

16

# M30245 Group

User's Manual

RENESAS 16-BIT SINGLE-CHIP MICROCOMPUTER  
M16C FAMILY / M16C/20 SERIES

Users Manual

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Technology Corp. without notice. Please review the latest information published by Renesas Technology Corp. through various means, including the Renesas Technology Corp. website (<http://www.renesas.com>).

Rev. 2.00  
Revision date: Oct 16, 2006

Renesas Technology  
[www.renesas.com](http://www.renesas.com)

## Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (<http://www.renesas.com>)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
  - (1) artificial life support devices or systems
  - (2) surgical implantations
  - (3) healthcare intervention (e.g., excision, administration of medication, etc.)
  - (4) any other purposes that pose a direct threat to human lifeRenesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.

## General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

### 1. Handling of Unused Pins

Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.

- The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

### 2. Processing at Power-on

The state of the product is undefined at the moment when power is supplied.

- The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.

In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.

In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

### 3. Prohibition of Access to Reserved Addresses

Access to reserved addresses is prohibited.

- The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

### 4. Clock Signals

After applying a reset, only release the reset line after the operating clock signal has become stable.

When switching the clock signal during program execution, wait until the target clock signal has stabilized.

- When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

### 5. Differences between Products

Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.

- The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

# How to Use This Manual

This user's manual is written for the M30245 group.

The reader of this manual is expected to have the basic knowledge of electric and logic circuits and microcomputers.

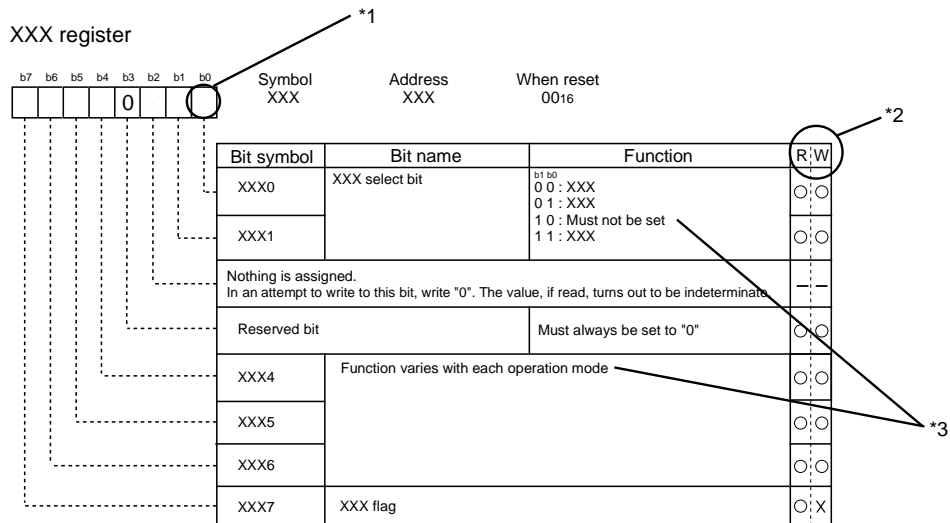
This manual explains a function of the following kind.

- M30245M8-XXXGP
- M30245MC-XXXGP
- M30245FCGP

These products have similar features except for the memories, which differ from one product to another. Be careful when writing a program, as the memories have different capacities.

RAM Size (Byte)	64K	128K	ROM Size (Byte)
10K		Flash memory version:M30245FCGP Mask ROM version:M30245MC-XXXGP	
5K	Mask ROM version:M30245M8-XXXGP		

The figure of each register configuration describes its functions and attributes as follows :



\*1

Blank: Set to "0" or "1" according to intended use  
 0: Set to "0"  
 1: Set to "1"  
 X: Nothing is assigned

\*2

R: Read  
 O.....Possible to read  
 X.....Impossible to read  
 -.....Nothing is assigned  
 W: Write  
 O.....Possible to write  
 X.....Written value is invalid  
     When write, value can be "0" or "1"  
 -.....Nothing is assigned

\*3

Terms to use here are explained as follows.

- Nothing is assigned  
Nothing is assigned to the bit concerned. When write, set "0" for new function in future plan.
- Must not be set  
Not select. The operation at having selected is not guaranteed.
- Reserved bit  
Reserved bit. Set the specified value.
- Function varies with each operation mode  
Bit function changes according to the mode of peripheral functions.
- Always set to "0" in A mode.  
Set the corresponding bit to "0" in A mode.
- Invalid in A mode  
The bit concerned has no function in A mode. Set the specified value.
- Valid when bit A="0"  
When bit A is "1", the bit concerned has no function. When bit A is "0", the bit concerned has function.

# Table of Contents

<b>Chapter 1. Hardware .....</b>	<b>1</b>
<b>Chapter 2. Peripheral Functions Usage .....</b>	<b>3</b>
<b>2.1 Protect .....</b>	<b>4</b>
2.1.1 Overview .....	4
2.1.2 Protect Operation .....	5
<b>2.2 Timer A .....</b>	<b>6</b>
2.2.1 Overview .....	6
2.2.2 Operation of Timer A (timer mode) .....	12
2.2.3 Operation of Timer A (timer mode, gate function selected) .....	14
2.2.4 Operation of Timer A (timer mode, pulse output function selected) .....	16
2.2.5 Operation of Timer A (event counter mode, reload type selected) .....	18
2.2.6 Operation of Timer A (event counter mode, free run type selected) .....	20
2.2.7 Operation of timer A (two-phase pulse signal process in event counter mode, normal mode selected) .....	22
2.2.8 Operation of timer A (two-phase pulse signal process in event counter mode, multiply-by-4 mode selected) .....	24
2.2.9 Operation of Timer A (one-shot timer mode) .....	26
2.2.10 Operation of Timer A (pulse width modulation mode, 16-bit PWM mode selected) .....	28
2.2.11 Operation of Timer A (pulse width modulation mode, 8-bit PWM mode selected) .....	31
2.2.12 Precautions for Timer A (timer mode) .....	34
2.2.13 Precautions for Timer A (event counter mode) .....	35
2.2.14 Precautions for Timer A (one-shot timer mode) .....	37
2.2.15 Precautions for Timer A (pulse width modulation mode) .....	38
<b>2.3 Clock-Synchronous Serial I/O .....</b>	<b>39</b>
2.3.1 Overview .....	39
2.3.2 Operation of Serial I/O (transmission in clock-synchronous serial I/O mode) .....	45
2.3.3 Operation of Serial I/O (reception in clock-synchronous serial I/O mode) .....	49
2.3.4 Precautions for Serial I/O (in clock-synchronous serial I/O mode) .....	53
<b>2.4 Clock-Asynchronous Serial I/O (UART) .....</b>	<b>55</b>
2.4.1 Overview .....	55
2.4.2 Operation of Serial I/O (transmission in UART mode) .....	64
2.4.3 Operation of Serial I/O (reception in UART mode) .....	68
2.4.4 Serial I/O Precautions (UART Mode) .....	72
2.4.5 Operation of Serial I/O (transmission used for SIM interface) .....	73
2.4.6 Operation of Serial I/O (reception used for SIM interface) .....	77
2.4.7 Clock Signals in used for the SIM Interface .....	81
<b>2.5 Serial Interface Special Function .....</b>	<b>85</b>
2.5.1 Overview .....	85
2.5.2 Operation of Serial Interface Special Function (transmission in master mode without delay) .....	94

2.5.3 Operation of Serial Interface Special Function (reception in master mode with clock delay) .....	98
2.5.4 Operation of Serial Interface Special Function (transmission in slave mode without delay) .....	102
2.5.5 Operation of Serial Interface Special Function (reception in slave mode with clock delay) .....	106
<b>2.6 Serial sound interface .....</b>	<b>110</b>
2.6.1 Overview .....	110
2.6.2 Example of Serial Sound Interface operation .....	116
2.6.3 Precautions for Serial Sound Interface .....	120
<b>2.7 Frequency synthesizer (PLL) .....</b>	<b>121</b>
2.7.1 Overview .....	121
2.7.2 Operation of frequency synthesizer .....	124
2.7.3 Precautions for Frequency synthesizer .....	126
<b>2.8 USB function .....</b>	<b>127</b>
2.8.1 Overview .....	127
2.8.2 USB function control .....	139
2.8.3 USB Interrupt .....	150
2.8.4 USB Operation (Suspend/Resume Function) .....	161
2.8.5 USB Operation (Endpoint 0) .....	169
2.8.6 USB Operation (Endpoints 1 to 4 Receive) .....	182
2.8.7 USB Operation (Endpoints 1 to 4 Transmit) .....	193
2.8.8 USB Operation (Interface with DMAC Transfer) .....	207
2.8.9 Precautions for USB .....	210
<b>2.9 A/D Converter .....</b>	<b>213</b>
2.9.1 Overview .....	213
2.9.2 Operation of A/D converter (one-shot mode) .....	218
2.9.3 Operation of A/D Converter (in one-shot mode, an external trigger selected) .....	220
2.9.4 Operation of A/D Converter (in repeat mode) .....	222
2.9.5 Operation of A/D Converter (in single sweep mode) .....	224
2.9.6 Operation of A/D Converter (in repeat sweep mode 0) .....	226
2.9.7 Operation of A/D Converter (in repeat sweep mode 1) .....	228
2.9.8 Precautions for A/D Converter .....	230
2.9.9 Method of A/D Conversion (10-bit mode) .....	231
2.9.10 Method of A/D Conversion (8-bit mode) .....	233
2.9.11 Absolute Accuracy and Differential Non-Linearity Error .....	235
2.9.12 Internal Equivalent Circuit of Analog Input .....	237
2.9.13 Sensor's Output Impedance under A/D Conversion (reference value) .....	238
<b>2.10 DMAC Usage .....</b>	<b>240</b>
2.10.1 Overview of the DMAC usage .....	240
2.10.2 Operation of DMAC (one-shot transfer mode) .....	245
2.10.3 Operation of DMAC (repeated transfer mode) .....	247
<b>2.11 CRC Calculation Circuit .....</b>	<b>249</b>
2.11.1 Overview .....	249
2.11.2 Operation of CRC Calculation Circuit .....	251
2.11.3 SFR Access Snoop Function .....	252
<b>2.12 Watchdog Timer .....</b>	<b>253</b>

2.12.1 Overview .....	253
2.12.2 Operation of Watchdog Timer (Watchdog timer interrupt) .....	256
<b>2.13 Address Match Interrupt Usage .....</b>	<b>258</b>
2.13.1 Overview of the address match interrupt usage .....	258
2.13.2 Operation of Address Match Interrupt .....	260
<b>2.14 Key-Input Interrupt Usage .....</b>	<b>262</b>
2.14.1 Overview of the key-input interrupt usage .....	262
2.14.2 Operation of Key-Input Interrupt .....	265
<b>2.15 Multiple interrupts Usage .....</b>	<b>267</b>
2.15.1 Overview of the Multiple interrupts usage .....	267
2.15.2 Multiple Interrupts Operation .....	272
<b>2.16 Power Control Usage .....</b>	<b>274</b>
2.16.1 Overview of the power control usage .....	274
2.16.2 Stop Mode Set-Up .....	280
2.16.3 Wait Mode Set-Up .....	281
2.16.4 Precautions in Power Control .....	282
<b>2.17 Programmable I/O Ports Usage .....</b>	<b>284</b>
2.17.1 Overview of the programmable I/O ports usage .....	284

## Chapter 3. Examples of Peripheral Functions

<b>Applications .....</b>	<b>293</b>
3.1 Long-Period Timers .....	295
3.2 Variable-Period Variable-Duty PWM Output .....	299
3.3 Buzzer Output .....	303
3.4 Solution for External Interrupt Pins Shortage .....	305
3.5 Memory to Memory DMA Transfer .....	307
3.7 Buzzer Output .....	311
3.6 CRC Calculation SFR Access Snoop Function in Clock Synchronous Serial Data Transmit .....	311
3.7 Transfer from USB FIFO to Serial Sound Interface .....	316
3.8 Controlling Power Using Stop Mode .....	321
3.9 Controlling Power Using Wait Mode .....	325

## Chapter 4. External Buses .....

<b>4.1 Overview of External Buses .....</b>	<b>330</b>
<b>4.2 Data Access .....</b>	<b>331</b>
4.2.1 Data Bus Width .....	331
4.2.2 Chip Selects and Address Bus .....	332
4.2.3 R/W Modes .....	333
<b>4.3 Connection Examples .....</b>	<b>334</b>
4.3.1 16-bit Memory to 16-bit Width Data Bus Connection Example .....	334
4.3.2 8-bit Memory to 16-bit Width Data Bus Connection Example .....	335



4.3.3 8-bit Memory to 8-bit Width Data Bus Connection Example .....	337
4.3.4 Two 8-bit and 16-Bit Memory to 16-Bit Width Data Bus Connection Example .....	338
4.3.5 Chip Selects and Address Bus .....	339
<b>4.4 Connectable Memories .....</b>	<b>340</b>
4.4.1 Operation Frequency and Access Time .....	340
4.4.2 Connecting Low-Speed Memory .....	343
4.4.3 Connectable Memories .....	346
<b>4.5 Releasing an External Bus (<math>\overline{\text{HOLD}}</math> input and <math>\overline{\text{HLDA}}</math> output).....</b>	<b>347</b>
<b>4.6 Precautions for External Bus .....</b>	<b>349</b>
<b>Chapter 5. Standard Characteristics .....</b>	<b>351</b>
<b>5.1 DC Standard Characteristics .....</b>	<b>352</b>
5.1.1 Port Standard Characteristics .....	352
5.1.2 VCC-ICC Characteristics .....	354

# Chapter 1

---

Hardware

See M30245 group datasheet.

# Chapter 2

---

## Peripheral Functions Usage

## 2.1 Protect

### 2.1.1 Overview

'Protect' is a function that causes a value held in a register to be unchanged even when a program runs away. The following is an overview of the protect function:

#### (1) Registers affected by the protect function

The registers affected by the protect function are:

- (a) System clock control registers 0, 1 (addresses 0006<sub>16</sub> and 0007<sub>16</sub>)
- (b) Processor mode registers 0, 1 (addresses 0004<sub>16</sub> and 0005<sub>16</sub>)
- (c) Frequency synthesizer-related registers (address 03DB<sub>16</sub> to 03DF<sub>16</sub>)

The values in registers (a) through (c) cannot be changed in write-protect state. To change values in the registers, put the individual registers in write-enabled state.

#### (2) Protect register

Figure 2.1.1 shows protect register.

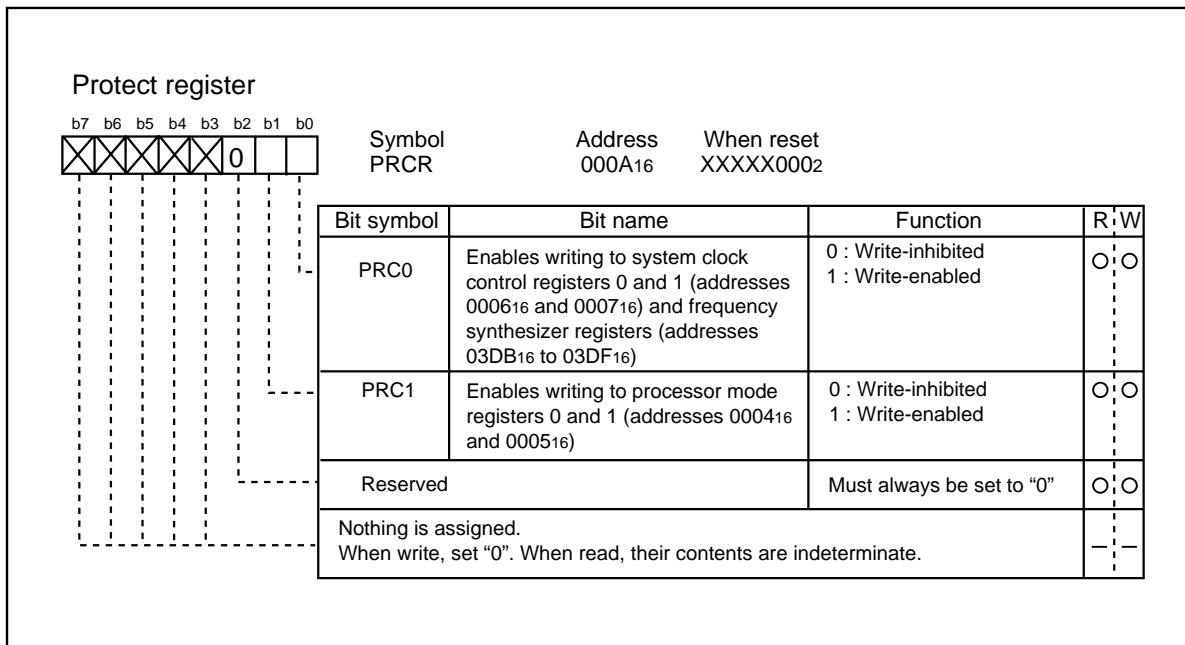


Figure 2.1.1. Protect register

### 2.1.2 Protect Operation

The following explains the protect operation. Figure 2.1.2 shows the set-up procedure.

- Operation
- (1) Setting "1" in the write-enable bit of system clock control registers 0 and 1 and frequency synthesizer-related registers causes system clock control register 0 and 1 and frequency synthesizer-related registers to be in write-enabled state.
  - (2) The contents of system clock control register 0 and 1 and these of frequency synthesizer-related registers are changed.
  - (3) Setting "0" in PRC0 causes system clock control register 0 and 1 and frequency synthesizer-related registers to be in write-inhibited state.
  - (4) To change the contents of processor mode register 0 and that of processor mode register 1, follow the same steps as in dealing with system clock control registers and frequency synthesizer-related registers.

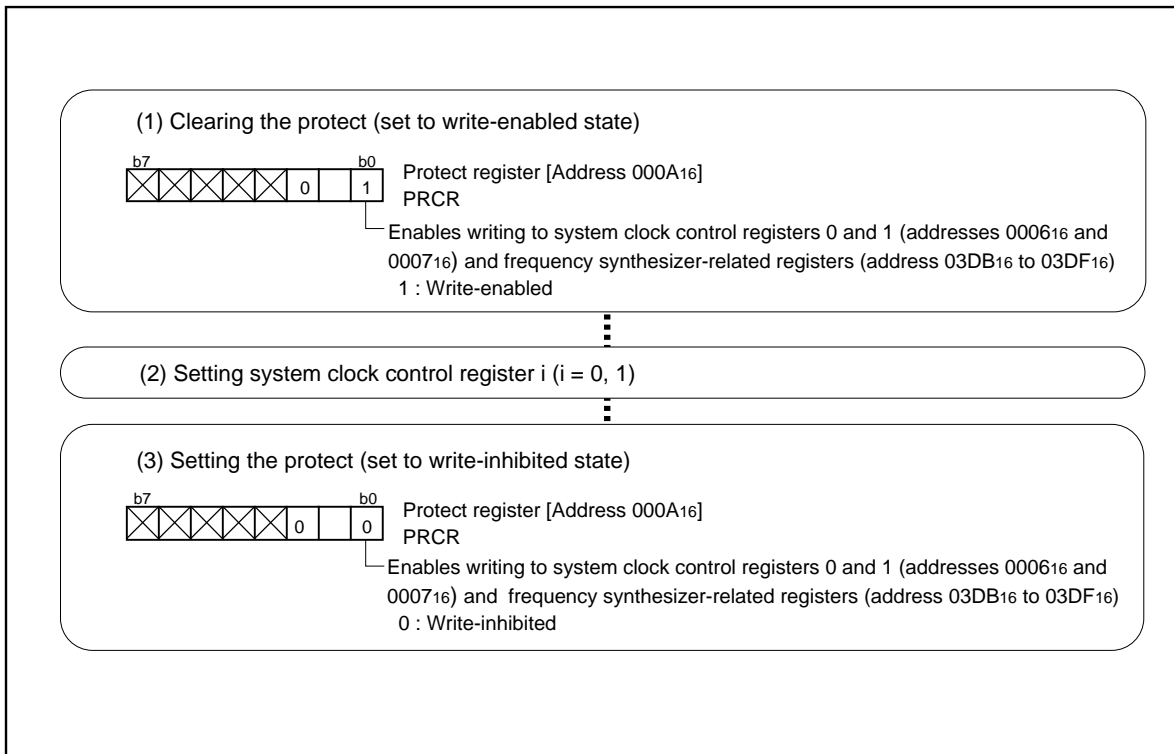


Figure 2.1.2. Set-up procedure for protect function

## 2.2 Timer A

### 2.2.1 Overview

The following is an overview for timer A, a 16-bit timer.

#### (1) Mode

Timer A operates in one of the four modes:

##### (a) Timer mode

In this mode, the internal count source is counted. Two functions can be selected: the pulse output function that reverses output from a port every time an overflow occurs, or the gate function which controls the count start/stop according to the input signal from a port.

##### (b) Event counter mode

This mode counts the pulses from the outside and the number of overflows in other timers. The free-run type, in which nothing is reloaded from the reload register, can be selected when an underflow occurs. The pulse output function can also be selected. Please refer to the timer mode explanation for details, as the operation is identical.

Furthermore, Timer A has a two-phase pulse signal processing function which generates an up count or down count in the event counter mode, depending on the phase of the two input signals. The normal mode or 4-multiplication mode can be selected depending on the phase detective method.

##### (c) One-shot timer mode

In this mode, the timer is started by the trigger and stops when the timer goes to "0". The trigger can be selected from the following 2 types: an overflow of the timer, or a software trigger. The pulse output function can also be selected. Please refer to the timer mode explanation for details, as the operation is identical.

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

In this mode, the arbitrary pulses are successively output. Either a 16-bit fixed-period PWM mode or 8-bit variable-period mode can be selected. The trigger for initiating output can also be selected. Please refer to the one-shot timer mode explanation for details, as the operation is identical.

#### (2) Count source

The internal count source can be selected from f1, f8, f32, and fc32. Clocks f1, f8, and f32 are derived by dividing the CPU's main clock by 1, 8, and 32 respectively. Clock fc32 is derived by dividing the CPU's secondary clock by 32.

**(3) Frequency division ratio**

In timer mode or pulse width modulation mode, [the value set in the timer register + 1] becomes the frequency division ratio. In event counter mode, [the set value + 1] becomes the frequency division ratio when a down count is performed, or [FFFF<sub>16</sub> - the set value + 1] becomes the frequency division ratio when an up count is performed. In one-shot timer mode, the value set in the timer register becomes the frequency division ratio.

The counter overflows (or underflows) when a count source equal to a frequency division ratio is input, and an interrupt occurs. For the pulse output function, the output from the port varies (the value in the port register does not vary).

**(4) Reading the timer**

Either in timer mode or in event counter mode, reading the timer register takes out the count at that moment. Read it in 16-bit units. The data either in one-shot timer mode or in pulse width modulation mode is indeterminate.

**(5) Writing to the timer**

To write to the timer register when a count is in progress, the value is written only to the reload register. When writing to the timer register when a count is stopped, the value is written both to the reload register and to the counter. Write a value in 16-bit units.

**(6) Relation between the input/output to/from the timer and the direction register**

With the output function of the timer, pulses are output regardless of the contents of the port direction register. To input an external signal to the timer, set the port direction register to input.

**(7) Pins related to timer A**

- |  |  |
|--|--|
| (a) TA0IN, TA1IN, TA2IN, TA3IN, TA4IN      | Input pins to timer A.   |
| (b) TA0OUT, TA1OUT, TA2OUT, TA3OUT, TA4OUT | Output pins from timer A. They become input pins to timer A when event counter mode is active. |

**(8) Registers related to timer A**

Figure 2.2.1 shows the memory map of timer A-related registers. Figures 2.2.2 through 2.2.5 show timer A-related registers.



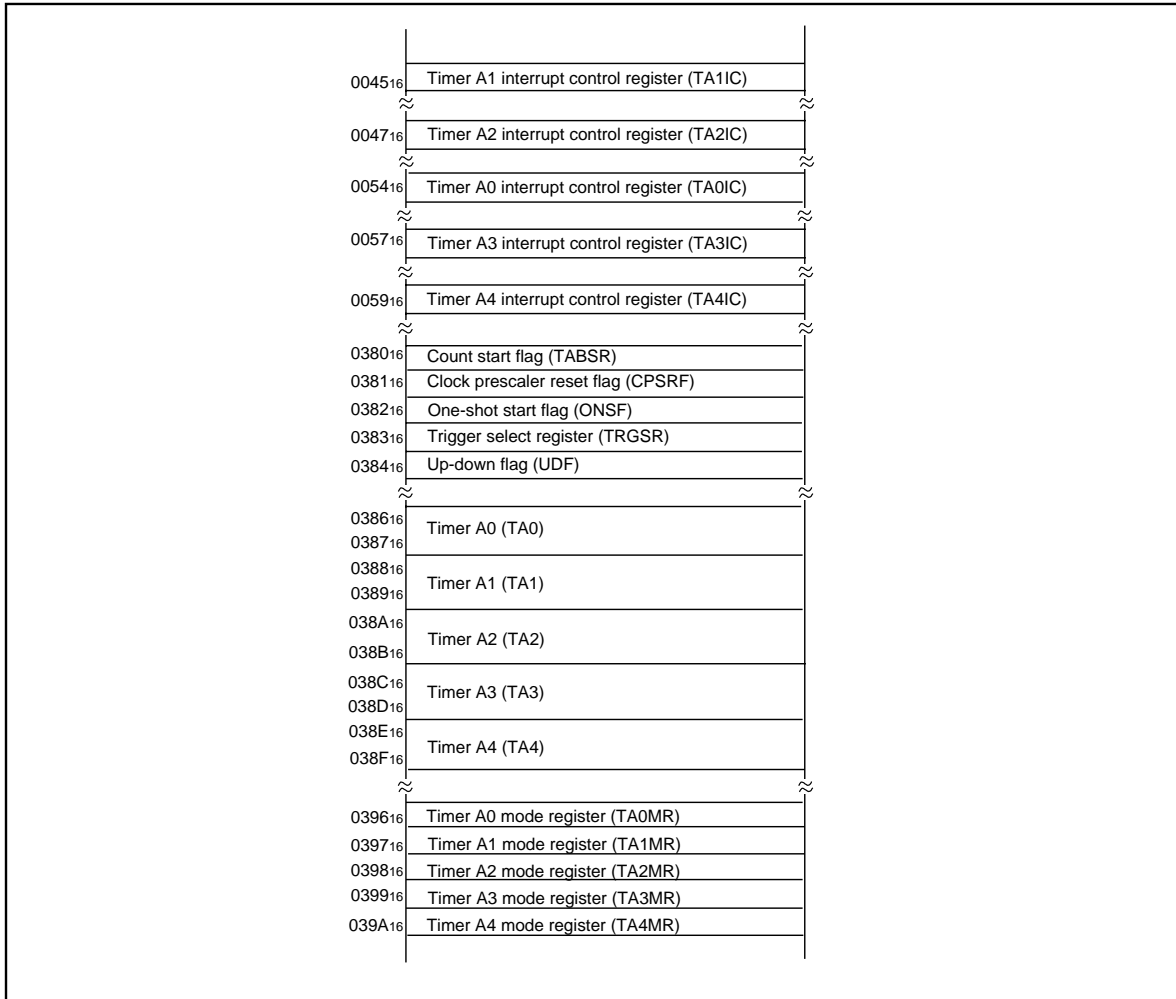


Figure 2.2.1. Memory map of timer A-related registers

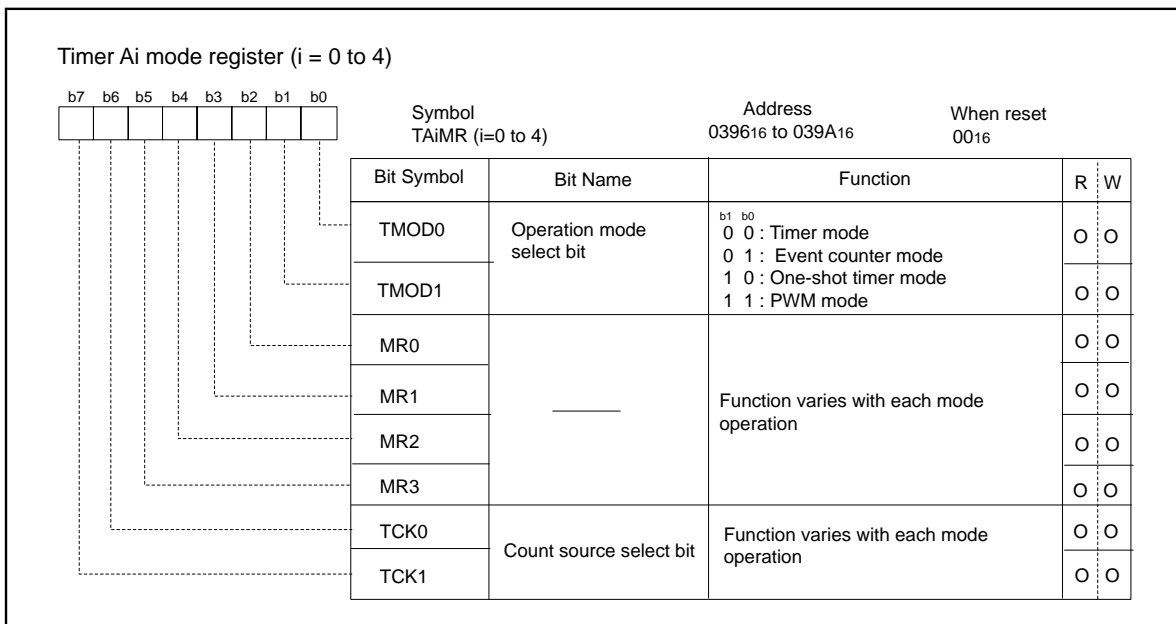


Figure 2.2.2. Timer A-related registers (1)

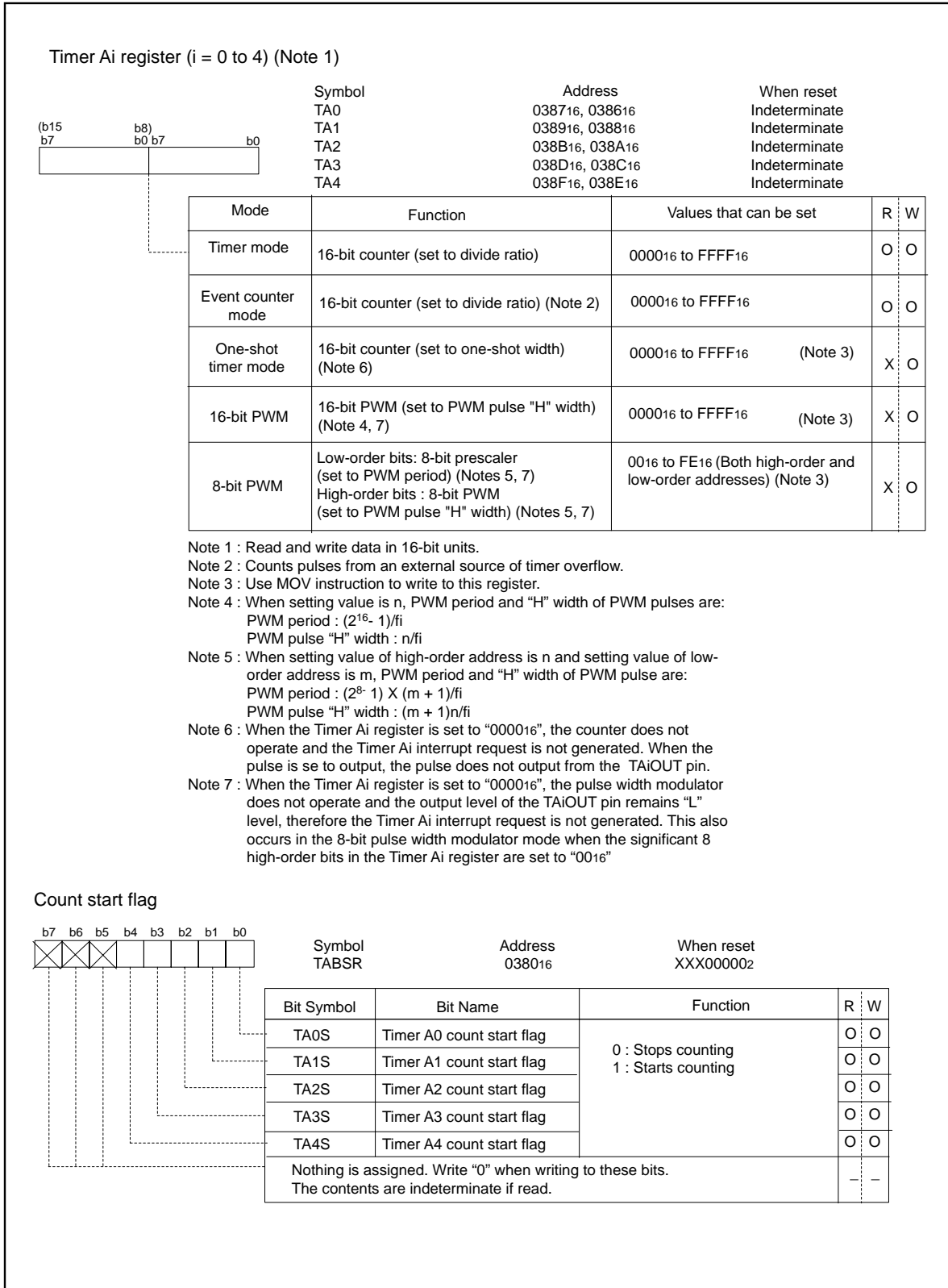


Figure 2.2.3. Timer A-related registers (2)

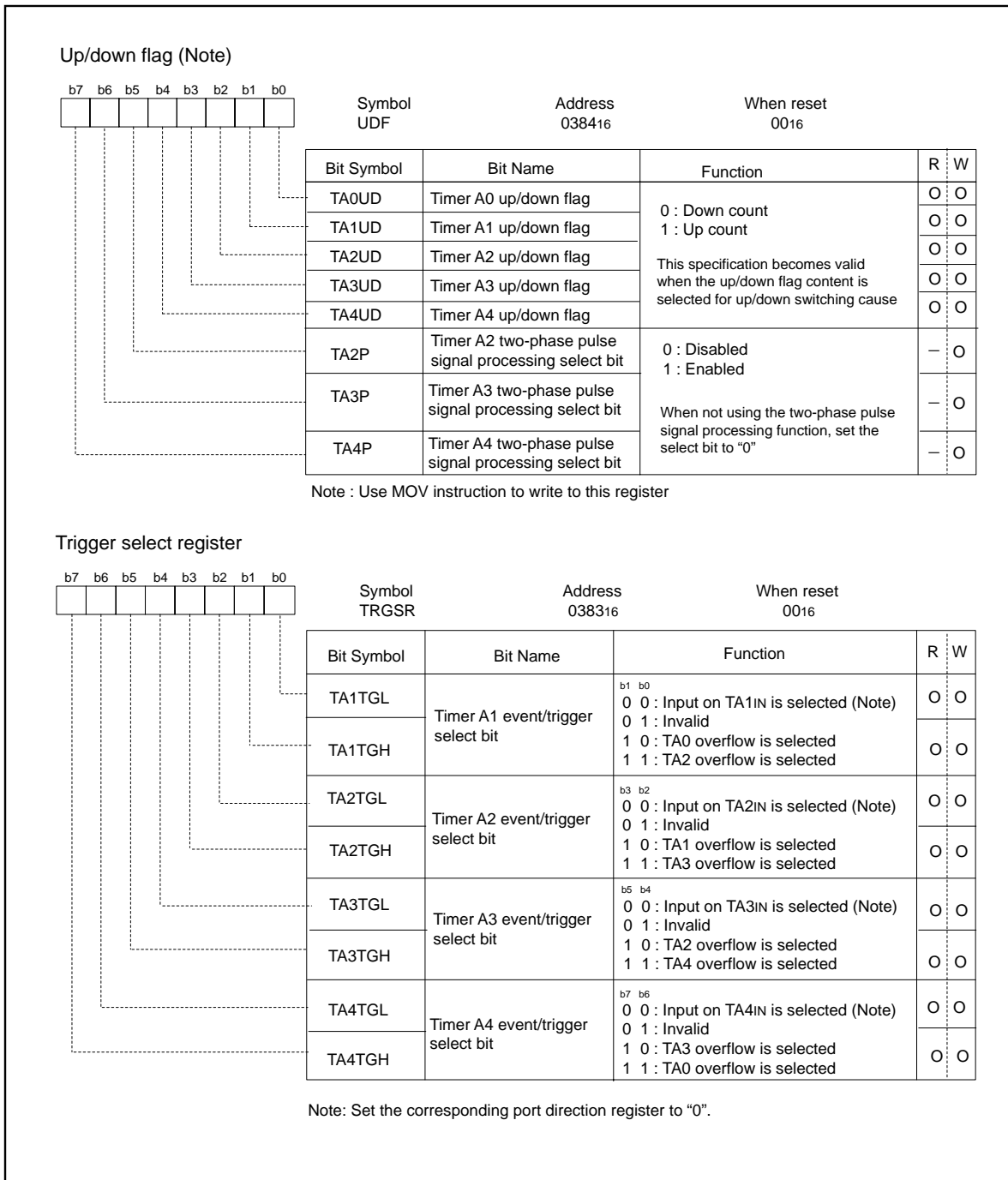


Figure 2.2.4. Timer A-related registers (3)

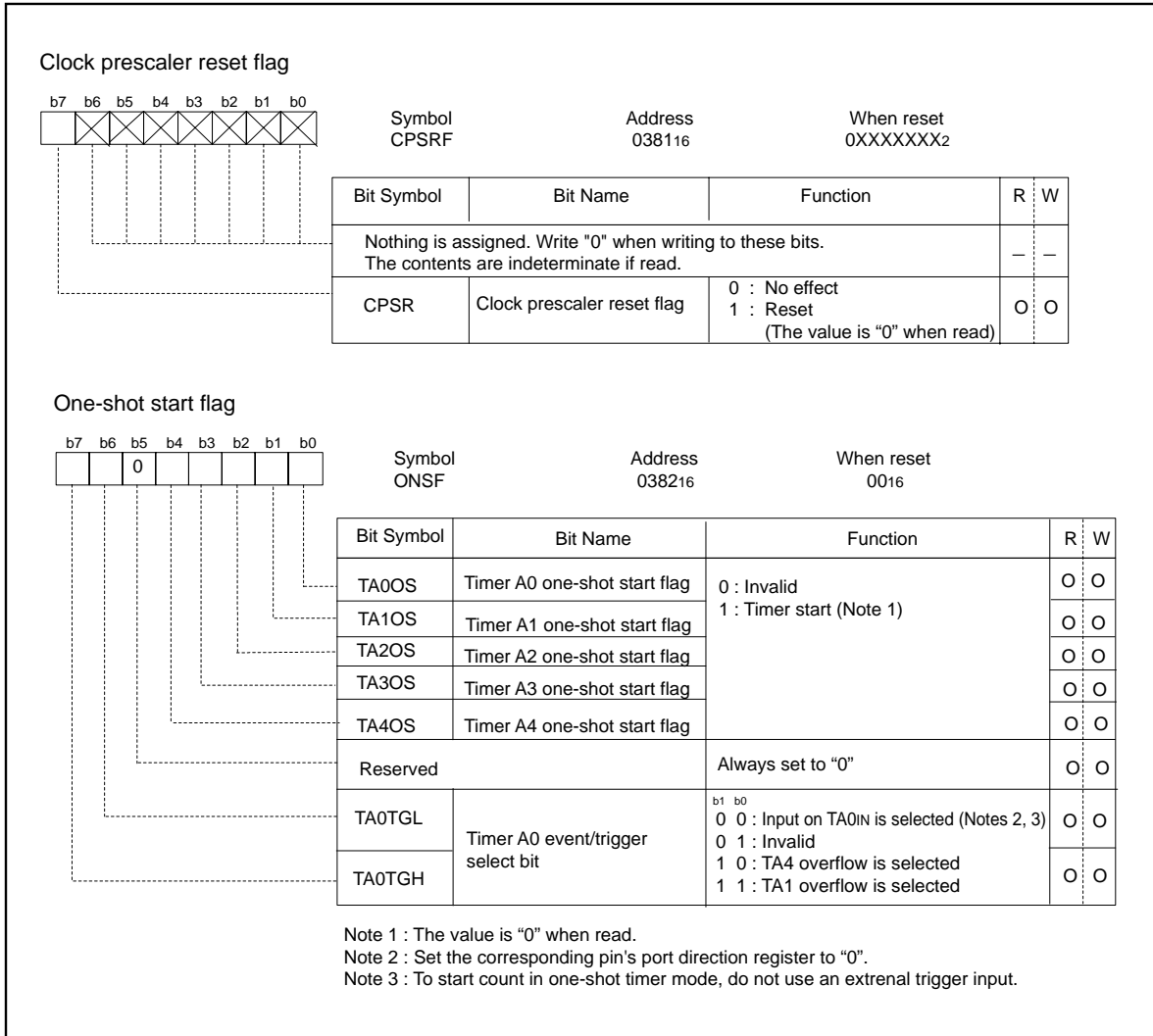


Figure 2.2.5. Timer A-related registers (4)

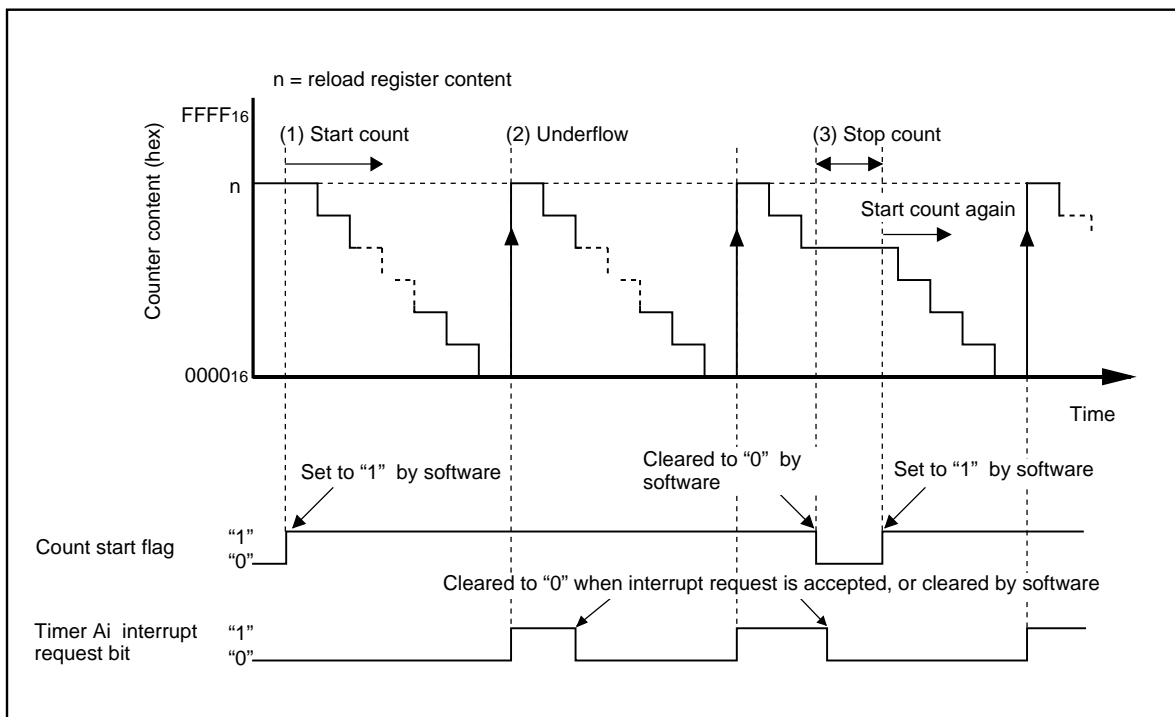
## 2.2.2 Operation of Timer A (timer mode)

In timer mode, choose functions from those listed in Table 2.2.1. Operations of the circled items are described below. Figure 2.2.6 shows the operation timing, and Figure 2.2.7 shows the set-up procedure.

**Table 2.2.1. Chosed functions**

Item	Set-up
Count source	○ Internal count source ( $f_1 / f_8 / f_{32} / f_{c32}$ )
Pulse output function	○ No pulses output
	Pulses output
Gate function	○ No gate function
	Performs count only for the period in which the TAIN pin is at "L" level
	Performs count only for the period in which the TAIN pin is at "H" level

- Operation
- (1) Setting the count start flag to "1" causes the counter to perform a down count on the count source.
  - (2) If an underflow occurs, the content of the reload register is reloaded, and the count continues. At this time, the timer Ai interrupt request bit goes to "1".
  - (3) Setting the count start flag to "0" causes the counter to hold its value and to stop.



**Figure 2.2.6. Operation timing of timer mode**

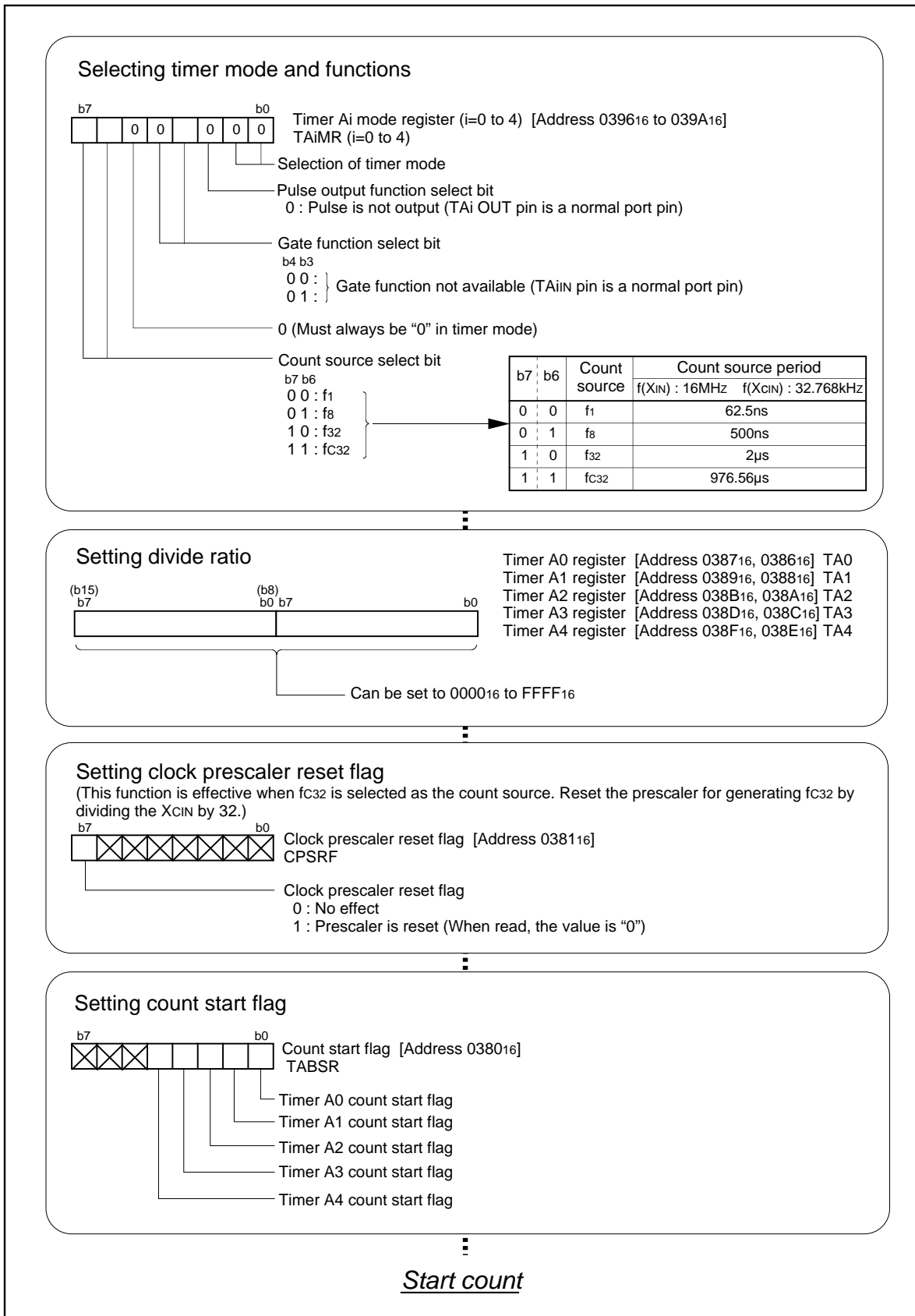


Figure 2.2.7. Set-up procedure of timer mode

### 2.2.3 Operation of Timer A (timer mode, gate function selected)

In timer mode, choose functions from those listed in Table 2.2.2. Operations of the circled items are described below. Figure 2.2.8 shows the operation timing, and Figure 2.2.9 shows the set-up procedure.

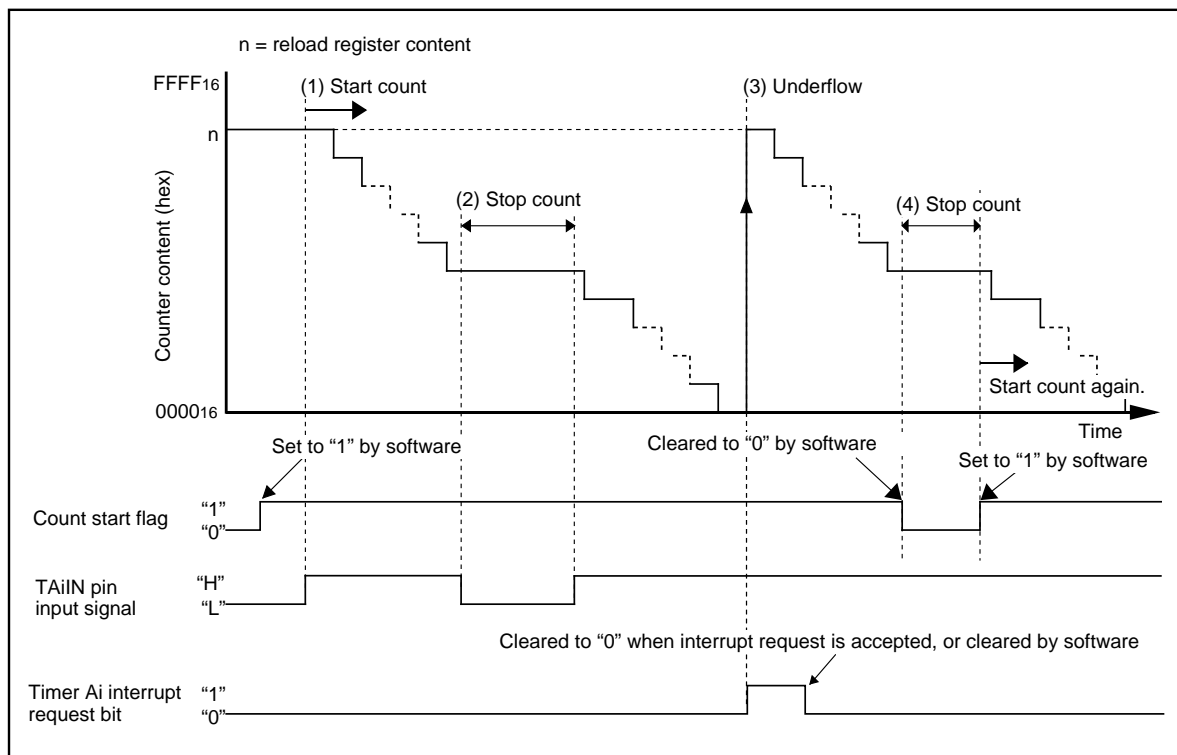
**Table 2.2.2. Chosen functions**

Item	Set-up
Count source	○ Internal count source( $f_1 / f_8 / f_{32} / f_{c32}$ )
Pulse output function	○ No pulses output
	Pulses output
Gate function	No gate function
	Performs count only for the period in which the TAIIN pin is at "L" level
	○ Performs count only for the period in which the TAIIN pin is at "H" level

- Operation
- (1) When the count start flag is set to "1" and the TAIIN pin inputs at "H" level, the counter performs a down count on the count source.
  - (2) When the TAIIN pin inputs at "L" level, the counter holds its value and stops.
  - (3) If an underflow occurs, the content of the reload register is reloaded and the count continues. At this time, the timer Ai interrupt request bit goes to "1".
  - (4) Setting the count start flag to "0" causes the counter to hold its value and to stop.

Note

- Make the pulse width of the signal input to the TAIIN pin not less than two cycles of the count source.



**Figure 2.2.8. Operation timing of timer mode, gate function selected**

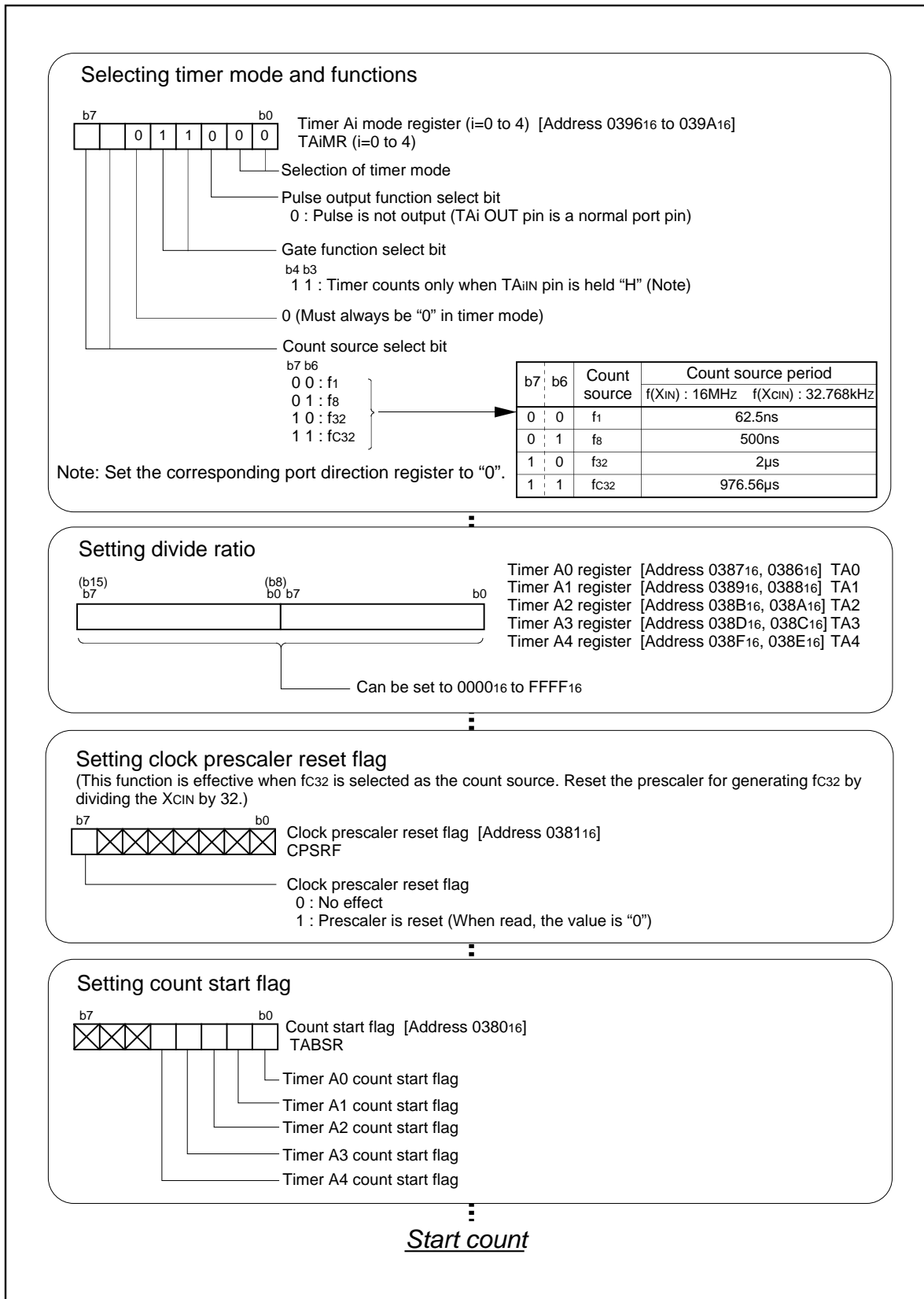


Figure 2.2.9. Set-up procedure of timer mode, gate function selected



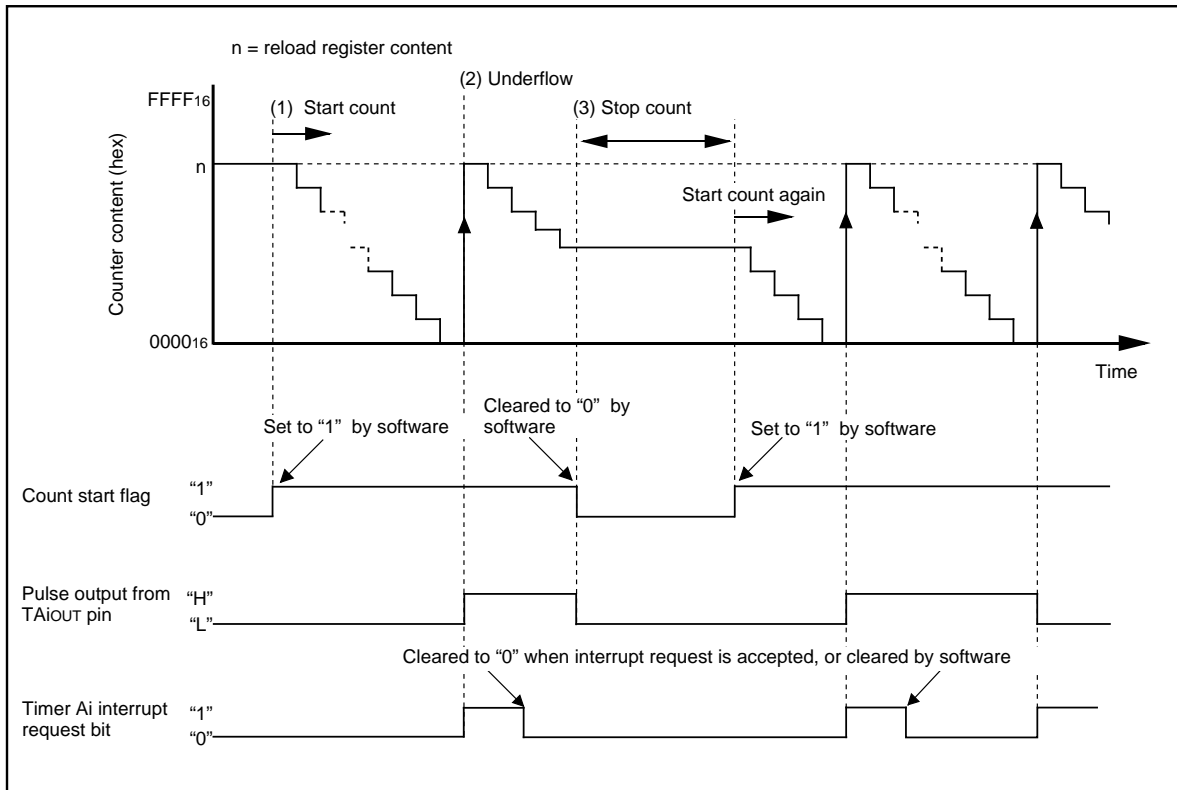
### 2.2.4 Operation of Timer A (timer mode, pulse output function selected)

In timer mode, choose functions from those listed in Table 2.2.3. Operations of the circled items are described below. Figure 2.2.10 shows the operation timing, and Figures 2.2.11 shows the set-up procedure.

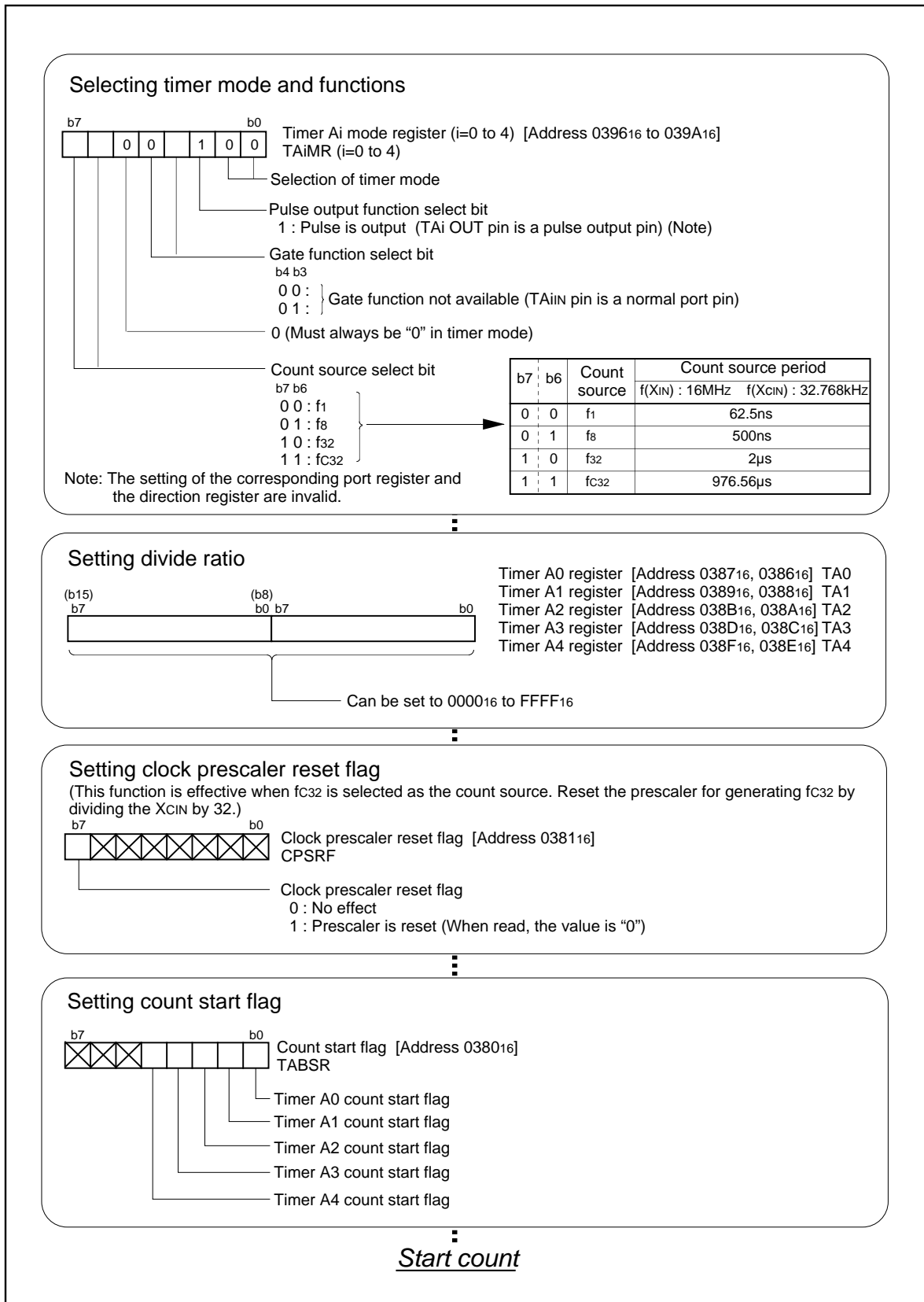
**Table 2.2.3. Chosen functions**

Item	Set-up	
Count source	O	Internal count source(f <sub>1</sub> / f <sub>8</sub> / f <sub>32</sub> / f <sub>c32</sub> )
Pulse output function		No pulses output
	O	Pulses output
Gate function	O	No gate function
		Performs count only for the period in which the TAIIN pin is at "L" level
		Performs count only for the period in which the TAIIN pin is at "H" level

- Operation (1) Setting the count start flag to "1" causes the counter to perform a down count on the count source.
- (2) If an underflow occurs, the content of the reload register is reloaded and the count continues. At this time, the timer Ai interrupt request bit goes to "1". Also, the output polarity of the TAIOUT pin reverses.
- (3) Setting the count start flag to "0" causes the counter to hold its value and to stop. Also, the TAIOUT pin outputs an "L" level.



**Figure 2.2.10. Operation timing of timer mode, pulse output function selected**



## 2.2.5 Operation of Timer A (event counter mode, reload type selected)

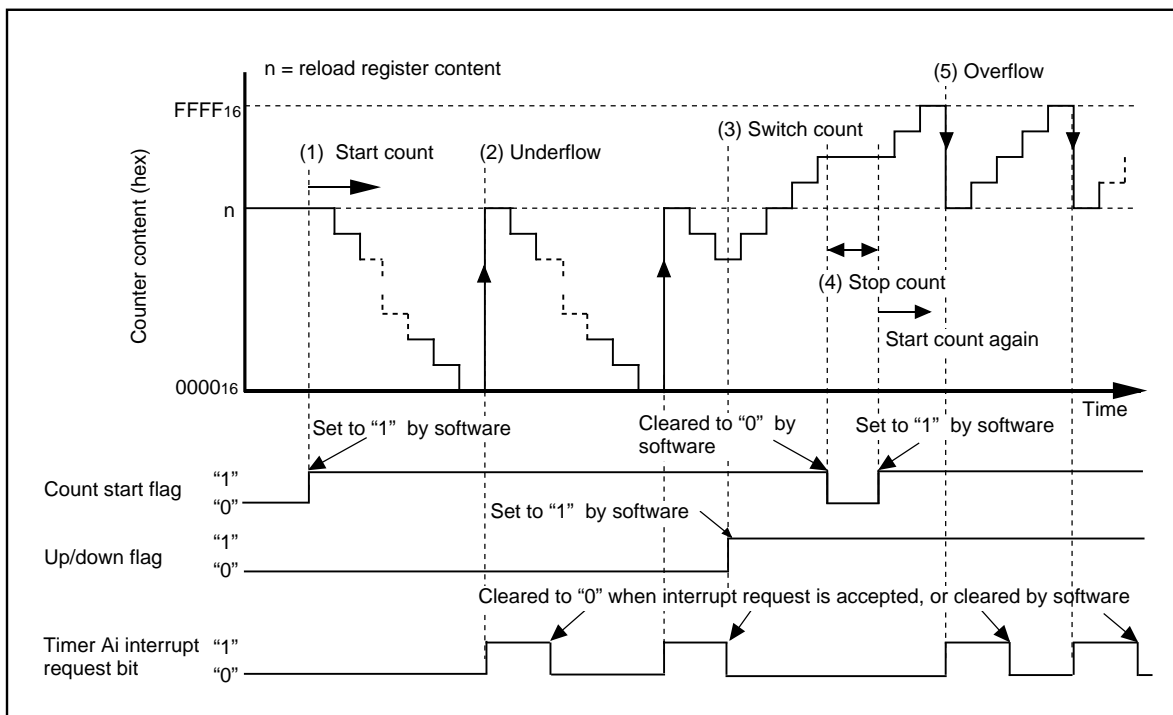
In event counter mode, choose functions from those listed in Table 2.2.4. Operations of the circled items are described below. Figure 2.2.12 shows the operation timing, and Figure 2.2.13 shows the set-up procedure.

**Table 2.2.4. Chosed functions**

Item	Set-up	Item	Set-up
Count source	○ Input signal to TAIin (counting falling edges)	Pulse output function	○ No pulses output
			Pulses output
	Input signal to TAIin (counting rising edges)	Count operation type	○ Reload type
	Free-run type		
TAj overflow		Factor for switching between up and down	○ Content of up/down flag
			Input signal to TAIout

Note:  $j = i - 1$ , but  $j = 4$  when  $i = 0$ .

- Operation (1) Setting the count start flag to “1” causes the counter to count the falling edges of the count source.
- (2) If an underflow occurs, the content of the reload register is reloaded, and the count continues. At this time, the timer Ai interrupt request bit goes to “1”.
- (3) If switching from an up count to a down count or vice versa while a count is in progress, the switch takes effect from the next effective edge of the count source.
- (4) Setting the count start flag to “0” causes the counter to hold its value and to stop.
- (5) If an overflow occurs, the content of the reload register is reloaded, and the count continues. At this time, the timer Ai interrupt request bit goes to “1”.



**Figure 2.2.12. Operation timing of event counter mode, reload type selected**

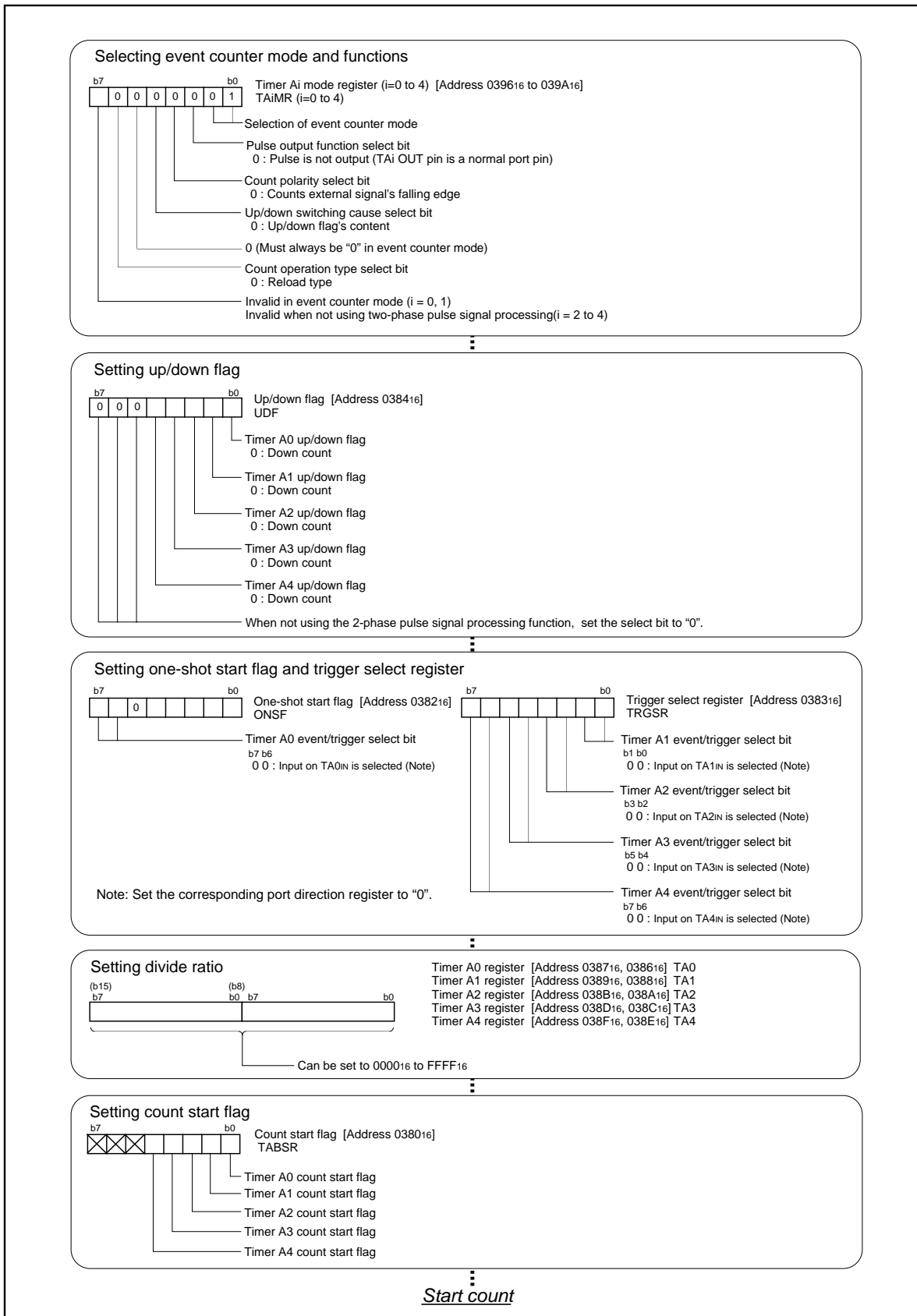


Figure 2.2.13. Set-up procedure of event counter mode, reload type selected

## 2.2.6 Operation of Timer A (event counter mode, free run type selected)

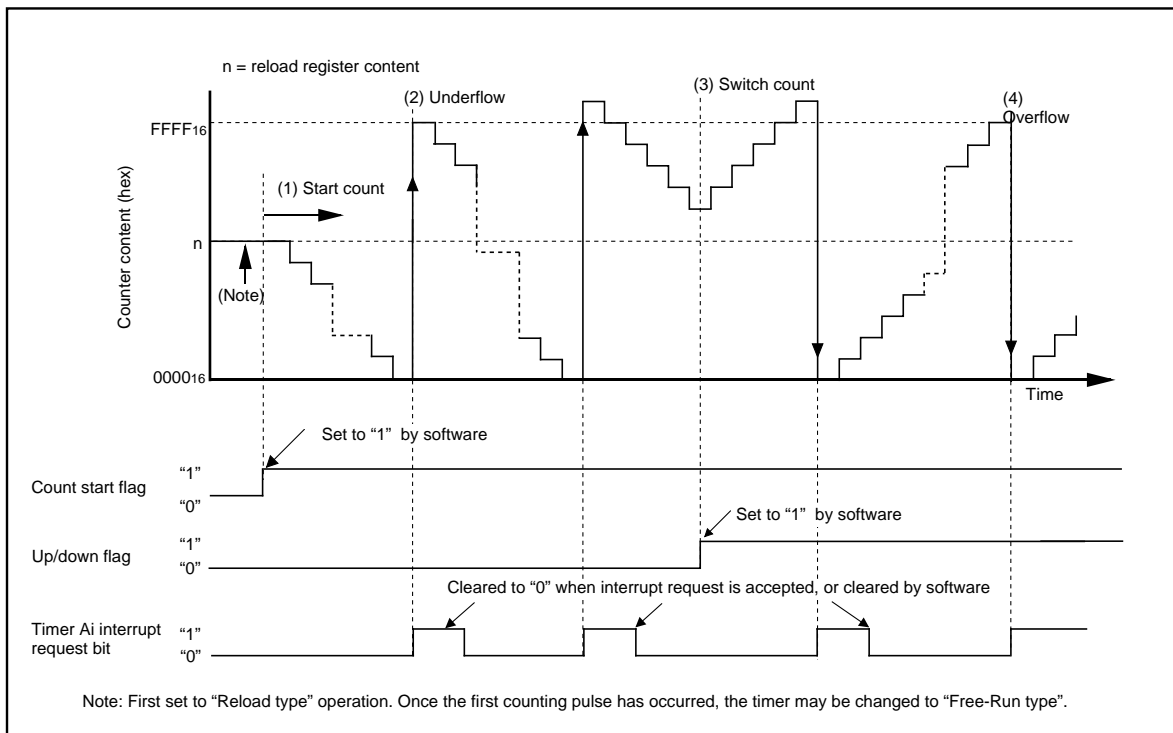
In event counter mode, choose functions from those listed in Table 2.2.5. Operations of the circled items are described below. Figure 2.2.14 shows the operation timing, and Figure 2.2.15 shows the set-up procedure.

**Table 2.2.5. Chosen functions**

Item	Set-up	Item	Set-up
Count source	○ Input signal to TAI <sub>iN</sub> (counting falling edges)	Pulse output function	○ No pulses output
	Input signal to TAI <sub>iN</sub> (counting rising edges)		Pulses output
		TA <sub>j</sub> overflow	Count operation type
			○ Free-run type
		Factor for switching between up and down	○ Content of up/down flag
			Input signal to TAI <sub>iOUT</sub>

Note:  $j = i - 1$ , but  $j = 4$  when  $i = 0$

- Operation (1) Setting the count start flag to “1” causes the counter to count the falling edges of the count source.
- (2) Even if an underflow occurs, the content of the reload register is not reloaded, but the count continues. At this time, the timer Ai interrupt request bit goes to “1”.
- (3) If switching from an up count to a down count or vice versa while a count is in progress, the switch takes effect from the next effective edge of the count source.
- (4) Even if an overflow occurs, the content of the reload register is not reloaded, but the count continues. At this time, the timer Ai interrupt request bit goes to “1”.



**Figure 2.2.14. Operation timing of event counter mode, free run type selected**

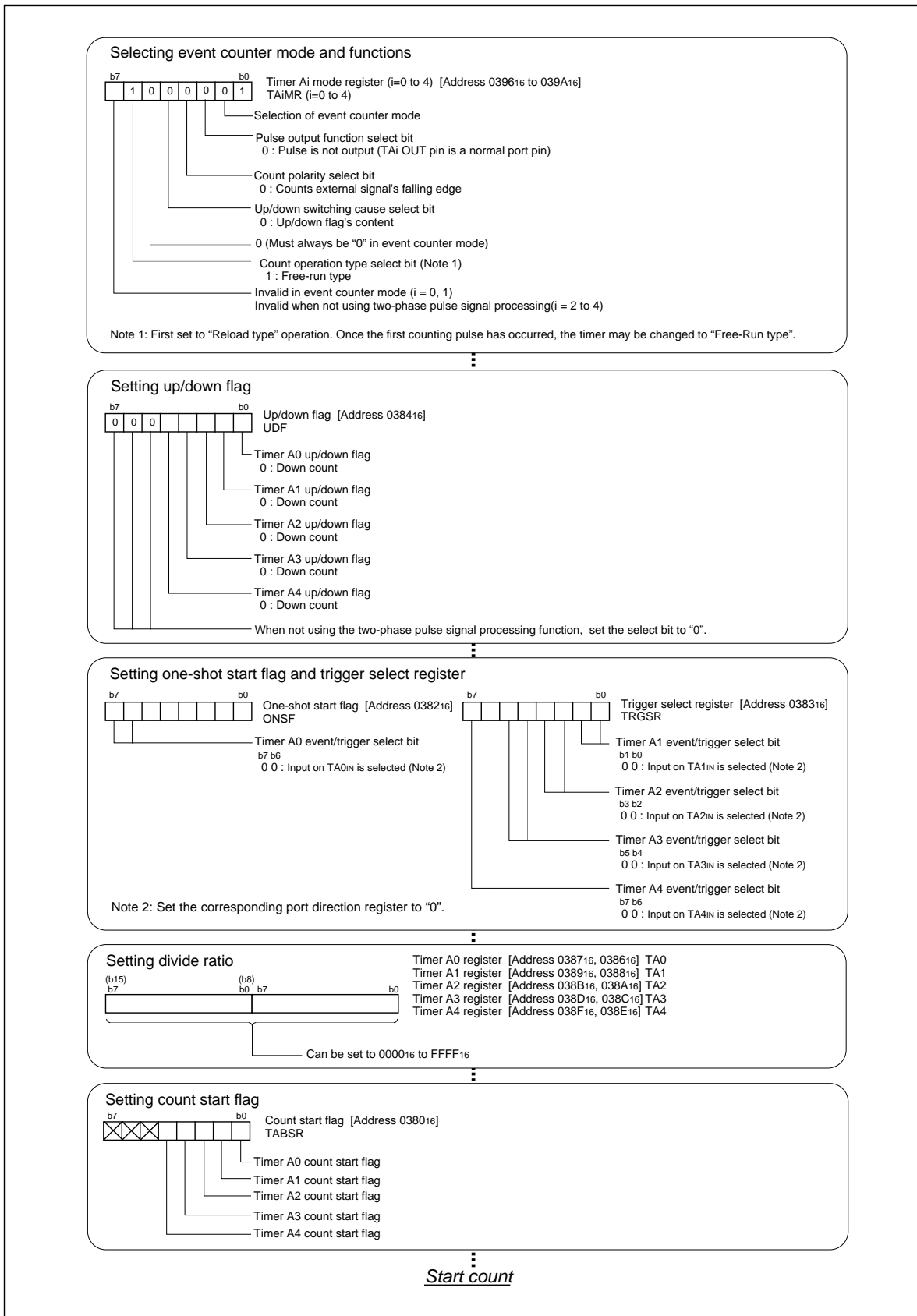


Figure 2.2.15. Set-up procedure of event counter mode, free run type selected

### 2.2.7 Operation of timer A (two-phase pulse signal process in event counter mode, normal mode selected)

In processing two-phase pulse signals in event counter mode, choose functions from those listed in Table 2.2.6. Operations of the circled items are described below. Figure 2.2.16 shows the operation timing, and Figure 2.2.17 shows the set-up procedure.

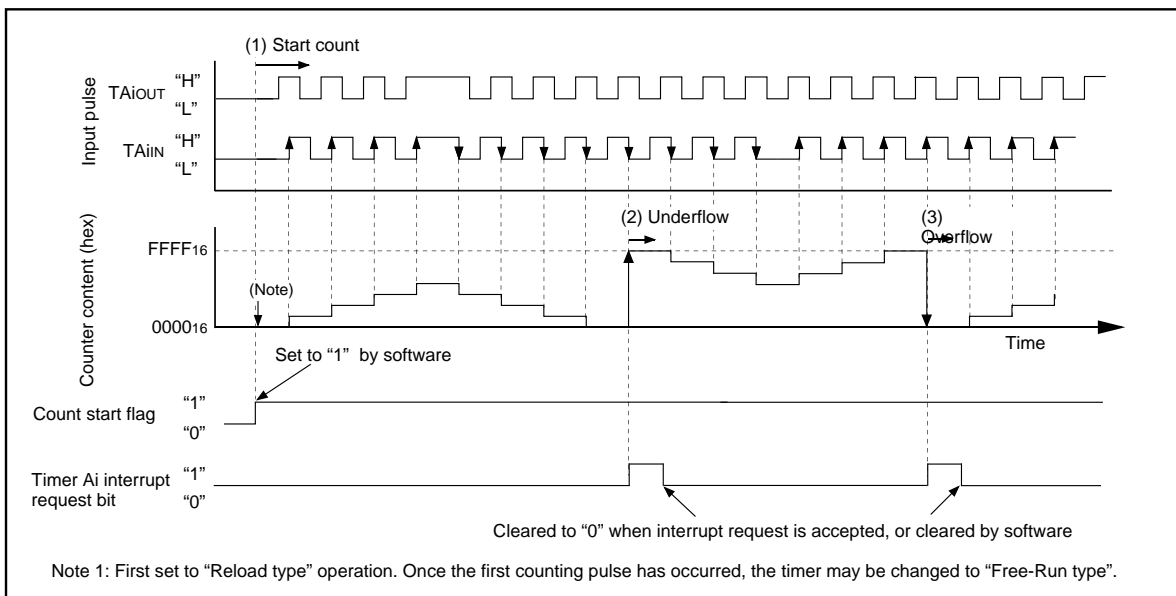
**Table 2.2.6. Chosen functions**

Item	Set-up
Count operation type	Reload type
	<input type="radio"/> Free run type
Two-phase pulses process (Note)	<input type="radio"/> Normal processing
	4-multiplication processing

Note: Timer A3 alone can be selected. Timer A2 is solely used for normal processes, and timer A4 is solely used for 4 multiplication processes.

- Operation
- (1) Setting the count start flag to "1" causes the counter to count effective edges of the count source.
  - (2) Even if an underflow occurs, the content of the reload register is not reloaded, but the count continues. At this time, the timer Ai interrupt request bit goes to "1".
  - (3) Even if an overflow occurs, the content of the reload register is not reloaded, but the count continues. At this time, the timer Ai interrupt request bit goes to "1".

- Note
- The up count or down count conditions are as follows:
    - If a rising edge is present at the TAIiN pin when the input signal level to the TAIoUT pin is "H", an up count is performed.
    - If a falling edge is present at the TAIiN pin when the input signal level to the TAIoUT pin is "H", a down count is performed.
  - Set TAIiN pin and TAIoUT pin's port direction register to "0".



**Figure 2.2.16. Operation timing of two-phase pulse signal process in event counter mode, normal mode selected**

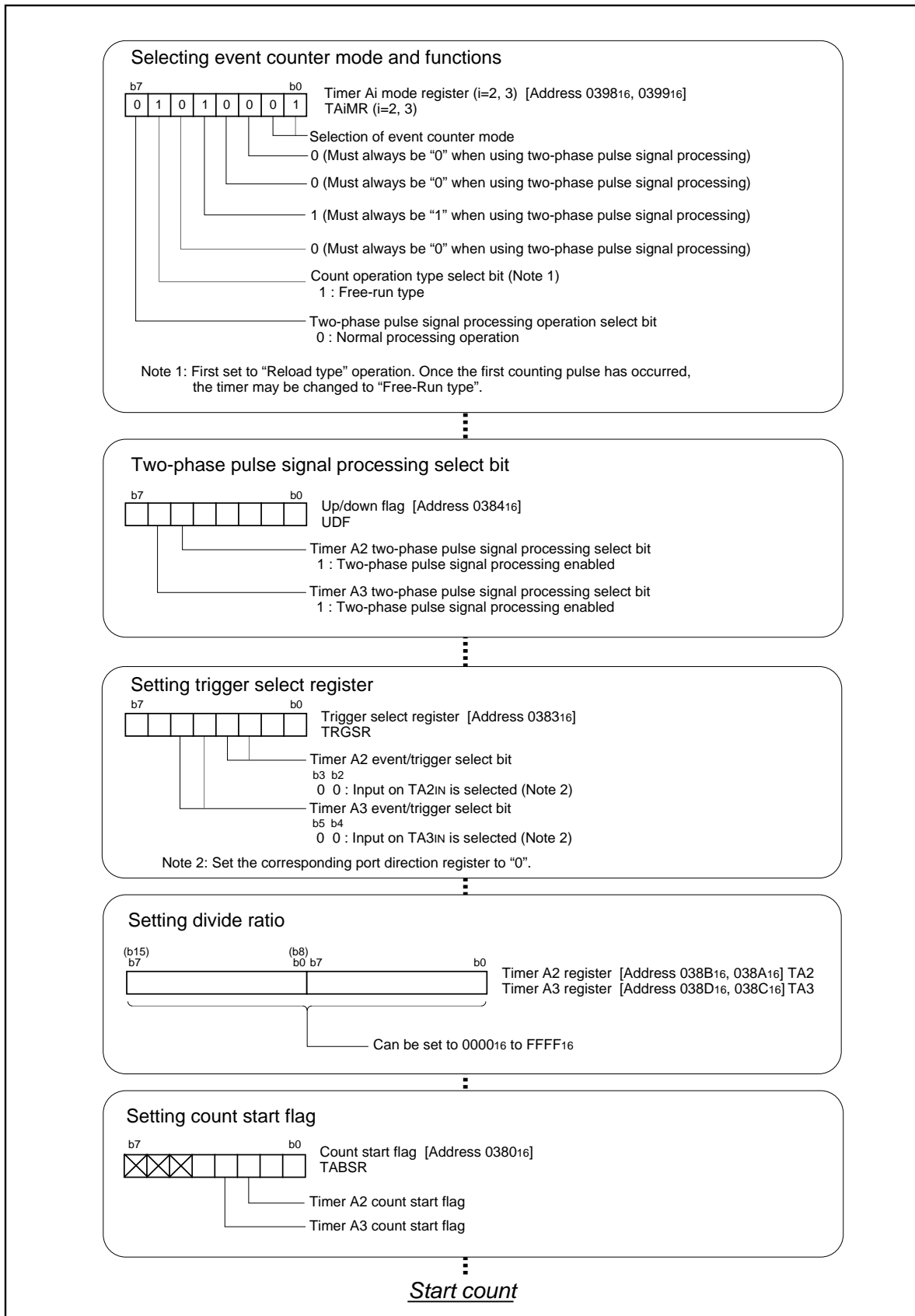


Figure 2.2.17. Set-up procedure of two-phase pulse signal process in event counter mode, normal mode selected



## 2.2.8 Operation of timer A (two-phase pulse signal process in event counter mode, multiply-by-4 mode selected)

In processing two-phase pulse signals in event counter mode, choose functions from those listed in Table 2.2.7. Operations of the circled items are described below. Figure 2.2.18 shows the operation timing, and Figure 2.2.19 shows the set-up procedure.

**Table 2.2.7. Chosed functions**

Item	Set-up		Item	Set-up	
Count operation type		Reload type	Processing two- phase pulses (Note)		Normal processing
	○	Free run type		○	4-multiplication processing

Note: Timer A3 alone can be selected. Timer A2 is solely used for normal processes, and timer A4 is solely used for 4-multiplication processes.

- Operation
- (1) Setting the count start flag to "1" causes the counter to count effective edges of the count source.
  - (2) Even if an underflow occurs, the content of the reload register is not reloaded, but the count continues. At this time, the interrupt request bit goes to "1".
  - (3) Even if an overflow occurs, the content of the reload register is not reloaded, but the count continues. At this time, the interrupt request bit goes to "1".

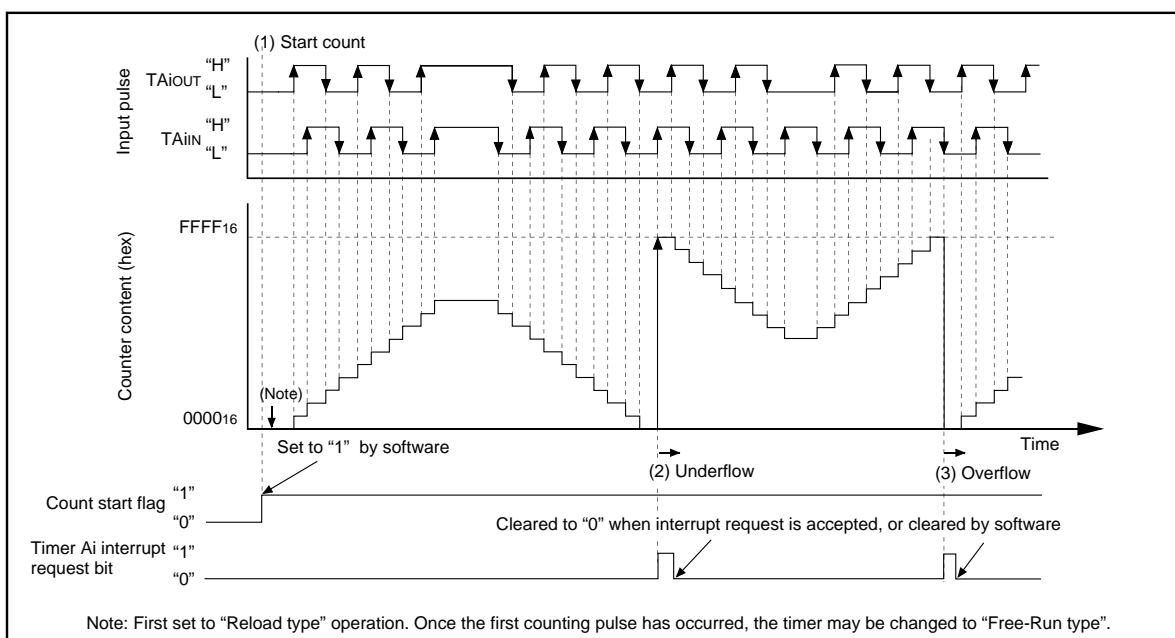
Note

- The up count or down count conditions are as follows:

**Table 2.2.8. The up count or down count conditions**

	Input signal to the TAIOUT pin	Input signal to the TAIIN pin		Input signal to the TAIOUT pin	Input signal to the TAIIN pin
Up count	"H" level	Rising	Down count	"H" level	Falling
	"L" level	Falling		"L" level	Rising
	Rising	"L" level		Rising	"H" level
	Falling	"H" level		Falling	"L" level

- Set TAIIN pin and TAIOUT pin's port direction register to "0".



**Figure 2.2.18. Operation timing of two-phase pulse signal process in event counter mode, multiply-by-4 mode selected**

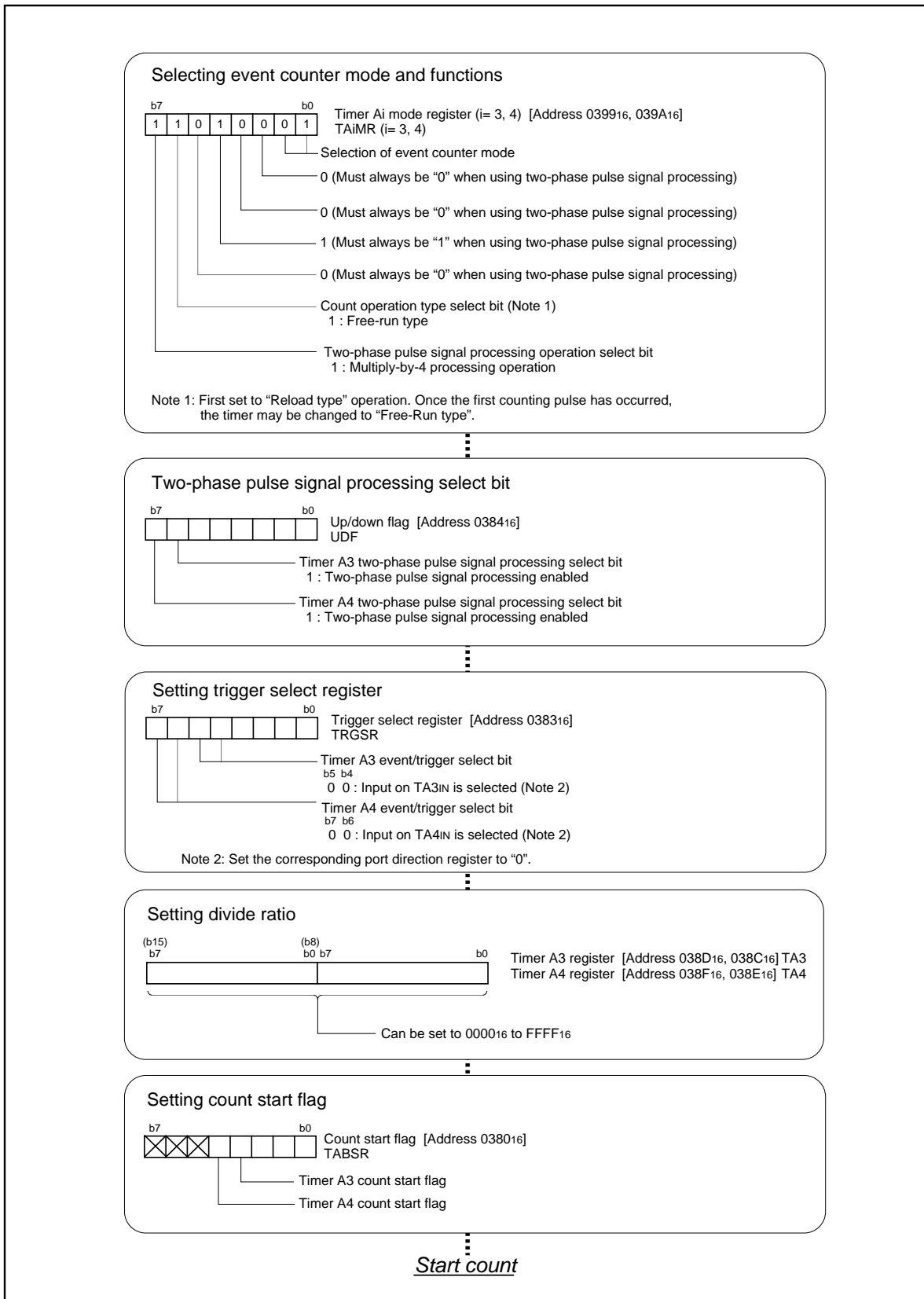


Figure 2.2.19. Set-up procedure of two-phase pulse signal process in event counter mode, multiply-by-4 mode selected

## 2.2.9 Operation of Timer A (one-shot timer mode)

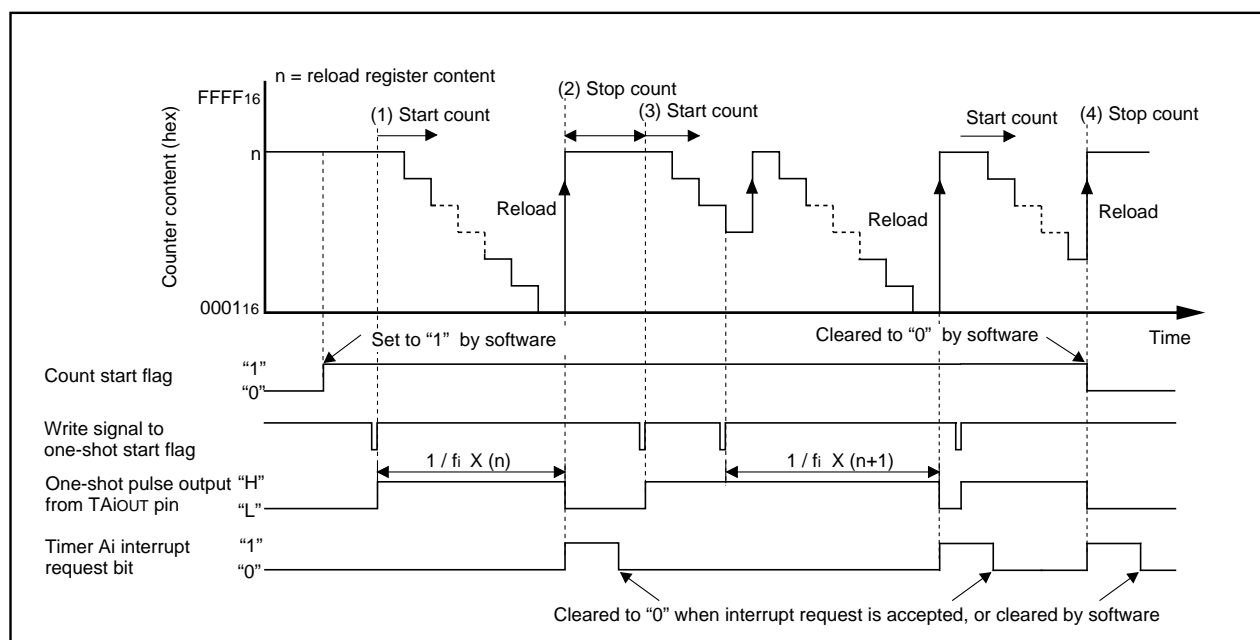
In one-shot timer mode, choose functions from those listed in Table 2.2.9. Operations of the circled items are described below. Figure 2.2.20 shows the operation timing, and Figures 2.2.21 shows the set-up procedure.

**Table 2.2.9. Chosed functions**

Item	Set-up
Count source	○ Internal count source ( $f_1 / f_8 / f_{32} / f_{c32}$ )
Pulse output function	No pulses output
	○ Pulses output
Count start condition	External trigger input (falling edge of input signal to the TAIIN pin)
	External trigger input (rising edge of input signal to the TAIIN pin)
	Timer overflow (TAJ/TAK overflow)
	○ Writing "1" to the one-shot start flag

Note:  $j = i - 1$ , but  $j = 4$  when  $i = 0$ ;  $k = i + 1$ , but  $k = 0$  when  $i = 4$ .

- Operation
- (1) Setting the one-shot start flag to "1" with the count start flag set to "1" causes the counter to perform a down count on the count source. At this time, the TAIOUT pin outputs an "H" level.
  - (2) The instant the value of the counter becomes "0000<sub>16</sub>", the TAIOUT pin outputs an "L" level, and the counter reloads the content of the reload register and stops counting. At this time, the timer Ai interrupt request bit goes to "1".
  - (3) If a trigger occurs while a count is in progress, the counter reloads the value in the reload register again and continues counting. The reload timing is in step with the next count source input after the trigger.
  - (4) Setting the count start flag to "0" causes the counter to stop and to reload the content of the reload register. Also, the TAIOUT pin outputs an "L" level. At this time, the timer Ai interrupt request bit goes to "1".



**Figure 2.2.20. Operation timing of one-shot mode**

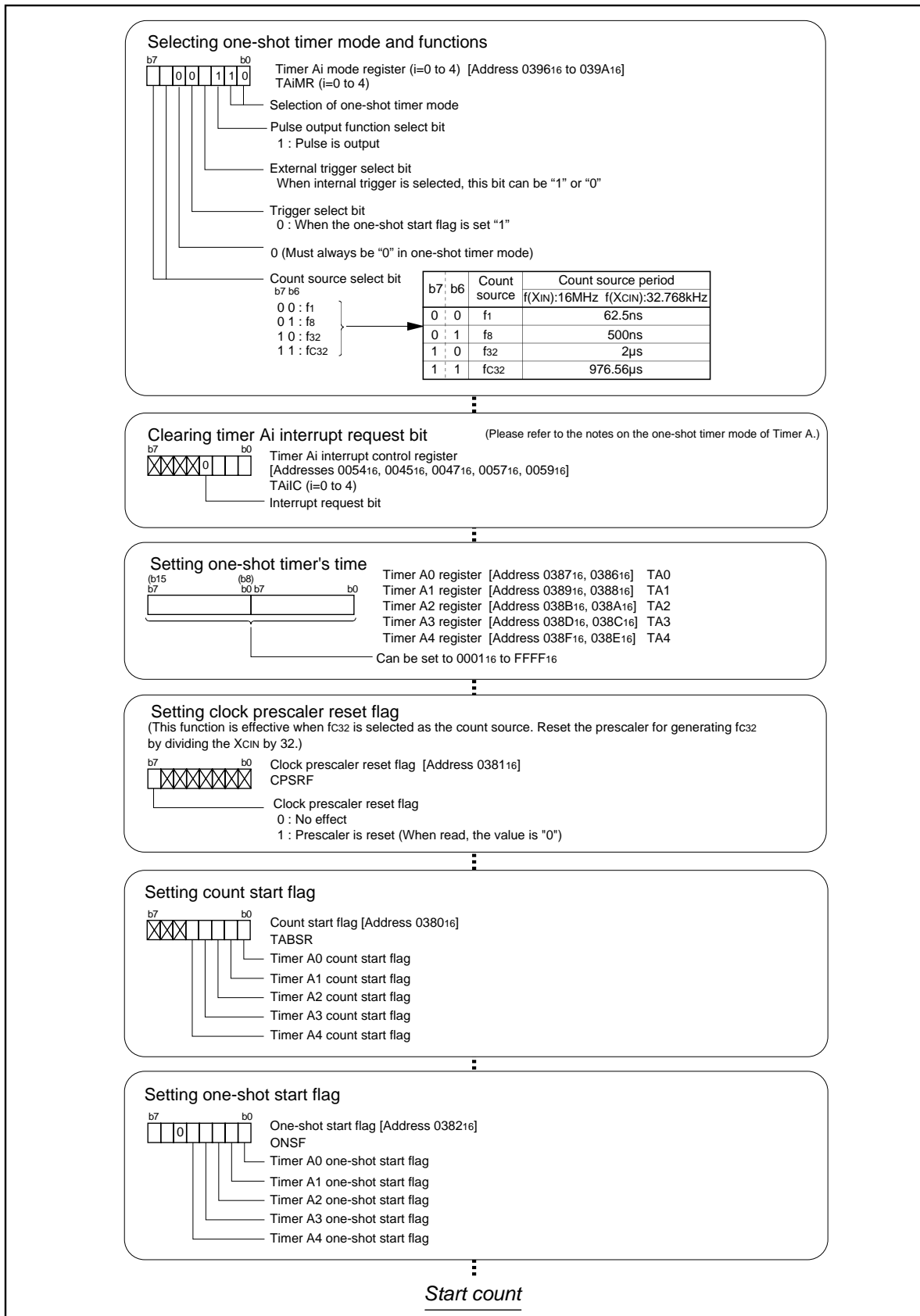


Figure 2.2.21. Set-up procedure of one-shot mode

### 2.2.10 Operation of Timer A (pulse width modulation mode, 16-bit PWM mode selected)

In pulse width modulation mode, choose functions from those listed in Table 2.2.10. Operations of the circled items are described below. Figure 2.2.22 shows the operation timing, and Figures 2.2.23 and 2.2.24 show the set-up procedure.

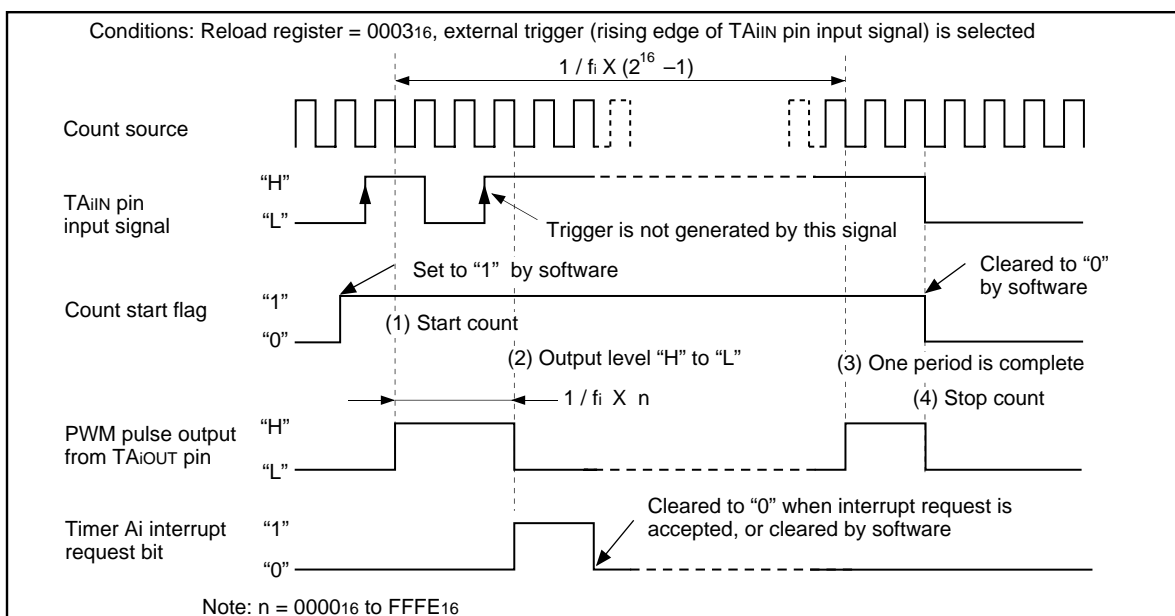
**Table 2.2.10. Chosed functions**

Item	Set-up	
Count source	<input type="radio"/>	Internal count source ( $f_1 / f_8 / f_{32} / f_{c32}$ )
PWM mode	<input type="radio"/>	16-bit PWM
	<input type="radio"/>	8-bit PWM
Count start condition	<input type="radio"/>	External trigger input (falling edge of input signal to the TAIiN pin)
	<input type="radio"/>	External trigger input (rising edge of input signal to the TAIiN pin)
	<input type="radio"/>	Timer overflow (TAj/TAK overflow)

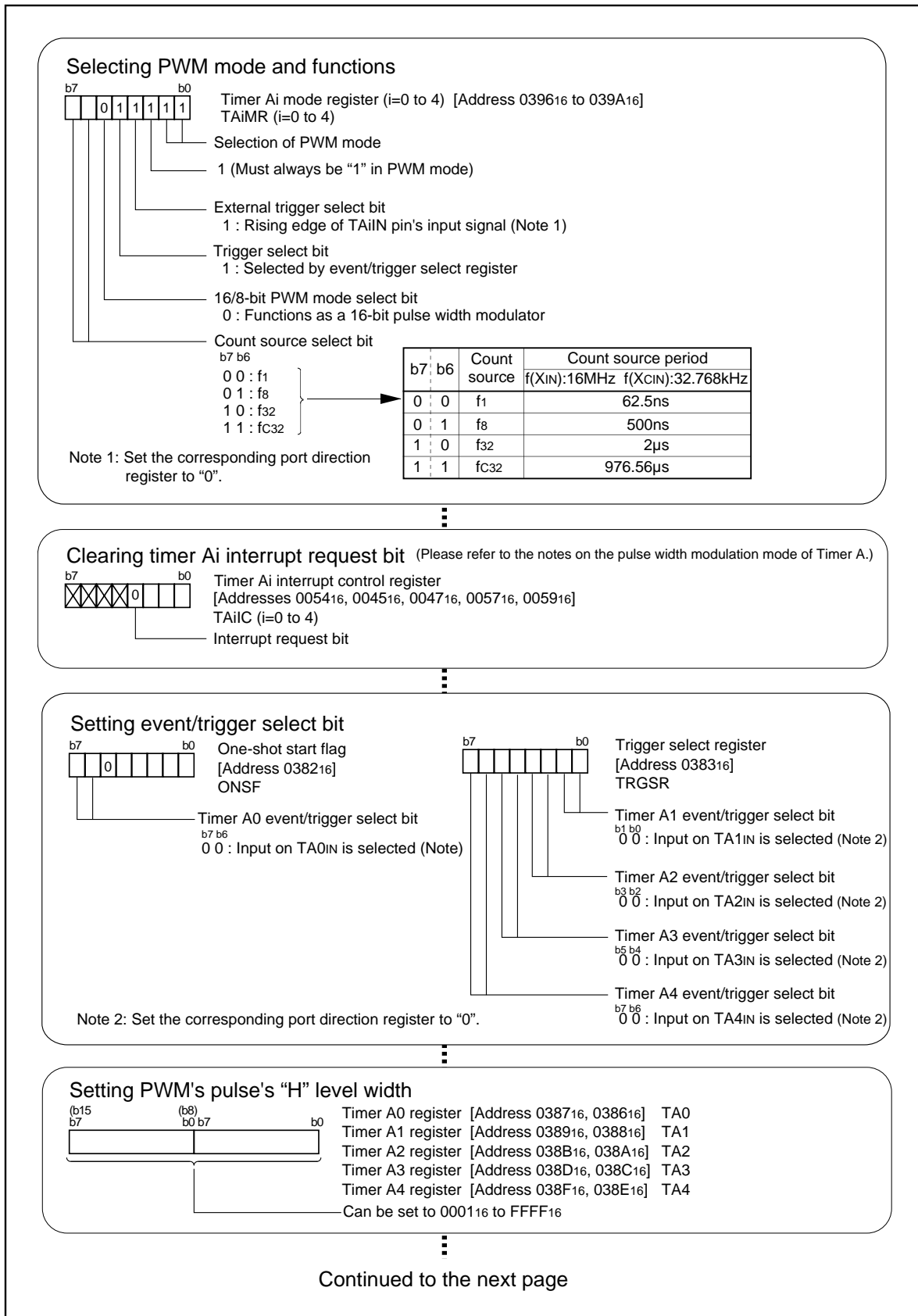
Note:  $j = i - 1$ , but  $j = 4$  when  $i = 0$ ;  $k = i + 1$ , but  $k = 0$  when  $i = 4$ .

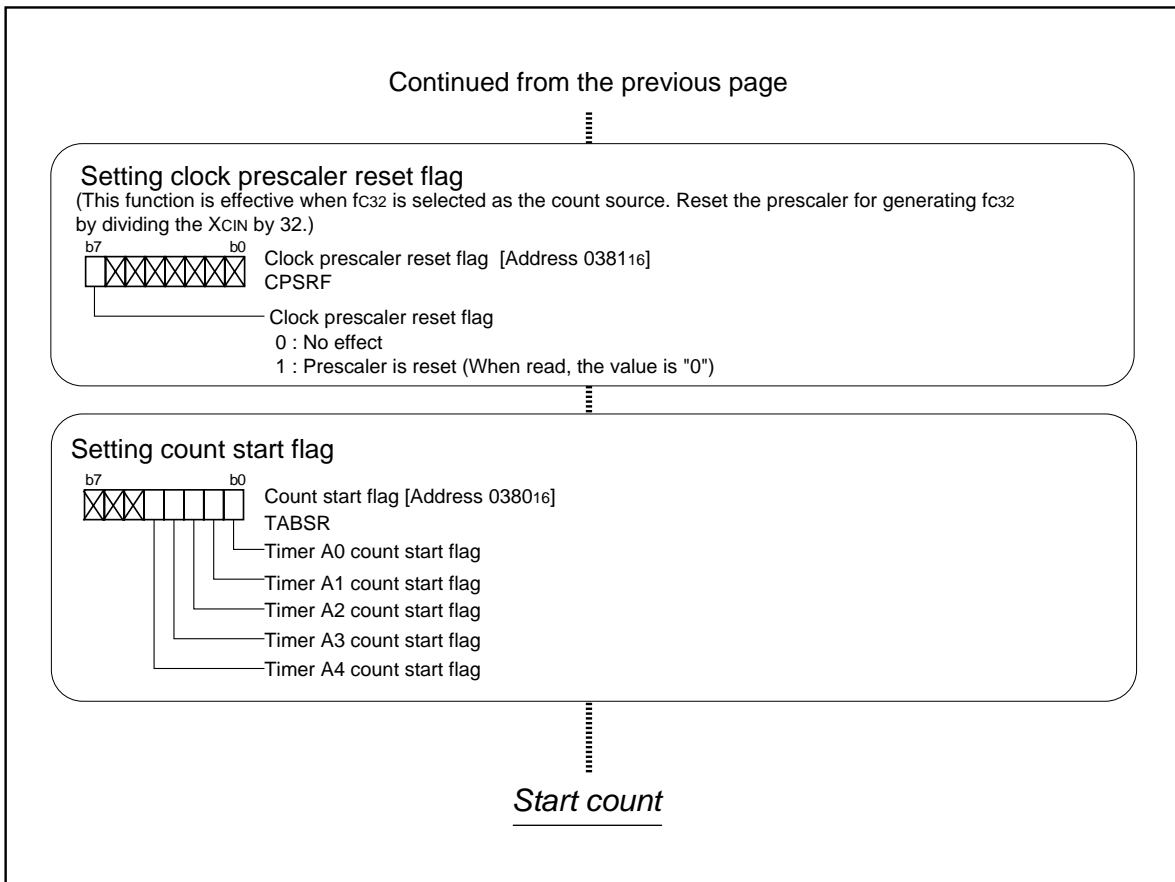
- Operation
- (1) If the TAIiN pin input level changes from "L" to "H" with the count start flag set to "1", the counter performs a down count on the count source. Also, the TAIOUT pin outputs an "H" level.
  - (2) The TAIOUT pin output level changes from "H" to "L" when a set time period elapses. At this time, the timer Ai interrupt request bit goes to "1".
  - (3) The counter reloads the content of the reload register every time PWM pulses are output for one cycle, and continues counting.
  - (4) Setting the count start flag to "0" causes the counter to hold its value and to stop. Also, the TAIOUT outputs an "L" level.

- Note
- The period of PWM pulses becomes  $(2^{16} - 1)/f_i$ , and the "H" level pulse width becomes  $n/f_i$ . If the timer Ai register is set to "0000<sub>16</sub>", the pulse width modulator does not work, and the the TAIOUT pin output level remains at "L".  
( $f_i$  : frequency of the count source  $f_1, f_8, f_{32}, f_{c32}$ ;  $n$  : value of the timer)
  - Set TAIiN pin's port direction register to "0".



**Figure 2.2.22. Operation timing of pulse width modulation mode, 16-bit PWM mode selected**





**Figure 2.2.24. Set-up procedure of pulse width modulation mode, 16-bit PWM mode selected (2)**

**2.2.11 Operation of Timer A (pulse width modulation mode, 8-bit PWM mode selected)**

In pulse width modulation mode, choose functions from those listed in Table 2.2.11. Operations of the circled items are described below. Figure 2.2.25 shows the operation timing, and Figures 2.2.26 and 2.2.27 show the set-up procedure.

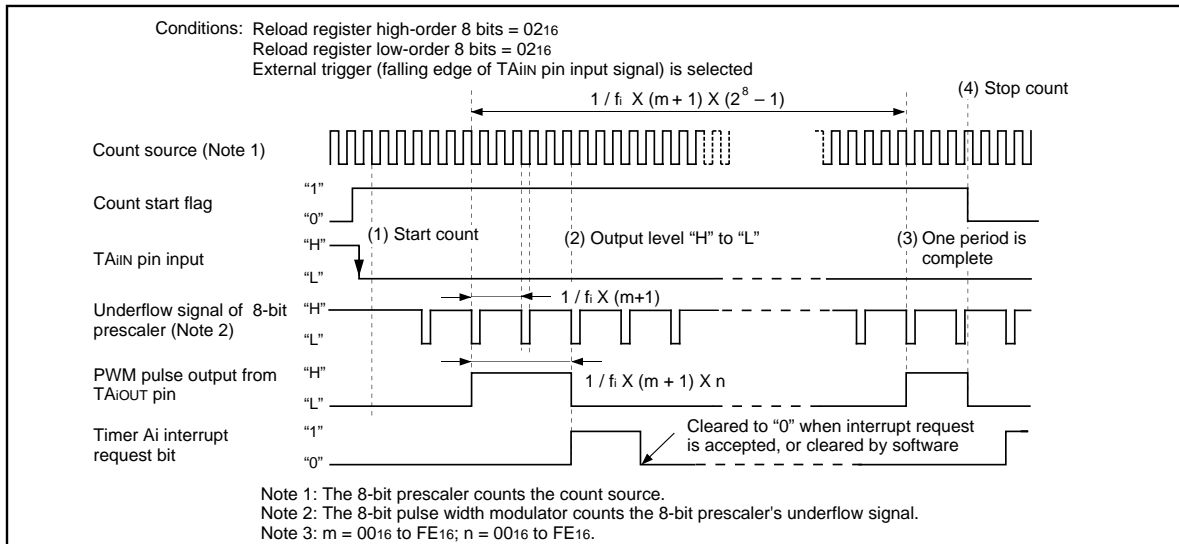
**Table 2.2.11. Chosed functions**

Item	Set-up	
Count source	<b>○</b>	Internal count source (f <sub>1</sub> / f <sub>8</sub> / f <sub>32</sub> / f <sub>c32</sub> )
PWM mode		16-bit PWM
	<b>○</b>	8-bit PWM
Count start condition	<b>○</b>	External trigger input (falling edge of input signal to the TAIiN pin)
		External trigger input (rising edge of input signal to the TAIiN pin)
		Timer overflow (TAj/TAK overflow)

Note: j = i - 1, but j = 4 when i = 0; k = i + 1, but k = 0 when i = 4.

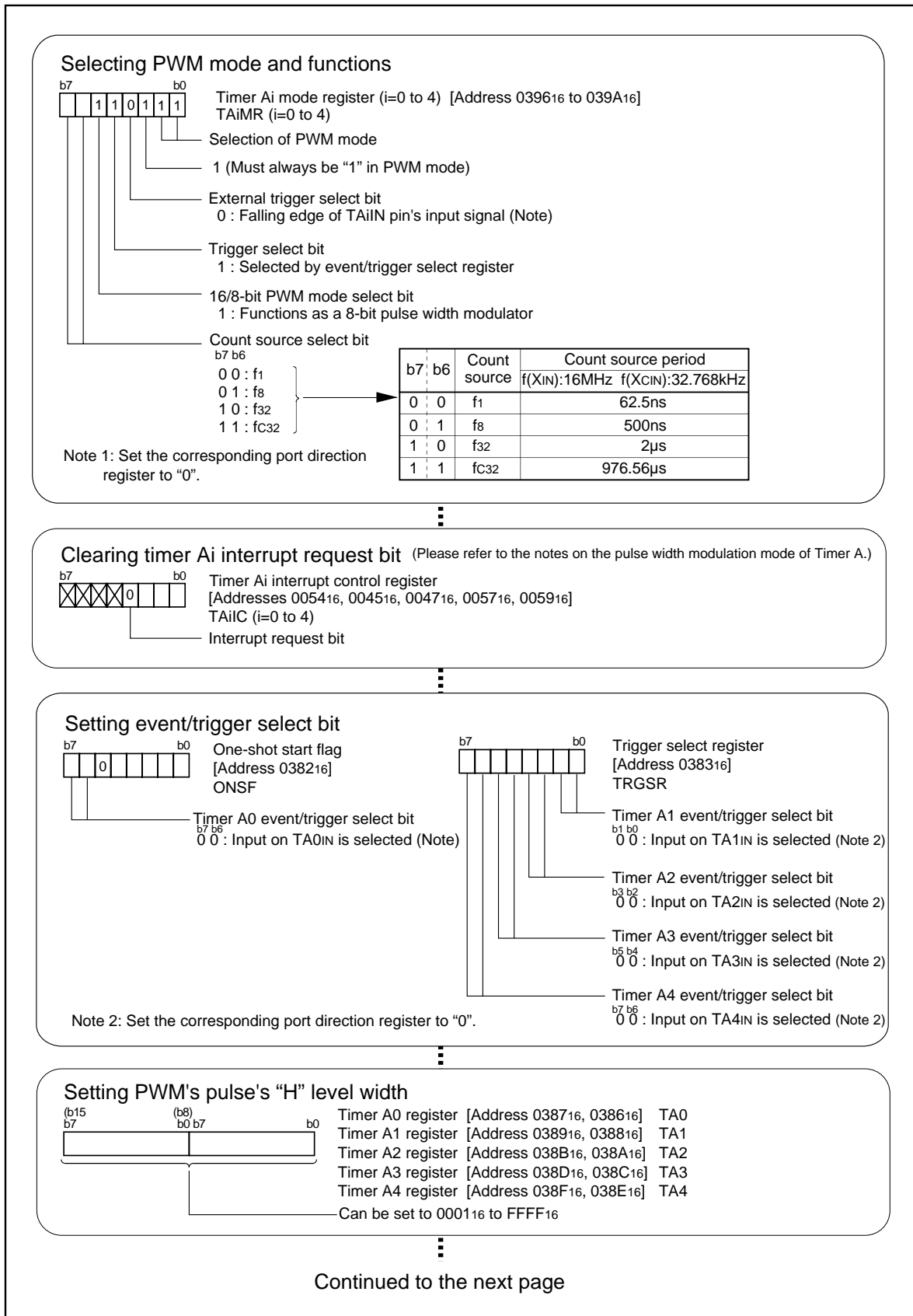
- Operation
- (1) If the TAIiN pin input level changes from “H” to “L” with the count start flag set to “1”, the counter performs a down count on the count source. Also, the TAIOUT pin outputs an “H” level.
  - (2) The TAIOUT pin output level changes from “H” to “L” when a set time period elapses. At this time, the timer Ai interrupt request bit goes to “1”.
  - (3) The counter reloads the content of the reload register every time PWM pulses are output for one cycle, and continues counting.
  - (4) Setting the count start flag to “0” causes the counter to hold its value and to stop. Also, the TAIOUT pin outputs an “L” level.

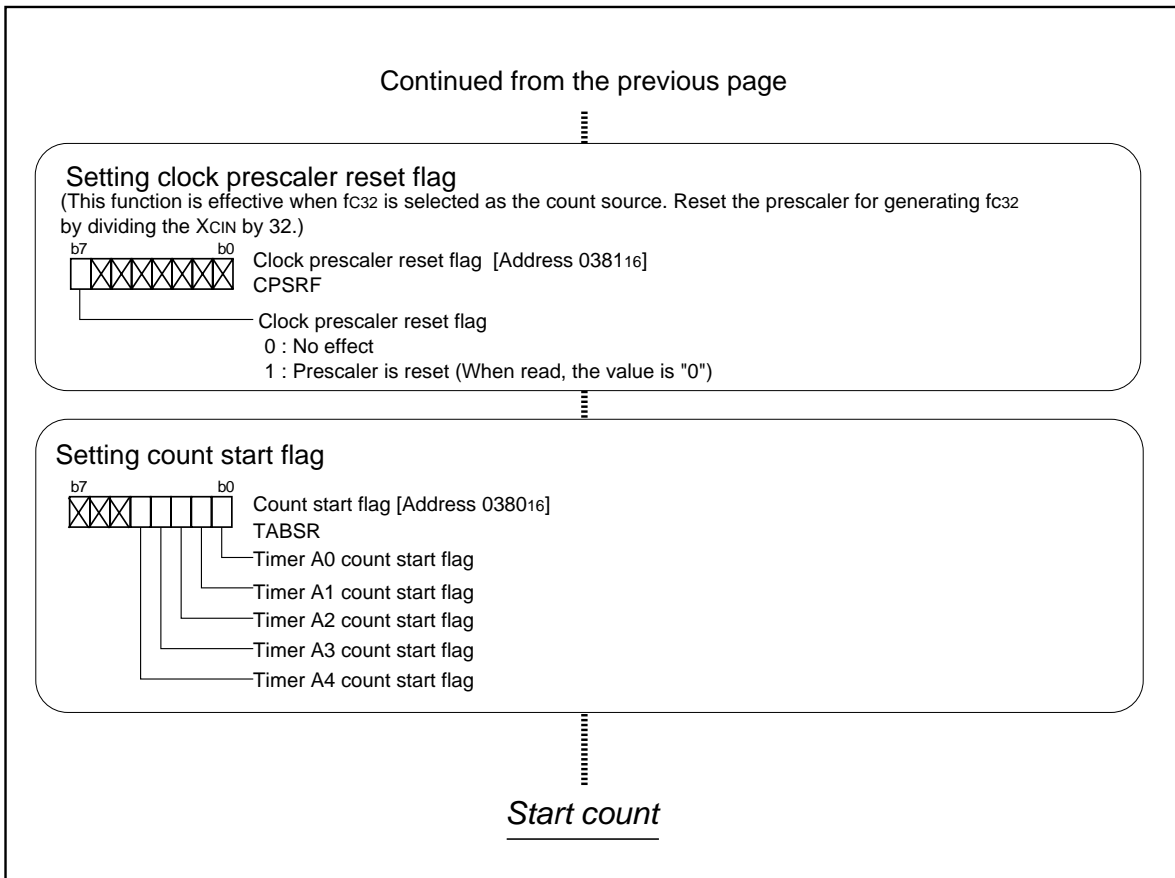
- Note
- The period of PWM pulses becomes  $(m + 1) \times (2^8 - 1) / f_i$ , and the “H” level pulse width becomes  $n \times (m + 1) / f_i$ . If “0016” is set in the eight higher-order bits of the timer Ai register, the pulse width modulator does not work, and the the TAIOUT pin output level remains at “L”.
  - (f<sub>i</sub> : frequency of the count source f<sub>1</sub>, f<sub>8</sub>, f<sub>32</sub>, f<sub>c32</sub>; n : value of the timer)
  - When a trigger is generated, the TAIout pin outputs “L” level of same amplitude as “H” level of the set PWM pulse, after which it starts PWM pulse output.
  - Set TAIiN pin’s port direction register to “0”.



**Figure 2.2.25. Operation timing of pulse width modulation mode, with 8-bit PWM mode selected**



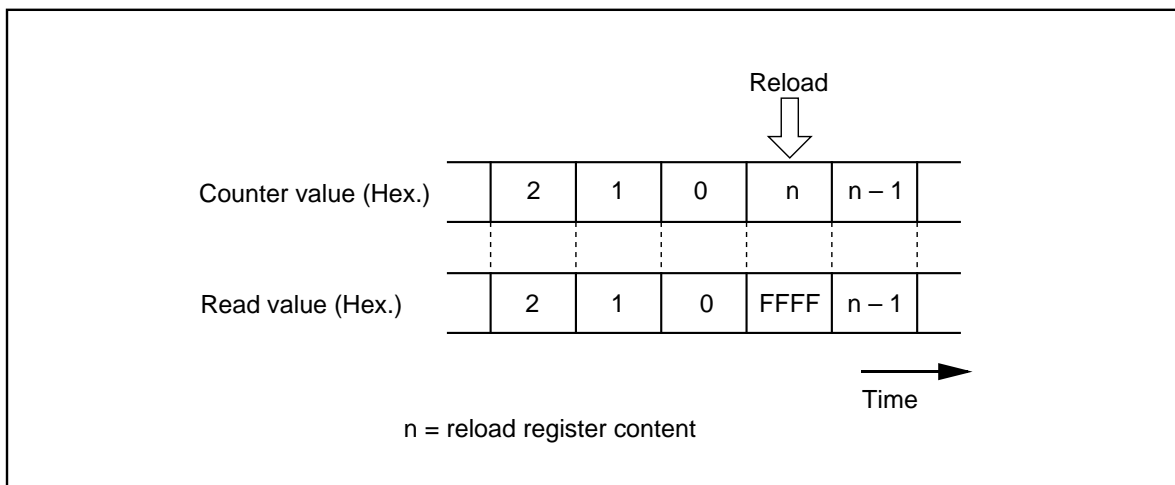




**Figure 2.2.27. Set-up procedure of pulse width modulation mode, 8-bit PWM mode selected (2)**

### 2.2.12 Precautions for Timer A (timer mode)

- (1) To clear reset, the count start flag is set to "0". Set a value in the timer Ai register, then set the flag to "1".
- (2) Reading the timer Ai register while a count is in progress allows reading, with arbitrary timing, the value of the counter. Reading the timer Ai register with the reload timing shown in Figure 2.2.28 gets "FFFF<sub>16</sub>". Reading the timer Ai register after setting a value in the timer Ai register with a count halted but before the counter starts counting gets a proper value.



**Figure 2.2.28. Reading timer Ai register**

### 2.2.13 Precautions for Timer A (event counter mode)

- (1) To clear reset, the count start flag is set to "0". Set a value in the timer Ai register, then set the flag to "1".
- (2) Reading the timer Ai register while a count is in progress allows reading, with arbitrary timing, the value of the counter. Reading the timer Ai register with the reload timing shown in Figure 2.2.29 gets "FFFF<sub>16</sub>" by underflow or "0000<sub>16</sub>" by overflow. Reading the timer Ai register after setting a value in the timer Ai register with a count halted but before the counter starts counting gets a proper value.
- (3) Please note the standards for the differences between the 2 pulses used in the two-phase pulse signals input signals to the TAiIN pin and TAiOUT pin (i = 2, 3, 4), as shown in Figure 2.2.30.
- (4) When free run type is selected, if count is stopped, set a value in the timer Ai register again.

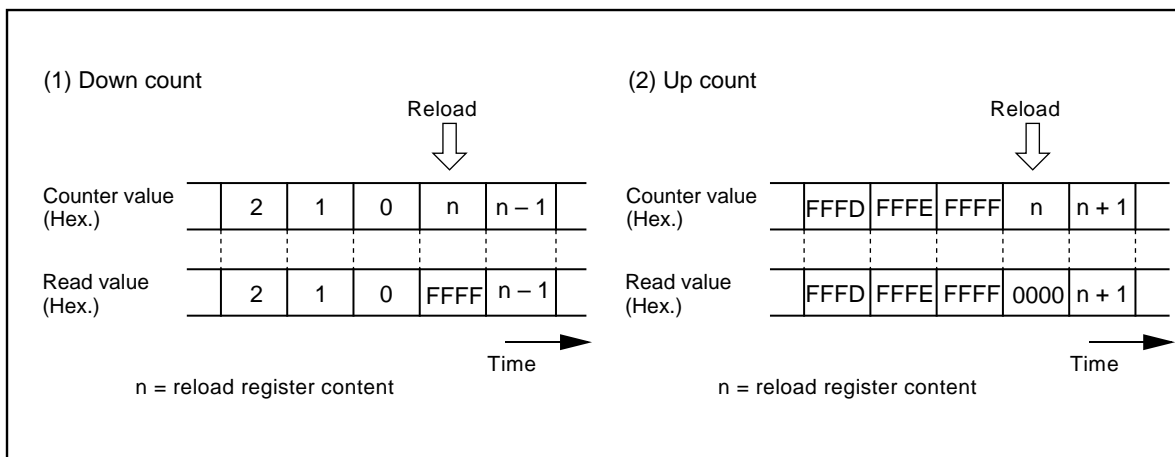


Figure 2.2.29. Reading timer Ai register

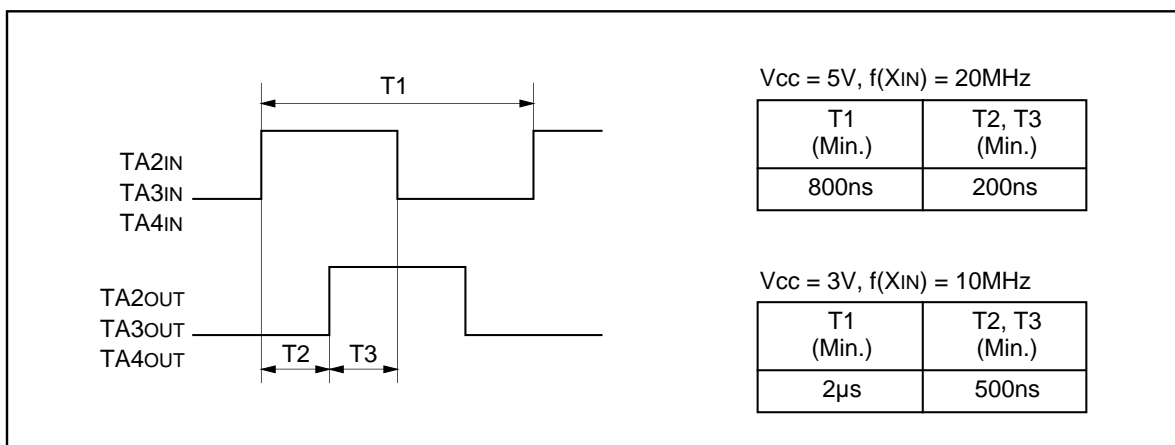


Figure 2.2.30. Standard of 2-phase pulses

- (5) In the case of using as “Free-Run type”, the timer register contents may be unknown when counting begins. If the timer register is set before counting has started, then the starting value will be unknown.
- In the case where the up/down count will not be changed.

Enable the “Reload” function and write to the timer register before counting begins. Rewrite the value to the timer register immediately after counting has started. If counting up, rewrite “0000<sub>16</sub>” to the timer register. If counting down, rewrite “FFFF<sub>16</sub>” to the timer register. This will cause the same operation as “Free-Run type” mode.
  - In the case where the up/down count has changed.

First set to “Reload type” operation. Once the first counting pulse has occurred, the timer may be changed to “Free-Run type”.

### 2.2.14 Precautions for Timer A (one-shot timer mode)

- (1) At reset, the count start flag is set to "0". Set a value in the timer Ai register, then set the flag to "1".
- (2) Setting the count start flag to "0" while a count is in progress causes as follows:
  - The counter stops counting and a content of reload register is reloaded.
  - The TAIOUT pin outputs "L" level.
  - The interrupt request generated and the timer Ai interrupt request bit goes to "1".
- (3) The timer Ai interrupt request bit goes to "1" if the timer's operation mode is set using any of the following procedures:
  - Selecting one-shot timer mode after reset.
  - Changing operation mode from timer mode to one-shot timer mode.
  - Changing operation mode from event counter mode to one-shot timer mode.Therefore, to use timer Ai interrupt (interrupt request bit), set timer Ai interrupt request bit to "0" after the above listed changes have been made.
- (4) If a trigger occurs while a count is in progress, after the counter performs one down count following the reoccurrence of a trigger, the reload register contents are reloaded, and the count continues. To generate a trigger while a count is in progress, generate the second trigger after an elapse longer than one cycle of the timer's count source after the previous trigger occurred.

### 2.2.15 Precautions for Timer A (pulse width modulation mode)

- (1) To clear reset, the count start flag is set to "0". Set a value in the timer Ai register, then set the flag to "1".
  
- (2) The timer Ai interrupt request bit becomes "1" if setting operation mode of the timer in compliance with any of the following procedures:
  - Selecting PWM mode after reset.
  - Changing operation mode from timer mode to PWM mode.
  - Changing operation mode from event counter mode to PWM mode.Therefore, to use timer Ai interrupt (interrupt request bit), set timer Ai interrupt request bit to "0" after the above listed changes have been made.
  
- (3) Setting the count start flag to "0" while PWM pulses are being output causes the counter to stop counting. If the TAIOUT pin is outputting an "H" level in this instance, the output level goes to "L", and the timer Ai interrupt request bit goes to "1". If the TAIOUT pin is outputting an "L" level in this instance, the level does not change, and the timer Ai interrupt request is not generated.

## 2.3 Clock-Synchronous Serial I/O

### 2.3.1 Overview

Clock-synchronous serial I/O carries out 8-bit data communications in synchronization with the clock. The following is an overview of the clock-synchronous serial I/O.

#### (1) Transmission/reception format

8-bit data

#### (2) Transfer rate

If the internal clock is selected as the transfer clock, the divide-by-2 frequency, resulting from the bit rate generator division, becomes the transfer rate. The bit rate generator count source can be selected from the following: f<sub>1</sub>, f<sub>8</sub>, and f<sub>32</sub>. Clocks f<sub>1</sub>, f<sub>8</sub>, and f<sub>32</sub> are derived by dividing the CPU's main clock by 1, 8, and 32 respectively.

Furthermore, if an external clock is selected as the transfer clock, the clock frequency input to the CLK pin becomes the transfer rate.

#### (3) Error detection

Only overrun errors can be detected. Overrun error is an error that occurs if the serial interface starts receiving the next data item before reading the contents of the UART<sub>i</sub> receive buffer register and receives the 7th bit of the next data.

#### (4) How to deal with an error

- When receiving data, read an error flag and reception data simultaneously to determine which error has occurred. If the data read is erroneous, initialize the error flag and the UART<sub>i</sub> receive buffer register, then receive the data again.

##### To initialize the UART<sub>i</sub> receive buffer register

1. Set the receive enable bit to "0" (disable reception).
  2. Set the serial I/O mode select bit to "0002" (invalid serial I/O).
  3. Set the serial I/O mode select bit.
  4. Set the receive enable bit to "1" again (enable reception).
- To transmit data again due to an error such as staggered serial clock caused by noise, set the UART<sub>i</sub> transmit buffer register again, then transmit the data again.

##### To set the UART<sub>i</sub> transmit buffer register again

1. Set the serial I/O mode select bits to "0002" (invalidate serial I/O).
2. Set the serial I/O mode select bits again.
3. Set the transmit enable bit to "1" (enable transmission), then set transmission data in the UART<sub>i</sub> transmit buffer register.

#### (5) Function selection

For clock-synchronous serial I/O, the following functions can be selected:

##### (a) CTS/RTS function

In the  $\overline{\text{CTS}}$  function, an external IC can start transmission/reception by inputting an "L" level to the  $\overline{\text{CTS}}$  pin. The  $\overline{\text{CTS}}$  pin input level is detected when transmission/reception starts. Therefore, if the level is set to "H" during transmission/reception, it will stop from the next data.

The  $\overline{\text{RTS}}$  function informs an external IC that  $\overline{\text{RTS}}$  is reception-ready and has changed to "L".  $\overline{\text{RTS}}$  goes back to "H" at the first falling edge of the transfer clock.



The clock-synchronous serial I/O has three types of  $\overline{\text{CTS}}/\overline{\text{RTS}}$  functions to choose from:

- $\overline{\text{CTS}}/\overline{\text{RTS}}$  functions disabled       $\overline{\text{CTS}}/\overline{\text{RTS}}$  pin is a programmable I/O port.
- $\overline{\text{CTS}}$  function only enabled       $\overline{\text{CTS}}/\overline{\text{RTS}}$  pin performs the  $\overline{\text{CTS}}$  function.
- $\overline{\text{RTS}}$  function only enabled       $\overline{\text{CTS}}/\overline{\text{RTS}}$  pin performs the  $\overline{\text{RTS}}$  function.

#### (b) Function for choosing CLK polarity

This function switches the polarity of the transfer clock. The following operations are available:

- Data is input at the falling edge of the transfer clock, and is output at the rising edge.
- Data is input at the rising edge of the transfer clock, and is output at the falling edge.

#### (c) Function for choosing which bit to transmit/receive first

This function is to choose whether to transmit/receive data from bit 0 or from bit 7. Choose either of the following:

- LSB first      Data is transmitted/received from bit 0.
- MSB first      Data is transmitted/received from bit 7.

#### (d) Function for choosing continuous receive mode

Continuous receive mode is a mode in which reading the receive buffer register makes the reception-enabled status ready. In this mode, there is no need to write dummy data to the transmit buffer register so as to make the reception-enabled status ready. But at the time of starting reception, read the receive buffer register into a dummy manner.

- Normal mode      Writing dummy data to the transmit buffer register makes the reception enabled status ready.
- Continuous receive mode      Reading the reception buffer register makes the reception-enabled status ready.

#### (e) Data logic select function

This function is to reverse data when writing to transmit buffer register or reading from receive buffer register.

#### (f) Function for choosing a transmission interrupt factor

The timing to generate a transmission interrupt can be selected from the following: the instant the transmission buffer is emptied or the instant the transmission register is emptied. When transmission buffer empty timing is selected, an interrupt occurs when transmitted data is moved from the transmission buffer to the transmission register. Therefore, data can be transmitted in succession. When transmission register empty timing is selected, an interrupt occurs when data transmission is complete.

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

This function is to reverse a polarity of TxD port output level and a polarity of RxD port input level.

Following are some examples in which various functions (a) through (g) are selected:

- Transmission Operation WITH:  $\overline{\text{CTS}}$  function, transmission at falling edge of transfer clock, LSB First, interrupt at instant transmission buffer is emptied
- Transmission Operation WITH:  $\overline{\text{CTS}}/\overline{\text{RTS}}$  function disabled, transmission at falling edge of transfer clock, LSB First, interrupt at instant transmission is completed
- Reception Operation WITH:  $\overline{\text{RTS}}$  function, reception at falling edge of transfer clock, LSB First, suc-

cessive reception mode disabled

### (6) Input to the serial I/O and the direction register

To input an external signal to the serial I/O, select the function select register A to I/O port and set the direction register to input.

### (7) Pins related to the serial I/O

- $\overline{CTS0}$ ,  $\overline{CTS1}$ ,  $\overline{CTS2}$ ,  $\overline{CTS3}$  pins    Input pins for the  $\overline{CTS}$  function
- $\overline{RTS0}$ ,  $\overline{RTS1}$ ,  $\overline{RTS2}$ ,  $\overline{RTS3}$  pins    Output pins for the  $\overline{RTS}$  function
- CLK0, CLK1, CLK2, CLK3 pins    Input/output pins for the transfer clock
- RxD0, RxD1, RxD2, RxD3 pins    Input pins for data
- TxD0, TxD1, TxD2, TxD3 pins    Output pins for data (Note)

Note: Since TxD2 pin is N-channel open drain, this pin needs pull-up resistor.

### (8) Registers related to the serial I/O

Figure 2.3.1 shows the memory map of serial I/O-related registers, and Figures 2.3.2 to 2.3.4 show serial I/O-related registers.

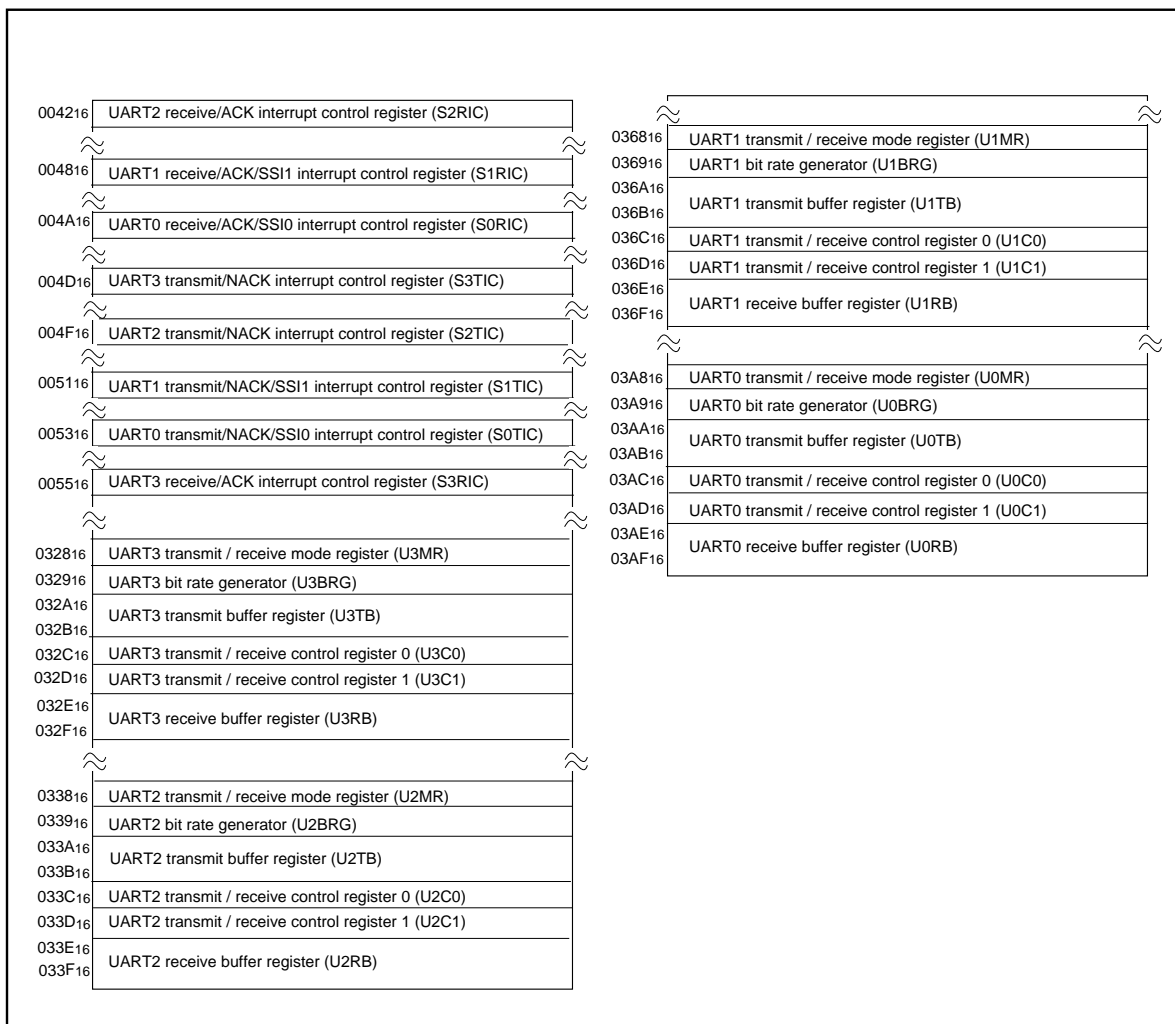


Figure 2.3.1. Memory map of serial I/O-related registers

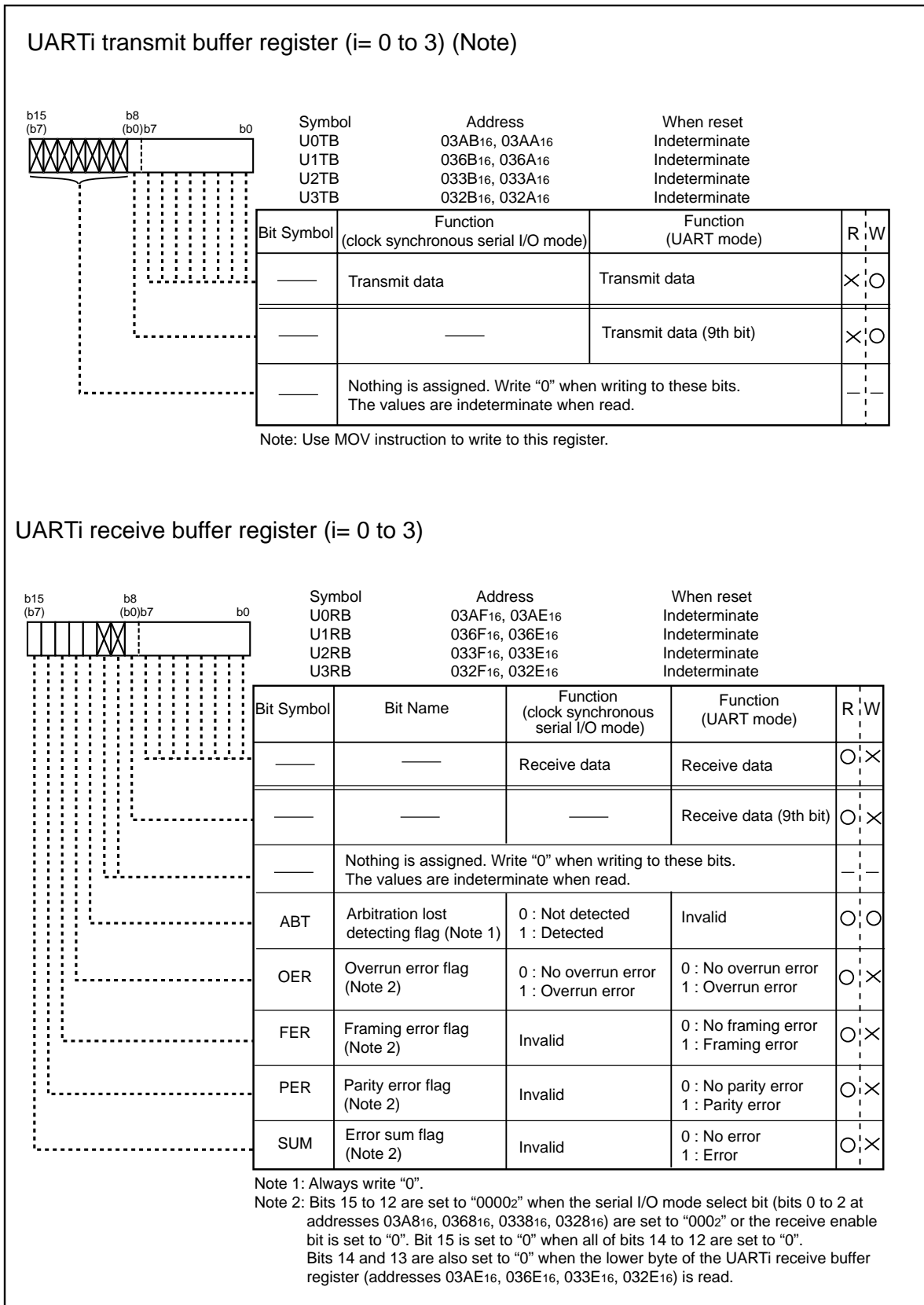


Figure 2.3.2. Serial I/O-related registers (1)

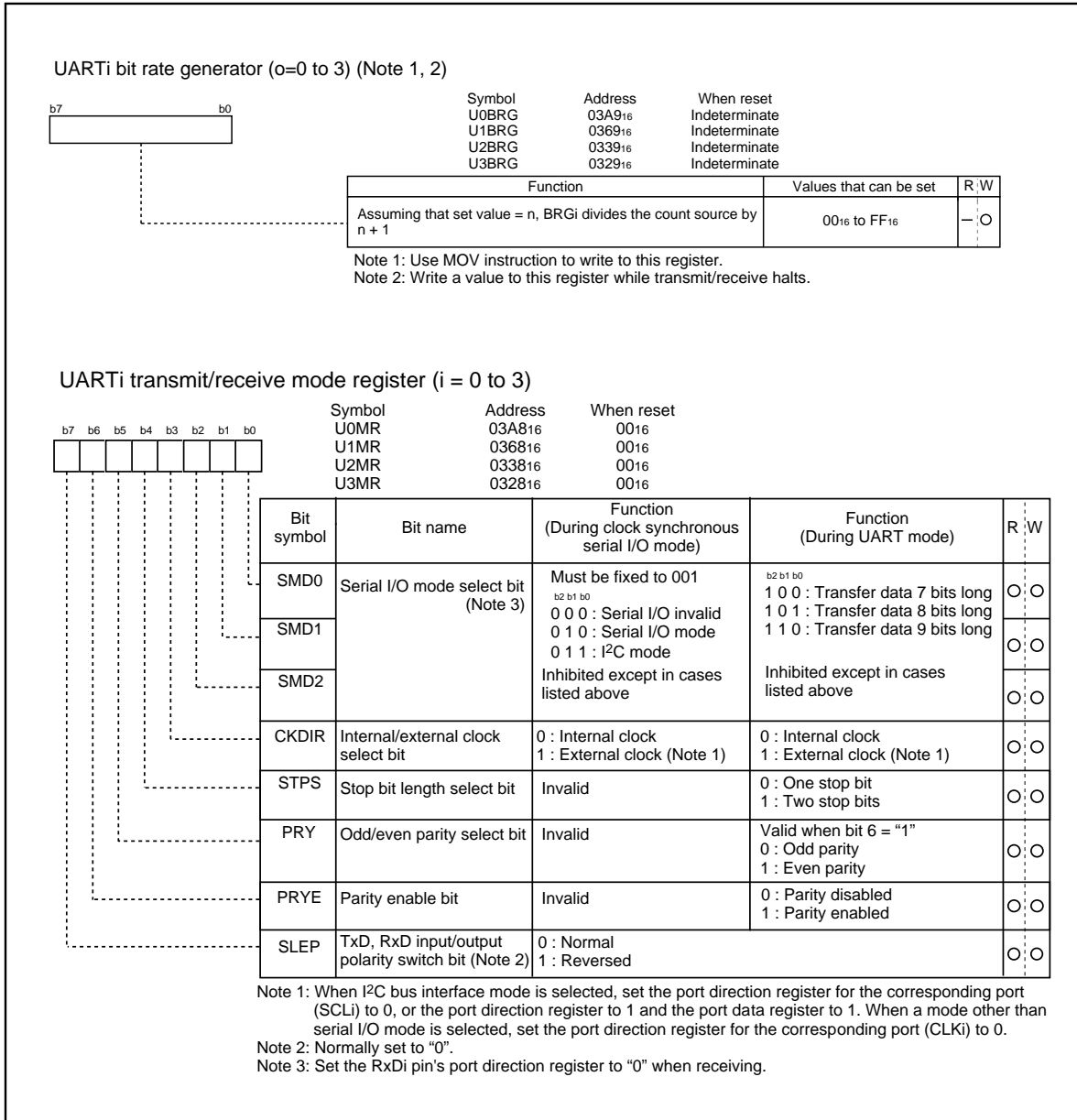


Figure 2.3.3. Serial I/O-related registers (2)

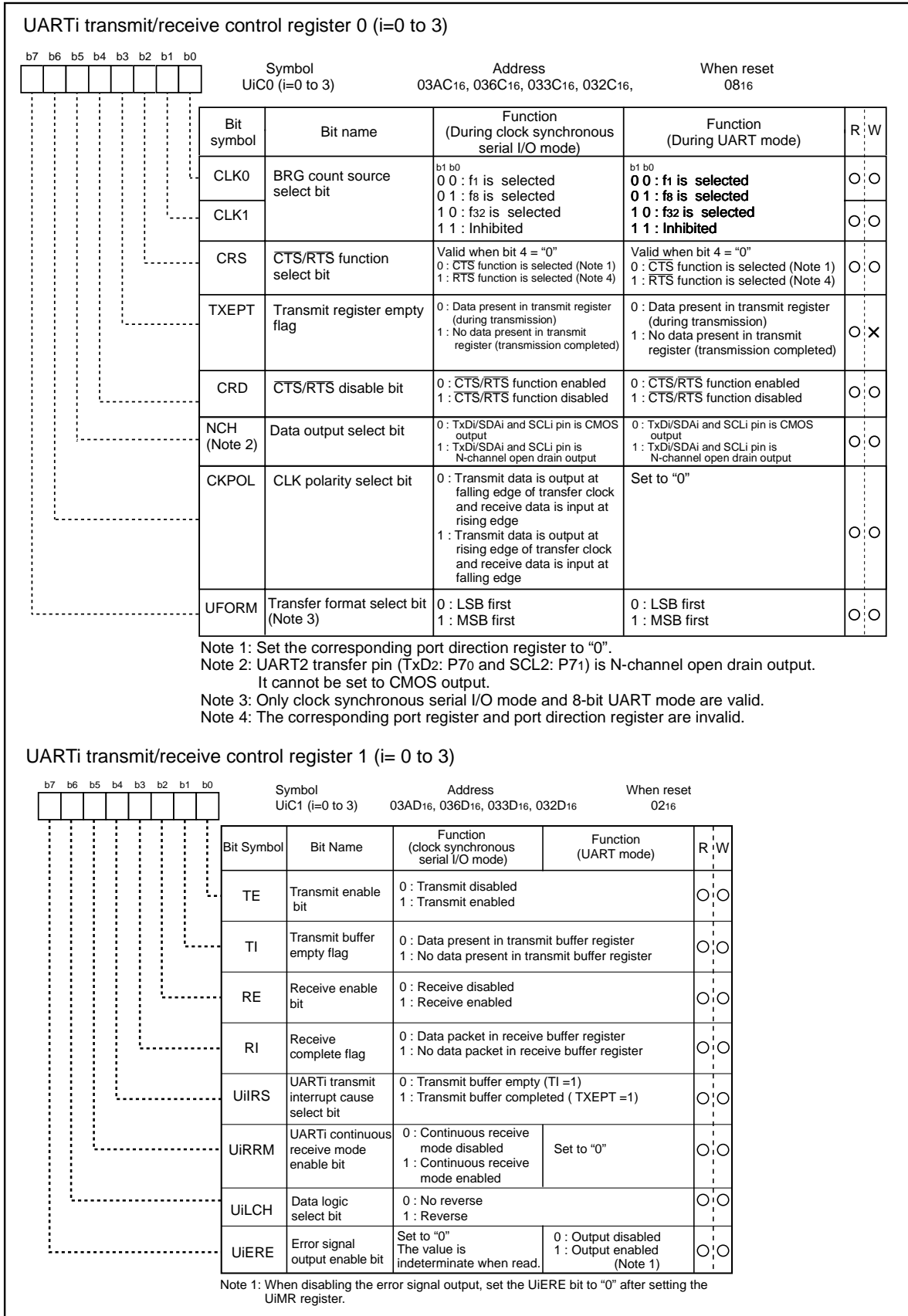


Figure 2.3.4. Serial I/O-related registers (3)

### 2.3.2 Operation of Serial I/O (transmission in clock-synchronous serial I/O mode)

In transmitting data in clock-synchronous serial I/O mode, choose functions from those listed in Table 2.3.1. Operations of the circled items are described below. Figure 2.3.5 shows the operation timing, and Figures 2.3.6 and 2.3.7 show the set-up procedures.

**Table 2.3.1. Chosed functions**

Item	Set-up		Item	Set-up	
Transfer clock source	<input type="radio"/>	Internal clock ( $f_1 / f_8 / f_{32}$ )	Transfer clock	<input type="radio"/>	LSB first
		External clock (CLKi pin)			MSB first
$\overline{\text{CTS}}$ function	<input type="radio"/>	$\overline{\text{CTS}}$ function enabled	Transmission interrupt factor	<input type="radio"/>	Transmission buffer empty
		$\overline{\text{CTS}}$ function disabled			Transmission complete
CLK polarity	<input type="radio"/>	Output transmission data at the falling edge of the transfer clock	Data logic select function	<input type="radio"/>	No reverse
		Output transmission data at the rising edge of the transfer clock			Reverse
			TxD, RxD I/O polarity reverse bit	<input type="radio"/>	No reverse
					Reverse

- Operation
- (1) Setting the transmit enable bit to "1" and writing transmission data to the UARTi transmit buffer register makes data transmissible status ready.
  - (2) When input to the  $\overline{\text{CTS}}_i$  pin goes to "L" level, transmission starts (the  $\overline{\text{CTS}}_i$  pin must be controlled on the reception side).
  - (3) In synchronization with the first falling edge of the transfer clock, transmission data held in the UARTi transmit buffer register is transmitted to the UARTi transmit register. At this time, the UARTi transmit interrupt request bit goes to "1". Also, the first bit of the transmission data is transmitted from the TxDi pin. Then the data is transmitted bit by bit from the lower order in synchronization with the falling edges.
  - (4) When transmission of 1-byte data is completed, the transmit register empty flag goes to "1", which indicates that transmission is completed. The transfer clock stops at "H" level.
  - (5) If the next transmission data is set in the UARTi transmit buffer register while transmission is in progress (before the eighth bit has been transmitted), the data is transmitted in succession.

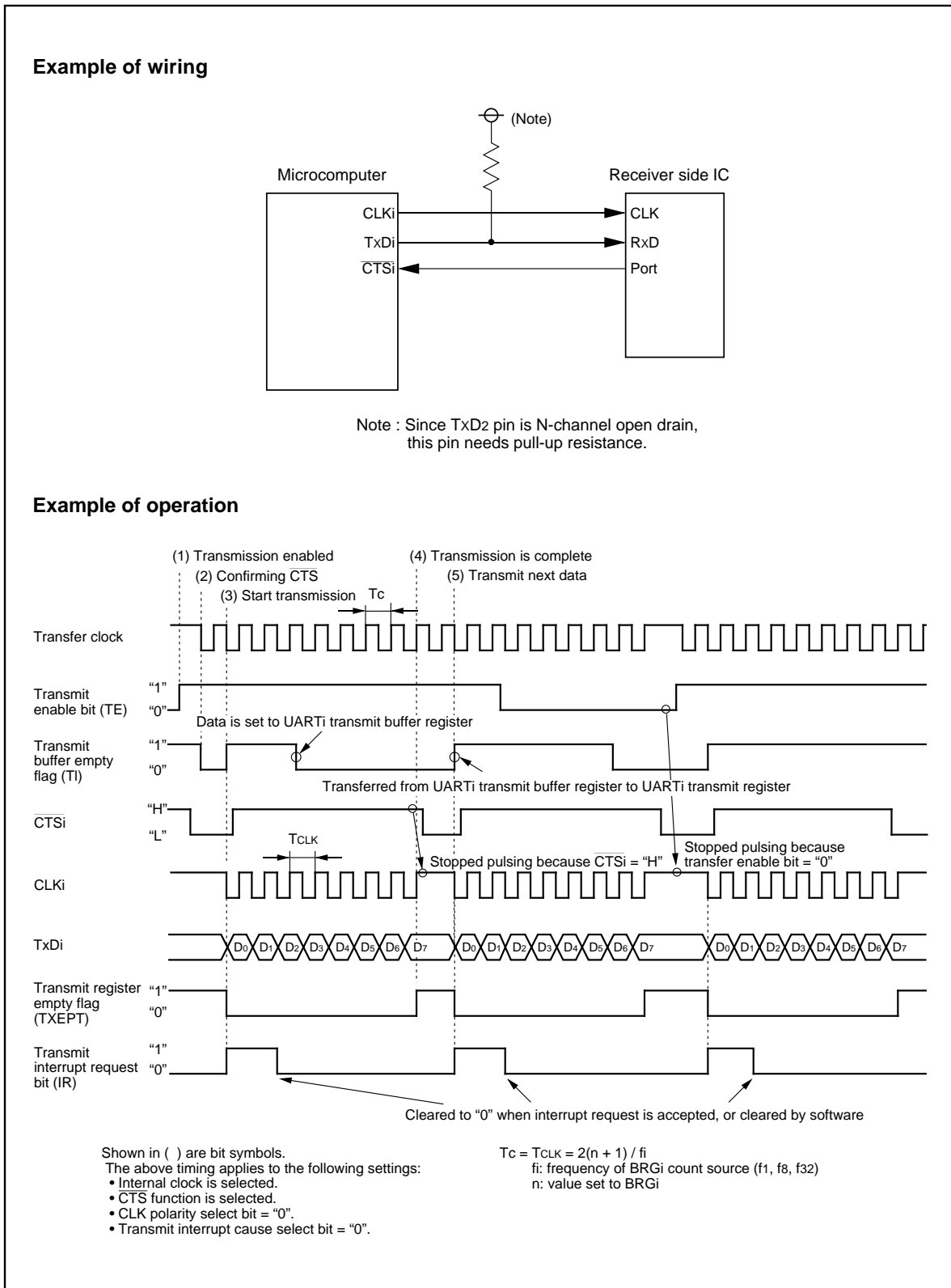


Figure 2.3.5. Operation timing of transmission in clock-synchronous serial I/O mode

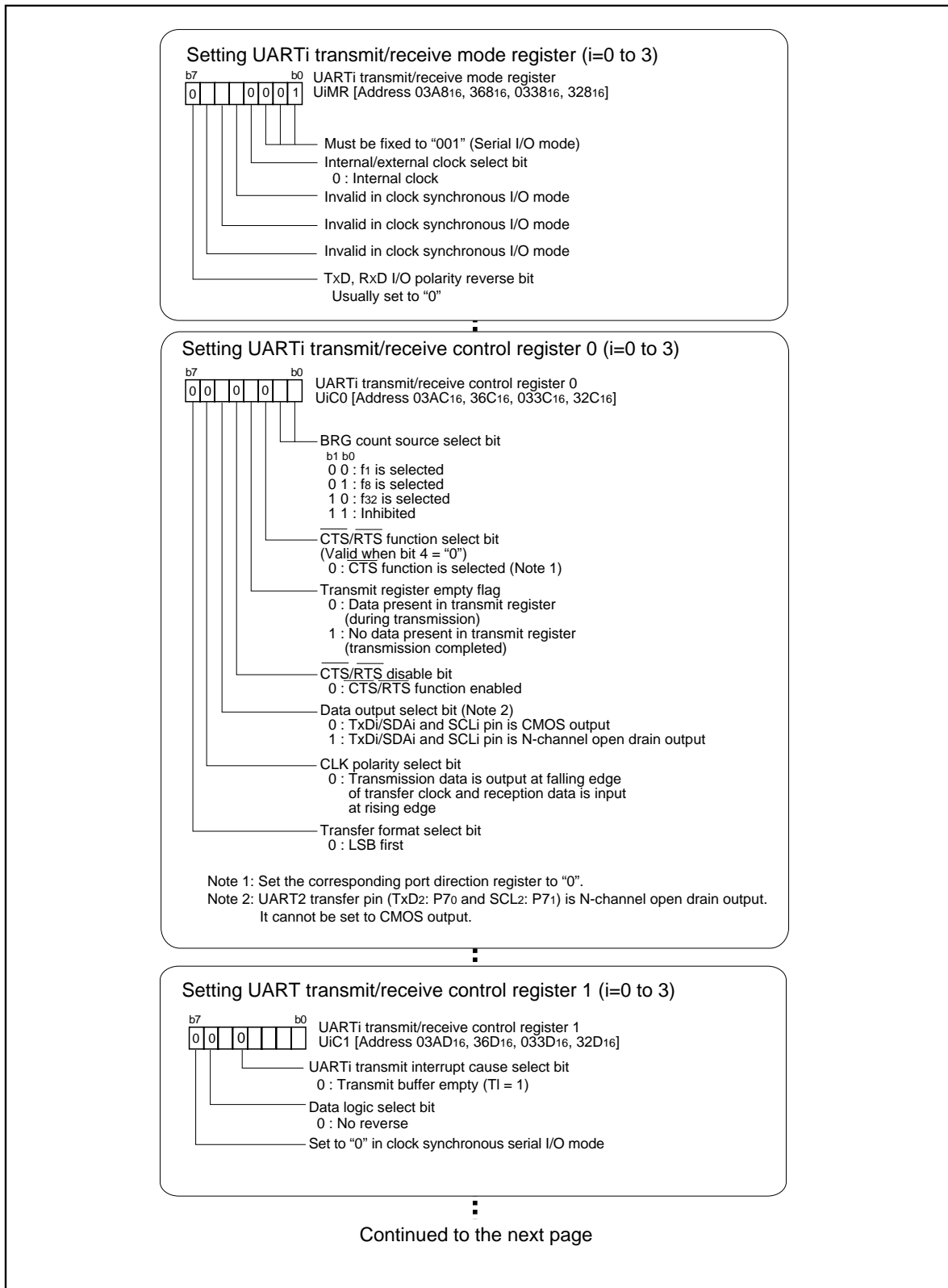


Figure 2.3.6. Set-up procedure of transmission in clock-synchronous serial I/O mode (1)



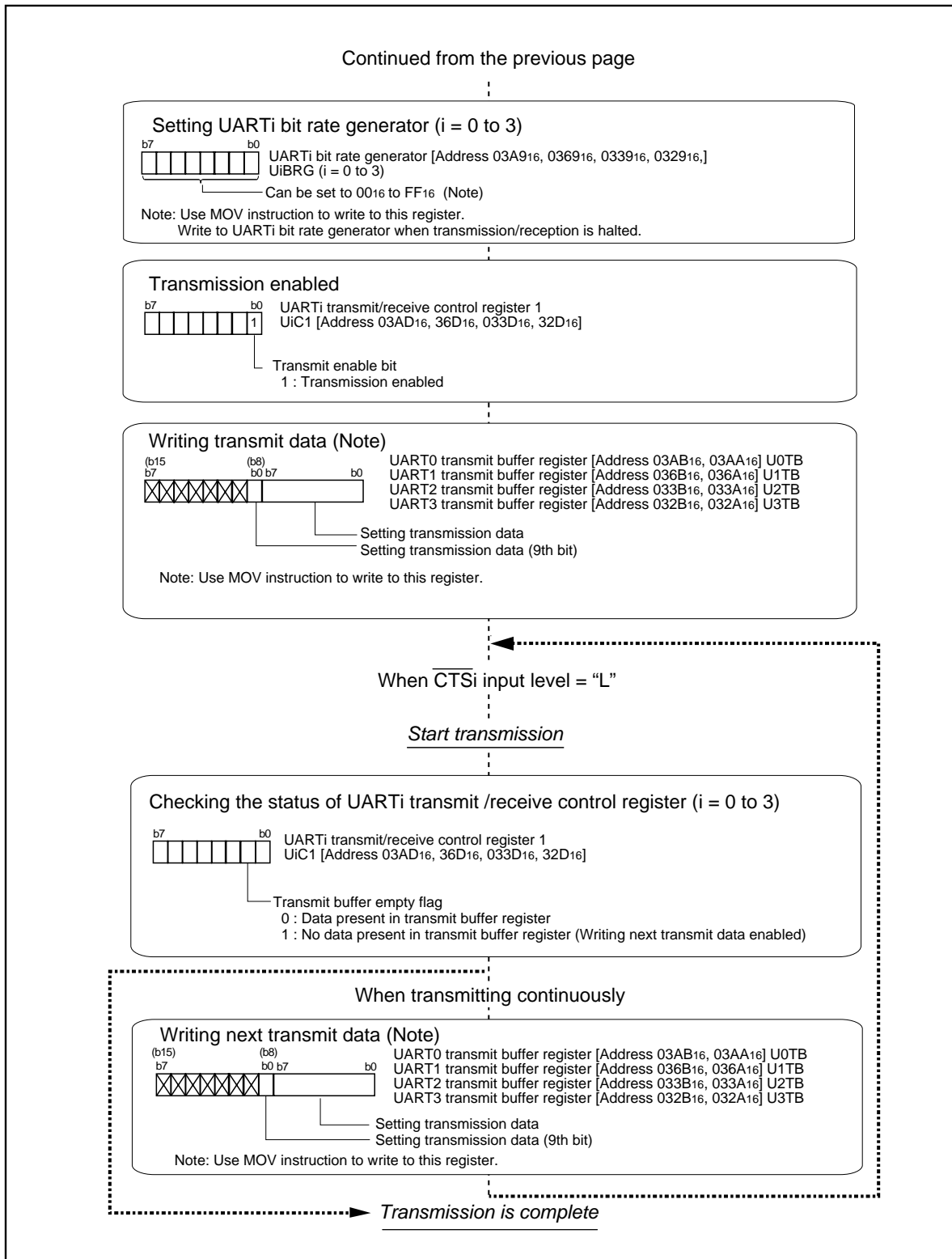


Figure 2.3.7. Set-up procedure of transmission in clock-synchronous serial I/O mode (2)

### 2.3.3 Operation of Serial I/O (reception in clock-synchronous serial I/O mode)

In receiving data in clock-synchronous serial I/O mode, choose functions from those listed in Table 2.3.2. Operations of the circled items are described below. Figure 2.3.8 shows the operation timing, and Figures 2.3.9 and 2.3.10 show the set-up procedures.

**Table 2.3.2. Chosed functions**

Item	Set-up		Item	Set-up	
Transfer clock source	<input type="radio"/>	Internal clock (f <sub>1</sub> / f <sub>8</sub> / f <sub>32</sub> )	Transfer clock	<input type="radio"/>	LSB first
		External clock (CLKi pin)			MSB first
RTS function	<input type="radio"/>	RTS function enabled	Continuous receive mode	<input type="radio"/>	Disabled
		RTS function disabled			Enabled
CLK polarity	<input type="radio"/>	Output transmission data at the falling edge of the transfer clock	Data logic select function	<input type="radio"/>	No reverse
		Output transmission data at the rising edge of the transfer clock			Reverse
			Tx/D, Rx/D I/O polarity reverse bit	<input type="radio"/>	No reverse
					Reverse

- Operation
- (1) Writing dummy data to the UARTi transmit buffer register, setting the receive enable bit to “1”, and the transmit enable bit to “1”, makes the data receivable status ready. At this time, the output from the  $\overline{\text{RTSi}}$  pin goes to “L” level, which informs the transmission side that the data receivable status is ready (output the transfer clock from the IC on the transmission side after checking that the  $\overline{\text{RTS}}$  output has gone to “L” level).
  - (2) In synchronization with the first rising edge of the transfer clock, the input signal to the RxDi pin is stored in the highest bit of the UARTi receive register. Then, data is taken in by shifting right the content of the UARTi reception data in synchronization with the rising edges of the transfer clock.
  - (3) When 1-byte data lines up in the UARTi receive register, the content of the UARTi receive register is transmitted to the UARTi receive buffer register. The transfer clock stops at “H” level. At this time, the receive complete flag and the UARTi receive interrupt request bit goes to “1”.
  - (4) The receive complete flag goes to “0” when the lower-order byte of the UARTi buffer register is read.

Note

- Set CLKi and RxDi pins' port direction register to “0”.

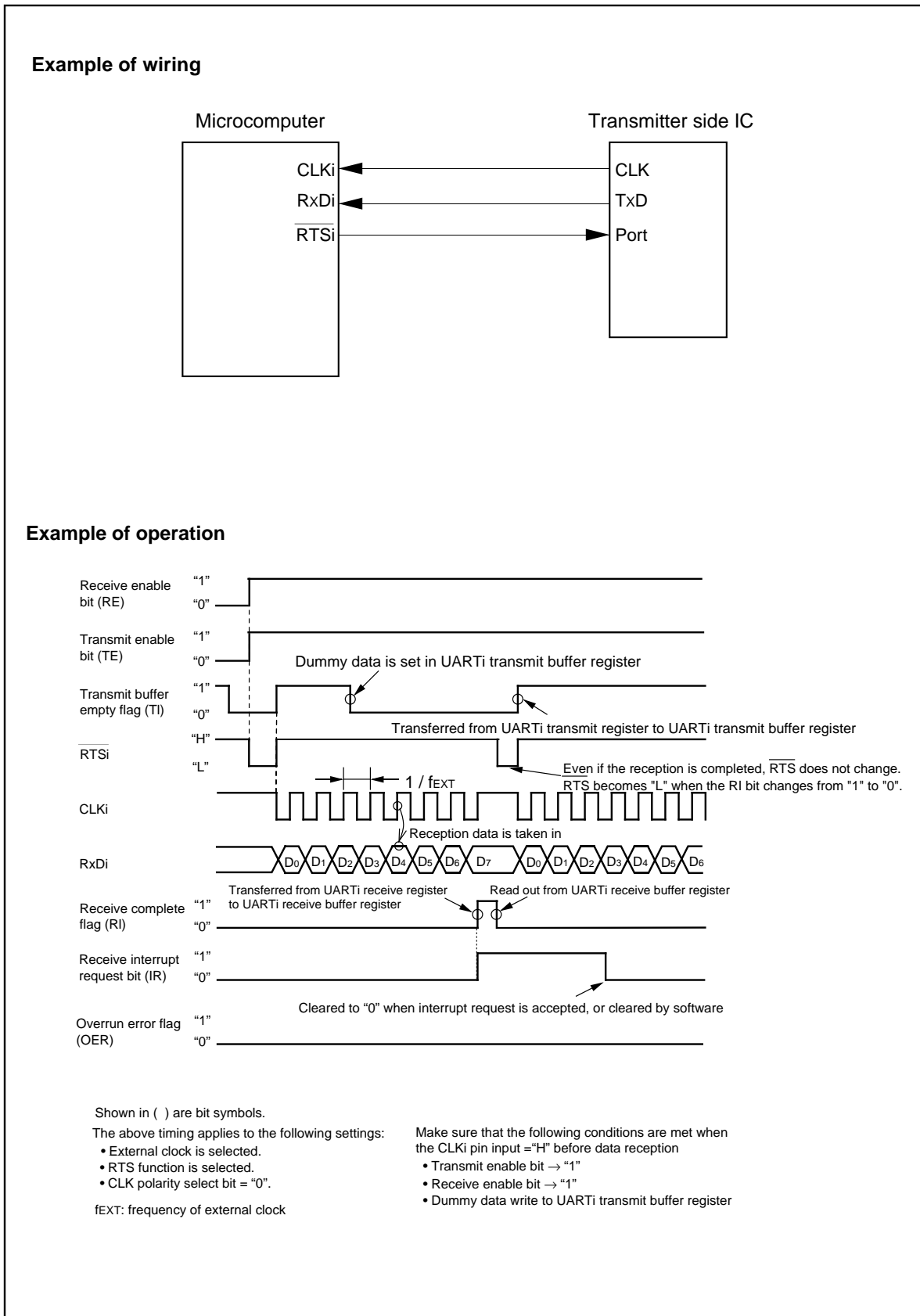


Figure 2.3.8. Operation timing of reception in clock-synchronous serial I/O mode

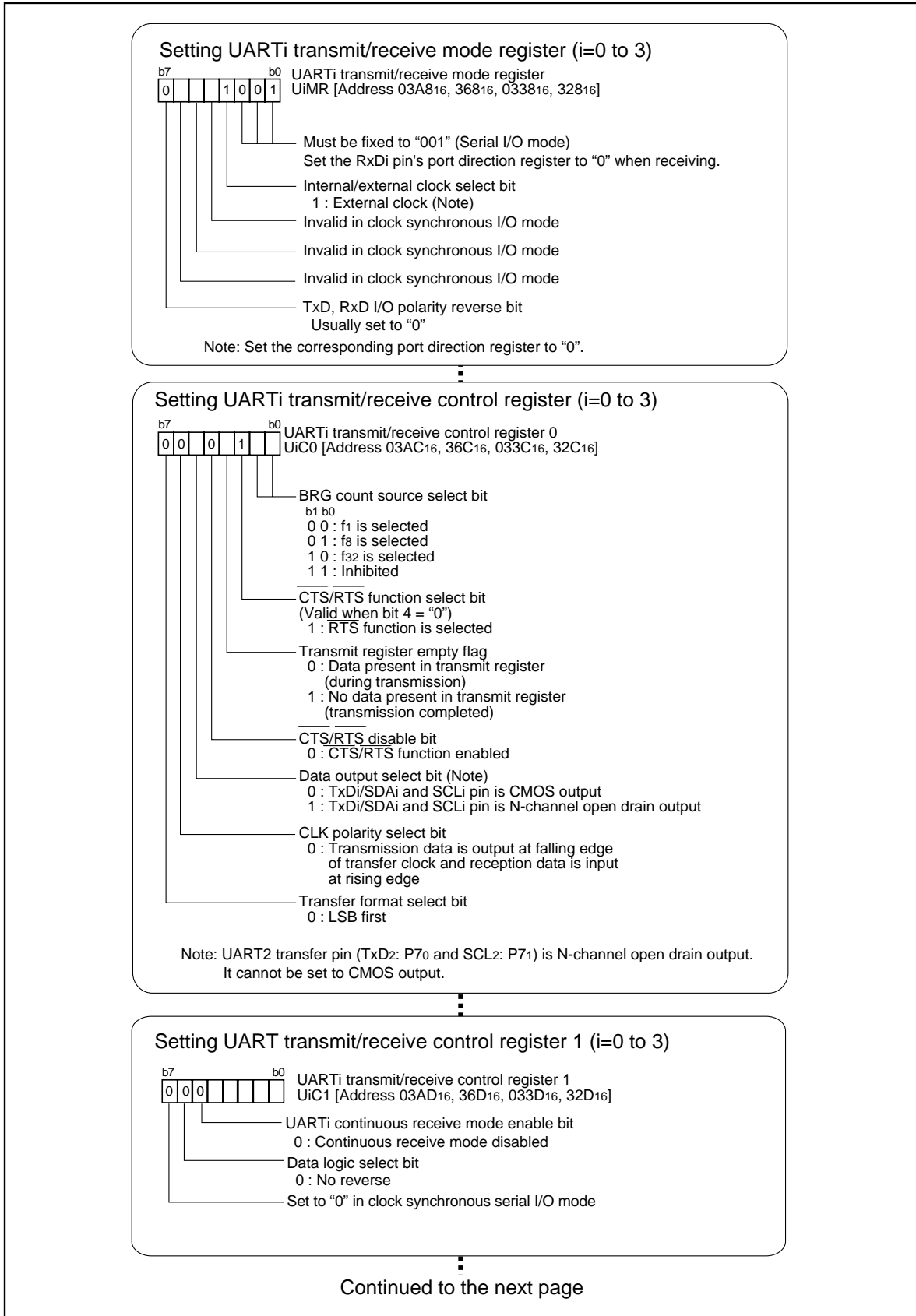


Figure 2.3.9. Set-up procedure of reception in clock-synchronous serial I/O mode (1)

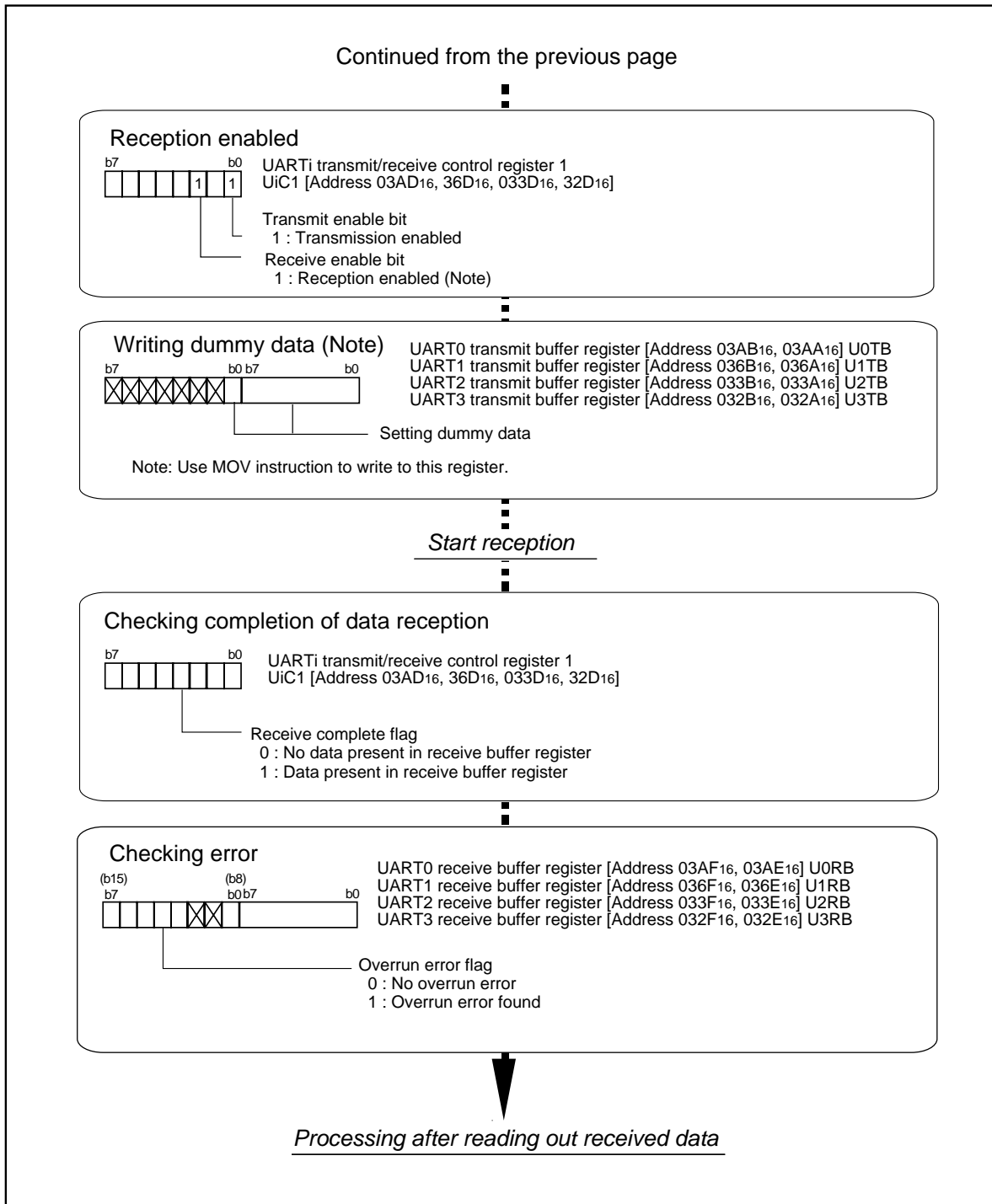


Figure 2.3.10. Set-up procedure of reception in clock-synchronous serial I/O mode (2)

### 2.3.4 Precautions for Serial I/O (in clock-synchronous serial I/O mode)

#### Transmission/reception

- (1) With an external clock selected, and choosing the  $\overline{\text{RTSi}}$  function, the output level of the  $\overline{\text{RTSi}}$  pin goes to "L" when the data-receivable status becomes ready, which informs the transmission side that the reception has become ready. The output level of the  $\overline{\text{RTSi}}$  pin goes to "H" when reception starts. So if the  $\overline{\text{RTSi}}$  pin is connected to the  $\overline{\text{CTS}_i}$  pin on the transmission side, the circuit can transmission and reception data with consistent timing. With the internal clock, the  $\overline{\text{RTS}}$  function has no effect. Figure 2.3.11 shows an example of wiring.

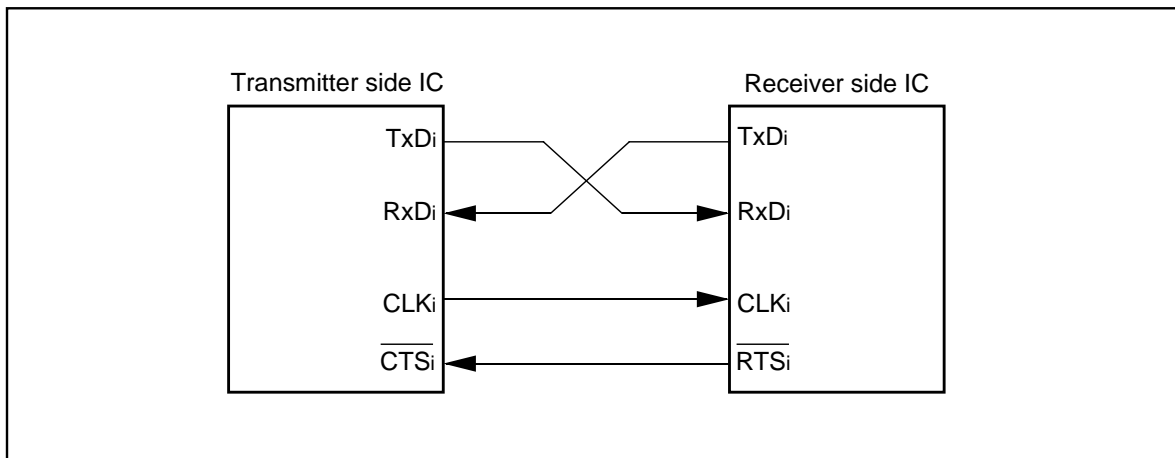


Figure 2.3.11. Example of wiring

### Transmission

- (1) With an external clock selected, perform the following set-up procedure with the CLKi pin input level = "H" if the CLK polarity select bit = "0" or with the CLKi pin input level = "L" if the CLK polarity select bit = "1":
  1. Set the transmit enable bit (to "1")
  2. Write transmission data to the UARTi transmit buffer register
  3. "L" level input to the  $\overline{\text{CTS}}_i$  pin (when the  $\overline{\text{CTS}}$  function is selected)

Reception (1) In operating the clock-synchronous serial I/O, operating a transmitter generates a shift clock. Fix settings for transmission even when using the device only for reception. Dummy data is output to the outside from the TxDi pin (transmission pin) when receiving data.

- (2) With the internal clock selected, setting the transmit enable bit to "1" (transmission-enabled status) and setting dummy data in the UARTi transmission buffer register generates a shift clock.

With the external clock selected, a shift clock is generated when the transmit enable bit is set to "1", dummy data is set in the UARTi transmit buffer register, and the external clock is input to the CLKi pin.

- (3) When receiving data in succession, an overrun error occurs if the serial interface starts receiving the next data item while the receive complete flag is 1 (before reading the contents of the UARTi receive buffer register) and receives the 7th bit of the next data item, and then the overrun error flag is set to "1". In this instance, the next data is written to the UARTi receive buffer register, so handle with this problem by writing programs on transmission side and reception side so that the previous data is transmitted again.

If an overrun error occurs, the UARTi receive interrupt request bit does not go to "1".

- (4) To receive data in succession, set dummy data in the lower-order byte of the UARTi transmit buffer register every time reception is made. In continuous receive mode, when the receive buffer is read out, the unit simultaneously goes to a receive enable state without having to set dummy data back to the transmit buffer register again.

- (5) With an external clock selected, perform the following set-up procedure with the CLKi pin input level = "H" if the CLK polarity select bit = "0" or with the CLKi pin input level = "L" if the CLK polarity select bit = "1":

1. Set receive enable bit (to "1")
2. Set transmit enable bit (to "1")
3. Write dummy data to the UARTi transmit buffer register

- (6) Output from the  $\overline{\text{RTS}}$  pin goes to "L" level as soon as the receive enable bit is set to "1". This is not related to the content of the transmit buffer empty flag or the content of the transmit enable bit. Output from the RTS pin goes to "H" level when reception starts, and goes to "L" level when reception is completed. This is not related to the content of the transmit buffer empty flag or the content of the receive complete flag.

## 2.4 Clock-Asynchronous Serial I/O (UART)

### 2.4.1 Overview

UART handles communications by means of character-by-character synchronization. The transmission side and the reception side are independent of each other, so full-duplex communication is possible. The following is an overview of the clock-asynchronous serial I/O.

#### (1) Transmission/reception format

Figure 2.4.1 shows the transmission/reception format, and Table 2.4.1 shows the names and functions of transmission data.

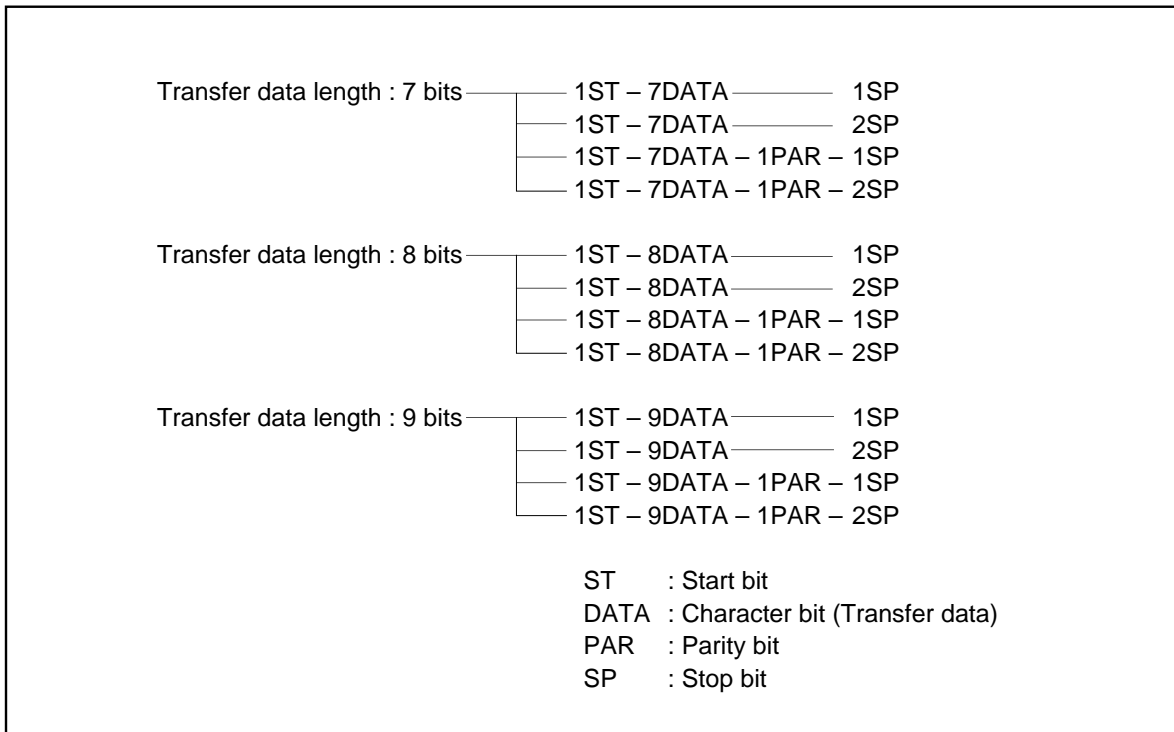


Figure 2.4.1. Transmission/reception format

Table 2.4.1. Transmission data names and functions

Name	Function
ST (start bit)	A 1-bit "L" signal to be added immediately before character bits. This bit signals the start of data transmission.
DATA (character bits)	Transmission data set in the UARTi transmit buffer register.
PAR (parity bit)	A signal to be added immediately after character bits so as to increase data reliability. The level of this signal so varies that the total number of 1's in character bits and this bit always becomes even or odd depending on which parity is chosen, even or odd.
SP (stop bit)	Either 1-bit or 2-bit "H" signal to be added immediately after character bits (after the parity bit if parity is checked). This / they signals the end of data transmission.



**(2) Transfer rate**

The divide-by-16 frequency, resulting from division in the bit rate generator (BRG), becomes the transfer rate. The count source for the transfer rate register can be selected from f<sub>1</sub>, f<sub>8</sub>, f<sub>32</sub>, and the input from the CLK pin. Clocks f<sub>1</sub>, f<sub>8</sub>, f<sub>32</sub> are derived by dividing the CPU's main clock by 1, 8, and 32 respectively.

**Table 2.4.2. Example of baud rate setting**

Baud rate (bps)	BRG's count source	System clock : 16MHz		System clock : 7.3728MHz	
		BRG's set value : n	Actual time (bps)	BRG's set value : n	Actual time (bps)
600	f <sub>8</sub>	207 (CF <sub>16</sub> )	601	95 (5F <sub>16</sub> )	600
1200	f <sub>8</sub>	103 (67 <sub>16</sub> )	1202	47 (2F <sub>16</sub> )	1200
2400	f <sub>8</sub>	51 (33 <sub>16</sub> )	2404	23 (17 <sub>16</sub> )	2400
4800	f <sub>1</sub>	207 (CF <sub>16</sub> )	4808	95 (5F <sub>16</sub> )	4800
9600	f <sub>1</sub>	103 (67 <sub>16</sub> )	9615	47 (2F <sub>16</sub> )	9600
14400	f <sub>1</sub>	68 (44 <sub>16</sub> )	14493	31 (1F <sub>16</sub> )	14400
19200	f <sub>1</sub>	51 (33 <sub>16</sub> )	19231	23 (17 <sub>16</sub> )	19200
28800	f <sub>1</sub>	34 (22 <sub>16</sub> )	28571	15 (F <sub>16</sub> )	28800
31250	f <sub>1</sub>	33 (21 <sub>16</sub> )	31250		

**(3) An error detection**

In UART mode, detected errors are shown in Table 2.4.3.

**Table 2.4.3. Error detection**

Type of error	Description	When the flag turns on	How to clear the flag
Overrun error	<ul style="list-style-type: none"> <li>This error occurs when the serial interface starts receiving the next data item before reading the contents of the UART<sub>i</sub> receive buffer register and receives the bit preceding the final stop bit of the next data item.</li> <li>The contents of the UART<sub>i</sub> receive buffer register are undefined.</li> <li>The UART<sub>i</sub> receive interrupt request bit does not go to "1".</li> </ul>	The error is detected when data is transferred from the UART <sub>i</sub> receive register to the UART <sub>i</sub> receive buffer register.	<ul style="list-style-type: none"> <li>Set the serial I/O mode select bits to "0002".</li> <li>Set the receive enable bit to "0".</li> </ul>
Framing error	<ul style="list-style-type: none"> <li>This error occurs when the stop bit falls short of the set number of stop bits.</li> </ul>		<ul style="list-style-type: none"> <li>Set the serial I/O mode select bits to "0002".</li> <li>Set the receive enable bit to "0".</li> </ul>
Parity error	<ul style="list-style-type: none"> <li>With parity enabled, this error occurs when the total number of 1's in character bits and the parity bit is different from the specified number.</li> </ul>		<ul style="list-style-type: none"> <li>Read the lower-order byte of the UART<sub>i</sub> receive buffer register.</li> </ul>
Error-sum flag	<ul style="list-style-type: none"> <li>This flag turns on when any error (overrun, framing, or parity) is detected.</li> </ul>		<ul style="list-style-type: none"> <li>When all error (overrun, framing, and parity) are removed, the flag is cleared.</li> </ul>

#### (4) How to deal with an error

When receiving data, read an error flag and reception data simultaneously to determine which error has occurred. If the data read is erroneous, initialize the error flag and the UARTi receive buffer register, then receive the data again.

##### To initialize the UARTi receive buffer register

1. Set the receive enable bit to "0" (disable reception).
2. Set the receive enable bit to "1" again (enable reception).

To transmit data again due to an error on the reception side, set the UARTi transmit buffer register again, then transmit the data again.

##### To set the UARTi transmit buffer register again

1. Set the serial I/O mode select bits to "0002" (invalidate serial I/O).
2. Set the serial I/O mode select bits again.
3. Set the transmit enable bit to "1" (enable transmission), then set transmission data in the UARTi transmit buffer register.

#### (5) Functions selection

In operating UART, the following functions can be used:

##### (a) CTS/RTS function

$\overline{\text{CTS}}$  function is a function in which an external IC can start transmission/reception by means of inputting an "L" level to the  $\overline{\text{CTS}}$  pin. The  $\overline{\text{CTS}}$  pin input level is detected when transmission/reception starts, so if the level is gone to "H" while transmission/reception is in progress, transmission/reception stops at the next data.

$\overline{\text{RTS}}$  function is a function to inform an external IC that  $\overline{\text{RTS}}$  pin output level has changed to "L" when reception is ready.  $\overline{\text{RTS}}$  regoes to "H" at the first falling edge of the transfer clock.

When using clock-asynchronous serial I/O, choose one of three types of  $\overline{\text{CTS}}/\overline{\text{RTS}}$  functions.

- $\overline{\text{CTS}}/\overline{\text{RTS}}$  functions disabled       $\overline{\text{CTS}}/\overline{\text{RTS}}$  pin is a programmable I/O port.
- $\overline{\text{CTS}}$  function only enabled       $\overline{\text{CTS}}/\overline{\text{RTS}}$  pin performs the  $\overline{\text{CTS}}$  function.
- $\overline{\text{RTS}}$  function only enabled       $\overline{\text{CTS}}/\overline{\text{RTS}}$  pin performs the  $\overline{\text{RTS}}$  function.

##### (b) Data logic select function

This function is to reverse data when writing to transmit buffer register or reading from receive buffer register.

##### (c) LSB/MSB first select function

This function is to choose whether to transmit/receive data from bit 0 or bit 7. This is valid when the transfer data length is 8 bits long.

Choose either of the following:

- LSB first    Data is transmitted/received from bit 0.
- MSB first    Data is transmitted/received from bit 7.

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

This function is to reverse a polarity of TxD port output level and a polarity of RxD port input level.

**(e) Bus collision detection function**

This function is to sample the output level of the TxD pin and the input level of the RxD pin; if their values are different, then an interrupt request occurs.

The following examples are described in section 2.4.2 and 2.4.3.

- Transmission WITH:  $\overline{\text{CTS}}$  function, WITHOUT: other functions
- Reception WITH:  $\overline{\text{RTS}}$  function, WITHOUT: other functions

Also, the SIM interface is used by adding some extra settings in clock-asynchronous serial I/O mode.

Direct or inverse format is selected by connecting SIM card.

The following examples are described in section 2.4.4 and 2.4.5.

- Transmission WITH: direct format
- Reception WITH: direct format

**(6) Input to the serial I/O and the direction register**

To input an external signal to the serial I/O, set the direction register of the relevant port to input.

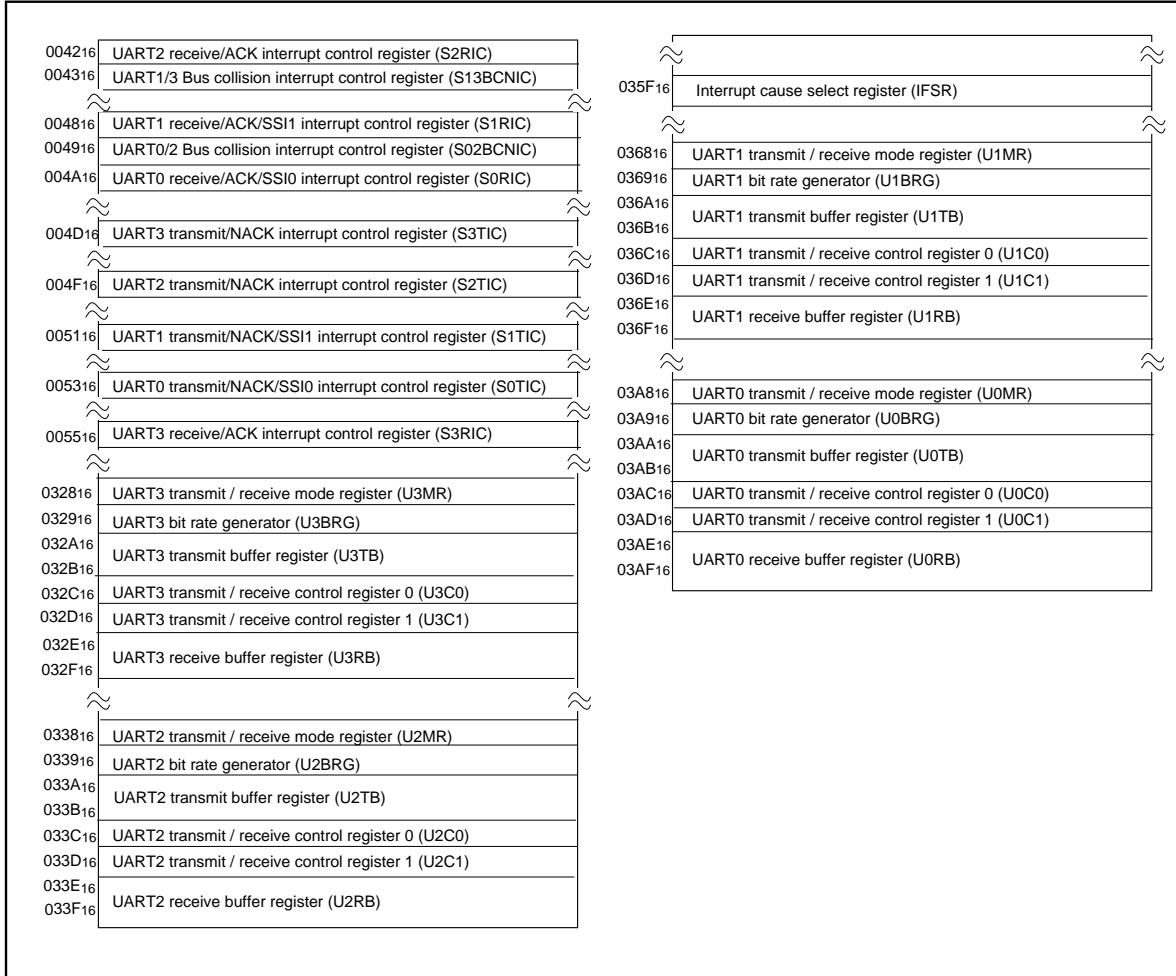
**(7) Pins related to the serial I/O**

- $\overline{\text{CTS}}_0, \overline{\text{CTS}}_1, \overline{\text{CTS}}_2, \overline{\text{CTS}}_3$  pins :Input pins for the  $\overline{\text{CTS}}$  function
- $\overline{\text{RTS}}_0, \overline{\text{RTS}}_1, \overline{\text{RTS}}_2, \overline{\text{RTS}}_3$  pins :Output pins for the  $\overline{\text{RTS}}$  function
- $\text{CLK}_0, \text{CLK}_1, \text{CLK}_2, \text{CLK}_3$  pins :Input pins for the transfer clock
- $\text{RxD}_0, \text{RxD}_1, \text{RxD}_2, \text{RxD}_3$  pins :Input pins for data
- $\text{TxD}_0, \text{TxD}_1, \text{TxD}_2, \text{TxD}_3$  pins :Output pins for data (Note)

Note: Since TxD2 pin is N-channel open drain, this pin needs pull-up resistor.

**(8) Registers related to the serial I/O**

Figure 2.4.2 shows the memory map of serial I/O-related registers, and Figures 2.4.3 to 2.4.6 show UARTi-related registers.



**Figure 2.4.2. Memory map of UARTi-related registers**

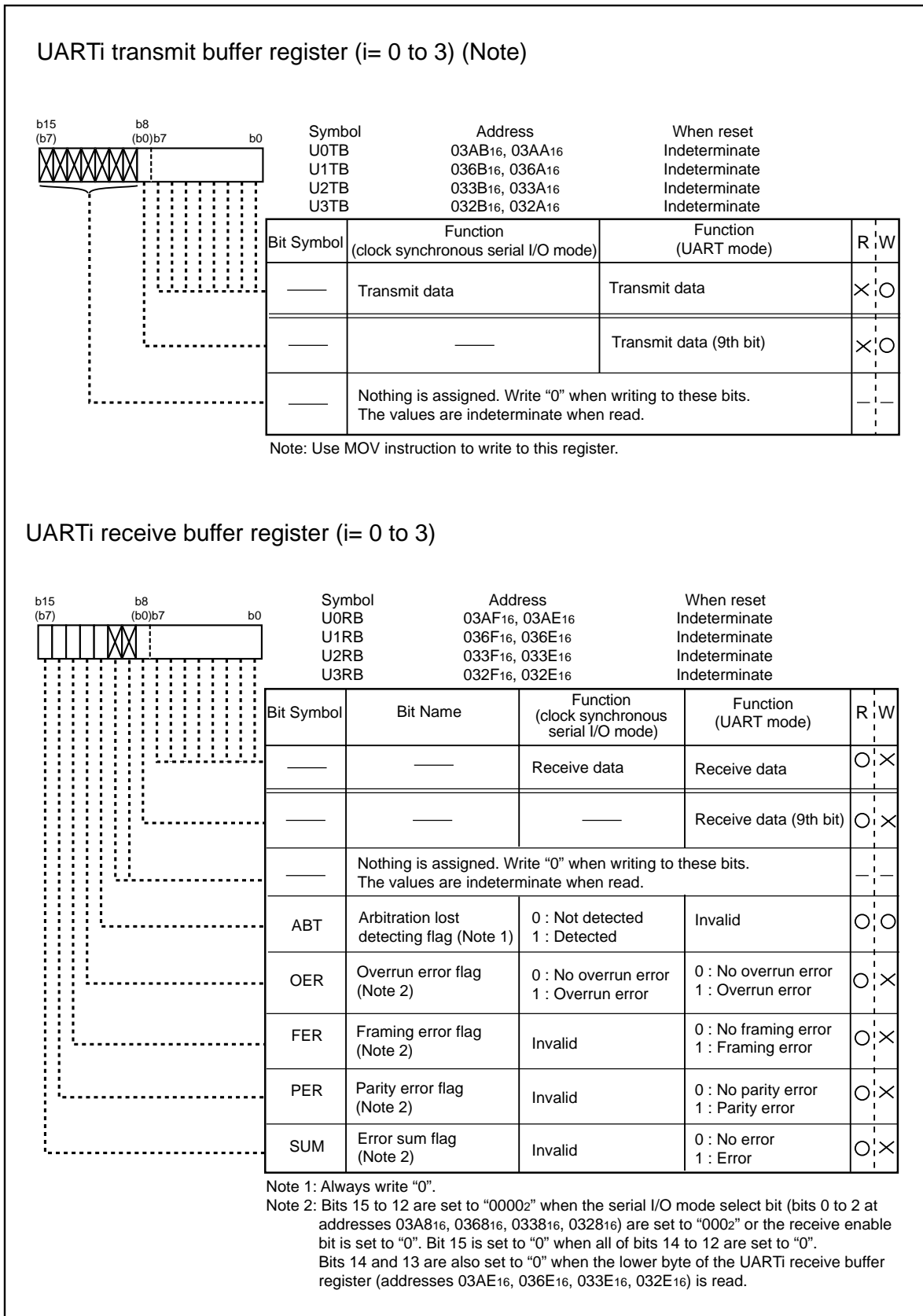


Figure 2.4.3. UART<sub>i</sub>-related registers (1)

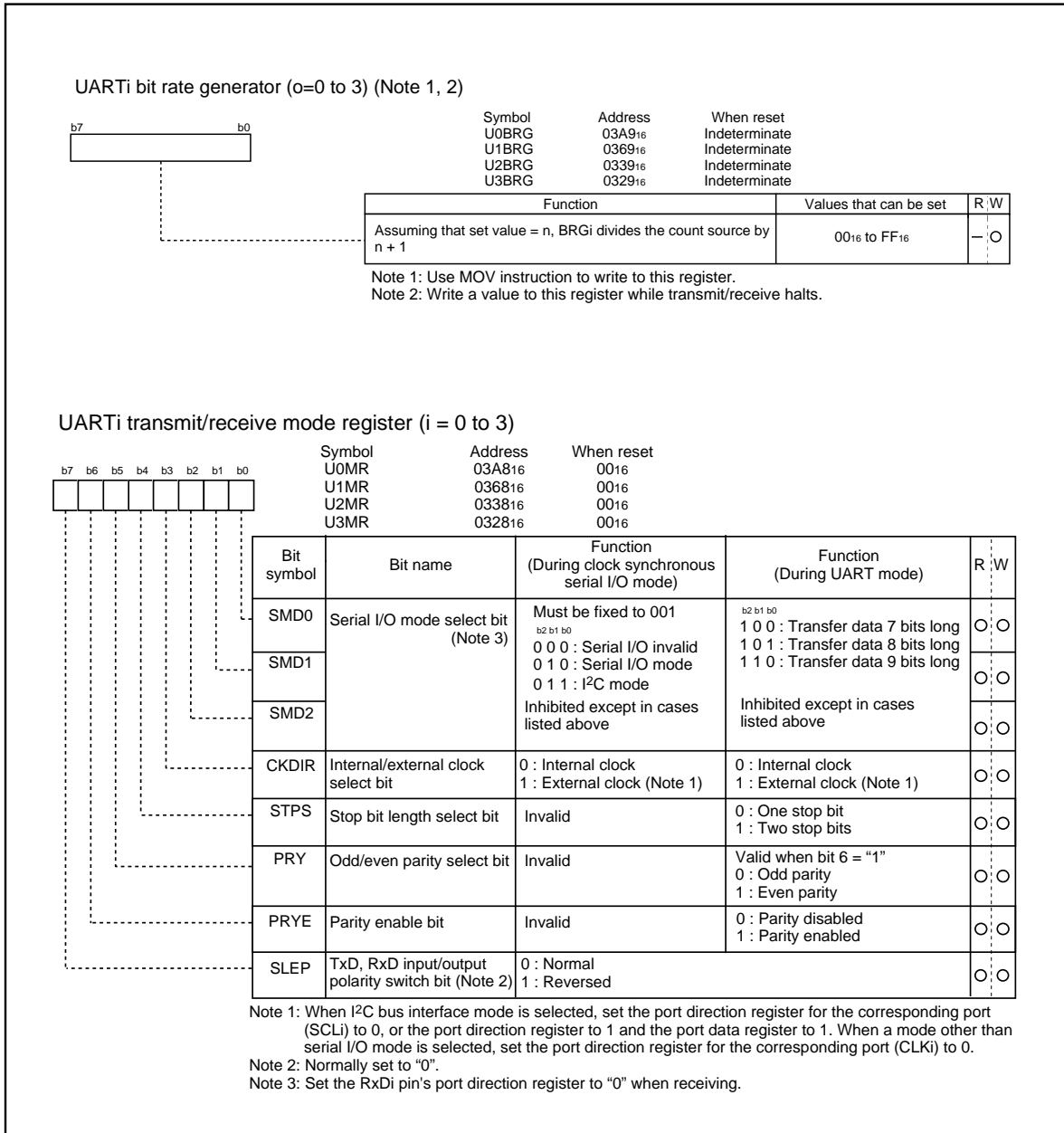


Figure 2.4.4. UARTi-related registers (2)

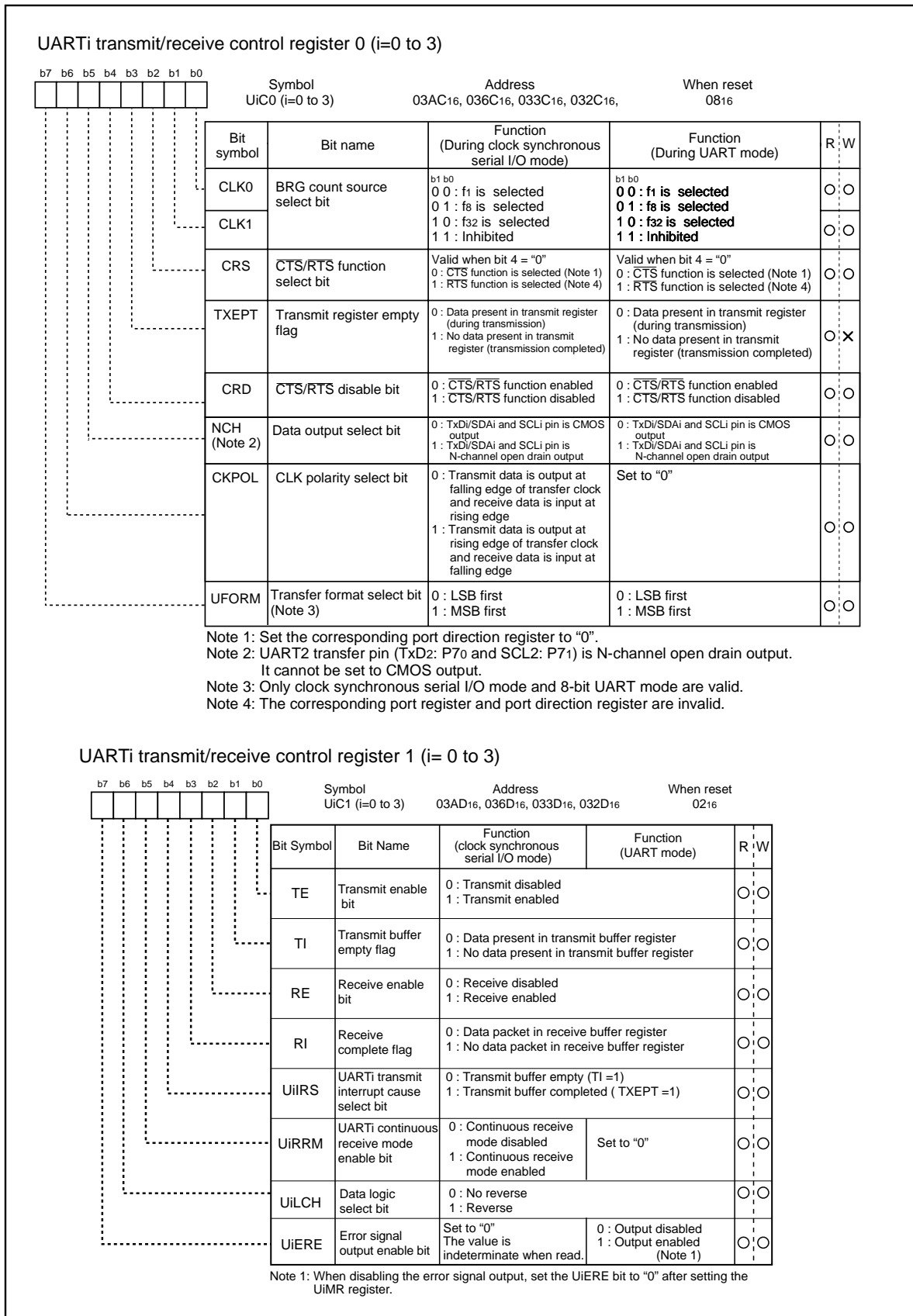
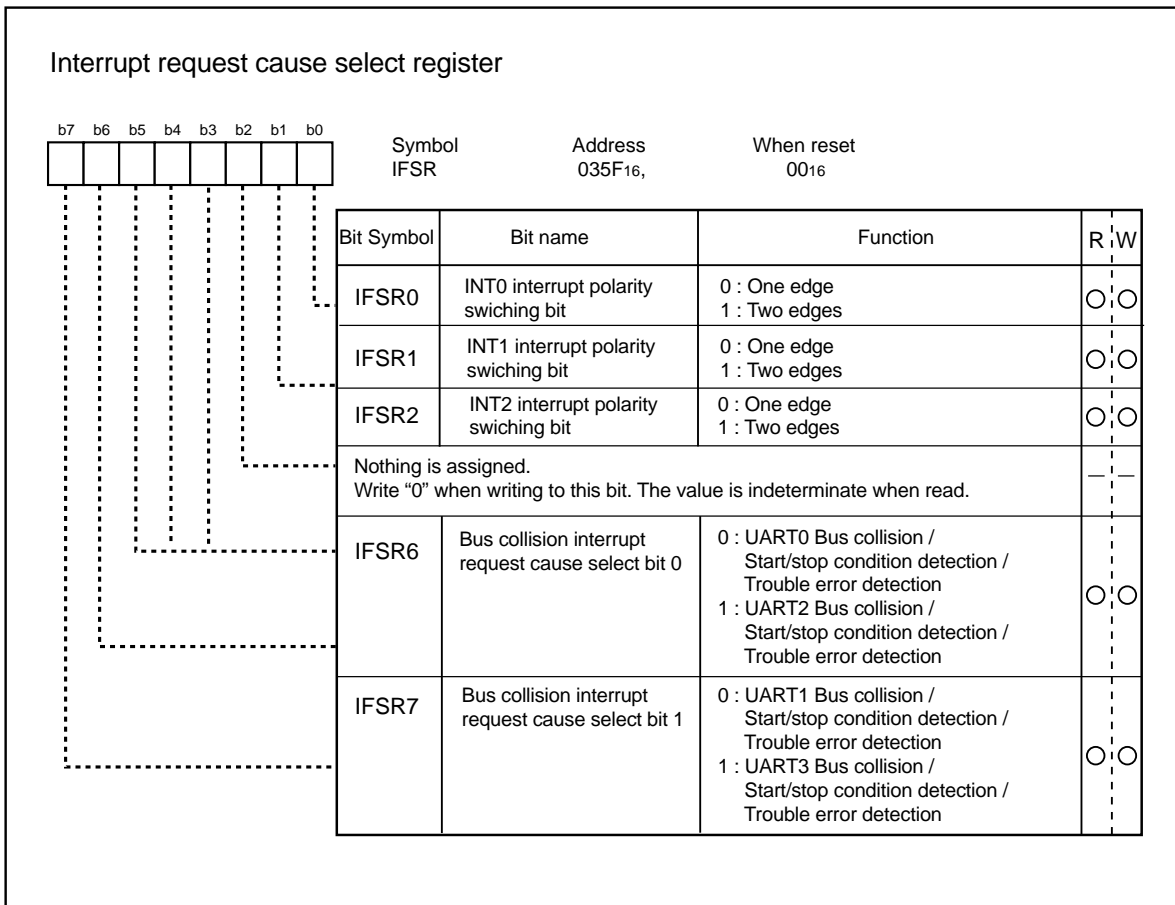


Figure 2.4.5. UARTi-related registers (3)



**Figure 2.4.6. UARTi-related registers (4)**



## 2.4.2 Operation of Serial I/O (transmission in UART mode)

In transmitting data in UART mode, choose functions from those listed in Table 2.4.4. Operations of the circled items are described below. Figure 2.4.7 shows the operation timing, and Figures 2.4.8 and 2.4.9 show the set-up procedures.

**Table 2.4.4. Chosed functions**

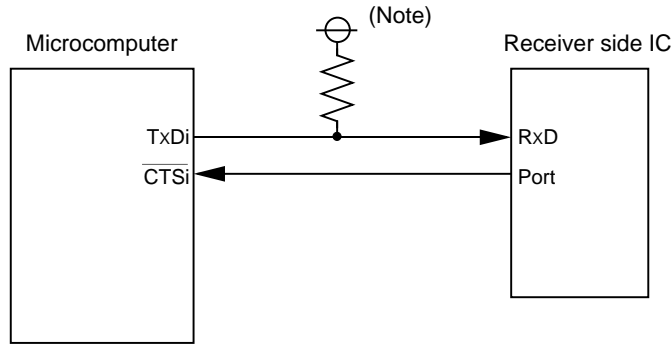
Item	Set-up	Item	Set-up
Transfer clock source	<input type="radio"/> Internal clock ( $f_1 / f_8 / f_{32}$ )	Data logic select function	<input type="radio"/> No reverse
	<input type="radio"/> External clock (CLKi pin)		<input type="radio"/> Reverse
$\overline{\text{CTS}}$ function	<input type="radio"/> $\overline{\text{CTS}}$ function enabled	TxD, RxD I/O polarity reverse bit	<input type="radio"/> No reverse
	<input type="radio"/> $\overline{\text{CTS}}$ function disabled		<input type="radio"/> Reverse
Transmission interrupt factor	<input type="radio"/> Transmission buffer empty	Bus collision detection function	<input type="radio"/> Not selected
	<input type="radio"/> Transmission complete		<input type="radio"/> Selected

- Operation
- (1) Setting the transmit enable bit to "1" and writing transmission data to the UARTi transmit buffer register readies the data transmissible status.
  - (2) When input to the  $\overline{\text{CTS}}_i$  pin goes to "L", transmission starts (the  $\overline{\text{CTS}}_i$  pin needs to be controlled on the reception side).
  - (3) Transmission data held in the UARTi transmit buffer register is transmitted to the UARTi transmit register. At this time, the first bit (the start bit) of the transmission data is transmitted from the TxDi pin. Then, data is transmitted, bit by bit, in sequence: LSB, ..., MSB, parity bit, and stop bit(s).
  - (4) When the stop bit(s) is (are) transmitted, the transmit register empty flag goes to "1", which indicates that transmission is completed. At this time, the UARTi transmit interrupt request bit goes to "1". The transfer clock stops at "H" level.
  - (5) If the transmission condition of the next data is ready when transmission is completed, a start bit is generated following to stop bit(s), and the next data is transmitted.

Note

- Set  $\overline{\text{CTS}}_i$  pin's port direction register to "0".

Example of wiring



Note: Since TxDi pin is N-channel open drain, this pin needs pull-up resistance.

Example of operation

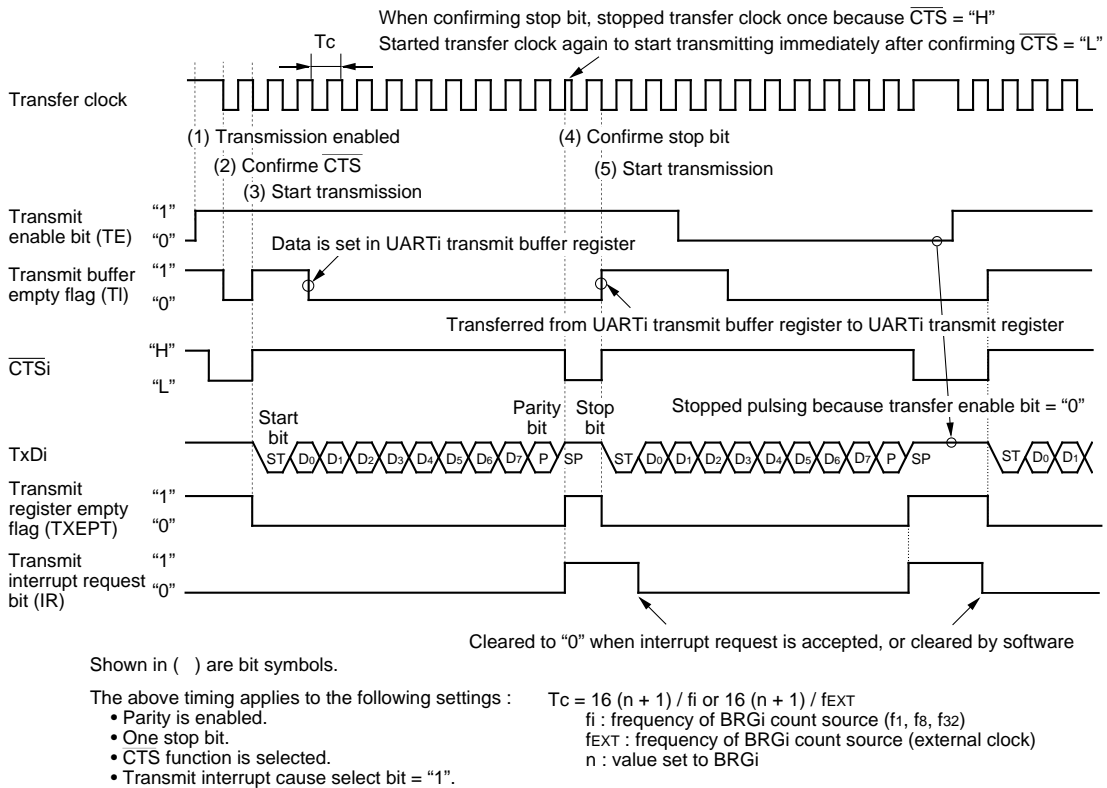


Figure 2.4.7. Operation timing of transmission in UART mode

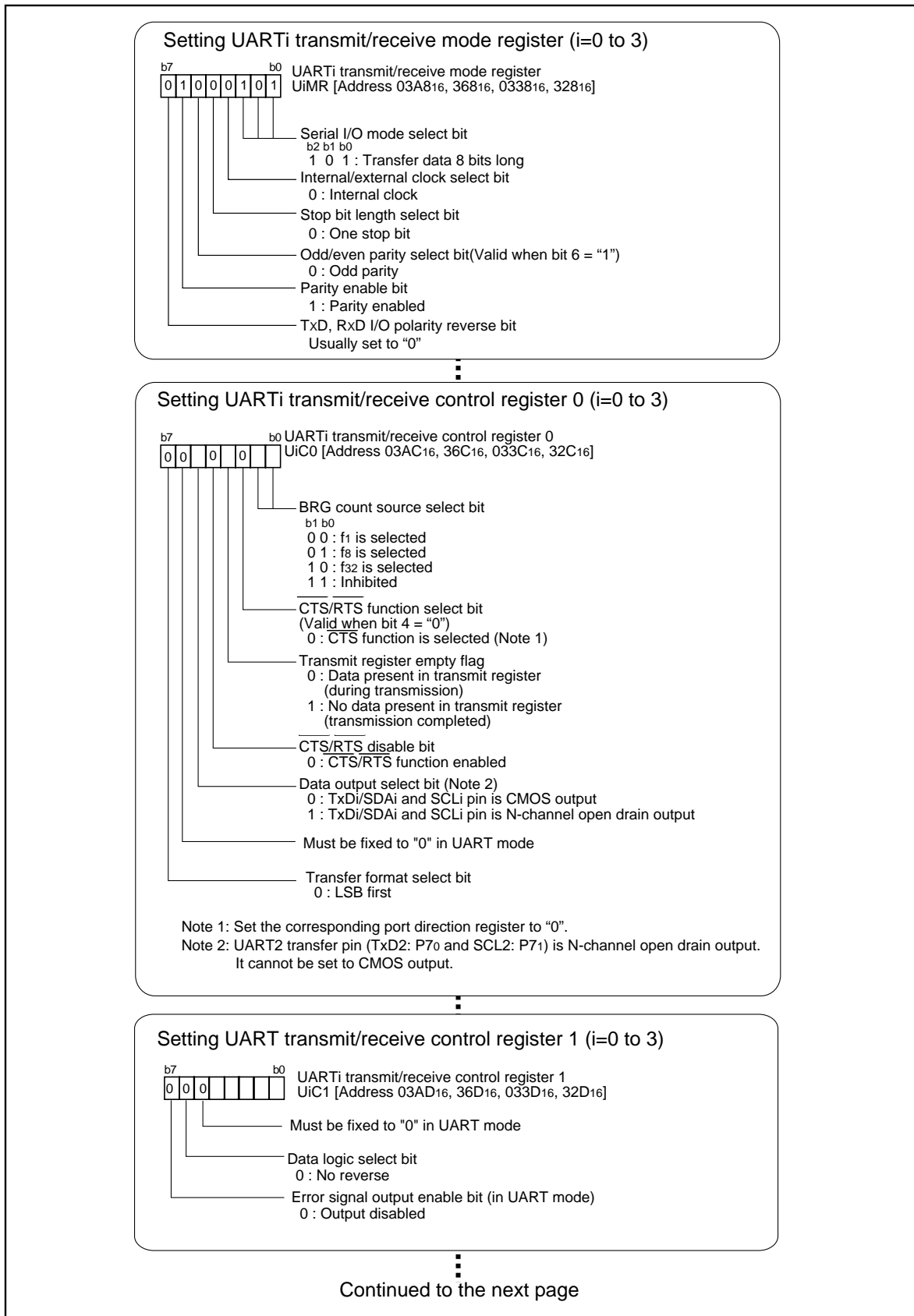
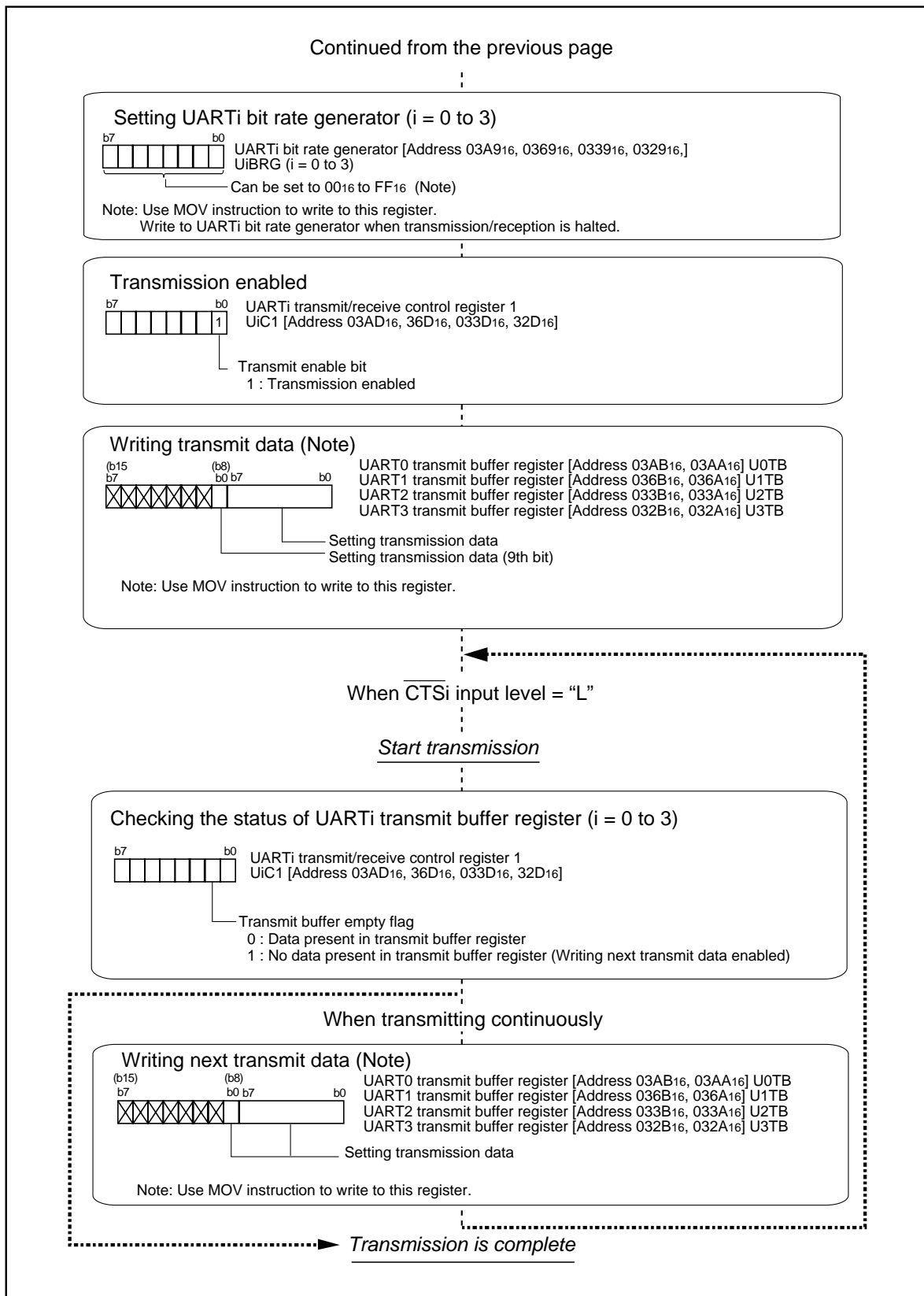


Figure 2.4.8. Set-up procedure of transmission in UART mode (1)



**Figure 2.4.9. Set-up procedure of transmission in UART mode (2)**

### 2.4.3 Operation of Serial I/O (reception in UART mode)

In receiving data in UART mode, choose functions from those listed in Table 2.4.5. Operations of the circled items are described below. Figure 2.4.10 shows the operation timing, and Figures 2.4.11 and 2.4.12 show the set-up procedures.

**Table 2.4.5. Choosed functions**

Item	Set-up		Item	Set-up	
Transfer clock source	<input type="radio"/>	Internal clock (f <sub>1</sub> / f <sub>8</sub> / f <sub>32</sub> )	Tx/D, Rx/D I/O polarity reverse bit	<input type="radio"/>	No reverse
		External clock (CLKi pin)			Reverse
RTS function	<input type="radio"/>	RTS function enabled	Bus collision detection function	<input type="radio"/>	Not selected
		RTS function disabled			Selected
Data logic select function	<input type="radio"/>	No reverse			
		Reverse			

- Operation
- (1) Setting the receive enable bit to “1” readies data-receivable status. At this time, output from the  $\overline{\text{RTSi}}$  pin goes to “L” level to inform the transmission side that the receivable status is ready.
  - (2) When the first bit (the start bit) of reception data is received from the RxDi pin, output from the  $\overline{\text{RTS}}$  goes to “H” level. Then, data is received, bit by bit, in sequence: LSB, ..., MSB, and stop bit(s).
  - (3) When the stop bit(s) is (are) received, the content of the UARTi receive register is transmitted to the UARTi receive buffer register. At this time, the receive complete flag goes to “1” to indicate that the reception is completed, the UARTi receive interrupt request bit goes to “1”, and output from the  $\overline{\text{RTS}}$  pin goes to “L” level.
  - (4) The receive complete flag goes to “0” when the lower-order byte of the UARTi buffer register is read.

- Note
- Set RxDi pin's port direction register to “0”.

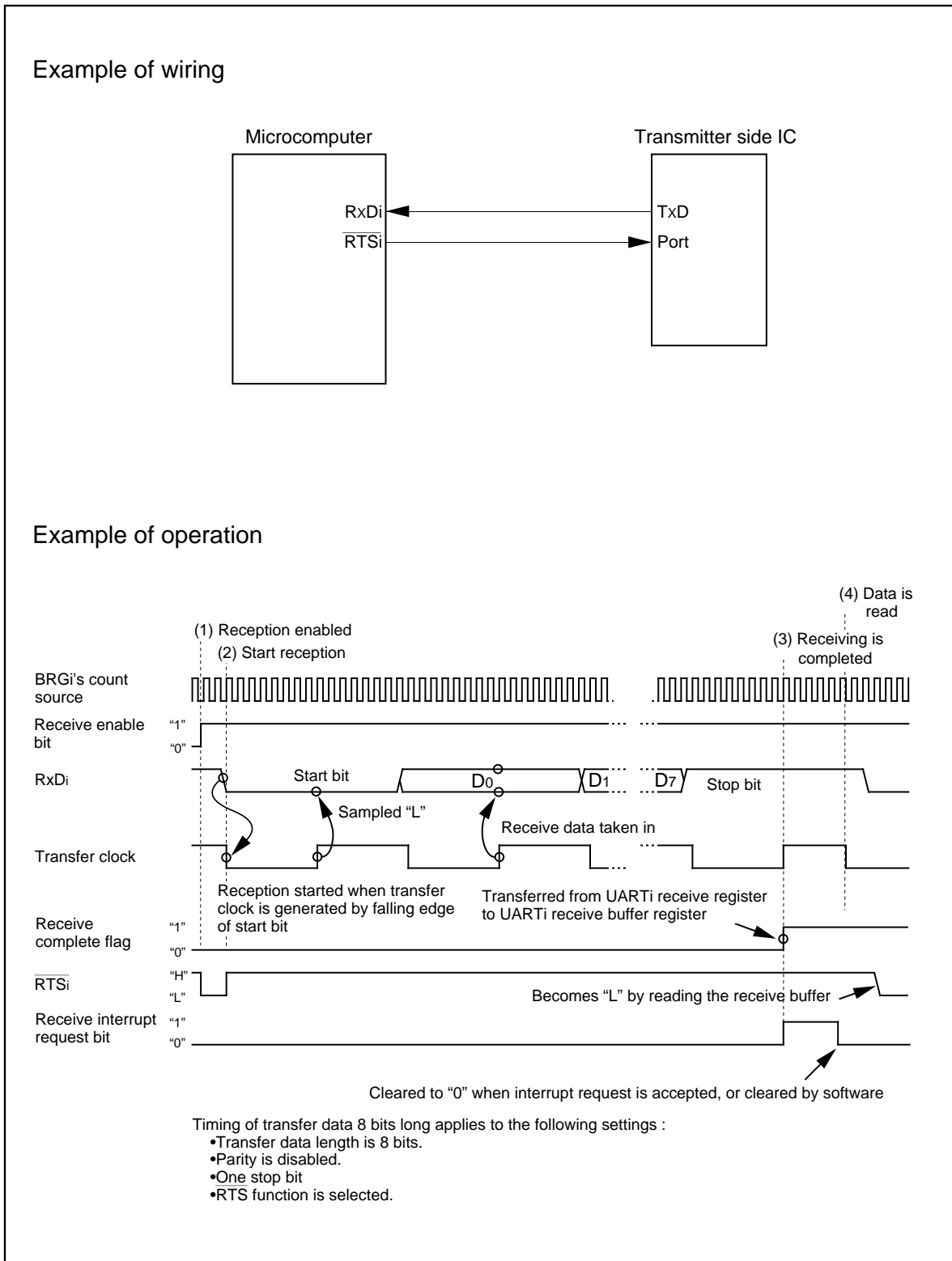


Figure 2.4.10. Operation timing of reception in UART mode

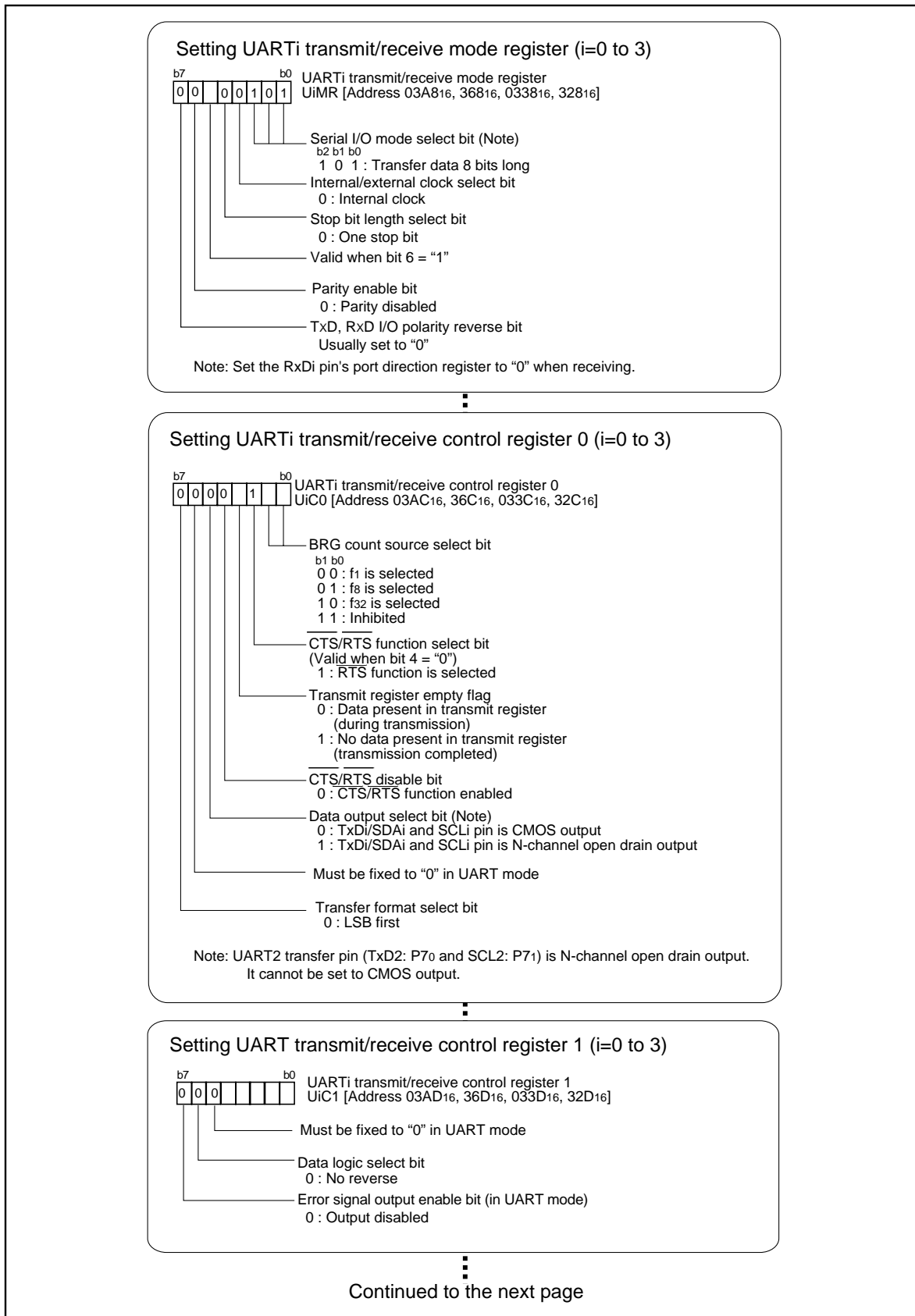


Figure 2.4.11. Set-up procedure of reception in UART mode (1)

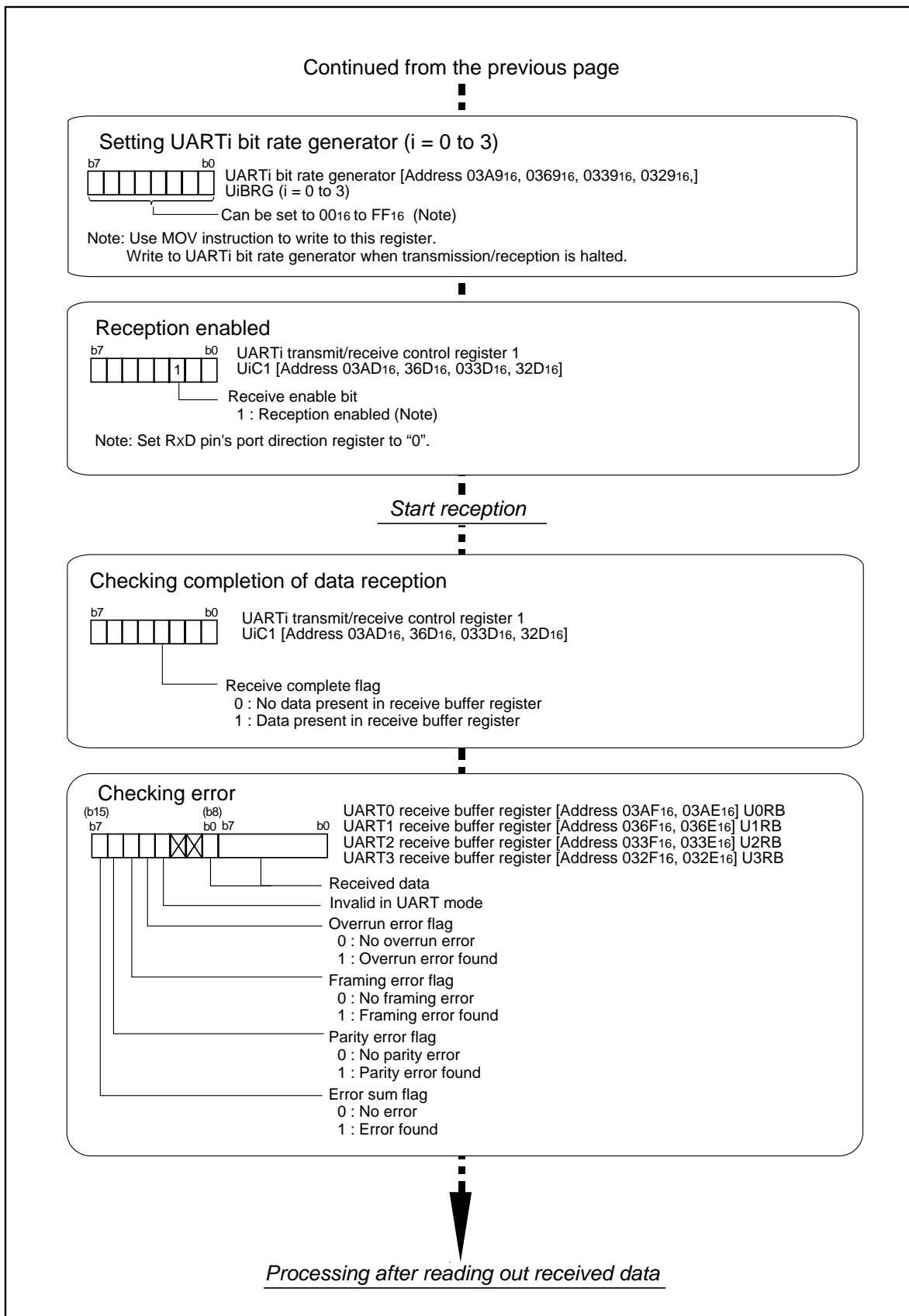


Figure 2.4.12. Set-up procedure of reception in UART mode (2)



#### 2.4.4 Serial I/O Precautions (UART Mode)

**Description** When the level of the CLKi and CTSi pins goes to “H” (Note 1), if the UiMR register is set to any of the following, the UiERE bit in the UiC1 register is set to “1” (parity error signal output enabled). When the PRYE bit in the UiMR register is set to “1” while the UiERE bit is “1” (parity error signal output enabled), the TXDi pin outputs “L” level if a parity error occurs while receiving data. To prevent this, set the UiERE bit after setting the UiMR register.

- Change the setting of bits SMD2 to SMD0 from “0002” (serial I/O disabled) to “1012” (UART mode transfer data length 8 bits).
- Change the setting of bits SMD2 to SMD0 from “0012” (clock synchronous serial I/O mode) to “1002” (UART mode transfer data length 7 bits).
- Change the setting of bits SMD2 to SMD0 from “0012” (clock synchronous serial I/O mode) to “1012” (UART mode transfer data length 8 bits).
- Change the setting of bits SMD2 to SMD0 from “0012” (clock synchronous serial I/O mode) to “1102” (UART mode transfer data length 9 bits).
- Change the setting of bits SMD2 to SMD0 from “0102” (I<sup>2</sup>C mode) to “1012” (UART mode transfer data length 8 bits).

**Note 1:** If the pins are not used as CLKi or CTSi, these conditions apply when the pin level goes to “H”.

### 2.4.5 Operation of Serial I/O (transmission used for SIM interface)

In transmitting data in UARTi (i=0 to 3) mode (used for SIM interface), choose functions from those listed in Table 2.4.6. Operations of the circled items are described below. Figure 2.4.13 shows the operation timing, and Figures 2.4.14 and 2.4.15 show the set-up procedures.

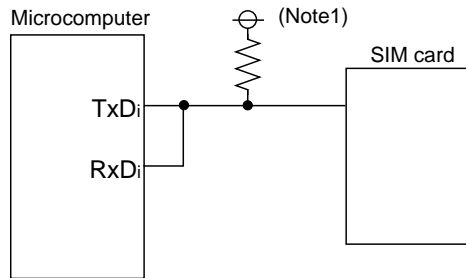
**Table 2.4.6. Chosed functions**

Item	Set-up		Item	Set-up	
Transfer data format	○	Direct format	Transfer clock source	○	Internal clock (f <sub>1</sub> /f <sub>8</sub> /f <sub>32</sub> )
		Inverse format			External clock (CLKi pin)

- Operation
- (1) Setting the transmit enable bit and receive enable bit to "1" and writing transmission data to the UARTi (i=0 to 3) transmit buffer register readies the data transmissible status. Set UARTi (i=0 to 3) transfer interrupt for being enabled.
  - (2) Transmission data held in the UARTi (i=0 to 3) transmit buffer register is transmitted to the UARTi (i=0 to 3) transmit register. At this time, the first bit (the start bit) of the transmission data is transmitted from the TxDi (i=0 to 3) pin. Then, data is transmitted, bit by bit, in sequence: LSB, ..., MSB, parity bit, and stop bit(s).
  - (3) When the stop bit(s) is (are) transmitted, the transmit register empty flag goes to "1", which indicates that transmission is completed. At this time, the UARTi (i=0 to 3) transmit interrupt request bit goes to "1". The transfer clock stops at "H" level.
  - (4) If the transmission condition of the next data is ready when transmission is completed, a start bit is generated following to stop bit(s), and the next data is transmitted.
  - (5) If a parity error occurs, an L is output from the SIM card, and the RxDi (i=0 to 3) terminal turns to "L" level. Check the RxDi (i=0 to 3) terminal's level within the UARTi (i=0 to 3) transmission interrupt routine, and if it is found to be at the "L" level, then handle the error.

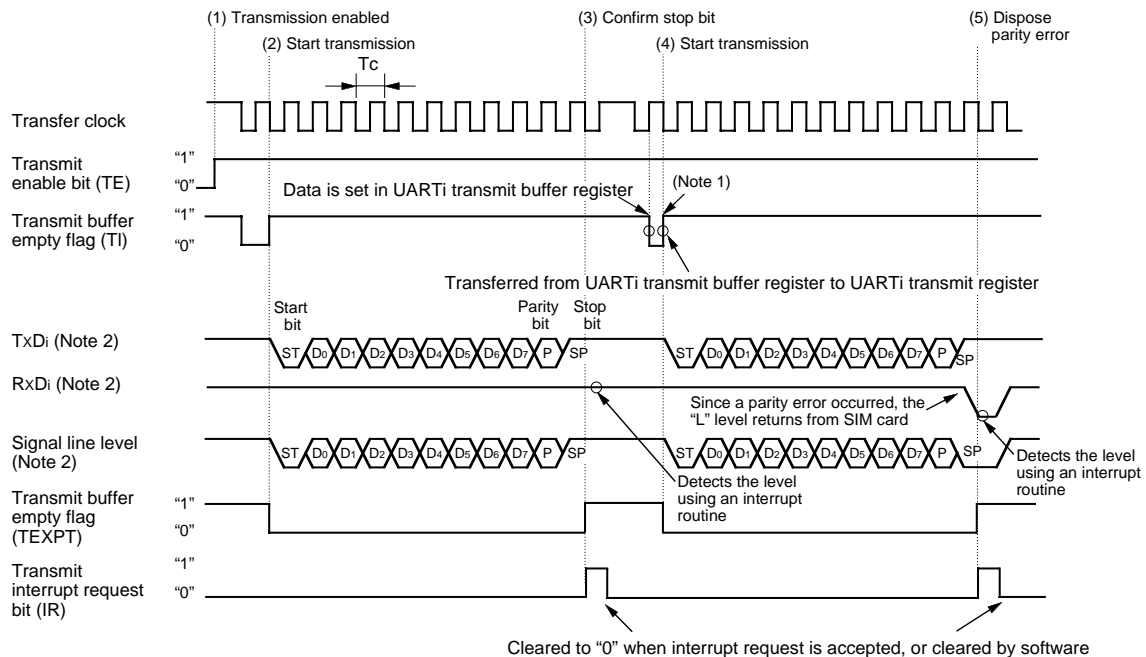
- Note
- The parity error level is determined within a UARTi (i=0 to 3) transmission interrupt. When a transmission interrupt request occurs, set the priority level of the transmission interrupt higher than those of other interrupts so that the interrupt routine can be immediately carried out. Either in the main routine or in an interrupt routine, the interrupt inhibition time has to be made as short as possible.
  - Set the RxDi (i=0 to 3) pin's port direction register to input.
  - Select N-channel open drain output for TxDi pin with data output select bit of UARTi (i=0 to 3) transmit/receive control register 0.

Example of wiring



Note1: TxDi pin is N-channel open drain and needs a pull-up resistance.  
 Note2: i=0 to 3

Example of operation (when direct format)



Shown in ( ) are bit symbols.

The above timing applies to the following settings :

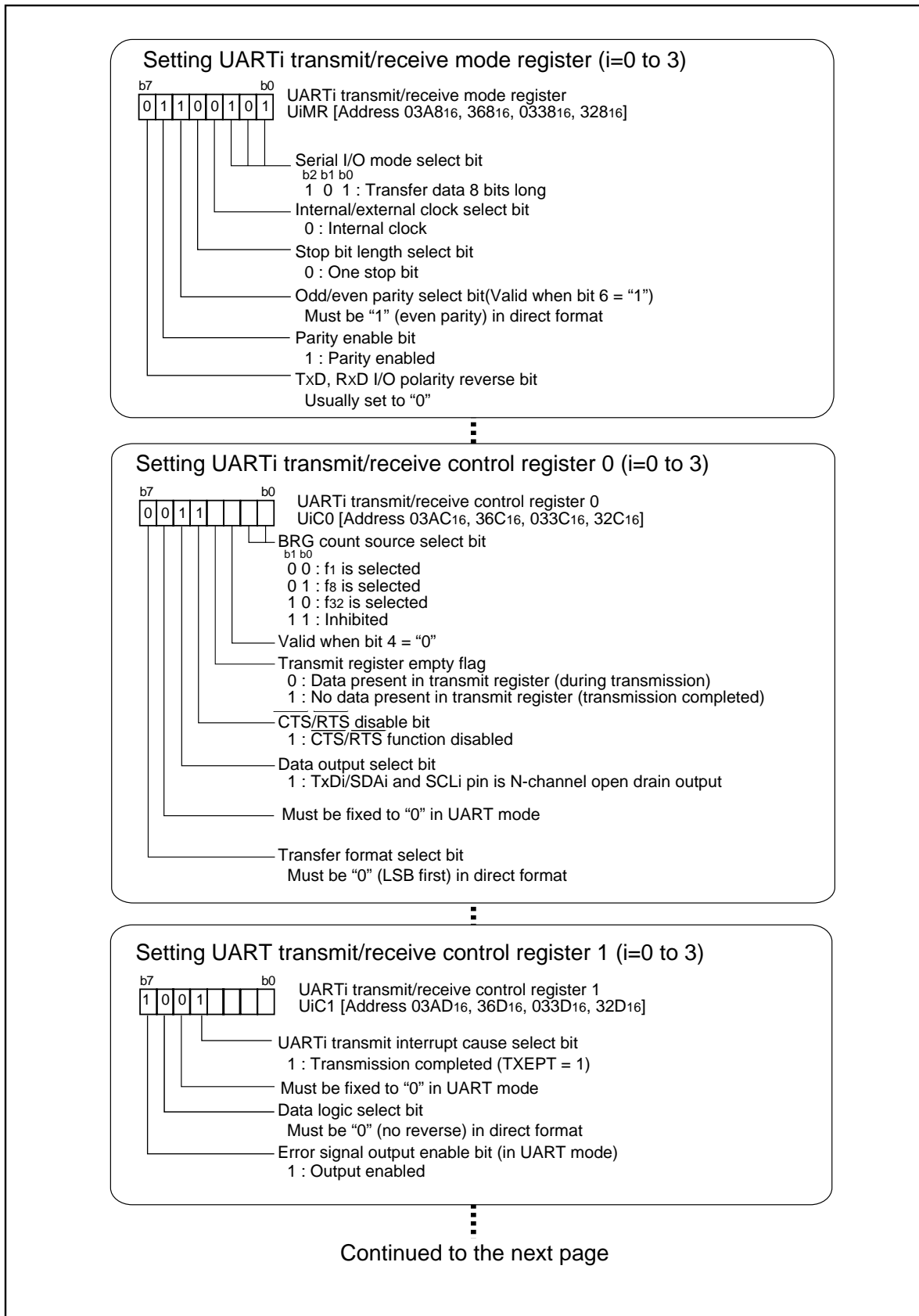
- Parity is enabled.
- One stop bit.
- Transmit interrupt cause select bit = "1".

$$Tc = 16(n + 1) / fi \text{ or } 16(n + 1) / fEXT$$

fi : frequency of BRGi count source (f1, f8, f32)  
 fEXT : frequency of BRGi count source (external clock)  
 n : value set to BRGi

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.  
 Note 2: TxDi and RxDi are connected in the manner of wired OR as shown in the connection diagram. Therefore, the signal levels of TxDi and RxDi should be the same, but the output signals are shown separately for ease of understanding. Also, the signal level resulting from connecting TxDi and RxDi is shown as a signal line level.  
 Note 3: i = 0 to 3

Figure 2.4.13. Operation timing of transmission in UART mode (used for SIM interface)



**Figure 2.4.14. Set-up procedure of transmission in UART mode (used for SIM interface) (1)**

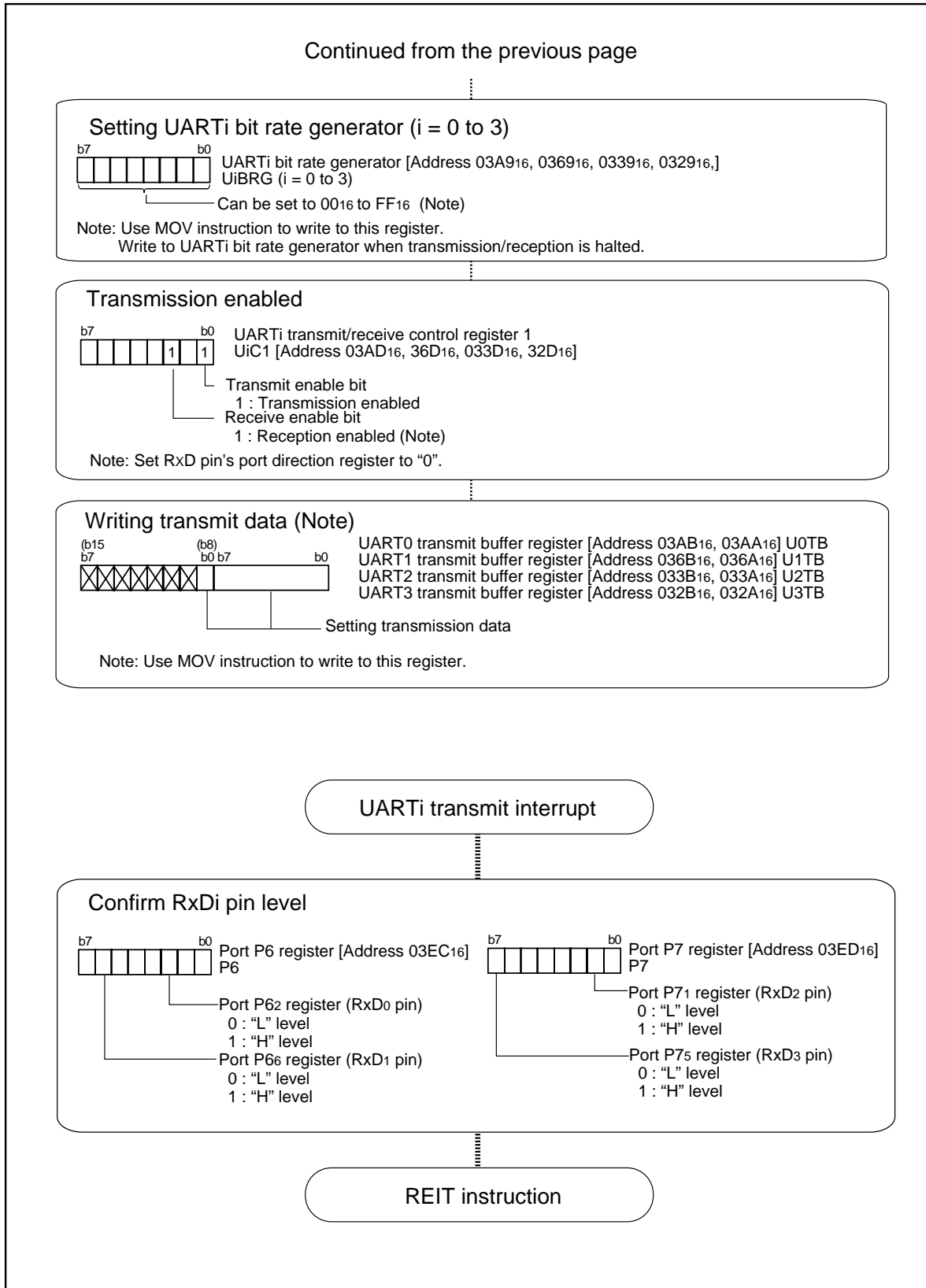


Figure 2.4.15. Set-up procedure of transmission in UART mode (used for SIM interface) (2)

## 2.4.6 Operation of Serial I/O (reception used for SIM interface)

In receiving data in UARTi (i=0 to 3) mode (used for SIM interface), choose functions from those listed in Table 2.4.7. Operations of the circled items are described below. Figure 2.4.16 shows the operation timing, and Figures 2.4.17 to 2.4.18 show the set-up procedures.

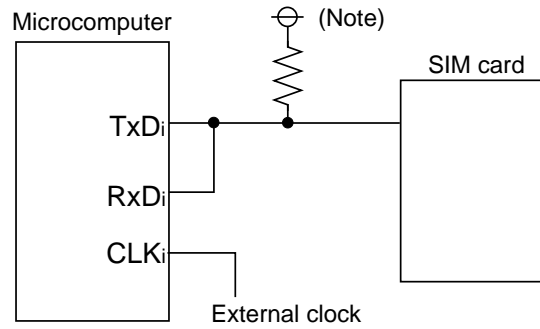
**Table 2.4.7. Chosed functions**

Item	Set-up		Item	Set-up	
Transfer data format		Direct format	Transfer clock source		Internal clock (f <sub>1</sub> /f <sub>8</sub> /f <sub>32</sub> )
	○	Inverse format		○	External clock (CLKi pin)

- Operation
- (1) Setting the transmit enable bit and receive enable bit to “1” readies data-receivable status.
  - (2) When the first bit (the start bit) of reception data is received from the RxDi (i=0 to 3) pin, data is received, bit by bit, in sequence: LSB, ..., MSB, and stop bit(s).
  - (3) When the stop bit(s) is (are) received, the content of the UARTi (i=0 to 3) receive register is transmitted to the UARTi (i=0 to 3) receive buffer register.  
At this time, the receive complete flag goes to “1” to indicate that the reception is completed, and the UARTi (i=0 to 3) receive interrupt request bit goes to “1”.
  - (4) The receive complete flag goes to “0” when the lower-order byte of the UARTi (i=0 to 3) buffer register is read.
  - (5) When the parity error is occurred, TxDi (i=0 to 3) pin goes to “L” level.

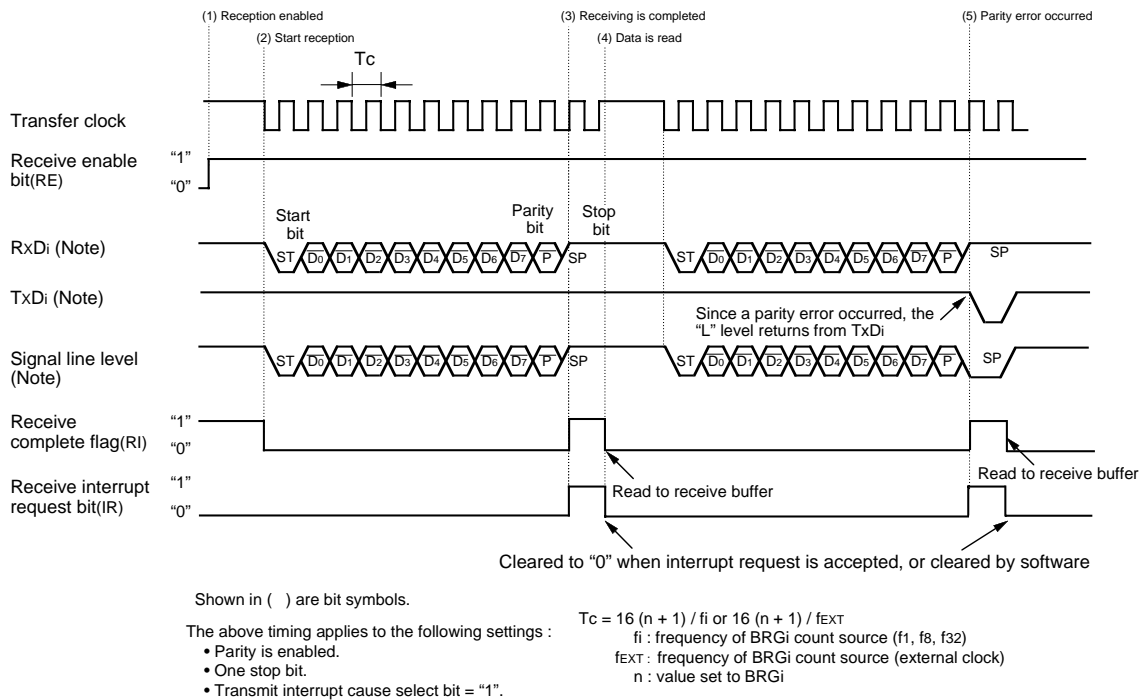
- Note
- Set the RxDi and CLKi pins' port direction register to “0”.
  - Select N-channel open drain output for TxDi (i=0 to 3) pin with data output select bit of UARTi (i=0 to 3) transmit/receive control register 0.

Example of wiring



Note1: TxDi pin is N-channel open drain and needs a pull-up resistance.  
 Note2: i=0 to 3

Example of operation (when inverted format)



Note : TxDi and RxDi are connected in the manner of wired OR as shown in the connection diagram. So TxDi and RxDi ought to become the same signal from the logical standpoint, but the output signals turn complex, so they are shown separately. Also, the signal level resulting from connecting TxDi and RxDi is shown as a signal line level.

Figure 2.4.16. Operation timing of reception in UART mode (used for SIM interface)

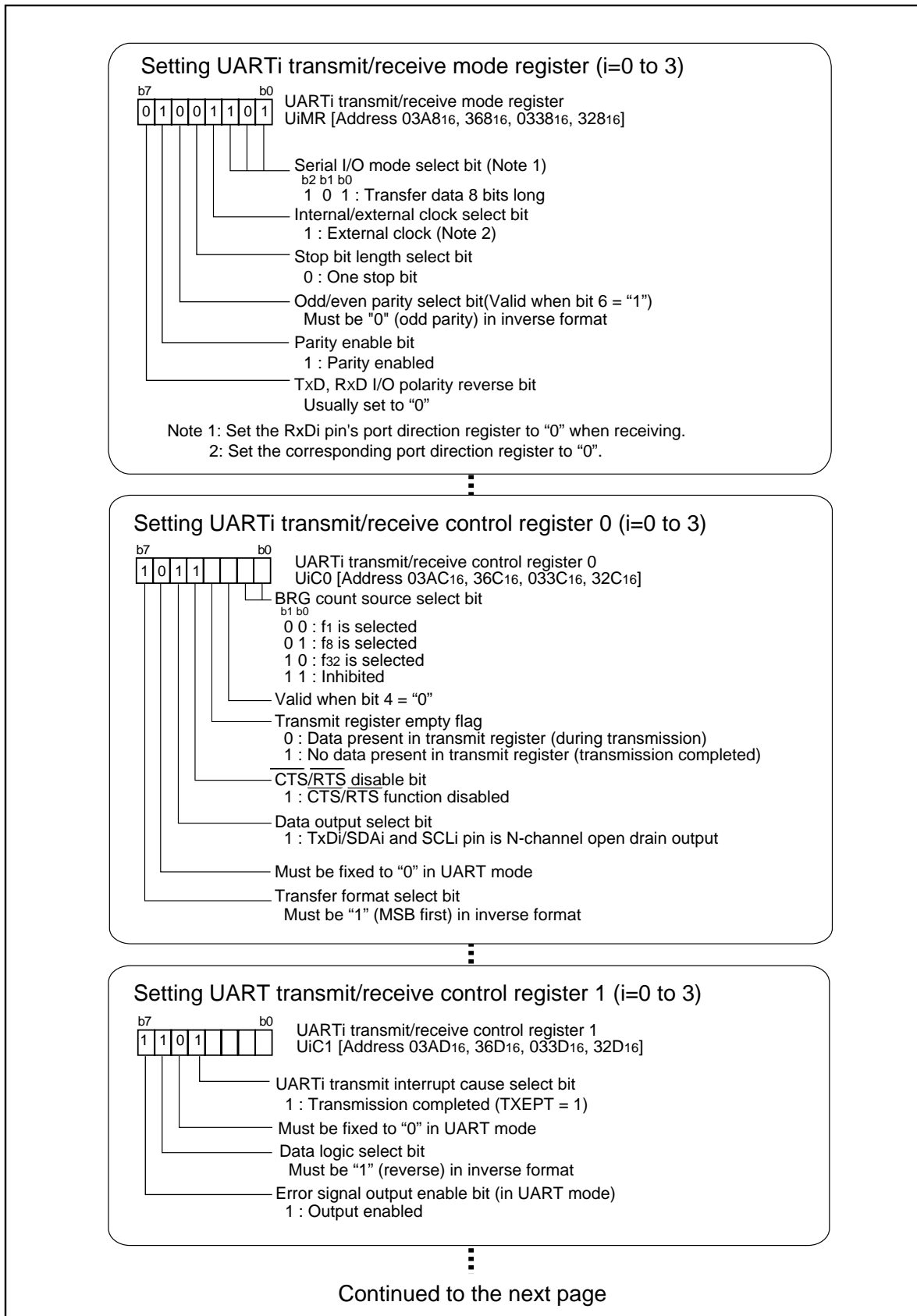
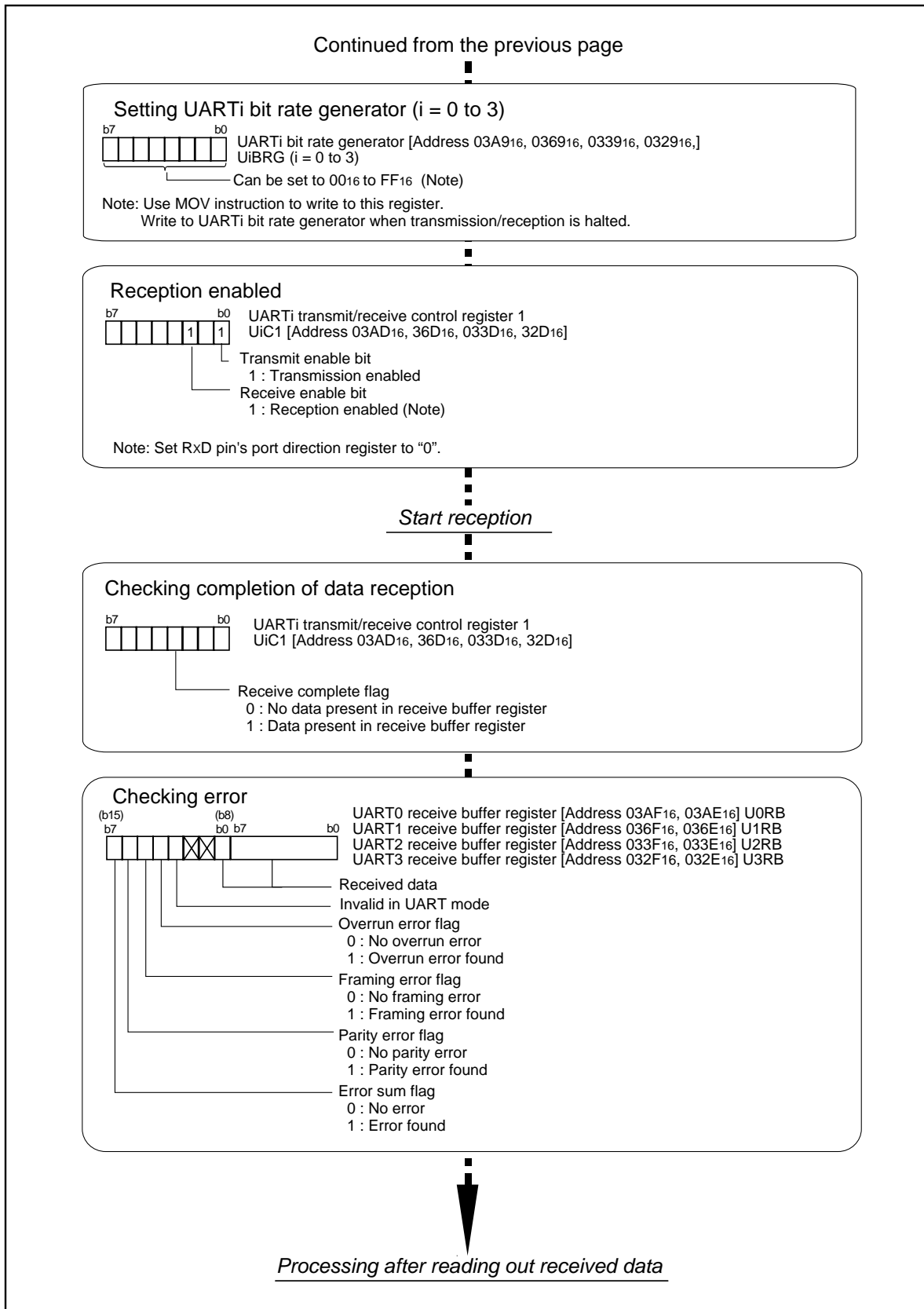


Figure 2.4.17. Set-up procedure of reception in UART mode (used for SIM interface) (1)





**Figure 2.4.18. Set-up procedure of reception in UART mode (used for SIM interface)(2)**

### 2.4.7 Clock Signals in used for the SIM Interface

In conforming to the SIM interface, the UART clock signal within the SIM card needs to conform to the UART<sub>i</sub> (i=0 to 3) clock signal within the microprocessor. Two examples are given here as means of generating a UART<sub>i</sub> clock signal within the microprocessor.

- \* In the case of setting a value equal to or less than (1/256 X 1/16) in the division rate of UART<sub>i</sub> clock  
Choose f<sub>1</sub> for the UART's source clock signal and set an optional value in the bit rate generator.
- \* In the case of setting a value equal to or greater than (1/256 X 1/16) in the division rate of UART<sub>i</sub> clock  
Set the bit rate generator to "0", turn the source clock signal to timer output and set an optional value in the timer.

Let F be the clock signal within the SIM card and D be the bit rate adjustment factor, then the formula for the UART clock signal becomes as follows. Figure 2.4.19 shows an example of connection.

- In the case of setting a value equal to or less than (1/256 X 1/16) in the division rate of UART<sub>i</sub> clock  
UART<sub>i</sub> clock signal within microprocessor = UART clock within SIM card

$$f_1 \times \frac{1}{\text{Bit rate generator} + 1} \times \frac{1}{16} = f_1 \times \frac{1}{\text{Timer Aj counter} + 1} \times \text{flip-flop} \times \frac{1}{F/D}$$

Let X<sub>IN</sub> = 16 MHz, timer Aj counter = 1, F = 372, and D = 1, then the value to be set in the bit rate generator becomes

$$16 \times \frac{1}{\text{Bit rate generator} + 1} \times \frac{1}{16} = 16 \times \frac{1}{1+1} \times \frac{1}{2} \times \frac{1}{372/1}$$

Bit rate generator = 92

Table 2.4.8 shows an example of setting in the UART<sub>i</sub> bit rate generator.

- In the case of setting a value equal to or greater than (1/256 X 1/16) in the division rate of UART<sub>i</sub> clock  
UART<sub>i</sub> clock signal within microprocessor = UART clock within SIM card

$$f_1 \times \frac{1}{\text{Timer Ak counter} + 1} \times \text{flip-flop} \times \frac{1}{\text{Bit rate generator} + 1} \times \frac{1}{16}$$

$$= f_1 \times \frac{1}{\text{Timer Aj counter} + 1} \times \text{flip-flop} \times \frac{1}{F/D}$$

Let X<sub>IN</sub> = 16 MHz, timer Aj counter = 3, bit rate generator = 0, F = 1860, and D = 1, then the value to be set in the timer Ak counter becomes

$$16 \times \frac{1}{\text{Timer Ak counter} + 1} \times \frac{1}{2} \times \frac{1}{0+1} \times \frac{1}{16} = 16 \times \frac{1}{3+1} \times \frac{1}{2} \times \frac{1}{1860/1}$$

Timer Ak counter = 464

Table 2.4.9 shows an example of setting in the timer Ak counter.

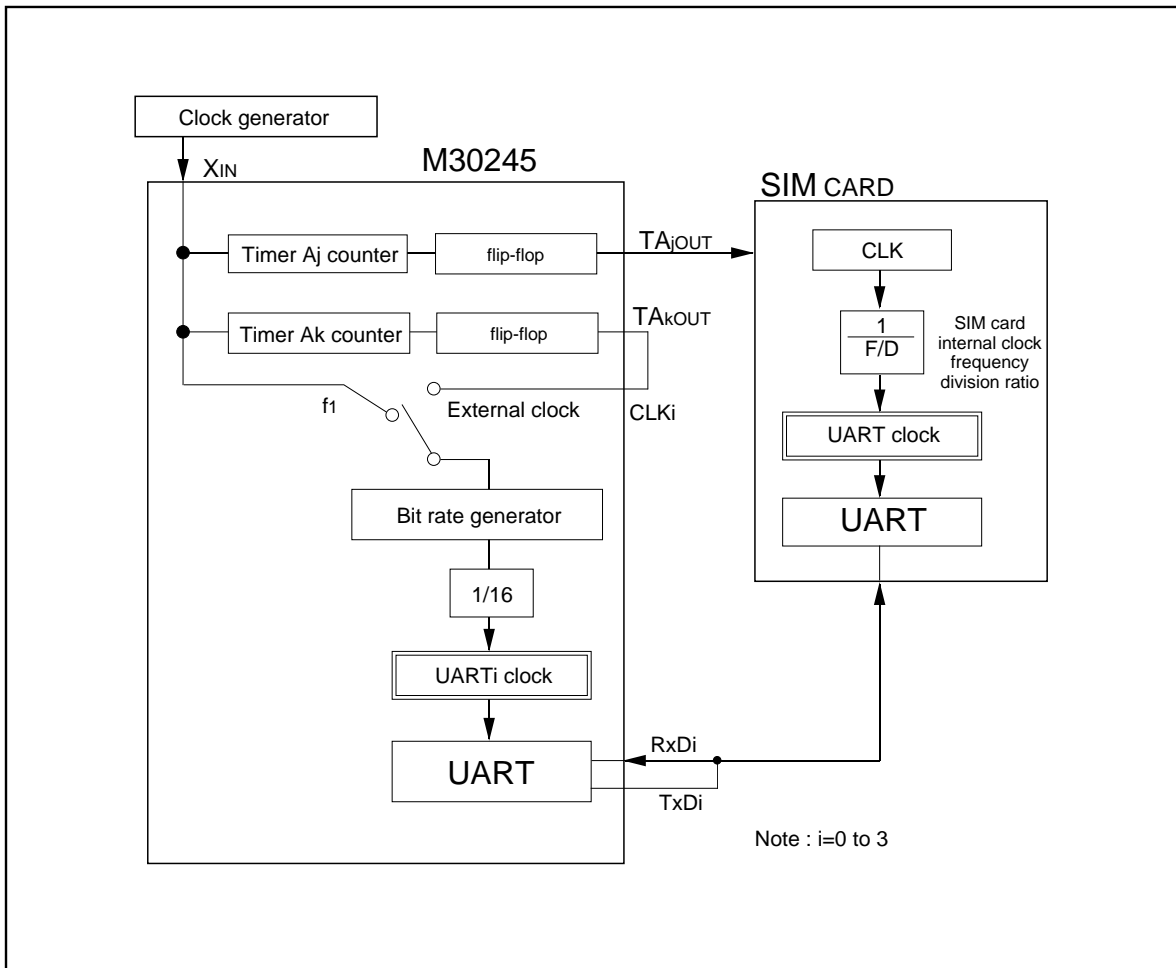



Figure 2.4.19. Example of connection

Table 2.4.8. UARTi bit rate adjustment factor (i=0 to 3)

SIM card internal clock F(Hz)	Bit rate D	F/D	UARTi bit rate generator set value	SIM card internal clock F(Hz)	Bit rate D	F/D	UARTi bit rate generator set value
372	1	372	92	1116	1	1116	
	2	186			2	558	
	4	93			4	279	
	8				8		
	16				16		
	1/2	744	185		1/2	2232	
	1/4	1488			1/4	4464	
	1/8	2976			1/8	8928	
	1/16	5952			1/16	17856	
	1/32	11904			1/32	35712	
	1/64	23808			1/64	71424	
558	1	558		1488	1	1488	
	2	279			2	744	185
	4				4	372	92
	8				8	186	
	16				16	93	
	1/2	1116			1/2	2976	
	1/4	2232			1/4	5952	
	1/8	4464			1/8	11904	
	1/16	8928			1/16	23808	
	1/32	17856			1/32	47616	
	1/64	35712			1/64	95232	
744	1	744	185	1860	1	1860	
	2	372	92		2	930	
	4	186			4	465	
	8	93			8		
	16				16		
	1/2	1488			1/2	3720	
	1/4	2976			1/4	7440	
	1/8	5952			1/8	14880	
	1/16	11904			1/16	29760	
	1/32	23808			1/32	59520	
	1/64	47616			1/64	119040	

 Combination impossible to deal with due to the current specifications of M30245

 Combination in which the F/D itself does not become an integer


Setting example under the following conditions.


$f(X_{IN})=16$  MHz

Timer Ak counter set value = 1

Table 2.4.9. TimerAi register adjustment factor

SIM card internal clock F(Hz)	Bit rate D	F/D	Timer Ai value	SIM card internal clock F(Hz)	Bit rate D	F/D	Timer Aj value
372	1	372	92	1116	1	1116	278
	2	186			2	558	
	4	93			4	279	
	8				8		
	16				16		
	1/2	744	185		1/2	2232	557
	1/4	1488	371		1/4	4464	1115
	1/8	2976	743		1/8	8928	2231
	1/16	5952	1487		1/16	17856	4463
	1/32	11904	2975		1/32	35712	8927
	1/64	23808	5951		1/64	71424	17855
	558	1	558			1488	1
2		279		2	744		185
4				4	372		92
8				8	186		
16				16	93		
1/2		1116	278	1/2	2976		743
1/4		2232	557	1/4	5952		1487
1/8		4464	1115	1/8	11904		2975
1/16		8928	2231	1/16	23808		5951
1/32		17856	4463	1/32	47616		11903
1/64		35712	8927	1/64	95232		23807
744		1	744	185	1860		1
	2	372	92	2		930	
	4	186		4		465	
	8	93		8			
	16			16			
	1/2	1488	371	1/2		3720	929
	1/4	2976	743	1/4		7440	1859
	1/8	5952	1487	1/8		14880	3719
	1/16	11904	2975	1/16		29760	7439
	1/32	23808	5951	1/32		59520	14879
	1/64	47616	11903	1/64		119040	29759

 Combination impossible to deal with due to the current specifications of M30245

 Combination in which the F/D itself does not become an integer

Setting example under the following conditions.

$f(XIN)=16$  MHz

Timer Aj counter set value = 3, UARTi bit rate generator set value = 0

## 2.5 Serial Interface Special Function

### 2.5.1 Overview

Serial interface special function can control communications on the serial bus using  $\overline{\text{SSi}}$  input pins. The following is an overview of the serial interface special function.

#### (1) Transmission/reception format

8-bit data

#### (2) Transfer rate

If the internal clock is selected as the transfer clock, the divide-by-2 frequency, resulting from the bit rate generator division, becomes the transfer rate. The bit rate generator count source can be selected from the following: f1, f8, and f32. Clocks f1, f8, and f32 are derived by dividing the CPU's main clock by 1, 8, and 32 respectively.

Furthermore, if an external clock is selected as the transfer clock, the clock frequency input to the CLK pin becomes the transfer rate.

#### (3) Error detection

Fault error can be detected in the master mode.

When an "L" signal is input to an  $\overline{\text{SSi}}$  pin in the multiple master system, it is judged there is another master existed, and the TxDi, RxDi and CLKi pins all become high impedance. Moreover, the fault error interrupt request bit becomes "1" and a fault error interrupt is generated.

#### (4) How to deal with an error

When the fault error flag is set to "0", output is restored to the clock output and data output pins. In the master mode, if an  $\overline{\text{SSi}}$  input pin is H level, "0" can be written for the fault error flag. When an  $\overline{\text{SSi}}$  input pin is L level, "0" cannot be written for the fault error flag. In the slave mode, the "0" can be written for the fault error flag regardless of the input to the  $\overline{\text{SSi}}$  input pins.

#### (5) Function selection

For serial interface special function, the following functions can be selected:

##### (a) Function for choosing CLK polarity

This function switches the CLK polarity of the transfer clock. The following operations are available:

- Data is input at the falling edge of the transfer clock, and is output at the rising edge.
- Data is input at the rising edge of the transfer clock, and is output at the falling edge.

##### (b) Function for setting clock phase

This function switches the phase of the transfer clock. Choose either of the following:

- Without clock delay
- With clock delay

##### (c) Function for setting serial input pin

This function switches the serial bus control privilege between the master mode and slave mode. Choose either of the following:

- Master mode
- Slave mode

Following are some examples in which various functions (a) through (c) are selected:

- Transmission Operation WITH: outputting transmission data at falling edge of transfer clock, no clock delay, master mode
- Reception Operation WITH: inputting reception data at rising edge of transfer clock, clock delay, master mode
- Transmission Operation WITH: outputting transmission data at falling edge of transfer clock, no clock delay, slave mode
- Reception Operation WITH: inputting reception data at rising edge of transfer clock, clock delay, slave mode

#### **(6) Input to the serial interface special function and the direction register**

To input an external signal to the serial interface special function, set the direction register of the relevant port to input.

#### **(7) Pins related to the serial interface special function**

- CLK0, CLK1, CLK2, CLK3 pins    Input/output pins for the transfer clock
- RxD0/SRxD0, RxD1/SRxD1, RxD2/SRxD2, RxD3/SRxD3 pins  
  Input pins for data
- TxD0/STxD0, TxD1/STxD1, TxD2/STxD2, TxD3/STxD3 pins  
  Output pins for data

#### **(8) Registers related to the serial I/O**

Figure 2.5.1 shows the memory map of serial interface special function-related registers, and Figures 2.5.2 to 2.5.7 show serial interface special function-related registers.

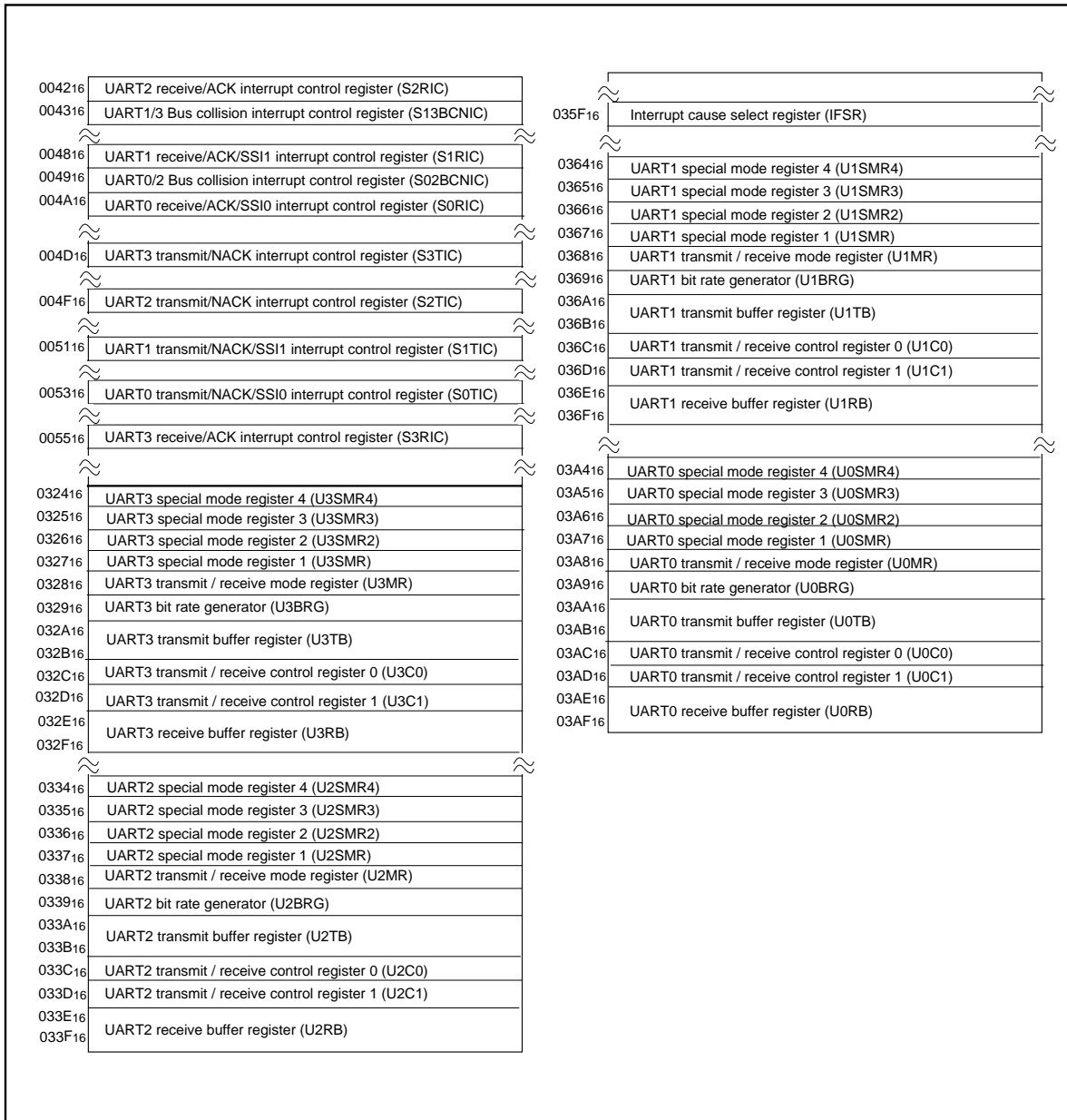


Figure 2.5.1. Memory map of serial interface special function-related registers



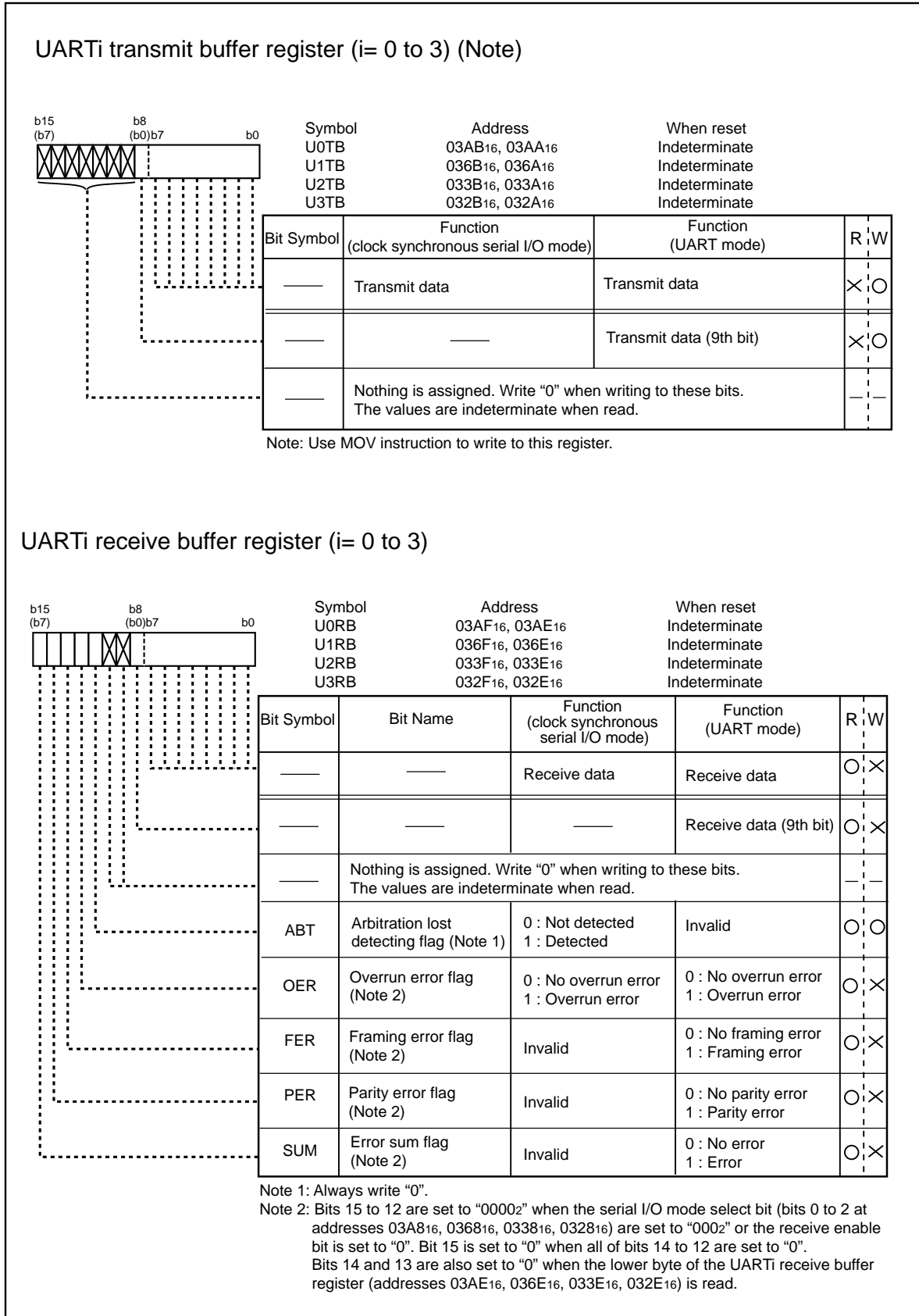


Figure 2.5.2. Serial interface special function-related registers (1)

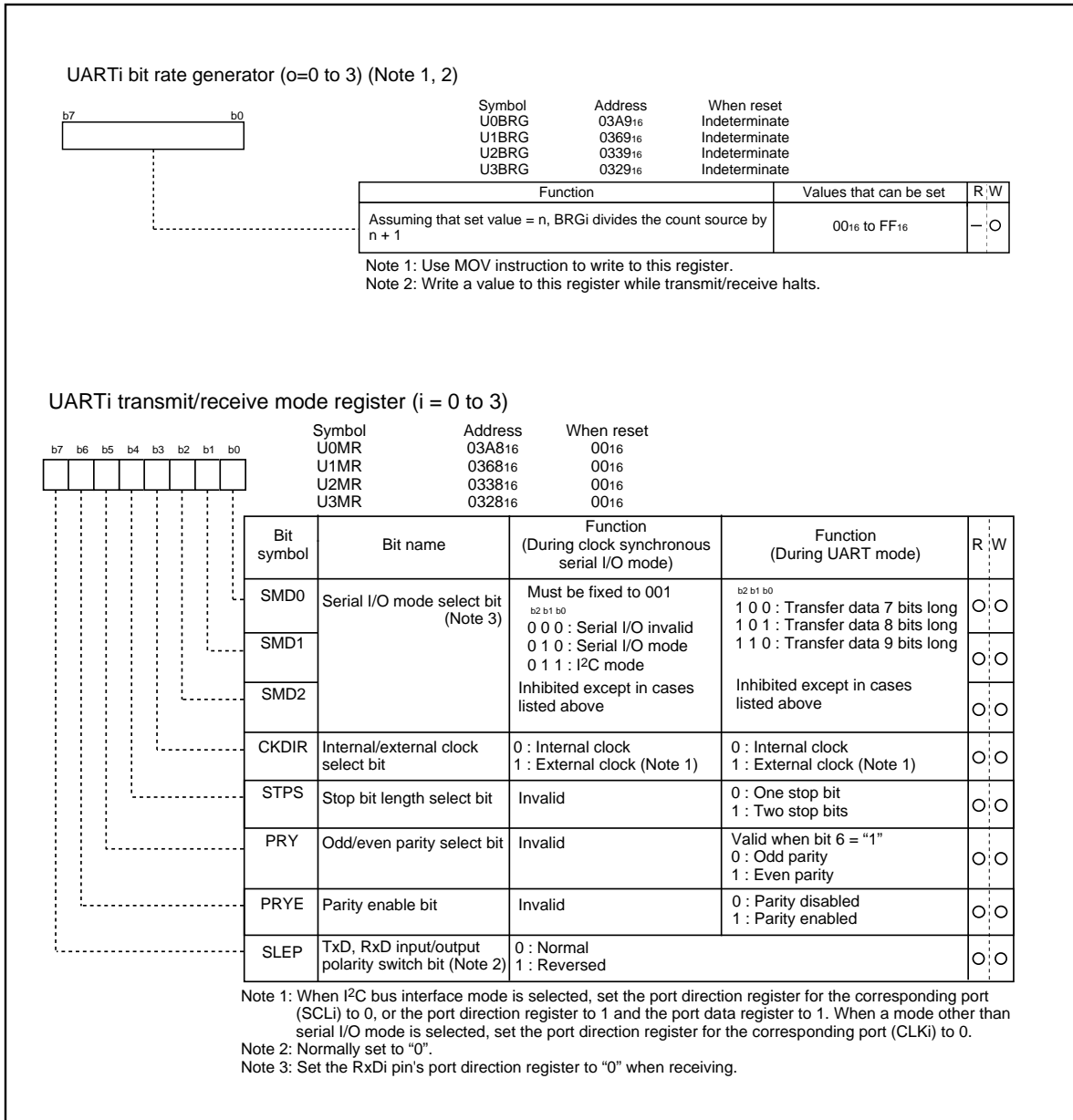


Figure 2.5.3. Serial interface special function-related registers (2)

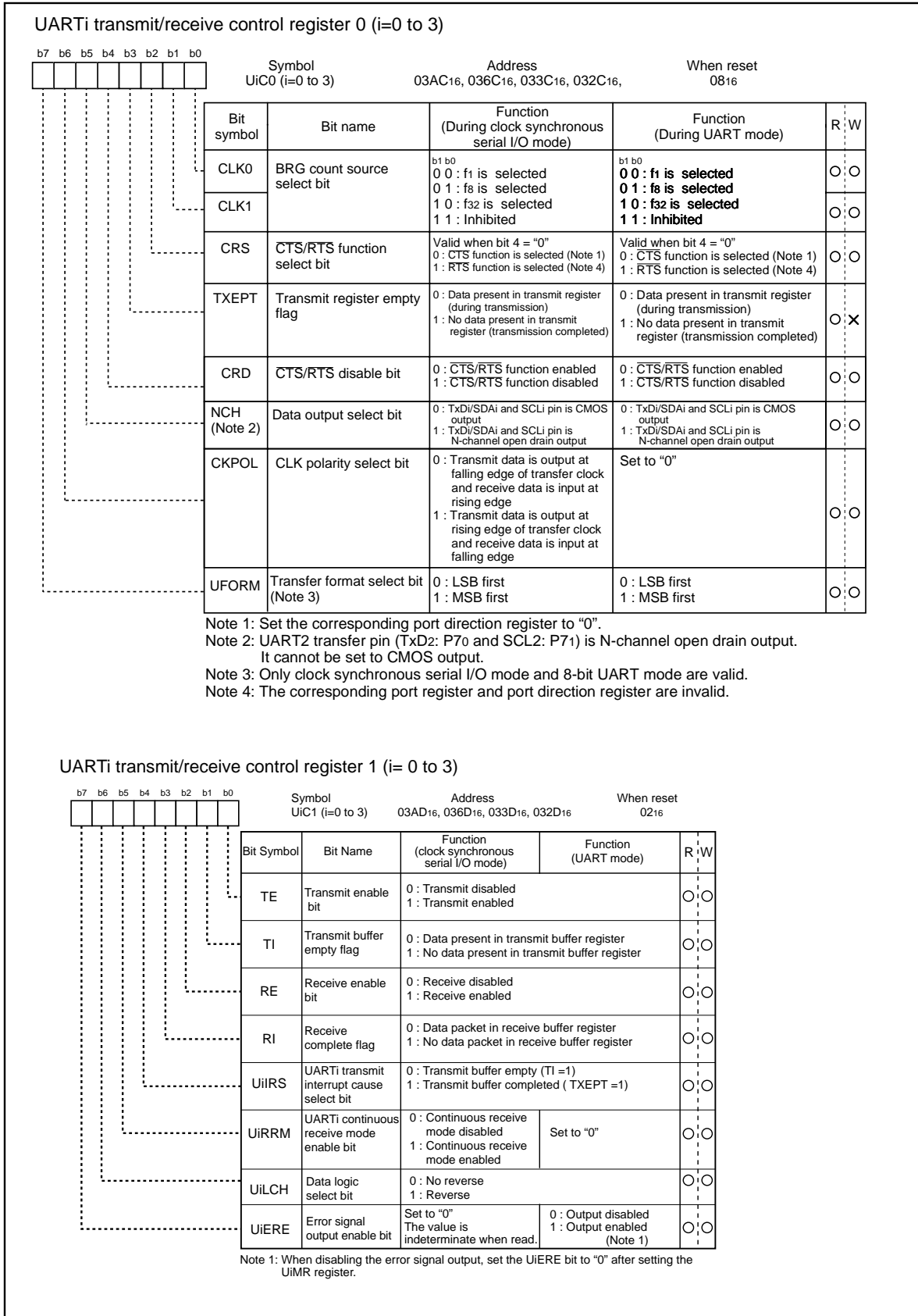


Figure 2.5.4. Serial interface special function-related registers (3)

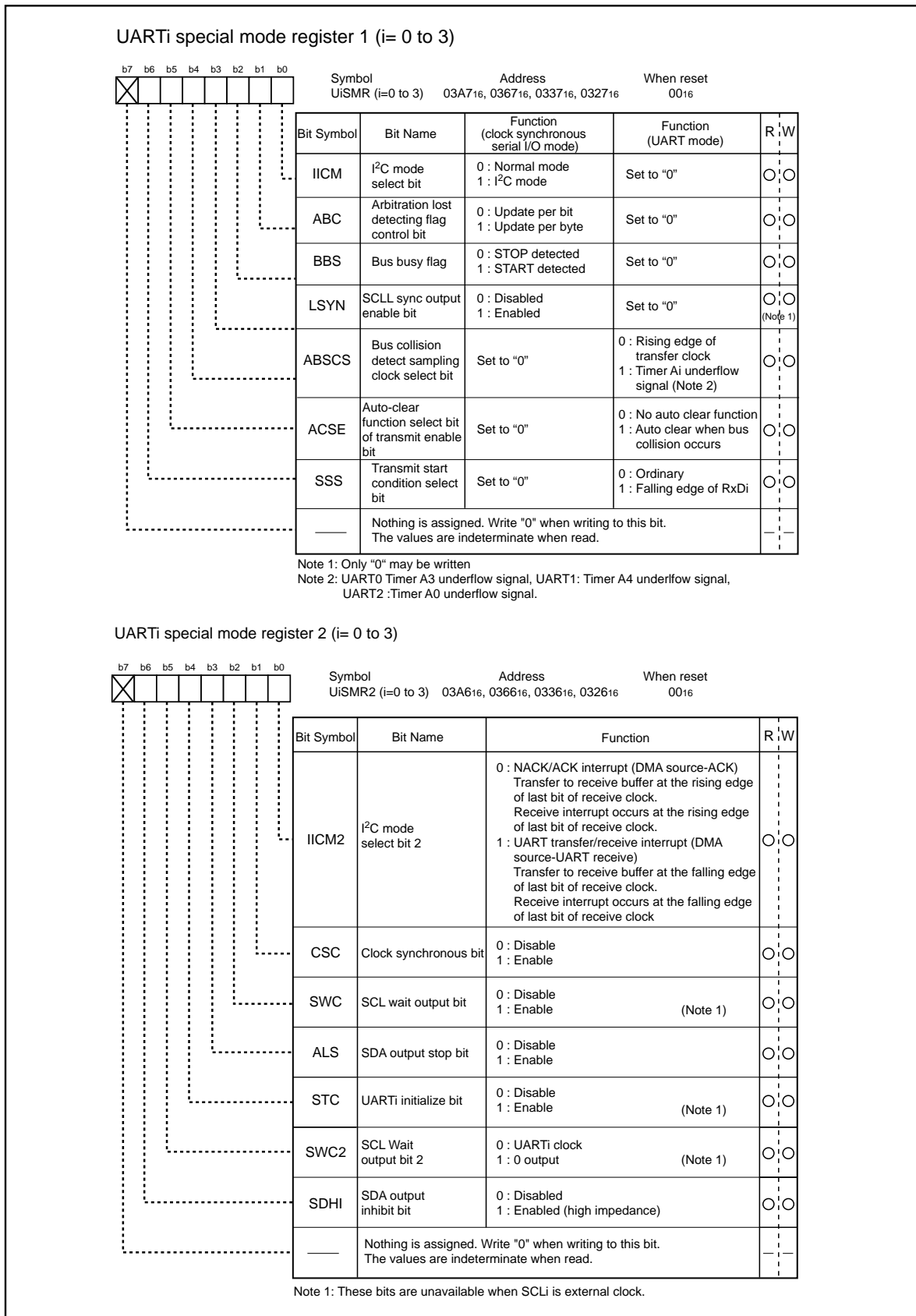


Figure 2.5.5. Serial interface special function-related registers (4)

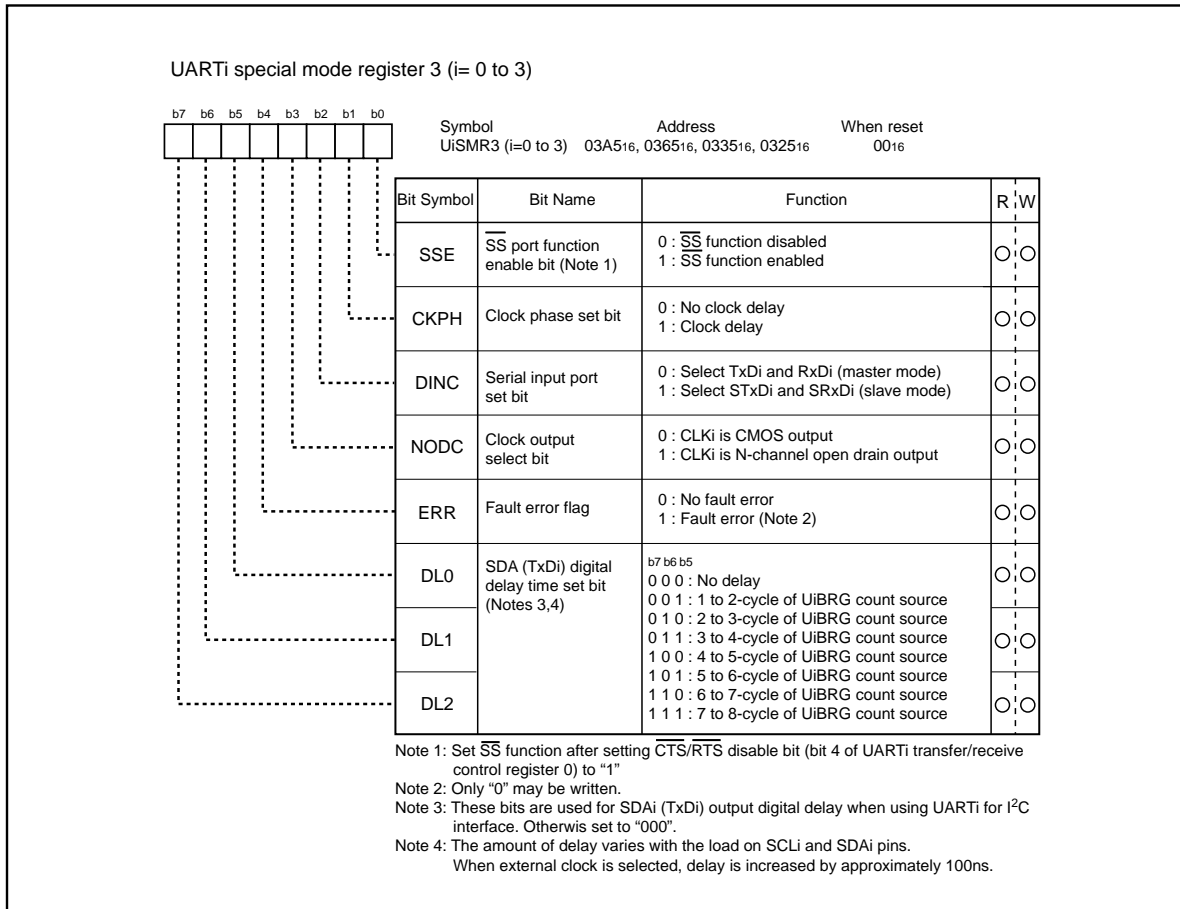


Figure 2.5.6. Serial interface special function-related registers (5)

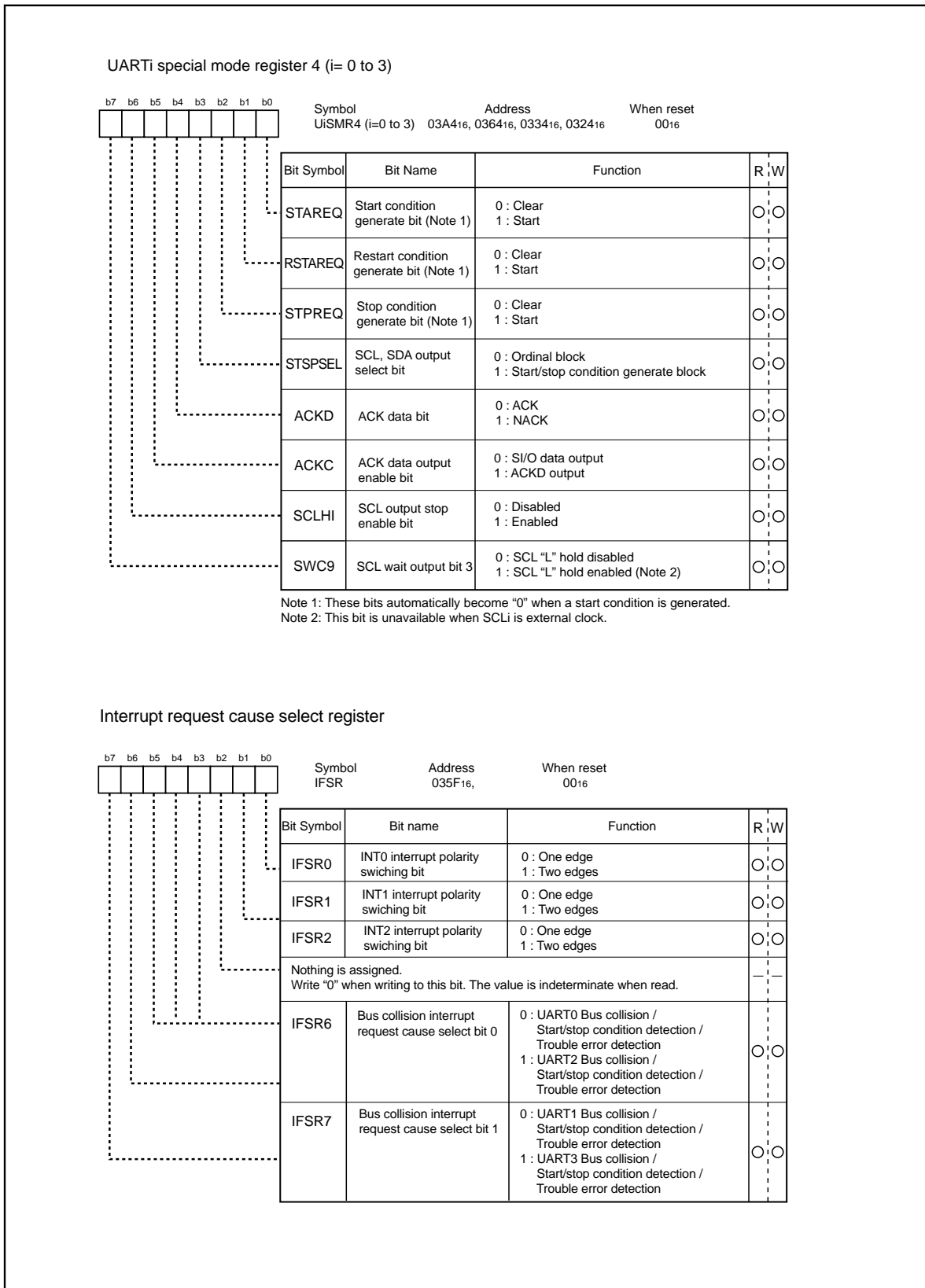


Figure 2.5.7. Serial interface special function-related registers (6)

## 2.5.2 Operation of Serial Interface Special Function (transmission in master mode without delay)

In transmitting data in serial interface special function master mode, choose functions from those listed in Table 2.5.1. Operations of the circled items are described below. Figure 2.5.8 shows the operation timing, and Figures 2.5.9 and 2.5.10 show the set-up procedures.

**Table 2.5.1. Chosed functions**

Item	Set-up	Item	Set-up
Transfer clock source	<input type="radio"/> Internal clock ( $f_1 / f_8 / f_{32}$ )	$\overline{SSi}$ port function enable	<input type="checkbox"/> $\overline{SSi}$ function disabled
	<input type="checkbox"/> External clock (CLKi pin)		<input type="radio"/> $\overline{SSi}$ function enabled
CLK polarity	<input type="radio"/> Output transmission data at the falling edge of the transfer clock	Clock phase set	<input type="radio"/> Without clock delay
	<input type="checkbox"/> Output transmission data at the rising edge of the transfer clock		<input type="checkbox"/> With clock delay
Transmission interrupt factor	<input type="radio"/> Transmission buffer empty	Serial input port set	<input type="radio"/> TxDi, RxDi selected (master mode)
	<input type="checkbox"/> Transmission complete		<input type="checkbox"/> STxDi, SRxDi selected (slave mode)

- Operation
- (1) Set an  $\overline{SS}$  port of the receiver side IC to output "L" level.
  - (2) Setting the transmit enable bit to "1" and writing transmission data to the UARTi transmit buffer register makes data transmissible status ready.
  - (3) In synchronization with the first falling edge of the transfer clock, transmission data held in the UARTi transmit buffer register is transmitted to the UARTi transmit register. At this time, the UARTi transmit interrupt request bit goes to "1". Also, the first bit of the transmission data is transmitted from the TxDi pin. Then the data is transmitted bit by bit from the lower order in synchronization with the falling edges.
  - (4) When transmission of 1-byte data is completed, the transmit register empty flag goes to "1", which indicates that transmission is completed. The transfer clock stops at "L" level.
  - (5) If the next transmission data is set in the UARTi transmit buffer register while transmission is in progress (before the eighth bit has been transmitted), the data is transmitted in succession.

Note

- Set  $\overline{SSi}$  pin to "H" level. If "L" level is input to the pin, a fault error will be generated.

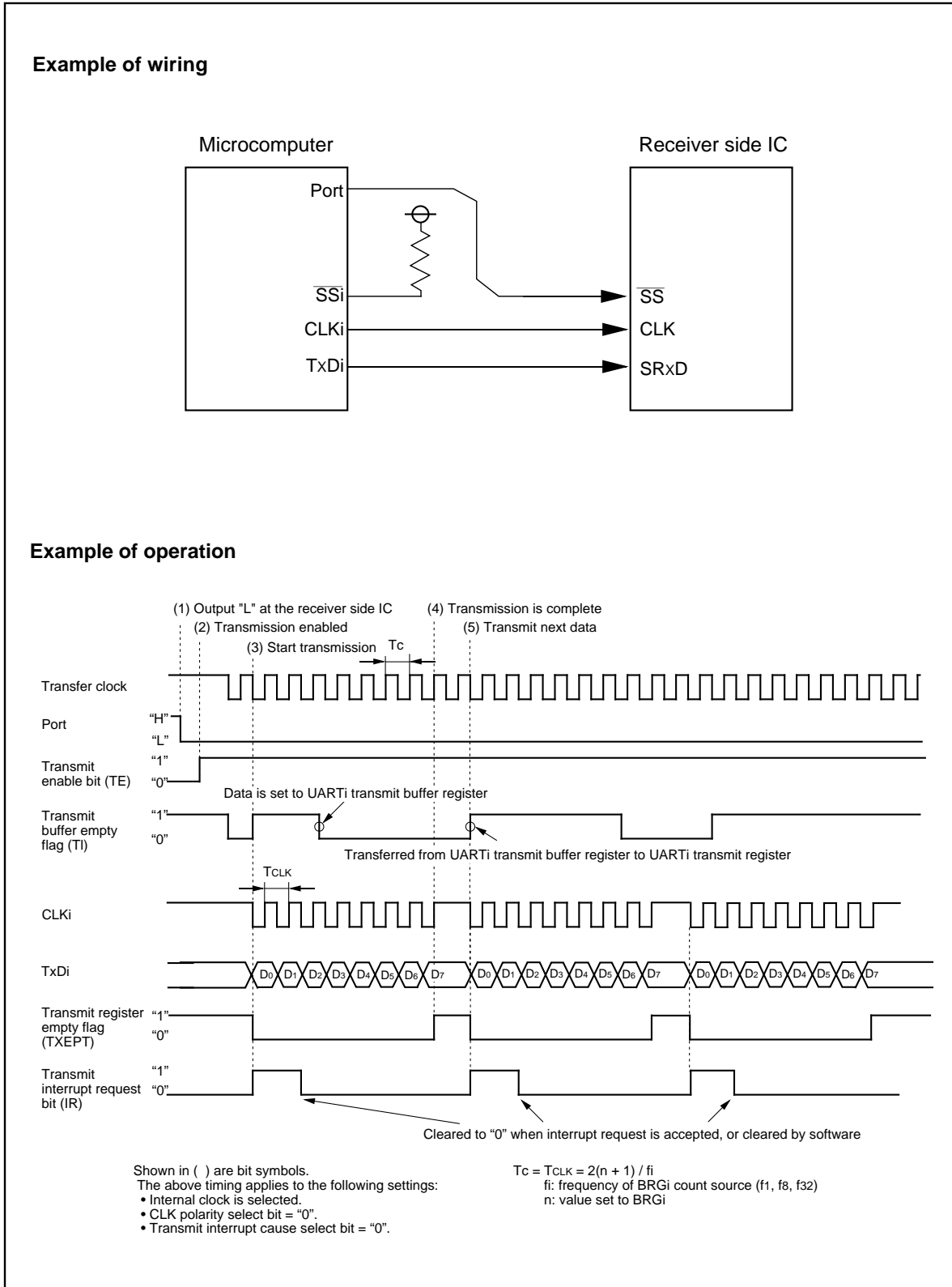


Figure 2.5.8. Operation timing of transmission in serial interface special function master mode, without clock delay



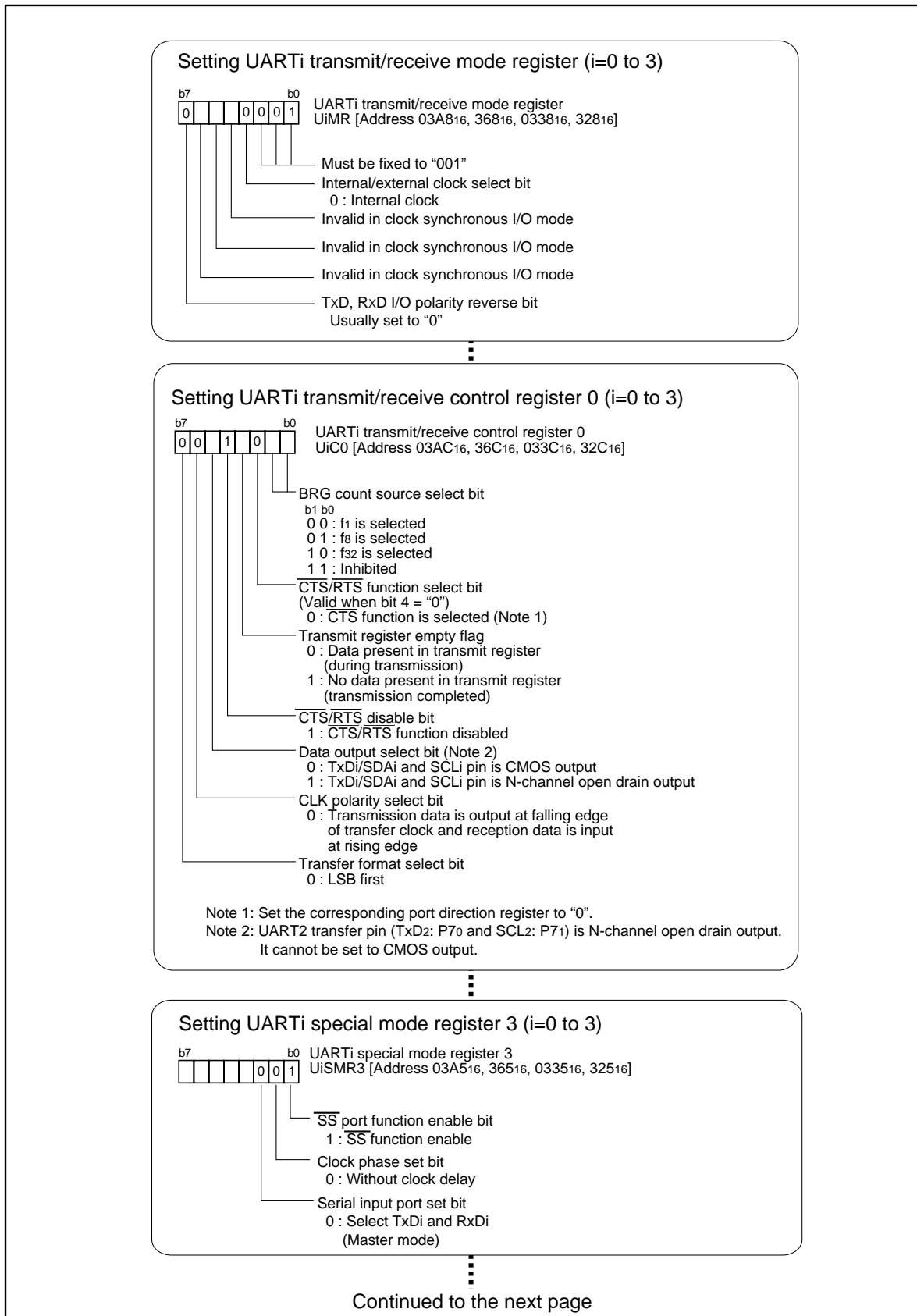


Figure 2.5.9. Set-up procedure of transmission in serial interface special function master mode, without clock delay (1)

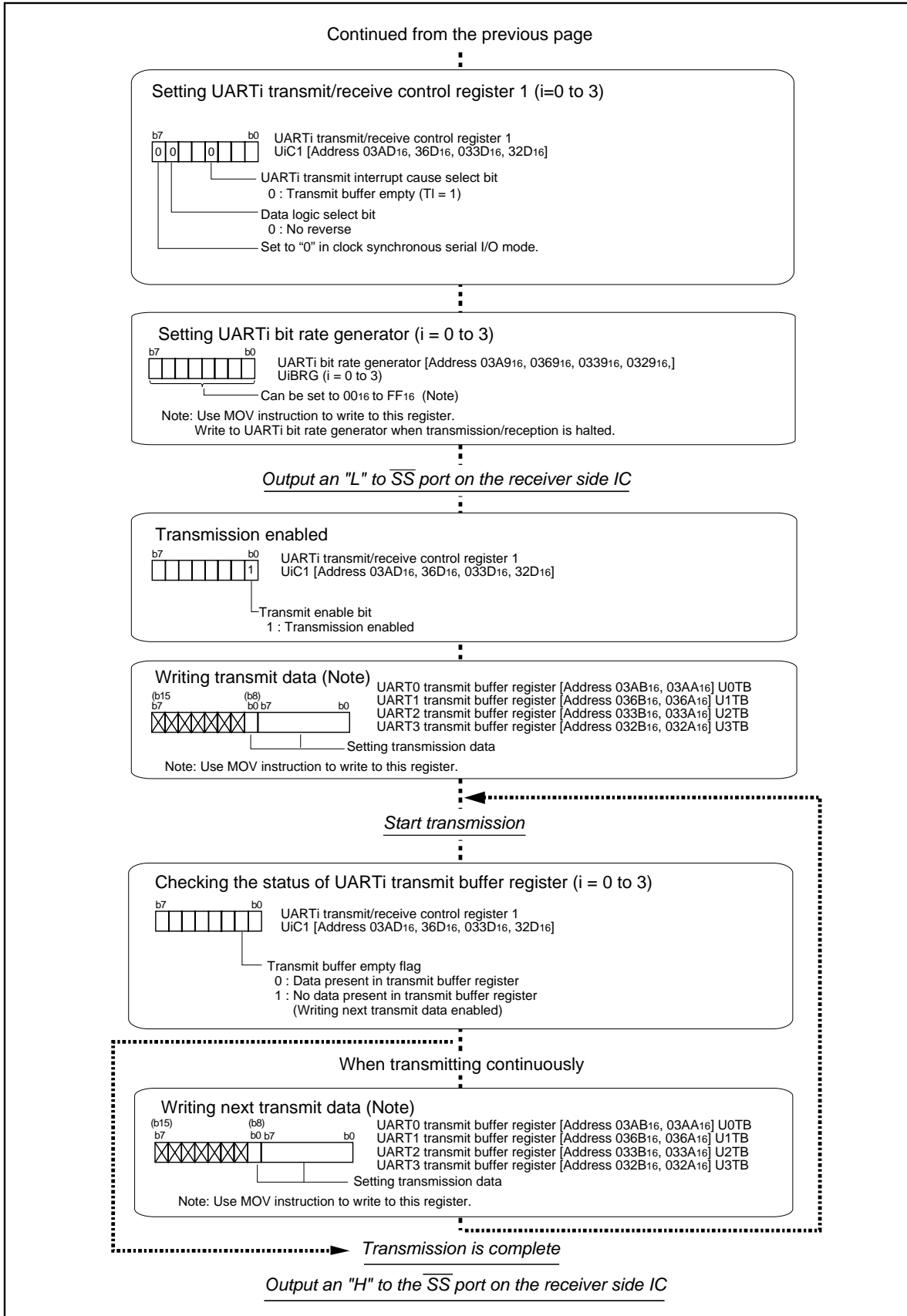


Figure 2.5.10. Set-up procedure of transmission in serial interface special function master mode, without clock delay (2)

### 2.5.3 Operation of Serial Interface Special Function (reception in master mode with clock delay)

In receiving data in serial interface special function master mode, choose functions from those listed in Table 2.5.2. Operations of the circled items are described below. Figure 2.5.11 shows the operation timing, and Figures 2.5.12 and 2.5.13 show the set-up procedures.

**Table 2.5.2. Chosed functions**

Item	Set-up		Item	Set-up	
Transfer clock source	<input type="radio"/>	Internal clock ( $f_1 / f_8 / f_{32}$ )	SSi port function enable	<input type="checkbox"/>	SSi function disabled
		External clock (CLKi pin)		<input type="radio"/>	SSi function enabled
CLK polarity	<input type="radio"/>	Output reception data at the rising edge of the transfer clock	Clock phase set	<input type="checkbox"/>	Without clock delay
		Output reception data at the falling edge of the transfer clock		<input type="radio"/>	With clock delay
Continuous receive mode	<input type="radio"/>	Disabled	Serial input port set	<input type="radio"/>	TxDi, RxDi selected (master mode)
		Enabled			STxDi, SRxDi selected (slave mode)

- Operation
- (1) Set an  $\overline{SS}$  port of the transmitter side IC to output "L" level.
  - (2) Writing dummy data to the UARTi transmit buffer register, setting the receive enable bit to "1", and the transmit enable bit to "1", makes the data receivable status ready.
  - (3) In synchronization with the first rising edge of the transfer clock, the input signal to the RxDi pin is stored in the highest bit of the UARTi receive register. Then, data is taken in by shifting right the content of the UARTi reception data in synchronization with the rising edges of the transfer clock.
  - (4) When 1-byte data lines up in the UARTi receive register, the content of the UARTi receive register is transmitted to the UARTi receive buffer register. At this time, the receive complete flag and the UARTi receive interrupt request bit goes to "1".
  - (5) The receive complete flag goes to "0" when the lower-order byte of the UARTi buffer register is read.

- Note
- Set RxDi pins' port direction register to "0".
  - Set  $\overline{SSi}$  pin to "H" level. If "L" level is input to the pin, a fault error will be generated.

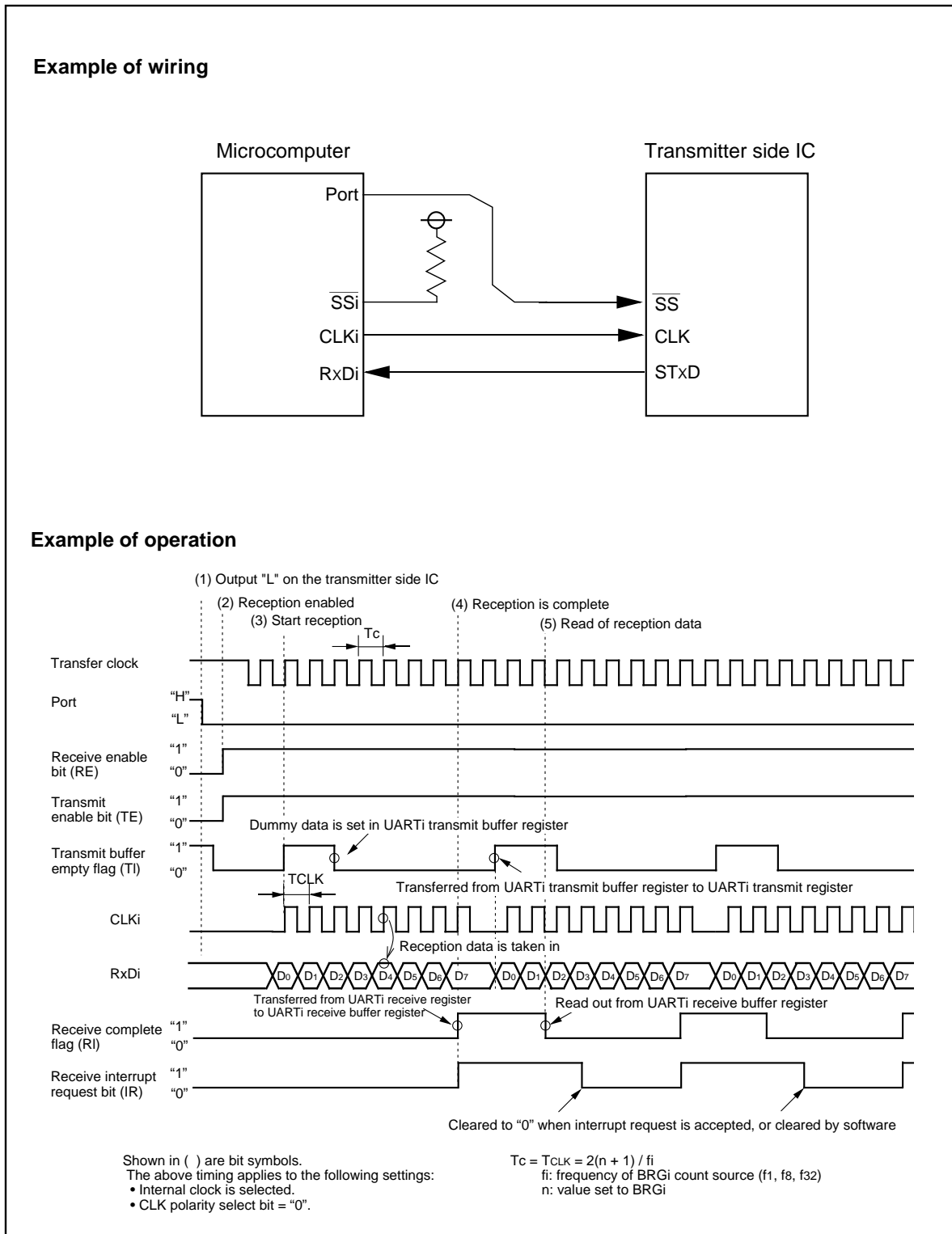


Figure 2.5.11. Operation timing of reception in serial interface special function master mode, with clock delay

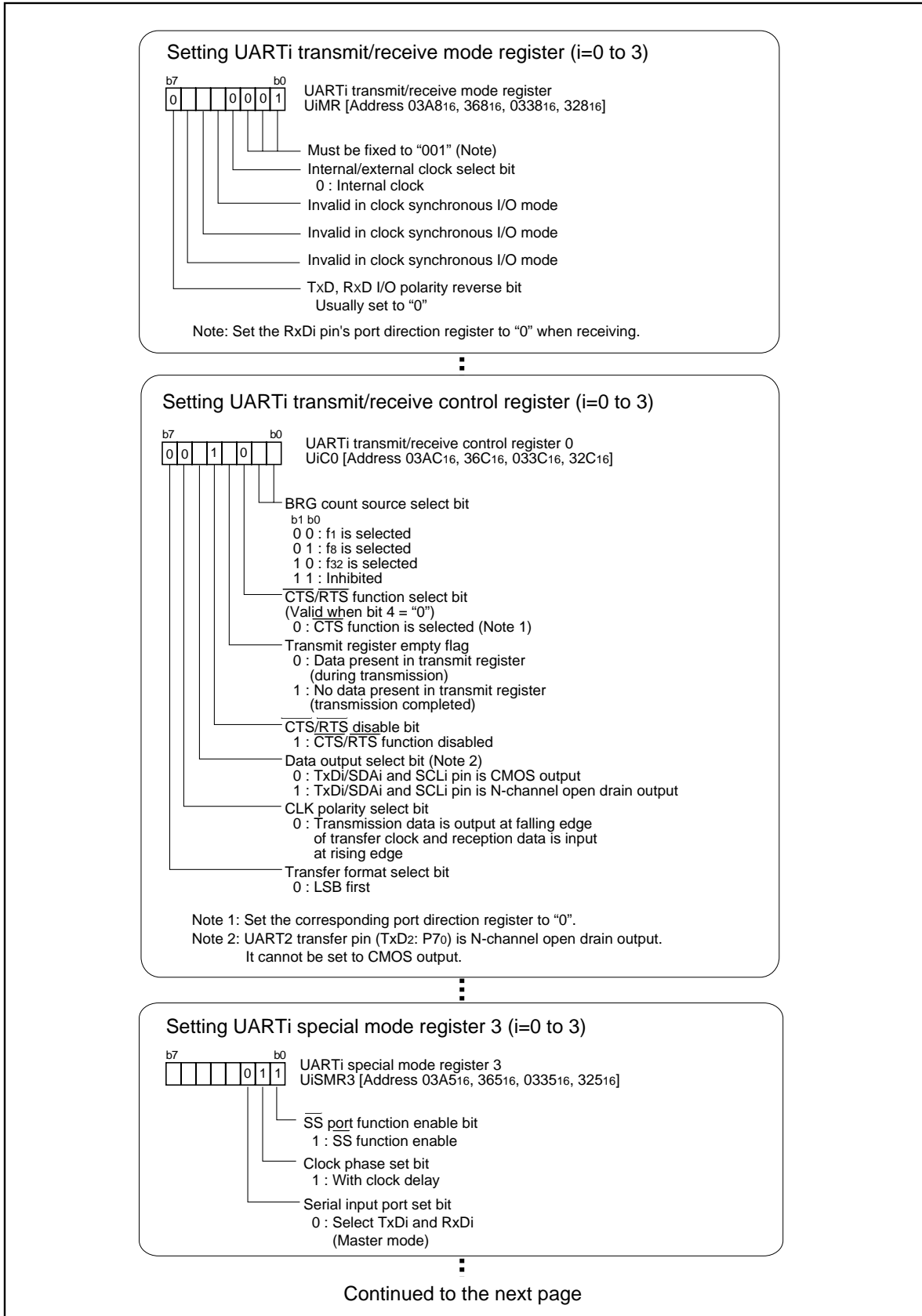


Figure 2.5.12. Set-up procedure of reception in serial interface special function master mode, with clock delay (1)

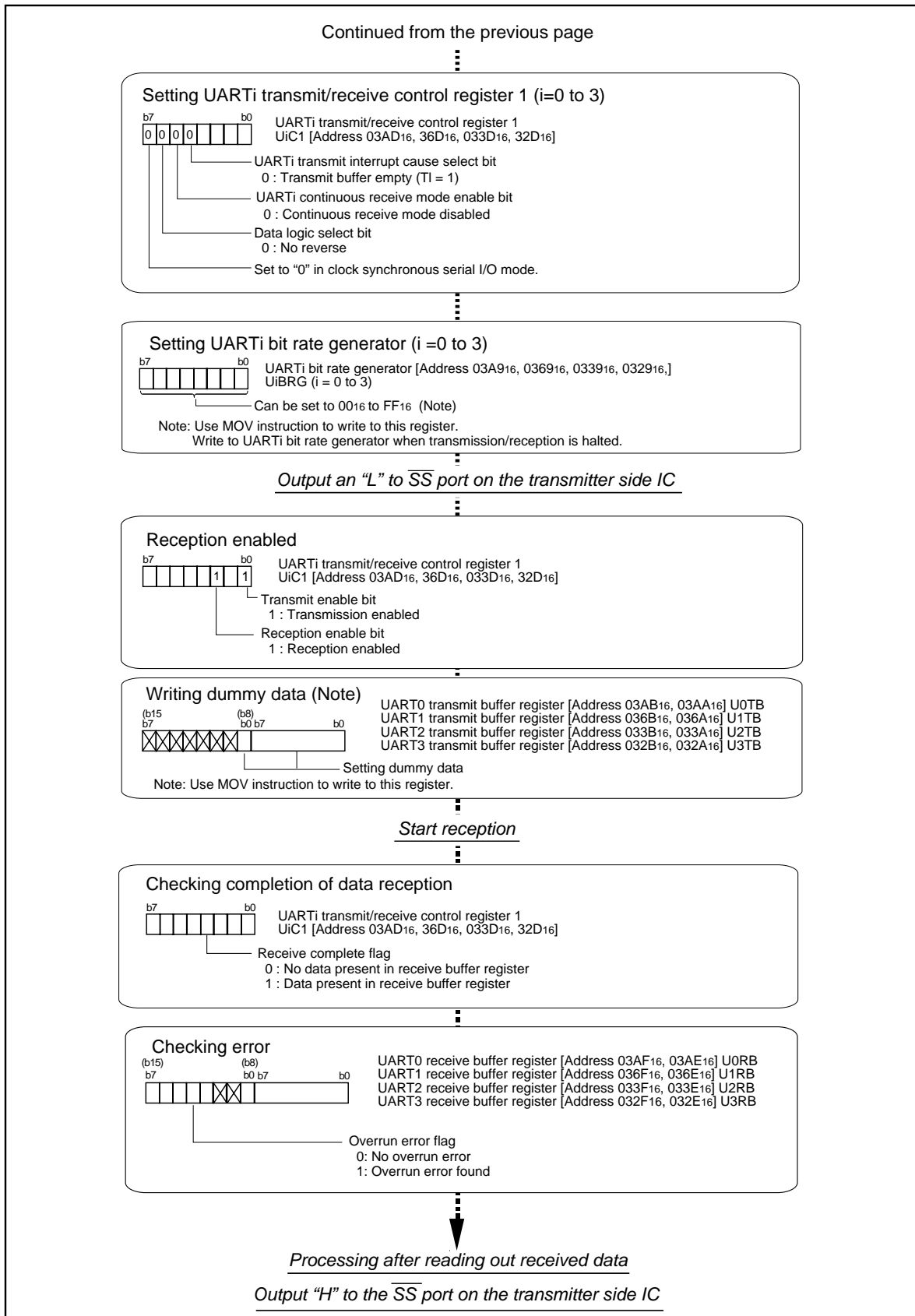


Figure 2.5.13. Set-up procedure of reception in serial interface special function master mode, with clock delay (2)

### 2.5.4 Operation of Serial Interface Special Function (transmission in slave mode without delay)

In transmitting data in serial interface special function slave mode, choose functions from those listed in Table 2.5.3. Operations of the circled items are described below. Figure 2.5.14 shows the operation timing, and Figures 2.5.15 and 2.5.16 show the set-up procedures.

**Table 2.5.3. Chosed functions**

Item	Set-up		Item	Set-up	
Transfer clock source		Internal clock (f <sub>1</sub> / f <sub>8</sub> / f <sub>32</sub> )	SSi port function enable		SSi function disabled
	○	External clock (CLKi pin)		○	SSi function enabled
CLK polarity	○	Output transmission data at the falling edge of the transfer clock	Clock phase set	○	Without clock delay
		Output transmission data at the rising edge of the transfer clock			With clock delay
Transmission interrupt factor	○	Transmission buffer empty	Serial input port set		TxDi, RxDi selected (master mode)
		Transmission complete		○	STxDi, SRxDi selected (slave mode)

- Operation
- (1) Input "L" level to an  $\overline{\text{SSi}}$  port by the output from the receiver side IC's port.
  - (2) Setting the transmit enable bit to "1" and writing transmission data to the UARTi transmit buffer register makes data transmissible status ready.
  - (3) In synchronization with the first falling edge of the transfer clock, transmission data held in the UARTi transmit buffer register is transmitted to the UARTi transmit register. At this time, the UARTi transmit interrupt request bit goes to "1". Also, the first bit of the transmission data is transmitted from the STxDi pin. Then the data is transmitted bit by bit from the lower order in synchronization with the falling edges.
  - (4) When transmission of 1-byte data is completed, the transmit register empty flag goes to "1", which indicates that transmission is completed.
  - (5) If the next transmission data is set in the UARTi transmit buffer register while transmission is in progress (before the eighth bit has been transmitted), the data is transmitted in succession.

Note

- Set CLKi pin's port direction register to "0".

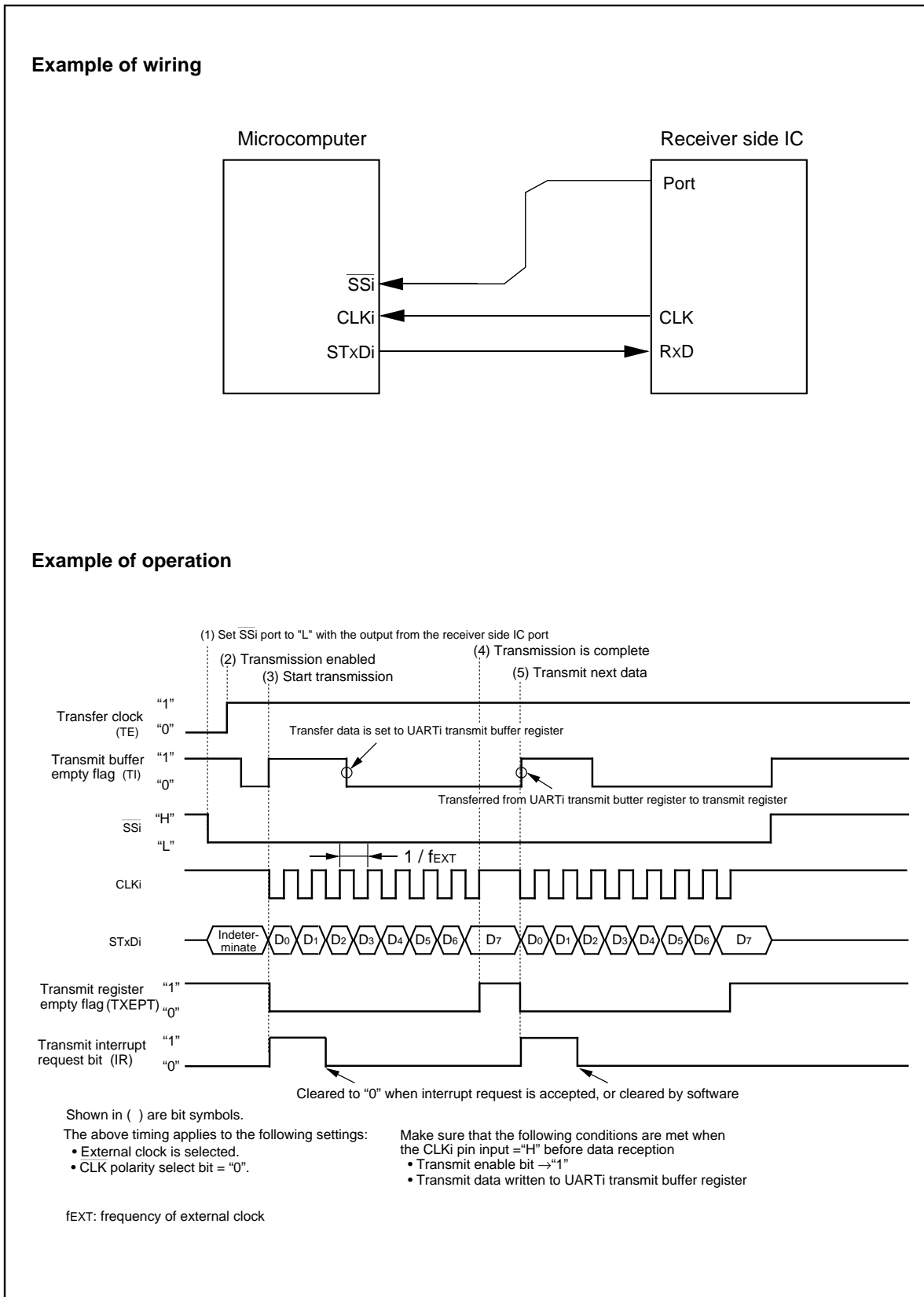


Figure 2.5.14. Operation timing of transmission in serial interface special function slave mode, without clock delay



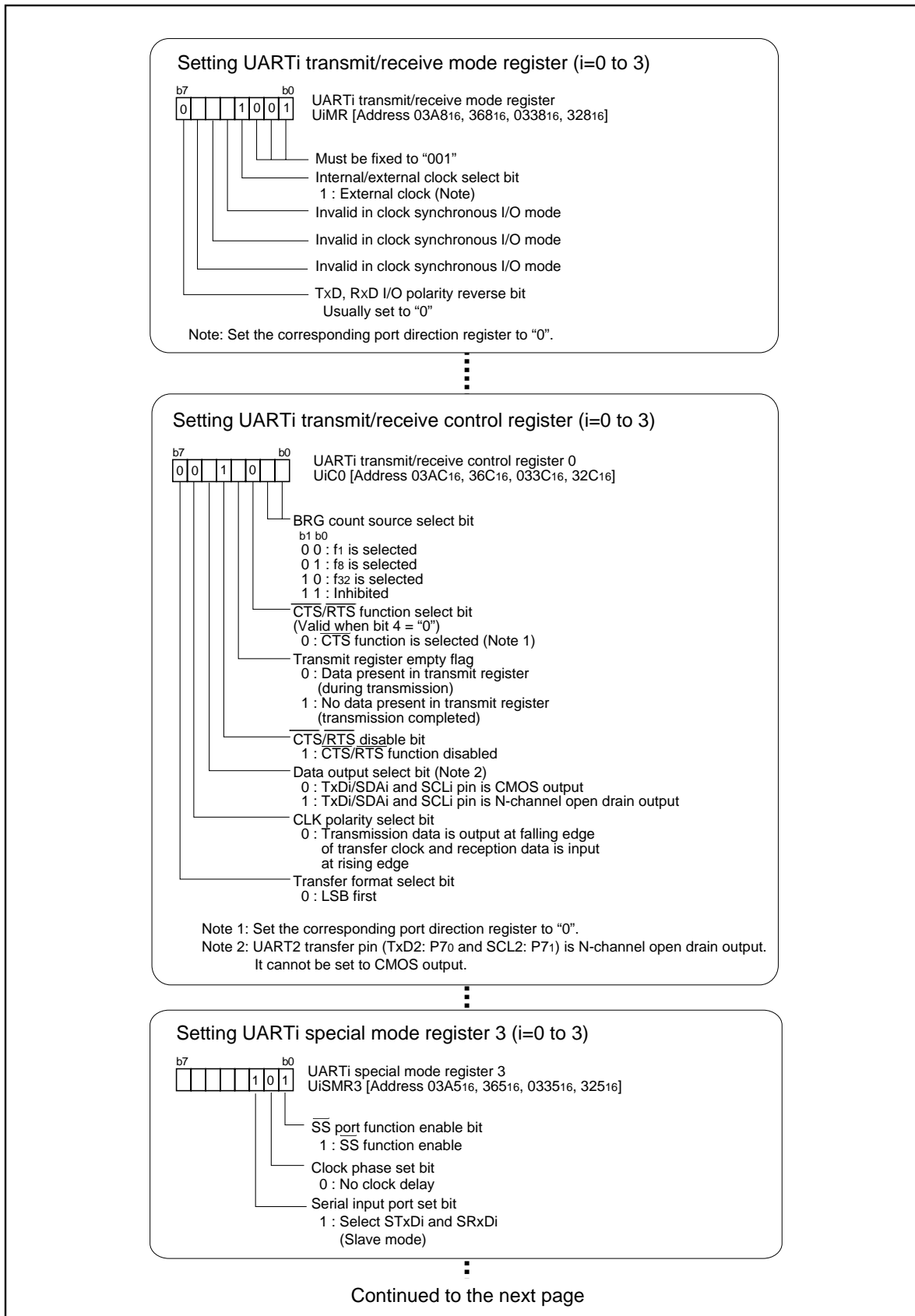


Figure 2.5.15. Set-up procedure of transmission in serial interface special function slave mode, without clock delay (1)

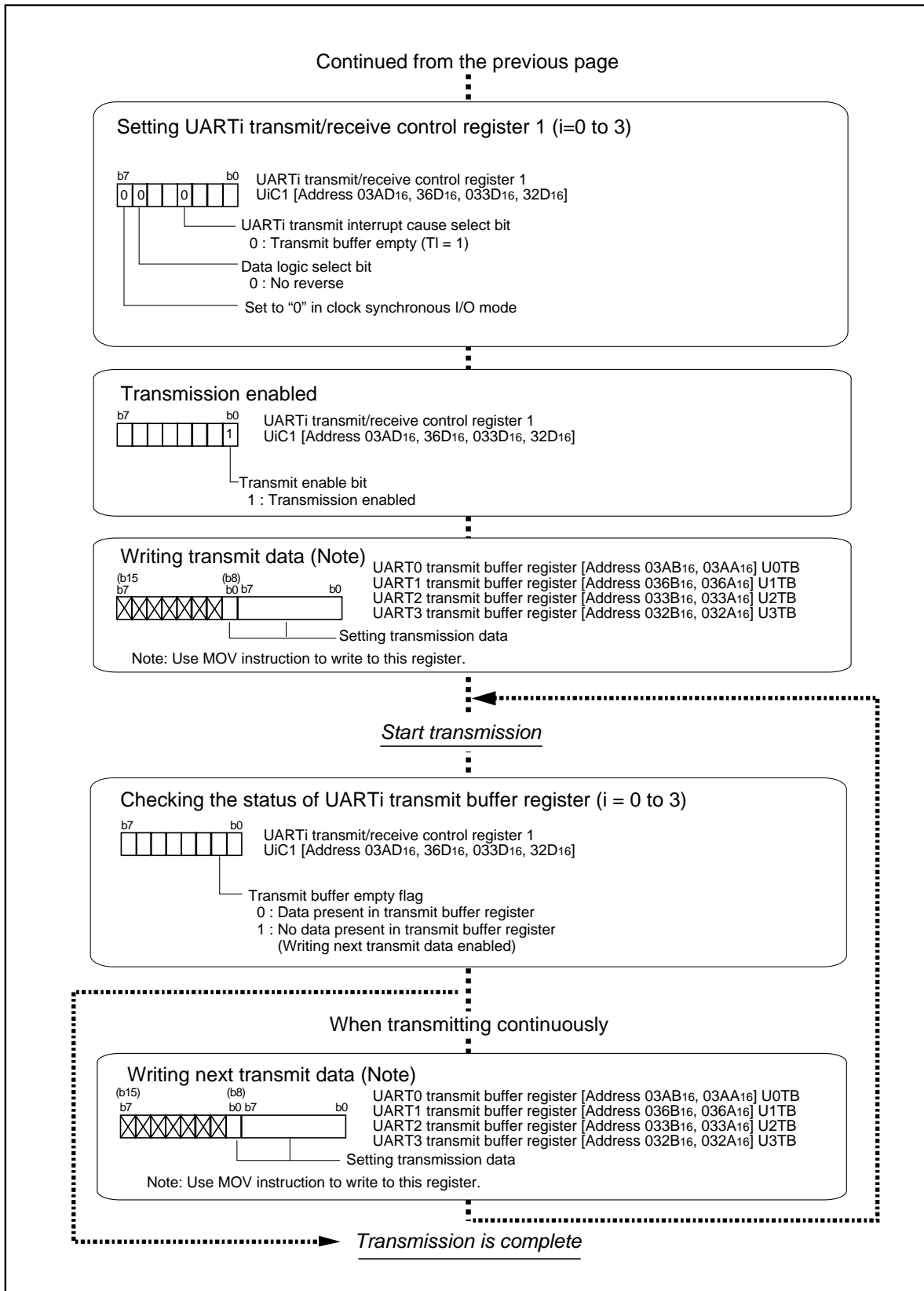


Figure 2.5.16. Set-up procedure of transmission in serial interface special function slave mode, without clock delay (2)

### 2.5.5 Operation of Serial Interface Special Function (reception in slave mode with clock delay)

In receiving data in serial interface special function slave mode, choose functions from those listed in Table 2.5.4. Operations of the circled items are described below. Figure 2.5.17 shows the operation timing, and Figures 2.5.18 and 2.5.19 show the set-up procedures.

**Table 2.5.4. Chosed functions**

Item	Set-up	Item	Set-up
Transfer clock source	Internal clock ( $f_1 / f_8 / f_{32}$ )	SSi port function enable	SSi function disabled
	○ External clock (CLKi pin)		○ SSi function enabled
CLK polarity	○ Output reception data at the rising edge of the transfer clock	Clock phase set	Without clock delay
	Output reception data at the falling edge of the transfer clock		○ With clock delay
Continuous receive mode	○ Disabled	Serial input port set	TxDi, RxDi selected (master mode)
	Enabled		○ STxDi, SRxDi selected (slave mode)

- Operation
- (1) An  $\overline{\text{SSi}}$  port is input "L" level which outputs from the transmitter side IC port.
  - (2) Writing dummy data to the UARTi transmit buffer register, setting the receive enable bit to "1", and the transmit enable bit to "1", makes the data receivable status ready.
  - (3) In synchronization with the first rising edge of the transfer clock, the input signal to the SRxDi pin is stored in the highest bit of the UARTi receive register. Then, data is taken in by shifting right the content of the UARTi reception data in synchronization with the rising edges of the transfer clock.
  - (4) When 1-byte data lines up in the UARTi receive register, the content of the UARTi receive register is transmitted to the UARTi receive buffer register. At this time, the receive complete flag and the UARTi receive interrupt request bit goes to "1".
  - (5) The receive complete flag goes to "0" when the lower-order byte of the UARTi buffer register is read.

Note

- Set CLKi and SRxDi pins' port direction register to "0".

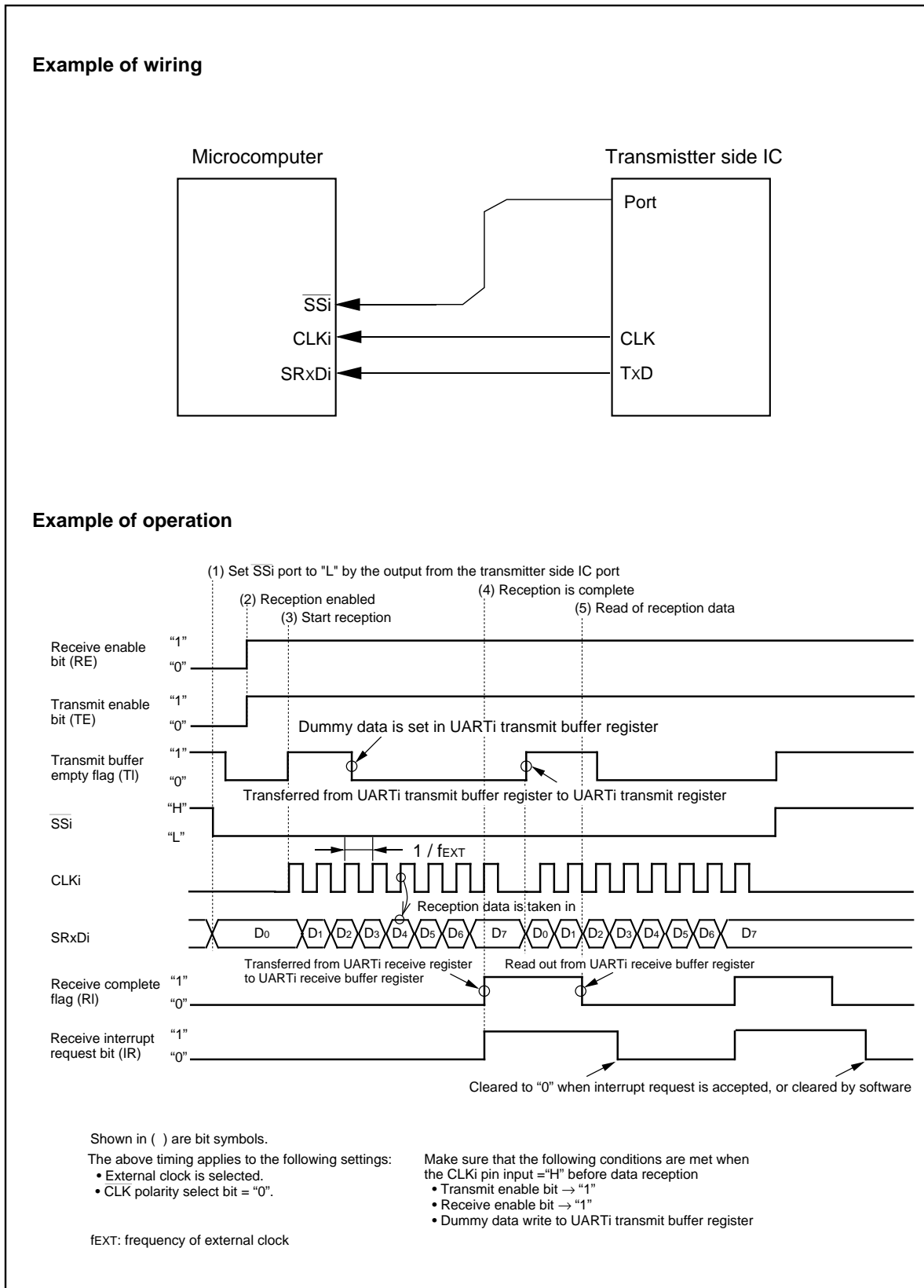


Figure 2.5.17. Operation timing of reception in serial interface special function slave mode, with clock delay

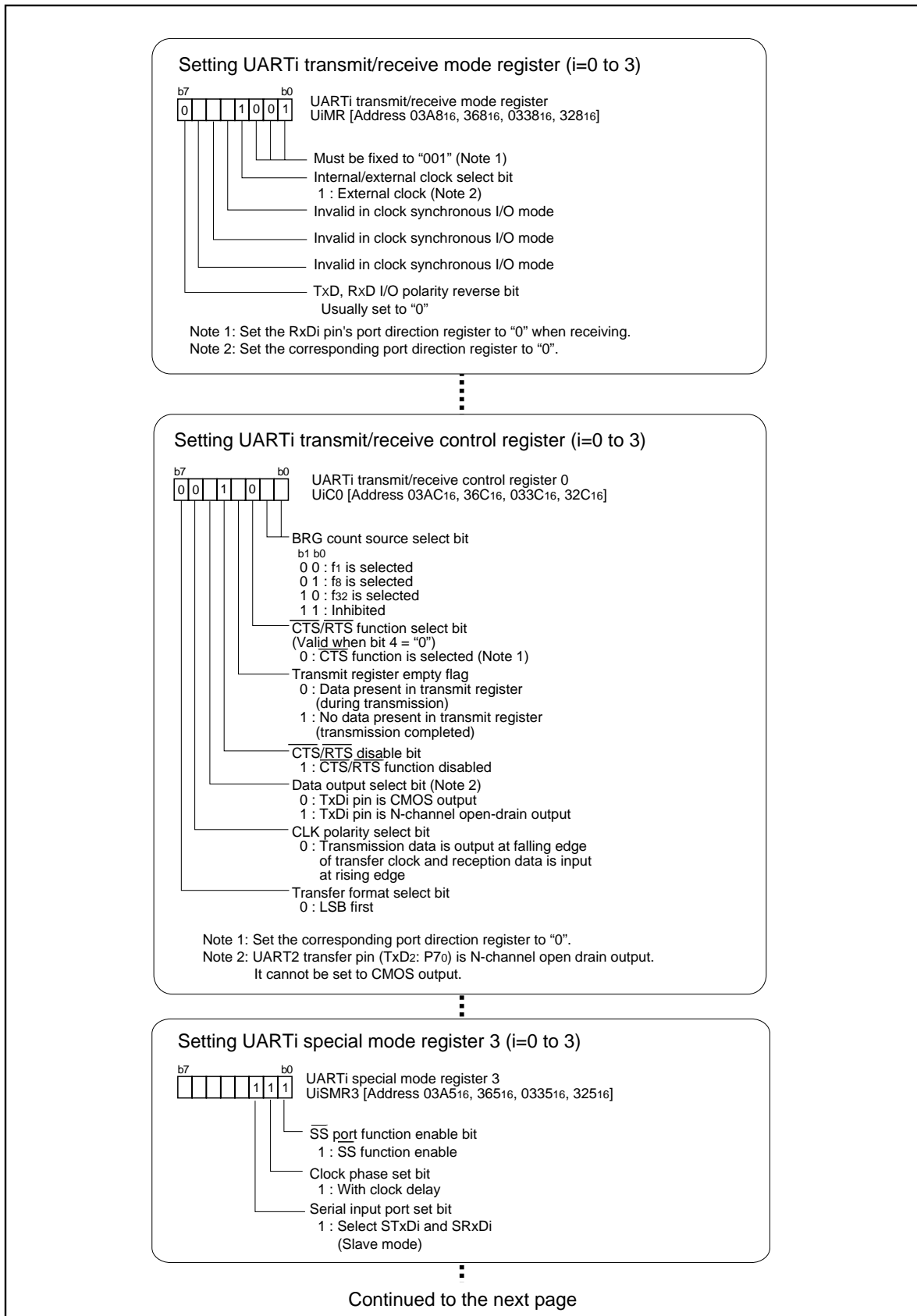


Figure 2.5.18. Set-up procedure of reception in serial interface special function slave mode, with clock delay (1)

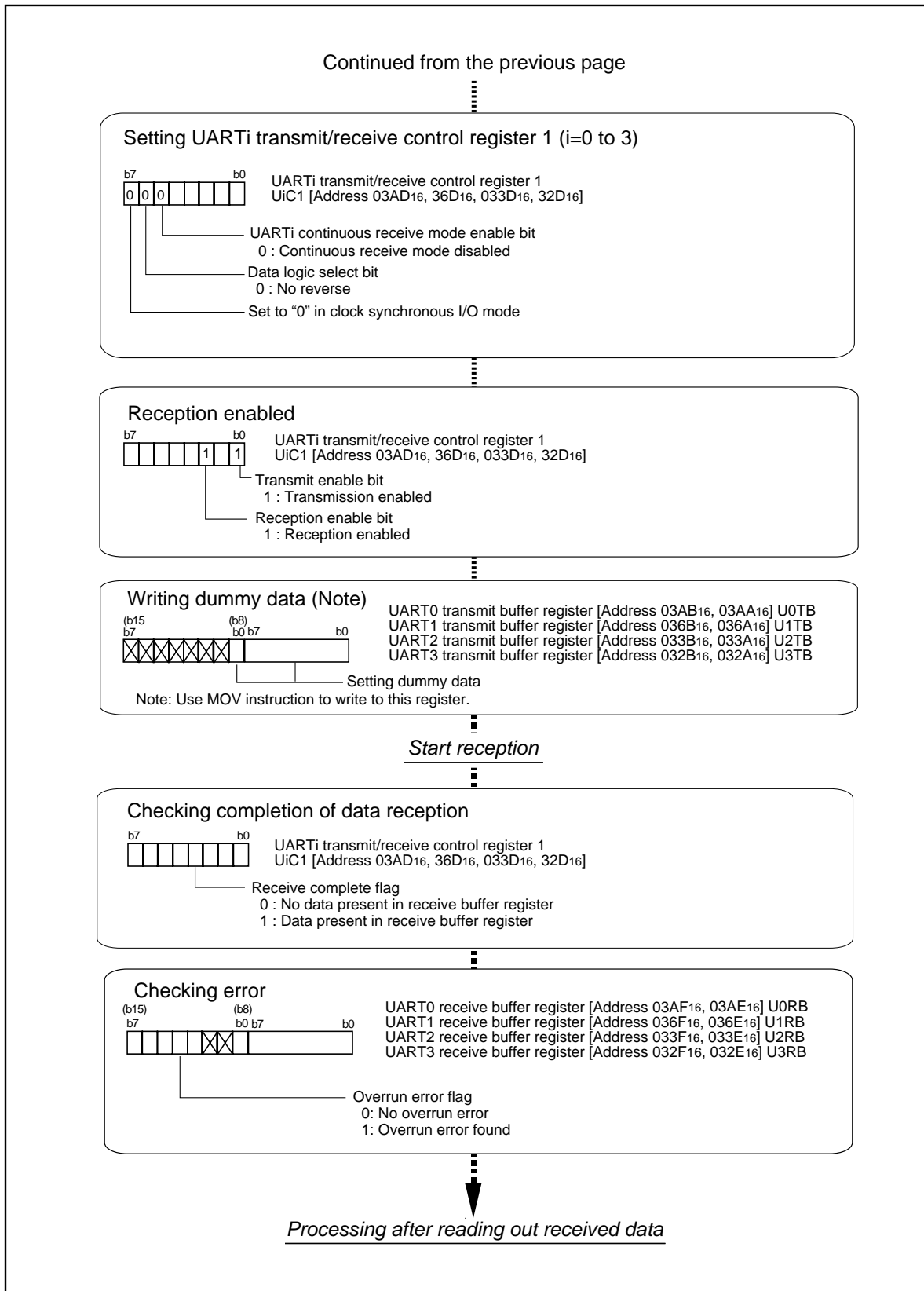


Figure 2.5.19. Set-up procedure of reception in serial interface special function slave mode, with clock delay (2)

## 2.6 Serial sound interface

### 2.6.1 Overview

The Serial Sound Interface (SSI) is a synchronous serial data interface used primarily for transferring digital audio data. The bus of the 30245 Serial Sound Interface has four lines:

- Continuous serial clock (SCK)
- Word (channel) select (WS)
- Serial data out (XMIT)
- Serial data in (RX)

A basic Serial Sound Interface-based communication system has two Serial Sound Interfaces and a master controller, which generates both SCK and WS. The Serial Sound Interface that generates the control signals (SCK and WS) operates as a master, and the Serial Sound Interface that receives the external control signals operates as slave. The 30245 Sound Serial Interface can operate only as a slave. The transmitter/receiver must change channels on every WS transition. Through separate transmit and receive pins, simultaneous transmit and receive can be performed in synchronization with the same SCK and WS signals.

The following is an overview of the Serial Sound Interface.

#### ● **Transmission/reception format**

The data path is designed to work with the data format of the USB audio class device specifications.

The transmitter/receiver must change channels on every WS transition. The number of SCKs within a WS high/low period is set as the channel width. The channel width can be selected from among 16 bits, 24 bits, and 32 bits using channel width select bits 0 and 1 (bits 4 and 5 of the SSIMR0 register). If the number of SCKs exceeds the channel width, the receiver will stop receiving data until the next WS edge and the transmitter continues to transmit "0". However, if the number of the SCKs falls short of the channel data width, both the transmitter and the receiver will immediately switch to transmit and receive, respectively, of the next channel data item.

#### ● **Select function**

In the Serial Sound Interface function, the following features can be selected.

##### **(1) Rate feedback function**

When used with the USB interface, the Serial Sound Interface can count the number of WSs or SCKs per USB frame. The count value is loaded into the serial sound interface xRF register (x = 0 or 1) on the falling edge of each SOF pulse generated by the USB core. The SOF pulse is a frame delimiter used in USB communication. The value read from the register is the count from the immediately preceding USB frame.

##### **(2) Channel width selection function**

Channel widths of 32, 24 and 16 bits can be used to transmit and receive data. The width can be selected by the channel width select bits 0 and 1 in serial sound interface mode register 0.

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

This function is to choose whether to transmit/receive data from bit 1 or bit 7. This is valid when the transfer data length is 8 bits long.

Choose either of the following:

- LSB first      Data is transmitted/received from bit 0.
- MSB first      Data is transmitted/received from bit 7.

**(4) Multiple receive format select function**

If the number of SCKs in a WS high/low period is less than the channel width, the data can be placed either MSB or LSB justified.

**(5) SCK polarity select function**

This function selects whether transmit and receive data are synchronized to the rising or falling edge of WS. This is selected by the SCK polarity bit in the serial sound interface x mode register 1.

**(6) WS polarity select function**

This function is to transmit/receive data synchronized to the rising edge or the falling edge of WS. This is selected by the WS polarity select bit in the Serial Sound Interface x mode register 1.

**(7) WS delay select function**

Either of the following modes may be selected for the channel change timing:

- Normal WS mode  
WS transitions one SCK period before a channel change. (The channel changes one SCK period after WS transitions.)
- Delayed WS mode  
WS transitions concurrently with a channel change.

**● Input to the Serial Sound Interface function and the corresponding direction register**

When inputting an external signal to the Serial Sound Interface, set the corresponding port's direction register as input.

**● Serial Sound Interface function-related pins**

- |     |                     |                      |
|-----|---------------------|----------------------|
| (1) | SCLK0 and SCLK1pins | Transfer clock input |
| (2) | WS0 and WS1pins     | Channel clock input  |
| (3) | RX0 and RX1pins     | Data input           |
| (4) | XMIT0 and XMIT1pins | Data output          |

**● Serial Sound Interface function-related register**

Figure 2.6.1 shows a memory map of the frequency synthesizer-related registers. Figures 2.6.2 and 2.6.3 show the configuration of Serial Sound Interface function related-registers, respectively.

Set the port's direction register appropriately and disable the clock synchronous serial and the UART which share the port.



0310 <sub>16</sub>	Serial Sound Interface 0 mode register 0 (SS0MR0)
0311 <sub>16</sub>	Serial Sound Interface 0 mode register 1 (SS0MR1)
0312 <sub>16</sub>	Reserved
0313 <sub>16</sub>	Reserved
0314 <sub>16</sub>	Serial Sound Interface 0 transmit buffer register (SS0TXB)
0315 <sub>16</sub>	
0316 <sub>16</sub>	Serial Sound Interface 0 receive buffer register (SS0RXB)
0317 <sub>16</sub>	
0318 <sub>16</sub>	Serial Sound Interface 0 RF register (SS0RF)
0319 <sub>16</sub>	
031A <sub>16</sub>	Reserved
0370 <sub>16</sub>	Serial Sound Interface 1 mode register 0 (SSI1MR0)
0371 <sub>16</sub>	Serial Sound Interface 1 mode register 1 (SSI1MR1)
0372 <sub>16</sub>	Reserved
0373 <sub>16</sub>	Reserved
0374 <sub>16</sub>	Serial Sound Interface 1 transmit buffer register (SSI1TXB)
0375 <sub>16</sub>	
0376 <sub>16</sub>	Serial Sound Interface 1 receive buffer register (SSI1RXB)
0377 <sub>16</sub>	
0378 <sub>16</sub>	Serial Sound Interface 1 RF register (SSI1RF)
0379 <sub>16</sub>	

Figure 2.6.1. Memory map of Serial Sound Interface function-related registers

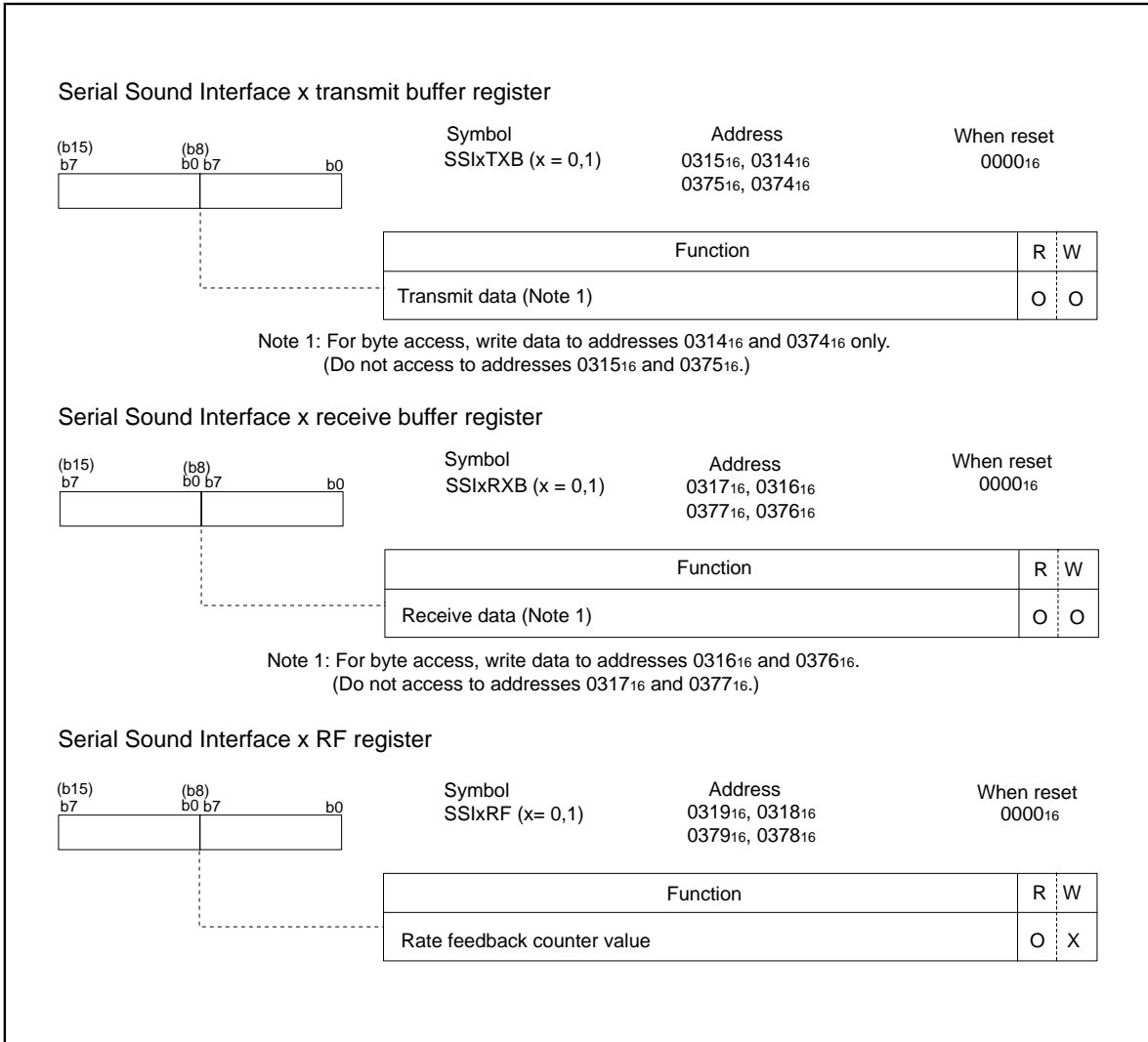


Figure 2.6.2. Serial Sound Interface function-related registers (1)

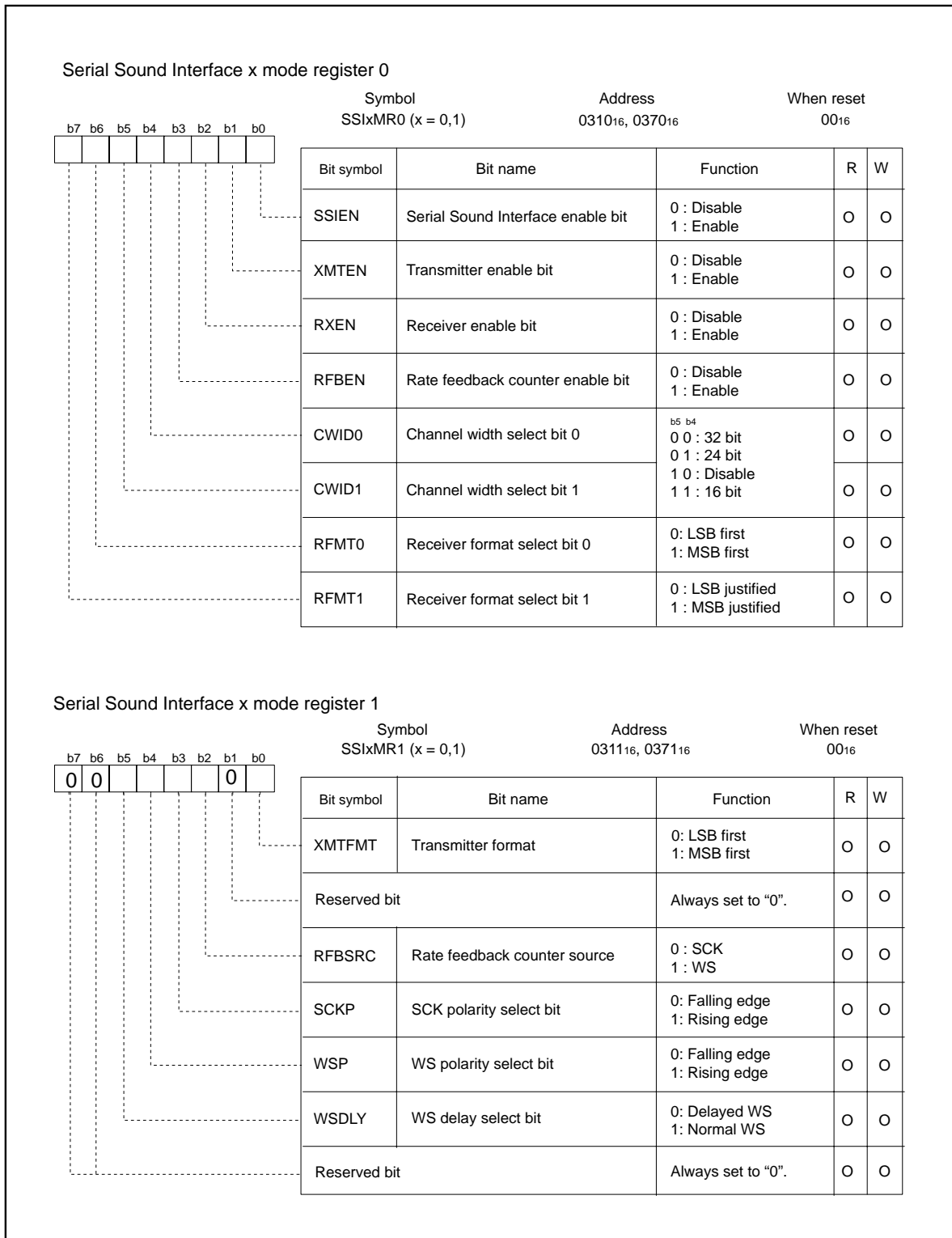


Figure 2.6.3. Serial Sound Interface function-related registers (2)

In the case of the USB audio class, the following stream is output.

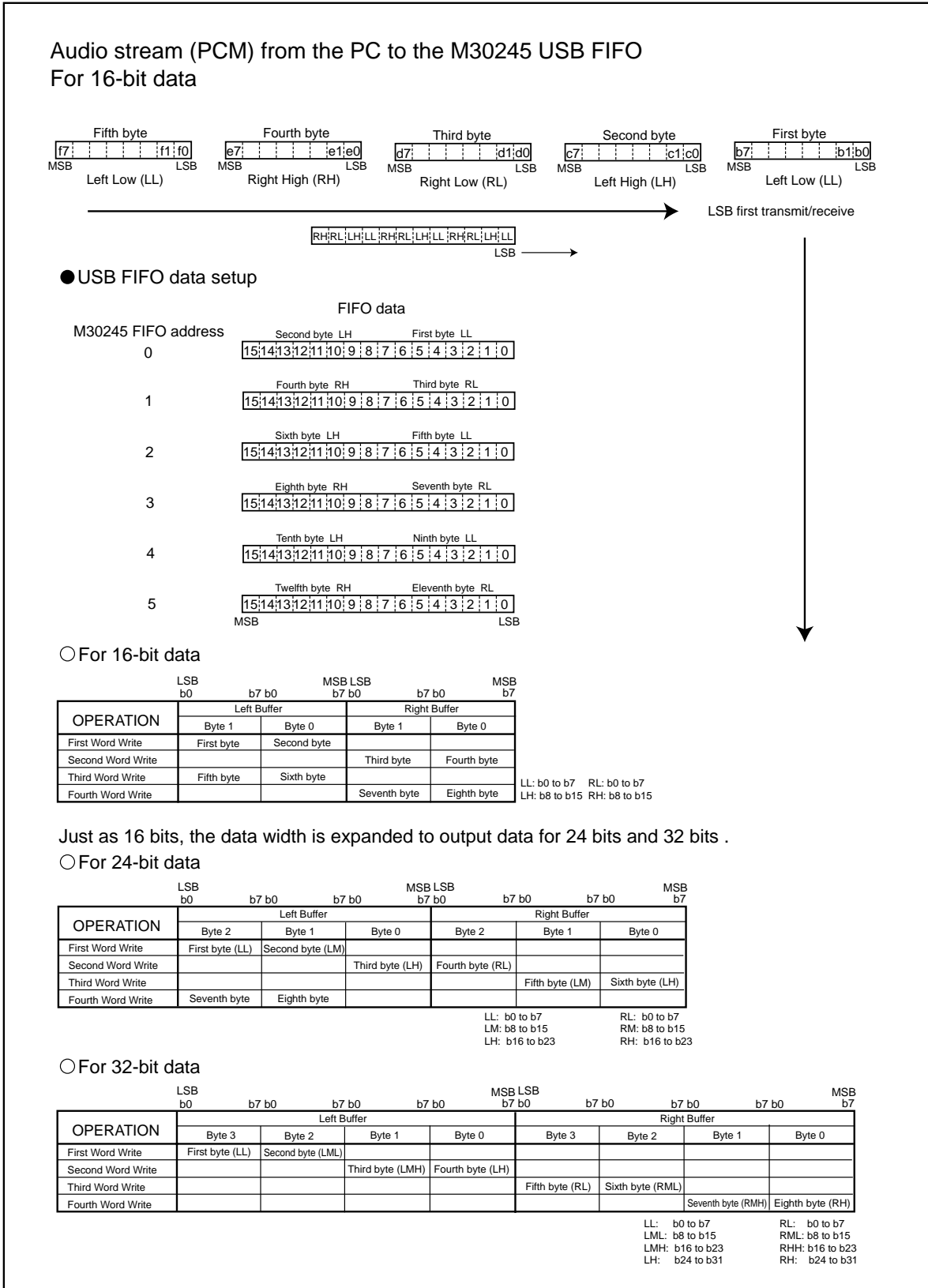


Figure 2.6.4. Example of Audio stream (PCM) from the PC to the M30245 USB FIFO

## 2.6.2 Example of Serial Sound Interface operation

When using Serial Sound Interface (SSI), the DMA is recommended for reading and writing data quickly from the receive buffer to the transmit buffer. A programming example using DMA is shown below Figure 2.6.5 shows an example of Serial Sound Interface transmit timing, and Figure 2.6.6 shows an example of Serial Sound Interface receive timing.

```

/**** Serial Sound Interface initialization routine *****/
Serial Sound Interface is initialized by disabling.
The DMA to use is also initialized.
In this example, one DMA is used for each Serial Sound Interface input and output.

*****/

ssi1mr1 = 0x00;          /* SSI STOP */
ssi1mr0 = 0x00;          /* SSI STOP */
dm0sl = 0x00;           /* DMA0 STOP */
dm0con = 0x00;          /* DMA0 STOP */
dm1sl = 0x00;           /* DMA1 STOP */
dm1con = 0x00;          /* DMA1 STOP */

/**** Setting example *****/
Audio transmission can be set before or after audio reception
The DMA to use is set.
DMA0 = audio data transmission and DMA1= audio data reception

*****/

dm0ic = 0x06;           /* DMA completion interrupt enabled */
dm0sl = 0x0e;           /* DMA0 factor = SSI 1 transmit */
sar0 = (unsigned long)&txb_buffer; /* source address: buffer RAM, etc. */
dar0 = (unsigned long)&ssi1txb;    /* destination address: SSI 1 transmit buffer */
tcr0 = txb_counter;    /* DMA0 transfer cycle setting */
dm1ic = 0x06;           /* DMA completion interrupt enabled */
dm1sl = 0x0a;           /* DMA1 factor = SSI 1 receive */
sar1 = (unsigned long)&ssi1rx;     /* source address: SSI 1 receive */
dar1 = (unsigned long)&rx_buffer;  /* destination address: buffer RAM, etc. */
tcr1 = rx_counter;     /* DMA1 transfer cycle setting */

/**** Activation processing routine *****/
DMA activation. DMA0 can be activated before or after DMA1.

*****/

dm0con = 0x18;          /* DMA0 start [16bit, SRC = inc, single] */
dm1con = 0x28;          /* DMA1 start [16bit, DES = inc, single] */

/*****Serial Sound Interface is stopped at this point.
Serial Sound Interface is enabled and audio data transmission/reception starts.*****/

(Continued to the next page.)

```

(Continued from the previous page.)

```
/***** Serial Sound Interface activation routine for 16 bits *****/
#ifdef OUT_Q_BIT_NO_16
    ssi1mr0 = 0x01;          /* SSIEN = 1 */
    ssi1mr0 = 0xf1;        /* 16bit / MSB justified */
    ssi1mr1 = 0x21;        /* SCK neg, WS neg    MSB first normal */
    ssi1mr0 = 0xf7;        /* 16bit / Tx enable, Rx enable MSB justified */
#endif

/***** Serial Sound Interface activation routine for 24 bits *****/
#ifdef OUT_Q_BIT_NO_24
    ssi1mr0 = 0x01;          /* SSIEN = 1 */
    ssi1mr0 = 0xd1;        /* 24bit / MSB justified */
    ssi1mr1 = 0x21;        /* SCK neg, WS neg    MSB first normal */
    ssi1mr0 = 0xd7;        /* 24bit / Tx enable, Rx enable MSB justified */
#endif

/***** Serial Sound Interface activation routine for 32 bits *****/
#ifdef OUT_Q_BIT_NO_32
    ssi1mr0 = 0x01;          /* SSIEN = 1 */
    ssi1mr0 = 0xc1;        /* 32bit / MSB justified */
    ssi1mr1 = 0x21;        /* SCK neg, WS neg    MSB first normal */
    ssi1mr0 = 0xc7;        /* 32bit / Tx enable, Rx enable MSB justified */
#endif
```

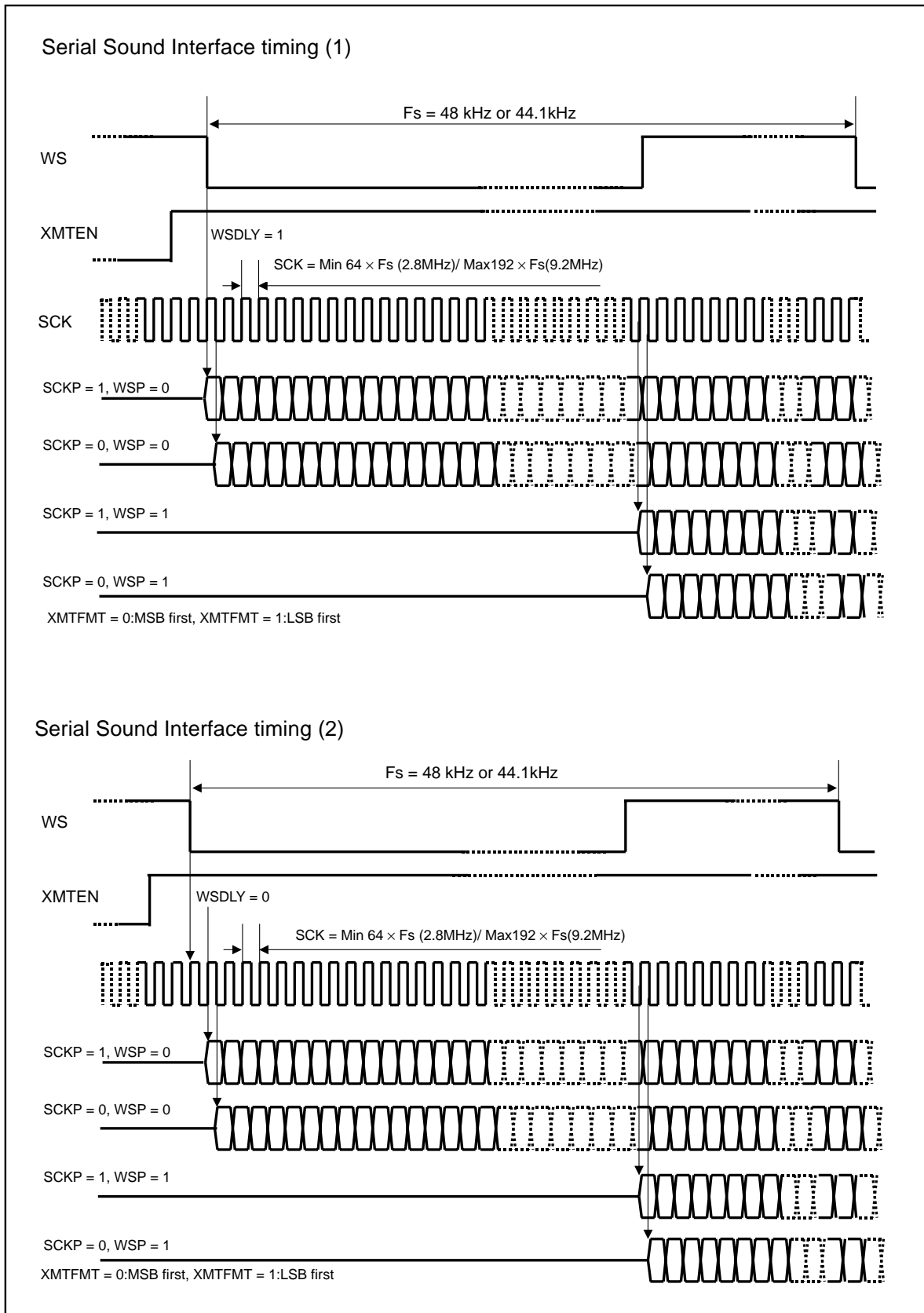


Figure 2.6.5. Example of Serial Sound Interface transmit timing

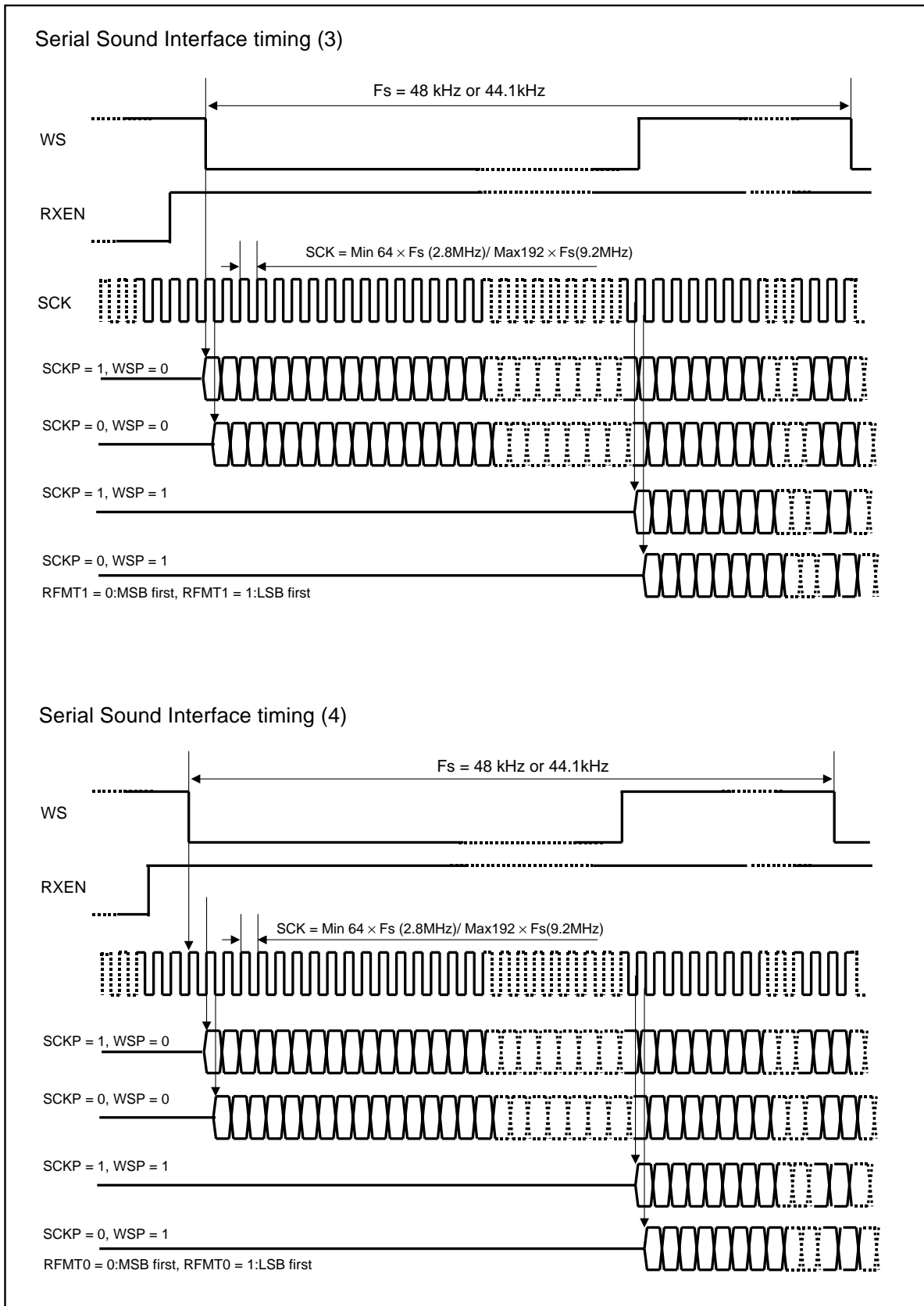


Figure 2.6.6. Example of Serial Sound Interface receive timing



### 2.6.3 Precautions for Serial Sound Interface

Description For flash memory version SSI transmission data must be latched as the following timing by a receiver.

- SCKP=0 (falling edge) : within 3 BCLK cycles from the rising edge of SCK
- SCKP=1 (rising edge) : within 3 BCLK cycles from the falling edge of SCK

## 2.7 Frequency synthesizer (PLL)

This paragraph explains the registers setting method and the notes related to the frequency synthesizer (PLL circuit).

### 2.7.1 Overview

The frequency synthesizer generates the 48MHz clock that is necessary for the USB block and the fSYN clock. These clocks are a multiple of the external input standard clock f(XIN). Figure 2.7.1 shows the frequency synthesizer circuit block diagram.

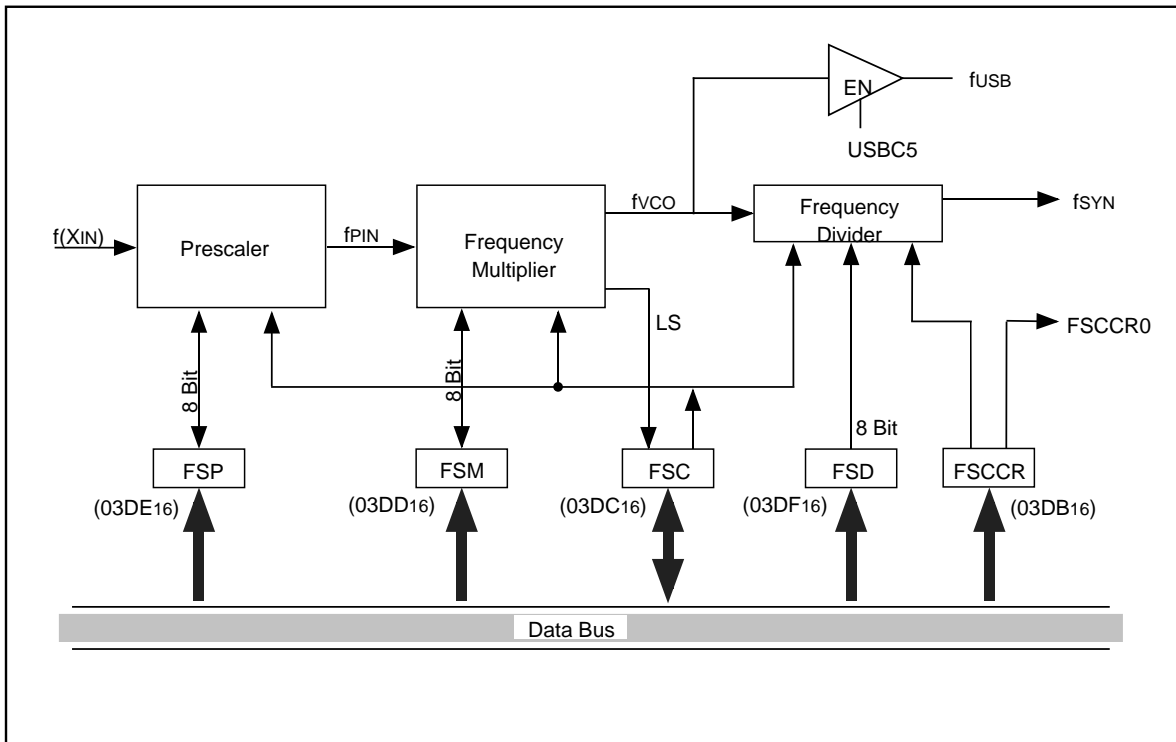


Figure 2.7.1. Frequency synthesizer circuit block diagram

#### (1) Related Registers

Figure 2.7.2 shows a memory location diagram for the frequency synthesizer related registers; Figures 2.7.3 and 2.7.4 show the composition of the frequency synthesizer related registers.

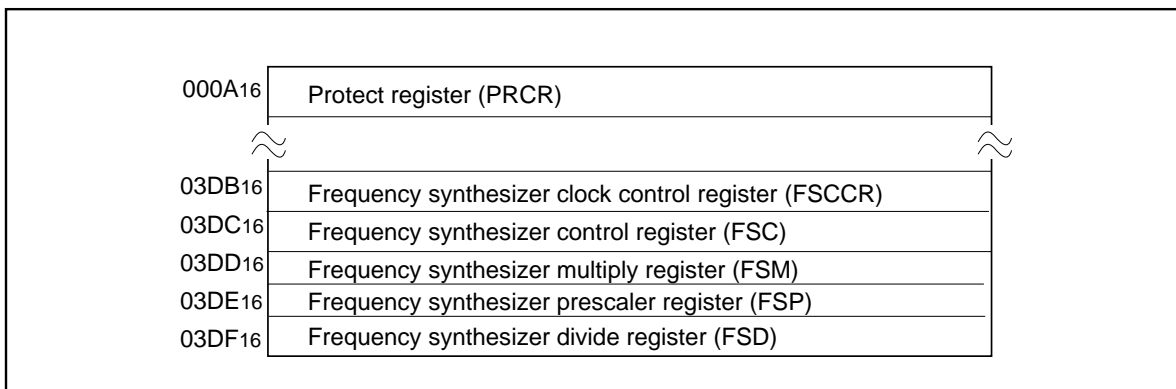


Figure 2.7.2. Memory map of frequency synthesizer related registers

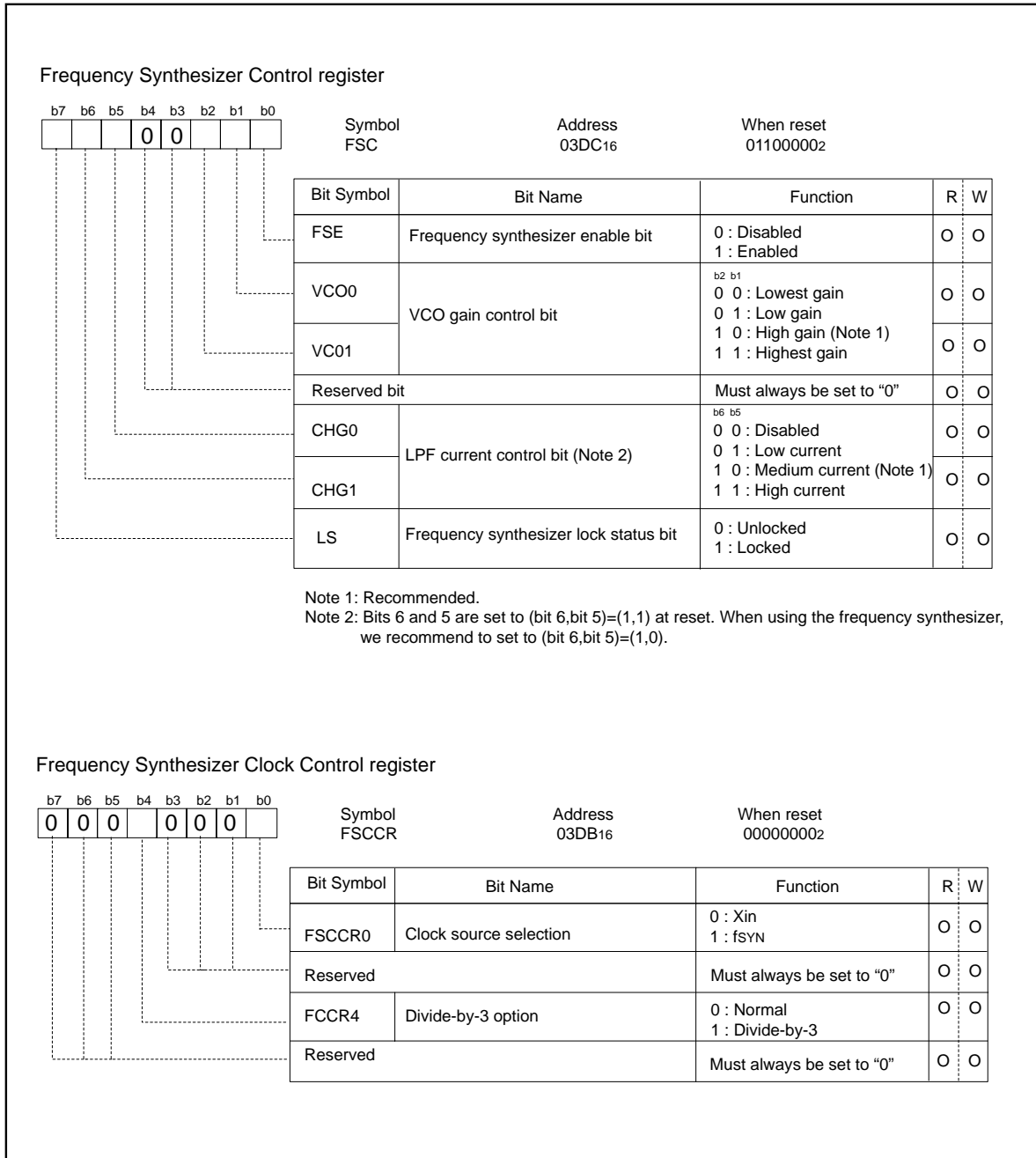


Figure 2.7.3. Frequency synthesizer registers (1)

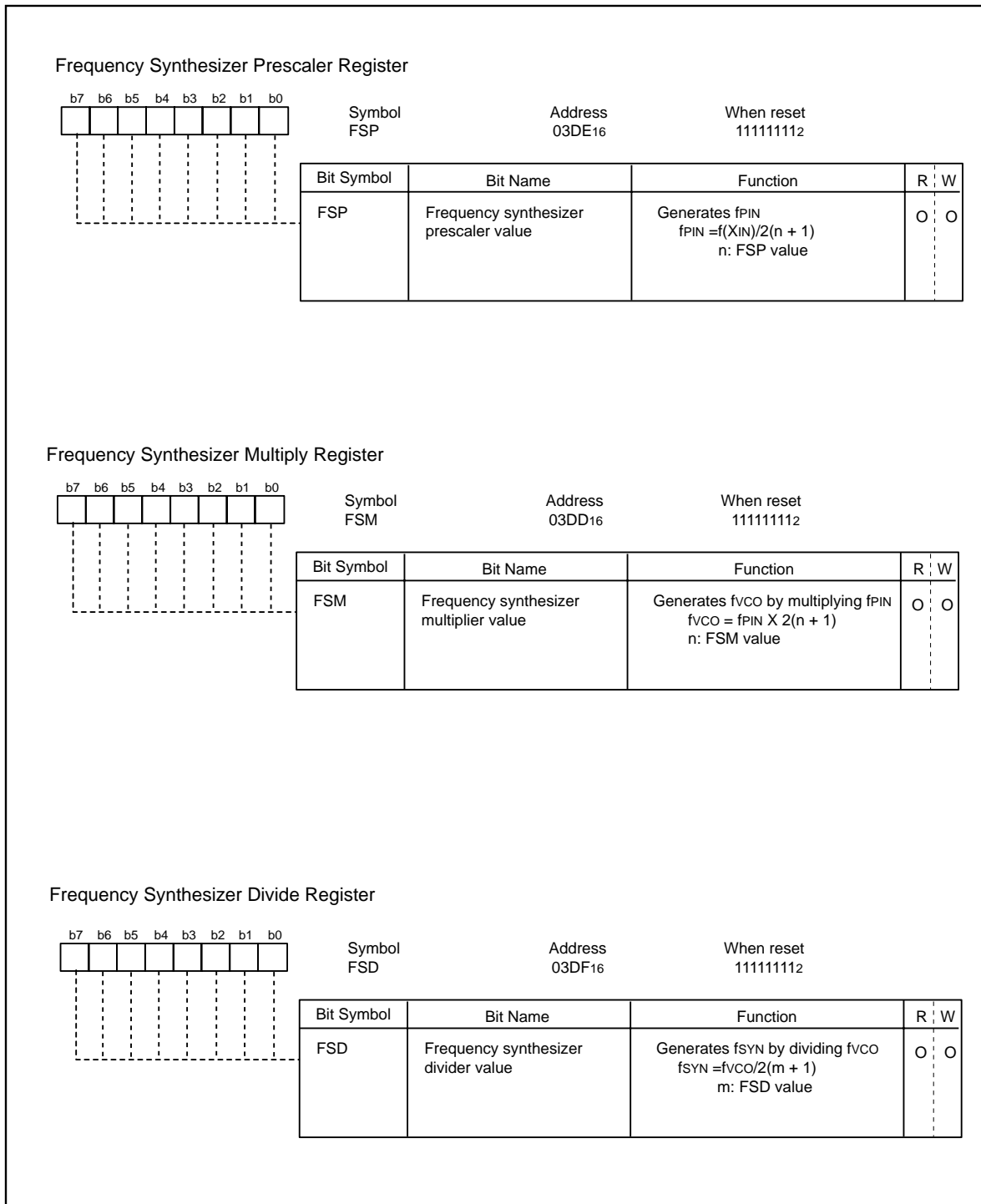


Figure 2.7.4. Frequency synthesizer registers (2)

### 2.7.2 Operation of frequency synthesizer

The following explains how to setup after hardware reset. Table 2.7.1 to 2.7.3 show frequency synthesizer related registers setting examples.

- Operation
- (1) Cancel the protect register.
  - (2) Set the frequency synthesizer related registers to generate the 48MHz clock that is necessary for the fUSB.
  - (3) Enable the frequency synthesizer by setting frequency synthesizer control register.
  - (4) The protect register should be set to write disabled. A 3ms wait is necessary.
  - (5) The frequency synthesizer locked status bit should be checked. It is necessary to recheck after a wait of 0.1ms if it is "0" (unlocked).
  - (6) Enable USB clock.
  - (7) After waiting four cycles of the  $\phi$  or greater, the USB enable bit should be set to "1".  
A minimum delay of 250ns is needed before performing any other USB related registers read/write operations.

#### ●Prescaler

Clock  $f(XIN)$  is prescaled down by the frequency synthesizer prescaler register (FSP) to generate  $f_{PIN}$ . When the frequency synthesizer prescaler register is set at 255, the prescaler is disabled and  $f_{PIN} = f(XIN)$ . Table 2.7.1 shows some examples of how the frequency synthesizer prescaler register is set.  
 $f_{PIN} = f(XIN) / 2(n+1)$  n: FSP value

Note: The value of  $f_{PIN}$  should not be set below 1 MHz.

**Table 2.7.1. Example of setting the frequency synthesizer prescaler register (FSP)**

$f_{PIN}$	FSP Value Dec (Hex)	$fX_{(IN)}$
12 MHz	255 (FF <sub>16</sub> )	12.00 MHz
1 MHz	7 (07 <sub>16</sub> )	16.00 MHz
1 MHz	5 (05 <sub>16</sub> )	12.00 MHz
2 MHz	3 (03 <sub>16</sub> )	16.00 MHz
2 MHz	2 (02 <sub>16</sub> )	12.00 MHz
3 MHz	1 (01 <sub>16</sub> )	12.00 MHz
6 MHz	0 (00 <sub>16</sub> )	12.00 MHz

### ●Frequency Multiplier

$f_{VCO}$  is generated via the Frequency Synthesizer Multiplier register (FSM: address 03DD<sub>16</sub>). When the Frequency Multiplier register is set to 255, multiplication is disabled and  $f_{VCO} = f_{PIN}$ . The value of  $n$  should be set so that  $f_{VCO}$  becomes 48MHz. Table 2.7.2 shows some examples of how the frequency synthesizer multiply register is set.

$$f_{VCO} = f_{PIN} \times 2^{(n+1)} \quad n: \text{FSM value}$$

**Table 2.7.2. Example of Setting the Frequency Multiplier Register (FSM)**

$f_{PIN}$	FSM Value Dec (Hex)	$f_{VCO}$
1 MHz	23 (17 <sub>16</sub> )	48 MHz
2 MHz	11 (0B <sub>16</sub> )	48 MHz
4 MHz	5 (05 <sub>16</sub> )	48 MHz
6 MHz	3 (03 <sub>16</sub> )	48 MHz
8 MHz	2 (02 <sub>16</sub> )	48 MHz
12 MHz	1 (01 <sub>16</sub> )	48 MHz

### ●Frequency Divider

Clock  $f_{SYN}$  is a divided down version of  $f_{VCO}$ .  $f_{SYN}$  is generated via the frequency synthesizer divide register (FSD). When the frequency synthesizer divider register is set to 255, division is disabled and  $f_{SYN} = f_{VCO}$ . Table 2.7.3 shows some examples of how the frequency synthesizer division register is set.

$$f_{SYN} = f_{VCO} / 2^{(m+1)} \quad m: \text{FSD value}$$

Note 1: Set  $f_{SYN}$  to 12MHz or lower.

**Table 2.7.3. Example of Setting the frequency synthesizer divide register (FSD)**

$f_{VCO}$	FSD Value Dec (Hex)	$f_{SYN}$
48 MHz	1 (01 <sub>16</sub> )	12.00 MHz
	2 (02 <sub>16</sub> )	8.00 MHz
	2 (02 <sub>16</sub> )	16.00 MHz (Note 1)
	3 (03 <sub>16</sub> )	6.00 MHz
	127 (7F <sub>16</sub> )	187.50 kHz

Note 1:  $f_{SYN} = f_{VCO} / (m+1)$  when FSCCR4=1 and m=2.

### 2.7.3 Precautions for Frequency synthesizer

- (1) Bits 6 and 5 of frequency synthesizer control register are set to (bit6, bit5)=(1, 1) at reset. When using the frequency synthesizer, we recommended to set to (bit6, bit5)=(1, 0).
- (2) Set f<sub>SYN</sub> to 12 MHz or lower.
- (3) The value of f<sub>PIN</sub> should not be set below 1 MHz.
- (4) When the frequency synthesizer is enabled, do not use the output of the frequency synthesizer until after a 2~5ms delay. That will stabilize the output. Also, after the frequency synthesizer has been enabled, because the output is temporarily (2-5ms) unstable, the contents of none of the registers should be changed.
- (5) When using the frequency synthesizer, connect a low pass filter to the LPF terminal.
- (6) The following setup for the frequency synthesizer should be done after hardware reset.
  1. Cancel the protect register.
  2. Set the frequency synthesizer related registers to generate the 48MHz clock that is necessary for the f<sub>USB</sub>.
  3. Enable the frequency synthesizer by setting frequency synthesizer control register.
  4. The protect register should be set to write disabled. A 3ms wait is necessary.
  5. The frequency synthesizer locked status bit should be checked. It is necessary to recheck after a wait of 0.1ms if it is "0" (unlocked).
  6. Enable USB clock.
  7. After waiting four cycles of the f or greater, the USB enable bit should be set to "1".  
A minimum delay of 250ns is needed before performing any other USB related registers read/write operations.

## 2.8 USB function

### 2.8.1 Overview

The USB function control unit of the M30245 group is compliant with USB2.0 specification and supports Full-Speed transfer. USB2.0 specification defines the following four kinds of transfer types:

- Control Transfer
- Isochronous Transfer
- Interrupt Transfer
- Bulk Transfer

The USB function control unit is provided with 9 endpoints including endpoint 0, endpoints 1 to 4 OUT (receive) and endpoints 1 to 4 IN (transmit), each of which having an FIFO. The endpoint 0 can apply only to the control transfer (same in transfer form as the bulk transfer); while endpoints 1 to 4 IN/OUT can apply to the bulk transfer, isochronous transfer and interrupt transfer. The size and the starting position of endpoints 1 to 4 IN/OUT FIFO can be set according to the user's system (The size and the starting position of endpoint 0 IN/OUT FIFO are fixed). Further, when the double buffer mode is enabled, the buffer which has twice as much as the set size is available for the IN/OUT FIFO. When the continuous receive/transmit mode is enabled, data can be transferred at a high speed (in bulk transfer).

The USB related interrupts include USB suspend interrupt, USB resume interrupt, USB reset interrupt, USB endpoint 0 interrupt, USB function interrupt, and USB SOF interrupt. The USB function control unit can control the state transition of the USB device based on these interrupt requests.

#### (1) Transfer Type

The USB specifications largely concern 2 types, including the one for the host side (PC/Hub) to control the connected peripheral devices and the other for the peripheral device side (Device) which is connected to the real machine. Further, depending on the number of data dealt with in the device, the peripheral devices which require more data (concerning image, audio, etc.) at one time have the communication specification with high transfer speed (12Mbps) called Full-Speed function, while the peripheral devices (keyboard, mouse) which require less data have the communication specification with low transfer speed (1.5Mbps) called Low-Speed function. Further, a communication specification with even higher speed is available which is called Hi-Speed function (480Mbps).

These communication specifications are each determined by device class of the peripheral devices and the transfer type to be used is determined for each peripheral device.

The M30245 group supports the following four kinds of transfer types:

##### ● Control Transfer

This transfer is used for communication of request-response form (bi-directional) in aperiodic communication which occurs suddenly. This is mainly used at the time of setup. As all the devices have to be supported in the standard device request, control transfer is supported without any exception for the devices compliant with USB.

##### ● Bulk Transfer

This transfer is used to transfer the data which delay does not cause any problem in aperiodic communication which occurs suddenly. This is used to transfer large quantities of data.

Hardware detects errors in order to guarantee transfer data. A request for retransmission is issued by detecting of any error. For example, they include the output data from printer and image data of scanner.



### ● Interrupt Transfer

This transfer is used to notify the host of aperiodic and low-frequency data from the device. For example, they include the notification of out of paper in printer and data concerning devices such as the mouse and the keyboard.

### ● Isochronous Transfer

This transfer is used for continuous and periodic communication. Once the communication path is established, a transfer rate is guaranteed with a limited delay. The maximum size of transfer data is specified by the endpoint, which is read by the host as the configuration data of the device. Based on this data, transfer within the frame is scheduled and the bus time required for transfer of the maximum size of data is secured with preference. Although the band width and the transfer rate of data transfer are guaranteed, retransmission is not executed even if an error exists in the transfer. This is used for streaming data such as animation and audio data which requires real time.

## (2) Communication Protocol

Host CPU has the initiative for the entire USB communication. Even when data are transmitted to the host from the device, the host gives the right of use to the device before data are transmitted. The host, in order to process multiple transfers simultaneously, schedules each transfer in packet unit within a frame of 1ms interval. The frames image is shown below:

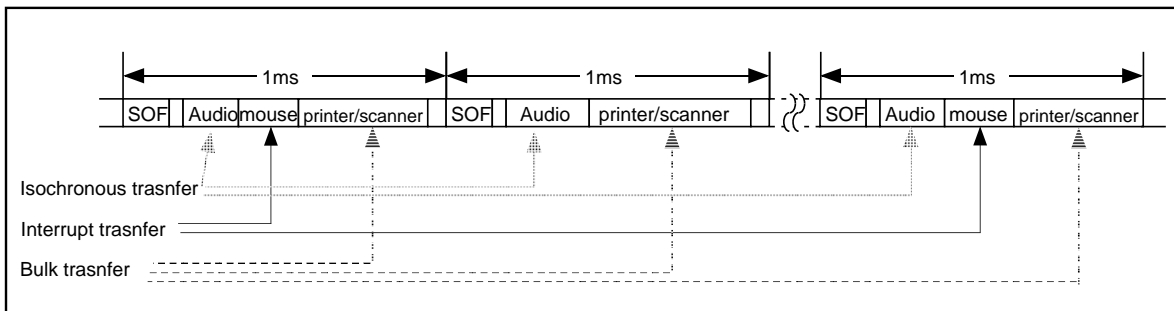


Figure 2.8.1. Frame image

### ● Packet

A packet is the unit in which the host CPU or the device secures a bus. In the USB, data are transmitted/received in the packet unit.

A packet is a group of bit data strings (fields), which starts with the SOP (Start-of-Packet) as a part of the SYNC (synchronous data) field. This is followed by the PID field which identifies a packet type and, then, each data field of a frame number/ address/ data field, etc., finally ending with EOP (End-of-Packet) which indicates the end of a packet. The packet types and formats are shown below:

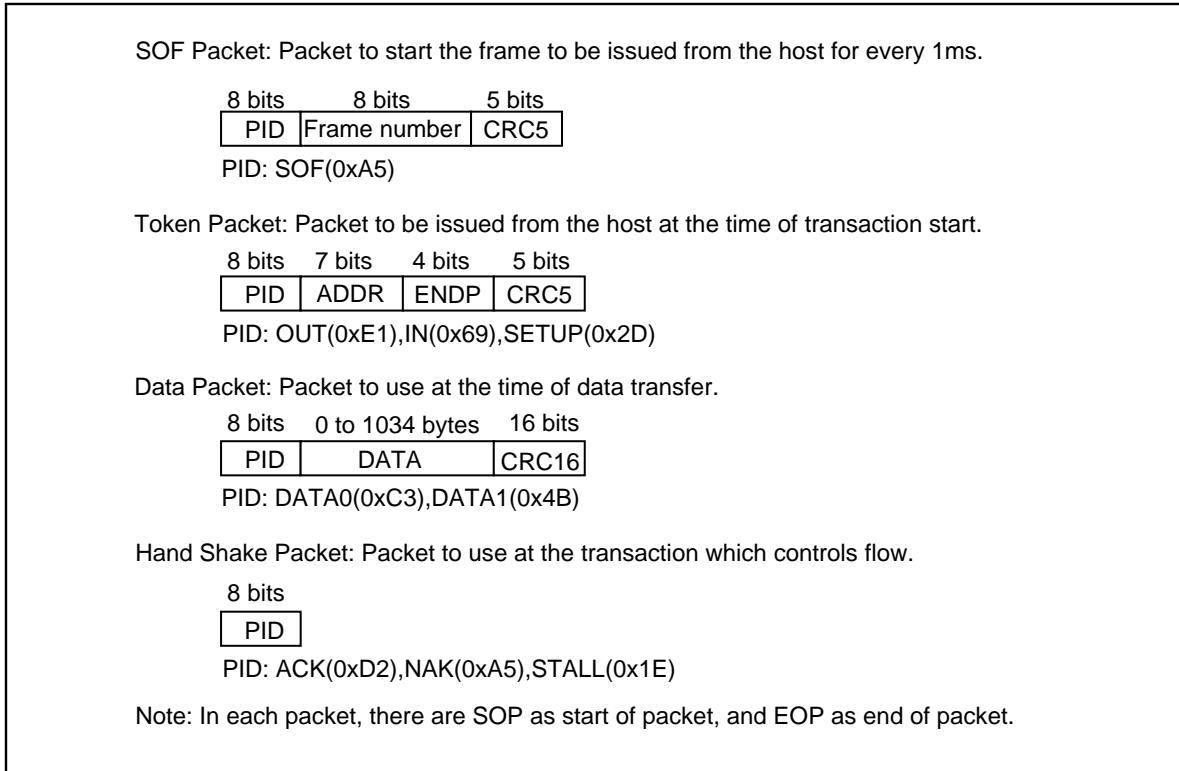


Figure 2.8.2. Kinds of packet

Table 2.8.1. List of USB packet recognitions

PID type	PID name	Process overview
Token	SETUP	Reports the operation request to device by the host CPU
	IN	Requests the data transmit to device by the host CPU
	OUT	Requests the data receive to device by the host CPU
	SOF	Indicates the start of frame to device by the host CPU
DATA	DATA0	Indicates that the sequence bit of transmit/receive data is even number
	DATA1	Indicates that the sequence bit of transmit/receive data is odd number
Handshake	ACK	Reports that the transmit data was correctly completed
	NAK	Reports that the device is in the communication wait state
	STALL	Reports that the communication was incorrectly completed

● **Transaction**

A transaction is the unit in which the host CPU schedules one frame. Each transaction is configured with packet, and the transaction types are determined according to the configuration pattern. The transaction types and formats are shown below:

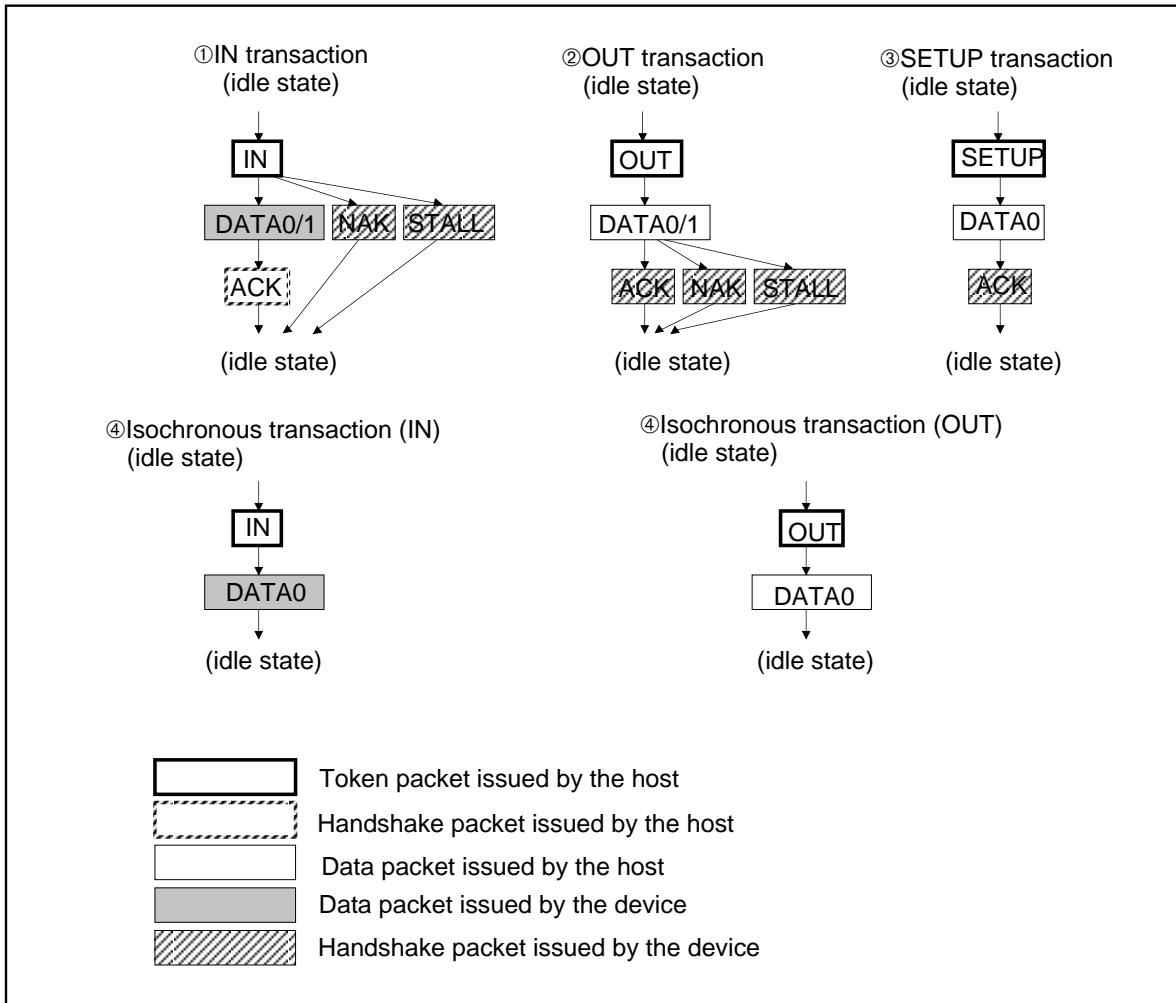


Figure 2.8.3. Formats of transactions

### ● Communication Sequence

The control transfer is used common to all devices at the time of setup, which consists of three kinds of stages being combined for one processing. The control transfer starts with setup stage. According to the content, data stage (control read transfer or control write transfer) is executed, followed by status stage being executed to finally complete one processing. In control transfer, use of endpoint 0 has been specified.

The communication sequence of control transfer is shown in Figure 2.8.4.

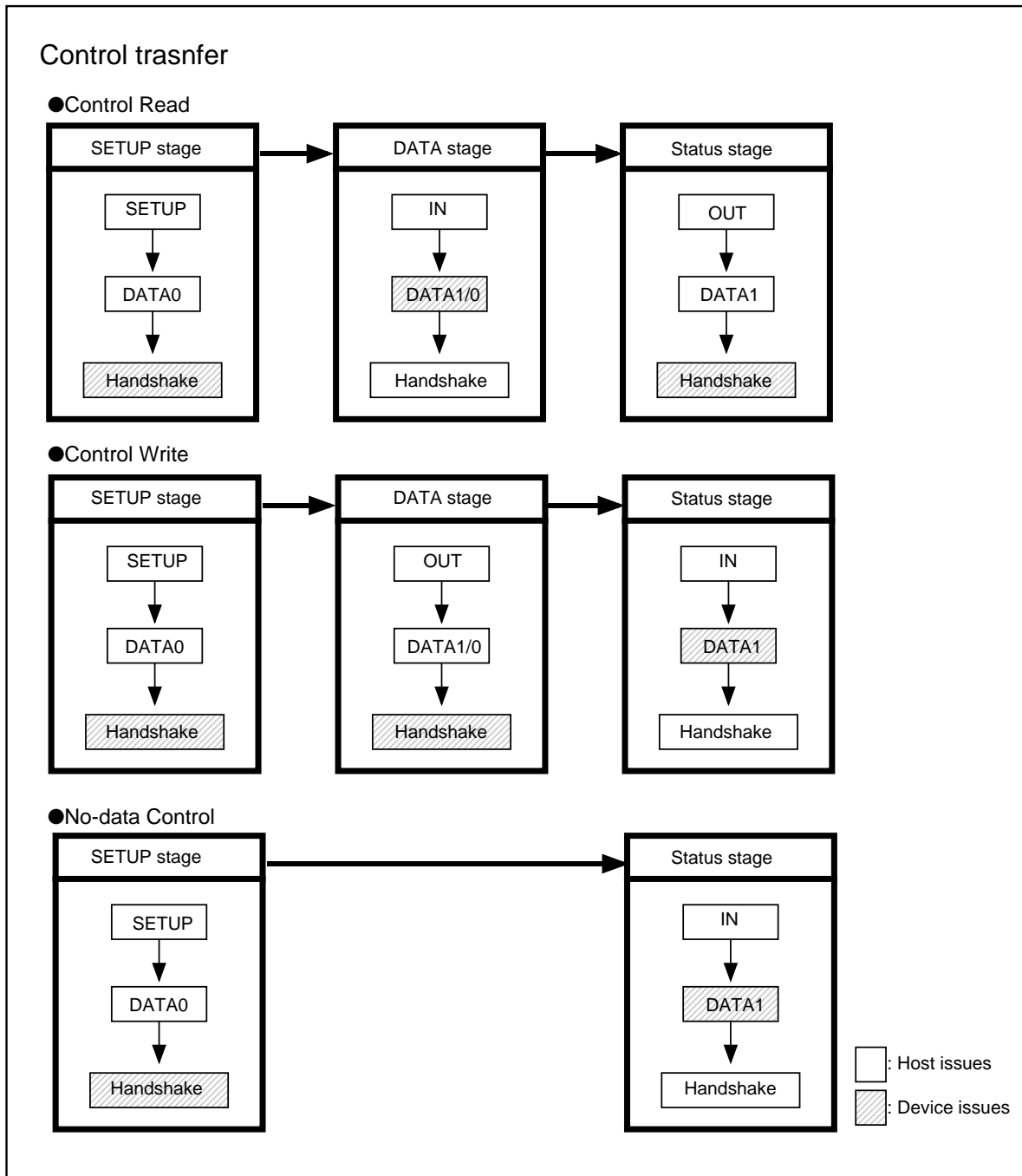


Figure 2.8.4. Control transfer communication sequence

**● Control Read Transfer**

In setup stage, host notifies the device that it is control read transfer. Then, in data stage, data are transmitted from the device to host through repetition of IN transaction. Finally, in status stage, OUT transaction is executed (that the host transmits an empty packet of data length (0) to the device) to complete the control read transfer.

**● Control Write Transfer**

In setup stage, host notifies the device that it is control write transfer. Then, in data stage, data are transmitted from the host to device through repetition of OUT transaction. Finally, in status stage, IN transaction is executed (that the device transmits an empty packet of data length (0) to the host) to complete the control write transfer.

**● No Control Data Transfer**

In setup stage, host notifies the device that it is no control data transfer. Then, in status stage, IN transaction is executed (that the device transmits an empty packet of data length (0) to the host) to complete the no control data transfer.

The execution result of setup stage and data stage are notified to host CPU in status stage. For details of response format of control transfers, refer to USB2.0 specification.

**● Device Request**

Concerning setup transaction in the setup stage of control transfer, format of its data phase has been defined, which is called device request.

For standard (0) type, it is called standard device request, which is the basic device request to be supported by all the USB devices.

For class (1) type, it is called class request. The USB implementers forum (USB IF) defines a device class, and determines the configuration required in the class and the class request.

For each data format of the device request, refer to USB2.0 specification or the specification for each class.

### (3) Bulk Transfer

#### ● Bulk IN Transfer

In bulk IN transfer which data are transmitted from the device to the host CPU, IN transactions are repeated. When transmit data are available in IN FIFO, the M30245 group issues a data packet to the IN token. When, during the handshake phase of each transaction, the M30245 group has normally received ACK packet issued by the host PC, it toggles DATA0 and DATA1 of data packet on next data phase. This serves to ensure handshake. The M30245 group executes the following responses when the data are not transmitted normally:

- When the received IN token is destroyed, response is not executed.
- When ACK handshake was not included in the transmit data, it is retransmitted on next IN token.
- When the M30245 group was stalling, STALL handshake is returned.
- When the transmit data are not available in IN FIFO, NAK handshake is returned.

#### ● Bulk OUT Transfer

In bulk OUT transfer which data are transmitted from the host CPU to the device, OUT transactions are repeated.

The M30245 group has normally received a data packet, and then returns ACK handshake. Normal receiving is the status which is free of any bit stuffing error or CRC error and which data PID have been correctly received. When, during the handshake phase of each transaction, the host PC has normally received ACK packet issued by the M30245 group, it toggles DATA0 and DATA1 of data packet on next data phase. This serves to ensure handshake. The M30245 group executes the following responses when the data are not received normally:

- When the received OUT token is destroyed, response is not executed.
- When the M30245 group was stalling, STALL handshake is returned. Also, when the packet, which is exceeding receivable data size, is transmitted, STALL handshake is returned.
- When inconsistency of the sequence bits is detected in the received data, ACK handshake is returned.
- When OUT FIFO of the M30245 group could not receive full data, NAK handshake is returned.

For details, refer to USB2.0 specification.

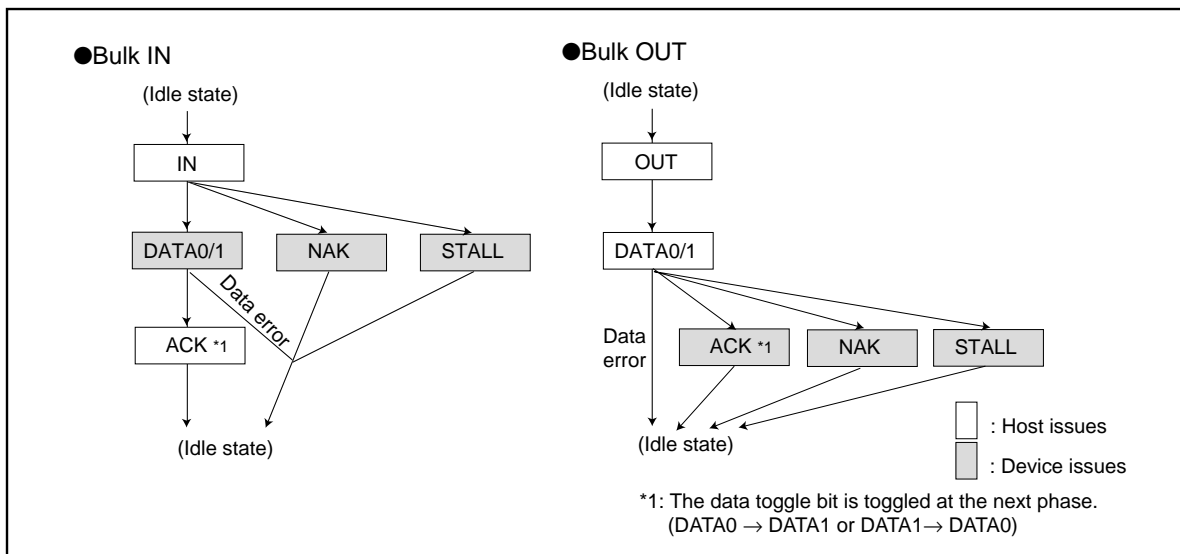


Figure 2.8.5. Bulk transfer

#### (4) Isochronous Transfer

##### ● Isochronous IN Transfer

In isochronous IN transfer which data are transferred from the device to the host CPU, isochronous (IN) transactions are repeated. Isochronous transaction does not have the handshake phase. The data packet consists only of DATA0. Toggling with DATA1 is not performed. When transmit data is available in IN FIFO, the M30245 group issues a data packet to the IN token. The M30245 group executes the following responses when the data are not transmitted normally:

- When the received IN token is destroyed, the data are not issued.
- When the transmit data are not available in IN FIFO, an empty packet of data length (0) is issued.

##### ● Isochronous OUT Transfer

In isochronous OUT transfer which data are transferred from the host CPU to the device, isochronous (OUT) transactions are repeated. Isochronous transaction does not have the handshake phase. The data packet consists only of DATA0. Toggling with DATA1 is not performed.

The M30245 group has received a data packet, indicates whether or not the data content is normal by using a status flag. The M30245 group executes the following responses when the data are not received normally:

- When the received OUT token is destroyed, the data are not received.
- When the received data are destroyed (bit stuffing error or CRC error occur), the data are received.
- When the packet, which is exceeding receivable data size, is transmitted, the data are not received.
- When OUT FIFO of the M30245 group could not receive the full data, the data are not received.

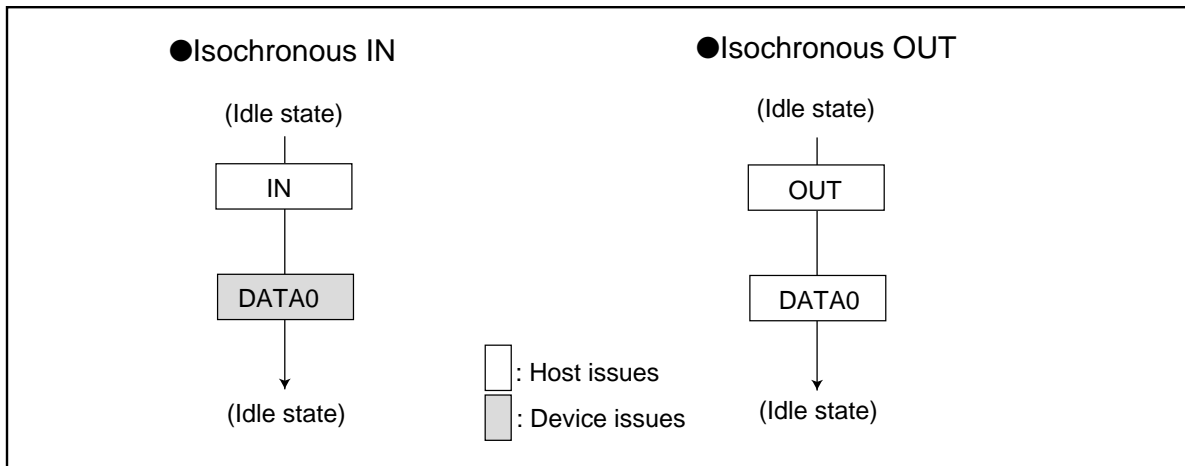


Figure 2.8.6. Isochronous transfer

#### (5) Interrupt Transfer

This transfer form is same as the bulk transfer. Refer to “(3) Bulk Transfer” of this manual.

## (6) Device State

The device has states and, transits between the states. The M30245 group does not execute state transition on the hardware. Control it by the software based on requests of the related USB interrupt request. The device state transition is shown in Figure 2.8.7.

A series of processes from bus connection to configuration is called “emulation”. Each state is explained as follows:

### ① Connection State:

This is the state which the device has been connected to the bus.

### ② Powered State:

This is the state where the hub has been completed the configuration and has been supplied power to the bus.

### ③ Default State:

This is the state where the reset signal has been received from the host CPU. It is responded as default address (0). This is the unconfigured state (configuration 0).

### ④ Address State:

This is the state which the SET\_ADDRESS standard device request has been received and a device address other than “0” has been assigned. This is the unconfigured state (Configuration 0).

### ⑤ Configured State

This is the state which endpoint 0 has been received the SET\_CONFIGURATION standard device request and the device has been configured.

### ⑥ Suspend State

This is the state which inactive state followed 3ms or more.

If a bus active has been detected, the state shifts to the former one. In the M30245 group, when a bus reset is detected in suspend state, detects bus active and shifts to the former state, then, detects the bus reset.

## (7) USB Related Registers Memory Mapping

The USB related registers memory mapping is shown in Figure 2.8.8. The list of USB related registers items is shown in Table 2.8.2.



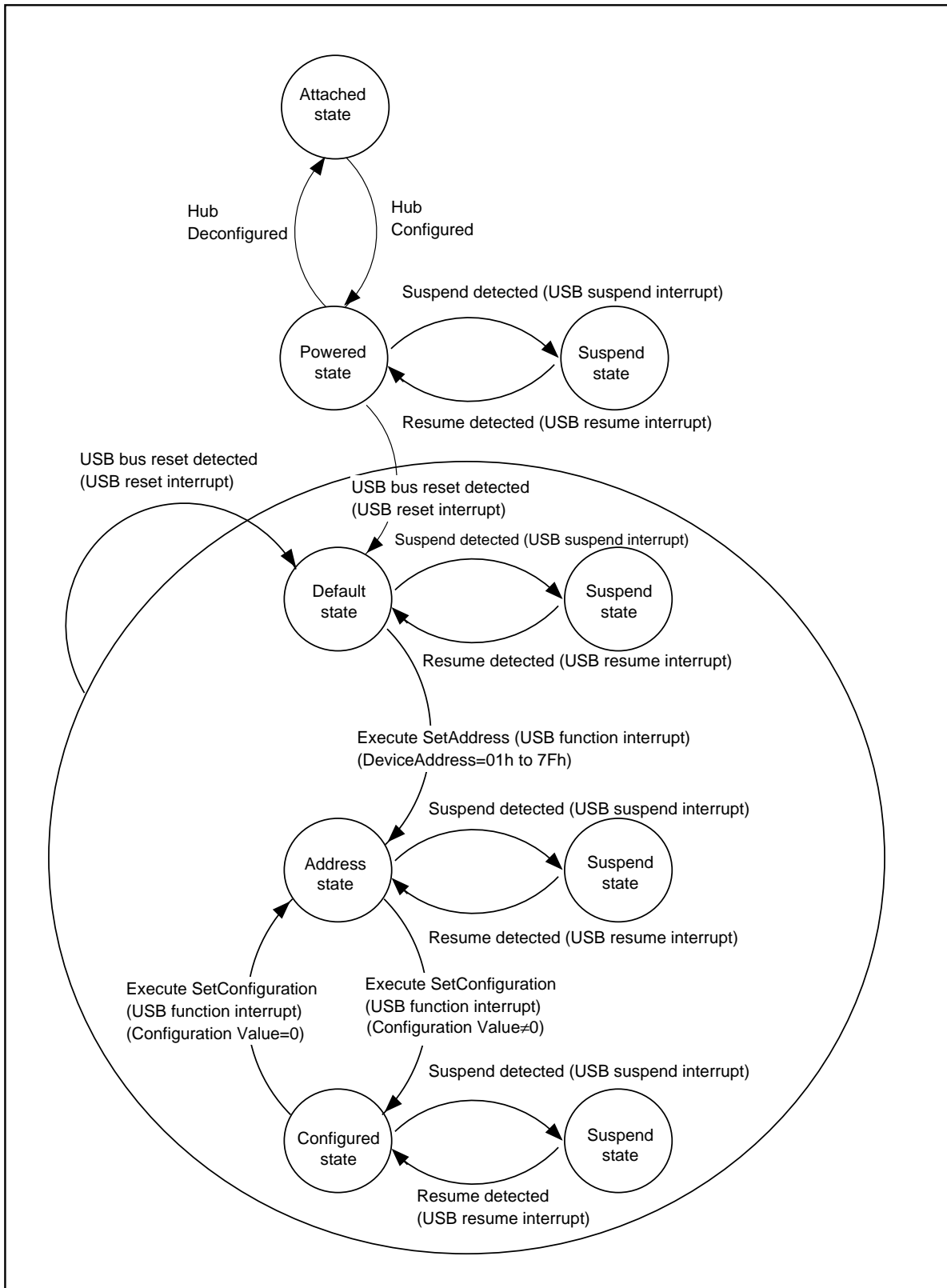


Figure 2.8.7. Device state transition

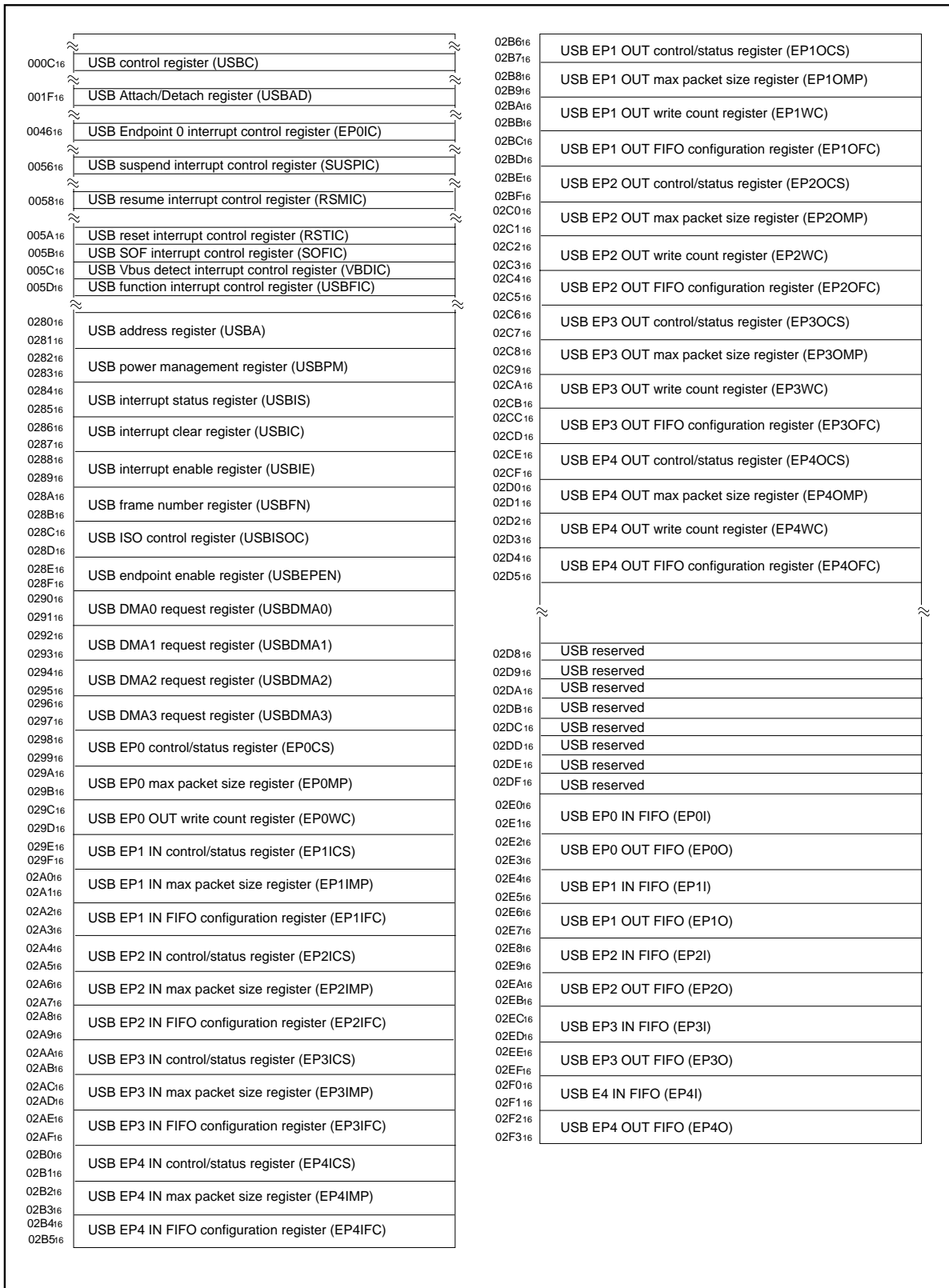


Figure 2.8.8. USB registers memory mapping

**Table 2.8.2. List of USB Related Registers Items**

Section name	Register name
2.8.2 USB function control	USB control register, USB Attach/Detach register, USB endpoint enable register, USB endpoint x(x=0-4) IN FIFO data register, USB endpoint x(x=0-4) OUT FIFO data register
2.8.3 USB Interrupt	USB function interrupt status register, USB function interrupt clear register, USB function interrupt enable register, USB frame number register
2.8.4 USB Operation (Suspend/Resume Function)	USB power management register
2.8.5 USB Operation (Endpoint 0)	USB address register, USB endpoint 0 control and status register, USB endpoint 0 MAXP register, USB endpoint 0 OUT write count register
2.8.6 USB Operation (Endpoint 1-4 reception)	USB endpoint x(x=1-4) OUT control and status register, USB endpoint x(x=1-4) OUT MAXP register, USB endpoint x(x=1-4) OUT write count register, USB endpoint x(x=1-4) OUT FIFO configuration register
2.8.7 USB Operation (Endpoint 1-4 transmission)	USB ISO control register, USB endpoint x(x=1-4) IN control and status register, USB endpoint x(x=1-4) IN MAXP register, USB endpoint x(x=1-4) IN FIFO configuration register
2.8.8 USB Operation (Interface of USB and DMAC transfer)	USB DMAx(x=0-3) request register

## 2.8.2 USB function control

The USB function control unit needs to be enabled for using the USB function. The initialization procedure of the USB function control unit is explained below:

### (1) Related Registers

#### ● USB control register

This register is used to control each operation of the USB function control unit. When using the USB function, be sure to set USB clock enable bit to “1” before USB enable bit is set to “1”. This register is not affected by the USB reset signal. After the USB is enabled (USBC7 =“1”), a minimum 187.5 ns of delay (three cycles of BCLK) is required before performing any other USB register read/write operations.

#### ●USB clock enable bit

This bit is used to enable/disable the USB clock (fusb). This clock is supplied from frequency synthesizer and is required for the USB operation. Set this bit to “1” when enabling the USB clock.

#### ●USB SOF port select bit

This bit is used to enable/disable a SOF signal output on the P92 pin. Set this bit to “1” when using the USB SOF signal. In this case, set the port P92 to output mode. When this bit is set to “1”, low pulse is always output for about 166ns (2 cycles of the 12MHz USB clock) at start of the frame packet.

#### ●USB enable bit

This bit is used to enable/disable the USB block. When this bit is set to “1”, USB function is enabled. After setting “1” to this bit, wait for at least 250ns and, then, read or write the other USB related register.

The configuration of USB control register is shown in Figure 2.8.9.

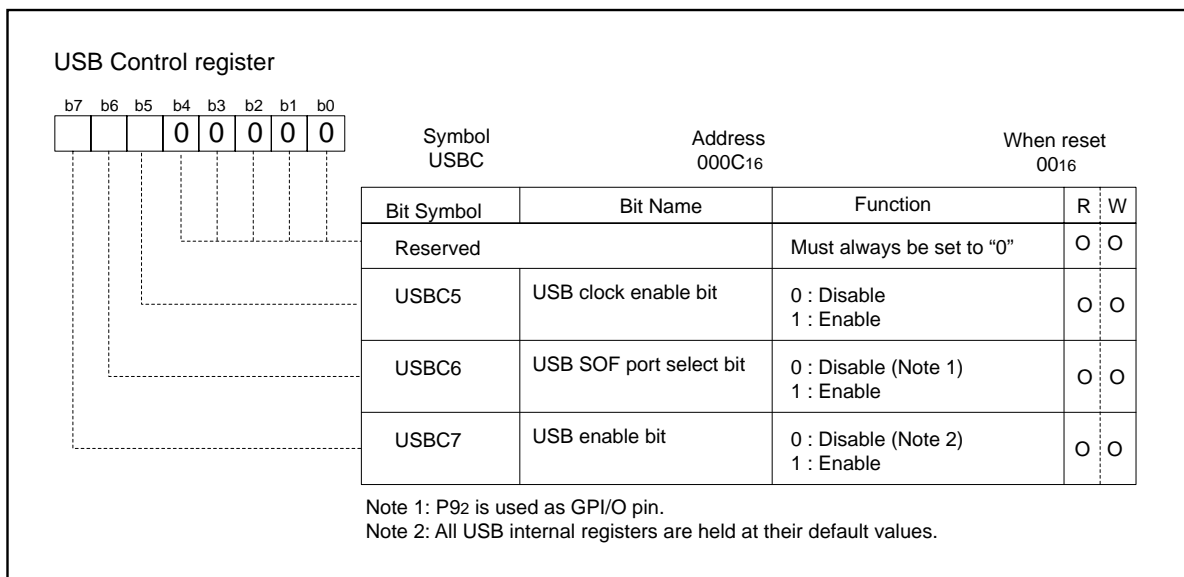


Figure 2.8.9. USB control register

### ● USB attach/detach register

This register is used to control attach/detach from the USB host without physically attaching/detaching the USB cable.

#### •Port 9<sub>0</sub>-Second bit

The port P9<sub>0</sub> operates as standard port when this bit is set to “0”. Connect a 1.5kΩ resistance between the USB D+ pin and the Uvcc pin. (The timing of D+ line pull-up is depending on the Uvcc pin.)

When this bit is set to “1”, the P9<sub>0</sub> has the USB attach/detach function, serving as the power supply pin for pull-up to the D+ line. When using the USB attach/detach function, be sure to connect between the USB D+ pin and the P9<sub>0</sub>/ATTACH pin via a 1.5kΩ pull-up resistance. Also, in either case, be sure to connect the UVcc pin to the power source.

#### •Attach/detach bit

This bit is valid when port 9<sub>0</sub>-Second bit is set to “1”. When this bit is set to “0”, supply of UVcc pin voltage to P9<sub>0</sub> is stopped and the USB cable becomes a detach state artificially. When this bit is set to “1”, the USB cable becomes a attach state artificially since the Uvcc pin voltage is supplied to the P9<sub>0</sub> and D+ line is pulled up. After frequency synthesizer is stabilized, set “1” (attach state) to this bit.

#### •Vbus detect enable bit

This bit is used to enable Vbus detection by setting to “1”. When enabling the Vbus detection, connect the Vbus pin of USB connector to a VbusDTCT pin.

When attach/detach bit is changed from “0” to “1” or from “1” to “0”, time required till the host recognizes attach/detach varies according to board resistance factor/ capacity factor/ USB cable capacity of the device, host's board characteristics, and processing speed. Perform sufficient evaluation through controlling attach/detach bit in accordance with the actual user's system.

The configuration of USB attach/detach register is shown in Figure 2.8.10.

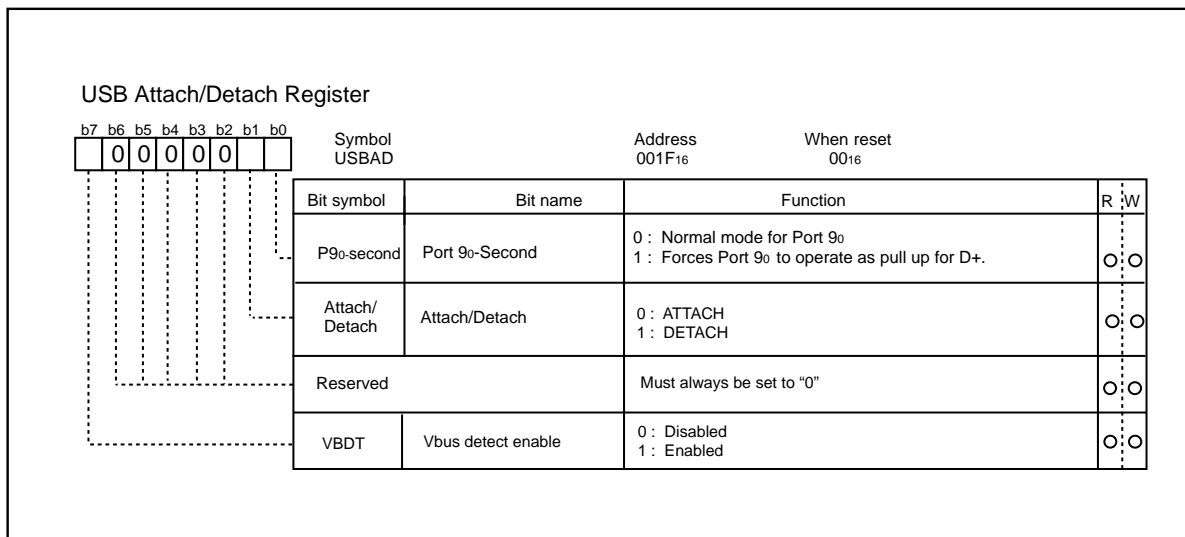


Figure 2.8.10. USB Attach/Detach register

● **USB endpoint enable register**

Endpoints 1 to 4 are used to enable endpoint IN/OUT FIFOs for use. The endpoint 0 is always enabled and cannot be disabled by software. All Endpoints 1 to 4 are disabled after reset.

The configuration of USB endpoint enable register is shown in Figure 2.8.11.

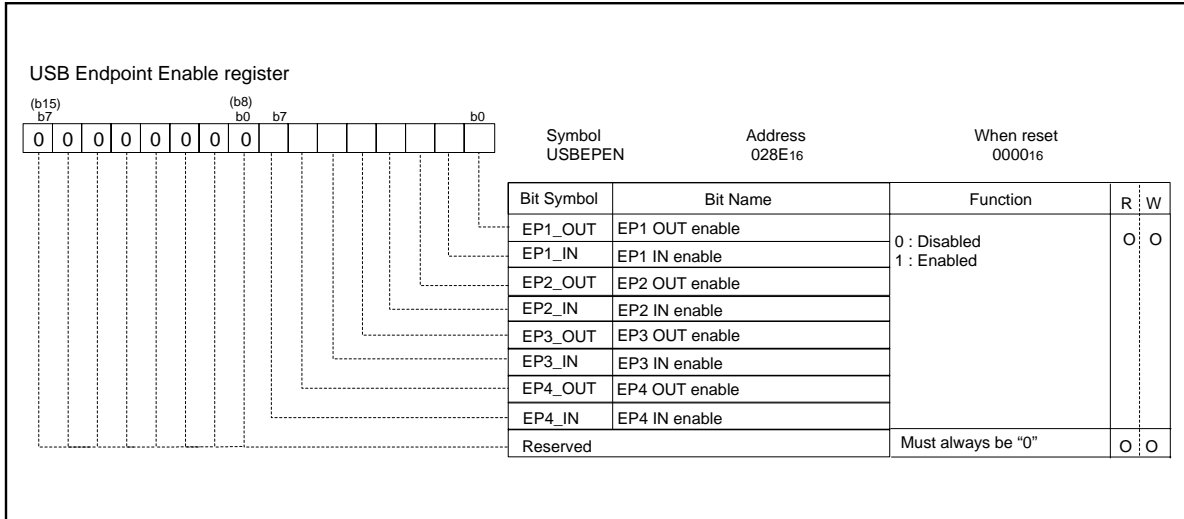


Figure 2.8.11. USB endpoint enable register

● **USB endpoint x(x=0 to 4) IN FIFO data register**

Endpoints 0 to 4 respectively have their IN FIFOs. At the time of transmission to the host PC, write the transmit data in these registers. Access these registers in word cycle or byte cycle to the lower byte.

The configuration of USB x(x=0~4) IN FIFO data register is shown in Figure 2.8.12.

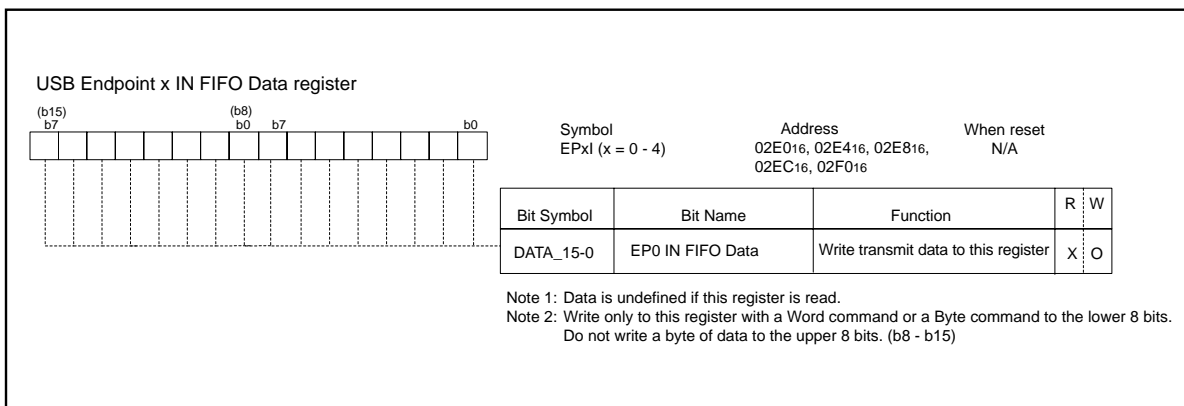


Figure 2.8.12. USB x(x=0~4) IN FIFO data register

● USB endpoint x(x=0 to 4) OUT FIFO data register

Endpoints 0 to 4 respectively have their OUT FIFOs. When data are received from the host PC, read the receive data from these registers. Access these registers in word cycle or byte cycle to the lower byte.

The configuration of USB x(x=0~4) OUT FIFO data register is shown in Figure 2.8.13.

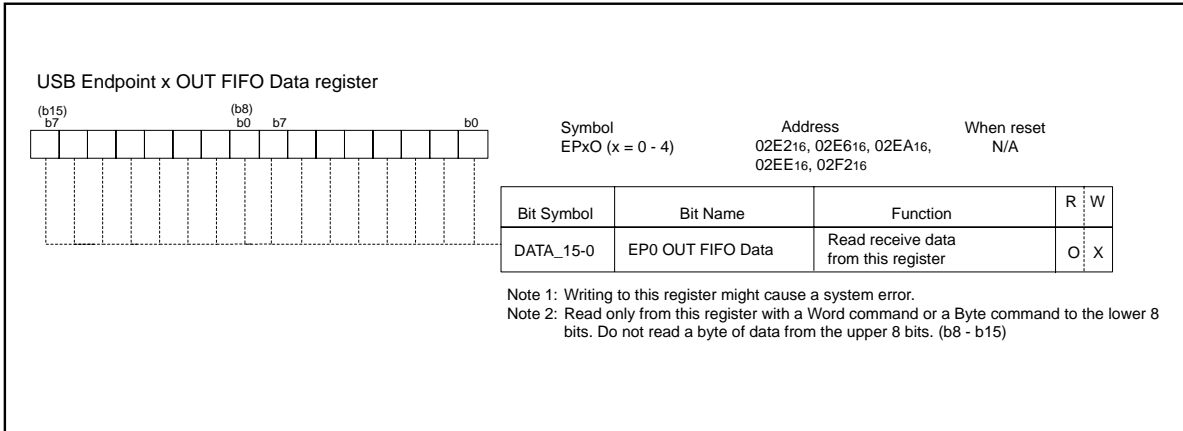


Figure 2.8.13. USB x(x=0~4) OUT FIFO data register

The endpoint x IN/OUT FIFO mapping is shown in Figure 2.8.14.

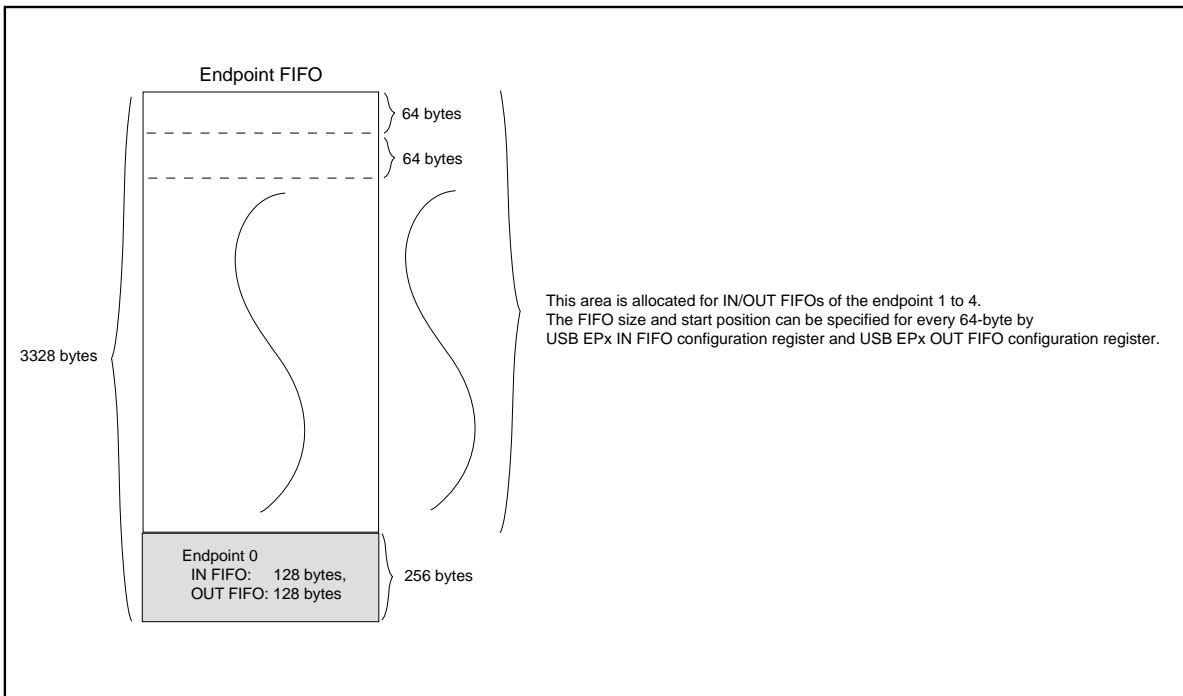


Figure 2.8.14. Endpoint x IN/OUT FIFO mapping

**(2) Enable of USB Function Control Unit**

The initialization procedure of the USB function control unit of the M30245 group after hardware reset is explained below. Further, for power supply being supplied from the USB, the total driving current has to be controlled to keep equal to or below 100mA.

**● Setting of the frequency synthesizer**

- 1: Clear protect.
- 2: Set the frequency synthesizer related registers to generate 48MHz required for the fUSB.
- 3: Enable the frequency synthesizer by setting bit 0 in the frequency synthesizer control register to "1".
- 4: Disable protect.
- 5: Wait for 3ms to stabilize the frequency synthesizer.
- 6: Check frequency synthesizer lock status bit. If the frequency synthesizer is in unlock state, waiting for 0.1ms and rechecking are repeated.

**● Setting of the USB function control unit**

- 7: Set the USBC5 to "1" to enable the USB clock.
- 8: Set the USB attach/detach register to "0316" (Port P90 is set as the power supply pin for pull-up to the D+ line), and set the USBC7 to "1" to enable the USB block. For operation of the USB related registers, a minimum 187.5ns of wait (three cycles of BCLK) is required.

Initialize the endpoint to be used after the USB function control unit being enabled.

The setting example of frequency synthesizer division is shown in Figure 2.8.15. The setup timing of frequency synthesizer after hardware reset is shown in Figure 2.8.16. The initialization procedure of frequency synthesizer and USB function control unit are shown in Figure 2.8.17 and Figure 2.8.18. The initialization procedure of endpoint is shown in Figure 2.8.19 and Figure 2.8.20.



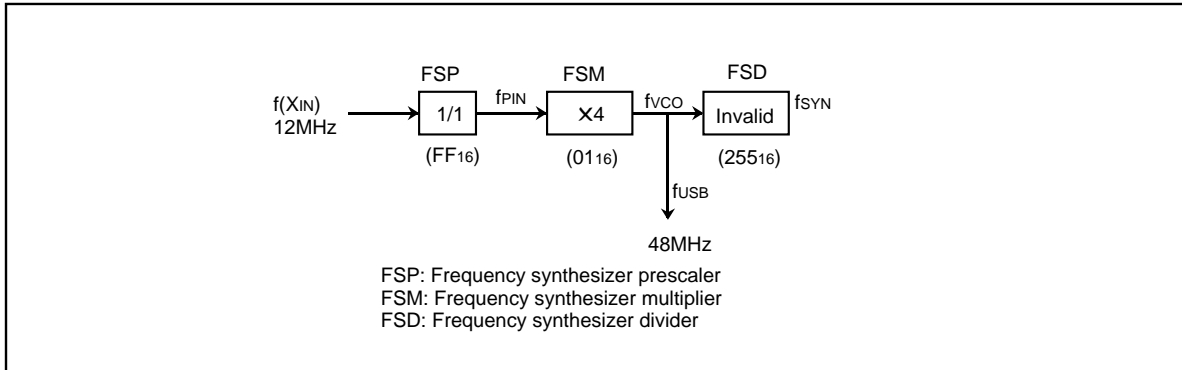


Figure 2.8.15. Setting example of frequency synthesizer division

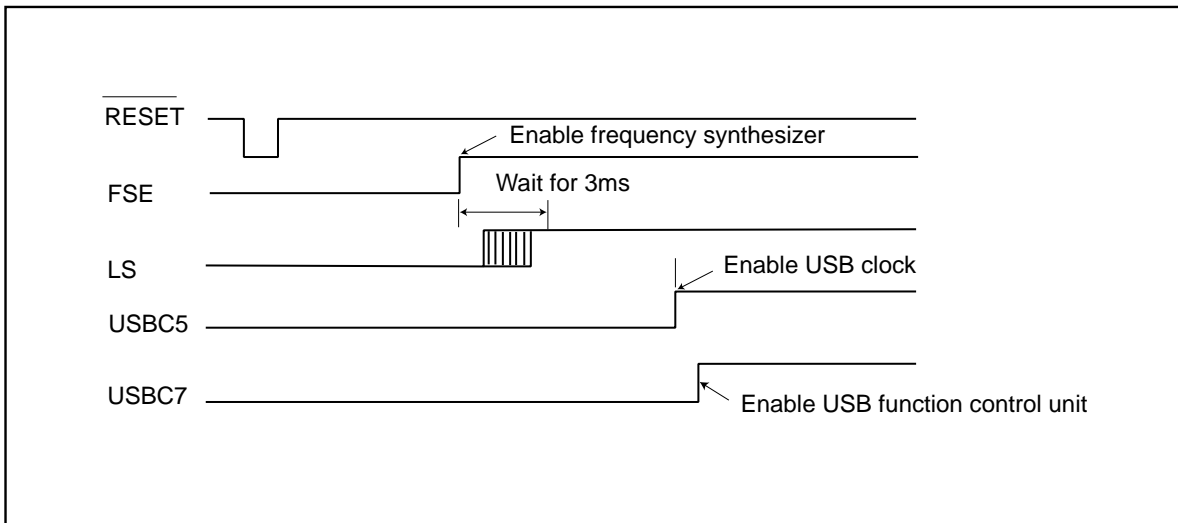


Figure 2.8.16. Setup timing of frequency synthesizer after hardware reset

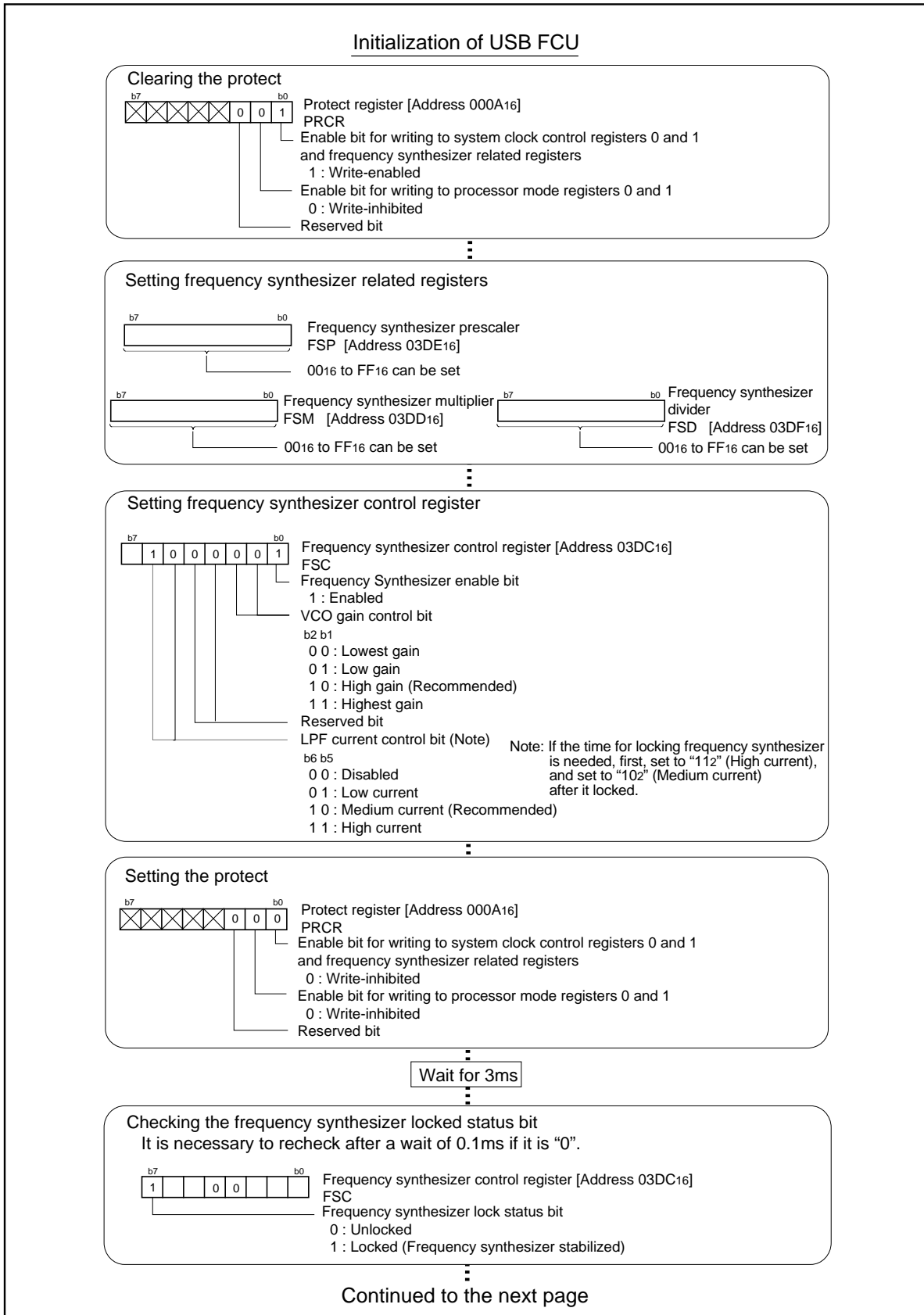
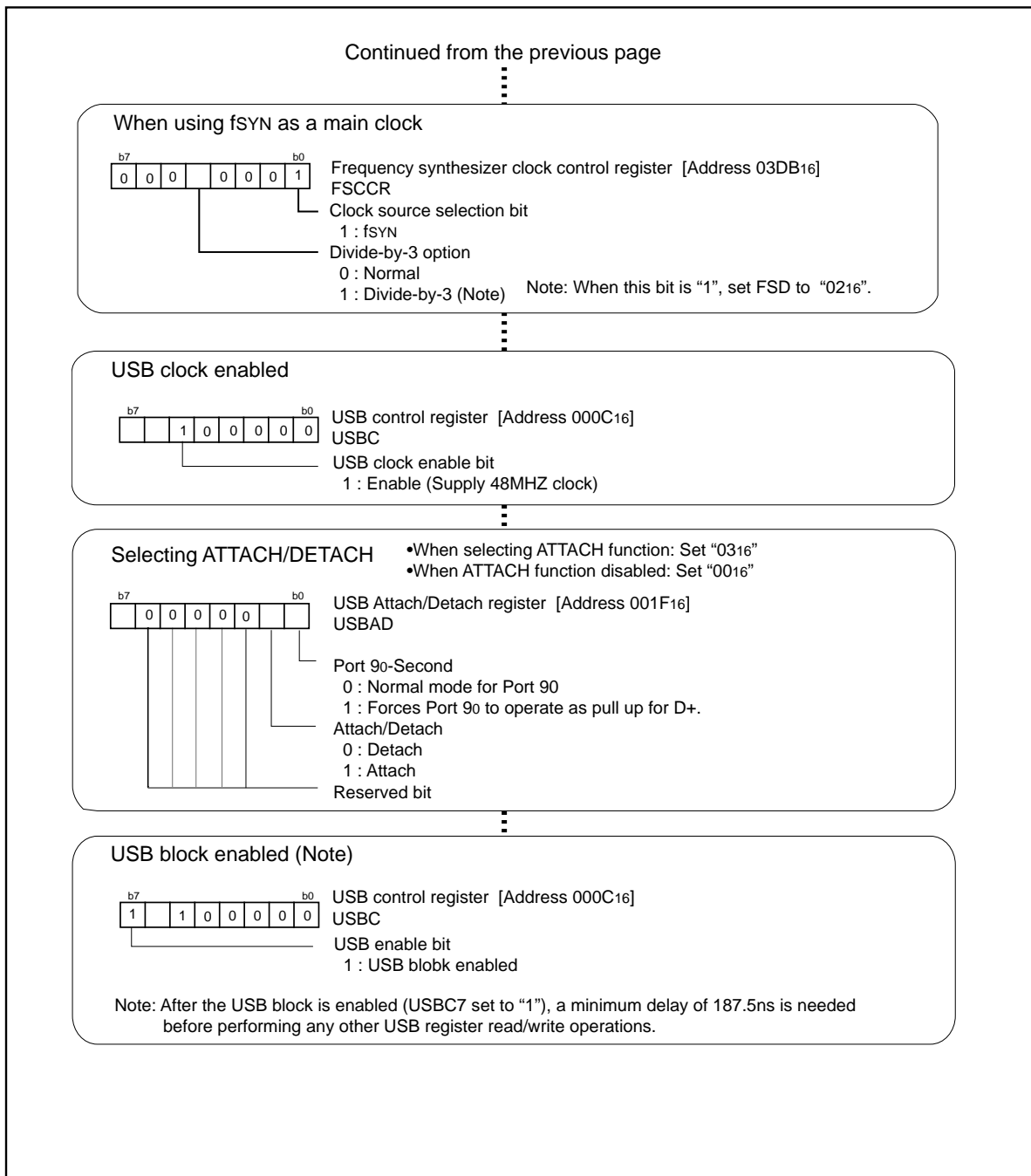
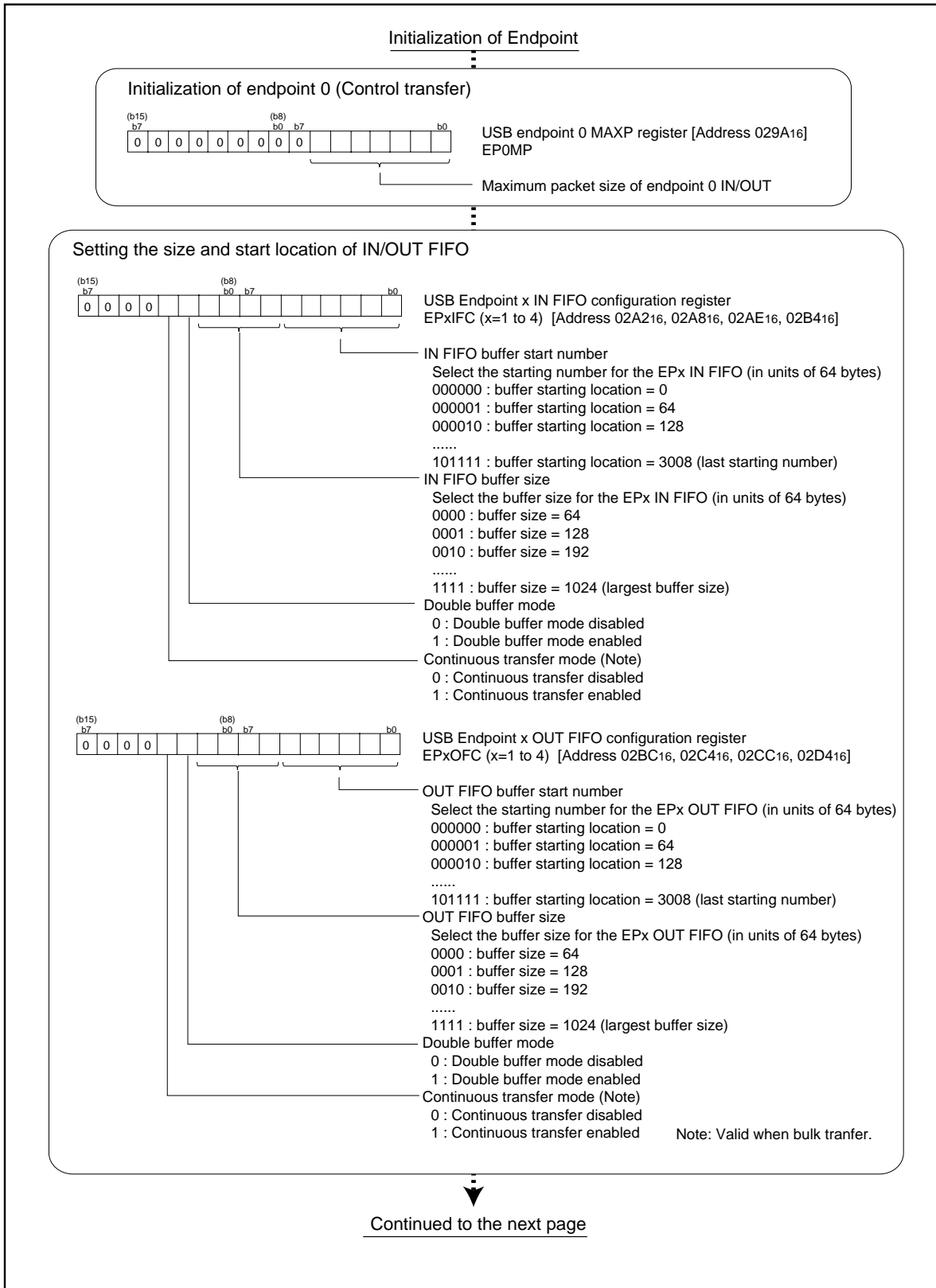


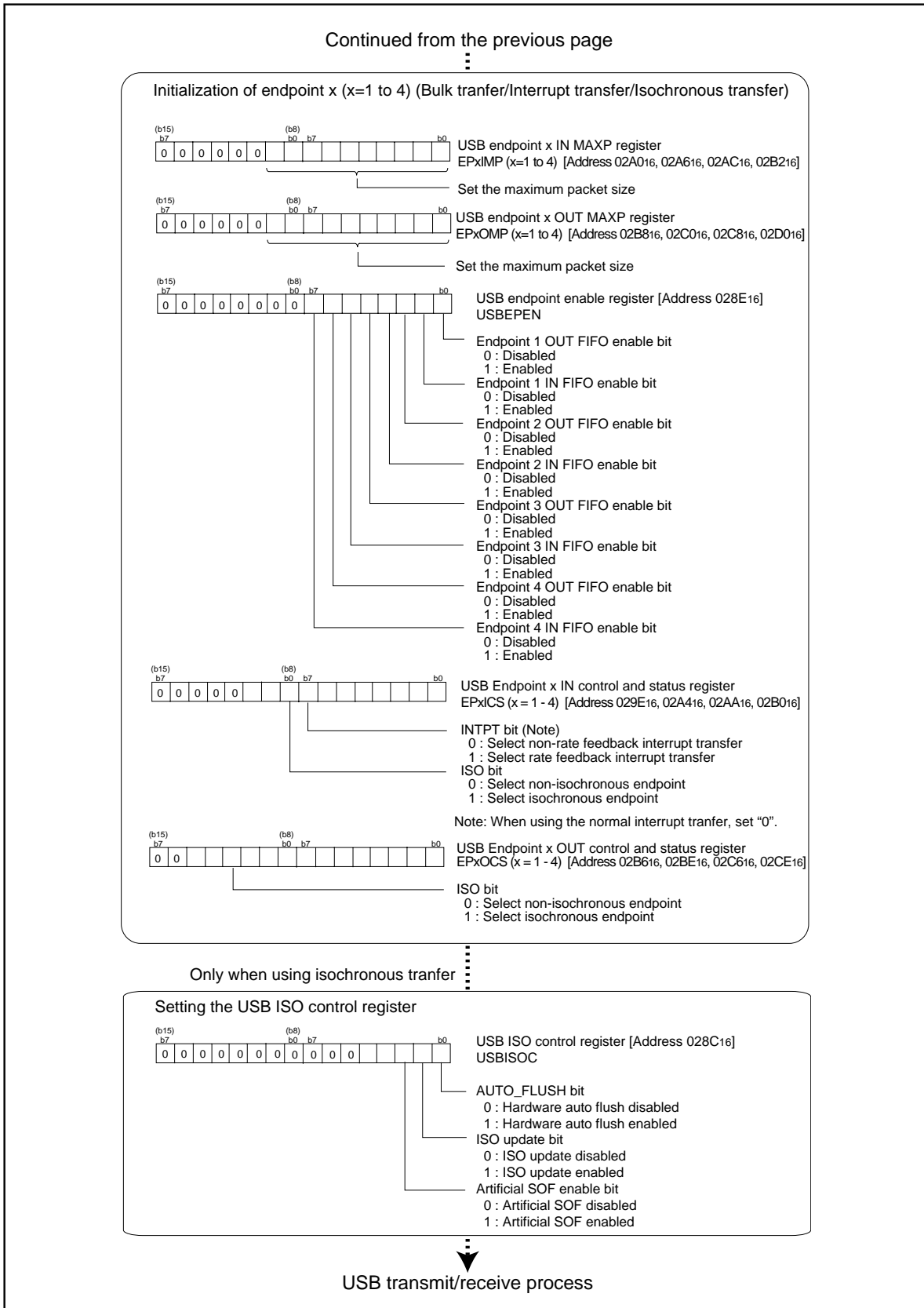
Figure 2.8.17. Initialization procedure of frequency synthesizer and USB function control unit (1)



**Figure 2.8.18. Initialization procedure of frequency synthesizer and USB function control unit (2)**



**Figure 2.8.19. Initialization procedure of endpoint (1)**



**Figure 2.8.20. Initialization procedure of endpoint (1)**

### (3) Disable of USB Function Control Unit

After the USB function control unit being enabled, if the system design requires to disable the USB function, follow the procedure below:

- 1: Disable the USB clock by clearing USB enable bit (USBC7) to "0".
- 2: Disable the USB clock by clearing USB clock enable bit (USBC5) to "0".
- 3: Disable the frequency synthesizer by clearing frequency synthesizer enable bit (FSE) to "0".

Normally, for system design which continues enabling the USB function, disabling the USB function control unit is not required.

### (4) Vbus Detection

During USB self-powered operation, the Vbus detect function is used to switch into bus power only when the device is connected to the host PC and power supply is available from the Vbus, for minimizing battery consumption. To use the Vbus detect function, it is necessary that the VbusDTCT pin is processed by hardware and the Vbus detect interrupt is set by software. The VbusDTCT pin is used for the Vbus detect function. When operating the USB in self-powered mode, connect the Vbus line from the USB connector to the VbusDTCT pin. For enable/disable of the Vbus detect function, set Vbus detect enable bit (bit 7 at address 001F16) of USB attach/detach register to "1". Set the interrupt priority level by using USB Vbus detect interrupt control register (VBDIC: address 005C16). Each time the USB host powers ON/OFF, a Vbus detect interrupt will be occurred. When a Vbus detect interrupt is occurred, the Vbus detect state bit located in the port 9 data register (bit 1 at address 03F116) should be read to determine if the Vbus is powered ON/OFF.

To avoid receiving a false Vbus detect interrupt at start-up, the Vbus detect should be enabled before enabling the Vbus detect interrupt. Use the following procedure when enabling the Vbus detect function:

- 1: Enable a Vbus detect by setting "1" to Vbus detect enable bit (bit 7 at address 001F16).
- 2: Clear the Vbus detect interrupt request by setting "0" to Vbus detect interrupt request bit (bit 3 at address 005C16).
- 3: Enable the Vbus detect interrupt by setting the Vbus detect interrupt priority level greater than "0002" (bit 0 to 2 at address 005C16).

### 2.8.3 USB Interrupt

The USB related interrupts include USB suspend interrupt, USB resume interrupt, USB reset interrupt, USB endpoint 0 interrupt, USB function interrupt, and USB SOF interrupt.

#### (1) Related Registers

##### ● USB function interrupt status register

This register is used to judge the USB function interrupt factor. This is the read-only register which indicates each interrupt request state of endpoint  $x(x=1\sim4)$  IN interrupt, endpoint  $x(x=1\sim4)$  OUT interrupt, and error interrupt. On occurrence of an interrupt request, this is set to "1". Each interrupt status flag can be cleared to "0" by setting "1" to the corresponding bit of USB function interrupt clear register.

- **Endpoint 1 IN Interrupt Status Flag**
- **Endpoint 2 IN Interrupt Status Flag**
- **Endpoint 3 IN Interrupt Status Flag**
- **Endpoint 4 IN Interrupt Status Flag**

These flags indicate endpoint  $x(x=1\sim4)$  IN interrupt request state, respectively. These flags are set to "1" at the corresponding endpoint that has been enabled by USB function interrupt enable register in the following cases:

- The endpoint is enabled from a disabled state
- One buffer data is successfully transmitted
- Buffer flush has been executed by either AUTO\_FLUSH or FLUSH bit (bit 6 at address EPxICS) being set to "1" while buffer data exists in the IN FIFO.

- **Endpoint 1 OUT Interrupt Status Flag**
- **Endpoint 2 OUT Interrupt Status Flag**
- **Endpoint 3 OUT Interrupt Status Flag**
- **Endpoint 4 OUT Interrupt Status Flag**

These each indicate endpoint  $x(x=1\sim4)$  OUT interrupt request state. When a data is successfully received, these flags are set to "1" at the corresponding endpoint.

##### • **Error Interrupt Status Flag**

This flag indicates that an error has been occurred at the endpoint  $x(x=0\sim4)$ . This flag is set to "1" in the following cases:

- The EP0CSR4 (FORCE\_STALL) flag of endpoint 0 is set to "1"
- The EP0CSR5 (SETUP\_END) flag of endpoint 0 is set to "1"
- The INxCSR2 (UNDER\_RUN) flag of endpoint 1 to 4 IN is set to "1"
- The OUTxCSR2 (OVER\_RUN) flag of endpoint 1 to 4 OUT is set to "1"
- The OUTxCSR3 (FORCE\_STALL) flag of endpoint 1 to 4 OUT is set to "1"
- The OUTxCSR4 (DATA\_ERR) flag of endpoint 1 to 4 OUT is set to "1"

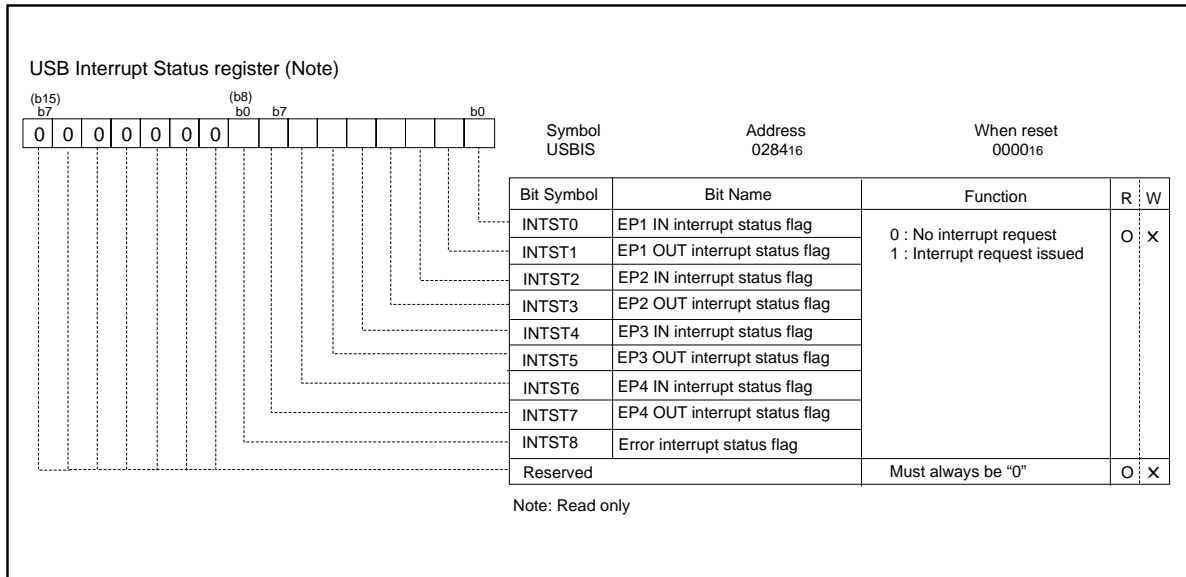


Figure 2.8.21. USB function interrupt status register

#### ● USB function interrupt clear register

This register is used to clear the USB function interrupt request factor.

The interrupt status flag corresponding to USB function interrupt status register is cleared to "0" by setting "1" to the interrupt status clear flag.

The configuration of USB function interrupt status register is shown in Figure 2.8.22.

#### ● USB function interrupt enable register

This register is used to set the USB function interrupt request factor.

The USB function interrupt request occurs when the request of interrupt which set the enable bit to "1" occurs.

The configuration of USB function interrupt enable register is shown in Figure 2.8.23.



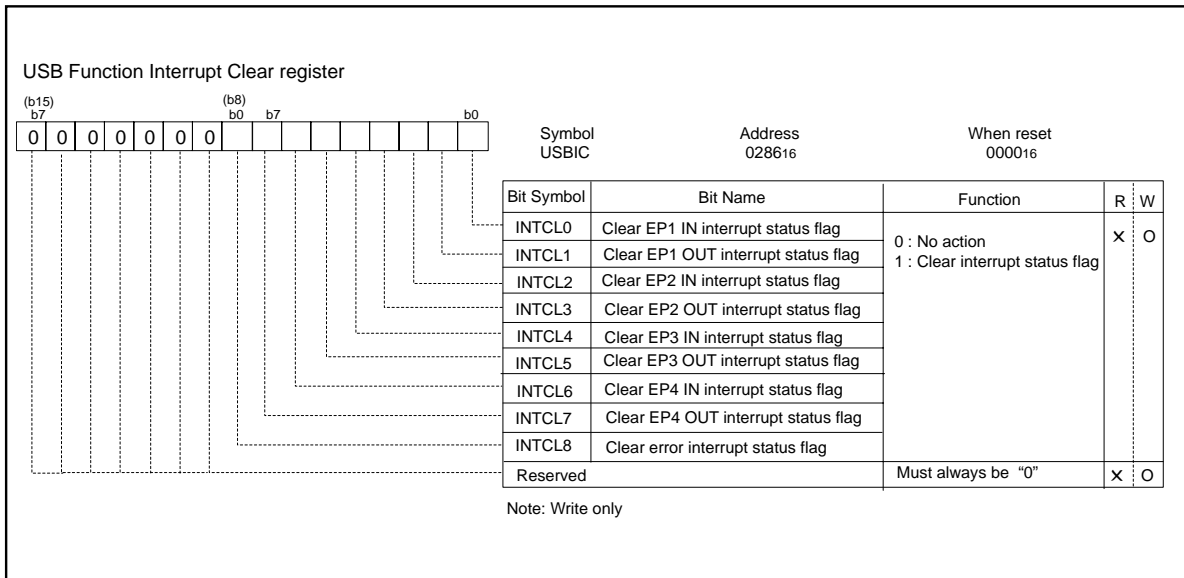


Figure 2.8.22. USB function interrupt clear register

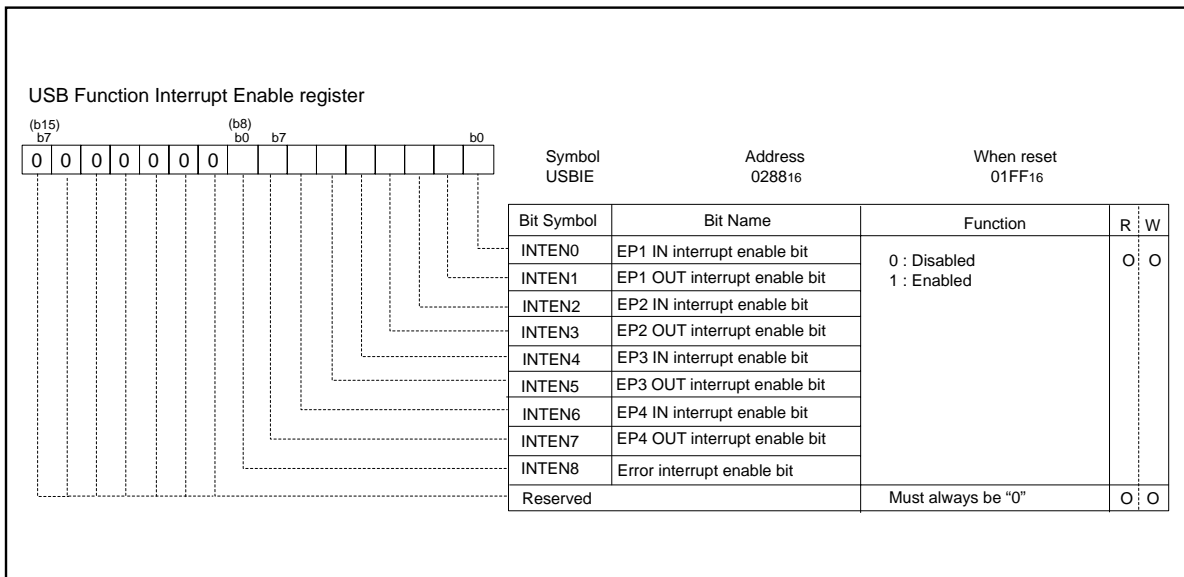


Figure 2.8.23. USB function interrupt enable register

### ● USB frame number register

This register is used to contain 11-bit frame number of SOF token received from the host CPU. This is the read-only register.

The configuration of USB frame number register is shown in Figure 2.8.24.

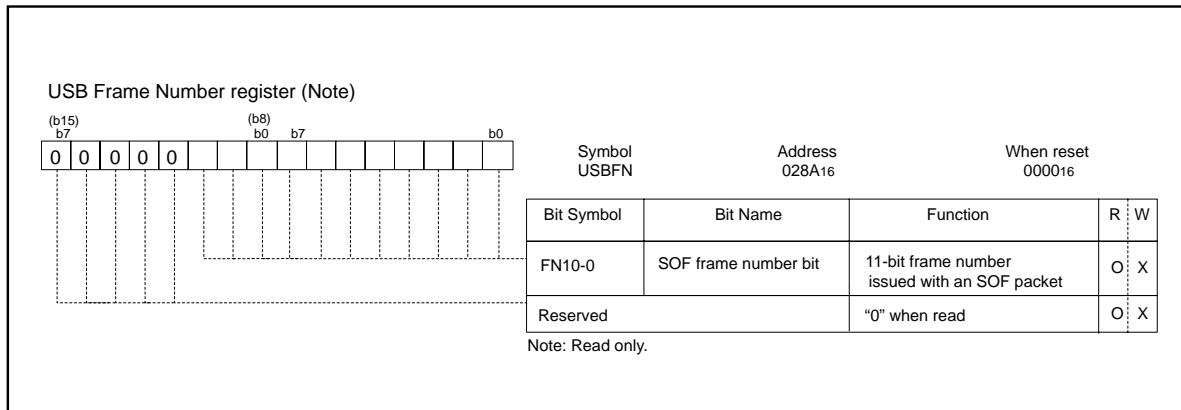


Figure 2.8.24. USB frame number register

**(2) USB Endpoint 0 Interrupt**

In the endpoint 0 interrupt, the interrupt request occurs when the data transmit/receive of endpoint 0 are completed. Set the interrupt priority level by using USB endpoint 0 interrupt control register (EP0IC: address 004616). The interrupt request bit of the EP0IC is set to "1" and the USB endpoint 0 interrupt occurs when one of the following events occur:

- A data is successfully received
- A data is successfully transmitted
- The DATA\_END bit of the EP0CS register is cleared to "0"
- The SETUP\_END flag of the EP0CS register is set to "1"

**● Mask Function of Endpoint 0 Interrupt Factor**

By setting the DATA\_END\_MASK bit of USB endpoint 0 control and status register, the M30245 group can control whether or not to clear the DATA\_END flag as the endpoint 0 interrupt factor. Clearing of the DATA\_END flag is masked at the time of resetting. (The DATA\_END flag is not cleared as an endpoint 0 interrupt factor.)

### (3) USB Function Interrupt

The USB function interrupts include the endpoint  $x(x=1\sim4)$  IN interrupt, endpoint  $x(x=1\sim4)$  OUT interrupt, and error interrupt. An interrupt request occurs on completion of data transmit/receive or on occurrence of an error such as overrun/underrun, setting the status flag which is the factor of the interrupt request inside USB function interrupt status register to "1". When using the USB function interrupt, set the interrupt priority level at USB function interrupt control register (address 005D16) and the corresponding bit of USB function interrupt enable register to "1".

The USB function interrupt involves multiple interrupt request factors. Therefore, during processing of the USB function interrupt, an interrupt request may occur newly and the interrupt status flag can be changed by it. When performing USB function interrupt processing, be sure to first save contents of interrupt status register and to clear the status flag. Then, process the interrupt request that has occurred when the interrupt process has been received based on the saved data value.

#### ● Endpoint $x(x=1\sim4)$ IN Interrupt

In the endpoint  $x(x=1\sim4)$  IN interrupt, when each USB endpoint  $x$  IN interrupt status flag (INTST0,2,4,6) of the corresponding endpoints of USB function interrupt status register is set to "1", an interrupt request occurs. Each flag INTST0, 2, 4, 6 is set to "1" in one of the following cases:

- The corresponding bit of USB endpoint enable register (USBEPEN: address 028E16) is set to "1". (The endpoint is enabled from a disabled state.)
- A data is successfully transmitted
- AUTO FLUSH of hardware has been executed or FLUSH bit of corresponding USB endpoint  $x$  IN control and status register (EPxICS: addresses 029E16, 02A416, 02AA16, 02B016) being set to "1" while one or two packet data exist in the IN FIFO.
- The last ACK for control read transfer is destroyed.

#### ● Endpoint $x(x=1\sim4)$ OUT Interrupt

In the endpoint  $x(x=1\sim4)$  OUT interrupt, when each USB endpoint  $x$  OUT interrupt status flag (INTST1,3,5,7) of the corresponding endpoints of USB function interrupt status register is set to "1", an interrupt request occurs. When a data is successfully received at the corresponding endpoint, each flag INTST1, 3, 5, 7 is set to "1".

#### ● Error Interrupt

In the error interrupt, when the error interrupt status flag (INTST8) of USB function interrupt status register is set to "1", an interrupt request occurs. The INTST8 is set to "1" in one of the following cases:

- The FORCE\_STALL flag of endpoint 0 control and status register (EP0CS) is set to "1".
- The SETUP\_END flag of EP0CS is set to "1".
- The UNDER\_RUN flag of USB endpoint  $x$  IN control and status register (EPxICS: addresses 029E16, 02A416, 02AA16, 02B016) is set to "1". (Due to delay in writing of data to FIFO, underrun has occurred at any one of the IN endpoints that are used for isochronous transfer.)
- The OVER\_RUN flag of USB endpoint  $x$  OUT control and status register (EPxOCS: addresses 02B616, 02BE16, 02C616, 02CE16) is set to "1". (Due to delay in reading of data from FIFO, overrun has occurred at any one of the OUT endpoints that are used for isochronous transfer.)
- The FORCE\_STALL flag of EPxOCS is set to "1".
- The DATA\_ERR flag of EPxOCS is set to "1".

**(4) USB Reset Interrupt**

This interrupt is used for detection of the USB reset. This occurs when the USB function control unit has received reset signal from the host CPU (or detected SE0 on the D+/D- line for at least 2.5 $\mu$ s). At this time, all the USB internal registers are made into reset state. To resume communication, each endpoint needs to be initialized. When using the USB reset interrupt, set the interrupt priority level at USB reset interrupt control register (RSTIC: address 005A16).

**(5) USB Suspend Interrupt**

This interrupt is used for detection of inactivity on the USB bus line. The suspend status flag of USB power management register (USBPM: address 028216) is set when the USB function control unit has detected suspend signal on the USB bus line (or not detected any bus activity on the D+/D- line for at least 3ms). Simultaneously, the USB suspend interrupt occurs.

When using the USB suspend interrupt, set the interrupt priority level at USB suspend interrupt control register (SUSPIC: address 005616).

**(6) USB Resume Interrupt**

This interrupt is used for detection of activity on the USB bus line while the device is in suspend state. When the USB function control unit has received resume signal on the USB bus line (or detected activity on the D+/D- line), the interrupt request occurs, and the USB resume interrupt occurs by setting "1" to resume interrupt request bit of USB resume interrupt control register (RSMIC: address 005816).

When using the USB suspend interrupt, set the interrupt priority level at the RSMIC.

**(7) USB SOF (Start of Frame) Interrupt**

This interrupt is valid to control the isochronous transfer. When a valid SOF PID is detected, receive of an SOF packet is recognized and an interrupt request occurs. The frame number (11 bits) of the SOF packet received from the host is automatically stored in USB frame number register.

In isochronous transfer, P92 can be used as the SOF output pin by setting "1" to USB SOF port select bit of USB control register. (Set P92 to output port.) Every time that the SOF signal is received from the host, the P92 outputs "Low" for about 166ns (two cycles of the 12MHz USB clock.).

When using the USB SOF interrupt, set the interrupt priority level at USB SOF interrupt control register (SOFIC: address 005B16).

**● Artificial SOF Function**

If the SOF packet from the host PC is destroyed due to any cause, and thus the SOF packet is not correctly received within 1 ms of the start of the frame, a virtual SOF receive operation is performed (and an USB SOF interrupt is generated). Even if the SOF packet is destroyed due to any cause, a new frame can be formed by this function without waiting for the next SOF packet. This function operates once to virtually receive an SOF packet after two SOF packets have been received correctly.

**(8) USB Function Interrupt Routine**

This interrupt is used to control data flow. This occurs on completion of data transmit/receive or on occurrence of an error such as overrun/underrun. When using the USB function interrupt, set the interrupt priority level at USB function interrupt control register (address 005D16) and the corresponding bit of USB function interrupt enable register to "1".

The related registers are shown in Figure 2.8.25, and the USB function interrupt routine is shown in Figure 2.8.26.

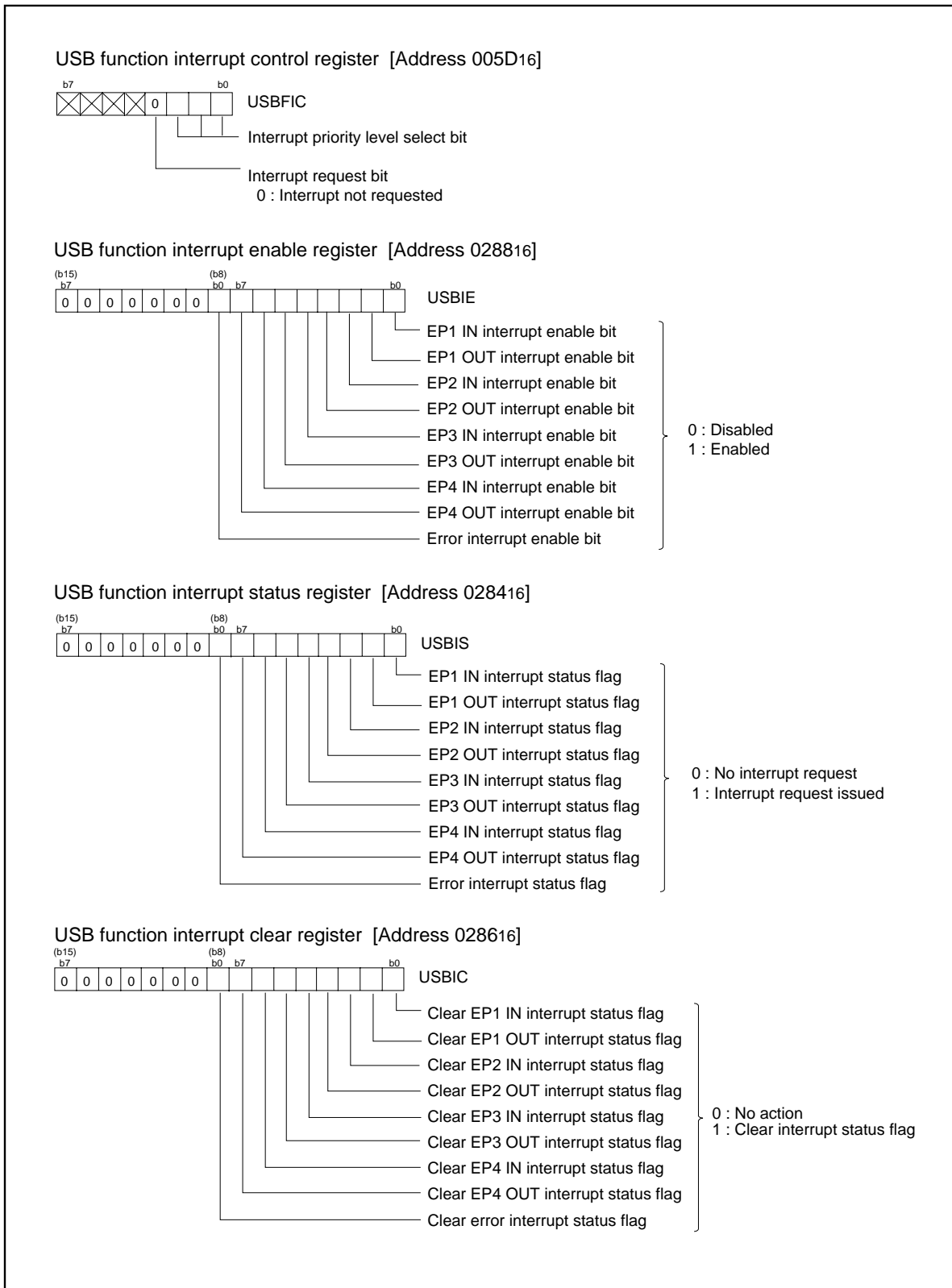
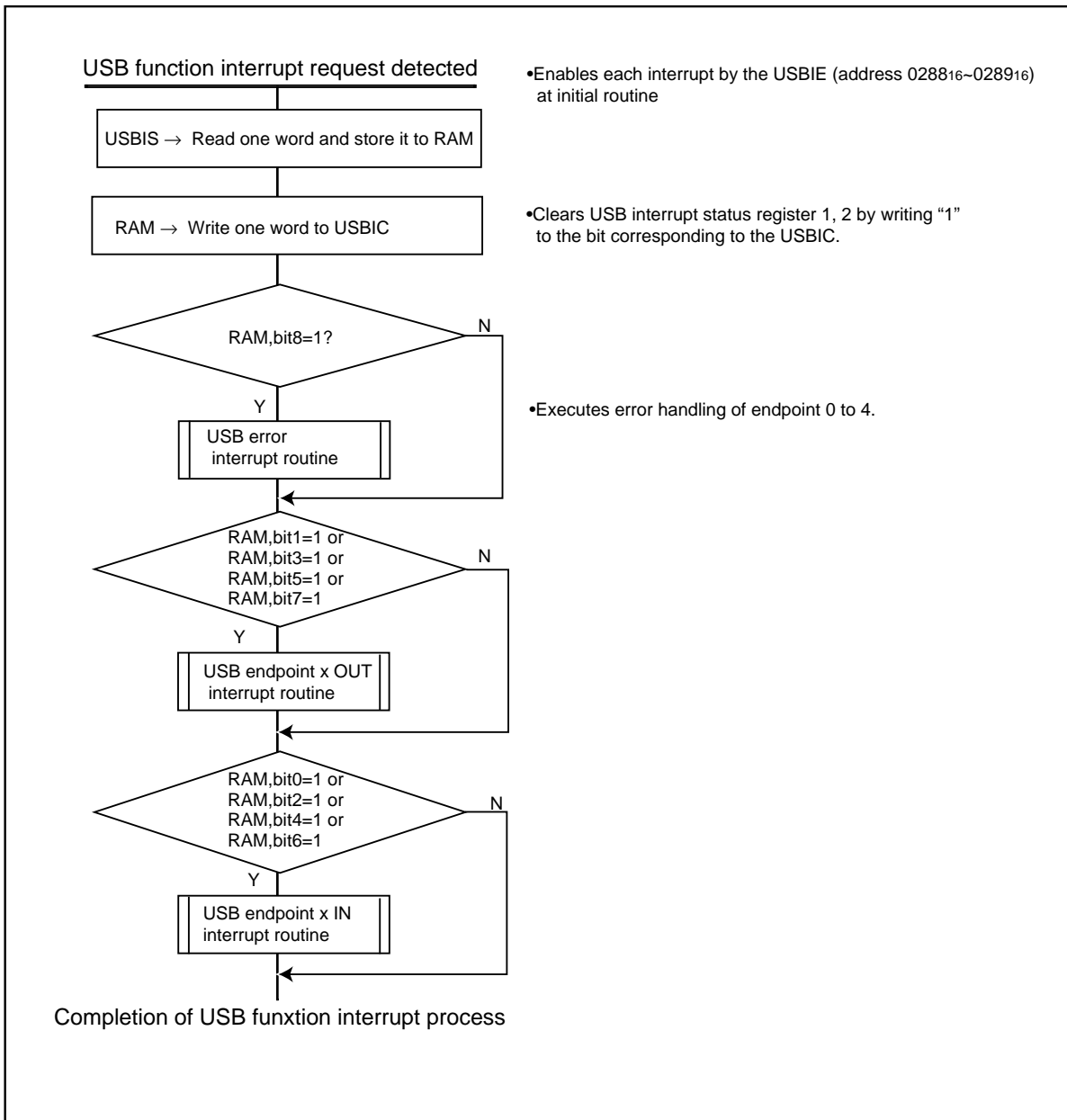


Figure 2.8.25. USB function interrupt related registers





**Figure 2.8.26. USB function interrupt processing routine**

## 2.8.4 USB Operation (Suspend/Resume Function)

The USB device has received the suspend signal from the host CPU following the power input state, and then controls power supply and shifts the state into the suspend state.

And, by receiving the resume signal from the host CPU (or transmitting the resume signal to the host CPU in the case of remote wakeup), it returns to the state before shifting into the suspend state and resumes the USB communication.

This section explains how the M30245 group controls a shift into suspend state/recovery at the time of resume in the state which the USB function control unit is enabled.

### (1) Related Registers

#### ● USB power management register

This register is used to control the suspend/resume by the USB function control unit.

#### • USB Suspend Status Flag

When the USB function control unit does not detect any bus activity on D+/D- line for at least 3ms, the USB suspend status flag is set. Simultaneously, the USB suspend interrupt request occurs. This flag is automatically cleared in the following cases:

- The active signal from the host CPU has been detected on the USB's D+/D- line. (When the resume signal has been received and, simultaneously, the USB resume interrupt request has occurred.)
- Transmission of the resume signal to the host CPU has been completed. (After USB remote wakeup bit being set to "1", when clearing it to "0" to stop resume signal transmission.)

If the USB clock has been disabled during the suspend mode, this flag is not cleared until after the USB clock is re-enabled.

#### • USB remote wakeup bit

When the USB suspend signal status flag is set to "1", the USB function control unit transmits the resume signal to the host CPU while setting "1" to USB remote wakeup bit.

Set USB remote wakeup bit to "1" in order to transmit the resume signal to the host CPU and to return to the previous state (remote wakeup) in the USB suspend state. Retain this bit at "1" for min. 1ms to max. 15ms before completing the resume signal transmission by clearing to "0".

The configuration of USB power management register is shown in Figure 2.8.27.

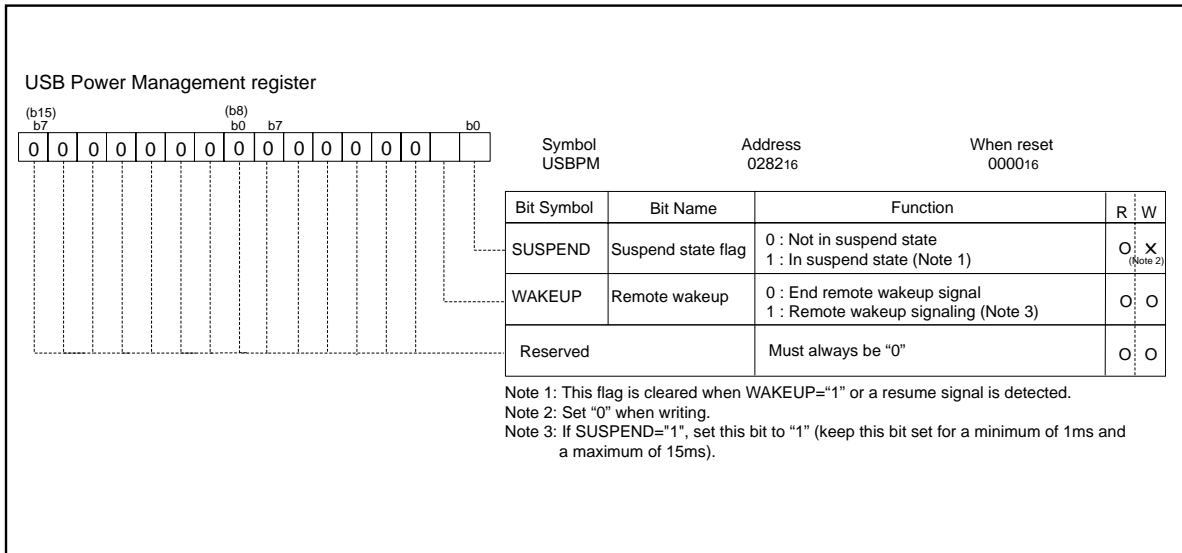


Figure 2.8.27. USB power management register

## (2) USB Suspend Function

In the M30245 group, the USB suspend status flag (SUSPEND) of USB power management register (address 0282<sub>16</sub>) is set to "1" when the suspend signal has been received from the host CPU (or not detected any bus activity on the D+/D- line for at least 3ms). Simultaneously, the USB suspend interrupt request occurs.

To shift into the USB suspend state, control the USB function control unit in the following procedure. Further, for changes in frequency synthesizer control register (address 03DC<sub>16</sub>) or system clock control register1 (address 0007<sub>16</sub>), clearing the corresponding bit of protect register (address 000A<sub>16</sub>) is required.

### ● USB Suspend Mode Control

- 1: Set USB clock enable bit of USB control register to "0". Do not write to USB internal registers (other than the USBC, USBAD, and frequency synthesizer related registers) when the USB clock has been disabled in the suspend state.
- 2: During the bus power supply operation as a low-power device, control it to reduce the total driving current to 500 $\mu$ A or below (or 2.5mA or below when remote wakeup has been enabled as a high-power device by the host CPU). For details of the power control in suspend, refer to USB2.0 specification.
- 3: Set frequency synthesizer enable bit of frequency synthesizer control register to "0".
- 4: Set the return interrupt from the USB suspend state. Set USB resume interrupt control register (address 0058<sub>16</sub>). (When remote wakeup has been enabled, enable interrupt control register of the peripheral functions used in remote wakeup.)
- 5: Set I flag to "1".
- 6: Stop the system clock by setting all clock stop control bit (bit 0 of CM1) to 1.
- 7: During the bus power supply operation, after setting the low-power consumption mode by disabling interrupts that are not used in return from the USB suspend state, etc., execute the low-power consumption mode.

Note: When the device is in self-powered operation, the above control is not required.

### (3) USB Resume Function

#### ● Returning Routine from USB Suspend State

To return from the USB suspend state, the M30245 group uses the USB resume interrupt occurred by receiving the resume signal from the host or the interrupt for remote wakeup for transmitting the resume signal to the host.

#### - Returning by Resume Interrupt

When the resume signal is received from the host CPU during the USB suspend state (when detected any bus activity on D+/D- line in suspend detect state), the USB resume interrupt request occurs, setting "1" to interrupt request bit of USB resume interrupt control register (address 005816). When the USB clock is operated, the USB suspend status flag is automatically set to "0" at this time. For returning from the suspend state by the USB resume interrupt, follow the procedure below:

- 1: Return the USB function control unit. (Refer to the next page.)
- 2: Enable other functions as circumstances demand.

#### - Returning by Remote Wakeup

When clock operation is started by the remote wakeup interrupt (other than the USB resume interrupt) during the USB suspend state, transmit the resume signal to the host CPU as follows:

- 1: Return the USB function control unit. (Refer to the next page.)
- 2: Set USB remote wakeup bit to "1" and transmit the resume signal to the host CPU. (Retain "1" for min. 1ms to max. 15ms.)
- 3: Set USB remote wakeup bit to "0" and complete the resume signal transmission. The USB suspend status flag is automatically cleared at this time.

Also, when returning from the stop mode, the main clock dividing ratio has been set to 8-dividing mode, for which resetting is required. Wait for enough oscillation stabilization time before resetting main clock division select bit of system clock control register 0 (address 000616). (For details, refer to "Clock-Generating Circuit" of Chapter 1 "Hardware".)

**- Returning Routine of USB Function Control Unit**

To return from the USB suspend state to the previous state, perform return control of the USB function control unit as follows.

Further, clearing the bit of protect register (address 000A16) is required for changes in frequency synthesizer control register (address 03DC16).

- 1: Set frequency synthesizer enable bit of frequency synthesizer control register to "1".
- 2: Wait for 3ms.
- 3: Check that frequency synthesizer lock status bit of frequency synthesizer control register is set to "1". When this bit has been set to "0", wait for 0.1ms, and then, check again. Repeat the re-check until the bit becomes "1".
- 4: Set USB clock enable bit of USB control register to "1".

Do not write to USB-related registers (other than the USBC, USBAD, and frequency synthesizer related registers) when the USB clock has been disabled in the suspend state.

#### (4) USB Suspend Interrupt Request Processing Routine

When using the USB suspend interrupt, set USB suspend interrupt control register (address 005616). In the USB suspend interrupt, when the USB suspend status flag (SUSPEND) of USB power management register is set to "1", the interrupt request occurs.

The USB suspend interrupt request processing routine is shown in Figure 2.8.28 and Figure 2.8.29.

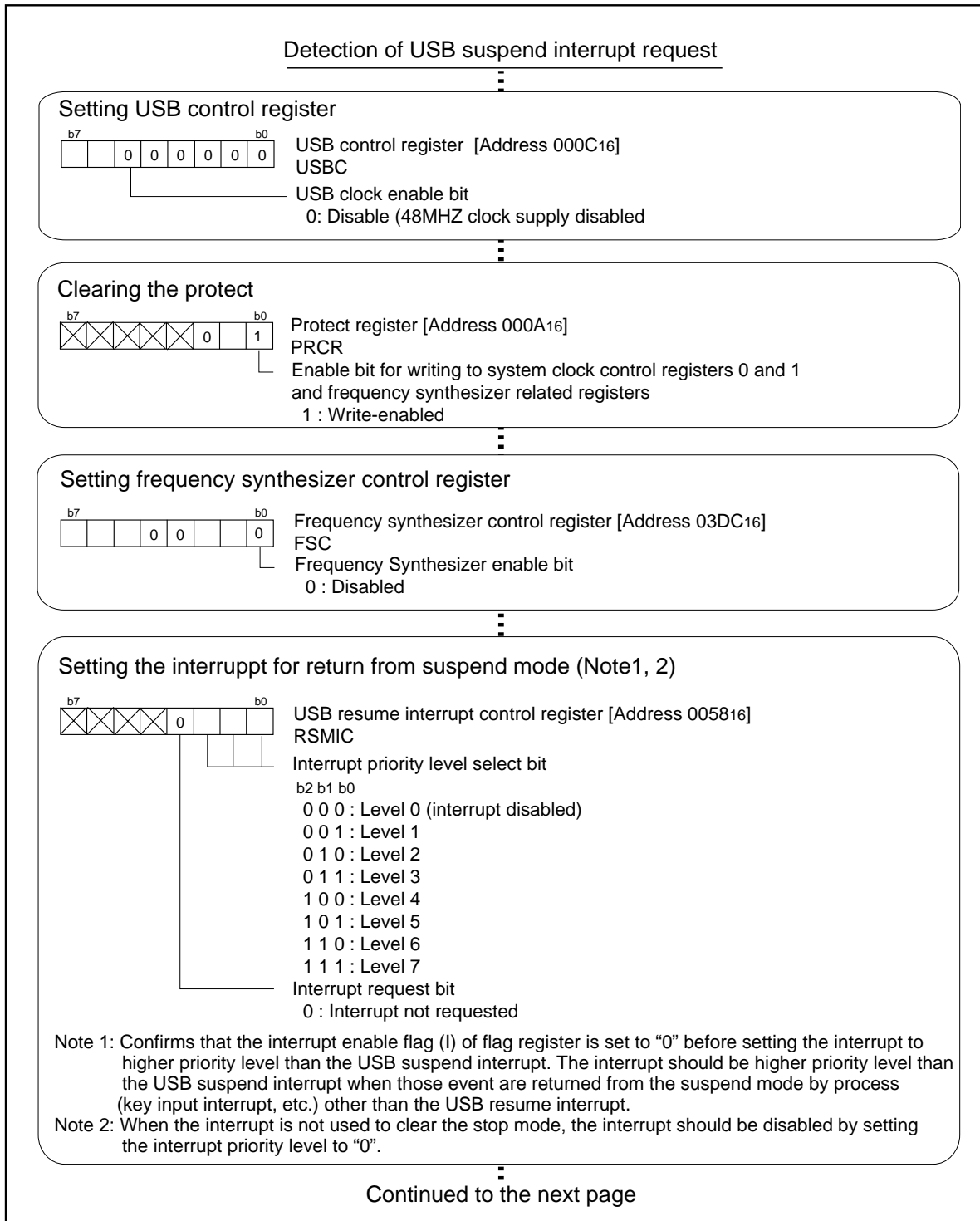
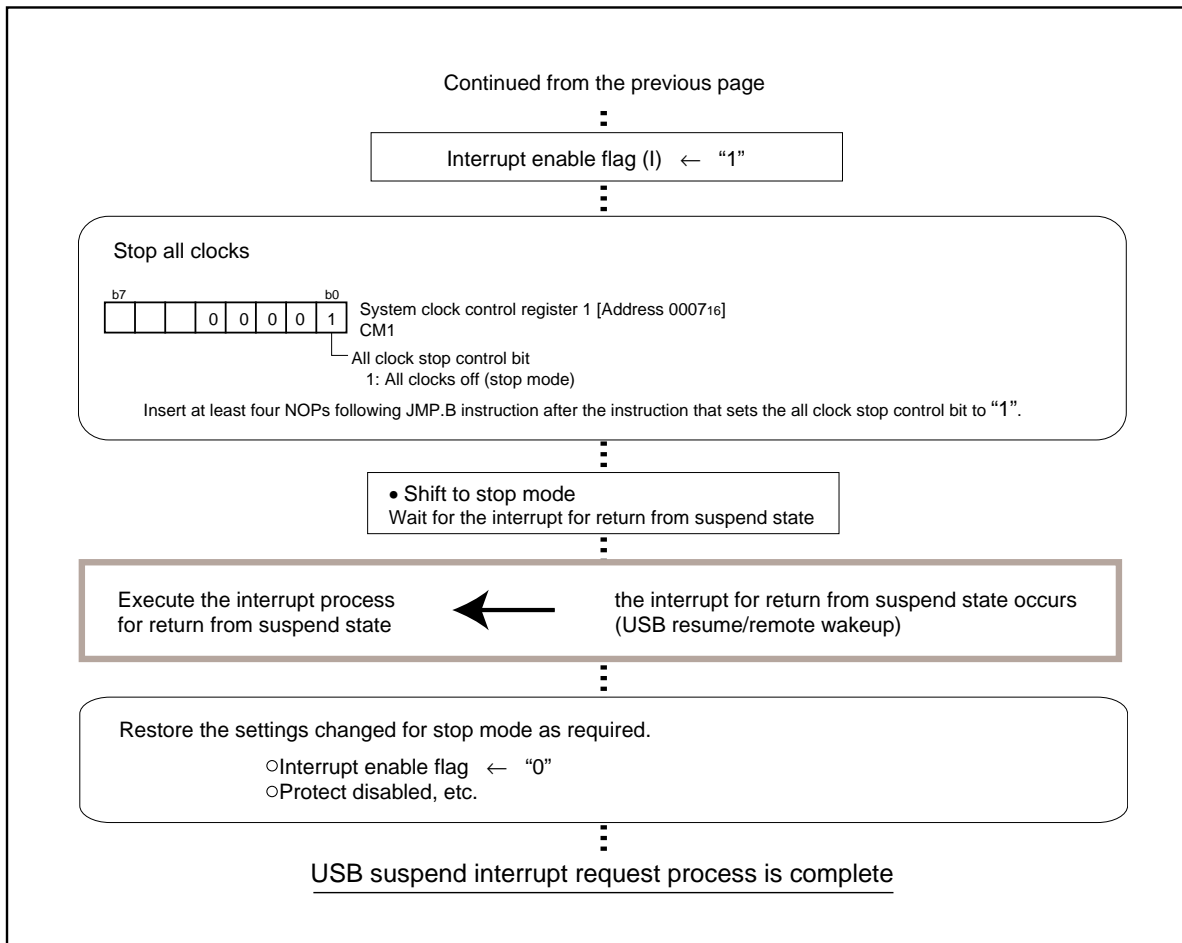


Figure 2.8.28. USB suspend interrupt request processing routine (1)



**Figure 2.8.29. USB suspend interrupt request processing routine (2)**

#### (5) USB Resume Interrupt Request Processing Routine

When the resume signal is received from the host CPU during the USB suspend mode (when detected any bus activity on D+/D- line in suspend detect state), the USB resume interrupt request occurs. The USB suspend status flag is automatically set to "0" at this time.

The USB resume interrupt request processing routine is shown in Figure 2.8.30.



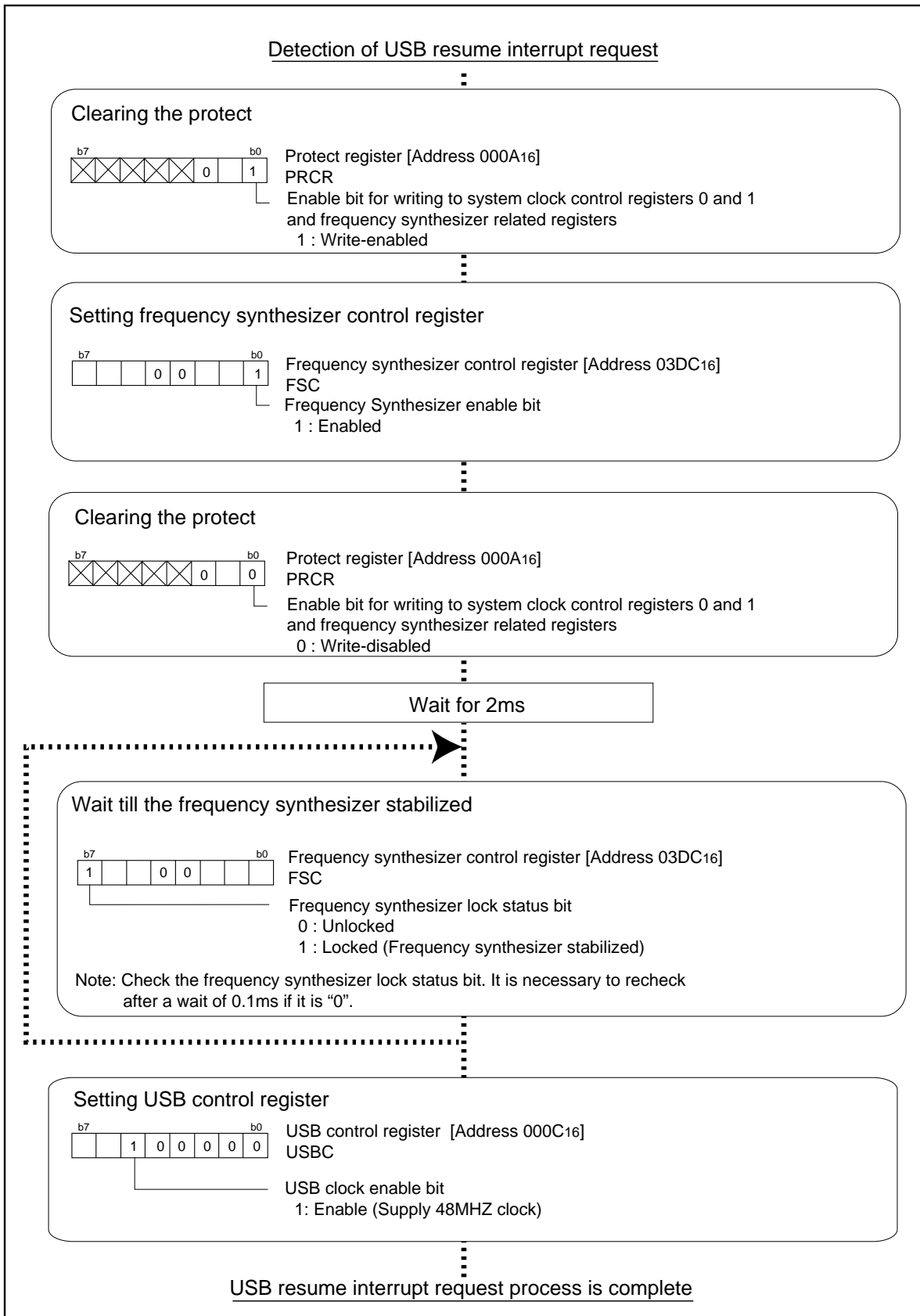


Figure 2.8.30. USB resume interrupt request processing routine

## 2.8.5 USB Operation (Endpoint 0)

Endpoint 0 is used only for control transfer.

Endpoint 0 FIFO consists of total 256 bytes including IN (transmit) FIFO and OUT (receive) FIFO each respectively of 128 bytes. The starting position is allocated from the 3072nd byte to the 3327th byte of the endpoint FIFO. Both the endpoint 0 FIFO size and the starting position are fixed. The FIFO size to be used is determined by the USB endpoint 0 maximum packet size.

The packet data received from the host CPU are written in endpoint 0 OUT FIFO. When a data receive request is issued by the host CPU while data already exists in the OUT FIFO, responds with NAK automatically. When packet data are transmitted to the host CPU, the transmit data are written in the endpoint 0 IN FIFO. When a data transmit request is issued by the host CPU before the data are written in IN FIFO, responds with NAK automatically. A packet data can realize higher transmit/receive by enabling continuous transfer.

When an error is detected in control transfer, responds with STALL automatically and the error detection is reported. Based on the status of the endpoint 0 communication, data transmit/receive is controlled in accordance with the device request which is received from the host CPU.

### (1) Related Registers

#### ● USB address register

USB address register maintains 7 bits addresses of the USB function control unit that are allocated by the host CPU. The USB function control unit of the M30245 group responds to the token packet for the address retained in this register.

When the USB function control unit is in the initial state or has received a reset signal from the host CPU, this register value is the default address "000016". When the USB block has been disabled (bit 7 of USB control register is set to "0"), this register value is the address "000016".

After receiving the SET\_ADDRESS request from the host CPU, rewrite USB address register and update the address.

For rewriting USB address register, follow the procedure below:

- When the device is in the default state (USB address register value is "000016"):
  - 1: When USB address register has received the SET\_ADDRESS request from the host CPU, store the new address data in the USB address register.
  - 2: When the status phase of the SET\_ADDRESS request is completed, USB address register is automatically rewritten into the address written in above-mentioned 1.: When the status phase is not normally completed, USB address register is not rewritten.
- When the device is in the address state (USB address register value is other than "000016"):
  - 1: When USB address register has received the SET\_ADDRESS request from the host CPU, confirm that the status phase of SET\_ADDRESS request completes.
  - 2: Store the new address data in USB address register. USB address register is rewritten into the new address data.

The configuration of USB address register is shown in Figure 2.8.31.

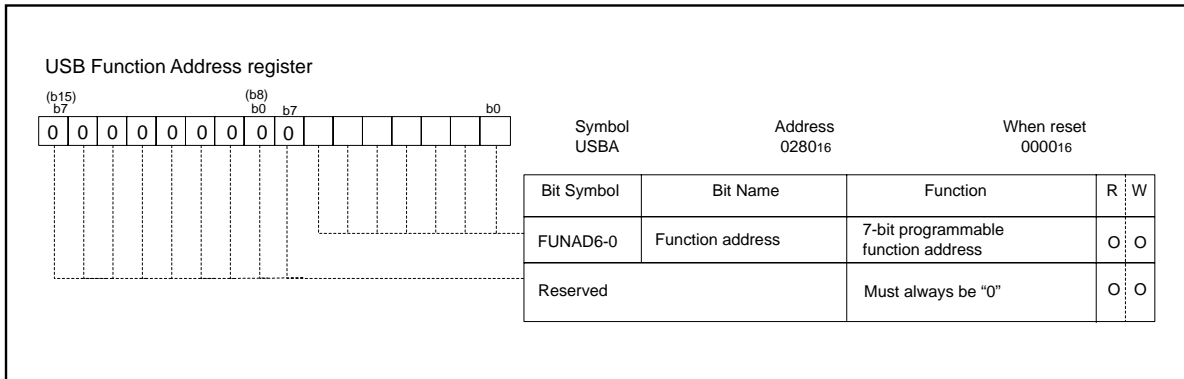


Figure 2.8.31. USB address register

### ● USB endpoint 0 control and status register

USB endpoint 0 control and status register consists of the bits concerning control information and status information of endpoint 0.

#### • OUT\_BUF\_RDY Flag

This flag shows the status of OUT FIFO.

The OUT\_BUF\_RDY flag is set to "1" in the following cases:

- The setup packet is received.
- One data packet is received from the host CPU in the data phase.

Set the OUT\_BUF\_RDY flag to "0" by setting "1" to CLR\_OUT\_BUF\_RDY bit after reading the OUT FIFO.

#### • IN\_BUF\_RDY Flag

This flag shows the status of IN FIFO.

When this flag is set to "1", shows that data to be transmitted to the host CPU exists in FIFO.

When transmission of the IN FIFO data is completed or when the SETUP\_END flag is set to "1", this flag is cleared to "0".

#### • SETUP Flag

When the setup packet is received from the host CPU, this flag is set to "1".

The OUT\_BUF\_RDY flag is also set to "1" at this time.

This flag is cleared by setting "1" to CLR\_SETUP bit.

#### • DATA\_END Flag

This flag shows the status phase control of control transfer.

After the status phase is started or when a new SETUP packet is received, this flag is automatically set to "0".

When DATA\_END\_MASK bit is set to "1" (at the time of resetting), the DATA\_END flag is always set to "0", and endpoint 0 interrupt factor does not occur by clearing the DATA\_END flag to "0".

#### • FORCE\_STALL Flag

This flag shows the error occurrence in control transfer.

This flag is set to "1" for reporting an error when at least one of the following conditions occurs:

- IN token without SETUP stage is received.
- An incorrect data toggle is received in the STATUS stage. (DATA0 is used.)
- An incorrect data toggle is received in the SETUP stage. (DATA1 is used.)
- Data exceeding the one specified in the SETUP stage are required. (IN token is received after the DATA\_END flag is set.)

- Data exceeding the one specified in the SETUP stage are required. (An OUT token is received after the DATA\_END flag is set.)
- Data exceeding the one specified are received in USB endpoint 0 MAXP register.

Except for when an incorrect data toggle is received in the SETUP stage, on occurrence of the above condition, STALL is transmitted to the IN/OUT token with a problem. When an incorrect data toggle is received in the SETUP stage, ACK is returned to the SETUP stage and STALL is returned to next IN/OUT token.

The STALL handshake occurring by the above condition completes the control transfer in operation by being transmitted to one transaction. The packet after STALL handshake is regarded as the start of new control transfer. Set this flag to "0" by writing "1" to CLR\_FORCE\_STALL bit.

- **SETUP\_END flag**

This flag shows the interrupt in control transfer.

This flag is set to "1" when at least one of the following conditions occurs:

- Transmission of the data which had amount of data set by setup phase during data phase processing is completed. (The status phase is started before the DATA\_END flag is set.)
- A new SETUP packet is received before status phase is completed.

When this flag is set to "1" and transmit data to the host CPU exists, IN\_BUF\_RDY bit is cleared to "0" and the IN FIFO data is destroyed. Discontinue access to FIFO and process the preceding setup.

Also, when a new SETUP packet is received right after the SETUP\_END flag is set to "1" (when the next new SETUP packet is received before the data phase or the status phase is completed), the SETUP flag and the OUT\_BUF\_RDY flag in addition to the SETUP\_END flag are set to "1", indicating that a new SETUP packet data exists in OUT FIFO.

Set the SETUP\_END flag to "0" by writing "1" to CLR\_SETUP\_END bit.

- **CLR\_OUT\_BUF\_RDY bit**

This bit controls clearing of the OUT\_BUF\_RDY flag.

This bit is set to "1" after reading the data packet from OUT FIFO. When this flag is written to "1", the OUT\_BUF\_RDY flag is cleared to "0".

When the OUT\_BUF\_RDY flag is set to "1" by receiving the SETUP token, the USB function control unit responds with NAK to the data request from the host CPU.

Until decoding of request data from the host CPU is completed, do not set this bit to "1" (nor the OUT\_BUF\_RDY flag is set to "0".)

- **SET\_IN\_BUF\_RDY bit**

This bit controls setting of the IN\_BUF\_RDY flag to "1".

Completion of one buffer data write is notified to the USB function control unit.

Set this bit to "1" after writing the data packet to IN FIFO. When this bit is written to "1", the IN\_BUF\_RDY flag is set to "1".

- **CLR\_SETUP bit**

This bit controls clearing of the SETUP flag.

Set this bit to "1" after decoding the SETUP packet. When this bit is written to "1", the SETUP flag is cleared to "0".

- **SET\_DATA\_END bit**

This bit controls setting of the DATA\_END flag to "1".

When the last data has been written in IN FIFO in the IN data phase or when the last data has been read from OUT FIFO in the OUT data phase, set this bit to "1". When this bit is set to "1", the DATA\_END flag is set to "1". At this time simultaneously, set the CLR\_OUT\_BUF\_RDY bit or SET\_IN\_BUF\_RDY bit to "1".

Completion of processing of the data which had amount of data set by setup phase is notified to the USB function control unit, and the process shifts into status phase processing.

- **CLR\_FORCE\_STALL bit**

This bit controls clearing of the FORCE\_STALL flag.

When this bit is written to "1", the FORCE\_STALL flag is cleared to "0".

- **SEND\_STALL bit**

This bit controls the STALL response to the host CPU.

To respond with STALL when an invalid request has received from the host CPU, set this bit to "1" simultaneously as CLR\_OUT\_BUF\_RDY bit is set to "1".

When this bit is set to "1", the USB function control unit transmits, to the host CPU, the STALL handshake concerning all the IN/OUT transactions. When a new valid SETUP packet is received, clear a data by writing "0" in this bit.

- **DATA\_END\_MASK bit**

This bit controls whether the DATA\_END flag clear becomes an endpoint 0 interrupt factor.

When this bit is set to "1", clearing the DATA\_END flag does not become an endpoint 0 interrupt factor. This bit is set to "1" at the time of reset.

The configuration of USB endpoint 0 control and status register is shown in Figure 2.8.32.

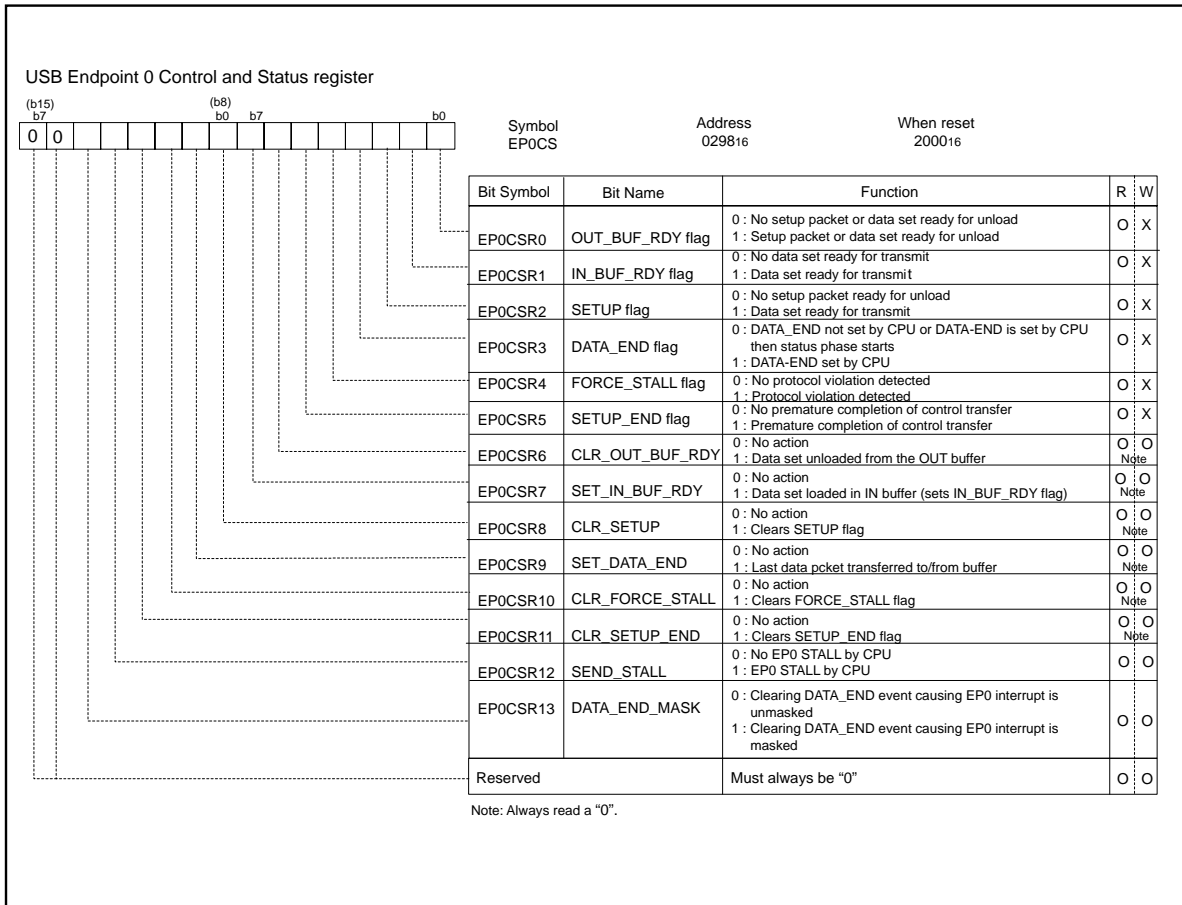


Figure 2.8.32. USB endpoint 0 control and status register

### ● USB endpoint 0 MAXP register

This register indicates the IN/OUT maximum packet size of endpoint 0.

When a GET\_DESCRIPTION request is received from the host CPU, write to this register to change the IN/OUT maximum packet size value of endpoint 0.

Set the packet size value (8, 16 or 32 bytes) specified by control transfer. The default value is 8 bytes.

The configuration of USB endpoint 0 MAXP register is shown in Figure 2.8.33.

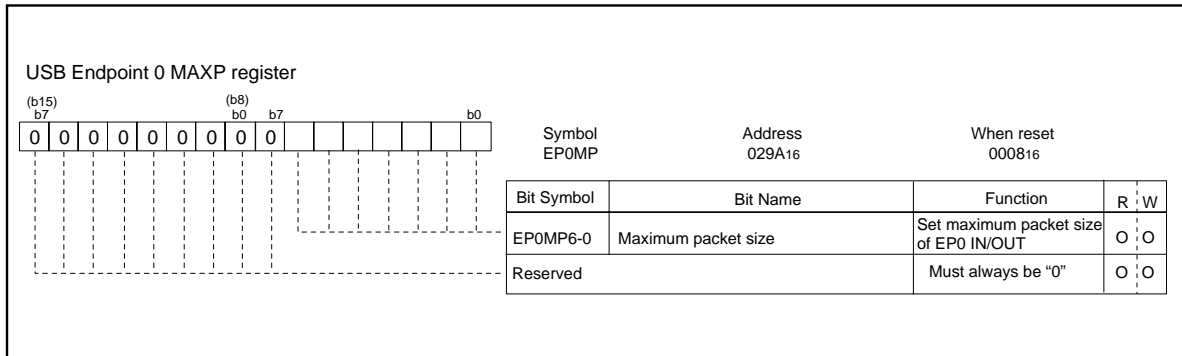


Figure 2.8.33. USB endpoint 0 MAXP register

### ● USB endpoint 0 OUT write count register

This register contains the number of bytes of the current data set in the OUT FIFO. When the USB function control unit completes the data packet receive from the host CPU, set the value of this register. When one buffer data receive completes, read this register and determine the number of bytes to be read from OUT FIFO. This register value is not decremented even if the data is read from OUT FIFO.

When CLR\_OUT\_BUF\_RDY bit of the EP0CSR is set to "1", this register value is cleared to "0".

The configuration of USB endpoint 0 OUT write count register is shown in Figure 2.8.34.

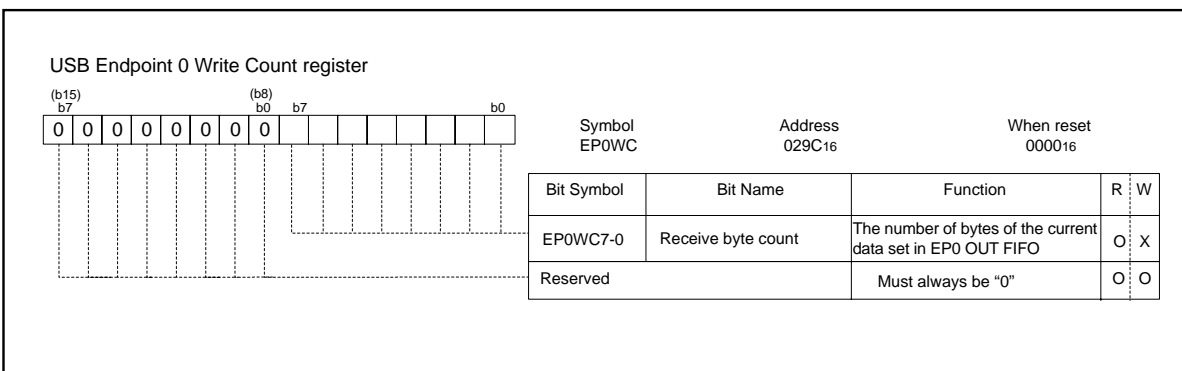


Figure 2.8.34. USB endpoint 0 OUT write count register

## (2) Control Transfer: Endpoint 0 Receive

The endpoint 0 receives the packet data from the host CPU in the setup stage or the data stage by the control write transfer. When the receive of a valid SETUP packet or a data packet completes, the SETUP flag and the OUT\_BUF\_RDY flag are automatically set to "1", and the number of bytes of receive data are set in USB endpoint 0 OUT write count register (address 029C16). Read the data of only amount equal to received byte count from the endpoint 0 OUT FIFO. Every time that one-byte data is read from OUT FIFO, the internal write pointer is automatically decremented by "2" in word access and by "1" in byte access. The contents of internal write pointer cannot be read.

When the data read from OUT FIFO is completed, simultaneously set "1" to CLR\_OUT\_BUF\_RDY bit and SET\_DATA\_END bit. (When the SETUP packet is received, clearing the SETUP flag by setting "1" to CLR\_SETUP bit is required.) Therefore, the OUT\_BUF\_RDY flag is cleared and the DATA\_END flag is set to "1".

The USB function unit proceeds to the status phase process when the DATA\_END flag is set to "1". When the status phase completes, the DATA\_END flag is cleared to "0".

Manage the stage of control transfer by software.

When the SETUP packet is received, the USB endpoint 0 interrupt occurs regardless of setting of continuous transfer mode enable bit (The OUT\_BUF\_RDY flag and the SETUP flag are set to "1").

### ● Example of one packet data receive procedure

- 1: Check that one packet data is received in OUT FIFO.
- 2: Read the number of bytes of receive packet data from USB endpoint 0 OUT write count register. Determine the amount of data to read from OUT FIFO.
- 3: Read the data of only amount equal to determined in the above-mentioned 2: from OUT FIFO. To analyze the received data, the subsequent stage and operation are determined based on the read data.
- 4: With CLR\_OUT\_BUF\_RDY bit being set to "1", the OUT\_BUF\_RDY flag is cleared to complete fetch of the receive one packet, and manage the next stage control. At this time, when the SETUP packet is received, the SETUP flag is also cleared by setting "1" to CLR\_SETUP bit.
  - For shifting into the status stage even if the next data to be received or transmitted does not exist, simultaneously set CLR\_OUT\_BUF\_RDY bit (and also CLR\_SETUP bit for the SETUP packet) and SET\_DATA\_END bit to "1".
  - For responding with STALL response to the next token, simultaneously set CLR\_OUT\_BUF\_RDY bit (and also CLR\_SETUP bit for the SETUP packet) and SEND\_STALL bit to "1".
  - When the valid new SETUP packet is received after setting SEND\_STALL bit, clear SEND\_STALL bit and set CLR\_OUT\_BUF\_RDY bit to "1" (and also CLR\_SETUP bit for the SETUP packet).



### (3) Control Transfer: Endpoint 0 Transmit

The endpoint 0 transmits the packet data to the host CPU in the data stage by the control read after completion of receive request analysis process in the setup stage.

Write one packet data to be transmitted in IN FIFO. Every time that one-byte data is written in IN FIFO, the internal write pointer is automatically incremented by "2" in word access and by "1" in byte access. The contents of internal write pointer cannot be read. When the data write in IN FIFO is completed, set IN\_BUF\_RDY flag to "1" by setting "1" to SET\_IN\_BUF\_RDY bit. When an empty packet (with 0 data length) is transmitted, data is not written in IN FIFO and SET\_IN\_BUF\_RDY bit is set to "1".

At this time, one packet transmission is prepared and is transmitted by the USB function control unit in the next IN token.

The IN\_BUF\_RDY flag is automatically set to "0", when one packet data transmission is completed to the host CPU (or on receiving ACK) or when the SETUP\_END flag is set to "1". After writing the last data packet in IN FIFO, set SET\_IN\_BUF\_RDY bit to "1" and, simultaneously set SET\_DATA\_END bit to "1". Both the IN\_BUF\_RDY flag and DATA\_END flag are set to "1". When the DATA\_END flag becomes "1" after completion of transmission of the last data packet, the USB function control unit proceeds to the status phase processing.

When the status phase is completed, the DATA\_END flag is cleared to "0".

Manage the stage of control transfer by software.

#### ● Example of one packet data transmit procedure

- 1: Check that the packet data does not exist in IN FIFO (the IN\_BUF\_RDY flag is "0") before writing the data of the 2nd packet and after of data stage.
- 2: The data is written in IN FIFO based on the amount specified on the SETUP stage.  
When an empty packet (with 0 data length) is transmitted, the data is not written in IN FIFO.  
The subsequent stage and operation are determined.
- 3: With SET\_IN\_BUF\_RDY bit being set to "1", one packet transmission is prepared, and the next stage control is managed.
  - For shifting into the status stage even if the next data to be transmitted does not exist, simultaneously set SET\_IN\_BUF\_RDY bit and SET\_DATA\_END bit.
  - When the next empty packet is transmitted, set SET\_IN\_BUF\_RDY bit to "1" and continue transmitting processing.

**(4) Control Transfer: Example of Standard Device Request Receive**

The control transfer includes the setup stage, data stage and status stage.

Which one of write transfer, read transfer and no data transfer is executed in the data stage is determined by the content of the setup data acquired in the setup stage.

Examples of the receive processing routine of the SET\_ADDRESS request and the GET\_CONFIGURATION request are described.

For rewriting USB address register when the SET\_ADDRESS request is received, follow the procedure below:

● **When the device is in the default state (USB address register value is “0”):**

- 1: When USB address register is received the SET\_ADDRESS request from the host CPU, store the new address data in the USB address register.
- 2: When the status phase of the SET\_ADDRESS request is completed, USB address register is rewritten into the address written in above-mentioned 1:. When the status phase is not normally completed, USB address register is not rewritten.

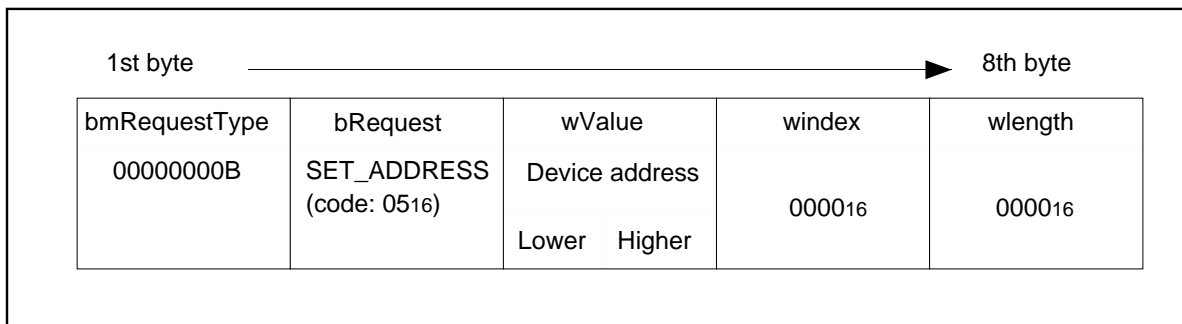
● **When the device is in the address state (USB address register value is other than “0”):**

- 1: When USB address register is received the SET\_ADDRESS request from the host CPU, confirm that the status phase of SET\_ADDRESS request completes.
- 2: Store the new address data in USB address register.

The USB function control unit applies this address to all the subsequent device accesses.

The SET\_ADDRESS request is shown in Figure 2.8.35, the device address acquisition processing routine of USB SET\_ADDRESS request is shown in Figure 2.8.36 and Figure 2.8.37, the device configuration notification processing routine of GET\_CONFIGURATION request is shown in Figure 2.8.38 and Figure 2.8.39.

Describe these processing to endpoint 0 interrupt processing.



**Figure 2.8.35. SET\_ADDRESS Request**

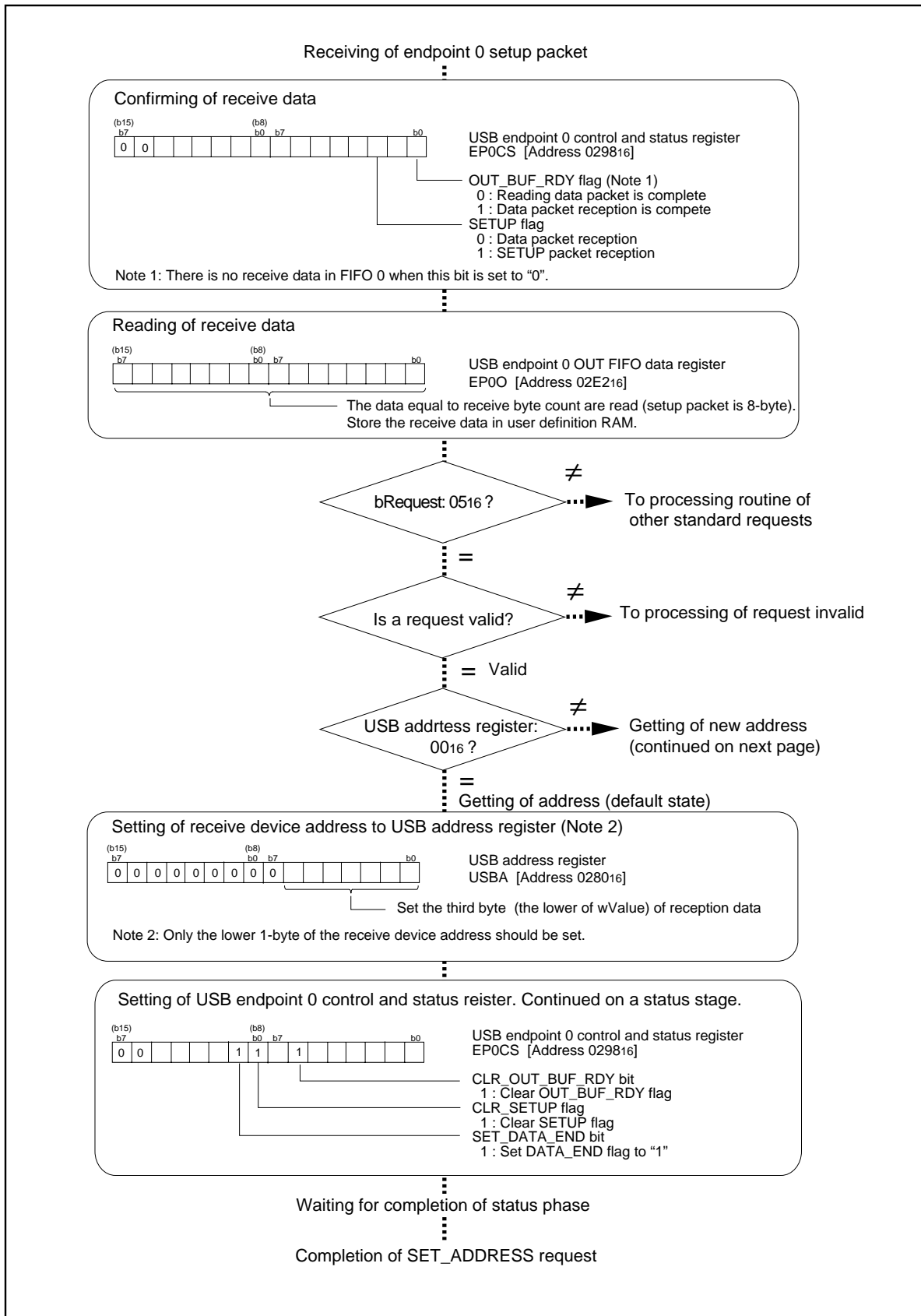


Figure 2.8.36. Processing routine (1) for getting device address when receiving SET\_ADDRESS request

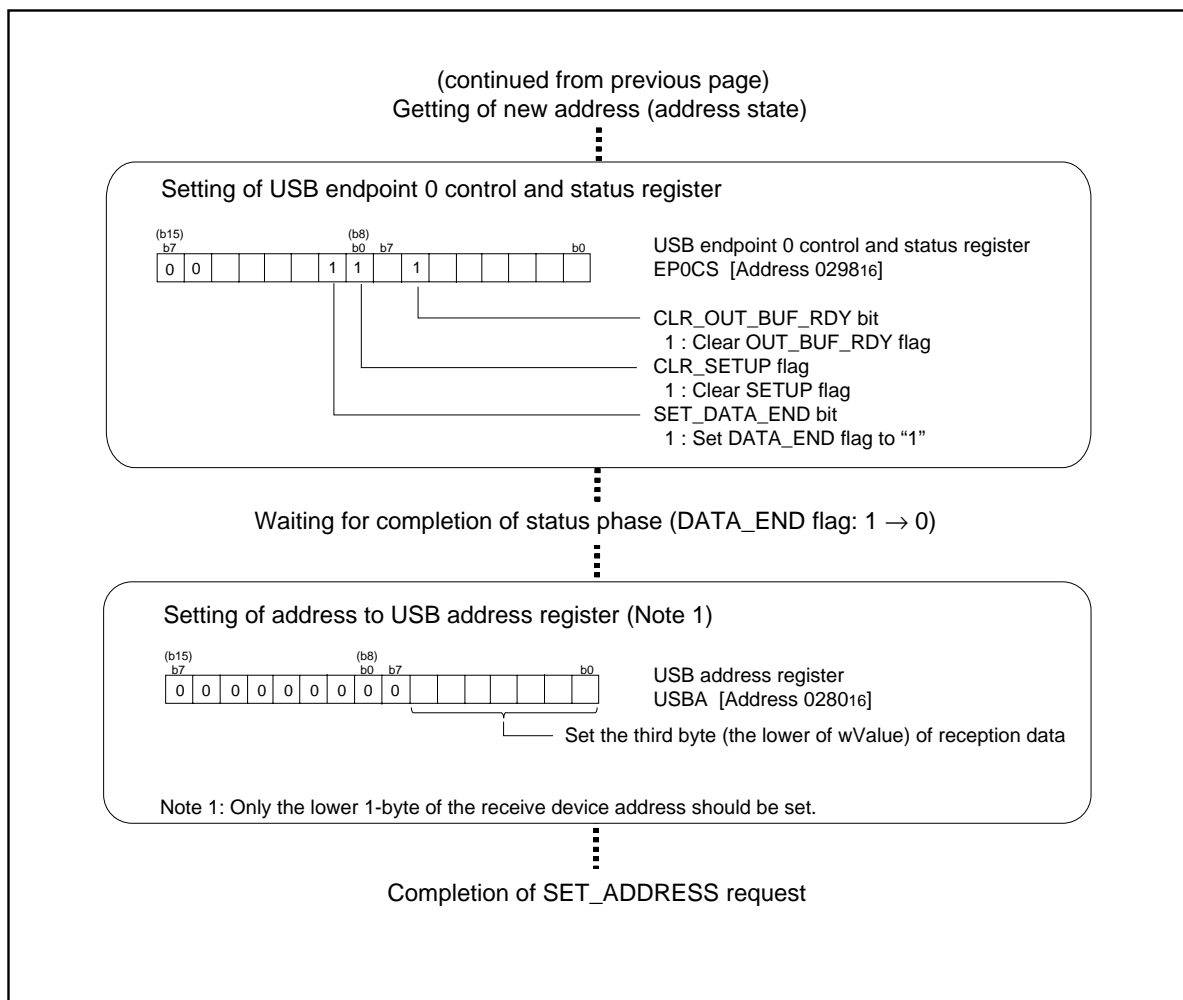
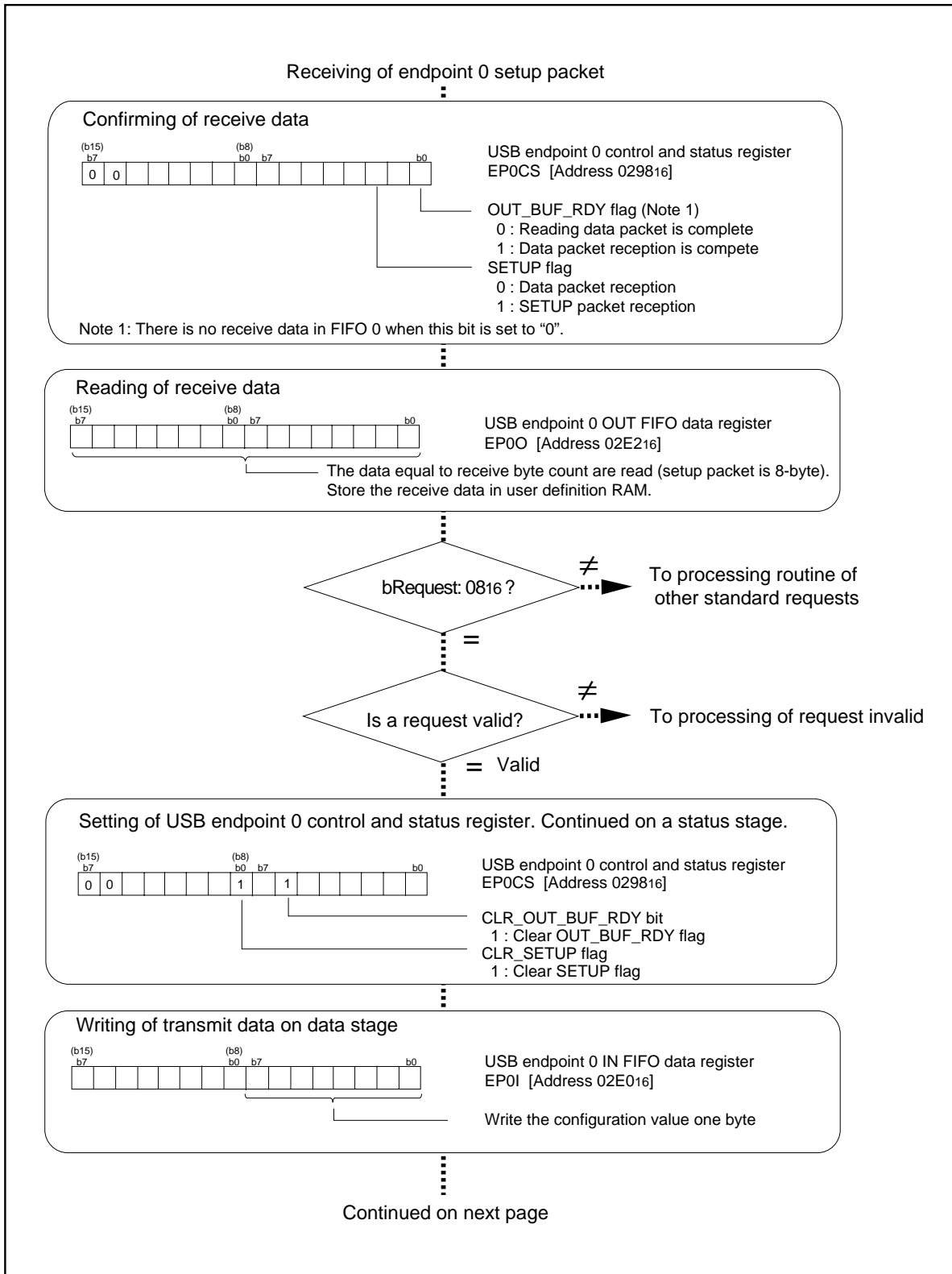
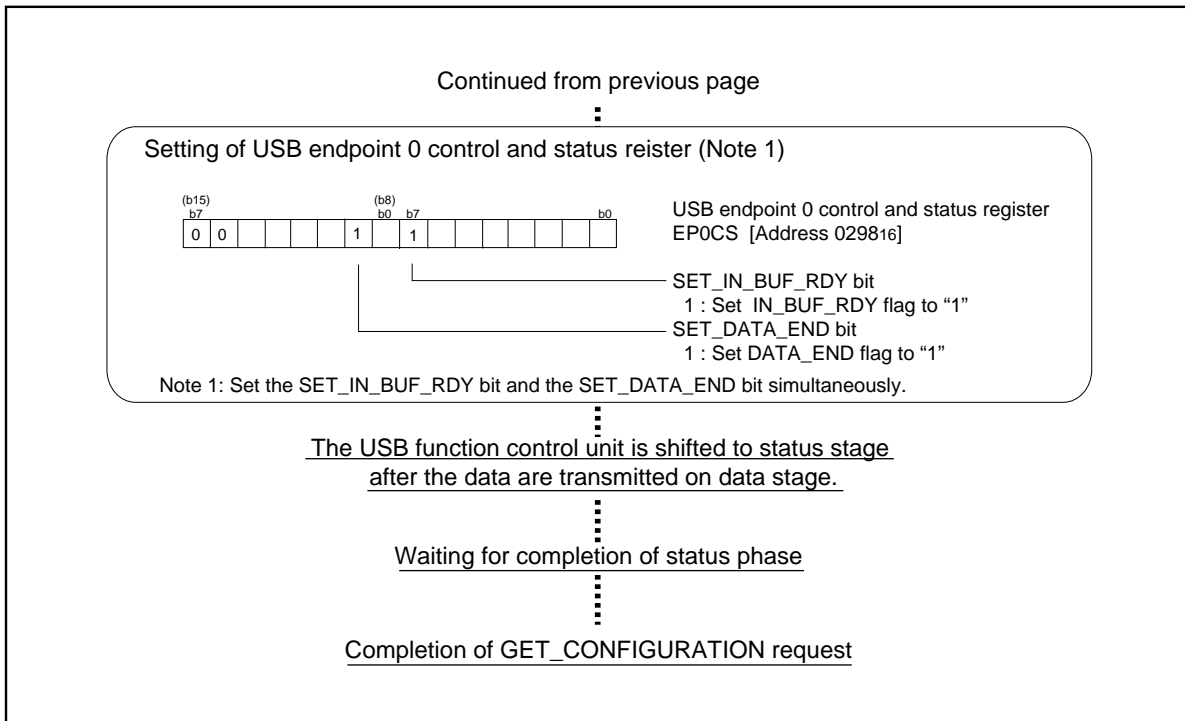


Figure 2.8.37. Processing routine (2) for getting device address when receiving SET\_ADDRESS request



**Figure 2.8.38. Device configuration notification processing routine (1) when receiving GET\_CONFIGURATION request**



**Figure 2.8.39. Device configuration notification processing routine (2) when receiving GET\_CONFIGURATION request**

### 2.8.6 USB Operation (Endpoints 1 to 4 Receive)

Endpoints 1 to 4 can apply to the isochronous transfer, bulk transfer and interrupt transfer.

The endpoints 1 to 4 respectively have their IN (transmit) FIFOs and OUT (receive) FIFOs.

For using the endpoints 1 to 4 OUT, enable each endpoint OUT FIFO by USB endpoint enable register (address 028E16). The size and the starting location (every 64 bytes) of each endpoint x(x=1 to 4) OUT FIFO can be set according to the user's system. The buffer size of OUT FIFO can be set to a maximum of 1024 bytes per 64 bytes for one endpoint. When the double buffer mode is enabled, the buffer which has twice as much as the set size is available for the OUT FIFO. The size and starting location of FIFO, the double buffer mode enable can be set by USB endpoint x OUT FIFO configuration register (EPxOFC).

When one buffer data is received from the host CPU, the data are written to the endpoint x OUT FIFO and the number of bytes of receive packet data are stored in USB endpoint x OUT write count register. When a data receive request from the host CPU occurs while data are already written and OUT FIFO cannot be received, NAK is automatically transmitted in bulk transfer/interrupt transfer, and an overrun occurs in isochronous transfer, not receiving the packet data.

The data receive from the host CPU is controlled based on the communication status of endpoints 1 to 4 OUT.

The default of endpoints 1 to 4 is bulk transfer. Each endpoint should be initialized in order to use other transfer modes.

The receive of endpoints 1 to 4 can select the following functions:

#### Continuous Receive Mode

This function is used for receiving data from the host PC at a higher speed. This mode can be set only for endpoints 1 to 4 OUT bulk transfer. With continuous transfer mode bit of the EPxOFC being set to "1", the continuous receive mode is enabled. The USB function control unit writes the receive data from the host PC in OUT FIFO sequentially by the maximum packet size that is set in USB endpoint x OUT MAXP register (EPxOMP). (When the last one packet is smaller than the size set in the EPxOMP, it is received as a short packet.)

When continuous receive mode is enabled, the buffer size has to be equal to an integral multiple of the EPxOMP. Further, the user's system has to be comprehended beforehand that the receive data from the host PC are equal to the buffer size or includes a short packet.

#### AUTO\_CLR Function

When receive data from OUT FIFO are read, both the OUT\_BUF\_STS0 and the OUT\_BUF\_STS1 flags are updated without CLR\_OUT\_BUF\_RDY bit being set to "1". The AUTO\_CLR function is enabled by setting AUTO\_CLR bit of the EPxOCS to "1". The AUTO\_CLR function is available both in the continuous receive mode and in the continuous receive mode disable of endpoints 1 to 4 OUT (Not available with endpoint 0).

**(1) Related Registers****● USB endpoint x(x=1 to 4) OUT control and status register****•OUT\_BUF\_STS1, OUT\_BUF\_STS0 flags**

These flags indicate OUT FIFO state.

At the time of reading the receive data from the host PC, read these flags to confirm the OUT FIFO state. When the OUT\_BUF\_STS1 and the OUT\_BUF\_STS0 flags are respectively set to "002", there are no data in OUT FIFO. When they are respectively set to "102", there are only one buffer data in double buffer. (Invalid for single buffer.) When they are respectively set to "112", there are one buffer data in single buffer while there are two buffer data in double buffer. When they are respectively set to "012", it is invalid.

These flags are updated when one of the following events occurs:

- One valid buffer data is successfully received from the host.
- One buffer data is successfully fetched from OUT FIFO.

CLR\_OUT\_BUF\_RDY bit is set to "1" after read of one receive data from OUT FIFO completes. (When the AUTO\_CLR function is enabled, these flags are updated without CLR\_OUT\_BUF\_RDY bit being set to "1".)

- The OUT FIFO buffer data are flushed. (When FLUSH bit is set to "1".)

**•OVER\_RUN flag**

This flag indicates occurrence of an overrun in isochronous transfer. The bit is valid only in isochronous transfer. When OUT FIFO is not empty and disables receiving at start of the OUT token from the host CPU, occurrence of an overrun is recognized, setting this bit to "1".

Clear this flag by writing "1" to CLR\_OVER\_RUN bit.

**•FORCE\_STALL flag**

This flag indicates occurrence of a packet size error.

When the data packet, which size exceeds USB endpoint x OUT MAXP register value, is transmitted from the host CPU, this flag becomes "1". While this bit is set to "1", the USB function control unit does not receive packet data. If it is in bulk transfer, also, STALL handshake is transmitted to the host CPU.

Clear this flag by writing "1" to CLR\_FORCE\_STALL bit.

**•DATA\_ERR flag**

This flag indicates occurrence of data error in isochronous transfer. The bit is valid only in isochronous transfer. If any bit stuffing error or CRC error is detected in the received packet, this flag becomes "1".

Clear this flag by writing "1" to CLR\_DATA\_ERR bit.

**•CLR\_OUT\_BUF\_RDY bit**

This bit controls OUT FIFO. Set this bit to "1" after one receive buffer data is read from OUT FIFO. Completion of one buffer data fetch is notified to the USB function control unit and, simultaneously, the OUT\_BUF\_STS0 and OUT\_BUF\_STS1 flags are updated.

When the AUTO\_CLR function is enabled, this bit does not need to be set up.

**•CLR\_OVER\_RUN bit**

The OVER\_RUN flag is cleared to "0" by setting "1" to this bit.

**•CLR\_FORCE\_STALL bit**

The FORCE\_STALL flag is cleared to "0" by setting "1" to this bit.

**•CLR\_DATA\_ERR bit**

The DATA\_ERR flag is cleared to "0" by setting "1" to this bit.



**•TOGGLE\_INIT bit**

This bit initializes data toggle sequence bit in bulk/interrupt transfer.

With this bit being set to "1", the PID of the next packet to be received from the host CPU becomes DATA0. When initialization of the data toggle sequence is requested from the host CPU at the time of configuration, etc., set TOGGLE\_INIT bit and initialize PID to DATA0 before starting the OUT endpoint communication.

At this time, the internal read/write counter of OUT FIFO is also initialized.

On completing PID initialization, this bit is automatically cleared to "0".

**•FLUSH bit**

This bit controls the OUT FIFO packet.

With this bit being set to "1", one buffer data received in OUT FIFO is flushed out from the OUT FIFO.

- When there is one buffer data in OUT FIFO, the OUT FIFO becomes empty.

At this time, the OUT\_BUF\_STS1 and OUT\_BUF\_STS0 flags are updated from "112"("102") to "002".

- When there are two buffer data in OUT FIFO, the older data is flushed out from the OUT FIFO.

At this time, the OUT\_BUF\_STS1 and OUT\_BUF\_STS0 flags are updated from "112" to "102" (This indicates that one more buffer data is left inside the OUT FIFO).

The receive data may be destroyed if this bit is set to "1" during USB transfer.

Read the OUT\_BUF\_STS1 and OUT\_BUF\_STS0 flags and confirm that there are data in the OUT FIFO, and then, set this bit to "1".

On completing one buffer data destruction, this bit is automatically cleared to "0".

**•ISO bit**

Set this bit to "1" in order to use an endpoint in isochronous transfer. Set this bit to "0" in order to use an endpoint in bulk/interrupt transfer.

**•SEND\_STALL bit**

This bit controls the STALL response to the host CPU in bulk transfer/interrupt transfer.

Set this bit to "1" when the OUT endpoint is in STALL state. While this bit is set to "1", the USB function control unit transmits the STALL handshake concerning all the OUT transactions to the host CPU. When the OUT endpoint has returned from STALL state, clear this bit by writing "0". The OUT endpoint communication is resumed.

**•AUTO\_CLR bit**

This bit controls setting of CLR\_OUT\_BUF\_RDY bit.

With this bit being set to "1", when one receive buffer data is read from OUT FIFO, the OUT\_BUF\_STS1 and OUT\_BUF\_STS0 flags are automatically updated without CLR\_OUT\_BUF\_RDY bit being set to "1". With this bit being set to "0", on completing one receive buffer data fetch from OUT FIFO, CLR\_OUT\_BUF\_RDY bit has to be set to "1" by software.

The configuration of USB endpoint x OUT control and status register is shown in Figure 2.8.40.

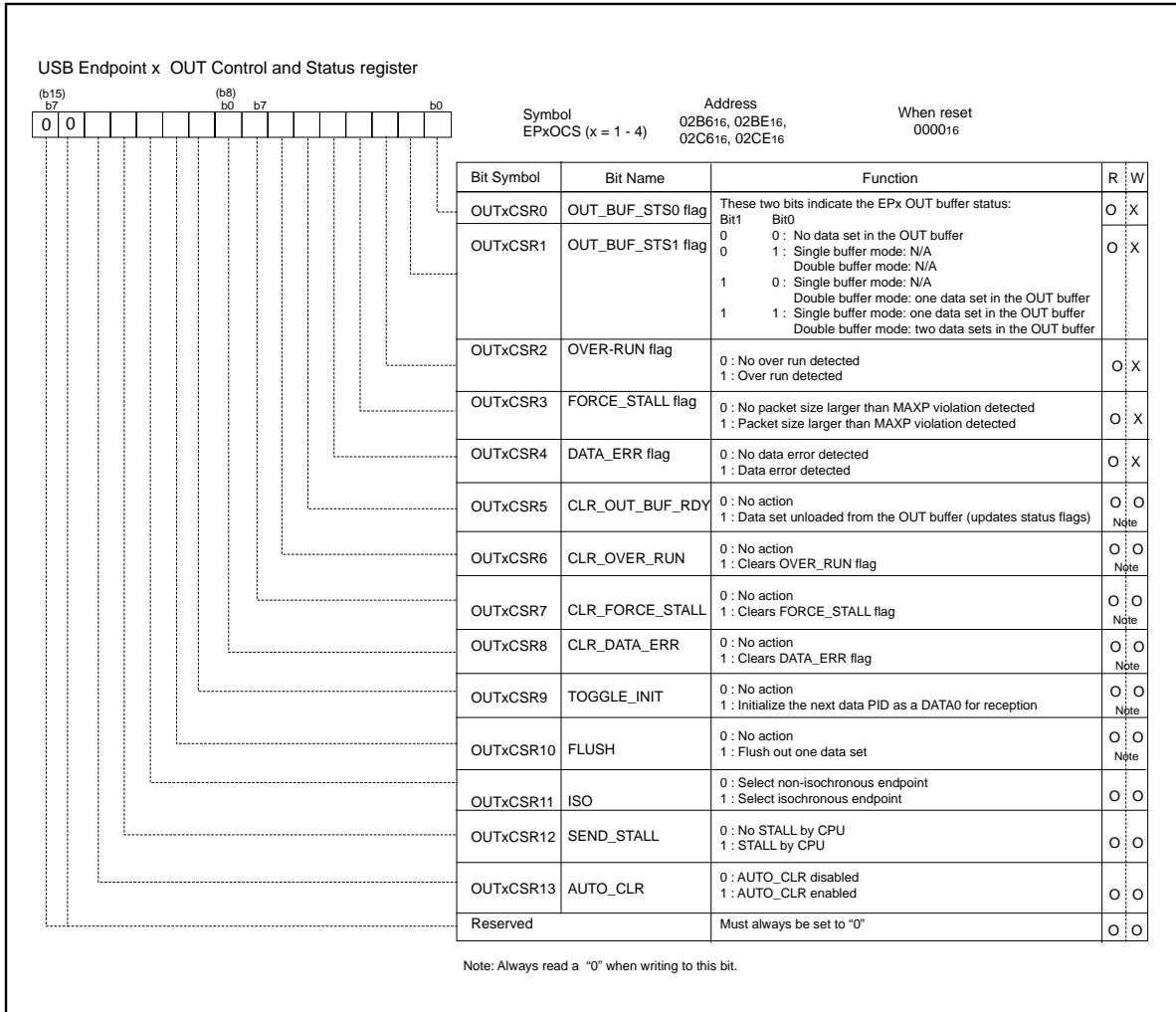


Figure 2.8.40. USB endpoint x(x=1 to 4) OUT control and status register

### ● USB endpoint x(x=1 to 4) OUT MAXP register

This register indicates endpoint x(x=1~4) OUT maximum packet size. The default value is 0 byte. When the endpoint is initialized due to any reason such as that the request for setting the endpoint (SET\_DESCRIPTOR, SET\_CONFIGURATION, SET\_INTERFACE, etc.) is received from the host CPU, change the endpoint x OUT maximum packet size value by writing in this register. Set a packet size value specified for every transfer type to be used.

The configuration of USB endpoint x(x=1 to 4) OUT MAXP register is shown in Figure 2.8.41.

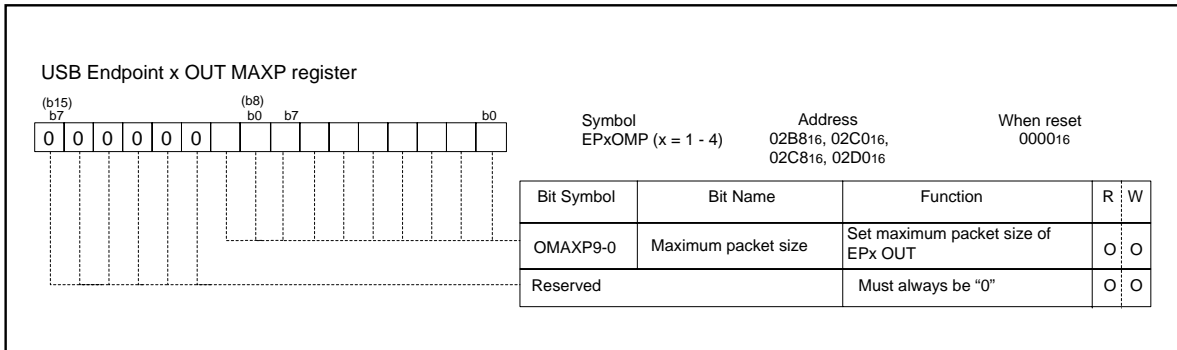


Figure 2.8.41. USB endpoint x(x=1 to 4) OUT MAXP register

### ● USB endpoint x(x=1 to 4) OUT write count register

This 11-bit register contains the number of bytes of one buffer data written in the endpoint x(x=1~4) OUT FIFO. This register is for read-only. When the USB function control unit completes the data packet receive from the host CPU, set the value of this register. When one buffer data receive completes, read this register and determine the byte count of the data to be read from OUT FIFO. This register value is not decremented even if the data are read from USB endpoint x OUT FIFO register. When this register is read while there are two buffer data in OUT FIFO in the double buffer mode, the number of bytes of the packet data received at first is already stored. When CLR\_OUT\_BUF\_RDY bit is set to "1" after one buffer data is read from OUT FIFO, this register value is updated to the number of bytes of buffer data subsequently received.

The configuration of USB endpoint x(x=1 to 4) OUT write count register is shown in Figure 2.8.42.

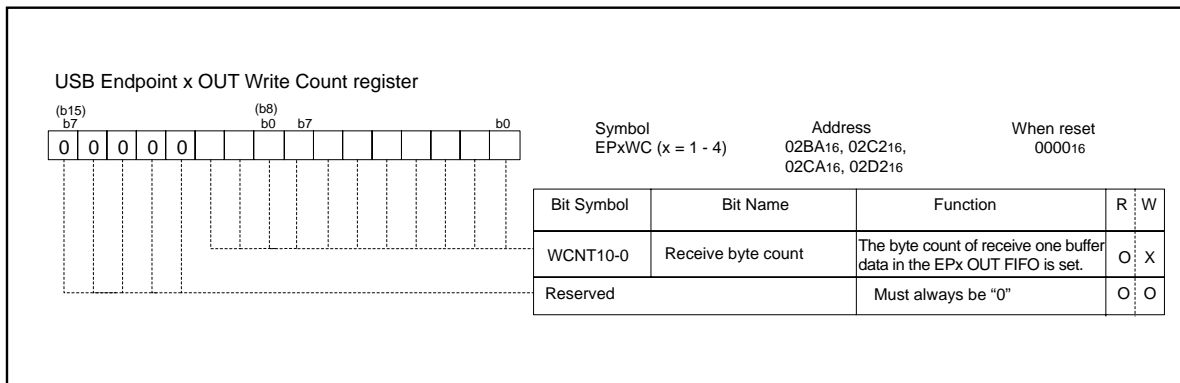


Figure 2.8.42. USB endpoint x(x=1 to 4) OUT write count register

### ● USB endpoint x(x=1 to 4) OUT FIFO configuration register

This register sets endpoint x(x=1~4) OUT FIFO.

#### ●BUF\_NUM

This bit sets the starting location of the endpoint x(x= 1~4) OUT FIFO per 64 bytes. For example, when OUT FIFO is allocated, starting at the 320th byte, the set value is "0001012".

#### ●BUF\_SIZ

This bit sets one buffer size of the endpoint x(x= 1~4) OUT FIFO per 64 bytes. For example, when 256 bytes is set, the set value is "01002".

#### ●DBL\_BUF

With this bit being set to "1", OUT FIFO of the corresponding endpoint is changed into double buffer mode. The byte count for a valid OUT FIFO becomes twice as much as the value specified by the BUF\_SIZ at the time of double buffer. Set carefully not to overlap with the FIFO start position of other endpoints.

#### ●CONTINUE

This bit enables continuous transfer mode.

Set this bit to "1" when continuous transfer is enabled. The bit is valid only in bulk transfer.

The USB function control unit writes the receive data from the host PC in OUT FIFO sequentially by one packet size (the maximum packet size set in the EPxOMP) and receives continuously until one buffer full or a short packet is received.

When continuous receive mode is enabled, the BUF\_SIZ has to be equal to an integral multiple of the EPxOMP. Further, the user's system has to be comprehended beforehand that the receive data from the host PC are equal to the buffer size or includes a short packet.

Pay attention to the following when setting the BUF\_NUM/BUF\_SIZ:

- Not exceed 3072 bytes in OUT FIFO starting location + OUT FIFO size.
- Not overlap Endpoint FIFOs each other.

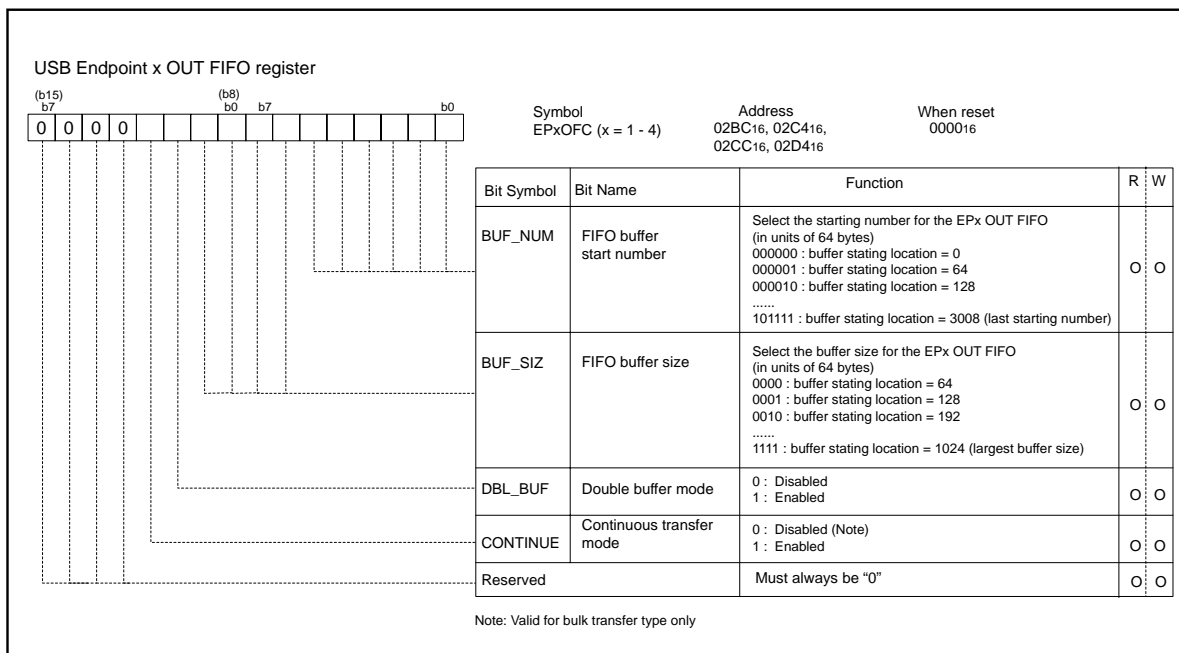


Figure 2.8.43. USB endpoint x(x=1 to 4) OUT FIFO configuration register

## (2) Bulk Transfer: Endpoints 1 to 4 Receive

### ● Setting of Transfer Type

When endpoints 1 to 4 OUT are used for bulk transfer, ISO bit of USB endpoint x(x=1 to 4) OUT control and status register is set to "0" for bulk transfer setting.

Also, for initialization of toggle sequence bit in bulk transfer, set TOGGLE\_INIT bit to "1" and initialize PID to DATA0.

Set by using USB endpoint x OUT FIFO configuration register in order to enable double buffer mode and continuous receive mode.

Set AUTO\_CLR bit of USB endpoint x OUT control and status register to "1" in order to use the AUTO\_CLR function.

### ● Receive Operation

When one packet data (Note 1) is received in OUT FIFO, the OUT\_BUF\_STS1 and the OUT\_BUF\_STS0 flags of the corresponding EPxOCS are automatically updated. In single buffer mode (when double buffer mode bit is "0"), these flags are updated from "002" to "112". In double buffer mode, they are updated as follows:

- When the first one packet data (Note 1) of the double buffer has been written to the OUT FIFO and the second packet data (Note 1) is ready to be written, the OUT\_BUF\_STS1 and OUT\_BUF\_STS0 flags are updated from "002" to "102".
- When two packet data (Note 1) have been written in OUT FIFO, the OUT\_BUF\_STS1 and OUT\_BUF\_STS0 flags are updated from "102" to "112".

Note 1: In continuous transfer enable, read the description by substituting the underlined part with "buffer data". The USB function control unit writes the receive data from the host PC in OUT FIFO sequentially by one packet size (the maximum packet size set in the EPxOMP), receives continuously until one buffer full or a short packet is received.

When the OUT token is received from the host CPU while SEND\_STALL bit is set to "1", STALL response is automatically returned.

When there is a packet space in OUT FIFO, on receiving the OUT token from the host CPU in the current data toggle sequence bit, the data are received and ACK response is returned. At this time, the OUT FIFO status is updated (updates the OUT\_BUF\_STS1 and OUT\_BUF\_STS0 flags) and data toggle sequence bit is toggled (DATA0 → DATA1 or DATA1 → DATA0). Further, the endpoint x OUT interrupt request occurs.

When there is a packet space in OUT FIFO, on receiving the OUT token from the host CPU in the toggle which is different from the current data toggle sequence bit, it is regarded that the ACK having responded for the packet previously received has dropped and, therefore, the host CPU has transmitted the same data. Only ACK response, therefore, is returned without receiving the data.

When the OUT token is received from the host CPU while there are already data in OUT FIFO and packet data cannot be received, NAK response is automatically returned.

When a packet, which size exceeds the maximum packet size, is transmitted from the host CPU, STALL response automatically is returned without receiving the data. At this time, the FORCE\_STALL flag is set to "1" and, when error interrupt has been enabled by USB function interrupt enable register, an error interrupt request occurs (INTST8 is set to "1").

When an error is detected in bulk OUT transfer, a response is not returned without ACK and NAK responses (Error checks such as CRC check and bit-justification, conforming to USB2.0 specification, are automatically performed. So the error does not have to be controlled by software).

### ● Fetch of Receive Data

On receiving one packet data (Note 2), the received packet data (Note 2) from OUT FIFO is read. Fetch one packet data (Note 2) in the following procedure:

- 1: Confirm that there are receive data in the OUT FIFO by the statuses of the OUT\_BUF\_STS1 and OUT\_BUF\_STS0 flags.
- 2: Determine the data byte count to be read from the OUT FIFO by reading USB endpoint x(x=1 to 4) OUT write count register .
- 3: Read the data byte count determined in the above 2: from the OUT FIFO.

Every time that 1(2)-byte data are read from the OUT FIFO, the internal write pointer is automatically decremented by one(two). (Content of the internal write pointer cannot be read.)

- 4: Set CLR\_OUT\_BUF\_RDY bit to "1" to complete one receive packet data fetch (Note 2).

At this time, the OUT FIFO status (OUT\_BUF\_STS1 and OUT\_BUF\_STS0 flags) are updated, enabling a receive of the next one packet data (Note 2).

- In Single Buffer Mode

The OUT\_BUF\_STS1 and OUT\_BUF\_STS0 flags are updated from "112" (the OUT FIFO full) to "002" (the OUT FIFO empty).

- In Double Buffer Mode

When there are one more packet data (Note 2) in the OUT FIFO, the OUT\_BUF\_STS1 and OUT\_BUF\_STS1 flags are updated from "112" (the OUT FIFO full) to "102" (one data set in the OUT FIFO). In this case, the second packet data (Note 2) can be continuously fetched.

When there are no data packet does in the OUT FIFO, the OUT\_BUF\_STS1 and OUT\_BUF\_STS1 flags are updated from "102" (one data set in the OUT FIFO) to "002" (the OUT FIFO empty).

When one packet data (Note 2) is read from the OUT FIFO while the AUTO\_CLR function is enabled (AUTO\_CLR bit is "1"), the OUT\_BUF\_STS1 and OUT\_BUF\_STS0 flags are automatically updated without CLR\_OUT\_BUF\_RDY bit being set to "1".

Note 2: In continuous transfer enable, read the description by substituting the underlined part with "buffer data". On receiving one buffer full (data equal to byte count set in the BUF\_SIZ) or a short packet, one buffer data receive is completed. Also, the BUF\_SIZ has to be equal to an integral multiple of the EPxOMP.

### (3) Isochronous Transfer: Endpoints 1 to 4 Receive

#### ● Setting of Transfer Type

When endpoints 1 to 4 OUT are used for isochronous transfer, ISO bit of USB endpoint x(x=1 to 4) OUT control and status register is set to "1" for isochronous transfer setting.

#### ● Receive Operation

When there is a packet space in OUT FIFO, on receiving the OUT token from the host CPU, the data are received. At this time, the OUT FIFO status is updated, the endpoint x OUT interrupt request occurs. When an error is detected in the received packet, simultaneously, the DATA\_ERR flag is set to "1". (Error checks such as CRC check, conforming to USB2.0 specification, are automatically performed.)

When the OUT token is received from the host CPU while there are already data in OUT FIFO and packet data cannot be received, an overrun error occurs. At this time, the OVER\_RUN flag is set to "1".

Further, when a packet, which size exceeds the maximum packet size, is transmitted from the host CPU, the FORCE\_STALL flag is set to "1" without receiving the data. While error interrupt has been enabled by USB function interrupt enable register, an error interrupt request occurs when any one of the OVER\_RUN flag, FORCE\_STALL flag or DATA\_ERR flag is set to "1" (INTST8 is set to "1").

#### ● Fetch of Receive Data

The fetch procedure of endpoint x OUT receive data in the isochronous transfer is same as the bulk transfer.

Refer to "● Fetch of Receive Data" of "(2) Bulk Transfer: Endpoints 1 to 4 Receive". (Although continuous transfer is valid for the bulk transfer only.)

**(4) Interrupt Transfer: Endpoints 1 to 4 Receive****● Setting of Transfer Type**

When endpoints 1 to 4 OUT are used for interrupt transfer, ISO bit of USB endpoint x(x=1 to 4) OUT control and status register is set to "0" for interrupt transfer setting.

Also, for initialization of toggle sequence bit in interrupt transfer, set TOGGLE\_INIT bit to "1" and initialize PID to DATA0.

**● Receive Operation**

The endpoint x OUT receive operation in the interrupt transfer is same as the bulk transfer.

Refer to "● Receive Operation" of "(2) Bulk Transfer: Endpoints 1 to 4 Receive".

**● Fetch of Receive Data**

The fetch procedure of endpoint x OUT receive data in the interrupt transfer is same as the bulk transfer.

Refer to "● Fetch of Receive Data" of "(2) Bulk Transfer: Endpoints 1 to 4 Receive". (Although continuous transfer is valid for the bulk transfer only.)

**(5) Precautions for Receive****● Read from OUT FIFO**

Be sure to confirm the OUT\_BUF\_STS1 and OUT\_BUF\_STS0 flags states when reading data from the OUT FIFO. Based on these flags states, judge whether there are receive data in the OUT FIFO. Be sure to read the byte count of data specified by USB endpoint x OUT write count register value before setting CLR\_OUT\_BUF\_RDYbit to "1" when reading data from the OUT FIFO. If the CLR\_OUT\_BUF\_RDY bit is set to "1" during fetching of data from the OUT FIFO, the setting can cause malfunction of the internal read pointer.

**Table 2.8.3. Status on Endpoint 1 to 4 OUT FIFOs**

OUT_BUF_STS1	OUT_BUF_STS0	Single buffer [Specify OUT FIFO size by the BUF_SIZ <sup>*1</sup> ]	Double buffer [OUT FIFO size = (The number of bytes specified by the BUF_SIZ <sup>*1</sup> )× 2]
0	0	No data Space equal to one buffer	No data Space equal to two buffer
0	1	Invalid	Invalid
1	0	Invalid	One data set in the OUT FIFO Space equal to one buffer
1	1	One data set in the OUT FIFO No space in the OUT FIFO	Two data set in the OUT FIFO No space in the OUT FIFO

\*1: Bits 6 to 9 of EPxOFC.

**● PID Initialization**

When TOGGLE\_INIT bit is set to "1", the read/write counter inside the FIFO is initialized. To initialize the PID, set TOGGLE\_INIT bit to "1" when the OUT FIFO is empty (the OUT\_BUF\_STS0 and OUT\_BUF\_STS1 flags are "002").





### 2.8.7 USB Operation (Endpoints 1 to 4 Transmit)

Endpoints 1 to 4 can apply to the isochronous transfer, bulk transfer and interrupt transfer.

The endpoints 1 to 4 respectively have their IN (transmit) FIFOs and OUT (receive) FIFOs.

For using the endpoints 1 to 4 IN, enable each endpoint IN FIFO by USB endpoint enable register (address 028E16). The size and the starting location (every 64 bytes) of each endpoint  $x(x=1\sim 4)$  IN FIFO can be set according to the user's system. The buffer size of IN FIFO can be set to a maximum of 1024 bytes per 64 bytes for one endpoint. When the double buffer mode is enabled, the buffer which has twice as much as the set size is available for the IN FIFO. The size and starting location of FIFO, the double buffer mode enable can be set by USB endpoint  $x$  IN FIFO configuration register(EPxIFC).

When packet data are transmitted to the host CPU, the data are written to the endpoint  $x$  IN FIFO. When a data transmit request from the host CPU occurs before the data are written to IN FIFO, NAK is automatically transmitted in bulk transfer and interrupt transfer, or an empty packet (with 0 data length) is automatically transmitted in isochronous transfer.

The data transmitted to the host CPU is controlled based on the communication status of endpoints 1 to 4 IN. The default of endpoints 1 to 4 is bulk transfer. Each endpoint should be initialized in order to use other transfer modes.

The transmit of endpoints 1 to 4 can select the following functions.

#### Continuous Transmit Mode

This function is used for transmitting data at a higher speed. This mode can be set only for endpoints 1 to 4 IN bulk transfer. With continuous transfer mode bit of the EPxIFC being set to "1", the continuous transmit mode is enabled.

In continuous transmit mode, the USB function control unit, by dividing the data in IN FIFO into one packet size (the maximum packet size set in the EPxIMP) units, transmits them one by one to the host PC (When the last one packet is smaller than the size set in the EPxIMP, it is transmitted as a short packet.) When continuous transmit mode is enabled, the IN FIFO size has to be equal to an integral multiple of the maximum packet size.

#### AUTO\_SET Function

With AUTO\_SET bit of EPxICS being set to "1", the AUTO\_SET function is enabled. When transmit data of the buffer size (specified in the BUF\_SIZ) is written to the IN FIFO in AUTO\_SET enable state, the IN\_BUF\_STS0 and IN\_BUF\_STS1 flags are updated without SET\_IN\_BUF\_RDY bit being set to "1". However, when a short packet (data whose size is smaller than the EPxIMP value in continuous transfer disable or than the BUF\_SIZ value in continuous transfer enable) has been written, the IN\_BUF\_STS1, IN\_BUF\_STS0 flags are not automatically updated. In these cases, the completion of data transmit ready is indicated by setting SET\_IN\_BUF\_RDY bit to "1". The AUTO\_SET function is useable both in continuous transmit mode and in continuous transmit mode disable of endpoints 1 to 4 IN (Not available with endpoint 0).

**(1) Related Registers****● USB ISO control register**

This register controls isochronous transfer of endpoints 1 to 4. This register setting is valid for all the isochronous transfer IN endpoints that are used simultaneously.

**•AUTO\_FLUSH bit**

This bit controls transmit packet data destruction in isochronous transfer. This bit can be used only when setting "1" to both ISO\_UPDATE bit and ISO bit. The bit is valid only for the IN endpoints 1 to 4 in isochronous transfer.

While ISO\_UPDATE bit = "1", AUTO\_FLUSH bit = "1", and ISO bit (INxCSR8) = "1", the USB function control unit, at the time of detecting SOF packet (on receiving from the host PC or on the artificial SOF operation), automatically flushes old data packet inside the IN FIFO if both the IN\_BUF\_STS1 and the IN\_BUF\_STS0 flags are "1" (IN FIFO full state). In isochronous transfer for double buffer, use the AUTO\_FLUSH function.

**•ISO\_UPDATE bit**

This bit controls the transmit timing of packet data in isochronous transfer. The bit is valid only for the IN endpoints 1 to 4 in isochronous transfer.

While ISO\_UPDATE bit = "0" and ISO bit = "1", the USB function control unit, at the time of receiving of an IN token from the host CPU, transmits one packet of the IN FIFO data if SET\_IN\_BUF\_RDY bit of the corresponding endpoint has been set beforehand to "1".

While ISO\_UPDATE bit = "1" and ISO bit = "1", a packet control internal signal is not output to the transmit control circuit inside the USB function control unit even if SET\_IN\_BUF\_RDY bit of the corresponding endpoint is set to "1" (even if a data packet whose size is equal to the maximum packet size is written to the IN FIFO when AUTO\_SET function is enabled). Setting "1" to SET\_IN\_BUF\_RDY bit, which is the packet control signal, is delayed until the next SOF is received so that transmission of the IN FIFO data packet is delayed. The USB function control unit, on detecting that there are transmit packet data at the time of status updating of the IN FIFO, operates artificially as having no transmit packet data until the next SOF packet is detected. On detecting SOF packet, one packet of data which has been set to the IN FIFO is transmitted to the IN token from the host CPU.

•Artificial SOF enable bit

This bit enables the artificial SOF function.

With this bit being set to “1”, when a SOF packet from the host PC has been destroyed due to any cause and no valid SOF packet has been received even after 1ms from the preceding start of the frame, the artificial SOF receive is operated. (And the USB SOF interrupt request, also, occurs.)

Therefore, even if the SOF packet is destroyed due to any cause, a new frame can be formed by this function without waiting for the next SOF packet. The artificial SOF receive function is operated once after the valid SOF packet is received twice.

•Artificial SOF status flag

This flag is the artificial SOF function status flag.

This flag is valid when the artificial SOF function has been enabled (Artificial SOF enable bit is “1”).

With this flag being set to “1”, an artificial SOF receive has occurred by the artificial SOF function.

This flag is cleared by setting “1” to CLR\_ART\_SOF bit .

•Artificial SOF status clear bit

Artificial SOF status flag is cleared to “0” by setting “1” to this bit.

The configuration of USB ISO control register is shown in Figure 2.8.45.

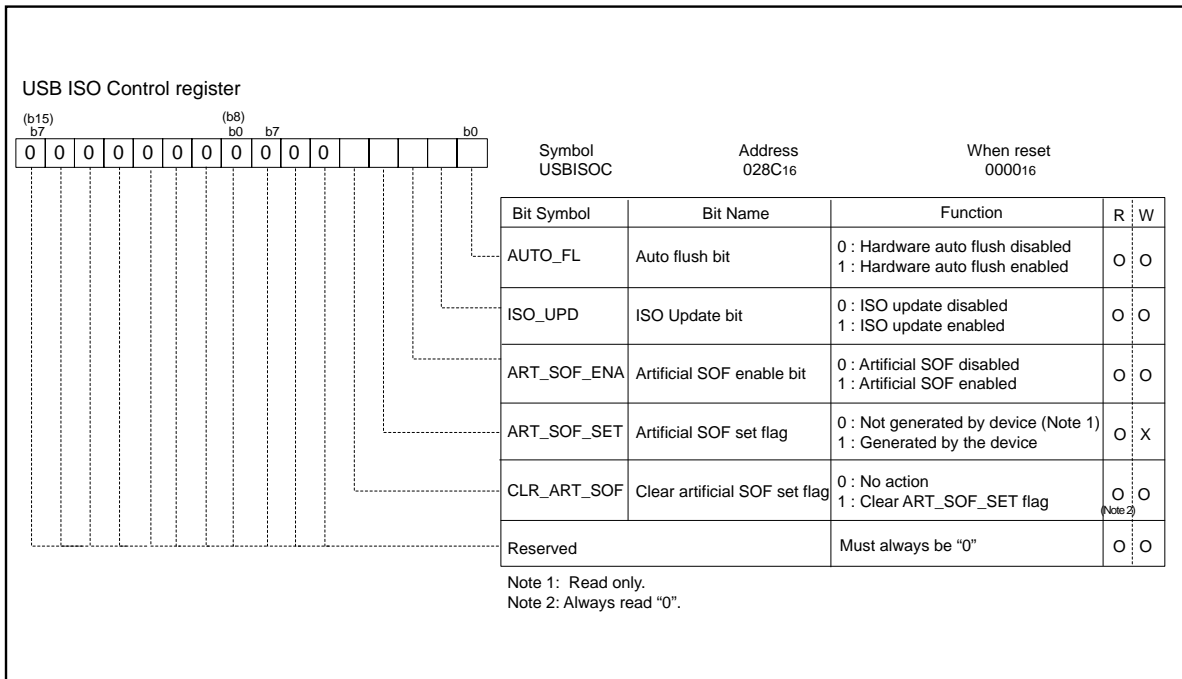


Figure 2.8.45. USB ISO control register

**● USB endpoint x(x=1 to 4) IN control and status register****•IN\_BUF\_STS1, IN\_BUF\_STS0 flags**

These flags indicate IN FIFO state.

At the time of writing the data to be transmitted to the host PC in IN FIFO, read these flags to confirm the IN FIFO state.

They are respectively set to "112" at the time of resetting. When the corresponding endpoint becomes enable state from disable state, the IN\_BUF\_STS1 and the IN\_BUF\_STS0 flags are respectively cleared to "002", automatically. When they are "002", there are no data in IN FIFO. When they are respectively set to "012", there are only one buffer data in double buffer (Invalid for single buffer). When they are respectively set to "112", there are no space in IN FIFO. (There are one buffer data in single buffer while there are two buffer data in double buffer.) When they are respectively set to "102", it is invalid.

These flags are updated when one of the following events occurs:

- One buffer data of IN FIFO is successfully transmitted to the host PC.
- One buffer data is successfully prepared in IN FIFO

SET\_IN\_BUF\_RDY bit is set to "1" when writing of one transmit data to the IN FIFO completes. Or, at the time of AUTO\_SET enable, when the data count equal to the EPxIMP (or the BUF\_SIZ in continuous transfer enable) has been written in the FIFO. However, when a short packet has been written at the time of AUTO\_SET enable, these flags are not automatically updated. In such cases, set SET\_IN\_BUF\_RDY bit to "1" by software.

- One buffer data is flushed.

**•UNDER\_RUN flag**

This flag indicates occurrence of an underrun in isochronous transfer. The bit is valid only in isochronous IN transfer. When there are no data packet in IN FIFO at start of the IN token from the host CPU, occurrence of an underrun is recognized and this flag is set to "1". This flag is cleared to "0" by setting "1" to CLR\_UNDER\_RUN bit.

**•SET\_IN\_BUF\_RDY bit**

This bit controls IN FIFO. When there is a space in IN FIFO (when IN\_BUF\_STS1, IN\_BUF\_STS0="002" in single buffer, or IN\_BUF\_STS1, IN\_BUF\_STS0="002" or "012" in double buffer enable), data can be written in the IN FIFO. One transmit data is prepared by setting this bit to "1" after writing the transmit data to the IN FIFO. (To transmit an empty packet, set this bit to "1" without writing any data.) When this bit is set to "1", the completion of one transmit data ready is notified to the USB function control unit and, simultaneously, the IN FIFO status (IN\_BUF\_STS1, IN\_BUF\_STS0 flags) is updated. In the AUTO\_SET enable, when a short packet (data whose size is smaller than the EPxIMP value in continuous transfer disable or the BUF\_SIZ value in continuous transfer enable) has been written, the IN\_BUF\_STS1 and IN\_BUF\_STS0 flags are not automatically updated. In this case, set this bit to "1".

**•CLR\_UNDER\_RUN bit**

The UNDER\_RUN flag is cleared to "0" by setting "1" to this bit.

**•TOGGLE\_INIT bit**

This bit initializes data toggle bit required in bulk and interrupt transfer.

When initialization of the data toggle sequence is requested from the host CPU at the time of configuration, etc., set this bit to "1" before starting the IN endpoint communication and initialize PID to DATA0. At this time, the internal read/write pointer of IN FIFO is also initialized.

**•FLUSH bit**

This bit controls the IN FIFO packet.

Read the IN\_BUF\_STS1 and IN\_BUF\_STS0 flags and confirm that there are data in the IN FIFO, and then, set this bit to "1". When the IN FIFO is flushed, the IN\_BUF\_STS1 and IN\_BUF\_STS0 flags are updated as follows:

- When there is one buffer data in IN FIFO, the IN FIFO becomes empty.

At this time, the IN\_BUF\_STS1 and IN\_BUF\_STS0 flags are updated to "002".

- When two buffer data exist in IN FIFO, the older data is flushed.

At this time, the IN\_BUF\_STS1 and IN\_BUF\_STS0 flags are updated to "012". (This indicates that one more buffer data is left inside the IN FIFO.)

The transmit data may be destroyed if this bit is set to "1" during USB transfer.

On completing one buffer data flush, this bit is automatically cleared to "0".

**•INTPT bit**

This bit controls transfer mode in interrupt transfer. Only when using the IN endpoint for the rate feedback interrupt transfer, set this bit to "1".

With this bit being set to "1", when an IN token is received from the host CPU, IN FIFO data are transmitted regardless of the IN\_BUF\_STS1 and IN\_BUF\_STS0 flag states or the data toggle.

Fix this bit at "0" for isochronous transfer, bulk transfer, and normal interrupt transfer.

**•ISO bit**

This bit controls isochronous transfer. With this bit being set to "1", the IN endpoint is used for isochronous transfer. Fix this bit at "0" for bulk transfer and interrupt transfer.

**•SEND\_STALL bit**

This bit controls the STALL response to the host CPU.

Set this bit to "1" when the IN endpoint is in STALL state. While this bit is set to "1", the USB function control unit transmits the STALL handshake concerning all the IN transactions to the host CPU.

When the IN endpoint has returned from STALL state, write "0" to clear this bit. The IN endpoint communication is resumed.

**-AUTO\_SET bit**

This bit controls setting of SET\_IN\_BUF\_RDY bit.

With this bit being set to “1”, when one data packet whose is equal to the maximum packet size (EPxIMP set value) has been written to IN FIFO in continuous transmit disable, or, when data equal to the buffer size (byte count set in the BUF\_SIZ of the EPxIFC) have been written to IN FIFO in continuous transmit enable, the IN\_BUF\_STS1 and IN\_BUF\_STS0 flags are updated without SET\_IN\_BUF\_RDY bit being set to “1”.

However, when a short packet (data whose size is smaller than the EPxIMP value in continuous transfer disable or the BUF\_SIZ value in continuous transfer enable) has been written, IN\_BUF\_STS1 and IN\_BUF\_STS0 flags are not automatically updated. In such cases, set SET\_IN\_BUF\_RDY bit to “1” by software.

With this bit being set to “0”, set SET\_IN\_BUF\_RDY bit to “1” by software after the transmit data are written to IN FIFO.

The configuration of USB endpoint x (x=1 to 4) IN control and status register is shown in Figure 2.8.46.

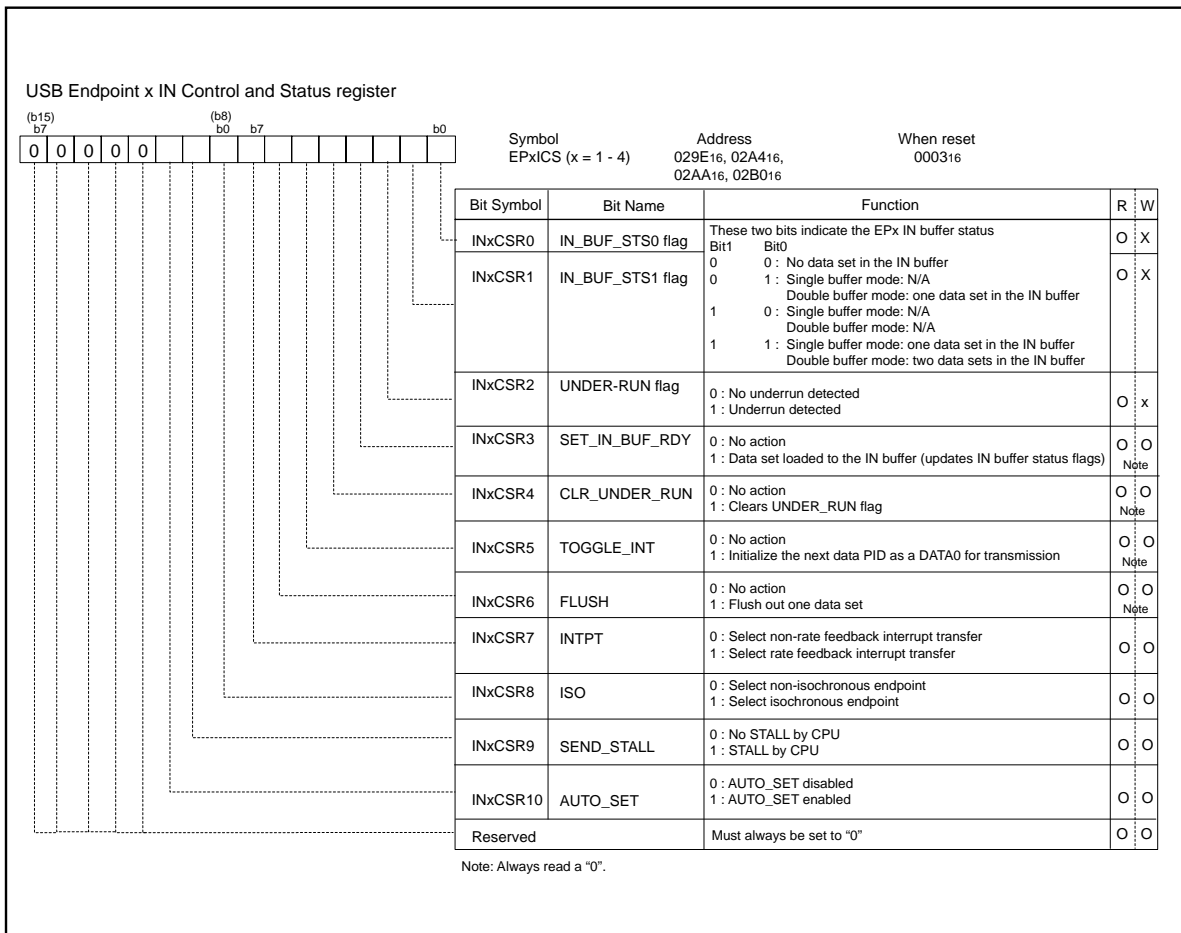


Figure 2.8.46. USB endpoint x(x=1 to 4) IN control and status register

● **USB endpoint x(x=1 to 4) IN MAXP register**

This register indicates endpoint x(x=1 to 4) IN maximum packet size. The default value is 0 byte.

When the endpoint is initialized due to any reason such as that the request for setting the endpoint (SET\_DESCRIPTOR, SET\_CONFIGURATION, SET\_INTERFACE, etc.) is received from the host CPU, change the endpoint x IN maximum packet size value by writing in this register. Set a packet size value specified for every transfer type to be used.

The configuration of USB endpoint x(x=1 to 4) IN MAXP register is shown in Figure 2.8.47.

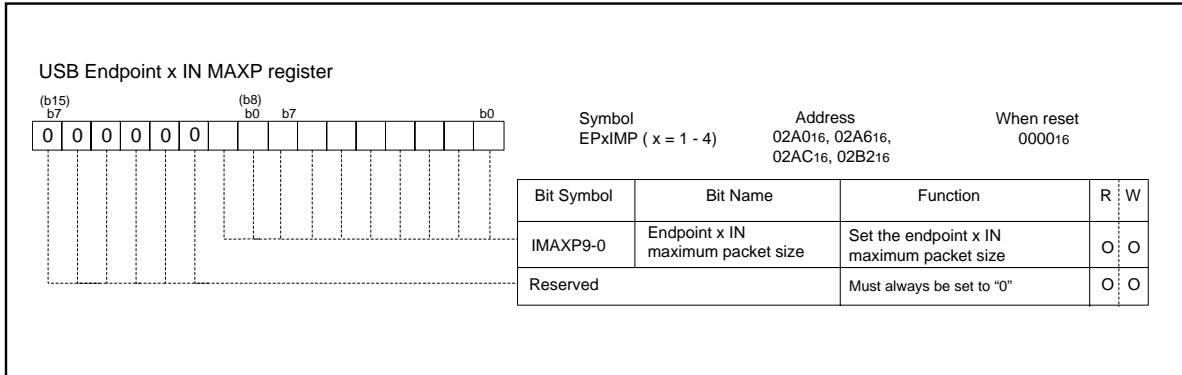


Figure 2.8.47. USB endpoint x(x=1 to 4) IN MAXP register



### ● USB endpoint x(x=1 to 4) IN FIFO configuration register

This register sets endpoint x(x=1 to 4) IN FIFO.

#### ●BUF\_NUM

This bit sets the starting location of the endpoint x(x= 1 to 4) IN FIFO per 64 bytes. For example, when IN FIFO is allocated, starting at the 320th byte, the set value is “0001012”.

#### ●BUF\_SIZ

This bit sets one buffer size of the endpoint x(x= 1 to 4) IN FIFO per 64 bytes. For example, when 256 bytes is set, the set value is “01002”.

#### ●DBL\_BUF

With this bit being set to “1”, IN FIFO of the corresponding endpoint is changed into double buffer mode. The byte count for a valid IN FIFO becomes twice as much as the value specified by the BUF\_SIZ at the time of double buffer. Set carefully not to overlap with the FIFO start position of other endpoints.

#### ●CONTINUE

Set this bit to “1” when continuous transmit is enabled. The bit is valid only in bulk transfer.

The USB function control unit, by dividing one buffer data equal to the byte count set in the BUF\_SIZ in the IN FIFO into one packet size (the maximum packet size set in the EPxIMP) units, transmits them one by one to the host PC. (When the last one packet is smaller than the size set in the EPxIMP, it is transmitted as a short packet.) When continuous transmit mode is enabled, the value set in the BUF\_SIZ has to be equal to an integral multiple of the maximum packet size.

Pay attention to the following when setting this register:

- Not exceed 3072 bytes in IN FIFO starting location + IN FIFO size.
- Not overlap endpoint FIFOs each other.

The configuration of USB endpoint x(x=1 to 4) IN FIFO configuration register is shown in Figure 2.8.48.

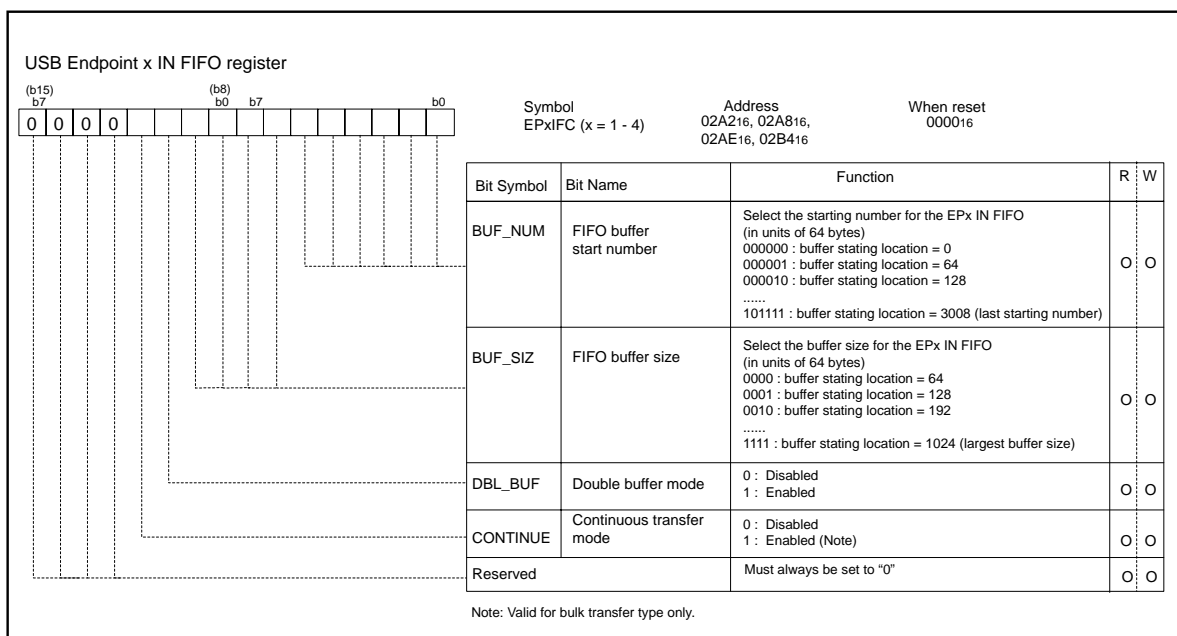


Figure 2.8.48. USB endpoint x(x=1 to 4) IN FIFO configuration register

## (2) Bulk Transfer: Endpoints 1 to 4 Transmit

### ● Setting of Transfer Type

When endpoints 1 to 4 IN are used for bulk transfer, both ISO bit and INTPT bit of USB endpoint x IN control and status register are set to "0" for bulk transfer setting.

Also, for initialization of toggle sequence bit in bulk transfer, set TOGGLE\_INIT bit to "1" and initialize PID to DATA0.

In order to enable double buffer mode and continuous transmit mode, use USB endpoint x IN FIFO configuration register for setting.

In order to use the AUTO\_SET function, set AUTO\_SET bit of USB endpoint x IN control and status register to "1".

### ● Transmit Data Preparation

The transmit data has to be beforehand prepared in IN FIFO in order to transmit data. Prepare one packet data (Note 1) to the IN FIFO in the following procedure:

1: Confirm that there is a packet space in the IN FIFO.

Read the IN\_BUF\_STS1 and IN\_BUF\_STS0 flags and confirm that they are either "002" (the IN FIFO empty) or "012" (writable of second data in double buffer).

2: Write one packet data (Note 1) to be transmitted to the IN FIFO.

Every time 1-byte data are written to the IN FIFO, the internal write pointer is automatically incremented by one. Contents of the internal write pointer cannot be read. For transmitting an empty packet (with 0 data length), do not write data to the IN FIFO.

3: Set SET\_IN\_BUF\_RDY bit to "1".

At this time, the IN FIFO status is updated as follows (the IN\_BUF\_STS1 and IN\_BUF\_STS0 flags are updated) and the transmit preparation is completed:

#### •In Single Buffer Mode

The IN\_BUF\_STS1 and IN\_BUF\_STS0 flags are updated from "002" (the IN FIFO empty) to "112" (the IN FIFO full).

#### •In Double Buffer Mode

When the first packet data (Note 1) of double buffer is written while there is a space in IN FIFO, the IN\_BUF\_STS0 and IN\_BUF\_STS1 flags are updated from "002" to "012", indicating that the second packet data (Note 1) is ready to be written to the IN FIFO. In this case, second packet data can be continuously prepared.

When there is only the first packet data (Note 1) in the IN FIFO and second packet data (Note 1) is written, the IN\_BUF\_STS0 and IN\_BUF\_STS1 flags are updated from "012" to "112", indicating that no more data can be written to the IN FIFO.

The USB function control unit transmits one packet data (Note 1) in the next IN token.

Note 1: In continuous transfer enable, read the description by substituting the underlined part with "buffer data". As for one buffer data, when SET\_IN\_BUF\_RDY bit is set to "1" after data less than the value set in the BUF\_SIZ are written to IN FIFO, one buffer data transmit is prepared. Also, the BUF\_SIZ has to be equal to an integral multiple of the EPxIMP.

While the AUTO\_SET is enabled (AUTO\_SET bit is "1"), when one data packet whose size is equal to the maximum packet size (EPxIMP set value) has been written to IN FIFO in continuous transmit disable, or, when data equal to the buffer size (byte count set in the BUF\_SIZ of the EPxIFC) has been written to IN FIFO in continuous transmit enable, the IN\_BUF\_STS1 and IN\_BUF\_STS0 flags are updated without SET\_IN\_BUF\_RDY bit being set to "1". However, when a short packet (data whose size is smaller than the EPxIMP value in continuous transfer disable or the BUF\_SIZ value in continuous transfer enable) has been written, the IN\_BUF\_STS1 and IN\_BUF\_STS0 flags are not automatically updated. In such cases, set the SET\_IN\_BUF\_RDY bit to "1" by software.

### ● Transmit Operation

On completing transmitting of one packet data (Note 2) to the host, the IN\_BUF\_STS0 and IN\_BUF\_STS1 flags are automatically updated. In single buffer mode (when double buffer mode bit is "0"), these flags are updated from "112" to "002". In double buffer mode, they are updated as follows:

- While there are two packet data (Note 2) in IN FIFO, the IN\_BUF\_STS0 and IN\_BUF\_STS1 flags are updated from "112" to "012" when one of the data is transmitted, indicating that one more transmit data is left inside the IN FIFO.
- When there is one packet data (Note 2) in IN FIFO, the IN\_BUF\_STS0 and IN\_BUF\_STS0 flags are updated from "012" to "002" when the data are transmitted, indicating that the IN FIFO becomes empty.

Note 2: In continuous transfer enable, read the description by substituting the underlined part with "buffer data". The USB function control unit transmits the transmit data in sequence by one packet size (the maximum packet size set in the EPxIMP). (When the last one packet is smaller than the size set in the EPxIMP, it is received as a short packet.)

When IN token is received from the host CPU while SEND\_STALL bit is set to "1", STALL response is automatically returned.

When IN token is received from the host CPU while there are no packet data in the IN FIFO, NAK response automatically is returned.

When IN token is received from the host CPU while there are packet data in the IN FIFO, data are transmitted by using the current data toggle sequence bit. On completing one packet data transmit (on receiving ACK from the host CPU), the IN FIFO status is updated (the IN\_BUF\_STS1 and IN\_BUF\_STS0 flags are updated) and data toggle sequence bit is toggled (DATA0 → DATA1, or DATA1 → DATA0). At this time, the endpoint x IN interrupt request occurs. When one packet data has been unsuccessfully transmitted (ACK not received from the host CPU), the data are re-transmitted in the next IN token (the same data are transmitted in the same toggle).

### (3) Isochronous Transfer: Endpoints 1 to 4 Transmit

#### ● Type of Transmit Transfer

When endpoints 1 to 4 IN are used for isochronous transfer, ISO bit and INTPT bit of USB endpoint x IN control and status register are respectively set to “1” and to “0”, for isochronous transfer setting.

#### ● Transmit Data Preparation

The endpoint x IN packet data preparation procedure in the isochronous transfer is same as the bulk transfer.

Refer to “● Transmit Data Preparation” of “(2) Bulk Transfer: Endpoints 1 to 4 Transmit” (Continuous transfer is valid for the bulk transfer only).

#### ● Transmit Operation

A packet whose PID is DATA0 is transmitted to the IN token from the host CPU.

When IN token is received from the host CPU while there are no data in IN FIFO, an empty packet (with 0 data length) is automatically transmitted, an underrun error occurs, and the UNDER\_RUN flag is set to “1”.

When the UNDER\_RUN flag is set to “1” while the error interrupt is enabled by USB function interrupt enable register, an error interrupt request occurs (INTST8 is set to “1”).

When IN token is received from the host CPU while there are packet data in IN FIFO, data are transmitted. At this time, the IN FIFO status is updated and the endpoint x IN interrupt request occurs. In isochronous transfer, timing in which data are transmitted to the IN token from the host CPU differs by setting of ISO\_UPDATE bit. When ISO\_UPDATE bit is set to “1”, setting “1” to SET\_IN\_BUF\_RDY bit, which is the packet control signal, is delayed until the next SOF packet is received so that transmission of the IN FIFO data packet is delayed. At this time, whether there are actual packet data or not, it is replied to the IN token that there are no packet data in the IN FIFO until the next SOF packet is detected.

Also, for flushing the IN FIFO in isochronous IN transfer by using software, the AUTO FLUSH function is used.

While ISO\_UPDATE bit = “1”, AUTO\_FLUSH bit = “1”, and ISO bit (INxCSR8) = “1”, the USB function control unit, at the time of detecting a SOF packet (from the host PC or on the artificial SOF), automatically flushes old data packet inside the IN FIFO if both the IN\_BUF\_STS1 and the IN\_BUF\_STS0 flags are “1” (IN FIFO full state). In isochronous transfer for double buffer, use the AUTO FLUSH function.

(4) Interrupt Transfer: Endpoints 1 to 4 Transmit

#### ● Setting of Transfer Type

Interrupt transfer setting has two kinds including the normal interrupt transfer and the rate feedback interrupt transfer.

When endpoints 1 to 4 IN are used for the normal interrupt transfer, ISO bit and INTPT bit of USB endpoint x IN control and status register are respectively to "0".

When the isochronous device has the rate feedback function and endpoints 1 to 4 IN are used for rate feedback interrupt transfer, not only ISO bit and INTPT bit of USB endpoint x IN control and status register are respectively set to "0" and to "1" but also double buffer mode enable bit of USB endpoint x IN FIFO configuration register is set to "0" so that single buffer is enabled.

Also, for initialization of toggle sequence bit in interrupt transfer, set TOGGLE\_INIT bit to "1" and initialize PID to DATA0.

#### ● Transmit Data Preparation

Normal Interrupt Transfer:

The endpoint x IN packet data preparation procedure in the normal interrupt transfer is same as the bulk transfer.

Refer to "● Transmit Data Preparation" of "(2) Bulk Transfer: Endpoints 1 to 4 Transmit" (Continuous transfer is valid for the bulk transfer only).

Rate Feedback Interrupt Transfer:

In real application, transmit data to the host CPU has to be always prepared. Prepare one transmit data to the IN FIFO in the following procedure:

For details of the following 1 and 2, refer to the single buffer mode parts in "● Transmit Data Preparation (2, 3)" of "(2) Bulk Transfer: Endpoints 1 to 4 Transmit" (Continuous transfer is valid for the bulk transfer only).

1: Write one packet data to be transmitted to the IN FIFO. At the time of writing the data, pay attention to the timing so that an IN token is not received from the host. Every time 1-byte data are written to the IN FIFO, the internal write pointer is automatically incremented by one. Contents of the internal write pointer cannot be read. For transmitting an empty packet (with 0 data length), do not write data to the IN FIFO.

2: Set SET\_IN\_BUF\_RDY bit to "1" after writing of the data to the IN FIFO are completed.

At this time, the IN FIFO state is updated and one packet transmit is prepared. The USB function control unit transmits this data to the IN token until next transmit data are updated.

#### ● Transmit Operation

Normal Interrupt Transfer:

The endpoint x IN transmit operation in the normal interrupt transfer is same as the bulk transfer. Refer to "● Transmit Operation" of "(2) Bulk Transfer: Endpoints 1 to 4 Transmit".

**Rate Feedback Interrupt Transfer:**

In real application, rate feedback interrupt transfer always has data to be transmitted to the host. Therefore, the device does not repond with NAK to the IN token from the host in this transfer. On receiving IN token from the host CPU, the IN FIFO data are always transmitted in the current data sequence bit regardless of the IN\_BUF\_STS0 and IN\_BUF\_STS1 values. Except this point, the transmit operation is the same as the normal interrupt transfer.

When IN token is received from the host CPU while SEND\_STALL bit being set to "1", STALL response is automatically returned. On receiving IN token from the host CPU, the IN FIFO data are transmitted in the current data sequence bit . On completing one data transmit (on receiving ACK from the host CPU), the IN FIFO status is updated, data toggle sequence bit is toggled (DATA0 →DATA1 or DATA1→DATA0), and the endpoint x IN interrupt request occurs. At this time, unlike the normal interrupt transfer, the IN FIFO data are not deleted, which is retained until the next packet data are updated. When one data transmit has not been unsuccessfully completed (an ACK not received from the host CPU), the data are re-transmitted in the next IN token (the same data are transmitted in the same toggle).

**(5) Precautions for Transmit****● Writing to IN FIFO**

Be sure to confirm that there is a space in the IN FIFO before writing data to the IN FIFO in preparation for packet data to the IN FIFO.

The IN FIFO state is indicated by the IN\_BUF\_STS1 and the IN\_BUF\_STS0 flags. Based on these flags states, determine the count of data packets set in the IN FIFO.

The IN FIFO status (IN\_BUF\_STS1 and IN\_BUF\_STS0 flags) is updated when transmit data are prepared in the IN FIFO (SET\_IN\_BUF\_RDY bit is set to "1"), when transmitting of one data to the host CPU is completed, or when data inside the IN FIFO have been flushed (AUTO\_FLUSH bit or FLUSH bit has functioned.)

**Table 2.8.4. Status on Endpoint 1 to 4 IN FIFOs**

IN_BUF_STS1	IN_BUF_STS0	Single buffer [Specify IN FIFO size by the BUF_SIZ <sup>*)</sup> ]	Double buffer [IN FIFO size = (The number of bytes specified by the BUF_SIZ <sup>*)</sup> X 2]
0	0	No data Space equal to one buffer	No data Space equal to two buffer
0	1	Invalid	Invalid
1	0	Invalid	One data set in the IN FIFO Space equal to one buffer
1	1	One data set in the IN FIFO No space in the IN FIFO	Two data set in the IN FIFO No space in the IN FIFO

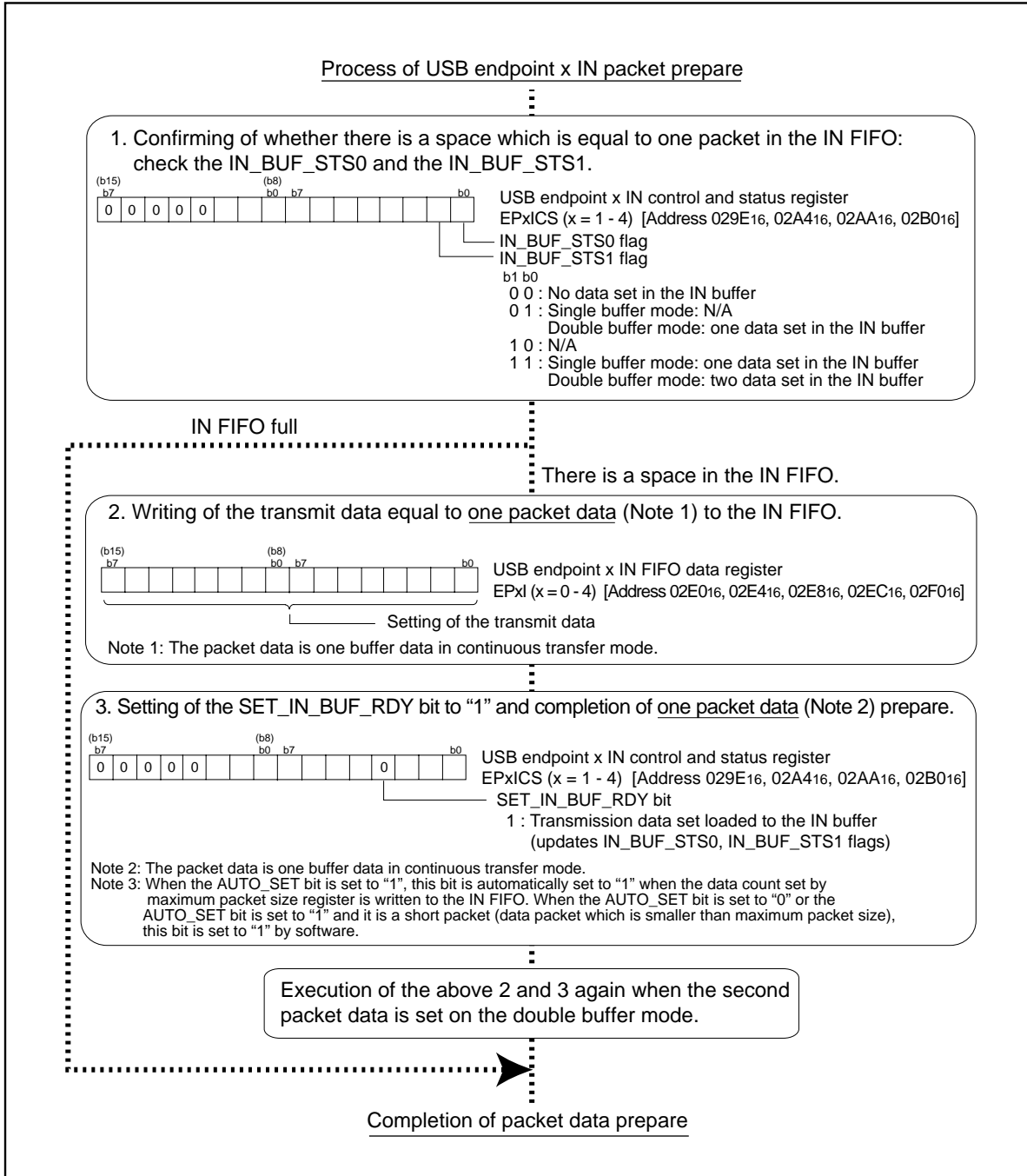
\*1: Bits 6 to 9 of EPxIFC.

**● PID Initialization**

When TOGGLE\_INIT bit is set to "1", the read/write counter inside the FIFO is initialized. To initialize the PID, set TOGGLE\_INIT bit to "1" in the IN FIFO is empty state (the IN\_BUF\_STS0 and IN\_BUF\_STS1 flags are "002").

**(6) USB Transmit (Endpoints 1 to 4 IN): Example**

The endpoints 1 to 4 IN transmit packet prepare routine (continuous transfer disable) is shown in Figure 2.8.49. In addition to packet prepare process, error process by the UNDER\_RUN flag is required in isochronous transfer.



**Figure 2.8.49. Endpoint 1 to 4 IN packet prepare routine**

### 2.8.8 USB Operation (Interface with DMAC Transfer)

The M30245 group can select a USB (USB0/USB1/USB2/USB3) as the DMA request factor. The USB0 corresponds to DMA0, USB1 to DMA1, USB2 to DMA2, and USB3 to DMA3. The DMA request factor origin of USB0/USB1/USB2/USB3 is also set by setting any one of endpoints 1 to 4 IN/OUT factors to USB DMAx(x=0 to 3) request register .

The DMA request factor of USB0/USB1/USB2/USB3 occurs, under particular conditions, not only on occurrence of an interrupt request of each endpoint but also on write/read to/from IN/OUT FIFO.

#### (1) Related Registers

##### ● USB DMAx(x=0 to 3) request register

This register sets the DMA request factor origin of USB0/USB1/USB2/USB3. When, under particular conditions, write/read to/from the FIFO of the endpoint selected by this register or an event such as the endpoint's interrupt request occurs, a DMA request occurs. This register can be set "1" only to 1 bit. When multiple bits are simultaneously set to "1", the setting becomes invalid. Other DMA related registers also need to be set before a valid value is set; for example, "000112" (USB0/USB1/USB2/USB3) is set to DMA request cause select bits (b4,b3,b2,b1,b0) of DMAx(x=0 to 3) request cause select register (addresses 03B816, 03BA16, 03B016, 03B216).

The configuration of USB DMAx(x=0 to 3) request register is shown in Figure 2.8.50.

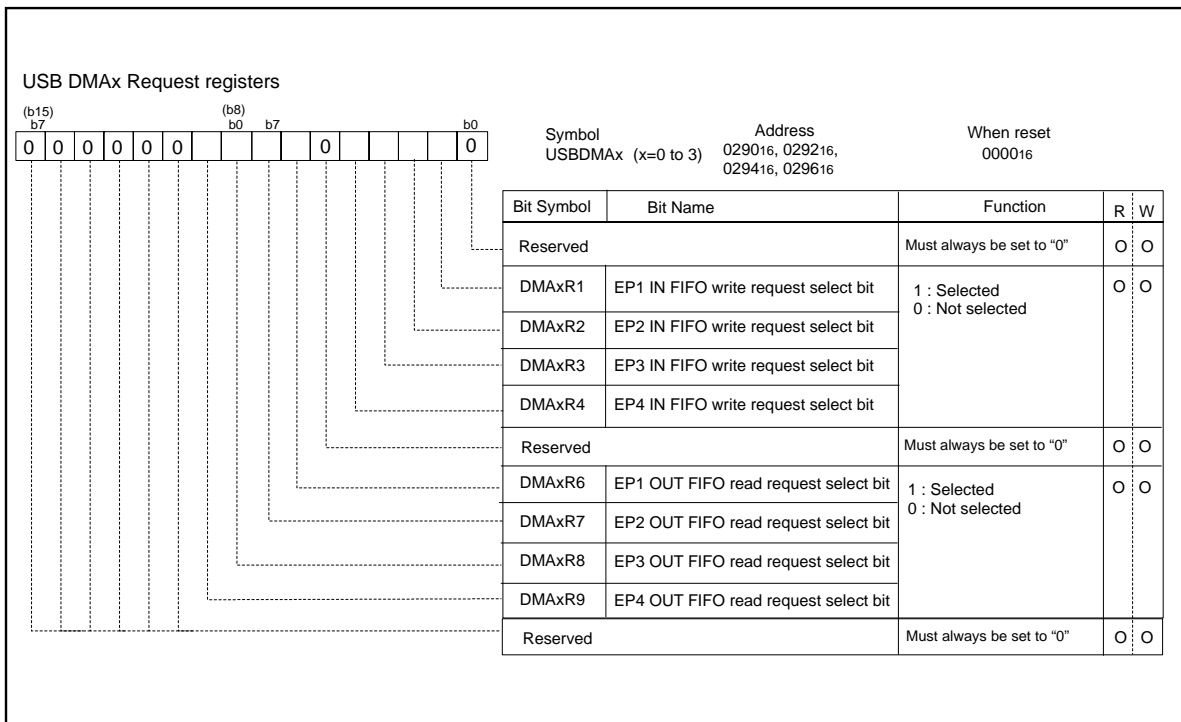


Figure 2.8.50. USB DMAx(x=0 to 3) request register



## (2) DMA Request by Endpoint x OUT

### ● DMA Request Factors

When endpoint 1 to 4 OUT FIFO write request select bit is set to the DMA request factor origin of USB0/USB1/USB2/USB3, the DMA request factor includes the following three kinds. On occurrence of an event when all the specified conditions have been satisfied for each factor, the DMA request of DMA0/DMA1/DMA2/DMA3 occurs.

#### •Factor 1

Conditions:

- DMA enable bit of DMA<sub>i</sub> control register is set to "1" (enable).
- Any one of endpoint x(x=1 to 4) OUT FIFO read request select bit in USB DMA<sub>x</sub>(x=0 to 3) request register is set to "1". The other bits are set to "0" (valid setting).

Event:

The OUT FIFO state of the endpoint x OUT which is set in USB DMA<sub>x</sub>(x=0 to 3) request register has been updated, and the OUT\_BUF\_STS1 and OUT\_BUF\_STS0 flags are set to "112" at the time of single buffer and "102" at the time of double buffer. (When data of one or more buffers are received in OUT FIFO. At this time, when one packet receive is completed, the endpoint x OUT interrupt request simultaneously occurs.)

#### •Factor 2

Conditions:

- DMA enable bit of DMA<sub>i</sub> control register is set to "1" (enable).
- The OUT\_BUF\_STS1 and OUT\_BUF\_STS0 flags of endpoint x OUT which is set in USB DMA<sub>x</sub>(x=0 to 3) request register are set to "102" or "112". (When data of one or more packets are received in OUT FIFO.)
- There is no selection of USB DMA<sub>x</sub>(x=0 to 3) request register ("0016").

Event:

Any one of endpoint x(x=1 to 4) OUT FIFO read request select bit in USB DMA<sub>x</sub>(x=0 to 3) request register is set to "1". The other bits are set to "0" (valid setting).

#### •Factor 3

Conditions:

- DMAenable bit of DMA<sub>i</sub> control register is set to "1" (enable).
- The OUT\_BUF\_STS1 and OUT\_BUF\_STS0 flags of endpoint x OUT which is set in USB DMA<sub>x</sub>(x=0 to 3) request register are set to "102" or "112". (When data of one or more packets are received in OUT FIFO.)
- Any one of endpoint x(x=1 to 4) OUT FIFO read request select bit in USB DMA<sub>x</sub>(x=0 to 3) request register is set to "1". The other bits are set to "0" (valid setting).

Event:

1-byte (1-word) data is read from the endpoint x OUT FIFO which is set in USB DMA<sub>x</sub>(x=0 to 3) request register .

### ● Reading of Endpoint x OUT FIFO in DMA Transfer

The DMA request factor of USB0/USB1/USB2/USB3 corresponds to read from the endpoints 1 to 4 OUT FIFO (Factor 3). Therefore, with endpoint x OUT FIFO being specified to the DMA source pointer and the transfer source address direction being fixed, when DMA transfer is executed by Factor 1 (Factor 2 or Factor 3), Factor 3 occurs. Therefore, when one buffer data (one packet data) is read from OUT FIFO by DMA transfer, it is possible that the 1st byte (1st word) data is DMA transferred by Factor 1 (Factor 2 or Factor 3) and the other data, starting from the 2nd byte (2nd word) up to the last byte (last word), are DMA transferred by Factor 3.

For details of DMA transfer, refer to "Chapter 2. DMAC".

### (3) DMA Request by Endpoint x IN

#### ● DMA Request Factor

When endpoint x(x=1 to 4) IN FIFO write request select bit is set to the DMA request factor origin of USB0/USB1/USB2/USB3, the DMA request factor includes the following three kinds. On occurrence of an event when all the specified conditions have been satisfied for each factor, the DMA request of DMA0/DMA1/DMA2/DMA3 occurs.

##### •Factor 1

Conditions:

- DMA enable bit of DMA<sub>i</sub> control register is set to "1" (enable).
- Any one endpoint x(x=1 to 4) IN FIFO write request select bit in USB DMA<sub>x</sub>(x=0 to 3) request register is set to "1". The other bits are set to "0" (valid setting).

Event:

The IN FIFO state of the endpoint x IN which is set in USB DMA<sub>x</sub>(x=0 to 3) request register has been updated, and the IN\_BUF\_STS1 and IN\_BUF\_STS0 flags are set to "002" at the time of single buffer and "012" at the time of double buffer. (When there are the space of one or more packets in the IN FIFO. At this time, when one packet transfer is completed, the endpoint x IN interrupt request simultaneously occurs.)

##### •Factor 2

Conditions:

- DMA enable bit of DMA<sub>i</sub> control register is set to "1" (enable).
- The IN\_BUF\_STS1 and IN\_BUF\_STS0 flags of endpoint x IN which is set in USB DMA<sub>x</sub>(x=0 to 3) request register are set to "002" or "012". (When there are the space of one or more packets in the IN FIFO.)
- There is no selection of USB DMA<sub>x</sub>(x=0 to 3) request register ("0016").

Event:

Any one endpoint x(x=1 to 4) IN FIFO write request select bit in USB DMA<sub>x</sub>(x=0 to 3) request register is set to "1". The other bits are set to "0" (valid setting).

##### •Factor 3

Conditions:

- DMA enable bit of DMA<sub>i</sub> control register is set to "1" (enable).
- The IN\_BUF\_STS1 and IN\_BUF\_STS0 flags of endpoint x IN which is set in USB DMA<sub>x</sub>(x=0 to 3) request register are set to "002" or "012". (When there are the space of one or more packets in the IN FIFO.)
- Any one of endpoint x(x=1 to 4) IN FIFO write request select bit in USB DMA<sub>x</sub>(x=0 to 3) request register is set to "1". The other bits are set to "0" (valid setting).

Event:

1-byte (1-word) data is written in the endpoint x IN FIFO which is set in USB DMA<sub>x</sub>(x=0 to 3) request register .

#### ● DMA Transfer to Endpoint x IN FIFO

The DMA request factor of USB0/USB1/USB2/USB3 corresponds to write in the endpoints 1~4 IN FIFO (Factor 3). Therefore, with endpoint x IN FIFO being specified to the DMA destination pointer and the transfer destination address direction being fixed, when DMA transfer is executed by Factor 1 (Factor 2 or Factor 3), Factor 3 occurs. Therefore, when one buffer data is written in IN FIFO by DMA transfer, it is possible that the 1st byte (1st word) data is DMA transferred by Factor 1 (Factor 2 or Factor 3) and the other data, starting from the 2nd byte (2nd word) up to the last byte (last word), are DMA transferred by Factor 3.

For details of DMA transfer, refer to "Chapter 2.10 DMAC".

## 2.8.9 Precautions for USB

### (1) USB Communication

"In applications requiring high-reliability, we recommend providing the system with protective measures such as USB function initialization by software or USB reset by the host to prevent USB communication from being terminated unexpectedly, for example due to external causes such as noise."

### (2) Peripheral Circuit

The peripheral circuit block diagram is shown in Figure 2.8.51, the passive part of LPF pin is shown in Figure 2.8.52 and the connection diagram of decoupling capacitor is shown in Figure 2.8.53.

- USB2.0 specification specifies the driver impedance 28~44Ω (See 7.1.1.1 "Full-speed (12Mb/s) Driver Characteristics"). Connect a serial resistor (recommended value: 27~33Ω) to the USB D+ pin and the USB D- pin to satisfy this specification. Also connect, if required, the capacitors between the USB D+ pin/USB D- pin and the Vss pin. These capacitors control ringing or adjust the times of rising/falling and the crossover point of D+/D-. As the numerical values and the configuration of the peripheral components need to be adjusted according to differences in characteristic impedance and layout of the mount printed circuit board. Therefore, fully evaluate on the system in use and observe waveforms before adjusting the connection or disconnection and the values of the resistance and the capacitor.
- When the USB Attach/Detach function is not used, connect the UVcc pin and the USB D+ pin via a 1.5kΩ resistance (D+ line pull-up timing depends on the UVcc pin).  
When the USB Attach/Detach function is used, connect the P90/ATTACH pin and the USB D+ pin via a 1.5kΩ resistance. Irrespective of use of the USB Attach/Detach function, connect the UVcc pin to the power supply. In addition, the time required for the host PC to recognize the USB Attach/Detach varies depending on the whole system state such as substrate resistance components, capacitance components, USB cable capacitance, and substrate characteristics and processing speed of the host. Fully evaluate on the system subject to actual use.
- Connect all the passive parts of the LPF pin as close to the LPF pin as possible.
- Connect an insulating connector (ferrite beads) between the AVss pin and the digital Vss, between the AVcc pin and the digital Vcc. Also at this time, when an A/D converter is used, connect the Vref pin to the AVcc pin.

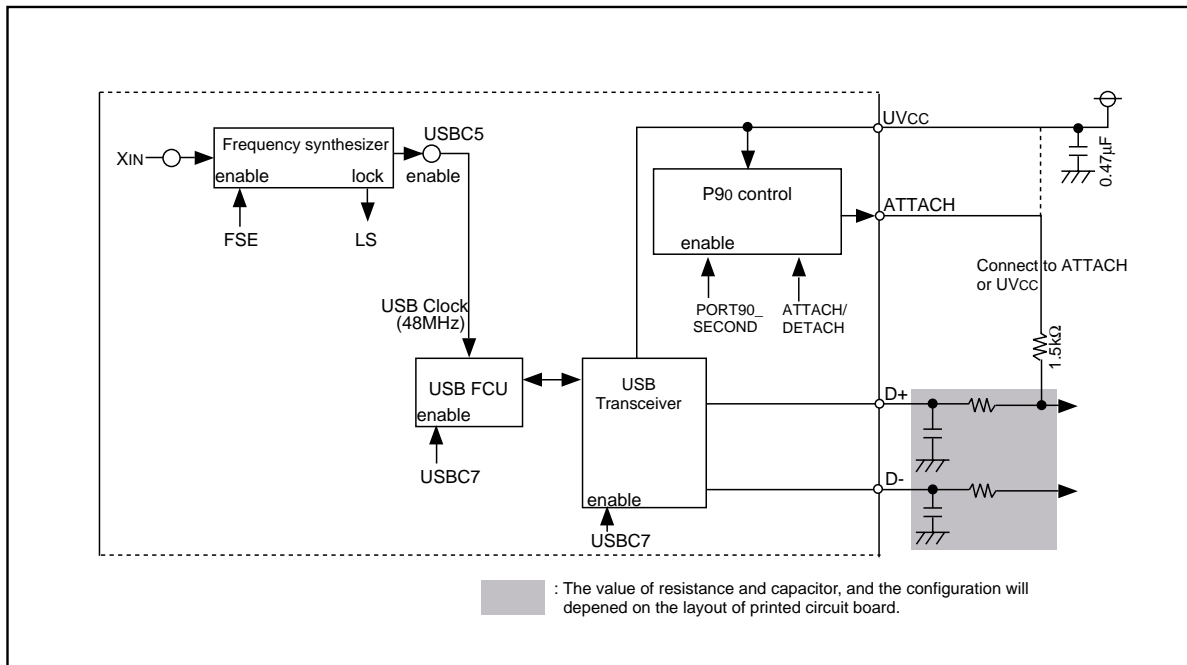


Figure 2.8.51. Peripheral circuit block diagram

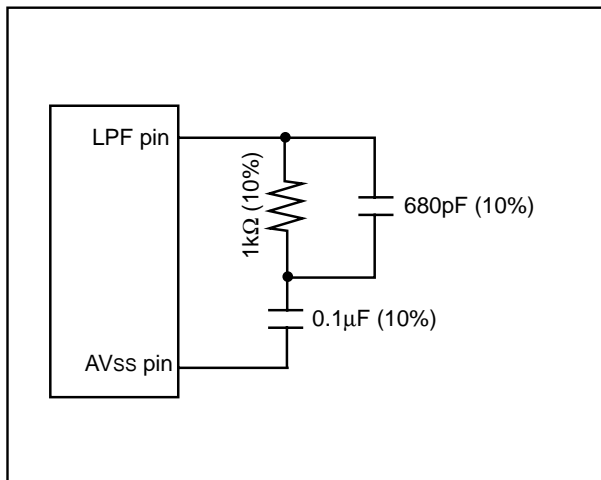


Figure 2.8.52. Passive part of LPF pin

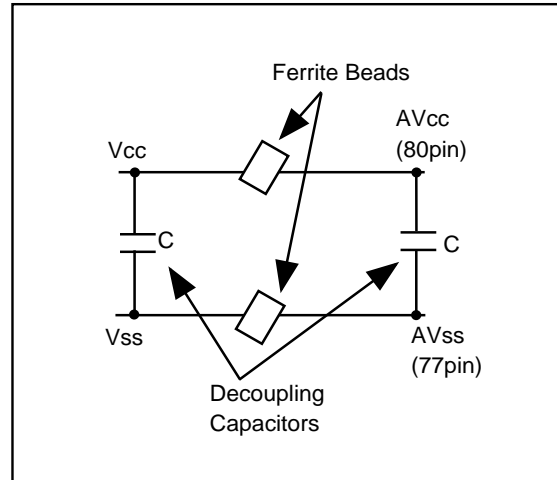


Figure 2.8.53. Decoupling capacitor

**(3) Register, Bit**

- When the USB reset interrupt request occurs, all the USB internal registers become reset state. To resume communication, each endpoint needs to be initialized.
- All the USB related registers (16-bit registers) except USB endpoint x(x=0 to 4) IN FIFO data register (EPxI), USB endpoint x(x=0 to 4) OUT FIFO data register (EPxO), USB control register (USBC), and USB attach/detach register (USBAD) are available for word access and byte access. The EPxI and the EPxO are only available for word access or byte access to the lower bytes. The USBC and the USBAD of 8-bit registers are only available for byte access. After software reset, contents of all the USB related registers are retained.
- While the USB clock is held disabled in suspend mode, writing in the USB internal registers (other than USBC, USBAD, and frequency synthesizer-related registers) is disabled.

**(4) Packet Data Destruction**

- When FLUSH bit of endpoint x OUT control and status register (EPxOCS) is set to "1" during USB transfer, the receive data may be destroyed. Be sure to set FLUSH bit of the EPxOCS to "1" only when there are data in OUT FIFO (OUT\_BUF\_STS1 and OUT\_BUF\_STS0 are set to "102" or "112").
- When FLUSH bit of USB endpoint x IN control and status register (EPxICS) is set to "1" during USB transfer, the transmit data may be destroyed. Be sure to read the IN\_BUF\_STS1 and the IN\_BUF\_STS0 flags and to confirm that there are data in IN FIFO before setting FLUSH bit of the EPxICS to "1".

In isochronous transfer, use AUTO\_FLUSH bit (bit 0 of address 028C16).

## 2.9 A/D Converter

### 2.9.1 Overview

The A/D converter used in the M30245 group operates on a successive conversion basis. The following is an overview of the A/D converter.

#### (1) Mode

The A/D converter operates in one of five modes:

##### (a) One-shot mode

Carries out A/D conversion on input level of one specified pin only once.

##### (b) Repetition mode

Repeatedly carries out A/D conversion on input level of one specified pin.

##### (c) One-shot sweep mode

Carries out A/D conversion on input level of two or more specified pins only once.

##### (d) Repeated sweep mode 0

Repeatedly carries out A/D conversion on input level of two or more specified pins.

##### (e) Repeated sweep mode 1

Repeatedly carries out A/D conversion on input level of two or more specified pins. This mode is different from the repeated sweep mode 0 in that weights can be assigned to specifying pins control the number of conversion times.

#### (2) Operation clock

The operation clock can be selected from the following:  $f_{AD}$ , divide-by-2  $f_{AD}$ , divide-by-3  $f_{AD}$ , and divide-by-4  $f_{AD}$ . The  $f_{AD}$  frequency is equal to that of the CPU's main clock ( $f_{AD}=f(X_{IN})$ ). Set the operation clock to 10 MHz or lower. When  $f(X_{IN})$  is over 10 MHz, the  $f_{AD}$  frequency must be under 10 MHz by dividing.

#### (3) Conversion time

Number of conversion for A/D converter varies depending on resolution as given. Table 2.9.1 shows relation between the A/D converter operation clock and conversion time.

Sample & Hold function selected:

33  $\phi_{AD}$  cycles for 10-bit resolution, or 28  $\phi_{AD}$  cycles for 8-bit resolution

No Sample & Hold function:

59  $\phi_{AD}$  cycles for 10-bit resolution, or 49  $\phi_{AD}$  cycles for 8-bit resolution

**Table 2.9.1. Conversion time every operation clock**

Frequency select bit 1		1	0	0	1
Frequency select bit 0		0	0	1	1
A/D converter's operation clock		$\phi_{AD} = f_{AD}/3$	$\phi_{AD} = f_{AD}/4$	$\phi_{AD} = f_{AD}/2$	$\phi_{AD} = f_{AD}$
Min. conversion cycles (Note 1)	8-bit mode	28 $\times \phi_{AD}$			
	10-bit mode	33 $\times \phi_{AD}$			
Min. conversion time (Note 2)	8-bit mode	8.4 $\mu$ s	11.2 $\mu$ s	5.6 $\mu$ s	2.8 $\mu$ s
	10-bit mode	9.9 $\mu$ s	13.2 $\mu$ s	6.6 $\mu$ s	3.3 $\mu$ s

Note 1: The number of conversion cycles per one analog input pin.

Note 2: The conversion time per one analog input pin (when  $f_{AD} = f(X_{IN}) = 10$  MHz)

**(4) Functions selection****(a) Sample & Hold function**

Sample & Hold function samples input voltage when A/D conversion starts and carries out A/D conversion on the voltage sampled. When A/D conversion starts, input voltage is sampled for 3 cycles of the operation clock. When the Sample & Hold function is selected, set the operation clock for A/D conversion to 1 MHz or higher.

**(b) 8-bit A/D to 10-bit A/D switching function**

Either 8-bit resolution or 10-bit resolution can be selected. When 8-bit resolution is selected, the 8 higher-order bits of the 10-bit A/D are subjected to A/D conversion. The equations for 10-bit resolution and 8-bit resolution are given below:

$$\text{10-bit resolution} \quad (V_{\text{ref}} \times n / 2^{10}) - (V_{\text{ref}} \times 0.5 / 2^{10}) \quad (n = 1 \text{ to } 1023), 0 (n = 0)$$

$$\text{8-bit resolution} \quad (V_{\text{ref}} \times n / 2^8) - (V_{\text{ref}} \times 0.5 / 2^{10}) \quad (n = 1 \text{ to } 255), 0 (n = 0)$$

**(c) A/D conversion by external trigger**

The user can select software or an external pin input to start A/D conversion.

**(d) Connecting or cutting Vref**

Cutting Vref allows decrease of the current flowing into the A/D converter. To decrease the microcomputer's power consumption, cut Vref. To carry out A/D conversion, start A/D conversion 1  $\mu$ s or longer after connecting Vref.

**(5) Input to A/D converter and the relation of direction register**

To use the A/D converter, set the direction register of the relevant port to input.

**(6) Pins related to A/D converter**

(a) AN0 pin through AN7 pin	Input pins of the A/D converter
(b) AVcc pin	Power source pin of the analog section
(c) VREF pin	Input pin of reference voltage
(d) AVss pin	GND pin of the analog section
(e) ADTRG pin	Trigger input pin of the A/D converter

**(7) A/D converter related registers**

Figure 2.9.1 shows the memory map of A/D converter-related registers, and Figures 2.9.2 and 2.9.3 show A/D converter-related registers.

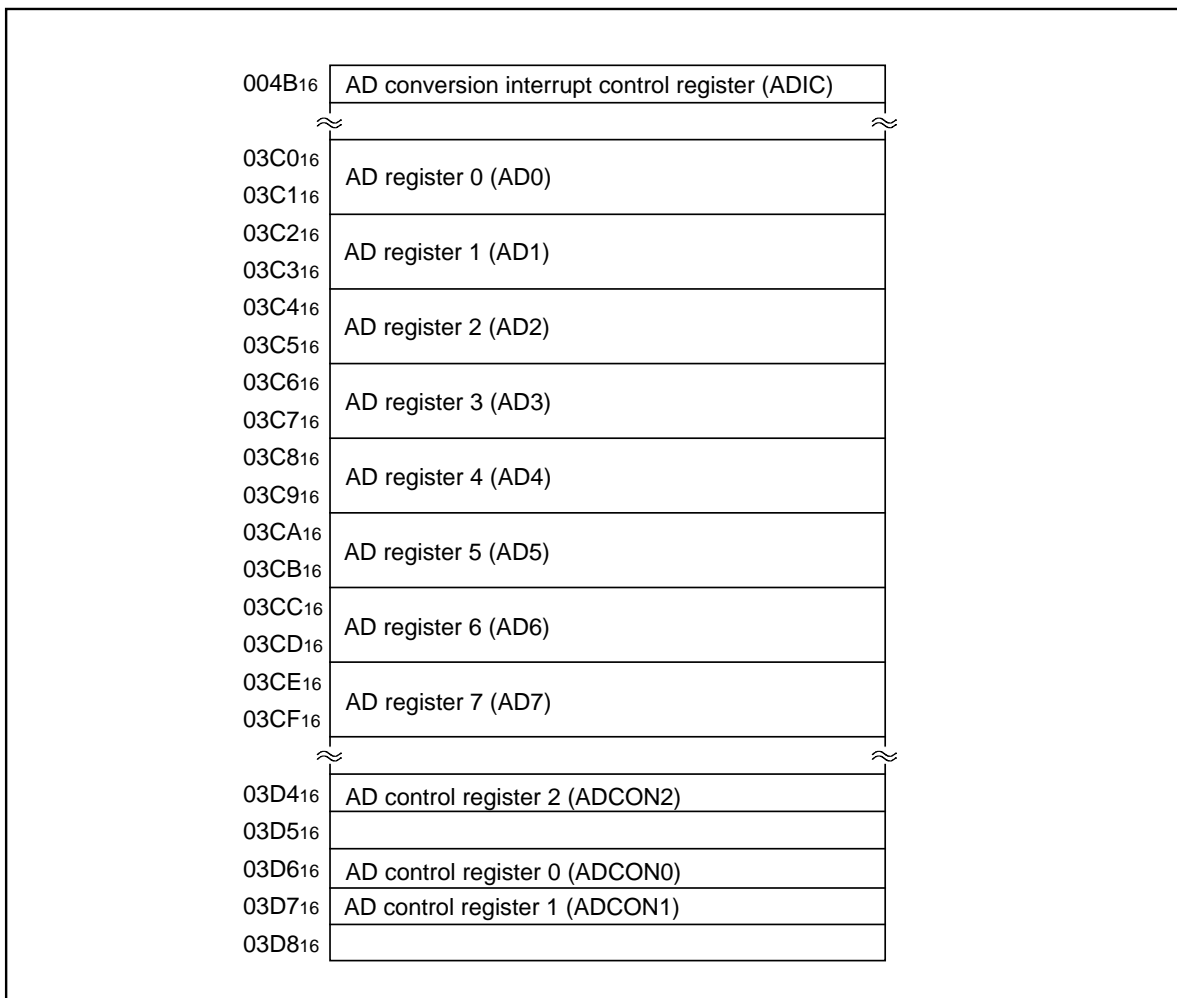


Figure 2.9.1. Memory map of A/D converter-related registers



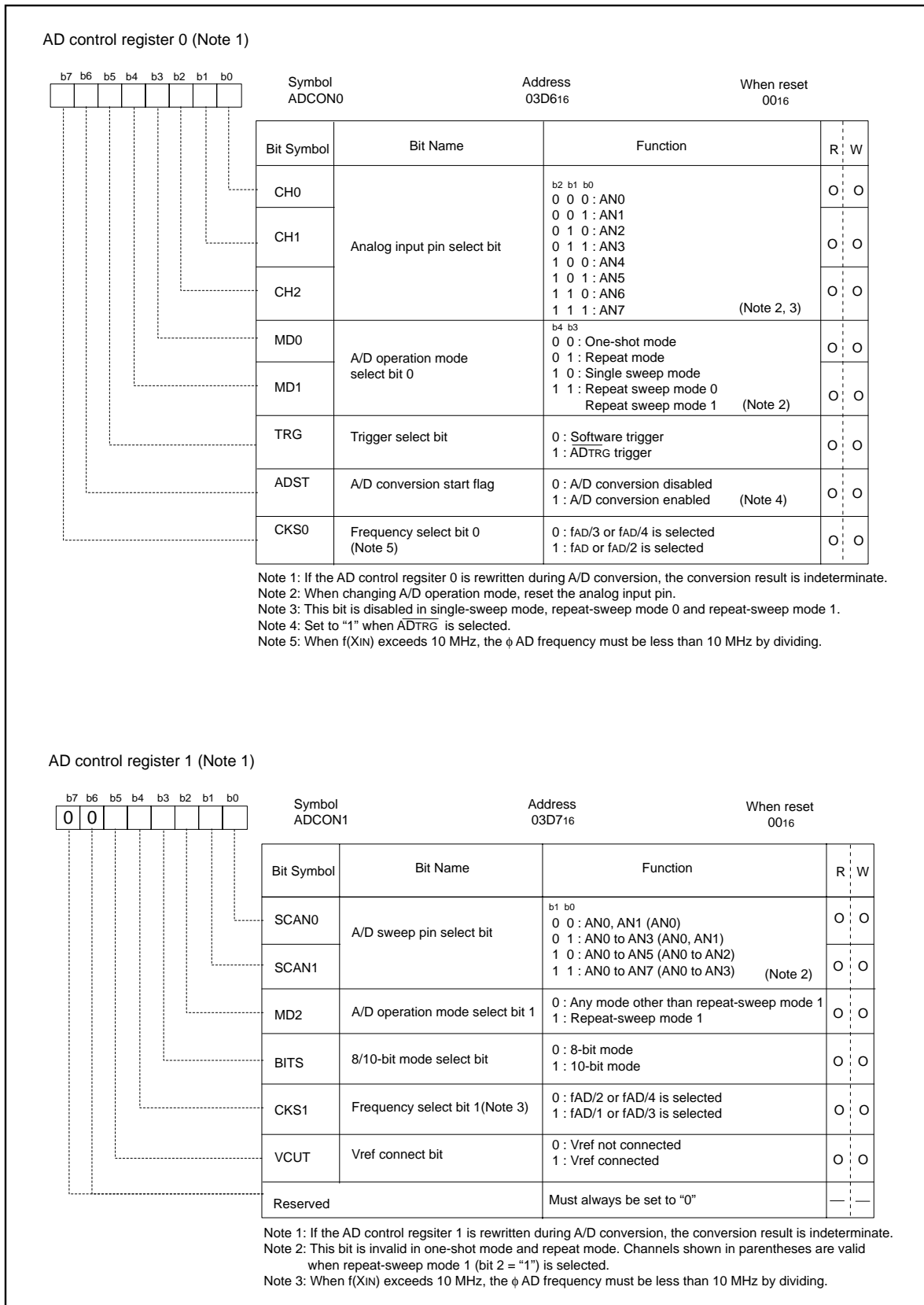


Figure 2.9.2. AD converter-related registers (1)

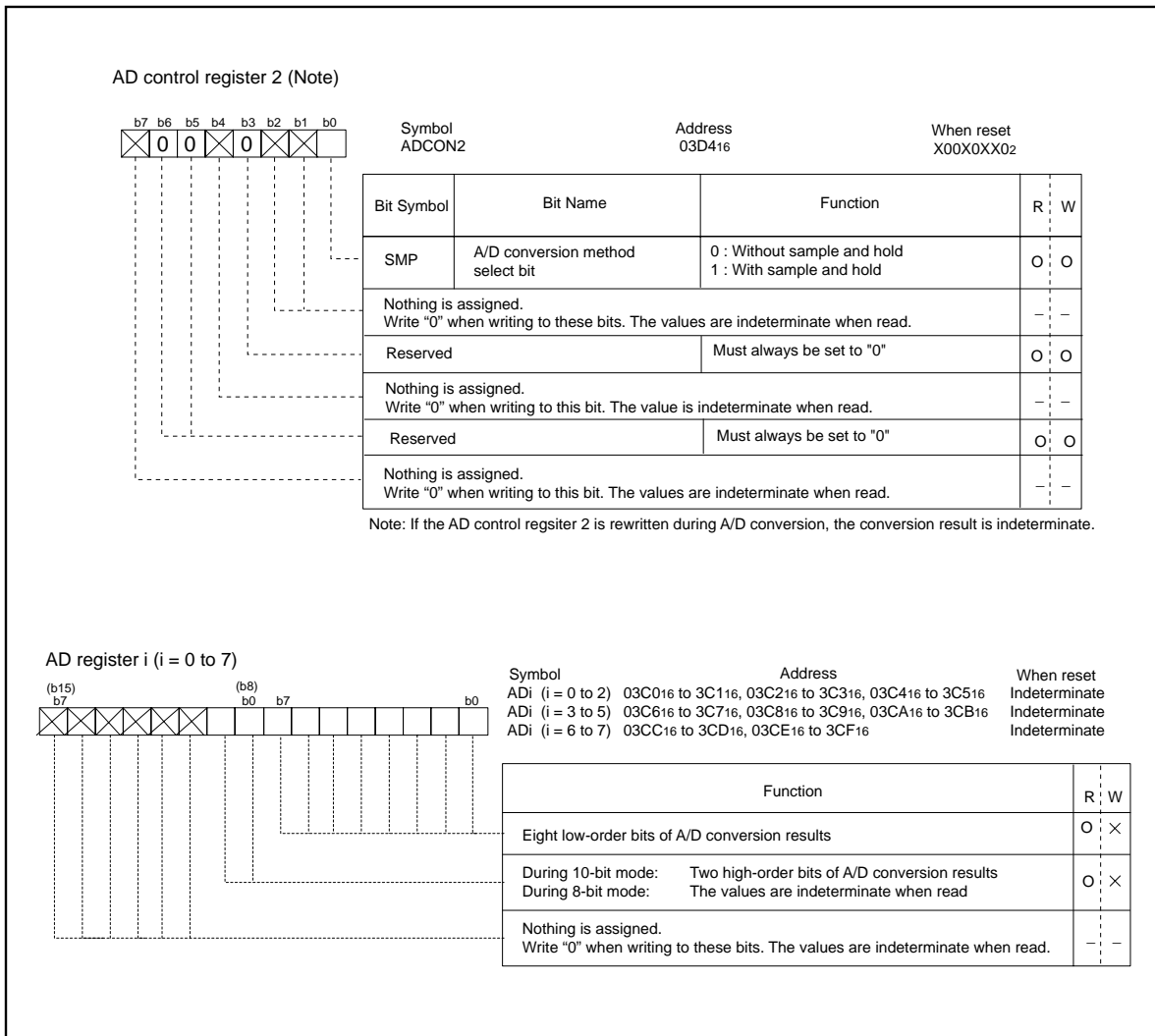


Figure 2.9.3. A/D converter-related registers (2)

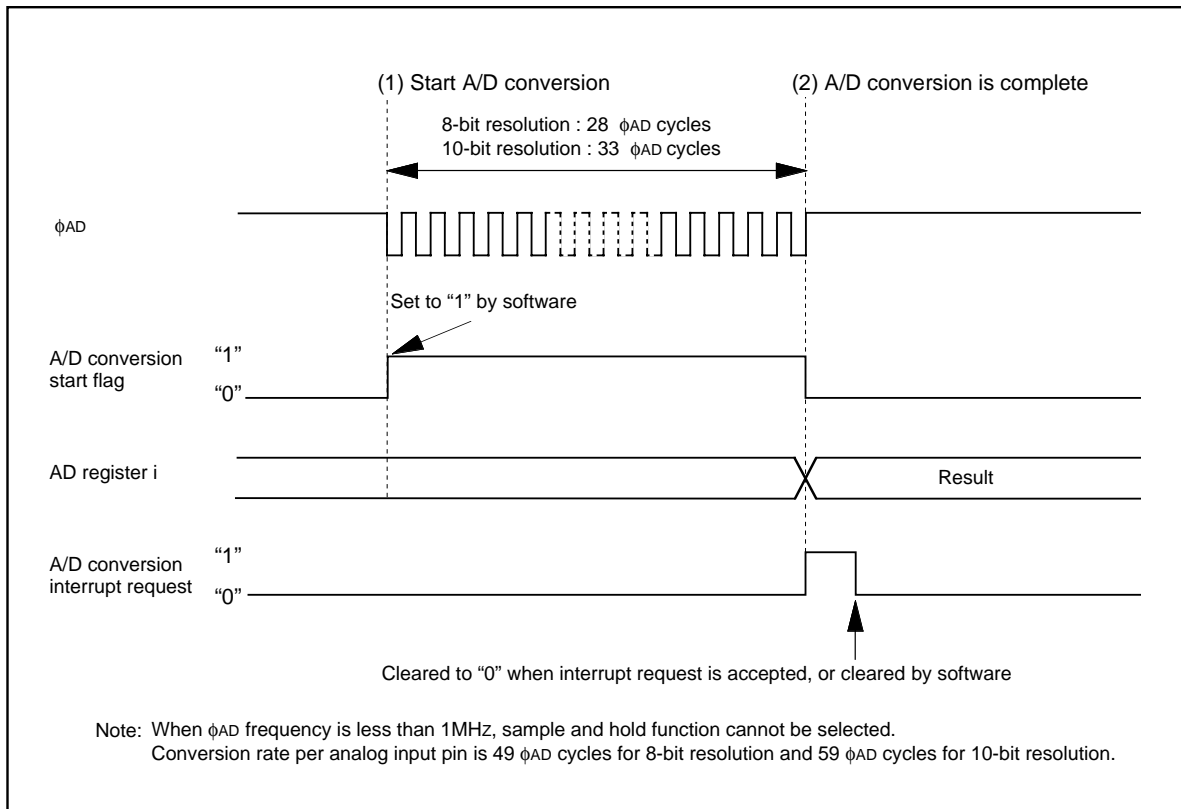
## 2.9.2 Operation of A/D converter (one-shot mode)

In one-shot mode, choose functions from those listed in Table 2.9.2. Operations of the circled items are described below. Figure 2.9.4 shows the operation timing, and Figure 2.9.5 shows the set-up procedure.

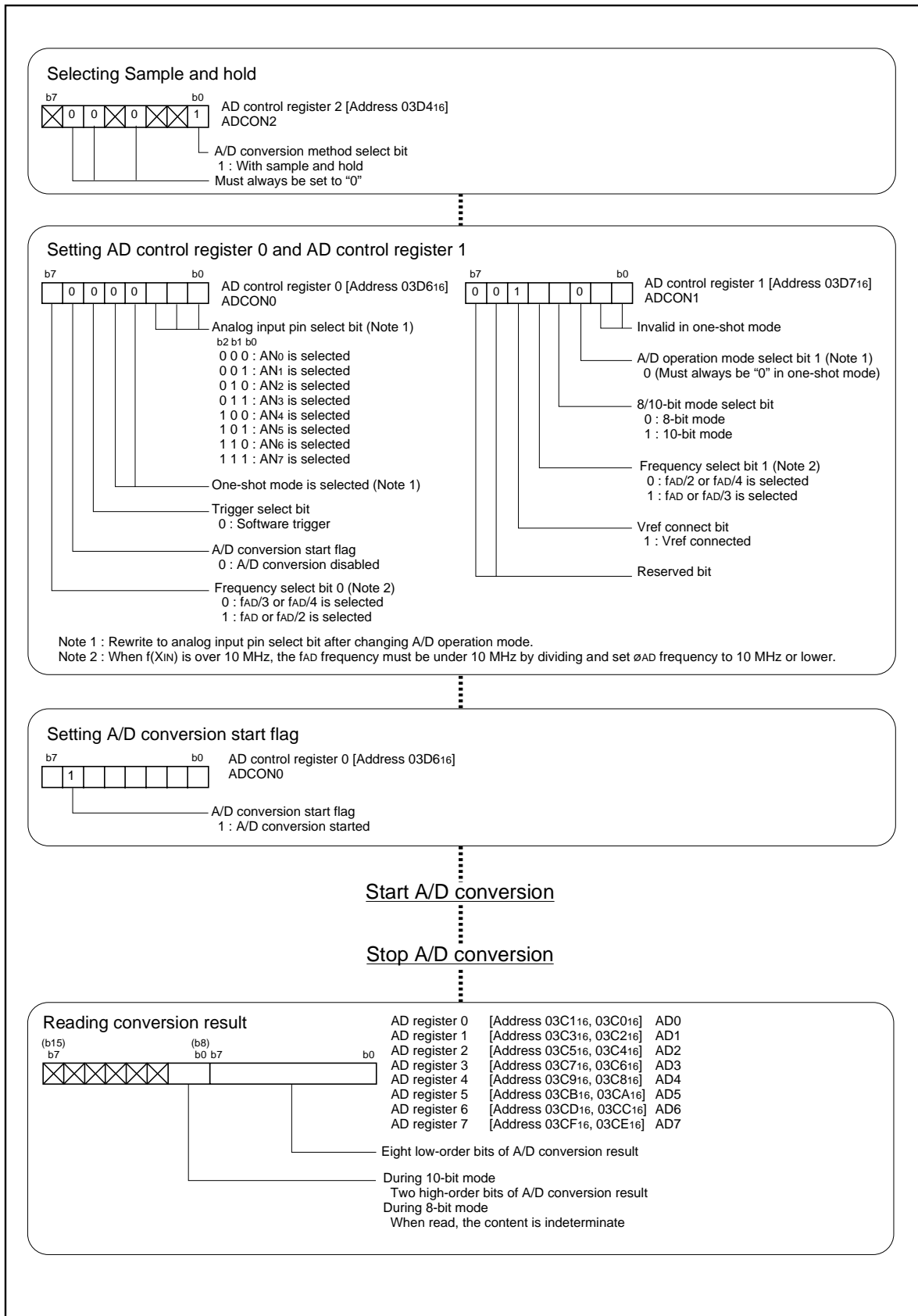
**Table 2.9.2. Chosed functions**

Item	Set-up
Operation clock $\phi_{AD}$	<input type="radio"/> Divided-by-4 $f_{AD}$ / divided-by-3 $f_{AD}$ / divided-by-2 $f_{AD}$ / $f_{AD}$
Resolution	<input type="radio"/> 8-bit / 10-bit
Analog input pin	<input type="radio"/> One of AN <sub>0</sub> pin to AN <sub>7</sub> pin
Trigger for starting A/D conversion	<input type="radio"/> Software trigger
	<input type="radio"/> Trigger by $\overline{AD_{TRG}}$
Sample & Hold	<input type="radio"/> Not activated
	<input type="radio"/> Activated

- Operation (1) Setting the A/D conversion start flag to "1" causes the A/D converter to begin operating.
- (2) After A/D conversion is completed, the content of the successive comparison register (conversion result) is transmitted to AD register i. At this time, the A/D conversion interrupt request bit goes to "1". Also, the A/D conversion start flag goes to "0", and the A/D converter stops operating.



**Figure 2.9.4. Operation timing of one-shot mode**



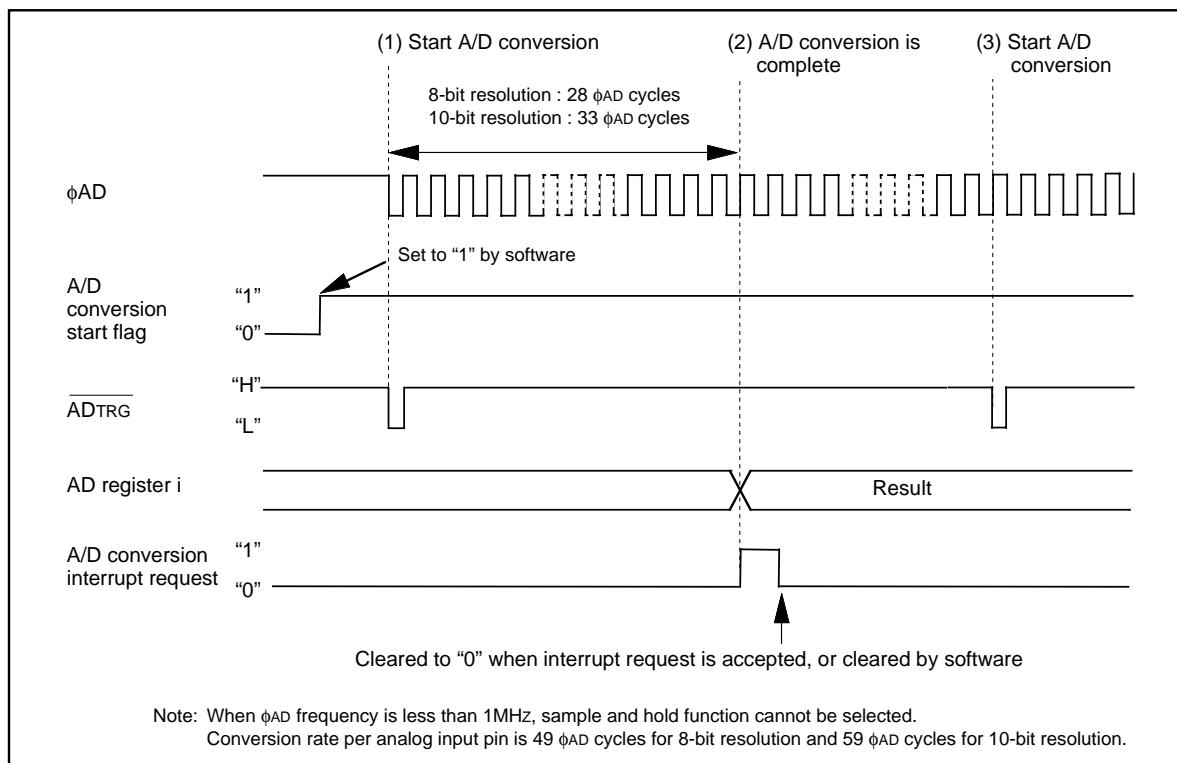
### 2.9.3 Operation of A/D Converter (in one-shot mode, an external trigger selected)

In one-shot mode, choose functions from those listed in Table 2.9.3. Operations of the circled items are described below. Figure 2.9.6 shows timing chart, and Figure 2.9.7 shows the set-up procedure.

**Table 2.9.3. Chosed functions**

Item	Set-up	
Operation clock $\phi_{AD}$	<input type="radio"/>	Divided-by-4 $f_{AD}$ / divided-by-3 $f_{AD}$ / divided-by-2 $f_{AD}$ / $f_{AD}$
Resolution	<input type="radio"/>	8-bit / 10-bit
Analog input pin	<input type="radio"/>	One of AN <sub>0</sub> pin to AN <sub>7</sub> pin
Trigger for starting A/D conversion	<input type="radio"/>	Software trigger
	<input type="radio"/>	Trigger by $\overline{ADTRG}$
Sample & Hold	<input type="radio"/>	Not activated
	<input type="radio"/>	Activated

- Operation
- (1) If the level of the  $\overline{ADTRG}$  changes from "H" to "L" with the A/D conversion start flag set to "1", the A/D converter begins operating.
  - (2) After A/D conversion is completed, the content of the successive comparison register (conversion result) is transmitted to AD register i. At this time, the A/D conversion interrupt request bit goes to "1". Also the A/D converter stops operating.
  - (3) If the level of the  $\overline{ADTRG}$  pin changes from "H" to "L", the A/D converter carries out conversion from step (1) again. If the level of the  $\overline{ADTRG}$  pin changes from "H" to "L" while conversion is in progress, the A/D converter stops the A/D conversion in process, and carries out conversion from step (1) again.



**Figure 2.9.6. Operation timing of one-shot mode, with an external trigger selected**

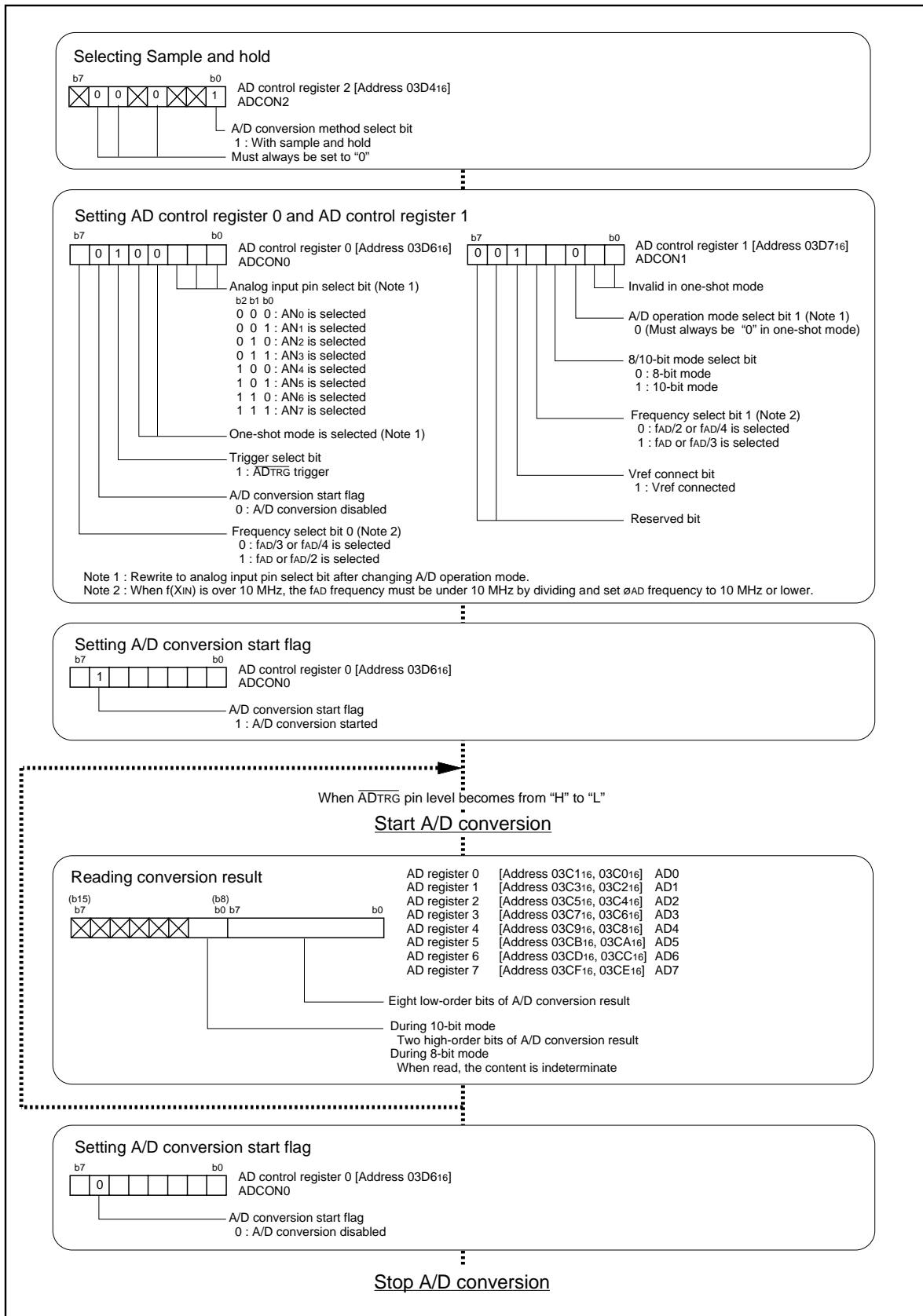


Figure 2.9.7. Set-up procedure of one-shot mode, with an external trigger selected

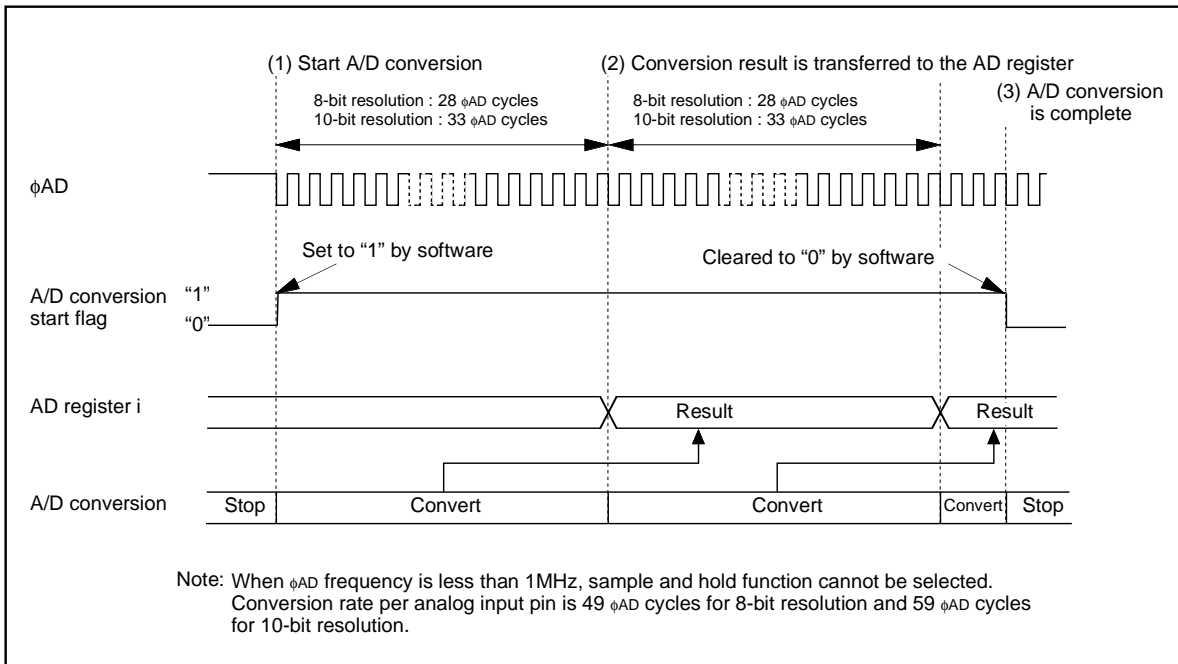
### 2.9.4 Operation of A/D Converter (in repeat mode)

In repeat mode, choose functions from those listed in Table 2.9.4. Operations of the circled items are described below. Figure 2.9.8 shows timing chart, and Figure 2.9.9 shows the set-up procedure.

**Table 2.9.4. Chosed functions**

Item	Set-up	
Operation clock $\phi_{AD}$	<input type="radio"/>	Divided-by-4 $f_{AD}$ / divided-by-3 $f_{AD}$ / divided-by-2 $f_{AD}$ / $f_{AD}$
Resolution	<input type="radio"/>	8-bit / 10-bit
Analog input pin	<input type="radio"/>	One of AN <sub>0</sub> pin to AN <sub>7</sub> pin
Trigger for starting A/D conversion	<input type="radio"/>	Software trigger
	<input type="radio"/>	Trigger by ADTRG
Sample & Hold	<input type="radio"/>	Not activated
	<input type="radio"/>	Activated

- Operation
- (1) Setting the A/D conversion start flag to "1" causes the A/D converter to start operating.
  - (2) After the first conversion is completed, the content of the successive comparison register (conversion result) is transmitted to AD register i. The A/D conversion interrupt request bit does not go to "1".
  - (3) The A/D converter continues operating until the A/D conversion start flag is set to "0" by software. The conversion result is transmitted to AD register i every time a conversion is completed.



**Figure 2.9.8. Operation timing of repeat mode**

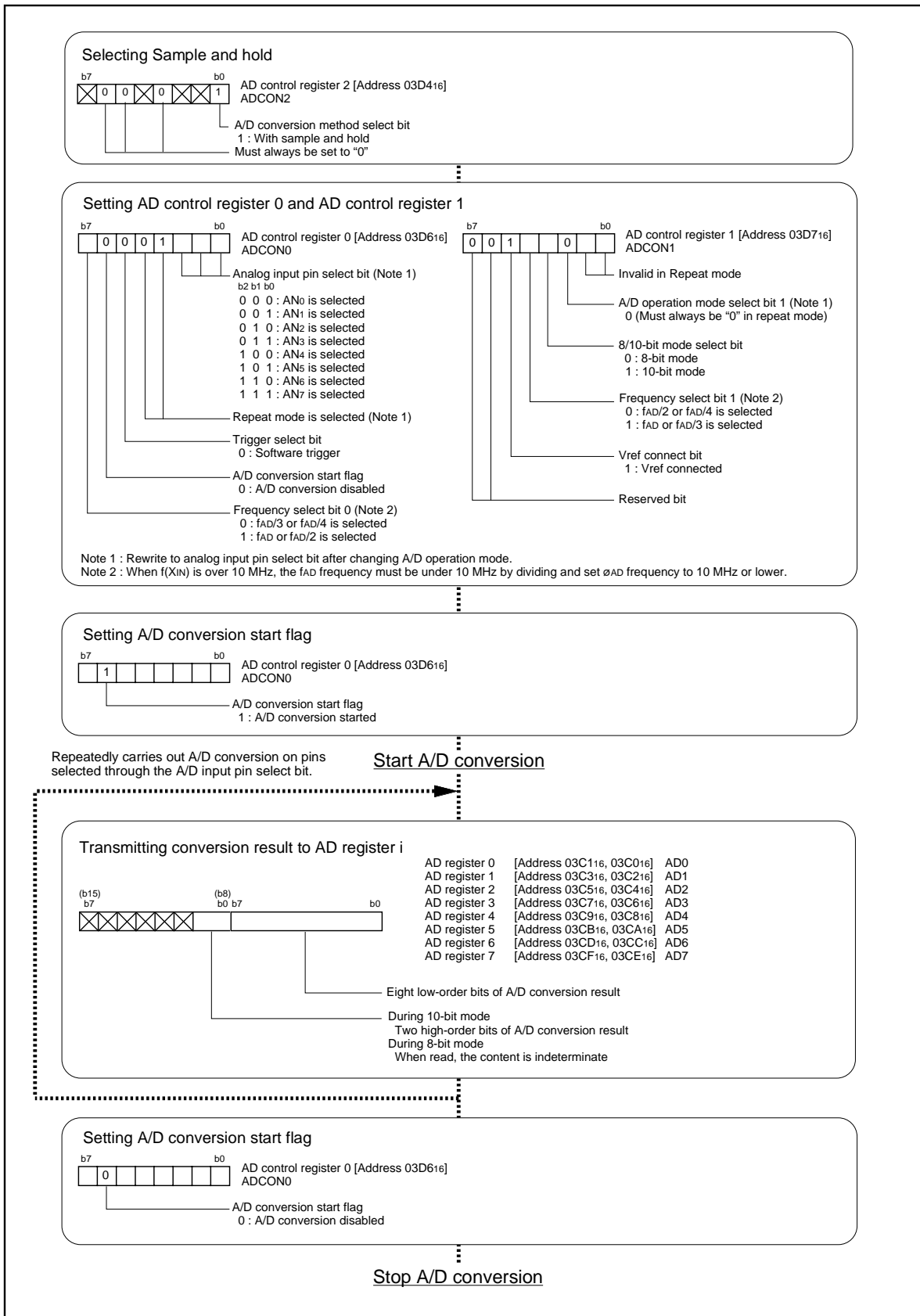


Figure 2.9.9. Set-up procedure of repeat mode



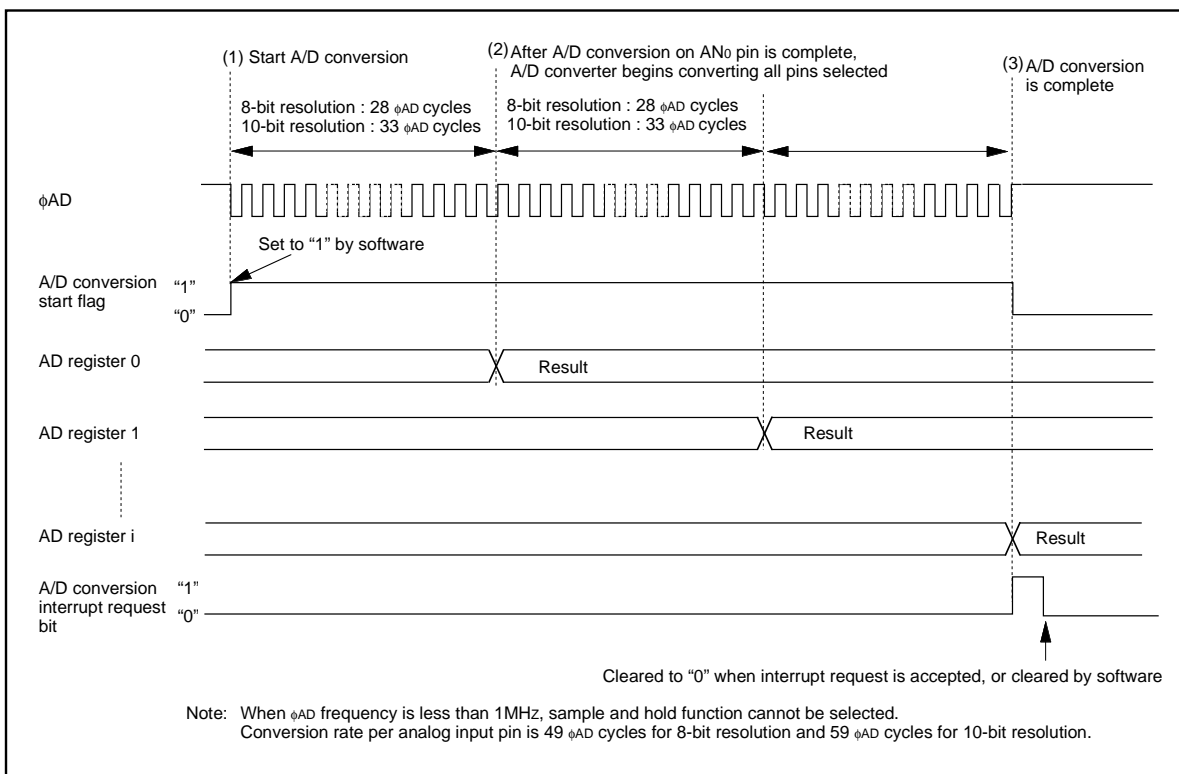
### 2.9.5 Operation of A/D Converter (in single sweep mode)

In single sweep mode, choose functions from those listed in Table 2.9.5. Operations of the circled items are described below. Figure 2.9.10 shows timing chart, and Figure 2.9.11 shows the set-up procedure.

**Table 2.9.5. Chosed functions**

Item	Set-up	Item	Set-up
Operation clock AD	○ Divided-by-4 $f_{AD}$ / divided-by-3 $f_{AD}$ / divided-by-2 $f_{AD}$ / $f_{AD}$	Trigger for starting A/D conversion	○ Software trigger
			○ Trigger by $AD_{TRG}$
Resolution	○ 8-bit / 10-bit	Sample & Hold	○ Not activated
Analog input pin	○ $AN_0$ and $AN_1$ (2 pins) / $AN_0$ to $AN_3$ (4 pins) / $AN_0$ to $AN_5$ (6 pins) / $AN_0$ to $AN_7$ (8 pins)		○ Activated

- Operation
- (1) Setting the A/D conversion start flag to "1" causes the A/D converter to start the conversion on voltage input to the  $AN_0$  pin.
  - (2) After the A/D conversion of voltage input to the  $AN_0$  pin is completed, the content of the successive comparison register (conversion result) is transmitted to AD register 0. The A/D converter converts all analog input pins selected by the user. The conversion result is transmitted to AD register  $i$  corresponding to each pin, every time conversion on one pin is completed.
  - (3) When the A/D conversion on all the analog input pins selected is completed, the A/D conversion interrupt request bit goes to "1". At this time, the A/D conversion start flag goes to "0". The A/D converter stops operating.



**Figure 2.9.10. Operation timing of single sweep mode**

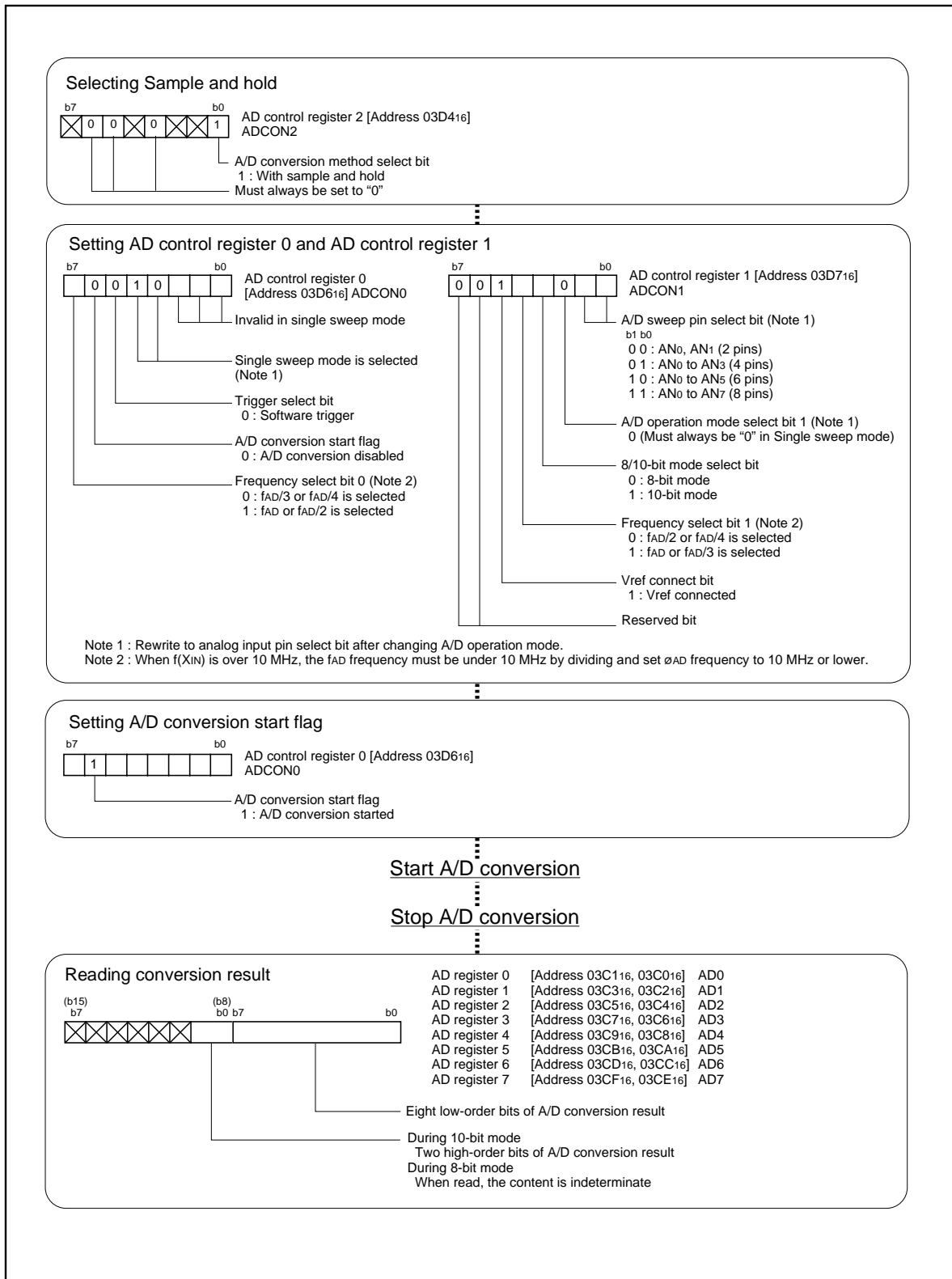


Figure 2.9.11. Set-up procedure of single sweep mode

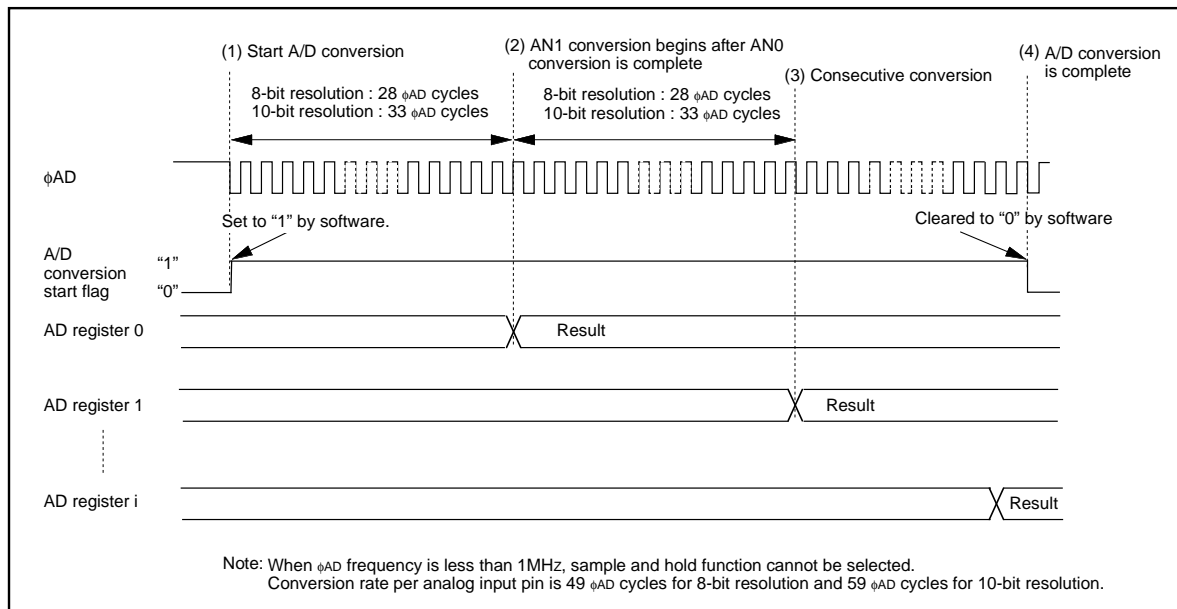
### 2.9.6 Operation of A/D Converter (in repeat sweep mode 0)

In repeat sweep 0 mode, choose functions from those listed in Table 2.9.6. Operations of the circled items are described below. Figure 2.9.12 shows timing chart, and Figure 2.9.13 shows the set-up procedure.

**Table 2.9.6. Chosed functions**

Item	Set-up	Item	Set-up
Operation clock $\phi_{AD}$	○ Divided-by-4 $f_{AD}$ / divided-by-3 $f_{AD}$ / divided-by-2 $f_{AD}$ / $f_{AD}$	Trigger for starting A/D conversion	○ Software trigger
			○ Trigger by $\overline{ADTRG}$
Resolution	○ 8-bit / 10-bit	Sample & Hold	○ Not activated
Analog input pin	○ $AN_0$ and $AN_1$ (2 pins) / $AN_0$ to $AN_3$ (4 pins) / $AN_0$ to $AN_5$ (6 pins) / $AN_0$ to $AN_7$ (8 pins)		○ Activated

- Operation
- (1) Setting the A/D conversion start flag to "1" causes the A/D converter to start the conversion on voltage input to the  $AN_0$  pin.
  - (2) After the A/D conversion of voltage input to the  $AN_0$  pin is completed, the content of the successive comparison register (conversion result) is transmitted to AD register 0.
  - (3) The A/D converter converts all pins selected by the user. The conversion result is transmitted to AD register  $i$  corresponding to each pin every time A/D conversion on the pin is completed. The A/D conversion interrupt request bit does not go to "1".
  - (4) The A/D converter continues operating until the A/D conversion start flag is set to "0" by software.



**Figure 2.9.12. Operation timing of repeat sweep 0 mode**

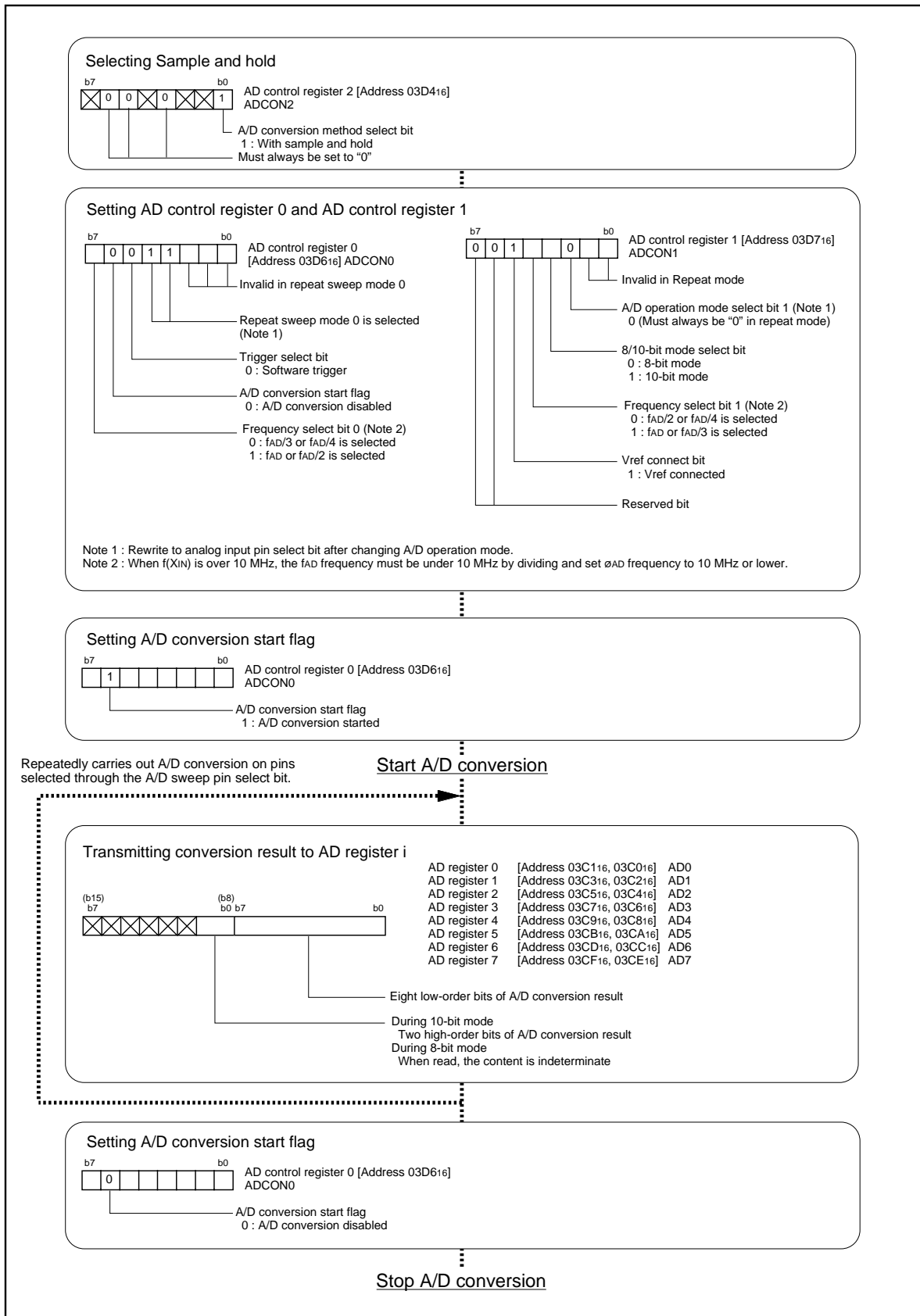


Figure 2.9.13. Set-up procedure of repeat sweep 0 mode

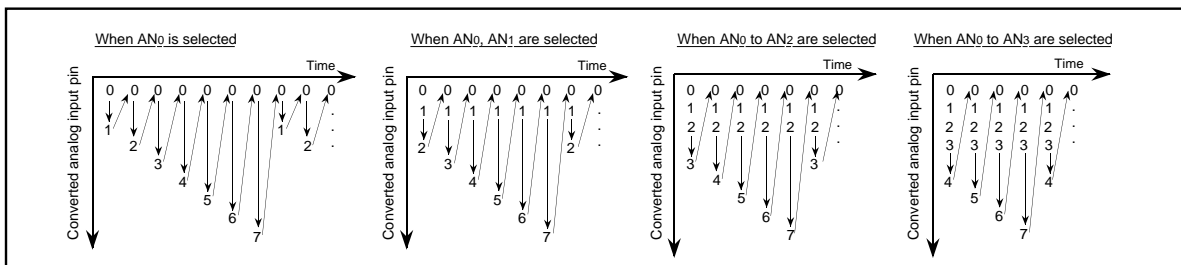
### 2.9.7 Operation of A/D Converter (in repeat sweep mode 1)

In repeat sweep 1 mode, choose functions from those listed in Table 2.9.7. Operations of the circled items are described below. Figure 2.9.14 shows ANi pin's sweep sequence, Figure 2.9.15 shows timing chart, and Figure 2.9.16 shows the set-up procedure.

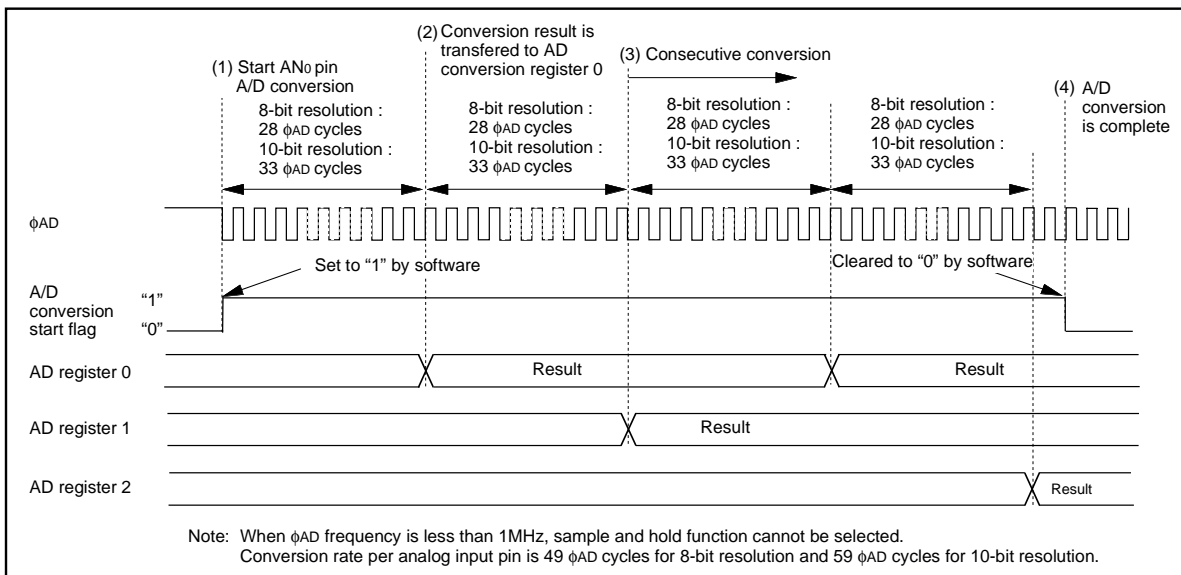
**Table 2.9.7. Chosed functions**

Item	Set-up	Item	Set-up
Operation clock AD	<input type="radio"/> Divided-by-4 f <sub>AD</sub> / divided-by-3 f <sub>AD</sub> / divided-by-2 f <sub>AD</sub> / f <sub>AD</sub>	Trigger for starting A/D conversion	<input type="radio"/> Software trigger
Resolution	<input type="radio"/> 8-bit / 10-bit		Trigger by $\overline{ADTRG}$
Analog input pin	<input type="radio"/> AN <sub>0</sub> (1 pin) / AN <sub>0</sub> and AN <sub>1</sub> (2 pins) / AN <sub>0</sub> to AN <sub>2</sub> (3 pins) / AN <sub>0</sub> to AN <sub>3</sub> (4 pins)	Sample & Hold	<input type="radio"/> Not activated
			<input type="radio"/> Activated

- Operation
- (1) Setting the A/D conversion start flag to "1" causes the A/D converter to start conversion on voltage input to the AN<sub>0</sub> pin.
  - (2) After the A/D conversion on voltage input to the AN<sub>0</sub> pin is completed, the content of the successive comparison register (conversion result) is transmitted to AD register 0.
  - (3) Every time the A/D converter carries out A/D conversion on a selected analog input pin, the A/D converter carries out A/D conversion on only one unselected pin, and then the A/D converter carries out A/D conversion from the AN<sub>0</sub> pin again. The conversion result is transmitted to the corresponding AD register i every time conversion on a pin is completed. The A/D conversion interrupt request bit does not go to "1".
  - (4) The A/D converter continues operating until software goes the A/D conversion start flag to "0".



**Figure 2.9.14. ANi pin's sweep sequence in repeat sweep mode**



**Figure 2.9.15. Operation timing of repeat sweep 1 mode**

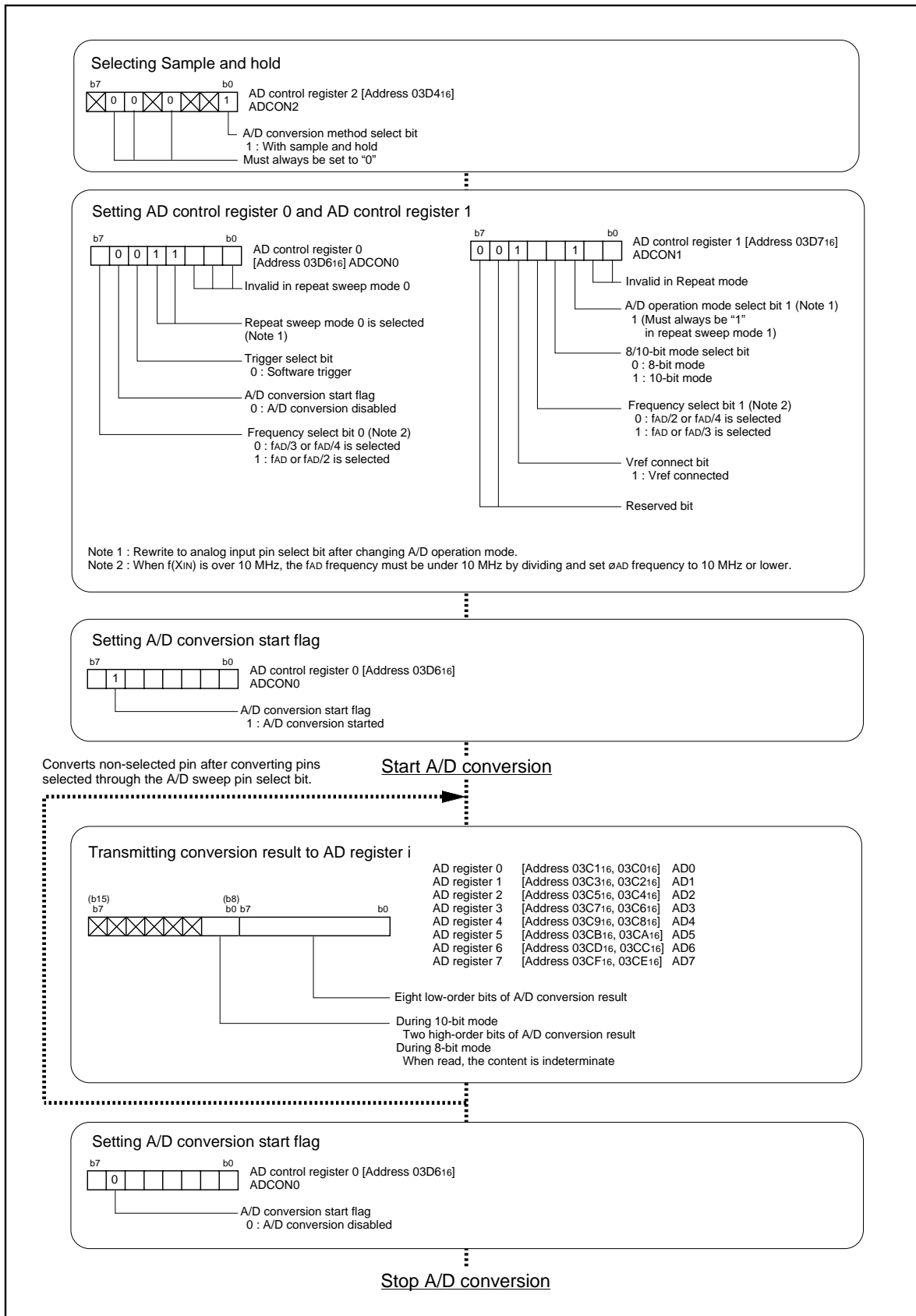
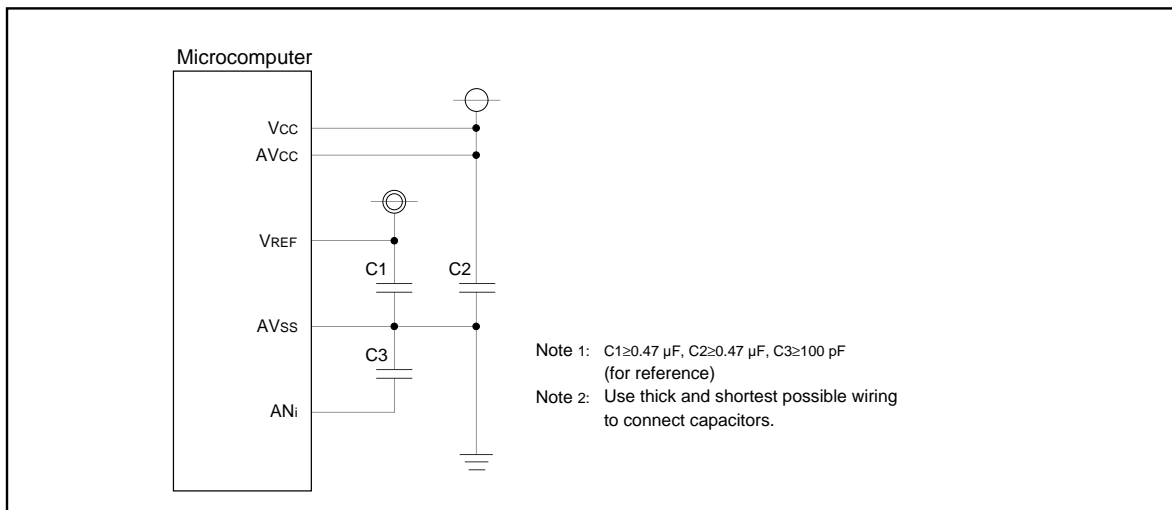


Figure 2.9.16. Set-up procedure of repeat sweep 1 mode

### 2.9.8 Precautions for A/D Converter

- (1) Write to each bit (except bit 6) of AD control register 0, to each bit of AD control register 1, and to bit 0 of AD control register 2 when A/D conversion is stopped (before a trigger occurs). In particular, when the Vref connection bit is changed from 0 to 1, start A/D conversion after an elapse of 1  $\mu$ s or longer.
- (2) To reduce conversion error due to noise, connect a voltage to the AVcc pin and to the Vref pin from an independent source. It is recommended to connect a capacitor between the AVss pin and the AVcc pin, between the AVss pin and the Vref pin, and between the AVss pin and the analog input pin (ANi). Figure 2.9.17 shows the an example of connecting the capacitors to these pins.



**Figure 2.9.17. Use of capacitors to reduce noise**

- (3) Set the direction register of the following ports to input: the port corresponding to a pin to be used as an analog input pin and external trigger input pin (P93).
- (4) Rewrite to analog input pin after changing A/D operation mode.
- (5) When using the one-shot or single sweep mode
  - Confirm that A/D conversion is complete before reading the AD register.
  - (Note: When A/D conversion interrupt request bit is set, it shows that A/D conversion is completed.)
- (6) When using the repeat mode or repeat sweep mode 0 or 1
  - Use the undivided main clock as the internal CPU clock.
- (7) In using a key-input interrupt, none of the 8 pins (AN0 through AN7) can be used as an A/D conversion port (if the A/D input voltage goes to "L" level, a key-input interrupt occurs).
- (8) Use  $\phi$ AD under 10 MHz. When XIN is over 10 MHz, divide it.

### 2.9.9 Method of A/D Conversion (10-bit mode)

(1) The A/D converter compares the reference voltage (Vref) generated internally based on the contents of the successive comparison register with the analog input voltage (VIN) input from the analog input pin. The comparison result is stored in the successive comparison register and then VIN is converted to digital value (successive comparison method). If a trigger occurs, the A/D converter carries out the following:

1. Fixes bit 9 of the successive comparison register.

Compares Vref with VIN: [In this instance, the contents of the successive comparison register are "1000000002" (default).]

Bit 9 of the successive comparison register varies depending on the comparison result as follows.

If  $V_{ref} < V_{IN}$ , then "1" is assigned to bit 9.

If  $V_{ref} > V_{IN}$ , then "0" is assigned to bit 9.

2. Fixes bit 8 of the successive comparison register.

Sets bit 8 of the successive comparison register to "1", then compares Vref with VIN.

Bit 8 of the successive comparison register varies depending on the comparison result as follows:

If  $V_{ref} < V_{IN}$ , then "1" is assigned to bit 8.

If  $V_{ref} > V_{IN}$ , then "0" is assigned to bit 8.

3. Fixes bit 7 through bit 0 of the successive comparison register.

Carries out step 2 above on bit 7 through bit 0.

After bit 0 is fixed, the contents of the successive comparison register (conversion result) are transmitted to AD register i.

Vref is generated based on the latest content of the successive comparison register. Table 2.9.8 shows the relationship of the successive comparison register contents and Vref. Table 2.9.9 shows how the successive comparison register and Vref vary while A/D conversion is in progress. Figure 2.9.18 shows theoretical A/D conversion characteristics.

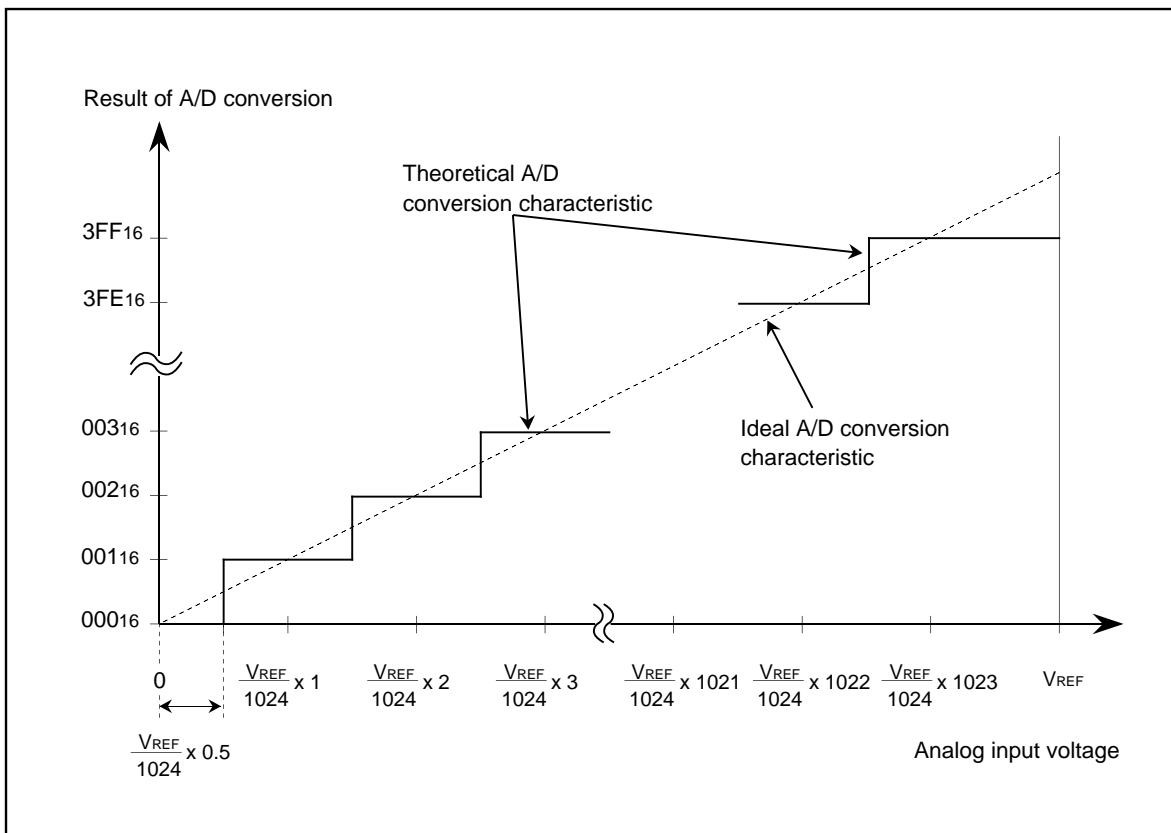
**Table 2.9.8. Relationship of the successive comparison register contents and Vref**

Successive approximation register : n	Vref (V)
0	0
1 to 1023	$\frac{V_{REF}}{1024} \times n - \frac{V_{REF}}{2048}$



**Table 2.9.9. Variation of the successive comparison register and Vref while A/D conversion is in progress (10-bit mode)**

	Successive approximation register	Vref change
A/D converter stopped	b9 <span style="border: 1px solid black; padding: 2px;">1 0 0 0 0 0 0 0 0 0</span> b0	$\frac{V_{REF}}{2}$ [V]
1st comparison	<span style="border: 1px solid black; padding: 2px;">1 0 0 0 0 0 0 0 0 0</span>	$\frac{V_{REF}}{2} - \frac{V_{REF}}{2048}$ [V]
2nd comparison	n9 <span style="border: 1px solid black; padding: 2px;">1 0 0 0 0 0 0 0 0 0</span>	$\frac{V_{REF}}{2} \pm \frac{V_{REF}}{4} - \frac{V_{REF}}{2048}$ [V] $\left( \begin{matrix} n_9 = 1 & + & \frac{V_{REF}}{4} \\ n_9 = 0 & - & \frac{V_{REF}}{4} \end{matrix} \right)$
3rd comparison	n9 n8 <span style="border: 1px solid black; padding: 2px;">1 0 0 0 0 0 0 0 0 0</span>	$\frac{V_{REF}}{2} \pm \frac{V_{REF}}{4} \pm \frac{V_{REF}}{8} - \frac{V_{REF}}{2048}$ [V] $\left( \begin{matrix} n_8 = 1 & + & \frac{V_{REF}}{8} \\ n_8 = 0 & - & \frac{V_{REF}}{8} \end{matrix} \right)$
...	...	...
10th comparison	n9 n8 n7 n6 n5 n4 n3 n2 n1 <span style="border: 1px solid black; padding: 2px;">1</span>	$\frac{V_{REF}}{2} \pm \frac{V_{REF}}{4} \pm \frac{V_{REF}}{8} \pm \dots \pm \frac{V_{REF}}{1024} - \frac{V_{REF}}{2048}$ [V]
Conversion complete	n9 n8 n7 n6 n5 n4 n3 n2 n1 n0	
	This data transfers to the bit 0 to bit 9 of AD register i.	



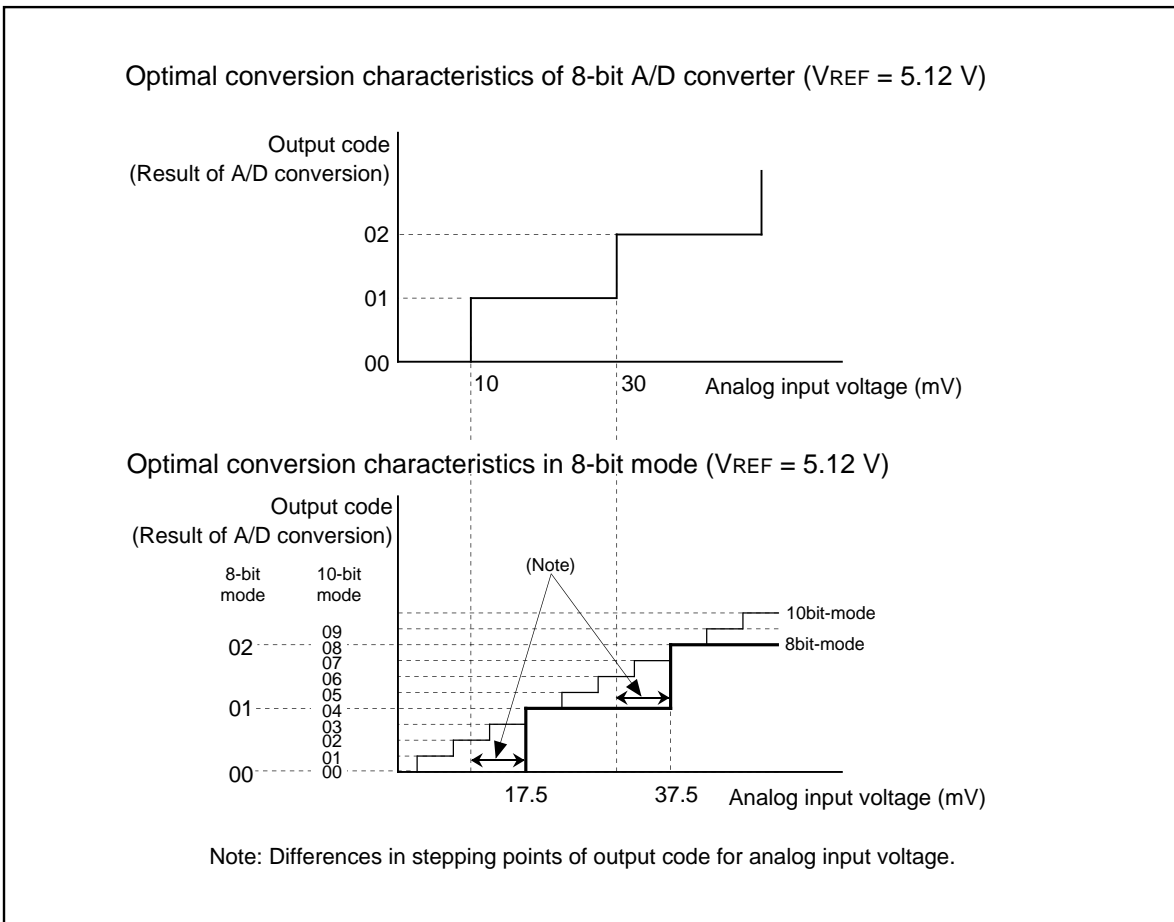
**Figure 2.9.18. Theoretical A/D conversion characteristics (10-bit mode)**

### 2.9.10 Method of A/D Conversion (8-bit mode)

(1) In 8-bit mode, 8 higher-order bits of the 10-bit successive comparison register becomes A/D conversion result. Hence, if compared to a result obtained by using an 8-bit A/D converter, the voltage compared is different by  $3 V_{REF}/2048$  (see what are underscored in Table 2.9.10), and differences in stepping points of output codes occur as shown in Figure 2.9.19.

**Table 2.9.10. The comparison voltage in 8-bit mode compared to 8-bit A/D converter**

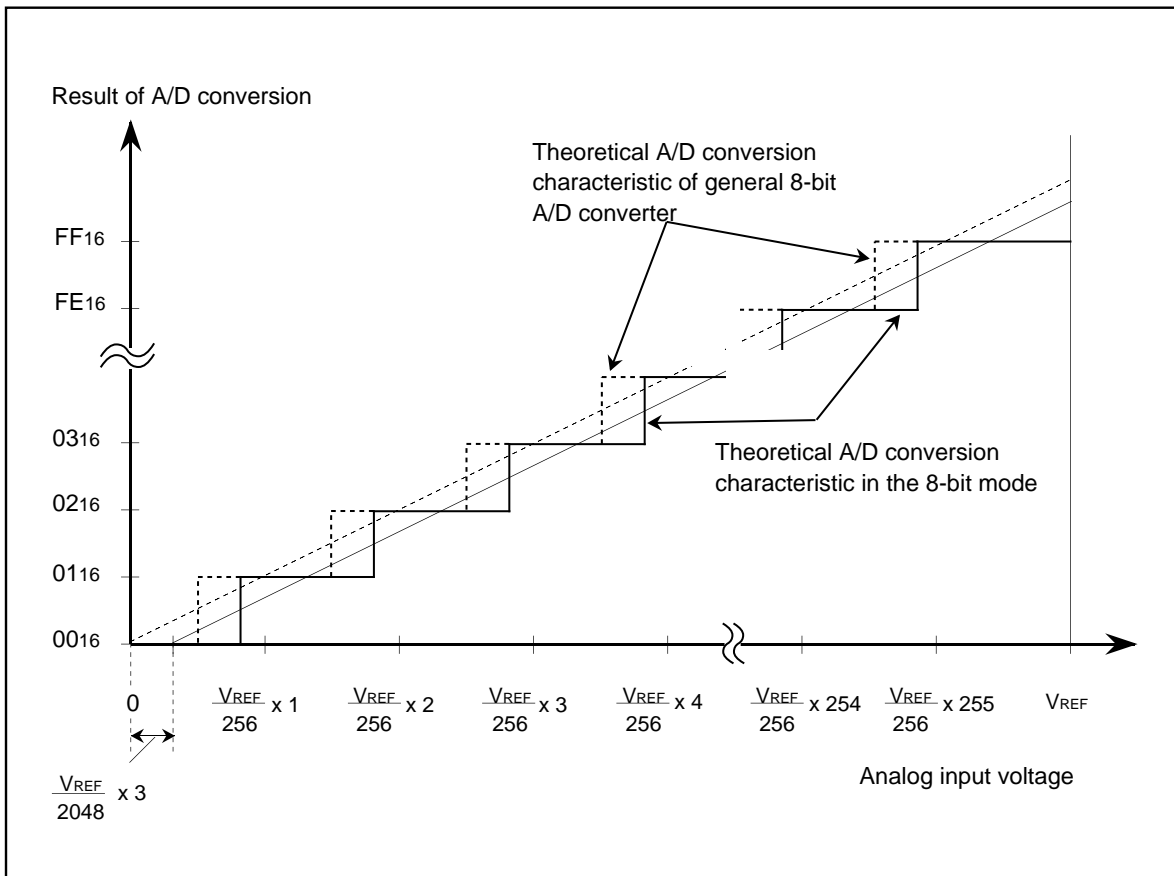
		8-bit mode	8-bit A/D converter
Comparison voltage $V_{ref}$	n = 0	0	0
	n = 1 to 255	$\frac{V_{REF}}{2^8} \times n - \frac{V_{REF}}{2^{10}} \times 0.5$	$\frac{V_{REF}}{2^8} \times n - \frac{V_{REF}}{2^8} \times 0.5$



**Figure 2.9.19. The level conversion characteristics of 8-bit mode and 8-bit A/D converter**

**Table 2.9.11. Variation of the successive comparison register and Vref while A/D conversion is in progress (8-bit mode)**

	Successive approximation register	Vref change
A/D converter stopped	$\begin{matrix} b9 & & & & & & & & & & b0 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$	$\frac{V_{REF}}{2}$ [V]
1st comparison	$\begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$	$\frac{V_{REF}}{2} - \frac{V_{REF}}{2048}$ [V]
2nd comparison	$\begin{matrix} n9 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \uparrow & & & & & & & & & & \\ \text{1st comparison result} & & & & & & & & & & \end{matrix}$	$\frac{V_{REF}}{2} \pm \frac{V_{REF}}{4} - \frac{V_{REF}}{2048}$ [V] $\begin{matrix} n9=1 & + & \frac{V_{REF}}{4} \\ n9=0 & - & \frac{V_{REF}}{4} \end{matrix}$
3rd comparison	$\begin{matrix} n9 & n8 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \uparrow & \uparrow & & & & & & & & & \\ \text{2nd comparison result} & & & & & & & & & & \end{matrix}$	$\frac{V_{REF}}{2} \pm \frac{V_{REF}}{4} \pm \frac{V_{REF}}{8} - \frac{V_{REF}}{2048}$ [V] $\begin{matrix} n8=1 & + & \frac{V_{REF}}{8} \\ n8=0 & - & \frac{V_{REF}}{8} \end{matrix}$
...	...	...
8th comparison	$\begin{matrix} n9 & n8 & n7 & n6 & n5 & n4 & n3 & 1 & 0 & 0 & 0 \end{matrix}$	$\frac{V_{REF}}{2} \pm \frac{V_{REF}}{4} \pm \frac{V_{REF}}{8} \pm \dots \pm \frac{V_{REF}}{256} - \frac{V_{REF}}{2048}$ [V]
Conversion complete	$\begin{matrix} n9 & n8 & n7 & n6 & n5 & n4 & n3 & n2 & 0 & 0 & 0 \end{matrix}$ <p>This data transfers to bit 0 to bit 7 of AD register i.</p>	



**Figure 2.9.20. Theoretical A/D conversion characteristics (8-bit mode)**

### 2.9.11 Absolute Accuracy and Differential Non-Linearity Error

- **Absolute accuracy**

Absolute accuracy is the difference between output code based on the theoretical A/D conversion characteristics, and actual A/D conversion result. When measuring absolute accuracy, the voltage at the middle point of the width of analog input voltage (1-LSB width), that can meet the expectation of outputting an equal code based on the theoretical A/D conversion characteristics, is used as an analog input voltage. For example, if 10-bit resolution is used and if  $V_{REF}$  (reference voltage) = 5.12 V, then 1-LSB width becomes 5 mV, and 0 mV, 5 mV, 10 mV, 15 mV, 20 mV, ... are used as analog input voltages. If analog input voltage is 25 mV, "absolute accuracy =  $\pm 3\text{LSB}$ " refers to the fact that actual A/D conversion falls on a range from "0021<sub>16</sub>" to "0081<sub>16</sub>" though an output code, "0051<sub>16</sub>", can be expected from the theoretical A/D conversion characteristics. Zero error and full-scale error are included in absolute accuracy.

Also, all the output codes for analog input voltage between  $V_{REF}$  and  $V_{CC}$  becomes "3FF1<sub>16</sub>".

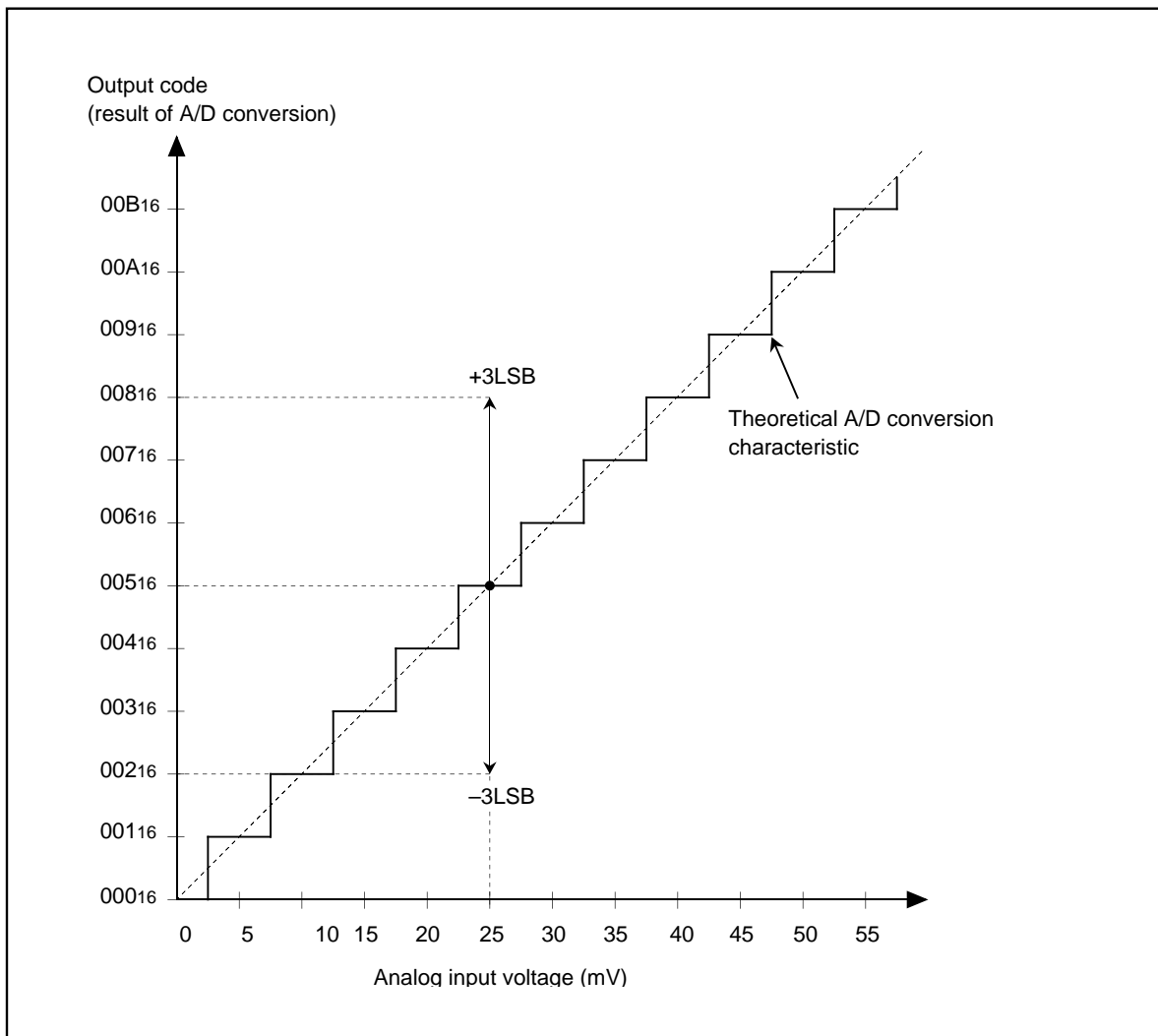


Figure 2.9.21. Absolute accuracy (10-bit resolution)

- **Differential non-linearity error**

Differential non-linearity error refers to the difference between 1-LSB width based on the theoretical A/D conversion characteristics (an analog input width that can meet the expectation of outputting an equal code) and an actually measured 1-LSB width (analog input voltage width that outputs an equal code). If 10-bit resolution is used and if  $V_{REF}$  (reference voltage) = 5.12 V, "differential non-linearity error =  $\pm 1\text{LSB}$ " refers to the fact that 1-LSB width actually measured falls on a range from 0 mV to 10 mV though 1-LSB width based on the theoretical A/D conversion characteristics is 5 mV.

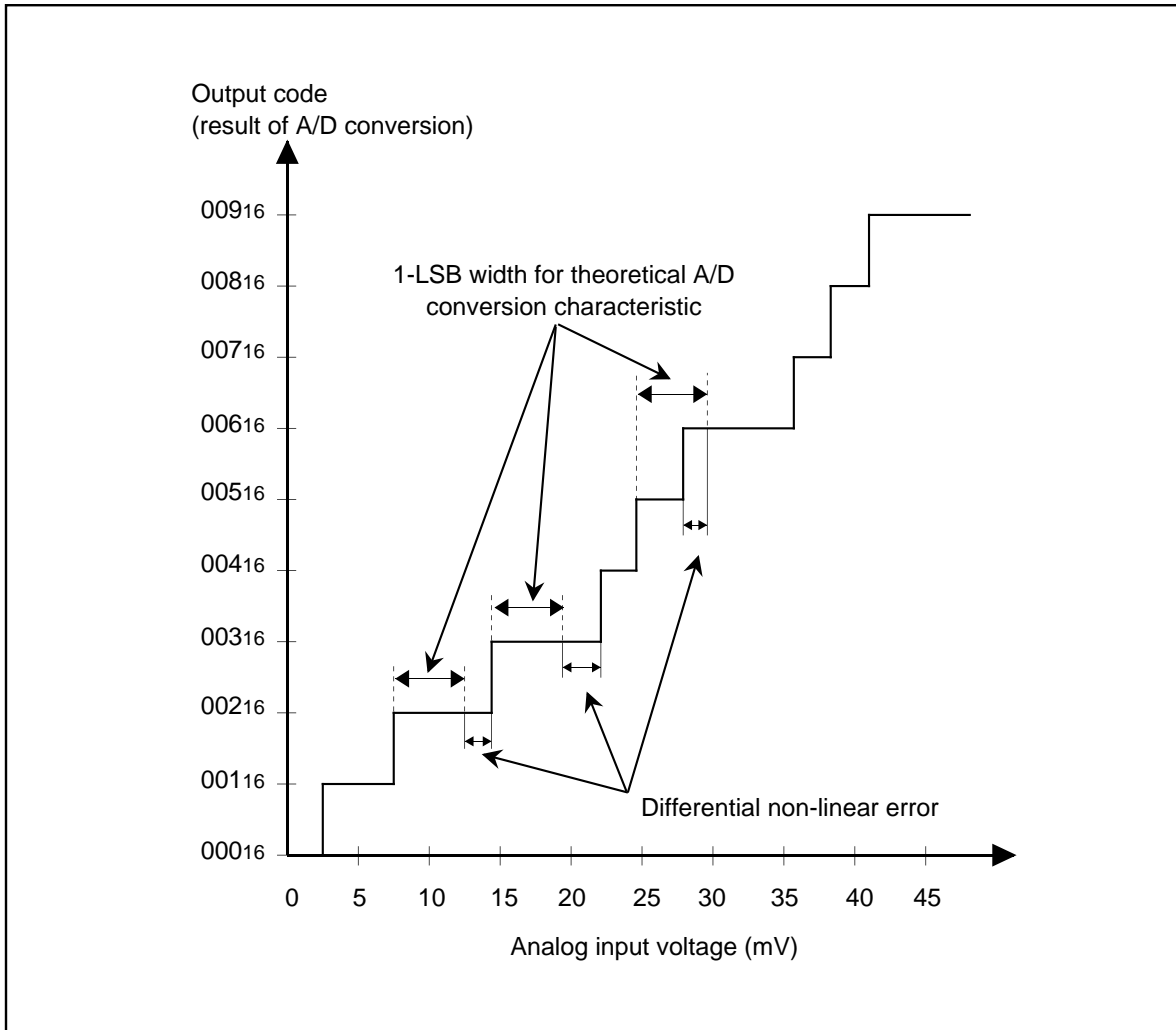


Figure 2.9.22. Differential non-linearity error (10-bit resolution)

### 2.9.12 Internal Equivalent Circuit of Analog Input

Figure 2.9.23 shows the internal equivalent circuit of analog input.

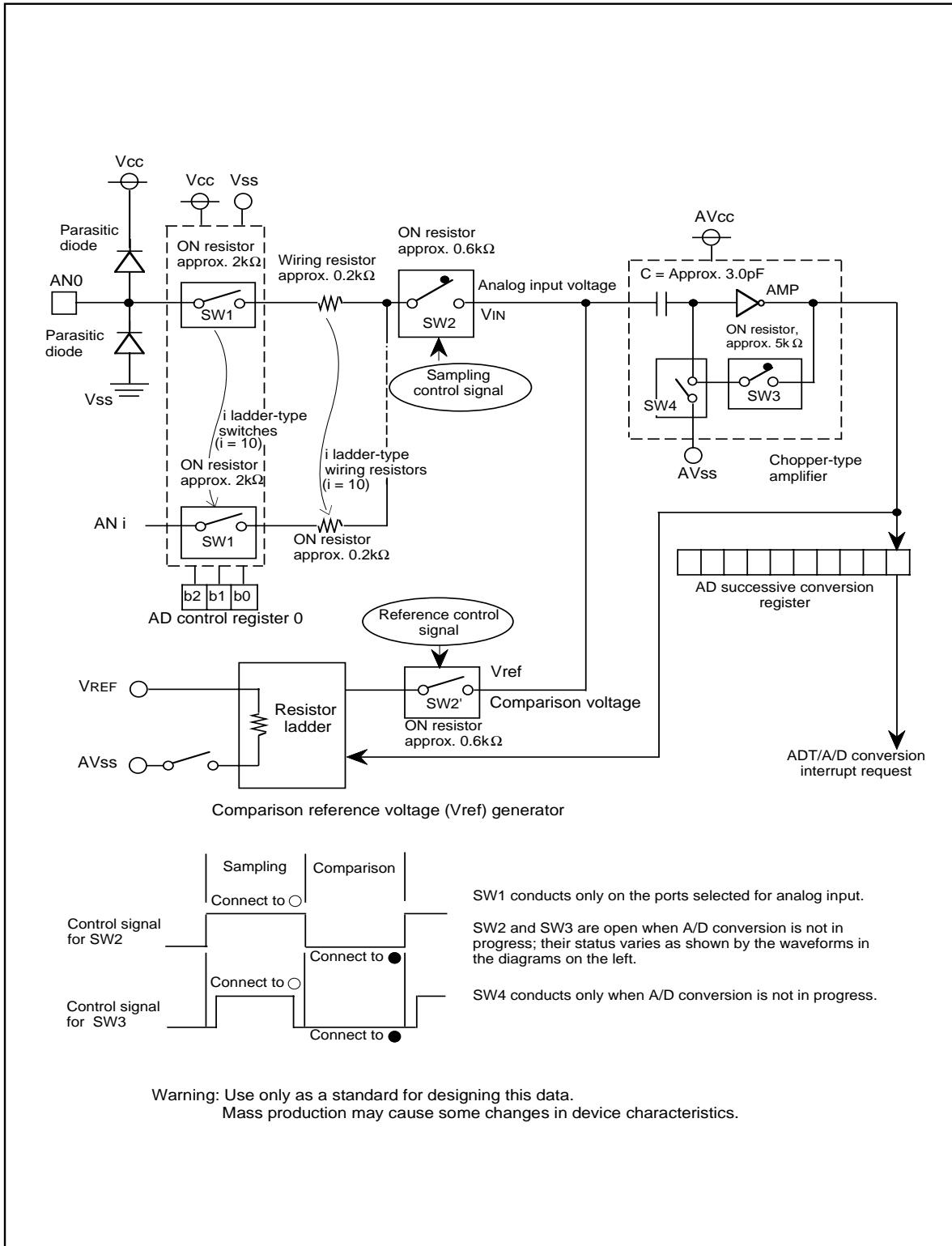


Figure 2.9.23. Internal equivalent circuit to analog input

### 2.9.13 Sensor's Output Impedance under A/D Conversion (reference value)

To carry out A/D conversion properly, charging the internal capacitor C shown in Figure 2.9.23 has to be completed within a specified period of time. With T as the specified time, time T is the time that switches SW2 and SW3 are connected to O in Figure 2.9.23. Let output impedance of sensor equivalent circuit be R0, microcomputer's internal resistance be R, precision (error) of the A/D converter be X, and the A/D converter's resolution be Y (Y is 1024 in the 10-bit mode, and 256 in the 8-bit mode).

$$V_C \text{ is generally } V_C = V_{IN} \left\{ 1 - e^{-\frac{t}{C(R_0 + R)}} \right\}$$

$$\text{And when } t = T, \quad V_C = V_{IN} - \frac{X}{Y} V_{IN} = V_{IN} \left( 1 - \frac{X}{Y} \right)$$

$$e^{-\frac{T}{C(R_0 + R)}} = \frac{X}{Y}$$

$$-\frac{T}{C(R_0 + R)} = \ln \frac{X}{Y}$$

$$\text{Hence, } R_0 = -\frac{T}{C \cdot \ln \frac{X}{Y}} - R$$

With the model shown in Figure 2.9.24 as an example, when the difference between  $V_{IN}$  and  $V_C$  becomes 0.1LSB, we find impedance R0 when voltage between pins  $V_C$  changes from 0 to  $V_{IN} - (0.1/1024) V_{IN}$  in time T. (0.1/1024) means that A/D precision drop due to insufficient capacitor charge is held to 0.1LSB at time of A/D conversion in the 10-bit mode. Actual error however is the value of absolute precision added to 0.1LSB. When  $f(X_{IN}) = 10$  MHz,  $T = 0.3$   $\mu$ s in the A/D conversion mode with sample & hold. Output impedance R0 for sufficiently charging capacitor C within time T is determined as follows.

$T = 0.3$   $\mu$ s,  $R = 7.8$  k $\Omega$ ,  $C = 3$  pF,  $X = 0.1$ , and  $Y = 1024$ . Hence,

$$R_0 = -\frac{0.3 \times 10^{-6}}{3.0 \times 10^{-12} \cdot \ln \frac{0.1}{1024}} - 7.8 \times 10^3 \approx 3.0 \times 10^3$$

Thus, the allowable output impedance of the sensor circuit capable of thoroughly driving the A/D converter turns out to be approximately 3.0 k $\Omega$ . Tables 2.9.12 and 2.9.13 show output impedance values based on the LSB values.

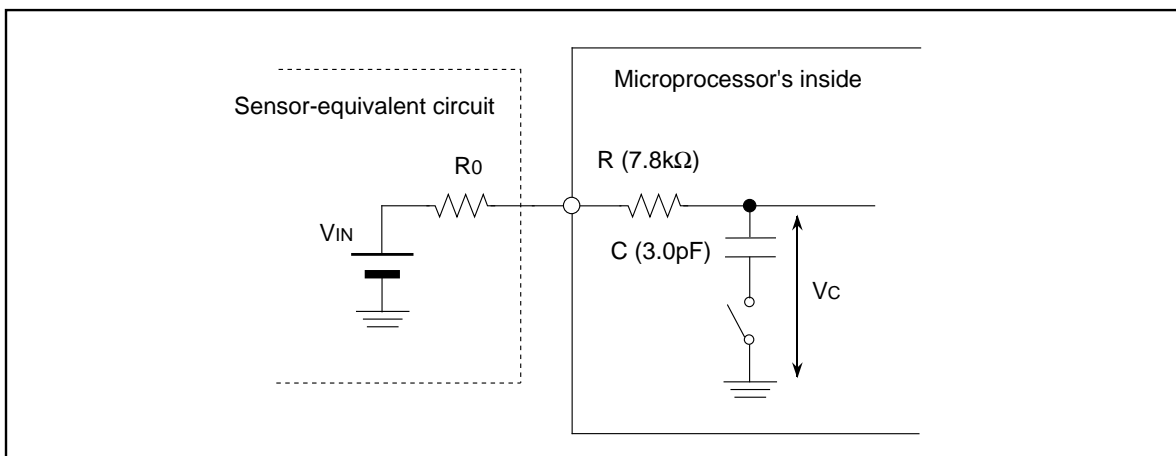


Figure 2.9.24 A circuit equivalent to the A/D conversion terminal

Table 2.9.12. Relation between output impedance and precision (error) of A/D converter (10-bit mode) Reference value

f(Xin) (MHz)	Cycle ( $\mu$ s)	Sampling time ( $\mu$ s)	R (k)	C (pF)	Resolution (LSB)	R0 (k)
10	0.1	0.3 (3 x cycle, Sample & hold bit is enabled)	7.8	3.0	0.1	3.0
					0.3	4.5
					0.5	5.3
					0.7	5.9
					0.9	6.4
					1.1	6.8
					1.3	7.2
					1.5	7.5
					1.7	7.8
					1.9	8.1
10	0.1	0.2 (2 x cycle, Sample & hold bit is disabled)	7.8	3.0	0.3	0.4
					0.5	0.9
					0.7	1.3
					0.9	1.7
					1.1	2.0
					1.3	2.2
					1.5	2.4
					1.7	2.6
1.9	2.8					

Table 2.9.13. Relation between output impedance and precision (error) of A/D converter (8-bit mode) Reference value

f(Xin) (MHz)	Cycle ( $\mu$ s)	Sampling time ( $\mu$ s)	R (k)	C (pF)	Resolution (LSB)	R0 (k)
10	0.1	0.3 (3 x cycle, Sample & hold bit is enabled)	7.8	3.0	0.1	4.9
					0.3	7.0
					0.5	8.2
					0.7	9.1
					0.9	9.9
					1.1	10.5
					1.3	11.1
					1.5	11.7
					1.7	12.1
					1.9	12.6
10	0.1	0.2 (2 x cycle, Sample & hold bit is disabled)	7.8	3.0	0.1	0.7
					0.3	2.1
					0.5	2.9
					0.7	3.5
					0.9	4.0
					1.1	4.4
					1.3	4.8
					1.5	5.2
1.7	5.5					
1.9	5.8					



## 2.10 DMAC Usage

### 2.10.1 Overview of the DMAC usage

DMAC transfers one data item held in the source address to the destination address every time a transfer request is generated. The following is an overview of the DMAC usage.

#### (1) Source address and destination address

Both the register which indicates a source and the register which indicates a destination comprise of 24 bits, so that each can cover a 1M bytes space. After transfer of one bit of data is completed, the address in either the source register or the destination register can be incremented. However, both registers cannot be incremented. The links between the source and destination are as follows:

- (a) A fixed address from an arbitrary 1M bytes space
- (b) An arbitrary 1M bytes space from a fixed address
- (c) A fixed address from another fixed address  
([0020<sub>16</sub> to 003F<sub>16</sub> and 0180<sub>16</sub> to 019F<sub>16</sub> ] cannot be accessed)

#### (2) The number of bits of data transferred

The number of bit of data indicated by the transfer counter is transferred. If a 16-bit transfer is selected, up to 128K bytes can be transferred. If an 8-bit transfer is selected, up to 64K bytes can be transferred. The transfer counter is decremented each time one bit of data is transferred, and a DMA interrupt request occurs when the transfer counter underflows.

#### (3) DMA transfer factor

The DMA transfer factor can be selected from the following 31 factors: falling edge/two edges of  $\overline{\text{INT0}}$ / $\overline{\text{INT1}}$ / $\overline{\text{INT2}}$  pin, timer A0 interrupt request through timer A4 interrupt request, UART0 transmission/NACK/SS interface 0 transmission interrupt request, UART0 reception/ACK/SS interface 0 reception interrupt request, UART1 transmission/NACK/SS interface 1 transmission interrupt request, UART1 reception/ACK/SS interface 1 reception interrupt request, UART2 transmission/NACK interrupt request, UART2 reception/ACK interrupt request, UART3 transmission/NACK interrupt request, UART3 reception/ACK interrupt request, USB0/USB1/USB2/USB3 function interrupt request, A/D conversion interrupt request, software trigger, and DMA trigger.

Software trigger is always enabled. When software trigger is selected, DMA transfer is generated by writing "1" to software DMA interrupt request bit. When other factor is selected, DMA transfer is generated by generating corresponding interrupt request.

#### (4) Channel priority

High to low priority: DMA0, DMA1, DMA2, DMA3

#### (5) Writing to a register

When writing to the source register or the destination register with DMA enabled, the content of the register with a fixed address will change at the time of writing. Therefore, the user should not write to a register with a fixed address when the DMA enable bit is set to "1". The contents of the register with 'forward direction' selected, and the transfer counter, are changed when reloaded. A reload occurs either when the transfer counter underflows, or when the DMA enable bit is re-enabled, after having been disabled.

The reload register can be written to, as in normal conditions.

**(6) Reading to a register**

The reload register can be read to, as in normal conditions.

**(7) Switching function****(a) Switching between one-shot transfer and repeated transfer**

'One-shot transfer' refers to a mode in which DMA is disabled after the transfer counter underflows.

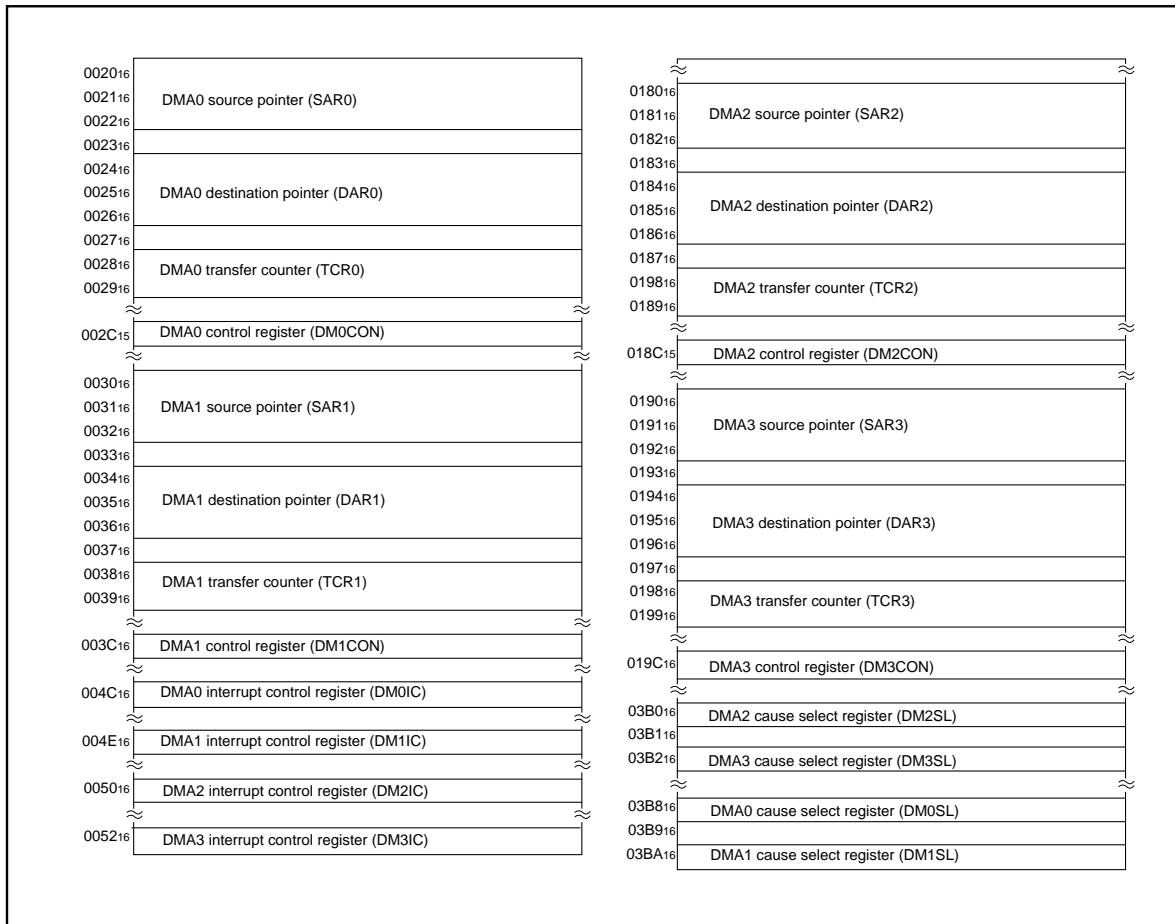
'Repeated transfer' refers to a mode in which a reload is carried out after the transfer counter underflows. The reload is carried out for the transfer counter and on the address pointer subjected to forward direction.

The following examples are described in section 2.10.2 and 2.10.3.

- A fixed address from an arbitrary 1M byte space, one-shot transfer
- An arbitrary 1M byte space from a fixed address, repeated transfer

**(8) Registers related to DMAC**

Figure 2.10.1 shows the memory map of DMAC-related registers, and Figures 2.10.2 and 2.10.4 show DMAC-related registers.



**Figure 2.10.1. Memory map of DMAC-related registers**

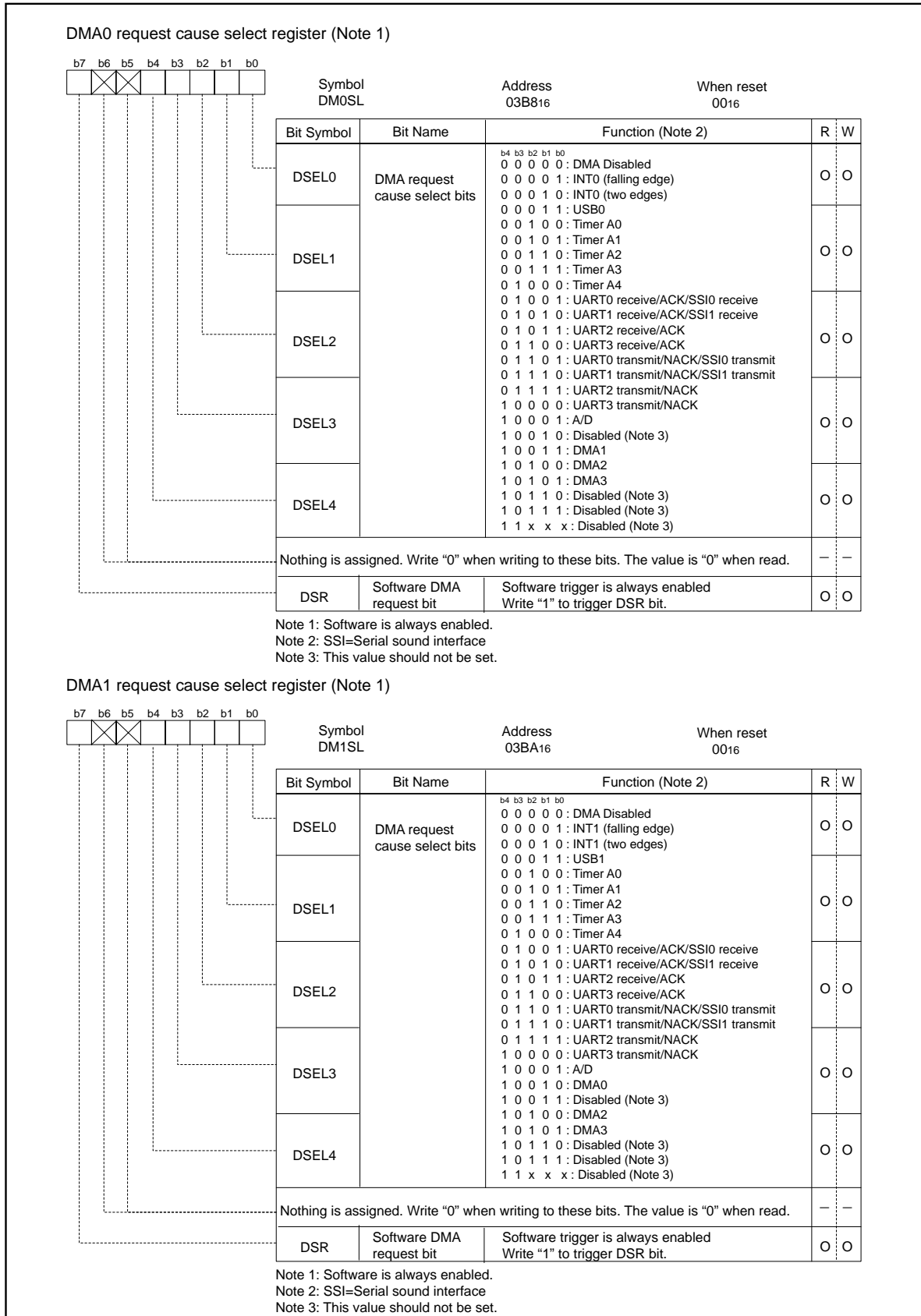


Figure 2.10.2. DMAC-related registers (1)

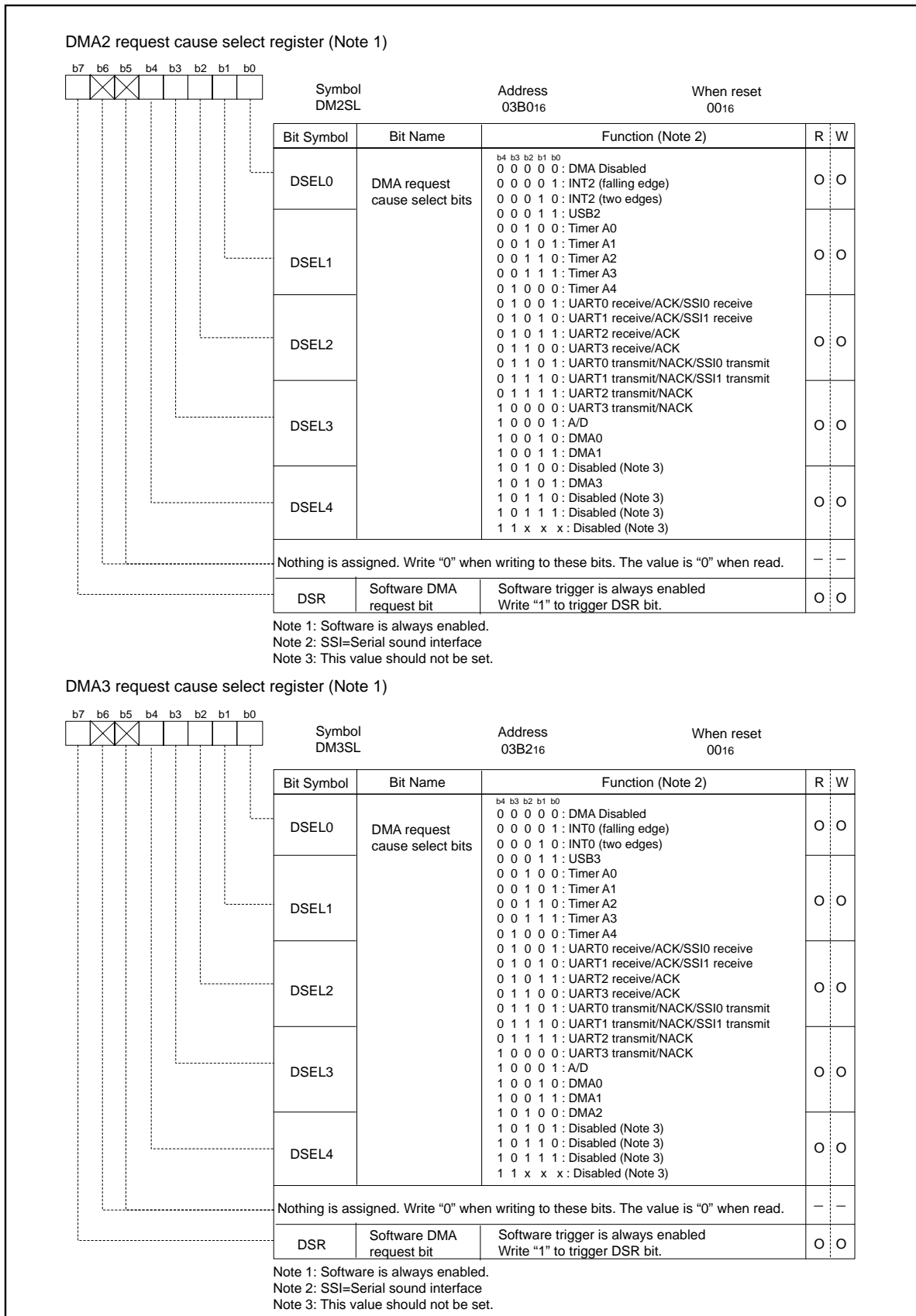


Figure 2.10.3. DMAC-related registers (2)

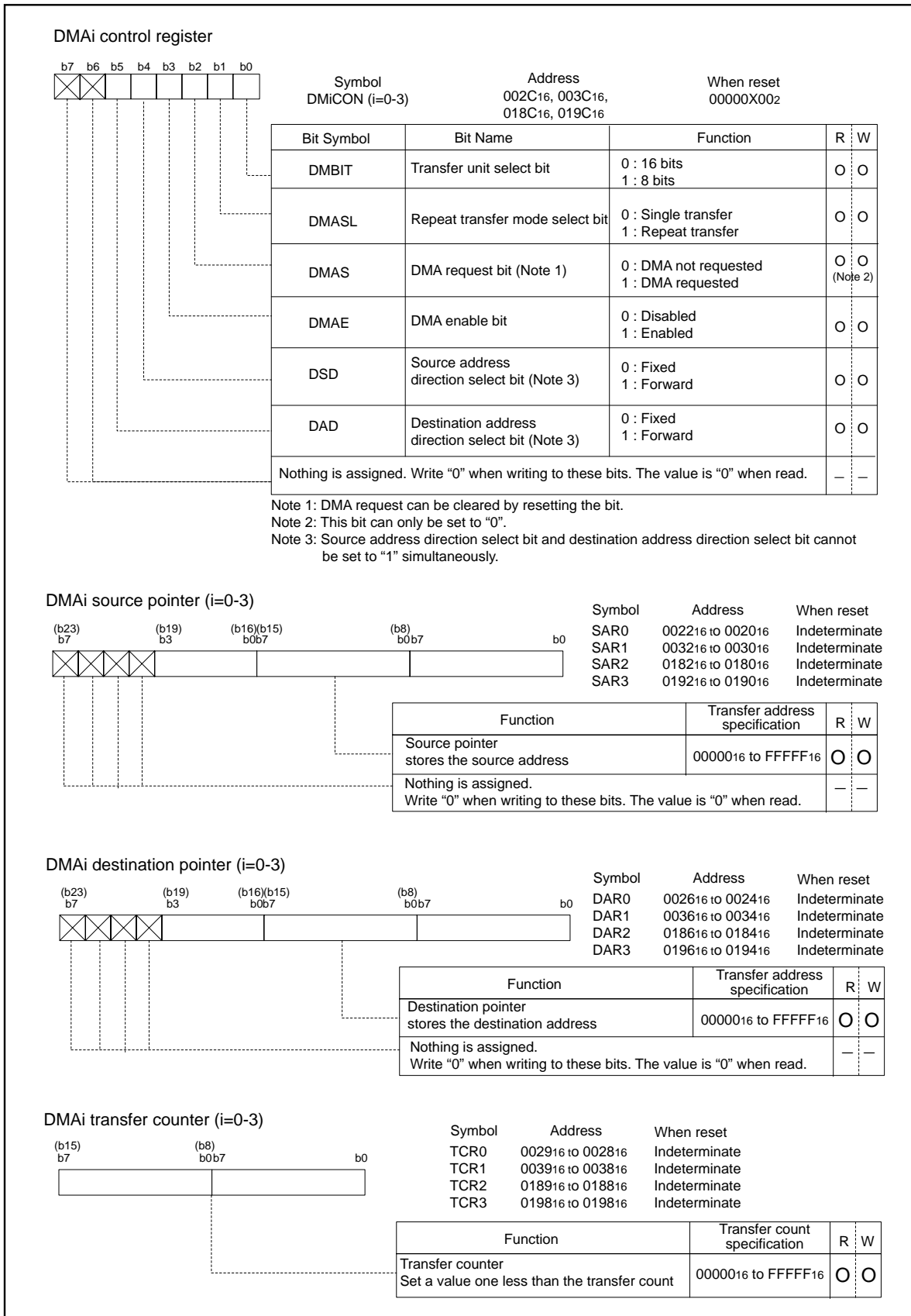


Figure 2.10.4. DMAC-related registers (3)

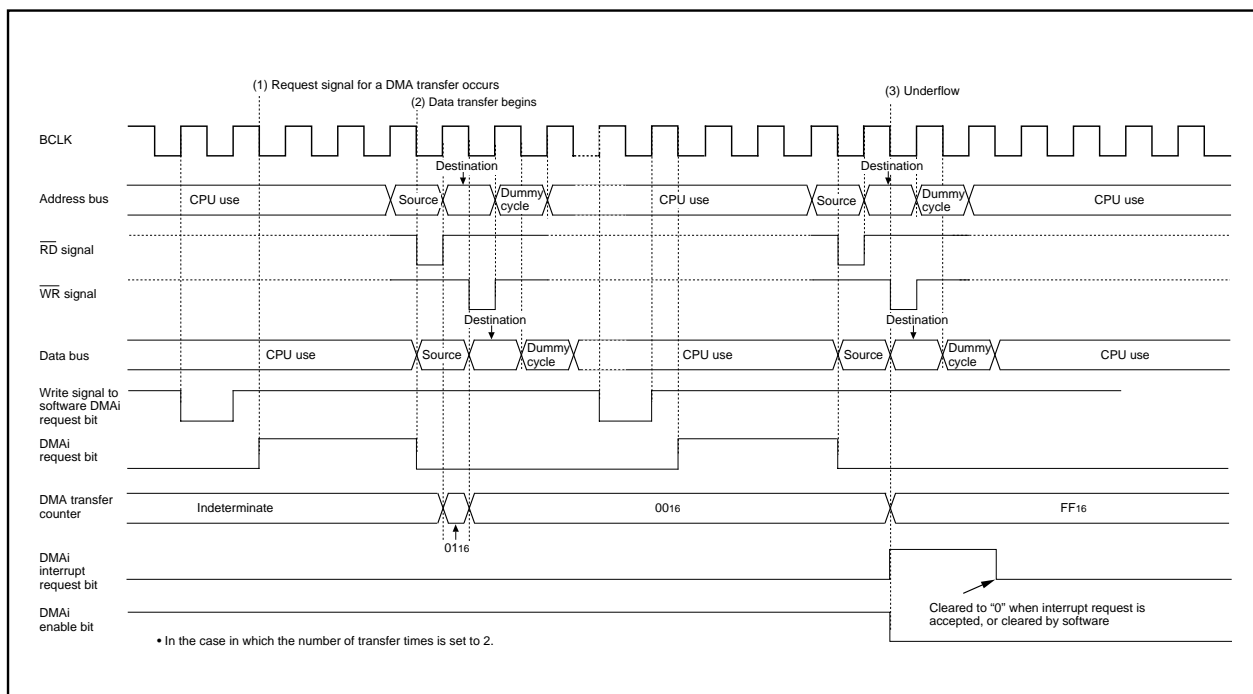
### 2.10.2 Operation of DMAC (one-shot transfer mode)

In one-shot transfer mode, choose functions from the items shown in Table 2.10.1. Operations of the circled items are described below. Figure 2.10.5 shows an example of operation and Figure 2.10.6 shows the set-up procedure.

**Table 2.10.1. Chosed functions**

Item	Set-up	
Transfer space	<b>○</b>	Fixed address from an arbitrary 1 M bytes space
		Arbitrary 1 M bytes space from a fixed address
		Fixed address from fixed address
Unit of transfer	<b>○</b>	8 bits
		16 bits

- Operation
- (1) When software trigger is selected, setting software DMA request bit to “1” generates a DMA transfer request signal.
  - (2) If DMAC is active, data transfer starts, and the contents of the address indicated by the DMAi forward-direction address pointer are transferred to the address indicated by the DMAi destination pointer. When data transfer starts directly after DMAC becomes active, the value of the DMAi transfer counter reload register is reloaded to the DMAi transfer counter, and the value of the DMAi source pointer is reloaded by the DMAi forward-direction address pointer. Each time a DMA transfer request signal is generated, 1 byte of data is transferred. The DMAi transfer counter is down counted, and the DMAi forward-direction address pointer is up counted.
  - (3) If the DMA transfer counter underflows, the DMA enable bit changes to “0” and DMA transfer is completed. The DMA interrupt request bit changes to “1” simultaneously.



**Figure 2.10.5. Example of operation of one-shot transfer mode**

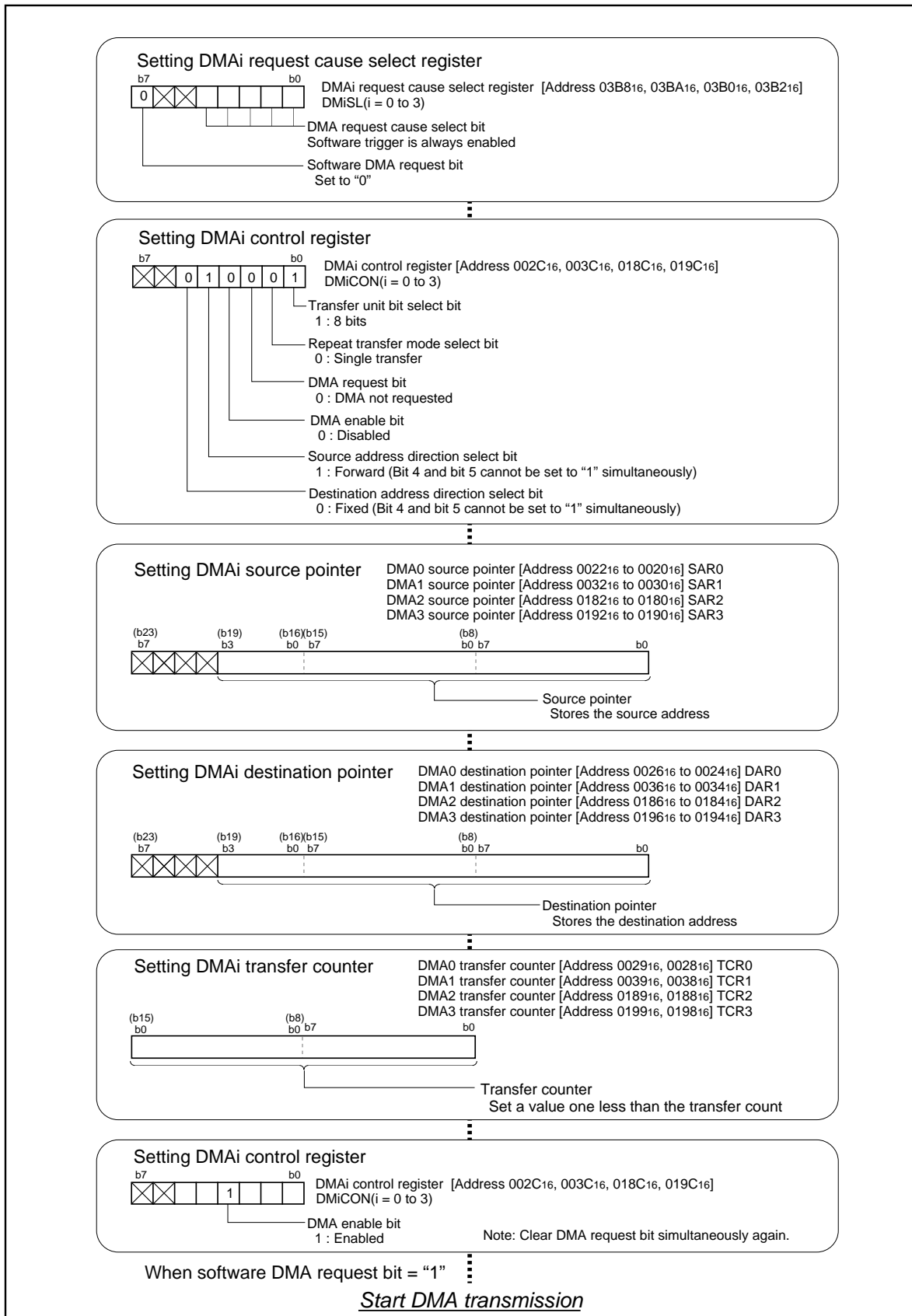


Figure 2.10.6. Set-up procedure of one-shot transfer mode

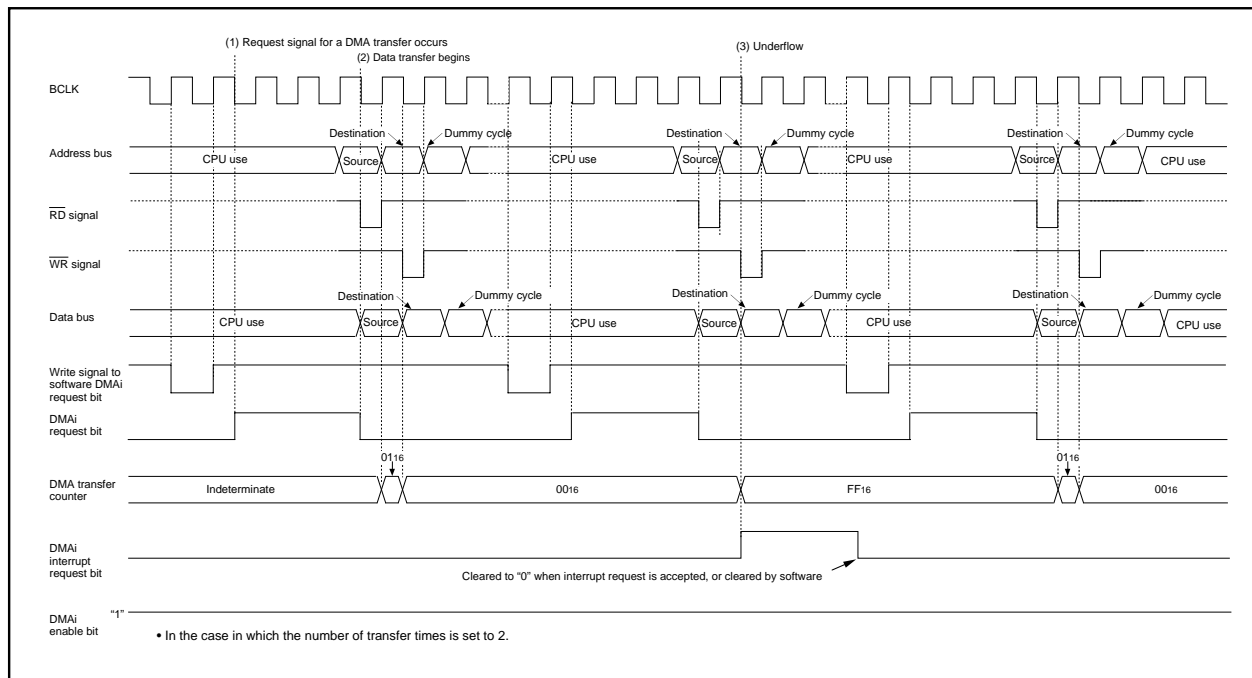
### 2.10.3 Operation of DMAC (repeated transfer mode)

In repeat transfer mode, choose functions from the items shown in Table 2.10.2. Operations of the circled items are described below. Figure 2.10.7 shows an example of operation and Figure 2.10.8 shows the set-up procedure.

**Table 2.10.2. Chosed functions**

Item	Set-up
Transfer space	Fixed address from an arbitrary 1 M bytes space
	<input type="radio"/> Arbitrary 1 M bytes space from a fixed address
	Fixed address from fixed address
Unit of transfer	8 bits
	<input type="radio"/> 16 bits

- Operation
- (1) When software trigger is selected, setting software DMA request bit to "1" generates a DMA transfer request signal.
  - (2) If DMAC is active, data transfer starts, and the contents of the address indicated by the DMAi forward-direction address pointer are transferred to the address indicated by the DMAi destination pointer. When data transfer starts directly after DMAC becomes active, the value of the DMAi transfer counter reload register is reloaded to the DMAi transfer counter, and the value of the DMAi source pointer is reloaded by the DMAi forward-direction address pointer. Each time a DMA transfer request signal is generated, 2 byte of data is transferred. The DMAi transfer counter is down counted, and the DMAi forward-direction address pointer is up counted.
  - (3) Though DMAi transfer counter is underflowed, DMA enable bit is still "1". The DMA interrupt request bit changes to "1" simultaneously.
  - (4) After DMAi transfer counter is underflowed, when the next DMA request is generated, DMA transfer is repeated from (1).



**Figure 2.10.7. Example of operation of repeated transfer mode**



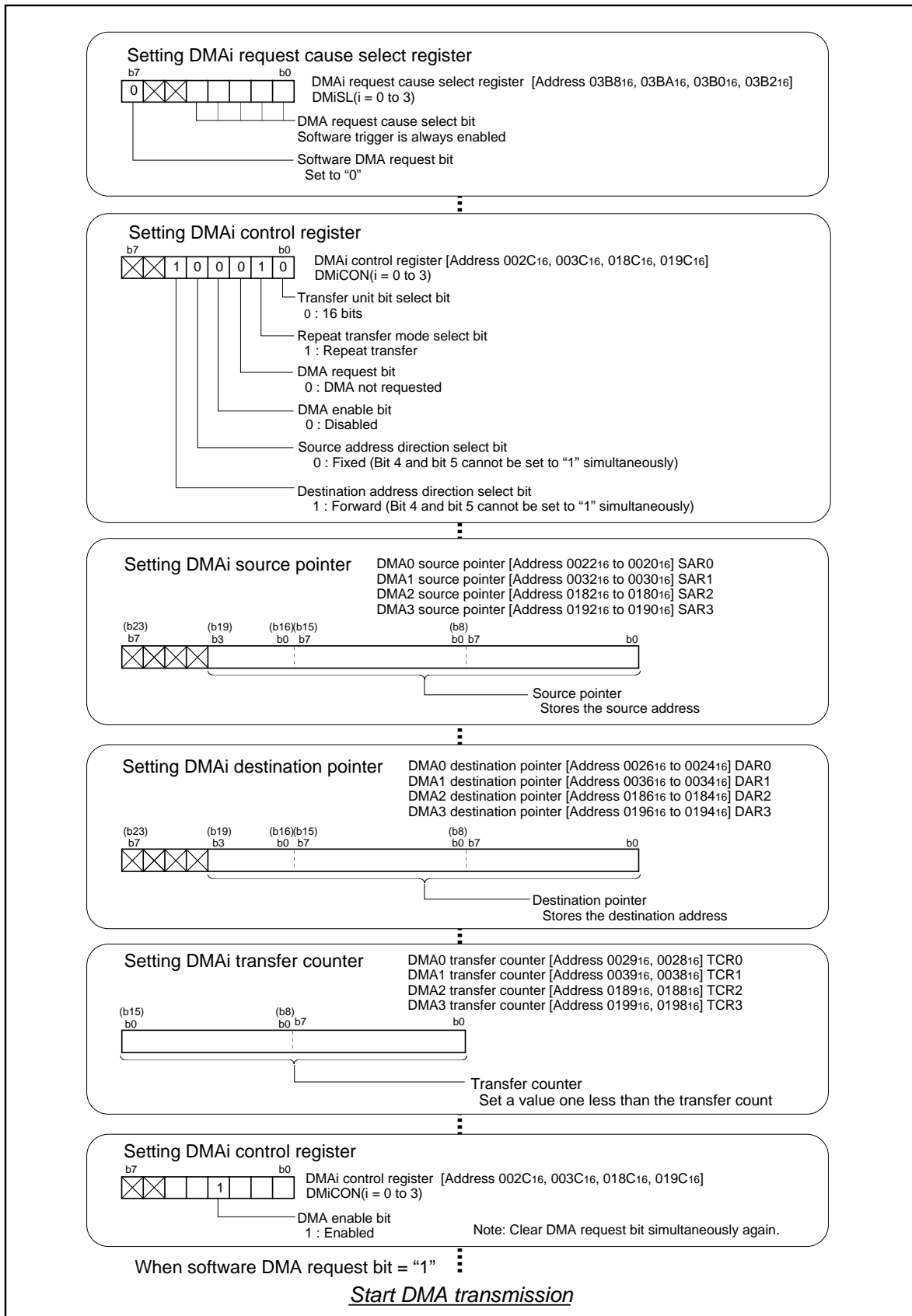


Figure 2.10.8. Set-up procedure of repeated transfer mode

## 2.11 CRC Calculation Circuit

### 2.11.1 Overview

Cyclic Redundancy Check (CRC) is a method that compares CRC code formed from transmission data by use of a polynomial generation with CRC check data so as to detect errors in transmission data. Using the CRC calculation circuit allows generation of CRC code. The microcomputer uses a generator polynomial of CRC\_CCITT ( $X^{16} + X^{12} + X^5 + 1$ ) or CRC-16 ( $X^{16} + X^{15} + X^2 + 1$ ) to generate CRC code.

And, the CRC circuit includes the ability to snoop reads and writes to SFR addresses. This can be used to accumulate the CRC value on a stream of data without using extra bandwidth to explicitly write data into the CRCIN register.

#### (1) Registers related to CRC calculation circuit

Figure 2.11.1 shows the memory map of CRC-related registers, and Figure 2.11.2 shows CRC-related registers.

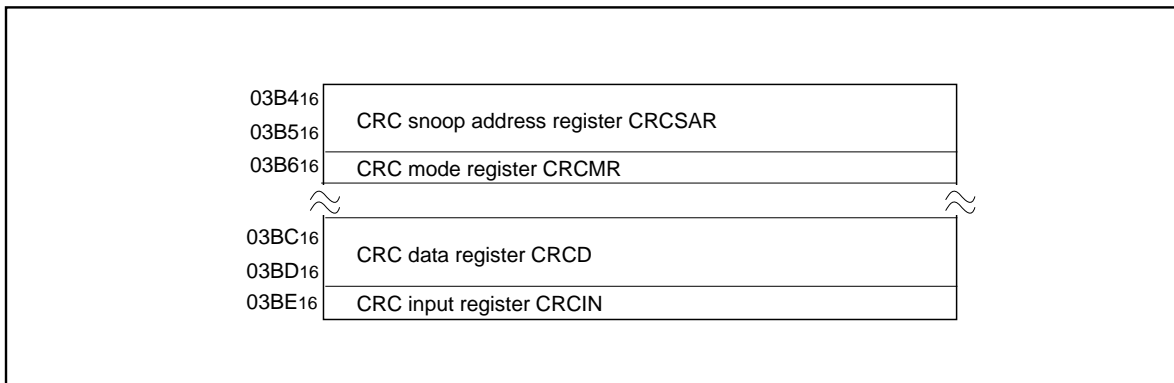


Figure 2.11.1. Memory map of CRC-related registers

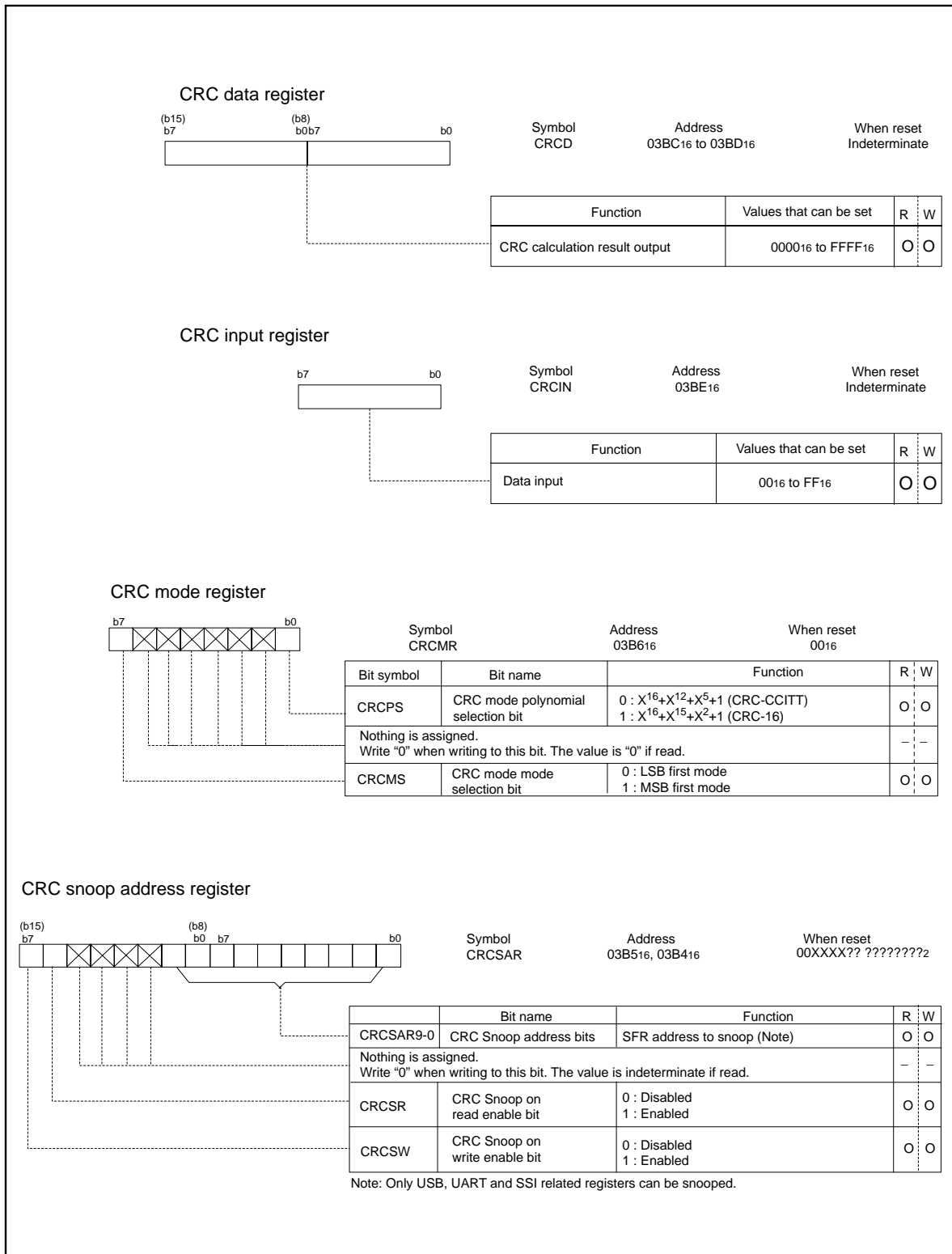


Figure 2.11.2. CRC-related registers

### 2.11.2 Operation of CRC Calculation Circuit

The following describes the operation of the CRC calculation. Figure 2.11.3 shows an example of calculation using the CRC-CCITT.

- Operation
- (1) Select CRC-CCITT or CRC-16, and LSB first or MSB first by bit 0 and bit 5 of CRC mode register.
  - (2) The CRC calculation circuit sets an initial value in the CRC data register.
  - (3) Writing 1 byte data to the CRC input register generates CRC code based on the data register. CRC code generation for 1 byte data finishes in two machine cycles.
  - (4) When several bytes of CRC calculation is performed in succession, write the following data in the CRC input register continuously.
  - (5) The content of CRC data register after all data is written becomes CRC code.

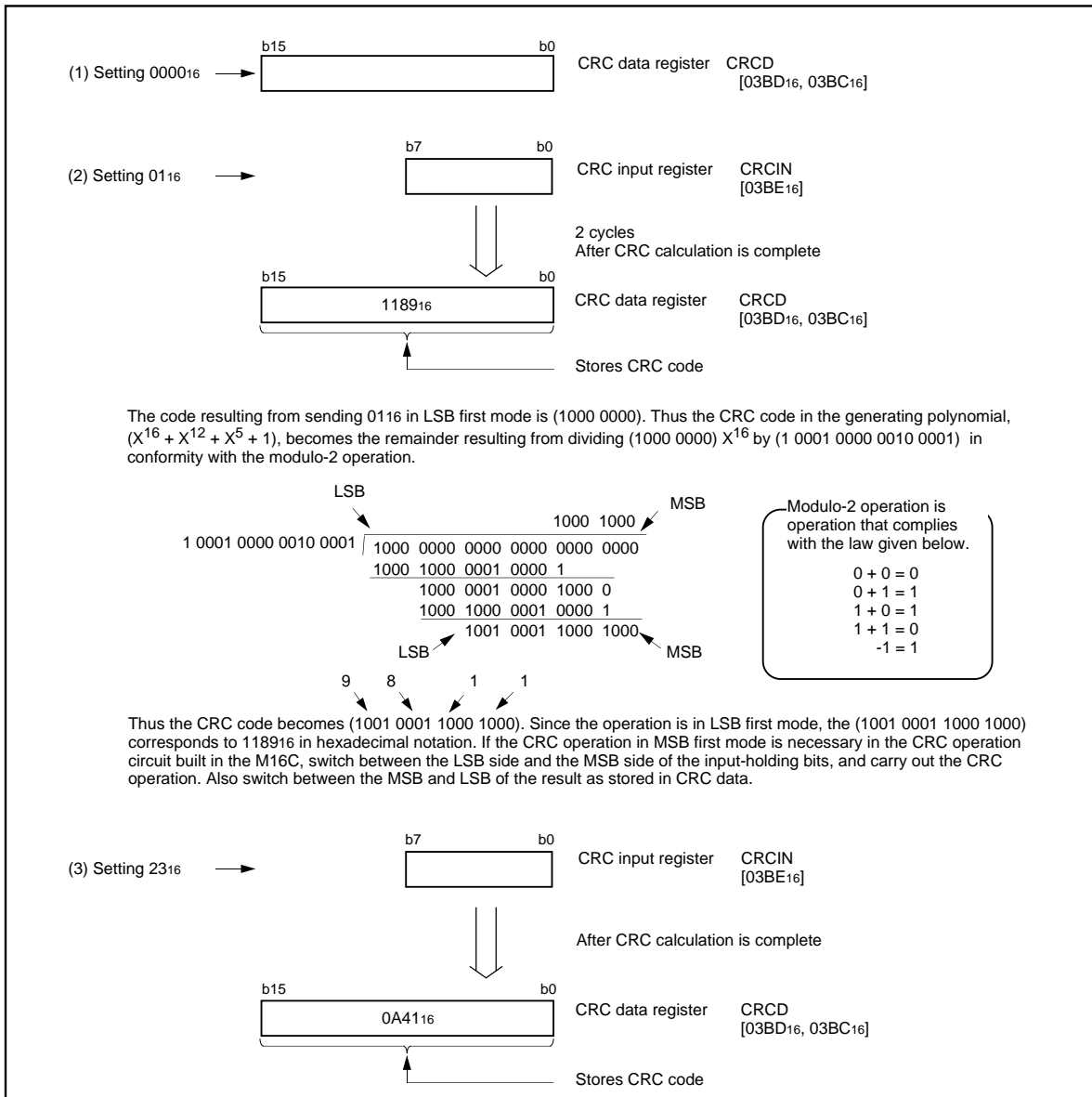


Figure 2.11.3. Calculation example using the CRC calculation circuit (when using CRC-CCITT)

### 2.11.3 SFR Access Snoop Function

The CRC calculation circuit includes the ability to snoop write/read to/from the SFR addresses and to execute CRC automatic calculation (SFR access snoop function). In order to execute CRC calculation for data which have been written/read to/from the SFR, setting data to CRC input register again is not required. The target SFRs include USB-related registers, UART-related registers, and Serial Sound Interface-related registers.

- Operation
- (1) The bit 1 of CRC mode register selects either CRC-CCITT or CRC-16, and the bit 7 selects either LSB first or MSB first.
  - (2) The target SFR addresses are set to CRC snoop address register (bit 0 to 9 of CRCSAR). Snooping of writing to the target SFR is enabled with CRC snoop on write enable bit (bit 15 of CRCSAR), and snooping of reading from the target SFR is enabled with CRC snoop on read enable bit (bit 14 of CRCSAR).
  - (3) The initial value  $0000_{16}$  is set to CRC data register.
  - (4) If writing into the target SFR is executed by either the CPU or DMA while "1" is set to CRC snoop on write enable bit, the CRC calculation circuit will store the data written to the target SFR in CRC input register, and executes CRC calculation. Similarly, if reading from the target SFR by the CPU or DMA while "1" is set to CRC snoop on read enable bit, calculation circuit will store the data read from the target SFR in CRC input register, and executes CRC calculation. The CRC calculation circuit can only calculate CRC codes on data 1-byte at a time. Therefore, if a target SFR is accessed in a word (16-bit) bus cycle, only 1-byte data are stored into CRC input register.
  - (5) When 1-byte data is stored in CRC input register, CRC codes are generated in CRC data register based on the stored data and the content of CRC data register. Generation of CRC codes for 1-byte of data is completed in two machine cycles.

## 2.12 Watchdog Timer

### 2.12.1 Overview

The watchdog timer can detect a runaway program using its 15-bit timer prescaler. The following is an overview of the watchdog timer.

#### (1) Watchdog timer start procedure

When reset, the watchdog timer is in stopped state. Writing to the watchdog timer start register initializes the watchdog timer to 7FFF<sub>16</sub> and causes it to start performing a down count. The watchdog timer, once started operating, cannot be stopped by any means other than stopping conditions.

#### (2) Watchdog timer stop conditions

The watchdog timer stops in any one of the following states:

- (a) Period in which the CPU is in stopped state
- (b) Period in which the CPU is in waiting state
- (c) Period in which the microcomputer is in hold state

#### (3) Watchdog timer initialization

The watchdog timer is initialized to 7FFF<sub>16</sub> in the cases given below, and begins a down count.

- (a) When the watchdog timer writes to the watchdog timer start register while a count is in progress
- (b) When the watchdog timer underflows

#### (4) Runaway detection

When the watchdog timer underflows, either a watchdog timer interrupt occurs or reset is selected depending on the setting of the watchdog timer function select bit. In writing a program, write to the watchdog timer start register before the watchdog timer underflows.

The watchdog timer interrupt occurs regardless of the status of the interrupt enable flag (I flag). In processing a watchdog timer interrupt, set the software reset bit to "1" to reset software.

**(5) Watchdog timer cycle**

The watchdog timer cycle varies depending on the BCLK and the frequency division ratio of the prescaler selected.

Table 2.12.1 shows the watchdog timer cycle.

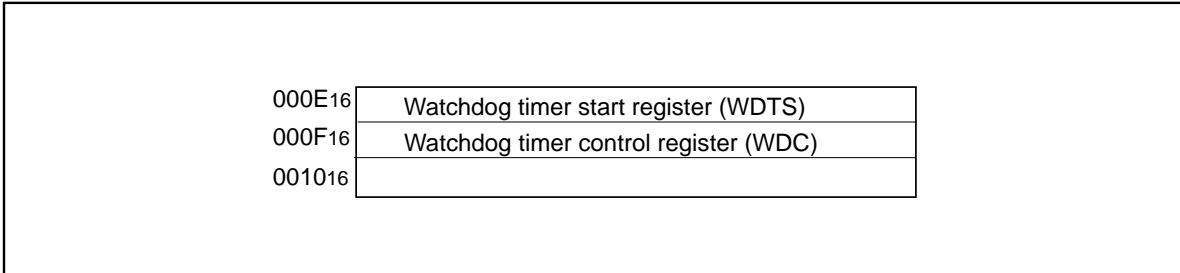
**Table 2.12.1. The watchdog timer cycle ( $f(X_{IN}) = 16\text{MHz}$ )**

CM07	CM06	CM17	CM16	BCLK	WDC7	Period
0	0	0	0	16MHz	0	Approx. 32.8ms (Note)
					1	Approx. 262.1ms (Note)
0	0	0	1	8MHz	0	Approx. 65.5ms (Note)
					1	Approx. 524.3ms (Note)
0	0	1	0	4MHz	0	Approx. 131.1ms (Note)
					1	Approx. 1.049s (Note)
0	0	1	1	1MHz	0	Approx. 524.3ms (Note)
					1	Approx. 4.194s (Note)
0	1	Invalid	Invalid	2MHz	0	Approx. 262.1ms (Note)
					1	Approx. 2.097s (Note)
1	Invalid	Invalid	Invalid	32kHz	Invalid	Approx. 2s (Note)

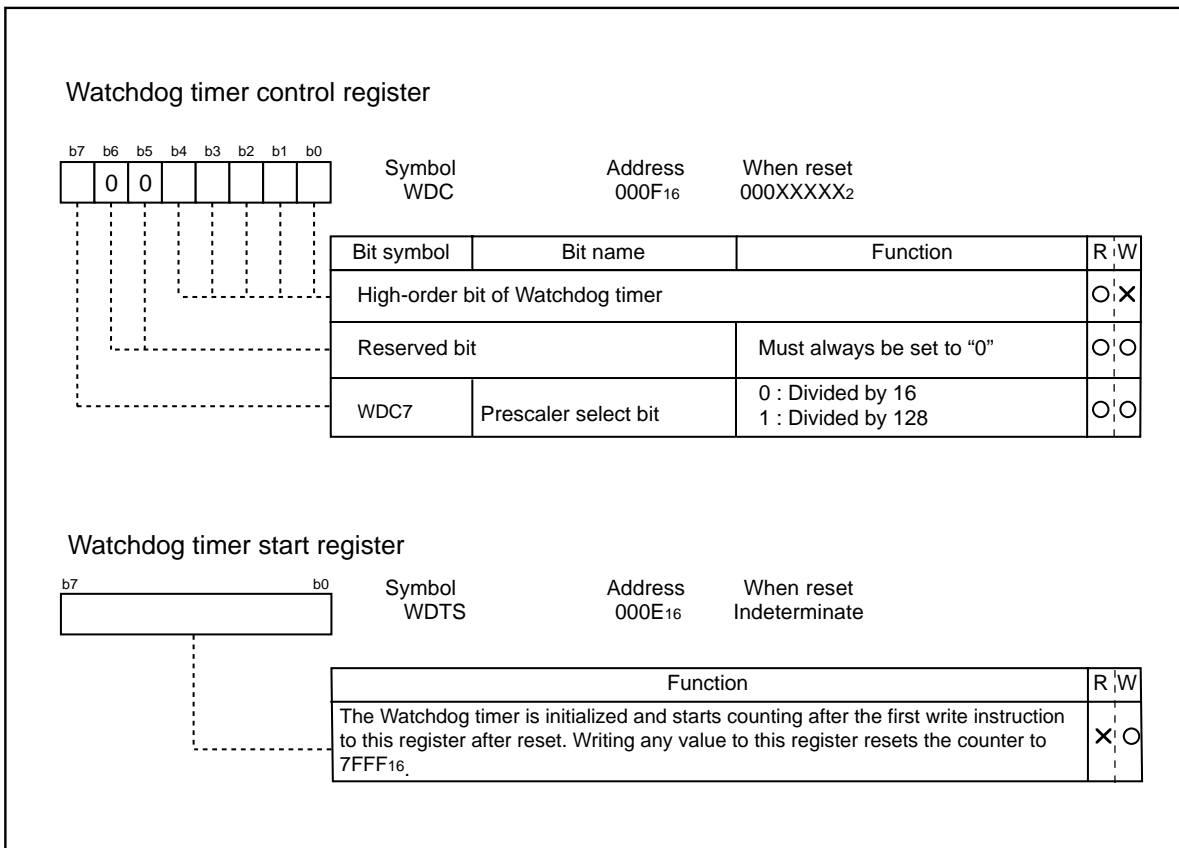
Note: An error due to the prescaler occurs.

**(6) Registers related to the watchdog timer**

Figure 2.12.1 shows the memory map of watchdog timer-related registers, and Figure 2.12.2 shows watchdog timer-related registers.



**Figure 2.12.1. Memory map of watchdog timer-related registers**



**Figure 2.12.2. Watchdog timer-related registers**



### 2.12.2 Operation of Watchdog Timer (Watchdog timer interrupt)

The following is an operation of the watchdog timer using watchdog timer interrupt. Figure 2.12.3 shows the operation timing, and Figure 2.12.4 shows the set-up procedure.

- Operation
- (1) Writing to the watchdog timer start register initializes the watchdog timer to 7FFF<sub>16</sub> and causes it to start a down count.
  - (2) With a count in progress, writing to the watchdog timer start register again initializes the watchdog timer to 7FFF<sub>16</sub> and causes it to resume counting.
  - (3) Either executing the WAIT instruction or going to the stopped state causes the watchdog timer to hold the count in progress and to stop counting. The watchdog timer resumes counting after returning from the execution of the WAIT instruction or from the stopped state.
  - (4) If the watchdog timer underflows, it is initialized to 7FFF<sub>16</sub> and continues counting. At this time, a watchdog timer interrupt occurs.

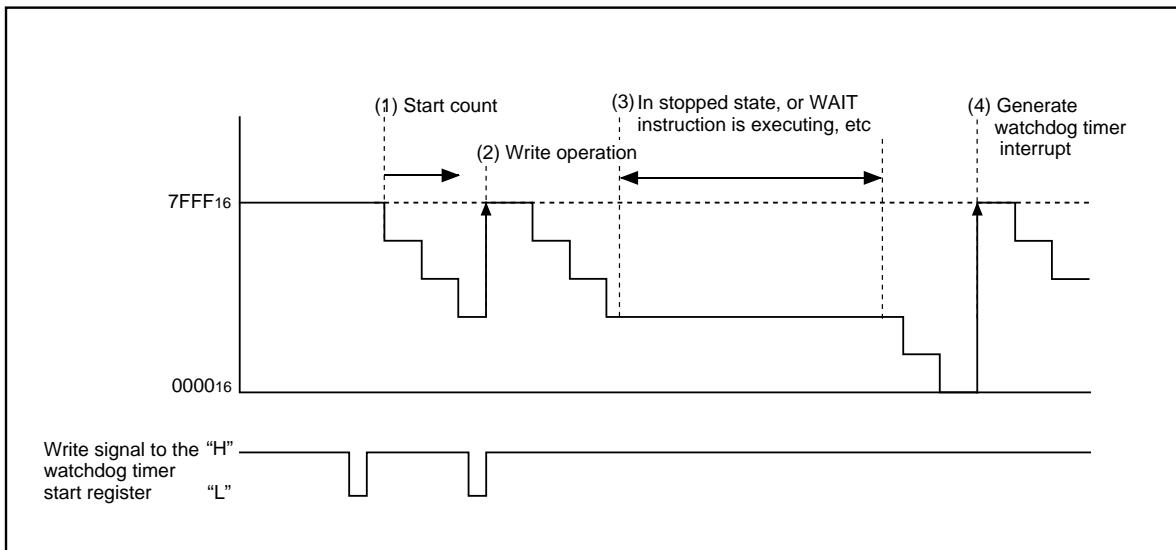
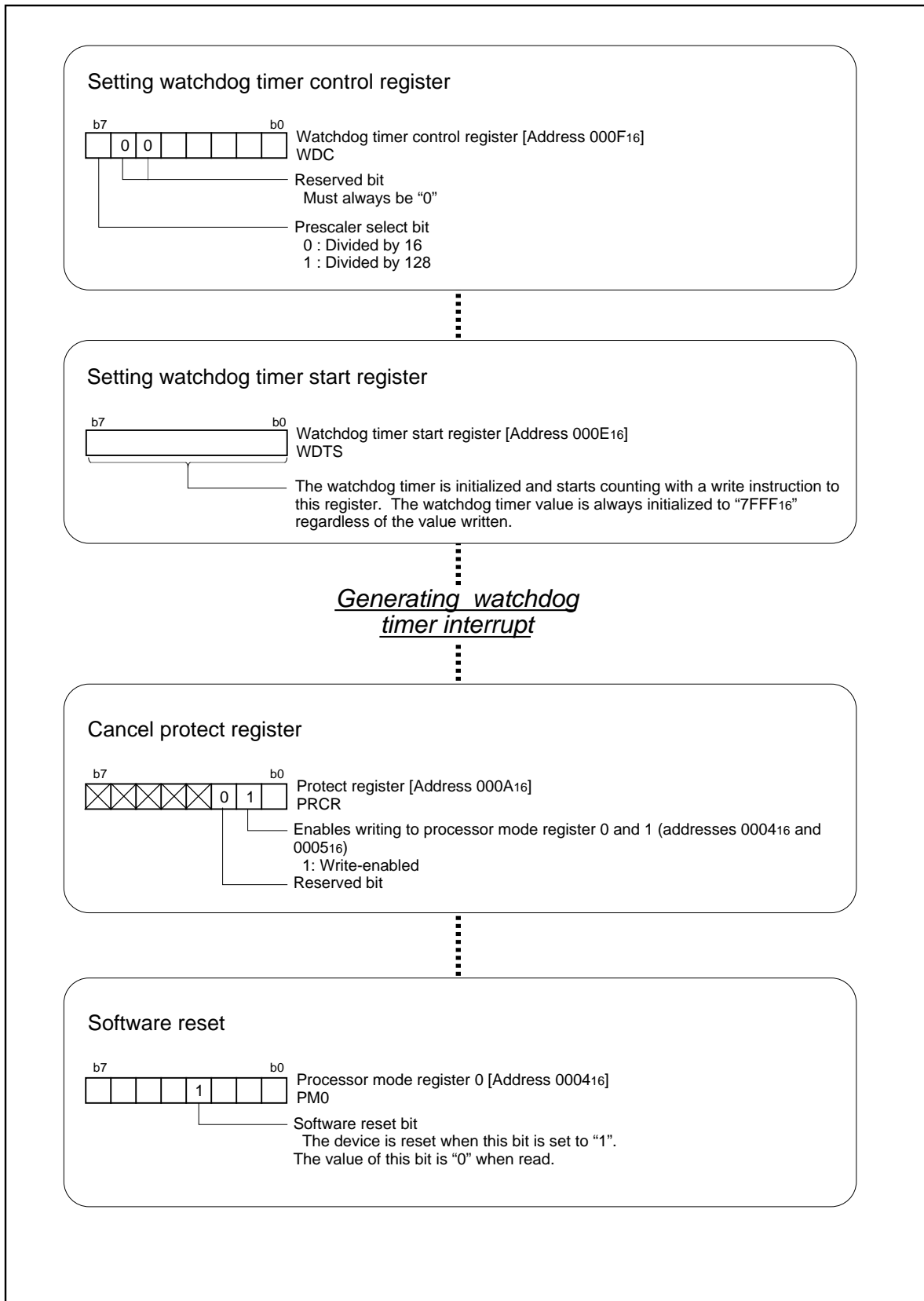


Figure 2.12.3. Operation timing of watchdog timer (watchdog timer interrupt)



**Figure 2.12.4. Set-up procedure of watchdog timer interrupt (watchdog timer interrupt)**

## 2.13 Address Match Interrupt Usage

### 2.13.1 Overview of the address match interrupt usage

The address match interrupt is used for correcting a ROM or for a simplified debugging-purpose monitor. When the external bus is used for 8 bits, the address match interrupt is not used to the external areas. The following is an overview of the address match interrupt usage.

#### (1) Enabling/disabling the address match interrupt

The address match interrupt enable bit can be used to enable and disable an address match interrupt. It is affected neither by the processor interrupt priority level (IPL) nor the interrupt enable flag (I flag).

#### (2) Timing of the address match interrupt

An interrupt occurs immediately before executing the instruction in the address indicated by the address match interrupt register. Set the first address of the instruction in the address match interrupt register. Setting a half address of an instruction or an address of tabulated data does not generate an address match interrupt.

The first instruction of an interrupt routine does not generate an address match interrupt either.

#### (3) Returning from an address match interrupt

The address put in the stack when an address match interrupt occurs depends on the instruction not yet executed (the instruction the address match interrupt register indicates). The return address is not put in the stack. For this reason, to return from an address match interrupt, either rewrite the content of the stack and use the REIT instruction or use the POP instruction to restore the stack to the state as it was before the interrupt occurred and return by use of a jump instruction.

<Instructions whose address is added to by 2 when an address match interrupt occurs>					
• 16-bit operation code instructions					
• 8-bit operation code instructions given below					
ADD.B:S	#IMM8,dest	SUB.B:S	#IMM8,dest	AND.B:S	#IMM8,dest
OR.B:S	#IMM8,dest	MOV.B:S	#IMM8,dest	STZ.B:S	#IMM8,dest
STNZ.B:S	#IMM8,dest	STZX.B:S	#IMM81,#IMM82,dest		
CMP.B:S	#IMM8,dest	PUSHM	src	POPM	dest
JMPS	#IMM8	JSRS	#IMM8		
MOV.B:S	#IMM,dest	(However, dest = A0/A1)			
<Instructions whose address is added to by 1 when an address match interrupt occurs>					
• Instructions other than those listed above					

**Figure 2.13.1. Unexecuted instructions and corresponding stacked addresses**

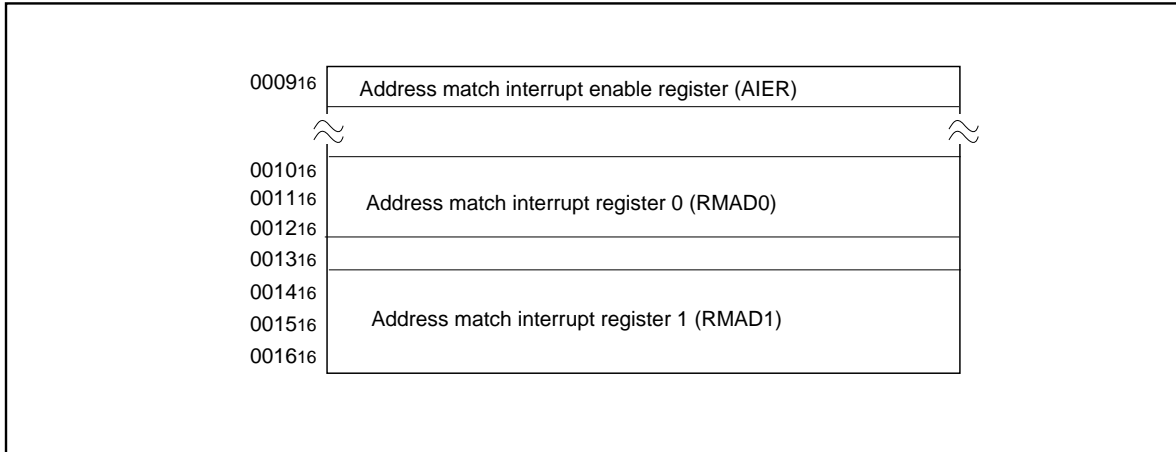
Figure 2.13.1 shows unexecuted instructions and corresponding the stacked addresses.

#### (4) How to determine an address match interrupt

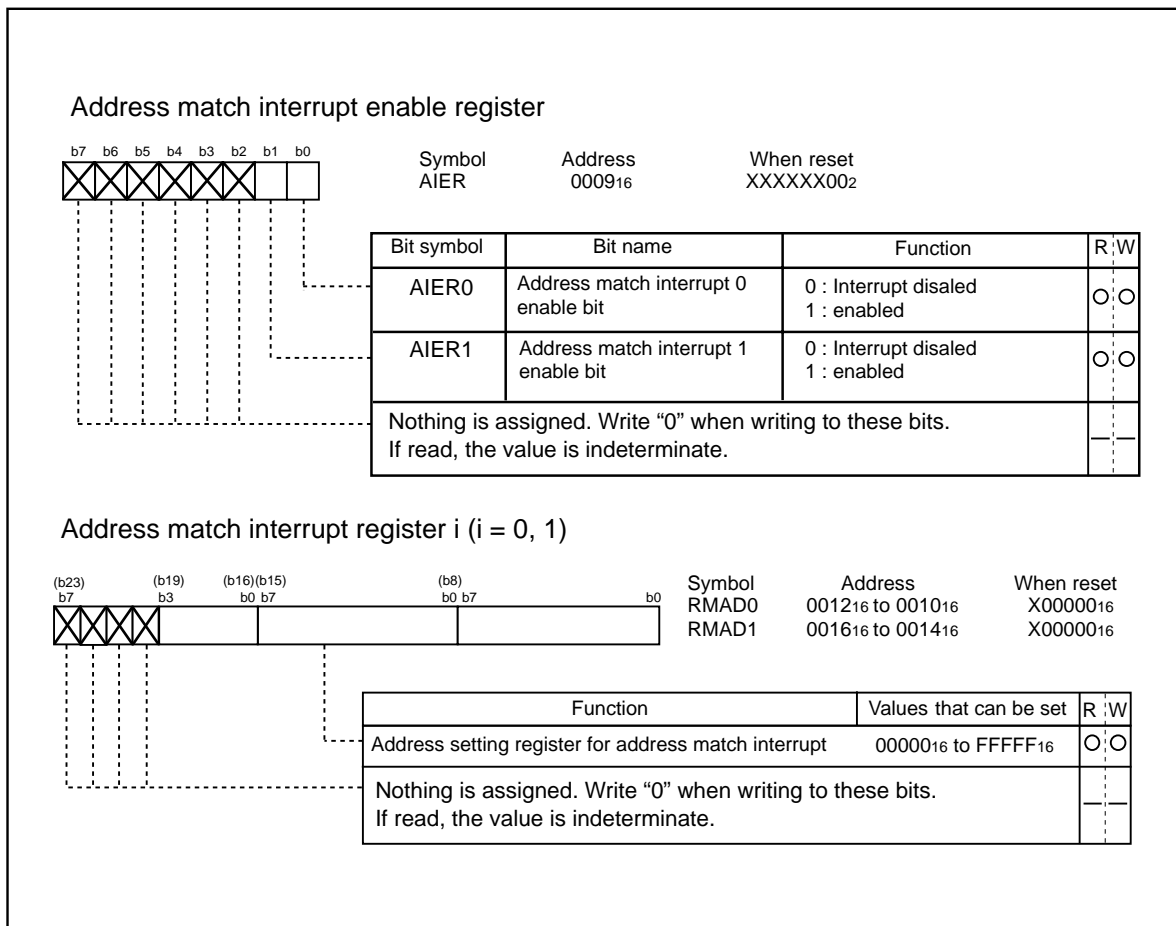
Address match interrupts can be set at two different locations. However, both location will have the same vector address. Therefore, it is necessary to determine which interrupt has occurred; address match interrupt 0 or address match interrupt 1. Using the content of the stack, etc., determine which interrupt has occurred according to the first part of the address match interrupt routine.

**(5) Registers related to the address match interrupt**

Figure 2.13.2 shows the memory map of address match interrupt-related registers, and Figure 2.13.3 shows address match interrupt-related registers.



**Figure 2.13.2. Memory map of address match interrupt-related registers**



**Figure 2.13.3. Address match interrupt-related registers**

### 2.13.2 Operation of Address Match Interrupt

The following is an operation of address match interrupt. Figure 2.13.4 shows the set-up procedure of address match interrupt, and Figure 2.13.5 shows the overview of the address match interrupt handling routine.

- Operation
- (1) The address match interrupt handling routine sets an address to be used to cause the address match interrupt register to generate an interrupt.
  - (2) Setting the address match enable flag to "1" enables an interrupt to occur.
  - (3) An address match interrupt occurs immediately before the instruction in the address indicated by the address match interrupt register as a program is executed.

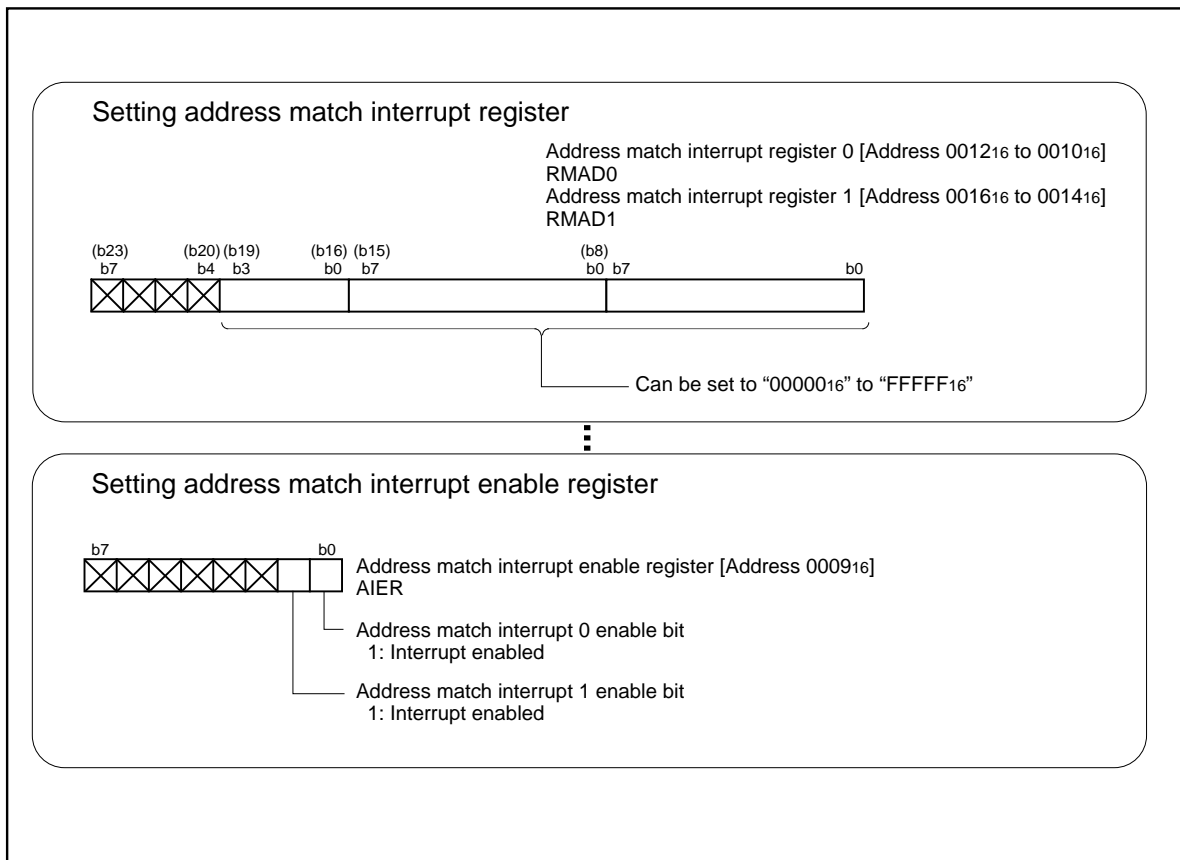


Figure 2.13.4. Set-up procedure of address match interrupt

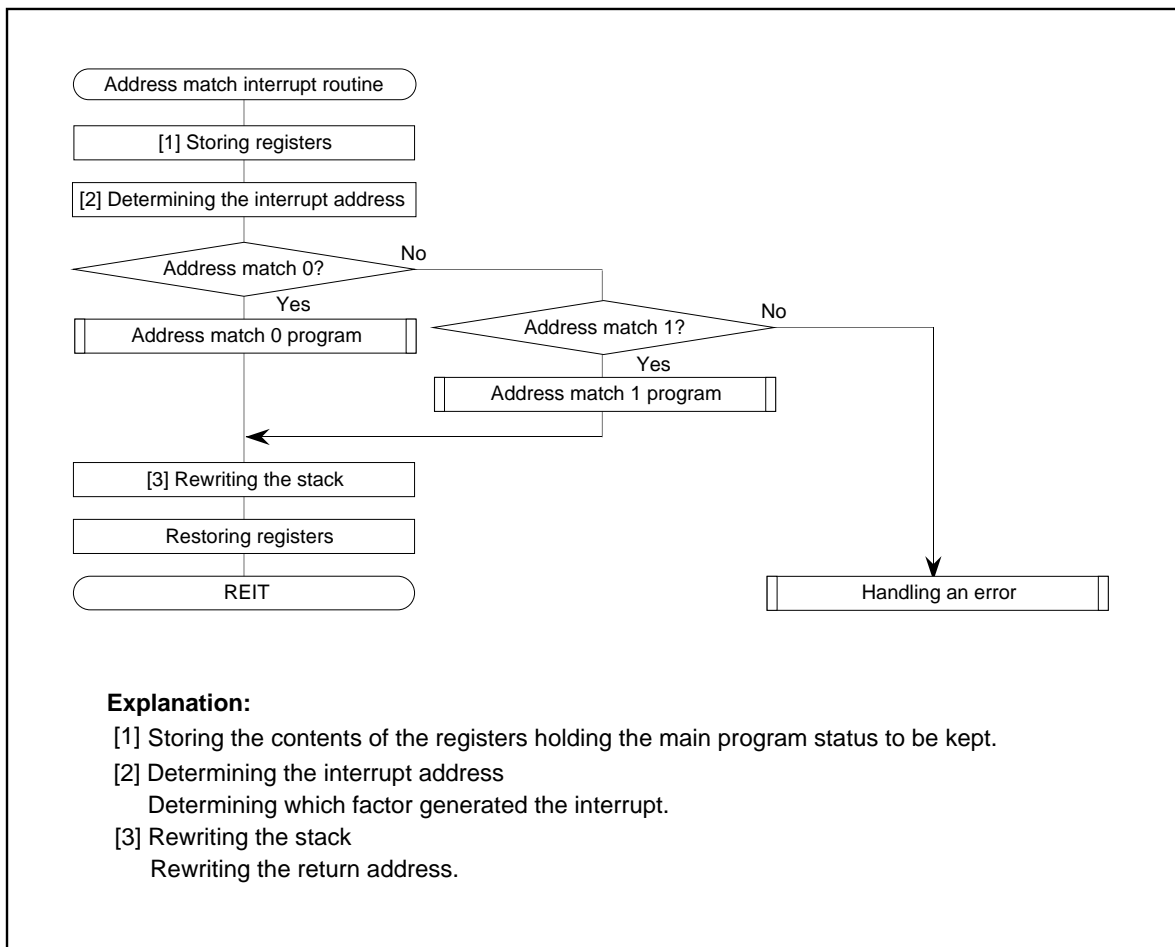


Figure 2.13.5. Overview of the address match interrupt handling routine

## 2.14 Key-Input Interrupt Usage

### 2.14.1 Overview of the key-input interrupt usage

Key-input interrupt can be generated by a falling edge, rising edge or both edges input to any Port 10 pin. It can also be used as a Key-on wake up function for canceling the wait mode or stop mode. It is possible to select the edge of the Key input interrupt for P10 with bits 0 and 1 of key input mode register. This register is also used to enable or disable Port 10 pins that are to be used for Key-input interrupts. Port 10 can be configured with pull-up resistors using the pull-up control resistor.

The following is an overview of the key-input interrupt usage:

#### (1) Enabling/disabling the key-input interrupt

The key-input interrupt can be enabled and disabled using the key-input mode register (03F9<sub>16</sub>) and the key-input interrupt register (0041<sub>16</sub>). The key-input interrupt is affected by the interrupt priority level (IPL) and the interrupt enable flag (I flag). A falling edge, rising edge or both edges input to any Port 10 pin can be selected by P10 Key-input edge select bits (bit0 and bit1 of 03F9<sub>16</sub>).

#### (2) Occurrence timing of the key-input interrupt

With key-input interrupt acceptance enabled, pins P10<sub>0</sub> through P10<sub>7</sub>, which are set to input, become key-input interrupt pins ( $\overline{KI0}$  through  $\overline{KI7}$ ). A Key-input interrupt occurs when the selected edge is input to a Key-input interrupt pin. At this moment, the level of other key-input interrupt pins must be "H". No interrupt occurs when the level of other key-input interrupt pins is "L".

#### (3) How to determine a key-input interrupt

A key-input interrupt occurs when the selected edge is input to one of eight pins, but each pin has the same vector address. Therefore, read the input level of Port P10 in the key-input interrupt routine to determine the interrupted pin.

#### (4) Registers related to the key-input interrupt

Figure 2.14.1 shows the memory map of key-input interrupt-related registers, and Figure 2.14.2 and 2.14.3 show key-input interrupt-related registers.

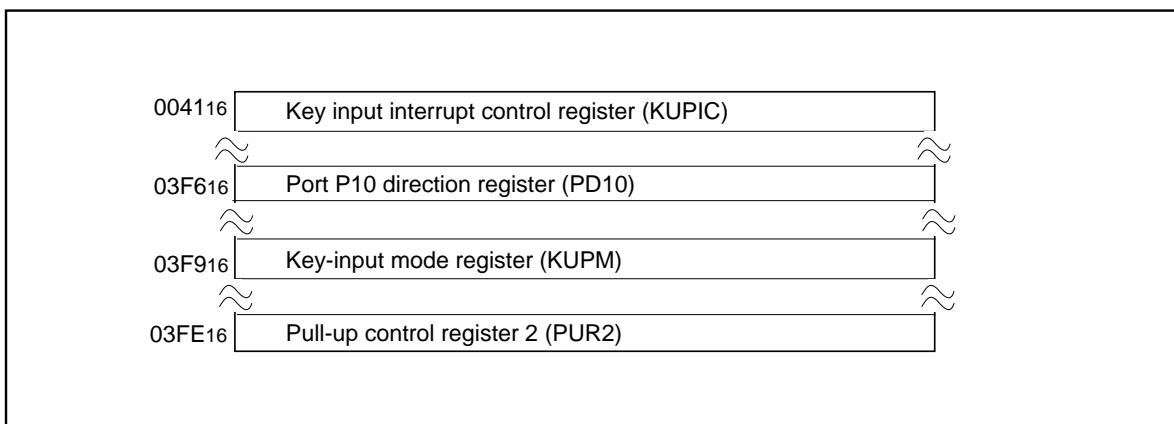


Figure 2.14.1. Memory map of key-input interrupt-related registers

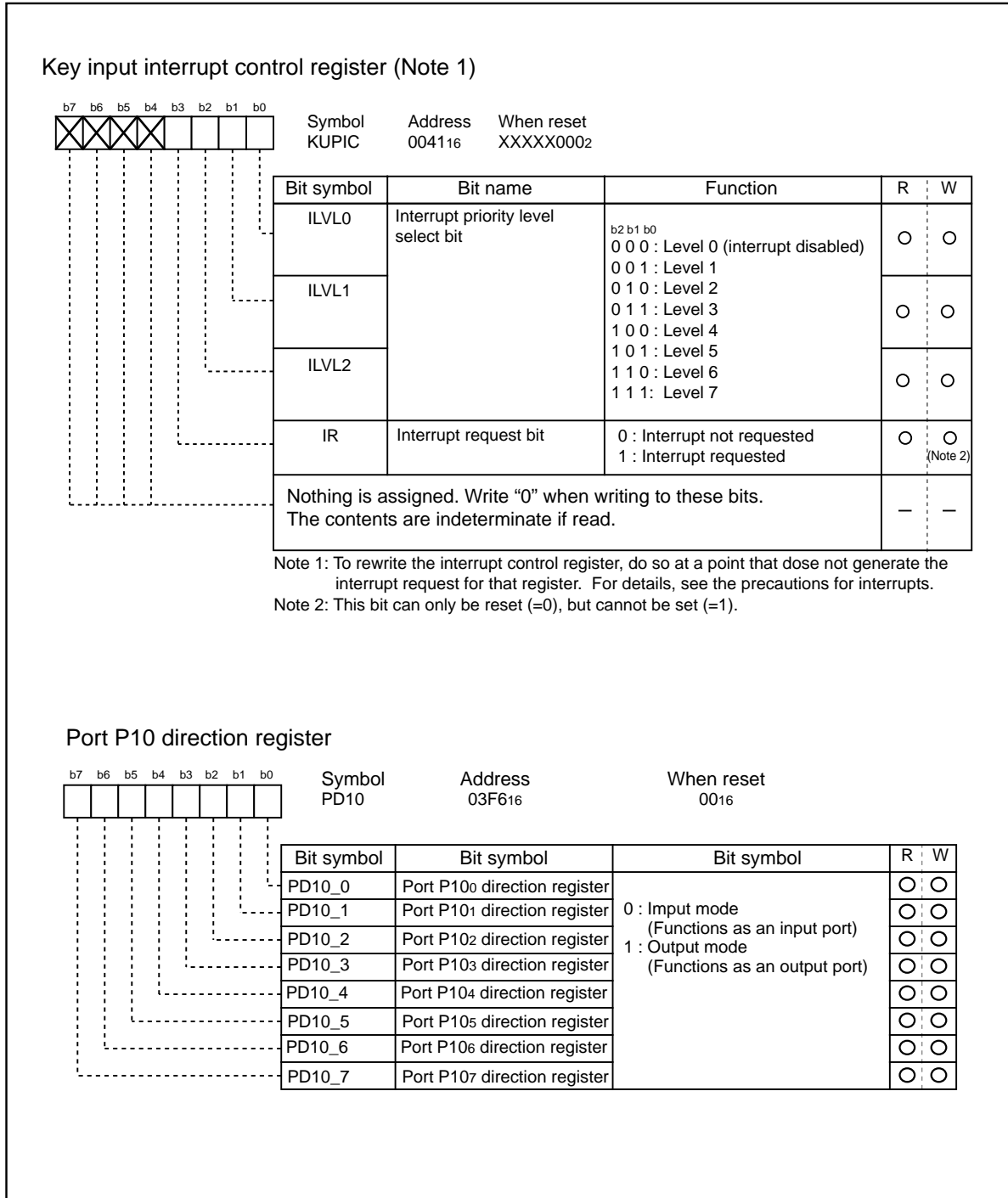


Figure 2.14.2. key-input interrupt-related registers (1)



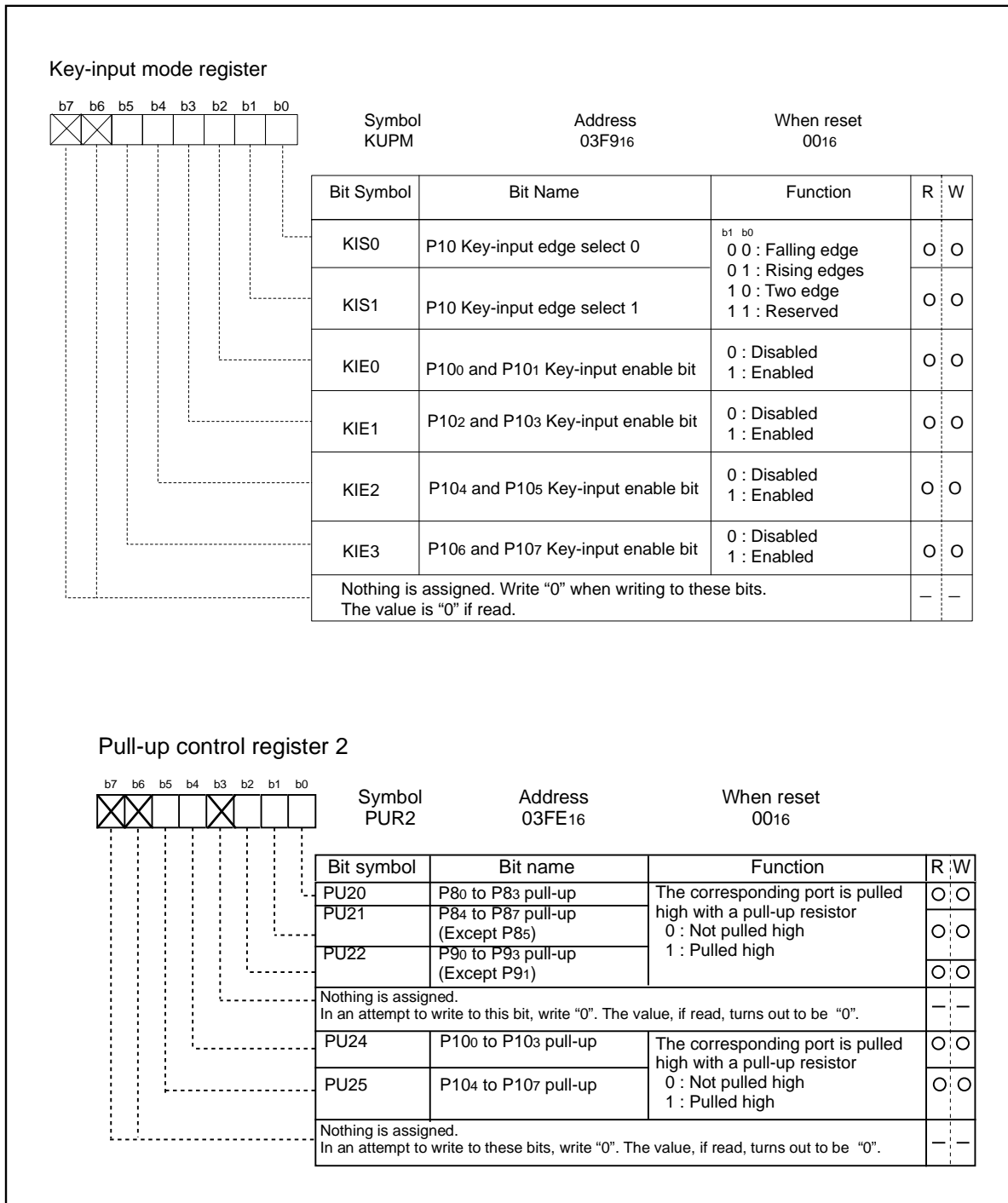


Figure 2.14.3. key-input interrupt-related registers (2)

### 2.14.2 Operation of Key-Input Interrupt

The following is an operation of key-input interrupt. Figure 2.14.4 shows an example of a circuit that uses the key-input interrupt, Figure 2.14.5 shows an example of operation of key-input interrupt, and Figure 2.14.6 shows the setting procedure of key-input interrupt.

- Operation
- (1) Set the direction register of the ports to be changed to key-input interrupt pins to input, and set the pull-up function.
  - (2) Setting the key-input interrupt control register and setting the interrupt enable flag makes the interrupt-enabled state ready.
  - (3) If a falling edge is input to either  $\overline{KI0}$  through  $\overline{KI7}$ , the key-input interrupt request bit goes to "1".

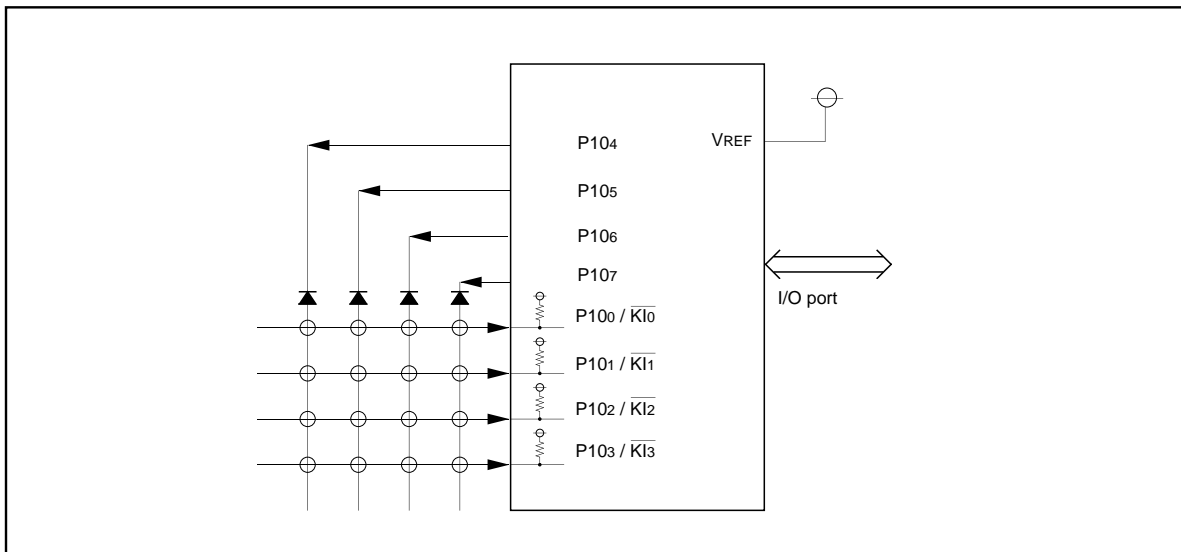


Figure 2.14.4. Example of circuit using the key-input interrupt

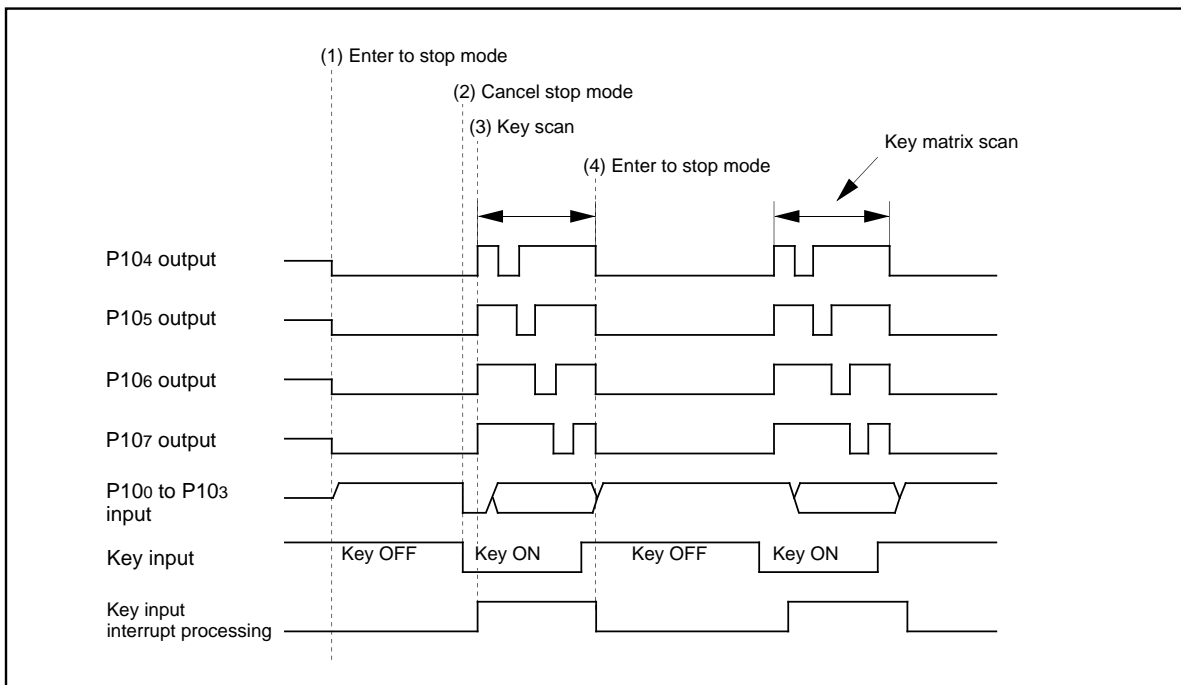


Figure 2.14.5. Example of operation of key-input interrupt

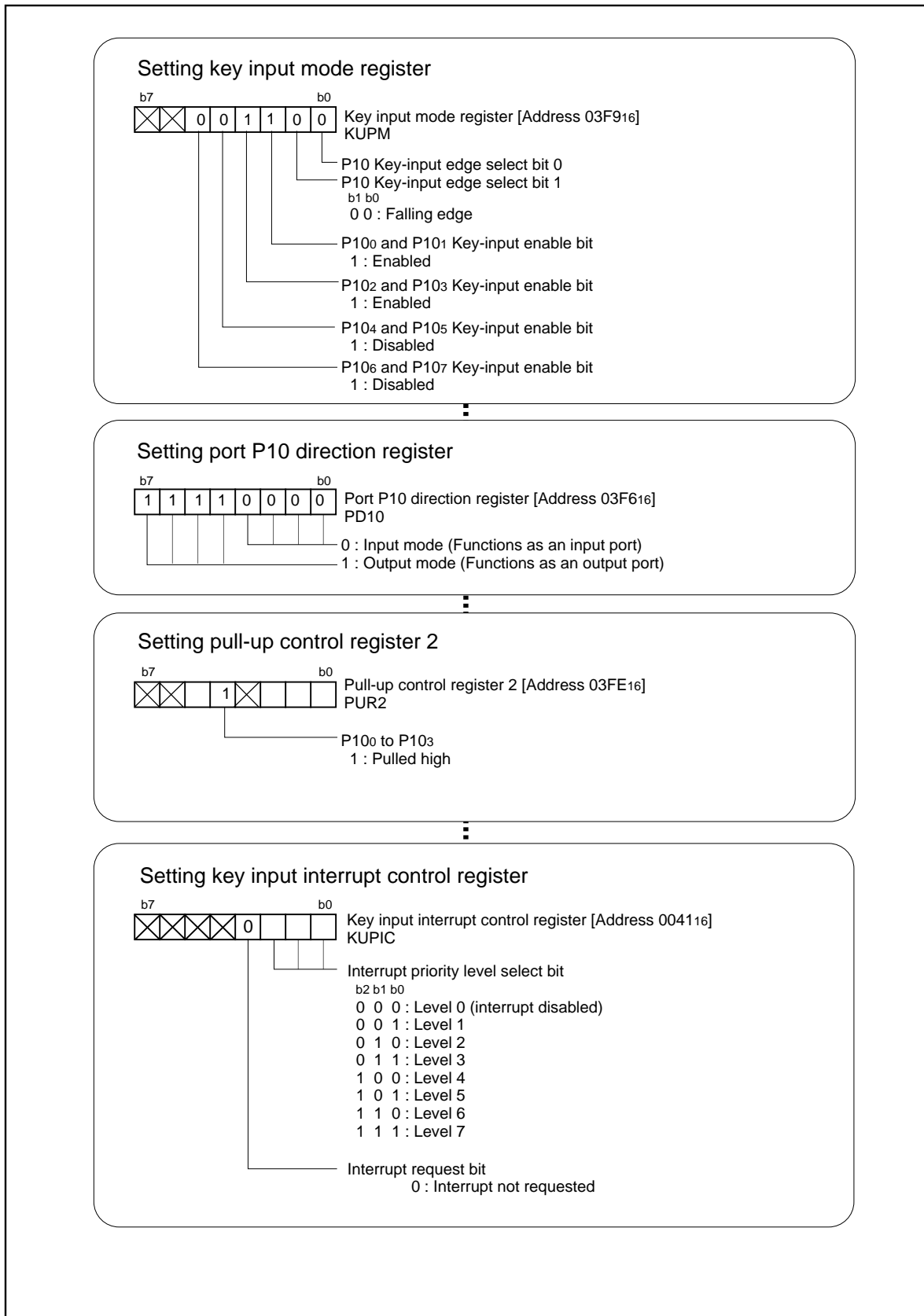


Figure 2.14.6. Set-up procedure of key-input interrupt

## 2.15 Multiple interrupts Usage

### 2.15.1 Overview of the Multiple interrupts usage

The following is an overview of the multiple interrupts usage.

#### (1) 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 non-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 select 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 2.15.1 shows the memory map of the interrupt control registers, and Figure 2.15.2 shows the interrupt control registers.

0041 <sub>16</sub>	Key input interrupt control register (KUPIC)
0042 <sub>16</sub>	UART2 receive/ACK interrupt control register (S2RIC)
0043 <sub>16</sub>	UART1/3 Bus collision interrupt control register (S13BCNIC)
0044 <sub>16</sub>	INT1 interrupt control register (INT1IC)
0045 <sub>16</sub>	Timer A1 interrupt control register (TA1IC)
0046 <sub>16</sub>	USB Endpoint 0 interrupt control register (EP0IC)
0047 <sub>16</sub>	Timer A2 interrupt control register (TA2IC)
0048 <sub>16</sub>	UART1 receive/ACK/SSI1 interrupt control register (S1RIC)
0049 <sub>16</sub>	UART0/2 Bus collision interrupt control register (S02BCNIC)
004A <sub>16</sub>	UART0 receive/ACK/SSI0 interrupt control register (S0RIC)
004B <sub>16</sub>	AD conversion interrupt control register (ADIC)
004C <sub>16</sub>	DMA0 interrupt control register (DM0IC)
004D <sub>16</sub>	UART3 transmit/NACK interrupt control register (S3TIC)
004E <sub>16</sub>	DMA1 interrupt control register (DM1IC)
004F <sub>16</sub>	UART2 transmit/NACK interrupt control register (S2TIC)
0050 <sub>16</sub>	DMA2 interrupt control register (DM2IC)
0051 <sub>16</sub>	UART1 transmit/NACK/SSI1 interrupt control register (S1TIC)
0052 <sub>16</sub>	DMA3 interrupt control register (DM3IC)
0053 <sub>16</sub>	UART0 transmit/NACK/SSI0 interrupt control register (S0TIC)
0054 <sub>16</sub>	Timer A0 interrupt control register (TA0IC)
0055 <sub>16</sub>	UART3 receive/ACK interrupt control register (S3RIC)
0056 <sub>16</sub>	USB suspend interrupt control register (SUSPIC)
0057 <sub>16</sub>	Timer A3 interrupt control register (TA3IC)
0058 <sub>16</sub>	USB resume interrupt control register (RSMIC)
0059 <sub>16</sub>	Timer A4 interrupt control register (TA4IC)
005A <sub>16</sub>	USB reset interrupt control register (RSTIC)
005B <sub>16</sub>	USB SOF interrupt control register (SOFIC)
005C <sub>16</sub>	USB Vbus detect interrupt control register (VBDIC)
005D <sub>16</sub>	USB function interrupt control register (USBFIC)
005E <sub>16</sub>	INT2 interrupt control register (INT2IC)
005F <sub>16</sub>	INT0 interrupt control register (INT0IC)

Figure 2.15.1. Memory map of the interrupt control registers

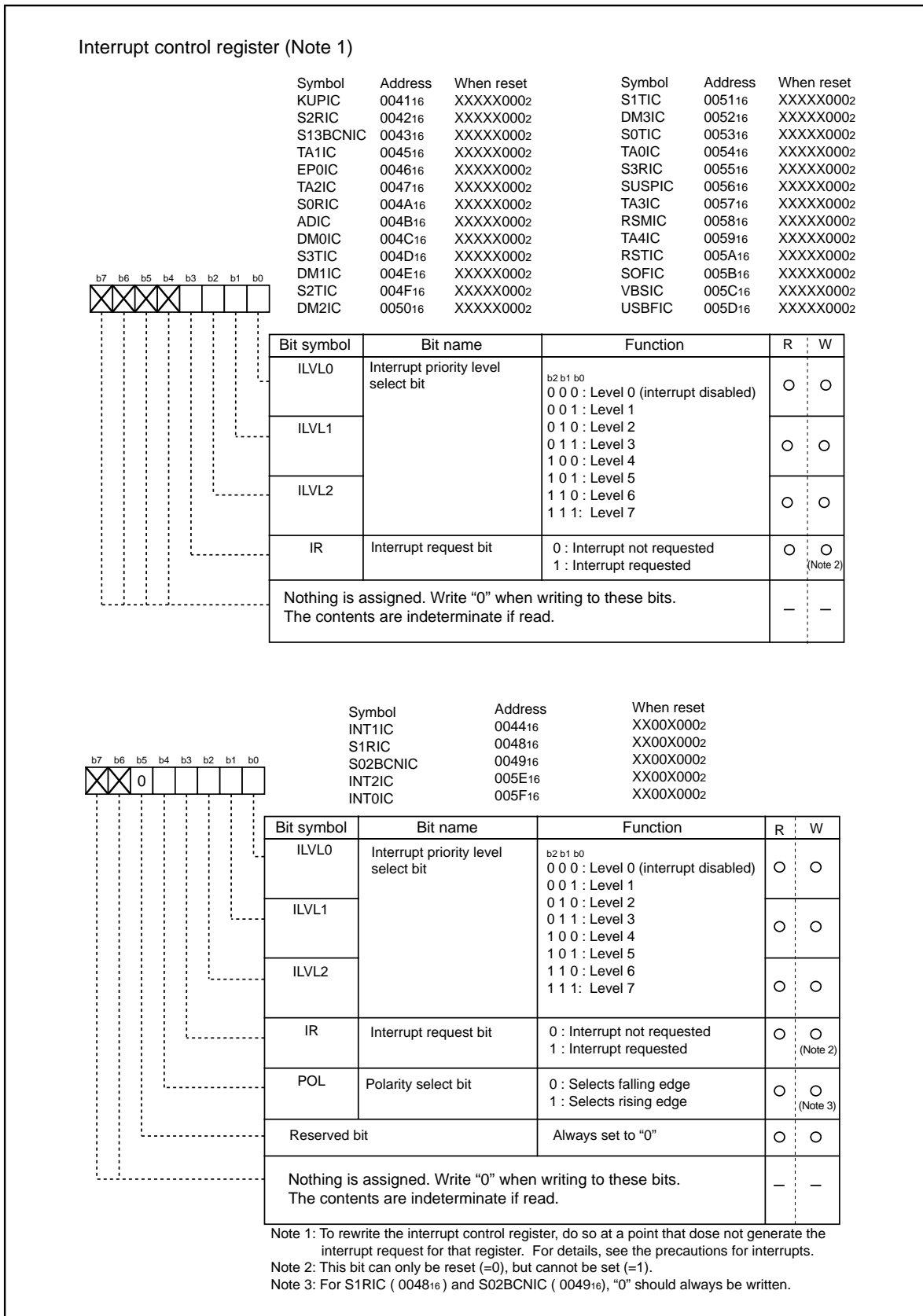


Figure 2.15.2. Interrupt control registers

### (2) 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.

The content is changed when the I flag is changed causes the acceptance of the interrupt request in the following timing:

- When changing the I flag using the REIT instruction, the acceptance of the interrupt takes effect as the REIT instruction is executed.
- When changing the I flag using one of the FCLR, FSET, POPC, and LDC instructions, the acceptance of the interrupt is effective as the next instruction is executed.

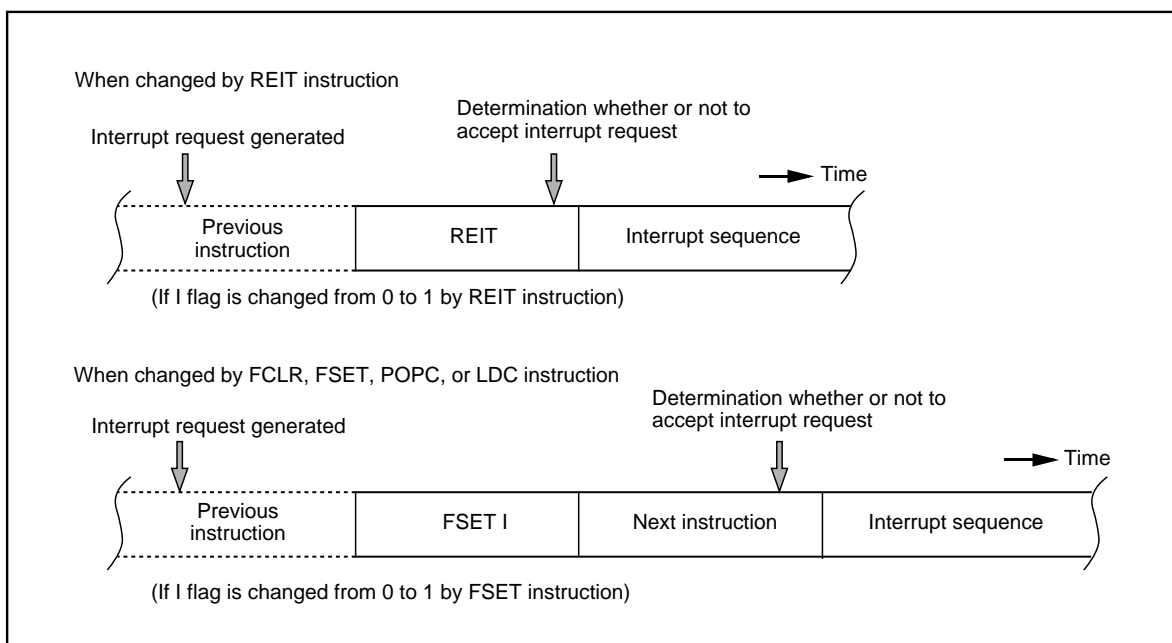


Figure 2.15.3. The timing of reflecting the change in the I flag to the interrupt

### (3) 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").

### (4) 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 2.15.1 shows the settings of interrupt priority levels and Table 2.15.2 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 2.15.1. Settings of interrupt priority levels**

Interrupt priority level select bit	Interrupt priority level	Priority order
b2 b1 b0 0 0 0	Level 0 (interrupt disabled)	_____
0 0 1	Level 1	<div style="text-align: center;">           Low            ↓            High         </div>
0 1 0	Level 2	
0 1 1	Level 3	
1 0 0	Level 4	
1 0 1	Level 5	
1 1 0	Level 6	
1 1 1	Level 7	

**Table 2.15.2. Interrupt levels enabled according to the contents of the IPL**

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

When either the IPL or the interrupt priority level is changed, the new level is reflected to the interrupt in the following timing:

- When changing the IPL using the REIT instruction, the reflection takes effect as of the instruction that is executed in 2 clock cycles after the last clock cycle involved in the REIT instruction.
- When changing the IPL using either the POPC, LDC or LDIPPL instruction, the reflection takes effect as of the instruction that is executed in 3 cycles after the last clock cycle involved in the instruction used.
- When changing the interrupt priority level using the MOV or similar instruction, the reflection takes effect as of the instruction that is executed in 2 clock cycles after the last clock cycle involved in the instruction used.

### (5) 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 2.15.4 shows the priorities of hardware interrupts.

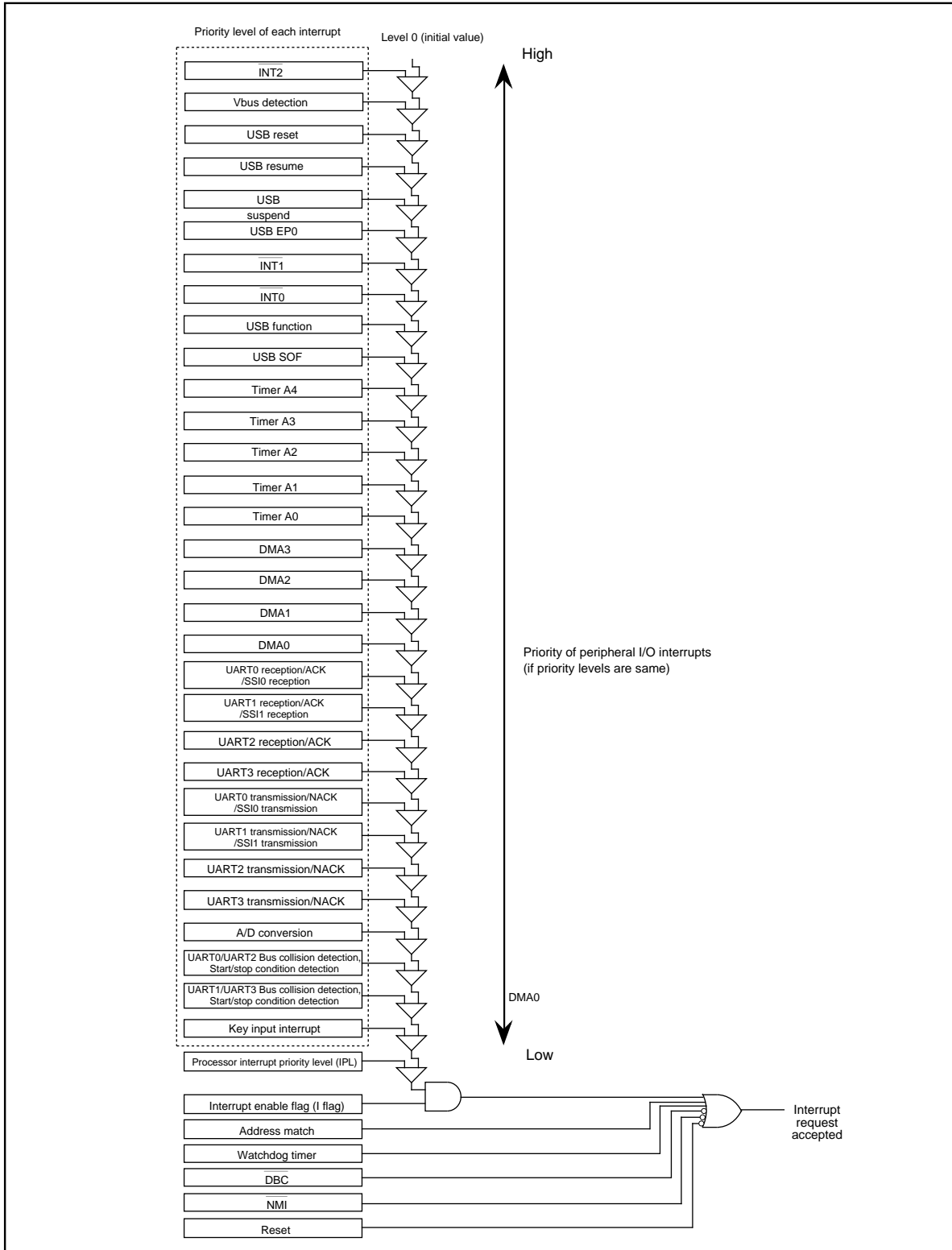
Software interrupts are not affected by the interrupt priority. If an instruction is executed, control branches invariably to the interrupt routine.

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

**Figure 2.15.4. Hardware interrupts priorities**

**(6) Interrupt resolution circuit**

When two or more interrupts are generated simultaneously, this circuit selects the interrupt with the highest priority level. Figure 2.15.5 shows the circuit that judges the interrupt priority level.



**Figure 2.15.5. Interrupts resolution circuit**



### 2.15.2 Multiple Interrupts Operation

The state when control branched to an interrupt routine is described below:

- The interrupt enable flag (I flag) is set to "0" (the interrupt is disabled).
- The interrupt request bit of the accepted interrupt is set to "0".
- The processor interrupt priority level (IPL) is assigned to the same interrupt priority level as assigned to the accepted interrupt.

Setting the interrupt enable flag (I flag) to "1" within an interrupt routine allows an interrupt request assigned a priority higher than the IPL to be accepted.

An interrupt request that is not accepted because of low priority will be held. If the condition following is met when the REIT instruction returns the IPL and the interrupt priority is determined, then the interrupt request being held is accepted.

Interrupt priority level of the interrupt request being held > Returned the IPL

Figure 2.15.6 shows the example of the multiple interrupts operation.

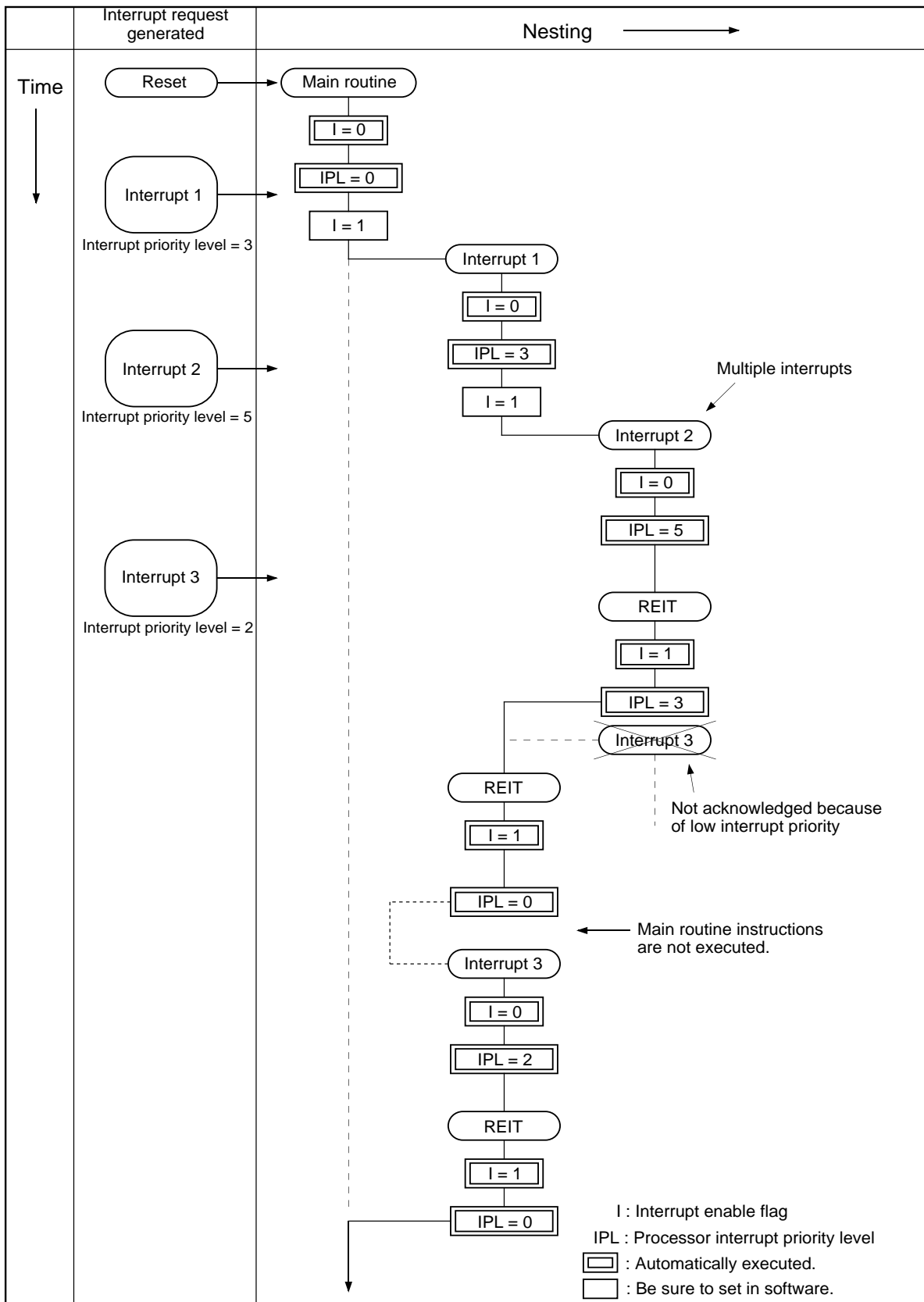


Figure 2.15.6. Example of the multiple interrupts operation

## 2.16 Power Control Usage

### 2.16.1 Overview of the power control usage

'Power Control' refers to the reduction of CPU power consumption by stopping the CPU and oscillators, or decreasing the operation clock. The following is a description of the three available power control modes:

#### (1) 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 selected. 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 according to the BCLK selected. 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 secondary clock. Each peripheral function operates according to its assigned clock.

###### • Low power consumption 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 secondary clock. The only peripheral functions that operate are those with the sub-clock selected as the count source.

##### (b) Wait mode

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

##### (c) Stop mode

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

Figure 2.16.1 is the state transition diagram of the above modes.

#### (2) Switching the driving capacity of the oscillation circuit

Both the main clock and the secondary clock have the ability to switch the driving capacity.

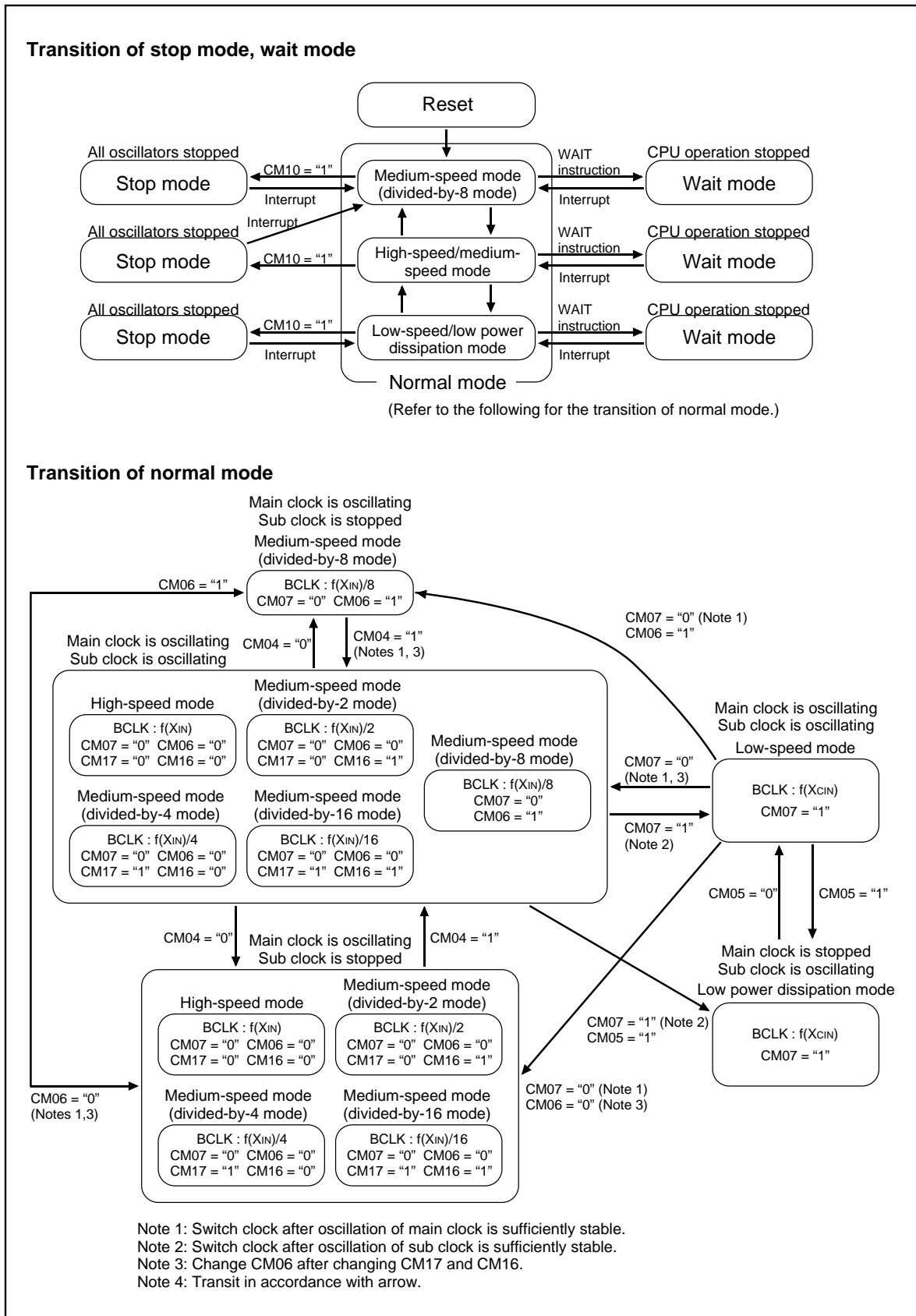


Figure 2.16.1. State transition diagram of power control mode

**(3) Returning from stop mode**

The stop mode can be canceled by hardware reset, or by generating an interrupt request. 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 not to be used for clearing must be set to level 0 before changing to stop mode.

If an interrupt is used to cancel stop mode, that interrupt routine is processed. If only a hardware reset and NMI interrupt are used to cancel stop mode, the priority level of all interrupts must be set to level 0 before changing to stop mode.

When changing from high-speed/medium mode to stop mode and at reset, the main clock division select bit 0 (bit 6 at address 000616) is set to "1". When changing from low-speed/low power dissipation mode, the value before stop mode is retained.

**(4) Returning from wait mode**

The wait mode can be canceled by hardware reset, or by generating an interrupt request. If an interrupt is to be used to cancel wait mode, that interrupt must first have been enabled and the priority level of the interrupt not to be used for clearing must be set to level 0 before changing to wait mode.

If an interrupt is used to cancel wait mode, the microcomputer selects the clock used when the WAIT instruction is executed for BCLK and restarts operating in that interrupt routine. If only a hardware reset or NMI interrupt will be used to cancel wait mode, the priority level of all interrupts must be set to level 0 before changing to stop mode.

Table 2.16.1 shows the interrupts that can be used for canceling stop mode and wait mode.

**(5) BCLK in returning from wait mode or stop mode****(a) Returning from wait mode**

The processor immediately returns to the BCLK, which was in use before entering wait mode.

**(b) Returning from stop mode**

CM06 is set to "1" when the device enters stop mode after selecting the main clock for BCLK. CM17 and CM16 do not change state. In this case, when restored from stop mode, the device starts operating in divided-by-8 mode.

When the device enters stop mode after selecting the subclock for BCLK, CM06, CM17, CM16, and CM07 all do not change state. In this case, when restored from stop mode, the device starts operating in low-speed mode.

**Table 2.16.1. Interrupts available for clearing stop mode and wait mode**

Interrupt for clearing	Wait mode		Stop mode
	CM02 = 0	CM02 = 1(Note 6), CM07 = 0, CM05 = 0	
UART0/UART2 Bus collision detection, Start/stop condition detection interrupt	Possible	Note 1	Note 1
UART1/UART3 Bus collision detection, Start/stop condition detection interrupt	Possible	Note 1	Note 1
DMA0 interrupt	Impossible	Impossible	Impossible
DMA1 interrupt	Impossible	Impossible	Impossible
DMA2 interrupt	Impossible	Impossible	Impossible
DMA3 interrupt	Impossible	Impossible	Impossible
Key input interrupt	Possible	Possible	Possible
A/D interrupt	Note 3	Impossible	Impossible
UART0 transmit/NACK /SSI0 transmit interrupt (Note7)	Possible	Note 1	Note 1
UART0 receive/ACK /SSI0 receive interrupt (Note 7)	Possible	Note 1	Note 1
UART1 transmit/NACK /SSI1 transmit interrupt (Note7)	Possible	Note 1	Note 1
UART1 receive/ACK /SSI1 receive interrupt (Note 7)	Possible	Note 1	Note 1
UART2 transmit/NACK interrupt (Note 7)	Possible	Note 1	Note 1
UART2 receive/ACK interrupt (Note 7)	Possible	Note 1	Note 1
UART3 transmit/NACK interrupt (Note 7)	Possible	Note 1	Note 1
UART3 receive/ACK interrupt (Note 7)	Possible	Note 1	Note 1
Timer A0 interrupt	Possible	Note 2, Note 4	Note 2
Timer A1 interrupt	Possible	Note 2, Note 4	Note 2
Timer A2 interrupt	Possible	Note 2, Note 4	Note 2
Timer A3 interrupt	Possible	Note 2, Note 4	Note 2
Timer A4 interrupt	Possible	Note 2, Note 4	Note 2
USB suspend interrupt	Possible	Possible	Impossible
USB resume interrupt	Possible	Possible	Note 5
USB reset interrupt	Possible	Possible	Impossible
USB SOF interrupt	Possible	Possible	Impossible
USB Vbus detection interrupt	Possible	Possible	Possible
USB function interrupt	Possible	Possible	Impossible
USB EP0 interrupt	Possible	Possible	Impossible
INT0 interrupt	Possible	Possible	Possible
INT1 interrupt	Possible	Possible	Possible
INT2 interrupt	Possible	Possible	Possible
NMI interrupt	Possible	Possible	Possible

Note 1: Can be used when an external clock is selected.

Note 2: Can be used when the external signal is being counted in event counter mode.

Note 3: Can be used in one-shot mode and one-shot sweep mode.

Note 4: Can be used when count source is fc32.

Note 5: Only when USB suspend mode.

Note 6: When the MCU running in low-speed or low power dissipation mode, do not enter wait mode with CM02 is set to "1".

Note 7: When I<sup>2</sup>C mode is selected, NACK/ACK, start/stop condition detection interrupt are selected, and when SS pin is selected, trouble error interrupt is selected.

### (5) Sequence of returning from stop mode

Sequence of returning from stop mode is oscillation start-up time and interrupt sequence. When interrupt is generated in stop mode, CM10 becomes "0" and clearing stop mode. Starting oscillation and supplying BCLK execute the interrupt sequence as follow:

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

- CPU gets the interrupt information (the interrupt number and interrupt request level) by reading address  $00000_{16}$ . The interrupt request bit of the interrupt written in address  $00000_{16}$  will then be set to "0".
- 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.
- Sets the interrupt enable flag (I flag), the debug flag (D flag), and the stack pointer assignment 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)
- Saves the content of the temporary register (Note) within the CPU in the stack area.
- Saves the content of the program counter (PC) in the stack area.
- Sets the interrupt priority level of the accepted instruction in the IPL.

Note: This register cannot be utilized by the user.

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

Figure 2.16.2 shows the sequence of returning from stop mode.

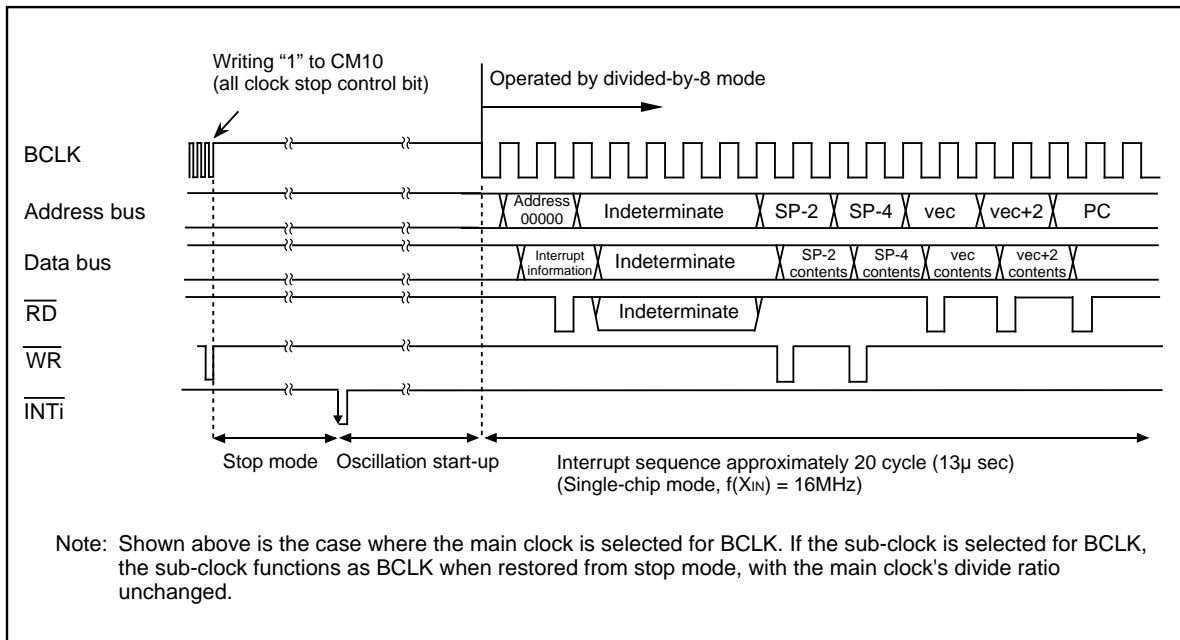


Figure 2.16.2. Sequence of returning from stop mode

### (6) Registers related to power control

Figure 2.16.3 shows the memory map of power control-related registers, and Figure 2.16.4 shows power control-related registers.

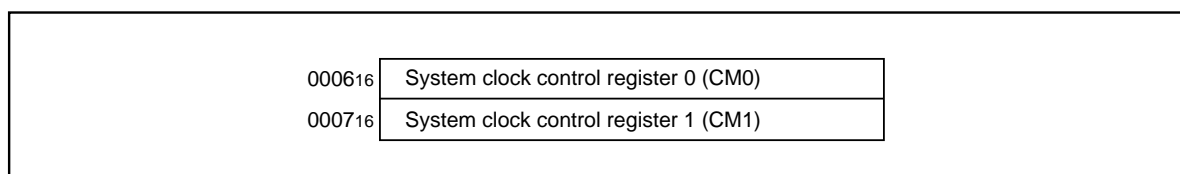


Figure 2.16.3. Memory map of power control-related registers

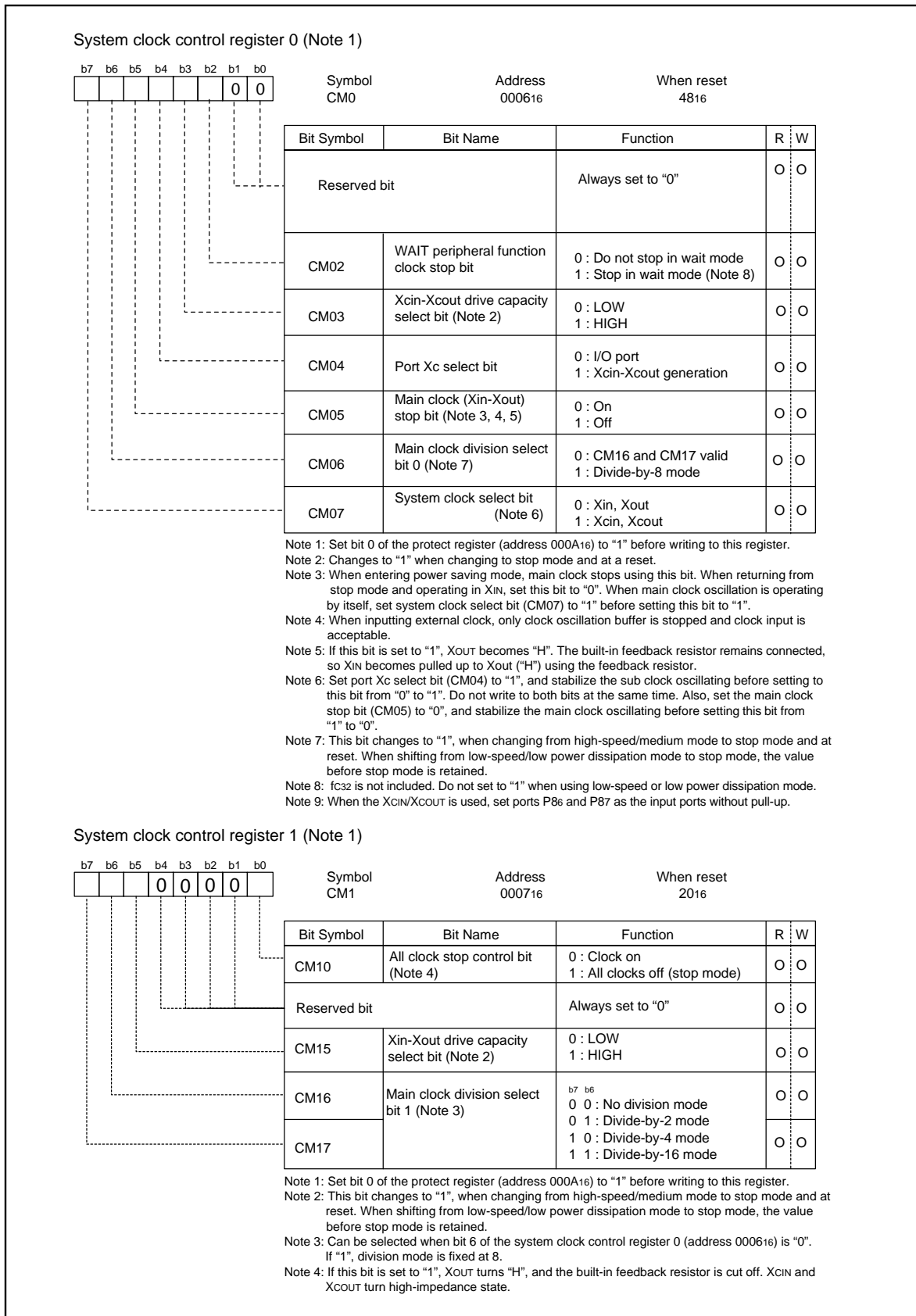


Figure 2.16.4. Power control-related registers



### 2.16.2 Stop Mode Set-Up

Settings and operation for entering stop mode are described here.

- Operation
- (1) Enables the interrupt used for returning from stop mode.
  - (2) Sets the interrupt enable flag (I flag) to "1".
  - (3) Clearing the protection and setting all clock stop control bit to "1" stops oscillation and causes the processor to go into stop mode.

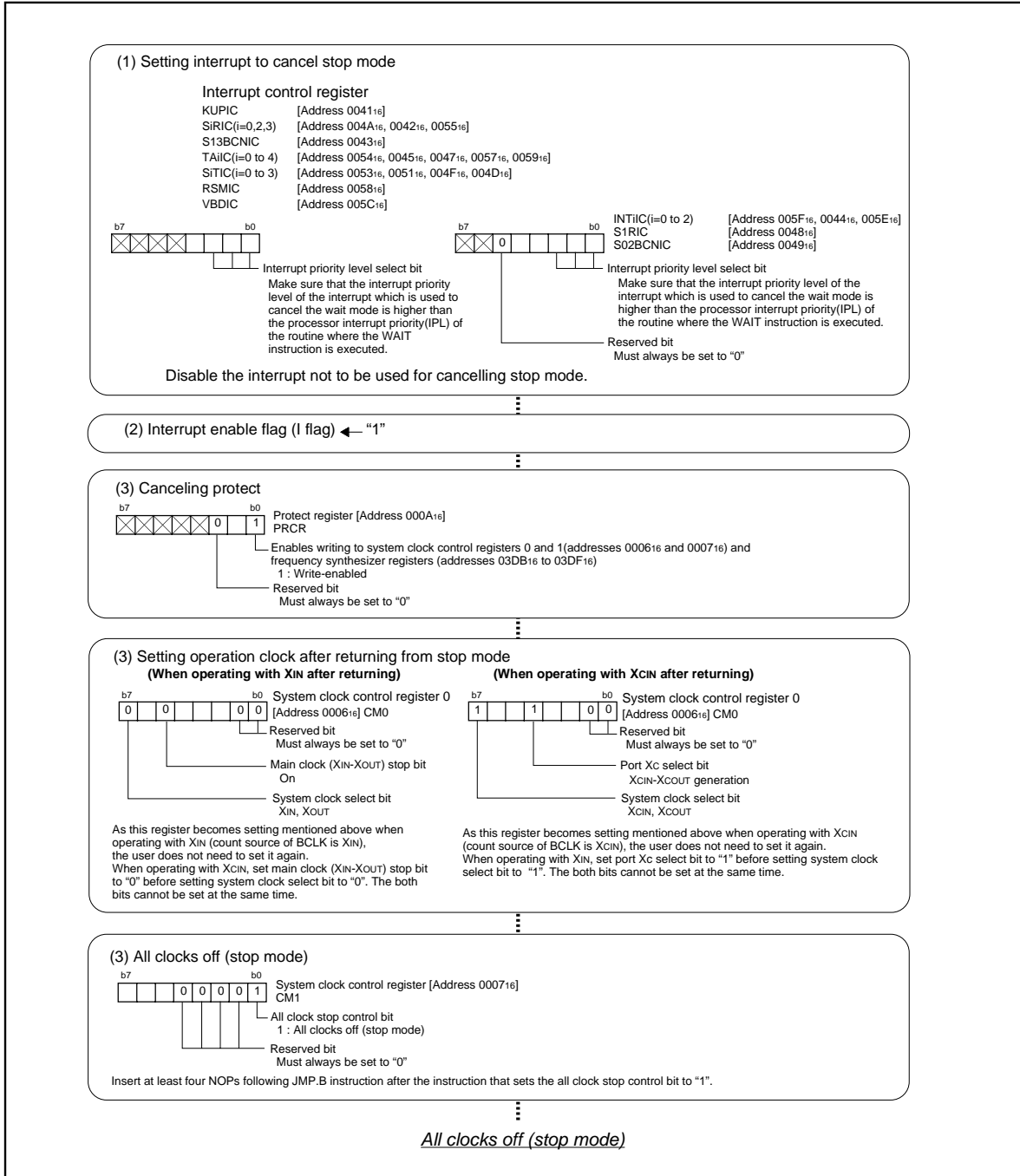


Figure 2.16.5. Example of stop mode set-up

### 2.16.3 Wait Mode Set-Up

Settings and operation for entering wait mode are described here.

- Operation
- (1) Enables the interrupt used for returning from wait mode.
  - (2) Sets the interrupt enable flag (I flag) to "1".
  - (3) Clears the protection and changes the content of the system clock control register.
  - (4) Executes the WAIT instruction.

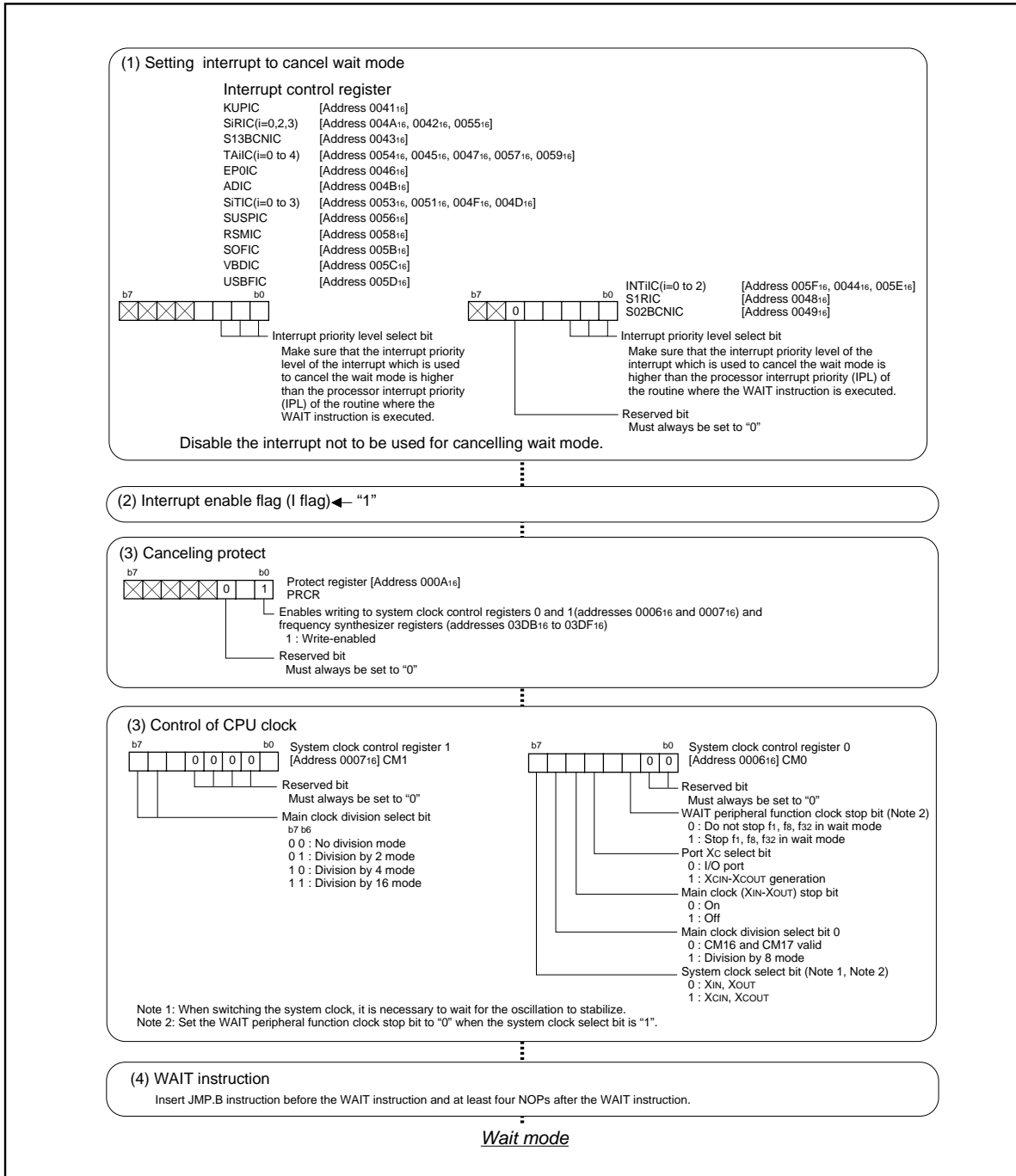


Figure 2.16.6. Example of wait mode set-up

### 2.16.4 Precautions in Power Control

- (1) The processor does not switch to stop mode when the  $\overline{\text{NMI}}$  pin is at "L" level.
- (2) When returning from stop mode by hardware reset,  $\overline{\text{RESET}}$  pin must be set to "L" level until main clock oscillation is stabilized.
- (3) When entering wait mode, insert a JMP.B instruction before a WAIT instruction. Do not execute any instructions which can generate a write to RAM between the JMP.B and WAIT instructions. Disable the DMA transfers, if a DMA transfer may occur between the JMP.B and WAIT instructions. After the WAIT instruction, insert at least 4 NOP instructions. When entering wait mode, the instruction queue roadstead the instructions following WAIT, and depending on timing, some of these may execute before the microcomputer enters wait mode.

Program example when entering wait mode

```

Program Example:   JMP.B   L1   ; Insert JMP.B instruction before WAIT instruction
                  L1:
                   FSET    I    ;
                   WAIT    ; Enter wait mode
                   NOP     ; More than 4 NOP instructions
                   NOP
                   NOP
                   NOP

```

- (4) When entering stop mode, insert a JMP.B instruction immediately after executing an instruction which sets the CM10 bit in the CM1 register to "1", and then insert at least 4 NOP instructions. When entering stop mode, the instruction queue reads ahead the instructions following the instruction which sets the CM10 bit to "1" (all clock stops), and, some of these may execute before the microcomputer enters stop mode or before the interrupt routine for returning from stop mode.

Program example when entering stop mode

```

Program Example:   FSET    I
                  BSET    CM10 ; Enter stop mode
                  JMP.B   L2   ; Insert JMP.B instruction
                  L2:
                   NOP     ; More than 4 NOP instructions
                   NOP
                   NOP
                   NOP

```

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

(6) Suggestions to reduce power consumption

**(a) Ports**

The processor retains the state of each programmable I/O port even when it goes to wait mode or to stop mode. A current flows in active I/O ports. A pass current flows in input ports that float. When entering wait mode or stop mode, set non-used ports to input and stabilize the potential.

**(b) Memory expansion mode and microprocessor mode**

When the MCU enters wait mode while operating in memory expansion mode or microprocessor mode, a pin functioning as part of the address or data bus retains its state on the bus before wait mode is entered. Shift to single-chip mode and output an arbitrary value in order to reduce current consumption. By shifting to single-chip mode, a pin which was functioning as part of the bus becomes a general-purpose port and can output an arbitrary value. Set the port registers and direction registers after shifting to single-chip mode (this implies that any control pins (CS, WR, RD, etc..) being used for access of an external device be changed as well).

The same applies to stop mode.

**(c) A/D converter**

A current always flows in the VREF pin. When entering wait mode or stop mode, set the Vref connection bit to "0" so that no current flows into the VREF pin.

**(d) Stopping peripheral functions**

In wait mode, stop non-used peripheral functions using the WAIT peripheral function clock stop bit. However, peripheral function clock fc32 does not stop so that the peripherals using fc32 do not contribute to the power saving. When the MCU running in low-speed or low power dissipation mode, do not enter WAIT mode with this bit set to "1".

**(e) External clock**

When using an external clock input for the CPU clock, set the main clock stop bit to "1". Setting the main clock stop bit to "1" causes the XOUT pin not to operate and the power consumption goes down (when using an external clock input, the clock signal is input regardless of the content of the main clock stop bit).

## 2.17 Programmable I/O Ports Usage

### 2.17.1 Overview of the programmable I/O ports usage

Eighty-one programmable I/O ports and one input-only port are available. I/O pins also serve as I/O pins for built-in peripheral functions.

Each port has a direction register that defines the I/O direction and also has a port register for I/O data. In addition, each port has a pull-up control register that defines pull-up in terms of 4 bits. The input-only port has neither direction register nor pull-up control bit.

The following is an overview of the programmable I/O ports usage:

#### (1) Writing to a port register

With the direction register set to output, the level of the written values from each relevant pin is output by writing to a port register. The output level conforms to CMOS output. Port P70 and P71 are N channel open drain. Writing to the port register, with the direction register set to input, inputs a value to the port register, but nothing is output to the relevant pins. The output level remains floating.

In memory expansion and microprocessor mode, the contents of corresponding port and direction registers of pins A0 to A19, D0 to D15,  $\overline{CS0}$  to  $\overline{CS3}$ , RD,  $\overline{WRL/WR}$ ,  $\overline{WRH/BHE}$ , ALE, RDY,  $\overline{HOLD}$ ,  $\overline{HLDA}$  and BCLK cannot be modified.

#### (2) Reading a port register

With the direction register set to output, reading a port register takes out the content of the port register, not the content of the pin. With the direction register set to input, reading the port register takes out the content of the pin.

#### (3) Input-only port

P85 is used as the input-only port, it also serves as  $\overline{NMI}$ . P85 has no direction register. Pull-up cannot be set to this port. As  $\overline{NMI}$  cannot be disabled, an  $\overline{NMI}$  interrupt occurs if a falling edge is input to P85. Use P85 for reading the level input at this time only.

#### (4) Setting pull-up

The pull-up control bit allows setting of the pull-up, in terms of 4 bits, either in use or not in use. For the four bits chosen, pull-up is effective only in the ports whose direction register is set to input. Pull-up is not effective in ports whose direction register is set to output.

Do not set pull-up of corresponding pin when  $\overline{XCIN/XCOUT}$  is set or a port is used as A/D input.

Pull-up can be set for P0 to P3, P40 to P43, P50 to P53 in only single-chip mode. Pull-up cannot be set for P0 to P3, P40 to P43, P50 to P53 in memory expansion and microprocessor modes. The contents of register can be changed, but the pull-up resistance is not connected.

#### (5) Setting drive capacity

A normal drive and N-channel high drive for the N-channel transistor drive capacity of port P7 can be selected. Port P7 can be configured to drive an LED by increasing the drive strength of the corresponding N-channel transistor bits.

**(6) I/O functions of built-in peripheral devices**

Table 2.17.1 shows relation between ports and I/O functions of built-in peripheral devices.

**Table 2.17.1. Relation between ports and I/O functions of built-in peripheral devices**

Port	Internal peripheral device I/O pins
P6	I/O pins/Serial sound interface, I <sup>2</sup> C and SPI communication pins for UART0 and UART1
P7	Timer A0 to A3 I/O pins / I/O pins or I <sup>2</sup> C, SPI communication pins for UART2 and UART3 / LED drive output pins
P80, P81	Timer A4 I/O pins
P82, P83, P84	Input pins for external interrupt
P86, P87	Sub-clock oscillation circuit I/O pins
P90	Attach/Detach control pin for USB
P92	SOF output pin for USB
P93	A/D trigger input pin
P10	A/D converter input pins / key-input interrupt function input pins

**(7) Examples of working on non-used pins**

Table 2.17.2 contains examples of working on non-used pins. There are shown here for mere examples. In practical use, make suitable changes and perform sufficient evaluation in compliance with you application.

**(a) Single-chip mode****Table 2.17.2. Examples of working on unused pins in single-chip mode**

Pin name	Connection
Ports P0 to P10 (excluding P85)	After setting for input mode, connect every pin to VSS or VCC via a resistor; or after setting for output mode, leave these pins open. (Note 1, Note 2, Note 3)
XOUT (Note 1)	Open
$\overline{\text{NMI}}$	Connect to VCC via a resistor (pull-up)
UVCC, AVCC	Connect to VCC
AVSS, VREF, BYTE	Connect to VSS
USB D+, USB D-	Open
LPF	Open
VbusDTCT	Open

Note 1: When an external clock is input to the XIN pin.

Note 2: If setting these pins in output mode and opening them, ports are in input mode until switched into output mode by use of software after reset. Thus the voltage levels of the pins become unstable, and there can be instances in which the power source current increases while the ports are in input mode.

In view of an instance in which the contents of the direction registers change due to a runaway generated by noise or other causes, setting the contents of the direction registers periodically by use of software increases program reliability.

Note 3: Output "L" if port P70 and P71 are set to output mode.  
Port P70 and P71 are N channel open drain output.

**(b) Memory expansion mode, microprocessor mode****Table 2.17.3. Examples of working on unused pins in memory expansion mode or microprocessor mode**

Pin name	Connection
Ports P6 to P10 (excluding P8s)	After setting for input mode, connect every pin to Vss or Vcc via a resistor; or after setting for output mode, leave these pins open. (Note 2, Note 3, Note 4)
P4s/ $\overline{CS1}$ to P47/ $\overline{CS3}$	After setting for port input mode and setting CS1 to CS3 output enable bit to "0", connect to Vcc via a resistor (pull-up)
$\overline{BHE}$ (Note 5), ALE(Note 5), $\overline{HLDA}$ (Note 5), XOUT (Note 1), BCLK	Open
HOLD, RDY, $\overline{NMI}$	Connect via resistor to Vcc (pull-up)
UVcc, AVcc	Connect to Vss
AVss, VREF	Open
USB D+, USB D-	Open
LPF	Open
VbusDTCT (Note 6)	Open

Note 1: When an external clock is input to the XIN pin.

Note 2: If setting these pins in output mode and opening them, ports are in input mode until switched into output mode by use of software after reset. Thus the voltage levels of the pins become unstable, and there can be instances in which the power source current increases while the ports are in input mode.

In view of an instance in which the contents of the direction registers change due to a runaway generated by noise or other causes, setting the contents of the direction registers periodically by use of software increases program reliability.

Note 3: Make wiring as short as possible (not more than 2 cm from the microcomputer's pins) in working on non-used pins.

Note 4: Output "L" if port P70 and P71 are set to output mode. Port P70 and P71 are N channel open drain output.

Note 5: When a Vss level is connected to the CNVss pin, these pins are input ports until the processor mode is switched by use of software after reset. Thus the voltage levels of the pins destabilize, and there can be an increase in the power source current while these pins are input ports.

Note 6: VbusDTCT pin is pulled down internally.

**(8) Registers related to the programmable I/O ports**

Figure 2.17.1 shows the memory map of programmable I/O ports-related registers, and Figures 2.17.2 to 2.17.5 show programmable I/O ports-related registers.

03E0 <sub>16</sub>	Port P0 (P0)
03E1 <sub>16</sub>	Port P1 (P1)
03E2 <sub>16</sub>	Port P0 direction register (PD0)
03E3 <sub>16</sub>	Port P1 direction register (PD1)
03E4 <sub>16</sub>	Port P2 (P2)
03E5 <sub>16</sub>	Port P3 (P3)
03E6 <sub>16</sub>	Port P2 direction register (PD2)
03E7 <sub>16</sub>	Port P3 direction register (PD3)
03E8 <sub>16</sub>	Port P4 (P4)
03E9 <sub>16</sub>	Port P5 (P5)
03EA <sub>16</sub>	Port P4 direction register (PD4)
03EB <sub>16</sub>	Port P5 direction register (PD5)
03EC <sub>16</sub>	Port P6 (P6)
03ED <sub>16</sub>	Port P7 (P7)
03EE <sub>16</sub>	Port P6 direction register (PD6)
03EF <sub>16</sub>	Port P7 direction register (PD7)
03F0 <sub>16</sub>	Port P8 (P8)
03F1 <sub>16</sub>	Port P9 (P9)
03F2 <sub>16</sub>	Port P8 direction register (PD8)
03F3 <sub>16</sub>	Port P9 direction register (PD9)
03F4 <sub>16</sub>	Port P10 (P10)
03F5 <sub>16</sub>	
03F6 <sub>16</sub>	Port P10 direction register (PD10)
≈	≈
03FA <sub>16</sub>	P7 drive capacity register (P7DR)
03FB <sub>16</sub>	
03FC <sub>16</sub>	Pull-up control register 0 (PUR0)
03FD <sub>16</sub>	Pull-up control register 1 (PUR1)
03FE <sub>16</sub>	Pull-up control register 2 (PUR2)
03FF <sub>16</sub>	Port control register (PCR)

**Figure 2.17.1. Memory map of programmable I/O ports-related registers**



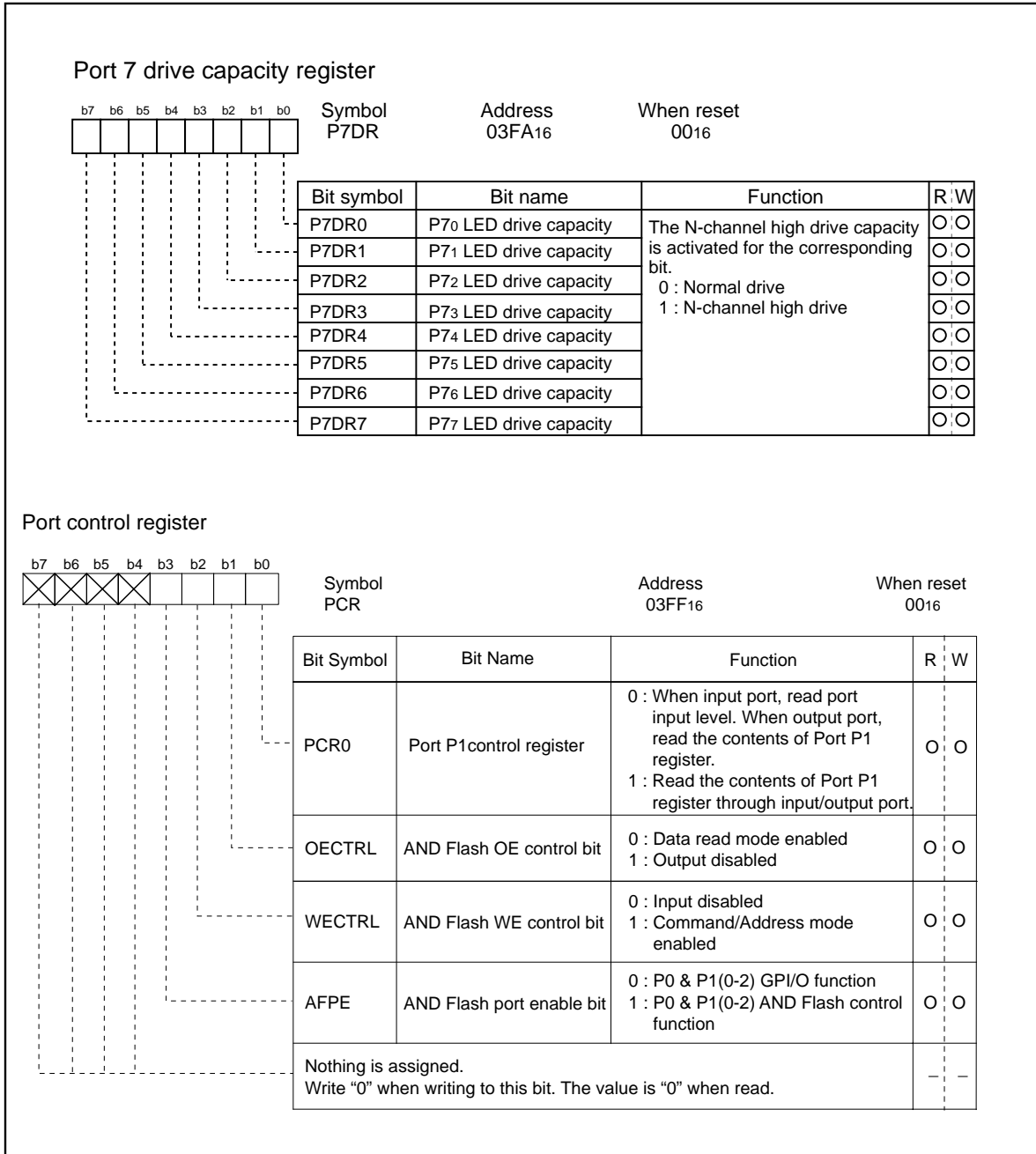


Figure 2.17.2. Programmable I/O ports-related registers (1)

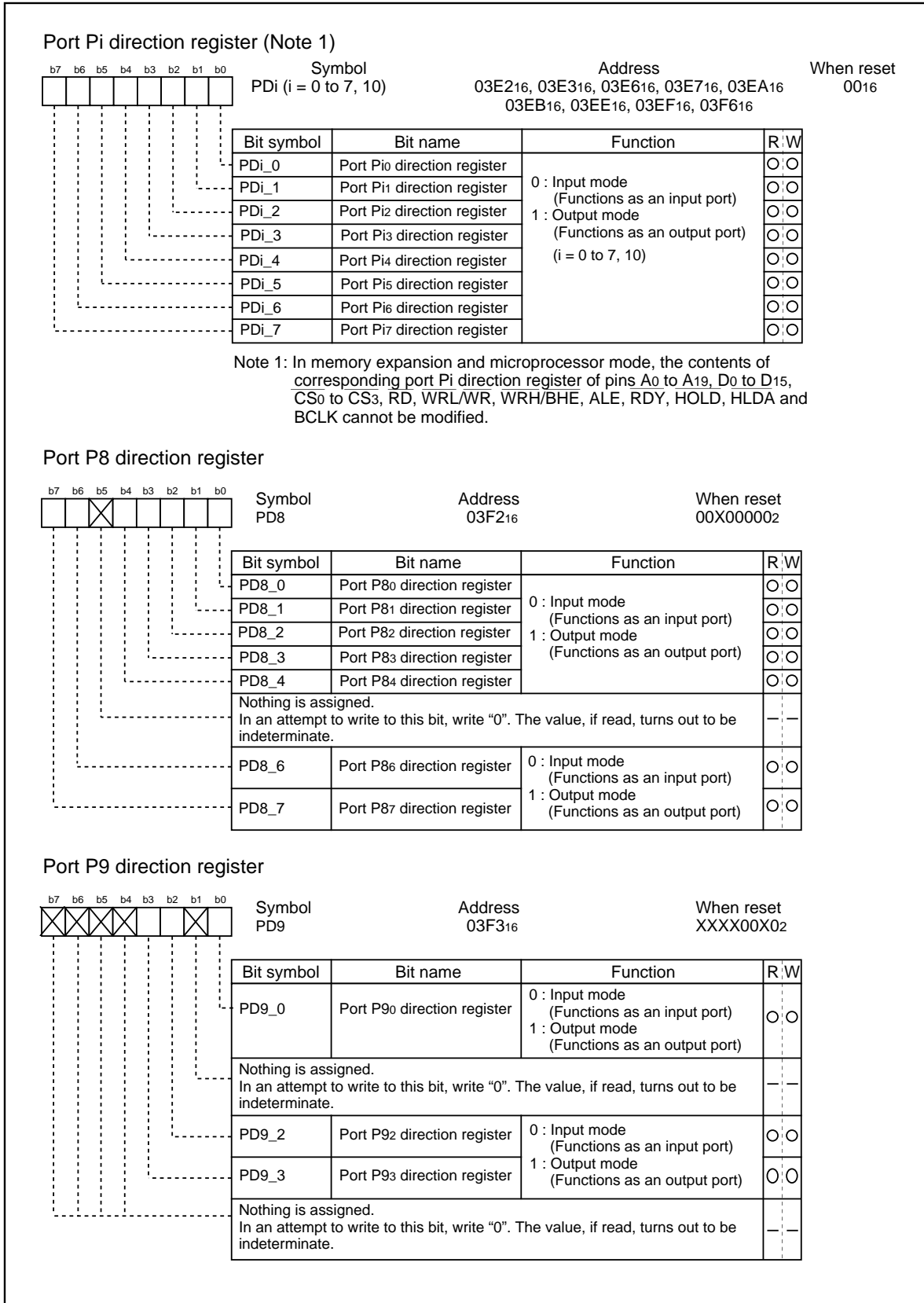
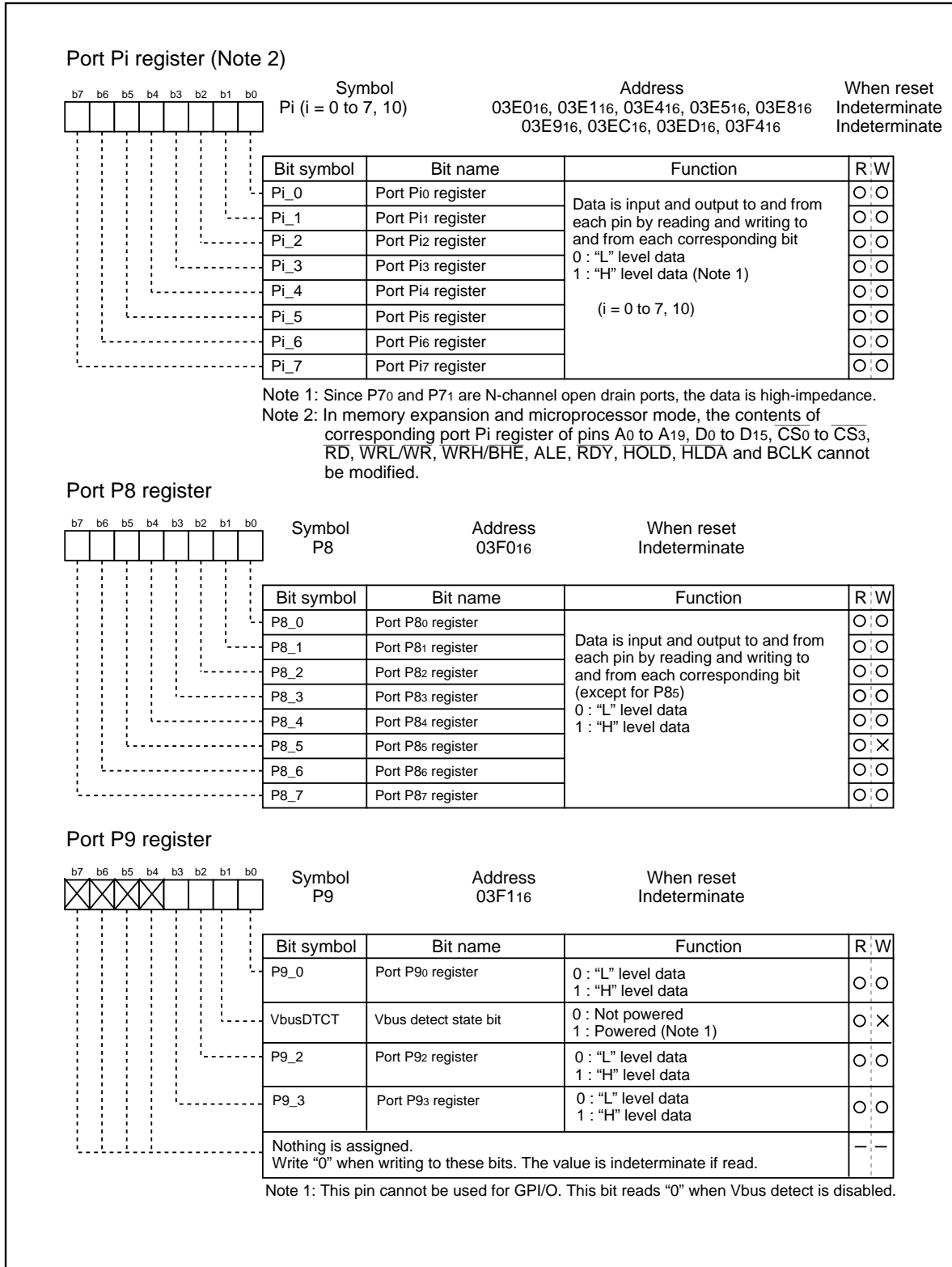


Figure 2.17.3. Programmable I/O ports-related registers (2)



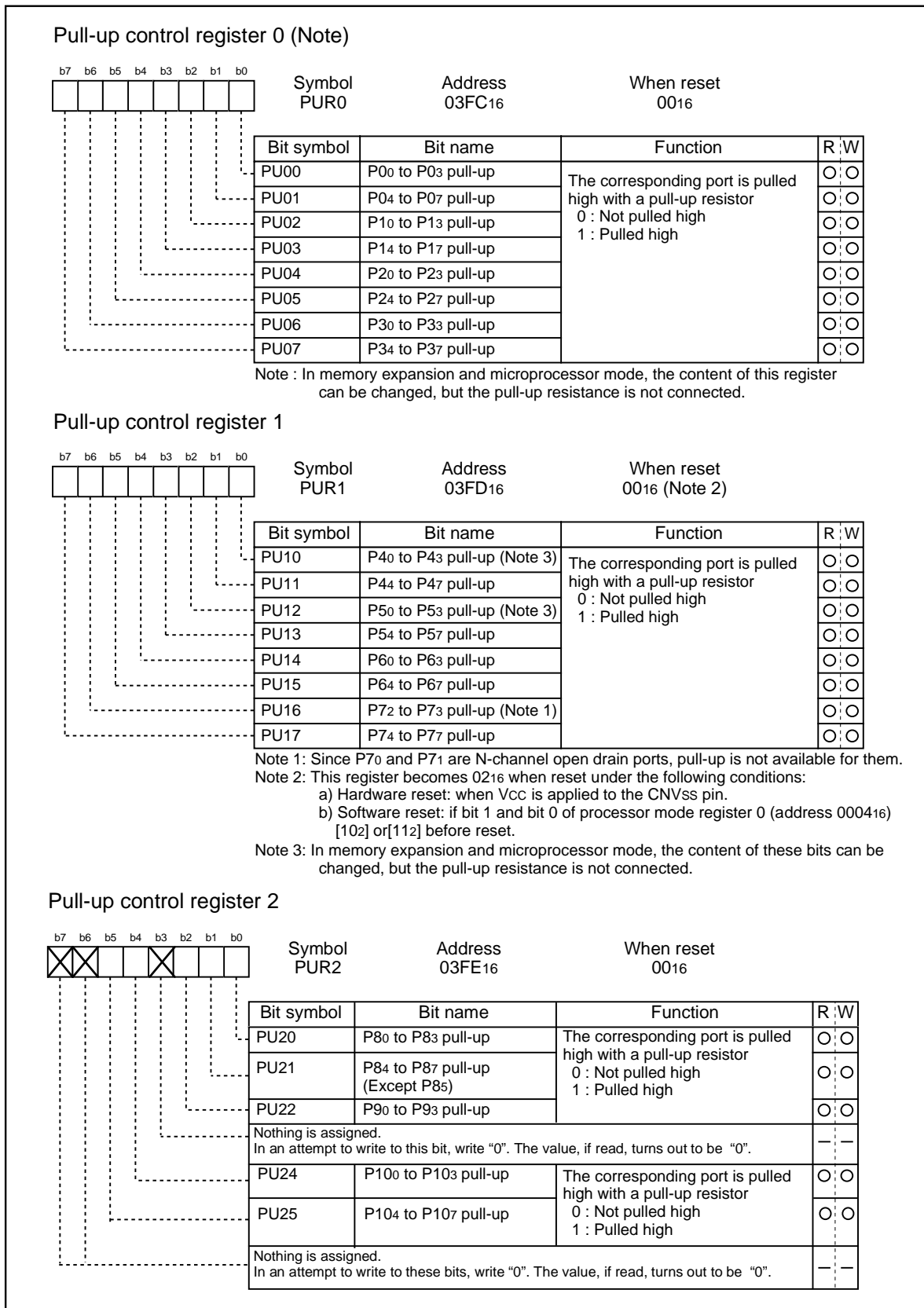


Figure 2.17.5. Programmable I/O ports-related registers (4)

THIS PAGE IS BLANK FOR REASONS OF LAYOUT.

# Chapter 3

---

## Examples of Peripheral Functions Applications

This chapter presents applications in which peripheral functions built in the M30245 are used. They are shown here as examples. In practical use, make suitable changes and perform sufficient evaluation. For basic use, see Chapter 2 Peripheral Functions Usage.

### 3.1 Long-Period Timers

**Overview** In this process, Timer A0 and Timer A1 are connected to make a 16-bit timer with a 16-bit prescaler. Figure 3.1.1 shows the operation timing, Figure 3.1.2 shows the connection diagram, and Figures 3.1.3 and 3.1.4 show the set-up procedure.

Use the following peripheral functions:

- Timer mode of timer A
- Event counter mode of timer A

#### Specifications

- (1) Set timer A0 to timer mode, and set timer A1 to event counter mode.
- (2) Perform a count on count source f1 using timer A0 to count for 1 ms, and perform a count on timer A0 using timer A1 to count for 1 second.
- (3) Connect a 16-MHz oscillator to XIN.

**Operation**

- (1) Setting the count start flag to "1" causes the counter to begin counting. The counter of timer A0 performs a down count on count source f1.
- (2) If the counter of timer A0 underflows, the counter reloads the content of the reload register and continues counting. At this time, the timer A0 interrupt request bit goes to "1". The counter of timer A1 performs a down count on underflows in timer A0.
- (3) If the counter of timer A1 underflows, the counter reloads the content of the reload register and continues counting. At this time, the timer A1 interrupt request bit goes to "1".

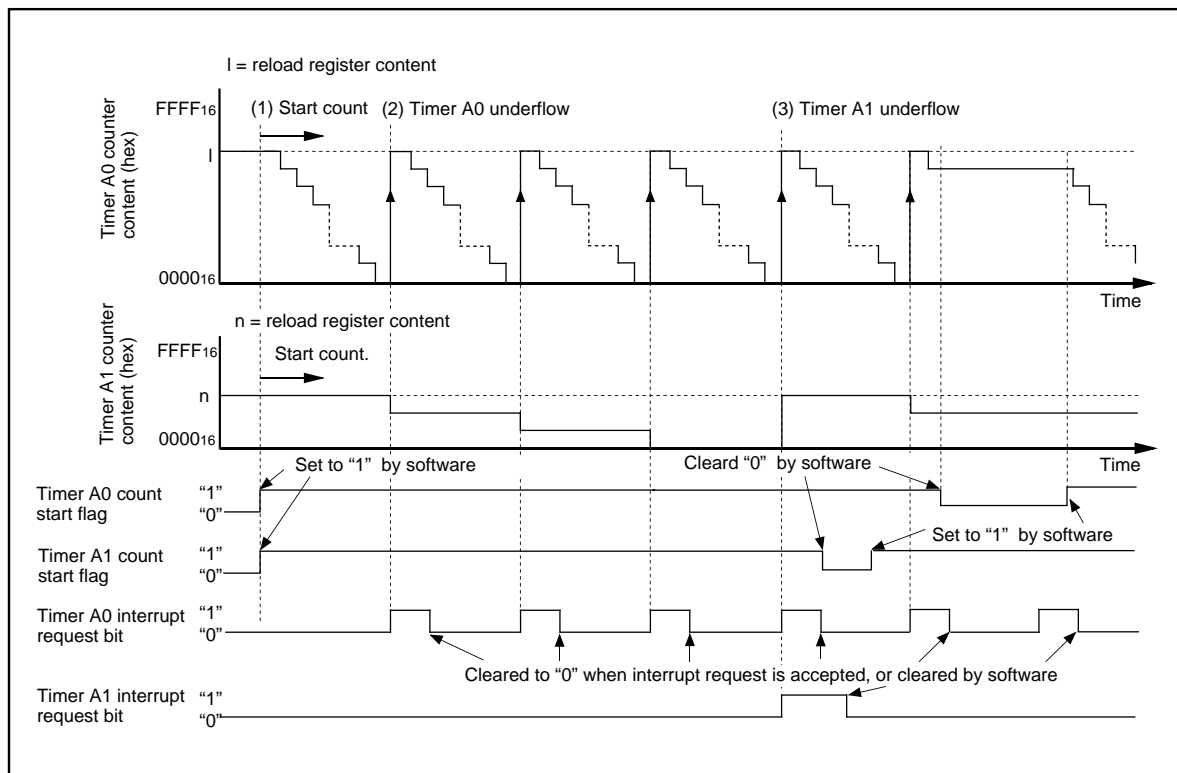


Figure 3.1.1. Operation timing of long-period timers



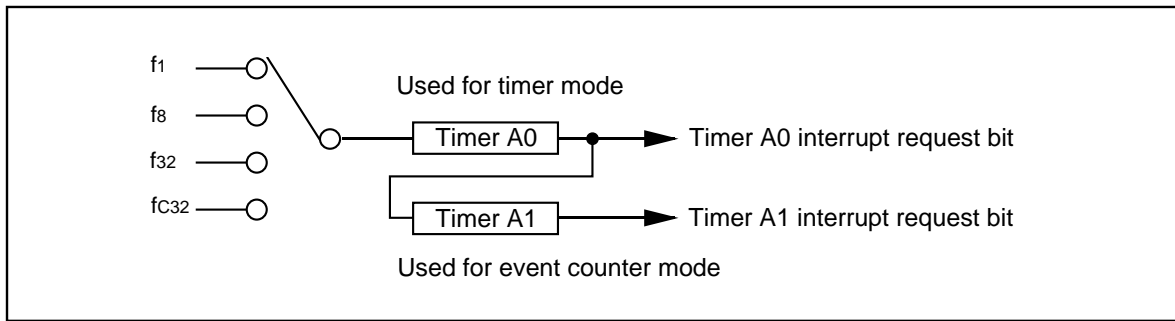


Figure 3.1.2. Connection diagram of long-period timers

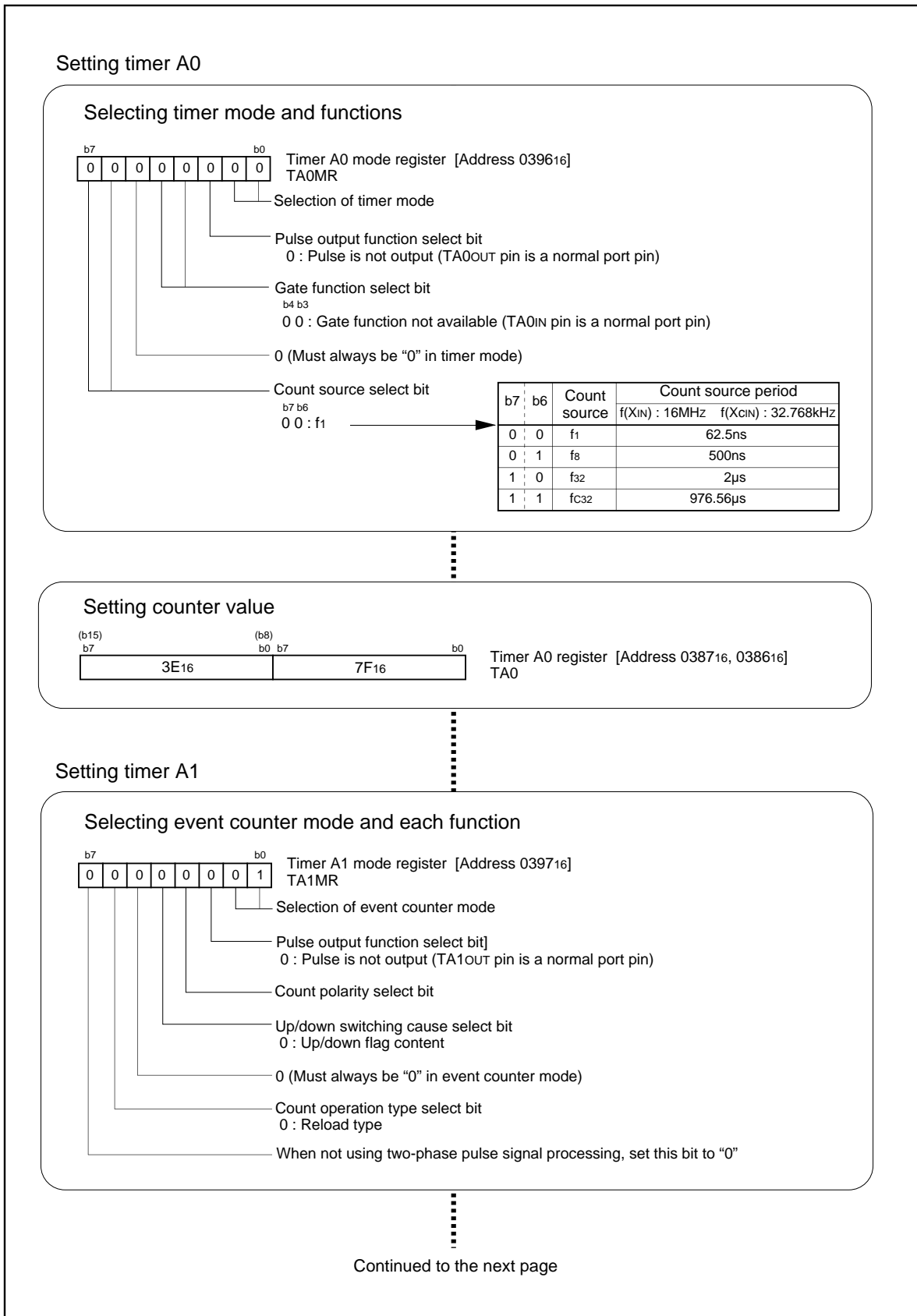
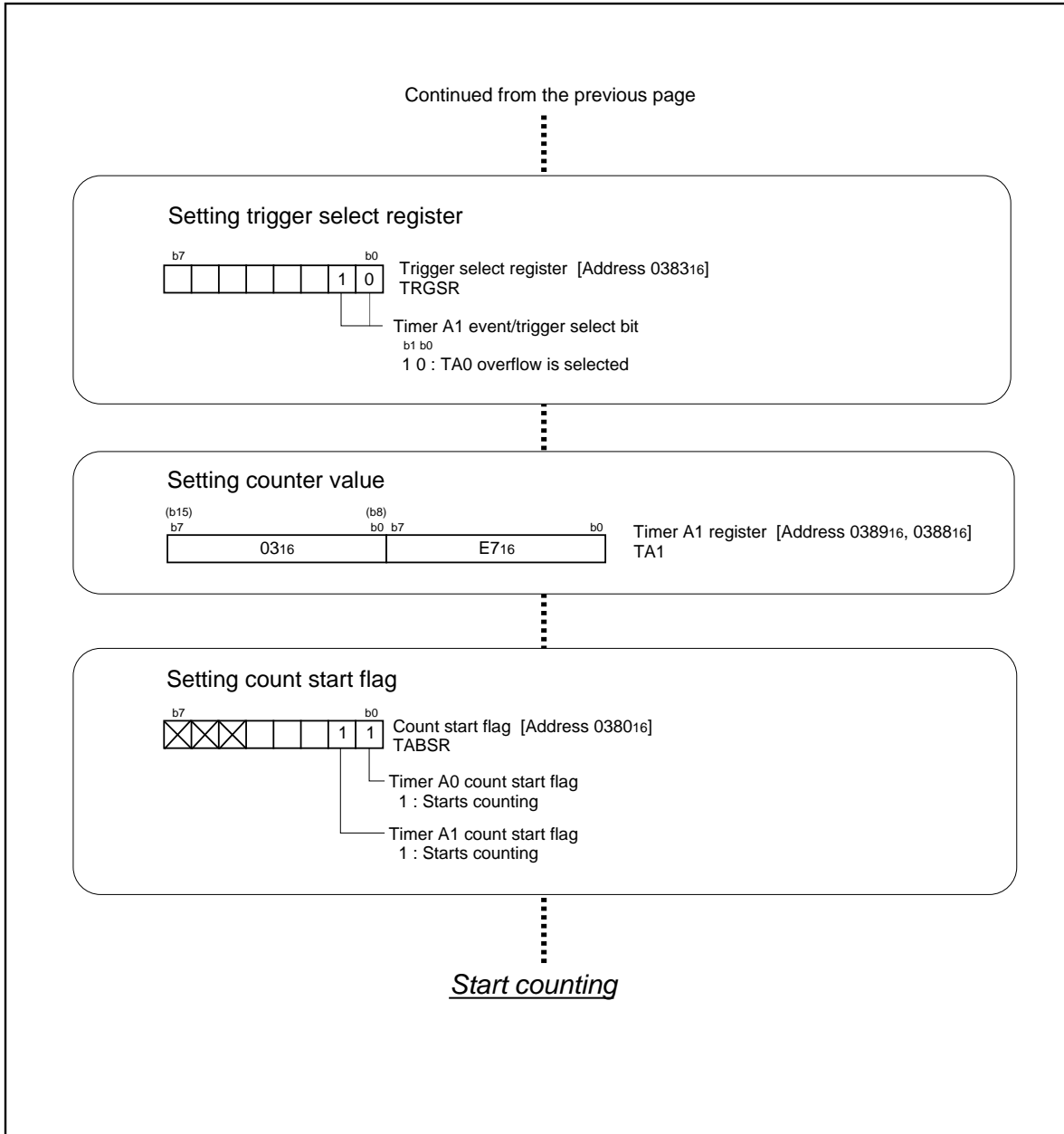


Figure 3.1.3. Set-up procedure of long-period timers (1)



**Figure 3.1.4. Set-up procedure of long-period timers (2)**

### 3.2 Variable-Period Variable-Duty PWM Output

**Overview** In this process, Timer A0 and A1 are used to generate variable-period, variable-duty PWM output. Figure 3.2.1 shows the operation timing, Figure 3.2.2 shows the connection diagram, and Figures 3.2.3 and 3.2.4 show the set-up procedure.

Use the following peripheral functions:

- Timer mode of timer A
- One-shot timer mode of timer A

#### Specifications

- (1) Set timer A0 in timer mode, and set timer A1 in one-shot timer mode with pulse-output function.
- (2) Set 1 ms, the PWM period, to timer A0. Set 500  $\mu$ s, the width of PWM "H" pulse, to timer A1. Both timer A0 and timer A1 use f1 for the count source.
- (3) Connect a 16-MHz oscillator to XIN.

**Operation**

- (1) Setting the count start flag to "1" causes the counter of timer A0 to begin counting. The counter of timer A0 performs a down count on count source f1.
- (2) If the counter of timer A0 underflows, the counter reloads the content of the reload register and continues counting. At this time, the timer A0 interrupt request bit goes to "1".
- (3) An underflow in timer A0 triggers the counter of timer A1 and causes it to begin counting. When the counter of timer A1 begins counting, the output level of the TA1OUT pin goes to "H".
- (4) As soon as the count of the counter of timer A1 becomes "0000<sub>16</sub>", the output level of TA1OUT pin goes to "L", and the counter reloads the content of the reload register and stops counting. At the same time, the timer A1 interrupt request bit goes to "1".

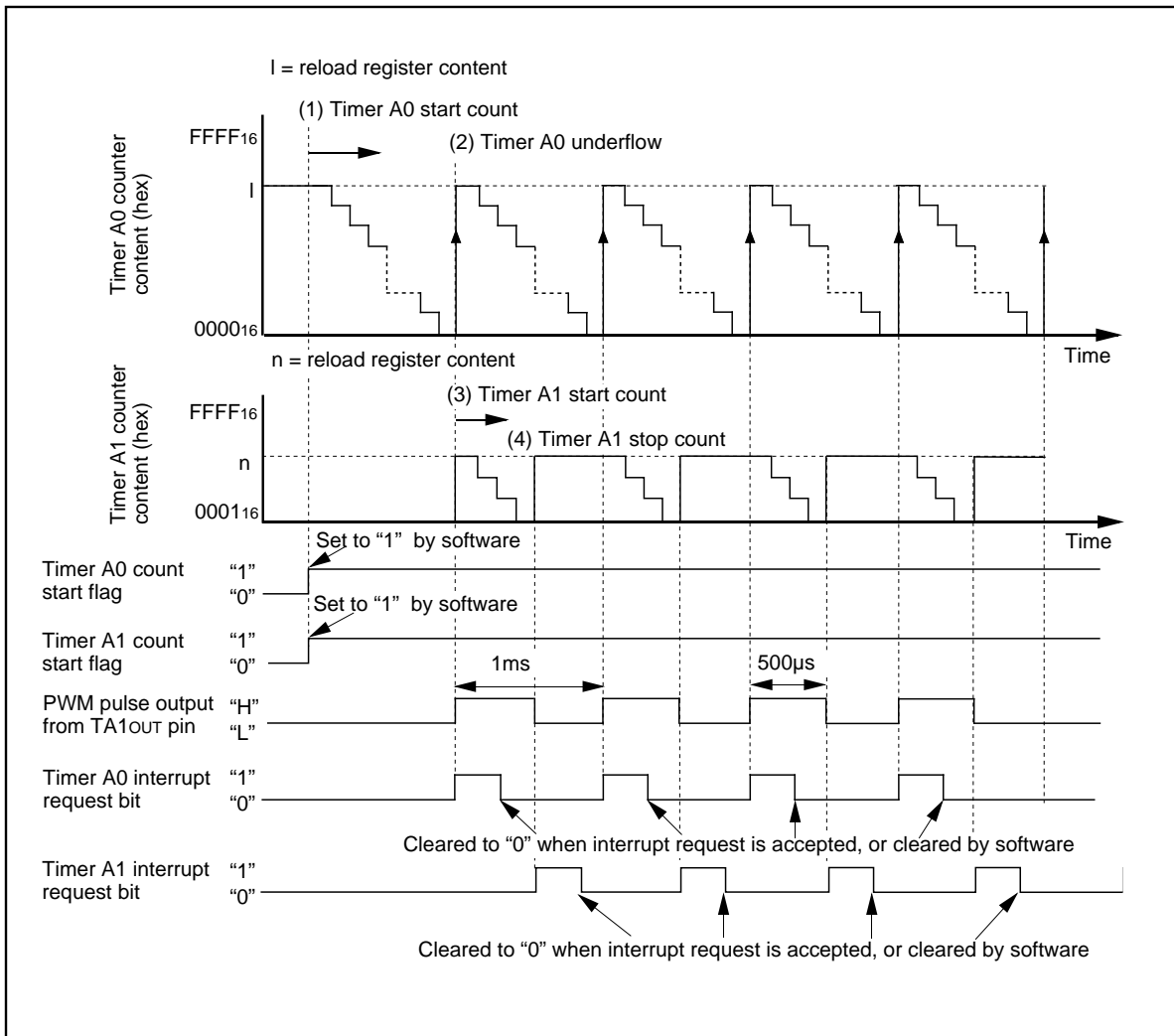


Figure 3.2.1. Operation timing of variable-period variable-duty PWM output

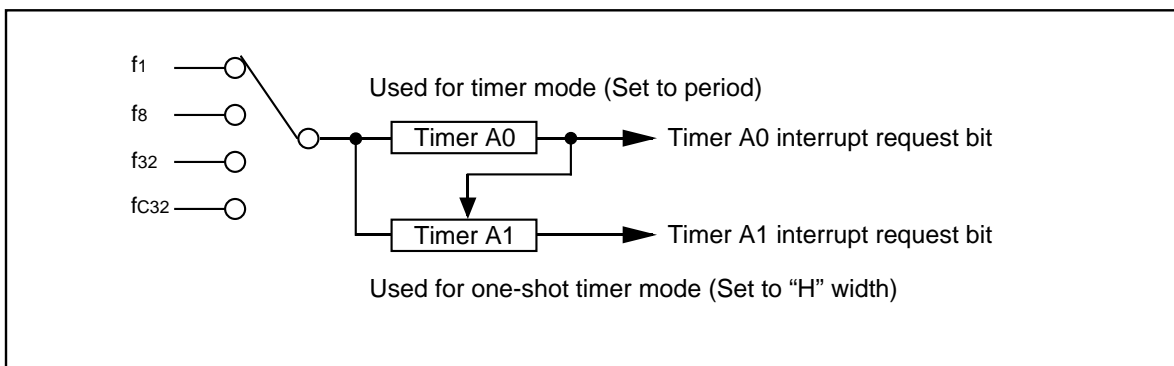


Figure 3.2.2. Connection diagram of variable-period variable-duty PWM output

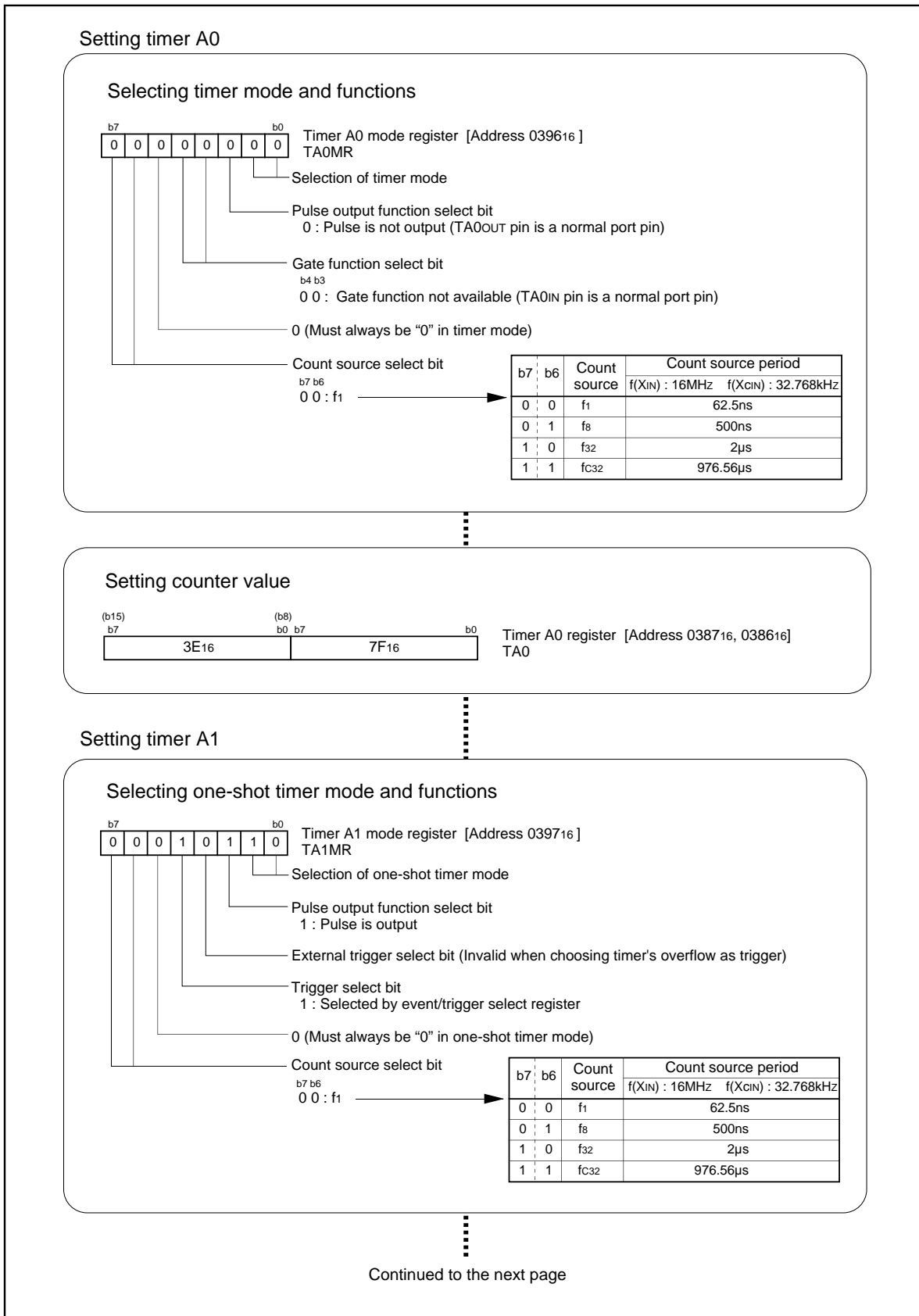


Figure 3.2.3. Set-up procedure of variable-period variable-duty PWM output (1)

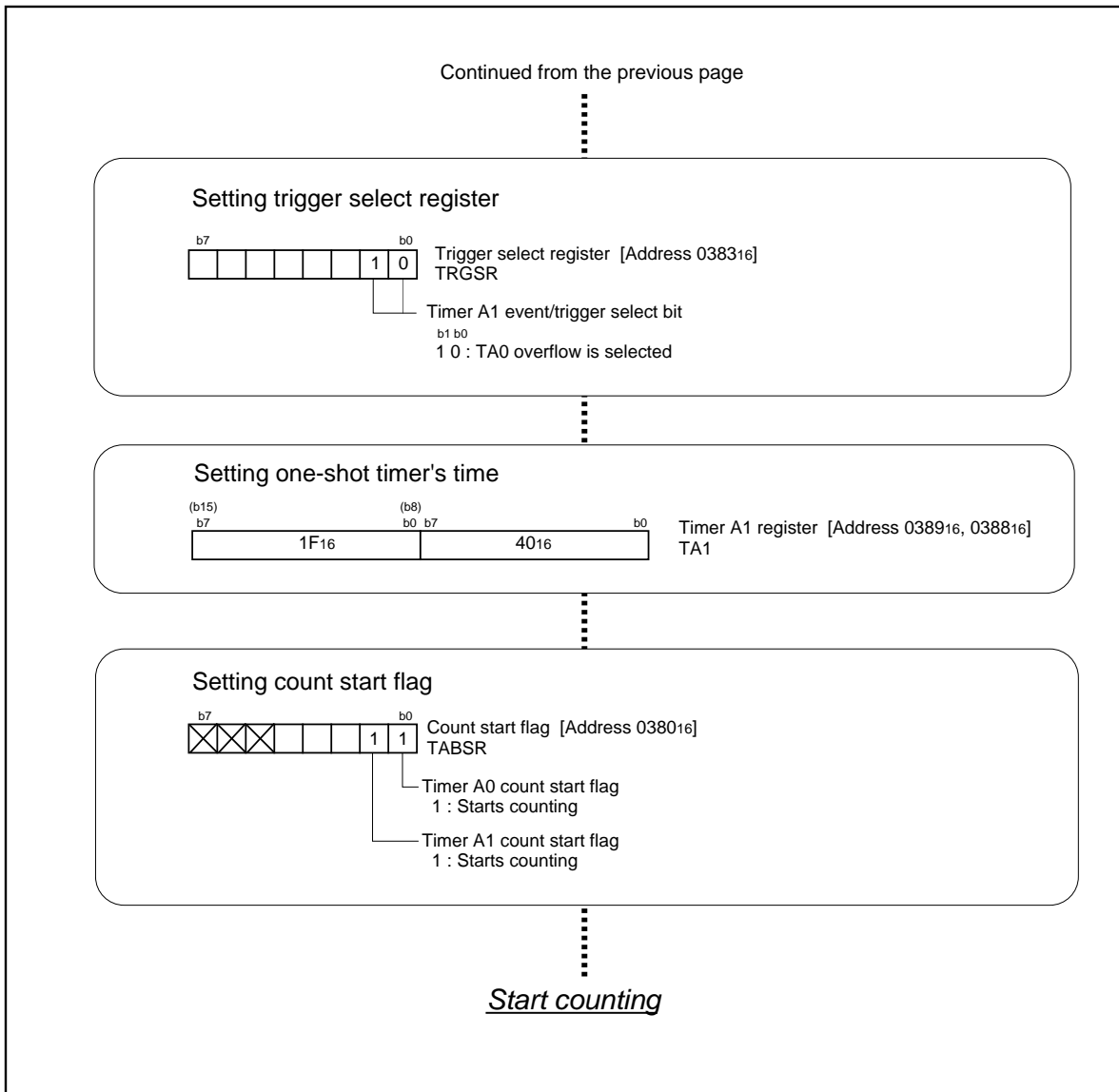


Figure 3.2.4. Set-up procedure of variable-period variable-duty PWM output (2)

### 3.3 Buzzer Output

**Overview** The timer mode is used to make the buzzer ring. Figure 3.3.1 shows the operation timing, and Figure 3.3.2 shows the set-up procedure.

Use the following peripheral function:

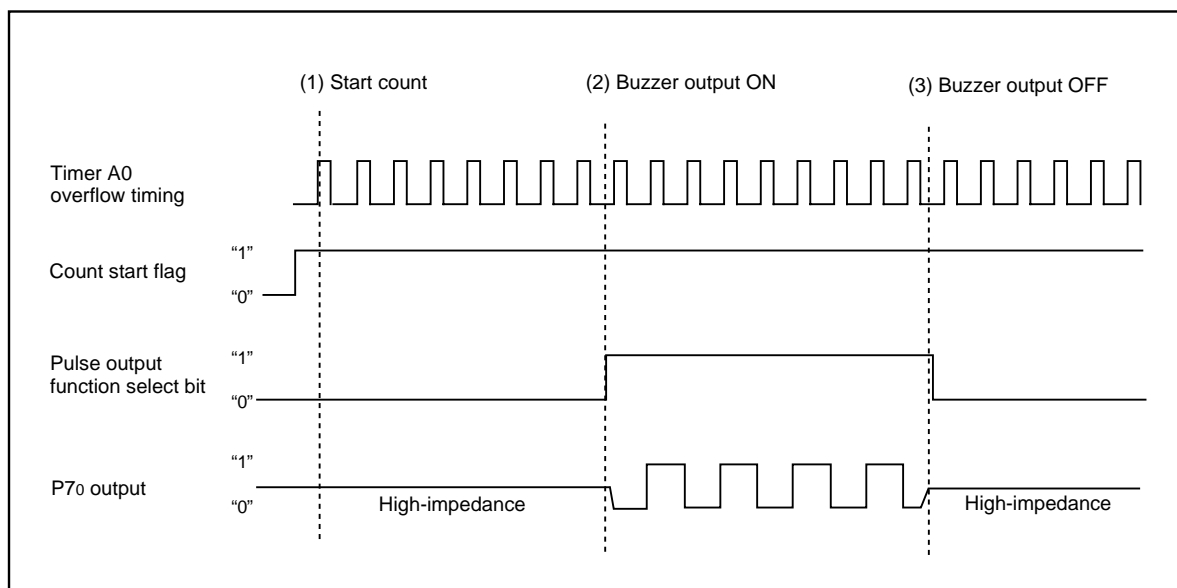
- The pulse-outputting function in timer mode of timer A.

#### Specifications

- (1) Sound a 2-kHz buzz beep by use of timer A0.
- (2) Effect pull-up in the relevant port by use of a pull-up resistor. When the buzzer is off, set the port high-impedance, and stabilize the potential resulting from pulling up.
- (3) Connect a 16-MHz oscillator to XIN.

**Operation**

- (1) The microcomputer begins performing a count on timer A0. Timer A0 has disabled interrupts.
- (2) The microcomputer begins pulse output by setting the pulse output function select bit to "Pulse output effected". P70 changes into TA0OUT pin and outputs 2-kHz pulses.
- (3) The microcomputer stops outputting pulses by setting the pulse output function select bit to "Pulse output not effected". P70 goes to an input pin, and the output from the pin becomes high-impedance.



**Figure 3.3.1. Operation timing of buzzer output**



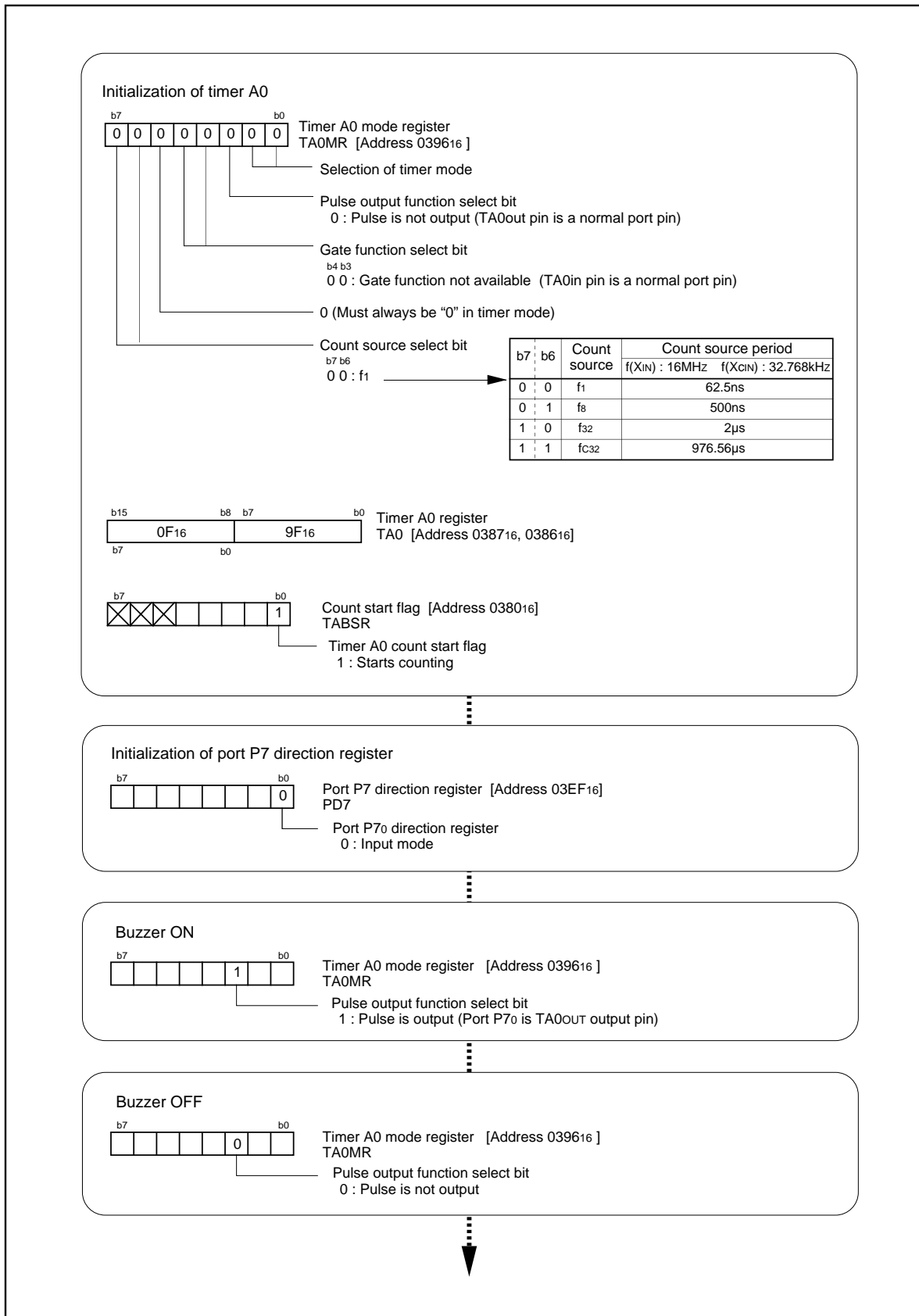


Figure 3.3.2. Set-up procedure of buzzer output

### 3.4 Solution for External Interrupt Pins Shortage

**Overview** The following are solution for external interrupt pins shortage. Figure 3.4.1 shows the set-up procedure.

Use the following peripheral function:

- Event counter mode of timer A

**Specifications**

(1) Inputting a falling edge to the TA0IN pin generates a timer A0 interrupt.

**Operation** (1) Set timer A0 to event counter mode, set timer to "0", and set interrupt priority levels in timer A0.

(2) Inputting a falling edge to the TA0IN pin generates a timer A0 interrupt.

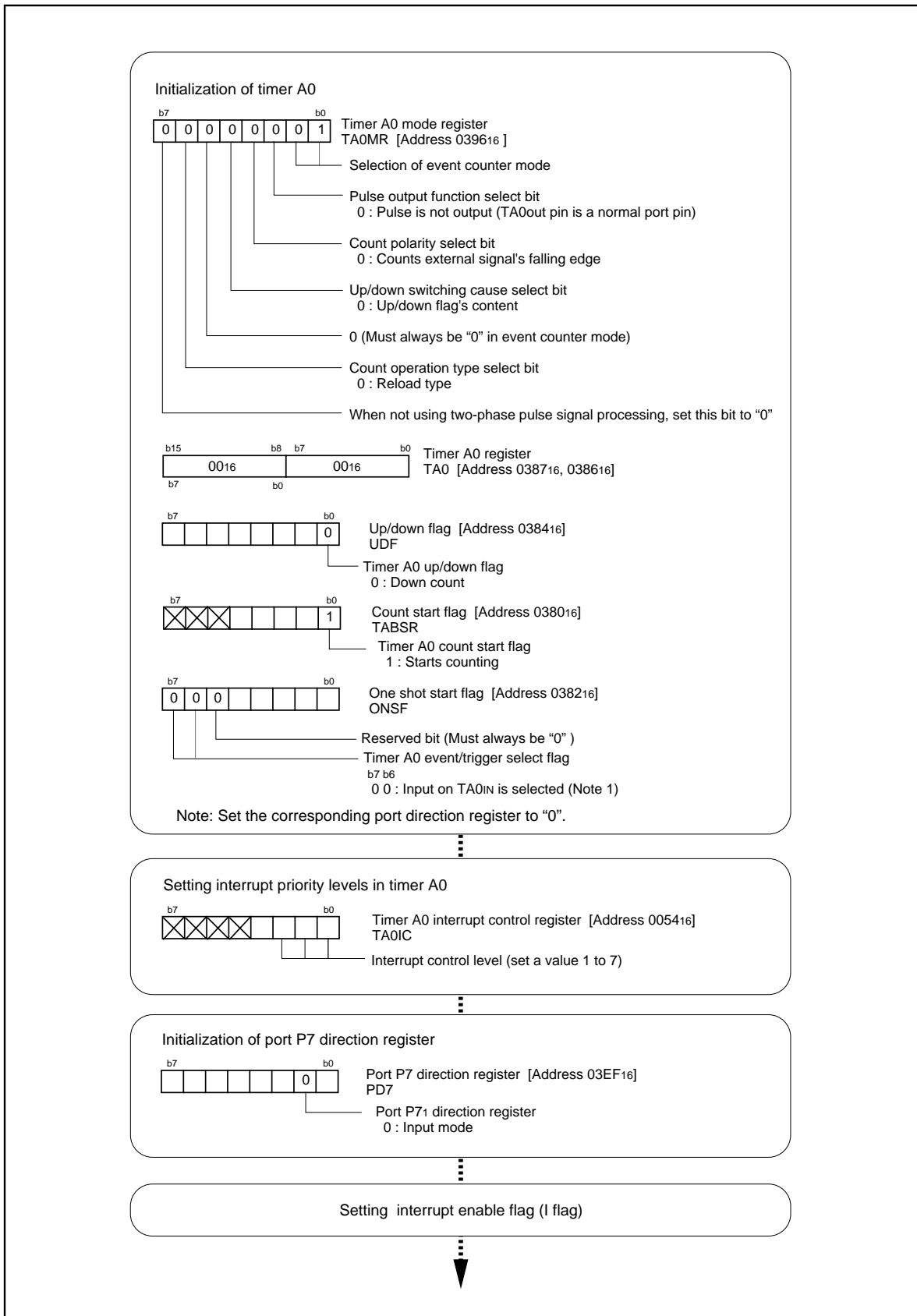


Figure 3.4.1. Set-up procedure of solution for a shortage of external interrupt pins

### 3.5 Memory to Memory DMA Transfer

**Overview** The following are steps for changing both source address and destination address to transfer data from memory to another. The DMA transfer utilizes the workings that assign a higher priority to the DMA0 transfer if transfer requests simultaneously occur in two DMA channels. Figure 3.5.1 shows the operation timing, Figure 3.5.2 shows the block diagram, and Figures 3.5.3 and 3.5.4 show the set-up procedure.

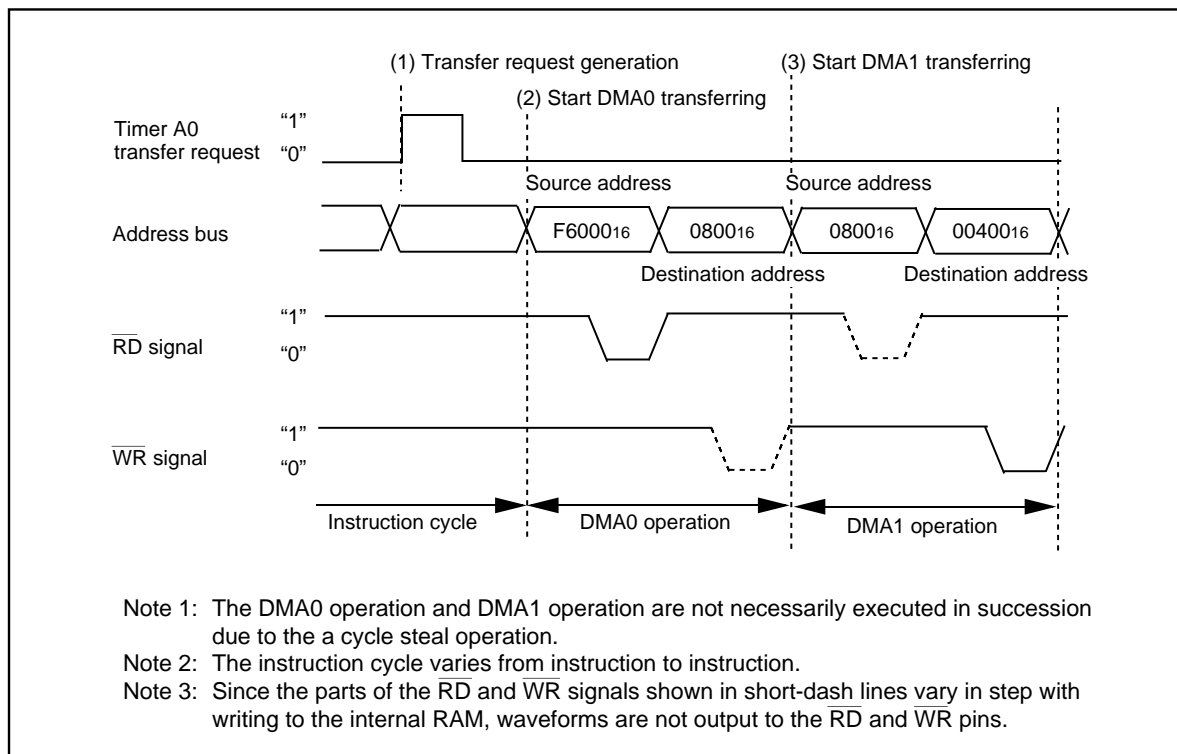
Use the following peripheral functions:

- Timer mode of timer A
- Two DMAC channels
- One-byte temporary RAM (address 0800<sub>16</sub>)

#### Specifications

- (1) Transfer the content of memory extending over 128 bytes from address F6000<sub>16</sub> to a 128-byte area starting from address 00400<sub>16</sub>. Transfer the content every time a timer A0 interrupt request occurs.
- (2) Use DMA0 for a transfer from the source to built-in memory, and DMA1 for a transfer from built-in memory to the destination.

- Operation**
- (1) A timer A interrupt request occurs. Though both a DMA0 transfer request and a DMA1 transfer request occur simultaneously, the former is executed first.
  - (2) DMA0 receives a transfer request and transfers data from the source to the built-in memory. At this time, the source address is incremented.
  - (3) Next, DMA1 receives a transfer request and transfers data involved from built-in memory to the destination. At this time, the destination address is incremented.



**Figure 3.5.1. Operation timing of memory to memory DMA transfer**

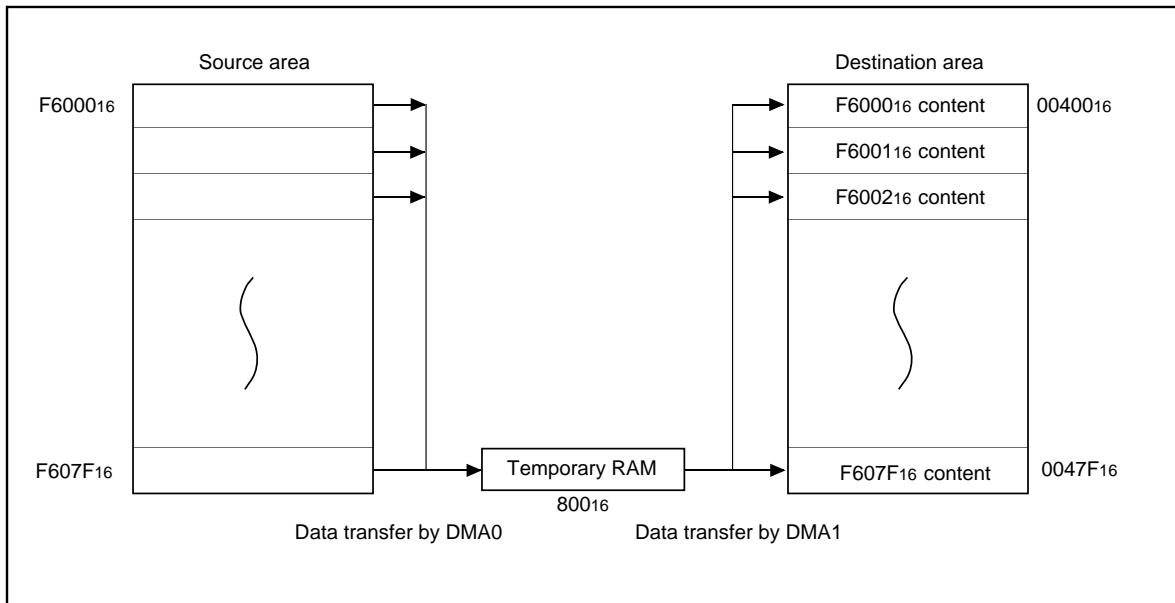


Figure 3.5.2. Block diagram of memory to memory DMA transfer

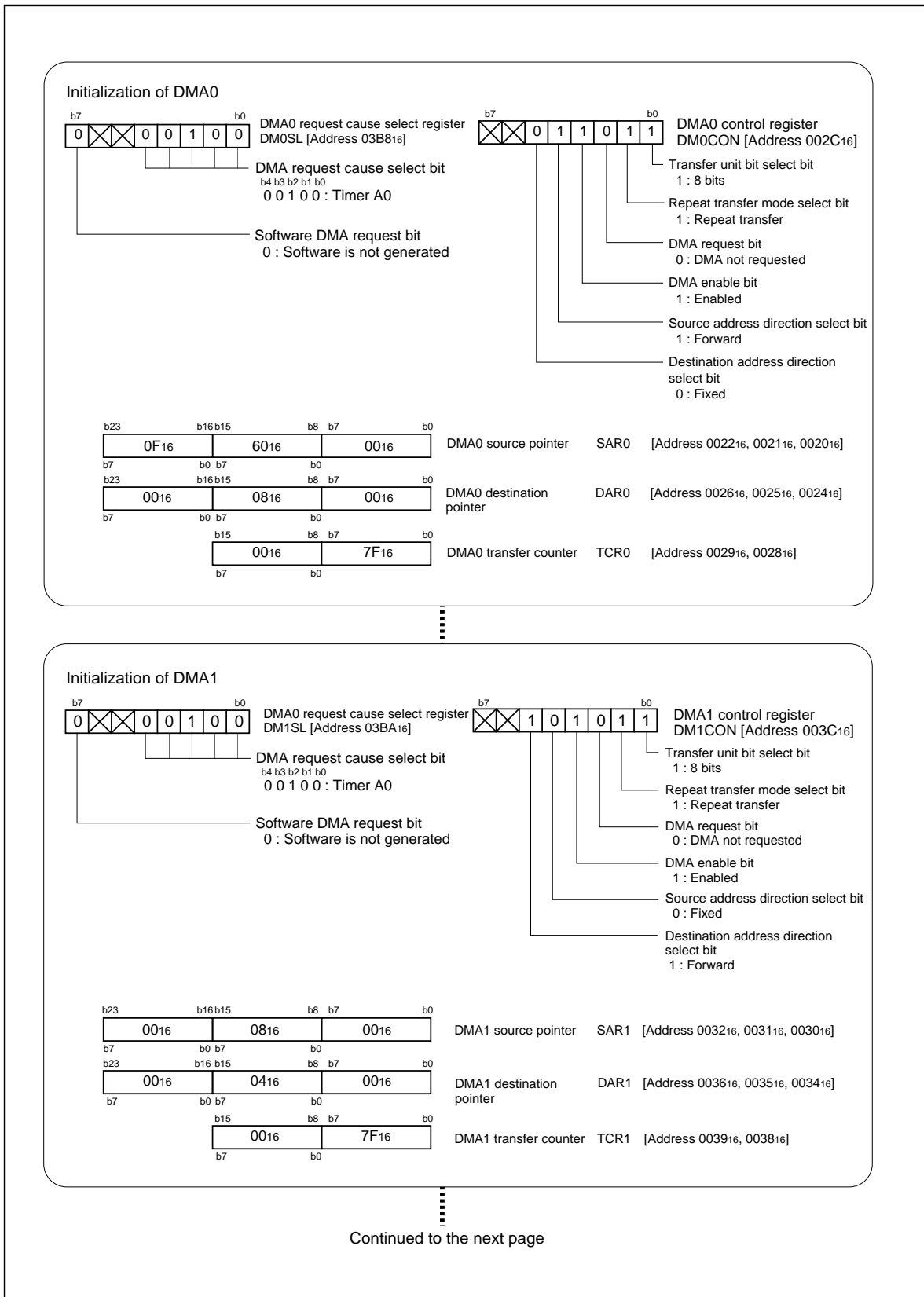


Figure 3.5.3. Set-up procedure of memory to memory DMA transfer (1)

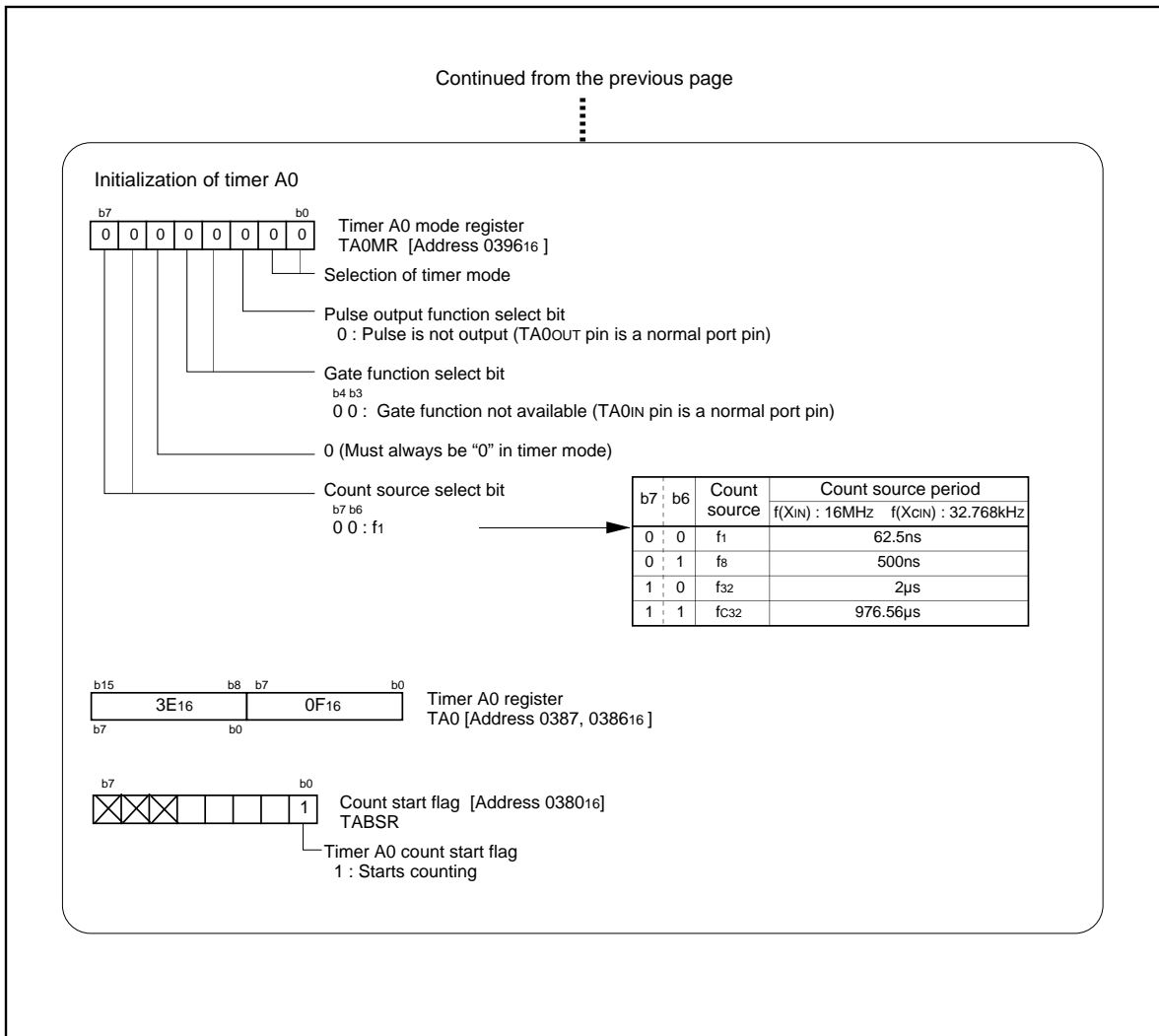


Figure 3.5.4. Set-up procedure of memory to memory DMA transfer (2)

### 3.6 CRC Calculation SFR Access Snoop Function in Clock Synchronous Serial Data Transmit

**Overview** The M30245 group, by use of DMAC, transfers data from the internal RAM to the UART1 and the result is transferred to the UART1 by use of SFR access snoop function. The block diagram is shown in Figure 3.6.1 and the setting routine is shown in Figure 3.6.2 to Figure 3.6.4.

The peripheral functions to be used are as follows:

- DMAC 1 Channel
- Internal RAM (address 00400<sub>16</sub>) 512 bytes
- UART1 (Clock synchronous serial I/O mode)
- CRC calculation circuit
- SFR access snoop function

#### Specifications

- (1) Data transfer is performed starting at address 00400<sub>16</sub> from the area with 512 bytes to the UART1. Data are transferred from area between the address 00400<sub>16</sub> and the 512nd byte to the UART1. Transfer is executed every time 1 byte of serial transmit is completed.
- (2) Use the DMA0 to transfer data from the internal RAM to the UART1. Select the UART1 transmit to the DMA0 request factor. Select the single transfer mode and set the DMA0 transfer counter to 511 bytes (512-1).
- (3) Set the CRC calculation circuit to the CRC-CCITT and set CRC snoop address register to the address of UART1 transmit buffer register (write snoop).
- (4) On completing the DMA, 2-byte data of CRC data register (calculation result) are transferred to the UART1 and operation is completed.

#### Operation

- (1) Initialize the UART1 related registers.
- (2) Initialize the DMA0 related registers in DMA disable state.
- (3) Set the DMA0 transfer counter to the transfer data consisting of 511 bytes (in this case, 8-bit transfer).
- (4) Initialize the CRC calculation circuit and the SFR access snoop function.
- (5) Set the software DMA request bit of DMA0 to "1". At this time, 1st byte data are transferred from RAM to the transmit buffer of the UART1. Simultaneously, the transfer source address is incremented and the content of the transfer counter is down-counted. The transferred data are automatically written in CRC input register by the SFR access snoop function.
- (6) When the transmit buffer of the UART1 becomes writable state, the DMA transfer request is occurred by the UART1. At this time, the next data are transferred from RAM to the transmit buffer of the UART1. Simultaneously, the transfer source address is incremented and the content of the transfer counter is down-counted. The transferred data are automatically written in CRC input register by the SFR access snoop function.
- (7) As a result of repetition of the above (6), when the DMA0 transfer counter underflow, DMA enable bit is set to "0" to complete the DMA0 transfer. Simultaneously, the DMA0 interrupt request occurs. When the DMA0 interrupt request is detected, CRC data register (2 bytes) is read, it is transferred to the UART1 transmit buffer sequentially.



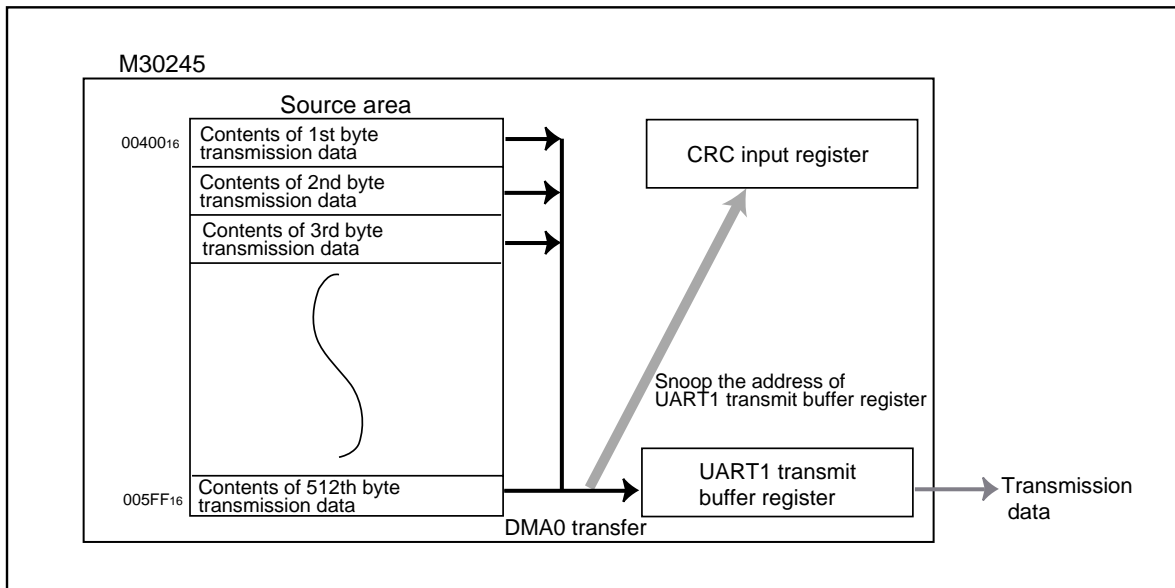


Figure 3.6.1. Block diagram of DMA transfer from RAM to UART and SFR snooping function

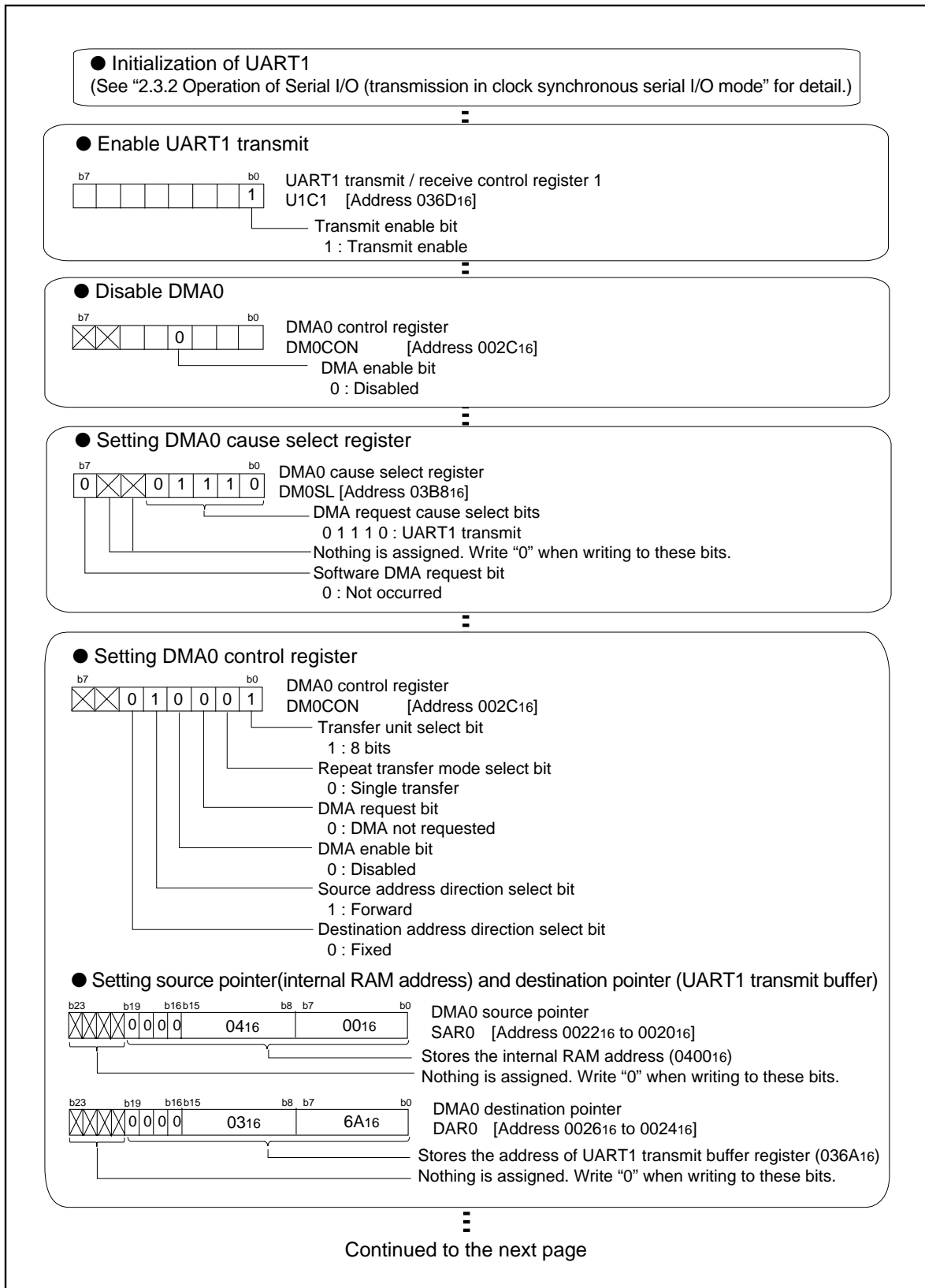


Figure 3.6.2. Setting routine (1) of DMA transfer from RAM to UART using SFR snooping function

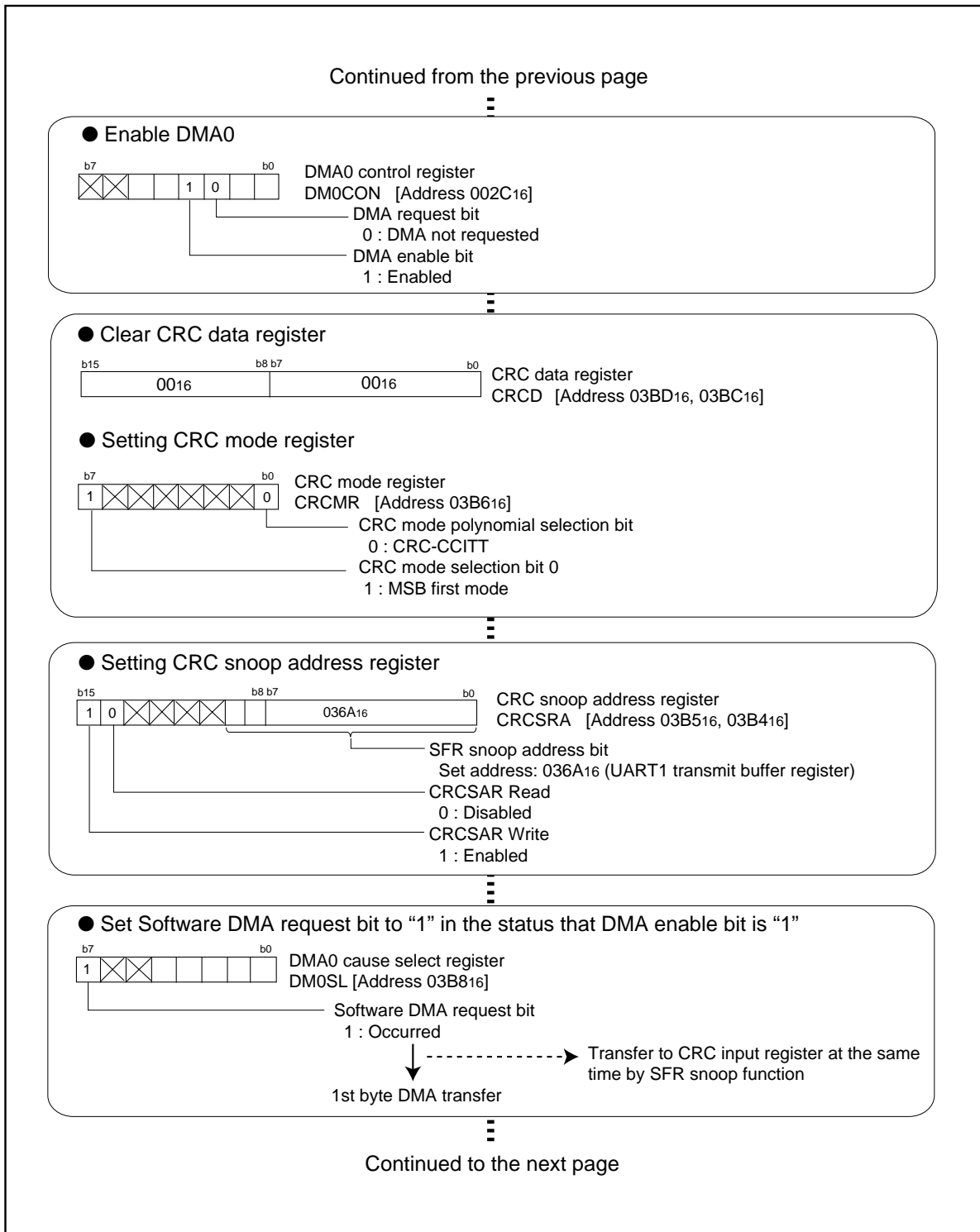
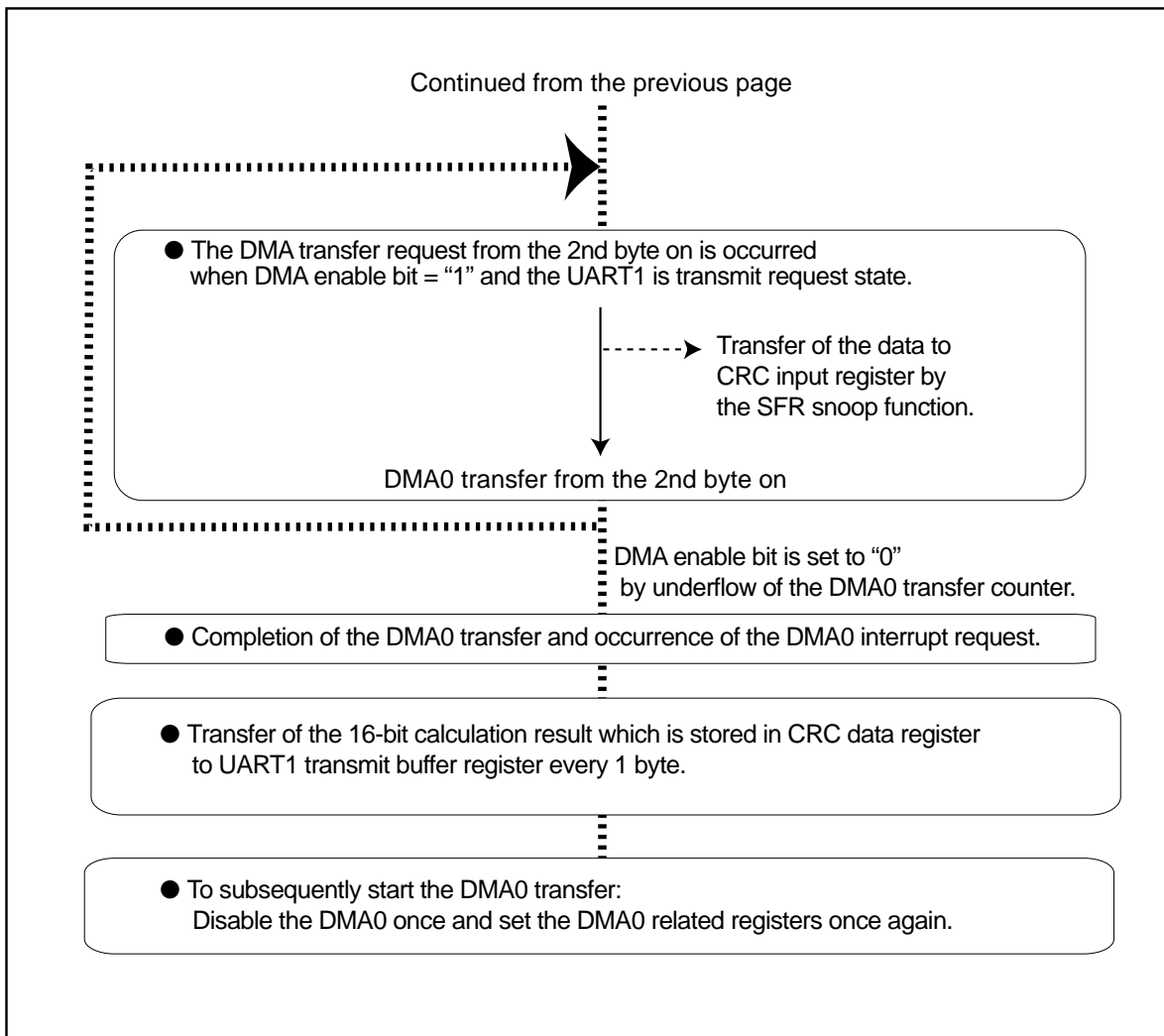


Figure 3.6.3. Setting routine (2) of DMA transfer from RAM to UART using SFR snooping function



**Figure 3.6.4. Setting routine (3) of DMA transfer from RAM to UART using SFR snooping function**

### 3.7 Transfer from USB FIFO to Serial Sound Interface

**Overview** The M30245 group, by use of DMAC, transfers data from the USB endpoint 1 OUT FIFO to SS interface 1 transmit buffer register and fetches one packet data.

The block diagram is shown in Figure 3.7.1 and the setting routine is shown in Figure 3.7.2 to Figure 3.7.4.

The peripheral functions to be used are as follows:

- DMAC 1 channel
- USB endpoint 1 OUT (Receive)
- Serial sound interface 1

#### Specifications

- (1) Receive packet data of the endpoint 1 OUT FIFO are transferred to SS interface 1 transmit buffer register. Transfer is executed every time the DMA transfer factor of the serial sound interface 1 occurs.
- (2) Use the DMA0 to transfer data from the endpoint 1 OUT FIFO to SS interface 1 transmit buffer register. Select the serial sound interface 1 transmit to the DMA0 request factor. Select the single transfer mode and set the DMA0 transfer counter to  $1/2 \times$  (the data count of one packet received with endpoint 1 OUT)  $-1$ .
- (3) Set the endpoint 1 OUT maximum packet size to 288 bytes (when sampling 48KHz/ 24-bit/ stereo) and disable the AUTO\_CLR function. The data count of receive packet of endpoint 1 (endpoint 1 OUT write count register) is set to 288 bytes. Endpoint 1 OUT is used in isochronous transfer.
- (4) On completing the DMA0 transfer, fetch of one packet data from the endpoint 1 OUT FIFO is completed by setting CLR\_OUT\_BUF\_RDY bit of endpoint 1 to "1".

#### Operation

- (1) Initialize the DMA0 related registers in the state which DMA is disabled and USB DMA0 request register is not selected (in this case, 16-bit transfer).
- (2) When the OUT\_BUF\_STS1 flag of endpoint 1 is set to "1" and packet data receive has been detected, set the DMA0 transfer counter to the  $1/2 \times$  (the data count of receive one packet)  $-1$  (in this application example, 143 value is set).
- (3) Set DMA enable bit of DMA0CON to "1" (DMA0 is enabled). Then, the DMA0 transfer request from the serial sound interface occurs.
- (4) When the transfer request is received, the DMA0 transfers the 1st word (16-bit) data from the endpoint 1 OUT FIFO to the serial sound interface 1. Simultaneously, the content of the transfer counter is down-counted. Then, the DMA0 transfer request from the serial sound interface occurs .
- (5) As a result of repetition of the above (4), when the DMA0 transfer counter underflow, DMA enable bit is set to "0" to complete the DMA0 transfer. Simultaneously, the DMA0 interrupt request occurs. When the DMA0 interrupt request is detected, set CLR\_OUT\_BUF\_RDY bit of endpoint 1 OUT to "1".

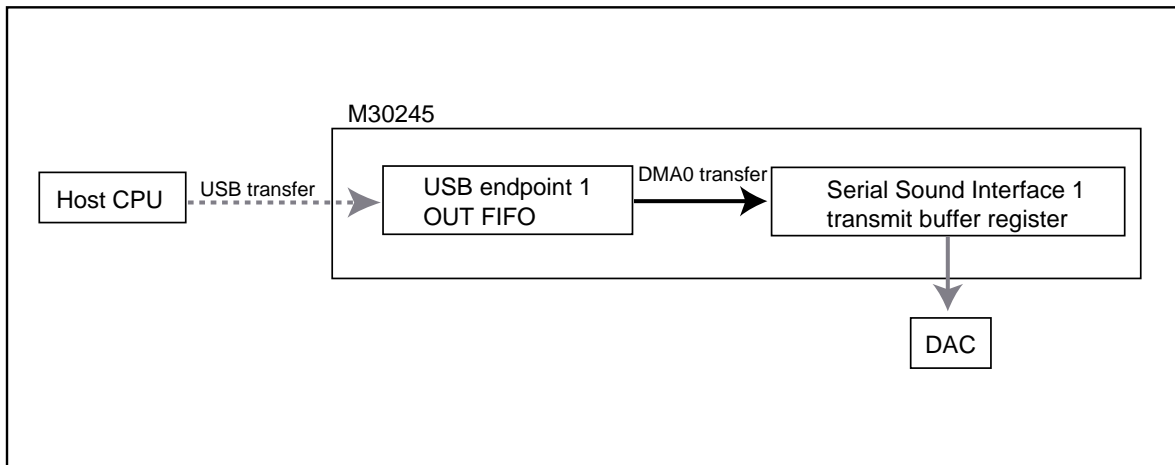


Figure 3.7.1. Block diagram of DMA transfer from USB FIFO to serial sound interface

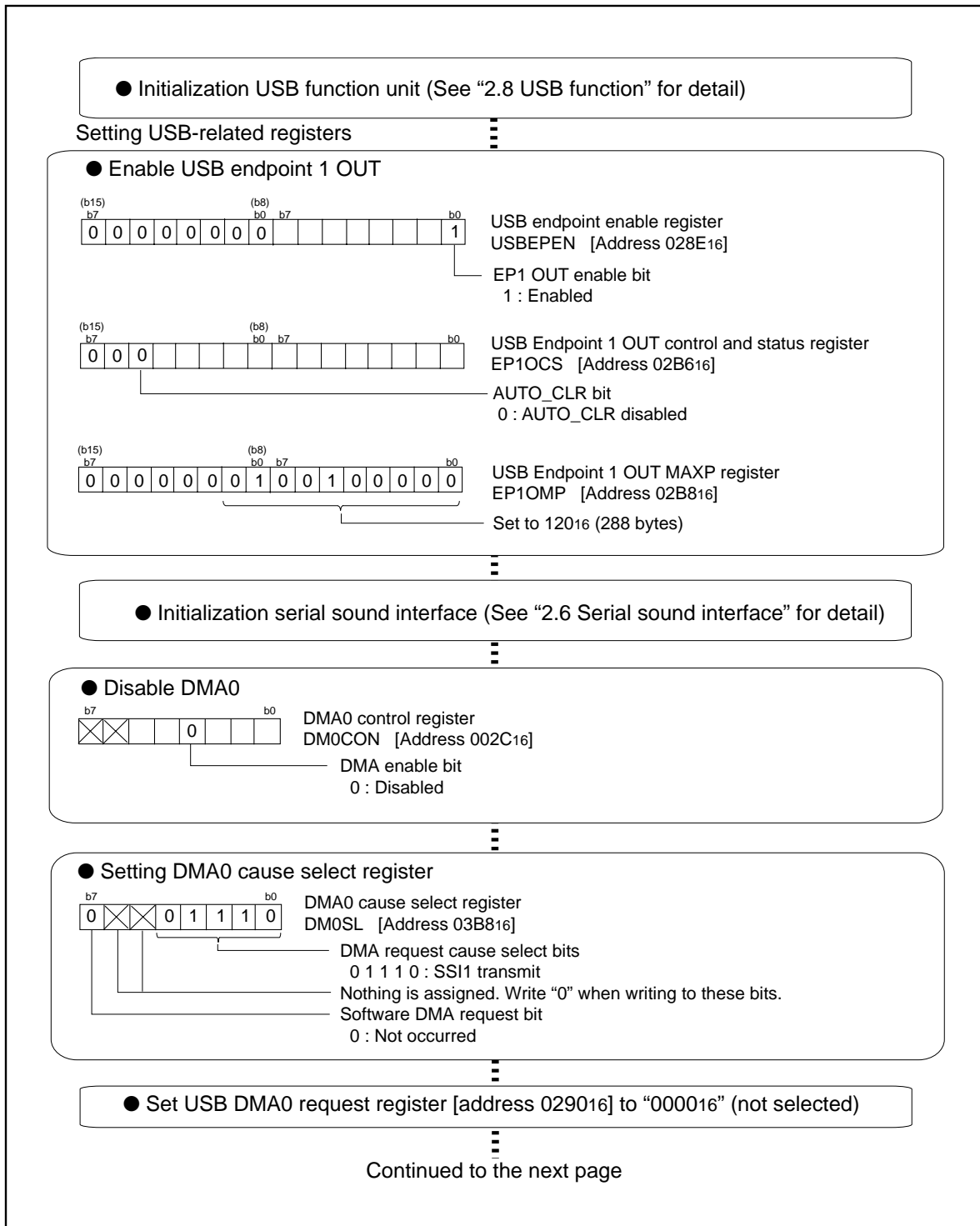


Figure 3.7.2. Setting routine (1) of DMA transfer from USB OUT FIFO to serial sound interface

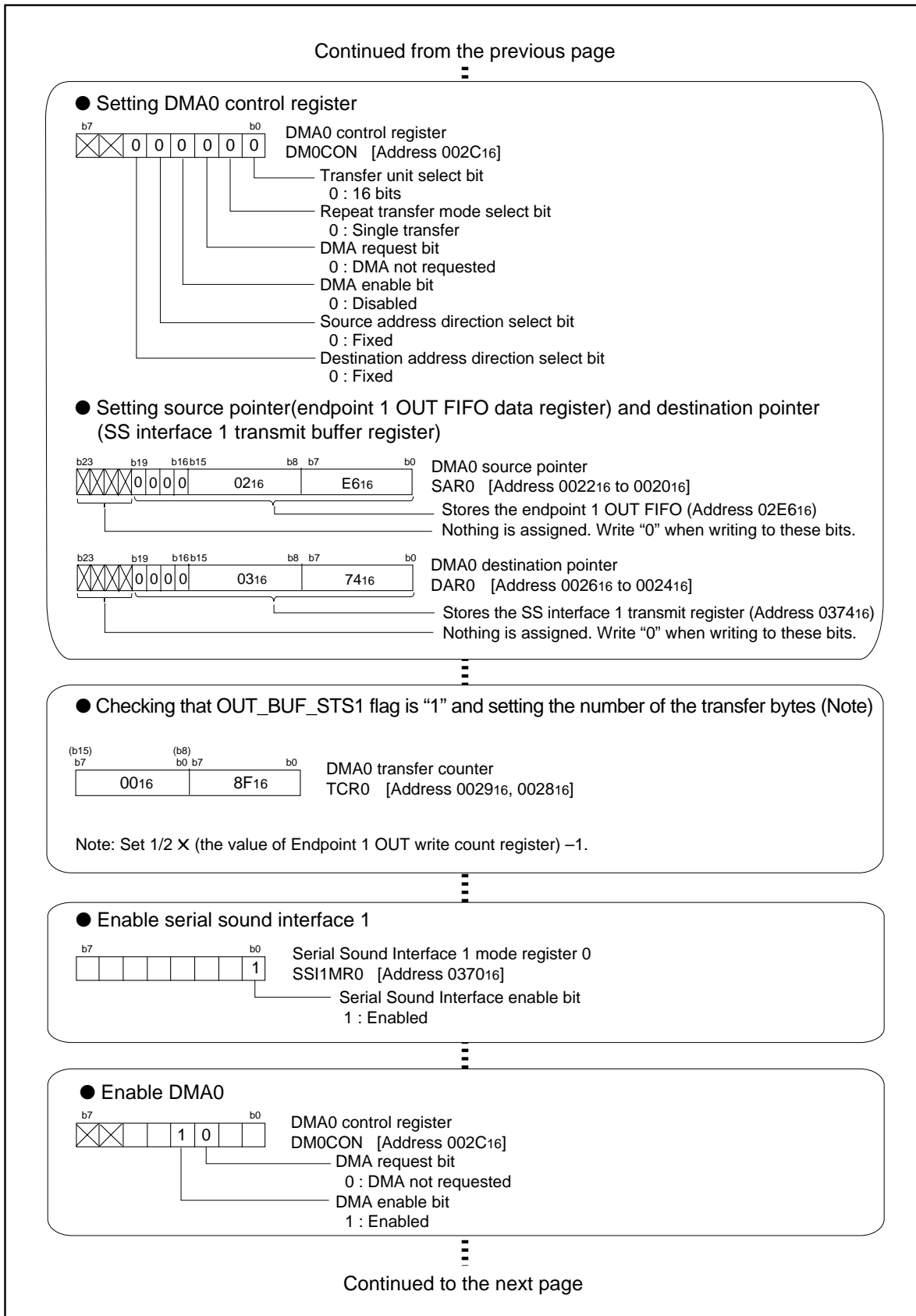


Figure 3.7.3. Setting routine (2) of DMA transfer from USB OUT FIFO to serial sound interface



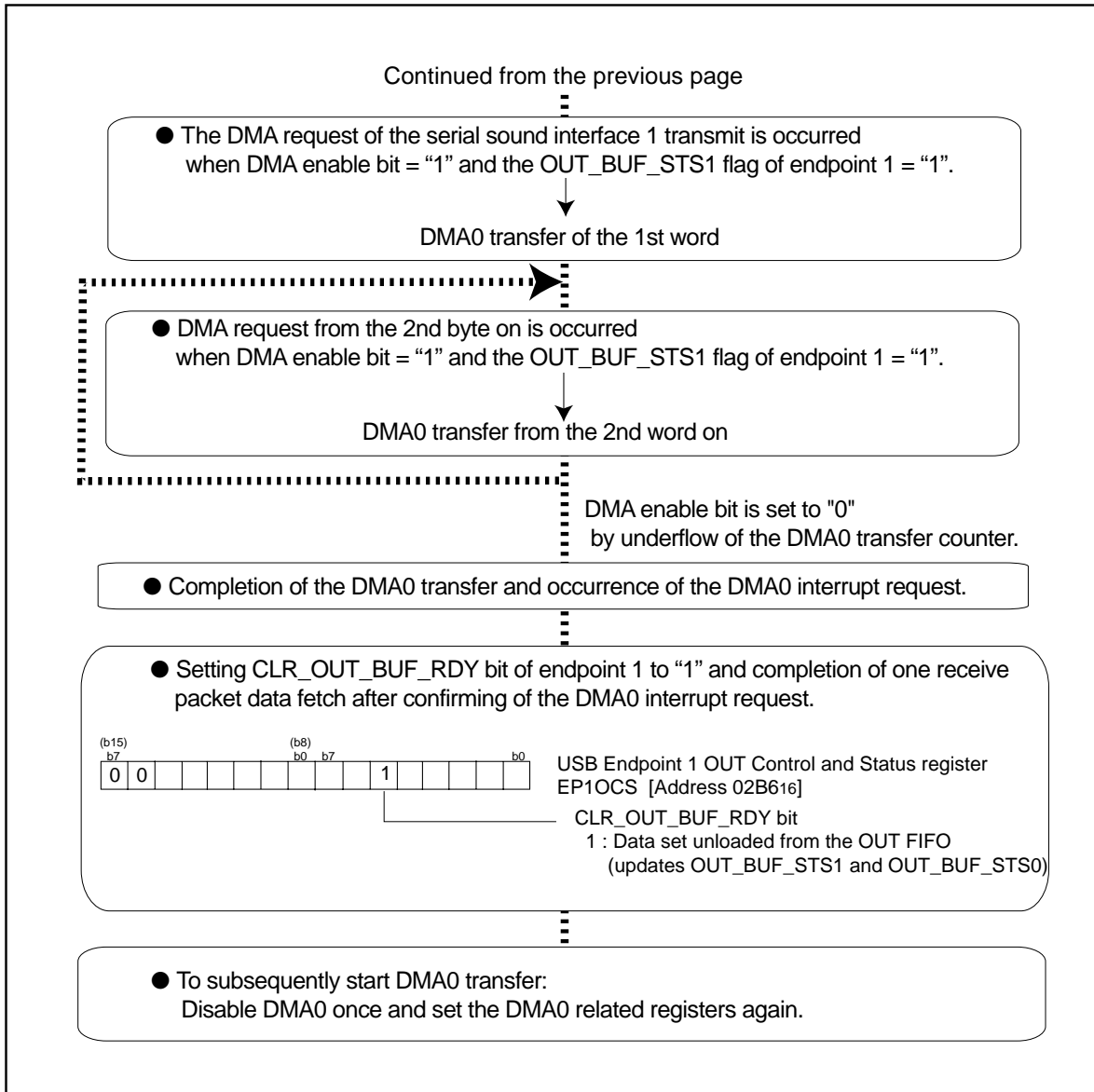


Figure 3.7.4. Setting routine (3) of DMA transfer from USB OUT FIFO to serial sound interface

### 3.8 Controlling Power Using Stop Mode

**Overview** The following are steps for controlling power using stop mode. Figure 3.8.1 shows the operation timing, Figure 3.8.2 shows an example of circuit, and Figures 3.8.3 and 3.8.4 show the set-up procedure.

Use the following peripheral functions:

- Key-input interrupts
- Stop mode
- Pull-up function

This example is not performed USB power control. Please refer section 2.8.4 for the power control of USB related.

#### Specifications

- (1) Use P00 through P03 for the scan output pins of a key matrix. Use the input pins ( $\overline{KI0}$  through  $\overline{KI3}$ ) of the key-input interrupt function for the key-input reading pins. The pull-up function is also used.
- (2) If a key-input interrupt request occurs, clear the stop mode and read a key.

- Operation**
- (1) Enable a key-input interrupt and set the pull-up function to pins  $\overline{KI0}$  through  $\overline{KI3}$ . Change the output of P00 through P03 to "L" and enter stop mode.
  - (2) If a key is pressed, "L" is input to one of pins  $\overline{KI0}$  through  $\overline{KI3}$  to clear stop mode. A key-input interrupt occurs to execute the key-input interrupt handling routine.
  - (3) Sequentially set P00 through P03 to "L" to determine which key was pressed.
  - (4) When the process to determine the key pressed is completed, change the output from P00 through P03 to "L" again and enter stop mode.

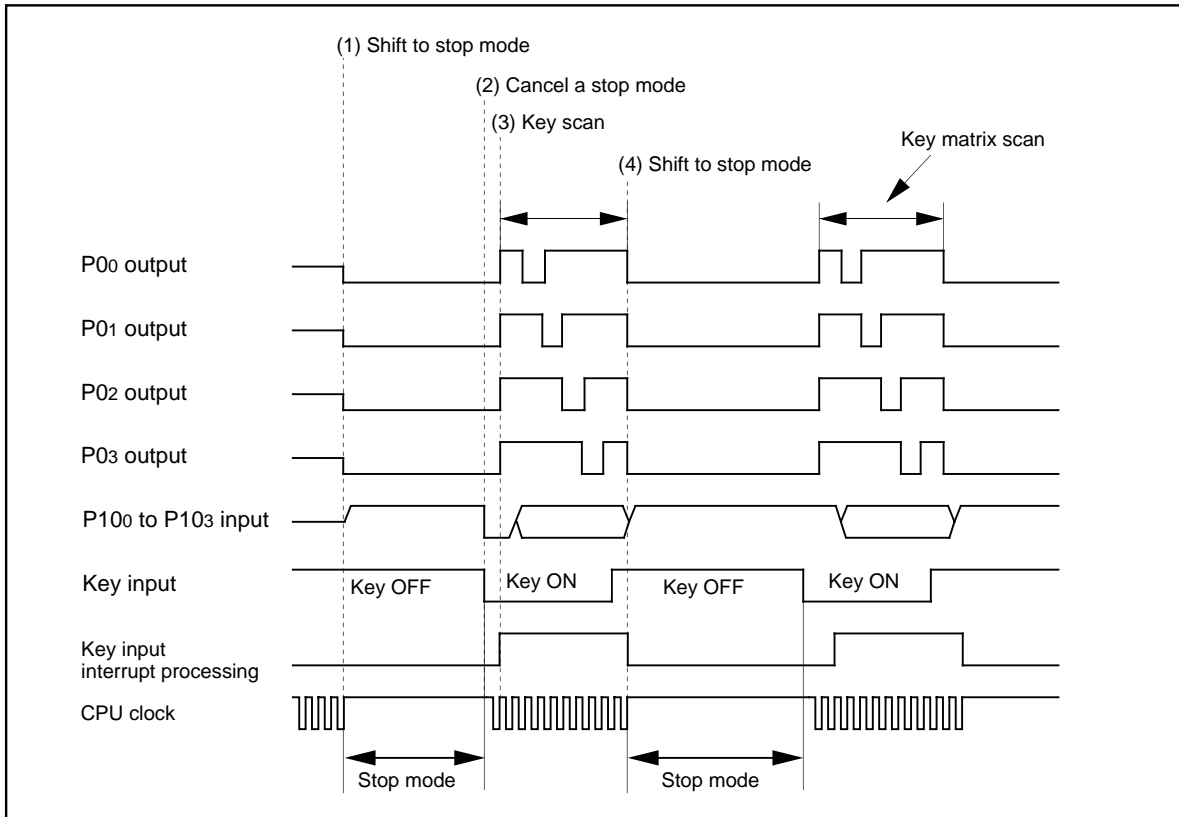


Figure 3.8.1. Operation timing of controlling power using stop mode

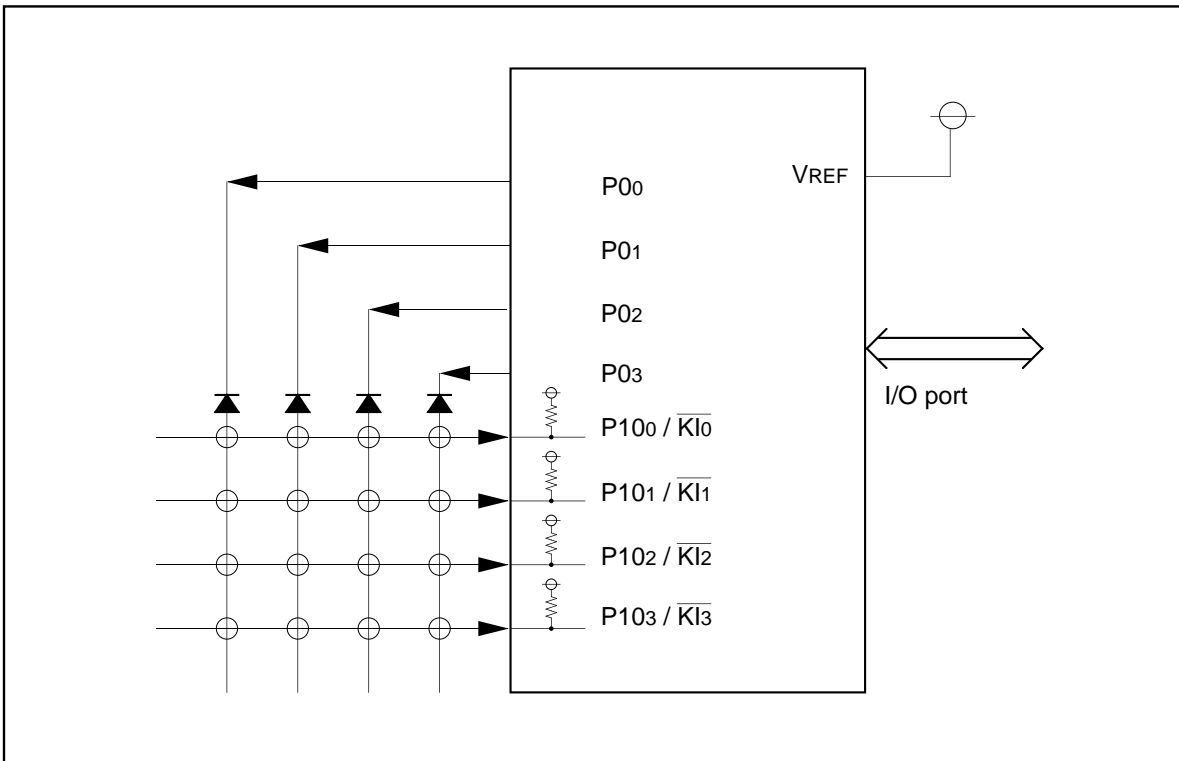


Figure 3.8.2. Example of circuit of controlling power using stop mode

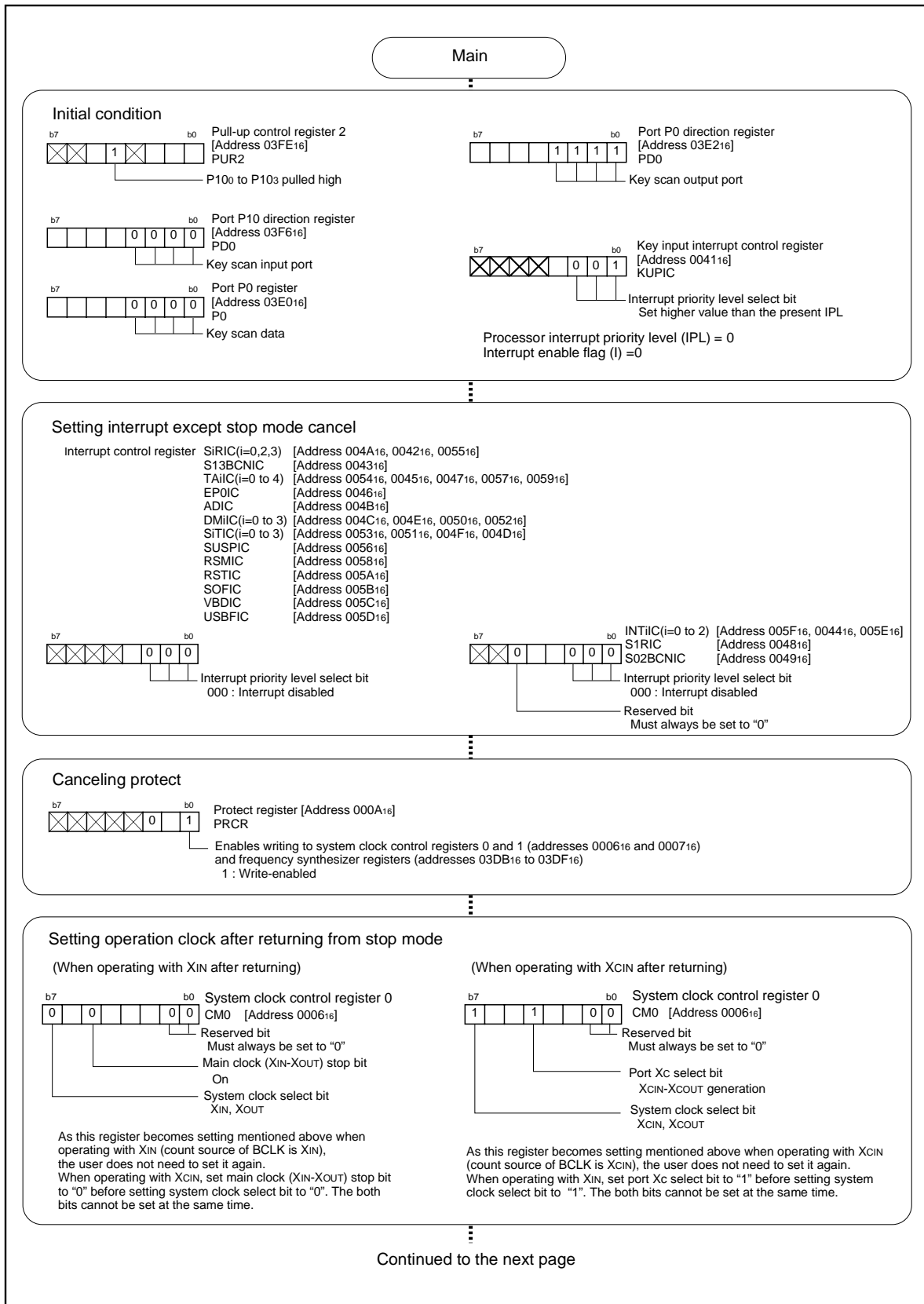


Figure 3.8.3. Set-up procedure of controlling power using stop mode (1)

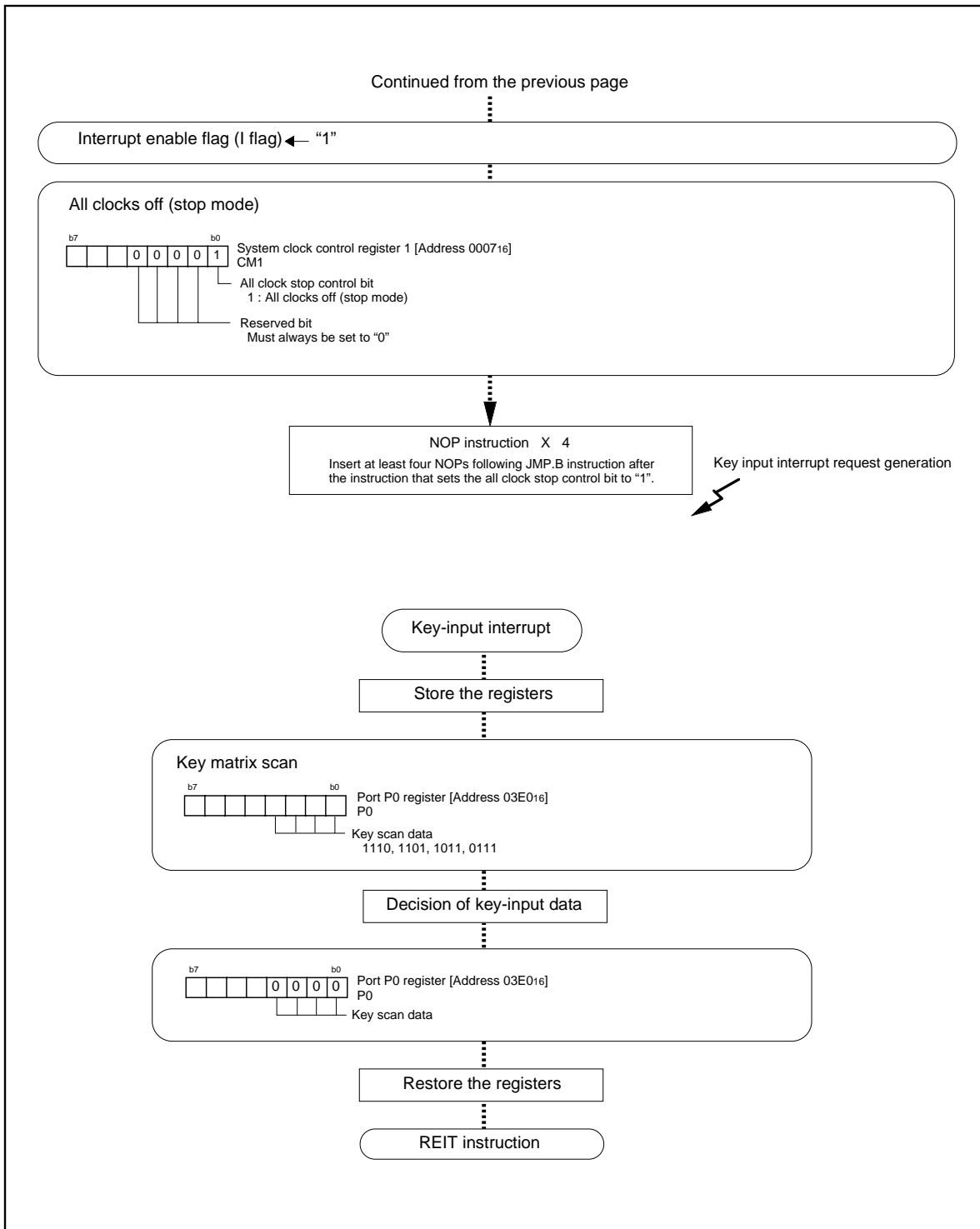


Figure 3.8.4. Set-up procedure of controlling power using stop mode (2)

### 3.9 Controlling Power Using Wait Mode

**Overview** The following are steps for controlling power using wait mode. Figure 3.9.1 shows the operation timing, and Figures 3.9.2 to 3.9.4 show the set-up procedure.

Use the following peripheral functions:

- Timer mode of timer A
- Wait mode

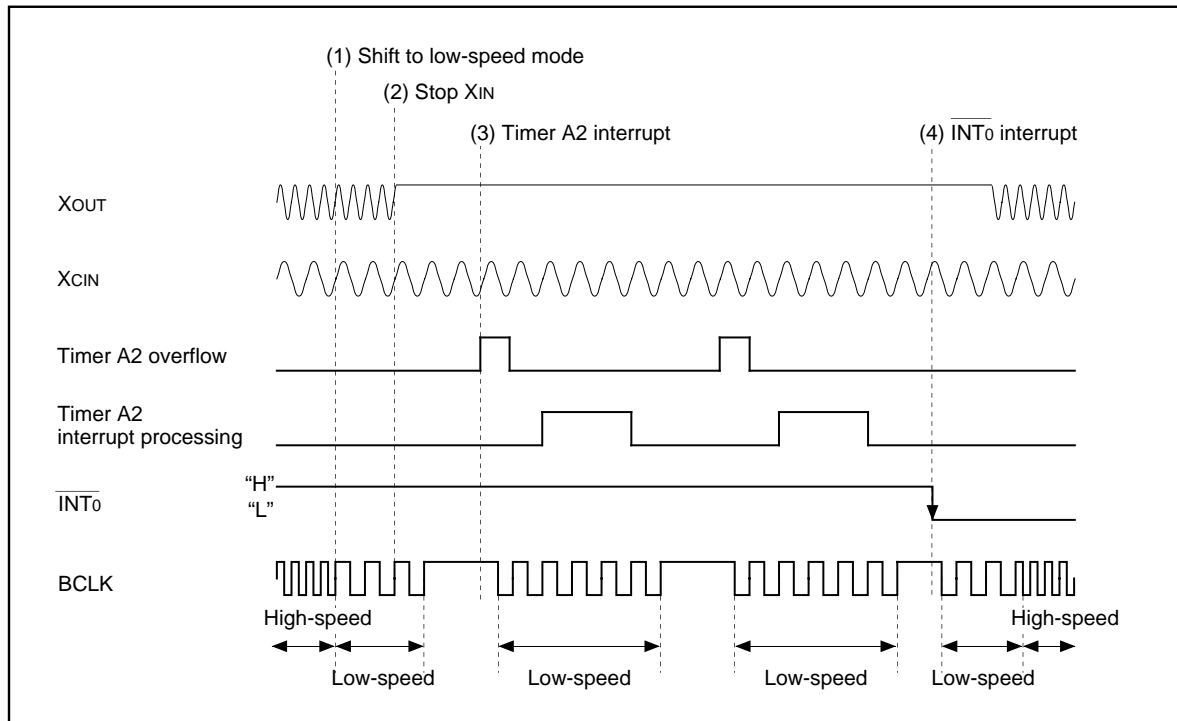
A flag named “F-WIT” is used in the set-up procedure. The purpose of this flag is to decide whether or not to clear wait mode. If F\_WIT = “1” in the main program, the wait mode is entered; if F\_WIT = “0”, the wait mode is cleared.

#### Specifications

- (1) Connect a 32.768-kHz oscillator to XCIN to serve as the timer count source. As interrupts occur every one second, which is a count the timer reaches, the controller returns from wait mode and count the clock using a program.
- (2) Clear wait mode if a  $\overline{\text{INT0}}$  interrupt request occurs.

#### Operation

- (1) Switch the system clock from XIN to XCIN to get low-speed mode.
- (2) Stop XIN and enter wait mode. In this instance, enable the timer A2 interrupt and the  $\overline{\text{INT0}}$  interrupt.
- (3) When a timer A2 interrupt request occurs (at 1-second intervals), start supplying the BCLK from XCIN. At this time, count the clock within the routine that handles the timer A2 interrupts and enter wait mode again.
- (4) If a  $\overline{\text{INT0}}$  interrupt occurs, start supplying the BCLK from XCIN. Start the XIN oscillation within the  $\overline{\text{INT0}}$  interrupt, and switch the system clock to XIN.



**Figure 3.9.1. Operation timing of controlling power using wait mode**

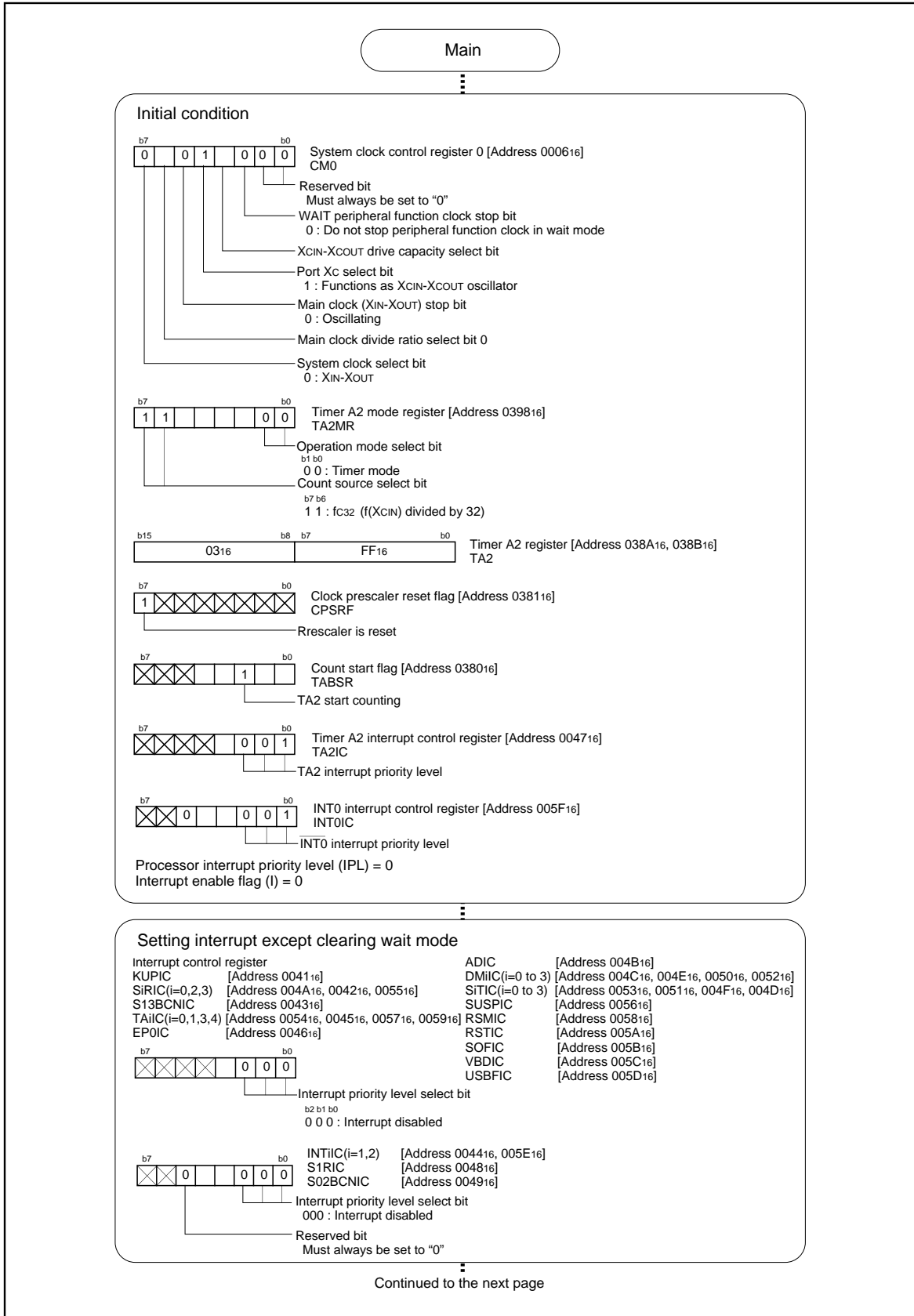


Figure 3.9.2. Set-up procedure of controlling power using wait mode (1)

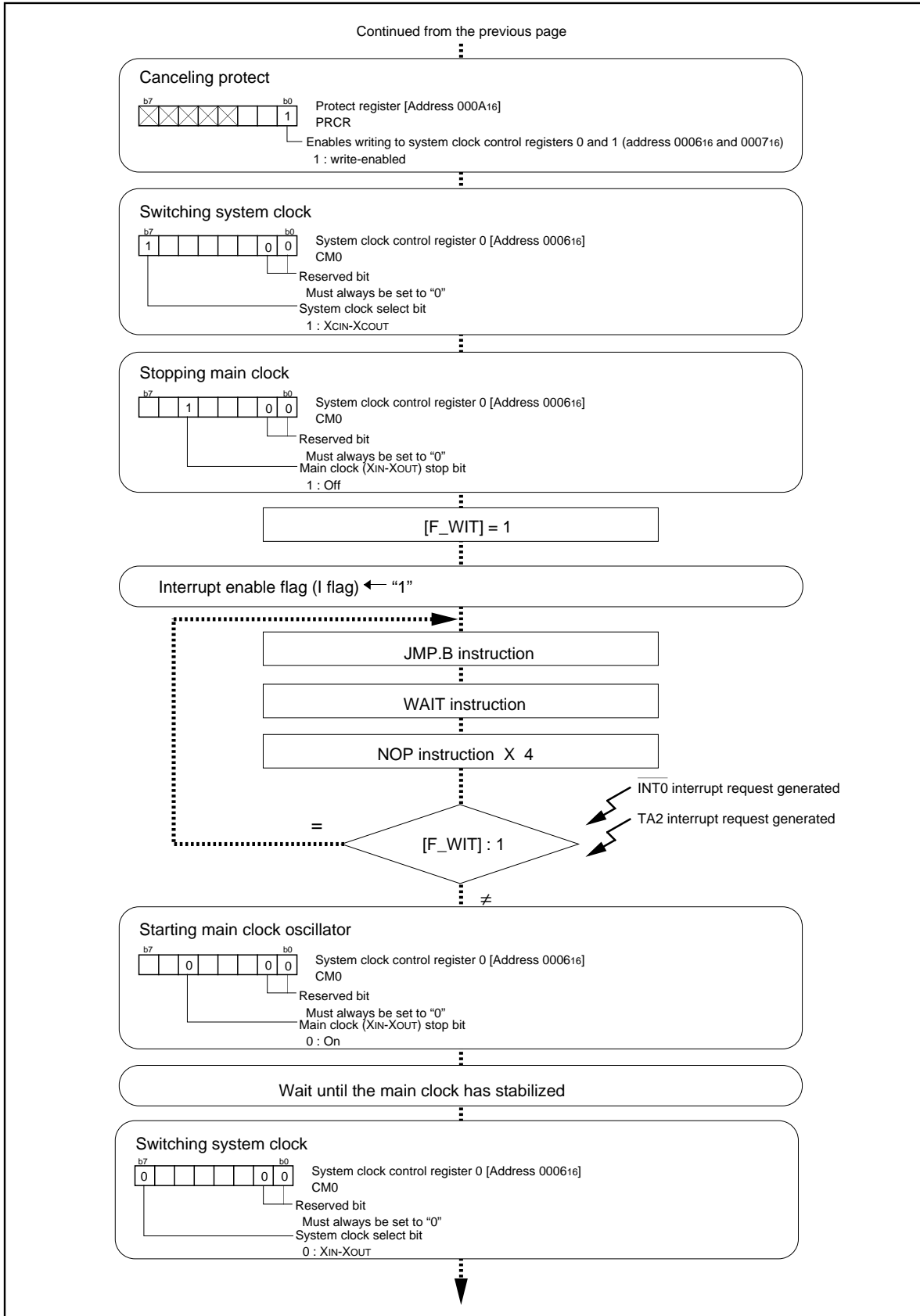


Figure 3.9.3. Set-up procedure of controlling power using wait mode (2)



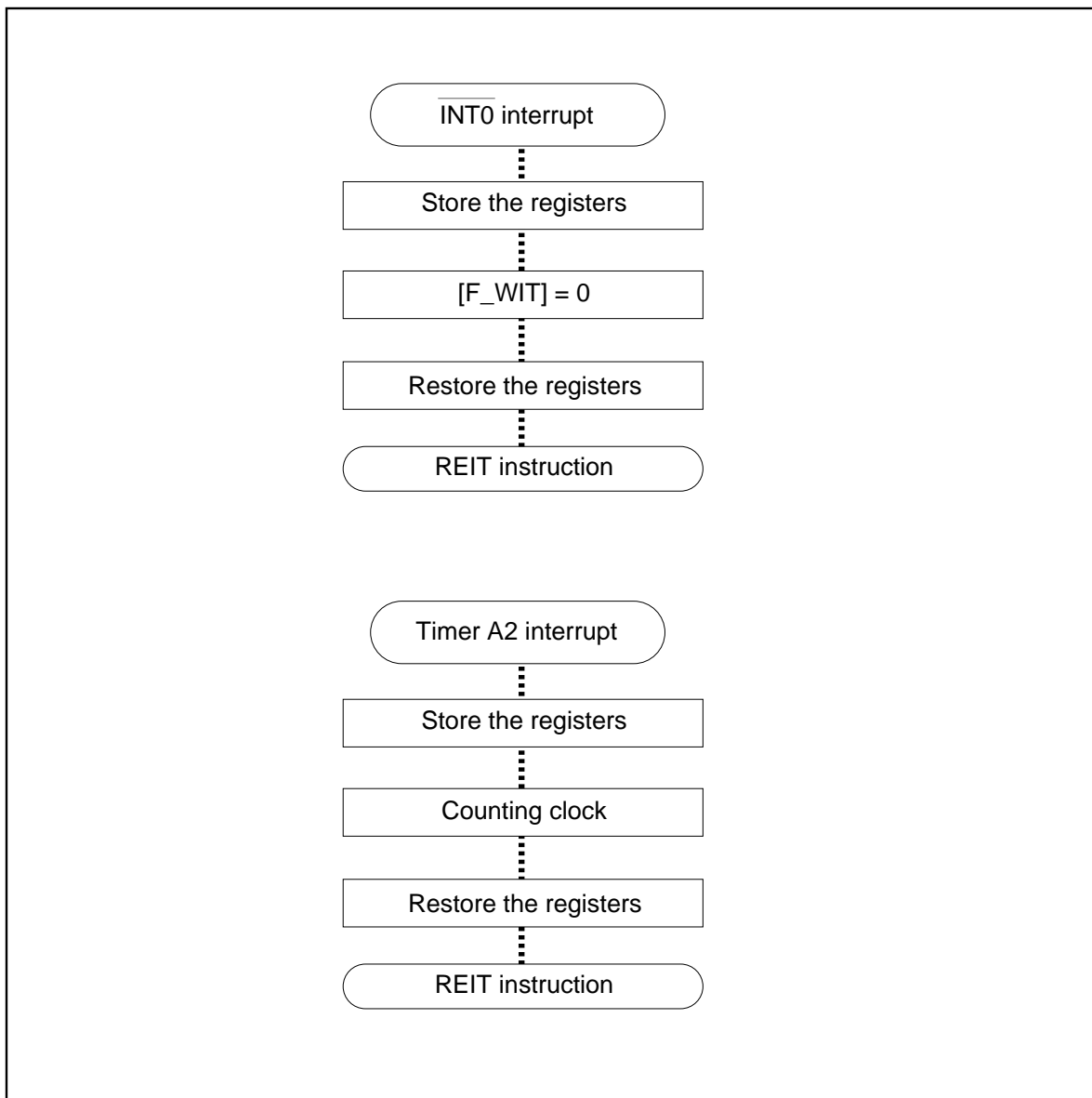


Figure 3.9.4. Set-up procedure of controlling power using wait mode (3)

# Chapter 4

---

## External Buses

### 4.1 Overview of External Buses

Memory and I/O external expansion can be connected to microcomputer easily by using external buses. When memory expansion mode or microprocessor mode is selected for processor mode, some of the pins function as the address bus, the data bus, and as control signals and this makes the external buses be able to operate.

When accessing an external area, 8-bit data bus width or 16-bit data bus width can be selected, based on the BYTE pin level. 16-bit width is used to access an internal area, regardless of the level of the BYTE pin. Fix the BYTE pin either to "H" or "L" level. 8-bit and 16-bit data bus widths cannot be used together in an external area.

Make sure the BYTE pin is fixed to "H" level when an 8-bit bus width is selected and "L" level when a 16-bit bus width is selected.

## 4.2 Data Access

### 4.2.1 Data Bus Width

If the voltage level input to the BYTE pin is “H”, the external data bus width becomes 8 bits, and P10 (/D8) through P17 (/D15) can be used as I/O ports (Figure 4.2.1).

If the voltage level input to the BYTE pin is “L”, the external data bus width becomes 16 bits, and P00 (/D0) through P07 (/D7), and P10 (/D8) through P17 (/D15) operate as a data bus (D0 through D15) (Figure 4.2.1).

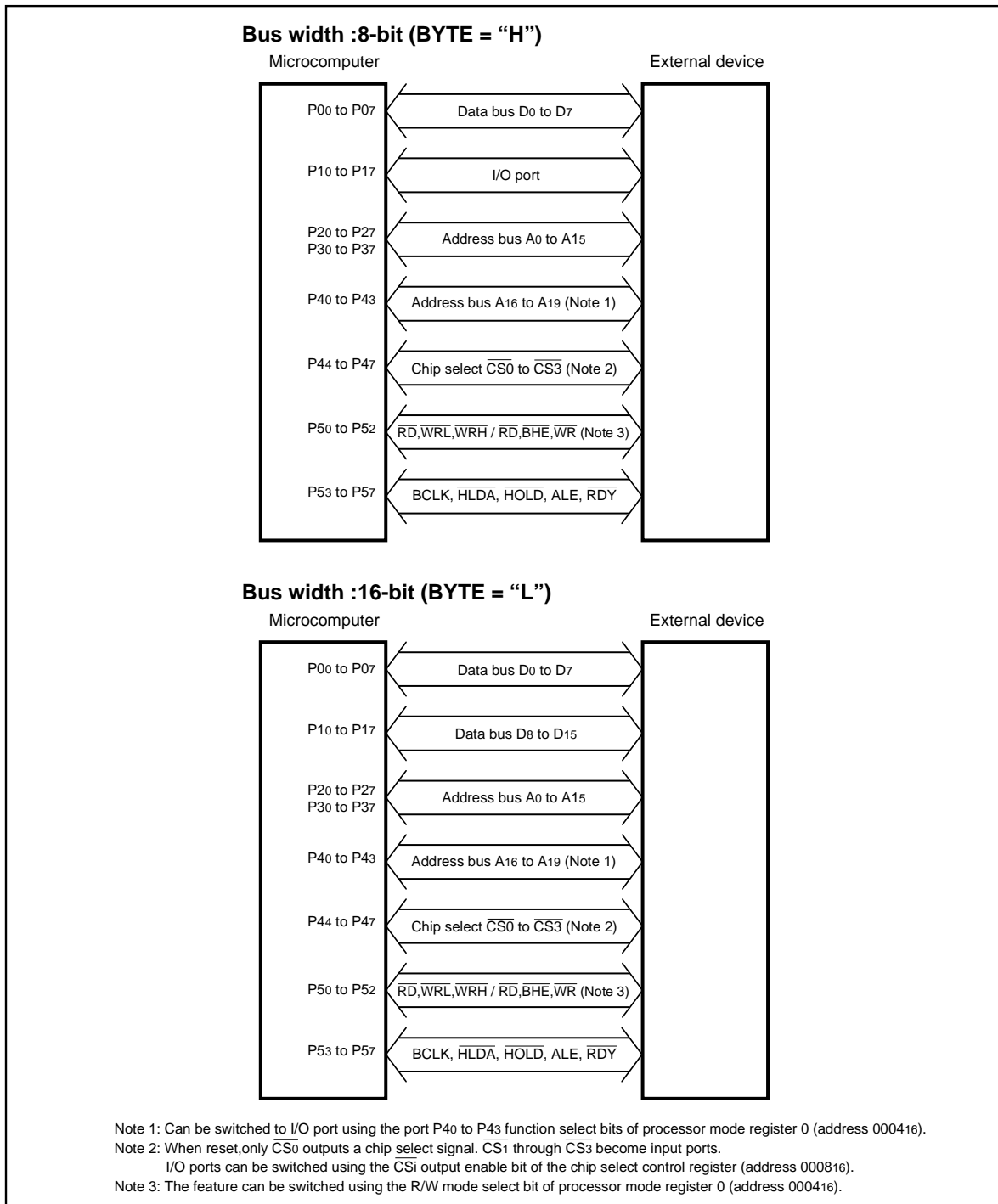


Figure 4.2.1. Level of BYTE pin and external data bus width

### 4.2.2 Chip Selects and Address Bus

Chip selects (P44/ $\overline{CS0}$  through P47/ $\overline{CS3}$ ) are output in areas resulting from dividing a 1-M byte memory space into four. To use the chip select, the chip select output must be enabled by setting the chip select control register. Figure 4.2.2 shows addresses in which chip selects become active (“L”). Since the extent of the internal area and the external area in memory expansion mode is different from those in microprocessor mode, there is a difference between areas for which  $\overline{CS0}$  is output. When an internal ROM/RAM area is being accessed, no chip select is output, and the address bus does not change (the address of the external area that was accessed previously is held).

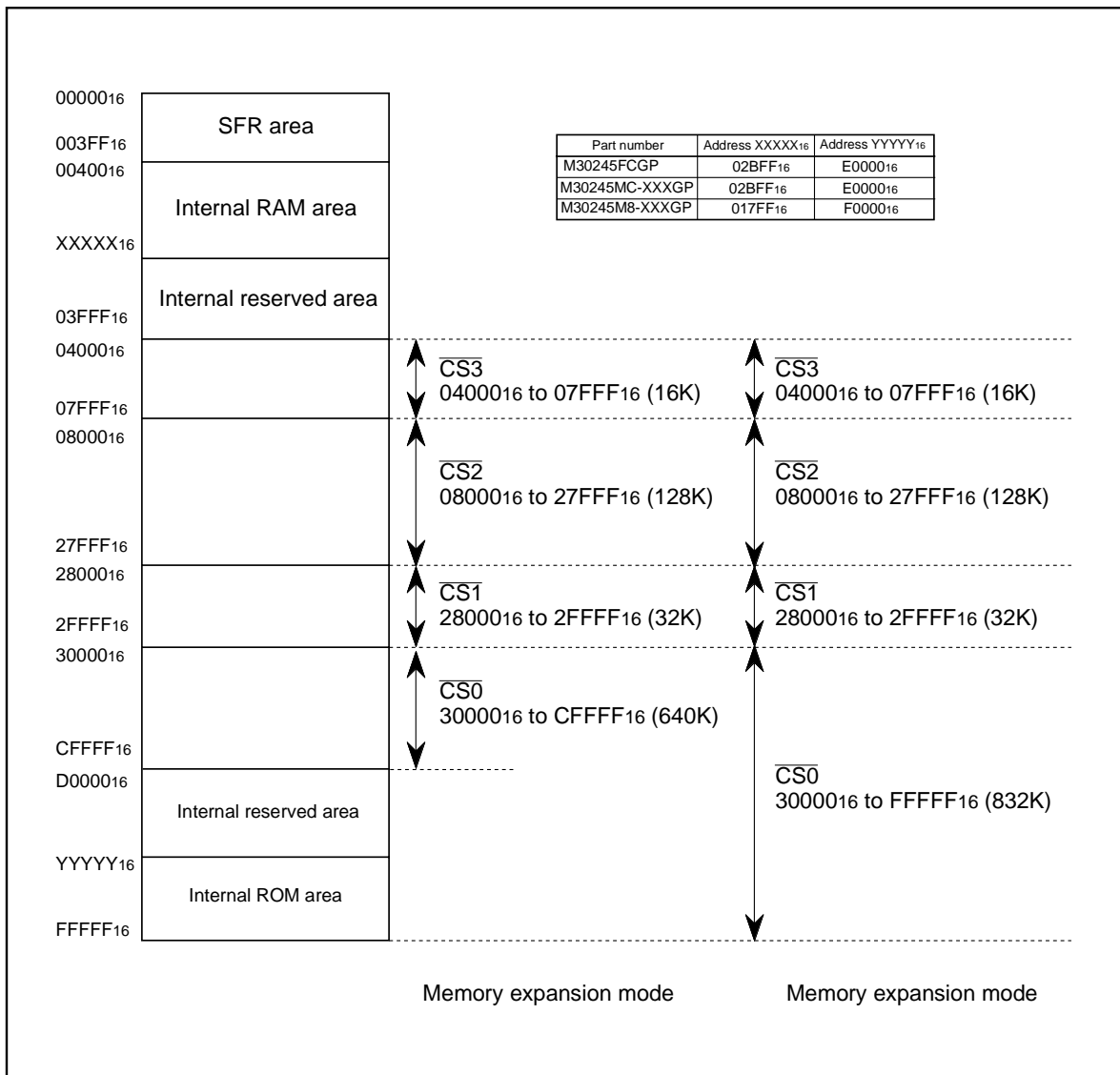


Figure 4.2.2. Addresses in which chip selects turn active (“L”)

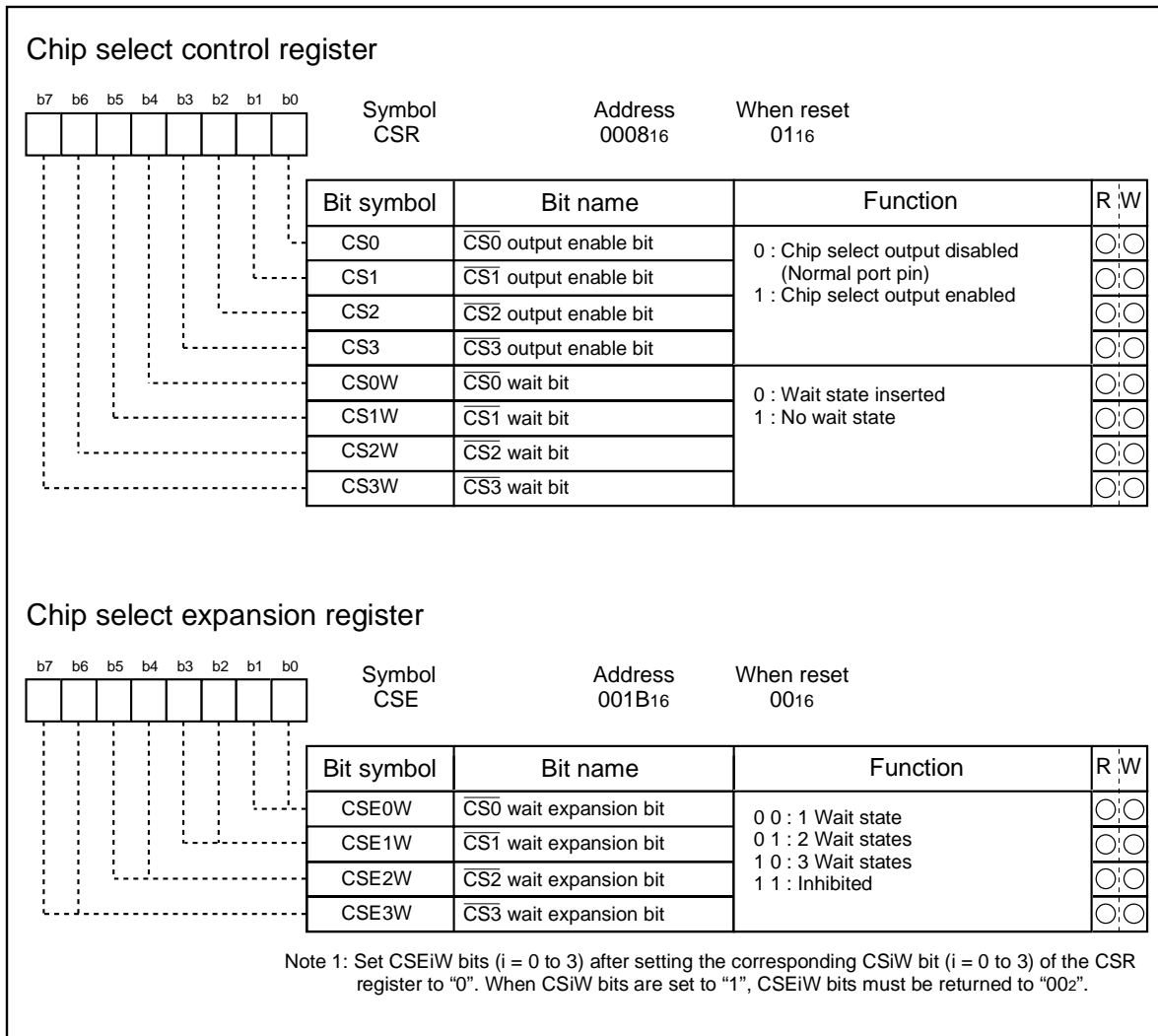


Figure 4.2.3. Level of BYTE pin and external data bus width

### 4.2.3 R/W Modes

The read/write signal that is output when accessing an external area can be selected between the  $\overline{RD}/\overline{BHE}/\overline{WR}$  and the  $\overline{RD}/\overline{WRH}/\overline{WRL}$  modes by setting the R/W mode select bit (bit 2) of the processor mode register 0 (address 0004<sub>16</sub>). Use the  $\overline{RD}/\overline{BHE}/\overline{WR}$  mode to access an 8-bit and a 16-bit wide RAM, and the  $\overline{RD}/\overline{WRH}/\overline{WRL}$  to access a 16-bit wide RAM.

When the M30245 is reset, the  $\overline{RD}/\overline{BHE}/\overline{WR}$  mode is selected by default. To switch over the R/W mode, change the  $\overline{RD}/\overline{BHE}/\overline{WR}$  to the  $\overline{RD}/\overline{WRH}/\overline{WRL}$  mode before accessing an external RAM.

Refer to the connection examples of  $\overline{RD}/\overline{BHE}/\overline{WR}$  and  $\overline{RD}/\overline{WRH}/\overline{WRL}$  shown in Section 4.3, "Connection Examples."

### 4.3 Connection Examples

#### 4.3.1 16-bit Memory to 16-bit Width Data Bus Connection Example

Figure 4.3.1 shows an example of connecting M5M51016BTP (SRAM). In this diagram, when reset the microcomputer starts operating in single-chip mode. Change this mode to memory expansion mode in a program.

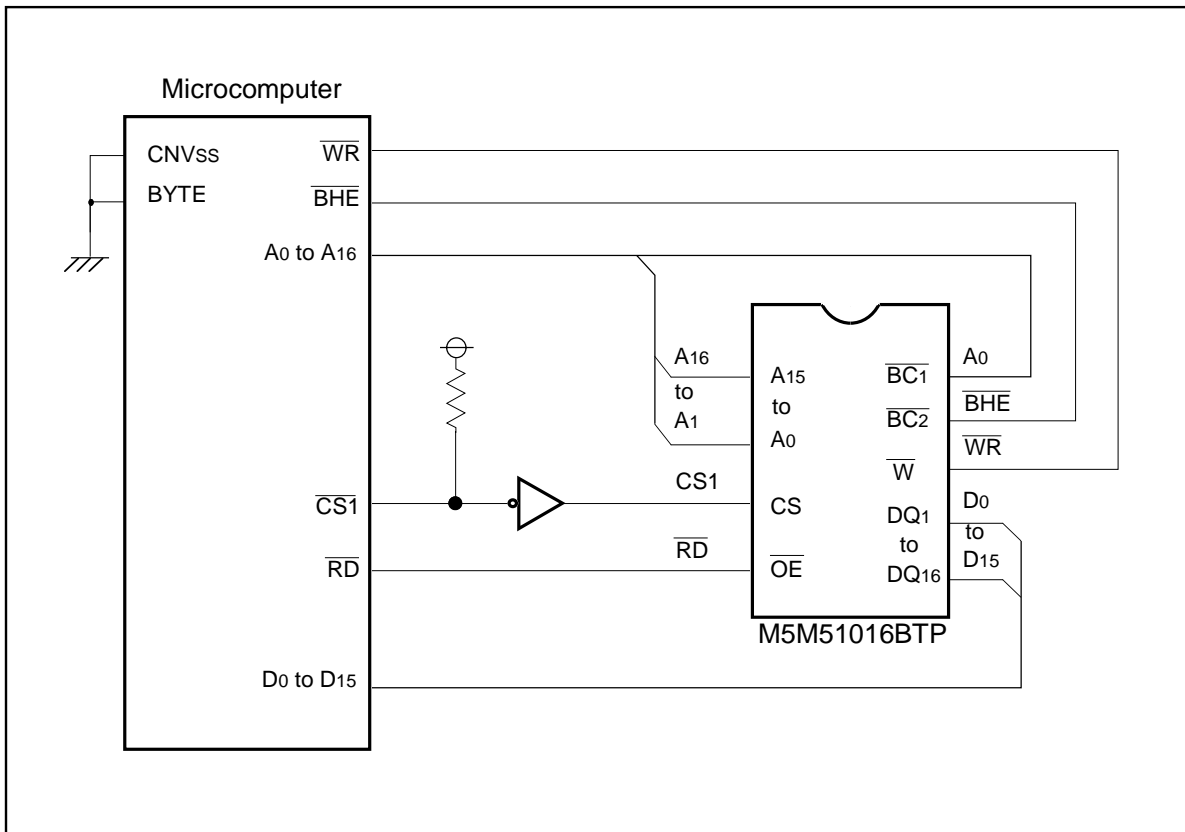


Figure 4.3.1. Example of connecting M5M51016BTP

### 4.3.2 8-bit Memory to 16-bit Width Data Bus Connection Example

Figure 4.3.2 shows an example of connecting two M5M5278's (SRAM) to a 16-bit data bus. In this diagram, when reset the microcomputer starts operating in single-chip mode. Change this mode to memory expansion mode in a program.

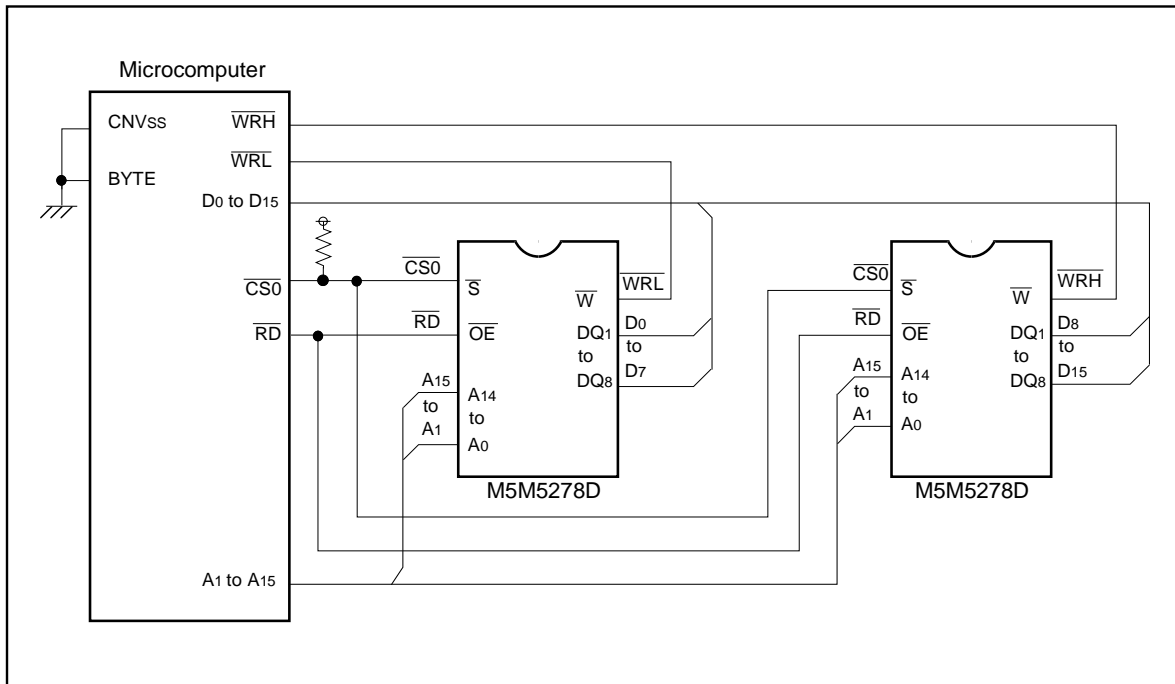


Figure 4.3.2. Example of connecting two M5M5278's to a 16-bit data bus



Figure 4.3.3 shows how to connect two Am29LV008B (flash memory). In 16-bit bus mode, the  $\overline{\text{BHE}}/\overline{\text{WRH}}$  pin functions as  $\overline{\text{BHE}}$ . When connecting 8-bit flash memory chips to the 16-bit bus, make sure the microcomputer's  $\overline{\text{WRL}}$  pin is connected to the  $\overline{\text{WR}}$  pins on both flash memory chips, and that data is written to the flash memory in units of 16 bits beginning with an even address.

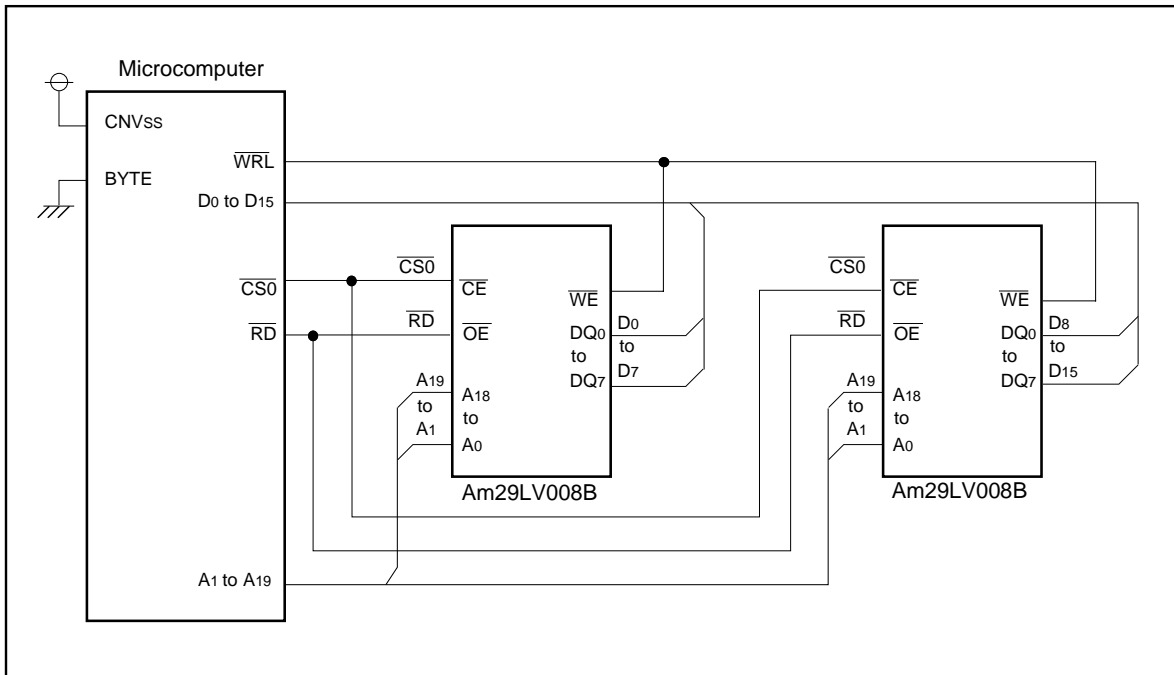


Figure 4.3.3. Example of connecting two Am29LV008B's to a 16-bit data bus

### 4.3.3 8-bit Memory to 8-bit Width Data Bus Connection Example

Figure 4.3.4 shows an example of connecting two M5M5278's (SRAM) to an 8-bit data bus. In this diagram, when reset the microcomputer starts operating in single-chip mode. Change this mode to memory expansion mode in a program.

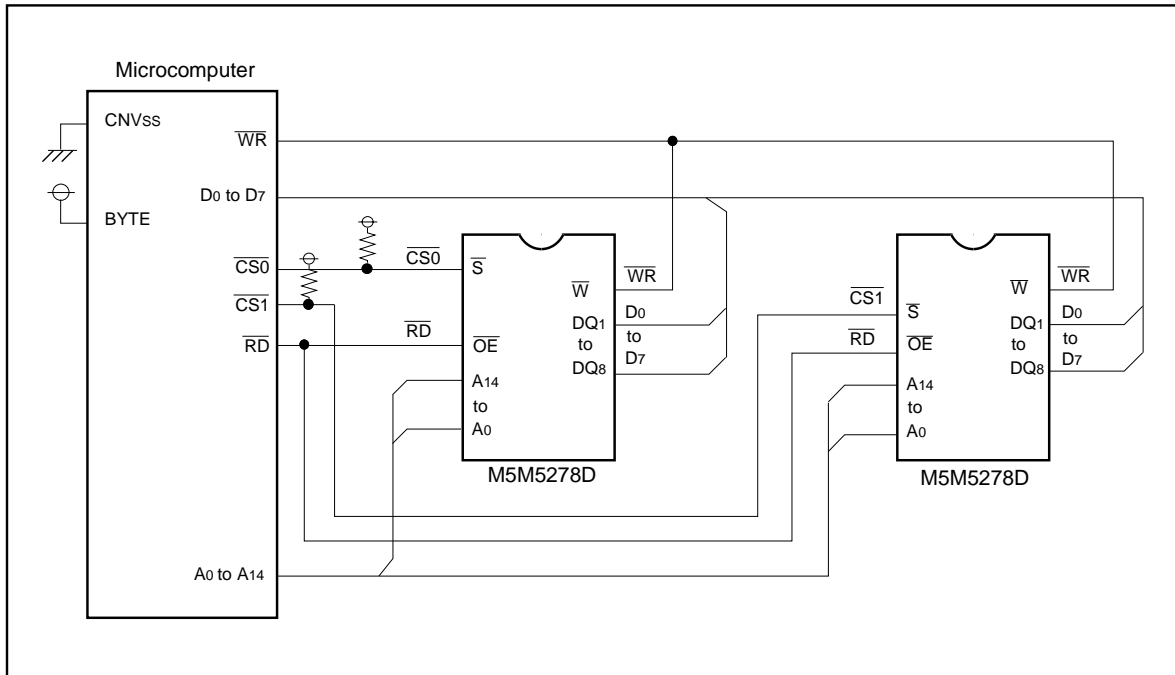


Figure 4.3.4. Example of connecting two M5M5278's to an 8-bit data bus

#### 4.3.4 Two 8-bit and 16-Bit Memory to 16-Bit Width Data Bus Connection Example

Figure 4.3.5 shows an example of connecting M5M28F102 (16-bit flash memory) and two M5M5278's (8-bit SRAM) to a 16-bit data bus.

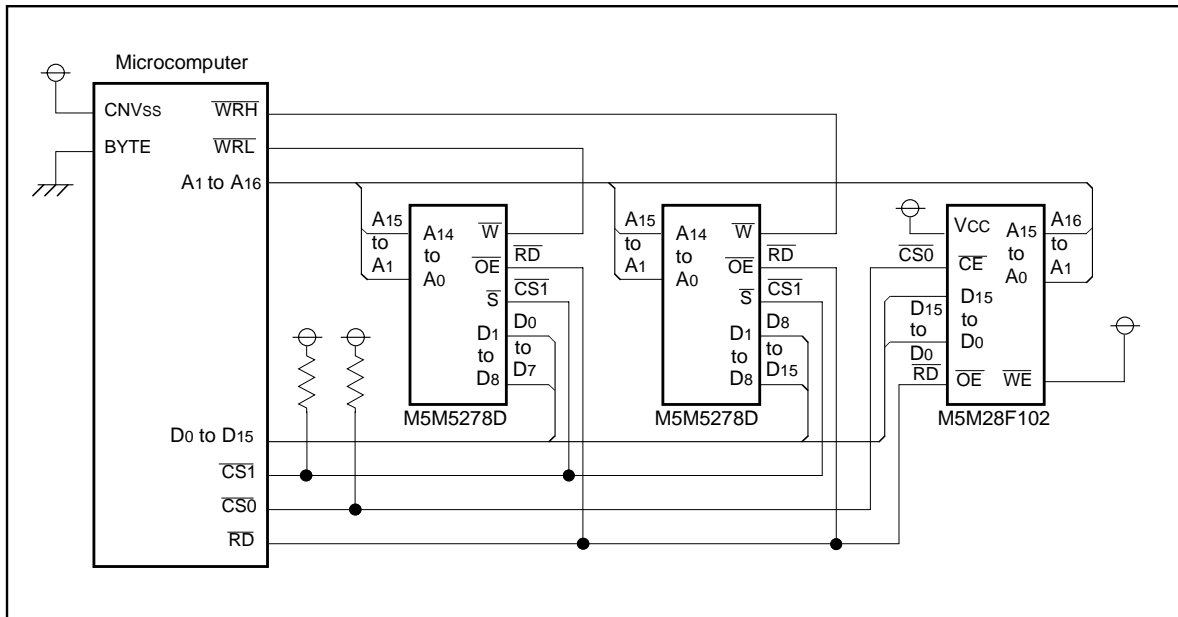


Figure 4.3.5. Example of connection of two 8-bit memories and one 16-bit memory to 16-bit width data bus

### 4.3.5 Chip Selects and Address Bus

When there are insufficient chip select signals, it is necessary to generate chip selects externally. Figure 4.3.6 shows an example of a connection in which the CS2 (128K bytes) area is divided into four 32K byte areas.

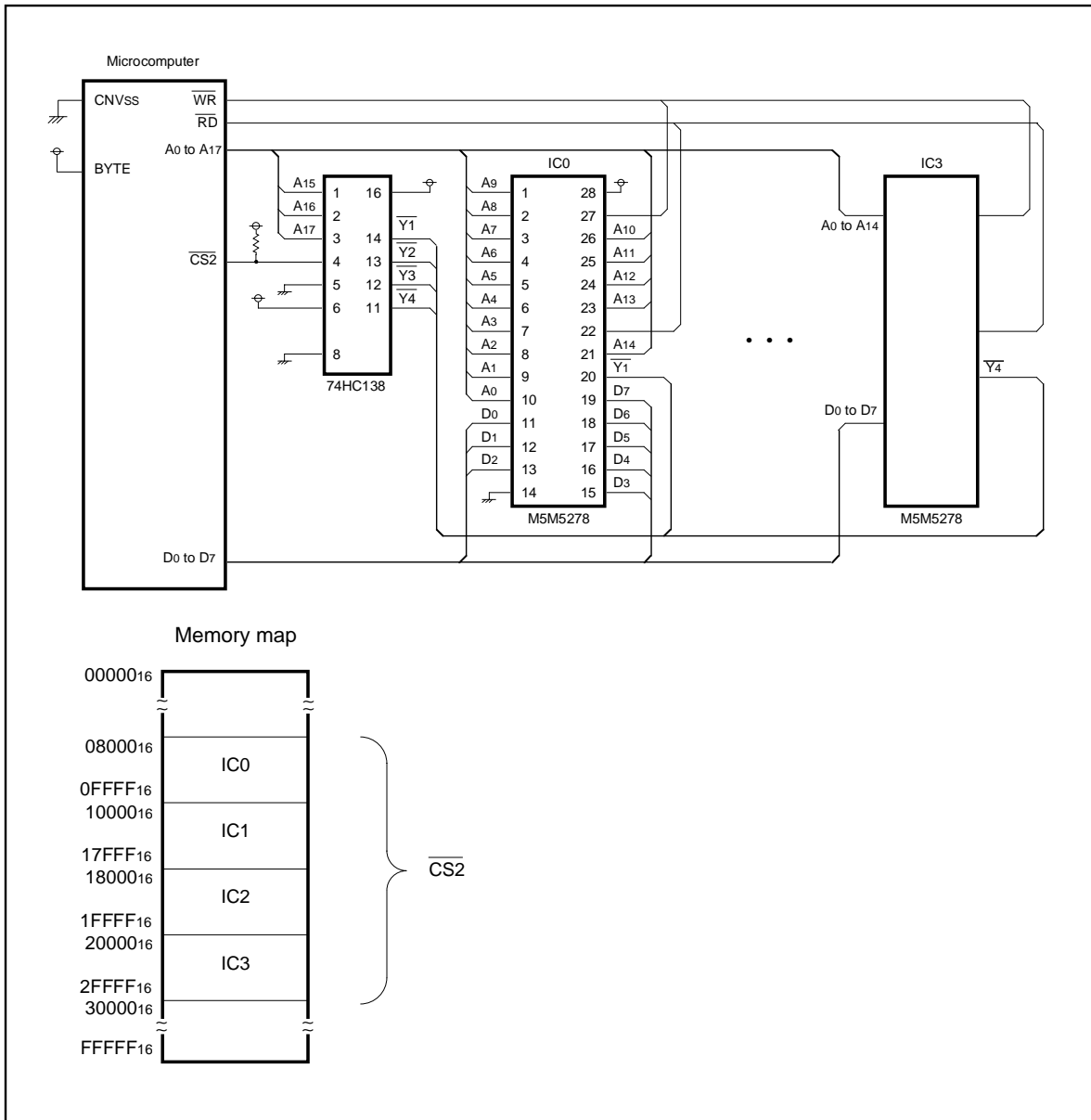


Figure 4.3.6. Chip selects and address bus

## 4.4 Connectable Memories

### 4.4.1 Operation Frequency and Access Time

Connectable memories depend upon the BCLK frequency  $f(\text{BCLK})$ . The frequency of  $f(\text{BCLK})$  is equal to that of the BCLK, and is contingent on the oscillator's frequency and on the settings in the system clock select bits (bit 6 of address 0006<sub>16</sub>, and bits 6 and 7 of address 0007<sub>16</sub>).

The following are the conditional equations for the connections. Meet these conditions minimally. Figures 4.4.1 and 4.4.2 show the relation between the frequency of BCLK and memory.

#### (1) Read cycle time (tCR)/write cycle time (tCW)

Read cycle time (tCR) and write cycle time (tCW) must satisfy the following conditional expressions:

- **With the Wait option cleared**

$$t\text{CR} < 10^9/f(\text{BCLK}) \text{ and } t\text{CW} < 2 \times 10^9/f(\text{BCLK})$$

(When CSxW = 1 read: one cycle of BCLK write: two cycles of BCLK)

- **With the Wait option selected**

$$t\text{CR} < (m+1) \times 10^9/f(\text{BCLK}) \text{ and } t\text{CW} < (m+1) \times 10^9/f(\text{BCLK})$$

(When CSxW = 0 and the number of the expansion waits is selected by the CSExW bit)

(m denotes the number of Wait states: m = "1" when 1 wait selected, "m = 2" when 2 waits selected, and "m = 3" when 3 waits selected)

#### (2) Address access time [ta(A)]

Address access time [ta(A)] must satisfy the following conditional expressions:

##### (a) Vcc = 3.0 to 3.6 V

- **With the Wait option cleared**

$$t\text{a}(\text{A}) < 10^9/f(\text{BCLK}) - 80(\text{ns})^*$$

- **With the Wait option selected**

$$t\text{a}(\text{A}) < (m+1) \times 10^9/f(\text{BCLK}) - 80(\text{ns})^*$$

(m = "1" when 1 wait selected, "m = 2" when 2 waits selected, and "m = 3" when 3 waits selected)

$$^*80(\text{ns}) = t\text{d}(\text{BCLK} - \text{AD}) + t\text{su}(\text{DB} - \text{RD}) - t\text{h}(\text{BCLK} - \text{RD})$$

$$= (\text{address output delay time}) + (\text{data input setup time}) - (\text{RD signal output hold time})$$

#### (3) Chip select access time [ta(S)]

Chip select access time [ta(S)] must satisfy the following conditional expressions:

##### (a) Vcc = 3.0 to 3.6 V

- **With the Wait option cleared**

$$t\text{a}(\text{S}) < 10^9/f(\text{BCLK}) - 80(\text{ns})^*$$

- **With the Wait option selected**

$$t\text{a}(\text{S}) < (m+1) \times 10^9/f(\text{BCLK}) - 80(\text{ns})^*$$

(m = "1" when 1 wait selected, "m = 2" when 2 waits selected, and "m = 3" when 3 waits selected)

$$^*80(\text{ns}) = t\text{d}(\text{BCLK} - \text{CS}) + t\text{su}(\text{DB} - \text{RD}) - t\text{h}(\text{BCLK} - \text{RD})$$

$$= (\text{chip select output delay time}) + (\text{data input setup time}) - (\text{RD signal output hold time})$$

**(4) Output enable time [ta(OE)]**

Output enable time [ta(OE)] must satisfy the following conditional expressions:

**(a) Vcc = 3.0 to 3.6 V**

- **With the Wait option cleared**

$$ta(OE) < 10^9 / (f(BCLK) \times 2) - 60(\text{ns}) = tac1(\text{RD-DB})$$

- **With the Wait option selected**

$$ta(OE) < (m+0.5) \times 10^9 / f(BCLK) - 60(\text{ns}) = tac2(\text{RD-DB})$$

(m = "1" when 1 wait selected, "m = 2" when 2 waits selected, and "m = 3" when 3 waits selected)

**(5) Data setup time [tsu(D)]**

Data setup time [tsu(D)] must satisfy the following conditional expressions:

**(a) Vcc = 3.0 to 3.6 V**

- **PM16 = 0 (WR width normal)**

$$tsu(D) < (n - 0.5) \times 10^9 / f(BCLK) - 40(\text{ns}) = td(\text{DB} - \text{WR})$$

- **PM16 = 1 (WR width expanded)**

$$tsu(D) < n \times 10^9 / f(BCLK) - 40(\text{ns}) = td(\text{DB} - \text{WR})$$

$$\begin{aligned} *40(\text{ns}) &= td(\text{BCLK} - \text{DB}) - th(\text{BCLK} - \text{WR}) \\ &= (\text{data output delay time}) - (\text{WR signal output hold time}) \end{aligned}$$

(n = 1 (no wait), n = 2 (1 wait), n = 3 (2 waits), n = 4 (3 waits))

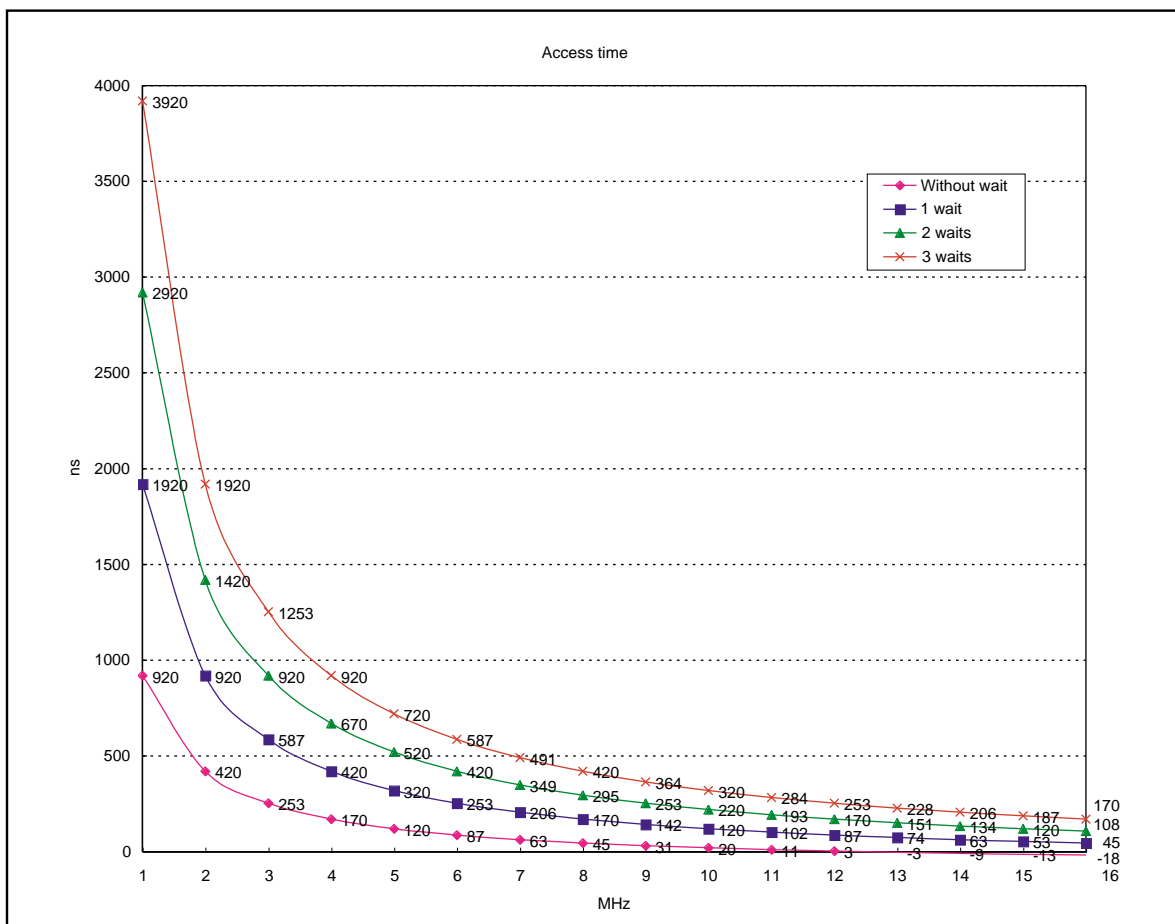


Figure 4.4.1. Relation between the frequency of BCLK and memory (1)

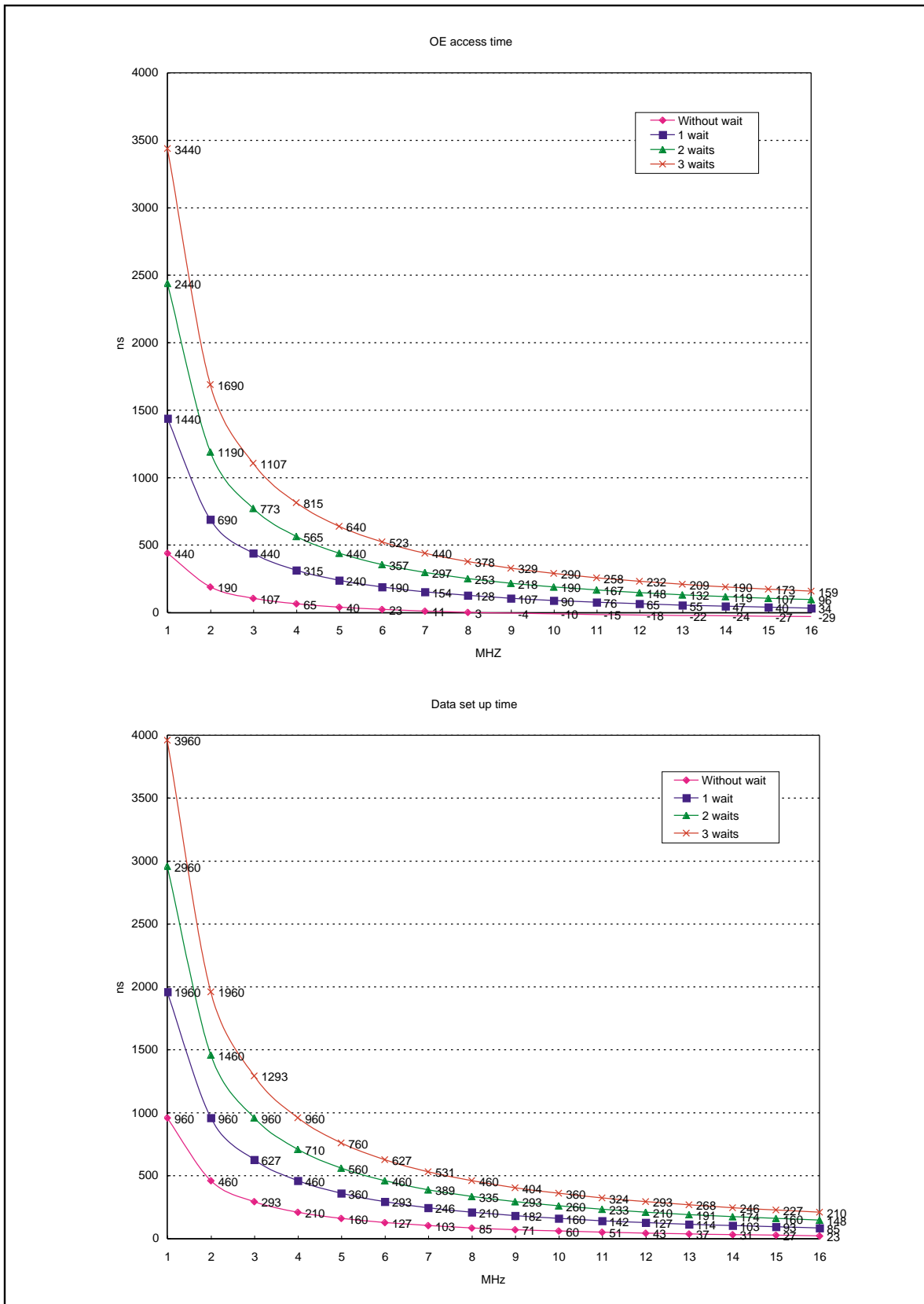


Figure 4.4.1. Relation between the frequency of BCLK and memory (2)

## 4.4.2 Connecting Low-Speed Memory

To connect memory with long access time [ $t_a(A)$ ], either decrease the frequency of BCLK or set a software wait. Using the RDY feature allows you to connect memory having the timing that precludes connection though you set software wait.

### (1) Using software wait

Set software wait for the external memory areas by using either of bits CS0W through CS3W of the chip select control register (address 000816) or the chip select expansion register (address 001B16). When using the RDY signal, the corresponding CS0W through CS3W bit must be set to "0".

Bits CS0W through CS3W of the chip select control register correspond to chip select CS0 through CS3, respectively. If these bits are set to "1", the read bus cycle results in one cycle of BCLK and the write bus cycle results in two cycles of BCLK; if these bits are set to "0", the read/write cycle results in two, three, or four cycles of BCLK according to the chip select expansion register setting. When the corresponding bit of the chip select control register is set to "0", the chip select expansion register setting becomes valid. When this bit is set to "1", set the corresponding bit of the chip select expansion register to "002".

When reset, the value of the chip select control register and the chip select expansion register is set to "0016".

These control bits do not affect the SFR area and the internal ROM/RAM area.

Figure 4.4.1 shows software wait and bus cycle, and Figure 4.4.3 shows relation of processor mode and the wait bit (CSiW, CSiEW).

**Table 4.4.1. Software waits and bus cycles**

Area	CSxW (Note 2)	CSExW (Note 2)	Bus Cycles	
			Read	Write
SFR	Invalid	Invalid	2 BCLK cycles	2 BCLK cycles
Internal ROM/RAM	Invalid	Invalid	1 BCLK cycle	1 BCLK cycle
External memory areas	0	00	2 BCLK cycles	2 BCLK cycles
	0	01	3 BCLK cycles	3 BCLK cycles
	0	10	4 BCLK cycles	4 BCLK cycles
	0	11	Do not set	
	1	00	1 BCLK cycle	2 BCLK cycles

Note 1: When using the  $\overline{\text{RDY}}$  signal, set to "0".

Note 2: Set CSEiW bits ( $i = 0$  to 3) after setting the corresponding CSiW bit ( $i = 0$  to 3) of the CSR register to "0". When CSiW bits are set to "1", CSEiW bits must be returned to "002".



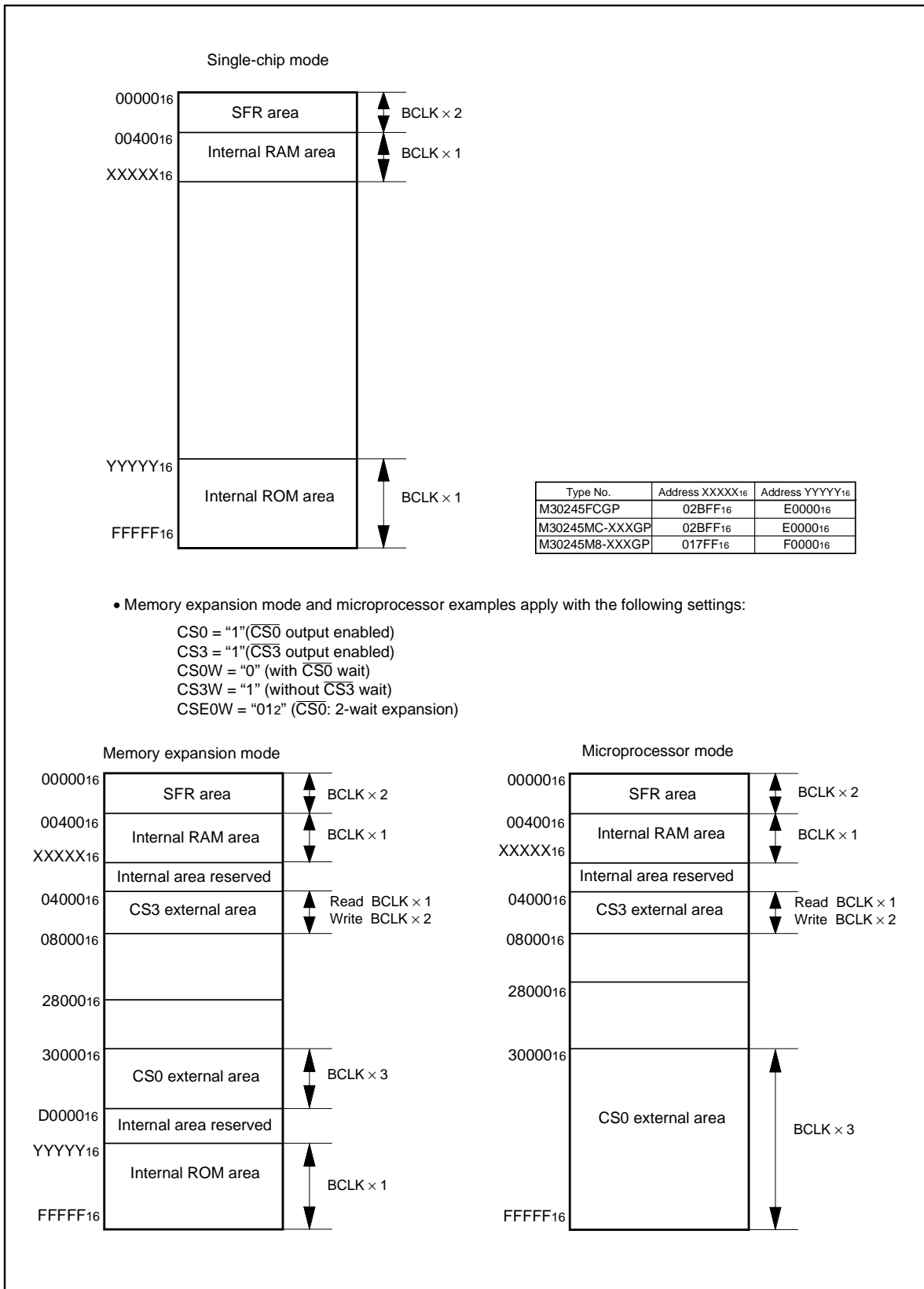


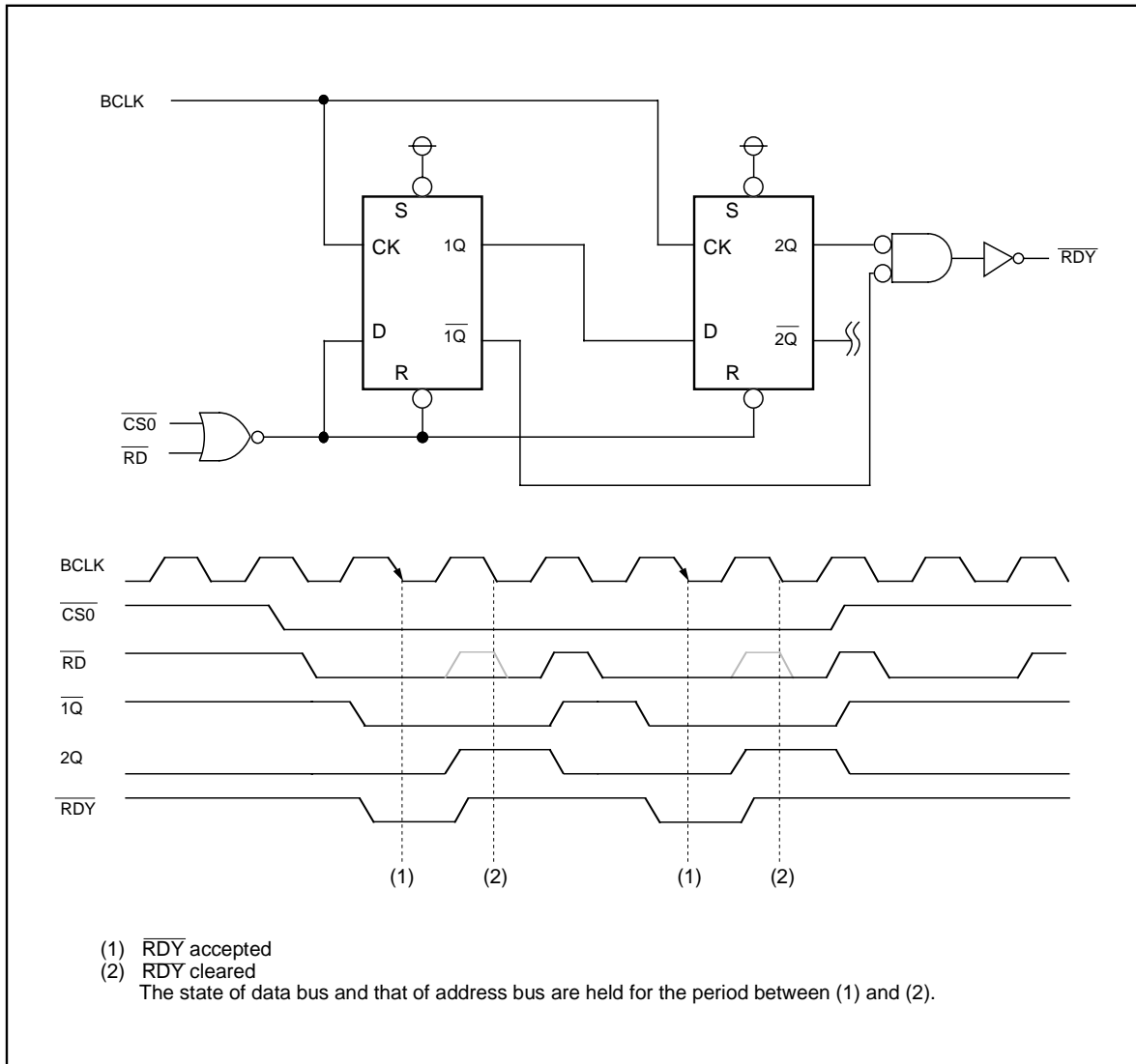
Figure 4.4.3. Relation of processor mode and the wait bit (CSiW, CSiEW)

**(2)  $\overline{\text{RDY}}$  function usage**

To use the  $\overline{\text{RDY}}$  function, set a software wait.

The  $\overline{\text{RDY}}$  function operates when the BCLK signal falls with the  $\overline{\text{RDY}}$  pin at "L"; the bus does not vary for 1 BCLK, and the state at that moment is held.

The  $\overline{\text{RDY}}$  function holds the state of bus for the period in which the  $\overline{\text{RDY}}$  pin is at "L", and releases it when the BCLK signal falls with the  $\overline{\text{RDY}}$  pin at "H". Figure 4.4.4 shows an example of  $\overline{\text{RDY}}$  circuit ( $f(\text{XIN})=10\text{MHz}$ ) that holds the state of bus for 1 BCLK.



**Figure 4.4.4. Example of  $\overline{\text{RDY}}$  circuit holding state of bus for 1 BCLK ( $f(\text{XIN})=10\text{MHz}$ )**

### 4.4.3 Connectable Memories

Connectable memories and their maximum frequencies are given here;  
M30245 group maximum frequency is  
16MHz (without the wait) for  $V_{cc}=3V$ ,

#### (1) Flash memories (Read only mode)

(a) 3V without wait

Maximum frequency (MHz)	Model No.
3.57	M5M29GB/T160BVP-80

(b) 3V with wait

Maximum frequency (MHz)	Model No.
8.33	M5M29GB/T160BVP-80

#### (2) SRAM

(a) 3V without wait

Maximum frequency (MHz)	Model No.
5.12	M5M54R08AJ-12 M5M54R16AJ, ATP-12

(b) 3V without wait

Maximum frequency (MHz)	Model No.
10.0	M5M54R08AJ-12 M5M54R16AJ, ATP-12

#### 4.5 Releasing an External Bus ( $\overline{\text{HOLD}}$ input and $\overline{\text{HLDA}}$ output)

The Hold feature is to relinquish the address bus, the data bus, and the control bus on M30245 side in line with the Hold request from the bus master other than M30245 when the two or more bus masters share the address bus, the data bus, and the control bus. The Hold feature is effective only in memory expansion mode and microprocessor mode.

The sequence of using the Hold feature may be:

1. The external bus master turns the input level of the  $\overline{\text{HOLD}}$  terminal to "L".
2. When M30245 becomes ready to relinquish buses, each bus becomes high-impedance state at the falling edge of BCLK.
3. The  $\overline{\text{HLDA}}$  terminal becomes "L" at the rising edge of the next BCLK.
4. The external bus master uses a bus.
5. When the external bus master finishes using a bus, the external bus master returns the input level of the  $\overline{\text{HOLD}}$  terminal to "H".
6. The output from  $\overline{\text{HLDA}}$  terminal becomes "H" at the rising edge of the next BCLK.
7. Each bus returns from the high-impedance state to the former state at the falling edge of the next BCLK.

As given above, each bus invariably gets in the high-impedance state while the  $\overline{\text{HLDA}}$  output is "L". Also, M30245 does not relinquish buses during a bus cycle. That is, if a Hold request comes in during a bus cycle, the  $\overline{\text{HLDA}}$  output become "L" after that bus cycle finishes.

In the Hold state, the state of each terminal becomes as follows.

- **Address bus A0 to A19**

High-impedance state. The case in which A16 to A19 are used as ports P40 to P43 (64K byte address space) in microprocessor mode and in memory expansion mode too falls under this category.

- **Data bus D0 to D15**

High-impedance state. The case in which D8 to D15 are used as ports P10 to P17 (8-bit external bus width) in microprocessor mode and in memory expansion mode too falls under this category.

- **$\overline{\text{RD}}$ ,  $\overline{\text{WR}}$ ,  $\overline{\text{WRL}}$ ,  $\overline{\text{WRH}}$ ,  $\overline{\text{BHE}}$**

High-impedance state.

- **ALE**

An internal clock signal having the same phase as BCLK is output.

- **$\overline{\text{CS0}}$  to  $\overline{\text{CS3}}$**

High-impedance state. The case in which ports are selected by the chip selection control register too falls under this category.

Figure.4.5.1 shows an example of relinquishing external buses.

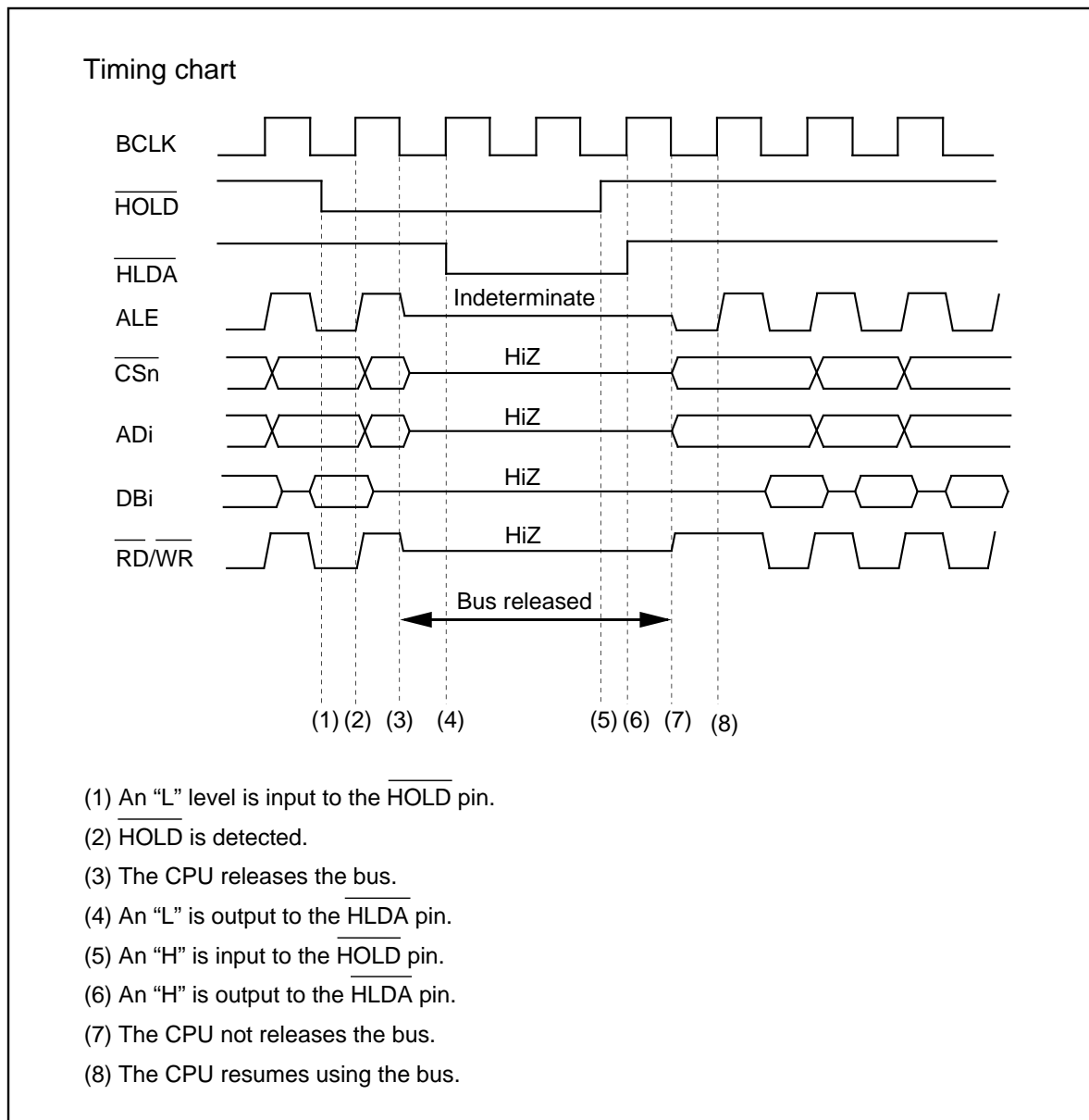


Figure 4.5.1. Example of releasing the external bus

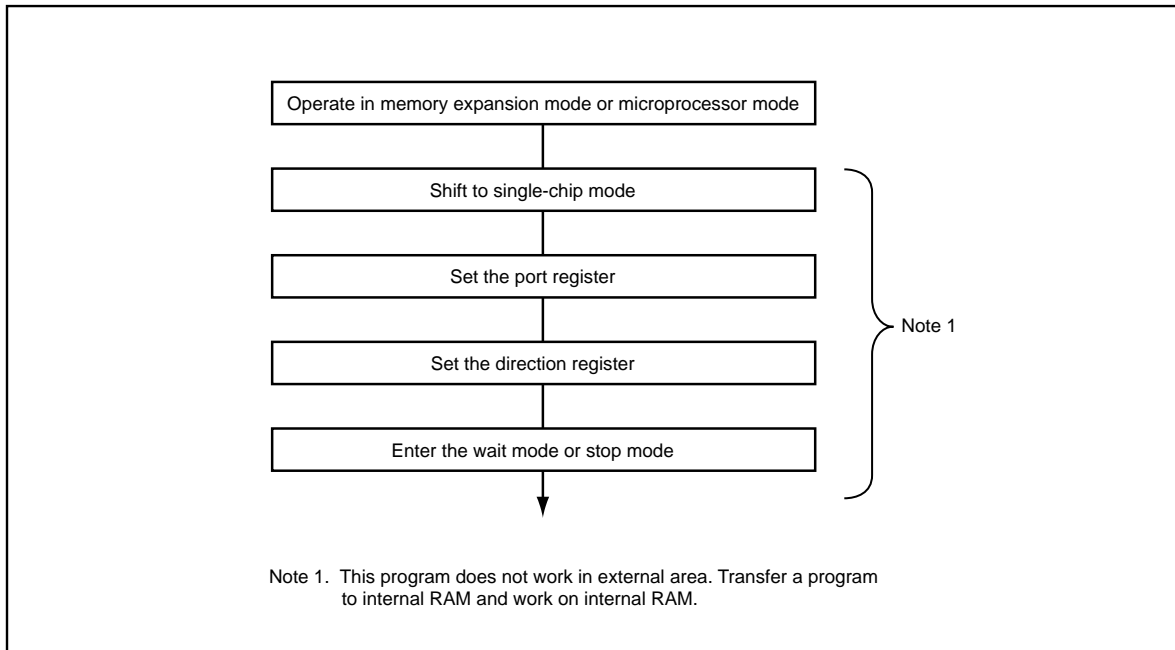
## 4.6 Precautions for External Bus

**Description** When the MCU enters wait mode while operating in memory expansion mode or microprocessor mode, a pin functioning as part of the address or data bus retains its state on the bus before wait mode is entered. Shift to single-chip mode and output an arbitrary value in order to reduce current consumption. By shifting to single-chip mode, a pin which was functioning as part of the bus becomes a general-purpose port and can output an arbitrary value. Set the port registers and direction registers after shifting to single-chip mode (this implies that any control pins ( $\overline{CS}$ ,  $\overline{WR}$ ,  $\overline{RD}$ , etc..) being used for access of an external device be changed as well).

If the port registers and direction registers are set while in memory expansion mode or microprocessor mode, the operation will be ignored.

This is similar when entering stop mode.

Figure 4.6.1 shows the setting procedure to enter wait mode or stop mode.



**Figure 4.6.1. Setting procedure to enter wait mode or stop mode**

THIS PAGE IS BLANK FOR REASONS OF LAYOUT.

# Chapter 5

---

## Standard Characteristics



### 5.1 DC Standard Characteristics

Standard characteristics described in this chapter are just examples and not guaranteed. For rated values, refer to "Electrical Characteristics" of Datasheet.

#### 5.1.1 Port Standard Characteristics

Figures 5.1.1 to 5.1.4 show port standard characteristics.

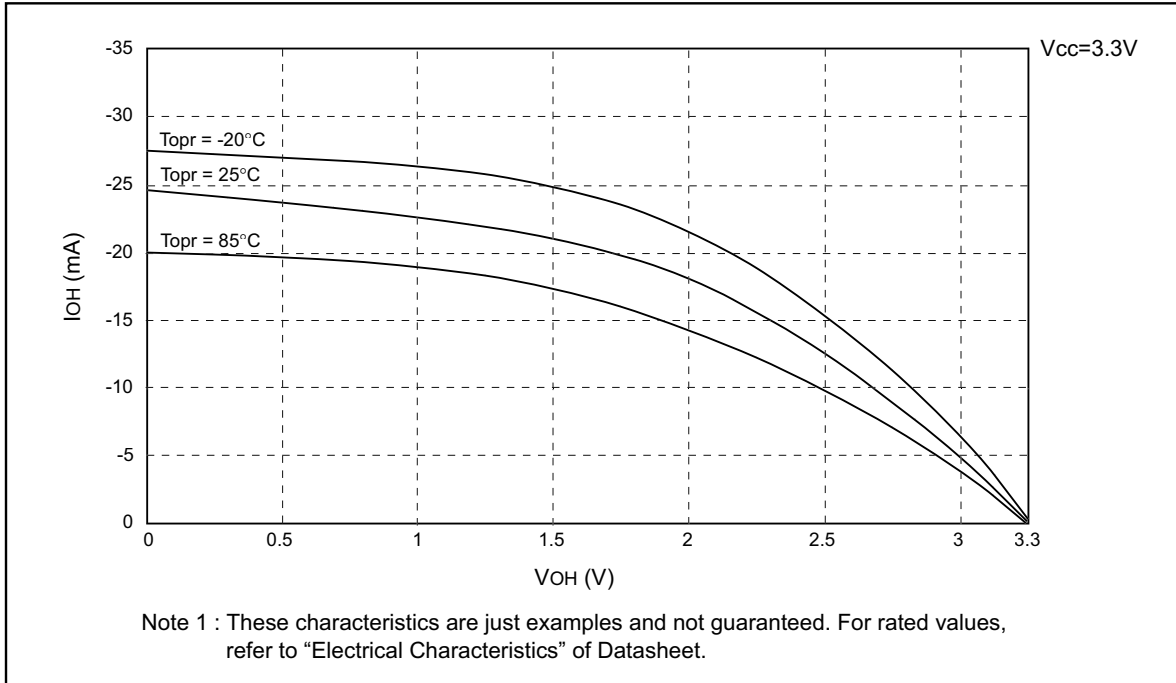


Figure 5.1.1. VOH-IOH standard characteristics of ports P0 to P10 (except P63, P67, and P85)

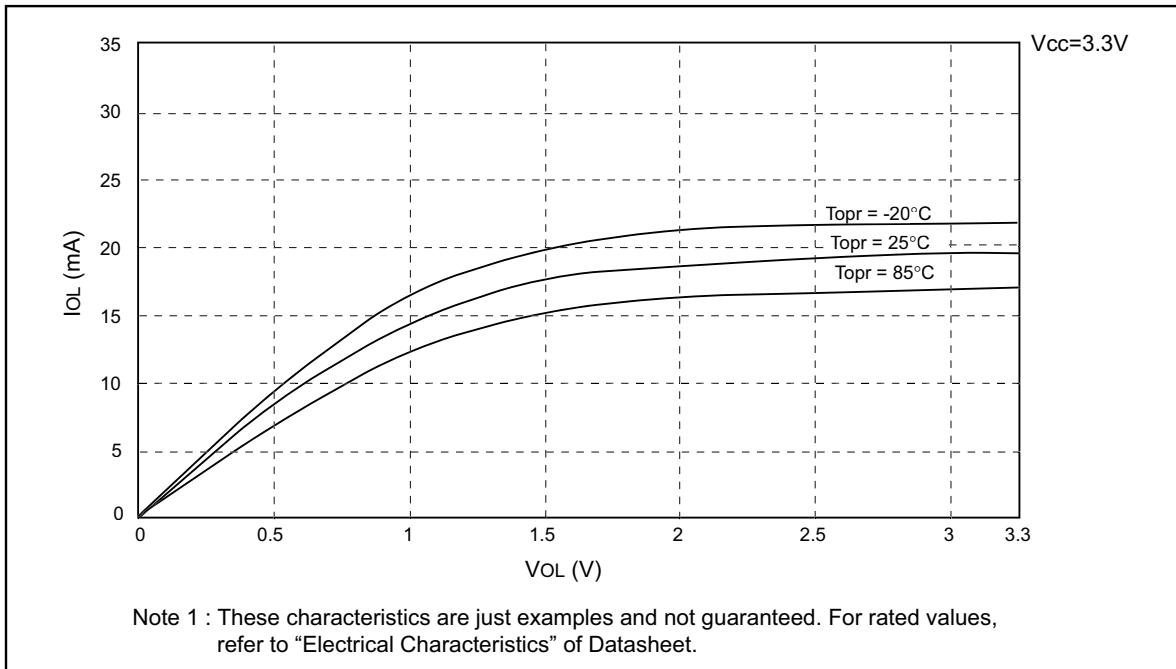


Figure 5.1.2. VOL-IOL standard characteristics of ports P0 to P10 (except P63, P67, and P85)

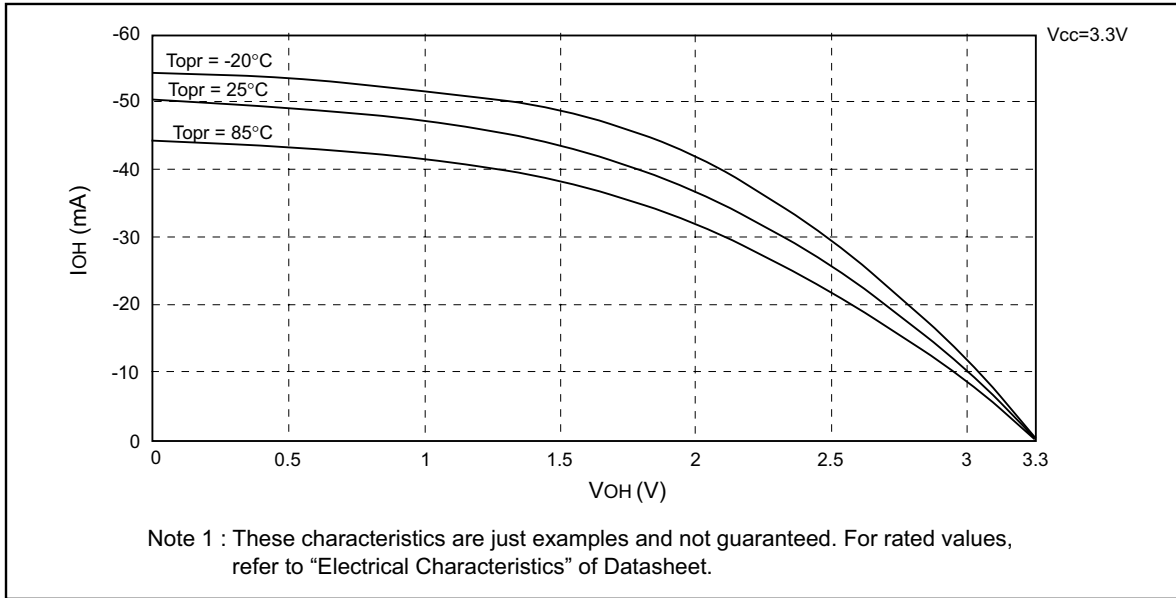


Figure 5.1.3. VOH-IOH standard characteristics of ports P63 to P67

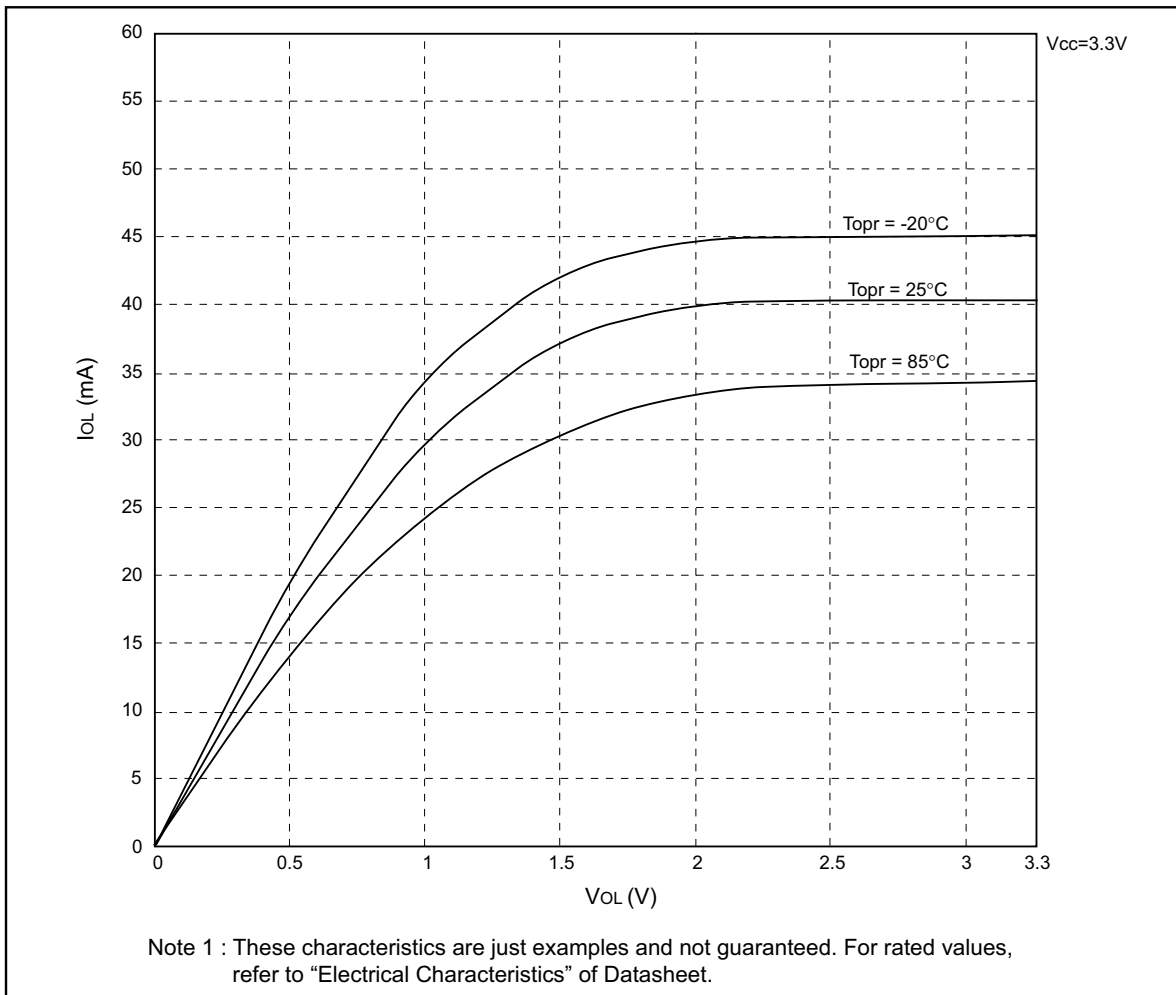


Figure 5.1.4. VOL-IOL standard characteristics of ports P63, P67, and P7 (when high drive selected)

### 5.1.2 Vcc-ICC Characteristics

Figure 5.1.5 and Figure 5.1.6 show Vcc-ICC characteristics.

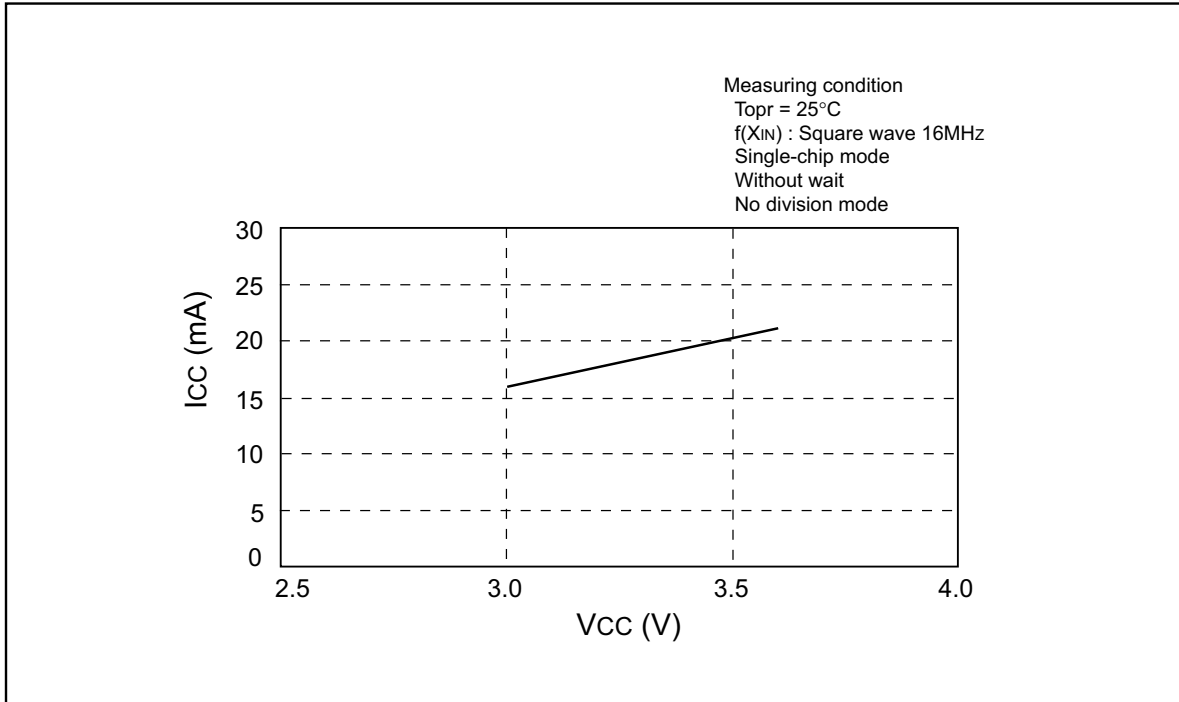


Figure 5.1.5. Vcc-icc characteristics (Mask version)

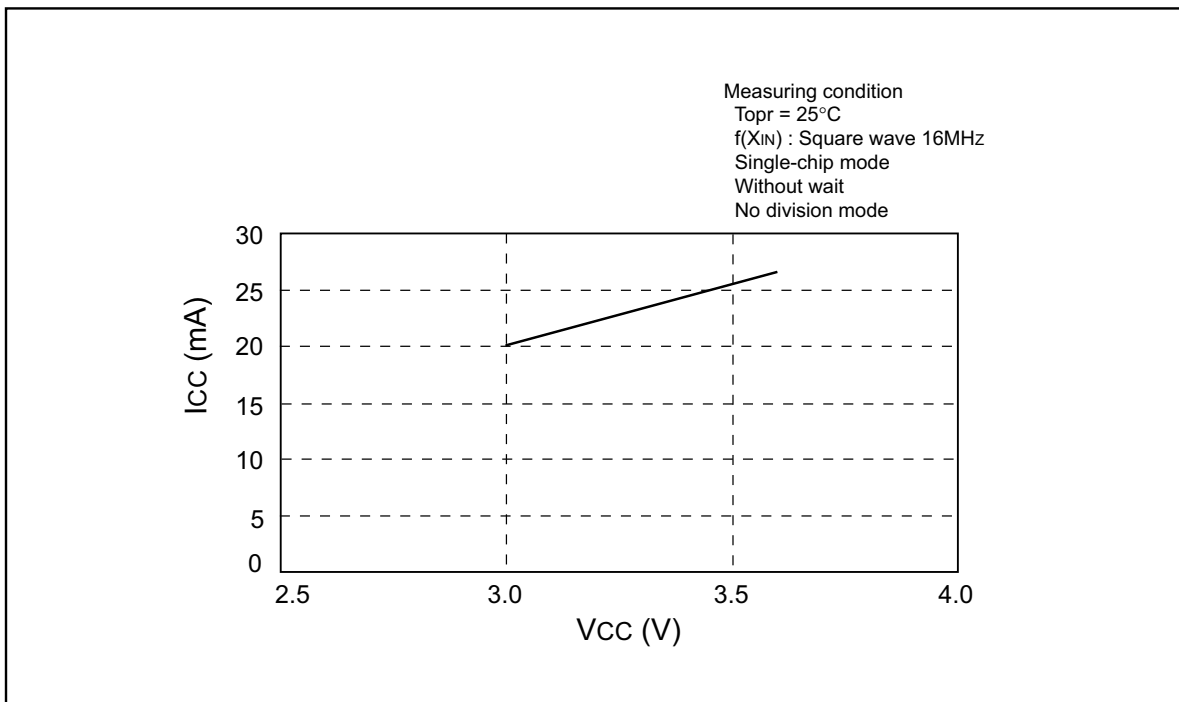


Figure 5.1.6. Vcc-icc characteristics (Flash version)

REVISION HISTORY	M30245 Group User's Manual
------------------	----------------------------

Rev.	Date	Description	
		Page	Summary
A	Jan 24, 2003	–	First edition issued
2.00	Oct 16, 2006	6 10 11 28 37 39 42 43 44  47 50 51  54 56 57 60 61 62  66 70  72 74 75 79  88 89 90  91 92 93 96 100  104 108  110-120	2.2.1 (c) "3 types" → "2 types", "an external input signal" deleted Figure 2.2.4 UDF: bit 5-7; R "○" → "–" Figure 2.2.5 ONSF Note 3 added 2.2.10 deleted Figure 2.2.34 deleted 2.3.1 (3) "Error detection" revised Figure 2.3.2 UiRB Note 1 deleted, ABT; W "×" → "○" Figure 2.3.3 UiMR; When reset revised, Note 3 added Figure 2.3.4 UiC0: bit 5; "TxDi" → "TxDi/SDAi and SCLi" revised, Note 2 revised, Note 4 added UiCi Note 1 added Figure 2.3.6 UiC0: bit 5; "TxDi" → "TxDi/SDAi and SCLi" revised Figure 2.3.8 Example of operation revised Figure 2.3.9 UiMR: bit 0-2 "Set the corresponding ..... to "0" when receiving." → "Set the RxDi .... to "0" when receiving." revised UiC0: bit5; "TxDi" → "TxDi/SDAi and SCLi" revised Reception (3) revised Table 2.4.3 Overrun error Description; revised (c) LSB/MSB first select function added Figure 2.4.3 UiRB Note 1 deleted, ABT; W "×" → "○" Figure 2.4.4 UiMR; When reset revised, Note 3 added Figure 2.4.5 UiC0: bit 5; "TxDi" → "TxDi/SDAi and SCLi" revised, Note 2 revised, Note 4 added UiCi Note 1 added Figure 2.4.8 UiC0: bit 5; "TxDi" → "TxDi/SDAi and SCLi" revised Figure 2.4.11 UiMR Note 1 added, UiC0, bit 5; "TxDi" → "TxDi/SDAi and SCLi" revised 2.4.4 added Figure 2.4.13 Example of operation (when direct format) revised, Note 2 added Figure 2.4.14 UiC0: bit 5; "TxDi" → "TxDi/SDAi and SCLi" revised Figure 2.4.17 UiMR Note 1 added, UiC0, bit 5; "TxDi" → "TxDi/SDAi and SCLi" revised Figure 2.5.2 UiRB Note 1 deleted, ABT; W "×" → "○" Figure 2.5.3 UiMR; When reset revised, Note 3 added Figure 2.5.4 UiC0: bit 5; "TxDi" → "TxDi/SDAi and SCLi" revised, Note 2 revised, Note 4 added UiCi Note 1 added Figure 2.5.5 UiSMR: bit 7 revised, UiSMR2: bit 7 revised, Note 1 added Figure 2.5.6 UiSMR3: bit 5-7 revised, Note 4 "The amount of delay varies with the load on SCLi and SDAi pins." added Figure 2.5.7 Note 2 added Figure 2.5.9 UiC0: bit 5; "TxDi" → "TxDi/SDAi and SCLi" revised Figure 2.5.12 UiMR Note 1 added, UiC0: bit5; "TxDi" → "TxDi/SDAi and SCLi" revised Figure 2.5.15 UiC0: bit 5; "TxDi" → "TxDi/SDAi and SCLi" revised Figure 2.5.18 UiMR Note 1 added, UiC0: bit 5; "TxDi" → "TxDi/SDAi and SCLi" revised 2.6 added

REVISION HISTORY

M30245 Group User's Manual

Rev.	Date	Description	
		Page	Summary
2.00	Oct 16, 2006	122	Figure 2.7.3 FSC: bit 2, 1 revised
		124	Table 2.7.1 revised
		125	Table 2.7.3 revised
		138	2.8.2 •USB control register; "the minimum 250ns of delay" → "a minimum 187.5 ns of delay (three cycles of BCLK)"
		143	(2) Enable of USB Function Control Unit; 3,7,8 revised
		144	Figure 2.8.16 "Wait for 4 or more cycles of φ" deleted
		145	Figure 2.8.17 FSC: bit 2, 1 revised
		146	Figure 2.8.18 revised
		154	(2) USB Endpoint 0 Interrupt: "• The SETUP_END flag ....." added
		155	(3) USB Function Interrupt: "• The last ACK for control rea ...." added
		157	• Artificial SOF Function revised
		163	• USB Suspend Mode Control: 1 revised and 5, 6, Note added
		165	-Returning Routine of USB Function Control Unit: 5 deleted, "(other than the FSC)" → "(other than the USBC,USBAD,and frequency synthesizer related registers)"
		167	Figure 2.8.29 added
		174	• USB endpoint 0 MAXP register: "SET_DESCRIPTOR" → "GET_DESCRIPTOR"
		210	2.8.9 "(1) USB Communication" added, (2) Peripheral Circuit; "between the USB D+pin and the USB D-pin or" deleted
		211	Figure 2.8.51 revised
		212	(3) Register, Bit: "(addresses 030016 to 033C16, excepting FSC)" → "(other than the USBC,USBAD,and frequency synthesizer related registers)"
		250	Figure 2.11.2 CRCMR, CRCSAR; When reset revised CRCSAR Note added
		252	2.11.3 "The target SFRs include the .... the UART-related registers." → "The target SFRs include .... Serial Sound Interface-related registers."
		258	2.13.1 "When the external bus .... to the external areas." added
		274	2.16.1 (2) "Reducing the driving capacity .... in power consumption." deleted
		276	(3) revised, (4) added
		280	Figure 2.16.5 (3); "following JMP.B instruction" added
		281	Figure 2.16.6 "Insert at least four NOPs after the WAIT instruction is executed." → "Insert JMP.B instruction before .... NOPs after the WAIT instruction."
		282	2.16.4 (3) revised, (4) added
283	(d) deleted		
284	Table 2.17.1 revised		
286	Table 2.17.3 AVSS, VREF, USB D+, USB D-, LPF, VbusDTCT, Note 6 added		
287	Figure 2.17.2 PCR revised		
289	Figure 2.17.3 PD9; When reset revised		
303	3.3 deleted		
320	Figure 3.7.4 "The DMA transfer request from .... is transmit request state." → "The DMA transfer request from .... in transmit request state."		
327	Figure 3.9.3 revised		
329	Chapter 4 added		
351	Chapter 5 added		

---

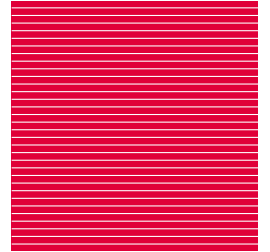
**RENESAS 16-BIT SINGLE-CHIP MICROCOMPUTER  
USER'S MANUAL  
M30245 Group**

**Publication Data :** Rev.A Jan 24, 2003  
Rev.2.00 Oct 16, 2006  
**Published by :** Sales Strategic Planning Div.  
Renesas Technology Corp.

---

© 2006. Renesas Technology Corp., All rights reserved. Printed in Japan.

# M30245 Group User's Manual



Renesas Technology Corp.  
2-6-2, Ote-machi, Chiyoda-ku, Tokyo, 100-0004, Japan