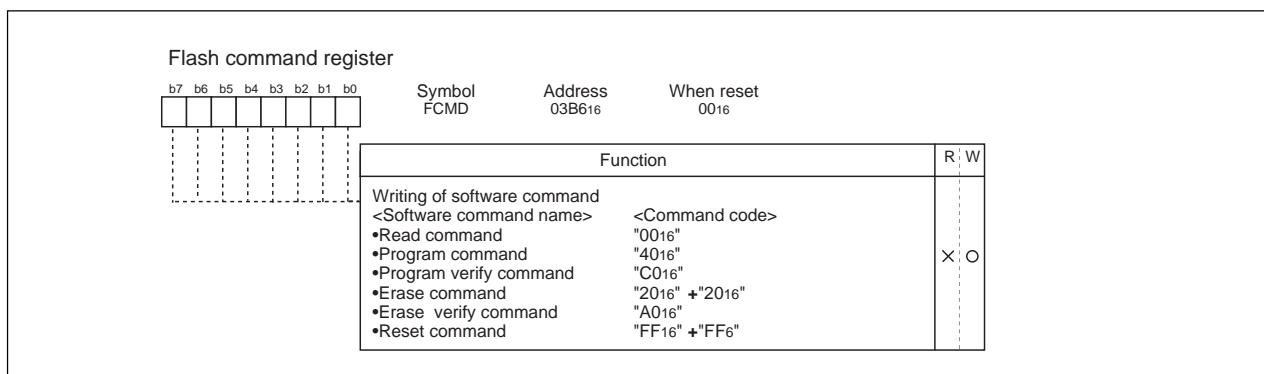


Figure BB-2. Flash command register

## Microcomputer Mode and Boot Mode

The control program for CPU rewrite mode must be written into the user ROM or boot ROM area in parallel I/O mode beforehand. (If the control program is written into the boot ROM area, the standard serial I/O mode becomes unusable.)

See Figure AA-3 for details about the boot ROM area.

Normal microcomputer mode is entered when the microcomputer is reset with pulling CNVSS pin low (Vss). In this case, the CPU starts operating using the control program in the user ROM area.

When the microcomputer is reset by pulling the P52 pin high (VCC), the CNVSS pin high (VPPH), the CPU starts operating using the control program in the boot ROM area. This mode is called the "boot" mode.

The control program in the boot ROM area can also be used to rewrite the user ROM area.

## CPU rewrite mode operation procedure

The internal flash memory can be operated on to program, read, verify, or erase it while being placed on-board by writing commands from the CPU to the flash memory control register (addresses 03B416, 03B516) and flash command register (address 03B616). Note that when in CPU rewrite mode, the boot ROM area cannot be accessed for program, read, verify, or erase operations. Before this can be accomplished, a CPU write control program must be written into the boot ROM area in parallel input/output mode. The following shows a CPU rewrite mode operation procedure.

### <Start procedure (Note 1)>

- (1) Apply VPPH to the CNVSS/VPP pin and VCC to the port P52 pin for reset release. Or the user can jump from the user ROM area to the boot ROM area using the JMP instruction and execute the CPU write control program. In this case, set the CPU write mode select bit of the flash memory control register to "1" before applying VPPH to the CNVSS/VPP pin.
- (2) After transferring the CPU write control program from the boot ROM area to the internal RAM, jump to this control program in RAM. (The operations described below are controlled by this program.)
- (3) Set the CPU rewrite mode select bit to "1".
- (4) Read the CPU rewrite mode monitor flag to see that the CPU rewrite mode is enabled.
- (5) Execute operation on the flash memory by writing software commands to the flash command register.

Note 1: In addition to the above, various other operations need to be performed, such as for entering the data to be written to flash memory from an external source (e.g., serial I/O), initializing the ports, and writing to the watchdog timer.

### <Clearing procedure>

- (1) Apply VSS to the CNVSS/VPP pin.
- (2) Set the CPU rewrite mode select bit to "0".



## CPU Rewrite Mode

---

### Precautions on CPU Rewrite Mode

Described below are the precautions to be observed when rewriting the flash memory in CPU rewrite mode.

#### (1) Operation speed

During erase/program mode, set BCLK to one of the following frequencies by changing the divide ratio:

5 MHz or less when wait bit (bit 7 at address 0005<sub>16</sub>) = 0 (without internal access wait state)

10 MHz or less when wait bit (bit 7 at address 0005<sub>16</sub>) = 1 (with internal access wait state)

#### (2) Instructions inhibited against use

The instructions listed below cannot be used during CPU rewrite mode because they refer to the internal data of the flash memory:

UND instruction, INTO instruction, JMPS instruction, JSRS instruction, and BRK instruction

#### (3) Interrupts inhibited against use

No interrupts can be used that look up the fixed vector table in the flash memory area. Maskable interrupts may be used by setting the interrupt vector table in a location outside the flash memory area.

## Software Commands

Table BB-1 lists the software commands available with the M30201 (flash memory version).

When CPU rewrite mode is enabled, write software commands to the flash command register to specify the operation to erase or program.

The content of each software command is explained below.

**Table BB-1. List of Software Commands (CPU Rewrite Mode)**

Command	First bus cycle			Second bus cycle		
	Mode	Address	Data (D <sub>0</sub> to D <sub>7</sub> )	Mode	Address	Data (D <sub>0</sub> to D <sub>7</sub> )
Read	Write	03B6 <sub>16</sub>	00 <sub>16</sub>			
Program	Write	03B6 <sub>16</sub>	40 <sub>16</sub>	Write	Program address	Program data
Program verify	Write	03B6 <sub>16</sub>	C0 <sub>16</sub>	Read	Verify address	Verify data
Erase	Write	03B6 <sub>16</sub>	20 <sub>16</sub>	Write	03B6 <sub>16</sub>	20 <sub>16</sub>
Erase verify	Write	03B6 <sub>16</sub>	A0 <sub>16</sub>	Read	Verify address	Verify data
Reset	Write	03B6 <sub>16</sub>	FF <sub>16</sub>	Write	03B6 <sub>16</sub>	FF <sub>16</sub>

### Read Command (00<sub>16</sub>)

The read mode is entered by writing the command code "00<sub>16</sub>" to the flash command register in the first bus cycle. When an address to be read is input in one of the bus cycles that follow, the content of the specified address is read out at the data bus (D<sub>0</sub>–D<sub>7</sub>), 8 bits at a time.

The read mode is retained intact until another command is written.

After reset and after the reset command is executed, the read mode is set.

### Program Command (40<sub>16</sub>)

The program mode is entered by writing the command code "40<sub>16</sub>" to the flash command register in the first bus cycle. When the user execute an instruction to write byte data to the desired address (e.g., STE instruction) in the second bus cycle, the flash memory control circuit executes the program operation. The program operation requires approximately 20 μs. Wait for 20 μs or more before the user go to the next processing.

During program operation, the watchdog timer remains idle, with the value "7FFF<sub>16</sub>" set in it.

Note 1: The write operation is not completed immediately by writing a program command once. The user must always execute a program-verify command after each program command executed. And if verification fails, the user need to execute the program command repeatedly until the verification passes. See Figure 1.BB.3 for an example of a programming flowchart.

## CPU Rewrite Mode

---

### Program-verify command (C0<sub>16</sub>)

The program-verify mode is entered by writing the command code "C0<sub>16</sub>" to the flash command register in the first bus cycle. When the user execute an instruction (e.g., LDE instruction) to read byte data from the address to be verified (the previously programmed address) in the second bus cycle, the content that has actually been written to the address is read out from the memory.

The CPU compares this read data with the data that it previously wrote to the address using the program command. If the compared data do not match, the user need to execute the program and program-verify operations one more time.

### Erase command (20<sub>16</sub> + 20<sub>16</sub>)

The flash memory control circuit executes an erase operation by writing command code "20<sub>16</sub>" to the flash command register in the first bus cycle and the same command code to the flash command register again in the second bus cycle. The erase operation requires approximately 20 ms. Wait for 20 ms or more before the user go to the next processing.

Before this erase command can be performed, all memory locations to be erased must have had data "00<sub>16</sub>" written to by using the program and program-verify commands. During erase operation, the watchdog timer remains idle, with the value "7FFF<sub>16</sub> set in it.

Note 1: The erase operation is not completed immediately by writing an erase command once. The user must always execute an erase-verify command after each erase command executed. And if verification fails, the user need to execute the erase command repeatedly until the verification passes. See Figure BB-3 for an example of an erase flowchart.

### Erase-verify command (A0<sub>16</sub>)

The erase-verify mode is entered by writing the command code "A0<sub>16</sub>" to the flash command register in the first bus cycle. When the user execute an instruction to read byte data from the address to be verified (e.g., LDE instruction) in the second bus cycle, the content of the address is read out.

The CPU must sequentially erase-verify memory contents one address at a time, over the entire area erased. If any address is encountered whose content is not "FF<sub>16</sub>" (not erased), the CPU must stop erase-verify at that point and execute erase and erase-verify operations one more time.

Note 1: If any unerased memory location is encountered during erase-verify operation, be sure to execute erase and erase-verify operations one more time. In this case, however, the user does not need to write data "00<sub>16</sub>" to memory before erasing.

**Reset command (FF16 + FF16)**

The reset command is used to stop the program command or the erase command in the middle of operation. After writing command code "4016" or "2016" twice to the flash command register, write command code "FF16" to the flash command register in the first bus cycle and the same command code to the flash command register again in the second bus cycle. The program command or erase command is disabled, with the flash memory placed in read mode.

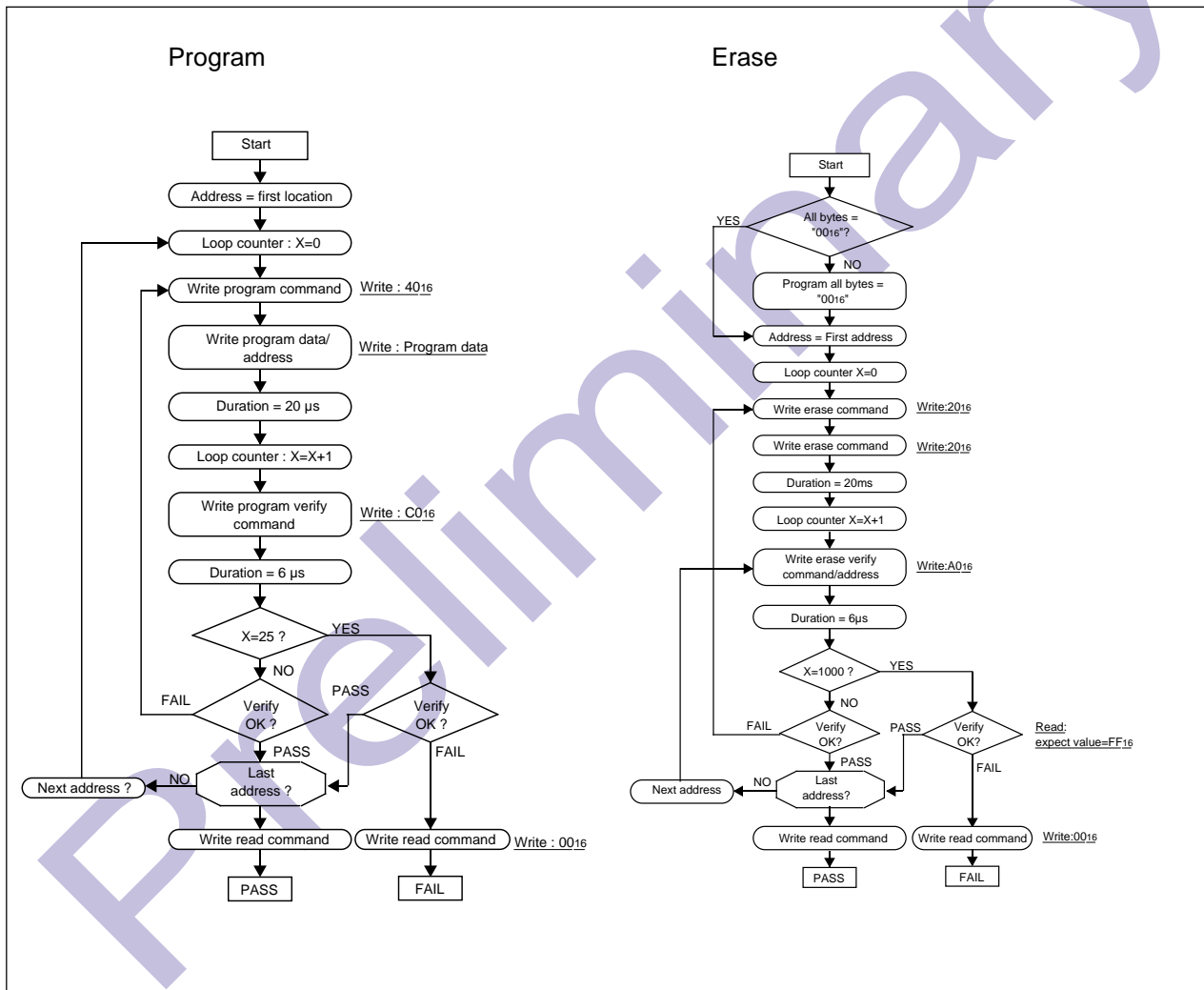


Figure BB-3. Program and erase execution flowchart in the CPU rewrite mode

## Appendix Parallel I/O Mode

## Description of Pin Function (Flash Memory Parallel I/O Mode)

Pin name	Signal name	I/O	Function
Vcc, Vss	Power supply input		Apply 5 V $\pm$ 10 % to the Vcc pin and 0 V to the Vss pin.
CNVss	CNVss	I	Apply 12 V $\pm$ 5 % to the CNVss pin.
RESET	Reset input	I	Connect this pin to Vss.
XIN	Clock input	I	Connect a ceramic or crystal resonator between the XIN and XOUT pins. When entering an externally derived clock, enter it from XIN and leave XOUT open.
XOUT	Clock output	O	
AVcc, AVss	Analog power supply input		Connect AVss to Vss and AVcc to Vcc, respectively.
VREF	Reference voltage input	I	Connect this pin to Vss.
P00 to P07	Data I/O D0 to D7	I/O	These are data D0–D7 input/output pins.
P10 to P17	Address input A8 to A15	I	These are address A8–A15 input pins.
P30 to P33	Address input A4 to A7	I	These are address A4–A7 input pins.
P34 to P35	Input port P3	I	Enter low signals to these pins.
P40	WE input	I	This is a WE input pin.
P41	OE input	I	This is a OE input pin.
P43	CE input	I	This is a CE input pin.
P42, P44, P45	Input port P4	I	Enter high signals or low signals to these pins.
P50	Address input A17	I	This is address A17 input pin.
P51	VRFY input	I	Apply VIH (5 V) to this pin when VPP = VPPH (12 V), or VIL (0 V) when VPP = VPPL (5 V).
P52	Input port P5	I	Enter low signal to this pin.
P53, P54	Input port P5	I	Enter high signals or low signals to these pins.
P60 to P63	Address input A0 to A3	I	These are address A0–A3 input pins.
P64 to P67	Input port P6	I	Enter high signals or low signals to these pins.
P70 to P71	Input port P7	I	Enter high signals or low signals to these pins.

## Appendix Parallel I/O Mode

### Parallel I/O Mode

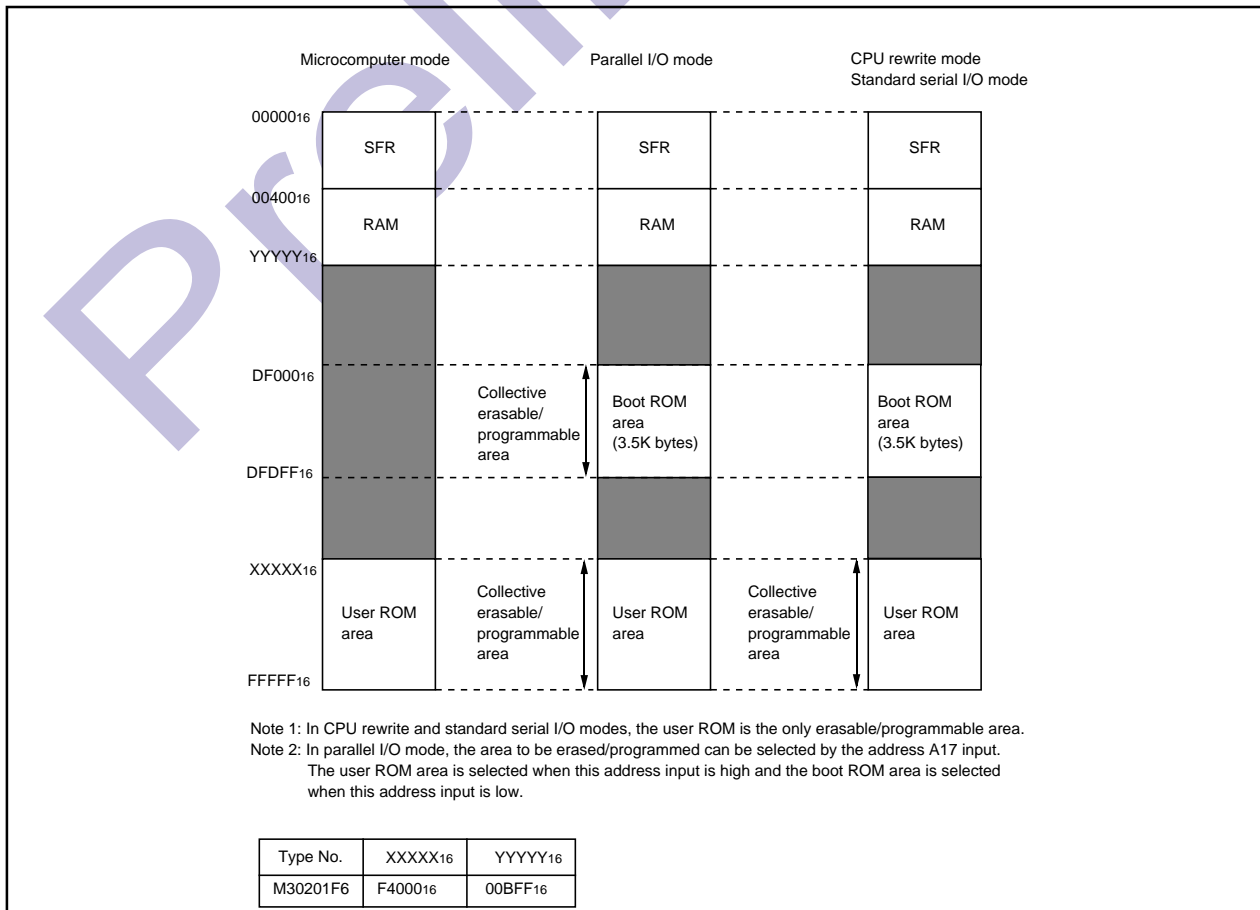
The parallel I/O mode is entered by making connections shown in Figures CC-2 and CC-3 and then turning the VPPH power supply on. In this mode, the M30201 (flash memory version) operates in a manner similar to the NOR flash memory M5M28F101 from Mitsubishi. Note, however, that there are some differences with regard to the functions not available with the microcomputer (function of read device identification code) and matters related to memory capacity.

Table CC-2 shows pin relationship between the M30201 and M5M28F101 in parallel I/O mode.

**Table CC-2. Pin relationship in parallel I/O mode**

	M30201(flash memory version)	M5M28F101
Vcc	Vcc	Vcc
Vss	Vss	Vss
Address input	P60 to P63, P30 to P33, P10 to P17, P50	A0 to A15, A17
Data I/O	P00 to P07	D0 to D7
OE input	P41	$\overline{OE}$
CE input	P43	$\overline{CE}$
WE input	P40	$\overline{WE}$
VERFY input (Note)	P51	—

Note: The VRFY input only selects read-only or read/write mode, and does not have any pin associated with it on the M5M28F101.



**Figure CC-1. Block diagram of flash memory version**

Appendix Parallel I/O Mode

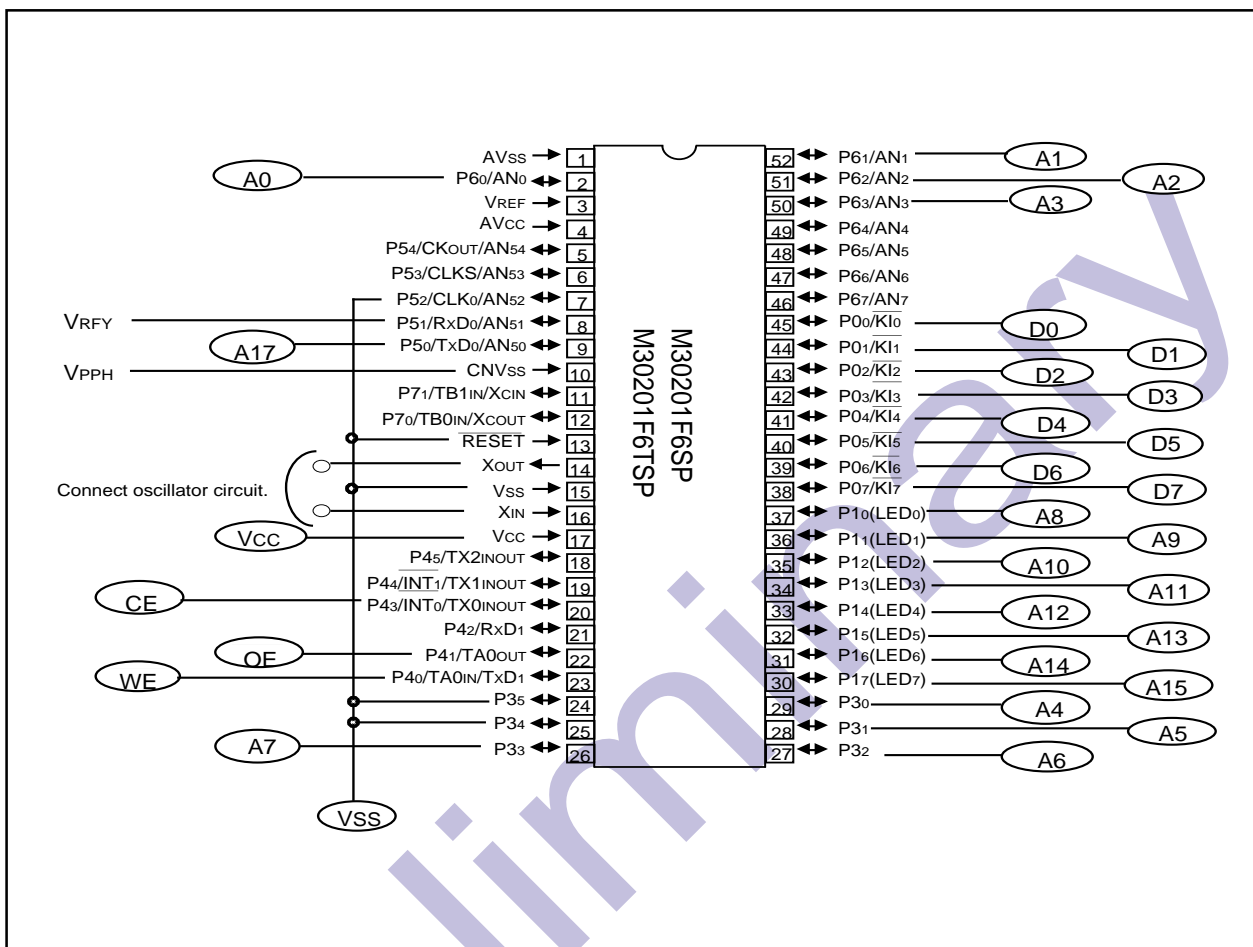


Figure CC-2. Pin connection diagram in parallel I/O mode (1)

Appendix Parallel I/O Mode

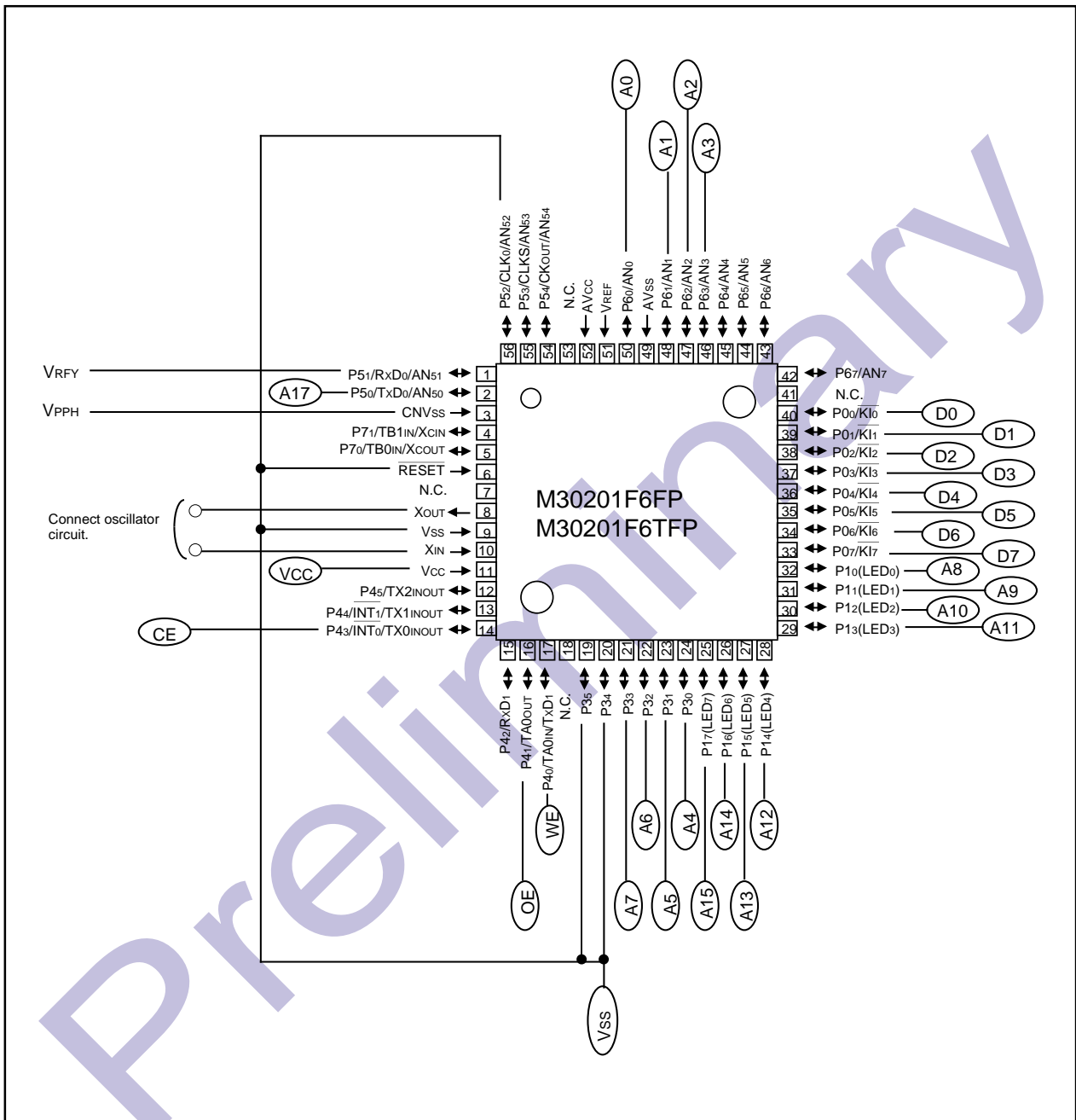


Figure CC-3. Pin connection diagram in parallel I/O mode (2)



## Appendix Parallel I/O Mode

**User ROM and Boot ROM Areas**

In parallel I/O mode, the user ROM and boot ROM areas shown in Figure CC-1 can be rewritten.

In the boot ROM area, an erase block operation is applied to only one 4 K byte block. The boot ROM area has had a standard serial I/O mode control program stored in it when shipped from the Mitsubishi factory.

Therefore, using the device in standard serial input/output mode, the user does not need to write to the boot ROM area.

**Functional Outline (Parallel I/O Mode)**

In parallel I/O mode, bus operation modes—Read, Output Disable, Standby, and Write—are selected by the status of the  $\overline{CE}$ ,  $\overline{OE}$ ,  $\overline{WE}$ ,  $V_{RFY}$ , and  $CNVSS$  input pins.

The contents of erase, program, and other operations are selected by writing a software command. The data in memory can only be read out by a read after software command input.

Program and erase operations are controlled using software commands.

**Table CC-3. Relationship between control signals and bus operation modes**

Mode \ Pin name		$\overline{CE}$	$\overline{OE}$	$\overline{WE}$	$V_{RFY}$	$V_{PP}$	D <sub>0</sub> to D <sub>7</sub>
Read only	Read	V <sub>IL</sub>	V <sub>IL</sub>	V <sub>IH</sub>	V <sub>IL</sub>	V <sub>PPH</sub>	Data output
	Output disabled	V <sub>IL</sub>	V <sub>IH</sub>	V <sub>IH</sub>	V <sub>IL</sub>	V <sub>PPH</sub>	Hi-Z
	Stand by	V <sub>IH</sub>	X	X	V <sub>IL</sub>	V <sub>PPH</sub>	Hi-Z
Read/Write	Read	V <sub>IL</sub>	V <sub>IL</sub>	V <sub>IH</sub>	V <sub>IH</sub>	V <sub>PPH</sub>	Data output
	Output disabled	V <sub>IL</sub>	V <sub>IH</sub>	V <sub>IH</sub>	V <sub>IH</sub>	V <sub>PPH</sub>	Hi-Z
	Stand by	V <sub>IH</sub>	X	X	V <sub>IH</sub>	V <sub>PPH</sub>	Hi-Z
	Write	V <sub>IL</sub>	V <sub>IH</sub>	V <sub>IL</sub>	V <sub>IH</sub>	V <sub>PPH</sub>	Data input

Note: X can be V<sub>IL</sub> or V<sub>IH</sub>.

## Appendix Parallel I/O Mode

---

The following explains about bus operation modes, software commands, and status register.

### Bus Operation Modes

Read-only mode is entered by applying  $V_{PPH}$  to the CNVSS pin and a low voltage to the VRFY pin. Read-only mode has three states: Read, Output Disable, and Standby which are selected by setting the  $\overline{CE}$ ,  $\overline{OE}$ , and  $\overline{WE}$  pins high or low.

Read-write mode is entered by applying  $V_{PPH}$  to the CNVSS pin and a high voltage to the VRFY pin. Read-write mode has four states: Read, Output Disable, Standby, and Write which are selected by setting the  $\overline{CE}$ ,  $\overline{OE}$ , and  $\overline{WE}$  pins high or low.

#### Read

The Read mode is entered by pulling the  $\overline{WE}$  pin high when the  $\overline{CE}$  and  $\overline{OE}$  pins are low. In Read mode, the data corresponding to each software command entered is output from the data I/O pins D<sub>0</sub>–D<sub>7</sub>.

#### Output Disable

The Output Disable mode is entered by pulling the  $\overline{CE}$  pin low and the  $\overline{WE}$  and  $\overline{OE}$  pins high. Also, the data I/O pins are placed in the high-impedance state.

#### Standby

The Standby mode is entered by driving the  $\overline{CE}$  pin high. Also, the data I/O pins are placed in the high-impedance state.

#### Write

The Write mode is entered by applying  $V_{PPH}$  to the CNVSS pin and a high voltage to the VRFY pin and then pulling the  $\overline{WE}$  pin low when the  $\overline{CE}$  pin is low and  $\overline{OE}$  pin is high. In this mode, the device accepts the software commands or write data entered from the data I/O pins. A program, erase, or some other operation is initiated depending on the content of the software command entered here. The input data such as address is latched at the falling edge of  $\overline{WE}$  pin. The input data such as software command is latched at the rising edge of  $\overline{WE}$  pin.

## Appendix Parallel I/O Mode

**Software Commands**

Table CC-4 lists the software commands available with the M30201 (flash memory version). By entering a software command from the data I/O pins (D<sub>0</sub>–D<sub>7</sub>) in Write mode, specify the content of the operation, such as erase or program operation, to be performed.

The following explains the content of each software command.

**Table CC-4. Software command list (parallel I/O mode)**

Command	First bus cycle			Second bus cycle		
	Mode	Address	Data (D <sub>0</sub> to D <sub>7</sub> )	Mode	Address	Data (D <sub>0</sub> to D <sub>7</sub> )
Read	Write	x	00 <sub>16</sub>			
Program	Write	x	40 <sub>16</sub>	Write	Program address	Program data
Program verify	Write	x	C0 <sub>16</sub>	Read	x	Verify data
Erase	Write	x	20 <sub>16</sub>	Write	x	20 <sub>16</sub>
Erase verify	Write	Verify address	A0 <sub>16</sub>	Read	x	Verify data
Reset	Write	x	FF <sub>16</sub>	Write	x	FF <sub>16</sub>

**Read Command (00<sub>16</sub>)**

The read mode is entered by writing the command code “00<sub>16</sub>” in the first bus cycle. When an address to be read is input in one of the bus cycles that follow, the content of the specified address is read out at the data I/O pins (D<sub>0</sub>–D<sub>7</sub>).

The read mode is retained intact until another command is written.

After reset and after the reset command is executed, the read mode is set.

**Program Command (40<sub>16</sub>)**

The program mode is entered by writing the command code “40<sub>16</sub>” in the first bus cycle. When an address and data to be program is write in the second bus cycle, the flash memory control circuit executes the program operation. The program operation requires approximately 20 μs. Wait for 20 μs or more before the user go to the next processing.

Note 1: The write operation is not completed immediately by writing a program command once. The user must always execute a program-verify command after each program command executed. And if verification fails, the user need to execute the program command repeatedly until the verification passes. See Figure CC-4 for an example of a programming flowchart.

## Appendix Parallel I/O Mode

---

### Program-verify command (C016)

The program-verify mode is entered by writing the command code "C016" in the first bus cycle and the verify data is output from the data I/O pins (D0–D7) in the second bus cycle.

### Erase command (2016 + 2016)

The flash memory control circuit executes an erase operation by writing command code "2016" in the first bus cycle and the same command code again in the second bus cycle. The erase operation requires approximately 20 ms. Wait for 20 ms or more before the user go to the next processing. Before this erase command can be performed, all memory locations to be erased must have had data "0016" written to by using the program and program-verify commands.

Note 1: The erase operation is not completed immediately by writing an erase command once. The user must always execute an erase-verify command after each erase command executed. And if verification fails, the user need to execute the erase command repeatedly until the verification passes. See Figure CC-4 for an example of an erase flowchart.

### Erase-verify command (A016)

The erase-verify mode is entered by writing the command code "A016" in the first bus cycle and the verify data is output from the data I/O pins (D0–D7) in the second bus cycle.

Note 1: If any unerased memory location is encountered during erase-verify operation, be sure to execute erase and erase-verify operations one more time. In this case, however, the user does not need to write data "0016" to memory before erasing.

**Reset command (FF<sub>16</sub> + FF<sub>16</sub>)**

The reset command is used to stop the program command or the erase command in the middle of operation. After writing command code “40<sub>16</sub>” or “20<sub>16</sub>” twice, write command code “FF<sub>16</sub>” in the first bus cycle and the same command code again in the second bus cycle. The program command or erase command is disabled, with the flash memory placed in read mode.

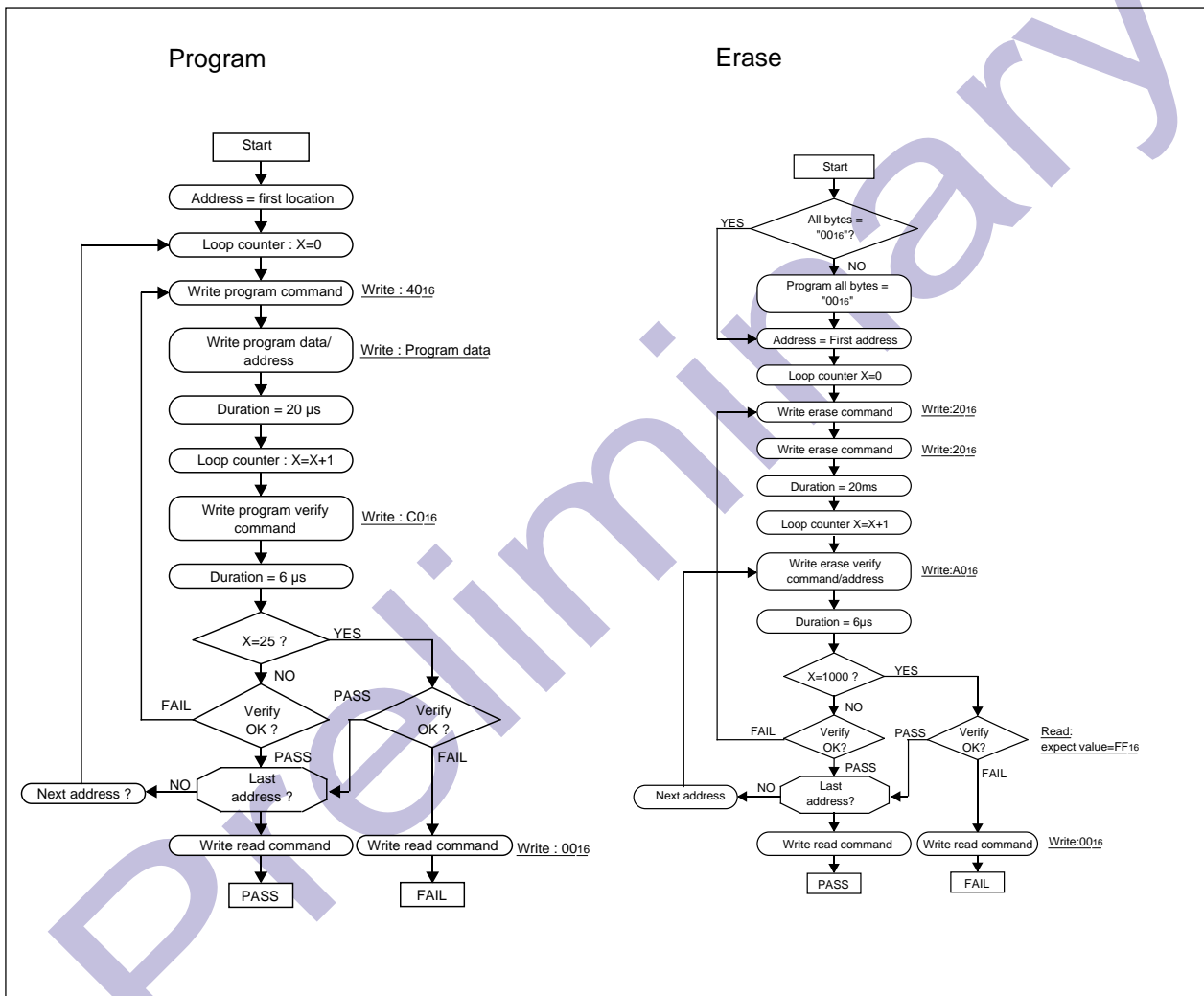


Figure CC-4. Program and erase execution flowchart in the CPU rewrite mode

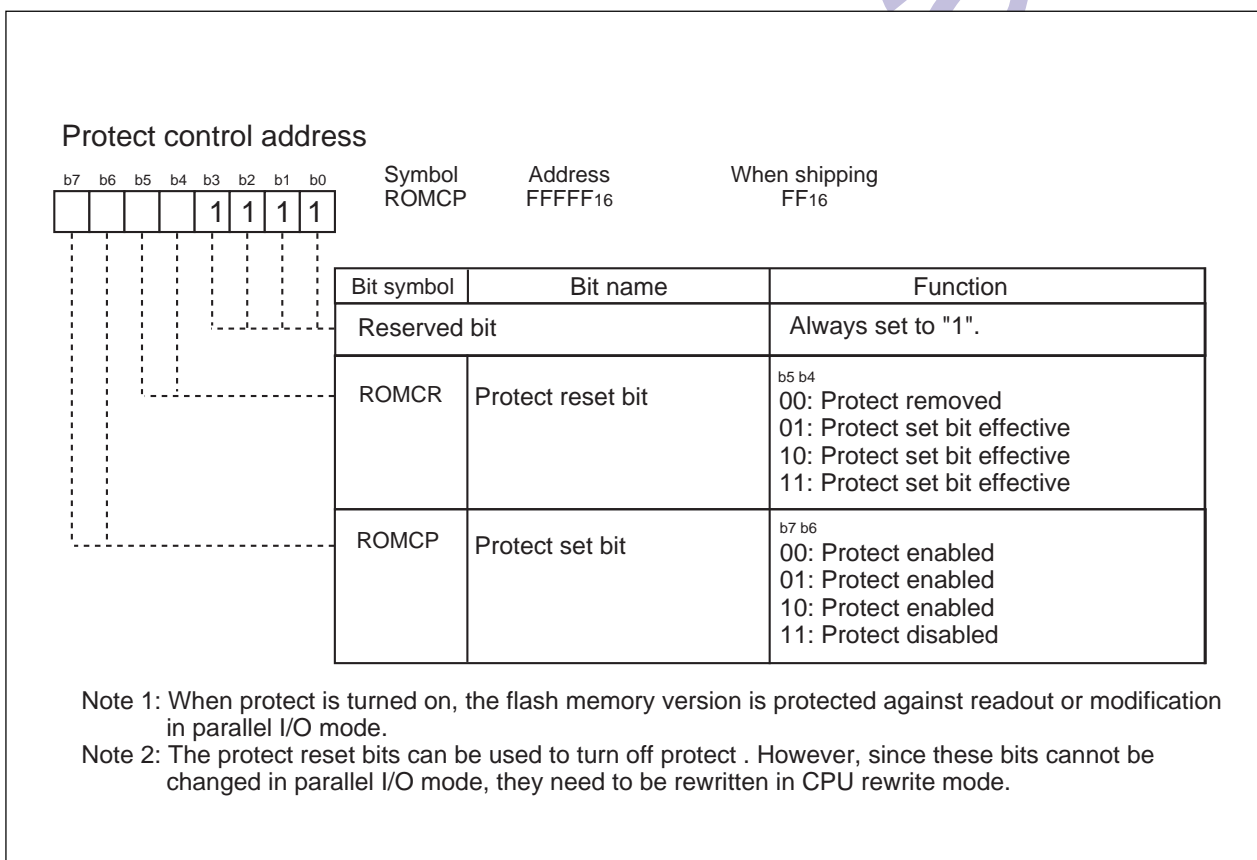
## Appendix Parallel I/O Mode

**Protect function**

In parallel I/O mode, the internal flash memory has the "protect function" available. This function protects the flash memory contents from being read or rewritten easily.

Depending on the content at the protect control address (FFFFF<sub>16</sub>) in parallel I/O mode, this function inhibits the flash memory contents against read or modification. The protect control address (FFFFF<sub>16</sub>) is shown in Figure CC-5. (This address exists in the user ROM area.)

The protect function is enabled by setting one of the two protect set bits to "0", so that the internal flash memory contents are inhibited against read or modification. The protect function is disabled by setting both of the two protect reset bits to "00", so that the internal flash memory contents can be read or modified. Once the protect function is set, the user cannot change settings of the protect clear bits while in parallel I/O mode. Settings of the protect reset bits can only be changed in CPU rewrite mode.



**Figure CC-5. Protect control address**

## Appendix Standard Serial I/O Mode

## Pin functions (Flash memory standard serial I/O mode)

Pin	Name	I/O	Description
Vcc,Vss	Power input		Apply 5V $\pm$ 10 % to Vcc pin and 0 V to Vss pin.
CNVss	CNVss	I	Apply 12V $\pm$ 5 % to this pin.
RESET	Reset input	I	Reset input pin. While reset is "L" level, a 20 cycle or longer clock must be input to XIN pin.
XIN	Clock input	I	Connect a ceramic resonator or crystal oscillator between XIN and XOUT pins. To input an externally generated clock, input it to XIN pin and open XOUT pin.
XOUT	Clock output	O	
AVcc, AVss	Analog power supply input		Connect AVss to Vss and AVcc to Vcc, respectively.
VREF	Reference voltage input	I	Enter the reference voltage for AD from this pin.
P0 <sub>0</sub> to P0 <sub>7</sub>	Input port P0	I	Input "H" or "L" level signal or open.
P1 <sub>0</sub> to P1 <sub>7</sub>	Input port P1	I	Input "H" or "L" level signal or open.
P3 <sub>0</sub> to P3 <sub>5</sub>	Input port P3	I	Input "H" or "L" level signal or open.
P4 <sub>0</sub> to P4 <sub>5</sub>	Input port P4	I	Input "H" or "L" level signal or open.
P5 <sub>4</sub>	Input port P5	I	Input "H" or "L" level signal or open.
P5 <sub>0</sub>	TxD output	O	Serial data output pin.
P5 <sub>1</sub>	RxD input	I	Serial data input pin.
P5 <sub>2</sub>	SCLK input	I	Serial clock input pin.
P5 <sub>3</sub>	BUSY output	O	BUSY signal output pin.
P6 <sub>0</sub> to P6 <sub>7</sub>	Input port P6	I	Input "H" or "L" level signal or open.
P7 <sub>0</sub> to P7 <sub>1</sub>	Input port P7	I	Input "H" or "L" level signal or open.

Appendix Standard Serial I/O Mode

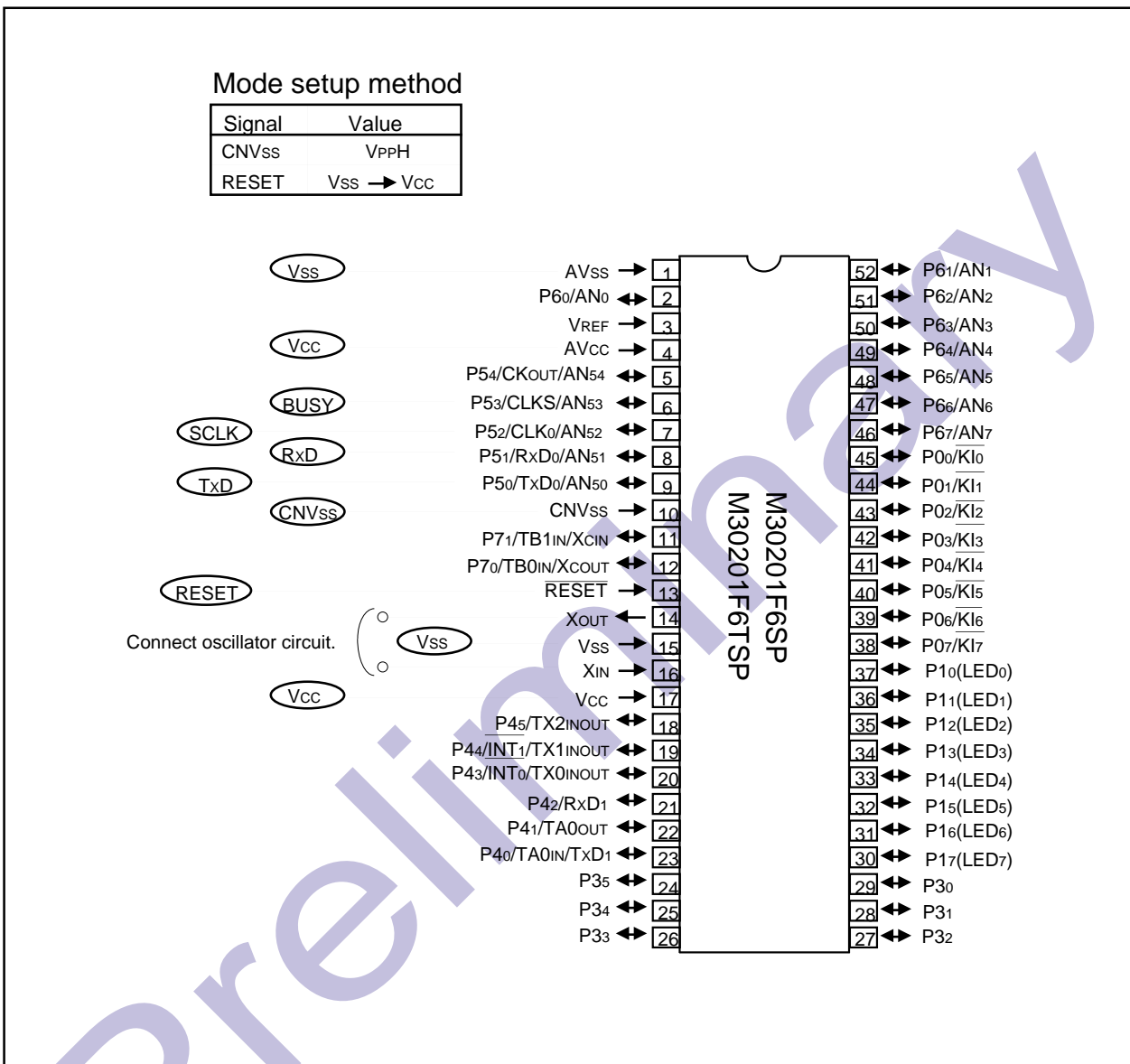


Figure DD-1. Pin connections for serial I/O mode (1)



Appendix Standard Serial I/O Mode

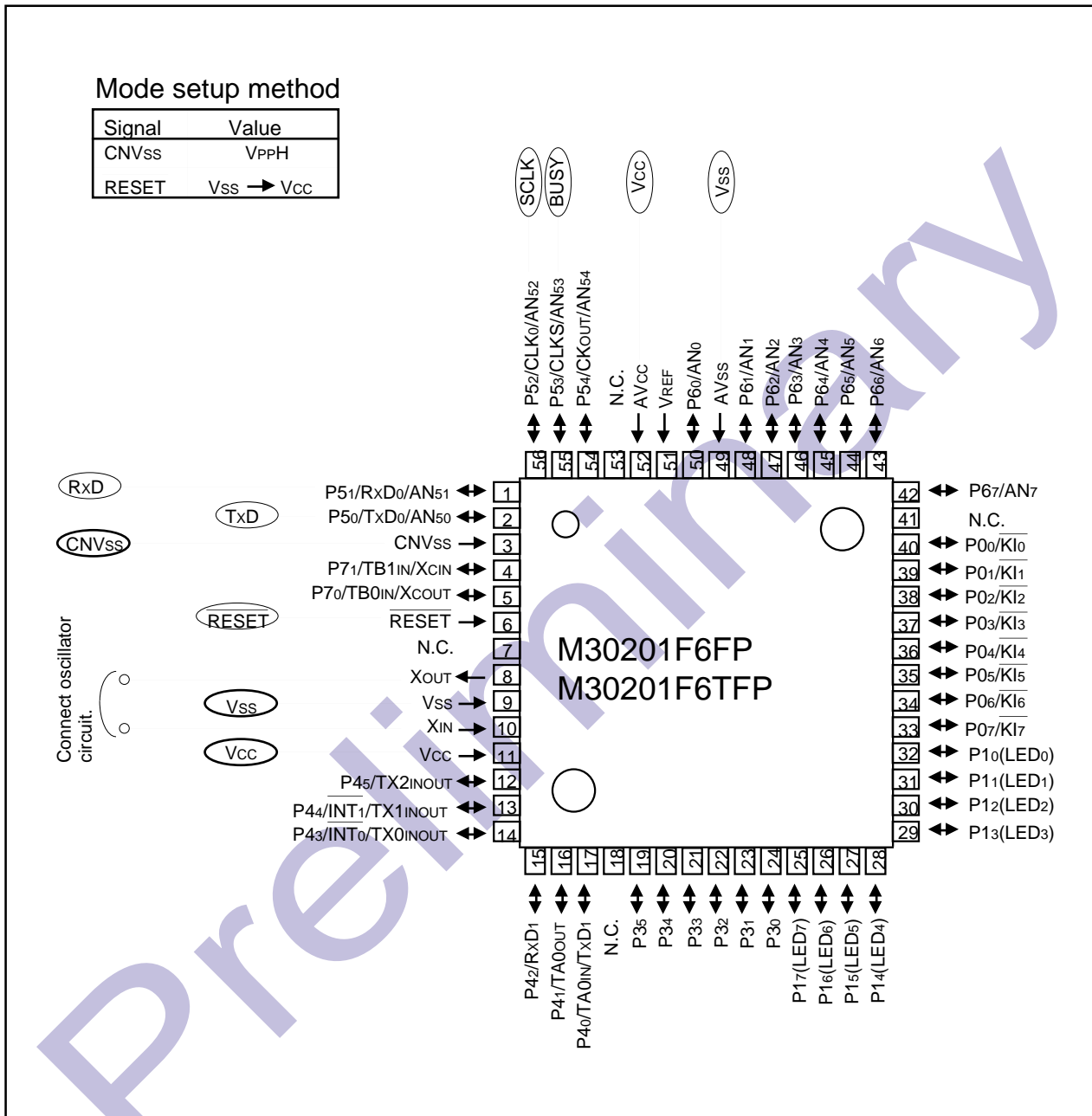


Figure DD-2. Pin connections for serial I/O mode (2)

## Appendix Standard Serial I/O Mode

---

### Standard Serial I/O Mode

The standard serial I/O mode serially inputs and outputs the software commands, addresses and data necessary for operating (read, program, erase, etc.) the internal flash memory. It uses a purpose-specific serial programmer.

The standard serial I/O mode differs from the parallel I/O mode in that the CPU controls operations like rewriting (uses the CPU rewrite mode) in the flash memory or serial input for rewriting data. The standard serial I/O mode is started by clearing the reset with VPPH at the CNVss pin. (For the normal microprocessor mode, set CNVss to "L".)

This control program is written in the boot ROM area when shipped from Mitsubishi Electric. Therefore, if the boot ROM area is rewritten in the parallel I/O mode, the standard serial I/O mode cannot be used. Figures DD-1 and DD-2 show the pin connections for the standard serial I/O mode. Serial data I/O uses three UART0 pins: CLK0, RxD0, and TxD0 and port P53 (BUSY).

The CLK0 pin is the transfer clock input pin and it transfers the external transfer clock. The TxD0 pin outputs the CMOS signal. The P53 (BUSY) pin outputs an "L" level when reception setup ends and an "H" level when the reception operation starts. Transmission and reception data is transferred serially in 8-byte blocks.

In the standard serial I/O mode, only the user ROM area shown in Figure CC-1 can be rewritten, the boot ROM area cannot.

The standard serial I/O mode has a 7-byte ID code. When the flash memory is not blank and the ID code does not match the content of the flash memory, the command sent from the programmer is not accepted.

### Function Overview (Standard Serial I/O Mode)

In the standard serial I/O mode, software commands, addresses and data are input and output between the flash memory and an external device (serial programmer, etc.) using a clock synchronized serial I/O (UART0) and P53. In reception, the software commands, addresses and program data are synchronized with the rise of the transfer clock input to the CLK0 pin and input into the flash memory via the RxD0 pin. In transmission, the read data and status are synchronized with the fall of the transfer clock and output to the outside from the TxD0 pin.

The TxD1 pin is CMOS output. Transmission is in 8-bit blocks and LSB first.

When busy, either during transmission or reception, or while executing an erase operation or program, the P53 (BUSY) pin is "H" level. Accordingly, do not start the next transmission until the P53 (BUSY) pin is "L" level.

Also, data in memory and the status register can be read after inputting a software command. It is possible to check flash memory operating status or whether a program or erase operation ended successfully or in error by reading the status register.

Software commands and the status register are explained here following.

## Appendix Standard Serial I/O Mode

## Software Commands

Table DD-1 lists software commands. In the standard serial I/O mode, erase operations, programs and reading are controlled by transferring software commands via the RxD pin. Software commands are explained here below.

Table DD-1. Software commands (Standard serial I/O mode)

	Control command		2nd byte	3rd byte	4th byte	5th byte	6th byte		When ID is not verificate
1	Page read	FF <sub>16</sub>	Address (middle)	Address (high)	Data output	Data output	Data output	Data output to 259th byte	Not acceptable
2	Page program	41 <sub>16</sub>	Address (middle)	Address (high)	Data input	Data input	Data input	Data input to 259th byte	Not acceptable
3	Erase all unlocked blocks	A7 <sub>16</sub>	D0 <sub>16</sub>						Not acceptable
4	Read status register	70 <sub>16</sub>	SRD output	SRD1 output					Acceptable
5	Clear status register	50 <sub>16</sub>							Not acceptable
6	Read lockbit status	71 <sub>16</sub>	Address (middle)	Address (high)	Lock bit data output				Not acceptable
7	ID check function	F5 <sub>16</sub>	Address (low)	Address (middle)	Address (high)	ID size	ID1	To ID7	Acceptable
8	Download function	FA <sub>16</sub>	Size (low)	Size (high)	Check-sum	Data input	To required number of times		Not acceptable
9	Version data output function	FB <sub>16</sub>	Version data output	Version data output	Version data output	Version data output	Version data output	Version data output to 9th byte	Acceptable
14	Boot area output function	FC <sub>16</sub>	Address (middle)	Address (high)	Data output	Data output	Data output	Data output to 259th byte	Not acceptable

Note1: Shading indicates transfer from flash memory microcomputer to serial programmer. All other data is transferred from the serial programmer to the flash memory microcomputer.

Note2: SRD refers to status register data. SRD1 refers to status register 1 data.

Note3: All commands can be accepted when the flash memory is totally blank.

## Appendix Standard Serial I/O Mode

**Page Read Command**

This command reads the specified page (256 bytes) in the flash memory sequentially one byte at a time. Execute the page read command as explained here following.

- (1) Send the "FF16" command code in the 1st byte of the transmission.
- (2) Send addresses A8 to A15 and A16 to A23 in the 2nd and 3rd bytes of the transmission respectively.
- (3) From the 4th byte onward, data (D0–D7) for the page (256 bytes) specified with addresses A8 to A23 will be output sequentially from the smallest address first in sync with the rise of the clock.

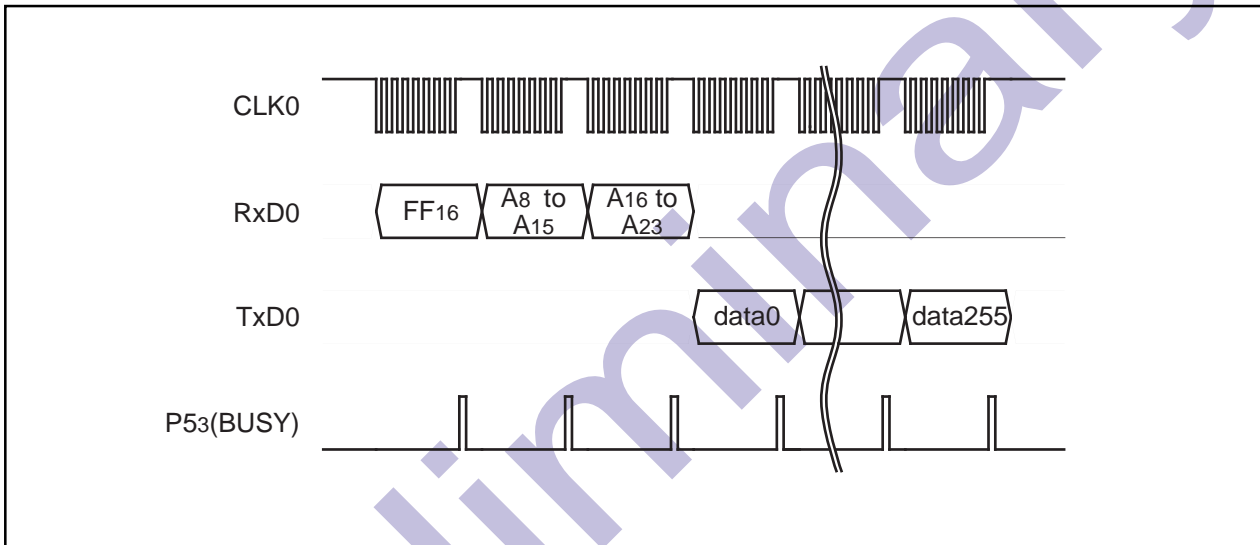


Figure DD-3. Timing for page read

**Read Status Register Command**

This command reads status information. When the "7016" command code is sent in the 1st byte of the transmission, the contents of the status register (SRD) specified in the 2nd byte of the transmission and the contents of status register 1 (SRD1) specified in the 3rd byte of the transmission are read.

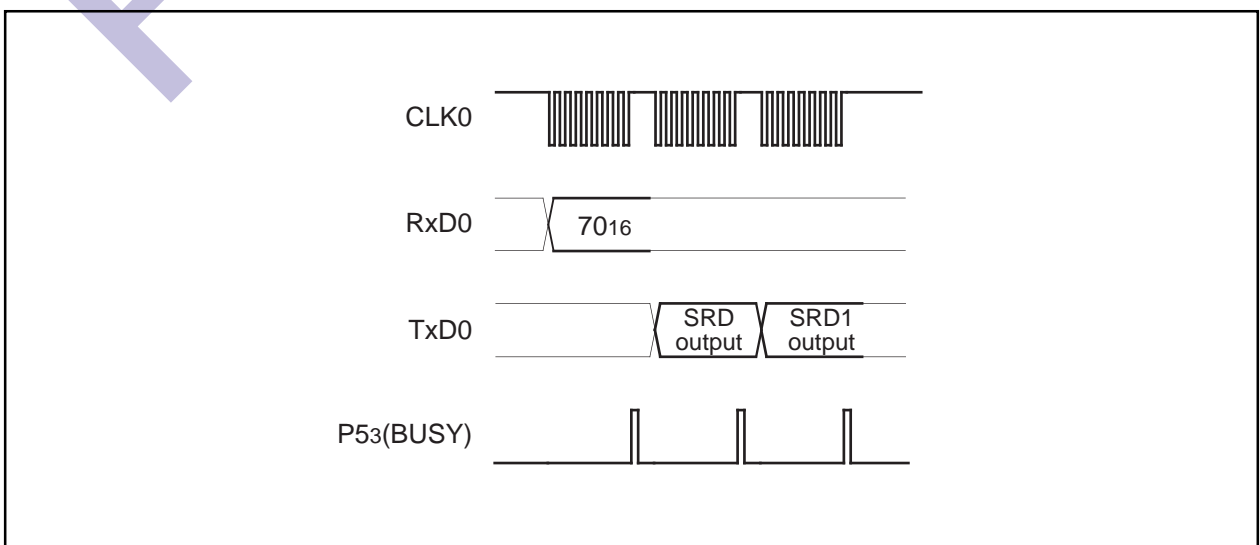


Figure DD-4. Timing for reading the status register

Appendix Standard Serial I/O Mode

**Clear Status Register Command**

This command clears the bits (SR3–SR4) which are set when the status register operation ends in error. When the “5016” command code is sent in the 1st byte of the transmission, the aforementioned bits are cleared. When the clear status register operation ends, the P53 (BUSY) signal changes from the “H” to the “L” level.

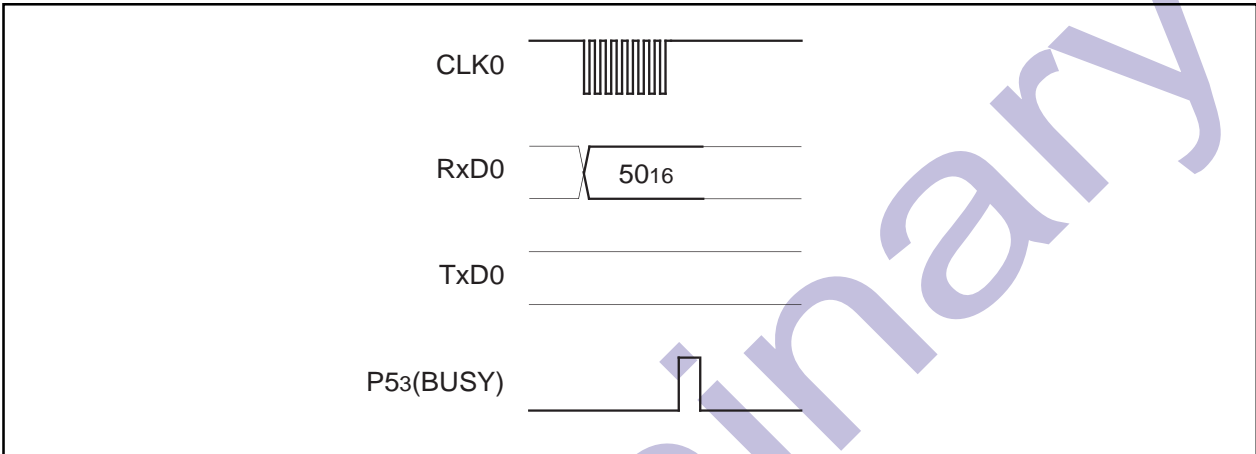


Figure DD-5. Timing for clearing the status register

**Page Program Command**

This command writes the specified page (256 bytes) in the flash memory sequentially one byte at a time. Execute the page program command as explained here following.

- (1) Send the “4116” command code in the 1st byte of the transmission.
- (2) Send addresses A8 to A15 and A16 to A23 in the 2nd and 3rd bytes of the transmission respectively.
- (3) From the 4th byte onward, as write data (D0–D7) for the page (256 bytes) specified with addresses A8 to A23 is input sequentially from the smallest address first, that page is automatically written.

When reception setup for the next 256 bytes ends, the P53 (BUSY) signal changes from the “H” to the “L” level. The result of the page program can be known by reading the status register. For more information, see the section on the status register.

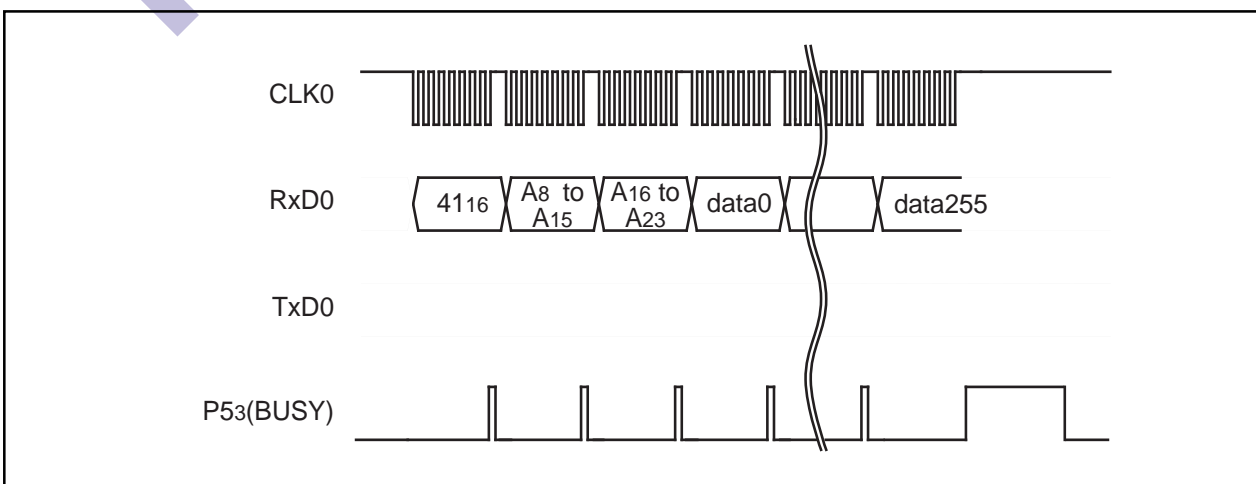


Figure DD-6. Timing for the page program

## Appendix Standard Serial I/O Mode

**Erase All Unlocked Blocks Command**

This command erases the content of all blocks. Execute the erase all unlocked blocks command as explained here following.

- (1) Send the "A716" command code in the 1st byte of the transmission.
- (2) Send the verify command code "D016" in the 2nd byte of the transmission. With the verify command code, the erase operation will start and continue for all blocks in the flash memory.

When block erasing ends, the P53 (BUSY) signal changes from the "H" to the "L" level. The result of the erase operation can be known by reading the status register.

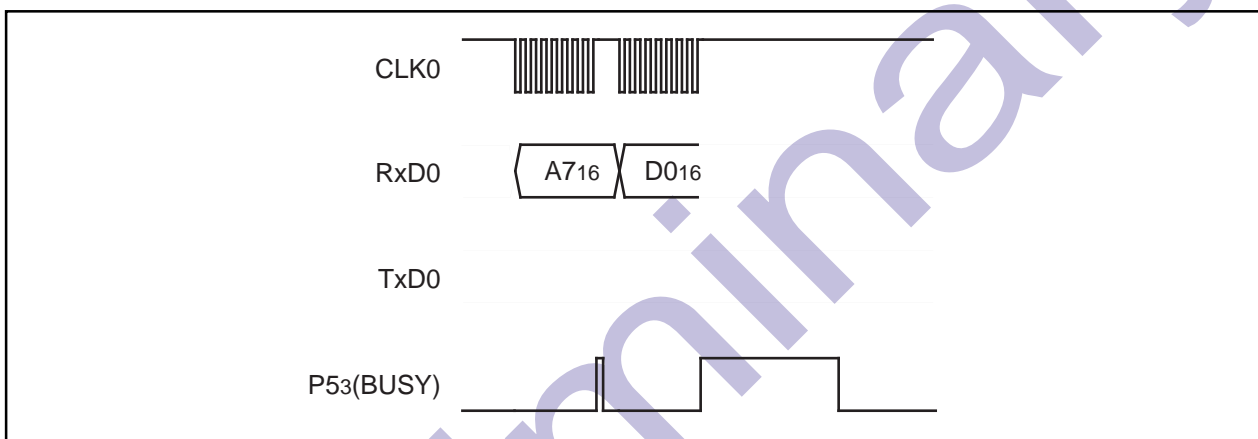


Figure DD-7. Timing for erasing all unlocked blocks

**Read Lock Bit Status Command**

This command reads the lock bit status of the specified block. Execute the read lock bit status command as explained here following.

- (1) Send the "7116" command code in the 1st byte of the transmission.
- (2) Send addresses A8 to A15 and A16 to A23 in the 2nd and 3rd bytes of the transmission respectively.
- (3) The lock bit data of the specified block is output in the 4th byte of the transmission. Write the highest address of the specified block for addresses A8 to A23.

The M30201 (flash memory version) does not have the lock bit, so the read value is always "1" (block unlock).

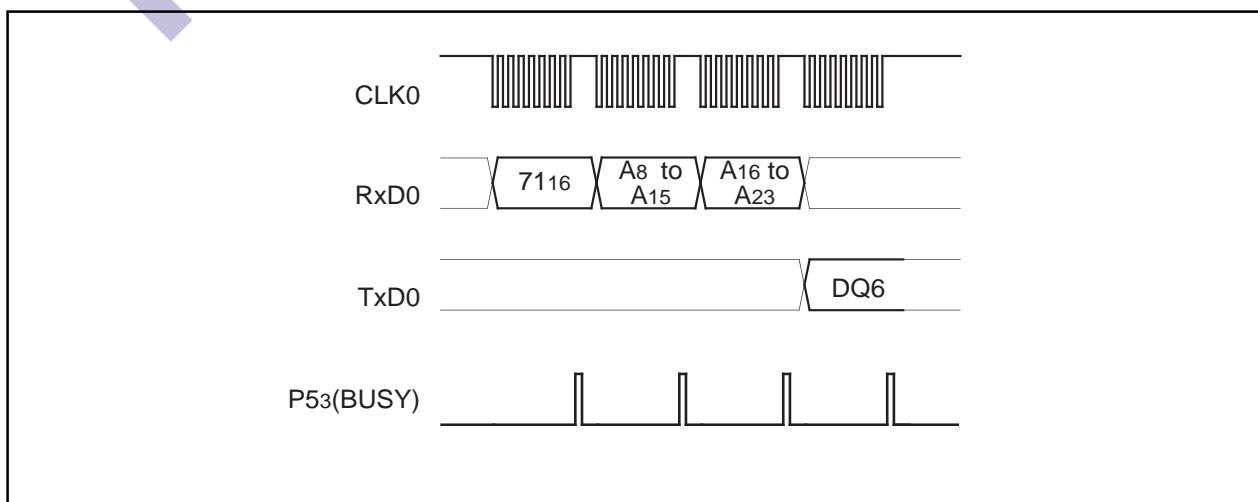


Figure DD-8. Timing for reading lock bit status

## Appendix Standard Serial I/O Mode

**Download Command**

This command downloads a program to the RAM for execution. Execute the download command as explained here following.

- (1) Send the "FA16" command code in the 1st byte of the transmission.
- (2) Send the program size in the 2nd and 3rd bytes of the transmission.
- (3) Send the check sum in the 4th byte of the transmission. The check sum is added to all data sent in the 5th byte onward.
- (4) The program to execute is sent in the 5th byte onward.

When all data has been transmitted, if the check sum matches, the downloaded program is executed. The size of the program will vary according to the internal RAM.

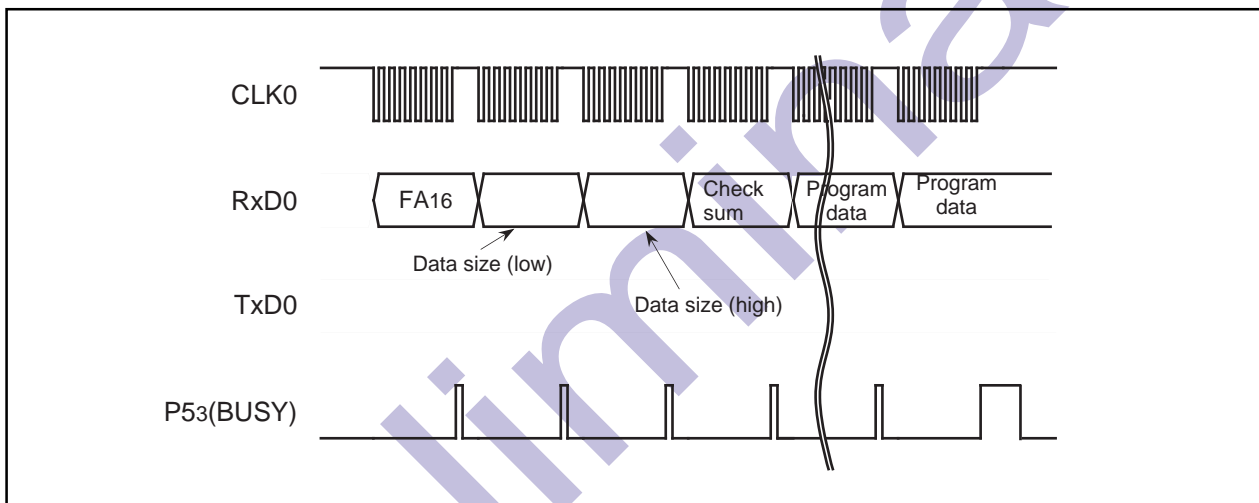


Figure DD-9. Timing for download

## Appendix Standard Serial I/O Mode

**Version Information Output Command**

This command outputs the version information of the control program stored in the boot area. Execute the version information output command as explained here following.

- (1) Send the "FB16" command code in the 1st byte of the transmission.
- (2) The version information will be output from the 2nd byte onward. This data is composed of 8 ASCII code characters.

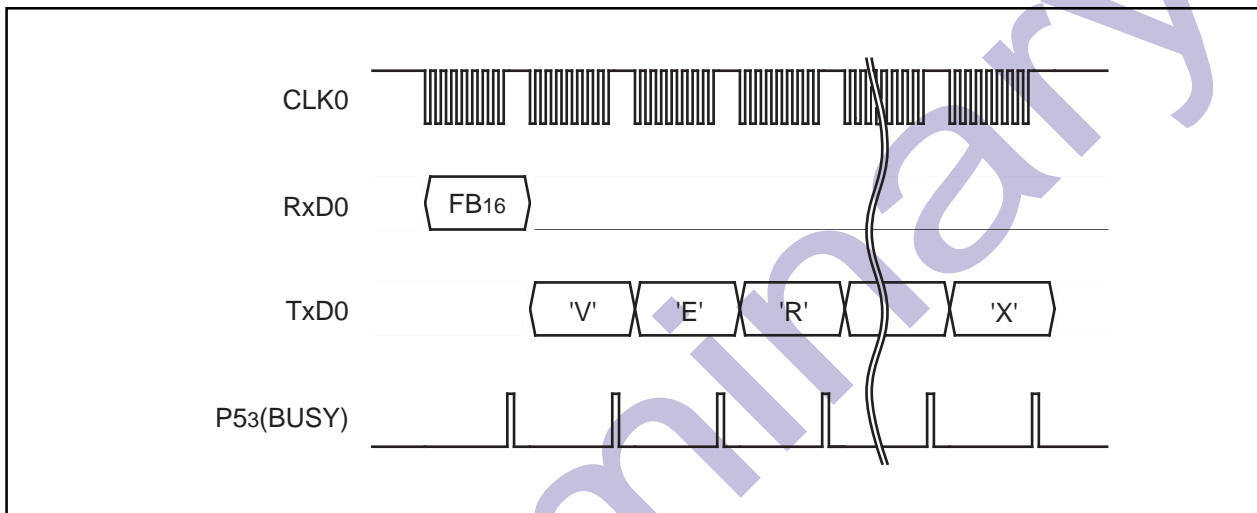


Figure DD-10. Timing for version information output

**Boot Area Output Command**

This command outputs the control program stored in the boot area in one page blocks (256 bytes). Execute the boot area output command as explained here following.

- (1) Send the "FC16" command code in the 1st byte of the transmission.
- (2) Send addresses A8 to A15 and A16 to A23 in the 2nd and 3rd bytes of the transmission respectively.
- (3) From the 4th byte onward, data (D0–D7) for the page (256 bytes) specified with addresses A8 to A23 will be output sequentially from the smallest address first, in sync with the rise of the clock.

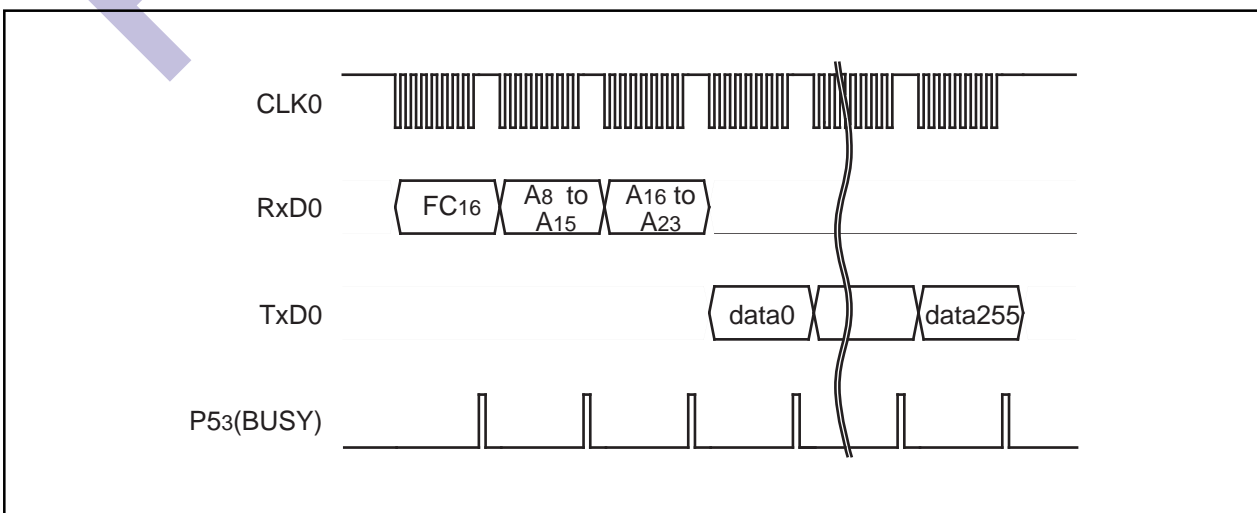


Figure DD-11. Timing for boot area output



Appendix Standard Serial I/O Mode

**ID Check**

This command checks the ID code. Execute the boot ID check command as explained here following.

- (1) Send the "F5<sub>16</sub>" command code in the 1st byte of the transmission.
- (2) Send addresses A<sub>0</sub> to A<sub>7</sub>, A<sub>8</sub> to A<sub>15</sub> and A<sub>16</sub> to A<sub>23</sub> of the 1st byte of the ID code in the 2nd, 3rd and 4th bytes of the transmission respectively.
- (3) Send the number of data sets of the ID code in the 5th byte.
- (4) The ID code is sent in the 6th byte onward, starting with the 1st byte of the code.

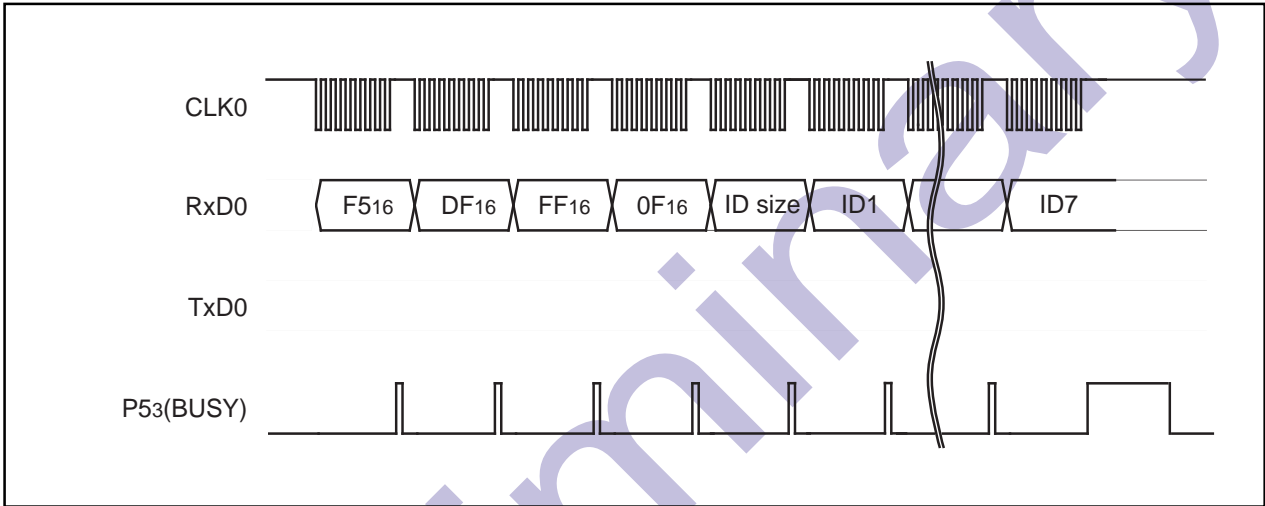


Figure DD-12. Timing for the ID check

**ID Code**

When the flash memory is not blank, the ID code sent from the serial programmer and the ID code written in the flash memory are compared to see if they match. If the codes do not match, the command sent from the serial programmer is not accepted. An ID code contains 8 bits of data. Area is, from the 1st byte, addresses 0FFFDF<sub>16</sub>, 0FFFE3<sub>16</sub>, 0FFFE7<sub>16</sub>, 0FFFEF<sub>16</sub>, 0FFFF3<sub>16</sub>, and 0FFFF7<sub>16</sub>. Write a program into the flash memory, which already has the ID code set for these addresses.

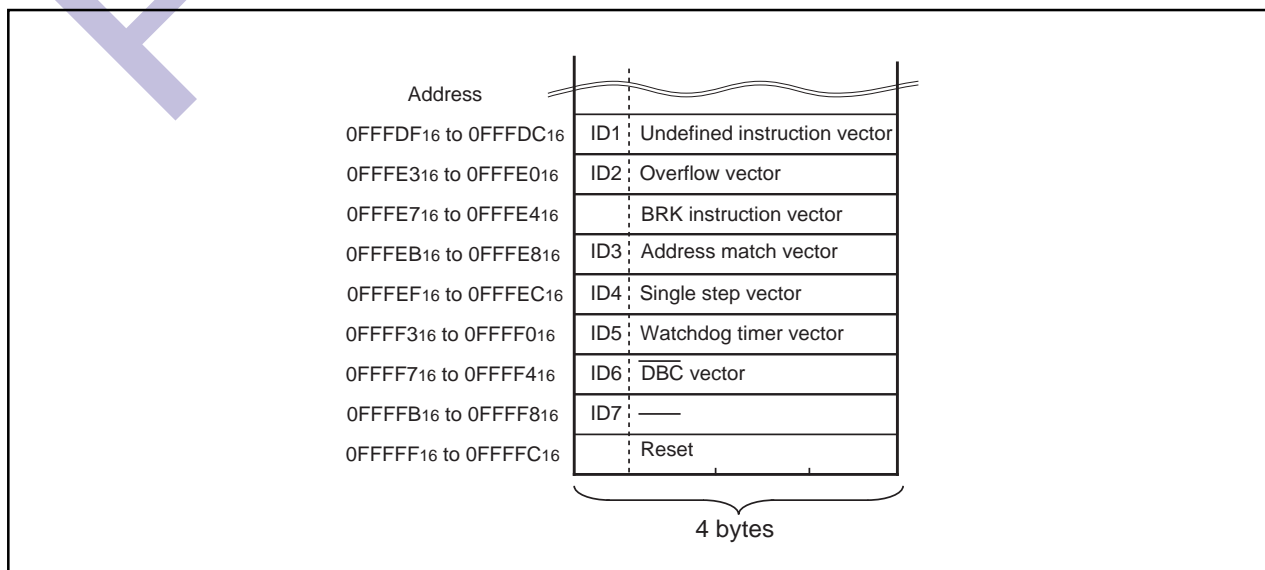


Figure DD-13. ID code storage addresses

## Appendix Standard Serial I/O Mode

**Status Register (SRD)**

The status register indicates operating status of the flash memory and status such as whether an erase operation or a program ended successfully or in error. It can be read by writing the read status register command (70<sub>16</sub>). Also, the status register is cleared by writing the clear status register command (50<sub>16</sub>). Table DD-2 gives the definition of each status register bit. After clearing the reset, the status register outputs "80<sub>16</sub>".

**Table DD-2. Status register (SRD)**

SRD0 bits	Status name	Definition	
		"1"	"0"
SR7 (bit7)	Status bit	Ready	Busy
SR6 (bit6)	Reserved	-	-
SR5 (bit5)	Erase bit	Terminated in error	Terminated normally
SR4 (bit4)	Program bit	Terminated in error	Terminated normally
SR3 (bit3)	Reserved	-	-
SR2 (bit2)	Reserved	-	-
SR1 (bit1)	Reserved	-	-
SR0 (bit0)	Reserved	-	-

**Status Bit (SR7)**

The status bit indicates the operating status of the flash memory. When power is turned on, "1" (ready) is set for it. The bit is set to "0" (busy) during an auto write or auto erase operation, but it is set back to "1" when the operation ends.

**Erase Bit (SR5)**

The erase bit reports the operating status of the auto erase operation. If an erase error occurs, it is set to "1". When the erase status is cleared, it is set to "0".

**Program Bit (SR4)**

The program bit reports the operating status of the auto write operation. If a write error occurs, it is set to "1". When the program status is cleared, it is set to "0".

## Appendix Standard Serial I/O Mode

**Status Register 1 (SRD1)**

Status register 1 indicates the status of serial communications, results from ID checks and results from check sum comparisons. It can be read after the SRD by writing the read status register command (7016). Also, status register 1 is cleared by writing the clear status register command (5016).

Table DD-3 gives the definition of each status register 1 bit. "0016" is output when power is turned ON and the flag status is maintained even after the reset.

**Table DD-3. Status register 1 (SRD1)**

SRD1 bits	Status name	Definition	
		"1"	"0"
SR15 (bit7)	Boot update completed bit	Update completed	Not update
SR14 (bit6)	Reserved	-	-
SR13 (bit5)	Reserved	-	-
SR12 (bit4)	Checksum match bit	Match	Mismatch
SR11 (bit3) SR10 (bit2)	ID check completed bits	00 01 10 11	Not verified Verification mismatch Reserved Verified
SR9 (bit1)	Data receive time out	Time out	Normal operation
SR8 (bit0)	Reserved	-	-

**Boot Update Completed Bit (SR15)**

This flag indicates whether the control program was downloaded to the RAM or not, using the download function.

**Check Sum Consistency Bit (SR12)**

This flag indicates whether the check sum matches or not when a program, is downloaded for execution using the download function.

**ID Check Completed Bits (SR11 and SR10)**

These flags indicate the result of ID checks. Some commands cannot be accepted without an ID check.

**Data Reception Time Out (SR9)**

This flag indicates when a time out error is generated during data reception. If this flag is attached during data reception, the received data is discarded and the microcomputer returns to the command wait state.

## Appendix Standard Serial I/O Mode

**Example Circuit Application for The Standard Serial I/O Mode**

The below figure shows a circuit application for the standard serial I/O mode. Control pins will vary according to programmer, therefore see the programmer manual for more information.

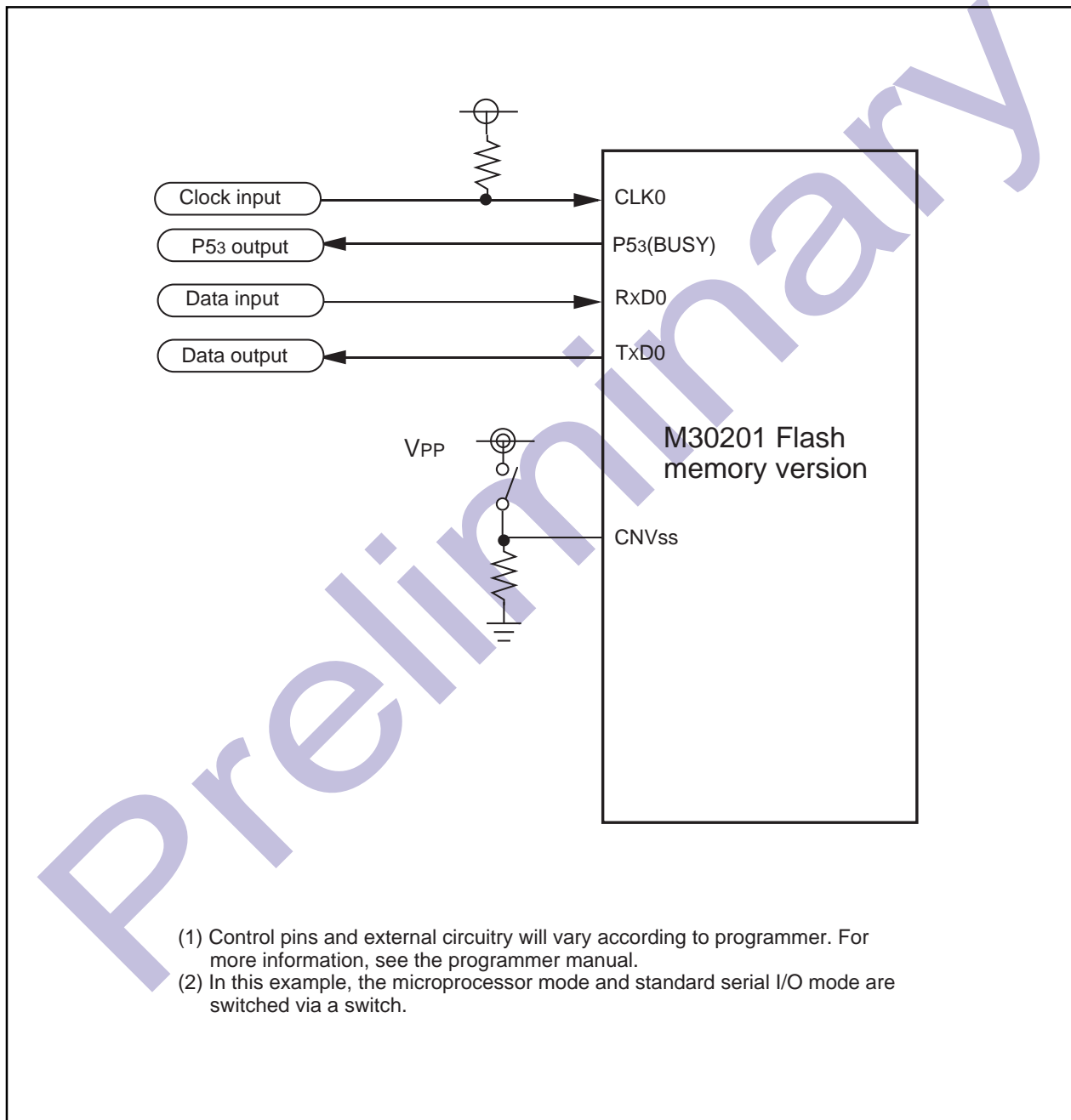
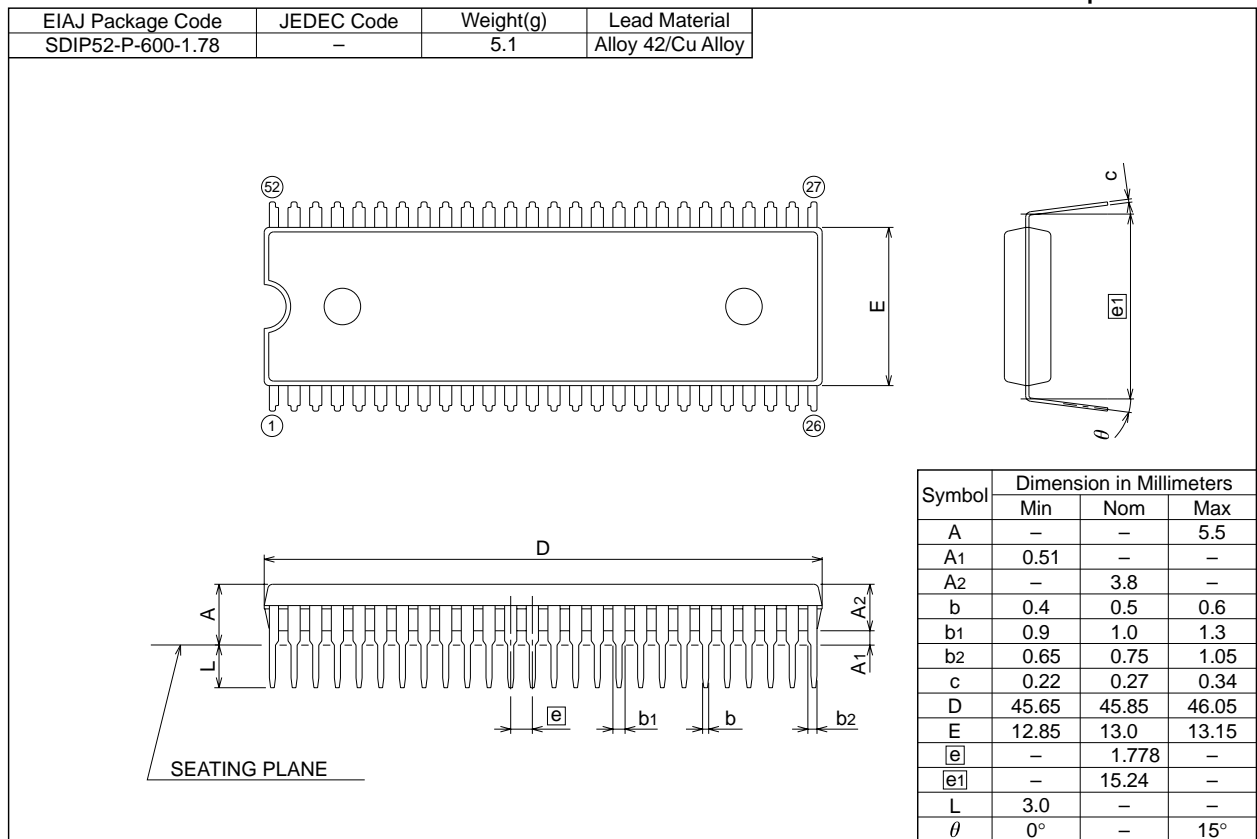


Figure DD-14. Example circuit application for the standard serial I/O mode

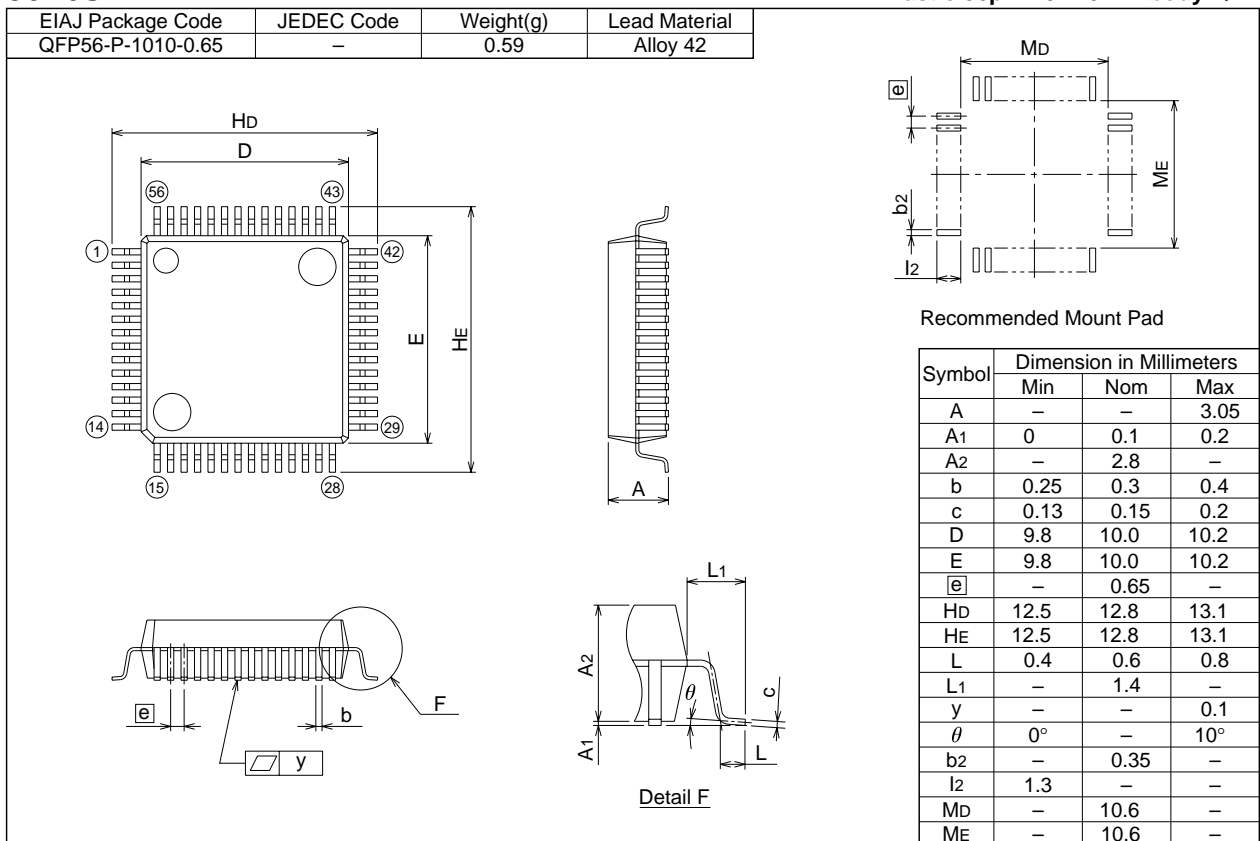
### 52P4B

Plastic 52pin 600mil SDIP



### 56P6S-A

Plastic 56pin 10X10mm body QFP



### Keep safety first in your circuit designs!

- Mitsubishi Electric Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage. Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of non-flammable material or (iii) prevention against any malfunction or mishap.

### Notes regarding these materials

- These materials are intended as a reference to assist our customers in the selection of the Mitsubishi semiconductor product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Mitsubishi Electric Corporation or a third party.
- Mitsubishi Electric Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
- All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Mitsubishi Electric Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Mitsubishi Electric Corporation or an authorized Mitsubishi Semiconductor product distributor for the latest product information before purchasing a product listed herein.  
The information described here may contain technical inaccuracies or typographical errors. Mitsubishi Electric Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.  
Please also pay attention to information published by Mitsubishi Electric Corporation by various means, including the Mitsubishi Semiconductor home page (<http://www.mitsubishichips.com>).
- When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Mitsubishi Electric Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
- Mitsubishi Electric Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Mitsubishi Electric Corporation or an authorized Mitsubishi Semiconductor product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
- The prior written approval of Mitsubishi Electric Corporation is necessary to reprint or reproduce in whole or in part these materials.
- If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.  
Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
- Please contact Mitsubishi Electric Corporation or an authorized Mitsubishi Semiconductor product distributor for further details on these materials or the products contained therein.

MITSUBISHI SEMICONDUCTORS  
M30201 Group DATA SHEET REV.D

---

April First Edition 1998

July Second Edition 1998

February Third Edition 1999

May Fourth Edition 1999

Edited by

Committee of editing of Mitsubishi Semiconductor DATA SHEET

Published by

---

Mitsubishi Electric Corp., Kitaitami Works

This book, or parts thereof, may not be reproduced in any form without  
permission of Mitsubishi Electric Corporation.

©1999 MITSUBISHI ELECTRIC CORPORATION