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

The M3062GF8N 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 100-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, low voltage (2.7V to 3.6V), 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.

## Features

- Memory capacity ..... ROM 64K bytes
  RAM 8K bytes
- Shortest instruction execution time ..... 100ns (f(XIN)=10MHz, VCC=2.7V to 3.6V)
- Supply voltage ..... 2.7V to 3.6V (f(XIN)=10MHz, without software wait)
- Low power consumption ..... 30.0mW (VCC = 3V, f(XIN)=10MHz, without software wait)
- Interrupts ..... 25 internal and 8 external interrupt sources, 4 software interrupt sources; 7 levels (including key input interrupt)
- Multifunction 16-bit timer ..... 5 output timers + 6 input timers
- Serial I/O ..... 5 channels
  (3 for UART or clock synchronous, 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 ..... 87 lines
- Input port ..... 1 line (P85 shared with $\overline{NMI}$ pin)
- Memory expansion ..... Available (to a maximum of 1M bytes)
- Chip select output ..... 4 lines
- Clock generating circuit ..... 2 built-in clock generation circuits
  (built-in feedback resistor, and external ceramic or quartz oscillator)

## Applications

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

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

Central Processing Unit (CPU) ..... 10
Reset ..... 13
Processor Mode ..... 20
Clock Generating Circuit ..... 33
Protection ..... 42
Interrupts ..... 43
Watchdog Timer ..... 63
DMAC ..... 65
Timer ..... 75
Timers' functions for three-phase motor control ..... 93
Serial I/O ..... 105
A-D Converter ..... 146
D-A Converter ..... 156
CRC Calculation Circuit ..... 158
Programmable I/O Ports ..... 160
Electrical characteristics ..... 170
Flash memory version ..... 185

## Pin Configuration

Figure 1.1.1 and Figure 1.1.2 show the pin configurations (top view).

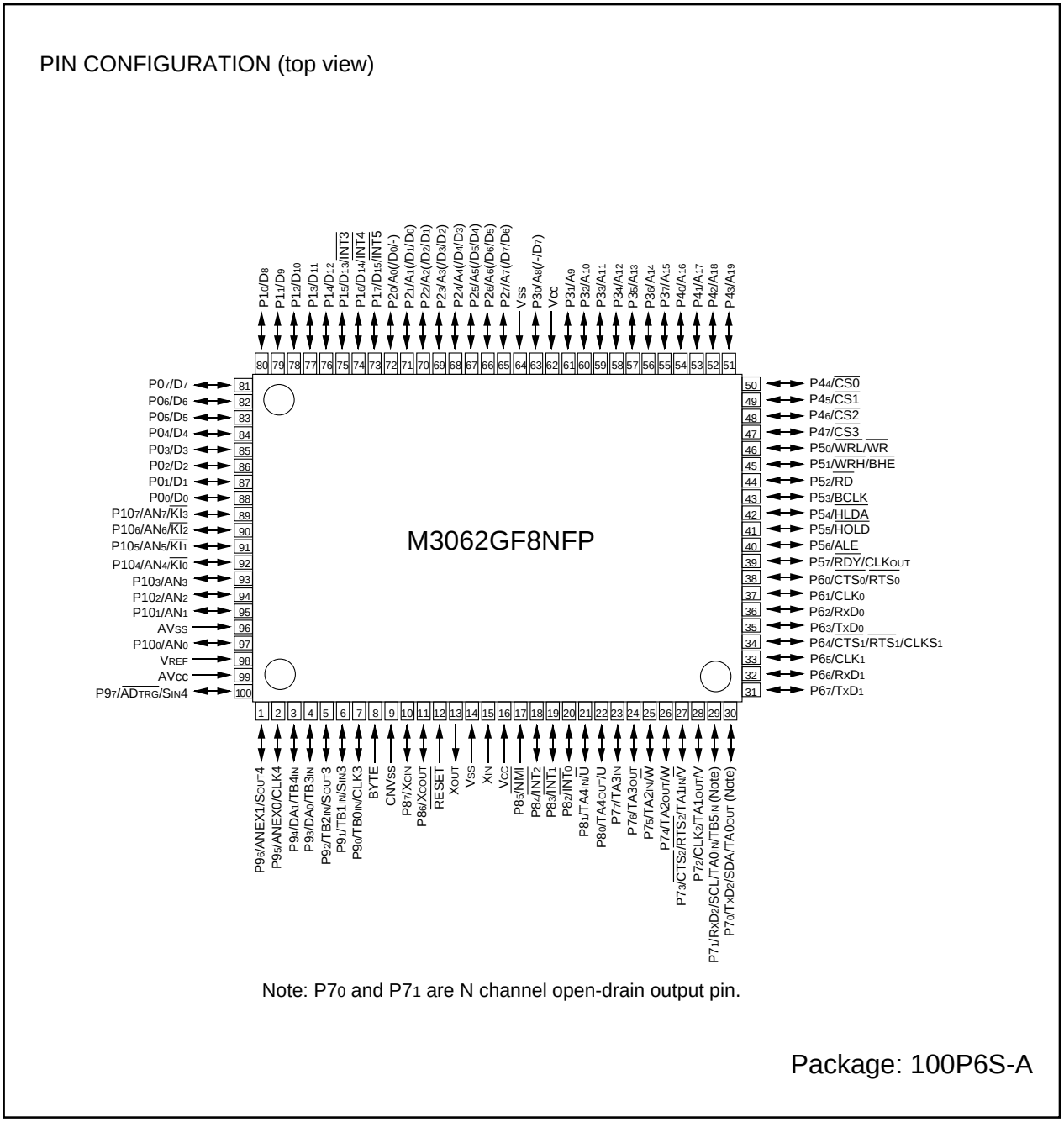
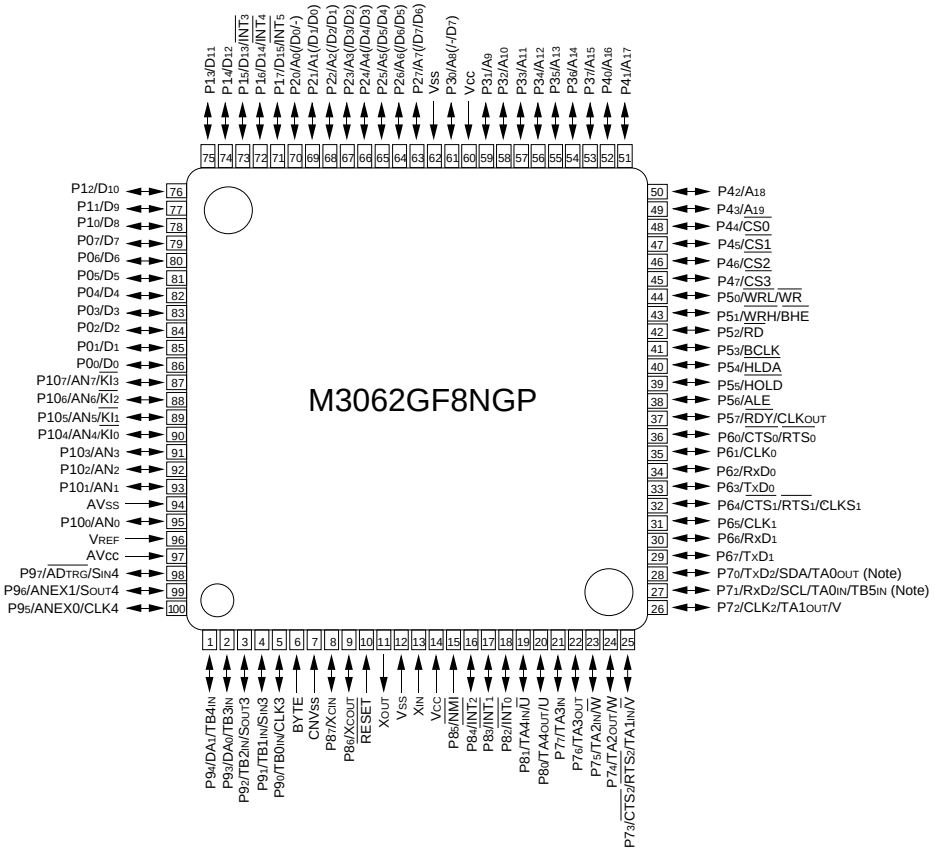



**Figure 1.1.1. Pin configuration (top view)**

PIN CONFIGURATION (top view)




Note: P70 and P71 are N channel open-drain output pin.

Package: 100P6Q-A

**Figure 1.1.2. Pin configuration (top view)**

## Block Diagram

Figure 1.1.3 is a block diagram of the M3062GF8N.

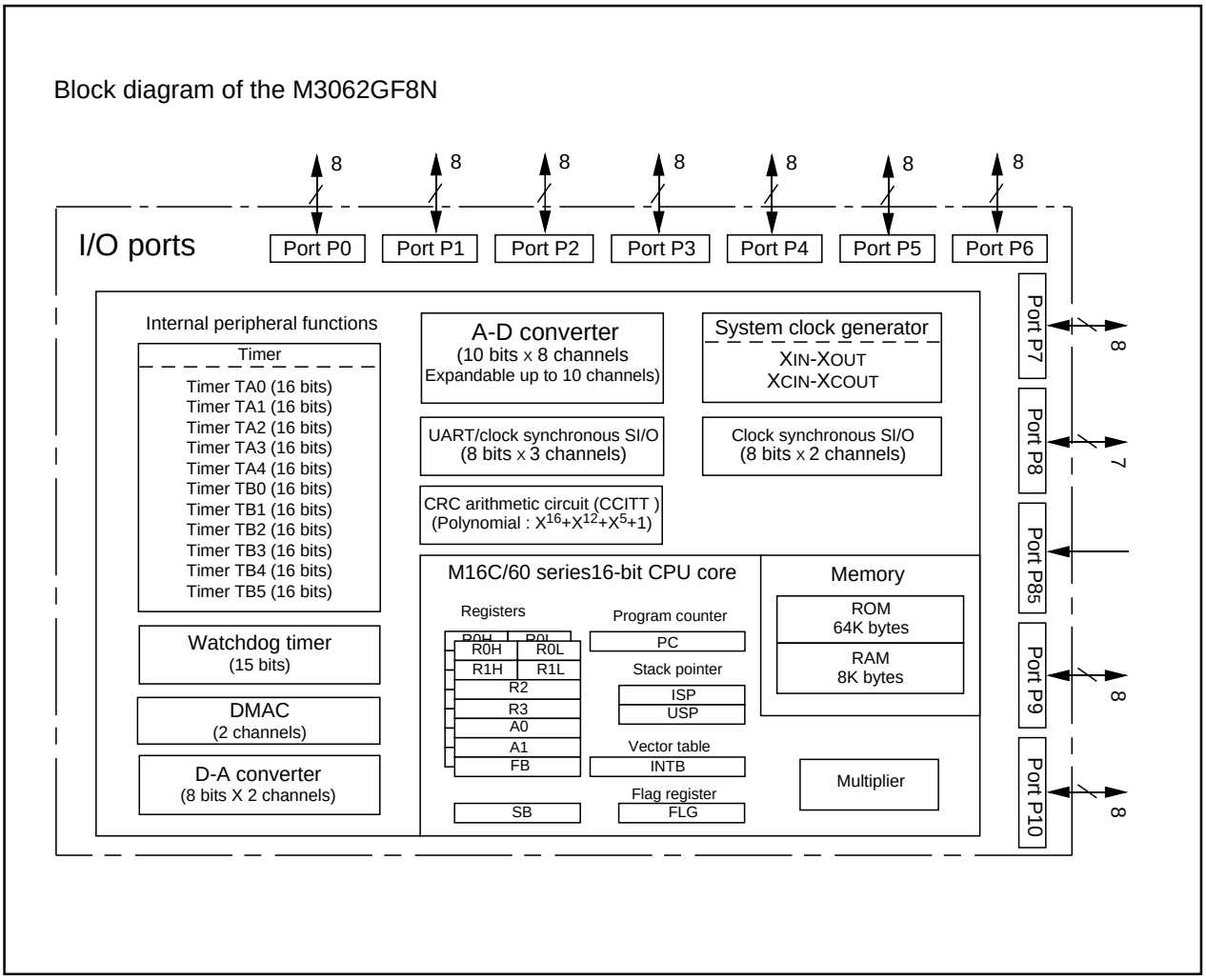



**Figure 1.1.3. Block diagram of M3062GF8N**

## Performance Outline

Table 1.1.1 is a performance outline of M3062GF8N.

**Table 1.1.1. Performance outline of M3062GF8N**

| Item | | Performance |
|---|---|---|
| Number of basic instructions | | 91 instructions |
| Shortest instruction execution time | | 100ns(f($X_{IN}$)=10MHz, $V_{CC}$=2.7V to 3.6V) |
| Memory capacity | ROM | 64K bytes |
| | RAM | 8K bytes |
| I/O port | P0 to P10 (except $P8_5$) | 8 bits x 10, 7 bits x 1 |
| Input port | $P8_5$ | 1 bit x 1 |
| Multifunction timer | TA0, TA1, TA2, TA3, TA4 | 16 bits x 5 |
| | TB0, TB1, TB2, TB3, TB4, TB5 | 16 bits x 6 |
| Serial I/O | UART0, UART1, UART2 | (UART or clock synchronous) x 3 |
| | SI/O3, SI/O4 | (Clock synchronous) x 2 |
| 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 8 external sources, 4 software sources, 7 levels |
| Clock generating circuit | | 2 built-in clock generation circuits<br>(built-in feedback resistor, and external ceramic or quartz oscillator) |
| Supply voltage | | 2.7V to 3.6V (f($X_{IN}$)=10MHz, without software wait) |
| Power consumption | | 30.0mW (f($X_{IN}$) =10MHz, $V_{CC}$=3V without software wait) |
| I/O characteristics | I/O withstand voltage | 3V |
| | Output current | 1mA |
| Memory expansion | | Available (to a maximum of 1M bytes) |
| Device configuration | | CMOS high performance silicon gate |
| Package | | 100-pin plastic mold QFP |

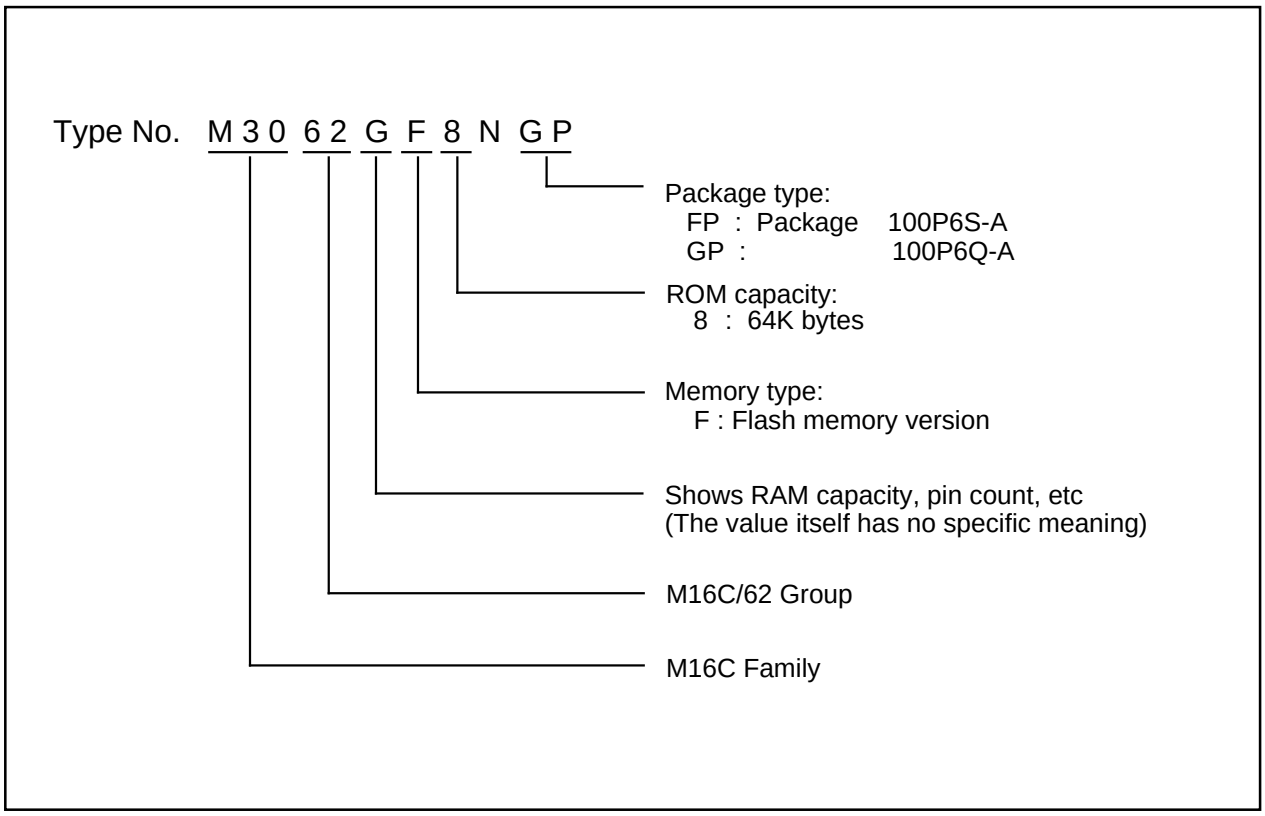



**Figure 1.1.4. Type No., memory size, and package**

## Pin Description

| Pin name | Signal name | I/O type | Function |
|---|---|---|---|
| $V_{CC}$, $V_{SS}$ | Power supply input | | Supply 2.7V to 3.6 V to the $V_{CC}$ pin. Supply 0 V to the $V_{SS}$ pin. |
| $CNV_{SS}$ | $CNV_{SS}$ | Input | This pin switches between processor modes. Connect this pin to the $V_{SS}$ pin when after a reset you want to start operation in single-chip mode (memory expansion mode) or the $V_{CC}$ pin when starting operation in microprocessor mode. |
| $\overline{RESET}$ | Reset input | Input | A "L" on this input resets the microcomputer. |
| $X_{IN}$<br>$X_{OUT}$ | Clock input<br>Clock output | Input<br>Output | These pins are provided for the main clock generating circuit.Connect a ceramic resonator or crystal between the $X_{IN}$ and the $X_{OUT}$ pins. To use an externally derived clock, input it to the $X_{IN}$ pin and leave the $X_{OUT}$ pin open. |
| BYTE | External data bus width select input | Input | This pin selects the width of an external data bus. A 16-bit width is selected when this input is "L"; an 8-bit width is selected when this input is "H". This input must be fixed to either "H" or "L". Connect this pin to the $V_{SS}$ pin when not using external data bus. |
| $AV_{CC}$ | Analog power supply input | | This pin is a power supply input for the A-D converter. Connect this pin to $V_{CC}$. |
| $AV_{SS}$ | Analog power supply input | | This pin is a power supply input for the A-D converter. Connect this pin to $V_{SS}$. |
| $V_{REF}$ | Reference voltage input | Input | This pin is a reference voltage input for the A-D converter. |
| $P0_0$ to $P0_7$ | I/O port P0 | Input/output | This is an 8-bit CMOS I/O port. It has an input/output port direction register that allows the user to set each pin for input or output individually. When used for input in single-chip mode, the port can be set to have or not have a pull-up resistor in units of four bits by software. In memory expansion and microprocessor modes, selection of the internal pull-resistor is not available. |
| $D_0$ to $D_7$ | | Input/output | When set as a separate bus, these pins input and output data ($D_0$–$D_7$). |
| $P1_0$ to $P1_7$ | I/O port P1 | Input/output | This is an 8-bit I/O port equivalent to P0. $P1_5$ to $P1_7$ also function as external interrupt pins as selected by software. |
| $D_8$ to $D_{15}$ | | Input/output | When set as a separate bus, these pins input and output data ($D_8$–$D_{15}$). |
| $P2_0$ to $P2_7$ | I/O port P2 | Input/output | This is an 8-bit I/O port equivalent to P0. |
| $A_0$ to $A_7$ | | Output | These pins output 8 low-order address bits ($A_0$–$A_7$). |
| $A_0/D_0$ to $A_7/D_7$ | | Input/output | If the external bus is set as an 8-bit wide multiplexed bus, these pins input and output data ($D_0$–$D_7$) and output 8 low-order address bits ($A_0$–$A_7$) separated in time by multiplexing. |
| $A_0$<br>$A_1/D_0$ to $A_7/D_6$ | | Output<br>Input/output | If the external bus is set as a 16-bit wide multiplexed bus, these pins input and output data ($D_0$–$D_6$) and output address ($A_1$–$A_7$) separated in time by multiplexing. They also output address ($A_0$). |
| $P3_0$ to $P3_7$ | I/O port P3 | Input/output | This is an 8-bit I/O port equivalent to P0. |
| $A_8$ to $A_{15}$ | | Output | These pins output 8 middle-order address bits ($A_8$–$A_{15}$). |
| $A_8/D_7$, $A_9$ to $A_{15}$ | | Input/output<br>Output | If the external bus is set as a 16-bit wide multiplexed bus, these pins input and output data ($D_7$) and output address ($A_8$) separated in time by multiplexing. They also output address ($A_9$–$A_{15}$). |
| $P4_0$ to $P4_7$ | I/O port P4 | Input/output | This is an 8-bit I/O port equivalent to P0. |
| $A_{16}$ to $A_{19}$, $\overline{CS_0}$ to $\overline{CS_3}$ | | Output<br>Output | These pins output $A_{16}$–$A_{19}$ and $\overline{CS_0}$–$\overline{CS_3}$ signals. $A_{16}$–$A_{19}$ are 4 high-order address bits. $\overline{CS_0}$–$\overline{CS_3}$ are chip select signals used to specify an access space. |

### Pin Description

| Pin name | Signal name | I/O type | Function |
|---|---|---|---|
| $P5_0$ to $P5_7$ | I/O port P5 | Input/output | This is an 8-bit I/O port equivalent to P0. In single-chip mode, $P5_7$ in this port outputs a divide-by-8 or divide-by-32 clock of $X_{IN}$ or a clock of the same frequency as $X_{CIN}$ as selected by software. |
| $\overline{WRL}$ / $\overline{WR}$,<br>$\overline{WRH}$ / $\overline{BHE}$,<br>$\overline{RD}$,<br>$\overline{BCLK}$,<br>$\overline{HLDA}$,<br>$\overline{HOLD}$,<br><br>$\overline{ALE}$,<br>$\overline{RDY}$ | | Output<br>Output<br>Output<br>Output<br>Output<br>Input<br><br>Output<br>Input | Output $\overline{WRL}$, $\overline{WRH}$ ($\overline{WR}$ and $\overline{BHE}$), $\overline{RD}$, $\overline{BCLK}$, $\overline{HLDA}$, and ALE signals. $\overline{WRL}$ and $\overline{WRH}$, and $\overline{BHE}$ and $\overline{WR}$ can be switched using software control.<br>■ $\overline{WRL}$, $\overline{WRH}$, and $\overline{RD}$ selected<br>With a 16-bit external data bus, data is written to even addresses when the $\overline{WRL}$ signal is "L" and to the odd addresses when the $\overline{WRH}$ signal is "L". Data is read when $\overline{RD}$ is "L".<br>■ $\overline{WR}$, $\overline{BHE}$, and $\overline{RD}$ selected<br>Data is written when $\overline{WR}$ is "L". Data is read when $\overline{RD}$ is "L". Odd addresses are accessed when $\overline{BHE}$ is "L". Use this mode when using an 8-bit external data bus.<br>While the input level at the $\overline{HOLD}$ pin is "L", the microcomputer is placed in the hold state. While in the hold state, $\overline{HLDA}$ outputs a "L" level. ALE is used to latch the address. While the input level of the $\overline{RDY}$ pin is "L", the microcomputer is in the ready state. |
| $P6_0$ to $P6_7$ | I/O port P6 | Input/output | This is an 8-bit I/O port equivalent to P0. When used for input in single-chip, memory expansion, and microprocessor modes, the port can be set to have or not have a pull-up resistor in units of four bits by software. Pins in this port also function as UART0 and UART1 I/O pins as selected by software. |
| $P7_0$ to $P7_7$ | I/O port P7 | Input/output | This is an 8-bit I/O port equivalent to P6 ($P7_0$ and $P7_1$ are N channel open-drain output). Pins in this port also function as timer $A_0$–$A_3$, timer B5 or UART2 I/O pins as selected by software. |
| $P8_0$ to $P8_4$,<br>$P8_6$,<br>$P8_7$,<br>$P8_5$ | I/O port P8<br><br><br>I/O port $P8_5$ | Input/output<br>Input/output<br>Input/output<br>Input | $P8_0$ to $P8_4$, $P8_6$, and $P8_7$ are I/O ports with the same functions as P6. Using software, they can be made to function as the I/O pins for timer A4 and the input pins for external interrupts. $P8_6$ and $P8_7$ 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 $P8_6$ ($X_{COUT}$ pin) and $P8_7$ ($X_{CIN}$ pin). $P8_5$ is an input-only port that also functions for $\overline{NMI}$. The $\overline{NMI}$ interrupt is generated when the input at this pin changes from "H" to "L". The $\overline{NMI}$ function cannot be cancelled using software. The pull-up cannot be set for this pin. |
| $P9_0$ to $P9_7$ | I/O port P9 | Input/output | This is an 8-bit I/O port equivalent to P6. 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. |
| $P10_0$ to $P10_7$ | I/O port P10 | Input/output | This is an 8-bit I/O port equivalent to P6. Pins in this port also function as A-D converter input pins as selected by software. Furthermore, $P10_4$–$P10_7$ also function as input pins for the key input interrupt function. |

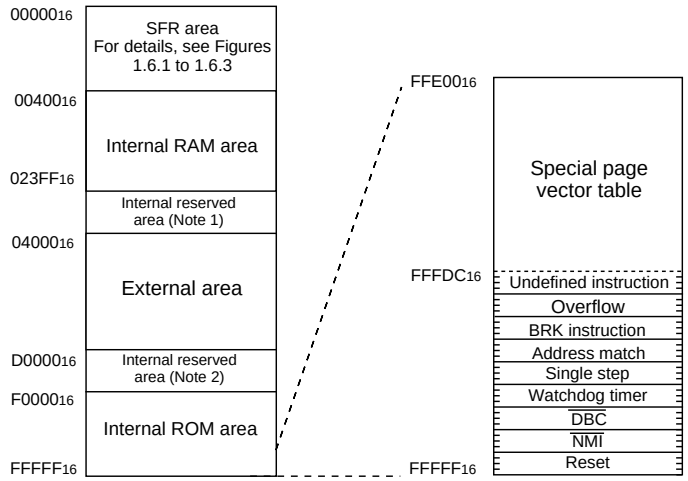
## Operation of Functional Blocks

The M3062GF8NFP 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 1.3.1 is a memory map of the M3062GF8NFP. The address space extends the 1M bytes from address $00000_{16}$ to $FFFFF_{16}$. From $FFFFF_{16}$ down is ROM. For example, in the M3062GF8NFP, there is 64K bytes of internal ROM from $F0000_{16}$ to $FFFFF_{16}$. The vector table for fixed interrupts such as the reset and $\overline{NMI}$ are mapped to $FFFDC_{16}$ to $FFFFF_{16}$. 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 M3062GF8NFP, 8K bytes of internal RAM is mapped to the space from $00400_{16}$ to $023FF_{16}$. 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 1.6.1 to 1.6.3 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.
In memory expansion mode and microprocessor mode, a part of the spaces are reserved and cannot be used. For example, in the M3062GF8NFP, the following spaces cannot be used.

- The space between $02400_{16}$ and $03FFF_{16}$ (Memory expansion and microprocessor modes)
- The space between $D0000_{16}$ and $EFFFF_{16}$ (Memory expansion mode)




Note 1: During memory expansion and microprocessor modes, can not be used.
Note 2: In memory expansion mode, can not be used.

**Figure 1.3.1. Memory map**

## Central Processing Unit (CPU)

The CPU has a total of 13 registers shown in Figure 1.4.1. Seven of these registers (R0, R1, R2, R3, A0, A1, and FB) come in two sets; therefore, these have two register banks.

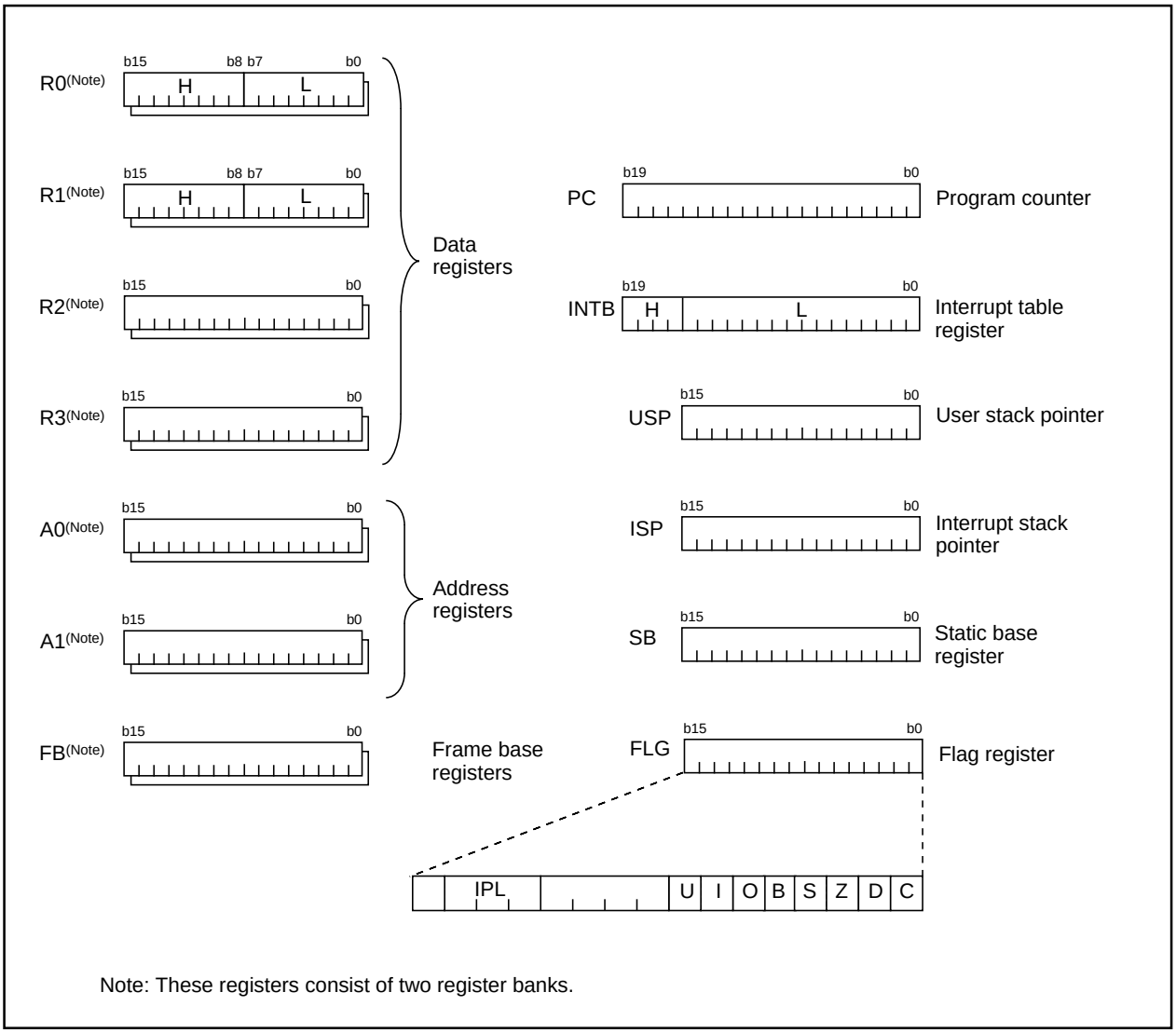



**Figure 1.4.1. Central processing unit register**

### (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 1.4.2 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.

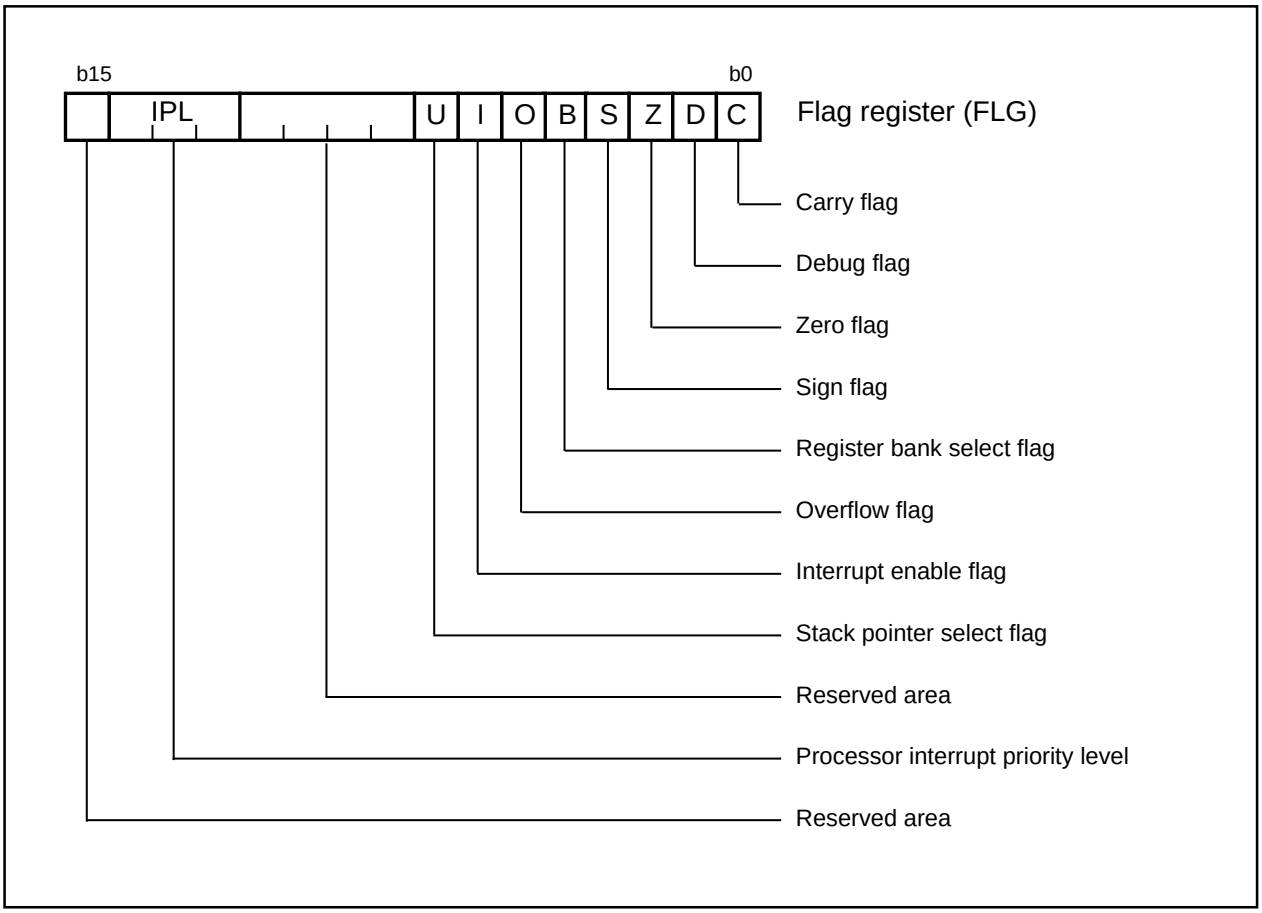



**Figure 1.4.2. Flag register (FLG)**

## 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” (0.2VCC max.) for at least 20 cycles. When the reset pin level is then returned to the “H” level while main clock is stable, the reset status is cancelled and program execution resumes from the address in the reset vector table.

Figure 1.5.1 shows the example reset circuit. Figure 1.5.2 shows the reset sequence.

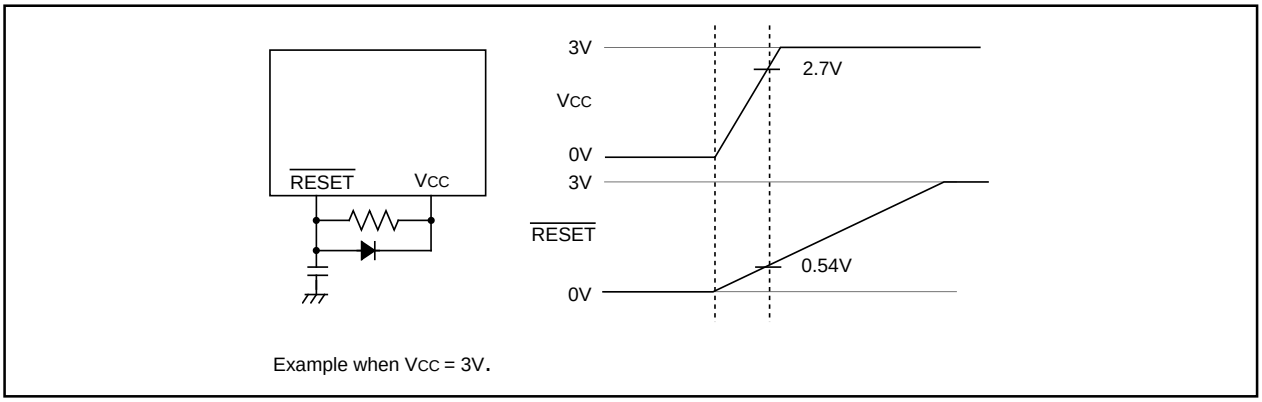



**Figure 1.5.1. Example reset circuit**

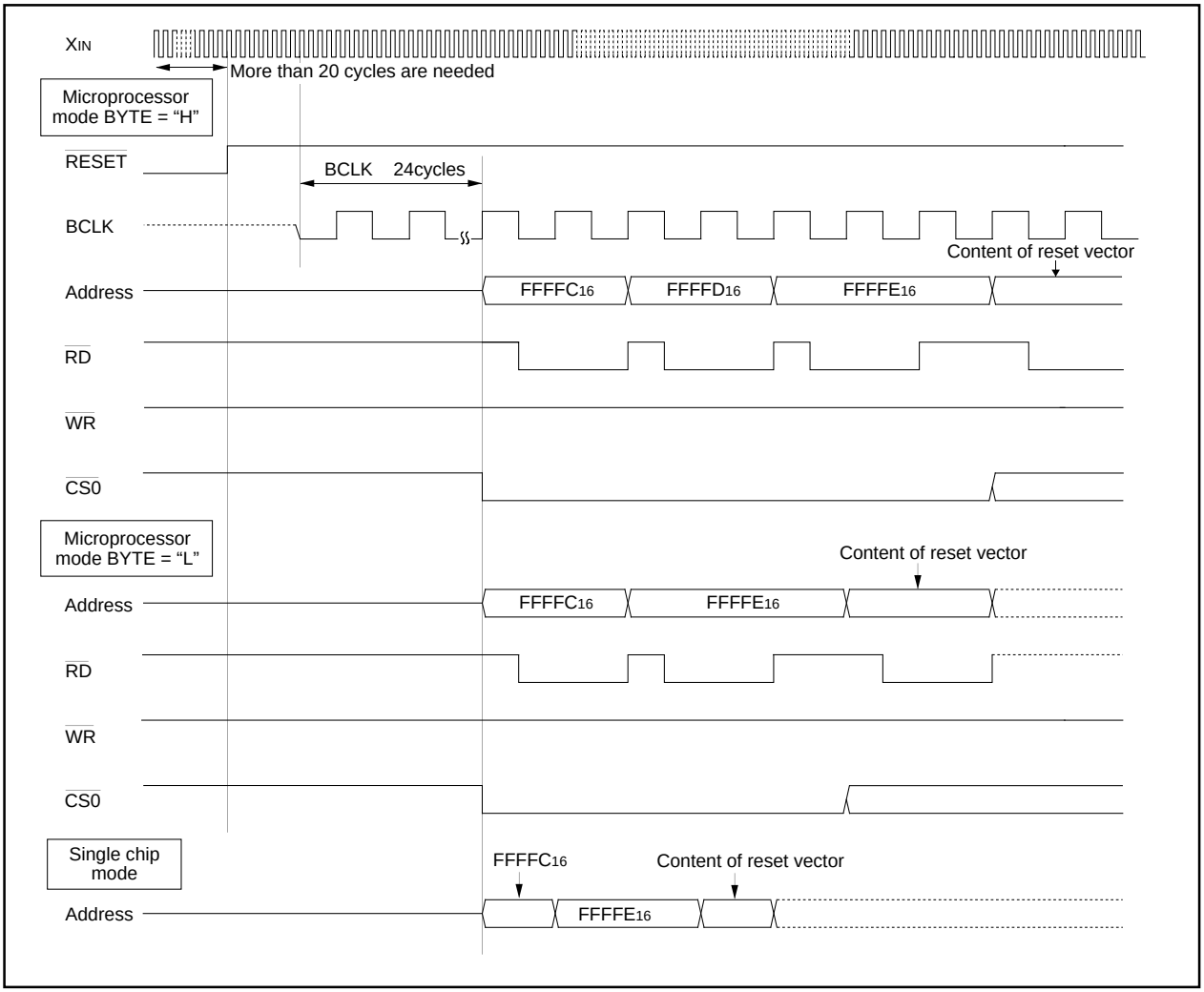



**Figure 1.5.2. Reset sequence**

Table 1.5.1 shows the statuses of the other pins while the $\overline{\text{RESET}}$ pin level is “L”. Figures 1.5.3 and 1.5.4 show the internal status of the microcomputer immediately after the reset is cancelled.

**Table 1.5.1. Pin status when $\overline{\text{RESET}}$ pin level is “L”**

| Pin name | Status | | |
|---|---|---|---|
| | CNVSS = VSS | CNVSS = VCC | |
| | | BYTE = VSS | BYTE = VCC |
| P0 | Input port (floating) | Data input (floating) | Data input (floating) |
| P1 | Input port (floating) | Data input (floating) | Input port (floating) |
| P2, P3, P40 to P43 | Input port (floating) | Address output (undefined) | Address output (undefined) |
| P44 | Input port (floating) | $\overline{\text{CS0}}$ output (“H” level is output) | $\overline{\text{CS0}}$ output (“H” level is output) |
| P45 to P47 | Input port (floating) | Input port (floating)<br>(pull-up resistor is on) | Input port (floating)<br>(pull-up resistor is on) |
| P50 | Input port (floating) | $\overline{\text{WR}}$ output (“H” level is output) | $\overline{\text{WR}}$ output (“H” level is output) |
| P51 | Input port (floating) | $\overline{\text{BHE}}$ output (undefined) | $\overline{\text{BHE}}$ output (undefined) |
| P52 | Input port (floating) | $\overline{\text{RD}}$ output (“H” level is output) | $\overline{\text{RD}}$ output (“H” level is output) |
| P53 | Input port (floating) | BCLK output | BCLK output |
| P54 | Input port (floating) | $\overline{\text{HLDA}}$ output (The output value depends on the input to the $\overline{\text{HOLD}}$ pin) | $\overline{\text{HLDA}}$ output (The output value depends on the input to the $\overline{\text{HOLD}}$ pin) |
| P55 | Input port (floating) | $\overline{\text{HOLD}}$ input (floating) | $\overline{\text{HOLD}}$ input (floating) |
| P56 | Input port (floating) | ALE output (“L” level is output) | ALE output (“L” level is output) |
| P57 | Input port (floating) | $\overline{\text{RDY}}$ input (floating) | $\overline{\text{RDY}}$ input (floating) |
| P6, P7, P80 to P84, P86, P87, P9, P10 | Input port (floating) | Input port (floating) | Input port (floating) |

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

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 1: When the $V_{CC}$ level is applied to the $CNV_{SS}$ pin, it is $03_{16}$ at a reset.
Note 2: "$00_{16}$" is read out when set bit 7 (SDDS) of the UART2 special mode register ( address $0377_{16}$) to "1".

**Figure 1.5.3. Device's internal status after a reset is cleared**

| | | | | | |
|---|---|---|---|---|---|
| (58) | Count start flag | $(0380_{16})$··· | $00_{16}$ | (84) | A-D control register 1 | $(03D7_{16})$··· | $00_{16}$ |
| (59) | Clock prescaler reset flag | $(0381_{16})$··· | 0 X X X X X X X | (85) | D-A control register | $(03DC_{16})$··· | $00_{16}$ |
| (60) | One-shot start flag | $(0382_{16})$··· | 0 0 X 0 0 0 0 0 | (86) | Port P0 direction register | $(03E2_{16})$··· | $00_{16}$ |
| (61) | Trigger select flag | $(0383_{16})$··· | $00_{16}$ | (87) | Port P1 direction register | $(03E3_{16})$··· | $00_{16}$ |
| (62) | Up-down flag | $(0384_{16})$··· | $00_{16}$ | (88) | Port P2 direction register | $(03E6_{16})$··· | $00_{16}$ |
| (63) | Timer A0 mode register | $(0396_{16})$··· | $00_{16}$ | (89) | Port P3 direction register | $(03E7_{16})$··· | $00_{16}$ |
| (64) | Timer A1 mode register | $(0397_{16})$··· | $00_{16}$ | (90) | Port P4 direction register | $(03EA_{16})$··· | $00_{16}$ |
| (65) | Timer A2 mode register | $(0398_{16})$··· | $00_{16}$ | (91) | Port P5 direction register | $(03EB_{16})$··· | $00_{16}$ |
| (66) | Timer A3 mode register | $(0399_{16})$··· | $00_{16}$ | (92) | Port P6 direction register | $(03EE_{16})$··· | $00_{16}$ |
| (67) | Timer A4 mode register | $(039A_{16})$··· | $00_{16}$ | (93) | Port P7 direction register | $(03EF_{16})$··· | $00_{16}$ |
| (68) | Timer B0 mode register | $(039B_{16})$··· | 0 0 ? ? 0 0 0 0 | (94) | Port P8 direction register | $(03F2_{16})$··· | 0 0 X 0 0 0 0 0 |
| (69) | Timer B1 mode register | $(039C_{16})$··· | 0 0 ? X 0 0 0 0 | (95) | Port P9 direction register | $(03F3_{16})$··· | $00_{16}$ |
| (70) | Timer B2 mode register | $(039D_{16})$··· | 0 0 ? X 0 0 0 0 | (96) | Port P10 direction register | $(03F6_{16})$··· | $00_{16}$ |
| (71) | UART0 transmit/receive mode register | $(03A0_{16})$··· | $00_{16}$ | (97) | Pull-up control register 0 | $(03FC_{16})$··· | $00_{16}$ |
| (72) | UART0 transmit/receive control register 0 | $(03A4_{16})$··· | 0 0 0 0 1 0 0 0 | (98) | Pull-up control register 1(Note1) | $(03FD_{16})$··· | $00_{16}$ |
| (73) | UART0 transmit/receive control register 1 | $(03A5_{16})$··· | 0 0 0 0 0 0 1 0 | (99) | Pull-up control register 2 | $(03FE_{16})$··· | $00_{16}$ |
| (74) | UART1 transmit/receive mode register | $(03A8_{16})$··· | $00_{16}$ | (100) | Port control register | $(03FF_{16})$··· | $00_{16}$ |
| (75) | UART1 transmit/receive control register 0 | $(03AC_{16})$··· | 0 0 0 0 1 0 0 0 | (101) | Data registers (R0/R1/R2/R3) | | $0000_{16}$ |
| (76) | UART1 transmit/receive control register 1 | $(03AD_{16})$··· | 0 0 0 0 0 0 1 0 | (102) | Address registers (A0/A1) | | $0000_{16}$ |
| (77) | UART transmit/receive control register 2 | $(03B0_{16})$··· | X 0 0 0 0 0 0 0 | (103) | Frame base register (FB) | | $0000_{16}$ |
| (78) | Flash identification register (Note2) | $(03B4_{16})$··· | $00_{16}$ | (104) | Interrupt table register (INTB) | | $00000_{16}$ |
| (79) | Flash memory control register 0 (Note2) | $(03B7_{16})$··· | X X 0 0 0 0 0 1 | (105) | User stack pointer (USP) | | $0000_{16}$ |
| (80) | DMA0 cause select register | $(03B8_{16})$··· | $00_{16}$ | (106) | Interrupt stack pointer (ISP) | | $0000_{16}$ |
| (81) | DMA1 cause select register | $(03BA_{16})$··· | $00_{16}$ | (107) | Static base register (SB) | | $0000_{16}$ |
| (82) | A-D control register 2 | $(03D4_{16})$··· | 0 0 0 0 X X X 0 | (108) | Flag register (FLG) | | $0000_{16}$ |
| (83) | A-D control register 0 | $(03D6_{16})$··· | 0 0 0 0 0 ? ? ? | | | | |

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.

Note1: When the $V_{CC}$ level is applied to the $CNV_{SS}$ pin, it is $02_{16}$ at a reset.
Note2: This register is only exist in flash memory version.

**Figure 1.5.4. Device's internal status after a reset is cleared**

| | |
|---|---|
| $0000_{16}$ | |
| $0001_{16}$ | |
| $0002_{16}$ | |
| $0003_{16}$ | |
| $0004_{16}$ | Processor mode register 0 (PM0) |
| $0005_{16}$ | Processor mode register 1(PM1) |
| $0006_{16}$ | System clock control register 0 (CM0) |
| $0007_{16}$ | System clock control register 1 (CM1) |
| $0008_{16}$ | Chip select control register (CSR) |
| $0009_{16}$ | Address match interrupt enable register (AIER) |
| $000A_{16}$ | Protect register (PRCR) |
| $000B_{16}$ | |
| $000C_{16}$ | |
| $000D_{16}$ | |
| $000E_{16}$ | Watchdog timer start register (WDTS) |
| $000F_{16}$ | Watchdog timer control register (WDC) |
| $0010_{16}$ | Address match interrupt register 0 (RMAD0) |
| $0011_{16}$ | |
| $0012_{16}$ | |
| $0013_{16}$ | |
| $0014_{16}$ | Address match interrupt register 1 (RMAD1) |
| $0015_{16}$ | |
| $0016_{16}$ | |
| $0017_{16}$ | |
| $0018_{16}$ | |
| $0019_{16}$ | |
| $001A_{16}$ | |
| $001B_{16}$ | |
| $001C_{16}$ | |
| $001D_{16}$ | |
| $001E_{16}$ | |
| $001F_{16}$ | |
| $0020_{16}$ | DMA0 source pointer (SAR0) |
| $0021_{16}$ | |
| $0022_{16}$ | |
| $0023_{16}$ | |
| $0024_{16}$ | DMA0 destination pointer (DAR0) |
| $0025_{16}$ | |
| $0026_{16}$ | |
| $0027_{16}$ | |
| $0028_{16}$ | DMA0 transfer counter (TCR0) |
| $0029_{16}$ | |
| $002A_{16}$ | |
| $002B_{16}$ | |
| $002C_{16}$ | DMA0 control register (DM0CON) |
| $002D_{16}$ | |
| $002E_{16}$ | |
| $002F_{16}$ | |
| $0030_{16}$ | DMA1 source pointer (SAR1) |
| $0031_{16}$ | |
| $0032_{16}$ | |
| $0033_{16}$ | |
| $0034_{16}$ | DMA1 destination pointer (DAR1) |
| $0035_{16}$ | |
| $0036_{16}$ | |
| $0037_{16}$ | |
| $0038_{16}$ | DMA1 transfer counter (TCR1) |
| $0039_{16}$ | |
| $003A_{16}$ | |
| $003B_{16}$ | |
| $003C_{16}$ | DMA1 control register (DM1CON) |
| $003D_{16}$ | |
| $003E_{16}$ | |
| $003F_{16}$ | |

| | |
|---|---|
| $0040_{16}$ | |
| $0041_{16}$ | |
| $0042_{16}$ | |
| $0043_{16}$ | |
| $0044_{16}$ | INT3 interrupt control register (INT3IC) |
| $0045_{16}$ | Timer B5 interrupt control register (TB5IC) |
| $0046_{16}$ | Timer B4 interrupt control register (TB4IC) |
| $0047_{16}$ | Timer B3 interrupt control register (TB3IC) |
| $0048_{16}$ | SI/O4 interrupt control register (S4IC)<br>INT5 interrupt control register (INT5IC) |
| $0049_{16}$ | SI/O3 interrupt control register (S3IC)<br>INT4 interrupt control register (INT4IC) |
| $004A_{16}$ | Bus collision detection interrupt control register (BCNIC) |
| $004B_{16}$ | DMA0 interrupt control register (DM0IC) |
| $004C_{16}$ | DMA1 interrupt control register (DM1IC) |
| $004D_{16}$ | Key input interrupt control register (KUPIC) |
| $004E_{16}$ | A-D conversion interrupt control register (ADIC) |
| $004F_{16}$ | UART2 transmit interrupt control register (S2TIC) |
| $0050_{16}$ | UART2 receive interrupt control register (S2RIC) |
| $0051_{16}$ | UART0 transmit interrupt control register (S0TIC) |
| $0052_{16}$ | UART0 receive interrupt control register (S0RIC) |
| $0053_{16}$ | UART1 transmit interrupt control register (S1TIC) |
| $0054_{16}$ | UART1 receive interrupt control register (S1RIC) |
| $0055_{16}$ | Timer A0 interrupt control register (TA0IC) |
| $0056_{16}$ | Timer A1 interrupt control register (TA1IC) |
| $0057_{16}$ | Timer A2 interrupt control register (TA2IC) |
| $0058_{16}$ | Timer A3 interrupt control register (TA3IC) |
| $0059_{16}$ | Timer A4 interrupt control register (TA4IC) |
| $005A_{16}$ | Timer B0 interrupt control register (TB0IC) |
| $005B_{16}$ | Timer B1 interrupt control register (TB1IC) |
| $005C_{16}$ | Timer B2 interrupt control register (TB2IC) |
| $005D_{16}$ | INT0 interrupt control register (INT0IC) |
| $005E_{16}$ | INT1 interrupt control register (INT1IC) |
| $005F_{16}$ | INT2 interrupt control register (INT2IC) |
| $0060_{16}$ | |
| $0061_{16}$ | |
| $0062_{16}$ | |
| $0063_{16}$ | |
| $0064_{16}$ | |
| $0065_{16}$ | |
| ≈ | ≈ |
| $032A_{16}$ | |
| $032B_{16}$ | |
| $032C_{16}$ | |
| $032D_{16}$ | |
| $032E_{16}$ | |
| $032F_{16}$ | |
| $0330_{16}$ | |
| $0331_{16}$ | |
| $0332_{16}$ | |
| $0333_{16}$ | |
| $0334_{16}$ | |
| $0335_{16}$ | |
| $0336_{16}$ | |
| $0337_{16}$ | |
| $0338_{16}$ | |
| $0339_{16}$ | |
| $033A_{16}$ | |
| $033B_{16}$ | |
| $033C_{16}$ | |
| $033D_{16}$ | |
| $033E_{16}$ | |
| $033F_{16}$ | |

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

**Figure 1.6.1. Location of peripheral unit control registers (1)**

| | |
|---|---|
| 0340$_{16}$ | Timer B3, 4, 5 count start flag (TBSR) |
| 0341$_{16}$ | |
| 0342$_{16}$ | Timer A1-1 register (TA11) |
| 0343$_{16}$ | |
| 0344$_{16}$ | Timer A2-1 register (TA21) |
| 0345$_{16}$ | |
| 0346$_{16}$ | Timer A4-1 register (TA41) |
| 0347$_{16}$ | |
| 0348$_{16}$ | Three-phase PWM control register 0(INVC0) |
| 0349$_{16}$ | Three-phase PWM control register 1(INVC1) |
| 034A$_{16}$ | Three-phase output buffer register 0(IDB0) |
| 034B$_{16}$ | Three-phase output buffer register 1(IDB1) |
| 034C$_{16}$ | Dead time timer(DTT) |
| 034D$_{16}$ | Timer B2 interrupt occurrence frequency set counter(ICTB2) |
| 034E$_{16}$ | |
| 034F$_{16}$ | |
| 0350$_{16}$ | Timer B3 register (TB3) |
| 0351$_{16}$ | |
| 0352$_{16}$ | Timer B4 register (TB4) |
| 0353$_{16}$ | |
| 0354$_{16}$ | Timer B5 register (TB5) |
| 0355$_{16}$ | |
| 0356$_{16}$ | |
| 0357$_{16}$ | |
| 0358$_{16}$ | |
| 0359$_{16}$ | |
| 035A$_{16}$ | |
| 035B$_{16}$ | Timer B3 mode register (TB3MR) |
| 035C$_{16}$ | Timer B4 mode register (TB4MR) |
| 035D$_{16}$ | Timer B5 mode register (TB5MR) |
| 035E$_{16}$ | |
| 035F$_{16}$ | Interrupt cause select register (IFSR) |
| 0360$_{16}$ | SI/O3 transmit/receive register (S3TRR) |
| 0361$_{16}$ | |
| 0362$_{16}$ | SI/O3 control register (S3C) |
| 0363$_{16}$ | SI/O3 bit rate generator (S3BRG) |
| 0364$_{16}$ | SI/O4 transmit/receive register (S4TRR) |
| 0365$_{16}$ | |
| 0366$_{16}$ | SI/O4 control register (S4C) |
| 0367$_{16}$ | SI/O4 bit rate generator (S4BRG) |
| 0368$_{16}$ | |
| 0369$_{16}$ | |
| 036A$_{16}$ | |
| 036B$_{16}$ | |
| 036C$_{16}$ | |
| 036D$_{16}$ | |
| 036E$_{16}$ | |
| 036F$_{16}$ | |
| 0370$_{16}$ | |
| 0371$_{16}$ | |
| 0372$_{16}$ | |
| 0373$_{16}$ | |
| 0374$_{16}$ | |
| 0375$_{16}$ | UART2 special mode register 3(U2SMR3) |
| 0376$_{16}$ | UART2 special mode register 2(U2SMR2) |
| 0377$_{16}$ | UART2 special mode register (U2SMR) |
| 0378$_{16}$ | UART2 transmit/receive mode register (U2MR) |
| 0379$_{16}$ | UART2 bit rate generator (U2BRG) |
| 037A$_{16}$ | UART2 transmit buffer register (U2TB) |
| 037B$_{16}$ | |
| 037C$_{16}$ | UART2 transmit/receive control register 0 (U2C0) |
| 037D$_{16}$ | UART2 transmit/receive control register 1 (U2C1) |
| 037E$_{16}$ | UART2 receive buffer register (U2RB) |
| 037F$_{16}$ | |

| | |
|---|---|
| 0380$_{16}$ | Count start flag (TABSR) |
| 0381$_{16}$ | Clock prescaler reset flag (CPSRF) |
| 0382$_{16}$ | One-shot start flag (ONSF) |
| 0383$_{16}$ | Trigger select register (TRGSR) |
| 0384$_{16}$ | Up-down flag (UDF) |
| 0385$_{16}$ | |
| 0386$_{16}$ | Timer A0 (TA0) |
| 0387$_{16}$ | |
| 0388$_{16}$ | Timer A1 (TA1) |
| 0389$_{16}$ | |
| 038A$_{16}$ | Timer A2 (TA2) |
| 038B$_{16}$ | |
| 038C$_{16}$ | Timer A3 (TA3) |
| 038D$_{16}$ | |
| 038E$_{16}$ | Timer A4 (TA4) |
| 038F$_{16}$ | |
| 0390$_{16}$ | Timer B0 (TB0) |
| 0391$_{16}$ | |
| 0392$_{16}$ | Timer B1 (TB1) |
| 0393$_{16}$ | |
| 0394$_{16}$ | Timer B2 (TB2) |
| 0395$_{16}$ | |
| 0396$_{16}$ | Timer A0 mode register (TA0MR) |
| 0397$_{16}$ | Timer A1 mode register (TA1MR) |
| 0398$_{16}$ | Timer A2 mode register (TA2MR) |
| 0399$_{16}$ | Timer A3 mode register (TA3MR) |
| 039A$_{16}$ | Timer A4 mode register (TA4MR) |
| 039B$_{16}$ | Timer B0 mode register (TB0MR) |
| 039C$_{16}$ | Timer B1 mode register (TB1MR) |
| 039D$_{16}$ | Timer B2 mode register (TB2MR) |
| 039E$_{16}$ | |
| 039F$_{16}$ | |
| 03A0$_{16}$ | UART0 transmit/receive mode register (U0MR) |
| 03A1$_{16}$ | UART0 bit rate generator (U0BRG) |
| 03A2$_{16}$ | UART0 transmit buffer register (U0TB) |
| 03A3$_{16}$ | |
| 03A4$_{16}$ | UART0 transmit/receive control register 0 (U0C0) |
| 03A5$_{16}$ | UART0 transmit/receive control register 1 (U0C1) |
| 03A6$_{16}$ | UART0 receive buffer register (U0RB) |
| 03A7$_{16}$ | |
| 03A8$_{16}$ | UART1 transmit/receive mode register (U1MR) |
| 03A9$_{16}$ | UART1 bit rate generator (U1BRG) |
| 03AA$_{16}$ | UART1 transmit buffer register (U1TB) |
| 03AB$_{16}$ | |
| 03AC$_{16}$ | UART1 transmit/receive control register 0 (U1C0) |
| 03AD$_{16}$ | UART1 transmit/receive control register 1 (U1C1) |
| 03AE$_{16}$ | UART1 receive buffer register (U1RB) |
| 03AF$_{16}$ | |
| 03B0$_{16}$ | UART transmit/receive control register 2 (UCON) |
| 03B1$_{16}$ | |
| 03B2$_{16}$ | |
| 03B3$_{16}$ | |
| 03B4$_{16}$ | Flash identification register (FIDR) (Note1) |
| 03B5$_{16}$ | |
| 03B6$_{16}$ | |
| 03B7$_{16}$ | Flash memory control register 0 (FMR0) (Note1) |
| 03B8$_{16}$ | DMA0 request cause select register (DM0SL) |
| 03B9$_{16}$ | |
| 03BA$_{16}$ | DMA1 request cause select register (DM1SL) |
| 03BB$_{16}$ | |
| 03BC$_{16}$ | CRC data register (CRCD) |
| 03BD$_{16}$ | |
| 03BE$_{16}$ | CRC input register (CRCIN) |
| 03BF$_{16}$ | |

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 1.6.2. Location of peripheral unit control registers (2)**

| | |
|---|---|
| $03C0_{16}$ | A-D register 0 (AD0) |
| $03C1_{16}$ | |
| $03C2_{16}$ | A-D register 1 (AD1) |
| $03C3_{16}$ | |
| $03C4_{16}$ | A-D register 2 (AD2) |
| $03C5_{16}$ | |
| $03C6_{16}$ | A-D register 3 (AD3) |
| $03C7_{16}$ | |
| $03C8_{16}$ | A-D register 4 (AD4) |
| $03C9_{16}$ | |
| $03CA_{16}$ | A-D register 5 (AD5) |
| $03CB_{16}$ | |
| $03CC_{16}$ | A-D register 6 (AD6) |
| $03CD_{16}$ | |
| $03CE_{16}$ | A-D register 7 (AD7) |
| $03CF_{16}$ | |
| $03D0_{16}$ | |
| $03D1_{16}$ | |
| $03D2_{16}$ | |
| $03D3_{16}$ | |
| $03D4_{16}$ | A-D control register 2 (ADCON2) |
| $03D5_{16}$ | |
| $03D6_{16}$ | A-D control register 0 (ADCON0) |
| $03D7_{16}$ | A-D control register 1 (ADCON1) |
| $03D8_{16}$ | D-A register 0 (DA0) |
| $03D9_{16}$ | |
| $03DA_{16}$ | D-A register 1 (DA1) |
| $03DB_{16}$ | |
| $03DC_{16}$ | D-A control register (DACON) |
| $03DD_{16}$ | |
| $03DE_{16}$ | |
| $03DF_{16}$ | |
| $03E0_{16}$ | Port P0 (P0) |
| $03E1_{16}$ | Port P1 (P1) |
| $03E2_{16}$ | Port P0 direction register (PD0) |
| $03E3_{16}$ | Port P1 direction register (PD1) |
| $03E4_{16}$ | Port P2 (P2) |
| $03E5_{16}$ | Port P3 (P3) |
| $03E6_{16}$ | Port P2 direction register (PD2) |
| $03E7_{16}$ | Port P3 direction register (PD3) |
| $03E8_{16}$ | Port P4 (P4) |
| $03E9_{16}$ | Port P5 (P5) |
| $03EA_{16}$ | Port P4 direction register (PD4) |
| $03EB_{16}$ | Port P5 direction register (PD5) |
| $03EC_{16}$ | Port P6 (P6) |
| $03ED_{16}$ | Port P7 (P7) |
| $03EE_{16}$ | Port P6 direction register (PD6) |
| $03EF_{16}$ | Port P7 direction register (PD7) |
| $03F0_{16}$ | Port P8 (P8) |
| $03F1_{16}$ | Port P9 (P9) |
| $03F2_{16}$ | Port P8 direction register (PD8) |
| $03F3_{16}$ | Port P9 direction register (PD9) |
| $03F4_{16}$ | Port P10 (P10) |
| $03F5_{16}$ | |
| $03F6_{16}$ | Port P10 direction register (PD10) |
| $03F7_{16}$ | |
| $03F8_{16}$ | |
| $03F9_{16}$ | |
| $03FA_{16}$ | |
| $03FB_{16}$ | |
| $03FC_{16}$ | Pull-up control register 0 (PUR0) |
| $03FD_{16}$ | Pull-up control register 1 (PUR1) |
| $03FE_{16}$ | Pull-up control register 2 (PUR2) |
| $03FF_{16}$ | Port control register (PCR) |

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

**Figure 1.6.3. Location of peripheral unit control registers (3)**

## Software Reset

Writing “1” to bit 3 of the processor mode register 0 (address $0004_{16}$) applies a (software) reset to the microcomputer. A software reset has the same effect as a hardware reset. The contents of internal RAM are preserved.

## Processor Mode

### (1) Types of Processor Mode

One of three processor modes can be selected: single-chip mode, memory expansion mode, and microprocessor mode. The functions of some pins, the memory map, and the access space differ according to the selected processor mode.

- **Single-chip mode**

  In single-chip mode, only internal memory space (SFR, internal RAM, and internal ROM) can be accessed. However, after the reset has been released and the operation of shifting from the microprocessor mode has started (“H” applied to the CNVSS pin), the internal ROM area cannot be accessed even if the CPU shifts to the single-chip mode.

  Ports P0 to P10 can be used as programmable I/O ports or as I/O ports for the internal peripheral functions.

- **Memory expansion mode**

  In memory expansion mode, external memory can be accessed in addition to the internal memory space (SFR, internal RAM, and internal ROM). However, after the reset has been released and the operation of shifting from the microprocessor mode has started (“H” applied to the CNVSS pin), the internal ROM area cannot be accessed even if the CPU shifts to the memory expansion mode.

  In this mode, some of the pins function as the address bus, the data bus, and as control signals. The number of pins assigned to these functions depends on the bus and register settings. (See “Bus Settings” for details.)

- **Microprocessor mode**

  In microprocessor mode, the SFR, internal RAM, and external memory space can be accessed. The internal ROM area cannot be accessed.

  In this mode, some of the pins function as the address bus, the data bus, and as control signals. The number of pins assigned to these functions depends on the bus width and register settings. (See “Bus Settings” for details.)

### (2) Setting Processor Modes

The processor mode is set using the CNVSS pin and the processor mode bits (bits 1 and 0 at address $0004_{16}$). Do not set the processor mode bits to “$10_2$”.

Regardless of the level of the CNVSS pin, changing the processor mode bits selects the mode. Therefore, never change the processor mode bits when changing the contents of other bits. Do not change the processor mode bits simultaneously with other bits when changing the processor mode bits “$01_2$” or “$11_2$”. Change the processor mode bits after changing the other bits. Also do not attempt to shift to or from the microprocessor mode within the program stored in the internal ROM area.

- **Applying Vss to CNVss pin**

  The microcomputer begins operation in single-chip mode after being reset. Memory expansion mode is selected by writing “$01_2$” to the processor mode bits.

- **Applying Vcc to CNVss pin**

  The microcomputer starts to operate in microprocessor mode after being reset.

Figure 1.7.1 shows the processor mode register 0 and 1.

Figure 1.7.2 shows the memory maps applicable for each of the modes.

Processor mode register 0 (Note 1)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| PM0 | $000_{16}$ | $00_{16}$ (Note 2) |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| PM00 | Processor mode bit | b1 b0<br>0 0: Single-chip mode<br>0 1: Memory expansion mode<br>1 0: Do not set<br>1 1: Microprocessor mode | ○ | ○ |
| PM01 | | | ○ | ○ |
| PM02 | R/W mode select bit | 0 : $\overline{RD}$,$\overline{BHE}$,$\overline{WR}$<br>1 : $\overline{RD}$,$\overline{WRH}$,$\overline{WRL}$ | ○ | ○ |
| PM03 | Software reset bit | The device is reset when this bit is set to “1”. The value of this bit is “0” when read. | ○ | ○ |
| PM04 | Multiplexed bus space select bit | b5 b4<br>0 0 : Multiplexed bus is not used<br>0 1 : Allocated to CS2 space<br>1 0 : Allocated to CS1 space<br>1 1 : Allocated to entire space (Note4) | ○ | ○ |
| PM05 | | | ○ | ○ |
| PM06 | Port $P4_0$ to $P4_3$ function select bit (Note 3) | 0 : Address output<br>1 : Port function<br>(Address is not output) | ○ | ○ |
| PM07 | BCLK output disable bit | 0 : BCLK is output<br>1 : BCLK is not output<br>(Pin is left floating) | ○ | ○ |

Note 1: Set bit 1 of the protect register (address $000A_{16}$) to “1” when writing new values to this register.
Note 2: If the $V_{CC}$ voltage is applied to the $CNV_{SS}$, the value of this register when reset is $03_{16}$. (PM00 and PM01 both are set to “1”.)
Note 3: Valid in microprocessor and memory expansion modes.
Note 4: If the entire space is of multiplexed bus in memory expansion mode, choose an 8-bit width.The processor operates using the separate bus after reset is revoked, so the entire space multiplexed bus cannot be chosen in microprocessor mode.
$P3_1$ to $P3_7$ become a port if the entire space multiplexed bus is chosen, so only 256 bytes can be used in each chip select.

Processor mode register 1 (Note )

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | X | X | 0 |

| Symbol | Address | When reset |
|---|---|---|
| PM1 | $0005_{16}$ | $00000XX0_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| Reserved bit | | Must always be set to “0” | ○ | ○ |
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be indeterminate. | | | — | — |
| Reserved bit | | Must always be set to “0” | ○ | ○ |
| Reserved bit | | Must always be set to “0” | ○ | ○ |
| Reserved bit | | Must always be set to “0” | ○ | ○ |
| Reserved bit | | Must always be set to “0” | ○ | ○ |
| PM17 | Wait bit | 0 : No wait state<br>1 : Wait state inserted | ○ | ○ |

Note : Set bit 1 of the protect register (address $000A_{16}$) to “1” when writing new values to this register.

**Figure 1.7.1. Processor mode register 0 and 1**

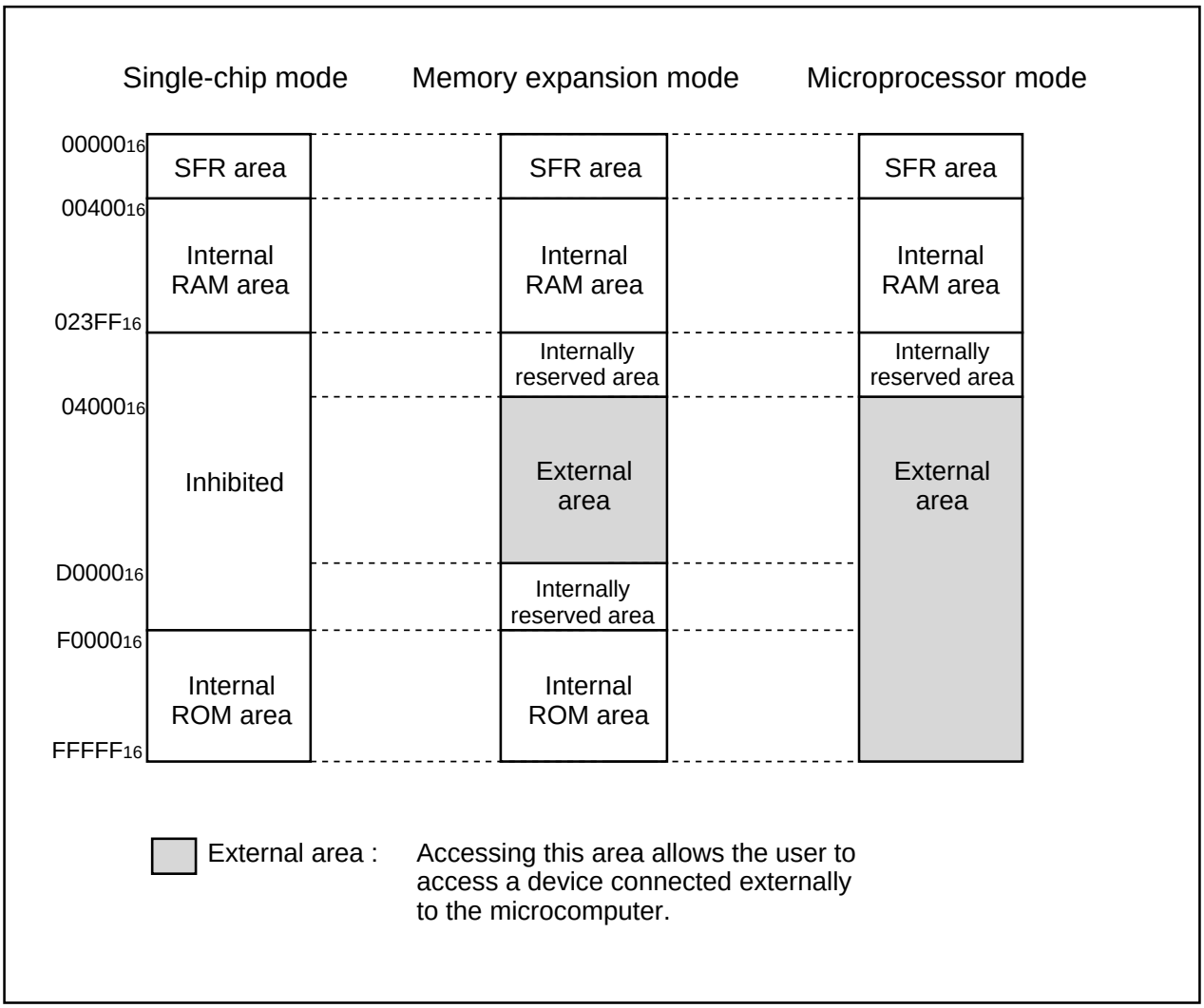



**Figure 1.7.2. Memory maps in each processor mode**

## Bus Settings

The BYTE pin and bits 4 to 6 of the processor mode register 0 (address $0004_{16}$) are used to change the bus settings. Table 1.8.1 shows the factors used to change the bus settings.

**Table 1.8.1. Factors for switching bus settings**

| Bus setting | Switching factor |
|---|---|
| Switching external address bus width | Bit 6 of processor mode register 0 |
| Switching external data bus width | BYTE pin |
| Switching between separate and multiplex bus | Bits 4 and 5 of processor mode register 0 |

### (1) Selecting external address bus width

The address bus width for external output in the 1M bytes of address space can be set to 16 bits (64K bytes address space) or 20 bits (1M bytes address space). When bit 6 of the processor mode register 0 is set to “1”, the external address bus width is set to 16 bits, and P2 and P3 become part of the address bus. $P4_0$ to $P4_3$ can be used as programmable I/O ports. When bit 6 of processor mode register 0 is set to “0”, the external address bus width is set to 20 bits, and P2, P3, and $P4_0$ to $P4_3$ become part of the address bus.

### (2) Selecting external data bus width

The external data bus width can be set to 8 or 16 bits. (Note, however, that only the separate bus can be set.) When the BYTE pin is “L”, the bus width is set to 16 bits; when “H”, it is set to 8 bits. (The internal bus width is permanently set to 16 bits.) While operating, fix the BYTE pin either to “H” or to “L”.

### (3) Selecting separate/multiplex bus

The bus format can be set to multiplex or separate bus using bits 4 and 5 of the processor mode register 0.

- **Separate bus**

  In this mode, the data and address are input and output separately. The data bus can be set using the BYTE pin to be 8 or 16 bits. When the BYTE pin is “H”, the data bus is set to 8 bits and P0 functions as the data bus and P1 as a programmable I/O port. When the BYTE pin is “L”, the data bus is set to 16 bits and P0 and P1 are both used for the data bus.

  When the separate bus is used for access, a software wait can be selected.

- **Multiplex bus**

  In this mode, data and address I/O are time multiplexed. With the BYTE pin = “H”, the 8 bits from $D_0$ to $D_7$ are multiplexed with $A_0$ to $A_7$.

  With the BYTE pin = “L”, the 8 bits from $D_0$ to $D_7$ are multiplexed with $A_1$ to $A_8$. $D_8$ to $D_{15}$ are not multiplexed. In this case, the external devices connected to the multiplexed bus are mapped to the microcomputer’s even addresses (every 2nd address). To access these external devices, access the even addresses as bytes.

  The ALE signal latches the address. It is output from $P5_6$.

  Before using the multiplex bus for access, be sure to insert a software wait.

  If the entire space is of multiplexed bus in memory expansion mode, choose an 8-bit width.

  The processor operates using the separate bus after reset is revoked, so the entire space multiplexed bus cannot be chosen in microprocessor mode.

  $P3_1$ to $P3_7$ become a port if the entire space multiplexed bus is chosen, so only 256 bytes can be used in each chip select.

**Table 1.8.2. Pin functions for each processor mode**

| Processor mode | Single-chip mode | Memory expansion mode/microprocessor modes | | | | Memory expansion mode |
|---|---|---|---|---|---|---|
| Multiplexed bus space select bit | | "01", "10" [Either CS1 or CS2 is for multiplexed bus and others are for separate bus] | | "00" (separate bus) | | "11" (Note 1) [multiplexed bus for the entire space] |
| Data bus width BYTE pin level | | 8 bits "H" | 16 bits "L" | 8 bits "H" | 16 bits "L" | 8 bit "H" |
| $P0_0$ to $P0_7$ | I/O port | Data bus | Data bus | Data bus | Data bus | I/O port |
| $P1_0$ to $P1_7$ | I/O port | I/O port | Data bus | I/O port | Data bus | I/O port |
| $P2_0$ | I/O port | Address bus /data bus(Note 2) | Address bus | Address bus | Address bus | Address bus /data bus |
| $P2_1$ to $P2_7$ | I/O port | Address bus /data bus(Note 2) | Address bus /data bus(Note 2) | Address bus | Address bus | Address bus /data bus |
| $P3_0$ | I/O port | Address bus | Address bus /data bus(Note 2) | Address bus | Address bus | $A_8/D_7$ |
| $P3_1$ to $P3_7$ | I/O port | Address bus | Address bus | Address bus | Address bus | I/O port |
| $P4_0$ to $P4_3$ Port $P4_0$ to $P4_3$ function select bit = 1 | I/O port | I/O port | I/O port | I/O port | I/O port | I/O port |
| $P4_0$ to $P4_3$ Port $P4_0$ to $P4_3$ function select bit = 0 | I/O port | Address bus | Address bus | Address bus | Address bus | I/O port |
| $P4_4$ to $P4_7$ | I/O port | $\overline{CS}$ (chip select) or programmable I/O port (For details, refer to "Bus control") | | | | |
| $P5_0$ to $P5_3$ | I/O port | Outputs $\overline{RD}$, $\overline{WRL}$, $\overline{WRH}$, and BCLK or $\overline{RD}$, $\overline{BHE}$, $\overline{WR}$, and BCLK (For details, refer to "Bus control") | | | | |
| $P5_4$ | I/O port | $\overline{HLDA}$ | $\overline{HLDA}$ | $\overline{HLDA}$ | $\overline{HLDA}$ | $\overline{HLDA}$ |
| $P5_5$ | I/O port | $\overline{HOLD}$ | $\overline{HOLD}$ | $\overline{HOLD}$ | $\overline{HOLD}$ | $\overline{HOLD}$ |
| $P5_6$ | I/O port | ALE | ALE | ALE | ALE | ALE |
| $P5_7$ | I/O port | $\overline{RDY}$ | $\overline{RDY}$ | $\overline{RDY}$ | $\overline{RDY}$ | $\overline{RDY}$ |

Note 1: If the entire space is of multiplexed bus in memory expansion mode, choose an 8-bit width.
The processor operates using the separate bus after reset is revoked, so the entire space multiplexed bus cannot be chosen in microprocessor mode.
$P3_1$ to $P3_7$ become a port if the entire space multiplexed bus is chosen, so only 256 bytes can be used in each chip select.
Note 2: Address bus when in separate bus mode.

## Bus Control

The following explains the signals required for accessing external devices and software waits. The signals required for accessing the external devices are valid when the processor mode is set to memory expansion mode and microprocessor mode. The software waits are valid in all processor modes.

**(1) Address bus/data bus**

The address bus consists of the 20 pins $A_0$ to $A_{19}$ for accessing the 1M bytes of address space.

The data bus consists of the pins for data I/O. When the BYTE pin is “H”, the 8 ports $D_0$ to $D_7$ function as the data bus. When BYTE is “L”, the 16 ports $D_0$ to $D_{15}$ function as the data bus.

When a change is made from single-chip mode to memory expansion mode, the value of the address bus is undefined until external memory is accessed.

**(2) Chip select signal**

The chip select signal is output using the same pins as $P4_4$ to $P4_7$. Bits 0 to 3 of the chip select control register (address $0008_{16}$) set each pin to function as a port or to output the chip select signal. The chip select control register is valid in memory expansion mode and microprocessor mode. In single-chip mode, $P4_4$ to $P4_7$ function as programmable I/O ports regardless of the value in the chip select control register.

In microprocessor mode, only $\overline{CS0}$ outputs the chip select signal after the reset state has been cancelled. $\overline{CS1}$ to $\overline{CS3}$ function as input ports. Figure 1.9.1 shows the chip select control register.

The chip select signal can be used to split the external area into as many as four blocks. Table 1.9.1 shows the external memory areas specified using the chip select signal.

**Table 1.9.1. External areas specified by the chip select signals (Note)**

| Processor mode | Chip select signal | | | |
|---|---|---|---|---|
| | $\overline{CS0}$ | $\overline{CS1}$ | $\overline{CS2}$ | $\overline{CS3}$ |
| Memory expansion mode | $30000_{16}$ to $CFFFF_{16}$ (640K bytes) | $28000_{16}$ to $2FFFF_{16}$ (32K bytes) | $08000_{16}$ to $27FFF_{16}$ (128K bytes) | $04000_{16}$ to $07FFF_{16}$ (16K bytes) |
| Microprocessor mode | $30000_{16}$ to $FFFFF_{16}$ (832K bytes) | | | |

Note :Be sure to set bit 3 (reserved bit) of processor mode register 1 to “0”.

Chip select control register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| CSR | $0008_{16}$ | $01_{16}$ |

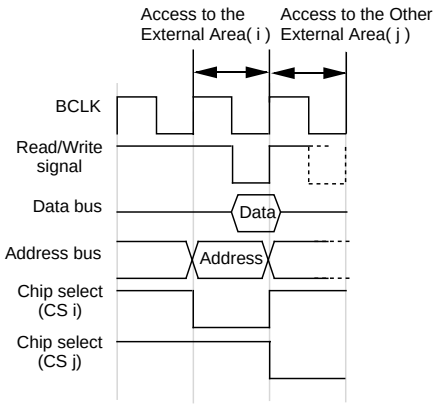
| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| CS0 | $\overline{CS0}$ output enable bit | 0 : Chip select output disabled (Normal port pin)<br>1 : Chip select output enabled | ○ | ○ |
| CS1 | $\overline{CS1}$ output enable bit | | ○ | ○ |
| CS2 | $\overline{CS2}$ output enable bit | | ○ | ○ |
| CS3 | $\overline{CS3}$ output enable bit | | ○ | ○ |
| CS0W | $\overline{CS0}$ wait bit | 0 : Wait state inserted<br>1 : No wait state | ○ | ○ |
| CS1W | $\overline{CS1}$ wait bit | | ○ | ○ |
| CS2W | $\overline{CS2}$ wait bit | | ○ | ○ |
| CS3W | $\overline{CS3}$ wait bit | | ○ | ○ |

**Figure 1.9.1. Chip select control register**

The timing of the chip select signal changing to “L”(active) is synchronized with the address bus. But the timing of the chip select signal changing to “H” depends on the area which will be accessed in the next cycle. Figure 1.9.2 shows the output example of the address bus and chip select signal.

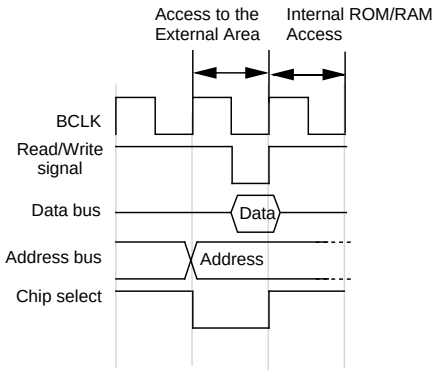
Example 1) After access the external area, both the address signal and the chip select signal change concurrently in the next cycle.

In this example, after access to the external area(i), an access to the area indicated by the other chip select signal(j) will occur in the next cycle. In this case, both the address bus and the chip select signal change between the two cycles.



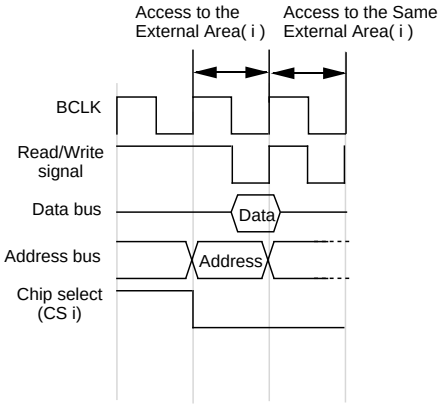

Example 2) After access the external area, only the chip select signal changes in the next cycle (the address bus does not change).

In this example, an access to the internal ROM or the internal RAM in the next cycle will occur, after access to the external area. In this case, the chip select signal changes between the two cycles, but the address does not change.



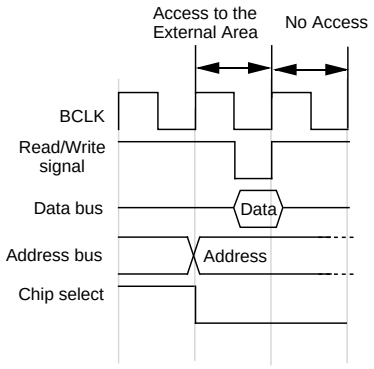

Example 3) After access the external area, only the address bus changes in the next cycle (the chip select signal does not change).

In this example, after access to the external area(i), an access to the area indicated by the same chip select signal(i) will occur in the next cycle. In this case, the address bus changes between the two cycles, but the chip select signal does not change.




Example 4) After access the external area, either the address signal and the chip select signal do not change in the next cycle.

In this example, any access to any area does not occur in the next cycle (either instruction prefetch does not occur). In this case,either the address bus and chip select signal do not change between the two cycles.




Note : These examples show the address bus and chip select signal within the successive two cycles.
According to the combination of these examples, the chip select can be elongated to over 2cycles.

**Figure 1.9.2. Output Examples about Address Bus and Chip Select Signal (Separated Bus without Wait)**

### (3) Read/write signals

With a 16-bit data bus (BYTE pin =“L”), bit 2 of the processor mode register 0 (address $0004_{16}$) select the combinations of $\overline{RD}$, $\overline{BHE}$, and $\overline{WR}$ signals or $\overline{RD}$, $\overline{WRL}$, and $\overline{WRH}$ signals. With an 8-bit data bus (BYTE pin = “H”), use the combination of $\overline{RD}$, $\overline{WR}$, and $\overline{BHE}$ signals. (Set bit 2 of the processor mode register 0 (address $0004_{16}$) to “0”.) Tables 1.9.2 and 1.9.3 show the operation of these signals.
After a reset has been cancelled, the combination of $\overline{RD}$, $\overline{WR}$, and $\overline{BHE}$ signals is automatically selected. When switching to the $\overline{RD}$, $\overline{WRL}$, and $\overline{WRH}$ combination, do not write to external memory until bit 2 of the processor mode register 0 (address $0004_{16}$) has been set (Note).

Note: Before attempting to change the contents of the processor mode register 0, set bit 1 of the protect register (address $000A_{16}$) to “1”.

**Table 1.9.2. Operation of $\overline{RD}$, $\overline{WRL}$, and $\overline{WRH}$ signals**

| Data bus width | $\overline{RD}$ | $\overline{WRL}$ | $\overline{WRH}$ | Status of external data bus |
|---|---|---|---|---|
| 16-bit (BYTE = “L”) | L | H | H | Read data |
| | H | L | H | Write 1 byte of data to even address |
| | H | H | L | Write 1 byte of data to odd address |
| | H | L | L | Write data to both even and odd addresses |

**Table 1.9.3. Operation of $\overline{RD}$, $\overline{WR}$, and $\overline{BHE}$ signals**

| Data bus width | $\overline{RD}$ | $\overline{WR}$ | $\overline{BHE}$ | A0 | Status of external data bus |
|---|---|---|---|---|---|
| 16-bit (BYTE = “L”) | H | L | L | H | Write 1 byte of data to odd address |
| | L | H | L | H | Read 1 byte of data from odd address |
| | H | L | H | L | Write 1 byte of data to even address |
| | L | H | H | L | Read 1 byte of data from even address |
| | H | L | L | L | Write data to both even and odd addresses |
| | L | H | L | L | Read data from both even and odd addresses |
| 8-bit (BYTE = “H”) | H | L | Not used | H / L | Write 1 byte of data |
| | L | H | Not used | H / L | Read 1 byte of data |

### (4) ALE signal

The ALE signal latches the address when accessing the multiplex bus space. Latch the address when the ALE signal falls.

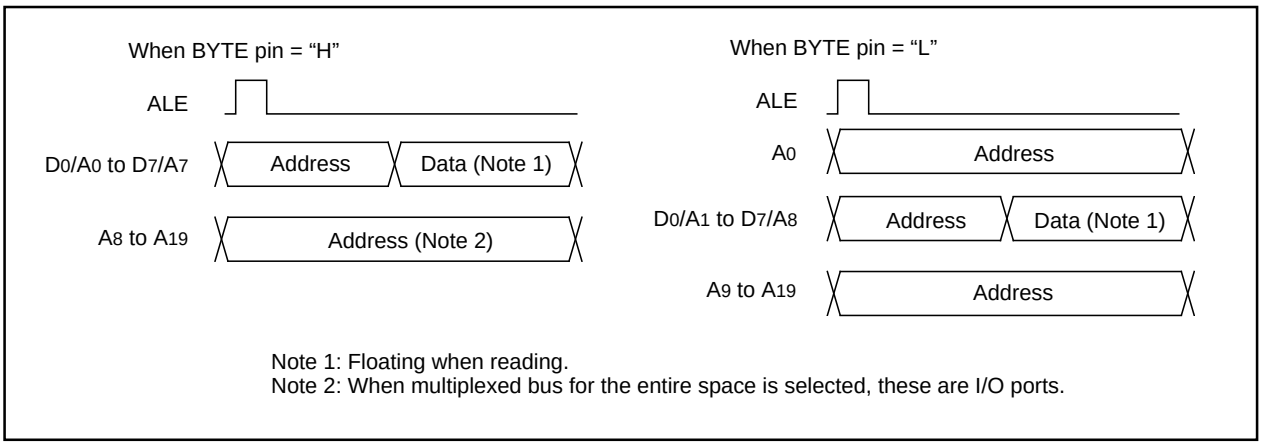



**Figure 1.9.3. ALE signal and address/data bus**

### (5) The $\overline{RDY}$ signal

$\overline{RDY}$ is a signal that facilitates access to an external device that requires long access time. As shown in Figure 1.9.4, if an “L” is being input to the $\overline{RDY}$ at the BCLK falling edge, the bus turns to the wait state. If an “H” is being input to the $\overline{RDY}$ pin at the BCLK falling edge, the bus cancels the wait state. Table 1.9.4 shows the state of the microcomputer with the bus in the wait state, and Figure 1.9.4 shows an example in which the $\overline{RD}$ signal is prolonged by the $\overline{RDY}$ signal.

The $\overline{RDY}$ signal is valid when accessing the external area during the bus cycle in which bits 4 to 7 of the chip select control register (address $0008_{16}$) are set to “0”. The $\overline{RDY}$ signal is invalid when setting “1” to all bits 4 to 7 of the chip select control register (address $0008_{16}$), but the $\overline{RDY}$ pin should be treated as properly as in non-using.

**Table 1.9.4. Microcomputer status in wait state (Note)**

| Item | Status |
|---|---|
| Oscillation | On |
| R/$\overline{W}$ signal, address bus, data bus, $\overline{CS}$<br>ALE signal, $\overline{HLDA}$, programmable I/O ports | Maintain status when $\overline{RDY}$ signal received |
| Internal peripheral circuits | On |

Note: The $\overline{RDY}$ signal cannot be received immediately prior to a software wait.

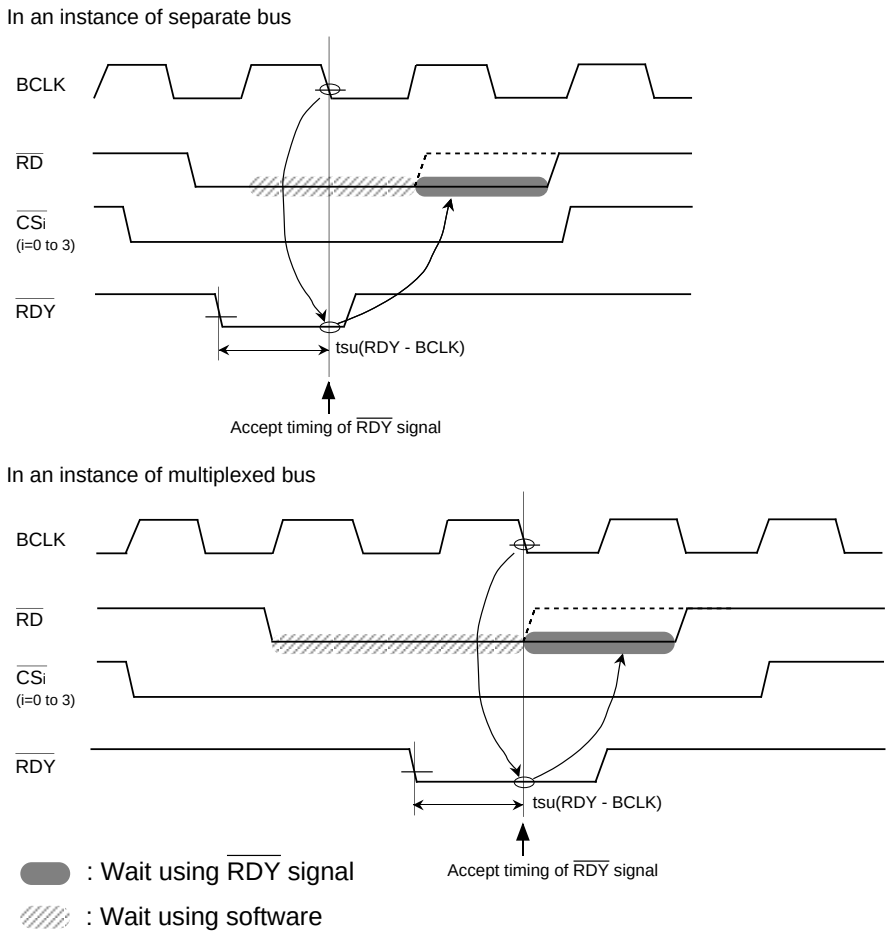



**Figure 1.9.4. Example of $\overline{RD}$ signal extended by $\overline{RDY}$ signal**

### (6) Hold signal

The hold signal is used to transfer the bus privileges from the CPU to the external circuits. Inputting "L" to the $\overline{\text{HOLD}}$ pin places the microcomputer in the hold state at the end of the current bus access. This status is maintained and "L" is output from the $\overline{\text{HLDA}}$ pin as long as "L" is input to the $\overline{\text{HOLD}}$ pin. Table 1.9.5 shows the microcomputer status in the hold state.

Bus-using priorities are given to $\overline{\text{HOLD}}$, DMAC, and CPU in order of decreasing precedence.

$\overline{\text{HOLD}}$ > DMAC > CPU

**Figure 1.9.5. Bus-using priorities**

**Table 1.9.5. Microcomputer status in hold state**

| Item | | Status |
|---|---|---|
| Oscillation | | ON |
| R/$\overline{\text{W}}$ signal, address bus, data bus, $\overline{\text{CS}}$, $\overline{\text{BHE}}$ | | Floating |
| Programmable I/O ports | P0, P1, P2, P3, P4, P5 | Floating |
| | P6, P7, P8, P9, P10 | Maintains status when hold signal is received |
| $\overline{\text{HLDA}}$ | | Output "L" |
| Internal peripheral circuits | | ON (but watchdog timer stops) |
| ALE signal | | Undefined |

### (7) External bus status when the internal area is accessed

Table 1.9.6 shows the external bus status when the internal area is accessed.

**Table 1.9.6. External bus status when the internal area is accessed**

| Item | | SFR accessed | Internal ROM/RAM accessed |
|---|---|---|---|
| Address bus | | Address output | Maintain status before accessed address of external area |
| Data bus | When read | Floating | Floating |
| | When write | Output data | Undefined |
| $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{WRL}}$, $\overline{\text{WRH}}$ | | $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{WRL}}$, $\overline{\text{WRH}}$ output | Output "H" |
| $\overline{\text{BHE}}$ | | $\overline{\text{BHE}}$ output | Maintain status before accessed status of external area |
| $\overline{\text{CS}}$ | | Output "H" | Output "H" |
| ALE | | Output "L" | Output "L" |

### (8) BCLK output

The user can choose the BCLK output by use of bit 7 of processor mode register 0 ($0004_{16}$) (Note). When set to “1”, the output floating.

Note: Before attempting to change the contents of the processor mode register 0, set bit 1 of the protect register (address $000A_{16}$) to “1”.

### (9) Software wait

A software wait can be inserted by setting the wait bit (bit 7) of the processor mode register 1 (address $0005_{16}$) (Note) and bits 4 to 7 of the chip select control register (address $0008_{16}$).

A software wait is inserted in the internal ROM/RAM area and in the external memory 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 or three BCLK cycles. After the microcomputer has been reset, this bit defaults to “0”. When set to “1”, a wait is applied to all memory areas (two or three BCLK cycles), regardless of the contents of bits 4 to 7 of the chip select control register. Set this bit after referring to the recommended operating conditions (main clock input oscillation frequency) of the electric characteristics. However, when the user is using the $\overline{RDY}$ signal, the relevant bit in the chip select control register’s bits 4 to 7 must be set to “0”.

When the wait bit of the processor mode register 1 is “0”, software waits can be set independently for each of the 4 areas selected using the chip select signal. Bits 4 to 7 of the chip select control register correspond to chip selects $\overline{CS0}$ to $\overline{CS3}$. When one of these bits is set to “1”, the bus cycle is executed in one BCLK cycle. When set to “0”, the bus cycle is executed in two or three BCLK cycles. These bits default to “0” after the microcomputer has been reset.

The SFR area is always accessed in two BCLK cycles regardless of the setting of these control bits. Also, insert a software wait if using the multiplex bus to access the external memory area.

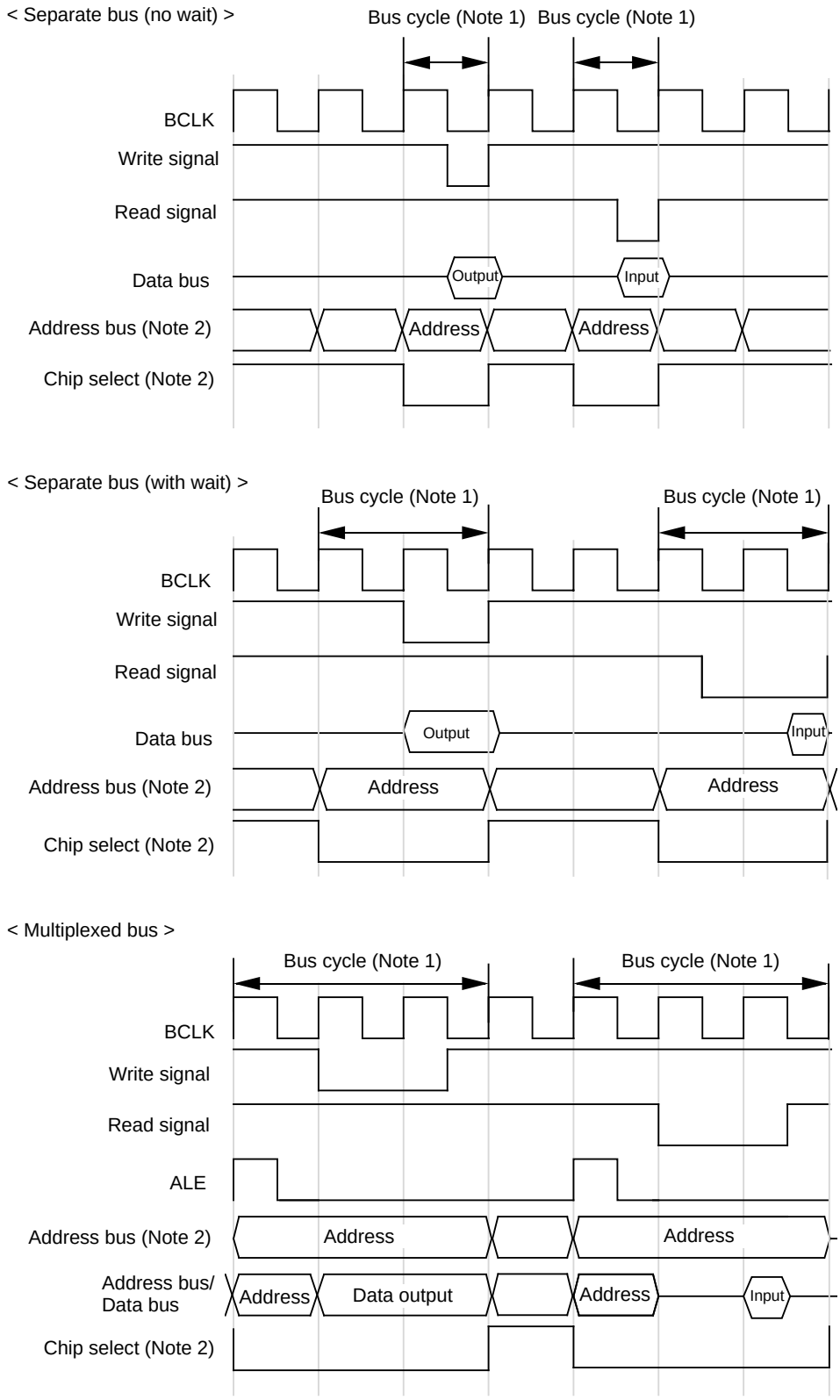
Table 1.9.7 shows the software wait and bus cycles. Figure 1.9.6 shows example of bus timing when using software waits.

Note: Before attempting to change the contents of the processor mode register 1, set bit 1 of the protect register (address $000A_{16}$) to “1”.

**Table 1.9.7. Software waits and bus cycles**

| Area | Bus status | Wait bit | Bits 4 to 7 of chip select control register | Bus cycle |
|---|---|---|---|---|
| SFR | ——— | Invalid | Invalid | 2 BCLK cycles |
| Internal ROM/RAM | ——— | 0 | Invalid | 1 BCLK cycle |
| | ——— | 1 | Invalid | 2 BCLK cycles |
| External memory area | Separate bus | 0 | 1 | 1 BCLK cycle |
| | Separate bus | 0 | 0 | 2 BCLK cycles |
| | Separate bus | 1 | 0 (Note) | 2 BCLK cycles |
| | Multiplex bus | 0 | 0 | 3 BCLK cycles |
| | Multiplex bus | 1 | 0 (Note) | 3 BCLK cycles |

Note: When using the $\overline{RDY}$ signal, always set to “0”.




Note 1: These example timing charts indicate bus cycle length.
After this bus cycle sometimes come read and write cycles in succession.
Note 2: The address bus and chip select may be extended depending on the CPU status such as that of the instruction queue buffer.

**Figure 1.9.6. Typical bus timings using software wait**

## 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 1.10.1. Main clock and sub-clock generating circuits**

| | Main clock generating circuit | Sub-clock generating circuit |
|---|---|---|
| Use of clock | • CPU's operating clock source<br>• Internal peripheral units' operating clock source | • CPU's operating clock source<br>• Timer A/B's count clock source |
| 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 1.10.1 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 1.10.2 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 1.10.1 and 1.10.2 vary with each oscillator used. Use the values recommended by the manufacturer of your oscillator.

Note: Insert a damping resistor if required. The resistance will vary depending on the oscillator and the oscillation drive capacity setting. Use the value recommended by the maker of the oscillator.
When the oscillation drive capacity is set to low, check that oscillation is stable. Also, if the oscillator manufacturer's data sheet specifies that a feedback resistor be added external to the chip, insert a feedback resistor between XIN and XOUT following the instruction.

**Figure 1.10.1. Examples of main clock**

Note: Insert a damping resistor if required. The resistance will vary depending on the oscillator and the oscillation drive capacity setting. Use the value recommended by the maker of the oscillator.
When the oscillation drive capacity is set to low, check that oscillation is stable. Also, if the oscillator manufacturer's data sheet specifies that a feedback resistor be added external to the chip, insert a feedback resistor between XCIN and XCOUT following the instruction.

**Figure 1.10.2. Examples of sub-clock**

## Clock Control

Figure 1.10.3 shows the block diagram of the clock generating circuit.

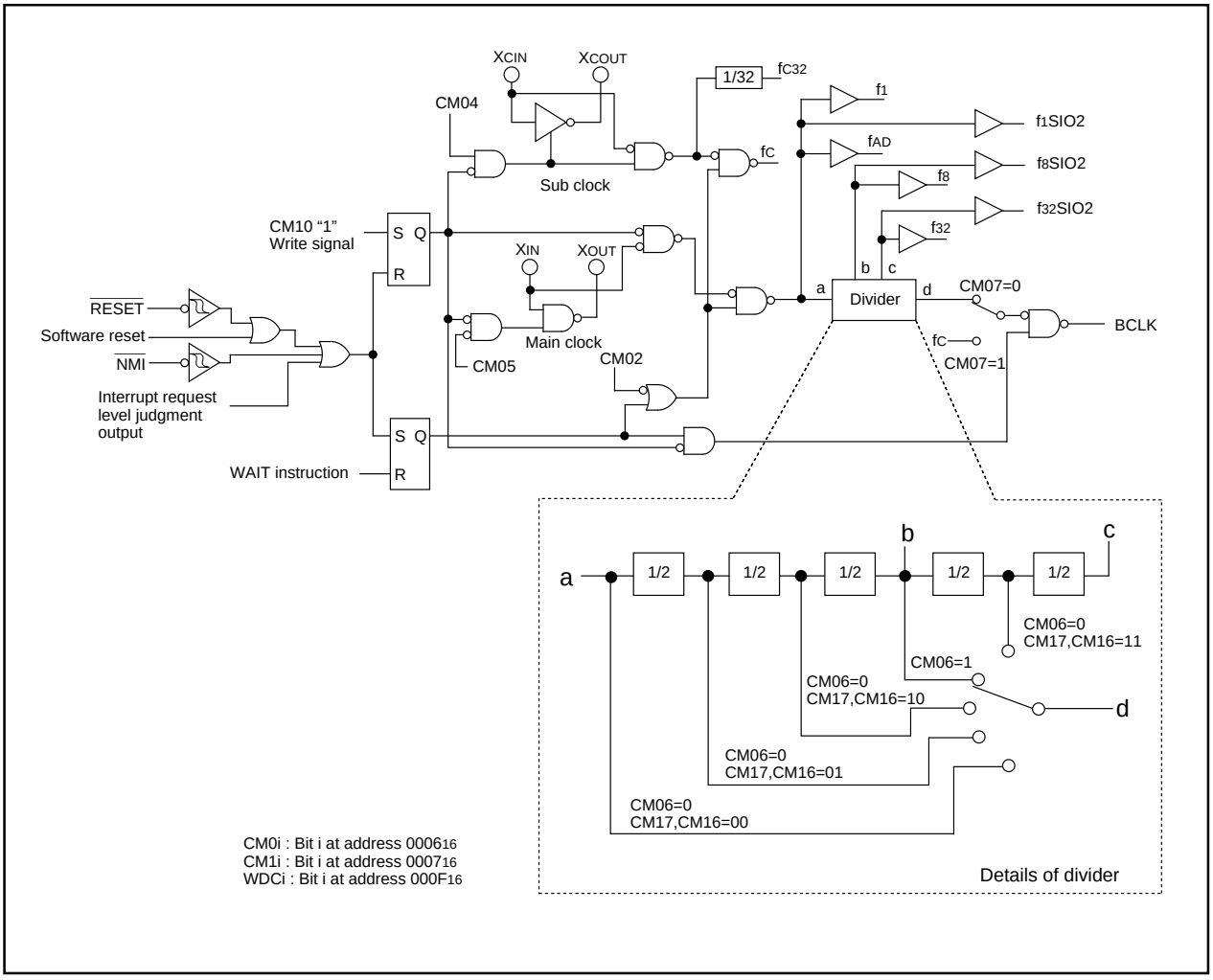



**Figure 1.10.3. Clock generating circuit**

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 $0006_{16}$). 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 $X_{IN}$-$X_{OUT}$ drive capacity select bit (bit 5 at address $0007_{16}$). 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 $X_C$ select bit (bit 4 at address $0006_{16}$), the sub-clock can be selected as the BCLK by using the system clock select bit (bit 7 at address $0006_{16}$). 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 $X_{CIN}$-$X_{COUT}$ drive capacity select bit (bit 3 at address $0006_{16}$). 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 $X_{CIN}$/$X_{COUT}$ is used, set ports $P8_6$ and $P8_7$ 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 $0004_{16}$) in the memory expansion and the microprocessor modes.

The main clock division select bit 0(bit 6 at address $0006_{16}$) 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 $0006_{16}$) 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.

Figure 1.10.4 shows the system clock control registers 0 and 1.

System clock control register 0 (Note 1)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| CM0 | $000_{16}$ | $48_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| CM00 | Clock output function select bit (Valid only in single-chip mode) | b1 b0<br>0 0 : I/O port $P5_7$<br>0 1 : $f_C$ output<br>1 0 : $f_8$ output<br>1 1 : $f_{32}$ output | ○ | ○ |
| CM01 | | | ○ | ○ |
| CM02 | WAIT peripheral function clock stop bit | 0 : Do not stop peripheral function clock in wait mode<br>1 : Stop peripheral function clock in wait mode (Note 8) | ○ | ○ |
| CM03 | $X_{CIN}$-$X_{COUT}$ drive capacity select bit (Note 2) | 0 : LOW<br>1 : HIGH | ○ | ○ |
| CM04 | Port $X_C$ select bit | 0 : I/O port<br>1 : $X_{CIN}$-$X_{COUT}$ generation (Note 9) | ○ | ○ |
| CM05 | Main clock ($X_{IN}$-$X_{OUT}$) stop bit (Note 3, 4, 5) | 0 : On<br>1 : Off | ○ | ○ |
| CM06 | Main clock division select bit 0 (Note 7) | 0 : CM16 and CM17 valid<br>1 : Division by 8 mode | ○ | ○ |
| CM07 | System clock select bit (Note 6) | 0 : $X_{IN}$, $X_{OUT}$<br>1 : $X_{CIN}$, $X_{COUT}$ | ○ | ○ |

Note 1: Set bit 0 of the protect register (address $000A_{16}$) to "1" before writing to this register.
Note 2: Changes to "1" when shifting to stop mode and at a reset.
Note 3: When entering low power dissipation mode, main clock stops by using this bit. To stop the main clock, when the sub clock oscillation is stable, set system clock select bit (CM07) to "1" before setting this bit to "1".
Note 4: When inputting external clock, only clock oscillation buffer is stopped and clock input is acceptable.
Note 5: If this bit is set to "1", $X_{OUT}$ turns "H". The built-in feedback resistor remains being connected, so $X_{IN}$ turns pulled up to $X_{OUT}$ ("H") via the feedback resistor.
Note 6: Set port $X_C$ select bit (CM04) to "1" and stabilize the sub-clock oscillating before setting this bit from "0" to "1". Do not write to both bits at the same time. And also, set the main clock stop bit (CM05) to "0" and stabilize the main clock oscillating before setting this bit from "1" to "0".
Note 7: 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.
Note 8: $f_{C32}$ is not included. Do not set to "1" when using low-speed or low power dissipation mode.
Note 9: When the $X_{CIN}$/$X_{COUT}$ is used, set ports $P8_6$ and $P8_7$ as the input ports without pull-up.

System clock control register 1 (Note 1)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | 0 | 0 | 0 | 0 | |

| Symbol | Address | When reset |
|---|---|---|
| CM1 | $0007_{16}$ | $20_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| CM10 | All clock stop control bit (Note4) | 0 : Clock on<br>1 : All clocks off (stop mode) | ○ | ○ |
| Reserved bit | | Must always be set to "0" | ○ | ○ |
| Reserved bit | | Must always be set to "0" | ○ | ○ |
| Reserved bit | | Must always be set to "0" | ○ | ○ |
| Reserved bit | | Must always be set to "0" | ○ | ○ |
| CM15 | $X_{IN}$-$X_{OUT}$ drive capacity select bit (Note 2) | 0 : LOW<br>1 : HIGH | ○ | ○ |
| CM16 | Main clock division select bit 1 (Note 3) | b7 b6<br>0 0 : No division mode<br>0 1 : Division by 2 mode<br>1 0 : Division by 4 mode<br>1 1 : Division by 16 mode | ○ | ○ |
| CM17 | | | | |

Note 1: Set bit 0 of the protect register (address $000A_{16}$) to "1" before writing to this register.
Note 2: 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.
Note 3: Can be selected when bit 6 of the system clock control register 0 (address $0006_{16}$) is "0". If "1", division mode is fixed at 8.
Note 4: If this bit is set to "1", $X_{OUT}$ turns "H", and the built-in feedback resistor is cut off. $X_{CIN}$ and $X_{COUT}$ turn high-impedance state.

**Figure 1.10.4. Clock control registers 0 and 1**

## Clock Output

In single-chip mode, the clock output function select bits (bits 0 and 1 at address $0006_{16}$) enable $f_8$, $f_{32}$, or fc to be output from the $P5_7$/$CLK_{OUT}$ pin. When the WAIT peripheral function clock stop bit (bit 2 at address $0006_{16}$) is set to “1”, the output of $f_8$ and $f_{32}$ stops when a WAIT instruction is executed.

## Stop Mode

Writing “1” to the all-clock stop control bit (bit 0 at address $0007_{16}$) stops all oscillation and the microcomputer enters stop mode. In stop mode, the content of the internal RAM is retained provided that $V_{CC}$ remains above 2V.

Because the oscillation , BCLK, $f_1$ to $f_{32}$, $f_{1SIO2}$ to $f_{32SIO2}$, $f_C$, $f_{C32}$, and $f_{AD}$ 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 UARTi(i = 0 to 2), SI/O3,4 functions provided an external clock is selected. Table 1.10.2 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 $\overline{NMI}$ 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 $0006_{16}$) is set to “1”. When shifting from low-speed/low power dissipation mode to stop mode, the value before stop mode is retained.

**Table 1.10.2. Port status during stop mode**

| Pin | | Memory expansion mode<br>Microprocessor mode | Single-chip mode |
|---|---|---|---|
| Address bus, data bus, $\overline{CS0}$ to $\overline{CS3}$, $\overline{BHE}$ | | Retains status before stop mode | |
| $\overline{RD}$, $\overline{WR}$, $\overline{WRL}$, $\overline{WRH}$ | | “H” | |
| $\overline{HLDA}$, BCLK | | “H” | |
| ALE | | “H” | |
| Port | | Retains status before stop mode | Retains status before stop mode |
| $CLK_{OUT}$ | When fc selected | Valid only in single-chip mode | “H” |
| | When $f_8$, $f_{32}$ selected | Valid only in single-chip mode | 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 $f_{C32}$ does not stop so that the peripherals using $f_{C32}$ 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 1.10.3 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 $\overline{NMI}$ interrupt is used to cancel wait mode, change the priority level of all interrupt to 0,then shift to wait mode.

**Table 1.10.3. Port status during wait mode**

| Pin | | Memory expansion mode<br>Microprocessor mode | Single-chip mode |
|---|---|---|---|
| Address bus, data bus, $\overline{CS0}$ to $\overline{CS3}$, $\overline{BHE}$ | | Retains status before wait mode | |
| $\overline{RD}$, $\overline{WR}$, $\overline{WRL}$, $\overline{WRH}$ | | “H” | |
| $\overline{HLDA}$,BCLK | | “H” | |
| ALE | | “H” | |
| Port | | Retains status before wait mode | Retains status before wait mode |
| $CLK_{OUT}$ | When $f_C$ selected | Valid only in single-chip mode | Does not stop |
| | When $f_8$, $f_{32}$ selected | Valid only in single-chip mode | Does not stop when the WAIT peripheral function clock stop bit is “0”.<br>When the WAIT peripheral function clock stop bit is “1”, the status immediately prior to entering wait mode is maintained. |

## Status Transition of BCLK

Power dissipation can be reduced and low-voltage operation achieved by changing the count source for BCLK. Table 1.10.4 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 $0006_{16}$) 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

$f_C$ 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

$f_C$ is the BCLK and the main clock is stopped.

Note : Before the count source for BCLK can be changed from $X_{IN}$ to $X_{CIN}$ 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 1.10.4. 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 |

## 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 subclock. 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 subclock. The only peripheral functions that operate are those with the subclock selected as the count source.

**(b) Wait mode**

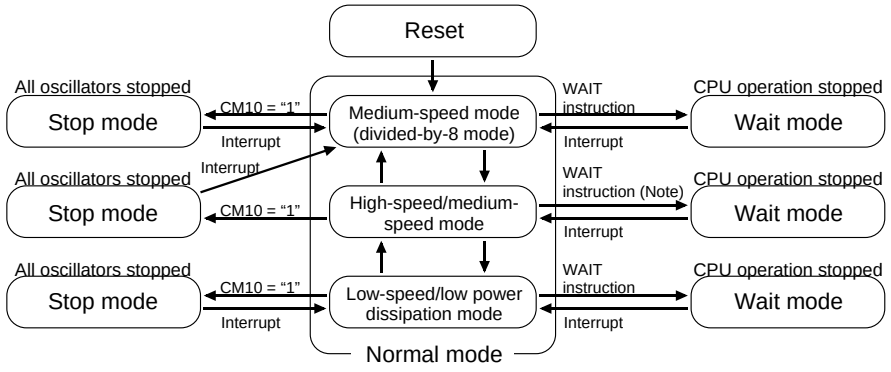
The CPU operation is stopped. The oscillators do not stop.

**(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.

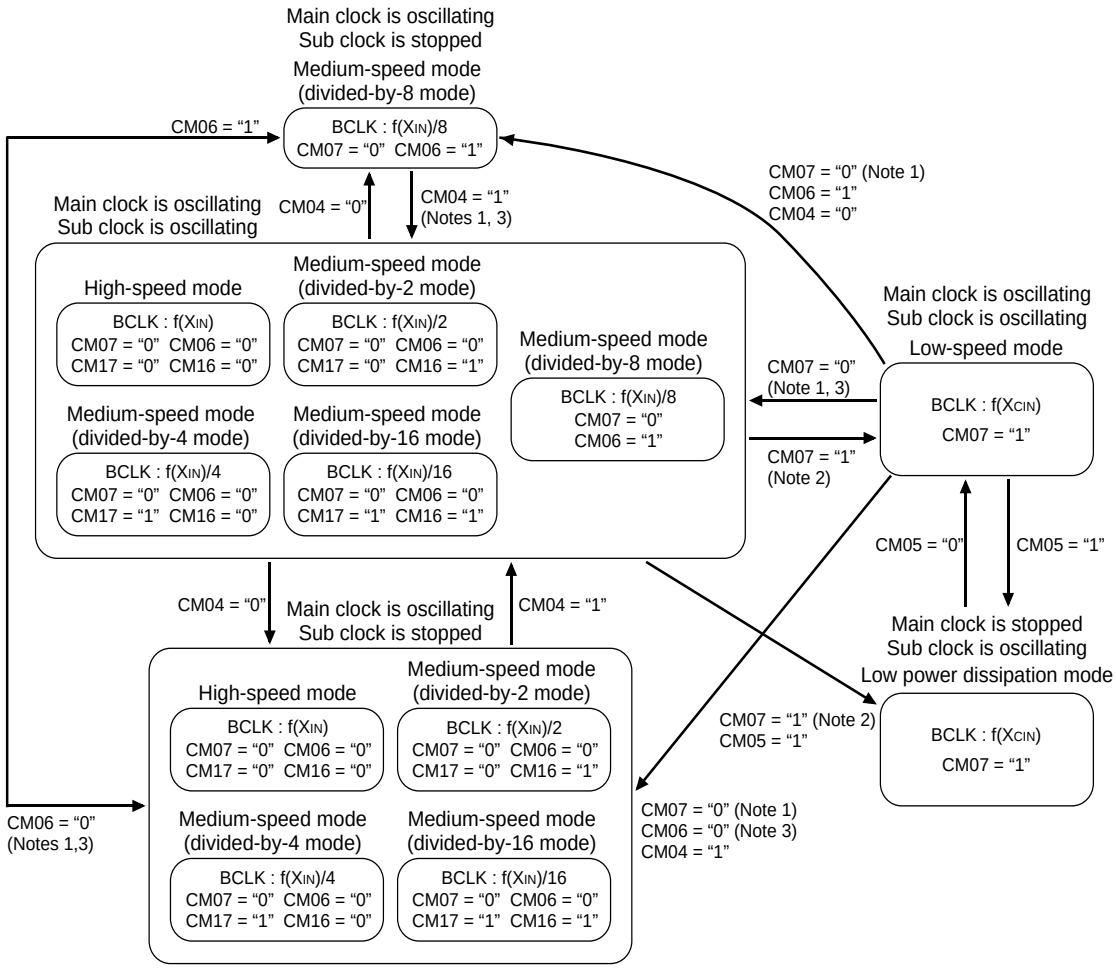
Figure 1.10.5 is the state transition diagram of the above modes.

**Transition of stop mode, wait mode**




Note: When shifting to the wait mode in operating with the main clock, set up the bit 6 (CM06) of the system clock control register 0, and the bits 6 and 7 (CM16,CM17) of the system clock control register 1, and execute WAIT instruction after selecting the main clock operation divided by 8 or 16.

**Transition of normal mode**




Note 1: Switch clock after oscillation of main clock is sufficiently stable.
Note 2: Switch clock after oscillation of sub clock is sufficiently stable.
Note 3: Change CM06 after changing CM17 and CM16.
Note 4: Transit in accordance with arrow.

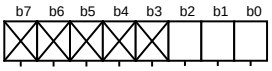
**Figure 1.10.5. State transition diagram of Power control mode**

## 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 1.10.6 shows the protect register. The values in the processor mode register 0 (address $0004_{16}$), processor mode register 1 (address $0005_{16}$), system clock control register 0 (address $0006_{16}$), system clock control register 1 (address $0007_{16}$), port P9 direction register (address $03F3_{16}$) , SI/O3 control register (address $0362_{16}$) and SI/O4 control register (address $0366_{16}$) 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/Oi control register (i=3,4) write-enable bit (bit 2 at address $000A_{16}$), 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 $000A_{16}$) and processor mode register 0 and 1 write-enable bit (bit 1 at $000A_{16}$) 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”.

Protect register




| Symbol | Address | When reset |
|---|---|---|
| PRCR | $000A_{16}$ | $XXXXX000_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| PRC0 | Enables writing to system clock control registers 0 and 1 (addresses $0006_{16}$ and $0007_{16}$) | 0 : Write-inhibited<br>1 : Write-enabled | ○ | ○ |
| PRC1 | Enables writing to processor mode registers 0 and 1 (addresses $0004_{16}$ and $0005_{16}$) | 0 : Write-inhibited<br>1 : Write-enabled | ○ | ○ |
| PRC2 | Enables writing to port P9 direction register (address $03F3_{16}$) and SI/Oi control register (i=3,4) (addresses $0362_{16}$ and $0366_{16}$) (Note) | 0 : Write-inhibited<br>1 : Write-enabled | ○ | ○ |
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be indeterminate. | | | — | — |

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 be reset by the program.

**Figure 1.10.6. Protect register**

## Overview of Interrupt

### Type of Interrupts

Figure 1.11.1 lists the types of interrupts.

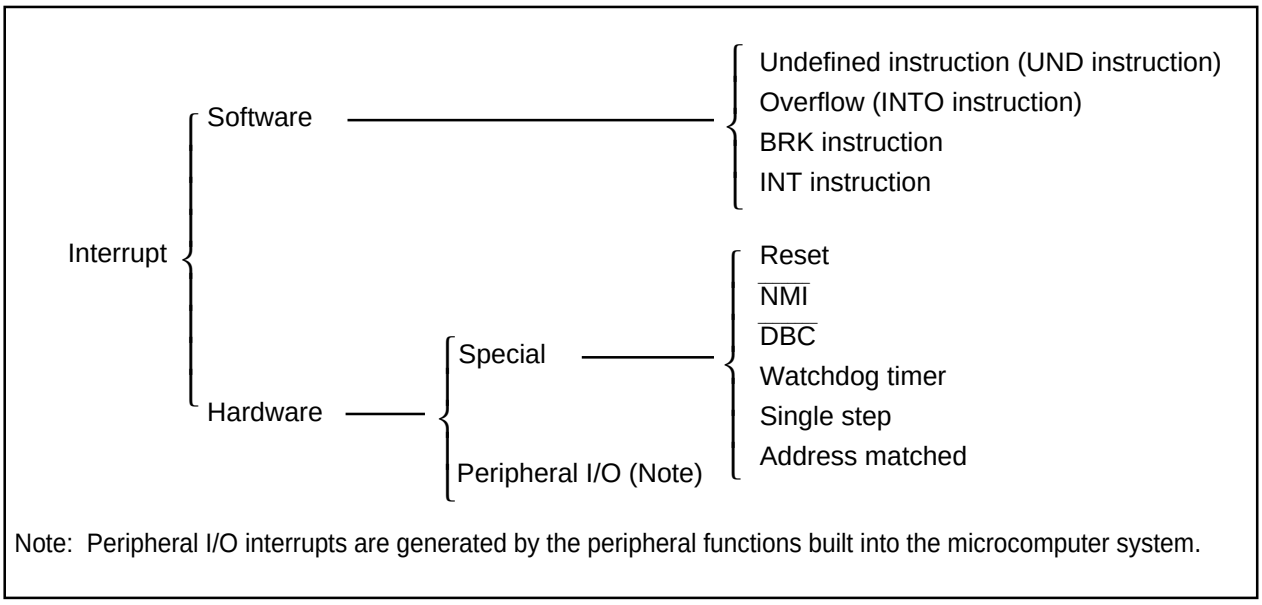



**Figure 1.11.1. Classification of interrupts**

- 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 interrupt**
  An INT interrupt occurs when specifying 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 $\overline{\text{INT5}}$ 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 1.11.2 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.

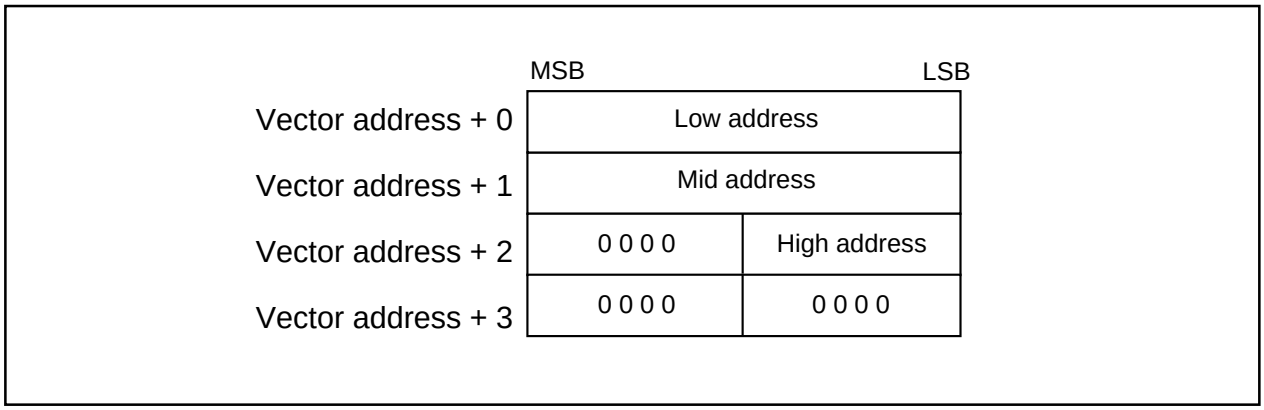



**Figure 1.11.2. Format for specifying interrupt vector addresses**

- **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 FFFDC16 to FFFFF16. One vector table comprises four bytes. Set the first address of interrupt routine in each vector table. Table 1.11.1 shows the interrupts assigned to the fixed vector tables and addresses of vector tables.

**Table 1.11.1. Interrupts assigned to the fixed vector tables and addresses of vector tables**

| Interrupt source | Vector table addresses<br>Address (L) to address (H) | Remarks |
|---|---|---|
| Undefined instruction | FFFDC16 to FFFDF16 | Interrupt on UND instruction |
| Overflow | FFFE016 to FFFE316 | Interrupt on INTO instruction |
| BRK instruction | FFFE416 to FFFE716 | If the vector contains FF16, program execution starts from the address shown by the vector in the variable vector table |
| Address match | FFFE816 to FFFEB16 | There is an address-matching interrupt enable bit |
| Single step (Note) | FFFEC16 to FFFEF16 | Do not use |
| Watchdog timer | FFFF016 to FFFF316 | |
| $\overline{\text{DBC}}$ (Note) | FFFF416 to FFFF716 | Do not use |
| $\overline{\text{NMI}}$ | FFFF816 to FFFFB16 | External interrupt by input to $\overline{\text{NMI}}$ pin |
| Reset | FFFFC16 to FFFFF16 | |

Note: Interrupts used for debugging purposes only.

- **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 1.11.2 shows the interrupts assigned to the variable vector tables and addresses of vector tables.

**Table 1.11.2. Interrupts assigned to the variable vector tables and addresses of vector tables**

| Software interrupt number | Vector table address<br>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) | $\overline{\text{INT3}}$ | |
| 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 2) | |
| Software interrupt number 9 | +36 to +39 (Note 1) | SI/O3/$\overline{\text{INT4}}$ (Note 2) | |
| 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 3) | |
| Software interrupt number 16 | +64 to +67 (Note 1) | UART2 receive/ACK (Note 3) | |
| 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<br>to<br>Software interrupt number 63 | +128 to +131 (Note 1)<br>to<br>+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: It is selected by interrupt request cause bit (bit 6, 7 in address $035F_{16}$ ).
Note 3: When IIC mode is selected, NACK and ACK interrupts are selected.

## Interrupt Control

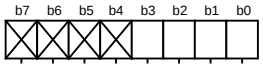
Descriptions are given here regarding how to enable or disable maskable interrupts and how to set the priority to be accepted. What is described here does not apply to non-maskable interrupts.

Enable or disable a maskable interrupt using the interrupt enable flag (I flag), interrupt priority level selection 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 1.11.3 shows the memory map of the interrupt control registers.

Interrupt control register (Note 2)

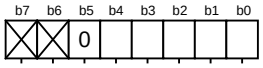
| Symbol | Address | When reset |
|---|---|---|
| TBiIC(i=3 to 5) | $0045_{16}$ to $0047_{16}$ | $XXXXX000_2$ |
| BCNIC | $004A_{16}$ | $XXXXX000_2$ |
| DMiIC(i=0, 1) | $004B_{16}$, $004C_{16}$ | $XXXXX000_2$ |
| KUPIC | $004D_{16}$ | $XXXXX000_2$ |
| ADIC | $004E_{16}$ | $XXXXX000_2$ |
| SiTIC(i=0 to 2) | $0051_{16}$, $0053_{16}$, $004F_{16}$ | $XXXXX000_2$ |
| SiRIC(i=0 to 2) | $0052_{16}$, $0054_{16}$, $0050_{16}$ | $XXXXX000_2$ |
| TAiIC(i=0 to 4) | $0055_{16}$ to $0059_{16}$ | $XXXXX000_2$ |
| TBiIC(i=0 to 2) | $005A_{16}$ to $005C_{16}$ | $XXXXX000_2$ |




| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| ILVL0 | Interrupt priority level select bit | b2 b1 b0<br>0 0 0 : Level 0 (interrupt disabled)<br>0 0 1 : Level 1<br>0 1 0 : Level 2<br>0 1 1 : Level 3<br>1 0 0 : Level 4<br>1 0 1 : Level 5<br>1 1 0 : Level 6<br>1 1 1 : Level 7 | ○ | ○ |
| ILVL1 | | | ○ | ○ |
| ILVL2 | | | ○ | ○ |
| IR | Interrupt request bit | 0 : Interrupt not requested<br>1 : Interrupt requested | ○ | ○ (Note 1) |
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be indeterminate. | | | – | – |

Note 1: This bit can only be accessed for reset (= 0), but cannot be accessed for set (= 1).
Note 2: To rewrite the interrupt control register, do so at a point that does not generate the interrupt request for that register. For details, see the precautions for interrupts.

| Symbol | Address | When reset |
|---|---|---|
| INTiIC(i=3) | $0044_{16}$ | $XX00X000_2$ |
| SiIC/INTjIC (i=4, 3) | $0048_{16}$, $0049_{16}$ | $XX00X000_2$ |
| (j=5, 4) | $0048_{16}$, $0049_{16}$ | $XX00X000_2$ |
| INTiIC(i=0 to 2) | $005D_{16}$ to $005F_{16}$ | $XX00X000_2$ |




| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| ILVL0 | Interrupt priority level select bit | b2 b1 b0<br>0 0 0 : Level 0 (interrupt disabled)<br>0 0 1 : Level 1<br>0 1 0 : Level 2<br>0 1 1 : Level 3<br>1 0 0 : Level 4<br>1 0 1 : Level 5<br>1 1 0 : Level 6<br>1 1 1 : Level 7 | ○ | ○ |
| ILVL1 | | | ○ | ○ |
| ILVL2 | | | ○ | ○ |
| IR | Interrupt request bit | 0: Interrupt not requested<br>1: Interrupt requested | ○ | ○ (Note 1) |
| POL | Polarity select bit | 0 : Selects falling edge<br>1 : Selects rising edge | ○ | ○ |
| Reserved bit | | Must always be set to “0” | ○ | ○ |
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be indeterminate. | | | – | – |

Note 1: This bit can only be accessed for reset (= 0), but cannot be accessed for set (= 1).
Note 2: To rewrite the interrupt control register, do so at a point that does not generate the interrupt request for that register. For details, see the precautions for interrupts.

**Figure 1.11.3. 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 1.11.3 shows the settings of interrupt priority levels and Table 1.11.4 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 1.11.3. Settings of interrupt priority levels**

| Interrupt priority level select bit (b2 b1 b0) | Interrupt priority level | Priority order |
|---|---|---|
| 0 0 0 | Level 0 (interrupt disabled) | ——— |
| 0 0 1 | Level 1 | Low |
| 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 | High |

**Table 1.11.4. Interrupt levels enabled according to the contents of the IPL**

| IPL ($IPL_2$ $IPL_1$ $IPL_0$) | Enabled interrupt priority levels |
|---|---|
| 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 changing an interrupt control register in a sate of interrupts being disabled, please read the following precautions on instructions used before changing the register.

(1) Changing a non-interrupt request bit

If an interrupt request for an interrupt control register is generated during an instruction to rewrite the register is being executed, there is a case that the interrupt request bit is not set and consequently the interrupt is ignored. This will depend on the instruction. If this creates problems, use the below instructions to change the register.

Instructions : AND, OR, BCLR, BSET

(2) Changing the interrupt request bit

When attempting to clear the interrupt request bit of an interrupt control register, the interrupt request bit is not cleared sometimes. This will depend on the instruction. If this creates problems, use the below instructions to change the register.

Instructions : MOV

## 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:

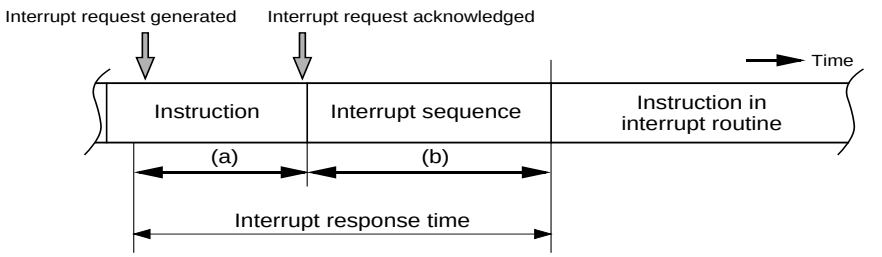
(1) CPU gets the interrupt information (the interrupt number and interrupt request level) by reading address $00000_{16}$. After this, the corresponding interrupt request bit becomes “0”.
(2) Saves the content of the flag register (FLG) as it was immediately before the start of interrupt sequence in 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 1.11.4 shows the interrupt response time.




(a) Time from interrupt request is generated to when the instruction then under execution is completed.
(b) Time in which the instruction sequence is executed.

**Figure 1.11.4. 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 1.11.5.

**Table 1.11.5. 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 $\overline{\text{DBC}}$ 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.

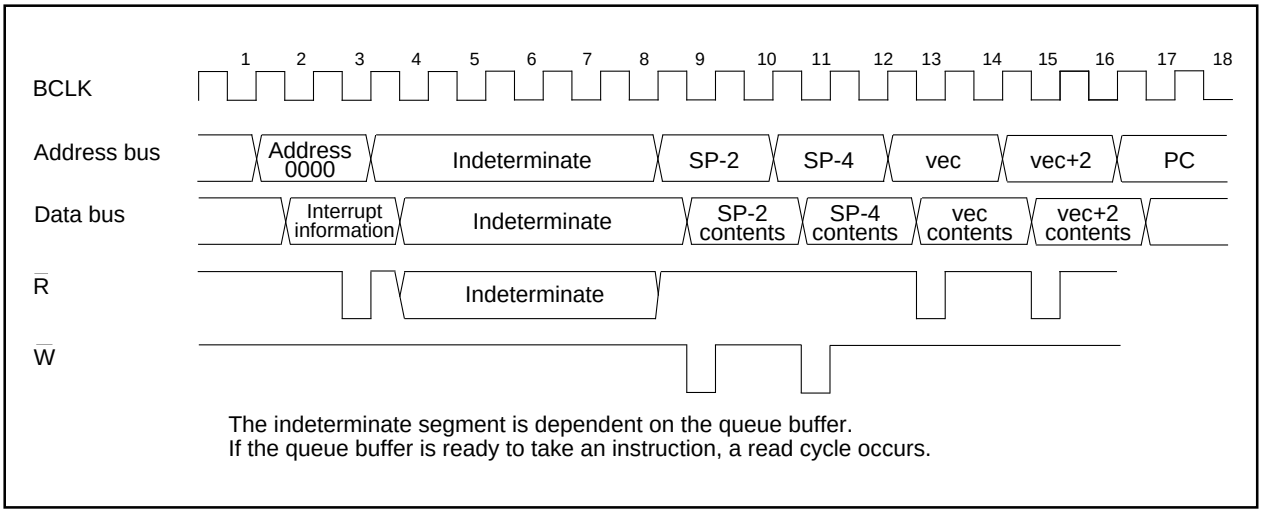



**Figure 1.11.5. Time required for executing the interrupt sequence**

## 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 1.11.6 is set in the IPL.

**Table 1.11.6. Relationship between interrupts without interrupt priority levels and IPL**

| Interrupt sources without priority levels | Value set in the IPL |
|---|---|
| Watchdog timer, $\overline{\text{NMI}}$ | 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 1.11.6 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).

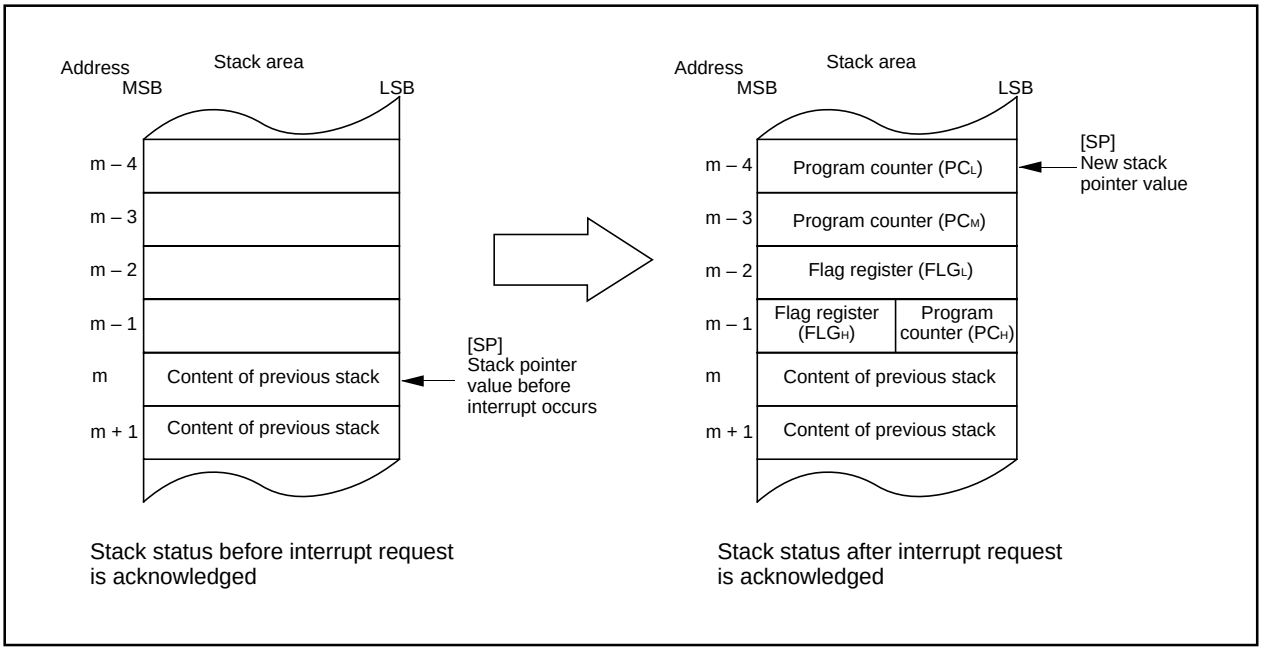
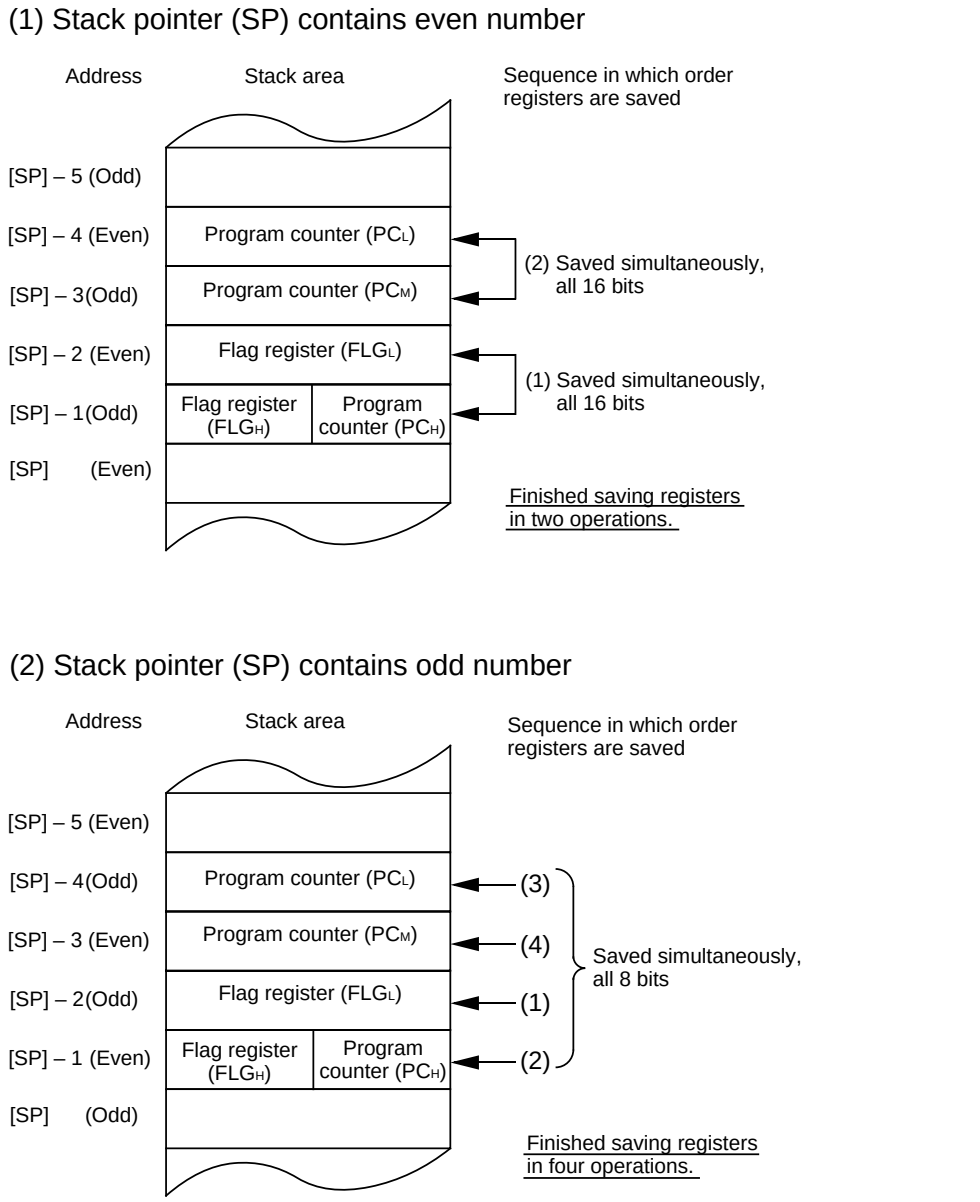



**Figure 1.11.6. State of stack before and after acceptance of interrupt request**

The operation of saving registers carried out in the interrupt sequence is dependent on whether the content of the stack pointer (Note) , at the time of acceptance of an interrupt request, is even or odd. If the content of the stack pointer (Note) is even, the content of the flag register (FLG) and the content of the program counter (PC) are saved, 16 bits at a time. If odd, their contents are saved in two steps, 8 bits at a time. Figure 1.11.7 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).




Note: [SP] denotes the initial value of the stack pointer (SP) when interrupt request is acknowledged.
After registers are saved, the SP content is [SP] minus 4.

**Figure 1.11.7. 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 1.11.8 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 1.11.8. 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 1.11.9 shows the circuit that judges the interrupt priority level.

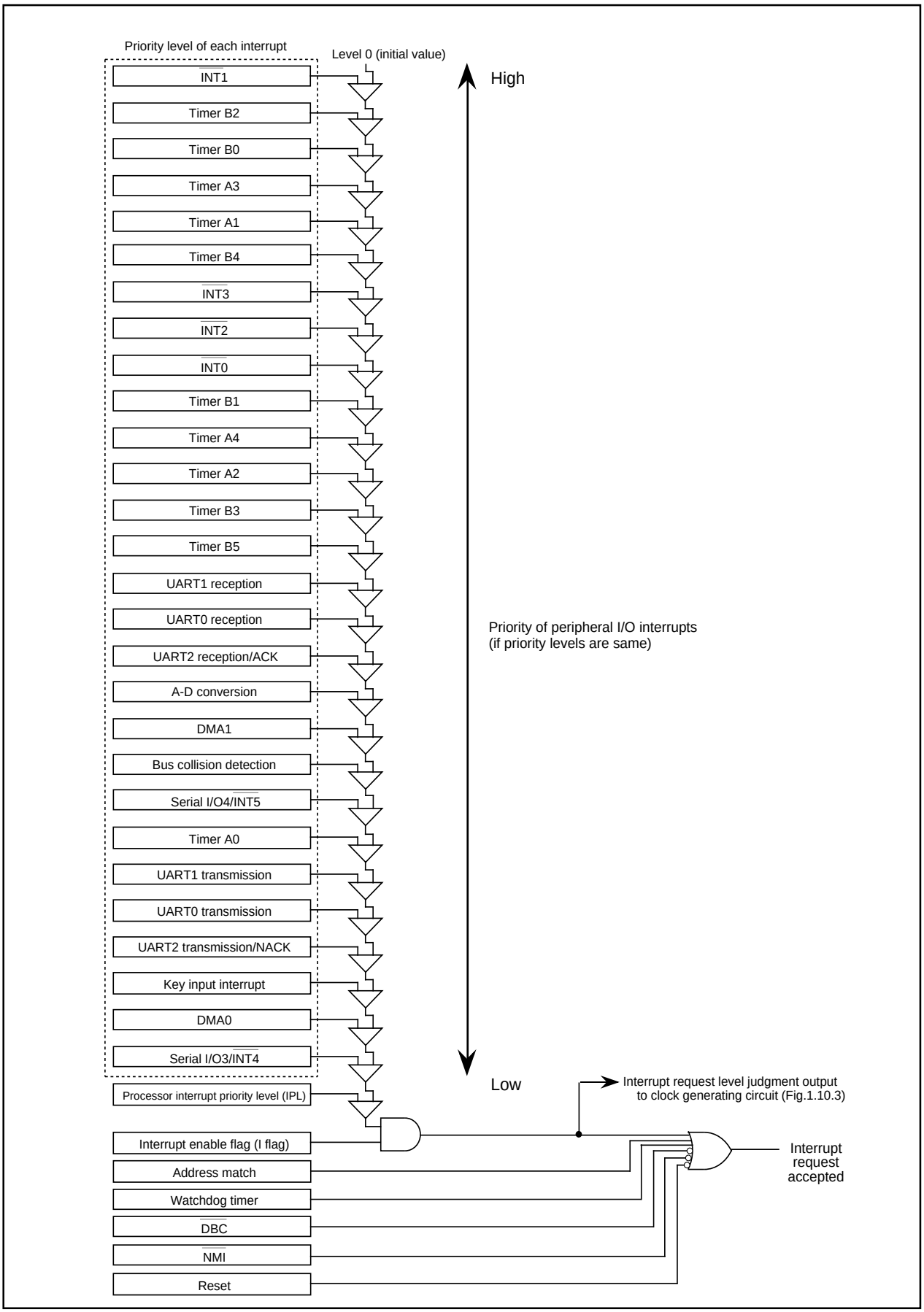



**Figure 1.11.9. Maskable interrupts priorities (peripheral I/O interrupts)**

## $\overline{\text{INT}}$ Interrupt

$\overline{\text{INT0}}$ to $\overline{\text{INT5}}$ are triggered by the edges of external inputs. The edge polarity is selected using the polarity select bit.

Of interrupt control registers, $0048_{16}$ is used both as serial I/O4 and external interrupt $\overline{\text{INT5}}$ input control register, and $0049_{16}$ is used both as serial I/O3 and as external interrupt $\overline{\text{INT4}}$ input control register. Use the interrupt request cause select bits - bits 6 and 7 of the interrupt request cause select register ($035F_{16}$) - to specify which interrupt request cause to select. After having set an interrupt request cause, be sure to clear the corresponding interrupt request bit before enabling an interrupt.

Either of the interrupt control registers - $0048_{16}$, $0049_{16}$ - has the polarity-switching bit. Be sure to set this bit to “0” to select an serial I/O as the interrupt request cause.

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 ($035F_{16}$). To select both edges, set the polarity switching bit of the corresponding interrupt control register to ‘falling edge’ (“0”).

Figure 1.11.10 shows the Interrupt request cause select register.

Interrupt request cause select register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| IFSR | $035F_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| IFSR0 | INT0 interrupt polarity switching bit | 0 : One edge<br>1 : Two edges | ○ | ○ |
| IFSR1 | INT1 interrupt polarity switching bit | 0 : One edge<br>1 : Two edges | ○ | ○ |
| IFSR2 | INT2 interrupt polarity switching bit | 0 : One edge<br>1 : Two edges | ○ | ○ |
| IFSR3 | INT3 interrupt polarity switching bit | 0 : One edge<br>1 : Two edges | ○ | ○ |
| IFSR4 | INT4 interrupt polarity switching bit | 0 : One edge<br>1 : Two edges | ○ | ○ |
| IFSR5 | INT5 interrupt polarity switching bit | 0 : One edge<br>1 : Two edges | ○ | ○ |
| IFSR6 | Interrupt request cause select bit | 0 : SIO3<br>1 : INT4 | ○ | ○ |
| IFSR7 | Interrupt request cause select bit | 0 : SIO4<br>1 : INT5 | ○ | ○ |

**Figure 1.11.10. Interrupt request cause select register**

## NMI Interrupt

An NMI interrupt is generated when the input to the P85/NMI pin changes from “H” to “L”. The 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 cancelling 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 1.11.11 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.

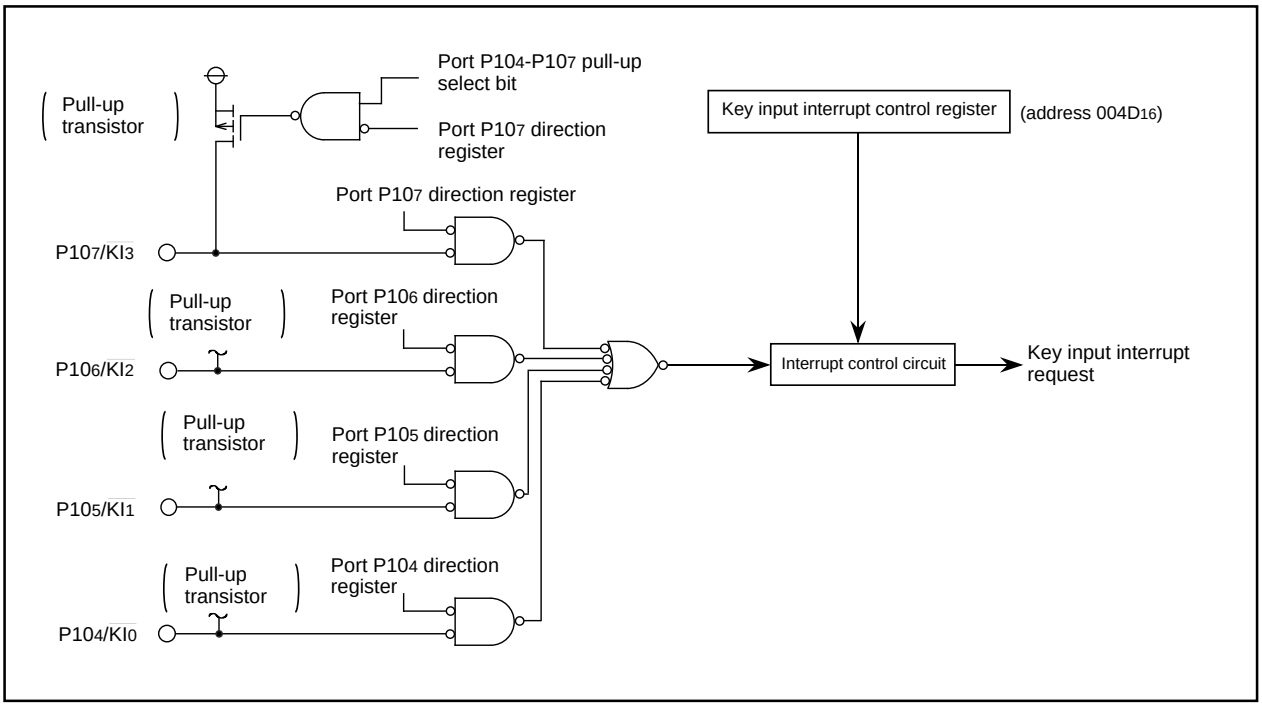


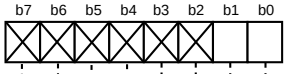

**Figure 1.11.11. Block diagram of key input interrupt**

## 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. Note that when using the external data bus in width of 8 bits, the address match interrupt cannot be used for external area.

Figure 1.11.12 shows the address match interrupt-related registers.

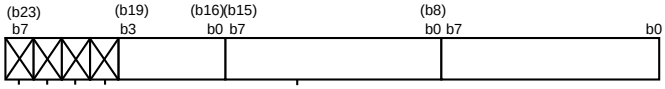
Address match interrupt enable register




| Symbol | Address | When reset |
|---|---|---|
| AIER | $0009_{16}$ | $XXXXXX00_{2}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| AIER0 | Address match interrupt 0 enable bit | 0 : Interrupt disabled<br>1 : Interrupt enabled | ○ | ○ |
| AIER1 | Address match interrupt 1 enable bit | 0 : Interrupt disabled<br>1 : Interrupt enabled | ○ | ○ |
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be indeterminated. | | | — | — |

Address match interrupt register i (i = 0, 1)




| Symbol | Address | When reset |
|---|---|---|
| RMAD0 | $0012_{16}$ to $0010_{16}$ | $X00000_{16}$ |
| RMAD1 | $0016_{16}$ to $0014_{16}$ | $X00000_{16}$ |

| Function | Values that can be set | R | W |
|---|---|---|---|
| Address setting register for address match interrupt | $00000_{16}$ to $FFFFF_{16}$ | ○ | ○ |
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be indeterminated. | | — | — |

**Figure 1.11.12. Address match interrupt-related registers**

## Precautions for Interrupts

### (1) Reading address $00000_{16}$

- 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 $00000_{16}$ will then be set to “0”.
  Even if the address $00000_{16}$ 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 $00000_{16}$ by software.

### (2) Setting the stack pointer

- The value of the stack pointer immediately after reset is initialized to $0000_{16}$. 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 $\overline{NMI}$ interrupt, initialize the stack pointer at the beginning of a program. Concerning the first instruction immediately after reset, generating any interrupts including the $\overline{NMI}$ interrupt is prohibited.

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

- The $\overline{NMI}$ interrupt can not be disabled. Be sure to connect $\overline{NMI}$ pin to Vcc via a pull-up resistor if unused. Be sure to work on it.
- The $\overline{NMI}$ pin also serves as $P8_5$, 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 $\overline{NMI}$ interrupt is input.
- Do not attempt to go into stop mode with the input to the $\overline{NMI}$ pin being in the “L” state. With the input to the $\overline{NMI}$ 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 $\overline{NMI}$ pin being in the “L” state. With the input to the $\overline{NMI}$ 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 $\overline{NMI}$ 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 380 ns width is necessary for the signal input to pins $\overline{INT_0}$ through $\overline{INT_5}$ regardless of the CPU operation clock.
- When the polarity of the $\overline{INT_0}$ to $\overline{INT_5}$ pins is changed or the interrupt request cause of the software interrupt numbers 8 to 9 is changed, the interrupt request bit is sometimes set to “1”. After these changes were made, set the interrupt request bit to “0”. Figure 1.11.13 shows the procedure for changing the $\overline{INT}$ interrupt generate factor.

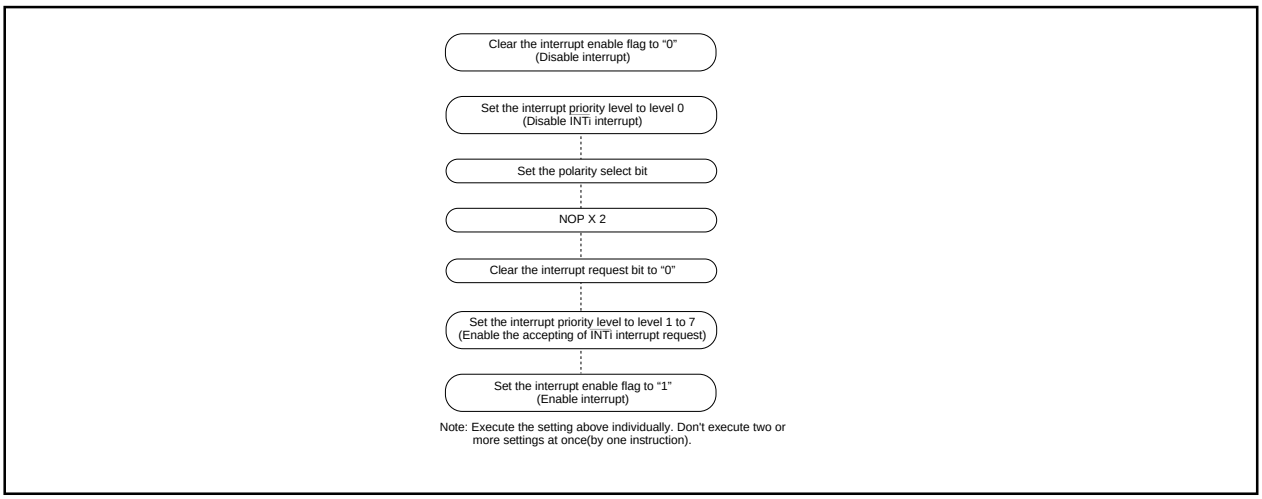



**Figure 1.11.13. Switching condition of $\overline{\text{INT}}$ interrupt request**

## (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 changing an interrupt control register in a sate of interrupts being disabled, please read the following precautions on instructions used before changing the register.

(1) Changing a non-interrupt request bit

If an interrupt request for an interrupt control register is generated during an instruction to rewrite the register is being executed, there is a case that the interrupt request bit is not set and consequently the interrupt is ignored. This will depend on the instruction. If this creates problems, use the below instructions to change the register.

Instructions : AND, OR, BCLR, BSET

(2) Changing the interrupt request bit

When attempting to clear the interrupt request bit of an interrupt control register, the interrupt request bit is not cleared sometimes. This will depend on the instruction. If this creates problems, use the below instructions to change the register.

Instructions : MOV

## 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 $X_{IN}$ is selected for the BCLK, bit 7 of the watchdog timer control register (address $000F_{16}$) selects the prescaler division ratio (by 16 or by 128). When $X_{CIN}$ is selected as the BCLK, the prescaler is set for division by 2 regardless of bit 7 of the watchdog timer control register (address $000F_{16}$). 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 $X_{IN}$ chosen for BCLK**

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

**With $X_{CIN}$ chosen for BCLK**

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

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 $000E_{16}$) and when a watchdog timer interrupt request is generated. The prescaler is initialized only when the microcomputer is reset. After a reset is cancelled, the watchdog timer and prescaler are both stopped. The count is started by writing to the watchdog timer start register (address $000E_{16}$). In stop mode, wait mode and hold state, the watchdog timer and prescaler are stopped. Counting is resumed from the held value when the modes or state are released.
Figure 1.12.1 shows the block diagram of the watchdog timer. Figure 1.12.2 shows the watchdog timer-related registers.

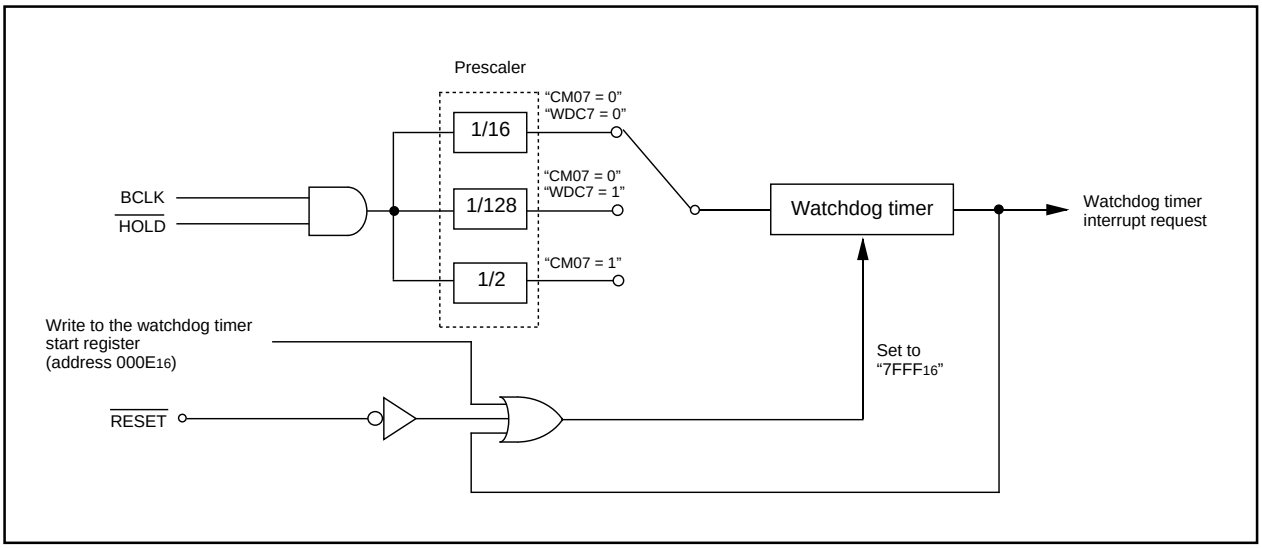



**Figure 1.12.1. Block diagram of watchdog timer**

Watchdog timer control register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | 0 | 0 | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| WDC | $000F_{16}$ | $000XXXXX_{2}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| High-order bit of watchdog timer | | | ○ | × |
| Reserved bit | | Must always be set to “0” | ○ | ○ |
| Reserved bit | | Must always be set to “0” | ○ | ○ |
| WDC7 | Prescaler select bit | 0 : Divided by 16<br>1 : Divided by 128 | ○ | ○ |

Watchdog timer start register

| b7 | b0 |
|---|---|
| | |

| Symbol | Address | When reset |
|---|---|---|
| WDTS | $000E_{16}$ | Indeterminate |

| Function | R | W |
|---|---|---|
| The watchdog timer is initialized and starts counting after a write instruction to this register. The watchdog timer value is always initialized to “$7FFF_{16}$” regardless of whatever value is written. | × | ○ |

**Figure 1.12.2. 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 1.13.1 shows the block diagram of the DMAC. Table 1.13.1 shows the DMAC specifications. Figures 1.13.2 to 1.13.4 show the registers used by the DMAC.

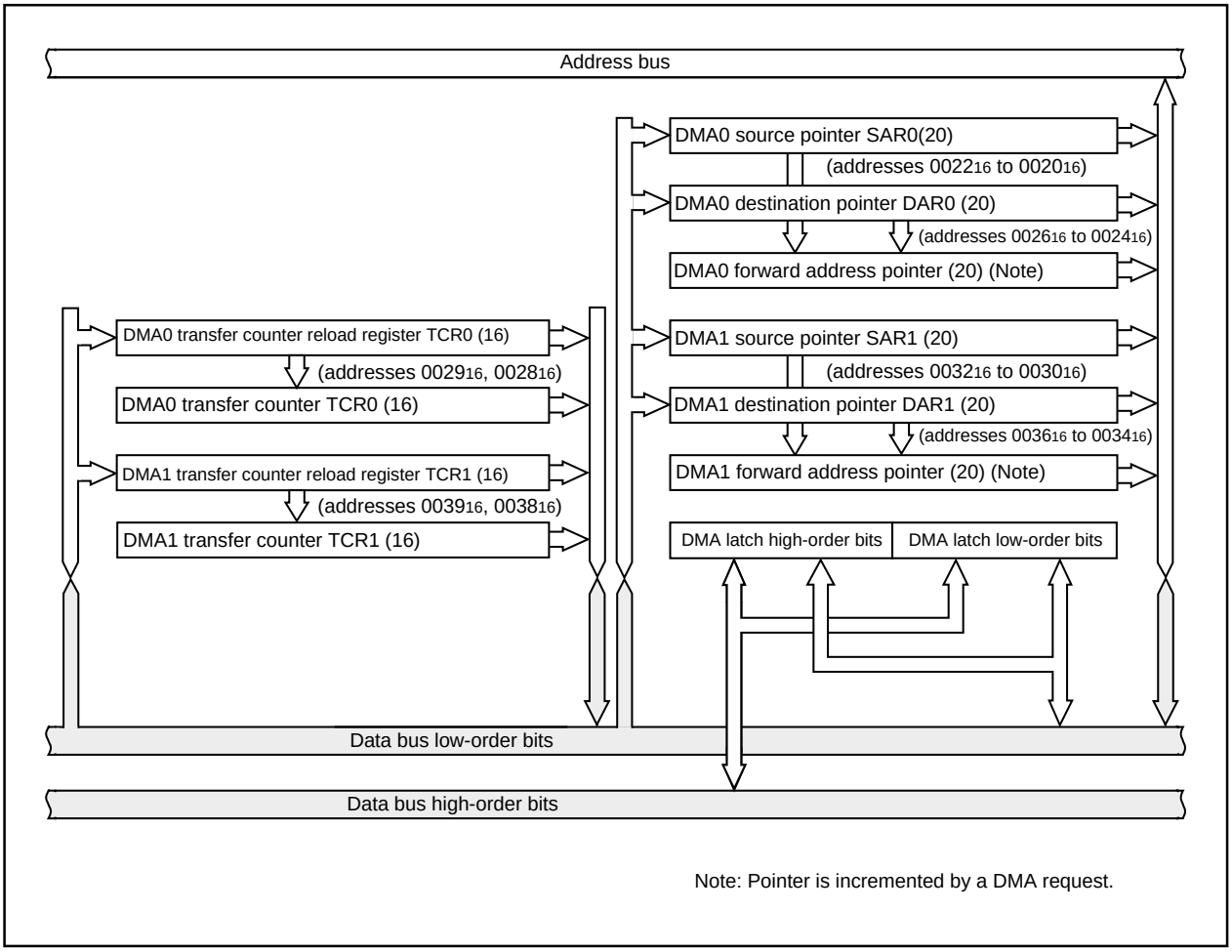



**Figure 1.13.1. Block diagram of DMAC**

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 1.13.1. DMAC specifications**

| Item | Specification |
|---|---|
| No. of channels | 2 (cycle steal method) |
| Transfer memory space | • From any address in the 1M bytes space to a fixed address<br>• From a fixed address to any address in the 1M bytes space<br>• From a fixed address to a fixed address<br>(Note that DMA-related registers [0020$_{16}$ to 003F$_{16}$] 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 $\overline{\text{INT0}}$ or $\overline{\text{INT1}}$ or both edge<br>Timer A0 to timer A4 interrupt requests<br>Timer B0 to timer B5 interrupt requests<br>UART0 transfer and reception interrupt requests<br>UART1 transfer and reception interrupt requests<br>UART2 transfer and reception interrupt requests<br>Serial I/O3, 4 interrpt requests<br>A-D conversion interrupt requests<br>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 | • Single transfer mode<br>After the transfer counter underflows, the DMA enable bit turns to "0", and the DMAC turns inactive<br>• Repeat transfer mode<br>After the transfer counter underflows, the value of the transfer counter reload register is reloaded to the transfer counter.<br>The DMAC remains active unless a "0" is written to the DMA enable bit. |
| 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.<br>When the DMAC is active, data transfer starts every time a DMA transfer request signal occurs. |
| Inactive | • When the DMA enable bit is set to "0", the DMAC is inactive.<br>• After the transfer counter underflows in single transfer mode |
| 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.<br>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.<br>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.

DMA0 request cause select register

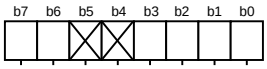
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | X | X | | | | |

| Symbol | Address | When reset |
|---|---|---|
| DM0SL | $03B8_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| DSEL0 | DMA request cause select bit | b3 b2 b1 b0<br>0 0 0 0 : Falling edge of $\overline{\text{INT0}}$ pin<br>0 0 0 1 : Software trigger<br>0 0 1 0 : Timer A0<br>0 0 1 1 : Timer A1<br>0 1 0 0 : Timer A2<br>0 1 0 1 : Timer A3<br>0 1 1 0 : Timer A4 (DMS=0)<br>/two edges of INT0 pin (DMS=1)<br>0 1 1 1 : Timer B0 (DMS=0)<br>Timer B3 (DMS=1)<br>1 0 0 0 : Timer B1 (DMS=0)<br>Timer B4 (DMS=1)<br>1 0 0 1 : Timer B2 (DMS=0)<br>Timer B5 (DMS=1)<br>1 0 1 0 : UART0 transmit<br>1 0 1 1 : UART0 receive<br>1 1 0 0 : UART2 transmit<br>1 1 0 1 : UART2 receive<br>1 1 1 0 : A-D conversion<br>1 1 1 1 : UART1 transmit | ○ | ○ |
| DSEL1 | | | ○ | ○ |
| DSEL2 | | | ○ | ○ |
| DSEL3 | | | ○ | ○ |
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be “0”. | | | – | – |
| DMS | DMA request cause expansion select bit | 0 : Normal<br>1 : Expanded cause | ○ | ○ |
| DSR | Software DMA request bit | If software trigger is selected, a DMA request is generated by setting this bit to “1” (When read, the value of this bit is always “0”) | ○ | ○ |

**Figure 1.13.2. DMAC register (1)**

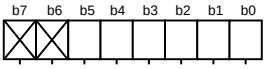
DMA1 request cause select register




| Symbol | Address | When reset |
|---|---|---|
| DM1SL | $03BA_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| DSEL0 | DMA request cause select bit | b3 b2 b1 b0<br>0 0 0 0 : Falling edge of $\overline{\text{INT1}}$ pin<br>0 0 0 1 : Software trigger<br>0 0 1 0 : Timer A0<br>0 0 1 1 : Timer A1<br>0 1 0 0 : Timer A2<br>0 1 0 1 : Timer A3(DMS=0)<br>/serial I/O3 (DMS=1)<br>0 1 1 0 : Timer A4 (DMS=0)<br>/serial I/O4 (DMS=1)<br>0 1 1 1 : Timer B0 (DMS=0)<br>/two edges of INT1 (DMS=1)<br>1 0 0 0 : Timer B1<br>1 0 0 1 : Timer B2<br>1 0 1 0 : UART0 transmit<br>1 0 1 1 : UART0 receive<br>1 1 0 0 : UART2 transmit<br>1 1 0 1 : UART2 receive<br>1 1 1 0 : A-D conversion<br>1 1 1 1 : UART1 receive | ○ | ○ |
| DSEL1 | | | ○ | ○ |
| DSEL2 | | | ○ | ○ |
| DSEL3 | | | ○ | ○ |
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be “0”. | | | – | – |
| DMS | DMA request cause expansion select bit | 0 : Normal<br>1 : Expanded cause | ○ | ○ |
| DSR | Software DMA request bit | If software trigger is selected, a DMA request is generated by setting this bit to “1” (When read, the value of this bit is always “0”) | ○ | ○ |

DMAi control register



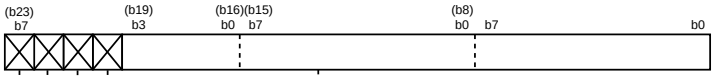

| Symbol | Address | When reset |
|---|---|---|
| DMiCON(i=0,1) | $002C_{16}$, $003C_{16}$ | $00000X00_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| DMBIT | Transfer unit bit select bit | 0 : 16 bits<br>1 : 8 bits | ○ | ○ |
| DMASL | Repeat transfer mode select bit | 0 : Single transfer<br>1 : Repeat transfer | ○ | ○ |
| DMAS | DMA request bit (Note 1) | 0 : DMA not requested<br>1 : DMA requested | ○ | ○<br>(Note 2) |
| DMAE | DMA enable bit | 0 : Disabled<br>1 : Enabled | ○ | ○ |
| DSD | Source address direction select bit (Note 3) | 0 : Fixed<br>1 : Forward | ○ | ○ |
| DAD | Destination address direction select bit (Note 3) | 0 : Fixed<br>1 : Forward | ○ | ○ |
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be “0”. | | | – | – |

Note 1: DMA request can be cleared by resetting the bit.
Note 2: This bit can only be set to “0”.
Note 3: Source address direction select bit and destination address direction select bit cannot be set to “1” simultaneously.

**Figure 1.13.3. DMAC register (2)**

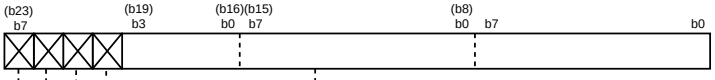
DMAi source pointer (i = 0, 1)




| Symbol | Address | When reset |
|---|---|---|
| SAR0 | $0022_{16}$ to $0020_{16}$ | Indeterminate |
| SAR1 | $0032_{16}$ to $0030_{16}$ | Indeterminate |

| Function | Transfer address specification | R | W |
|---|---|---|---|
| • Source pointer<br>Stores the source address | $00000_{16}$ to $FFFFF_{16}$ | ○ | ○ |
| Nothing is assigned.<br>In an attempt to write to these bits, write "0". The value, if read, turns out to be "0". | | – | – |

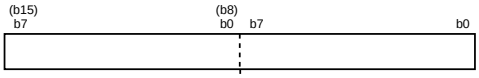
DMAi destination pointer (i = 0, 1)




| Symbol | Address | When reset |
|---|---|---|
| DAR0 | $0026_{16}$ to $0024_{16}$ | Indeterminate |
| DAR1 | $0036_{16}$ to $0034_{16}$ | Indeterminate |

| Function | Transfer address specification | R | W |
|---|---|---|---|
| • Destination pointer<br>Stores the destination address | $00000_{16}$ to $FFFFF_{16}$ | ○ | ○ |
| Nothing is assigned.<br>In an attempt to write to these bits, write "0". The value, if read, turns out to be "0". | | – | – |

DMAi transfer counter (i = 0, 1)




| Symbol | Address | When reset |
|---|---|---|
| TCR0 | $0029_{16}$, $0028_{16}$ | Indeterminate |
| TCR1 | $0039_{16}$, $0038_{16}$ | Indeterminate |

| Function | Transfer count specification | R | W |
|---|---|---|---|
| • Transfer counter<br>Set a value one less than the transfer count | $0000_{16}$ to $FFFF_{16}$ | ○ | ○ |

**Figure 1.13.4. 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. In memory expansion mode and microprocessor mode, the number of read and write bus cycles also depends on the level of the BYTE pin. 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 BYTE pin level**

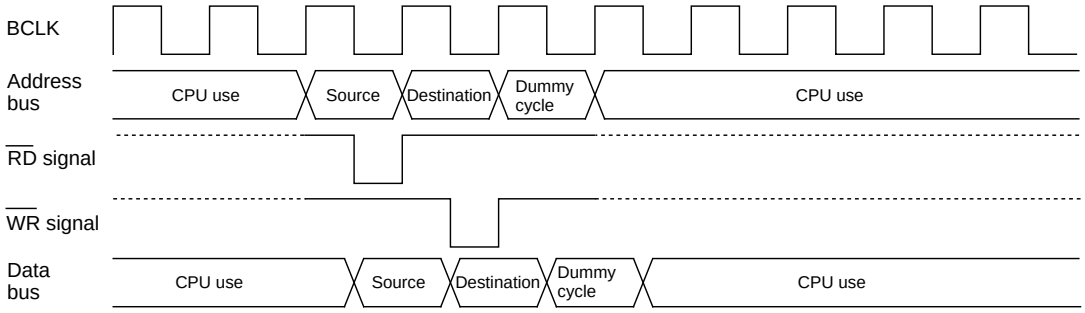
When transferring 16-bit data over an 8-bit data bus (BYTE pin = “H”) in memory expansion mode and microprocessor mode, the 16 bits of data are sent in two 8-bit blocks. Therefore, two bus cycles are required for reading the data and two are required for writing the data. Also, in contrast to when the CPU accesses internal memory, when the DMAC accesses internal memory (internal ROM, internal RAM, and SFR), these areas are accessed using the data size selected by the BYTE pin.

**(c) Effect of software wait**

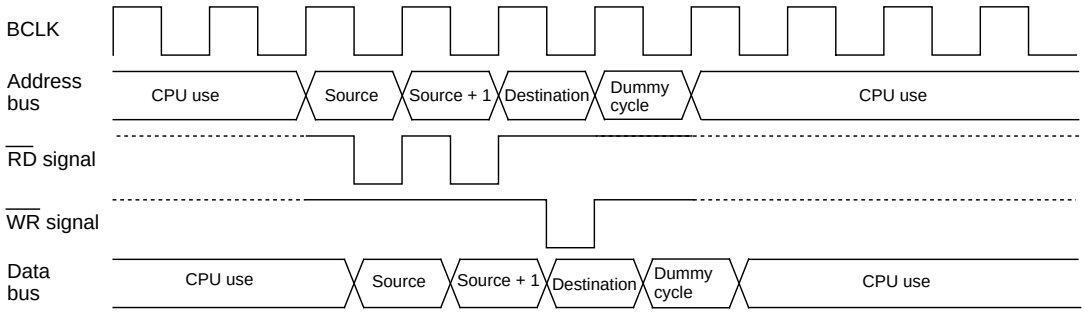
When the SFR area or a memory area with a software wait is accessed, the number of cycles is increased for the wait by 1 bus cycle. The length of the cycle is determined by BCLK.

Figure 1.13.5 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 1.13.5, if data is being transferred in 16-bit units on an 8-bit bus, two bus cycles are required for both the source read cycle and the destination write cycle.

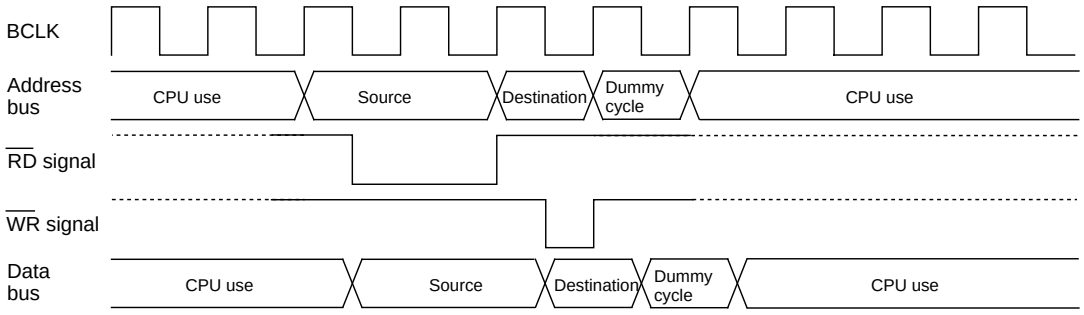
**(1) 8-bit transfers**
**16-bit transfers and the source address is even.**



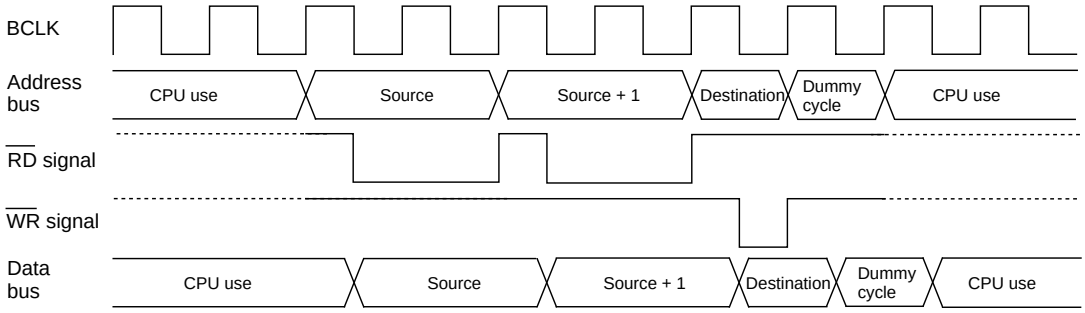

**(2) 16-bit transfers and the source address is odd**
**Transferring 16-bit data on an 8-bit data bus (In this case, there are also two destination cycles).**




**(3) One wait is inserted into the source read under the conditions in (1)**




**(4) One wait is inserted into the source read under the conditions in (2)**
**(When 16-bit data is transferred on an 8-bit data bus, there are two destination cycles).**




Note: The same timing changes occur with the respective conditions at the destination as at the source.

**Figure 1.13.5. Example of the transfer cycles for a source read**

### (2) DMAC transfer cycles

Any combination of even or odd transfer read and write addresses is possible. Table 1.13.2 shows the number of DMAC transfer cycles.

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

No. of transfer cycles per transfer unit = No. of read cycles x j + No. of write cycles x k

**Table 1.13.2. No. of DMAC transfer cycles**

| Transfer unit | Bus width | Access address | Single-chip mode | | Memory expansion mode Microprocessor mode | |
|---|---|---|---|---|---|---|
| | | | No. of read cycles | No. of write cycles | No. of read cycles | No. of write cycles |
| 8-bit transfers (DMBIT= “1”) | 16-bit (BYTE= “L”) | Even | 1 | 1 | 1 | 1 |
| | | Odd | 1 | 1 | 1 | 1 |
| | 8-bit (BYTE = “H”) | Even | — | — | 1 | 1 |
| | | Odd | — | — | 1 | 1 |
| 16-bit transfers (DMBIT= “0”) | 16-bit (BYTE = “L”) | Even | 1 | 1 | 1 | 1 |
| | | Odd | 2 | 2 | 2 | 2 |
| | 8-bit (BYTE = “H”) | Even | — | — | 2 | 2 |
| | | Odd | — | — | 2 | 2 |

**Coefficient j, k**

| Internal memory | | | External memory | | |
|---|---|---|---|---|---|
| Internal ROM/RAM No wait | Internal ROM/RAM With wait | SFR area | Separate bus No wait | Separate bus With wait | Multiplex bus |
| 1 | 2 | 2 | 1 | 2 | 3 |

### 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 DMAi 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.

The DMA request bit turns to “1” if a DMA transfer request signal occurs, and turns to “0” immediately before data transfer starts. 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 $\overline{\text{INTi}}$ pin (i depends on which DMAC channel is used).

Selecting the $\overline{\text{INTi}}$ 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 $\overline{\text{INTi}}$ 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 1.13.6 An example of DMA transfer effected by external factors.

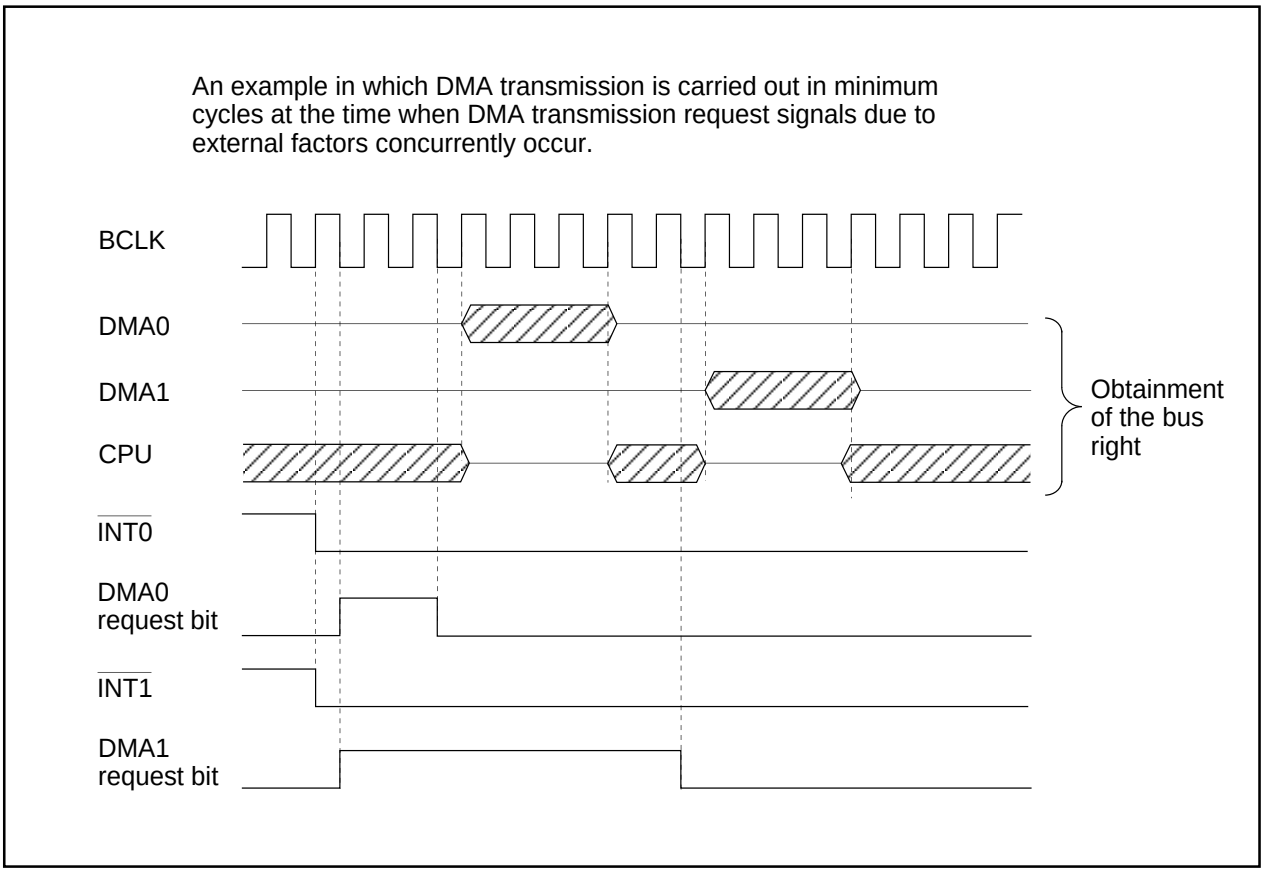



**Figure 1.13.6. An example of DMA transfer effected by external factors**

## 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 1.14.1 and 1.14.2 show the block diagram of timers.

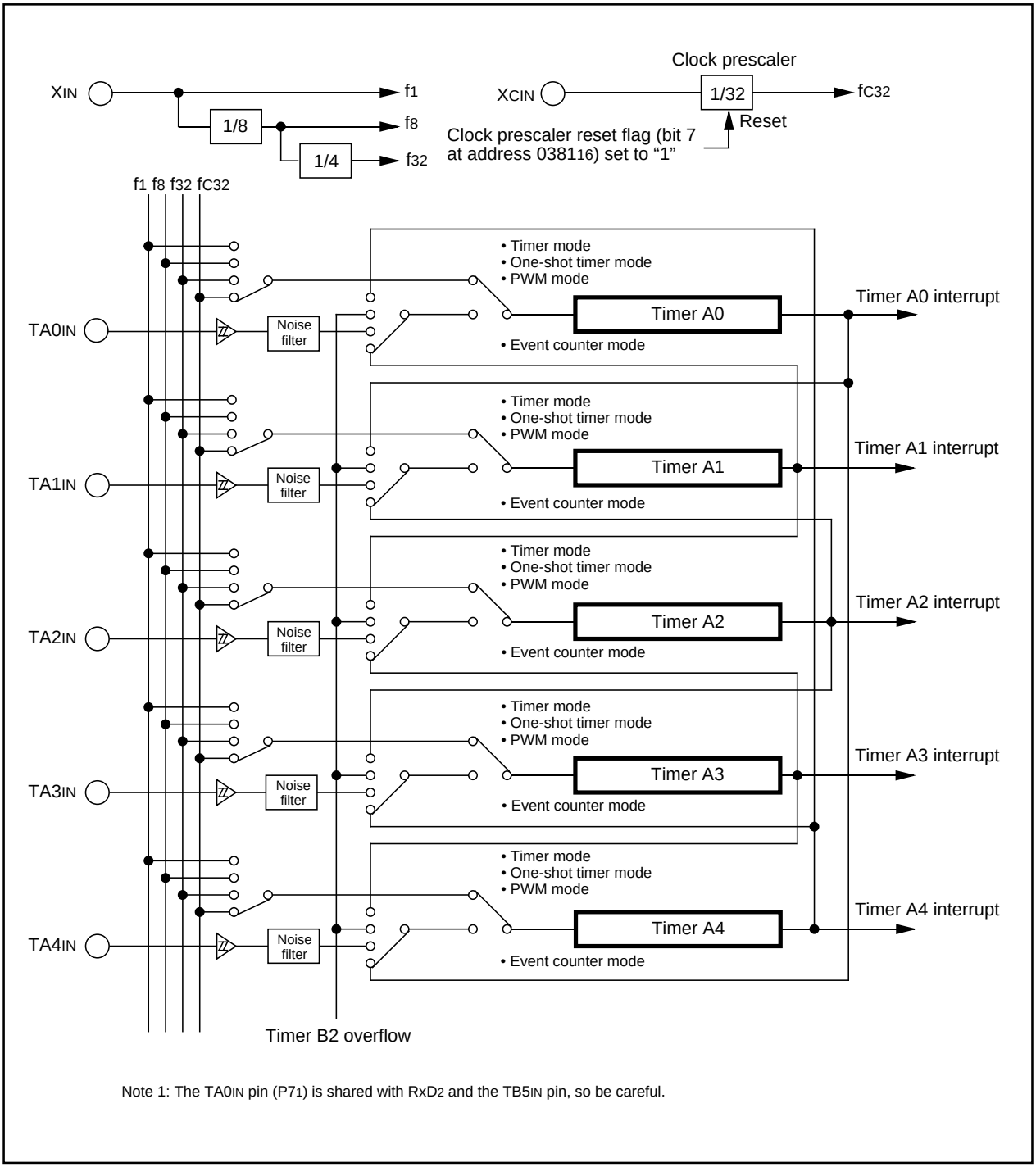



**Figure 1.14.1. Timer A block diagram**

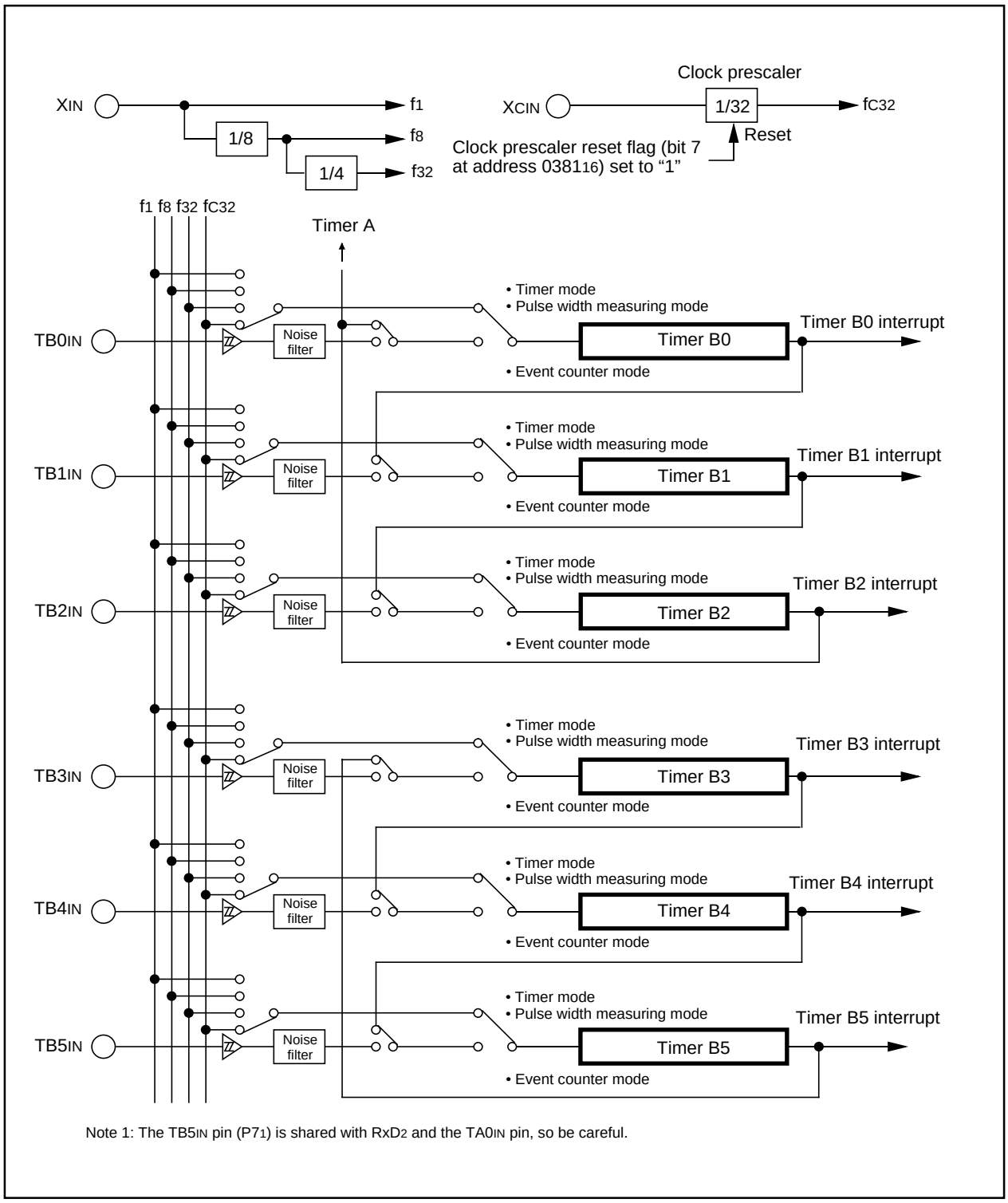



**Figure 1.14.2. Timer B block diagram**

## Timer A

Figure 1.14.3 shows the block diagram of timer A. Figures 1.14.4 to 1.14.6 show the timer A-related registers.

Except in event counter mode, timers A0 through A4 all have the same function. 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 over flow.
- One-shot timer mode: The timer stops counting when the count reaches “0000$_{16}$”.
- Pulse width modulation (PWM) mode: The timer outputs pulses of a given width.

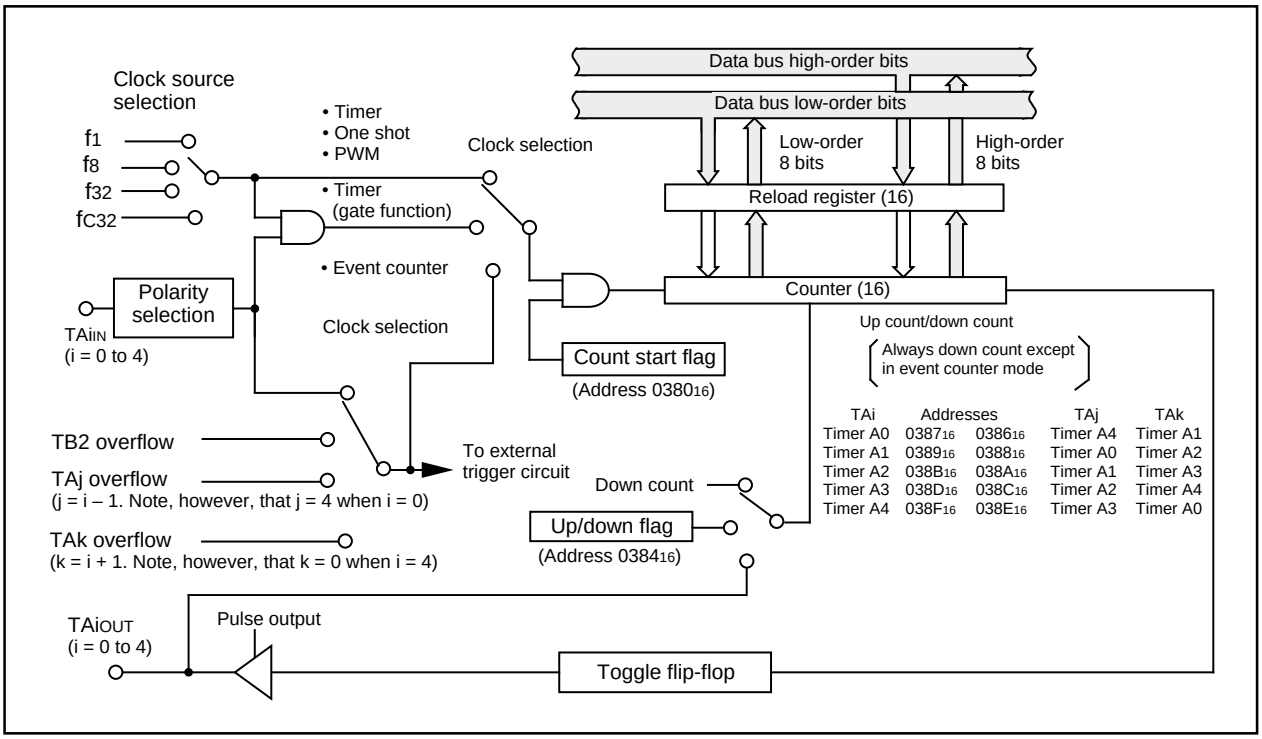



**Figure 1.14.3. Block diagram of timer A**

Timer Ai mode register

| Symbol | Address | When reset |
|---|---|---|
| TAiMR(i=0 to 4) | 0396$_{16}$ to 039A$_{16}$ | 00$_{16}$ |

b7 b6 b5 b4 b3 b2 b1 b0

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| TMOD0 | Operation mode select bit | b1 b0<br>0 0 : Timer mode<br>0 1 : Event counter mode<br>1 0 : One-shot timer mode<br>1 1 : Pulse width modulation (PWM) mode | ○ | ○ |
| TMOD1 | | | ○ | ○ |
| MR0 | Function varies with each operation mode | | ○ | ○ |
| MR1 | | | ○ | ○ |
| MR2 | | | ○ | ○ |
| MR3 | | | ○ | ○ |
| TCK0 | Count source select bit (Function varies with each operation mode) | | ○ | ○ |
| TCK1 | | | ○ | ○ |

**Figure 1.14.4. Timer A-related registers (1)**

Timer Ai register (Note 1)

(b15) b7 — (b8) b0 b7 — b0

| Symbol | Address | When reset |
|---|---|---|
| TA0 | $0387_{16}$,$0386_{16}$ | Indeterminate |
| TA1 | $0389_{16}$,$0388_{16}$ | Indeterminate |
| TA2 | $038B_{16}$,$038A_{16}$ | Indeterminate |
| TA3 | $038D_{16}$,$038C_{16}$ | Indeterminate |
| TA4 | $038F_{16}$,$038E_{16}$ | Indeterminate |

| Function | Values that can be set | R | W |
|---|---|---|---|
| • Timer mode<br>Counts an internal count source | $0000_{16}$ to $FFFF_{16}$ | O | O |
| • Event counter mode<br>Counts pulses from an external source or timer overflow | $0000_{16}$ to $FFFF_{16}$ | O | O |
| • One-shot timer mode<br>Counts a one shot width | $0000_{16}$ to $FFFF_{16}$<br>(Note 2,4) | X | O |
| • Pulse width modulation mode (16-bit PWM)<br>Functions as a 16-bit pulse width modulator | $0000_{16}$ to $FFFE_{16}$<br>(Note 3,4) | X | O |
| • Pulse width modulation mode (8-bit PWM)<br>Timer low-order address functions as an 8-bit prescaler and high-order address functions as an 8-bit pulse width modulator | $00_{16}$ to $FE_{16}$<br>(High-order address)<br>$00_{16}$ to $FF_{16}$<br>(Low-order address)<br>(Note 3,4) | X | O |

Note 1: Read and write data in 16-bit units.
Note 2: When the timer Ai register is set to "$0000_{16}$", the counter does not operate and the timer Ai interrupt request is not generated. When the pulse is set to output, the pulse does not output from the TAiOUT pin.
Note 3: When the timer Ai register is set to "$0000_{16}$", the pulse width modulator does not operate and the output level of the TAiOUT pin remains "L" level, therefore the timer Ai interrupt request is not generated. This also occurs in the 8-bit pulse width modulator mode when the significant 8 high-order bits in the timer Ai register are set to "$00_{16}$".
Note 4: Use MOV instruction to write to this register.

Count start flag

b7 b6 b5 b4 b3 b2 b1 b0

| Symbol | Address | When reset |
|---|---|---|
| TABSR | $0380_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| TA0S | Timer A0 count start flag | 0 : Stops counting<br>1 : Starts counting | O | O |
| TA1S | Timer A1 count start flag | | O | O |
| TA2S | Timer A2 count start flag | | O | O |
| TA3S | Timer A3 count start flag | | O | O |
| TA4S | Timer A4 count start flag | | O | O |
| TB0S | Timer B0 count start flag | | O | O |
| TB1S | Timer B1 count start flag | | O | O |
| TB2S | Timer B2 count start flag | | O | O |

Up/down flag (Note 1)

b7 b6 b5 b4 b3 b2 b1 b0

| Symbol | Address | When reset |
|---|---|---|
| UDF | $0384_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| TA0UD | Timer A0 up/down flag | 0 : Down count<br>1 : Up count<br><br>This specification becomes valid when the up/down flag content is selected for up/down switching cause | O | O |
| TA1UD | Timer A1 up/down flag | | O | O |
| TA2UD | Timer A2 up/down flag | | O | O |
| TA3UD | Timer A3 up/down flag | | O | O |
| TA4UD | Timer A4 up/down flag | | O | O |
| TA2P | Timer A2 two-phase pulse signal processing select bit | 0 : two-phase pulse signal processing disabled<br>1 : two-phase pulse signal processing enabled (Note 2)<br><br>When not using the two-phase pulse signal processing function, set the select bit to "0" | X | O |
| TA3P | Timer A3 two-phase pulse signal processing select bit | | X | O |
| TA4P | Timer A4 two-phase pulse signal processing select bit | | X | O |

Note 1: Use MOV instruction to write to this register.
Note 2: Set the TAiIN and TAiOUT pins correspondent port direction registers to "0".

**Figure 1.14.5. Timer A-related registers (2)**

### One-shot start flag

| Symbol | Address | When reset |
|---|---|---|
| ONSF | $0382_{16}$ | $00X00000_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| TA0OS | Timer A0 one-shot start flag | 1 : Timer start<br>When read, the value is "0" | ○ | ○ |
| TA1OS | Timer A1 one-shot start flag | | ○ | ○ |
| TA2OS | Timer A2 one-shot start flag | | ○ | ○ |
| TA3OS | Timer A3 one-shot start flag | | ○ | ○ |
| TA4OS | Timer A4 one-shot start flag | | ○ | ○ |
| Nothing is assigned.<br>In an attempt to write to this bit, write "0". The value, if read, turns out to be indeterminate. | | | — | — |
| TA0TGL | Timer A0 event/trigger select bit | b7 b6<br>0 0 : Input on TA0IN is selected (Note)<br>0 1 : TB2 overflow is selected<br>1 0 : TA4 overflow is selected<br>1 1 : TA1 overflow is selected | ○ | ○ |
| TA0TGH | | | ○ | ○ |

Note: Set the corresponding port direction register to "0".

### Trigger select register

| Symbol | Address | When reset |
|---|---|---|
| TRGSR | $0383_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| TA1TGL | Timer A1 event/trigger select bit | b1 b0<br>0 0 : Input on TA1IN is selected (Note)<br>0 1 : TB2 overflow is selected<br>1 0 : TA0 overflow is selected<br>1 1 : TA2 overflow is selected | ○ | ○ |
| TA1TGH | | | ○ | ○ |
| TA2TGL | Timer A2 event/trigger select bit | b3 b2<br>0 0 : Input on TA2IN is selected (Note)<br>0 1 : TB2 overflow is selected<br>1 0 : TA1 overflow is selected<br>1 1 : TA3 overflow is selected | ○ | ○ |
| TA2TGH | | | ○ | ○ |
| TA3TGL | Timer A3 event/trigger select bit | b5 b4<br>0 0 : Input on TA3IN is selected (Note)<br>0 1 : TB2 overflow is selected<br>1 0 : TA2 overflow is selected<br>1 1 : TA4 overflow is selected | ○ | ○ |
| TA3TGH | | | ○ | ○ |
| TA4TGL | Timer A4 event/trigger select bit | b7 b6<br>0 0 : Input on TA4IN is selected (Note)<br>0 1 : TB2 overflow is selected<br>1 0 : TA3 overflow is selected<br>1 1 : TA0 overflow is selected | ○ | ○ |
| TA4TGH | | | ○ | ○ |

Note: Set the corresponding port direction register to "0".

### Clock prescaler reset flag

| Symbol | Address | When reset |
|---|---|---|
| CPSRF | $0381_{16}$ | $0XXXXXXX_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| Nothing is assigned.<br>In an attempt to write to these bits, write "0". The value, if read, turns out to be indeterminate. | | | — | — |
| CPSR | Clock prescaler reset flag | 0 : No effect<br>1 : Prescaler is reset<br>(When read, the value is "0") | ○ | ○ |

**Figure 1.14.6. Timer A-related registers (3)**

### (1) Timer mode

In this mode, the timer counts an internally generated count source. (See Table 1.14.1.) Figure 1.14.7 shows the timer Ai mode register in timer mode.

**Table 1.14.1. Specifications of timer mode**

| Item | Specification |
|---|---|
| Count source | $f_1$, $f_8$, $f_{32}$, $f_{C32}$ |
| Count operation | • Down count<br>• When the timer underflows, it reloads the reload register contents before continuing counting |
| 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 |
| $TAi_{IN}$ pin function | Programmable I/O port or gate input |
| $TAi_{OUT}$ 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 | • When counting stopped<br>When a value is written to timer Ai register, it is written to both reload register and counter<br>• When counting in progress<br>When a value is written to timer Ai register, it is written to only reload register (Transferred to counter at next reload time) |
| Select function | • Gate function<br>Counting can be started and stopped by the $TAi_{IN}$ pin's input signal<br>• Pulse output function<br>Each time the timer underflows, the $TAi_{OUT}$ pin's polarity is reversed |

Timer Ai mode register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | 0 | | | | 0 | 0 |

| Symbol | Address | When reset |
|---|---|---|
| TAiMR(i=0 to 4) | $039_{16}$ to $039A_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| TMOD0 | Operation mode select bit | b1 b0<br>0 0 : Timer mode | ○ | ○ |
| TMOD1 | | | ○ | ○ |
| MR0 | Pulse output function select bit | 0 : Pulse is not output (TAiOUT pin is a normal port pin)<br>1 : Pulse is output (Note 1) (TAiOUT pin is a pulse output pin) | ○ | ○ |
| MR1 | Gate function select bit | b4 b3<br>0 X (Note 2): Gate function not available (TAiIN pin is a normal port pin)<br>1 0 : Timer counts only when TAiIN pin is held "L" (Note 3)<br>1 1 : Timer counts only when TAiIN pin is held "H" (Note 3) | ○ | ○ |
| MR2 | | | ○ | ○ |
| MR3 | 0 (Must always be "0" in timer mode) | | ○ | ○ |
| TCK0 | Count source select bit | b7 b6<br>0 0 : $f_1$<br>0 1 : $f_8$<br>1 0 : $f_{32}$<br>1 1 : $f_{C32}$ | ○ | ○ |
| TCK1 | | | ○ | ○ |

Note 1: The settings of the corresponding port register and port direction register are invalid.
Note 2: The bit can be "0" or "1".
Note 3: Set the corresponding port direction register to "0".

**Figure 1.14.7. Timer Ai mode register in timer mode**

## (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 1.14.2 lists timer specifications when counting a single-phase external signal. Figure 1.14.8 shows the timer Ai mode register in event counter mode.

Table 1.14.3 lists timer specifications when counting a two-phase external signal. Figure 1.14.9 shows the timer Ai mode register in event counter mode.

**Table 1.14.2. Timer specifications in event counter mode (when not processing two-phase pulse signal)**

| Item | Specification |
|---|---|
| Count source | • External signals input to TAiIN pin (effective edge can be selected by software)<br>• TB2 overflow, TAj overflow |
| Count operation | • Up count or down count can be selected by external signal or software<br>• When the timer overflows or underflows, it reloads the reload register con tents before continuing counting (Note) |
| Divide ratio | 1/ (FFFF16 - n + 1) for up count<br>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 | • When counting stopped<br>When a value is written to timer Ai register, it is written to both reload register and counter<br>• When counting in progress<br>When a value is written to timer Ai register, it is written to only reload register (Transferred to counter at next reload time) |
| Select function | • Free-run count function<br>Even when the timer overflows or underflows, the reload register content is not reloaded to it<br>• Pulse output function<br>Each time the timer overflows or underflows, the TAiOUT pin's polarity is reversed |

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

Timer Ai mode register
(When not using two-phase pulse signal processing)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | 0 | | | | 0 | 1 |

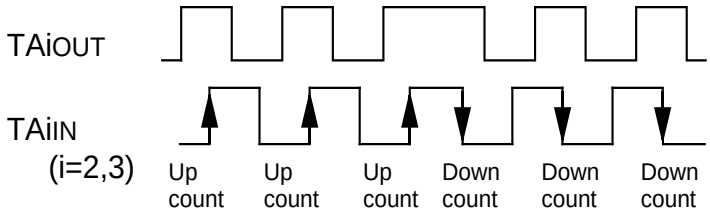
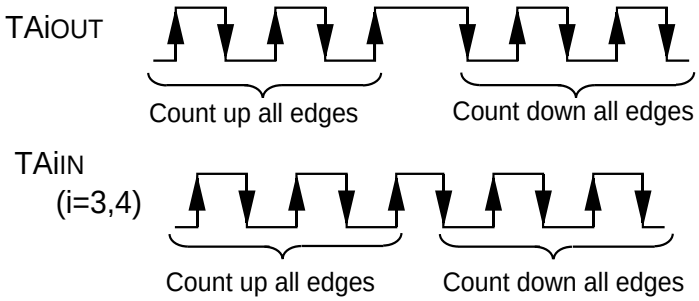
| Symbol | Address | When reset |
|---|---|---|
| TAiMR(i = 0 to 4) | 0396₁₆ to 039A₁₆ | 00₁₆ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| TMOD0 | Operation mode select bit | b1 b0<br>0 1 : Event counter mode (Note 1) | O | O |
| TMOD1 | | | O | O |
| MR0 | Pulse output function select bit | 0 : Pulse is not output (TAiOUT pin is a normal port pin)<br>1 : Pulse is output (Note 2) (TAiOUT pin is a pulse output pin) | O | O |
| MR1 | Count polarity select bit (Note 3) | 0 : Counts external signal's falling edge<br>1 : Counts external signal's rising edge | O | O |
| MR2 | Up/down switching cause select bit | 0 : Up/down flag's content<br>1 : TAiOUT pin's input signal (Note 4) | O | O |
| MR3 | 0 (Must always be "0" in event counter mode) | | O | O |
| TCK0 | Count operation type select bit | 0 : Reload type<br>1 : Free-run type | O | O |
| TCK1 | Invalid when not using two-phase pulse signal processing<br>Can be "0" or "1" | | O | O |

Note 1: In event counter mode, the count source is selected by the event / trigger select bit (addresses 0382₁₆ and 0383₁₆).
Note 2: The settings of the corresponding port register and port direction register are invalid.
Note 3: Valid only when counting an external signal.
Note 4: When an "L" signal is input to the TAiOUT pin, the downcount is activated. When "H", the upcount is activated. Set the corresponding port direction register to "0".

**Figure 1.14.8. Timer Ai mode register in event counter mode**

**Table 1.14.3. 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 TAiOUT pin |
| Count operation | • Up count or down count can be selected by two-phase pulse signal<br>• When the timer overflows or underflows, the reload register content is reloaded and the timer starts over again (Note 1) |
| Divide ratio | 1/ ($FFFF_{16}$ - n + 1) for up count<br>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 TAiOUT 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<br>When a value is written to timer A2, A3, or A4 register, it is written to both reload register and counter<br>• When counting in progress<br>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 2) | • Normal processing operation (timer A2 and timer A3)<br>The timer counts up rising edges or counts down falling edges on the TAiIN pin when input signal on the TAiOUT pin is “H”.<br>  <br><br>• Multiply-by-4 processing operation (timer A3 and timer A4)<br>If the phase relationship is such that the TAiIN pin goes “H” when the input signal on the TAiOUT pin is “H”, the timer counts up rising and falling edges on the TAiOUT and TAiIN pins. If the phase relationship is such that the TAiIN pin goes “L” when the input signal on the TAiOUT pin is “H”, the timer counts down rising and falling edges on the TAiOUT and TAiIN pins.<br>  <br> |

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

Note 2: 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 Ai mode register
(When using two-phase pulse signal processing)

| b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|
| | | 0 | 1 | 0 | 0 | 0 | 1 |

| Symbol | Address | When reset |
|---|---|---|
| TAiMR(i = 2 to 4) | $039_{16}$ to $039A_{16}$ | $00_{16}$ |

| | Bit name | Function | R | W |
|---|---|---|---|---|
| TMOD0 | Operation mode select bit | b1 b0<br>0 1 : Event counter mode | ○ | ○ |
| TMOD1 | | | ○ | ○ |
| MR0 | 0 (Must always be “0” when using two-phase pulse signal processing) | | ○ | ○ |
| MR1 | 0 (Must always be “0” when using two-phase pulse signal processing) | | ○ | ○ |
| MR2 | 1 (Must always be “1” when using two-phase pulse signal processing) | | ○ | ○ |
| MR3 | 0 (Must always be “0” when using two-phase pulse signal processing) | | ○ | ○ |
| TCK0 | Count operation type select bit | 0 : Reload type<br>1 : Free-run type | ○ | ○ |
| TCK1 | Two-phase pulse processing operation select bit (Note 1)(Note 2) | 0 : Normal processing operation<br>1 : Multiply-by-4 processing operation | ○ | ○ |

Note 1: This bit is valid for timer A3 mode register. Timer A2 is fixed to normal processing operation, and timer A4 is fixed to multiply-by-4 processing operation.
Note 2: When performing two-phase pulse signal processing, make sure the two-phase pulse signal processing operation select bit (address $0384_{16}$) is set to “1”. Also, always be sure to set the event/trigger select bit (addresses $0382_{16}$ and $0383_{16}$) to “00”.

**Figure 1.14.9. Timer Ai mode register in event counter mode**

### (3) One-shot timer mode

In this mode, the timer operates only once. (See Table 1.14.4.) When a trigger occurs, the timer starts up and continues operating for a given period. Figure 1.14.10 shows the timer Ai mode register in one-shot timer mode.

**Table 1.14.4. Timer specifications in one-shot timer mode**

| Item | Specification |
|---|---|
| Count source | $f_1$, $f_8$, $f_{32}$, $f_{C32}$ |
| Count operation | • The timer counts down<br>• When the count reaches $0000_{16}$, the timer stops counting after reloading a new count<br>• If a trigger occurs when counting, the timer reloads a new count and restarts counting |
| Divide ratio | 1/n n : Set value |
| Count start condition | • An external trigger is input<br>• The timer overflows<br>• The one-shot start flag is set (= 1) |
| Count stop condition | • A new count is reloaded after the count has reached $0000_{16}$<br>• The count start flag is reset (= 0) |
| Interrupt request generation timing | The count reaches $0000_{16}$ |
| $TAi_{IN}$ pin function | Programmable I/O port or trigger input |
| $TAi_{OUT}$ 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 | • When counting stopped<br>When a value is written to timer Ai register, it is written to both reload register and counter<br>• When counting in progress<br>When a value is written to timer Ai register, it is written to only reload register (Transferred to counter at next reload time) |

Timer Ai mode register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | 0 | | | | 1 | 0 |

| Symbol | Address | When reset |
|---|---|---|
| TAiMR(i = 0 to 4) | $0396_{16}$ to $039A_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| TMOD0 | Operation mode select bit | b1 b0<br>1 0 : One-shot timer mode | ○ | ○ |
| TMOD1 | | | ○ | ○ |
| MR0 | Pulse output function select bit | 0 : Pulse is not output (TAiOUT pin is a normal port pin)<br>1 : Pulse is output (Note 1) (TAiOUT pin is a pulse output pin) | ○ | ○ |
| MR1 | External trigger select bit (Note 2) | 0 : Falling edge of TAiIN pin's input signal (Note 3)<br>1 : Rising edge of TAiIN pin's input signal (Note 3) | ○ | ○ |
| MR2 | Trigger select bit | 0 : One-shot start flag is valid<br>1 : Selected by event/trigger select bits | ○ | ○ |
| MR3 | 0 (Must always be "0" in one-shot timer mode) | | ○ | ○ |
| TCK0 | Count source select bit | b7 b6<br>0 0 : $f_1$<br>0 1 : $f_8$<br>1 0 : $f_{32}$<br>1 1 : $f_{C32}$ | ○ | ○ |
| TCK1 | | | ○ | ○ |

Note 1: The settings of the corresponding port register and port direction register are invalid.
Note 2: Valid only when the $TAi_{IN}$ pin is selected by the event/trigger select bit (addresses $0382_{16}$ and $0383_{16}$). If timer overflow is selected, this bit can be "1" or "0".
Note 3: Set the corresponding port direction register to "0".

**Figure 1.14.10. Timer Ai mode register in one-shot timer mode**

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

In this mode, the timer outputs pulses of a given width in succession. (See Table 1.14.5.) In this mode, the counter functions as either a 16-bit pulse width modulator or an 8-bit pulse width modulator. Figure 1.14.11 shows the timer Ai mode register in pulse width modulation mode. Figure 1.14.12 shows the example of how a 16-bit pulse width modulator operates. Figure 1.14.13 shows the example of how an 8-bit pulse width modulator operates.

**Table 1.14.5. Timer specifications in pulse width modulation mode**

| Item | Specification |
|---|---|
| Count source | $f_1$, $f_8$, $f_{32}$, $f_{C32}$ |
| Count operation | • The timer counts down (operating as an 8-bit or a 16-bit pulse width modulator)<br>• The timer reloads a new count at a rising edge of PWM pulse and continues counting<br>• The timer is not affected by a trigger that occurs when counting |
| 16-bit PWM | • High level width n / fi n : Set value<br>• Cycle time $(2^{16}-1)$ / fi fixed |
| 8-bit PWM | • High level width n×(m+1) / fi n : values set to timer Ai register's high-order address<br>• Cycle time $(2^{8}-1)$× (m+1) / fi m : values set to timer Ai register's low-order address |
| Count start condition | • External trigger is input<br>• The timer overflows<br>• The count start flag is set (= 1) |
| Count stop condition | • The count start flag is reset (= 0) |
| Interrupt request generation timing | PWM pulse goes "L" |
| $TAi_{IN}$ pin function | Programmable I/O port or trigger input |
| $TAi_{OUT}$ pin function | Pulse output |
| Read from timer | When timer Ai register is read, it indicates an indeterminate value |
| Write to timer | • When counting stopped<br>When a value is written to timer Ai register, it is written to both reload register and counter<br>• When counting in progress<br>When a value is written to timer Ai register, it is written to only reload register (Transferred to counter at next reload time) |

Timer Ai mode register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | 1 | 1 | 1 |

| Symbol | Address | When reset |
|---|---|---|
| TAiMR(i=0 to 4) | $039_{16}$ to $039A_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| TMOD0 | Operation mode select bit | b1 b0<br>1 1 : PWM mode | ○ | ○ |
| TMOD1 | | | ○ | ○ |
| MR0 | 1 (Must always be "1" in PWM mode) | | ○ | ○ |
| MR1 | External trigger select bit (Note 1) | 0: Falling edge of $TAi_{IN}$ pin's input signal (Note 2)<br>1: Rising edge of $TAi_{IN}$ pin's input signal (Note 2) | ○ | ○ |
| MR2 | Trigger select bit | 0: Count start flag is valid<br>1: Selected by event/trigger select bits | ○ | ○ |
| MR3 | 16/8-bit PWM mode select bit | 0: Functions as a 16-bit pulse width modulator<br>1: Functions as an 8-bit pulse width modulator | ○ | ○ |
| TCK0 | Count source select bit | b7 b6<br>0 0 : $f_1$<br>0 1 : $f_8$<br>1 0 : $f_{32}$<br>1 1 : $f_{C32}$ | ○ | ○ |
| TCK1 | | | ○ | ○ |

Note 1: Valid only when the $TAi_{IN}$ pin is selected by the event/trigger select bit (addresses $038_{16}$ and $038_{16}$). If timer overflow is selected, this bit can be "1" or "0".
Note 2: Set the corresponding port direction register to "0".

**Figure 1.14.11. Timer Ai mode register in pulse width modulation mode**

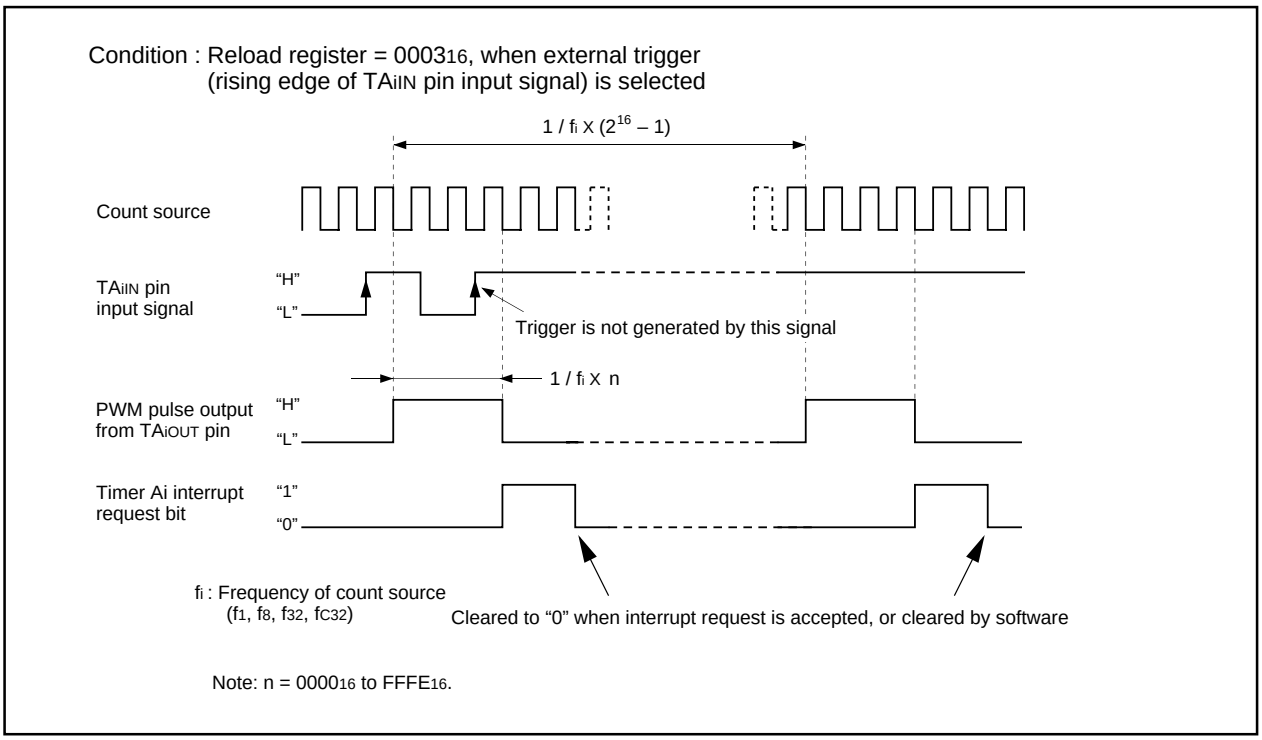



**Figure 1.14.12. Example of how a 16-bit pulse width modulator operates**

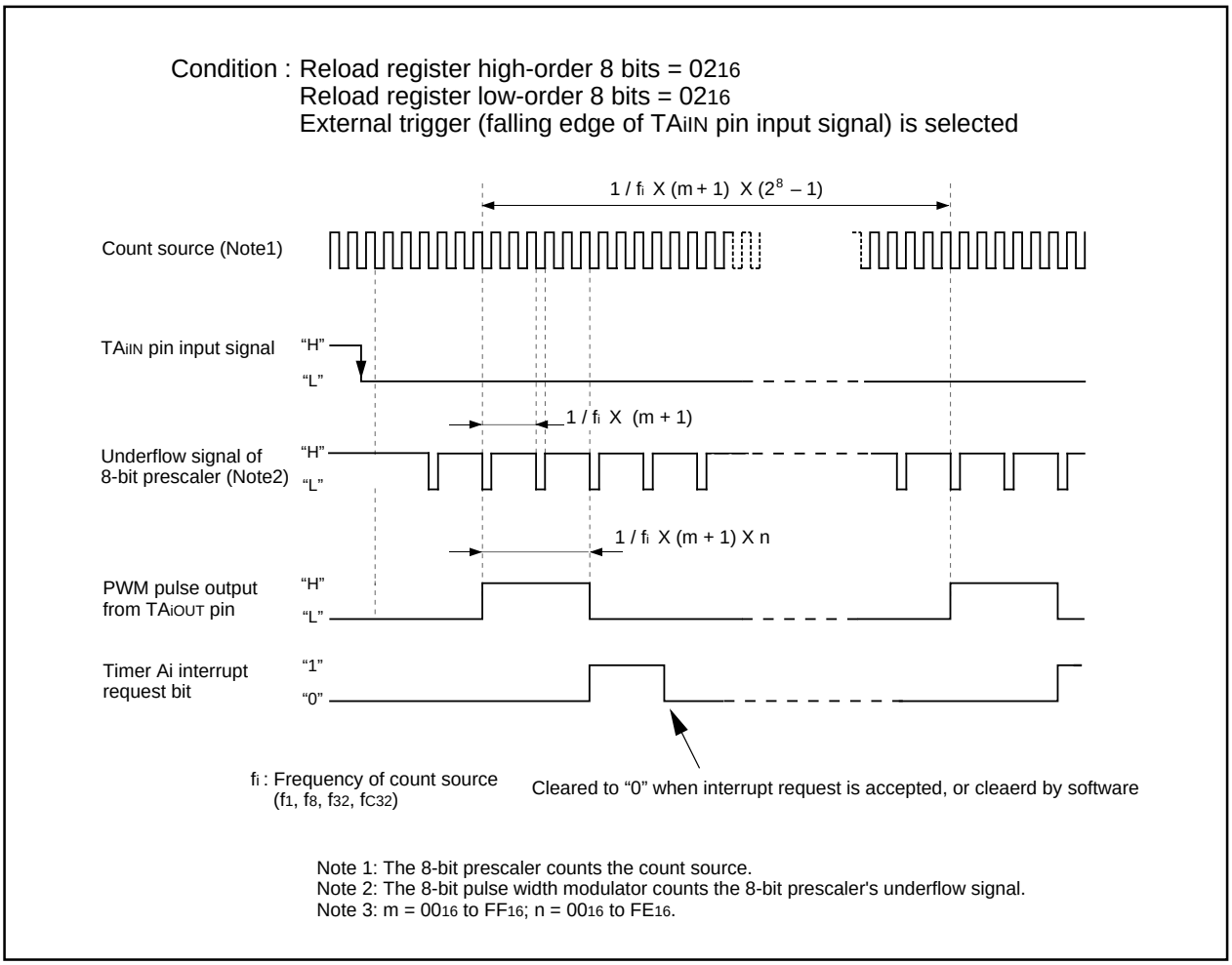



**Figure 1.14.13. Example of how an 8-bit pulse width modulator operates**

## Timer B

Figure 1.14.14 shows the block diagram of timer B. Figures 1.14.15 and 1.14.16 show the timer B-related registers.

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.

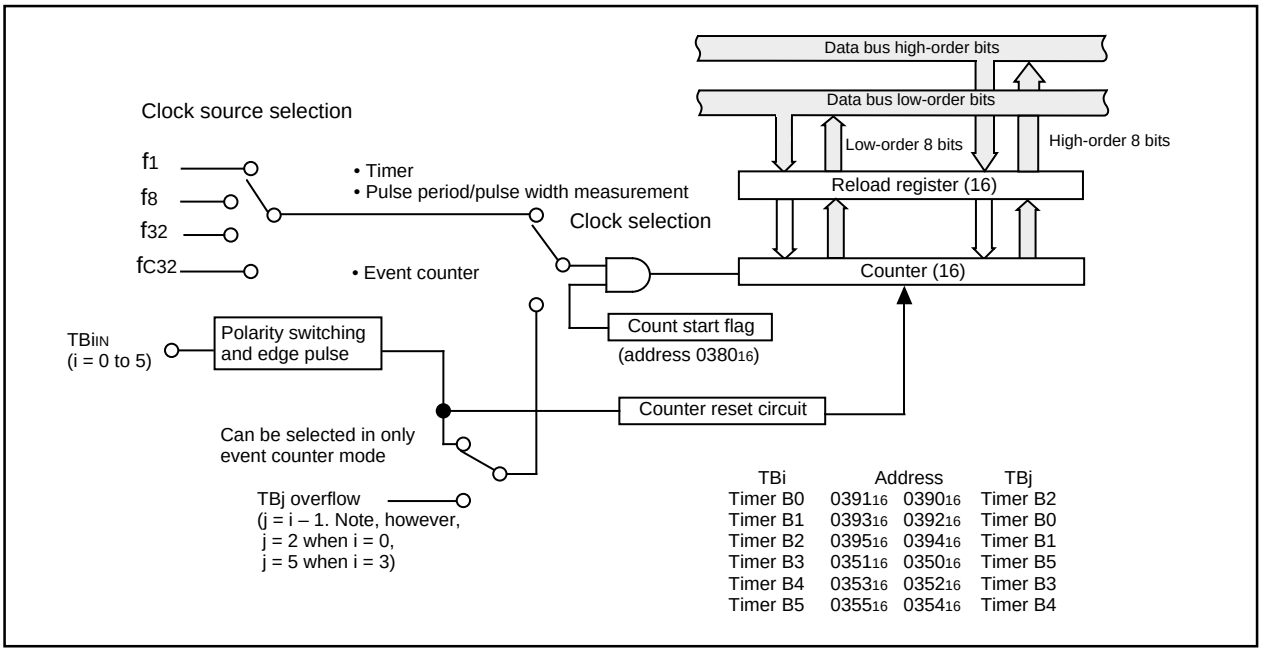



**Figure 1.14.14. Block diagram of timer B**

Timer Bi mode register

| Symbol | Address | When reset |
|---|---|---|
| TBiMR(i = 0 to 5) | 039B₁₆ to 039D₁₆ | 00XX0000₂ |
| | 035B₁₆ to 035D₁₆ | 00XX0000₂ |

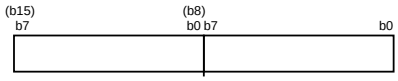
b7 b6 b5 b4 b3 b2 b1 b0

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| TMOD0 | Operation mode select bit | b1 b0<br>0 0 : Timer mode<br>0 1 : Event counter mode<br>1 0 : Pulse period/pulse width measurement mode<br>1 1 : Must not be set. | ○ | ○ |
| TMOD1 | | | ○ | ○ |
| MR0 | Function varies with each operation mode | | ○ | ○ |
| MR1 | | | ○ | ○ |
| MR2 | | | ○ (Note 1)<br>× (Note 2) | ○<br>× |
| MR3 | | | ○ | × |
| TCK0 | Count source select bit (Function varies with each operation mode) | | ○ | ○ |
| TCK1 | | | ○ | ○ |

Note 1: Timer B0, timer B3.
Note 2: Timer B1, timer B2, timer B4, timer B5.

**Figure 1.14.15. Timer B-related registers (1)**

### Timer Bi register (Note)

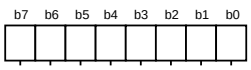



| Symbol | Address | When reset |
|---|---|---|
| TB0 | $039_{16}$, $039_{0_{16}}$ | Indeterminate |
| TB1 | $039_{3_{16}}$, $039_{2_{16}}$ | Indeterminate |
| TB2 | $039_{5_{16}}$, $039_{4_{16}}$ | Indeterminate |
| TB3 | $035_{1_{16}}$, $035_{0_{16}}$ | Indeterminate |
| TB4 | $035_{3_{16}}$, $035_{2_{16}}$ | Indeterminate |
| TB5 | $035_{5_{16}}$, $035_{4_{16}}$ | Indeterminate |

| Function | Values that can be set | R | W |
|---|---|---|---|
| • Timer mode<br>Counts the timer's period | $0000_{16}$ to $FFFF_{16}$ | ○ | ○ |
| • Event counter mode<br>Counts external pulses input or a timer overflow | $0000_{16}$ to $FFFF_{16}$ | ○ | ○ |
| • Pulse period / pulse width measurement mode<br>Measures a pulse period or width | —— | ○ | × |

Note: Read and write data in 16-bit units.

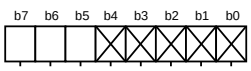
### Count start flag




| Symbol | Address | When reset |
|---|---|---|
| TABSR | $038_{0_{16}}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| TA0S | Timer A0 count start flag | 0 : Stops counting<br>1 : Starts counting | ○ | ○ |
| TA1S | Timer A1 count start flag | | ○ | ○ |
| TA2S | Timer A2 count start flag | | ○ | ○ |
| TA3S | Timer A3 count start flag | | ○ | ○ |
| TA4S | Timer A4 count start flag | | ○ | ○ |
| TB0S | Timer B0 count start flag | | ○ | ○ |
| TB1S | Timer B1 count start flag | | ○ | ○ |
| TB2S | Timer B2 count start flag | | ○ | ○ |

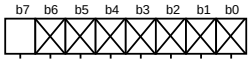
### Timer B3, 4, 5 count start flag




| Symbol | Address | When reset |
|---|---|---|
| TBSR | $034_{0_{16}}$ | $000XXXXX_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be indeterminate. | | | — | — |
| TB3S | Timer B3 count start flag | 0 : Stops counting<br>1 : Starts counting | ○ | ○ |
| TB4S | Timer B4 count start flag | | ○ | ○ |
| TB5S | Timer B5 count start flag | | ○ | ○ |

### Clock prescaler reset flag




| Symbol | Address | When reset |
|---|---|---|
| CPSRF | $038_{1_{16}}$ | $0XXXXXXX_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be indeterminate. | | | — | — |
| CPSR | Clock prescaler reset flag | 0 : No effect<br>1 : Prescaler is reset<br>(When read, the value is “0”) | ○ | ○ |

**Figure 1.14.16. Timer B-related registers (2)**

### (1) Timer mode

In this mode, the timer counts an internally generated count source. (See Table 1.14.6.) Figure 1.14.17 shows the timer Bi mode register in timer mode.

**Table 1.14.6. Timer specifications in timer mode**

| Item | Specification |
|---|---|
| Count source | $f_1$, $f_8$, $f_{32}$, $fc_{32}$ |
| Count operation | • Counts down<br>• When the timer underflows, it reloads the reload register contents before continuing counting |
| 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 | • When counting stopped<br>When a value is written to timer Bi register, it is written to both reload register and counter<br>• When counting in progress<br>When a value is written to timer Bi register, it is written to only reload register (Transferred to counter at next reload time) |

Timer Bi mode register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | 0 | 0 |

| Symbol | Address | When reset |
|---|---|---|
| TBiMR(i=0 to 5) | $039B_{16}$ to $039D_{16}$ | $00XX0000_2$ |
| | $035B_{16}$ to $035D_{16}$ | $00XX0000_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| TMOD0 | Operation mode select bit | b1 b0<br>0 0 : Timer mode | ○ | ○ |
| TMOD1 | | | ○ | ○ |
| MR0 | Invalid in timer mode<br>Can be “0” or “1” | | ○ | ○ |
| MR1 | | | ○ | ○ |
| MR2 | 0 (Must always be “0” in timer mode ; i = 0, 3) | | ○ (Note 1) | ○ |
| | Nothing is assiigned (i = 1, 2, 4, 5).<br>In an attempt to write to this bit, write “0”. The value, if read, turns out to be indeterminate. | | × (Note 2) | × |
| MR3 | Invalid in timer mode.<br>In an attempt to write to this bit, write “0”. The value, if read in timer mode, turns out to be indeterminate. | | ○ | × |
| TCK0 | Count source select bit | b7 b6<br>0 0 : $f_1$<br>0 1 : $f_8$<br>1 0 : $f_{32}$<br>1 1 : $fc_{32}$ | ○ | ○ |
| TCK1 | | | ○ | ○ |

Note 1: Timer B0, timer B3.
Note 2: Timer B1, timer B2, timer B4, timer B5.

**Figure 1.14.17. Timer Bi mode register in timer mode**

### (2) Event counter mode

In this mode, the timer counts an external signal or an internal timer's overflow. (See Table 1.14.7.) Figure 1.14.18 shows the timer Bi mode register in event counter mode.

**Table 1.14.7. Timer specifications in event counter mode**

| Item | Specification |
|---|---|
| Count source | • External signals input to TBiIN pin<br>• Effective edge of count source can be a rising edge, a falling edge, or falling and rising edges as selected by software |
| Count operation | • Counts down<br>• When the timer underflows, it reloads the reload register contents before continuing counting |
| 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 | • When counting stopped<br>When a value is written to timer Bi register, it is written to both reload register and counter<br>• When counting in progress<br>When a value is written to timer Bi register, it is written to only reload register (Transferred to counter at next reload time) |

Timer Bi mode register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | 0 | 1 |

| Symbol | Address | When reset |
|---|---|---|
| TBiMR(i=0 to 5) | 039B16 to 039D16 | 00XX0000₂ |
| | 035B16 to 035D16 | 00XX0000₂ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| TMOD0 | Operation mode select bit | b1 b0<br>0 1 : Event counter mode | ○ | ○ |
| TMOD1 | | | ○ | ○ |
| MR0 | Count polarity select bit (Note 1) | b3 b2<br>0 0 : Counts external signal's falling edges<br>0 1 : Counts external signal's rising edges<br>1 0 : Counts external signal's falling and rising edges<br>1 1 : Must not be set. | ○ | ○ |
| MR1 | | | ○ | ○ |
| MR2 | 0 (Must always be "0" in event counter mode; i = 0, 3) | | ○ (Note 2) | ○ |
| | Nothing is assigned (i = 1, 2, 4, 5).<br>In an attempt to write to this bit, write "0". The value, if read, turns out to be indeterminate. | | × (Note 3) | × |
| MR3 | Invalid in event counter mode.<br>In an attempt to write to this bit, write "0". The value, if read in event counter mode, turns out to be indeterminate. | | ○ | × |
| TCK0 | Invalid in event counter mode.<br>Can be "0" or "1". | | ○ | ○ |
| TCK1 | Event clock select | 0 : Input from TBiIN pin (Note 4)<br>1 : TBj overflow<br>(j = i – 1; however, j = 2 when i = 0, j = 5 when i = 3) | ○ | ○ |

Note 1: Valid only when input from the TBiIN pin is selected as the event clock.
If timer's overflow is selected, this bit can be "0" or "1".
Note 2: Timer B0, timer B3.
Note 3: Timer B1, timer B2, timer B4, timer B5.
Note 4: Set the corresponding port direction register to "0".

**Figure 1.14.18. Timer Bi mode register in event counter mode**

### (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 1.14.8.) Figure 1.14.19 shows the timer Bi mode register in pulse period/pulse width measurement mode. Figure 1.14.20 shows the operation timing when measuring a pulse period. Figure 1.14.21 shows the operation timing when measuring a pulse width.

**Table 1.14.8. Timer specifications in pulse period/pulse width measurement mode**

| Item | Specification |
|---|---|
| Count source | $f_1$, $f_8$, $f_{32}$, $f_{C32}$ |
| Count operation | • Up count<br>• Counter value "$0000_{16}$" is transferred to reload register at measurement pulse's effective edge and the timer continues counting |
| Count start condition | Count start flag is set (= 1) |
| Count stop condition | Count start flag is reset (= 0) |
| Interrupt request generation timing | • When measurement pulse's effective edge is input (Note 1)<br>• When an overflow occurs. (Simultaneously, the timer Bi overflow flag changes to "1". The timer Bi overflow flag changes to "0" when the count start flag is "1" and a value is written to the timer Bi mode register.) |
| $TBi_{IN}$ 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.

Timer Bi mode register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | 1 | 0 |

| Symbol | Address | When reset |
|---|---|---|
| TBiMR(i=0 to 5) | $039B_{16}$ to $039D_{16}$ | $00XX0000_2$ |
| | $035B_{16}$ to $035D_{16}$ | $00XX0000_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| TMOD0 | Operation mode select bit | b1 b0<br>1 0 : Pulse period / pulse width measurement mode | ○ | ○ |
| TMOD1 | | | ○ | ○ |
| MR0 | Measurement mode select bit | b3 b2<br>0 0 : Pulse period measurement (Interval between measurement pulse's falling edge to falling edge)<br>0 1 : Pulse period measurement (Interval between measurement pulse's rising edge to rising edge)<br>1 0 : Pulse width measurement (Interval between measurement pulse's falling edge to rising edge, and between rising edge to falling edge)<br>1 1 : Must not be set. | ○ | ○ |
| MR1 | | | ○ | ○ |
| MR2 | 0 (Must always be "0" in pulse period/pulse width measurement mode; i = 0, 3) | | ○ (Note 2) | ○ |
| | Nothing is assigned (i = 1, 2, 4, 5).<br>In an attempt to write to this bit, write "0". The value, if read, turns out to be indeterminate. | | × (Note 3) | × |
| MR3 | Timer Bi overflow flag ( Note 1) | 0 : Timer did not overflow<br>1 : Timer has overflowed | ○ | × |
| TCK0 | Count source select bit | b7 b6<br>0 0 : $f_1$<br>0 1 : $f_8$<br>1 0 : $f_{32}$<br>1 1 : $f_{C32}$ | ○ | ○ |
| TCK1 | | | ○ | ○ |

Note 1: It is indeterminate when reset. 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". This flag cannot be set to "1" by software.
Note 2: Timer B0, timer B3.
Note 3: Timer B1, timer B2, timer B4, timer B5.

**Figure 1.14.19. Timer Bi mode register in pulse period/pulse width measurement mode**

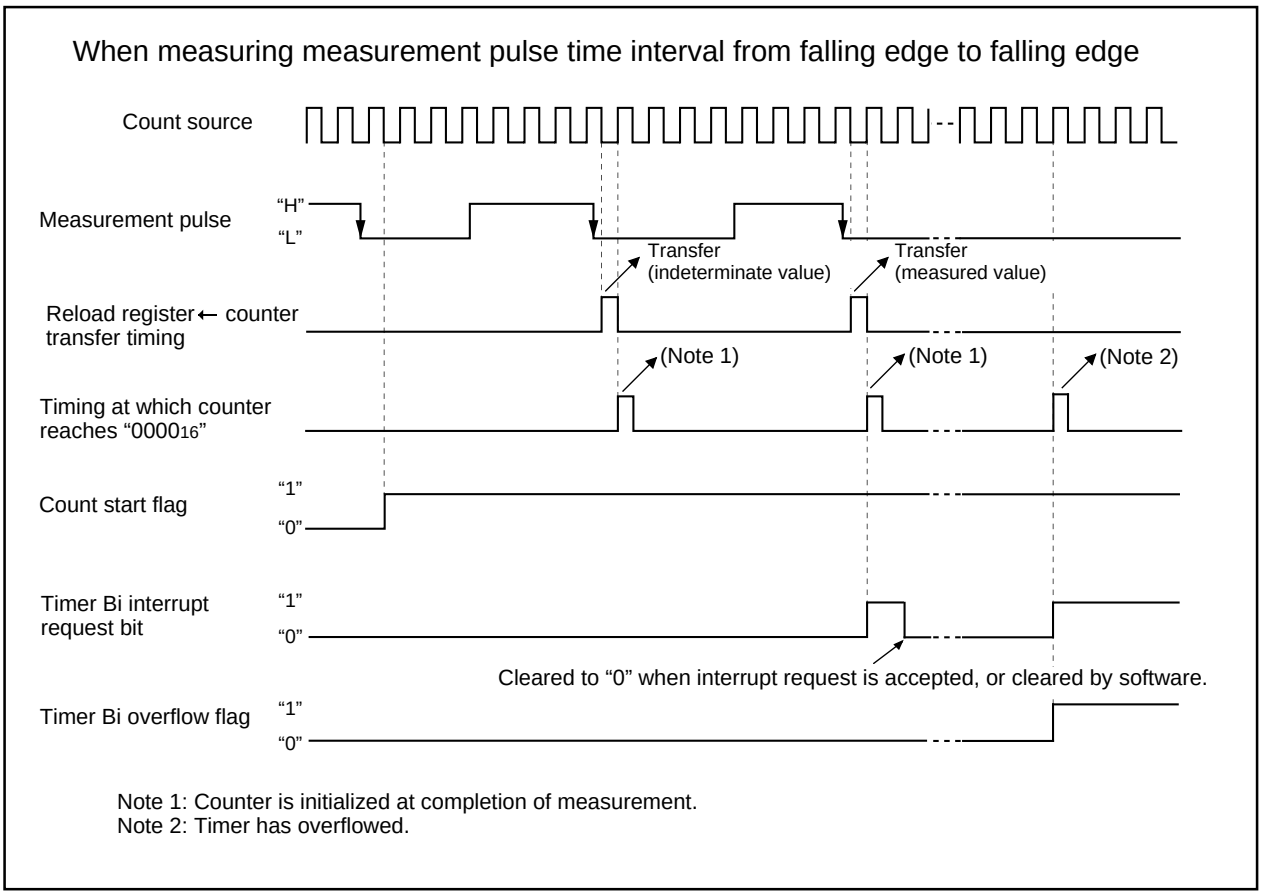



**Figure 1.14.20. Operation timing when measuring a pulse period**

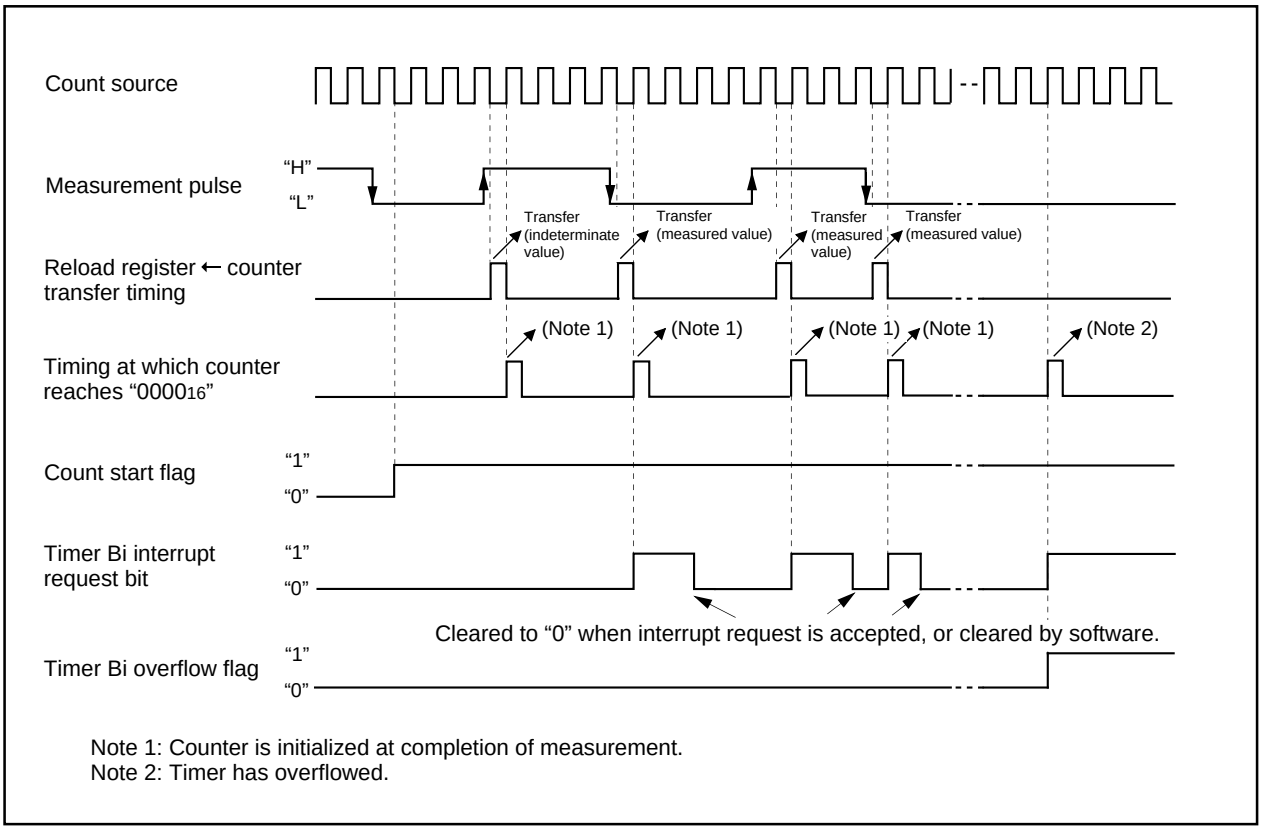



**Figure 1.14.21. Operation timing when measuring a pulse width**

## Timers' functions for three-phase motor control

Use of more than one built-in timer A and timer B provides the means of outputting three-phase motor driving waveforms.

Figures 1.15.1 to 1.15.3 show registers related to timers for three-phase motor control.

Three-phase PWM control register 0

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| INVC0 | $0348_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Description | R | W |
|---|---|---|---|---|
| INV00 | Effective interrupt output polarity select bit | 0: A timer B2 interrupt occurs when the timer A1 reload control signal is "1".<br>1: A timer B2 interrupt occurs when the timer A1 reload control signal is "0".<br>**Effective only in three-phase mode 1** | ○ | ○ |
| INV01 | Effective interrupt output specification bit (Note 4) | 0: Not specified.<br>1: Selected by the effective interrupt output polarity selection bit.<br>**Effective only in three-phase mode 1** | ○ | ○ |
| INV02 | Mode select bit (Note 2) | 0: Normal mode<br>1: Three-phase PWM output mode | ○ | ○ |
| INV03 | Output control bit | 0: Output disabled<br>1: Output enabled | ○ | ○ |
| INV04 | Positive and negative phases concurrent L output disable function enable bit | 0: Feature disabled<br>1: Feature enabled | ○ | ○ |
| INV05 | Positive and negative phases concurrent L output detect flag | 0: Not detected yet<br>1: Already detected | ○ | ○ (Note 1) |
| INV06 | Modulation mode select bit (Note 3) | 0: Triangular wave modulation mode<br>1: Sawtooth wave modulation mode | ○ | ○ |
| INV07 | Software trigger bit | 1: Trigger generated<br>The value, when read, is "0". | ○ | ○ |

Note 1: No value other than "0" can be written.
Note 2: Selecting three-phase PWM output mode causes $P8_0$, $P8_1$, and $P7_2$ through $P7_5$ to output U, $\overline{U}$, V, $\overline{V}$, W, and $\overline{W}$, and works the timer for setting short circuit prevention time, the U, V, W phase output control circuits, and the circuit for setting timer B2 interrupt frequency.
Note 3: **In triangular wave modulation mode:**
The short circuit prevention timer starts in synchronization with the falling edge of timer Ai output.
The data transfer from the three-phase buffer register to the three-phase output shift register is made only once in synchronization with the transfer trigger signal after writing to the three-phase output buffer register.
**In sawtooth wave modulation mode:**
The short circuit prevention timer starts in synchronization with the falling edge of timer A output and with the transfer trigger signal.
The data transfer from the three-phase output buffer register to the three-phase output shift register is made with respect to every transfer trigger.
Note 4: To write "1" to bit 1 (INV01) of the three-phase PWM control register 0, set in advance the content of the timer B2 interrupt occurrences frequency set counter.

Three-phase PWM control register 1

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | 0 | | | | |

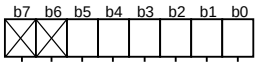
| Symbol | Address | When reset |
|---|---|---|
| INVC1 | $0349_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Description | R | W |
|---|---|---|---|---|
| INV10 | Timer Ai start trigger signal select bit | 0: Timer B2 overflow signal<br>1: Timer B2 overflow signal, signal for writing to timer B2 | ○ | ○ |
| INV11 | Timer A1-1, A2-1, A4-1 control bit | 0: Three-phase mode 0<br>1: Three-phase mode 1 | ○ | ○ |
| INV12 | Short circuit timer count source select bit | 0 : Must not be set.<br>1 : $f_1/2$ (Note) | ○ | ○ |
| Noting is assigned.<br>In an attempt to write to this bit, write "0". The value, if read, turns out to be "0". | | | — | — |
| Reserved bit | | Must always be set to "0" | ○ | ○ |
| Noting is assigned.<br>In an attempt to write to these bits, write "0". The value, if read, turns out to be "0". | | | — | — |

Note : To use three-phase PWM output mode, write "1" to INV12.

**Figure 1.15.1. Registers related to timers for three-phase motor control**

### Three-phase output buffer register 0




| Symbol | Address | When reset |
|---|---|---|
| IDB0 | $034A_{16}$ | $00_{16}$ |

| Bit Symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| DU0 | U phase output buffer 0 | Setting in U phase output buffer 0 | (Note) | ○ |
| DUB0 | $\overline{U}$ phase output buffer 0 | Setting in $\overline{U}$ phase output buffer 0 | (Note) | ○ |
| DV0 | V phase output buffer 0 | Setting in V phase output buffer 0 | (Note) | ○ |
| DVB0 | $\overline{V}$ phase output buffer 0 | Setting in $\overline{V}$ phase output buffer 0 | (Note) | ○ |
| DW0 | W phase output buffer 0 | Setting in W phase output buffer 0 | (Note) | ○ |
| DWB0 | $\overline{W}$ phase output buffer 0 | Setting in $\overline{W}$ phase output buffer 0 | (Note) | ○ |
| Nothing is assigned. In an attempt to write to these bits, write "0". The value, if read, turns out to be "0". | | | — | — |

Note: When executing read instruction of this register, the contents of three-phase shift register is read out.

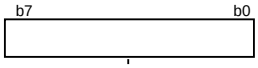
### Three-phase output buffer register 1

| Symbol | Address | When reset |
|---|---|---|
| IDB1 | $034B_{16}$ | $00_{16}$ |

| Bit Symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| DU1 | U phase output buffer 1 | Setting in U phase output buffer 1 | (Note) | ○ |
| DUB1 | $\overline{U}$ phase output buffer 1 | Setting in $\overline{U}$ phase output buffer 1 | (Note) | ○ |
| DV1 | V phase output buffer 1 | Setting in V phase output buffer 1 | (Note) | ○ |
| DVB1 | $\overline{V}$ phase output buffer 1 | Setting in $\overline{V}$ phase output buffer 1 | (Note) | ○ |
| DW1 | W phase output buffer 1 | Setting in W phase output buffer 1 | (Note) | ○ |
| DWB1 | $\overline{W}$ phase output buffer 1 | Setting in $\overline{W}$ phase output buffer 1 | (Note) | ○ |
| Nothing is assigned. In an attempt to write to these bits, write "0". The value, if read, turns out to be "0". | | | — | — |

Note: When executing read instruction of this register, the contents of three-phase shift register is read out.

### Dead time timer (Note)

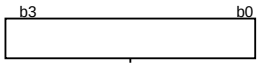



| Symbol | Address | When reset |
|---|---|---|
| DTT | $034C_{16}$ | Indeterminate |

| Function | Values that can be set | R | W |
|---|---|---|---|
| Set dead time timer | 1 to 255 | — | ○ |

Note: Use MOV instruction to write to this register.

### Timer B2 interrupt occurrences frequency set counter (Note 1, 2, 3)




| Symbol | Address | When reset |
|---|---|---|
| ICTB2 | $034D_{16}$ | Indeterminate |

| Function | Values that can be set | R | W |
|---|---|---|---|
| Set occurrence frequency of timer B2 interrupt request | 1 to 15 | — | ○ |

Note 1: In setting 1 to bit 1 (INV01) - the effective interrupt output specification bit - of three-phase PWM control register 0, do not change the B2 interrupt occurrences frequency set counter to deal with the timer function for three-phase motor control.
Note 2: Do not write at the timing of an overflow occurrence in timer B2.
Note 3: Use MOV instruction to write to this register.

**Figure 1.15.2. Registers related to timers for three-phase motor control**

### Timer Ai register (Note 1)

(b15) b7 — (b8) b0 b7 — b0

| Symbol | Address | When reset |
|---|---|---|
| TA1 | $0389_{16}$,$0388_{16}$ | Indeterminate |
| TA2 | $038B_{16}$,$038A_{16}$ | Indeterminate |
| TA4 | $038F_{16}$,$038E_{16}$ | Indeterminate |
| TB2 | $0395_{16}$,$0394_{16}$ | Indeterminate |

| Function | Values that can be set | R | W |
|---|---|---|---|
| • Timer mode<br>Counts an internal count source | $0000_{16}$ to $FFFF_{16}$ | ○ | ○ |
| • One-shot timer mode<br>Counts a one shot width | $0000_{16}$ to $FFFF_{16}$<br>(Note 2, 3) | × | ○ |

Note 1: Read and write data in 16-bit units.
Note 2: When the timer Ai register is set to "$0000_{16}$", the counter does not operate and a timer Ai interrupt does not occur.
Note 3: Use MOV instruction to write to this register.

### Timer Ai-1 register (Note)

(b15) b7 — (b8) b0 b7 — b0

| Symbol | Address | When reset |
|---|---|---|
| TA11 | $0343_{16}$,$0342_{16}$ | Indeterminate |
| TA21 | $0345_{16}$,$0344_{16}$ | Indeterminate |
| TA41 | $0347_{16}$,$0346_{16}$ | Indeterminate |

| Function | Values that can be set | R | W |
|---|---|---|---|
| Counts an internal count source | $0000_{16}$ to $FFFF_{16}$ | ○ | ○ |

Note: Read and write data in 16-bit units.

### Trigger select register

b7 b6 b5 b4 b3 b2 b1 b0

| Symbol | Address | When reset |
|---|---|---|
| TRGSR | $0383_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| TA1TGL | Timer A1 event/trigger select bit | b1 b0<br>0 0 : Input on $TA1_{IN}$ is selected (Note)<br>0 1 : TB2 overflow is selected<br>1 0 : TA0 overflow is selected<br>1 1 : TA2 overflow is selected | ○ | ○ |
| TA1TGH | | | ○ | ○ |
| TA2TGL | Timer A2 event/trigger select bit | b3 b2<br>0 0 : Input on $TA2_{IN}$ is selected (Note)<br>0 1 : TB2 overflow is selected<br>1 0 : TA1 overflow is selected<br>1 1 : TA3 overflow is selected | ○ | ○ |
| TA2TGH | | | ○ | ○ |
| TA3TGL | Timer A3 event/trigger select bit | b5 b4<br>0 0 : Input on $TA3_{IN}$ is selected (Note)<br>0 1 : TB2 overflow is selected<br>1 0 : TA2 overflow is selected<br>1 1 : TA4 overflow is selected | ○ | ○ |
| TA3TGH | | | ○ | ○ |
| TA4TGL | Timer A4 event/trigger select bit | b7 b6<br>0 0 : Input on $TA4_{IN}$ is selected (Note)<br>0 1 : TB2 overflow is selected<br>1 0 : TA3 overflow is selected<br>1 1 : TA0 overflow is selected | ○ | ○ |
| TA4TGH | | | ○ | ○ |

Note: Set the corresponding port direction register to "0".

### Count start flag

b7 b6 b5 b4 b3 b2 b1 b0

| Symbol | Address | When reset |
|---|---|---|
| TABSR | $0380_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| TA0S | Timer A0 count start flag | 0 : Stops counting<br>1 : Starts counting | ○ | ○ |
| TA1S | Timer A1 count start flag | | ○ | ○ |
| TA2S | Timer A2 count start flag | | ○ | ○ |
| TA3S | Timer A3 count start flag | | ○ | ○ |
| TA4S | Timer A4 count start flag | | ○ | ○ |
| TB0S | Timer B0 count start flag | | ○ | ○ |
| TB1S | Timer B1 count start flag | | ○ | ○ |
| TB2S | Timer B2 count start flag | | ○ | ○ |

**Figure 1.15.3. Registers related to timers for three-phase motor control**

## Three-phase motor driving waveform output mode (three-phase PWM output mode)

Setting "1" in the mode select bit (bit 2 at $0348_{16}$) shown in Figure 1.15.1 - causes three-phase PWM output mode that uses four timers A1, A2, A4, and B2 to be selected. As shown in Figure 1.15.4, set timers A1, A2, and A4 in one-shot timer mode, set the trigger in timer B2, and set timer B2 in timer mode using the respective timer mode registers.

Timer Ai mode register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | 0 | 1 | | 0 | 1 | 0 |

| Symbol | Address | When reset |
|---|---|---|
| TA1MR | $0397_{16}$ | $00_{16}$ |
| TA2MR | $0398_{16}$ | $00_{16}$ |
| TA3MR | $039A_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| TMOD0 | Operation mode select bit | b1 b0<br>1 0 : One-shot timer mode | ○ | ○ |
| TMOD1 | | | ○ | ○ |
| MR0 | Pulse output function select bit | 0 (Must always be "0" in three-phase PWM output mode) | ○ | ○ |
| MR1 | External trigger select bit | Invalid in three-phase PWM output mode | ○ | ○ |
| MR2 | Trigger select bit | 1 : Selected by event/trigger select register | ○ | ○ |
| MR3 | 0 (Must always be "0" in one-shot timer mode) | | ○ | ○ |
| TCK0 | Count source select bit | b7 b6<br>0 0 : $f_1$<br>0 1 : $f_8$<br>1 0 : $f_{32}$<br>1 1 : $fc_{32}$ | ○ | ○ |
| TCK1 | | | ○ | ○ |

Timer B2 mode register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | 0 | | | 0 | 0 |

| Symbol | Address | When reset |
|---|---|---|
| TB2MR | $039D_{16}$ | $00XX0000_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| TMOD0 | Operation mode select bit | b1 b0<br>0 0 : Timer mode | ○ | ○ |
| TMOD1 | | | ○ | ○ |
| MR0 | Invalid in timer mode<br>Can be "0" or "1" | | ○ | ○ |
| MR1 | | | ○ | ○ |
| MR2 | 0 (Must always be "0" in timer mode) | | ○ | ○ |
| MR3 | Invalid in timer mode.<br>In an attempt to write to this bit, write "0". When read in timer mode, its content is indeterminate. | | ○ | × |
| TCK0 | Count source select bit | b7 b6<br>0 0 : $f_1$<br>0 1 : $f_8$<br>1 0 : $f_{32}$<br>1 1 : $fc_{32}$ | ○ | ○ |
| TCK1 | | | ○ | ○ |

**Figure 1.15.4. Timer mode registers in three-phase PWM output mode**

Figure 1.15.5 shows the block diagram for three-phase PWM output mode. In three-phase PWM output mode, the positive-phase waveforms (U phase, V phase, and W phase) and negative waveforms ($\overline{U}$ phase, $\overline{V}$ phase, and $\overline{W}$ phase), six waveforms in total, are output from $P8_0$, $P8_1$, $P7_2$, $P7_3$, $P7_4$, and $P7_5$ as active on the "L" level. Of the timers used in this mode, timer A4 controls the U phase and $\overline{U}$ phase, timer A1 controls the V phase and $\overline{V}$ phase, and timer A2 controls the W phase and $\overline{W}$ phase respectively; timer B2 controls the periods of one-shot pulse output from timers A4, A1, and A2.

In outputting a waveform, dead time can be set so as to cause the "L" level of the positive waveform output (U phase, V phase, and W phase) not to lap over the "L" level of the negative waveform output ($\overline{U}$ phase, $\overline{V}$ phase, and $\overline{W}$ phase).

To set short circuit time, use three 8-bit timers sharing the reload register for setting dead time. A value from 1 through 255 can be set as the count of the timer for setting dead time. The timer for setting dead time works as a one-shot timer. If a value is written to the dead time timer ($034C_{16}$), the value is written to the reload register shared by the three timers for setting dead time.

Any of the timers for setting dead time takes the value of the reload register into its counter, if a start trigger comes from its corresponding timer, and performs a down count in line with the clock source selected by the dead time timer count source select bit (bit 2 at $0349_{16}$). The timer can receive another trigger again before the workings due to the previous trigger are completed. In this instance, the timer performs a down count from the reload register's content after its transfer, provoked by the trigger, to the timer for setting dead time.

Since the timer for setting dead time works as a one-shot timer, it starts outputting pulses if a trigger comes; it stops outputting pulses as soon as its content becomes $00_{16}$, and waits for the next trigger to come.

The positive waveforms (U phase, V phase, and W phase) and the negative waveforms ($\overline{U}$ phase, $\overline{V}$ phase, and $\overline{W}$ phase) in three-phase PWM output mode are output from respective ports by means of setting "1" in the output control bit (bit 3 at $0348_{16}$). Setting "0" in this bit causes the ports to be the state of set by port direction register. This bit can be set to "0" not only by use of the applicable instruction, but by entering a falling edge in the $\overline{NMI}$ terminal or by resetting. Also, if "1" is set in the positive and negative phases concurrent L output disable function enable bit (bit 4 at $0348_{16}$) causes one of the pairs of U phase and $\overline{U}$ phase, V phase and $\overline{V}$ phase, and W phase and $\overline{W}$ phase concurrently go to "L", as a result, the port becomes the state of set by port direction register.

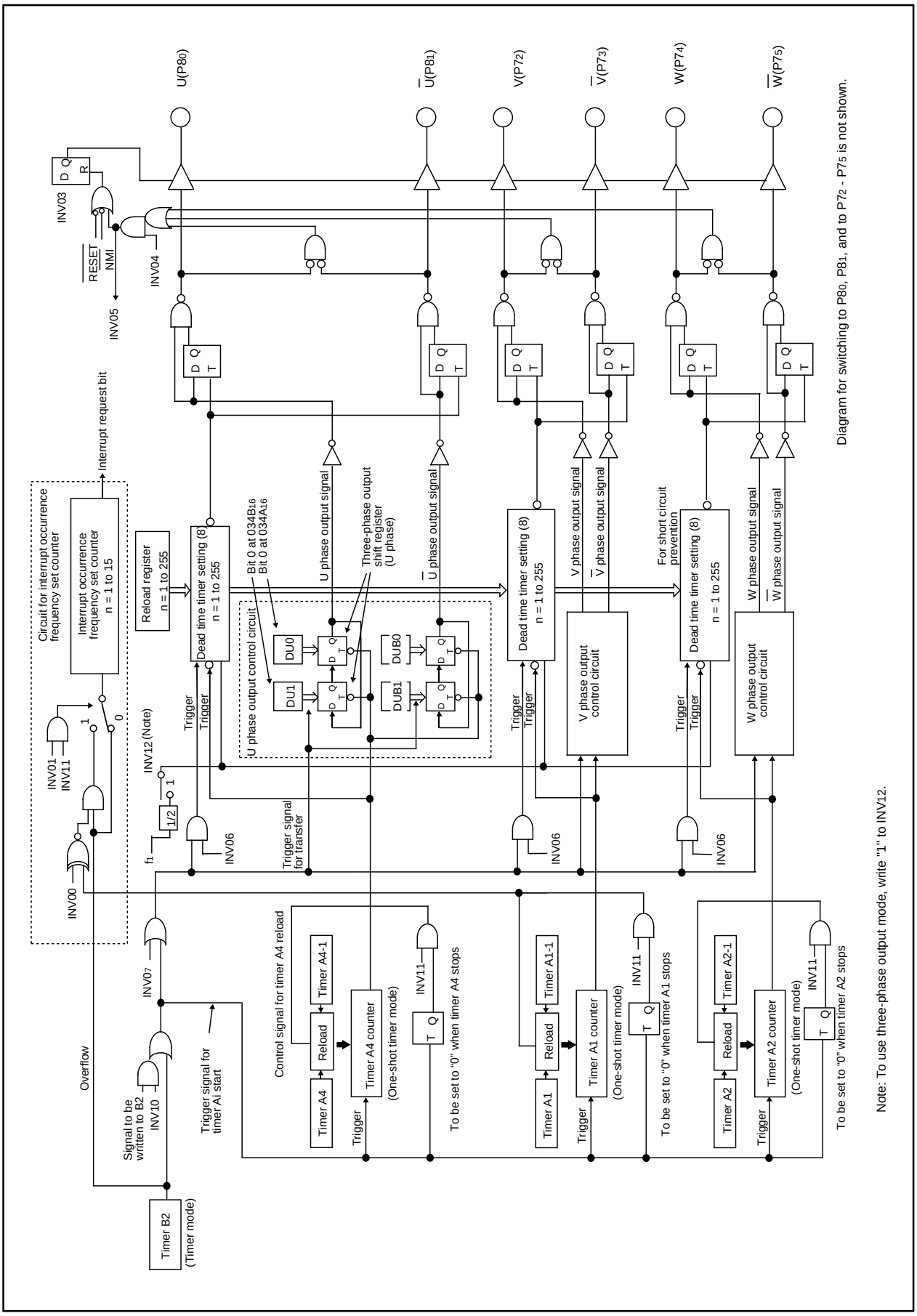


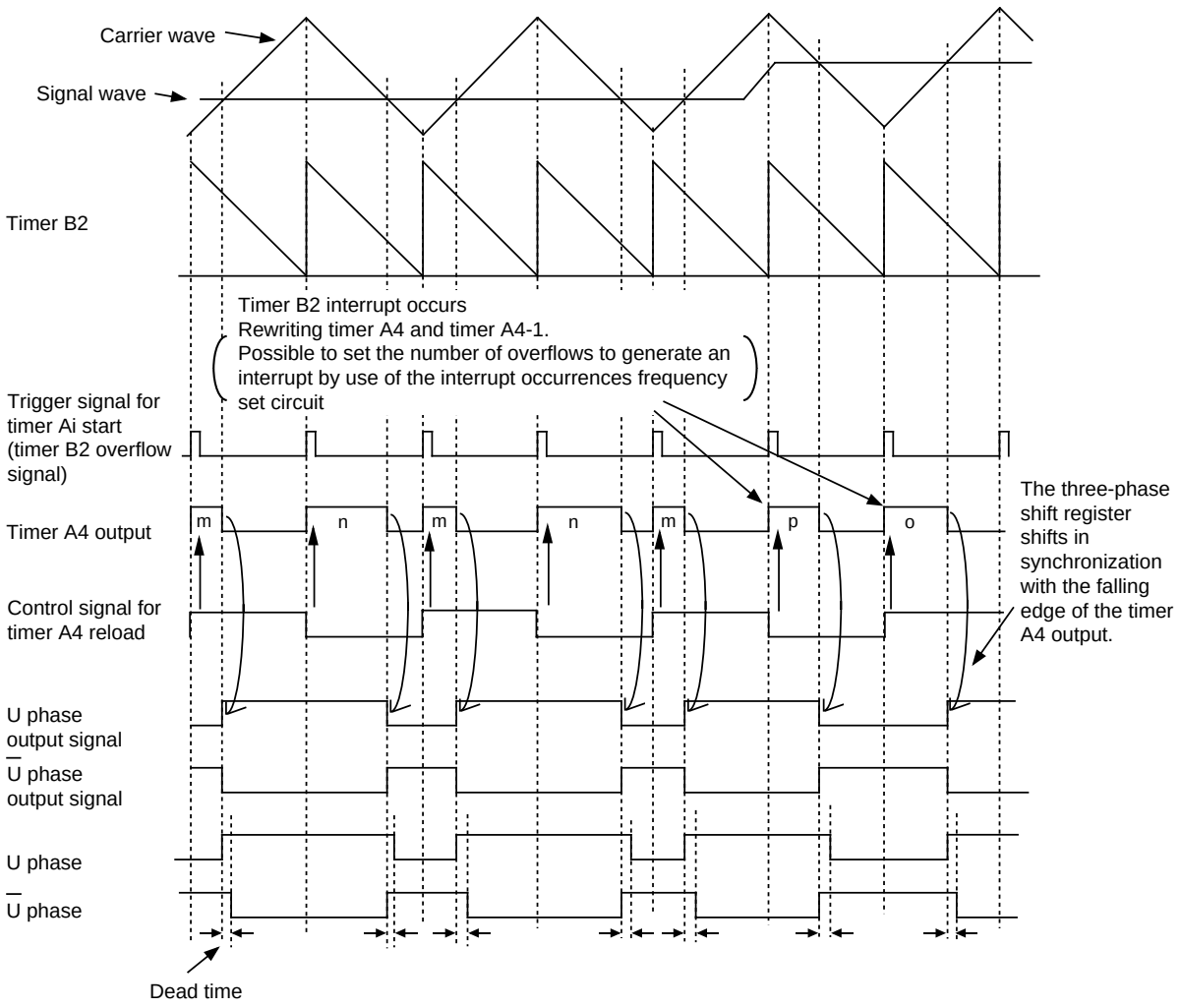

**Figure 1.15.5. Block diagram for three-phase PWM output mode**

### Triangular wave modulation

To generate a PWM waveform of triangular wave modulation, set "0" in the modulation mode select bit (bit 6 at $0348_{16}$). Also, set "1" in the timers A4-1, A1-1, A2-1 control bit (bit 1 at $0349_{16}$). In this mode, each of timers A4, A1, and A2 has two timer registers, and alternately reloads the timer register's content to the counter every time timer B2 counter's content becomes $0000_{16}$. If "0" is set to the effective interrupt output specification bit (bit 1 at $0348_{16}$), the frequency of interrupt requests that occur every time the timer B2 counter's value becomes $0000_{16}$ can be set by use of the timer B2 counter ($034D_{16}$) for setting the frequency of interrupt occurrences. The frequency of occurrences is given by (setting; setting $\neq$ 0).
Setting "1" in the effective interrupt output specification bit (bit 1 at $0348_{16}$) provides the means to choose which value of the timer A1 reload control signal to use, "0" or "1", to cause timer B2's interrupt request to occur. To make this selection, use the effective interrupt output polarity selection bit (bit 0 at $0348_{16}$).
An example of U phase waveform is shown in Figure 1.15.6, and the description of waveform output workings is given below. Set "1" in DU0 (bit 0 at $034A_{16}$). And set "0" in DUB0 (bit 1 at $034A_{16}$). In addition, set "0" in DU1 (bit 0 at $034B_{16}$) and set "1" in DUB1 (bit 1 at $034B_{16}$). Also, set "0" in the effective interrupt output specification bit (bit 1 at $0348_{16}$) to set a value in the timer B2 interrupt occurrence frequency set counter. By this setting, a timer B2 interrupt occurs when the timer B2 counter's content becomes $0000_{16}$ as many as (setting) times. Furthermore, set "1" in the effective interrupt output specification bit (bit 1 at $0348_{16}$), set "0" in the effective interrupt output polarity select bit (bit 0 at $0348_{16}$) and set "1" in the interrupt occurrence frequency set counter ($034D_{16}$). These settings cause a timer B2 interrupt to occur every other interval when the U phase output goes to "H".
When the timer B2 counter's content becomes $0000_{16}$, timer A4 starts outputting one-shot pulses. In this instance, the content of DU1 (bit 0 at $034B_{16}$) and that of DU0 (bit 0 at $034A_{16}$) are set in the three-phase output shift register (U phase), the content of DUB1 (bit 1 at $034B_{16}$) and that of DUB0 (bit 1 at $034A_{16}$) are set in the three-phase output shift register ($\overline{U}$ phase). After triangular wave modulation mode is selected, however, no setting is made in the shift register even though the timer B2 counter's content becomes $0000_{16}$.
The value of DU0 and that of DUB0 are output to the U terminal ($P8_0$) and to the $\overline{U}$ terminal ($P8_1$) respectively. When the timer A4 counter counts the value written to timer A4 ($038F_{16}$, $038E_{16}$) and when timer A4 finishes outputting one-shot pulses, the three-phase shift register's content is shifted one position, and the value of DU1 and that of DUB1 are output to the U phase output signal and to $\overline{U}$ phase output signal respectively. At this time, one-shot pulses are output from the timer for setting dead time used for setting the time over which the "L" level of the U phase waveform does not lap over the "L" level of the $\overline{U}$ phase waveform, which has the opposite phase of the former. The U phase waveform output that started from the "H" level keeps its level until the timer for setting dead time finishes outputting one-shot pulses even though the three-phase output shift register's content changes from "1" to "0" by the effect of the one-shot pulses. When the timer for setting dead time finishes outputting one-shot pulses, "0" already shifted in the three-phase shift register goes effective, and the U phase waveform changes to the "L" level. When the timer B2 counter's content becomes $0000_{16}$, the timer A4 counter starts counting the value written to timer A4-1 ($0347_{16}$, $0346_{16}$), and starts outputting one-shot pulses. When timer A4 finishes outputting one-shot pulses, the three-phase shift register's content is shifted one position, but if the three-phase output shift register's content changes from "0" to "1" as a result of the shift, the output level changes from "L" to "H" without waiting for the timer for setting dead time to finish outputting one-shot pulses. A U phase waveform is generated by these workings repeatedly. With the exception that the three-phase output shift register on the $\overline{U}$ phase side is used, the workings in generating a $\overline{U}$ phase waveform, which has the opposite phase of the U phase waveform, are the same as in generating a U

phase waveform. In this way, a waveform can be picked up from the applicable terminal in a manner in which the "L" level of the U phase waveform doesn't lap over that of the $\overline{U}$ phase waveform, which has the opposite phase of the U phase waveform. The width of the "L" level too can be adjusted by varying the values of timer B2, timer A4, and timer A4-1. In dealing with the V and W phases, and $\overline{V}$ and $\overline{W}$ phases, the latter are of opposite phase of the former, have the corresponding timers work similarly to dealing with the U and $\overline{U}$ phases to generate an intended waveform.

**A carrier wave of triangular waveform**




Note: Set to triangular wave modulation mode and to three-phase mode 1.

**Figure 1.15.6. Timing chart of operation (1)**

Assigning certain values to DU0 (bit 0 at 034A16) and DUB0 (bit 1 at 034A16), and to DU1 (bit 0 at 034B16) and DUB1 (bit 1 at 034B16) allows the user to output the waveforms as shown in Figure 1.15.7, that is, to output the U phase alone, to fix $\overline{U}$ phase to "H", to fix the U phase to "H," or to output the $\overline{U}$ phase alone.

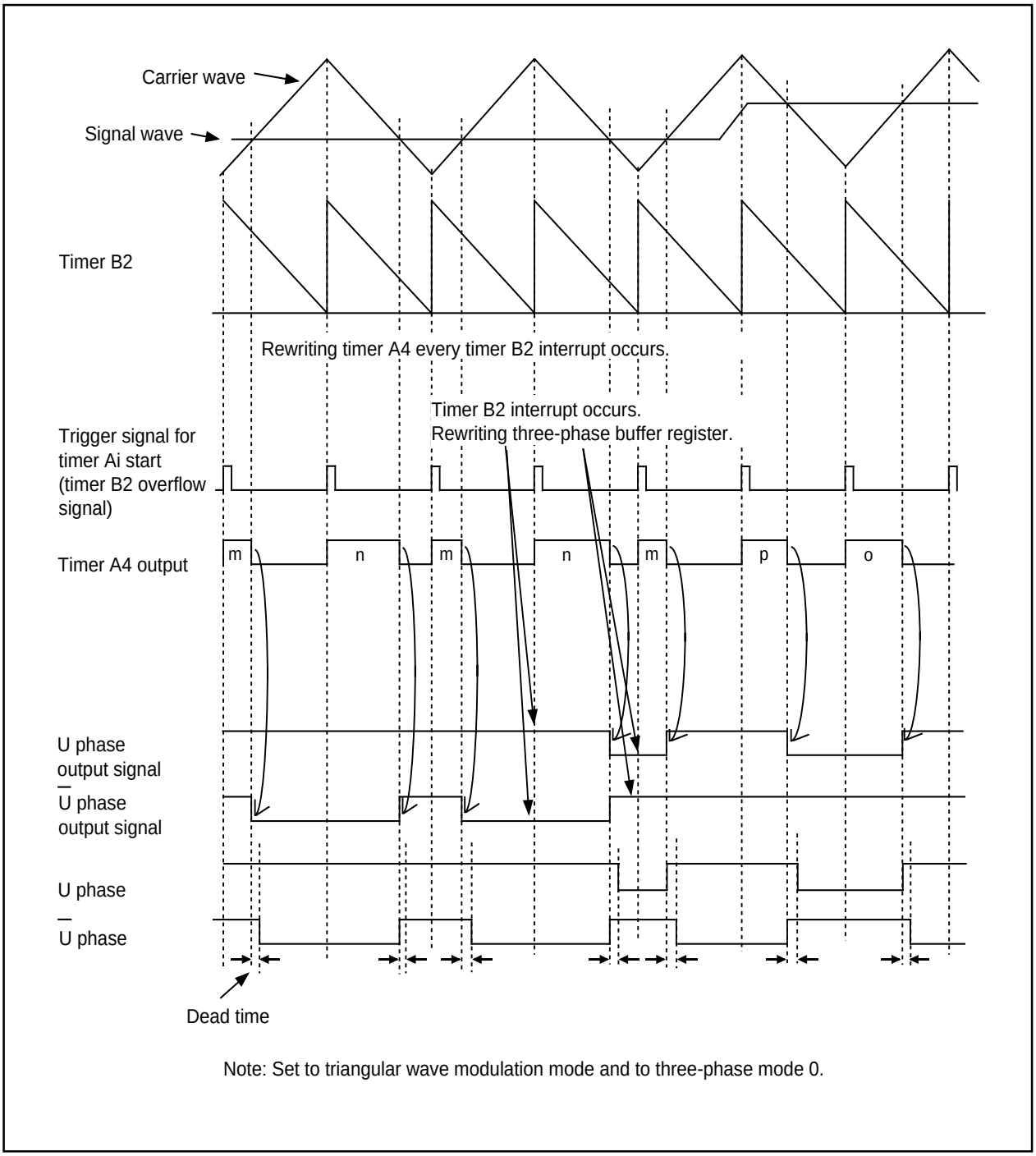



**Figure 1.15.7. Timing chart of operation (2)**

### Sawtooth modulation

To generate a PWM waveform of sawtooth wave modulation, set "1" in the modulation mode select bit (bit 6 at $0348_{16}$). Also, set "0" in the timers A4-1, A1-1, and A2-1 control bit (bit 1 at $0349_{16}$). In this mode, the timer registers of timers A4, A1, and A2 comprise conventional timers A4, A1, and A2 alone, and reload the corresponding timer register's content to the counter every time the timer B2 counter's content becomes $0000_{16}$. The effective interrupt output specification bit (bit 1 at $0348_{16}$) and the effective interrupt output polarity select bit (bit 0 at $0348_{16}$) go nullified.

An example of U phase waveform is shown in Figure 1.15.8, and the description of waveform output workings is given below. Set "1" in DU0 (bit 0 at $034A_{16}$), and set "0" in DUB0 (bit 1 at $034A_{16}$). In addition, set "0" in DU1 (bit 0 at $034A_{16}$) and set "1" in DUB1 (bit 1 at $034A_{16}$).

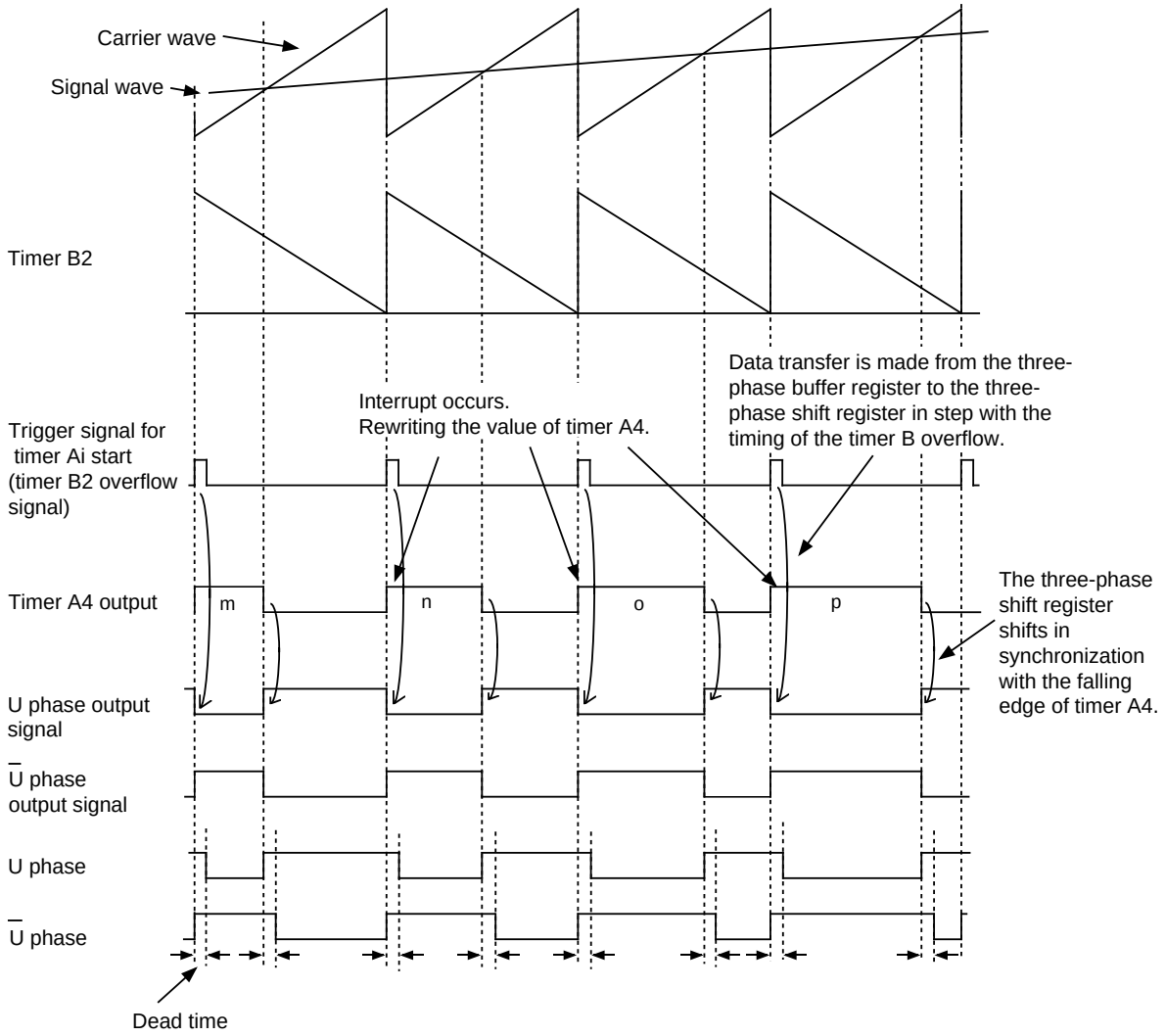
When the timber B2 counter's content becomes $0000_{16}$, timer B2 generates an interrupt, and timer A4 starts outputting one-shot pulses at the same time. In this instance, the contents of the three-phase buffer registers DU1 and DU0 are set in the three-phase output shift register (U phase), and the contents of DUB1 and DUB0 are set in the three-phase output shift register ($\overline{U}$ phase). After this, the three-phase buffer register's content is set in the three-phase shift register every time the timer B2 counter's content becomes $0000_{16}$.

The value of DU0 and that of DUB0 are output to the U terminal ($P8_0$) and to the $\overline{U}$ terminal ($P8_1$) respectively. When the timer A4 counter counts the value written to timer A4 ($038F_{16}$, $038E_{16}$) and when timer A4 finishes outputting one-shot pulses, the three-phase output shift register's content is shifted one position, and the value of DU1 and that of DUB1 are output to the U phase output signal and to the $\overline{U}$ output signal respectively. At this time, one-shot pulses are output from the timer for setting dead time used for setting the time over which the "L" level of the U phase waveform doesn't lap over the "L" level of the $\overline{U}$ phase waveform, which has the opposite phase of the former. The U phase waveform output that started from the "H" level keeps its level until the timer for setting dead time finishes outputting one-shot pulses even though the three-phase output shift register's content changes from "1" to "0 "by the effect of the one-shot pulses. When the timer for setting dead time finishes outputting one-shot pulses, 0 already shifted in the three-phase shift register goes effective, and the U phase waveform changes to the "L" level. When the timer B2 counter's content becomes $0000_{16}$, the contents of the three-phase buffer registers DU1 and DU0 are set in the three-phase output shift register (U phase), and the contents of DUB1 and DUB0 are set in the three-phase output shift register ($\overline{U}$ phase) again.

A U phase waveform is generated by these workings repeatedly. With the exception that the three-phase output shift register on the $\overline{U}$ phase side is used, the workings in generating a $\overline{U}$ phase waveform, which has the opposite phase of the U phase waveform, are the same as in generating a U phase waveform. In this way, a waveform can be picked up from the applicable terminal in a manner in which the "L" level of the U phase waveform doesn't lap over that of the $\overline{U}$ phase waveform, which has the opposite phase of the U phase waveform. The width of the "L" level too can be adjusted by varying the values of timer B2 and timer A4. In dealing with the V and W phases, and $\overline{V}$ and $\overline{W}$ phases, the latter are of opposite phase of the former, have the corresponding timers work similarly to dealing with the U and $\overline{U}$ phases to generate an intended waveform.

Setting "1" both in DUB0 and in DUB1 provides a means to output the U phase alone and to fix the $\overline{U}$ phase output to "H" as shown in Figure 1.15.9.

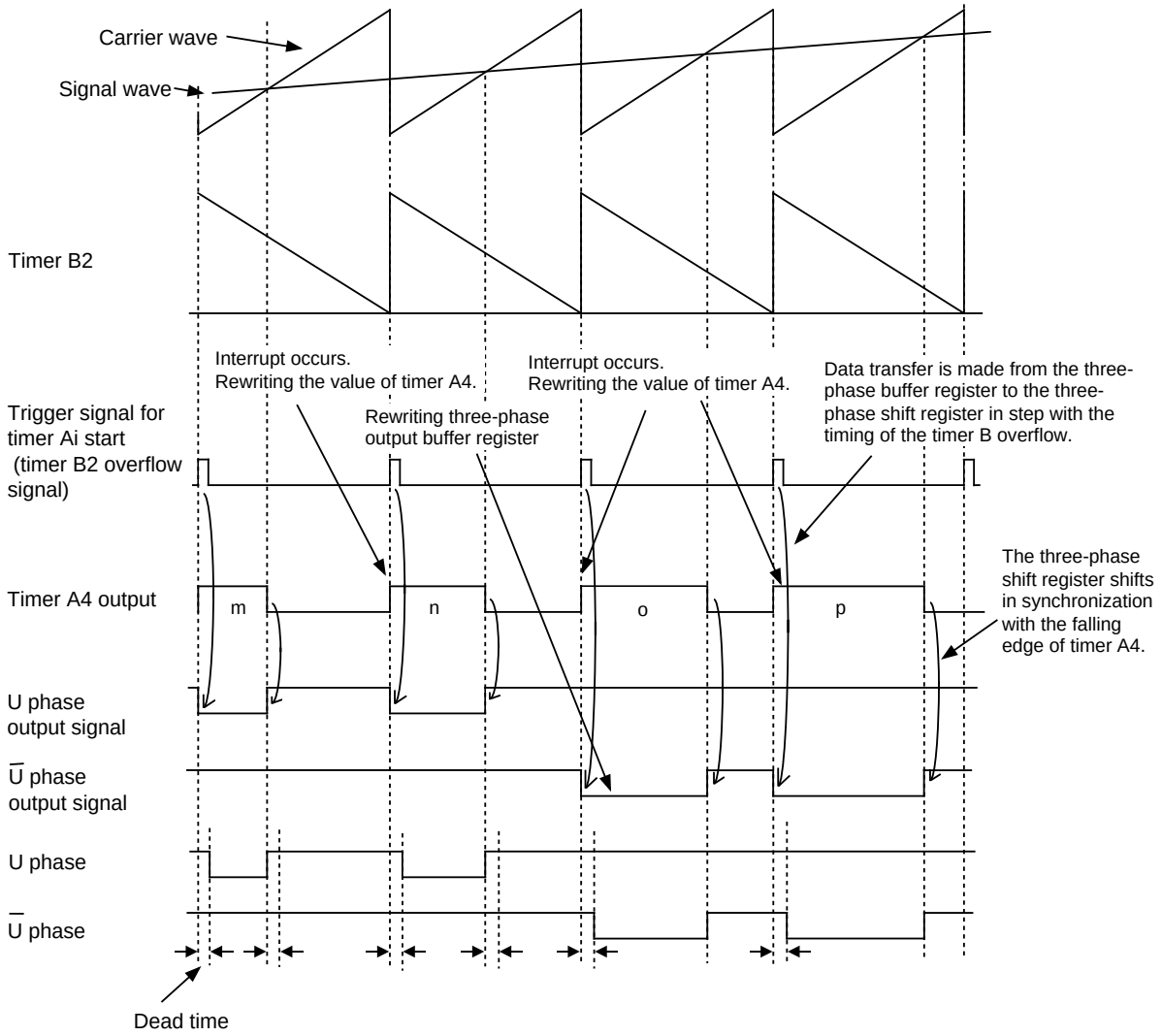
**A carrier wave of sawtooth waveform**




Note: Set to sawtooth modulation mode and to three-phase mode 0.

**Figure 1.15.8. Timing chart of operation (3)**

**A carrier wave of sawtooth waveform**




Note: Set to sawtooth modulation mode and to three-phase mode 0.

**Figure 1.15.9. Timing chart of operation (4)**

## 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 1.16.1 shows the block diagram of UART0, UART1 and UART2. Figures 1.16.2 and 1.16.3 show the block diagram of the transmit/receive unit.

UARTi (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 $03A0_{16}$, $03A8_{16}$ and $0378_{16}$) determine whether UARTi is used as a clock synchronous serial I/O or as a UART. Although a few functions are different, UART0, UART1 and UART2 have almost the same functions. 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 TxD pin and the RxD pin are different in level.

Table 1.16.1 shows the comparison of functions of UART0 through UART2, and Figures 1.16.4 to 1.16.9 show the registers related to UARTi.

Note: SIM : Subscriber Identity Module

**Table 1.16.1. Comparison of functions of UART0 through UART2**

| Function | UART0 | UART1 | UART2 |
|---|---|---|---|
| CLK polarity selection | Possible (Note 1) | Possible (Note 1) | Possible (Note 1) |
| 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 1) |
| 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, RxD I/O polarity switch | Impossible | Impossible | Possible |
| TxD, RxD port output format | CMOS output | CMOS output | N-channel open-drain output |
| 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.

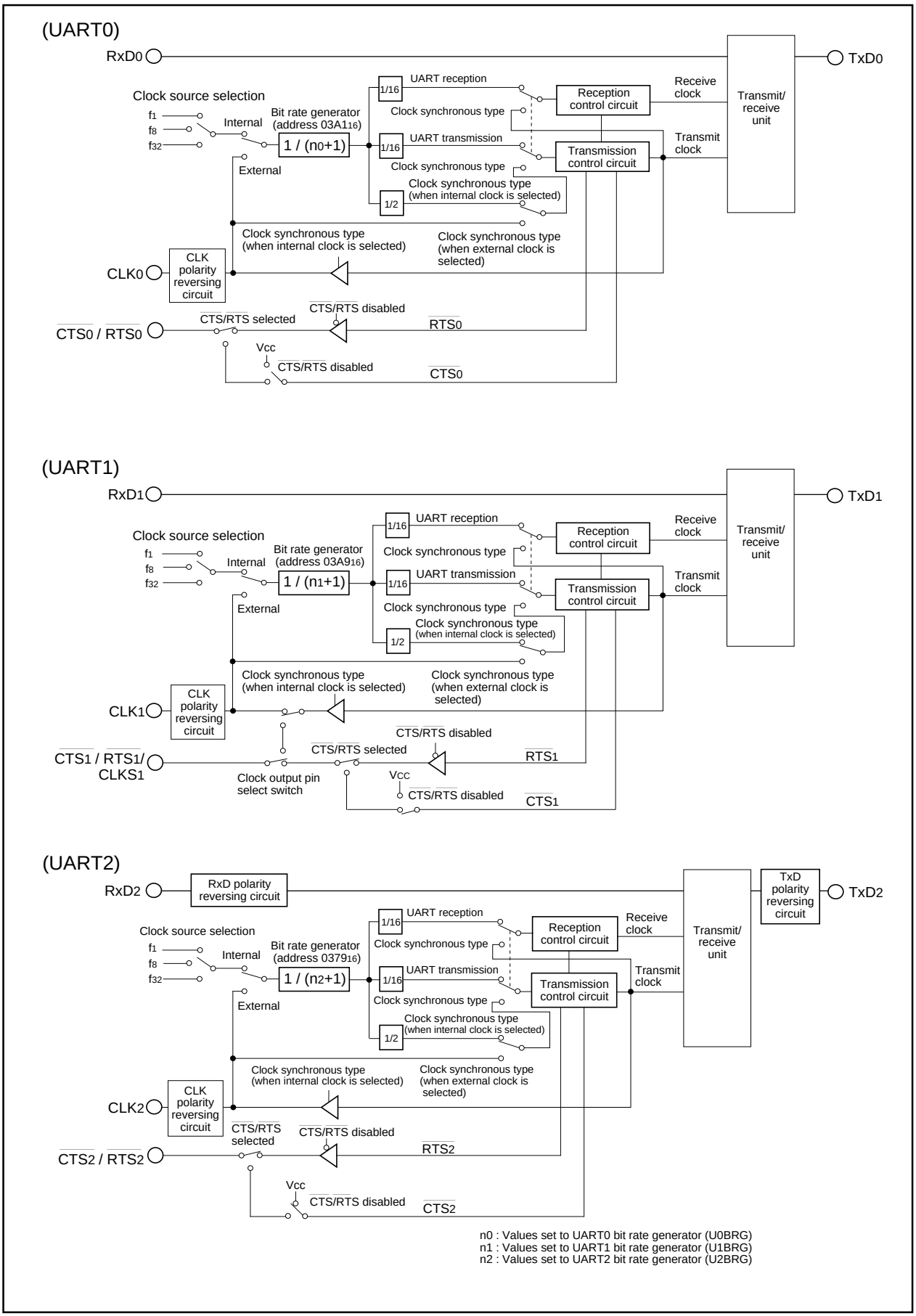



**Figure 1.16.1. Block diagram of UARTi (i = 0 to 2)**

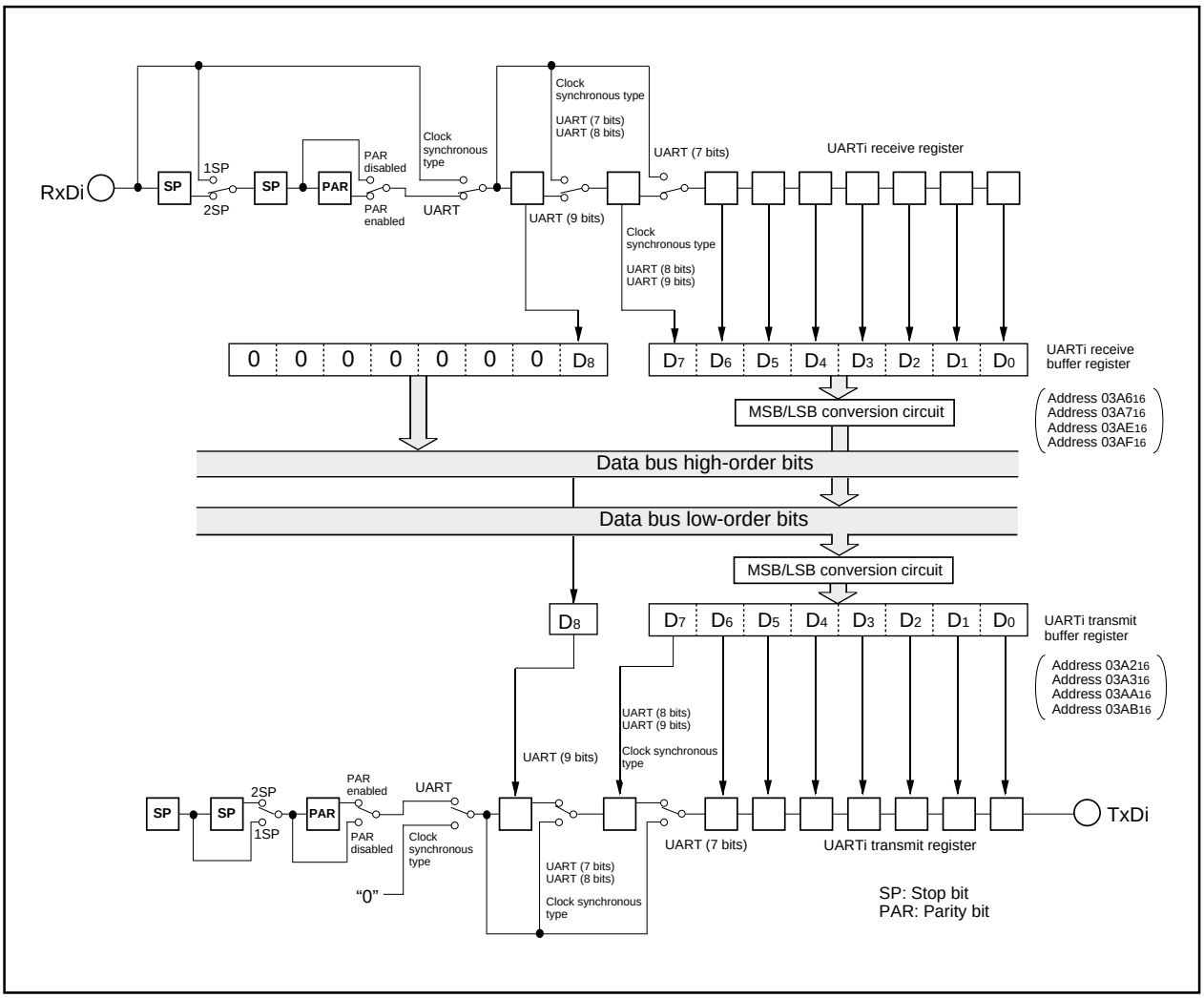



**Figure 1.16.2. Block diagram of UARTi (i = 0, 1) transmit/receive unit**

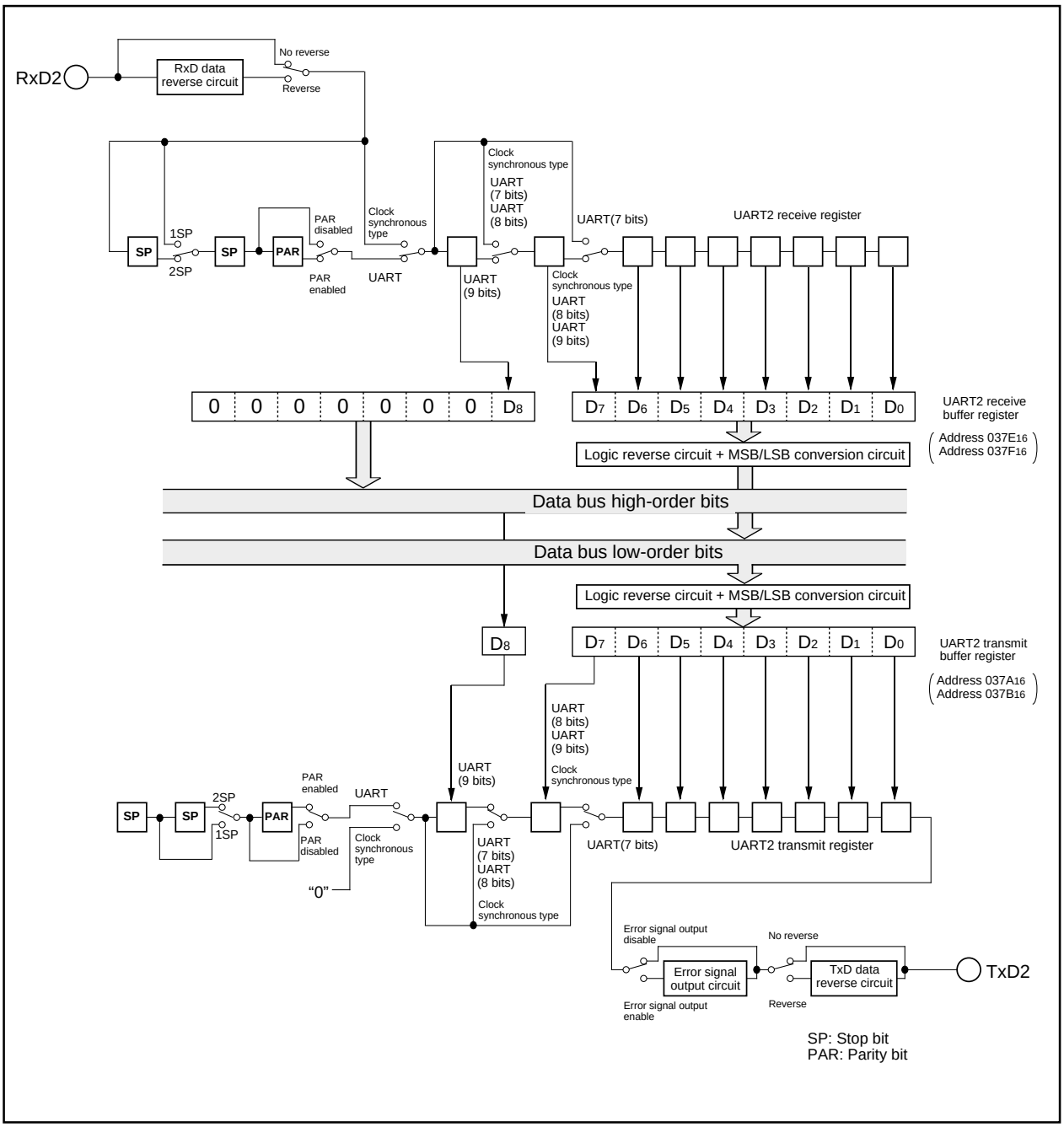
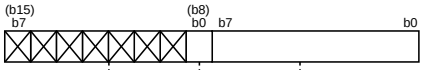



**Figure 1.16.3. Block diagram of UART2 transmit/receive unit**

UARTi transmit buffer register (Note)

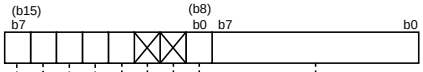



| Symbol | Address | When reset |
|---|---|---|
| U0TB | $03A3_{16}$, $03A2_{16}$ | Indeterminate |
| U1TB | $03AB_{16}$, $03AA_{16}$ | Indeterminate |
| U2TB | $037B_{16}$, $037A_{16}$ | Indeterminate |

| Function | R | W |
|---|---|---|
| Transmit data | × | ○ |
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be indeterminate. | — | — |

Note: Use MOV instruction to write to this register.

UARTi receive buffer register

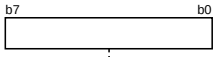



| Symbol | Address | When reset |
|---|---|---|
| U0RB | $03A7_{16}$, $03A6_{16}$ | Indeterminate |
| U1RB | $03AF_{16}$, $03AE_{16}$ | Indeterminate |
| U2RB | $037F_{16}$, $037E_{16}$ | Indeterminate |

| Bit symbol | Bit name | Function (During clock synchronous serial I/O mode) | Function (During UART mode) | R | W |
|---|---|---|---|---|---|
| — | ——— | Receive data | Receive data | ○ | × |
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be “0”. | | | | — | — |
| ABT | Arbitration lost detecting flag (Note 2) | 0 : Not detected<br>1 : Detected | Invalid | ○ | ○ |
| OER | Overrun error flag (Note 1) | 0 : No overrun error<br>1 : Overrun error found | 0 : No overrun error<br>1 : Overrun error found | ○ | × |
| FER | Framing error flag (Note 1) | Invalid | 0 : No framing error<br>1 : Framing error found | ○ | × |
| PER | Parity error flag (Note 1) | Invalid | 0 : No parity error<br>1 : Parity error found | ○ | × |
| SUM | Error sum flag (Note 1) | Invalid | 0 : No error<br>1 : Error found | ○ | × |

Note 1: Bits 15 through 12 are set to “0” when the serial I/O mode select bit (bits 2 to 0 at addresses $03A0_{16}$, $03A8_{16}$ and $0378_{16}$) are set to “$000_2$” or the receive enable bit is set to “0”.
(Bit 15 is set to “0” when bits 14 to 12 all are set to “0”.) Bits 14 and 13 are also set to “0” when the lower byte of the UARTi receive buffer register (addresses $03A6_{16}$, $03AE_{16}$ and $037E_{16}$) is read out.
Note 2: Arbitration lost detecting flag is allocated to U2RB and noting but “0” may be written. Nothing is assigned in bit 11 of U0RB and U1RB. When write, set “0”. The value, if read, turns out to be “0”.

UARTi bit rate generator (Note 1, 2)




| Symbol | Address | When reset |
|---|---|---|
| U0BRG | $03A1_{16}$ | Indeterminate |
| U1BRG | $03A9_{16}$ | Indeterminate |
| U2BRG | $0379_{16}$ | Indeterminate |

| Function | Values that can be set | R | W |
|---|---|---|---|
| Assuming that set value = n, BRGi divides the count source by n + 1 | $00_{16}$ to $FF_{16}$ | × | ○ |

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

**Figure 1.16.4. Serial I/O-related registers (1)**

UARTi transmit/receive mode register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| UiMR(i=0,1) | $03A0_{16}$, $03A8_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function (During clock synchronous serial I/O mode) | Function (During UART mode) | R | W |
|---|---|---|---|---|---|
| SMD0 | Serial I/O mode select bit | Must be fixed to 001 | b2 b1 b0<br>1 0 0 : Transfer data 7 bits long<br>1 0 1 : Transfer data 8 bits long<br>1 1 0 : Transfer data 9 bits long<br>0 0 0 : Serial I/O invalid<br>0 1 0 : Must not be set.<br>0 1 1 : Must not be set.<br>1 1 1 : Must not be set. | ○ | ○ |
| SMD1 | | b2 b1 b0<br>0 0 0 : Serial I/O invalid<br>0 1 0 : Must not be set.<br>0 1 1 : Must not be set.<br>1 1 1 : Must not be set. | | ○ | ○ |
| SMD2 | | | | ○ | ○ |
| CKDIR | Internal/external clock select bit | 0 : Internal clock<br>1 : External clock (Note) | 0 : Internal clock<br>1 : External clock (Note) | ○ | ○ |
| STPS | Stop bit length select bit | Invalid | 0 : One stop bit<br>1 : Two stop bits | ○ | ○ |
| PRY | Odd/even parity select bit | Invalid | Valid when bit 6 = “1”<br>0 : Odd parity<br>1 : Even parity | ○ | ○ |
| PRYE | Parity enable bit | Invalid | 0 : Parity disabled<br>1 : Parity enabled | ○ | ○ |
| SLEP | Sleep select bit | Must always be “0” | 0 : Sleep mode deselected<br>1 : Sleep mode selected | ○ | ○ |

Note : Set the corresponding port direction register to “0”.

UART2 transmit/receive mode register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| U2MR | $0378_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function (During clock synchronous serial I/O mode) | Function (During UART mode) | R | W |
|---|---|---|---|---|---|
| SMD0 | Serial I/O mode select bit | Must be fixed to 001 | b2 b1 b0<br>1 0 0 : Transfer data 7 bits long<br>1 0 1 : Transfer data 8 bits long<br>1 1 0 : Transfer data 9 bits long<br>0 0 0 : Serial I/O invalid<br>0 1 0 : Must not be set.<br>0 1 1 : Must not be set.<br>1 1 1 : Must not be set. | ○ | ○ |
| SMD1 | | b2 b1 b0<br>0 0 0 : Serial I/O invalid<br>0 1 0 : (Note 1)<br>0 1 1 : Must not be set.<br>1 1 1 : Must not be set. | | ○ | ○ |
| SMD2 | | | | ○ | ○ |
| CKDIR | Internal/external clock select bit | 0 : Internal clock<br>1 : External clock (Note 2) | Must always be “0” | ○ | ○ |
| STPS | Stop bit length select bit | Invalid | 0 : One stop bit<br>1 : Two stop bits | ○ | ○ |
| PRY | Odd/even parity select bit | Invalid | Valid when bit 6 = “1”<br>0 : Odd parity<br>1 : Even parity | ○ | ○ |
| PRYE | Parity enable bit | Invalid | 0 : Parity disabled<br>1 : Parity enabled | ○ | ○ |
| IOPOL | TxD, RxD I/O polarity reverse bit | 0 : No reverse<br>1 : Reverse<br>Usually set to “0” | 0 : No reverse<br>1 : Reverse<br>Usually set to “0” | ○ | ○ |

Note 1: Bit 2 to bit 0 are set to “$010_2$” when $I^2C$ mode is used.
Note 2: Set the corresponding port direction register to “0”.

**Figure 1.16.5. Serial I/O-related registers (2)**

UARTi transmit/receive control register 0

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| UiC0(i=0,1) | $03A4_{16}$, $03AC_{16}$ | $08_{16}$ |

| Bit symbol | Bit name | Function (During clock synchronous serial I/O mode) | Function (During UART mode) | R | W |
|---|---|---|---|---|---|
| CLK0 | BRG count source select bit | b1 b0<br>0 0 : $f_1$ is selected<br>0 1 : $f_8$ is selected<br>1 0 : $f_{32}$ is selected<br>1 1 : Must not be set. | b1 b0<br>0 0 : $f_1$ is selected<br>0 1 : $f_8$ is selected<br>1 0 : $f_{32}$ is selected<br>1 1 : Must not be set. | ○ | ○ |
| CLK1 | | | | ○ | ○ |
| CRS | $\overline{CTS}/\overline{RTS}$ function select bit | Valid when bit 4 = "0"<br>0 : $\overline{CTS}$ function is selected (Note 1)<br>1 : $\overline{RTS}$ function is selected (Note 2) | Valid when bit 4 = "0"<br>0 : $\overline{CTS}$ function is selected (Note 1)<br>1 : $\overline{RTS}$ function is selected (Note 2) | ○ | ○ |
| TXEPT | Transmit register empty flag | 0 : Data present in transmit register (during transmission)<br>1 : No data present in transmit register (transmission completed) | 0 : Data present in transmit register (during transmission)<br>1 : No data present in transmit register (transmission completed) | ○ | × |
| CRD | $\overline{CTS}/\overline{RTS}$ disable bit | 0 : $\overline{CTS}/\overline{RTS}$ function enabled<br>1 : $\overline{CTS}/\overline{RTS}$ function disabled ($P6_0$ and $P6_4$ function as programmable I/O port) | 0 : $\overline{CTS}/\overline{RTS}$ function enabled<br>1 : $\overline{CTS}/\overline{RTS}$ function disabled ($P6_0$ and $P6_4$ function as programmable I/O port) | ○ | ○ |
| NCH | Data output select bit | 0 : TxDi pin is CMOS output<br>1 : TxDi pin is N-channel open-drain output | 0: TxDi pin is CMOS output<br>1: TxDi pin is N-channel open-drain output | ○ | ○ |
| CKPOL | CLK polarity select bit | 0 : Transmit data is output at falling edge of transfer clock and receive data is input at rising edge<br>1 : Transmit data is output at rising edge of transfer clock and receive data is input at falling edge | Must always be "0" | ○ | ○ |
| UFORM | Transfer format select bit | 0 : LSB first<br>1 : MSB first | Must always be "0" | ○ | ○ |

Note 1: Set the corresponding port direction register to "0".
Note 2: The settings of the corresponding port register and port direction register are invalid.

UART2 transmit/receive control register 0

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | X | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| U2C0 | $037C_{16}$ | $08_{16}$ |

| Bit symbol | Bit name | Function (During clock synchronous serial I/O mode) | Function (During UART mode) | R | W |
|---|---|---|---|---|---|
| CLK0 | BRG count source select bit | b1 b0<br>0 0 : $f_1$ is selected<br>0 1 : $f_8$ is selected<br>1 0 : $f_{32}$ is selected<br>1 1 : Must not be set. | b1 b0<br>0 0 : $f_1$ is selected<br>0 1 : $f_8$ is selected<br>1 0 : $f_{32}$ is selected<br>1 1 : Must not be set. | ○ | ○ |
| CLK1 | | | | ○ | ○ |
| CRS | $\overline{CTS}/\overline{RTS}$ function select bit | Valid when bit 4 = "0"<br>0 : $\overline{CTS}$ function is selected (Note 1)<br>1 : $\overline{RTS}$ function is selected (Note 2) | Valid when bit 4 = "0"<br>0 : $\overline{CTS}$ function is selected (Note 1)<br>1 : $\overline{RTS}$ function is selected (Note 2) | ○ | ○ |
| TXEPT | Transmit register empty flag | 0 : Data present in transmit register (during transmission)<br>1 : No data present in transmit register (transmission completed) | 0 : Data present in transmit register (during transmission)<br>1 : No data present in transmit register (transmission completed) | ○ | × |
| CRD | $\overline{CTS}/\overline{RTS}$ disable bit | 0 : $\overline{CTS}/\overline{RTS}$ function enabled<br>1 : $\overline{CTS}/\overline{RTS}$ function disabled ($P7_3$ functions programmable I/O port) | 0 : $\overline{CTS}/\overline{RTS}$ function enabled<br>1 : $\overline{CTS}/\overline{RTS}$ function disabled ($P7_3$ functions programmable I/O port) | ○ | ○ |
| Nothing is assigned.<br>In an attempt to write to this bit, write "0". The value, if read, turns out to be "0". | | | | — | — |
| CKPOL | CLK polarity select bit | 0 : Transmit data is output at falling edge of transfer clock and receive data is input at rising edge<br>1 : Transmit data is output at rising edge of transfer clock and receive data is input at falling edge | Must always be "0" | ○ | ○ |
| UFORM | Transfer format select bit (Note 3) | 0 : LSB first<br>1 : MSB first | 0 : LSB first<br>1 : MSB first | ○ | ○ |

Note 1: Set the corresponding port direction register to "0".
Note 2: The settings of the corresponding port register and port direction register are invalid.
Note 3: Only clock synchronous serial I/O mode and 8-bit UART mode are valid.

**Figure 1.16.6. Serial I/O-related registers (3)**

UARTi transmit/receive control register 1

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| X | X | X | X | | | | |

| Symbol | Address | When reset |
|---|---|---|
| UiC1(i=0,1) | $03A5_{16}$,$03AD_{16}$ | $02_{16}$ |

| Bit symbol | Bit name | Function (During clock synchronous serial I/O mode) | Function (During UART mode) | R | W |
|---|---|---|---|---|---|
| TE | Transmit enable bit | 0 : Transmission disabled<br>1 : Transmission enabled | 0 : Transmission disabled<br>1 : Transmission enabled | ○ | ○ |
| TI | Transmit buffer empty flag | 0 : Data present in transmit buffer register<br>1 : No data present in transmit buffer register | 0 : Data present in transmit buffer register<br>1 : No data present in transmit buffer register | ○ | × |
| RE | Receive enable bit | 0 : Reception disabled<br>1 : Reception enabled | 0 : Reception disabled<br>1 : Reception enabled | ○ | ○ |
| RI | Receive complete flag | 0 : No data present in receive buffer register<br>1 : Data present in receive buffer register | 0 : No data present in receive buffer register<br>1 : Data present in receive buffer register | ○ | × |
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be “0”. | | | | – | – |

UART2 transmit/receive control register 1

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| U2C1 | $037D_{16}$ | $02_{16}$ |

| Bit symbol | Bit name | Function (During clock synchronous serial I/O mode) | Function (During UART mode) | R | W |
|---|---|---|---|---|---|
| TE | Transmit enable bit | 0 : Transmission disabled<br>1 : Transmission enabled | 0 : Transmission disabled<br>1 : Transmission enabled | ○ | ○ |
| TI | Transmit buffer empty flag | 0 : Data present in transmit buffer register<br>1 : No data present in transmit buffer register | 0 : Data present in transmit buffer register<br>1 : No data present in transmit buffer register | ○ | × |
| RE | Receive enable bit | 0 : Reception disabled<br>1 : Reception enabled | 0 : Reception disabled<br>1 : Reception enabled | ○ | ○ |
| RI | Receive complete flag | 0 : No data present in receive buffer register<br>1 : Data present in receive buffer register | 0 : No data present in receive buffer register<br>1 : Data present in receive buffer register | ○ | × |
| U2IRS | UART2 transmit interrupt cause select bit | 0 : Transmit buffer empty (TI = 1)<br>1 : Transmit is completed (TXEPT = 1) | 0 : Transmit buffer empty (TI = 1)<br>1 : Transmit is completed (TXEPT = 1) | ○ | ○ |
| U2RRM | UART2 continuous receive mode enable bit | 0 : Continuous receive mode disabled<br>1 : Continuous receive mode enabled | Must always be "0" | ○ | ○ |
| U2LCH | Data logic select bit | 0 : No reverse<br>1 : Reverse | 0 : No reverse<br>1 : Reverse | ○ | ○ |
| U2ERE | Error signal output enable bit | Must be fixed to “0” | 0 : Output disabled<br>1 : Output enabled | ○ | ○ |

**Figure 1.16.7. Serial I/O-related registers (4)**

UART transmit/receive control register 2

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| X | 0 | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| UCON | $03B0_{16}$ | $X0000000_2$ |

| Bit symbol | Bit name | Function (During clock synchronous serial I/O mode) | Function (During UART mode) | R | W |
|---|---|---|---|---|---|
| U0IRS | UART0 transmit interrupt cause select bit | 0 : Transmit buffer empty (TI = 1)<br>1 : Transmission completed (TXEPT = 1) | 0 : Transmit buffer empty (TI = 1)<br>1 : Transmission completed (TXEPT = 1) | ○ | ○ |
| U1IRS | UART1 transmit interrupt cause select bit | 0 : Transmit buffer empty (TI = 1)<br>1 : Transmission completed (TXEPT = 1) | 0 : Transmit buffer empty (TI = 1)<br>1 : Transmission completed (TXEPT = 1) | ○ | ○ |
| U0RRM | UART0 continuous receive mode enable bit | 0 : Continuous receive mode disabled<br>1 : Continuous receive mode enable | Must always be “0” | ○ | ○ |
| U1RRM | UART1 continuous receive mode enable bit | 0 : Continuous receive mode disabled<br>1 : Continuous receive mode enabled | Must always be “0” | ○ | ○ |
| CLKMD0 | CLK/CLKS select bit 0 | Valid when bit 5 = “1”<br>0 : Clock output to CLK1<br>1 : Clock output to CLKS1 | Invalid | ○ | ○ |
| CLKMD1 | CLK/CLKS select bit 1 (Note) | 0 : Normal mode (CLK output is CLK1 only)<br>1 : Transfer clock output from multiple pins function selected | Must always be “0” | ○ | ○ |
| Reserved bit | | Must always be set to “0” | | ○ | ○ |
| Nothing is assigned.<br>In an attempt to write to this bit, write “0”. The value, if read, turns out to be indeterminate. | | | | — | — |

Note: When using multiple pins to output the transfer clock, the following requirements must be met:
• UART1 internal/external clock select bit (bit 3 at address $03A8_{16}$) = “0”.

UART2 special mode register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| U2SMR | $0377_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function (During clock synchronous serial I/O mode) | Function (During UART mode) | R | W |
|---|---|---|---|---|---|
| IICM | $I^2C$ mode select bit | 0 : Normal mode<br>1 : $I^2C$ mode | Must always be “0” | ○ | ○ |
| ABC | Arbitration lost detecting flag control bit | 0 : Update per bit<br>1 : Update per byte | Must always be “0” | ○ | ○ |
| BBS | Bus busy flag | 0 : STOP condition detected<br>1 : START condition detected | Must always be “0” | ○ (Note1) | ○ |
| LSYN | SCLL sync output enable bit | 0 : Disabled<br>1 : Enabled | Must always be “0” | ○ | ○ |
| ABSCS | Bus collision detect sampling clock select bit | Must always be “0” | 0 : Rising edge of transfer clock<br>1 : Underflow signal of timer A0 | ○ | ○ |
| ACSE | Auto clear function select bit of transmit enable bit | Must always be “0” | 0 : No auto clear function<br>1 : Auto clear at occurrence of bus collision | ○ | ○ |
| SSS | Transmit start condition select bit | Must always be “0” | 0 : Ordinary<br>1 : Falling edge of $RxD_2$ | ○ | ○ |
| SDDS | SDA digital delay select bit (Note 2, Note 3) | 0 : Analog delay output is selected<br>1 : Digital delay output is selected<br>(must always be “0” when not using $I^2C$ mode) | Must always be “0” | ○ | ○ |

Note 1: Nothing but “0” may be written.
Note 2: When not in $I^2C$ mode, do not set this bit by writing a “1”. During normal mode, fix it to “0”. When this bit = “0”, UART2 special mode register 3 (U2SMR3 at address $0375_{16}$) bits 7 to 5 (DL2 to DL0 = SDA digital delay setup bits) are initialized to “000”, with the analog delay circuit selected. Also, when SDDS = “0”, the U2SMR3 register cannot be read or written to.
Note 3: When analog delay is selected, only the analog delay value is effective; when digital delay is selected, only the digital delay value is effective.

**Figure 1.16.8. Serial I/O-related registers (5)**

UART2 special mode register 2 ($I^2C$ bus exclusive use register)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| U2SMR2 | $037_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function ($I^2C$ bus exclusive use) | R | W |
|---|---|---|---|---|
| IICM2 | $I^2C$ mode select bit 2 | Refer to Table 1.17.11 | ○ | ○ |
| CSC | Clock-synchronous bit | 0 : Disabled<br>1 : Enabled | ○ | ○ |
| SWC | SCL wait output bit | 0 : Disabled<br>1 : Enabled | ○ | ○ |
| ALS | SDA output stop bit | 0 : Disabled<br>1 : Enabled | ○ | ○ |
| STAC | UART2 initialization bit | 0 : Disabled<br>1 : Enabled | ○ | ○ |
| SWC2 | SCL wait output bit 2 | 0: UART2 clock<br>1: 0 output | ○ | ○ |
| SDHI | SDA output disable bit | 0: Enabled<br>1: Disabled (high impedance) | ○ | ○ |
| SHTC | Start/stop condition control bit | Set this bit to “1” in $I^2C$ mode<br>(refer to Table 1.17.12) | ○ | ○ |

UART2 special mode register 3 ($I^2C$ bus exclusive use register)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| U2SMR3 | $037_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function ($I^2C$ bus exclusive use register) | R | W |
|---|---|---|---|---|
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be indeterminate. However, when SDDS = “1”, the value “0” is read out (Note 1) | | | — | — |
| DL0 | SDA digital delay setup bit<br>(Note 1, Note 2, Note 3) | b7 b6 b5<br>0 0 0 : Must not be set when using $I^2C$ mode<br>0 0 1 : 1 to 2 cycle(s) of 1/f($X_{IN}$)<br>0 1 0 : 2 to 3 cycles of 1/f($X_{IN}$)<br>0 1 1 : 3 to 4 cycles of 1/f($X_{IN}$)<br>1 0 0 : 4 to 5 cycles of 1/f($X_{IN}$)<br>1 0 1 : 5 to 6 cycles of 1/f($X_{IN}$)<br>1 1 0 : 6 to 7 cycles of 1/f($X_{IN}$)<br>1 1 1 : 7 to 8 cycles of 1/f($X_{IN}$)<br>} Digital delay is selected | ○ | ○ |
| DL1 | | | ○ | ○ |
| DL2 | | | ○ | ○ |

Note 1: This bit can be read or written to when UART2 special mode register (U2SMR at address $037_{16}$) bit 7 (SDDS: SDA digital delay select bit) = “1”. When the initial value of UART2 special mode register 3 (U2SMR3) is read after setting SDDS = “1”, the value is “$00_{16}$”. When writing to UART2 special mode register 3 (U2SMR3) after setting SDDS = “1”, be sure to write 0's to bits 0–4. When SDDS = “0”, this register cannot be written to; when read, the value is indeterminate.

Note 2: These bits are initialized to “000” when SDDS = “0”. After a reset, these bits are set to “000”. However, because these bits can be read only when SDDS = “1”, the value read from these bits when SDDS = “0” is indeterminate.

Note 3: The amount of delay varies with the load on SCL and SDA pins. Also, when using an external clock, the amount of delay increases by about 200 ns, so be sure to take this into account when using the device.

**Figure 1.16.9. Serial I/O-related registers (6)**

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

The clock synchronous serial I/O mode uses a transfer clock to transmit and receive data. Tables 1.16.2 and 1.16.3 list the specifications of the clock synchronous serial I/O mode. Figure 1.16.10 shows the UARTi transmit/receive mode register.

**Table 1.16.2. Specifications of clock synchronous serial I/O mode (1)**

| Item | Specification |
|---|---|
| Transfer data format | • Transfer data length: 8 bits |
| Transfer clock | • When internal clock is selected (bit 3 at addresses $03A0_{16}$, $03A8_{16}$, $0378_{16}$ = "0") : fi/ 2(n+1) (Note 1) fi = $f_1$, $f_8$, $f_{32}$<br>• When external clock is selected (bit 3 at addresses $03A0_{16}$, $03A8_{16}$, $0378_{16}$ = "1") : Input from CLKi pin |
| Transmission/reception control | • $\overline{CTS}$ function, $\overline{RTS}$ function, $\overline{CTS}$ and $\overline{RTS}$ function invalid: selectable |
| Transmission start condition | • To start transmission, the following requirements must be met:<br>– Transmit enable bit (bit 0 at addresses $03A5_{16}$, $03AD_{16}$, $037D_{16}$) = "1"<br>– Transmit buffer empty flag (bit 1 at addresses $03A5_{16}$, $03AD_{16}$, $037D_{16}$) = "0"<br>– When $\overline{CTS}$ function selected, $\overline{CTS}$ input level = "L"<br>• Furthermore, if external clock is selected, the following requirements must also be met:<br>– CLKi polarity select bit (bit 6 at addresses $03A4_{16}$, $03AC_{16}$, $037C_{16}$) = "0": CLKi input level = "H"<br>– CLKi polarity select bit (bit 6 at addresses $03A4_{16}$, $03AC_{16}$, $037C_{16}$) = "1": CLKi input level = "L" |
| Reception start condition | • To start reception, the following requirements must be met:<br>– Receive enable bit (bit 2 at addresses $03A5_{16}$, $03AD_{16}$, $037D_{16}$) = "1"<br>– Transmit enable bit (bit 0 at addresses $03A5_{16}$, $03AD_{16}$, $037D_{16}$) = "1"<br>– Transmit buffer empty flag (bit 1 at addresses $03A5_{16}$, $03AD_{16}$, $037D_{16}$) = "0"<br>• Furthermore, if external clock is selected, the following requirements must also be met:<br>– CLKi polarity select bit (bit 6 at addresses $03A4_{16}$, $03AC_{16}$, $037C_{16}$) = "0": CLKi input level = "H"<br>– CLKi polarity select bit (bit 6 at addresses $03A4_{16}$, $03AC_{16}$, $037C_{16}$) = "1": CLKi input level = "L" |
| Interrupt request generation timing | • When transmitting<br>– Transmit interrupt cause select bit (bits 0, 1 at address $03B0_{16}$, bit 4 at address $037D_{16}$) = "0": Interrupts requested when data transfer from UARTi transfer buffer register to UARTi transmit register is completed<br>– Transmit interrupt cause select bit (bits 0, 1 at address $03B0_{16}$, bit 4 at address $037D_{16}$) = "1": Interrupts requested when data transmission from UARTi transfer register is completed<br>• When receiving<br>– Interrupts requested when data transfer from UARTi receive register to UARTi receive buffer register is completed |
| Error detection | • Overrun error (Note 2)<br>This error occurs when the next data is ready before contents of UARTi receive buffer register are read out |

Note 1: "n" denotes the value $00_{16}$ to $FF_{16}$ 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.

**Table 1.16.3. Specifications of clock synchronous serial I/O mode (2)**

| Item | Specification |
|---|---|
| Select function | • CLK polarity selection<br>Whether transmit data is output/input timing at the rising edge or falling edge of the transfer clock can be selected<br>• LSB first/MSB first selection<br>Whether transmission/reception begins with bit 0 or bit 7 can be selected<br>• Continuous receive mode selection<br>Reception is enabled simultaneously by a read from the receive buffer register<br>• Transfer clock output from multiple pins selection (UART1)<br>UART1 transfer clock can be chosen by software to be output from one of the two pins set<br>• Switching serial data logic (UART2)<br>Whether to reverse data in writing to the transmission buffer register or reading the reception buffer register can be selected.<br>• TxD, RxD I/O polarity reverse (UART2)<br>This function is reversing TxD port output and RxD port input. All I/O data level is reversed. |

UARTi transmit/receive mode registers

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| 0 | | | | | 0 | 0 | 1 |

| Symbol | Address | When reset |
|---|---|---|
| UiMR(i=0,1) | $03A0_{16}$, $03A8_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| SMD0 | Serial I/O mode select bit | b2 b1 b0<br>0 0 1 : Clock synchronous serial I/O mode | ○ | ○ |
| SMD1 | | | ○ | ○ |
| SMD2 | | | ○ | ○ |
| CKDIR | Internal/external clock select bit | 0 : Internal clock<br>1 : External clock (Note) | ○ | ○ |
| STPS | Invalid in clock synchronous serial I/O mode | | ○ | ○ |
| PRY | | | ○ | ○ |
| PRYE | | | ○ | ○ |
| SLEP | 0 (Must always be “0” in clock synchronous serial I/O mode) | | ○ | ○ |

Note : Set the corresponding port direction register to “0”.

UART2 transmit/receive mode register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| 0 | | | | | 0 | 0 | 1 |

| Symbol | Address | When reset |
|---|---|---|
| U2MR | $0378_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| SMD0 | Serial I/O mode select bit | b2 b1 b0<br>0 0 1 : Clock synchronous serial I/O mode | ○ | ○ |
| SMD1 | | | ○ | ○ |
| SMD2 | | | ○ | ○ |
| CKDIR | Internal/external clock select bit | 0 : Internal clock<br>1 : External clock (Note 2) | ○ | ○ |
| STPS | Invalid in clock synchronous serial I/O mode | | ○ | ○ |
| PRY | | | ○ | ○ |
| PRYE | | | ○ | ○ |
| IOPOL | TxD, RxD I/O polarity reverse bit (Note 1) | 0 : No reverse<br>1 : Reverse | ○ | ○ |

Note 1: Usually set to “0”.
Note 2: Set the corresponding port direction register to “0”.

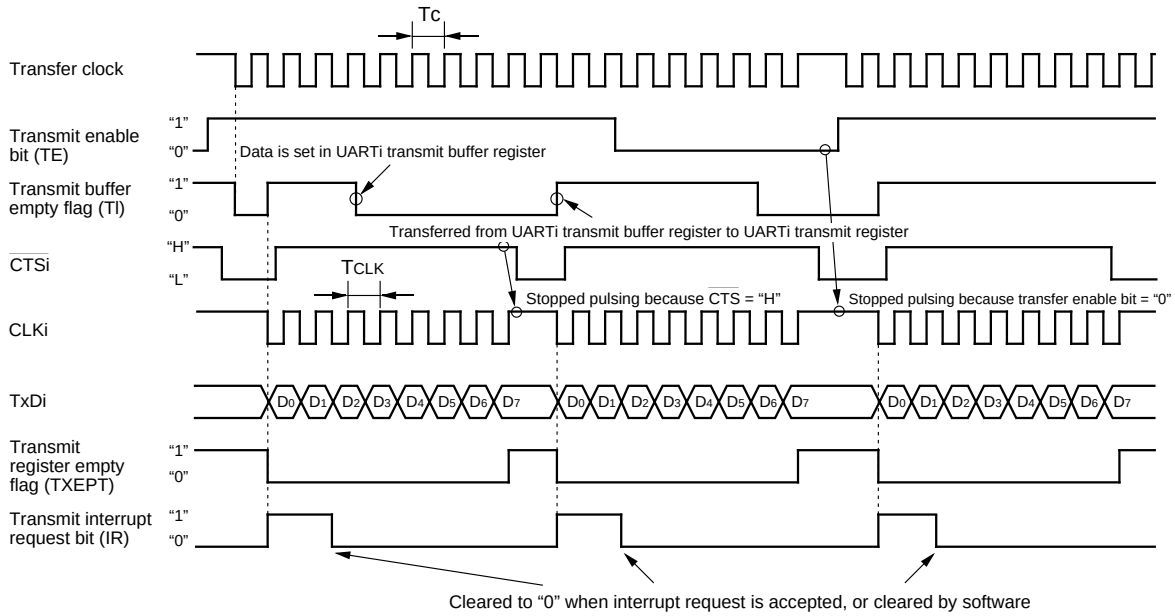
**Figure 1.16.10. UARTi transmit/receive mode register in clock synchronous serial I/O mode**

Table 1.16.4 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 function is not selected. 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 1.16.4. 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 ($P6_3$, $P6_7$, $P7_0$) | Serial data output | (Outputs dummy data when performing reception only) |
| RxDi ($P6_2$, $P6_6$, $P7_1$) | Serial data input | Port $P6_2$, $P6_6$ and $P7_1$ direction register (bits 2 and 6 at address $03EE_{16}$, bit 1 at address $03EF_{16}$)= “0”<br>(Can be used as an input port when performing transmission only) |
| CLKi ($P6_1$, $P6_5$, $P7_2$) | Transfer clock output | Internal/external clock select bit (bit 3 at address $03A0_{16}$, $03A8_{16}$, $0378_{16}$) = “0” |
| | Transfer clock input | Internal/external clock select bit (bit 3 at address $03A0_{16}$, $03A8_{16}$, $0378_{16}$) = “1”<br>Port $P6_1$, $P6_5$ and $P7_2$ direction register (bits 1 and 5 at address $03EE_{16}$, bit 2 at address $03EF_{16}$) = “0” |
| $\overline{CTS}$i/$\overline{RTS}$i ($P6_0$, $P6_4$, $P7_3$) | $\overline{CTS}$ input | $\overline{CTS}$/$\overline{RTS}$ disable bit (bit 4 at address $03A4_{16}$, $03AC_{16}$, $037C_{16}$) =“0”<br>$\overline{CTS}$/$\overline{RTS}$ function select bit (bit 2 at address $03A4_{16}$, $03AC_{16}$, $037C_{16}$) = “0”<br>Port $P6_0$, $P6_4$ and $P7_3$ direction register (bits 0 and 4 at address $03EE_{16}$, bit 3 at address $03EF_{16}$) = “0” |
| | $\overline{RTS}$ output | $\overline{CTS}$/$\overline{RTS}$ disable bit (bit 4 at address $03A4_{16}$, $03AC_{16}$, $037C_{16}$) = “0”<br>$\overline{CTS}$/$\overline{RTS}$ function select bit (bit 2 at address $03A4_{16}$, $03AC_{16}$, $037C_{16}$) = “1” |
| | Programmable I/O port | $\overline{CTS}$/$\overline{RTS}$ disable bit (bit 4 at address $03A4_{16}$, $03AC_{16}$, $037C_{16}$) = “1” |

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

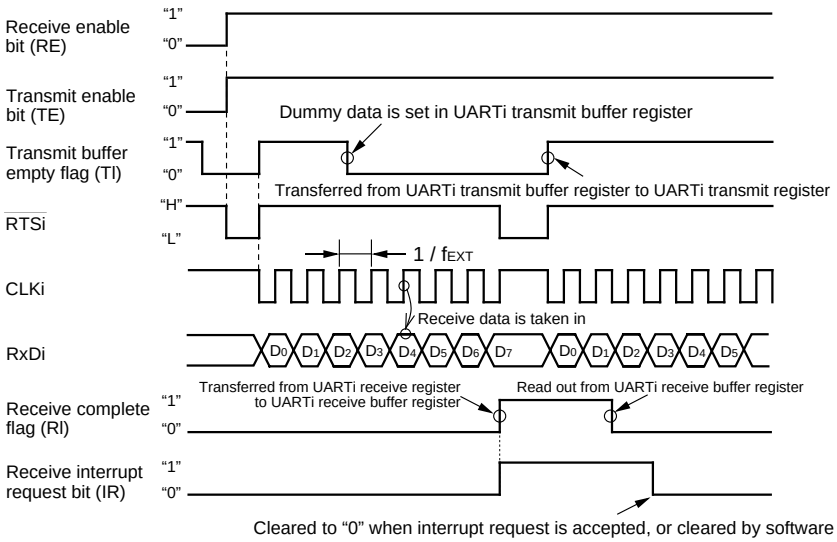



Shown in ( ) are bit symbols.

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".

Tc = TCLK = 2(n + 1) / fi
fi: frequency of BRGi count source (f1, f8, f32)
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 1.16.11. Typical transmit/receive timings in clock synchronous serial I/O mode**

**(a) Polarity select function**

As shown in Figure 1.16.12, the CLK polarity select bit (bit 6 at addresses 03A4$_{16}$, 03AC$_{16}$, 037C$_{16}$) allows selection of the polarity of the transfer clock.

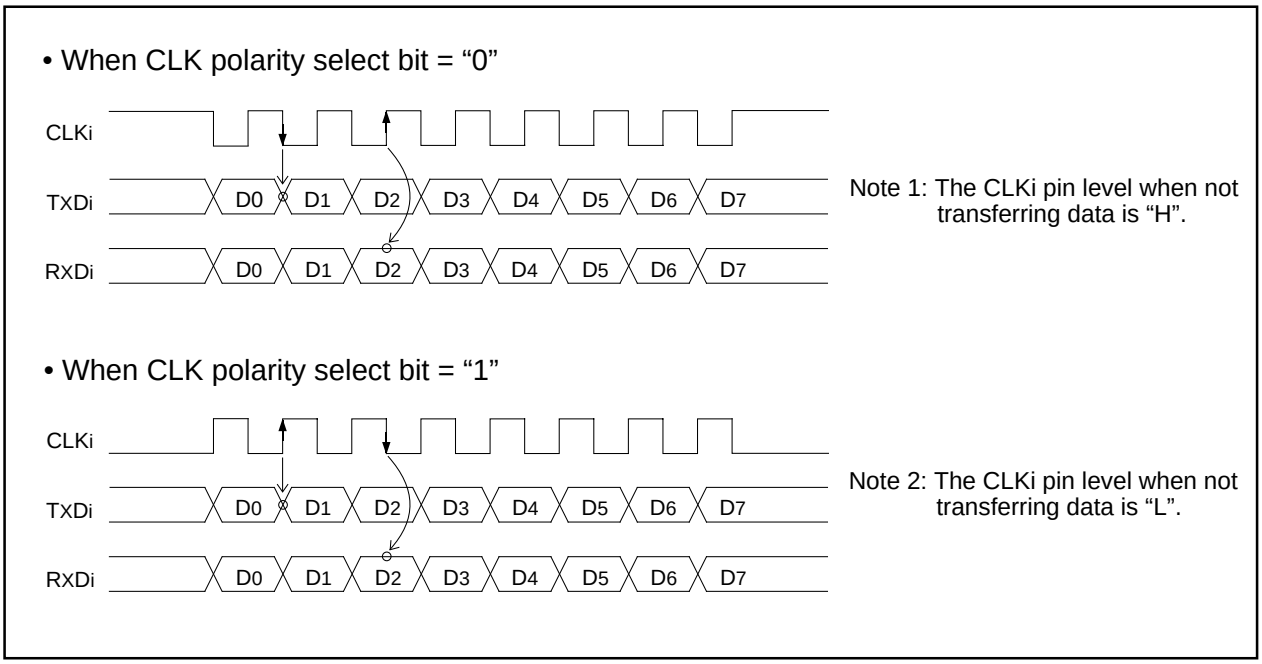



**Figure 1.16.12. Polarity of transfer clock**

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

As shown in Figure 1.16.13, when the transfer format select bit (bit 7 at addresses 03A4$_{16}$, 03AC$_{16}$, 037C$_{16}$) = “0”, the transfer format is “LSB first”; when the bit = “1”, the transfer format is “MSB first”.

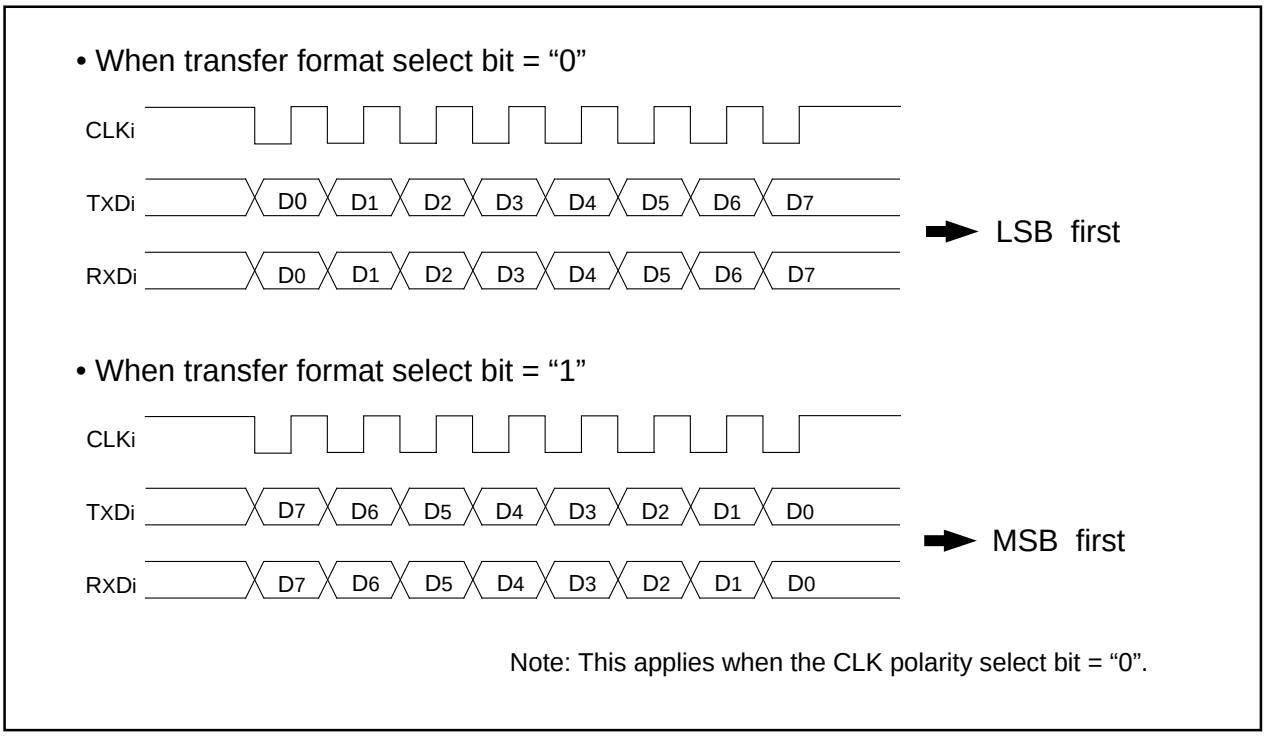



**Figure 1.16.13. Transfer format**

**(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 $03B0_{16}$). (See Figure 1.16.14.) The multiple pins function is valid only when the internal clock is selected for UART1.

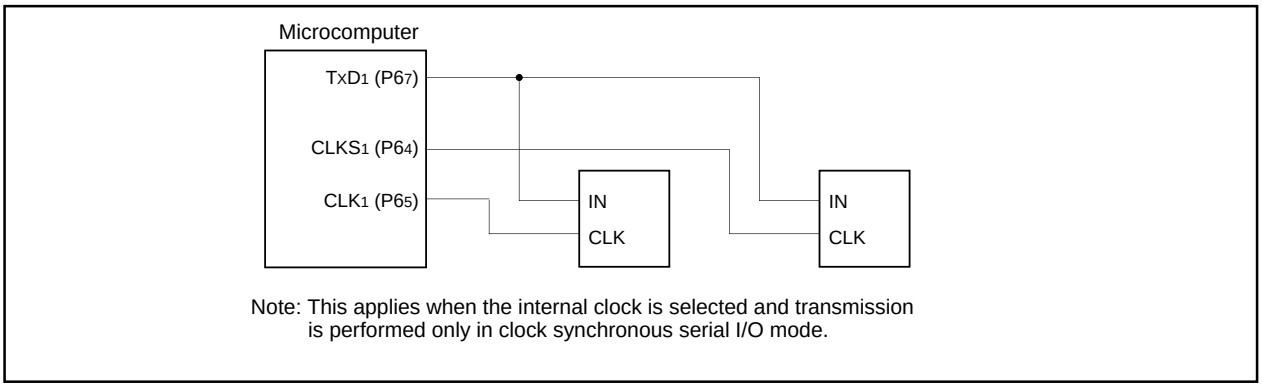



**Figure 1.16.14. The transfer clock output from the multiple pins function usage**

**(d) Continuous receive mode**

If the continuous receive mode enable bit (bits 2 and 3 at address $03B0_{16}$, bit 5 at address $037D_{16}$) 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.

**(e) Serial data logic switch function (UART2)**

When the data logic select bit (bit6 at address $037D_{16}$) = "1", and writing to transmit buffer register or reading from receive buffer register, data is reversed. Figure 1.16.15 shows the example of serial data logic switch timing.

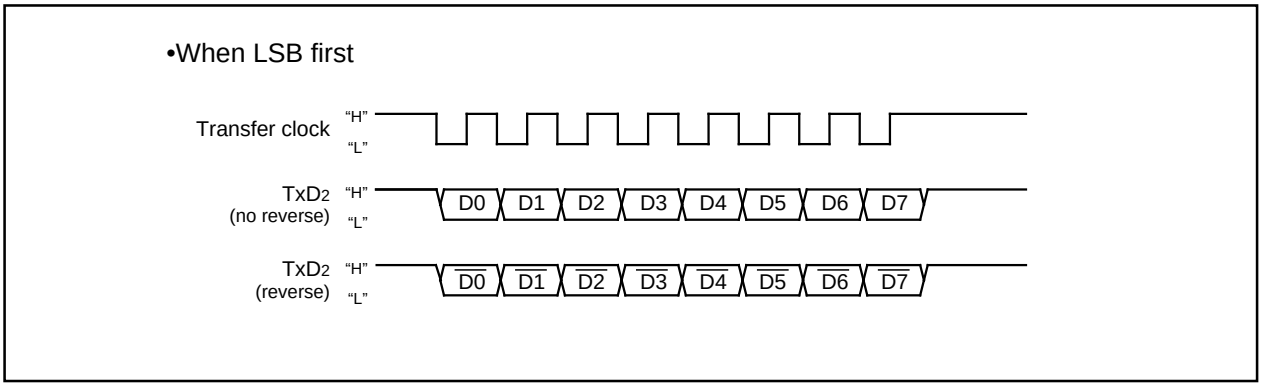



**Figure 1.16.15. Serial data logic switch timing**

## (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 1.16.5 and 1.16.6 list the specifications of the UART mode. Figure 1.16.16 shows the UARTi transmit/receive mode register.

**Table 1.16.5. Specifications of UART Mode (1)**

| Item | Specification |
|---|---|
| Transfer data format | • Character bit (transfer data): 7 bits, 8 bits, or 9 bits as selected<br>• Start bit: 1 bit<br>• Parity bit: Odd, even, or nothing as selected<br>• Stop bit: 1 bit or 2 bits as selected |
| Transfer clock | • When internal clock is selected (bit 3 at addresses $03A0_{16}$, $03A8_{16}$, $0378_{16}$ = “0”) :<br>fi/16(n+1) (Note 1) fi = $f_1$, $f_8$, $f_{32}$<br>• When external clock is selected (bit 3 at addresses $03A0_{16}$, $03A8_{16}$ =“1”) :<br>$f_{EXT}$/16(n+1) (Note 1) (Note 2) (Do not set external clock for UART2) |
| Transmission/reception control | • $\overline{CTS}$ function, $\overline{RTS}$ function, $\overline{CTS}$ and $\overline{RTS}$ function invalid: selectable |
| Transmission start condition | • To start transmission, the following requirements must be met:<br>- Transmit enable bit (bit 0 at addresses $03A5_{16}$, $03AD_{16}$, $037D_{16}$) = “1”<br>- Transmit buffer empty flag (bit 1 at addresses $03A5_{16}$, $03AD_{16}$, $037D_{16}$) = “0”<br>- When $\overline{CTS}$ function selected, $\overline{CTS}$ input level = “L” |
| Reception start condition | • To start reception, the following requirements must be met:<br>- Receive enable bit (bit 2 at addresses $03A5_{16}$, $03AD_{16}$, $037D_{16}$) = “1”<br>- Start bit detection |
| Interrupt request generation timing | • When transmitting<br>- Transmit interrupt cause select bits (bits 0,1 at address $03B0_{16}$, bit4 at address $037D_{16}$) = “0”: Interrupts requested when data transfer from UARTi transfer buffer register to UARTi transmit register is completed<br>- Transmit interrupt cause select bits (bits 0, 1 at address $03B0_{16}$, bit4 at address $037D_{16}$) = “1”: Interrupts requested when data transmission from UARTi transfer register is completed<br>• When receiving<br>- Interrupts requested when data transfer from UARTi receive register to UARTi receive buffer register is completed |
| Error detection | • Overrun error (Note 3)<br>This error occurs when the next data is ready before contents of UARTi receive buffer register are read out<br>• Framing error<br>This error occurs when the number of stop bits set is not detected<br>• Parity error<br>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<br>• Error sum flag<br>This flag is set (= 1) when any of the overrun, framing, and parity errors is encountered |

Note 1: ‘n’ denotes the value $00_{16}$ to $FF_{16}$ that is set to the UARTi bit rate generator.

Note 2: $f_{EXT}$ is input from the CLKi pin.

Note 3: If an overrun error occurs, the UARTi receive buffer will have the next data written in. Note also that the UARTi receive interrupt request bit does not change.

**Table 1.16.6. Specifications of UART Mode (2)**

| Item | Specification |
|---|---|
| Select function | • Sleep mode selection (UART0, UART1)<br>This mode is used to transfer data to and from one of multiple slave micro-computers<br>• Serial data logic switch (UART2)<br>This function is reversing logic value of transferring data. Start bit, parity bit and stop bit are not reversed.<br>• TxD, RxD I/O polarity switch (UART2)<br>This function is reversing TxD port output and RxD port input. All I/O data level is reversed. |

UARTi transmit / receive mode registers

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| UiMR(i=0,1) | $03A0_{16}$, $03A8_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| SMD0 | Serial I/O mode select bit | b2 b1 b0<br>1 0 0 : Transfer data 7 bits long<br>1 0 1 : Transfer data 8 bits long<br>1 1 0 : Transfer data 9 bits long | ○ | ○ |
| SMD1 | | | ○ | ○ |
| SMD2 | | | ○ | ○ |
| CKDIR | Internal / external clock select bit | 0 : Internal clock<br>1 : External clock (Note) | ○ | ○ |
| STPS | Stop bit length select bit | 0 : One stop bit<br>1 : Two stop bits | ○ | ○ |
| PRY | Odd / even parity select bit | Valid when bit 6 = “1”<br>0 : Odd parity<br>1 : Even parity | ○ | ○ |
| PRYE | Parity enable bit | 0 : Parity disabled<br>1 : Parity enabled | ○ | ○ |
| SLEP | Sleep select bit | 0 : Sleep mode deselected<br>1 : Sleep mode selected | ○ | ○ |

Note : Set the corresponding port direction register to “0”.

UART2 transmit / receive mode register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| U2MR | $0378_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| SMD0 | Serial I/O mode select bit | b2 b1 b0<br>1 0 0 : Transfer data 7 bits long<br>1 0 1 : Transfer data 8 bits long<br>1 1 0 : Transfer data 9 bits long | ○ | ○ |
| SMD1 | | | ○ | ○ |
| SMD2 | | | ○ | ○ |
| CKDIR | Internal / external clock select bit | Must always be “0” | ○ | ○ |
| STPS | Stop bit length select bit | 0 : One stop bit<br>1 : Two stop bits | ○ | ○ |
| PRY | Odd / even parity select bit | Valid when bit 6 = “1”<br>0 : Odd parity<br>1 : Even parity | ○ | ○ |
| PRYE | Parity enable bit | 0 : Parity disabled<br>1 : Parity enabled | ○ | ○ |
| IOPOL | TxD, RxD I/O polarity reverse bit (Note) | 0 : No reverse<br>1 : Reverse | ○ | ○ |

Note: Usually set to “0”.

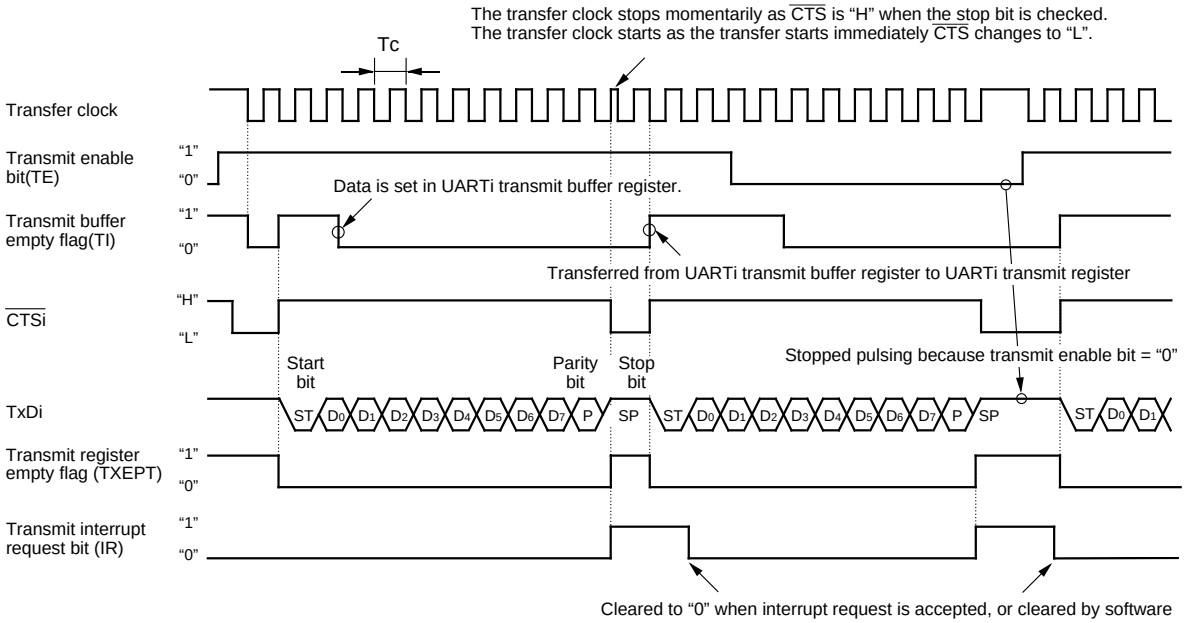
**Figure 1.16.16. UARTi transmit/receive mode register in UART mode**

Table 1.16.7 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 1.16.7. Input/output pin functions in UART mode**

| Pin name | Function | Method of selection |
|---|---|---|
| TxDi ($P6_3$, $P6_7$, $P7_0$) | Serial data output | |
| RxDi ($P6_2$, $P6_6$, $P7_1$) | Serial data input | Port $P6_2$, $P6_6$ and $P7_1$ direction register (bits 2 and 6 at address $03EE_{16}$, bit 1 at address $03EF_{16}$)= “0”<br>(Can be used as an input port when performing transmission only) |
| CLKi ($P6_1$, $P6_5$, $P7_2$) | Programmable I/O port | Internal/external clock select bit (bit 3 at address $03A0_{16}$, $03A8_{16}$, $0378_{16}$) = “0” |
| | Transfer clock input | Internal/external clock select bit (bit 3 at address $03A0_{16}$, $03A8_{16}$) = “1”<br>Port $P6_1$, $P6_5$ direction register (bits 1 and 5 at address $03EE_{16}$) = “0”<br>(Do not set external clock for UART2) |
| $\overline{CTS}$i/$\overline{RTS}$i ($P6_0$, $P6_4$, $P7_3$) | $\overline{CTS}$ input | $\overline{CTS}$/$\overline{RTS}$ disable bit (bit 4 at address $03A4_{16}$, $03AC_{16}$, $037C_{16}$) =“0”<br>$\overline{CTS}$/$\overline{RTS}$ function select bit (bit 2 at address $03A4_{16}$, $03AC_{16}$, $037C_{16}$) = “0”<br>Port $P6_0$, $P6_4$ and $P7_3$ direction register (bits 0 and 4 at address $03EE_{16}$, bit 3 at address $03EF_{16}$) = “0” |
| | $\overline{RTS}$ output | $\overline{CTS}$/$\overline{RTS}$ disable bit (bit 4 at address $03A4_{16}$, $03AC_{16}$, $037C_{16}$) = “0”<br>$\overline{CTS}$/$\overline{RTS}$ function select bit (bit 2 at address $03A4_{16}$, $03AC_{16}$, $037C_{16}$) = “1” |
| | Programmable I/O port | $\overline{CTS}$/$\overline{RTS}$ disable bit (bit 4 at address $03A4_{16}$, $03AC_{16}$, $037C_{16}$) = “1” |

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

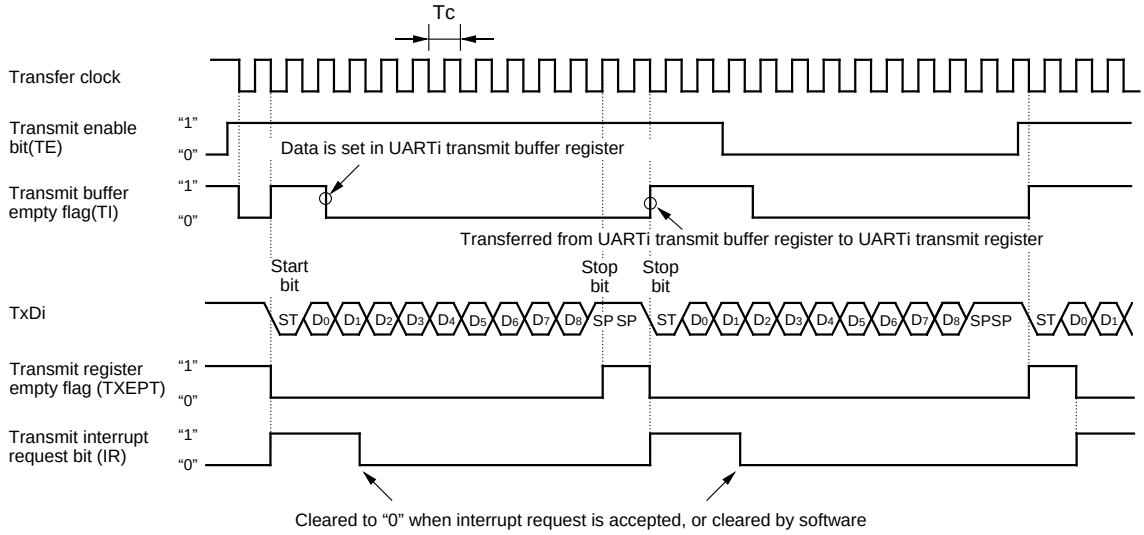



Shown in ( ) are bit symbols.

The above timing applies to the following settings :
- Parity is enabled.
- One stop bit.
- CTS function is selected.
- Transmit interrupt cause select bit = "1".

Tc = 16 (n + 1) / fi or 16 (n + 1) / $f_{EXT}$
fi : frequency of BRGi count source ($f_1$, $f_8$, $f_{32}$)
$f_{EXT}$ : frequency of BRGi count source (external clock)
n : value set to BRGi

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



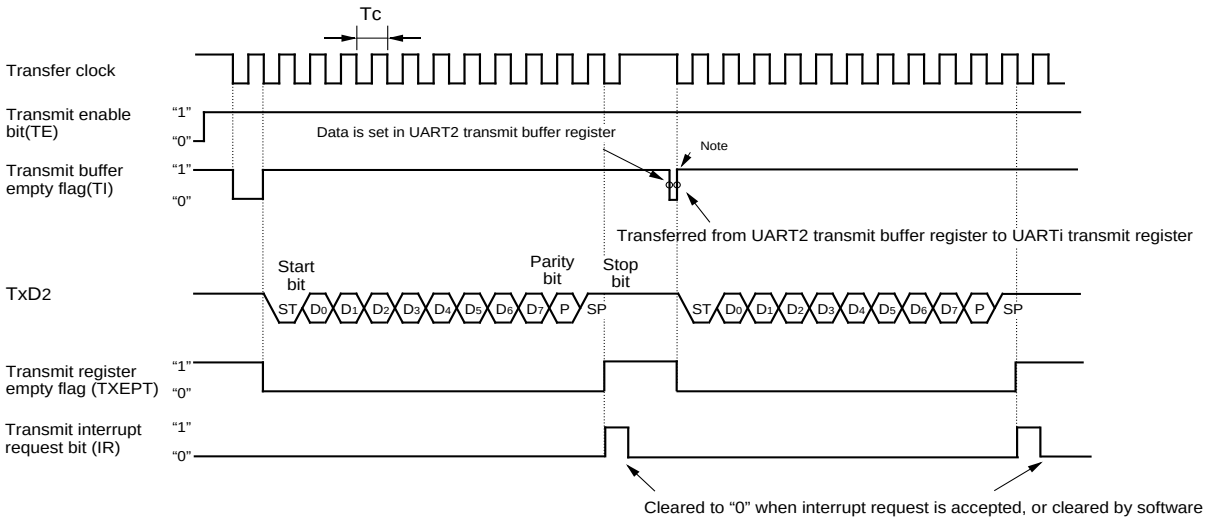

Shown in ( ) are bit symbols.

The above timing applies to the following settings :
- Parity is disabled.
- Two stop bits.
- CTS function is disabled.
- Transmit interrupt cause select bit = "0".

Tc = 16 (n + 1) / fi or 16 (n + 1) / $f_{EXT}$
fi : frequency of BRGi count source ($f_1$, $f_8$, $f_{32}$)
$f_{EXT}$ : frequency of BRGi count source (external clock)
n : value set to BRGi

**Figure 1.16.17. Typical transmit timings in UART mode(UART0,UART1)**

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




Shown in ( ) are bit symbols.

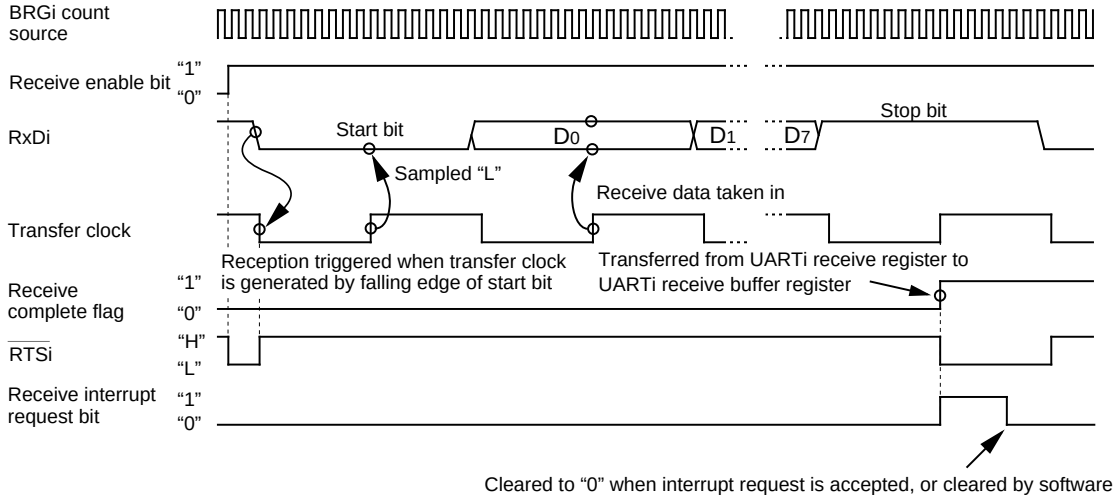
The above timing applies to the following settings :
- Parity is enabled.
- One stop bit.
- Transmit interrupt cause select bit = "1".

Tc = 16 (n + 1) / fi
fi : frequency of BRG2 count source ($f_1$, $f_8$, $f_{32}$)
n : value set to BRG2

Note: The transmit is started with overflow timing of BRG after having written in a value at the transmit buffer in the above timing.

**Figure 1.16.18. Typical transmit timings in UART mode(UART2)**

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




The above timing applies to the following settings :
- Parity is disabled.
- One stop bit.
- RTS function is selected.

**Figure 1.16.19. Typical receive timing in UART mode**

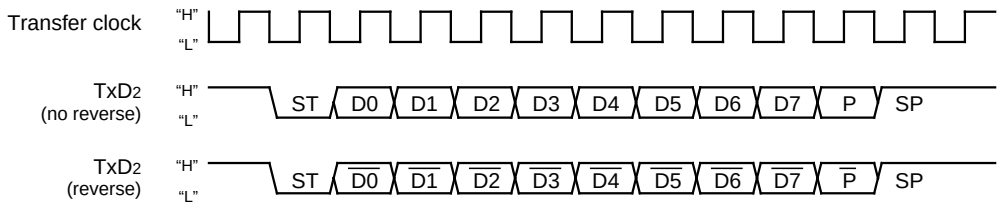
**(a) Sleep mode (UART0, UART1)**

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 $03A0_{16}$, $03A8_{16}$) 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 $037D_{16}$) is assigned 1, data is inverted in writing to the transmission buffer register or reading the reception buffer register. Figure 1.16.20 shows the example of timing for switching serial data logic.

• When LSB first, parity enabled, one stop bit




ST : Start bit
P : Even parity
SP : Stop bit

**Figure 1.16.20. Timing for switching serial data logic**

**(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 1.16.21 shows the example of detection timing of a bus collision (in UART mode).

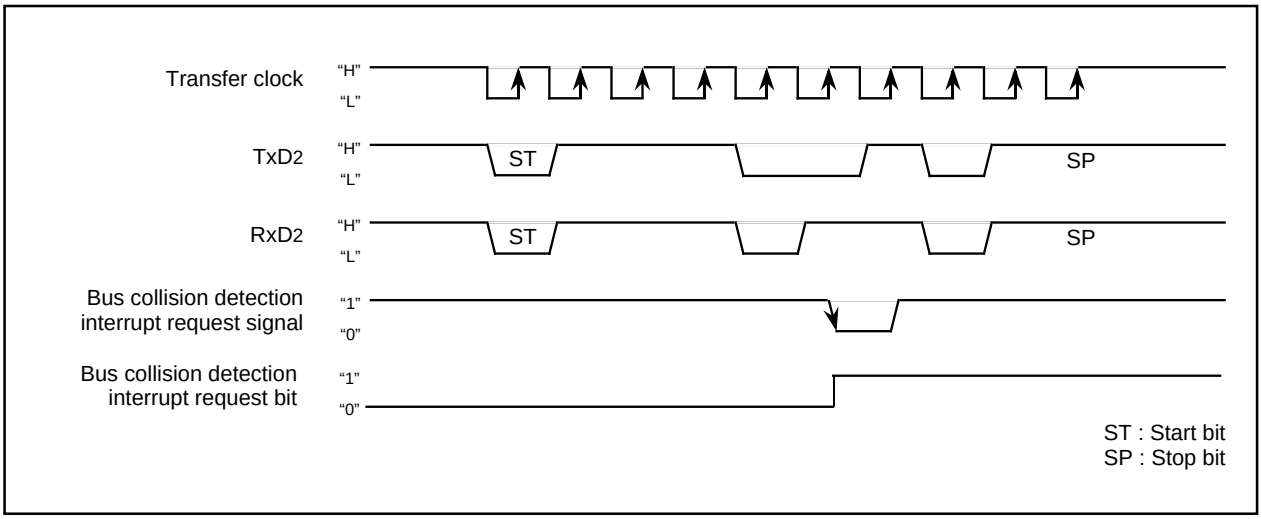



**Figure 1.16.21. Detection timing of a bus collision (in UART mode)**

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

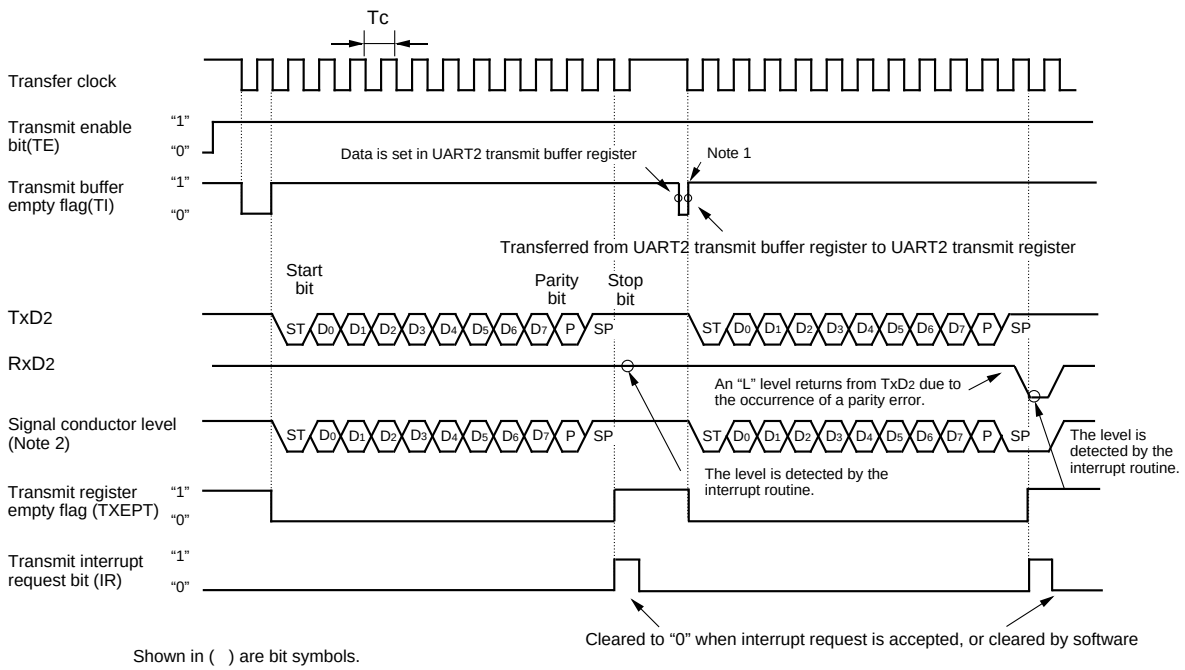
The SIM interface is used for connecting the microcomputer with a memory card or the like; adding some extra settings in UART2 clock-asynchronous serial I/O mode allows the user to effect this function. Table 1.16.8 shows the specifications of clock-asynchronous serial I/O mode (used for the SIM interface).

**Table 1.16.8. Specifications of clock-asynchronous serial I/O mode (used for the SIM interface)**

| Item | Specification |
|---|---|
| Transfer data format | • Transfer data 8-bit UART mode (bit 2 through bit 0 of address $0378_{16}$ = “$101_2$”)<br>• One stop bit (bit 4 of address $0378_{16}$ = “0”)<br>• With the direct format chosen<br>Set parity to “even” (bit 5 and bit 6 of address $0378_{16}$ = “1” and “1” respectively)<br>Set data logic to “direct” (bit 6 of address $037D_{16}$ = “0”).<br>Set transfer format to LSB (bit 7 of address $037C_{16}$ = “0”).<br>• With the inverse format chosen<br>Set parity to “odd” (bit 5 and bit 6 of address $0378_{16}$ = “0” and “1” respectively)<br>Set data logic to “inverse” (bit 6 of address $037D_{16}$ = “1”)<br>Set transfer format to MSB (bit 7 of address $037C_{16}$ = “1”) |
| Transfer clock | • With the internal clock chosen (bit 3 of address $0378_{16}$ = “0”) : fi / 16 (n + 1) (Note 1) : fi=$f_1$, $f_8$, $f_{32}$<br>(Do not set external clock) |
| Transmission / reception control | • Disable the $\overline{CTS}$ and $\overline{RTS}$ function (bit 4 of address $037C_{16}$ = “1”) |
| Other settings | • The sleep mode select function is not available for UART2<br>• Set transmission interrupt factor to “transmission completed” (bit 4 of address $037D_{16}$ = “1”) |
| Transmission start condition | • To start transmission, the following requirements must be met:<br>- Transmit enable bit (bit 0 of address $037D_{16}$) = “1”<br>- Transmit buffer empty flag (bit 1 of address $037D_{16}$) = “0” |
| Reception start condition | • To start reception, the following requirements must be met:<br>- Reception enable bit (bit 2 of address $037D_{16}$) = “1”<br>- Detection of a start bit |
| Interrupt request generation timing | • When transmitting<br>When data transmission from the UART2 transmit register is completed (bit 4 of address $037D_{16}$ = “1”)<br>• When receiving<br>When data transfer from the UART2 receive register to the UART2 receive buffer register is completed |
| Error detection | • Overrun error (see the specifications of clock-asynchronous serial I/O) (Note 2)<br>• Framing error (see the specifications of clock-asynchronous serial I/O)<br>• Parity error (see the specifications of clock-asynchronous serial I/O)<br>- On the reception side, an “L” level is output from the $TxD_2$ pin by use of the parity error signal output function (bit 7 of address $037D_{16}$ = “1”) when a parity error is detected<br>- On the transmission side, a parity error is detected by the level of input to the $RxD_2$ pin when a transmission interrupt occurs<br>• The error sum flag (see the specifications of clock-asynchronous serial I/O) |

Note 1: ‘n’ denotes the value $00_{16}$ to $FF_{16}$ 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.

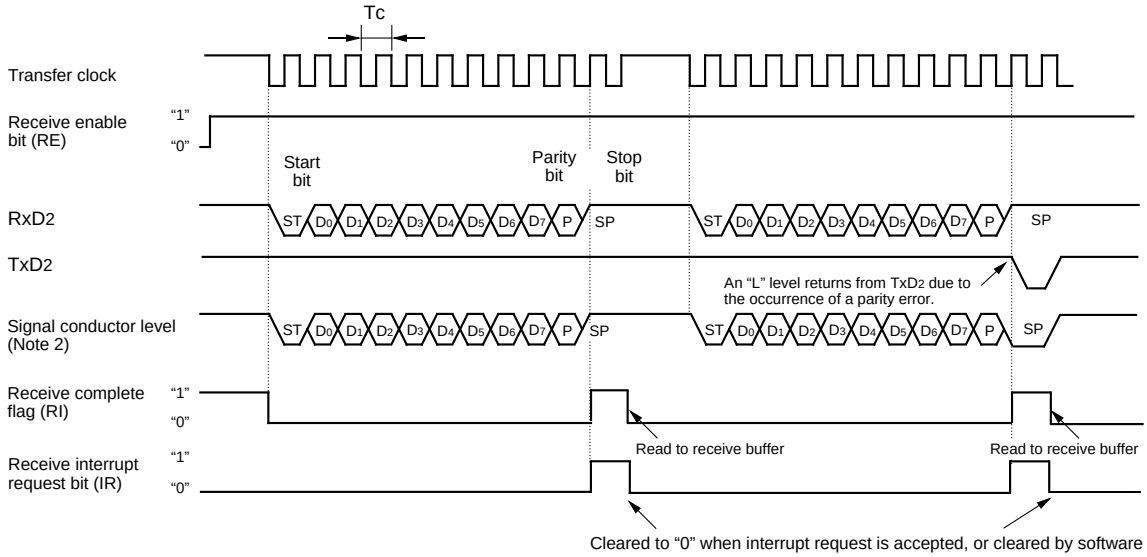



Shown in ( ) are bit symbols.

The above timing applies to the following settings :
- Parity is enabled.
- One stop bit.
- Transmit interrupt cause select bit = "1".

Tc = 16 (n + 1) / fi
fi : frequency of BRG2 count source ($f_1$, $f_8$, $f_{32}$)
n : value set to BRG2

Note 1: The transmit is started with overflow timing of BRG after having written in a value at the transmit buffer in the above timing.




Shown in ( ) are bit symbols.

The above timing applies to the following settings :
- Parity is enabled.
- One stop bit.
- Transmit interrupt cause select bit = "0".

Tc = 16 (n + 1) / fi
fi : frequency of BRG2 count source ($f_1$, $f_8$, $f_{32}$)
n : value set to BRG2

Note 2: Equal in waveform because TxD2 and RxD2 are connected.

**Figure 1.16.22. Typical transmit/receive timing in UART mode (used for the SIM interface)**

**(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 1.16.23 shows the output timing of the parity error signal.

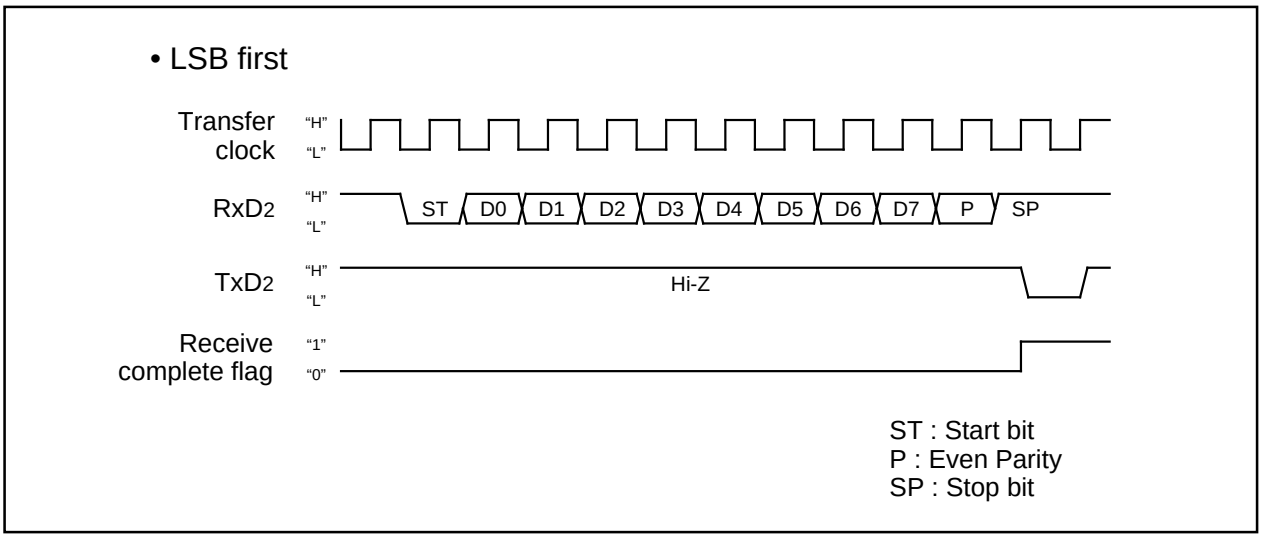



**Figure 1.16.23. Output timing of the parity error signal**

**(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 1.16.24 shows the SIM interface format.

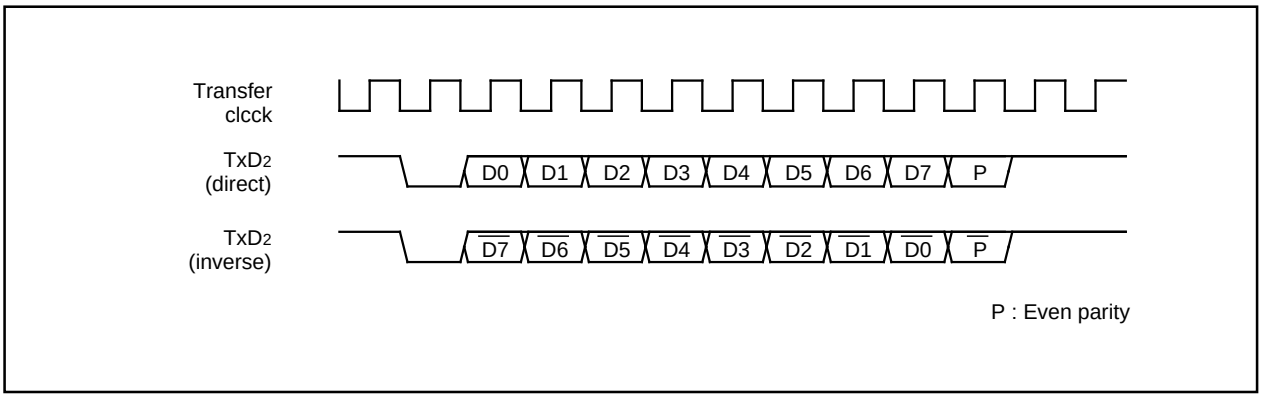



**Figure 1.16.24. SIM interface format**

Figure 1.16.25 shows the example of connecting the SIM interface. Connect TxD2 and RxD2 and apply pull-up.

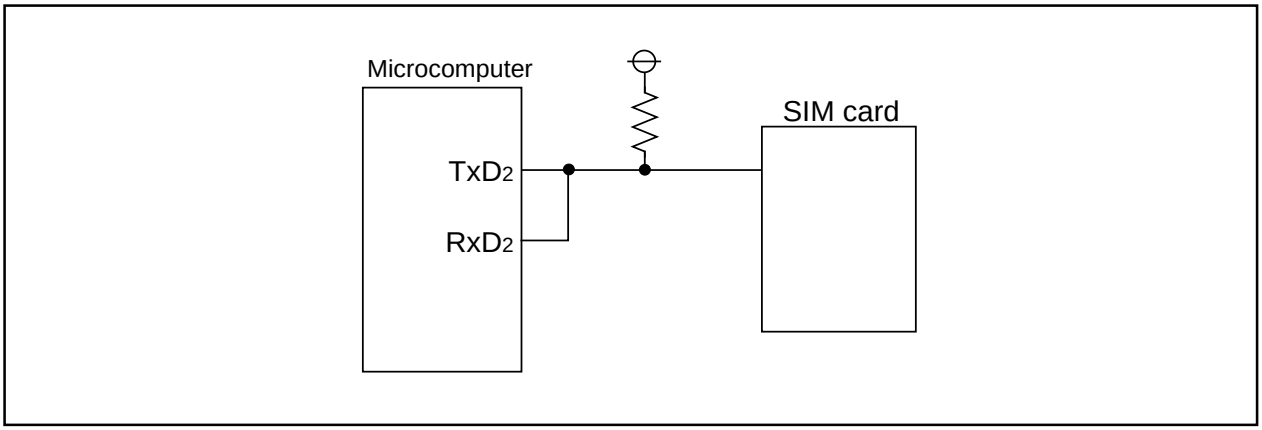



**Figure 1.16.25. Connecting the SIM interface**

## UART2 Special Mode Register

The UART2 special mode register (address $0377_{16}$) is used to control UART2 in various ways.
Figure 1.16.26 shows the UART2 special mode register.
Bit 0 of the UART2 special mode register ($0377_{16}$) is used as the $I^2C$ mode select bit.
Setting “1” in the $I^2C$ mode select bit (bit 0) goes the circuit to achieve the $I^2C$ bus (simplified $I^2C$ bus) interface effective.
Table 1.16.9 shows the relation between the $I^2C$ 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.

UART2 special mode register

b7 b6 b5 b4 b3 b2 b1 b0

| Symbol | Address | When reset |
|---|---|---|
| U2SMR | $0377_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function (During clock synchronous serial I/O mode) | Function (During UART mode) | R | W |
|---|---|---|---|---|---|
| IICM | $I^2C$ mode select bit | 0 : Normal mode<br>1 : $I^2C$ mode | Must always be “0” | ○ | ○ |
| ABC | Arbitration lost detecting flag control bit | 0 : Update per bit<br>1 : Update per byte | Must always be “0” | ○ | ○ |
| BBS | Bus busy flag | 0 : STOP condition detected<br>1 : START condition detected | Must always be “0” | ○ | ○ (Note1) |
| LSYN | SCL L sync output enable bit | 0 : Disabled<br>1 : Enabled | Must always be “0” | ○ | ○ |
| ABSCS | Bus collision detect sampling clock select bit | Must always be “0” | 0 : Rising edge of transfer clock<br>1 : Underflow signal of timer A0 | ○ | ○ |
| ACSE | Auto clear function select bit of transmit enable bit | Must always be “0” | 0 : No auto clear function<br>1 : Auto clear at occurrence of bus collision | ○ | ○ |
| SSS | Transmit start condition select bit | Must always be “0” | 0 : Ordinary<br>1 : Falling edge of $RxD_2$ | ○ | ○ |
| SDDS | SDA digital delay select bit (Note 2, Note 3) | 0 : Analog delay output is selected<br>1 : Digital delay output is selected<br>(must always be “0” when not using $I^2C$ mode) | Must always be “0” | ○ | ○ |

Note 1: Nothing but “0” may be written.
Note 2: When not in $I^2C$ mode, do not set this bit by writing a “1”. During normal mode, fix it to “0”. When this bit = “0”, UART2 special mode register 3 (U2SMR3 at address $0375_{16}$) bits 7 to 5 (DL2 to DL0 = SDA digital delay setup bits) are initialized to “000”, with the analog delay circuit selected. Also, when SDDS = “0”, the U2SMR3 register cannot be read or written to.
Note 3: When analog delay is selected, only the analog delay value is effective; when digital delay is selected, only the digital delay value is effective.

UART2 special mode register 3 ($I^2C$ bus exclusive use register)

b7 b6 b5 b4 b3 b2 b1 b0

| Symbol | Address | When reset |
|---|---|---|
| U2SMR3 | $0375_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function ($I^2C$ bus exclusive use register) | R | W |
|---|---|---|---|---|
| | | Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be indeterminate. However, when SDDS = “1”, the value “0” is read out (Note 1) | — | — |
| DL0 | SDA digital delay setup bit (Note 1, Note 2, Note 3) | b7 b6 b5<br>0 0 0 : Must not be set when using $I^2C$ mode<br>0 0 1 : 1 to 2 cycle(s) of 1/f($X_{IN}$)<br>0 1 0 : 2 to 3 cycles of 1/f($X_{IN}$)<br>0 1 1 : 3 to 4 cycles of 1/f($X_{IN}$)<br>1 0 0 : 4 to 5 cycles of 1/f($X_{IN}$)<br>1 0 1 : 5 to 6 cycles of 1/f($X_{IN}$)<br>1 1 0 : 6 to 7 cycles of 1/f($X_{IN}$)<br>1 1 1 : 7 to 8 cycles of 1/f($X_{IN}$)<br>} Digital delay is selected | ○ | ○ |
| DL1 | | | ○ | ○ |
| DL2 | | | ○ | ○ |

Note 1: This bit can be read or written to when UART2 special mode register (U2SMR at address $0377_{16}$) bit 7 (SDDS: SDA digital delay select bit) = “1”. When the initial value of UART2 special mode register 3 (U2SMR3) is read after setting SDDS = “1”, the value is “$00_{16}$”. When writing to UART2 special mode register 3 (U2SMR3) after setting SDDS = “1”, be sure to write 0's to bits 0–4. When SDDS = “0”, this register cannot be written to; when read, the value is indeterminate.
Note 2: These bits are initialized to “000” when SDDS = “0”. After a reset, these bits are set to “000”. However, because these bits can be read only when SDDS = “1”, the value read from these bits when SDDS = “0” is indeterminate.
Note 3: The amount of delay varies with the load on SCL and SDA pins. Also, when using an external clock, the amount of delay increases by about 200 ns, so be sure to take this into account when using the device.

**Figure 1.16.26. UART2 special mode register**

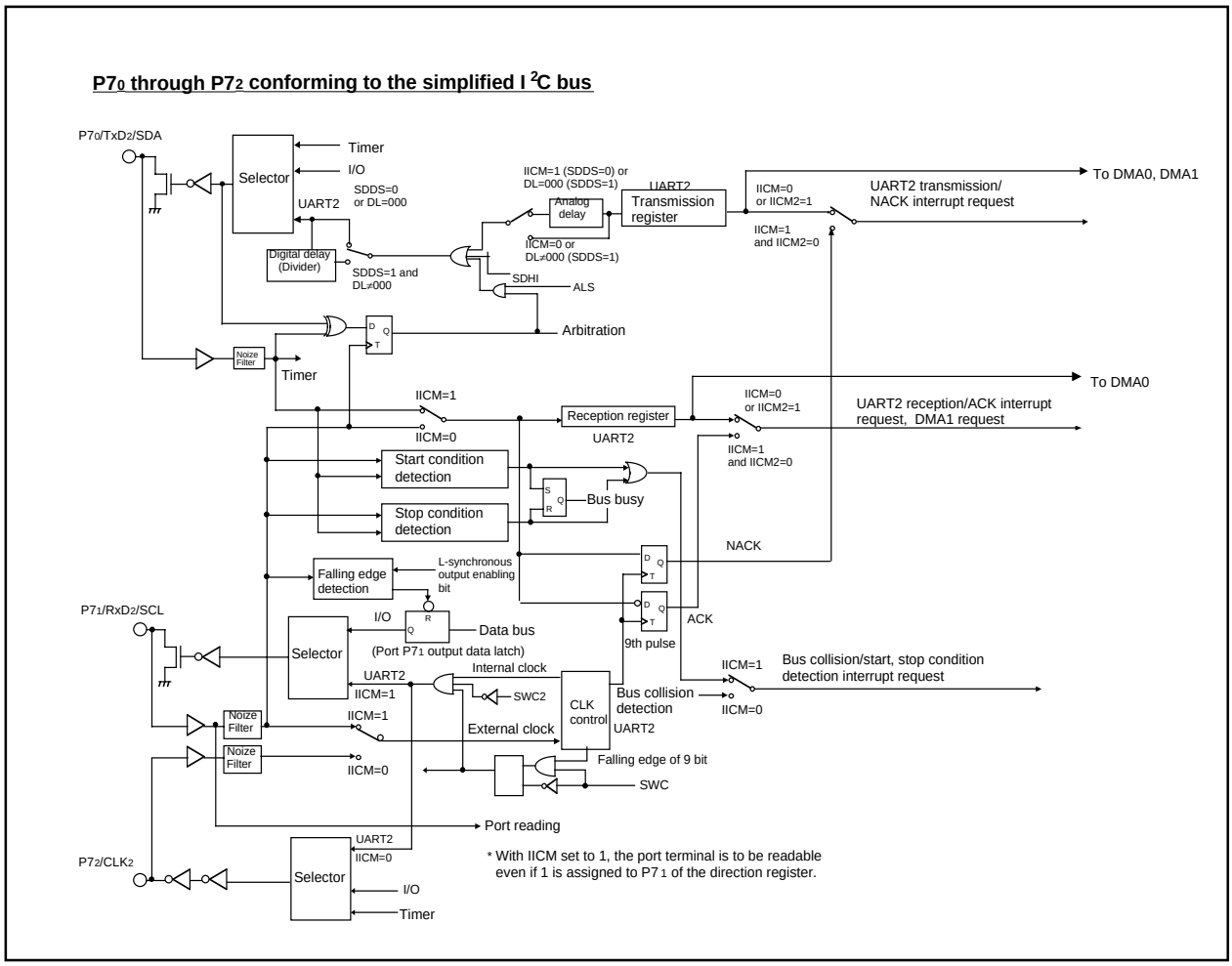



**Figure 1.16.27. Functional block diagram for I²C mode**

## Table 1.16.9. Features in I²C mode

| | Function | Normal mode | I²C 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 |
| 5 | $P7_0$ at the time when UART2 is in use | $TxD_2$ (output) | SDA (input/output) (Note 3) |
| 6 | $P7_1$ at the time when UART2 is in use | $RxD_2$ (input) | SCL (input/output) |
| 7 | $P7_2$ at the time when UART2 is in use | $CLK_2$ | $P7_2$ |
| 8 | DMA1 factor at the time when 1 1 0 1 is assigned to the DMA request factor selection bits | UART2 reception | Acknowledgment detection (ACK) |
| 9 | Noise filter width | 15ns | 200ns |
| 10 | Reading $P7_1$ | Reading the terminal when 0 is assigned to the direction register | Reading the terminal regardless of the value of the direction register |
| 11 | Initial value of UART2 output | H level (when 0 is assigned to the CLK polarity select bit) | The value set in latch $P7_0$ when the port is selected |

Note 1: Make the settings given below when I²C mode is in use.
Set 0 1 0 in bits 2, 1, 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.

Figure 1.16.27 shows the functional block diagram for $I^2C$ mode. Setting “1” in the $I^2C$ mode select bit (IICM) causes ports $P7_0$, $P7_1$, and $P7_2$ to work as data transmission-reception terminal SDA, clock input-output terminal SCL, and port $P7_2$ 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 $0377_{16}$) 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 f1 using UART2 special mode register 3 (at address $0375_{16}$). Delay circuit select conditions are shown in Table 1.16.10.

**Table 1.16.10. 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 $P7_1$ (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 $P7_0$. 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 ($P7_0$) is detected with the SCL terminal ($P7_1$) staying “H”. The stop condition detection interrupt refers to the interrupt that occurs when the rising edge of the SDA terminal ($P7_0$) is detected with the SCL terminal ($P7_1$) 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 ($0377_{16}$) 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 ($037F_{16}$, $037E_{16}$), 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 $P7_1$ data register to “0” in synchronization with the SCL terminal level going to “L”.

Some other functions added are explained here. Figure 1.16.28 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.

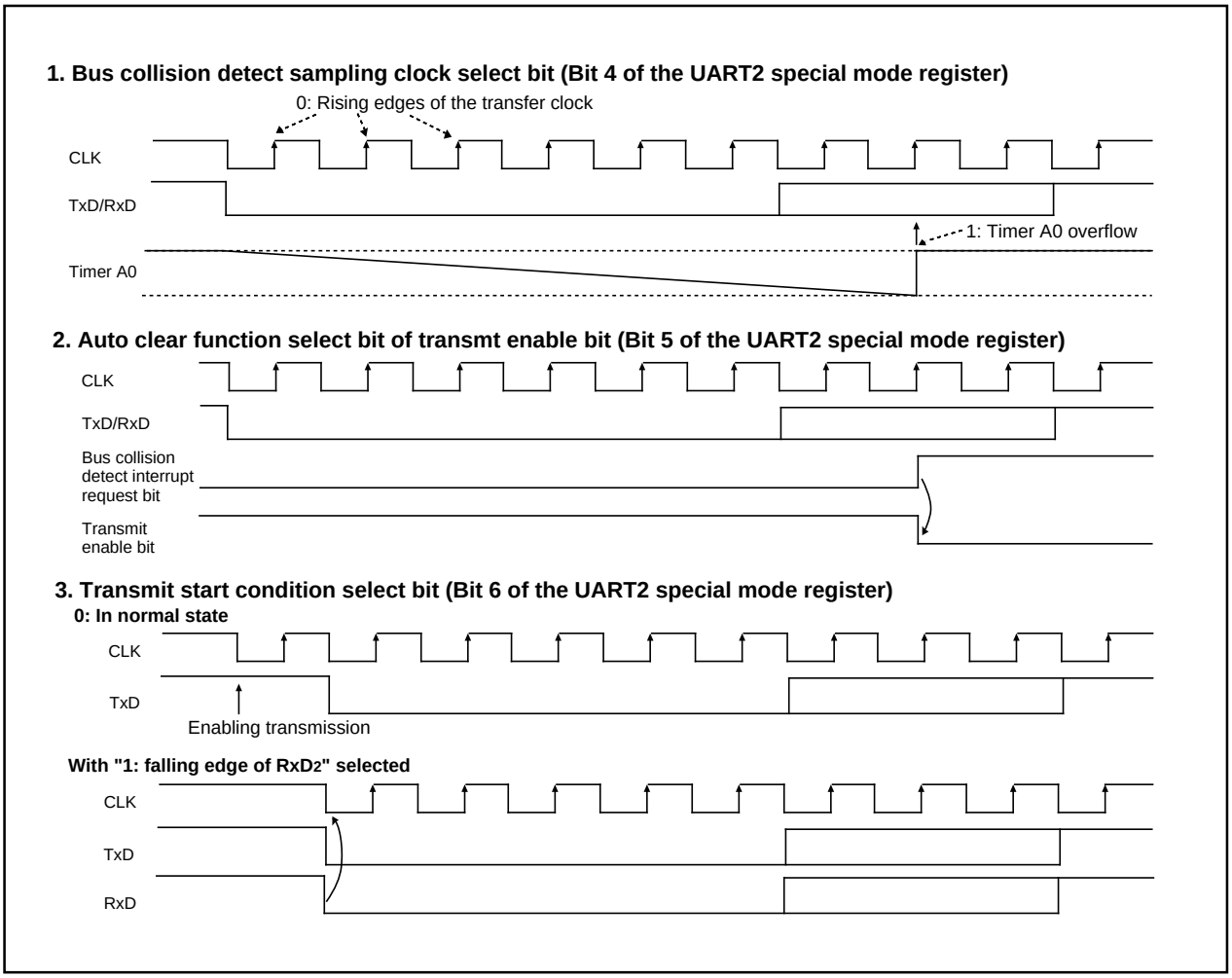



**Figure 1.16.28. Some other functions added**

## UART2 Special Mode Register 2

UART2 special mode register 2 (address $037_{16}$) is used to further control UART2 in I²C mode. Figure 1.16.29 shows the UART2 special mode register 2.

UART2 special mode register 2 (I²C bus exclusive use register)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| U2SMR2 | $037_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| IICM2 | I²C mode select bit 2 | Refer to Table 1.16.11 | ○ | ○ |
| CSC | Clock-synchronous bit | 0 : Disabled<br>1 : Enabled | ○ | ○ |
| SWC | SCL wait output bit | 0 : Disabled<br>1 : Enabled | ○ | ○ |
| ALS | SDA output stop bit | 0 : Disabled<br>1 : Enabled | ○ | ○ |
| STAC | UART2 initialization bit | 0 : Disabled<br>1 : Enabled | ○ | ○ |
| SWC2 | SCL wait output bit 2 | 0: UART2 clock<br>1: 0 output | ○ | ○ |
| SDHI | SDA output disable bit | 0: Enabled<br>1: Disabled (high impedance) | ○ | ○ |
| SHTC | Start/stop condition control bit | Set this bit to “1” in I²C mode<br>(refer to Table 1.16.12) | ○ | ○ |

**Figure 1.16.29. UART2 special mode register 2**

Bit 0 of the UART2 special mode register 2 (address $0376_{16}$) is used as the $I^2C$ mode select bit 2. Table 1.16.11 shows the types of control to be changed by $I^2C$ mode select bit 2 when the $I^2C$ mode select bit is set to “1”. Table 1.16.12 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 $I^2C$ mode.

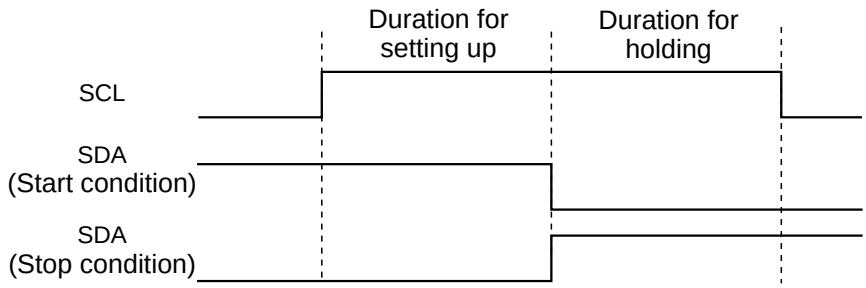
**Table 1.16.11. Functions changed by $I^2C$ mode select bit 2**

| | 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 1.16.12. Timing characteristics of detecting the start condition and the stop condition (Note 1)**

| |
|---|
| 3 to 6 cycles < duration for setting-up (Note 2) |
| 3 to 6 cycles < duration for holding (Note 2) |

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.



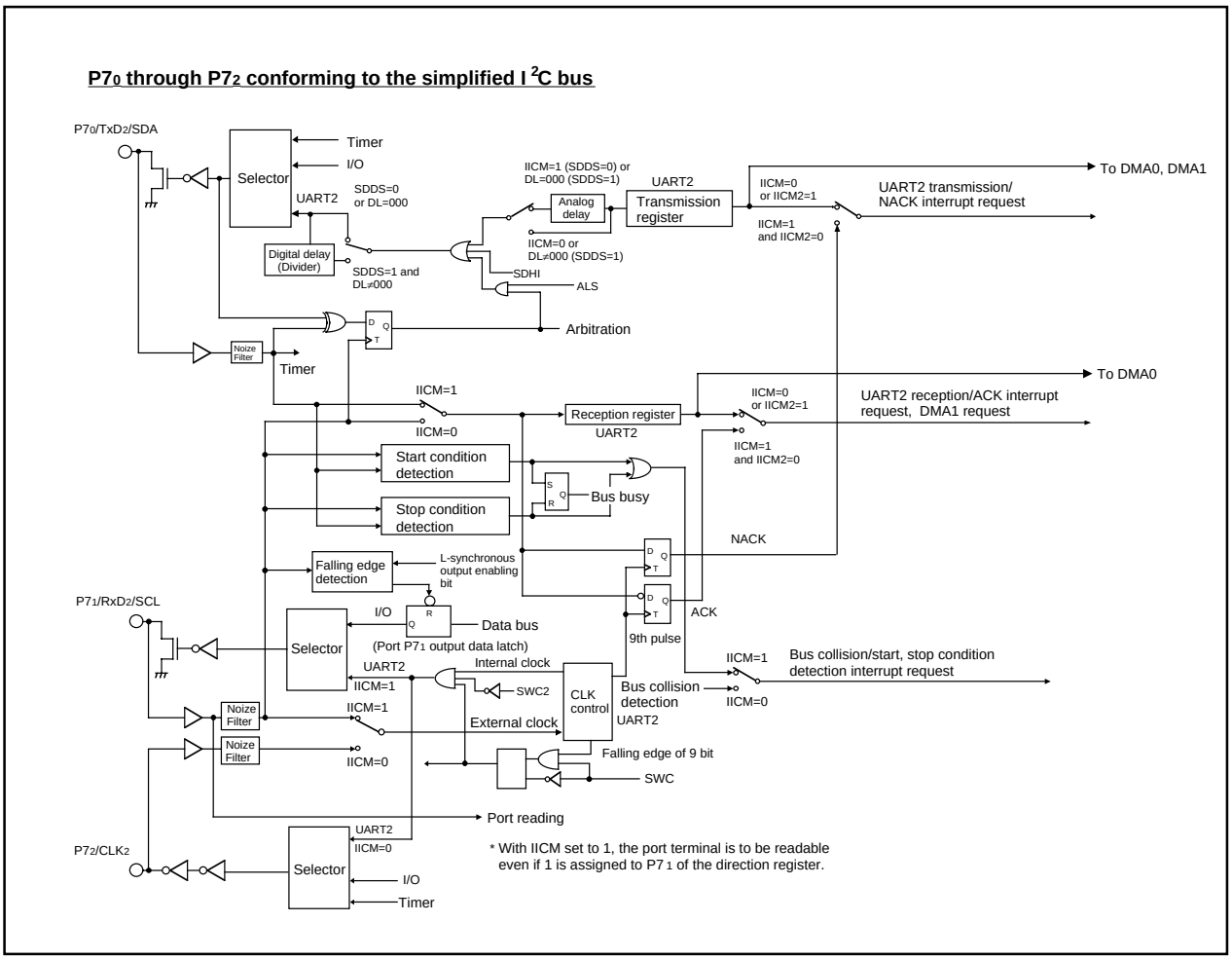



**Figure 1.16.30. Functional block diagram for I²C mode**

Functions available in I²C mode are shown in Figure 1.16.30 — a functional block diagram.

Bit 3 of the UART2 special mode register 2 (address 037$_{16}$) 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 037$_{16}$) 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 (037$_{16}$) 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”.

Bit 4 of the UART2 special mode register 2 (address $0376_{16}$) 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 ($0376_{16}$) 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 “1” 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 ($0376_{16}$) 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 1.16.31 shows the S I/O3, 4 block diagram, and Figure 1.16.32 shows the S I/O3, 4 related register.

Table 1.16.13 shows the specifications of S I/O3, 4.

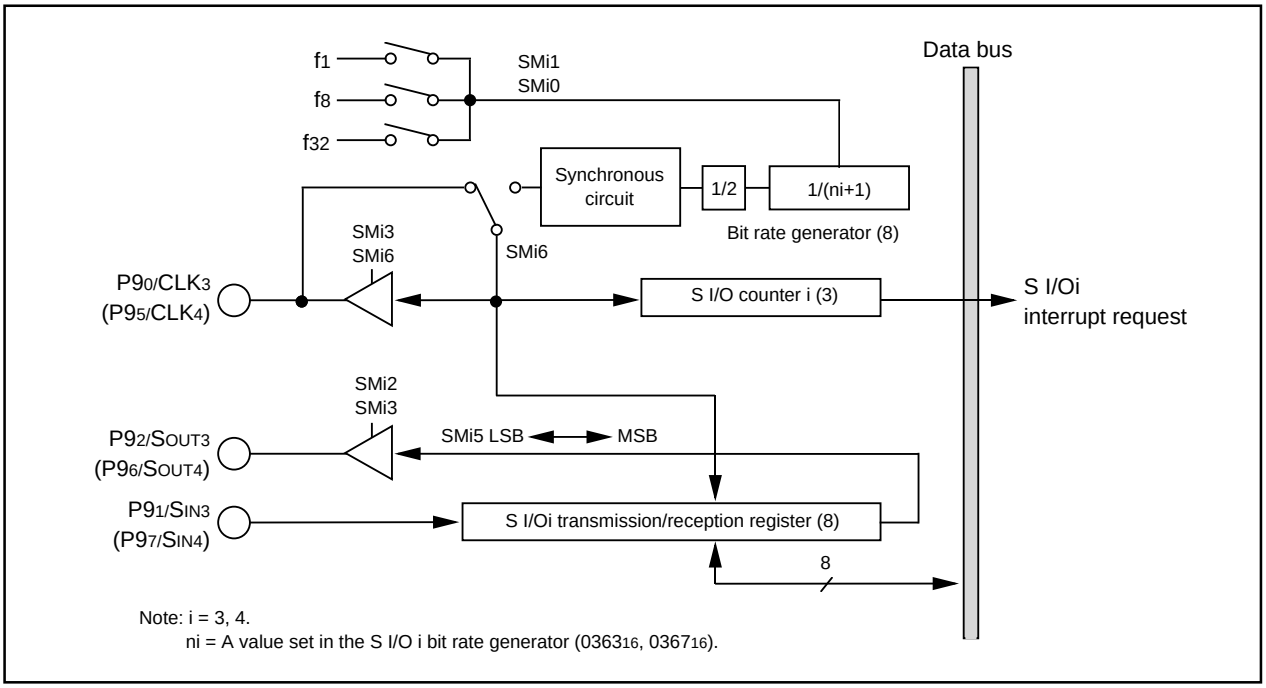
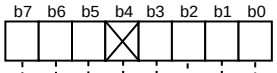



**Figure 1.16.31. S I/O3, 4 block diagram**

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

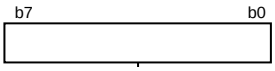



| Symbol | Address | When reset |
|---|---|---|
| SiC | $036_{16}$, $0366_{16}$ | $40_{16}$ |

| Bit symbol | Bit name | Description | R | W |
|---|---|---|---|---|
| SMi0 | Internal synchronous clock select bit | b1 b0<br>0 0 : Selecting $f_1$<br>0 1 : Selecting $f_8$<br>1 0 : Selecting $f_{32}$<br>1 1 : Must not be set. | ○ | ○ |
| SMi1 | | | ○ | ○ |
| SMi2 | SOUTi output disable bit | 0 : SOUTi output<br>1 : SOUTi output disable(high impedance) | ○ | ○ |
| SMi3 | S I/Oi port select bit (Note 2) | 0 : Input-output port<br>1 : SOUTi 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<br>1 : MSB first | ○ | ○ |
| SMi6 | Synchronous clock select bit (Note 2) | 0 : External clock<br>1 : Internal clock | ○ | ○ |
| SMi7 | SOUTi initial value set bit | Effective when SMi3 = 0<br>0 : L output<br>1 : H output | ○ | ○ |

Note 1: Set "1" in bit 2 of the protection register ($000A_{16}$) 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 SI/Oi port select bit (i = 3, 4) to "0", be sure to set the sync clock select bit to "1".

SI/Oi bit rate generator (Note 1, 2)

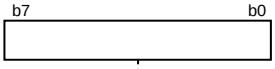



| Symbol | Address | When reset |
|---|---|---|
| S3BRG | $0363_{16}$ | Indeterminate |
| S4BRG | $0367_{16}$ | Indeterminate |

| Indeterminate | Values that can be set | R | W |
|---|---|---|---|
| Assuming that set value = n, BRGi divides the count source by n + 1 | $00_{16}$ to $FF_{16}$ | | ○ |

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

SI/Oi transmit/receive register (Note)




| Symbol | Address | When reset |
|---|---|---|
| S3TRR | $0360_{16}$ | Indeterminate |
| S4TRR | $0364_{16}$ | Indeterminate |

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

Note: Write a value to this register while transmit/receive halts.

**Figure 1.16.32. S I/O3, 4 related register**

**Table 1.16.13. Specifications of S I/O3, 4**

| Item | Specifications |
|---|---|
| Transfer data format | • Transfer data length: 8 bits |
| Transfer clock | • With the internal clock selected (bit 6 of $0362_{16}$, $0366_{16}$ = "1"): $f_1/2(ni+1)$, $f_8/2(ni+1)$, $f_{32}/2(ni+1)$ (Note 1)<br>• With the external clock selected (bit 6 of $0362_{16}$, $0366_{16}$ = 0):Input from the CLKi terminal (Note 2) |
| Conditions for transmission/ reception start | • To start transmit/reception, the following requirements must be met:<br>- Select the synchronous clock (use bit 6 of $0362_{16}$, $0366_{16}$).<br>Select a frequency dividing ratio if the internal clock has been selected (use bits 0 and 1 of $0362_{16}$, $0366_{16}$).<br>- SOUTi initial value set bit (use bit 7 of $0362_{16}$, $0366_{16}$)= 1.<br>- S I/Oi port select bit (bit 3 of $0362_{16}$, $0366_{16}$) = 1.<br>- Select the transfer direction (use bit 5 of $0362_{16}$, $0366_{16}$)<br>-Write transfer data to SI/Oi transmit/receive register ($0360_{16}$, $0364_{16}$)<br>• To use S I/Oi interrupts, the following requirements must be met:<br>- Clear the SI/Oi interrupt request bit before writing transfer data to the SI/Oi transmit/receive register (bit 3 of $0049_{16}$, $0048_{16}$) = 0. |
| Interrupt request generation timing | • Rising edge of the last transfer clock. (Note 3) |
| Select function | • LSB first or MSB first selection<br>Whether transmission/reception begins with bit 0 (LSB) or bit 7 (MSB) can be selected.<br>• Function for setting an SOUTi initial value selection<br>When using an external clock for the transfer clock, the user can choose the SOUTi pin output level during a non-transfer time. For details on how to set, see Figure 1.16.33. |
| Precaution | • 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 $0360_{16}$, $0364_{16}$) during a transfer.<br>• When the internal clock is selected for the transfer clock, SOUTi 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 $0360_{16}$, $0364_{16}$) during this time, SOUTi is placed in the high-impedance state immediately upon writing and the data hold time is thereby reduced. |

Note 1: n is a value from $00_{16}$ through $FF_{16}$ 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 $0360_{16}$, $0364_{16}$), the CLKi pin input must be in the high state. Also, before rewriting the SI/Oi Control Register (addresses $0362_{16}$, $0366_{16}$)'s bit 7 (SOUTi initial value set bit), make sure the CLKi 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.

## ■ 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 1.16.33 shows the timing chart for setting an SOUTi initial value and how to set it.

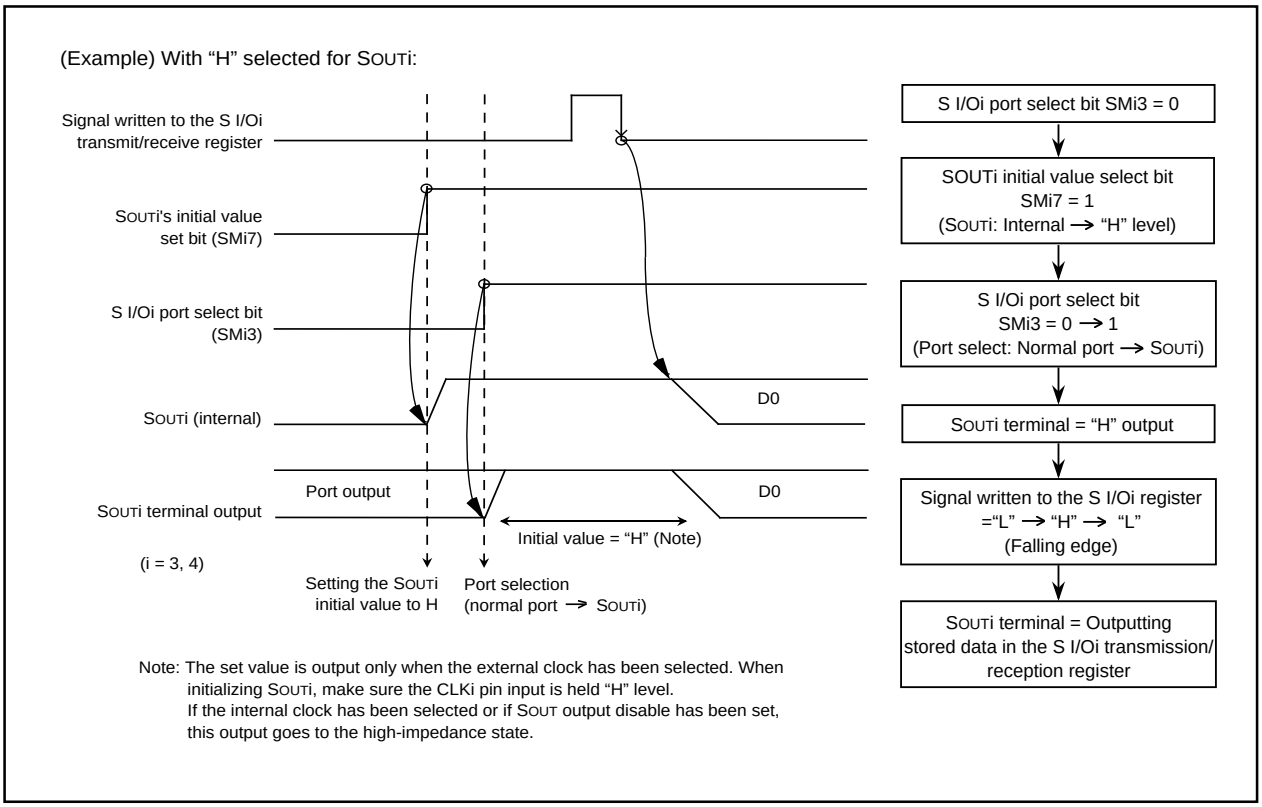



**Figure 1.16.33. Timing chart for setting SOUTi's initial value and how to set it**

## ■ S I/Oi operation timing

Figure 1.16.34 shows the S I/Oi operation timing

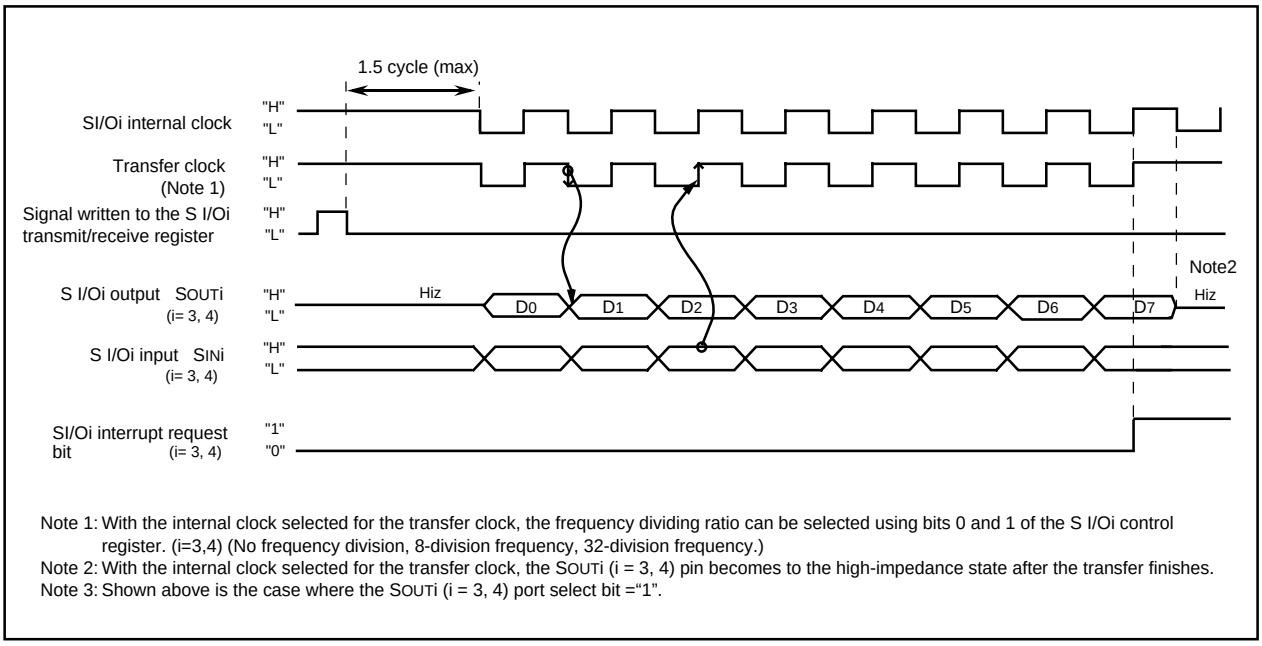



**Figure 1.16.34. S I/Oi operation timing chart**

## A-D Converter

The A-D converter consists of one 10-bit successive approximation A-D converter circuit with a capacitive coupling amplifier. Pins $P10_0$ to $P10_7$, $P9_5$, and $P9_6$ 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 $03D7_{16}$) can be used to isolate the resistance ladder of the A-D converter from the reference voltage input pin ($V_{REF}$) when the A-D converter is not used. Doing so stops any current flowing into the resistance ladder from $V_{REF}$, reducing the power dissipation. When using the A-D converter, start A-D conversion only after setting bit 5 of $03D7_{16}$ to connect $V_{REF}$. 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 1.17.1 shows the performance of the A-D converter. Figure 1.17.1 shows the block diagram of the A-D converter, and Figures 1.17.2 and 1.17.3 show the A-D converter-related registers.

**Table 1.17.1. Performance of A-D converter**

| Item | Performance |
|---|---|
| Method of A-D conversion | Successive approximation (capacitive coupling amplifier) |
| Analog input voltage (Note 1) | 0V to $AV_{CC}$ ($V_{CC}$) |
| Operating clock $\phi_{AD}$ (Note 2) | $V_{CC}$ = 3V divide-by-2 of $f_{AD}$/divide-by-4 of $f_{AD}$, $f_{AD}$=f($X_{IN}$) |
| Resolution | 8-bit or 10-bit (selectable) |
| Absolute precision | $V_{CC}$ = 3V • Without sample and hold function (8-bit resolution) ±2LSB<br>• Without sample and hold function (10-bit resolution) ±8LSB |
| Operating modes | One-shot mode, repeat mode, single sweep mode, repeat sweep mode 0, and repeat sweep mode 1 |
| Analog input pins | 8pins ($AN_0$ to $AN_7$) + 2pins (ANEX0 and ANEX1) |
| A-D conversion start condition | • Software trigger<br>A-D conversion starts when the A-D conversion start flag changes to “1”<br>• External trigger (can be retriggered)<br>A-D conversion starts when the A-D conversion start flag is “1” and the $\overline{ADTRG}$/$P9_7$ input changes from “H” to “L” |
| Conversion speed per pin | • Without sample and hold function<br>8-bit resolution: 49 $\phi_{AD}$ cycles, 10-bit resolution: 59 $\phi_{AD}$ cycles |

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

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

Note 3: Setting the bit 0 (reserved bit) of the A-D control register 2 (address $03D4_{16}$) to “0”, without sample & hold function is selected.
Setting the bit 4 (reserved bit) of the A-D control register 1 (address $03D7_{16}$) to “0”, $f_{AD}/2$ or $f_{AD}/4$ are selected as AD operation clock.

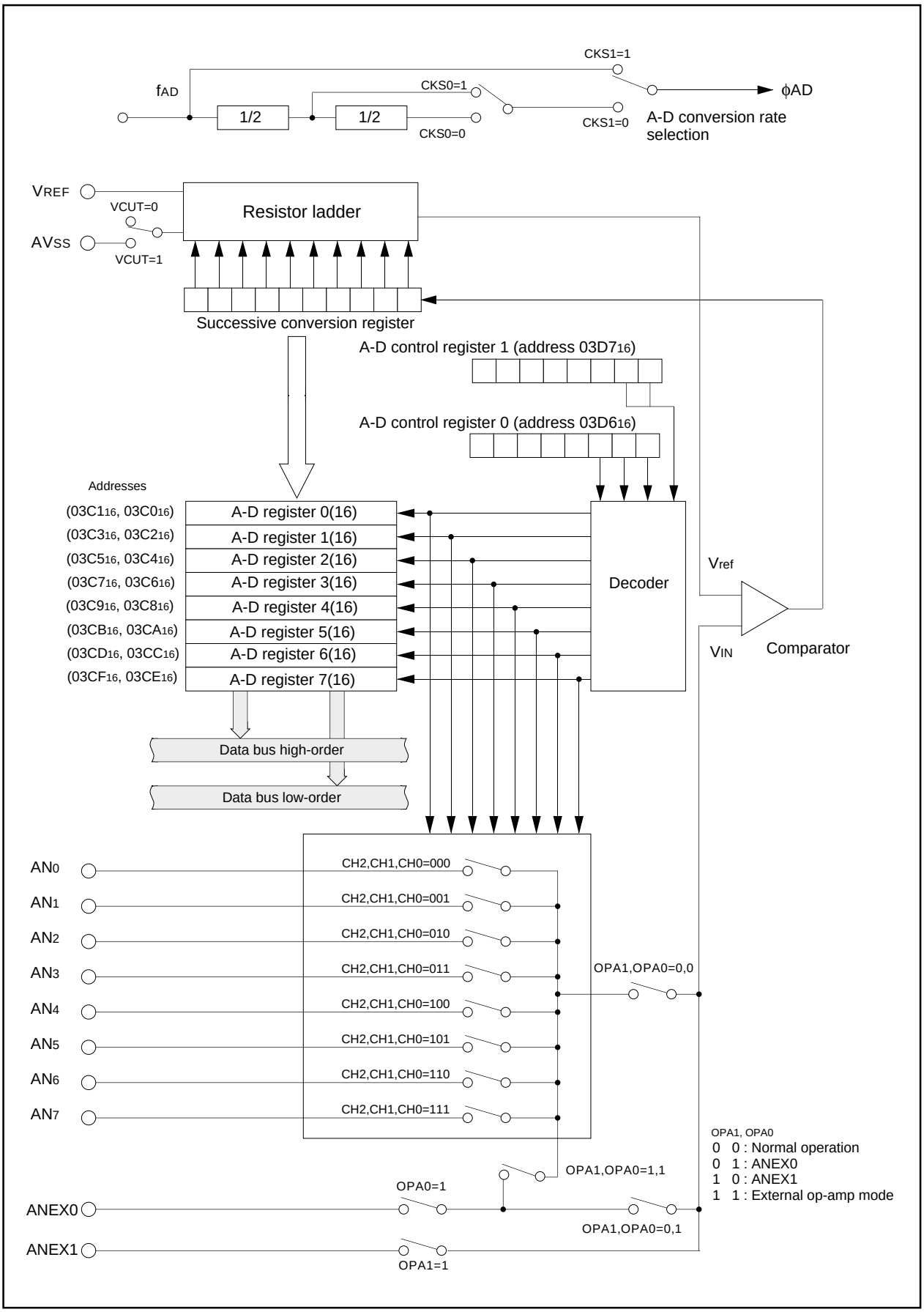



**Figure 1.17.1. Block diagram of A-D converter**

A-D control register 0 (Note 1)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| ADCON0 | $03D6_{16}$ | $00000XXX_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| CH0 | Analog input pin select bit | b2 b1 b0<br>0 0 0 : $AN_0$ is selected<br>0 0 1 : $AN_1$ is selected<br>0 1 0 : $AN_2$ is selected<br>0 1 1 : $AN_3$ is selected<br>1 0 0 : $AN_4$ is selected<br>1 0 1 : $AN_5$ is selected<br>1 1 0 : $AN_6$ is selected<br>1 1 1 : $AN_7$ is selected (Note 2) | ○ | ○ |
| CH1 | | | ○ | ○ |
| CH2 | | | ○ | ○ |
| MD0 | A-D operation mode select bit 0 | b4 b3<br>0 0 : One-shot mode<br>0 1 : Repeat mode<br>1 0 : Single sweep mode<br>1 1 : Repeat sweep mode 0<br>Repeat sweep mode 1 (Note 2) | ○ | ○ |
| MD1 | | | ○ | ○ |
| TRG | Trigger select bit | 0 : Software trigger<br>1 : $AD_{TRG}$ trigger | ○ | ○ |
| ADST | A-D conversion start flag | 0 : A-D conversion disabled<br>1 : A-D conversion started | ○ | ○ |
| CKS0 | Frequency select bit 0 | 0 : $f_{AD}/4$ is selected<br>1 : $f_{AD}/2$ is selected | ○ | ○ |

Note 1: If the A-D control register is rewritten during A-D conversion, the conversion result is indeterminate.
Note 2: When changing A-D operation mode, set analog input pin again.

A-D control register 1 (Note)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | 0 | | | | |

| Symbol | Address | When reset |
|---|---|---|
| ADCON1 | $03D7_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| SCAN0 | A-D sweep pin select bit | When single sweep and repeat sweep mode 0 are selected<br>b1 b0<br>0 0 : $AN_0$, $AN_1$ (2 pins)<br>0 1 : $AN_0$ to $AN_3$ (4 pins)<br>1 0 : $AN_0$ to $AN_5$ (6 pins)<br>1 1 : $AN_0$ to $AN_7$ (8 pins) | ○ | ○ |
| SCAN1 | | When repeat sweep mode 1 is selected<br>b1 b0<br>0 0 : $AN_0$ (1 pin)<br>0 1 : $AN_0$, $AN_1$ (2 pins)<br>1 0 : $AN_0$ to $AN_2$ (3 pins)<br>1 1 : $AN_0$ to $AN_3$ (4 pins) | ○ | ○ |
| MD2 | A-D operation mode select bit 1 | 0 : Any mode other than repeat sweep mode 1<br>1 : Repeat sweep mode 1 | ○ | ○ |
| BITS | 8/10-bit mode select bit | 0 : 8-bit mode<br>1 : 10-bit mode | ○ | ○ |
| Reserved bit | | Must always be set to “0” | ○ | ○ |
| VCUT | Vref connect bit | 0 : Vref not connected<br>1 : Vref connected | ○ | ○ |
| OPA0 | External op-amp connection mode bit | b7 b6<br>0 0 : ANEX0 and ANEX1 are not used<br>0 1 : ANEX0 input is A-D converted<br>1 0 : ANEX1 input is A-D converted<br>1 1 : External op-amp connection mode | ○ | ○ |
| OPA1 | | | ○ | ○ |

Note: If the A-D control register is rewritten during A-D conversion, the conversion result is indeterminate.

**Figure 1.17.2. A-D converter-related registers (1)**

A-D control register 2 (Note)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| X | X | X | X | 0 | 0 | 0 | 0 |

| Symbol | Address | When reset |
|---|---|---|
| ADCON2 | $03D4_{16}$ | $0000XXX0_{2}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| Reserved bit | | Must always be set to “0” | ○ | ○ |
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be “0”. | | | — | — |

Note: If the A-D control register is rewritten during A-D conversion, the conversion result is indeterminate.

A-D register i

| Symbol | Address | When reset |
|---|---|---|
| ADi(i=0 to 7) | $03C0_{16}$ to $03CF_{16}$ | Indeterminate |

| (b15) b7 | | | | | | | (b8) b0 | b7 | b0 |
|---|---|---|---|---|---|---|---|---|---|
| X | X | X | X | X | X | X | | | |

| Function | R | W |
|---|---|---|
| Eight low-order bits of A-D conversion result | ○ | × |
| • During 10-bit mode<br>Two high-order bits of A-D conversion result | ○ | × |
| • During 8-bit mode<br>When read, the content is indeterminate | × | × |
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be “0”. | — | — |

**Figure 1.17.3. A-D converter-related registers (2)**

### (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 1.17.2 shows the specifications of one-shot mode. Figure 1.17.4 shows the A-D control register in one-shot mode.

**Table 1.17.2. 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 | • End of A-D conversion (A-D conversion start flag changes to “0”, except when external trigger is selected)<br>• Writing “0” to A-D conversion start flag |
| Interrupt request generation timing | End of A-D conversion |
| Input pin | One of $AN_0$ to $AN_7$, as selected |
| Reading of result of A-D converter | Read A-D register corresponding to selected pin |

A-D control register 0 (Note 1)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | 0 | 0 | | | |

| Symbol | Address | When reset |
|---|---|---|
| ADCON0 | $03D6_{16}$ | $00000XXX_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| CH0 | Analog input pin select bit | b2 b1 b0<br>0 0 0 : $AN_0$ is selected<br>0 0 1 : $AN_1$ is selected<br>0 1 0 : $AN_2$ is selected<br>0 1 1 : $AN_3$ is selected<br>1 0 0 : $AN_4$ is selected<br>1 0 1 : $AN_5$ is selected<br>1 1 0 : $AN_6$ is selected<br>1 1 1 : $AN_7$ is selected (Note 2) | O | O |
| CH1 | | | O | O |
| CH2 | | | O | O |
| MD0 | A-D operation mode select bit 0 | b4 b3<br>0 0 : One-shot mode (Note 2) | O | O |
| MD1 | | | O | O |
| TRG | Trigger select bit | 0 : Software trigger<br>1 : $AD_{TRG}$ trigger | O | O |
| ADST | A-D conversion start flag | 0 : A-D conversion disabled<br>1 : A-D conversion started | O | O |
| CKS0 | Frequency select bit 0 | 0: $f_{AD}/4$ is selected<br>1: $f_{AD}/2$ is selected | O | O |

Note 1: If the A-D control register is rewritten during A-D conversion, the conversion result is indeterminate.
Note 2: When changing A-D operation mode, set analog input pin again.

A-D control register 1 (Note)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | 1 | 0 | | 0 | | |

| Symbol | Address | When reset |
|---|---|---|
| ADCON1 | $03D7_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| SCAN0 | A-D sweep pin select bit | Invalid in one-shot mode | O | O |
| SCAN1 | | | O | O |
| MD2 | A-D operation mode select bit 1 | Set to “0” when this mode is selected | O | O |
| BITS | 8/10-bit mode select bit | 0 : 8-bit mode<br>1 : 10-bit mode | O | O |
| Reserved bit | | Must always be set to “0” | O | O |
| VCUT | Vref connect bit | 1 : Vref connected | O | O |
| OPA0 | External op-amp connection mode bit | b7 b6<br>0 0 : ANEX0 and ANEX1 are not used<br>0 1 : ANEX0 input is A-D converted<br>1 0 : ANEX1 input is A-D converted<br>1 1 : External op-amp connection mode | O | O |
| OPA1 | | | O | O |

Note: If the A-D control register is rewritten during A-D conversion, the conversion result is indeterminate.

**Figure 1.17.4. A-D conversion register in one-shot mode**

## (2) Repeat mode

In repeat mode, the pin selected using the analog input pin select bit is used for repeated A-D conversion. Table 1.17.3 shows the specifications of repeat mode. Figure 1.17.5 shows the A-D control register in repeat mode.

**Table 1.17.3. 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 $AN_0$ to $AN_7$, as selected |
| Reading of result of A-D converter | Read A-D register corresponding to selected pin (at any time) |

A-D control register 0 (Note 1)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | 0 | 1 | | | |

| Symbol | Address | When reset |
|---|---|---|
| ADCON0 | $03D6_{16}$ | $00000XXX_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| CH0 | Analog input pin select bit | b2 b1 b0<br>0 0 0 : $AN_0$ is selected<br>0 0 1 : $AN_1$ is selected<br>0 1 0 : $AN_2$ is selected<br>0 1 1 : $AN_3$ is selected<br>1 0 0 : $AN_4$ is selected<br>1 0 1 : $AN_5$ is selected<br>1 1 0 : $AN_6$ is selected<br>1 1 1 : $AN_7$ is selected (Note 2) | ○ | ○ |
| CH1 | | | ○ | ○ |
| CH2 | | | ○ | ○ |
| MD0 | A-D operation mode select bit 0 | b4 b3<br>0 1 : Repeat mode (Note 2) | ○ | ○ |
| MD1 | | | ○ | ○ |
| TRG | Trigger select bit | 0 : Software trigger<br>1 : $AD_{TRG}$ trigger | ○ | ○ |
| ADST | A-D conversion start flag | 0 : A-D conversion disabled<br>1 : A-D conversion started | ○ | ○ |
| CKS0 | Frequency select bit 0 | 0 : $f_{AD}/4$ is selected<br>1 : $f_{AD}/2$ is selected | ○ | ○ |

Note 1: If the A-D control register is rewritten during A-D conversion, the conversion result is indeterminate.
Note 2: When changing A-D operation mode, set analog input pin again.

A-D control register 1 (Note)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | 1 | 0 | | 0 | | |

| Symbol | Address | When reset |
|---|---|---|
| ADCON1 | $03D7_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| SCAN0 | A-D sweep pin select bit | Invalid in repeat mode | ○ | ○ |
| SCAN1 | | | ○ | ○ |
| MD2 | A-D operation mode select bit 1 | Set to “0” when this mode is selected | ○ | ○ |
| BITS | 8/10-bit mode select bit | 0 : 8-bit mode<br>1 : 10-bit mode | ○ | ○ |
| Reserved bit | | Must always be set to “0” | ○ | ○ |
| VCUT | Vref connect bit | 1 : Vref connected | ○ | ○ |
| OPA0 | External op-amp connection mode bit | b7 b6<br>0 0 : ANEX0 and ANEX1 are not used<br>0 1 : ANEX0 input is A-D converted<br>1 0 : ANEX1 input is A-D converted<br>1 1 : External op-amp connection mode | ○ | ○ |
| OPA1 | | | ○ | ○ |

Note: If the A-D control register is rewritten during A-D conversion, the conversion result is indeterminate.

**Figure 1.17.5. A-D conversion register in repeat mode**

### (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 1.17.4 shows the specifications of single sweep mode. Figure 1.17.6 shows the A-D control register in single sweep mode.

**Table 1.17.4. 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 | • End of A-D conversion (A-D conversion start flag changes to “0”, except when external trigger is selected)<br>• Writing “0” to A-D conversion start flag |
| Interrupt request generation timing | End of A-D conversion |
| Input pin | $AN_0$ and $AN_1$ (2 pins), $AN_0$ to $AN_3$ (4 pins), $AN_0$ to $AN_5$ (6 pins), or $AN_0$ to $AN_7$ (8 pins) |
| Reading of result of A-D converter | Read A-D register corresponding to selected pin |

A-D control register 0 (Note)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | 1 | 0 | | | |

| Symbol | Address | When reset |
|---|---|---|
| ADCON0 | $03D6_{16}$ | $00000XXX_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| CH0 | Analog input pin select bit | Invalid in single sweep mode | O | O |
| CH1 | | | O | O |
| CH2 | | | O | O |
| MD0 | A-D operation mode select bit 0 | b4 b3<br>1 0 : Single sweep mode | O | O |
| MD1 | | | O | O |
| TRG | Trigger select bit | 0 : Software trigger<br>1 : $AD_{TRG}$ trigger | O | O |
| ADST | A-D conversion start flag | 0 : A-D conversion disabled<br>1 : A-D conversion started | O | O |
| CKS0 | Frequency select bit 0 | 0 : $f_{AD}/4$ is selected<br>1 : $f_{AD}/2$ is selected | O | O |

Note: If the A-D control register is rewritten during A-D conversion, the conversion result is indeterminate.

A-D control register 1 (Note 1)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | 1 | 0 | | 0 | | |

| Symbol | Address | When reset |
|---|---|---|
| ADCON1 | $03D7_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| SCAN0 | A-D sweep pin select bit | When single sweep and repeat sweep mode 0 are selected<br>b1 b0<br>0 0 : $AN_0$, $AN_1$ (2 pins)<br>0 1 : $AN_0$ to $AN_3$ (4 pins)<br>1 0 : $AN_0$ to $AN_5$ (6 pins)<br>1 1 : $AN_0$ to $AN_7$ (8 pins) | O | O |
| SCAN1 | | | O | O |
| MD2 | A-D operation mode select bit 1 | Set to “0” when this mode is selected | O | O |
| BITS | 8/10-bit mode select bit | 0 : 8-bit mode<br>1 : 10-bit mode | O | O |
| Reserved bit | | Must always be set to “0” | | |
| VCUT | Vref connect bit | 1 : Vref connected | O | O |
| OPA0 | External op-amp connection mode bit (Note 2) | b7 b6<br>0 0 : ANEX0 and ANEX1 are not used<br>0 1 : ANEX0 input is A-D converted<br>1 0 : ANEX1 input is A-D converted<br>1 1 : External op-amp connection mode | O | O |
| OPA1 | | | O | O |

Note 1: If the A-D control register is rewritten during A-D conversion, the conversion result is indeterminate.
Note 2: Neither ‘01’ nor ‘10’ can be selected with the external op-amp connection mode bit.

**Figure 1.17.6. A-D conversion register in single sweep mode**

### (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 1.17.5 shows the specifications of repeat sweep mode 0. Figure 1.17.7 shows the A-D control register in repeat sweep mode 0.

**Table 1.17.5. 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 | $AN_0$ and $AN_1$ (2 pins), $AN_0$ to $AN_3$ (4 pins), $AN_0$ to $AN_5$ (6 pins), or $AN_0$ to $AN_7$ (8 pins) |
| Reading of result of A-D converter | Read A-D register corresponding to selected pin (at any time) |

A-D control register 0 (Note)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | 1 | 1 | | | |

| Symbol | Address | When reset |
|---|---|---|
| ADCON0 | $03D6_{16}$ | $00000XXX_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| CH0 | Analog input pin select bit | Invalid in repeat sweep mode 0 | ○ | ○ |
| CH1 | | | ○ | ○ |
| CH2 | | | ○ | ○ |
| MD0 | A-D operation mode select bit 0 | b4 b3<br>1 1 : Repeat sweep mode 0 | ○ | ○ |
| MD1 | | | ○ | ○ |
| TRG | Trigger select bit | 0 : Software trigger<br>1 : $AD_{TRG}$ trigger | ○ | ○ |
| ADST | A-D conversion start flag | 0 : A-D conversion disabled<br>1 : A-D conversion started | ○ | ○ |
| CKS0 | Frequency select bit 0 | 0 : $f_{AD}/4$ is selected<br>1 : $f_{AD}/2$ is selected | ○ | ○ |

Note: If the A-D control register is rewritten during A-D conversion, the conversion result is indeterminate.

A-D control register 1 (Note 1)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | 1 | 0 | | 0 | | |

| Symbol | Address | When reset |
|---|---|---|
| ADCON1 | $03D7_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| SCAN0 | A-D sweep pin select bit | When single sweep and repeat sweep mode 0 are selected<br>b1 b0<br>0 0 : $AN_0$, $AN_1$ (2 pins)<br>0 1 : $AN_0$ to $AN_3$ (4 pins)<br>1 0 : $AN_0$ to $AN_5$ (6 pins)<br>1 1 : $AN_0$ to $AN_7$ (8 pins) | ○ | ○ |
| SCAN1 | | | ○ | ○ |
| MD2 | A-D operation mode select bit 1 | Set to “0” when this mode is selected | ○ | ○ |
| BITS | 8/10-bit mode select bit | 0 : 8-bit mode<br>1 : 10-bit mode | ○ | ○ |
| Reserved bit | | Must always be set to “0” | ○ | ○ |
| VCUT | Vref connect bit | 1 : Vref connected | ○ | ○ |
| OPA0 | External op-amp connection mode bit (Note 2) | b7 b6<br>0 0 : ANEX0 and ANEX1 are not used<br>0 1 : ANEX0 input is A-D converted<br>1 0 : ANEX1 input is A-D converted<br>1 1 : External op-amp connection mode | ○ | ○ |
| OPA1 | | | ○ | ○ |

Note 1: If the A-D control register is rewritten during A-D conversion, the conversion result is indeterminate.
Note 2: Neither “01” nor “10” can be selected with the external op-amp connection mode bit.

**Figure 1.17.7. A-D conversion register in repeat sweep mode 0**

### (5) Repeat sweep mode 1

In repeat sweep mode 1, all pins are used for A-D conversion with emphasis on the pin or pins selected using the A-D sweep pin select bit. Table 1.17.6 shows the specifications of repeat sweep mode 1. Figure 1.17.8 shows the A-D control register in repeat sweep mode 1.

**Table 1.17.6. 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<br>Example : $AN_0$ selected $AN_0 \rightarrow AN_1 \rightarrow AN_0 \rightarrow AN_2 \rightarrow AN_0 \rightarrow AN_3$, 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 ; $AN_0$ (1 pin), $AN_0$ and $AN_1$ (2 pins), $AN_0$ to $AN_2$ (3 pins), $AN_0$ to $AN_3$ (4 pins) |
| Reading of result of A-D converter | Read A-D register corresponding to selected pin (at any time) |

A-D control register 0 (Note)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | 1 | 1 | | | |

| Symbol | Address | When reset |
|---|---|---|
| ADCON0 | $03D6_{16}$ | $00000XXX_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| CH0 | Analog input pin select bit | Invalid in repeat sweep mode 1 | ○ | ○ |
| CH1 | | | ○ | ○ |
| CH2 | | | ○ | ○ |
| MD0 | A-D operation mode select bit 0 | b4 b3<br>1 1 : Repeat sweep mode 1 | ○ | ○ |
| MD1 | | | ○ | ○ |
| TRG | Trigger select bit | 0 : Software trigger<br>1 : $AD_{TRG}$ trigger | ○ | ○ |
| ADST | A-D conversion start flag | 0 : A-D conversion disabled<br>1 : A-D conversion started | ○ | ○ |
| CKS0 | Frequency select bit 0 | 0 : $f_{AD}/4$ is selected<br>1 : $f_{AD}/2$ is selected | ○ | ○ |

Note: If the A-D control register is rewritten during A-D conversion, the conversion result is indeterminate.

A-D control register 1 (Note 1)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | 1 | 0 | | 1 | | |

| Symbol | Address | When reset |
|---|---|---|
| ADCON1 | $03D7_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| SCAN0 | A-D sweep pin select bit | When repeat sweep mode 1 is selected<br>b1 b0<br>0 0 : $AN_0$ (1 pin)<br>0 1 : $AN_0$, $AN_1$ (2 pins)<br>1 0 : $AN_0$ to $AN_2$ (3 pins)<br>1 1 : $AN_0$ to $AN_3$ (4 pins) | ○ | ○ |
| SCAN1 | | | ○ | ○ |
| MD2 | A-D operation mode select bit 1 | Set to “1” when this mode is selected | ○ | ○ |
| BITS | 8/10-bit mode select bit | 0 : 8-bit mode<br>1 : 10-bit mode | ○ | ○ |
| Reserved bit | | Must always be set to “0” | ○ | ○ |
| VCUT | Vref connect bit | 1 : Vref connected | ○ | ○ |
| OPA0 | External op-amp connection mode bit (Note 2) | b7 b6<br>0 0 : ANEX0 and ANEX1 are not used<br>0 1 : ANEX0 input is A-D converted<br>1 0 : ANEX1 input is A-D converted<br>1 1 : External op-amp connection mode | ○ | ○ |
| OPA1 | | | ○ | ○ |

Note 1: If the A-D control register is rewritten during A-D conversion, the conversion result is indeterminate.
Note 2: Neither ‘01’ nor ‘10’ can be selected with the external op-amp connection mode bit.

**Figure 1.17.8. 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 $03D4_{16}$) to “1”. When sample and hold is selected, the rate of conversion of each pin increases. As a result, a 28 $\phi_{AD}$ cycle is achieved with 8-bit resolution and 33 $\phi_{AD}$ with 10-bit resolution. Sample and hold can be selected in all modes. However, in all modes, be sure to specify before starting A-D conversion whether sample and hold is to be used.

### (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 $03D7_{16}$) 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 $03D7_{16}$) 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 $03D7_{16}$) is “1” and bit 7 is “1”, input via $AN_0$ to $AN_7$ 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 1.17.9 is an example of how to connect the pins in external operation amp mode.

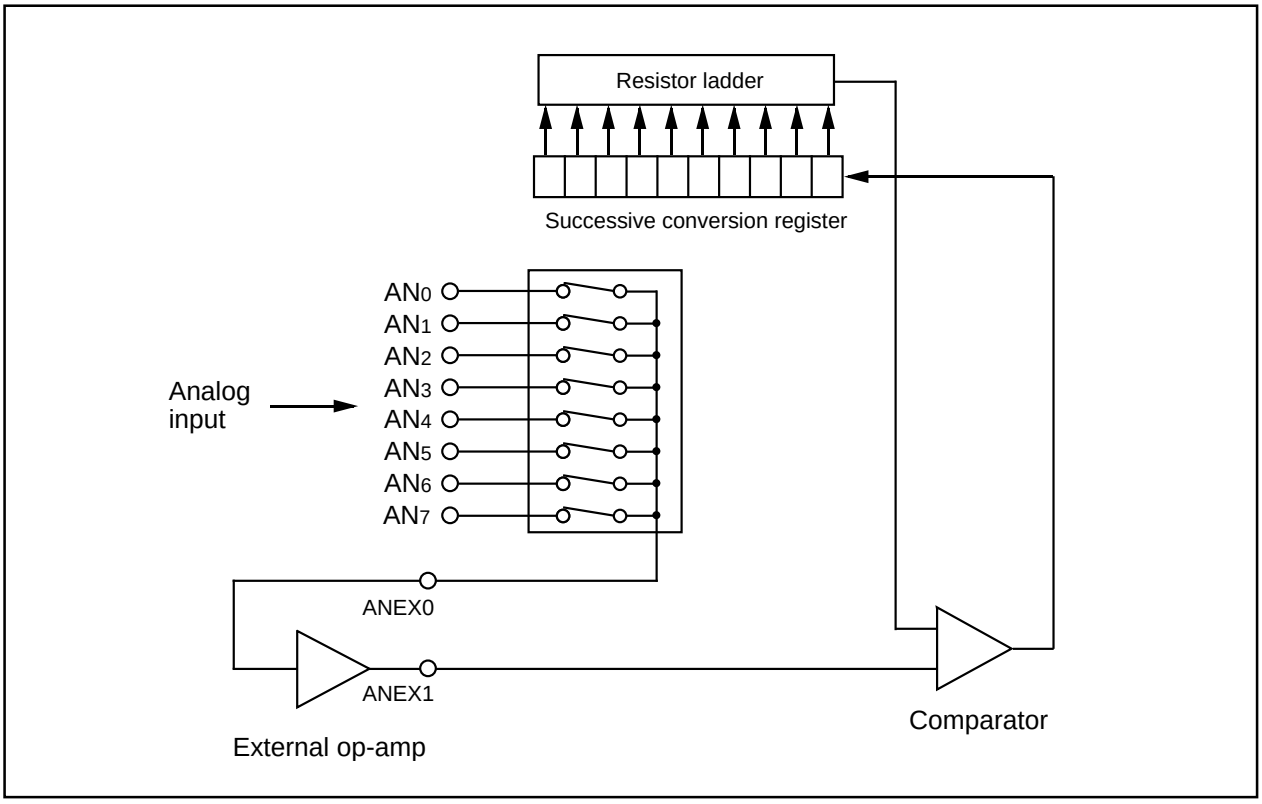



**Figure 1.17.9. Example of external op-amp connection mode**

## 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 1.18.1 lists the performance of the D-A converter. Figure 1.18.1 shows the block diagram of the D-A converter. Figure 1.18.2 shows the D-A control register. Figure 1.18.3 shows the D-A converter equivalent circuit.

**Table 1.18.1. Performance of D-A converter**

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

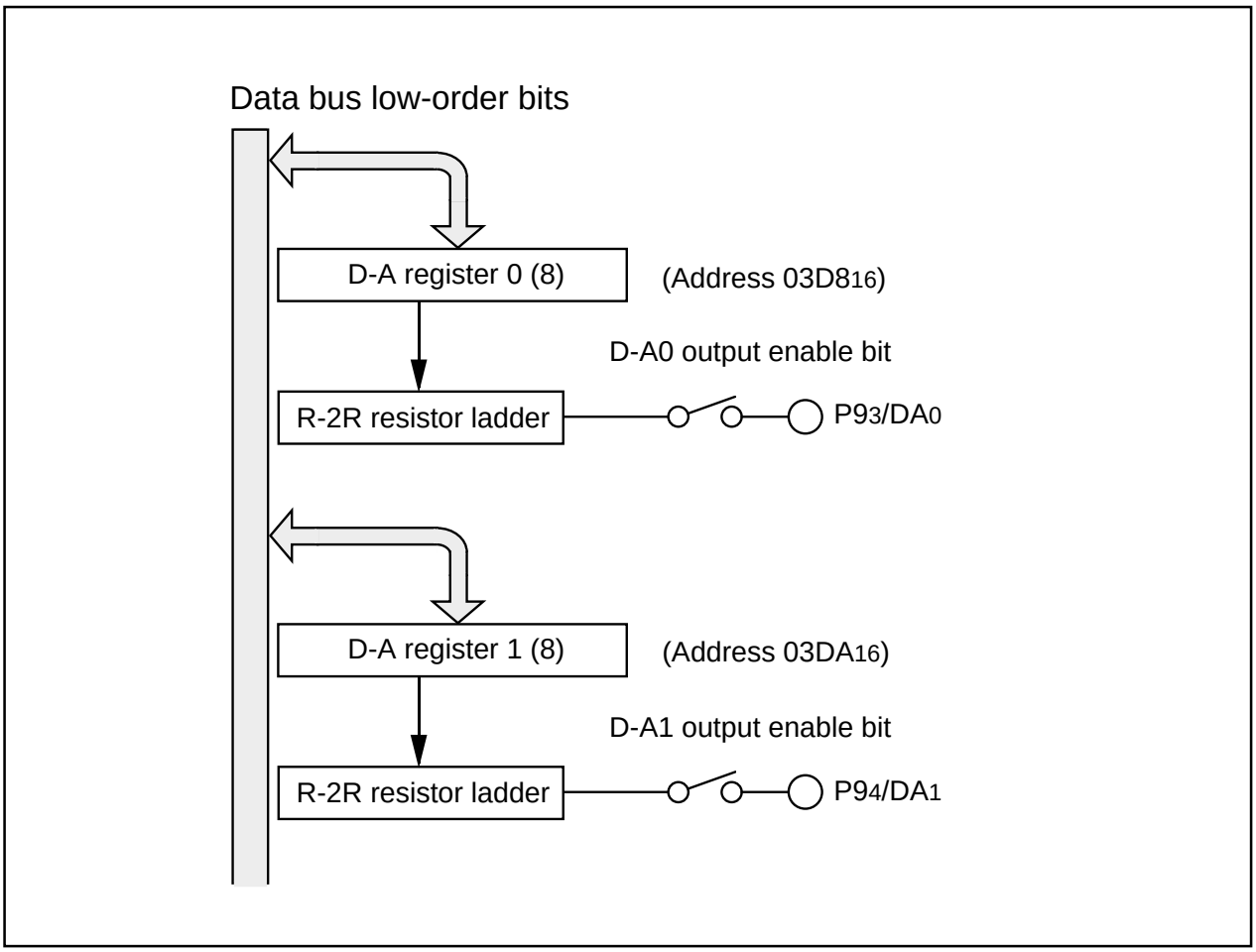



**Figure 1.18.1. Block diagram of D-A converter**

D-A control register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| ✕ | ✕ | ✕ | ✕ | ✕ | ✕ | | |

| Symbol | Address | When reset |
|---|---|---|
| DACON | $03DC_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| DA0E | D-A0 output enable bit | 0 : Output disabled<br>1 : Output enabled | ○ | ○ |
| DA1E | D-A1 output enable bit | 0 : Output disabled<br>1 : Output enabled | ○ | ○ |
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be “0” | | | — | — |

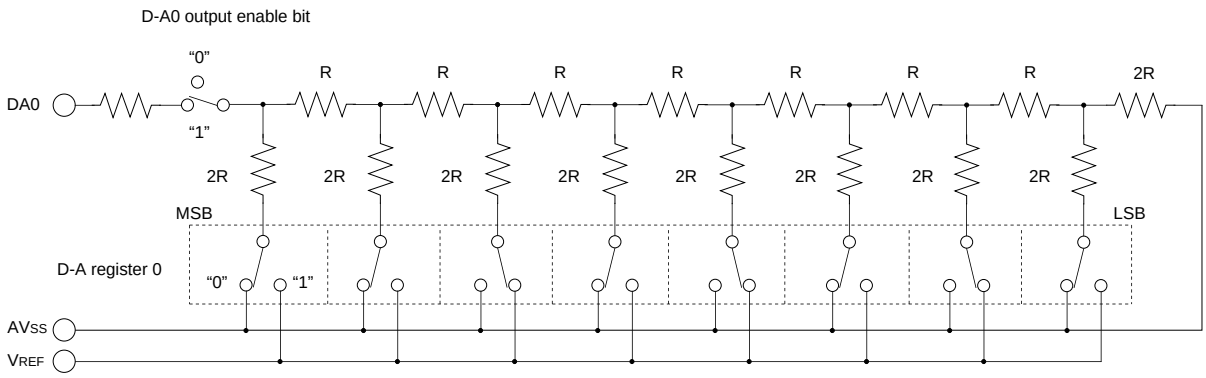
D-A register

| b7 | b0 |
|---|---|
| | |

| Symbol | Address | When reset |
|---|---|---|
| DAi (i = 0,1) | $03D8_{16}$, $03DA_{16}$ | Indeterminate |

| Function | R | W |
|---|---|---|
| Output value of D-A conversion | ○ | ○ |

**Figure 1.18.2. D-A control register**




Note 1: The above diagram shows an instance in which the D-A register is assigned $2A_{16}$.
Note 2: The same circuit as this is also used for D-A1.
Note 3: To reduce the current consumption when the D-A converter is not used, set the D-A output enable bit to 0 and set the D-A register to $00_{16}$ so that no current flows in the resistors Rs and 2Rs.

**Figure 1.18.3. D-A converter equivalent circuit**

## 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 1.19.1 shows the block diagram of the CRC circuit. Figure 1.19.2 shows the CRC-related registers. Figure 1.19.3 shows the calculation example using the CRC calculation circuit.

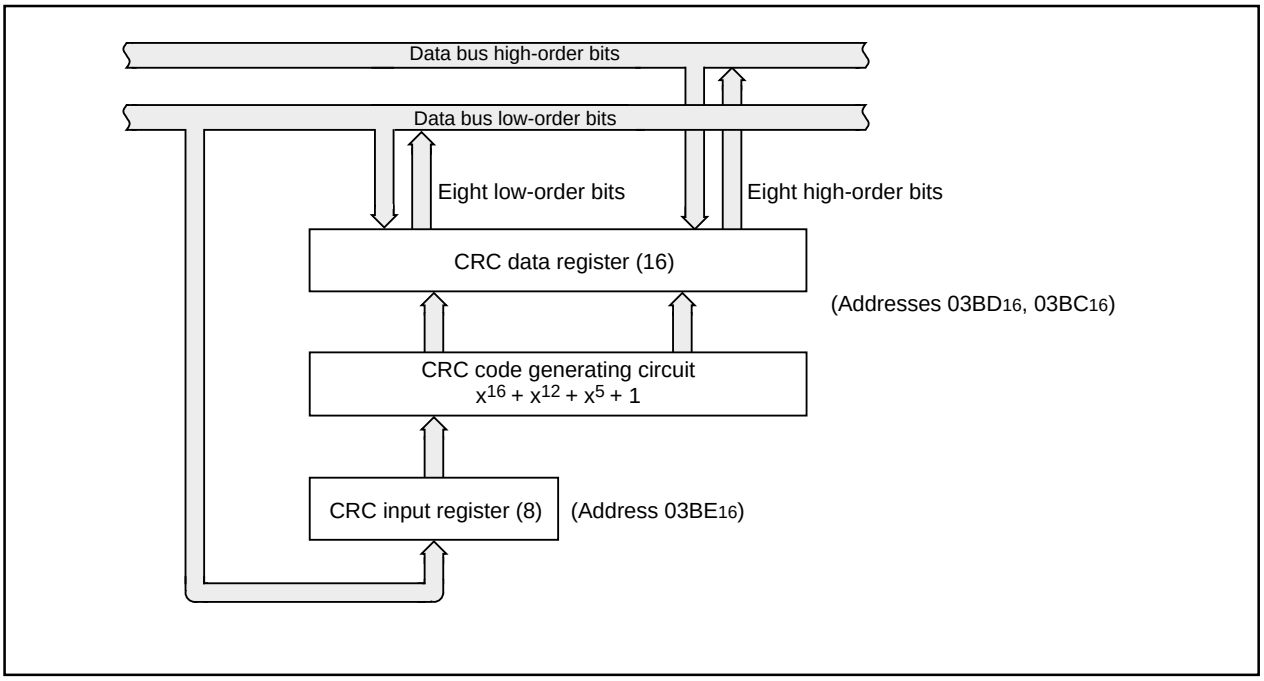



**Figure 1.19.1. Block diagram of CRC circuit**

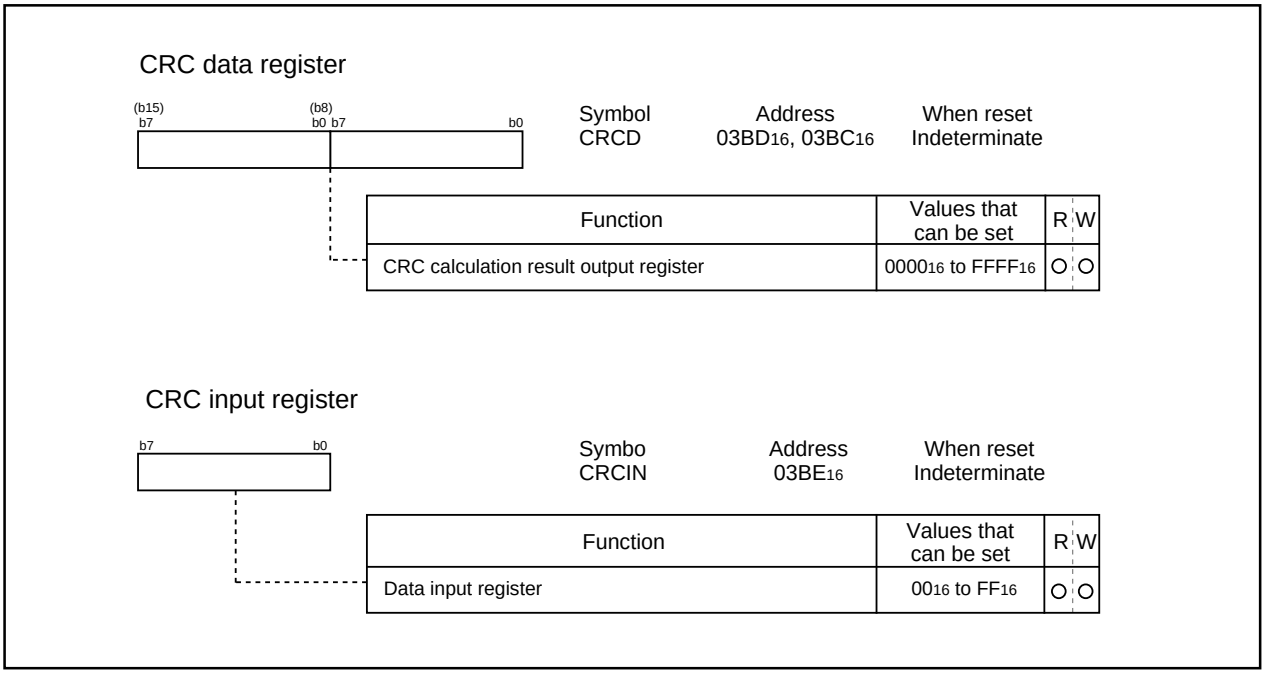
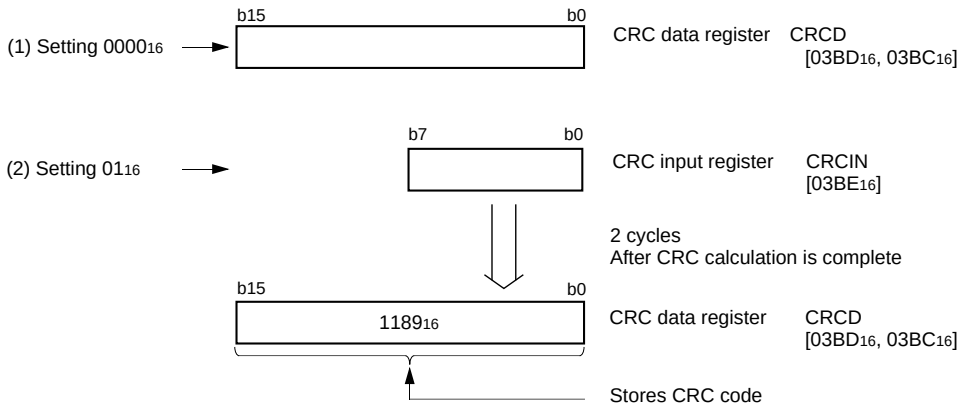



**Figure 1.19.2. CRC-related registers**




The code resulting from sending $01_{16}$ in LSB first mode is (1000 0000). Thus the CRC code in the generating polynomial, ($X^{16} + X^{12} + X^5 + 1$), becomes the remainder resulting from dividing (1000 0000) $X^{16}$ by (1 0001 0000 0010 0001) in conformity with the modulo-2 operation.

```
                                                  LSB                                  MSB
                                                                          1000 1000
1 0001 0000 0010 0001 ) 1000 0000 0000 0000 0000 0000
                        1000 1000 0001 0000 1
                                1000 0001 0000 1000 0
                                1000 1000 0001 0000 1
                                     1001 0001 1000 1000
                                LSB                       MSB
                                9    8    1    1
```

Modulo-2 operation is operation that complies with the law given below.

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0
-1 = 1

Thus the CRC code becomes (1001 0001 1000 1000). Since the operation is in LSB first mode, the (1001 0001 1000 1000) corresponds to $1189_{16}$ in hexadecimal notation. If the CRC operation in MSB first mode is necessary in the CRC operation circuit built in the M16C, switch between the LSB side and the MSB side of the input-holding bits, and carry out the CRC operation. Also switch between the MSB and LSB of the result as stored in CRC data.

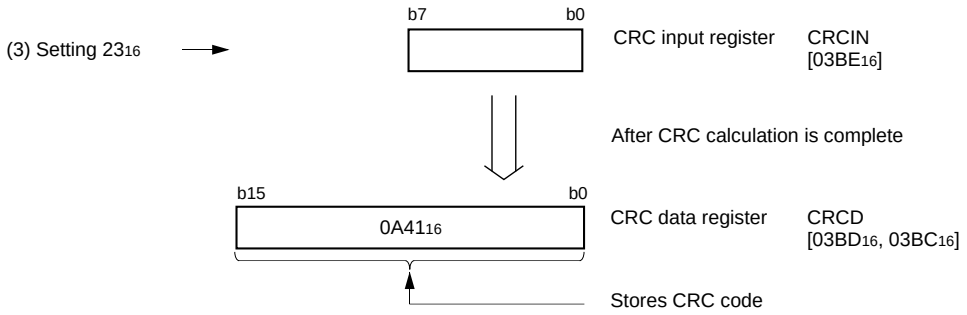



**Figure 1.19.3. Calculation example using the CRC calculation circuit**

## Programmable I/O Ports

There are 87 programmable I/O ports: P0 to P10 (excluding $P8_5$). Each port can be set independently for input or output using the direction register. A pull-up resistance for each block of 4 ports can be set. $P8_5$ is an input-only port and has no built-in pull-up resistance.

Figures 1.20.1 to 1.20.4 show the programmable I/O ports. Figure 1.20.5 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 1.20.6 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.

In memory expansion and microprocessor mode, the contents of corresponding direction register of pins $A_0$ to $A_{19}$, $D_0$ to $D_{15}$, $\overline{CS0}$ to $\overline{CS3}$, $\overline{RD}$, $\overline{WRL}$/$\overline{WR}$, $\overline{WRH}$/$\overline{BHE}$, ALE, $\overline{RDY}$, $\overline{HOLD}$, $\overline{HLDA}$ and BCLK cannot be modified.

Note: There is no direction register bit for $P8_5$.

### (2) Port registers

Figure 1.20.7 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.

In memory expansion and microprocessor mode, the contents of corresponding port register of pins $A_0$ to $A_{19}$, $D_0$ to $D_{15}$, $\overline{CS0}$ to $\overline{CS3}$, $\overline{RD}$, $\overline{WRL}$/$\overline{WR}$, $\overline{WRH}$/$\overline{BHE}$, ALE, $\overline{RDY}$, $\overline{HOLD}$, $\overline{HLDA}$ and BCLK cannot be modified.

### (3) Pull-up control registers

Figure 1.20.8 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.

However, in memory expansion mode and microprocessor mode, the pull-up control register of P0 to P3, $P4_0$ to $P4_3$, and P5 is invalid. The contents of register can be changed, but the pull-up resistance is not connected.

### (4) Port control register

Figure 1.20.9 shows the port control register.

The bit 0 of port control register is used to read port P1 as follows:

0 : When port P1 is input port, port input level is read.
When port P1 is output port , the contents of port P1 register is read.

1 : The contents of port P1 register is read always.

This register is valid in the following:

- External bus width is 8 bits in microprocessor mode or memory expansion mode.
- Port P1 can be used as a port in multiplexed bus for the entire space.

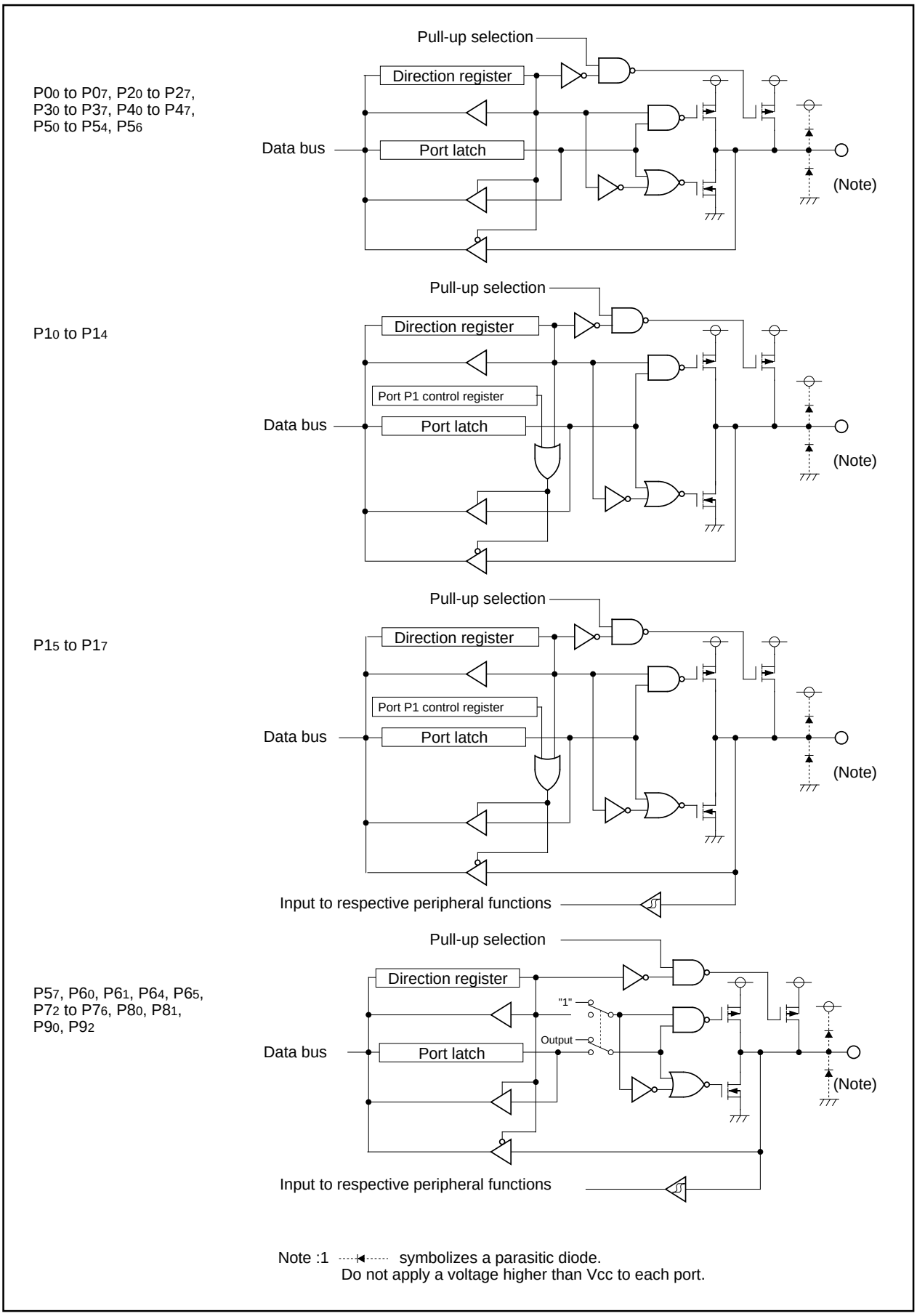
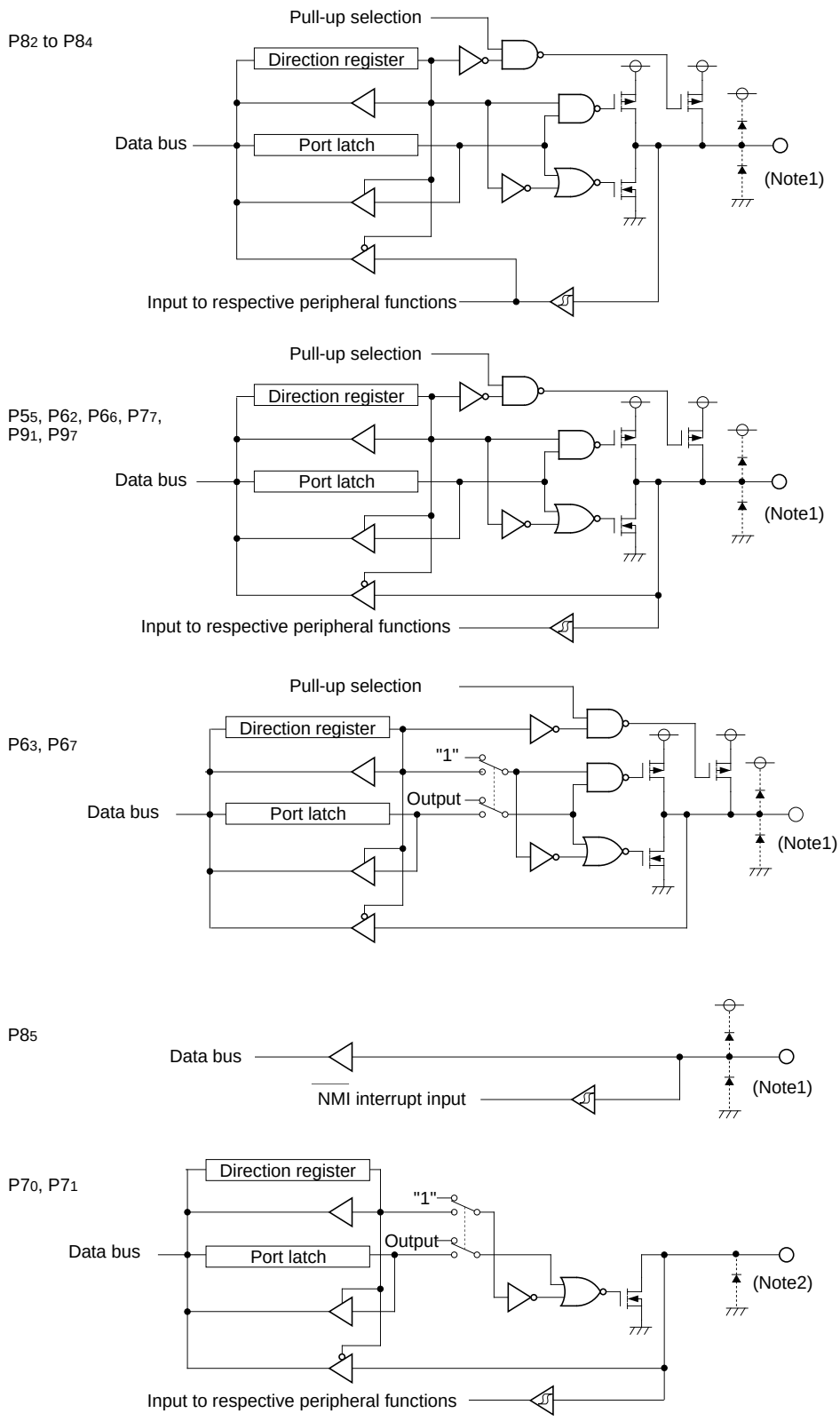



**Figure 1.20.1. Programmable I/O ports (1)**




Note :1 symbolizes a parasitic diode.
Do not apply a voltage higher than Vcc to each port.
Note :2 symbolizes a parasitic diode.

**Figure 1.20.2. Programmable I/O ports (2)**

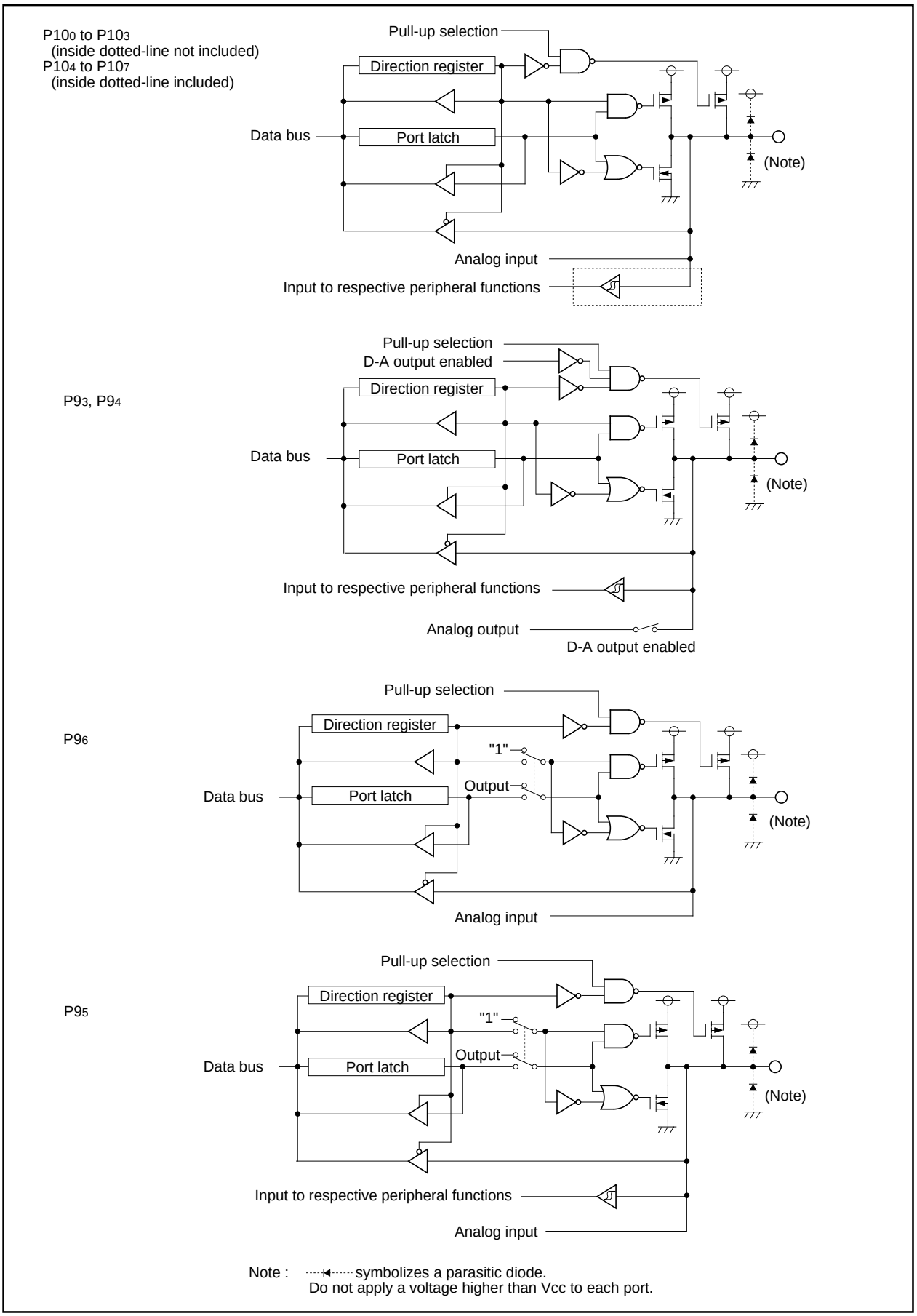



**Figure 1.20.3. Programmable I/O ports (3)**

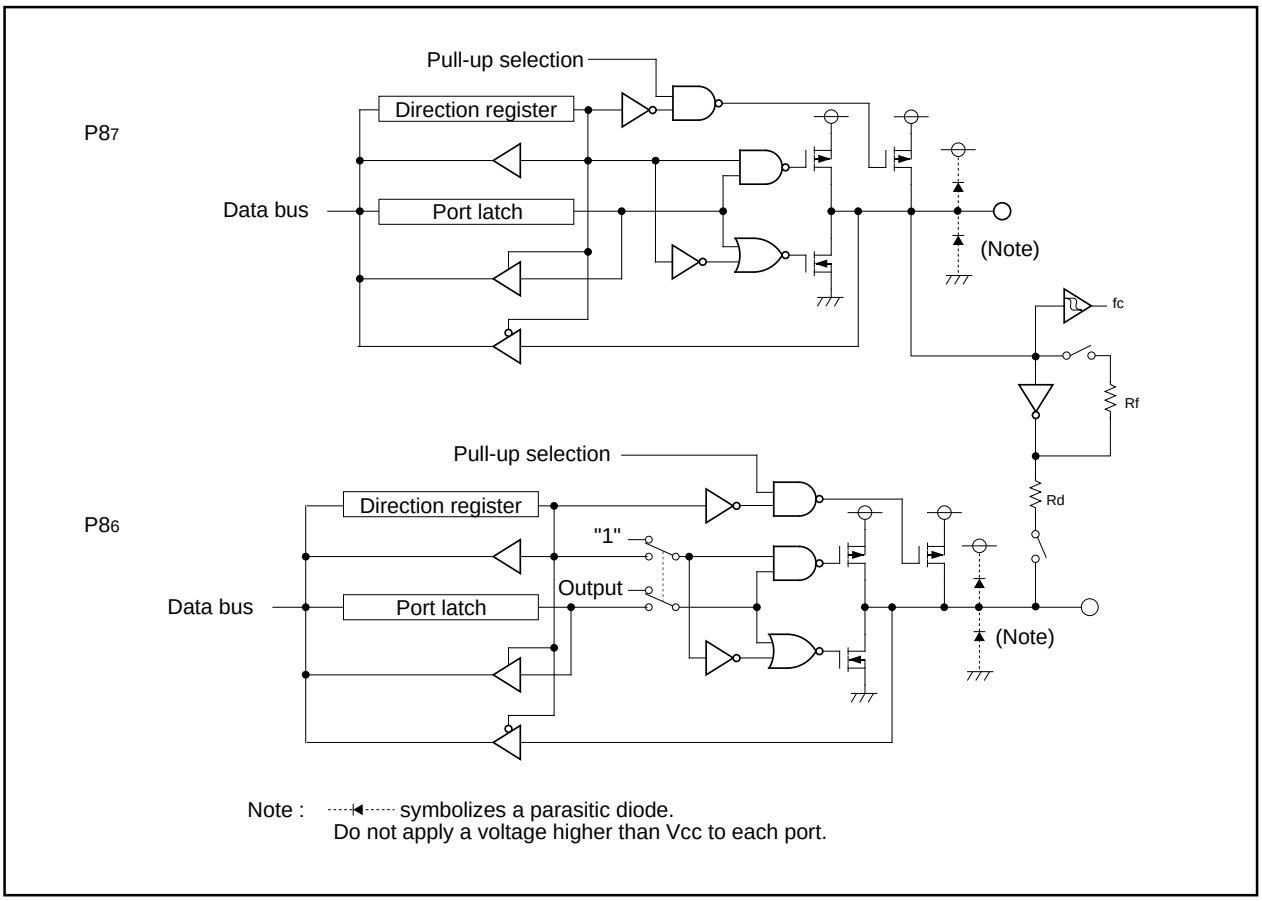



**Figure 1.20.4. Programmable I/O ports (4)**

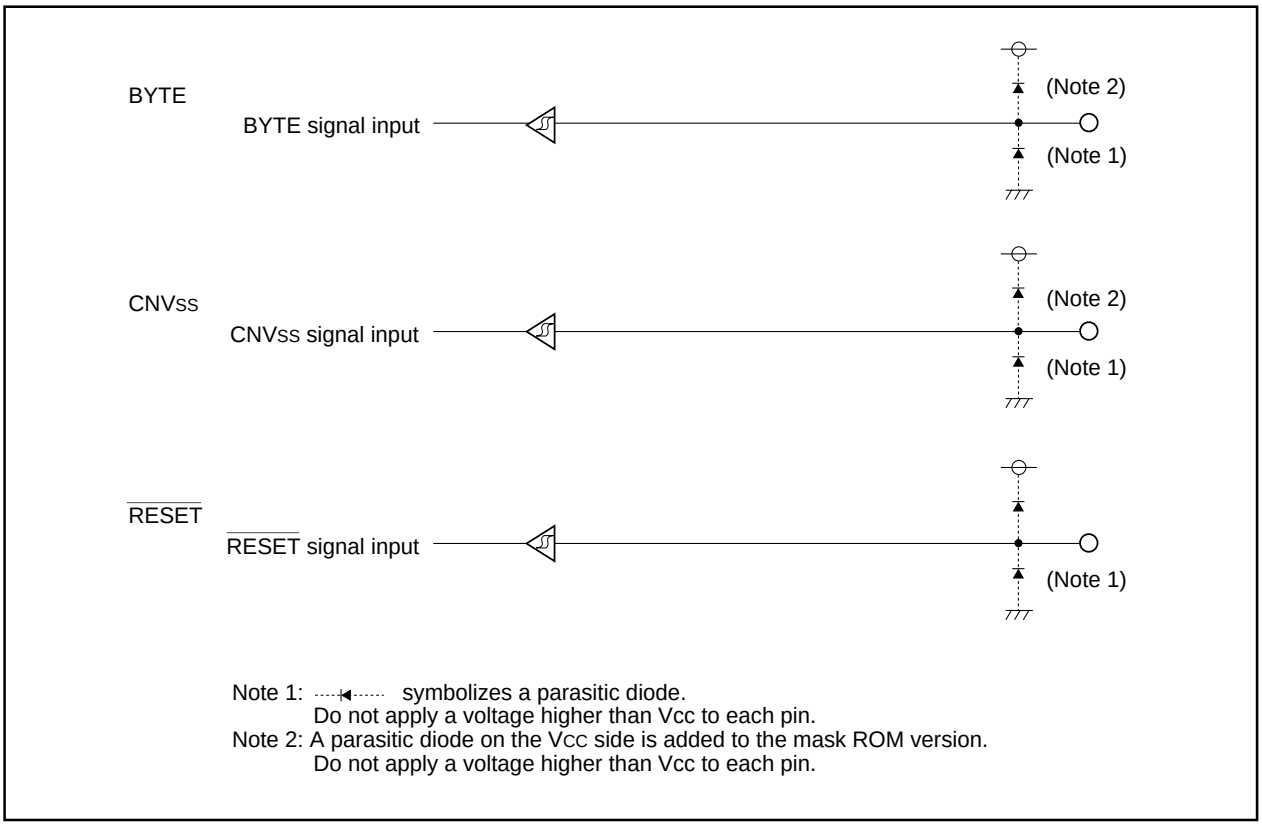



**Figure 1.20.5. I/O pins**

Port Pi direction register (Note 1, 2)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| PDi (i = 0 to 10, except 8) | $03E2_{16}$, $03E3_{16}$, $03E6_{16}$, $03E7_{16}$, $03EA_{16}$ $03EB_{16}$, $03EE_{16}$, $03EF_{16}$, $03F3_{16}$, $03F6_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| PDi_0 | Port $Pi_0$ direction register | 0 : Input mode (Functions as an input port) 1 : Output mode (Functions as an output port) (i = 0 to 10 except 8) | ○ | ○ |
| PDi_1 | Port $Pi_1$ direction register | | ○ | ○ |
| PDi_2 | Port $Pi_2$ direction register | | ○ | ○ |
| PDi_3 | Port $Pi_3$ direction register | | ○ | ○ |
| PDi_4 | Port $Pi_4$ direction register | | ○ | ○ |
| PDi_5 | Port $Pi_5$ direction register | | ○ | ○ |
| PDi_6 | Port $Pi_6$ direction register | | ○ | ○ |
| PDi_7 | Port $Pi_7$ direction register | | ○ | ○ |

Note 1: Set bit 2 of protect register (address $000A_{16}$) to “1” before rewriting to the port P9 direction register.
Note 2: In memory expansion and microprocessor mode, the contents of corresponding port Pi direction register of pins $A_0$ to $A_{19}$, $D_0$ to $D_{15}$, $CS_0$ to $CS_3$, RD, WRL/WR, WRH/BHE, ALE, RDY, HOLD, HLDA and BCLK cannot be modified.

Port P8 direction register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | ☒ | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| PD8 | $03F2_{16}$ | $00X0000_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| PD8_0 | Port $P8_0$ direction register | 0 : Input mode (Functions as an input port) 1 : Output mode (Functions as an output port) | ○ | ○ |
| PD8_1 | Port $P8_1$ direction register | | ○ | ○ |
| PD8_2 | Port $P8_2$ direction register | | ○ | ○ |
| PD8_3 | Port $P8_3$ direction register | | ○ | ○ |
| PD8_4 | Port $P8_4$ direction register | | ○ | ○ |
| Nothing is assigned. In an attempt to write to this bit, write “0”. The value, if read, turns out to be indeterminate. | | | — | — |
| PD8_6 | Port $P8_6$ direction register | 0 : Input mode (Functions as an input port) 1 : Output mode (Functions as an output port) | ○ | ○ |
| PD8_7 | Port $P8_7$ direction register | | ○ | ○ |

**Figure 1.20.6. Direction register**

Port Pi register (Note 2)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| Pi (i = 0 to 10, except 8) | 03E0$_{16}$, 03E1$_{16}$, 03E4$_{16}$, 03E5$_{16}$, 03E8$_{16}$ | Indeterminate |
| | 03E9$_{16}$, 03EC$_{16}$, 03ED$_{16}$, 03F1$_{16}$, 03F4$_{16}$ | Indeterminate |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| Pi_0 | Port Pi$_0$ register | Data is input and output to and from each pin by reading and writing to and from each corresponding bit<br>0 : “L” level data<br>1 : “H” level data (Note 1)<br><br>(i = 0 to 10 except 8) | ○ | ○ |
| Pi_1 | Port Pi$_1$ register | | ○ | ○ |
| Pi_2 | Port Pi$_2$ register | | ○ | ○ |
| Pi_3 | Port Pi$_3$ register | | ○ | ○ |
| Pi_4 | Port Pi$_4$ register | | ○ | ○ |
| Pi_5 | Port Pi$_5$ register | | ○ | ○ |
| Pi_6 | Port Pi$_6$ register | | ○ | ○ |
| Pi_7 | Port Pi$_7$ register | | ○ | ○ |

Note 1: Since P7$_0$ and P7$_1$ are N-channel open drain ports, the data is high-impedance.
Note 2: In memory expansion and microprocessor mode, the contents of corresponding port Pi register of pins A$_0$ to A$_{19}$, D$_0$ to D$_{15}$, $\overline{CS_0}$ to $\overline{CS_3}$, $\overline{RD}$, $\overline{WRL}$/$\overline{WR}$, $\overline{WRH}$/$\overline{BHE}$, ALE, $\overline{RDY}$, $\overline{HOLD}$, $\overline{HLDA}$ and BCLK cannot be modified.

Port P8 register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| P8 | 03F0$_{16}$ | Indeterminate |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| P8_0 | Port P8$_0$ register | Data is input and output to and from each pin by reading and writing to and from each corresponding bit (except for P8$_5$)<br>0 : “L” level data<br>1 : “H” level data | ○ | ○ |
| P8_1 | Port P8$_1$ register | | ○ | ○ |
| P8_2 | Port P8$_2$ register | | ○ | ○ |
| P8_3 | Port P8$_3$ register | | ○ | ○ |
| P8_4 | Port P8$_4$ register | | ○ | ○ |
| P8_5 | Port P8$_5$ register | | ○ | × |
| P8_6 | Port P8$_6$ register | | ○ | ○ |
| P8_7 | Port P8$_7$ register | | ○ | ○ |

**Figure 1.20.7. Port register**

Pull-up control register 0 (Note)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| PUR0 | $03FC_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| PU00 | $P0_0$ to $P0_3$ pull-up | The corresponding port is pulled high with a pull-up resistor<br>0 : Not pulled high<br>1 : Pulled high | ○ | ○ |
| PU01 | $P0_4$ to $P0_7$ pull-up | | ○ | ○ |
| PU02 | $P1_0$ to $P1_3$ pull-up | | ○ | ○ |
| PU03 | $P1_4$ to $P1_7$ pull-up | | ○ | ○ |
| PU04 | $P2_0$ to $P2_3$ pull-up | | ○ | ○ |
| PU05 | $P2_4$ to $P2_7$ pull-up | | ○ | ○ |
| PU06 | $P3_0$ to $P3_3$ pull-up | | ○ | ○ |
| PU07 | $P3_4$ to $P3_7$ pull-up | | ○ | ○ |

Note : In memory expansion and microprocessor mode, the content of this register can be changed, but the pull-up resistance is not connected.

Pull-up control register 1

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| PUR1 | $03FD_{16}$ | $00_{16}$ (Note 2) |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| PU10 | $P4_0$ to $P4_3$ pull-up (Note 3) | The corresponding port is pulled high with a pull-up resistor<br>0 : Not pulled high<br>1 : Pulled high | ○ | ○ |
| PU11 | $P4_4$ to $P4_7$ pull-up | | ○ | ○ |
| PU12 | $P5_0$ to $P5_3$ pull-up (Note 3) | | ○ | ○ |
| PU13 | $P5_4$ to $P5_7$ pull-up (Note 3) | | ○ | ○ |
| PU14 | $P6_0$ to $P6_3$ pull-up | | ○ | ○ |
| PU15 | $P6_4$ to $P6_7$ pull-up | | ○ | ○ |
| PU16 | $P7_2$ to $P7_3$ pull-up (Note 1) | | ○ | ○ |
| PU17 | $P7_4$ to $P7_7$ pull-up | | ○ | ○ |

Note 1: Since $P7_0$ and $P7_1$ are N-channel open drain ports, pull-up is not available for them.
Note 2: When the $V_{CC}$ level is being impressed to the $CNV_{SS}$ terminal, this register becomes to $02_{16}$ when reset (PU11 becomes to "1").
Note 3: In memory expansion and microprocessor mode, the content of these bits can be changed, but the pull-up resistance is not connected.

Pull-up control register 2

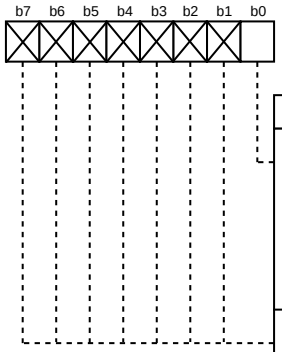
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| X | X | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| PUR2 | $03FE_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| PU20 | $P8_0$ to $P8_3$ pull-up | The corresponding port is pulled high with a pull-up resistor<br>0 : Not pulled high<br>1 : Pulled high | ○ | ○ |
| PU21 | $P8_4$ to $P8_7$ pull-up (Except $P8_5$) | | ○ | ○ |
| PU22 | $P9_0$ to $P9_3$ pull-up | | ○ | ○ |
| PU23 | $P9_4$ to $P9_7$ pull-up | | ○ | ○ |
| PU24 | $P10_0$ to $P10_3$ pull-up | | ○ | ○ |
| PU25 | $P10_4$ to $P10_7$ pull-up | | ○ | ○ |
| Nothing is assigned.<br>In an attempt to write to these bits, write "0". The value, if read, turns out to be "0". | | | — | — |
| | | | — | — |

**Figure 1.20.8. Pull-up control register**

Port control register




| Symbpl | Address | When reset |
|---|---|---|
| PCR | $03FF_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| PCR0 | Port P1 control register | 0 : When input port, read port input level. When output port, read the contents of port P1 register.<br>1 : Read the contents of port P1 register though input/output port. | ○ | ○ |
| Nothing is assigned.<br>In an attempt to write to these bits, write “0”. The value, if read, turns out to be “0”. | | | — | — |

**Figure 1.20.9. Port control register**

**Table 1.20.1. Example connection of unused pins in single-chip mode**

| Pin name | Connection |
|---|---|
| Ports P0 to P10 (excluding P8$_5$) | After setting for input mode, connect every pin to $V_{SS}$ via a resistor (pull-down); or after setting for output mode, leave these pins open. |
| $X_{OUT}$ (Note) | Open |
| $\overline{NMI}$ | Connect via resistor to $V_{CC}$ (pull-up) |
| $AV_{CC}$ | Connect to $V_{CC}$ |
| $AV_{SS}$, $V_{REF}$, BYTE | Connect to $V_{SS}$ |

Note: With external clock input to $X_{IN}$ pin.

**Table 1.20.2. Example connection of unused pins in memory expansion mode and microprocessor mode**

| Pin name | Connection |
|---|---|
| Ports P6 to P10 (excluding P8$_5$) | After setting for input mode, connect every pin to $V_{SS}$ via a resistor (pull-down); or after setting for output mode, leave these pins open. |
| P4$_5$ / $\overline{CS1}$ to P4$_7$ / $\overline{CS3}$ | Set ports to input mode, set output enable bits of $\overline{CS1}$ through $\overline{CS3}$ to 0, and connect to Vcc via resistors (pull-up). |
| $\overline{BHE}$, ALE, $\overline{HLDA}$, $X_{OUT}$ (Note 1), BCLK (Note 2) | Open |
| $\overline{HOLD}$, $\overline{RDY}$, $\overline{NMI}$ | Connect via resistor to $V_{CC}$ (pull-up) |
| $AV_{CC}$ | Connect to $V_{CC}$ |
| $AV_{SS}$, $V_{REF}$ | Connect to $V_{SS}$ |

Note 1: With external clock input to $X_{IN}$ pin.
Note 2: When the BCLK output disable bit (bit 7 at address $0004_{16}$) is set to “1”, connect to $V_{CC}$ via a resistor (pull-up).

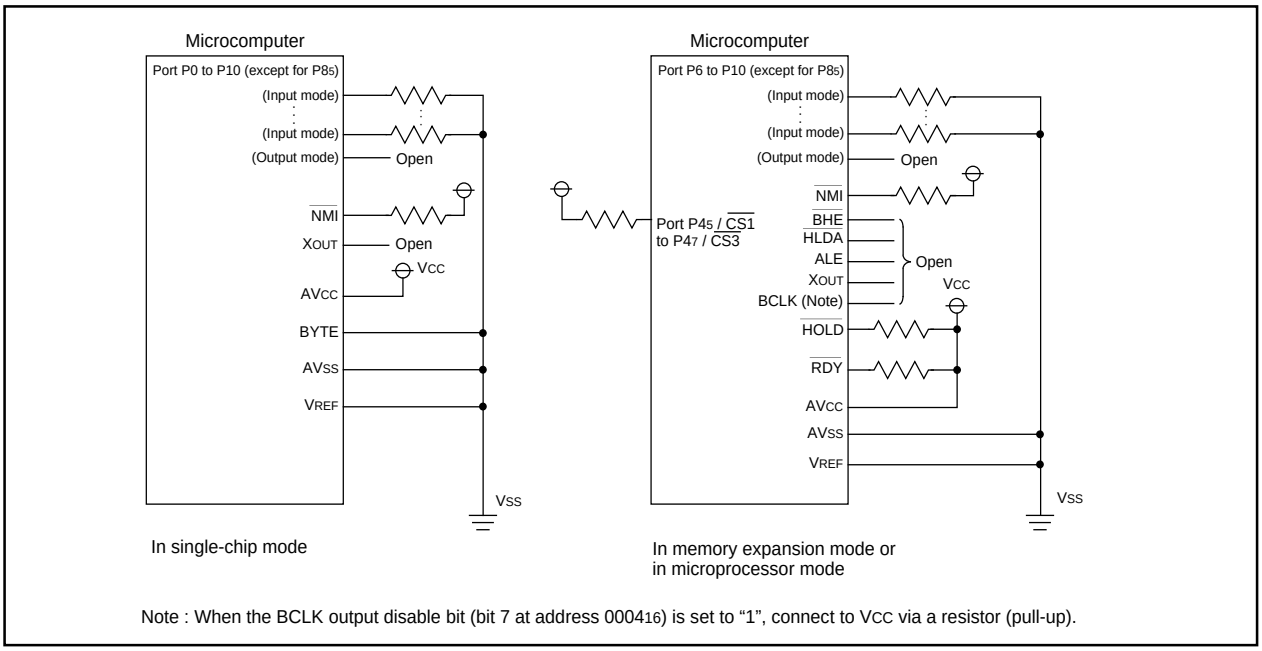



**Figure 1.20.10. Example connection of unused pins**

## Electrical characteristics

**Table 1.26.1. Absolute maximum ratings**

| Symbol | Parameter | | Condition | Rated value | Unit |
|---|---|---|---|---|---|
| $V_{CC}$ | Supply voltage | | $V_{CC}$=$AV_{CC}$ | - 0.3 to 4.2 | V |
| $AV_{CC}$ | Analog supply voltage | | $V_{CC}$=$AV_{CC}$ | - 0.3 to 4.2 | V |
| $V_I$ | Input voltage | $\overline{RESET}$, $CNV_{SS}$, BYTE, $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_2$ to $P7_7$, $P8_0$ to $P8_7$, $P9_0$ to $P9_7$, $P10_0$ to $P10_7$, $V_{REF}$, $X_{IN}$ | | - 0.3 to Vcc + 0.3 | V |
| | | $P7_0$, $P7_1$ | | - 0.3 to 4.2 | V |
| $V_O$ | Output voltage | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_2$ to $P7_7$, $P8_0$ to $P8_4$, $P8_6$, $P8_7$, $P9_0$ to $P9_7$, $P10_0$ to $P10_7$, $X_{OUT}$ | | - 0.3 to Vcc + 0.3 | V |
| | | $P7_0$, $P7_1$ | | - 0.3 to 4.2 | V |
| $P_d$ | Power dissipation | | Topr=25 °C | 300 | mW |
| $T_{opr}$ | Operating ambient temperature | | | - 20 to 85 / -40 to 85(Note) | °C |
| $T_{stg}$ | Storage temperature | | | - 65 to 150 | °C |

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

**Table 1.26.2. Recommended operating conditions (referenced to $V_{CC}$ = 2.7V to 3.6V at Topr = –20°C to 85°C / – 40°C to 85°C(Note3) unless otherwise specified)**

| Symbol | Parameter | | | Standard Min. | Standard Typ. | Standard Max. | Unit |
|---|---|---|---|---|---|---|---|
| Vcc | Supply voltage | | | 2.7 | 3.0 | 3.6 | V |
| AVcc | Analog supply voltage | | | | Vcc | | V |
| Vss | Supply voltage | | | | 0 | | V |
| AVss | Analog supply voltage | | | | 0 | | V |
| $V_{IH}$ | HIGH input voltage | $P3_1$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_2$ to $P7_7$, $P8_0$ to $P8_7$, $P9_0$ to $P9_7$, $P10_0$ to $P10_7$, $X_{IN}$, RESET, $CNV_{SS}$, BYTE | | 0.8Vcc | | Vcc | V |
| | | $P7_0$, $P7_1$ | | 0.8Vcc | | 4.2 | V |
| | | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ (during single-chip mode) | | 0.8Vcc | | Vcc | V |
| | | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ (data input function during memory expansion and microprocessor modes) | | 0.5Vcc | | Vcc | V |
| $V_{IL}$ | LOW input voltage | $P3_1$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_0$ to $P7_7$, $P8_0$ to $P8_7$, $P9_0$ to $P9_7$, $P10_0$ to $P10_7$, $X_{IN}$, RESET, $CNV_{SS}$, BYTE | | 0 | | 0.2Vcc | V |
| | | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ (during single-chip mode) | | 0 | | 0.2Vcc | V |
| | | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ (data input function during memory expansion and microprocessor modes) | | 0 | | 0.16Vcc | V |
| $I_{OH}$ (peak) | HIGH peak output current | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_2$ to $P7_7$, $P8_0$ to $P8_4$, $P8_6$, $P8_7$, $P9_0$ to $P9_7$, $P10_0$ to $P10_7$ | | | | - 10.0 | mA |
| $I_{OH}$ (avg) | HIGH average output current | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_2$ to $P7_7$, $P8_0$ to $P8_4$, $P8_6$, $P8_7$, $P9_0$ to $P9_7$, $P10_0$ to $P10_7$ | | | | - 5.0 | mA |
| $I_{OL}$ (peak) | LOW peak output current | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_0$ to $P7_7$, $P8_0$ to $P8_4$, $P8_6$, $P8_7$, $P9_0$ to $P9_7$, $P10_0$ to $P10_7$ | | | | 10.0 | mA |
| $I_{OL}$ (avg) | LOW average output current | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_0$ to $P7_7$, $P8_0$ to $P8_4$, $P8_6$, $P8_7$, $P9_0$ to $P9_7$, $P10_0$ to $P10_7$ | | | | 5.0 | mA |
| f ($X_{IN}$) | Main clock input oscillation frequency (Note4) | No wait | Vcc=2.7V to 3.6V | 0 | | 10 | MHz |
| | | with wait | Vcc=2.7V to 3.6V | 0 | | 10 | MHz |
| f ($X_{CIN}$) | Subclock oscillation frequency | | | | 32.768 | 50 | kHz |

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

Note 2: The total $I_{OL}$ (peak) for all ports must be 80mA max. The total $I_{OH}$ (peak) for all ports must be 80mA max.

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

Note 4: Execute case without wait, program / erase of flash memory by $V_{CC}$=3.0V to 3.6V and f(BCLK) ≤ 6.25 MHz. Execute case with wait, program / erase of flash memory by $V_{CC}$=3.0V to 3.6V and f(BCLK) ≤ 10.0 MHz.

**Table 1.26.3. Electrical characteristics (referenced to $V_{CC}$ = 2.7V to 3.6V, $V_{SS}$ = 0V at Topr = – 20°C to 85°C / – 40°C to 85°C(Note1), f($X_{IN}$) = 10MHz without wait unless otherwise specified)**

| Symbol | Parameter | | | Measuring condition | | Standard Min | Standard Typ. | Standard Max. | Unit |
|---|---|---|---|---|---|---|---|---|---|
| $V_{OH}$ | HIGH output voltage | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_2$ to $P7_7$, $P8_0$ to $P8_4$, $P8_6$,$P8_7$, $P9_0$ to $P9_7$, $P10_0$ to $P10_7$ | | $I_{OH}$=–1mA | | 2.5 | | | V |
| $V_{OH}$ | HIGH output voltage | $X_{OUT}$ | HIGHPOWER | $I_{OH}$=–0.1mA | | 2.5 | | | V |
| | | | LOWPOWER | $I_{OH}$=–50µA | | 2.5 | | | |
| | HIGH output voltage | $X_{COUT}$ | HIGHPOWER | With no load applied | | | 3.0 | | V |
| | | | LOWPOWER | With no load applied | | | 1.6 | | |
| $V_{OL}$ | LOW output voltage | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_0$ to $P7_7$, $P8_0$ to $P8_4$, $P8_6$,$P8_7$, $P9_0$ to $P9_7$, $P10_0$ to $P10_7$ | | $I_{OL}$=1mA | | | | 0.5 | V |
| $V_{OL}$ | LOW output voltage | $X_{OUT}$ | HIGHPOWER | $I_{OL}$=0.1mA | | | | 0.5 | V |
| | | | LOWPOWER | $I_{OL}$=50µA | | | | 0.5 | |
| | LOW output voltage | $X_{COUT}$ | HIGHPOWER | With no load applied | | | 0 | | V |
| | | | LOWPOWER | With no load applied | | | 0 | | |
| $V_{T+}$–$V_{T-}$ | Hysteresis | $\overline{HOLD}$, $\overline{RDY}$, $TA0_{IN}$ to $TA4_{IN}$, $TB0_{IN}$ to $TB5_{IN}$, $\overline{INT_0}$ to $\overline{INT_5}$, $\overline{NMI}$, $\overline{AD_{TRG}}$, $\overline{CTS_0}$ to $\overline{CTS_2}$, SCL, SDA, $CLK_0$ to $CLK_4$, $TA2_{OUT}$ to $TA4_{OUT}$, $\overline{KI_0}$ to $\overline{KI_3}$, $RxD_0$ to $RxD_2$, $S_{IN3}$, $S_{IN4}$ | | | | 0.2 | | 0.8 | V |
| $V_{T+}$–$V_{T-}$ | Hysteresis | $\overline{RESET}$ | | | | 0.2 | | 1.8 | V |
| $I_{IH}$ | HIGH input current | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_0$ to $P7_7$, $P8_0$ to $P8_7$, $P9_0$ to $P9_7$, $P10_0$ to $P10_7$, $X_{IN}$, $\overline{RESET}$, $CNV_{SS}$, BYTE | | $V_I$=3V | | | | 4.0 | µA |
| $I_{IL}$ | LOW input current | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_0$ to $P7_7$, $P8_0$ to $P8_7$, $P9_0$ to $P9_7$, $P10_0$ to $P10_7$, $X_{IN}$, $\overline{RESET}$, $CNV_{SS}$, BYTE | | $V_I$=0V | | | | –4.0 | µA |
| $R_{PULLUP}$ | Pull-up resistance | $P0_0$ to $P0_7$, $P1_0$ to $P1_7$, $P2_0$ to $P2_7$, $P3_0$ to $P3_7$, $P4_0$ to $P4_7$, $P5_0$ to $P5_7$, $P6_0$ to $P6_7$, $P7_2$ to $P7_7$, $P8_0$ to $P8_4$, $P8_6$,$P8_7$, $P9_0$ to $P9_7$, $P10_0$ to $P10_7$ | | $V_I$=0V | | 20 | 120 | 500 | kΩ |
| $Rf_{XIN}$ | Feedback resistance | $X_{IN}$ | | | | | 3.0 | | MΩ |
| $Rf_{XCIN}$ | Feedback resistance | $X_{CIN}$ | | | | | 10.0 | | MΩ |
| $V_{RAM}$ | RAM retention voltage | | | When clock is stopped | | 2.0 | | | V |
| $I_{CC}$ | Power supply current | In single-chip mode, the output pins are open and other pins are $V_{SS}$ | | Flash memory 3V version | f($X_{IN}$)=10MHz Square wave, no division | | 10.0 | 21.25 | mA |
| | | | | Flash memory 3V version | f($X_{CIN}$)=32kHz Square wave | | 225 | | µA |
| | | | | Flash memory 3V version program | f($X_{IN}$)=10MHz Square wave, division by 2 | | 19.0 | | mA |
| | | | | Flash memory 3V version erase | f($X_{IN}$)=10MHz Square wave, division by 2 | | 21.0 | | mA |
| | | | | Flash memory 3V version | f($X_{CIN}$)=32kHz When a WAITinstruction is executed. Oscillation capacity High (Note 2) | | 5.8 | | µA |
| | | | | | f($X_{CIN}$)=32kHz When a WAITinstruction is executed. Oscillation capacity Low (Note 2) | | 2.7 | | µA |
| | | | | | When clock is stopped Topr=25°C | | | 2.0 | µA |
| | | | | | When clock is stopped Topr=85°C | | | 40.0 | |

Note 1: Specify a product of -40°C to 85°C to use it.
Note 2: With one timer operated using fC32.

**Table 1.26.4. A-D conversion characteristics (referenced to $V_{CC}$ = $AV_{CC}$ = $V_{REF}$ = 2.7V to 3.6V, $V_{SS}$ = $AV_{SS}$ = 0V, at Topr = – 20°C to 85°C / – 40°C to 85°C (Note2), f($X_{IN}$)=10MHz 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 (8 bit) | $V_{REF}$ =$V_{CC}$=3V, $\phi_{AD}$=$f_{AD}$/2 | | | ±2 | LSB |
| | | Sample & hold function not available (10 bit) | $V_{REF}$ =$V_{CC}$=3V, $\phi_{AD}$=$f_{AD}$/2 | | | ±8 | LSB |
| $R_{LADDER}$ | Ladder resistance | | $V_{REF}$ =$V_{CC}$ | 10 | | 40 | kΩ |
| $t_{CONV}$ | Conversion time (8bit) | | | 9.8 | | | µs |
| | Conversion time (10bit) | | | 11.8 | | | µs |
| $V_{REF}$ | Reference voltage | | | 2.4 | | $V_{CC}$ | V |
| $V_{IA}$ | Analog input voltage | | | 0 | | $V_{REF}$ | V |

Note 1: Connect $AV_{CC}$ pin to $V_{CC}$ pin and apply the same electric potential.

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

Note 3: Divide the $f_{AD}$ and make AD operation clock frequency ($\phi_{AD}$) equal to or lower than $f_{AD}$/2. A case without sample & hold function turn AD operation clock frequency ($\phi_{AD}$) into 250 kHz or more.

**Table 1.26.5. D-A conversion characteristics (referenced to $V_{CC}$ = 2.7V to 3.6V, $V_{SS}$ = $AV_{SS}$ = 0V, $V_{REF}$=3V, at Topr = – 20°C to 85°C / – 40°C to 85°C (Note2), f($X_{IN}$)=10MHz unless otherwise specified)**

| Symbol | Parameter | Measuring condition | Standard | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max | |
| – | Resolution | | | | 8 | Bits |
| – | Absolute accuracy | | | | 1.0 | % |
| $t_{su}$ | Setup time | | | | 3 | µs |
| $R_O$ | Output resistance | | 4 | 15 | 25 | kΩ |
| $I_{VREF}$ | Reference power supply input current | (Note1) | | | 1.0 | mA |

Note 1: This applies when using one D-A converter, with the D-A register for the unused D-A converter set to “$00_{16}$”. The A-D converter's ladder resistance is not included.

Also, when D-A register contents are not “$00_{16}$”, the current $I_{VREF}$ 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°C to 85°C to use it.

**Table 1.26.6. Flash memory version electrical characteristics (referenced to $V_{CC}$ = 3.0V to 3.6V, at Topr =0 to 60°C unless otherwise specified)**

| Parameter | Standard | | | Unit |
|---|---|---|---|---|
| | Min. | Typ. | Max. | |
| 32K bytes block erase time | | 0.3 | 8 | s |
| Chip erase time | | 2 | 8 x Number of blocks to erase | s |
| 64K bytes area word program time | | 0.5 | 5 | s |

Note: The number of blocks to erase is affected by $\overline{WP}$ and the lock bit.

**Table 1.26.7. Flash memory version program voltage and read operation voltage characteristics (at Topr = 0°C to 60°C)**

| Flash program voltage | Flash read operation voltage |
|---|---|
| $V_{CC}$=3.0V to 3.6V | $V_{CC}$=2.7V to 3.6V |

**Timing requirements**

**(referenced to $V_{CC}$ = 3V, $V_{SS}$ = 0V, at Topr = – 20°C to 85°C / – 40°C to 85°C (*) unless otherwise specified)**

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

**Table 1.26.8. 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 |

**Table 1.26.9. Memory expansion and microprocessor modes**

| Symbol | Parameter | Standard | | Unit |
|---|---|---|---|---|
| | | Min. | Max. | |
| $t_{ac1(RD-DB)}$ | Data input access time (no wait) | | (Note) | ns |
| $t_{ac2(RD-DB)}$ | Data input access time (with wait) | | (Note) | ns |
| $t_{ac3(RD-DB)}$ | Data input access time (when accessing multiplex bus area) | | (Note) | ns |
| $t_{su(DB-RD)}$ | Data input setup time | 80 | | ns |
| $t_{su(RDY-BCLK)}$ | $\overline{RDY}$ input setup time | 60 | | ns |
| $t_{su(HOLD-BCLK)}$ | $\overline{HOLD}$ input setup time | 80 | | ns |
| $t_{h(RD-DB)}$ | Data input hold time | 0 | | ns |
| $t_{h(BCLK-RDY)}$ | $\overline{RDY}$ input hold time | 0 | | ns |
| $t_{h(BCLK-HOLD)}$ | $\overline{HOLD}$ input hold time | 0 | | ns |
| $t_{d(BCLK-HLDA)}$ | $\overline{HLDA}$ output delay time | | 100 | ns |

Note: Calculated according to the BCLK frequency as follows:

$$t_{ac1(RD-DB)} = \frac{10^9}{f(BCLK) \times 2} - 90 \quad [ns]$$

$$t_{ac2(RD-DB)} = \frac{3 \times 10^9}{f(BCLK) \times 2} - 90 \quad [ns]$$

$$t_{ac3(RD-DB)} = \frac{3 \times 10^9}{f(BCLK) \times 2} - 90 \quad [ns]$$

**Timing requirements**

**(referenced to $V_{CC}$ = 3V, $V_{SS}$ = 0V, at Topr = – 20°C to 85°C / – 40°C to 85°C (*) unless otherwise specified)**

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

**Table 1.26.10. Timer A input (counter input in event counter mode)**

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

**Table 1.26.11. Timer A input (gating input in timer mode)**

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

**Table 1.26.12. Timer A input (external trigger input in one-shot timer mode)**

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

**Table 1.26.13. Timer A input (external trigger input in pulse width modulation mode)**

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

**Table 1.26.14. Timer A input (up/down input in event counter mode)**

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

**Timing requirements**

**(referenced to $V_{CC}$ = 3V, $V_{SS}$ = 0V, at Topr = – 20°C to 85°C / – 40°C to 85°C (*) unless otherwise specified)**

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

**Table 1.26.15. Timer B input (counter input in event counter mode)**

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

**Table 1.26.16. Timer B input (pulse period measurement mode)**

| Symbol | Parameter | Standard | | Unit |
|---|---|---|---|---|
| | | Min. | Max. | |
| $t_{c(TB)}$ | $TBi_{IN}$ input cycle time | 600 | | ns |
| $t_{w(TBH)}$ | $TBi_{IN}$ input HIGH pulse width | 300 | | ns |
| $t_{w(TBL)}$ | $TBi_{IN}$ input LOW pulse width | 300 | | ns |

**Table 1.26.17. Timer B input (pulse width measurement mode)**

| Symbol | Parameter | Standard | | Unit |
|---|---|---|---|---|
| | | Min. | Max. | |
| $t_{c(TB)}$ | $TBi_{IN}$ input cycle time | 600 | | ns |
| $t_{w(TBH)}$ | $TBi_{IN}$ input HIGH pulse width | 300 | | ns |
| $t_{w(TBL)}$ | $TBi_{IN}$ input LOW pulse width | 300 | | ns |

**Table 1.26.18. A-D trigger input**

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

**Table 1.26.19. 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 1.26.20. 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 |

**Switching characteristics (referenced to $V_{CC}$ = 3V, $V_{SS}$ = 0V at Topr = – 20°C to 85°C / – 40°C to 85°C (Note 3), CM15 = "1" unless otherwise specified)**

**Table 1.26.21. Memory expansion and microprocessor modes (with no wait)**

| Symbol | Parameter | Measuring condition | Standard | | Unit |
|---|---|---|---|---|---|
| | | | Min. | Max. | |
| $t_{d(BCLK-AD)}$ | Address output delay time | Figure 1.26.1 | | 60 | ns |
| $t_{h(BCLK-AD)}$ | Address output hold time (BCLK standard) | | 4 | | ns |
| $t_{h(RD-AD)}$ | Address output hold time (RD standard) | | 0 | | ns |
| $t_{h(WR-AD)}$ | Address output hold time (WR standard) | | 0 | | ns |
| $t_{d(BCLK-CS)}$ | Chip select output delay time | | | 60 | ns |
| $t_{h(BCLK-CS)}$ | Chip select output hold time (BCLK standard) | | 4 | | ns |
| $t_{d(BCLK-ALE)}$ | ALE signal output delay time | | | 60 | ns |
| $t_{h(BCLK-ALE)}$ | ALE signal output hold time | | – 4 | | ns |
| $t_{d(BCLK-RD)}$ | RD signal output delay time | | | 60 | ns |
| $t_{h(BCLK-RD)}$ | RD signal output hold time | | 0 | | ns |
| $t_{d(BCLK-WR)}$ | WR signal output delay time | | | 60 | ns |
| $t_{h(BCLK-WR)}$ | WR signal output hold time | | 0 | | ns |
| $t_{d(BCLK-DB)}$ | Data output delay time (BCLK standard) | | | 80 | ns |
| $t_{h(BCLK-DB)}$ | Data output hold time (BCLK standard) | | 4 | | ns |
| $t_{d(DB-WR)}$ | Data output delay time (WR standard) | | (Note1) | | ns |
| $t_{h(WR-DB)}$ | Data output hold time (WR standard)(Note2) | | 0 | | ns |

Note 1: Calculated according to the BCLK frequency as follows:

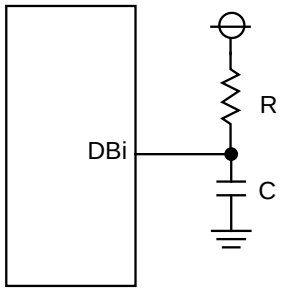
$$td(DB - WR) = \frac{10^9}{f(BCLK) \times 2} - 80 \quad [ns]$$

Note 2: This is standard value shows the timing when the output is off, and doesn't show hold time of data bus.
Hold time of data bus is different by capacitor volume and pull-up (pull-down) resistance value.
Hold time of data bus is expressed in

$$t = -CR \times \ln(1 - V_{OL} / V_{CC})$$

by a circuit of the right figure.
For example, when $V_{OL}$ = 0.2$V_{CC}$, C = 30pF, R = 1kΩ, hold time of output "L" level is

$$t = -30pF \times 1k\Omega \times \ln(1 - 0.2V_{CC} / V_{CC})$$
$$= 6.7ns.$$




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

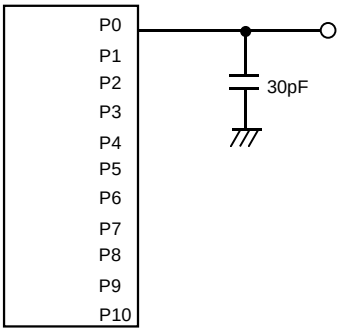



**Figure 1.26.1. Port P0 to P10 measurement circuit**

**Switching characteristics (referenced to $V_{CC}$ = 3V, $V_{SS}$ = 0V at Topr = – 20°C to 85°C / – 40°C to 85°C (Note 3), CM15 = "1" unless otherwise specified)**

**Table 1.26.22. Memory expansion and microprocessor modes (when accessing external memory area with wait)**

| Symbol | Parameter | Measuring condition | Standard | | Unit |
|---|---|---|---|---|---|
| | | | Min. | Max. | |
| $t_{d(BCLK-AD)}$ | Address output delay time | Figure 1.26.1 | | 60 | ns |
| $t_{h(BCLK-AD)}$ | Address output hold time (BCLK standard) | | 4 | | ns |
| $t_{h(RD-AD)}$ | Address output hold time (RD standard) | | 0 | | ns |
| $t_{h(WR-AD)}$ | Address output hold time (WR standard) | | 0 | | ns |
| $t_{d(BCLK-CS)}$ | Chip select output delay time | | | 60 | ns |
| $t_{h(BCLK-CS)}$ | Chip select output hold time (BCLK standard) | | 4 | | ns |
| $t_{d(BCLK-ALE)}$ | ALE signal output delay time | | | 60 | ns |
| $t_{h(BCLK-ALE)}$ | ALE signal output hold time | | – 4 | | ns |
| $t_{d(BCLK-RD)}$ | RD signal output delay time | | | 60 | ns |
| $t_{h(BCLK-RD)}$ | RD signal output hold time | | 0 | | ns |
| $t_{d(BCLK-WR)}$ | WR signal output delay time | | | 60 | ns |
| $t_{h(BCLK-WR)}$ | WR signal output hold time | | 0 | | ns |
| $t_{d(BCLK-DB)}$ | Data output delay time (BCLK standard) | | | 80 | ns |
| $t_{h(BCLK-DB)}$ | Data output hold time (BCLK standard) | | 4 | | ns |
| $t_{d(DB-WR)}$ | Data output delay time (WR standard) | | (Note1) | | ns |
| $t_{h(WR-DB)}$ | Data output hold time (WR standard)(Note2) | | 0 | | ns |

Note 1: Calculated according to the BCLK frequency as follows:

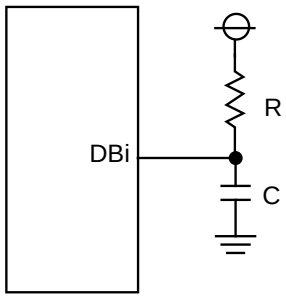
$$td(DB - WR) = \frac{10^9}{f(BCLK)} - 80 \quad [ns]$$

Note 2: This is standard value shows the timing when the output is off, and doesn't show hold time of data bus.
Hold time of data bus is different by capacitor volume and pull-up (pull-down) resistance value.
Hold time of data bus is expressed in

$$t = -CR \times \ln(1 - V_{OL} / V_{CC})$$

by a circuit of the right figure.
For example, when $V_{OL}$ = 0.2$V_{CC}$, C = 30pF, R = 1kΩ, hold time of output "L" level is

$$t = -30pF \times 1k\Omega \times \ln(1 - 0.2V_{CC} / V_{CC})$$
$$= 6.7ns.$$




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

**Switching characteristics (referenced to $V_{CC}$ = 3V, $V_{SS}$ = 0V at Topr = – 20°C to 85°C / – 40°C to 85°C (Note 2), CM15 = "1" unless otherwise specified)**

**Table 1.26.23. Memory expansion and microprocessor modes (when accessing external memory area with wait, and select multiplexed bus)**

| Symbol | Parameter | Measuring condition | Standard | | Unit |
|---|---|---|---|---|---|
| | | | Min. | Max. | |
| $t_{d(BCLK-AD)}$ | Address output delay time | Figure 1.26.1 | | 60 | ns |
| $t_{h(BCLK-AD)}$ | Address output hold time (BCLK standard) | | 4 | | ns |
| $t_{h(RD-AD)}$ | Address output hold time (RD standard) | | (Note1) | | ns |
| $t_{h(WR-AD)}$ | Address output hold time (WR standard) | | (Note1) | | ns |
| $t_{d(BCLK-CS)}$ | Chip select output delay time | | | 60 | ns |
| $t_{h(BCLK-CS)}$ | Chip select output hold time (BCLK standard) | | 4 | | ns |
| $t_{h(RD-CS)}$ | Chip select output hold time (RD standard) | | (Note1) | | ns |
| $t_{h(WR-CS)}$ | Chip select output hold time (WR standard) | | (Note1) | | ns |
| $t_{d(BCLK-RD)}$ | RD signal output delay time | | | 60 | ns |
| $t_{h(BCLK-RD)}$ | RD signal output hold time | | 0 | | ns |
| $t_{d(BCLK-WR)}$ | WR signal output delay time | | | 60 | ns |
| $t_{h(BCLK-WR)}$ | WR signal output hold time | | 0 | | ns |
| $t_{d(BCLK-DB)}$ | Data output delay time (BCLK standard) | | | 80 | ns |
| $t_{h(BCLK-DB)}$ | Data output hold time (BCLK standard) | | 4 | | ns |
| $t_{d(DB-WR)}$ | Data output delay time (WR standard) | | (Note1) | | ns |
| $t_{h(WR-DB)}$ | Data output hold time (WR standard) | | (Note1) | | ns |
| $t_{d(BCLK-ALE)}$ | ALE signal output delay time (BCLK standard) | | | 60 | ns |
| $t_{h(BCLK-ALE)}$ | ALE signal output hold time (BCLK standard) | | – 4 | | ns |
| $t_{d(AD-ALE)}$ | ALE signal output delay time (Address standard) | | (Note1) | | ns |
| $t_{h(ALE-AD)}$ | ALE signal output hold time(Address standard) | | 40 | | ns |
| $t_{d(AD-RD)}$ | Post-address RD signal output delay time | | 0 | | ns |
| $t_{d(AD-WR)}$ | Post-address WR signal output delay time | | 0 | | ns |
| $t_{dZ(RD-AD)}$ | Address output floating start time | | | 8 | ns |

Note 1: Calculated according to the BCLK frequency as follows:

$$th(RD - AD) = \frac{10^9}{f(BCLK) \times 2} \quad [ns]$$

$$th(WR - AD) = \frac{10^9}{f(BCLK) \times 2} \quad [ns]$$

$$th(RD - CS) = \frac{10^9}{f(BCLK) \times 2} \quad [ns]$$

$$th(WR - CS) = \frac{10^9}{f(BCLK) \times 2} \quad [ns]$$

$$td(DB - WR) = \frac{10^9 \times 3}{f(BCLK) \times 2} - 80 \quad [ns]$$

$$th(WR - DB) = \frac{10^9}{f(BCLK) \times 2} \quad [ns]$$

$$td(AD - ALE) = \frac{10^9}{f(BCLK) \times 2} - 45 \quad [ns]$$

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

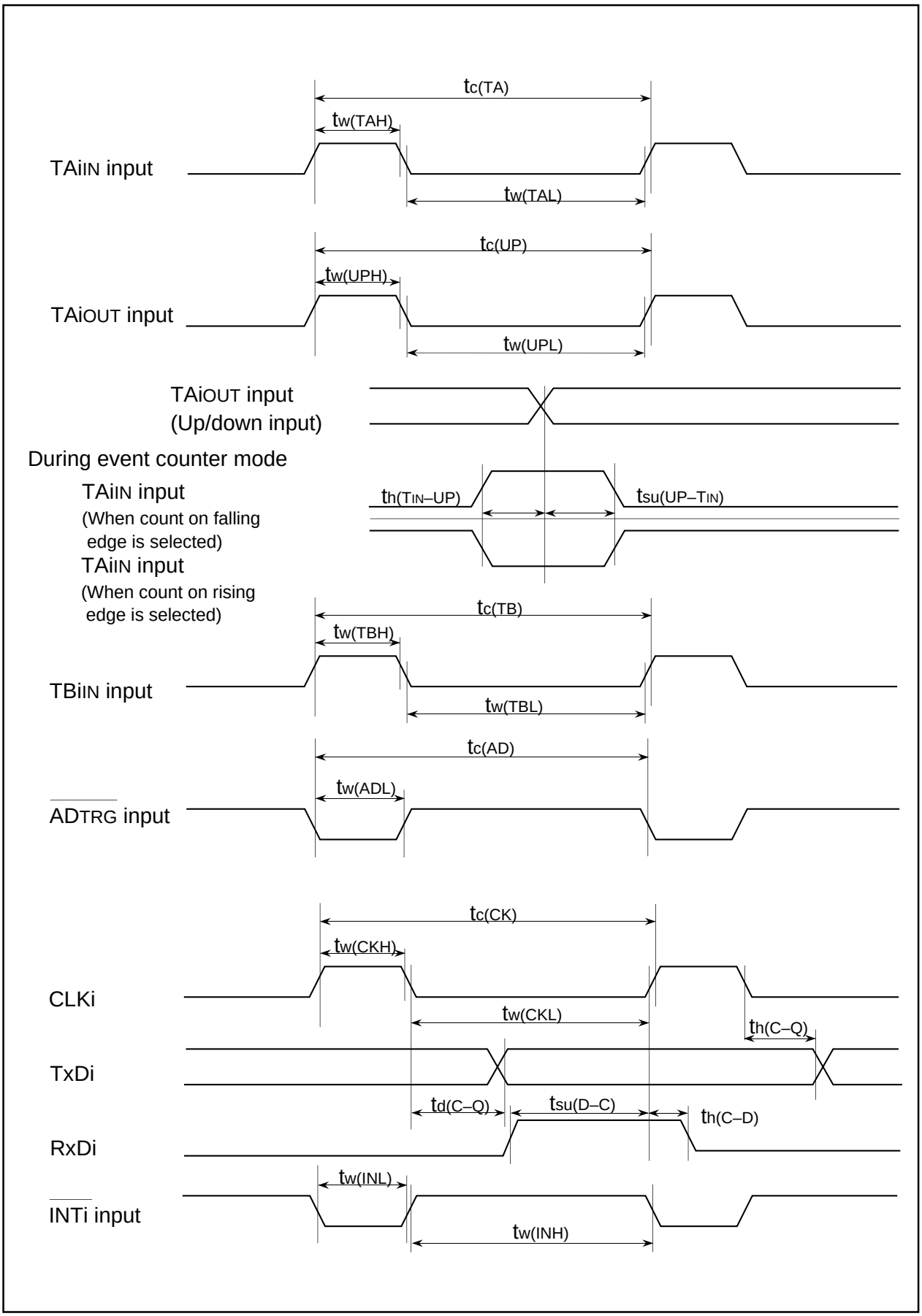
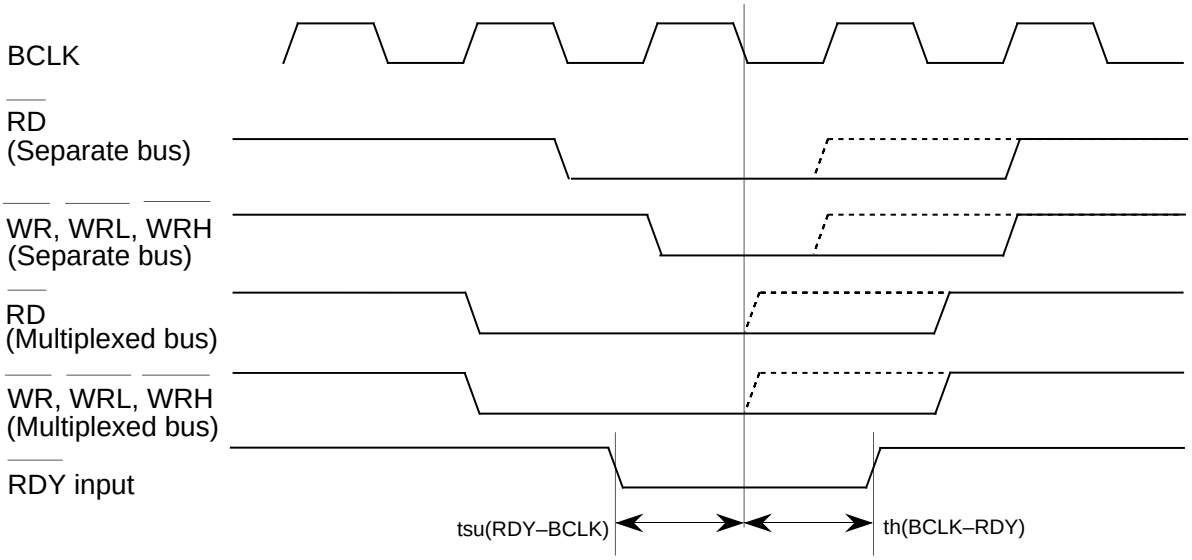


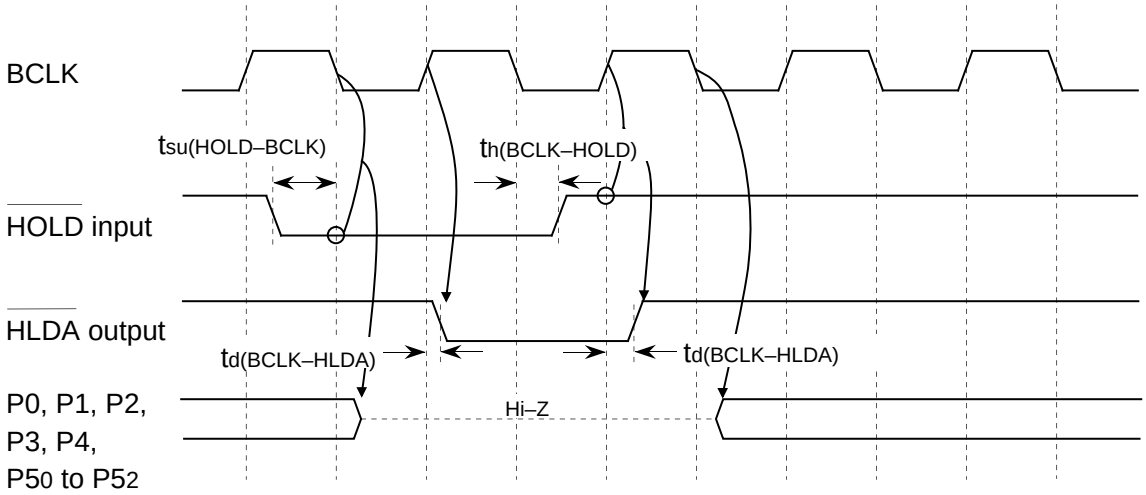

**Figure 1.26.2. Timing diagram (1)**

**Memory Expansion Mode and Microprocessor Mode**

**(Valid only with wait)**




**(Valid with or without wait)**



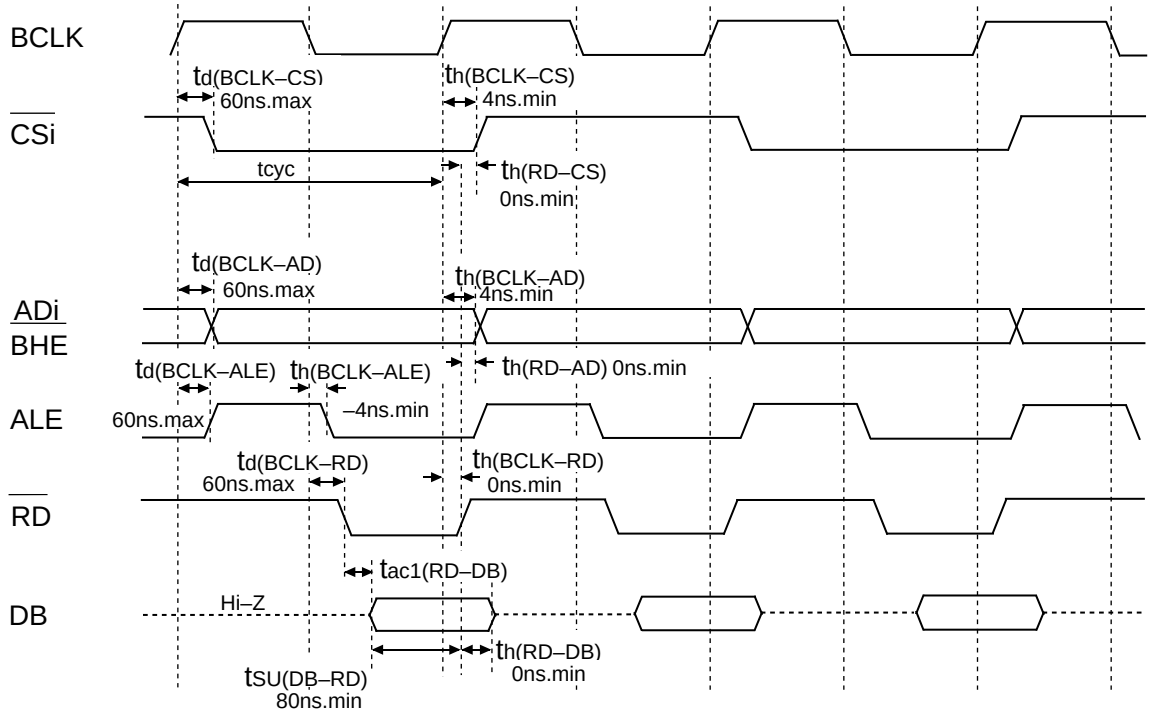

Note: The above pins are set to high-impedance regardless of the input level of the BYTE pin and bit (PM06) of processor mode register 0 selects the function of ports P40 to P43.

Measuring conditions :
- $V_{CC}$=3V
- Input timing voltage : Determined with $V_{IL}$=0.6V, $V_{IH}$=2.4V
- Output timing voltage : Determined with $V_{OL}$=1.5V, $V_{OH}$=1.5V

**Figure 1.26.3. Timing diagram (2)**

## Memory Expansion Mode and Microprocessor Mode
**(With no wait)**

**Read timing**




**Write timing**

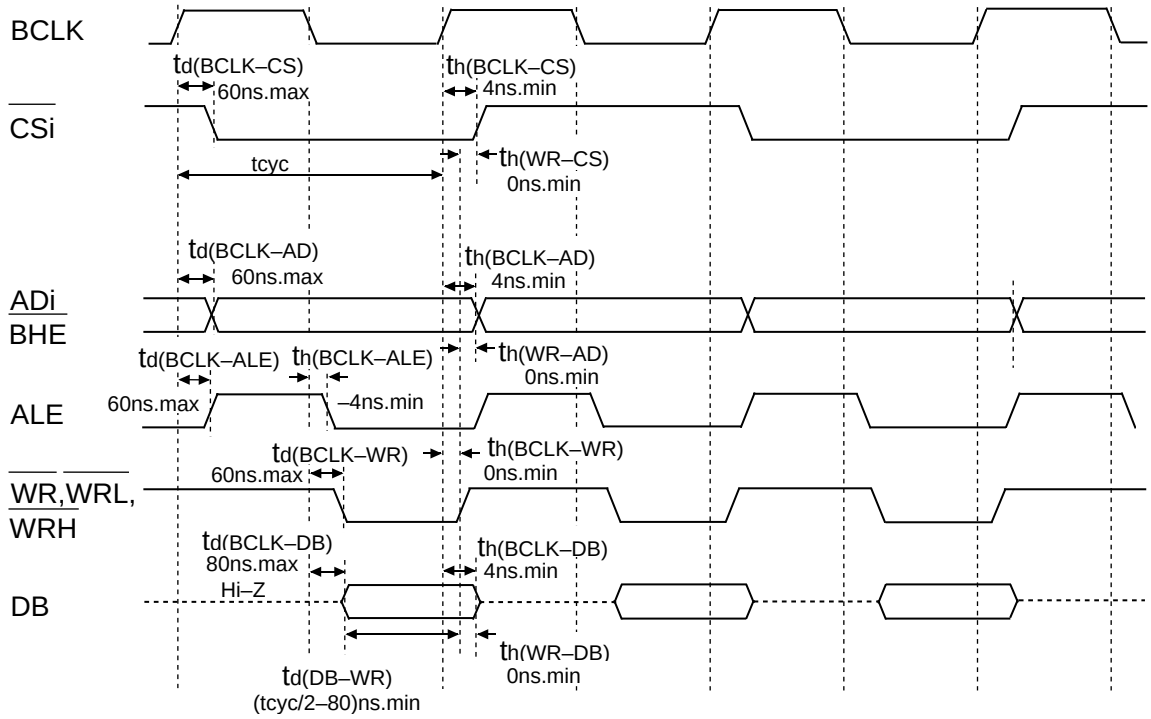
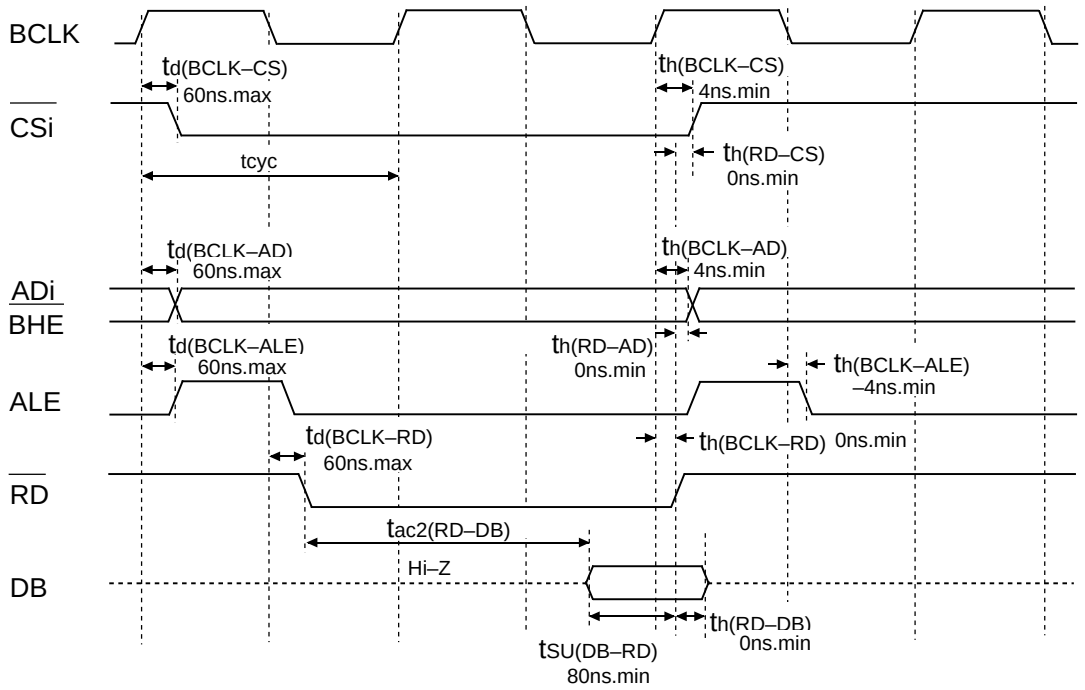



**Figure 1.26.4. Timing diagram (3)**

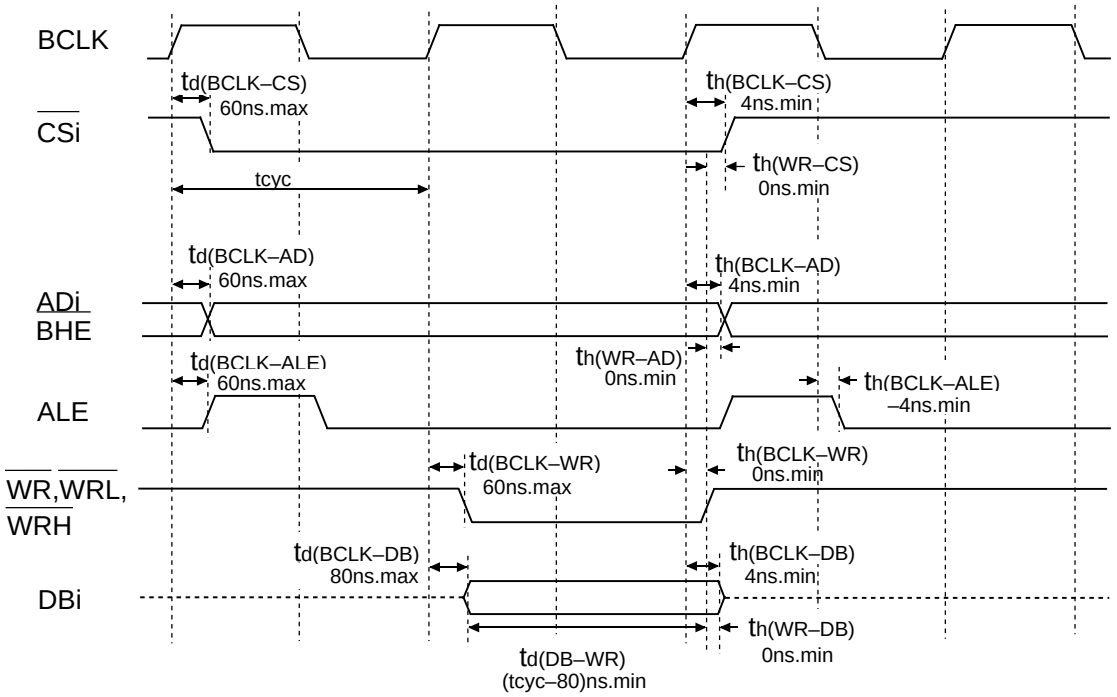
## Memory Expansion Mode and Microprocessor Mode

**(When accessing external memory area with wait)**

**Read timing**




**Write timing**




Measuring conditions :

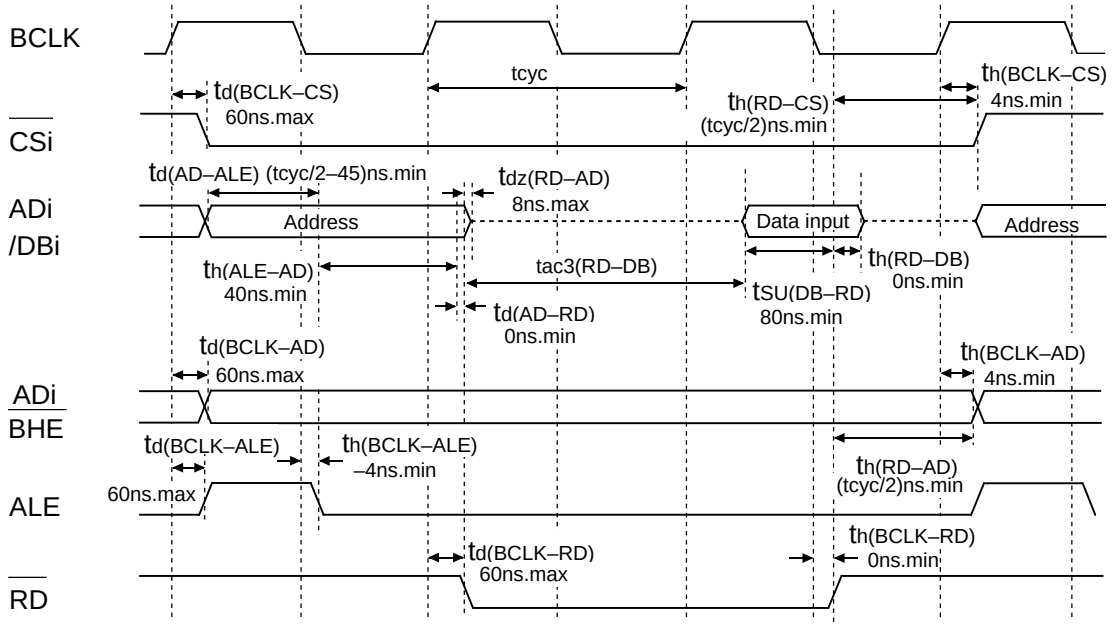
- $V_{CC}$=3V
- Input timing voltage : Determined with $V_{IL}$=0.48V, $V_{IH}$=1.5V
- Output timing voltage : Determined with $V_{OL}$=1.5V, $V_{OH}$=1.5V

**Figure 1.26.5. Timing diagram (4)**

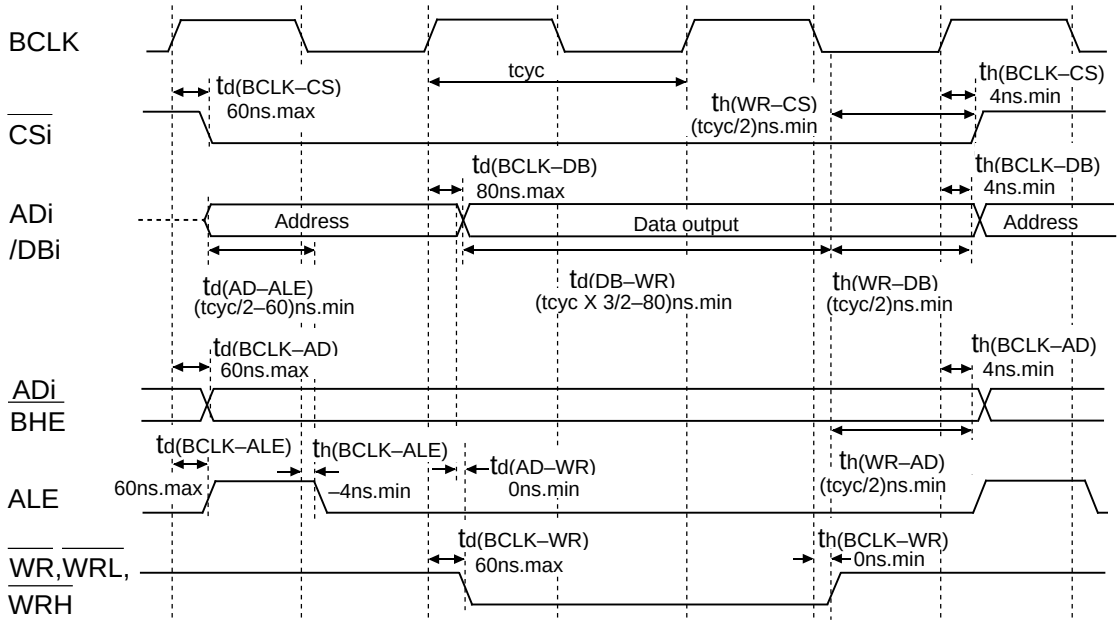
## Memory Expansion Mode and Microprocessor Mode

**(When accessing external memory area with wait, and select multiplexed bus)**

**Read timing**




**Write timing**




Measuring conditions :
- $V_{CC}$=3V
- Input timing voltage : Determined with $V_{IL}$=0.48V,$V_{IH}$=1.5V
- Output timing voltage : Determined with $V_{OL}$=1.5V,$V_{OH}$=1.5V

**Figure 1.26.6. Timing diagram (5)**

## Outline Performance (Flash Memory Version)

Table 1.28.1 shows the outline performance of the M3062GF8N (flash memory version).

**Table 1.28.1. Outline performance of the M3062GF8N (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 1.28.1 |
| | Boot ROM area | One division (4 Kbytes) (Note 1) |
| Program method | | In units of word/byte (Note 2) |
| 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 1: 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.
Note 2: Can be programmed in byte unit only when using parallel I/O mode.

## Flash Memory

The M3062GF8N (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 1.28.1, 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.

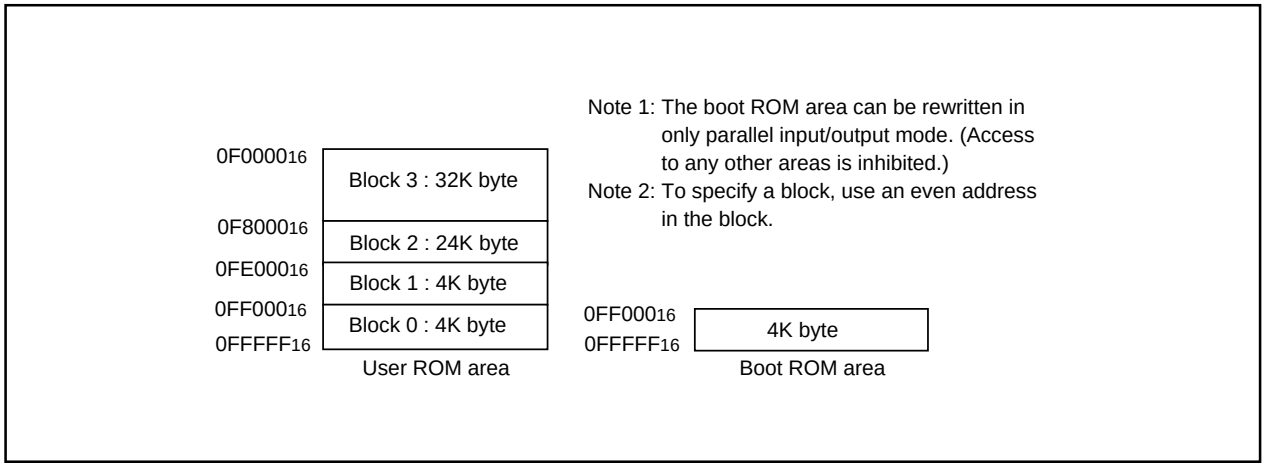



**Figure 1.28.1. Block diagram of flash memory version**

## 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 1.28.1 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 1.28.1 for details about the boot ROM area.

Normal microcomputer mode is entered when the microcomputer is reset with pulling $CNV_{SS}$ 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 $P5_5$ pin low, the $CNV_{SS}$ pin high, and the $P5_0$ 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 (When rewriting the use ROM area in boot mode, bit 5 of the flash memory control register 0 must be set to “1”. Write to this bit only when executing out of an area other than the internal flash memory).

### Block Address

Block addresses refer to an 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 $03B7_{16}$) 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. Write data into even address in 16-bit units. Do not write 16-bit data into odd address or data in 8-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. Read data from an even address in the user ROM area when reading the status register.

Figure 1.29.1 shows the flash identification register and flash memory control register 0.

Bit 0 of the flash memory control register 0 is the RY/$\overline{BY}$ status flag used exclusively to read the operating status of the flash memory. During programming, erase and lock-bit programming 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 to this bit only when executing out of an area other than the internal flash memory. Also only when $\overline{NMI}$ pin is "H" level. To set this bit to “1”, it is necessary to write “0” and then write “1” in succession. To set this bit 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” when RY/$\overline{BY}$ status flag is “1”.

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 6 of the flash memory control register 0 is the program status flag used exclusively to read the operating status of the auto program operation. If a program error occurs, it is set to “1”. Otherwise, it is “0”.

Bit 7 of the flash memory control register 0 is the erase status flag used exclusively to read the operating status of the auto erase operation. If an erase error occurs, it is set to “1”. Otherwise, it is “0”.

Figure 1.29.2 shows a flowchart for setting/releasing the CPU rewrite mode. Always perform operation as indicated in these flowcharts.

Flash identification register

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| FIDR | $03B4_{16}$ | $00_{16}$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| FIDR0 | Flash identification value | Flash value output<br>HND: $00_{16}$ | ○ | × |

Procedure

(1) Write $FF_{16}$ to the address $03B4_{16}$
(2) Read address $03B4_{16}$
Read value = $FF_{16}$ ••• DINOR flash memory
Read value = $00_{16}$ ••• HND flash memory

Flash memory control register 0

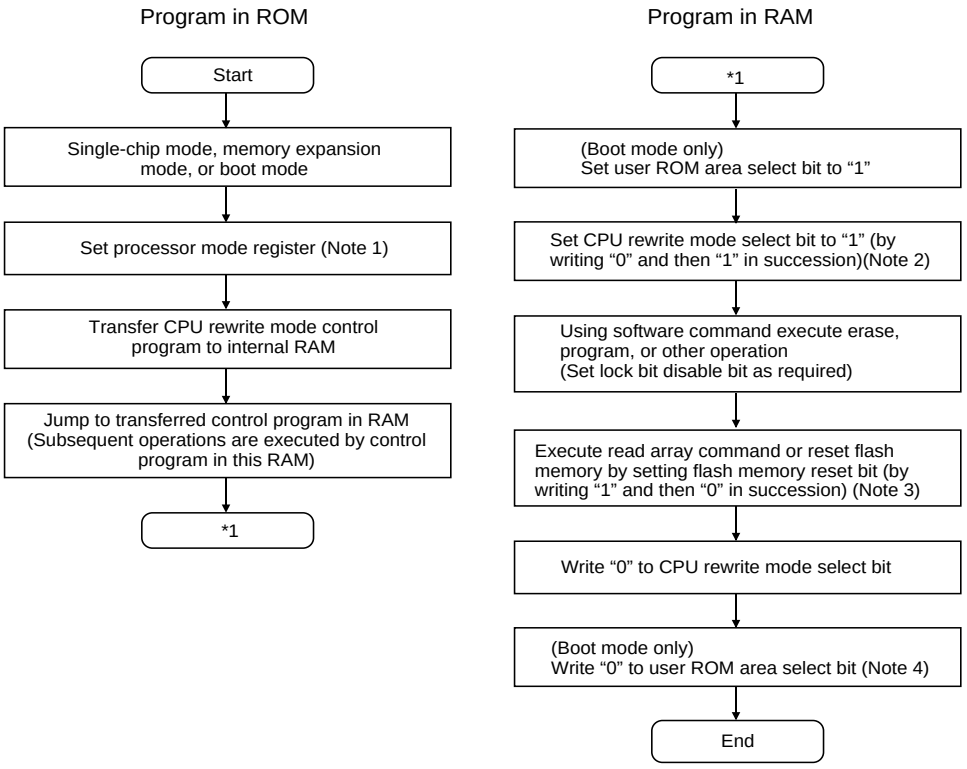
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | 0 | | | | |

| Symbol | Address | When reset |
|---|---|---|
| FMR0 | $03B7_{16}$ | $XX000001_2$ |

| Bit symbol | Bit name | Function | R | W |
|---|---|---|---|---|
| FMR00 | RY/$\overline{BY}$ status flag | 0: Busy (being written or erased)<br>1: Ready | ○ | × |
| FMR01 | CPU rewrite mode select bit (Note 1) | 0: Normal mode (Software commands invalid)<br>1: CPU rewrite mode (Software commands acceptable) | ○ | ○ |
| FMR02 | Lock bit disable select bit (Note 2) | 0: Block lock by lock bit data is enabled<br>1: Block lock by lock bit data is disabled | ○ | ○ |
| FMR03 | Flash memory reset bit (Note 3) | 0: Normal operation<br>1: Reset | ○ | ○ |
| Reserved bit | | Must always be set to “0” | ○ | ○ |
| FMR05 | User ROM area select bit (Note 4) (Effective in only boot mode) | 0: Boot ROM area is accessed<br>1: User ROM area is accessed | ○ | ○ |
| FMR06 | Program status flag | 0: Pass<br>1: Error | ○ | × |
| FMR07 | Erase status flag | 0: Pass<br>1: Error | ○ | × |

Note 1: For this 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 to this bit only when executing out of an area other than the internal flash memory. Also only when $\overline{NMI}$ pin is “H” level. Clear this bit to “0” after read array command.
Note 2: For this bit to be set to “1”, the user needs to write a “0” and then a “1” to it in succession when the CPU rewrite mode select bit = “1”. 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.
Note 3: Effective only when the CPU rewrite mode select bit = 1. After write “1”, write “0” when RY/$\overline{BY}$ status flag is “1”.
Note 4: Write to this bit only when executing out of an area other than the internal flash memory.

**Figure 1.29.1. Flash identification register and flash memory control register 0**




Note 1: During CPU rewrite mode, set the BCLK as shown below using the main clock divide ratio select bit (bit 6 at address $0006_{16}$ and bits 6 and 7 at address $0007_{16}$):
6.25 MHz or less when wait bit (bit 7 at address $0005_{16}$) = "0" (without internal access wait state)
10.0 MHz or less when wait bit (bit 7 at address $0005_{16}$) = "1" (with internal access wait state)
Note 2: For CPU rewrite mode select 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 to this bit only when executing out of an area other than the internal flash memory. Also only when $\overline{NMI}$ pin is "H" level.
Note 3: Before exiting the CPU rewrite mode after completing erase or program operation, always be sure to execute a read array command or reset the flash memory.
Note 4: "1" can be set. However, when this bit is "1", user ROM area is accessed.

**Figure 1.29.2. CPU rewrite mode set/reset 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 $0006_{16}$ and bits 6 and 7 at address $0007_{16}$):

6.25 MHz or less when wait bit (bit 7 at address $0005_{16}$) = 0 (without internal access wait state)

10.0 MHz or less when wait bit (bit 7 at address $0005_{16}$) = 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 to automatically initialize the flash identification register and flash memory control register 0 to “0”, then return to normal operation. However, these two interrupts' jump addresses are located in the fixed vector table and there must exsist a routine to be executed. Since the rewrite operation is halted when an $\overline{\text{NMI}}$ or watchdog timer interrupts occurs, you must reset the CPU rewite mode select bit to “1” and the perform the erase/program operation again.

**(4) Access disable**

Write to CPU rewrite mode select bit and user ROM area select bit only when executing out of an area other than the internal flash memory.

**(5) How to access**

For CPU rewrite mode select bit and lock bit disable select 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 to CPU rewrite mode select bit and user ROM area 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.

**(6)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.

**(7)Using the lock bit**

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

## Software Commands

Table 1.29.1 lists the software commands available with the M3062GF8N (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 ($D_8$ to $D_{15}$) is ignored. The content of each software command is explained below.

**Table 1.29.1. List of software commands (CPU rewrite mode)**

| Command | First bus cycle | | | Second bus cycle | | |
|---|---|---|---|---|---|---|
| | Mode | Address | Data ($D_0$ to $D_7$) | Mode | Address | Data ($D_0$ to $D_7$) |
| Read array | Write | X | $FF_{16}$ | | | |
| Read status register | Write | X | $70_{16}$ | Read | X | SRD (Note 2) |
| Clear status register | Write | X | $50_{16}$ | | | |
| Program (Note 3) | Write | WA | $40_{16}$ | Write | WA (Note 3) | WD (Note 3) |
| Block erase | Write | X | $20_{16}$ | Write | BA (Note 4) | $D0_{16}$ |
| Erase all unlock block | Write | X | $A7_{16}$ | Write | X | $D0_{16}$ |
| Lock bit program | Write | BA | $77_{16}$ | Write | BA | $D0_{16}$ |
| Read lock bit status | Write | X | $71_{16}$ | Read | BA | $D_6$ (Note 5) |

Note 1: When a software command is input, the high-order byte of data ($D_8$ to $D_{15}$) is ignored.
Note 2: SRD = Status Register Data (Set an address to even address in the user ROM area)
Note 3: WA = Write Address (even address), WD = Write Data (16-bit data)
Note 4: BA = Block Address (Enter the maximum address of each block that is an even address.)
Note 5: $D_6$ corresponds to the block lock status. Block not locked when $D_6$ = 1, block locked when $D_6$ = 0.
Note 6: X denotes a given address in the user ROM area (that is an even address).

**Read Array Command ($FF_{16}$)**

The read array mode is entered by writing the command code "$FF_{16}$" 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 ($D_0$–$D_{15}$), 16 bits at a time.

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

However, please begin to read data in the following procedures when a user uses read array command after program command.

(1) Set $FF_{16}$, $FF_{16}$, $FF_{16}$, $FF_{16}$ to arbitrary continuing four address beforehand
(2) Input the top address which $FF_{16}$ was set at (in read array mode)
(3) Input the top address till $FFFF_{16}$ agrees with the value that begins to have been read
(4) Input top address +2
(5) Input top address +2 till $FFFF_{16}$ agrees with the value that begins to have been read
(6) Input an arbitrary address

**Read Status Register Command ($70_{16}$)**

When the command code "$70_{16}$" is written in the first bus cycle, the content of the status register is read out at the data bus ($D_0$–$D_7$) by a read in the second bus cycle (Set an address to even address in the user ROM area).

The status register is explained in the next section.

**Clear Status Register Command ($50_{16}$)**

This command is used to clear the bits SR4 and SR5 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 "$50_{16}$" in the first bus cycle.

**Program Command (40$_{16}$)**

Program operation starts when the command code "40$_{16}$" is written in the first bus cycle. Then, if the address and data to program are written in the 2nd bus cycle, program operation (data programming and verification) will start. Make an address in the first bus cycle same as an address to program by the second bus cycle.

Whether the write operation is completed can be confirmed by reading the status register or the RY/$\overline{BY}$ status flag. When the program starts, the read status register mode is accessed automatically and the content of the status register is read into the data bus (D0 - D7). The status register bit 7 (SR7) is set to 0 at the same time the write operation starts and is returned to 1 upon completion of the write operation. In this case, the read status register mode remains active until the Read Array command (FF$_{16}$) is written.

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

At program end, program results can be checked by reading the status register.

Figure 1.29.3 shows an example of a 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.

Do a command to use in right after of program command as follows

Make an address in the first bus cycle same as an address to program by the second bus cycle of program command.

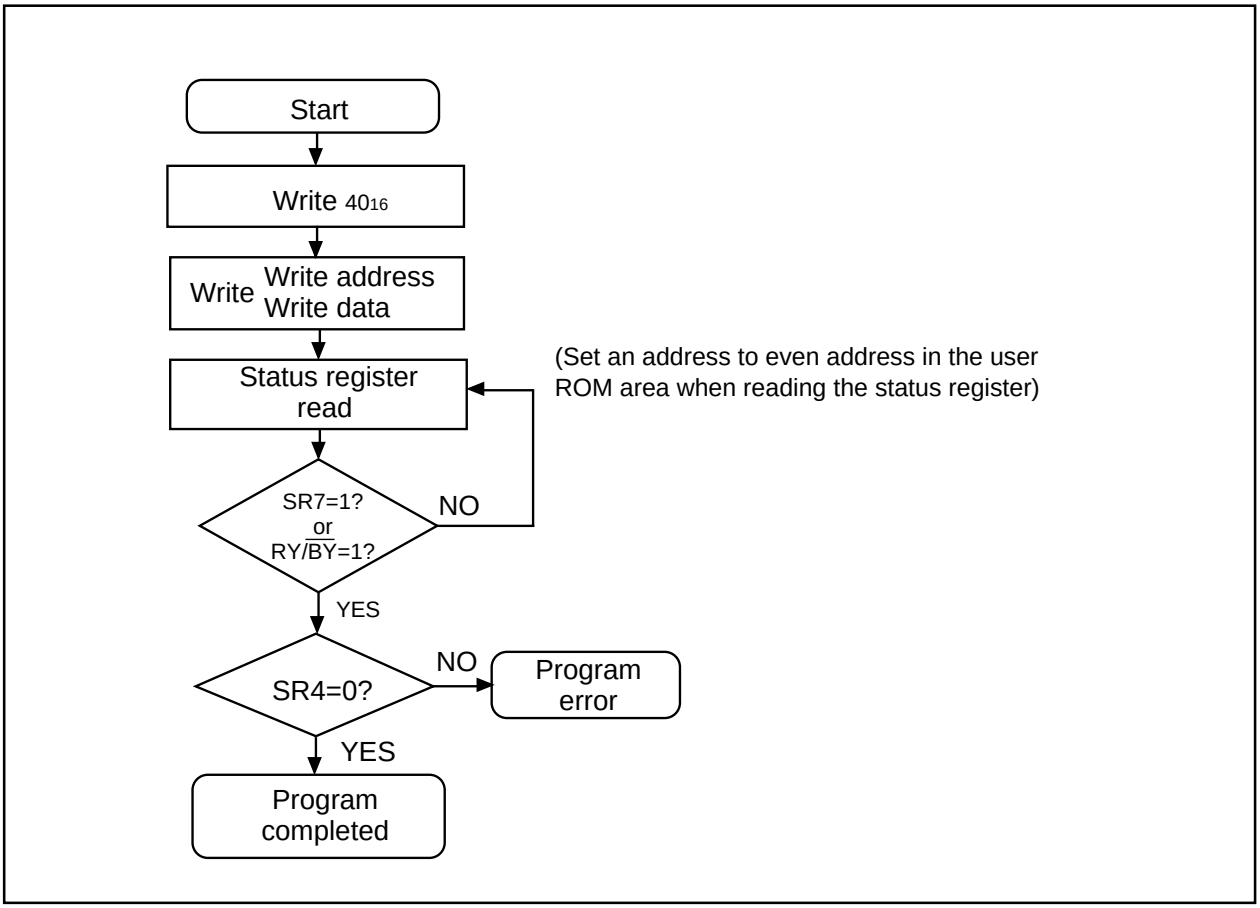



**Figure 1.29.3. Program flowchart**

**Block Erase Command ($20_{16}$/$D0_{16}$)**

By writing the command code "$20_{16}$" in the first bus cycle and the confirmation command code "$D0_{16}$" 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.

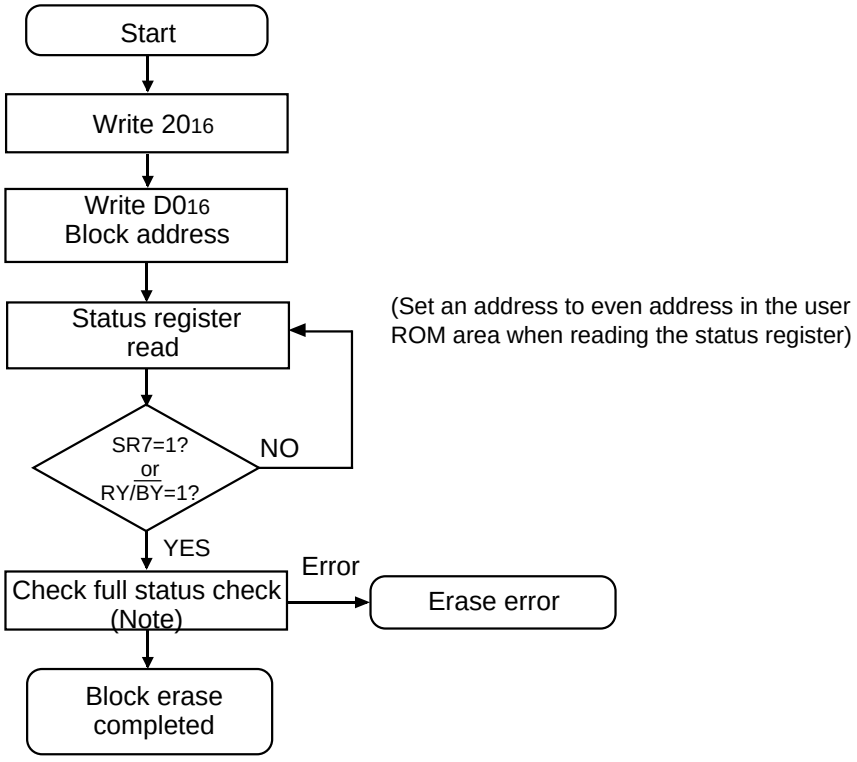
Whether the auto erase operation is completed can be confirmed by reading the status register or the flash memory control register 0. At the same time the auto erase operation starts, the read status register mode is automatically entered, so the content of the status register can be read out. The status 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 ($FF_{16}$) or Read Lock Bit Status command ($71_{16}$) is written or the flash memory is reset using its reset bit.

The RY/$\overline{BY}$ 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 1.29.4 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.




Note: Refer to Figure 1.29.7 .

**Figure 1.29.4. Block erase flowchart**

**Erase All Unlock Blocks Command ($A7_{16}/D0_{16}$)**

By writing the command code "$A7_{16}$" in the first bus cycle and the confirmation command code "$D0_{16}$" 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 select 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 ($77_{16}/D0_{16}$)**

By writing the command code "$77_{16}$" in the first bus cycle and the confirmation command code "$D0_{16}$" 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). Make an address in the first bus cycle same as an address to block by the second bus cycle.

Figure 1.29.5 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.

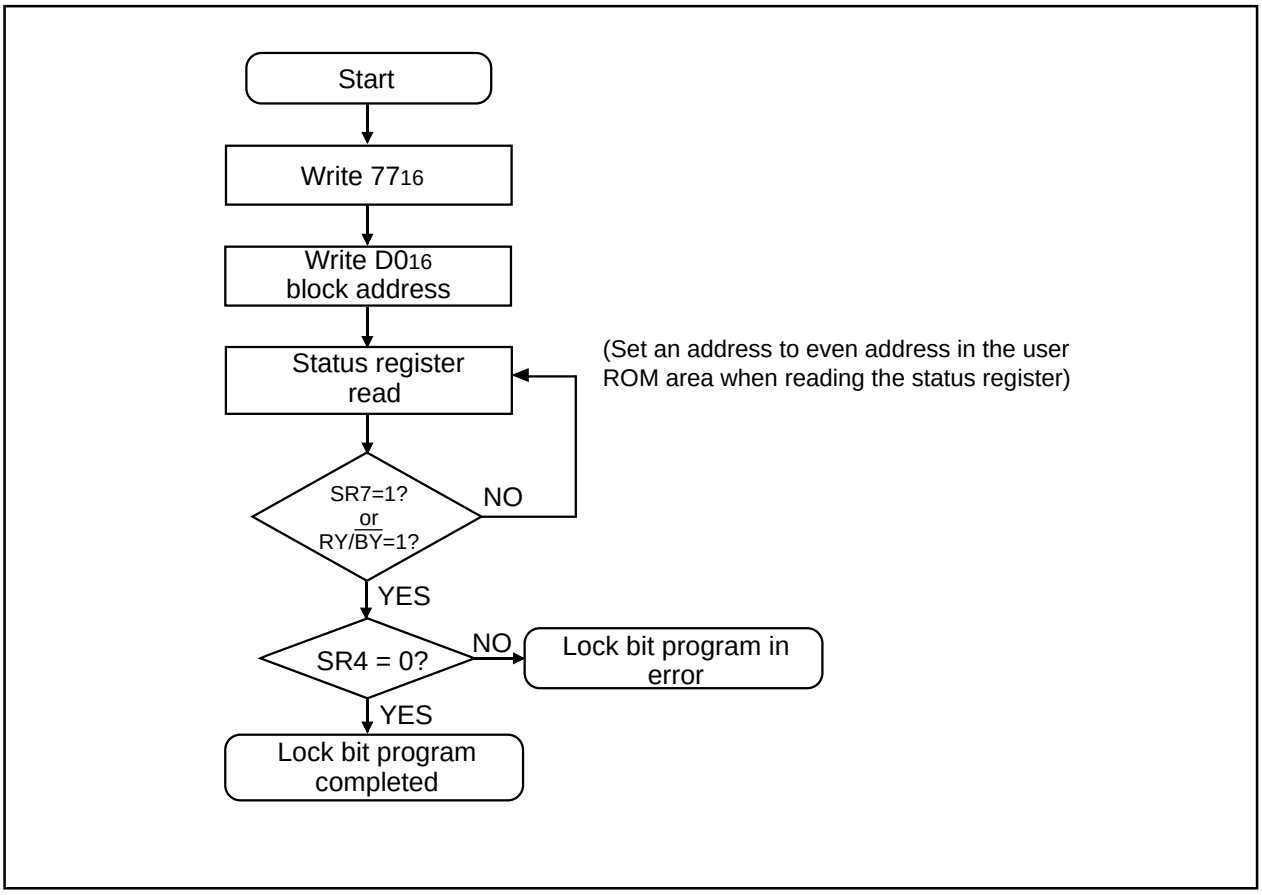
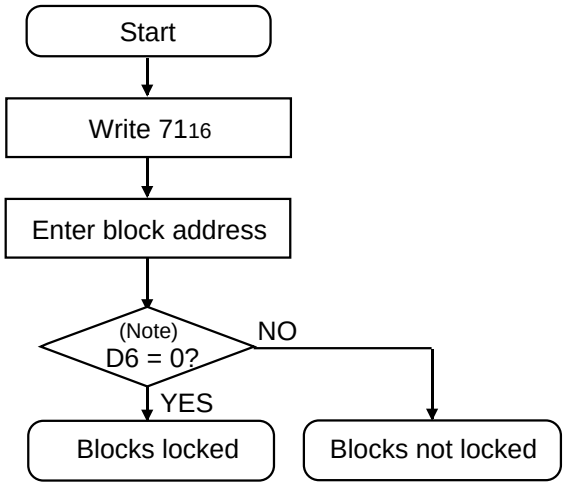



**Figure 1.29.5. Lock bit program flowchart**

**Read Lock Bit Status Command (71$_{16}$)**

By writing the command code "71$_{16}$" 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 1.29.6 shows an example of a read lock bit program flowchart.




Note: Data bus bit 6.

**Figure 1.29.6. Read lock bit status flowchart**

## Data Protect Function (Block Lock)

Each block in Figure 1.28.1 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 shows the operating state of the flash memory and whether erase operations and programs ended successfully or in error. It can be read in the following ways.

(1) By reading an arbitrary even address from the user ROM area after writing the read status register command ($70_{16}$)

(2) By reading an arbitrary even address from the user ROM area in the period from when the program starts or erase operation starts to when the read array command ($FF_{16}$) is input

Table 1.29.2 shows the status register.

Also, the status register can be cleared in the following way.

(1) By writing the clear status register command ($50_{16}$)

After a reset, the status register is set to "$80_{16}$".

Each bit in this register is explained below.

**Sequencer status (SR7)**

After power-on, the sequencer status is set to 1(ready).

The sequencer status indicates the operating status of the device. This status bit is set to "0" (busy) during write or erase operation and is set to "1" upon completion of these operations.

**Erase status (SR5)**

The erase status informs the operating status of erase operation to the CPU. When an erase error occurs, it is set to "1".

The erase status is reset to "0" when cleared.

**Program status (SR4)**

The program status informs the operating status of 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 ($20_{16}$) is not the confirmation command ($D0_{16}$), 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

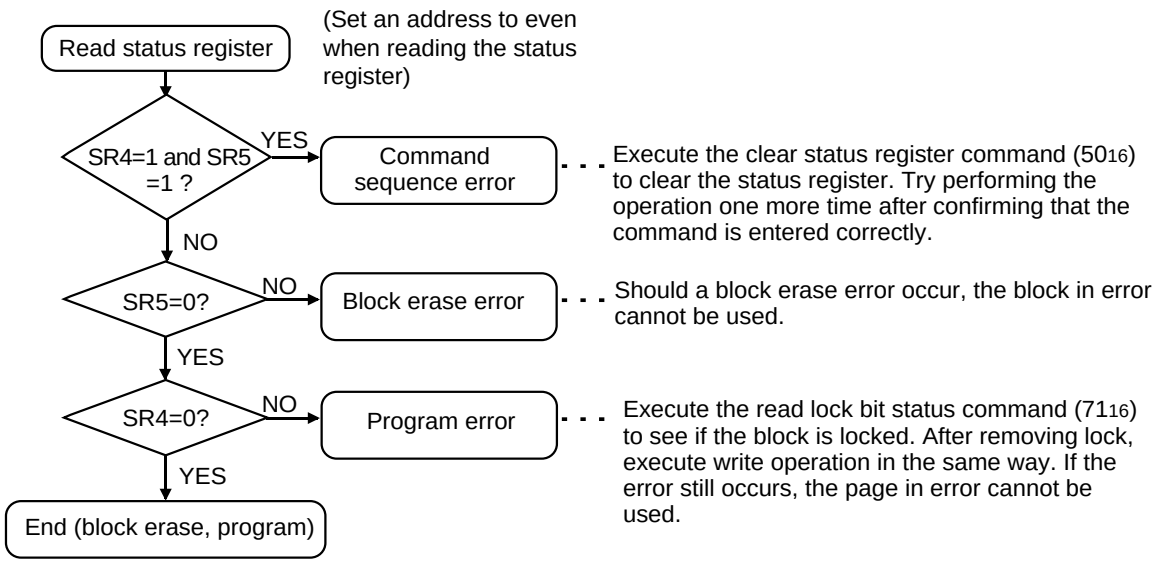
(2) When the data entered in the second bus cycle of lock bit program ($77_{16}$/$D0_{16}$), block erase ($20_{16}$/$D0_{16}$), or erase all unlock blocks ($A7_{16}$/$D0_{16}$) is not the $D0_{16}$ or $FF_{16}$. However, if $FF_{16}$ is entered, read array is assumed and the command that has been set up in the first bus cycle is canceled.

**Table 1.29.2. Definition of each bit in status register**

| Each bit of SRD | Status name | Definition | |
|---|---|---|---|
| | | "1" | "0" |
| SR7 (bit7) | Sequencer 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) | Reserved | - | - |
| 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 1.29.7 shows a full status check flowchart and the action to be taken when each error occurs.




Note: When one of SR5 to SR4 is set to 1, none of the program, erase all blocks, and block erase commands is accepted. Execute the clear status register command ($50_{16}$) before executing these commands.

**Figure 1.29.7. 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 ($0FFFFF_{16}$). Figure 1.30.1 shows the ROM code protect control address ($0FFFFF_{16}$). (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.

ROM code protect control address

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| Symbol | Address | When reset |
|---|---|---|
| ROMCP | $0FFFFF_{16}$ | $FF_{16}$ |

| Bit symbol | Bit name | Function |
|---|---|---|
| Reserved bit | | Always set this bit to 1. |
| ROMCP2 | ROM code protect level 2 set bit (Note 1, 2) | b3 b2<br>00: Protect enabled<br>01: Protect enabled<br>10: Protect enabled<br>11: Protect disabled |
| ROMCR | ROM code protect reset bit (Note 3) | b5 b4<br>00: Protect removed<br>01: Protect set bit effective<br>10: Protect set bit effective<br>11: Protect set bit effective |
| ROMCP1 | ROM code protect level 1 set bit (Note 1) | b7 b6<br>00: Protect enabled<br>01: Protect enabled<br>10: Protect enabled<br>11: Protect disabled |

Note 1: When ROM code protect is turned on, the on-chip flash memory is protected against readout or modification in parallel input/output mode.
Note 2: When ROM code protect level 2 is turned on, ROM code readout by a shipment inspection LSI tester, etc. also is inhibited.
Note 3: The ROM code protect reset bits can be used to turn off ROM code protect level 1 and ROM code protect level 2. However, since these bits cannot be changed in parallel input/output mode, they need to be rewritten in serial input/output or some other mode.

**Figure 1.30.1. 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 $0FFFDF_{16}$, $0FFFE3_{16}$, $0FFFEB_{16}$, $0FFFEF_{16}$, $0FFFF3_{16}$, $0FFFF7_{16}$, and $0FFFFB_{16}$. Write a program which has had the ID code preset at these addresses to the flash memory.

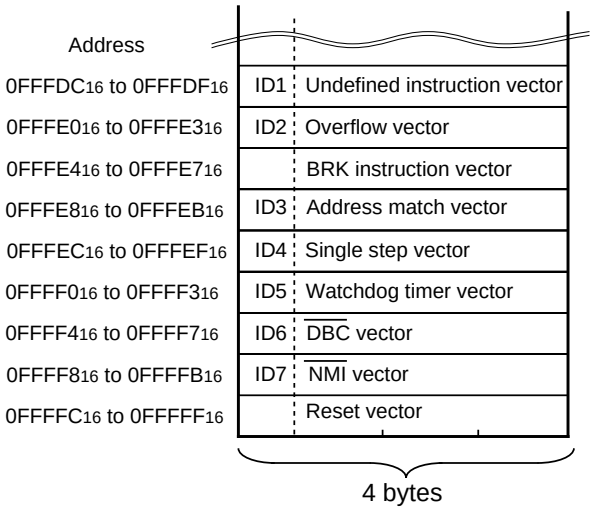



**Figure 1.30.2. ID code store addresses**

## 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 programer supporting M3062GF8N (flash memory version).
Refer to the instruction manual of each programer 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 1.28.1 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 1.28.1.
The boot ROM area is 4 Kbytes in size. In parallel I/O mode, it is located at addresses $0FF000_{16}$ through $0FFFFF_{16}$. 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 4 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.

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

| Pin | Name | I/O | Description |
|---|---|---|---|
| $V_{CC}$,$V_{SS}$ | Power input | | Apply program/erase protection voltage to Vcc pin and 0 V to Vss pin. |
| $CNV_{SS}$ | $CNV_{SS}$ | I | Connect to Vcc pin. |
| $\overline{RESET}$ | Reset input | I | Reset input pin. While reset is "L" level, a 20 cycle or longer clock must be input to XIN pin. |
| $X_{IN}$ | Clock input | I | Connect a ceramic resonator or crystal oscillator between $X_{IN}$ and $X_{OUT}$ pins. To input an externally generated clock, input it to $X_{IN}$ pin and open $X_{OUT}$ pin. |
| $X_{OUT}$ | Clock output | O | |
| BYTE | BYTE | I | Connect this pin to Vcc or Vss. |
| $AV_{CC}$, $AV_{SS}$ | Analog power supply input | | Connect AVss to Vss and AVcc to Vcc, respectively. |
| $V_{REF}$ | Reference voltage input | I | Enter the reference voltage for AD from this pin. |
| $P0_0$ to $P0_7$ | Input port P0 | I | Input "H" or "L" level signal or open. |
| $P1_0$ to $P1_7$ | Input port P1 | I | Input "H" or "L" level signal or open. |
| $P2_0$ to $P2_7$ | Input port P2 | I | Input "H" or "L" level signal or open. |
| $P3_0$ to $P3_7$ | Input port P3 | I | Input "H" or "L" level signal or open. |
| $P4_0$ to $P4_7$ | Input port P4 | I | Input "H" or "L" level signal or open. |
| $P5_1$ to $P5_4$, $P5_6$, $P5_7$ | Input port P5 | I | Input "H" or "L" level signal or open. |
| $P5_0$ | $\overline{CE}$ input | I | Input "H" level signal. |
| $P5_5$ | $\overline{EPM}$ input | I | Input "L" level signal. |
| $P6_0$ to $P6_3$ | Input port P6 | I | Input "H" or "L" level signal or open. |
| $P6_4$ | BUSY output | O | Standard serial I/O mode 1: BUSY signal output pin<br>Standard serial I/O mode 2: Monitors the boot program operation check signal output pin. |
| $P6_5$ | SCLK input | I | Standard serial I/O mode 1: Serial clock input pin<br>Standard serial I/O mode 2: Input "L". |
| $P6_6$ | RxD input | I | Serial data input pin |
| $P6_7$ | TxD output | O | Serial data output pin |
| $P7_0$ to $P7_7$ | Input port P7 | I | Input "H" or "L" level signal or open. |
| $P8_0$ to $P8_4$, $P8_6$, $P8_7$ | Input port P8 | I | Input "H" or "L" level signal or open. |
| $P8_5$ | $\overline{NMI}$ input | I | Connect this pin to Vcc. |
| $P9_0$ to $P9_7$ | Input port P9 | I | Input "H" or "L" level signal or open. |
| $P10_0$ to $P10_7$ | Input port P10 | I | Input "H" or "L" level signal or open. |

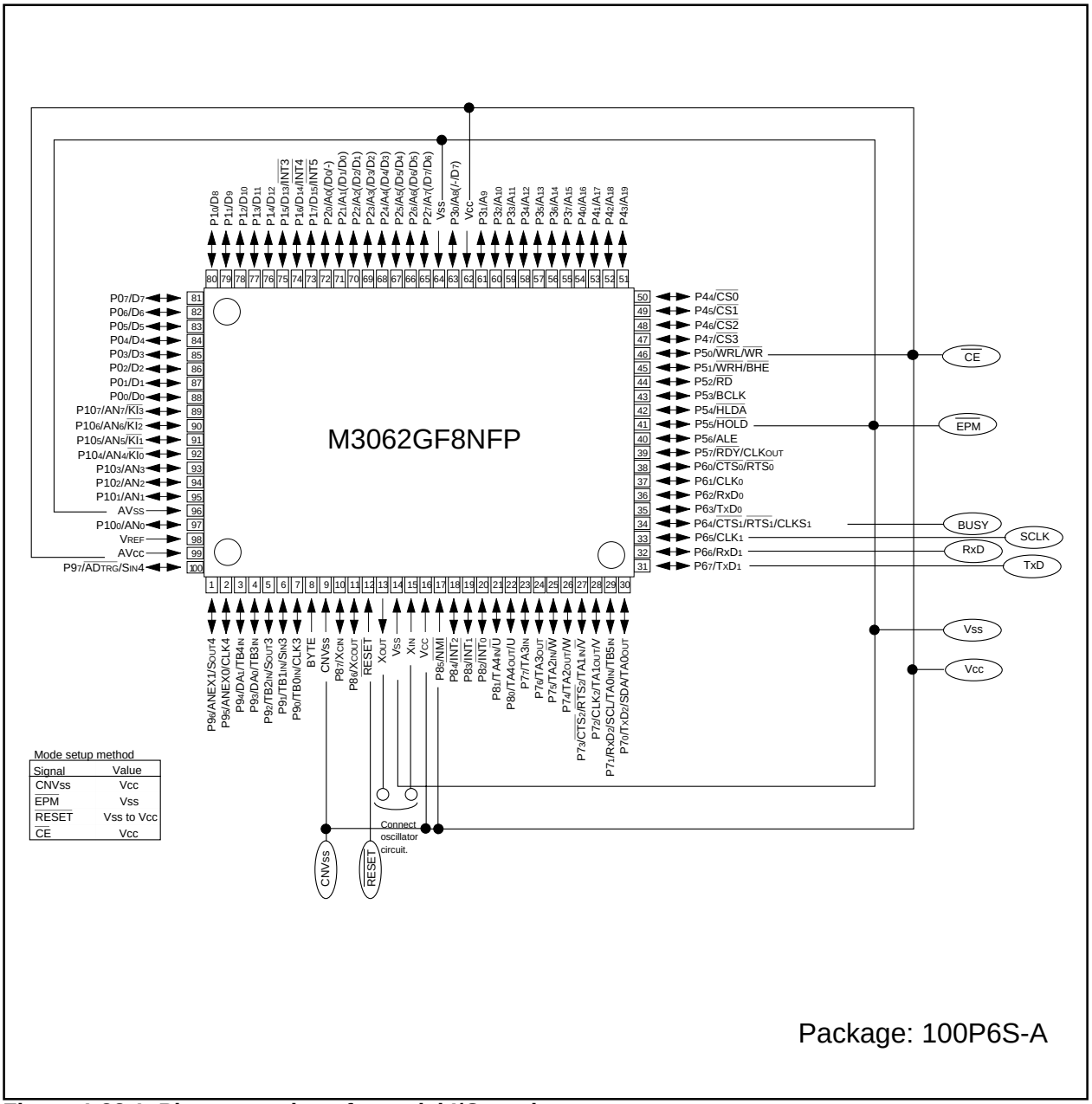



**Figure 1.32.1. Pin connections for serial I/O mode**

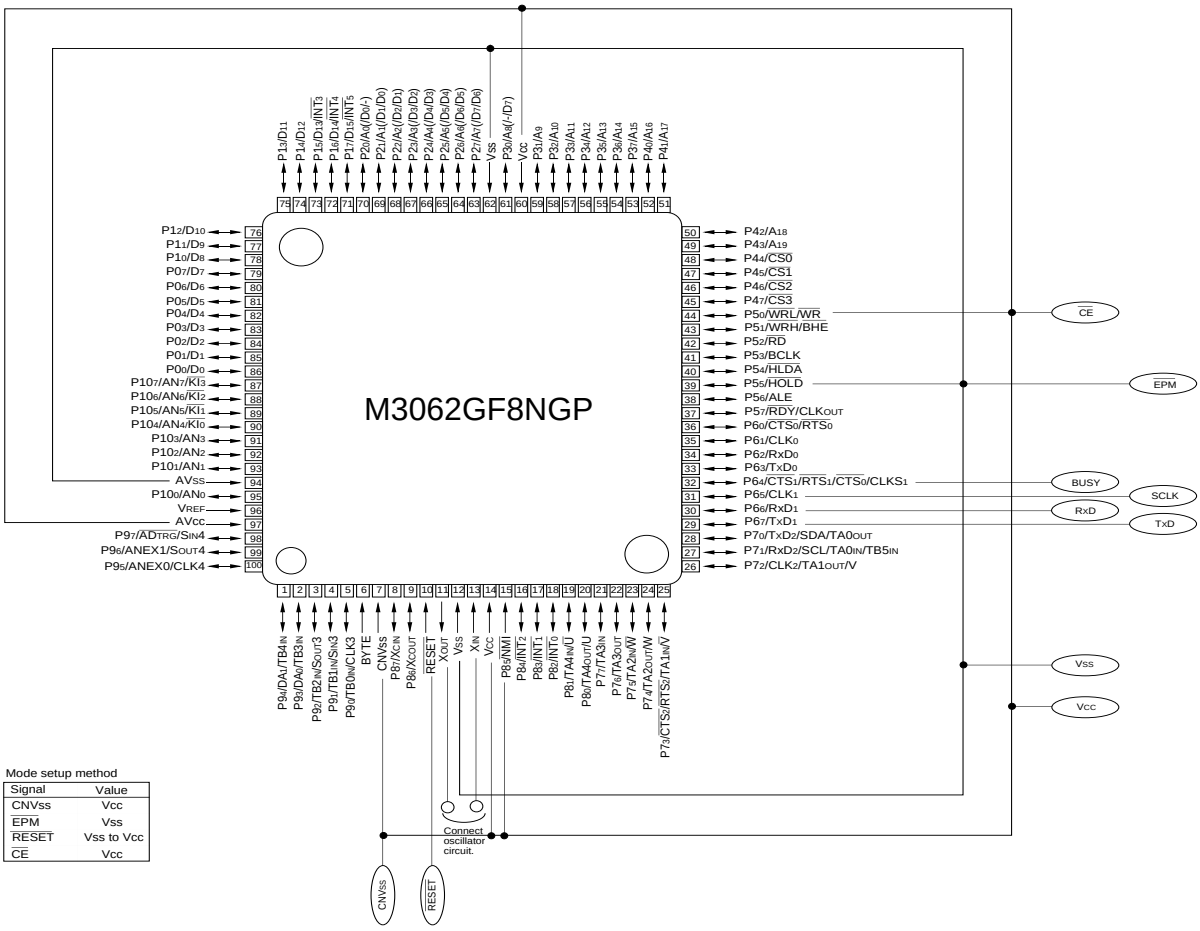



**Figure 1.32.1. Pin connections for serial I/O mode (2)**

## 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 asynchronized. 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 $P5_0$ ($\overline{CE}$) pin is "H" level, the $P5_5$ ($\overline{EPM}$) 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 1.32.1 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 asynchronized) according to the level of $CLK_1$ pin when the reset is released.

To use standard serial I/O mode 1 (clock synchronized), set the $CLK_1$ pin to "H" level and release the reset. The operation uses the four UART1 pins $CLK_1$, $RxD_1$, $TxD_1$ and $RTS_1$ (BUSY). The $CLK_1$ pin is the transfer clock input pin through which an external transfer clock is input. The $TxD_1$ pin is for CMOS output. The $RTS_1$ (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 asynchronized), set the $CLK_1$ pin to "L" level and release the reset. The operation uses the two UART1 pins $RxD_1$ and $TxD_1$.

In the standard serial I/O mode, only the user ROM area indicated in Figure 1.32.18 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 programer, etc.) using 4-wire clock-synchronized serial I/O (UART1). Standard serial I/O mode 1 is engaged by releasing the reset with the $P6_5$ ($CLK_1$) 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 $CLK_1$ pin, and are then input to the MCU via the $RxD_1$ pin. In transmission, the read data and status are synchronized with the fall of the transfer clock, and output from the $TxD_1$ pin.

The $TxD_1$ 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 $RTS_1$ (BUSY) pin is "H" level. Accordingly, always start the next transfer after the $RTS_1$ (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.

## Software Commands

Table 1.32.1 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 1.32.1. 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 | $FF_{16}$ | Address (middle) | Address (high) | Data output | Data output | Data output | Data output to 259th byte | Not acceptable |
| 2 | Page program | $41_{16}$ | Address (middle) | Address (high) | Data input | Data input | Data input | Data input to 259th byte | Not acceptable |
| 3 | Block erase | $20_{16}$ | Address (middle) | Address (high) | $D0_{16}$ | | | | Not acceptable |
| 4 | Erase all unlocked blocks | $A7_{16}$ | $D0_{16}$ | | | | | | Not acceptable |
| 5 | Read status register | $70_{16}$ | SRD output | SRD1 output | | | | | Acceptable |
| 6 | Clear status register | $50_{16}$ | | | | | | | Not acceptable |
| 7 | Read lock bit status | $71_{16}$ | Address (middle) | Address (high) | Lock bit data output | | | | Not acceptable |
| 8 | Lock bit program | $77_{16}$ | Address (middle) | Address (high) | $D0_{16}$ | | | | Not acceptable |
| 9 | Lock bit enable | $7A_{16}$ | | | | | | | Not acceptable |
| 10 | Lock bit disable | $75_{16}$ | | | | | | | Not acceptable |
| 11 | ID check function | $F5_{16}$ | Address (low) | Address (middle) | Address (high) | ID size | ID1 | To ID7 | Acceptable |
| 12 | Download function | $FA_{16}$ | Size (low) | Size (high) | Check-sum | Data input | To required number of times | | Not acceptable |
| 13 | Version data output function | $FB_{16}$ | 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 | $FC_{16}$ | Address (middle) | Address (high) | Data output | Data output | Data output | Data output to 259th byte | Not acceptable |
| 15 | Read check data | $FD_{16}$ | 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 "$FF_{16}$" command code with the 1st byte.

(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ with the 2nd and 3rd bytes respectively.

(3) From the 4th byte onward, data ($D_0$–$D_7$) for the page (256 bytes) specified with addresses $A_8$ to $A_{23}$ will be output sequentially from the smallest address first in sync with the fall of the clock.

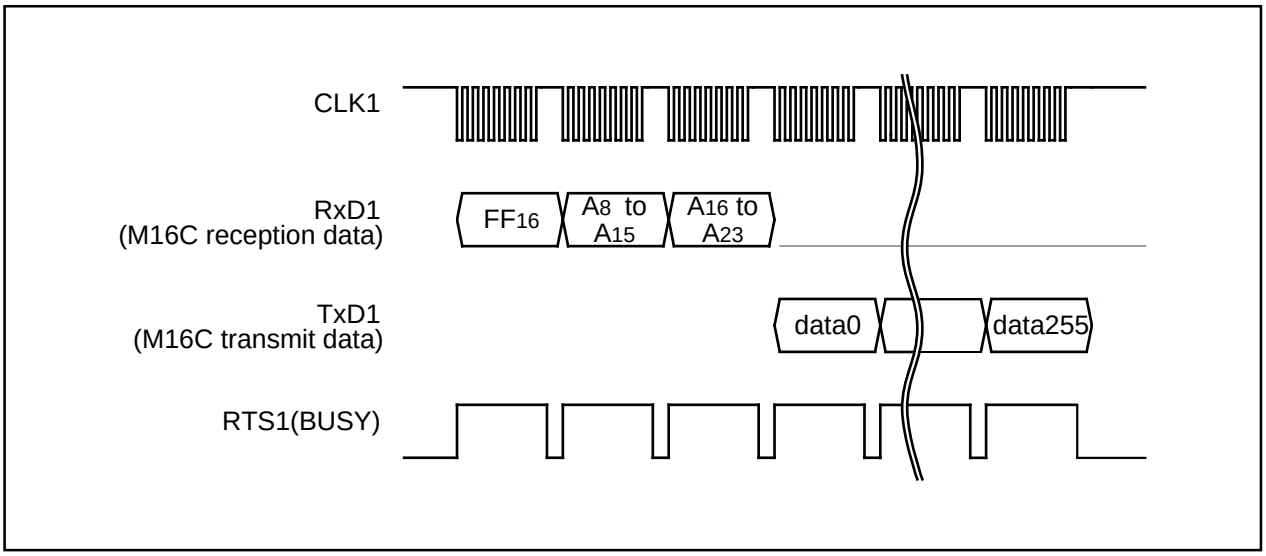



**Figure 1.32.2. Timing for page read**

**Read Status Register Command**

This command reads status information. When the "$70_{16}$" 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.

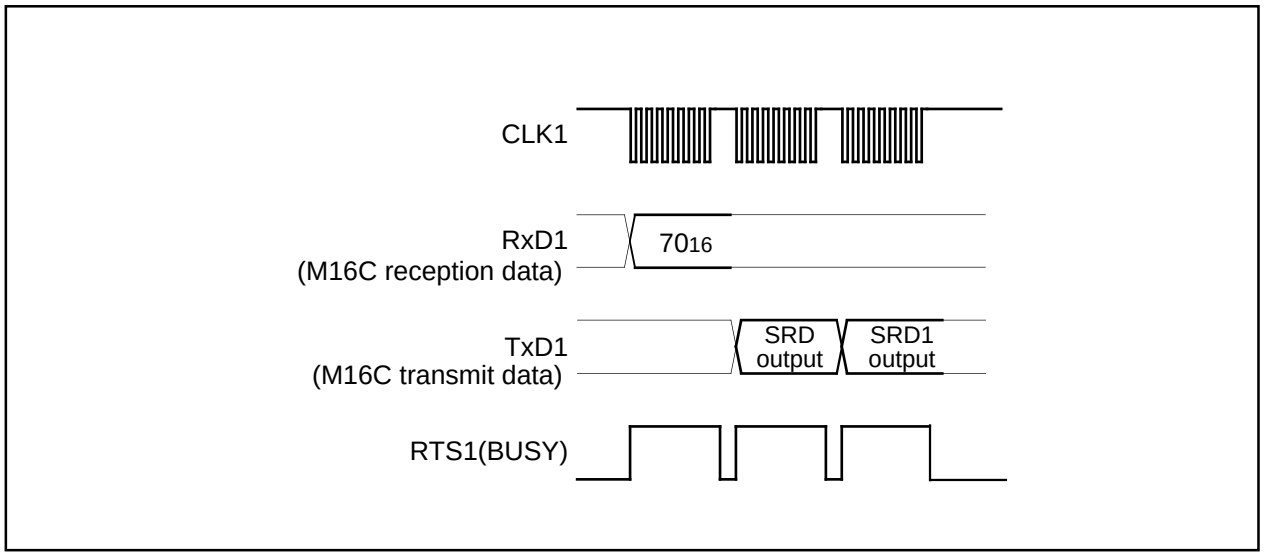



**Figure 1.32.3. Timing for reading the status register**

**Clear Status Register Command**

This command clears the bits (SR4, SR5) which are set when the status register operation ends in error. When the "$50_{16}$" command code is sent with the 1st byte, the aforementioned bits are cleared. When the clear status register operation ends, the $RTS_1$ (BUSY) signal changes from the "H" to the "L" level.

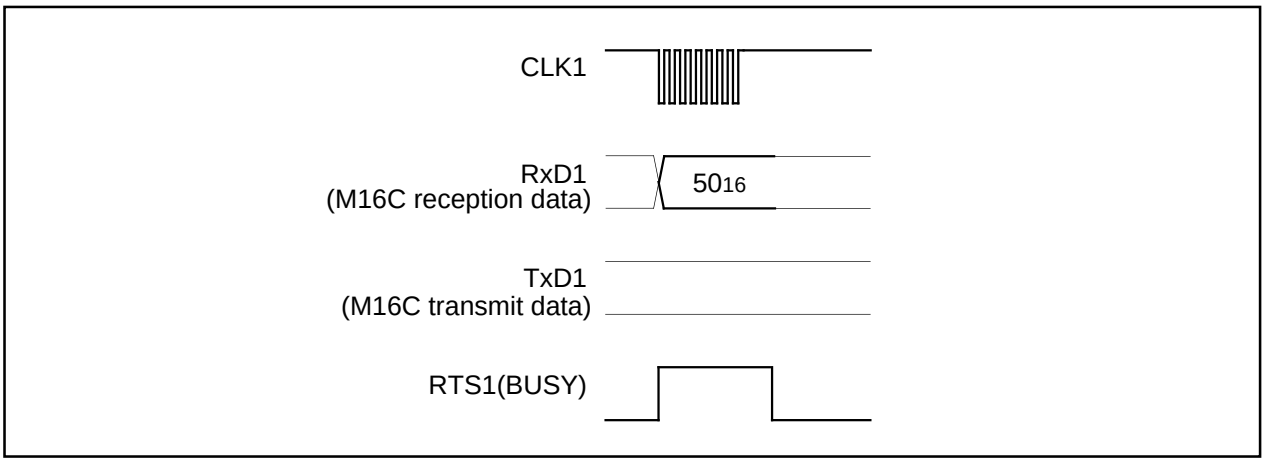



**Figure 1.32.4. Timing for clearing the status register**

**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 "$41_{16}$" command code with the 1st byte.
(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ with the 2nd and 3rd bytes respectively.
(3) From the 4th byte onward, as write data ($D_0$–$D_7$) for the page (256 bytes) specified with addresses $A_8$ to $A_{23}$ is input sequentially from the smallest address first, that page is automatically written.

When reception setup for the next 256 bytes ends, the $RTS_1$ (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.

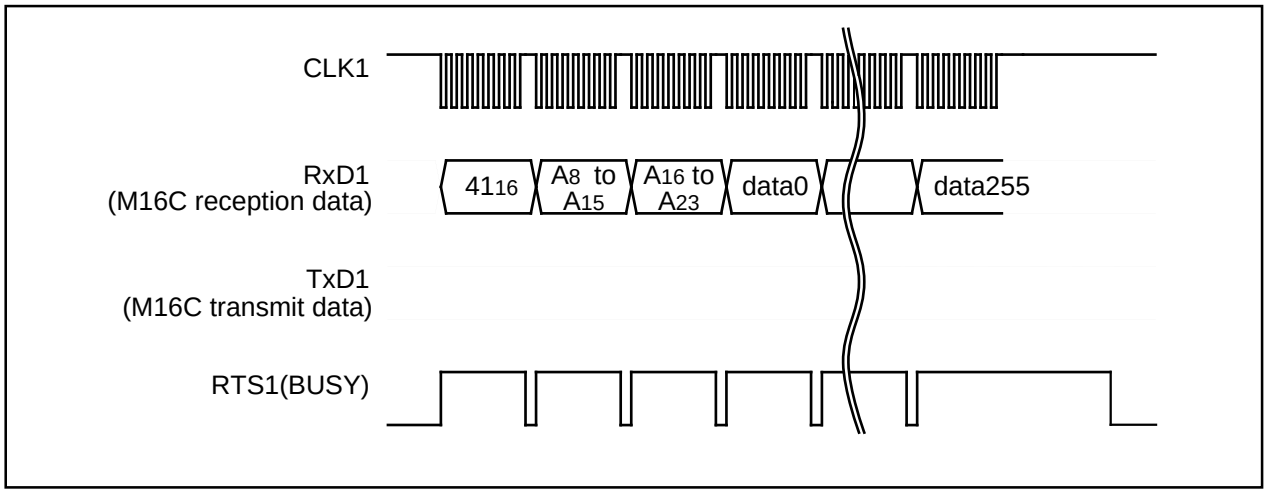



**Figure 1.32.5. Timing for the page program**

**Block Erase Command**

This command erases the data in the specified block. Execute the block erase command as explained here following.

(1) Transfer the “$20_{16}$” command code with the 1st byte.

(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ with the 2nd and 3rd bytes respectively.

(3) Transfer the verify command code “$D0_{16}$” 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 $A_8$ to $A_{23}$.

When block erasing ends, the $RTS_1$ (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.

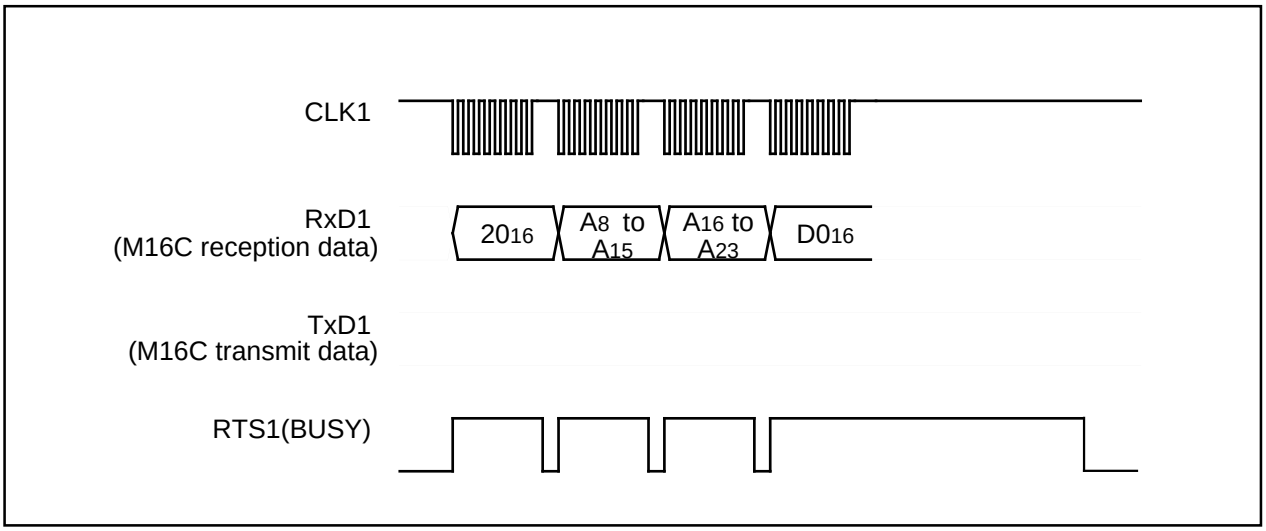



**Figure 1.32.6. Timing for block erasing**

**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 "$A7_{16}$" command code with the 1st byte.

(2) Transfer the verify command code "$D0_{16}$" 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 $RTS_1$ (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.

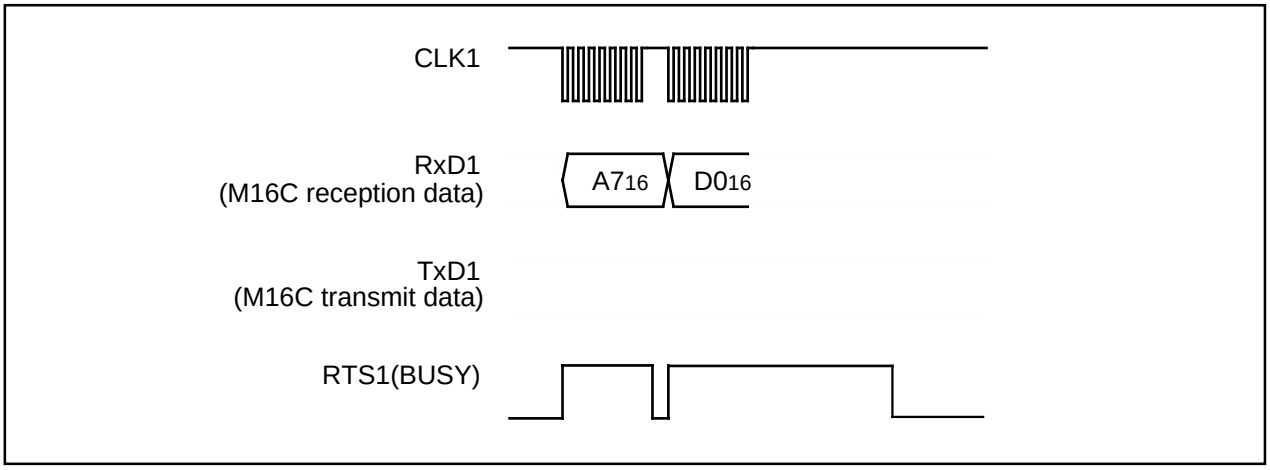



**Figure 1.32.7. Timing for erasing all unlocked blocks**

**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 "$77_{16}$" command code with the 1st byte.

(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ with the 2nd and 3rd bytes respectively.

(3) Transfer the verify command code "$D0_{16}$" 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 $A_8$ to $A_{23}$.

When writing ends, the $RTS_1$ (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.

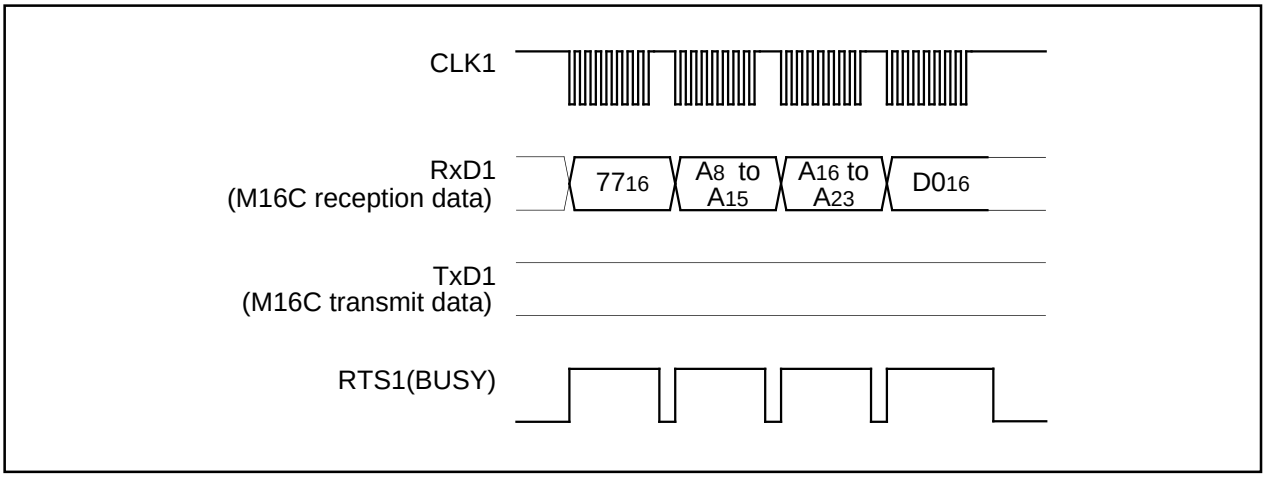



**Figure 1.32.8 Timing for the lock bit program**

**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 "$71_{16}$" command code with the 1st byte.

(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ 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 $A_8$ to $A_{23}$.

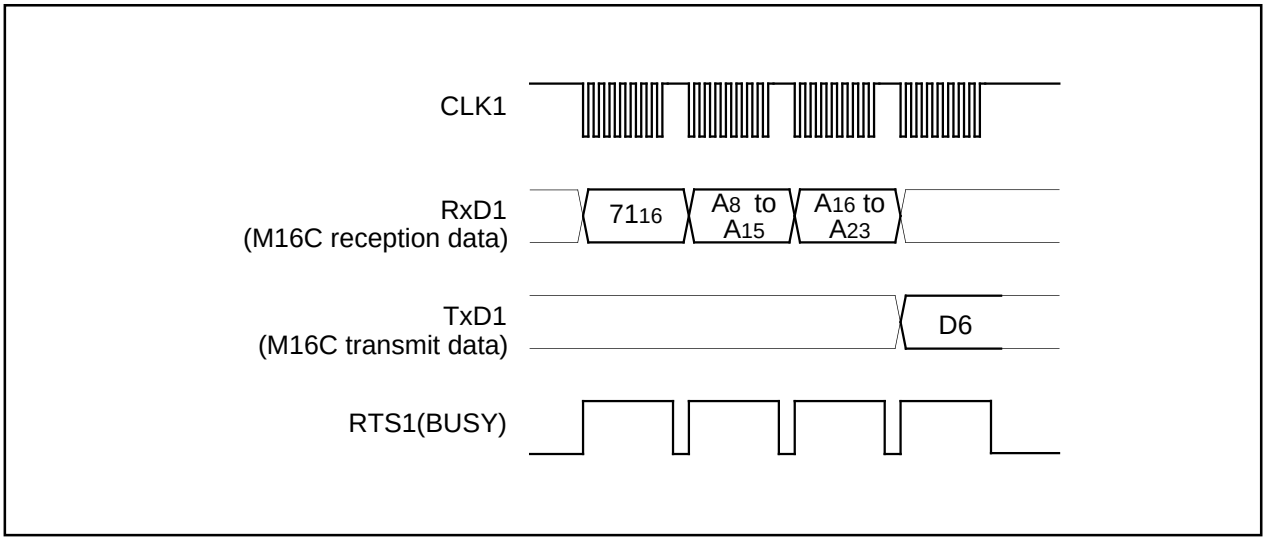



**Figure 1.32.9. Timing for reading lock bit status**

**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 "$7A_{16}$" 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.

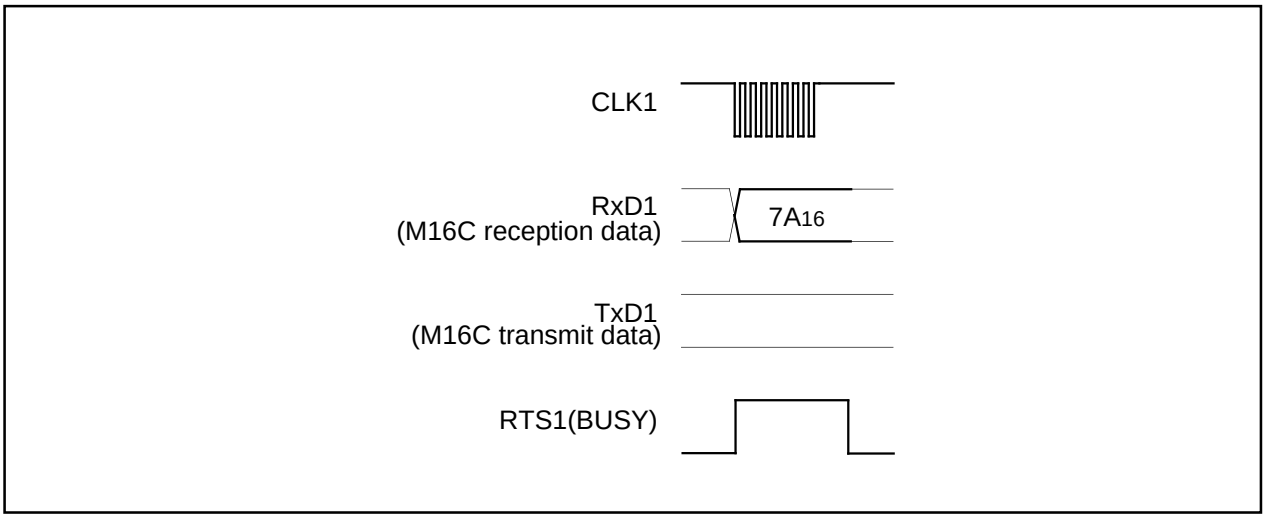



**Figure 1.32.10. Timing for enabling the lock bit**

**Lock Bit Disable Command**

This command disables the lock bit. The command code “75$_{16}$” 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.

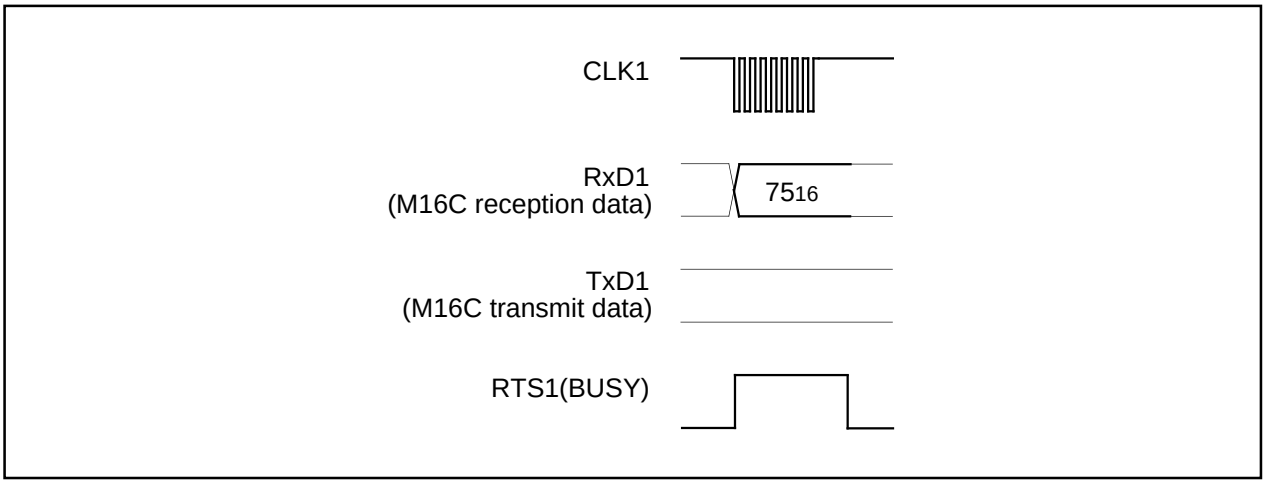



**Figure 1.32.11. Timing for disabling the lock bit**

**Download Command**

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

(1) Transfer the “FA$_{16}$” 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.

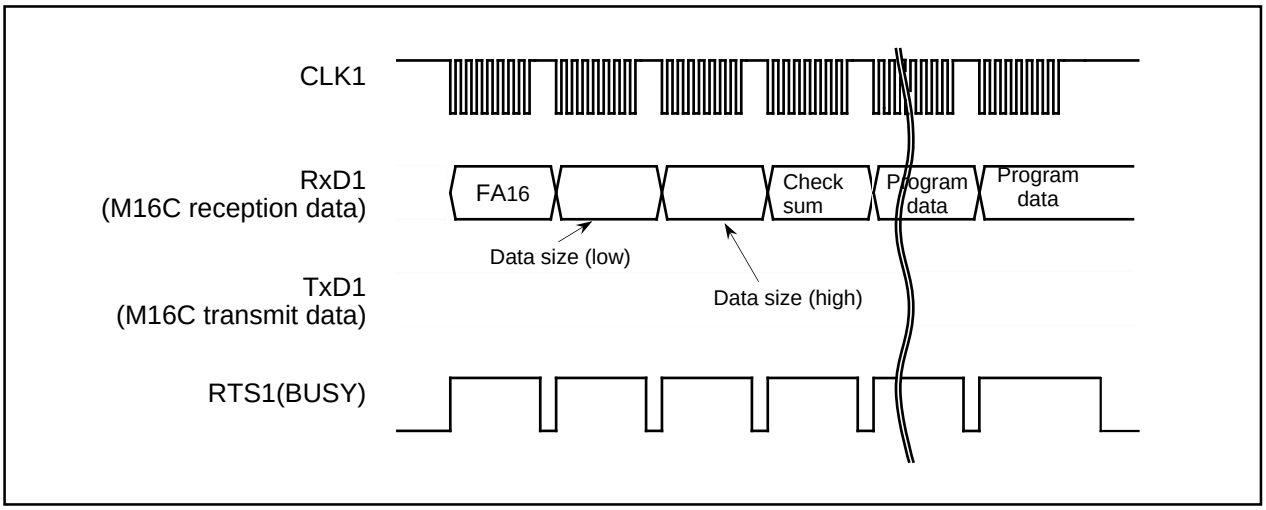



**Figure 1.32.12. Timing for download**

**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 “$FB_{16}$” 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.

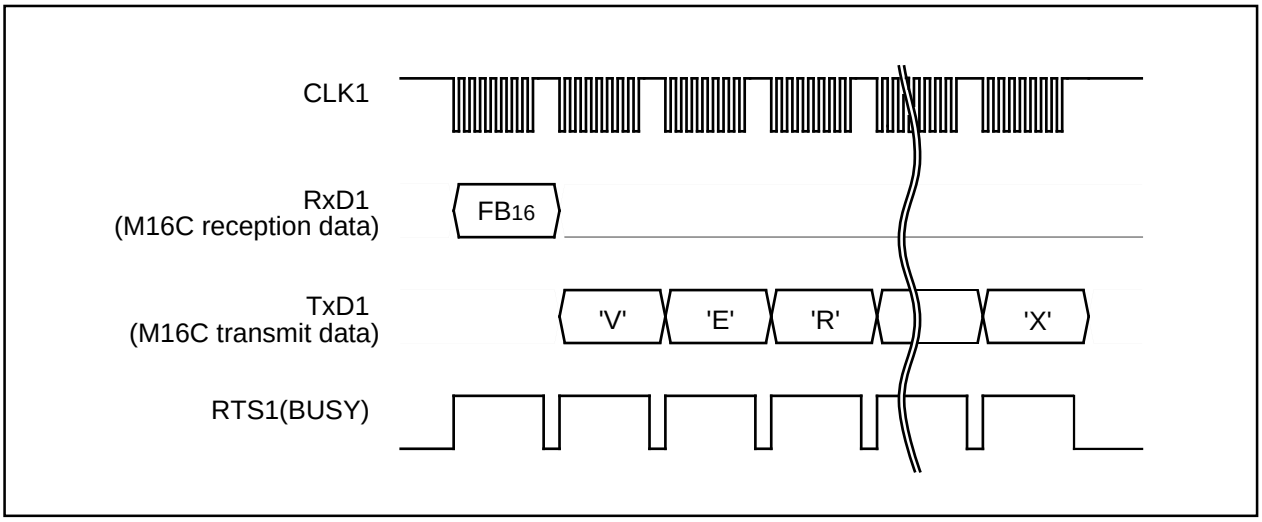



**Figure 1.32.13. Timing for version information output**

**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 “$FC_{16}$” command code with the 1st byte.

(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ with the 2nd and 3rd bytes respectively.

(3) From the 4th byte onward, data ($D_0$–$D_7$) for the page (256 bytes) specified with addresses $A_8$ to $A_{23}$ will be output sequentially from the smallest address first, in sync with the fall of the clock.

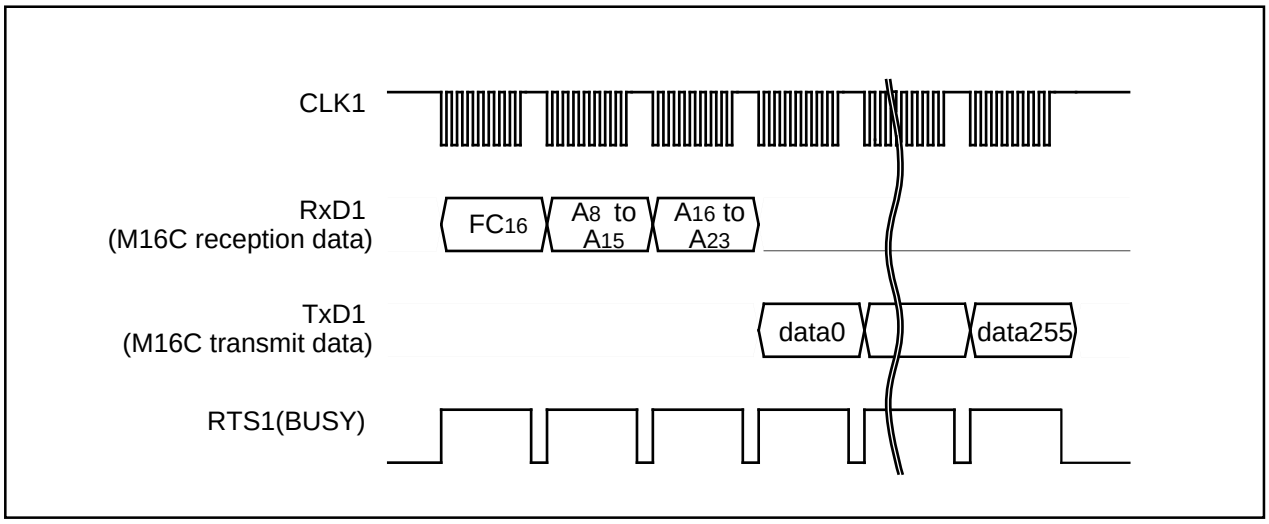



**Figure 1.32.14. Timing for boot ROM area output**

**ID Check**

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

(1) Transfer the “F5$_{16}$” command code with the 1st byte.

(2) Transfer addresses A$_0$ to A$_7$, A$_8$ to A$_{15}$ and A$_{16}$ to A$_{23}$ 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.

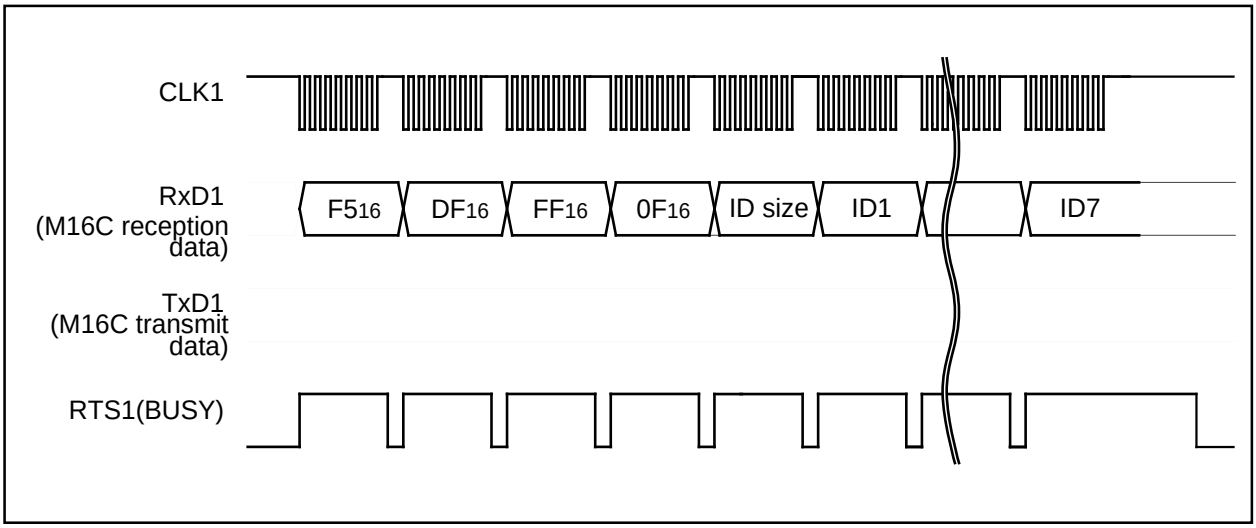



**Figure 1.32.15. Timing for the ID check**

**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 0FFFDF$_{16}$, 0FFFE3$_{16}$, 0FFFEB$_{16}$, 0FFFEF$_{16}$, 0FFFF3$_{16}$, 0FFFF7$_{16}$ and 0FFFFB$_{16}$. Write a program into the flash memory, which already has the ID code set for these addresses.

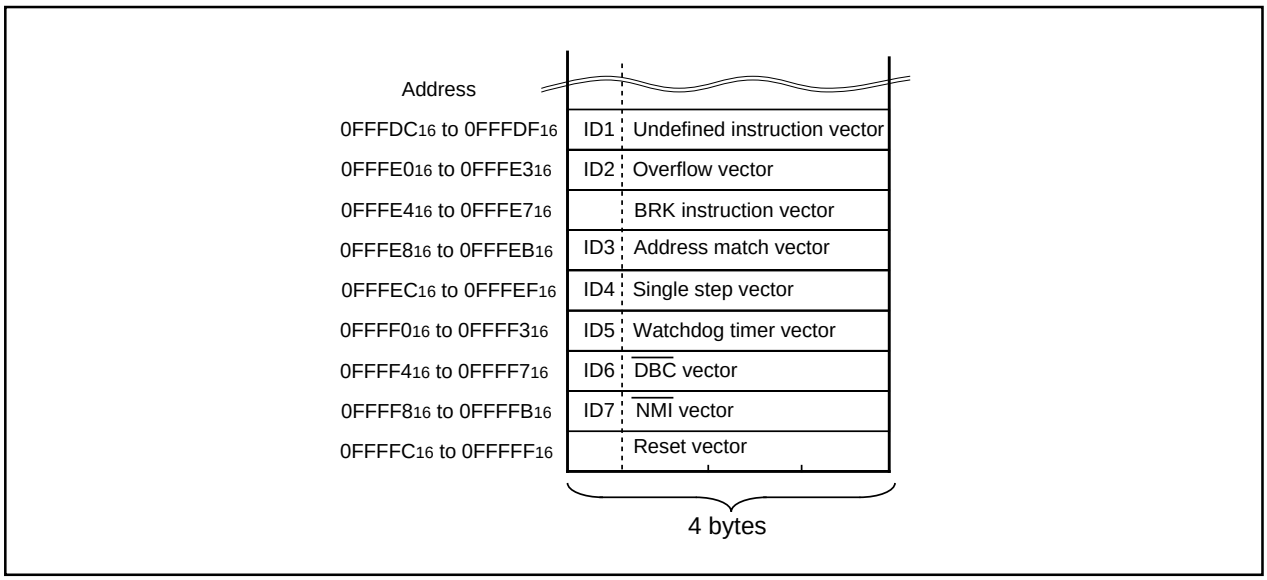



**Figure 1.32.16. ID code storage addresses**

**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 "$FD_{16}$" 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.

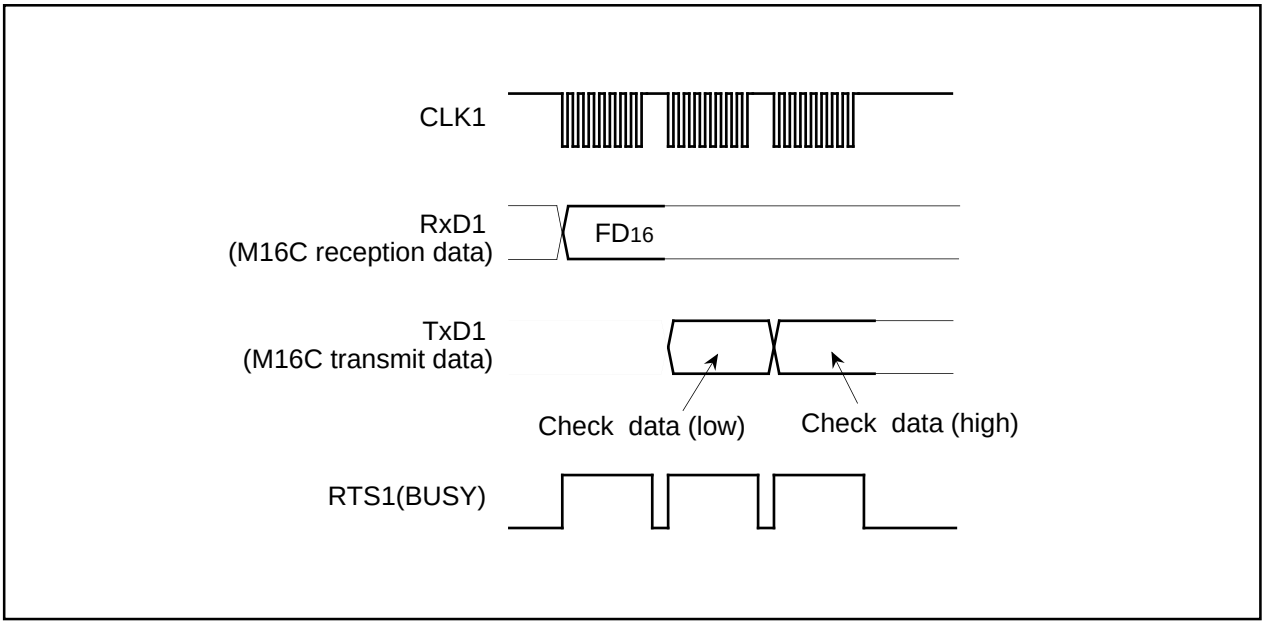



**Figure 1.32.17. Timing for the read check data**

## Data Protection (Block Lock)

Each of the blocks in Figure 1.32.18 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” before the block was erased is set to “1” (unlocked) after erasing, therefore the block is actually unlocked with the lock bit.

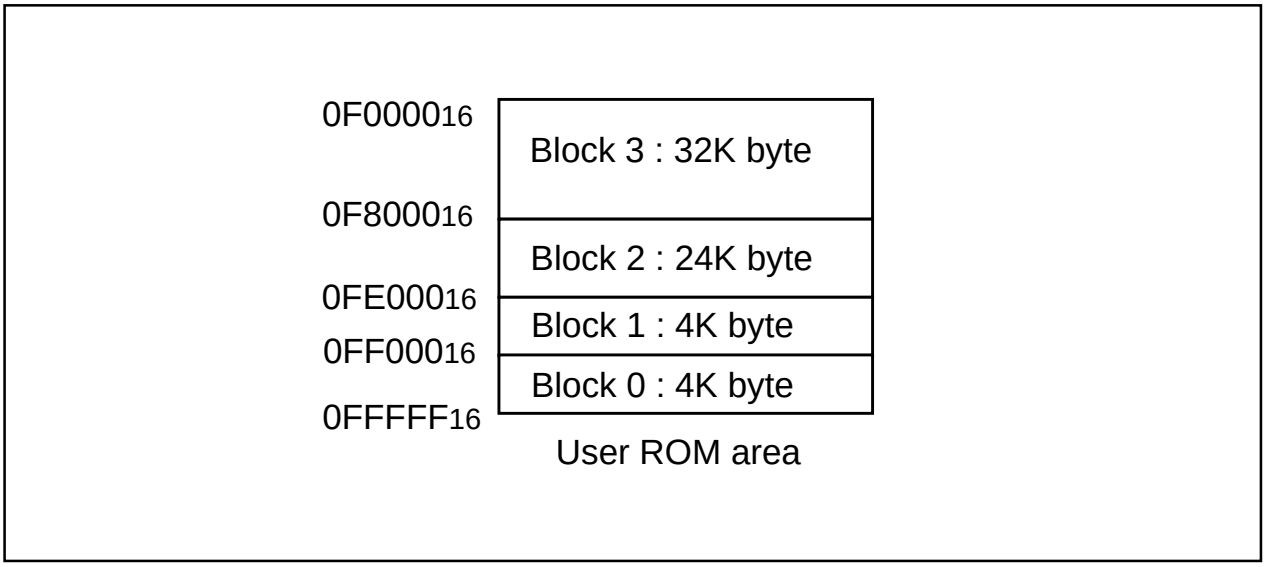



**Figure 1.32.18. Blocks in the user area**

## 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 ($70_{16}$). Also, the status register is cleared by writing the clear status register command ($50_{16}$). Table 1.32.2 gives the definition of each status register bit. After clearing the reset, the status register outputs "$80_{16}$".

**Table 1.32.2. Status register (SRD)**

| SRD0 bits | Status name | Definition | |
|---|---|---|---|
| | | "1" | "0" |
| SR7 (bit7) | Sequencer 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) | Reserved | - | - |
| SR2 (bit2) | Reserved | - | - |
| SR1 (bit1) | Reserved | - | - |
| SR0 (bit0) | Reserved | - | - |

**Sequencer status (SR7)**

After power-on, the sequencer status is set to 1(ready).

The sequencer status indicates the operating status of the device. This status bit is set to "0" (busy) during write or erase operation and is set to 1 upon completion of these operations.

**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".

## 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 ($70_{16}$). Also, status register 1 is cleared by writing the clear status register command ($50_{16}$).
Table 1.32.3 gives the definition of each status register 1 bit. "$00_{16}$" is output when power is turned ON and the flag status is maintained even after the reset.

**Table 1.32.3. Status register 1 (SRD1)**

| SRD1 bits | Status name | Definition | |
|---|---|---|---|
| | | "1" | "0" |
| SR15 (bit7) | Boot update completed bit | Update completed | Not update |
| SR14 (bit6) | Flash identification value | HND | DINOR |
| SR13 (bit5) | Reserved | - | - |
| SR12 (bit4) | Check sum match bit | Match | Mismatch |
| SR11 (bit3)<br>SR10 (bit2) | ID check completed bits | 00 Not verified<br>01 Verification mismatch<br>10 Reserved<br>11 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.

**Flash Identification Value (SR14)**

This flag indicates whether the flash memor type is HND or DINOR.

**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)**

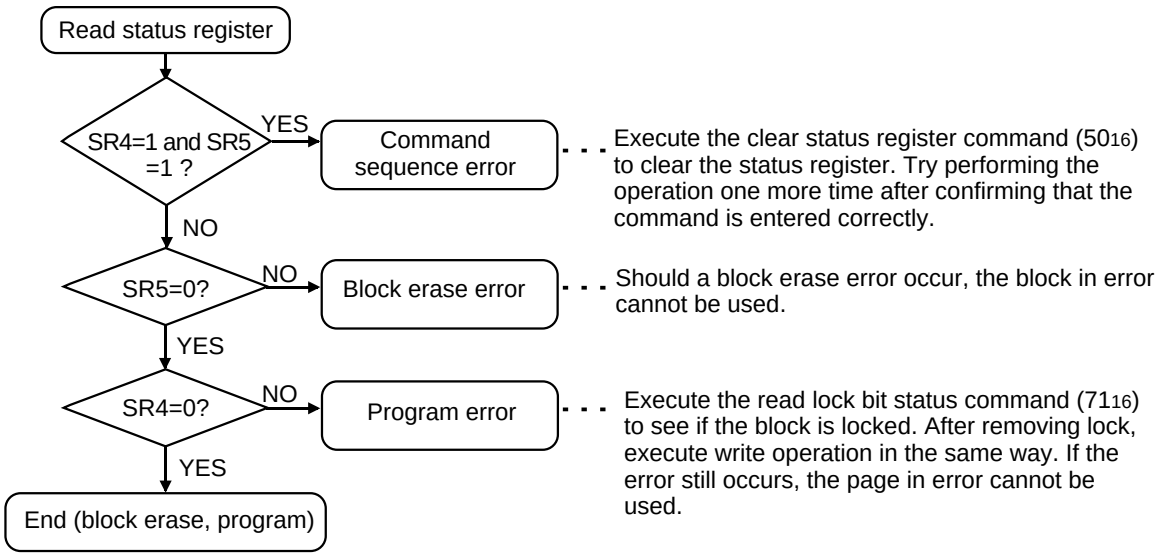
These flags indicate the result of ID checks. Some commands cannot be accepted without an ID check.

**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 1.32.19 shows a flowchart of the full status check and explains how to remedy errors which occur.

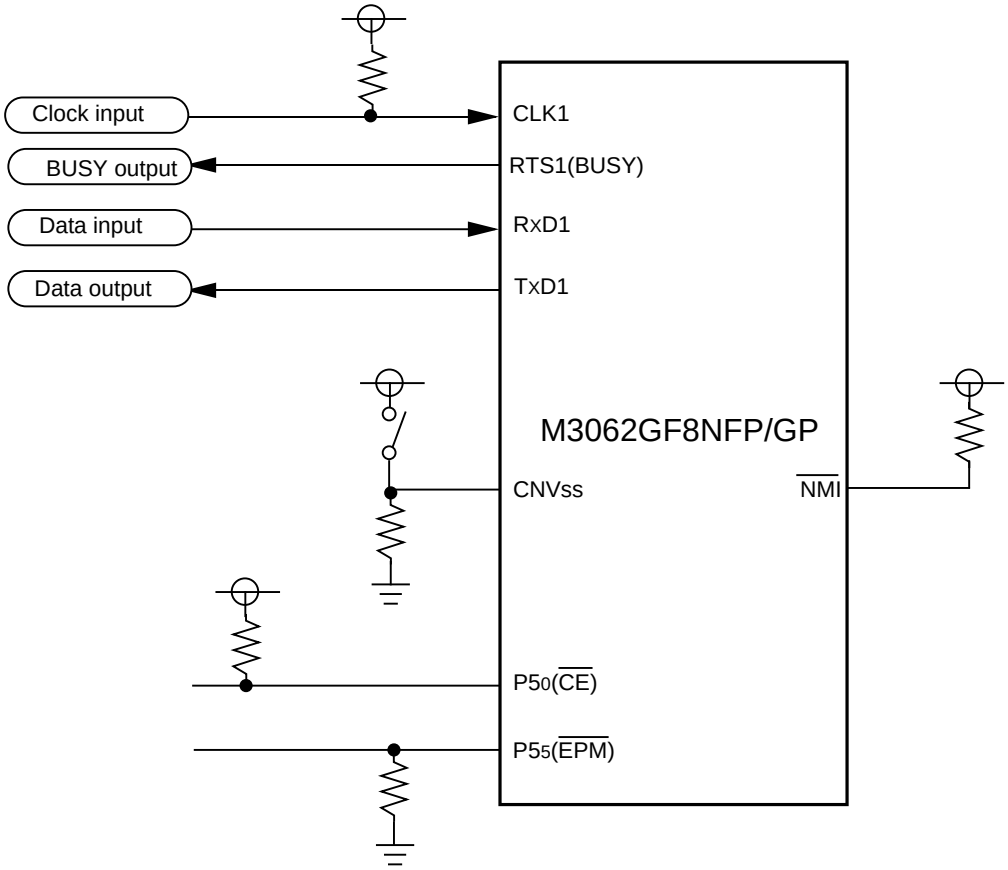



Note: When one of SR5 to SR4 is set to 1, none of the program, erase all blocks, and block erase commands is accepted. Execute the clear status register command ($50_{16}$) before executing these commands.

**Figure 1.32.19. 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.




(1) Control pins and external circuitry will vary according to peripheral unit. For more information, see the peripheral unit manual.
(2) In this example, the microprocessor mode and standard serial I/O mode are switched via a switch.

**Figure 1.32.20. Example circuit application for the standard serial I/O mode 1**

## Overview of standard serial I/O mode 2 (clock asynchronized)

In standard serial I/O mode 2, software commands, addresses and data are input and output between the MCU and peripheral units (serial programer, etc.) using 2-wire clock-asynchronized serial I/O (UART1). Standard serial I/O mode 2 is engaged by releasing the reset with the $P6_5$ ($CLK_1$) pin "L" level.
The $TxD_1$ 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 1.32.21) 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 1.32.21).

(1) Transmit "$B0_{16}$" 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 "$B0_{16}$" check code. If the oscillation frequency is anything other than 10 or 16 MHz, the MCU does not output anything.

(2) Transmit "$00_{16}$" from a peripheral unit 16 times. (The MCU with internal flash memory sets the bit rate generator so that "$00_{16}$" can be successfully received.)

(3) The MCU with internal flash memory outputs the "$B0_{16}$" 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 "$B0_{16}$" successfully, change the oscillation frequency of the main clock.

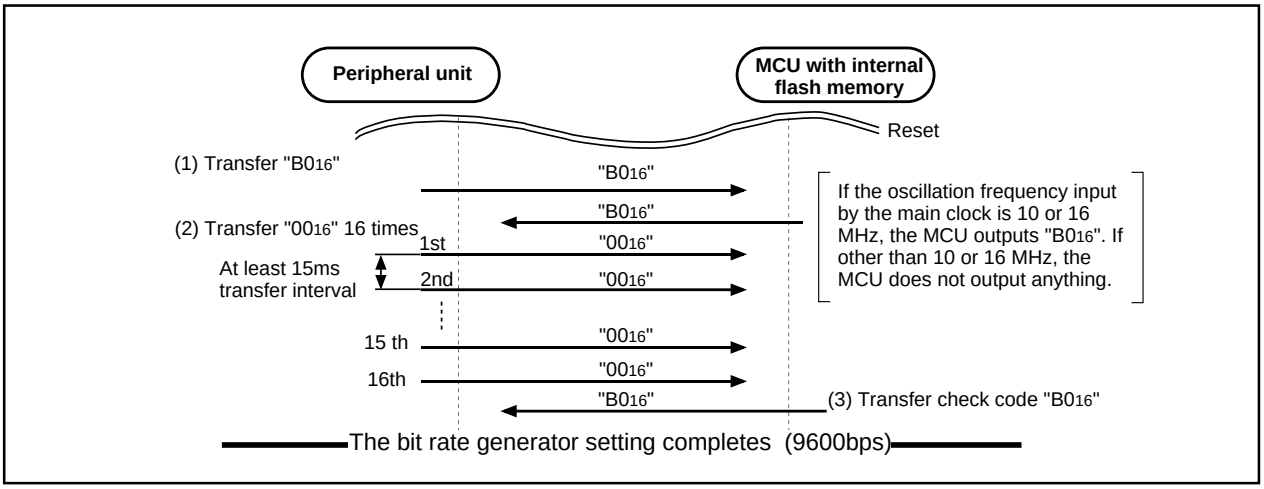



**Figure 1.32.21. Peripheral unit and initial communication**

## How frequency is identified

When "00$_{16}$" 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 1.32.4 gives the operation frequency and the baud rate that can be attained for.

**Table 1.32.4 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 |
|---|---|---|---|---|
| (Note) | √ | √ | √ | √ |
| (Note) | √ | √ | √ | – |
| (Note) | √ | √ | √ | – |
| 10MHz | √ | √ | – | √ |
| 8MHz | √ | √ | – | √ |
| 7.3728MHz | √ | √ | √ | √ |
| 6MHz | √ | √ | √ | – |
| 5MHz | √ | √ | – | – |
| 4.5MHz | √ | √ | – | √ |
| 4.194304MHz | √ | √ | √ | – |
| 4MHz | √ | √ | – | – |
| 3.58MHz | √ | √ | √ | √ |
| 3MHz | √ | √ | √ | – |
| 2MHz | √ | – | – | – |

Note : Operating maximum frequency of a microcomputer is 10 MHz.

√ : Communications possible
– : Communications not possible

## Software Commands

Table 1.32.5 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 1.32.5. 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 | $FF_{16}$ | Address (middle) | Address (high) | Data output | Data output | Data output | Data output to 259th byte | Not acceptable |
| 2 | Page program | $41_{16}$ | Address (middle) | Address (high) | Data input | Data input | Data input | Data input to 259th byte | Not acceptable |
| 3 | Block erase | $20_{16}$ | Address (middle) | Address (high) | $D0_{16}$ | | | | Not acceptable |
| 4 | Erase all unlocked blocks | $A7_{16}$ | $D0_{16}$ | | | | | | Not acceptable |
| 5 | Read status register | $70_{16}$ | SRD output | SRD1 output | | | | | Acceptable |
| 6 | Clear status register | $50_{16}$ | | | | | | | Not acceptable |
| 7 | Read lock bit status | $71_{16}$ | Address (middle) | Address (high) | Lock bit data output | | | | Not acceptable |
| 8 | Lock bit program | $77_{16}$ | Address (middle) | Address (high) | $D0_{16}$ | | | | Not acceptable |
| 9 | Lock bit enable | $7A_{16}$ | | | | | | | Not acceptable |
| 10 | Lock bit disable | $75_{16}$ | | | | | | | Not acceptable |
| 11 | ID check function | $F5_{16}$ | Address (low) | Address (middle) | Address (high) | ID size | ID1 | To ID7 | Acceptable |
| 12 | Download function | $FA_{16}$ | Size (low) | Size (high) | Check-sum | Data input | To required number of times | | Not acceptable |
| 13 | Version data output function | $FB_{16}$ | 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 | $FC_{16}$ | Address (middle) | Address (high) | Data output | Data output | Data output | Data output to 259th byte | Not acceptable |
| 15 | Read check data | $FD_{16}$ | Check data (low) | Check data (high) | | | | | Not acceptable |
| 16 | Baud rate 9600 | $B0_{16}$ | $B0_{16}$ | | | | | | Acceptable |
| 17 | Baud rate 19200 | $B1_{16}$ | $B1_{16}$ | | | | | | Acceptable |
| 18 | Baud rate 38400 | $B2_{16}$ | $B2_{16}$ | | | | | | Acceptable |
| 19 | Baud rate 57600 | $B3_{16}$ | $B3_{16}$ | | | | | | 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 "$FF_{16}$" command code with the 1st byte.

(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ with the 2nd and 3rd bytes respectively.

(3) From the 4th byte onward, data ($D_0$–$D_7$) for the page (256 bytes) specified with addresses $A_8$ to $A_{23}$ will be output sequentially from the smallest address first.

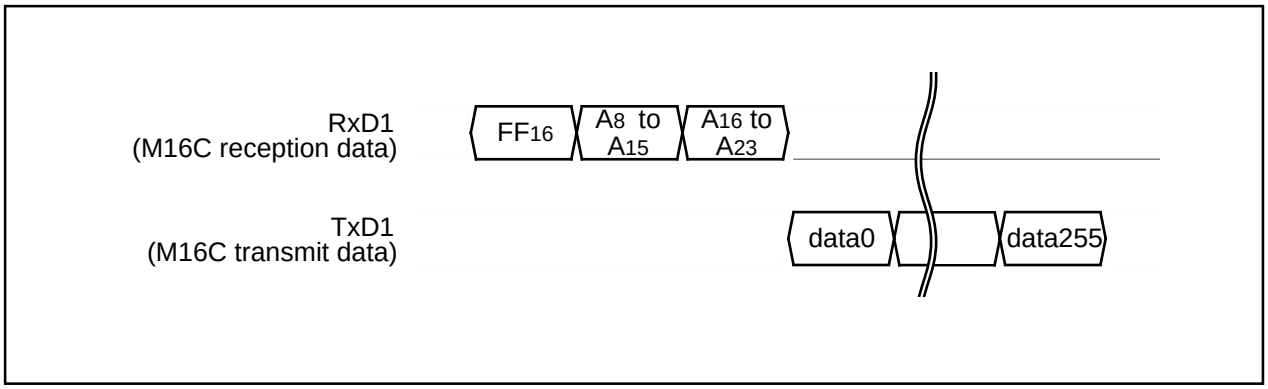



**Figure 1.32.22. Timing for page read**

**Read Status Register Command**

This command reads status information. When the "$70_{16}$" 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.

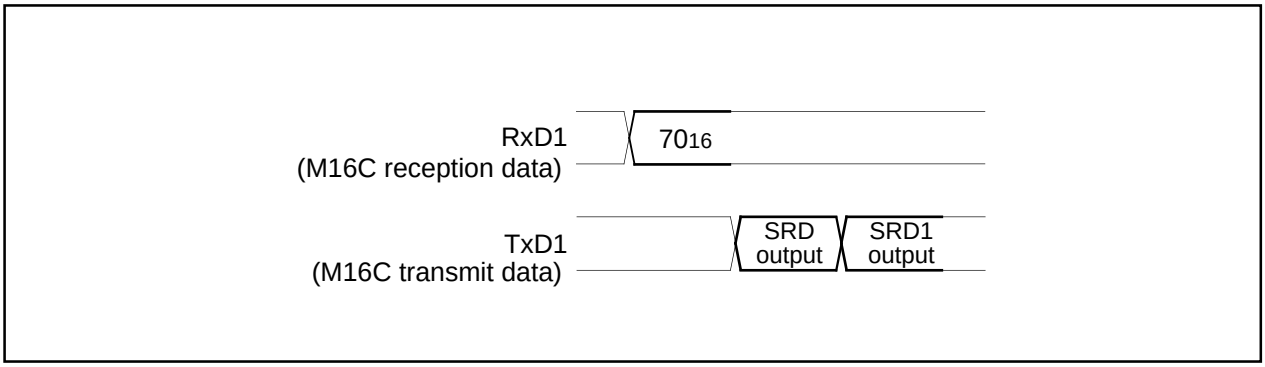



**Figure 1.32.23. Timing for reading the status register**

**Clear Status Register Command**

This command clears the bits (SR4, SR5) which are set when the status register operation ends in error. When the “$50_{16}$” command code is sent with the 1st byte, the aforementioned bits are cleared.

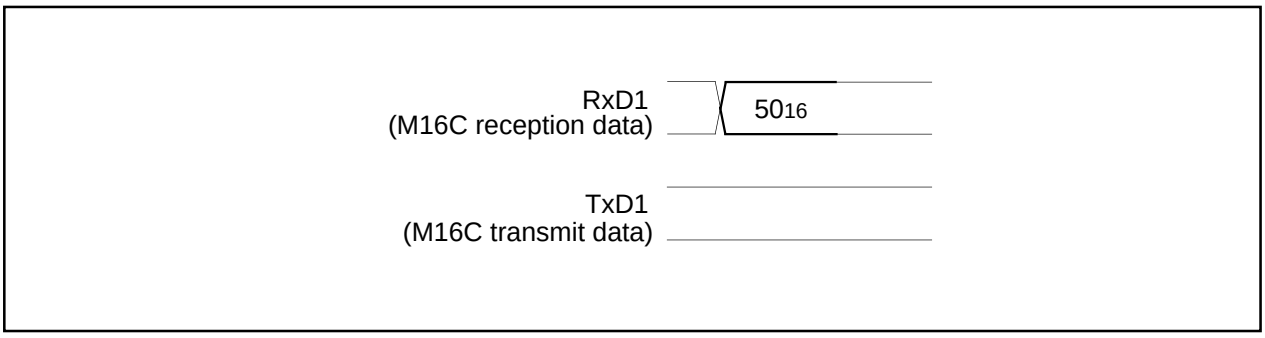



**Figure 1.32.24. Timing for clearing the status register**

**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 “$41_{16}$” command code with the 1st byte.

(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ with the 2nd and 3rd bytes respectively.

(3) From the 4th byte onward, as write data ($D_0$–$D_7$) for the page (256 bytes) specified with addresses $A_8$ to $A_{23}$ 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.

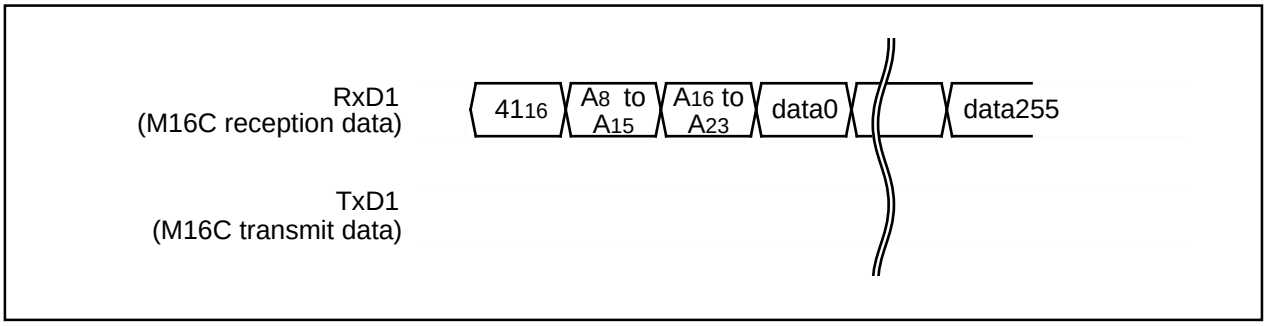



**Figure 1.32.25. Timing for the page program**

**Block Erase Command**

This command erases the data in the specified block. Execute the block erase command as explained here following.

(1) Transfer the “$20_{16}$” command code with the 1st byte.

(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ with the 2nd and 3rd bytes respectively.

(3) Transfer the verify command code “$D0_{16}$” 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 $A_8$ to $A_{23}$.

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.

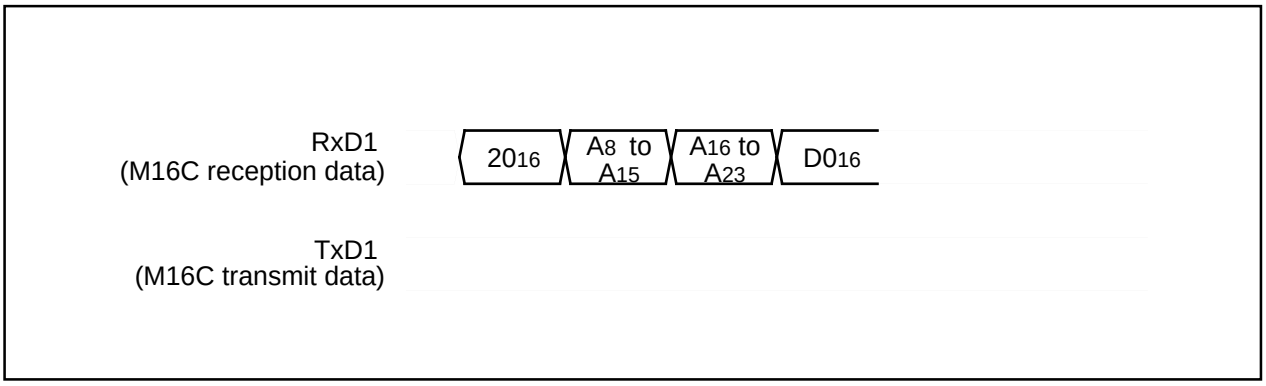



**Figure 1.32.26. Timing for block erasing**

**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 “$A7_{16}$” command code with the 1st byte.

(2) Transfer the verify command code “$D0_{16}$” 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.

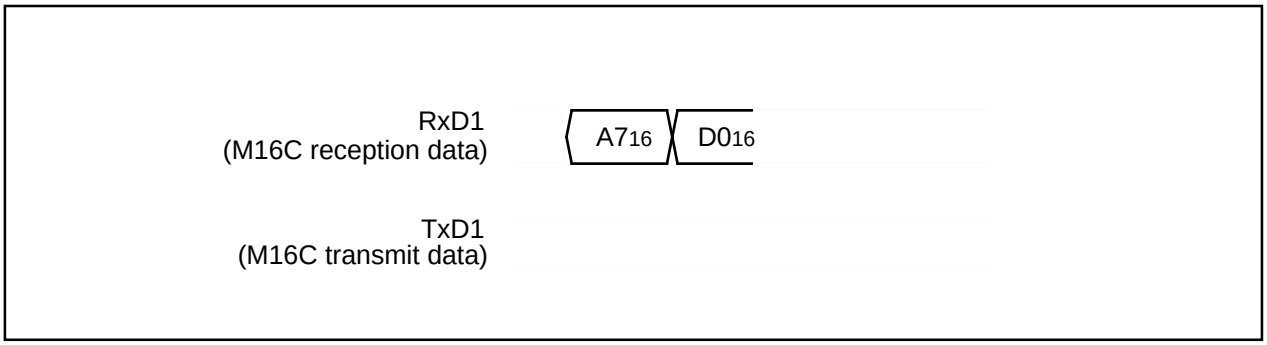



**Figure 1.32.27. Timing for erasing all unlocked blocks**

**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 “$77_{16}$” command code with the 1st byte.

(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ with the 2nd and 3rd bytes respectively.

(3) Transfer the verify command code “$D0_{16}$” 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 $A_8$ to $A_{23}$.

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.

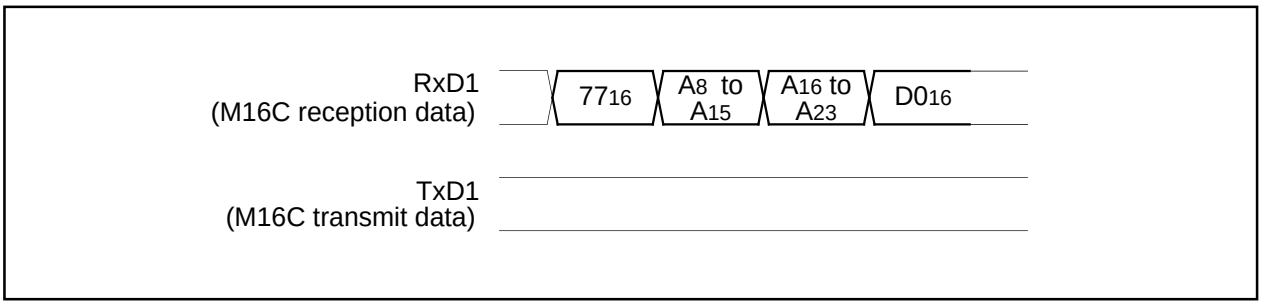



**Figure 1.32.28. Timing for the lock bit program**

**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 “$71_{16}$” command code with the 1st byte.

(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ 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 $A_8$ to $A_{23}$.

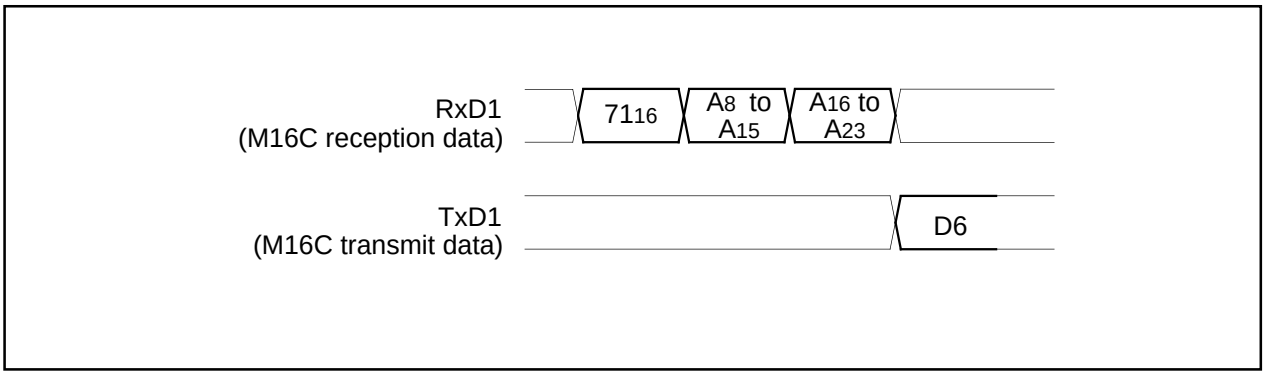



**Figure 1.32.29. Timing for reading lock bit status**

**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 “$7A_{16}$” 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.

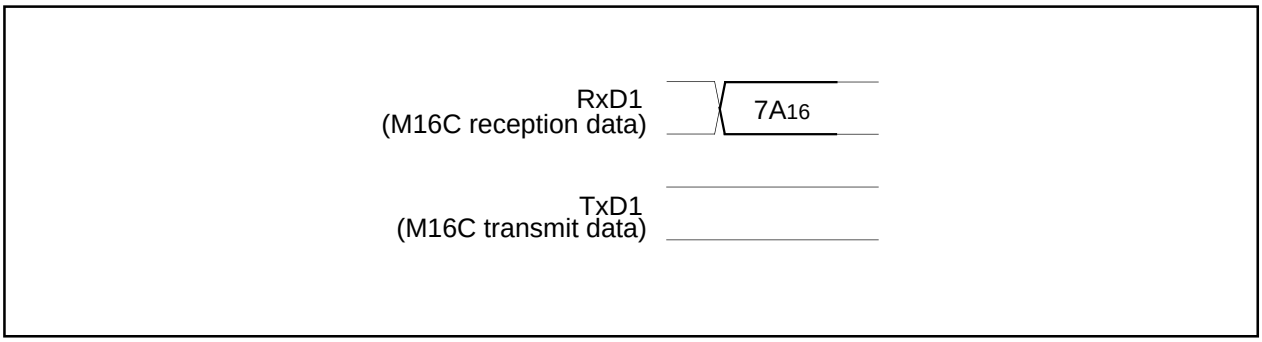



**Figure 1.32.30. Timing for enabling the lock bit**

**Lock Bit Disable Command**

This command disables the lock bit. The command code “75$_{16}$” 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.

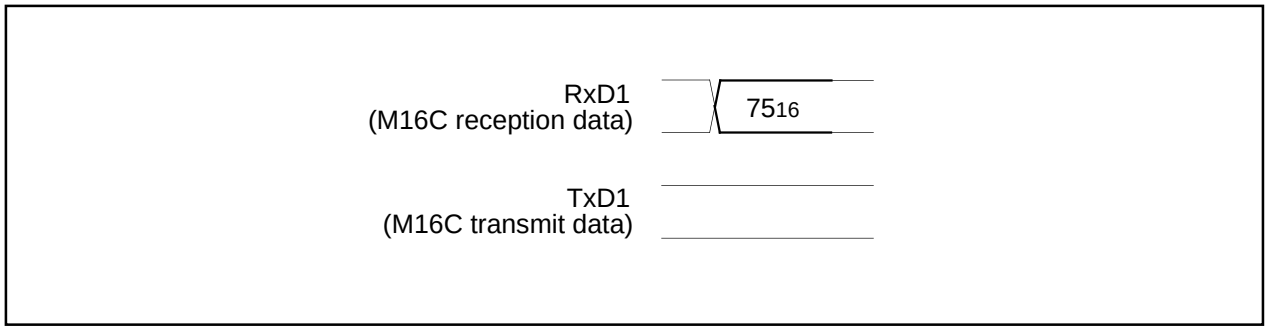



**Figure 1.32.31. Timing for disabling the lock bit**

**Download Command**

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

(1) Transfer the “FA$_{16}$” 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.

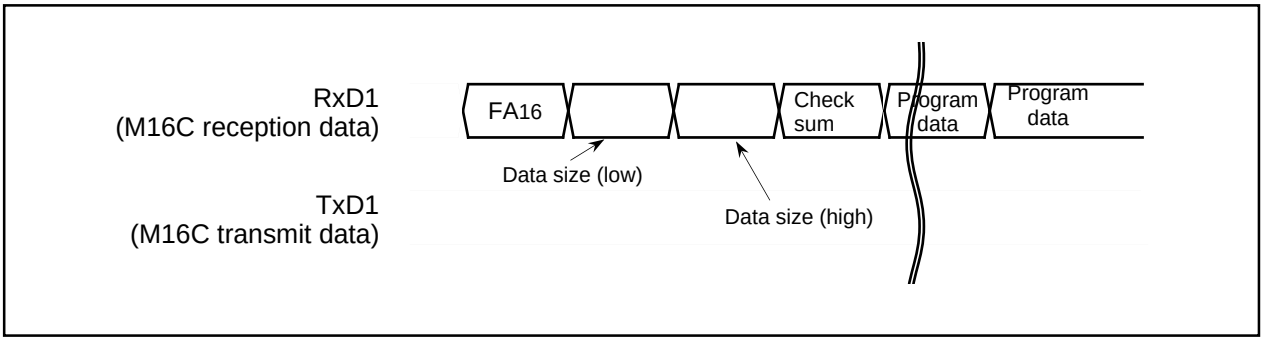



**Figure 1.32.32. Timing for download**

**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 “$FB_{16}$” 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.

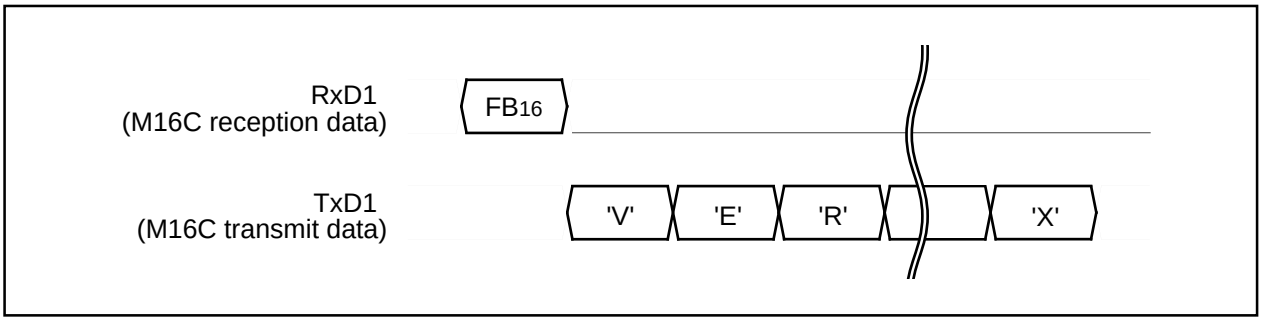



**Figure 1.32.33. Timing for version information output**

**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 “$FC_{16}$” command code with the 1st byte.

(2) Transfer addresses $A_8$ to $A_{15}$ and $A_{16}$ to $A_{23}$ with the 2nd and 3rd bytes respectively.

(3) From the 4th byte onward, data ($D_0$–$D_7$) for the page (256 bytes) specified with addresses $A_8$ to $A_{23}$ will be output sequentially from the smallest address first.

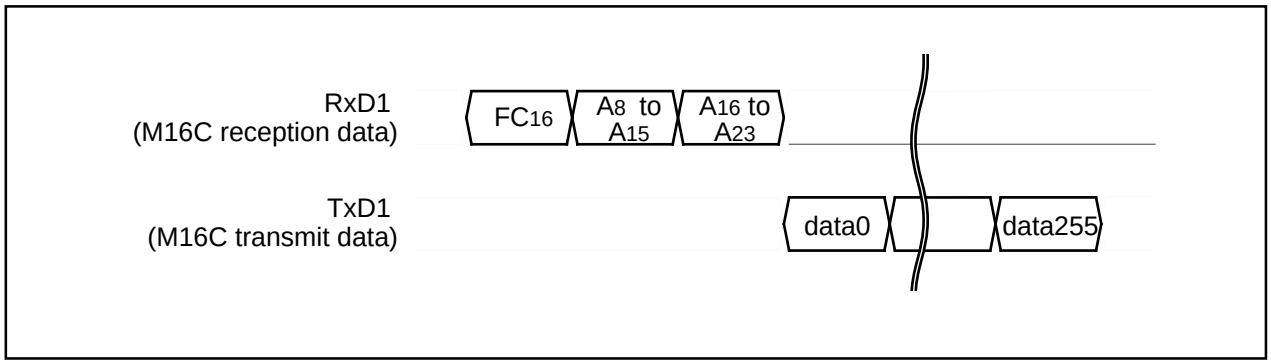



**Figure 1.32.34. Timing for boot ROM area output**

**ID Check**

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

(1) Transfer the “F5$_{16}$” command code with the 1st byte.

(2) Transfer addresses A$_0$ to A$_7$, A$_8$ to A$_{15}$ and A$_{16}$ to A$_{23}$ 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.

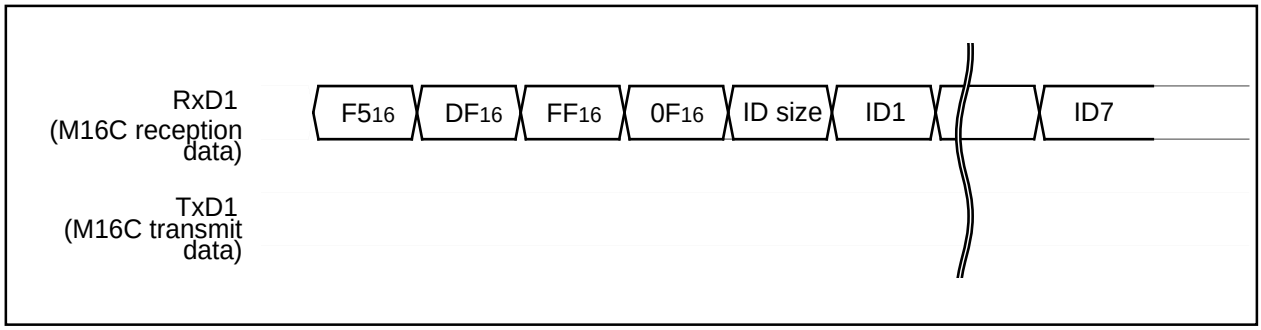



**Figure 1.32.35. Timing for the ID check**

**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 0FFFDF$_{16}$, 0FFFE3$_{16}$, 0FFFEB$_{16}$, 0FFFEF$_{16}$, 0FFFF3$_{16}$, 0FFFF7$_{16}$ and 0FFFFB$_{16}$. Write a program into the flash memory, which already has the ID code set for these addresses.

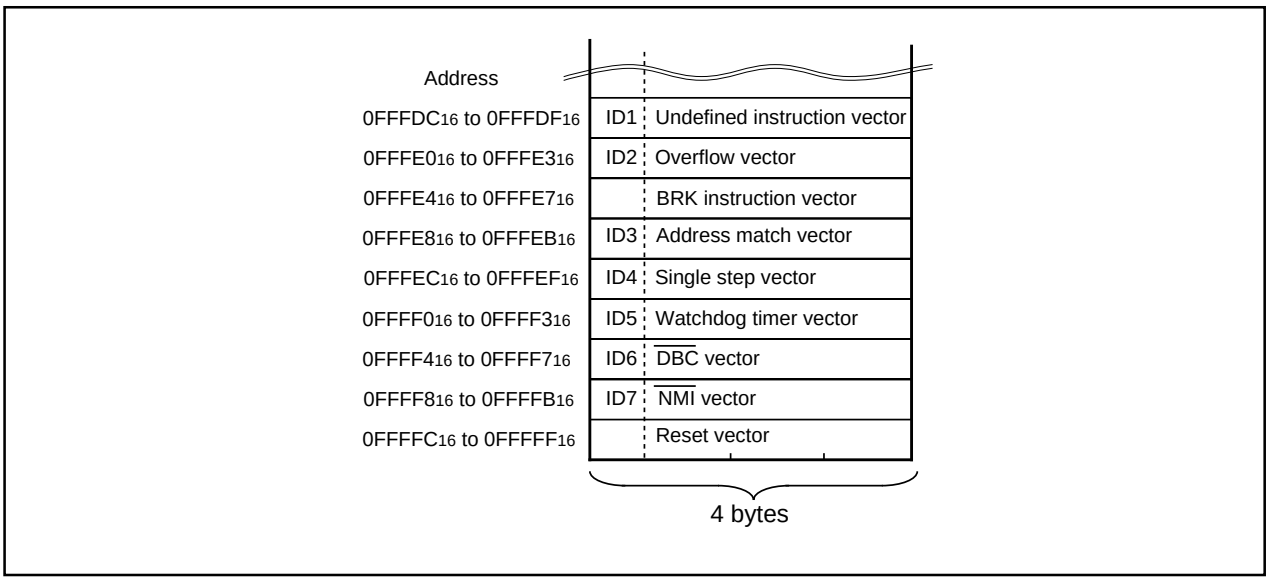



**Figure 1.32.36. ID code storage addresses**

**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.

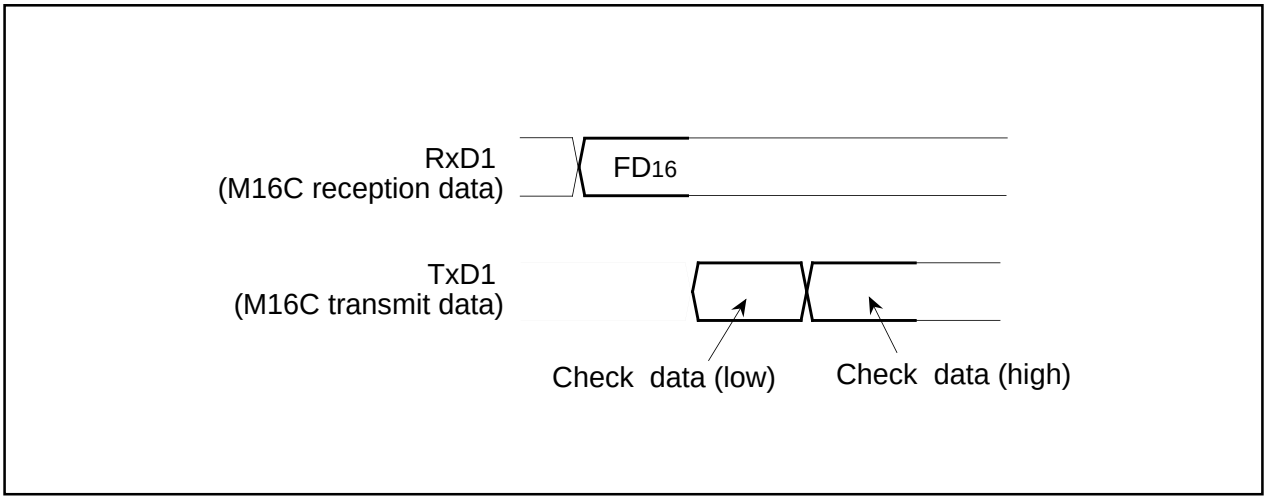



**Figure 1.32.37. Timing for the read check data**

**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.

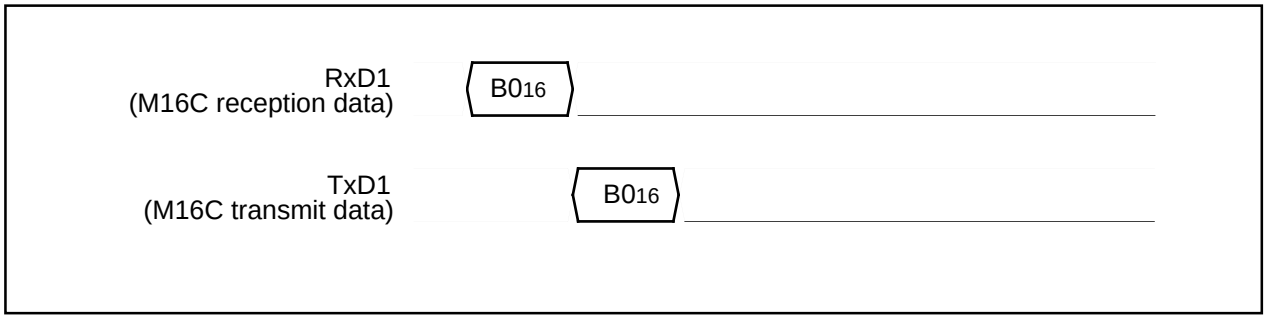



**Figure 1.32.38. Timing of baud rate 9600**

**Baud Rate 19200**

This command changes baud rate to 19,200 bps. Execute it as follows.

(1) Transfer the "B1$_{16}$" command code with the 1st byte.

(2) After the "B1$_{16}$" check code is output with the 2nd byte, change the baud rate to 19,200 bps.

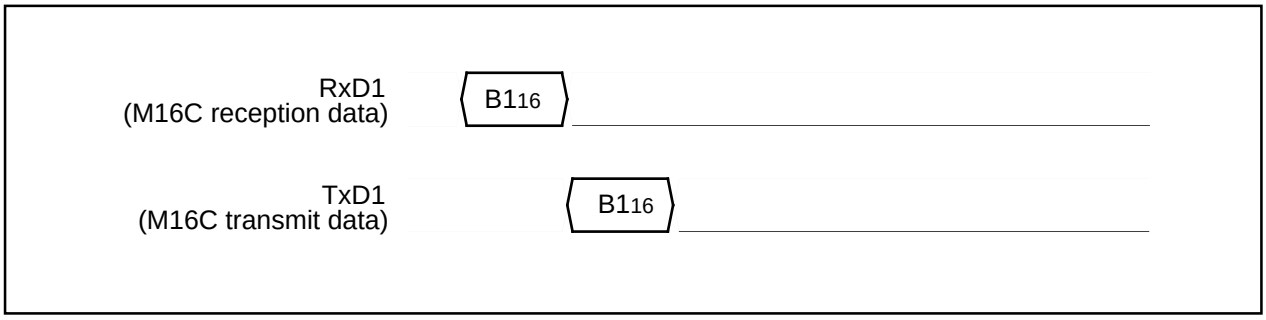



**Figure 1.32.39. Timing of baud rate 19200**

**Baud Rate 38400**

This command changes baud rate to 38,400 bps. Execute it as follows.

(1) Transfer the "B2$_{16}$" command code with the 1st byte.

(2) After the "B2$_{16}$" check code is output with the 2nd byte, change the baud rate to 38,400 bps.

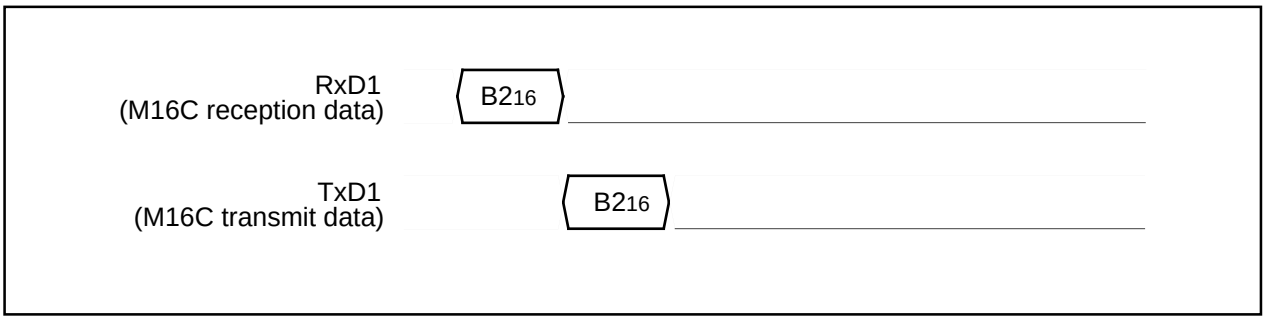



**Figure 1.32.40. Timing of baud rate 38400**

**Baud Rate 57600**

This command changes baud rate to 57,600 bps. Execute it as follows.

(1) Transfer the "B3$_{16}$" command code with the 1st byte.

(2) After the "B3$_{16}$" check code is output with the 2nd byte, change the baud rate to 57,600 bps.

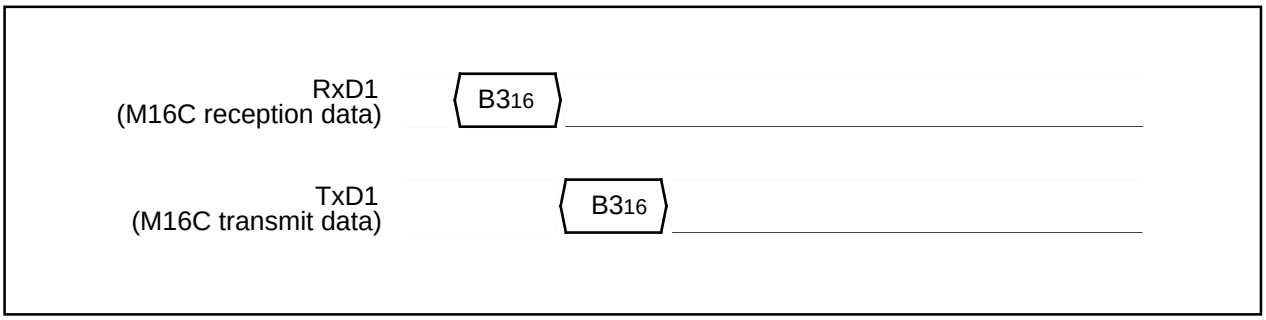
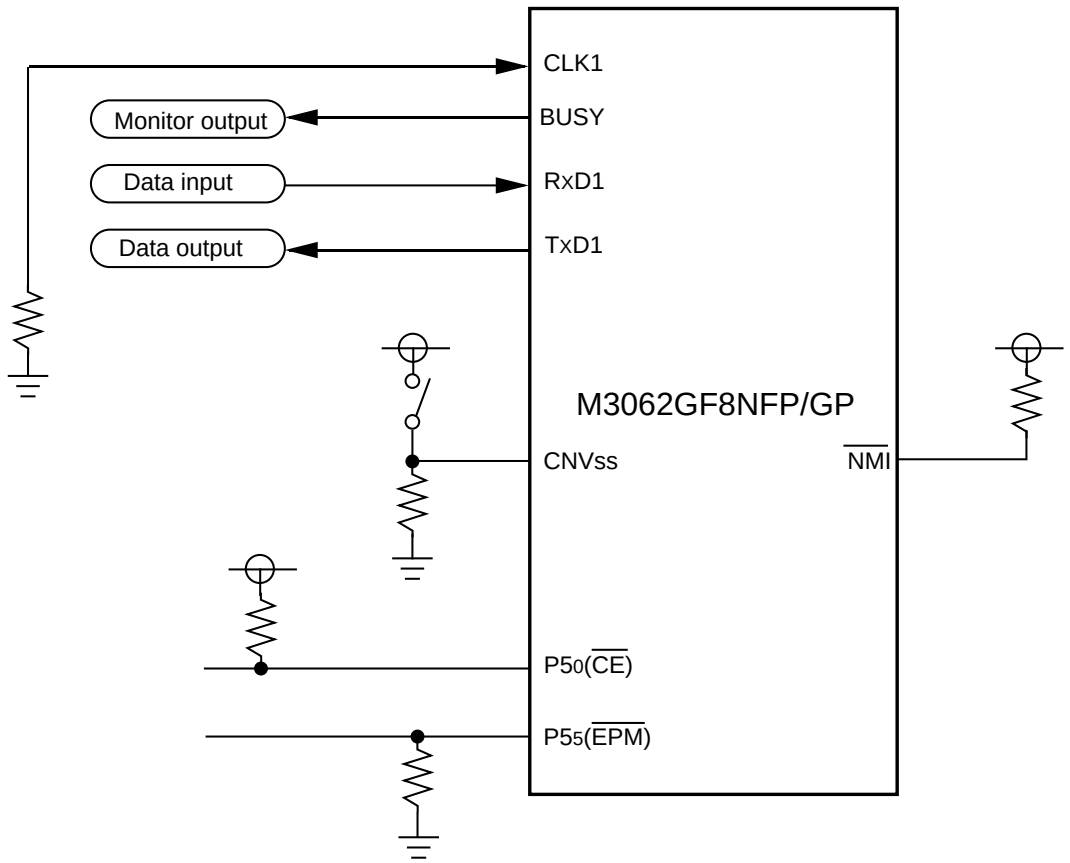



**Figure 1.32.41. Timing of baud rate 57600**

## 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.




(1) In this example, the microprocessor mode and standard serial I/O mode are switched via a switch.

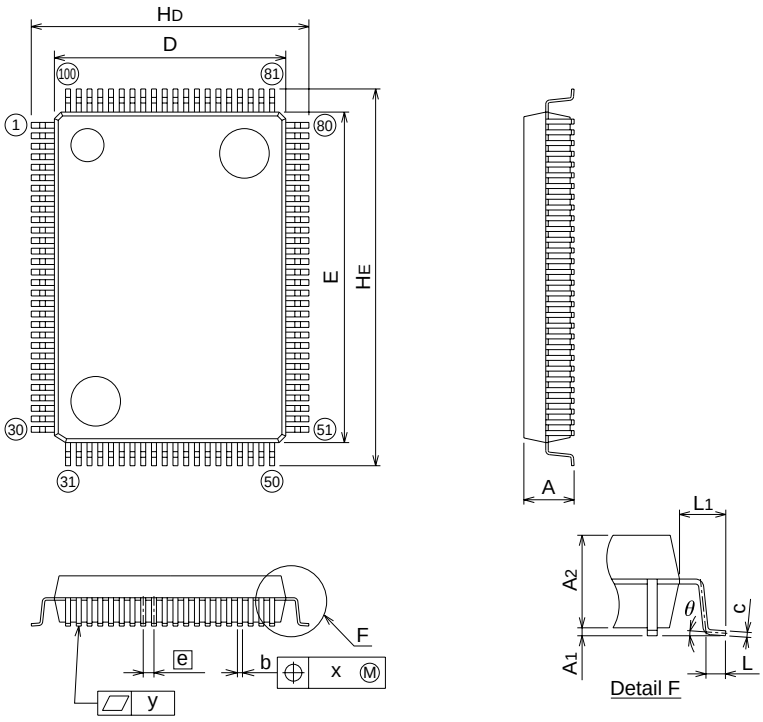
**Figure 1.32.42. Example circuit application for the standard serial I/O mode 2**

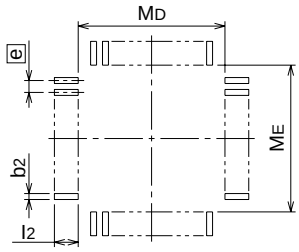
## Package Outline

### 100P6S-A (MMP)

**Plastic 100pin 14×20mm body QFP**

| EIAJ Package Code | JEDEC Code | Weight(g) | Lead Material |
|---|---|---|---|
| QFP100-P-1420-0.65 | – | 1.58 | Alloy 42 |



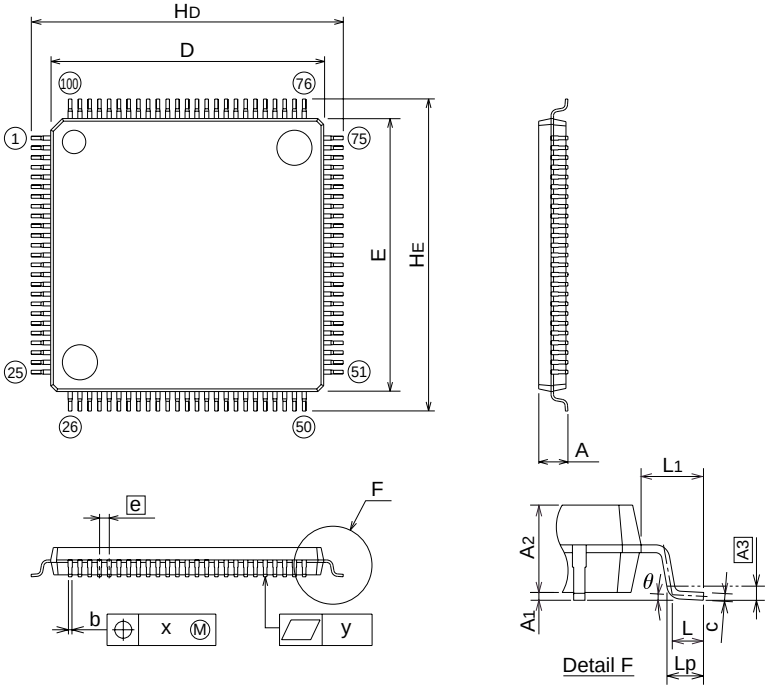

Recommended Mount Pad

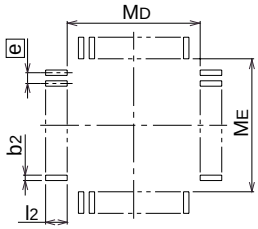
| Symbol | Dimension in Millimeters | | |
|---|---|---|---|
| | Min | Nom | Max |
| A | – | – | 3.05 |
| $A_1$ | 0 | 0.1 | 0.2 |
| $A_2$ | – | 2.8 | – |
| b | 0.25 | 0.3 | 0.4 |
| c | 0.13 | 0.15 | 0.2 |
| D | 13.8 | 14.0 | 14.2 |
| E | 19.8 | 20.0 | 20.2 |
| e | – | 0.65 | – |
| $H_D$ | 16.5 | 16.8 | 17.1 |
| $H_E$ | 22.5 | 22.8 | 23.1 |
| L | 0.4 | 0.6 | 0.8 |
| $L_1$ | – | 1.4 | – |
| x | – | – | 0.13 |
| y | – | – | 0.1 |
| θ | 0° | – | 10° |
| $b_2$ | – | 0.35 | – |
| $I_2$ | 1.3 | – | – |
| $M_D$ | – | 14.6 | – |
| $M_E$ | – | 20.6 | – |

### 100P6Q-A (MMP)

**Plastic 100pin 14×14mm body LQFP**

| EIAJ Package Code | JEDEC Code | Weight(g) | Lead Material |
|---|---|---|---|
| LQFP100-P-1414-0.50 | – | 0.63 | Cu Alloy |




Recommended Mount Pad

| Symbol | Dimension in Millimeters | | |
|---|---|---|---|
| | Min | Nom | Max |
| A | – | – | 1.7 |
| $A_1$ | 0 | 0.1 | 0.2 |
| $A_2$ | – | 1.4 | – |
| b | 0.13 | 0.18 | 0.28 |
| c | 0.105 | 0.125 | 0.175 |
| D | 13.9 | 14.0 | 14.1 |
| E | 13.9 | 14.0 | 14.1 |
| e | – | 0.5 | – |
| $H_D$ | 15.8 | 16.0 | 16.2 |
| $H_E$ | 15.8 | 16.0 | 16.2 |
| L | 0.3 | 0.5 | 0.7 |
| $L_1$ | – | 1.0 | – |
| Lp | 0.45 | 0.6 | 0.75 |
| A3 | – | 0.25 | – |
| x | – | – | 0.08 |
| y | – | – | 0.1 |
| θ | 0° | – | 10° |
| $b_2$ | – | 0.225 | – |
| $I_2$ | 0.9 | – | – |
| $M_D$ | – | 14.4 | – |
| $M_E$ | – | 14.4 | – |

## Differences between M3062GF8N and M16C/62M

### Differences between M3062GF8N and M16C/62M(Note)

| Item | M3062GF8N | M16C/62M |
|---|---|---|
| Shortest instruction execution time | 100ns (f(XIN)=10MHz, VCC=2.7V to 3.6V) | 100ns (f(XIN)=10MHz, VCC=2.7V to 3.6V)<br>142.9ns (f(XIN)=7MHz, VCC=2.2V to 3.6V with software one-wait) |
| Supply voltage | 2.7V to 3.6V (f(XIN)=10MHz, without software wait) | 2.7V to 3.6V (f(XIN)=10MHz, without software wait)<br>2.4V to 2.7V (f(XIN)=7MHz, without software wait)<br>2.2V to 2.4V (f(XIN)=7MHz with software one-wait) |
| Low power consumption | 30.0mW (VCC = 3V, f(XIN)=10MHz, without software wait) | 28.5mW (VCC = 3V, f(XIN)=10MHz, without software wait) |

Note: About the details and the electric characteristics, refer to data sheet.

### Differences in SFR between M3062GF8N and M16C/62M

| Address | Register name | M3062GF8N | M16C/62M |
|---|---|---|---|
| $0005_{16}$ | Processor mode register 1 (PM1) | b3 Reserved bit | b3 Internal reserved area expansion bit |
| $03D7_{16}$ | A-D control register 1 (ADCON1) | b0 Reserved bit | b0 Frequency select bit 1 |
| $03B4_{16}$ | Flash identification register (FIDR) | Have | Reserved register |
| $03B6_{16}$ | Flash memory control register 1 (FMR1) | Reserved register | Have |
| $03B7_{16}$ | Flash memory control register 0 (FMR0) | b7 Erase status flag<br>b6 Program status flag | b7 Nothing is assigned<br>b6 Nothing is assigned |

| REVISION HISTORY | M3062GF8NFP/GP DATA SHEET |
|---|---|

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| B | 12/04/01 | 8 | Note 4 is partly revised. |
| | | 9 | Standard values of RPULLUP in Table 1.26.3 are revised. |
| | | 10 | Table 1.26.6 is partly revised. |
| | | 23 | Note 2 is added in Table 1.28.1. |
| | | 24 | Note 2 in Figure 1.28.1 is partly revised. |
| | | 25 | Explanation of “Microcomputer Mode and Boot Mode” is partly added. |
| | | 25 | Explanation of “Block Address” is partly revised. |
| | | 26 | Explanation of “Outline Performance (CPU Rewrite Mode)” is partly added. |
| | | 30 | Table 1.29.1 is partly revised. |
| | | 30 | Explanation of “Read Status Register Command” is partly added. |
| | | 31 | Figure 1.29.3 is partly revised. |
| | | 32 | Figure 1.29.4 is partly revised. |
| | | 33 | Figure 1.29.5 is partly revised. |
| | | 35 | Explanation of “Status Register” is partly revised. |
| | | 37 | Figure 1.29.7 is partly revised. |
| 1.0 | 06/02/02 | 10-169 | Explanation is added. |
| | | 188 | Explanation of “Outline Performance (CPU Rewrite Mode)” is partly added. |
| | | 192 | Table 1.29.1 is partly revised. |
| | | 192 | Explanation of “Read Array Command” is partly revised. |
| | | 192 | Explanation of “Read Status Register Command” is partly revised. |
| | | 192 | Explanation of “Program Command” is partly revised. |
| | | 193-195 | Figure 1.29.3 to Figure 1.29.5 are partly revised. |
| | | 195 | Explanation of “Lock Bit Program Command” is partly revised. |
| | | 237 | “Package outline” is added. |
| | | 238 | Explanation of “Differences between M3062GF8N and M16C/62M” is added. |
| 1.1 | 18/02/02 | 173 | Table 1.26.7. Flash memory version program voltage and read operation voltage characteristics is added. |

Renesas Technology Corp.
Nippon Bldg.,6-2,Otemachi 2-chome,Chiyoda-ku,Tokyo,100-0004 Japan

**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.

© 2002 MITSUBISHI ELECTRIC CORP.
Printed in Japan (RODI) II
New publication, effective Feb. 2002.
Specifications subject to change without notice.

RENESAS
Renesas Technology Corp.