

To all our customers

---

## **Regarding the change of names mentioned in the document, such as Mitsubishi Electric and Mitsubishi XX, to Renesas Technology Corp.**

---

The semiconductor operations of Hitachi and Mitsubishi Electric were transferred to Renesas Technology Corporation on April 1st 2003. These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.) Accordingly, although Mitsubishi Electric, Mitsubishi Electric Corporation, Mitsubishi Semiconductors, and other Mitsubishi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp. Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

Note : Mitsubishi Electric will continue the business operations of high frequency & optical devices and power devices.

Renesas Technology Corp.  
Customer Support Dept.  
April 1, 2003

## Description

### Description

The M16C/62A (80-pin version) group of single-chip microcomputers are built using the high-performance silicon gate CMOS process using a M16C/60 Series CPU core and are packaged in a 80-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. They also feature a built-in multiplier and DMAC, making them ideal for controlling office, communications, industrial equipment, and other high-speed processing applications.

The M16C/62A (80-pin version) group includes a wide range of products with different internal memory types and sizes and various package types.

### Features

- Memory capacity .....ROM (See Figure 1.1.3. ROM Expansion)  
RAM 3K to 20K bytes
- Shortest instruction execution time .....62.5ns (f(XIN)=16MHz, VCC=5V)  
100ns (f(XIN)=10MHz, VCC=3V, with software one-wait) : Mask ROM, flash memory 5V version
- Supply voltage .....4.2V to 5.5V (f(XIN)=16MHz, without software wait) : Mask ROM, flash memory 5V version  
2.7V to 5.5V (f(XIN)=10MHz with software one-wait) : Mask ROM, flash memory 5V version
- Low power consumption .....25.5mW ( f(XIN)=10MHz, with software one-wait, VCC = 3V)
- Interrupts .....25 internal and 5 external interrupt sources, 4 software interrupt sources; 7 levels (including key input interrupt)
- Multifunction 16-bit timer .....5 output timers + 6 input timers (3 for timer function only)
- Serial I/O .....5 channels (2 for UART or clock synchronous, 1 for UART, 2 for clock synchronous)
- DMAC .....2 channels (trigger: 24 sources)
- A-D converter .....10 bits X 8 channels (Expandable up to 10 channels)
- D-A converter .....8 bits X 2 channels
- CRC calculation circuit .....1 circuit
- Watchdog timer .....1 line
- Programmable I/O .....70 lines
- Input port .....1 line (P85 shared with  $\overline{\text{NMI}}$  pin)
- Clock generating circuit .....2 built-in clock generation circuits  
(built-in feedback resistor, and external ceramic or quartz oscillator)

Note: Memory expansion mode and microprocessor mode are not supported.

### Applications

Audio, cameras, office equipment, communications equipment, portable equipment

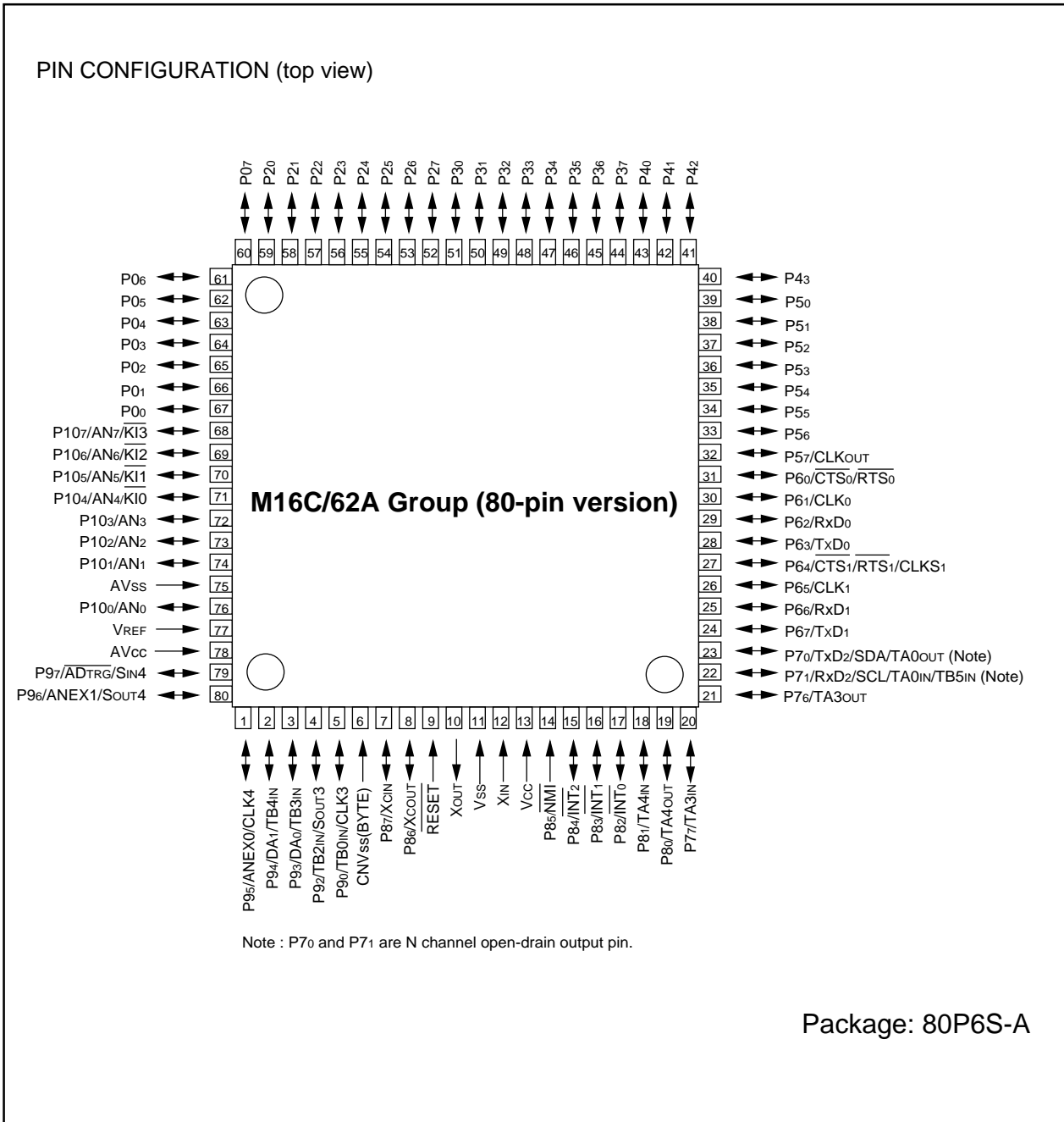
## -----Table of Contents-----

About the M16C/62A (80-pin version) group .. 7	Timer ..... 68
Central Processing Unit (CPU) ..... 11	Serial I/O ..... 86
Reset ..... 14	A-D Converter ..... 127
Processor Mode ..... 21	D-A Converter ..... 137
Clock Generating Circuit ..... 26	CRC Calculation Circuit ..... 139
Protection ..... 35	Programmable I/O Ports ..... 141
Interrupts ..... 36	Electric Characteristics ..... 154
Watchdog Timer ..... 56	Flash memory version ..... 183
DMAC ..... 58	

Description

**Pin Configuration**

Figures 1.1.1 show the pin configurations (top view).

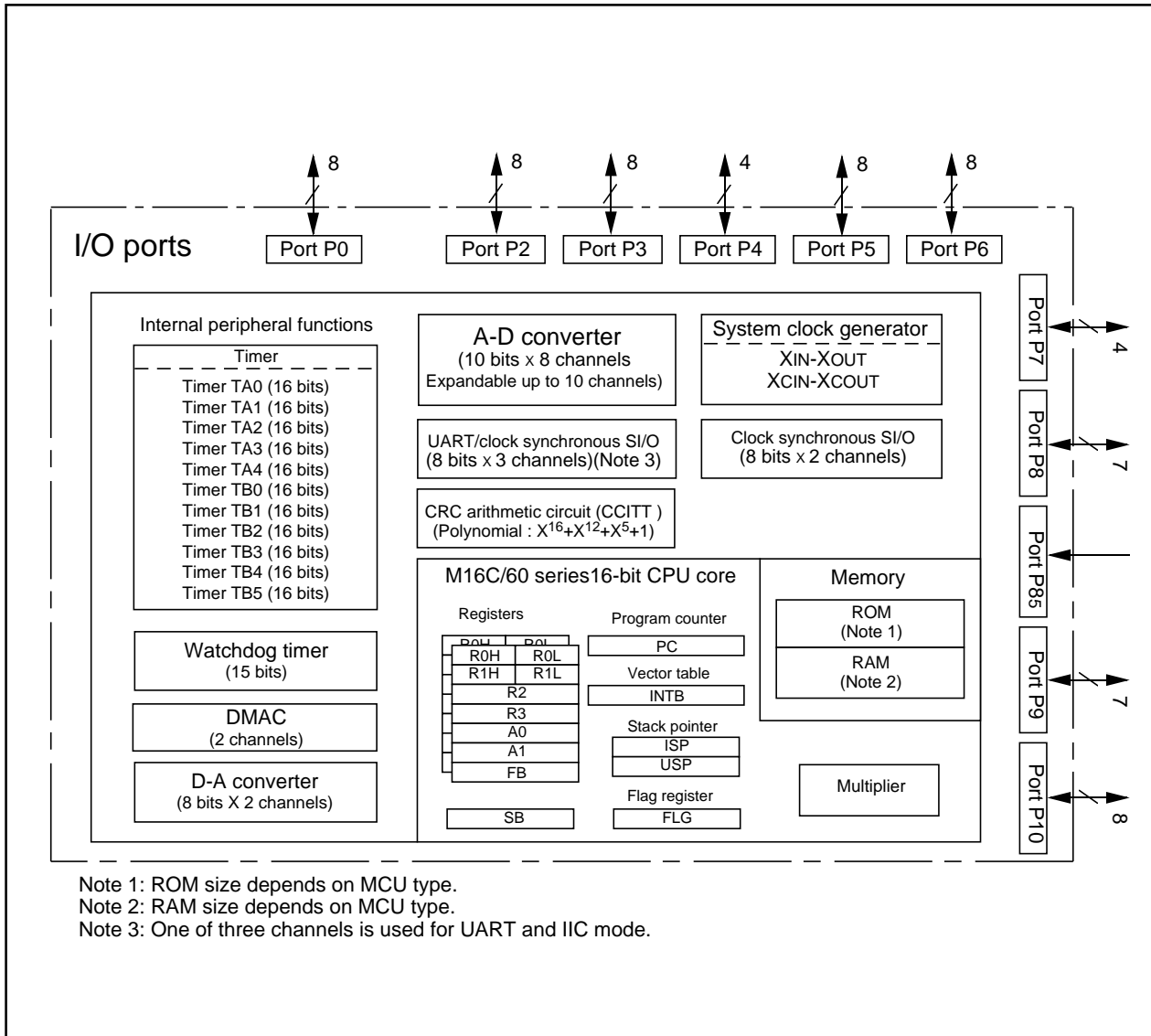


**Figure 1.1.1. Pin configuration (top view)**

Description

**Block Diagram**

Figure 1.1.2 is a block diagram of the M16C/62A (80-pin version) group.



**Figure 1.1.2. Block diagram of M16C/62A (80-pin version) group**

Description

**Performance Outline**

Table 1.1.1 is a performance outline of M16C/62A (80-pin version) group.

**Table 1.1.1. Performance outline of M16C/62A (80-pin version) group**

Item		Performance
Number of basic instructions		91 instructions
Shortest instruction execution time		62.5ns(f(XIN)=16MHz, VCC=5V) 100ns (f(XIN)=10MHz, VCC=3V, with software one-wait) : Mask ROM, flash memory 5V version
Memory capacity	ROM	(See the figure 1.1.3. ROM Expansion)
	RAM	3K to 20K bytes
I/O port	P0 to P10 (except P85)	8 bits x 6, 7 bits x 2, 4 bits x 2
Input port	P85	1 bit x 1
Multifunction timer	TA0, TA3, TA4	16 bits x 3 (timer mode, internal/external event count, one-shot timer mode and pulse width measurement mode)
	TB0, TB2, TB3, TB4, TB5	16 bits x 5 (timer mode, internal/external event count and pulse period/pulse width measurement mode)
	TA1, TA2	16 bits x 2 (timer mode, internal event count and a trigger through one-shot timer mode occurs.)
	TB1	16 bits x 1 (timer mode and internal event count)
Serial I/O	UART0, UART1, UART2	(UART or clock synchronous) x 2, UART x 1(UART2)
	SI/O3, SI/O4	(Clock synchronous) x 2 (SI/O3 is output only)
A-D converter		10 bits x (8 + 2) channels
D-A converter		8 bits x 2
DMAC		2 channels (trigger: 24 sources)
CRC calculation circuit		CRC-CCITT
Watchdog timer		15 bits x 1 (with prescaler)
Interrupt		25 internal and 5 external sources, 4 software sources, 7 levels
Clock generating circuit		2 built-in clock generation circuits (built-in feedback resistor, and external ceramic or quartz oscillator)
Supply voltage		4.2V to 5.5V (f(XIN)=16MHz, without software wait) : Mask ROM, flash memory 5V version 2.7V to 5.5V (f(XIN)=10MHz with software one-wait) : Mask ROM, flash memory 5V version
Power consumption		25.5mW (f(XIN) = 10MHz, VCC=3V with software one-wait)
I/O characteristics	I/O withstand voltage	5V
	Output current	5mA
Device configuration		CMOS high performance silicon gate
Package		80-pin plastic mold QFP

Note : M16C/62A (80-pin version) group does not support memory expansion or microprocessor mode.

Description

Mitsubishi plans to release the following products in the M16C/62A (80-pin version) group:

- (1) Support for mask ROM version and flash memory version
- (2) ROM capacity
- (3) Package
  - 80P6S-A : Plastic molded QFP (mask ROM and flash memory versions)

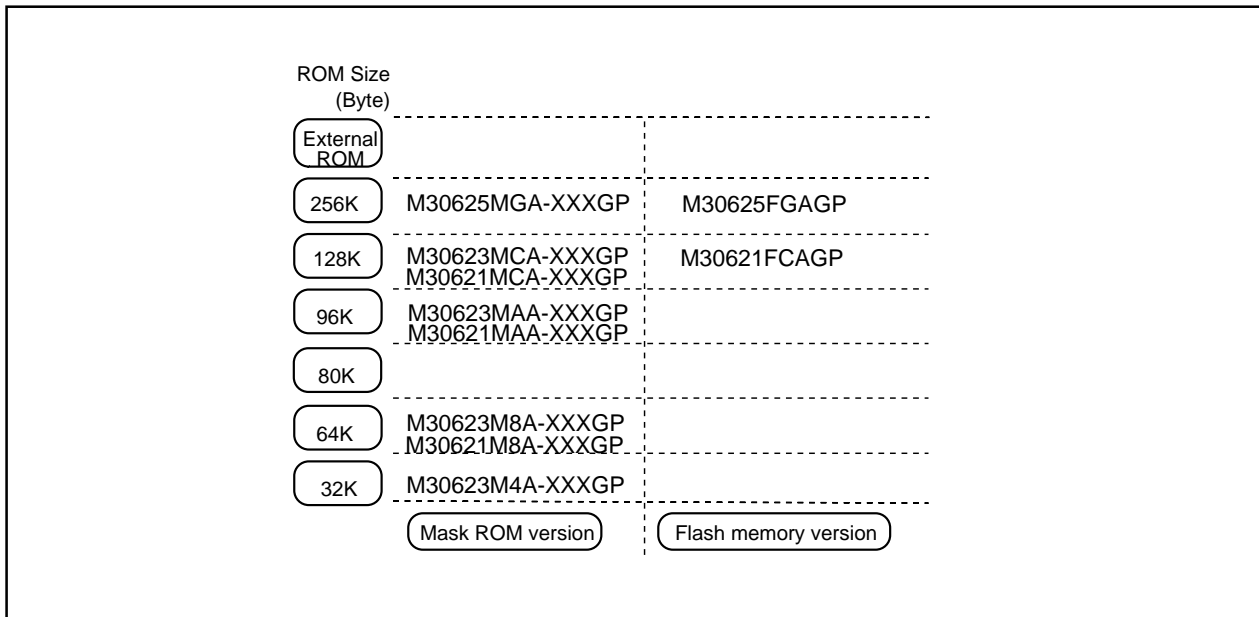


Figure 1.1.3. ROM expansion

The M16C/62A (80-pin version) group products currently supported are listed in Table 1.1.2.

Table 1.1.2. M16C/62A (80-pin version) group

As of November 2001

Type No	ROM capacity	RAM capacity	Package type	Remarks
M30623M4A-XXXGP	32 Kbytes	3 Kbytes	80P6S-A	mask ROM version
M30623M8A-XXXGP	64 Kbytes	4 Kbytes	80P6S-A	
M30623MAA-XXXGP	96 Kbytes	5 Kbytes	80P6S-A	
M30623MCA-XXXGP	128 Kbytes	5 Kbytes	80P6S-A	
M30621M8A-XXXGP	64 Kbytes	10 Kbytes	80P6S-A	
M30621MAA-XXXGP	96 Kbytes	10 Kbytes	80P6S-A	
M30621MCA-XXXGP	128 Kbytes	10 Kbytes	80P6S-A	
M30625MGA-XXXGP	256 Kbytes	20 Kbytes	80P6S-A	
M30621FCAGP	128 Kbytes	10 Kbytes	80P6S-A	Flash memory 5V version
M30625FGAGP	256 Kbytes	20 Kbytes	80P6S-A	

Description

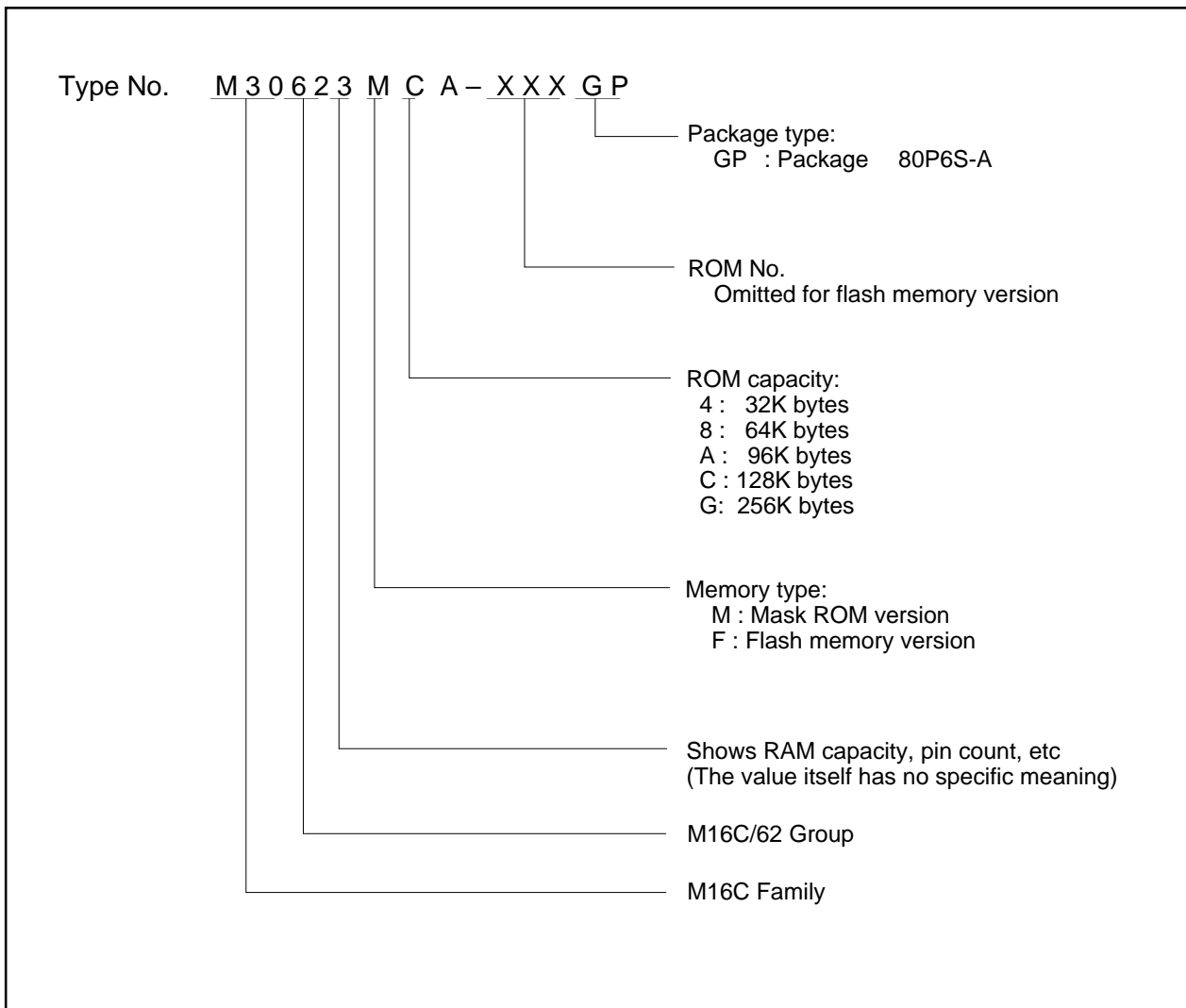


Figure 1.1.4. Type No., memory size, and package

## Description

**About the M16C/62A (80-pin version) group**

The M16C/62A (80-pin version) group is packaged in a 80-pin plastic mold package. The number of pins in comparison with the 100-pin package products is decreased. So be careful about the following.

- (a) The M16C/62A (80-pin version) group supports single chip mode alone. It supports neither memory expansion mode nor microprocessor mode.
- (b) The input/output ports given below are absent from the M16C/62A (80-pin version) group. To stabilize the internal state, set to output mode the direction register of each input/output port. Failing in setting to output mode involves an increase in current consumption.

<Pins absent from the 80-pin version>

P10 to P17, P44 to P47, P72 to P75, P91

- (c)  $\overline{\text{INT3}}$  to  $\overline{\text{INT5}}$  allocated to P15 to P17 cannot be used. Keep the  $\overline{\text{INT3}}$  interrupt control register disabled for interrupts. The  $\overline{\text{INT4}}$  interrupt control register and the  $\overline{\text{INT5}}$  interrupt control register are shared with SI/O3 and SI/O4. When the user don't use them as SI/O3 and SI/O4, set them disabled for interrupts.
- (d) The output pins of timers A1 and A2 - TA1IN, TA1OUT, TA2IN and TA2OUT - allocated to P72 to P75 cannot be used. In connection with this, the gate function and pulse outputting function of timers A1 and A2 cannot be used. Use timer mode and internal event count, or use as trigger signal generation in one-shot timer mode.
- (e) The UART2 input/output pins - CLK2 and  $\overline{\text{CTS2}}/\overline{\text{RTS2}}$  - allocated to P72 and P73 cannot be used. In connection with this, UART2 solely as UART of the internal clock can be used. And UART2 must be used by setting the  $\overline{\text{CTS}}/\overline{\text{RTS}}$  disable bit (bit 4 at address 037C16) to "1".
- (f) The input pin TB1IN of timer B1 allocated to P91 cannot be used. With timer B1 under this state, use only timer mode or the internal event count.
- (g) The input pin SIN3 of serial I/O3 allocated to P91 cannot be used. In connection with this, use serial I/O3 as a serial I/O exclusive to transmission.
- (h) The output pins for three-phase motor control allocated to P72 to P75 cannot be used. So set to 0 (ordinary mode) the mode select bit (bit 2) of three-phase PWM control register 0.



## Pin Description

### Pin Description

Pin name	Signal name	I/O	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	I	This pin switches between processor modes. Connect it to the Vss pin.
(BYTE)	External data bus width select input	I	This pin is connected to CNVss in microcomputer. Connect this pin to Vss.
RESET	Reset input	I	An "L" on this input resets the microcomputer.
XIN XOUT	Clock input Clock output	I O	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 this pin to Vcc.
AVss	Analog power supply input		This pin is a power supply input for the A-D converter. Connect this pin to Vss.
VREF	Reference voltage input	I	This pin is a reference voltage input for the A-D converter.
P00 to P07	I/O port P0	I/O	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.
P20 to P27	I/O port P2	I/O	This is an 8-bit I/O port equivalent to P0.
P30 to P37	I/O port P3	I/O	This is an 8-bit I/O port equivalent to P0.
P40 to P43	I/O port P4	I/O	This is a 4-bit I/O port equivalent to P0.
P50 to P57	I/O port P5	I/O	This is an 8-bit I/O port equivalent to P0. In single-chip mode, P57 in this port outputs a divide-by-8 or divide-by-32 clock of XIN or a clock of the same frequency as XCIN as selected by software.
P60 to P67	I/O port P6	I/O	This is an 8-bit I/O port equivalent to P0. Pins in this port also function as UART0 and UART1 I/O pins as selected by software.
P70, P71, P76, P77	I/O port P7	I/O	This is a 4-bit I/O port equivalent to P0 (P70 and P71 are N channel open-drain output). Pins in this port also function as timer A0–A3, timer B5 or UART2 I/O pins as selected by software.
P80 to P84, P86,P87, P85	I/O port P8  I/O port P85	I/O  I	P80 to P84, P86, and P87 are I/O ports with the same functions as P0. Using software, they can be made to function as the I/O pins for timer A4 and the input pins for external interrupts. P86 and P87 can be set using software to function as the I/O pins for a sub clock generation circuit. In this case, connect a quartz oscillator between P86 (XCOUT pin) and P87 (XCIN pin). P85 is an input-only port that also functions for NMI. The NMI interrupt is generated when the input at this pin changes from "H" to "L". The NMI function cannot be cancelled using software. The pull-up cannot be set for this pin.

## Pin Description

---

### Pin Description

Pin name	Signal name	I/O	Function
P90, P92 to P97	I/O port P9	I/O	This is an 7-bit I/O port equivalent to P0. Pins in this port also function as SI/O3, 4 I/O pins, Timer B0–B4 input pins, D-A converter output pins, A-D converter extended input pins, or A-D trigger input pins as selected by software.
P100 to P107	I/O port P10	I/O	This is an 8-bit I/O port equivalent to P0. Pins in this port also function as A-D converter input pins. Furthermore, P104–P107 also function as input pins for the key input interrupt function.

Note: Memory expansion mode and microprocessor mode are not be supported.

## Operation of Functional Blocks

The M16C/62A (80-pin version) group 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, D-A converter, DMAC, CRC calculation circuit, A-D converter, and I/O ports.

The following explains each unit.

## Memory

Figure 1.4.1 is a memory map of the M16C/62A (80-pin version) group. The address space extends the 1M bytes from address  $00000_{16}$  to  $FFFFFF_{16}$ . From  $FFFFFF_{16}$  down is ROM. For example, in the M30623MCA-XXXGP, there is 128K bytes of internal ROM from  $E0000_{16}$  to  $FFFFFF_{16}$ . The vector table for fixed interrupts such as the reset and  $\overline{\text{NMI}}$  are mapped to  $FFFDC_{16}$  to  $FFFFFF_{16}$ . 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  $00400_{16}$  up is RAM. For example, in the M30623MCA-XXXGP, 5K bytes of internal RAM is mapped to the space from  $00400_{16}$  to  $017FF_{16}$ . 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  $00000_{16}$  to  $003FF_{16}$ . This area accommodates the control registers for peripheral devices such as I/O ports, A-D converter, serial I/O, and timers, etc. Figures 1.7.1 to 1.7.3 are location of peripheral unit control registers. 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  $FFE00_{16}$  to  $FFFDB_{16}$ . 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.

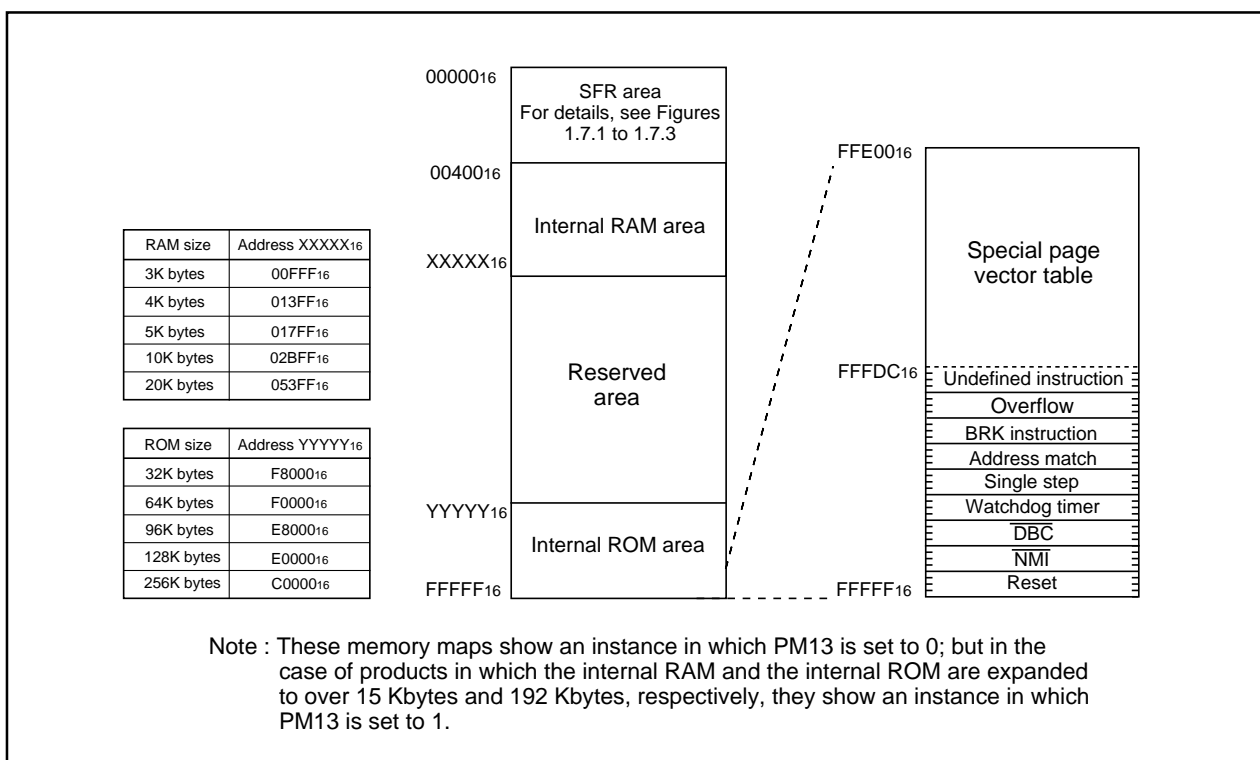


Figure 1.4.1. Memory map

## Central Processing Unit (CPU)

The CPU has a total of 13 registers shown in Figure 1.5.1. Seven of these registers (R0, R1, R2, R3, A0, A1, and FB) come in two sets; therefore, these have two register banks.

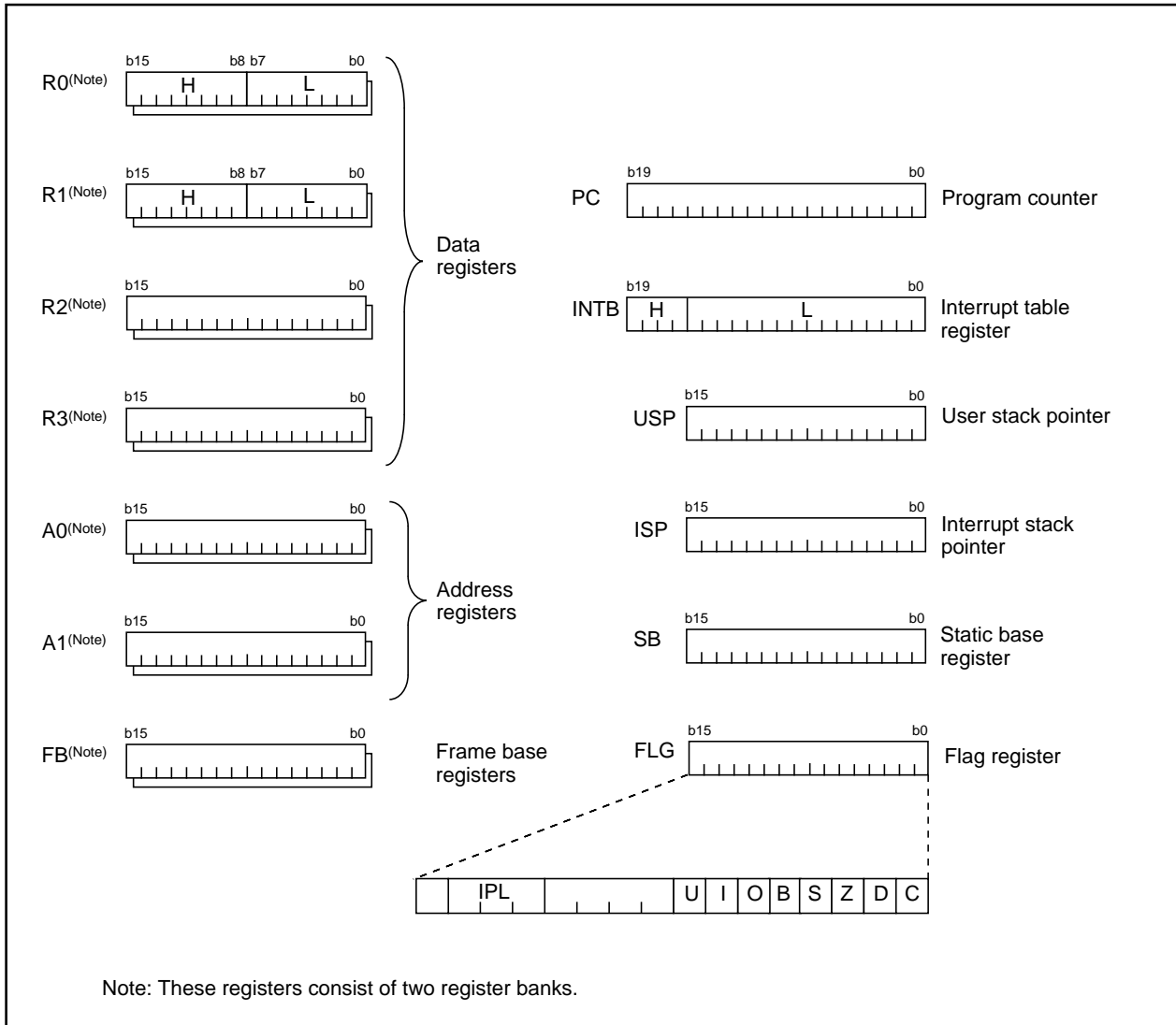


Figure 1.5.1. Central processing unit register

### (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 1.5.2 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.

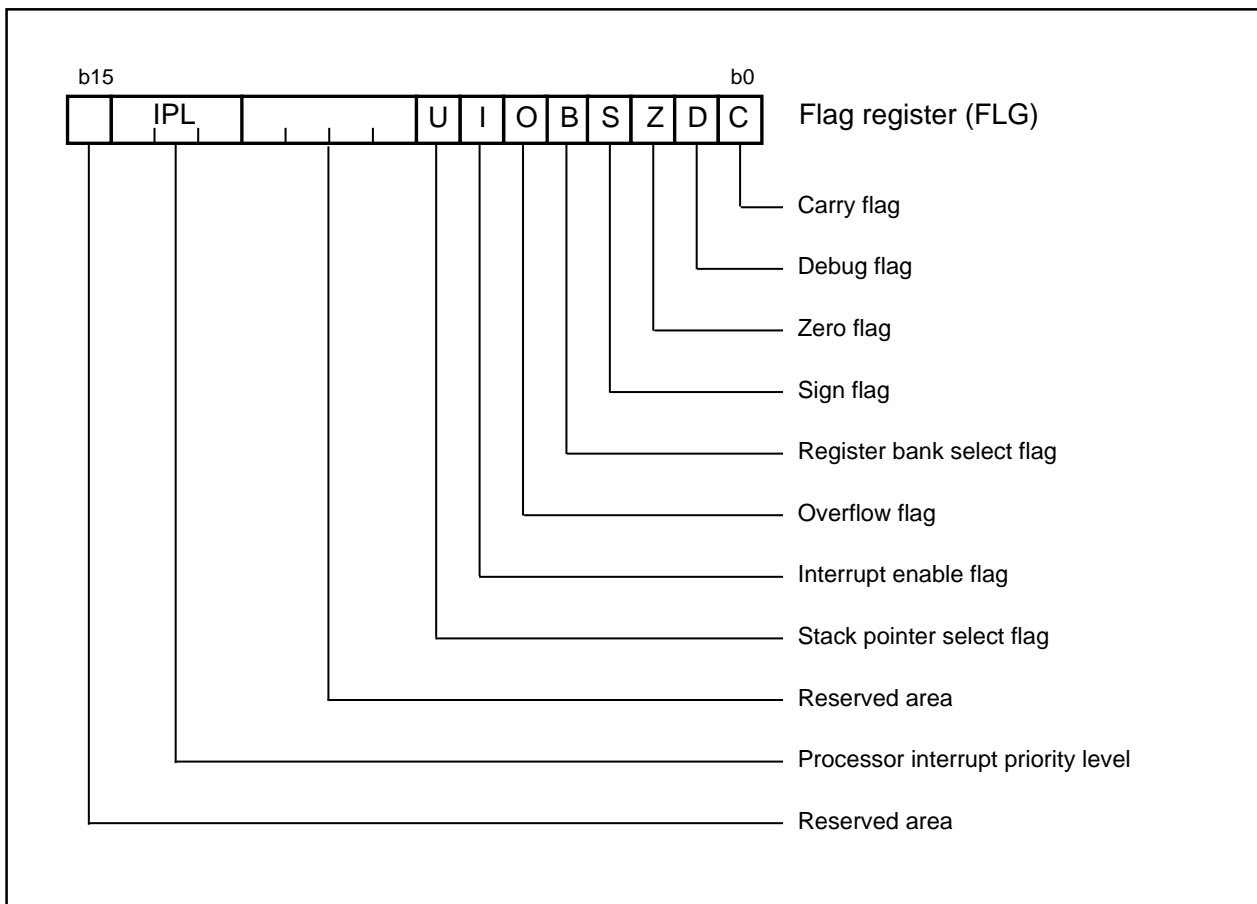


Figure 1.5.2. Flag register (FLG)

## Reset

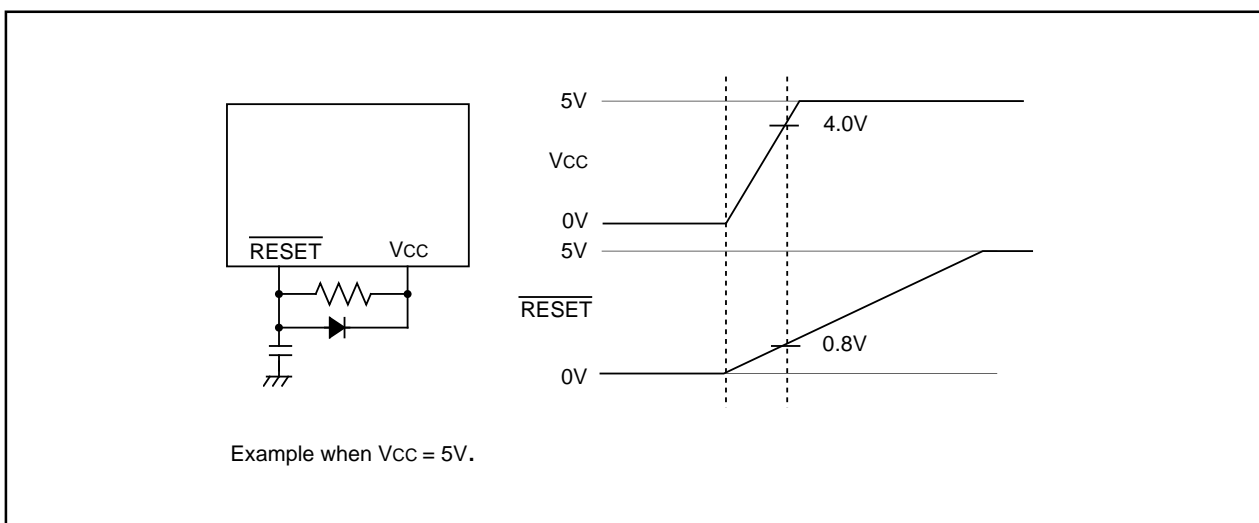
### Reset

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.

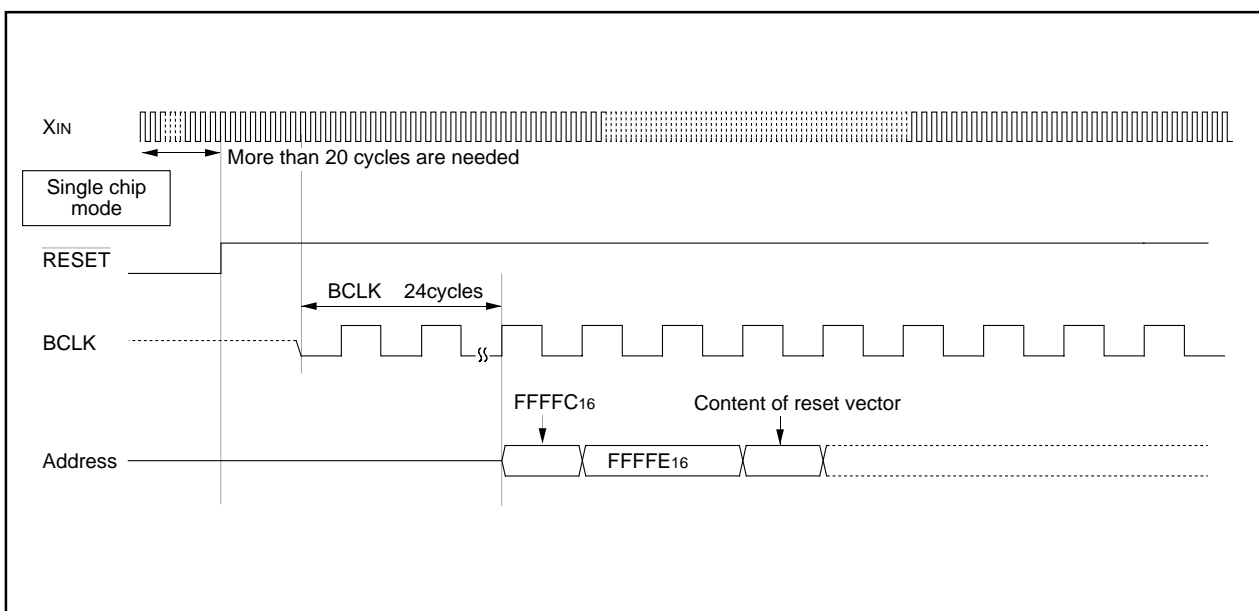
When the supply voltage is in the range where operation is guaranteed, a reset is effected by holding the reset pin level "L" (0.2V<sub>CC</sub> max.) for at least 20 cycles. When the reset pin level is then returned to the "H" level while main clock is stable, the reset status is cancelled and program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

The RAM is undefined at power on. The initial values must therefore be set. When a reset signal is applied while the CPU is writing a value to the RAM, the value may be set as unknown due to the termination of the CPU access.

Figure 1.6.1 shows the example reset circuit. Figure 1.6.2 shows the reset sequence.



**Figure 1.6.1. Example reset circuit**



**Figure 1.6.2. Reset sequence**

## Reset

---

Table 1.6.1 shows the statuses of the other pins while the  $\overline{\text{RESET}}$  pin level is "L". Figures 1.6.3 and 1.6.4 show the internal status of the microcomputer immediately after the reset is cancelled.

**Table 1.6.1. Pin status when  $\overline{\text{RESET}}$  pin level is "L"**

Pin name	Status
	CNVss = Vss
P0, P2, P3, P40 to P43, P5, P6, P70, P71, P76, P77, P80 to P84, P86, P87, P90, P92 to P97, P10	Input port (floating)



Reset

(1) Processor mode register 0	(0004 <sub>16</sub> )...	00 <sub>16</sub>	(28) UART1 transmit interrupt control register	(0053 <sub>16</sub> )...	XXXXXXXX?000
(2) Processor mode register 1	(0005 <sub>16</sub> )...	00000XXXX	(29) UART1 receive interrupt control register	(0054 <sub>16</sub> )...	XXXXXXXX?000
(3) System clock control register 0	(0006 <sub>16</sub> )...	01001000	(30) Timer A0 interrupt control register	(0055 <sub>16</sub> )...	XXXXXXXX?000
(4) System clock control register 1	(0007 <sub>16</sub> )...	001010000	(31) Timer A1 interrupt control register	(0056 <sub>16</sub> )...	XXXXXXXX?000
(5) Chip select control register	(0008 <sub>16</sub> )...	000000001	(32) Timer A2 interrupt control register	(0057 <sub>16</sub> )...	XXXXXXXX?000
(6) Address match interrupt enable register	(0009 <sub>16</sub> )...	XXXXXXXX00	(33) Timer A3 interrupt control register	(0058 <sub>16</sub> )...	XXXXXXXX?000
(7) Protect register	(000A <sub>16</sub> )...	XXXXXXXX00	(34) Timer A4 interrupt control register	(0059 <sub>16</sub> )...	XXXXXXXX?000
(8) Watchdog timer control register	(000F <sub>16</sub> )...	000???	(35) Timer B0 interrupt control register	(005A <sub>16</sub> )...	XXXXXXXX?000
(9) Address match interrupt register 0	(0010 <sub>16</sub> )...	00 <sub>16</sub>	(36) Timer B1 interrupt control register	(005B <sub>16</sub> )...	XXXXXXXX?000
	(0011 <sub>16</sub> )...	00 <sub>16</sub>	(37) Timer B2 interrupt control register	(005C <sub>16</sub> )...	XXXXXXXX?000
	(0012 <sub>16</sub> )...	XXXXXXXX0000	(38) INT0 interrupt control register	(005D <sub>16</sub> )...	XXXX00?000
(10) Address match interrupt register 1	(0014 <sub>16</sub> )...	00 <sub>16</sub>	(39) INT1 interrupt control register	(005E <sub>16</sub> )...	XXXX00?000
	(0015 <sub>16</sub> )...	00 <sub>16</sub>	(40) INT2 interrupt control register	(005F <sub>16</sub> )...	XXXX00?000
	(0016 <sub>16</sub> )...	XXXXXXXX0000	(41) Timer B3,4,5 count start flag	(0340 <sub>16</sub> )...	000XXXXXXXX
(11) DMA0 control register	(002C <sub>16</sub> )...	000000?00	(42) Three-phase PWM control register 0	(0348 <sub>16</sub> )...	00 <sub>16</sub>
(12) DMA1 control register	(003C <sub>16</sub> )...	000000?00	(43) Three-phase PWM control register 1	(0349 <sub>16</sub> )...	00 <sub>16</sub>
(13) INT3 interrupt control register	(0044 <sub>16</sub> )...	XXXX00?000	(44) Three-phase output buffer register 0	(034A <sub>16</sub> )...	00 <sub>16</sub>
(14) Timer B5 interrupt control register	(0045 <sub>16</sub> )...	XXXXX?0000	(45) Three-phase output buffer register 1	(034B <sub>16</sub> )...	00 <sub>16</sub>
(15) Timer B4 interrupt control register	(0046 <sub>16</sub> )...	XXXXX?0000	(46) Timer B3 mode register	(035B <sub>16</sub> )...	00?00000
(16) Timer B3 interrupt control register	(0047 <sub>16</sub> )...	XXXXX?0000	(47) Timer B4 mode register	(035C <sub>16</sub> )...	00?00000
(17) SI/O4 interrupt control register	(0048 <sub>16</sub> )...	XXXX00?000	(48) Timer B5 mode register	(035D <sub>16</sub> )...	00?00000
(18) SI/O3 interrupt control register	(0049 <sub>16</sub> )...	XXXX00?000	(49) Interrupt cause select register	(035F <sub>16</sub> )...	00 <sub>16</sub>
(19) Bus collision detection interrupt control register	(004A <sub>16</sub> )...	XXXXX?0000	(50) SI/O3 control register	(0362 <sub>16</sub> )...	40 <sub>16</sub>
(20) DMA0 interrupt control register	(004B <sub>16</sub> )...	XXXXX?0000	(51) SI/O4 control register	(0366 <sub>16</sub> )...	40 <sub>16</sub>
(21) DMA1 interrupt control register	(004C <sub>16</sub> )...	XXXXX?0000	(52) UART2 special mode register 3 (Note)	(0375 <sub>16</sub> )...	?
(22) Key input interrupt control register	(004D <sub>16</sub> )...	XXXXX?0000	(53) UART2 special mode register 2	(0376 <sub>16</sub> )...	00 <sub>16</sub>
(23) A-D conversion interrupt control register	(004E <sub>16</sub> )...	XXXXX?0000	(54) UART2 special mode register	(0377 <sub>16</sub> )...	00 <sub>16</sub>
(24) UART2 transmit interrupt control register	(004F <sub>16</sub> )...	XXXXX?0000	(55) UART2 transmit/receive mode register	(0378 <sub>16</sub> )...	00 <sub>16</sub>
(25) UART2 receive interrupt control register	(0050 <sub>16</sub> )...	XXXXX?0000	(56) UART2 transmit/receive control register 0	(037C <sub>16</sub> )...	00001000
(26) UART0 transmit interrupt control register	(0051 <sub>16</sub> )...	XXXXX?0000	(57) UART2 transmit/receive control register 1	(037D <sub>16</sub> )...	00000010
(27) UART0 receive interrupt control register	(0052 <sub>16</sub> )...	XXXXX?0000			

x : Nothing is mapped to this bit  
 ? : Undefined

The content of other registers are undefined when the microcomputer is reset. The initial values must therefore be set. The RAM is undefined at power on. The initial values must therefore be set. When a reset signal is applied while the CPU is writing a value to the RAM, the value may be set as unknown due to the termination of the CPU access.

Note: "0016" is read out when set bit 7 (SDDS) of the UART2 special mode register (address 0377<sub>16</sub>) to "1".

Figure 1.6.3. Device's internal status after a reset is cleared

Reset

(58) Count start flag	(0380 <sub>16</sub> )...	00 <sub>16</sub>	(84) A-D control register 1	(03D7 <sub>16</sub> )...	00 <sub>16</sub>
(59) Clock prescaler reset flag	(0381 <sub>16</sub> )...	0XXXXXX	(85) D-A control register	(03DC <sub>16</sub> )...	00 <sub>16</sub>
(60) One-shot start flag	(0382 <sub>16</sub> )...	00XX0000	(86) Port P0 direction register	(03E2 <sub>16</sub> )...	00 <sub>16</sub>
(61) Trigger select flag	(0383 <sub>16</sub> )...	00 <sub>16</sub>	(87) Port P1 direction register	(03E3 <sub>16</sub> )...	00 <sub>16</sub>
(62) Up-down flag	(0384 <sub>16</sub> )...	?? ? 0000	(88) Port P2 direction register	(03E6 <sub>16</sub> )...	00 <sub>16</sub>
(63) Timer A0 mode register	(0396 <sub>16</sub> )...	00 <sub>16</sub>	(89) Port P3 direction register	(03E7 <sub>16</sub> )...	00 <sub>16</sub>
(64) Timer A1 mode register	(0397 <sub>16</sub> )...	00 <sub>16</sub>	(90) Port P4 direction register	(03EA <sub>16</sub> )...	00 <sub>16</sub>
(65) Timer A2 mode register	(0398 <sub>16</sub> )...	00 <sub>16</sub>	(91) Port P5 direction register	(03EB <sub>16</sub> )...	00 <sub>16</sub>
(66) Timer A3 mode register	(0399 <sub>16</sub> )...	00 <sub>16</sub>	(92) Port P6 direction register	(03EE <sub>16</sub> )...	00 <sub>16</sub>
(67) Timer A4 mode register	(039A <sub>16</sub> )...	00 <sub>16</sub>	(93) Port P7 direction register	(03EF <sub>16</sub> )...	00 <sub>16</sub>
(68) Timer B0 mode register	(039B <sub>16</sub> )...	00??0000	(94) Port P8 direction register	(03F2 <sub>16</sub> )...	00XX0000
(69) Timer B1 mode register	(039C <sub>16</sub> )...	00?X0000	(95) Port P9 direction register	(03F3 <sub>16</sub> )...	00 <sub>16</sub>
(70) Timer B2 mode register	(039D <sub>16</sub> )...	00?X0000	(96) Port P10 direction register	(03F6 <sub>16</sub> )...	00 <sub>16</sub>
(71) UART0 transmit/receive mode register	(03A0 <sub>16</sub> )...	00 <sub>16</sub>	(97) Pull-up control register 0	(03FC <sub>16</sub> )...	00 <sub>16</sub>
(72) UART0 transmit/receive control register 0	(03A4 <sub>16</sub> )...	0000010000	(98) Pull-up control register 1	(03FD <sub>16</sub> )...	00 <sub>16</sub>
(73) UART0 transmit/receive control register 1	(03A5 <sub>16</sub> )...	0000000100	(99) Pull-up control register 2	(03FE <sub>16</sub> )...	00 <sub>16</sub>
(74) UART1 transmit/receive mode register	(03A8 <sub>16</sub> )...	00 <sub>16</sub>	(100) Port control register	(03FF <sub>16</sub> )...	00 <sub>16</sub>
(75) UART1 transmit/receive control register 0	(03AC <sub>16</sub> )...	0000010000	(101) Data registers (R0/R1/R2/R3)		0000 <sub>16</sub>
(76) UART1 transmit/receive control register 1	(03AD <sub>16</sub> )...	0000000100	(102) Address registers (A0/A1)		0000 <sub>16</sub>
(77) UART transmit/receive control register 2	(03B0 <sub>16</sub> )...	XX00000000	(103) Frame base register (FB)		0000 <sub>16</sub>
(78) Flash memory control register 1 (Note)	(03B6 <sub>16</sub> )...	?? ? ? 0 ? ? ?	(104) Interrupt table register (INTB)		00000 <sub>16</sub>
(79) Flash memory control register 0 (Note)	(03B7 <sub>16</sub> )...	XX00000001	(105) User stack pointer (USP)		0000 <sub>16</sub>
(80) DMA0 cause select register	(03B8 <sub>16</sub> )...	00 <sub>16</sub>	(106) Interrupt stack pointer (ISP)		0000 <sub>16</sub>
(81) DMA1 cause select register	(03BA <sub>16</sub> )...	00 <sub>16</sub>	(107) Static base register (SB)		0000 <sub>16</sub>
(82) A-D control register 2	(03D4 <sub>16</sub> )...	0000XXXX0	(108) Flag register (FLG)		0000 <sub>16</sub>
(83) A-D control register 0	(03D6 <sub>16</sub> )...	000000???			

x : Nothing is mapped to this bit  
 ? : Undefined

The content of other registers are undefined when the microcomputer is reset. The initial values must therefore be set. The RAM is undefined at power on. The initial values must therefore be set. When a reset signal is applied while the CPU is writing a value to the RAM, the value may be set as unknown due to the termination of the CPU access.

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

Figure 1.6.4. Device's internal status after a reset is cleared

## SFR

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>	INT3 interrupt control register (INT3IC)*
0005 <sub>16</sub>	Processor mode register 1 (PM1)	0045 <sub>16</sub>	Timer B5 interrupt control register (TB5IC)
0006 <sub>16</sub>	System clock control register 0 (CM0)	0046 <sub>16</sub>	Timer B4 interrupt control register (TB4IC)
0007 <sub>16</sub>	System clock control register 1 (CM1)	0047 <sub>16</sub>	Timer B3 interrupt control register (TB3IC)
0008 <sub>16</sub>	Reserved register	0048 <sub>16</sub>	SI/O4 interrupt control register (S4IC)
0009 <sub>16</sub>	Address match interrupt enable register (AIER)		INT5 interrupt control register (INT5IC)*
000A <sub>16</sub>	Protect register (PRCR)	0049 <sub>16</sub>	SI/O3 interrupt control register (S3IC)
000B <sub>16</sub>			INT4 interrupt control register (INT4IC)*
000C <sub>16</sub>		004A <sub>16</sub>	Bus collision detection interrupt control register (BCNIC)
000D <sub>16</sub>		004B <sub>16</sub>	DMA0 interrupt control register (DM0IC)
000E <sub>16</sub>	Watchdog timer start register (WDTS)	004C <sub>16</sub>	DMA1 interrupt control register (DM1IC)
000F <sub>16</sub>	Watchdog timer control register (WDC)	004D <sub>16</sub>	Key input interrupt control register (KUPIC)
0010 <sub>16</sub>		004E <sub>16</sub>	A-D conversion interrupt control register (ADIC)
0011 <sub>16</sub>	Address match interrupt register 0 (RMAD0)	004F <sub>16</sub>	UART2 transmit interrupt control register (S2TIC)
0012 <sub>16</sub>		0050 <sub>16</sub>	UART2 receive interrupt control register (S2RIC)
0013 <sub>16</sub>		0051 <sub>16</sub>	UART0 transmit interrupt control register (S0TIC)
0014 <sub>16</sub>		0052 <sub>16</sub>	UART0 receive interrupt control register (S0RIC)
0015 <sub>16</sub>	Address match interrupt register 1 (RMAD1)	0053 <sub>16</sub>	UART1 transmit interrupt control register (S1TIC)
0016 <sub>16</sub>		0054 <sub>16</sub>	UART1 receive interrupt control register (S1RIC)
0017 <sub>16</sub>		0055 <sub>16</sub>	Timer A0 interrupt control register (TA0IC)
0018 <sub>16</sub>		0056 <sub>16</sub>	Timer A1 interrupt control register (TA1IC)
0019 <sub>16</sub>		0057 <sub>16</sub>	Timer A2 interrupt control register (TA2IC)
001A <sub>16</sub>		0058 <sub>16</sub>	Timer A3 interrupt control register (TA3IC)
001B <sub>16</sub>		0059 <sub>16</sub>	Timer A4 interrupt control register (TA4IC)
001C <sub>16</sub>		005A <sub>16</sub>	Timer B0 interrupt control register (TB0IC)
001D <sub>16</sub>		005B <sub>16</sub>	Timer B1 interrupt control register (TB1IC)
001E <sub>16</sub>		005C <sub>16</sub>	Timer B2 interrupt control register (TB2IC)
001F <sub>16</sub>		005D <sub>16</sub>	INT0 interrupt control register (INT0IC)
0020 <sub>16</sub>		005E <sub>16</sub>	INT1 interrupt control register (INT1IC)
0021 <sub>16</sub>	DMA0 source pointer (SAR0)	005F <sub>16</sub>	INT2 interrupt control register (INT2IC)
0022 <sub>16</sub>		0060 <sub>16</sub>	
0023 <sub>16</sub>		0061 <sub>16</sub>	
0024 <sub>16</sub>		0062 <sub>16</sub>	
0025 <sub>16</sub>	DMA0 destination pointer (DAR0)	0063 <sub>16</sub>	
0026 <sub>16</sub>		0064 <sub>16</sub>	
0027 <sub>16</sub>		0065 <sub>16</sub>	
0028 <sub>16</sub>			
0029 <sub>16</sub>	DMA0 transfer counter (TCR0)		
002A <sub>16</sub>			
002B <sub>16</sub>			
002C <sub>16</sub>	DMA0 control register (DM0CON)		
002D <sub>16</sub>			
002E <sub>16</sub>			
002F <sub>16</sub>			
0030 <sub>16</sub>			
0031 <sub>16</sub>	DMA1 source pointer (SAR1)		
0032 <sub>16</sub>			
0033 <sub>16</sub>			
0034 <sub>16</sub>			
0035 <sub>16</sub>	DMA1 destination pointer (DAR1)		
0036 <sub>16</sub>			
0037 <sub>16</sub>			
0038 <sub>16</sub>			
0039 <sub>16</sub>	DMA1 transfer counter (TCR1)		
003A <sub>16</sub>			
003B <sub>16</sub>			
003C <sub>16</sub>	DMA1 control register (DM1CON)		
003D <sub>16</sub>			
003E <sub>16</sub>			
003F <sub>16</sub>			

Note 1: M16C/62A (80-pin version) group is not provided with the functions, in whole or in part, of the registers marked with an \*. But the relevant registers need to be dealt with as given on page 7.

Note 2: Locations in the SFR area where nothing is allocated are reserved areas. Do not access these areas for read or write.

Figure 1.7.1. Location of peripheral unit control registers (1)

0340 <sub>16</sub>	Timer B3, 4, 5 count start flag (TBSR)	0380 <sub>16</sub>	Count start flag (TABSRS)
0341 <sub>16</sub>		0381 <sub>16</sub>	Clock prescaler reset flag (CPSRF)
0342 <sub>16</sub>	Timer A1-1 register (TA11)	0382 <sub>16</sub>	One-shot start flag (ONSF)
0343 <sub>16</sub>		0383 <sub>16</sub>	Trigger select register (TRGSR)
0344 <sub>16</sub>	Timer A2-1 register (TA21)	0384 <sub>16</sub>	Up-down flag (UDF)
0345 <sub>16</sub>		0385 <sub>16</sub>	
0346 <sub>16</sub>	Timer A4-1 register (TA41)	0386 <sub>16</sub>	Timer A0 register (TA0)
0347 <sub>16</sub>		0387 <sub>16</sub>	
0348 <sub>16</sub>	Three-phase PWM control register 0(INVC0)	0388 <sub>16</sub>	Timer A1 register (TA1)
0349 <sub>16</sub>	Three-phase PWM control register 1(INVC1)	0389 <sub>16</sub>	
034A <sub>16</sub>	Three-phase output buffer register 0(IDB0)	038A <sub>16</sub>	Timer A2 register (TA2)
034B <sub>16</sub>	Three-phase output buffer register 1(IDB1)	038B <sub>16</sub>	
034C <sub>16</sub>	Dead time timer(DTT)	038C <sub>16</sub>	Timer A3 register (TA3)
034D <sub>16</sub>	Timer B2 interrupt occurrence frequency set counter(ICTB2)	038D <sub>16</sub>	
034E <sub>16</sub>		038E <sub>16</sub>	Timer A4 register (TA4)
034F <sub>16</sub>		038F <sub>16</sub>	
0350 <sub>16</sub>	Timer B3 register (TB3)	0390 <sub>16</sub>	Timer B0 register (TB0)
0351 <sub>16</sub>			
0352 <sub>16</sub>	Timer B4 register (TB4)	0392 <sub>16</sub>	Timer B1 register (TB1)
0353 <sub>16</sub>			
0354 <sub>16</sub>	Timer B5 register (TB5)	0394 <sub>16</sub>	Timer B2 register (TB2)
0355 <sub>16</sub>			
0356 <sub>16</sub>		0396 <sub>16</sub>	Timer A0 mode register (TA0MR)
0357 <sub>16</sub>		0397 <sub>16</sub>	Timer A1 mode register (TA1MR)
0358 <sub>16</sub>		0398 <sub>16</sub>	Timer A2 mode register (TA2MR)
0359 <sub>16</sub>		0399 <sub>16</sub>	Timer A3 mode register (TA3MR)
035A <sub>16</sub>		039A <sub>16</sub>	Timer A4 mode register (TA4MR)
035B <sub>16</sub>	Timer B3 mode register (TB3MR)	039B <sub>16</sub>	Timer B0 mode register (TB0MR)
035C <sub>16</sub>	Timer B4 mode register (TB4MR)	039C <sub>16</sub>	Timer B1 mode register (TB1MR)
035D <sub>16</sub>	Timer B5 mode register (TB5MR)	039D <sub>16</sub>	Timer B2 mode register (TB2MR)
035E <sub>16</sub>		039E <sub>16</sub>	
035F <sub>16</sub>	Interrupt cause select register (IFSR)	039F <sub>16</sub>	
0360 <sub>16</sub>	SI/O3 transmit/receive register (S3TRR)	03A0 <sub>16</sub>	UART0 transmit/receive mode register (U0MR)
0361 <sub>16</sub>		03A1 <sub>16</sub>	UART0 bit rate generator (U0BRG)
0362 <sub>16</sub>	SI/O3 control register (S3C)	03A2 <sub>16</sub>	UART0 transmit buffer register (U0TB)
0363 <sub>16</sub>	SI/O3 bit rate generator (S3BRG)	03A3 <sub>16</sub>	
0364 <sub>16</sub>	SI/O4 transmit/receive register (S4TRR)	03A4 <sub>16</sub>	UART0 transmit/receive control register 0 (U0C0)
0365 <sub>16</sub>		03A5 <sub>16</sub>	UART0 transmit/receive control register 1 (U0C1)
0366 <sub>16</sub>	SI/O4 control register (S4C)	03A6 <sub>16</sub>	UART0 receive buffer register (U0RB)
0367 <sub>16</sub>	SI/O4 bit rate generator (S4BRG)	03A7 <sub>16</sub>	
0368 <sub>16</sub>		03A8 <sub>16</sub>	UART1 transmit/receive mode register (U1MR)
0369 <sub>16</sub>		03A9 <sub>16</sub>	UART1 bit rate generator (U1BRG)
036A <sub>16</sub>		03AA <sub>16</sub>	UART1 transmit buffer register (U1TB)
036B <sub>16</sub>		03AB <sub>16</sub>	
036C <sub>16</sub>		03AC <sub>16</sub>	UART1 transmit/receive control register 0 (U1C0)
036D <sub>16</sub>		03AD <sub>16</sub>	UART1 transmit/receive control register 1 (U1C1)
036E <sub>16</sub>		03AE <sub>16</sub>	UART1 receive buffer register (U1RB)
036F <sub>16</sub>		03AF <sub>16</sub>	
0370 <sub>16</sub>		03B0 <sub>16</sub>	UART transmit/receive control register 2 (UCON)
0371 <sub>16</sub>		03B1 <sub>16</sub>	
0372 <sub>16</sub>		03B2 <sub>16</sub>	
0373 <sub>16</sub>		03B3 <sub>16</sub>	
0374 <sub>16</sub>		03B4 <sub>16</sub>	
0375 <sub>16</sub>	UART2 special mode register 3 (U2SMR3)	03B5 <sub>16</sub>	
0376 <sub>16</sub>	UART2 special mode register 2 (U2SMR2)	03B6 <sub>16</sub>	Flash memory control register 1 (FMR1) (Note1)
0377 <sub>16</sub>	UART2 special mode register (U2SMR)	03B7 <sub>16</sub>	Flash memory control register 0 (FMR0) (Note1)
0378 <sub>16</sub>	UART2 transmit/receive mode register (U2MR)	03B8 <sub>16</sub>	DMA0 request cause select register (DM0SL)
0379 <sub>16</sub>	UART2 bit rate generator (U2BRG)	03B9 <sub>16</sub>	
037A <sub>16</sub>	UART2 transmit buffer register (U2TB)	03BA <sub>16</sub>	DMA1 request cause select register (DM1SL)
037B <sub>16</sub>			03BB <sub>16</sub>
037C <sub>16</sub>	UART2 transmit/receive control register 0 (U2C0)	03BC <sub>16</sub>	CRC data register (CRCD)
037D <sub>16</sub>	UART2 transmit/receive control register 1 (U2C1)	03BD <sub>16</sub>	
037E <sub>16</sub>	UART2 receive buffer register (U2RB)	03BE <sub>16</sub>	CRC input register (CRCIN)
037F <sub>16</sub>			

Note 1 : This register is only exist in flash memory version.  
Note 2 : Locations in the SFR area where nothing is allocated are reserved areas. Do not access these areas for read or write.

Figure 1.7.2. Location of peripheral unit control registers (2)

03C0 <sub>16</sub>	A-D register 0 (AD0)
03C1 <sub>16</sub>	
03C2 <sub>16</sub>	A-D register 1 (AD1)
03C3 <sub>16</sub>	
03C4 <sub>16</sub>	A-D register 2 (AD2)
03C5 <sub>16</sub>	
03C6 <sub>16</sub>	A-D register 3 (AD3)
03C7 <sub>16</sub>	
03C8 <sub>16</sub>	A-D register 4 (AD4)
03C9 <sub>16</sub>	
03CA <sub>16</sub>	A-D register 5 (AD5)
03CB <sub>16</sub>	
03CC <sub>16</sub>	A-D register 6 (AD6)
03CD <sub>16</sub>	
03CE <sub>16</sub>	A-D register 7 (AD7)
03CF <sub>16</sub>	
03D0 <sub>16</sub>	
03D1 <sub>16</sub>	
03D2 <sub>16</sub>	
03D3 <sub>16</sub>	
03D4 <sub>16</sub>	A-D control register 2 (ADCON2)
03D5 <sub>16</sub>	
03D6 <sub>16</sub>	A-D control register 0 (ADCON0)
03D7 <sub>16</sub>	A-D control register 1 (ADCON1)
03D8 <sub>16</sub>	D-A register 0 (DA0)
03D9 <sub>16</sub>	
03DA <sub>16</sub>	D-A register 1 (DA1)
03DB <sub>16</sub>	
03DC <sub>16</sub>	D-A control register (DACON)
03DD <sub>16</sub>	
03DE <sub>16</sub>	
03DF <sub>16</sub>	
03E0 <sub>16</sub>	Port P0 register (P0)
03E1 <sub>16</sub>	Port P1 register (P1) *
03E2 <sub>16</sub>	Port P0 direction register (PD0)
03E3 <sub>16</sub>	Port P1 direction register (PD1) *
03E4 <sub>16</sub>	Port P2 register (P2)
03E5 <sub>16</sub>	Port P3 register (P3)
03E6 <sub>16</sub>	Port P2 direction register (PD2)
03E7 <sub>16</sub>	Port P3 direction register (PD3)
03E8 <sub>16</sub>	Port P4 register (P4) *
03E9 <sub>16</sub>	Port P5 register (P5)
03EA <sub>16</sub>	Port P4 direction register (PD4) *
03EB <sub>16</sub>	Port P5 direction register (PD5)
03EC <sub>16</sub>	Port P6 register (P6)
03ED <sub>16</sub>	Port P7 register (P7) *
03EE <sub>16</sub>	Port P6 direction register (PD6)
03EF <sub>16</sub>	Port P7 direction register (PD7) *
03F0 <sub>16</sub>	Port P8 register (P8)
03F1 <sub>16</sub>	Port P9 register (P9) *
03F2 <sub>16</sub>	Port P8 direction register (PD8)
03F3 <sub>16</sub>	Port P9 direction register (PD9) *
03F4 <sub>16</sub>	Port P10 register (P10)
03F5 <sub>16</sub>	
03F6 <sub>16</sub>	Port P10 direction register (PD10)
03F7 <sub>16</sub>	
03F8 <sub>16</sub>	
03F9 <sub>16</sub>	
03FA <sub>16</sub>	
03FB <sub>16</sub>	
03FC <sub>16</sub>	Pull-up control register 0 (PUR0)
03FD <sub>16</sub>	Pull-up control register 1 (PUR1)
03FE <sub>16</sub>	Pull-up control register 2 (PUR2)
03FF <sub>16</sub>	Port control register (PCR)

Note 1: M16C/62A (80-pin version) group is not provided with the functions, in whole or in part, of the registers marked with an \*. But the relevant registers need to be dealt with as given on page 7.  
 Note 2: Locations in the SFR area where nothing is allocated are reserved areas. Do not access these areas for read or write.

**Figure 1.7.3. Location of peripheral unit control registers (3)**

## 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.

### Processor Mode

#### Single-chip mode

M16C/62A (80-pin version) group support single-chip mode only.

In single-chip mode, only internal memory space (SFR, internal RAM, and internal ROM) can be accessed. Ports P0 to P10 can be used as programmable I/O ports or as I/O ports for the internal peripheral functions.

Figure 1.8.1 shows the processor mode registers 0 and 1.

Figure 1.8.2 shows the memory map.

Processor Mode

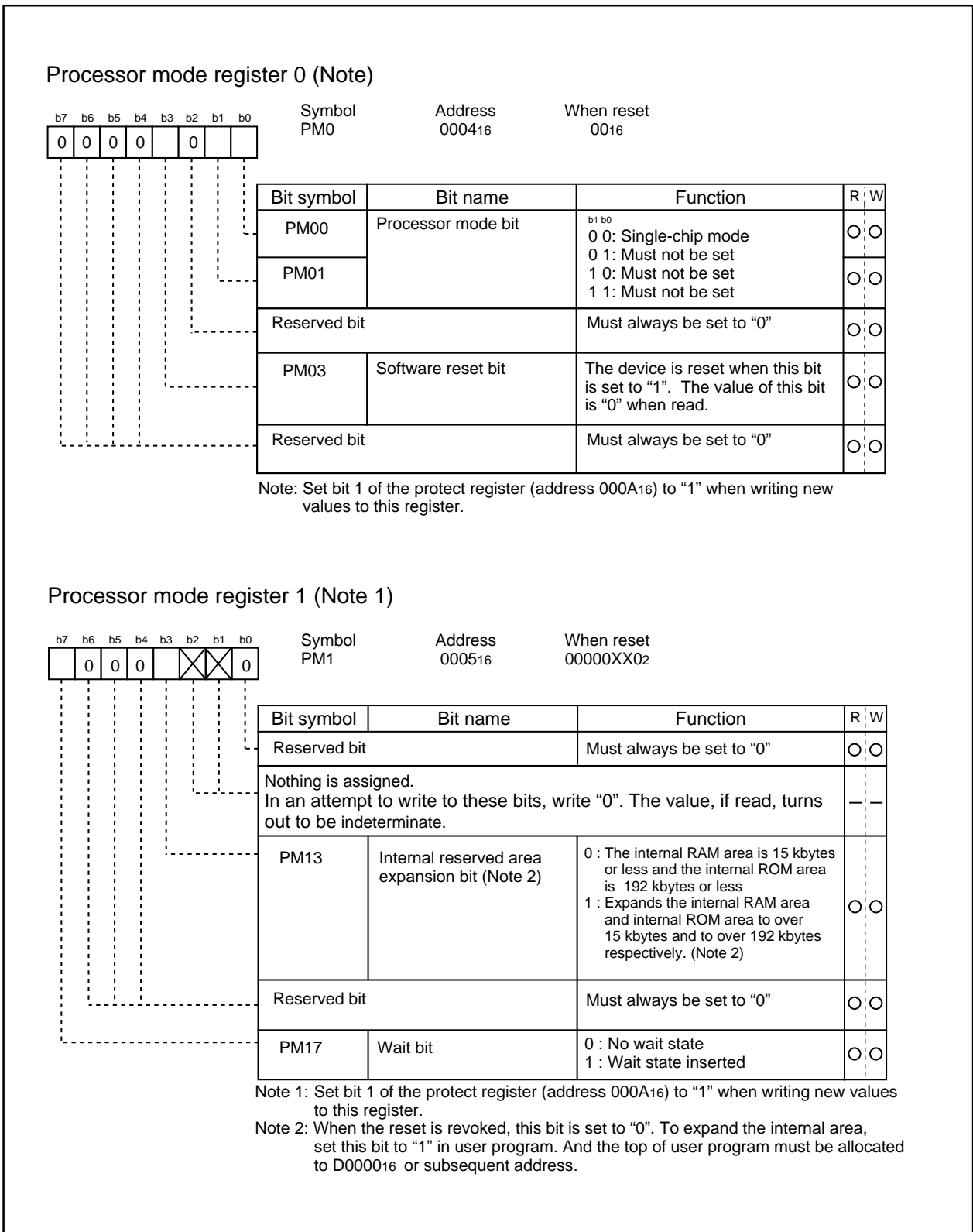
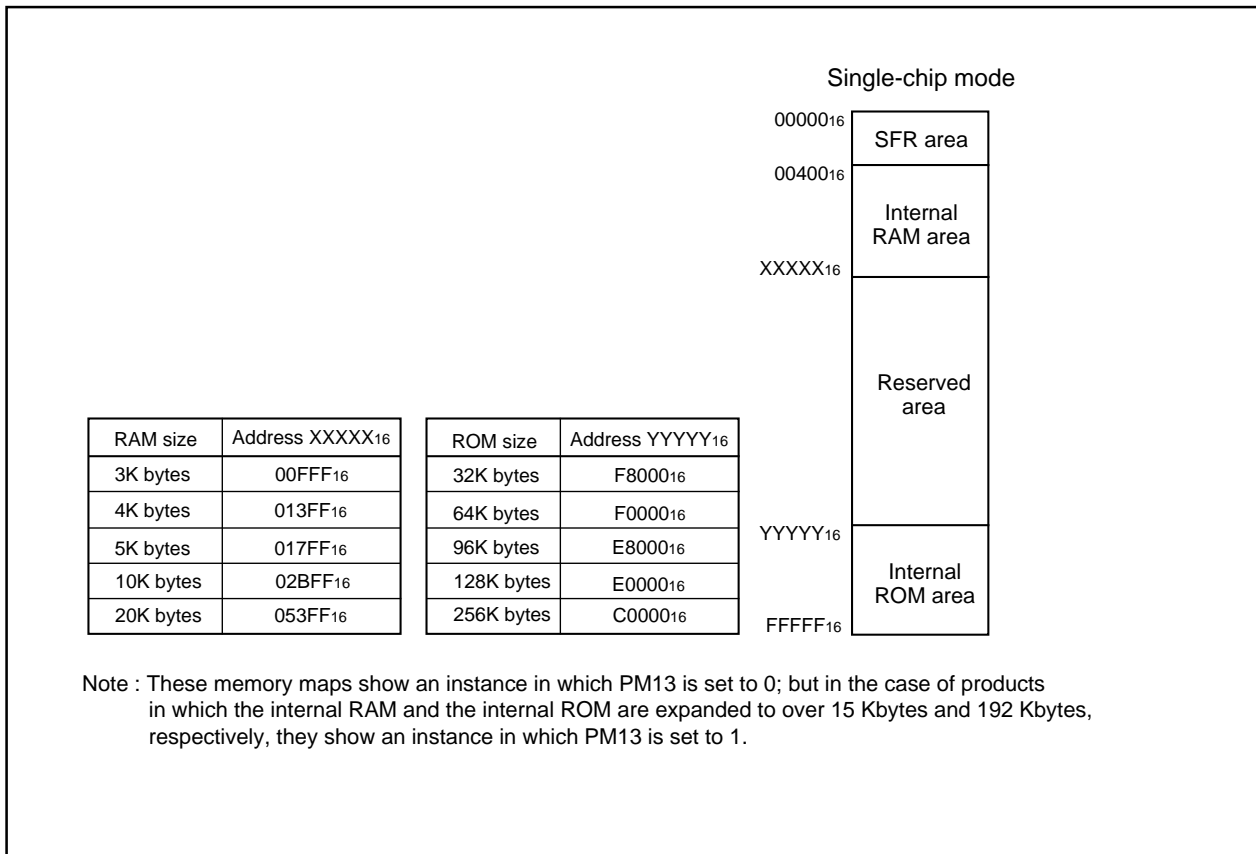


Figure 1.8.1. Processor mode registers 0 and 1



**Figure 1.8.2. Memory map**

### Internal Reserved Area Expansion Bit (PM13)

This bit expands the internal RAM area and the internal ROM area, and changes the chip select area. In M30625MGA/FGA, for example, to set this bit to “1” expands the internal RAM area and the internal ROM area to 20 Kbytes and 256 Kbytes respectively. When the reset is revoked, this bit is set to “0”. To expand the internal area, set this bit to “1” in user program. And the top of user program must be allocated to D0000<sub>16</sub> or subsequent address.

In the case of the product in which the internal ROM is 192 Kbytes or less and the internal RAM is 15 Kbytes or less, set this bit to “0”. The internal area is not expanded and any action is not affected, even if this bit is set to “1”.



## Software Wait

### Software wait

A software wait can be inserted by setting the wait bit (bit 7) of the processor mode register 1 (address 000516) (Note).

A software wait is inserted in the internal ROM/RAM area by setting the wait bit of the processor mode register 1. When set to "0", each bus cycle is executed in one BCLK cycle. When set to "1", each bus cycle is executed in two BCLK cycles. After the microcomputer has been reset, this bit defaults to "0". Set this bit after referring to the recommended operating conditions (main clock input oscillation frequency) of the electric characteristics.

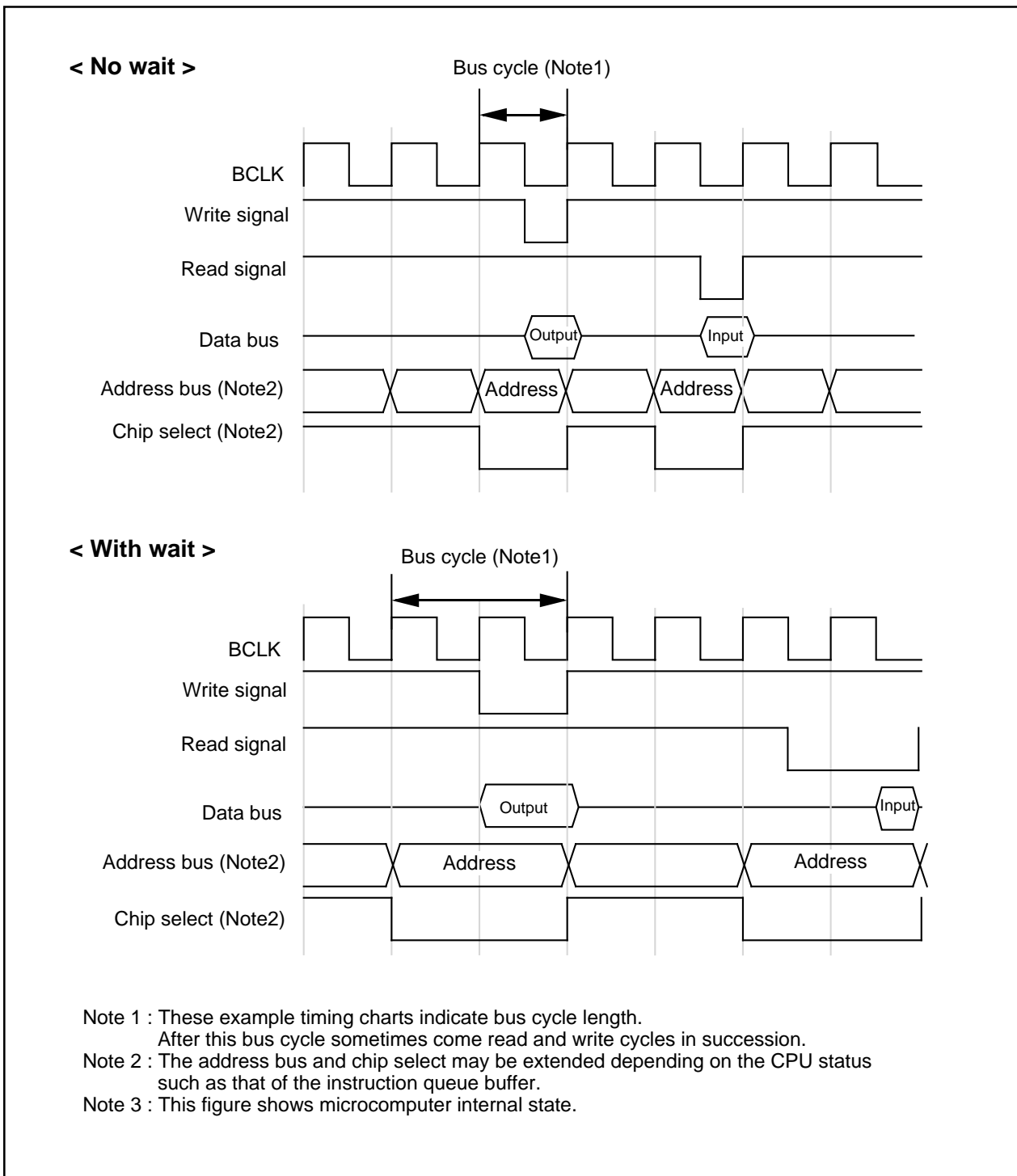
The SFR area is always accessed in two BCLK cycles regardless of the setting of this control bit.

Table 1.8.1 shows the software wait and bus cycles. Figure 1.8.3 shows example bus timing when using software waits.

Note: Before attempting to change the contents of the processor mode register 1, set bit 1 of the protect register (address 000A16) to "1".

**Table 1.8.1. 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



**Figure 1.8.3. Typical bus timings using software wait**

## Clock Generating Circuit

### Clock Generating Circuit

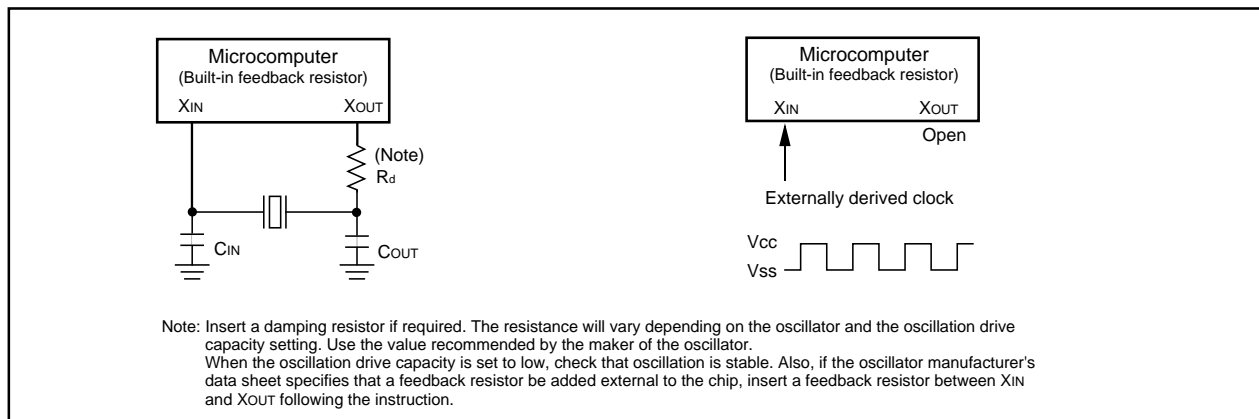
The clock generating circuit contains two oscillator circuits that supply the operating clock sources to the CPU and internal peripheral units.

**Table 1.9.1. Main clock and sub clock generating circuits**

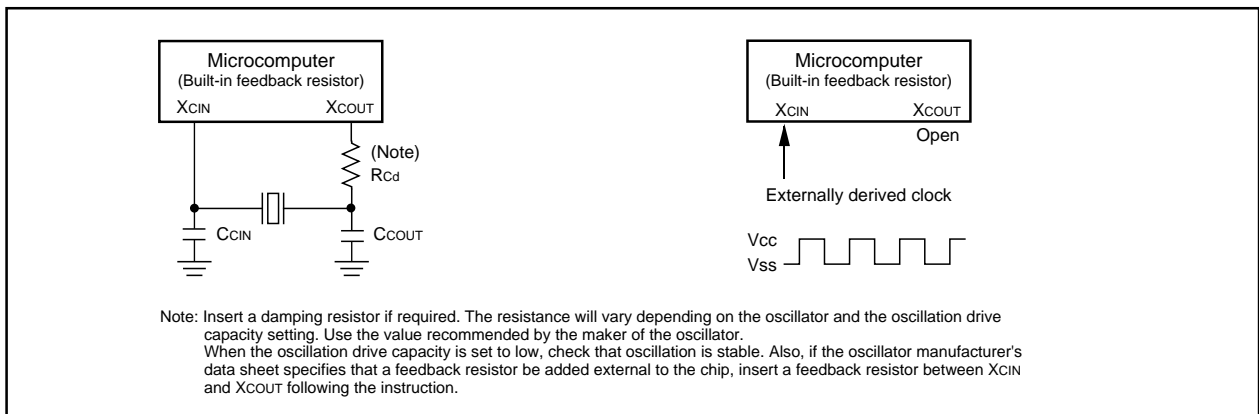
	Main clock generating circuit	Sub clock generating circuit
Use of clock	<ul style="list-style-type: none"> <li>• CPU's operating clock source</li> <li>• Internal peripheral units' operating clock source</li> </ul>	<ul style="list-style-type: none"> <li>• CPU's operating clock source</li> <li>• Timer A/B's count clock source</li> </ul>
Usable oscillator	Ceramic or crystal oscillator	Crystal oscillator
Pins to connect oscillator	XIN, XOUT	XCIN, XCOUT
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 1.9.1 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 1.9.2 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 1.9.1 and 1.9.2 vary with each oscillator used. Use the values recommended by the manufacturer of your oscillator.



**Figure 1.9.1. Examples of main clock**



**Figure 1.9.2. Examples of sub clock**

## Clock Generating Circuit

### Clock Control

Figure 1.9.3 shows the block diagram of the clock generating circuit.

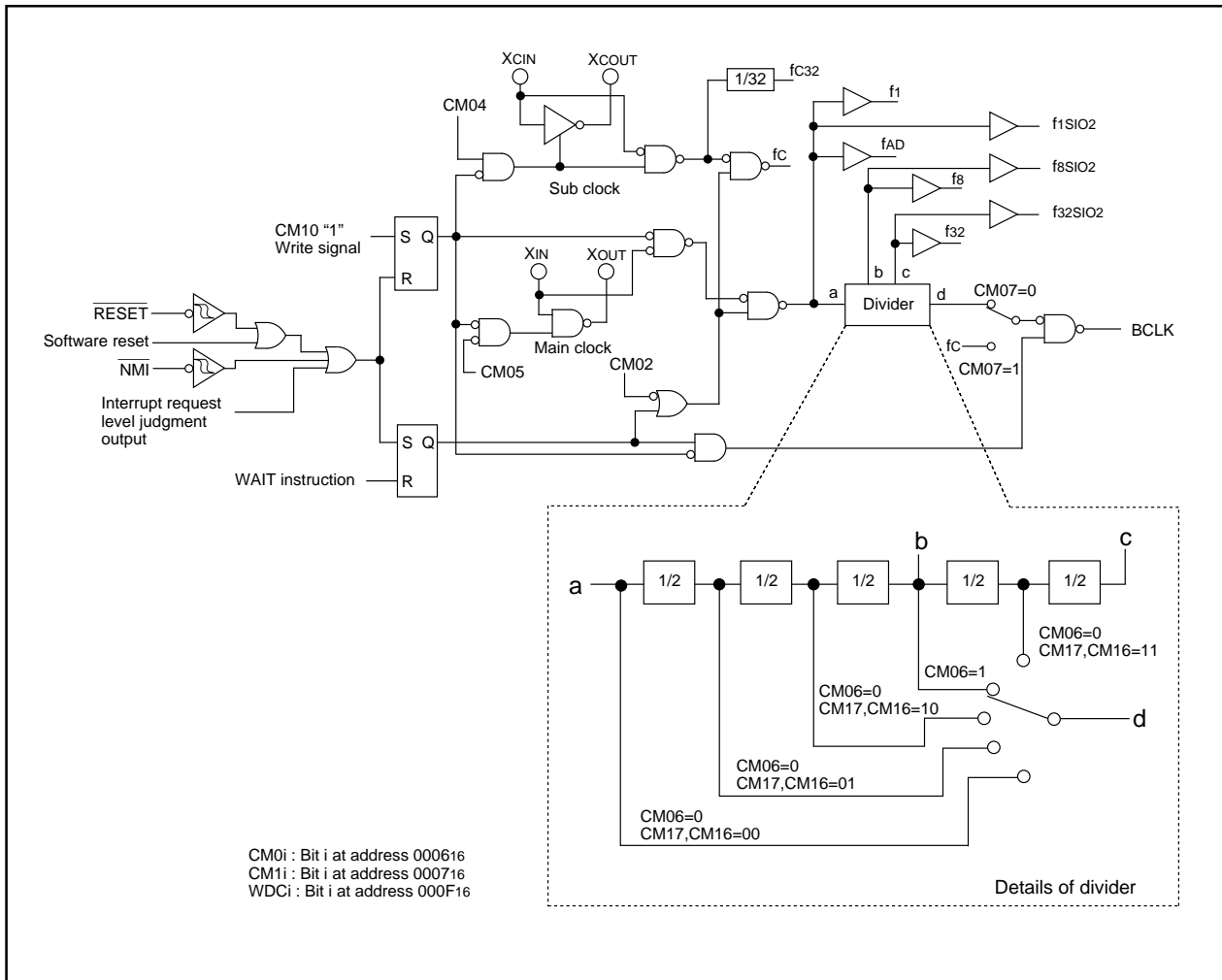


Figure 1.9.3. Clock generating circuit

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 the 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 X<sub>IN</sub>-X<sub>OUT</sub> 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 X<sub>c</sub> select bit (bit 4 at address 0006<sub>16</sub>), the sub-clock can be selected as the 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 X<sub>CIN</sub>-X<sub>COU</sub>T 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.

When the X<sub>CIN</sub>/X<sub>COU</sub>T is used, set ports P86 and P87 as the input ports without pull-up.

### (3) BCLK

The BCLK is the clock that drives the CPU, and is  $f_c$  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 BCLK signal can be output from BCLK pin by the BCLK output disable bit (bit 7 at address 0004<sub>16</sub>) in the memory expansion and the microprocessor modes.

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( $f_1$ , $f_8$ , $f_{32}$ , $f_{1SIO2}$ , $f_{8SIO2}$ , $f_{32SIO2}$ , $f_{AD}$ )

The clock for the peripheral devices is derived from the main clock or by dividing it by 1, 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) $f_{c32}$

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

### (6) $f_c$

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

## Clock Generating Circuit

Figure 1.9.4 shows the system clock control registers 0 and 1.

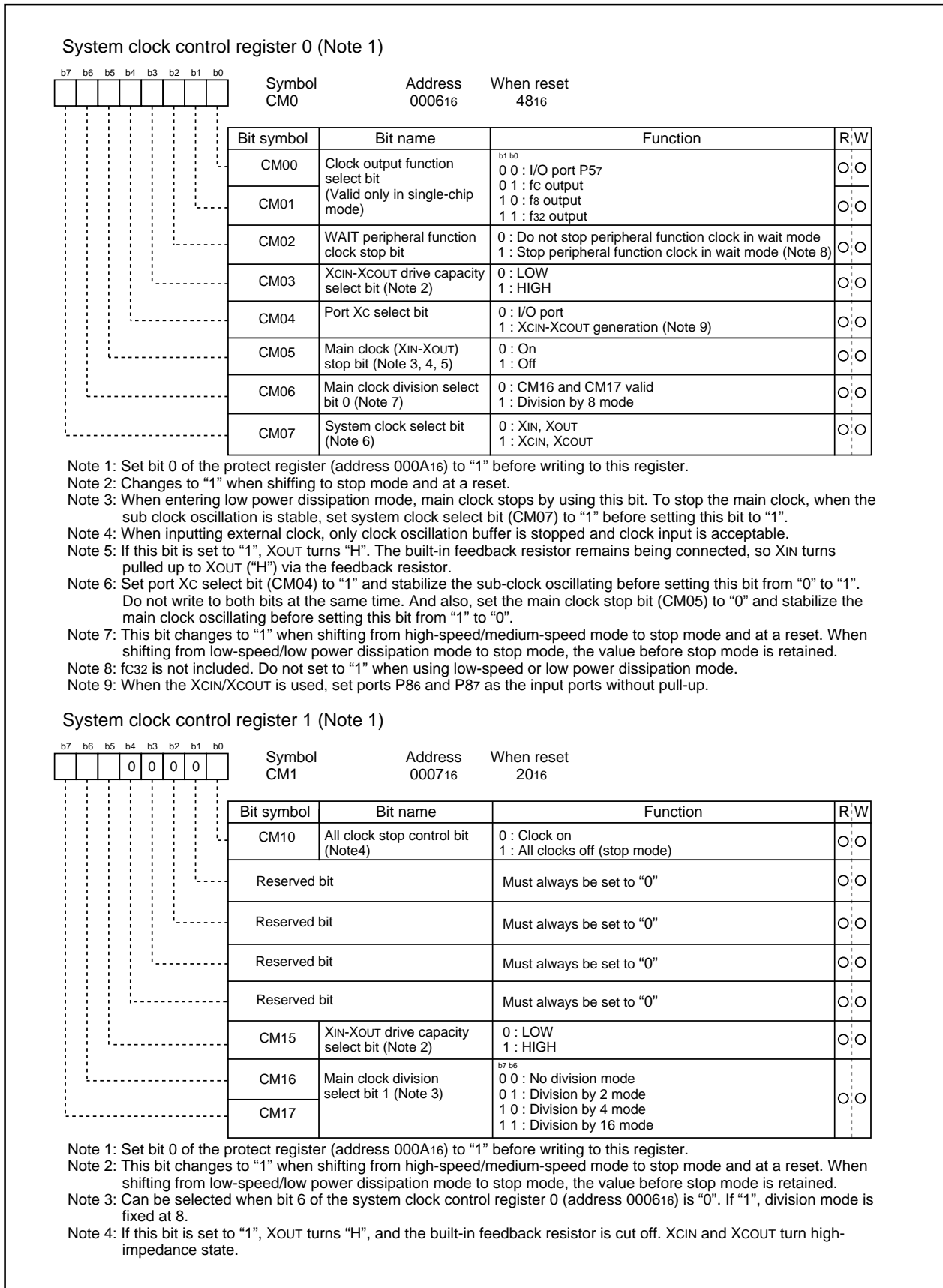


Figure 1.9.4. Clock control registers 0 and 1

## Clock Generating Circuit

### Clock Output

In single-chip mode, the clock output function select bits (bits 0 and 1 at address 0006<sub>16</sub>) enable f<sub>8</sub>, f<sub>32</sub>, or f<sub>c</sub> to be output from the P57/CLKOUT pin. When the WAIT peripheral function clock stop bit (bit 2 at address 0006<sub>16</sub>) is set to "1", the output of f<sub>8</sub> and f<sub>32</sub> stops when a WAIT instruction is executed.

### 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 V<sub>CC</sub> remains above 2V.

Because the oscillation, BCLK, f<sub>1</sub> to f<sub>32</sub>, f<sub>1SIO2</sub> to f<sub>32SIO2</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 and timer B operate provided that the event counter mode is set to an external pulse, and UART<sub>i</sub>(i = 0 to 2), SI/O3,4 functions provided an external clock is selected. Table 1.9.2 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, and the priority level of the interrupt which is not used to cancel must have been changed to 0. If returning by an interrupt, that interrupt routine is executed. If only a hardware reset or an  $\overline{\text{NMI}}$  interrupt is used to cancel stop mode, change the priority level of all interrupt to 0, then shift to stop mode.

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 1.9.2. Port status during stop mode**

Pin		Single-chip mode
Port		Retains status before stop mode
CLKOUT	When f <sub>c</sub> selected	"H"
	When f <sub>8</sub> , f <sub>32</sub> selected	Retains status before stop mode

## Wait Mode

When a WAIT instruction is executed, the BCLK stops and the microcomputer enters the wait mode. In this mode, oscillation continues but the 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. However, peripheral function clock fc32 does not stop so that the peripherals using fc32 do not contribute to the power saving. When the MCU running in low-speed or low power dissipation mode, do not enter WAIT mode with this bit set to “1”. Table 1.9.3 shows the status of the ports in wait mode.

Wait mode is cancelled by a hardware reset or an interrupt. If an interrupt is used to cancel wait mode, that interrupt must first have been enabled, and the priority level of the interrupt which is not used to cancel must have been changed to 0. If returning by an interrupt, the clock in which the WAIT instruction executed is set to BCLK by the microcomputer, and the action is resumed from the interrupt routine. If only a hardware reset or an NMI interrupt is used to cancel wait mode, change the priority level of all interrupt to 0, then shift to wait mode.

**Table 1.9.3. Port status during wait mode**

Pin		Single-chip mode
Port		Retains status before wait mode
CLKOUT	When fc selected	Does not stop
	When f8, f32 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 retained.



## Status Transition Of BCLK

Power dissipation can be reduced and low-voltage operation achieved by changing the count source for BCLK. Table 1.9.4 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 0006<sub>16</sub>) 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 the 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 1.9.4. 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

CM1i : bit i of the address 0007<sub>16</sub>

CM0i : bit i of the address 0006<sub>16</sub>

## Power control

The following is a description of the three available power control modes:

### Modes

Power control is available in three modes.

#### (a) Normal operation mode

- **High-speed mode**

Divide-by-1 frequency of the main clock becomes the BCLK. The CPU operates with the BCLK. Each peripheral function operates according to its assigned clock.

- **Medium-speed mode**

Divide-by-2, divide-by-4, divide-by-8, or divide-by-16 frequency of the main clock becomes the BCLK. The CPU operates with the BCLK. Each peripheral function operates according to its assigned clock.

- **Low-speed mode**

fc becomes the BCLK. The CPU operates according to the fc clock. The fc clock is supplied by the sub-clock. Each peripheral function operates according to its assigned clock.

- **Low power dissipation mode**

The main clock operating in low-speed mode is stopped. The CPU operates according to the fc clock. The fc clock is supplied by the sub-clock. The only peripheral functions that operate are those with the sub-clock selected as the count source.

#### (b) Wait mode

The CPU operation is stopped. The oscillators do not stop.

#### (c) Stop mode

All oscillators stop. The CPU and all built-in peripheral functions stop. This mode, among the three modes listed here, is the most effective in decreasing power consumption.

Figure 1.9.5 is the state transition diagram of the above modes.

Power control

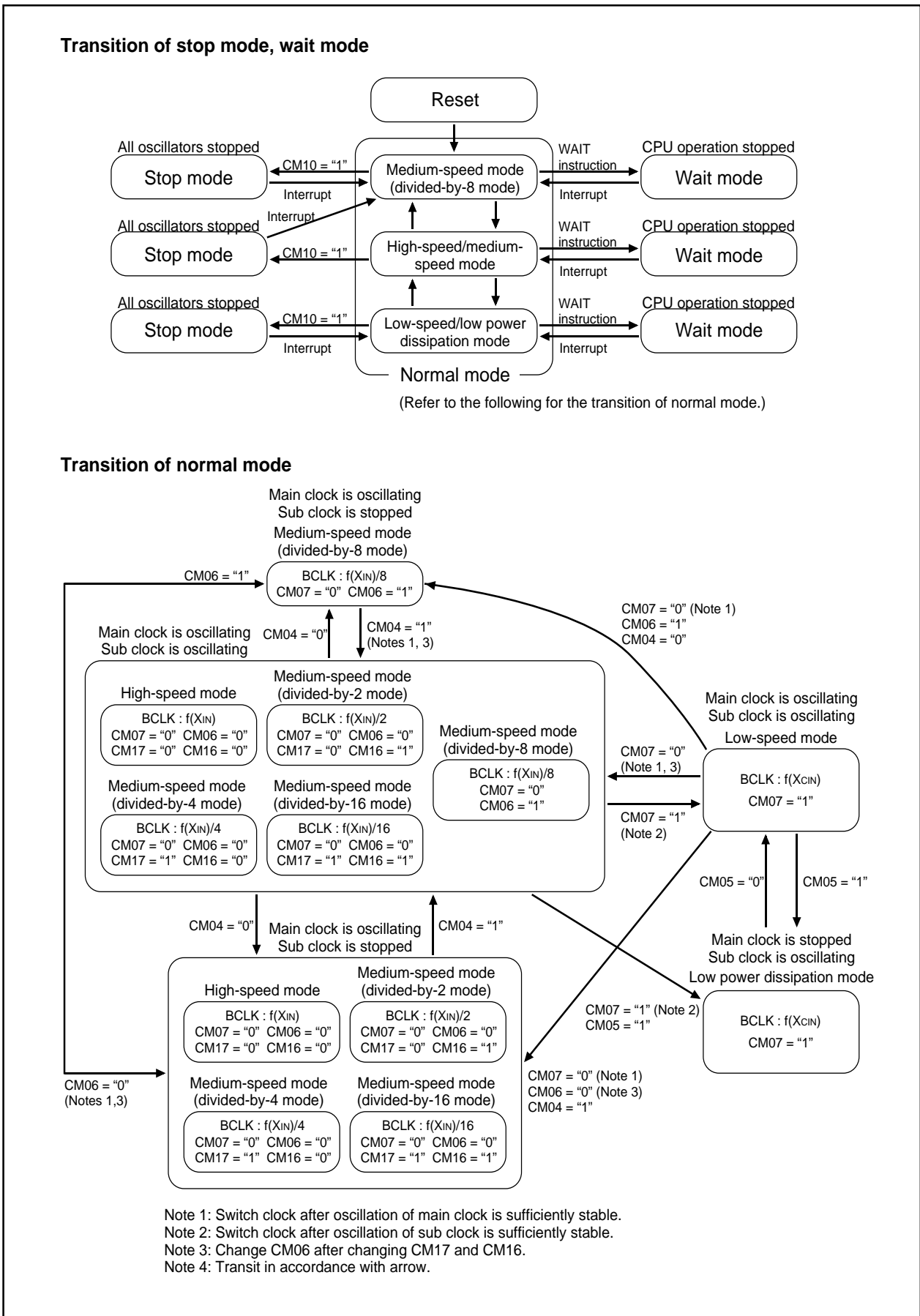


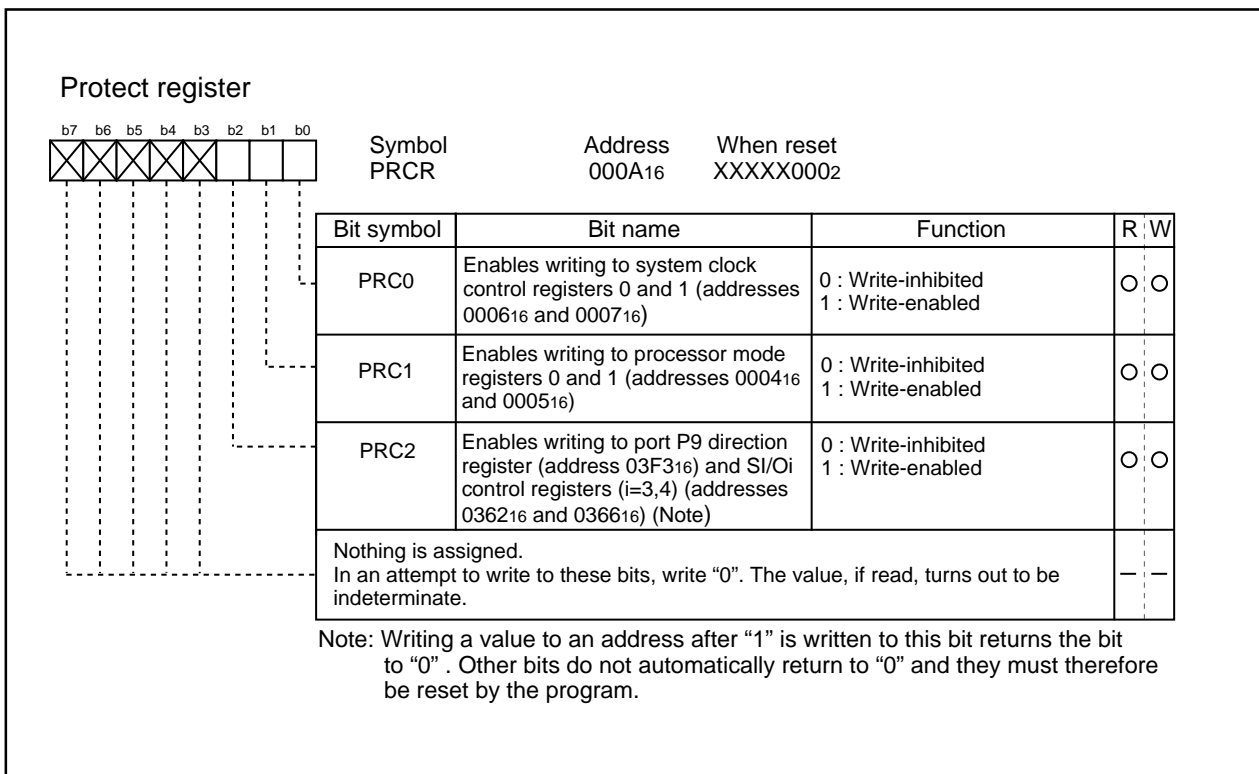
Figure 1.9.5. State transition diagram of Power control mode

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 1.9.6 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>), port P9 direction register (address 03F3<sub>16</sub>), SI/O3 control register (address 0362<sub>16</sub>), and SI/O4 control register (address 0366<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 P9.

If, after "1" (write-enabled) has been written to the port P9 direction register and SI/O<sub>i</sub> control register (i=3,4) 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".

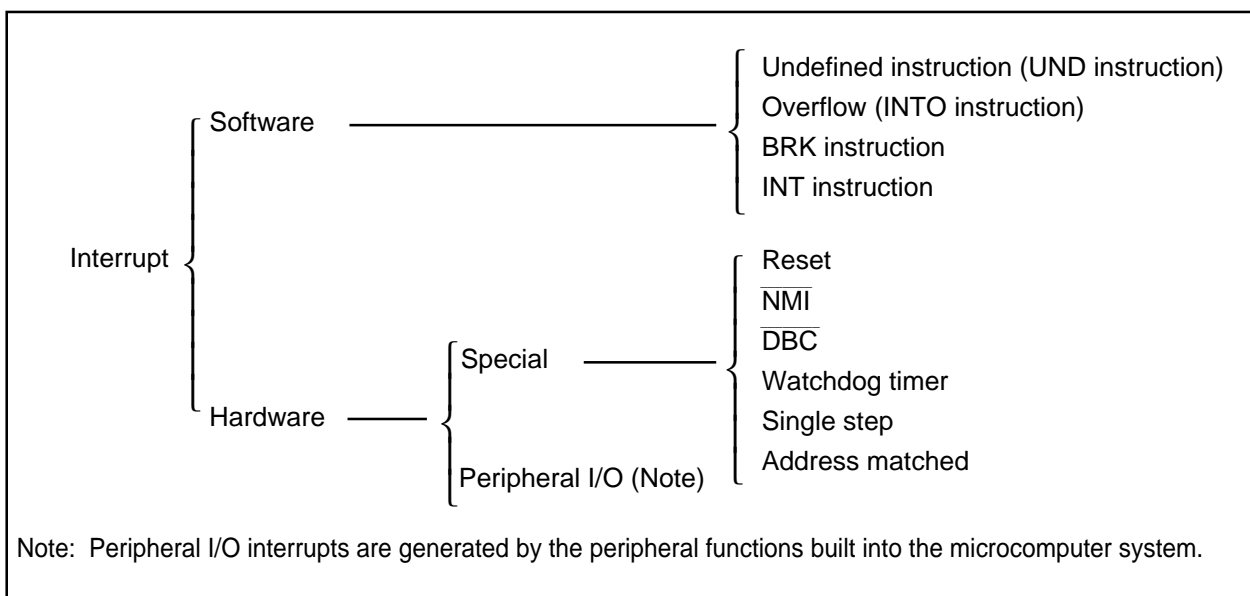


**Figure 1.9.6. Protect register**

## Overview of Interrupt

### Type of Interrupts

Figure 1.10.1 lists the types of interrupts.



**Figure 1.10.1. 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.

## 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 instruction 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. If change the U flag to "0" and select the interrupt stack pointer (ISP), and then execute 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  $\overline{\text{RESET}}$  pin.

- **$\overline{\text{NMI}}$  interrupt**

An  $\overline{\text{NMI}}$  interrupt occurs if an “L” is input to the  $\overline{\text{NMI}}$  pin.

- **$\overline{\text{DBC}}$  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. Built-in peripheral functions are dependent on classes of products, so the interrupt factors too are dependent on classes of products. 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.

- **Bus collision detection interrupt**

This is an interrupt that the serial I/O bus collision detection generates.

- **DMA0 interrupt, DMA1 interrupt**

These are interrupts that DMA generates.

- **Key-input interrupt**

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

- **A-D conversion interrupt**

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

- **UART0, UART1, UART2/NACK, SI/O3 and SI/O4 transmission interrupt**

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

- **UART0, UART1, UART2/ACK, SI/O3 and SI/O4 reception interrupt**

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

- **Timer A0 interrupt through timer A4 interrupt**

These are interrupts that timer A generates

- **Timer B0 interrupt through timer B5 interrupt**

These are interrupts that timer B generates.

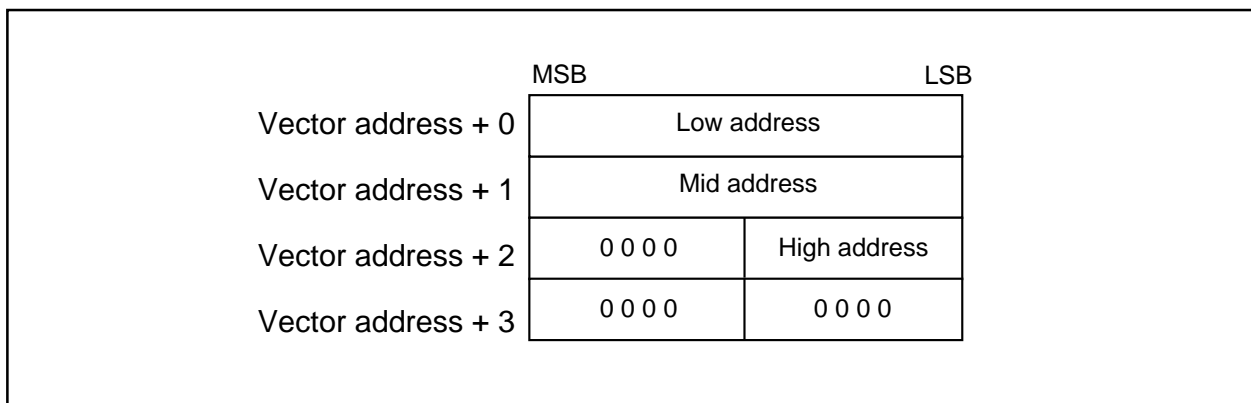
- **$\overline{\text{INT0}}$  interrupt through  $\overline{\text{INT2}}$  interrupt**

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

### 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 1.10.2 shows the format for specifying the address.

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.



**Figure 1.10.2. 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 1.10.1 shows the interrupts assigned to the fixed vector tables and addresses of vector tables.

**Table 1.10.1. Interrupts assigned to the fixed vector tables and addresses of vector tables**

Interrupt source	Vector table addresses Address (L) to address (H)	Remarks
Undefined instruction	FFFD <sub>C16</sub> to FFFD <sub>F16</sub>	Interrupt on UND instruction
Overflow	FFFE <sub>016</sub> to FFFE <sub>316</sub>	Interrupt on INTO instruction
BRK instruction	FFFE <sub>416</sub> to FFFE <sub>716</sub>	If the vector contains FF <sub>16</sub> , program execution starts from the address shown by the vector in the variable vector table
Address match	FFFE <sub>816</sub> to FFFE <sub>B16</sub>	There is an address-matching interrupt enable bit
Single step (Note)	FFFE <sub>C16</sub> to FFFE <sub>F16</sub>	Do not use
Watchdog timer	FFFF <sub>016</sub> to FFFF <sub>316</sub>	
DBC (Note)	FFFF <sub>416</sub> to FFFF <sub>716</sub>	Do not use
NMI	FFFF <sub>816</sub> to FFFF <sub>B16</sub>	External interrupt by input to $\overline{\text{NMI}}$ pin
Reset	FFFF <sub>C16</sub> to FFFF <sub>F16</sub>	

Note: Interrupts used for debugging purposes only.



## Interrupt

### • 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 1.10.2 shows the interrupts assigned to the variable vector tables and addresses of vector tables.

**Table 1.10.2. Interrupts assigned to the variable vector tables and addresses of vector tables**

Software interrupt number	Vector table address Address (L) to address (H)	Interrupt source	Remarks
Software interrupt number 0	+0 to +3 (Note 1)	BRK instruction	Cannot be masked I flag
Software interrupt number 4	+16 to +19 (Note 1)	$\overline{\text{INT3}}$ (Note 4)	
Software interrupt number 5	+20 to +23 (Note 1)	Timer B5	
Software interrupt number 6	+24 to +27 (Note 1)	Timer B4	
Software interrupt number 7	+28 to +31 (Note 1)	Timer B3	
Software interrupt number 8	+32 to +35 (Note 1)	SI/O4/ $\overline{\text{INT5}}$ (Note 3, 4)	
Software interrupt number 9	+36 to +39 (Note 1)	SI/O3/ $\overline{\text{INT4}}$ (Note 3, 4)	
Software interrupt number 10	+40 to +43 (Note 1)	Bus collision detection	
Software interrupt number 11	+44 to +47 (Note 1)	DMA0	
Software interrupt number 12	+48 to +51 (Note 1)	DMA1	
Software interrupt number 13	+52 to +55 (Note 1)	Key input interrupt	
Software interrupt number 14	+56 to +59 (Note 1)	A-D	
Software interrupt number 15	+60 to +63 (Note 1)	UART2 transmit/NACK (Note 2)	
Software interrupt number 16	+64 to +67 (Note 1)	UART2 receive/ACK (Note 2)	
Software interrupt number 17	+68 to +71 (Note 1)	UART0 transmit	
Software interrupt number 18	+72 to +75 (Note 1)	UART0 receive	
Software interrupt number 19	+76 to +79 (Note 1)	UART1 transmit	
Software interrupt number 20	+80 to +83 (Note 1)	UART1 receive	
Software interrupt number 21	+84 to +87 (Note 1)	Timer A0	
Software interrupt number 22	+88 to +91 (Note 1)	Timer A1	
Software interrupt number 23	+92 to +95 (Note 1)	Timer A2	
Software interrupt number 24	+96 to +99 (Note 1)	Timer A3	
Software interrupt number 25	+100 to +103 (Note 1)	Timer A4	
Software interrupt number 26	+104 to +107 (Note 1)	Timer B0	
Software interrupt number 27	+108 to +111 (Note 1)	Timer B1	
Software interrupt number 28	+112 to +115 (Note 1)	Timer B2	
Software interrupt number 29	+116 to +119 (Note 1)	$\overline{\text{INT0}}$	
Software interrupt number 30	+120 to +123 (Note 1)	$\overline{\text{INT1}}$	
Software interrupt number 31	+124 to +127 (Note 1)	$\overline{\text{INT2}}$	
Software interrupt number 32 to Software interrupt number 63	+128 to +131 (Note 1) to +252 to +255 (Note 1)	Software interrupt	Cannot be masked I flag

Note 1: Address relative to address in interrupt table register (INTB).

Note 2: When IIC mode is selected, NACK and ACK interrupts are selected.

Note 3: It is selected by interrupt request cause select bits (bits 6, 7 in address 035F16).

Note 4: P15/ $\overline{\text{INT3}}$  to P17/ $\overline{\text{INT5}}$  do not connect to outside.  $\overline{\text{INT3}}$  to  $\overline{\text{INT5}}$  interrupt cannot be used in M16C/62A (80-pin version) group.

## 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, or 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 1.10.3 shows the interrupt control registers.

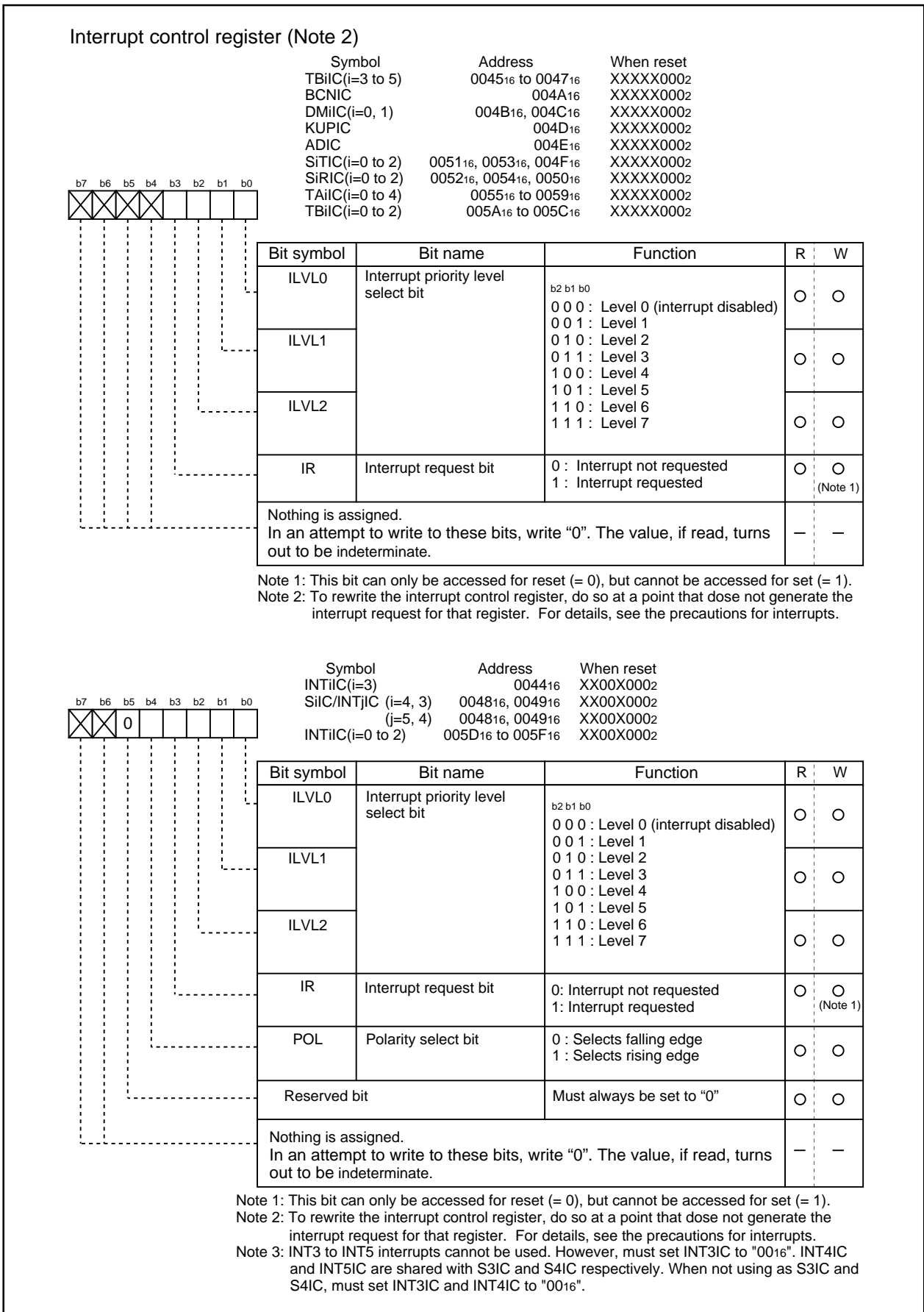


Figure 1.10.3. Interrupt control registers

### Interrupt Enable Flag (I 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 1.10.3 shows the settings of interrupt priority levels and Table 1.10.4 shows the interrupt levels enabled, according to the contents 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 1.10.3. 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 1.10.4. 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

**Rewrite the interrupt control register**

To rewrite the interrupt control register, do so at a point that does not generate the interrupt request for that register. If there is possibility of the interrupt request occur, rewrite the interrupt control register after the interrupt is disabled. 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 (four when using the HOLD function) 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.

When a instruction to rewrite the interrupt control register is executed but the interrupt is disabled, the interrupt request bit is not set sometimes even if the interrupt request for that register has been generated. This will depend on the instruction. If this creates problems, use the below instructions to change the register.

Instructions : AND, OR, BCLR, BSET

## 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 00000<sub>16</sub>. 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 1.10.4 shows the interrupt response time.

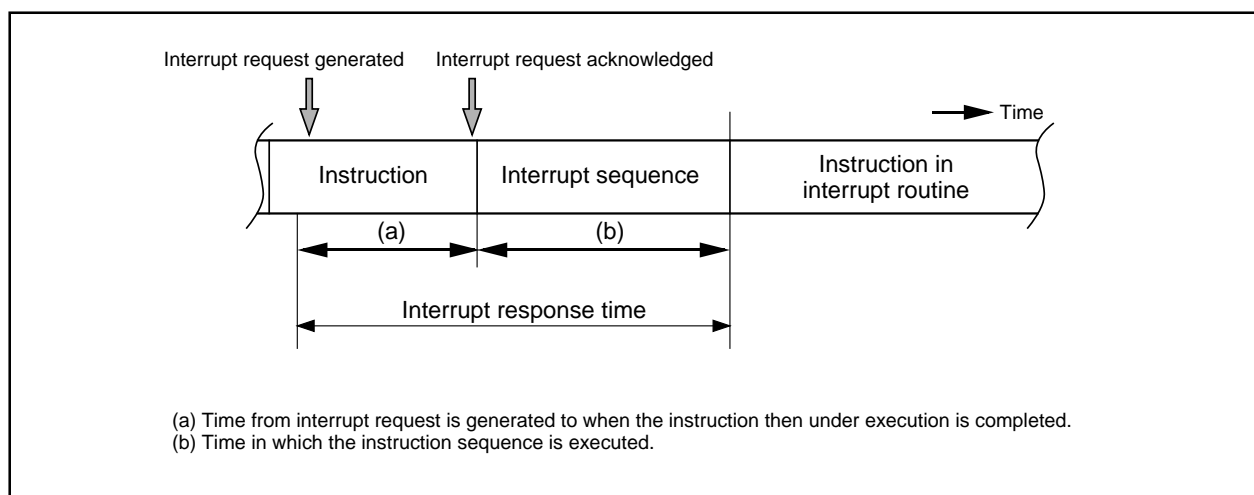


Figure 1.10.4. Interrupt response time

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

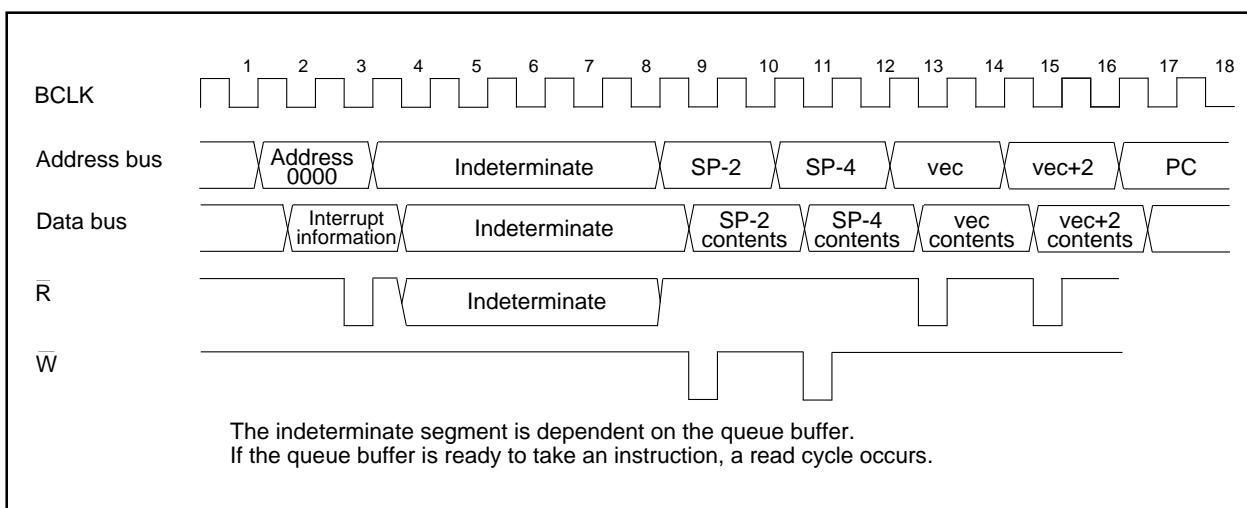
Time (b) is as shown in Table 1.10.5.

**Table 1.10.5. 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 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 1.10.5. 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 1.10.6 is set in the IPL.

**Table 1.10.6. Relationship between interrupts without interrupt priority levels and IPL**

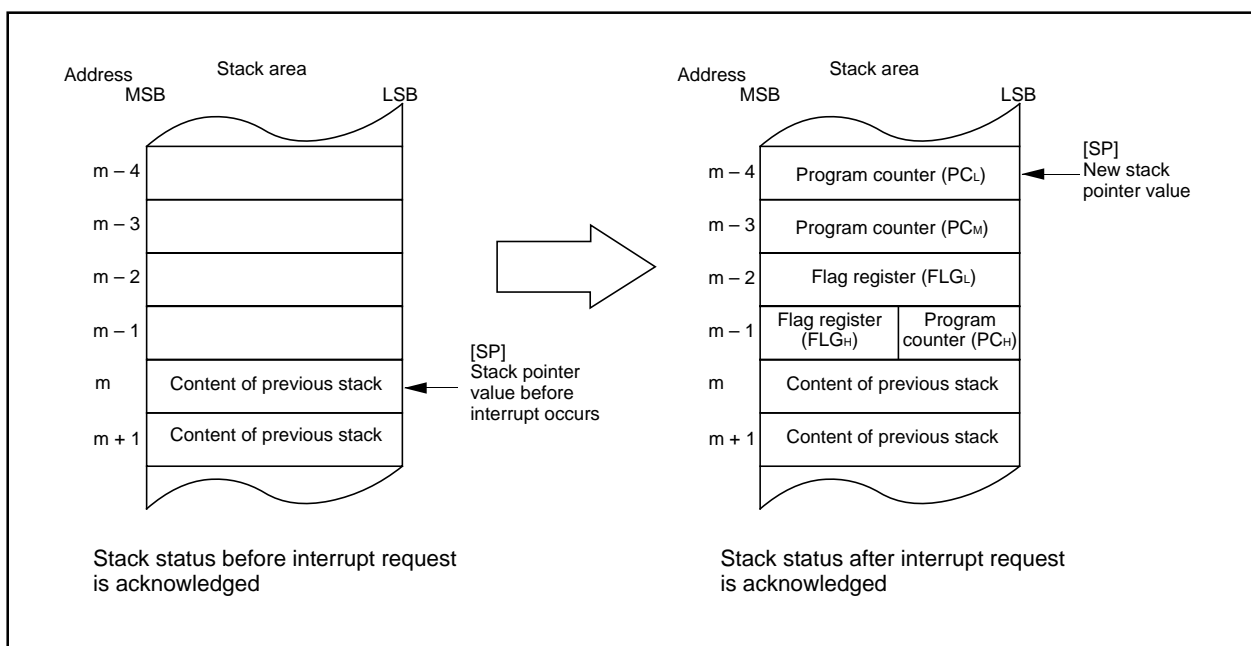
Interrupt sources without priority levels	Value set in the IPL
Watchdog timer, NMI	7
Reset	0
Other	Not changed

### 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 four higher-order bits of the program counter, and 4 upper-order bits and 8 lower-order bits of the FLG register, 16 bits in total, in the stack area, then saves 16 lower-order bits of the program counter. Figure 1.10.6 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).

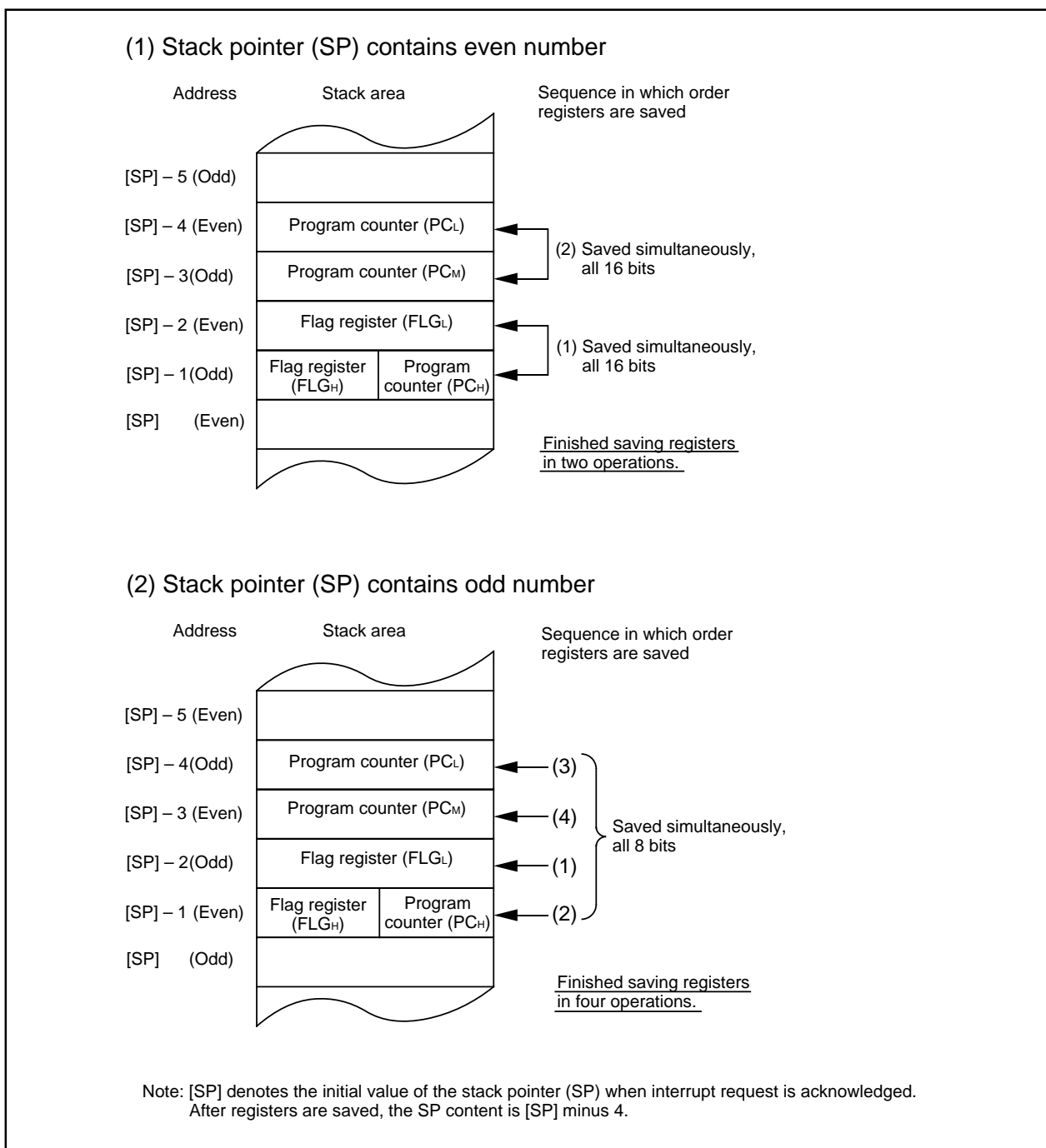


**Figure 1.10.6. State of stack before and after acceptance of interrupt request**



The operation of saving registers carried out in the interrupt sequence is dependent on whether the content of the stack pointer, 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 1.10.7 shows the operation of the saving registers.

Note: When any INT instruction in software numbers 32 to 63 has been executed, this is the stack pointer indicated by the U flag. Otherwise, it is the interrupt stack pointer (ISP).



**Figure 1.10.7. 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 1.10.8 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.

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

Figure 1.10.8. Hardware interrupts priorities

## Interrupt resolution circuit

When two or more interrupts are generated simultaneously, this circuit selects the interrupt with the highest priority level. Figure 1.10.9 shows the circuit that judges the interrupt priority level.

Interrupt

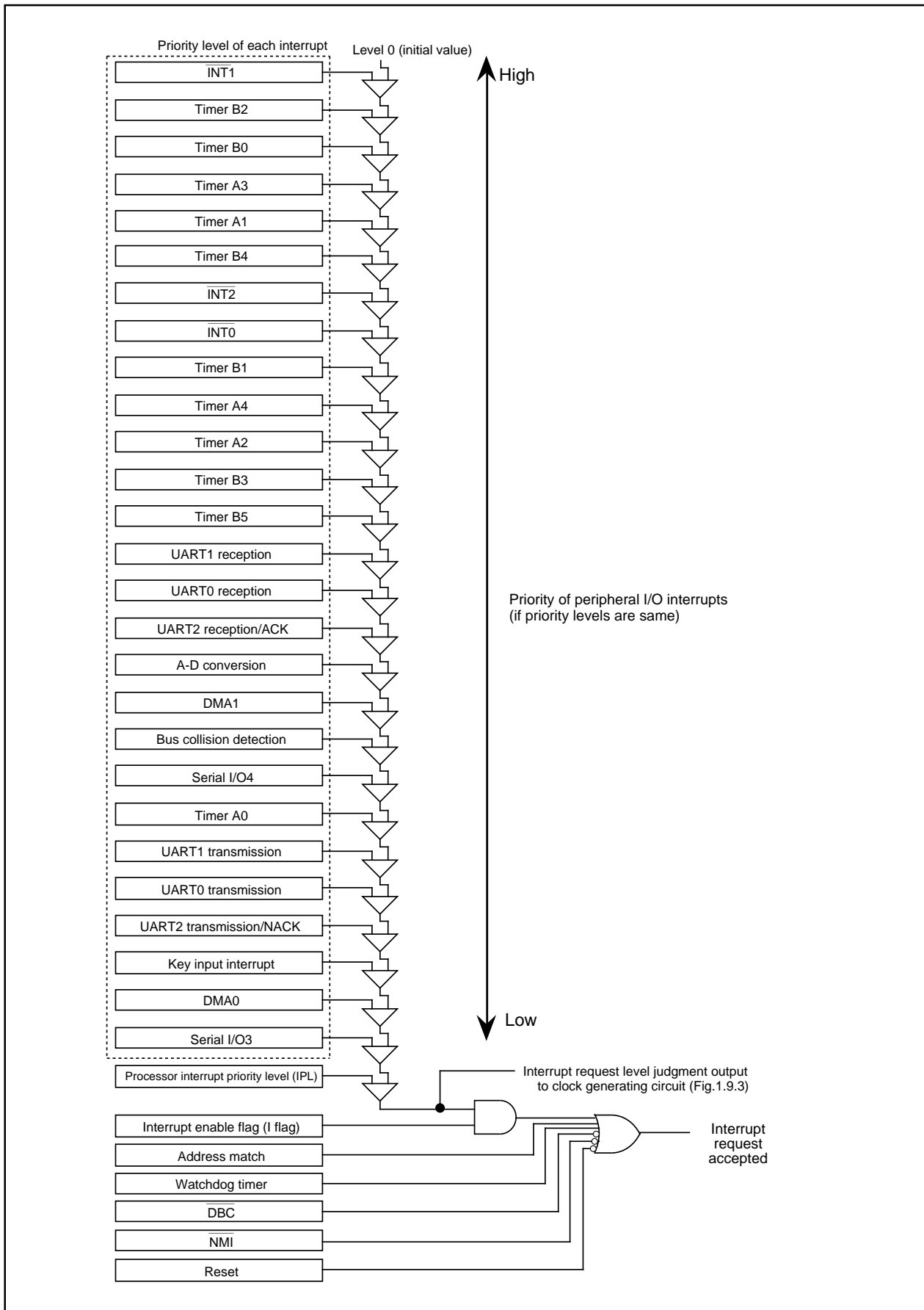


Figure 1.10.9. Maskable interrupts priorities (peripheral I/O interrupts)

## INT Interrupt

### INT Interrupt

INT0 to INT2 are triggered by the edges of external inputs. The edge polarity is selected using the polarity select bit.

As for external interrupt input, an interrupt can be generated both at the rising edge and at the falling edge by setting "1" in the INTi interrupt polarity switching bit of the interrupt request cause select register (035F16). To select both edges, set the polarity switching bit of the corresponding interrupt control register to 'falling edge' ("0").

Figure 1.10.10 shows the Interrupt request cause select register.

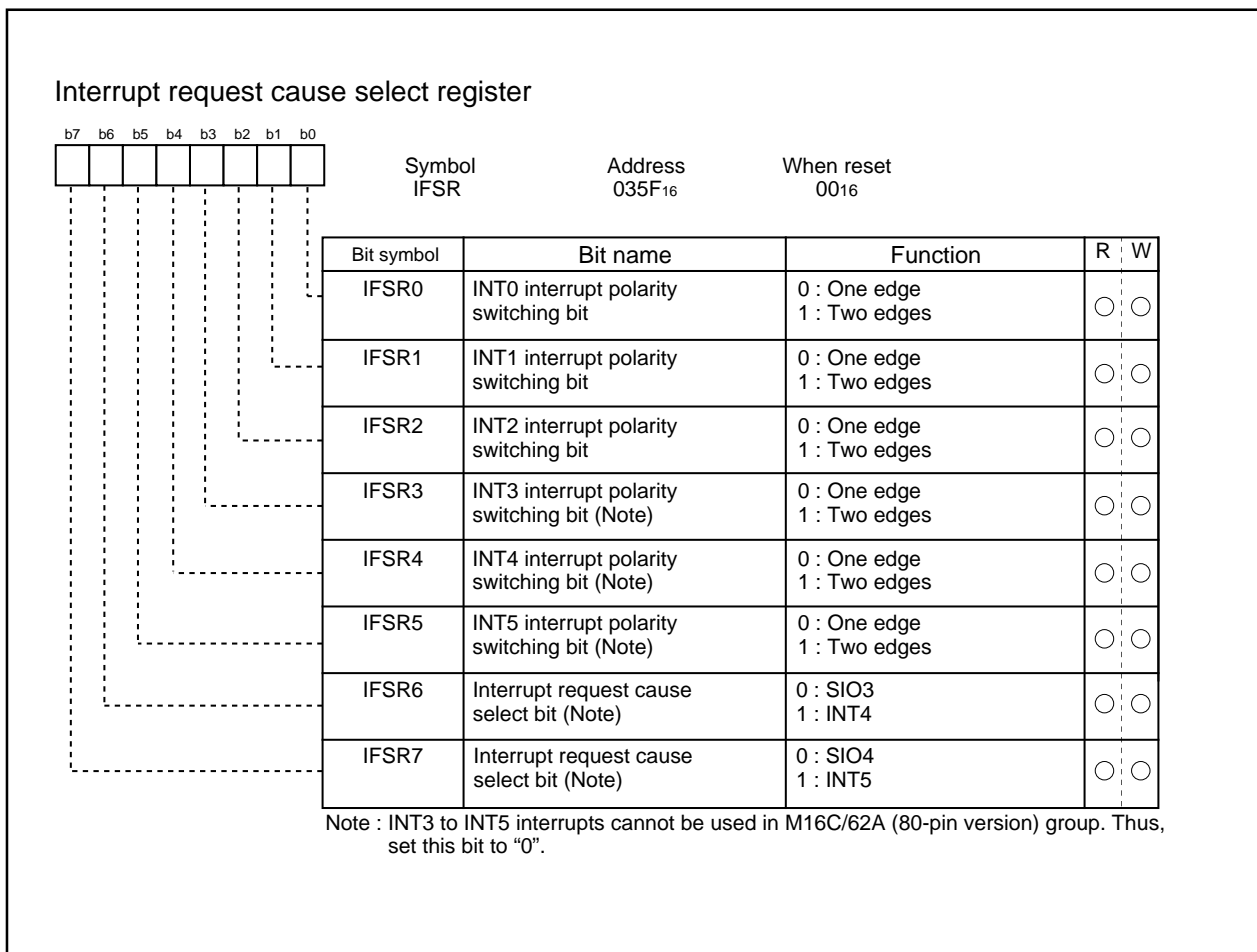


Figure 1.10.10. Interrupt request cause select register

### NMI Interrupt

An  $\overline{\text{NMI}}$  interrupt is generated when the input to the P85/ $\overline{\text{NMI}}$  pin changes from “H” to “L”. The  $\overline{\text{NMI}}$  interrupt is a non-maskable external interrupt. The pin level can be checked in the port P85 register (bit 5 at address 03F016).

This pin cannot be used as a normal port input.

### Key Input Interrupt

If the direction register of any of P104 to P107 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 canceling the wait mode or stop mode. However, if you intend to use the key input interrupt, do not use P104 to P107 as A-D input ports. Figure 1.10.11 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.

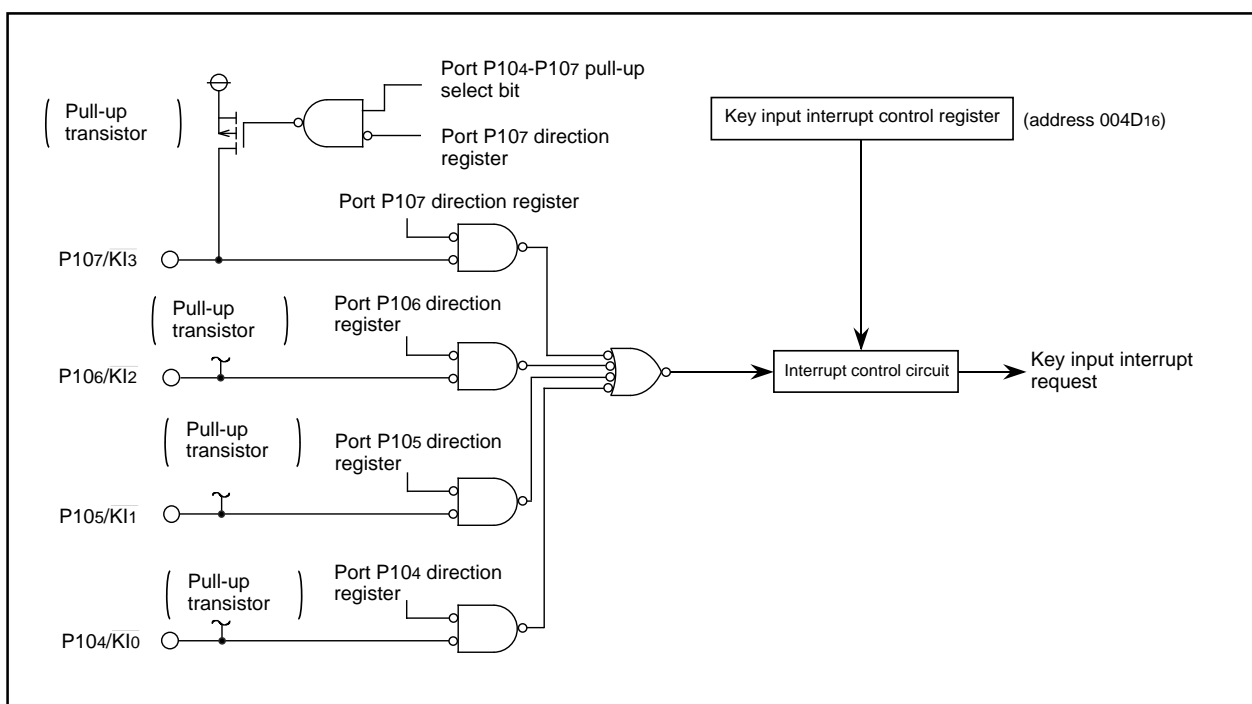


Figure 1.10.11. Block diagram of key input interrupt

### 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). For an address match interrupt, the value of the program counter (PC) that is saved to the stack area varies depending on the instruction being executed.

Figure 1.10.12 shows the address match interrupt-related registers.

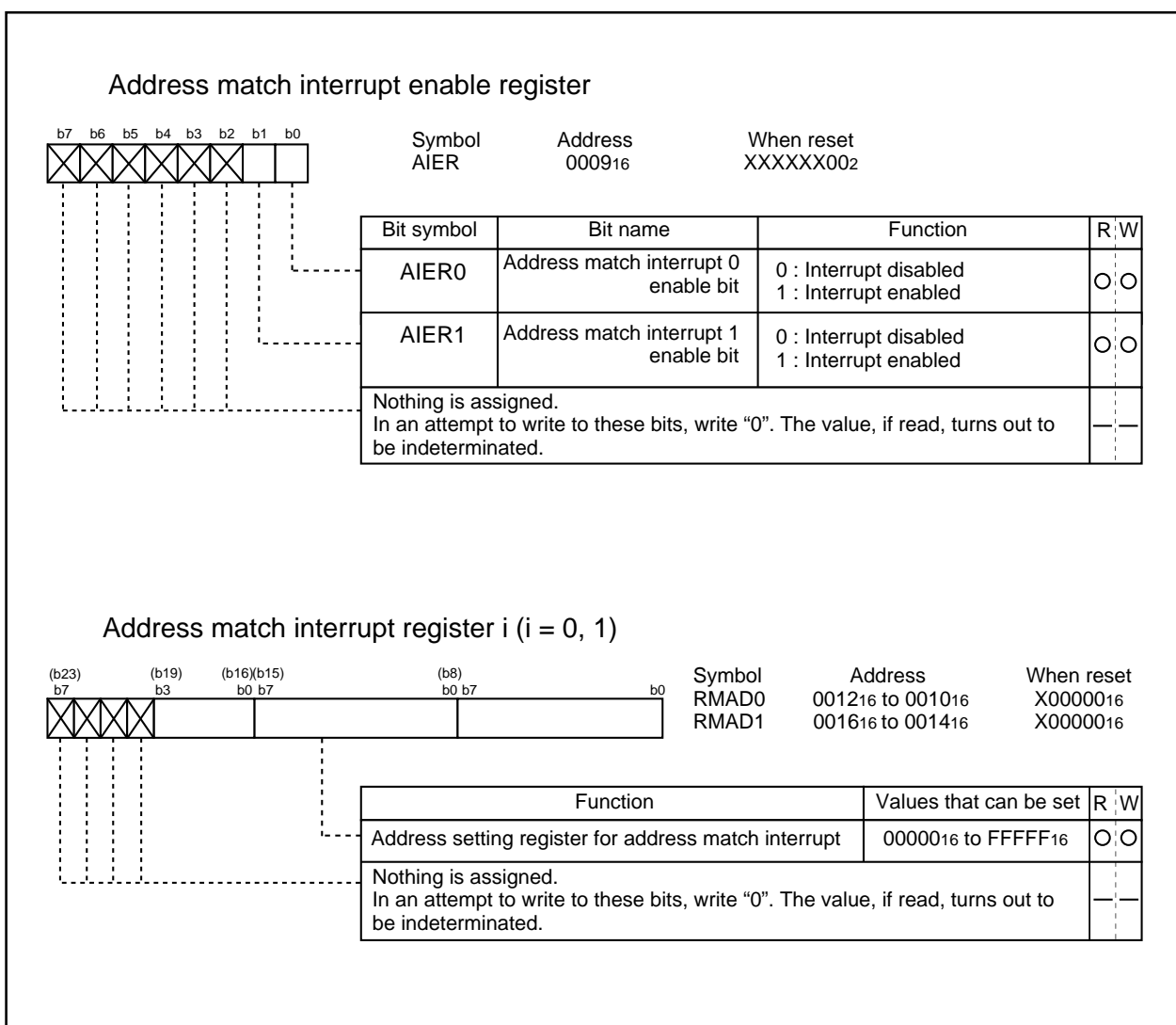


Figure 1.10.12. Address match interrupt-related registers

## Precautions for Interrupts

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

- When maskable interrupt is occurred, CPU reads 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".

Even if the address 00000<sub>16</sub> is read out by software, "0" is set to the enabled highest priority interrupt source request bit. Therefore interrupt can be canceled and unexpected interrupt can occur.

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. When using the  $\overline{\text{NMI}}$  interrupt, initialize the stack point at the beginning of a program. Concerning the first instruction immediately after reset, generating any interrupts including the  $\overline{\text{NMI}}$  interrupt is prohibited.

### (3) The $\overline{\text{NMI}}$ interrupt

- The  $\overline{\text{NMI}}$  interrupt can not be disabled. Be sure to connect  $\overline{\text{NMI}}$  pin to Vcc via a pull-up resistor if unused. Be sure to work on it.
- The  $\overline{\text{NMI}}$  pin also serves as P85, which is exclusively input. Reading the contents of the P8 register allows reading the pin value. Use the reading of this pin only for establishing the pin level at the time when the  $\overline{\text{NMI}}$  interrupt is input.
- Do not reset the CPU with the input to the  $\overline{\text{NMI}}$  pin being in the "L" state.
- Do not attempt to go into stop mode with the input to the  $\overline{\text{NMI}}$  pin being in the "L" state. With the input to the  $\overline{\text{NMI}}$  being in the "L" state, the CM10 is fixed to "0", so attempting to go into stop mode is turned down.
- Do not attempt to go into wait mode with the input to the  $\overline{\text{NMI}}$  pin being in the "L" state. With the input to the  $\overline{\text{NMI}}$  pin being in the "L" state, the CPU stops but the oscillation does not stop, so no power is saved. In this instance, the CPU is returned to the normal state by a later interrupt.
- Signals input to the  $\overline{\text{NMI}}$  pin require an "L" level of 1 clock or more, from the operation clock of the CPU.

### (4) External interrupt

- Either an "L" level or an "H" level of at least 250 ns width is necessary for the signal input to pins  $\overline{\text{INT0}}$  to  $\overline{\text{INT2}}$  regardless of the CPU operation clock.
- When the polarity of the  $\overline{\text{INT0}}$  to  $\overline{\text{INT2}}$  pins is changed, the interrupt request bit is sometimes set to "1". After changing the polarity, set the interrupt request bit to "0". Figure 1.10.13 shows the procedure for changing the  $\overline{\text{INT}}$  interrupt generate factor.

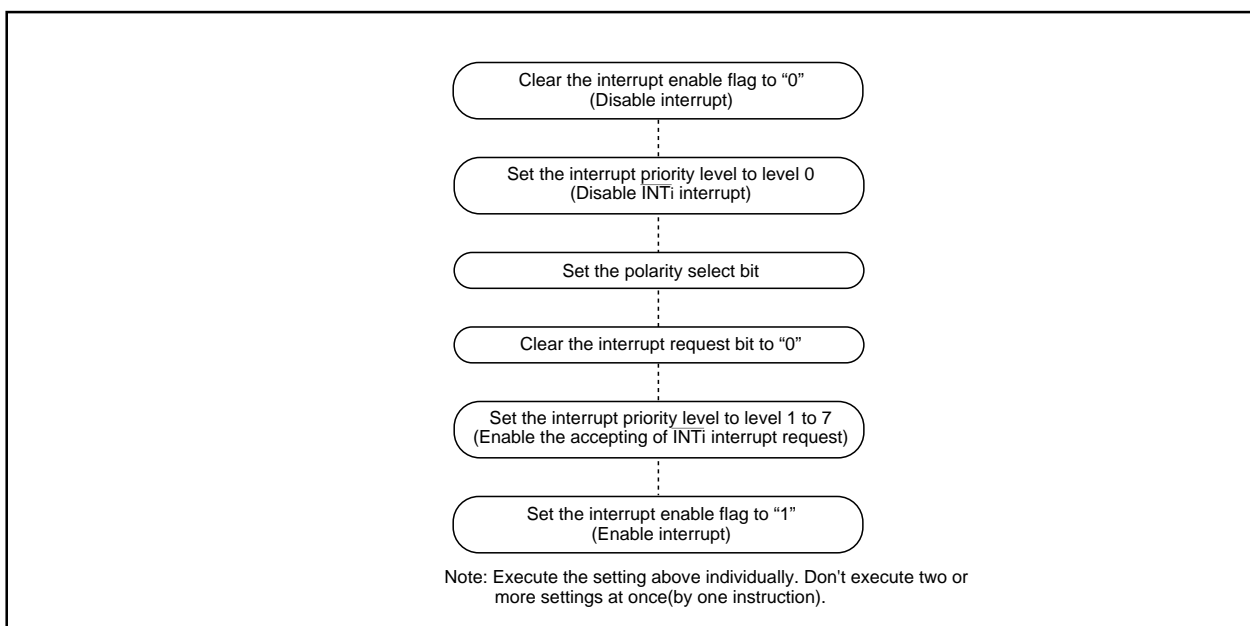


Figure 1.10.13. Switching condition of  $\overline{\text{INT}}$  interrupt request

### (5) Rewrite the interrupt control register

- To rewrite the interrupt control register, do so at a point that does not generate the interrupt request for that register. If there is possibility of the interrupt request occur, rewrite the interrupt control register after the interrupt is disabled. 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 (four when using the HOLD function) 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.

- When a instruction to rewrite the interrupt control register is executed but the interrupt is disabled, the interrupt request bit is not set sometimes even if the interrupt request for that register has been generated. This will depend on the instruction. If this creates problems, use the below instructions to change the register.

Instructions : AND, OR, BCLR, BSET



## Watchdog Timer

The watchdog timer has the function of detecting when the program is out of control. Therefore, we recommend using the watchdog timer to improve reliability of a system. 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 000F16) 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 000F16). Thus the watchdog timer's period can be calculated as given below. The watchdog timer's period is, however, subject to an error due to the prescaler.

### With XIN chosen for BCLK

$$\text{Watchdog timer period} = \frac{\text{prescaler dividing ratio (16 or 128) X watchdog timer count (32768)}}{\text{BCLK}}$$

### With XCIN chosen for BCLK

$$\text{Watchdog timer period} = \frac{\text{prescaler dividing ratio (2) X watchdog timer count (32768)}}{\text{BCLK}}$$

For example, suppose that BCLK runs at 16 MHz and that 16 has been chosen for the dividing ratio of the prescaler, then the watchdog timer's period becomes approximately 32.8 ms.

The watchdog timer is initialized by writing to the watchdog timer start register (address 000E16) 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 000E16). In stop mode and wait mode, the watchdog timer and prescaler are stopped. Counting is resumed from the held value when the modes are released. Figure 1.11.1 shows the block diagram of the watchdog timer. Figure 1.11.2 shows the watchdog timer-related registers.

## Watchdog Timer

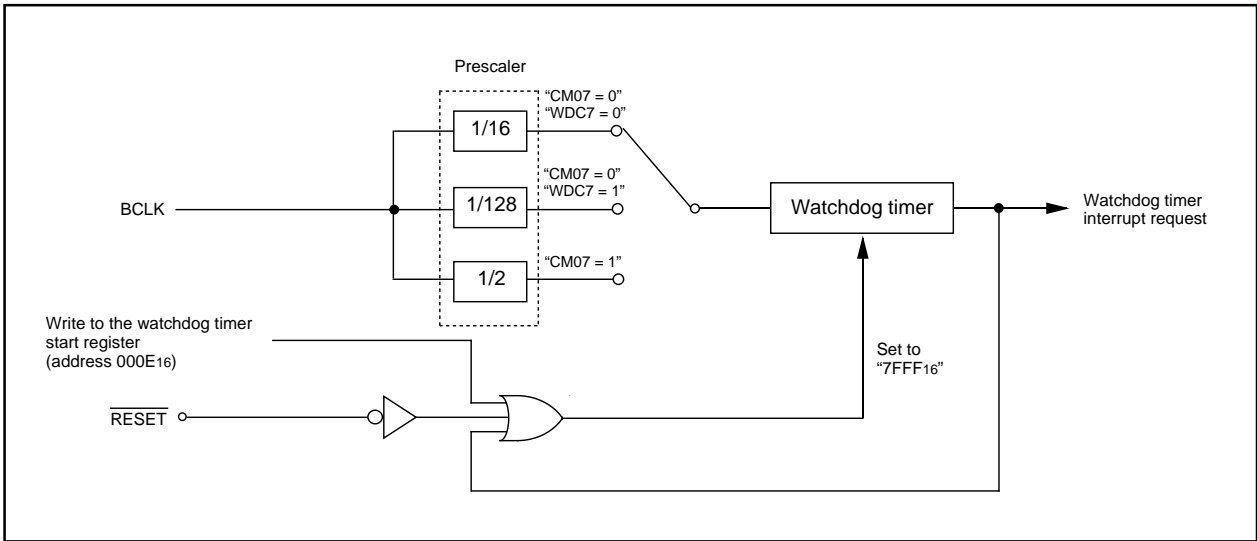


Figure 1.11.1. Block diagram of watchdog timer

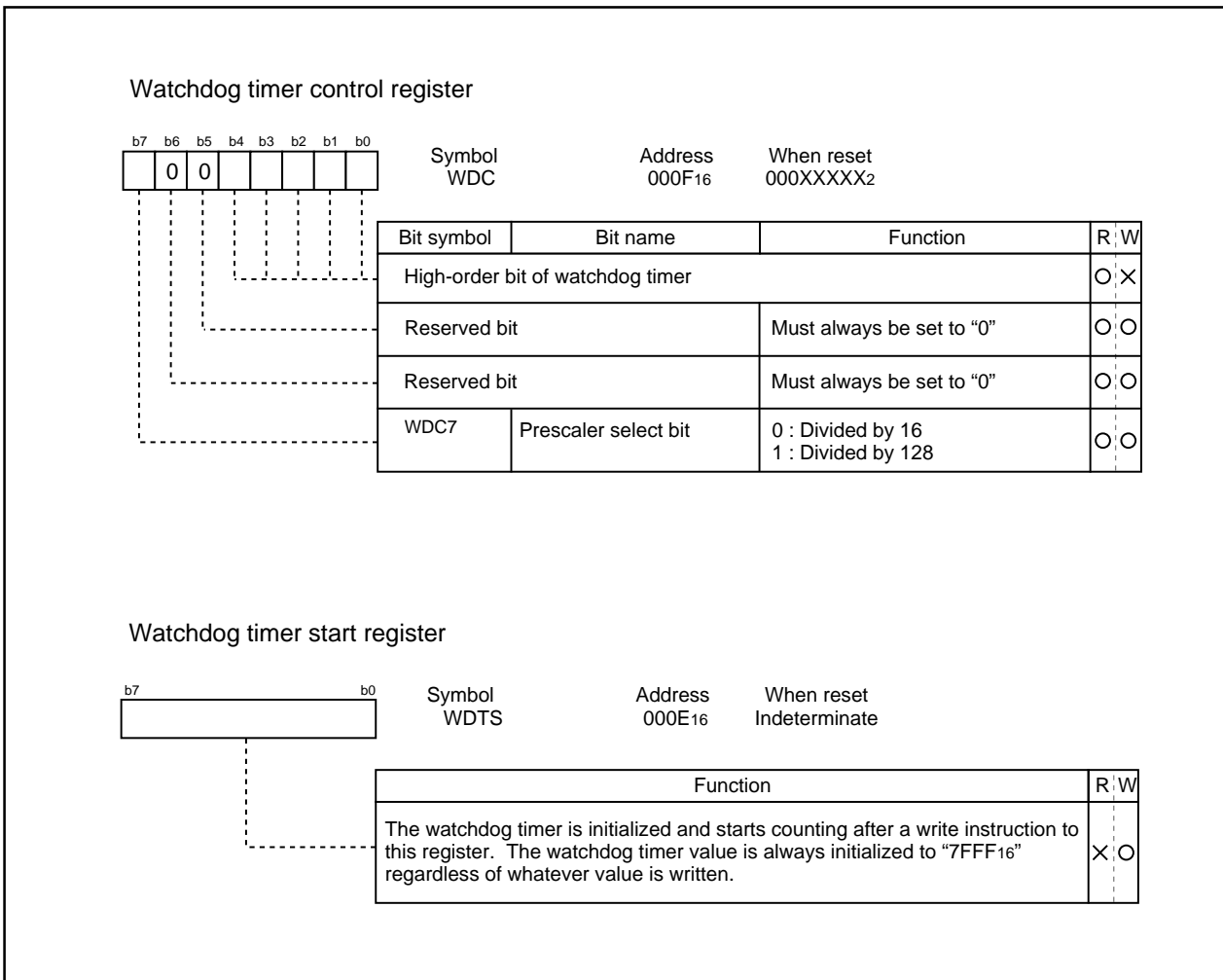
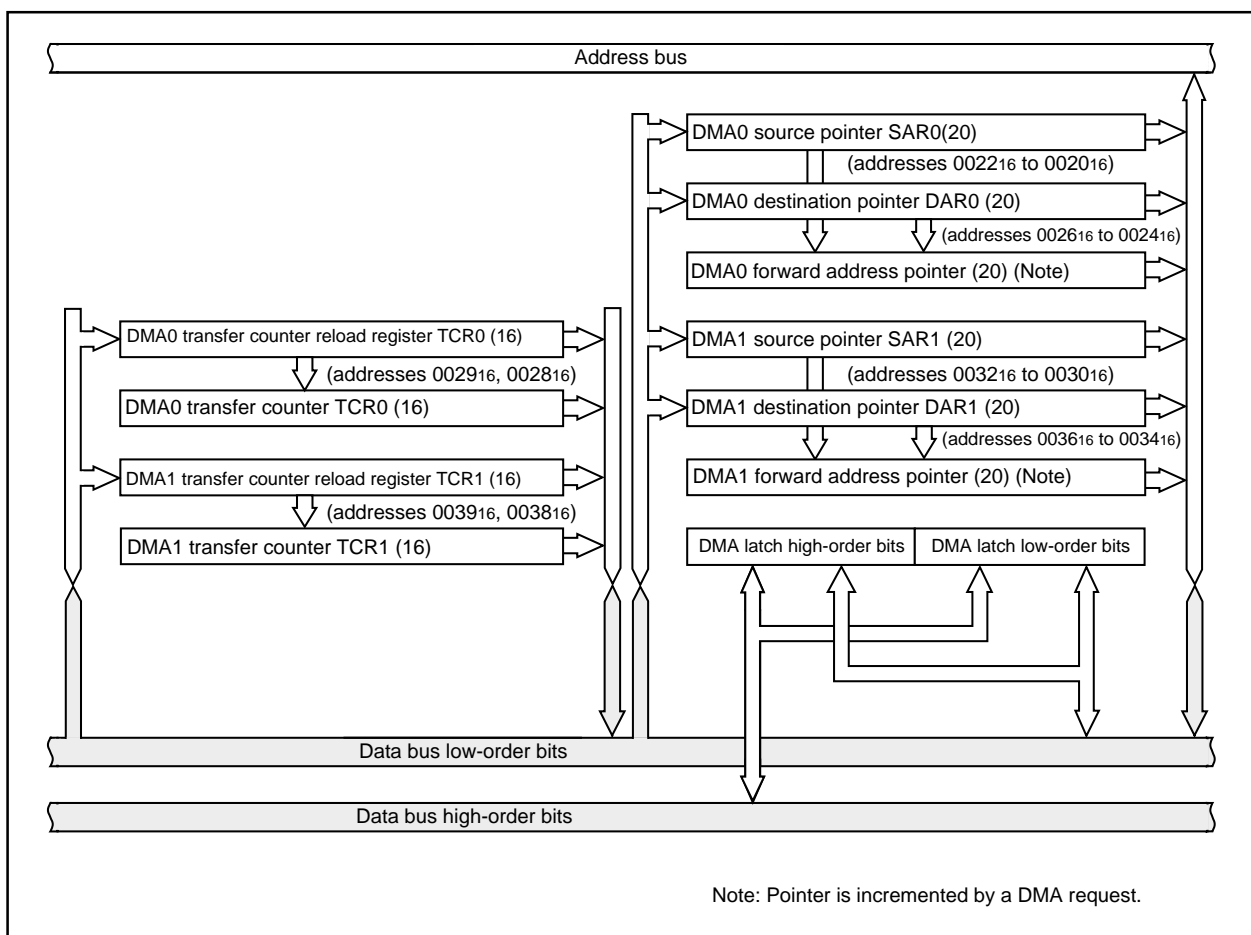


Figure 1.11.2. Watchdog timer control and start registers

## DMAC

This microcomputer has two DMAC (direct memory access controller) channels that allow data to be sent to memory without using the CPU. DMAC shares the same data bus with the CPU. The DMAC is given a higher right of using the bus than the CPU, which leads to working the cycle stealing method. On this account, the operation from the occurrence of DMA transfer request signal to the completion of 1-word (16-bit) or 1-byte (8-bit) data transfer can be performed at high speed. Figure 1.12.1 shows the block diagram of the DMAC. Table 1.12.1 shows the DMAC specifications. Figures 1.12.2 to 1.12.4 show the registers used by the DMAC.



**Figure 1.12.1. Block diagram of DMAC**

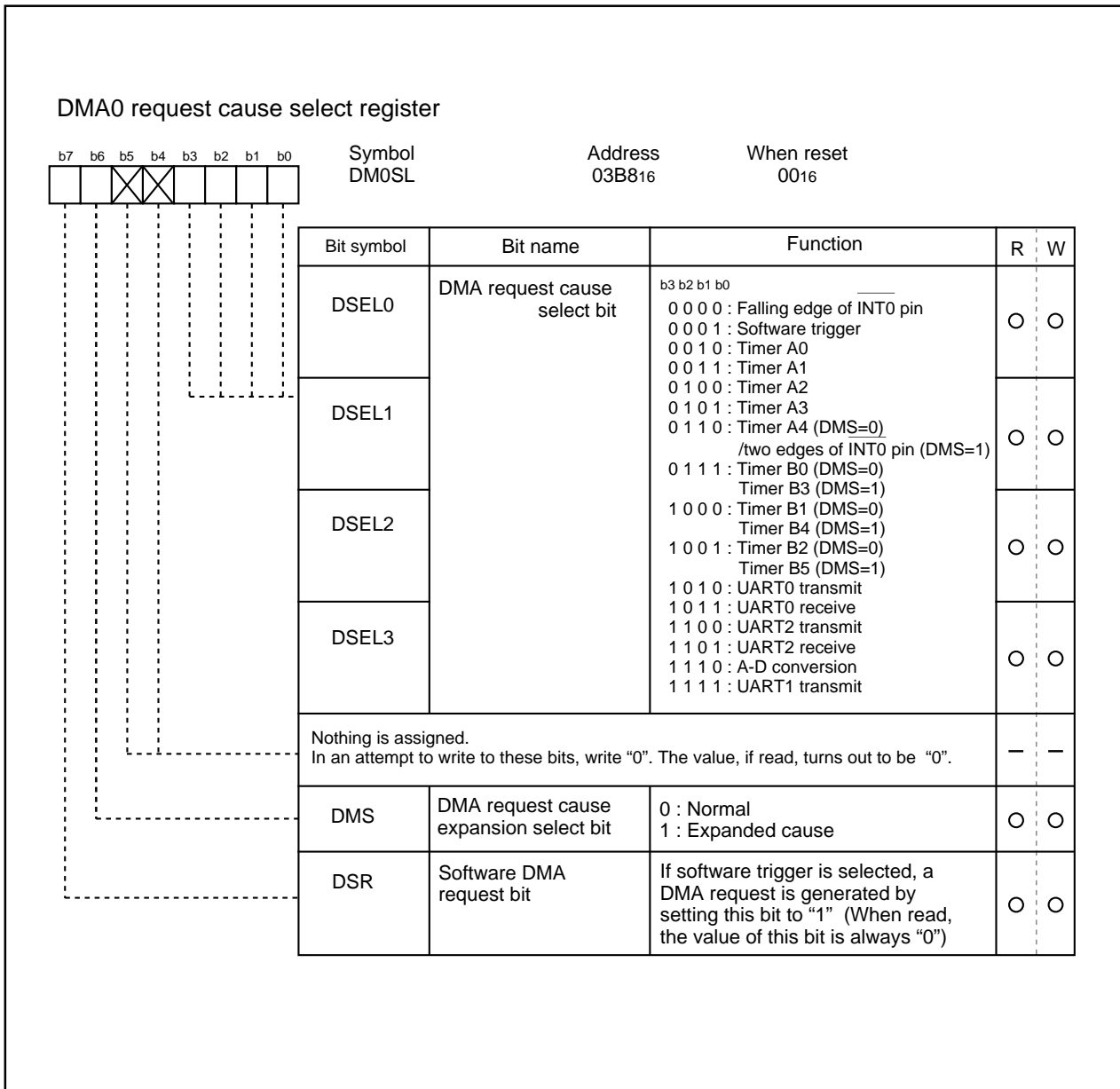
Either a write signal to the software DMA request bit or an interrupt request signal is used as a DMA transfer request signal. But the DMA transfer is affected neither by the interrupt enable flag (I flag) nor by the interrupt priority level. The DMA transfer doesn't affect any interrupts either.

If the DMAC is active (the DMA enable bit is set to 1), data transfer starts every time a DMA transfer request signal occurs. If the cycle of the occurrences of DMA transfer request signals is higher than the DMA transfer cycle, there can be instances in which the number of transfer requests doesn't agree with the number of transfers. For details, see the description of the DMA request bit.

Table 1.12.1. DMAC specifications

Item	Specification
No. of channels	2 (cycle steal method)
Transfer memory space	<ul style="list-style-type: none"> <li>• From any address in the 1M bytes space to a fixed address</li> <li>• From a fixed address to any address in the 1M bytes space</li> <li>• From a fixed address to a fixed address</li> </ul> (Note that DMA-related registers [0020 <sub>16</sub> to 003F <sub>16</sub> ] cannot be accessed)
Maximum No. of bytes transferred	128K bytes (with 16-bit transfers) or 64K bytes (with 8-bit transfers)
DMA request factors (Note)	Falling edge of $\overline{INT0}$ or $\overline{INT1}$ , or both edge Timer A0 to timer A4 interrupt requests Timer B0 to timer B5 interrupt requests UART0 transfer and reception interrupt requests UART1 transfer and reception interrupt requests UART2 transfer and reception interrupt requests Serial I/O3, 4 interrupt requests A-D conversion interrupt requests Software triggers
Channel priority	DMA0 takes precedence if DMA0 and DMA1 requests are generated simultaneously
Transfer unit	8 bits or 16 bits
Transfer address direction	forward/fixed (forward direction cannot be specified for both source and destination simultaneously)
Transfer mode	<ul style="list-style-type: none"> <li>• Single transfer mode After the transfer counter underflows, the DMA enable bit turns to "0", and the DMAC turns inactive</li> <li>• Repeat transfer mode After the transfer counter underflows, the value of the transfer counter reload register is reloaded to the transfer counter. The DMAC remains active unless a "0" is written to the DMA enable bit.</li> </ul>
DMA interrupt request generation timing	When an underflow occurs in the transfer counter
Active	When the DMA enable bit is set to "1", the DMAC is active. When the DMAC is active, data transfer starts every time a DMA transfer request signal occurs.
Inactive	<ul style="list-style-type: none"> <li>• When the DMA enable bit is set to "0", the DMAC is inactive.</li> <li>• After the transfer counter underflows in single transfer mode</li> </ul>
Reload timing for forward address pointer and transfer counter	At the time of starting data transfer immediately after turning the DMAC active, the value of one of source pointer and destination pointer - the one specified for the forward direction - is reloaded to the forward direction address pointer, and the value of the transfer counter reload register is reloaded to the transfer counter.
Writing to register	Registers specified for forward direction transfer are always write enabled. Registers specified for fixed address transfer are write-enabled when the DMA enable bit is "0".
Reading the register	Can be read at any time. However, when the DMA enable bit is "1", reading the register set up as the forward register is the same as reading the value of the forward address pointer.

Note: DMA transfer is not effective to any interrupt. DMA transfer is affected neither by the interrupt enable flag (I flag) nor by the interrupt priority level.



**Figure 1.12.2. DMAC register (1)**

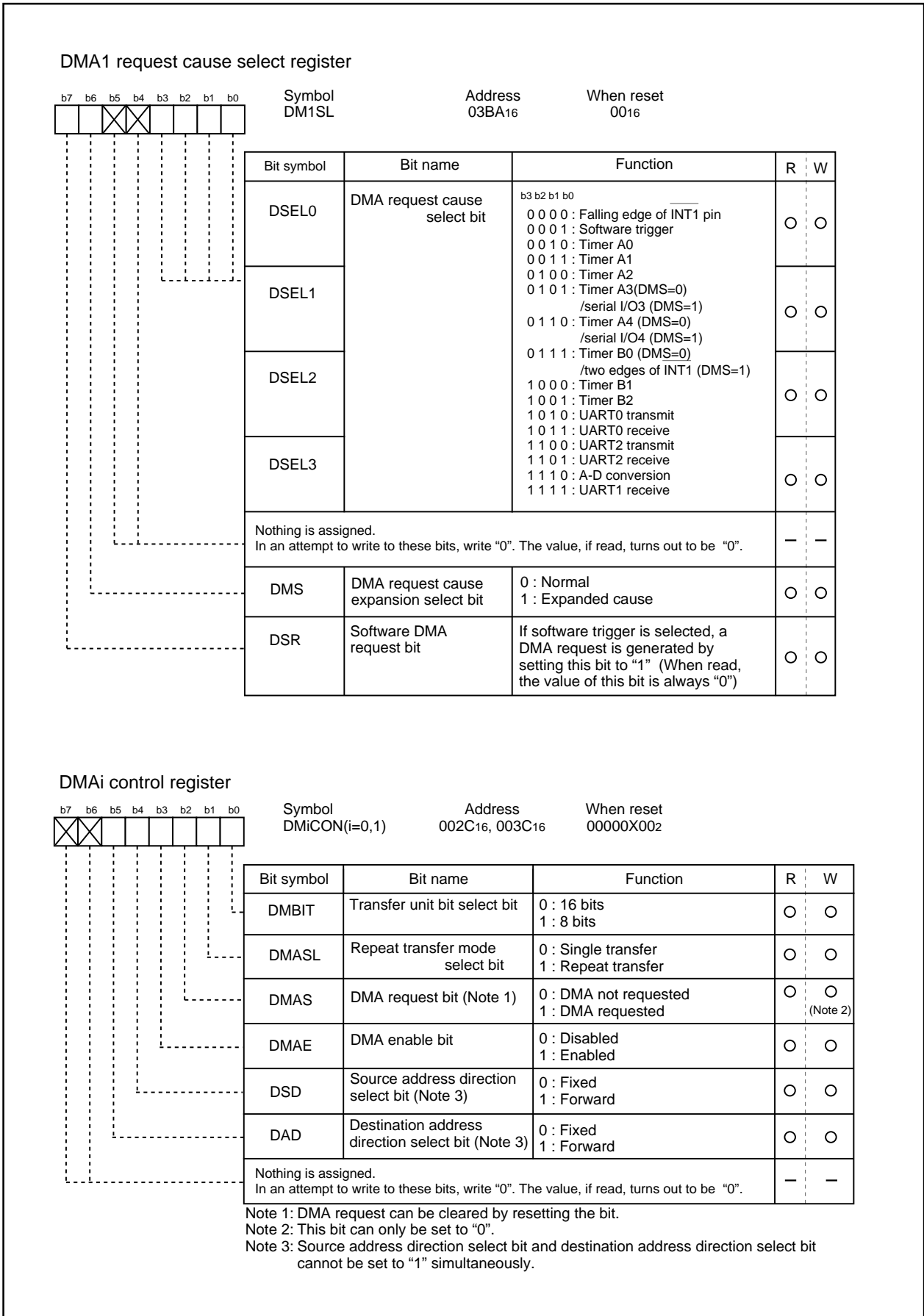


Figure 1.12.3. DMAC register (2)

DMAC

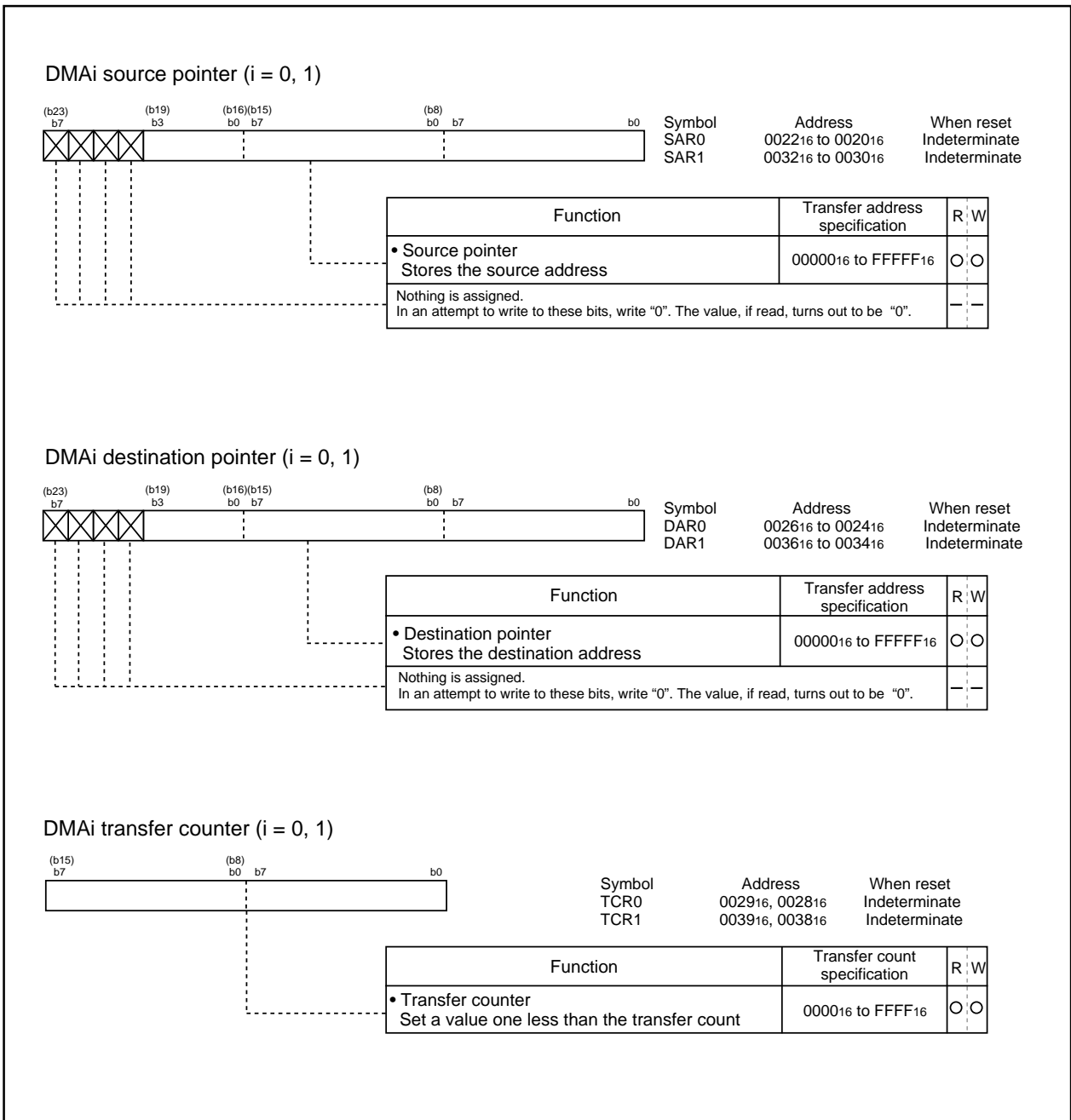


Figure 1.12.4. DMAC register (3)

## (1) Transfer cycle

The transfer cycle consists of the bus cycle in which data is read from memory or from the SFR area (source read) and the bus cycle in which the data is written to memory or to the SFR area (destination write). The number of read and write bus cycles depends on the source and destination addresses. Also, the bus cycle itself is longer when software waits are inserted.

### (a) Effect of source and destination addresses

When 16-bit data is transferred on a 16-bit data bus, and the source and destination both start at odd addresses, there are one more source read cycle and destination write cycle than when the source and destination both start at even addresses.

### (b) Effect of software wait

When the SFR area or a memory area with a software wait is accessed, the number of cycles is increased for the wait by 1 bus cycle. The length of the cycle is determined by BCLK.

Figure 1.12.5 shows the example of the transfer cycles for a source read. For convenience, the destination write cycle is shown as one cycle and the source read cycles for the different conditions are shown. In reality, the destination write cycle is subject to the same conditions as the source read cycle, with the transfer cycle changing accordingly. When calculating the transfer cycle, remember to apply the respective conditions to both the destination write cycle and the source read cycle. For example (2) in Figure 1.12.5, if data is being transferred in 16-bit units and source address is odd, two bus cycles are required for both the source read cycle and the destination write cycle.



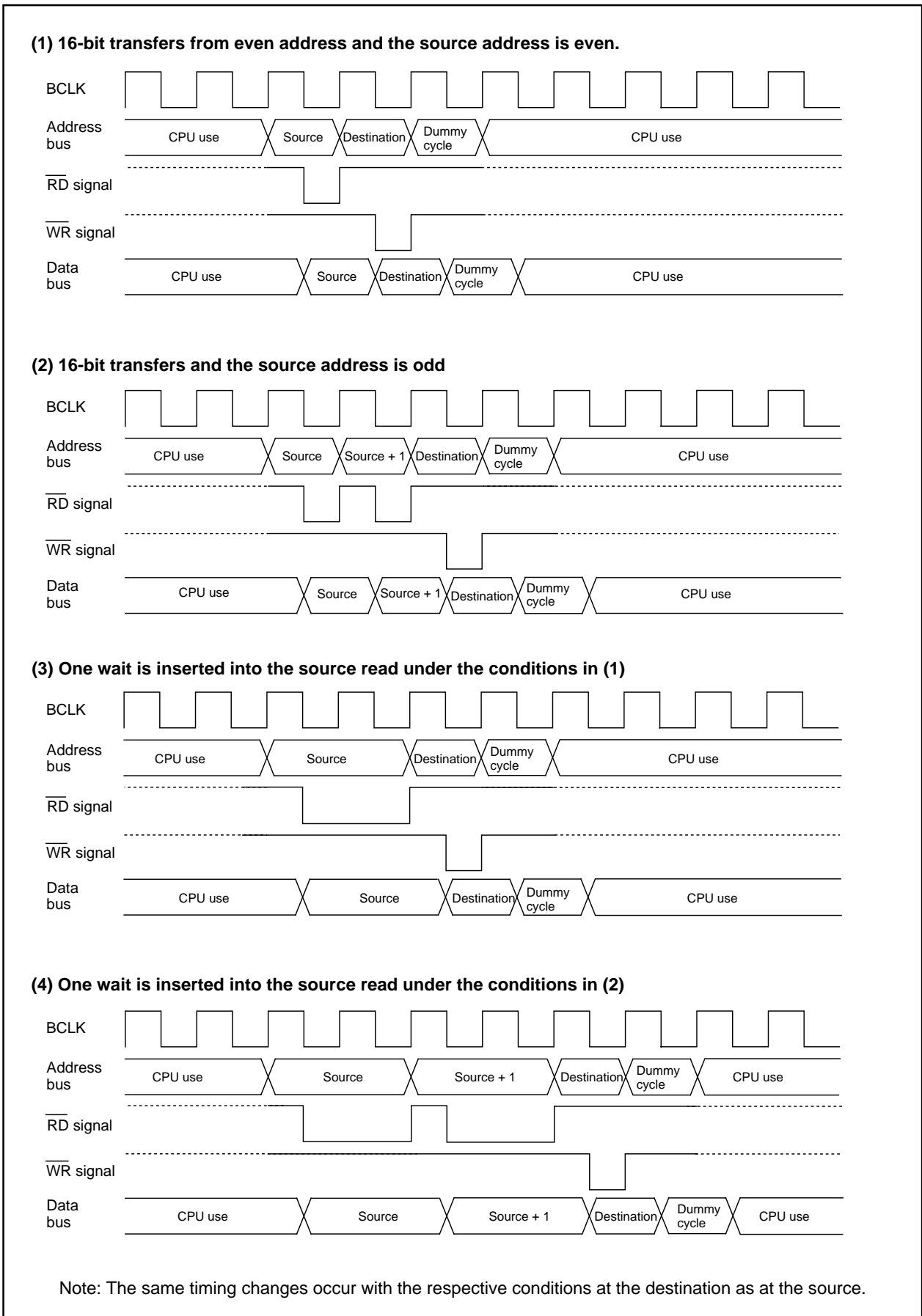


Figure 1.12.5. Example of the transfer cycles for a source read

DMAC

**(2) DMAC transfer cycles**

Any combination of even or odd transfer read and write addresses is possible. Table 1.12.2 shows the number of DMAC transfer cycles.

The number of DMAC transfer cycles can be calculated as follows:

$$\text{No. of transfer cycles per transfer unit} = \text{No. of read cycles} \times j + \text{No. of write cycles} \times k$$

**Table 1.12.2. No. of DMAC transfer cycles**

Transfer unit	Bus width	Access address	Single-chip mode	
			No. of read cycles	No. of write cycles
8-bit transfers (DMBIT= "1")	16-bit (BYTE= "L")	Even	1	1
		Odd	1	1
16-bit transfers (DMBIT= "0")	16-bit (BYTE = "L")	Even	1	1
		Odd	2	2

**Coefficient j, k**

Internal memory		
Internal ROM/RAM No wait	Internal ROM/RAM With wait	SFR area
1	2	2

### DMA enable bit

Setting the DMA enable bit to "1" makes the DMAC active. The DMAC carries out the following operations at the time data transfer starts immediately after DMAC is turned active.

- (1) Reloads the value of one of the source pointer and the destination pointer - the one specified for the forward direction - to the forward direction address pointer.
- (2) Reloads the value of the transfer counter reload register to the transfer counter.

Thus overwriting "1" to the DMA enable bit with the DMAC being active carries out the operations given above, so the DMAC operates again from the initial state at the instant "1" is overwritten to the DMA enable bit.

### DMA request bit

The DMAC can generate a DMA transfer request signal triggered by a factor chosen in advance out of DMA request factors for each channel.

DMA request factors include the following.

\* Factors effected by using the interrupt request signals from the built-in peripheral functions and software DMA factors (internal factors) effected by a program.

\* External factors effected by utilizing the input from external interrupt signals.

For the selection of DMA request factors, see the descriptions of the DMA<sub>i</sub> factor selection register.

The DMA request bit turns to "1" if the DMA transfer request signal occurs regardless of the DMAC's state (regardless of whether the DMA enable bit is set to "1" or "0"). It turns to "0" immediately before data transfer starts.

In addition, it can be set to "0" by use of a program, but cannot be set to "1".

There can be instances in which a change in DMA request factor selection bit causes the DMA request bit to turn to "1". So be sure to set the DMA request bit to "0" after the DMA request factor selection bit is changed.

If the DMAC is active, data transfer starts immediately, so the value of the DMA request bit, if read by use of a program, turns out to be "0" in most cases. To examine whether the DMAC is active, read the DMA enable bit.

Here follows the timing of changes in the DMA request bit.

#### (1) Internal factors

Except the DMA request factors triggered by software, the timing for the DMA request bit to turn to "1" due to an internal factor is the same as the timing for the interrupt request bit of the interrupt control register to turn to "1" due to several factors.

Turning the DMA request bit to "0" due to an internal factor is timed to be effected immediately before the transfer starts.

#### (2) External factors

An external factor is a factor caused to occur by the leading edge of input from the  $\overline{\text{INT}}_i$  pin (i depends on which DMAC channel is used).

Selecting the  $\overline{\text{INT}}_i$  pins as external factors using the DMA request factor selection bit causes input from these pins to become the DMA transfer request signals.

The timing for the DMA request bit to turn to "1" when an external factor is selected synchronizes with the signal's edge applicable to the function specified by the DMA request factor selection bit (synchronizes with the trailing edge of the input signal to each  $\overline{\text{INT}}_i$  pin, for example).

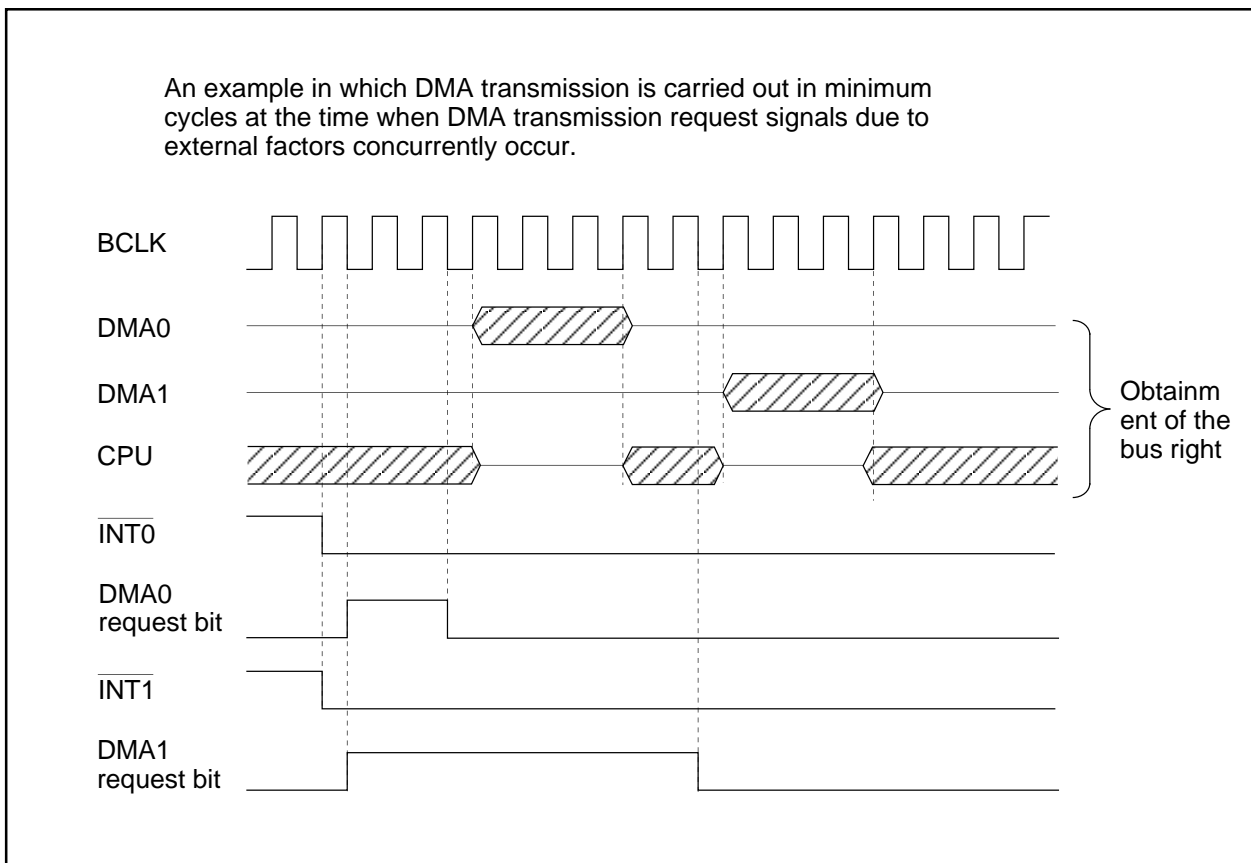
With an external factor selected, the DMA request bit is timed to turn to "0" immediately before data transfer starts similarly to the state in which an internal factor is selected.

**(3) The priorities of channels and DMA transfer timing**

If a DMA transfer request signal falls on a single sampling cycle (a sampling cycle means one period from the leading edge to the trailing edge of BCLK), the DMA request bits of applicable channels concurrently turn to "1". If the channels are active at that moment, DMA0 is given a high priority to start data transfer. When DMA0 finishes data transfer, it gives the bus right to the CPU. When the CPU finishes single bus access, then DMA1 starts data transfer and gives the bus right to the CPU.

An example in which DMA transfer is carried out in minimum cycles at the time when DMA transfer request signals due to external factors concurrently occur.

Figure 1.12.6 shows an example of DMA transfer effected by external factors.



**Figure 1.12.6. An example of DMA transfer effected by external factors**

## Timer

### Timer

There are eleven 16-bit timers. These timers can be classified by function into timers A (five) and timers B (six). All these timers function independently. Figures 1.13.1 and 1.13.2 show the block diagram of timers.

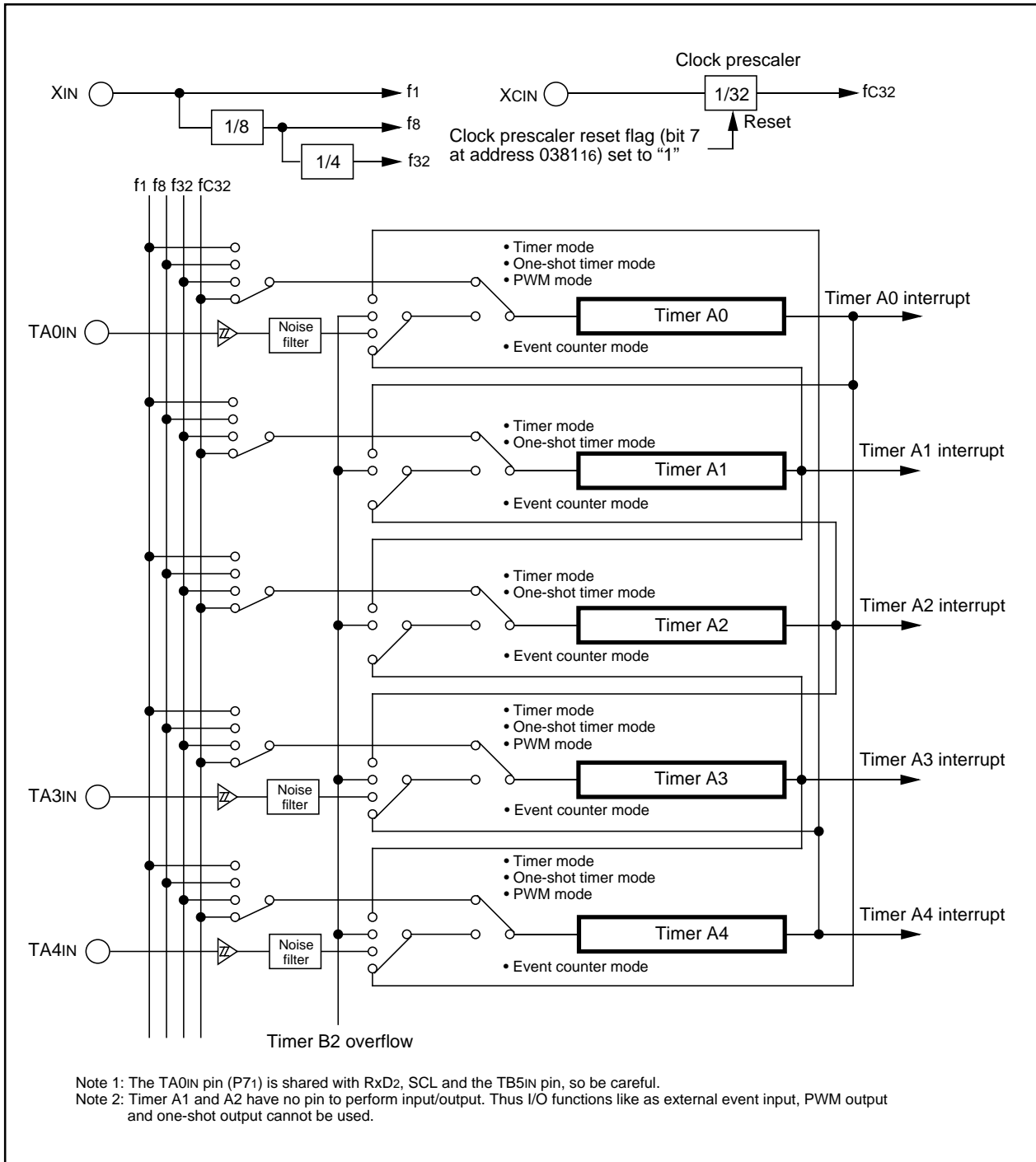


Figure 1.13.1. Timer A block diagram

Timer

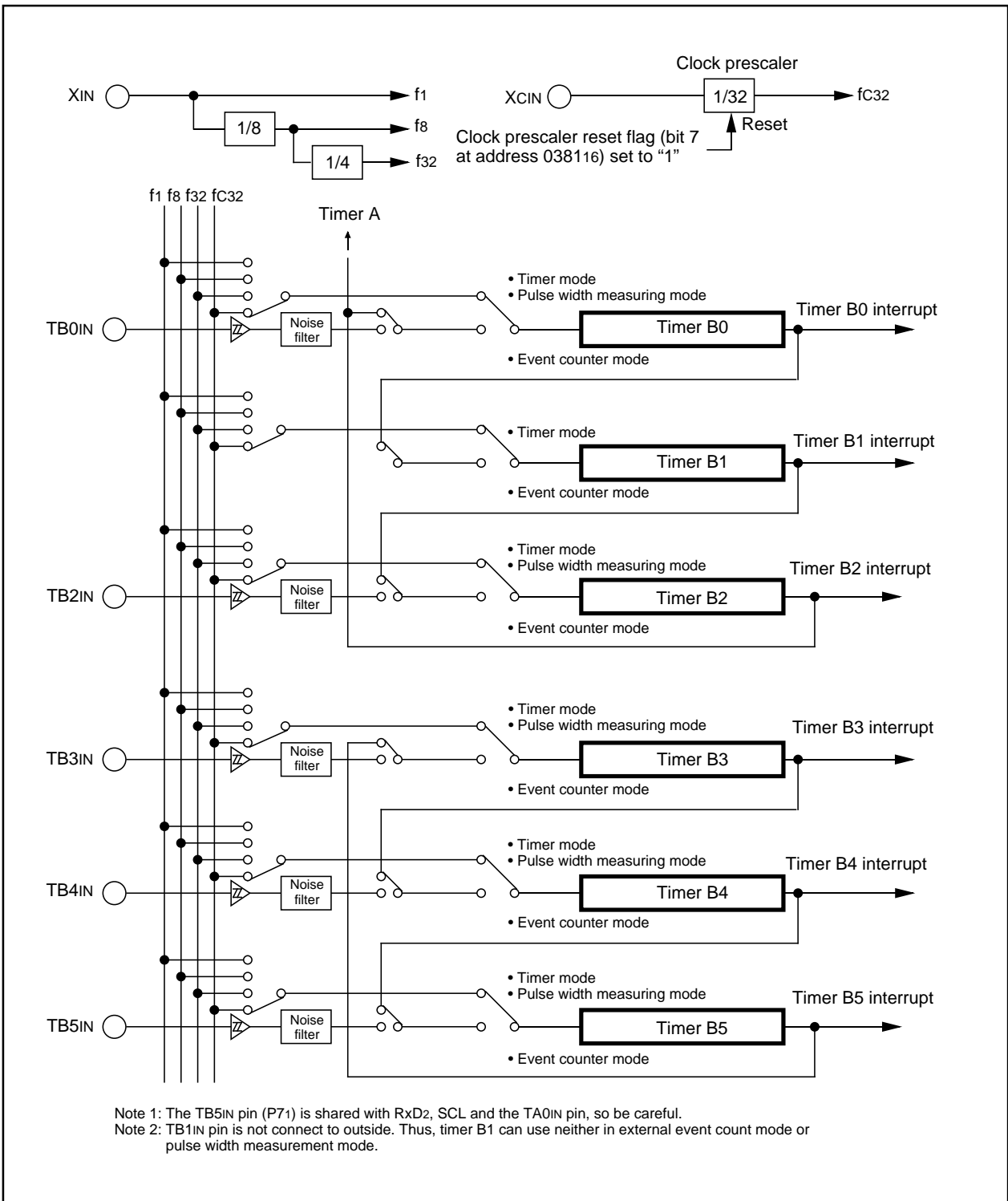


Figure 1.13.2. Timer B block diagram

## Timer A

### Timer A

Figure 1.13.3 shows the block diagram of timer A. Figures 1.13.4 to 1.13.6 show the timer A-related registers. Except in event counter mode, timers A0 through A4 all have the same function. However, in M16C/62A (80-pin version) group, timer A1 and A2 are used for internal timer since timer A1 and A2 have no pin to perform input/output. Use the timer Ai mode register (i = 0 to 4) 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 "0000<sub>16</sub>".
- Pulse width modulation (PWM) mode: The timer outputs pulses of a given width.

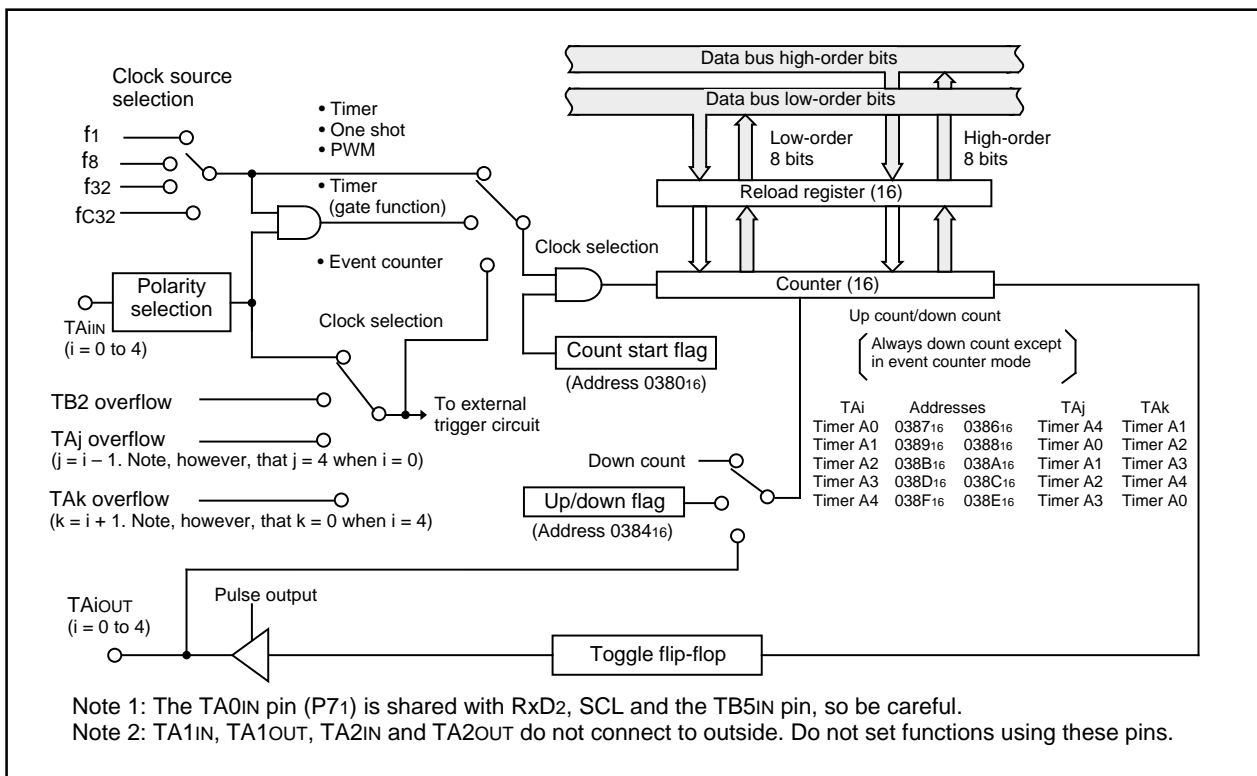


Figure 1.13.3. Block diagram of timer A

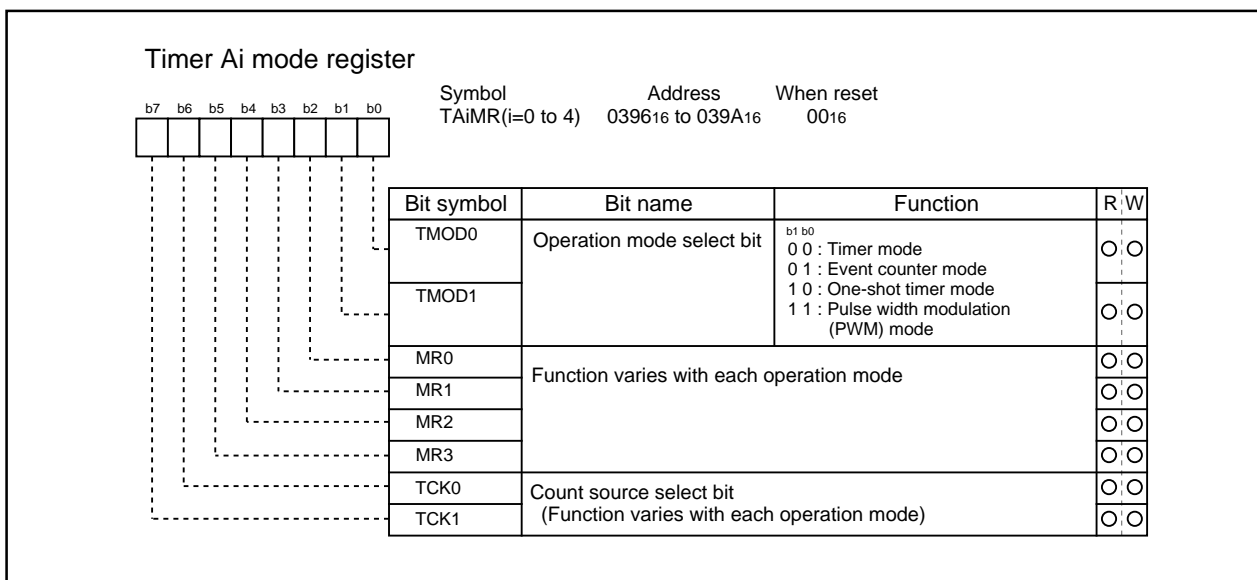


Figure 1.13.4. Timer A-related registers (1)

Timer A

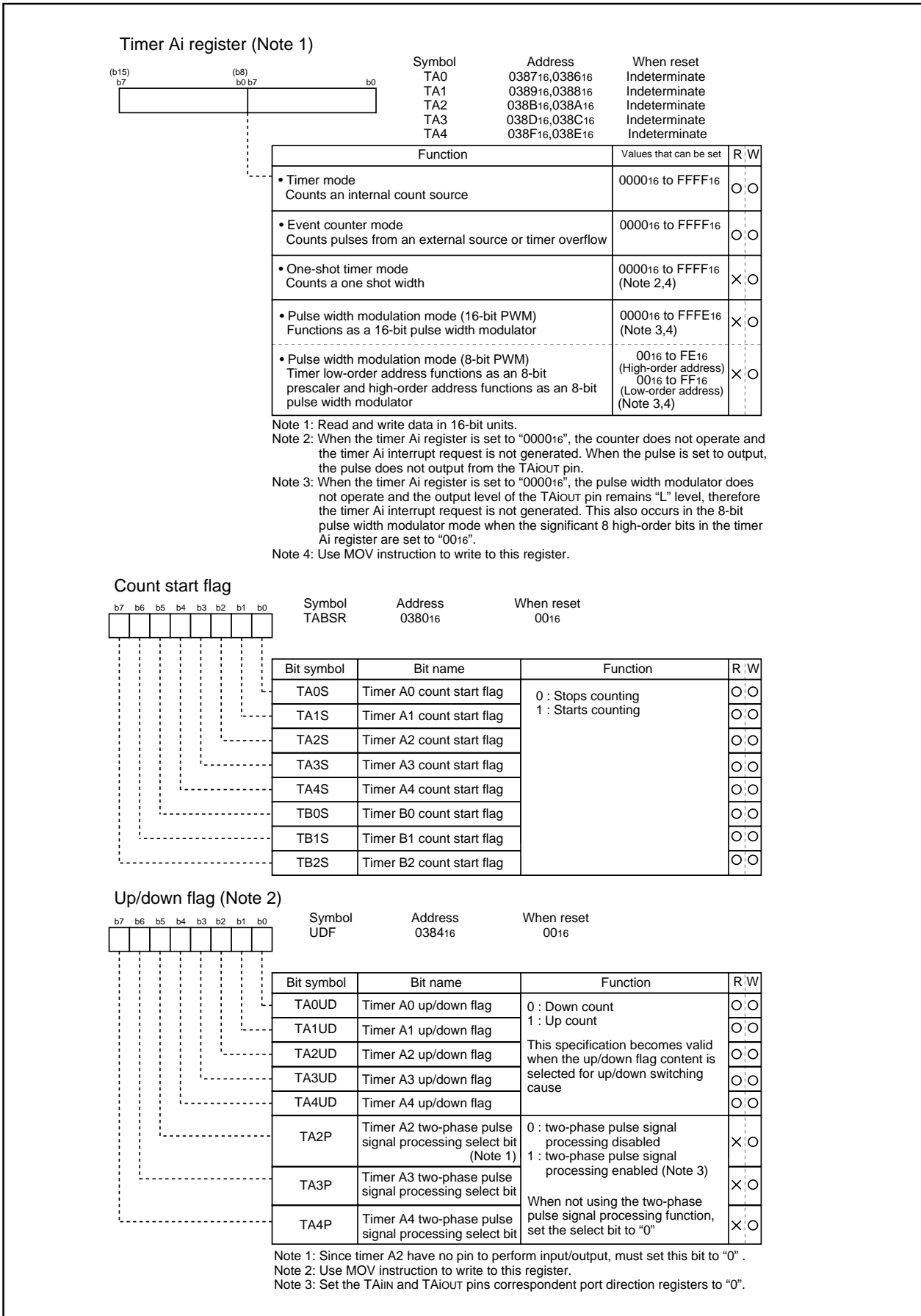


Figure 1.13.5. Timer A-related registers (2)



Timer A

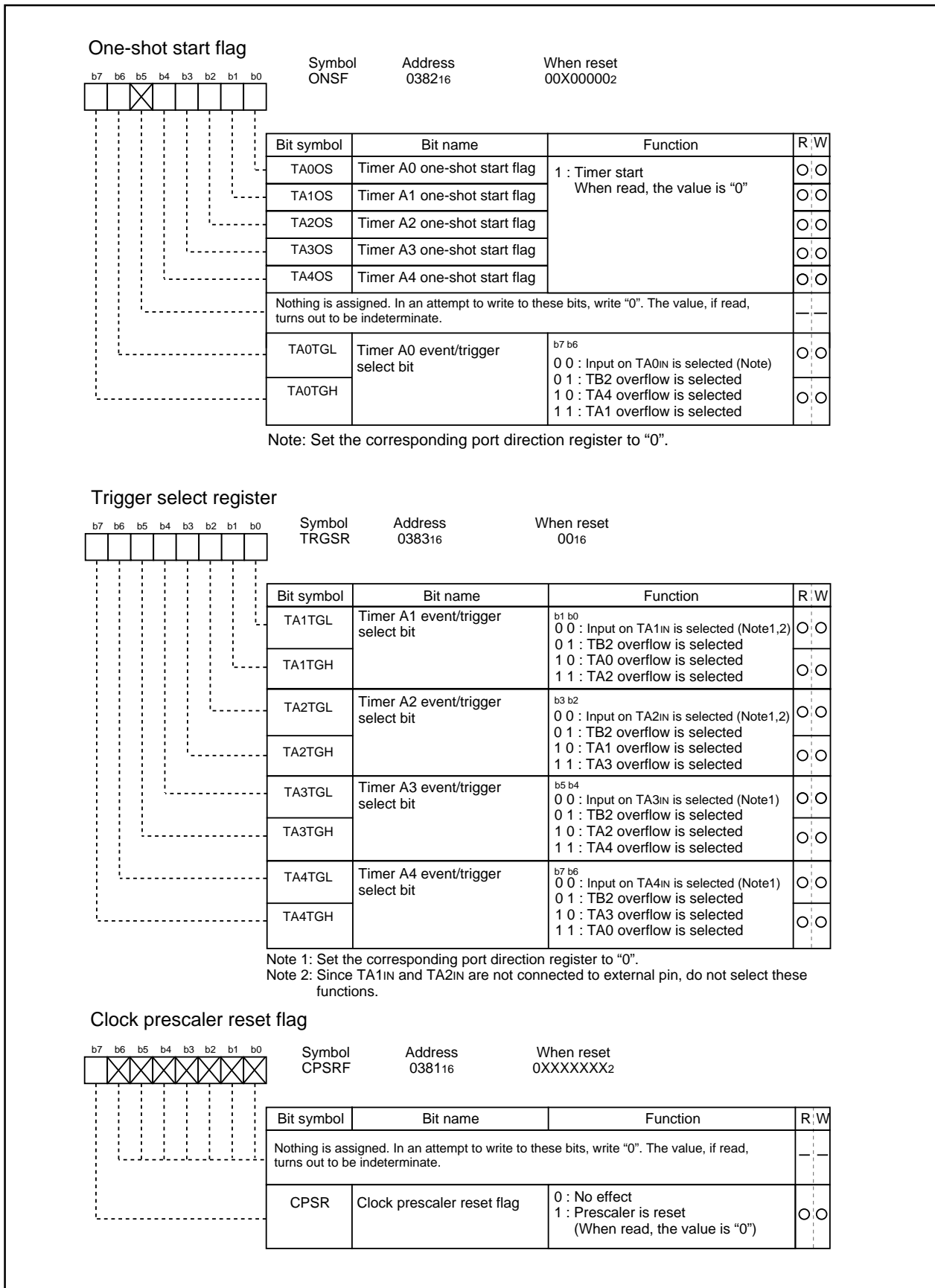


Figure 1.13.6. Timer A-related registers (3)

Timer A

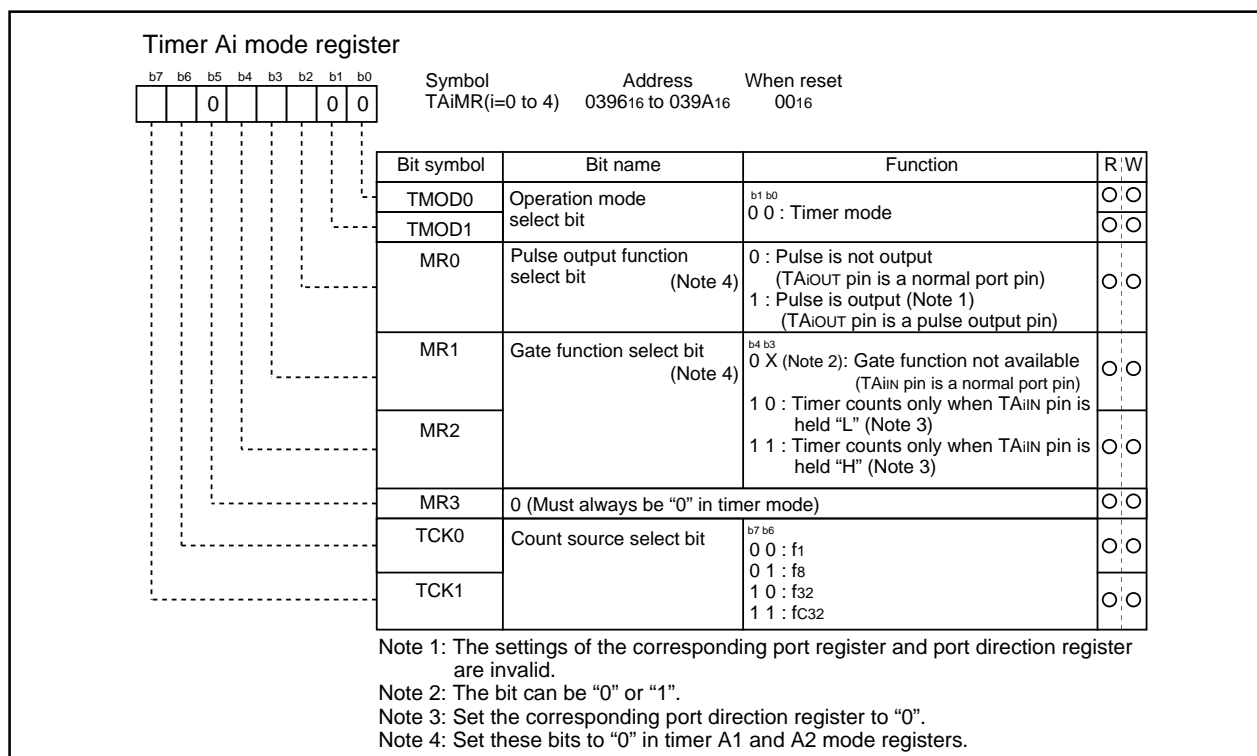
**(1) Timer mode**

In this mode, the timer counts an internally generated count source. (See Table 1.13.1.) Figure 1.13.7 shows the timer Ai mode register in timer mode.

**Table 1.13.1. 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
TAiIN pin function	Programmable I/O port or gate input
TAiOUT pin function	Programmable I/O port or pulse output
Read from timer	Count value can be read out by reading timer Ai register
Write to timer	<ul style="list-style-type: none"> <li>• When counting stopped When a value is written to timer Ai register, it is written to both reload register and counter</li> <li>• When counting in progress When a value is written to timer Ai 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 TAiIN pin's input signal</li> <li>• Pulse output function Each time the timer underflows, the TAiOUT pin's polarity is reversed</li> </ul>

Note: Timer A1 and A2 do not have I/O port (TAiIN and TAiOUT).



**Figure 1.13.7. Timer Ai 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. Timers A0 and A1 can count a single-phase external signal. Timers A2, A3, and A4 can count a single-phase and a two-phase external signal. Table 1.13.2 lists timer specifications when counting a single-phase external signal. Figure 1.13.8 shows the timer Ai mode register in event counter mode.

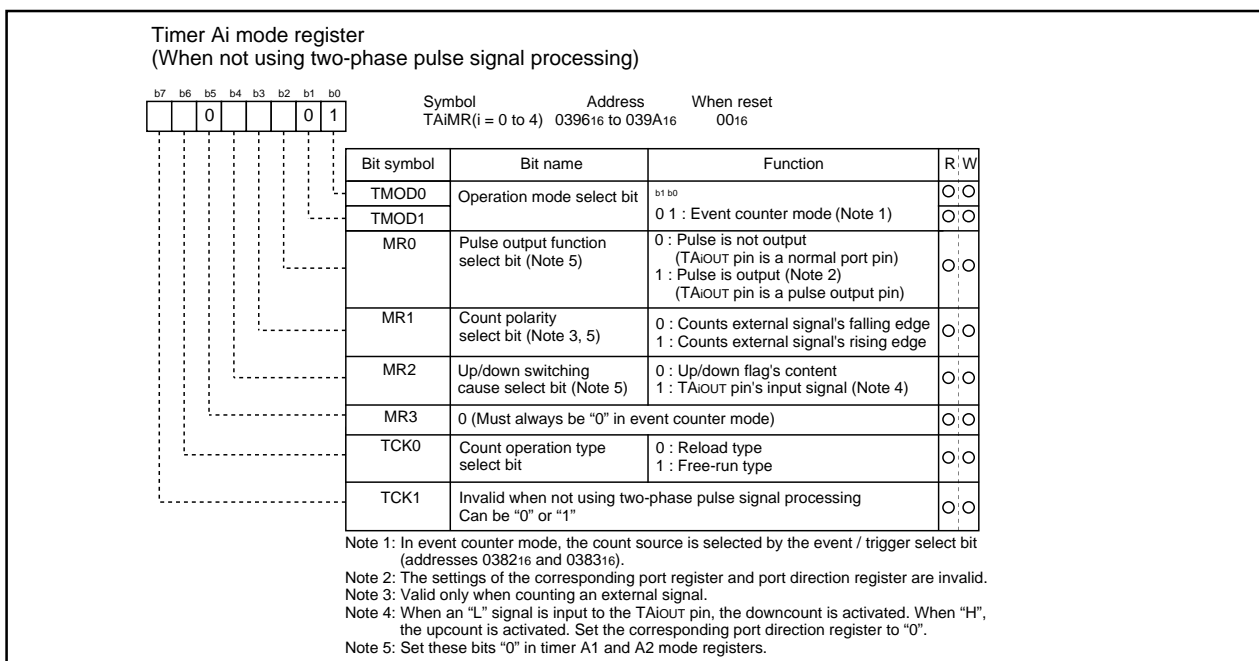
Table 1.13.3 lists timer specifications when counting a two-phase external signal. Figure 1.13.9 shows the timer Ai mode register in event counter mode.

**Table 1.13.2. 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 TAIin pin (effective edge can be selected by software)</li> <li>TB2 overflow, TAJ 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 <sub>16</sub> - 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
TAiin pin function	Programmable I/O port or count source input
TAiOUT 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 Ai register
Write to timer	<ul style="list-style-type: none"> <li>When counting stopped When a value is written to timer Ai register, it is written to both reload register and counter</li> <li>When counting in progress When a value is written to timer Ai 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 TAIOUT pin's polarity is reversed</li> </ul>

Note 1: This does not apply when the free-run function is selected.

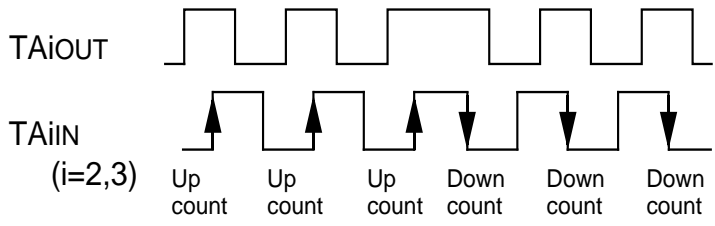
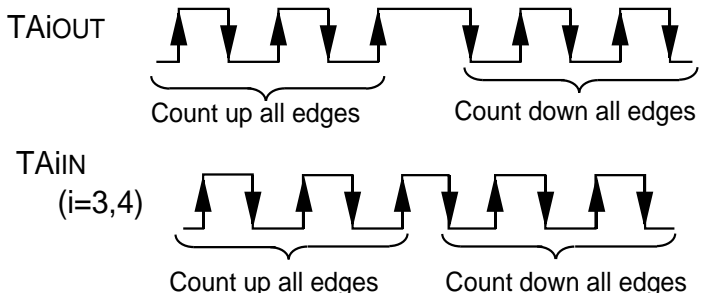
Note 2: Timer A1 and A2 do not have I/O port (TAiin and TAIOUT).



**Figure 1.13.8. Timer Ai mode register in event counter mode**

Timer A

**Table 1.13.3. Timer specifications in event counter mode (when processing two-phase pulse signal with timers A2, A3, and A4)**

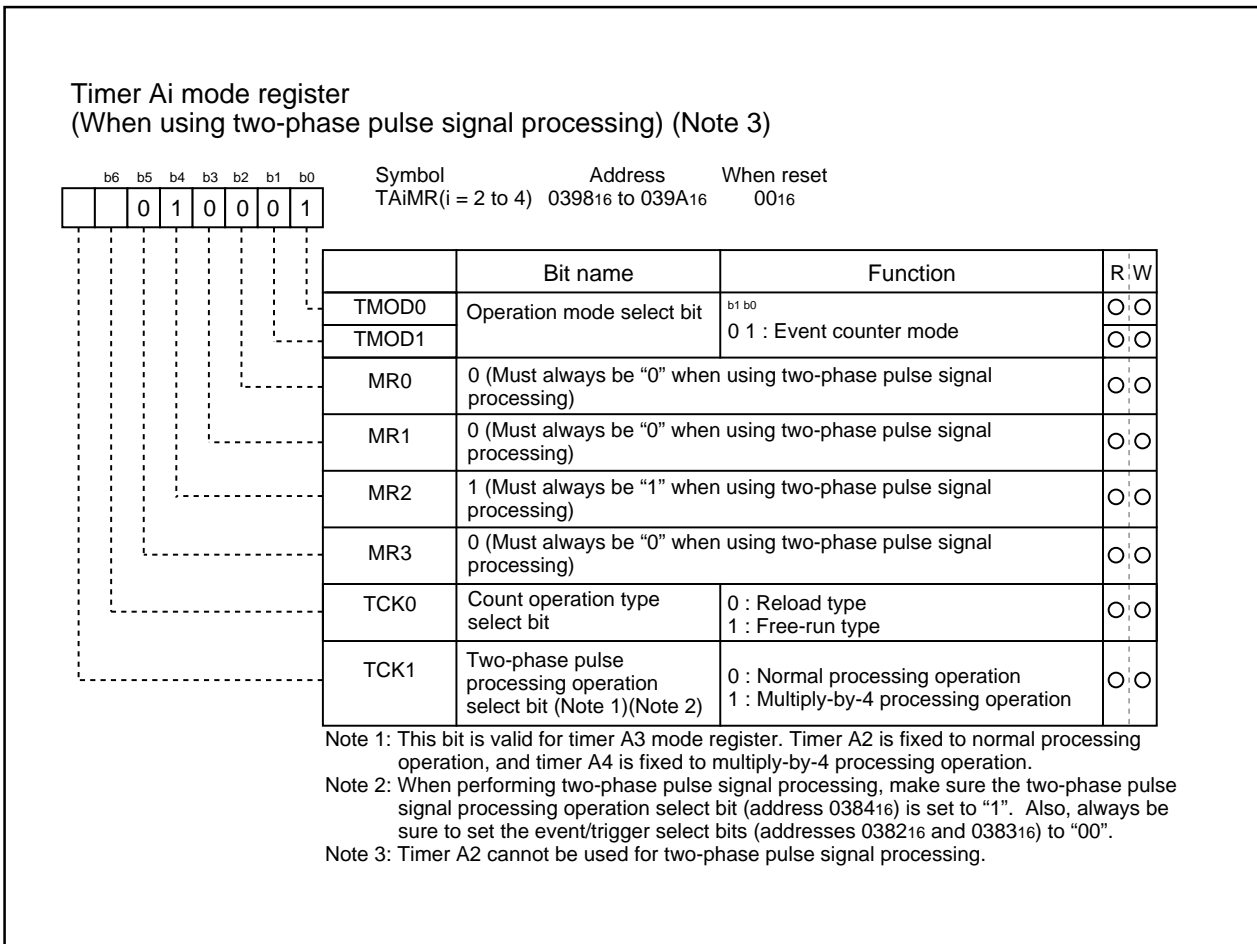
Item	Specification
Count source	• Two-phase pulse signals input to TAIiN or TAIiOUT pin
Count operation	• Up count or down count can be selected by two-phase pulse signal • When the timer overflows or underflows, the reload register content is reloaded and the timer starts over again (Note)
Divide ratio	1/ (FFFF <sub>16</sub> - 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	Timer overflows or underflows
TAiIN pin function	Two-phase pulse input (Set the TAIiN pin correspondent port direction register to "0")
TAiOUT pin function	Two-phase pulse input (Set the TAIiOUT pin correspondent port direction register to "0")
Read from timer	Count value can be read out by reading timer A2, A3, or A4 register
Write to timer	• When counting stopped When a value is written to timer A2, A3, or A4 register, it is written to both reload register and counter • When counting in progress When a value is written to timer A2, A3, or A4 register, it is written to only reload register. (Transferred to counter at next reload time.)
Select function (Note 3)	• Normal processing operation (timer A2 and timer A3) The timer counts up rising edges or counts down falling edges on the TAIiN pin when input signal on the TAIiOUT pin is "H"  • Multiply-by-4 processing operation (timer A3 and timer A4) If the phase relationship is such that the TAIiN pin goes "H" when the input signal on the TAIiOUT pin is "H", the timer counts up rising and falling edges on the TAIiOUT and TAIiN pins. If the phase relationship is such that the TAIiN pin goes "L" when the input signal on the TAIiOUT pin is "H", the timer counts down rising and falling edges on the TAIiOUT and TAIiN pins. 

Note 1: This does not apply when the free-run function is selected.

Note 2: Timer A1 and A2 do not have I/O port (TAiIN and TAIiOUT).

Note 3: Timer A3 alone can be selected. Timer A2 is fixed to normal processing operation, and timer A4 is fixed to multiply-by-4 processing operation.

Timer A



**Figure 1.13.9. Timer Ai mode register in event counter mode**

Timer A

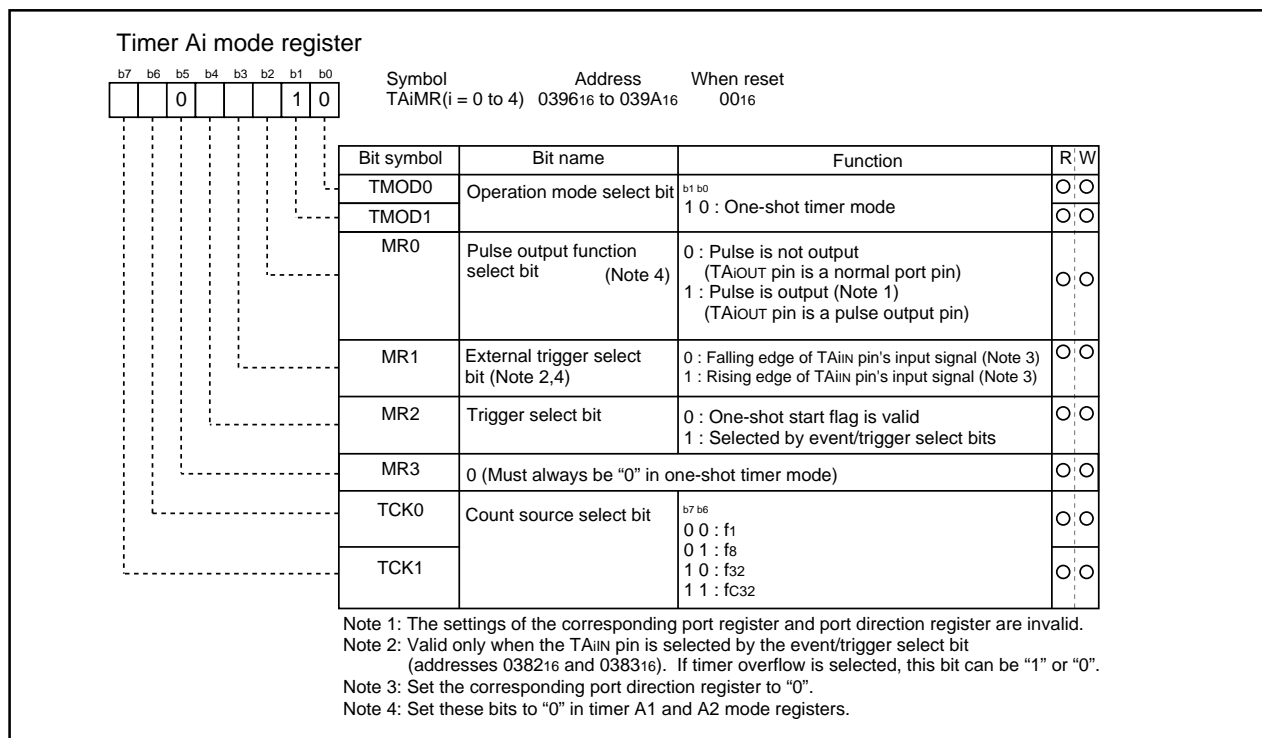
**(3) One-shot timer mode**

In this mode, the timer operates only once. (See Table 1.13.4.) When a trigger occurs, the timer starts up and continues operating for a given period. Figure 1.13.10 shows the timer Ai mode register in one-shot timer mode.

**Table 1.13.4. 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>
TAiIN pin function	Programmable I/O port or trigger input
TAiOUT pin function	Programmable I/O port or pulse output
Read from timer	When timer Ai 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 Ai register, it is written to both reload register and counter</li> <li>When counting in progress When a value is written to timer Ai register, it is written to only reload register (Transferred to counter at next reload time)</li> </ul>

Note: Timer A1 and A2 do not have I/O port (TAiIN and TAiOUT).



**Figure 1.13.10. Timer Ai mode register in one-shot timer mode**

Timer A

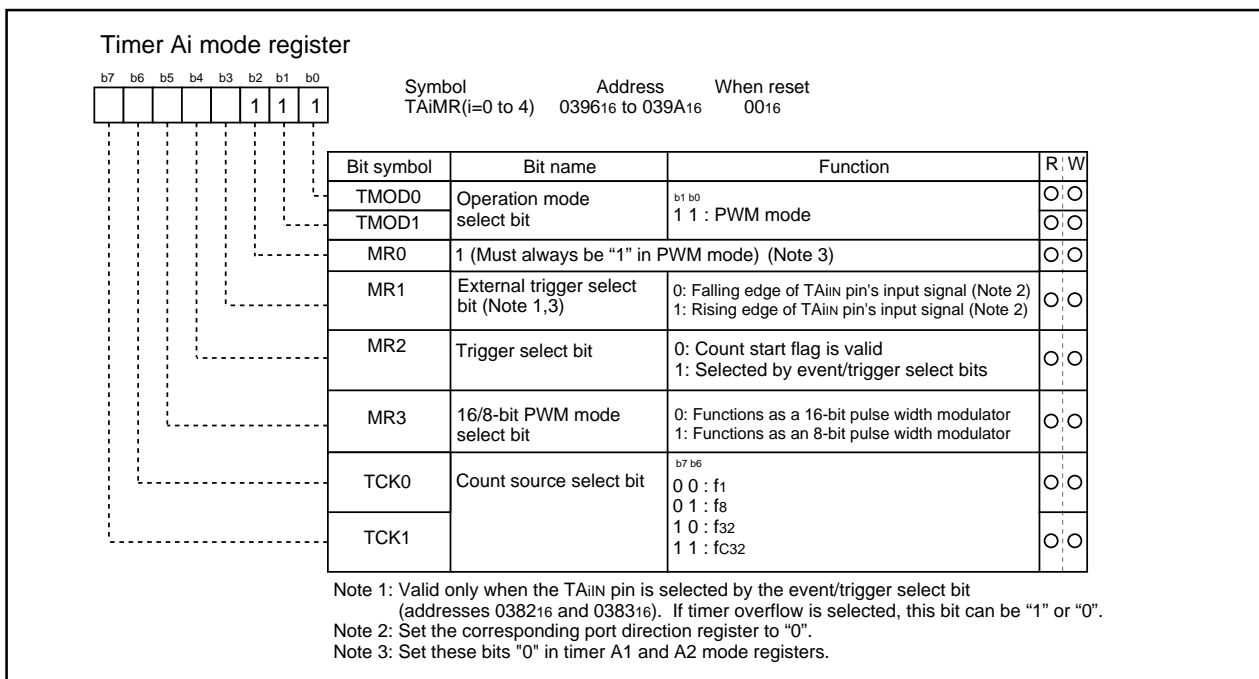
**(4) Pulse width modulation (PWM) mode**

In this mode, the timer outputs pulses of a given width in succession. (See Table 1.13.5.) In this mode, the counter functions as either a 16-bit pulse width modulator or an 8-bit pulse width modulator. Timer A1 and A2 have no output pin, so it doesn't work in this mode. Figure 1.13.11 shows the timer Ai mode register in pulse width modulation mode. Figure 1.13.12 shows the example of how a 16-bit pulse width modulator operates. Figure 1.13.13 shows the example of how an 8-bit pulse width modulator operates.

**Table 1.13.5. 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 Ai register's high-order address</li> <li>Cycle time <math>(2^8-1) \times (m+1) / f_i</math> m : values set to timer Ai 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	PWM pulse goes "L"
TAiIN pin function	Programmable I/O port or trigger input
TAiOUT pin function	Pulse output
Read from timer	When timer Ai 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 Ai register, it is written to both reload register and counter</li> <li>When counting in progress When a value is written to timer Ai register, it is written to only reload register (Transferred to counter at next reload time)</li> </ul>

Note: Timer A1 and A2 do not have I/O port (TAiIN and TAiOUT).



**Figure 1.13.11. Timer Ai mode register in pulse width modulation mode**

Timer A

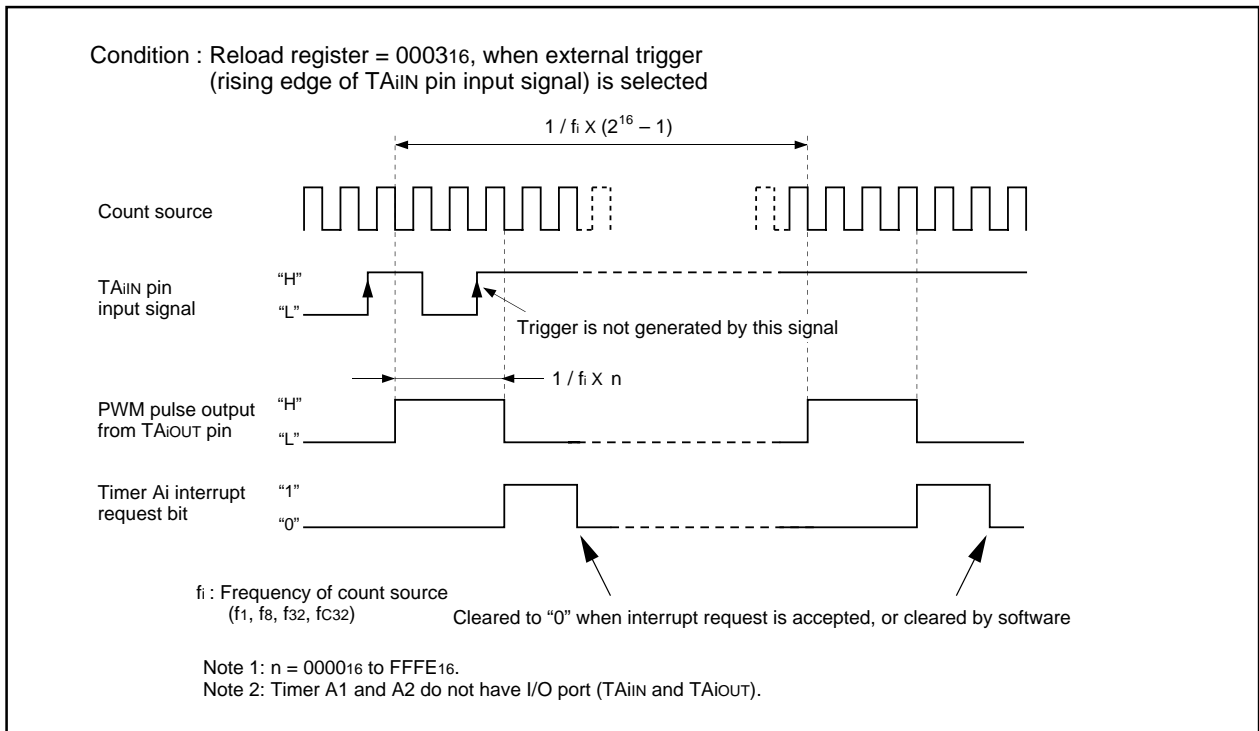


Figure 1.13.12. Example of how a 16-bit pulse width modulator operates

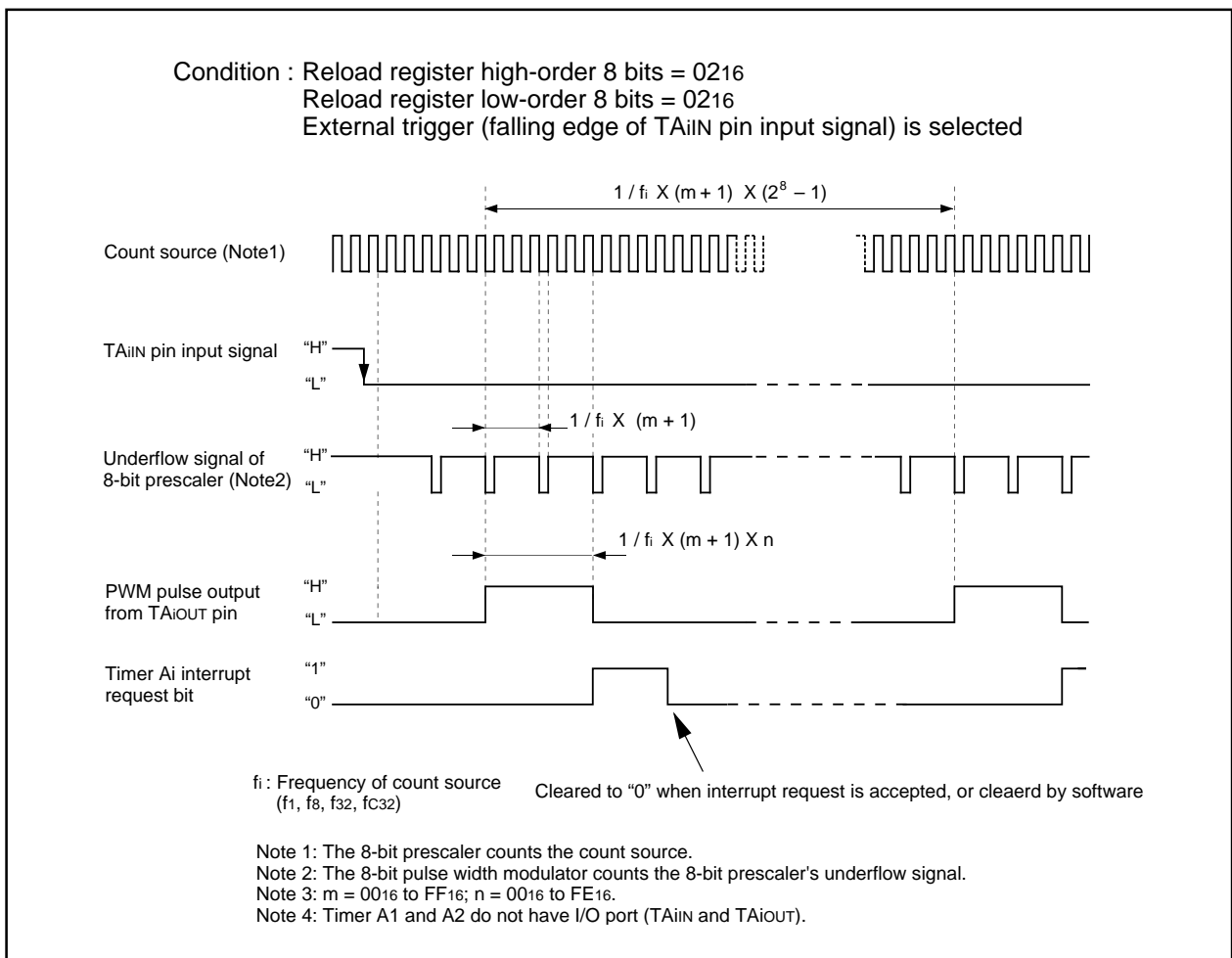


Figure 1.13.13. Example of how an 8-bit pulse width modulator operates



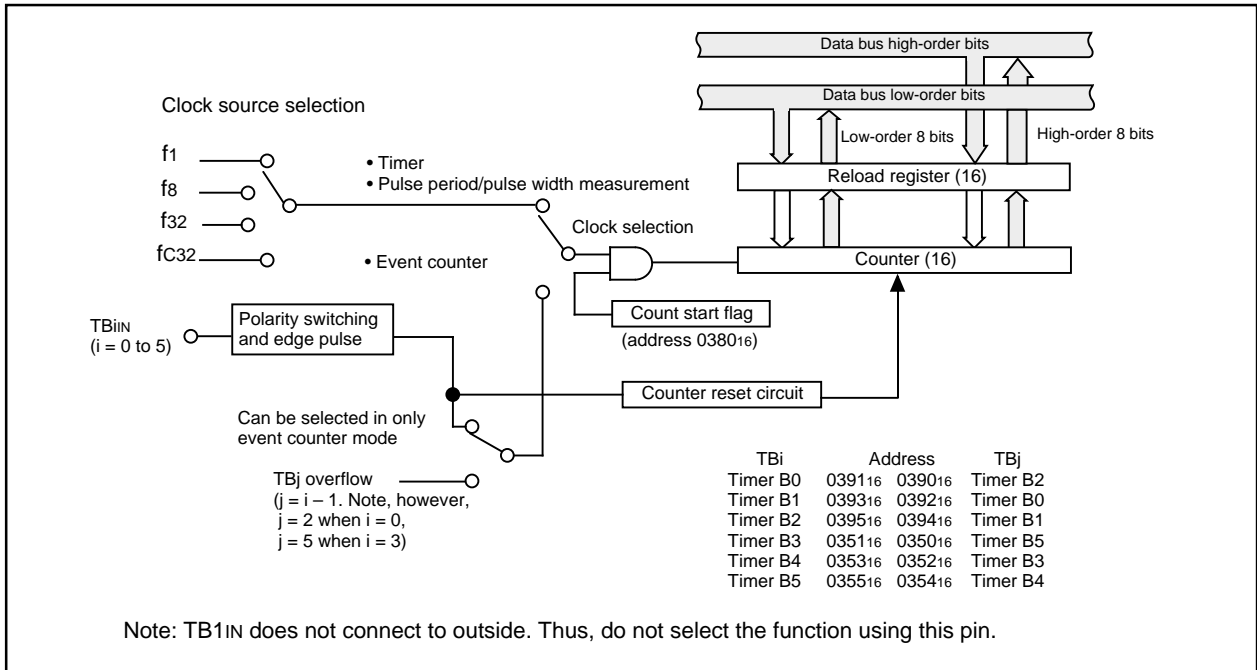
Timer B

**Timer B**

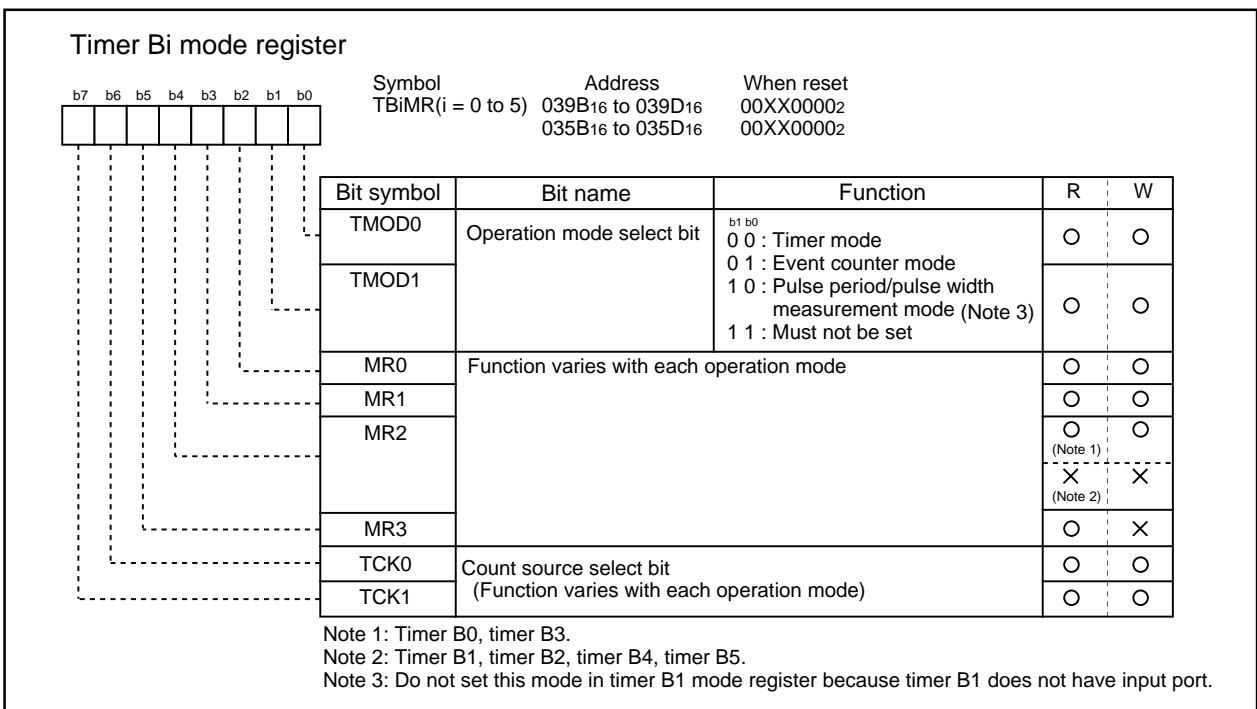
Figure 1.13.14 shows the block diagram of timer B. Figures 1.13.15 and 1.13.16 show the timer B-related registers. However, timer B1 is used for internal timer since timer B1 does not have input port. Use the timer Bi mode register (i = 0 to 5) bits 0 and 1 to choose the desired mode.

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.

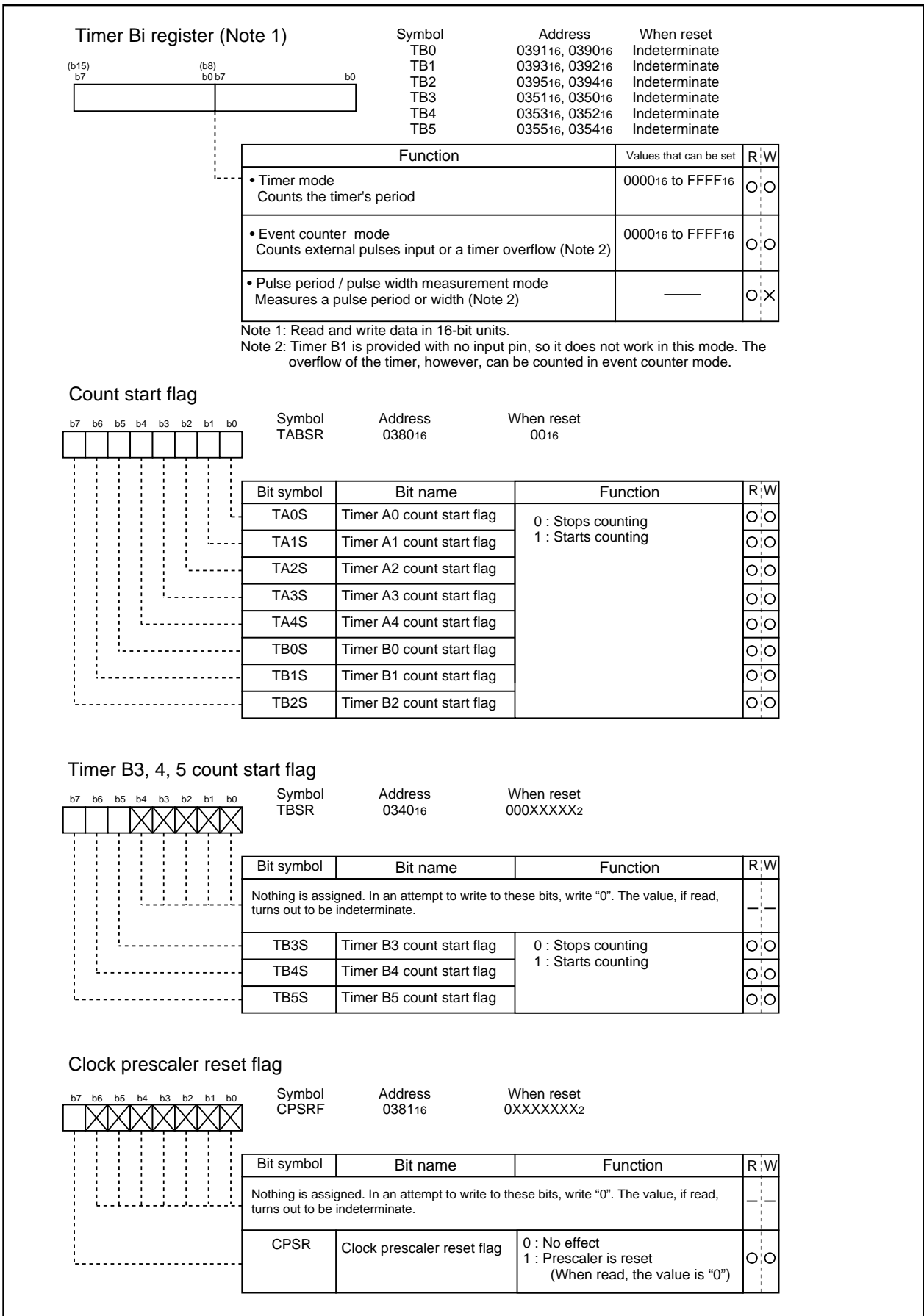


**Figure 1.13.14. Block diagram of timer B**



**Figure 1.13.15. Timer B-related registers (1)**

Timer B



Timer B

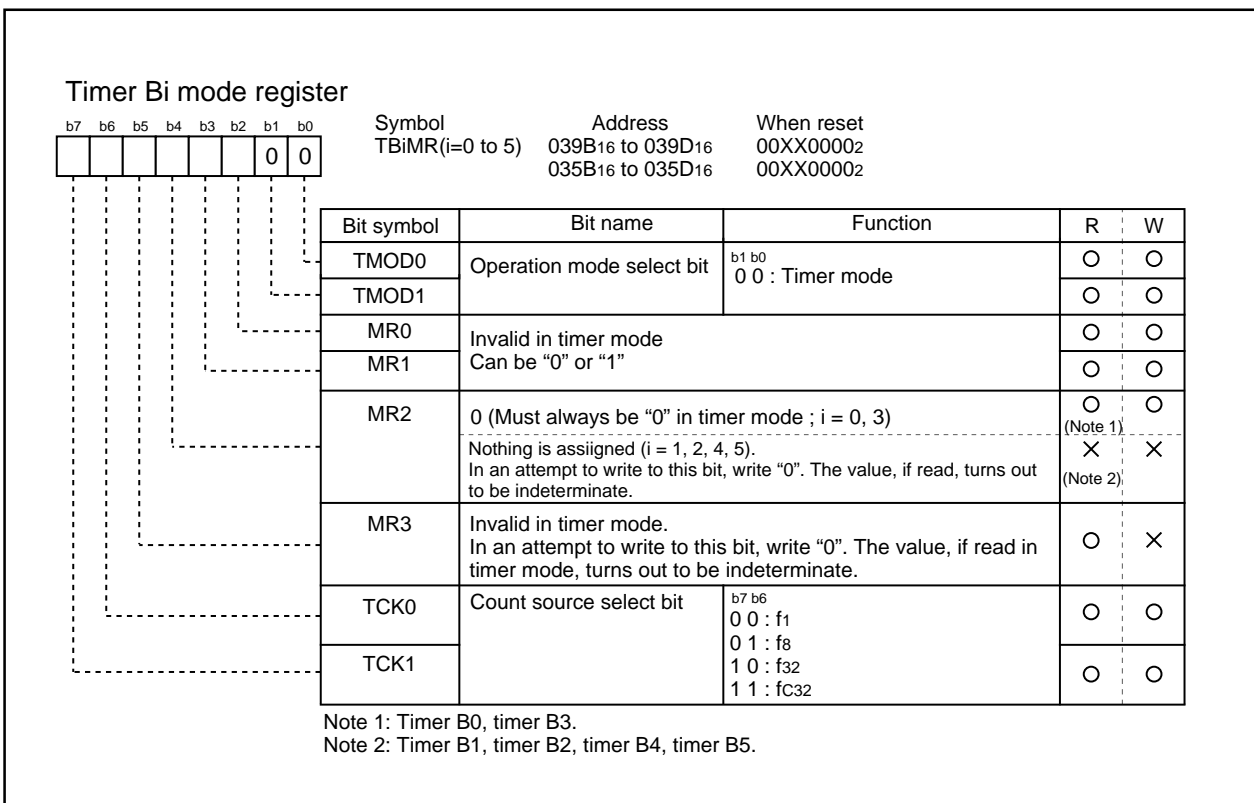
**(1) Timer mode**

In this mode, the timer counts an internally generated count source. (See Table 1.13.6.) Figure 1.13.17 shows the timer Bi mode register in timer mode.

**Table 1.13.6. 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>

Note: Timer B1 works exclusively as an internal timer since timer B1 does not have input port (TB1IN).



**Figure 1.13.17. Timer Bi mode register in timer mode**

Timer B

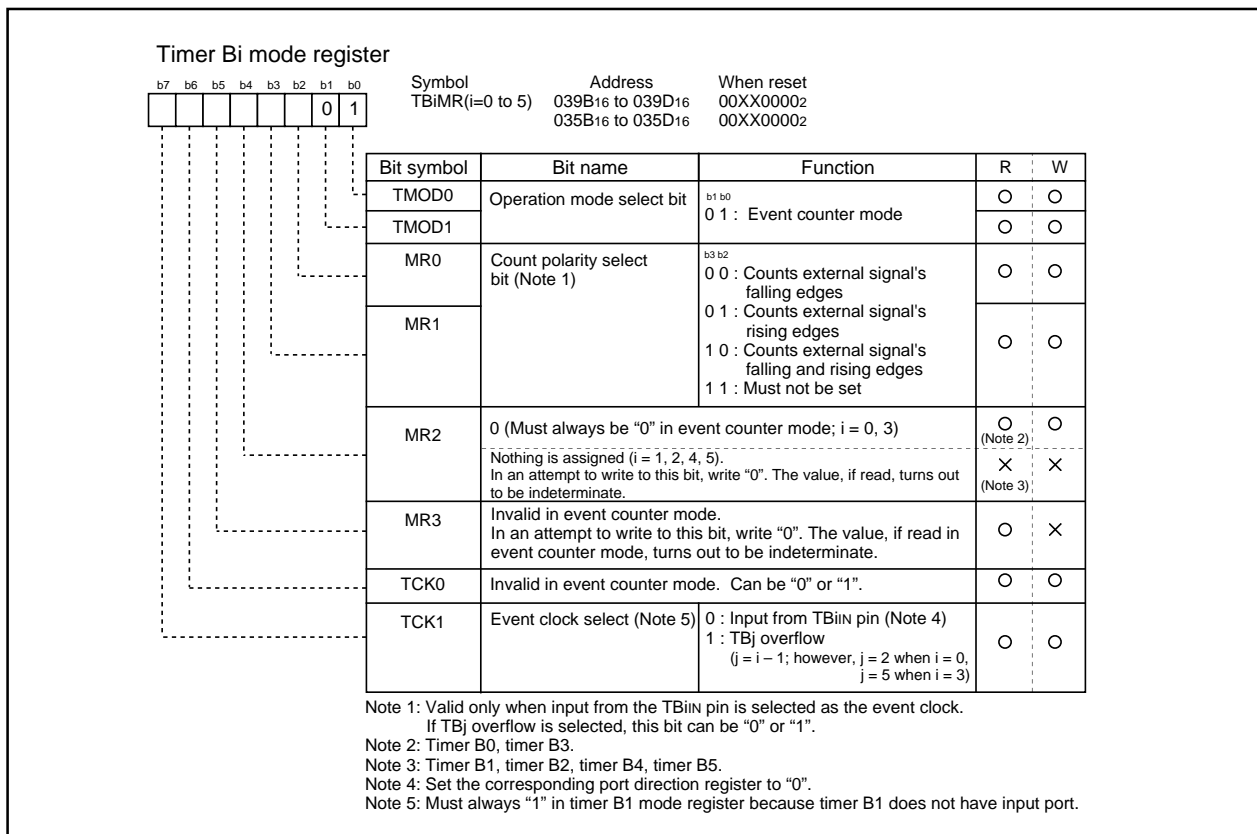
**(2) Event counter mode**

In this mode, the timer counts an external signal or an internal timer's overflow. (See Table 1.13.7.)  
 However, timer B1 works exclusively as an internal timer because timer B1 does not have input port.  
 Figure 1.13.18 shows the timer Bi mode register in event counter mode.

**Table 1.13.7. 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>

Note: Timer B1 works exclusively as an internal timer since timer B1 does not have input port (TB1IN).



**Figure 1.13.18. Timer Bi mode register in event counter mode**

Timer B

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

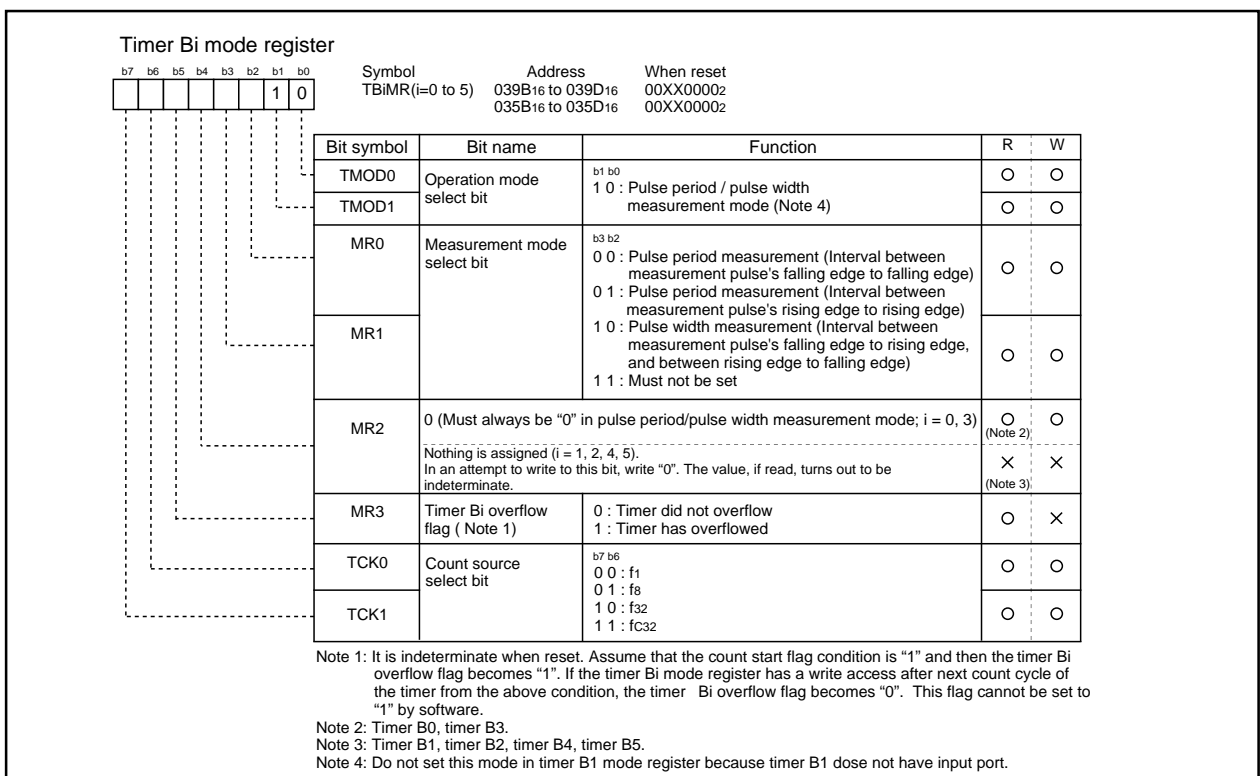
In this mode, the timer measures the pulse period or pulse width of an external signal. (See Table 1.13.8.) However, this function cannot be used since timer B1 does not have input port. Figure 1.13.19 shows the timer Bi mode register in pulse period/pulse width measurement mode. Figure 1.13.20 shows the operation timing when measuring a pulse period. Figure 1.13.21 shows the operation timing when measuring a pulse width.

**Table 1.13.8. 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 "0000<sub>16</sub>" 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". Assume that the count start flag condition is "1" and then the timer Bi overflow flag becomes "1". If the timer Bi mode register has a write access after next count cycle of the timer from the above condition, the timer Bi overflow flag becomes "0".)</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

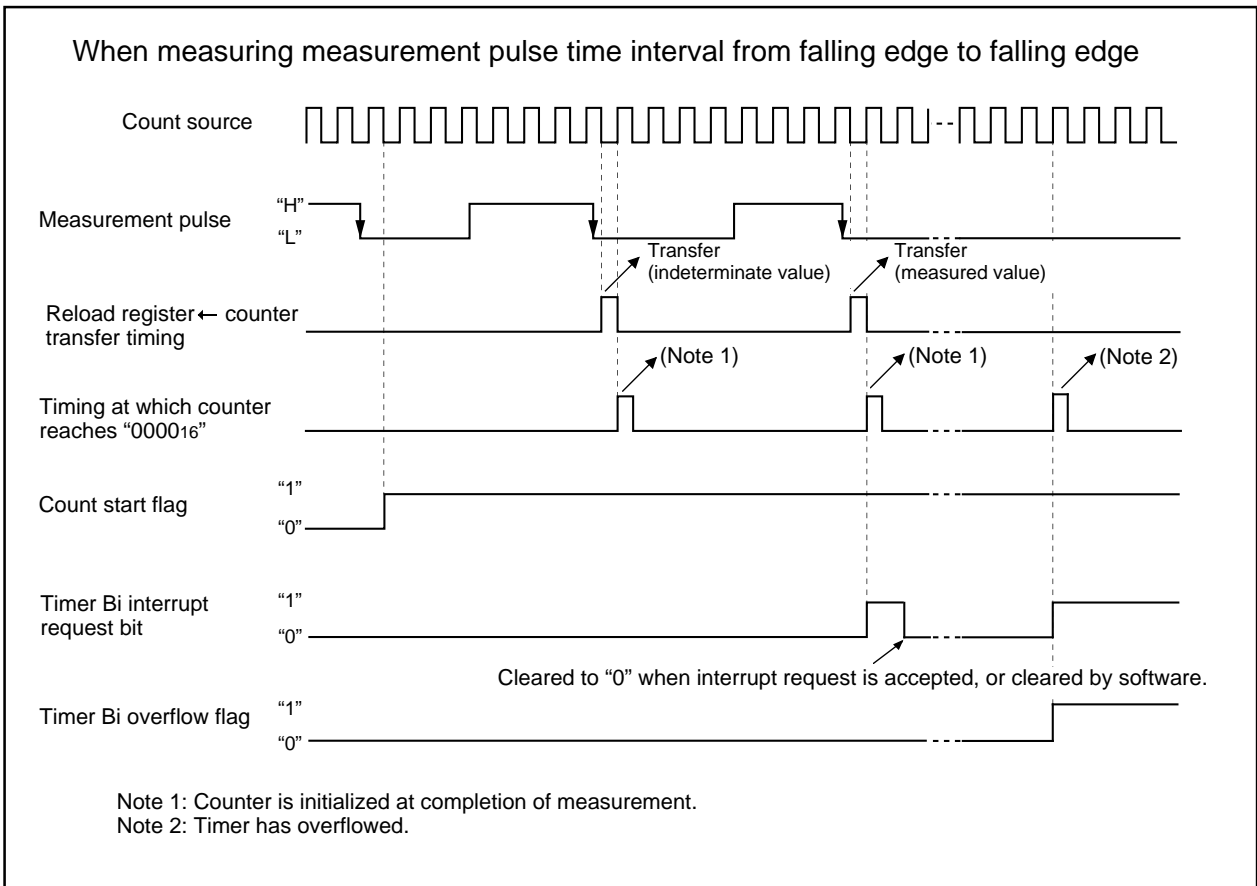
Note 1: An interrupt request is not generated when the first effective edge is input after the timer has started counting.

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

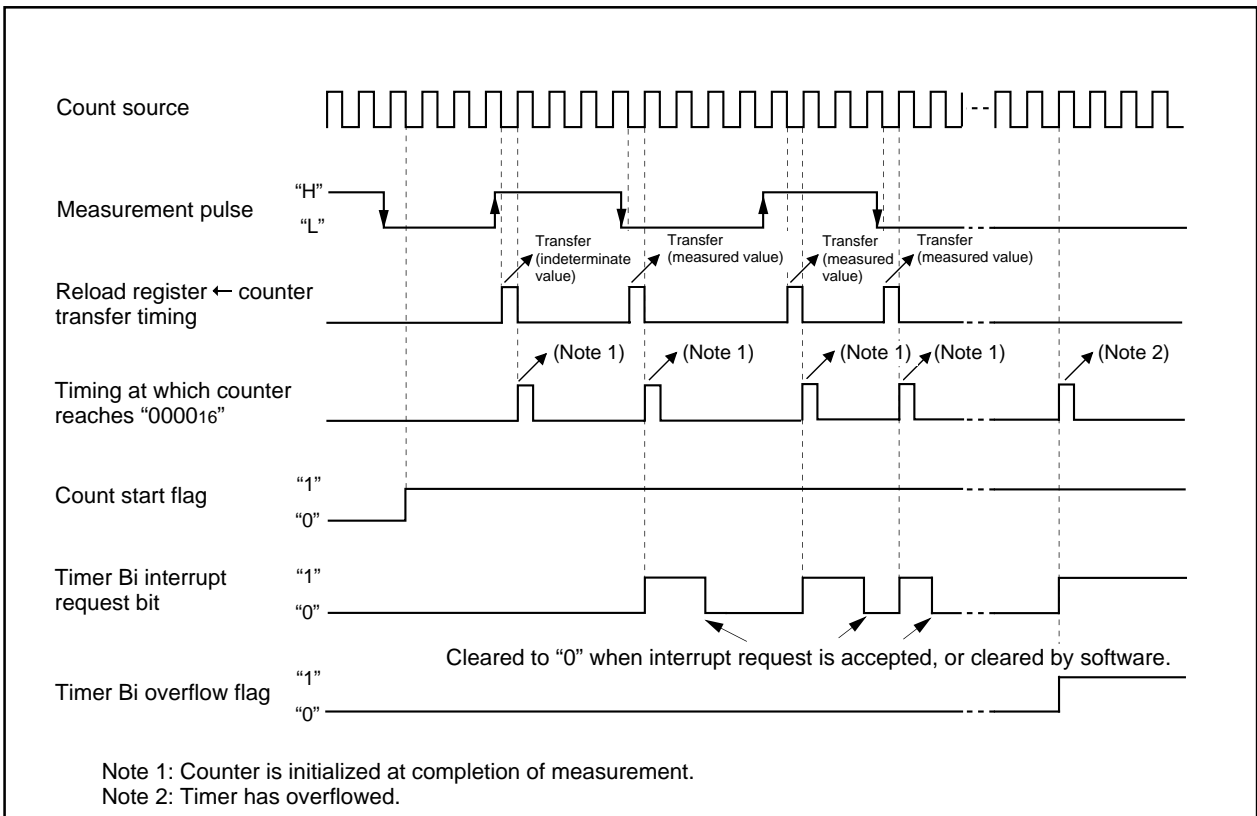


**Figure 1.13.19. Timer Bi mode register in pulse period/pulse width measurement mode**

Timer B



**Figure 1.13.20. Operation timing when measuring a pulse period**



**Figure 1.13.21. Operation timing when measuring a pulse width**

## Serial I/O

Serial I/O is configured as five channels: UART0, UART1, UART2, S I/O3 and S I/O4.

### UART0 to 2

UART0, UART1 and UART2 each have an exclusive timer to generate a transfer clock, so they operate independently of each other.

Figure 1.14.1 shows the block diagram of UART0, UART1 and UART2. Figures 1.14.2 and 1.14.3 show the block diagram of the transmit/receive unit.

UART<sub>i</sub> (i = 0 to 2) 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 03A0<sub>16</sub>, 03A8<sub>16</sub> and 0378<sub>16</sub>) determine whether UART<sub>i</sub> is used as a clock synchronous serial I/O or as a UART.

UART0 through UART2 are almost equal in their functions with minor exceptions. UART2, in particular, is used for the SIM interface with some extra settings added in clock-asynchronous serial I/O mode (Note). It also has the bus collision detection function that generates an interrupt request if the Tx<sub>D</sub> pin and the Rx<sub>D</sub> pin are different in level. UART and IIC mode can be used in UART2.

Table 1.14.1 shows the comparison of functions of UART0 through UART2, and Figures 1.14.4 to 1.14.9 show the registers related to UART<sub>i</sub>.

Note: SIM : Subscriber Identity Module

**Table 1.14.1. Comparison of functions of UART0 through UART2**

Function	UART0	UART1	UART2
CLK polarity selection	Possible (Note 1)	Possible (Note 1)	Possible (Note 5)
LSB first / MSB first selection	Possible (Note 1)	Possible (Note 1)	Possible (Note 2)
Continuous receive mode selection	Possible (Note 1)	Possible (Note 1)	Possible (Note 5)
Transfer clock output from multiple pins selection	Impossible	Possible (Note 1)	Impossible
Serial data logic switch	Impossible	Impossible	Possible (Note 4)
Sleep mode selection	Possible (Note 3)	Possible (Note 3)	Impossible
TxD, Rx <sub>D</sub> I/O polarity switch	Impossible	Impossible	Possible
TxD, Rx <sub>D</sub> port output format	CMOS output	CMOS output	N-channel open-drain output (Note 6)
Parity error signal output	Impossible	Impossible	Possible (Note 4)
Bus collision detection	Impossible	Impossible	Possible

Note 1: Only when clock synchronous serial I/O mode.

Note 2: Only when clock synchronous serial I/O mode and 8-bit UART mode.

Note 3: Only when UART mode.

Note 4: Using for SIM interface.

Note 5: Since CLK<sub>2</sub> and CTS<sub>2</sub>/RTS<sub>2</sub> do not connect to outside, this function cannot be used.

Note 6: Connect this pin to V<sub>cc</sub> via a pull-up resistor on the outside.

Serial I/O

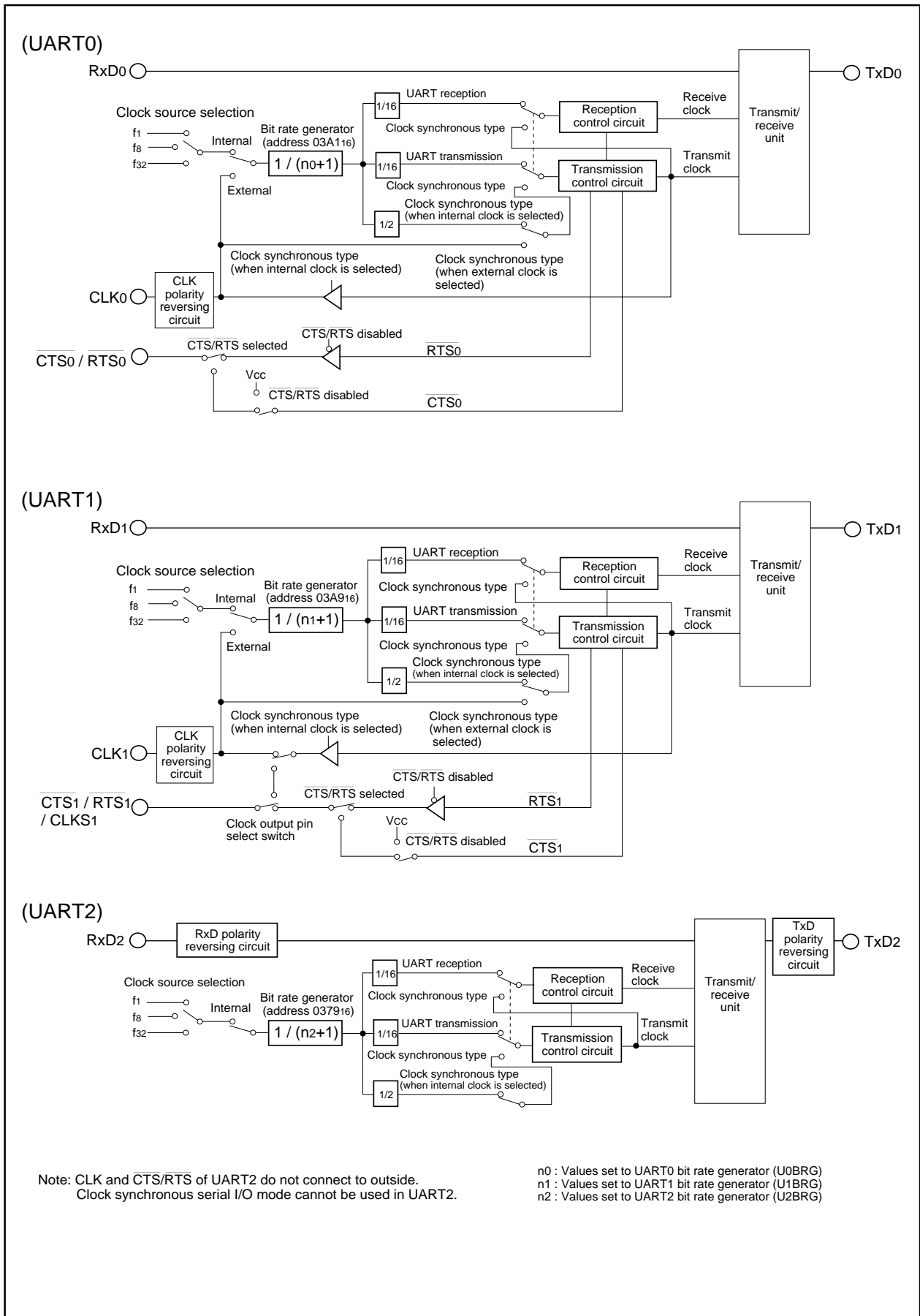


Figure 1.14.1. Block diagram of UARTi (i = 0 to 2)



Serial I/O

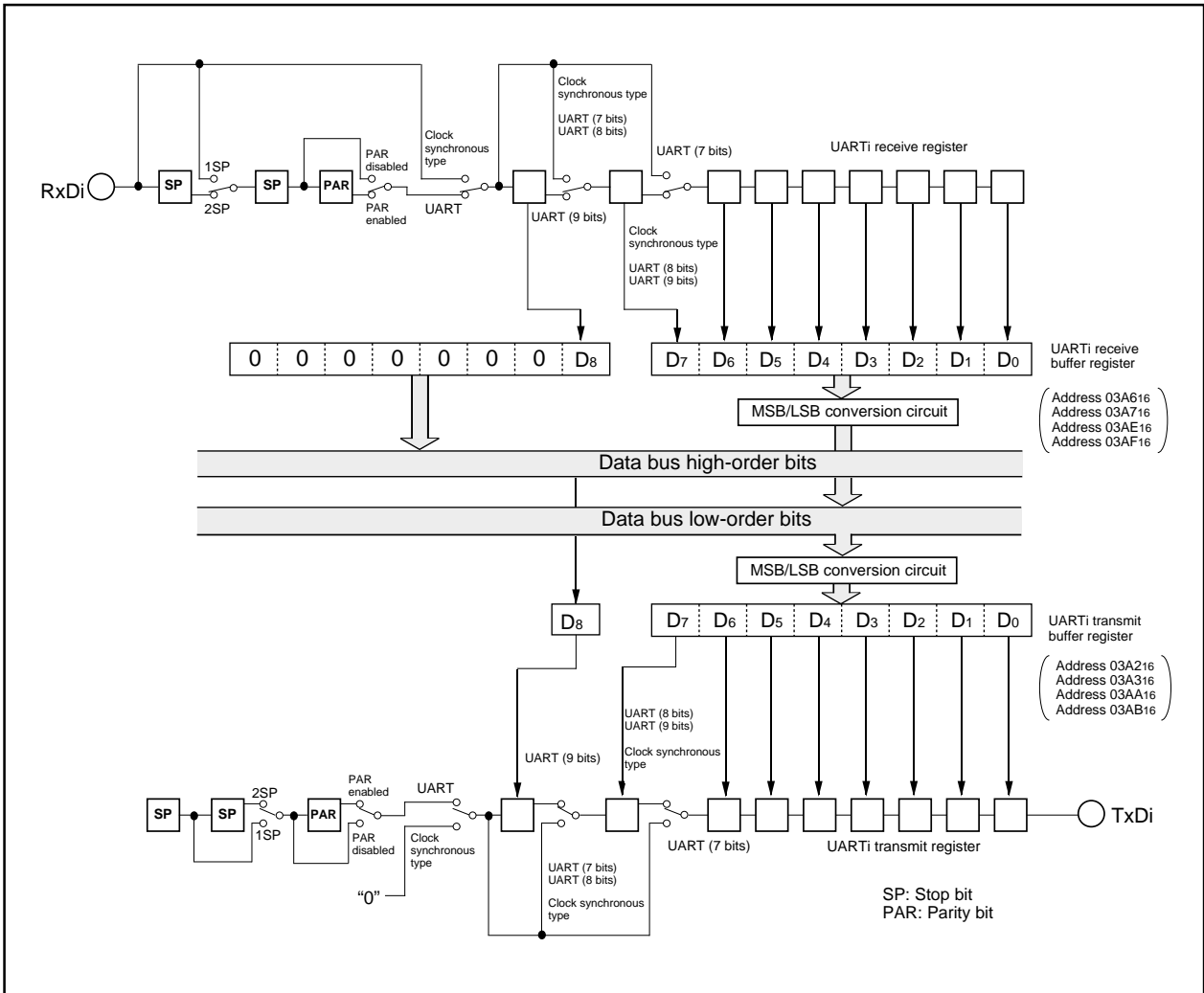


Figure 1.14.2. Block diagram of UARTi (i = 0, 1) transmit/receive unit

Serial I/O

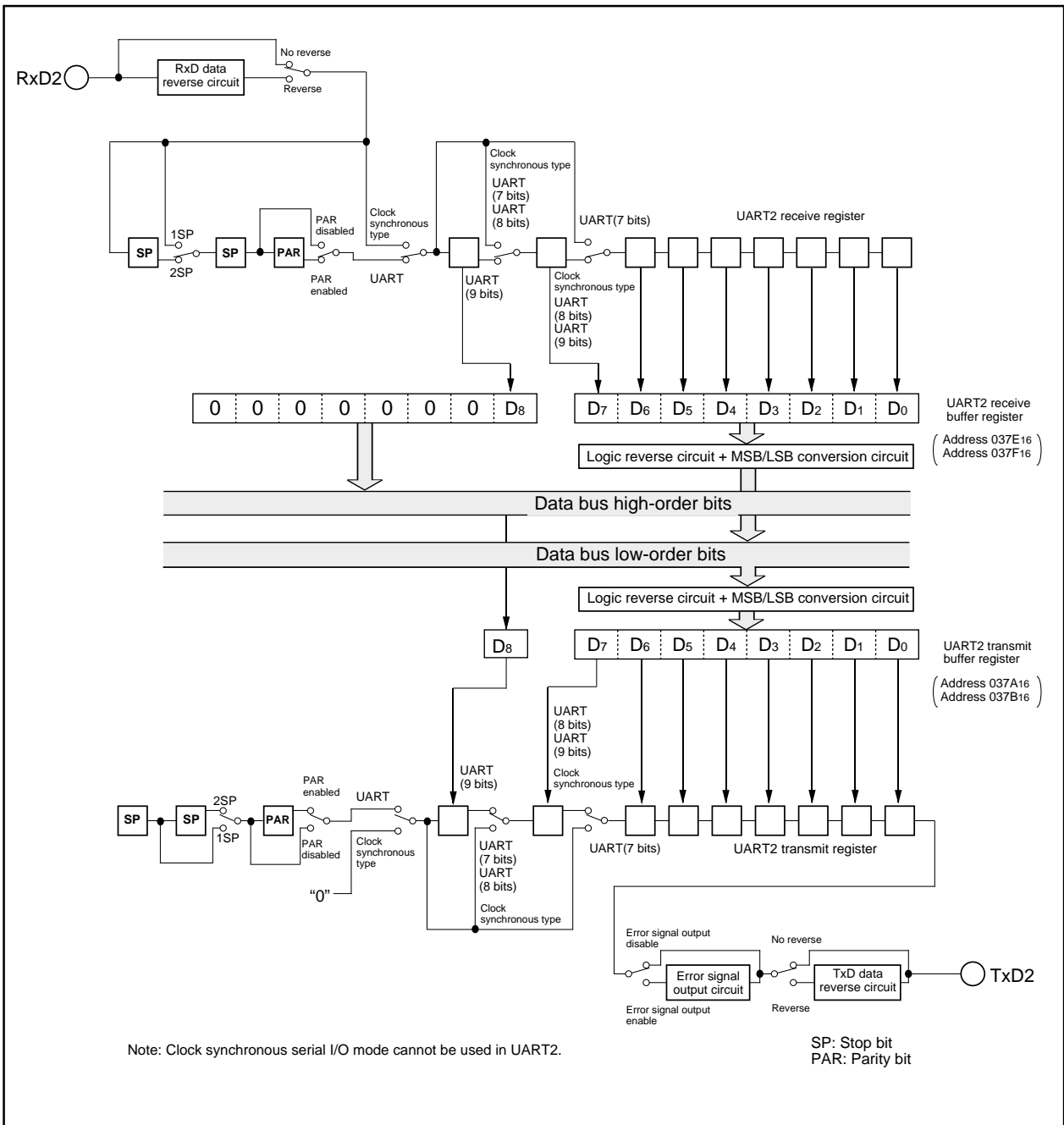


Figure 1.14.3. Block diagram of UART2 transmit/receive unit

Serial I/O

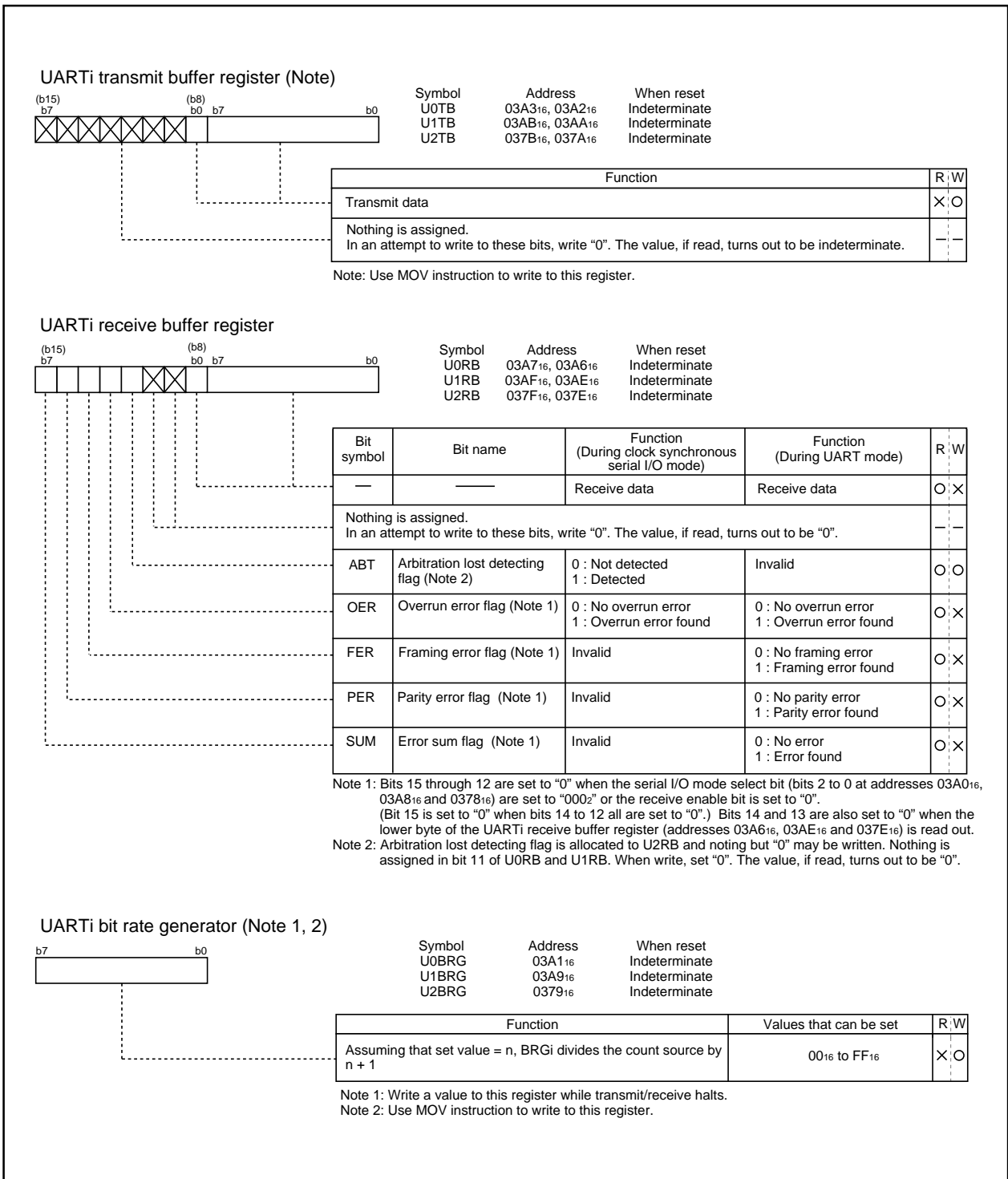


Figure 1.14.4. Serial I/O-related registers (1)

Serial I/O

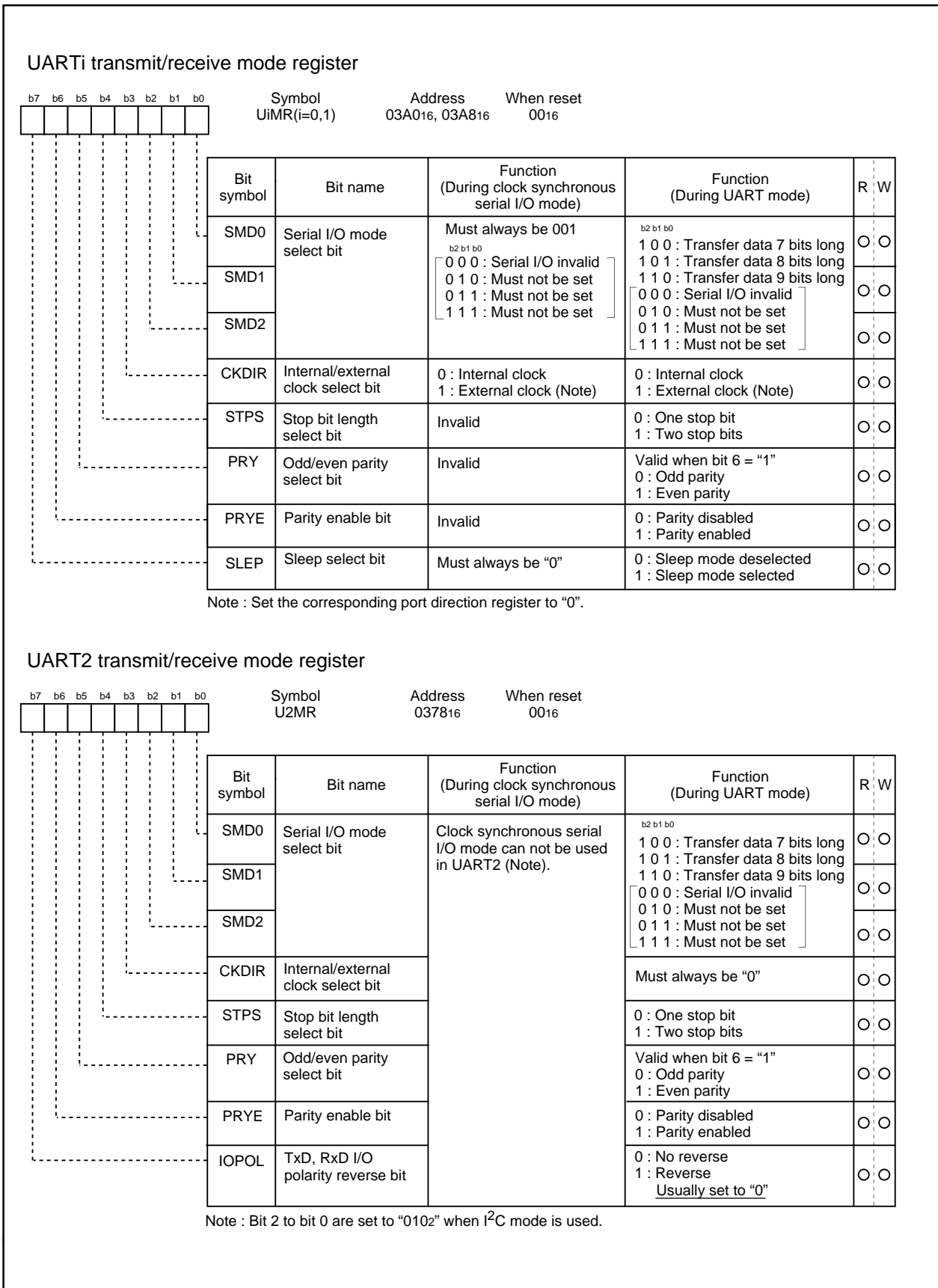


Figure 1.14.5. Serial I/O-related registers (2)

Serial I/O

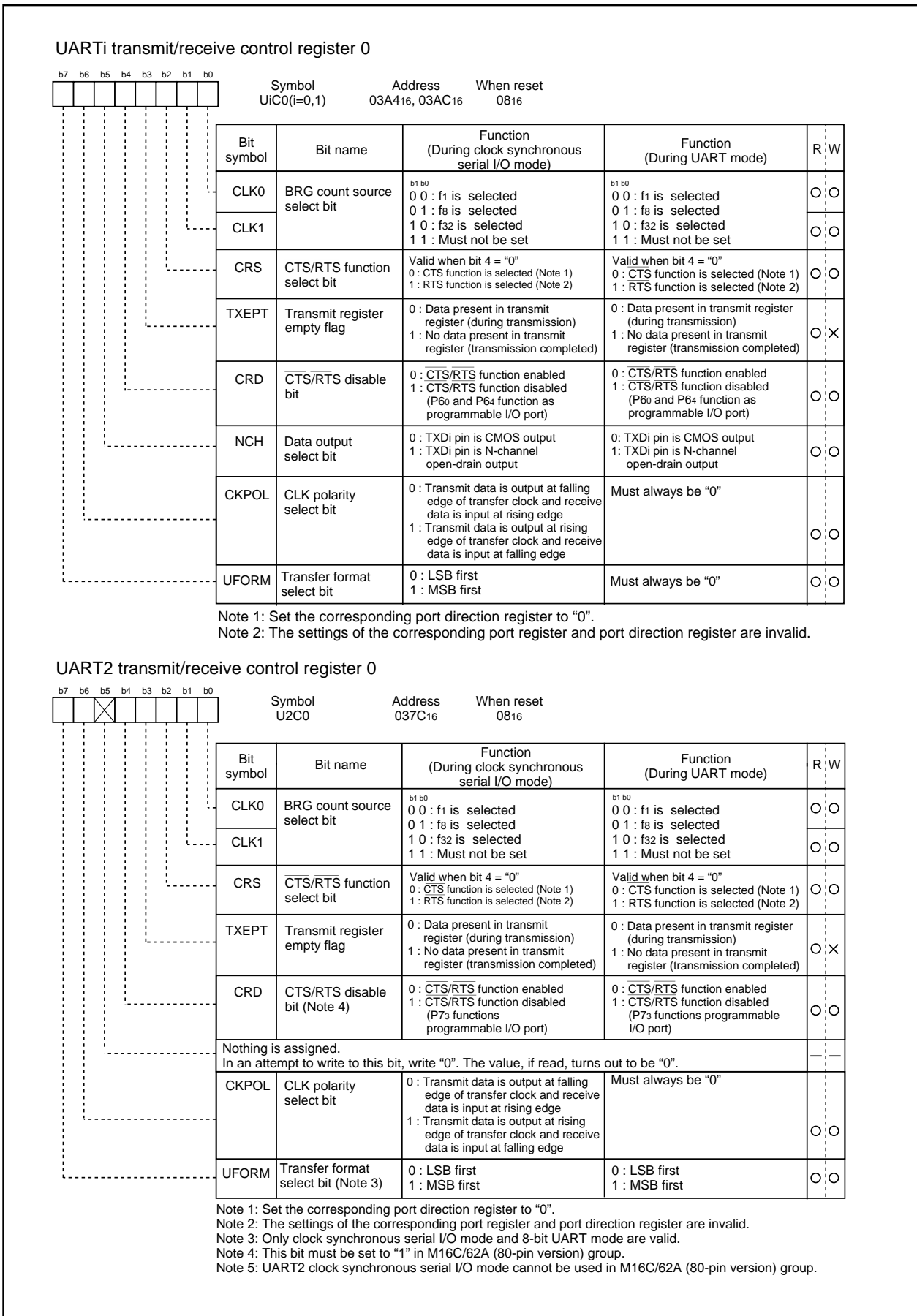


Figure 1.14.6. Serial I/O-related registers (3)

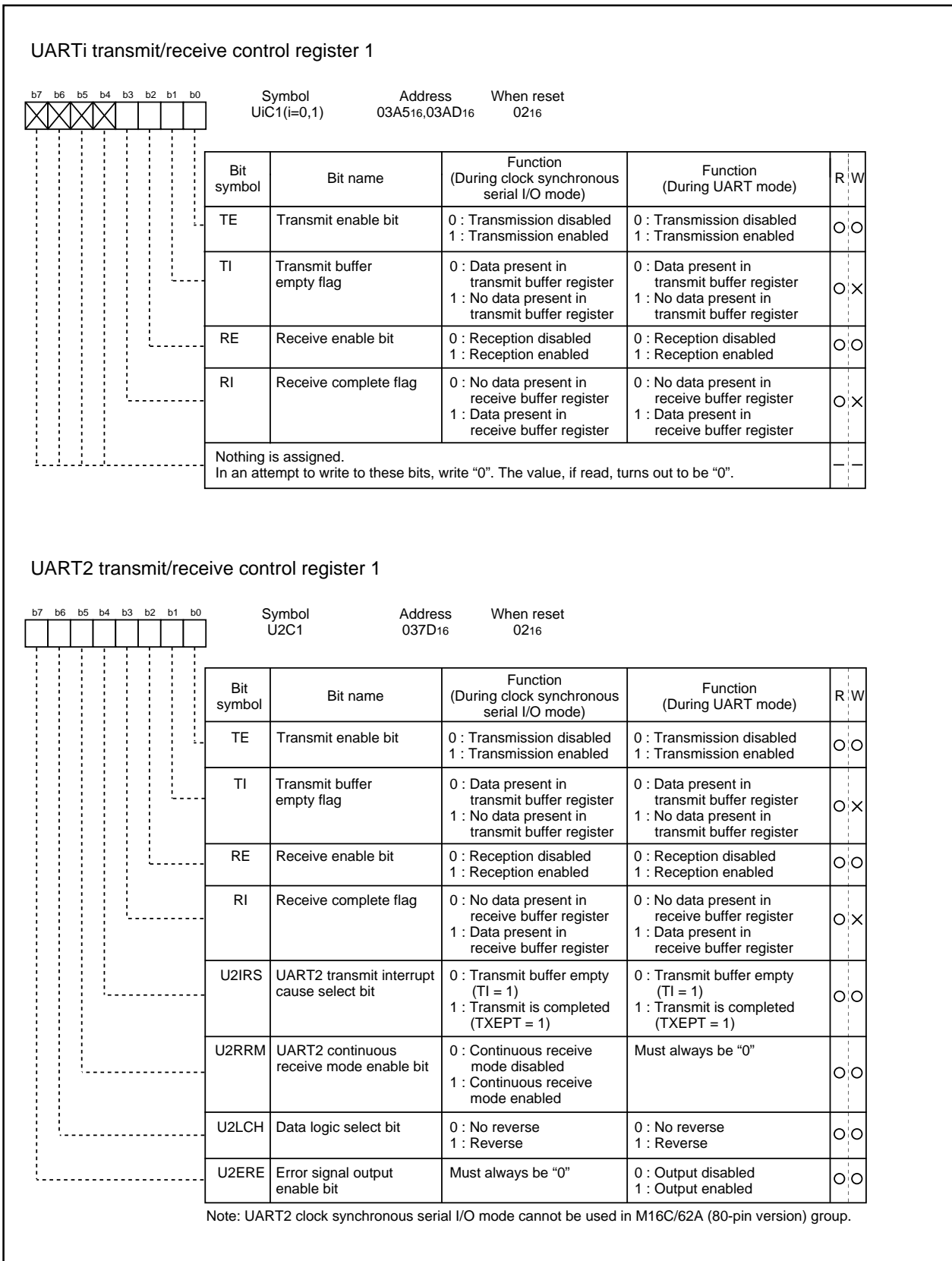


Figure 1.14.7. Serial I/O-related registers (4)

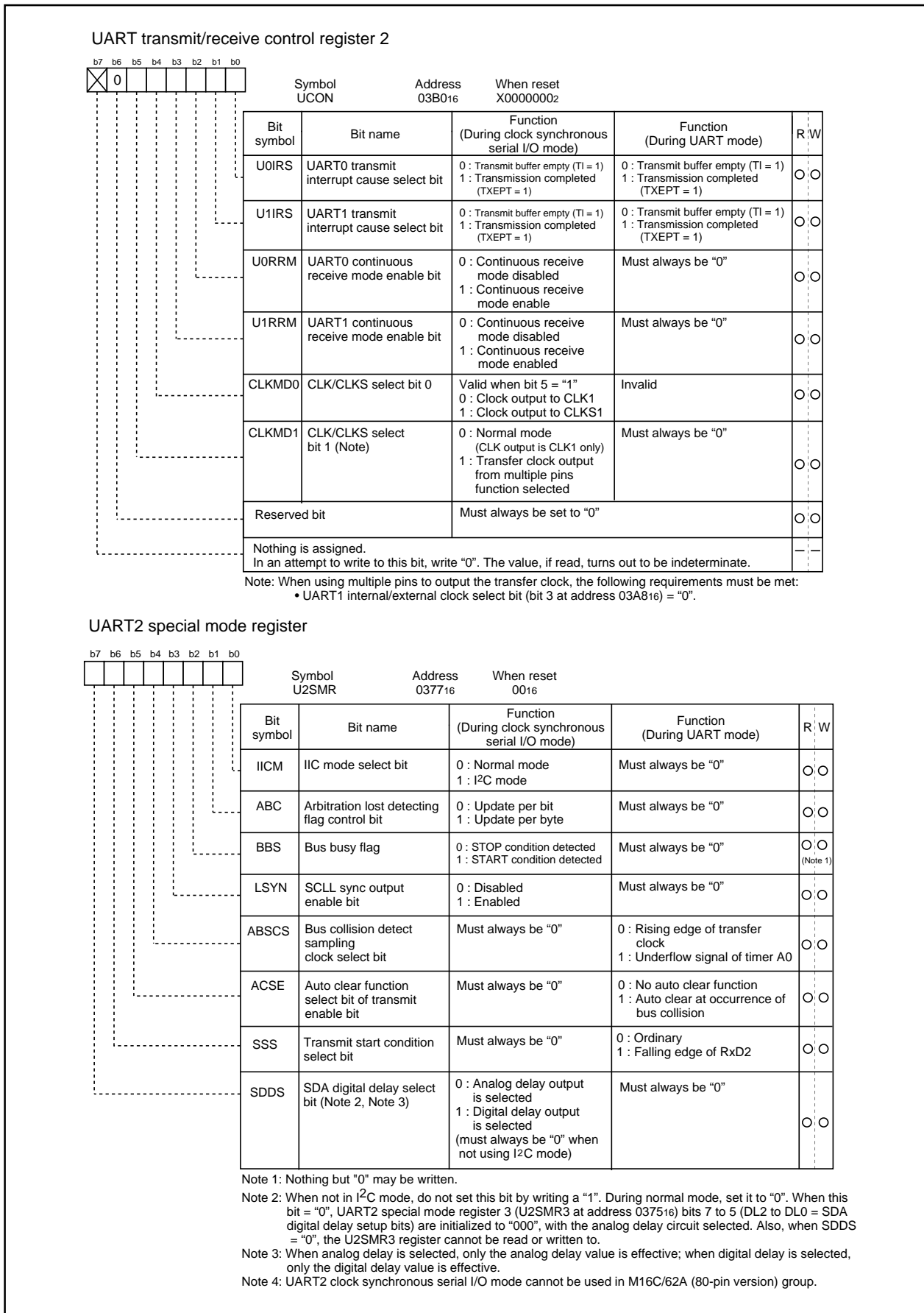


Figure 1.14.8. Serial I/O-related registers (5)

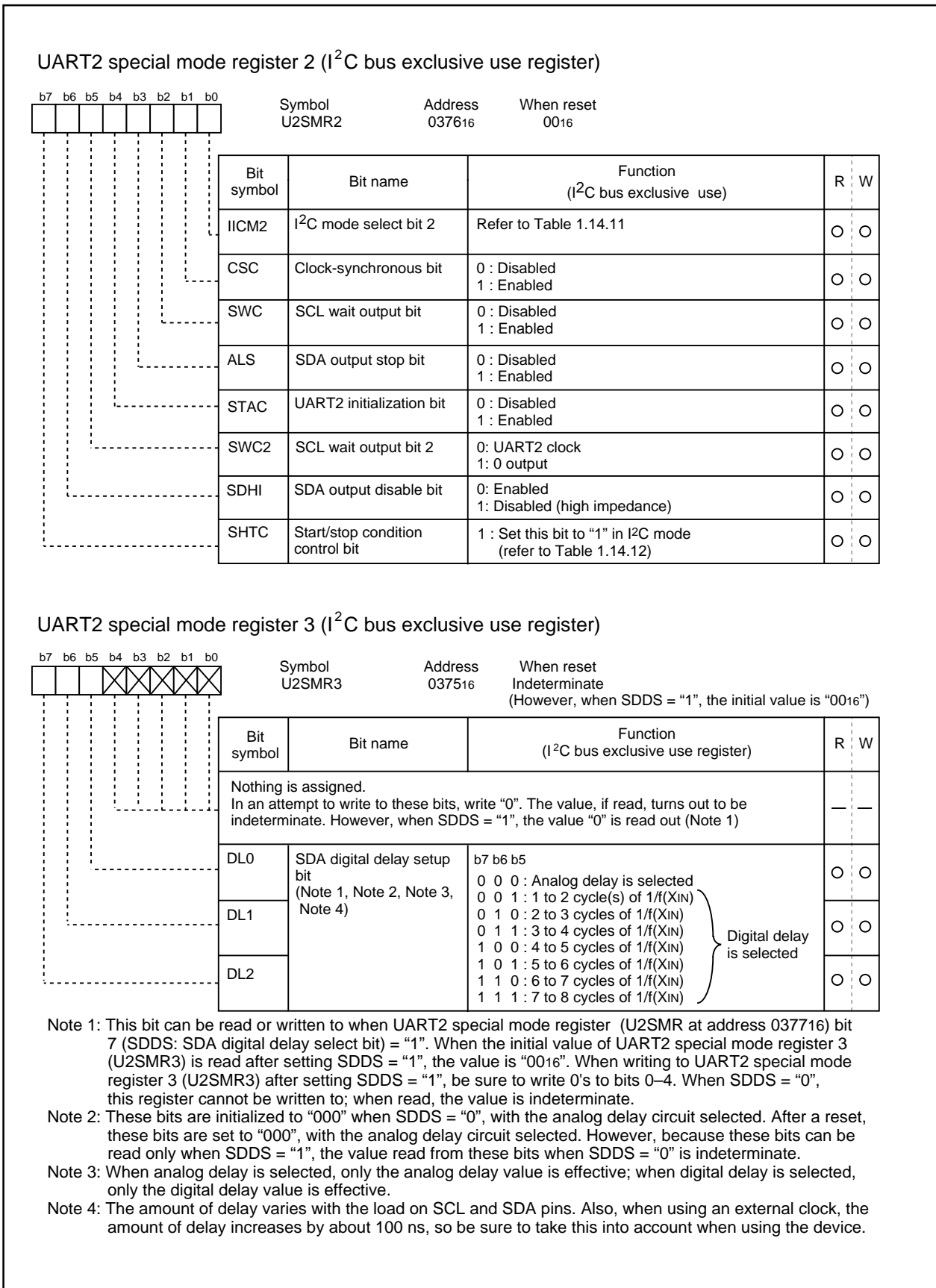


Figure 1.14.9. Serial I/O-related registers (6)



## 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. Tables 1.14.2 and 1.14.3 list the specifications of the clock synchronous serial I/O mode. Figure 1.14.10 shows the UARTi transmit/receive mode register. Clock synchronous serial I/O mode cannot be used in UART2.

**Table 1.14.2. Specifications of clock synchronous serial I/O mode (1)**

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 addresses 03A0<sub>16</sub>, 03A8<sub>16</sub> = "0") : <math>f_i / 2^{(n+1)}</math> (Note 1) <math>f_i = f_1, f_8, f_{32}</math></li> <li>When external clock is selected (bit 3 at addresses 03A0<sub>16</sub>, 03A8<sub>16</sub> = "1") : Input from CLKi pin</li> </ul>
Transmission/reception control	<ul style="list-style-type: none"> <li>CTS function, RTS function, CTS and RTS function invalid: selectable</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> <li>When <math>\overline{\text{CTS}}</math> function selected, <math>\overline{\text{CTS}}</math> input level = "L"</li> </ul> </li> <li>Furthermore, if external clock is selected, the following requirements must also be met:               <ul style="list-style-type: none"> <li>CLKi polarity select bit (bit 6 at addresses 03A4<sub>16</sub>, 03AC<sub>16</sub>) = "0" : CLKi input level = "H"</li> <li>CLKi polarity select bit (bit 6 at addresses 03A4<sub>16</sub>, 03AC<sub>16</sub>) = "1" : CLKi 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 addresses 03A5<sub>16</sub>, 03AD<sub>16</sub>) = "1"</li> <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> <li>Furthermore, if external clock is selected, the following requirements must also be met:               <ul style="list-style-type: none"> <li>CLKi polarity select bit (bit 6 at addresses 03A4<sub>16</sub>, 03AC<sub>16</sub>) = "0" : CLKi input level = "H"</li> <li>CLKi polarity select bit (bit 6 at addresses 03A4<sub>16</sub>, 03AC<sub>16</sub>) = "1" : CLKi 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 (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 bit (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 2)                This error occurs when the next data is ready before contents of UARTi receive buffer register are read out</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 UARTi receive buffer will have the next data written in. Note also that the UARTi receive interrupt request bit does not change.

Clock synchronous serial I/O mode

**Table 1.14.3. Specifications of clock synchronous serial I/O mode (2)**

Item	Specification
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 (UART1) (Note) UART1 transfer clock can be chosen by software to be output from one of the two pins set</li> </ul>

Note : Clock synchronous serial I/O mode cannot be used in UART2.

Clock synchronous serial I/O mode

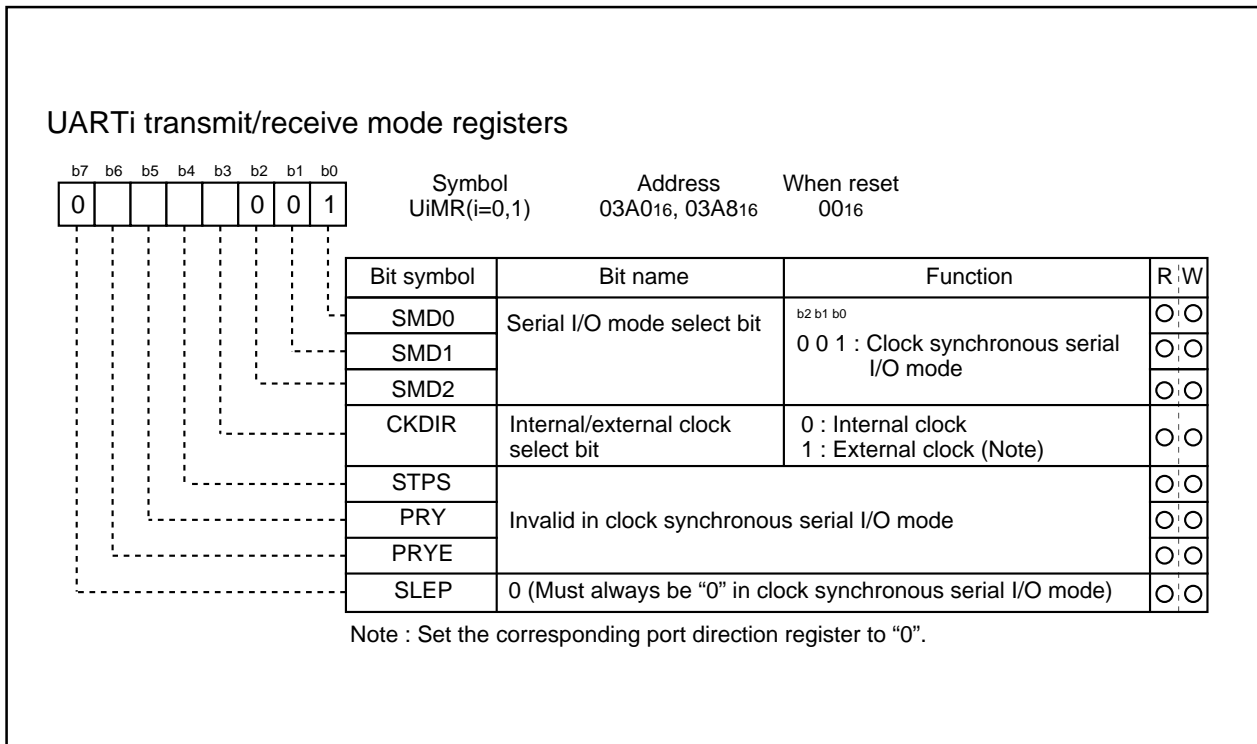


Figure 1.14.10. UARTi transmit/receive mode register in clock synchronous serial I/O mode

## Clock synchronous serial I/O mode

Table 1.14.4 lists the functions of the input/output pins during clock synchronous serial I/O mode. This table shows the pin functions when the transfer clock output from multiple pins is not selected. Note that for a period from when the UARTi operation mode is selected to when transfer starts, the TxDi pin outputs a "H". (If the N-channel open-drain is selected, this pin is in floating state.)

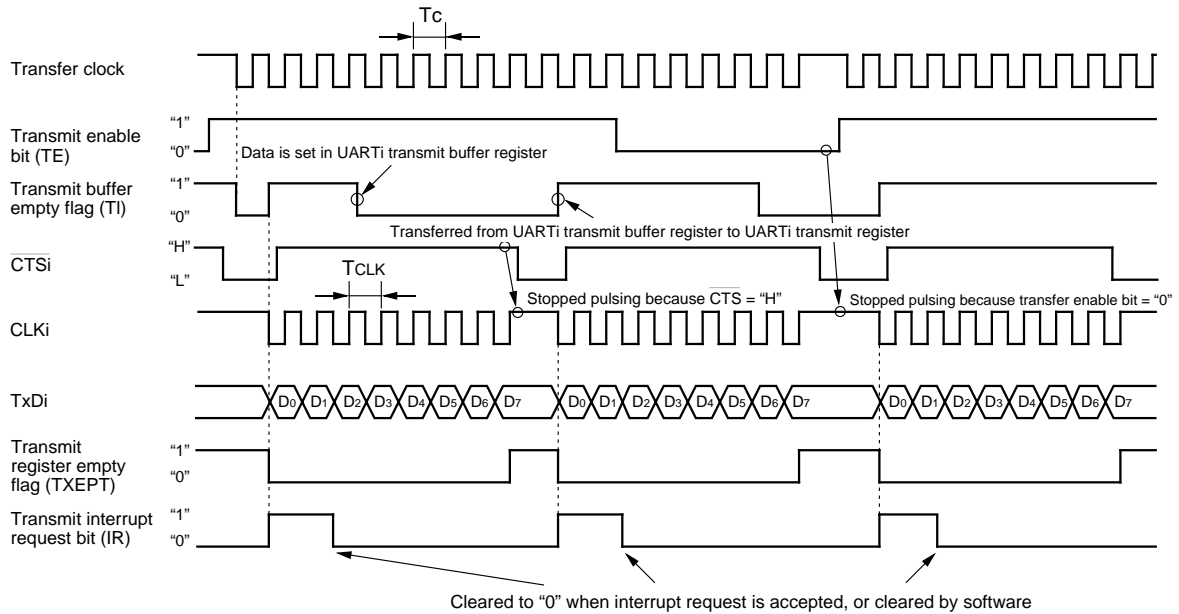
**Table 1.14.4. Input/output pin functions in clock synchronous serial I/O mode  
 (when transfer clock output from multiple pins is not selected)**

Pin name	Function	Method of selection
TxDi (P63, P67)	Serial data output	(Outputs dummy data when performing reception only)
RxDi (P62, P66)	Serial data input	Port P62 and P66 direction register (bits 2 and 6 at address 03EE16) = "0" (Can be used as an input port when performing transmission only)
CLKi (P61, P65)	Transfer clock output	Internal/external clock select bit (bit 3 at address 03A016, 03A816) = "0"
	Transfer clock input	Internal/external clock select bit (bit 3 at address 03A016, 03A816) = "1" Port P61 and P65 direction register (bits 1 and 5 at address 03EE16) = "0"
$\overline{\text{CTS}}/\overline{\text{RTS}}$ (P60, P64)	$\overline{\text{CTS}}$ input	$\overline{\text{CTS}}/\overline{\text{RTS}}$ disable bit (bit 4 at address 03A416, 03AC16) = "0" $\overline{\text{CTS}}/\overline{\text{RTS}}$ function select bit (bit 2 at address 03A416, 03AC16) = "0" Port P60 and P64 direction register (bits 0 and 4 at address 03EE16) = "0"
	$\overline{\text{RTS}}$ output	$\overline{\text{CTS}}/\overline{\text{RTS}}$ disable bit (bit 4 at address 03A416, 03AC16) = "0" $\overline{\text{CTS}}/\overline{\text{RTS}}$ function select bit (bit 2 at address 03A416, 03AC16) = "1"
	Programmable I/O port	$\overline{\text{CTS}}/\overline{\text{RTS}}$ disable bit (bit 4 at address 03A416, 03AC16) = "1"

Note: Clock synchronous serial I/O mode cannot be used in UART2.

**Clock synchronous serial I/O mode**

• Example of transmit timing (when internal clock is selected)



Shown in ( ) are bit symbols.

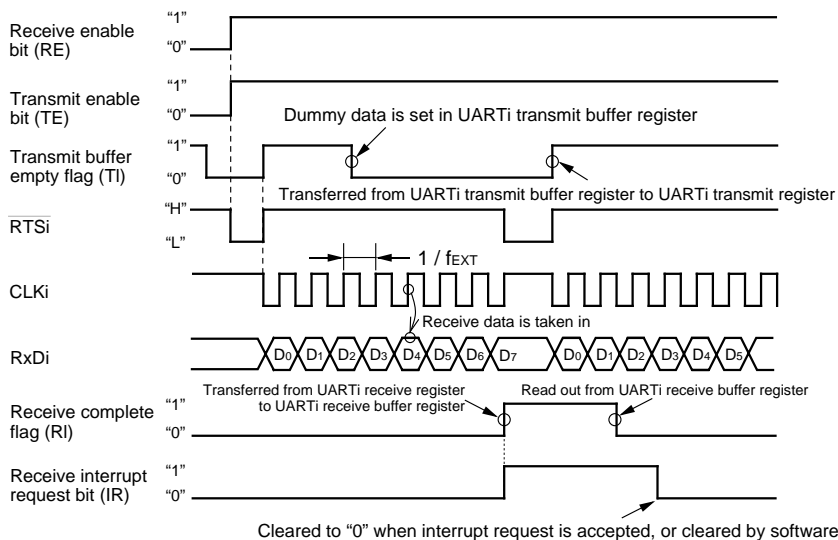
The above timing applies to the following settings:

- Internal clock is selected.
- CTS function is selected.
- CLK polarity select bit = "0".
- Transmit interrupt cause select bit = "0".

$$T_c = T_{CLK} = 2(n + 1) / f_i$$

$f_i$ : frequency of BRGi count source ( $f_1, f_8, f_{32}$ )  
 $n$ : value set to BRGi

• Example of receive timing (when external clock is selected)



Shown in ( ) are bit symbols.

The above timing applies to the following settings:

- External clock is selected.
- RTS function is selected.
- CLK polarity select bit = "0".

fEXT: frequency of external clock

Meet the following conditions are met when the CLK input before data reception = "H"

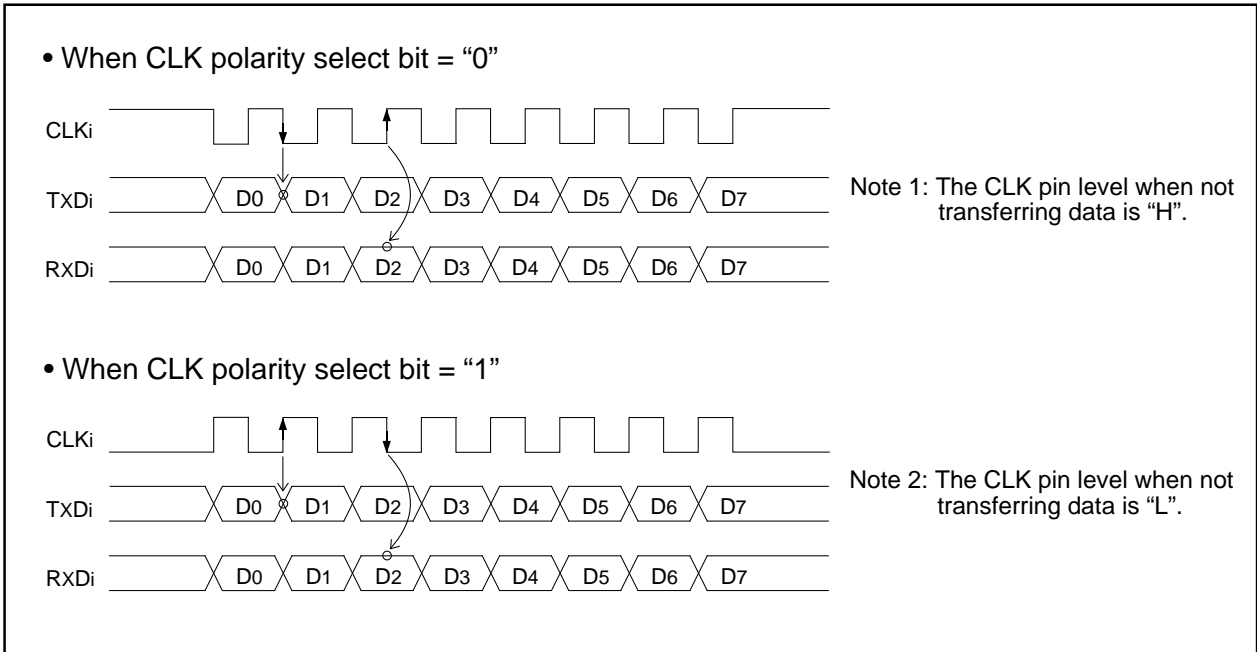
- Transmit enable bit → "1"
- Receive enable bit → "1"
- Dummy data write to UARTi transmit buffer register

**Figure 1.14.11. Typical transmit/receive timings in clock synchronous serial I/O mode**

**Clock synchronous serial I/O mode**

**(a) Polarity select function**

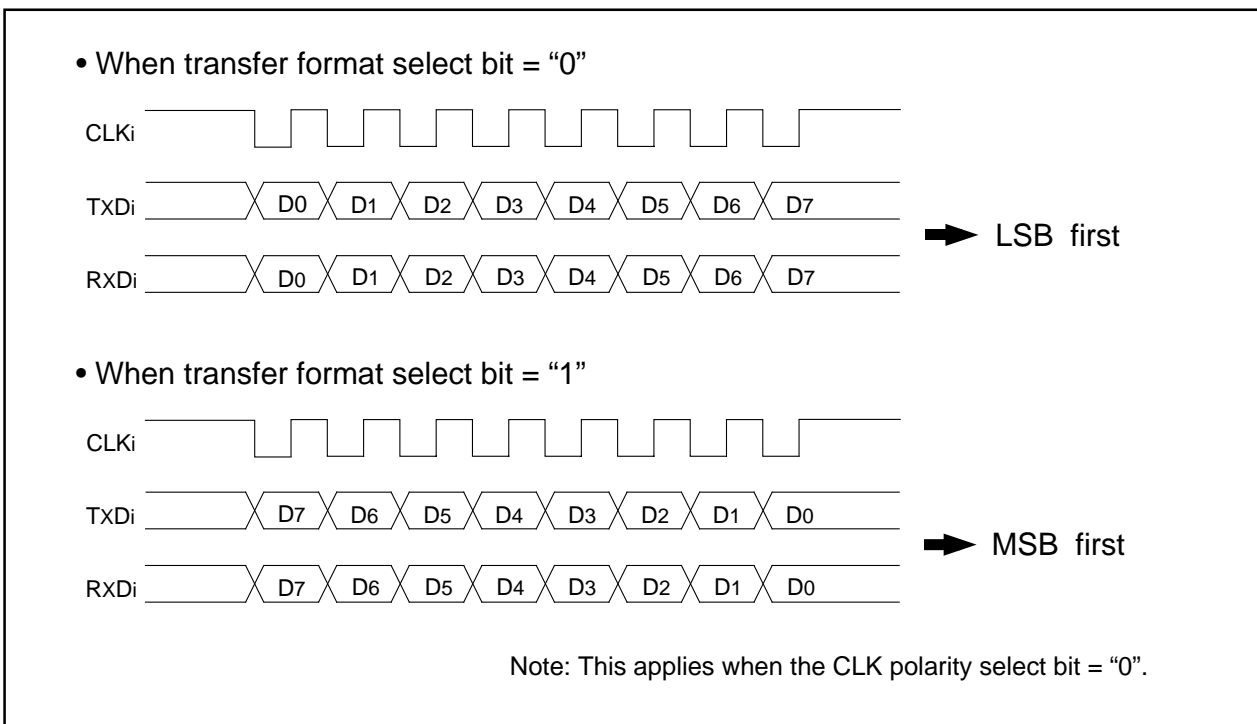
As shown in Figure 1.14.12, the CLK polarity select bit (bit 6 at addresses 03A4<sub>16</sub>, 03AC<sub>16</sub>) allows selection of the polarity of the transfer clock.



**Figure 1.14.12. Polarity of transfer clock**

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

As shown in Figure 1.14.13, when the transfer format select bit (bit 7 at addresses 03A4<sub>16</sub>, 03AC<sub>16</sub>) = "0", the transfer format is "LSB first"; when the bit = "1", the transfer format is "MSB first".



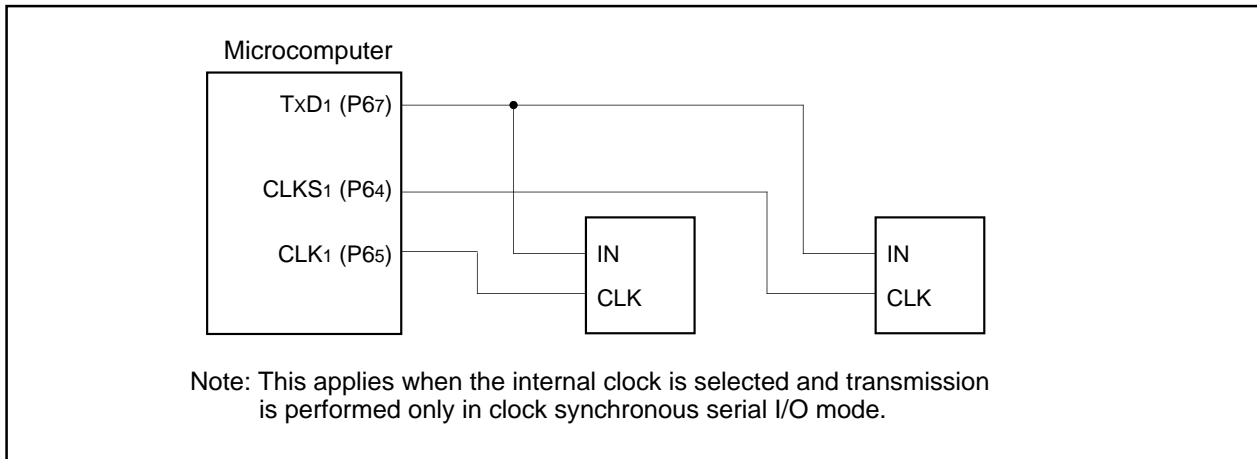
**Figure 1.14.13. Transfer format**

## Clock synchronous serial I/O mode

### (c) Transfer clock output from multiple pins function (UART1)

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 1.14.3.)

The multiple pins function is valid only when the internal clock is selected for UART1. Note that when this function is selected, UART1  $\overline{\text{CTS}}/\overline{\text{RTS}}$  function cannot be used.



**Figure 1.14.14. 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, bit 5 at address 037D16) 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. Tables 1.14.5 and 1.14.6 list the specifications of the UART mode. Figure 1.14.15 shows the UART<sub>i</sub> transmit/receive mode register.

**Table 1.14.5. Specifications of UART Mode (1)**

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>, 0378<sub>16</sub> = "0") : <math>f_i/16(n+1)</math> (Note 1) <math>f_i = f_1, f_8, f_{32}</math></li> <li>• When external clock is selected (bit 3 at addresses 03A0<sub>16</sub>, 03A8<sub>16</sub> = "1") : <math>f_{EXT}/16(n+1)</math>(Note 1,2,4)</li> </ul>
Transmission/reception control	<ul style="list-style-type: none"> <li>• CTS function, RTS function, CTS and RTS function invalid: selectable (Note 5)</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>, 037D<sub>16</sub>) = "1"</li> <li>- Transmit buffer empty flag (bit 1 at addresses 03A5<sub>16</sub>, 03AD<sub>16</sub>, 037D<sub>16</sub>) = "0"</li> <li>- When CTS function selected, CTS 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 addresses 03A5<sub>16</sub>, 03AD<sub>16</sub>, 037D<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>, bit4 at address 037D<sub>16</sub>) = "0": Interrupts requested when data transfer from UART<sub>i</sub> transfer buffer register to UART<sub>i</sub> transmit register is completed</li> <li>- Transmit interrupt cause select bits (bits 0, 1 at address 03B0<sub>16</sub>, bit4 at address 037D<sub>16</sub>) = "1": Interrupts requested when data transmission from UART<sub>i</sub> transfer register is completed</li> </ul> </li> <li>• When receiving               <ul style="list-style-type: none"> <li>- Interrupts requested when data transfer from UART<sub>i</sub> receive register to UART<sub>i</sub> 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 UART<sub>i</sub> 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>

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

Note 2: f<sub>EXT</sub> is input from the CLK<sub>i</sub> pin.

Note 3: If an overrun error occurs, the UART<sub>i</sub> receive buffer will have the next data written in. Note also that the UART<sub>i</sub> receive interrupt request bit does not change.

Note 4: Since CLK<sub>2</sub> does not have external port, external clock cannot be selected as UART<sub>2</sub> transfer clock.

Note 5: Set the CTS/RTS disable bit (bit 4 at address 037C<sub>16</sub>) to "1" because CTS<sub>2</sub>/RTS<sub>2</sub> does not have external port.



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

---

**Table 1.14.6. Specifications of UART Mode (2)**

Item	Specification
Select function	<ul style="list-style-type: none"><li data-bbox="507 342 1417 450">• Sleep mode selection (UART0, UART1) This mode is used to transfer data to and from one of multiple slave micro-computers</li><li data-bbox="507 461 1417 568">• Serial data logic switch (UART2) This function is reversing logic value of transferring data. Start bit, parity bit and stop bit are not reversed.</li><li data-bbox="507 580 1417 687">• TxD, RxD I/O polarity switch (UART2) This function is reversing TxD port output and RxD port input. All I/O data level is reversed.</li></ul>

Clock asynchronous serial I/O (UART) mode

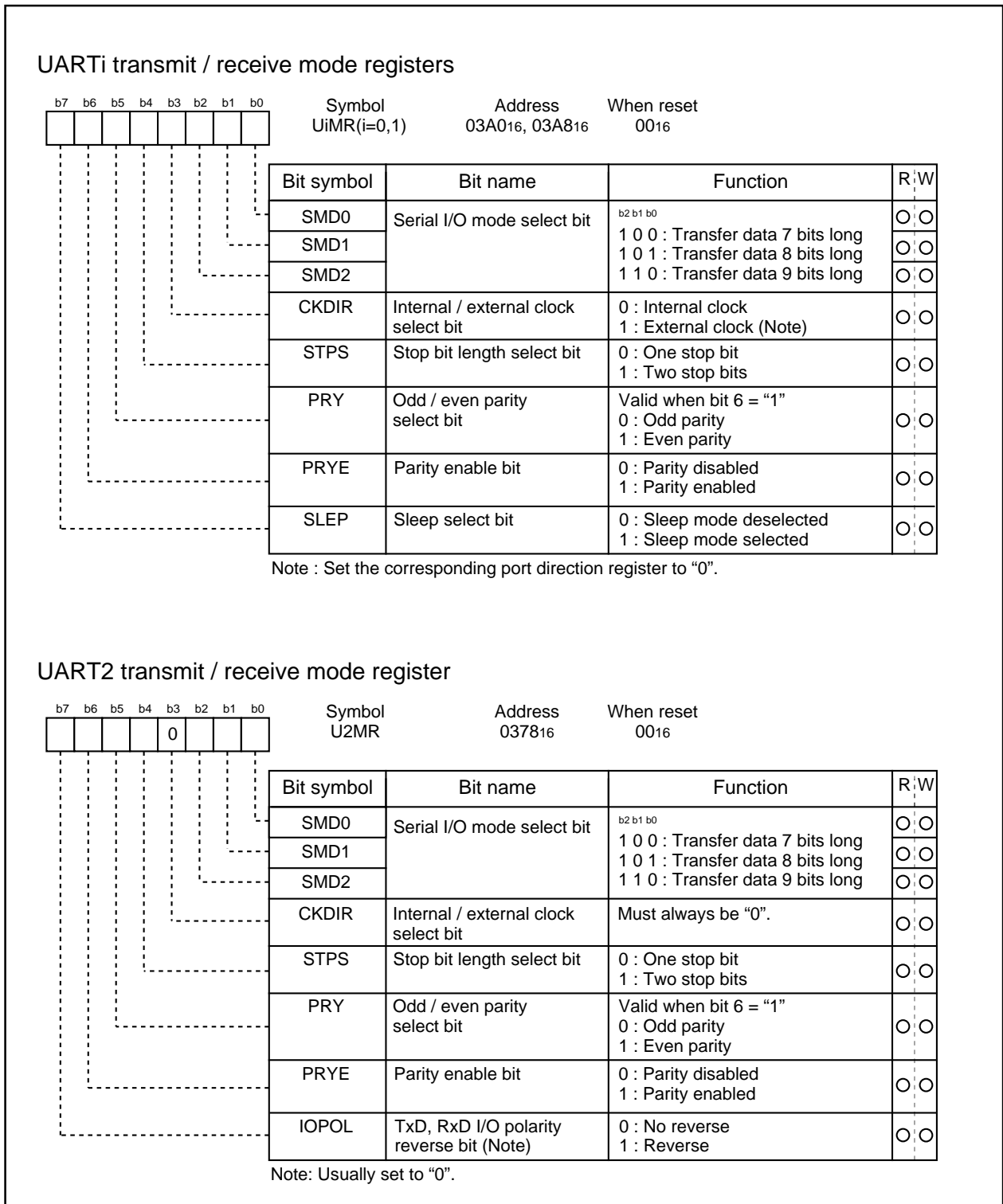


Figure 1.14.15. UART<sub>i</sub> transmit/receive mode register in UART mode

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

Table 1.14.7 lists the functions of the input/output pins during UART mode. Note that for a period from when the UARTi operation mode is selected to when transfer starts, the TxDi pin outputs an “H”. (If the N-channel open-drain is selected, this pin is in floating state.)

**Table 1.14.7. Input/output pin functions in UART mode**

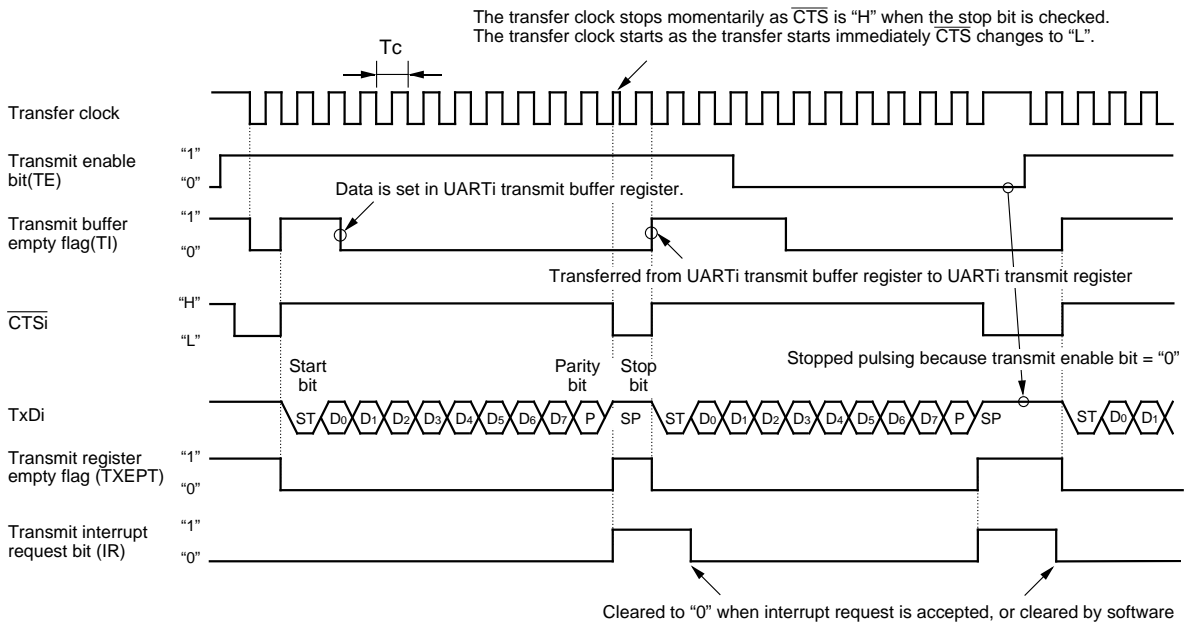
Pin name	Function	Method of selection
TxDi (P63, P67, P70)	Serial data output	
RxDi (P62, P66, P71)	Serial data input	Port P62, P66 and P71 direction register (bits 2 and 6 at address 03EE16, bit 1 at address 03EF16) = “0” (Can be used as an input port when performing transmission only)
CLKi (P61, P65)	Programmable I/O port	Internal/external clock select bit (bit 3 at address 03A016, 03A816) = “0”
	Transfer clock input	Internal/external clock select bit (bit 3 at address 03A016, 03A816) = “1” Port P61 and P65 direction register (bits 1 and 5 at address 03EE16) = “0”
$\overline{\text{CTS}}/\overline{\text{RTS}}$ (P60, P64)	$\overline{\text{CTS}}$ input	$\overline{\text{CTS}}/\overline{\text{RTS}}$ disable bit (bit 4 at address 03A416, 03AC16) = “0” $\overline{\text{CTS}}/\overline{\text{RTS}}$ function select bit (bit 2 at address 03A416, 03AC16) = “0” Port P60 and P64 direction register (bits 0 and 4 at address 03EE16) = “0”
	RTS output	$\overline{\text{CTS}}/\overline{\text{RTS}}$ disable bit (bit 4 at address 03A416, 03AC16) = “0” $\overline{\text{CTS}}/\overline{\text{RTS}}$ function select bit (bit 2 at address 03A416, 03AC16) = “1”
	Programmable I/O port	$\overline{\text{CTS}}/\overline{\text{RTS}}$ disable bit (bit 4 at address 03A416, 03AC16) = “1”

Note 1: Since CLK2(P72) does not have external port, use internal as UART2 transfer clock.

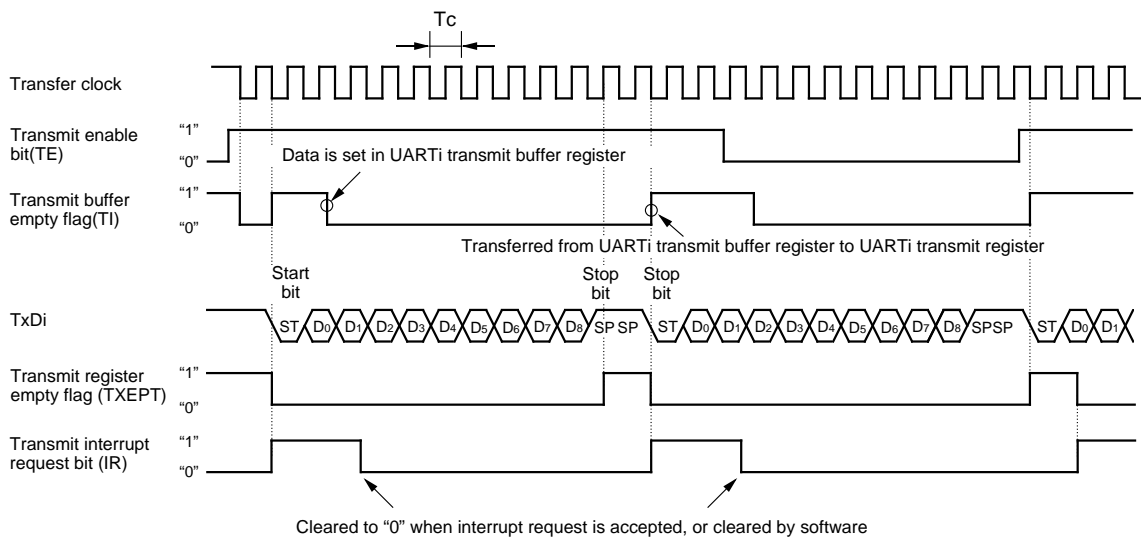
Note 2: Set the  $\overline{\text{CTS}}/\overline{\text{RTS}}$  disable bit (bit 4 at address 037C16) to “1” because  $\overline{\text{CTS}}/\overline{\text{RTS}}_2$ (P73) does not have external port.

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

• Example of transmit timing when transfer data is 8 bits long (parity enabled, one stop bit)



• Example of transmit timing when transfer data is 9 bits long (parity disabled, two stop bits)



**Figure 1.14.16. Typical transmit timings in UART mode(UART0, UART1)**

Clock asynchronous serial I/O (UART) mode

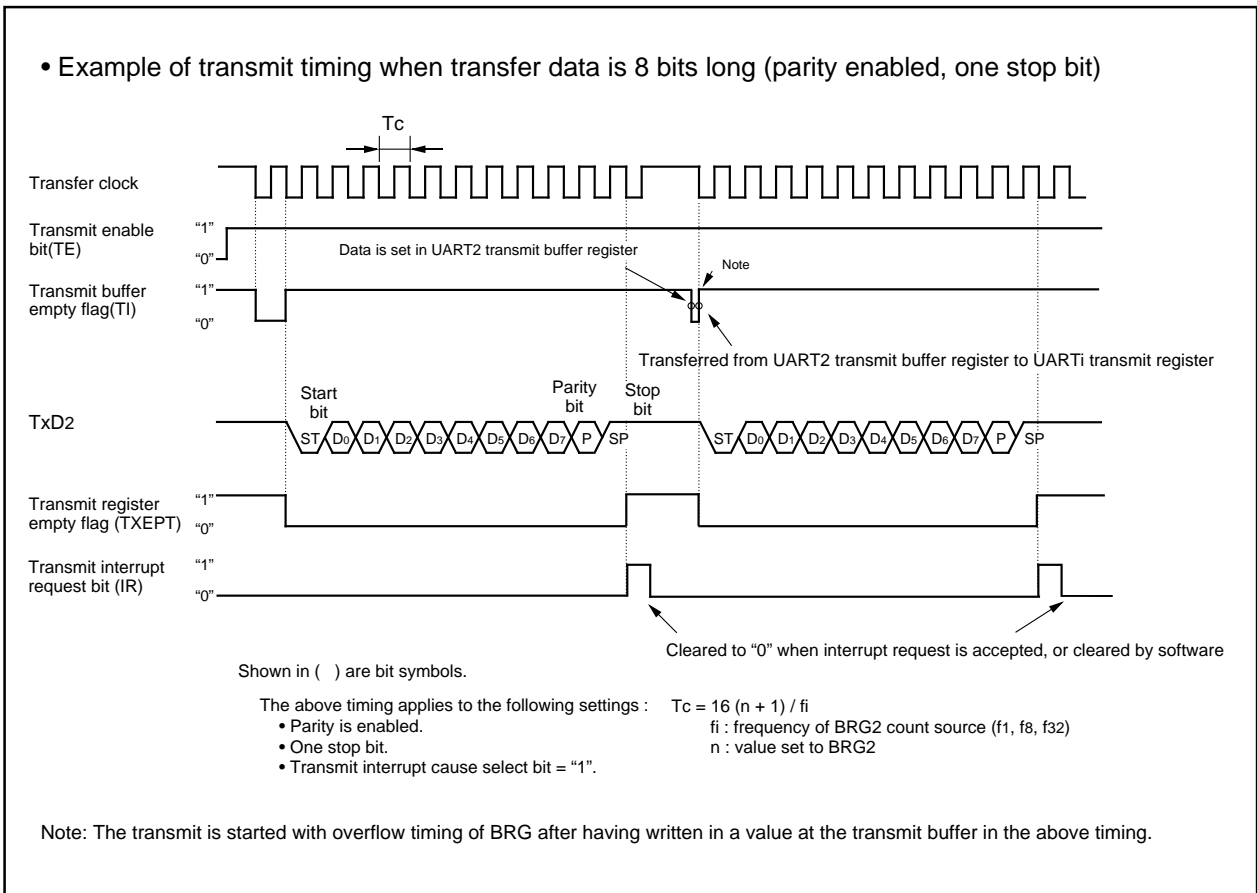
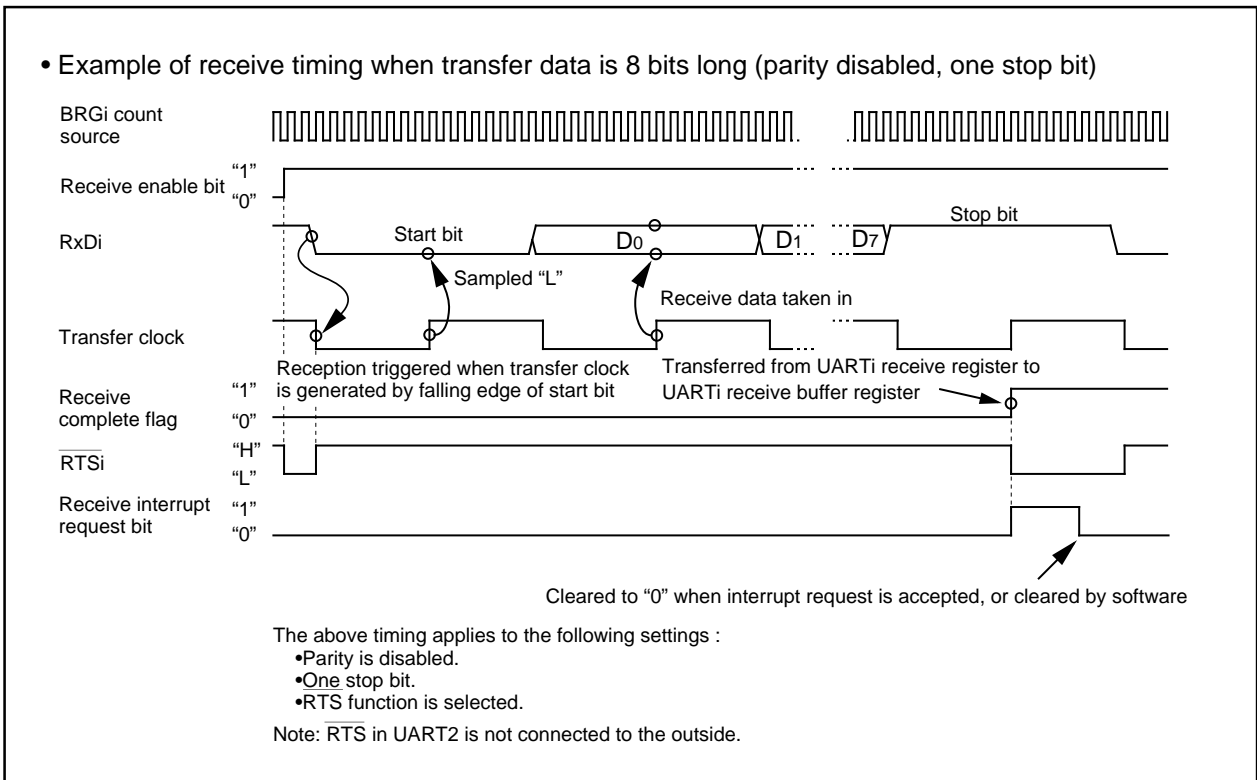


Figure 1.14.17. Typical transmit timings in UART mode(UART2)

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



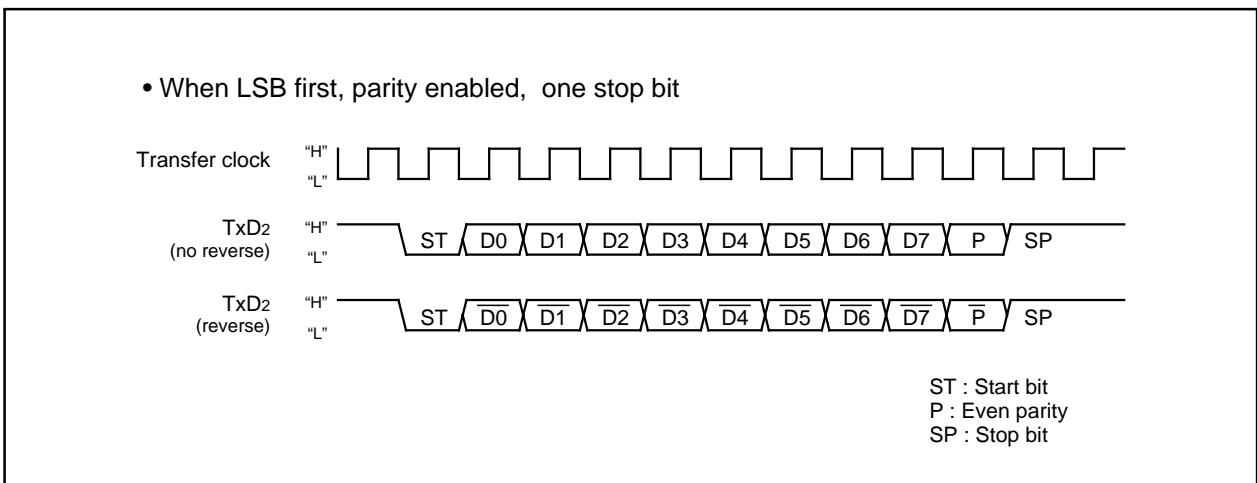
**Figure 1.14.18. Typical receive timing in UART mode**

**(a) Sleep mode (UART0, UART1)**

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".

**(b) Function for switching serial data logic (UART2)**

When the data logic select bit (bit 6 of address 037D16) is assigned "1", data is inverted in writing to the transmission buffer register or reading the reception buffer register. Figure 1.14.19 shows the example of timing for switching serial data logic.



**Figure 1.14.19. Timing for switching serial data logic**

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

### (c) TxD, RxD I/O polarity reverse function (UART2)

This function is to reverse TxD pin output and RxD pin input. The level of any data to be input or output (including the start bit, stop bit(s), and parity bit) is reversed. Set this function to "0" (not to reverse) for usual use.

### (d) Bus collision detection function (UART2)

This function is to sample the output level of the TxD pin and the input level of the RxD pin at the rising edge of the transfer clock; if their values are different, then an interrupt request occurs. Figure 1.14.20 shows the example of detection timing of a bus collision (in UART mode).

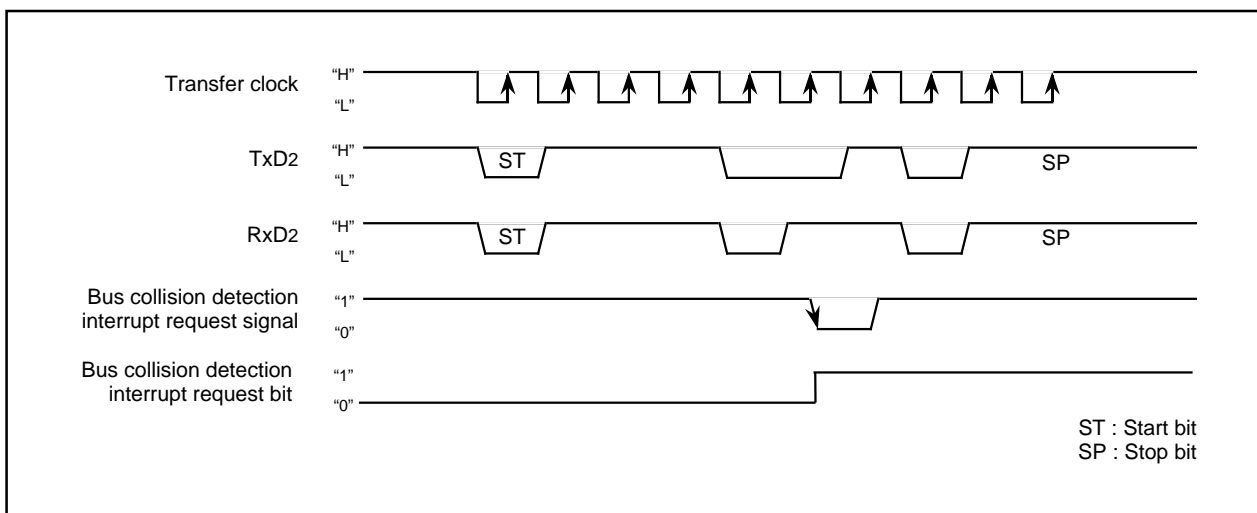


Figure 1.14.20. Detection timing of a bus collision (in UART mode)

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

### (3) Clock-asynchronous serial I/O mode (used for the SIM interface)

The SIM interface is used for connecting the microcomputer with a memory card or the like; adding some extra settings in UART2 clock-asynchronous serial I/O mode allows the user to effect this function. Table 1.14.8 shows the specifications of clock-asynchronous serial I/O mode (used for the SIM interface).

**Table 1.14.8. Specifications of clock-asynchronous serial I/O mode (used for the SIM interface)**

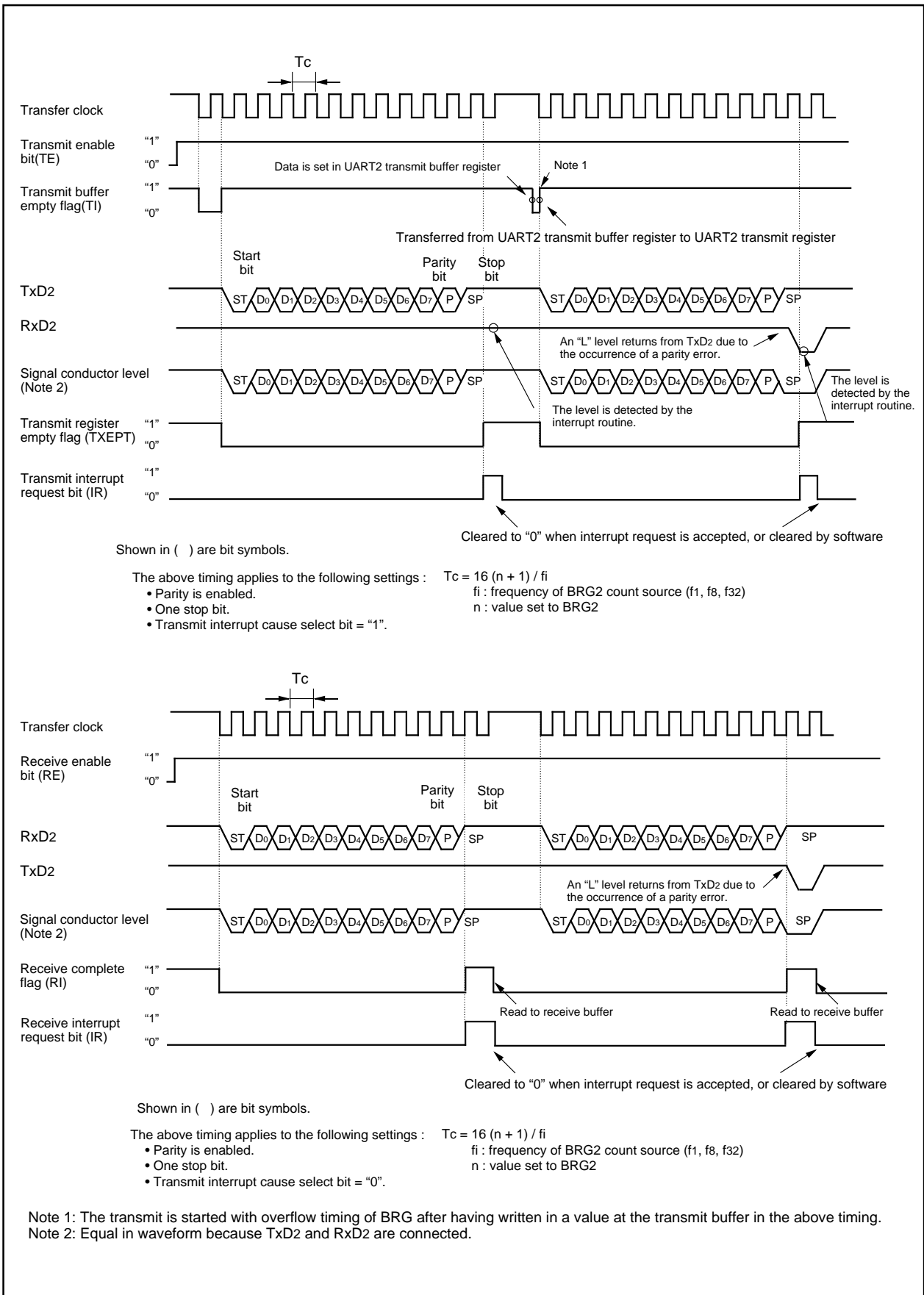
Item	Specification
Transfer data format	<ul style="list-style-type: none"> <li>• Transfer data 8-bit UART mode (bit 2 through bit 0 of address 0378<sub>16</sub> = "1012")</li> <li>• One stop bit (bit 4 of address 0378<sub>16</sub> = "0")</li> <li>• With the direct format chosen               <ul style="list-style-type: none"> <li>Set parity to "even" (bit 5 and bit 6 of address 0378<sub>16</sub> = "1" and "1" respectively)</li> <li>Set data logic to "direct" (bit 6 of address 037D<sub>16</sub> = "0").</li> <li>Set transfer format to LSB (bit 7 of address 037C<sub>16</sub> = "0").</li> </ul> </li> <li>• With the inverse format chosen               <ul style="list-style-type: none"> <li>Set parity to "odd" (bit 5 and bit 6 of address 0378<sub>16</sub> = "0" and "1" respectively)</li> <li>Set data logic to "inverse" (bit 6 of address 037D<sub>16</sub> = "1")</li> <li>Set transfer format to MSB (bit 7 of address 037C<sub>16</sub> = "1")</li> </ul> </li> </ul>
Transfer clock	<ul style="list-style-type: none"> <li>• With the internal clock chosen (bit 3 of address 0378<sub>16</sub> = "0") : <math>f_i / 16 (n + 1)</math>                (Note 1) : <math>f_i = f_1, f_8, f_{32}</math></li> </ul>
Transmission / reception control	<ul style="list-style-type: none"> <li>• Disable the CTS and RTS function (bit 4 of address 037C<sub>16</sub> = "1")</li> </ul>
Other settings	<ul style="list-style-type: none"> <li>• The sleep mode select function is not available for UART2</li> <li>• Set transmission interrupt factor to "transmission completed" (bit 4 of address 037D<sub>16</sub> = "1")</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 of address 037D<sub>16</sub>) = "1"</li> <li>- Transmit buffer empty flag (bit 1 of address 037D<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>- Reception enable bit (bit 2 of address 037D<sub>16</sub>) = "1"</li> <li>- Detection of a start bit</li> </ul> </li> </ul>
Interrupt request generation timing	<ul style="list-style-type: none"> <li>• When transmitting               <ul style="list-style-type: none"> <li>When data transmission from the UART2 transfer register is completed (bit 4 of address 037D<sub>16</sub> = "1")</li> </ul> </li> <li>• When receiving               <ul style="list-style-type: none"> <li>When data transfer from the UART2 receive register to the UART2 receive buffer register is completed</li> </ul> </li> </ul>
Error detection	<ul style="list-style-type: none"> <li>• Overrun error (see the specifications of clock-asynchronous serial I/O) (Note 2)</li> <li>• Framing error (see the specifications of clock-asynchronous serial I/O)</li> <li>• Parity error (see the specifications of clock-asynchronous serial I/O)               <ul style="list-style-type: none"> <li>- On the reception side, an "L" level is output from the TxD<sub>2</sub> pin by use of the parity error signal output function (bit 7 of address 037D<sub>16</sub> = "1") when a parity error is detected</li> <li>- On the transmission side, a parity error is detected by the level of input to the RxD<sub>2</sub> pin when a transmission interrupt occurs</li> </ul> </li> <li>• The error sum flag (see the specifications of clock-asynchronous serial I/O)</li> </ul>

Note 1: 'n' denotes the value 00<sub>16</sub> to FF<sub>16</sub> that is set to the UART2 bit rate generator.

Note 2: If an overrun error occurs, the UART2 receive buffer will have the next data written in. Note also that the UART2 receive interrupt request bit does not change.



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



**Figure 1.14.21. Typical transmit/receive timing in UART mode (used for the SIM interface)**

Clock asynchronous serial I/O (UART) mode

**(a) Function for outputting a parity error signal**

During reception, with the error signal output enable bit (bit 7 of address 037D16) assigned "1", you can output an "L" level from the TxD2 pin when a parity error is detected. And during transmission, comparing with the case in which the error signal output enable bit (bit 7 of address 037D16) is assigned "0", the transmission completion interrupt occurs in the half cycle later of the transfer clock. Therefore parity error signals can be detected by a transmission completion interrupt program. Figure 1.14.22 shows the output timing of the parity error signal.

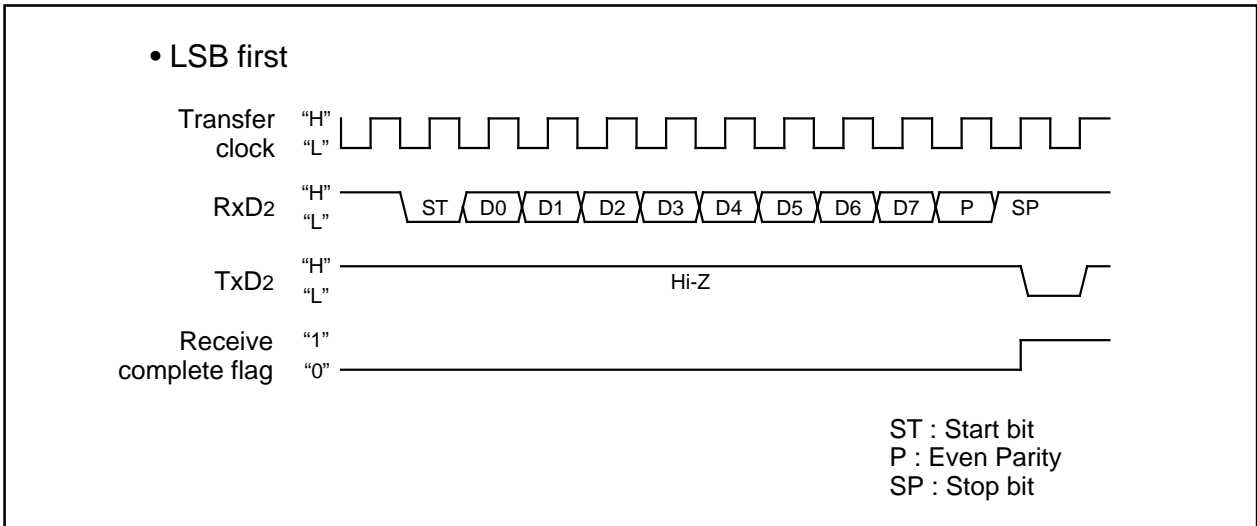


Figure 1.14.22. Output timing of the parity error signal

**(b) Direct format/inverse format**

Connecting the SIM card allows you to switch between direct format and inverse format. If you choose the direct format, D0 data is output from TxD2. If you choose the inverse format, D7 data is inverted and output from TxD2.

Figure 1.14.23 shows the SIM interface format.

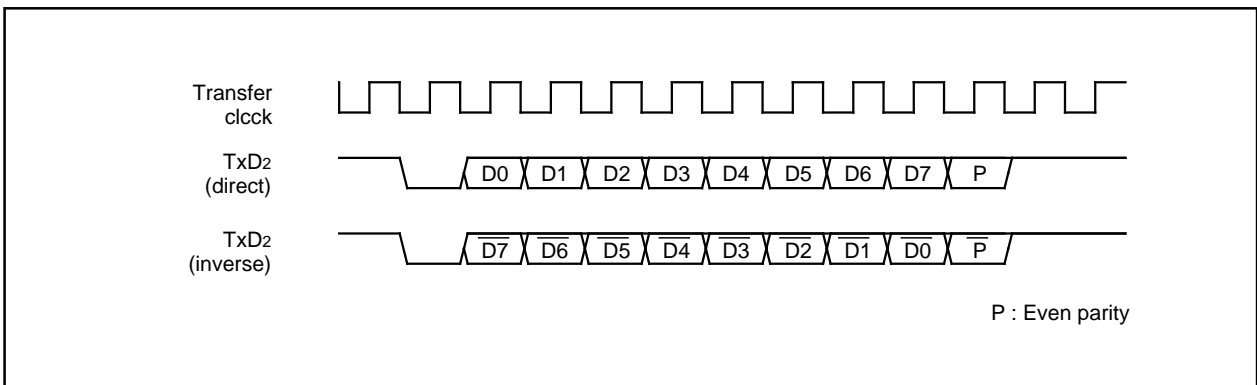


Figure 1.14.23. SIM interface format

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

---

Figure 1.14.24 shows the example of connecting the SIM interface. Connect TxD2 and RxD2 and apply pull-up.

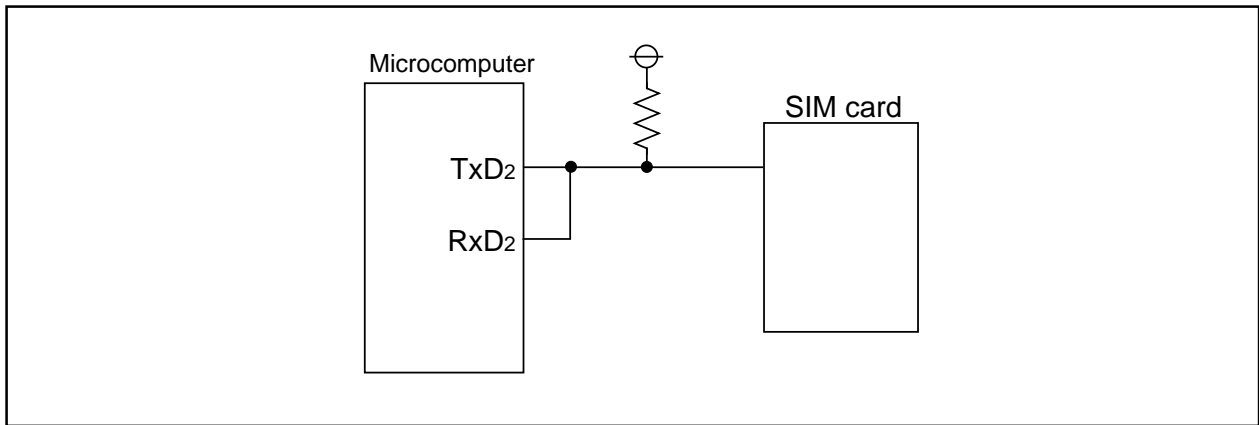


Figure 1.14.24. Connecting the SIM interface

## UART2 Special Mode Register

### UART2 Special Mode Register

The UART2 special mode register (address 037716) is used to control UART2 in various ways.

Figure 1.14.25 shows the UART2 special mode register.

Bit 0 of the UART2 special mode register (037716) is used as the I<sup>2</sup>C mode select bit.

Setting "1" in the I<sup>2</sup>C mode select bit (bit 0) goes the circuit to achieve the I<sup>2</sup>C bus (simplified I<sup>2</sup>C bus) interface effective.

Table 1.14.9 shows the relation between the I<sup>2</sup>C mode select bit and respective control workings.

Since this function uses clock-synchronous serial I/O mode, set this bit to "0" in UART mode.

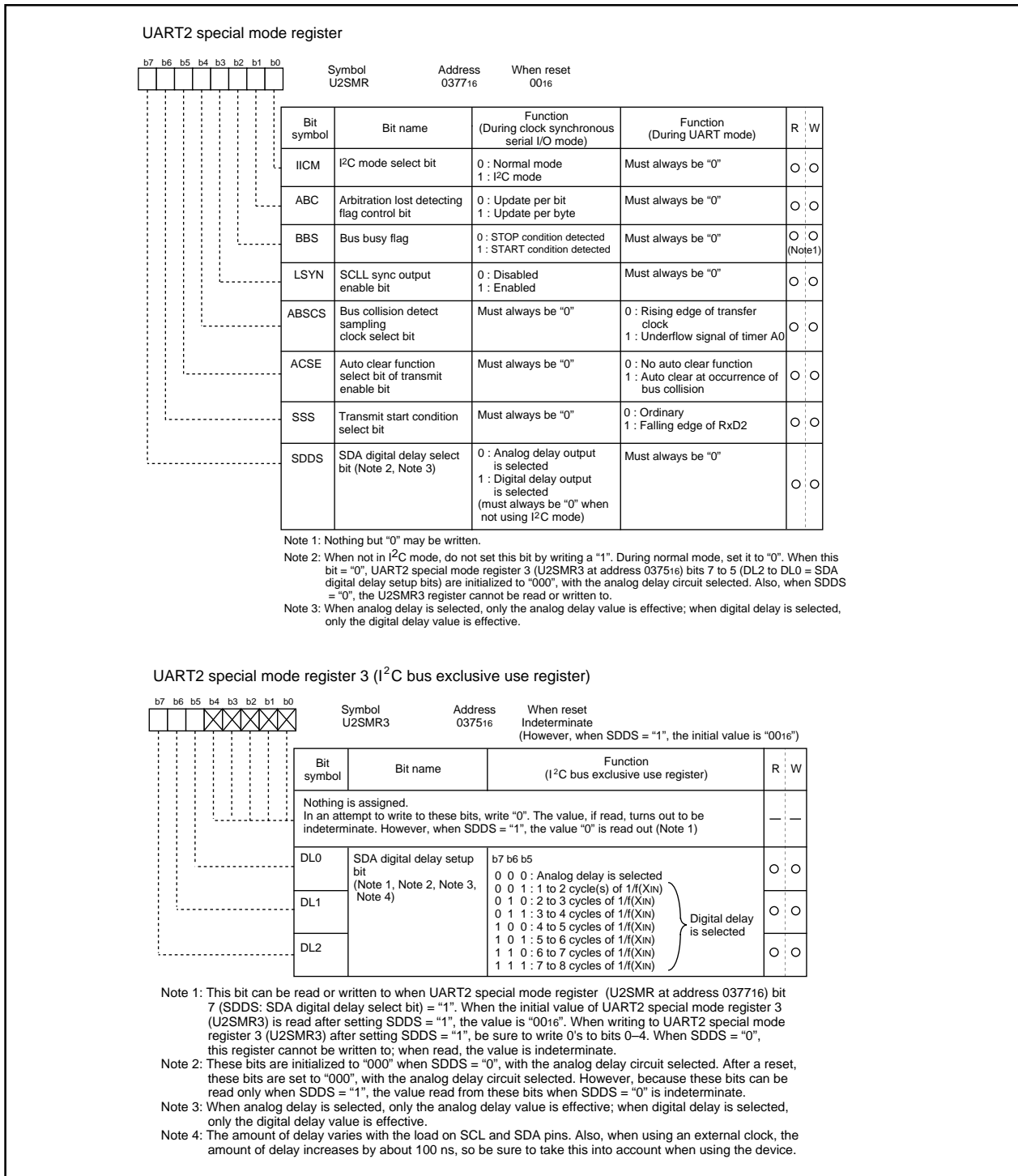
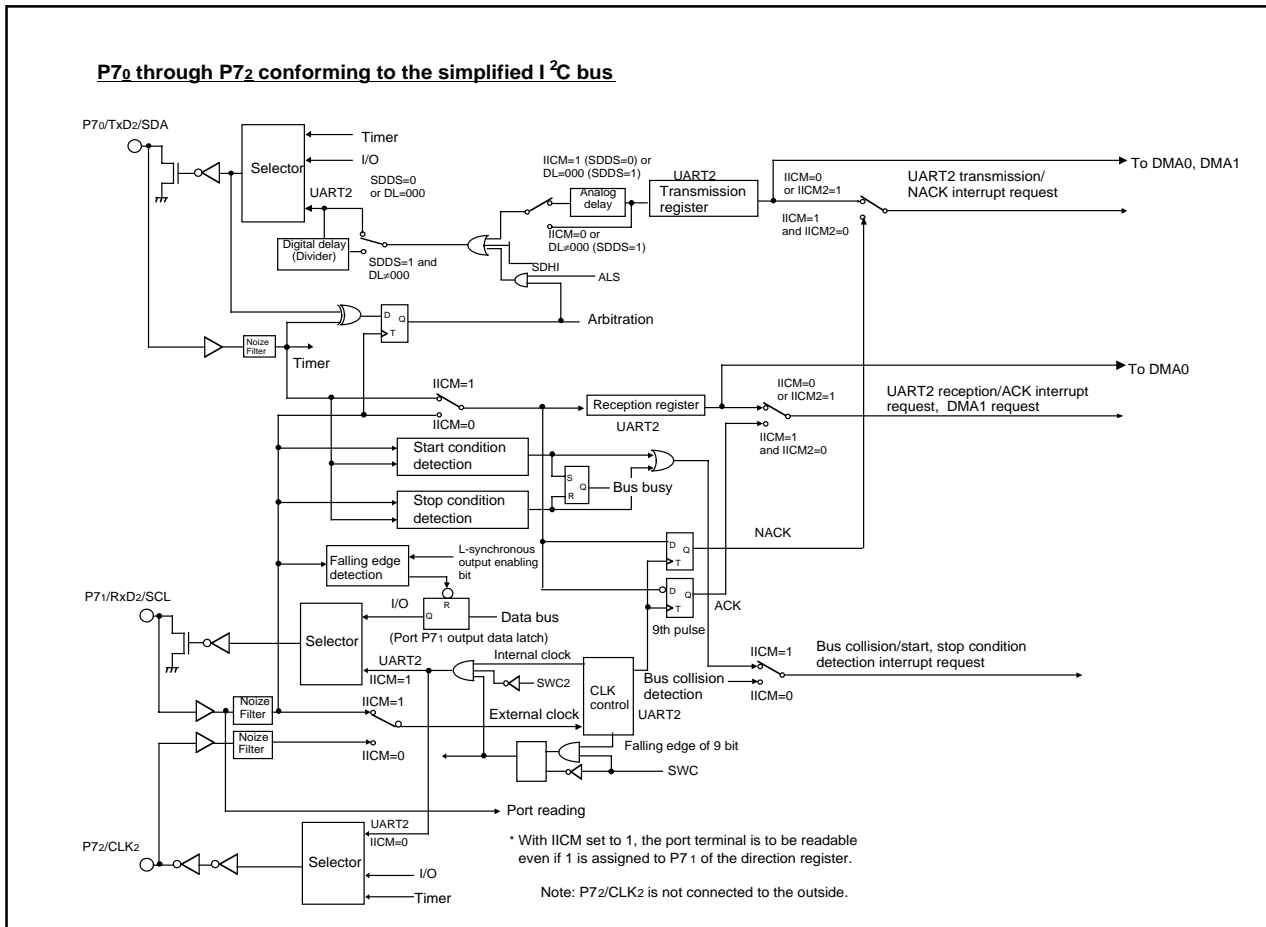


Figure 1.14.25. UART2 special mode register

## UART2 Special Mode Register



**Figure 1.14.26. Functional block diagram for I<sup>2</sup>C mode**

**Table 1.14.9. Features in I<sup>2</sup>C mode**

	Function	Normal mode	I <sup>2</sup> C mode (Note 1)
1	Factor of interrupt number 10 (Note 2)	Bus collision detection	Start condition detection or stop condition detection
2	Factor of interrupt number 15 (Note 2)	UART2 transmission	No acknowledgment detection (NACK)
3	Factor of interrupt number 16 (Note 2)	UART2 reception	Acknowledgment detection (ACK)
4	UART2 transmission output delay	Not delayed	Delayed (digital or analog delay can be selected)
5	P7 <sub>0</sub> at the time when UART2 is in use	TxD <sub>2</sub> (output)	SDA (input/output) (Note 3)
6	P7 <sub>1</sub> at the time when UART2 is in use	RxD <sub>2</sub> (input)	SCL (input/output)
7	DMA1 factor at the time when 1 1 0 1 is assigned to the DMA request factor selection bits	UART2 reception	Acknowledgment detection (ACK)
8	Noise filter width	15ns	50ns
9	Reading P7 <sub>1</sub>	Reading the terminal when 0 is assigned to the direction register	Reading the terminal regardless of the value of the direction register
10	Initial value of UART2 output	H level (when 0 is assigned to the CLK polarity select bit)	The value set in latch P7 <sub>0</sub> when the port is selected

Note 1: Make the settings given below when I<sup>2</sup>C mode is in use.

Set "0 1 0 2" in bits 2, 1, and 0 of the UART2 transmission/reception mode register.

Disable the RTS/CTS function. Choose the MSB First function.

Note 2: Follow the steps given below to switch from a factor to another.

1. Disable the interrupt of the corresponding number.
2. Switch from a factor to another.
3. Reset the interrupt request flag of the corresponding number.
4. Set an interrupt level of the corresponding number.

Note 3: Set an initial value of SDA transmission output when serial I/O is invalid.

## UART2 Special Mode Register

Figure 1.14.26 shows the functional block diagram for I<sup>2</sup>C mode. Setting “1” in the I<sup>2</sup>C mode select bit (IICM) causes ports P7<sub>0</sub>, P7<sub>1</sub>, and P7<sub>2</sub> to work as data transmission-reception terminal SDA, clock input-output terminal SCL, and port P7<sub>2</sub> respectively. A delay circuit is added to the SDA transmission output, so the SDA output changes after SCL fully goes to “L”. The SDA digital delay select bit (bit 7 at address 0377<sub>16</sub>) can be used to select between analog delay and digital delay. When digital delay is selected, the amount of delay can be selected in the range of 2 cycles to 8 cycles of f<sub>1</sub> using UART2 special mode register 3 (at address 0375<sub>16</sub>). Delay circuit select conditions are shown in Table 1.14.10.

**Table 1.14.10. Delay circuit select conditions**

	Register value			Contents
	IICM	SDDS	DL	
Digital delay is selected	1	1	001 to 111	When digital delay is selected, no analog delay is added. Only digital delay is effective.
Analog delay is selected	1	1	000	When DL is set to “000”, analog delay is selected no matter what value is set in SDDS.
		0	(000)	When SDDS is set to “0”, DL is initialized, so that DL = “000”.
No delay	0	0	(000)	When IICM = “0”, no delay circuit is selected. When IICM = “0”, however, always make sure SDDS = “0”.

An attempt to read Port P7<sub>1</sub> (SCL) results in getting the terminal’s level regardless of the content of the port direction register. The initial value of SDA transmission output in this mode goes to the value set in port P7<sub>0</sub>. The interrupt factors of the bus collision detection interrupt, UART2 transmission interrupt, and of UART2 reception interrupt turn to the start/stop condition detection interrupt, acknowledgment non-detection interrupt, and acknowledgment detection interrupt respectively.

The start condition detection interrupt refers to the interrupt that occurs when the falling edge of the SDA terminal (P7<sub>0</sub>) is detected with the SCL terminal (P7<sub>1</sub>) staying “H”. The stop condition detection interrupt refers to the interrupt that occurs when the rising edge of the SDA terminal (P7<sub>0</sub>) is detected with the SCL terminal (P7<sub>1</sub>) staying “H”. The bus busy flag (bit 2 of the UART2 special mode register) is set to “1” by the start condition detection, and set to “0” by the stop condition detection.

The acknowledgment non-detection interrupt refers to the interrupt that occurs when the SDA terminal level is detected still staying “H” at the rising edge of the 9th transmission clock. The acknowledgment detection interrupt refers to the interrupt that occurs when SDA terminal’s level is detected already went to “L” at the 9th transmission clock. Also, assigning 1 1 0 1 (UART2 reception) to the DMA1 request factor select bits provides the means to start up the DMA transfer by the effect of acknowledgment detection. Bit 1 of the UART2 special mode register (0377<sub>16</sub>) is used as the arbitration lost detecting flag control bit. Arbitration means the act of detecting the nonconformity between transmission data and SDA terminal data at the timing of the SCL rising edge. This detecting flag is located at bit 11 of the UART2 reception buffer register, and “1” is set in this flag when nonconformity is detected. Use the arbitration lost detecting flag control bit to choose which way to use to update the flag, bit by bit or byte by byte. When setting this bit to “1” and updated the flag byte by byte if nonconformity is detected, the arbitration lost detecting flag is set to “1” at the falling edge of the 9th transmission clock.

If update the flag byte by byte, must judge and clear (“0”) the arbitration lost detecting flag after completing the first byte acknowledge detect and before starting the next one byte transmission.

Bit 3 of the UART2 special mode register is used as SCL- and L-synchronous output enable bit. Setting this bit to “1” goes the P7<sub>1</sub> data register to “0” in synchronization with the SCL terminal level going to “L”.

## UART2 Special Mode Register

Some other functions added are explained here. Figure 1.14.27 shows their workings.

Bit 4 of the UART2 special mode register is used as the bus collision detect sampling clock select bit. The bus collision detect interrupt occurs when the RxD2 level and TxD2 level do not match, but the nonconformity is detected in synchronization with the rising edge of the transfer clock signal if the bit is set to "0". If this bit is set to "1", the nonconformity is detected at the timing of the overflow of timer A0 rather than at the rising edge of the transfer clock.

Bit 5 of the UART2 special mode register is used as the auto clear function select bit of transmit enable bit. Setting this bit to "1" automatically resets the transmit enable bit to "0" when "1" is set in the bus collision detect interrupt request bit (nonconformity).

Bit 6 of the UART2 special mode register is used as the transmit start condition select bit. Setting this bit to "1" starts the TxD transmission in synchronization with the falling edge of the RxD terminal.

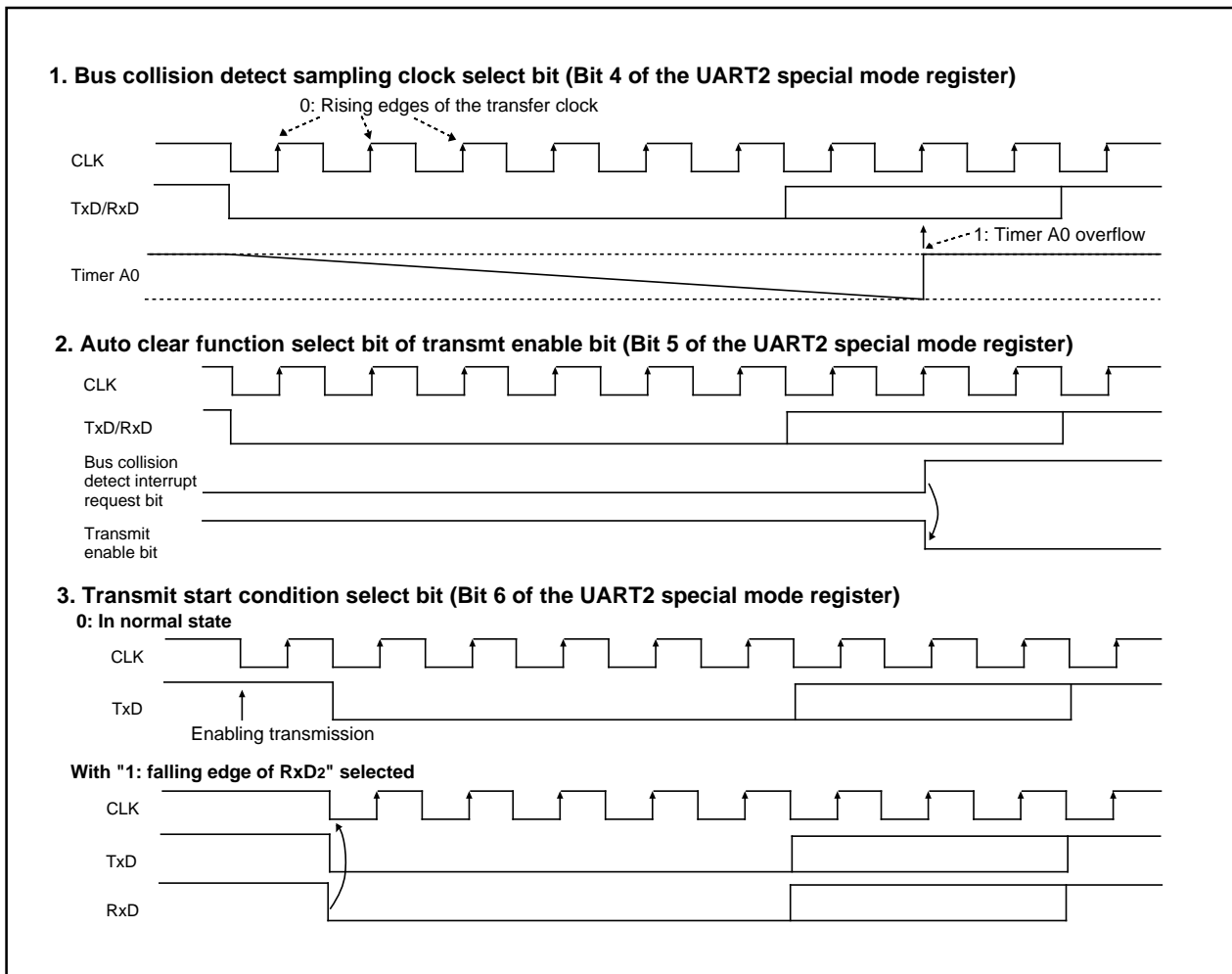


Figure 1.14.27. Some other functions added

## UART2 Special Mode Register 2

### UART2 Special Mode Register 2

UART2 special mode register 2 (address 0376<sub>16</sub>) is used to further control UART2 in I<sup>2</sup>C mode. Figure 1.14.28 shows the UART2 special mode register 2.

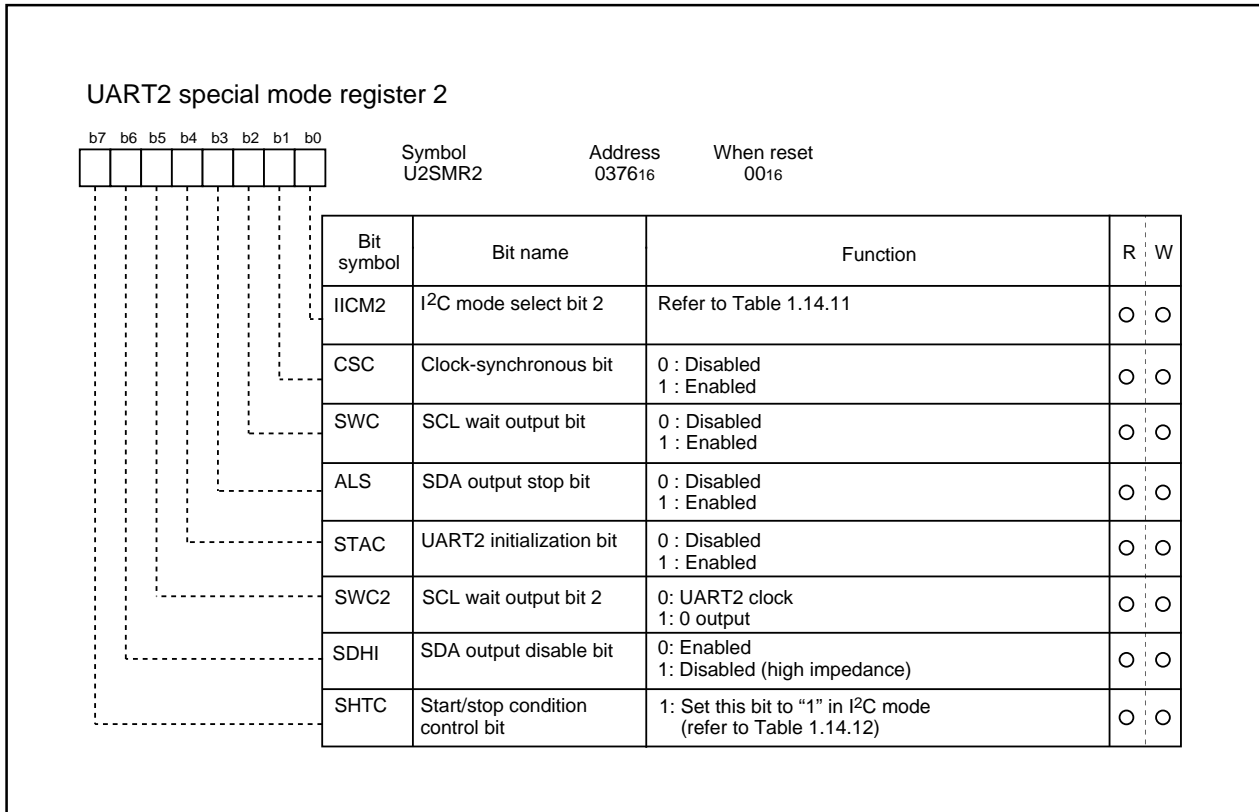


Figure 1.14.28. UART2 special mode register 2



## UART2 Special Mode Register 2

Bit 0 of the UART2 special mode register 2 (address 0376<sub>16</sub>) is used as the I<sup>2</sup>C mode select bit 2. Table 1.14.11 shows the types of control to be changed by I<sup>2</sup>C mode select bit 2 when the I<sup>2</sup>C mode select bit is set to "1". Table 1.14.12 shows the timing characteristics of detecting the start condition and the stop condition. Set the start/stop condition control bit (bit 7 of UART2 special mode register 2) to "1" in I<sup>2</sup>C mode.

**Table 1.14.11. Functions changed by I<sup>2</sup>C mode select bit 2**

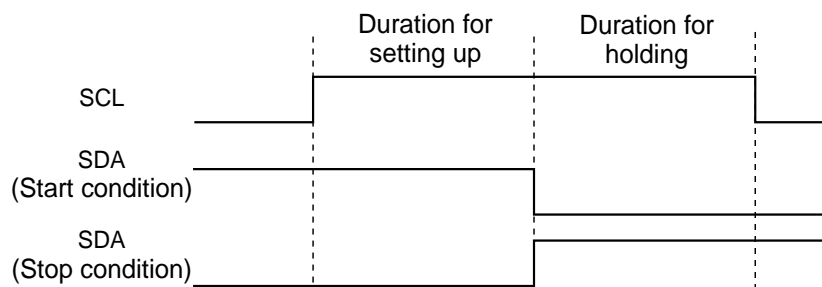
	Function	IICM2 = 0	IICM2 = 1
1	Factor of interrupt number 15	No acknowledgment detection (NACK)	UART2 transmission (the rising edge of the final bit of the clock)
2	Factor of interrupt number 16	Acknowledgment detection (ACK)	UART2 reception (the falling edge of the final bit of the clock)
3	DMA1 factor at the time when 1 1 0 1 is assigned to the DMA request factor selection bits	Acknowledgment detection (ACK)	UART2 reception (the falling edge of the final bit of the clock)
4	Timing for transferring data from the UART2 reception shift register to the reception buffer.	The rising edge of the final bit of the reception clock	The falling edge of the final bit of the reception clock
5	Timing for generating a UART2 reception/ACK interrupt request	The rising edge of the final bit of the reception clock	The falling edge of the final bit of the reception clock

**Table 1.14.12. Timing characteristics of detecting the start condition and the stop condition (Note 1)**

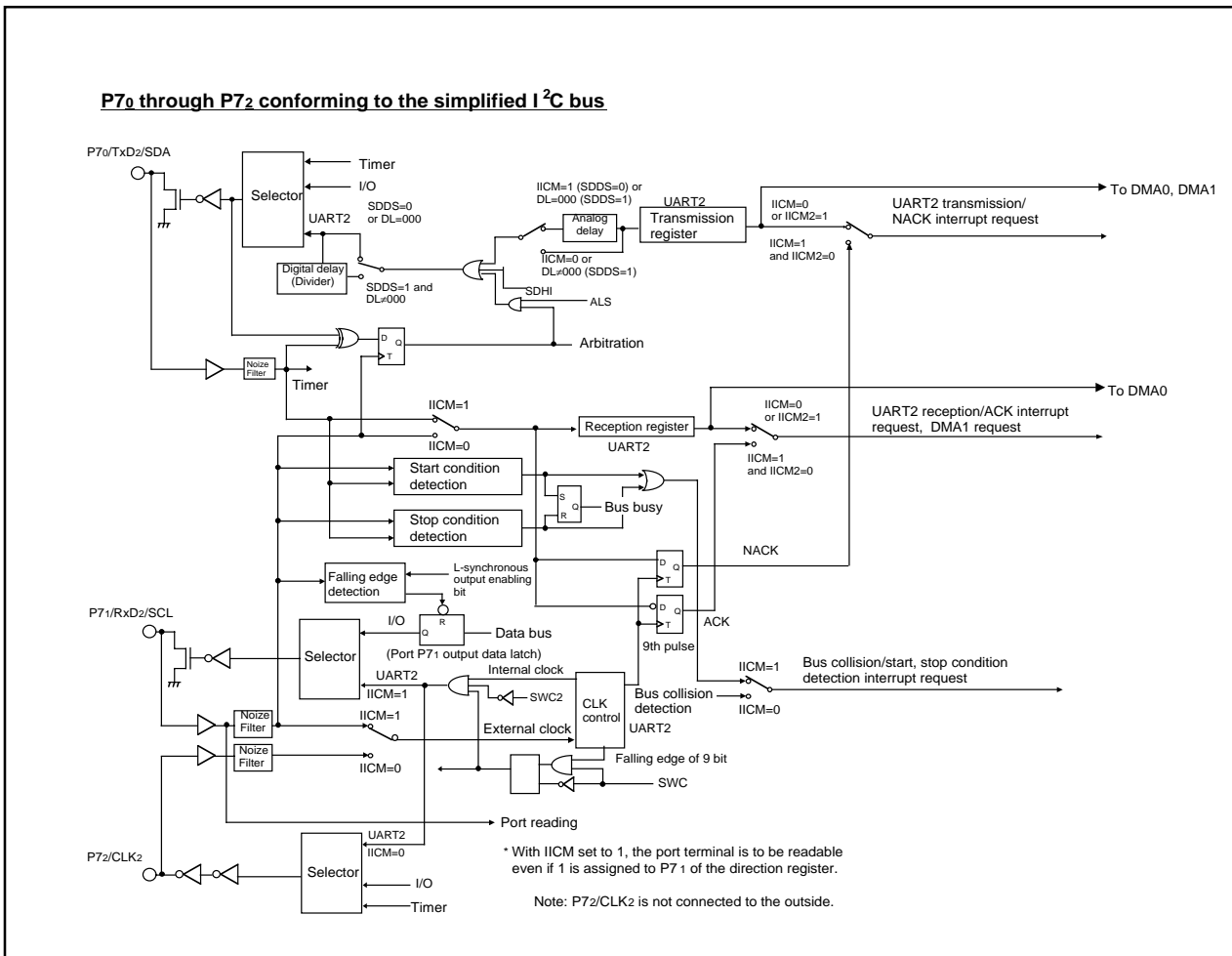
3 to 6 cycles < duration for setting-up (Note2)
3 to 6 cycles < duration for holding (Note2)

Note 1 : When the start/stop condition control bit SHTC is "1".

Note 2 : "cycles" is in terms of the input oscillation frequency  $f(X_{IN})$  of the main clock.



## UART2 Special Mode Register 2



**Figure 1.14.29. Functional block diagram for I<sup>2</sup>C mode**

Functions available in I<sup>2</sup>C mode are shown in Figure 1.14.29 — a functional block diagram.

Bit 3 of the UART2 special mode register 2 (address 037616) is used as the SDA output stop bit. Setting this bit to "1" causes an arbitration loss to occur, and the SDA pin turns to high-impedance state at the instant when the arbitration lost detecting flag is set to "1".

Bit 1 of the UART2 special mode register 2 (address 037616) is used as the clock synchronization bit. With this bit set to "1" at the time when the internal SCL is set to "H", the internal SCL turns to "L" if the falling edge is found in the SCL pin; and the baud rate generator reloads the set value, and start counting within the "L" interval. When the internal SCL changes from "L" to "H" with the SCL pin set to "L", stops counting the baud rate generator, and starts counting it again when the SCL pin turns to "H". Due to this function, the UART2 transmission-reception clock becomes the logical product of the signal flowing through the internal SCL and that flowing through the SCL pin. This function operates over the period from the moment earlier by a half cycle than falling edge of the UART2 first clock to the rising edge of the ninth bit. To use this function, choose the internal clock for the transfer clock.

Bit 2 of the UART2 special mode register 2 (037616) is used as the SCL wait output bit. Setting this bit to "1" causes the SCL pin to be fixed to "L" at the falling edge of the ninth bit of the clock. Setting this bit to "0" frees the output fixed to "L".

## UART2 Special Mode Register 2

---

Bit 4 of the UART2 special mode register 2 (address 0376<sub>16</sub>) is used as the UART2 initialization bit. Setting this bit to "1", and when the start condition is detected, the microcomputer operates as follows.

- (1) The transmission shift register is initialized, and the content of the transmission register is transferred to the transmission shift register. This starts transmission by dealing with the clock entered next as the first bit. The UART2 output value, however, doesn't change until the first bit data is output after the entrance of the clock, and remains unchanged from the value at the moment when the microcomputer detected the start condition.
- (2) The reception shift register is initialized, and the microcomputer starts reception by dealing with the clock entered next as the first bit.
- (3) The SCL wait output bit turns to "1". This turns the SCL pin to "L" at the falling edge of the ninth bit of the clock.

Starting to transmit/receive signals to/from UART2 using this function doesn't change the value of the transmission buffer empty flag. To use this function, choose the external clock for the transfer clock.

Bit 5 of the UART2 special mode register 2 (0376<sub>16</sub>) is used as the SCL pin wait output bit 2. Setting this bit to "1" with the serial I/O specified allows the user to forcibly output an "L" from the SCL pin even if UART2 is in operation. Setting this bit to "0" frees the "L" output from the SCL pin, and the UART2 clock is input/output.

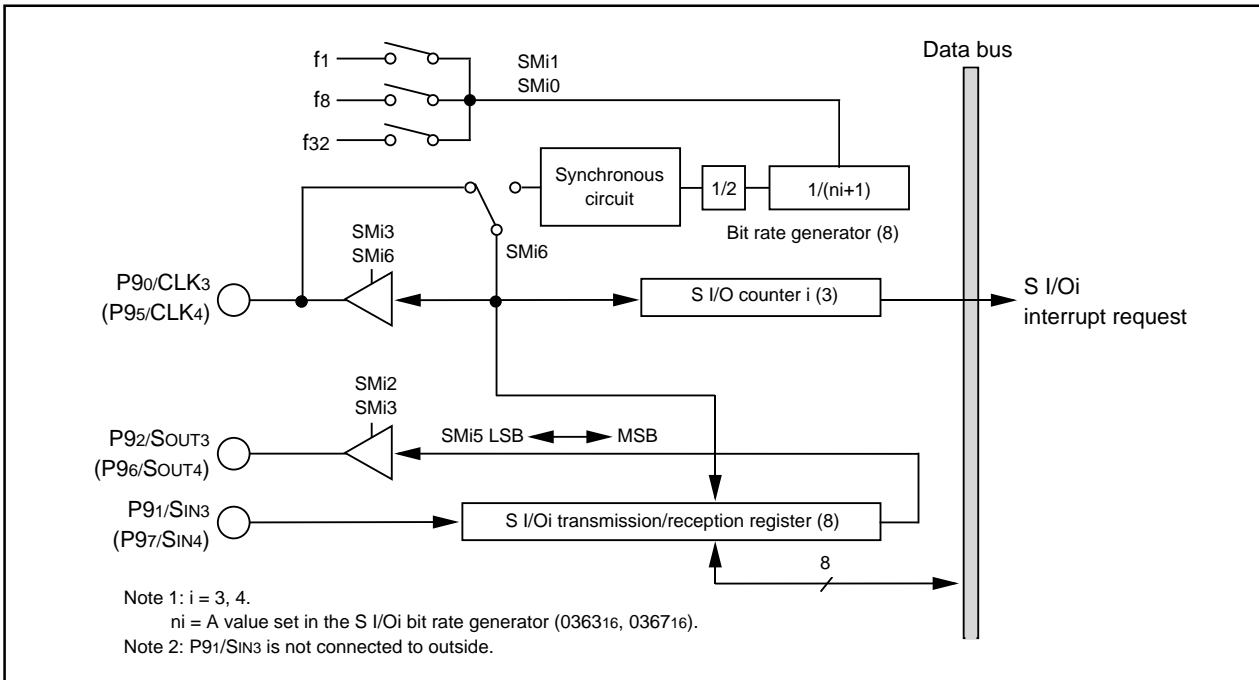
Bit 6 of the UART2 special mode register 2 (0376<sub>16</sub>) is used as the SDA output disable bit. Setting this bit to "1" forces the SDA pin to turn to the high-impedance state. Refrain from changing the value of this bit at the rising edge of the UART2 transfer clock. There can be instances in which arbitration lost detecting flag is turned on.

**S I/O3, 4**

S I/O3 and S I/O4 are exclusive clock-synchronous serial I/Os.

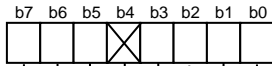
Figure 1.14.30 shows the S I/O3, 4 block diagram, and Figure 1.14.31 shows the S I/O3, 4 related register.

Table 1.14.13 shows the specifications of S I/O3, 4.



**Figure 1.14.30. S I/O3, 4 block diagram**

**S I/Oi control register (i = 3, 4) (Note 1)**

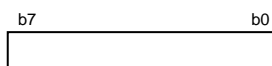


Symbol      Address      When reset  
 SiC      0362<sub>16</sub>, 0366<sub>16</sub>      40<sub>16</sub>

Bit symbol	Bit name	Description	R	W
SMi0	Internal synchronous clock select bit	b1 b0 0 0 : Selecting f1 0 1 : Selecting f8 1 0 : Selecting f32 1 1 : Must not be set.	○	○
SMi1			○	○
SMi2	Souri output disable bit	0 : Sour <sub>i</sub> output 1 : Sour <sub>i</sub> output disable(high impedance)	○	○
SMi3	S I/Oi port select bit (Note 2)	0 : Input-output port 1 : Sour <sub>i</sub> output, CLK function	○	○
Nothing is assigned. In an attempt to write to this bit, write "0". The value, if read, turns out to be "0".			—	—
SMi5	Transfer direction select bit	0 : LSB first 1 : MSB first	○	○
SMi6	Synchronous clock select bit (Note 2)	0 : External clock 1 : Internal clock	○	○
SMi7	Souri initial value set bit	Effective when SMi3 = 0 0 : L output 1 : H output	○	○

Note 1: Set "1" in bit 2 of the protection register (000A<sub>16</sub>) in advance to write to the S I/Oi control register (i = 3, 4).  
 Note 2: When using the port as an input/output port by setting the S I/Oi port select bit (i = 3, 4) to "0", be sure to set the sync clock select bit to "1".

**S I/Oi bit rate generator (Note 1, 2)**

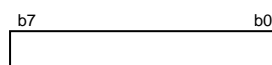


Symbol      Address      When reset  
 S3BRG      0363<sub>16</sub>      Indeterminate  
 S4BRG      0367<sub>16</sub>      Indeterminate

Indeterminate	Values that can be set	R	W
Assuming that set value = n, BRGi divides the count source by n + 1	00 <sub>16</sub> to FF <sub>16</sub>	○	○

Note 1: Write a value to this register while transmit/receive halts.  
 Note 2: Use MOV instruction to write to this register.

**S I/Oi transmit/receive register (Note 1, 2)**



Symbol      Address      When reset  
 S3TRR      0360<sub>16</sub>      Indeterminate  
 S4TRR      0364<sub>16</sub>      Indeterminate

Indeterminate	R	W
Transmission/reception starts by writing data to this register. After transmission/reception finishes, reception data is input.	○	○

Note 1: S I/O3 is exclusive to transmission.  
 Note 2: Write a value to this register while transmit/receive halts.

**Figure 1.14.31. S I/O3, 4 related register**

**Table 1.14.13. Specifications of S I/O3, 4**

Item	Specifications
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>• With the internal clock selected (bit 6 of 0362<sub>16</sub>, 0366<sub>16</sub> = "1"): <math>f1/2(ni+1)</math>, <math>f8/2(ni+1)</math>, <math>f32/2(ni+1)</math> (Note 1)</li> <li>• With the external clock selected (bit 6 of 0362<sub>16</sub>, 0366<sub>16</sub> = 0): Input from the CLK<sub>i</sub> terminal (Note 2)</li> </ul>
Conditions for transmission/reception start	<ul style="list-style-type: none"> <li>• To start transmit/reception, the following requirements must be met: <ul style="list-style-type: none"> <li>- Select the synchronous clock (use bit 6 of 0362<sub>16</sub>, 0366<sub>16</sub>).</li> <li>Select a frequency dividing ratio if the internal clock has been selected (use bits 0 and 1 of 0362<sub>16</sub>, 0366<sub>16</sub>).</li> <li>- SOUT<sub>i</sub> initial value set bit (use bit 7 of 0362<sub>16</sub>, 0366<sub>16</sub>) = 1.</li> <li>- S I/Oi port select bit (bit 3 of 0362<sub>16</sub>, 0366<sub>16</sub>) = 1.</li> <li>- Select the transfer direction (use bit 5 of 0362<sub>16</sub>, 0366<sub>16</sub>)</li> <li>- Write transfer data to SI/Oi transmit/receive register (0360<sub>16</sub>, 0364<sub>16</sub>)</li> </ul> </li> <li>• To use S I/Oi interrupts, the following requirements must be met: <ul style="list-style-type: none"> <li>- Clear the SI/Oi interrupt request bit before writing transfer data to the SI/Oi transmit/receive register (bit 3 of 0049<sub>16</sub>, 0048<sub>16</sub>) = 0.</li> </ul> </li> </ul>
Interrupt request generation timing	<ul style="list-style-type: none"> <li>• Rising edge of the last transfer clock. (Note 3)</li> </ul>
Select function	<ul style="list-style-type: none"> <li>• LSB first or MSB first selection Whether transmission/reception begins with bit 0 (LSB) or bit 7 (MSB) can be selected.</li> <li>• Function for setting an SOUT<sub>i</sub> initial value selection When using an external clock for the transfer clock, the user can choose the SOUT<sub>i</sub> pin output level during a non-transfer time. For details on how to set, see Figure 1.14.33.</li> </ul>
Precaution	<ul style="list-style-type: none"> <li>• Unlike UART0–2, SI/Oi (i = 3, 4) is not divided for transfer register and buffer. Therefore, do not write the next transfer data to the SI/Oi transmit/receive register (addresses 0360<sub>16</sub>, 0364<sub>16</sub>) during a transfer.</li> <li>• When the internal clock is selected for the transfer clock, SOUT<sub>i</sub> holds the last data for a 1/2 transfer clock period after it finished transferring and then goes to a high-impedance state. However, if the transfer data is written to the SI/Oi transmit/receive register (addresses 0360<sub>16</sub>, 0364<sub>16</sub>) during this time, SOUT<sub>i</sub> is placed in the high-impedance state immediately upon writing and the data hold time is thereby reduced.</li> </ul>

Note 1: n is a value from 00<sub>16</sub> through FF<sub>16</sub> set in the S I/Oi bit rate generator (i = 3, 4).

Note 2: With the external clock selected:

- Before data can be written to the SI/Oi transmit/receive register (addresses 0360<sub>16</sub>, 0364<sub>16</sub>), the CLK<sub>i</sub> pin input must be in the high state. Also, before rewriting the SI/Oi control register (addresses 0362<sub>16</sub>, 0366<sub>16</sub>)'s bit 7 (SOUT<sub>i</sub> initial value set bit), make sure the CLK<sub>i</sub> pin input is held high.
- The S I/Oi circuit keeps on with the shift operation as long as the synchronous clock is entered in it, so stop the synchronous clock at the instant when it counts to eight. The internal clock, if selected, automatically stops.

Note 3: If the internal clock is used for the synchronous clock, the transfer clock signal stops at the "H" state.

Note 4: SI/O3 is provided with no connection to the external pin, so is used exclusively for transmission.

■ **Functions for setting an SOUTi initial value**

When using an external clock for the transfer clock, the SOUTi pin output level during a non-transfer time can be set to the high or the low state. Figure 1.14.32 shows the timing chart for setting an SOUTi initial value and how to set it.

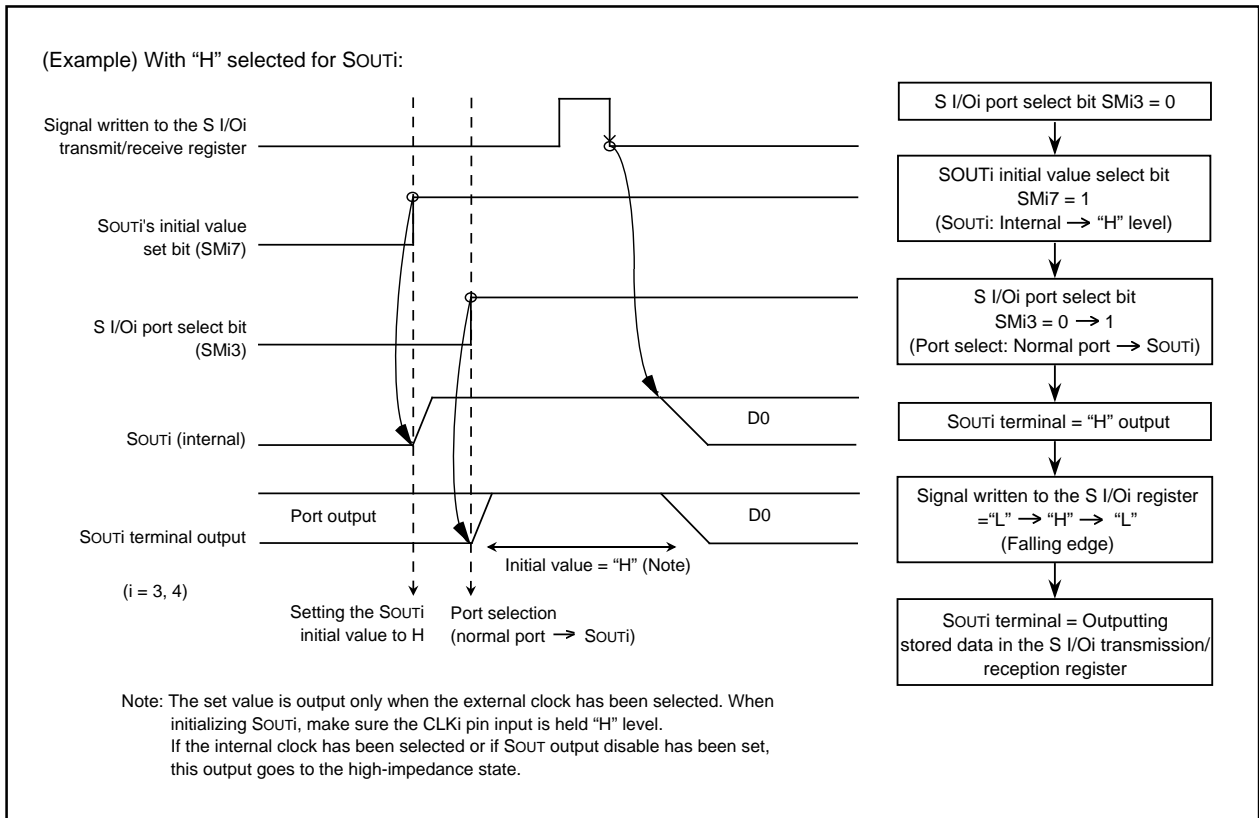


Figure 1.14.32. Timing chart for setting SOUTi's initial value and how to set it

■ **S I/Oi operation timing**

Figure 1.14.33 shows the S I/Oi operation timing

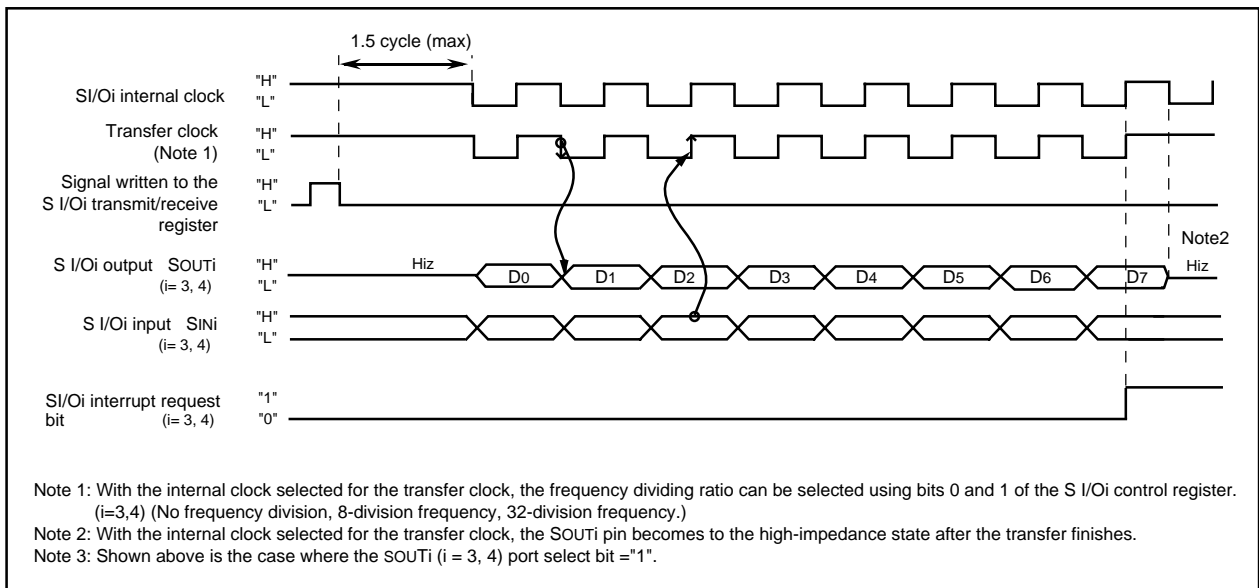


Figure 1.14.33. S I/Oi operation timing chart

## 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 P10<sub>0</sub> to P10<sub>7</sub>, P9<sub>5</sub>, and P9<sub>6</sub> 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 03D7<sub>16</sub>) 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 03D7<sub>16</sub> 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 1.15.1 shows the performance of the A-D converter. Figure 1.15.1 shows the block diagram of the A-D converter, and Figures 1.15.2 and 1.15.3 show the A-D converter-related registers.

**Table 1.15.1. 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\text{LSB}</math></li> <li>• With sample and hold function (8-bit resolution)  <math>\pm 2\text{LSB}</math></li> <li>• With sample and hold function (10-bit resolution)                AN<sub>0</sub> to AN<sub>7</sub> input : <math>\pm 3\text{LSB}</math>                ANEX<sub>0</sub> and ANEX<sub>1</sub> input (including mode in which external operation amp is connected) : <math>\pm 7\text{LSB}</math></li> </ul> VCC = 3V <ul style="list-style-type: none"> <li>• Without sample and hold function (8-bit resolution)  <math>\pm 2\text{LSB}</math></li> </ul>
Operating modes	One-shot mode, repeat mode, single sweep mode, repeat sweep mode 0, and repeat sweep mode 1
Analog input pins	8pins (AN <sub>0</sub> to AN <sub>7</sub> ) + 2pins (ANEX <sub>0</sub> and ANEX <sub>1</sub> )
A-D conversion start condition	<ul style="list-style-type: none"> <li>• Software trigger                A-D conversion starts when the A-D conversion start flag changes to "1"</li> <li>• External trigger (can be retrIGGERED)                A-D conversion starts when the A-D conversion start flag is "1" and the <math>\overline{\text{ADTRG/P97}}</math> input changes from "H" to "L"</li> </ul>
Conversion speed per pin	<ul style="list-style-type: none"> <li>• Without sample and hold function                8-bit resolution: 49 <math>\phi_{AD}</math> cycles, 10-bit resolution: 59 <math>\phi_{AD}</math> cycles</li> <li>• With sample and hold function                8-bit resolution: 28 <math>\phi_{AD}</math> cycles, 10-bit resolution: 33 <math>\phi_{AD}</math> cycles</li> </ul>

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

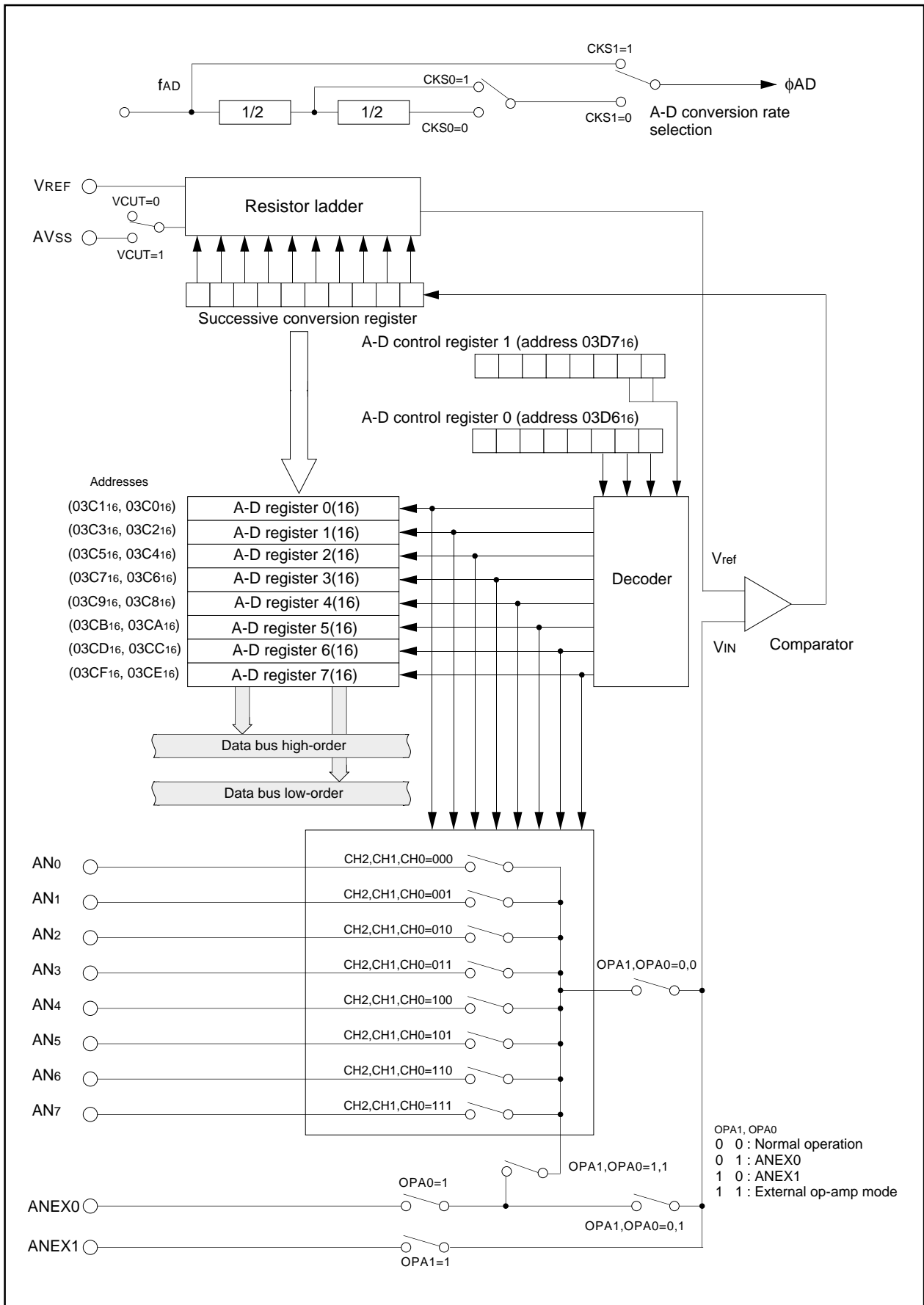
Note 2: Divide the frequency if f(XIN) exceeds 10MHz, and make  $\phi_{AD}$  frequency equal to or less than 10MHz.

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 1.15.1. Block diagram of A-D converter**

A-D Converter

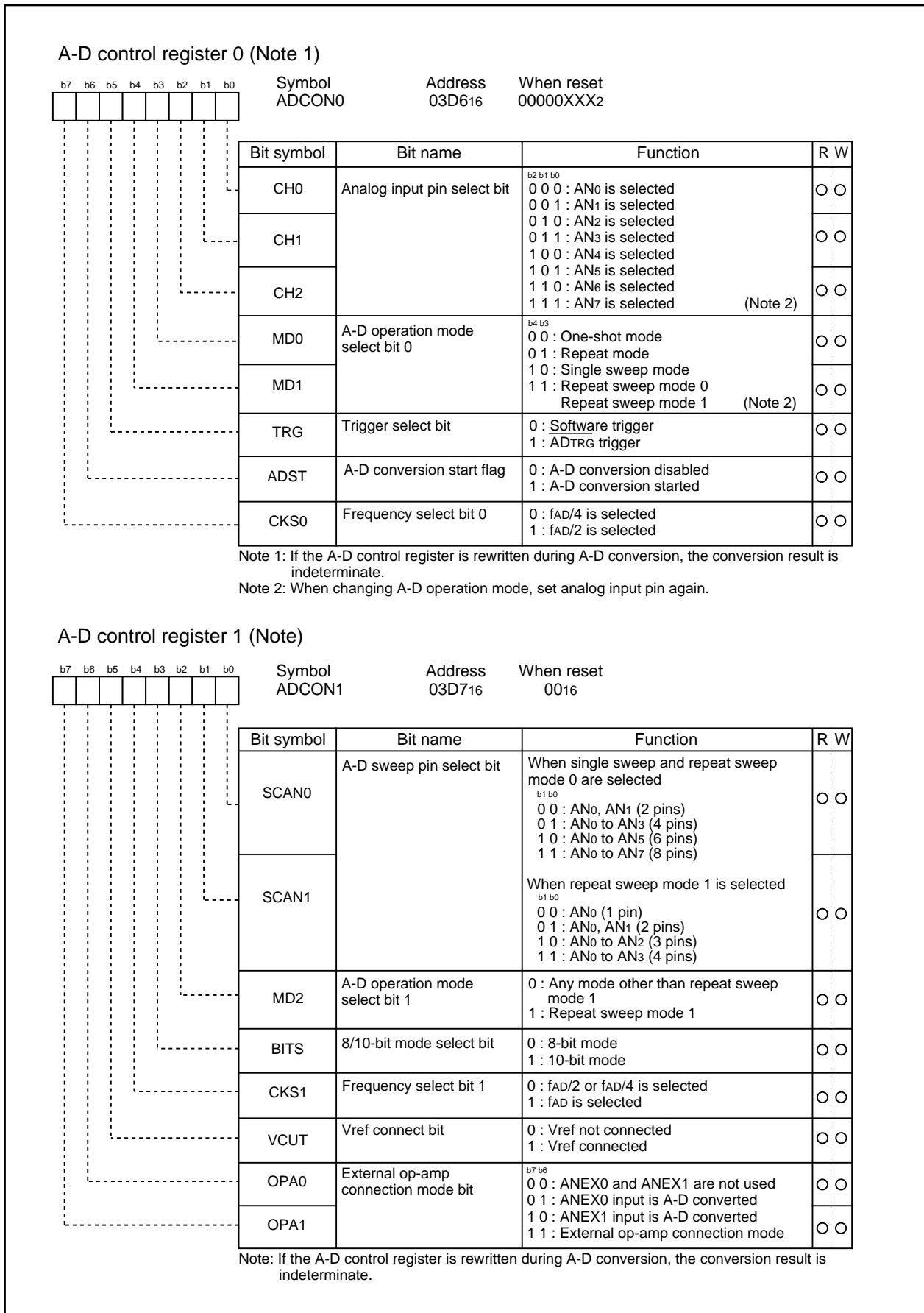


Figure 1.15.2. A-D converter-related registers (1)

A-D Converter

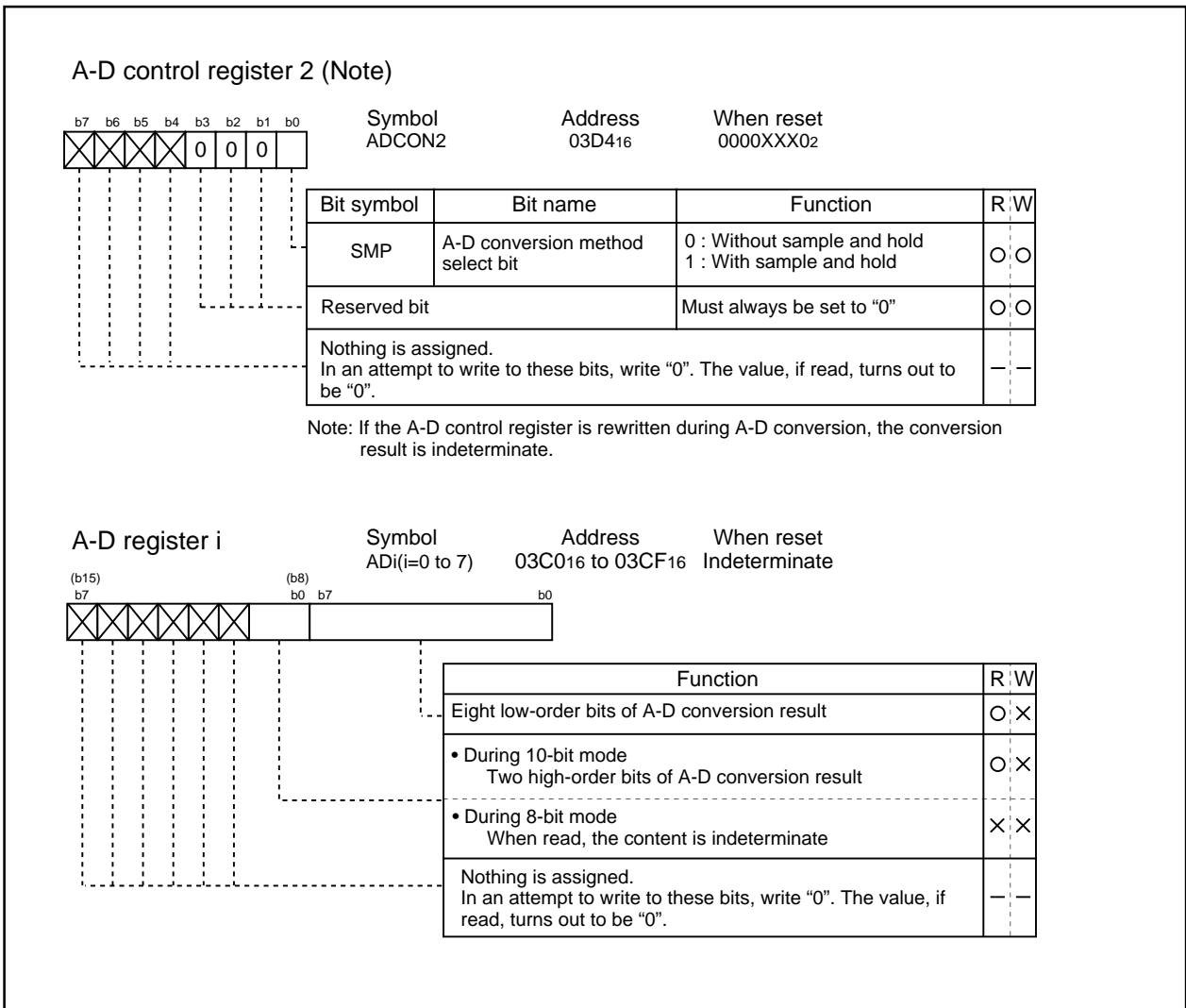


Figure 1.15.3. A-D converter-related registers (2)

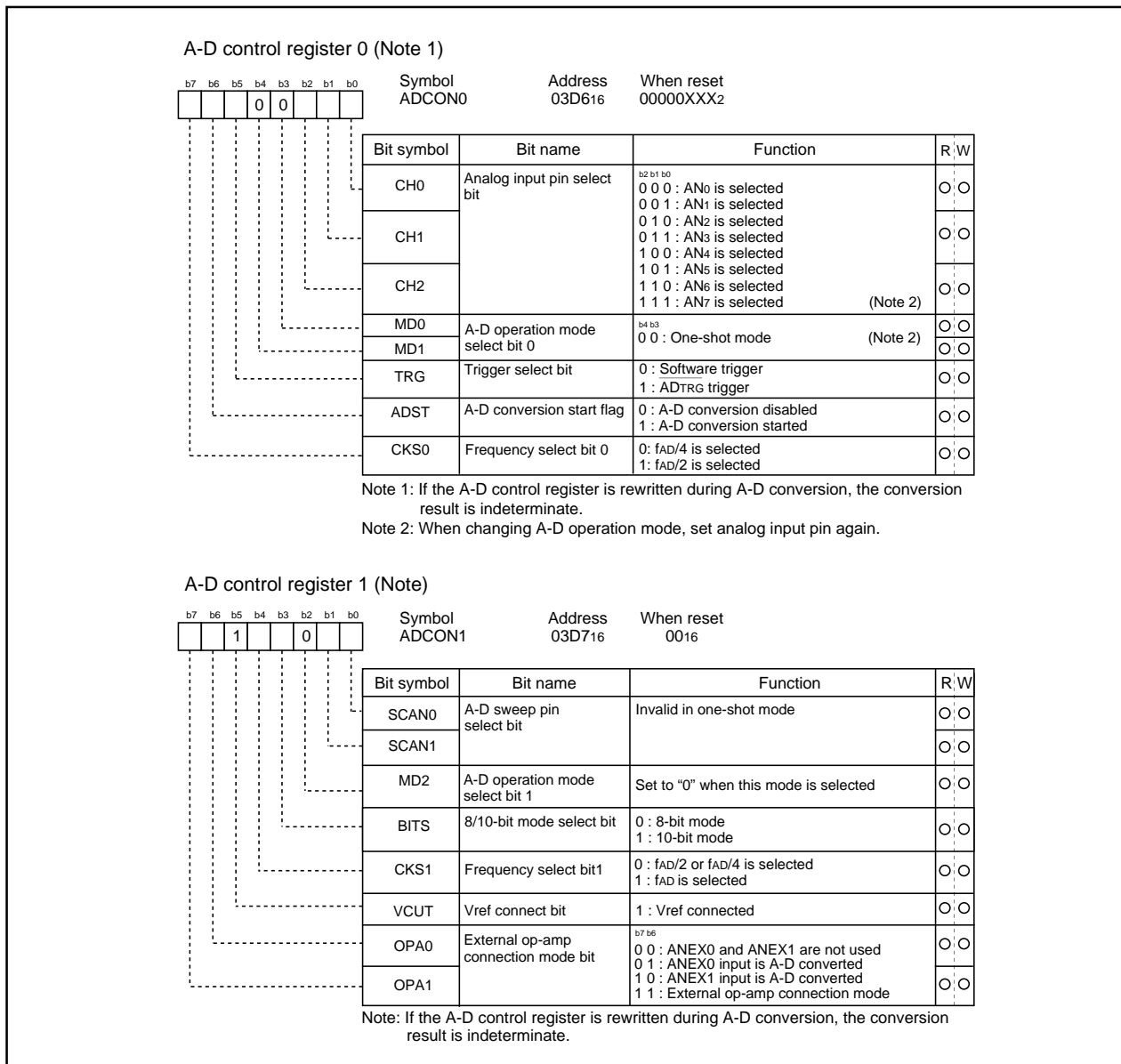
A-D Converter

**(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. Table 1.15.2 shows the specifications of one-shot mode. Figure 1.15.4 shows the A-D control register in one-shot mode.

**Table 1.15.2. 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", except when external trigger is selected)</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
Reading of result of A-D converter	Read A-D register corresponding to selected pin



**Figure 1.15.4. A-D conversion register in one-shot mode**

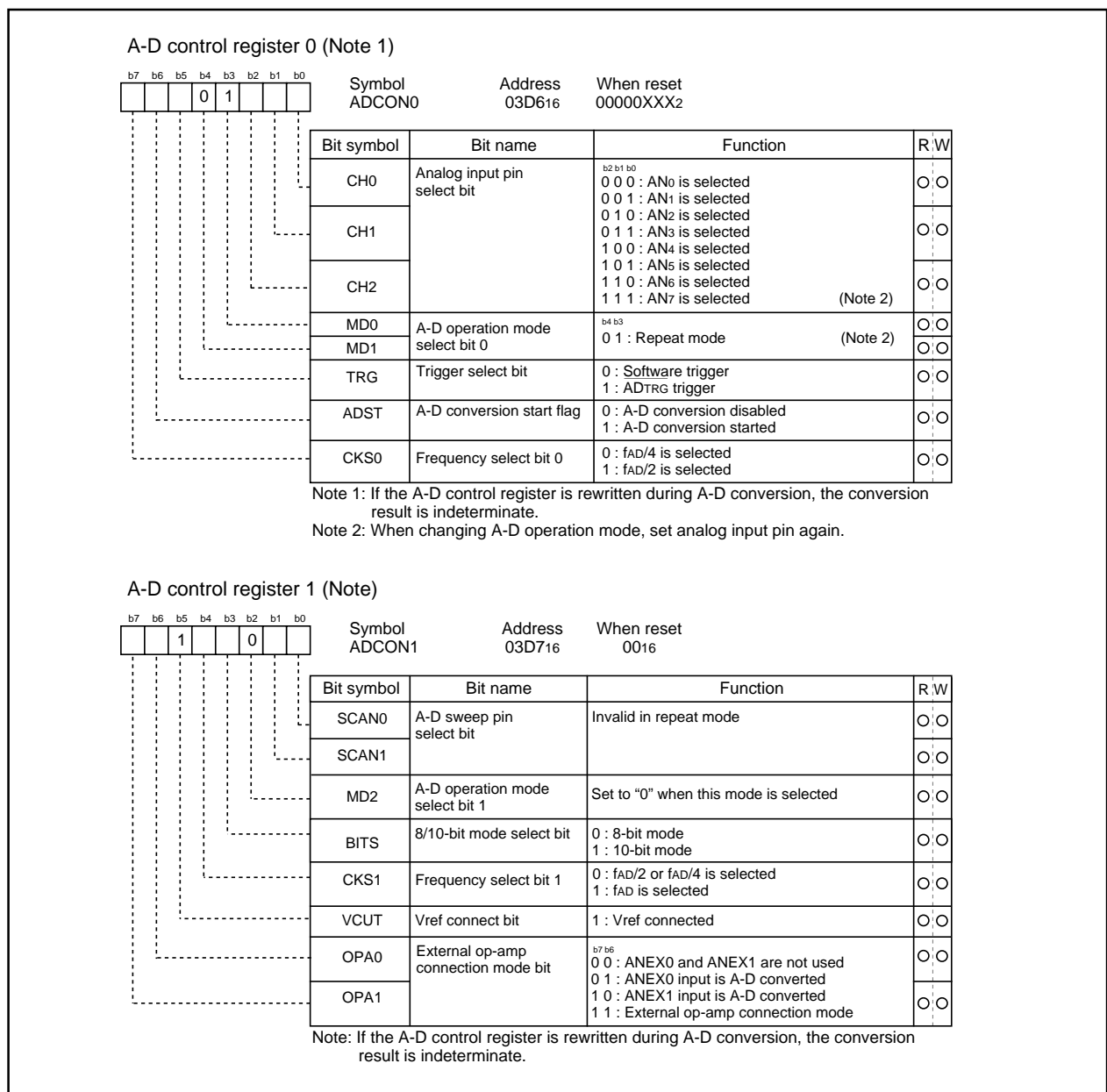
A-D Converter

**(2) Repeat mode**

In repeat mode, the pin selected using the analog input pin select bit is used for repeated A-D conversion. Table 1.15.3 shows the specifications of repeat mode. Figure 1.15.5 shows the A-D control register in repeat mode.

**Table 1.15.3. Repeat mode specifications**

Item	Specification
Function	The pin selected by the analog input pin select bit is used for repeated A-D conversion
Star 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
Reading of result of A-D converter	Read A-D register corresponding to selected pin (at any time)



**Figure 1.15.5. A-D conversion register in repeat mode**

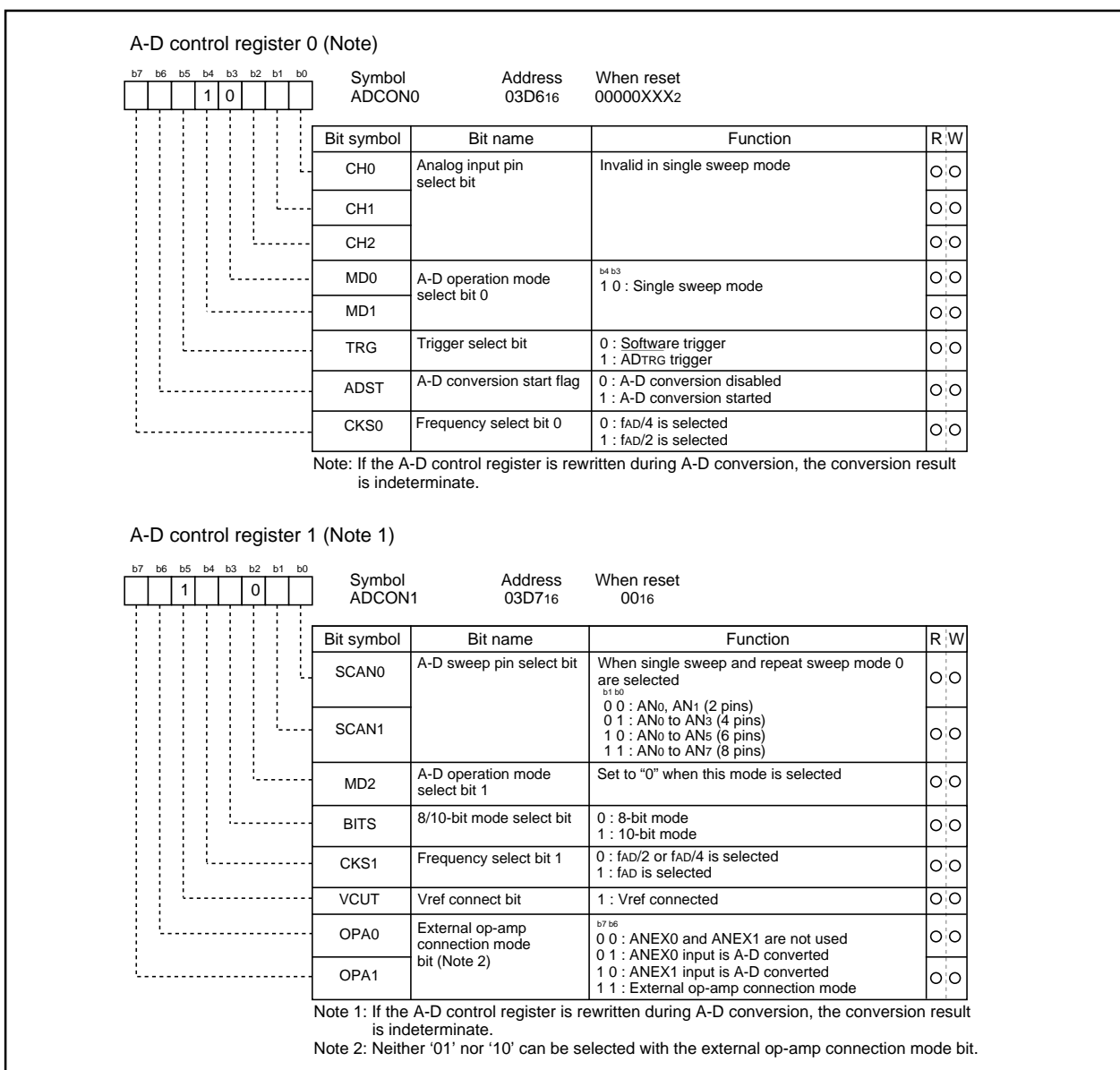
A-D Converter

**(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. Table 1.15.4 shows the specifications of single sweep mode. Figure 1.15.6 shows the A-D control register in single sweep mode.

**Table 1.15.4. 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", except when external trigger is selected)</li> <li>Writing "0" to A-D conversion start flag</li> </ul>
Interrupt request generation timing	End of A-D conversion
Input pin	AN <sub>0</sub> and AN <sub>1</sub> (2 pins), AN <sub>0</sub> to AN <sub>3</sub> (4 pins), AN <sub>0</sub> to AN <sub>5</sub> (6 pins), or AN <sub>0</sub> to AN <sub>7</sub> (8 pins)
Reading of result of A-D converter	Read A-D register corresponding to selected pin



**Figure 1.15.6. A-D conversion register in single sweep mode**

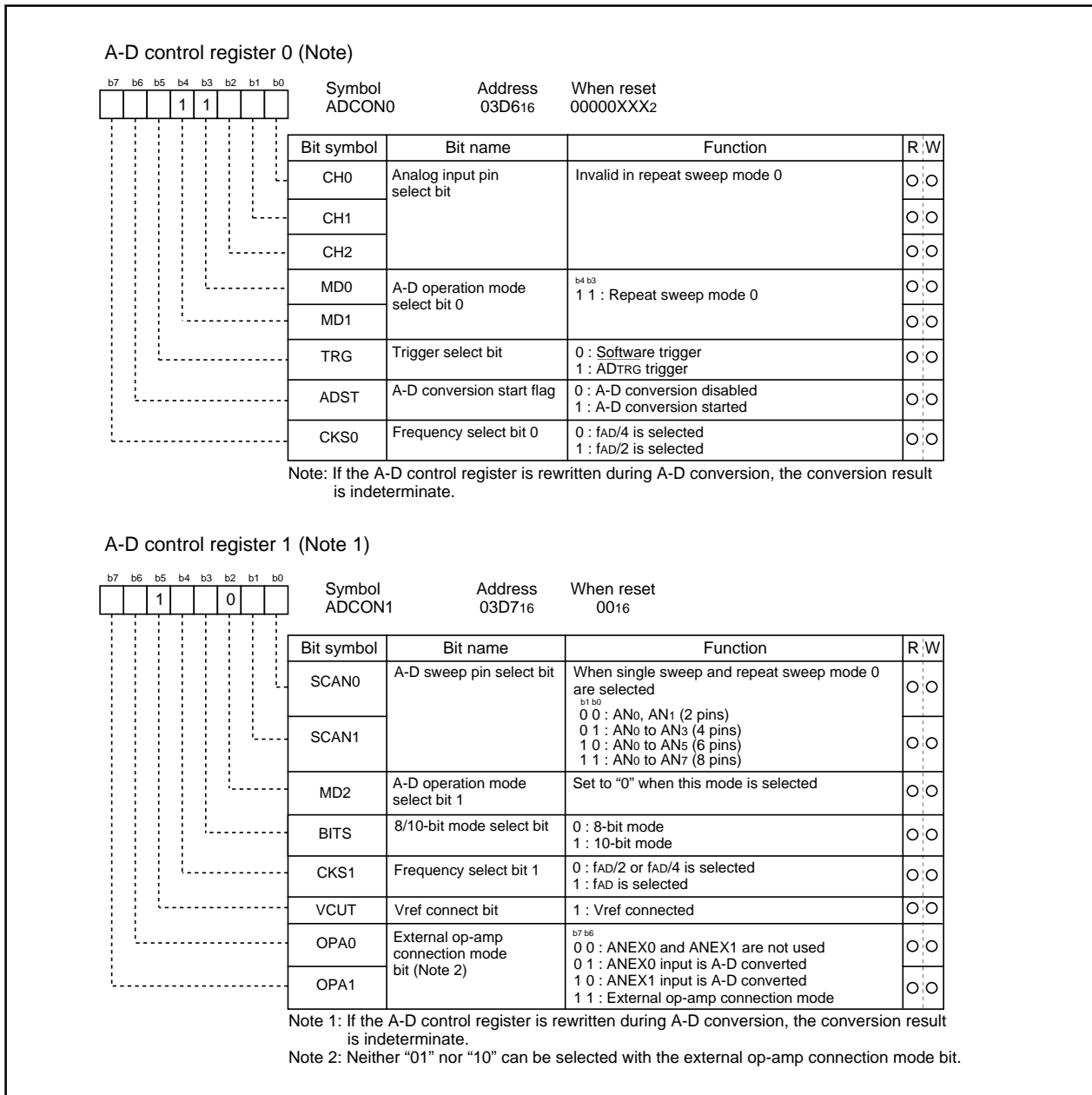
A-D Converter

**(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. Table 1.15.5 shows the specifications of repeat sweep mode 0. Figure 1.15.7 shows the A-D control register in repeat sweep mode 0.

**Table 1.15.5. Repeat sweep mode 0 specifications**

Item	Specification
Function	The pins selected by the A-D sweep pin select bit are used for repeat 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)
Reading of result of A-D converter	Read A-D register corresponding to selected pin (at any time)



**Figure 1.15.7. A-D conversion register in repeat sweep mode 0**

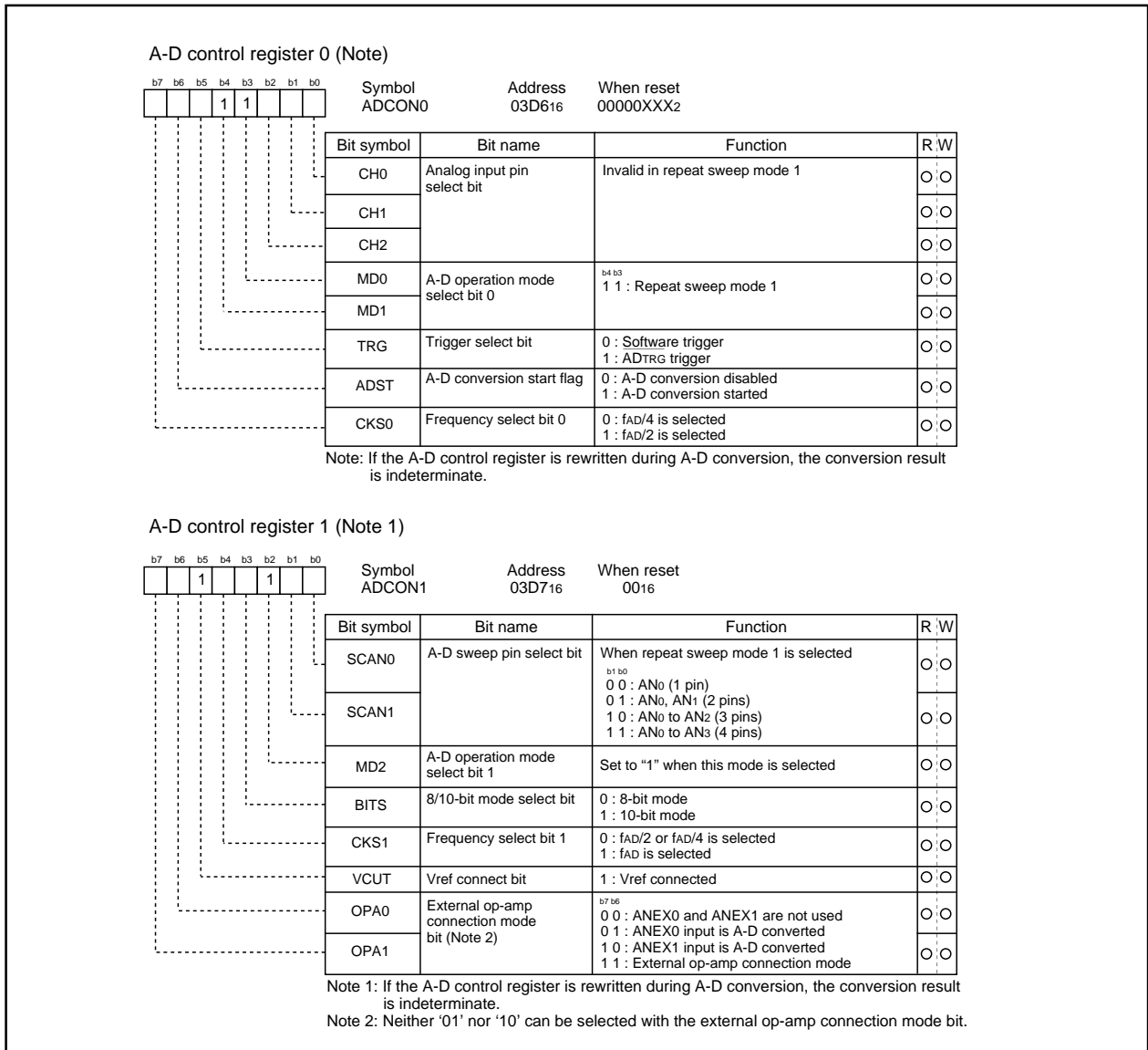
A-D Converter

**(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. Table 1.15.6 shows the specifications of repeat sweep mode 1. Figure 1.15.8 shows the A-D control register in repeat sweep mode 1.

**Table 1.15.6. Repeat sweep mode 1 specifications**

Item	Specification
Function	All pins perform repeat 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	With emphasis on these pins ; 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)
Reading of result of A-D converter	Read A-D register corresponding to selected pin (at any time)



**Figure 1.15.8. A-D conversion register in repeat sweep mode 1**



**(a) 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  $\emptyset$ AD cycle is achieved with 8-bit resolution and 33  $\emptyset$ 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.

**(b) Extended analog input pins**

In one-shot mode and repeat mode, the input via the extended analog input pins ANEX0 and ANEX1 can also be converted from analog to digital.

When bit 6 of the A-D control register 1 (address 03D716) is "1" and bit 7 is "0", input via ANEX0 is converted from analog to digital. The result of conversion is stored in A-D register 0.

When bit 6 of the A-D control register 1 (address 03D716) is "0" and bit 7 is "1", input via ANEX1 is converted from analog to digital. The result of conversion is stored in A-D register 1.

**(c) External operation amp connection mode**

In this mode, multiple external analog inputs via the extended analog input pins, ANEX0 and ANEX1, can be amplified together by just one operation amp and used as the input for A-D conversion.

When bit 6 of the A-D control register 1 (address 03D716) is "1" and bit 7 is "1", input via AN0 to AN7 is output from ANEX0. The input from ANEX1 is converted from analog to digital and the result stored in the corresponding A-D register. The speed of A-D conversion depends on the response of the external operation amp. Do not connect the ANEX0 and ANEX1 pins directly. Figure 1.15.9 is an example of how to connect the pins in external operation amp mode.

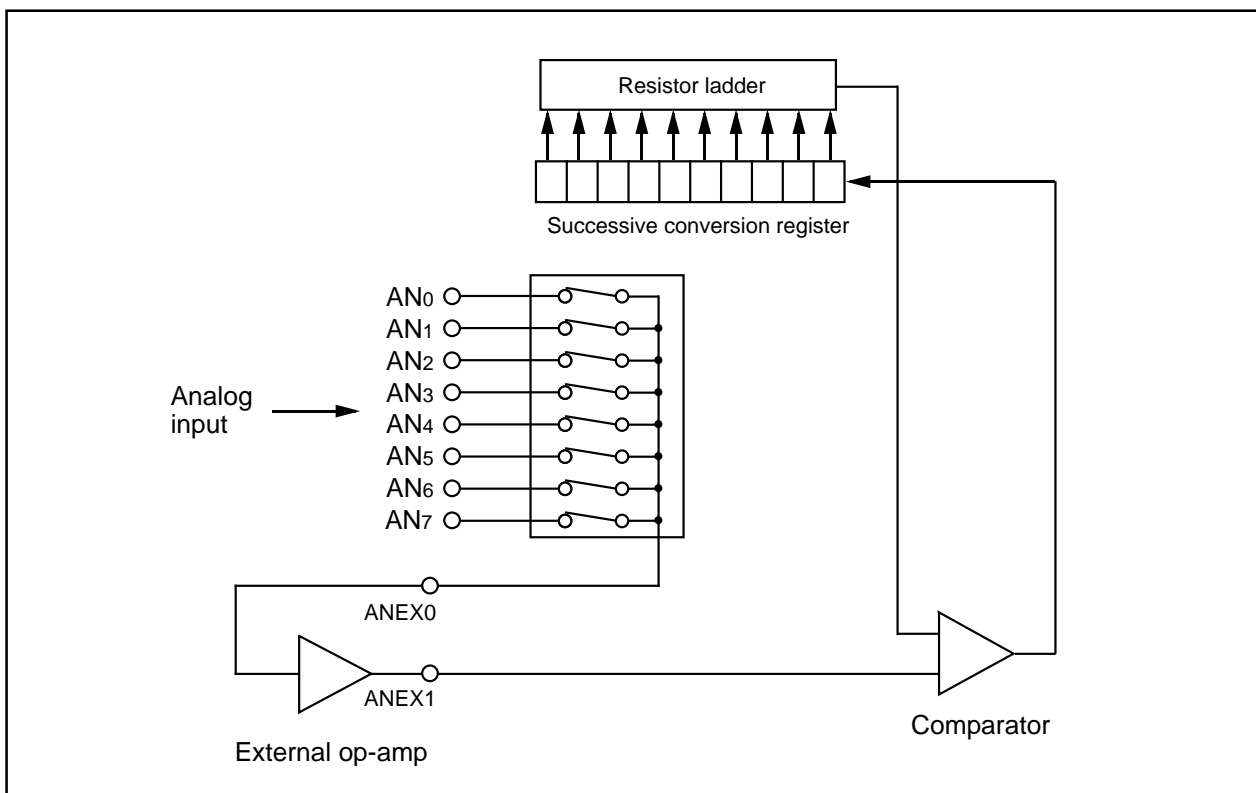


Figure 1.15.9. Example of external op-amp connection mode

## D-A Converter

### D-A Converter

This is an 8-bit, R-2R type D-A converter. The microcomputer contains two independent D-A converters of this type.

D-A conversion is performed when a value is written to the corresponding D-A register. Bits 0 and 1 (D-A output enable bits) of the D-A control register decide if the result of conversion is to be output. Do not set the target port to output mode if D-A conversion is to be performed. When the D-A output is enabled, the pull-up function of the corresponding port is automatically disabled.

Output analog voltage (V) is determined by a set value (n : decimal) in the D-A register.

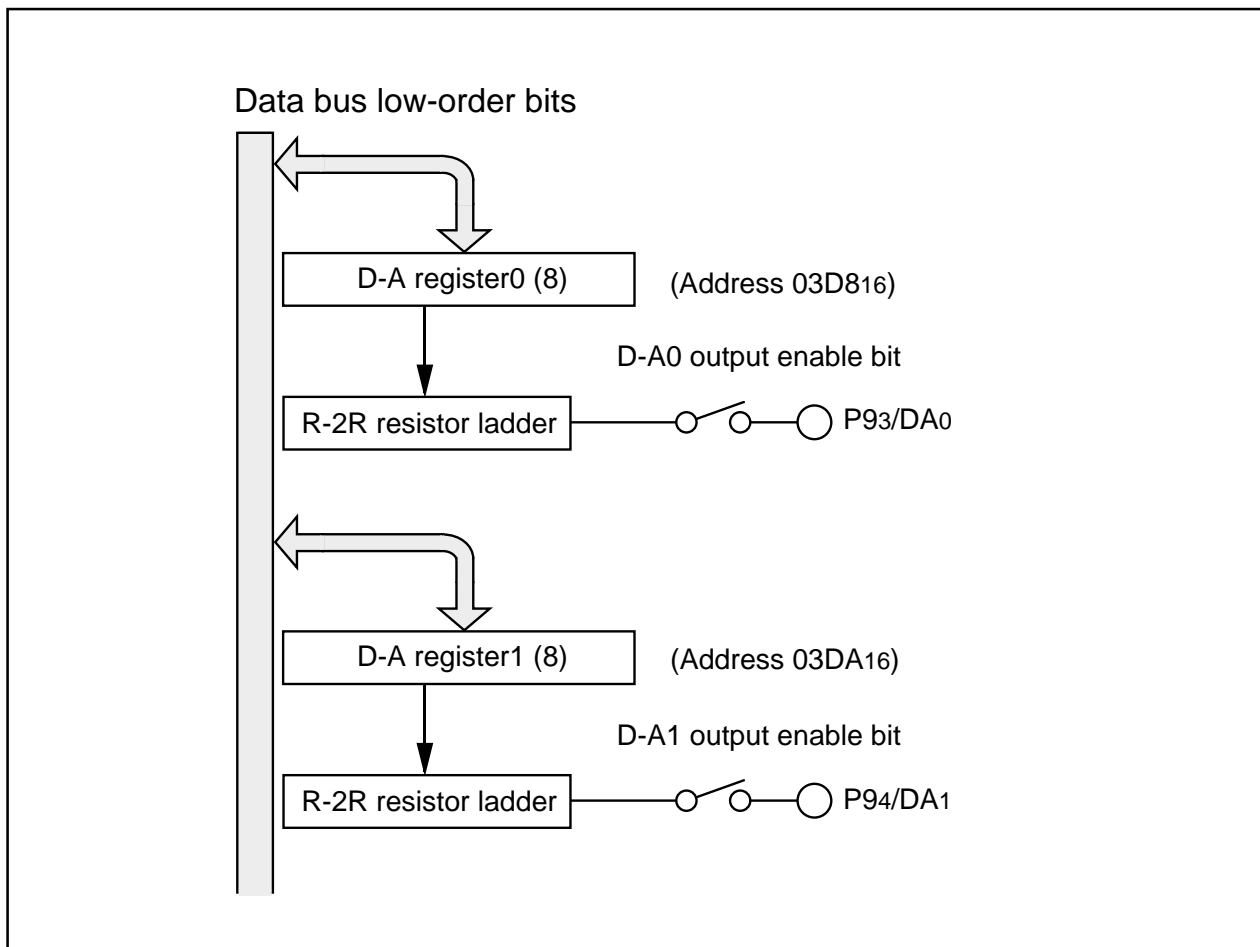
$$V = V_{REF} \times n / 256 \quad (n = 0 \text{ to } 255)$$

$V_{REF}$  : reference voltage

Table 1.16.1 lists the performance of the D-A converter. Figure 1.16.1 shows the block diagram of the D-A converter. Figure 1.16.2 shows the D-A control register. Figure J1.16.3 shows the D-A converter equivalent circuit.

**Table 1.16.1. Performance of D-A converter**

Item	Performance
Conversion method	R-2R method
Resolution	8 bits
Analog output pin	2 channels



**Figure 1.16.1. Block diagram of D-A converter**

D-A Converter

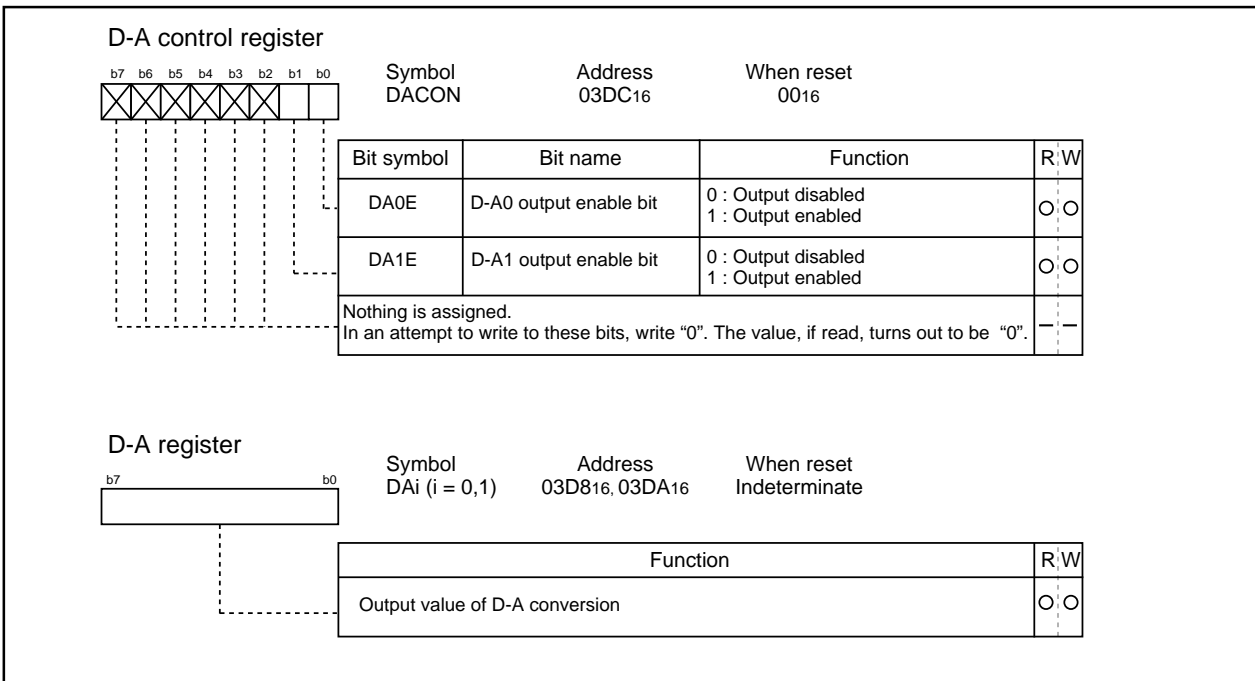


Figure 1.16.2. D-A control register

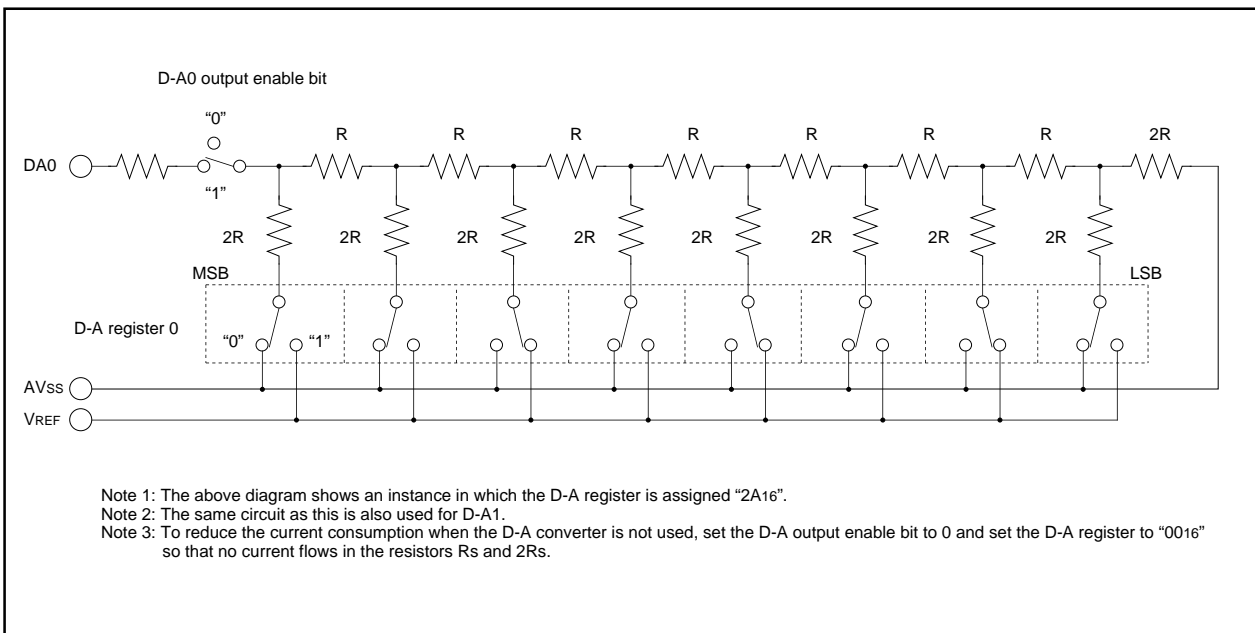


Figure 1.16.3. D-A converter equivalent circuit

## CRC

## CRC Calculation Circuit

The Cyclic Redundancy Check (CRC) calculation circuit detects an error in data blocks. The microcomputer uses a generator polynomial of CRC\_CCITT ( $X^{16} + X^{12} + X^5 + 1$ ) to generate CRC code.

The CRC code is a 16-bit code generated for a block of a given data length in multiples of 8 bits. The CRC code is set in a CRC data register each time one byte of data is transferred to a CRC input register after writing an initial value into the CRC data register. Generation of CRC code for one byte of data is completed in two machine cycles.

Figure 1.17.1 shows the block diagram of the CRC circuit. Figure 1.17.2 shows the CRC-related registers. Figure 1.17.3 shows the calculation example using the CRC calculation circuit

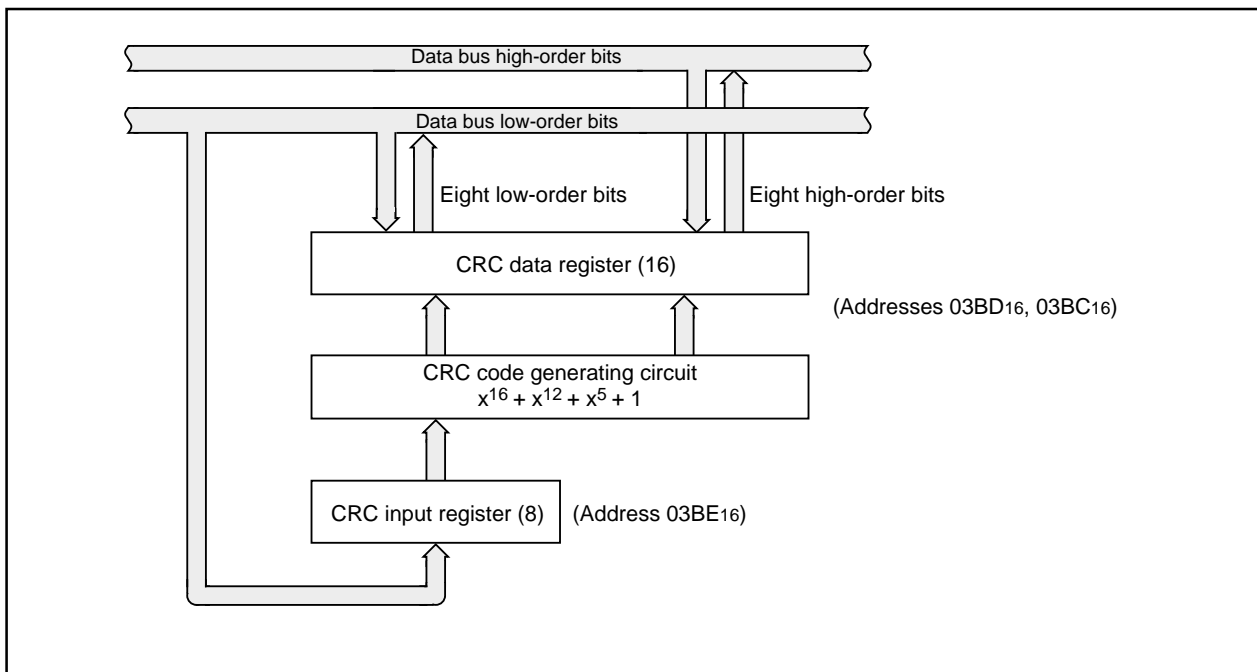


Figure 1.17.1. Block diagram of CRC circuit

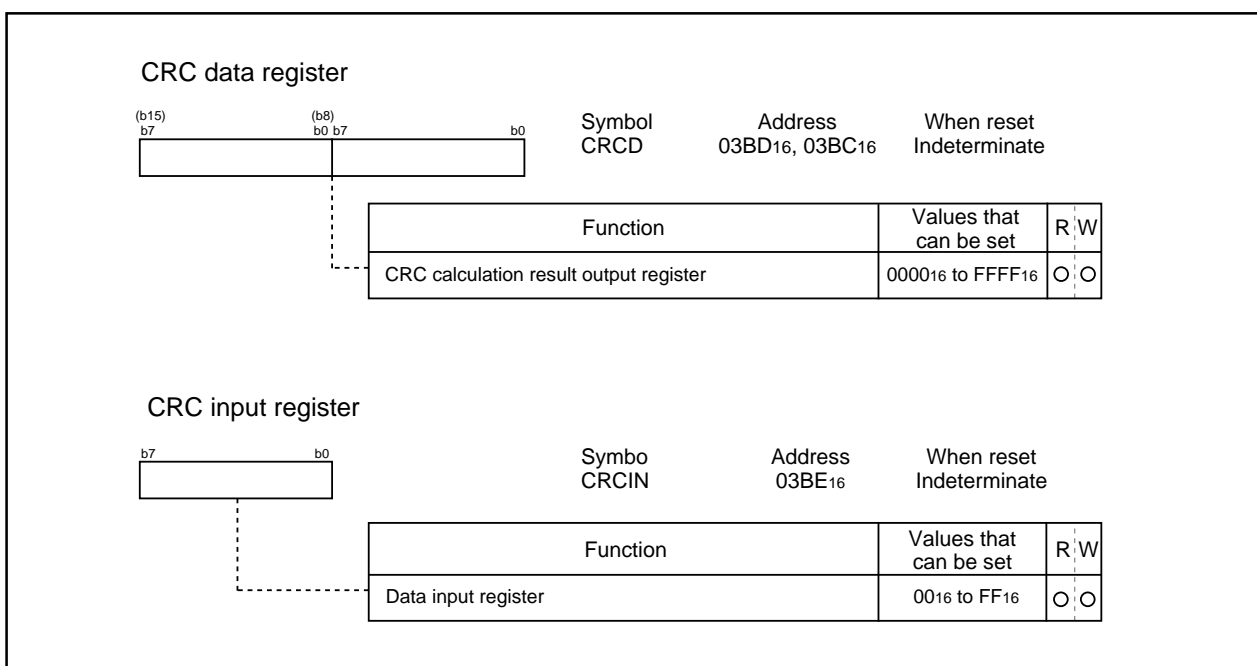


Figure 1.17.2. CRC-related registers

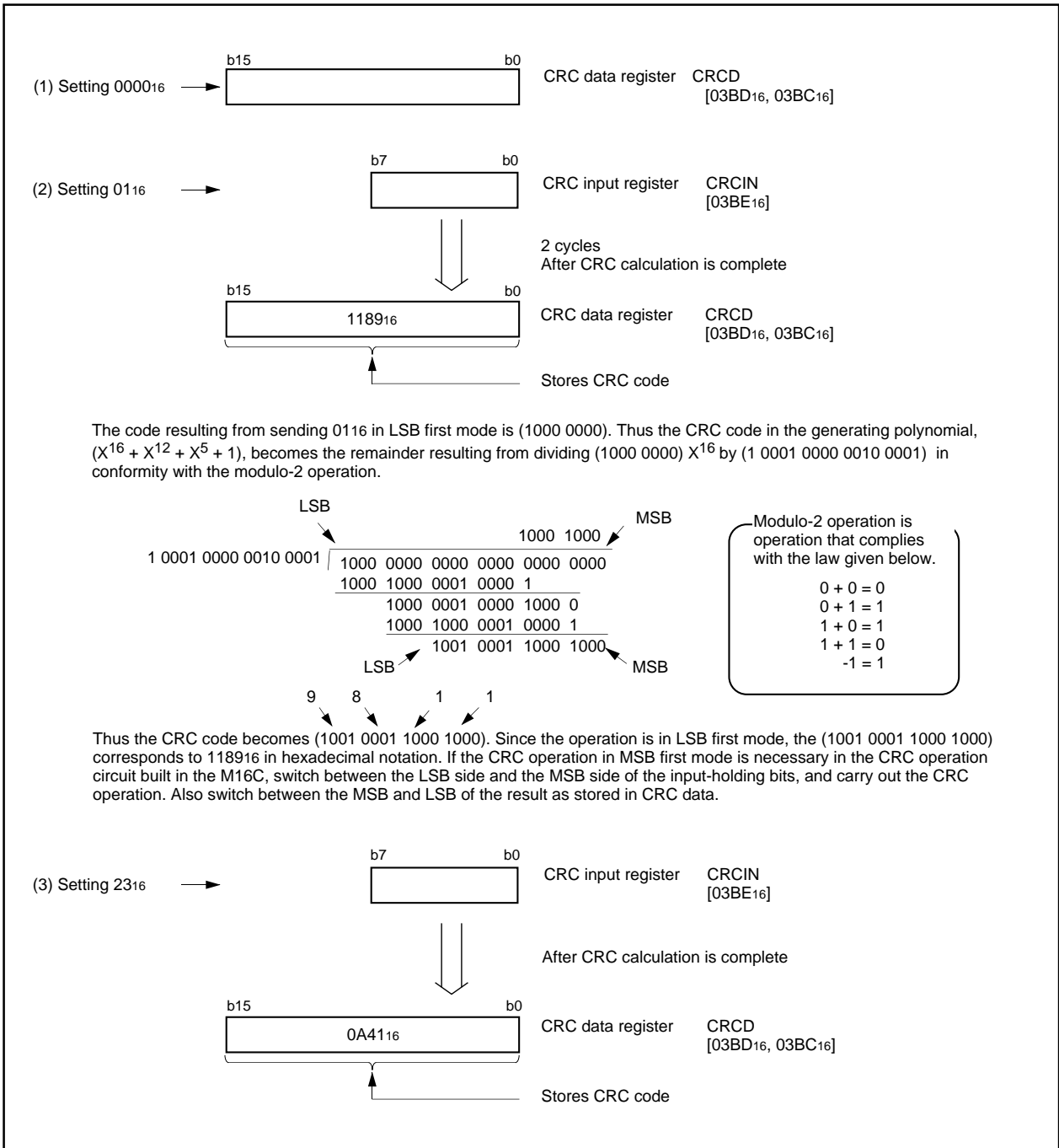


Figure 1.17.3. Calculation example using the CRC calculation circuit

## Programmable I/O Ports

The M16C/62A (80-pin version) group has 70 programmable input/output ports given below (except P85).

- P00–P07
- P20–P27
- P30–P37
- P40–P43
- P50–P57
- P60–P67
- P70, P71, P76, P77
- P80–P84, P86, P87 (P85 is input port)
- P90, P92–P97
- P100–P107

Note: P1, P44 to P47, P72 to P75, P91 are not connected to external pins.

Figures 1.18.1 to 1.18.4 show the programmable I/O ports. Figure 1.18.5 shows the I/O pins.

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 (other than the D-A converter), they function as outputs regardless of the contents of the direction registers. When pins are to be used as the outputs for the D-A converter, do not set the direction registers to output mode. See the descriptions of the respective functions for how to set up the built-in peripheral devices.

### (1) Direction registers

Figure 1.18.6 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.

Note: There is no direction register bit for P85.

### (2) Port registers

Figure 1.18.7 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 1.18.8 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.

Programmable I/O Port

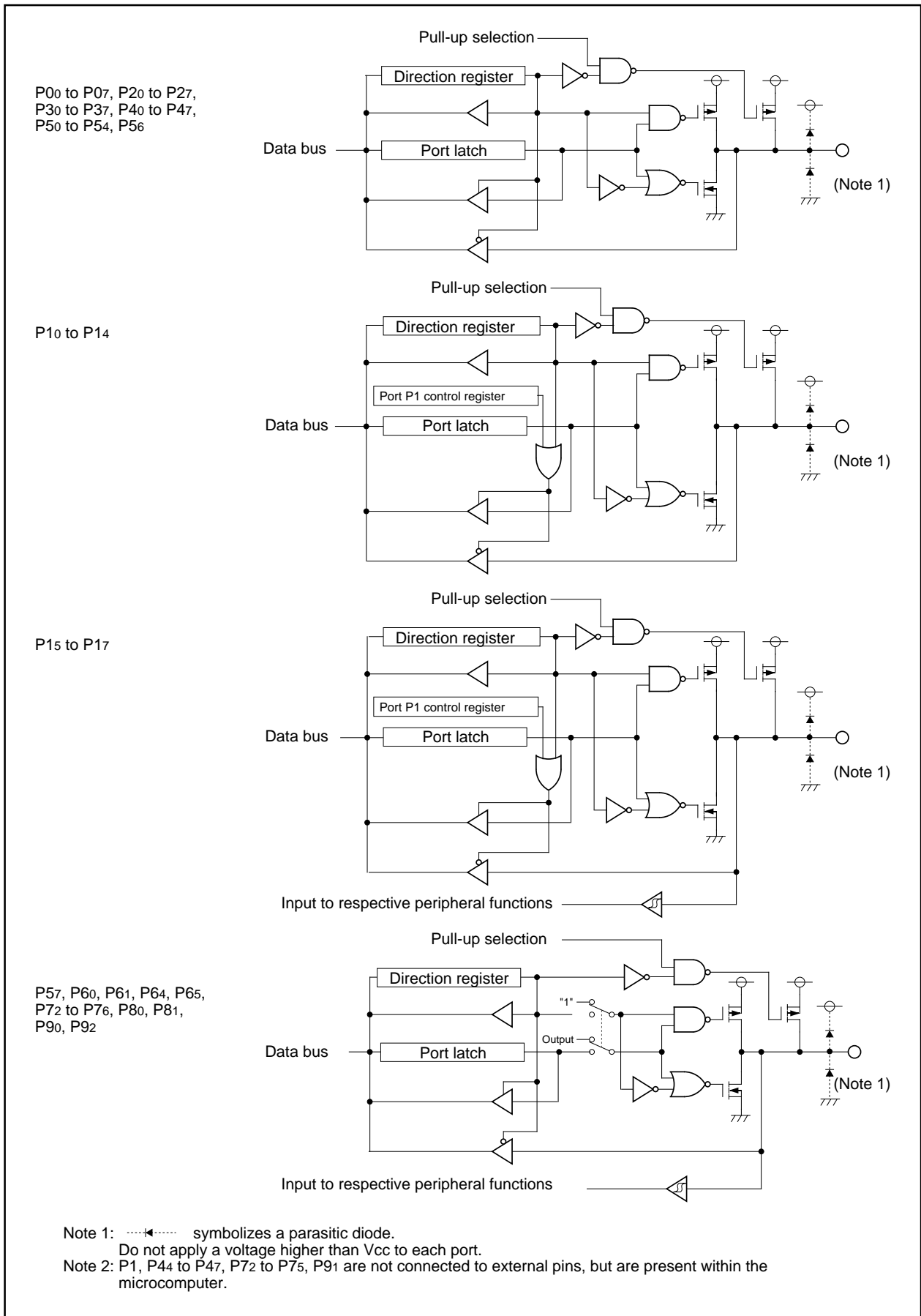


Figure 1.18.1. Programmable I/O ports (1)

Programmable I/O Port

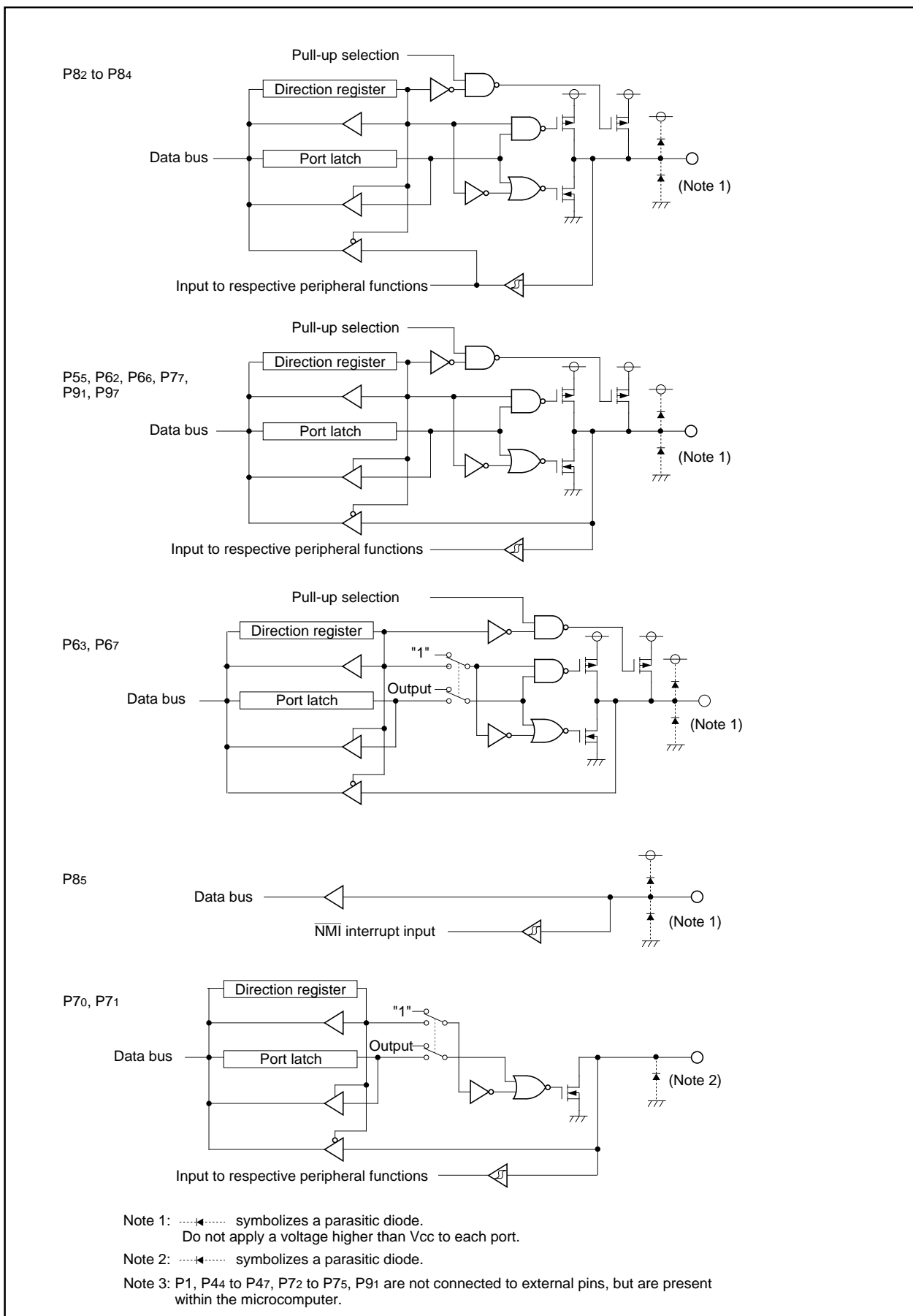


Figure 1.18.2. Programmable I/O ports (2)



Programmable I/O Port

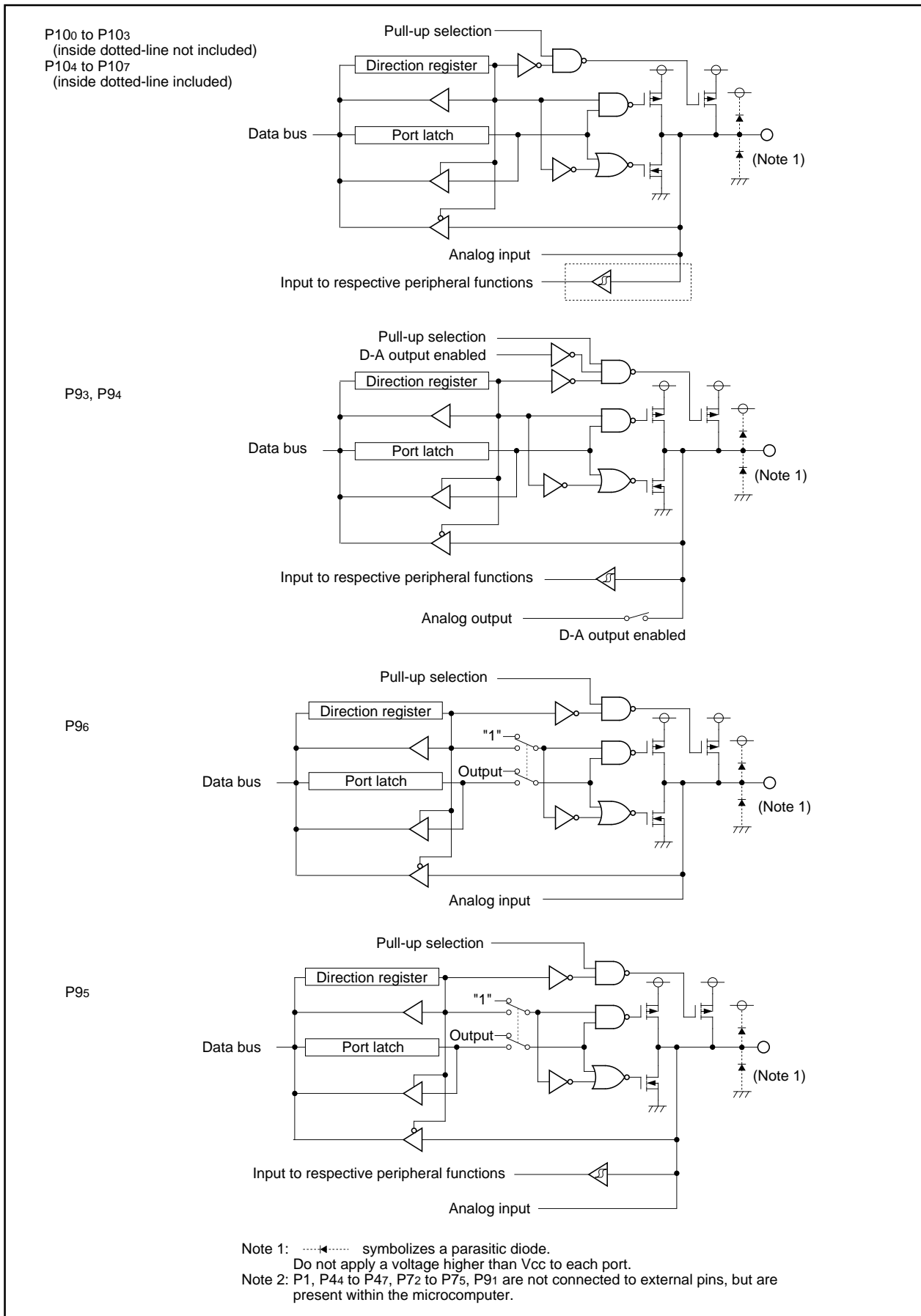


Figure 1.18.3. Programmable I/O ports (3)

Programmable I/O Port

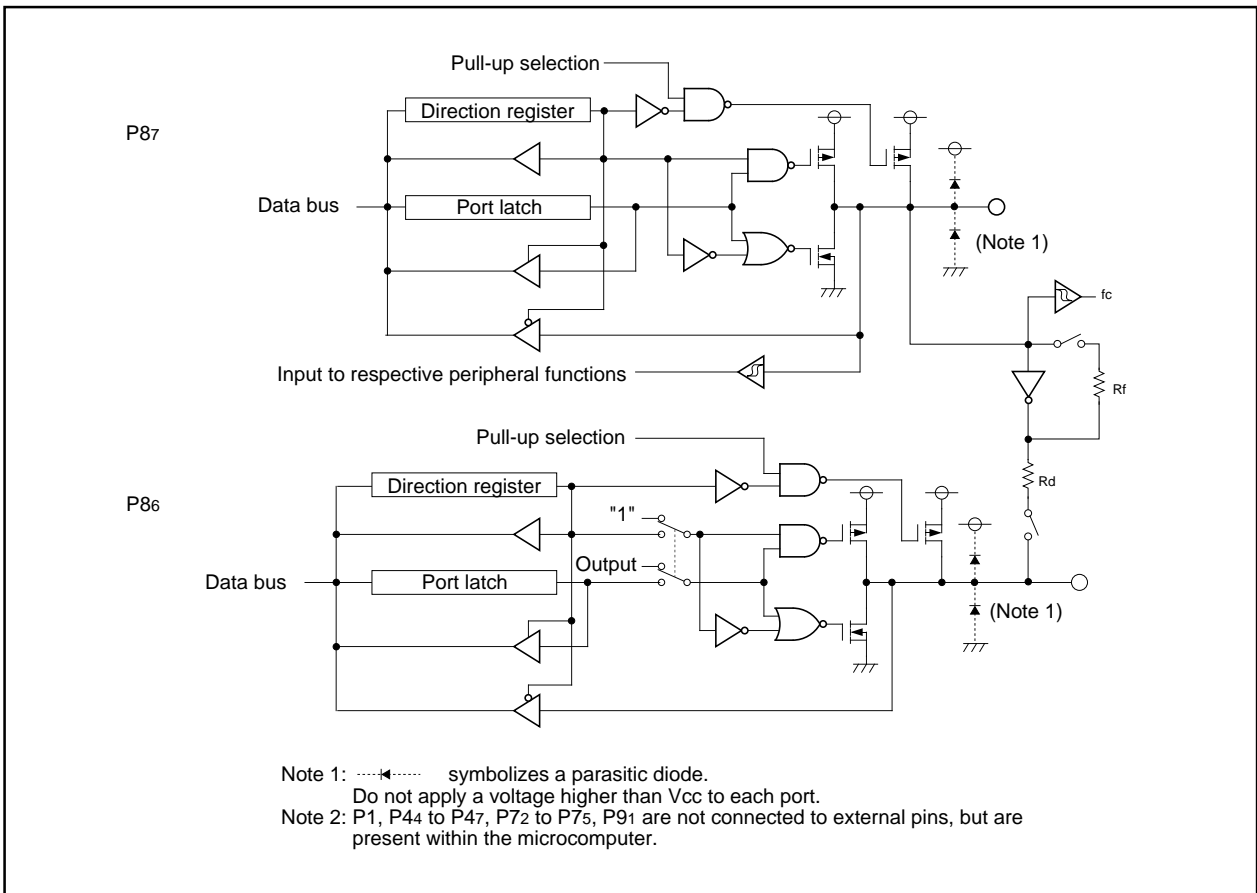


Figure 1.18.4. Programmable I/O ports (4)

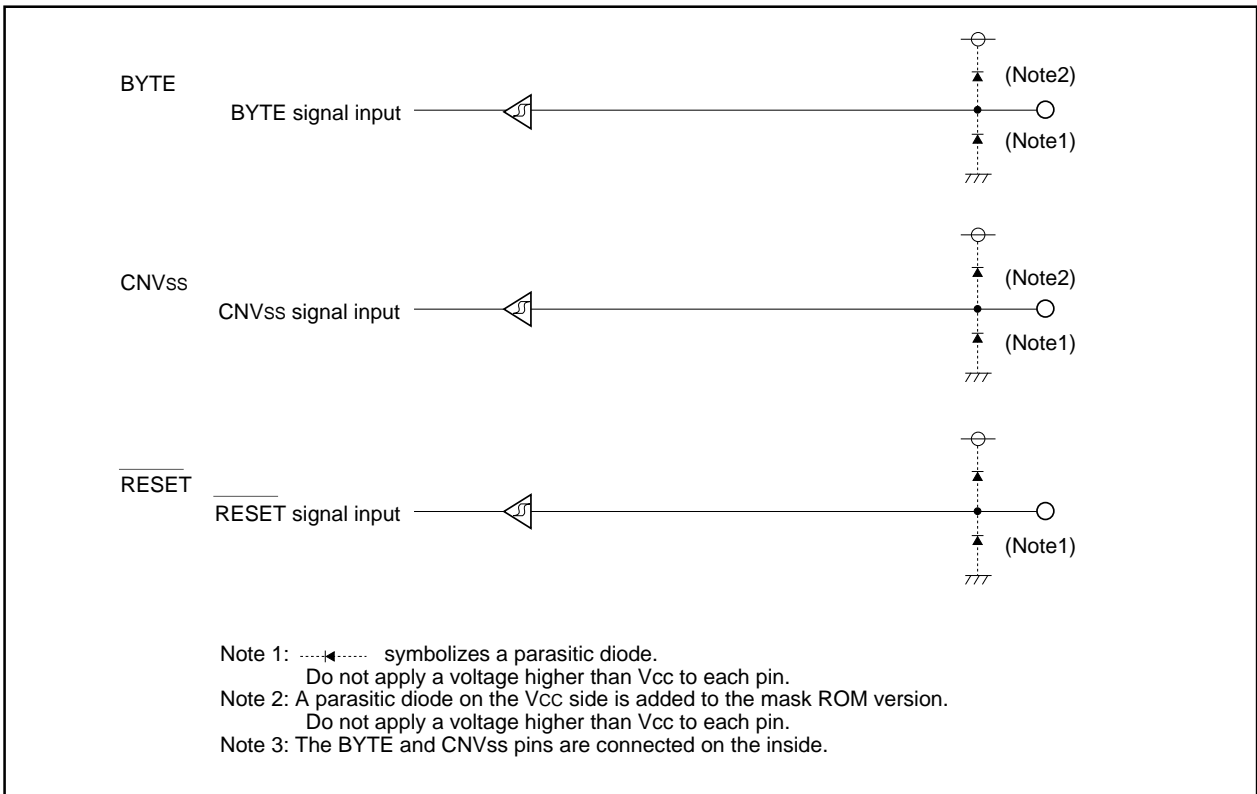


Figure 1.18.5. I/O pins

Programmable I/O Port

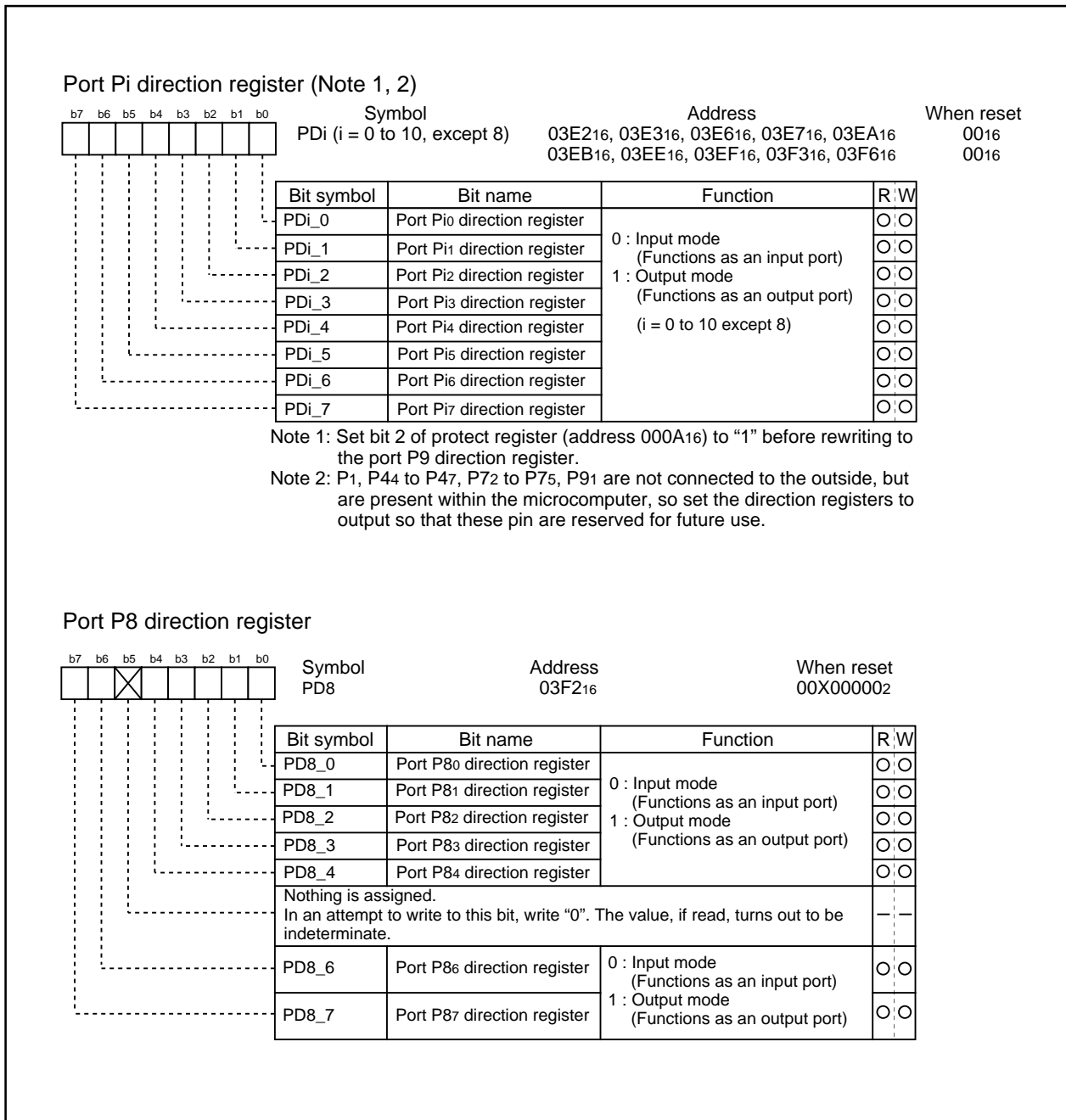


Figure 1.18.6. Direction register

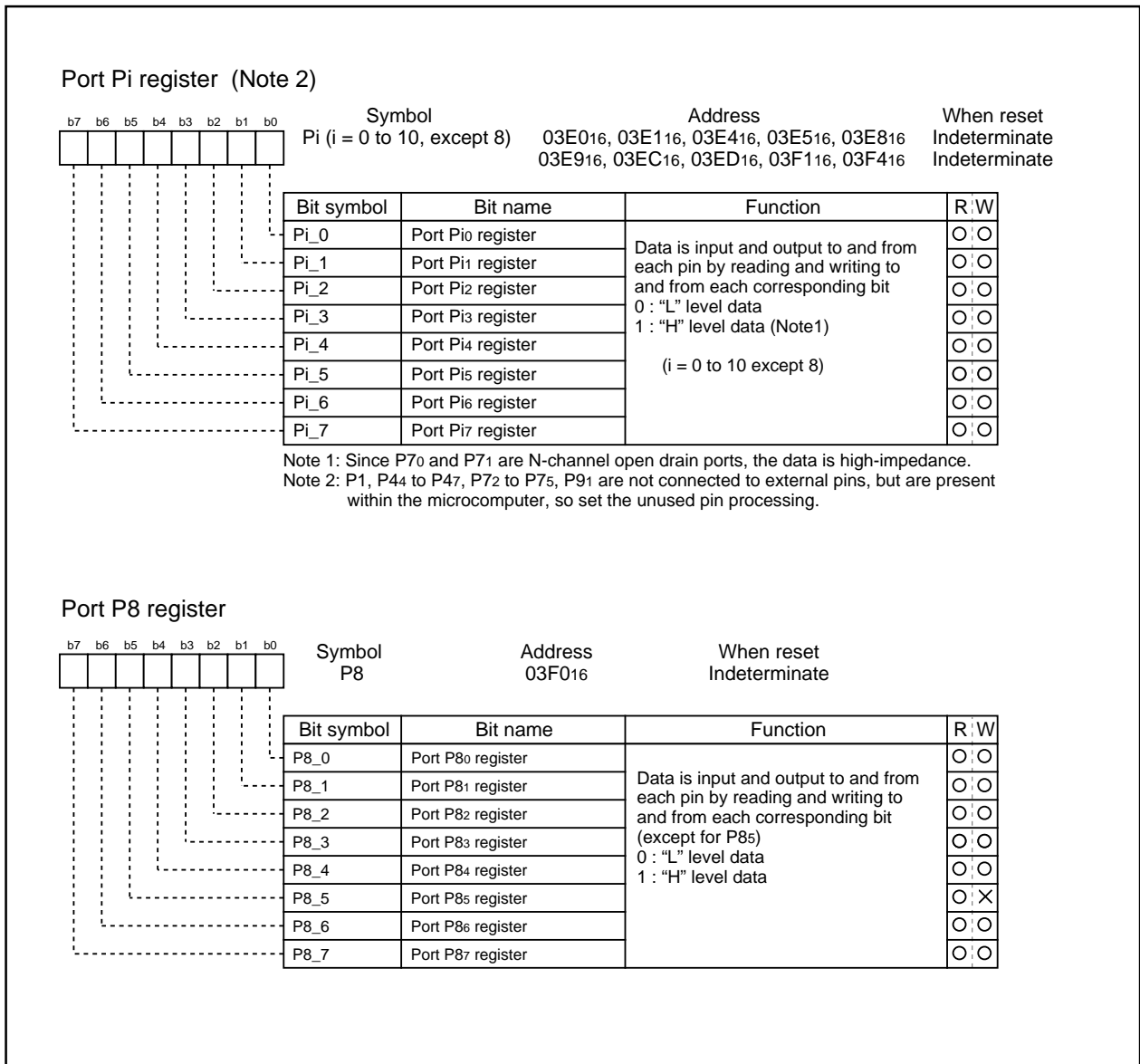
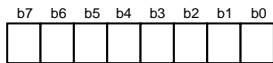


Figure 1.18.7. Port register

Programmable I/O Port

Pull-up control register 0 (Note)



Symbol  
PUR0

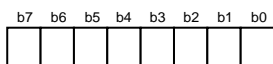
Address  
03FC<sub>16</sub>

When reset  
00<sub>16</sub>

Bit symbol	Bit name	Function	R/W
PU00	P00 to P03 pull-up	The corresponding port is pulled high with a pull-up resistor 0 : Not pulled high 1 : Pulled high	○ ○
PU01	P04 to P07 pull-up		○ ○
PU02	P10 to P13 pull-up		○ ○
PU03	P14 to P17 pull-up		○ ○
PU04	P20 to P23 pull-up		○ ○
PU05	P24 to P27 pull-up		○ ○
PU06	P30 to P33 pull-up		○ ○
PU07	P34 to P37 pull-up		○ ○

Note: P1 is not connected to external pins, but are present within the microcomputer, so set the unused pin processing.

Pull-up control register 1 (Note 2)



Symbol  
PUR1

Address  
03FD<sub>16</sub>

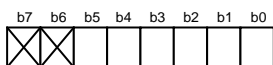
When reset  
00<sub>16</sub>

Bit symbol	Bit name	Function	R/W
PU10	P40 to P43 pull-up	The corresponding port is pulled high with a pull-up resistor 0 : Not pulled high 1 : Pulled high	○ ○
PU11	P44 to P47 pull-up		○ ○
PU12	P50 to P53 pull-up		○ ○
PU13	P54 to P57 pull-up		○ ○
PU14	P60 to P63 pull-up		○ ○
PU15	P64 to P67 pull-up		○ ○
PU16	P72 to P73 pull-up (Note 1)		○ ○
PU17	P74 to P77 pull-up		○ ○

Note 1: Since P70 and P71 are N-channel open drain ports, pull-up is not available for them.

Note 2: P44 to P47, P72 to P75 are not connected to external pins, but are present within the microcomputer, so set the unused pin processing.

Pull-up control register 2 (Note)



Symbol  
PUR2

Address  
03FE<sub>16</sub>

When reset  
00<sub>16</sub>

Bit symbol	Bit name	Function	R/W
PU20	P80 to P83 pull-up	The corresponding port is pulled high with a pull-up resistor 0 : Not pulled high 1 : Pulled high	○ ○
PU21	P84 to P87 pull-up (Except P85)		○ ○
PU22	P90 to P93 pull-up		○ ○
PU23	P94 to P97 pull-up		○ ○
PU24	P100 to P103 pull-up		○ ○
PU25	P104 to P107 pull-up		○ ○
Nothing is assigned. In an attempt to write to these bits, write "0". The value, if read, turns out to be "0".			— —

Note: P91 is not connected to external pins, but are present within the microcomputer, so set the unused pin processing.

Figure 1.18.8. Pull-up control register

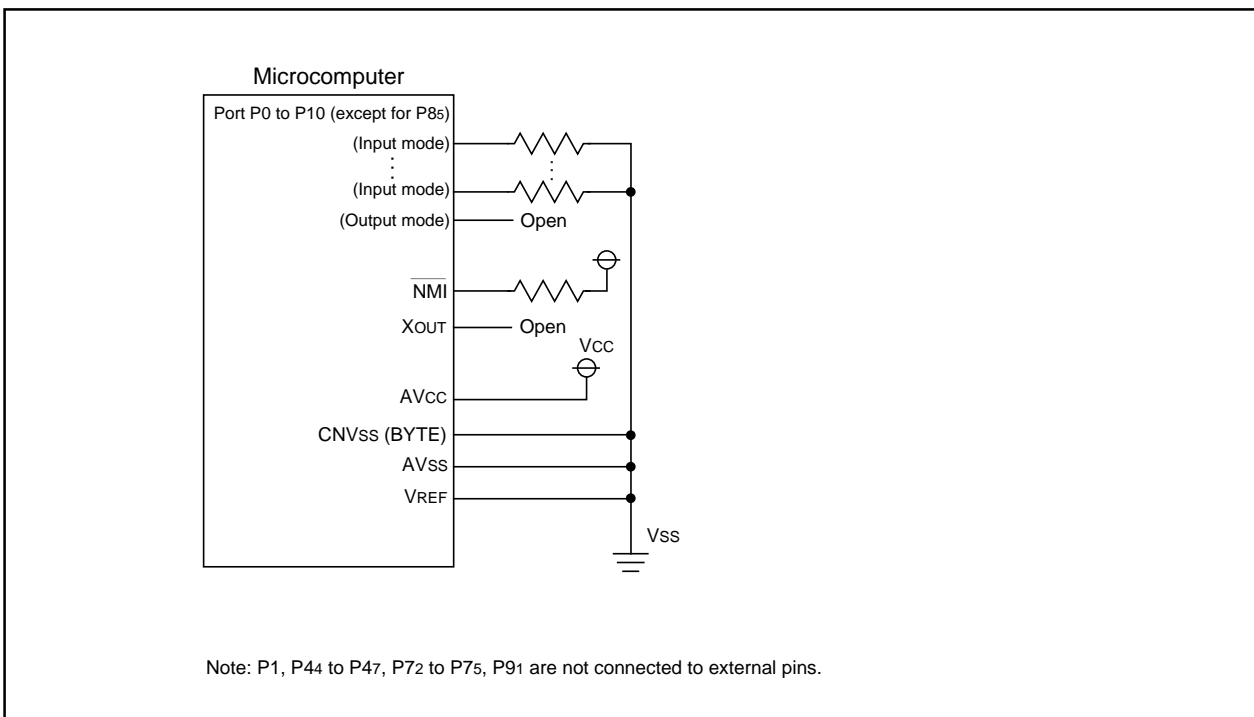
Programmable I/O Port

**Table 1.18.1. Example connection of unused pins in single-chip mode**

Pin name	Connection
Ports P0 to P10 (excluding P85) (Note 1)	After setting for input mode, connect every pin to Vss via a resistor; or after setting for output mode, leave these pins open.
XOUT (Note 2)	Open
$\overline{\text{NMI}}$	Connect via resistor to Vcc (pull-up)
AVcc	Connect to Vcc
AVss, VREF, BYTE	Connect to Vss

Note 1: P1, P44 to P47, P72 to P75, P91 are not connected to external pins, but are present within the microcomputer, so set the unused pin processing.

Note 2: With external clock input to XIN pin.



**Figure 1.18.9. Example connection of unused pins**

## Usage Precaution

### Timer A (timer mode)

- (1) Reading the timer Ai register while a count is in progress allows reading, with arbitrary timing, the value of the counter. Reading the timer Ai register with the reload timing gets "FFFF<sub>16</sub>". Reading the timer Ai register after setting a value in the timer Ai register with a count halted but before the counter starts counting gets a proper value.

### Timer A (event counter mode)

- (1) Reading the timer Ai register while a count is in progress allows reading, with arbitrary timing, the value of the counter. Reading the timer Ai register with the reload timing gets "FFFF<sub>16</sub>" by underflow or "0000<sub>16</sub>" by overflow. Reading the timer Ai register after setting a value in the timer Ai 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 TAIOUT pin outputs "L" level.
  - The interrupt request generated and the timer Ai interrupt request bit goes to "1".
- (2) The timer Ai 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 Ai interrupt (interrupt request bit), set timer Ai interrupt request bit to "0" after the above listed changes have been made.

### Timer A (pulse width modulation mode)

- (1) The timer Ai 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 Ai interrupt (interrupt request bit), set timer Ai 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 TAIOUT pin is outputting an "H" level in this instance, the output level goes to "L", and the timer Ai interrupt request bit goes to "1". If the TAIOUT pin is outputting an "L" level in this instance, the level does not change, and the timer Ai interrupt request bit does not becomes "1".

### 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 "FFFF<sub>16</sub>". 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.

### 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 switching to either wait mode or stop mode, instructions occupying four bytes either from the WAIT instruction or from the instruction that sets the all clock stop control bit to "1" within the instruction queue are prefetched and then the program stops. So put at least four NOPs in succession either to the WAIT instruction or to the instruction that sets the all clock stop control bit to "1".
- (3) When the MCU running in low-speed or low power dissipation mode, do not enter WAIT mode with WAIT peripheral function clock stop bit set to "1".

### Interrupts

- (1) Reading address 00000<sub>16</sub>
  - When maskable interrupt is occurred, CPU reads 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". Even if the address 00000<sub>16</sub> is read out by software, "0" is set to the enabled highest priority interrupt source request bit. Therefore interrupt can be canceled and unexpected interrupt can occur.  
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.  
When using the  $\overline{\text{NMI}}$  interrupt, initialize the stack point at the beginning of a program. Concerning the first instruction immediately after reset, generating any interrupts including the  $\overline{\text{NMI}}$  interrupt is prohibited.
- (3) The  $\overline{\text{NMI}}$  interrupt
  - The  $\overline{\text{NMI}}$  interrupt can not be disabled. Be sure to connect  $\overline{\text{NMI}}$  pin to Vcc via a pull-up resistor if unused.

Do not get either into stop mode with the  $\overline{\text{NMI}}$  pin set to "L".



## Usage precaution

### (4) External interrupt

- When the polarity of the  $\overline{\text{INT0}}$  to  $\overline{\text{INT2}}$  pins is changed, the interrupt request bit is sometimes set to "1". After changing the polarity, set the interrupt request bit to "0".

### (5) Rewrite the interrupt control register

- To rewrite the interrupt control register, do so at a point that does not generate the interrupt request for that register. If there is possibility of the interrupt request occur, rewrite the interrupt control register after the interrupt is disabled. 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 (four when using the HOLD function) 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.

- When a instruction to rewrite the interrupt control register is executed but the interrupt is disabled, the interrupt request bit is not set sometimes even if the interrupt request for that register has been generated. This will depend on the instruction. If this creates problems, use the below instructions to change the register.

Instructions : AND, OR, BCLR, BSET

## Noise

- (1) Insert bypass capacitor between Vcc and Vss pin for noise and latch up countermeasure.
  - Insert bypass capacitor (about 0.1  $\mu\text{F}$ ) and connect short and wide line between Vcc and Vss lines.

### **Items to be submitted when ordering masked ROM version**

Please submit the following when ordering masked ROM products:

- (1) Mask ROM confirmation form
- (2) Mask specification sheet
- (3) ROM data : Floppy disks

\*: 3.5-inch double-sided high-density disk (IBM format) is required per pattern.

## Electrical characteristics

### Electrical characteristics

**Table 1.20.1. Absolute maximum ratings**

Symbol	Parameter		Condition	Rated value	Unit
V <sub>cc</sub>	Supply voltage		V <sub>cc</sub> =AV <sub>cc</sub>	-0.3 to 6.5	V
AV <sub>cc</sub>	Analog supply voltage		V <sub>cc</sub> =AV <sub>cc</sub>	-0.3 to 6.5	V
V <sub>i</sub>	Input voltage	RESET, CNV <sub>ss</sub> (BYTE) P0 <sub>0</sub> to P0 <sub>7</sub> , P2 <sub>0</sub> to P2 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>7</sub> , P4 <sub>0</sub> to P4 <sub>3</sub> , P5 <sub>0</sub> to P5 <sub>7</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>6</sub> to P7 <sub>7</sub> , P8 <sub>0</sub> to P8 <sub>7</sub> , P9 <sub>0</sub> , P9 <sub>2</sub> to P9 <sub>7</sub> , P10 <sub>0</sub> to P10 <sub>7</sub> , VREF, X <sub>IN</sub>		-0.3 to V <sub>cc</sub> +0.3	V
		P7 <sub>0</sub> , P7 <sub>1</sub>		-0.3 to 6.5	V
V <sub>o</sub>	Output voltage	P0 <sub>0</sub> to P0 <sub>7</sub> , P2 <sub>0</sub> to P2 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>7</sub> , P4 <sub>0</sub> to P4 <sub>3</sub> , P5 <sub>0</sub> to P5 <sub>7</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>6</sub> to P7 <sub>7</sub> , P8 <sub>0</sub> to P8 <sub>4</sub> , P8 <sub>6</sub> , P8 <sub>7</sub> , P9 <sub>0</sub> , P9 <sub>2</sub> to P9 <sub>7</sub> , P10 <sub>0</sub> to P10 <sub>7</sub> , X <sub>OUT</sub>		-0.3 to V <sub>cc</sub> +0.3	V
		P7 <sub>0</sub> , P7 <sub>1</sub>		-0.3 to 6.5	V
P <sub>d</sub>	Power dissipation		T <sub>opr</sub> =25°C	300	mW
T <sub>opr</sub>	Operating ambient temperature			-20 to 85 / -40 to 85 (Note)	°C
T <sub>stg</sub>	Storage temperature			-65 to 150	°C

Note: Specify a product of -40 to 85°C to use it.

Electrical characteristics

**Table 1.20.2. Recommended operating conditions (referenced to Vcc = 2.7V to 5.5V at Topr = -20°C to 85°C / -40°C to 85°C (Note 3) unless otherwise specified)**

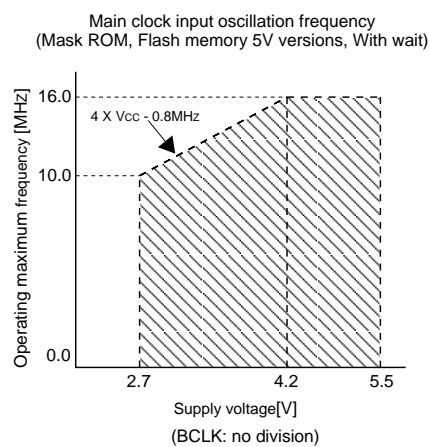
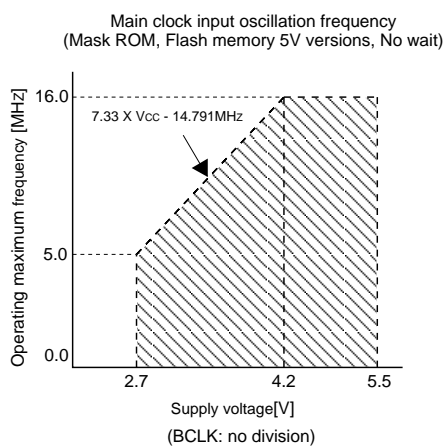
Symbol	Parameter		Standard			Unit	
			Min.	Typ.	Max.		
Vcc	Supply voltage		2.7	5.0	5.5	V	
AVcc	Analog supply voltage			Vcc		V	
Vss	Supply voltage			0		V	
AVss	Analog supply voltage			0		V	
VIH	HIGH input voltage	P00 to P07, P20 to P27, P30 to P37, P40 to P43, P50 to P57, P60 to P67, P76, P77, P80 to P87, P90, P92 to P97, P100 to P107, XIN, RESET, CNVss (BYTE)	0.8Vcc		Vcc	V	
		P70, P71	0.8Vcc		6.5	V	
VIL	LOW input voltage	P00 to P07, P20 to P27, P30 to P37, P40 to P43, P50 to P57, P60 to P67, P70, P71, P76, P77, P80 to P87, P90, P92 to P97, P100 to P107, XIN, RESET, CNVss (BYTE)	0		0.2Vcc	V	
IOH (peak)	HIGH peak output current	P00 to P07, P20 to P27, P30 to P37, P40 to P43, P50 to P57, P60 to P67, P76, P77, P80 to P84, P86, P87, P90, P92 to P97, P100 to P107			-10.0	mA	
IOH (avg)	HIGH average output current	P00 to P07, P20 to P27, P30 to P37, P40 to P43, P50 to P57, P60 to P67, P76, P77, P80 to P84, P86, P87, P90, P92 to P97, P100 to P107			-5.0	mA	
IOL (peak)	LOW peak output current	P00 to P07, P20 to P27, P30 to P37, P40 to P43, P50 to P57, P60 to P67, P70, P71, P76, P77, P80 to P84, P86, P87, P90, P92 to P97, P100 to P107			10.0	mA	
IOL (avg)	LOW average output current	P00 to P07, P20 to P27, P30 to P37, P40 to P43, P50 to P57, P60 to P67, P70, P71, P76, P77, P80 to P84, P86, P87, P90, P92 to P97, P100 to P107			5.0	mA	
f (XIN)	Main clock input oscillation frequency	No wait	Mask ROM, Flash memory 5V version (Note 5)	Vcc=4.2V to 5.5V	0	16	MHz
				Vcc=2.7V to 4.2V	0	$7.33 \times V_{cc} - 14.791$	MHz
		with wait	Mask ROM, Flash memory 5V version (Note 5)	Vcc=4.2V to 5.5V	0	16	MHz
				Vcc=2.7V to 4.2V	0	$4 \times V_{cc} - 0.8$	MHz
f (XCIN)	Subclock oscillation frequency			32.768	50	kHz	

Note 1: The mean output current is the mean value within 100ms.

Note 2: The total IOL (peak) for all ports must be 80mA max. The total IOH (peak) for all ports must be 80mA max.

Note 3: Specify a product of -40°C to 85°C to use it.

Note 4: Relationship between main clock oscillation frequency and supply voltage.



Note 5: Execute case without wait, program / erase of flash memory by Vcc=4.2V to 5.5V and f(BCLK) ≤ 6.25 MHz.  
 Execute case with wait, program / erase of flash memory by Vcc=4.2V to 5.5V and f(BCLK) ≤ 12.5 MHz.

Electrical characteristics

**Table 1.20.3. A-D conversion characteristics (referenced to  $V_{CC} = AV_{CC} = V_{REF} = 2.7V$  to  $5.5V$ ,  $V_{SS} = AV_{SS} = 0V$  at  $T_{opr} = -20^{\circ}C$  to  $85^{\circ}C$  /  $-40^{\circ}C$  to  $85^{\circ}C$  (Note 4) unless otherwise specified)**

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)	$V_{REF} = V_{CC} = 5V$	AN0 to AN7 input			$\pm 3$	LSB
				ANEX0, ANEX1 input, External op-amp connection mode			$\pm 7$	LSB
		Sample & hold function available(8bit)	$V_{REF} = V_{CC} = 5V$			$\pm 2$	LSB	
Sample & hold function not available(8bit)	$V_{REF} = V_{CC} = 3V, \emptyset AD=fAD/2$			$\pm 2$	LSB			
RLADDER	Ladder resistance		$V_{REF} = V_{CC}$	10		40	k $\Omega$	
tCONV	Conversion time(10bit), Sample & hold function available		$V_{REF} = V_{CC} = 5V, \emptyset AD=10MHz$	3.3			$\mu s$	
tCONV	Conversion time(8bit), Sample & hold function available		$V_{REF} = V_{CC} = 5V, \emptyset AD=10MHz$	2.8			$\mu s$	
tCONV	Conversion time(8bit), Sample & hold function not available		$V_{REF} = V_{CC} = 3V, \emptyset AD=fAD/2=5MHz$	9.8			$\mu s$	
tsAMP	Sampling time			0.3			$\mu s$	
VREF	Reference voltage			2.7		$V_{CC}$	V	
VIA	Analog input voltage			0		$V_{REF}$	V	

Note 1: Do f(XIN) in range of main clock input oscillation frequency prescribed with recommended operating conditions of table 1.20.2. Divide the f AD if f(XIN) exceeds 10MHz, and make AD operation clock frequency ( $\emptyset AD$ ) equal to or lower than 10MHz. And divide the f AD if  $V_{CC}$  is less than 4.2V, and make AD operation clock frequency ( $\emptyset AD$ ) equal to or lower than fAD/2.

Note 2: A case without sample & hold function turn AD operation clock frequency ( $\emptyset AD$ ) into 250 kHz or more in addition to a limit of Note 1.  
 A case with sample & hold function turn AD operation clock frequency ( $\emptyset AD$ ) into 1MHz or more in addition to a limit of Note 1.

Note 3: Connect AVCC pin to VCC pin and apply the same electric potential.

Note 4: Specify a product of  $-40^{\circ}C$  to  $85^{\circ}C$  to use it.

**Table 1.20.4. D-A conversion characteristics (referenced to  $V_{CC} = V_{REF} = 2.7V$  to  $5.5V$ ,  $V_{SS} = AV_{SS} = 0V$ , at  $T_{opr} = -20^{\circ}C$  to  $85^{\circ}C$  /  $-40^{\circ}C$  to  $85^{\circ}C$ (Note2) unless otherwise specified)**

Symbol	Parameter		Measuring condition	Standard			Unit
				Min.	Typ.	Max.	
-	Resolution					8	Bits
-	Absolute accuracy					1.0	%
t <sub>su</sub>	Setup time					3	$\mu s$
R <sub>o</sub>	Output resistance			4	10	20	k $\Omega$
I <sub>VREF</sub>	Reference power supply input current		(Note 1)			1.5	mA

Note 1: This applies when using one D-A converter, with the D-A register for the unused D-A converter set to "0016".

The A-D converter's ladder resistance is not included.

Also, when D-A register contents are not "00 16", the current I<sub>VREF</sub> always flows even though Vref may have been set to be unconnected by the A-D control register.

Note 2: Specify a product of  $-40^{\circ}C$  to  $85^{\circ}C$  to use it.

**Table 1.20.5. Flash memory version electrical characteristics**

(referenced to  $V_{CC} = 4.2V$  to  $5.5V$ , at  $T_{opr} = 0$  to  $60^{\circ}C$  unless otherwise specified)

Parameter	Standard			Unit
	Min.	Typ.	Max	
Page program time		6	120	ms
Block erase time		50	600	ms
Erase all unlocked blocks time		50 X n (Note)	600 X n (Note)	ms
Lock bit program time		6	120	ms

Note : n denotes the number of block erases.

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

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

**Table 1.20.6. Electrical characteristics (referenced to V<sub>CC</sub> = 4.2V to 5V, V<sub>SS</sub> = 0V at T<sub>opr</sub> = -20°C to 85°C / -40°C to 85°C (Note 2), f(X<sub>IN</sub>) = 16MHz unless otherwise specified)**

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> , P2 <sub>0</sub> to P2 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>7</sub> , P4 <sub>0</sub> to P4 <sub>3</sub> , P5 <sub>0</sub> to P5 <sub>7</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>6</sub> , P7 <sub>7</sub> , P8 <sub>0</sub> to P8 <sub>4</sub> , P8 <sub>6</sub> , P8 <sub>7</sub> , P9 <sub>0</sub> , P9 <sub>2</sub> to P9 <sub>7</sub> , P10 <sub>0</sub> to P10 <sub>7</sub>	I <sub>OH</sub> =-5mA	3.0			V	
V <sub>OH</sub>	HIGH output voltage	P0 <sub>0</sub> to P0 <sub>7</sub> , P2 <sub>0</sub> to P2 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>7</sub> , P4 <sub>0</sub> to P4 <sub>3</sub> , P5 <sub>0</sub> to P5 <sub>7</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>6</sub> , P7 <sub>7</sub> , P8 <sub>0</sub> to P8 <sub>4</sub> , P8 <sub>6</sub> , P8 <sub>7</sub> , P9 <sub>0</sub> , P9 <sub>2</sub> to P9 <sub>7</sub> , P10 <sub>0</sub> to P10 <sub>7</sub>	I <sub>OH</sub> =-200μA	4.7			V	
V <sub>OH</sub>	HIGH output voltage	X <sub>OUT</sub>	HIGHPOWER	I <sub>OH</sub> =-1mA	3.0		V	
			LOWPOWER	I <sub>OH</sub> =-0.5mA	3.0			
	HIGH output voltage	X <sub>COUT</sub>	HIGHPOWER	With no load applied		3.0	V	
			LOWPOWER	With no load applied		1.6		
V <sub>OL</sub>	LOW output voltage	P0 <sub>0</sub> to P0 <sub>7</sub> , P2 <sub>0</sub> to P2 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>7</sub> , P4 <sub>0</sub> to P4 <sub>3</sub> , P5 <sub>0</sub> to P5 <sub>7</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub> , P7 <sub>6</sub> , P7 <sub>7</sub> , P8 <sub>0</sub> to P8 <sub>4</sub> , P8 <sub>6</sub> , P8 <sub>7</sub> , P9 <sub>0</sub> , P9 <sub>2</sub> to P9 <sub>7</sub> , P10 <sub>0</sub> to P10 <sub>7</sub>	I <sub>OL</sub> =5mA			2.0	V	
V <sub>OL</sub>	LOW output voltage	P0 <sub>0</sub> to P0 <sub>7</sub> , P2 <sub>0</sub> to P2 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>7</sub> , P4 <sub>0</sub> to P4 <sub>3</sub> , P5 <sub>0</sub> to P5 <sub>7</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub> , P7 <sub>6</sub> , P7 <sub>7</sub> , P8 <sub>0</sub> to P8 <sub>4</sub> , P8 <sub>6</sub> , P8 <sub>7</sub> , P9 <sub>0</sub> , P9 <sub>2</sub> to P9 <sub>7</sub> , P10 <sub>0</sub> to P10 <sub>7</sub>	I <sub>OL</sub> =200μA			0.45	V	
V <sub>OL</sub>	LOW output voltage	X <sub>OUT</sub>	HIGHPOWER	I <sub>OL</sub> =1mA		2.0	V	
			LOWPOWER	I <sub>OL</sub> =0.5mA		2.0		
	LOW output voltage	X <sub>COUT</sub>	HIGHPOWER	With no load applied		0	V	
			LOWPOWER	With no load applied		0		
V <sub>T+</sub> -V <sub>T-</sub>	Hysteresis	TA0 <sub>IN</sub> , TA3 <sub>IN</sub> , TA4 <sub>IN</sub> , TB0 <sub>IN</sub> , TB2 <sub>IN</sub> to TB5 <sub>IN</sub> , INT <sub>0</sub> to INT <sub>2</sub> , ADTRG, CTS <sub>0</sub> , CTS <sub>1</sub> , CLK <sub>0</sub> , CLK <sub>1</sub> , CLK <sub>3</sub> , CLK <sub>4</sub> , TA3 <sub>OUT</sub> , TA4 <sub>OUT</sub> , NMI, K <sub>IO</sub> to K <sub>I3</sub> , SIN <sub>4</sub> , RxD <sub>0</sub> to RxD <sub>2</sub>		0.2		1.0	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> , P2 <sub>0</sub> to P2 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>7</sub> , P4 <sub>0</sub> to P4 <sub>3</sub> , P5 <sub>0</sub> to P5 <sub>7</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub> , P7 <sub>6</sub> , P7 <sub>7</sub> , P8 <sub>0</sub> to P8 <sub>7</sub> , P9 <sub>0</sub> , P9 <sub>2</sub> to P9 <sub>7</sub> , P10 <sub>0</sub> to P10 <sub>7</sub> , X <sub>IN</sub> , RESET, CNV <sub>SS</sub> (BYTE)	V <sub>I</sub> =5V			5.0	μA	
I <sub>IL</sub>	LOW input current	P0 <sub>0</sub> to P0 <sub>7</sub> , P2 <sub>0</sub> to P2 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>7</sub> , P4 <sub>0</sub> to P4 <sub>3</sub> , P5 <sub>0</sub> to P5 <sub>7</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>6</sub> , P7 <sub>7</sub> , P8 <sub>0</sub> to P8 <sub>7</sub> , P9 <sub>0</sub> , P9 <sub>2</sub> to P9 <sub>7</sub> , P10 <sub>0</sub> to P10 <sub>7</sub> , X <sub>IN</sub> , RESET, CNV <sub>SS</sub> (BYTE)	V <sub>I</sub> =0V			-5.0	μA	
R <sub>PULLUP</sub>	Pull-up resistance	P0 <sub>0</sub> to P0 <sub>7</sub> , P2 <sub>0</sub> to P2 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>7</sub> , P4 <sub>0</sub> to P4 <sub>3</sub> , P5 <sub>0</sub> to P5 <sub>7</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>6</sub> , P7 <sub>7</sub> , P8 <sub>0</sub> to P8 <sub>4</sub> , P8 <sub>6</sub> , P8 <sub>7</sub> , P9 <sub>0</sub> , P9 <sub>2</sub> to P9 <sub>7</sub> , P10 <sub>0</sub> to P10 <sub>7</sub>	V <sub>I</sub> =0V	30.0	50.0	167.0	kΩ	
R <sub>I<sub>XIN</sub></sub>	Feedback resistance	X <sub>IN</sub>			1.0		MΩ	
R <sub>I<sub>CXIN</sub></sub>	Feedback resistance	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	The output pins are open and other pins are V <sub>SS</sub>	Mask ROM version	f(X <sub>IN</sub> )=16MHz Square wave, no division		30.0	50.0	mA
			Flash memory 5V version	f(X <sub>IN</sub> )=16MHz Square wave, no division		32.5	50.0	
			Mask ROM version	f(X <sub>CIN</sub> )=32kHz Square wave		90.0		μA
			Flash memory 5V version	f(X <sub>CIN</sub> )=32kHz Square wave, in RAM		90.0		μA
			Flash memory 5V version	f(X <sub>CIN</sub> )=32kHz Square wave, in flash memory		2.2		mA
			Flash memory 5V version, Program	f(X <sub>IN</sub> )=16MHz Square wave, Division by 4		25		mA
			Flash memory 5V version, Erase	f(X <sub>IN</sub> )=16MHz Square wave, Division by 4		28		mA
				f(X <sub>CIN</sub> )=32kHz When a WAIT instruction is executed (Note 1)		4.0		μA
				T <sub>opr</sub> =25°C when clock is stopped			1.0	μA
	T <sub>opr</sub> =85°C when clock is stopped			20.0				

Note 1: With one timer operated using fc32.

Note 2: Specify a product of -40°C to 85°C to use it.

Timing ( $V_{CC} = 5V$ )

$V_{CC} = 5V$

**Timing requirements (referenced to  $V_{CC} = 5V$ ,  $V_{SS} = 0V$  at  $T_{opr} = -20^{\circ}C$  to  $85^{\circ}C$  /  $-40^{\circ}C$  to  $85^{\circ}C$  (\*))  
 unless otherwise specified)**

\* : Specify a product of  $-40^{\circ}C$  to  $85^{\circ}C$  to use it.

**Table 1.20.7. External clock input**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_c$	External clock input cycle time	62.5		ns
$t_{w(H)}$	External clock input HIGH pulse width	25		ns
$t_{w(L)}$	External clock input LOW pulse width	25		ns
$t_r$	External clock rise time		15	ns
$t_f$	External clock fall time		15	ns

Timing ( $V_{CC} = 5V$ )

$V_{CC} = 5V$

**Timing requirements (referenced to  $V_{CC} = 5V$ ,  $V_{SS} = 0V$  at  $T_{opr} = -20^{\circ}C$  to  $85^{\circ}C$  /  $-40^{\circ}C$  to  $85^{\circ}C$  (\*))  
 unless otherwise specified)**

\* : Specify a product of  $-40^{\circ}C$  to  $85^{\circ}C$  to use it.

**Table 1.20.8. Timer A input (counter input in event counter mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TA)}$	TAiIN input cycle time	100		ns
$t_{w(TAH)}$	TAiIN input HIGH pulse width	40		ns
$t_{w(TAL)}$	TAiIN input LOW pulse width	40		ns

**Table 1.20.9. Timer A input (gating input in timer mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TA)}$	TAiIN input cycle time	400		ns
$t_{w(TAH)}$	TAiIN input HIGH pulse width	200		ns
$t_{w(TAL)}$	TAiIN input LOW pulse width	200		ns

**Table 1.20.10. Timer A input (external trigger input in one-shot timer mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TA)}$	TAiIN input cycle time	200		ns
$t_{w(TAH)}$	TAiIN input HIGH pulse width	100		ns
$t_{w(TAL)}$	TAiIN input LOW pulse width	100		ns

**Table 1.20.11. Timer A input (external trigger input in pulse width modulation mode)**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{w(TAH)}$	TAiIN input HIGH pulse width	100		ns
$t_{w(TAL)}$	TAiIN input LOW pulse width	100		ns

**Table 1.20.12. Timer A input (up/down input in event counter mode)**

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



$V_{CC} = 5V$

**Timing requirements (referenced to  $V_{CC} = 5V$ ,  $V_{SS} = 0V$  at  $T_{opr} = -20^{\circ}C$  to  $85^{\circ}C$  /  $-40^{\circ}C$  to  $85^{\circ}C$  (\*))  
 unless otherwise specified)**

\* : Specify a product of  $-40^{\circ}C$  to  $85^{\circ}C$  to use it.

**Table 1.20.13. 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 1.20.14. 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 1.20.15. 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 1.20.16. A-D trigger input**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(AD)}$	$\overline{ADTRG}$ input cycle time (trigger able minimum)	1000		ns
$t_{w(ADL)}$	$\overline{ADTRG}$ input LOW pulse width	125		ns

**Table 1.20.17. Serial I/O**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(CK)}$	CLKi input cycle time	200		ns
$t_{w(CKH)}$	CLKi input HIGH pulse width	100		ns
$t_{w(CKL)}$	CLKi 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 1.20.18. External interrupt  $\overline{INTi}$  inputs**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{w(INH)}$	$\overline{INTi}$ input HIGH pulse width	250		ns
$t_{w(INL)}$	$\overline{INTi}$ input LOW pulse width	250		ns

Timing ( $V_{CC} = 5V$ )

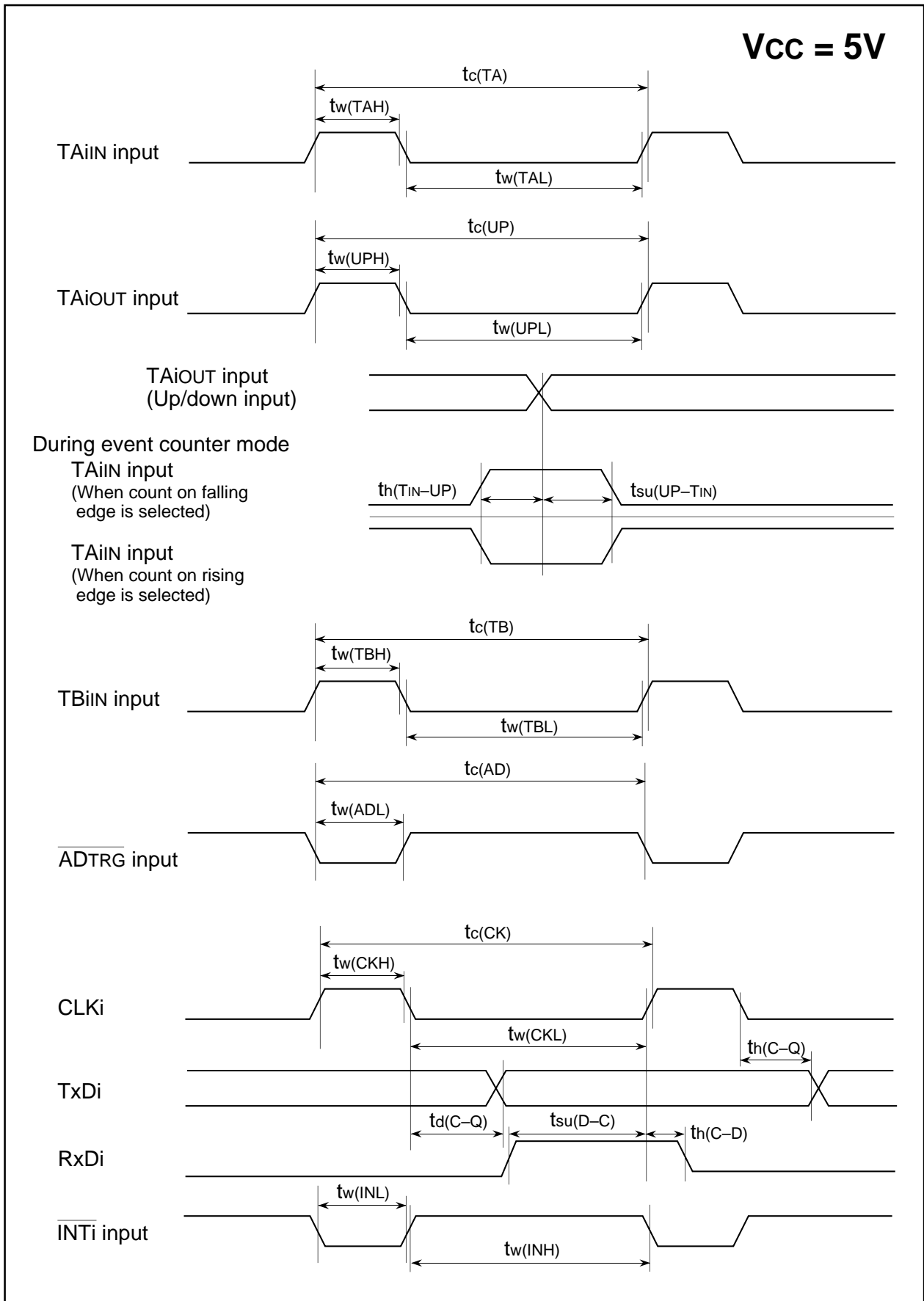


Figure 1.20.1.  $V_{CC}=5V$  timing diagram

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

V<sub>CC</sub> = 3V

**Table 1.20.19. Electrical characteristics**

(referenced to V<sub>CC</sub> = 2.7V to 3.3V, V<sub>SS</sub> = 0V at T<sub>opr</sub> = -20°C to 85°C / -40°C to 85°C (Note 1), f(X<sub>IN</sub>) = 10MHz (Note 2) with wait)

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> , P2 <sub>0</sub> to P2 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>7</sub> , P4 <sub>0</sub> to P4 <sub>3</sub> , P5 <sub>0</sub> to P5 <sub>7</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>6</sub> , P7 <sub>7</sub> , P8 <sub>0</sub> to P8 <sub>4</sub> , P8 <sub>6</sub> , P8 <sub>7</sub> , P9 <sub>0</sub> , P9 <sub>2</sub> to P9 <sub>7</sub> , P10 <sub>0</sub> to P10 <sub>7</sub>	I <sub>OH</sub> =-1mA	2.5			V
V <sub>OH</sub>	HIGH output voltage	X <sub>OUT</sub>	HIGHPOWER	I <sub>OH</sub> =-0.1mA	2.5		V
			LOWPOWER	I <sub>OH</sub> =-50μA	2.5		
	HIGH output voltage	X <sub>COU</sub> T	HIGHPOWER	With no load applied		3.0	V
			LOWPOWER	With no load applied		1.6	
V <sub>OL</sub>	LOW output voltage	P0 <sub>0</sub> to P0 <sub>7</sub> , P2 <sub>0</sub> to P2 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>7</sub> , P4 <sub>0</sub> to P4 <sub>3</sub> , P5 <sub>0</sub> to P5 <sub>7</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub> , P7 <sub>6</sub> , P7 <sub>7</sub> , P8 <sub>0</sub> to P8 <sub>4</sub> , P8 <sub>6</sub> , P8 <sub>7</sub> , P9 <sub>0</sub> , P9 <sub>2</sub> to P9 <sub>7</sub> , P10 <sub>0</sub> to P10 <sub>7</sub>	I <sub>OL</sub> =1mA			0.5	V
V <sub>OL</sub>	LOW output voltage	X <sub>OUT</sub>	HIGHPOWER	I <sub>OL</sub> =0.1mA		0.5	V
			LOWPOWER	I <sub>OL</sub> =50μA		0.5	
V <sub>T+</sub> -V <sub>T-</sub>	Hysteresis	TA0 <sub>IN</sub> , TA3 <sub>IN</sub> , TA4 <sub>IN</sub> , TB0 <sub>IN</sub> , TB2 <sub>IN</sub> to TB5 <sub>IN</sub> , INT <sub>0</sub> to INT <sub>2</sub> , ADTRG, CTS <sub>0</sub> , CTS <sub>1</sub> , CLK <sub>0</sub> , CLK <sub>1</sub> , CLK <sub>3</sub> , CLK <sub>4</sub> , TA3 <sub>OUT</sub> , TA4 <sub>OUT</sub> , NMI, K <sub>10</sub> to K <sub>13</sub> , SIN <sub>4</sub> , RxD <sub>0</sub> to RxD <sub>2</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> , P2 <sub>0</sub> to P2 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>7</sub> , P4 <sub>0</sub> to P4 <sub>3</sub> , P5 <sub>0</sub> to P5 <sub>7</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub> , P7 <sub>6</sub> , P7 <sub>7</sub> , P8 <sub>0</sub> to P8 <sub>7</sub> , P9 <sub>0</sub> , P9 <sub>2</sub> to P9 <sub>7</sub> , P10 <sub>0</sub> to P10 <sub>7</sub> , X <sub>IN</sub> , RESET, CNV <sub>SS</sub> (BYTE)	V <sub>I</sub> =3V			4.0	μA
I <sub>IL</sub>	LOW input current	P0 <sub>0</sub> to P0 <sub>7</sub> , P2 <sub>0</sub> to P2 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>7</sub> , P4 <sub>0</sub> to P4 <sub>3</sub> , P5 <sub>0</sub> to P5 <sub>7</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>0</sub> , P7 <sub>1</sub> , P7 <sub>6</sub> , P7 <sub>7</sub> , P8 <sub>0</sub> to P8 <sub>7</sub> , P9 <sub>0</sub> , P9 <sub>2</sub> to P9 <sub>7</sub> , P10 <sub>0</sub> to P10 <sub>7</sub> , X <sub>IN</sub> , RESET, CNV <sub>SS</sub> (BYTE)	V <sub>I</sub> =0V			-4.0	μA
R <sub>PULLUP</sub>	Pull-up resistance	P0 <sub>0</sub> to P0 <sub>7</sub> , P2 <sub>0</sub> to P2 <sub>7</sub> , P3 <sub>0</sub> to P3 <sub>7</sub> , P4 <sub>0</sub> to P4 <sub>3</sub> , P5 <sub>0</sub> to P5 <sub>7</sub> , P6 <sub>0</sub> to P6 <sub>7</sub> , P7 <sub>6</sub> , P7 <sub>7</sub> , P8 <sub>0</sub> to P8 <sub>4</sub> , P8 <sub>6</sub> , P8 <sub>7</sub> , P9 <sub>0</sub> , P9 <sub>2</sub> to P9 <sub>7</sub> , P10 <sub>0</sub> to P10 <sub>7</sub>	V <sub>I</sub> =0V	66.0	120.0	500.0	kΩ
R <sub>IXIN</sub>	Feedback resistance	X <sub>IN</sub>			3.0		MΩ
R <sub>ICXIN</sub>	Feedback resistance	X <sub>CIN</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	The output pins are open and other pins are V <sub>SS</sub>	Mask ROM version	f(X <sub>IN</sub> )=10MHz Square wave, no division	8.5	21.25	mA
			Flash memory 5V version	f(X <sub>IN</sub> )=10MHz Square wave, no division	12.0	21.25	
			Mask ROM version	f(X <sub>CIN</sub> )=32kHz Square wave	40.0		μA
			Flash memory 5V version	f(X <sub>CIN</sub> )=32kHz Square wave, in RAM	40.0		μA
			Flash memory 5V version	f(X <sub>CIN</sub> )=32kHz Square wave, in flash memory	800		μA
				f(X <sub>CIN</sub> )=32kHz When a WAIT instruction is executed. Oscillation capacity High (Note3)	2.8		μA
				f(X <sub>CIN</sub> )=32kHz When a WAIT instruction is executed. Oscillation capacity Low (Note3)	0.9		μA
	T <sub>opr</sub> =25°C when clock is stopped			1.0	μA		
	T <sub>opr</sub> =85°C when clock is stopped			20.0			

Note 1: Specify a product of -40°C to 85°C to use it.

Note 2: Mask ROM version and flash memory 5V version.

Note 3: With one timer operated using fc32.

Timing ( $V_{CC} = 3V$ )

$V_{CC} = 3V$

**Timing requirements**

(referenced to  $V_{CC} = 3V$ ,  $V_{SS} = 0V$  at  $T_{opr} = -20^{\circ}C$  to  $85^{\circ}C$  /  $-40^{\circ}C$  to  $85^{\circ}C$  (\*) unless otherwise specified)

\* : Specify a product of  $-40^{\circ}C$  to  $85^{\circ}C$  to use it.

**Table 1.20.20. 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		18	ns
$t_f$	External clock fall time		18	ns

Timing ( $V_{CC} = 3V$ )

$V_{CC} = 3V$

**Timing requirements**

(referenced to  $V_{CC} = 3V$ ,  $V_{SS} = 0V$  at  $T_{opr} = -20^{\circ}C$  to  $85^{\circ}C$  /  $-40^{\circ}C$  to  $85^{\circ}C$  (\*) unless otherwise specified)

\* : Specify a product of  $-40^{\circ}C$  to  $85^{\circ}C$  to use it.

**Table 1.20.21. Timer A input (counter input in event counter mode)**

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

**Table 1.20.22. Timer A input (gating input in timer mode)**

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

**Table 1.20.23. Timer A input (external trigger input in one-shot timer mode)**

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

**Table 1.20.24. Timer A input (external trigger input in pulse width modulation mode)**

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

**Table 1.20.25. Timer A input (up/down input in event counter mode)**

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

Timing ( $V_{CC} = 3V$ )

$V_{CC} = 3V$

**Timing requirements**

(referenced to  $V_{CC} = 3V$ ,  $V_{SS} = 0V$  at  $T_{opr} = -20^{\circ}C$  to  $85^{\circ}C$  /  $-40^{\circ}C$  to  $85^{\circ}C$  (\*) unless otherwise specified)

\* : Specify a product of  $-40^{\circ}C$  to  $85^{\circ}C$  to use it.

**Table 1.20.26. 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 1.20.27. 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 1.20.28. 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 1.20.29. A-D trigger input**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(AD)}$	$\overline{ADTRG}$ input cycle time (trigger able minimum)	1500		ns
$t_{w(ADL)}$	$\overline{ADTRG}$ input LOW pulse width	200		ns

**Table 1.20.30. Serial I/O**

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(CK)}$	CLKi input cycle time	300		ns
$t_{w(CKH)}$	CLKi input HIGH pulse width	150		ns
$t_{w(CKL)}$	CLKi 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 1.20.31. External interrupt  $\overline{INTi}$  inputs**

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

Timing ( $V_{CC} = 3V$ )

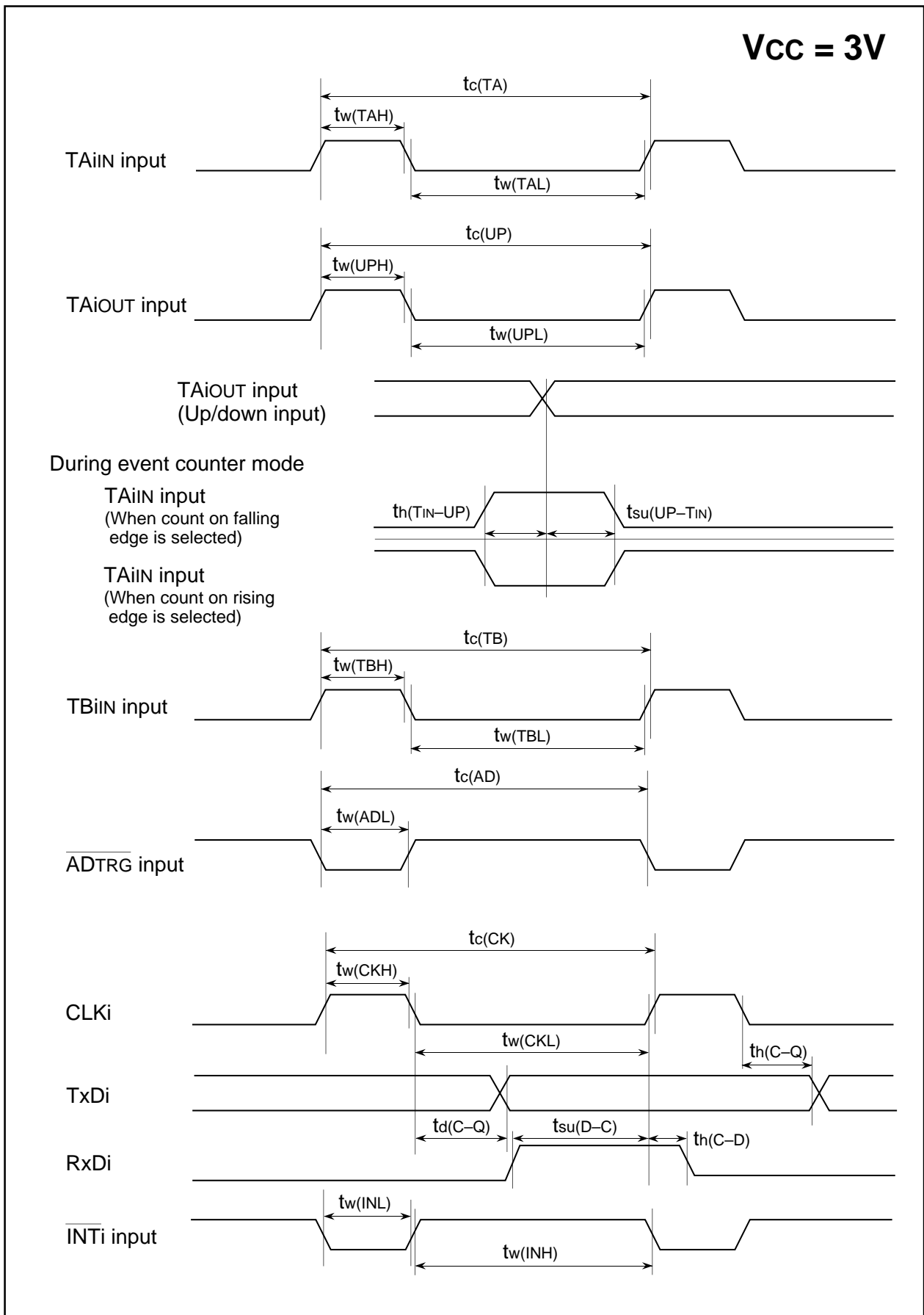


Figure 1.20.2.  $V_{CC}=3V$  timing diagram

GZZ-SH13-56B<98A0>

**MITSUBISHI ELECTRIC-CHIP 16-BIT  
 MICROCOMPUTER M30621M8A-XXXGP  
 MASK ROM CONFIRMATION FORM**

Mask ROM number	
-----------------	--

Receipt	Date :	
	Section head signature	Supervisor signature

Note : Please complete all items marked \* .

* Customer	Company name	TEL (      )	Issuance signature	Submitted by	Supervisor
	Date issued	Date :			

\*1. Check sheet

Mitsubishi processes the mask files generated by the mask file generation utilities out of those held on the floppy disks you give in to us, and forms them into masks. Hence, we assume liability provided that there is any discrepancy between the contents of these mask files and the ROM data to be burned into products we produce. Check thoroughly the contents of the mask files you give in.

Prepare 3.5 inches 2HD (IBM format) floppy disks. And store only one mask file in a floppy disk.

Microcomputer type No. :       M30621M8A-XXXGP

File code :      

--	--	--	--	--	--	--	--

 (hex)

Mask file name :      

--	--	--	--	--	--	--	--

 .MSK (alpha-numeric 8-digit)

\*2. Mark specification

The mark specification differs according to the type of package. After entering the mark specification on the separate mark specification sheet (for each package), attach that sheet to this masking check sheet for submission to Mitsubishi.

For the M30621M8A-XXXGP, submit the 80P6S mark specification sheet.

\*3. Usage Conditions

For our reference when of testing our products, please reply to the following questions about the usage of the products you ordered.

(1) Which kind of XIN-XOUT oscillation circuit is used?

- |   |   |
|---|---|
| <input type="checkbox"/> Ceramic resonator    | <input type="checkbox"/> Quartz-crystal oscillator      |
| <input type="checkbox"/> External clock input | <input type="checkbox"/> Other (                      ) |

What frequency do not use?

f(XIN) =  MHz



GZZ-SH13-56B<98A0>

Mask ROM number	
-----------------	--

**mitsubishi electric-chip 16-bit  
 MICROCOMPUTER M30621M8A-XXXGP  
 MASK ROM CONFIRMATION FORM**

(2) Which kind of XCIN-XCOUT oscillation circuit is used?

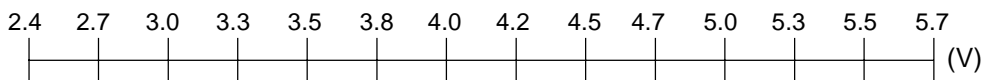
- Ceramic resonator                       Quartz-crystal oscillator  
 External clock input                       Other (                      )

What frequency do not use?

$f(XCIN) =$   kHz

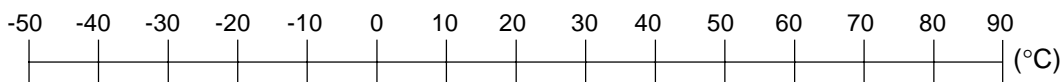
(3) Which operating supply voltage do you use?

(Circle the operating voltage range of use)



(4) Which operating ambient temperature do you use?

(Circle the operating temperature range of use)



(5) Do you use I<sup>2</sup>C (Inter IC) bus function?

- Not use                                       Use

(6) Do you use IE (Inter Equipment) bus function?

- Not use                                       Use

Thank you cooperation.

\*4. Special item (Indicate none if there is not specified item)

GZZ-SH13-57B<98A0>

**MITSUBISHI ELECTRIC-CHIP 16-BIT  
 MICROCOMPUTER M30621MAA-XXXGP  
 MASK ROM CONFIRMATION FORM**

Mask ROM number	
-----------------	--

Receipt	Date :	
	Section head signature	Supervisor signature

Note : Please complete all items marked \* .

* Customer	Company name	TEL (     )	Issuance signature	Submitted by	Supervisor
	Date issued	Date :			

\*1. Check sheet

Mitsubishi processes the mask files generated by the mask file generation utilities out of those held on the floppy disks you give in to us, and forms them into masks. Hence, we assume liability provided that there is any discrepancy between the contents of these mask files and the ROM data to be burned into products we produce. Check thoroughly the contents of the mask files you give in.

Prepare 3.5 inches 2HD (IBM format) floppy disks. And store only one mask file in a floppy disk.

Microcomputer type No. :      M30621MAA-XXXGP

File code :     

--	--	--	--	--	--	--	--

 (hex)

Mask file name :     

--	--	--	--	--	--	--	--

 .MSK (alpha-numeric 8-digit)

\*2. Mark specification

The mark specification differs according to the type of package. After entering the mark specification on the separate mark specification sheet (for each package), attach that sheet to this masking check sheet for submission to Mitsubishi.

For the M30621MAA-XXXGP, submit the 80P6S mark specification sheet.

\*3. Usage Conditions

For our reference when of testing our products, please reply to the following questions about the usage of the products you ordered.

(1) Which kind of X<sub>IN</sub>-X<sub>OUT</sub> oscillation circuit is used?

- Ceramic resonator                       Quartz-crystal oscillator  
 External clock input                       Other (                      )

What frequency do not use?

f(X<sub>IN</sub>) =  MHz

GZZ-SH13-57B<98A0>

Mask ROM number	
-----------------	--

**mitsubishi electric-chip 16-bit  
 MICROCOMPUTER M30621MAA-XXXGP  
 MASK ROM CONFIRMATION FORM**

(2) Which kind of XCIN-XCOUT oscillation circuit is used?

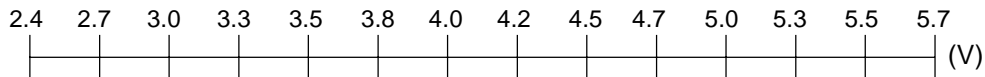
- Ceramic resonator                       Quartz-crystal oscillator  
 External clock input                       Other (                      )

What frequency do not use?

$f(XCIN) =$   kHz

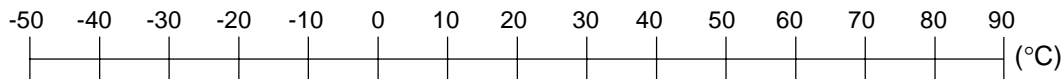
(3) Which operating supply voltage do you use?

(Circle the operating voltage range of use)



(4) Which operating ambient temperature do you use?

(Circle the operating temperature range of use)



(5) Do you use I<sup>2</sup>C (Inter IC) bus function?

- Not use                                       Use

(6) Do you use IE (Inter Equipment) bus function?

- Not use                                       Use

Thank you cooperation.

※4. Special item (Indicate none if there is not specified item)

GZZ-SH13-29B<95A0>

**MITSUBISHI ELECTRIC-CHIP 16-BIT  
 MICROCOMPUTER M30621MCA-XXXGP  
 MASK ROM CONFIRMATION FORM**

Mask ROM number	
-----------------	--

Receipt	Date :	
	Section head signature	Supervisor signature

Note : Please complete all items marked \* .

* Customer	Company name	TEL (     )	Issuance signature	Submitted by	Supervisor
	Date issued	Date :			

\*1. Check sheet

Mitsubishi processes the mask files generated by the mask file generation utilities out of those held on the floppy disks you give in to us, and forms them into masks. Hence, we assume liability provided that there is any discrepancy between the contents of these mask files and the ROM data to be burned into products we produce. Check thoroughly the contents of the mask files you give in.

Prepare 3.5 inches 2HD (IBM format) floppy disks. And store only one mask file in a floppy disk.

Microcomputer type No. :      M30621MCA-XXXGP

File code :     

--	--	--	--	--	--	--	--

 (hex)

Mask file name :     

--	--	--	--	--	--	--	--

 .MSK (alpha-numeric 8-digit)

\*2. Mark specification

The mark specification differs according to the type of package. After entering the mark specification on the separate mark specification sheet (for each package), attach that sheet to this masking check sheet for submission to Mitsubishi.

For the M30621MCA-XXXGP, submit the 80P6S mark specification sheet.

\*3. Usage Conditions

For our reference when of testing our products, please reply to the following questions about the usage of the products you ordered.

(1) Which kind of X<sub>IN</sub>-X<sub>OUT</sub> oscillation circuit is used?

- Ceramic resonator                       Quartz-crystal oscillator  
 External clock input                       Other (                      )

What frequency do not use?

f(X<sub>IN</sub>) =  MHz

GZZ-SH13-29B<95A0>

Mask ROM number	
-----------------	--

**mitsubishi electric-chip 16-bit  
 MICROCOMPUTER M30621MCA-XXXGP  
 MASK ROM CONFIRMATION FORM**

(2) Which kind of XCIN-XCOUT oscillation circuit is used?

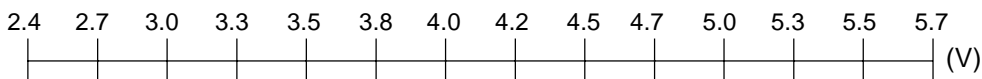
- Ceramic resonator                       Quartz-crystal oscillator  
 External clock input                       Other (                      )

What frequency do not use?

$f(XCIN) =$   kHz

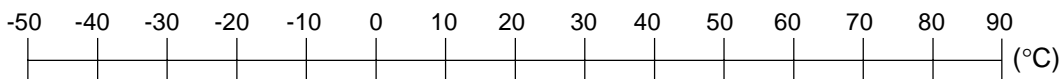
(3) Which operating supply voltage do you use?

(Circle the operating voltage range of use)



(4) Which operating ambient temperature do you use?

(Circle the operating temperature range of use)



(5) Do you use I<sup>2</sup>C (Inter IC) bus function?

- Not use     Use

(6) Do you use IE (Inter Equipment) bus function?

- Not use     Use

Thank you cooperation.

※4. Special item (Indicate none if there is not specified item)

GZZ-SH13-61B<98A0>

**MITSUBISHI ELECTRIC-CHIP 16-BIT  
 MICROCOMPUTER M30623M4A-XXXGP  
 MASK ROM CONFIRMATION FORM**

Mask ROM number	
-----------------	--

Receipt	Date :	
	Section head signature	Supervisor signature

Note : Please complete all items marked \* .

* Customer	Company name	TEL (     )	Issuance signature	Submitted by	Supervisor
	Date issued	Date :			

\*1. Check sheet

Mitsubishi processes the mask files generated by the mask file generation utilities out of those held on the floppy disks you give in to us, and forms them into masks. Hence, we assume liability provided that there is any discrepancy between the contents of these mask files and the ROM data to be burned into products we produce. Check thoroughly the contents of the mask files you give in.

Prepare 3.5 inches 2HD (IBM format) floppy disks. And store only one mask file in a floppy disk.

Microcomputer type No. :      M30623M4A-XXXGP

File code :     

--	--	--	--	--	--	--	--

 (hex)

Mask file name :     

--	--	--	--	--	--	--	--

 .MSK (alpha-numeric 8-digit)

\*2. Mark specification

The mark specification differs according to the type of package. After entering the mark specification on the separate mark specification sheet (for each package), attach that sheet to this masking check sheet for submission to Mitsubishi.

For the M30623M4A-XXXGP, submit the 80P6S mark specification sheet.

\*3. Usage Conditions

For our reference when of testing our products, please reply to the following questions about the usage of the products you ordered.

(1) Which kind of X<sub>IN</sub>-X<sub>OUT</sub> oscillation circuit is used?

- Ceramic resonator                       Quartz-crystal oscillator  
 External clock input                       Other (                      )

What frequency do not use?

f(X<sub>IN</sub>) =  MHz

GZZ-SH13-61B<98A0>

Mask ROM number	
-----------------	--

**MITSUBISHI ELECTRIC-CHIP 16-BIT  
MICROCOMPUTER M30623M4A-XXXGP  
MASK ROM CONFIRMATION FORM**

(2) Which kind of XCIN-XCOUT oscillation circuit is used?

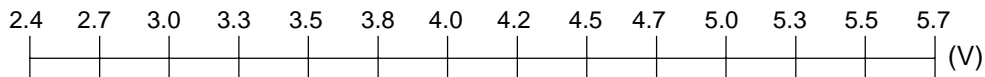
- Ceramic resonator                       Quartz-crystal oscillator  
 External clock input                       Other (                      )

What frequency do not use?

$f(XCIN) =$   kHz

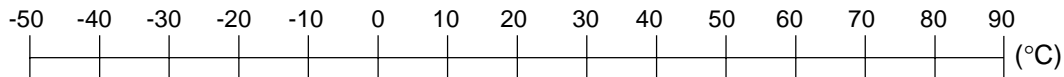
(3) Which operating supply voltage do you use?

(Circle the operating voltage range of use)



(4) Which operating ambient temperature do you use?

(Circle the operating temperature range of use)



(5) Do you use I<sup>2</sup>C (Inter IC) bus function?

- Not use                                       Use

(6) Do you use IE (Inter Equipment) bus function?

- Not use                                       Use

Thank you cooperation.

※4. Special item (Indicate none if there is not specified item)

GZZ-SH13-58B<98A0>

**MITSUBISHI ELECTRIC-CHIP 16-BIT  
 MICROCOMPUTER M30623M8A-XXXGP  
 MASK ROM CONFIRMATION FORM**

Mask ROM number	
-----------------	--

Receipt	Date :	
	Section head signature	Supervisor signature

Note : Please complete all items marked \* .

* Customer	Company name	TEL (      )	Issuance signature	Submitted by	Supervisor
	Date issued	Date :			

\*1. Check sheet

Mitsubishi processes the mask files generated by the mask file generation utilities out of those held on the floppy disks you give in to us, and forms them into masks. Hence, we assume liability provided that there is any discrepancy between the contents of these mask files and the ROM data to be burned into products we produce. Check thoroughly the contents of the mask files you give in.

Prepare 3.5 inches 2HD (IBM format) floppy disks. And store only one mask file in a floppy disk.

Microcomputer type No. :       M30623M8A-XXXGP

File code :      

--	--	--	--	--	--	--	--

 (hex)

Mask file name :      

--	--	--	--	--	--	--	--

 .MSK (alpha-numeric 8-digit)

\*2. Mark specification

The mark specification differs according to the type of package. After entering the mark specification on the separate mark specification sheet (for each package), attach that sheet to this masking check sheet for submission to Mitsubishi.

For the M30623M8A-XXXGP, submit the 80P6S mark specification sheet.

\*3. Usage Conditions

For our reference when of testing our products, please reply to the following questions about the usage of the products you ordered.

(1) Which kind of X<sub>IN</sub>-X<sub>OUT</sub> oscillation circuit is used?

- Ceramic resonator       Quartz-crystal oscillator  
 External clock input       Other (                      )

What frequency do not use?

f(X<sub>IN</sub>) = 

--

 MHz



GZZ-SH13-58B<98A0>

Mask ROM number	
-----------------	--

**MITSUBISHI ELECTRIC-CHIP 16-BIT  
 MICROCOMPUTER M30623M8A-XXXGP  
 MASK ROM CONFIRMATION FORM**

(2) Which kind of XCIN-XCOUT oscillation circuit is used?

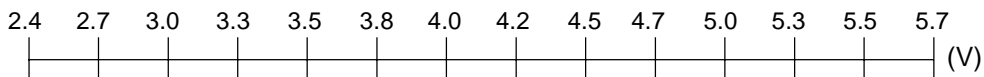
- Ceramic resonator                       Quartz-crystal oscillator  
 External clock input                       Other (                      )

What frequency do not use?

f(XCIN) =  kHz

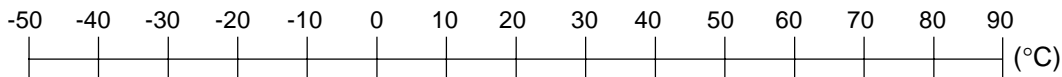
(3) Which operating supply voltage do you use?

(Circle the operating voltage range of use)



(4) Which operating ambient temperature do you use?

(Circle the operating temperature range of use)



(5) Do you use I<sup>2</sup>C (Inter IC) bus function?

- Not use     Use

(6) Do you use IE (Inter Equipment) bus function?

- Not use     Use

Thank you cooperation.

※4. Special item (Indicate none if there is not specified item)

GZZ-SH13-59B<98A0>

**MITSUBISHI ELECTRIC-CHIP 16-BIT  
 MICROCOMPUTER M30623MAA-XXXGP  
 MASK ROM CONFIRMATION FORM**

Mask ROM number	
-----------------	--

Receipt	Date :	
	Section head signature	Supervisor signature

Note : Please complete all items marked \* .

* Customer	Company name	TEL (      )	Issuance signature	Submitted by	Supervisor
	Date issued	Date :			

\*1. Check sheet

Mitsubishi processes the mask files generated by the mask file generation utilities out of those held on the floppy disks you give in to us, and forms them into masks. Hence, we assume liability provided that there is any discrepancy between the contents of these mask files and the ROM data to be burned into products we produce. Check thoroughly the contents of the mask files you give in.

Prepare 3.5 inches 2HD (IBM format) floppy disks. And store only one mask file in a floppy disk.

Microcomputer type No. :       M30623MAA-XXXGP

File code :      

--	--	--	--	--	--	--	--

 (hex)

Mask file name :      

--	--	--	--	--	--	--	--

 .MSK (alpha-numeric 8-digit)

\*2. Mark specification

The mark specification differs according to the type of package. After entering the mark specification on the separate mark specification sheet (for each package), attach that sheet to this masking check sheet for submission to Mitsubishi.

For the M30623MAA-XXXGP, submit the 80P6S mark specification sheet.

\*3. Usage Conditions

For our reference when of testing our products, please reply to the following questions about the usage of the products you ordered.

(1) Which kind of X<sub>IN</sub>-X<sub>OUT</sub> oscillation circuit is used?

- Ceramic resonator       Quartz-crystal oscillator  
 External clock input       Other (                      )

What frequency do not use?

f(X<sub>IN</sub>) = 

--

 MHz

GZZ-SH13-59B<98A0>

Mask ROM number	
-----------------	--

**mitsubishi electric-chip 16-bit  
 MICROCOMPUTER M30623MAA-XXXGP  
 MASK ROM CONFIRMATION FORM**

(2) Which kind of XCIN-XCOUT oscillation circuit is used?

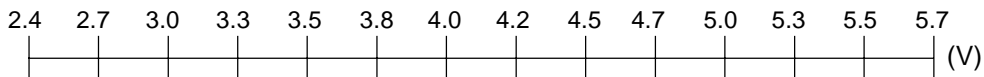
- Ceramic resonator                       Quartz-crystal oscillator  
 External clock input                       Other (                      )

What frequency do not use?

$f(XCIN) =$   kHz

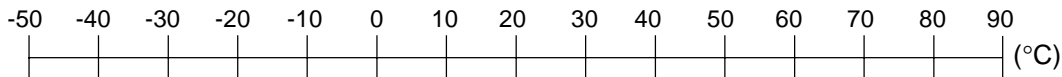
(3) Which operating supply voltage do you use?

(Circle the operating voltage range of use)



(4) Which operating ambient temperature do you use?

(Circle the operating temperature range of use)



(5) Do you use I<sup>2</sup>C (Inter IC) bus function?

- Not use     Use

(6) Do you use IE (Inter Equipment) bus function?

- Not use     Use

Thank you cooperation.

\*4. Special item (Indicate none if there is not specified item)

GZZ-SH13-60B<98A0>

**MITSUBISHI ELECTRIC-CHIP 16-BIT  
 MICROCOMPUTER M30623MCA-XXXGP  
 MASK ROM CONFIRMATION FORM**

Mask ROM number	
-----------------	--

Receipt	Date :	
	Section head signature	Supervisor signature

Note : Please complete all items marked \* .

* Customer	Company name	TEL (      )	Issuance signature	Submitted by	Supervisor
	Date issued	Date :			

\*1. Check sheet

Mitsubishi processes the mask files generated by the mask file generation utilities out of those held on the floppy disks you give in to us, and forms them into masks. Hence, we assume liability provided that there is any discrepancy between the contents of these mask files and the ROM data to be burned into products we produce. Check thoroughly the contents of the mask files you give in.

Prepare 3.5 inches 2HD (IBM format) floppy disks. And store only one mask file in a floppy disk.

Microcomputer type No. :       M30623MCA-XXXGP

File code :      

--	--	--	--	--	--	--	--

 (hex)

Mask file name :      

--	--	--	--	--	--	--	--

 .MSK (alpha-numeric 8-digit)

\*2. Mark specification

The mark specification differs according to the type of package. After entering the mark specification on the separate mark specification sheet (for each package), attach that sheet to this masking check sheet for submission to Mitsubishi.

For the M30623MCA-XXXGP, submit the 80P6S mark specification sheet.

\*3. Usage Conditions

For our reference when of testing our products, please reply to the following questions about the usage of the products you ordered.

(1) Which kind of XIN-XOUT oscillation circuit is used?

- |   |   |
|---|---|
| <input type="checkbox"/> Ceramic resonator    | <input type="checkbox"/> Quartz-crystal oscillator      |
| <input type="checkbox"/> External clock input | <input type="checkbox"/> Other (                      ) |

What frequency do not use?

f(XIN) =  MHz

GZZ-SH13-60B<98A0>

Mask ROM number	
-----------------	--

**MITSUBISHI ELECTRIC-CHIP 16-BIT  
 MICROCOMPUTER M30623MCA-XXXGP  
 MASK ROM CONFIRMATION FORM**

(2) Which kind of XCIN-XCOUT oscillation circuit is used?

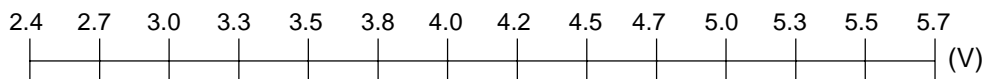
- Ceramic resonator                       Quartz-crystal oscillator  
 External clock input                       Other (                      )

What frequency do not use?

$f(XCIN) =$   kHz

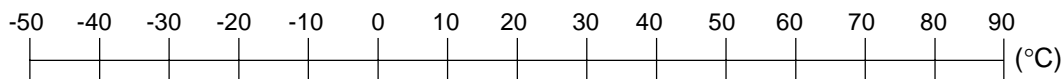
(3) Which operating supply voltage do you use?

(Circle the operating voltage range of use)



(4) Which operating ambient temperature do you use?

(Circle the operating temperature range of use)



(5) Do you use I<sup>2</sup>C (Inter IC) bus function?

- Not use     Use

(6) Do you use IE (Inter Equipment) bus function?

- Not use     Use

Thank you cooperation.

※4. Special item (Indicate none if there is not specified item)

GZZ-SH13-31B<95A0>

**MITSUBISHI ELECTRIC-CHIP 16-BIT  
 MICROCOMPUTER M30625MGA-XXXGP  
 MASK ROM CONFIRMATION FORM**

Mask ROM number	
-----------------	--

Receipt	Date :	
	Section head signature	Supervisor signature

Note : Please complete all items marked \* .

* Customer	Company name	TEL (     )	Issuance signature	Submitted by	Supervisor
	Date issued	Date :			

\*1. Check sheet

Mitsubishi processes the mask files generated by the mask file generation utilities out of those held on the floppy disks you give in to us, and forms them into masks. Hence, we assume liability provided that there is any discrepancy between the contents of these mask files and the ROM data to be burned into products we produce. Check thoroughly the contents of the mask files you give in.

Prepare 3.5 inches 2HD (IBM format) floppy disks. And store only one mask file in a floppy disk.

Microcomputer type No. :      M30625MGA-XXXGP

File code :     

--	--	--	--	--	--	--	--

 (hex)

Mask file name :     

--	--	--	--	--	--	--	--

 .MSK (alpha-numeric 8-digit)

\*2. Mark specification

The mark specification differs according to the type of package. After entering the mark specification on the separate mark specification sheet (for each package), attach that sheet to this masking check sheet for submission to Mitsubishi.

For the M30625MGA-XXXGP, submit the 80P6S mark specification sheet.

\*3. Usage Conditions

For our reference when of testing our products, please reply to the following questions about the usage of the products you ordered.

(1) Which kind of XIN-XOUT oscillation circuit is used?

- Ceramic resonator                       Quartz-crystal oscillator  
 External clock input                       Other (                      )

What frequency do not use?

f(XIN) =  MHz

GZZ-SH13-31B<95A0>

Mask ROM number	
-----------------	--

**MITSUBISHI ELECTRIC-CHIP 16-BIT  
MICROCOMPUTER M30625MGA-XXXGP  
MASK ROM CONFIRMATION FORM**

(2) Which kind of XCIN-XCOUT oscillation circuit is used?

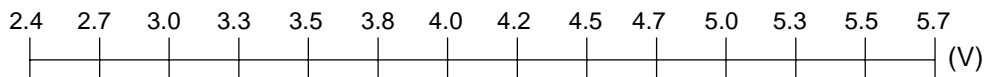
- Ceramic resonator                       Quartz-crystal oscillator  
 External clock input                       Other (                      )

What frequency do not use?

$f(\text{XCIN}) =$   kHz

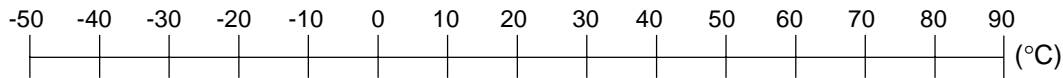
(3) Which operating supply voltage do you use?

(Circle the operating voltage range of use)



(4) Which operating ambient temperature do you use?

(Circle the operating temperature range of use)



(5) Do you use I<sup>2</sup>C (Inter IC) bus function?

- Not use                                       Use

(6) Do you use IE (Inter Equipment) bus function?

- Not use                                       Use

Thank you cooperation.

※4. Special item (Indicate none if there is not specified item)

Description (Flash Memory Version)

**Outline Performance (flash memory version)**

Table 1.21.1 shows the outline performance of the M16C/62A (80-pin flash memory version).

**Table 1.21.1. Outline performance of the M16C/62A (80-pin flash memory version)**

Item		Performance
Flash memory operation mode		Three modes (parallel I/O, standard serial I/O, CPU rewrite)
Erase block division	User ROM area	See Figure 1.21.1
	Boot ROM area	One division (8 Kbytes) (Note)
Program method		In units of pages (in units of 256 bytes)
Erase method		Collective erase/block erase
Program/erase control method		Program/erase control by software command
Protect method		Protected for each block by lock bit
Number of commands		8 commands
Program/erase count		100 times
Data Retention		10 years
ROM code protect		Parallel I/O and standard serial I/O modes are 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 Version)

**Flash Memory**

The M16C/62A (80-pin flash memory version) contains the flash memory that can be rewritten with a single voltage. 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.

The flash memory is divided into several blocks as shown in Figure 1.21.1, so that memory can be erased one block at a time. Each block has a lock bit to enable or disable execution of an erase or program operation, allowing for data in each block to be protected.

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.

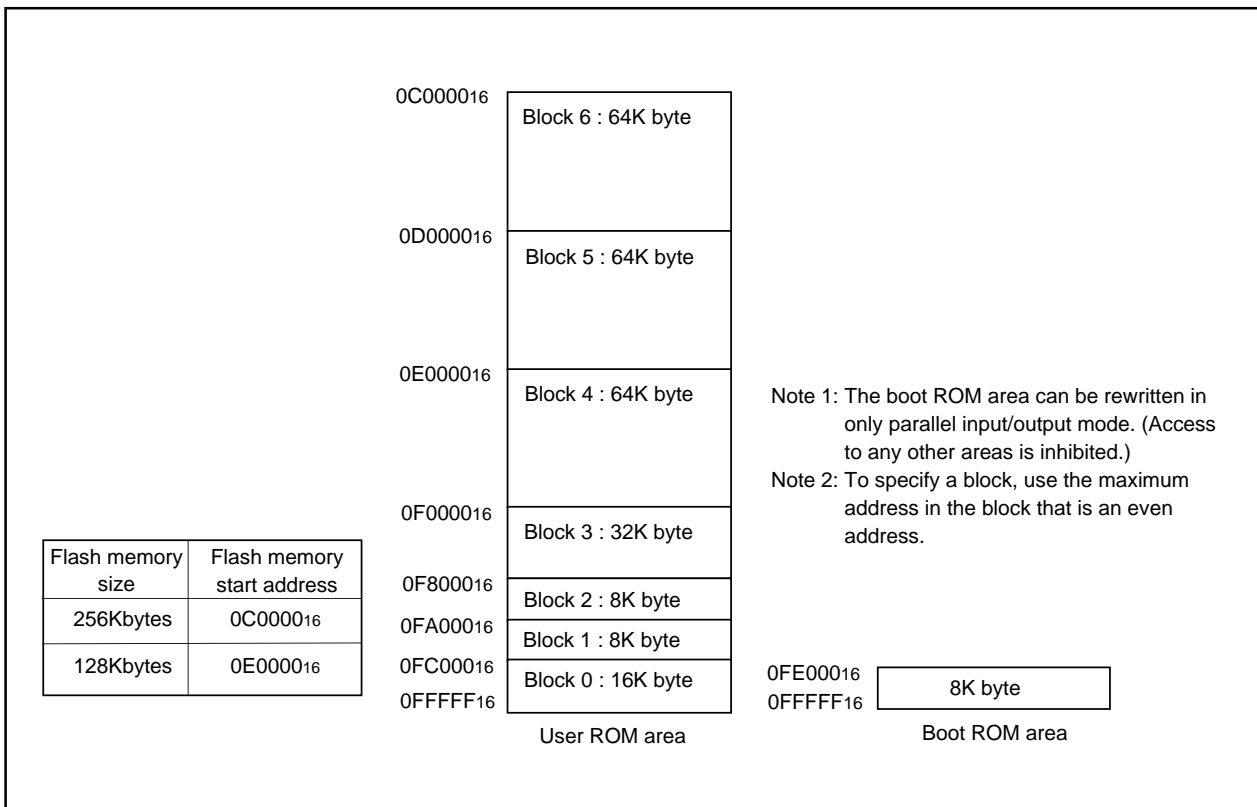


Figure 1.21.1. Block diagram of flash memory version

## 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, only the user ROM area shown in Figure 1.21.1 can be rewritten; the boot ROM area cannot be rewritten. Make sure the program and block erase commands are issued for only the user ROM area and each block 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 any area other than the internal flash memory before it can be executed.

## 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 1.21.1 for details about the boot ROM area.

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

When the microcomputer is reset by pulling the P55 pin low, the CNVss pin high, and the P50 pin high, 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.

## Block Address

Block addresses refer to the maximum even address of each block. These addresses are used in the block erase command, lock bit program command, and read lock status command.

## Outline Performance (CPU Rewrite Mode)

In the CPU rewrite mode, the CPU erases, programs and reads the internal flash memory as instructed by software commands. Operations must be executed from a memory other than the internal flash memory, such as the internal RAM.

When the CPU rewrite mode select bit (bit 1 at address 03B716) is set to "1", transition to CPU rewrite mode occurs and software commands can be accepted.

In the CPU rewrite mode, write to and read from software commands and data into even-numbered address ("0" for byte address A0) in 16-bit units. Always write 8-bit software commands into even-numbered address. Commands are ignored with odd-numbered addresses.

Use software commands to control program and erase operations. Whether a program or erase operation has terminated normally or in error can be verified by reading the status register.

Figure 1.22.1 shows the flash memory control register 0 and the flash memory control register 1.

Bit 0 of the flash memory control register 0 is the RY/B $\bar{Y}$  status flag used exclusively to read the operating status of the flash memory. During programming and erase operations, it is "0". Otherwise, it is "1".

Bit 1 of the flash memory control register 0 is the CPU rewrite mode select bit. The CPU rewrite mode is entered by setting this bit to "1", so that software commands become acceptable. In CPU rewrite mode, the CPU becomes unable to access the internal flash memory directly. Therefore, write bit 1 in an area other than the internal flash memory. Also only when  $\overline{\text{NMI}}$  pin is "H" level. To set this bit to "1", it is necessary to write "0" and then write "1" in succession. The bit can be set to "0" by only writing a "0".

Bit 2 of the flash memory control register 0 is a lock bit disable select bit. By setting this bit to "1", it is possible to disable erase and write protect (block lock) effectuated by the lock bit data. The lock bit disable select bit only disables the lock bit function; it does not change the lock data bit value. However, if an erase operation is performed when this bit = "1", the lock bit data that is "0" (locked) is set to "1" (unlocked) after erasure. To set this bit to "1", it is necessary to write "0" and then write "1" in succession. This bit can be manipulated only when the CPU rewrite mode select bit = "1".

Bit 3 of the flash memory control register 0 is the flash memory reset bit used to reset the control circuit of the internal flash memory. This bit is used when exiting CPU rewrite mode and when flash memory access has failed. When the CPU rewrite mode select bit is "1", writing "1" for this bit resets the control circuit. To release the reset, it is necessary to set this bit to "0".

Bit 5 of the flash memory control register 0 is a user ROM area select bit which is effective in only boot mode. If this bit is set to "1" in boot mode, the area to be accessed is switched from the boot ROM area to the user ROM area. When the CPU rewrite mode needs to be used in boot mode, set this bit to "1". Note that if the microcomputer is booted from the user ROM area, it is always the user ROM area that can be accessed and this bit has no effect. When in boot mode, the function of this bit is effective regardless of whether the CPU rewrite mode is on or off. Write to this bit only when executing out of an area other than the internal flash memory.

Bit 3 of the flash memory control register 1 turns power supply to the internal flash memory on/off. When this bit is set to "1", power is not supplied to the internal flash memory, thus power consumption can be reduced. However, in this state, the internal flash memory cannot be accessed. To set this bit to "1", it is necessary to write "0" and then write "1" in succession. Use this bit mainly in the low speed mode (when XCIN is the block count source of BCLK).

When the CPU is shifted to the stop or wait modes, power to the internal flash memory is automatically shut off. It is reconnected automatically when CPU operation is restored. Therefore, it is not particularly necessary to set flash memory control register 1.

## CPU Rewrite Mode (Flash Memory Version)

Figure 1.22.2 shows a flowchart for setting/releasing the CPU rewrite mode. Figure 1.22.3 shows a flowchart for shifting to the low speed mode. Always perform operation as indicated in these flowcharts.

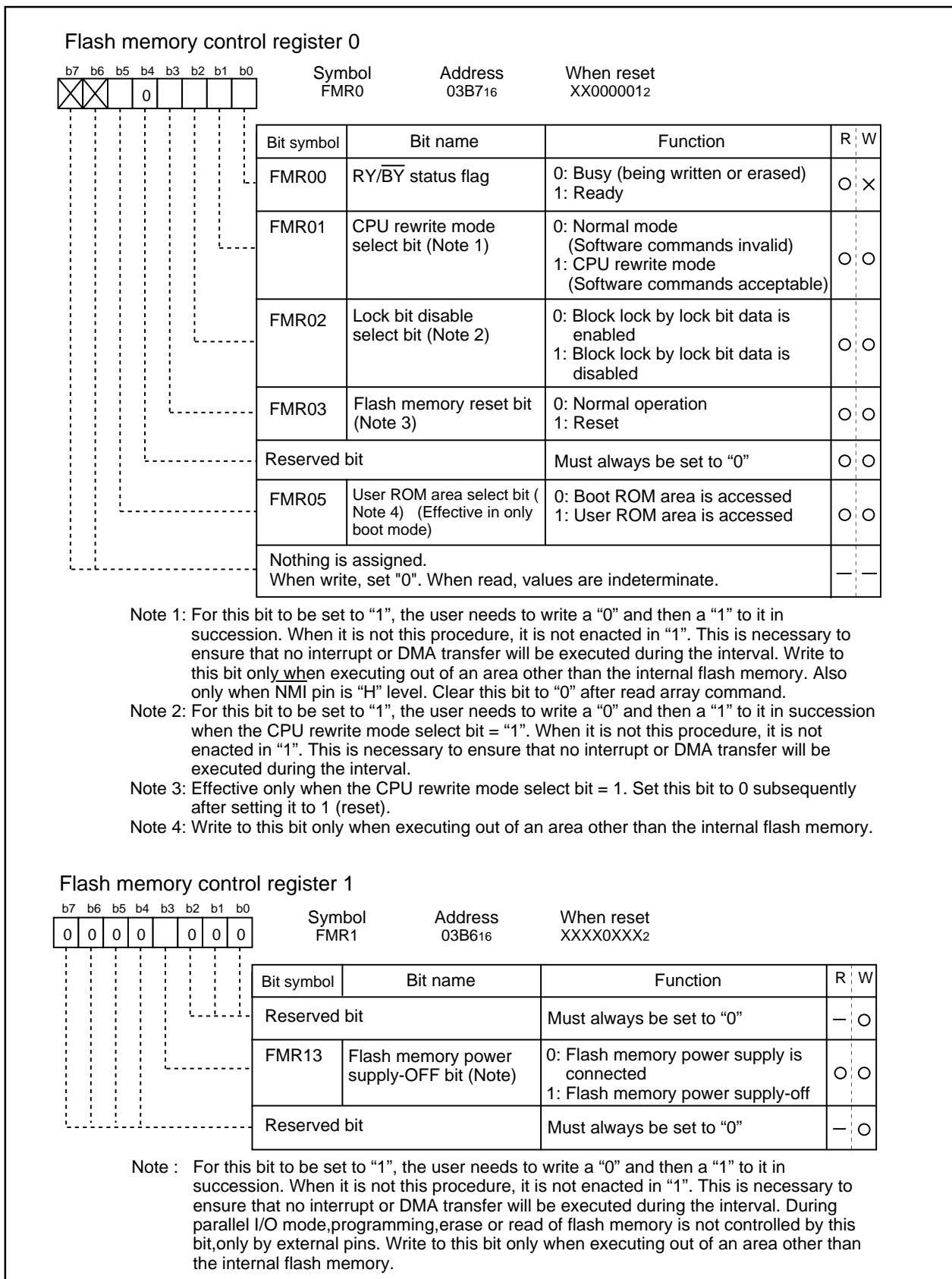


Figure 1.22.1. Flash memory control registers

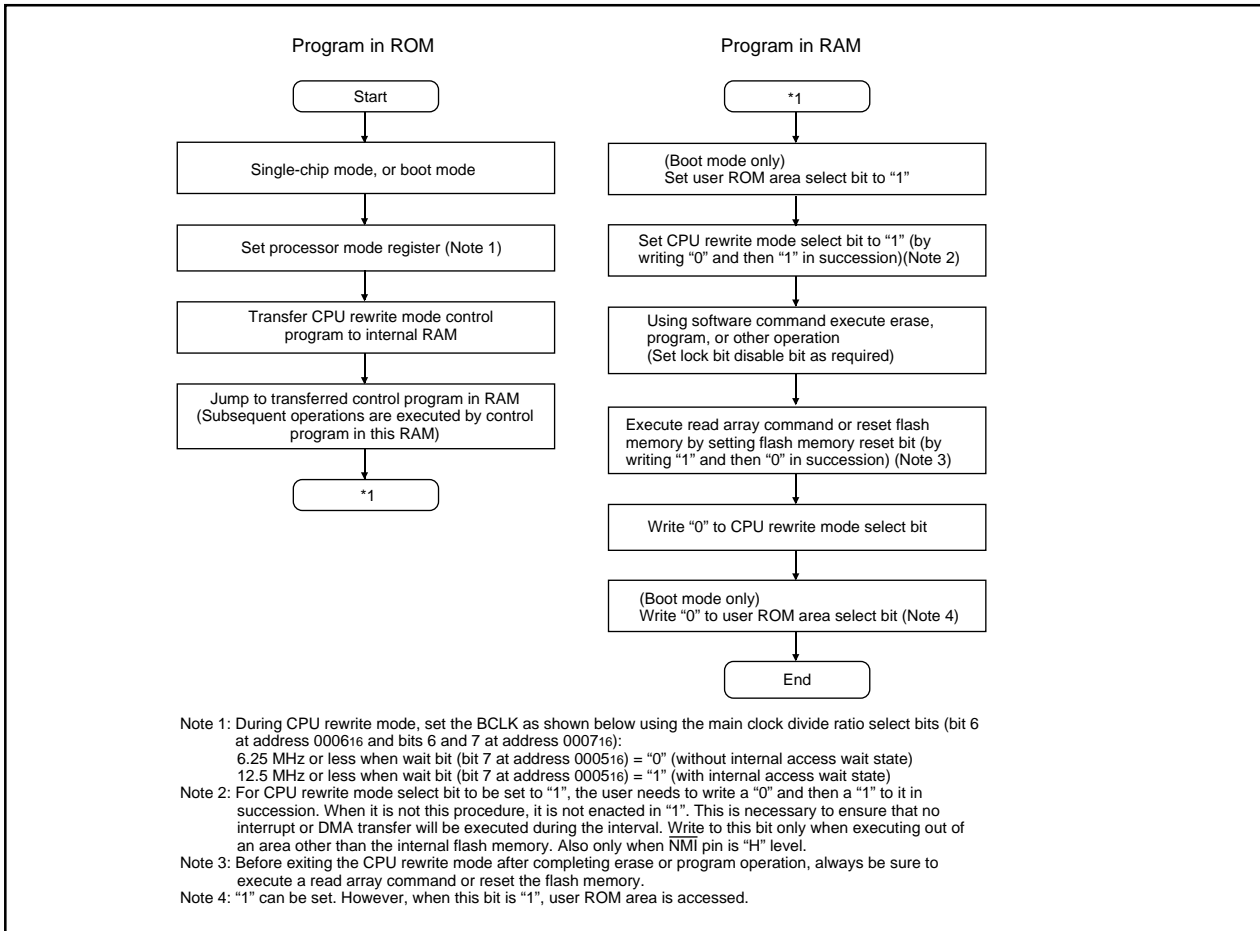


Figure 1.22.2. CPU rewrite mode set/reset flowchart

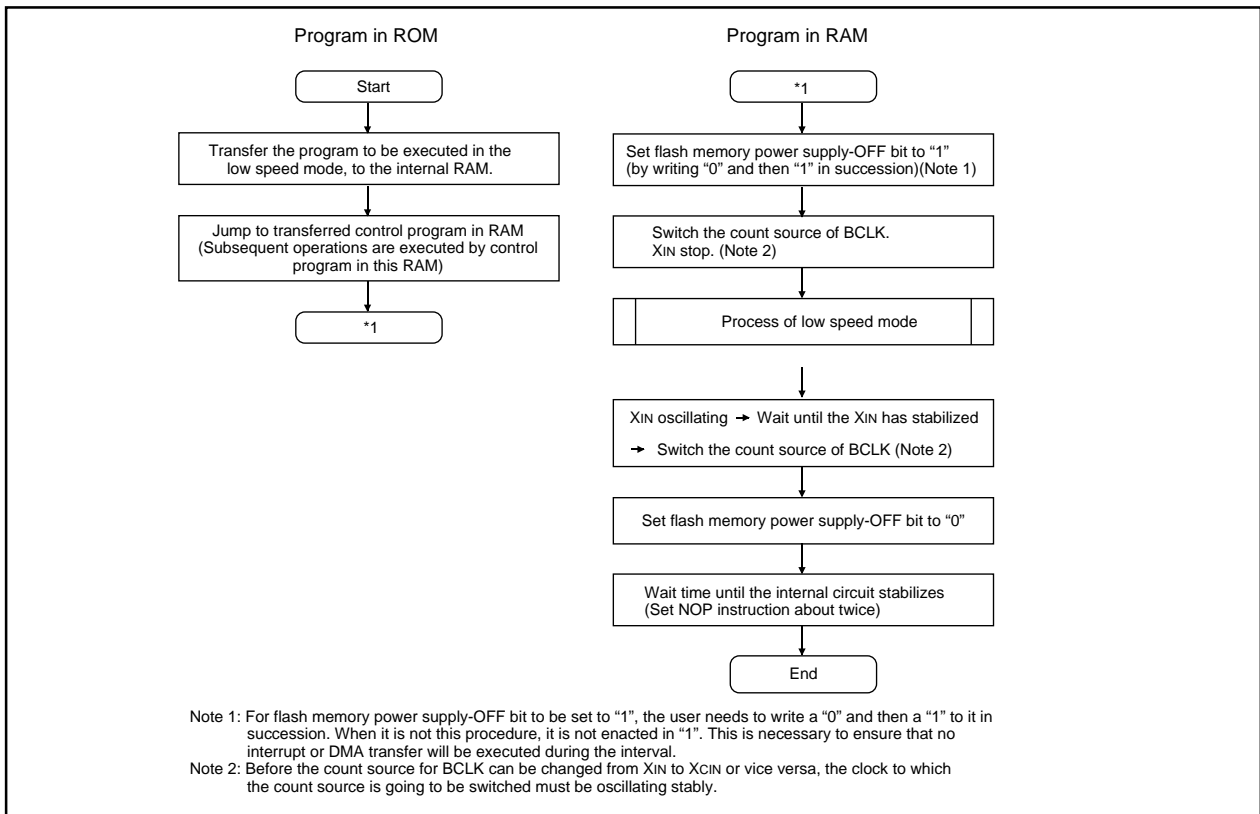


Figure 1.22.3. Shifting to the low speed mode flowchart

## 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 CPU rewrite mode, set the BCLK as shown below using the main clock divide ratio select bit (bit 6 at address 0006<sub>16</sub> and bits 6 and 7 at address 0007<sub>16</sub>):

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

12.5 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

The address match interrupt cannot be used during CPU rewrite mode because they refer to the internal data of the flash memory. If interrupts have their vector in the variable vector table, they can be used by transferring the vector into the RAM area. The  $\overline{\text{NMI}}$  and watchdog timer interrupts can be used because the flash memory control register 0 and 1 is forcibly initialized and return to normal mode when each interrupt occurs. But it is needed that the jump addresses for each interrupt are set in the fixed vector table and there is an interrupt program. Since the rewrite operation is halted when the  $\overline{\text{NMI}}$  and watchdog timer interrupts occur, it is needed that CPU rewriting mode select bit is set to "1" and the erase/program operation is performed over again.

### (4) Internal reserved area expansion bit (Bit 3 at address 0005<sub>16</sub>)

The reserved area of the internal memory can be changed by using the internal reserved area expansion bit (bit 3 at address 0005<sub>16</sub>). However, if the CPU rewrite mode select bit (bit 1 at address 03B7<sub>16</sub>) is set to 1, the internal reserved area expansion bit (bit 3 at address 0005<sub>16</sub>) also is set to 1 automatically. Similarly, if the CPU rewrite mode select bit (bit 1 at address 03B7<sub>16</sub>) is set to 0, the internal reserved area expansion bit (bit 3 at address 0005<sub>16</sub>) also is set to 0 automatically.

The precautions above apply to the products which RAM size is over 15 Kbytes or flash memory size is over 192 Kbytes.

### (5) Reset

Reset input is always accepted. After a reset, the addresses 0C0000<sub>16</sub> through 0CFFFF<sub>16</sub> are made a reserved area and cannot be accessed. Therefore, if your product has this area in the user ROM area, do not write any address of this area to the reset vector. This area is made accessible by changing the internal reserved area expansion bit (bit 3 at address 0005<sub>16</sub>) in a program.

### (6) Access disable

Write CPU rewrite mode select bit, flash memory power supply-OFF bit and user ROM area select bit only when executing out of an area other than the internal flash memory.

### (7) How to access

For CPU rewrite mode select bit, lock bit disable select bit, and flash memory power supply-OFF bit to be set to "1", the user needs to write a "0" and then a "1" to it in succession. When it is not this procedure, it is not enacted in "1". This is necessary to ensure that no interrupt or DMA transfer will be executed during the interval.

Write CPU rewrite mode select bit only when executing out of an area other than the internal flash memory. Also only when  $\overline{\text{NMI}}$  pin is "H" level.

**(8) Writing in the user ROM area**

If power is lost while rewriting blocks that contain the flash rewrite program with the CPU rewrite mode, those blocks may not be correctly rewritten and it is possible that the flash memory can no longer be rewritten after that. Therefore, it is recommended to use the standard serial I/O mode or parallel I/O mode to rewrite these blocks.

**(9) Using the lock bit**

To use the CPU rewrite mode, use a boot program that can set and cancel the lock command.

## CPU Rewrite Mode (Flash Memory Version)

## Software Commands

Table 1.22.1 lists the software commands available with the M16C/62A (80-pin flash memory version). After setting the CPU rewrite mode select bit to 1, write a software command to specify an erase or program operation. Note that when entering a software command, the upper byte (D<sub>8</sub> to D<sub>15</sub>) is ignored. The content of each software command is explained below.

**Table 1.22.1. List of software commands (CPU rewrite mode)**

Command	First bus cycle			Second bus cycle			Third 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> )	Mode	Address	Data (D <sub>0</sub> to D <sub>7</sub> )
Read array	Write	X (Note 6)	FF <sub>16</sub>						
Read status register	Write	X	70 <sub>16</sub>	Read	X	SRD (Note 2)			
Clear status register	Write	X	50 <sub>16</sub>						
Page program (Note 3)	Write	X	41 <sub>16</sub>	Write	WA0 (Note 3)	WD0 (Note 3)	Write	WA1	WD1
Block erase	Write	X	20 <sub>16</sub>	Write	BA (Note 4)	D0 <sub>16</sub>			
Erase all unlock blocks	Write	X	A7 <sub>16</sub>	Write	X	D0 <sub>16</sub>			
Lock bit program	Write	X	77 <sub>16</sub>	Write	BA	D0 <sub>16</sub>			
Read lock bit status	Write	X	71 <sub>16</sub>	Read	BA	D <sub>6</sub> (Note 5)			

Note 1: When a software command is input, the high-order byte of data (D<sub>8</sub> to D<sub>15</sub>) is ignored.

Note 2: SRD = Status Register Data

Note 3: WA = Write Address, WD = Write Data

WA and WD must be set sequentially from 00<sub>16</sub> to FE<sub>16</sub> (byte address; however, an even address). The page size is 256 bytes.

Note 4: BA = Block Address (Enter the maximum address of each block that is an even address.)

Note 5: D<sub>6</sub> corresponds to the block lock status. Block not locked when D<sub>6</sub> = 1, block locked when D<sub>6</sub> = 0.

Note 6: X denotes a given address in the user ROM area (that is an even address).

### Read Array Command (FF<sub>16</sub>)

The read array mode is entered by writing the command code "FF<sub>16</sub>" in the first bus cycle. When an even 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>15</sub>), 16 bits at a time.

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

### Read Status Register Command (70<sub>16</sub>)

When the command code "70<sub>16</sub>" is written in the first bus cycle, the content of the status register is read out at the data bus (D<sub>0</sub>–D<sub>7</sub>) by a read in the second bus cycle.

The status register is explained in the next section.

### Clear Status Register Command (50<sub>16</sub>)

This command is used to clear the bits SR<sub>3</sub> to 5 of the status register after they have been set. These bits indicate that operation has ended in an error. To use this command, write the command code "50<sub>16</sub>" in the first bus cycle.



**Page Program Command (41<sub>16</sub>)**

Page program allows for high-speed programming in units of 256 bytes. Page program operation starts when the command code "41<sub>16</sub>" is written in the first bus cycle. In the second bus cycle through the 129th bus cycle, the write data is sequentially written 16 bits at a time. At this time, the addresses A0-A7 need to be incremented by 2 from "00<sub>16</sub>" to "FE<sub>16</sub>." When the system finishes loading the data, it starts an auto write operation (data program and verify operation).

Whether the auto write operation is completed can be confirmed by reading the status register or the flash memory control register 0. At the same time the auto write operation starts, the read status register mode is automatically entered, so the content of the status register can be read out. The status register bit 7 (SR7) is set to 0 at the same time the auto write operation starts and is returned to 1 upon completion of the auto write operation. In this case, the read status register mode remains active until the Read Array command (FF<sub>16</sub>) or Read Lock Bit Status command (71<sub>16</sub>) is written or the flash memory is reset using its reset bit.

The RY/ $\overline{\text{BY}}$  status flag of the flash memory control register 0 is 0 during auto write operation and 1 when the auto write operation is completed as is the status register bit 7.

After the auto write operation is completed, the status register can be read out to know the result of the auto write operation. For details, refer to the section where the status register is detailed.

Figure 1.22.4 shows an example of a page program flowchart.

Each block of the flash memory can be write protected by using a lock bit. For details, refer to the section where the data protect function is detailed.

Additional writes to the already programmed pages are prohibited.

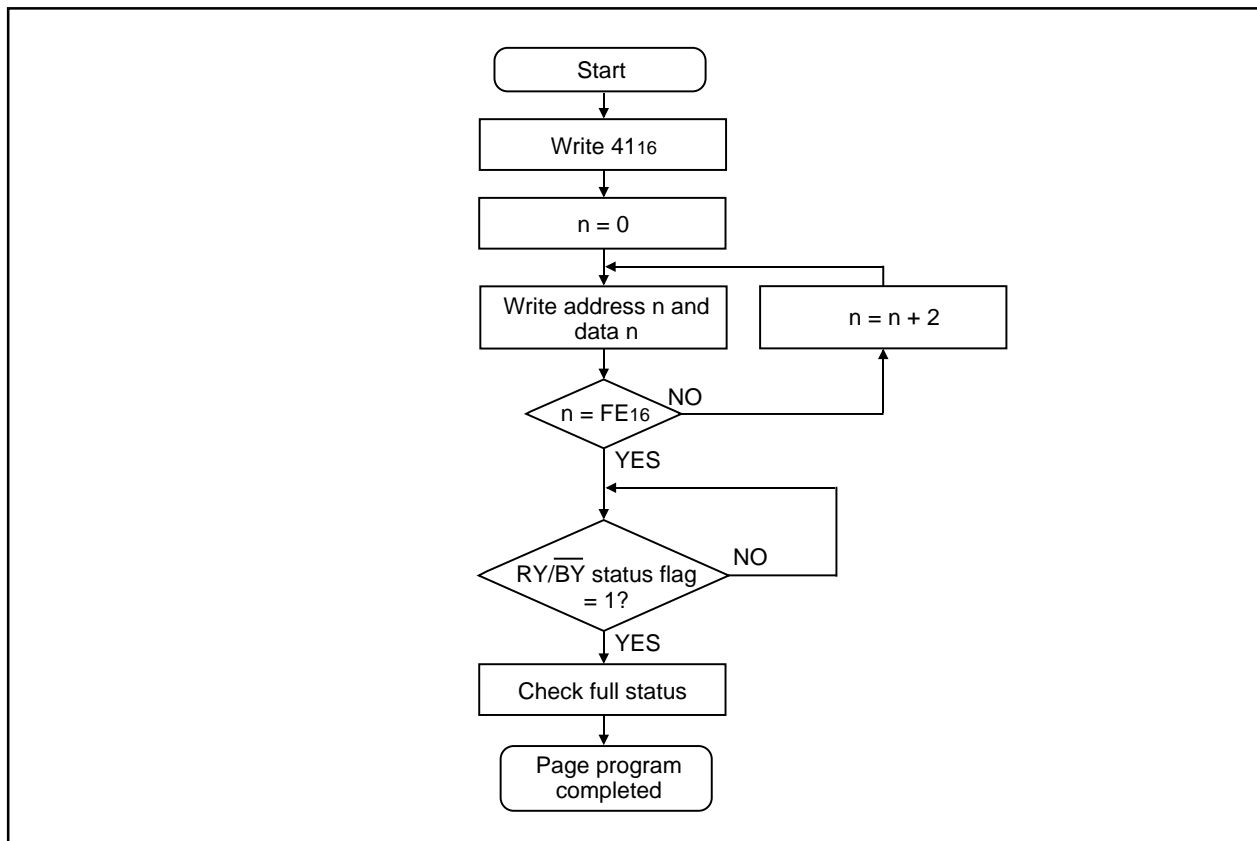


Figure 1.22.4. Page program flowchart

**Block Erase Command (2016/D016)**

By writing the command code "2016" in the first bus cycle and the confirmation command code "D016" in the second bus cycle that follows to the block address of a flash memory block, the system initiates an auto erase (erase and erase verify) operation.

Whether the auto erase operation is completed can be confirmed by reading the status register or the flash memory control register 0. At the same time the auto erase operation starts, the read status register mode is automatically entered, so the content of the status register can be read out. The status register bit 7 (SR7) is set to 0 at the same time the auto erase operation starts and is returned to 1 upon completion of the auto erase operation. In this case, the read status register mode remains active until the Read Array command (FF16) or Read Lock Bit Status command (7116) is written or the flash memory is reset using its reset bit.

The RY/ $\overline{\text{BY}}$  status flag of the flash memory control register 0 is 0 during auto erase operation and 1 when the auto erase operation is completed as is the status register bit 7.

After the auto erase operation is completed, the status register can be read out to know the result of the auto erase operation. For details, refer to the section where the status register is detailed.

Figure 1.22.5 shows an example of a block erase flowchart.

Each block of the flash memory can be protected against erasure by using a lock bit. For details, refer to the section where the data protect function is detailed.

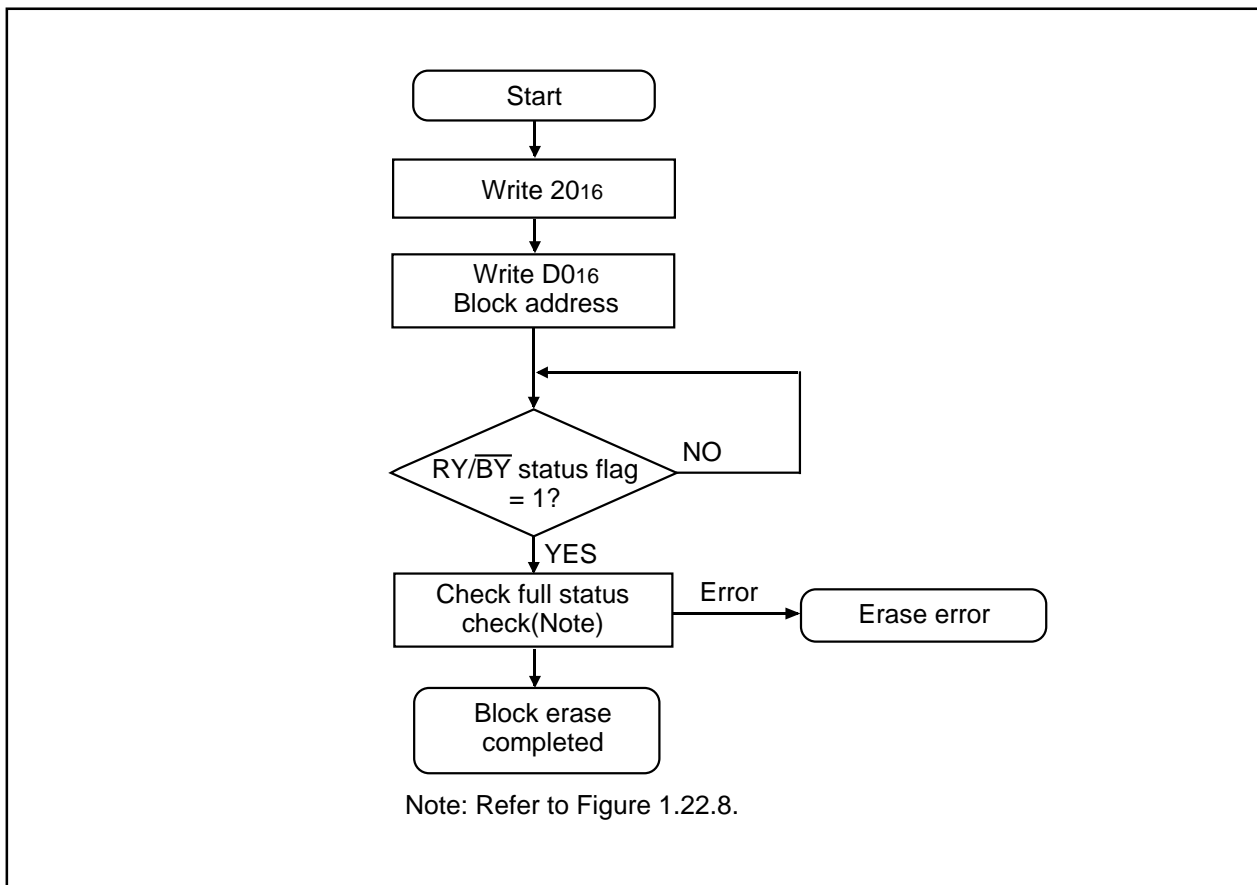


Figure 1.22.5. Block erase flowchart

**Erase All Unlock Blocks Command (A716/D016)**

By writing the command code "A716" in the first bus cycle and the confirmation command code "D016" in the second bus cycle that follows, the system starts erasing blocks successively.

Whether the erase all unlock blocks command is terminated can be confirmed by reading the status register or the flash memory control register 0, in the same way as for block erase. Also, the status register can be read out to know the result of the auto erase operation.

When the lock bit disable select bit of the flash memory control register 0 = 1, all blocks are erased no matter how the lock bit is set. On the other hand, when the lock bit disable bit = 0, the function of the lock bit is effective and only nonlocked blocks (where lock bit data = 1) are erased.

**Lock Bit Program Command (7716/D016)**

By writing the command code "7716" in the first bus cycle and the confirmation command code "D016" in the second bus cycle that follows to the block address of a flash memory block, the system sets the lock bit for the specified block to 0 (locked).

Figure 1.22.6 shows an example of a lock bit program flowchart. The status of the lock bit (lock bit data) can be read out by a read lock bit status command.

Whether the lock bit program command is terminated can be confirmed by reading the status register or the flash memory control register 0, in the same way as for page program.

For details about the function of the lock bit and how to reset the lock bit, refer to the section where the data protect function is detailed.

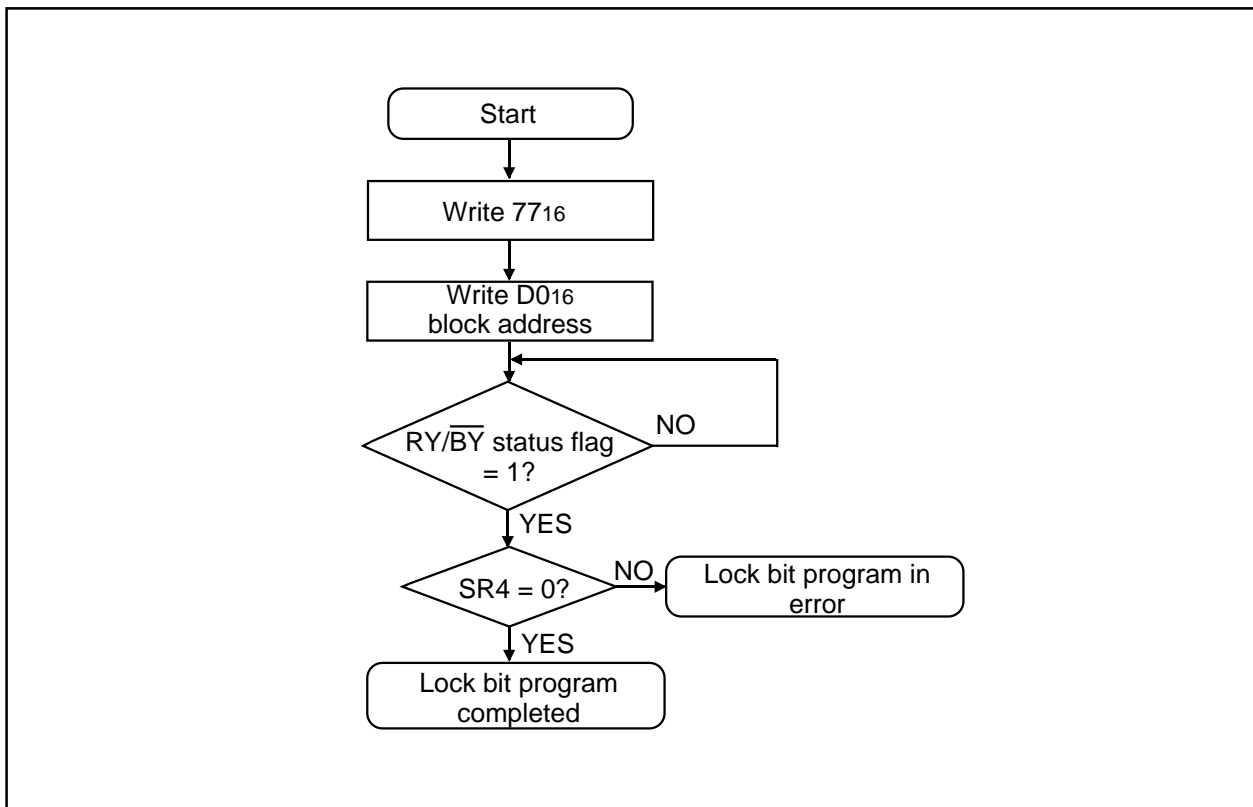


Figure 1.22.6. Lock bit program flowchart

**Read Lock Bit Status Command (7116)**

By writing the command code "7116" in the first bus cycle and then the block address of a flash memory block in the second bus cycle that follows, the system reads out the status of the lock bit of the specified block on to the data bus (D6).

Figure 1.22.7 shows an example of a read lock bit program flowchart.

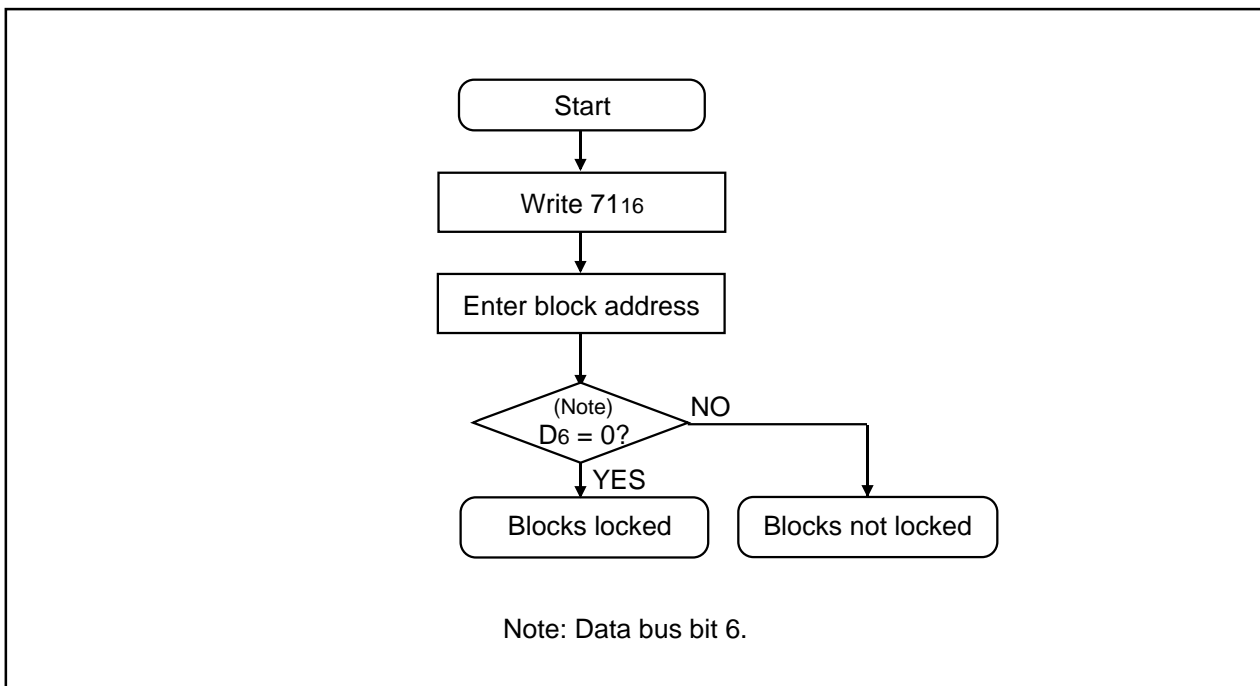


Figure 1.22.7. Read lock bit status flowchart

## Data Protect Function (Block Lock)

Each block in Figure 1.21.1 has a nonvolatile lock bit to specify that the block be protected (locked) against erase/write. The lock bit program command is used to set the lock bit to 0 (locked). The lock bit of each block can be read out using the read lock bit status command.

Whether block lock is enabled or disabled is determined by the status of the lock bit and how the flash memory control register 0's lock bit disable select bit is set.

- (1) When the lock bit disable select bit = 0, a specified block can be locked or unlocked by the lock bit status (lock bit data). Blocks whose lock bit data = 0 are locked, so they are disabled against erase/write. On the other hand, the blocks whose lock bit data = 1 are not locked, so they are enabled for erase/write.
- (2) When the lock bit disable select bit = 1, all blocks are nonlocked regardless of the lock bit data, so they are enabled for erase/write. In this case, the lock bit data that is 0 (locked) is set to 1 (nonlocked) after erasure, so that the lock bit-actuated lock is removed.

## Status Register

The status register indicates the operating status of the flash memory and whether an erase or program operation has terminated normally or in an error. The content of this register can be read out by only writing the read status register command (70<sub>16</sub>). Table 1.22.2 details the status register.

The status register is cleared by writing the Clear Status Register command (50<sub>16</sub>).

After a reset, the status register is set to "80<sub>16</sub>."

Each bit in this register is explained below.

### Write state machine (WSM) status (SR7)

After power-on, the write state machine (WSM) status is set to 1.

The write state machine (WSM) status indicates the operating status of the device, as for output on the RY/ $\overline{\text{BY}}$  pin. This status bit is set to 0 during auto write or auto erase operation and is set to 1 upon completion of these operations.

### Erase status (SR5)

The erase status informs the operating status of auto erase operation to the CPU. When an erase error occurs, it is set to 1.

The erase status is reset to 0 when cleared.

## CPU Rewrite Mode (Flash Memory Version)

**Program status (SR4)**

The program status informs the operating status of auto write operation to the CPU. When a write error occurs, it is set to 1.

The program status is reset to 0 when cleared.

When an erase command is in error (which occurs if the command entered after the block erase command (20<sub>16</sub>) is not the confirmation command (D0<sub>16</sub>), both the program status and erase status (SR5) are set to 1.

When the program status or erase status = 1, only the following flash commands will be accepted: Read Array, Read Status Register, and Clear Status Register.

Also, in one of the following cases, both SR4 and SR5 are set to 1 (command sequence error):

- (1) When the valid command is not entered correctly
- (2) When the data entered in the second bus cycle of lock bit program (77<sub>16</sub>/D0<sub>16</sub>), block erase (20<sub>16</sub>/D0<sub>16</sub>), or erase all unlock blocks (A7<sub>16</sub>/D0<sub>16</sub>) is not the D0<sub>16</sub> or FF<sub>16</sub>. However, if FF<sub>16</sub> is entered, read array is assumed and the command that has been set up in the first bus cycle is canceled.

**Block status after program (SR3)**

If excessive data is written (phenomenon whereby the memory cell becomes depressed which results in data not being read correctly), "1" is set for the program status after-program at the end of the page write operation. In other words, when writing ends successfully, "80<sub>16</sub>" is output; when writing fails, "90<sub>16</sub>" is output; and when excessive data is written, "88<sub>16</sub>" is output.

**Table 1.22.2. Definition of each bit in status register**

Each bit of SRD	Status name	Definition	
		"1"	"0"
SR7 (bit7)	Write state machine (WSM) status	Ready	Busy
SR6 (bit6)	Reserved	-	-
SR5 (bit5)	Erase status	Terminated in error	Terminated normally
SR4 (bit4)	Program status	Terminated in error	Terminated normally
SR3 (bit3)	Block status after program	Terminated in error	Terminated normally
SR2 (bit2)	Reserved	-	-
SR1 (bit1)	Reserved	-	-
SR0 (bit0)	Reserved	-	-

**Full Status Check**

By performing full status check, it is possible to know the execution results of erase and program operations. Figure 1.22.8 shows a full status check flowchart and the action to be taken when each error occurs.

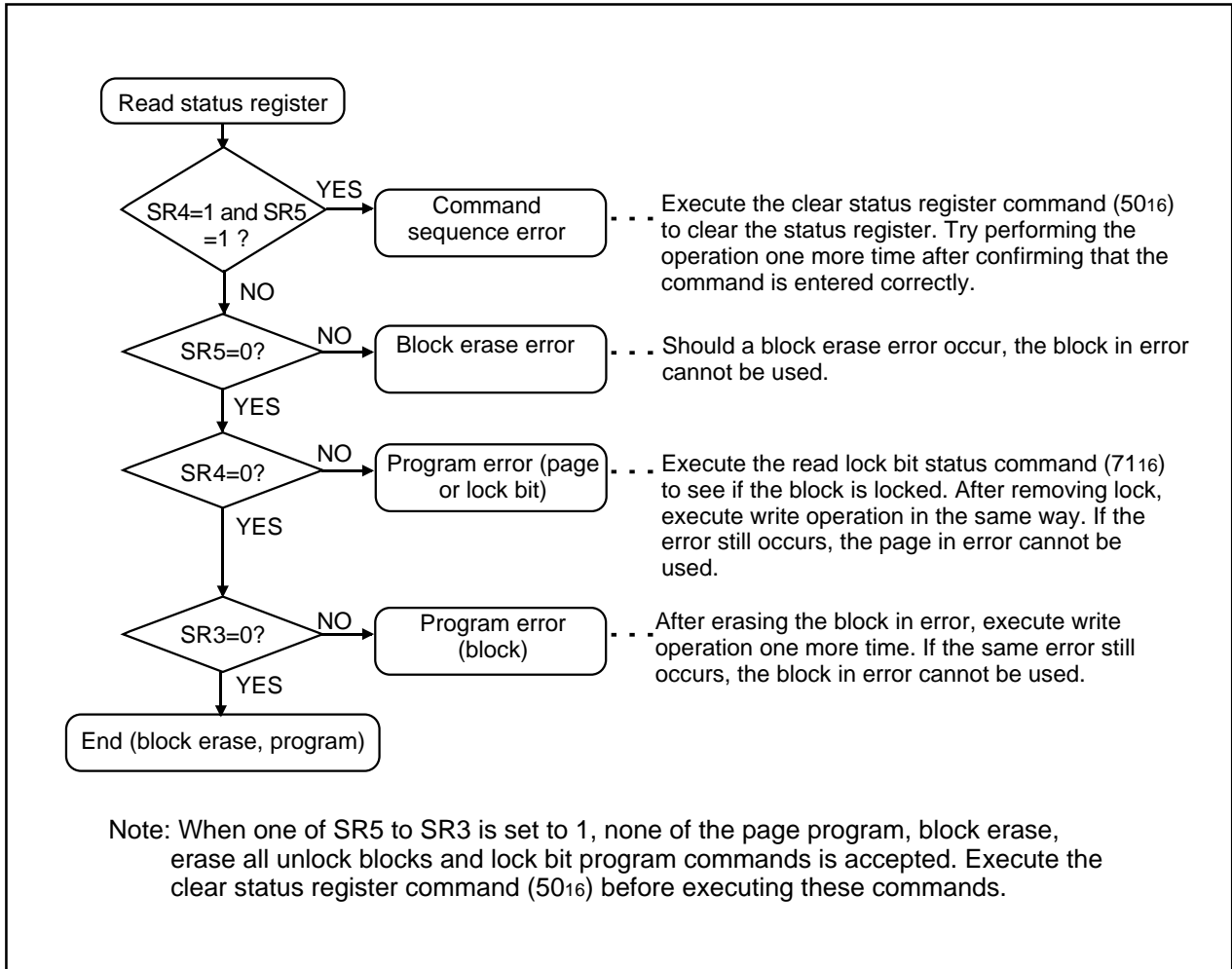


Figure 1.22.8. Full status check flowchart and remedial procedure for errors

## Functions To Inhibit Rewriting Flash Memory Version

To prevent the contents of the flash memory version from being read out or rewritten easily, the device incorporates a ROM code protect function for use in parallel I/O mode and an ID code check function for use in standard serial I/O mode.

### ROM code protect function

The ROM code protect function is used to prohibit reading out or modifying the contents of the flash memory during parallel I/O mode and is set by using the ROM code protect control address register (0FFFFFF<sub>16</sub>). Figure 1.23.1 shows the ROM code protect control address (0FFFFFF<sub>16</sub>). (This address exists in the user ROM area.)

If one of the pair of ROM code protect bits is set to 0, ROM code protect is turned on, so that the contents of the flash memory version are protected against readout and modification. ROM code protect is implemented in two levels. If level 2 is selected, the flash memory is protected even against readout by a shipment inspection LSI tester, etc. When an attempt is made to select both level 1 and level 2, level 2 is selected by default.

If both of the two ROM code protect reset bits are set to "00," ROM code protect is turned off, so that the contents of the flash memory version can be read out or modified. Once ROM code protect is turned on, the contents of the ROM code protect reset bits cannot be modified in parallel I/O mode. Use the serial I/O or some other mode to rewrite the contents of the ROM code protect reset bits.

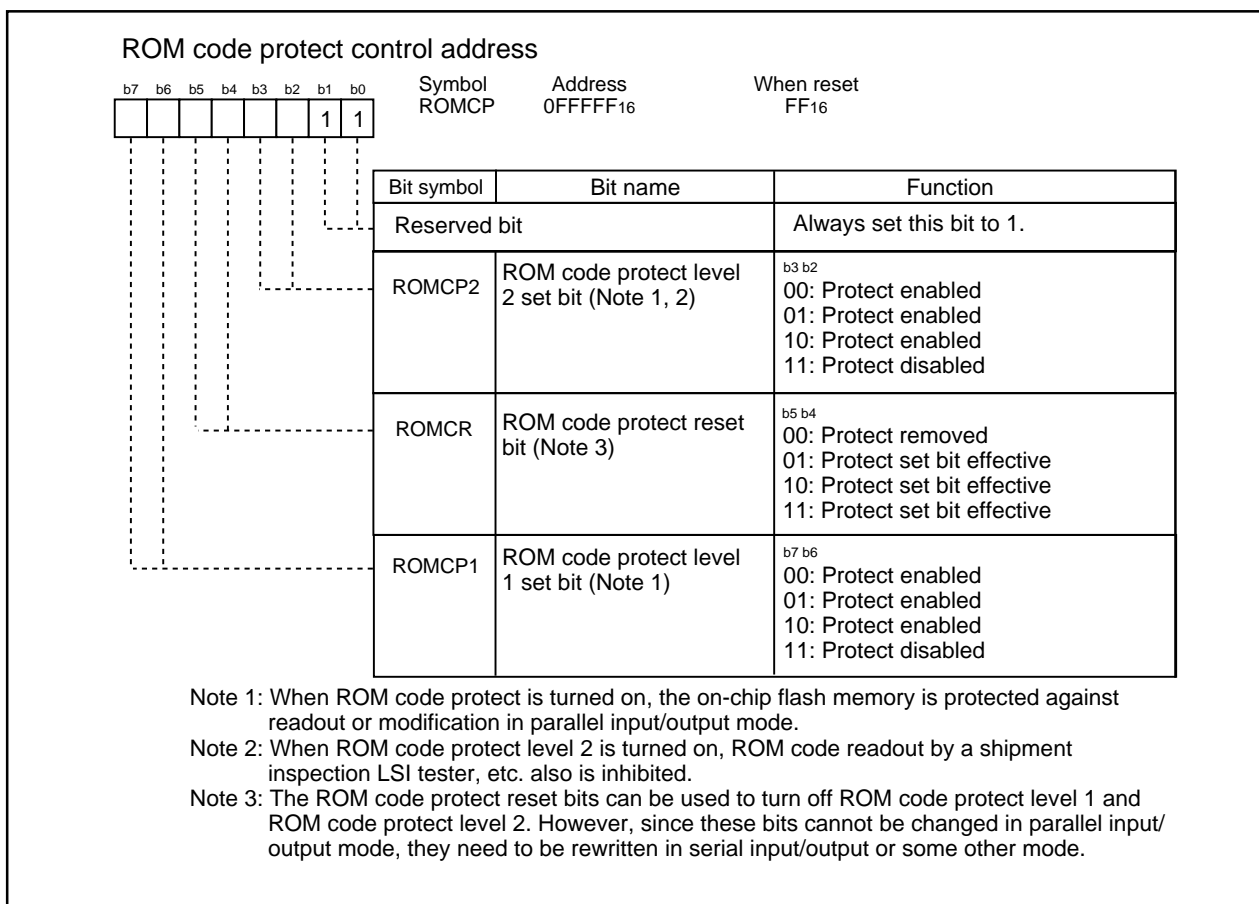


Figure 1.23.1. ROM code protect control address



## ID Code Check Function

Use this function in standard serial I/O mode. When the contents of the flash memory are not blank, the ID code sent from the peripheral unit is compared with the ID code written in the flash memory to see if they match. If the ID codes do not match, the commands sent from the peripheral unit are not accepted. The ID code consists of 8-bit data, the areas of which, beginning with the first byte, are 0FFFD<sub>16</sub>, 0FFFE<sub>316</sub>, 0FFFEB<sub>16</sub>, 0FFFEF<sub>16</sub>, 0FFFF<sub>316</sub>, 0FFFF7<sub>16</sub>, and 0FFFFB<sub>16</sub>. Write a program which has had the ID code preset at these addresses to the flash memory.

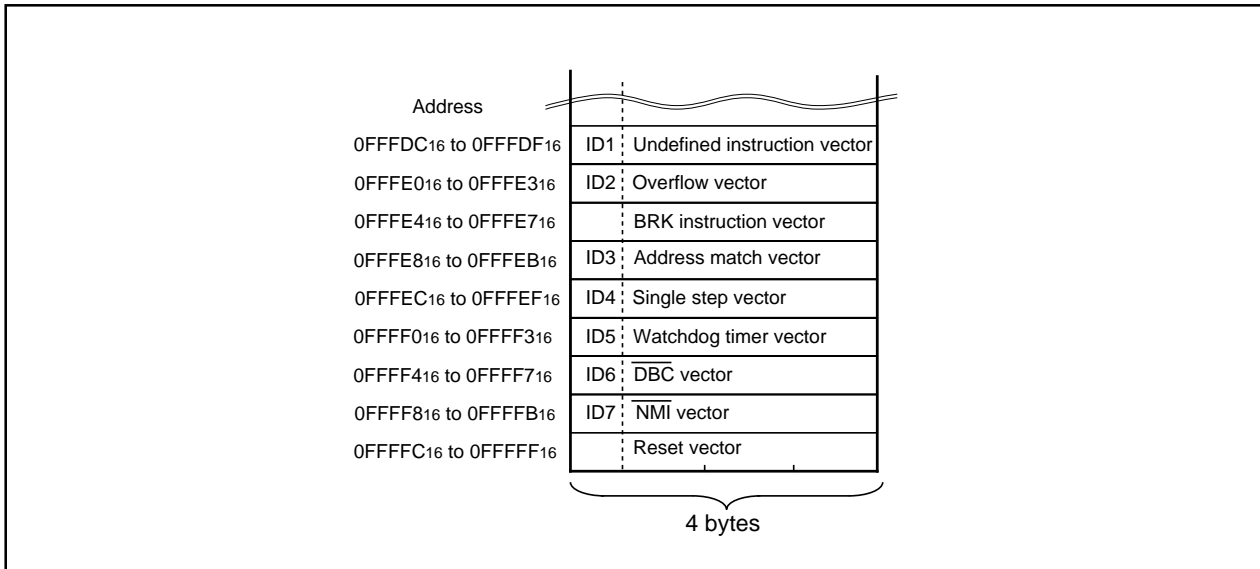


Figure 1.23.2. ID code store addresses

## Parallel I/O Mode

The parallel I/O mode inputs and outputs the software commands, addresses and data needed to operate (read, program, erase, etc.) the internal flash memory. This I/O is parallel.

Use an exclusive programmer supporting M16C/62A (80-pin flash memory version).

Refer to the instruction manual of each programmer maker for the details of use.

## User ROM and Boot ROM Areas

In parallel I/O mode, the user ROM and boot ROM areas shown in Figure 1.21.1 can be rewritten. Both areas of flash memory can be operated on in the same way.

Program and block erase operations can be performed in the user ROM area. The user ROM area and its blocks are shown in Figure 1.21.1.

The boot ROM area is 8 Kbytes in size. In parallel I/O mode, it is located at addresses 0FE000<sub>16</sub> through 0FFFFFF<sub>16</sub>. Make sure program and block erase operations are always performed within this address range. (Access to any location outside this address range is prohibited.)

In the boot ROM area, an erase block operation is applied to only one 8 Kbyte 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, you do not need to write to the boot ROM area.

Appendix Standard Serial I/O Mode (Flash Memory Version)

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

Pin	Name	I/O	Description
Vcc,Vss	Power input		Apply program/erase protection voltage to Vcc pin and 0 V to Vss pin.
CNVss (BYTE)	CNVss	I	Connect to Vcc pin.
$\overline{\text{RESET}}$	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.
P00 to P07	Input port P0	I	Input "H" or "L" level signal or open.
P20 to P27	Input port P2	I	Input "H" or "L" level signal or open.
P30 to P37	Input port P3	I	Input "H" or "L" level signal or open.
P40 to P43	Input port P4	I	Input "H" or "L" level signal or open.
P51 to P54, P56, P57	Input port P5	I	Input "H" or "L" level signal or open.
P50	$\overline{\text{CE}}$ input	I	Input "H" level signal.
P55	$\overline{\text{EPM}}$ input	I	Input "L" level signal.
P60 to P63	Input port P6	I	Input "H" or "L" level signal or open.
P64	BUSY output	O	Standard serial I/O mode 1: BUSY signal output pin Standard serial I/O mode 2: Monitors the boot program operation check signal output pin.
P65	SCLK input	I	Standard serial I/O mode 1: Serial clock input pin Standard serial I/O mode 2: Input "L".
P66	RxD input	I	Serial data input pin
P67	TxD output	O	Serial data output pin
P70 to P77	Input port P7	I	Input "H" or "L" level signal or open.
P80 to P84, P86, P87	Input port P8	I	Input "H" or "L" level signal or open.
P85	$\overline{\text{NMI}}$ input	I	Connect this pin to Vcc.
P90, P92 to P97	Input port P9	I	Input "H" or "L" level signal or open.
P100 to P107	Input port P10	I	Input "H" or "L" level signal or open.

Appendix Standard Serial I/O Mode (Flash Memory Version)

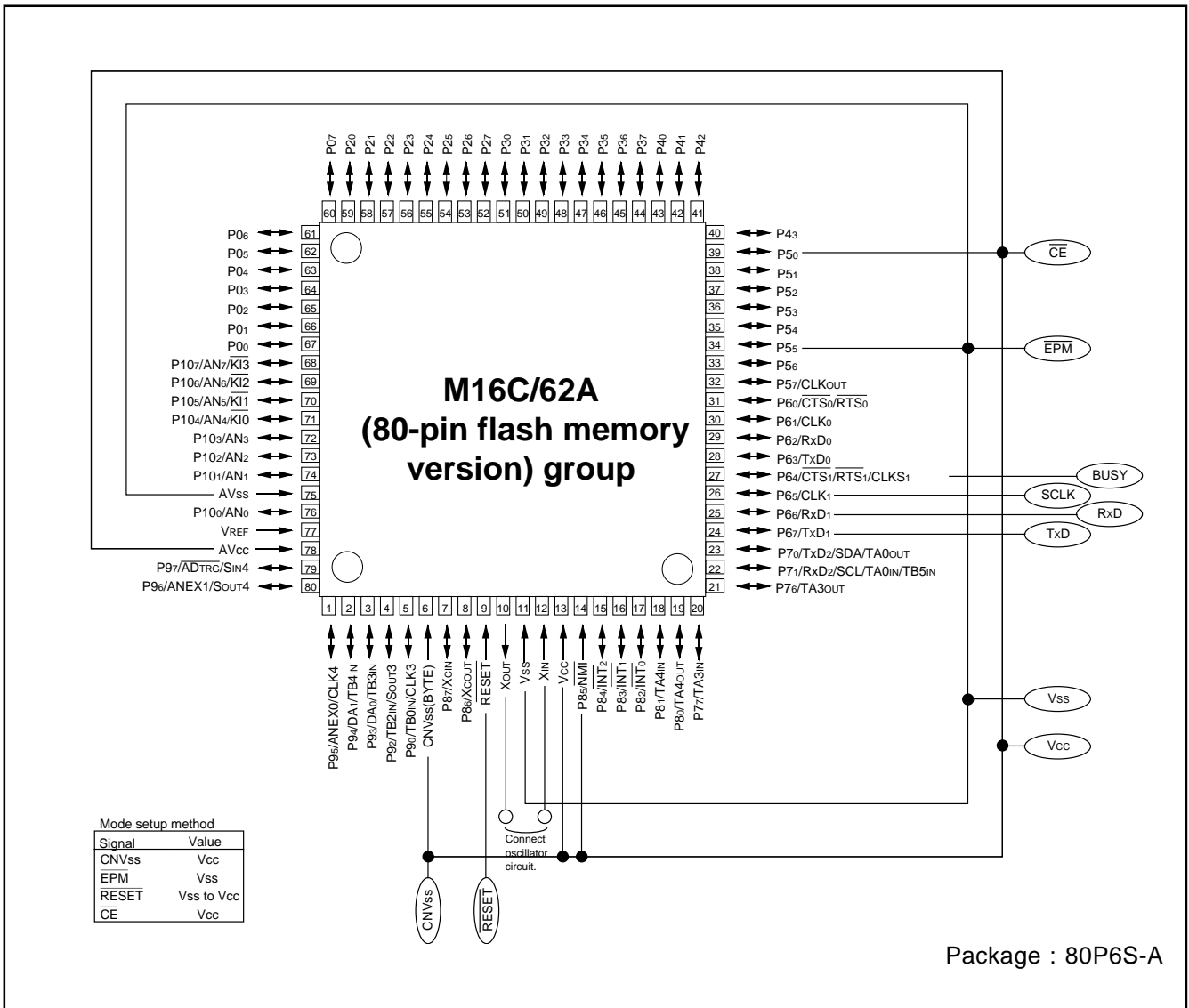


Figure 1.25.1. Pin connections for serial I/O mode

### Standard serial I/O mode

The standard serial I/O mode inputs and outputs the software commands, addresses and data needed to operate (read, program, erase, etc.) the internal flash memory. This I/O is serial. There are actually two standard serial I/O modes: mode 1, which is clock synchronized, and mode 2, which is asynchronous. Both modes require a purpose-specific peripheral unit.

The standard serial I/O mode is different from the parallel I/O mode in that the CPU controls flash memory rewrite (uses the CPU's rewrite mode), rewrite data input and so forth. It is started when the reset is released, which is done when the P50 ( $\overline{CE}$ ) pin is "H" level, the P55 ( $\overline{EPM}$ ) pin "L" level and the CNVss pin "H" level. (In the ordinary command mode, set CNVss pin to "L" level.)

This control program is written in the boot ROM area when the product is shipped from Mitsubishi. Accordingly, make note of the fact that the standard serial I/O mode cannot be used if the boot ROM area is rewritten in the parallel I/O mode. Figure 1.25.1 shows the pin connections for the standard serial I/O mode. Serial data I/O uses UART1 and transfers the data serially in 8-bit units. Standard serial I/O switches between mode 1 (clock synchronized) and mode 2 (clock asynchronous) according to the level of CLK1 pin when the reset is released.

To use standard serial I/O mode 1 (clock synchronized), set the CLK1 pin to "H" level and release the reset. The operation uses the four UART1 pins CLK1, RxD1, TxD1 and RTS1 (BUSY). The CLK1 pin is the transfer clock input pin through which an external transfer clock is input. The TxD1 pin is for CMOS output. The RTS1 (BUSY) pin outputs an "L" level when ready for reception and an "H" level when reception starts.

To use standard serial I/O mode 2 (clock asynchronous), set the CLK1 pin to "L" level and release the reset. The operation uses the two UART1 pins RxD1 and TxD1.

In the standard serial I/O mode, only the user ROM area indicated in Figure 1.21.1 can be rewritten. The boot ROM cannot.

In the standard serial I/O mode, a 7-byte ID code is used. When there is data in the flash memory, commands sent from the peripheral unit are not accepted unless the ID code matches.

### Overview of standard serial I/O mode 1 (clock synchronized)

In standard serial I/O mode 1, software commands, addresses and data are input and output between the MCU and peripheral units (serial programmer, etc.) using 4-wire clock-synchronized serial I/O (UART1). Standard serial I/O mode 1 is engaged by releasing the reset with the P65 (CLK1) pin "H" level.

In reception, software commands, addresses and program data are synchronized with the rise of the transfer clock that is input to the CLK1 pin, and are then input to the MCU via the RxD1 pin. In transmission, the read data and status are synchronized with the fall of the transfer clock, and output from the TxD1 pin.

The TxD1 pin is for CMOS output. Transfer is in 8-bit units with LSB first.

When busy, such as during transmission, reception, erasing or program execution, the RTS1 (BUSY) pin is "H" level. Accordingly, always start the next transfer after the RTS1 (BUSY) pin is "L" level.

Also, data and status registers in memory can be read after inputting software commands. Status, such as the operating state of the flash memory or whether a program or erase operation ended successfully or not, can be checked by reading the status register. Here following are explained software commands, status registers, etc.

Appendix Standard Serial I/O Mode 1 (Flash Memory Version)

**Software Commands**

Table 1.25.1 lists software commands. In the standard serial I/O mode 1, erase operations, programs and reading are controlled by transferring software commands via the RxD1 pin. Software commands are explained here below.

**Table 1.25.1. Software commands (Standard serial I/O mode 1)**

	Control command	1st byte transfer	2nd byte	3rd byte	4th byte	5th byte	6th byte		When ID is not verified
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	Block erase	20 <sub>16</sub>	Address (middle)	Address (high)	D0 <sub>16</sub>				Not acceptable
4	Erase all unlocked blocks	A7 <sub>16</sub>	D0 <sub>16</sub>						Not acceptable
5	Read status register	70 <sub>16</sub>	SRD output	SRD1 output					Acceptable
6	Clear status register	50 <sub>16</sub>							Not acceptable
7	Read lock bit status	71 <sub>16</sub>	Address (middle)	Address (high)	Lock bit data output				Not acceptable
8	Lock bit program	77 <sub>16</sub>	Address (middle)	Address (high)	D0 <sub>16</sub>				Not acceptable
9	Lock bit enable	7A <sub>16</sub>							Not acceptable
10	Lock bit disable	75 <sub>16</sub>							Not acceptable
11	ID check function	F5 <sub>16</sub>	Address (low)	Address (middle)	Address (high)	ID size	ID1	To ID7	Acceptable
12	Download function	FA <sub>16</sub>	Size (low)	Size (high)	Check-sum	Data input	To required number of times		Not acceptable
13	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 ROM area output function	FC <sub>16</sub>	Address (middle)	Address (high)	Data output	Data output	Data output	Data output to 259th byte	Not acceptable
15	Read check data	FD <sub>16</sub>	Check data (low)	Check data (high)					Not acceptable

Note 1: Shading indicates transfer from flash memory microcomputer to peripheral unit. All other data is transferred from the peripheral unit to the flash memory microcomputer.

Note 2: SRD refers to status register data. SRD1 refers to status register 1 data.

Note 3: All commands can be accepted when the flash memory is totally blank.

### 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) Transfer the "FF<sub>16</sub>" command code with the 1st byte.
- (2) Transfer addresses A<sub>8</sub> to A<sub>15</sub> and A<sub>16</sub> to A<sub>23</sub> with the 2nd and 3rd bytes respectively.
- (3) From the 4th byte onward, data (D<sub>0</sub>–D<sub>7</sub>) for the page (256 bytes) specified with addresses A<sub>8</sub> to A<sub>23</sub> will be output sequentially from the smallest address first in sync with the fall of the clock.

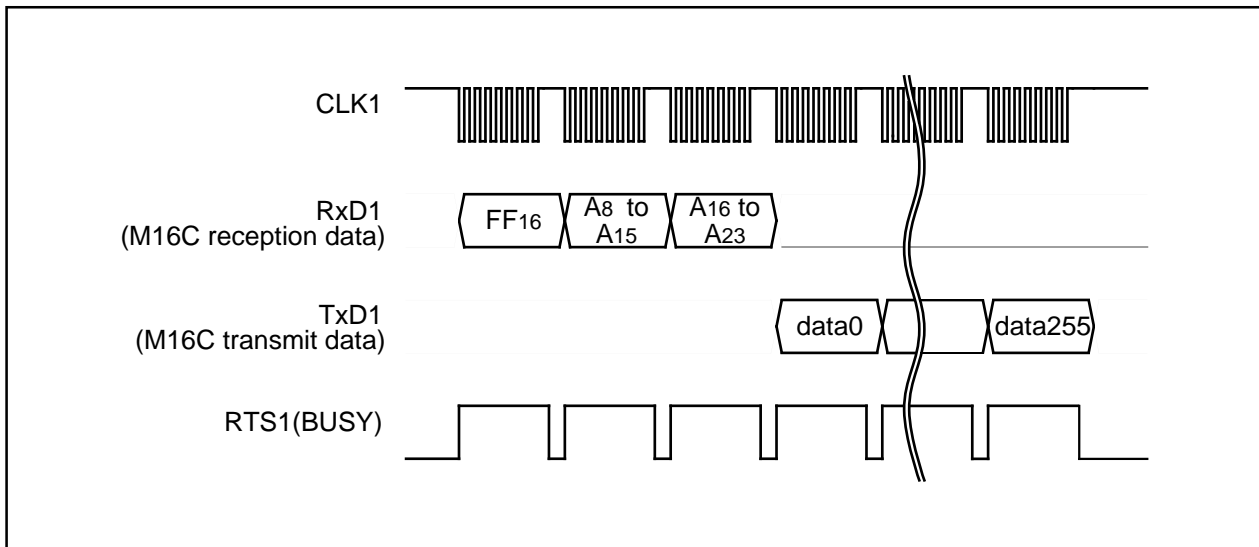


Figure 1.25.2. Timing for page read

### Read Status Register Command

This command reads status information. When the "70<sub>16</sub>" command code is sent with the 1st byte, the contents of the status register (SRD) specified with the 2nd byte and the contents of status register 1 (SRD1) specified with the 3rd byte are read.

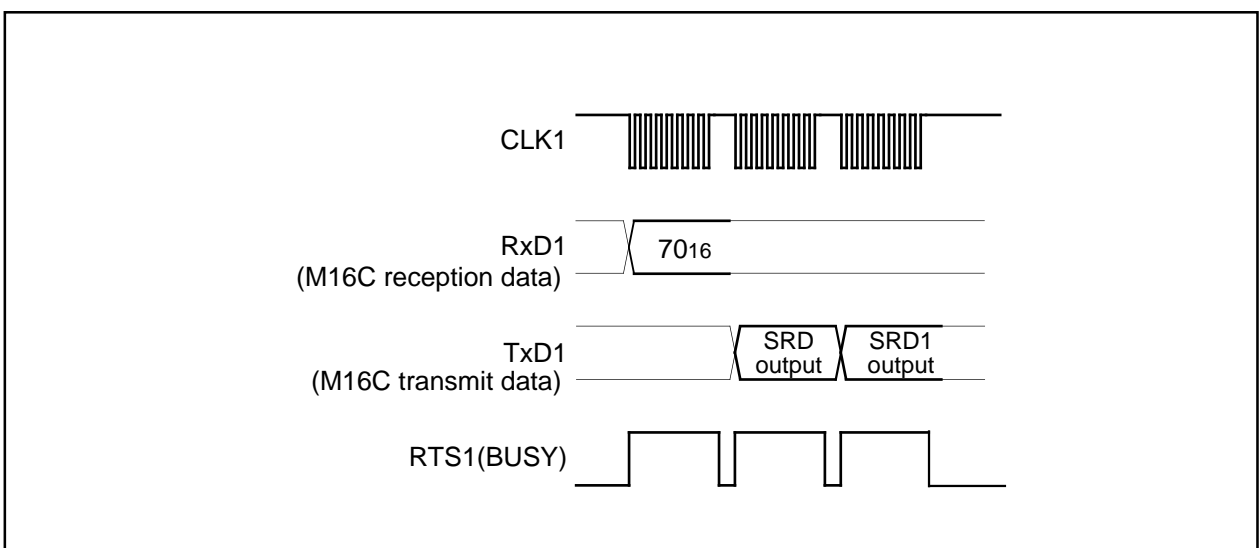
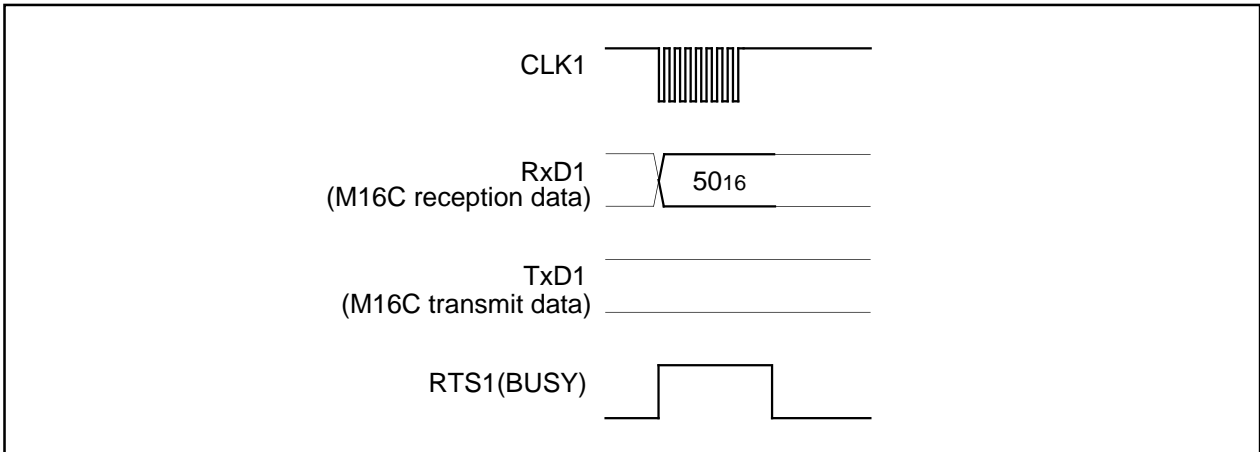


Figure 1.25.3. Timing for reading the status register



### Clear Status Register Command

This command clears the bits (SR3–SR5) which are set when the status register operation ends in error. When the “50<sub>16</sub>” command code is sent with the 1st byte, the aforementioned bits are cleared. When the clear status register operation ends, the RTS<sub>1</sub> (BUSY) signal changes from the “H” to the “L” level.



**Figure 1.25.4. Timing for clearing the status register**

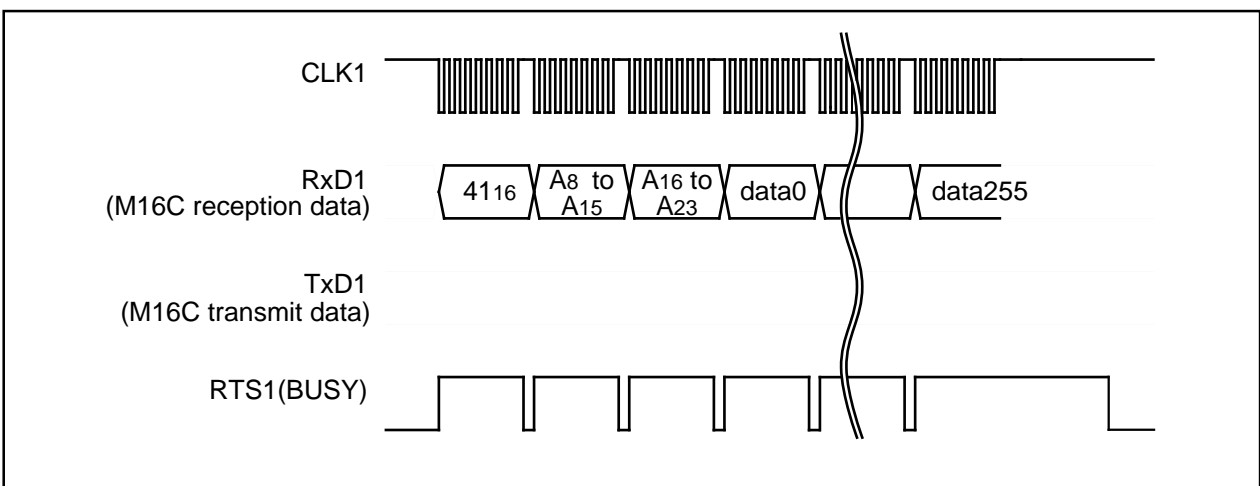
### Page Program Command

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) Transfer the “41<sub>16</sub>” command code with the 1st byte.
- (2) Transfer addresses A<sub>8</sub> to A<sub>15</sub> and A<sub>16</sub> to A<sub>23</sub> with the 2nd and 3rd bytes respectively.
- (3) From the 4th byte onward, as write data (D<sub>0</sub>–D<sub>7</sub>) for the page (256 bytes) specified with addresses A<sub>8</sub> to A<sub>23</sub> is input sequentially from the smallest address first, that page is automatically written.

When reception setup for the next 256 bytes ends, the RTS<sub>1</sub> (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.

Each block can be write-protected with the lock bit. For more information, see the section on the data protection function. Additional writing is not allowed with already programmed pages.



**Figure 1.25.5. Timing for the page program**

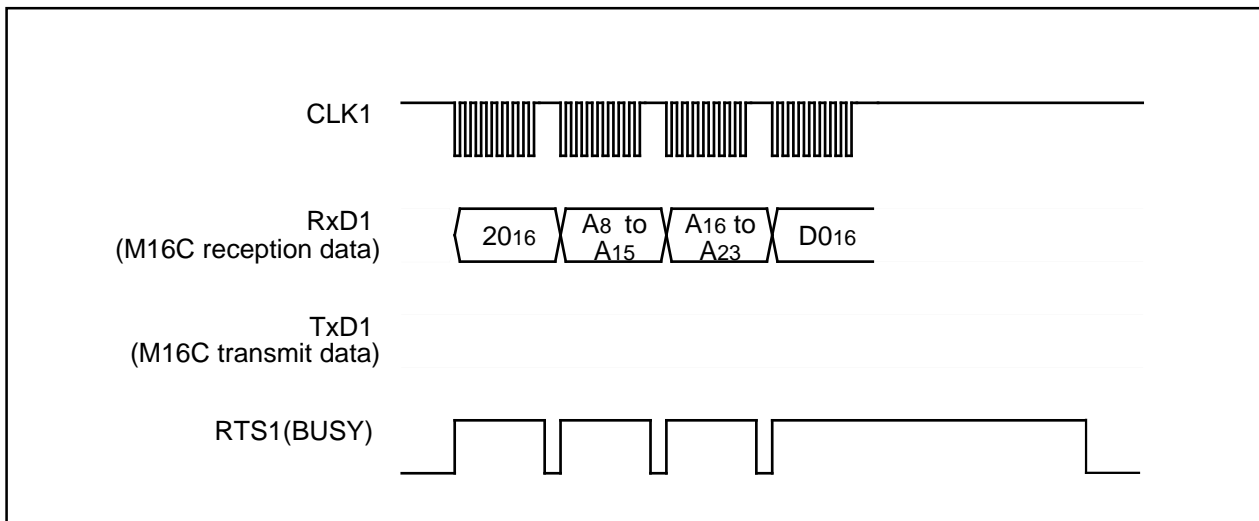
### Block Erase Command

This command erases the data in the specified block. Execute the block erase command as explained here following.

- (1) Transfer the "20<sub>16</sub>" command code with the 1st byte.
- (2) Transfer addresses A<sub>8</sub> to A<sub>15</sub> and A<sub>16</sub> to A<sub>23</sub> with the 2nd and 3rd bytes respectively.
- (3) Transfer the verify command code "D0<sub>16</sub>" with the 4th byte. With the verify command code, the erase operation will start for the specified block in the flash memory. Write the highest address of the specified block for addresses A<sub>8</sub> to A<sub>23</sub>.

When block erasing ends, the RTS<sub>1</sub> (BUSY) signal changes from the "H" to the "L" level. After block erase ends, the result of the block erase operation can be known by reading the status register. For more information, see the section on the status register.

Each block can be erase-protected with the lock bit. For more information, see the section on the data protection function.



**Figure 1.25.6. Timing for block erasing**

## Appendix Standard Serial I/O Mode 1 (Flash Memory Version)

**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) Transfer the "A7<sub>16</sub>" command code with the 1st byte.
- (2) Transfer the verify command code "D0<sub>16</sub>" with the 2nd byte. With the verify command code, the erase operation will start and continue for all blocks in the flash memory.

When block erasing ends, the RTS<sub>1</sub> (BUSY) signal changes from the "H" to the "L" level. The result of the erase operation can be known by reading the status register. Each block can be erase-protected with the lock bit. For more information, see the section on the data protection function.

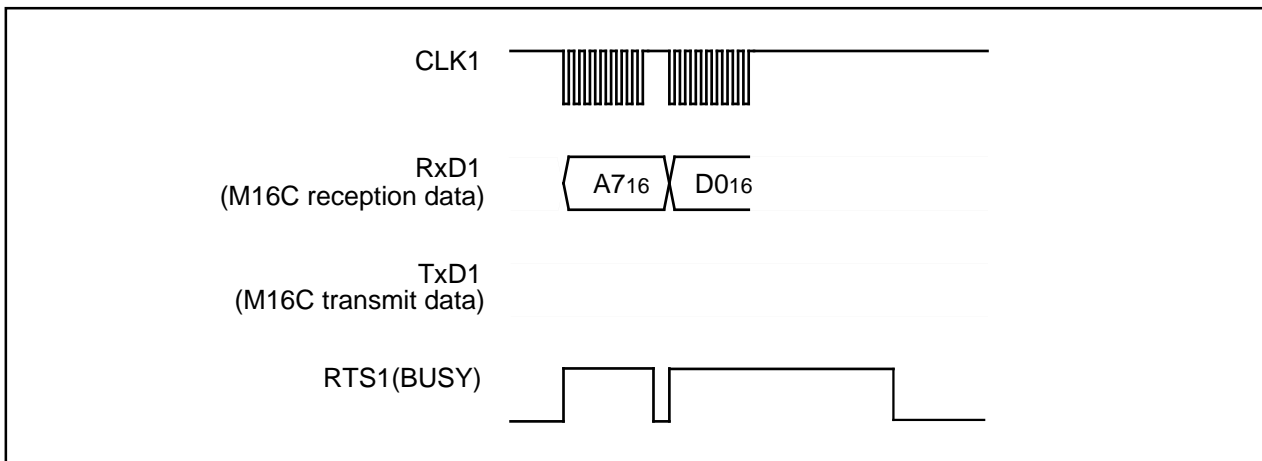


Figure 1.25.7. Timing for erasing all unlocked blocks

**Lock Bit Program Command**

This command writes "0" (lock) for the lock bit of the specified block. Execute the lock bit program command as explained here following.

- (1) Transfer the "77<sub>16</sub>" command code with the 1st byte.
- (2) Transfer addresses A8 to A15 and A16 to A23 with the 2nd and 3rd bytes respectively.
- (3) Transfer the verify command code "D0<sub>16</sub>" with the 4th byte. With the verify command code, "0" is written for the lock bit of the specified block. Write the highest address of the specified block for addresses A8 to A23.

When writing ends, the RTS<sub>1</sub> (BUSY) signal changes from the "H" to the "L" level. Lock bit status can be read with the read lock bit status command. For information on the lock bit function, reset procedure and so on, see the section on the data protection function.

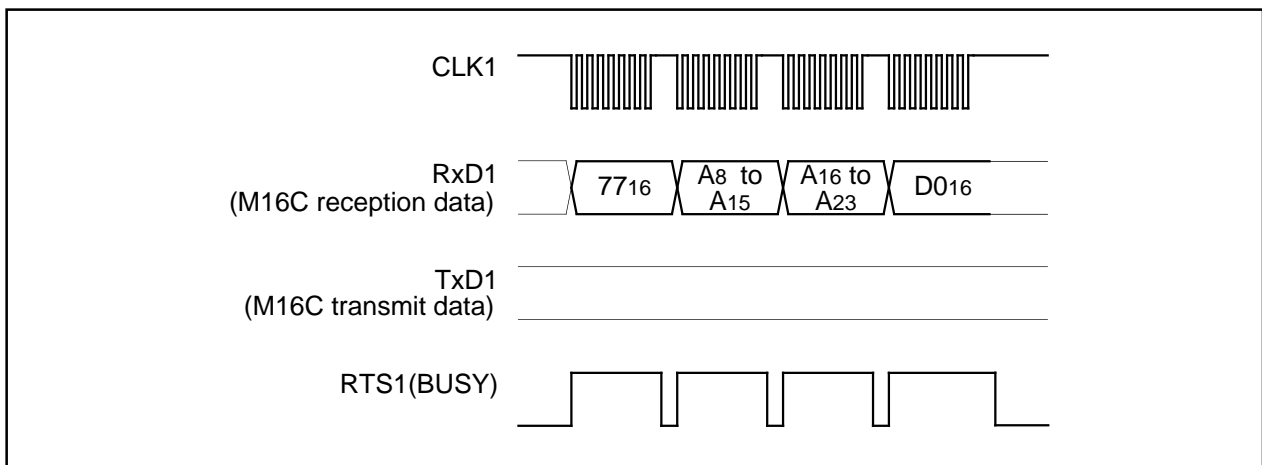


Figure 1.25.8. Timing for the lock bit program

### 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) Transfer the "7116" command code with the 1st byte.
- (2) Transfer addresses A8 to A15 and A16 to A23 with the 2nd and 3rd bytes respectively.
- (3) The lock bit data of the specified block is output with the 4th byte. The lock bit data is the 6th bit (D6) of the output data. Write the highest address of the specified block for addresses A8 to A23.

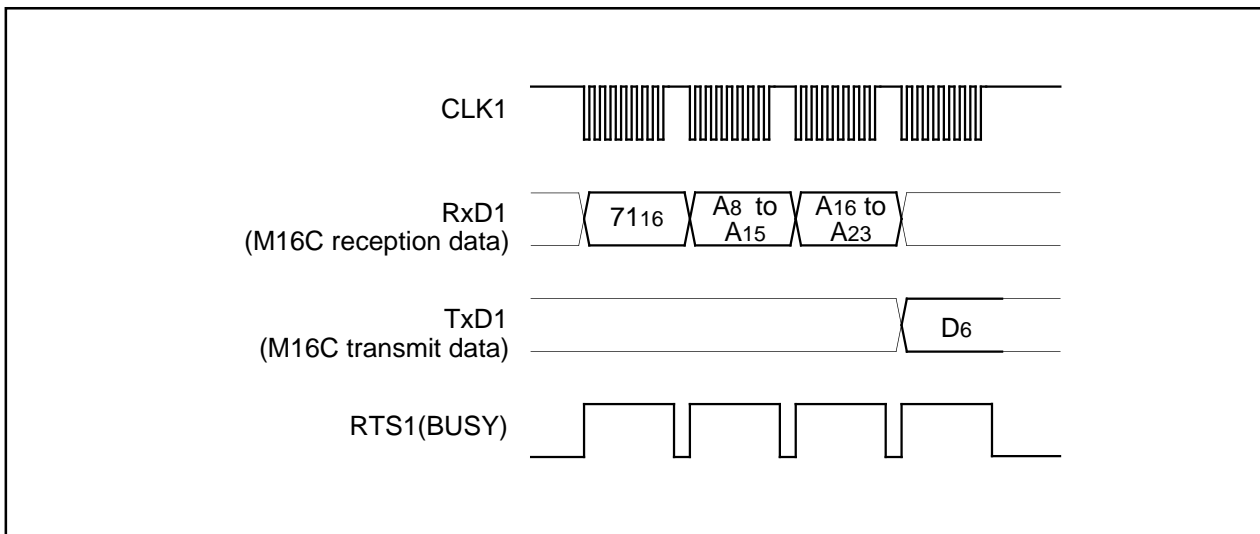


Figure 1.25.9. Timing for reading lock bit status

### Lock Bit Enable Command

This command enables the lock bit in blocks whose bit was disabled with the lock bit disable command. The command code "7A16" is sent with the 1st byte of the serial transmission. This command only enables the lock bit function; it does not set the lock bit itself.

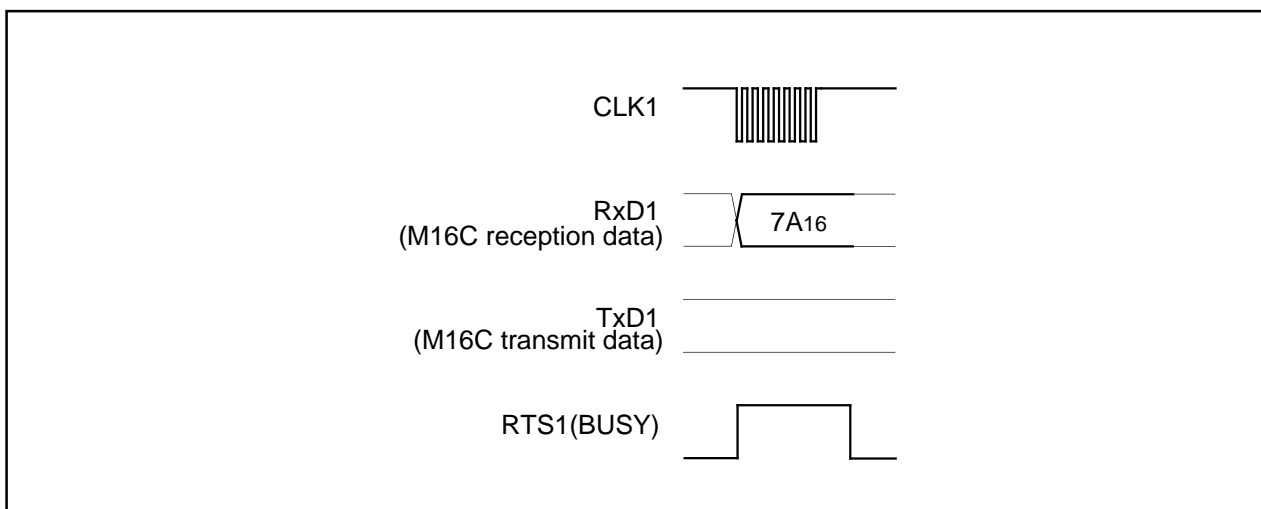


Figure 1.25.10. Timing for enabling the lock bit

### Lock Bit Disable Command

This command disables the lock bit. The command code "7516" is sent with the 1st byte of the serial transmission. This command only disables the lock bit function; it does not set the lock bit itself. However, if an erase command is executed after executing the lock bit disable command, "0" (locked) lock bit data is set to "1" (unlocked) after the erase operation ends. In any case, after the reset is cancelled, the lock bit is enabled.

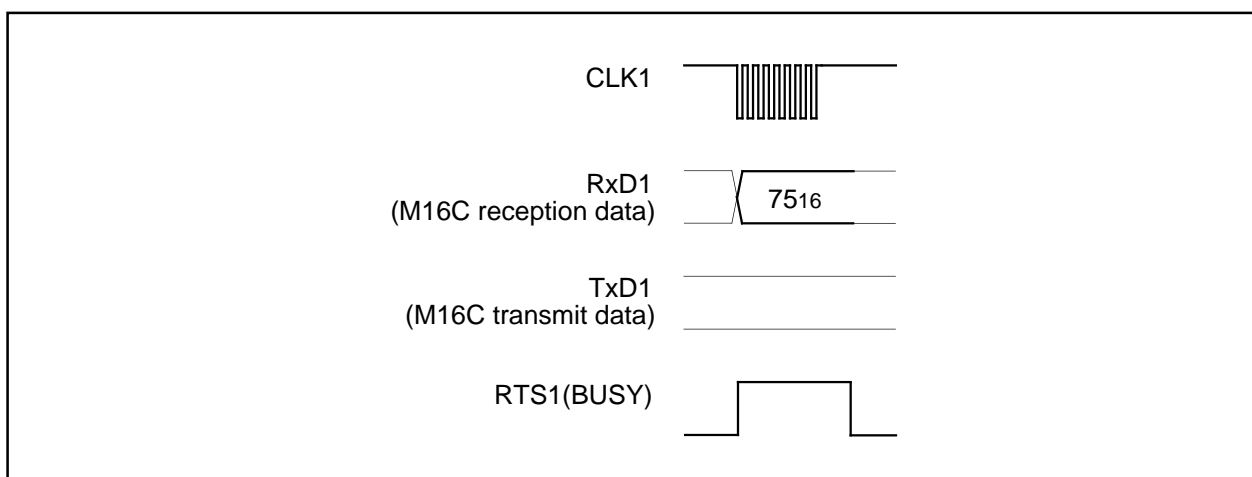


Figure 1.25.11. Timing for disabling the lock bit

### Download Command

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

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

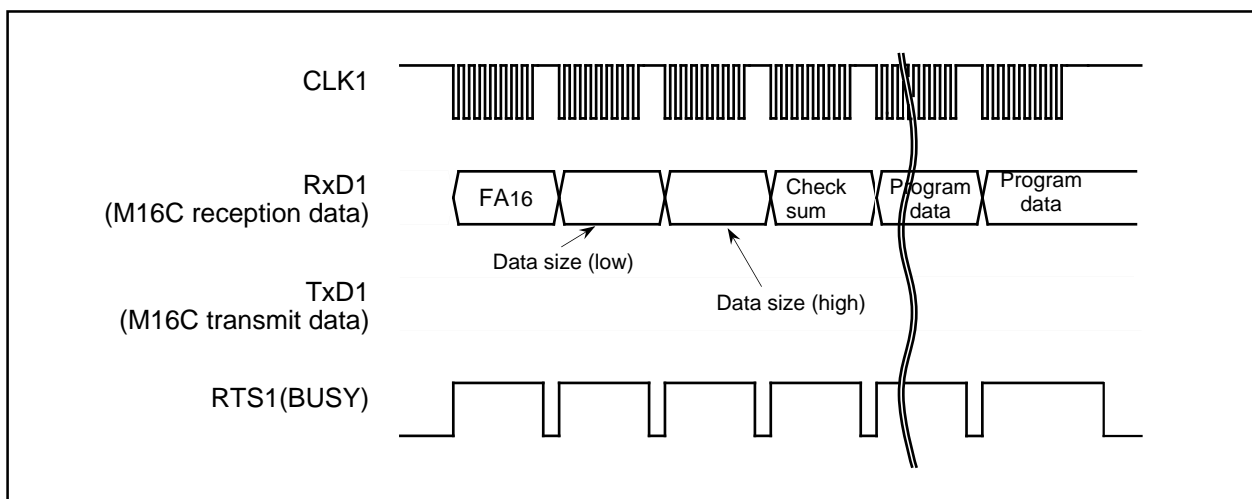


Figure 1.25.12. Timing for download

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

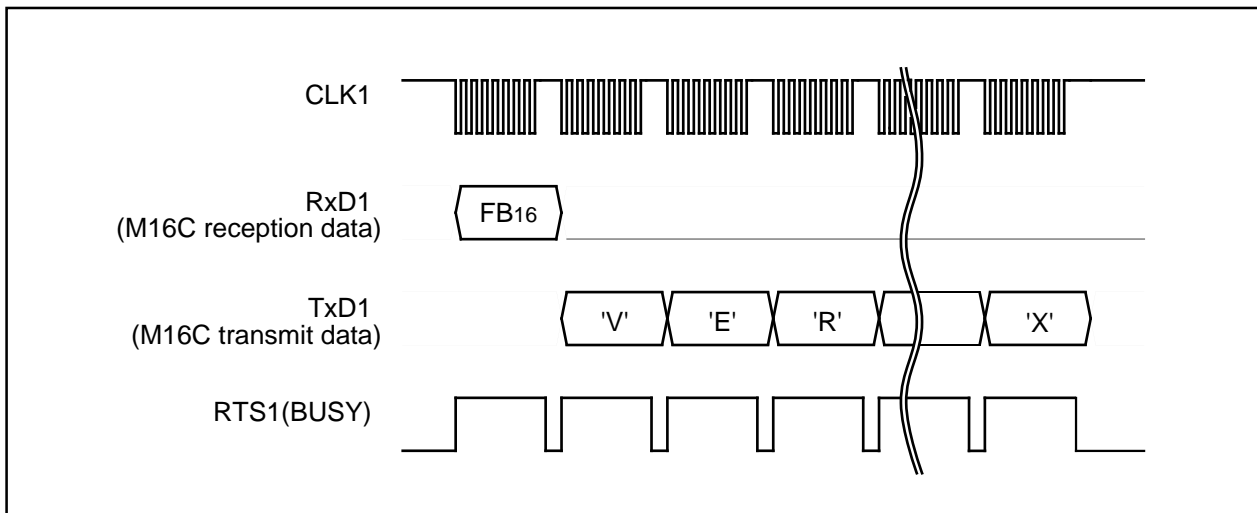


Figure 1.25.13. Timing for version information output

### Boot ROM Area Output Command

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

- (1) Transfer the "FC16" command code with the 1st byte.
- (2) Transfer addresses A8 to A15 and A16 to A23 with the 2nd and 3rd bytes 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 fall of the clock.

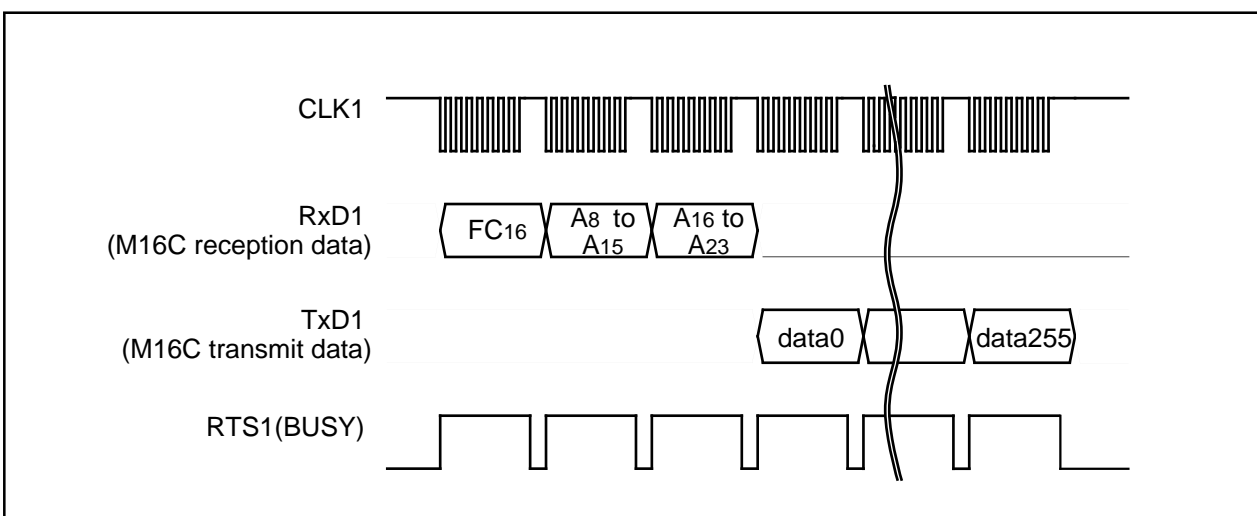
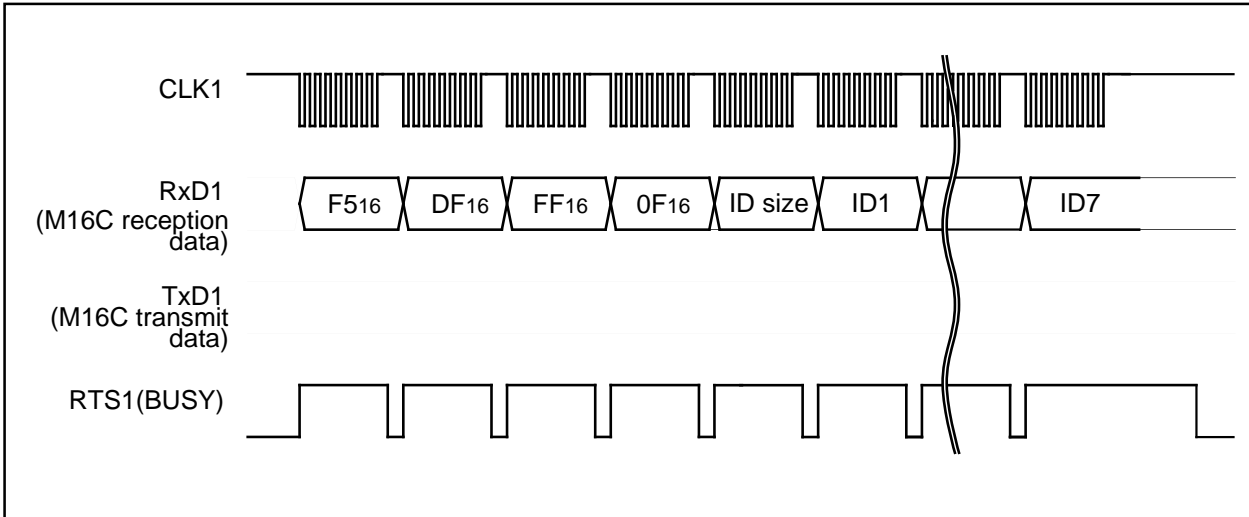


Figure 1.25.14. Timing for boot ROM area output

**ID Check**

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

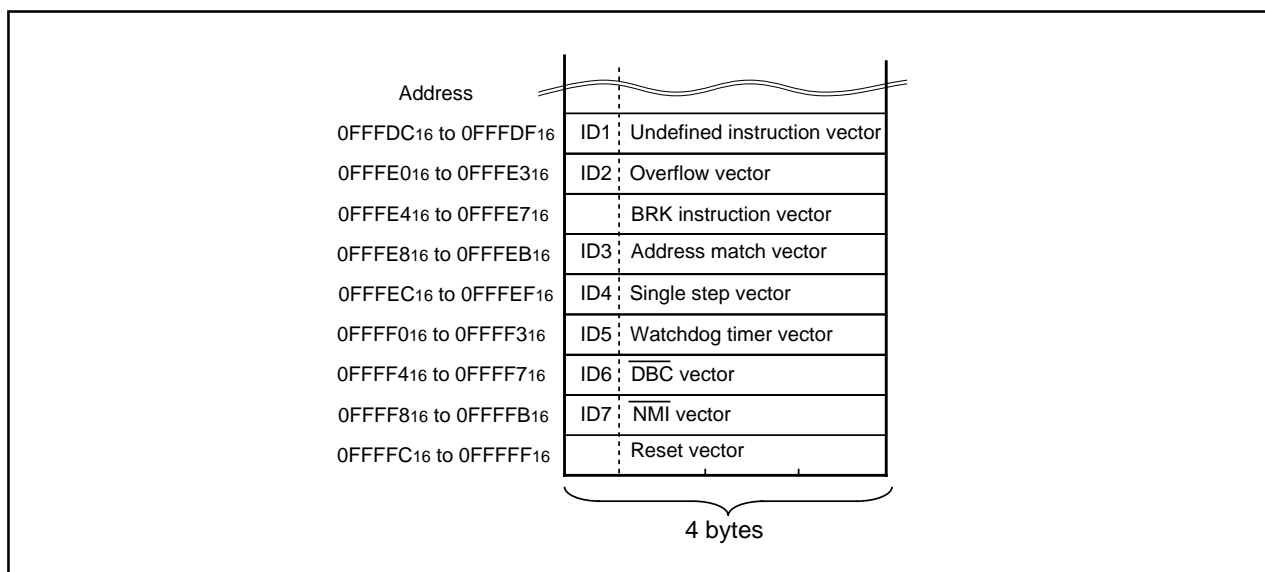
- (1) Transfer the "F5<sub>16</sub>" command code with the 1st byte.
- (2) Transfer addresses A<sub>0</sub> to A<sub>7</sub>, A<sub>8</sub> to A<sub>15</sub> and A<sub>16</sub> to A<sub>23</sub> of the 1st byte of the ID code with the 2nd, 3rd and 4th bytes respectively.
- (3) Transfer the number of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



**Figure 1.25.15. Timing for the ID check**

**ID Code**

When the flash memory is not blank, the ID code sent from the peripheral units 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 peripheral units is not accepted. An ID code contains 8 bits of data. Area is, from the 1st byte, addresses 0FFFD<sub>16</sub>, 0FFFE<sub>316</sub>, 0FFFE<sub>716</sub>, 0FFFE<sub>F16</sub>, 0FFFF<sub>316</sub>, 0FFFF<sub>716</sub> and 0FFFF<sub>B16</sub>. Write a program into the flash memory, which already has the ID code set for these addresses.



**Figure 1.25.16. ID code storage addresses**

### Read Check Data

This command reads the check data that confirms that the write data, which was sent with the page program command, was successfully received.

- (1) Transfer the "FD16" command code with the 1st byte.
- (2) The check data (low) is received with the 2nd byte and the check data (high) with the 3rd.

To use this read check data command, first execute the command and then initialize the check data. Next, execute the page program command the required number of times. After that, when the read check command is executed again, the check data for all of the read data that was sent with the page program command during this time is read. The check data is the result of CRC operation of write data.

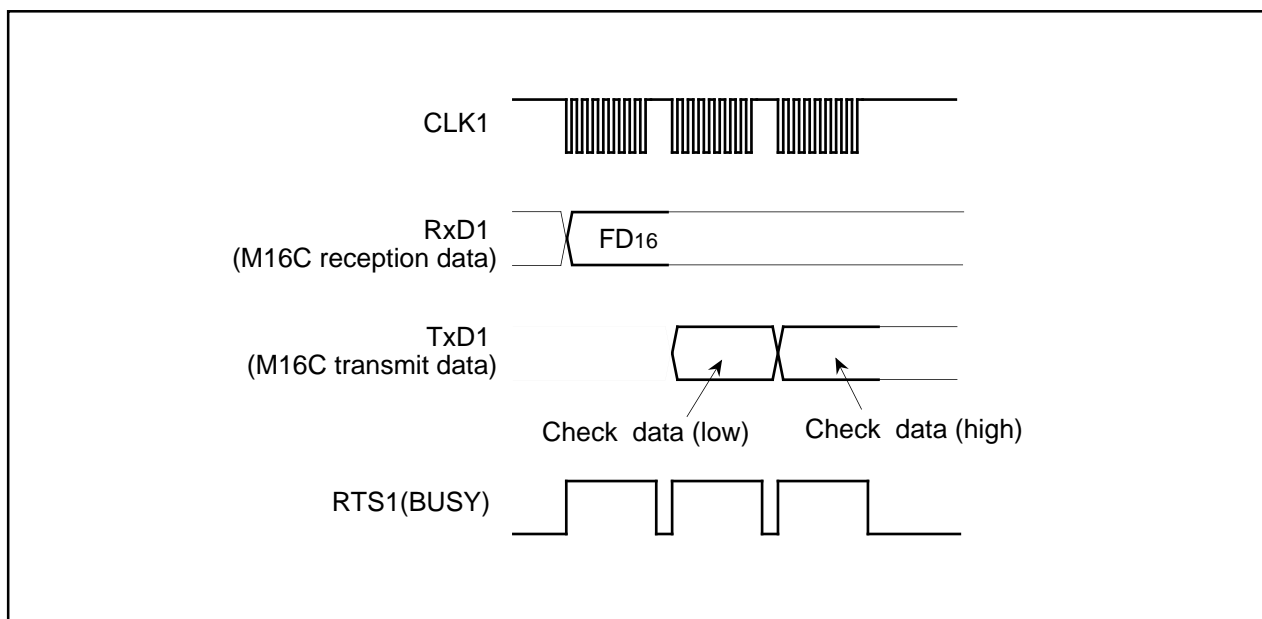


Figure 1.25.17. Timing for the read check data



## Data Protection (Block Lock)

Each of the blocks in Figure 1.25.18 have a nonvolatile lock bit that specifies protection (block lock) against erasing/writing. A block is locked (writing "0" for the lock bit) with the lock bit program command. Also, the lock bit of any block can be read with the read lock bit status command.

Block lock disable/enable is determined by the status of the lock bit itself and execution status of the lock bit disable and lock enable bit commands.

- (1) After the reset has been cancelled and the lock bit enable command executed, the specified block can be locked/unlocked using the lock bit (lock bit data). Blocks with a "0" lock bit data are locked and cannot be erased or written in. On the other hand, blocks with a "1" lock bit data are unlocked and can be erased or written in.
- (2) After the lock bit disable command has been executed, all blocks are unlocked regardless of lock bit data status and can be erased or written in. In this case, lock bit data that was "0" (locked) before the block was erased is set to "1" (unlocked) after erasing, therefore the block is actually unlocked with the lock bit.

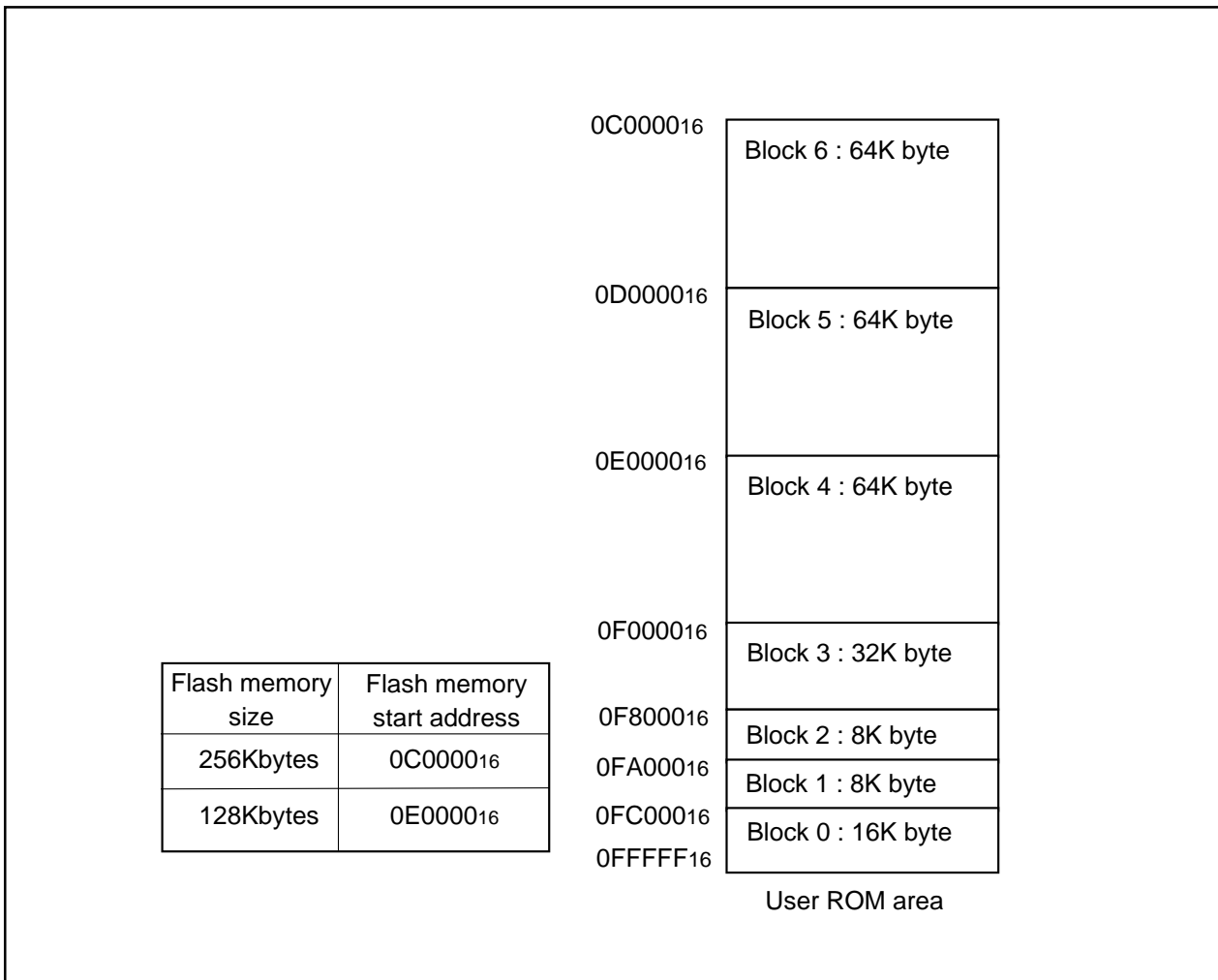


Figure 1.25.18. Blocks in the user area

## 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 1.25.2 gives the definition of each status register bit. After clearing the reset, the status register outputs "80<sub>16</sub>".

**Table 1.25.2. Status register (SRD)**

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

### Write State Machine (WSM) Status (SR7)

The write state machine (WSM) status 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 Status (SR5)

The erase status 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 Status (SR4)

The program status 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".

### Block Status After Program (SR3)

If excessive data is written (phenomenon whereby the memory cell becomes depressed which results in data not being read correctly), "1" is set for the block status after-program at the end of the page write operation. In other words, when writing ends successfully, "80<sub>16</sub>" is output; when writing fails, "90<sub>16</sub>" is output; and when excessive data is written, "88<sub>16</sub>" is output.

If "1" is written for any of the SR5, SR4 or SR3 bits, the page program, block erase, erase all unlocked blocks and lock bit program commands are not accepted. Before executing these commands, execute the clear status register command (50<sub>16</sub>) and clear the status register.

## 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 1.25.3 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 1.25.3. 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)	Check sum 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 Match 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)

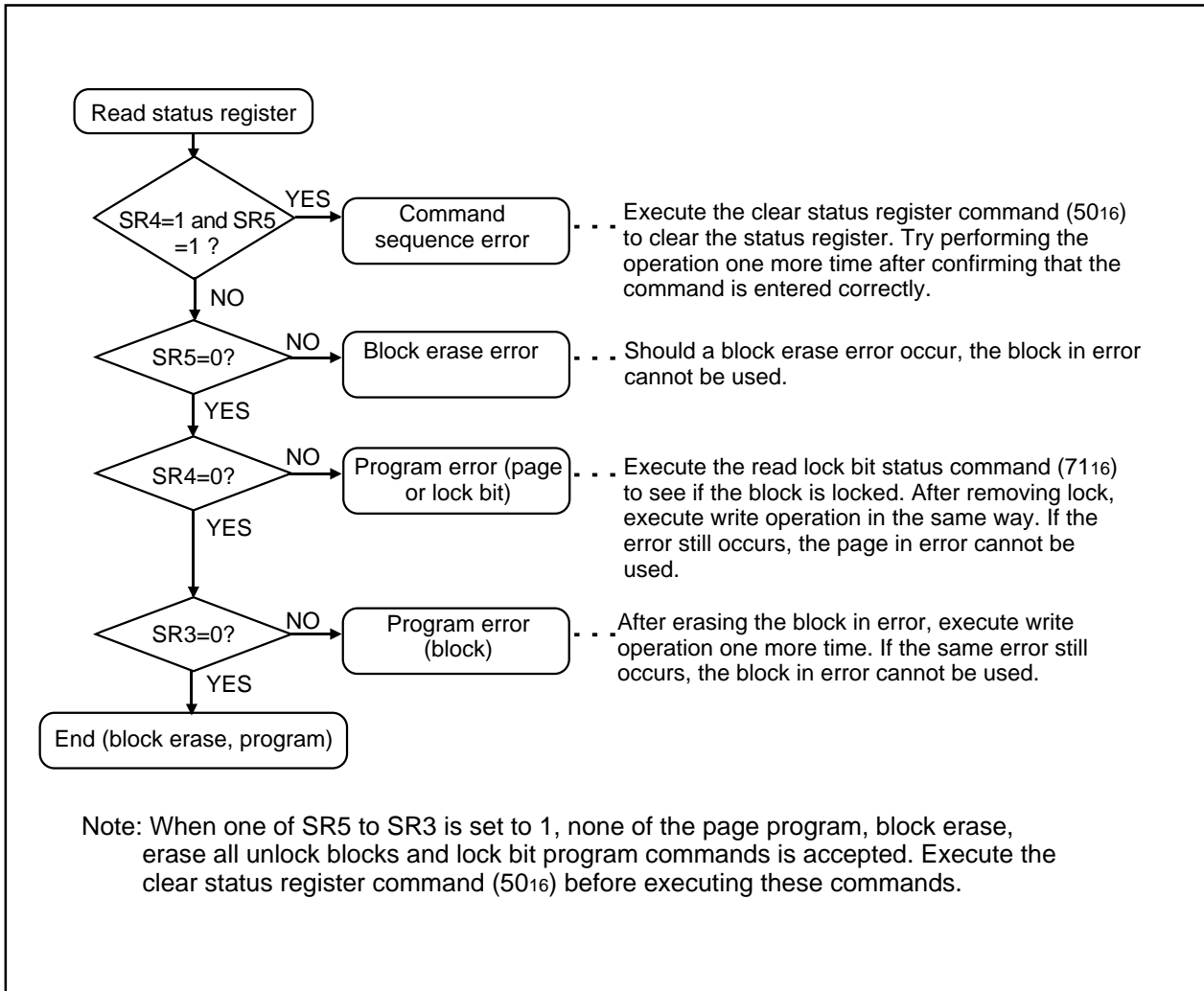
These flags indicate the result of ID checks. Some commands cannot be accepted without an ID check.

### Data Receive 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.

### Full Status Check

Results from executed erase and program operations can be known by running a full status check. Figure 1.25.19 shows a flowchart of the full status check and explains how to remedy errors which occur.



**Figure 1.25.19. Full status check flowchart and remedial procedure for errors**

### Example Circuit Application for The Standard Serial I/O Mode 1

The below figure shows a circuit application for the standard serial I/O mode 1. Control pins will vary according to programmer, therefore see the peripheral unit manual for more information.

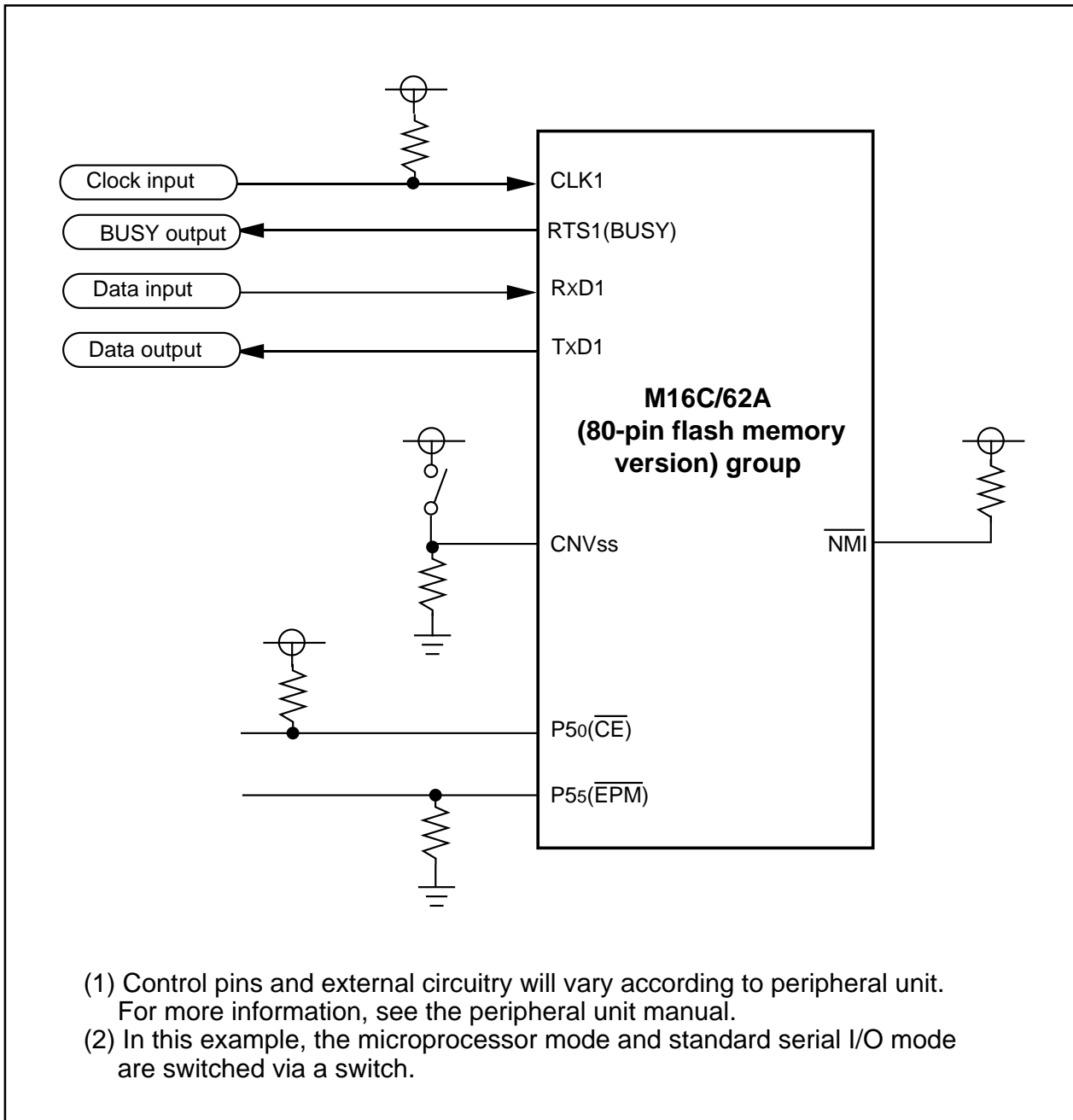


Figure 1.25.20. Example circuit application for the standard serial I/O mode 1

## Overview of standard serial I/O mode 2 (clock asynchronous)

In standard serial I/O mode 2, software commands, addresses and data are input and output between the MCU and peripheral units (serial programmer, etc.) using 2-wire clock-asynchronized serial I/O (UART1). Standard serial I/O mode 2 is engaged by releasing the reset with the P65 (CLK1) pin "L" level.

The TxD1 pin is for CMOS output. Data transfer is in 8-bit units with LSB first, 1 stop bit and parity OFF.

After the reset is released, connections can be established at 9,600 bps when initial communications (Figure 1.25.21) are made with a peripheral unit. However, this requires a main clock with a minimum 2 MHz input oscillation frequency. Baud rate can also be changed from 9,600 bps to 19,200, 38,400 or 57,600 bps by executing software commands. However, communication errors may occur because of the oscillation frequency of the main clock. If errors occur, change the main clock's oscillation frequency and the baud rate.

After executing commands from a peripheral unit that requires time to erase and write data, as with erase and program commands, allow a sufficient time interval or execute the read status command and check how processing ended, before executing the next command.

Data and status registers in memory can be read after transmitting software commands. Status, such as the operating state of the flash memory or whether a program or erase operation ended successfully or not, can be checked by reading the status register. Here following are explained initial communications with peripheral units, how frequency is identified and software commands.

### Initial communications with peripheral units

After the reset is released, the bit rate generator is adjusted to 9,600 bps to match the oscillation frequency of the main clock, by sending the code as prescribed by the protocol for initial communications with peripheral units (Figure 1.25.21).

- (1) Transmit "B016" from a peripheral unit. If the oscillation frequency input by the main clock is 10 or 16 MHz, the MCU with internal flash memory outputs the "B016" check code. If the oscillation frequency is anything other than 10 or 16 MHz, the MCU does not output anything.
- (2) Transmit "0016" from a peripheral unit 16 times. (The MCU with internal flash memory sets the bit rate generator so that "0016" can be successfully received.)
- (3) The MCU with internal flash memory outputs the "B016" check code and initial communications end successfully \*1. Initial communications must be transmitted at a speed of 9,600 bps and a transfer interval of a minimum 15 ms. Also, the baud rate at the end of initial communications is 9,600 bps.

\*1. If the peripheral unit cannot receive "B016" successfully, change the oscillation frequency of the main clock.

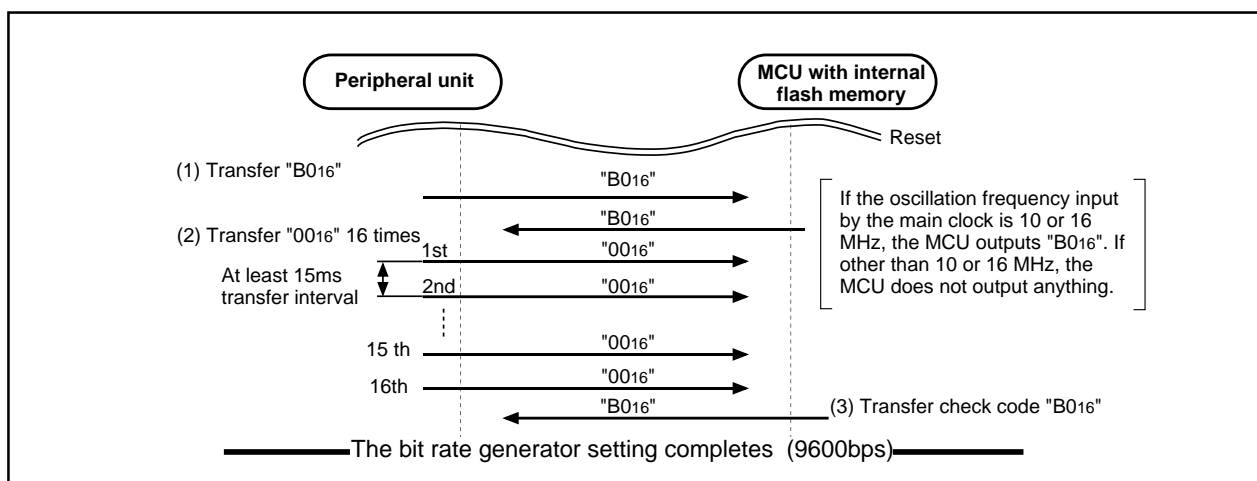


Figure 1.25.21. Peripheral unit and initial communication

## How frequency is identified

When "0016" data is received 16 times from a peripheral unit at a baud rate of 9,600 bps, the value of the bit rate generator is set to match the operating frequency (2 - 16 MHz). The highest speed is taken from the first 8 transmissions and the lowest from the last 8. These values are then used to calculate the bit rate generator value for a baud rate of 9,600 bps.

Baud rate cannot be attained with some operating frequencies. Table 1.25.4 gives the operation frequency and the baud rate that can be attained for.

**Table 1.25.4 Operation frequency and the baud rate**

Operation frequency (MHz)	Baud rate 9,600bps	Baud rate 19,200bps	Baud rate 38,400bps	Baud rate 57,600bps
16MHz	√	√	√	√
12MHz	√	√	√	—
11MHz	√	√	√	—
10MHz	√	√	—	√
8MHz	√	√	—	√
7.3728MHz	√	√	√	√
6MHz	√	√	√	—
5MHz	√	√	—	—
4.5MHz	√	√	—	√
4.194304MHz	√	√	√	—
4MHz	√	√	—	—
3.58MHz	√	√	√	√
3MHz	√	√	√	—
2MHz	√	—	—	—

√ : Communications possible

— : Communications not possible

## Software Commands

Table 1.25.5 lists software commands. In the standard serial I/O mode 2, erase operations, programs and reading are controlled by transferring software commands via the RxD1 pin. Standard serial I/O mode 2 adds four transmission speed commands - 9,600, 19,200, 38,400 and 57,600 bps - to the software commands of standard serial I/O mode 1. Software commands are explained here below.

**Table 1.25.5. Software commands (Standard serial I/O mode 2)**

	Control command	1st byte transfer	2nd byte	3rd byte	4th byte	5th byte	6th byte		When ID is not verified
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	Block erase	20 <sub>16</sub>	Address (middle)	Address (high)	D0 <sub>16</sub>				Not acceptable
4	Erase all unlocked blocks	A7 <sub>16</sub>	D0 <sub>16</sub>						Not acceptable
5	Read status register	70 <sub>16</sub>	SRD output	SRD1 output					Acceptable
6	Clear status register	50 <sub>16</sub>							Not acceptable
7	Read lock bit status	71 <sub>16</sub>	Address (middle)	Address (high)	Lock bit data output				Not acceptable
8	Lock bit program	77 <sub>16</sub>	Address (middle)	Address (high)	D0 <sub>16</sub>				Not acceptable
9	Lock bit enable	7A <sub>16</sub>							Not acceptable
10	Lock bit disable	75 <sub>16</sub>							Not acceptable
11	ID check function	F5 <sub>16</sub>	Address (low)	Address (middle)	Address (high)	ID size	ID1	To ID7	Acceptable
12	Download function	FA <sub>16</sub>	Size (low)	Size (high)	Check-sum	Data input	To required number of times		Not acceptable
13	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 ROM area output function	FC <sub>16</sub>	Address (middle)	Address (high)	Data output	Data output	Data output	Data output to 259th byte	Not acceptable
15	Read check data	FD <sub>16</sub>	Check data (low)	Check data (high)					Not acceptable
16	Baud rate 9600	B0 <sub>16</sub>	B0 <sub>16</sub>						Acceptable
17	Baud rate 19200	B1 <sub>16</sub>	B1 <sub>16</sub>						Acceptable
18	Baud rate 38400	B2 <sub>16</sub>	B2 <sub>16</sub>						Acceptable
19	Baud rate 57600	B3 <sub>16</sub>	B3 <sub>16</sub>						Acceptable

Note 1: Shading indicates transfer from flash memory microcomputer to peripheral unit. All other data is transferred from the peripheral unit to the flash memory microcomputer.

Note 2: SRD refers to status register data. SRD1 refers to status register 1 data.

Note 3: All commands can be accepted when the flash memory is totally blank.



### 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) Transfer the "FF16" command code with the 1st byte.
- (2) Transfer addresses A8 to A15 and A16 to A23 with the 2nd and 3rd bytes 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.

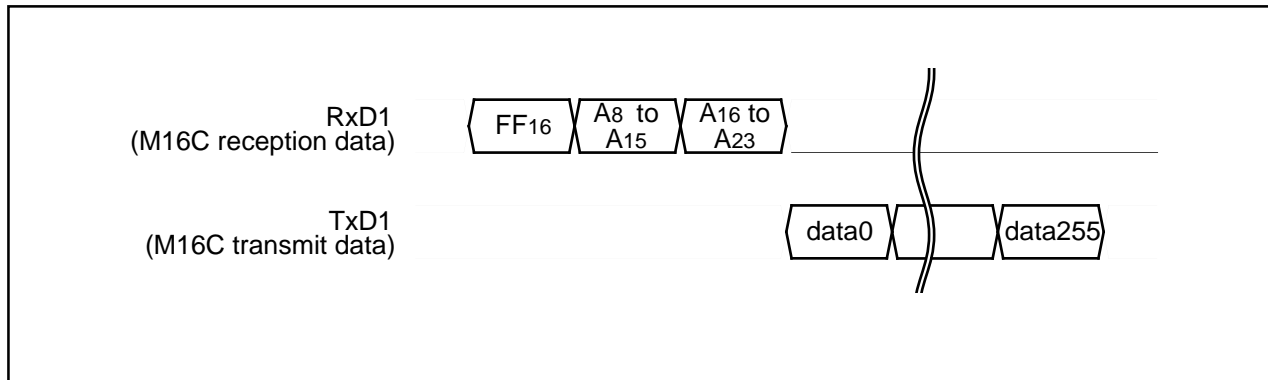


Figure 1.25.22. Timing for page read

### Read Status Register Command

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

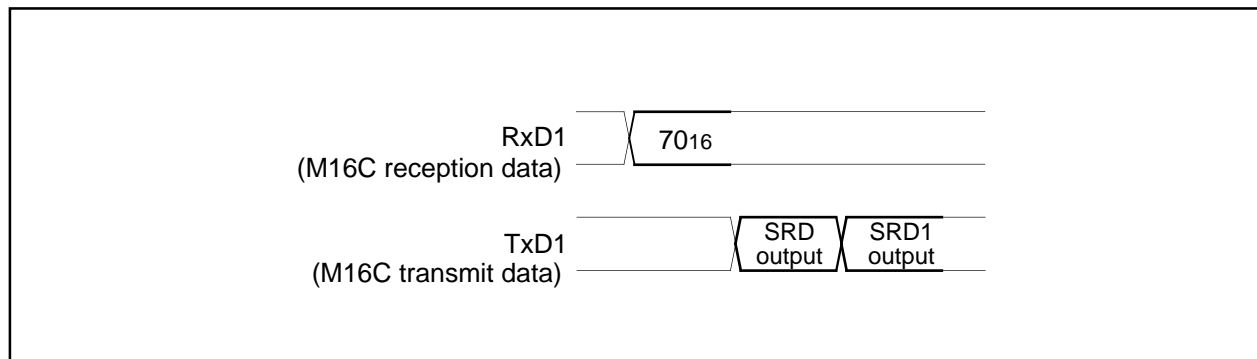
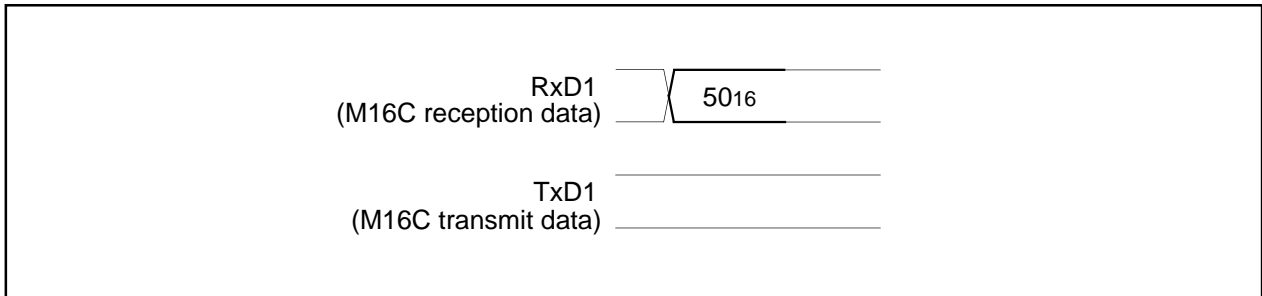


Figure 1.25.23. Timing for reading the status register

### Clear Status Register Command

This command clears the bits (SR3–SR5) which are set when the status register operation ends in error. When the “5016” command code is sent with the 1st byte, the aforementioned bits are cleared.



**Figure 1.25.24. Timing for clearing the status register**

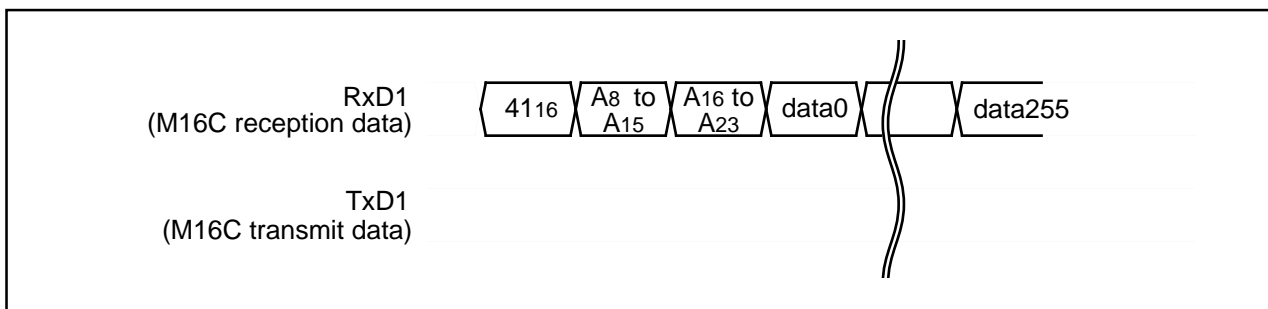
### Page Program Command

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) Transfer the “4116” command code with the 1st byte.
- (2) Transfer addresses A8 to A15 and A16 to A23 with the 2nd and 3rd bytes 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.

The result of the page program can be known by reading the status register. For more information, see the section on the status register.

Each block can be write-protected with the lock bit. For more information, see the section on the data protection function. Additional writing is not allowed with already programmed pages.



**Figure 1.25.25. Timing for the page program**

### Block Erase Command

This command erases the data in the specified block. Execute the block erase command as explained here following.

- (1) Transfer the "20<sub>16</sub>" command code with the 1st byte.
- (2) Transfer addresses A<sub>8</sub> to A<sub>15</sub> and A<sub>16</sub> to A<sub>23</sub> with the 2nd and 3rd bytes respectively.
- (3) Transfer the verify command code "D0<sub>16</sub>" with the 4th byte. With the verify command code, the erase operation will start for the specified block in the flash memory. Write the highest address of the specified block for addresses A<sub>8</sub> to A<sub>23</sub>.

After block erase ends, the result of the block erase operation can be known by reading the status register. For more information, see the section on the status register.

Each block can be erase-protected with the lock bit. For more information, see the section on the data protection function.

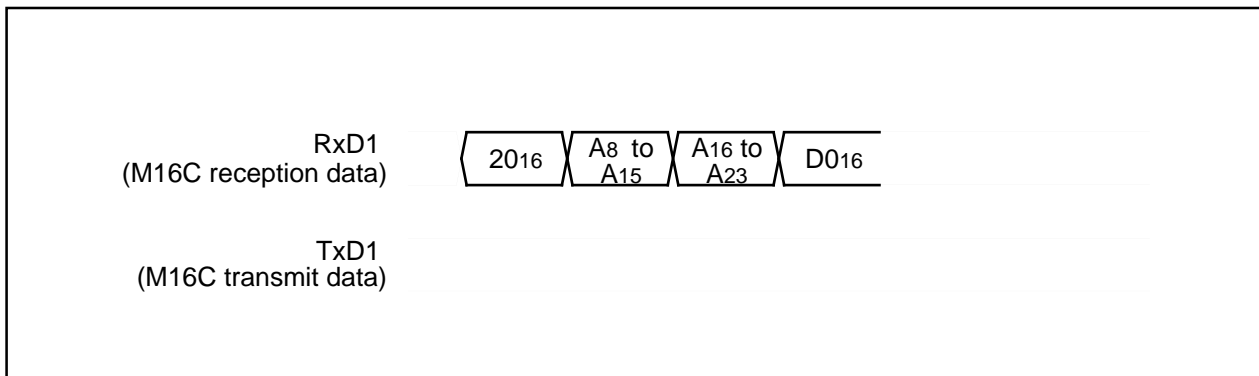


Figure 1.25.26. Timing for block erasing

### 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) Transfer the "A7<sub>16</sub>" command code with the 1st byte.
- (2) Transfer the verify command code "D0<sub>16</sub>" with the 2nd byte. With the verify command code, the erase operation will start and continue for all blocks in the flash memory.

The result of the erase operation can be known by reading the status register. Each block can be erase-protected with the lock bit. For more information, see the section on the data protection function.

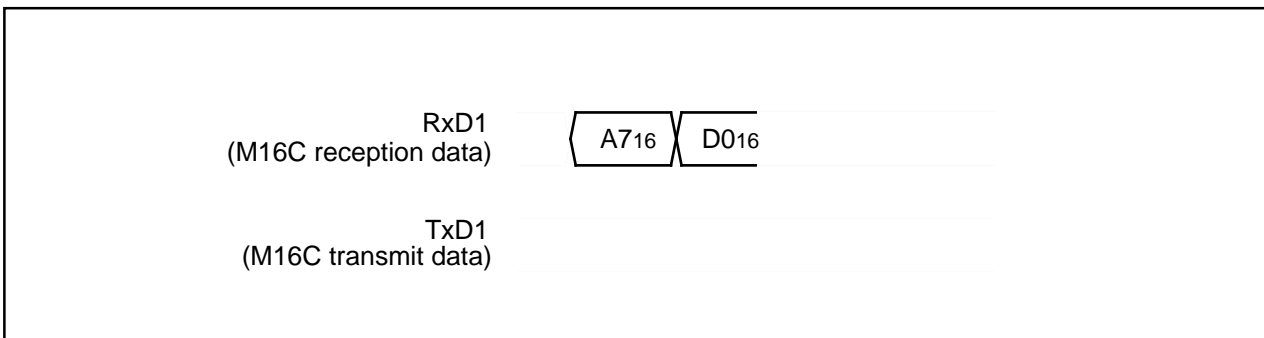


Figure 1.25.27. Timing for erasing all unlocked blocks

### Lock Bit Program Command

This command writes "0" (lock) for the lock bit of the specified block. Execute the lock bit program command as explained here following.

- (1) Transfer the "77<sub>16</sub>" command code with the 1st byte.
- (2) Transfer addresses A<sub>8</sub> to A<sub>15</sub> and A<sub>16</sub> to A<sub>23</sub> with the 2nd and 3rd bytes respectively.
- (3) Transfer the verify command code "D0<sub>16</sub>" with the 4th byte. With the verify command code, "0" is written for the lock bit of the specified block. Write the highest address of the specified block for addresses A<sub>8</sub> to A<sub>23</sub>.

Lock bit status can be read with the read lock bit status command. For information on the lock bit function, reset procedure and so on, see the section on the data protection function.

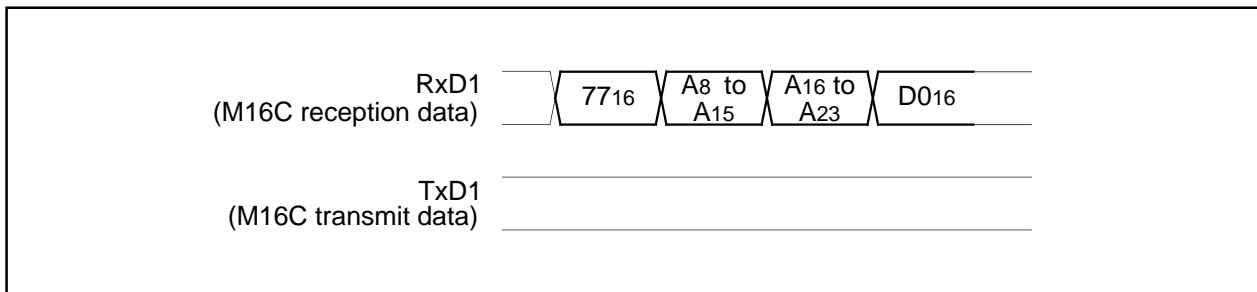


Figure 1.25.28. Timing for the lock bit program

### 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) Transfer the "7116" command code with the 1st byte.
- (2) Transfer addresses A8 to A15 and A16 to A23 with the 2nd and 3rd bytes respectively.
- (3) The lock bit data of the specified block is output with the 4th byte. The lock bit data is the 6th bit (D6) of the output data. Write the highest address of the specified block for addresses A8 to A23.

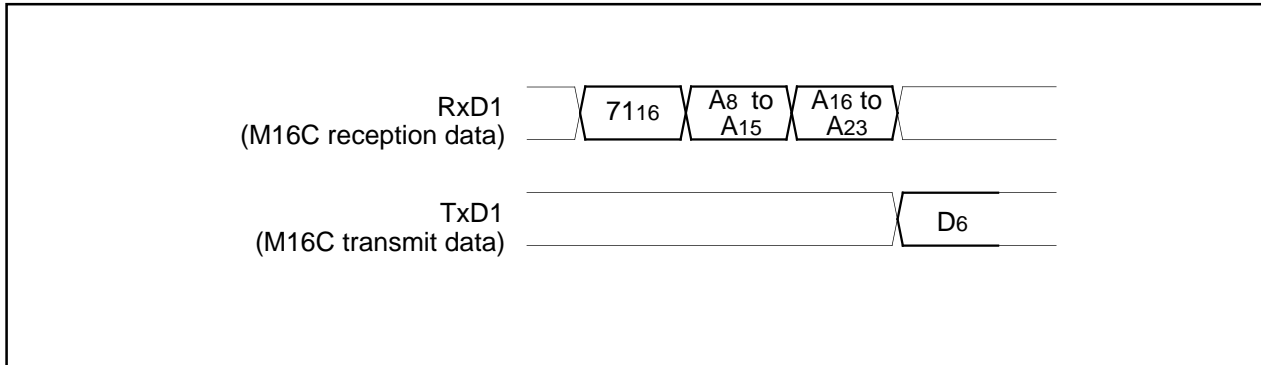


Figure 1.25.29. Timing for reading lock bit status

### Lock Bit Enable Command

This command enables the lock bit in blocks whose bit was disabled with the lock bit disable command. The command code "7A16" is sent with the 1st byte of the serial transmission. This command only enables the lock bit function; it does not set the lock bit itself.

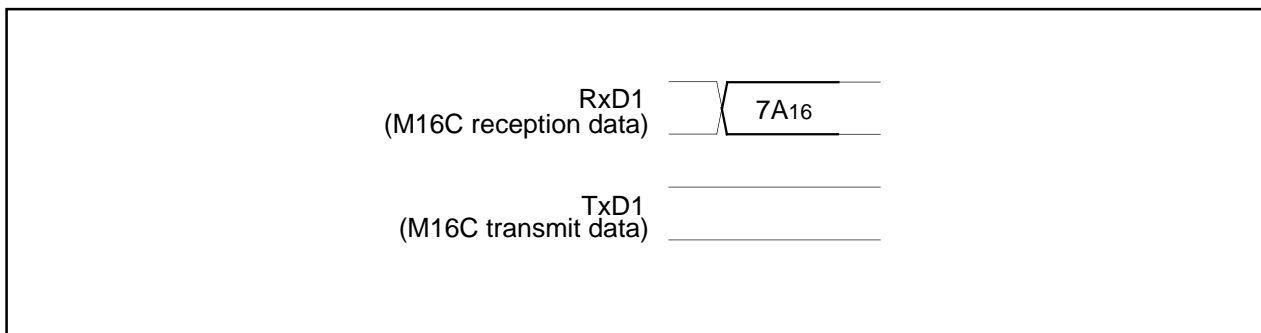
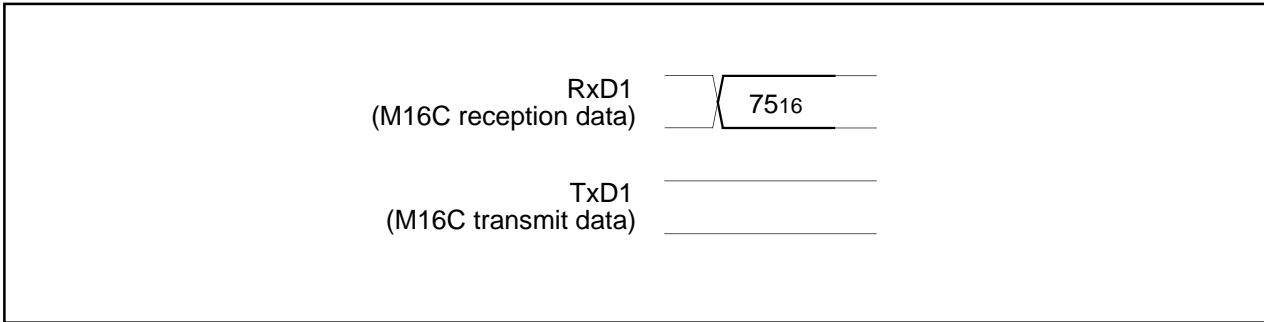


Figure 1.25.30. Timing for enabling the lock bit

### Lock Bit Disable Command

This command disables the lock bit. The command code "7516" is sent with the 1st byte of the serial transmission. This command only disables the lock bit function; it does not set the lock bit itself. However, if an erase command is executed after executing the lock bit disable command, "0" (locked) lock bit data is set to "1" (unlocked) after the erase operation ends. In any case, after the reset is cancelled, the lock bit is enabled.



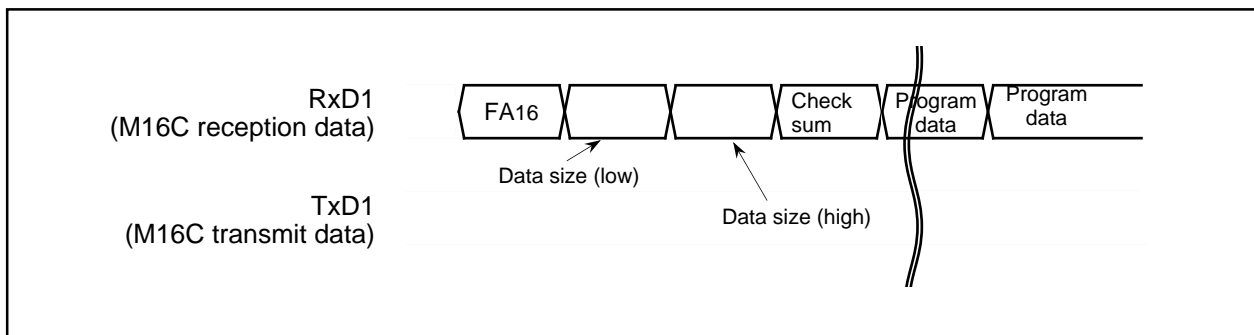
**Figure 1.25.31. Timing for disabling the lock bit**

### Download Command

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

- (1) Transfer the "FA16" command code with the 1st byte.
- (2) Transfer the program size with the 2nd and 3rd bytes.
- (3) Transfer the check sum with the 4th byte. The check sum is added to all data sent with the 5th byte onward.
- (4) The program to execute is sent with 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 1.25.32. Timing for download**

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

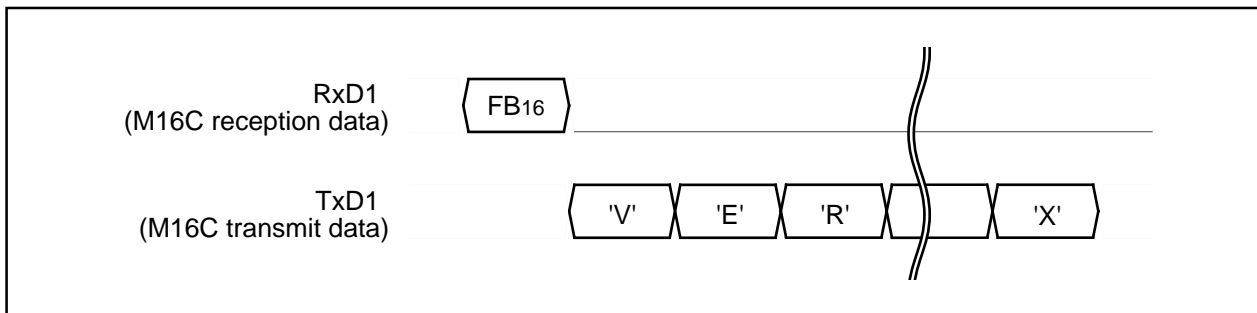


Figure 1.25.33. Timing for version information output

### Boot ROM Area Output Command

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

- (1) Transfer the "FC16" command code with the 1st byte.
- (2) Transfer addresses A8 to A15 and A16 to A23 with the 2nd and 3rd bytes 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.

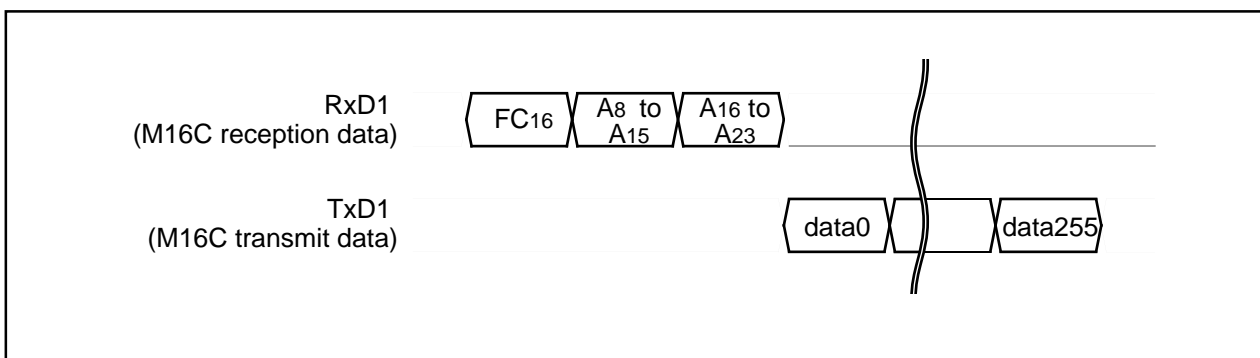
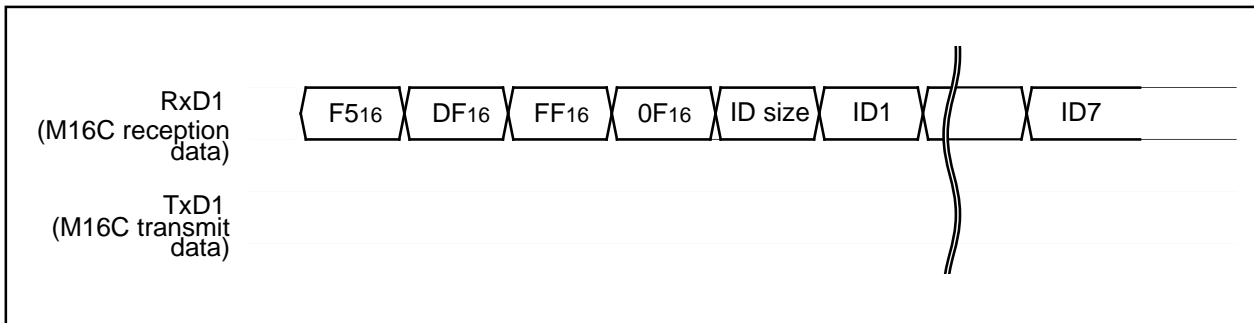


Figure 1.25.34. Timing for boot ROM area output

### ID Check

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

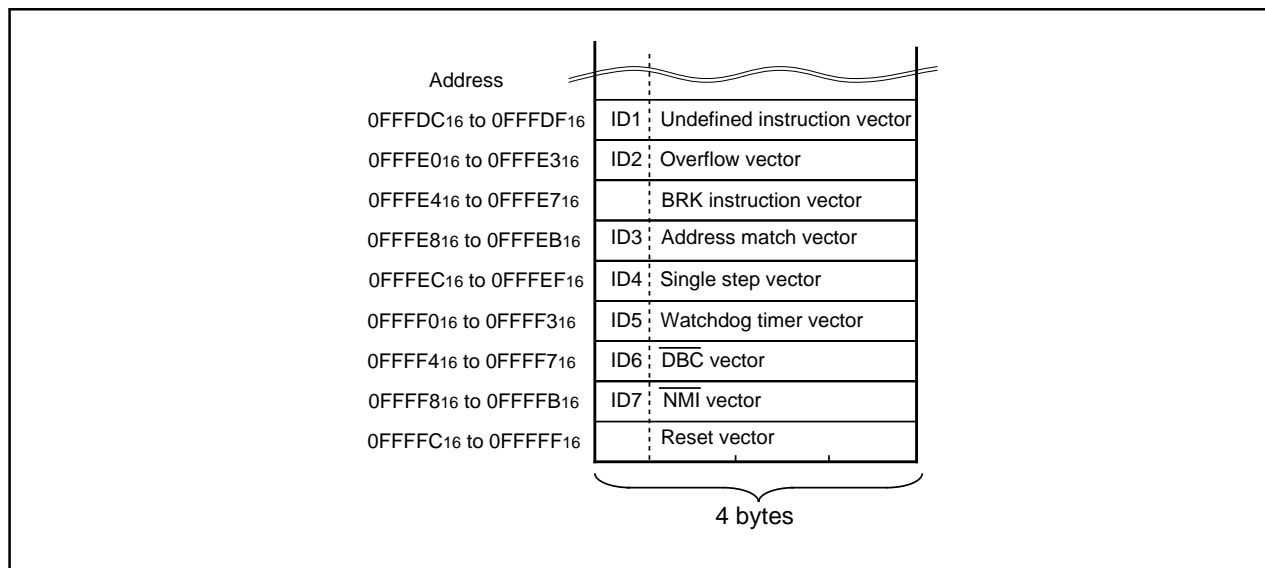
- (1) Transfer the "F5<sub>16</sub>" command code with the 1st byte.
- (2) Transfer addresses A<sub>0</sub> to A<sub>7</sub>, A<sub>8</sub> to A<sub>15</sub> and A<sub>16</sub> to A<sub>23</sub> of the 1st byte of the ID code with the 2nd, 3rd and 4th bytes respectively.
- (3) Transfer the number of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



**Figure 1.25.35. Timing for the ID check**

### ID Code

When the flash memory is not blank, the ID code sent from the peripheral units 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 peripheral units is not accepted. An ID code contains 8 bits of data. Area is, from the 1st byte, addresses 0FFFD<sub>16</sub>, 0FFFE<sub>316</sub>, 0FFFE<sub>B16</sub>, 0FFFE<sub>F16</sub>, 0FFFF<sub>316</sub>, 0FFFF<sub>716</sub> and 0FFFF<sub>B16</sub>. Write a program into the flash memory, which already has the ID code set for these addresses.



**Figure 1.25.36. ID code storage addresses**



### Read Check Data

This command reads the check data that confirms that the write data, which was sent with the page program command, was successfully received.

- (1) Transfer the "FD16" command code with the 1st byte.
- (2) The check data (low) is received with the 2nd byte and the check data (high) with the 3rd.

To use this read check data command, first execute the command and then initialize the check data. Next, execute the page program command the required number of times. After that, when the read check command is executed again, the check data for all of the read data that was sent with the page program command during this time is read. The check data is the result of CRC operation of write data.

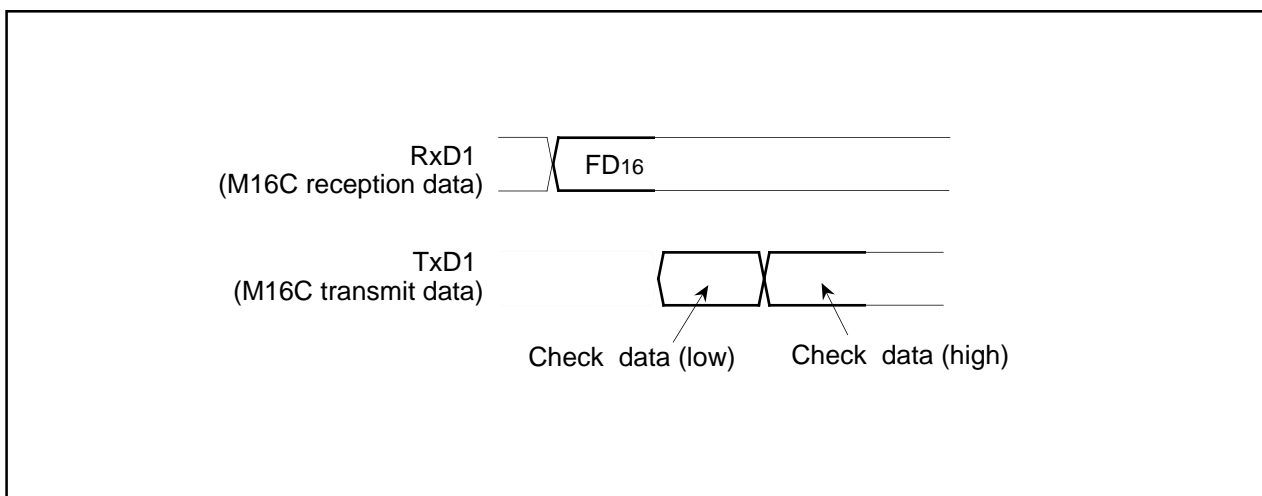


Figure 1.25.37. Timing for the read check data

### Baud Rate 9600

This command changes baud rate to 9,600 bps. Execute it as follows.

- (1) Transfer the "B016" command code with the 1st byte.
- (2) After the "B016" check code is output with the 2nd byte, change the baud rate to 9,600 bps.

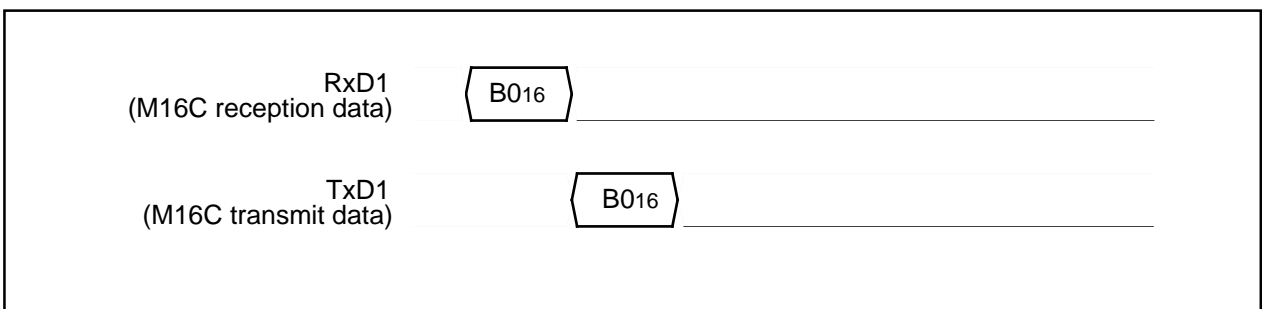


Figure 1.25.38. Timing of baud rate 9600

## Appendix Standard Serial I/O Mode 2 (Flash Memory Version)

**Baud Rate 19200**

This command changes baud rate to 19,200 bps. Execute it as follows.

- (1) Transfer the "B116" command code with the 1st byte.
- (2) After the "B116" check code is output with the 2nd byte, change the baud rate to 19,200 bps.

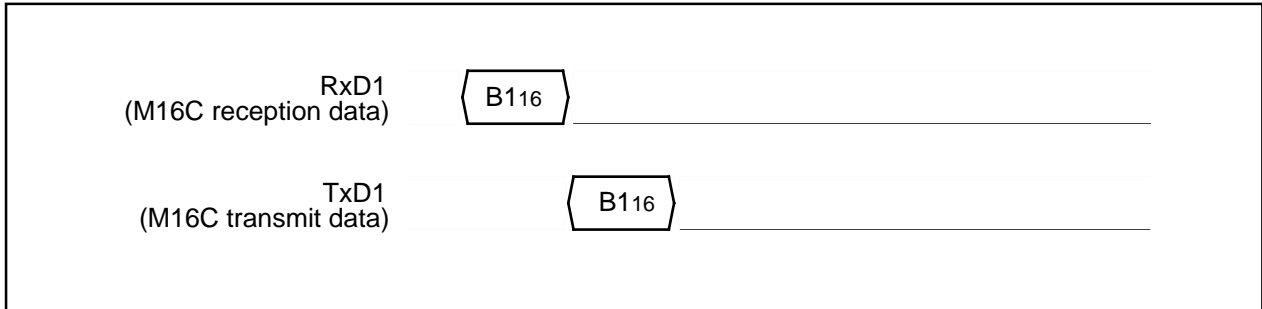


Figure 1.25.39. Timing of baud rate 19200

**Baud Rate 38400**

This command changes baud rate to 38,400 bps. Execute it as follows.

- (1) Transfer the "B216" command code with the 1st byte.
- (2) After the "B216" check code is output with the 2nd byte, change the baud rate to 38,400 bps.

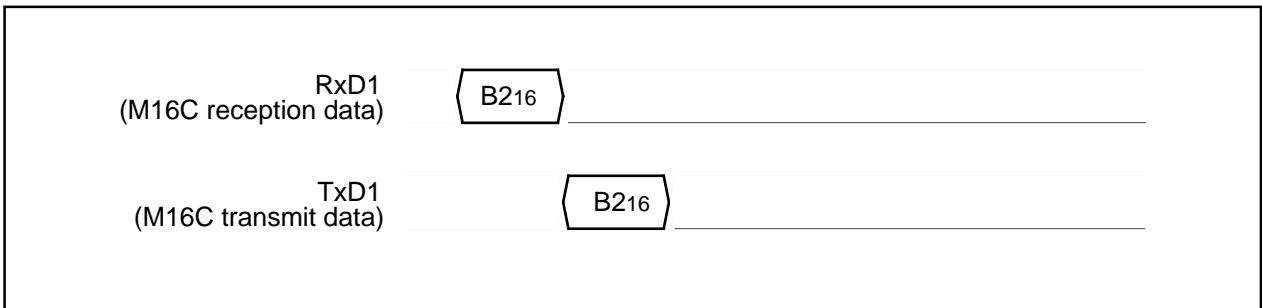


Figure 1.25.40. Timing of baud rate 38400

**Baud Rate 57600**

This command changes baud rate to 57,600 bps. Execute it as follows.

- (1) Transfer the "B316" command code with the 1st byte.
- (2) After the "B316" check code is output with the 2nd byte, change the baud rate to 57,600 bps.

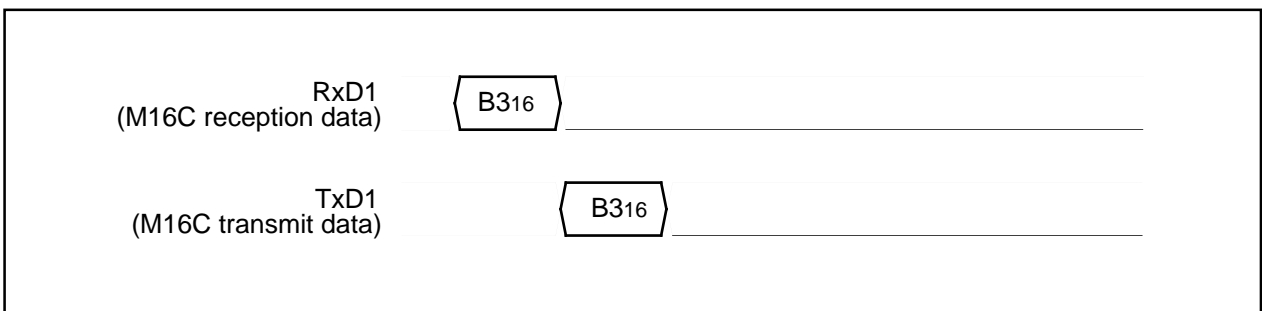


Figure 1.25.41. Timing of baud rate 57600

### Example Circuit Application for The Standard Serial I/O Mode 2

The below figure shows a circuit application for the standard serial I/O mode 2.

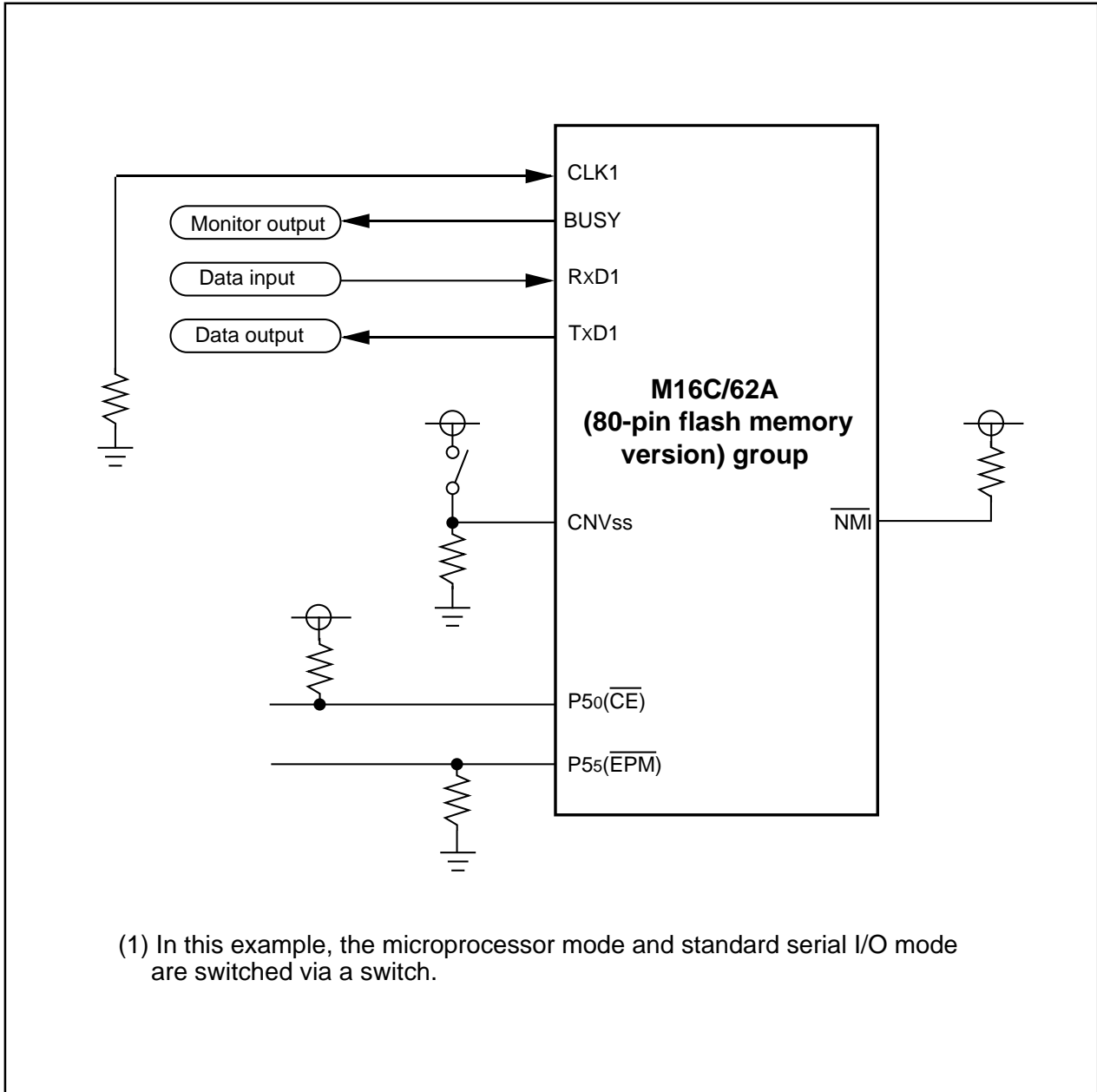


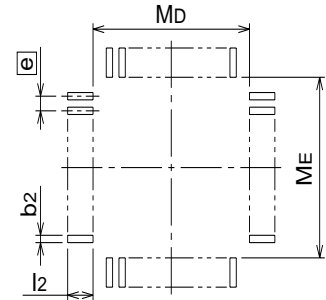
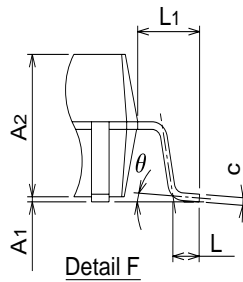
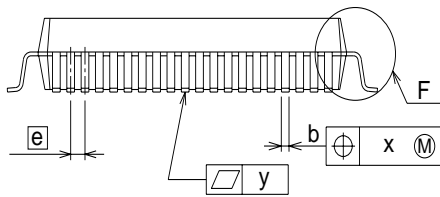
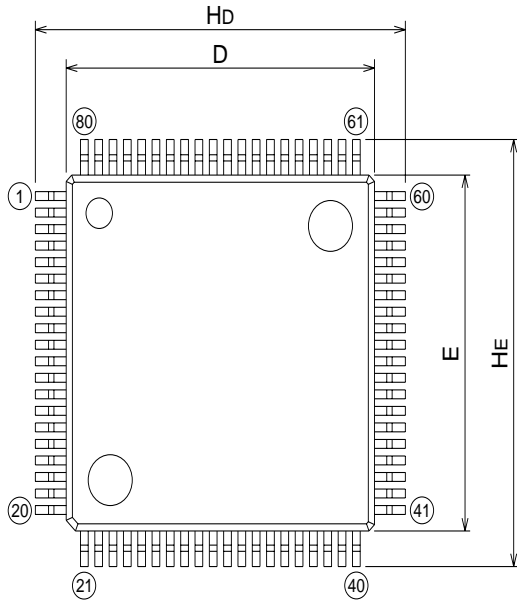
Figure 1.25.42. Example circuit application for the standard serial I/O mode 2

**80P6S-A**



**Plastic 80pin 14X14mm body QFP**

EIAJ Package Code	JEDEC Code	Weight(g)	Lead Material
QFP80-P-1414-0.65		1.11	Alloy 42



Recommended Mount Pad

Symbol	Dimension in Millimeters		
	Min	Nom	Max
A	-	-	3.05
A1	0	0.1	0.2
A2	-	2.8	-
b	0.25	0.3	0.4
c	0.13	0.15	0.2
D	13.8	14.0	14.2
E	13.8	14.0	14.2
e	-	0.65	-
HD	16.5	16.8	17.1
HE	16.5	16.8	17.1
L	0.4	0.6	0.8
L1	-	1.4	-
x	-	-	0.13
y	-	-	0.1
θ	0°	-	10°
b2	-	0.35	-
l2	1.3	-	-
MD	-	14.6	-
ME	-	14.6	-

### Differences between M16C/62A and M16C/62

Item	M16C/62A (80-pin)	M16C/62 (80-pin)
Serial I/O	No CTS/RTS separate function	CTS/RTS separate function
IIC bus mode	Analog or digital delay is selected as SDA delay	Only analog delay is selected as SDA delay
EPROM / one time PROM version	None	Have
Flash memory version	Standard serial I/O mode (clock asynchronous ) is supported	Clock synchronized only

### Differences in SFR between M16C/62A and M16C/62

Address	Register name	M16C/62A (80-pin)	M16C/62 (80-pin)
03B0 <sub>16</sub>	UART transmit/receive register 2 (U2CON)	b6 Reserved bit	b6 CTS/RTS separation bit
0375 <sub>16</sub>	UART2 special mode register 3 (U2SMR3)	Have	None
0377 <sub>16</sub>	UART2 special mode register (U2SMR)	b7 SDA digital delay select bit	b7 Reserved bit

## Revision History

Version	Contents for change	Revision date
REV. A1	<p>Page 35, Figure 1.9.6                      Note: Writing a value to an address after "1" is written to this bit returns the bit to "0". Other bits do not automatically return to "0" and they must therefore <u>be reset by the program.</u></p> <p>Page 124, Figure 1.14.31, bit 5 of the SI/Oi control register (i=3, 4)                      Transfer direction <u>lect</u> bit ---&gt;Transfer direction <u>select</u> bit</p> <p>Page 124, Figure 1.14.31, Note 2                      When using the port as an input/output port by setting the SI/Oi port select bit (i = 3, 4) to "<u>1</u>", be sure to set the sync clock select bit to "1".                      ---&gt;                      When using the port as an input/output port by setting the SI/Oi port select bit (i = 3, 4) to "<u>0</u>", be sure to set the sync clock select bit to "1".</p>	00.07.03
REV. B	<p>Page 2 Note is added in Figure 1.1.1.                      Page 7 (e) is partly revised.                      Page 10 Figure 1.4.1 is partly revised.                      Page 14 Explanation of "Reset" is partly added.                      Page 16 and page 17 Figure 1.6.3 and Figure 1.6.4 are partly revised.                      Page 22 Figure 1.8.1 is partly revised.                      Page 23 Figure 1.8.2 is partly revised.                      Page 23 "Internal Reserved Area Expansion Bit (PM13)" is added.                      Page 25 Figure 1.8.3 is partly revised.                      Page 28 Explanation of "(2) Sub-clock" is partly revised.                      Page 29 Figure 1.9.4 is partly revised.                      Page 30 Explanation of "Stop Mode" is partly revised.                      Page 31 Explanation of "Wait Mode" is partly revised.                      Page 33 Explanation of "Power Control" is partly revised.                      Page 51 Figure 1.10.10 is partly revised.                      Page 53 Explanation of "Address Match Interrupt" is partly revised.                      Page 54 Explanation of "Precautions for Interrupts" is partly revised.                      Page 55 Note is added in Figure 1.10.13.                      Page 56 Explanation of "Watchdog Timer" is partly revised.                      Page 66 Explanation of "DMA request bit" is partly revised.                      Page 70 Figure 1.13.3 is partly revised.                      Page 71 Figure 1.13.5 is partly revised.                      Page 74 Figure 1.13.8 is partly revised.                      Page 75 Table 1.13.3 is partly revised.                      Page 80 Figure 1.13.14 and figure 1.13.15 are partly revised.                      Page 83 Figure 1.13.18 is partly revised.                      Page 84 Table 1.13.8 is partly revised.                      Page 84 Figure 1.13.19 is partly revised.                      Page 90 Figure 1.14.4 is partly revised.                      Page 91 Figure 1.14.5 is partly revised.</p>	01.11.01
Revision history	M16C/62A Group (80-pin) data sheet	

Version	Contents for change	Revision date
REV. B	<p>Page 92 Figure 1.14.6 is partly revised.</p> <p>Page 93 Figure 1.14.7 is partly revised.</p> <p>Page 94 Figure 1.14.8 is partly revised.</p> <p>Page 95 Figure 1.14.9 is partly revised.</p> <p>Page 96 Table 1.14.2 is partly revised.</p> <p>Page 103 Table 1.14.5 is partly revised.</p> <p>Page 112 Figure 1.14.21 is partly revised.</p> <p>Page 113 Explanation of “(a) Function for outputting a parity error signal” is revised.</p> <p>Page 115 Figure 1.14.25 is partly revised.</p> <p>Page 116 Table 1.14.9 is partly revised.</p> <p>Page 124 Figure 1.14.31 is partly revised.</p> <p>Page 127 Note 2 in Table 1.15.1 is partly revised.</p> <p>Page 132 Table 1.15.3 is partly revised.</p> <p>Page 136 Explanation of “(a) Sample and hold” is partly revised.</p> <p>Page 137 Explanation of “D-A Converter” is partly revised.</p> <p>Page 138 Figure 1.16.3 is partly revised.</p> <p>Page 153 Explanation of “Items to be submitted when ordering masked ROM version” is revised.</p> <p>Page 154-166 All symbols of Ta are revised to Topr.</p> <p>Page 162 Table 1.20.19 is partly revised.(RPULLUP)</p> <p>Page 186 Explanation of “Outline Performance (CPU Rewrite Mode)” is partly revised.</p> <p>Page 187 Figure 1.22.1 is partly revised.</p> <p>Page 188 Figure 1.22.2 is partly revised.</p> <p>Page 189 Explanation of “(1) Operation speed” is partly revised.</p> <p>Page 189 Explanation of “(3) Interrupts inhibited against use” is revised.</p> <p>Page 189 Explanation of “(6) Access disable” is partly revised.</p> <p>Page 189 Explanation of “(7) How to access” is partly revised.</p> <p>Page 190 “(8) Writing in the user ROM area” and “(9) Using the lock bit” are added.</p> <p>Page 193 Figure 1.22.5 is partly revised.</p> <p>Page 195 Figure 1.22.7 is partly revised.</p> <p>Page 197 Explanation of “Program status (SR4)” is partly revised.</p> <p>Page 199 Explanation of “ROM code protect function” is partly revised.</p> <p>Page 201 Explanation of “Parallel I/O Mode” is partly revised.</p> <p>Page 202 Explanation of “Pins functions” is partly revised.</p> <p>Page 205 Explanation of “Overview of standard serial I/O mode 1 (clock synchronized)” is partly revised.</p> <p>Page 207 Explanation of “Page Read Command” is partly revised.</p> <p>Page 209 Explanation of “Block Erase Command” is partly revised.</p> <p>Page 211 Figure 1.29.9 and explanation of “Read Lock Bit Status Command” are partly revised.</p> <p>Page 213 Explanation of “Boot ROM area Output Command” is partly revised.</p> <p>Page 216 Explanation of “Data Protection (Block Lock)” is partly revised.</p> <p>Page 217 Explanation of “Block Status After Program (SR3)” is partly revised.</p> <p>Page 218 Table 1.25.3 is partly revised.</p> <p>Page 224 Explanation of “Page Read Command” is partly revised.</p>	01.11.01
Revision history	M16C/62A Group (80-pin) data sheet	

Version	Contents for change	Revision date
REV. B	Page 225 Explanation of "Clear Status Register Command" is partly revised. Page 225 Explanation of "Page Program Command" is partly revised. Page 226 Explanation of "Block Erase Command" is partly revised. Page 227 Explanation of "Erase All Unlocked Blocks Command" is partly revised. Page 227 Explanation of "Lock Bit Program Command" is partly revised. Page 228 Figure 1.25.30 and explanation of "Read Lock Bit Status Command" is partly revised. Page 230 Explanation of "Boot ROM Area Output Command" is partly revised. Page 235 Table of "Differences in SFR between M16C/62A and M16C/62" is added.	01.11.01
Revision history	M16C/62A Group (80-pin) data sheet	



### 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  
M16C/62A Group (80-pin)  
Specification REV.B

---

Nov. First Edition 2001

Edited by  
Committee of editing of Mitsubishi Semiconductor

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.

©2001 MITSUBISHI ELECTRIC CORPORATION