

8

# 7643 Group

User's Manual

RENESAS 8-BIT SINGLE-CHIP MICROCOMPUTER  
740 FAMILY / 7600 SERIES

User's 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>).

## Keep safety first in your circuit designs!

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

## Notes regarding these materials

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

## 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.

# BEFORE USING THIS MANUAL

This user's manual consists of the following three chapters. Refer to the chapter appropriate to your conditions, such as hardware design or software development. Chapter 3 also includes necessary information for systems development. You must refer to that chapter.

## 1. Organization

### ● CHAPTER 1 HARDWARE

This chapter describes features of the microcomputer and operation of each peripheral function.

### ● CHAPTER 2 APPLICATION

This chapter describes usage and application examples of peripheral functions, based mainly on setting examples of relevant registers.

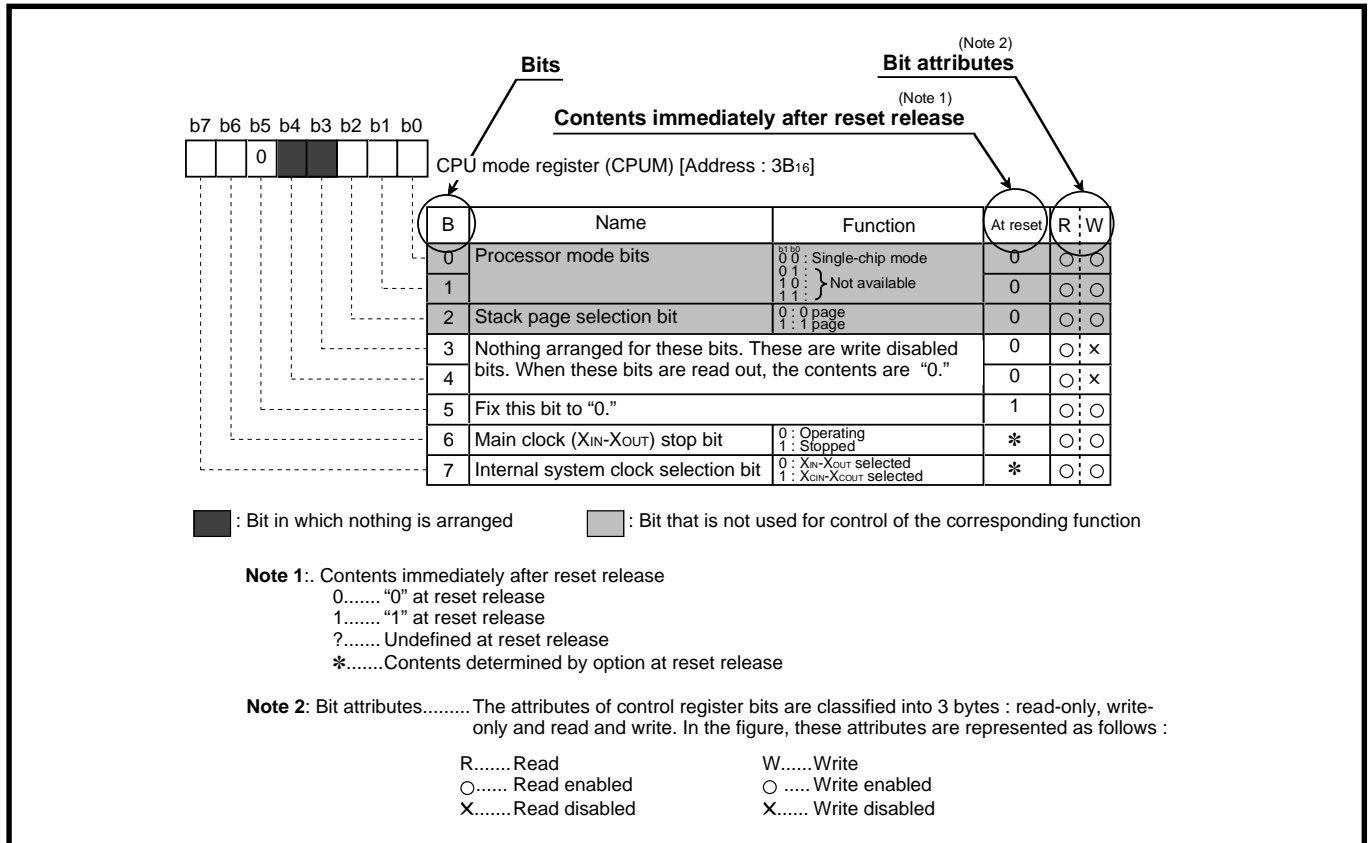
### ● CHAPTER 3 APPENDIX

This chapter includes necessary information for systems development using the microcomputer, such as the electrical characteristics, the notes, and the list of registers.

\*For the mask ROM confirmation form, the ROM programming confirmation form, and the mark specifications, refer to the "Renesas Technology" Homepage (<http://www.renesas.com/en/rom>).

## 2. Structure of register

The figure of each register structure describes its functions, contents at reset, and attributes as follows :



## 3. Supplementation

For details of development support tools, refer to the "Renesas Technology" Homepage (<http://www.renesas.com>).

# Table of contents

## CHAPTER 1 HARDWARE

<b>DESCRIPTION</b> .....	<b>2</b>
<b>FEATURES</b> .....	<b>2</b>
<b>APPLICATION</b> .....	<b>2</b>
<b>PIN CONFIGURATION (TOP VIEW)</b> .....	<b>3</b>
<b>FUNCTIONAL BLOCK DIAGRAM (Package: PRQP0080GB-A)</b> .....	<b>4</b>
<b>PIN DESCRIPTION</b> .....	<b>5</b>
<b>PART NUMBERING</b> .....	<b>7</b>
<b>GROUP EXPANSION</b> .....	<b>8</b>
Memory Type .....	
Memory Size .....	
Memory Expansion .....	8
Packages .....	8
<b>FUNCTIONAL DESCRIPTION</b> .....	<b>9</b>
CENTRAL PROCESSING UNIT (CPU) .....	9
MEMORY .....	13
I/O PORTS .....	15
INTERRUPTS .....	21
TIMERS .....	25
SERIAL I/O .....	27
UART .....	31
DMAC .....	37
USB FUNCTION .....	42
FREQUENCY SYNTHESIZER (PLL) .....	57
RESET CIRCUIT .....	59
CLOCK GENERATING CIRCUIT .....	61
PROCESSOR MODE .....	65
<b>FLASH MEMORY MODE</b> .....	<b>71</b>
<b>NOTES ON PROGRAMMING</b> .....	<b>98</b>
<b>USAGE NOTES</b> .....	<b>101</b>
<b>ROM ORDERING METHOD</b> .....	<b>102</b>
<b>FUNCTIONAL DESCRIPTION SUPPLEMENT</b> .....	<b>103</b>

## CHAPTER 2 APPLICATION

<b>2.1 I/O port</b> .....	<b>2</b>
2.1.1 Memory map .....	2
2.1.2 Related registers .....	3
2.1.3 Key-on wake-up interrupt application example .....	7
2.1.4 Terminate unused pins .....	9
2.1.5 Notes on I/O port .....	10
2.1.6 Termination of unused pins .....	11
<b>2.2 Timer</b> .....	<b>12</b>
2.2.1 Memory map .....	12
2.2.2 Related registers .....	13
2.2.3 Timer application examples .....	16
2.2.4 Notes on timer .....	22

<b>2.3 Serial I/O</b> .....	<b>23</b>
2.3.1 Memory map .....	23
2.3.2 Related registers .....	24
2.3.3 Serial I/O connection examples .....	27
2.3.4 Serial I/O application example .....	29
2.3.5 Notes on serial I/O .....	36
<b>2.4 UART</b> .....	<b>37</b>
2.4.1 Memory map .....	37
2.4.2 Related registers .....	38
2.4.3 UART transfer data format .....	45
2.4.4 Transfer bit rate .....	46
2.4.5 Operation of transmitting and receiving .....	47
2.4.6 UART application example .....	49
2.4.7 Notes on UART .....	73
<b>2.5 DMAC</b> .....	<b>75</b>
2.5.1 Memory map .....	75
2.5.2 Related registers .....	76
2.5.3 DMAC operation description .....	84
2.5.4 DMAC arbitration .....	88
2.5.5 Transfer time .....	88
2.5.6 DMAC application example .....	91
2.5.7 Notes on DMAC .....	95
<b>2.6 USB</b> .....	<b>96</b>
2.6.1 USB outline .....	96
2.6.2 Memory map .....	103
2.6.3 Related registers .....	104
2.6.4 USB transmit .....	124
2.6.5 USB receive .....	126
2.6.6 USB interrupts .....	127
2.6.7 Application example .....	128
2.6.8 Connection with other functions .....	148
2.6.9 Application circuit example .....	161
2.6.10 Notes on USB function .....	163
<b>2.7 Frequency synthesizer</b> .....	<b>167</b>
2.7.1 Memory map .....	167
2.7.2 Related registers .....	168
2.7.3 Functional description .....	171
2.7.4 Notes on frequency synthesizer .....	173
<b>2.8 External devices connection</b> .....	<b>174</b>
2.8.1 Memory map .....	174
2.8.2 Related registers .....	175
2.8.3 Functional description .....	176
2.8.4 Slow memory wait .....	177
2.8.5 HOLD function .....	180
2.8.6 Expanded data memory access .....	181
2.8.7 External devices connection example .....	182
2.8.8 Notes on external devices connection .....	186
<b>2.9 Reset</b> .....	<b>188</b>
2.9.1 Connection example of reset IC .....	188
2.9.2 Notes on reset .....	188

<b>2.10 Clock generating circuit</b> .....	<b>189</b>
2.10.1 Memory map .....	189
2.10.2 Related registers .....	190
2.10.3 Stop mode .....	193
2.10.4 Wait mode .....	194
2.10.5 Clock generating circuit application examples .....	195

## CHAPTER 3 APPENDIX

<b>3.1 Electrical characteristics</b> .....	<b>2</b>
3.1.1 Absolute maximum ratings .....	2
3.1.2 Recommended operating conditions (In $V_{CC} = 5\text{ V}$ ) .....	3
3.1.3 Electrical characteristics (In $V_{CC} = 5\text{ V}$ ) .....	4
3.1.4 Timing Requirements (In $V_{CC} = 5\text{ V}$ ) .....	6
3.1.5 Timing requirements and switching characteristics in memory expansion and microprocessor modes (In $V_{CC} = 5\text{ V}$ ) .....	7
3.1.6 Recommended Operating Conditions .....	8
3.1.7 Electrical Characteristics .....	9
3.1.8 Timing Requirements .....	11
3.1.9 Timing requirements and switching characteristics in memory expansion and microprocessor modes (In $V_{CC} = 3\text{ V}$ ) .....	12
<b>3.2 Standard characteristics</b> .....	<b>18</b>
3.2.1 Power source current standard characteristics .....	18
3.2.2 Port standard characteristics .....	19
<b>3.3 Notes on use</b> .....	<b>22</b>
3.3.1 Notes on interrupts .....	22
3.3.2 Notes on serial I/O .....	23
3.3.3 Notes on UART .....	24
3.3.4 Notes on DMAC .....	26
3.3.5 Notes on USB .....	27
3.3.6 Notes on frequency synthesizer .....	31
3.3.7 Notes on external devices connection .....	31
3.3.8 Notes on timer .....	33
3.3.9 Notes on Stop mode .....	33
3.3.10 Notes on reset .....	34
3.3.11 Notes on I/O port .....	34
3.3.12 Notes on programming .....	35
3.3.13 Termination of unused pins .....	37
3.3.14 Notes on CPU rewrite mode for flash memory version .....	38
<b>3.4 Countermeasures against noise</b> .....	<b>39</b>
3.4.1 Shortest wiring length .....	39
3.4.2 Connection of bypass capacitor across $V_{SS}$ line and $V_{CC}$ line .....	40
3.4.3 Oscillator concerns .....	41
3.4.4 Setup for I/O ports .....	42
3.4.5 Providing of watchdog timer function by software .....	43
<b>3.5 Control registers</b> .....	<b>44</b>
<b>3.6 Package outline</b> .....	<b>82</b>
<b>3.7 Machine instructions</b> .....	<b>84</b>
<b>3.8 List of instruction code</b> .....	<b>95</b>
<b>3.9 SFR memory map</b> .....	<b>96</b>
<b>3.10 Pin configuration</b> .....	<b>97</b>

# List of figures

## CHAPTER 1 HARDWARE

Fig. 1	M37643M8-XXXFP, M37643F8FP pin configuration .....	3
Fig. 2	M37643M8-XXXHP, M37643F8HP pin configuration .....	3
Fig. 3	Functional block diagram .....	4
Fig. 4	Part numbering .....	7
Fig. 5	Memory expansion .....	8
Fig. 6	7600 series CPU register structure .....	9
Fig. 7	Register push and pop at interrupt generation and subroutine call .....	10
Fig. 8	Structure of CPU mode register .....	12
Fig. 9	Memory map diagram .....	13
Fig. 10	Memory map of special function register (SFR) .....	14
Fig. 11	Structure of port control and port P2 pull-up control registers .....	15
Fig. 12	Port block diagram (1) .....	17
Fig. 13	Port block diagram (2) .....	18
Fig. 14	Port block diagram (3) .....	19
Fig. 15	Port block diagram (4) .....	20
Fig. 16	Interrupt control .....	21
Fig. 17	Structure of interrupt-related registers .....	23
Fig. 18	Connection example when using key input interrupt and port P2 block diagram ..	24
Fig. 19	Timer block diagram .....	25
Fig. 20	Structure of timer 123 mode register .....	26
Fig. 21	Structure of serial I/O control registers 1, 2 .....	27
Fig. 22	Block diagram of serial I/O .....	28
Fig. 23	Serial I/O timing .....	29
Fig. 24	UART block diagram .....	31
Fig. 25	UART transmit timing ( $\overline{\text{CTS}}$ function enabled) .....	32
Fig. 26	UART transmit timing (CTS function disabled) .....	33
Fig. 27	UART receiving timing (RTS function enabled) .....	33
Fig. 28	Structure of UART related registers .....	36
Fig. 29	DMACx (x = 0, 1) block diagram .....	37
Fig. 30	Structure of DMACx related register .....	38
Fig. 31	Timing chart for cycle steal transfer caused by hardware-related transfer request ...	40
Fig. 32	Timing chart for cycle steal transfer caused by software trigger transfer request .....	40
Fig. 33	Timing chart for burst transfer caused by hardware-related transfer request .....	41
Fig. 34	USB FCU (USB Function Control Unit) block .....	42
Fig. 35	Structure of USB control register .....	46
Fig. 36	Structure of USB address register .....	47
Fig. 37	Structure of USB power management register .....	47
Fig. 38	Structure of USB interrupt status register 1 .....	48
Fig. 39	Structure of USB interrupt status register 2 .....	49
Fig. 40	Structure of USB interrupt enable register 1 .....	49
Fig. 41	Structure of USB interrupt enable register 2 .....	50
Fig. 42	Structure of USB frame number registers .....	50
Fig. 43	Structure of USB endpoint 0 IN control register .....	51
Fig. 44	Structure of USB endpoint x (x = 1, 2) IN control register .....	52
Fig. 45	Structure of USB endpoint x (x = 1, 2) OUT control register .....	53



Fig. 46	Structure of USB endpoint x IN max. packet size register .....	54
Fig. 47	Structure of USB endpoint x OUT max. packet size register .....	54
Fig. 48	Structure of USB endpoint x (x = 0 to 2) OUT write count registers .....	55
Fig. 49	Structure of USB endpoint x (x = 0 to 2) FIFO register .....	55
Fig. 50	Structure of USB endpoint FIFO mode register .....	56
Fig. 51	Frequency synthesizer block diagram .....	57
Fig. 52	Structure of frequency synthesizer control register .....	58
Fig. 54	Reset sequence .....	59
Fig. 53	Reset circuit example .....	59
Fig. 55	Internal status at reset .....	60
Fig. 56	Ceramic resonator or quartz-crystal oscillator external circuit .....	61
Fig. 57	External clock input circuit .....	61
Fig. 58	Structure of clock control register .....	62
Fig. 59	Clock generating circuit block diagram .....	63
Fig. 60	State transitions of clock .....	64
Fig. 61	Memory maps in processor modes other than single-chip mode .....	65
Fig. 62	Structure of CPU mode register A .....	66
Fig. 63	Structure of CPU mode register B .....	66
Fig. 64	Software wait timing diagram .....	67
Fig. 65	RDY wait timing diagram .....	67
Fig. 66	Extended RDY wait (software wait plus RDY input anytime wait) timing diagram .....	68
Fig. 67	Hold function timing diagram .....	69
Fig. 68	STA (\$ zz), Y instruction sequence when EDMA enabled .....	70
Fig. 69	LDA (\$ zz), Y instruction sequence when EDMA enabled and T flag = "0" .....	70
Fig. 70	LDA (\$ zz), Y instruction sequence when EDMA enabled and T flag = "1" .....	70
Fig. 71	Block diagram of built-in flash memory .....	72
Fig. 72	Structure of flash memory control register .....	73
Fig. 73	CPU rewrite mode set/release flowchart .....	74
Fig. 74	Program flowchart .....	76
Fig. 75	Erase flowchart .....	77
Fig. 76	Full status check flowchart and remedial procedure for errors .....	79
Fig. 77	Structure of ROM code protect control .....	80
Fig. 78	ID code store addresses .....	81
Fig. 79	Pin connection diagram in standard serial I/O mode (1) .....	85
Fig. 80	Pin connection diagram in standard serial I/O mode (2) .....	86
Fig. 81	Timing for page read .....	88
Fig. 82	Timing for reading status register .....	88
Fig. 83	Timing for clear status register .....	89
Fig. 84	Timing for page program .....	89
Fig. 85	Timing for block erasing .....	90
Fig. 86	Timing for erase all blocks .....	90
Fig. 87	Timing for download .....	91
Fig. 88	Timing for version information output .....	92
Fig. 89	Timing for Boot ROM area output .....	92
Fig. 90	Timing for ID check .....	93
Fig. 91	ID code storage addresses .....	93
Fig. 92	Full status check flowchart and remedial procedure for errors .....	96
Fig. 93	Example circuit application for standard serial I/O mode .....	97
Fig. 94	Passive components near LPF pin .....	101
Fig. 95	Peripheral circuit .....	101
Fig. 96	Timing chart after interrupt occurs .....	103
Fig. 97	Time up to execution of interrupt processing routine .....	103

## CHAPTER 2 APPLICATION

Fig. 2.1.1 Memory map of registers related to I/O port .....	2
Fig. 2.1.2 Structure of Port Pi register .....	3
Fig. 2.1.3 Structure of Port P4, Port P7 registers .....	3
Fig. 2.1.4 Structure of Port Pi direction register (i = 0, 1, 2, 3, 5, 6, 8) .....	4
Fig. 2.1.5 Structure of Port P4 direction, Port P7 direction registers .....	4
Fig. 2.1.6 Structure of Port control register .....	5
Fig. 2.1.7 Structure of Port P2 pull-up control register .....	5
Fig. 2.1.8 Structure of Interrupt request register C .....	6
Fig. 2.1.9 Structure of Interrupt control register C .....	6
Fig. 2.1.10 Registers setting .....	7
Fig. 2.1.11 Connection diagram .....	8
Fig. 2.1.12 Control procedure .....	8
Fig. 2.2.1 Memory map of registers relevant to timers .....	12
Fig. 2.2.2 Structure of Timer i (i=1, 2, 3) .....	13
Fig. 2.2.3 Structure of Timer 123 mode register .....	13
Fig. 2.2.4 Structure of Interrupt request register B .....	14
Fig. 2.2.5 Structure of Interrupt request register C .....	14
Fig. 2.2.6 Structure of Interrupt control register B .....	15
Fig. 2.2.7 Structure of Interrupt control register C .....	15
Fig. 2.2.8 Timers connection and setting of division ratios .....	16
Fig. 2.2.9 Related registers setting .....	17
Fig. 2.2.10 Control procedure .....	18
Fig. 2.2.11 Peripheral circuit example .....	19
Fig. 2.2.12 Timers connection and setting of division ratios .....	19
Fig. 2.2.13 Relevant registers setting .....	20
Fig. 2.2.14 Control procedure .....	21
Fig. 2.3.1 Memory map of registers related to serial I/O .....	23
Fig. 2.3.2 Structure of Serial I/O shift register .....	24
Fig. 2.3.3 Structure of Serial I/O control register 1 .....	24
Fig. 2.3.4 Structure of Serial I/O control register 2 .....	25
Fig. 2.3.5 Structure of Interrupt request register C .....	26
Fig. 2.3.6 Structure of Interrupt control register C .....	26
Fig. 2.3.7 Serial I/O connection examples (1) .....	27
Fig. 2.3.8 Serial I/O connection examples (2) .....	28
Fig. 2.3.9 Connection diagram .....	29
Fig. 2.3.10 Timing chart .....	29
Fig. 2.3.11 Registers setting for transmitter .....	30
Fig. 2.3.12 Setting of serial I/O transmission data .....	30
Fig. 2.3.13 Control procedure of transmitter .....	31
Fig. 2.3.14 Connection diagram .....	32
Fig. 2.3.15 Registers setting for SPI compatible mode .....	33
Fig. 2.3.16 Control procedure of SPI compatible mode in slave .....	34
Fig. 2.3.17 Control procedure of SPI compatible mode in master .....	35

Fig. 2.4.1	Memory map of registers related to UART .....	37
Fig. 2.4.2	Structure of UART mode register .....	38
Fig. 2.4.3	Structure of UART control register .....	39
Fig. 2.4.4	Structure of UART status register .....	40
Fig. 2.4.5	Structure of UART RTS control register .....	40
Fig. 2.4.6	Structure of UART baud rate generator .....	41
Fig. 2.4.7	Structure of UART transmit/receive buffer registers 1, 2 .....	42
Fig. 2.4.8	Structure of Interrupt request register A .....	43
Fig. 2.4.9	Structure of Interrupt request register B .....	43
Fig. 2.4.10	Structure of Interrupt control register A .....	44
Fig. 2.4.11	Structure of Interrupt control register B .....	44
Fig. 2.4.12	UART transfer data format .....	45
Fig. 2.4.13	Connection diagram .....	49
Fig. 2.4.14	Timing chart .....	49
Fig. 2.4.15	Registers setting for transmitter .....	50
Fig. 2.4.16	Registers setting for receiver (1) .....	51
Fig. 2.4.17	Registers setting for receiver (2) .....	52
Fig. 2.4.18	Control procedure of transmitter .....	53
Fig. 2.4.19	Control procedure of receiver .....	54
Fig. 2.4.20	Connection diagram .....	56
Fig. 2.4.21	Registers setting related to UART address mode .....	57
Fig. 2.4.22	Control procedure (1) .....	58
Fig. 2.4.23	Control procedure (2) .....	59
Fig. 2.4.24	Connection diagram .....	60
Fig. 2.4.25	Registers setting (1) .....	61
Fig. 2.4.26	Registers setting (2) .....	62
Fig. 2.4.27	Registers setting (3) .....	63
Fig. 2.4.28	Control procedure (1) .....	64
Fig. 2.4.29	Control procedure (2) .....	65
Fig. 2.4.30	Connection diagram .....	66
Fig. 2.4.31	Registers setting (1) .....	67
Fig. 2.4.32	Registers setting (2) .....	68
Fig. 2.4.33	Registers setting (3) .....	69
Fig. 2.4.34	Registers setting (4) .....	70
Fig. 2.4.35	Control procedure (1) .....	71
Fig. 2.4.36	Control procedure (2) .....	72
Fig. 2.5.1	Memory map of registers related to DMAC .....	75
Fig. 2.5.2	Structure of DMAC index and status register .....	76
Fig. 2.5.3	Structure of DMAC channel x (x = 0, 1) mode register 1 .....	77
Fig. 2.5.4	Structure of DMAC channel 0 mode register 2 .....	79
Fig. 2.5.5	Structure of DMAC channel 1 mode register 2 .....	80
Fig. 2.5.6	Structure of DMAC channel x source registers Low, High .....	81
Fig. 2.5.7	Structure of DMAC channel x destination registers Low, High .....	81
Fig. 2.5.8	Structure of DMAC channel x transfer count registers Low, High .....	82
Fig. 2.5.9	Structure of Interrupt request register A .....	83
Fig. 2.5.10	Structure of Interrupt control register A .....	83
Fig. 2.5.11	Transfer mode overview .....	84
Fig. 2.5.12	Basic operation of registers transferring .....	85

Fig. 2.5.13	Timing chart for cycle steal transfer caused by hardware-related transfer request	89
Fig. 2.5.14	Timing chart for cycle steal transfer caused by software trigger transfer request	89
Fig. 2.5.15	Timing chart for burst transfer caused by hardware-related transfer request	90
Fig. 2.5.16	Setting of relevant registers (1)	92
Fig. 2.5.17	Setting of relevant registers (2)	93
Fig. 2.5.18	Control procedure	94
Fig. 2.6.1	1 frame image	97
Fig. 2.6.2	Packet type	97
Fig. 2.6.3	Transaction format	98
Fig. 2.6.4	Communication sequence of control transfer	99
Fig. 2.6.5	Device state transition	102
Fig. 2.6.6	Memory map of registers related to USB	103
Fig. 2.6.7	Structure of USB control register	104
Fig. 2.6.8	Structure of USB address register	105
Fig. 2.6.9	Structure of USB power management register	106
Fig. 2.6.10	Structure of USB interrupt status register 1	107
Fig. 2.6.11	Structure of USB interrupt status register 2	108
Fig. 2.6.12	Structure of USB interrupt enable register 1, USB interrupt enable register 2	109
Fig. 2.6.13	Structure of USB endpoint index register	110
Fig. 2.6.14	Structure of USB endpoint x IN control register	115
Fig. 2.6.15	Structure of USB endpoint x (x=1 to 2) OUT control register	117
Fig. 2.6.16	Structure of USB endpoint x (x=0 to 2) IN max. packet size register	118
Fig. 2.6.17	Structure of USB endpoint x (x=0 to 2) OUT max. packet size register	119
Fig. 2.6.18	Structure of USB endpoint x (x=0 to 2) OUT write count register	120
Fig. 2.6.19	Structure of USB endpoint x (x=0 to 2) FIFO register	121
Fig. 2.6.20	Structure of USB endpoint FIFO mode register	121
Fig. 2.6.21	Structure of clock control register	122
Fig. 2.6.22	Structure of frequency synthesizer control register	123
Fig. 2.6.23	Frequency synthesizer connection and setting of division ratios	128
Fig. 2.6.24	Registers setting (1)	129
Fig. 2.6.25	Registers setting (2)	130
Fig. 2.6.26	Registers setting (3)	131
Fig. 2.6.27	Control procedure (1) (USB block initial setting)	132
Fig. 2.6.28	Control procedure (2) (USB block generating)	133
Fig. 2.6.29	Control procedure (3) (endpoint initial setting)	134
Fig. 2.6.30	Registers setting (1) (USB endpoint 1 transmit)	135
Fig. 2.6.31	Control procedure (USB endpoint 1 IN interrupt routine)	136
Fig. 2.6.32	Registers setting (USB endpoint 1 OUT receive)	137
Fig. 2.6.33	Control procedure (USB endpoint 1 OUT interrupt routine)	138
Fig. 2.6.34	Structure of SET_ADDRESS request	139
Fig. 2.6.35	Register setting (processing when SET_ADDRESS is received)	139
Fig. 2.6.36	Control procedure	140
Fig. 2.6.37	Register setting (USB function interrupt routine)	142
Fig. 2.6.38	Control procedure (USB function interrupt routine)	143
Fig. 2.6.39	Register setting (USB suspend interrupt)	144
Fig. 2.6.40	Control procedure (USB suspend interrupt routine)	145
Fig. 2.6.41	Register setting (USB resume interrupt)	146
Fig. 2.6.42	Control procedure (USB resume interrupt routine)	147
Fig. 2.6.43	Connection diagram	148
Fig. 2.6.44	Register setting (1)	149
Fig. 2.6.45	Register setting (2)	150
Fig. 2.6.46	Register setting (3)	151
Fig. 2.6.47	Control procedure (1)	152

Fig. 2.6.48 Control procedure (2) .....	153
Fig. 2.6.49 Connection diagram .....	154
Fig. 2.6.50 Register setting (1) .....	155
Fig. 2.6.51 Register setting (2) .....	156
Fig. 2.6.52 Register setting (3) .....	157
Fig. 2.6.53 Register setting (4) .....	158
Fig. 2.6.54 Control procedure (1) .....	159
Fig. 2.6.55 Control procedure (2) .....	160
Fig. 2.6.56 Electronic instrument application example .....	161
Fig. 2.6.57 Encryption sytem application example .....	162
Fig. 2.6.58 Peripheral circuit example .....	164
Fig. 2.6.59 LPF peripheral circuit .....	164
Fig. 2.6.60 Connection of insulation connector .....	164
Fig. 2.7.1 Memory map of registers related to frequency synthesizer .....	167
Fig. 2.7.2 Structure of CPU mode register A .....	168
Fig. 2.7.3 Structure of Frequency synthesizer control register .....	168
Fig. 2.7.4 Structure of Frequency synthesizer multiply register 1 .....	169
Fig. 2.7.5 Structure of Frequency synthesizer multiply register 2 .....	169
Fig. 2.7.6 Structure of Frequency synthesizer divide register .....	170
Fig. 2.7.7 Block diagram for frequency synthesizer circuit .....	171
Fig. 2.7.8 Frequency synthesizer multiply register 2 setting example .....	171
Fig. 2.7.9 Frequency synthesizer multiply register 1 setting example .....	172
Fig. 2.7.10 Frequency synthesizer divide register setting example .....	172
Fig. 2.8.1 Memory map of registers related to external devices connection .....	174
Fig. 2.8.2 Structure of CPU mode register A .....	175
Fig. 2.8.3 Structure of CPU mode register B .....	175
Fig. 2.8.4 Software wait timing example .....	177
Fig. 2.8.5 RDY wait timing example .....	178
Fig. 2.8.6 Extended RDY wait (software wait plus RDY input anytime wait) timing example .....	179
Fig. 2.8.7 Hold function timing diagram .....	180
Fig. 2.8.8 Connection example of memory access up to 256 Kbytes .....	181
Fig. 2.8.9 External ROM and RAM example .....	182
Fig. 2.8.10 RDY function use example .....	183
Fig. 2.8.11 Read cycle (OE access, SRAM) .....	184
Fig. 2.8.12 Read cycle (OE access, EPROM) .....	184
Fig. 2.8.13 Write cycle (W control, SRAM) .....	185
Fig. 2.9.1 RAM backup system .....	188
Fig. 2.10.1 Memory map of registers related to clock generating circuit .....	189
Fig. 2.10.2 Structure of CPU mode register A .....	190
Fig. 2.10.3 Structure of Clock control register .....	190
Fig. 2.10.4 Structure of Frequency synthesizer control register .....	191
Fig. 2.10.5 Structure of Frequency synthesizer multiply register 1 .....	191
Fig. 2.10.6 Structure of Frequency synthesizer multiply register 2 .....	192
Fig. 2.10.7 Structure of Frequency synthesizer divide register .....	192
Fig. 2.10.8 Connection diagram .....	195
Fig. 2.10.9 Status transition diagram during power failure .....	195
Fig. 2.10.10 Setting of relevant registers .....	196
Fig. 2.10.11 Control procedure .....	197
Fig. 2.10.12 Structure of clock counter .....	198
Fig. 2.10.13 Initial setting of relevant registers .....	199
Fig. 2.10.14 Setting of relevant registers after detecting power failure .....	200
Fig. 2.10.15 Control procedure (1) .....	201
Fig. 2.10.16 Control procedure (2) .....	202

## CHAPTER 3 APPENDIX

Fig. 3.1.1 Circuit for measuring output switching characteristics (1) .....	13
Fig. 3.1.2 Circuit for measuring output switching characteristics (2) .....	13
Fig. 3.1.3 Timing diagram (1) .....	14
Fig. 3.1.4 Timing diagram (2) .....	15
Fig. 3.1.5 Timing diagram (3) .....	15
Fig. 3.1.6 Timing diagram (4); Memory expansion and microprocessor modes .....	16
Fig. 3.1.7 Timing diagram (5); Memory expansion and microprocessor modes .....	17
Fig. 3.2.1 Power source current standard characteristics (Ta = 25 °C) .....	18
Fig. 3.2.2 CMOS output port P-channel side characteristics (Ta = 25 °C) .....	19
Fig. 3.2.3 CMOS output port P-channel side characteristics (Ta = 70 °C) .....	19
Fig. 3.2.4 CMOS output port N-channel side characteristics (Ta = 25 °C) .....	20
Fig. 3.2.5 CMOS output port N-channel side characteristics (Ta = 70 °C) .....	20
Fig. 3.2.6 Port P2 <sub>0</sub> –P2 <sub>7</sub> at pull-up characteristics (Ta = 25 °C) .....	21
Fig. 3.2.7 Port P2 <sub>0</sub> –P2 <sub>7</sub> at pull-up characteristics (Ta = 70 °C) .....	21
Fig. 3.3.1 Sequence of setting external interrupt active edge .....	22
Fig. 3.3.2 Circuit example for the proper positions of the peripheral components .....	28
Fig. 3.3.3 Passive components near LPF pin .....	28
Fig. 3.3.4 Insulation connector connection .....	28
Fig. 3.3.5 Initialization of processor status register .....	35
Fig. 3.3.6 Sequence of PLP instruction execution .....	36
Fig. 3.3.7 Stack memory contents after PHP instruction execution .....	36
Fig. 3.4.1 Wiring for the RESET pin .....	39
Fig. 3.4.2 Wiring for clock I/O pins .....	39
Fig. 3.4.3 Bypass capacitor across the Vss line and the Vcc line .....	40
Fig. 3.4.4 Wiring for a large current signal line .....	41
Fig. 3.4.5 Wiring for signal lines where potential levels change frequently .....	41
Fig. 3.4.6 Vss pattern on the underside of an oscillator .....	42
Fig. 3.4.7 Setup for I/O ports .....	42
Fig. 3.4.8 Watchdog timer by software .....	43
Fig. 3.5.1 Structure of CPU mode register A .....	44
Fig. 3.5.2 Structure of CPU mode register B .....	44
Fig. 3.5.3 Structure of Interrupt request register A .....	45
Fig. 3.5.4 Structure of Interrupt request register B .....	45
Fig. 3.5.5 Structure of Interrupt request register C .....	46
Fig. 3.5.6 Structure of Interrupt control register A .....	46
Fig. 3.5.7 Structure of Interrupt control register B .....	47
Fig. 3.5.8 Structure of Interrupt control register C .....	47
Fig. 3.5.9 Structure of Port Pi .....	48
Fig. 3.5.10 Structure of Port P4, Port P7 .....	48
Fig. 3.5.11 Structure of Port Pi direction register .....	49
Fig. 3.5.12 Structure of Port P4, Port P7 direction registers .....	49
Fig. 3.5.13 Structure of Port control register .....	50
Fig. 3.5.14 Structure of Interrupt polarity select register .....	50

Fig. 3.5.15 Structure of Port P2 pull-up control register .....	51
Fig. 3.5.16 Structure of USB control register .....	51
Fig. 3.5.17 Structure of Clock control register .....	52
Fig. 3.5.18 Structure of Timer i .....	53
Fig. 3.5.19 Structure of Timer 123 mode register .....	53
Fig. 3.5.20 Structure of Serial I/O shift register .....	54
Fig. 3.5.21 Structure of Serial I/O control register 1 .....	55
Fig. 3.5.22 Structure of Serial I/O control register 2 .....	55
Fig. 3.5.23 Structure of Timer UART mode register .....	56
Fig. 3.5.24 Structure of UART baud rate generator .....	56
Fig. 3.5.25 Structure of UART status register .....	57
Fig. 3.5.26 Structure of UART control register .....	57
Fig. 3.5.27 Structure of UART transmit/receive buffer registers 1, 2 .....	58
Fig. 3.5.28 Structure of UART RTS control register .....	59
Fig. 3.5.29 Structure of DMAC index and status register .....	60
Fig. 3.5.30 Structure of DMAC channel x mode register 1 (x = 0, 1) .....	61
Fig. 3.5.31 Structure of DMAC channel 0 mode register 2 .....	62
Fig. 3.5.32 Structure of DMAC channel 1 mode register 2 .....	63
Fig. 3.5.33 Structure of DMAC channel x source registers Low, High .....	64
Fig. 3.5.34 Structure of DMAC channel x destination registers Low, High .....	64
Fig. 3.5.35 Structure of DMAC channel x transfer count registers Low, High (x = 0, 1) ....	65
Fig. 3.5.36 Structure of USB address register .....	66
Fig. 3.5.37 Structure of USB power management register .....	67
Fig. 3.5.38 Structure of USB interrupt status register 1 .....	68
Fig. 3.5.39 Structure of USB interrupt status register 2 .....	69
Fig. 3.5.40 Structure of USB interrupt enable register 1 .....	70
Fig. 3.5.41 Structure of USB interrupt enable register 2 .....	70
Fig. 3.5.42 Structure of USB endpoint index register .....	71
Fig. 3.5.43 Structure of USB endpoint x IN control register .....	72
Fig. 3.5.44 Structure of USB endpoint x OUT control register (x = 1, 2) .....	73
Fig. 3.5.45 Structure of USB endpoint x IN max. packet size register (x = 0 to 2) .....	74
Fig. 3.5.46 Structure of USB endpoint x OUT max. packet size register (x = 0 to 2) .....	74
Fig. 3.5.47 Structure of USB endpoint x OUT write control register (x = 0 to 2) .....	75
Fig. 3.5.48 Structure of USB endpoint FIFO mode register .....	75
Fig. 3.5.49 Structure of USB endpoint x FIFO register (x = 0 to 2) .....	76
Fig. 3.5.50 Structure of Flash memory control register .....	77
Fig. 3.5.51 Structure of Frequency synthesizer control register .....	78
Fig. 3.5.52 Structure of Frequency synthesizer multiply register 1 .....	78
Fig. 3.5.53 Structure of Frequency synthesizer multiply register 2 .....	79
Fig. 3.5.54 Structure of Frequency synthesizer divide register .....	80
Fig. 3.5.55 Structure of ROM code protect control register .....	81

# List of tables

## CHAPTER 1 HARDWARE

Table 1	Pin description (1)	5
Table 2	Pin description (2)	6
Table 3	Support products	8
Table 4	Push and pop instructions of accumulator or processor status register	10
Table 5	Set and clear instructions of each bit of processor status register	11
Table 6	List of I/O port function	16
Table 7	Interrupt vector addresses and priority	22
Table 8	Port functions in memory expansion mode and microprocessor mode	65
Table 9	Summary of M37643F8 (flash memory version)	71
Table 10	List of software commands (CPU rewrite mode)	76
Table 11	Definition of each bit in status register (SRD)	78
Table 12	Description of pin function (Standard Serial I/O Mode)	84
Table 13	Software commands (Standard serial I/O mode)	87
Table 14	Definition of each bit of status register (SRD)	94
Table 15	Bits of which state might be changed owing to software write	99

## CHAPTER 2 APPLICATION

Table 2.1.1	Termination of unused pins	9
Table 2.4.1	Setting examples of baud rate generator values and transfer bit rate values (f = 12 MHz)	46
Table 2.4.3	Error flags set condition and how to clear error flags	48
Table 2.5.1	Address directions and examples of transfer result (1)	86
Table 2.5.2	Address directions and examples of transfer result (2)	87
Table 2.5.3	Priority to use bus	88
Table 2.6.1	USB PID list	98
Table 2.6.2	IN FIFO States	125
Table 2.6.3	Bits of which state might be changed owing to software write	166
Table 2.10.2	State in Wait mode	194

## CHAPTER 3 APPENDIX

Table 3.1.1	Absolute maximum ratings	2
Table 3.1.2	Recommended operating conditions	3
Table 3.1.3	Electrical characteristics (1)	4
Table 3.1.4	Electrical characteristics (2)	5
Table 3.1.5	Timing requirements	6
Table 3.1.6	Timing requirements and switching characteristics in memory expansion and microprocessor modes	7
Table 3.1.7	Recommended operating conditions	8
Table 3.1.8	Electrical characteristics (1)	9
Table 3.1.9	Electrical characteristics (2)	10
Table 3.1.10	Timing requirements	11
Table 3.1.11	Timing requirements and switching characteristics in memory expansion and microprocessor modes	12
Table 3.3.1	Bits of which state might be changed owing to software write	30



THIS PAGE IS BLANK FOR REASONS OF LAYOUT.

# **CHAPTER 1**

---

## **OVERVIEW**

**DESCRIPTION**  
**FEATURES**  
**APPLICATION**  
**PIN CONFIGURATION**  
**FUNCTIONAL BLOCK**  
**PIN DESCRIPTION**  
**PART NUMBERING**  
**GROUP EXPANSION**  
**FUNCTIONAL DESCRIPTION**  
**FLASH MEMORY MODE**  
**NOTES ON PROGRAMMING**  
**USAGE NOTES**  
**DATA REQUIRED FOR MASK ORDERS**  
**FUNCTIONAL DESCRIPTION SUPPLEMENT**

## DESCRIPTION

The 7643 group is the 8-bit microcomputer based on the 7600 series core (740 family core compatible) technology.

The 7643 group is designed for PC peripheral devices, including the USB, DMAC, Serial I/O, UART, Timer and so on.

## FEATURES

<Microcomputer mode>

- Basic machine-language instructions ..... 71
- Minimum instruction execution time ..... 83 ns  
(at 24 MHz oscillation frequency,  $\phi = 12$  MHz)
- Memory size
  - ROM ..... 32 Kbytes
  - RAM ..... 1 Kbytes
- Programmable input/output ports ..... 66
- Software pull-up resistors ..... Built-in
- Interrupts ..... 14 sources, 14 vectors  
(external 3 including Key input, internal 10, software 1)
- USB function control unit
  - Transceiver ..... Full-Speed USB2.0 specification
- Timers ..... 8-bit X 3 (Timers 1, 2, 3)
- Serial Interface
  - Serial I/O ..... 8-bit X 1
  - UART ..... 8-bit X 1
- DMAC ..... 2 channels
- Clock generating circuit ..... Built-in  
(connect to external ceramic resonator or quartz-crystal oscillator)
- Power source voltage
  - At 24 MHz oscillation frequency,  $\phi = 12$  MHz ..... 4.15 to 5.25 V
  - At 24 MHz oscillation frequency,  $\phi = 6$  MHz ..... 3.00 to 3.60 V
- Operating temperature range ..... -20 to 70°C
- Packages
  - FP ..... PRQP0080GB-A (80-pin QFP)
  - HP ..... PLQP0080KB-A (80-pin LQFP)

<Flash memory mode>

- Power source voltage
  - At 24 MHz oscillation frequency,  $\phi = 12$  MHz ..... 4.15 to 5.25 V
  - At 24 MHz oscillation frequency,  $\phi = 6$  MHz ..... 3.00 to 3.60 V
- Program/Erase voltage
  - .....  $V_{CC} = 4.50$  V to 5.25 V, or 3.00 V to 3.60 V
  - .....  $V_{PP} = 4.50$  V to 5.25 V
  - At 24 MHz oscillation frequency,  $\phi = 6$  MHz (See Table 20.)
- Memory size
  - Flash ROM ..... 32 Kbytes
  - RAM ..... 2.5 Kbytes
- Flash memory mode ..... 3 modes
  - Parallel I/O mode
  - Standard serial I/O mode
  - CPU rewrite mode
- Programming method ..... Programming in unit of byte
- Erasing method
  - Batch erasing
  - Block erasing
- Program/Erase control by software command
- Command number ..... 6 commands
- Number of times for programming/erasing ..... 100
- ROM code protection
  - Available in parallel I/O mode and standard serial I/O mode
- Operating temperature range (at programming/erasing) .....  
..... Normal temperature

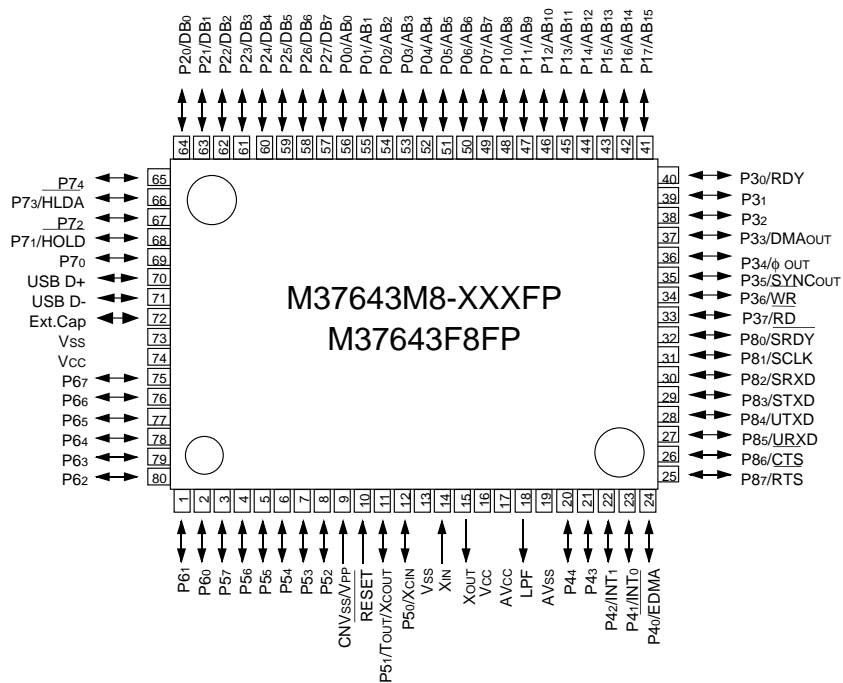
## APPLICATION

Audio, musical instrument, printer, scanner, modem, other PC peripheral devices

### ■Note

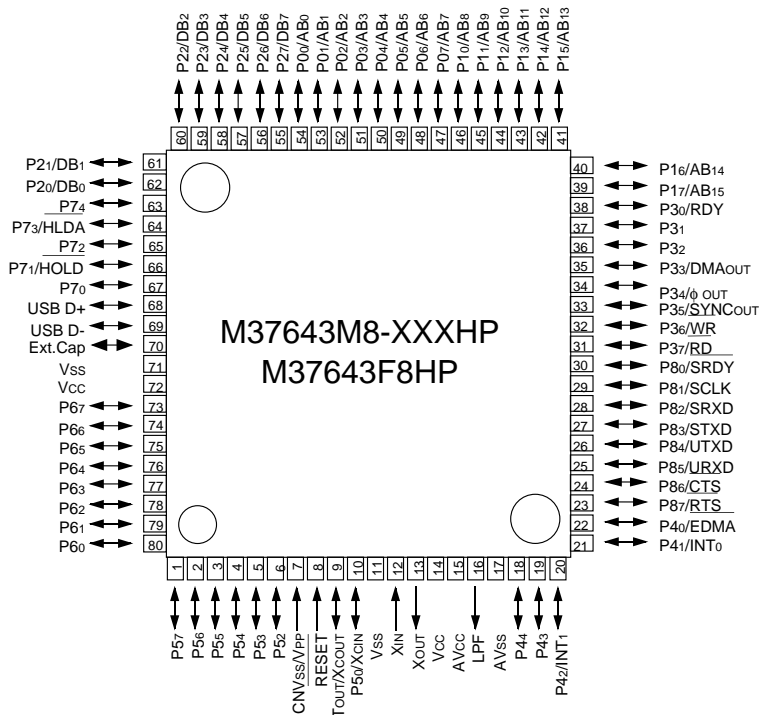
The flash memory version cannot be used for application embedded in the MCU card.

**PIN CONFIGURATION (TOP VIEW)**



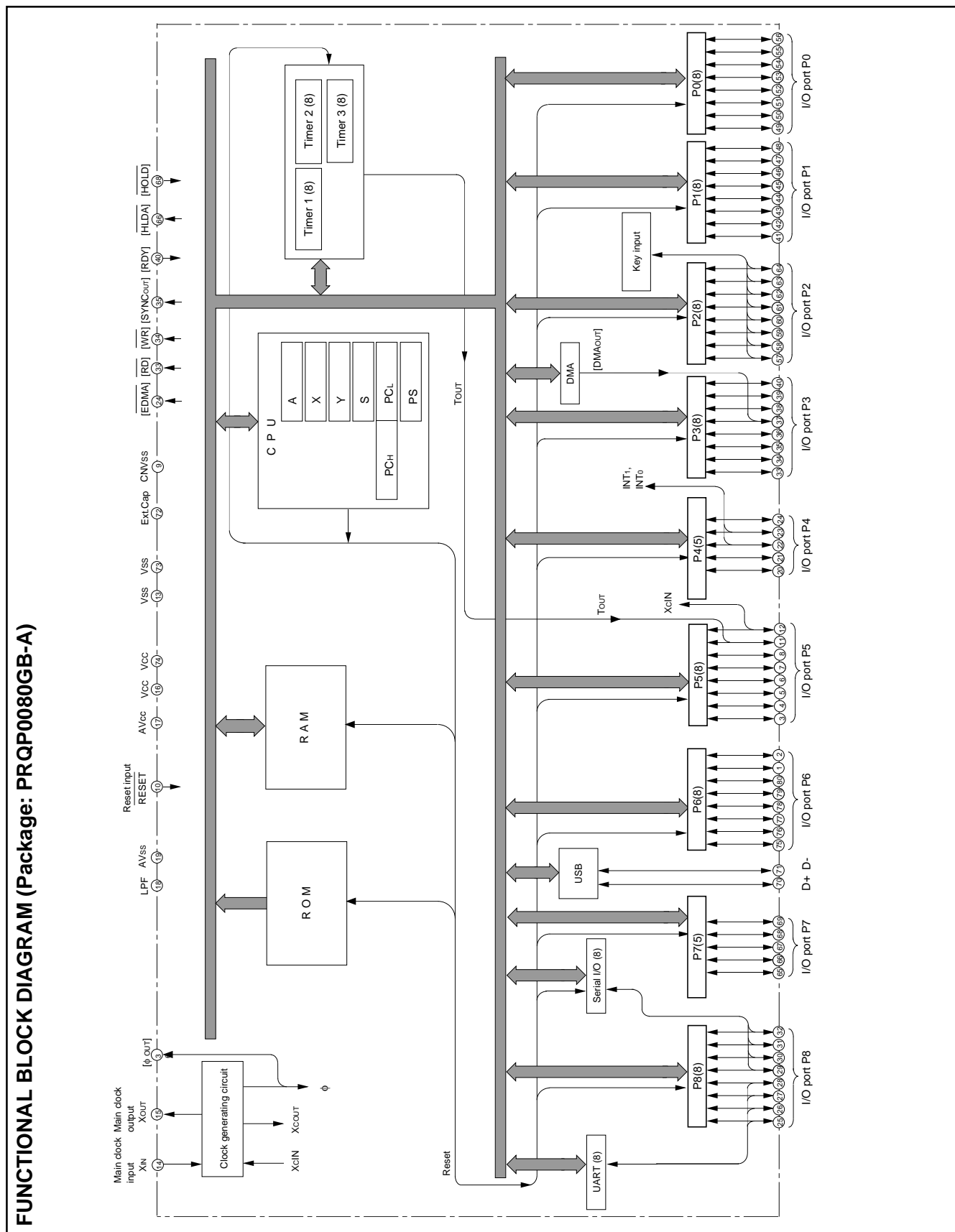
**Package type : PRQP0080GB-A (80P6N-A)**

**Fig. 1 M37643M8-XXXXFP, M37643F8FP pin configuration**



**Package type : PLQP0080KB-A (80P6Q-A)**

**Fig. 2 M37643M8-XXXXHP, M37643F8HP pin configuration**



**FUNCTIONAL BLOCK DIAGRAM (Package: PRQP0080GB-A)**

Fig. 3 Functional block diagram

## PIN DESCRIPTION

Table 1 Pin description (1)

Pin	Name	Function	
			Function except a port function
VCC, VSS	Power source	<ul style="list-style-type: none"> <li>Apply 4.15 V – 5.25 V for 5 V version or 3.00 V – 3.60 V for 3 V version to the Vcc pin. Apply 0 V to the Vss pin.</li> </ul>	
CNVss/VPP	CNVss	<ul style="list-style-type: none"> <li>This controls the MCU operating mode. Connect this pin to Vss. If connecting this pin to Vcc, the internal ROM is inhibited. In the flash memory version this pin functions as a VPP power supply input pin.</li> </ul>	
AVss/AVcc	Analog power supply	<ul style="list-style-type: none"> <li>These pins are the power supply inputs for analog circuitry.</li> </ul>	
RESET	Reset input	<ul style="list-style-type: none"> <li>Reset input pin for active “L.”</li> </ul>	
XIN	Clock input	<ul style="list-style-type: none"> <li>Connect a ceramic resonator or a quartz-crystal oscillator between the XIN and XOUT pins to set the oscillation frequency.</li> <li>If an external clock is used, connect the clock source to the XIN pin and leave the XOUT pin open.</li> </ul>	
XOUT	Clock output		
LPF	LPF	<ul style="list-style-type: none"> <li>Loop filter for the frequency synthesizer.</li> </ul>	
Ext. Cap.	3.3 V line power supply	<ul style="list-style-type: none"> <li>It is a capacitor connection pin for built-in DC-DC converter. At Vcc=5 V, use built-in DC-DC converter by permitting a USB line driver and connect a capacitor. Refer to "Notes on use" for details. Built-in DC-DC converter cannot be used at Vcc = 3.3 V. Supply 3.3V power supply to this pin from the externals.</li> </ul>	
USB D+	USB D+	<ul style="list-style-type: none"> <li>USB D+ voltage signal port. Connect a 27 to 33 Ω (recommended) resistor in series.</li> </ul>	
USB D-	USB D-	<ul style="list-style-type: none"> <li>USB D- voltage signal port. Connect a 27 to 33 Ω (recommended) resistor in series.</li> </ul>	
P00/AB0– P07/AB7	I/O port P0	<ul style="list-style-type: none"> <li>8-bit I/O port.</li> <li>CMOS compatible input level.</li> <li>CMOS 3-state output structure.</li> <li>I/O direction register allows each pin to be individually programmed as either input or output.</li> <li>When connecting an external memory, these function as the address bus.</li> </ul>	
P10/AB8– P17/AB15	I/O port P1	<ul style="list-style-type: none"> <li>8-bit I/O port.</li> <li>CMOS compatible input level.</li> <li>CMOS 3-state output structure.</li> <li>I/O direction register allows each pin to be individually programmed as either input or output.</li> <li>When connecting an external memory, these function as the address bus.</li> </ul>	
P20/DB0– P27/DB7	I/O port P2	<ul style="list-style-type: none"> <li>8-bit I/O port.</li> <li>CMOS compatible input level or VIH input level.</li> <li>CMOS 3-state output structure.</li> <li>I/O direction register allows each pin to be individually programmed as either input or output.</li> <li>When connecting an external memory, these function as the data bus.</li> </ul>	<ul style="list-style-type: none"> <li>Key-on wake-up interrupt input pin</li> </ul>
P30/RDY, P31, P32, P33/DMAOUT, P34/φ OUT, P35/SYNCOUT, P36/W $\bar{R}$ , P37/R $\bar{D}$	I/O port P3 (See Remarks.)	<ul style="list-style-type: none"> <li>8-bit I/O port.</li> <li>CMOS compatible input level.</li> <li>CMOS 3-state output structure.</li> <li>I/O direction register allows each pin to be individually programmed as either input or output.</li> <li>When connecting an external memory, these function as the control bus.</li> </ul>	
P40/EDMA, P41/INT0, P42/INT1, P43,P44	I/O port P4	<ul style="list-style-type: none"> <li>8-bit I/O port.</li> <li>CMOS compatible input level.</li> <li>CMOS 3-state output structure.</li> <li>I/O direction register allows each pin to be individually programmed as either input or output.</li> <li>When connecting an external memory, these function as the control bus.</li> </ul>	<ul style="list-style-type: none"> <li>External memory control pin</li> <li>External interrupt pin</li> </ul>
P50/XCIN, P51/TOUT/ XCOUT, P52,P53,P54, P55,P56,P57	I/O port P5	<ul style="list-style-type: none"> <li>8-bit I/O port.</li> <li>CMOS compatible input level.</li> <li>CMOS 3-state output structure.</li> <li>I/O direction register allows each pin to be individually programmed as either input or output.</li> </ul>	<ul style="list-style-type: none"> <li>Sub-clock generating input pin</li> <li>Timers 1, 2 pulse output pins</li> <li>Sub-clock generating output pin</li> </ul>

Table 2 Pin description (2)

Pin	Name	Function	
			Function except a port function
P60–P67	I/O port P6	<ul style="list-style-type: none"> <li>• 8-bit I/O port.</li> <li>• CMOS compatible input level.</li> <li>• CMOS 3-state output structure.</li> <li>• I/O direction register allows each pin to be individually programmed as either input or output.</li> </ul>	
P70, P71/HOLD, P72, P73/HLDA, P74	I/O port P7	<ul style="list-style-type: none"> <li>• 5-bit I/O port.</li> <li>• CMOS compatible input level.</li> <li>• CMOS 3-state output structure.</li> <li>• I/O direction register allows each pin to be individually programmed as either input or output.</li> </ul>	
P80/SRDY, P81/SCLK, P82/SRXD, P83/STXD, P84/UTXD, P85/URXD, P86/CTS, P87/RTS	I/O port P8	<ul style="list-style-type: none"> <li>• 8-bit I/O port.</li> <li>• CMOS compatible input level.</li> <li>• CMOS 3-state output structure.</li> <li>• I/O direction register allows each pin to be individually programmed as either input or output.</li> </ul>	<ul style="list-style-type: none"> <li>• Serial I/O pin</li> </ul>
			<ul style="list-style-type: none"> <li>• UART pin</li> </ul>

**Remarks**

## •DMAOUT pin

If externally detecting the timing of DMA execution, use the signal from this pin. It is "H" level during DMA transferring. This signal is valid in the memory expansion and microprocessor modes.

## •SYNCOUT pin

If externally detecting the timing of OP code fetch, use the signal from this pin. This signal is valid in the memory expansion and microprocessor modes.

## PART NUMBERING

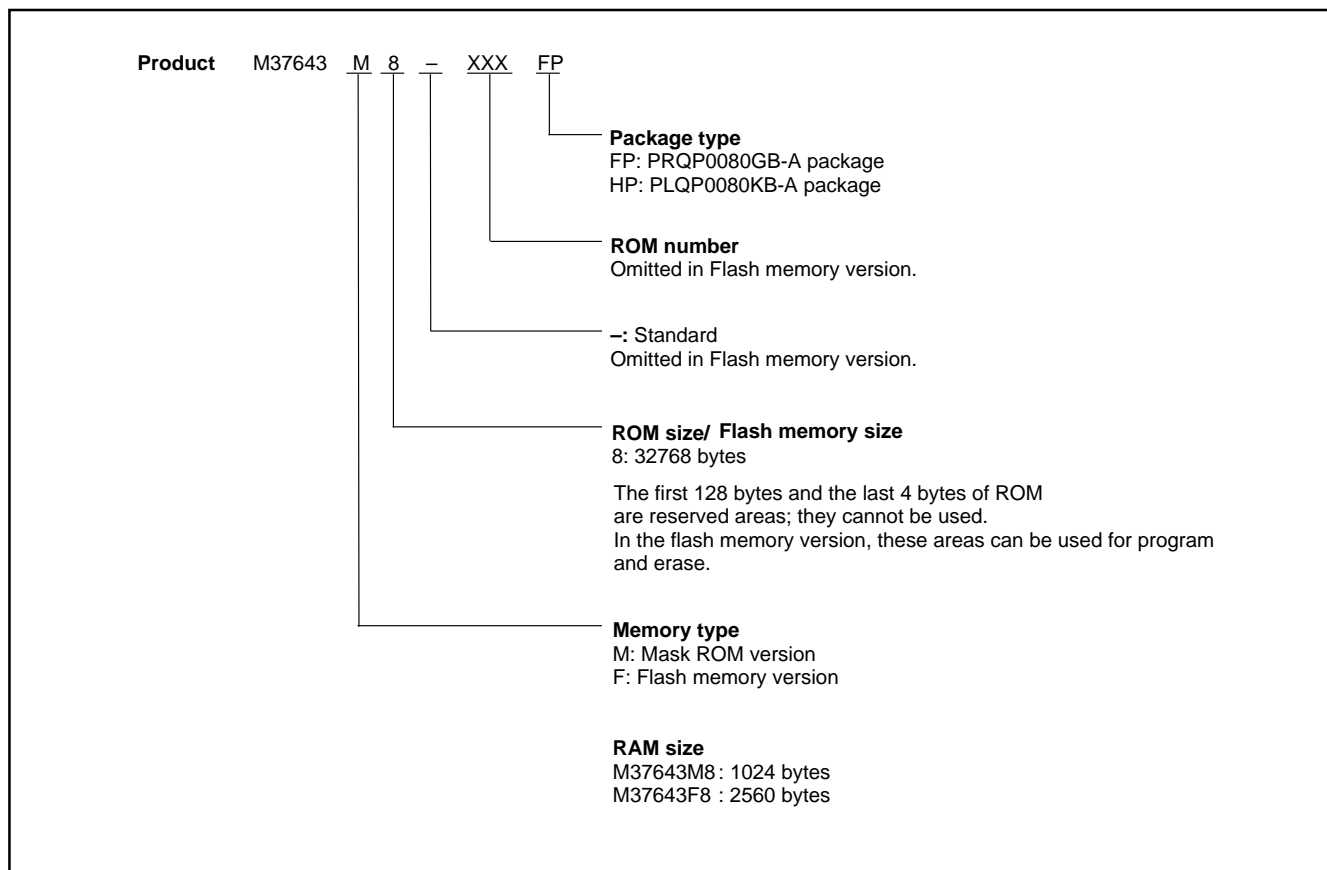


Fig. 4 Part numbering



**GROUP EXPANSION**

Renesas supports the 7643 group as follows.

**Memory Type**

Supports for mask ROM and flash memory versions.

**Memory Size**

ROM size ..... 32 Kbytes

RAM size ..... 1024 to 2560 bytes

**Packages**

PRQP0080GB-A ..... 0.8 mm-pitch plastic molded QFP

PLQP0080KB-A ..... 0.5 mm-pitch plastic molded LQFP

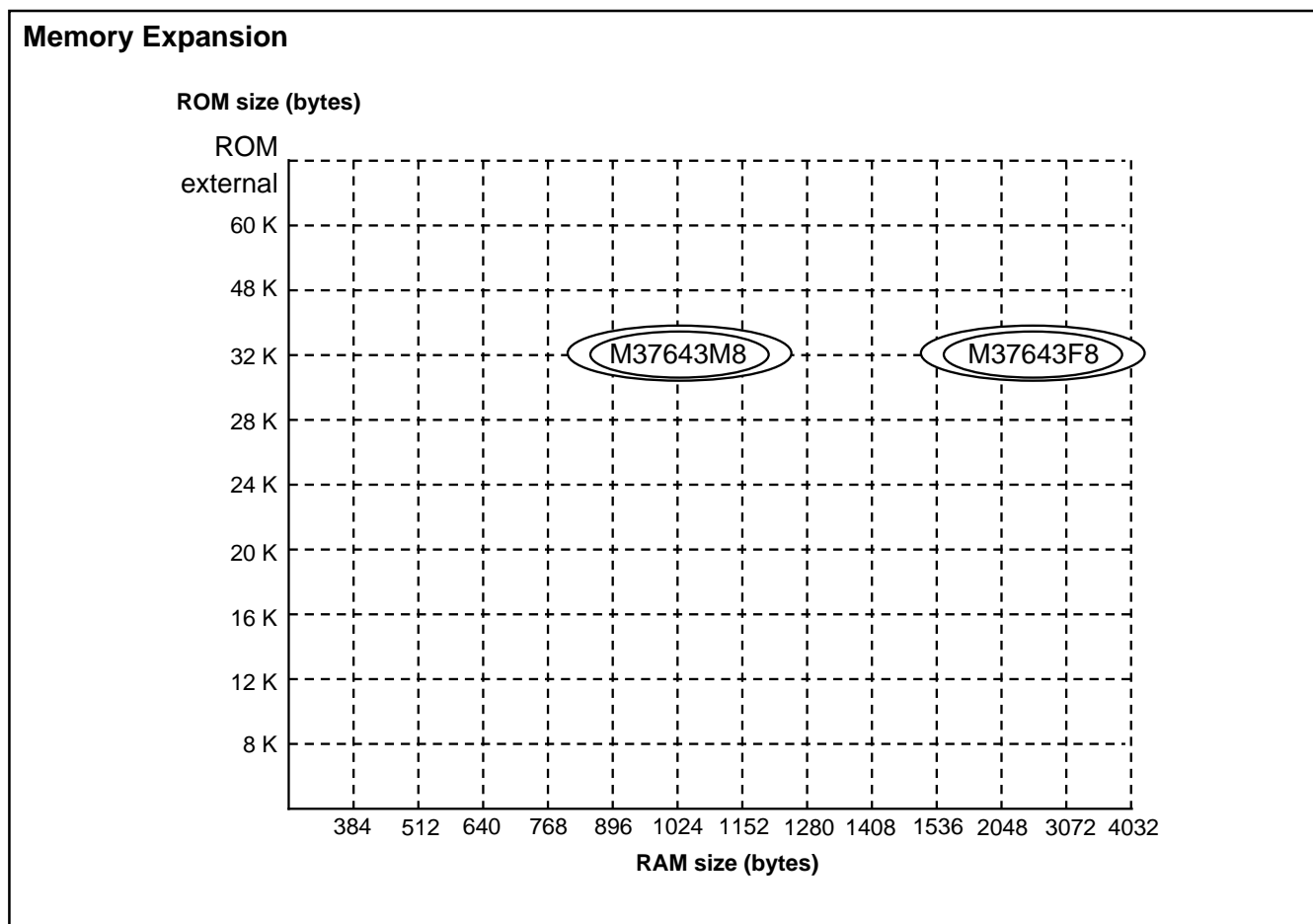


Fig. 5 Memory expansion

Currently supporting products are listed below.

Table 3 Support products

As of Aug. 2006

Part number	ROM size (bytes) ROM size for User in ( )	RAM size (bytes)	Package	Remarks
M37643M8-XXXFP	32768 (32636)	1024	PRQP0080GB-A	Mask ROM version
M37643M8-XXXHP			PLQP0080KB-A	
M37643F8FP	32768	2560	PRQP0080GB-A	Flash memory version
M37643F8HP			PLQP0080KB-A	

## FUNCTIONAL DESCRIPTION CENTRAL PROCESSING UNIT (CPU)

The 7643 group uses the standard 7600 series instruction set. Refer to the 7600 Series Software Manual for details on the instruction set. The 7600 series has an upward compatible instruction set, of which instruction execution cycles are shortened, for 740 series.

### [Accumulator (A)]

The accumulator is an 8-bit register. Data operations such as data transfer, etc., are executed mainly through the accumulator.

### [Index Register X (X)]

The index register X is an 8-bit register. In the index addressing modes, the value of the OPERAND is added to the contents of register X and specifies the real address.

### [Index Register Y (Y)]

The index register Y is an 8-bit register. In partial instruction, the value of the OPERAND is added to the contents of register Y and specifies the real address.

### [Stack Pointer (S)]

The stack pointer is an 8-bit register used during subroutine calls and interrupts. This register indicates start address of stored area (stack) for storing registers during subroutine calls and interrupts.

The low-order 8 bits of the stack address are determined by the contents of the stack pointer. The high-order 8 bits of the stack address are determined by the stack page selection bit. If the stack page selection bit is "0", the high-order 8 bits becomes "0016". If the stack page selection bit is "1", the high-order 8 bits becomes "0116".

The operations of pushing register contents onto the stack and popping them from the stack are shown in Figure 7.

Store registers other than those described in Figure 7 with program when the user needs them during interrupts or subroutine calls.

### [Program Counter (PC)]

The program counter is a 16-bit counter consisting of two 8-bit registers PCH and PCL. It is used to indicate the address of the next instruction to be executed.

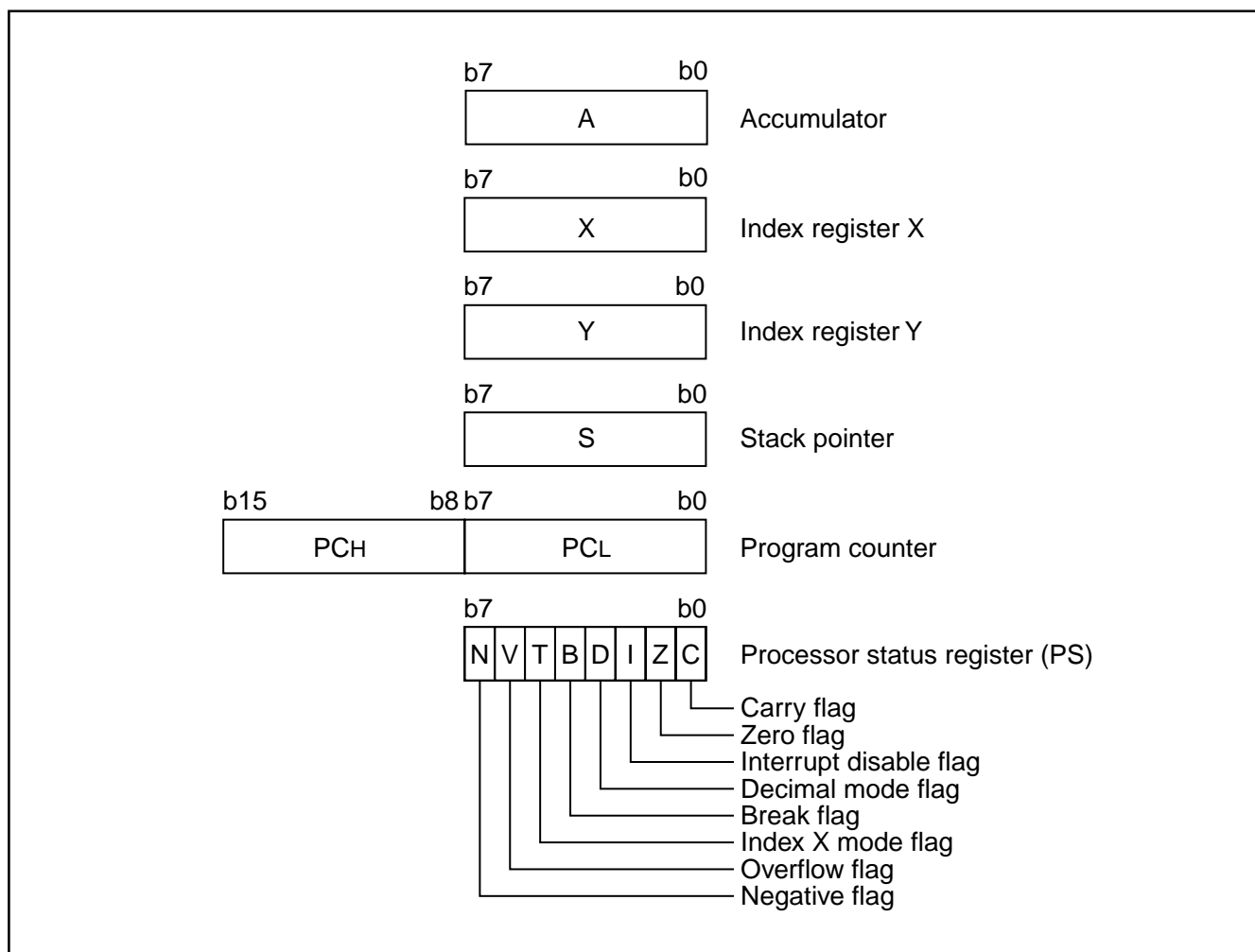


Fig. 6 7600 series CPU register structure

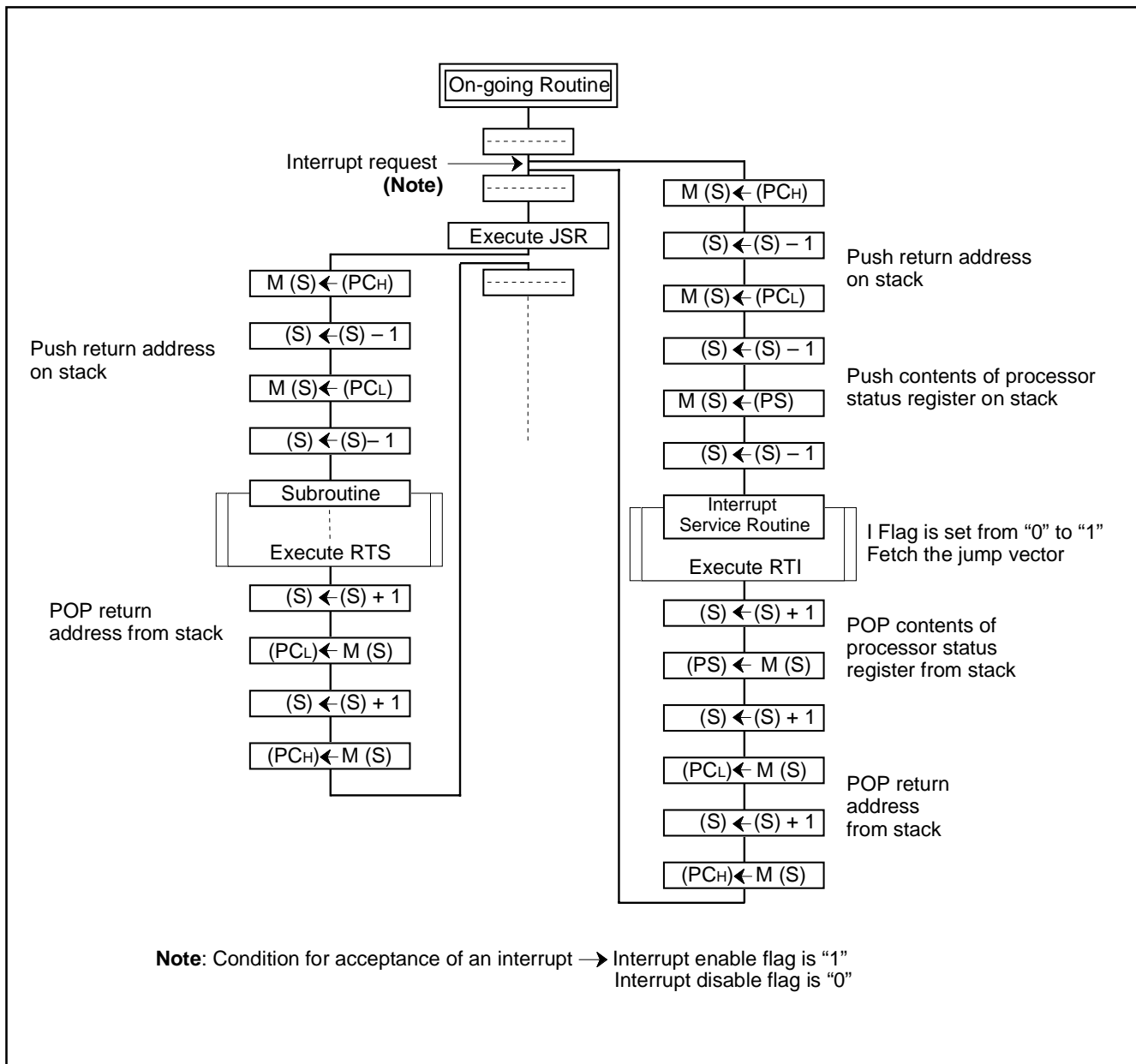


Fig. 7 Register push and pop at interrupt generation and subroutine call

Table 4 Push and pop instructions of accumulator or processor status register

	Push instruction to stack	Pop instruction from stack
Accumulator	PHA	PLA
Processor status register	PHP	PLP

**[Processor status register (PS)]**

The processor status register is an 8-bit register consisting of 5 flags which indicate the status of the processor after an arithmetic operation and 3 flags which decide MCU operation. Branch operations can be performed by testing the Carry (C) flag, Zero (Z) flag, Overflow (V) flag, or the Negative (N) flag. In decimal mode, the Z, V, N flags are not valid.

- Bit 0: Carry flag (C)

The C flag contains a carry or borrow generated by the arithmetic logic unit (ALU) immediately after an arithmetic operation. It can also be changed by a shift or rotate instruction.

- Bit 1: Zero flag (Z)

The Z flag is set if the result of an immediate arithmetic operation or a data transfer is "0", and cleared if the result is anything other than "0".

- Bit 2: Interrupt disable flag (I)

The I flag disables all interrupts except for the interrupt generated by the BRK instruction.

Interrupts are disabled when the I flag is "1".

- Bit 3: Decimal mode flag (D)

The D flag determines whether additions and subtractions are executed in binary or decimal. Binary arithmetic is executed when this flag is "0"; decimal arithmetic is executed when it is "1".

Decimal correction is automatic in decimal mode. Only the ADC and SBC instructions can execute decimal arithmetic.

- Bit 4: Break flag (B)

The B flag is used to indicate that the current interrupt was generated by the BRK instruction. The BRK flag in the processor status register is always "0". When the BRK instruction is used to generate an interrupt, the processor status register is pushed onto the stack with the break flag set to "1".

- Bit 5: Index X mode flag (T)

When the T flag is "0", arithmetic operations are performed between accumulator and memory. When the T flag is "1", direct arithmetic operations and direct data transfers are enabled between memory locations.

- Bit 6: Overflow flag (V)

The V flag is used during the addition or subtraction of one byte of signed data. It is set if the result exceeds +127 to -128. When the BIT instruction is executed, bit 6 of the memory location operated on by the BIT instruction is stored in the overflow flag.

- Bit 7: Negative flag (N)

The N flag is set if the result of an arithmetic operation or data transfer is negative. When the BIT instruction is executed, bit 7 of the memory location operated on by the BIT instruction is stored in the negative flag.

**Table 5 Set and clear instructions of each bit of processor status register**

	C flag	Z flag	I flag	D flag	B flag	T flag	V flag	N flag
Set instruction	SEC	–	SEI	SED	–	SET	–	–
Clear instruction	CLC	–	CLI	CLD	–	CLT	CLV	–

**[CPU Mode Registers A, B (CPUMA, CPUMB)] 0000<sub>16</sub>, 0001<sub>16</sub>**

The CPU mode register contains the stack page select bit and the CPU operating mode select bit and so on.

The CPU mode registers are allocated at address 0000<sub>16</sub>, 0001<sub>16</sub>.

**■ Notes**

Do not use the microprocessor mode in the flash memory version.

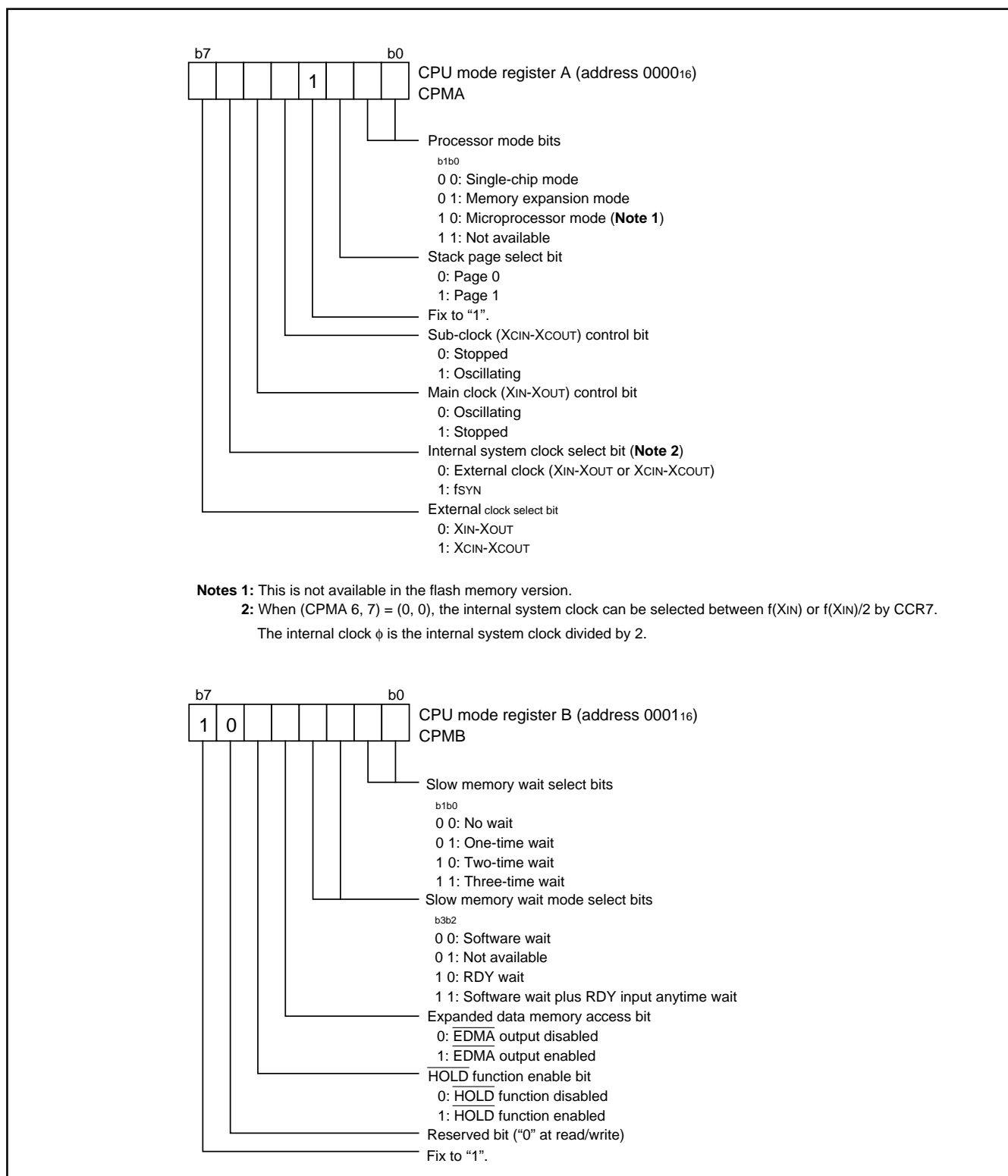


Fig. 8 Structure of CPU mode register

**MEMORY**

**Special Function Register (SFR) Area**

The Special Function Register area in the zero page contains control registers such as I/O ports and timers.

**RAM**

RAM is used for data storage and for stack area of subroutine calls and interrupts.

**ROM**

The first 128 bytes and the last 4 bytes of ROM are reserved for device testing and the rest is user area for storing programs. In the flash memory version, program and erase can be performed in the reserved area.

**Interrupt Vector Area**

The interrupt vector area contains reset and interrupt vectors.

**Zero Page**

Access to this area with only 2 bytes is possible in the zero page addressing mode.

**Special Page**

Access to this area with only 2 bytes is possible in the special page addressing mode.

Refer to page 74 for the memory map of memory expansion and microprocessor modes.

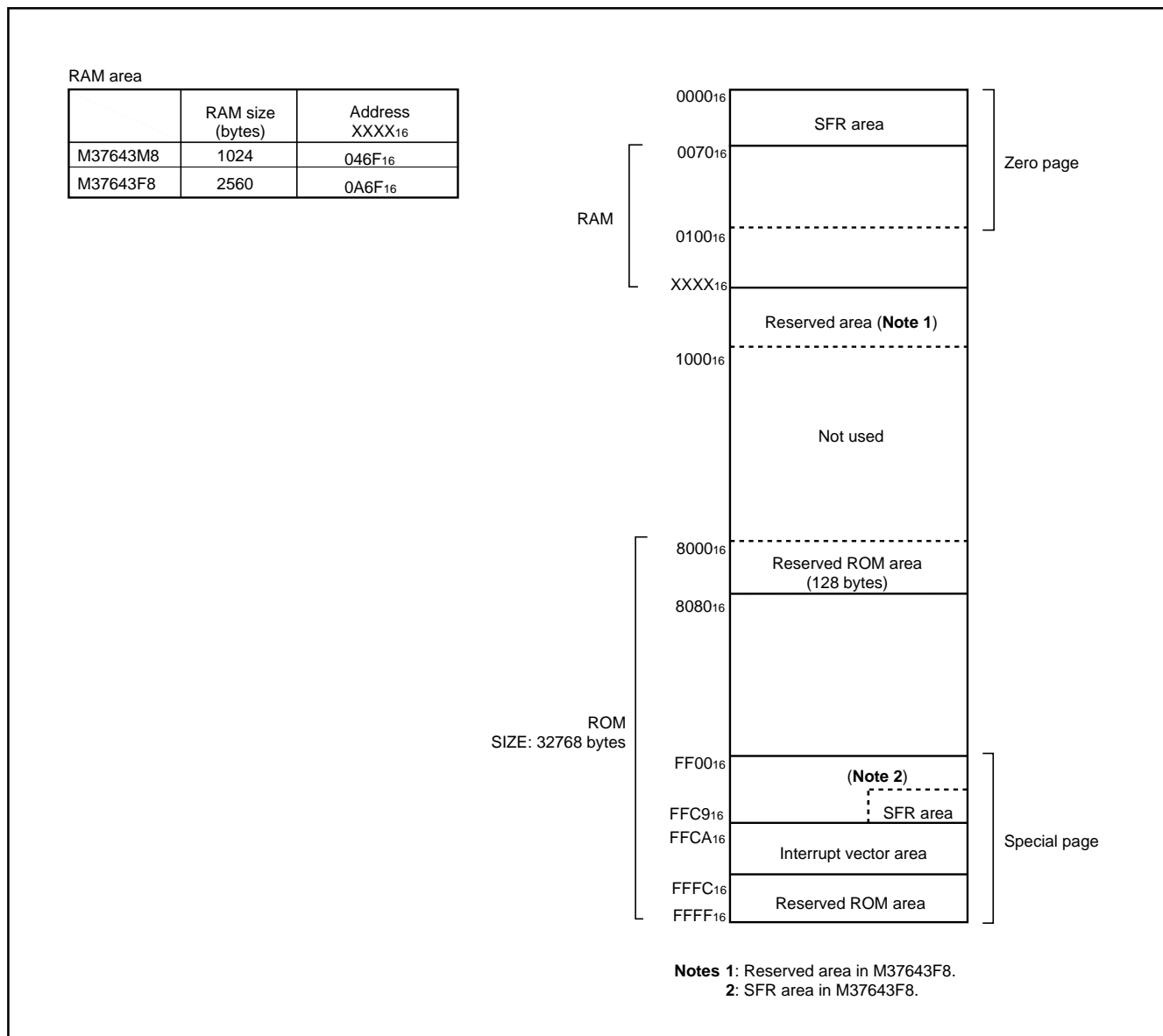


Fig. 9 Memory map diagram

0000 <sub>16</sub>	CPU mode register A (CPUA)	0038 <sub>16</sub>	Reserved ( <b>Note 1</b> )
0001 <sub>16</sub>	CPU mode register B (CPUB)	0039 <sub>16</sub>	Reserved ( <b>Note 1</b> )
0002 <sub>16</sub>	Interrupt request register A (IREQA)	003A <sub>16</sub>	Reserved ( <b>Note 1</b> )
0003 <sub>16</sub>	Interrupt request register B (IREQB)	003B <sub>16</sub>	Reserved ( <b>Note 1</b> )
0004 <sub>16</sub>	Interrupt request register C (IREQC)	003C <sub>16</sub>	Reserved ( <b>Note 1</b> )
0005 <sub>16</sub>	Interrupt control register A (ICONA)	003D <sub>16</sub>	Reserved ( <b>Note 1</b> )
0006 <sub>16</sub>	Interrupt control register B (ICONB)	003E <sub>16</sub>	Reserved ( <b>Note 1</b> )
0007 <sub>16</sub>	Interrupt control register C (ICONC)	003F <sub>16</sub>	DMAC index and status register (DMAIS)
0008 <sub>16</sub>	Port P0 (P0)	0040 <sub>16</sub>	DMAC channel x mode register 1 (DMAx1)
0009 <sub>16</sub>	Port P0 direction register (P0D)	0041 <sub>16</sub>	DMAC channel x mode register 2 (DMAx2)
000A <sub>16</sub>	Port P1 (P1)	0042 <sub>16</sub>	DMAC channel x source register Low (DMAxSL)
000B <sub>16</sub>	Port P1 direction register (P1D)	0043 <sub>16</sub>	DMAC channel x source register High (DMAxSH)
000C <sub>16</sub>	Port P2 (P2)	0044 <sub>16</sub>	DMAC channel x destination register Low (DMAxDL)
000D <sub>16</sub>	Port P2 direction register (P2D)	0045 <sub>16</sub>	DMAC channel x destination register High (DMAxDH)
000E <sub>16</sub>	Port P3 (P3)	0046 <sub>16</sub>	DMAC channel x transfer count register Low (DMAxCL)
000F <sub>16</sub>	Port P3 direction register (P3D)	0047 <sub>16</sub>	DMAC channel x transfer count register High (DMAxCH)
0010 <sub>16</sub>	Port control register (PTC)	0048 <sub>16</sub>	Reserved ( <b>Note 1</b> )
0011 <sub>16</sub>	Interrupt polarity select register (IPOL)	0049 <sub>16</sub>	Reserved ( <b>Note 1</b> )
0012 <sub>16</sub>	Port P2 pull-up control register (PUP2)	004A <sub>16</sub>	Reserved ( <b>Note 1</b> )
0013 <sub>16</sub>	USB control register (USBC)	004B <sub>16</sub>	Reserved ( <b>Note 1</b> )
0014 <sub>16</sub>	Port P6 (P6)	004C <sub>16</sub>	Reserved ( <b>Note 1</b> )
0015 <sub>16</sub>	Port P6 direction register (P6D)	004D <sub>16</sub>	Reserved ( <b>Note 1</b> )
0016 <sub>16</sub>	Port P5 (P5)	004E <sub>16</sub>	Reserved ( <b>Note 1</b> )
0017 <sub>16</sub>	Port P5 direction register (P5D)	004F <sub>16</sub>	Reserved ( <b>Note 1</b> )
0018 <sub>16</sub>	Port P4 (P4)	0050 <sub>16</sub>	USB address register (USBA)
0019 <sub>16</sub>	Port P4 direction register (P4D)	0051 <sub>16</sub>	USB power management register (USBPM)
001A <sub>16</sub>	Port P7 (P7)	0052 <sub>16</sub>	USB interrupt status register 1 (USBIS1)
001B <sub>16</sub>	Port P7 direction register (P7D)	0053 <sub>16</sub>	USB interrupt status register 2 (USBIS2)
001C <sub>16</sub>	Port P8 (P8)	0054 <sub>16</sub>	USB interrupt enable register 1 (USBIE1)
001D <sub>16</sub>	Port P8 direction register (P8D)	0055 <sub>16</sub>	USB interrupt enable register 2 (USBIE2)
001E <sub>16</sub>	Reserved ( <b>Note 1</b> )	0056 <sub>16</sub>	Reserved ( <b>Note 1</b> )
001F <sub>16</sub>	Clock control register (CCR)	0057 <sub>16</sub>	Reserved ( <b>Note 1</b> )
0020 <sub>16</sub>	Reserved ( <b>Note 1</b> )	0058 <sub>16</sub>	USB endpoint index register (USBINDEX)
0021 <sub>16</sub>	Reserved ( <b>Note 1</b> )	0059 <sub>16</sub>	USB endpoint x IN control register (IN_CSR)
0022 <sub>16</sub>	Reserved ( <b>Note 1</b> )	005A <sub>16</sub>	USB endpoint x OUT control register (OUT_CSR)
0023 <sub>16</sub>	Reserved ( <b>Note 1</b> )	005B <sub>16</sub>	USB endpoint x IN max. packet size register (IN_MAXP)
0024 <sub>16</sub>	Timer 1 (T1)	005C <sub>16</sub>	USB endpoint x OUT max. packet size register (OUT_MAXP)
0025 <sub>16</sub>	Timer 2 (T2)	005D <sub>16</sub>	USB endpoint x OUT write count register (WRT_CNT)
0026 <sub>16</sub>	Timer 3 (T3)	005E <sub>16</sub>	Reserved ( <b>Note 1</b> )
0027 <sub>16</sub>	Reserved ( <b>Note 1</b> )	005F <sub>16</sub>	USB endpoint FIFO mode register (USBFIFOMR)
0028 <sub>16</sub>	Reserved ( <b>Note 1</b> )	0060 <sub>16</sub>	USB endpoint 0 FIFO (USBFIFO0)
0029 <sub>16</sub>	Timer 123 mode register (T123M)	0061 <sub>16</sub>	USB endpoint 1 FIFO (USBFIFO1)
002A <sub>16</sub>	Serial I/O shift register (SIOSHT)	0062 <sub>16</sub>	USB endpoint 2 FIFO (USBFIFO2)
002B <sub>16</sub>	Serial I/O control register 1 (SIOCON1)	0063 <sub>16</sub>	Reserved ( <b>Note 1</b> )
002C <sub>16</sub>	Serial I/O control register 2 (SIOCON2)	0064 <sub>16</sub>	Reserved ( <b>Note 1</b> )
002D <sub>16</sub>	Reserved ( <b>Note 1</b> )	0065 <sub>16</sub>	Reserved ( <b>Note 1</b> )
002E <sub>16</sub>	Reserved ( <b>Note 1</b> )	0066 <sub>16</sub>	Reserved ( <b>Note 1</b> )
002F <sub>16</sub>	Reserved ( <b>Note 1</b> )	0067 <sub>16</sub>	Reserved ( <b>Note 1</b> )
0030 <sub>16</sub>	UART mode register (UMOD)	0068 <sub>16</sub>	Reserved ( <b>Note 1</b> )
0031 <sub>16</sub>	UART baud rate generator (UBRG)	0069 <sub>16</sub>	Reserved ( <b>Note 1</b> )
0032 <sub>16</sub>	UART status register (USTS)	006A <sub>16</sub>	Flash memory control register (FMCR) ( <b>Note 2</b> )
0033 <sub>16</sub>	UART control register (UCON)	006B <sub>16</sub>	Reserved ( <b>Note 1</b> )
0034 <sub>16</sub>	UART transmit/receive buffer register 1 (UTRB1)	006C <sub>16</sub>	Frequency synthesizer control register (FSC)
0035 <sub>16</sub>	UART transmit/receive buffer register 2 (UTRB2)	006D <sub>16</sub>	Frequency synthesizer multiply register 1 (FSM1)
0036 <sub>16</sub>	UART RTS control register (URTSC)	006E <sub>16</sub>	Frequency synthesizer multiply register 2 (FSM2)
0037 <sub>16</sub>	Reserved ( <b>Note 1</b> )	006F <sub>16</sub>	Frequency synthesizer divide register (FSD)
		FFC9 <sub>16</sub>	ROM code protect control register (ROMCP) ( <b>Note 3</b> )

**Notes 1:** Do not write any data to this addresses, because these areas are reserved.

**2:** This area is reserved in the mask ROM version.

**3:** This area is on the ROM in the mask ROM version.

Fig. 10 Memory map of special function register (SFR)

## I/O PORTS

### Direction Registers

The I/O ports P0–P8 have direction registers which determine the input/output direction of each individual pin. Each bit in a direction register corresponds to one pin, each pin can be set to be input port or output port.

When “0” is written to the bit corresponding to a pin, that pin becomes an input pin. When “1” is written to that bit, that pin becomes an output pin.

If data is read from a pin set to output, the value of the port output latch is read, not the value of the pin itself. Pins set to input are floating. If a pin set to input is written to, only the port output latch is written to and the pin remains floating.

### Slew Rate Control

By setting bits 0 to 5 of the port control register (address 0010<sub>16</sub>) to “1”, slew rate control is enabled. VIH<sub>L</sub> or CMOS level can be used as a port P2 input level.

### Pull-up Control

By setting the port P2 pull-up control register (address 0012<sub>16</sub>), pull-up of each pin of port P2 can be controlled with a program.

However, the contents of port P2 pull-up control register do not affect ports programmed as the output ports but as the input ports.

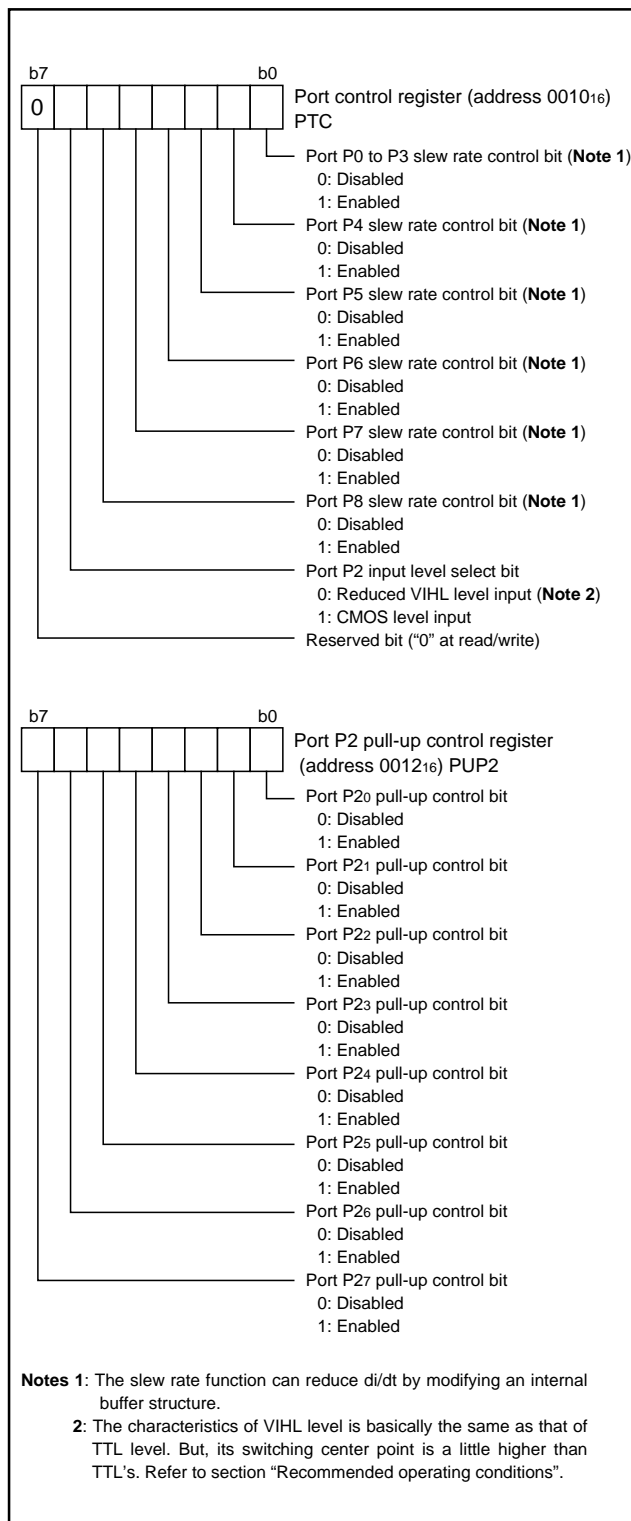


Fig. 11 Structure of port control and port P2 pull-up control registers



**Table 6 List of I/O port function**

Pin	Name	Input/Output	I/O format	Non-port function	Related SFRs	Ref. No.
P00/AB0– P07/AB7	Port P0	Input/Output, individual bits	CMOS input level CMOS 3-state output	Lower address output	CPU mode register A Port control register	(1)
P10/AB8– P17/AB15	Port P1			Higher address output		
P20/DB0– P27/DB7	Port P2		CMOS input level/VIHL input level CMOS 3-state output	Data bus I/O	CPU mode register A Port control register Port P2 pull-up control register	(2)
P30/RDY– P37/RD	Port P3		CMOS input level CMOS 3-state output	Control signal I/O	CPU mode register A CPU mode register B Port control register	(1) (3)
P40/EDMA, P41/INT0, P42/INT1, P43,P44	Port P4		CMOS input level CMOS 3-state output	Control signal I/O	CPU mode register A	(4)
External interrupt				CPU mode register B	(5)	
				Port control register Interrupt polarity select register		
P50/XCIN, P51/TOUT/ XCOUT	Port P5		CMOS input level CMOS 3-state output	Timer 1, Timer 2 output pin Sub-clock generat- ing input pin	CPU mode register A Port control register Clock control register Timer 123 mode register	(6) (7)
P52–P57		Port control register		(8)		
P60–P67	Port P6	CMOS input level/TTL input level CMOS 3-state output		Port control register	(9)	
P70,	Port P7	CMOS input level CMOS 3-state output		Port control register	(10)	
P71/HOLD, P72, P73/HLDA, P74			Control signal I/O	Port control register CPU mode register B	(11) (12) (13) (14)	
P80/SRDY, P81/SCLK, P82/SRXD, P83/STXD, P84/UTXD, P85/URXD, P86/CTS, P87/RTS	Port P8	CMOS input level CMOS 3-state output	Serial I/O I/O pin UART I/O pin	UART control registers Serial I/O control register 1 Serial I/O control register 2 Port control register	(15) (16) (17) (18) (19) (20) (21) (22)	

**Notes 1:** For details of the ports functions in modes other than single-chip mode, and how to use double-function ports as function I/O ports, refer to the applicable sections.

**2:** Make sure that the input level at each pin is either 0 V or V<sub>CC</sub> during execution of the STP instruction.

When an input level is at an intermediate potential, a rush current will flow from V<sub>CC</sub> to V<sub>SS</sub> through the input-stage gate.

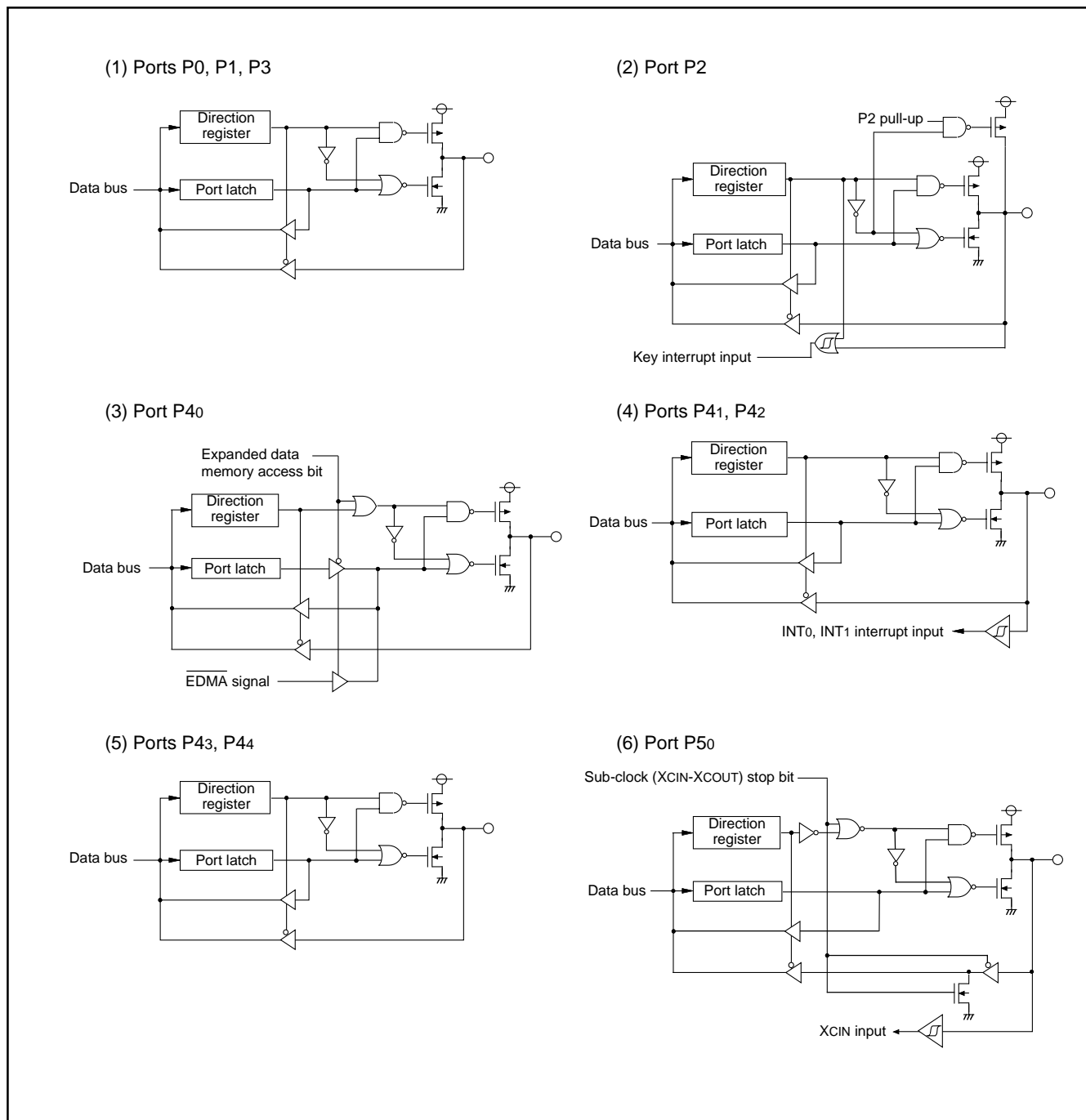


Fig. 12 Port block diagram (1)

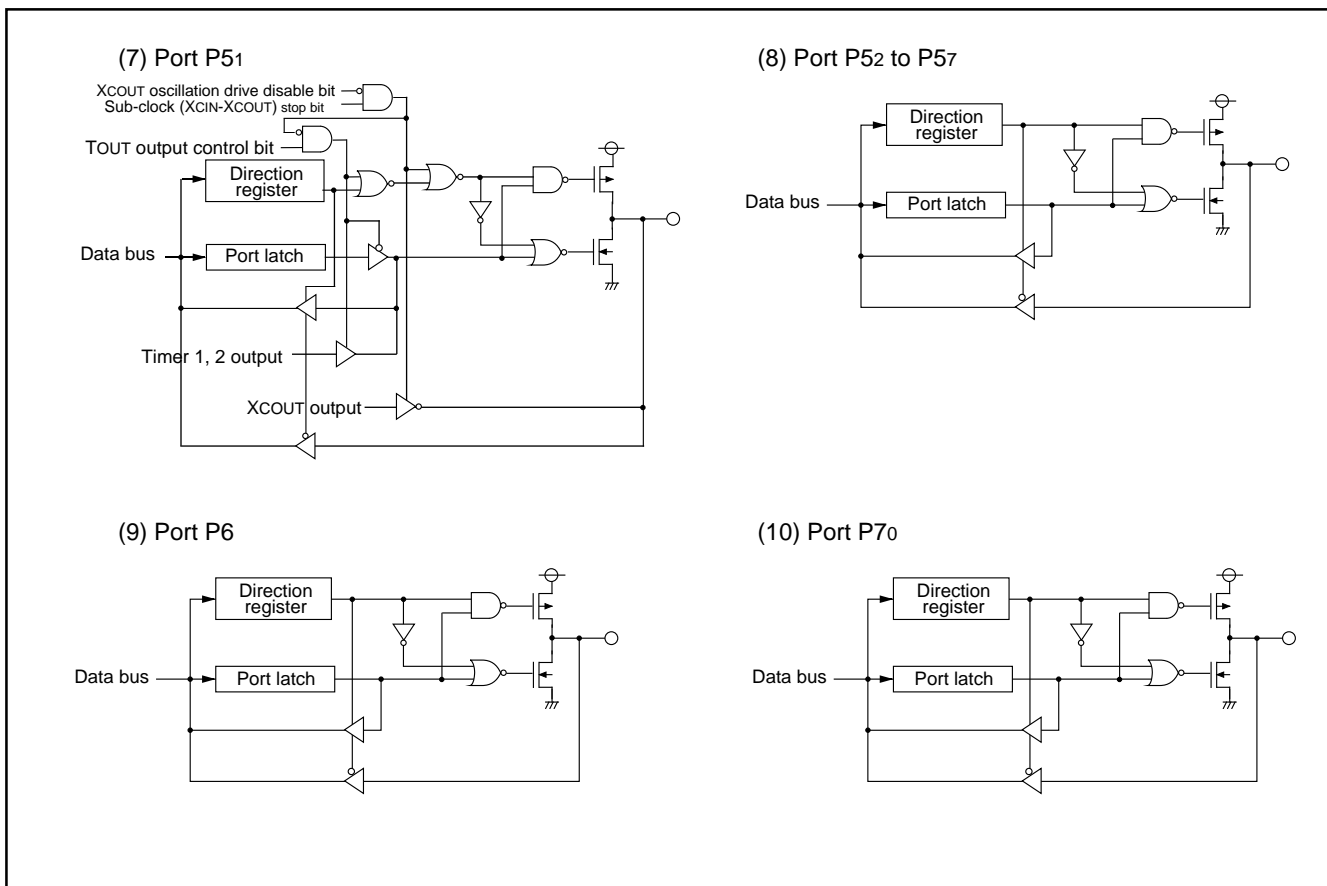


Fig. 13 Port block diagram (2)

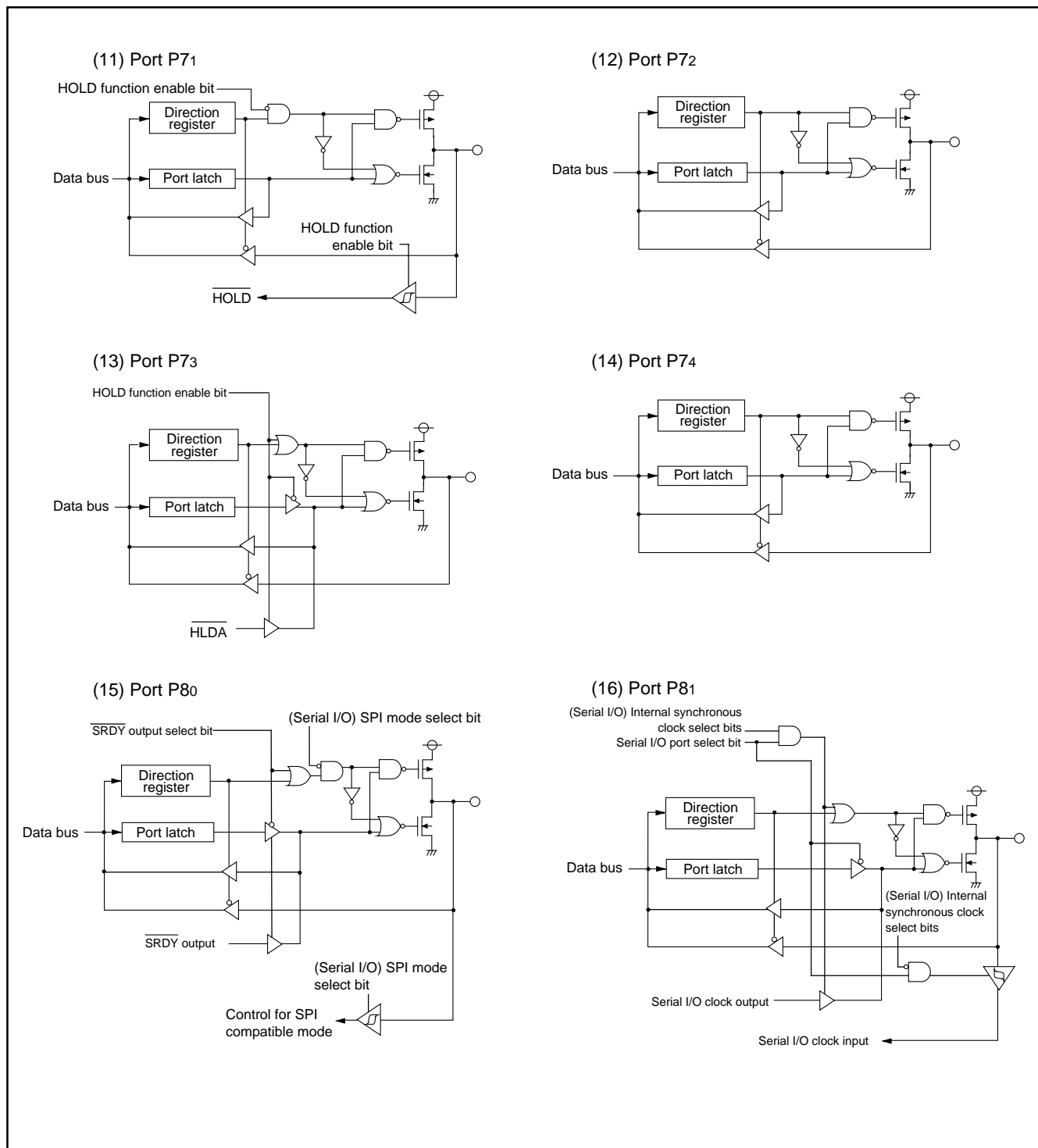


Fig. 14 Port block diagram (3)

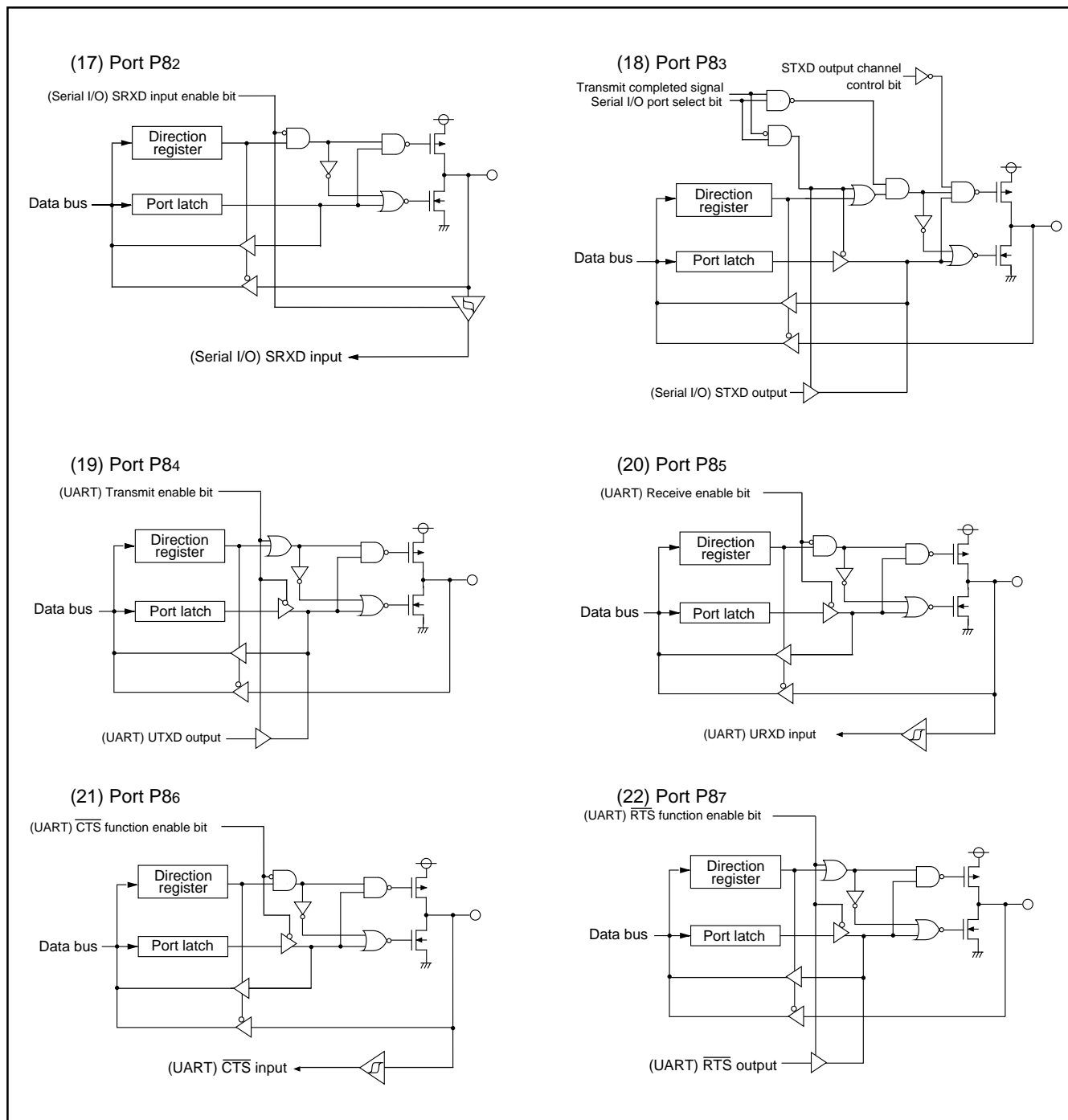


Fig. 15 Port block diagram (4)

## INTERRUPTS

There are fourteen interrupt sources: three externals, ten internals, and one software.

### Interrupt Control

Each interrupt except the BRK instruction interrupt has both an Interrupt Request Bit and an Interrupt Enable Bit, and is controlled by the Interrupt Disable Flag (I). An interrupt occurs if the corresponding Interrupt Request and Enable Bits are "1" and the Interrupt Disable Flag is "0".

Interrupt Enable Bits can be set or cleared by software. Interrupt Request Bits can be cleared by software, but cannot be set by software. Additionally, an active edge of INT0 and INT1 can be selected by using the interrupt polarity select register (address 001116).

The BRK instruction interrupt and reset cannot be disabled with any flag or bit. The I Flag disables all interrupts except the BRK instruction interrupt and reset. If several interrupts requests occur at the same time, the interrupt with the highest priority is accepted first.

### Interrupt Operation

When an interrupt request occurs, the following operations are automatically performed:

1. The processing being executed is stopped.
2. The contents of the program counter and processor status register are automatically pushed onto the stack.
3. The Interrupt Disable Flag is set and the corresponding interrupt request bit is cleared.
4. The interrupt jump destination address is read from the vector table into the program counter.

### Notes

When setting the followings, the interrupt request bit may be set to "1".

- When setting external interrupt active edge  
Related register: Interrupt polarity select register (address 001116)

When not requiring for the interrupt occurrence synchronized with these setting, take the following sequence.

- ①Set the corresponding Interrupt Enable Bit to "0" (disabled).
- ②Set the Interrupt Polarity Select Bit (Active Edge Switch Bit).
- ③Set the corresponding Interrupt Request Bit to "0" after 1 or more instructions have been executed.
- ④Set the corresponding Interrupt Enable Bit to "1" (enabled).

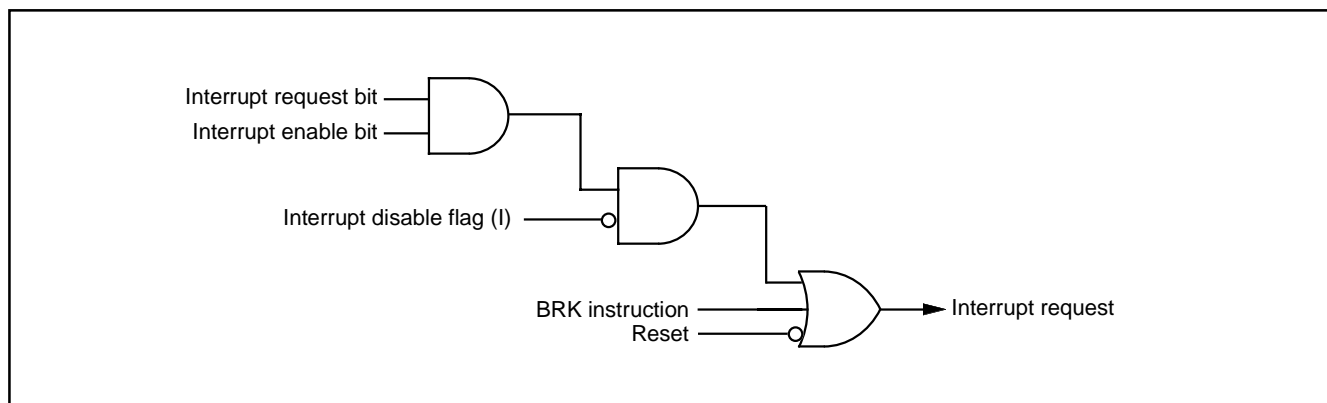


Fig. 16 Interrupt control

**Table 7 Interrupt vector addresses and priority**

Interrupt Source	Priority	Vector Addresses ( <b>Note 1</b> )		Interrupt Request Generating Conditions	Remarks
		High	Low		
Reset ( <b>Note 3</b> )	1	FFFB <sub>16</sub>	FFFA <sub>16</sub>	At reset	Non-maskable
USB function	2	FFF9 <sub>16</sub>	FFF8 <sub>16</sub>	( <b>Note 2</b> )	
Reserved area	-	FFF7 <sub>16</sub>	FFF6 <sub>16</sub>	Not used	
INT <sub>0</sub>	3	FFF5 <sub>16</sub>	FFF4 <sub>16</sub>	At detection of either rising or falling edge of INT <sub>0</sub> input	External interrupt (active edge selectable)
INT <sub>1</sub>	4	FFF3 <sub>16</sub>	FFF2 <sub>16</sub>	At detection of either rising or falling edge of INT <sub>1</sub> input	External interrupt (active edge selectable)
DMAC <sub>0</sub>	5	FFF1 <sub>16</sub>	FFF0 <sub>16</sub>	At completion of DMAC <sub>0</sub> transfer	
DMAC <sub>1</sub>	6	FFEF <sub>16</sub>	FFEE <sub>16</sub>	At completion of DMAC <sub>1</sub> transfer	
UART receive buffer full	7	FFED <sub>16</sub>	FFEC <sub>16</sub>	At completion of UART reception	
UART transmit	8	FFEB <sub>16</sub>	FFEA <sub>16</sub>	At completion of UART transmission	
UART summing error	9	FFE9 <sub>16</sub>	FFE8 <sub>16</sub>	At detection of UART summing error	
Reserved area	-	FFE7 <sub>16</sub>	FFE6 <sub>16</sub>	Not used	
Reserved area	-	FFE5 <sub>16</sub>	FFE4 <sub>16</sub>	Not used	
Reserved area	-	FFE3 <sub>16</sub>	FFE2 <sub>16</sub>	Not used	
Reserved area	-	FFE1 <sub>16</sub>	FFE0 <sub>16</sub>	Not used	
Reserved area	-	FFDF <sub>16</sub>	FFDE <sub>16</sub>	Not used	
Timer 1	10	FFDD <sub>16</sub>	FFDC <sub>16</sub>	At timer 1 underflow	
Timer 2	11	FFDB <sub>16</sub>	FFDA <sub>16</sub>	At timer 2 underflow	
Timer 3	12	FFD9 <sub>16</sub>	FFD8 <sub>16</sub>	At timer 3 underflow	
Reserved area	-	FFD7 <sub>16</sub>	FFD6 <sub>16</sub>	Not used	
Reserved area	-	FFD5 <sub>16</sub>	FFD4 <sub>16</sub>	Not used	
Serial I/O	13	FFD3 <sub>16</sub>	FFD2 <sub>16</sub>	At completion of serial I/O transmission/reception	
Reserved area	-	FFD1 <sub>16</sub>	FFD0 <sub>16</sub>	Not used	
Reserved area	-	FFCF <sub>16</sub>	FFCE <sub>16</sub>	Not used	
Key input (Key- on wake-up)	14	FFCD <sub>16</sub>	FFCC <sub>16</sub>	At falling of port P2 input logical level AND	External interrupt (falling valid)
BRK instruction	15	FFCB <sub>16</sub>	FFCA <sub>16</sub>	At BRK instruction execution	Non-maskable software interrupt

**Notes 1:** Vector addresses contain interrupt jump destination addresses.

**2:** USB function interrupt occurs owing to an interrupt request of the endpoint x (x = 0 to 2) IN, endpoint x (x = 1, 2) OUT, USB reset or suspend/resume.

**3:** Reset functions in the same way as an interrupt with the highest priority.

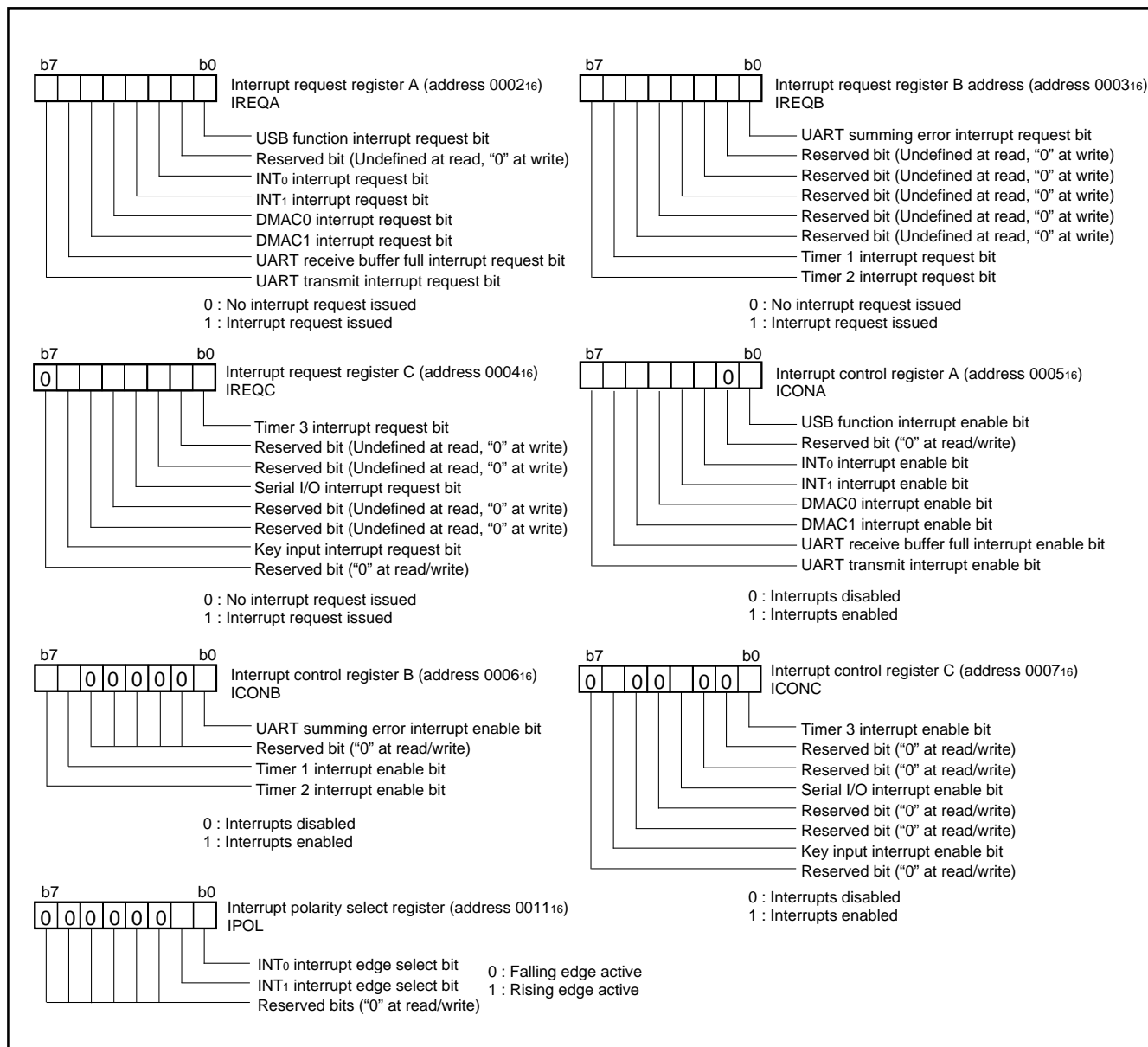


Fig. 17 Structure of interrupt-related registers



### Key Input Interrupt (Key-on Wake-Up)

A key input interrupt request is generated by applying "L" level to any pin of port P2 that have been set to input mode. In other words, it is generated when AND of input level goes from "1" to "0". An example

of using a key input interrupt is shown in Figure 18, where an interrupt request is generated by pressing one of the keys consisted as an active-low key matrix which inputs to ports P20–P24.

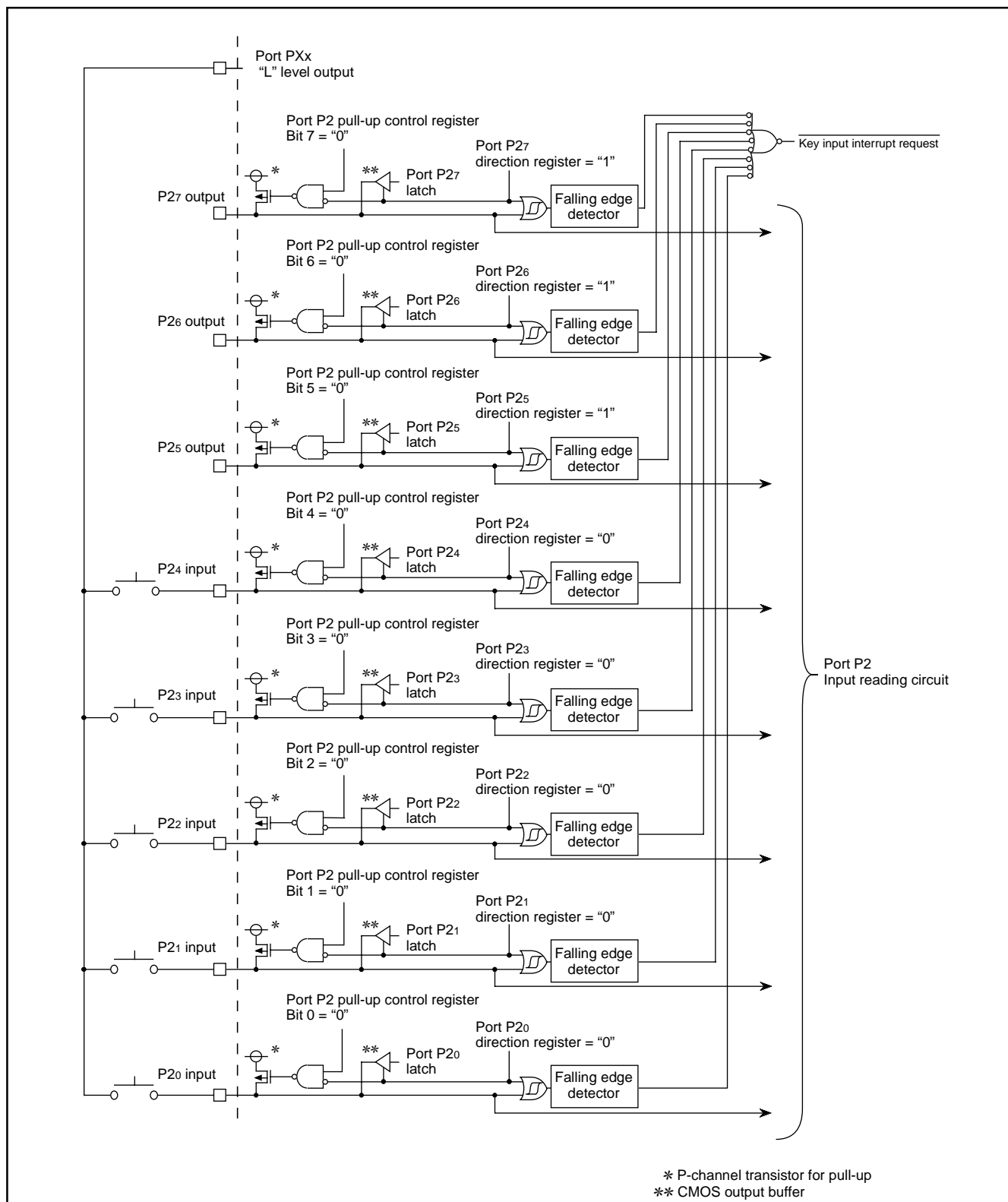


Fig. 18 Connection example when using key input interrupt and port P2 block diagram

### TIMERS

The 7643 group has three 8-bit timers: timer 1, timer 2, and timer 3. All timers are down count timers. When the timer reaches "0016", an underflow occurs at the next count pulse and the corresponding timer latch is reloaded into the timer and the count is continued. When a timer underflows, the interrupt request bit corresponding to that timer is set to "1".

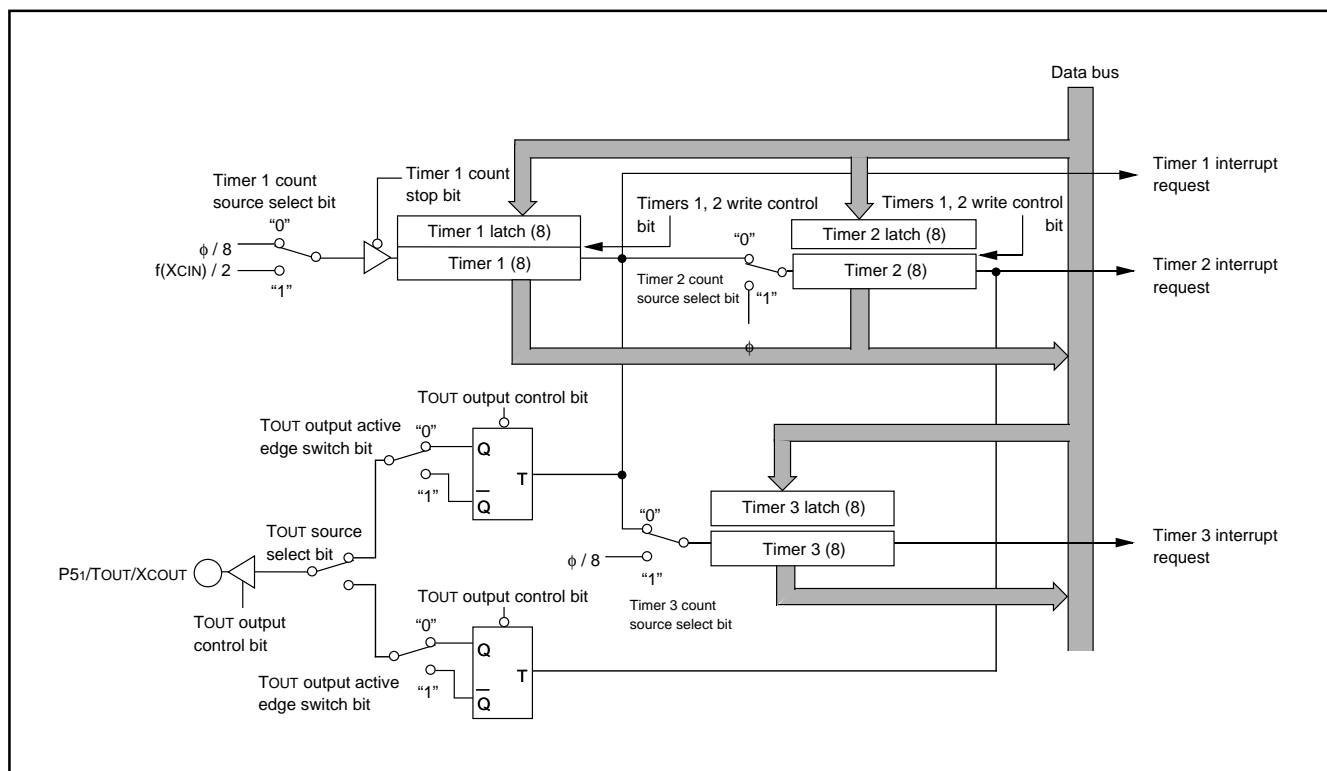


Fig. 19 Timer block diagram

### Timer 1, Timer 2, Timer 3

Timer 1, timer 2, and timer 3 are 8-bit timers. The count source for each timer can be selected by timer 123 mode register.

#### ● Timers 1, 2 Write Control

When the Timers 1, 2 Write Control Bit is "1" and the values are written in the address of timers 1 and 2, the values are loaded only in their latches. The values in the latches are loaded in timers 1 and 2 after timers 1 and 2 underflow.

When the Timers 1, 2 Write Control Bit is "0" and the values are written in the address of timers 1 and 2, the values are loaded in the timers 1 and 2 and their latches at the same time.

#### ● Timers 1, 2 Output Control

A signal of which polarity is inverted each time the timer selected by the TOUT Factor Select Bit underflows is output from the TOUT pin. This is enabled by setting the TOUT Output Control Bit to "1".

When the TOUT Output Active Edge Switch Bit is "0", the TOUT pin starts pulses output beginning at "H"; when this bit is "1", the TOUT pin starts pulses output beginning at "L".

When using a timer in this mode, set the port P51 direction register to output mode.

### ■ Notes

#### ● Timer 1 to Timer 3

Switching of the count sources of timers 1 to 3 does not affect the values of reload latches. However, that may make count operation started. Therefore, write values again in the order of timers 1, 2 and then timer 3 after their count sources have been switched.

#### ● Timers 1, 2 Write Control

When the value is to be written in latch only, unexpected value may be set in the timer if the writing in the latch and the timer underflow are performed at the same timing.

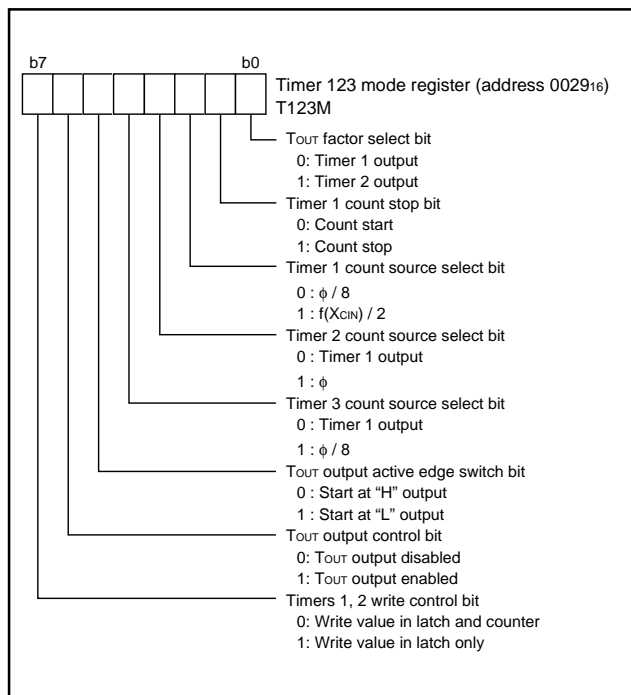


Fig. 20 Structure of timer 123 mode register

## SERIAL INTERFACE

### Serial I/O

The serial I/O can be used only for clock synchronous serial I/O. The transmitter and the receiver must use the same clock. If the internal synchronous clock is used, transfer is started by a write signal to the serial I/O shift register.

#### [Serial I/O Control Register 1 (SIOCON1)] 002B<sub>16</sub>

#### [Serial I/O Control Register 2 (SIOCON2)] 002C<sub>16</sub>

Each of the serial I/O control registers 1 and 2 contains eight bits which control various serial I/O functions.

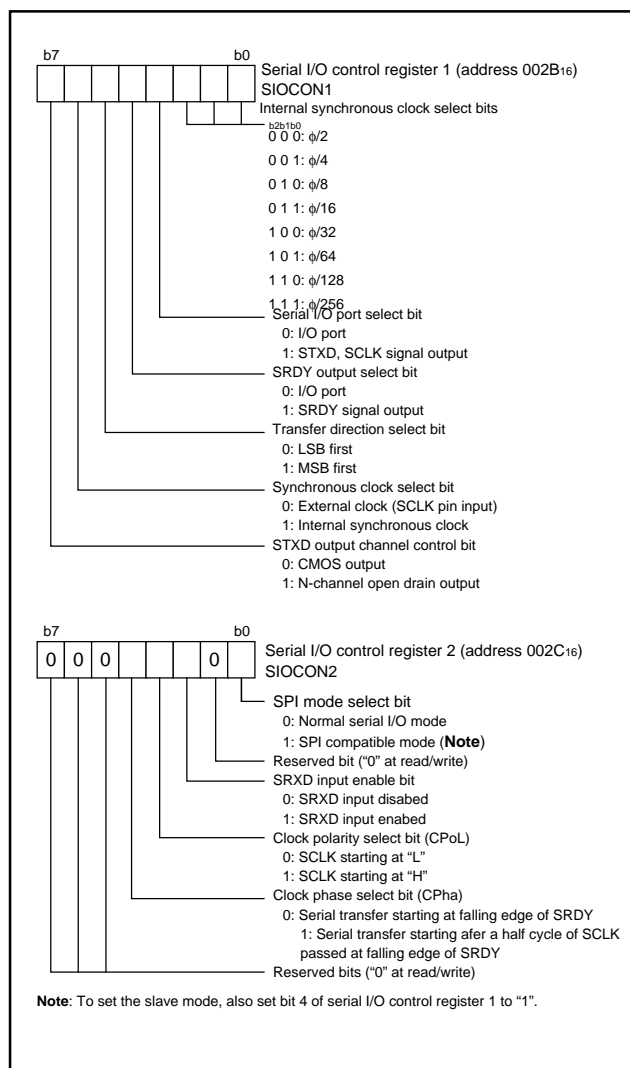


Fig. 21 Structure of serial I/O control registers 1, 2

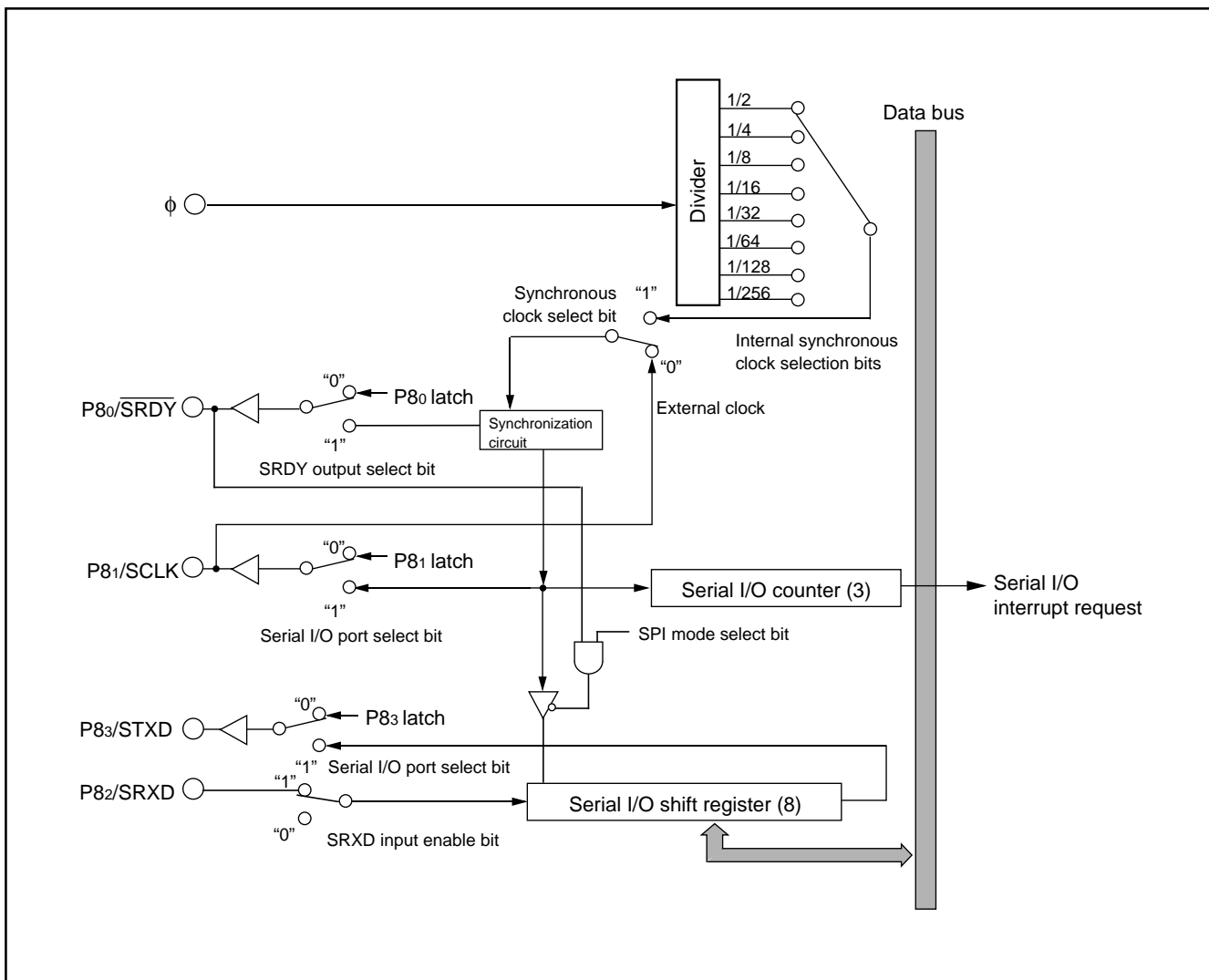


Fig. 22 Block diagram of serial I/O

● Serial I/O Normal Operation

The serial I/O counter is set to "7" by writing operation to the serial I/O shift register (address 002A16). When the SRDY Output Select bit is "1", the SRDY pin goes "L" after that writing. On the negative edge of the transfer clock the SRDY pin returns "H" and the data of the first bit is transmitted from the STXD pin. The remaining data are done from the STXD pin bit by bit on each falling edge of the transfer clock.

Additionally, the data is latched from the SRXD pin on each rising edge of the transfer clock and then the contents of the serial I/O shift register are shifted by one bit.

When the internal synchronous clock is selected as the transfer clock, the followings occur at counting eight transfer clocks:

- The serial I/O counter reaches "0"
- The transfer clock halts at "H"
- The serial I/O interrupt request bit is set to "1"
- The STXD pin goes a high-impedance state after an 8-bit transfer is completed.

When the external clock (SCLK pin input) is selected as the transfer clock, the followings occur at counting eight transfer clocks:

- The serial I/O counter reaches "0"
- The serial I/O interrupt request bit is set to "1"

In this case, the transfer clock needs to be controlled by the external source because the transfer clock does not halt. Additionally, the STXD pin does not go a high-impedance state after an 8-bit transfer is completed.

Figure 23 shows serial I/O timing.

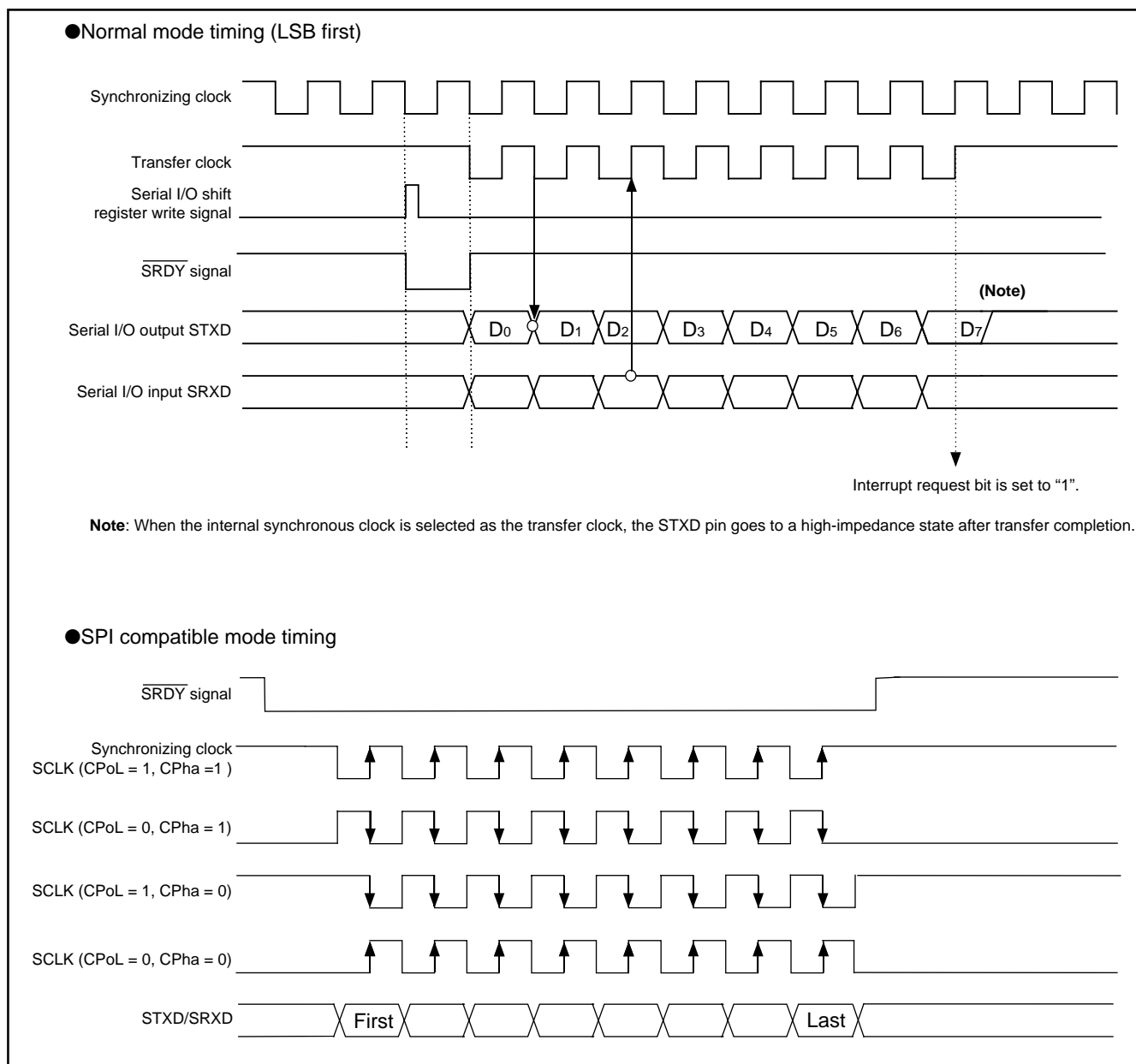


Fig. 23 Serial I/O timing

**● SPI Compatible Mode Operation**

Setting the SPI Mode Select Bit (bit 0 of SIOCON2) puts the serial I/O in SPI compatible mode. The Synchronous Clock Select Bit (bit 6 of SIOCON1) determines whether the serial I/O is an SPI master or slave. When the external clock (SCLK pin input) is selected ("0"), the serial I/O is in slave mode; When the internal synchronous clock is selected ("1"), the serial I/O is in master mode.

In SPI compatible mode the SRXD pin functions as a MISO (Master In/Slave Out) pin and the STXD pin functions as a MOSI (Master Out/Slave In) pin.

In slave mode the transmit data is output from the MISO pin and the receive data is input from the MISO pin. The  $\overline{\text{SRDY}}$  pin functions as the chip-select signal input pin from an external.

In master mode the transmit data is output from the MOSI pin and the receive data is input from the MISO pin. The  $\overline{\text{SRDY}}$  pin functions as the chip-select signal output pin to an external.

**● Slave Mode Operation**

In slave mode of SPI compatible mode 4 types of clock polarity and clock phase can be usable by bits 3 and 4 of serial I/O control register 2.

If the  $\overline{\text{SRDY}}$  pin is held "H", the shift clock is inhibited, the serial I/O counter is set to "7". If the  $\overline{\text{SRDY}}$  pin is held "L", then the shift clock will start.

Make sure during transfer to maintain the  $\overline{\text{SRDY}}$  input at "L" and not to write data to the serial I/O counter.

Figure 23 shows the serial I/O timing.

## UART

Twelve serial data transfer formats can be selected, and the transfer formats used by a transmitter and receiver must be identical.

The transmit and receive shift registers each have a buffer, but the two buffers have the same address in a memory. Since the shift register cannot be written to or read from directly, transmit data is written to the transmit buffer register, and receive data is read from the receive buffer register.

The transmit buffer register can also hold the next data to be transmitted, and the receive buffer register can hold a character while the next character is being received.

The transfer speed (baud rate) is expression as follows:

$$\text{Transfer speed (baud rate)} = f_i / \{(n + 1) \times 16\}$$

n: The contents of UART baud rate generator

f<sub>i</sub>: Using UART clock prescaling select bits, select any one of  $\phi$ ,  $\phi/8$ ,  $\phi/32$ ,  $\phi/256$

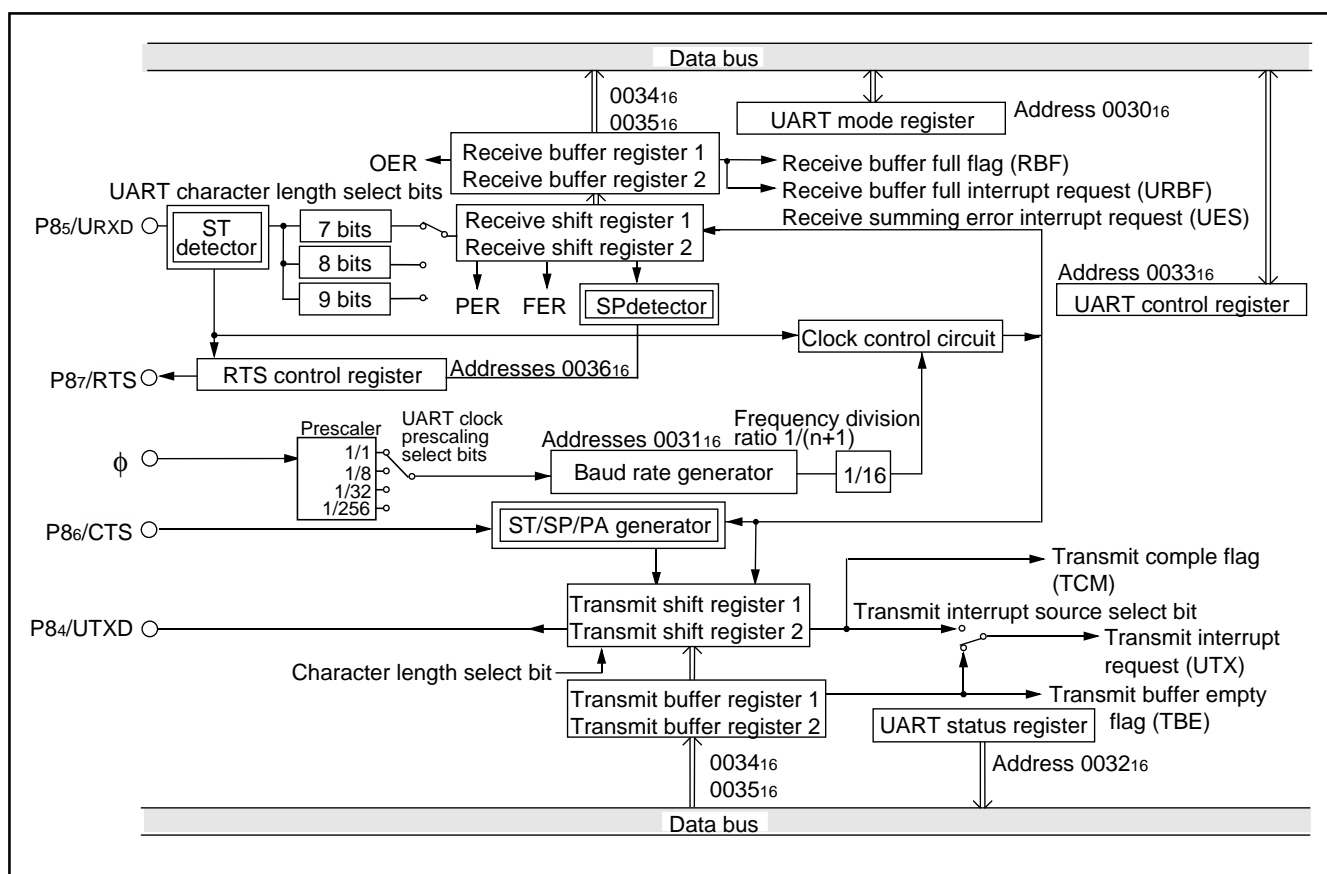


Fig. 24 UART block diagram



### ● UART Transmit Operation

Transmission starts when the Transmit Enable Bit is "1" and the Transmit Buffer Empty Flag is "0". Additionally, when CTS function enabled, the CTS pin must be "L" to be started. The data in which Start Bit and Stop Bit or Parity Bit are also added is transmitted from the low-order byte sequentially. When using 9-bit character length, set the data into the UART transmit buffer register 2 (high-order byte) first before the UART transmit buffer register 1 (low-order byte).

Once the transmission starts, the Transmit Enable Bit, the Transmit Buffer Empty Flag and the CTS pin state (when this is enabled) could not be checked until the transmission in progress has ended.

Transmission requires the following setup:

- (1) Define a baud rate by setting a value n (n = 0 to 255) into UART baud rate generator (addresses 003116).
- (2) Set the Transmit Initialization Bit (bit 2 of UCON) to "1". This will set the UART status register to "0316".
- (3) Select the interrupt source with the Transmit Interrupt Source Select Bit (bit 4 of UCON).
- (4) Configure the data format and clock selection by setting the UART mode register.
- (5) Set the CTS Function Enable Bit (bit 5 of UCON) if CTS function will be used.
- (6) Set the Transmit Enable Bit (bit 0 of UCON) to "1".

If updating a value of UART baud rate generator while the data is being transmitted, be sure to disable the transmission before updating. If the former data remains in the UART transmit buffer registers 1 and 2 at retransmission, an undefined data might be output.

### ● UART Receive Operation

Reception is enabled when the Receive Enable Bit is "1". Detection of the start bit makes transfer clocks generated and the data reception starts in the LSB first.

When using 9-bit character length, read the received data from the UART receive buffer register 2 (high-order byte) first before the UART receive buffer register 1 (low-order byte).

Reception requires the following setup:

- (1) Define a baud rate by setting a value n (n = 0 to 255) into UART baud rate generator (addresses 003116).
- (2) Set the Receive Initialization Bit (bit 3 of UCON) to "1".
- (3) Configure the data format and clock selection by setting the UART mode register.
- (4) Set the RTS Function Enable Bit (bit 5 of UCON) if RTS function will be used.
- (5) Set the Receive Enable Bit (bit 1 of UCON) to "1".

### ● CTS (Clear-to-Send) Function

As a transmitter, the UART can be configured to recognize the Clear-to-Send (CTS) input as a handshaking signal. This is enabled by setting the CTS Function Enable Bit (bit 5 of UCON) to "1". If CTS function is enabled, even when transmission is enabled and the UART transmit buffer register is filled with the data, the transmission never starts; but it will start when inputting "L" to the CTS pin.

Figures 25 and 26 show the UART transmit timings.

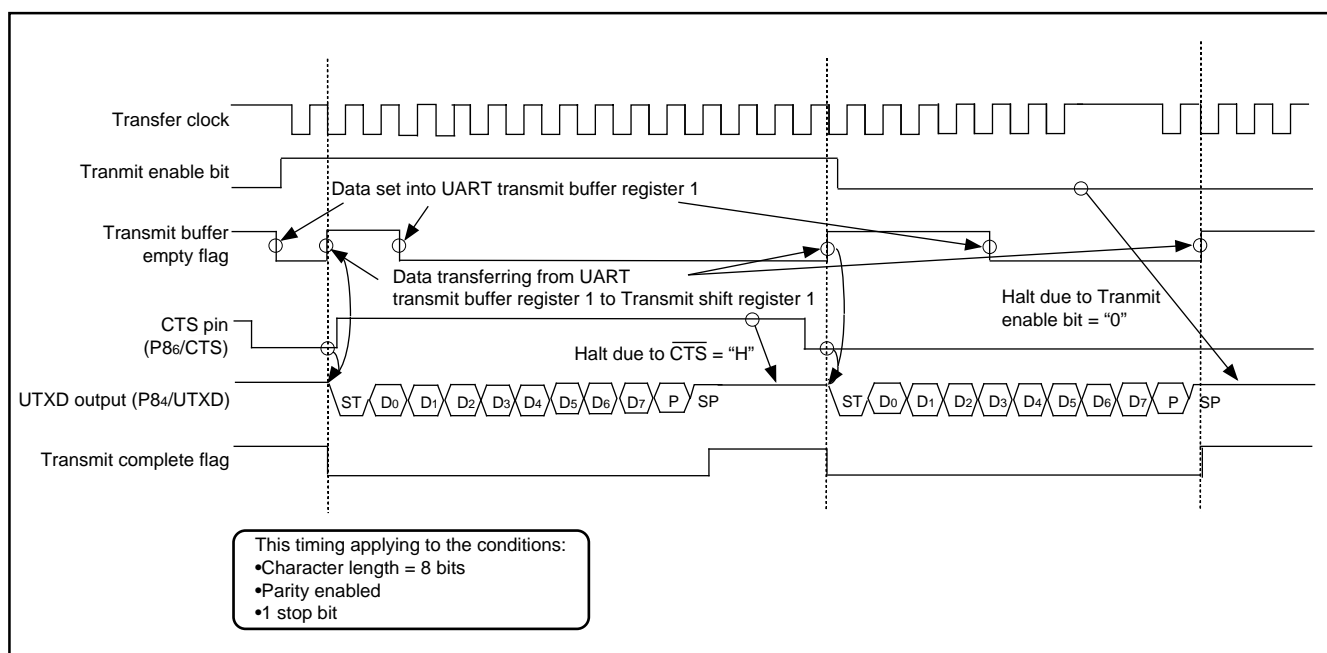


Fig. 25 UART transmit timing (CTS function enabled)

● **RTS (Request-to-Send) Function**

As a receiver, the UART can be configured to generate the Request-to-Send (RTS) handshaking signal. This is enabled by setting the RTS Function Enable Bit (bit 6 of UCON) to "1".

When reception is enabled, that is the Receive Enable Bit is "1", the RTS pin goes "L" to inform a transmitter that reception is possible. The RTS pin goes "H" at reception starting and does "L" at receiving of the last bit.

The delay time from the reception of the last stop bit to the assertion of RTS is selectable using the RTS Assertion Delay Count Select Bits.

When the Receive Enable Bit is set to "0" or the Receive initialization bit is set to "1", the RTS pin goes "H". Even when the Receive Enable Bit is set to "1", the RTS pin goes "H" if detecting an invalid start bit.

Figure 27 shows the UART receive timing.

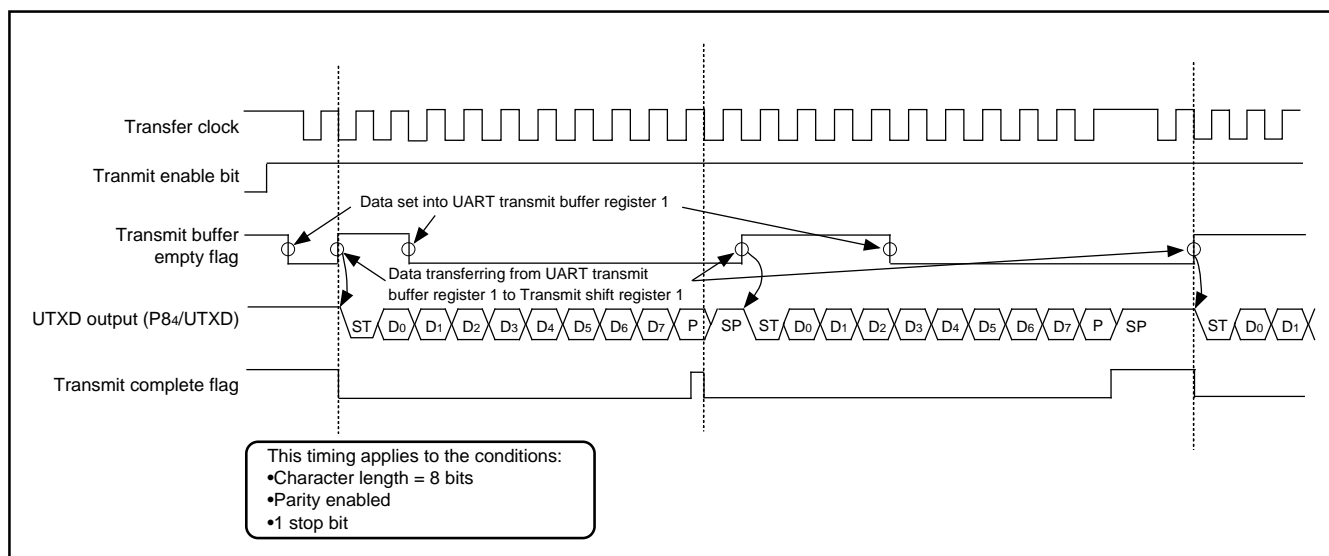


Fig. 26 UART transmit timing (CTS function disabled)

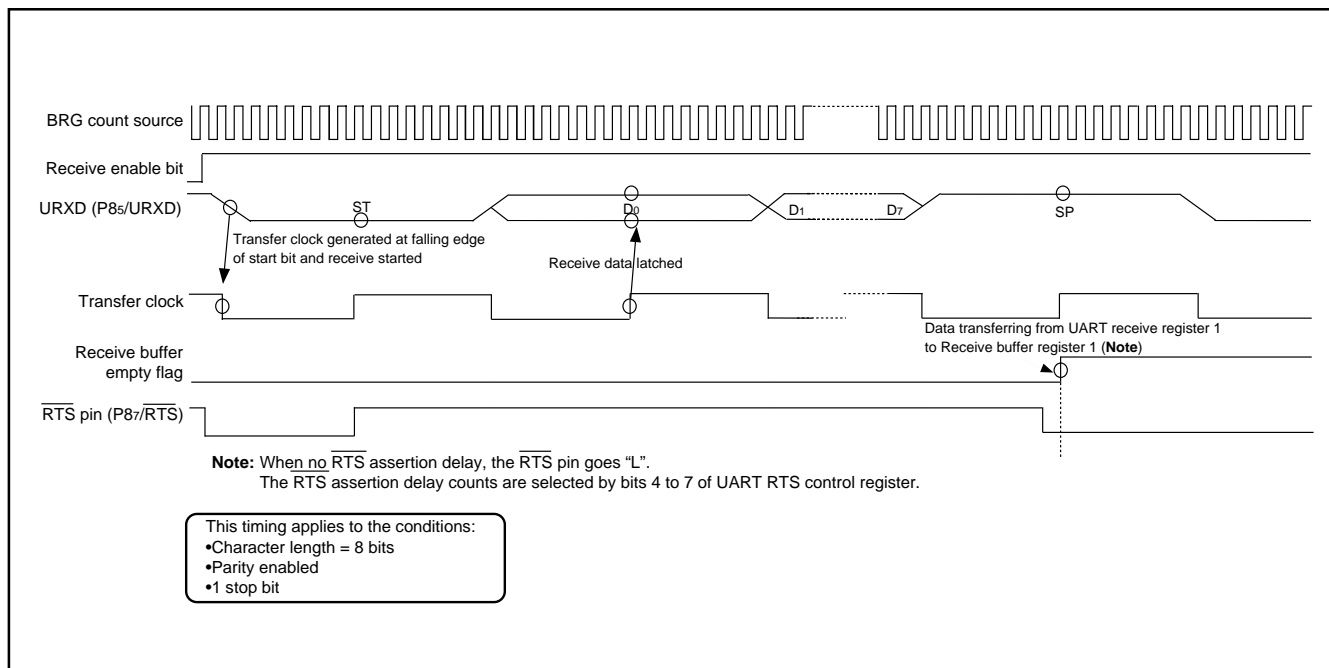


Fig. 27 UART receiving timing (RTS function enabled)

### ● UART Address Mode

The UART address mode is intended for use to communicate between the specified MCUs in a multi-MCU environment. The UART address mode can be used in either an 8-bit or 9-bit character length. An address is identified by the MSB of the incoming data being "1". The bit is "0" for non-address data.

When the MSB of the incoming data is "0" in the UART address mode, the Receive Buffer Full Flag is set to "1", but the Receive Buffer Full Interrupt Request Bit is not set to "1". When the MSB of the incoming data is "1", normal receive operation is performed. In the UART address mode an overrun error is not detected for reception of the 2nd and onward bytes. An occurrence of framing error or parity error sets the Summing Error Interrupt Request Bit to "1" and the data is not received independent of its MSB contents.

Usage of UART address mode is explained as follows:

- (1) Set the UART Address Mode Enable Bit to "1".
- (2) Sends the address data of a slave MCU first from a host MCU to all slave MCUs. The MSB of address data must be "1" and the remaining 7 bits specify the address.
- (3) The all slave MCUs automatically check for the received data whether its stop bit is valid or not, and whether the parity error occurs or not (when the parity enabled). If these errors occur, the Framing Error Flag or Parity Error Flag and the Summing Error Flag are set to "1". Then, the Summing Error Interrupt Request Bit is also set to "1".
- (4) When received data has no error, the all slave MCUs must judge whether the address of the received address data matches with their own addresses by a program. After the MSB being "1" is received, the UART Address Mode Enable Bit is automatically set to "0" (disabled).
- (5) The UART Address Mode Enable Bit of the slave MCUs which have been judged that the address does not match with them must be set to "1" (enabled) again by a program to disable reception of the following data.
- (6) Transmit the data of which MSB is "0" from the host MCU. The slave MCUs disabling the UART address mode receive the data, and their Receive Buffer Full Flags and the Receive Buffer Full Interrupt Request Bits are set to "1". For the other slave MCUs enabling the UART address mode, their Receive Buffer Full Flag are set to "1", but their Receive Buffer Full Interrupt Request Bits are not set to "1".
- (7) An overrun error cannot be detected after the first data has been received in UART Address Mode. Accordingly, even if the slave MCUs does not read the received data and the next data has been received, an overrun error does not occur.

Thus, a communication between a host MCU and the specified MCU can be realized.

### [UART Mode Register (UMOD)] 003016

The UART mode register consists of 8 bits which set a transfer data format and an used clock.

### [UART Baud Rate Generator (UBRG)] 003116

The UART baud rate generator determines the baud rate for transfer.

The baud rate generator divides the frequency of the count source by  $1/(n + 1)$ , where n is the value written to the baud rate generator.

The reset cannot affect the contents of baud rate generator.

### [UART Status Register (USTS)] 003216

The read-only UART status register consists of seven flags (bits 0 to 6) which indicate the UART operating status and various errors. When the UART address mode is enabled, the setting and clearing conditions of each flag differ from the following explanations. These differences are explained in section "UART Address Mode".

#### •Transmit complete flag (TCM)

In the case where no data is contained in the transmit buffer register, the Transmit Complete Flag (TCM) is set to "1" when the last bit in the transmit shift register is transmitted.

The TCM flag is also set to "1" at reset or initialization by setting the Transmit Initialization Bit (bit 2 of UCON). It is set to "0" when transmission starts, and it is kept during the transmission.

#### •Transmit buffer empty flag (TBE)

The Transmit Buffer Empty Flag (TBE) is set to "1" when the contents of the transmit buffer register are loaded into the transmit shift register. The TBE flag is also set "1" at the hardware reset or initialization by setting the Transmit Initialization Bit. It is set to "0" when a write operation is performed to the low-order byte of the transmit buffer register.

#### •Receive buffer full flag (RBF)

The Receive Buffer Full Flag (RBF) is set to "1" when the last stop bit of the data is received. The RBF flag is set to "0" when the low-order byte of the receive buffer register is read, at the hardware reset or initialization by setting the Transmit Initialization Bit.

### ●Receive Errors

If there is an error, it is detected at the same time that data is transferred from the receive shift register to the receive buffer register, and the Receive Buffer Full Flag is set to "1". The all error flags PER, FER, OER and SER are cleared to "0" when the UART status register is read, at the hardware reset or initialization by setting the Transmit Initialization Bit.

The Summing Error Flag (SER) is set to "1" when any one of the PER, FER and OER is set to "1".

The Parity Error Flag (PER) is set to "1" when the sum total of 1s of received data and the parity does not correspond with the selection with the Parity Select Bit (PMD). It is enabled only if the Parity Enable Bit (bit 5 of UMOD) is set to "1".

The Framing Error Flag (FER) is set to "1" when the number of stop bit of the received data does not correspond with the selection with the Stop Bit Length Select Bit (STB).

The Overrun Flag Flag (OER) is set to "1" if the previous data in the low-order byte of the receive buffer register 1 (addresses 0034<sub>16</sub>) is not read before the current receive operation is completed. It is also set "1" if any one of error flags is "1" for the previous data and the current receive operation is completed. Be sure to read UART status register to clear the error flags before the next reception has been completed.

#### **[UART Control Register (UCON)] 0033<sub>16</sub>**

The UART control register consists of eight control bits for the UART function. This register can enable the CTS, RTS and UART address mode.

If the Transmit Enable Bit (TEN) is set to "0" (disabled) while a data is being transmitted, the transmitting operation will stop after the data has been transmitted. If the Receive Enable Bit (REN) is set to "0" (disabled) while a data is being received, the receiving operation will stop after the data has been received.

When setting the Transmit Initialization Bit (TIN) to "1", the TEN bit is set to "0" and the UART status register will be set to "03<sub>16</sub>" after the data has been transmitted. To retransmit, set the TEN to "1" and set a data to the transmit buffer register again. The TIN bit will be cleared to "0" one cycle later after the TIN bit has been set to "1".

Setting the Receive Initialization Bit (RIN) to "1" sets all of the REN, RBF and the receive error flags (PER, FER, OER, SER) to "0". The RIN bit will be cleared to "0" one cycle later after the RIN bit has been set to "1".

When CTS or RTS function is disabled, pins CTS and RTS can be used as ordinary I/O ports, correspondingly.

#### **[UART Transmit/Receive Buffer Registers 1, 2 (UTRB1/UTRB2)] 0034<sub>16</sub>, 0035<sub>16</sub>**

The transmit buffer register and the receive buffer register are located at the same address. The transmit buffer register is write-only and the receive buffer register is read-only. If a character bit length is 7 bits, the MSB of received data is invalid. If a character bit length is 7 or 8 bits, the received contents of UTRB2 are also invalid. If a character bit length is 9 bits, the received high-order 7 bits of UTRB2 are "0".

#### **[UART RTS Control Register (URTS)] 0036<sub>16</sub>**

The delay time from the reception of the last stop bit to the assertion of RTS is selectable using the RTS Assertion Delay Count Select Bits. If the stop bit is detected before RTS assertion delay time has expired, the RTS pin is kept "H". The RTS assertion delay count starts after the last data reception is completed.

Setting the RIN bit to "1" resets the URTS. After setting the RIN bit to "1", set this URTS.

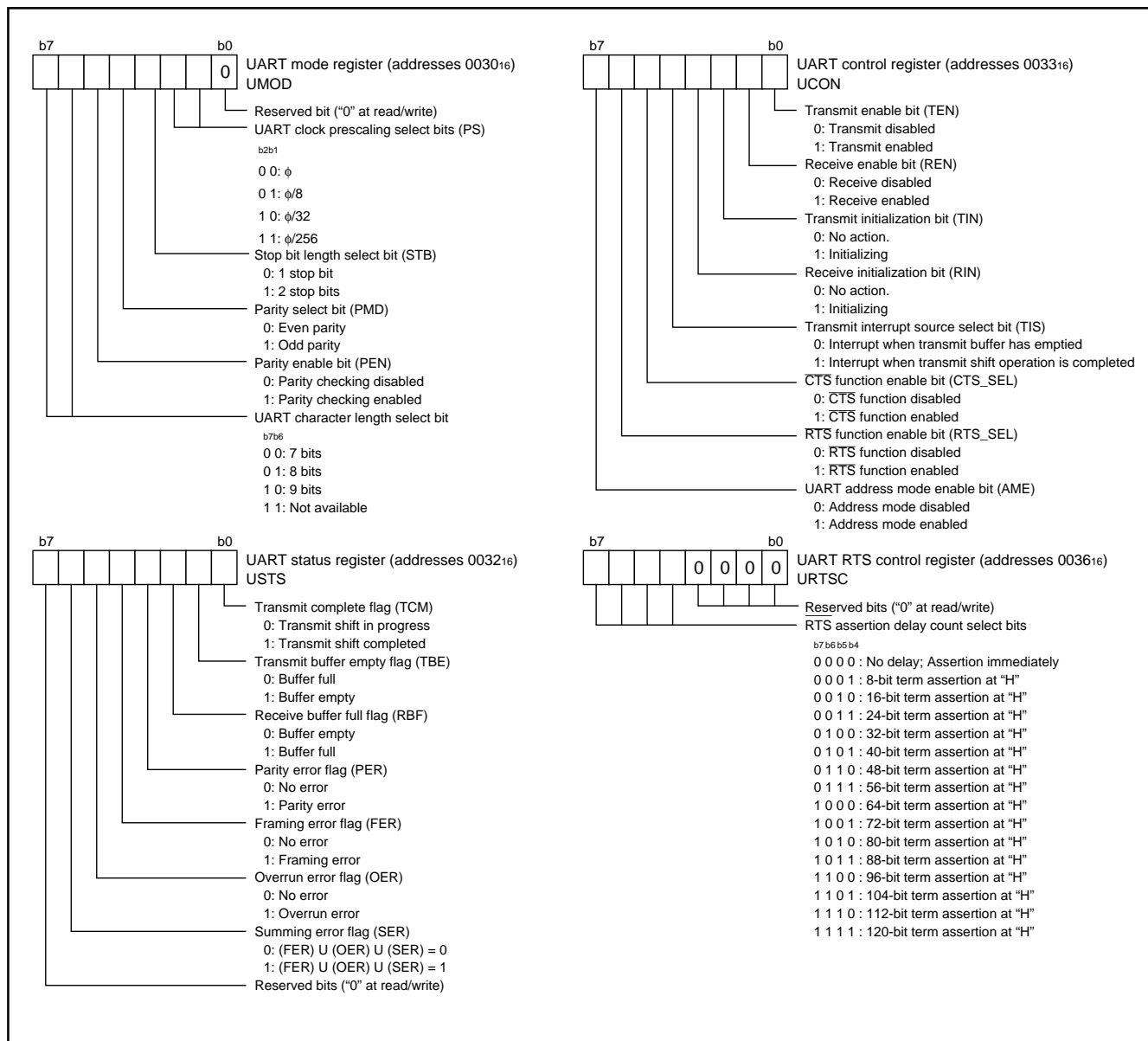


Fig. 28 Structure of UART related registers

**DMAC**

The 7643 group is equipped with 2 channels of DMAC (direct memory access controller) which enable high speed data transfer from a memory to a memory without use of the CPU.

The DMAC initiates the data transfer with an interrupt factor specified by the DMAC channel x (x = 0, 1) hardware transfer request source bit (DxHR), or with a software trigger.

The DxTMS [DMA Channel x (x = 0, 1) Transfer Mode Selection Bit] selects one of two transfer modes; cycle steal mode or burst transfer mode. In the cycle steal mode, the DMAC transfers one byte of data for each request. In the burst transfer mode, the DMAC transfers the number of bytes data specified by the transfer count register for each request. The count register is a 16-bit counter; the maximum number of data is 65,536 bytes per one request.

Figure 29 shows the DMA control block diagram and Figure 30 shows the structure of DMAC related registers.

**[DMAC Index and Status Register] DMAIS**

The DMAC Index and Status Register consists of various control bits for the DMAC and its status flags.

The DMA Channel Index Bit (DCI) selects which channel (0 or 1) will be accessed, since the mode registers, source registers, destination registers and transfer count register of both DMAC channels share the same SFR addresses, respectively.

**[DMAC Channel x (x = 0, 1) Mode Registers 1, 2] DMAxM1, DMAxM2**

The 16 bits of DMAC Channel x Mode Registers 1 and 2 control each operation of DMAC channels 0 and 1.

When the DMAC Channel x (x = 0, 1) Write Bit (DxDWC) is "0", data is simultaneously written into each latch and register of the Source Registers, Destination Register, and Transfer Count Registers. When this bit is "1", data is written only into their latches.

When data is read from each register, it must be read from the higher bytes first, then the lower bytes. When writing data, write to the lower bytes first, then the higher bytes.

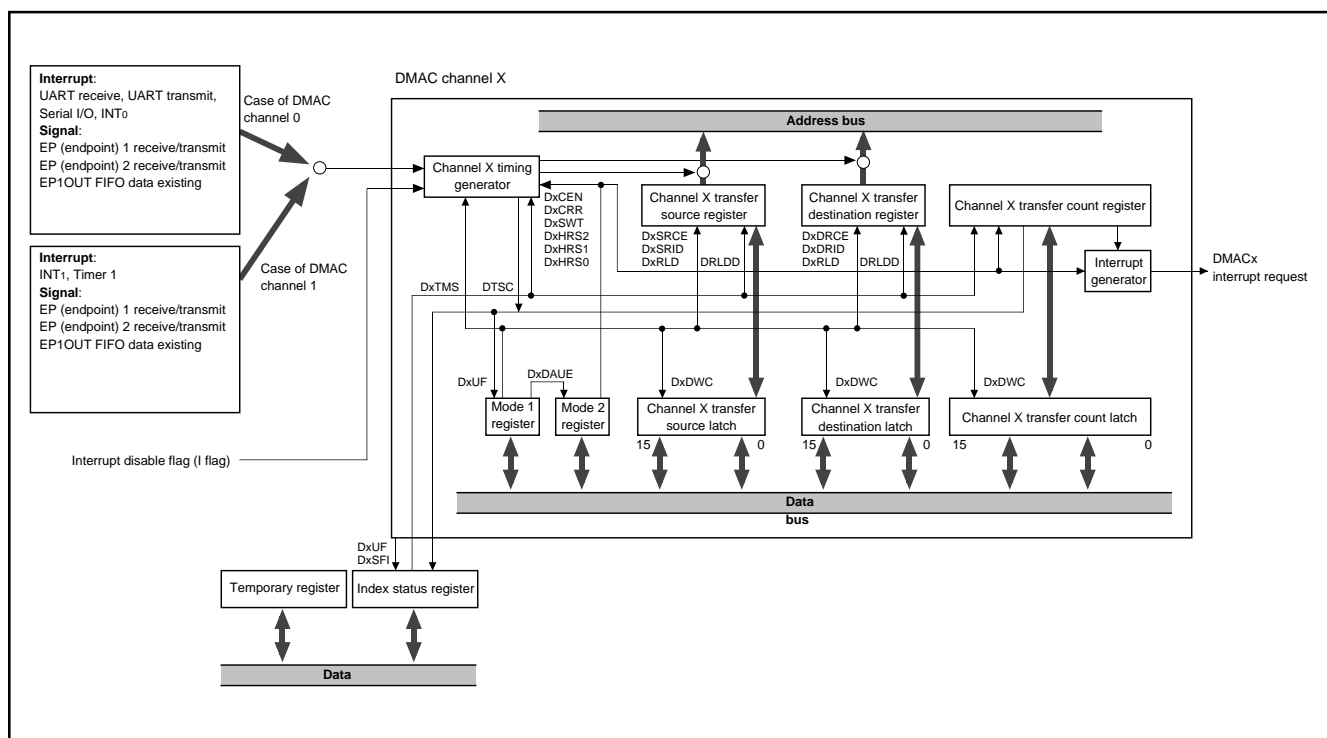


Fig. 29 DMACx (x = 0, 1) block diagram

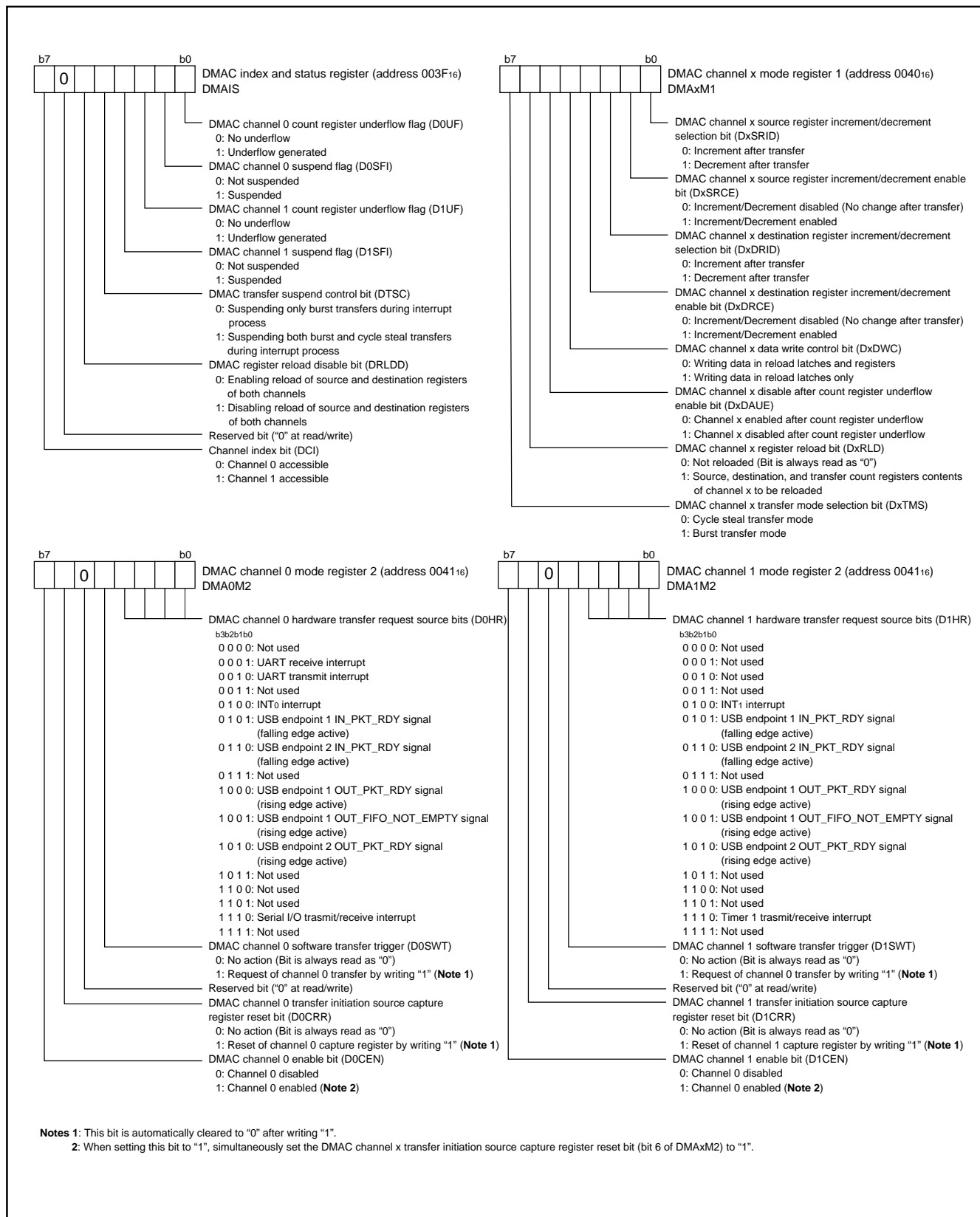


Fig. 30 Structure of DMACx related register

### (1) Cycle Steal Transfer Mode

When the DMAC Channel  $x$  ( $x = 0, 1$ ) Transfer Mode Selection Bit (DxTMS) is set to "0", the respective DMAC Channel  $x$  operates in the cycle steal transfer mode.

When a request of the specified transfer factor is generated, the selected channel transfers one byte of data from the address indicated by the Source Register into the address indicated by the Destination Register.

There are two kinds of DMA transfer triggers supported: hardware transfer factor and software trigger. Hardware transfer factors can be selected by the DMAC $x$  ( $x = 0, 1$ ) Hardware Transfer Request source Bit (DxHR). To only use the Interrupt Request Bit, the interrupt can be disabled by setting its Interrupt Enable Bit of Interrupt Control Register to "0".

The DMA transfer request as a software trigger can be generated by setting the DMA Channel  $x$  ( $x = 0, 1$ ) Software Transfer Trigger Bit (DxSWT) to "1".

The Source Registers and Transfer Destination Registers can be either decreased or increased by 1 after transfer completion by setting bits 0 to 3 in the DMAC Channel  $x$  ( $x = 0, 1$ ) Mode Register. When the Transfer Count Register underflows, the Source Registers and Destination Registers are reloaded from their latches if the DMAC Register Reload Disable Bit (DRLDD) is "0". The Transfer Count Register value is reloaded after an underflow regardless of DRLDD setting. At the same time, the DMAC Interrupt Request Bit and the DMA Channel  $x$  ( $x = 0, 1$ ) Count Register Underflow Flag are set to "1".

The DMAC Channel  $x$  Disable After Count Register Underflow Enable Bit (DxDAUE) is "1", the DMAC Channel  $x$  Enable Bit (DxCEN) goes to "0" at an underflows of Transfer Count Register. By setting the DMAC Channel  $x$  ( $x = 0, 1$ ) Register Reload Bit (DxRLD) to "1", the Source Registers, Destination Registers, and Transfer Count Registers can be updated to the values in their respective latches.



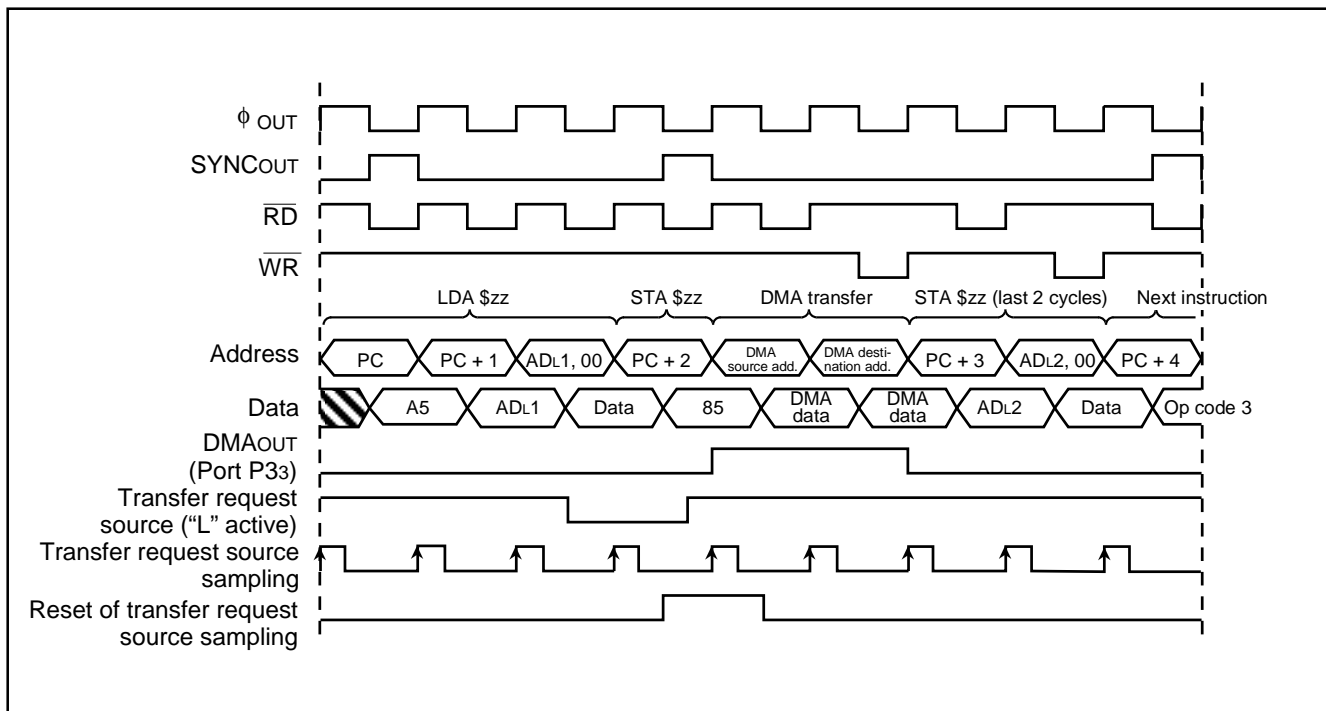


Fig. 31 Timing chart for cycle steal transfer caused by hardware-related transfer request

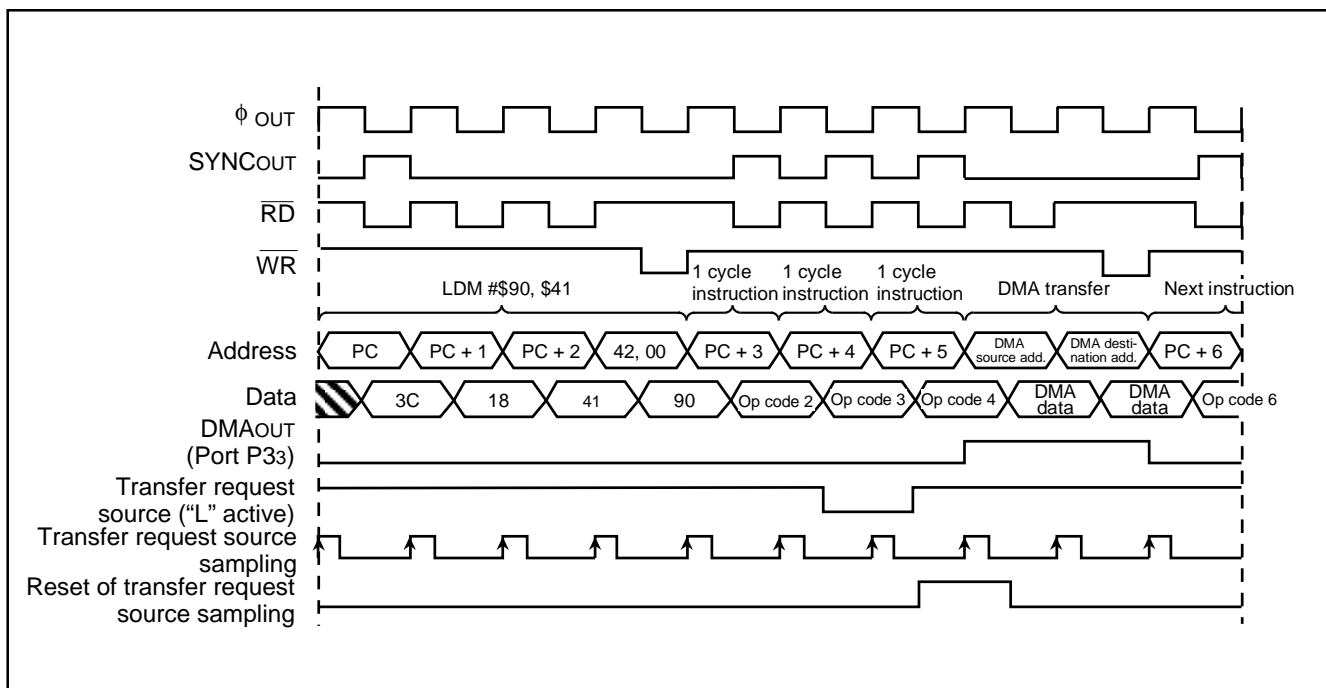


Fig. 32 Timing chart for cycle steal transfer caused by software trigger transfer request

### (2) Burst Transfer Mode

When the DMAC Channel x Transfer Mode Selection Bit (DxTMS) is set to "1", the respective DMAC channel operates in the burst transfer mode.

In the burst transfer mode, the DMAC continually transfers the number of bytes of data specified by the Transfer Count Register for one transfer request. Other than this, the burst transfer mode operation is the same as the cycle steal mode operation.

### Priority

The DMAC places a higher priority on Channel-0 transfer requests than on Channel-1 transfer requests.

If a Channel-0 transfer request occurs during a Channel-1 burst transfer operation, the DMAC completes the next transfer source and destination read/write operation first, and then starts the Channel-0 transfer operation. As soon as the Channel-0 transfer is completed, the DMAC resumes the Channel-1 transfer operation.

When an interrupt request occurs during any DMA operation, the transfer operation is suspended and the interrupt process routine is initiated. During the interrupt operation, the DMAC automatically sets the corresponding DMAC Channel x (x = 0, 1) Suspend Flag (DxSFI) to "1". As soon as the CPU completes the interrupt operation, the DMAC clears the flag to "0" and resumes the original operation from the point where it was suspended.

The suspended transfer due to the interrupt can also be resumed during its interrupt process routine by writing "1" to the DMAC Channel x (x = 0,1) Enable Bit (DxCEN).

The timing charts for a burst transfer caused by a hardware-related transfer request are shown in Figure 33.

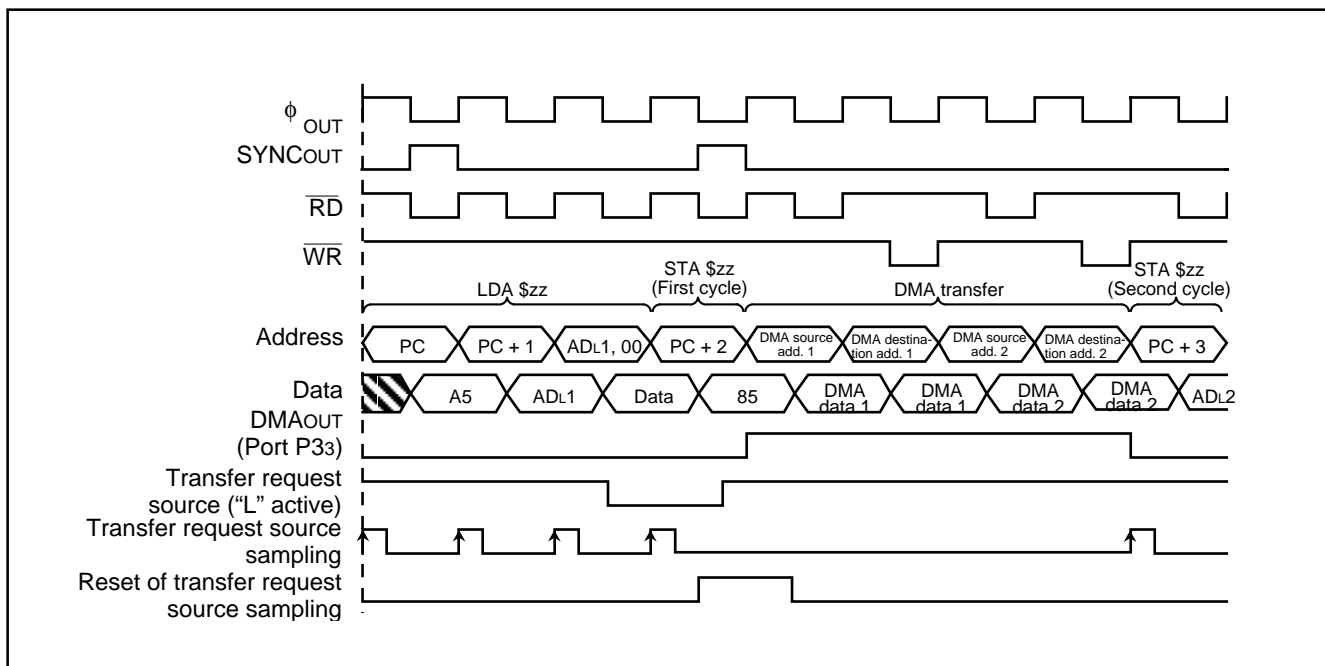


Fig. 33 Timing chart for burst transfer caused by hardware-related transfer request

## USB FUNCTION

The 7643 Group MCU is equipped with a USB Function Control Unit (USB FCU). This USB FCU allows the MCU to communicate with a host PC using a minimum amount of the MCU power. This built-in USB FCU complies with Full-Speed USB2.0 specification that supports four transfer types: Control Transfer, Isochronous Transfer, Interrupt Transfer, and Bulk Transfer. However, the 7643 Group can use three of Control Transfer, Interrupt Transfer and Bulk Transfer. This built-in USB FCU performs the data transfer error detection and transfer retry operation by hardware. The default transfer mode of the USB FCU is bulk transfer mode at reset. The user must set the USB FCU for the required transfer mode by software.

The USB FCU has three endpoints (Endpoint 0 to Endpoint 2). The EPINDEX bit selects one of these five endpoints for the USB FCU to use. Each endpoint has IN (transmit) FIFO and OUT (receive) FIFO. To use the USB FCU, the USB enable bit (USBC7) must be set to "1". The USB Function Interrupt is supported for this MCU.

Figure 34 shows the USB FCU (USB Function Control Unit) block diagram. The USB FCU consists of the SIE (Serial Interface Engine) performing the USB data transfer, GFI (Generic Function Interface) performing USB protocol handling, SIU (Serial Engine Interface Unit) performing a received address and endpoint decoding, MCI (Microcontroller Interface) handling the MCU interface or performing address decoding and synchronization of control signals, and the USB transceiver.

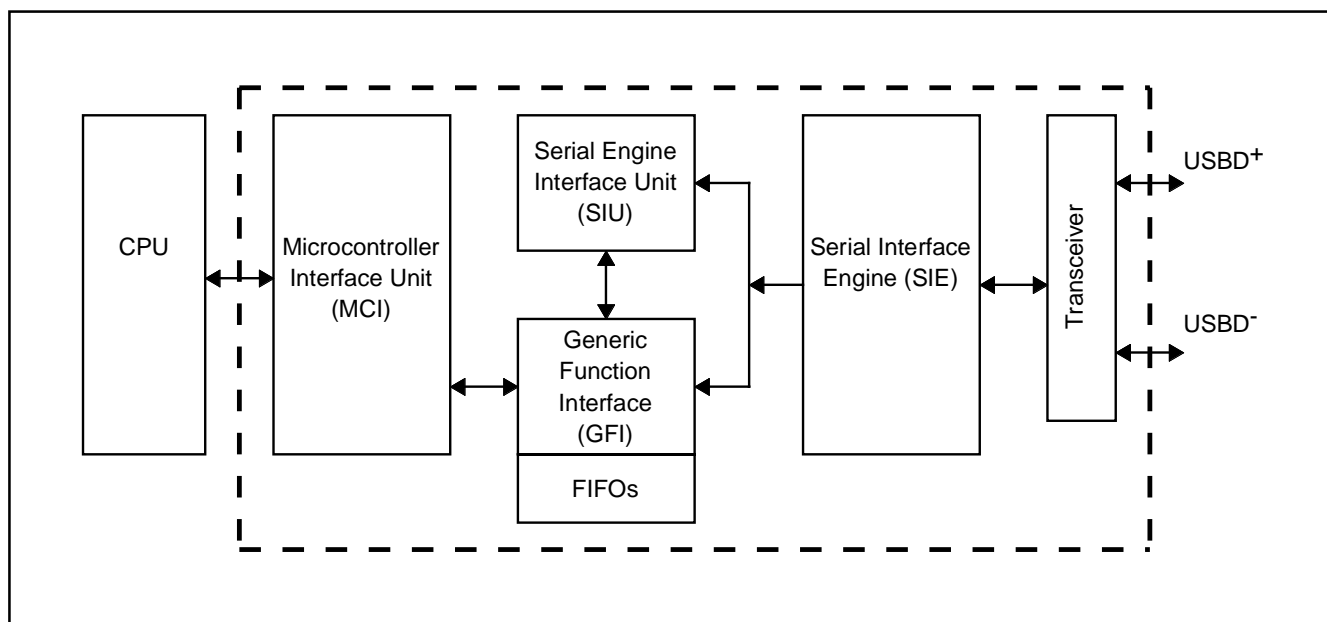


Fig. 34 USB FCU (USB Function Control Unit) block

## USB Transmission

Endpoint 0 to Endpoint 2 have IN (transmit) FIFOs individually. Each endpoint's FIFO is configured in following way:

Endpoint 0: 16-byte  
 Endpoint 1: 128-byte  
 Endpoint 2: Mode 0: 32-byte  
           Mode 1: 128-byte

When Endpoint 2 is used for data transmit, the IN FIFO size can be selected. Endpoint 2 have 2 modes programmable IN-FIFO. Each mode can be selected by the USB endpoint FIFO mode selection register (address 005F16).

When writing data to the USB Endpoint-x FIFO (addresses 006016 to 006216) in the SFR area, the internal write pointer for the IN FIFO is automatically increased by 1. When the AUTO\_SET bit is "1" and if the stored data reaches to the max. packet value set in USB Endpoint x IN max. packet size register (address 005B16), the USB FCU sets the IN\_PKT\_RDY bit to "1". When the AUTO\_SET bit is "0", the IN\_PKT\_RDY bit will not be automatically set to "1"; it must be set to "1" by software. (The AUTO\_SET bit function is not applicable to Endpoint 0.)

The USB FCU transmits the data when it receives the next IN token. The IN\_PKT\_RDY bit automatically goes to "0" when the data transfer is complete.

### ●Interrupt transfer mode

Endpoints 1 to 2 can be used in interrupt transfer mode. During a regular interrupt transfer, an interrupt transaction is similar to the bulk transfer. Therefore, there is no special setting required. When IN-endpoint is used for a rate feedback interrupt transfer, INTPT bit of the IN\_CSR register must be set to "1". The following steps show how to configure the IN-endpoint for the rate feedback interrupt transfer.

1. Set a value which is larger than 1/2 of the USB Endpoint-x FIFO size to the USB Endpoint x IN max. package size register.
2. Set INTPT bit to "1".
3. Flush the old data in the FIFO.
4. Store transmission data to the IN FIFO and set the IN\_PKT\_RDY bit to "1".
5. Repeat steps 3 and 4.

In a real application, the function-side always has transfer data when the function sends an endpoint in a rate feedback interrupt. Accordingly, the USB FCU never returns a NAK against the host IN token for the rate feedback interrupt. The USB FCU always transmits data in the FIFO in response to an IN token, regardless of IN\_PKT\_RDY. However, this premises that there is always an ACK response from Host PC after the 7643 Group has transmitted data to IN token.

When MAXP size  $\leq$  (a half of IN FIFO size), the IN FIFO can store two packets (called double buffer). At this time, the IN FIFO status can be checked by monitoring the IN\_PKT\_RDY bit and the TX\_NOT\_EPT flag. The TX\_NOT\_EPT flag is a read-only flag which shows the FIFO state. When IN\_PKY\_RDY = 0 and TX\_NOT\_EPT = 0, IN FIFO is empty. When IN\_PKY\_RDY = 0 and TX\_NOT\_EPT = 1, IN FIFO has one packet.

In double buffer mode, as long as the IN FIFO is not filled with double packets, IN\_PKT\_RDY will not be set to "1", even if it is set to "1" by software, but TX\_NOT\_EPT flag will be set to "1". In single buffer mode, if MAXP > (a half of IN FIFO), this condition never occurs.

When IN\_PKT\_RDY = "1" and TX\_NOT\_EPT = "1", IN FIFO holds two packets in double buffer mode and one packet in single packet mode. In single packet mode, when the IN\_PKT\_RDY bit is set to "1" by software, the TX\_NOT\_EPT flag is set to "1" as well. During double buffer mode, if you want to load two packets sequentially, you must set the IN\_PKT\_RDY bit to "1" each time a packet is loaded.

## USB Reception

Endpoint 0 to Endpoint 2 have OUT (receive) FIFOs individually.

Each endpoint's FIFO is configured in following way:

Endpoint 0: 16-byte

Endpoint 1: 128-byte

Endpoint 2: Mode 0: 32-byte

Mode 1: 128-byte

When Endpoint 2 is used for data receive, the OUT FIFO size can be selected. Endpoint 2 have 2 modes programmable IN-FIFO. Each mode can be selected by the USB endpoint FIFO mode selection register (address 005F<sub>16</sub>).

Data transmitted from the host-PC is stored in Endpoint x FIFO (0060<sub>16</sub> to 0062<sub>16</sub>). Every time the data is stored in the FIFO, the internal OUT FIFO write pointer is increased by 1. When one complete data packet is stored, the OUT\_PKT\_RDY flag is set to "1" and the number of received data packets is stored in USB Endpoint x OUT write count register. When the AUTO\_CLR bit is "1" and the received data is read out from the OUT FIFO, the OUT\_PKT\_RDY flag is cleared to "0". When the AUTO\_CLR bit is "1", the OUT\_PKT\_RDY flag will not be cleared automatically by the FIFO read; it must be cleared by software. (The AUTO-CLR bit function is not applicable in Endpoint 0.)

When MAXP size  $\leq$  (a half of OUT FIFO size), the OUT\_FIFO can receive 2 packets (double buffer). At this time, the OUT\_FIFO status can be checked by the OUT\_PKT\_RDY flag. When the FIFO holds two packets and one packet is read from the FIFO, the OUT\_PKT\_RDY flag is not cleared even if it is set to "0". (The flag returns from "0" to "1" in one  $\phi$  cycle after the read-out). During double buffer mode, the USB Endpoint x OUT write count register holds the number of previously received packets. This count register is updated after reading out one of packets in the OUT FIFO and clearing the OUT\_PKT\_RDY flag to "0".

## TOGGLE Initialization

In order to initialize the data toggle sequence bit of the endpoint, in other words, resetting the next data packet to DATA0; set the TOGGLE\_INT bit to "1" and then clear back to "0".

## USB Interrupts

The USB FCU has USB Function Interrupt.

### ●USB Function Interrupt (USBF-INT)

The USBF-INT is usable for the USB data flow control and power management. The USBF-INT request occurs at data transmit/receive completion, overrun/underrun, reset, or receiving suspend/resume signal. To enable this interrupt, the USB function interrupt enable bit in the interrupt control register A (address 000516) and the respective bit in the USB interrupt enable registers 1 and 2 (addresses 0005416 and 0005516) must be set to "1". When setting bit 7 in USB interrupt enable register 2 to "1", the suspend interrupt and the resume interrupt are enabled.

Endpoint x (x = 0 to 2) IN interrupt request occurs when the USB Endpoint x IN interrupt status flag (INTST 0, 2, 4) of USB interrupt status registers 1 and 2 (addresses 005216 and 005316) is "1". The USB Endpoint x IN interrupt status flag is set to "1" when the respective endpoint IN\_PKT\_RDY bit is "1".

Endpoint x (x = 0 to 2) OUT interrupt request occurs when the USB endpoint x OUT interrupt status flag (INTST3, 5) in USB interrupt status registers 1 and 2 is set to "1". The USB Endpoint x OUT interrupt status flag is set to "1" when the respective endpoint OUT\_PKT\_RDY flag is "1".

The USB reset interrupt request occurs when the USB reset interrupt status flag (INTST13) in USB interrupt status register 2 is set to "1". This flag is set when the SE0 is detected on the D+/D- line for at least 2.5  $\mu$ s. When this situation happens, all USB internal registers (addresses 005016 to 005F16), except this flag, are initialized to the default state at reset. The USB reset interrupt is always enabled.

The suspend/resume interrupt request occurs when either the USB resume signal interrupt status flag (INTST14) or the USB suspend signal interrupt status flag (INTST15) in USB interrupt status register 2 is set to "1".

The bits in both interrupt status registers 1 and 2 can be cleared by writing "1" to each bit.

## Suspend/Resume Functions

If no bus activity is detected on the D+/D- line for at least 3 ms, the USB suspend signal detect flag (SUSPEND) of the USB power control register (address 005116) and the USB suspend signal interrupt status flag of USB interrupt status register 2 are set to "1" and the suspend interrupt request occurs. The following procedure must be executed after pushing the internal registers (A, X, Y) to memories during the suspend interrupt process routine.

- (1) Clear all bits of USB interrupt status register 1 (address 005216) and USB interrupt status register 2 (address 005316) to "0".

- (2) Set the USB clock enable bit to "0". (After disabling the USB clock, do not write to any of the USB internal registers (addresses 005016 to 006216), except for the USB control register (address 001316), clock control register (address 001F16), and frequency synthesizer control register (address 006C16).)
- (3) Set the frequency synthesizer enable bit to "0".
- (4) Set the USB line driver current control bit to "1". (Always keep the USB line driver current control bit set to "0" during USB function operations. When operating at  $V_{cc} = 3.3$  V, this bit does not need to be set.)
- (5) Keep total drive current at 500  $\mu$ A or less.
- (6) Disable the timer 1 interrupt.
- (7) Disable the timer 2 interrupt. (Disable all the other external interrupts.)
- (8) Set the timer 1 interrupt request bit to "0".
- (9) Set the timer 2 interrupt request bit to "0".
- (10) Set the interrupt disable flag (I) to "0".
- (11) Execute the STP instruction.

At this point, the MCU will be in stop mode (suspend mode). Before executing the STP instruction, make sure to set the USB function interrupt request bit (bit 0 at address 000216) to "0" and the USB function interrupt enable bit (bit 0 at address 000516) to "1".

The USB suspend detect signal flag goes to "0" when the USB resume signal detect flag (RESUME) is set to "1". During suspend mode, if the clock operation is started up with a process (remote wake-up) other than the resume interrupt process (for example; reset or timer), make sure to clear the USB suspend detect signal flag to "0" when you set the USB remote wake-up bit to "1". When the USB FCU is in suspend mode and detects a non-idle signal on the D+/D- line, the USB resume detect flag and the USB resume signal interrupt status flag both go to "1" and a resume interrupt request occurs. At this point, pull the internal registers (A, X, Y) in this interrupt process routine. Take the following procedure in the USB resume interrupt process.

- (1) Set the USB line driver current control bit to "0". (When operating at  $V_{cc} = 3.3$  V, this bit does not need to be set.)
- (2) Set the frequency synthesizer enable bit to "1" and set a 2 ms wait or more .
- (3) Check the frequency synthesizer lock status bit. If "0", it must be checked again after a 0.1 ms wait.
- (4) Set the USB clock enable bit to "1".

Set the USB resume signal interrupt status flag to "0" after the wake-up sequence process. The USB resume detect flag goes to "0" at the same time. When the clock operation is started up with a remote wake-up, set the USB remote wake-up bit to "1" after the wake-up sequence process. (keep it set to "1" for a minimum of 10 ms and maximum of 15 ms). By doing this, the MCU will send a resume signal to the host CPU and let it know that the suspend state has been released.

After that, set the USB remote wake-up bit and the USB suspend detection flag to "0", because the USB suspend detection flag is not automatically cleared to "0" with a remote wake-up.

#### [USB Control Register] USBC

When using the USB function, the USB enable bit must be set to "1". The USB line driver supply bit must be set to "0" (DC-DC converter is disabled) when operating at  $V_{cc} = 3.3V$ . In this condition, the setting of the USB line driver current control bit has no effect on USB operations.

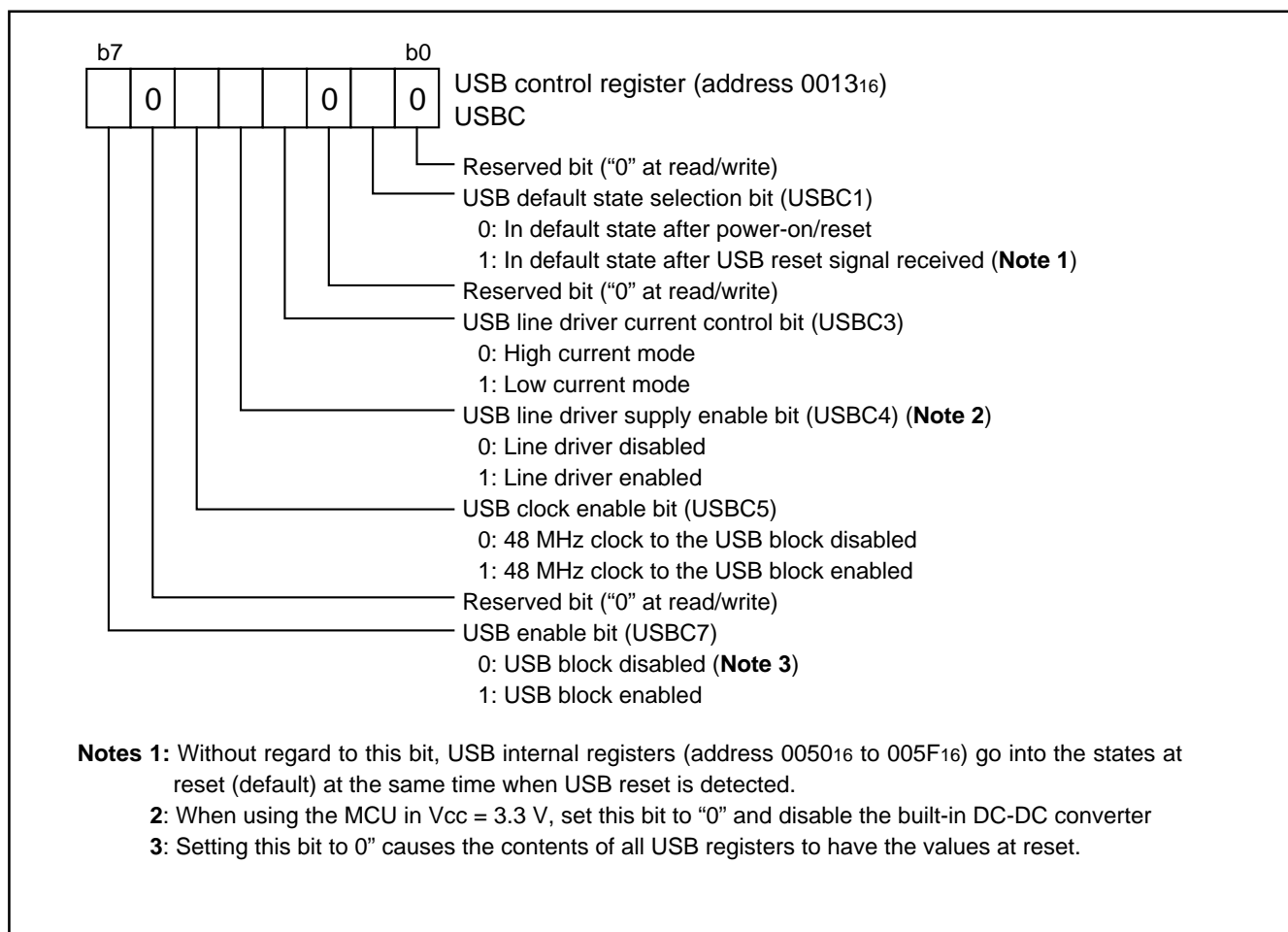


Fig. 35 Structure of USB control register

**[USB Address Register] USBA**

The USB address register maintains the USB function control unit address assigned by the host computer. When receiving the SET\_ADDRESS, keep it in this register. The values of this register are "0" when the device is not yet configured. The values of this register are also set to "0" when the USB block is disabled (bit 7 of USB control register is set to "0"). In addition, no matter what value is written to this register, it will have no effect on the set value.

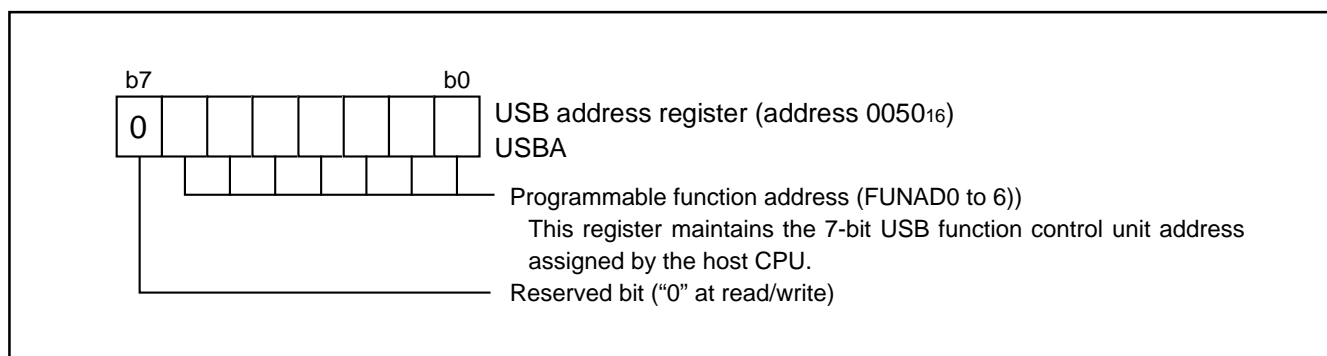


Fig. 36 Structure of USB address register

**[USB Power Management Register] USBPM**

The USB power management register is used for power management in the USB FCU. This register needs to be set only when using the remote wake-up to resume the MCU from suspend mode.

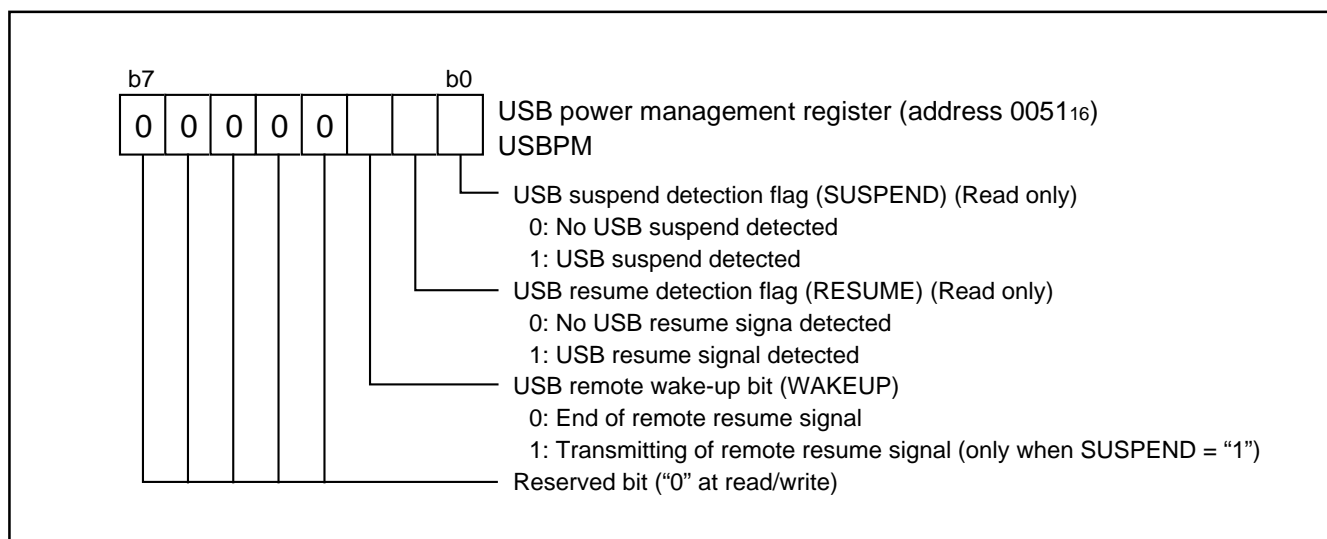


Fig. 37 Structure of USB power management register





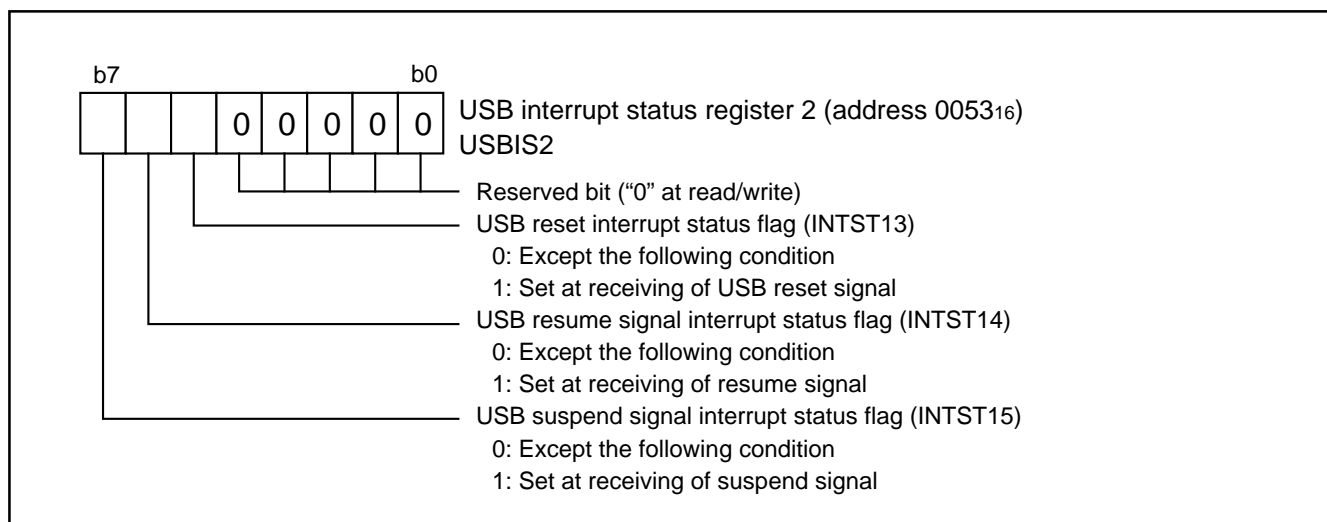


Fig. 39 Structure of USB interrupt status register 2

**[USB Interrupt Enable Registers 1 and 2] USBIE1, USBIE2**  
The USB interrupt enable registers are used to enable the USB

function interrupt. Upon reset, all USB interrupts except the USB suspend and USB resume interrupts are enabled.

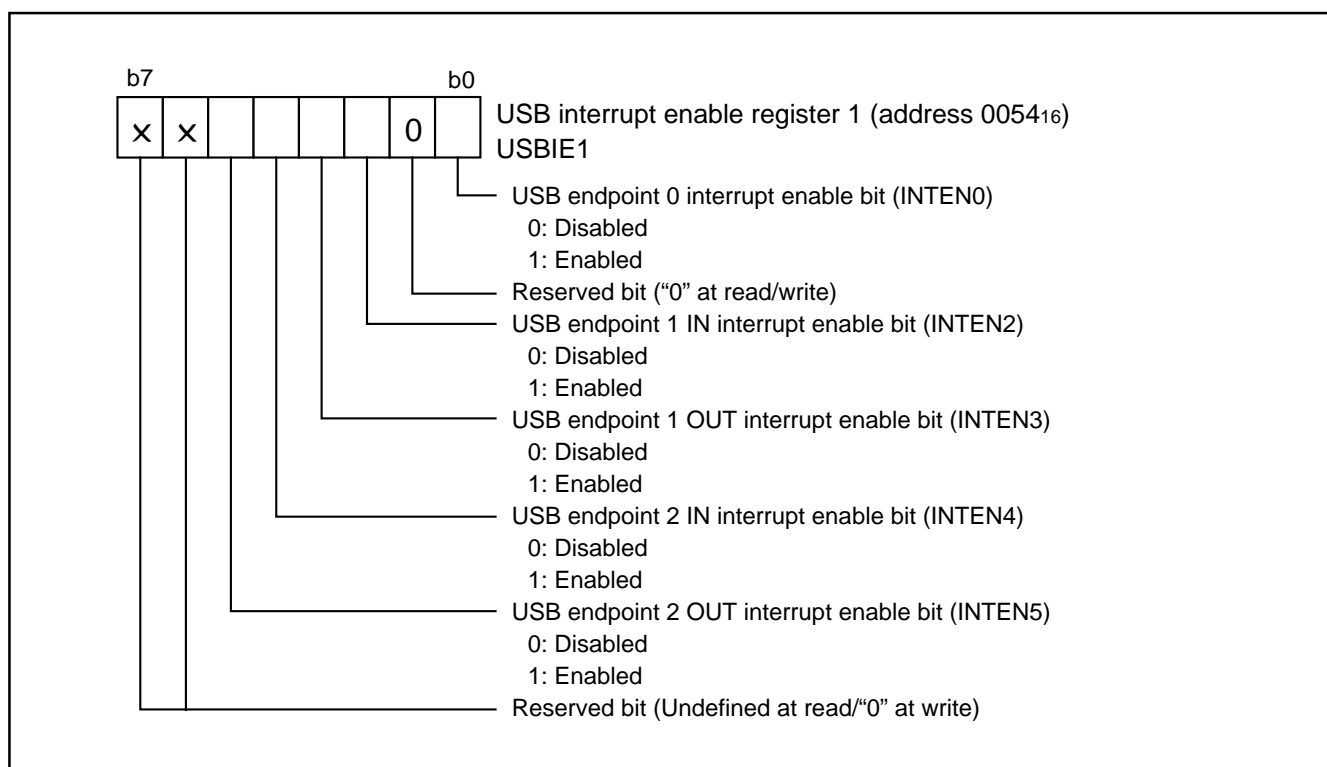


Fig. 40 Structure of USB interrupt enable register 1



**[USB Endpoint 0 IN Control Register ] IN\_CSR**

This register contains the control and status information of the endpoint 0. This USB FCU sets the OUT\_PKT\_RDY flag to "1" upon having received a data packet in the OUT FIFO. When reading its one data packet from the OUT FIFO, be sure to set this flag to "0".

After a SETUP token is received, the MCU is in the "decode wait state" until the OUT\_PKT\_RDY flag is cleared. If the OUT\_PKT\_RDY flag is not cleared (indicating that the host request has not been successfully decoded), the USB FCU keep returning a NAK to the host for all IN/OUT tokens.

Set the IN\_PKT\_RDY bit to "1" after the data packet has been written to the IN FIFO. If this bit is set to "1" even though nothing has been written to the IN FIFO, a "0" length data (NULL packet) is sent to the host. The SEND\_STALL bit is for sending a STALL to the host if an unsupported request is received by the USB FCU. This bit must be set to "1". When the OUT\_PKT\_RDY flag is set to "0" for request reception, the USB FCU transmits a STALL signal

to the Host CPU. Perform the following three processes simultaneously:

- Set SEND\_STALL bit to "1"
- Set DATA\_END bit to "1"
- Set OUT\_PKT\_RDY flag to "0" by setting SERVICED\_OUT\_PKT\_RDY bit to "1".

Note that if "0" is written to the SEND\_STALL bit before the CLEAR\_FEATURE (endpoint STALL) request has been received, the next STALL will not be generated.

The DATA\_END bit informs the USB FCU of the completion of the process indicated in the SETUP packet. Set this bit to "1" when the process requested in the SETUP packet is completed. (Control Read Transfer: set this bit after writing all of the requested data to the FIFO; Control Write Transfer: set this bit to "1" after reading all of the requested data from the FIFO.) When this bit is "1", the host request is ignored and a STALL is returned. After the status phase process is completed, the USB FCU automatically clears it to "0".

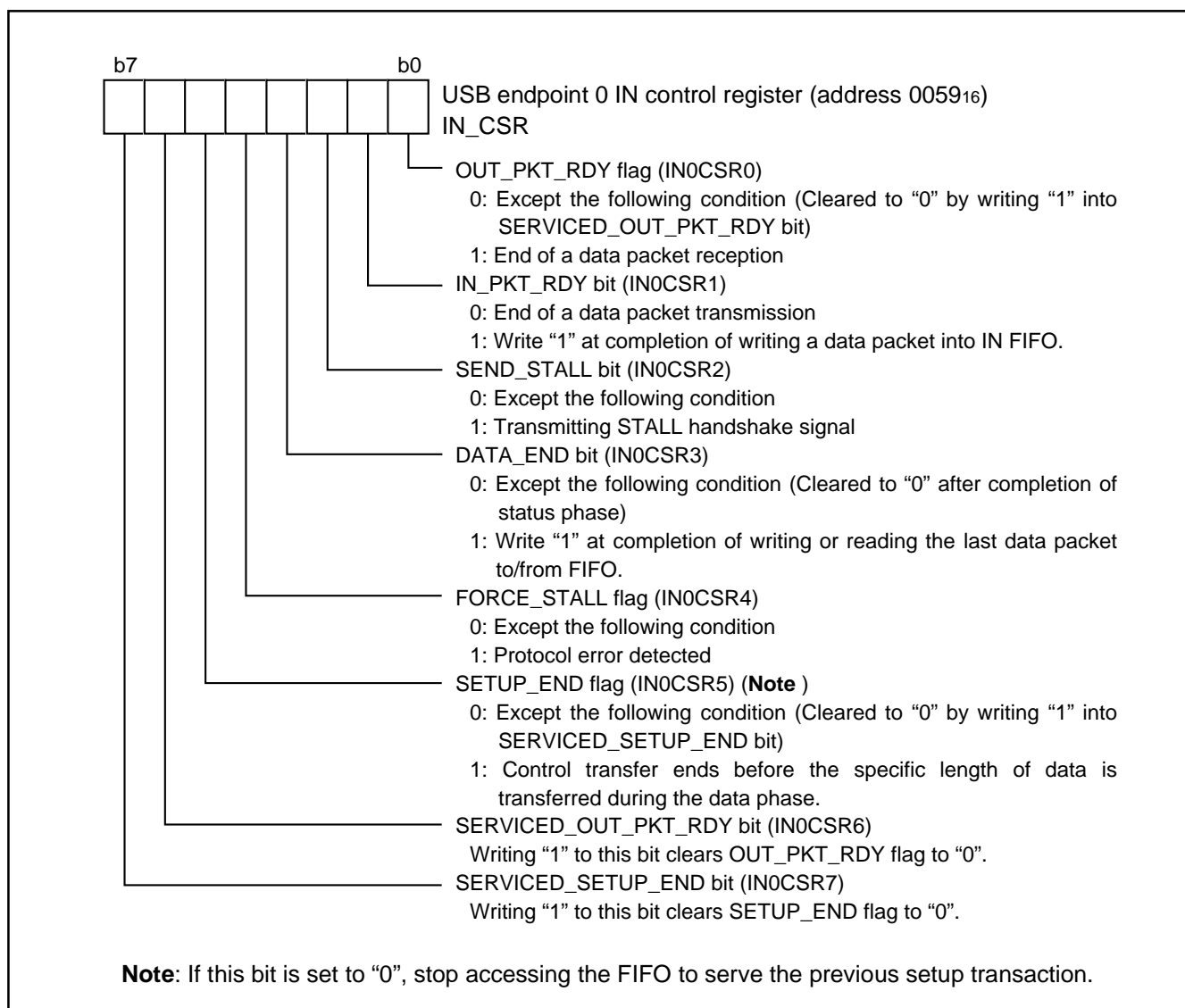


Fig. 43 Structure of USB endpoint 0 IN control register

**[USB Endpoint x (x = 1, 2) IN Control Register] IN\_CSR**

This register contains the control and status information of the respective IN Endpoints 1, 2.

Set the IN\_PKT\_RDY bit to "1" after the data packet has been written to the IN FIFO. This bit is cleared to "0" when the data transfer is completed. In a bulk IN transfer, this bit is cleared when an ACK signal is received from the host. If an ACK signal is not received, this bit (and the TX\_NOT\_EMPTY bit) remains as "1". This same data packet is sent after the next IN token is received. The FLUSH bit is for flushing the data in the IN FIFO.

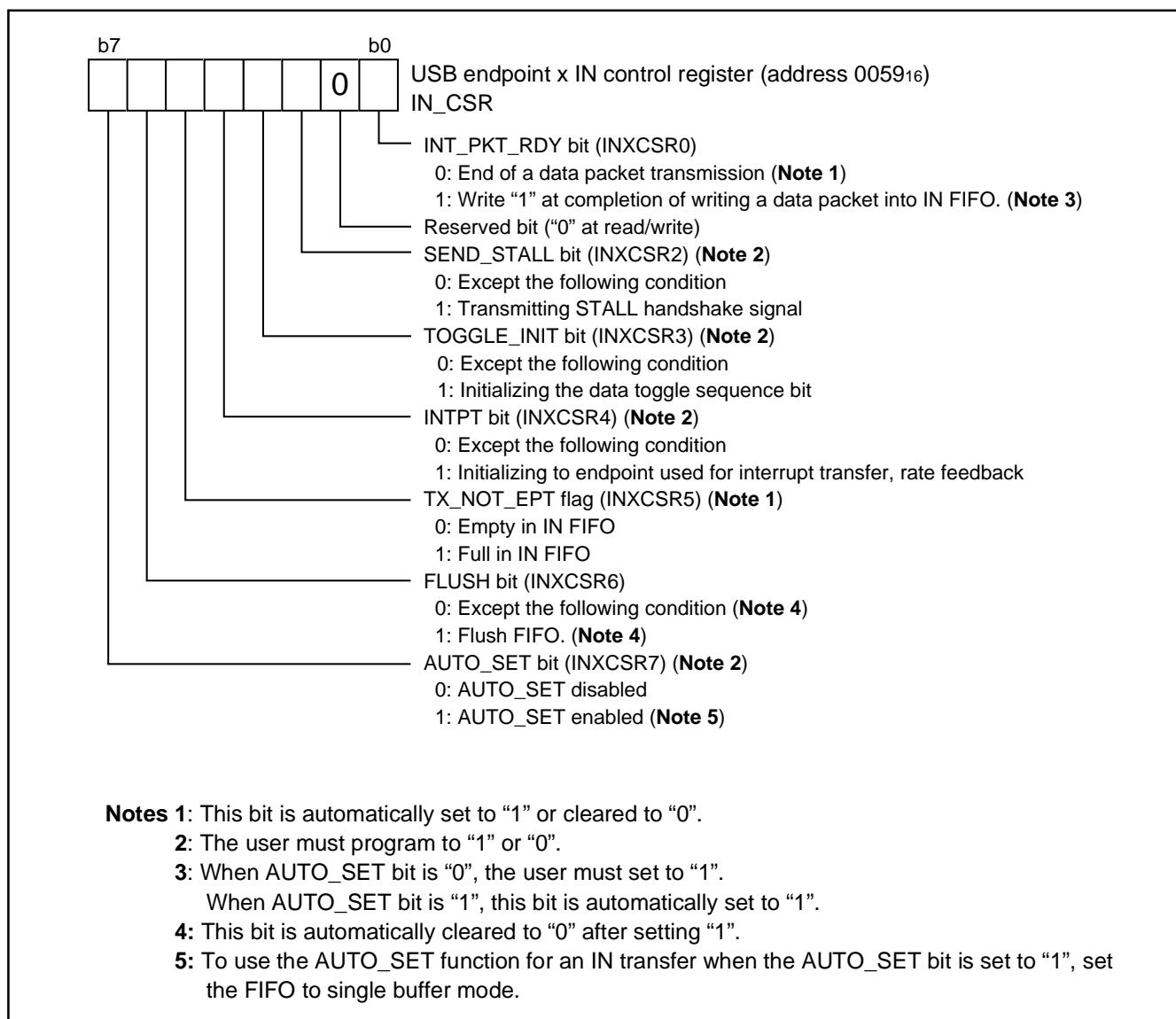


Fig. 44 Structure of USB endpoint x (x = 1, 2) IN control register

**[USB Endpoint x (x = 1, 2) OUT Control Register] OUT\_CSR**

This register contains the information and status of the respective OUT endpoints 1, 2. In the endpoint 0, all bits are reserved and cannot be used (they will all be read out as "0"). The USB FCU sets the OUT\_PKT\_RDY flag to "1" after a data packet has been received into the OUT FIFO. After reading the data packet in the OUT FIFO, clear this flag to "0". However, if there is still data in the OUT FIFO, the flag cannot be cleared even by writing "0" by software.

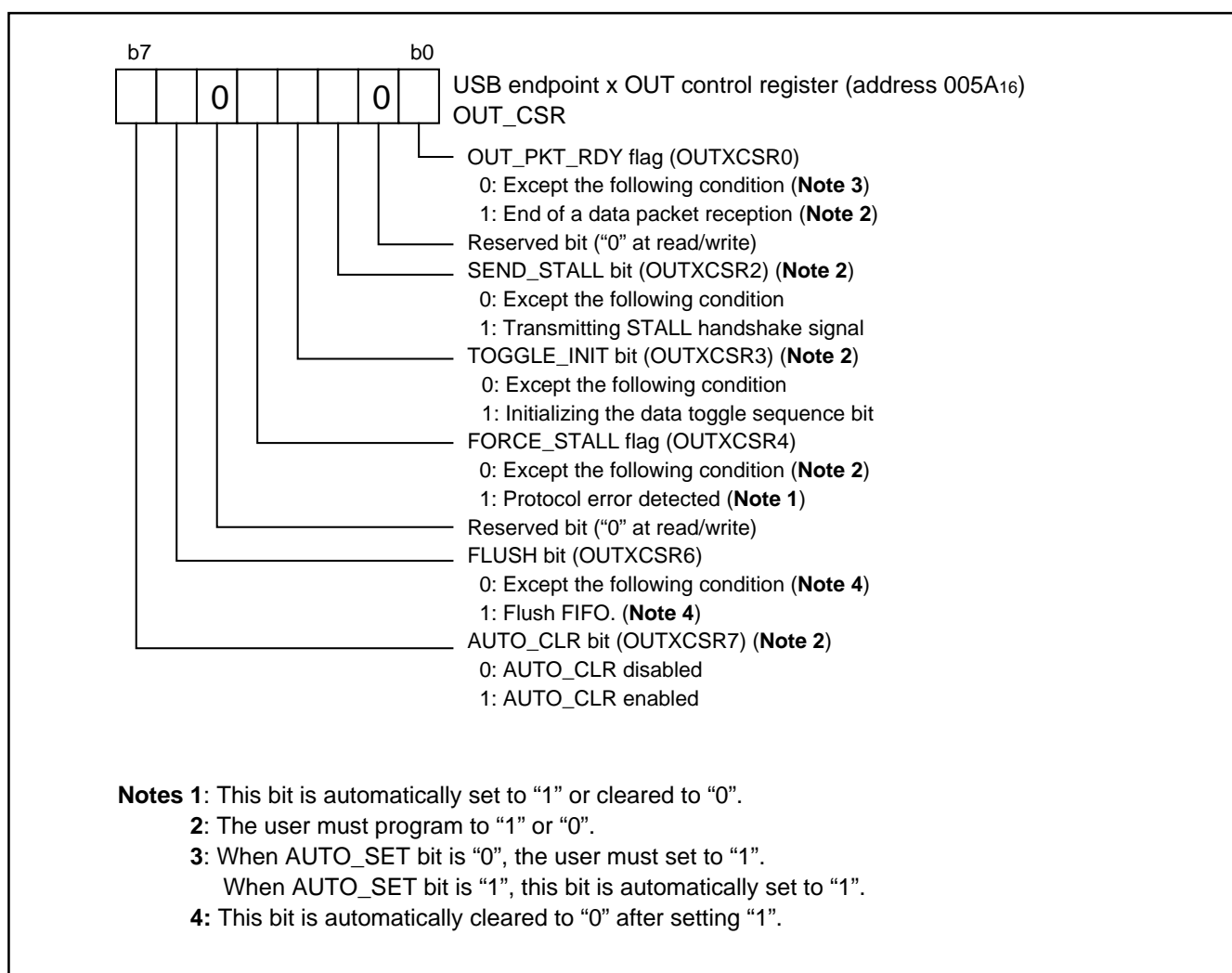


Fig. 45 Structure of USB endpoint x (x = 1, 2) OUT control register

**[USB Endpoint x (x = 0 to 2) IN Max. Packet Size Register] IN\_MAXP**

This register specifies the maximum packet size (MAXP) of an endpoint x IN packet. The value set for endpoint 1 is the number of transmitted bytes divided by 8, and the value set for endpoints 0 and 2 is the actual number of transmitted bytes. The CPU can change these values using the SET\_DESCRIPTOR command. The initial value for endpoints 0 and 2 is 8, and the initial value for endpoint 1 is 1.

**[USB Endpoint x (x = 0 to 2) OUT Max. Packet Size Register] OUT\_MAXP**

This register specifies the maximum packet size (MAXP) of an Endpoint x OUT packet. The value set for endpoint 1 is the number of received bytes divided by 8, and the value set for endpoints 0 and 2 is the actual number of received bytes. The CPU can change these values using the SET\_DESCRIPTOR command. The initial value for endpoints 0 and 2 is 8, and the initial value for

endpoint 1 is 1. When using the endpoint 0, both USB endpoint x IN max. packet size register (IN\_MAXP) and USB endpoint x OUT max. packet size register (OUT\_MAXP) are set to the same value. Changing one register's value effectively changes the value of the other register as well.

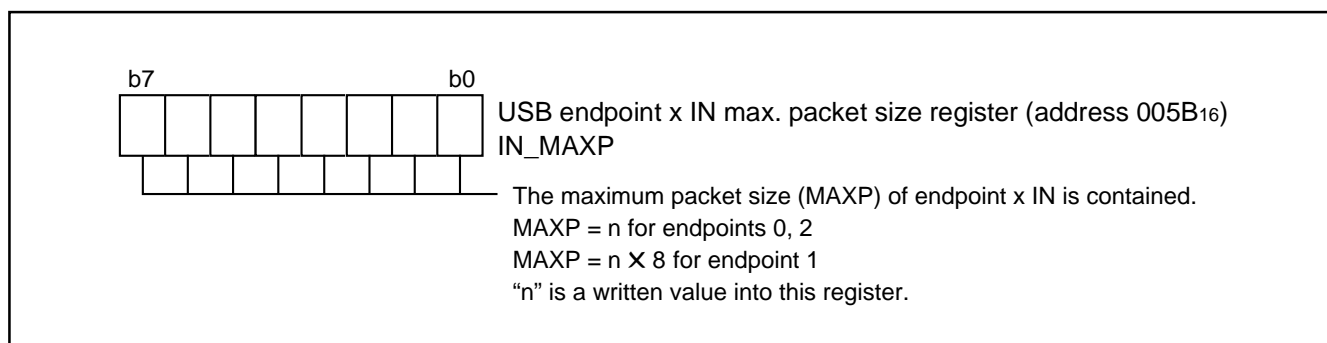


Fig. 46 Structure of USB endpoint x IN max. packet size register

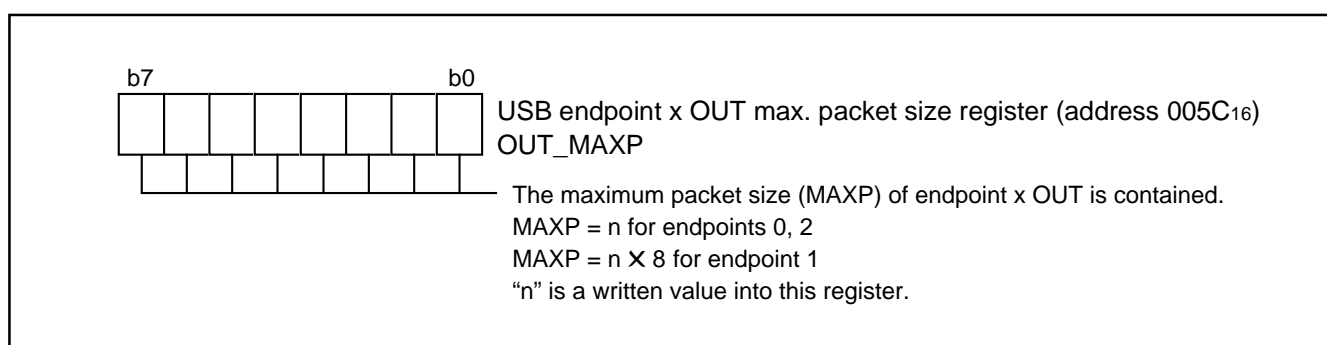


Fig. 47 Structure of USB endpoint x OUT max. packet size register

### [USB endpoint x (x = 0 to 2) OUT Write Count Registers] WRT\_CNTR

This register contains the number of bytes in the endpoint x OUT FIFO. This is a read-only register. This register must be read after the USB FCU has received a packet of data from the host.

When the OUT FIFO is in double buffer mode, the CPU first reads the received number of bytes of the former data packet. The next CPU read can obtain that of the new data packet.

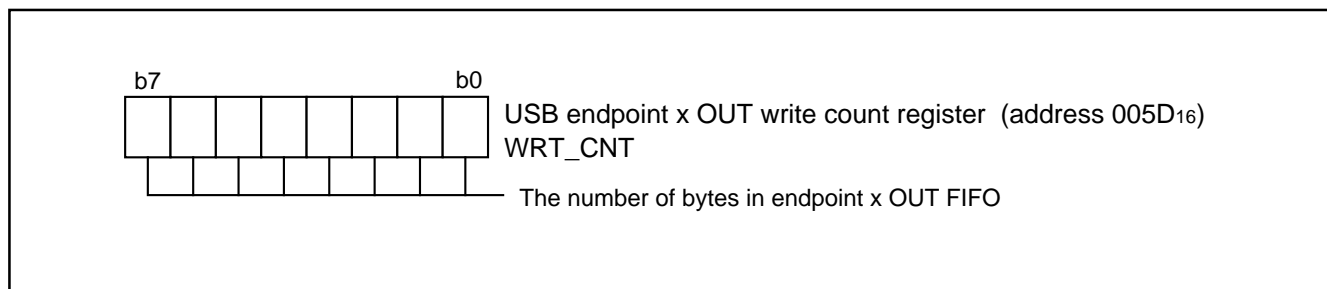


Fig. 48 Structure of USB endpoint x (x = 0 to 2) OUT write count registers

### [USB Endpoint x (x = 0 to 2) FIFO Register] USBFIFOx

These registers are the USB IN (transmit) and OUT (receive) FIFO data registers. Write data to the corresponding register, and read data from the corresponding register.

When the maximum packet size is equal to or less than half the FIFO size, these registers function in double buffer mode and can hold two packets of data. When the IN\_PKT\_RDY bit is "0" and

the TX\_NOT\_EMPTY bit is "1", these bits indicate that one packet of data is stored in the IN FIFO. When the OUT FIFO is in double buffer mode, the OUT\_PKT\_RDY flag remains as "1" after the first packet of data is read out (it actually goes to "0" and returns to "1" after one  $\phi$  cycle).

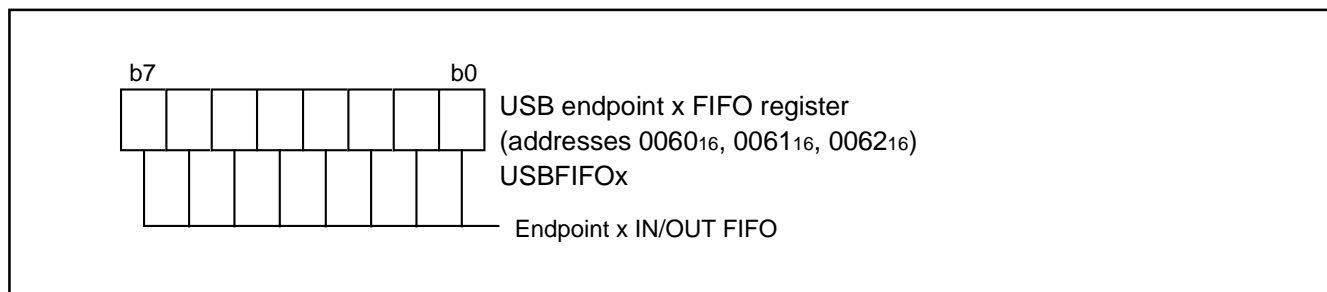


Fig. 49 Structure of USB endpoint x (x = 0 to 2) FIFO register



**[USB Endpoint FIFO Mode Selection Register] USBFIFOMR**

This register determines IN/OUT FIFO size mode for endpoint 1 or endpoint 2.

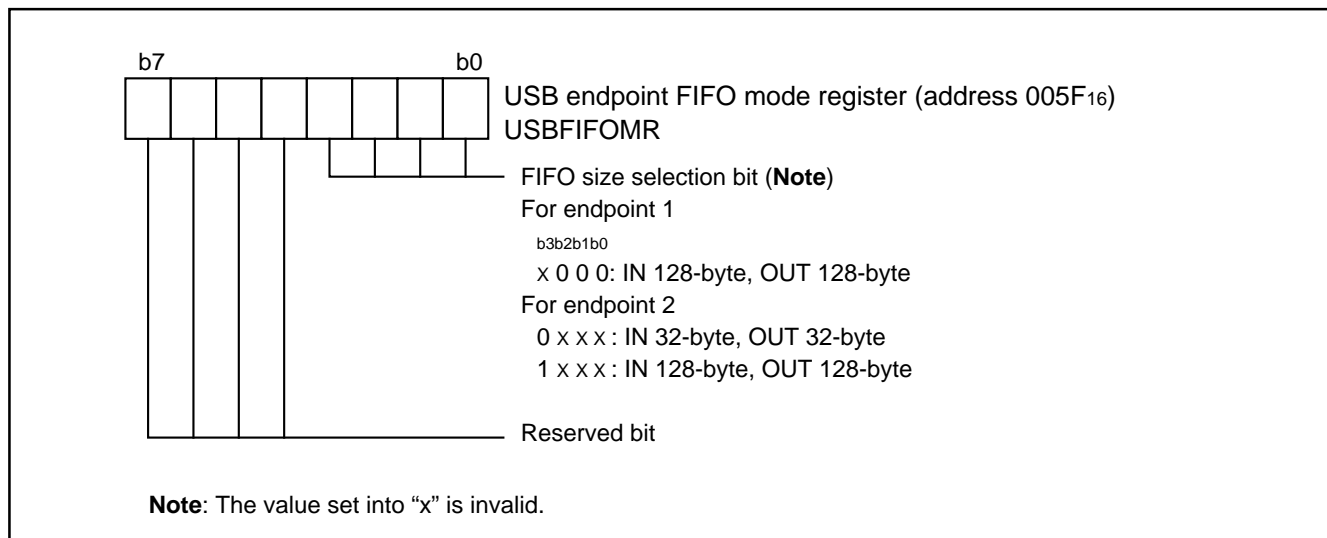


Fig. 50 Structure of USB endpoint FIFO mode register

## FREQUENCY SYNTHESIZER (PLL)

The frequency synthesizer generates the 48 MHz clock required by f<sub>USB</sub> and f<sub>SYN</sub>, which are multiples of the external input reference f(X<sub>IN</sub>). Figure 51 shows the block diagram for the frequency synthesizer circuit.

The Frequency Synthesizer Input Bit selects either f(X<sub>IN</sub>) or f(X<sub>CIN</sub>) as an input clock f<sub>IN</sub> for the frequency synthesizer.

The Frequency Synthesizer Multiply Register 2 (FSM2: address 006E16) divides f<sub>IN</sub> to generate f<sub>PIN</sub>, where

$$f_{PIN} = f_{IN} / 2(n + 1), \quad n: \text{value set to FSM2.}$$

When the value of Frequency Synthesizer Multiply Register 2 is set to 255, the division is not performed and f<sub>PIN</sub> will equal f<sub>IN</sub>.

f<sub>VCO</sub> is generated according to the contents of Frequency Synthesizer Multiply Register 1 (FSM1: address 006D16), where

$$f_{VCO} = f_{PIN} \times 2(n + 1), \quad n: \text{value set to FSM1.}$$

Set the value of FSM1 so that the value of f<sub>VCO</sub> is 48 MHz.

f<sub>SYN</sub> is generated according to the contents of the Frequency Synthesizer Divide Register (FSD: address 006F16), where

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

When the value of the Frequency Synthesizer Divide Register is set to 255, the division is not performed and f<sub>SYN</sub> becomes invalid.

### [Frequency Synthesizer Control Register] FSC

Setting the Frequency Synthesizer Enable Bit (FSE) to "1" enables the frequency synthesizer. When the Frequency Synthesizer Lock Status Bit (LS) is "1" in the frequency synthesizer enabled, this indicates that f<sub>SYN</sub> and f<sub>VCO</sub> have correct frequencies.

### ■Notes

Make sure to connect a low-pulse filter to the LPF pin when using the frequency synthesizer. In addition, please refer to "Programming Notes: Frequency Synthesizer" when recovering from a Hardware Reset.

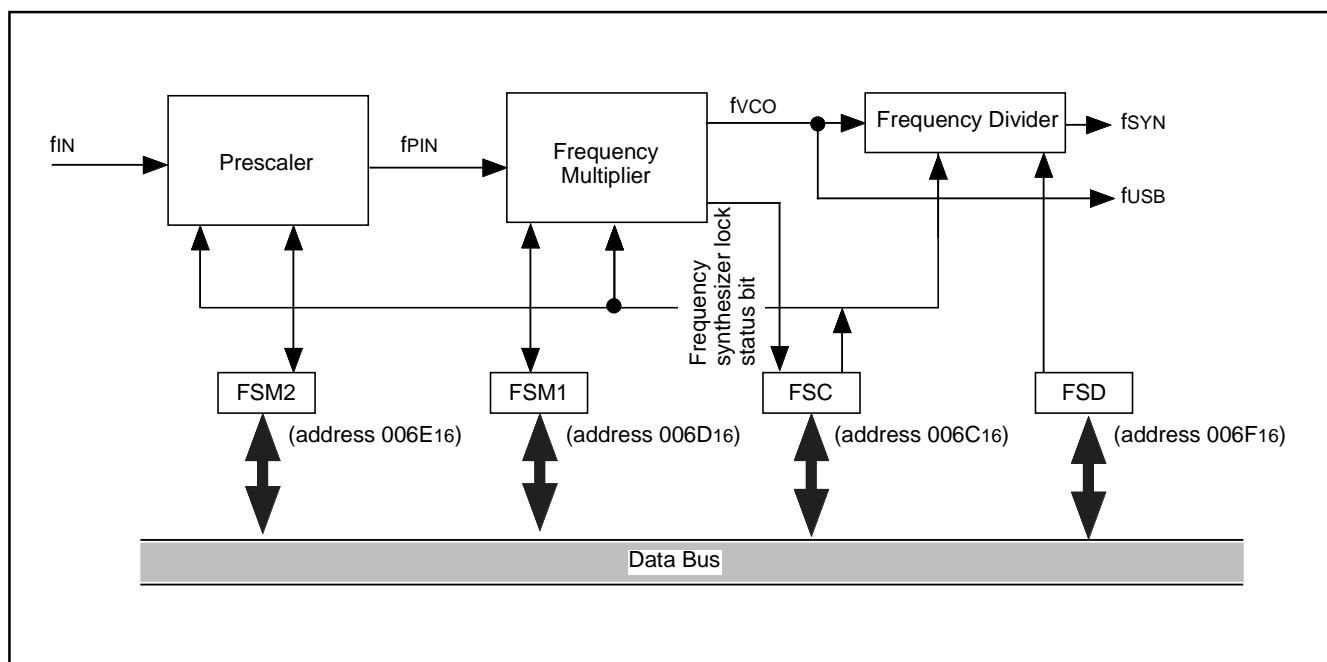


Fig. 51 Frequency synthesizer block diagram

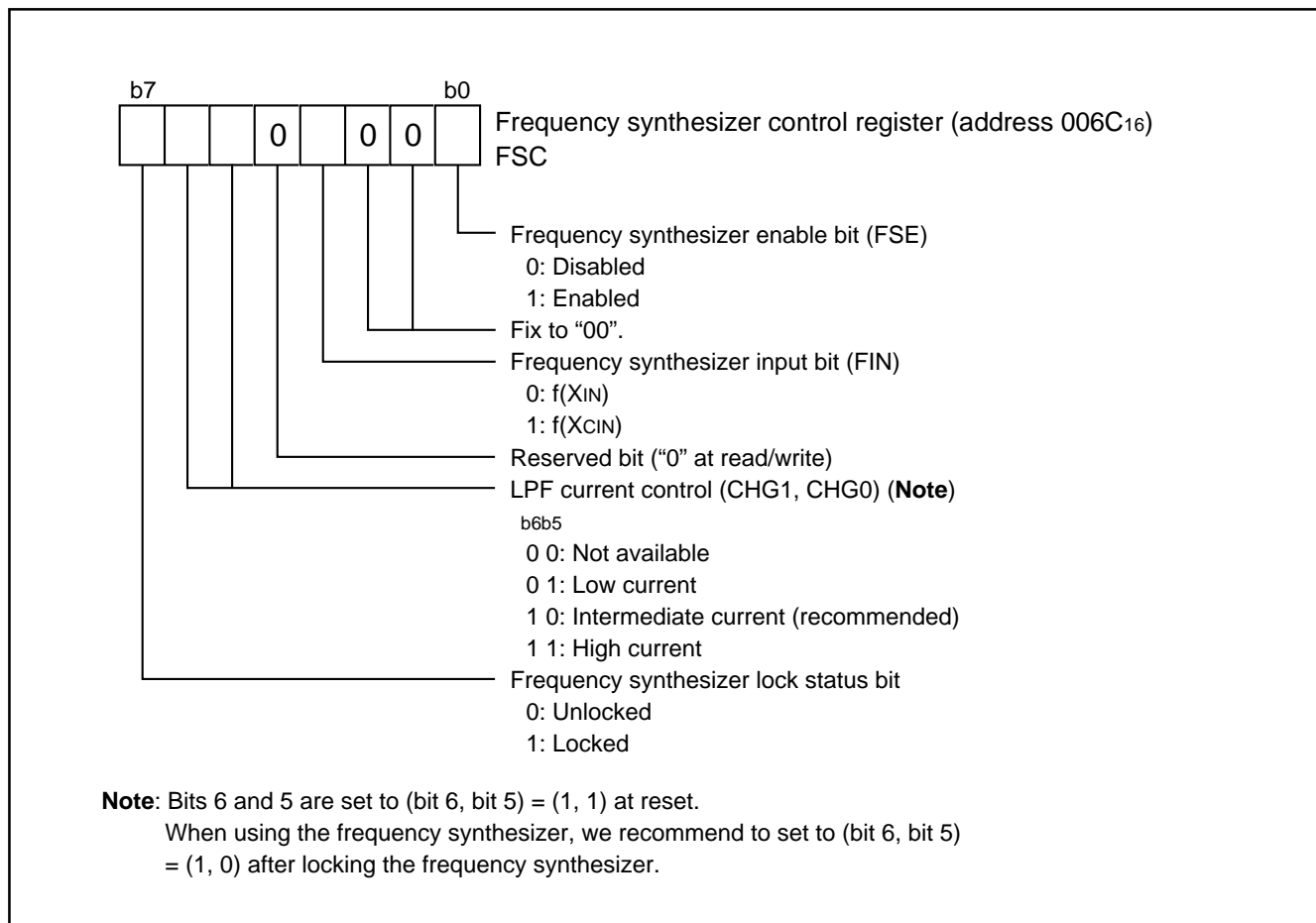


Fig. 52 Structure of frequency synthesizer control register

### RESET CIRCUIT

To reset the microcomputer, RESET pin should be held at an "L" level for 20 cycles or more of  $\phi$ . Then the RESET pin is returned to an "H" level, and reset is released. They must be performed when the power source voltages are between 3.00 V and 3.60 V or 4.15 V and 5.25 V.

After the reset is completed, the program starts from the address contained in address FFFA<sub>16</sub> (high-order byte) and address FFFB<sub>16</sub> (low-order byte).

After oscillation has restarted, the timers 1 and 2 secures waiting time for the internal clock  $\phi$  oscillation stabilized automatically by setting the timer 1 to "FF16" and timer 2 to "0116". The internal clock  $\phi$  retains "H" level until Timer 2's underflow and it cannot be supplied until the underflow.

The pins state during reset are follows:

- When CNVss = "H"
  - Ports P0, P1, P33 to P37 : Outputting
  - Pins other than above mentioned ports : Inputting
- When CNVss = "L"
  - All pins : Inputting.

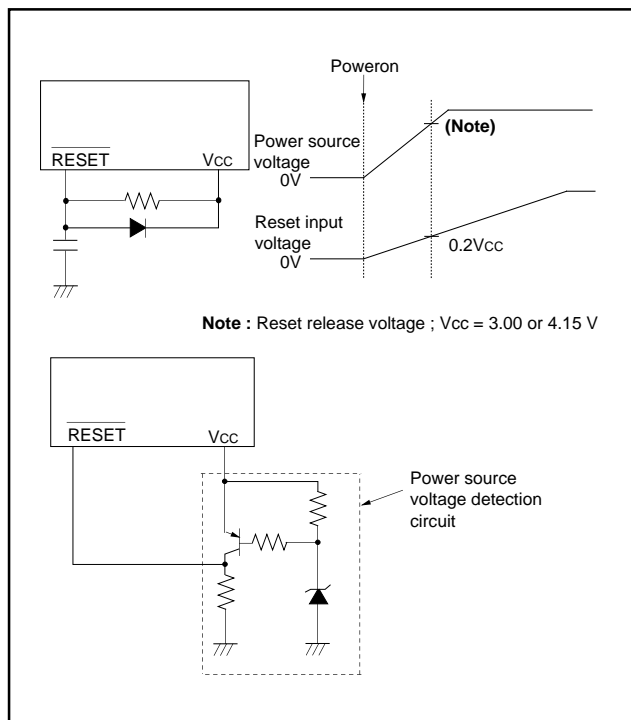


Fig. 53 Reset circuit example

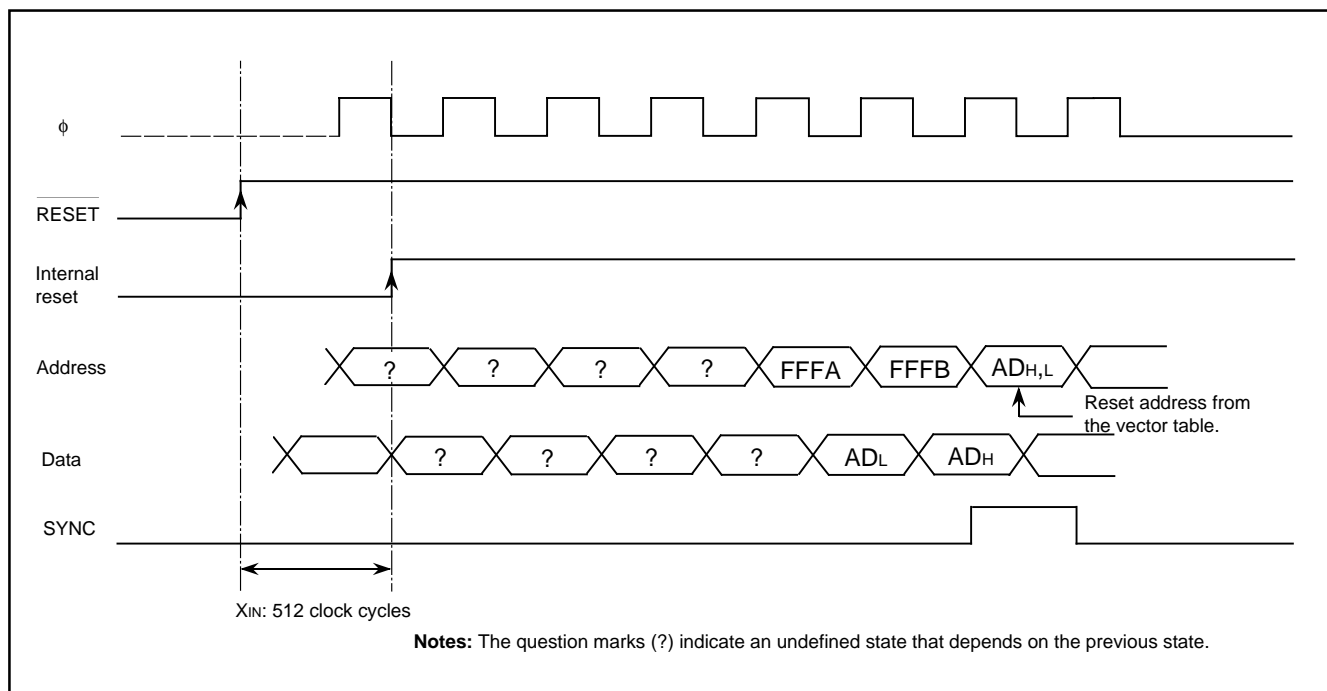


Fig. 54 Reset sequence

	Address	Register contents		Address	Register contents
(1) CPU mode register A (CPUA)	0000 <sub>16</sub>	000001100	(39) UART status register (USTS)	0032 <sub>16</sub>	000000011
(2) CPU mode register B (CPUB)	0001 <sub>16</sub>	100000011	(40) UART control register (UCON)	0033 <sub>16</sub>	00 <sub>16</sub>
(3) Interrupt request register A (IREQA)	0002 <sub>16</sub>	00 <sub>16</sub>	(41) UART RTS control register (URTSC)	0036 <sub>16</sub>	10000000
(4) Interrupt request register B (IREQB)	0003 <sub>16</sub>	00 <sub>16</sub>	(42) DMAC index and status register (DMAIS)	003F <sub>16</sub>	00 <sub>16</sub>
(5) Interrupt request register C (IREQC)	0004 <sub>16</sub>	00 <sub>16</sub>	(43) DMAC channel x mode register 1 (DMAx1)	0040 <sub>16</sub>	00 <sub>16</sub>
(6) Interrupt control register A (ICONA)	0005 <sub>16</sub>	00 <sub>16</sub>	(44) DMAC channel x mode register 2 (DMAx2)	0041 <sub>16</sub>	00 <sub>16</sub>
(7) Interrupt control register B (ICONB)	0006 <sub>16</sub>	00 <sub>16</sub>	(45) DMAC channel x source register Low (DMAxSL)	0042 <sub>16</sub>	00 <sub>16</sub>
(8) Interrupt control register C (ICONC)	0007 <sub>16</sub>	00 <sub>16</sub>	(46) DMAC channel x source register High (DMAxSH)	0043 <sub>16</sub>	00 <sub>16</sub>
(9) Port P0 (P0)	0008 <sub>16</sub>	00 <sub>16</sub>	(47) DMAC channel x destination register Low (DMAxDL)	0044 <sub>16</sub>	00 <sub>16</sub>
(10) Port P0 direction register (P0D)	0009 <sub>16</sub>	00 <sub>16</sub>	(48) DMAC channel x destination register High (DMAxDH)	0045 <sub>16</sub>	00 <sub>16</sub>
(11) Port P1 (P1)	000A <sub>16</sub>	00 <sub>16</sub>	(49) DMAC channel x transfer count register Low (DMAxCL)	0046 <sub>16</sub>	00 <sub>16</sub>
(12) Port P1 direction register (P1D)	000B <sub>16</sub>	00 <sub>16</sub>	(50) DMAC channel x transfer count register High (DMAxCH)	0047 <sub>16</sub>	00 <sub>16</sub>
(13) Port P2 (P2)	000C <sub>16</sub>	00 <sub>16</sub>	(51) USB address register (USBA)	0050 <sub>16</sub>	00 <sub>16</sub>
(14) Port P2 direction register (P2D)	000D <sub>16</sub>	00 <sub>16</sub>	(52) USB power management register (USBPM)	0051 <sub>16</sub>	00 <sub>16</sub>
(15) Port P3 (P3)	000E <sub>16</sub>	00 <sub>16</sub>	(53) USB interrupt status register 1 (USBIS1)	0052 <sub>16</sub>	00 <sub>16</sub>
(16) Port P3 direction register (P3D)	000F <sub>16</sub>	00 <sub>16</sub>	(54) USB interrupt status register 2 (USBIS2)	0053 <sub>16</sub>	00 <sub>16</sub>
(17) Port control register (PTC)	0010 <sub>16</sub>	00 <sub>16</sub>	(55) USB interrupt enable register 1 (USBIE1)	0054 <sub>16</sub>	X X 1 1 1 1 1 1
(18) Interrupt polarity select register (IPOL)	0011 <sub>16</sub>	00 <sub>16</sub>	(56) USB interrupt enable register 2 (USBIE2)	0055 <sub>16</sub>	0 0 1 X 0 0 X X
(19) Port P2 pull-up control register (PUP2)	0012 <sub>16</sub>	00 <sub>16</sub>	(57) USB endpoint index register (USBINDEX)	0058 <sub>16</sub>	00 <sub>16</sub>
(20) USB control register (USBC)	0013 <sub>16</sub>	00 <sub>16</sub>	(58) USB endpoint x IN control register (IN_CSR)	0059 <sub>16</sub>	00 <sub>16</sub>
(21) Port P6 (P6)	0014 <sub>16</sub>	00 <sub>16</sub>	(59) USB endpoint x OUT control register (OUT_CSR)	005A <sub>16</sub>	00 <sub>16</sub>
(22) Port P6 direction register (P6D)	0015 <sub>16</sub>	00 <sub>16</sub>	(60) USB endpoint x IN max. packet size register (IN_MAXP) (Note 1)	005B <sub>16</sub>	000001000
(23) Port P5 (P5)	0016 <sub>16</sub>	00 <sub>16</sub>	(61) USB endpoint x OUT max. packet size register (OUT_MAXP) (Note 1)	005C <sub>16</sub>	000001000
(24) Port P5 direction register (P5D)	0017 <sub>16</sub>	00 <sub>16</sub>	(62) USB endpoint x OUT write count register (WRT_CNT)	005D <sub>16</sub>	00 <sub>16</sub>
(25) Port P4 (P4)	0018 <sub>16</sub>	00 <sub>16</sub>	(63) USB endpoint FIFO mode register (USBFIFOMR)	005F <sub>16</sub>	00 <sub>16</sub>
(26) Port P4 direction register (P4D)	0019 <sub>16</sub>	00 <sub>16</sub>	(64) Flash memory control register (FMCR) (Note 3)	006A <sub>16</sub>	000000001
(27) Port P7 (P7)	001A <sub>16</sub>	00 <sub>16</sub>	(65) Frequency synthesizer control register (FSC)	006C <sub>16</sub>	011100000
(28) Port P7 direction register (P7D)	001B <sub>16</sub>	00 <sub>16</sub>	(66) Frequency synthesizer multiply register 1 (FSM1)	006D <sub>16</sub>	FF <sub>16</sub>
(29) Port P8 (P8)	001C <sub>16</sub>	00 <sub>16</sub>	(67) Frequency synthesizer multiply register 2 (FSM2)	006E <sub>16</sub>	FF <sub>16</sub>
(30) Port P8 direction register (P8D)	001D <sub>16</sub>	00 <sub>16</sub>	(68) Frequency synthesizer divide register (FSM2)	006F <sub>16</sub>	FF <sub>16</sub>
(31) Clock control register (CCR)	001F <sub>16</sub>	00 <sub>16</sub>	(69) ROM code protect control register (ROMCP) (Note 3)	FFC9 <sub>16</sub>	FF <sub>16</sub>
(32) Timer 1 (T1)	0024 <sub>16</sub>	FF <sub>16</sub>	(70) Processor status register	(PS)	X X X X X X X X
(33) Timer 2 (T2)	0025 <sub>16</sub>	00000001	(71) Program counter	(PC <sub>H</sub> )	FFF <sub>16</sub> contents
(34) Timer 3 (T3)	0026 <sub>16</sub>	FF <sub>16</sub>		(PC <sub>L</sub> )	FFFA <sub>16</sub> contents
(35) Timer 123 mode register (T123M)	0029 <sub>16</sub>	00 <sub>16</sub>			
(36) Serial I/O control register 1 (SIOCON1)	002B <sub>16</sub>	01000000			
(37) Serial I/O control register 2 (SIOCON2)	002C <sub>16</sub>	00011000			
(38) UART mode register (UMOD)	0030 <sub>16</sub>	00 <sub>16</sub>			

X : Not fixed

**Notes 1:** When using the endpoint 0 or endpoint 2, this contents are "08<sub>16</sub>". When using the endpoint 1, this contents are "01<sub>16</sub>".

**2:** Since the initial values for other than above mentioned registers and RAM contents are indefinite at reset, they must be set.

**3:** The flash memory control register and the ROM code protect control register exists in the flash memory version only.

Fig. 55 Internal status at reset

## CLOCK GENERATING CIRCUIT

The 7643 group has two built-in oscillation circuits. An oscillation circuit can be formed by connecting a resonator between XIN and XOUT (XCIN and XCOUT). Use the circuit constants in accordance with the resonator manufacturer's recommended values. No external resistor is needed between XIN and XOUT since a feed-back resistor exists on-chip. (An external feed-back resistor may be needed depending on conditions.) However, an external feed-back resistor is needed between XCIN and XCOUT.

When using an external clock, input the clocks to the XIN or XCIN pin and leave the XOUT or XCOUT pin open.

Immediately after power on, only the XIN oscillation circuit starts oscillating, and XCIN and XCOUT pins function as I/O ports.

## Frequency Control

The internal system clock can be selected among  $f_{SYN}$ ,  $f(XIN)$ ,  $f(XIN)/2$ , and  $f(XCIN)$ . The internal clock  $\phi$  is half the frequency of internal system clock.

### (1) $f_{SYN}$ clock

This is made by the frequency synthesizer.  $f(XIN)$  or  $f(XCIN)$  can be selected as its input clock. See also section "FREQUENCY SYNTHESIZER".

### (2) $f(XIN)$ clock

The frequency of internal system clock is the frequency of XIN pin.

### (3) $f(XIN)/2$ clock

The frequency of internal system clock is half the frequency of XIN pin.

### (4) $f(XCIN)$ clock

The frequency of internal system clock is the frequency of XCIN pin.

### ■Note

If you switch the oscillation between XIN - XOUT and XCIN - XCOUT, stabilize both XIN and XCIN oscillations. The sufficient time is required for the XCIN oscillation to stabilize, especially immediately after power on and at returning from the stop mode.

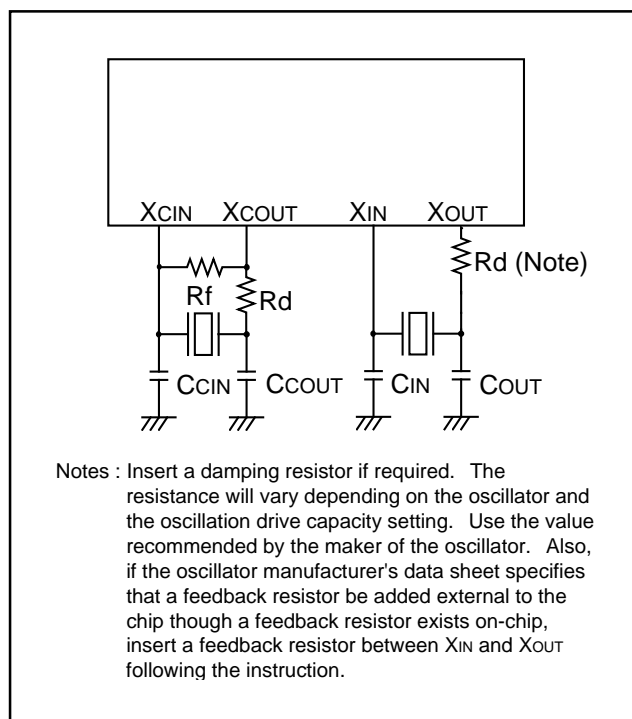


Fig. 56 Ceramic resonator or quartz-crystal oscillator external circuit

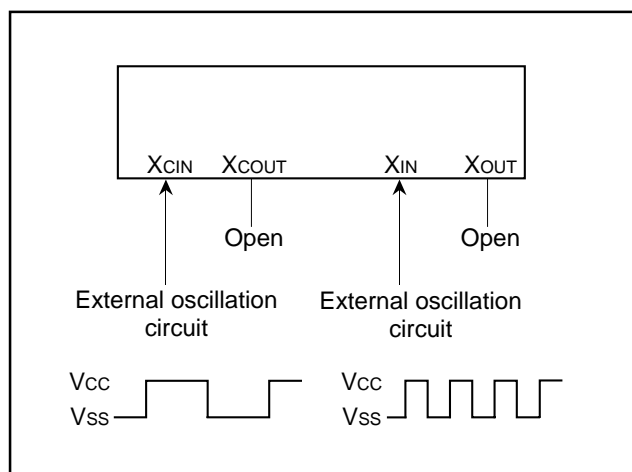


Fig. 57 External clock input circuit

### (5) Low power dissipation mode

- The low power dissipation operation can be realized by stopping the main clock  $X_{IN}$  when using  $f(X_{CIN})$  as the internal system clock. To stop the main clock, set the Main Clock ( $X_{IN}$ - $X_{OUT}$ ) Stop Bit of the CPU mode register A to "1".
- The low power dissipation operation can be realized by disabling the reversed amplifier when inputting external clocks to the  $X_{IN}$  pin or  $X_{CIN}$  pin. To disable the reversed amplifier, set the  $X_{COUT}$  Oscillation Drive Disable Bit (CCR5) or  $X_{OUT}$  Oscillation Drive Disable Bit (CCR6) of the clock control register to "1".

## Oscillation Control

### (1) Stop mode

If the STP instruction is executed, the internal clock  $\phi$  stops at "H" level, and  $X_{IN}$  and  $X_{CIN}$  oscillators stop. Then the timer 1 is set to "FF16" and the internal clock  $\phi$  divided by 8 is automatically selected as its count source. Additionally, the timer 2 is set to "0116" and the timer 1's output is automatically selected as its count source.

Set the Timer 1 and Timer 2 Interrupt Enable Bits to disabled ("0") before executing the STP instruction. When using an external interrupt to release the stop mode, set the Interrupt Enable Bit to be used to enabled ("1") and the Interrupt Disable Flag (I) to "0".

Oscillator restarts at reset or when an external interrupt including USB resume interrupts is received, but the internal clock  $\phi$  remains at "H" until the timer 2 underflows. The internal clock  $\phi$  is supplied for the first time when the timer 2 underflows. Therefore make sure not to set the Timer 1 Interrupt Request Bit and Timer 2 Interrupt Request Bit to "1" before the STP instruction stops the oscillator.

### (2) Wait mode

If the WIT instruction is executed, the internal clock  $\phi$  stops at "H" level, but the oscillator does not stop. The internal clock  $\phi$  restarts at reset or when an interrupt is received. Since the oscillator does not stop, normal operation can be started immediately after the internal clock  $\phi$  is restarted.

Set the Interrupt Enable Bit to be used to release the wait mode to enabled ("1") and the Interrupt Disable Flag (I) to "0".

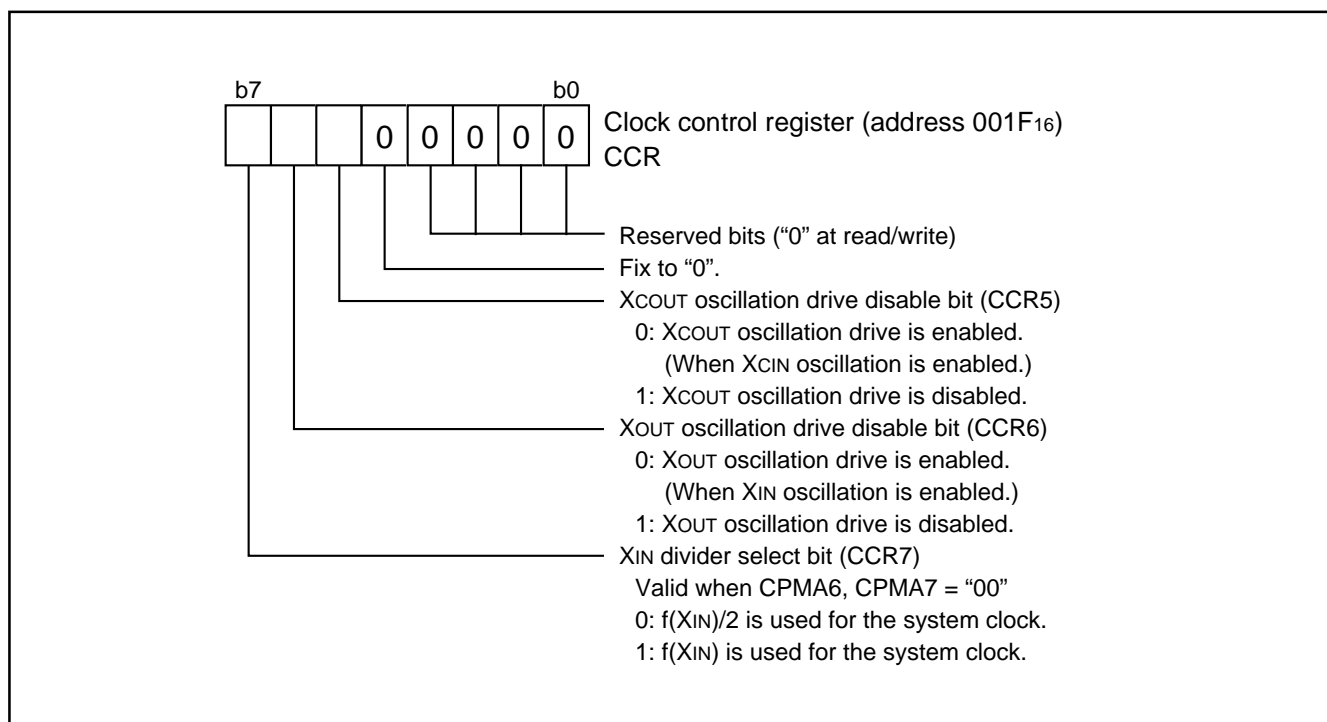


Fig. 58 Structure of clock control register

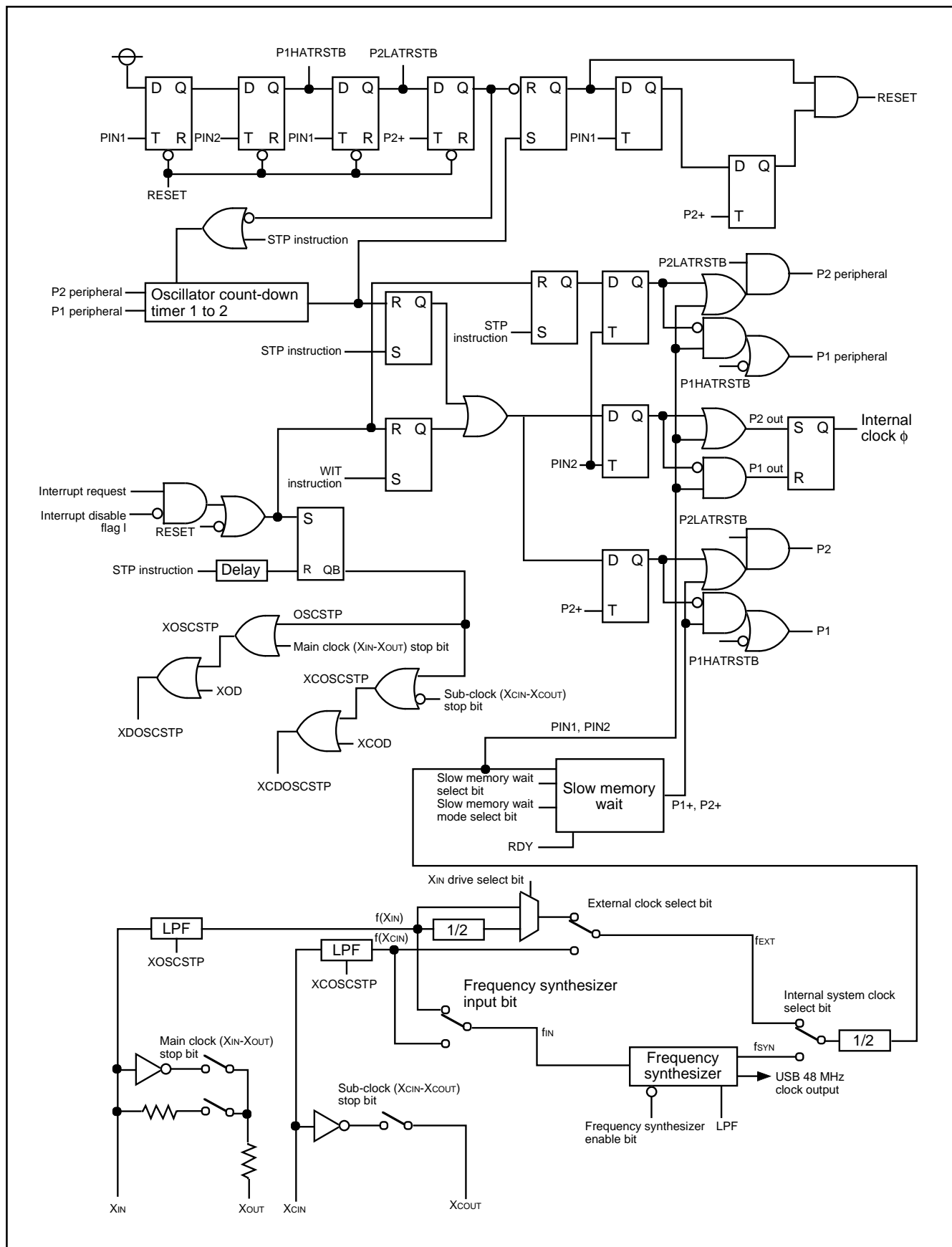


Fig. 59 Clock generating circuit block diagram



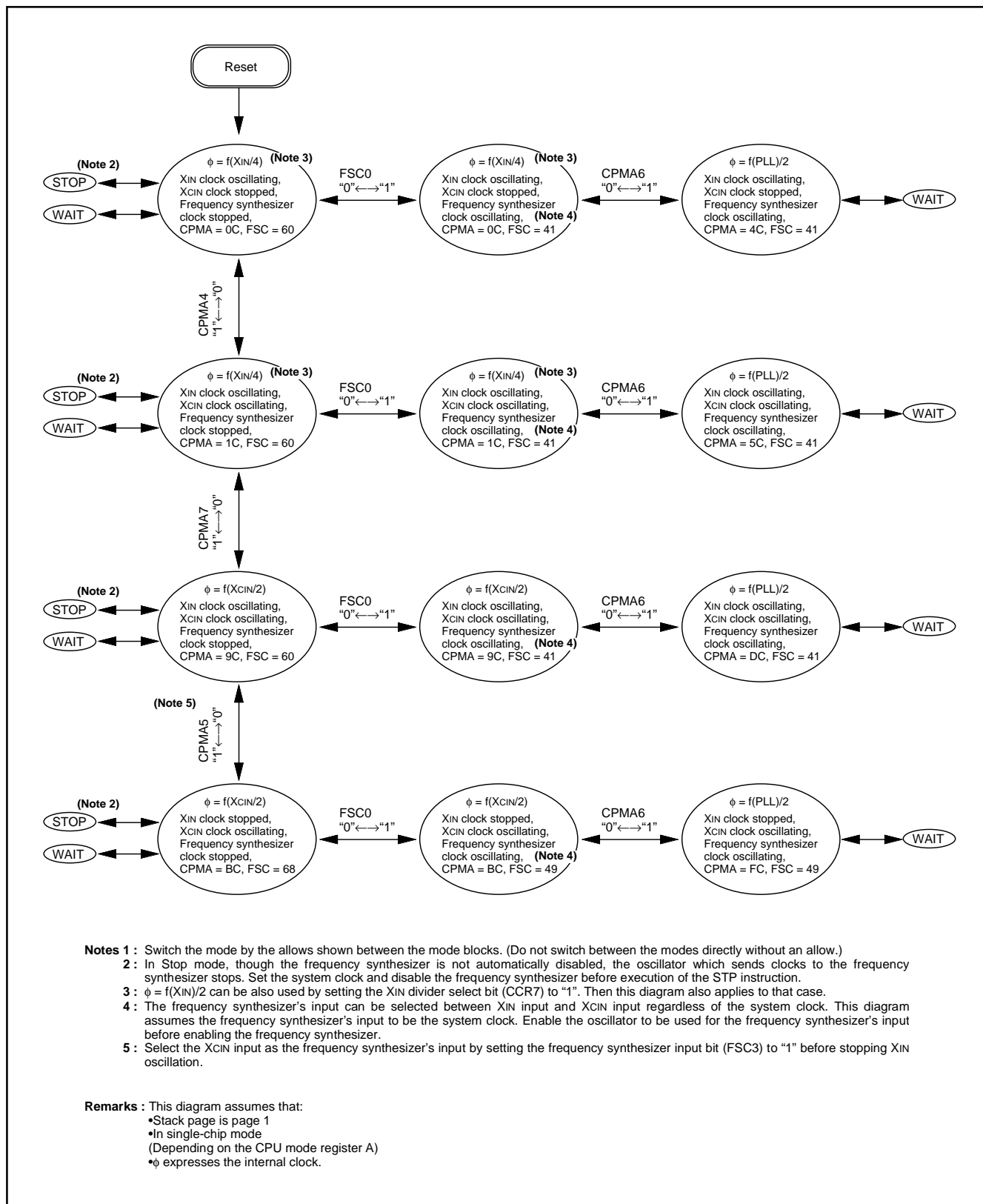


Fig. 60 State transitions of clock

## PROCESSOR MODE

Single-chip mode, memory expansion mode, and microprocessor mode which is only in the mask ROM version can be selected by using the Processor Mode Bits of CPU mode register A (bits 0 and 1 of address 0000<sub>16</sub>). In the memory expansion mode and microprocessor mode, a memory can be expanded externally via ports P0 to P3. In these modes, ports P0 to P3 lose their I/O port functions and become bus pins.

The port direction registers corresponding to those ports become external memory areas.

**Table 8 Port functions in memory expansion mode and microprocessor mode**

Port Name	Function
Port P0	Outputs low-order 8 bits of address.
Port P1	Outputs high-order 8 bits of address.
Port P2	Operates as I/O pins for data D7 to D0 (including instruction code).
Port P3	P30 is the RDY input pin. P31 and P32 function only as output pins P33 is the DMAOUT output pin. P34 is the $\phi$ OUT output pin. P35 is the SYNCOUT output pin. P36 is the $\overline{\text{WR}}$ output pin, and P37 is the $\overline{\text{RD}}$ output pin.
Port P4	P40 is the $\overline{\text{EDMA}}$ pin.

### (1) Single-chip mode

Select this mode by resetting the MCU with CNVss connected to Vss.

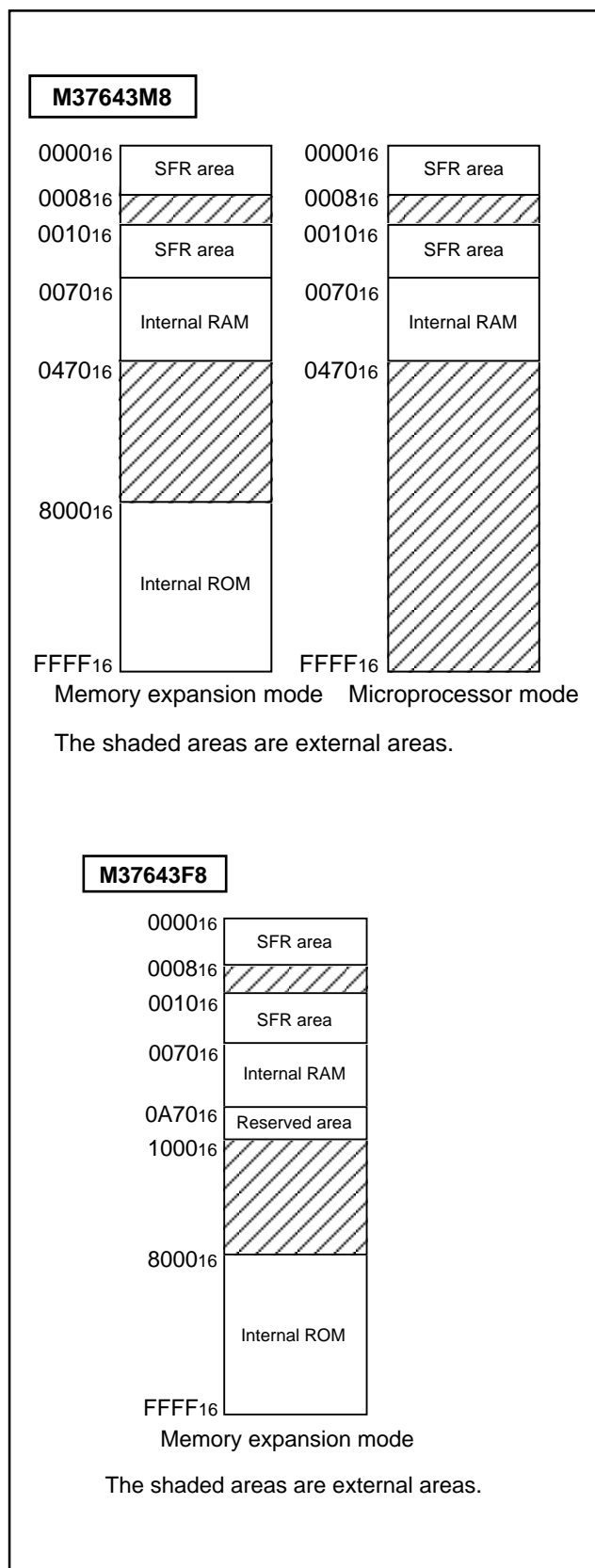
### (2) Memory expansion mode

Select this mode by setting the Processor Mode Bits (b1, b0) to "01" in software with CNVss connected to Vss. This mode enables external memory expansion while maintaining the validity of the internal ROM.

### (3) Microprocessor mode

Select this mode by resetting the MCU with CNVss connected to Vcc, or by setting the Processor Mode Bits (b1, b0) to "10" in software with CNVss connected to Vss. In the microprocessor mode, the internal ROM is no longer valid and an external memory must be used.

Do not set this mode in the flash memory version.



**Fig. 61 Memory maps in processor modes other than single-chip mode**



### Slow Memory Wait

The 7643 Group is equipped with the slow memory wait function (Software wait, RDY wait, and Extended RDY wait: software wait plus RDY input anytime wait) for easier interfacing with external devices that have long access times. The slow memory wait function can be enabled in the memory expansion mode and microprocessor mode. The appropriate wait mode is selected by setting bits 0 to 3 of CPU mode register B (address 000116). This function can extend the read cycle or write cycle only for access to an external memory. However, this wait function cannot be enabled for access to addresses 000816 to 000F16.

#### (1) Software wait

The software wait is selected by setting "00" to the Slow Memory Wait Mode Select Bits of CPU mode register B (address 000116). Read/write cycles ("L" width of RD pin/WR pin) can be extended by one to three  $\phi$  cycles. The number of cycles to be extended can be selected with the Slow Memory Wait Select Bits. When the software wait function is selected, the RDY pin status becomes invalid.

#### (2) RDY wait

RDY Wait is selected by setting "10" to the Slow Memory Wait Mode Select Bits of CPU mode register B (address 000116). When a fixed time of "L" is input to the RDY pin at the beginning of a read/write cycle (before  $\phi$  cycle falls), the MCU goes to the RDY state. The read/write cycle can then be extended by one to three  $\phi$  cycles. The number of  $\phi$  cycles to be added can be selected by the Slow Memory Wait Bits.

#### (3) Software wait + Extended RDY wait

Extended RDY Wait is selected by setting "11" to the Slow Memory Wait Mode Select Bits of CPU mode register B (address 000116). The read/write cycle can be extended when a fixed time of "L" is input to the RDY pin at the beginning of a read cycle (before  $\phi$  cycle falls). The RDY pin state is checked continually at each fall of  $\phi$  cycle until the RDY pin goes to "H". When "H" is input to the RDY pin, the wait is released within 1, 2, or 3  $\phi$  cycles (as selected with the Slow Memory Wait Bits).

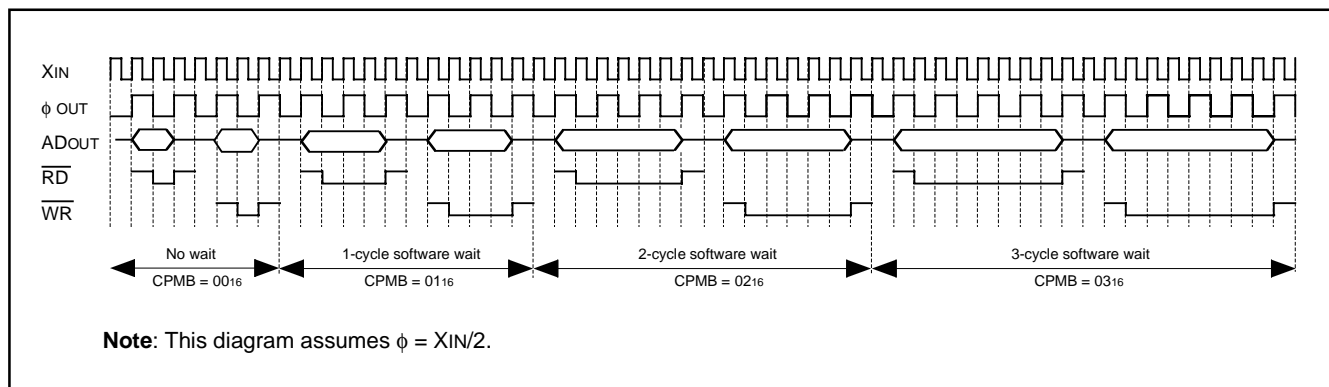


Fig. 64 Software wait timing diagram

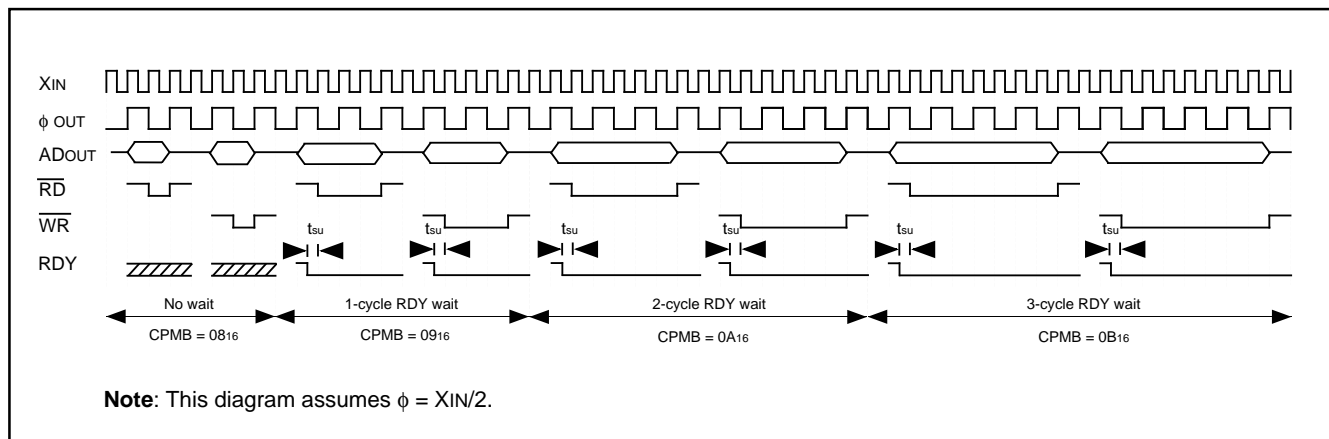


Fig. 65 RDY wait timing diagram

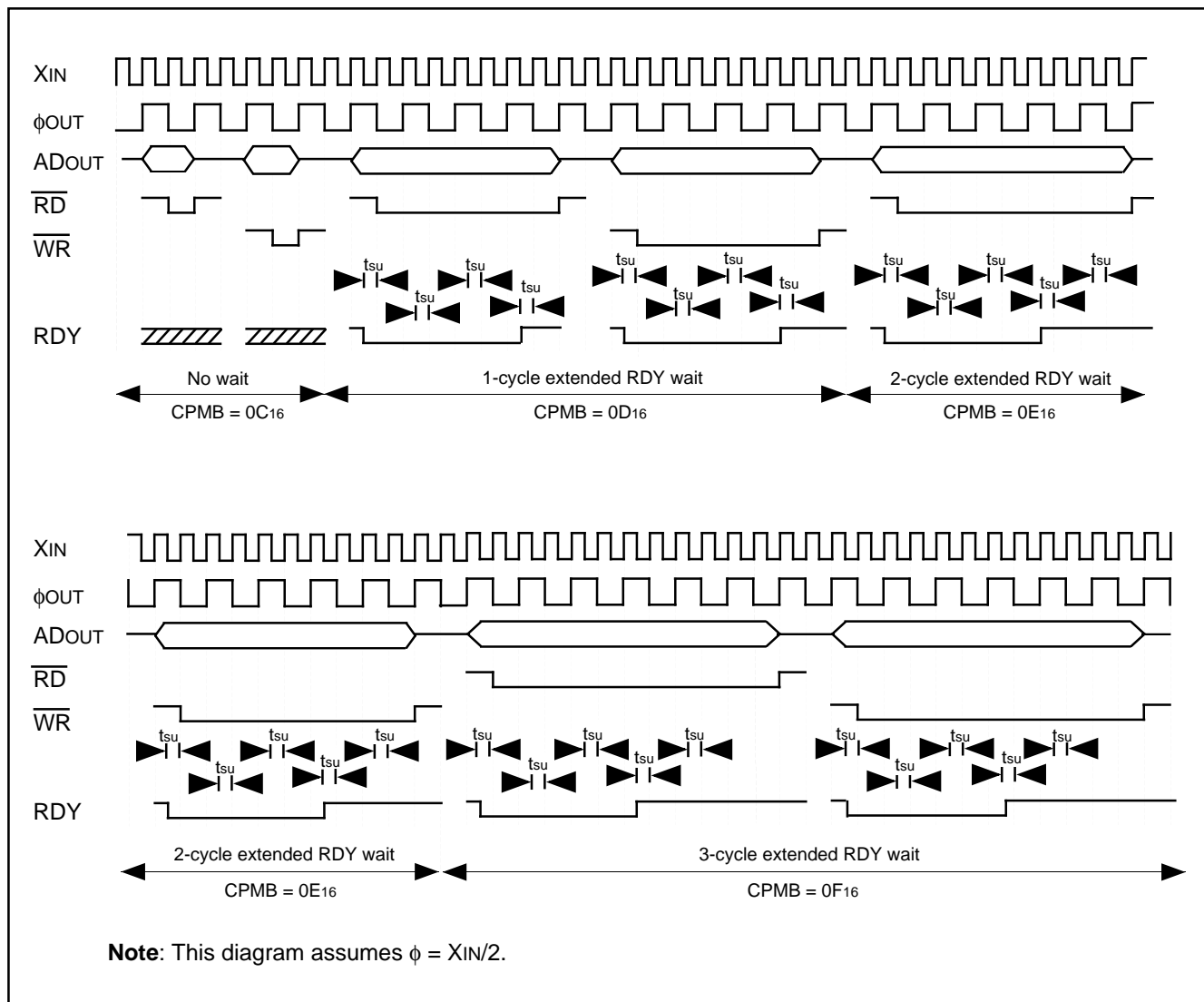


Fig. 66 Extended RDY wait (software wait plus RDY input anytime wait) timing diagram

## HOLD Function

The HOLD function is used for systems that consist of external circuits that access MCU buses without use of the CPU (Central Processing Unit). The HOLD function is used to generate the timing in which the MCU will relinquish the bus from the CPU to the external circuits. To use the HOLD function, set the HOLD function Enable Bit of CPU mode register B (address 000116) to "1". This function can be used with both the  $\overline{\text{HOLD}}$  pin and the  $\overline{\text{HLDA}}$  pin.

The HOLD signal is a signal from an external circuit requesting the MCU to relinquish use of the bus. When "L" level is input, the MCU goes to the HOLD state and remains so while the pin is at "L". The oscillator does not stop oscillating during the HOLD state, therefore allowing the internal peripheral functions to operate during this time.

When the MCU relinquishes use of the bus, "L" level is output from the  $\overline{\text{HLDA}}$  pin. The MCU makes ports P0 and P1 (address buses) and port P2 (data bus) tri-state outputs and holds port P37 (RD pin) and port P36 (WR pin) "H" level. Port P34 ( $\phi$  OUT pin) continues to oscillate. This function is not valid when the MCU is using the  $\text{IBF}_1$  function with the  $\overline{\text{HLDA}}$  pin.

## Expanded Data Memory Access

In Expanded Data Memory Access Mode, the MCU can access a data area larger than 64 Kbytes with the LDA (\$zz), Y (indirect Y) instruction and the STA (\$zz), Y (indirect Y) instruction.

To use this mode, set the Expanded Data Memory Access Bit of CPU mode register B (address 000116) to "1". In this case, port P40 ( $\overline{\text{EDMA}}$  pin) goes "L" level during the read/write cycle of the LDA or STA instruction.

The determination of which bank to access is done by using an I/O port to represent expanded addresses exceeding address bus AB15. For example, when accessing 4 banks, use two I/O ports to represent address buses AB16 and AB17.

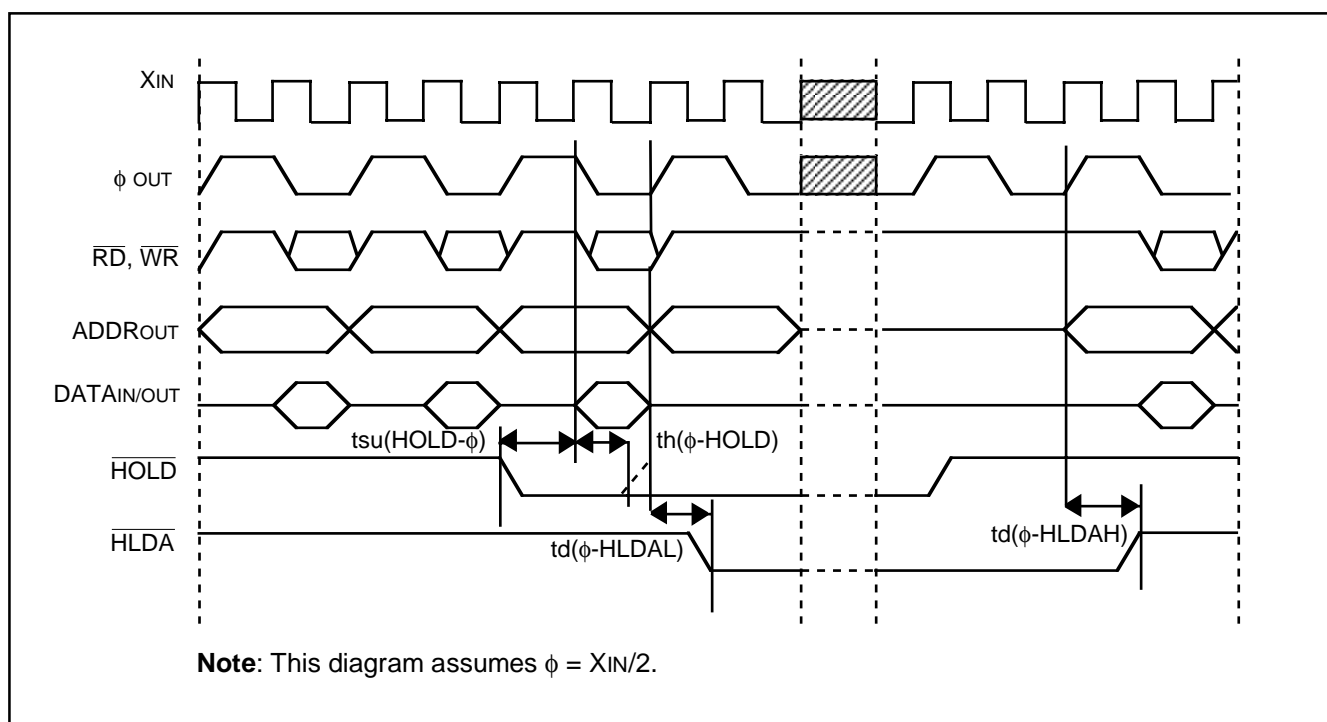


Fig. 67 Hold function timing diagram

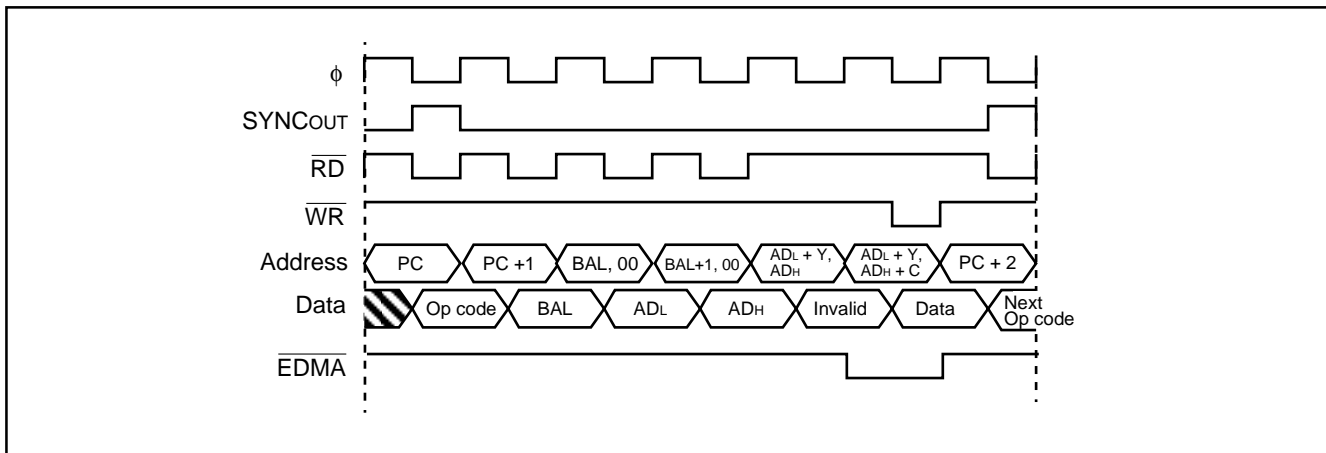


Fig. 68 STA (\$zz), Y instruction sequence when EDMA enabled

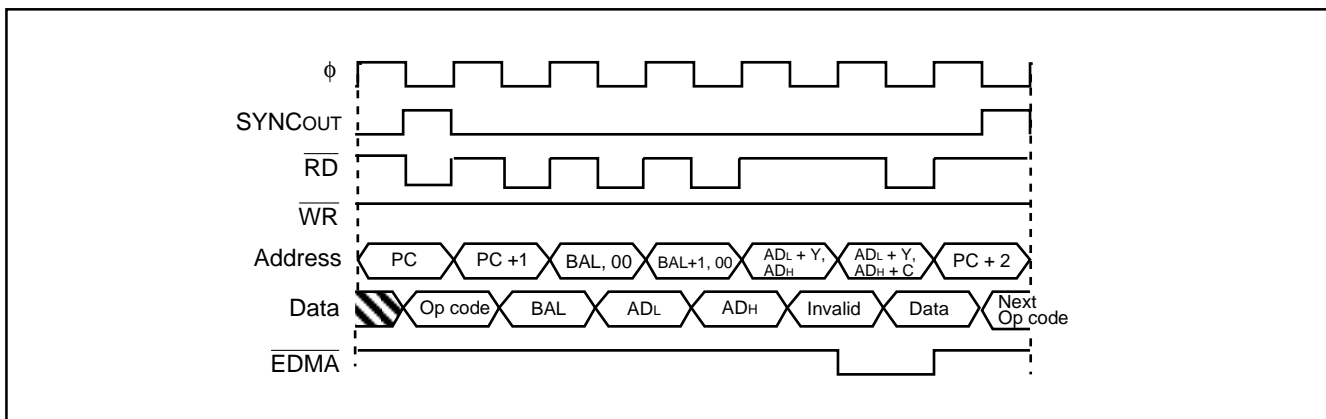


Fig. 69 LDA (\$zz), Y instruction sequence when EDMA enabled and T flag = "0"

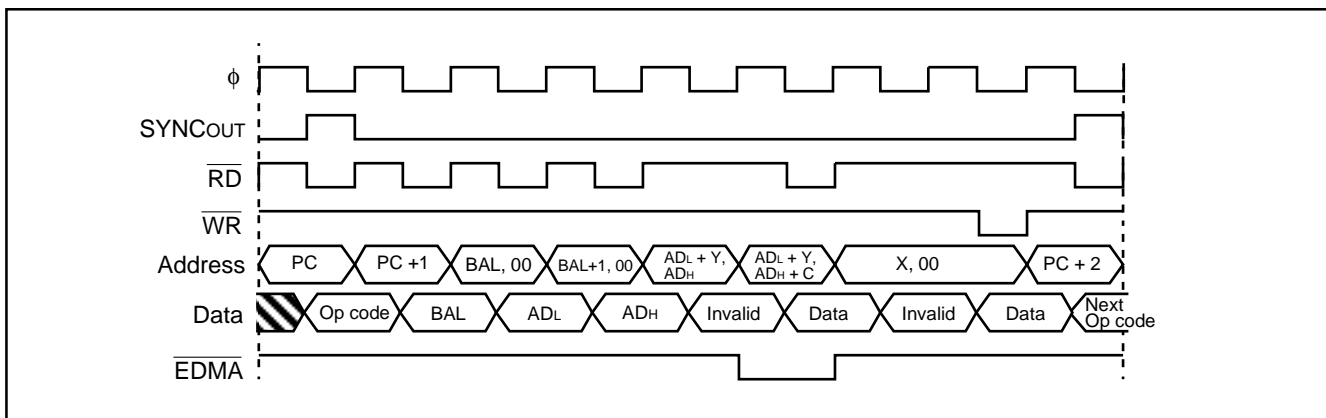


Fig. 70 LDA (\$zz), Y instruction sequence when EDMA enabled and T flag = "1"

## FLASH MEMORY MODE

The M37643F8FP/HP (flash memory version) has an internal new DINOR (Divided bit line NOR) flash memory that can be rewritten with a single power source when V<sub>CC</sub> is 5 V, and 2 power sources when V<sub>PP</sub> is 5 V and V<sub>CC</sub> is 3.3 V in the CPU rewrite and standard serial I/O modes.

For this flash memory, three flash memory modes are available in which to read, program, and erase: the parallel I/O and standard serial I/O modes in which the flash memory can be manipulated using a programmer and the CPU rewrite mode in which the flash memory can be manipulated by the Central Processing Unit (CPU).

## Summary

Table 9 lists the summary of the M37643F8 (flash memory version).

This flash memory version has some blocks on the flash memory as shown in Figure 71 and each block can be erased. The flash memory is divided into User ROM area and Boot ROM area.

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

**Table 9 Summary of M37643F8 (flash memory version)**

Item		Specifications
Power source voltage (For Program/Erase)		V <sub>CC</sub> = 3.00 – 3.60 V, 4.50 – 5.25 V (f(XIN) = 24 MHz, φ = 6 MHz) <b>(Note 1)</b>
V <sub>PP</sub> voltage (For Program/Erase)		V <sub>PP</sub> = 4.50 – 5.25 V
Flash memory mode		3 modes; Flash memory can be manipulated as follows: (1) CPU rewrite mode: Manipulated by the Central Processing Unit (CPU) (2) Parallel I/O mode: Manipulated using an external programmer <b>(Note 2)</b> (3) Standard serial I/O mode: Manipulated using an external programmer <b>(Note 2)</b> .
Erase block division	User ROM area	See Figure 71.
	Boot ROM area	1 block (4 Kbytes) <b>(Note 3)</b>
Program method		Byte program
Erase method		Batch erasing/Block erasing
Program/Erase control method		Program/Erase control by software command
Number of commands		6 commands
Number of program/Erase times		100 times
ROM code protection		Available in parallel I/O mode and standard serial I/O mode

**Notes 1:** After programming/erasing at V<sub>CC</sub> = 3.0 to 3.6 V, the MCU can operate only at V<sub>CC</sub> = 3.0 to 3.6 V.

After programming/erasing at V<sub>CC</sub> = 4.5 to 5.25 V or programming/erasing with the exclusive external equipment flash programmer, the MCU can operate at both V<sub>CC</sub> = 3.0 to 3.6 V and 4.15 to 5.25 V.

**2:** In the parallel I/O mode or the standard serial I/O mode, use the exclusive external equipment flash programmer which supports the 7643 Group (flash memory version).

**3:** The Boot ROM area has had a standard serial I/O mode control program stored in it when shipped from the factory. This Boot ROM area can be rewritten in only parallel I/O mode.



### (1) CPU Rewrite Mode

In CPU rewrite mode, the internal flash memory can be operated on (read, program, or erase) under control of the Central Processing Unit (CPU).

In CPU rewrite mode, only the User ROM area shown in Figure 71 can be rewritten; the Boot ROM area cannot be rewritten. Make sure the program and block erase commands are issued for only the User ROM area and each block area.

The control program for CPU rewrite mode can be stored in either User ROM or Boot ROM area. In the CPU rewrite mode, because the flash memory cannot be read from the CPU, the rewrite control program must be transferred to internal RAM area to be executed before it can be executed.

### Microcomputer Mode and Boot Mode

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

See Figure 71 for details about the Boot ROM area.

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

When the microcomputer is reset by pulling the P36 ( $\overline{CE}$ ) pin high, the P81 (SCLK) pin high, the CNVss pin high, the CPU starts operating using the control program in the Boot ROM area. This mode is called the "Boot" mode.

### Block Address

Block addresses refer to the maximum address of each block. These addresses are used in the block erase command.

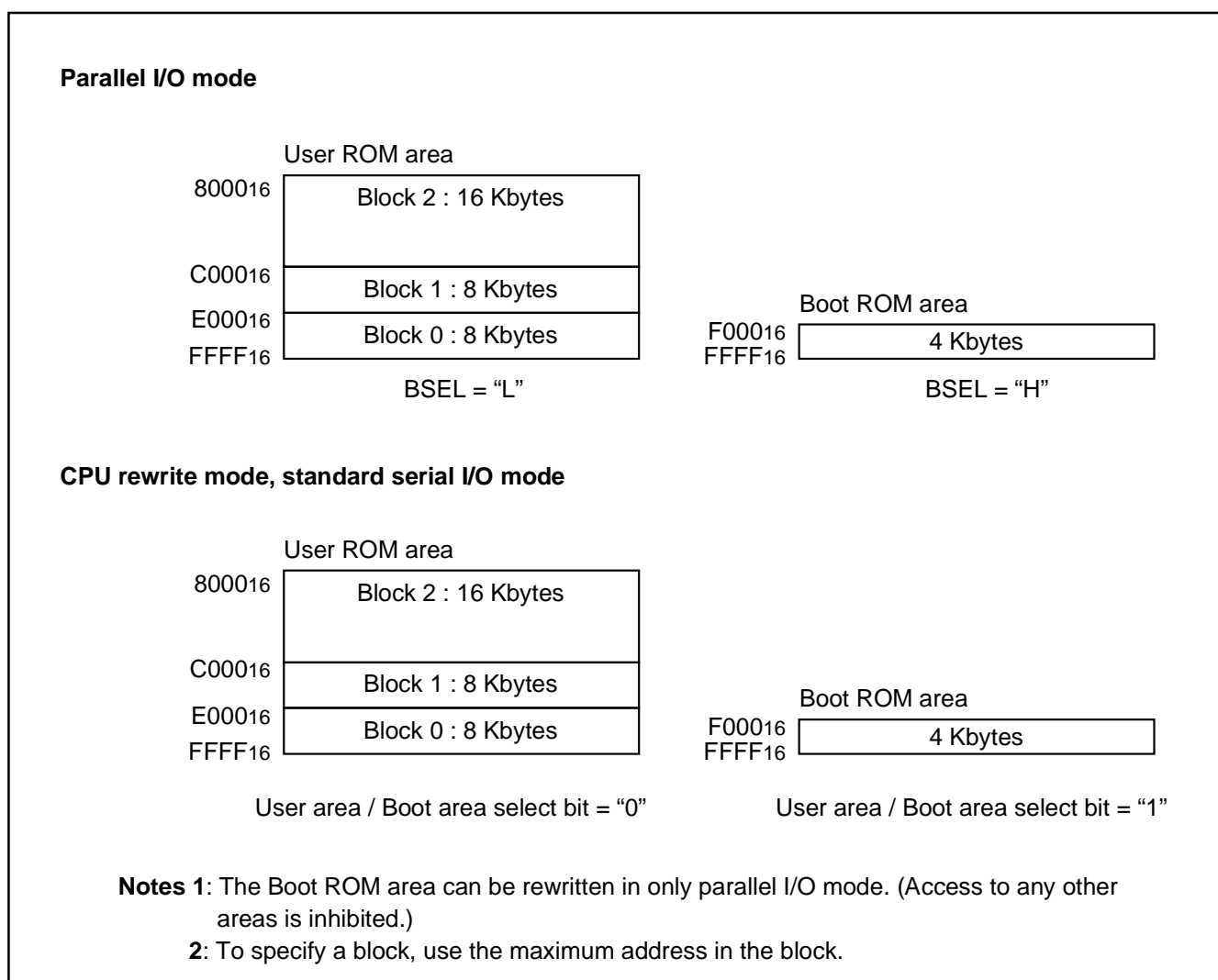


Fig. 71 Block diagram of built-in flash memory

### Outline Performance (CPU Rewrite Mode)

CPU rewrite mode is usable in the single-chip, memory expansion or Boot mode. The only User ROM area can be rewritten in CPU rewrite mode.

In CPU rewrite mode, the CPU erases, programs and reads the internal flash memory by executing software commands. This rewrite control program must be transferred to a memory such as the internal RAM before it can be executed.

The MCU enters CPU rewrite mode by applying 4.50 V to 5.25 V to the CNVss pin and setting "1" to the CPU Rewrite Mode Select Bit (bit 1 of address 006A16). Software commands are accepted once the mode is entered.

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

Figure 72 shows the flash memory control register.

Bit 0 is the RY/ $\overline{\text{BY}}$  status flag used exclusively to read the operating status of the flash memory. During programming and erase operations, it is "0" (busy). Otherwise, it is "1" (ready).

Bit 1 is the CPU Rewrite Mode Select Bit. When this bit is set to "1", the MCU enters CPU rewrite mode. Software commands are accepted once the mode is entered. In CPU rewrite mode, the

CPU becomes unable to access the internal flash memory directly. Therefore, use the control program in a memory other than internal flash memory for write to bit 1. To set this bit to "1", it is necessary to write "0" and then write "1" in succession. The bit can be set to "0" by only writing "0".

Bit 2 is the CPU Rewrite Mode Entry Flag. This flag indicates "1" in CPU rewrite mode, so that reading this flag can check whether CPU rewrite mode has been entered or not.

Bit 3 is the flash memory reset bit used to reset the control circuit of internal flash memory. This bit is used when exiting CPU rewrite mode and when flash memory access has failed. When the CPU Rewrite Mode Select Bit is "1", setting "1" for this bit resets the control circuit. To set this bit to "1", it is necessary to write "0" and then write "1" in succession. To release the reset, it is necessary to set this bit to "0".

Bit 4 is the User Area/Boot Area Select Bit. When this bit is set to "1", Boot ROM area is accessed, and CPU rewrite mode in Boot ROM area is available. In Boot mode, this bit is set to "1" automatically. Reprogramming of this bit must be in a memory other than internal flash memory.

Figure 73 shows a flowchart for setting/releasing CPU rewrite mode.

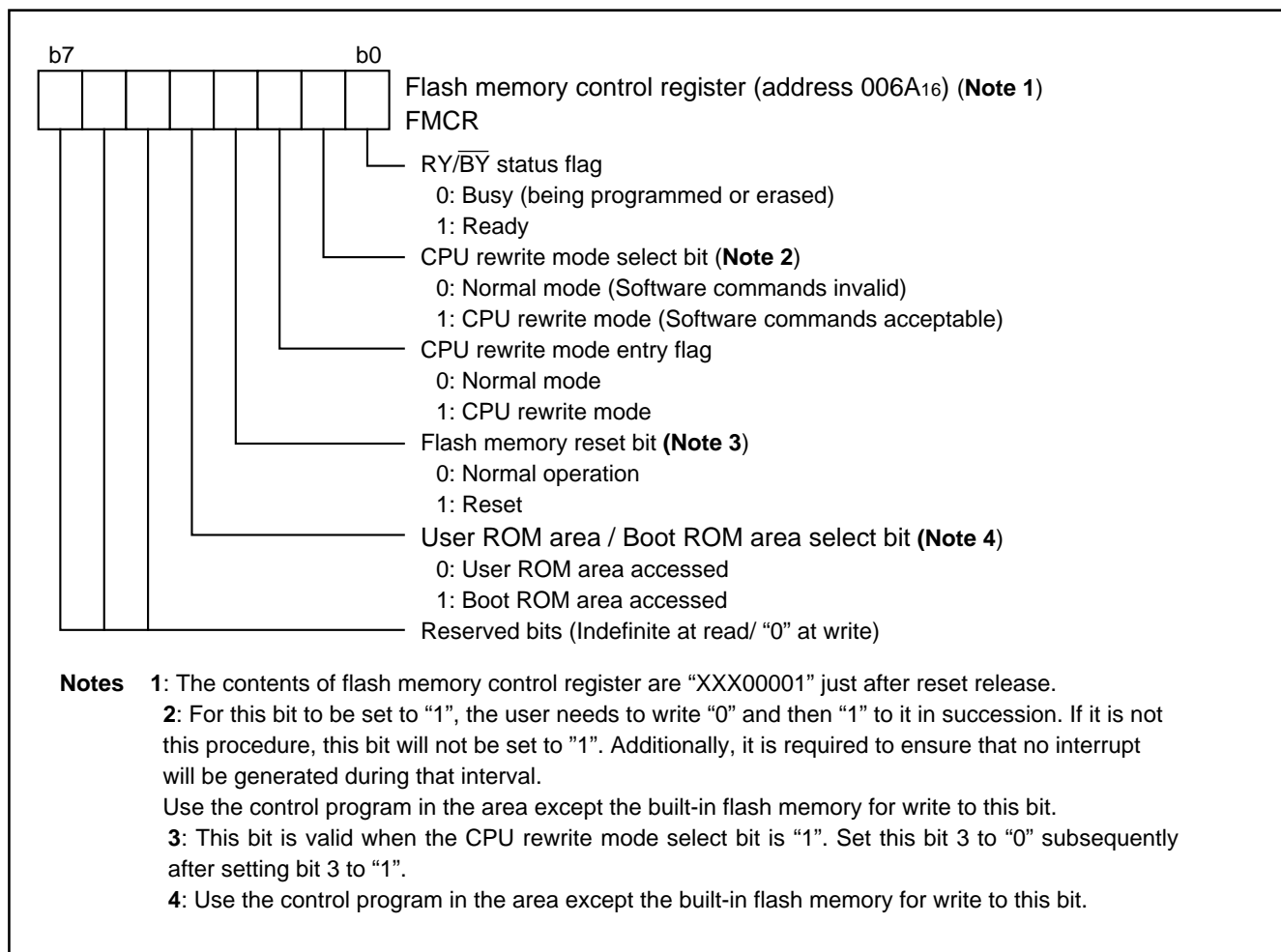


Fig. 72 Structure of flash memory control register

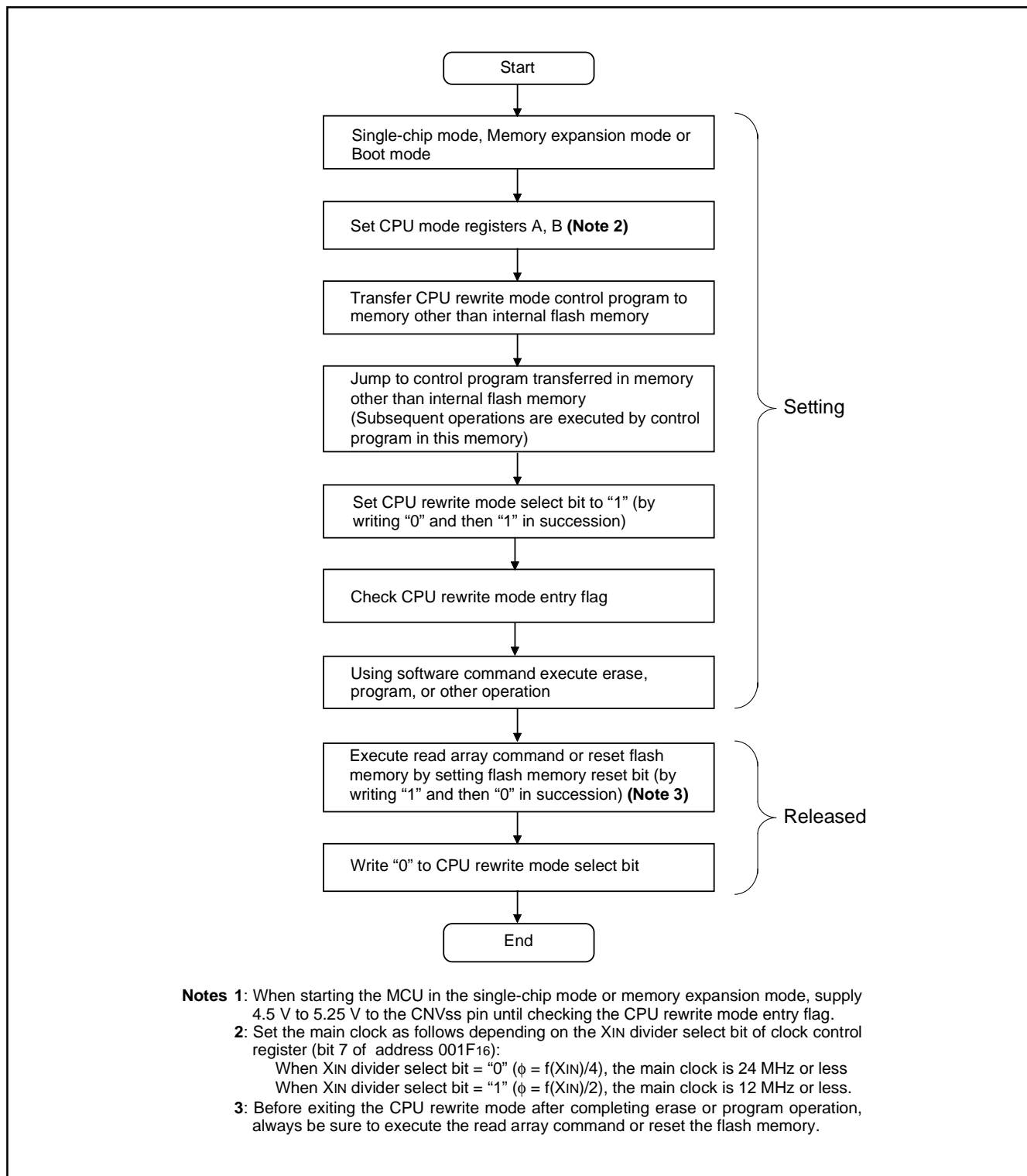


Fig. 73 CPU rewrite mode set/release flowchart

### Notes on CPU Rewrite Mode

The below notes applies when rewriting the flash memory in CPU rewrite mode.

#### ●Operation speed

During CPU rewrite mode, set the internal clock  $\phi$  to 6 MHz or less using the XIN Divider Select Bit (bit 7 of address 001F<sub>16</sub>).

#### ●Instructions inhibited against use

The instructions which refer to the internal data of the flash memory cannot be used during CPU rewrite mode .

#### ●Interrupts inhibited against use

The interrupts cannot be used during CPU rewrite mode because they refer to the internal data of the flash memory.

#### ●Reset

Reset is always valid. When CNVss is "H" at reset release, the program starts from the address stored in addresses FFFA<sub>16</sub> and FFFB<sub>16</sub> of the boot ROM area in order that CPU may start in boot mode.

## Software Commands (CPU Rewrite Mode)

Table 10 lists the software commands.

After setting the CPU Rewrite Mode Select Bit of the flash memory control register to "1", execute a software command to specify an erase or program operation.

Each software command is explained below.

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

The read array mode is entered by writing the command code "FF<sub>16</sub>" in the first bus cycle. When an address to be read is input in one of the bus cycles that follow, the contents of the specified address are read out at the data bus (DB<sub>0</sub> to DB<sub>7</sub>).

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

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

The read status register mode is entered by writing the command code "70<sub>16</sub>" in the first bus cycle. The contents of the status register are read out at the data bus (DB<sub>0</sub> to DB<sub>7</sub>) by a read in the second bus cycle.

The status register is explained in the next section.

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

This command is used to clear the bits SR4 and SR5 of the status register after they have been set. These bits indicate that operation has ended in an error. To use this command, write the command code "50<sub>16</sub>" in the first bus cycle.

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

Program operation starts when the command code "40<sub>16</sub>" is written in the first bus cycle. Then, if the address and data to program are written in the 2nd bus cycle, program operation (data programming and verification) will start.

Whether the write operation is completed can be confirmed by reading the status register or the RY/BY Status Flag of the flash memory control register. When the program starts, the read status

register mode is entered automatically and the contents of the status register is read at the data bus (DB<sub>0</sub> to DB<sub>7</sub>). The status register bit 7 (SR7) is set to "0" at the same time the write operation starts and is returned to "1" upon completion of the write operation. In this case, the read status register mode remains active until the next command is written.

The RY/BY Status Flag is "0" (busy) during write operation and "1" (ready) when the write operation is completed as is the status register bit 7.

At program end, program results can be checked by reading bit 4 (SR4) of the status register.

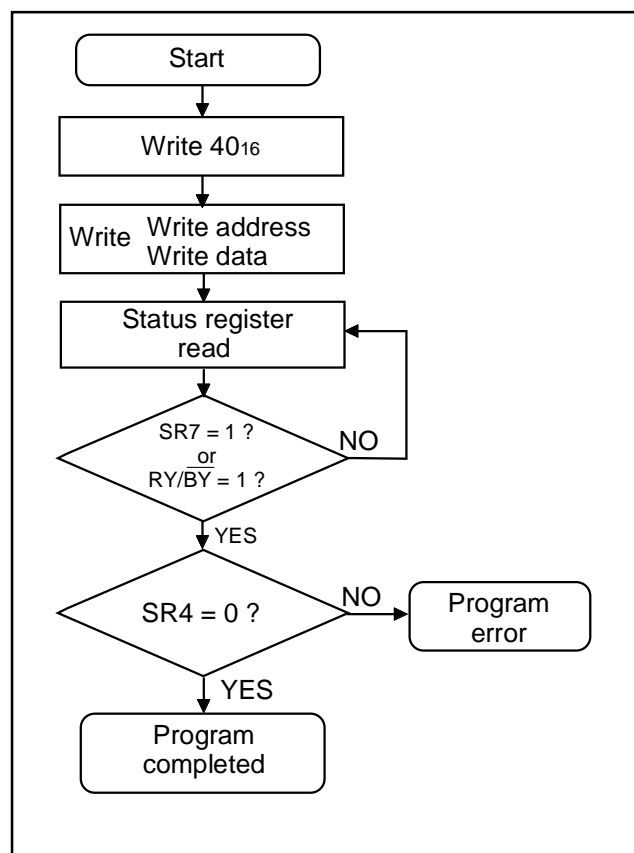


Fig. 74 Program flowchart

Table 10 List of software commands (CPU rewrite mode)

Command	Cycle number	First bus cycle			Second bus cycle		
		Mode	Address	Data (DB <sub>0</sub> to DB <sub>7</sub> )	Mode	Address	Data (DB <sub>0</sub> to DB <sub>7</sub> )
Read array	1	Write	X (Note 4)	FF <sub>16</sub>			
Read status register	2	Write	X	70 <sub>16</sub>	Read	X	SRD (Note 1)
Clear status register	1	Write	X	50 <sub>16</sub>			
Program	2	Write	X	40 <sub>16</sub>	Write	WA (Note 2)	WD (Note 2)
Erase all blocks	2	Write	X	20 <sub>16</sub>	Write	X	20 <sub>16</sub>
Block erase	2	Write	X	20 <sub>16</sub>	Write	BA (Note 3)	D0 <sub>16</sub>

Notes 1: SRD = Status Register Data

2: WA = Write Address, WD = Write Data

3: BA = Block Address to be erased (Input the maximum address of each block.)

4: X denotes a given address in the User ROM area .

### ●Erase All Blocks Command (20<sub>16</sub>/20<sub>16</sub>)

By writing the command code "20<sub>16</sub>" in the first bus cycle and the confirmation command code "20<sub>16</sub>" in the second bus cycle that follows, the operation of erase all blocks (erase and erase verify) starts.

Whether the erase all blocks command is terminated can be confirmed by reading the status register or the RY/ $\overline{\text{BY}}$  Status Flag of flash memory control register. When the erase all blocks operation starts, the read status register mode is entered automatically and the contents of the status register can be read out at the data bus (DB<sub>0</sub> to DB<sub>7</sub>). The status register bit 7 (SR7) is set to "0" at the same time the erase operation starts and is returned to "1" upon completion of the erase operation. In this case, the read status register mode remains active until another command is written.

The RY/ $\overline{\text{BY}}$  Status Flag is "0" during erase operation and "1" when the erase operation is completed as is the status register bit 7 (SR7).

After the erase all blocks end, erase results can be checked by reading bit 5 (SRS) of the status register. For details, refer to the section where the status register is detailed.

### ●Block Erase Command (20<sub>16</sub>/D0<sub>16</sub>)

By writing the command code "20<sub>16</sub>" in the first bus cycle and the confirmation command code "D0<sub>16</sub>" and the block address in the second bus cycle that follows, the block erase (erase and erase verify) operation starts for the block address of the flash memory to be specified.

Whether the block erase operation is completed can be confirmed by reading the status register or the RY/ $\overline{\text{BY}}$  Status Flag of flash memory control register. At the same time the block erase operation starts, the read status register mode is automatically entered, so that the contents of the status register can be read out. The status register bit 7 (SR7) is set to "0" at the same time the block erase operation starts and is returned to "1" upon completion of the block erase operation. In this case, the read status register mode remains active until the read array command (FF<sub>16</sub>) is written.

The RY/ $\overline{\text{BY}}$  Status Flag is "0" during block erase operation and "1" when the block erase operation is completed as is the status register bit 7.

After the block erase ends, erase results can be checked by reading bit 5 (SRS) of the status register. For details, refer to the section where the status register is detailed.

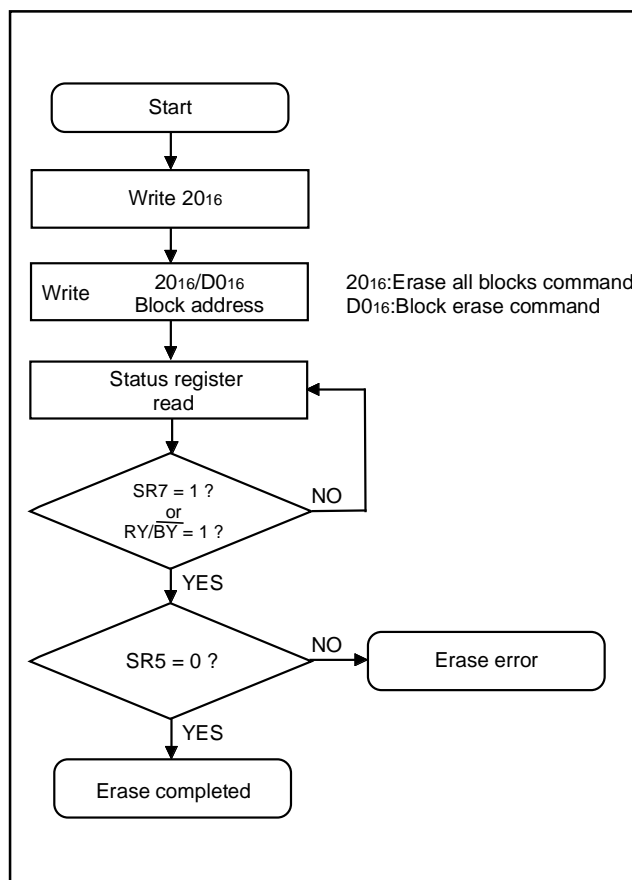


Fig. 75 Erase flowchart

## Status Register (SRD)

The status register shows the operating status of the flash memory and whether erase operations and programs ended successfully or in error. It can be read in the following ways:

- (1) By reading an arbitrary address from the User ROM area after writing the read status register command (70<sub>16</sub>)
- (2) By reading an arbitrary address from the User ROM area in the period from when the program starts or erase operation starts to when the read array command (FF<sub>16</sub>) is input.

Also, the status register can be cleared by writing the clear status register command (50<sub>16</sub>).

After reset, the status register is set to "80<sub>16</sub>".

Table 11 shows the status register. Each bit in this register is explained below.

### •Sequencer status (SR7)

The sequencer status indicates the operating status of the flash memory. This bit is set to "0" (busy) during write or erase operation and is set to "1" when these operations ends.

After power-on, the sequencer status is set to "1" (ready).

### •Erase status (SR5)

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

### •Program status (SR4)

The program status indicates the operating status of write operation. When a write error occurs, it is set to "1".

The program status is set to "0" when it is cleared.

If "1" is written for any of the SR5 and SR4 bits, the program, erase all blocks, and block erase commands are not accepted. Before executing these commands, execute the clear status register command (50<sub>16</sub>) and clear the status register.

Also, if any commands are not correct, both SR5 and SR4 are set to "1".

**Table 11 Definition of each bit in status register (SRD)**

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

## Full Status Check

By performing full status check, it is possible to know the execution results of erase and program operations. Figure 76 shows a

full status check flowchart and the action to be taken when each error occurs.

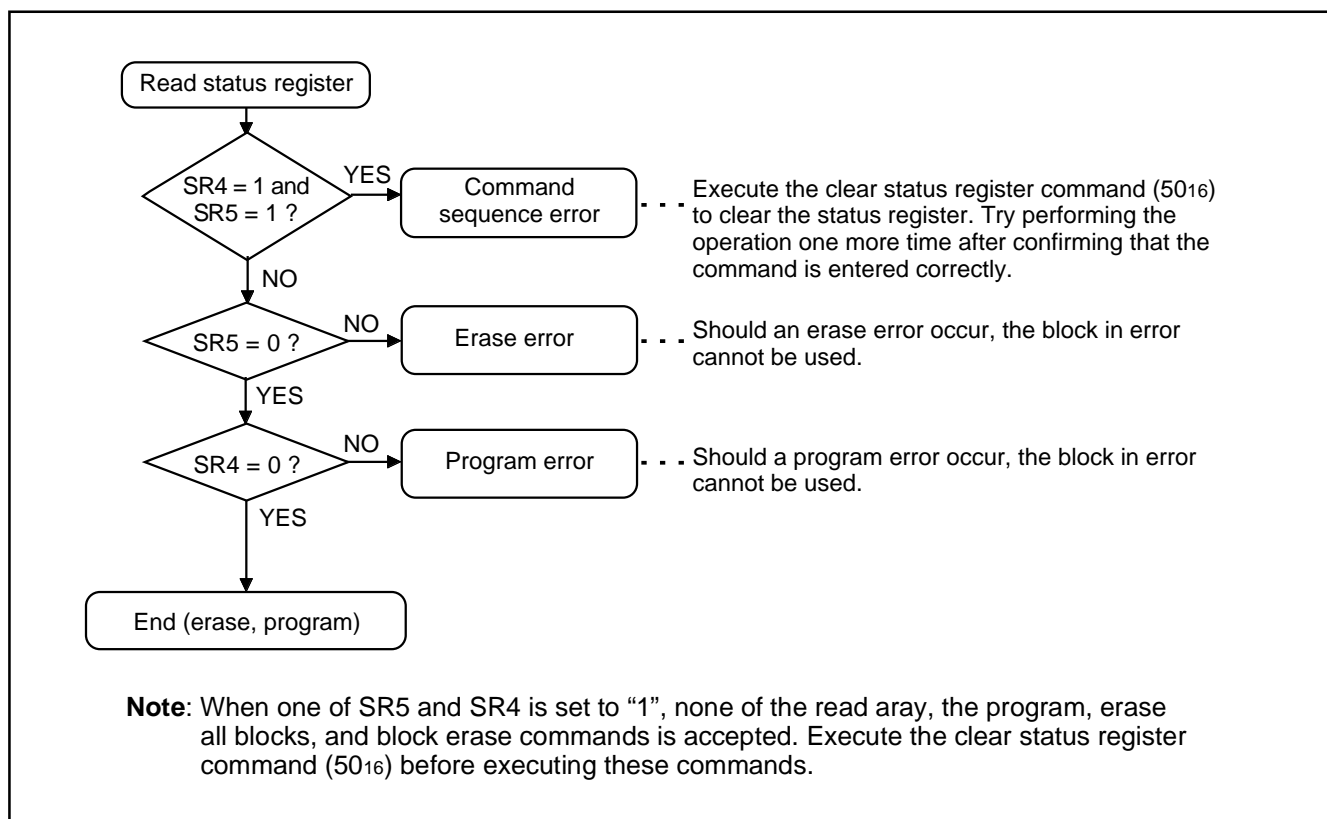


Fig. 76 Full status check flowchart and remedial procedure for errors



## Functions To Inhibit Rewriting Flash Memory Version

To prevent the contents of internal flash memory from being read out or rewritten easily, this MCU incorporates a ROM code protect function for use in parallel I/O mode and an ID code check function for use in standard serial I/O mode.

### ●ROM Code Protect Function (in Pararell I/O Mode)

The ROM code protect function is the function to inhibit reading out or modifying the contents of internal flash memory by using the ROM code protect control (address FFC916) in parallel I/O mode. Figure 77 shows the ROM code protect control (address FFC916). (This address exists in the User ROM area.)

If one or both of the pair of ROM Code Protect Bits is set to "0",

the ROM code protect is turned on, so that the contents of internal flash memory are protected against readout and modification. The ROM code protect is implemented in two levels. If level 2 is selected, the flash memory is protected even against readout by a shipment inspection LSI tester, etc. When an attempt is made to select both level 1 and level 2, level 2 is selected by default.

If both of the two ROM Code Protect Reset Bits are set to "00", the ROM code protect is turned off, so that the contents of internal flash memory can be read out or modified. Once the ROM code protect is turned on, the contents of the ROM Code Protect Reset Bits cannot be modified in parallel I/O mode. Use the serial I/O or CPU rewrite mode to rewrite the contents of the ROM Code Protect Reset Bits.

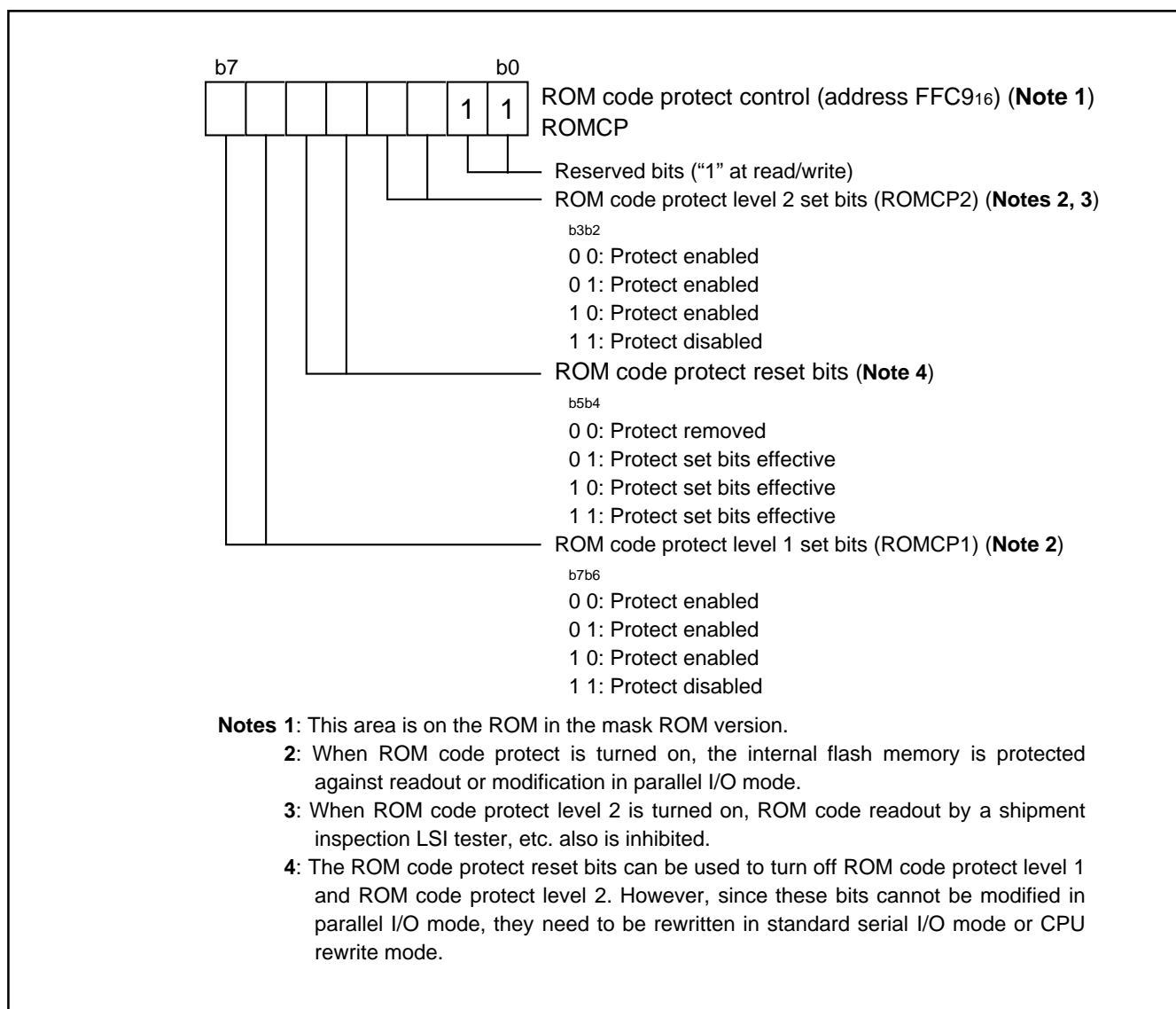


Fig. 77 Structure of ROM code protect control

### ID Code Check Function (in Standard serial I/O mode)

Use this function in standard serial I/O mode. When the contents of the flash memory are not blank, the ID code sent from the programmer is compared with the ID code written in the flash memory to see if they match. If the ID codes do not match, the commands sent from the programmer are not accepted. The ID code consists of 8-bit data, and its areas are FFC2<sub>16</sub> to FFC8<sub>16</sub>. Write a program which has had the ID code preset at these addresses to the flash memory.

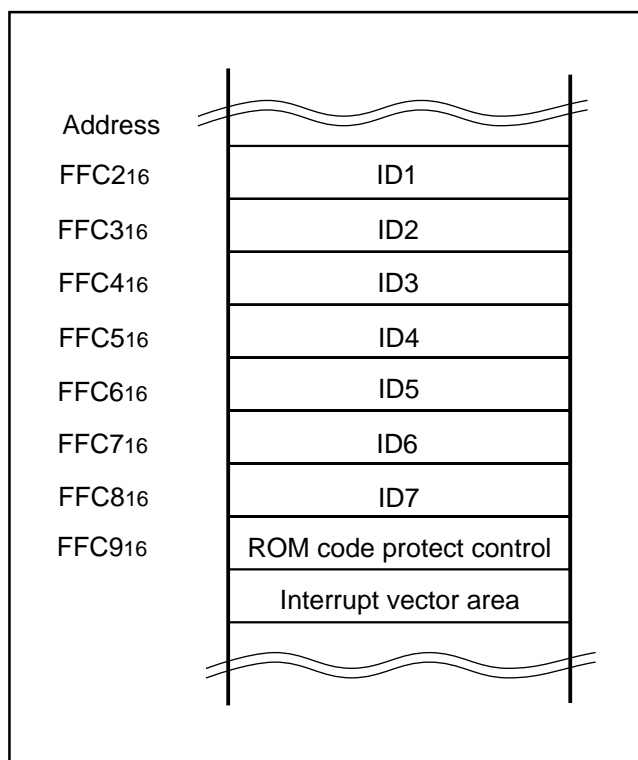


Fig. 78 ID code store addresses

## (2) Parallel I/O Mode

Parallel I/O mode is the mode which parallel output and input software command, address, and data required for the operations (read, program, erase, etc.) to a built-in flash memory. Use the exclusive external equipment flash programmer which supports the 7643 Group (flash memory version). Refer to each programmer maker's handling manual for the details of the usage.

### User ROM and Boot ROM Areas

In parallel I/O mode, the user ROM and boot ROM areas shown in Figure 71 can be rewritten. Both areas of flash memory can be operated on in the same way.

Program and block erase operations can be performed in the user ROM area. The user ROM area and its block is shown in Figure 71.

The boot ROM area is 4 Kbytes in size. It is located at addresses F000<sub>16</sub> through FFFF<sub>16</sub>. Make sure program and block erase operations are always performed within this address range. (Access to any location outside this address range is prohibited.)

In the Boot ROM area, an erase block operation is applied to only one 4 Kbyte block. The boot ROM area has had a standard serial I/O mode control program stored in it when shipped from the Renesas factory. Therefore, using the device in standard serial I/O mode, you do not need to write to the boot ROM area.

### (3) Standard serial I/O Mode

The standard serial I/O mode inputs and outputs the software commands, addresses and data needed to operate (read, program, erase, etc.) the internal flash memory. This I/O is clock synchronized serial. This mode requires the exclusive external equipment (flash programmer).

The standard serial I/O mode is different from the parallel I/O mode in that the CPU controls flash memory rewrite (uses the CPU rewrite mode), rewrite data input and so forth. The standard serial I/O mode is started by connecting "H" to the P3<sub>6</sub> ( $\overline{CE}$ ) pin and "H" to the P8<sub>1</sub> (SCLK) pin and "H" to the CNV<sub>ss</sub> pin (apply 4.5 V to 5.25 V to V<sub>pp</sub> from an external source), and releasing the reset operation. (In the ordinary microcomputer mode, set CNV<sub>ss</sub> pin to "L" level.)

This control program is written in the Boot ROM area when the product is shipped from Renesas. Accordingly, make note of the fact that the standard serial I/O mode cannot be used if the Boot ROM area is rewritten in parallel I/O mode. Figures 79 and 80 show the pin connections for the standard serial I/O mode.

In standard serial I/O mode, serial data I/O uses the four serial I/O pins SCLK, SRXD, STXD and  $\overline{SRDY}$  (BUSY). The SCLK pin is the transfer clock input pin through which an external transfer clock is input. The STXD pin is for CMOS output. The SRDY (BUSY) pin outputs "L" level when ready for reception and "H" level when reception starts.

Serial data I/O is transferred serially in 8-bit units.

In standard serial I/O mode, only the User ROM area shown in Figure 71 can be rewritten. The Boot ROM area cannot be rewritten.

In standard serial I/O mode, a 7-byte ID code is used. When there is data in the flash memory, commands sent from the peripheral

unit (programmer) are not accepted unless the ID code matches.

### Outline Performance (Standard Serial I/O Mode)

In standard serial I/O mode, software commands, addresses and data are input and output between the MCU and peripheral units (flash programmer, etc.) using 4-wire clock-synchronized serial I/O.

In reception, software commands, addresses and program data are synchronized with the rise of the transfer clock that is input to the SCLK pin, and are then input to the MCU via the SRXD pin. In transmission, the read data and status are synchronized with the fall of the transfer clock, and output from the STXD pin.

The STXD pin is for CMOS output. Transfer is in 8-bit units with LSB first.

When busy, such as during transmission, reception, erasing or program execution, the SRDY (BUSY) pin is "H" level. Accordingly, always start the next transfer after the SRDY (BUSY) pin is "L" level.

Also, data and status registers in a memory can be read after inputting software commands. Status, such as the operating state of the flash memory or whether a program or erase operation ended successfully or not, can be checked by reading the status register. Here following explains software commands, status registers, etc.

Table 12 Description of pin function (Standard Serial I/O Mode)

Pin name	Signal name	I/O	Function
Vcc,Vss	Power supply input		Apply 4.50 V – 5.25 V for 5 V version or 3.00 V – 3.60 V for 3 V version to the Vcc pin. Apply 0 V to the Vss pin.
CNVss	CNVss	I	This controls the MCU operating mode. Connect this pin to VPP (= 4.50 V – 5.25 V)
$\overline{\text{RESET}}$	Reset input	I	To reset, input “L” level for 20 cycles or longer clocks of $\phi$ .
XIN	Clock input		Connect a ceramic or crystal resonator between the XIN and XOUT pins. When inputting an externally derived clock, input it from XIN and leave XOUT open.
XOUT	Clock output		
AVcc, AVss	Analog power supply input		Apply 4.50 V – 5.25 V for 5 V version or 3.00 V – 3.60 V for 3 V version to the AVcc pin. Apply 0 V to the AVss pin.
LPF	LPF	O	Loop filter for the frequency synthesizer. When this pin is not used, leave this open.
Ext.Cap	3.3 V line power supply input	I	Power supply input pin for 3.3 V USB line driver. When this pin is not used, input “H” level.
USB D+	USB D+	I/O	USB D+ signal port. When this pin is not used, input “H” level.
USB D-	USB D-	I/O	USB D- signal port. When this pin is not used, input “L” level.
P00 to P07	I/O port P0	I/O	When these ports are not used, input “L” or “H” level, or leave them open in output mode.
P10 to P17	I/O port P1	I/O	
P20 to P27	I/O port P2	I/O	
P30 to P35, P37	I/O port P3	I/O	
P36	$\overline{\text{CE}}$ input	I	Input “H” level.
P40 to P44	I/O port P4	I/O	When these ports are not used, input “L” or “H” level, or leave them open in output mode.
P50 to P57	I/O port P5	I/O	
P60 to P67	I/O port P6	I/O	
P70 to P74	I/O port P7	I/O	
P80	BUSY output	O	This is a BUSY output pin.
P81	SCLK input	I	This is a serial clock input pin.
P82	SRXD input	I	This is a serial data input pin.
P83	STXD output	O	This is a serial data output pin.
P84 to P87	I/O port P8	I/O	When these ports are not used, input “L” or “H” level, or leave them open in output mode.

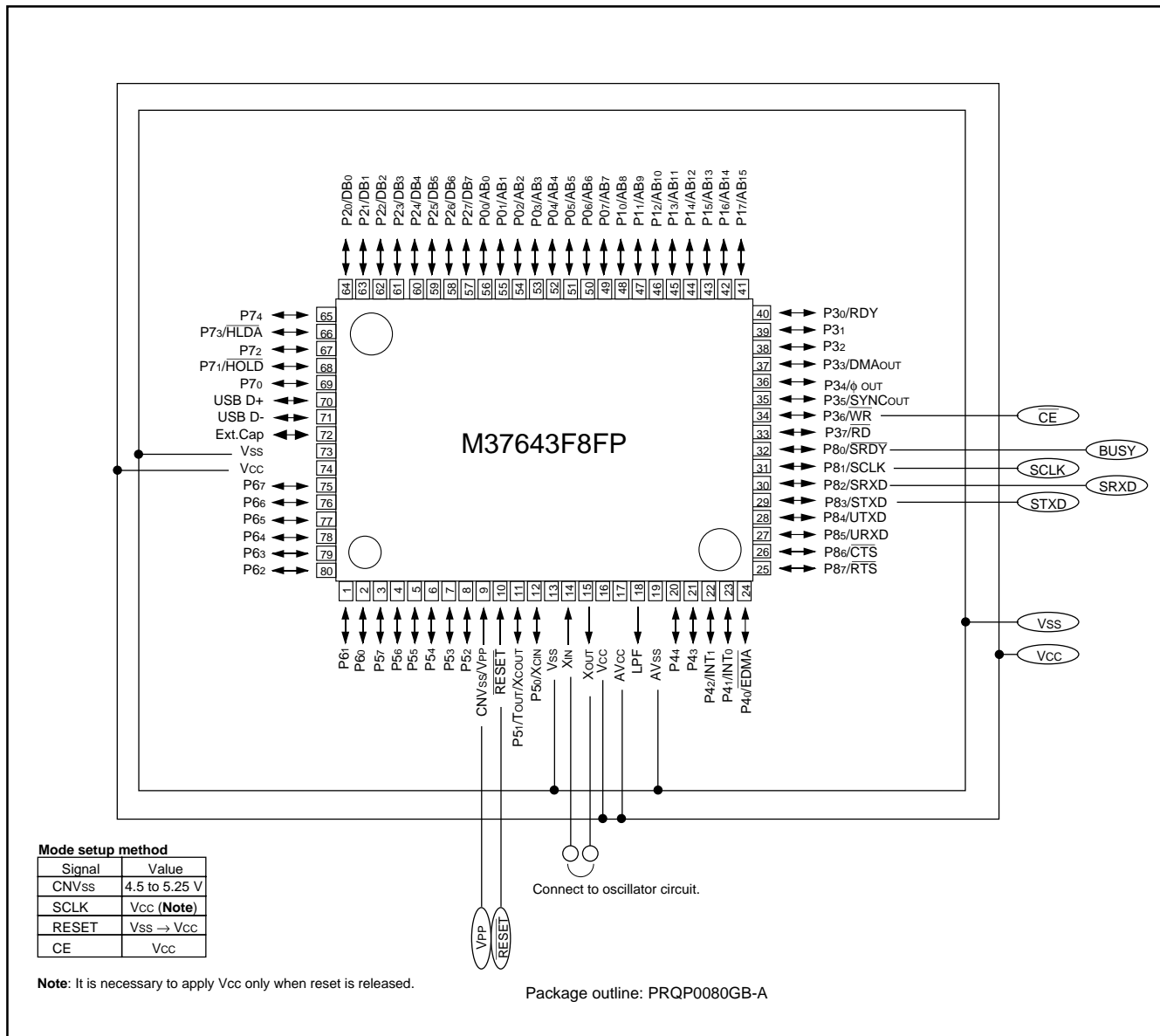


Fig. 79 Pin connection diagram in standard serial I/O mode (1)

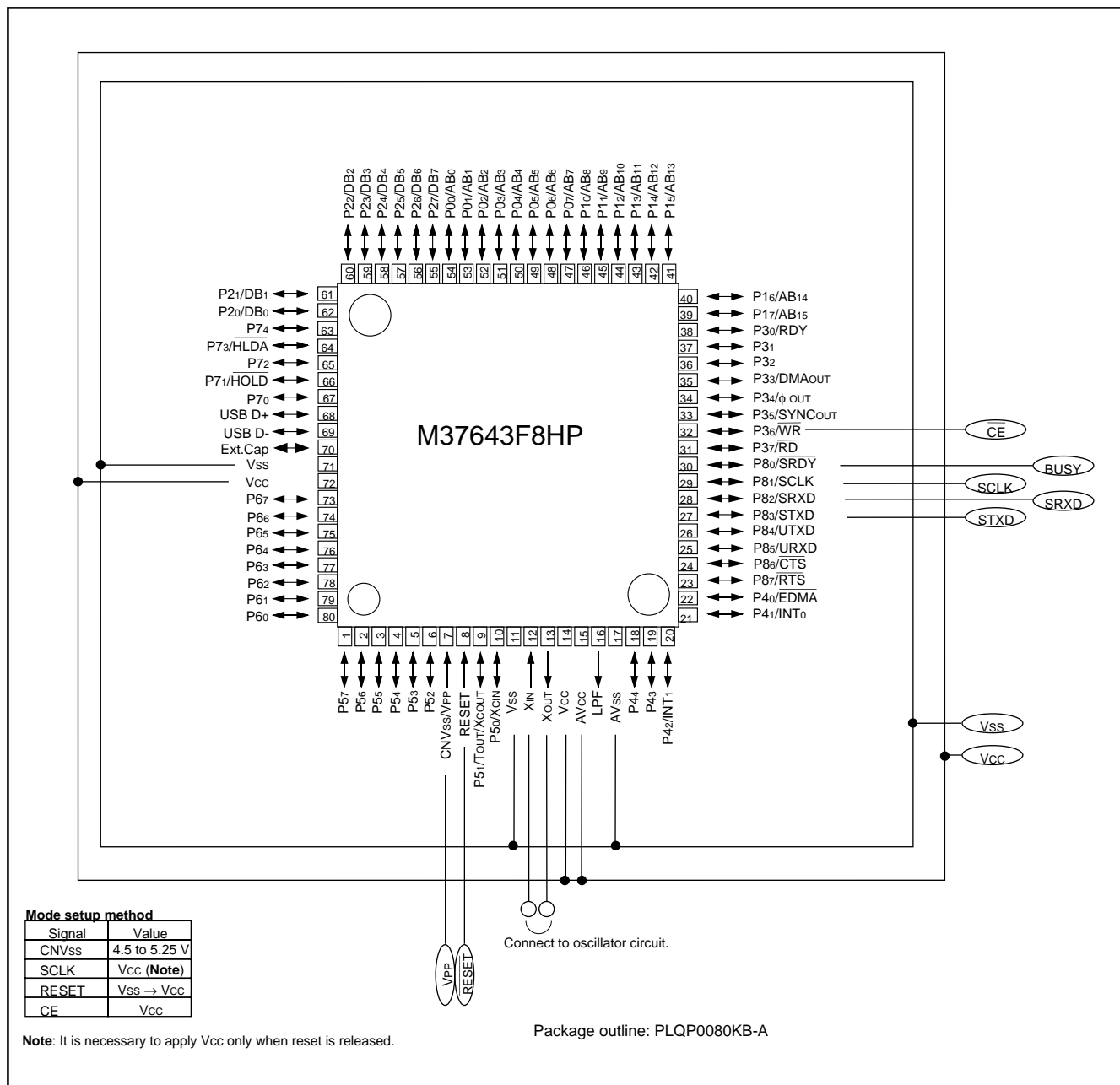


Fig. 80 Pin connection diagram in standard serial I/O mode (2)

## Software Commands (Standard Serial I/O Mode)

Table 13 lists software commands. In standard serial I/O mode, erase, program and read are controlled by transferring software

commands via the SRXD pin. Software commands are explained here below.

**Table 13 Software commands (Standard serial I/O mode)**

Control command	1st byte transfer	2nd byte	3rd byte	4th byte	5th byte	6th byte	.....	When ID is not verified
1 Page read	FF <sub>16</sub>	Address (middle)	Address (high)	Data output	Data output	Data output	Data output to 259th byte	Not acceptable
2 Page program	41 <sub>16</sub>	Address (middle)	Address (high)	Data input	Data input	Data input	Data input to 259th byte	Not acceptable
3 Block erase	20 <sub>16</sub>	Address (middle)	Address (high)	D0 <sub>16</sub>				Not acceptable
4 Erase all blocks	A7 <sub>16</sub>	D0 <sub>16</sub>						Not acceptable
5 Read status register	70 <sub>16</sub>	SRD output	SRD1 output					Acceptable
6 Clear status register	50 <sub>16</sub>							Not acceptable
7 ID code check	F5 <sub>16</sub>	Address (low)	Address (middle)	Address (high)	ID size	ID1	To ID7	Acceptable
8 Download function	FA <sub>16</sub>	Size (low)	Size (high)	Check-sum	Data input	To required number of times		Not acceptable
9 Version data output function	FB <sub>16</sub>	Version data output	Version data output	Version data output	Version data output	Version data output	Version data output to 9th byte	Acceptable
10 Boot ROM area output function	FC <sub>16</sub>	Address (middle)	Address (high)	Data output	Data output	Data output	Data output to 259th byte	Not acceptable

**Notes1:** Shading indicates transfer from the internal flash memory microcomputer to a programmer. All other data is transferred from an external equipment (programmer) to the internal flash memory microcomputer.

**2:** SRD refers to status register data. SRD1 refers to status register 1 data.

**3:** All commands can be accepted for the products of which boot ROM area is totally blank.

**4:** Address low is AB<sub>0</sub> to AB<sub>7</sub>; Address middle is AB<sub>8</sub> to AB<sub>15</sub>; Address high is AB<sub>16</sub> to AB<sub>23</sub>.



●Page Read Command

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

- (1) Transfer the "FF16" command code with the 1st byte.
- (2) Transfer addresses AB8 to AB15 and AB16 to AB23 with the 2nd and 3rd bytes respectively.
- (3) From the 4th byte onward, data (DB0 to DB7) for the page (256 bytes) specified with addresses AB8 to AB23 will be output sequentially from the smallest address first synchronized with the fall of the clock.

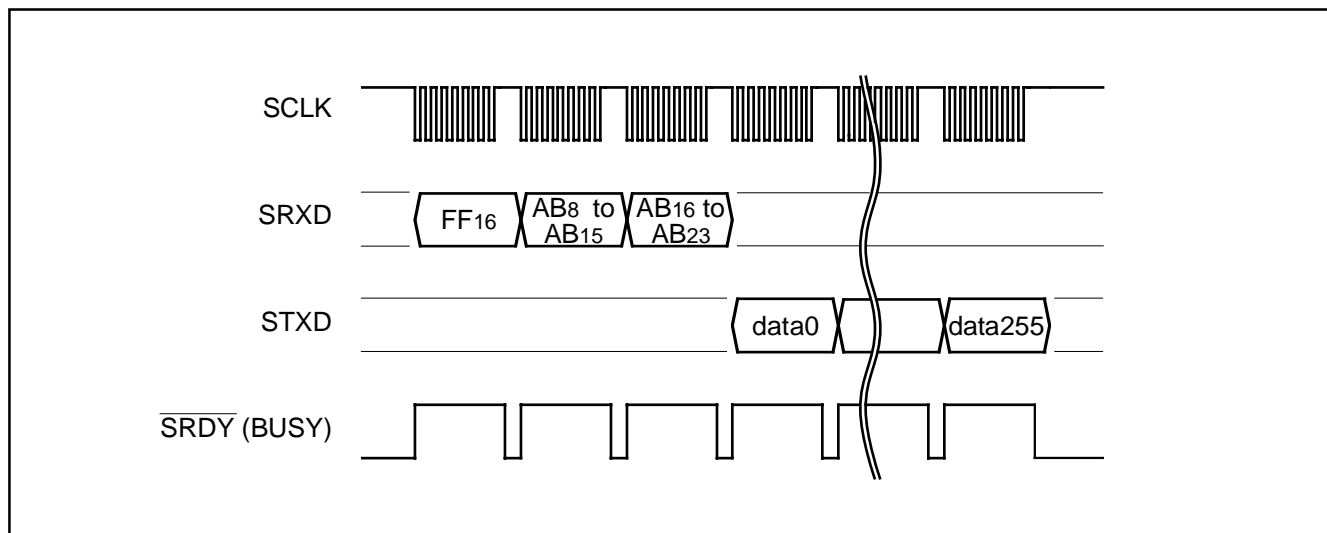


Fig. 81 Timing for page read

●Read Status Register Command

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

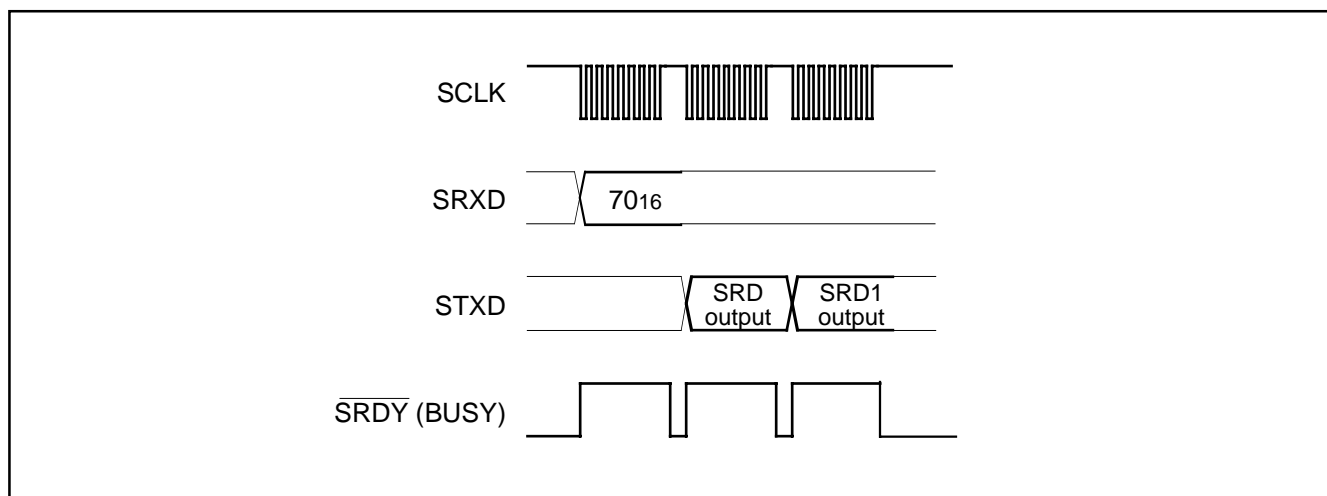


Fig. 82 Timing for reading status register

### ●Clear Status Register Command

This command clears the bits (SR3 to SR5) which are set when the status register operation ends in error. When the "5016" command code is sent with the 1st byte, the aforementioned bits are cleared. When the clear status register operation ends, the  $\overline{\text{SRDY}}$  (BUSY) signal changes from "H" to "L" level.

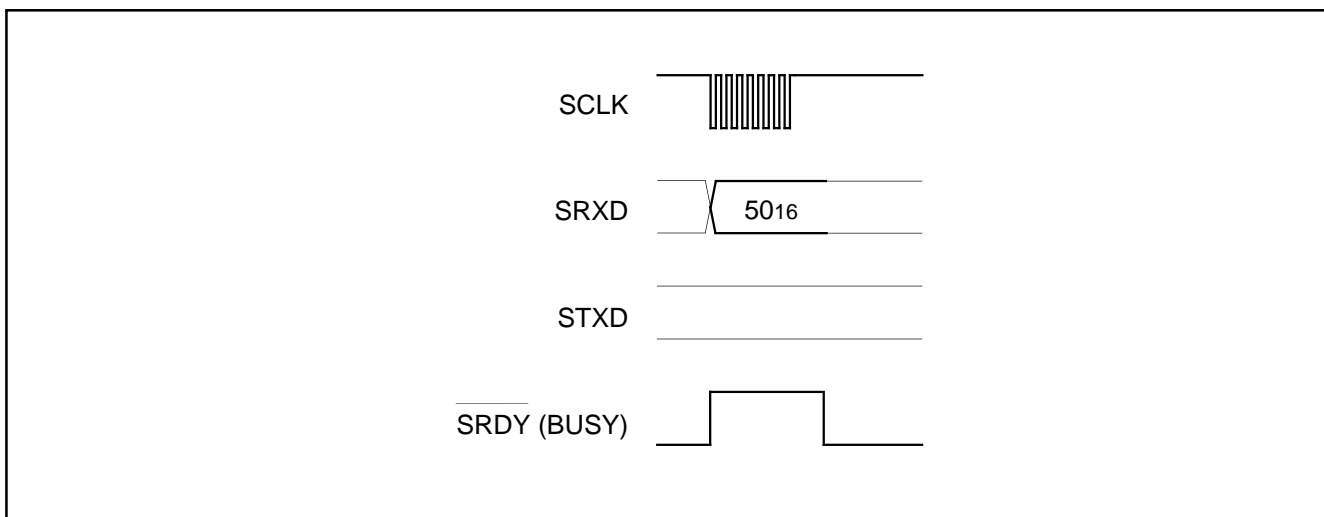


Fig. 83 Timing for clear status register

### ●Page Program Command

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

- (1) Transfer the "4116" command code with the 1st byte.
- (2) Transfer addresses AB8 to AB15 and AB16 to AB23 with the 2nd and 3rd bytes respectively.

- (3) From the 4th byte onward, as write data (DB0 to DB7) for the page (256 bytes) specified with addresses A8 to A23 is input sequentially from the smallest address first, that page is automatically written.

When reception setup for the next 256 bytes ends, the  $\overline{\text{SRDY}}$  (BUSY) signal changes from "H" to "L" level. The result of the page program can be known by reading the status register. For more information, see the section on the status register.

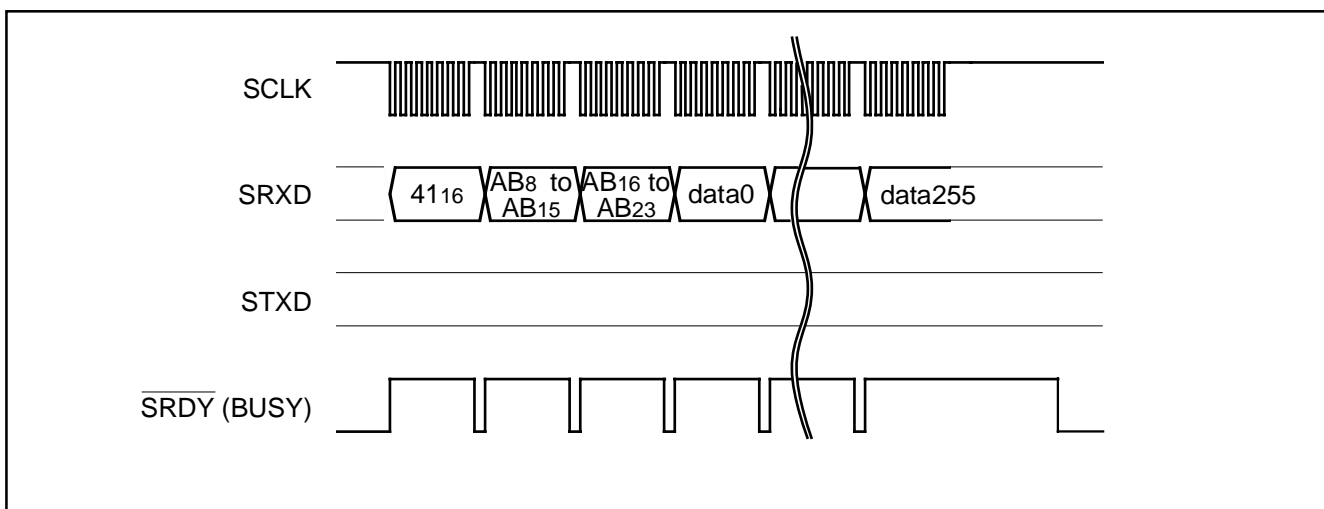


Fig. 84 Timing for page program

### ●Block Erase Command

This command erases the contents of the specified block. Execute the block erase command as explained here following.

- (1) Transfer the "20<sub>16</sub>" command code with the 1st byte.
- (2) Transfer addresses AB<sub>8</sub> to AB<sub>15</sub> and AB<sub>16</sub> to AB<sub>23</sub> with the 2nd and 3rd bytes respectively.
- (3) Transfer the verify command code "D0<sub>16</sub>" with the 4th byte.  
With the verify command code, the erase operation will start for the specified block in the flash memory. Set the addresses AB<sub>8</sub> to AB<sub>23</sub> to the maximum address of the specified block.

When block erasing ends, the  $\overline{\text{SRDY}}$  (BUSY) signal changes from "H" to "L" level. The result of the erase operation can be known by reading the status register.

For more information, see the section on the status register.

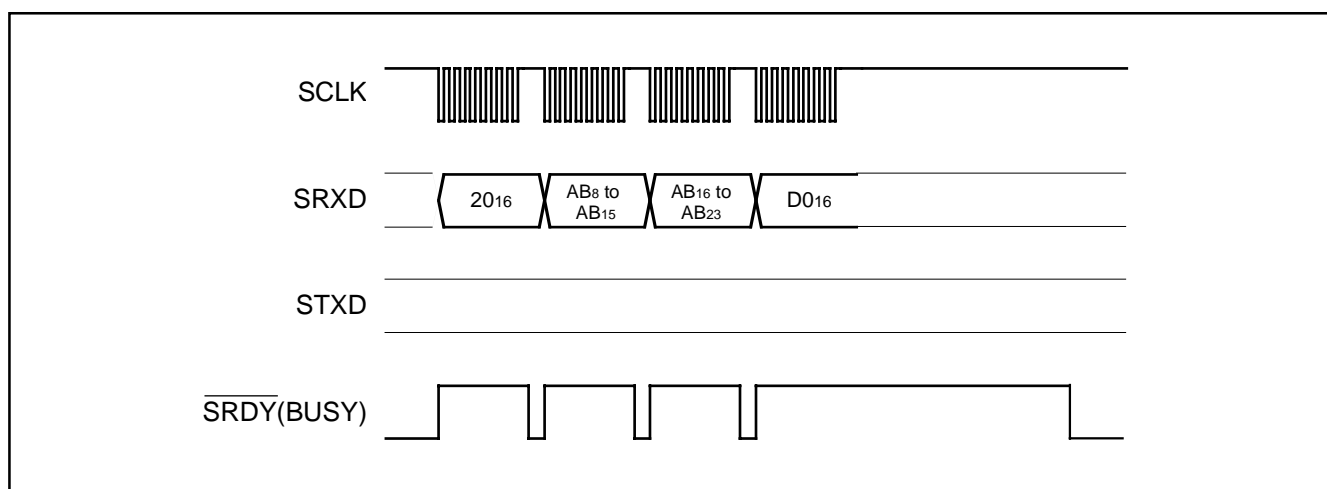


Fig. 85 Timing for block erasing

### ●Erase All Blocks Command

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

- (1) Transfer the "A7<sub>16</sub>" command code with the 1st byte.
- (2) Transfer the verify command code "D0<sub>16</sub>" with the 2nd byte.  
With the verify command code, the erase operation will start and continue for all blocks in the flash memory.

When erase all blocks end, the  $\overline{\text{SRDY}}$  (BUSY) signal changes from "H" to "L" level. The result of the erase operation can be known by reading the status register.

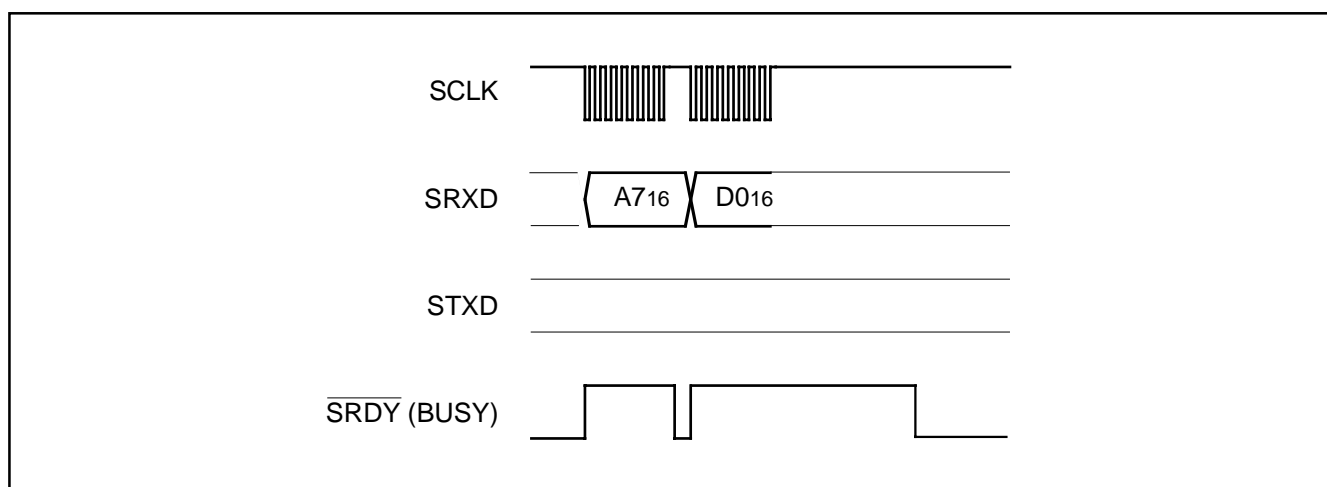


Fig. 86 Timing for erase all blocks

### ●Download Command

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

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

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

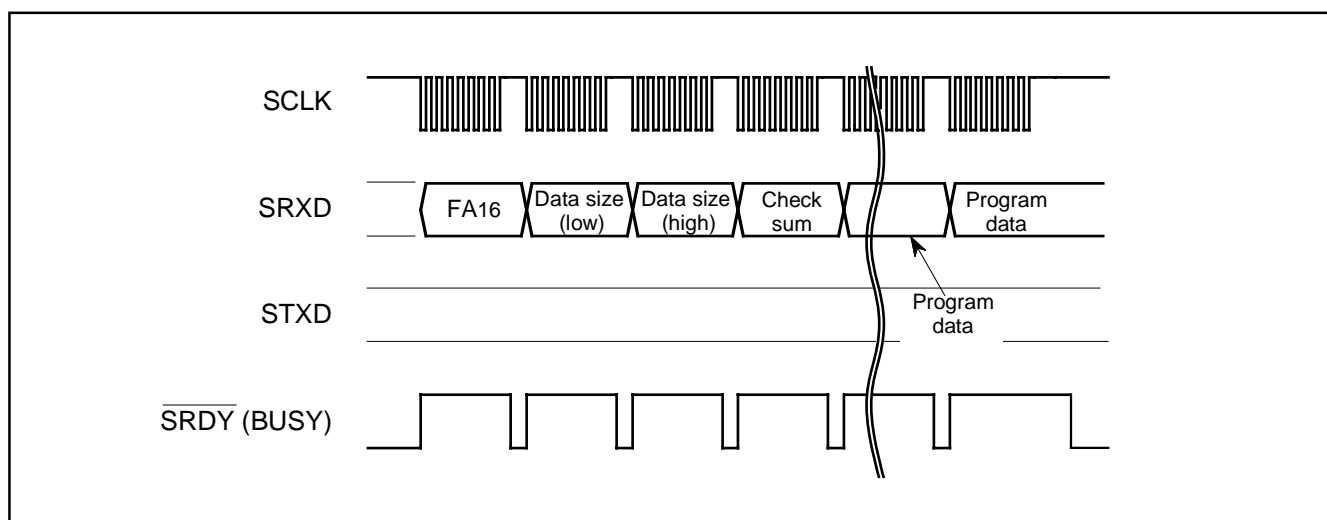


Fig. 87 Timing for download

●Version Information Output Command

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

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

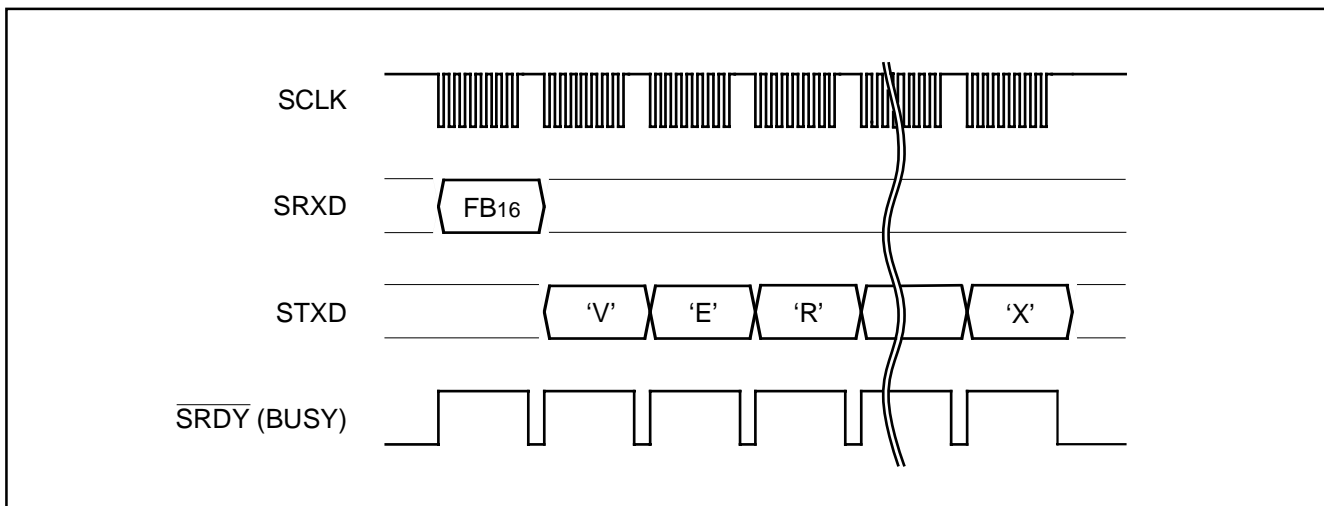


Fig. 88 Timing for version information output

●Boot ROM Area Output Command

This command reads the control program stored in the Boot ROM area in page (256 bytes) unit. Execute the Boot ROM area output command as explained here following.

- (1) Transfer the "FC16" command code with the 1st byte.
- (2) Transfer addresses AB8 to AB15 and AB16 to AB23 with the 2nd and 3rd bytes respectively.
- (3) From the 4th byte onward, data (DB0 to DB7) for the page (256 bytes) specified with addresses AB8 to AB23 will be output sequentially from the smallest address first synchronized with the fall of the clock.

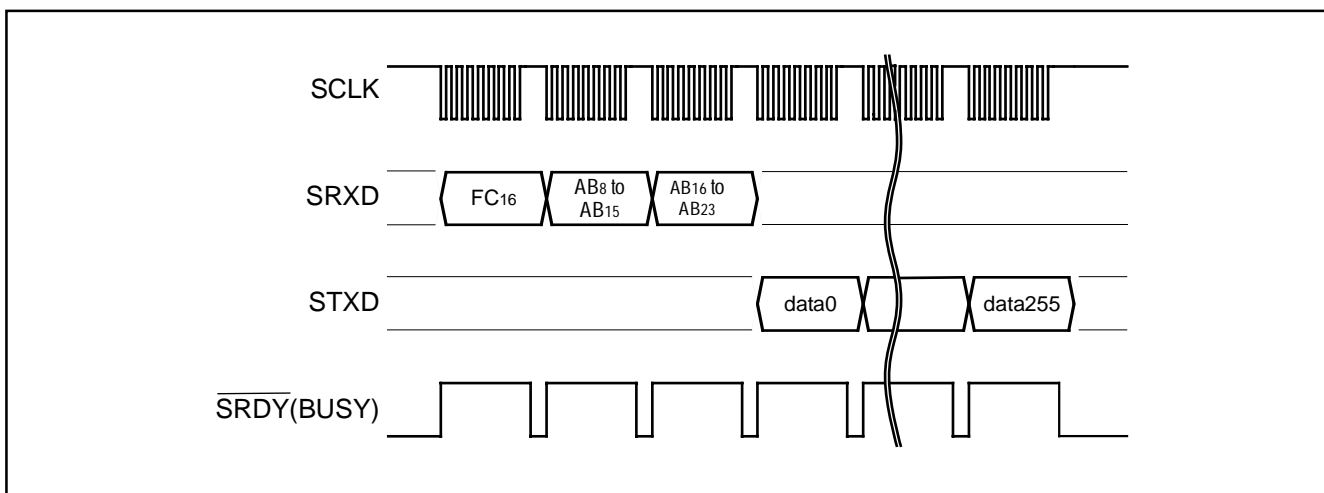


Fig. 89 Timing for Boot ROM area output

●ID Code Check

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

- (1) Transfer the "F5<sub>16</sub>" command code with the 1st byte.
- (2) Transfer addresses AB<sub>0</sub> to AB<sub>7</sub>, AB<sub>8</sub> to AB<sub>15</sub> and AB<sub>16</sub> to AB<sub>23</sub> ("00<sub>16</sub>") of the 1st byte of the ID code with the 2nd and 3rd respectively.
- (3) Transfer the number of data sets of the ID code with the 5th byte.
- (4) Transfer the ID code with the 6th byte onward, starting with the 1st byte of the code.

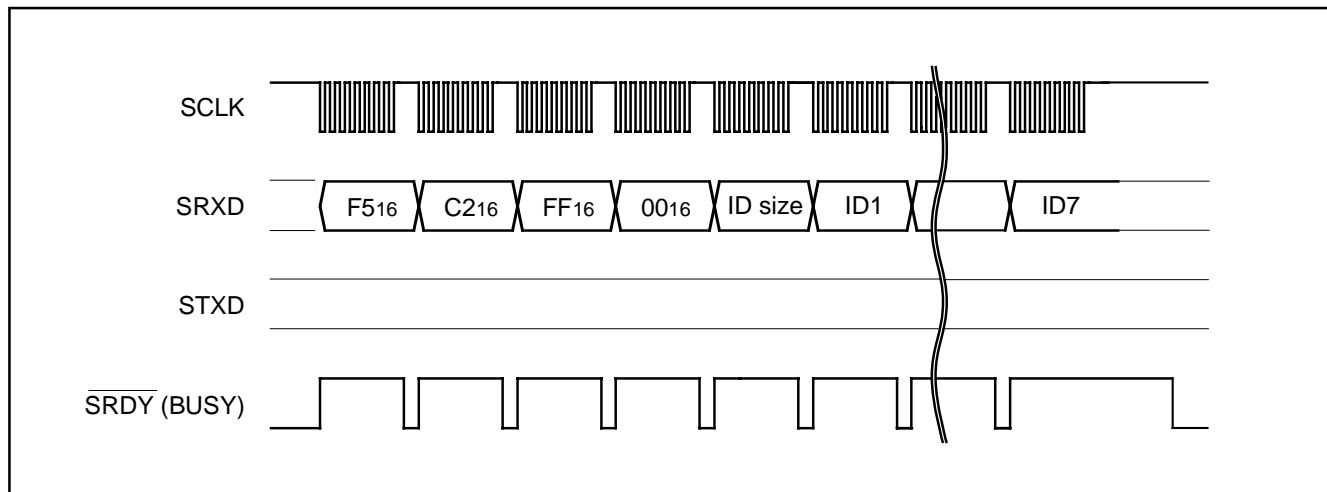


Fig. 90 Timing for ID check

●ID Code

When the flash memory is not blank, the ID code sent from the serial programmer and the ID code written in the flash memory are compared to see if they match. If the codes do not match, the command sent from the serial programmer is not accepted. An ID code contains 8 bits of data. Area is, from the 1st byte, addresses FFC2<sub>16</sub> to FFC8<sub>16</sub>. Write a program into the flash memory, which already has the ID code set for these addresses.

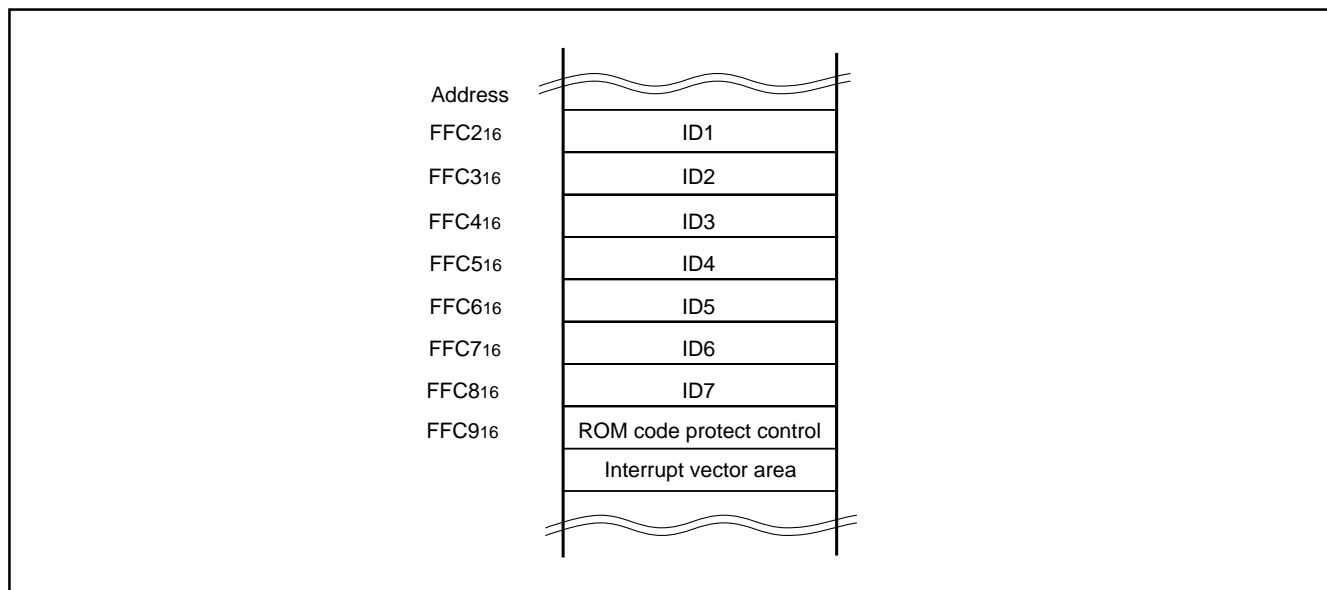


Fig. 91 ID code storage addresses

### ●Status Register (SRD)

The status register indicates operating status of the flash memory and status such as whether an erase operation or a program ended successfully or in error. It can be read by writing the read status register command (70<sub>16</sub>). Also, the status register is cleared by writing the clear status register command (50<sub>16</sub>).

Table 14 lists the definition of each status register bit. After releasing the reset, the status register becomes "80<sub>16</sub>".

### •Sequencer status (SR7)

The sequencer status indicates the operating status of the flash memory.

After power-on and recover from deep power down mode, the sequencer status is set to "1" (ready).

This status bit is set to "0" (busy) during write or erase operation and is set to "1" upon completion of these operations.

### •Erase status (SR5)

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

### •Program status (SR4)

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

Table 14 Definition of each bit of status register (SRD)

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

### ●Status Register 1 (SRD1)

The status register 1 indicates the status of serial communications, results from ID checks and results from check sum comparisons. It can be read after the status register (SRD) by writing the read status register command (70<sub>16</sub>). Also, status register 1 is cleared by writing the clear status register command (50<sub>16</sub>).

Table 15 lists the definition of each status register 1 bit. This register becomes "00<sub>16</sub>" when power is turned on and the flag status is maintained even after the reset.

### •Boot update completed bit (SR15)

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

### •Check sum consistency bit (SR12)

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

### •ID code check completed bits (SR11 and SR10)

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

### •Data reception time out (SR9)

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

Table 15 Definition of each bit of status register 1 (SRD1)

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



## Full Status Check

Results from executed erase and program operations can be known by running a full status check. Figure 92 shows a flowchart of the full status check and explains how to remedy errors which occur.

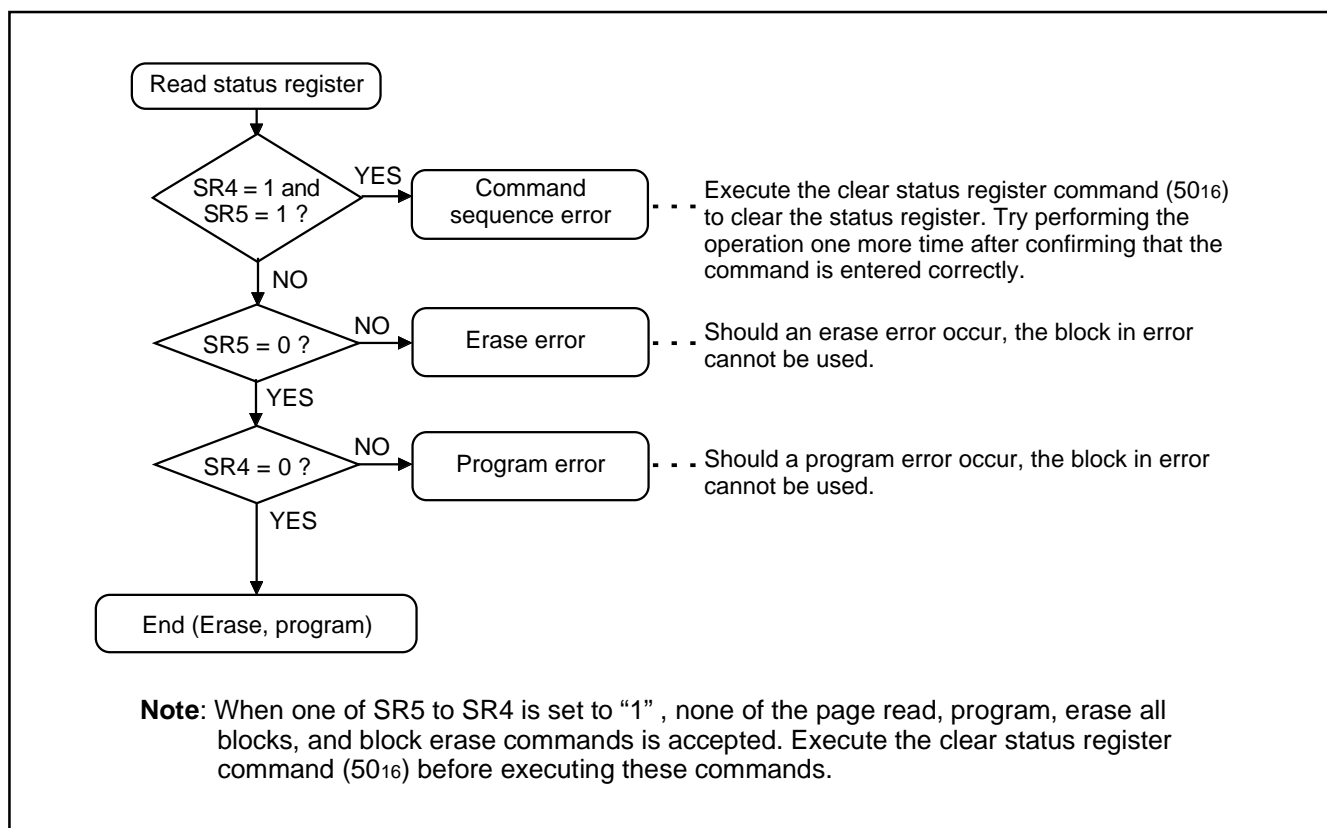


Fig. 92 Full status check flowchart and remedial procedure for errors

### Example Circuit Application for Standard Serial I/O Mode

Figure 93 shows a circuit application for the standard serial I/O mode. Control pins will vary according to a programmer, therefore see a programmer manual for more information.

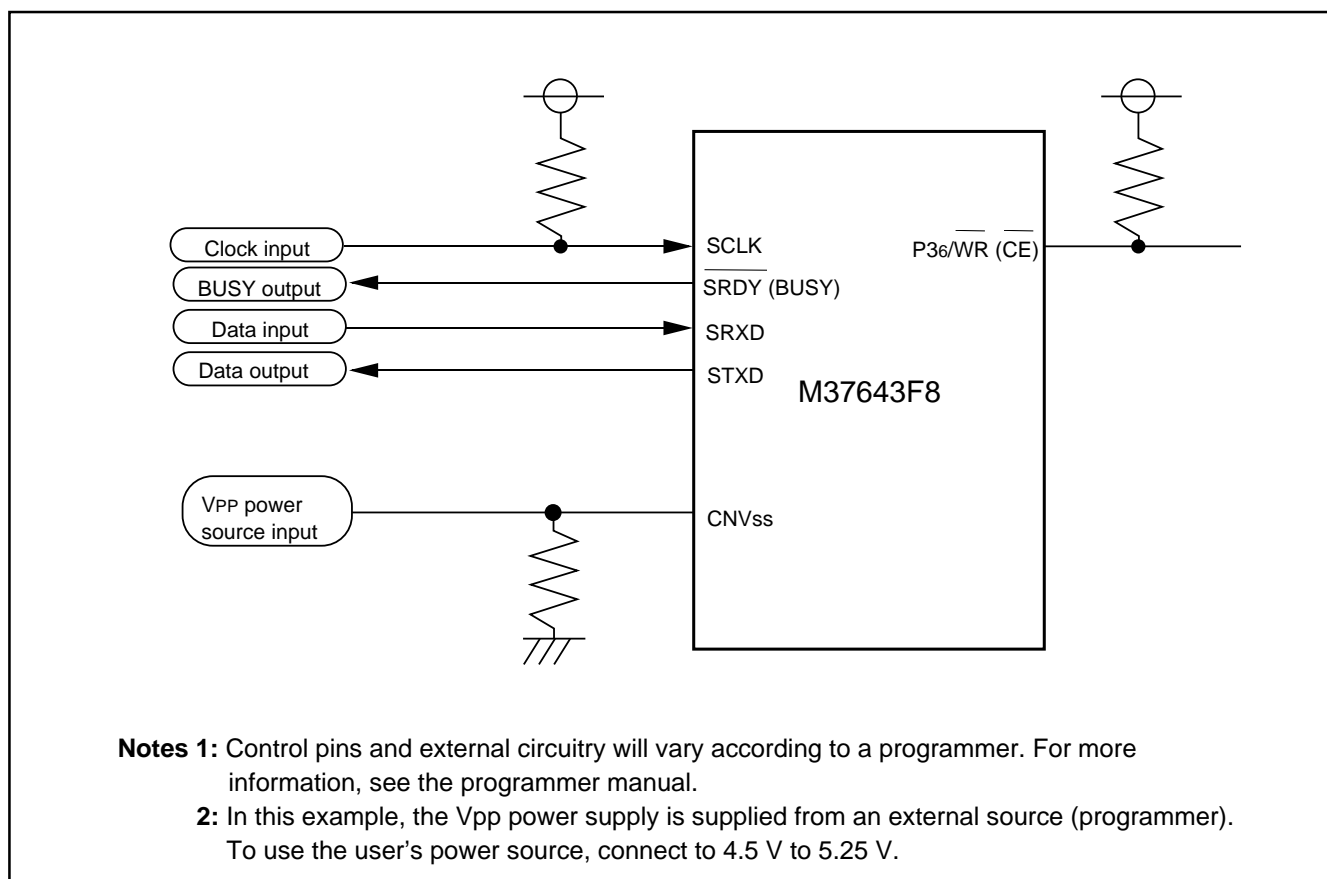


Fig. 93 Example circuit application for standard serial I/O mode

## NOTES ON PROGRAMMING

### Processor Status Register

•The contents of the processor status register (PS) after a reset are undefined, except for the interrupt disable flag (I) which is "1". After a reset, initialize flags which affect program execution. In particular, it is essential to initialize the index X mode (T) and the decimal mode (D) flags because of their effect on calculations.

•To reference the contents of the processor status register (PS), execute the **PHP** instruction once then read the contents of (S+1). If necessary, execute the **PLP** instruction to return the PS to its original status.

A **NOP** instruction must be executed after every **PLP** instruction.

•A **SEI** instruction must be executed before every **PLP** instruction. A **NOP** instruction must be executed before every **CLI** instruction.

### BRK Instruction

It can be detected that the **BRK** instruction interrupt event or the least priority interrupt event by referring the stored B flag state. Refer to the stored B flag state in the interrupt routine.

### Decimal Calculations

When decimal mode is selected, the values of the V flags are invalid.

The carry flag (C) is set to "1" if a carry is generated as a result of the calculation, or is cleared to "0" if a borrow is generated. To determine whether a calculation has generated a carry, the C flag must be initialized to "0" before each calculation. To check for a borrow, the C flag must be initialized to "1" before each calculation.

### Multiplication and Division Instructions

•The index X mode (T) and the decimal mode (D) flags do not affect the **MUL** and **DIV** instruction.

### Instruction Execution Time

The instruction execution time is obtained by multiplying the frequency of the internal clock  $\phi$  by the number of cycles needed to execute an instruction.

The number of cycles required to execute an instruction is shown in the list of machine instructions.

### Timers

•If a value n (between 0 and 255) is written to a timer latch, the frequency division ratio is  $1/(n+1)$ .

•P51/XCOUT/TOUT pin cannot function as an I/O port when XCIN - XCOUT is oscillating. When XCIN - XCOUT oscillation is not used or XCOUT oscillation drive is disabled, this pin can function as the TOUT output pin of the timer 1 or 2.

When using the TOUT output function and  $f(\text{XCIN})$  divided by 2 is used as the timer 1 count source (bit 2 of T123M = "1"), disable XCOUT oscillation drive (bit 5 of CCR = "1").

### Ports

•When the data register (port latch) of an I/O port is modified with the bit managing instruction (**SEB**, **CLB** instructions) the value of the unspecified bit may be changed.

•In standby state (the stop mode by executing the **STP** instruction, and the wait mode by executing the **WIT** instruction) for low-power dissipation, do not make input levels of an I/O port "undefined", especially for I/O ports of the P-channel and the N-channel open-drain.

Pull-up (connect the port to Vcc) or pull-down (connect the port to Vss) these ports through a resistor.

When determining a resistance value, note the following points:

- (1) External circuit
- (2) Variation of output levels during the ordinary operation

When using built-in pull-up or pull-down resistor, note on varied current values.

- (1) When setting as an input port : Fix its input level
- (2) When setting as an output port : Prevent current from flowing out to external

### Serial I/O

Do not write to the serial I/O shift register during a transfer when in SPI compatible mode.

### UART

•The all error flags PER, FER, OER and SER are cleared to "0" when the UART status register is read, at the hardware reset or initialization by setting the Transmit Initialization Bit. These flags are also cleared to "0" by execution of bit test instructions such as **BBC** and **BCS**.

•The transmission interrupt request bit might be set and the interrupt request is generated by setting the transmit initialization bit to "1" even when selecting timing that either of the following flags is set to "1" as timing where the transmission interrupt is generated:

- (1) Transmit buffer empty flag is set to "1"
- (2) Transmit complete flag is set to "1".

Therefore, when the transmit interrupt is used, set the transmit interrupt enable bit to transmit enabled as the following sequence:

- (1) Transmit initialization bit is set to "1"
- (2) Transmit interrupt request bit is set to "0"
- (3) Transmit interrupt enable bit is set to "1".

•Do not update a value of UART baud rate generator in the condition of transmission enabled or reception enabled. Disable transmission and reception before updating the value. If the former data remains in the UART transmit buffer registers 1 and 2 when transmission is enabled, an undefined data might be output.

•The receive buffer full interrupt request is not generated if receive errors are detected at receiving.

•If a character bit length is 7 bits, bit 7 of the UART transmit/receive buffer register 1 and bits 0 to 7 of the UART transmit/receive buffer register 2 are ignored at transmitting; they are invalid at receiving.

If a character bit length is 8 bits, bits 0 to 7 of the UART transmit/receive buffer register 2 are ignored at transmitting; they are invalid at receiving.

If a character bit length is 9 bits, bits 1 to 7 of the UART transmit/receive buffer register 2 are ignored at transmitting; they are "0" at receiving.

## USB

•When the USB Reset Interrupt Status Flag is kept at "1", all other flags in the USB internal registers (addresses 0050<sub>16</sub> to 005F<sub>16</sub>) will return to their reset status. However, the following registers are not affected by the USB reset: USB control register (address 0013<sub>16</sub>), Frequency synthesizer control register (address 006C<sub>16</sub>), Clock control register (address 001F<sub>16</sub>), and USB endpoint-x FIFO register (addresses 0060<sub>16</sub> to 0062<sub>16</sub>).

•When not using the USB function, set the USB Line Driver Supply Enable Bit of the USB control register (address 0013<sub>16</sub>) to "1" for power supply to the internal circuits (at V<sub>cc</sub> = 5V).

•The IN\_PKT\_RDY Bit can be set by software even when using the AUTO\_SET function.

•When writing to USB-related registers, set the USB Clock Enable Bit to "1", then perform the write after four  $\phi$  cycle waits.

•When using the MCU at V<sub>cc</sub> = 3.3V, set the USB Line Driver Supply Enable Bit to "0" (line driver disable). Note that setting the USB Line Driver Current Control Bit (USBC3) doesn't affect the USB operation.

•Read one packet data from the OUT FIFO before clearing the OUT\_PKT\_RDY Flag. If the OUT\_PKT\_RDY Flag is cleared while one packet data is being read, the internal read pointer cannot operate normally.

•Use the transfer instructions such as **LDA** and **STA** to set the registers: USB interrupt status registers 1, 2 (addresses 0052<sub>16</sub>, 0053<sub>16</sub>); USB endpoint 0 IN control register (address 0059<sub>16</sub>); USB endpoint x IN control register (address 0059<sub>16</sub>); USB endpoint x OUT control register (address 005A<sub>16</sub>). Do not use the read-modify-write instructions such as the **SEB** or the **CLB** instruction.

When writing to bits shown by Table 15 using the transfer instruction such as **LDA** or **STA**, a value which never affect its bit state is required. Take the following sequence to change these bits contents:

- (1) Store the register contents onto a variable or a data register.
- (2) Change the target bit on the variable or the data register. Simultaneously mask the bit so that its bit state cannot be changed. (See to Table 15.)
- (3) Write the value from the variable or the data register to the register using the transfer instruction such as **LDA** or **STA**.

•To use the AUTO\_SET function for an IN transfer when the AUTO\_SET bit is set to 1, set the FIFO to single buffer mode.

**Table 15 Bits of which state might be changed owing to software write**

Register name	Bit name	Value not affecting state ( <b>Note</b> )
USB endpoint 0 IN control register	IN_PKT_RDY (b1)	"0"
	DATA_END (b3)	"0"
	FORCE_STALL (b4)	"1"
USB endpoint x (x = 1, 2) IN control register	IN_PKT_RDY (b0)	"0"
USB endpoint x (x = 1, 2) OUT control register	OUT_PKT_RDY (b0)	"1"
	FORCE_STALL (b4)	"1"

**Note:** Writing this value will not change the bit state, because this value cannot be written to the bit by software.

## Frequency Synthesizer

•The frequency synthesizer and DC-DC converter must be set up as follows when recovering from a Hardware Reset:

- (1) Enable the frequency synthesizer after setting the frequency synthesizer related registers (addresses 006C16 to 006F16). Then wait for 2 ms.
- (2) Check the Frequency Synthesizer Lock Status Bit. If "0", wait for 0.1 ms and then recheck.
- (3) When using the USB built-in DC-DC converter, set the USB Line Driver Supply Enable Bit of the USB control register to "1". This setting must be done 2 ms or more after the setup described in step (1). The USB Line Driver Current Control Bit must be set to "0" at this time. (When Vcc = 3.3V, the setting explained in this step is not necessary.)
- (4) After waiting for (C + 1) ms so that the external capacitance pin (Ext. Cap. pin) can reach approximately 3.3 V, set the USB Clock Enable Bit to "1". At this time, "C" equals the capacitance ( $\mu$ F) of the capacitor connected to the Ext. Cap. pin. For example, if 2.2  $\mu$ F and 0.1  $\mu$ F capacitors are connected to the Ext. Cap. in parallel, the required wait will be (2.3 + 1) ms.
- (5) After enabling the USB clock, wait for 4 or more f cycles, and then set the USB Enable Bit to "1". After enabling USB clock, read or write the USB internal registers (address 005016 to 006216 with the exception of USBC, CCR and PSC) .

•Bits 6 and 5 of the frequency synthesizer control register (address 006C16) are initialized to "11" after reset release. Make sure to set bits 6 and 5 to "10" after the Frequency Synthesizer Lock Status Bit goes to "1".

•When using the frequency synthesized clock function, we recommend using the fastest frequency possible of f(XIN) or f(XCIN) as an input clock for the PLL. Owing to the PLL mechanism, the PLL controls the speed of multiplied clocks from the source clock. As a result, when the source clock input is lower, the generated clock becomes less stable. This is because more multipliers are needed and the speed control is very rough. Higher source clock input generates a stabler clock, as less multipliers are needed and the speed control is more accurate. However, if the input clock frequency is relatively high, the PLL clock generator can quickly lock-up the output clock to the source and make the output clock very stable.

•Set the value of frequency synthesizer multiply register 2 (FSM2) so that the fPIN is 1 MHz or higher.

## DMA

•In the memory expansion mode and microprocessor mode, the DMAOUT pin outputs "H" during a DMA transfer.

•Do not access the DMAC-related registers by using a DMAC transfer. The destination address data and the source address data will collide in the DMAC internal bus.

•When using the USB FIFO as the DMA transfer source, make sure that, if you use the AUTO\_SET function, short packet data

does not get mixed in with the transfer data.

•When setting the DMAC channel x enable bit (bit 7 of address 004116) to "1", be sure simultaneously to set the DMAC channel x transfer initiation source capture register reset bit (bit 6 of address 004116) to "1". If this is not performed, an incorrect data will be transferred at the same time when the DMAC is enabled.

## Memory Expansion Mode & Microprocessor Mode

•In both memory expansion mode and microprocessor mode, use the LDM instruction or STA instruction to write to port P3 (address 000E16). When using the Read-Modify-Write instruction (**SEB** instruction, **CLB** instruction) you will need to map a memory that the CPU can read from and write to.

•In the memory expansion mode, if the internal and external memory areas overlap, the internal memory becomes the valid memory for the overlapping area. When the CPU performs a read or a write operation on this overlapped area, the following things happen:

### (1) Read

The CPU reads out the data in the internal memory instead of in the external memory. Note that, since the CPU will output a proper read signal, address signal, etc., the memory data at the respective address will appear on the external data bus.

### (2) Write

The CPU writes data to both the internal and external memories.

•The wait function is serviceable at accessing an external memory.

## Stop Mode

•When the STP instruction is executed, bit 7 of the clock control register (address 001F16) goes to "0". To return from stop mode, reset CCR7 to "1".

•When using fSYN (set Internal System Clock Select Bit (CPMA6) to "1") as the internal system clock, switch CPMA6 to "0" before executing the **STP** instruction. Reset CPMA6 after the system returns from Stop Mode and the frequency synthesizer has stabilized.

CPMA6 does not need to be switched to "0" when using the **WIT** instruction.

•When the **STP** instruction is being executed, all bits except bit 4 of the timer 123 mode register (address 002916) are initialized to "0". It is not necessary to set T123M1 (Timer 1 Count Stop Bit) to "0" before executing the **STP** instruction. After returning from Stop Mode, reset the timer 1 (address 002416), timer 2 (address 002516), and the timer 123 mode register (address 002916).

## USAGE NOTES

### Oscillator Connection Notice

The built-in feedback register (1 M $\Omega$ ) and the dumping resistor (400  $\Omega$ ) is internally connected between pins XIN and XOUT.

### Power Source Voltage

When the power source voltage value of a microcomputer is less than the value which is indicated as the recommended operating conditions, the microcomputer does not operate normally and may perform unstable operation.

In a system where the power source voltage drops slowly when the power source voltage drops or the power supply is turned off, reset a microcomputer when the power source voltage is less than the recommended operating conditions and design a system not to cause errors to the system by this unstable operation.

### Power Supply Pins Treatment Notice

Please connect 0.1  $\mu$ F and 4.7  $\mu$ F capacitors in parallel between pins Vcc and Vss, and pins AVss and AVcc.

These capacitors must be connected as close as possible between the DC supply and GND pins, and also the analog supply pin and corresponding GND pin.

Wiring patterns for these supply and GND pins must be wider than other signal patterns.

These filter capacitors should not be placed near the LPF pins as they will cause noise problems

### Reset Pin Treatment Notice (Noise Elimination)

If the reset input signal rises very slowly, we recommend attaching a capacitor, such as a 1000 pF ceramic capacitor with excellent high frequency characteristics, between the RESET pin and the Vss pin.

Please note the following two issues for this capacitor connection.

- (1) Capacitor wiring pattern must be as short as possible (within 20 mm).
- (2) The user must perform an application level operation test.

### LPF Pin Treatment Notice

All passive components must be located as close as possible to the LPF pin.

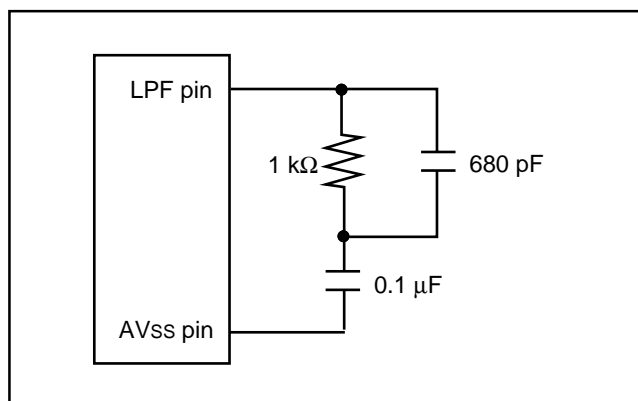


Fig. 94 Passive components near LPF pin

### AVss and AVcc Pin Treatment Notice (Noise Elimination)

An insulation connector (Ferrite Beads) must be connected between AVss and Vss pins and between AVcc and Vcc pins.

### USB Transceiver Treatment (Noise Elimination)

•The Full-Speed USB2.0 specification requires a driver impedance 28 to 44  $\Omega$ . (Refer to Clause 7.1.1.1 Full-speed (12 Mb/s) Driver Characteristics in the USB specification.) In order to meet the USB specification impedance requirements, connect a resistor (27  $\Omega$  to 33  $\Omega$  recommended) in series to the USB D+ pin and the USB D- pin.

In addition, in order to reduce the DC ringing and control the falling/rising timing of USB D+/D- and a crossover point, connect a capacitor between the USB D+/D- pins and the Vss pin if necessary. The values and structure of those peripheral elements depend on the impedance characteristics and the layout of the printed circuit board. Accordingly, evaluate your system and observe waveforms before actual use and decide use of elements and the values of resistors and capacitors.

•Connect a capacitor between the Ext. Cap. pin and the Vss pin. The capacitor should have a 2.2  $\mu$ F capacitor (Tantalum capacitor) and a 0.1  $\mu$ F capacitor (ceramic capacitor) connected in parallel. Figure 95 for the proper positions of the peripheral components.

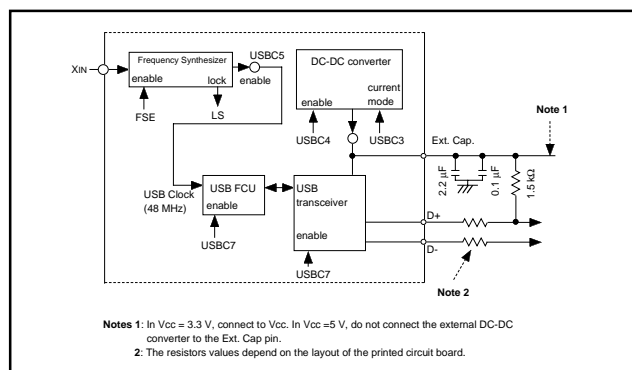


Fig.95 Peripheral circuit

•In Vcc = 3.3 V operation, connect the Ext. Cap. pin directly to the Vcc pin in order to supply power to the USB transceiver. In addition, you will need to disable the DC-DC converter in this operation (set bit 4 of the USB control register to "0".) If you are using the bus powered supply in Vcc = 3.3 V operation, the DC-DC converter must be placed outside the MCU.

•In Vcc = 5 V operation, do not connect the external DC-DC converter to the Ext. Cap. pin. Use the built-in DC-DC converter by enabling the USB line driver.

•Make sure the USB D+/D- lines do not cross any other wires. Keep a large GND area to protect the USB lines. Also, make sure you use a USB specification compliant connector for the connection.

## 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.

## Clock Input/Output Pin Wiring (Noise Elimination)

- (1) Make the wiring for the input/output pins as short as possible.
- (2) Make the wiring across the grounding lead of the capacitor which is connected to an oscillator and the Vss pin of the MCU as short as possible (within 20 mm)
- (3) Make sure to isolate the oscillation Vss pattern from other patterns for oscillation circuit-use only.

## Oscillator Wiring (Noise Elimination)

### (1) Keeping oscillator away from large current signal lines

Install a microcomputer (and especially an oscillator) as far as possible from signal lines, including USB signal lines, where a current larger than the tolerance of current value flows. When a large current flows through those signal lines, strong noise occurs because of mutual inductance.

### (2) Installing oscillator away from signal lines where potential levels change frequently

Install an oscillator and a connecting pattern of an oscillator away from signal lines where potential levels change frequently. Also, do not cross such signal lines over the clock lines or the signal lines which are sensitive to noise.

## Terminate Unused Pins

### (1) Output ports : Open

### (2) Input ports :

Connect each pin to Vcc or Vss through each resistor of 1 kΩ to 10 kΩ.

Ports that permit the selecting of a built-in pull-up or pull-down resistor can also use this resistor. As for pins whose potential affects to operation modes such as pins CNVss, INT or others, select the Vcc pin or the Vss pin according to their operation mode.

### (3) I/O ports :

- Set the I/O ports for the input mode and connect them to Vcc or Vss through each resistor of 1 kΩ to 10 kΩ.

Ports that permit the selecting of a built-in pull-up or pull-down resistor can also use this resistor. Set the I/O ports for the output mode and open them at "L" or "H".

- When opening them in the output mode, the input mode of the initial status remains until the mode of the ports is switched over to the output mode by the program after reset. Thus, the potential at these pins is undefined and the power source current may increase in the input mode. With regard to an effects on the system, thoroughly perform system evaluation on the user side.

- Since the direction register setup may be changed because of a program runaway or noise, set direction registers by program periodically to increase the reliability of program.

- At the termination of unused pins, perform wiring at the shortest possible distance (20 mm or less) from microcomputer pins.

## Electric Characteristic Differences Between Mask ROM and Flash Memory Version MCUs

There are differences in electric characteristics, operation margin, noise immunity, and noise radiation between Mask ROM and Flash Memory version MCUs due to the difference in the manufacturing processes.

When manufacturing an application system with the Flash Memory version and then switching to use of the Mask ROM version, please perform sufficient evaluations for the commercial samples of the Mask ROM version.

## ROM ORDERING METHOD

- 1.Mask ROM Order Confirmation Form
  - 2.Mark Specification Form
  - 3.Data to be written to ROM, in EPROM form (three identical copies) or one floppy disk.
- For the mask ROM confirmation and the mark specifications, refer to the "Renesas Technology Corp." Homepage (<http://www.renesas.com>).

**FUNCTIONAL DESCRIPTION SUPPLEMENT**  
**Timing After Interrupt**

The interrupt processing routine begins with the machine cycle following the completion of the instruction that is currently in execution. Figure 96 shows a timing chart after an interrupt occurs, and Figure 97 shows the time up to execution of the interrupt processing routine.

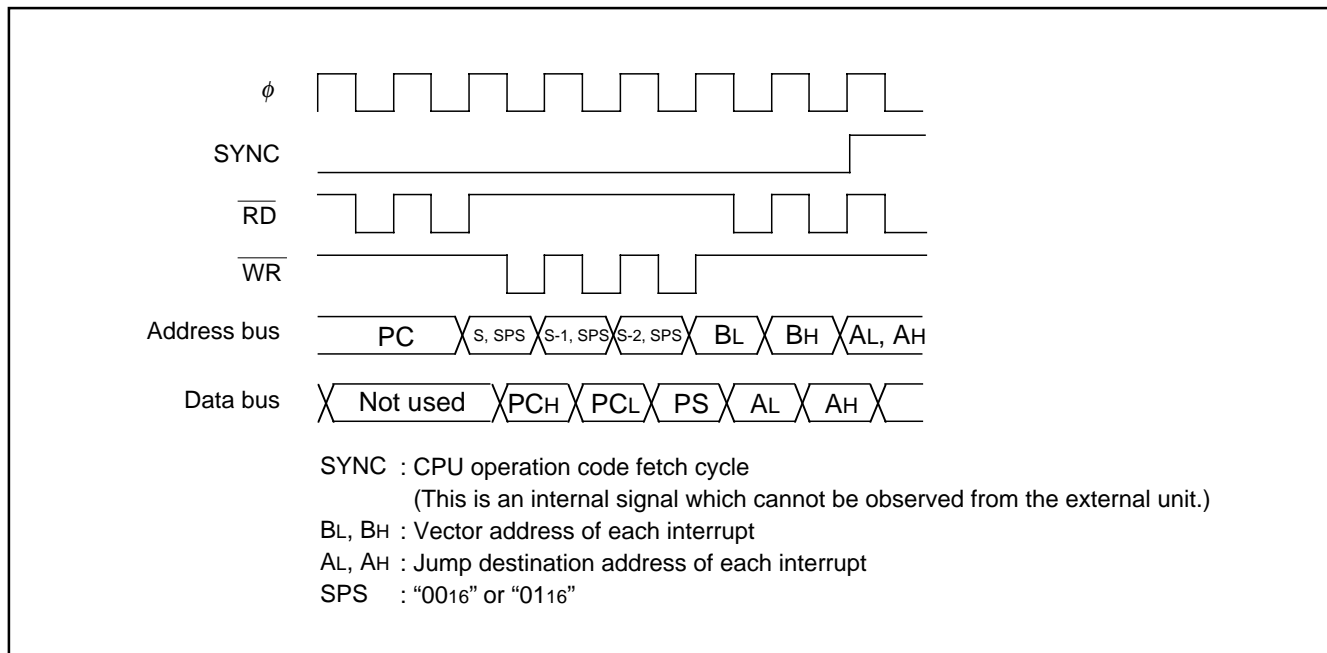


Fig. 96 Timing chart after interrupt occurs

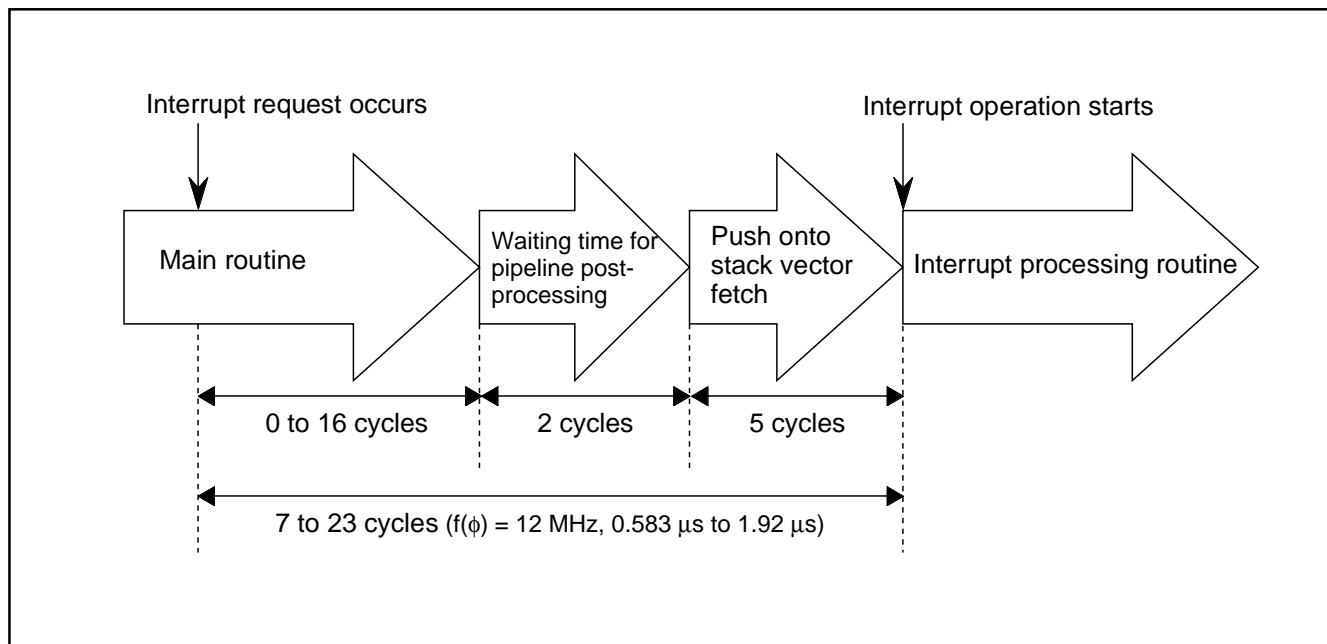


Fig. 97 Time up to execution of interrupt processing routine



THIS PAGE IS BLANK FOR REASONS OF LAYOUT.

# **CHAPTER 2**

---

## **APPLICATION**

**2.1 I/O port**

**2.2 Timer**

**2.3 Serial I/O**

**2.4 UART**

**2.5 DMAC**

**2.6 USB**

**2.7 Frequency synthesizer**

**2.8 External devices connection**

**2.9 Reset**

**2.10 Clock generating circuit**

## 2.1 I/O port

This paragraph explains the registers setting method and the notes related to the I/O port.

### 2.1.1 Memory map

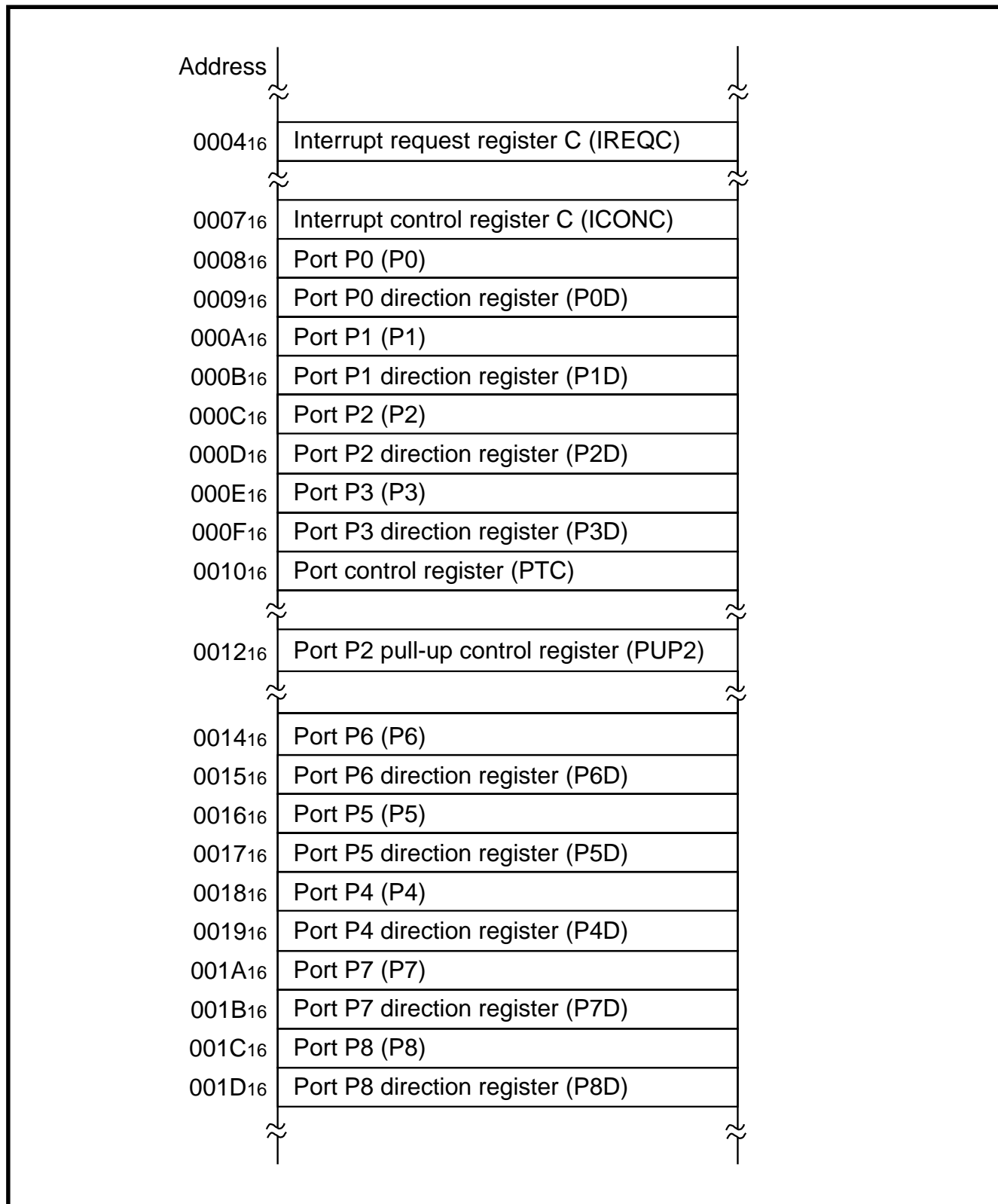


Fig. 2.1.1 Memory map of registers related to I/O port

## 2.1.2 Related registers

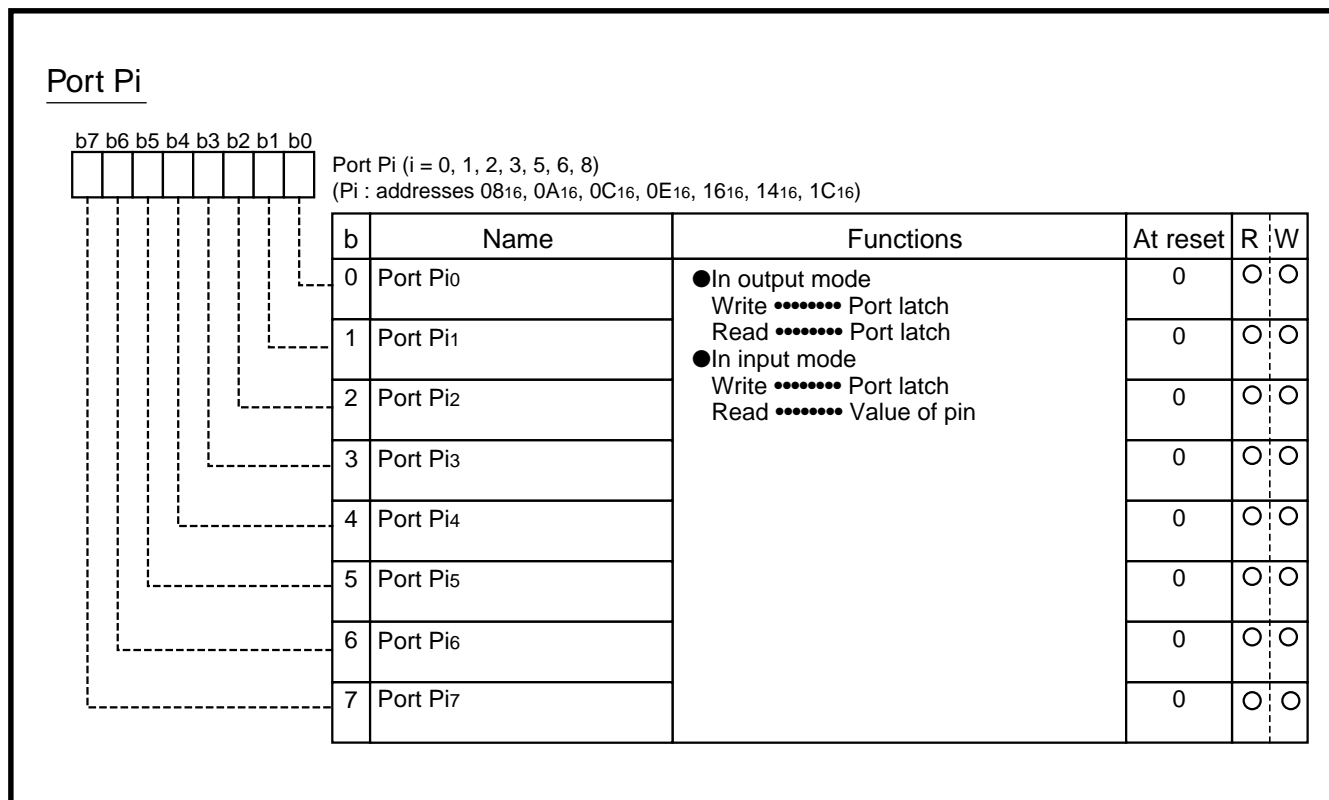


Fig. 2.1.2 Structure of Port Pi register

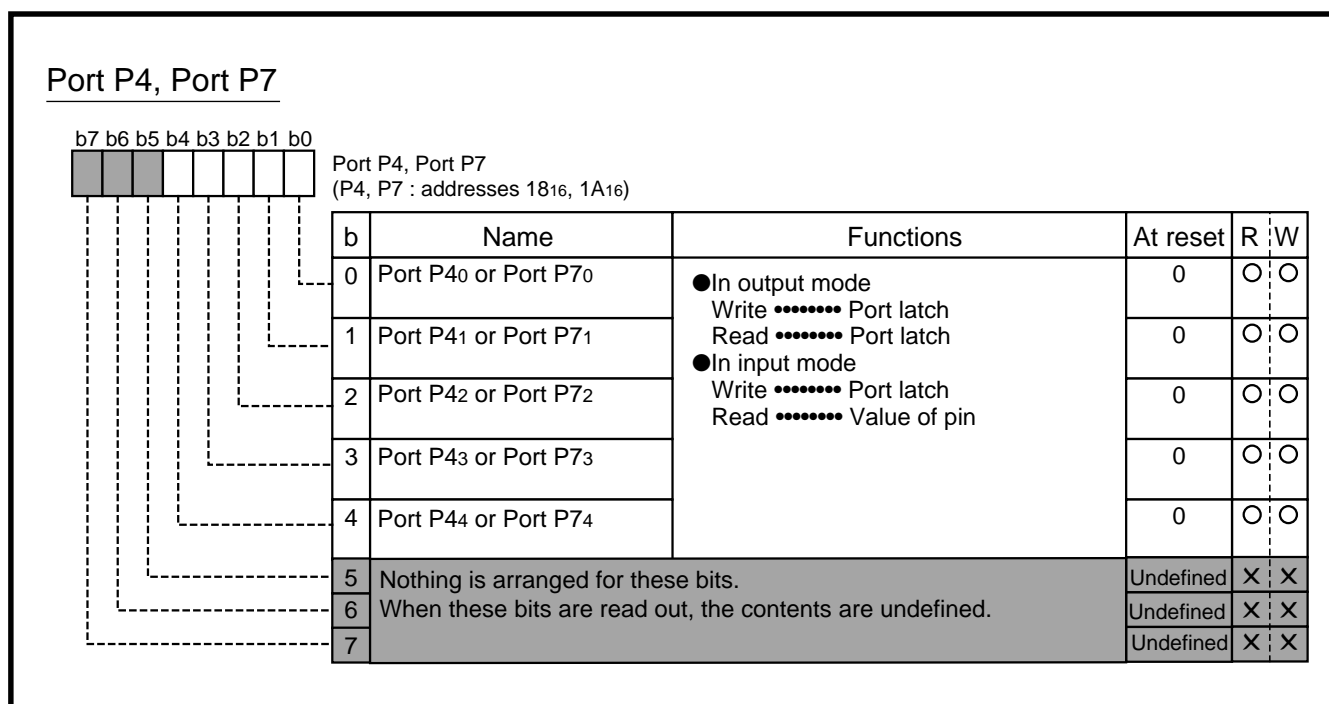


Fig. 2.1.3 Structure of Port P4, Port P7 registers

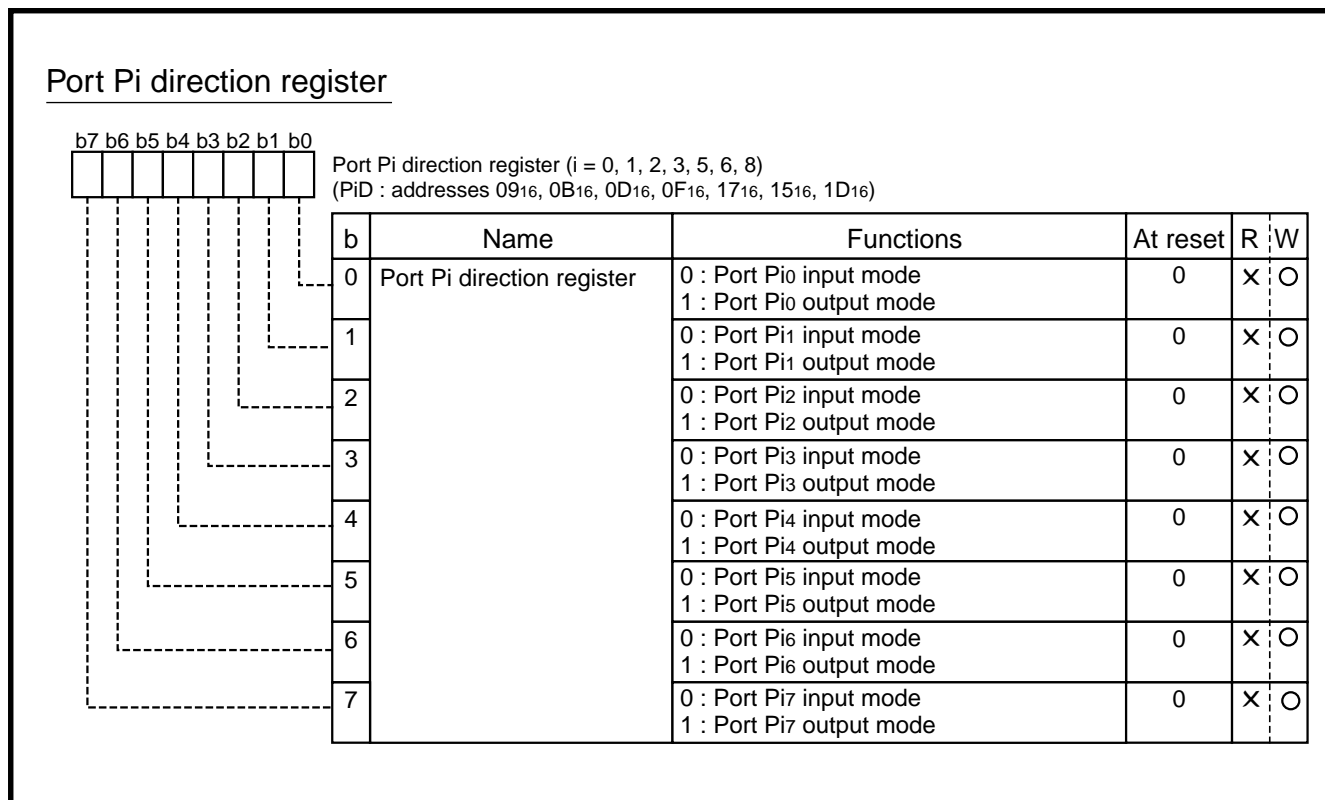
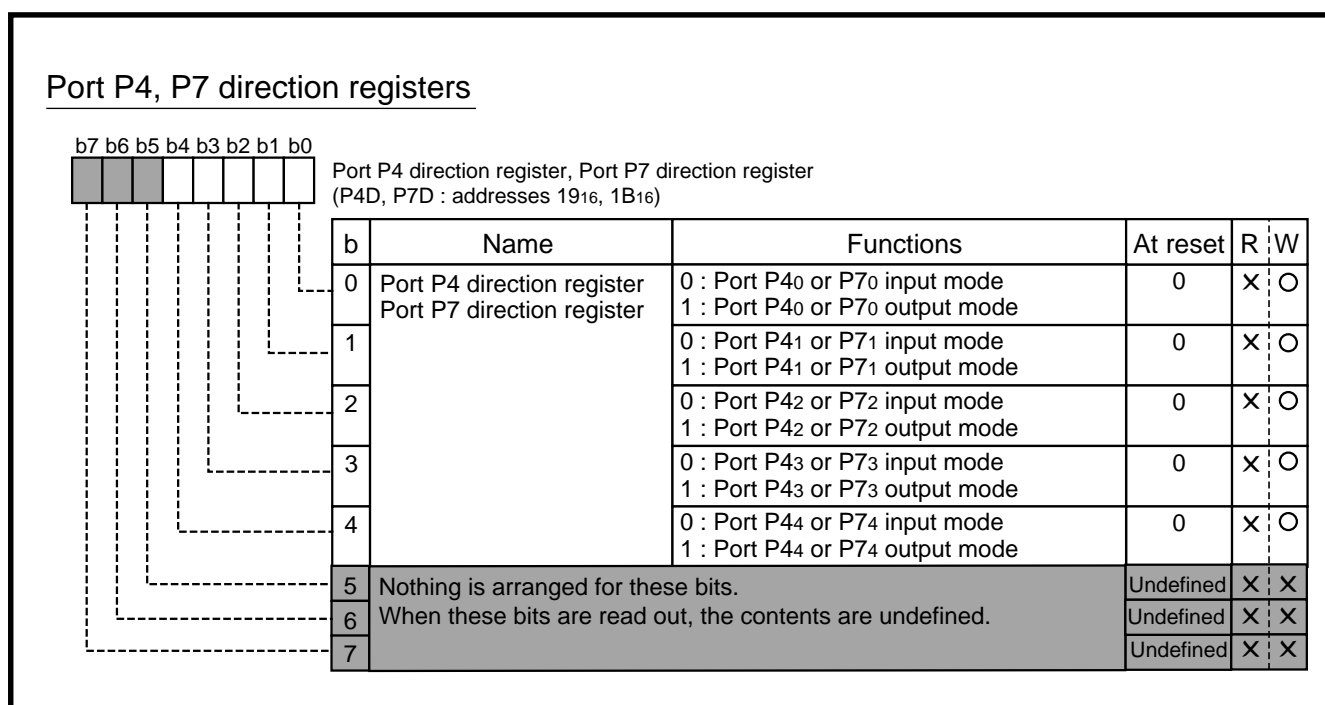
Fig. 2.1.4 Structure of Port Pi direction register ( $i = 0, 1, 2, 3, 5, 6, 8$ )

Fig. 2.1.5 Structure of Port P4 direction, Port P7 direction registers

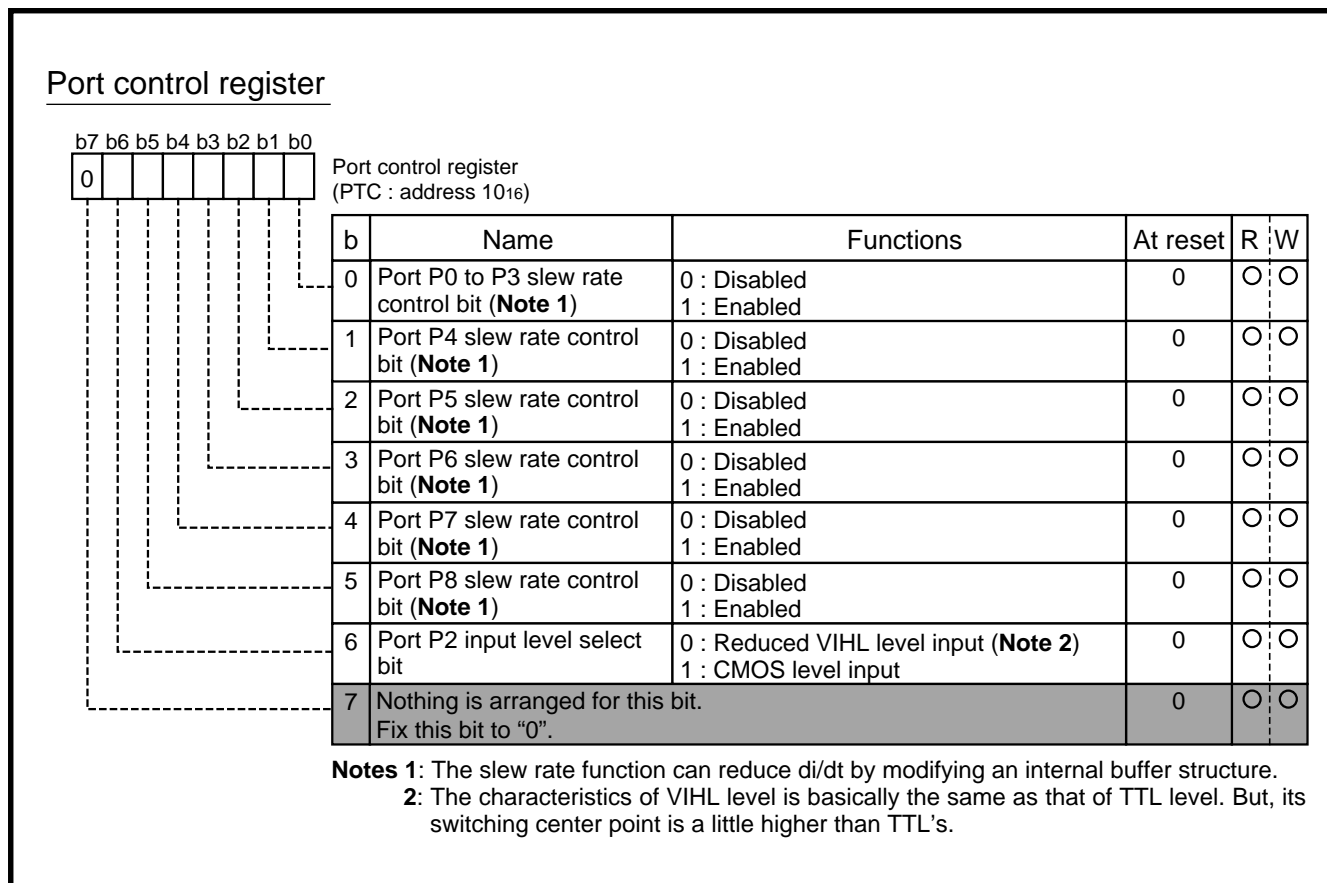


Fig. 2.1.6 Structure of Port control register

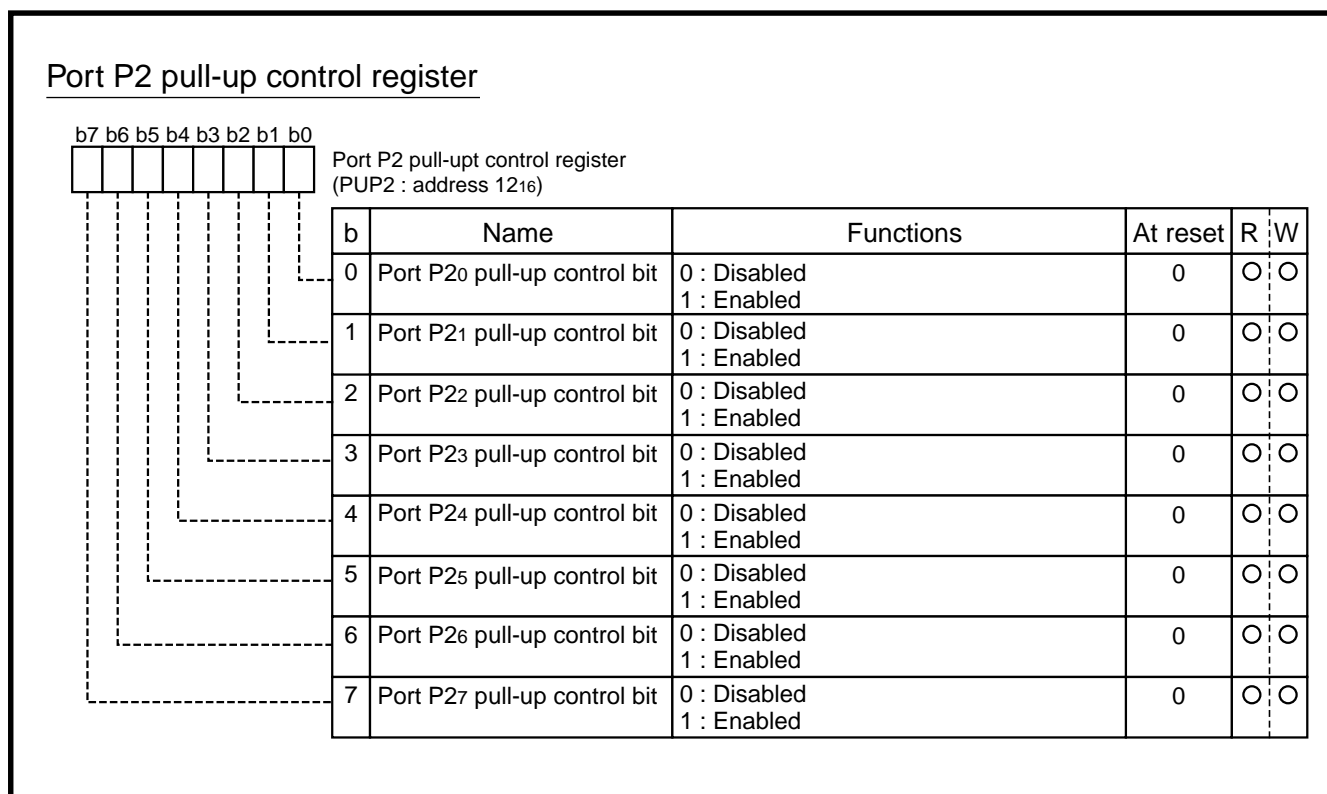


Fig. 2.1.7 Structure of Port P2 pull-up control register

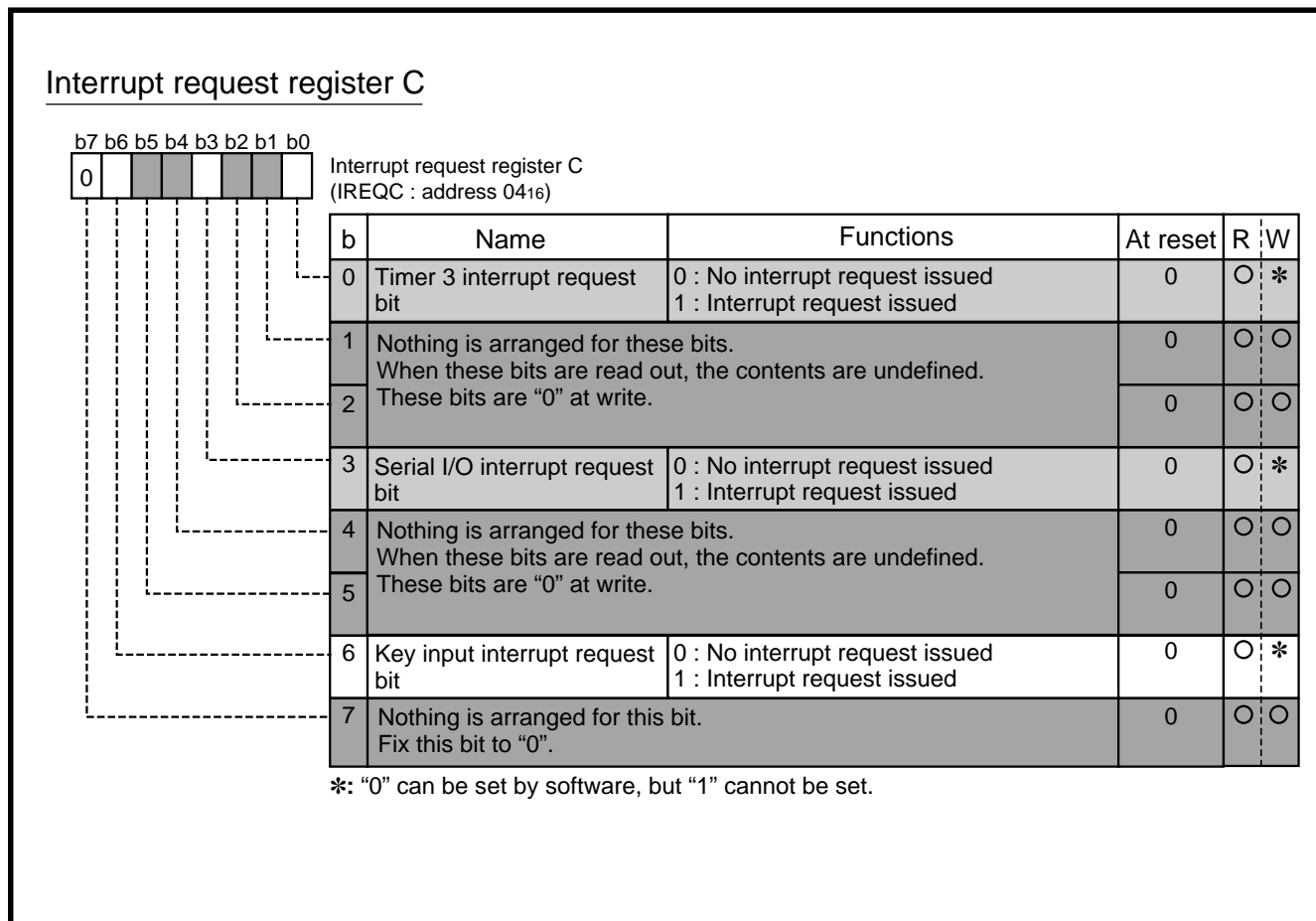


Fig. 2.1.8 Structure of Interrupt request register C

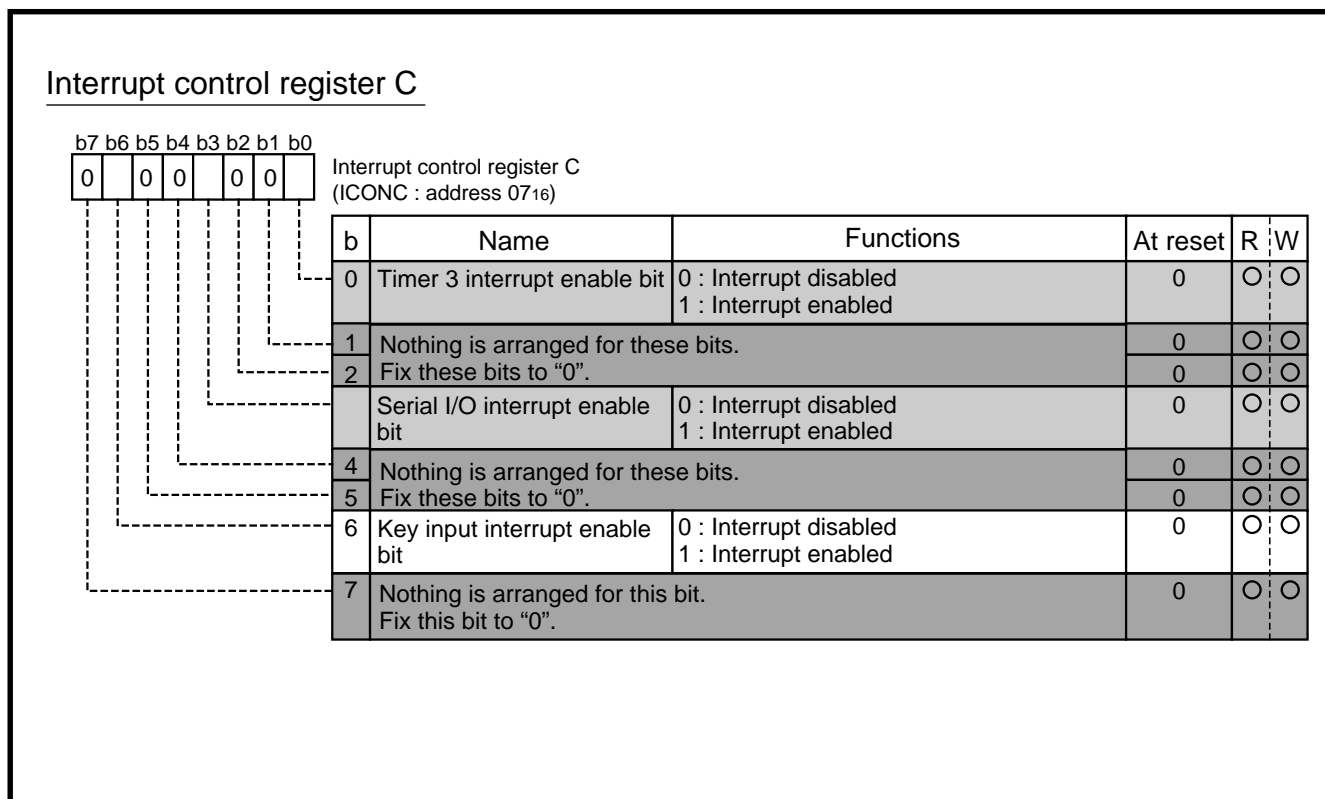


Fig. 2.1.9 Structure of Interrupt control register C

### 2.1.3 Key-on wake-up interrupt application example

**Outline :** Key-on wake-up is realized, using internal pull-up resistors.

**Specifications:** System is returned from the wait mode when the key-on wakeup interrupt occurs by input of the falling edge to port P2i.

Figure 2.1.10 shows the registers setting; Figure 2.1.11 shows a connection diagram; Figure 2.1.12 shows the control procedure.

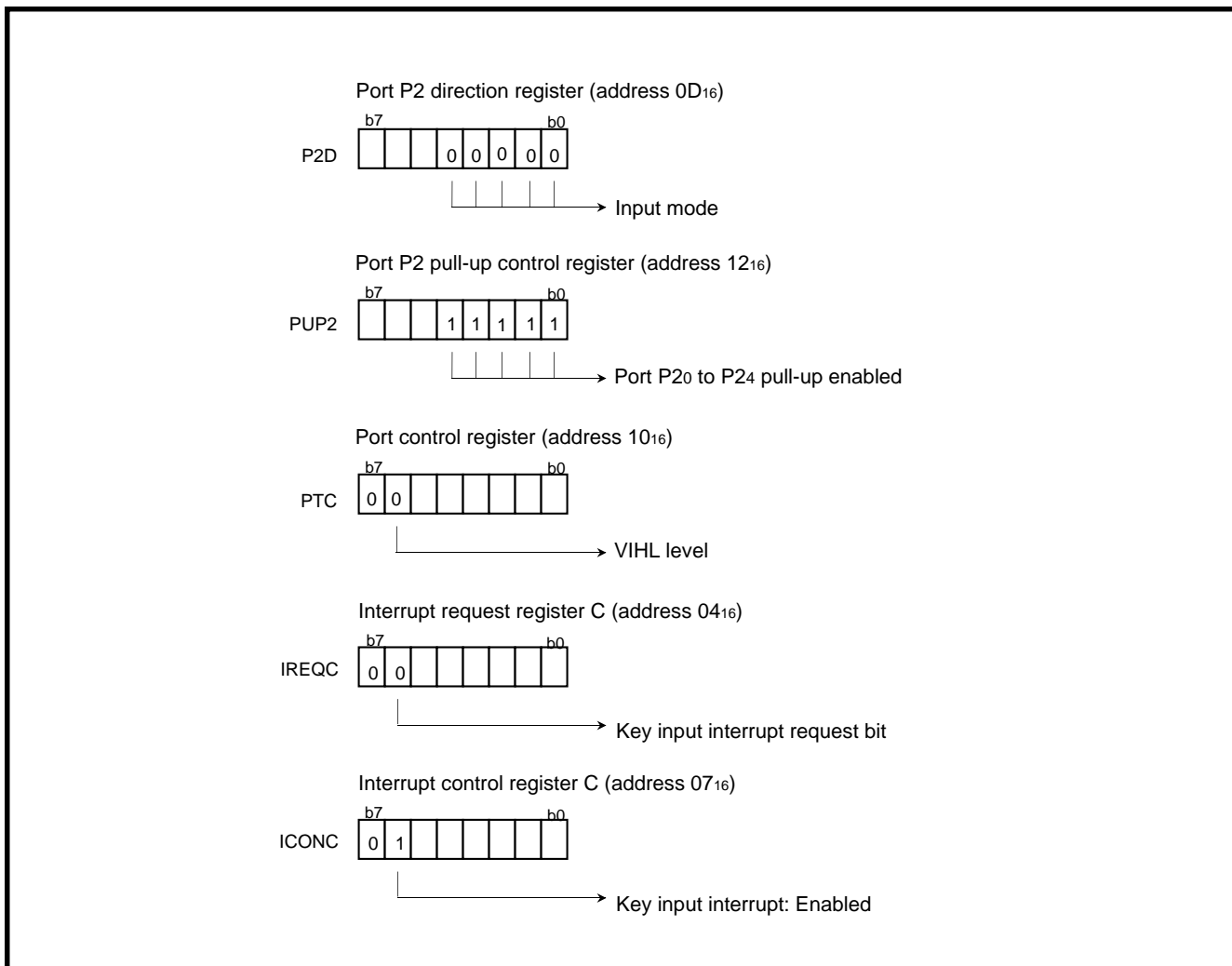


Fig. 2.1.10 Registers setting



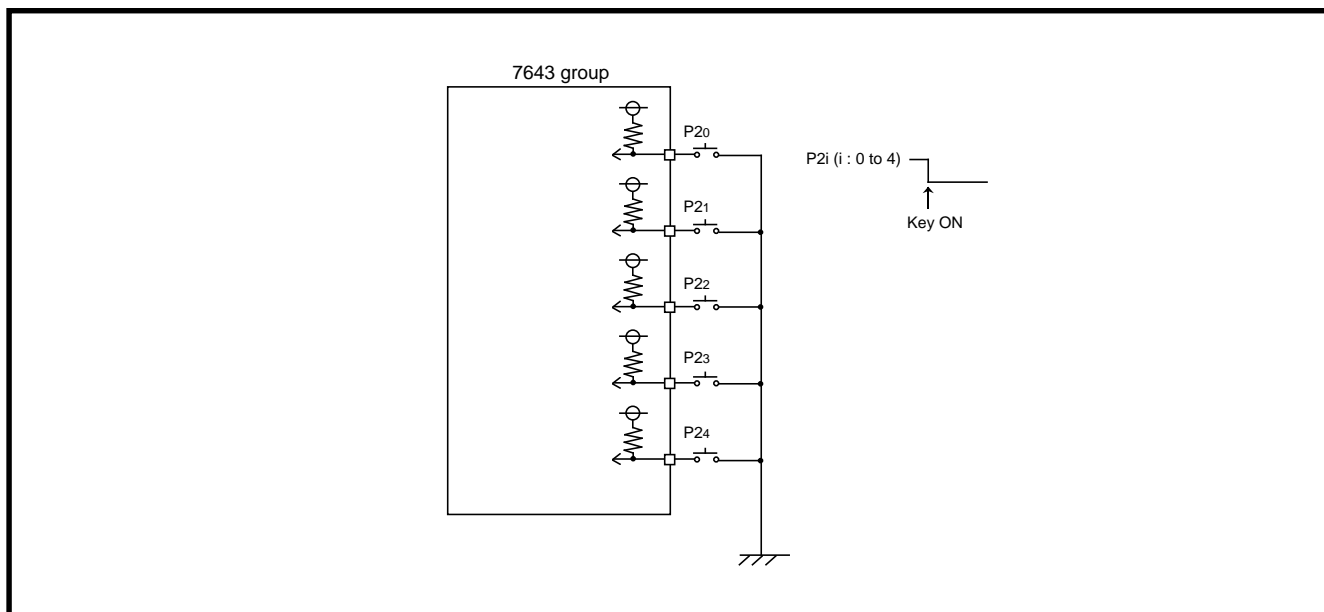


Fig. 2.1.11 Connection diagram

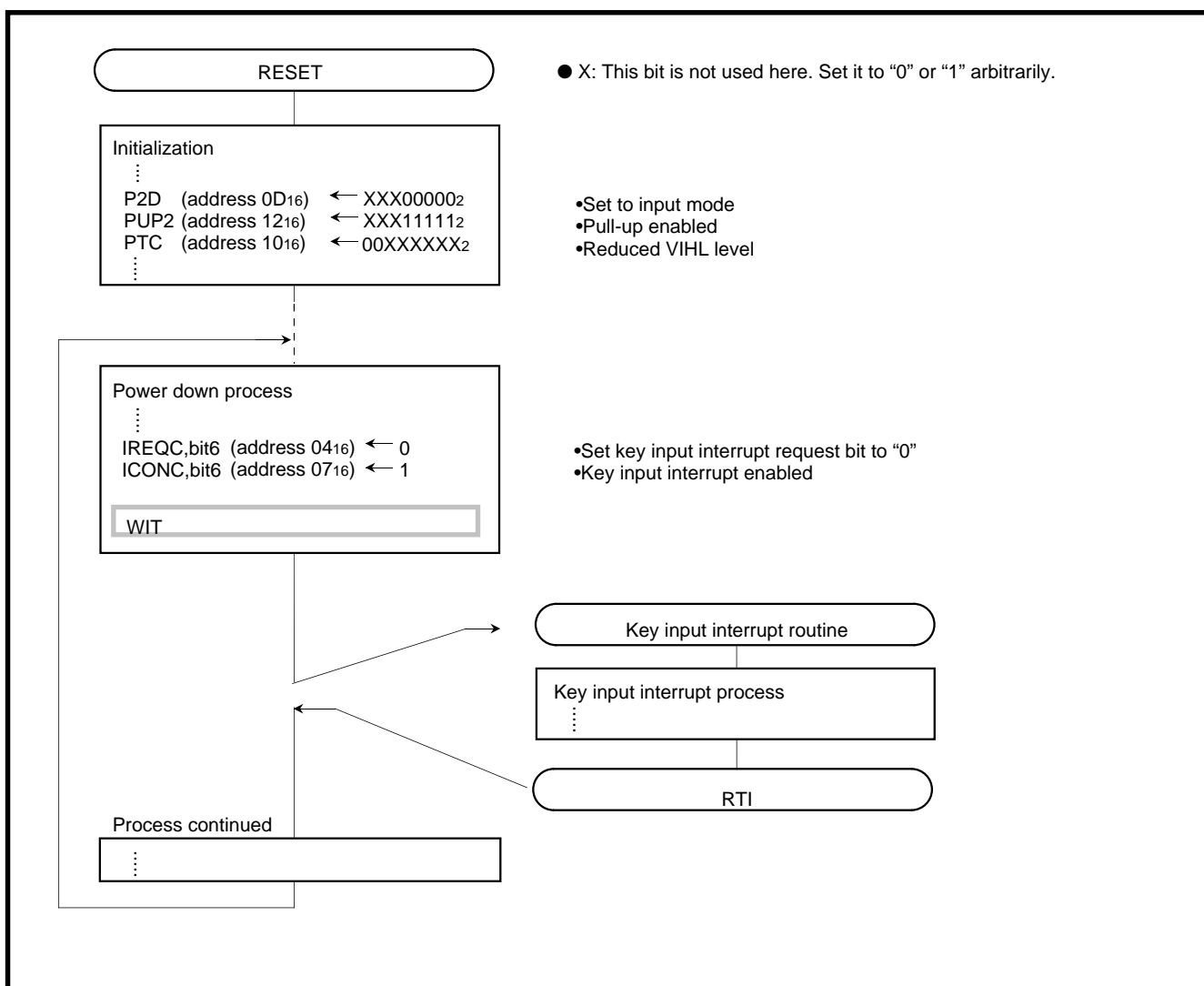


Fig. 2.1.12 Control procedure

### 2.1.4 Terminate unused pins

**Table 2.1.1 Termination of unused pins**

Pins/Ports name	Termination
P0, P1, P2, P3, P4, P5, P6, P7, P8	<ul style="list-style-type: none"> <li>• Set to the input mode and connect each to V<sub>CC</sub> or V<sub>SS</sub> through a resistor of 1 k<math>\Omega</math> to 10 k<math>\Omega</math>.</li> <li>• Set to the output mode and open at “L” or “H” output state.</li> </ul>
HOLD, RDY	Connect to V <sub>CC</sub> through a resistor (pull-up).
CNV <sub>SS</sub>	Connect to V <sub>CC</sub> or V <sub>SS</sub> .
AV <sub>SS</sub>	Connect to V <sub>SS</sub> (GND).
AV <sub>CC</sub>	Connect to V <sub>CC</sub> .
X <sub>OUT</sub>	Open (only when using external clock)
USB D+	Open
USB D-	
Ext. Cap.	Connect to V <sub>CC</sub> (DC-DC converter disabled) when the USB function is not used.

**Note:** This is the termination in case USB is not used.

### 2.1.5 Notes on I/O port

#### (1) Notes in standby state

In standby state\*1 for low-power dissipation, do not make input levels of an I/O port “undefined”. Pull-up (connect the port to VCC) or pull-down (connect the port to VSS) these ports through a resistor.

When determining a resistance value, note the following points:

- External circuit
- Variation of output levels during the ordinary operation

When using built-in pull-up resistor, note on varied current values:

- When setting as an input port : Fix its input level
- When setting as an output port : Prevent current from flowing out to external

#### ● Reason

The potential which is input to the input buffer in a microcomputer is unstable in the state that input levels of an I/O port are “undefined”. This may cause power source current.

\*1 standby state: stop mode by executing **STP** instruction  
wait mode by executing **WIT** instruction

#### (2) Modifying output data with bit managing instruction

When the port latch of an I/O port is modified with the bit managing instruction\*2, the value of the unspecified bit may be changed.

#### ● Reason

The bit managing instructions are read-modify-write form instructions for reading and writing data by a byte unit. Accordingly, when these instructions are executed on a bit of the port latch of an I/O port, the following is executed to all bits of the port latch.

- As for bit which is set for input port:  
The pin state is read in the CPU, and is written to this bit after bit managing.
- As for bit which is set for output port:  
The bit value is read in the CPU, and is written to this bit after bit managing.

Note the following:

- Even when a port which is set as an output port is changed for an input port, its port latch holds the output data.
- As for a bit of which is set for an input port, its value may be changed even when not specified with a bit managing instruction in case where the pin state differs from its port latch contents.

\*2 Bit managing instructions: **SEB** and **CLB** instructions

#### (3) Pull-up control

When using port P2, which includes a pull-up resistor, as an output port, its port pull-up control is invalidated, that is, pull-up cannot be enabled.

#### ● Reason

Pull-up/pull-down control is valid only when each direction register is set to the input mode.

## 2.1.6 Termination of unused pins

### (1) Terminate unused pins

#### ① I/O ports :

- Set the I/O ports for the input mode and connect them to Vcc or Vss through each resistor of 1 k $\Omega$  to 10 k $\Omega$ .

Ports that permit the selecting of a built-in pull-up resistor can also use this resistor. Set the I/O ports for the output mode and open them at "L" or "H".

- When opening them in the output mode, the input mode of the initial status remains until the mode of the ports is switched over to the output mode by the program after reset. Thus, the potential at these pins is undefined and the power source current may increase in the input mode. With regard to an effects on the system, thoroughly perform system evaluation on the user side.
- Since the direction register setup may be changed because of a program runaway or noise, set direction registers by program periodically to increase the reliability of program.

#### ② The AVss pin when not using the A/D converter :

- When not using the A/D converter, handle a power source pin for the A/D converter, AVss pin as follows:

AVss: Connect to the Vss pin.

### (2) Termination remarks

#### ① I/O ports :

Do not open in the input mode.

##### ● Reason

- The power source current may increase depending on the first-stage circuit.
- An effect due to noise may be easily produced as compared with proper termination ② and shown on the above.

#### ② I/O ports :

When setting for the input mode, do not connect to Vcc or Vss directly.

##### ● Reason

If the direction register setup changes for the output mode because of a program runaway or noise, a short circuit may occur between a port and Vcc (or Vss).

#### ③ I/O ports :

When setting for the input mode, do not connect multiple ports in a lump to Vcc or Vss through a resistor.

##### ● Reason

If the direction register setup changes for the output mode because of a program runaway or noise, a short circuit may occur between ports.

- At the termination of unused pins, perform wiring at the shortest possible distance (20 mm or less) from microcomputer pins.

## 2.2 Timer

This paragraph explains the registers setting method and the notes related to the timers.

### 2.2.1 Memory map

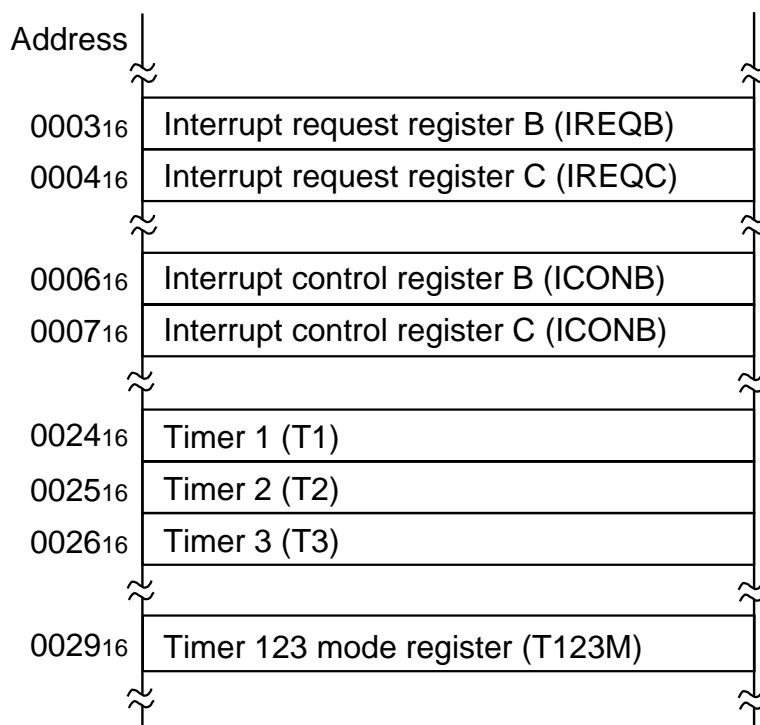


Fig. 2.2.1 Memory map of registers relevant to timers

## 2.2.2 Related registers

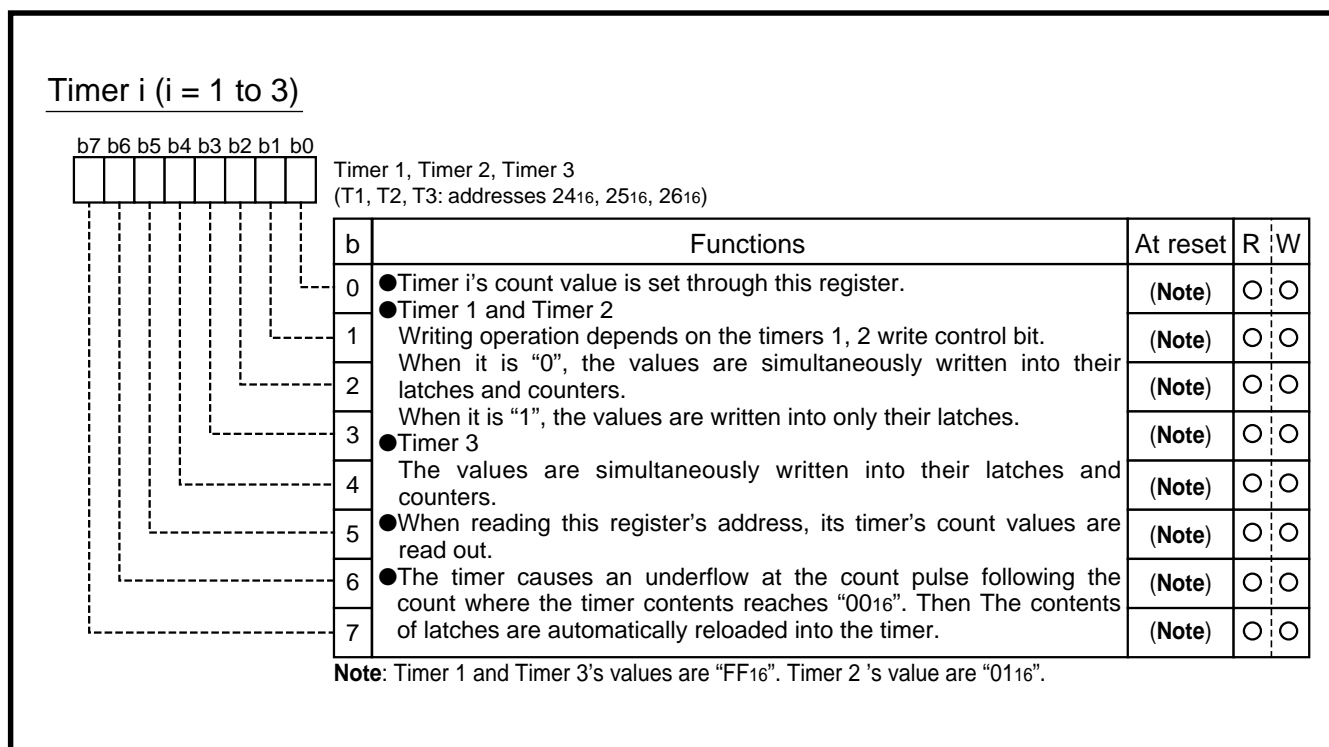


Fig. 2.2.2 Structure of Timer i (i=1, 2, 3)

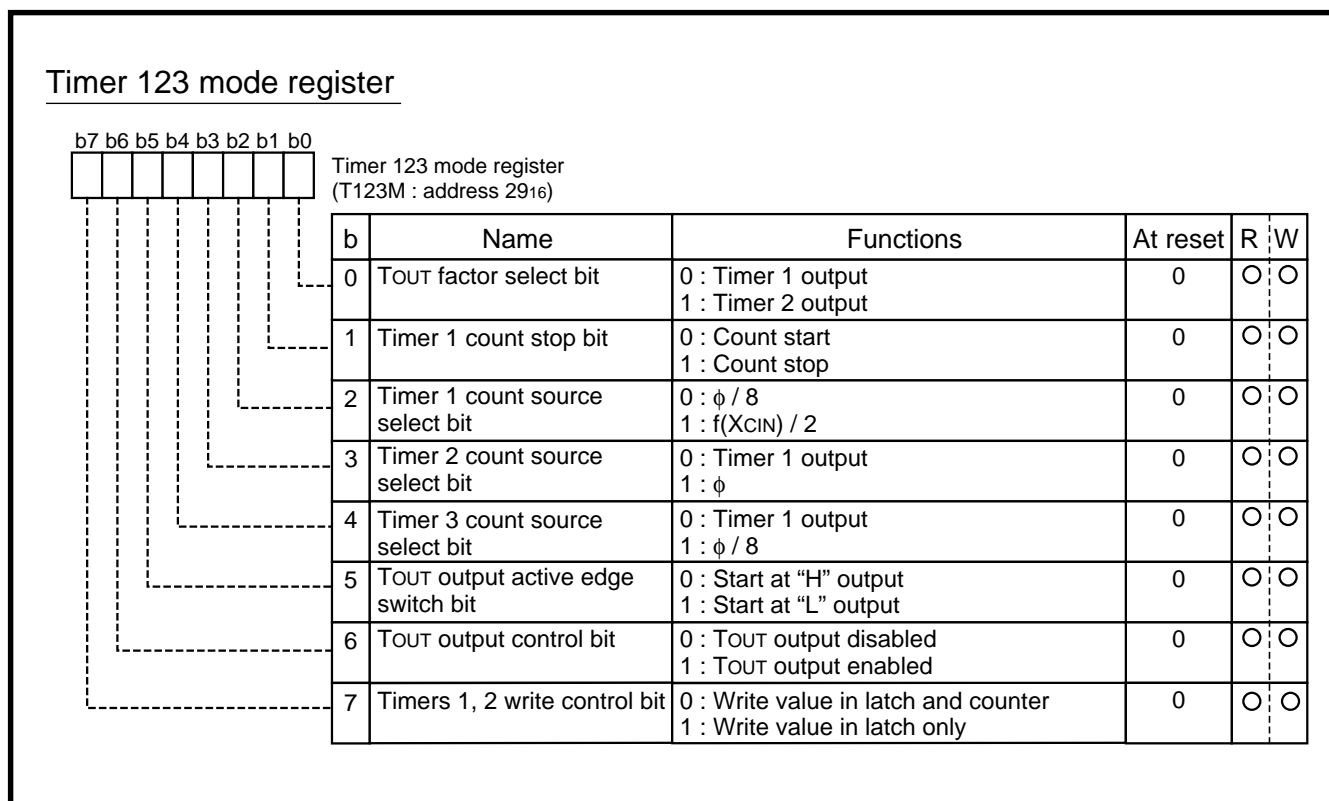


Fig. 2.2.3 Structure of Timer 123 mode register

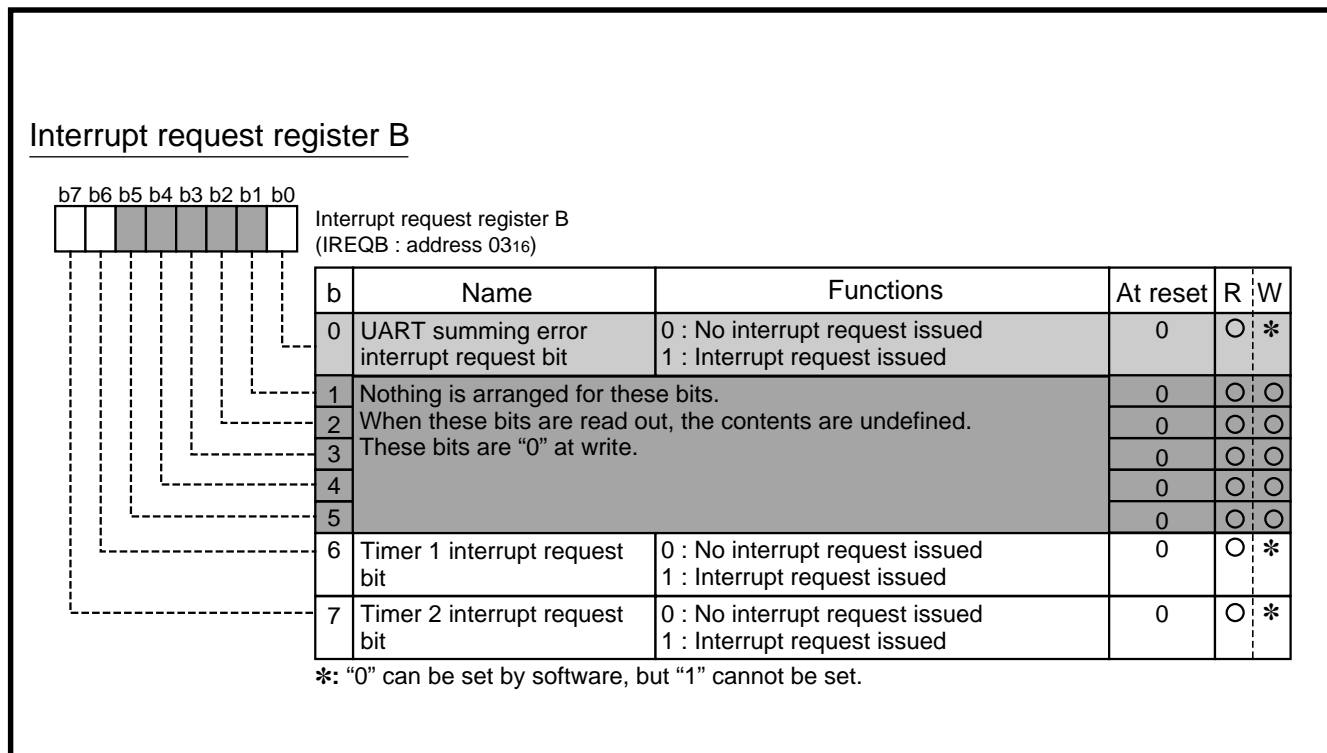


Fig. 2.2.4 Structure of Interrupt request register B

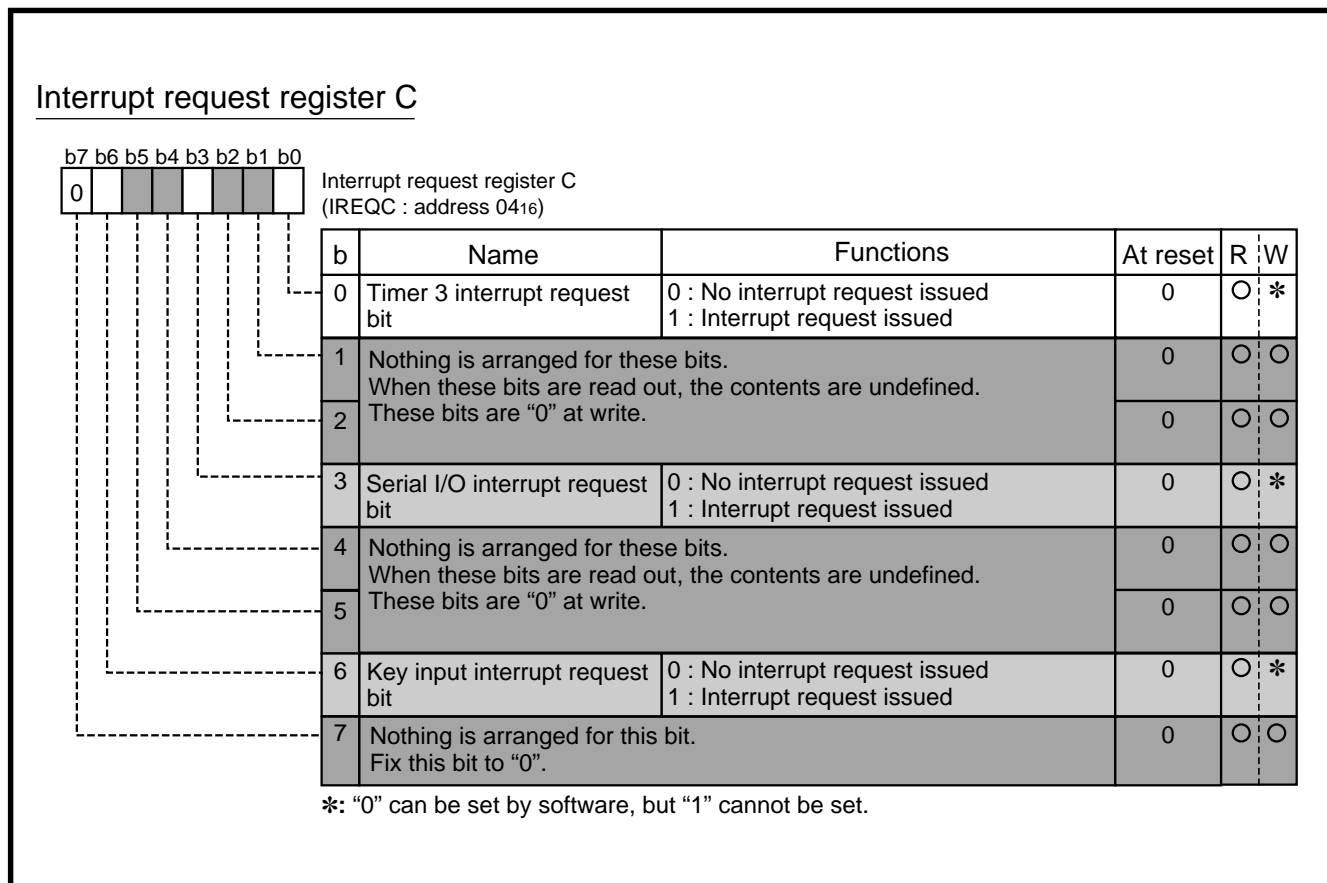


Fig. 2.2.5 Structure of Interrupt request register C

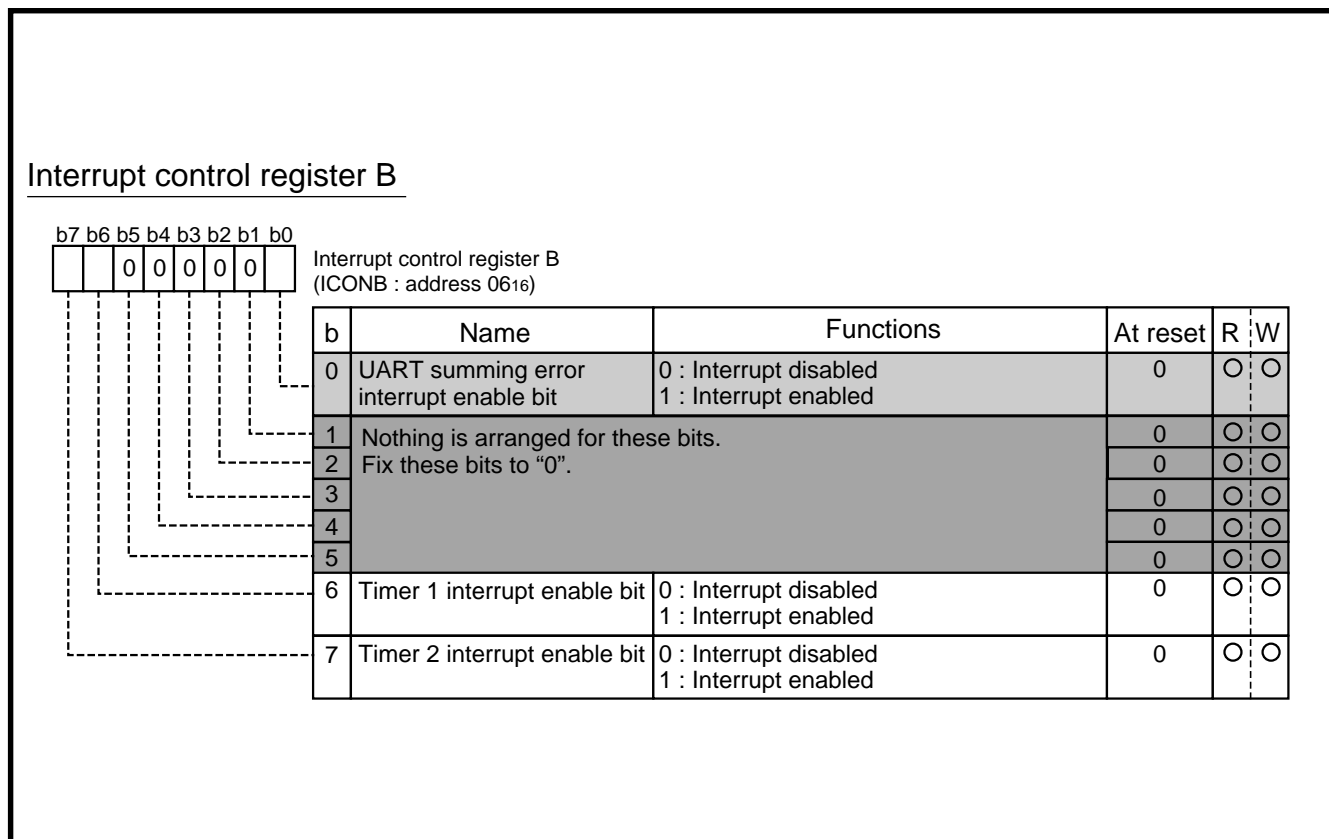


Fig. 2.2.6 Structure of Interrupt control register B

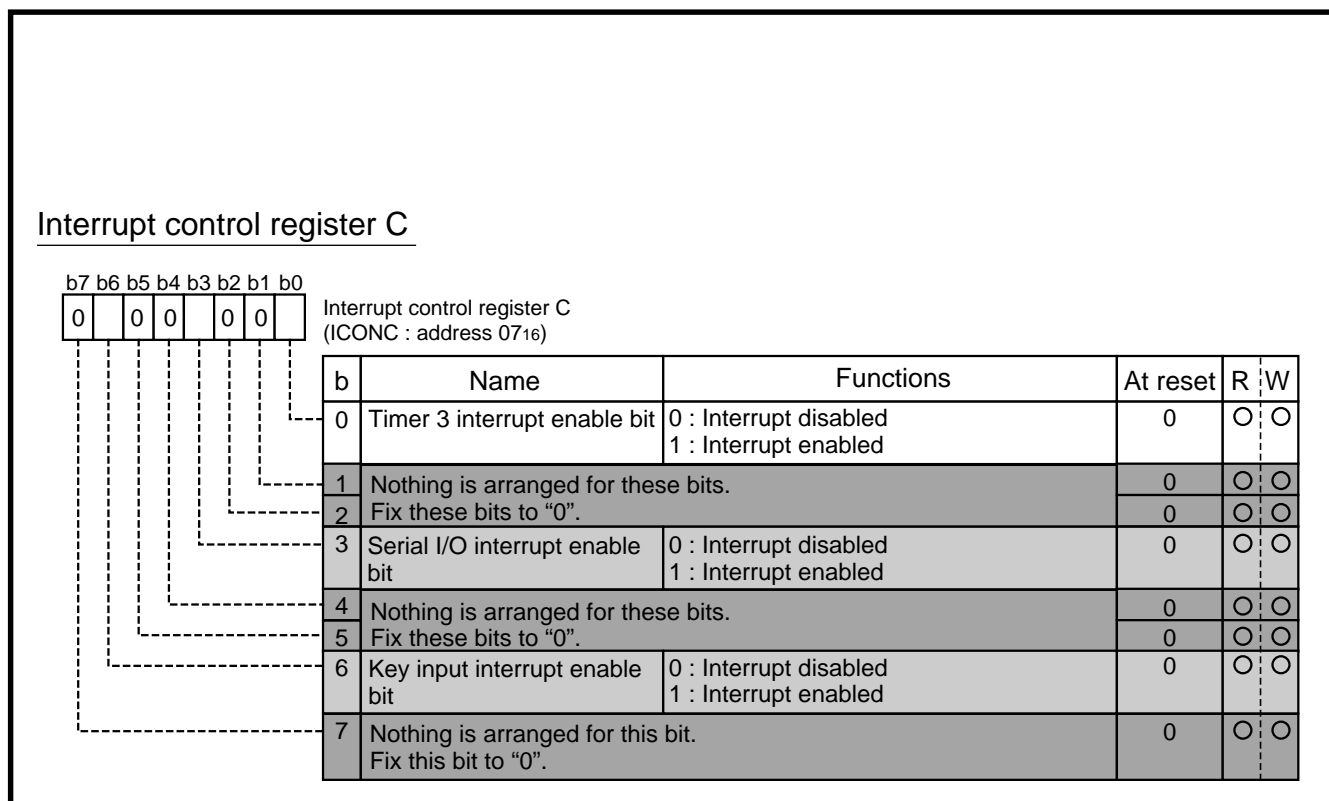


Fig. 2.2.7 Structure of Interrupt control register C



### 2.2.3 Timer application examples

#### (1) Basic functions and uses

##### [Function 1] Control of event interval (Timer 1 to Timer 3)

When a certain time, by setting a count value to each timer, has passed, the timer interrupt request occurs.

<Use>

- Generating of an output signal timing
- Generating of a wait time

##### [Function 2] Control of cyclic operation (Timer 1 to Timer 3)

The value of the timer latch is automatically written to the corresponding timer each time the timer underflows, and each timer interrupt request occurs in cycles.

<Use>

- Generating of cyclic interrupts
- Clock function (measurement of 1 s); see "(2) Timer application example 1"
- Control of a main routine cycle

##### [Function 3] Output of rectangular waveform (Timer 1, Timer 2)

The output levels of the T<sub>OUT</sub> pin is inverted each time the timer underflows.

<Use>

- Piezoelectric buzzer output; see "(3) Timer application example 2"
- Generating of the remote control carrier waveforms

#### (2) Timer application example 1: Clock function (measurement of 1 s)

**Outline:** The input clock is divided by the timer so that the clock can count up at 1 s intervals.

**Specifications:** •The clock  $f(X_{CIN}) = 32 \text{ kHz}$  is divided by the timer.

- The timer 2 interrupt request bit is checked in main routine, and if the interrupt request is issued, the clock is counted up.
- The timer 1 interrupt occurs every 10 ms to execute processing of other interrupts.

Figure 2.2.8 shows the timers connection and setting of division ratios; Figure 2.2.9 shows the related registers setting; Figure 2.2.10 shows the control procedure.

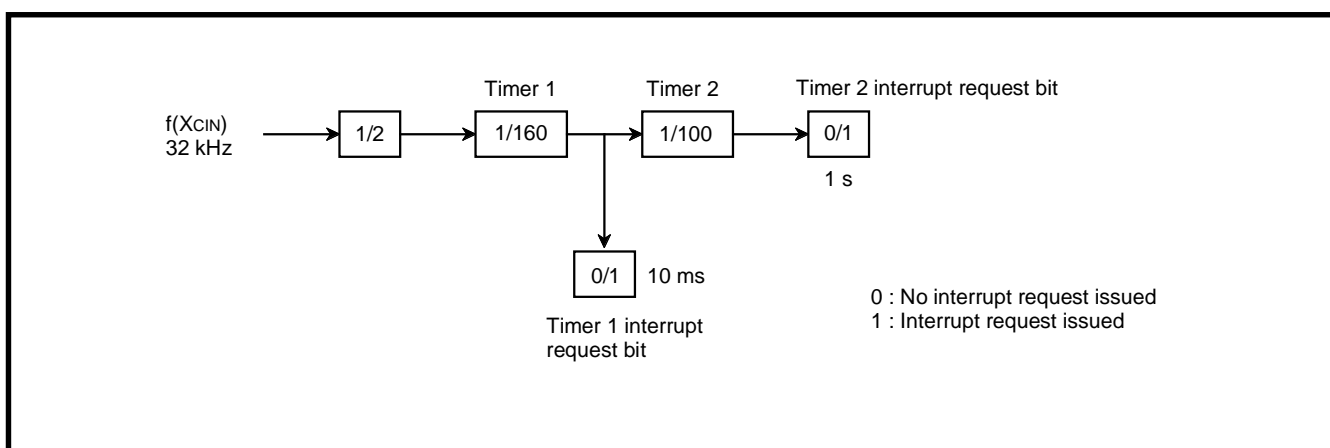


Fig. 2.2.8 Timers connection and setting of division ratios

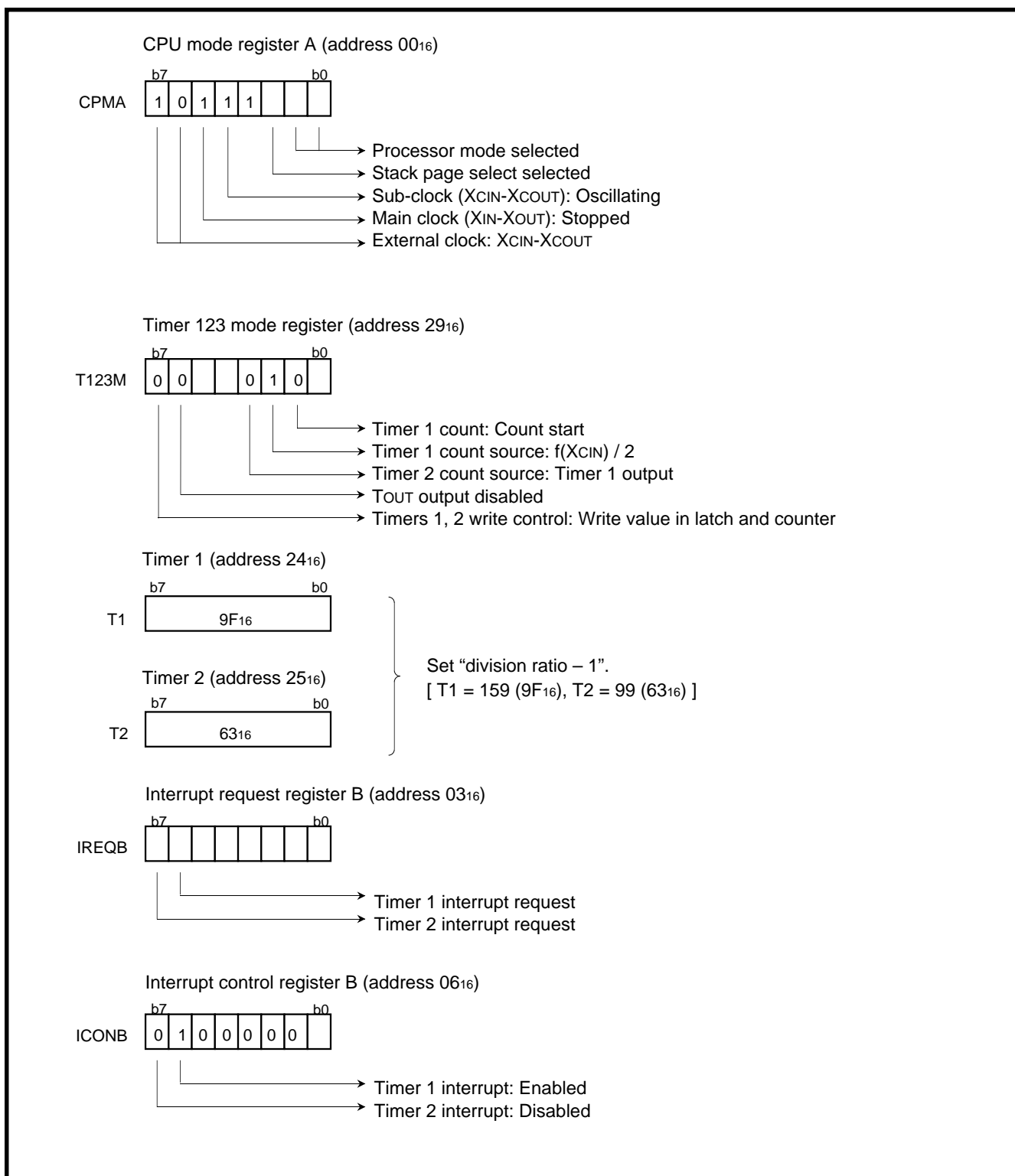


Fig. 2.2.9 Related registers setting

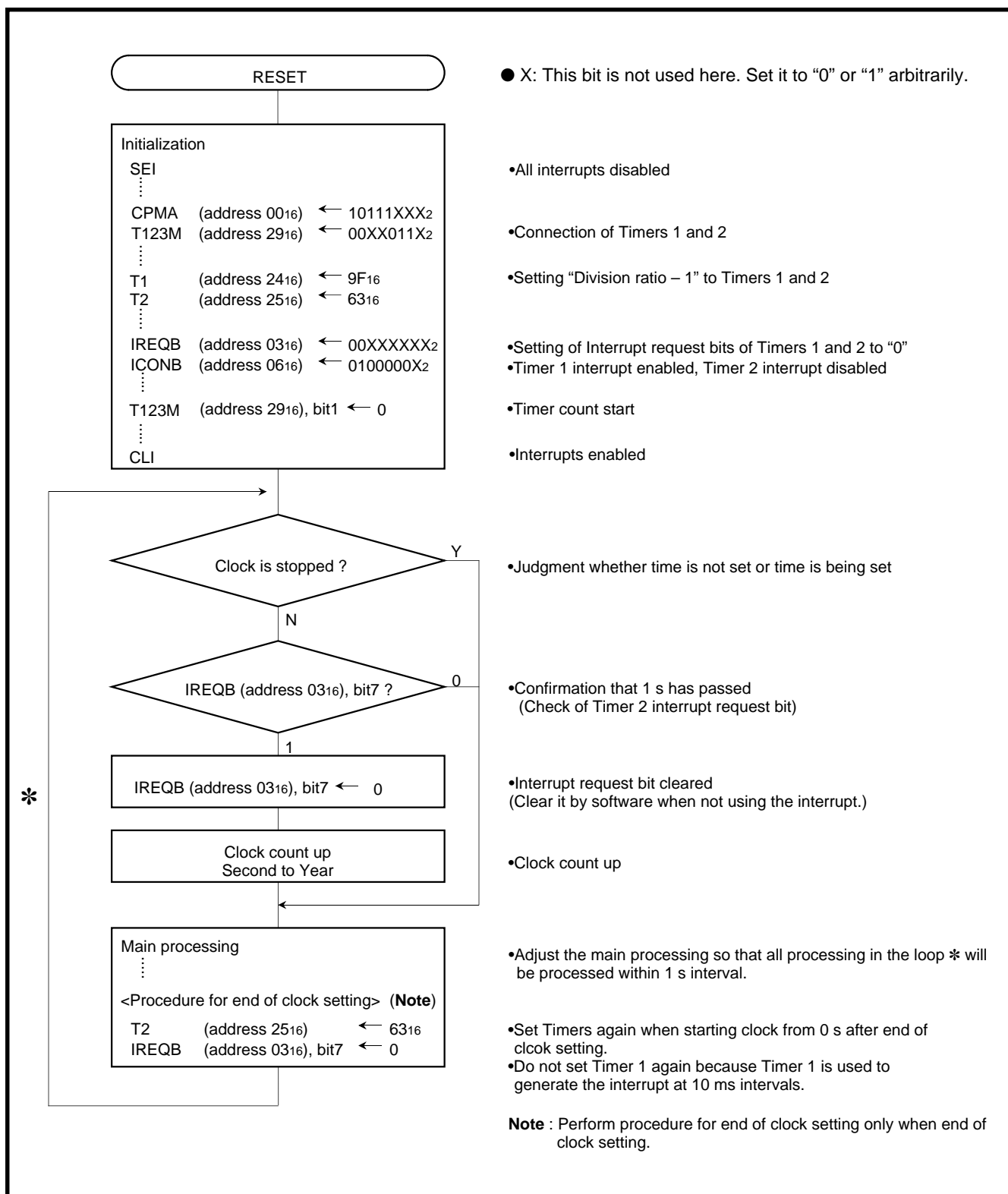


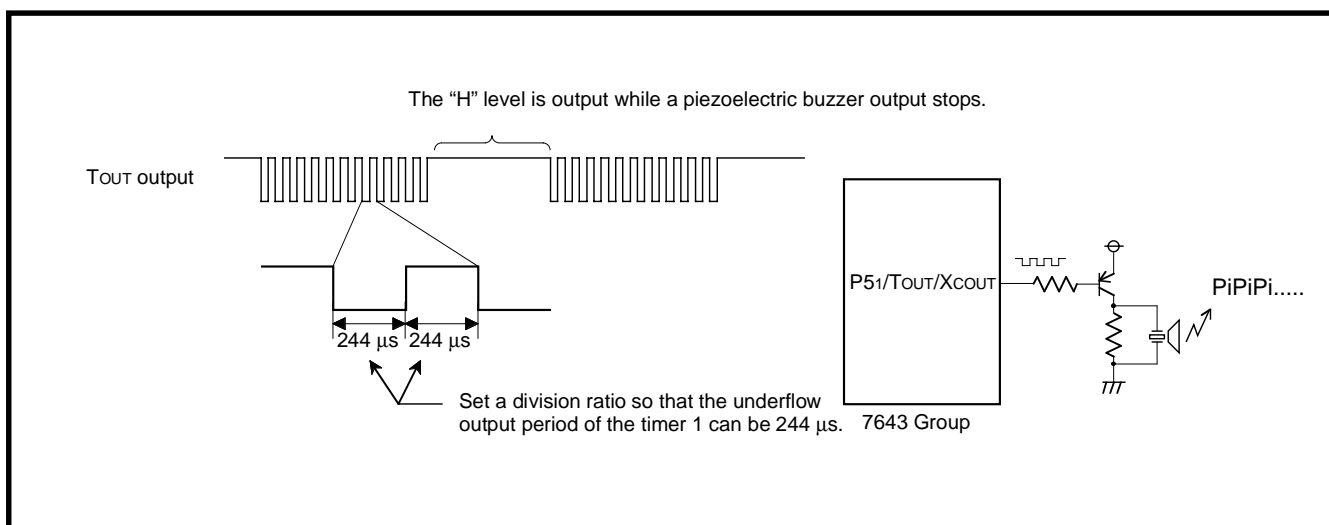
Fig. 2.2.10 Control procedure

**(3) Timer application example 2: Piezoelectric buzzer output**

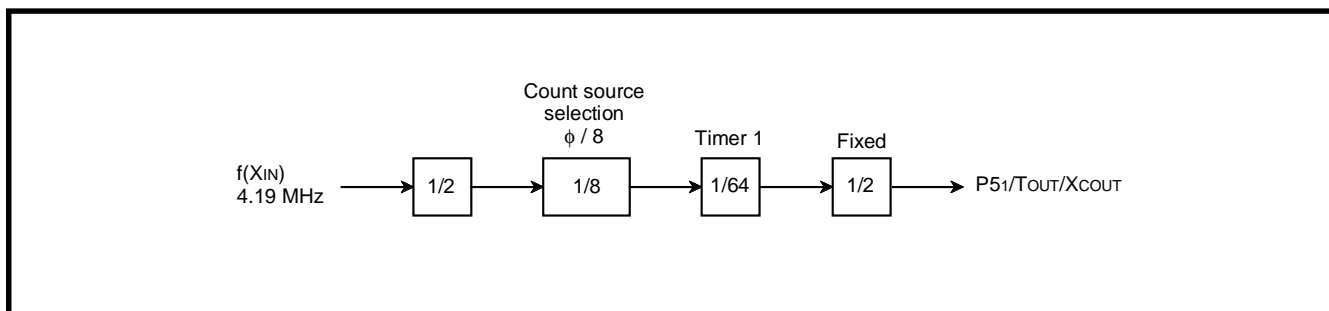
**Outline:** The rectangular waveform output function of the timer is applied for a piezoelectric buzzer output.

- Specifications:**
- The rectangular waveform, dividing the clock  $f(X_{IN}) = 4.19 \text{ MHz}$  ( $2^{22} \text{ Hz}$ ) into about 2 kHz (2048 Hz), is output from the P51/TOUT/XCOUT pin.
  - The level of the P51/TOUT/XCOUT pin is fixed to "H" while a piezoelectric buzzer output stops.

Figure 2.2.11 shows a peripheral circuit example, and Figure 2.2.12 shows the timers connection and setting of division ratios. Figure 2.2.13 shows the related registers setting, and Figure 2.2.14 shows the control procedure.



**Fig. 2.2.11 Peripheral circuit example**



**Fig. 2.2.12 Timers connection and setting of division ratios**

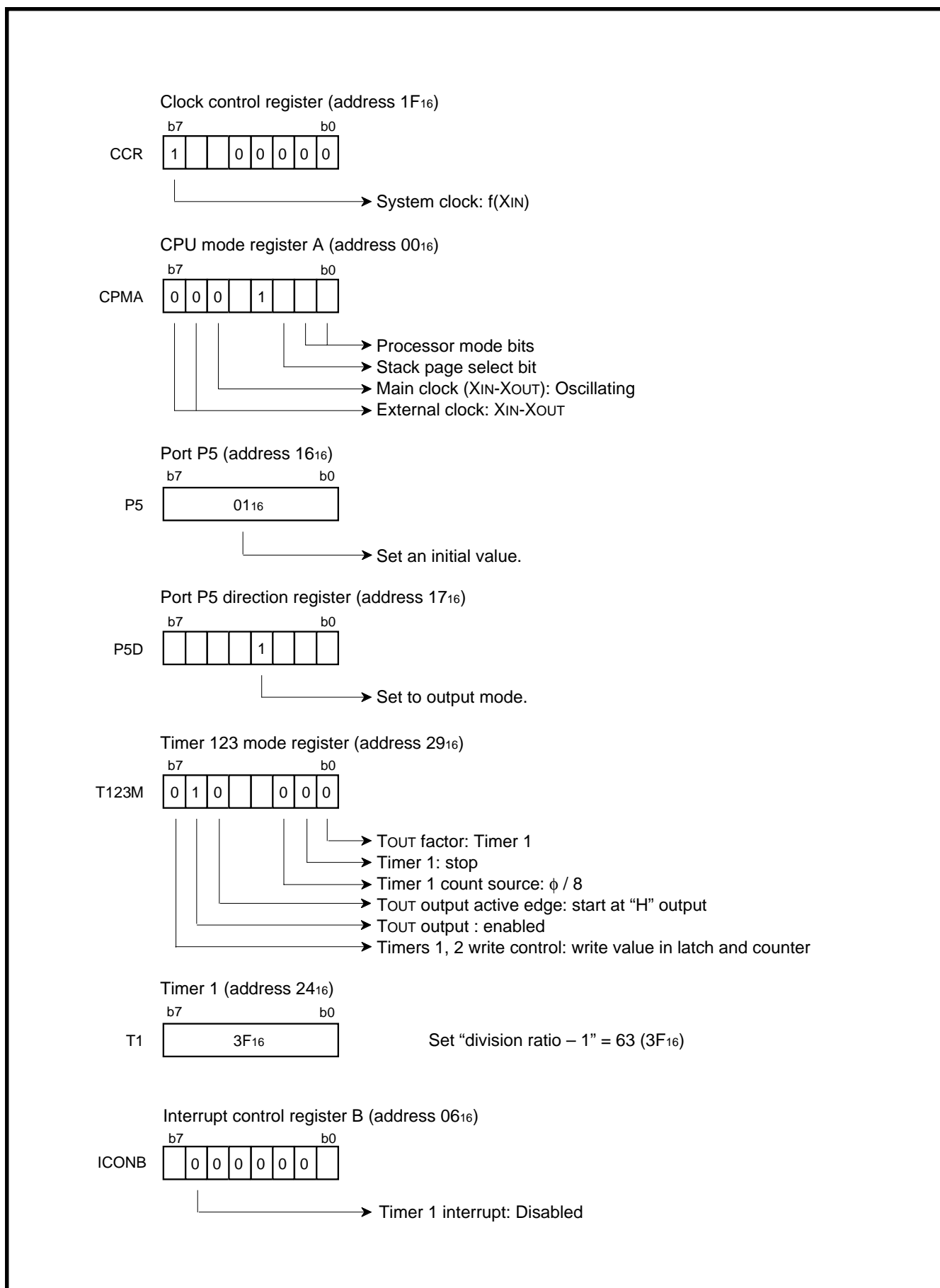


Fig. 2.2.13 Relevant registers setting

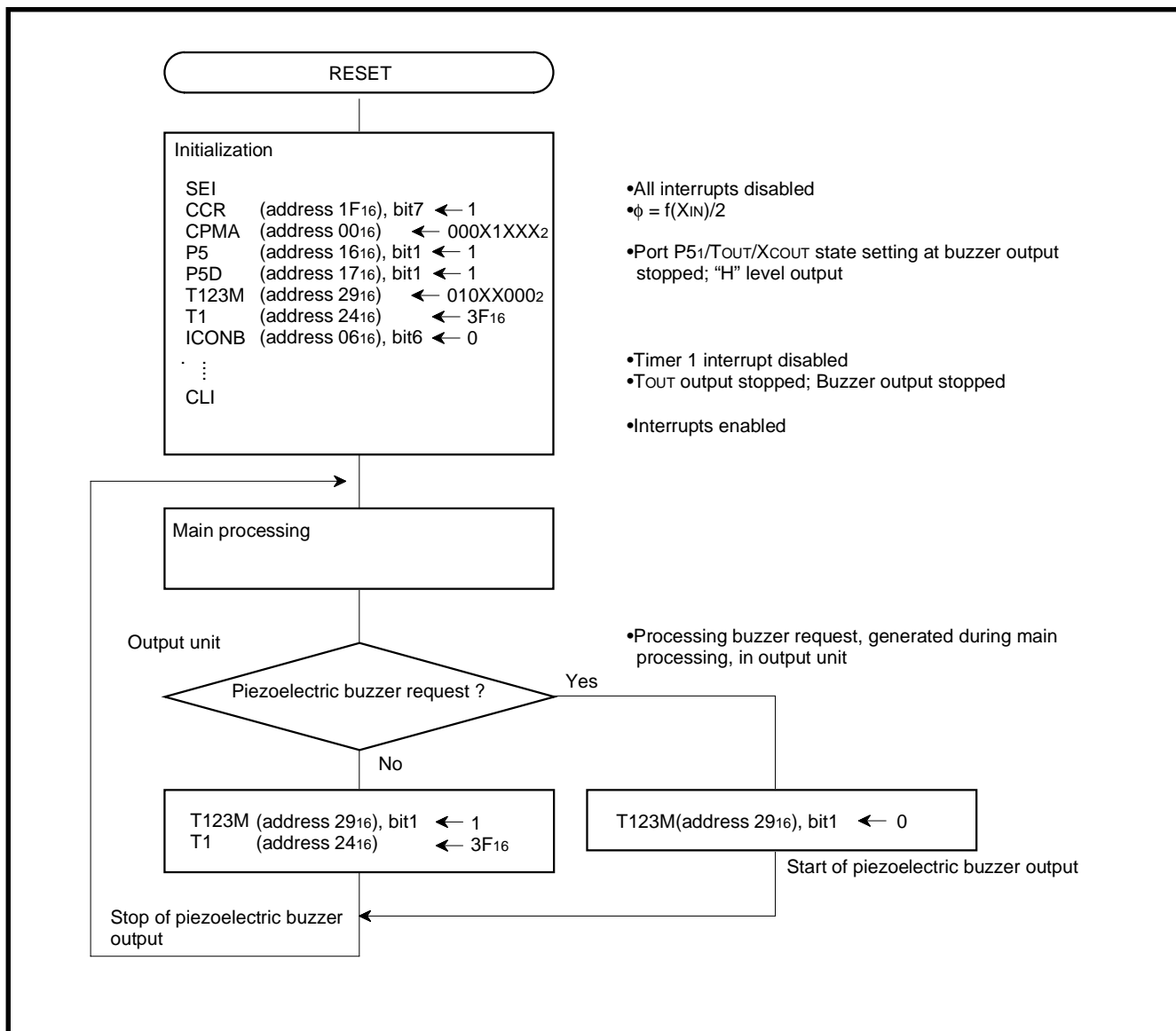


Fig. 2.2.14 Control procedure

### 2.2.4 Notes on timer

#### (1) Read/Write for timer

- The timer division ratio is :  $1 / (n + 1)$   
( $n = "0"$  to  $"255"$  written into the timer)
- When the value is loaded only in the latch, the value is loaded in the timer at the count pulse following the count where the timer reaches  $"00_{16}"$ .
- In the timers 1 to 3, switching of the count sources of timers 1 to 3 does not affect the values of reload latches. However, that may make count operation started. Therefore, write values again in the order of timers 1, 2 and then timer 3 after their count sources have been switched.
- The timer current count value can be read out by reading the timer.

#### (2) Pulse output

- When using the  $T_{OUT}$  output of timer 1 or timer 2, set bit 1 of port P5 direction register to  $"1"$  (output mode).
- The  $T_{OUT}$  output pin is shared with the  $X_{COUT}$  pin. Accordingly, when using  $f(X_{CIN})/2$  as the timer 1 count source (bit 2 of timer 123 mode register =  $"0"$ ),  $X_{COUT}$  oscillation drive must be disabled (bit 5 of clock control register =  $"1"$ ) to input clocks from the  $X_{CIN}$  pin.
- The  $P5_1/X_{COUT}/T_{OUT}$  pin cannot function as an ordinary I/O port while  $X_{CIN}-X_{COUT}$  is oscillating. When  $X_{CIN}-X_{COUT}$  oscillation is stopped or  $X_{COUT}$  oscillation drive is disabled, this can be used as the  $T_{OUT}$  output pin of timer 1 or 2.

#### (3) STP instruction

When the reset or STP instruction is being executed,  $"01_{16}"$  is set to the timer 2 and timer 2 latch, and  $"FF_{16}"$  is set to the timer 1 and timer 1 latch, and the timer 1 output is forcibly set as the timer 2 count source. Also, all bits except bit 4 of the timer 123 mode register (address  $0029_{16}$ ) are initialized to  $"0"$ . After returning from Stop Mode, reset the timer 1 (address  $0024_{16}$ ), timer 2 (address  $0025_{16}$ ), and the timer 123 mode register (address  $0029_{16}$ ).

## 2.3 Serial I/O

This paragraph explains the registers setting method and the notes related to the serial I/O.

### 2.3.1 Memory map

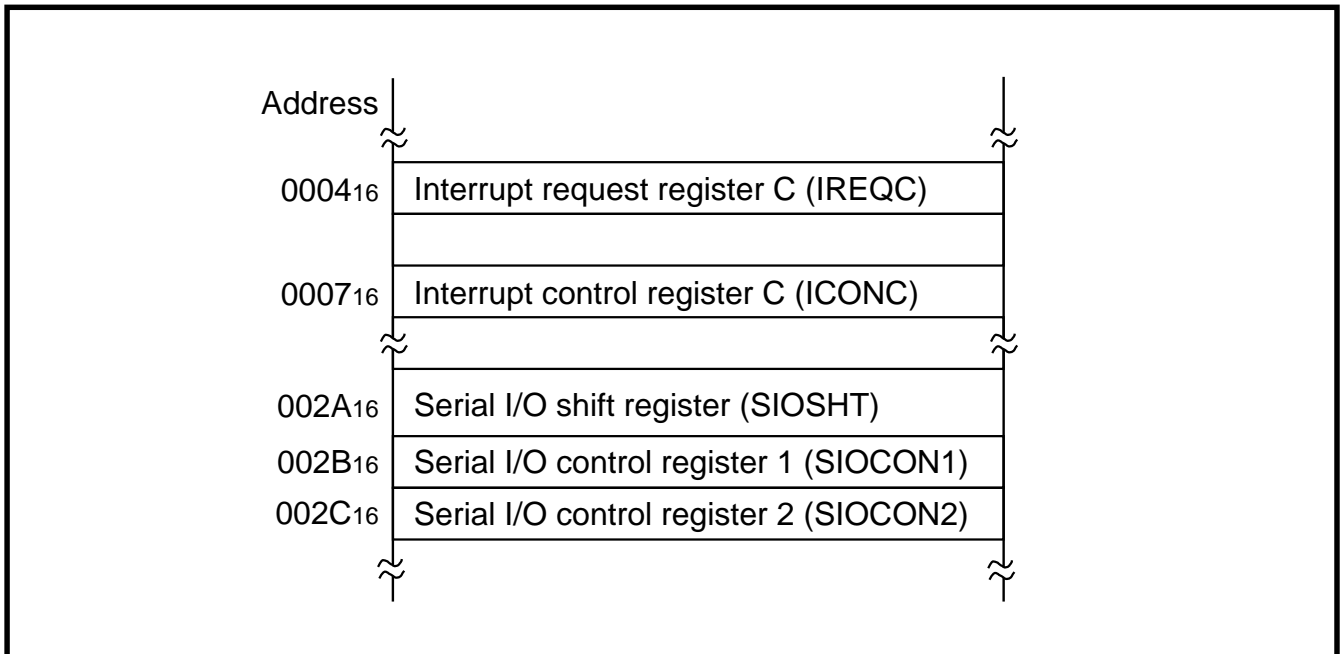


Fig. 2.3.1 Memory map of registers related to serial I/O



### 2.3.2 Related registers

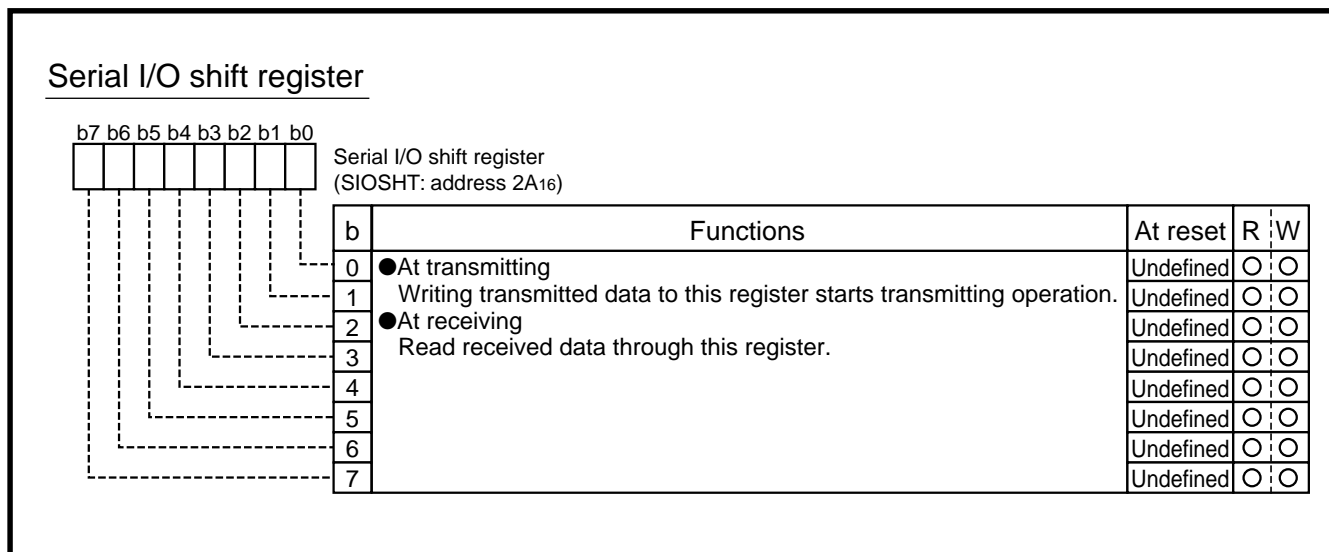


Fig. 2.3.2 Structure of Serial I/O shift register

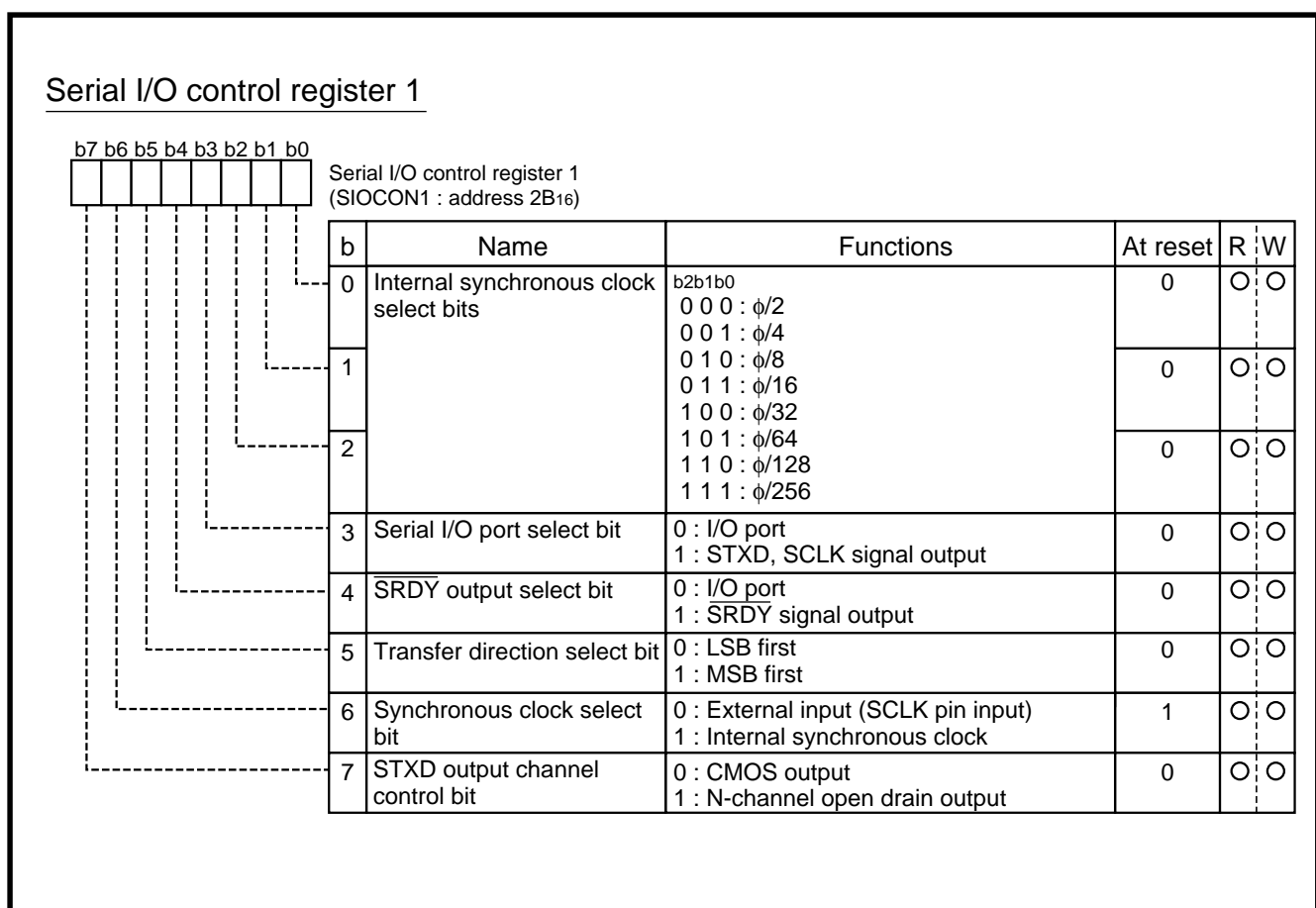
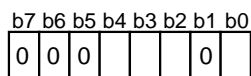


Fig. 2.3.3 Structure of Serial I/O control register 1

## Serial I/O control register 2



Serial I/O control register 2  
(SIOCON2 : address 2C16)

b	Name	Functions	At reset	R	W
0	SPI mode select bit	0 : Normal serial I/O mode 1 : SPI compatible mode ( <b>Note</b> )	0	○	○
1	Nothing is arranged for this bit. Fix this bit to "0".		0	○	○
2	SRXD input enable bit	0 : SRXD input disabled 1 : SRXD input enabled	0	○	○
3	Clock polarity select bit (CPoL)	0 : SCLK starting at "L" 1 : SCLK starting at "H"	1	○	○
4	Clock phase select bit (CPha)	0 : Serial transfer starting at falling edge of SRDY 1 : Serial transfer starting after a half cycle of SCLK passed at falling edge of SRDY	1	○	○
5	Nothing is arranged for these bits.		0	○	○
6	Fix these bits to "0".		0	○	○
7			0	○	○

**Note:** To set the slave mode, also set bit 4 of serial I/O control register 1 to "1".

Fig. 2.3.4 Structure of Serial I/O control register 2

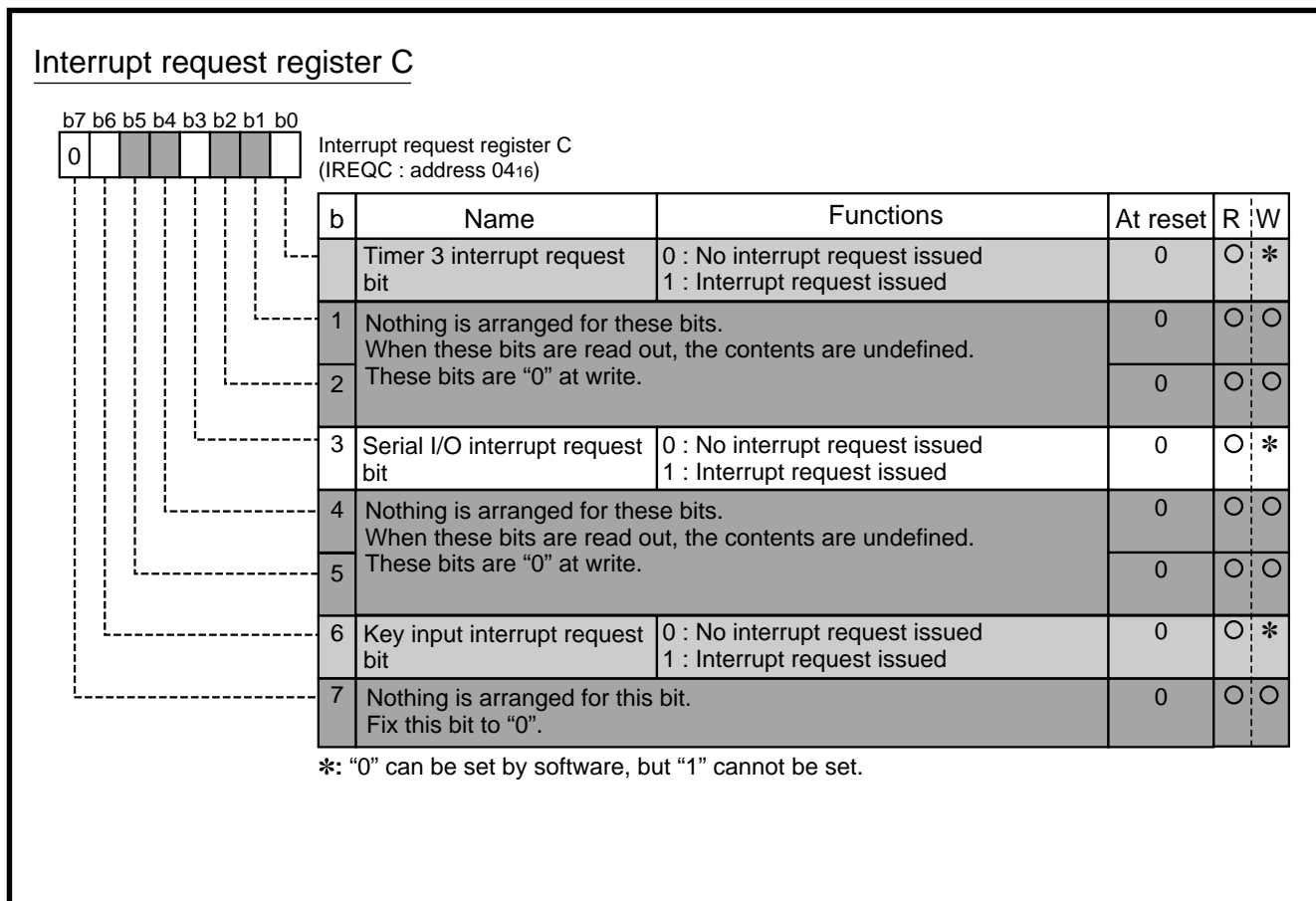


Fig. 2.3.5 Structure of Interrupt request register C

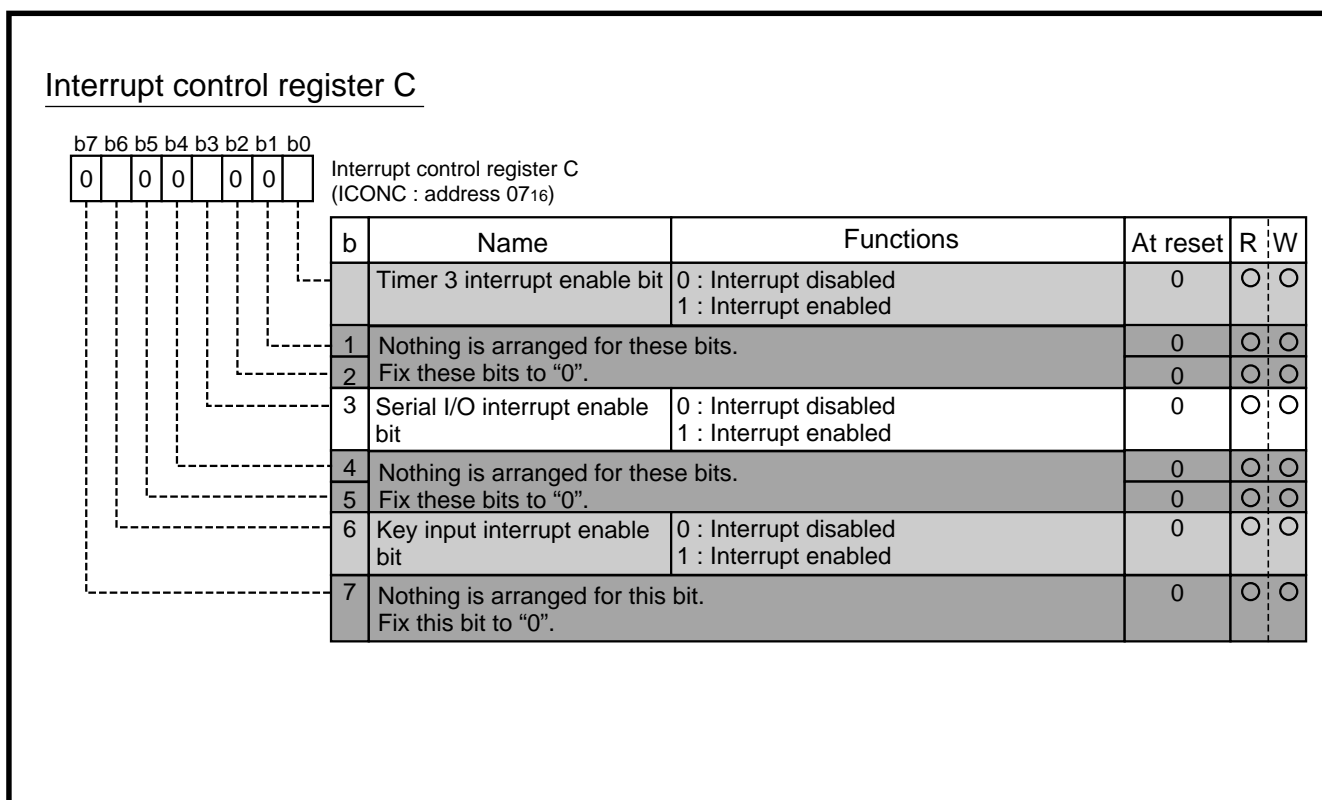


Fig. 2.3.6 Structure of Interrupt control register C

### 2.3.3 Serial I/O connection examples

#### (1) Control of peripheral IC equipped with CS pin

Figure 2.3.7 shows connection examples of a peripheral IC equipped with the CS pin.

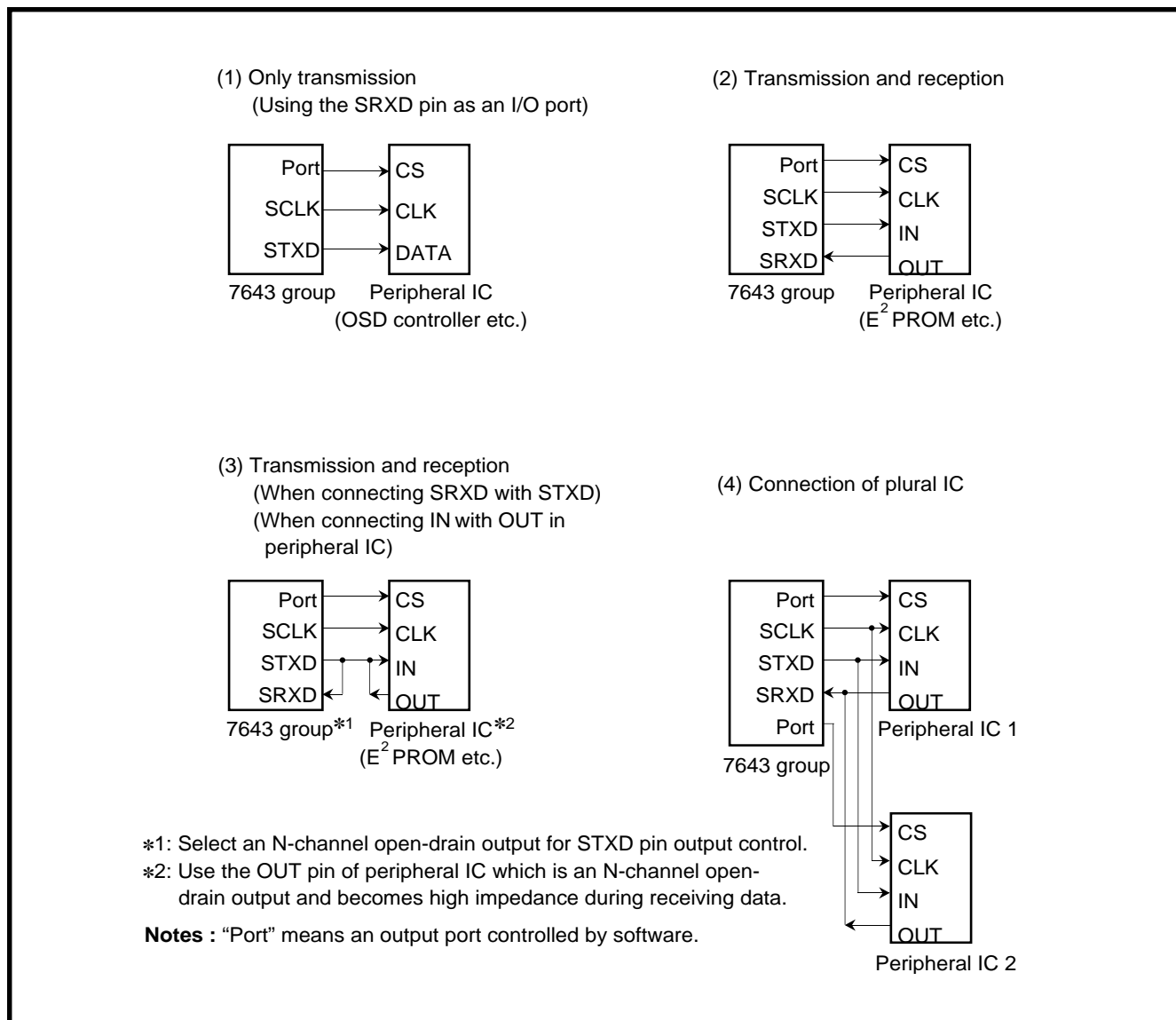
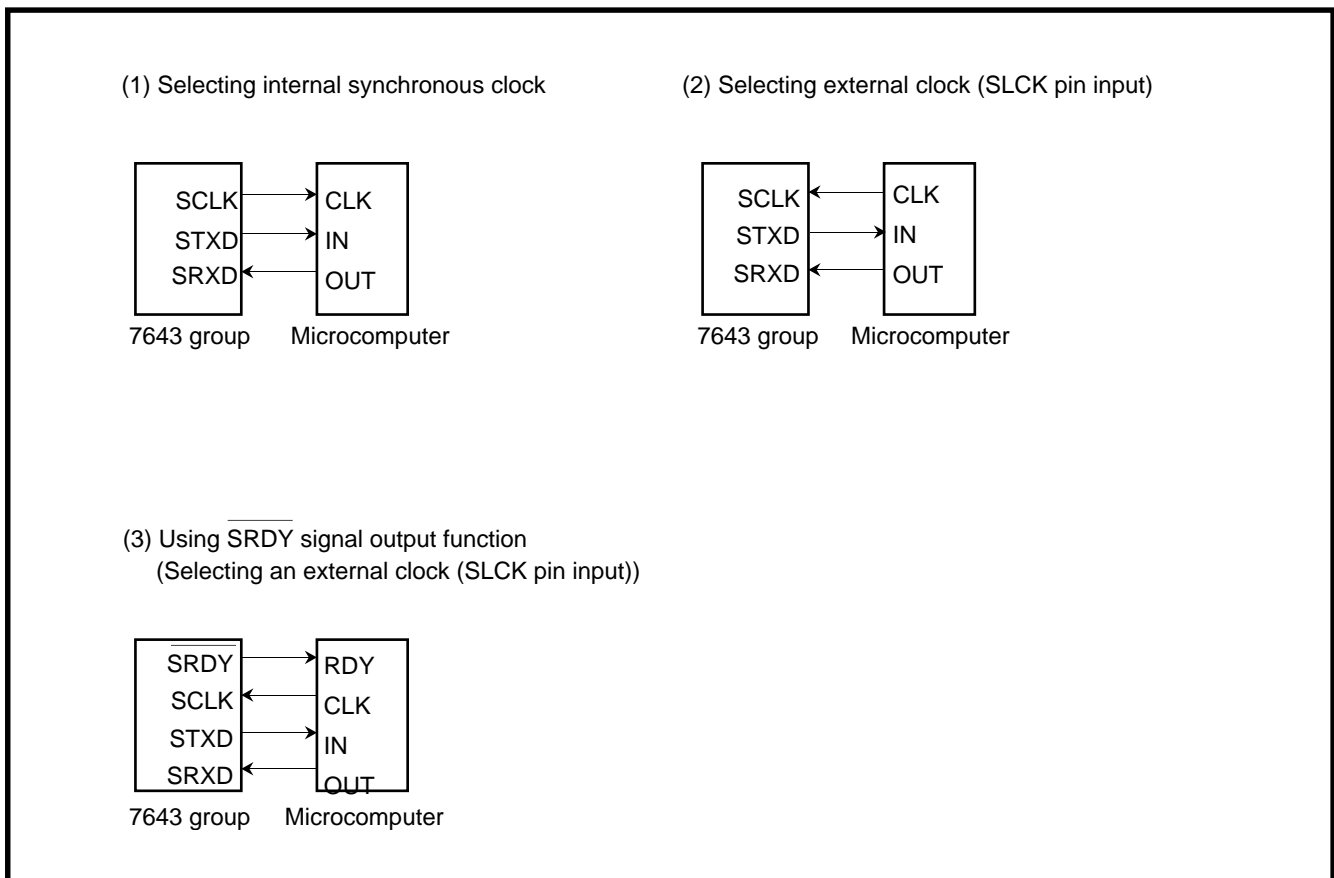


Fig. 2.3.7 Serial I/O connection examples (1)

**(2) Connection with microcomputer**

Figure 2.3.8 shows connection examples with another microcomputer.



**Fig. 2.3.8 Serial I/O connection examples (2)**

### 2.3.4 Serial I/O application example

#### (1) Output of serial data (control of peripheral IC)

**Outline :** Serial communication is performed, connecting port  $\overline{CS}$  pin of peripheral IC. To perform reception, it needs to write dummy data into serial I/O shift register.

Figure 2.3.9 shows a connection diagram, and Figure 2.3.10 shows a timing chart.

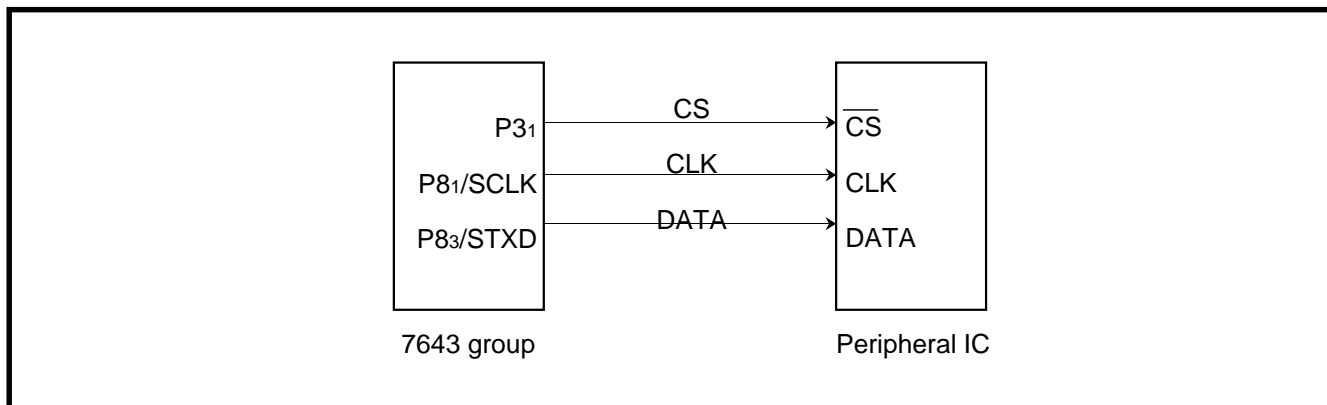


Fig. 2.3.9 Connection diagram

- Specifications :**
- Synchronous clock frequency : 187.5 kHz ( $f(X_{IN}) = 24 \text{ MHz}$  )
  - Transfer direction : LSB first
  - Serial I/O interrupt is not used.
  - Port P3<sub>i</sub> is connected to the  $\overline{CS}$  pin ("L" active) of the peripheral IC for transmission control; the output level of port P3<sub>i</sub> is controlled by software.

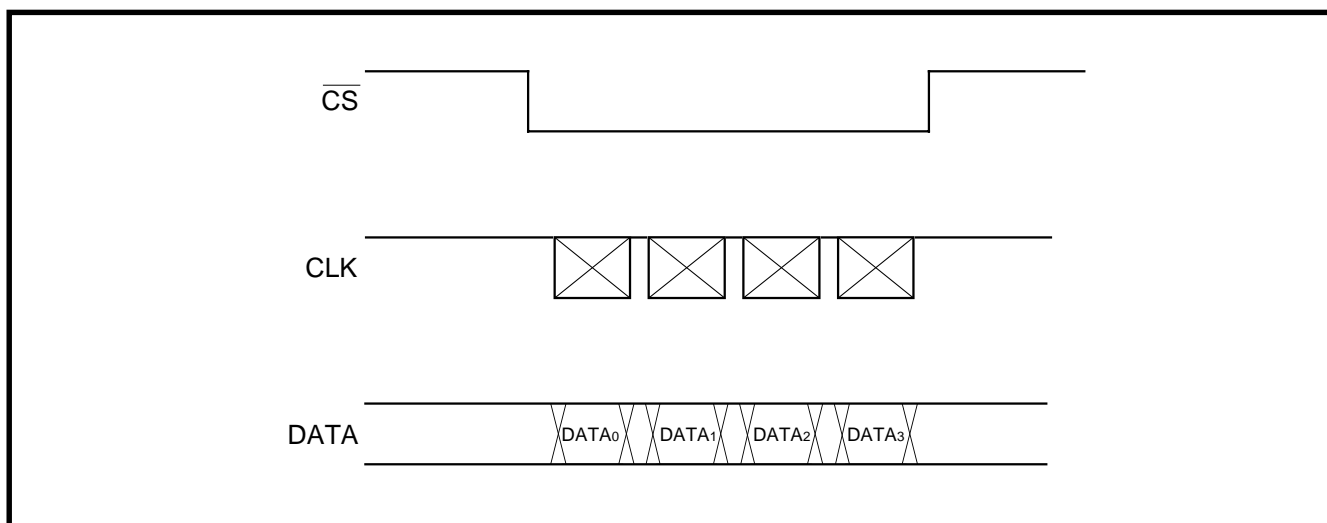


Fig. 2.3.10 Timing chart

Figure 2.3.11 shows the registers setting for the transmitter, and Figure 2.3.12 shows a setting of serial I/O transmission data.

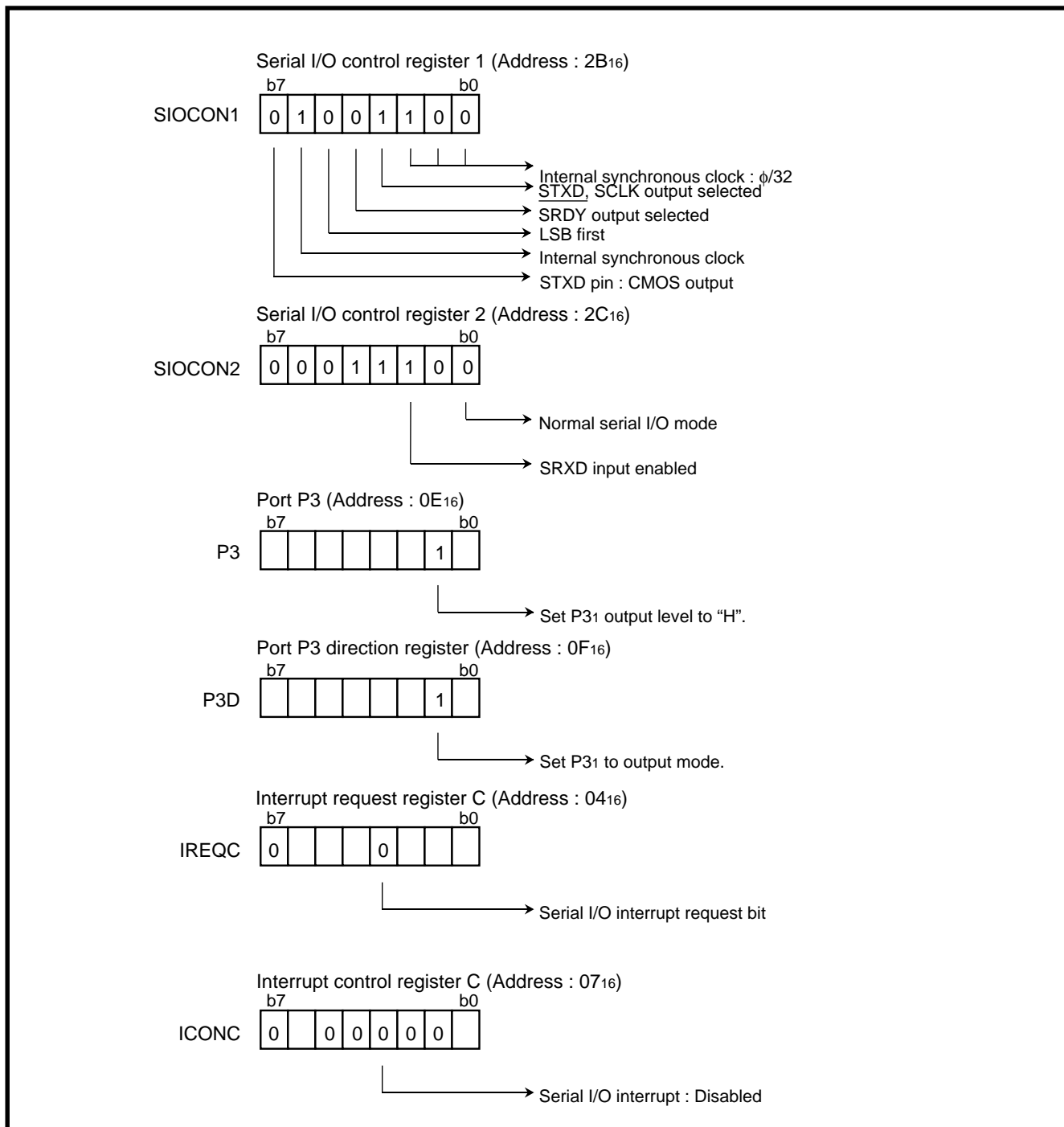


Fig. 2.3.11 Registers setting for transmitter

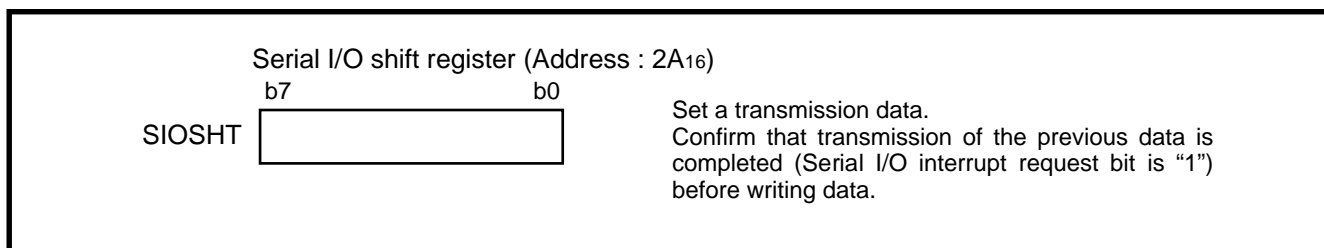


Fig. 2.3.12 Setting of serial I/O transmission data

When the registers are set as shown in Figure 2.3.13, the serial I/O can transmit 1-byte data by writing data into the serial I/O shift register.

Thus, after setting the  $\overline{CS}$  signal to "L", write the transmission data to the serial I/O shift register by each 1 byte, and return the  $\overline{CS}$  signal to "H" when all required data have been transmitted.

Figure 2.3.13 shows a control procedure of transmitter.

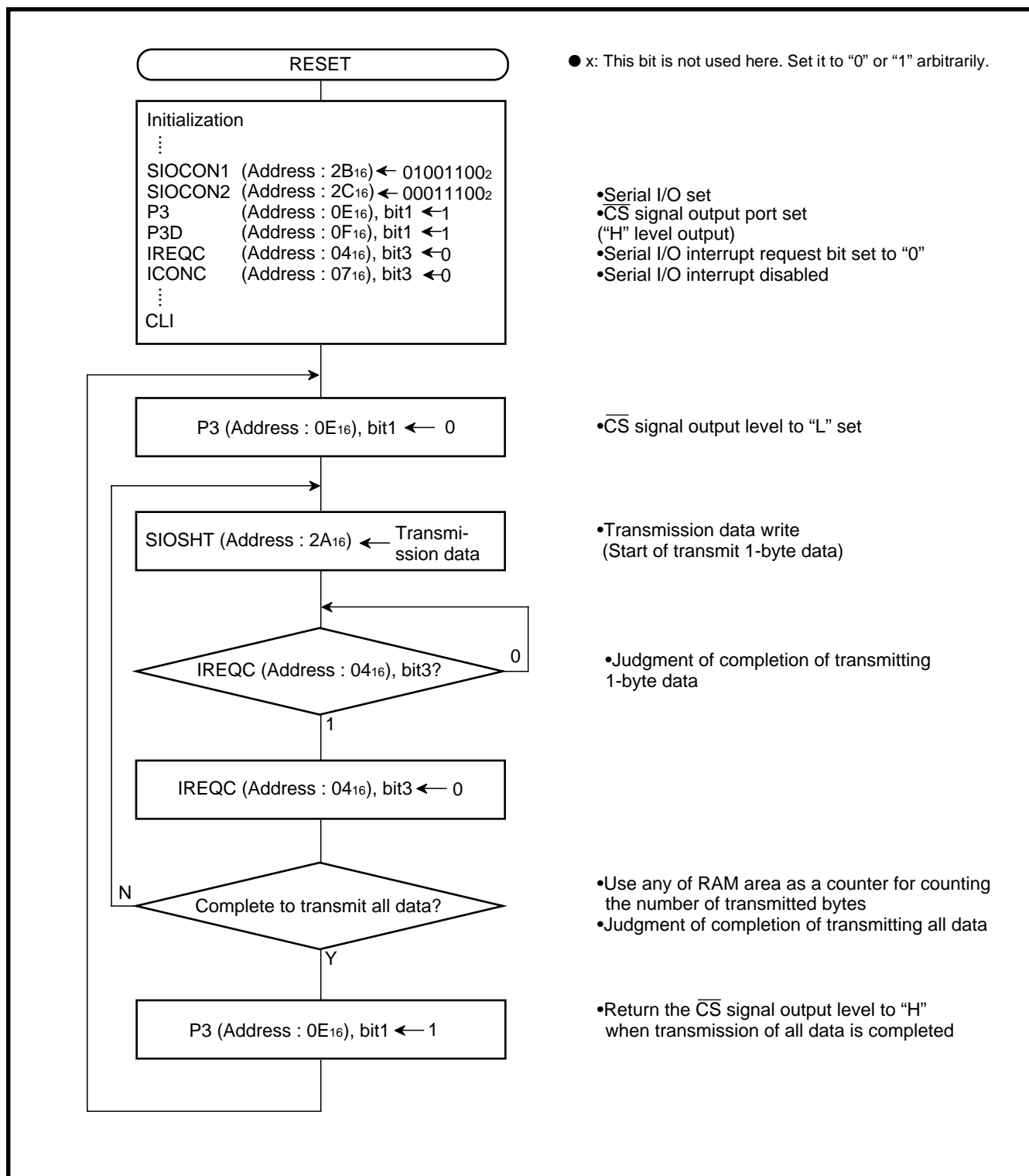


Fig. 2.3.13 Control procedure of transmitter



## (2) Serial communication using SPI compatible mode

### ●Explanation of SPI compatible mode

Setting the SPI mode select bit (bit 0 of SIOCON2) to "1" puts the serial I/O in SPI compatible mode. The synchronous clock select bit (bit 6 of SIOCON1) determines whether the serial I/O is an SPI master or slave. When the external clock (SCLK pin input) is selected ("0"), the serial I/O is in slave mode; when the internal synchronous clock is selected ("1"), the serial I/O is in master mode.

In SPI compatible mode the SRXD pin functions as a MISO (Master In/Slave Out) pin and the STXD pin functions as a MOSI (Master Out/Slave In) pin.

In slave mode the transmit data is output from the MISO pin and the receive data is input from the MISO pin. The SRDY pin functions as the chip-select signal input pin from an external.

In master mode the transmit data is output from the MOSI pin and the receive data is input from the MISO pin. The SRDY pin functions as the chip-select signal output pin to an external.

### ●Slave mode operation

In slave mode of SPI compatible mode 4 types of clock polarity and clock phase can be usable by bits 3 and 4 of serial I/O control register 2.

If the SRDY pin is held "H", the shift clock is inhibited, the serial I/O counter is set to "7". If the SRDY pin is held "L", then the shift clock will start.

Make sure during transfer to maintain the SRDY input at "L" and not to write data to the serial I/O counter.

**Outline :** Serial communication is performed between 7643 group MCUs, using SPI compatible mode.

**Specifications :**

- Synchronous clock frequency : 187.5 kHz ( $f(X_{IN}) = 24 \text{ MHz}$  )
- Transfer direction : LSB first

Figure 2.3.14 shows a connection diagram; Figure 2.3.15 shows the registers setting for SPI compatible mode; Figures 2.3.16 and 2.3.17 show a control procedure of SPI compatible mode.

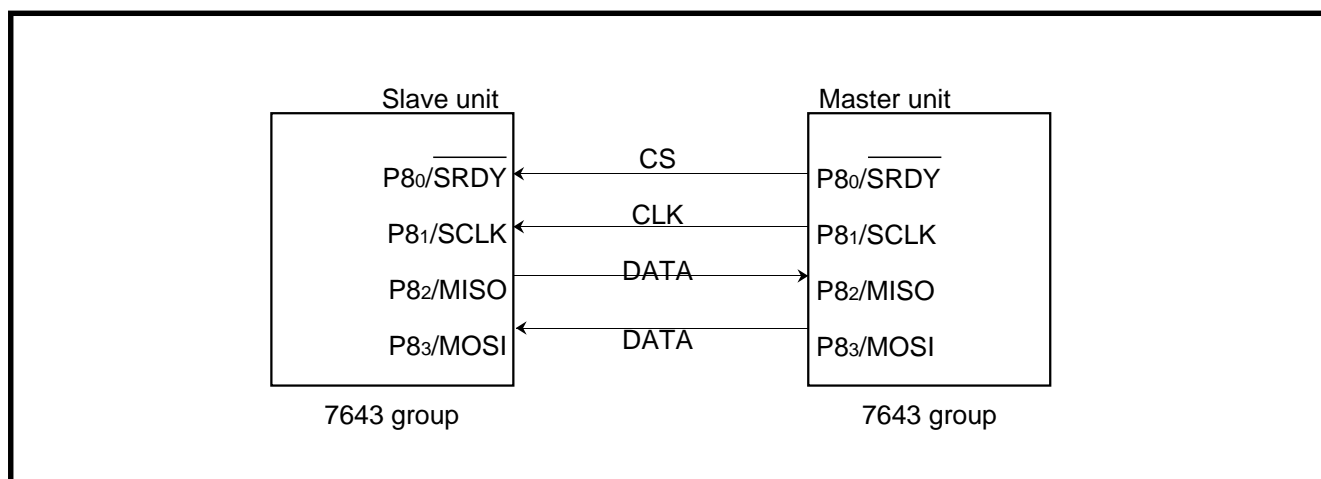
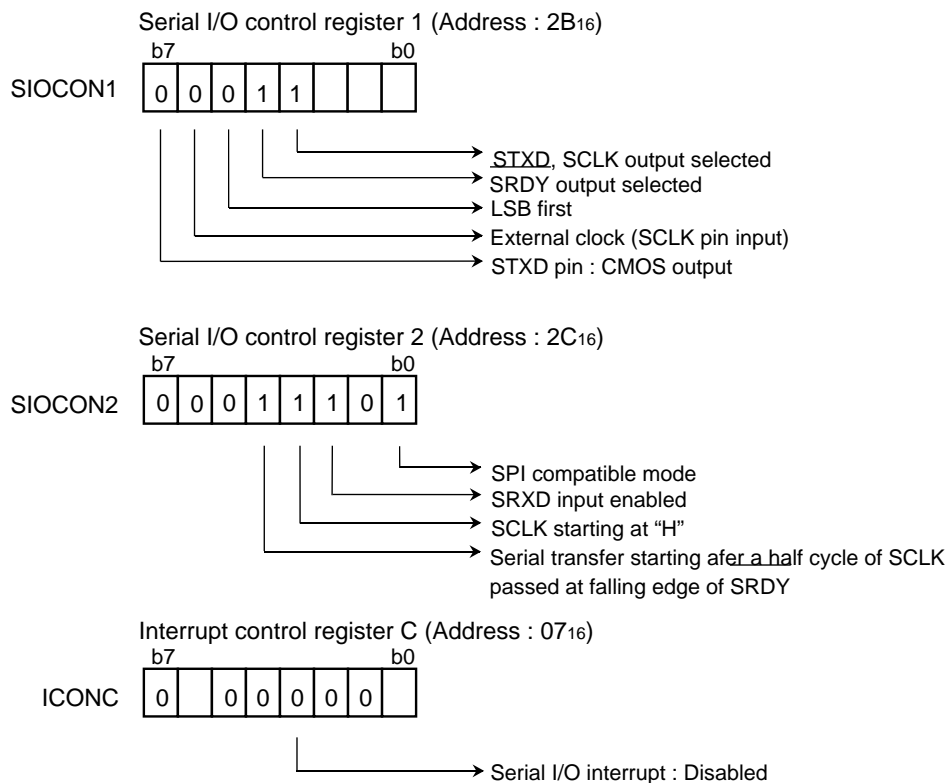


Fig. 2.3.14 Connection diagram

### ●Slave Unit



### ●Master Unit

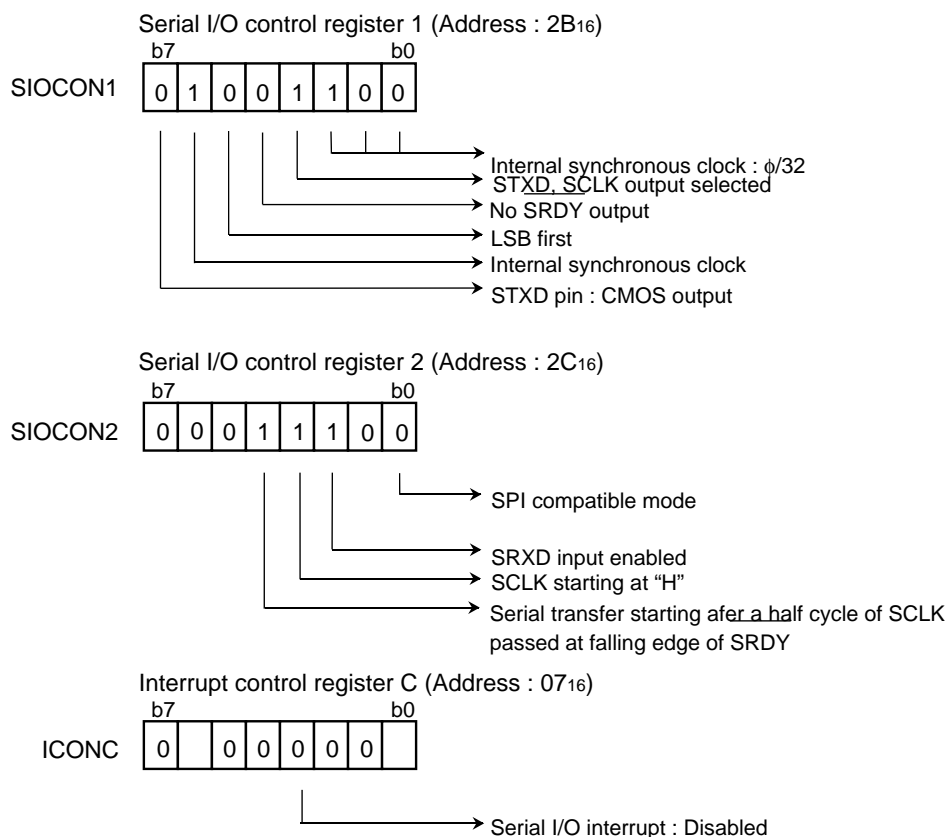


Fig. 2.3.15 Registers setting for SPI compatible mode

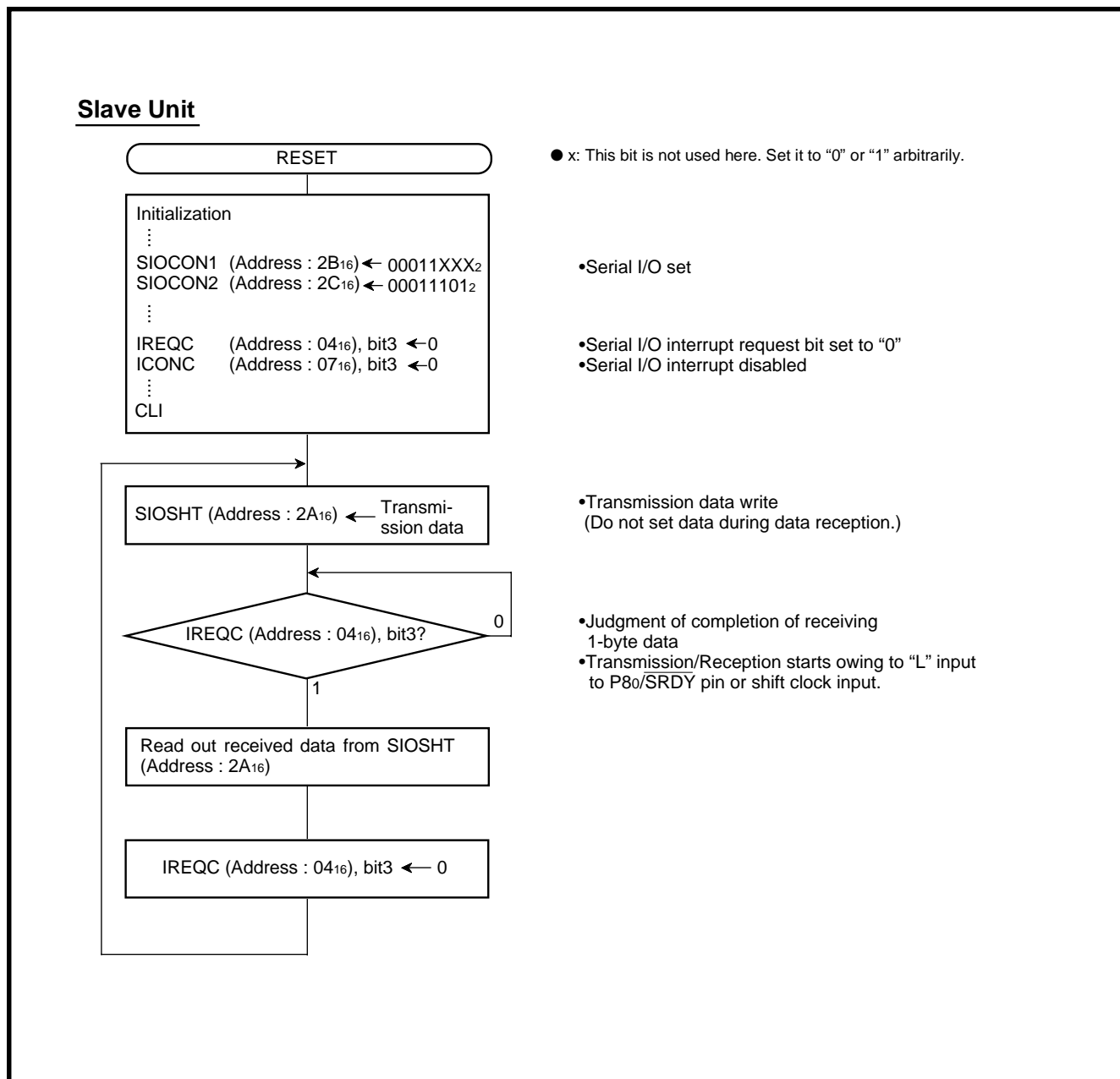


Fig. 2.3.16 Control procedure of SPI compatible mode in slave

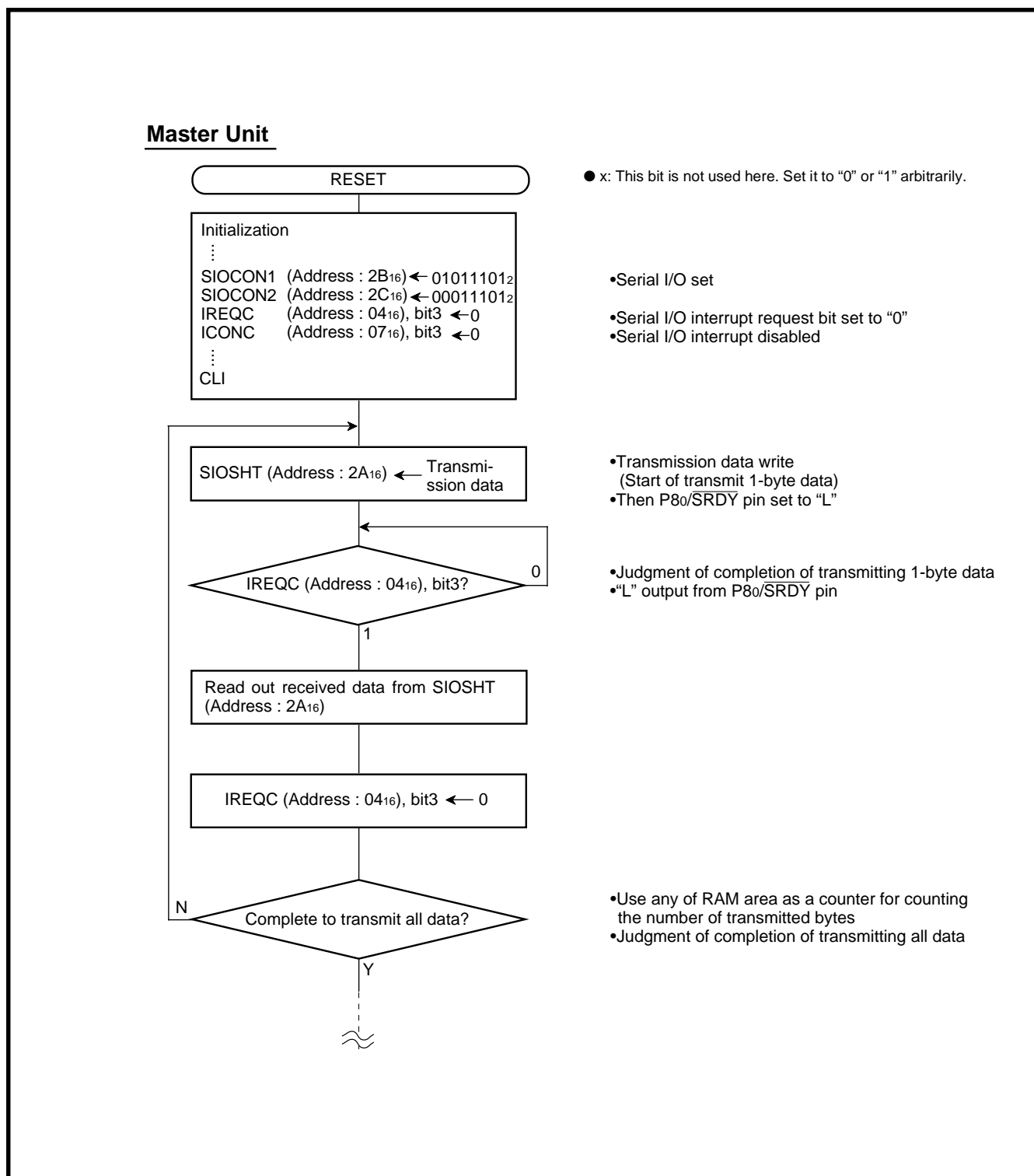


Fig. 2.3.17 Control procedure of SPI compatible mode in master

### 2.3.5 Notes on serial I/O

#### (1) Clock

When the external clock (SCLK pin input) is selected as the transfer clock, its transfer clock needs to be controlled by the external source because the serial I/O shift register will keep being shifted while transfer clock is input even after transfer completion.

#### (2) Reception

When the external clock (SCLK pin input) is selected as the transfer clock for reception, the receiving operation will start owing to the shift clock input even if write operation to the serial I/O shift register (SIOSHT) is not performed. The serial I/O interrupt request also occurs at completion of receiving. However, we recommend to write dummy data in the serial I/O shift register. Because this will cause followings and improve transfer reliability.

- Write to SIOSHT puts the SRDY pin to "L". This enables shift clock output of an external device.
- Write to SIOSHT clears the internal serial I/O counter.

**Note:** Do not read the serial I/O shift register which is shifting. Because this will cause incorrect-data read.

#### (3) STXD output

- When the internal synchronous clock is selected as the transfer clock, the STXD pin goes a high-impedance state after transfer completion.
- When the external clock (SCLK pin input) is selected as the transfer clock, the STXD pin does not go a high-impedance state after transfer completion.

#### (4) SPI compatible mode

- When using the SPI compatible mode, set the  $\overline{\text{SRDY}}$  select bit to "1" ( $\overline{\text{SRDY}}$  signal output).
- When the external clock is selected in SPI compatible mode, the SRXD pin functions as a data output pin and the STXD pin functions as a data input pin.
- Do not write to the serial I/O shift register (SIOSHT) during a transfer as slave when in SPI compatible mode.
- Master operation of SPI compatible mode requires the timings:
  - From write operation to the SIOSHT to  $\overline{\text{SRDY}}$  pin put to "L"  
Requires 2 cycles of internal clock  $\phi$  + 2 cycles of serial I/O synchronous clock + 35 ns
  - From  $\overline{\text{SRDY}}$  pin put to "L" to SCLK switch  
Requires 35 ns
  - From the last pulse of SCLK to  $\overline{\text{SRDY}}$  pin put to "H"  
Requires 35 ns.

## 2.4 UART

This paragraph explains the registers setting method and the notes related to the UART.

### 2.4.1 Memory map

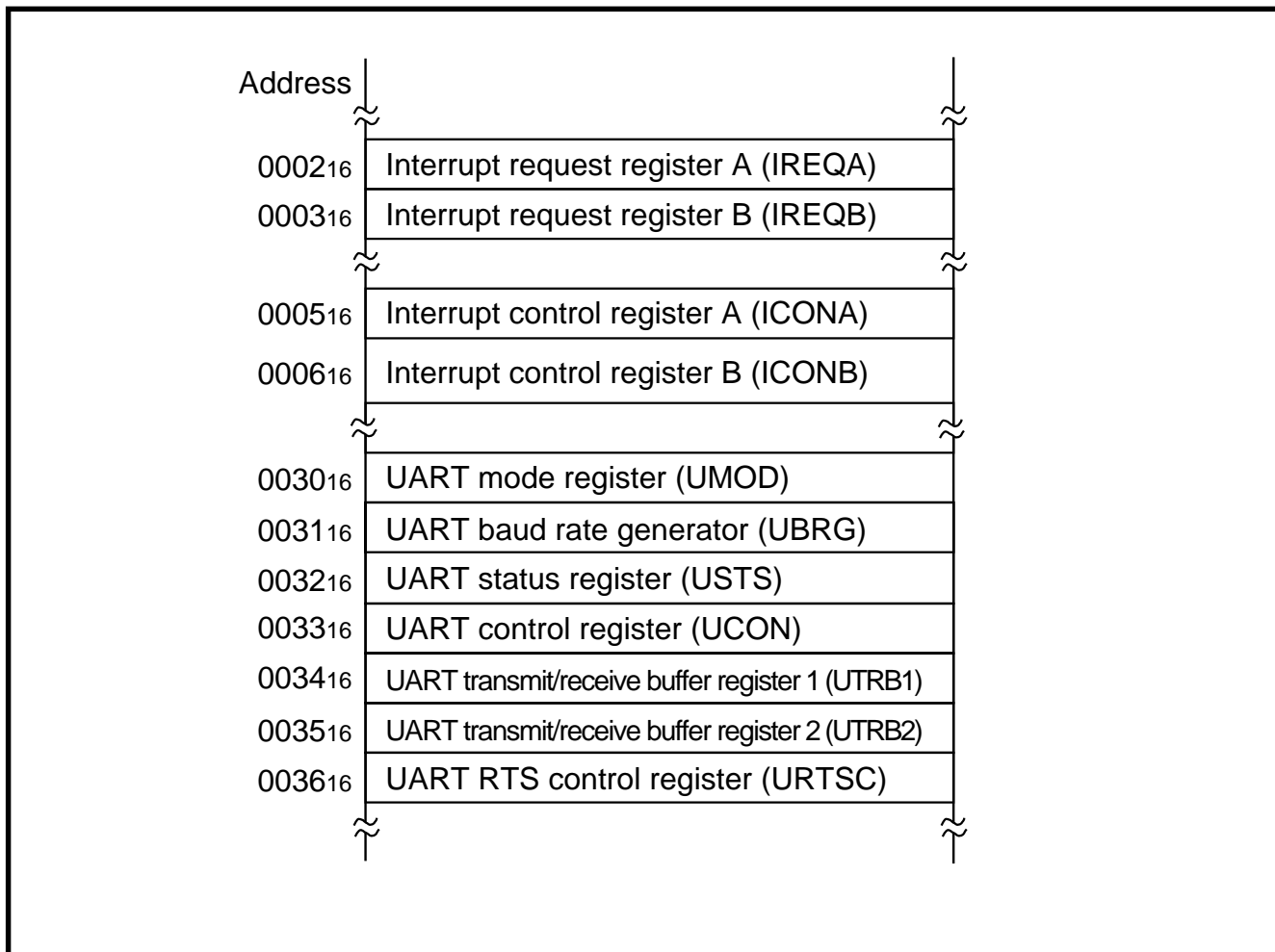


Fig. 2.4.1 Memory map of registers related to UART

## 2.4.2 Related registers

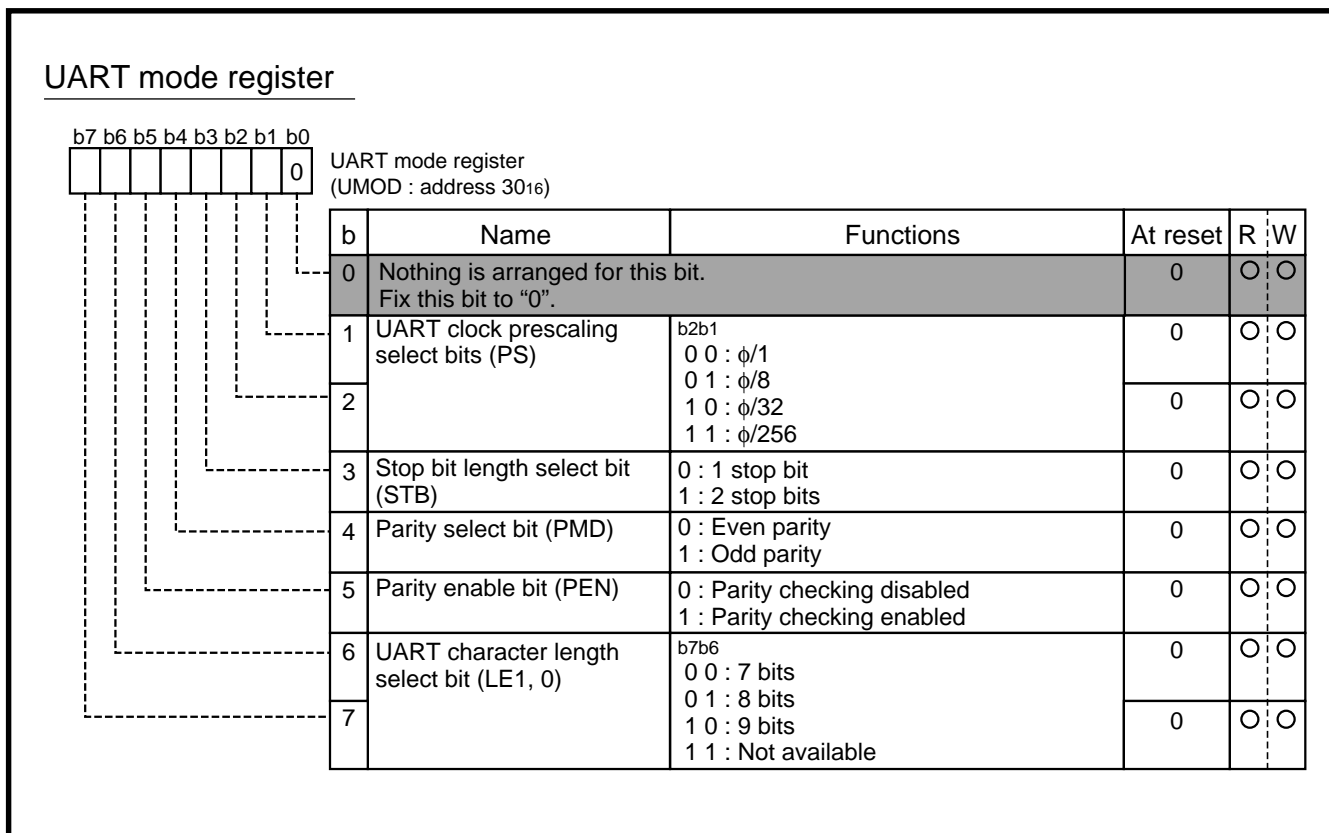


Fig. 2.4.2 Structure of UART mode register

## UART control register

UART control register  
(UCON : address 3316)

b	Name	Functions	At reset	R	W
0	Transmit enable bit (TEN)	0 : Transmit disabled 1 : Transmit enabled	0	○	○
1	Receive enable bit (REN)	0 : Receive disabled 1 : Receive enabled	0	○	○
2	Transmit initialization bit (TIN)	0 : No action 1 : Initializing ( <b>Note 1</b> )	0	○	○
3	Receive initialization bit (RIN)	0 : No action 1 : Initializing ( <b>Note 2</b> )	0	○	○
4	Transmit interrupt source select bit (TIS)	0 : Interrupt when transmit buffer has emptied 1 : Interrupt when transmit shift operation is completed	0	○	○
5	CTS function enable bit (CTS_SEL)	0 : $\overline{\text{CTS}}$ function disabled ( <b>Note 3</b> ) 1 : CTS function enabled	0	○	○
6	RTS function enable bit (RTS_SEL)	0 : $\overline{\text{RTS}}$ function disabled ( <b>Note 4</b> ) 1 : RTS function enabled	0	○	○
7	UART address mode enable bit (AME)	0 : Address mode disabled 1 : Address mode enabled	0	○	○

- Notes 1:** When setting the TIN bit to "1", the TEN bit is set to "0" and the UART status register will be set to "0316" after the data has been transmitted. To retransmit, set the TEN bit to "1" and set a data to the transmit buffer register again. The TIN bit will be cleared to "0" one cycle later after the TIN bit has been set to "1".
- 2:** Setting the RIN bit to "1" suspends the receiving operation and will set all of the REN, RBF and the receive error flags (PER, FER, OER, SER) to "0". The RIN bit will be cleared to "0" one cycle later after the RIN bit has been set to "1".
- 3:** When  $\overline{\text{CTS}}$  function is disabled (CTS\_SEL = "0"), P86 pin can be used as ordinary I/O ports.
- 4:** When  $\overline{\text{RTS}}$  function is disabled (RTS\_SEL = "0"), P83 pin can be used as ordinary I/O ports.

Fig. 2.4.3 Structure of UART control register



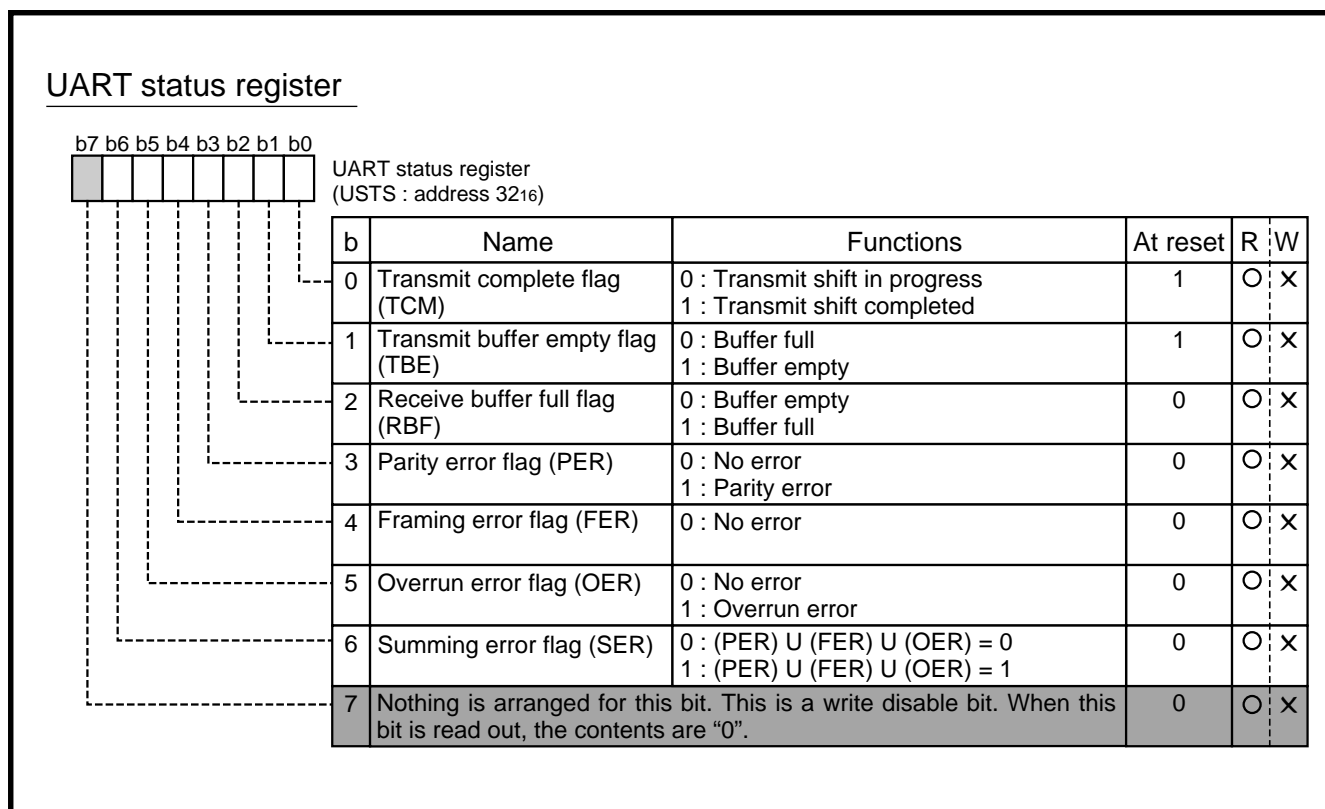


Fig. 2.4.4 Structure of UART status register

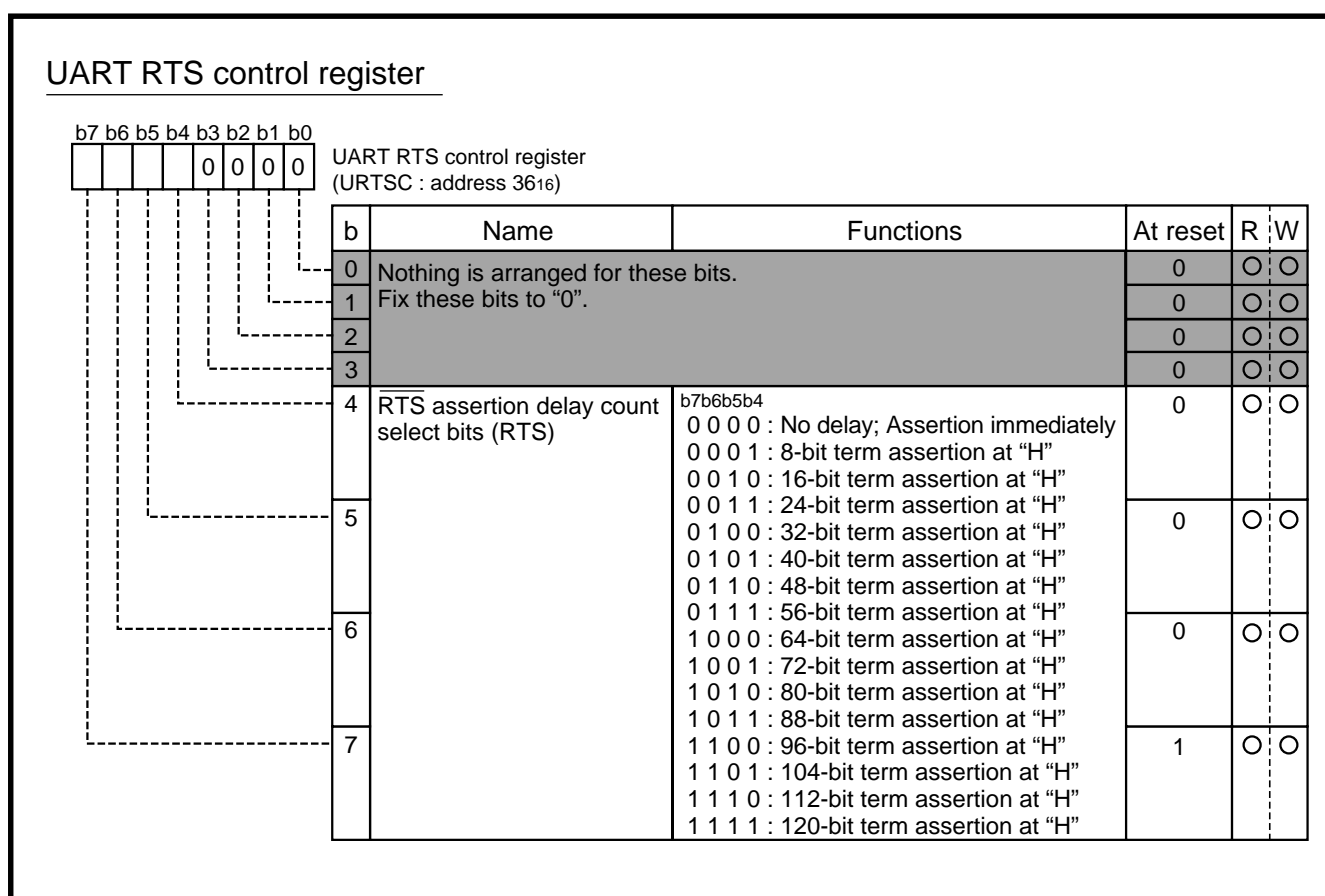


Fig. 2.4.5 Structure of UART RTS control register

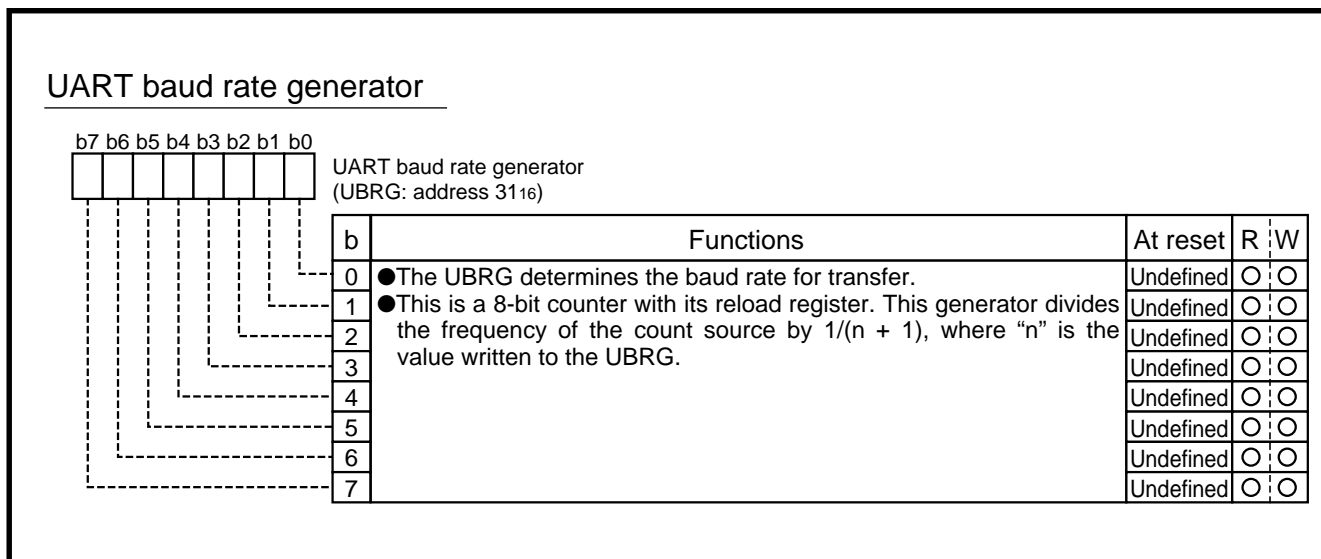


Fig. 2.4.6 Structure of UART baud rate generator

### UART transmit/receive buffer registers 1, 2

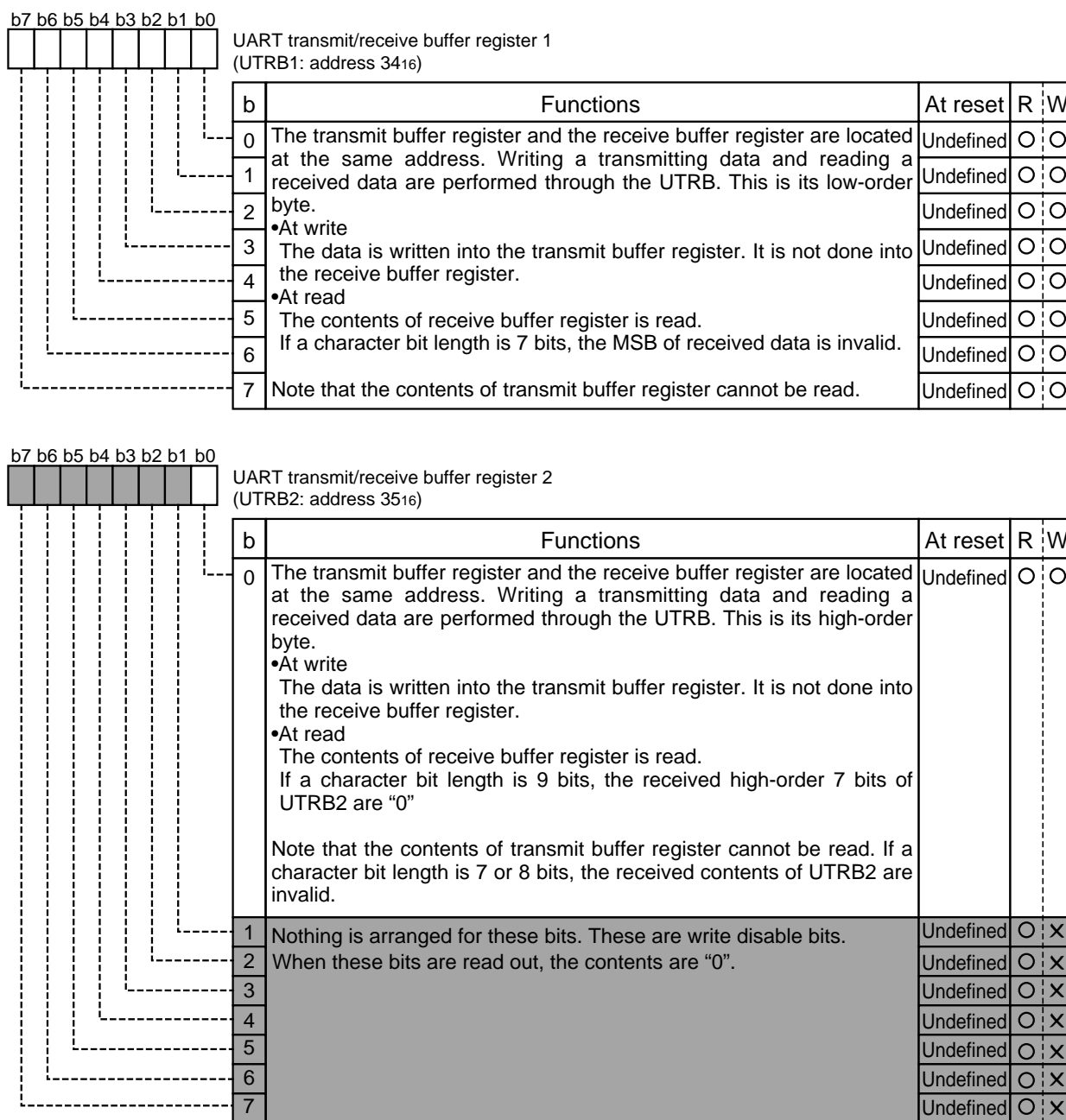


Fig. 2.4.7 Structure of UART transmit/receive buffer registers 1, 2

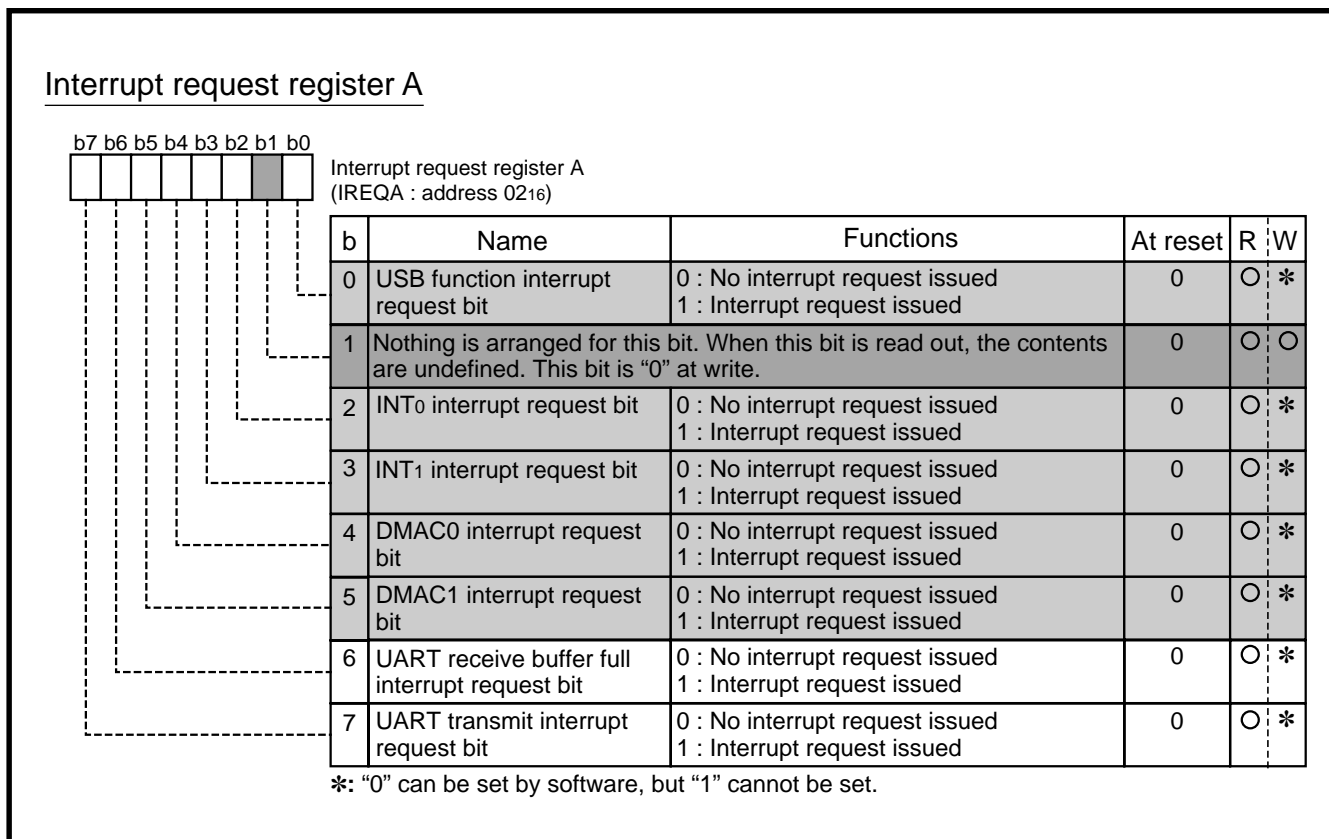


Fig. 2.4.8 Structure of Interrupt request register A

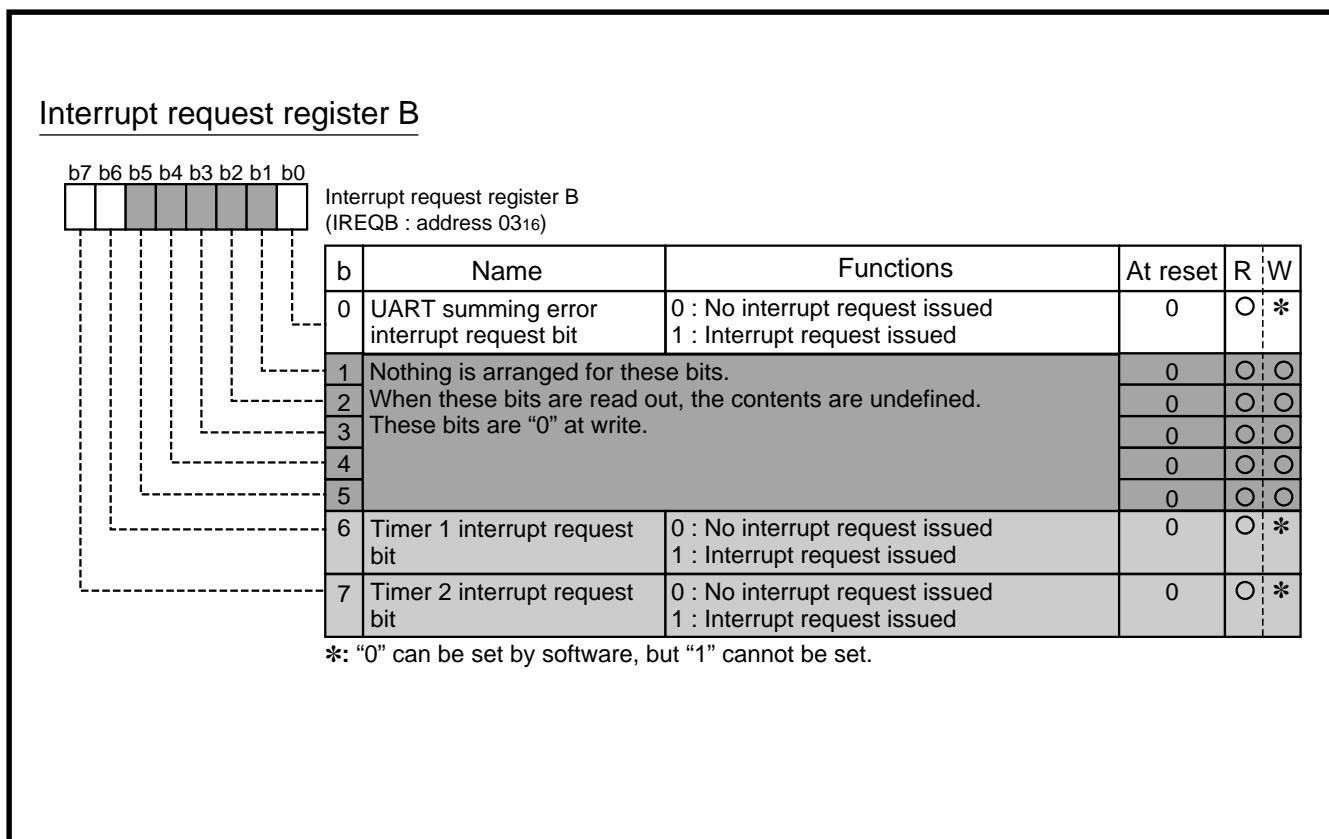


Fig. 2.4.9 Structure of Interrupt request register B

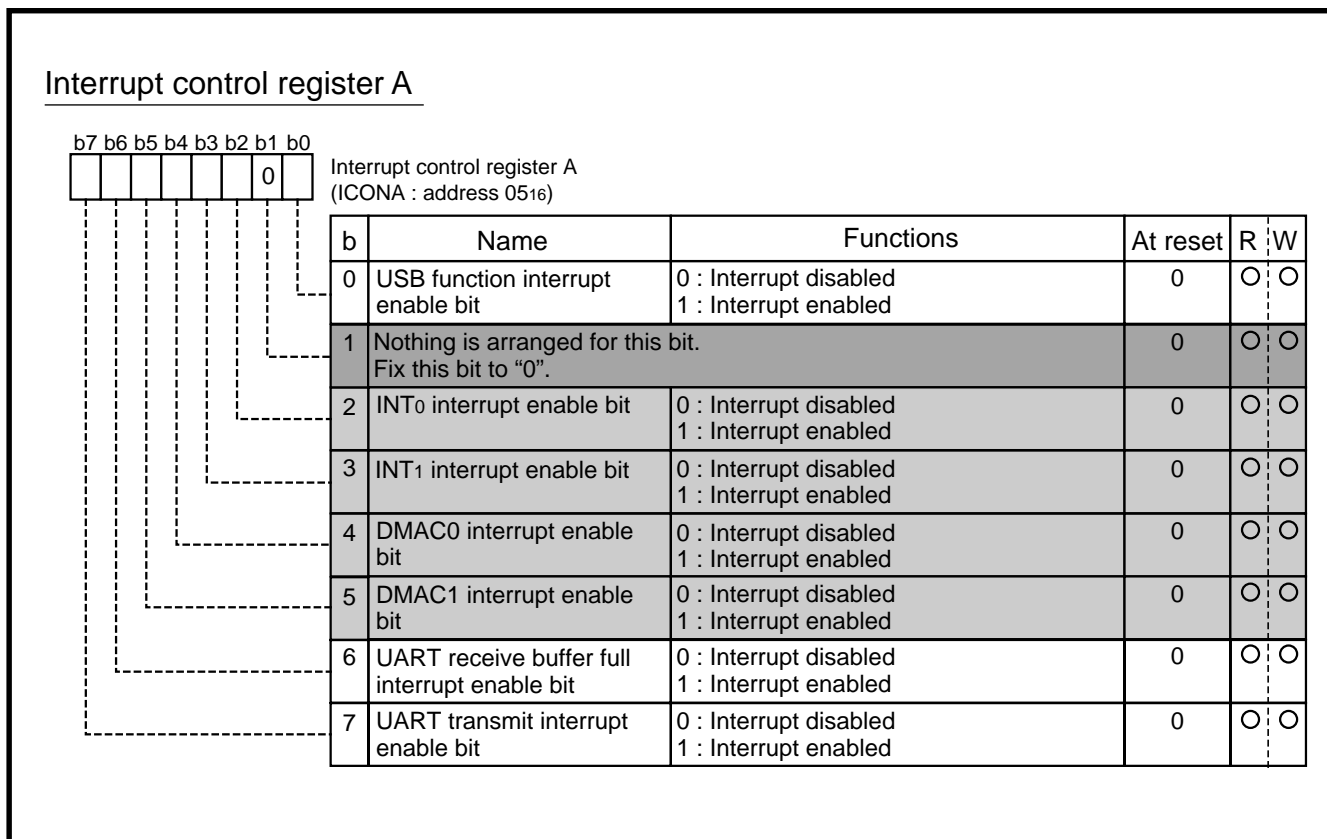


Fig. 2.4.10 Structure of Interrupt control register A

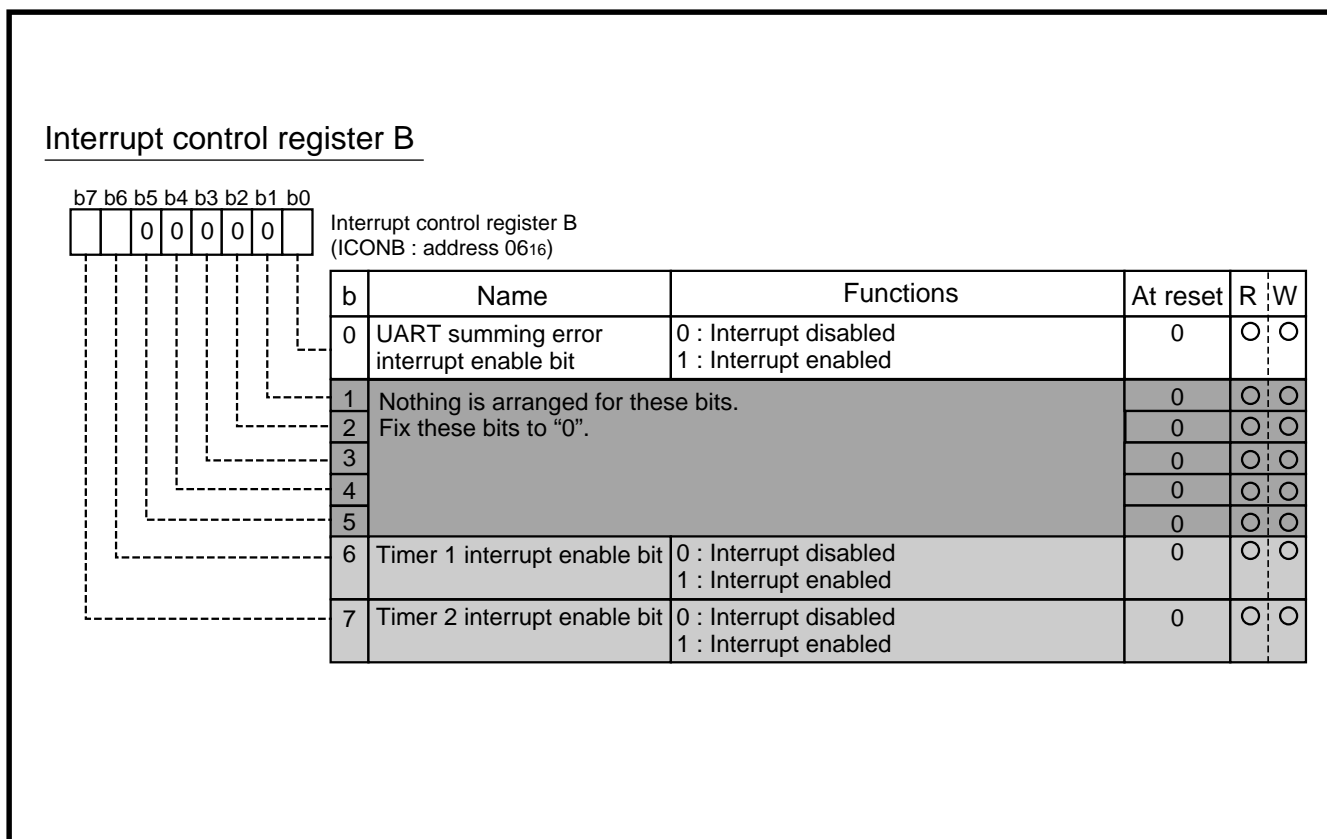


Fig. 2.4.11 Structure of Interrupt control register B

### 2.4.3 UART transfer data format

Figure 2.4.12 shows the UART transfer data format.

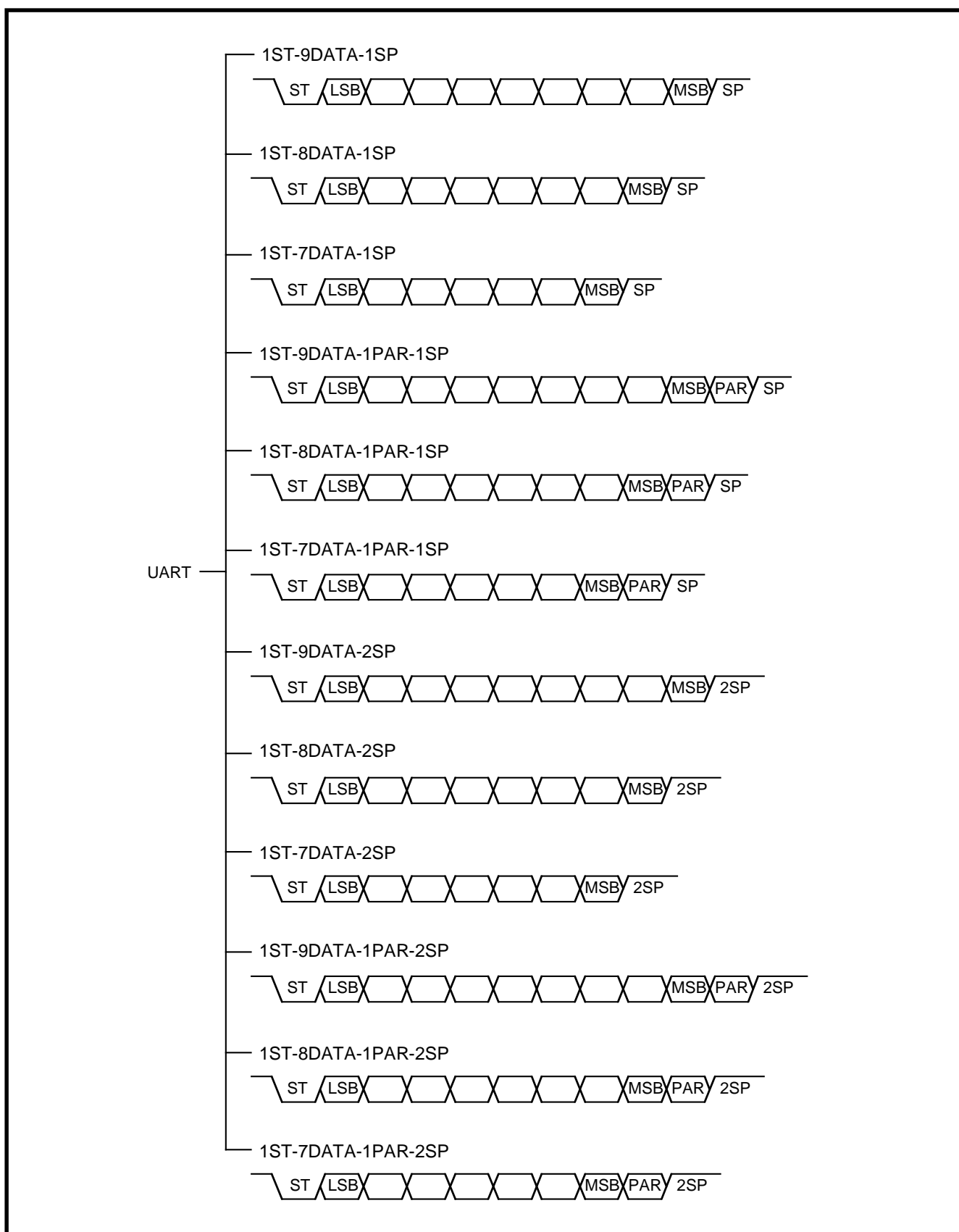


Fig. 2.4.12 UART transfer data format

### 2.4.4 Transfer bit rate

Table 2.4.1 shows setting examples of the baud rate generator (BRG) values and transfer bit rate values. The internal clock  $\phi$  is selected for the UART clock.

**Table 2.4.1 Setting examples of baud rate generator values and transfer bit rate values ( $\phi = 12$  MHz)**

$\phi/1$ (Note 1)		$\phi/8$ (Note 1)		$\phi/32$ (Note 1)		$\phi/256$ (Note 1)	
BRG setting value	Transfer bit rate (bps)	BRG setting value	Transfer bit rate (bps)	BRG setting value	Transfer bit rate (bps)	BRG setting value	Transfer bit rate (bps)
00 (00 <sub>16</sub> )	750,000.0						
01 (01 <sub>16</sub> )	375,000.0						
02 (02 <sub>16</sub> )	250,000.0						
03 (03 <sub>16</sub> )	187,500.0						
04 (04 <sub>16</sub> )	150,000.0						
05 (05 <sub>16</sub> )	125,000.0						
06 (06 <sub>16</sub> )	107,142.9						
07 (07 <sub>16</sub> )	93,750.0	00 (00 <sub>16</sub> )	93,750.0				
08 (08 <sub>16</sub> )	83,333.3	01 (01 <sub>16</sub> )	46,875.0				
09 (09 <sub>16</sub> )	75,000.0	02 (02 <sub>16</sub> )	31,250.0				
10 (0A <sub>16</sub> )	68,181.8	03 (03 <sub>16</sub> )	23,437.5	00 (00 <sub>16</sub> )	23,437.5		
11 (0B <sub>16</sub> )	65,250.0	04 (04 <sub>16</sub> )	18,750.0	01 (01 <sub>16</sub> )	11,718.7		
12 (0C <sub>16</sub> )	57,692.3	05 (05 <sub>16</sub> )	15,625.0	02 (02 <sub>16</sub> )	7,812.5		
13 (0D <sub>16</sub> )	53,571.4	06 (06 <sub>16</sub> )	13,392.8	03 (03 <sub>16</sub> )	5,859.4		
14 (0E <sub>16</sub> )	50,000.0	07 (07 <sub>16</sub> )	11,718.7	04 (04 <sub>16</sub> )	4,687.5		
15 (0F <sub>16</sub> )	46,875.0	08 (08 <sub>16</sub> )	10,416.6	05 (05 <sub>16</sub> )	3,906.3		
•	•	09 (09 <sub>16</sub> )	9,375.0	06 (06 <sub>16</sub> )	3,348.2		
•	•	10 (0A <sub>16</sub> )	8,522.7	07 (07 <sub>16</sub> )	2,929.7	00 (00 <sub>16</sub> )	2,929.7
•	•	11 (0B <sub>16</sub> )	7,812.5	08 (08 <sub>16</sub> )	2,604.2	01 (01 <sub>16</sub> )	1,464.8
•	•	12 (0C <sub>16</sub> )	7,211.5	09 (09 <sub>16</sub> )	2,343.7	02 (02 <sub>16</sub> )	976.6
•	•	13 (0D <sub>16</sub> )	6,696.4	10 (0A <sub>16</sub> )	2,130.6	03 (03 <sub>16</sub> )	732.4
•	•	14 (0E <sub>16</sub> )	6,250.0	11 (0B <sub>16</sub> )	1,953.1	04 (04 <sub>16</sub> )	585.9
•	•	15 (0F <sub>16</sub> )	5,859.3	12 (0C <sub>16</sub> )	1,802.8	05 (05 <sub>16</sub> )	488.3
•	•	•	•	13 (0D <sub>16</sub> )	1,674.1	06 (06 <sub>16</sub> )	418.5
•	•	•	•	14 (0E <sub>16</sub> )	1,562.5	07 (07 <sub>16</sub> )	366.2
•	•	•	•	15 (0F <sub>16</sub> )	1,464.8	08 (08 <sub>16</sub> )	325.5
•	•	•	•	•	•	09 (09 <sub>16</sub> )	292.9
•	•	•	•	•	•	10 (0A <sub>16</sub> )	266.3
•	•	•	•	•	•	11 (0B <sub>16</sub> )	244.1
•	•	•	•	•	•	12 (0C <sub>16</sub> )	225.3
•	•	•	•	•	•	13 (0D <sub>16</sub> )	209.2
•	•	•	•	•	•	14 (0E <sub>16</sub> )	195.3
•	•	•	•	•	•	15 (0F <sub>16</sub> )	183.1
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
253 (FD <sub>16</sub> )	2,952.7	253 (FD <sub>16</sub> )	369.0	253 (FD <sub>16</sub> )	92.2	253 (FD <sub>16</sub> )	11.5
254 (FE <sub>16</sub> )	2,941.1	254 (FE <sub>16</sub> )	367.6	254 (FE <sub>16</sub> )	91.9	254 (FE <sub>16</sub> )	11.4
255 (FF <sub>16</sub> )	2,929.7	255 (FF <sub>16</sub> )	366.2	255 (FF <sub>16</sub> )	91.6	255 (FF <sub>16</sub> )	11.4

**Notes 1:** Select the UART clock prescaling with bits 1 and 2 of UART mode register.

**2:** Equation of transfer bit rate:

$$\text{Transfer bit rate (bps)} = \frac{f_i^*}{(\text{BRG setting value} + 1) \times 16}$$

\* :  $f_i$  is selectable among  $\phi/1$ ,  $\phi/8$ ,  $\phi/32$ , and  $\phi/256$  with bits 1 and 2 of UART mode register.

## 2.4.5 Operation of transmitting and receiving

### (1) Transmit operation

- The transmit buffer empty flag (TBE) is set to “0” when the low-order byte of transmitted data is written into the UART transmit buffer register 1 in the condition of transmission enabled. When using 9-bit character length, set the data into the UART transmit buffer register 2 (high-order byte) first before the UART transmit buffer register 1 (low-order byte).
- If the transmit shift register is empty in the condition of  $\overline{\text{CTS}}$  function disabled, the transmitted data which is written into the UART transmit buffer register 1 will be transferred to the transmit shift register at the same time. When the TBE flag becomes “1”, the following data can be set to the UART transmit buffer. At this point, the UART transmit interrupt request occurs when the transmit interrupt source select bit (TIS) is “0”.
- When the  $\overline{\text{CTS}}$  function is enabled, the transmitted data is not transferred to the transmit shift register until “L” is input to the  $\overline{\text{CTS}}$  pin (P8<sub>6</sub>/ $\overline{\text{CTS}}$ ).
- The data is transmitted with the LSB first format. Once the transmission starts, it continues until the last bit has been transmitted even though clearing the transmit enable bit (TEN) to “0” (disabled) or inputting “H” to the  $\overline{\text{CTS}}$  pin.
- After completion of the last bit transmitting, if the TBE flag is “1”, or the TEN bit is “0” (disabled) or “H” is input to the  $\overline{\text{CTS}}$  pin, the transmit complete flag (TCM) goes to “1”. At this point, the UART transmit interrupt request occurs when the TIS bit is “1”.

### (2) Receive operation

- The data is received with the LSB first format in the condition of reception enabled.
- When the stop bit is detected, the received data is transferred from the receive shift register to the UART receive buffer register. At the same time, if there is no error, the receive buffer full flag (RBF) is set to “1” and the UART receive buffer full interrupt request occurs.
- If receive errors occur, the corresponding error flags of UART status register are set to “1” and the UART summing error interrupt request occurs.
- The receive buffer full flag (RBF) is set to “0” when the contents of UART receive buffer register 1 is read out. Then when the  $\overline{\text{RTS}}$  function is disabled, the following data can be received. When using 9-bit character length, read the data from the UART receive buffer register 2 (high-order byte) first before the UART receive buffer register 1 (low-order byte).
- When the  $\overline{\text{RTS}}$  function is enabled, the  $\overline{\text{RTS}}$  assertion delay count is specified by the UART  $\overline{\text{RTS}}$  control register. The delay time from the reception of the last stop bit to the start bit is selectable. The  $\overline{\text{RTS}}$  pin (P8<sub>7</sub>/ $\overline{\text{RTS}}$ ) outputs “H” during the delayed time. After that, the  $\overline{\text{RTS}}$  pin outputs “L” and a reception is enabled.
- If the start bit is detected in the term of “H” assertion of  $\overline{\text{RTS}}$ , its assertion count is suspended and the  $\overline{\text{RTS}}$  pin remains “H” output. After receiving the last stop bit, the count is resumed.



**(3) Countermeasure for errors**

Three errors can be detected at reception. Each error is detected simultaneously when the data is transferred from the receive shift register to the receive buffer register. If receive errors occur, the corresponding error flags of UART status register are set to "1". When any one of errors occurs, the summing error flag is set to "1" and the UART summing error interrupt request bit is also set to "1". If a receive error occurs, the reception does not set the UART receive buffer full interrupt request bit to "1".

If receive errors occur, initialize the error flags and the UART receive buffer register and then retransmit the data.

Table 2.4.3 shows the error flags set condition and how to clear error flags.

**Table 2.4.3 Error flags set condition and how to clear error flags**

<b>Error flag</b>	<b>Error flag set condition</b>	<b>How to clear error flag</b>
Overrun flag (OER)	<ul style="list-style-type: none"> <li>•If the previous data in the receive buffer register is not read before the current receive operation is completed.</li> <li>•If any one of error flags is "1" for the previous data and the current receive operation is completed.</li> </ul>	<ul style="list-style-type: none"> <li>•Reading UART status register</li> <li>•Hardware reset</li> <li>•Setting the receive initialization bit (RIN) to "1"</li> </ul>
Framing error flag (FER)	<ul style="list-style-type: none"> <li>•When the number of stop bit of the received data does not correspond with the selection with the stop bit length select bit (STB).</li> </ul>	
Parity error flag (PER)	<ul style="list-style-type: none"> <li>•When the sum total of 1s of received data and the parity does not correspond with the selection with the parity select bit (PMD).</li> </ul>	
Summing error flag (SER)	<ul style="list-style-type: none"> <li>•When any one of the PER, FER and OER is set to "1".</li> </ul>	

### 2.4.6 UART application example

#### (1) Data output (control of peripheral IC)

**Outline :** Data is transmitted and received, using the UART.

Figure 2.4.13 shows a connection diagram, and Figure 2.4.14 shows a timing chart.

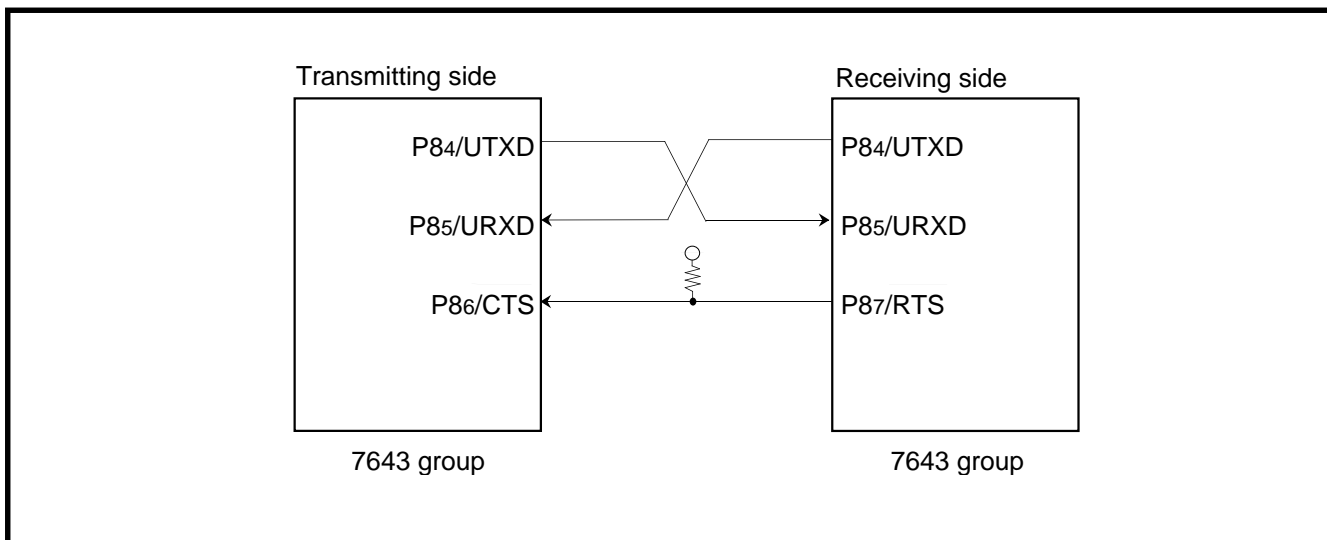


Fig. 2.4.13 Connection diagram

- Specifications :**
- Transmitter: UART is used.
  - Receiver: UART is used.
  - Transfer bit rate : 9600 bps ( $\phi = 12 \text{ MHz}$  divided by 1248)
  - Data format: 1ST-8DATA-2SP
  - Use of  $\overline{\text{CTS}}$  and  $\overline{\text{RTS}}$  functions
  - 2-byte data is transferred from the transmitting side to the receiving side at intervals of 10 ms generated by the timer.

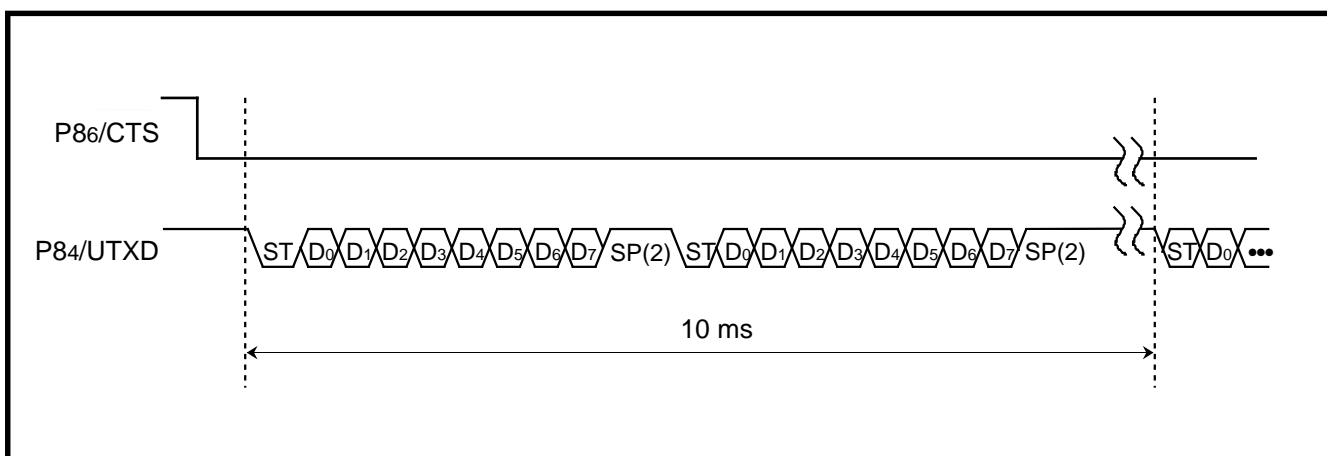


Fig. 2.4.14 Timing chart

Figure 2.4.15 shows the registers setting for the transmitter, and Figures 2.4.16 and 2.4.17 show the registers setting for the receiver.

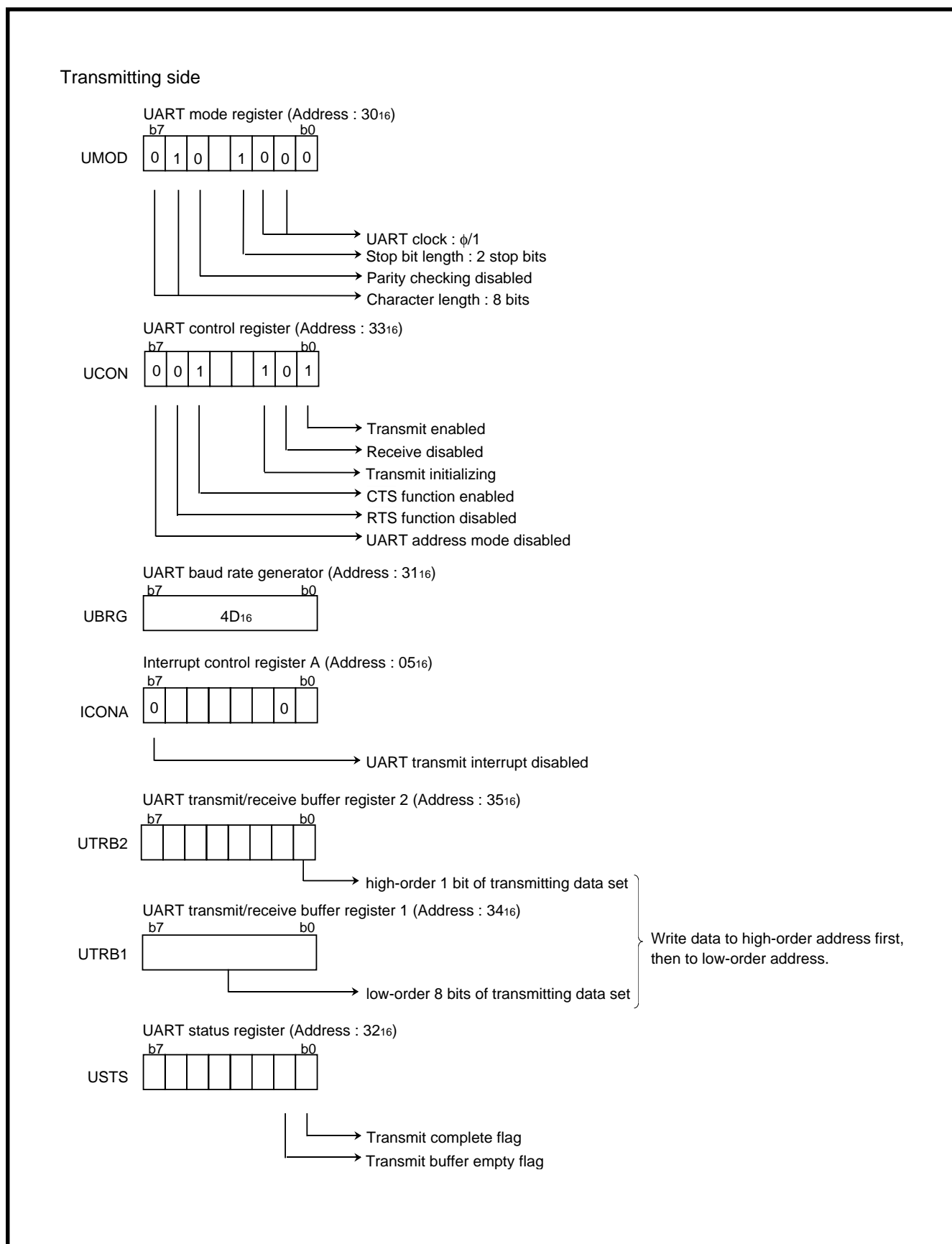


Fig. 2.4.15 Registers setting for transmitter

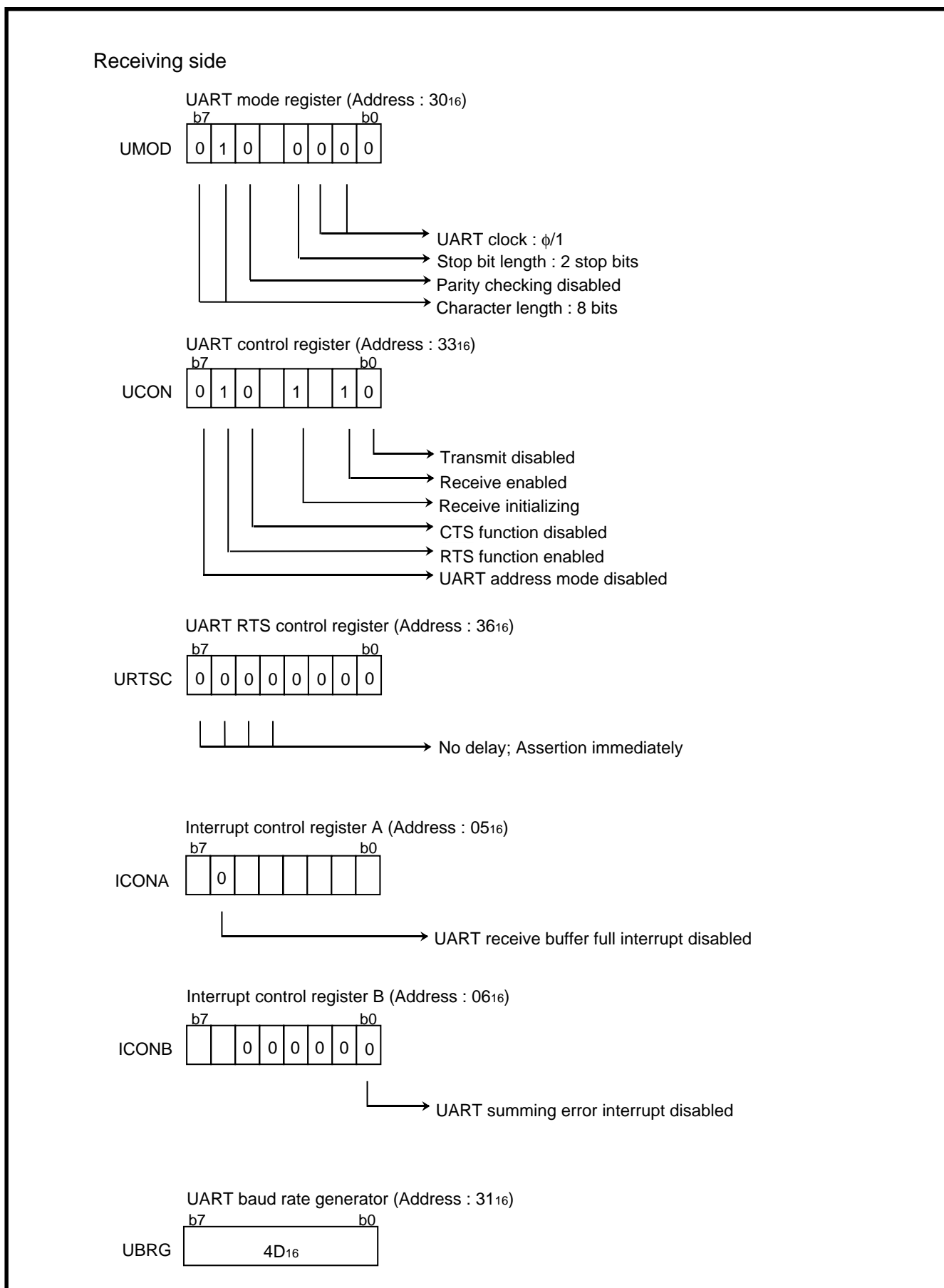


Fig. 2.4.16 Registers setting for receiver (1)

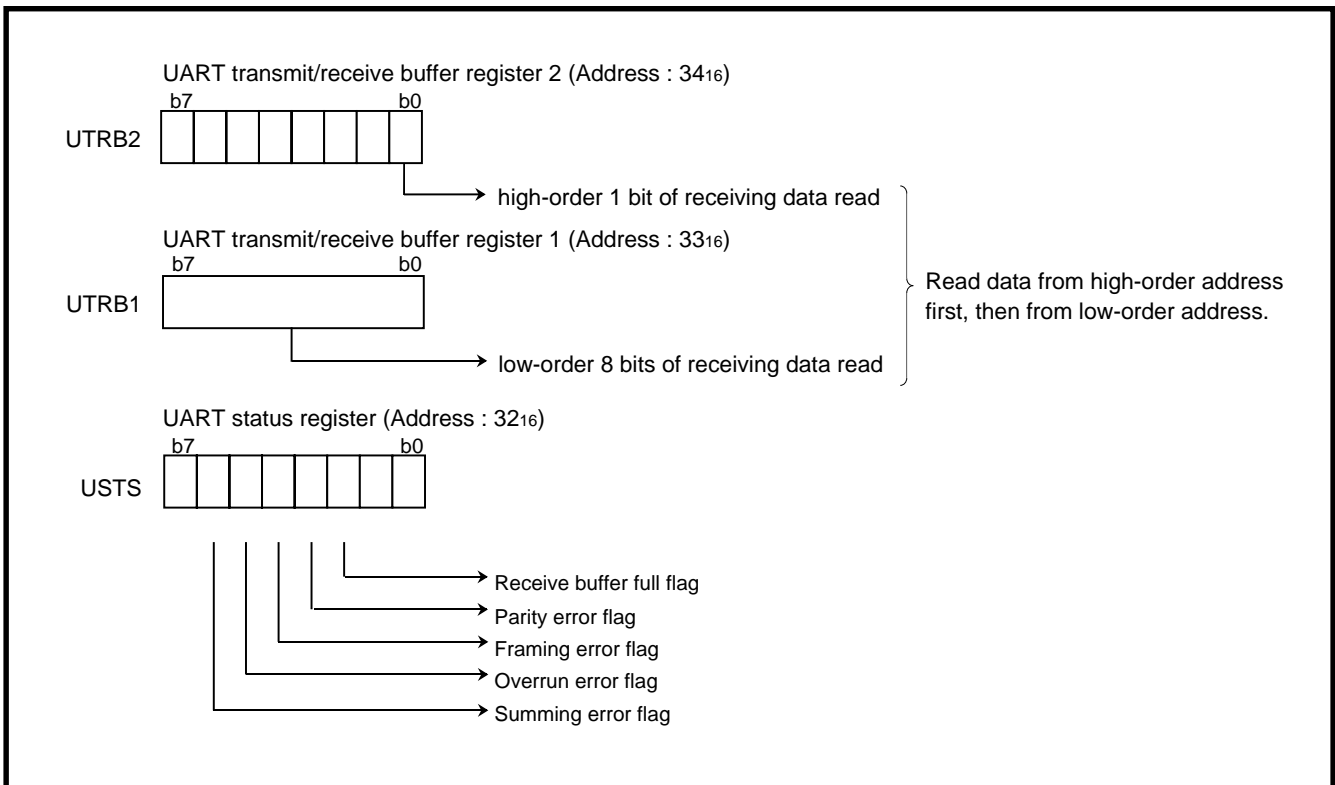


Fig. 2.4.17 Registers setting for receiver (2)

Figure 2.4.18 shows a control procedure of transmitter, and Figure 2.4.19 shows a control procedure of receiver.

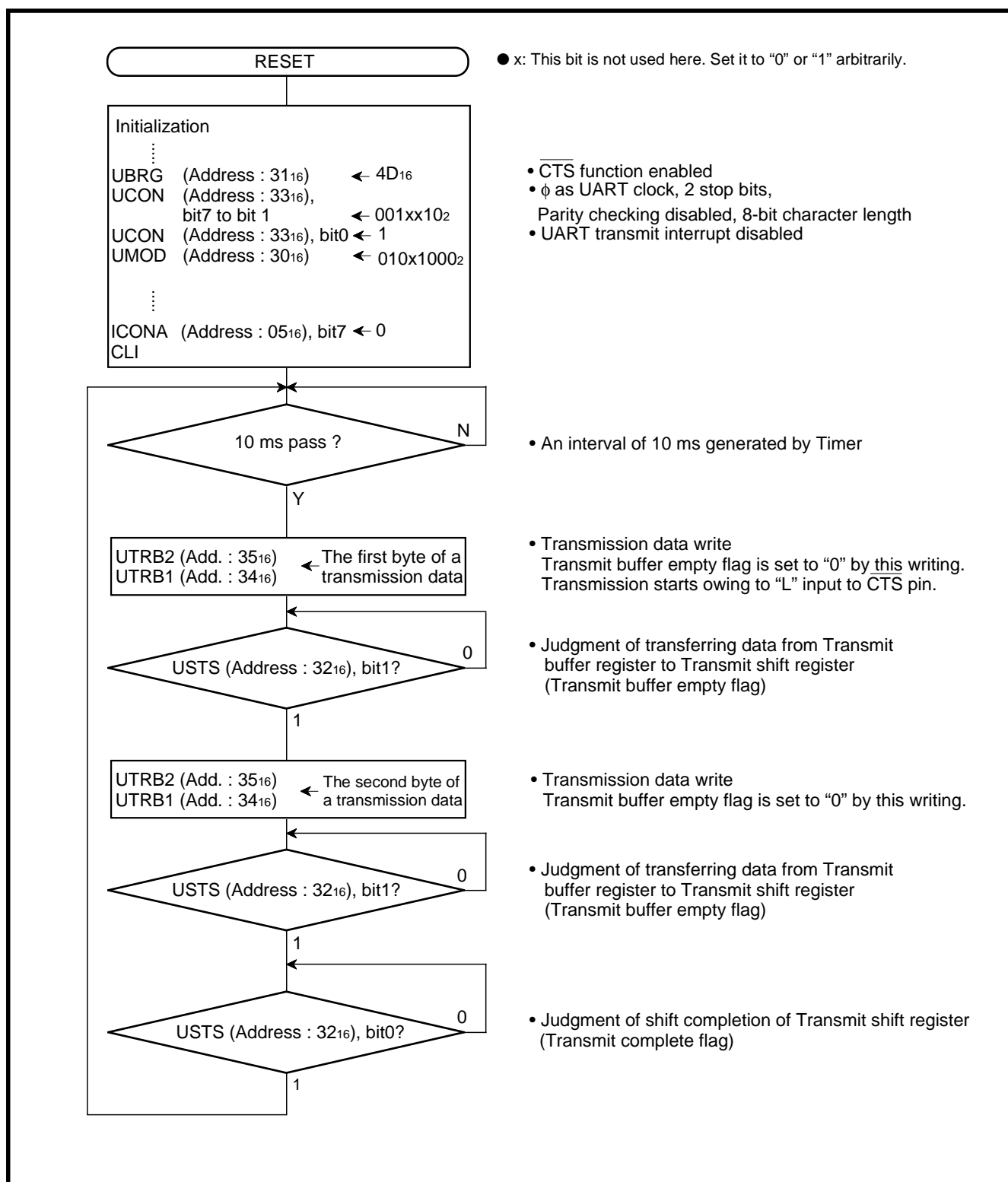


Fig. 2.4.18 Control procedure of transmitter

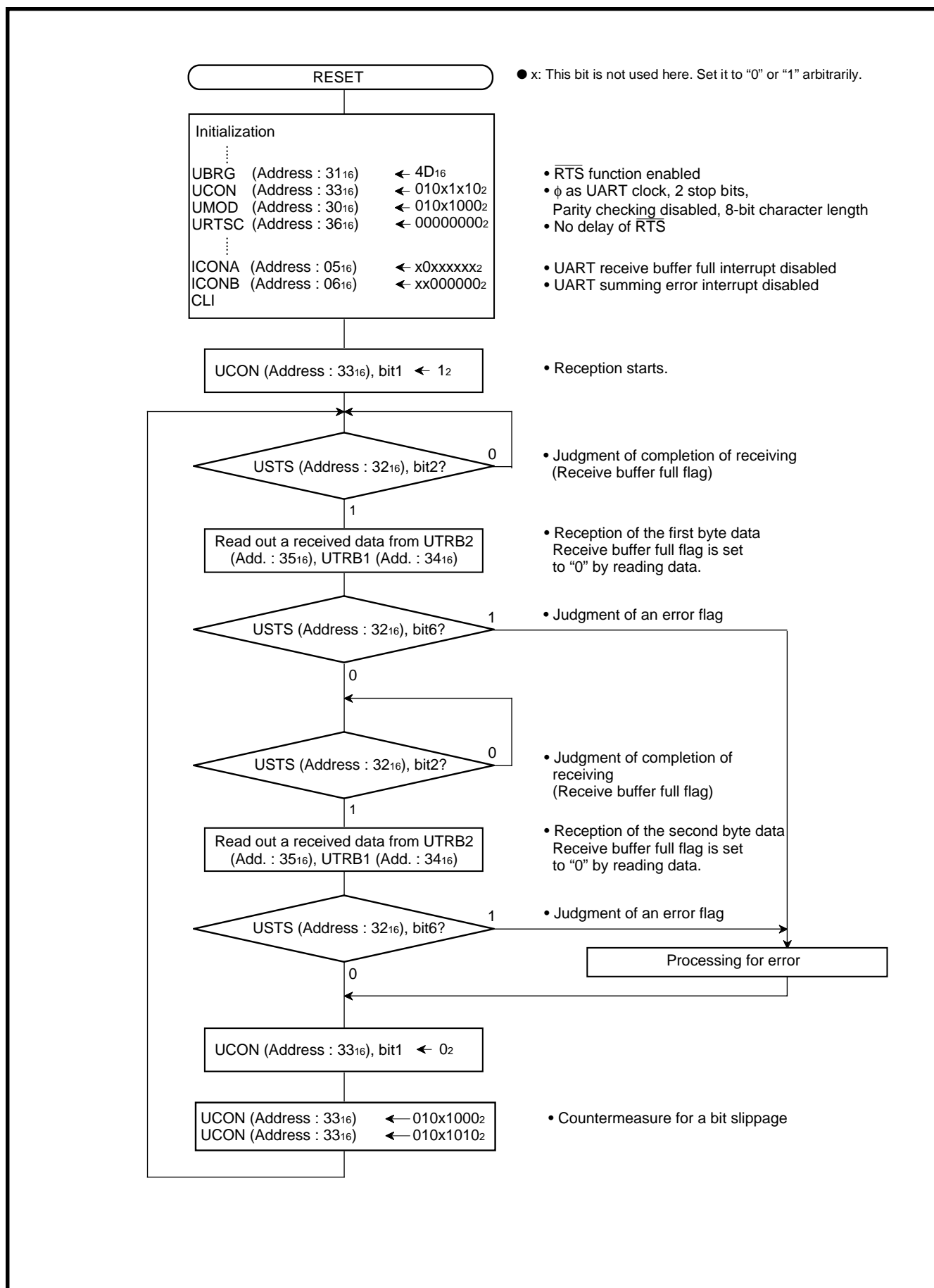


Fig. 2.4.19 Control procedure of receiver

## (2) UART address mode

### ●Operation explanation

The UART address mode is intended for use to communicate between the specified MCUs in a multi-MCU environment. The UART address mode can be used in either an 8-bit or 9-bit character length. An address is identified by the MSB of the incoming data being "1". The bit is "0" for non-address data.

When the MSB of the incoming data is "0" in the UART address mode, the Receive Buffer Full Flag is set to "1", but the Receive Buffer Full Interrupt Request Bit is not set to "1". When the MSB of the incoming data is "1", normal receive operation is performed. In the UART address mode an overrun error is not detected for reception of the 2nd and onward bytes. An occurrence of framing error or parity error sets the Summing Error Interrupt Request Bit to "1" and the data is not received independent of its MSB contents.

Usage of UART address mode is explained as follows:

- (1) Set the UART Address Mode Enable Bit to "1".
- (2) Sends the address data of a slave MCU first from a host MCU to all slave MCUs. The MSB of address data must be "1" and the remaining 7 bits specify the address.
- (3) The all slave MCUs automatically check for the received data whether its stop bit is valid or not, and whether the parity error occurs or not (when the parity enabled). If these errors occur, the Framing Error Flag or Parity Error Flag and the Summing Error Flag are set to "1". Then, the Summing Error Interrupt Request Bit is also set to "1".
- (4) When received data has no error, the all slave MCUs must judge whether the address of the received address data matches with their own addresses by a program. After the MSB being "1" is received, the UART Address Mode Enable Bit is automatically set to "0" (disabled).
- (5) The UART Address Mode Enable Bit of the slave MCUs which have be judged that the address does not match with them must be set to "1" (enabled) again by a program to disable reception of the following data.
- (6) Transmit the data of which MSB is "0" from the host MCU. The slave MCUs disabling the UART address mode receive the data, and their Receive Buffer Full Flags and the Receive Buffer Full Interrupt Request Bits are set to "1". For the other slave MCUs enabling the UART address mode, their Receive Buffer Full Flag are set to "1", but their Receive Buffer Full Interrupt Request Bits are not set to "1".
- (7) An overrun error cannot be detected after the first data has been received in UART Address Mode. Accordingly, even if the slave MCUs does not read the received data and the next data has been received, an overrun error does not occur.

Thus, a communication between a host MCU and the specified MCU can be realized.

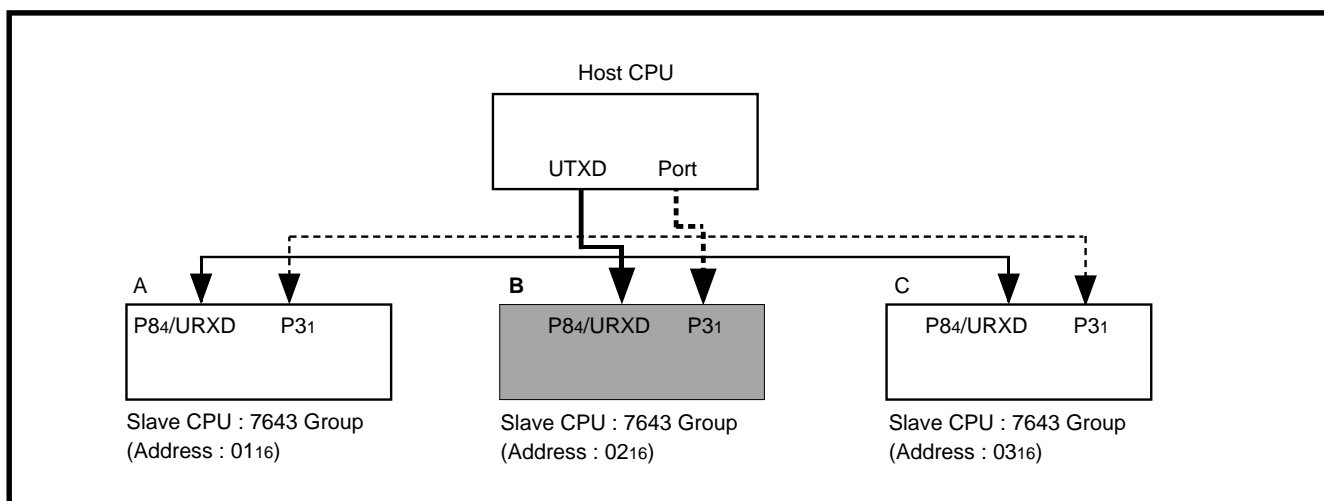


### ●UART address mode application example

**Outline** : The slave CPU (B) receives the data from the host CPU, using the UART address mode.

**Specifications** : •Transfer bit rate : 9600 bps  
 •Data format: 1ST-8DATA-2SP  
 •Use of port P3<sub>1</sub> for communication control

Figure 2.4.20 shows a connection diagram; Figure 2.4.21 shows the registers setting related to UART address mode; Figures 2.4.22 and 2.4.23 show the control procedures.



**Fig. 2.4.20 Connection diagram**

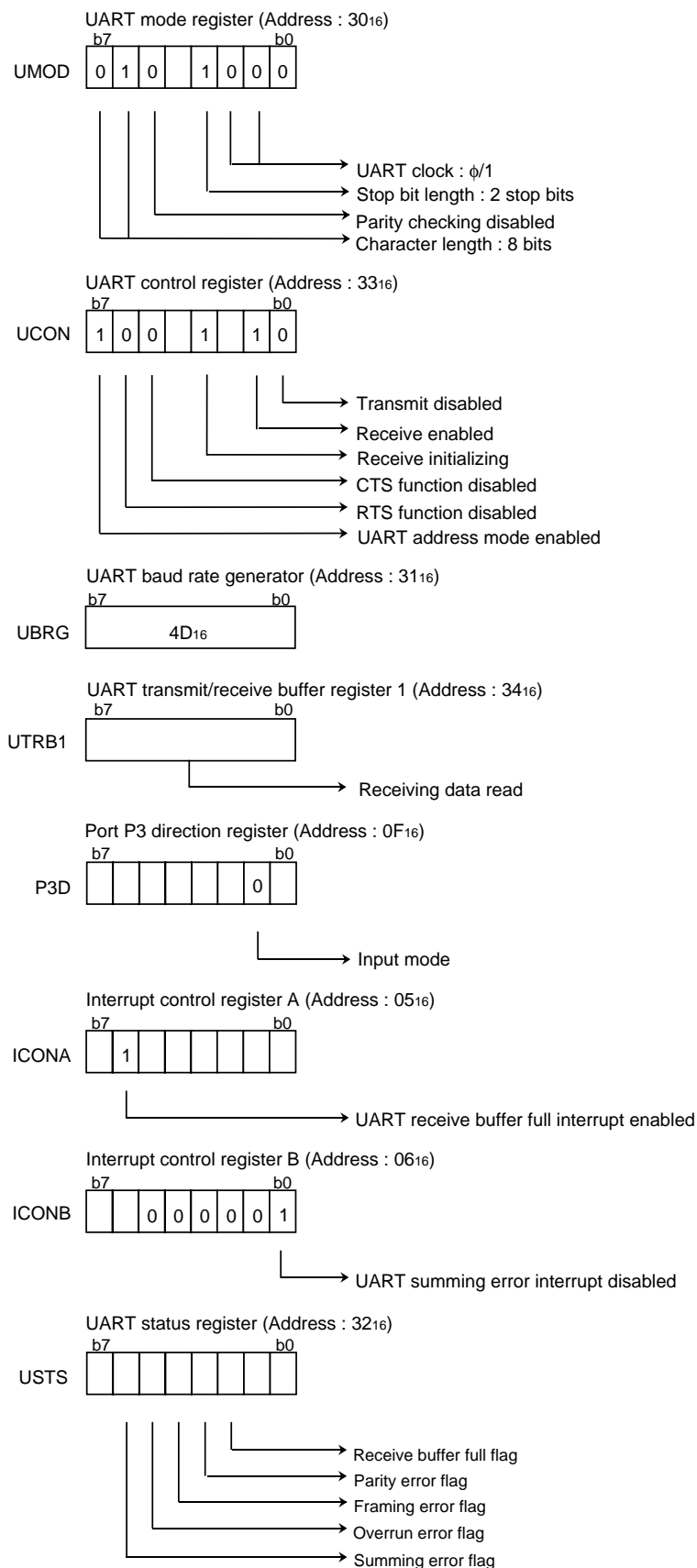


Fig. 2.4.21 Registers setting related to UART address mode

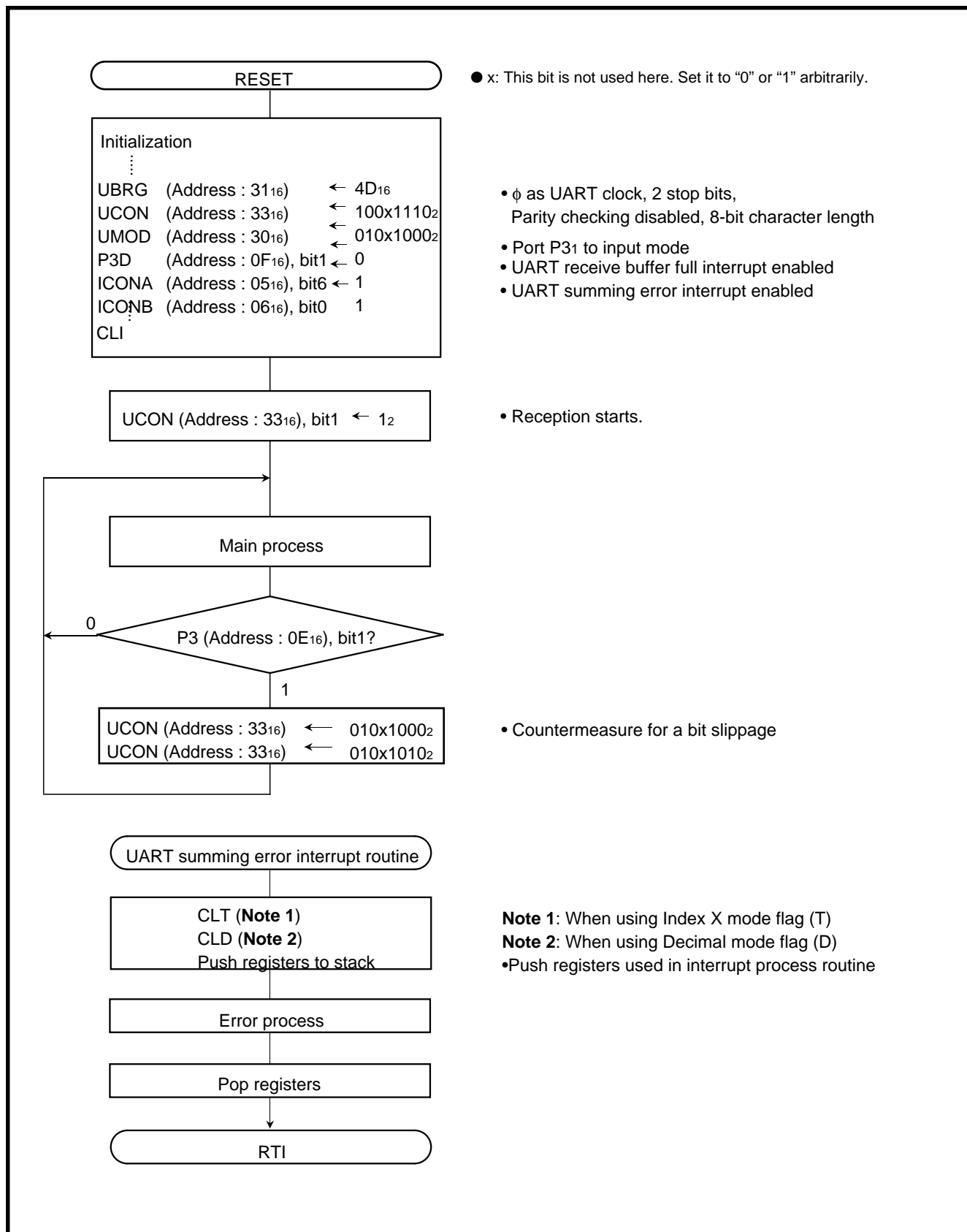


Fig. 2.4.22 Control procedure (1)

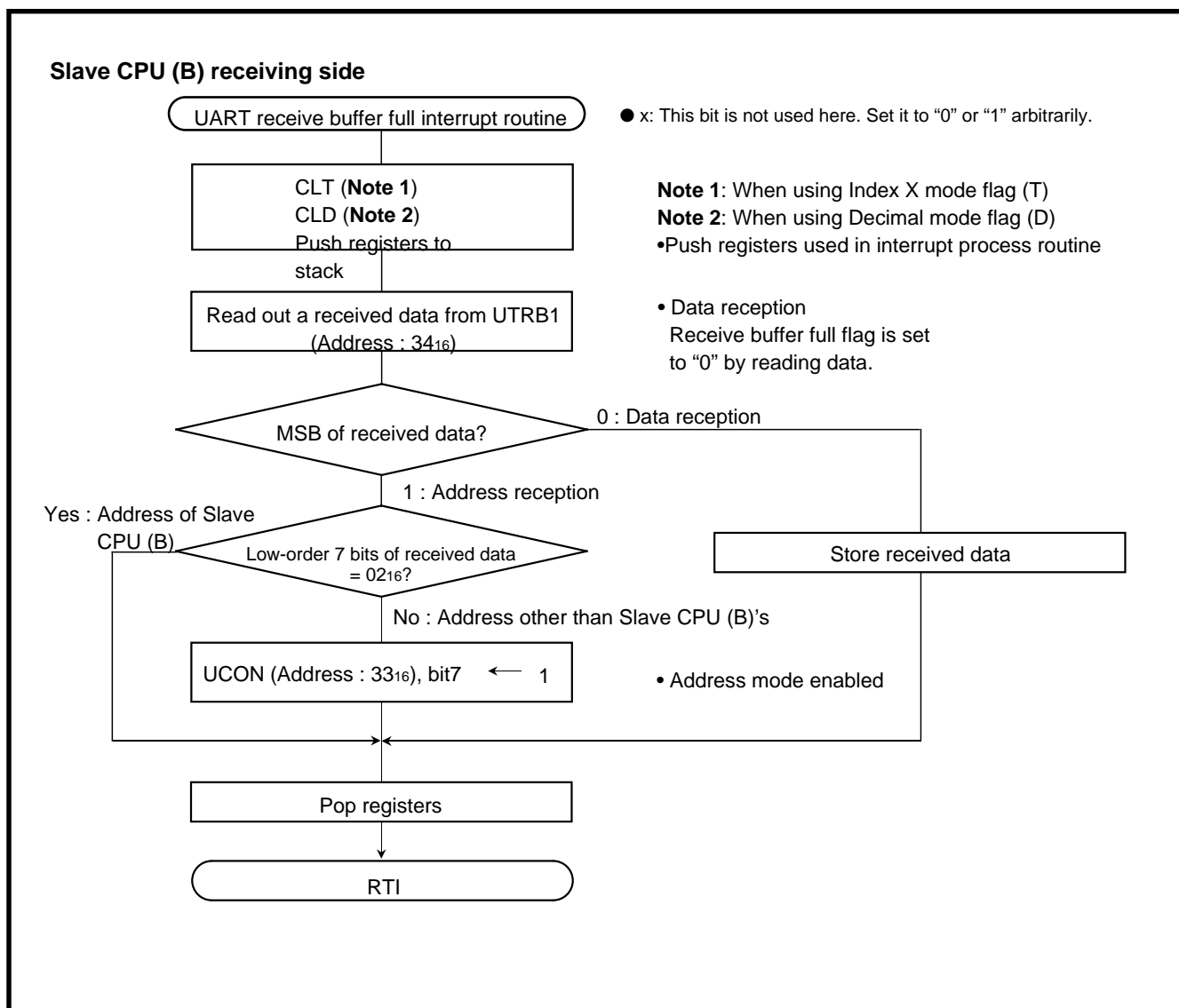


Fig. 2.4.23 Control procedure (2)

**(3) Data packet transfer (with no error processing) from USB FIFO to UART using DMA**

**Outline** : transmit the data in USB FIFO to Host CPU using DMA and UART.

- Specifications** :
- USB Endpoint1: OUT bulk transfer
  - USB Endpoint1: OUT interrupt
  - USB Endpoint1 packet size: 64 Bytes
  - UART transmit is used.
  - DMA cycle steal transfer mode (fixed address -> fixed address transfer)
  - DMA transfer unit : 64 bytes fixed (short packet un-supported)
  - UART transmit interrupt request (when transmitting shift operation is completed) is used for DMA factor
  - Transfer bit rate : 9600 bps ( $\phi = 12 \text{ MHz}$  divided by 1248)  
In this case,  $f(X_{IN})=24 \text{ MHz}$ , system clock= $f(X_{IN})$ .  $\phi$  is system clock divided by 2.
  - Data format: 1ST-8DATA-1SP
  - Parity bit is disabled
  - Re-transmit by transfer error is not performed
  - Use of CTS functions

The UART transmit is enabled by USB Endpoint1 OUT interrupt after USB data is received from the host PC. The UART transmit interrupt source is set for a trigger of DMA.

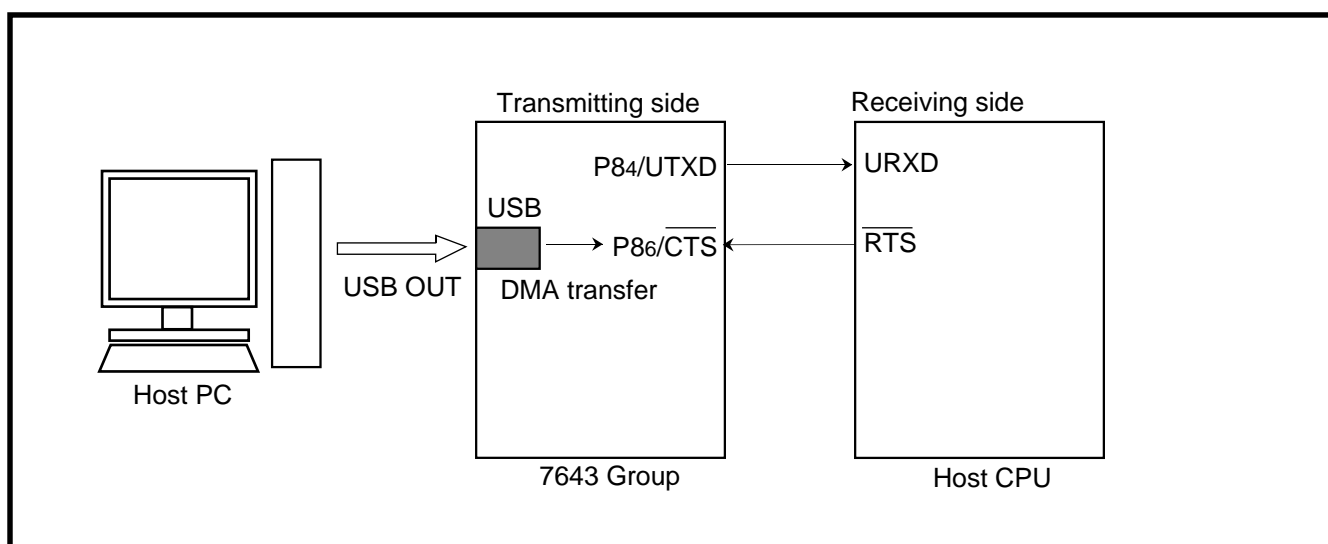
The first UART transmit interrupt source (UART transmit interrupt is not used) occurs by enable of UART transmit. By this UART transmit interrupt as a trigger of DMA, the data of USB FIFO is transmitted to the UART transmit/receive buffer register.

In the 7643 group, DMA interrupt occurs after DMA transmits the 64-byte data of USB FIFO to UART. In this time, the UART transmit is disabled and the OUT\_PKT\_RDY flag of USB Endpoint1 OUT is cleared to "0." As a result, the USB Endpoint1 can receive the following data.

Figure 2.4.24 shows a connection diagram.

Figure 2.4.25, 2.4.26 and 2.4.27 show register settings.

Figure 2.4.28 and 2.4.29 show control procedures.



**Fig. 2.4.24 Connection diagram**

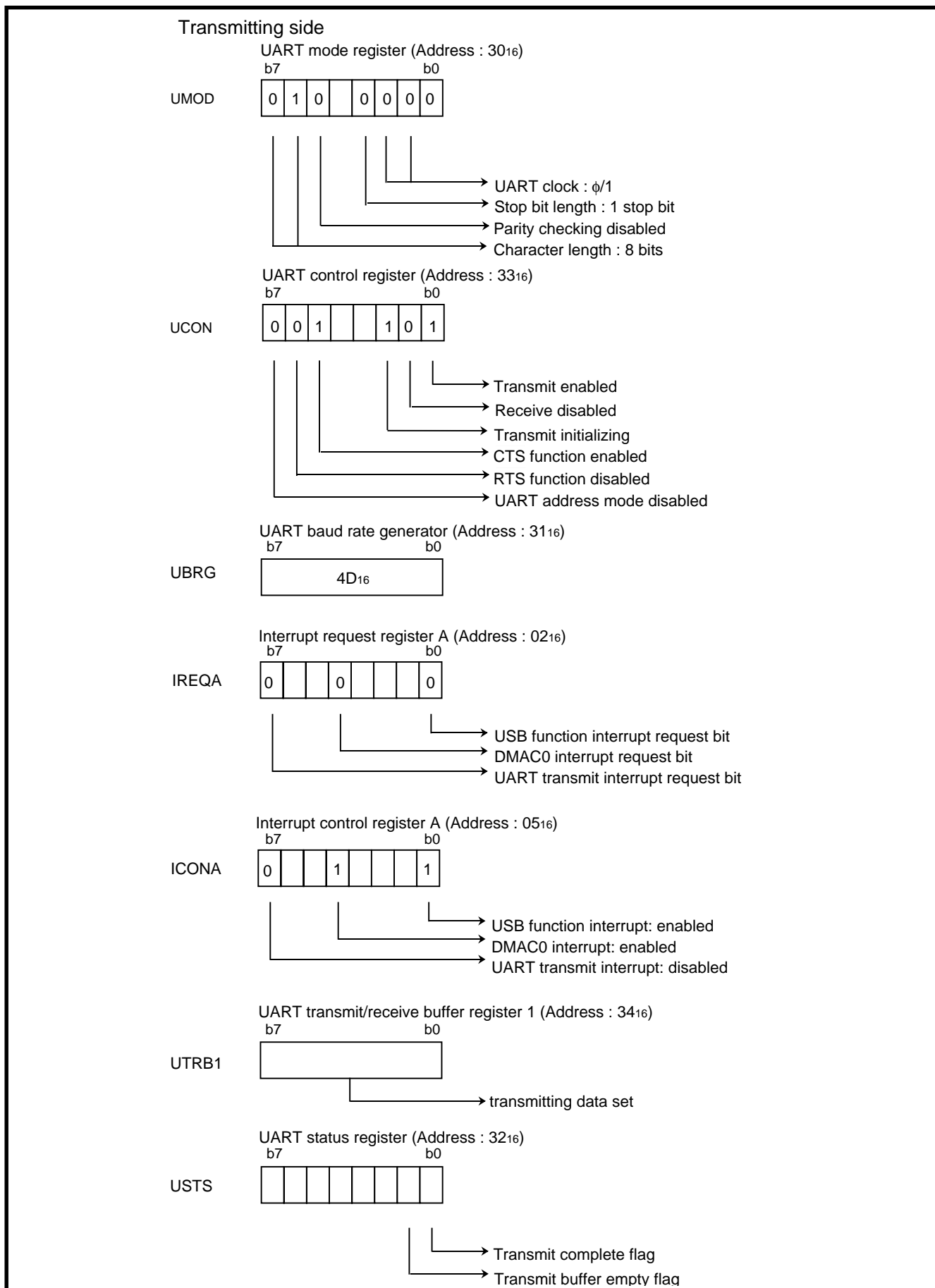


Fig. 2.4.25 Registers setting (1)

## Transmitting side

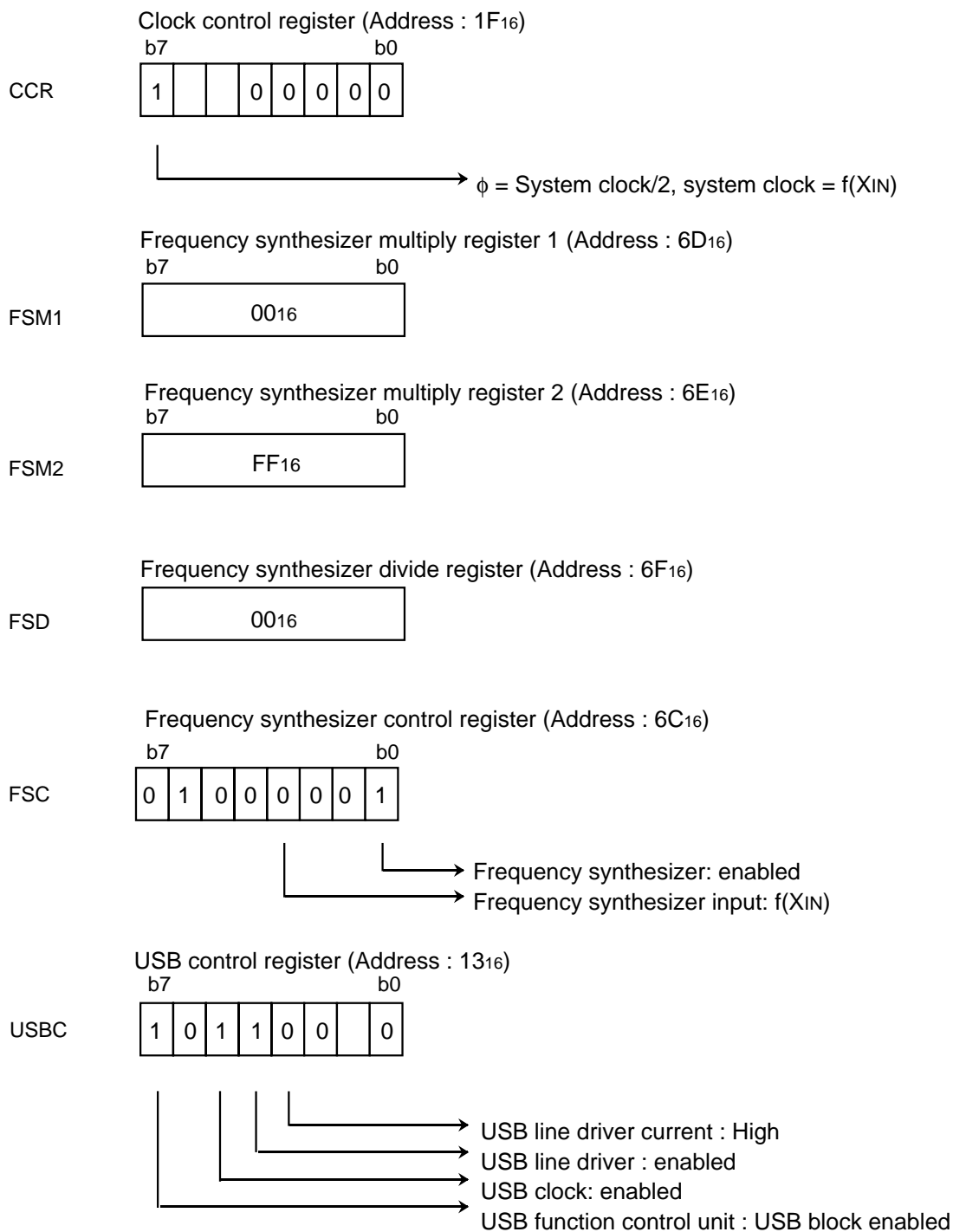


Fig. 2.4.26 Registers setting (2)

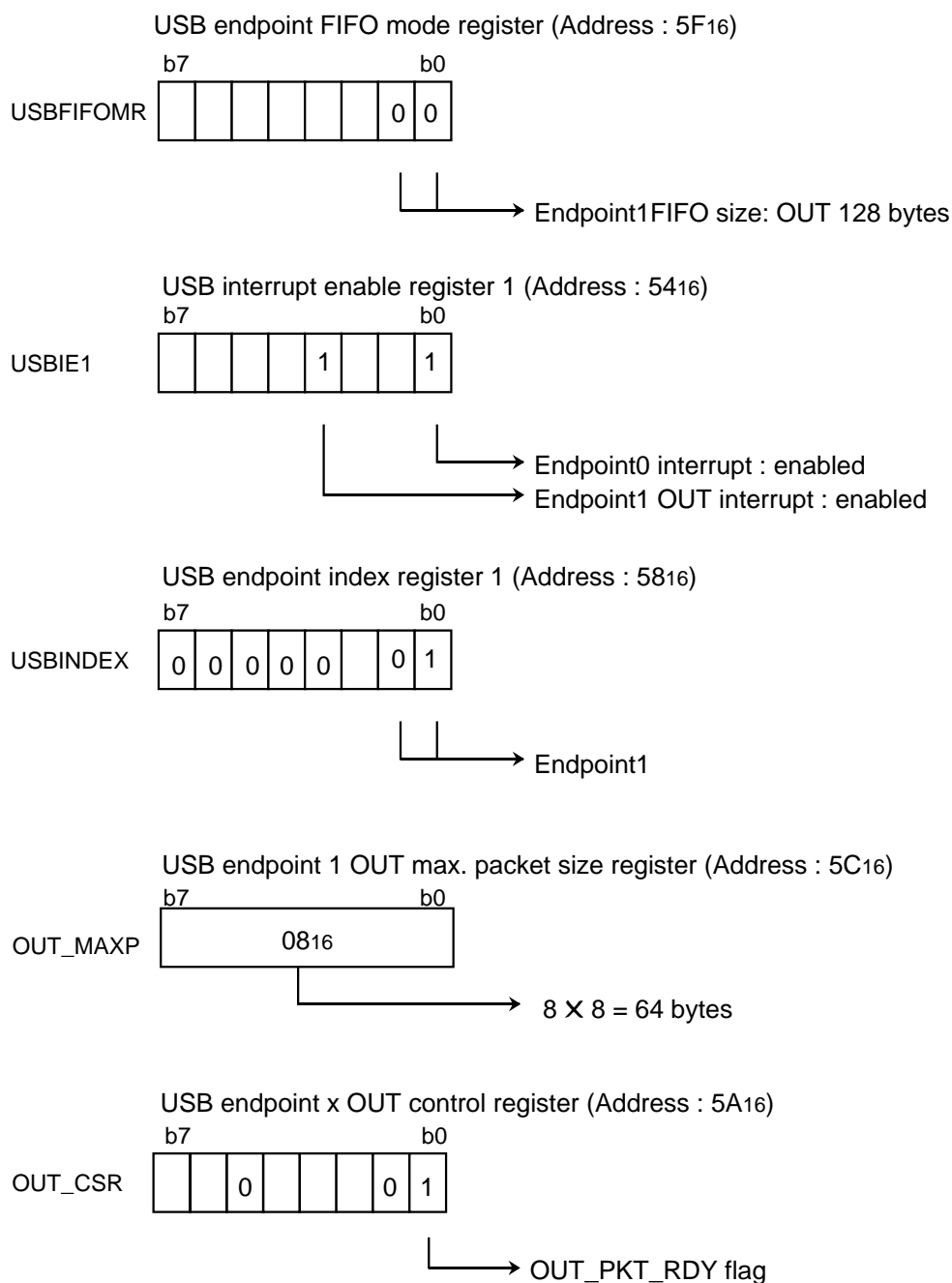


Fig. 2.4.27 Registers setting (3)



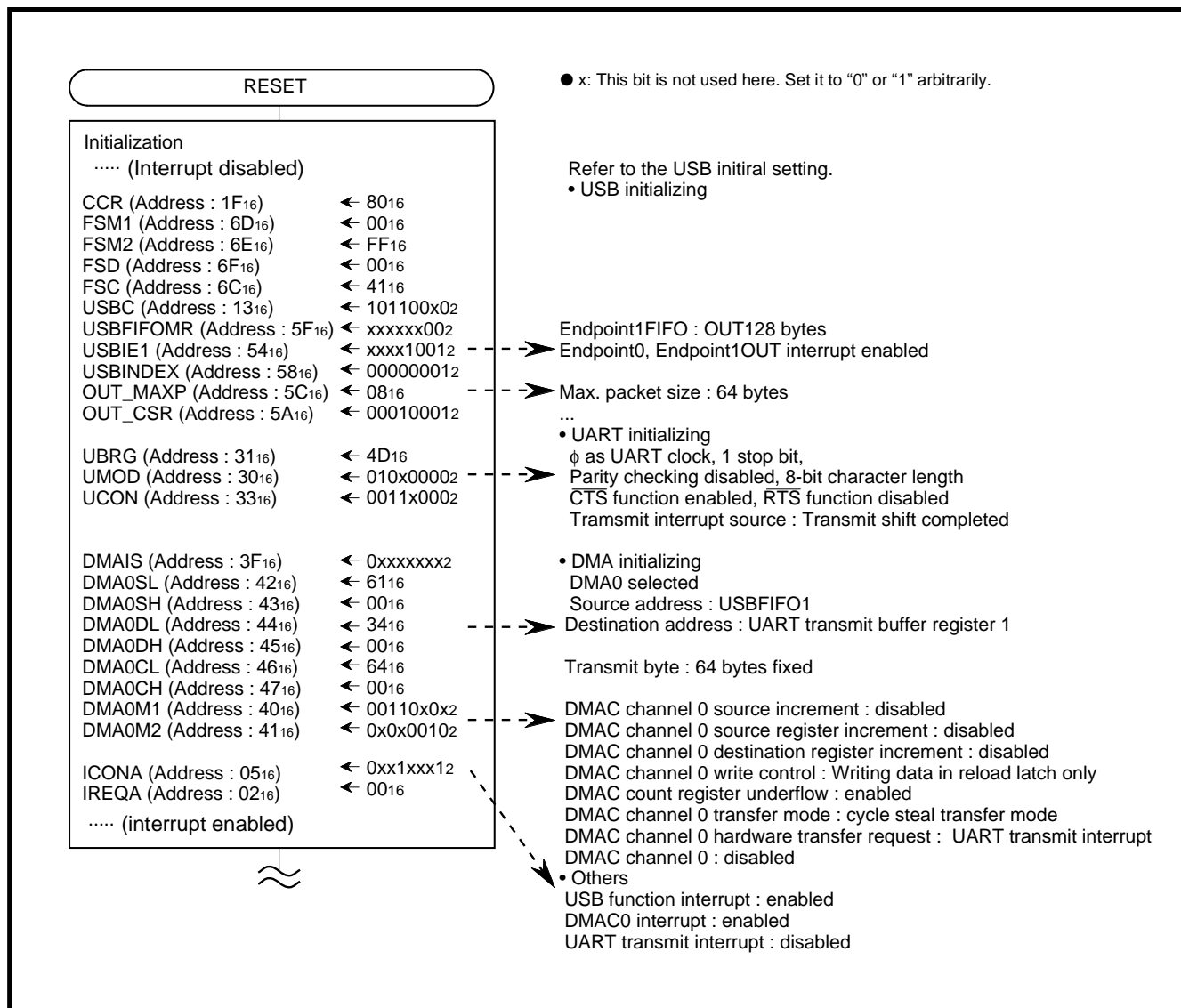


Fig. 2.4.28 Control procedure (1)

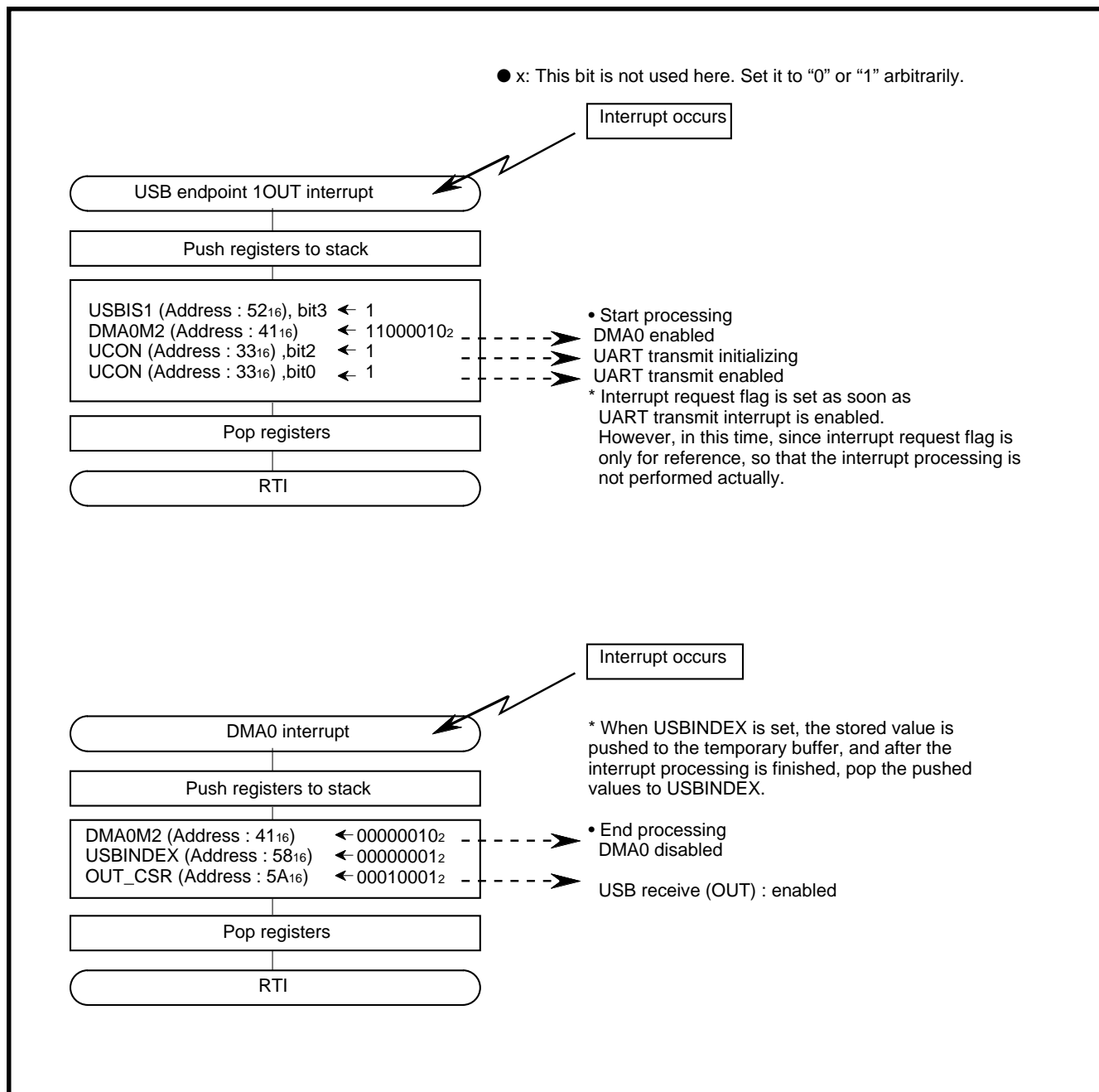


Fig. 2.4.29 Control procedure (2)

**(4) Data packet transfer (with no error processing) from UART to USB FIFO using DMA**

**Outline** : write the data received from host CPU by UART to USB FIFO by DMA, and transmit the data to host PC.

- Specifications** :
- USB Endpoint1: IN bulk transfer
  - USB Endpoint1: IN interrupt
  - USB Endpoint1 packet size: 64 Bytes
  - UART receive is used.
  - DMA cycle steal transfer mode (fixed address -> fixed address transfer)
  - DMA transfer unit : 64 bytes fixed (short packet un-supported)
  - UART receive buffer full interrupt request is used for DMA factor
  - Transfer bit rate : 9600 bps ( $\phi = 12 \text{ MHz}$  divided by 1248)
  - In this case,  $f(X_{IN})=24 \text{ MHz}$ , system clock= $f(X_{IN})$ .  $\phi$  is system clock divided by 2.
  - Data format: 1ST-8DATA-1SP
  - Parity bit is disabled
  - Re-transmit by transfer error is not performed
  - Use of  $\overline{\text{RTS}}$  functions

After the hardware reset, the UART receive and DMA are enabled in the initial setting. The UART receive interrupt source is set for a trigger of DMA.

When the UART receive is started, the UART receive interrupt source (UART receive buffer full interrupt and UART receive summing error interrupt are not used) occurs. By this UART receive interrupt as a trigger of DMA, the 1-byte data of UART transmit/receive buffer register is transferred to the USB FIFO.

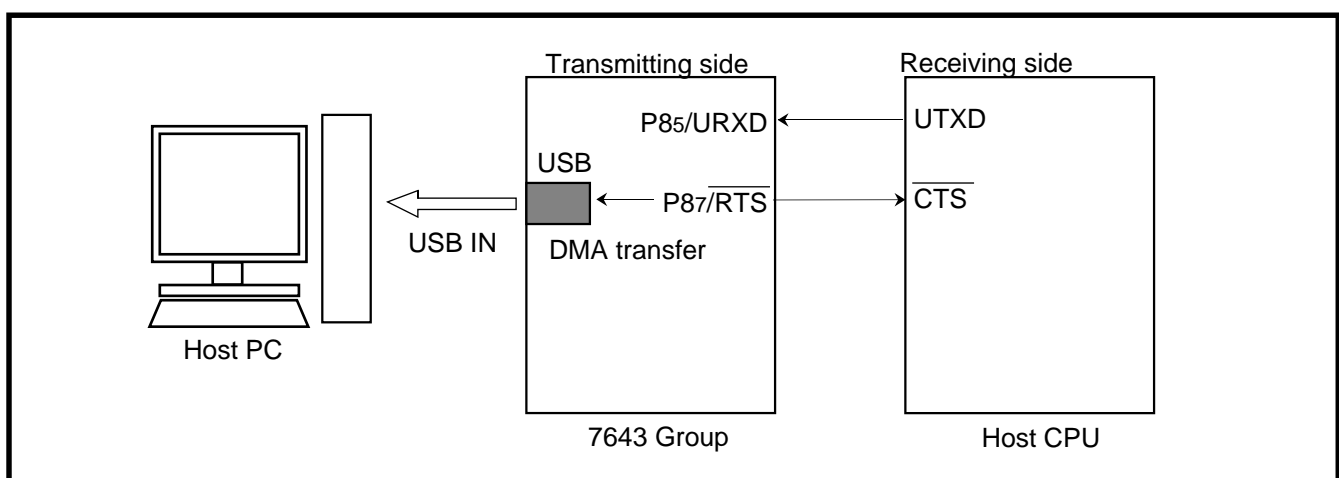
In the 7643 group, DMA interrupt occurs after DMA transfers the 64-byte UART receive data to USB FIFO. In this time, the DMA is disabled and the IN\_PKT\_RDY flag of USB Endpoint1 IN is set to "1." As a result, the USB Endpoint1 can transmit the data to host PC.

Meanwhile, in the 7643 Group, after the data of USB FIFO is transmitted to host PC, the USB Endpoint1 IN interrupt occurs. In this time, DMA is set to be enabled again and the DMA transfer of the next UART receive data to USB FIFO is started.

Figure 2.4.30 shows a connection diagram.

Figure 2.4.31, 2.4.32 and 2.4.33 show register settings.

Figure 2.4.34 and 2.4.35 show control procedures.



**Fig. 2.4.30 Connection diagram**

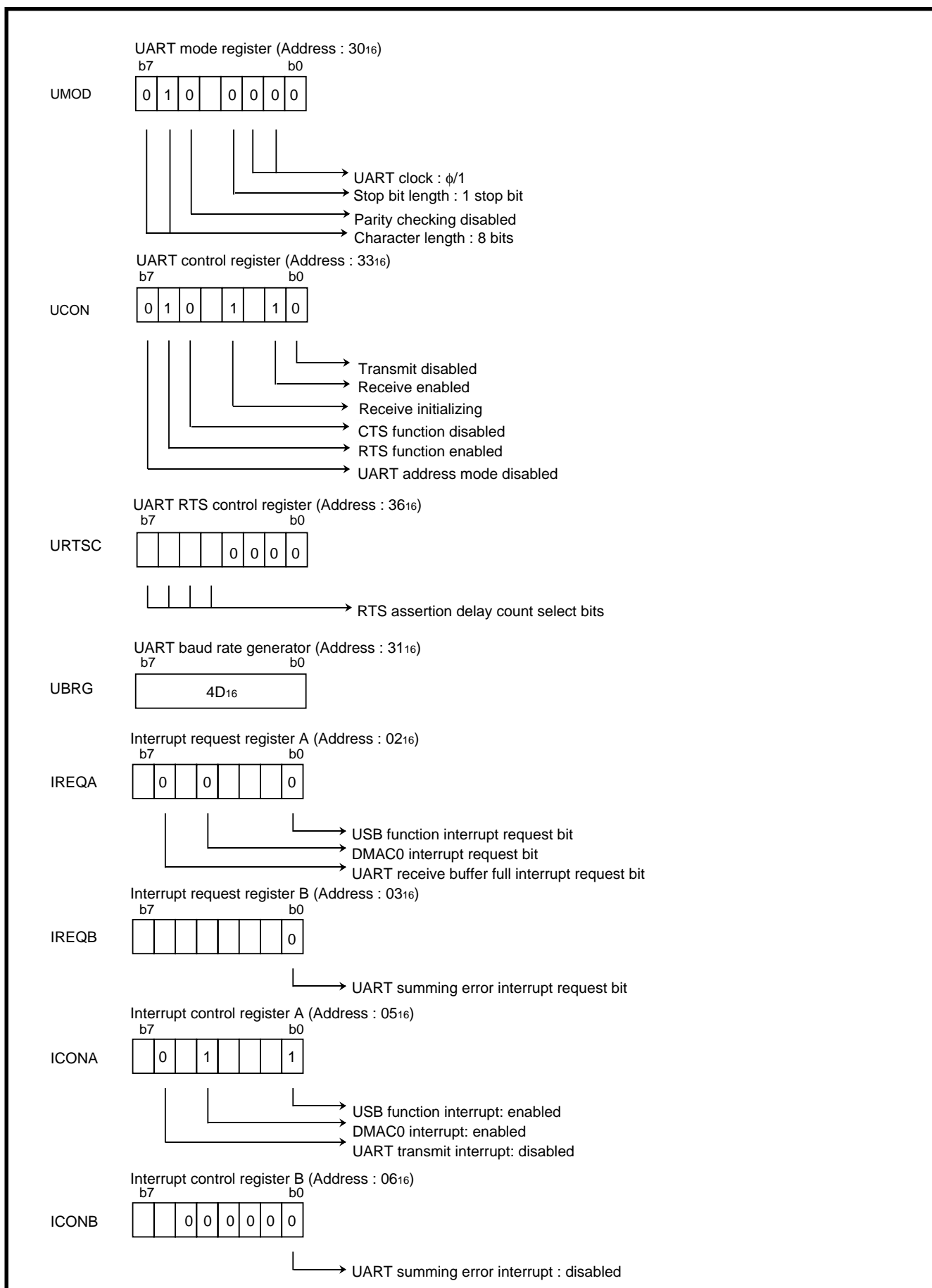


Fig. 2.4.31 Registers setting (1)

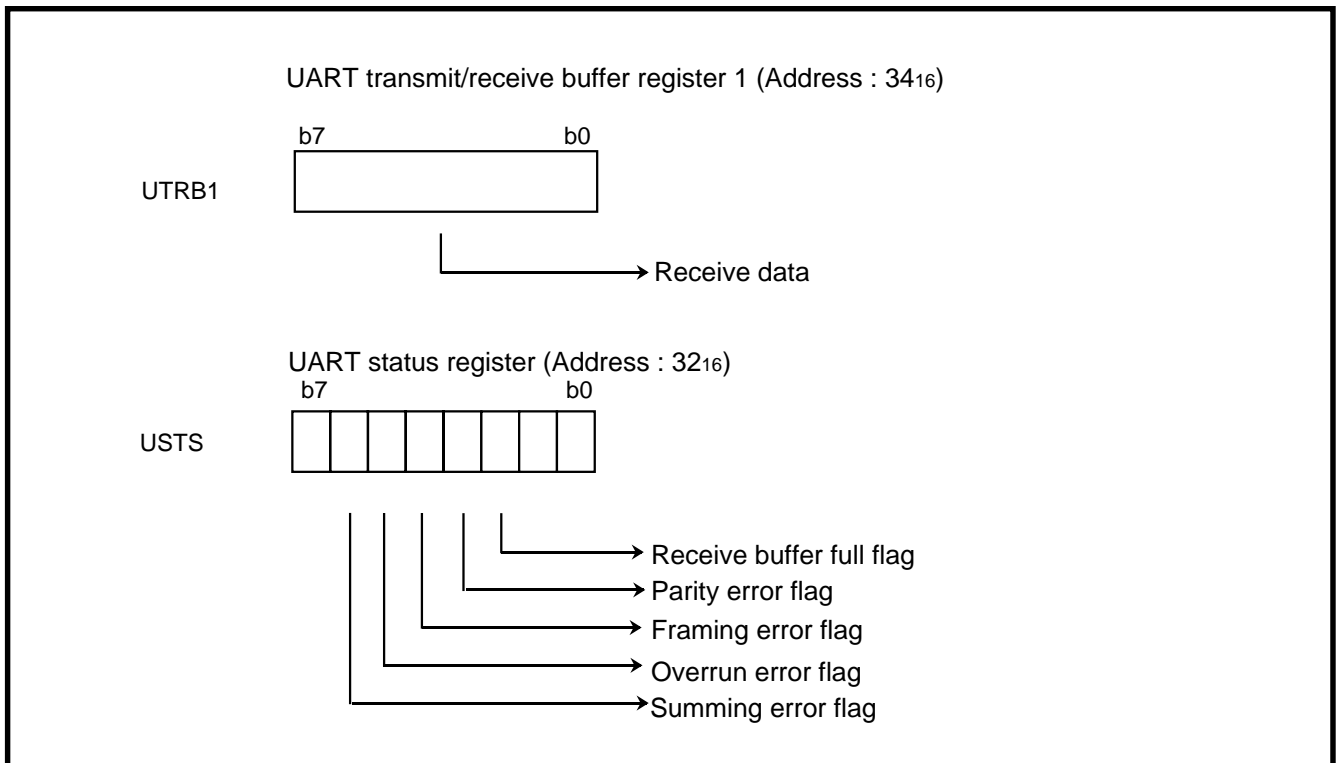


Fig. 2.4.32 Registers setting (2)

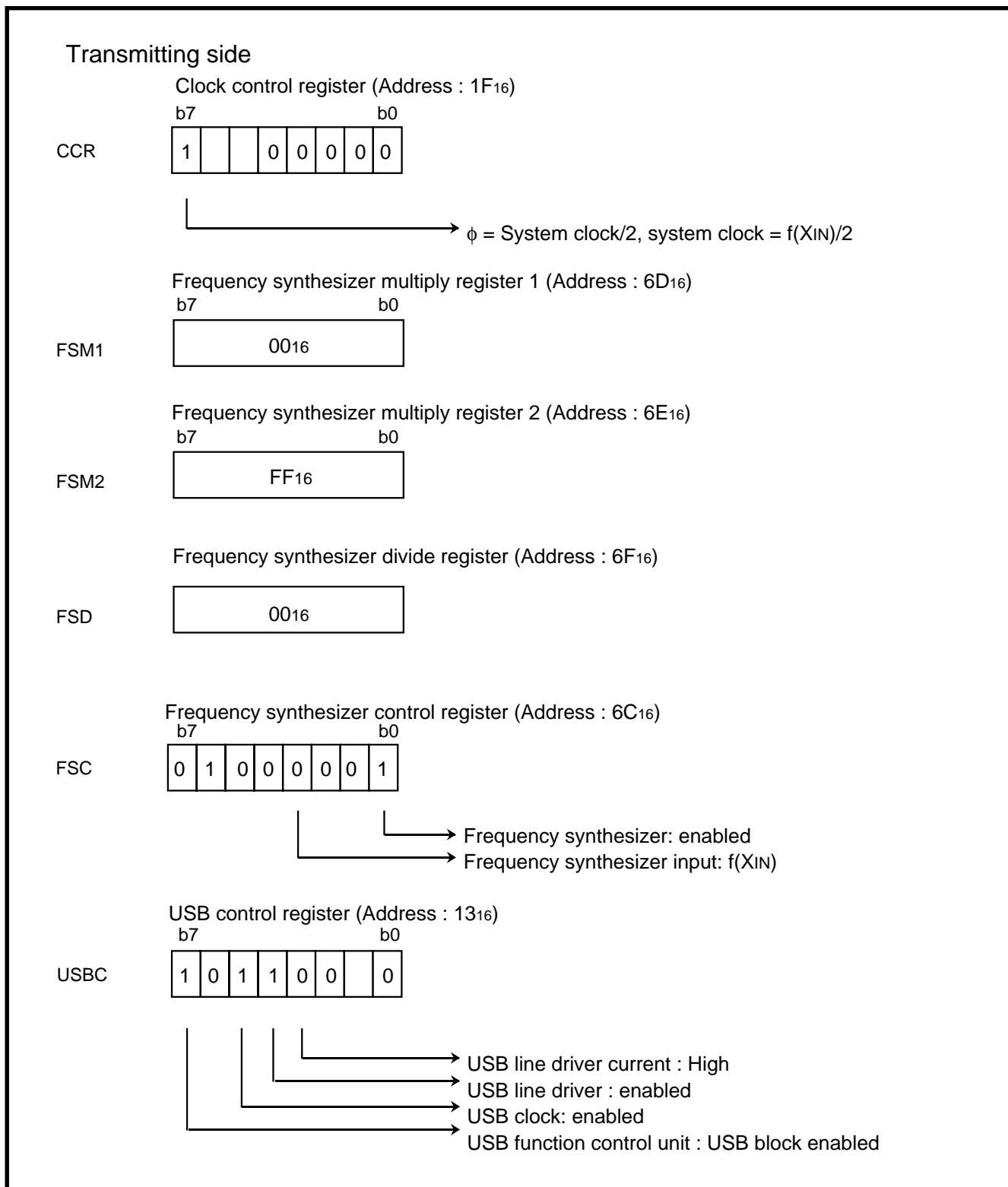


Fig. 2.4.33 Registers setting (3)

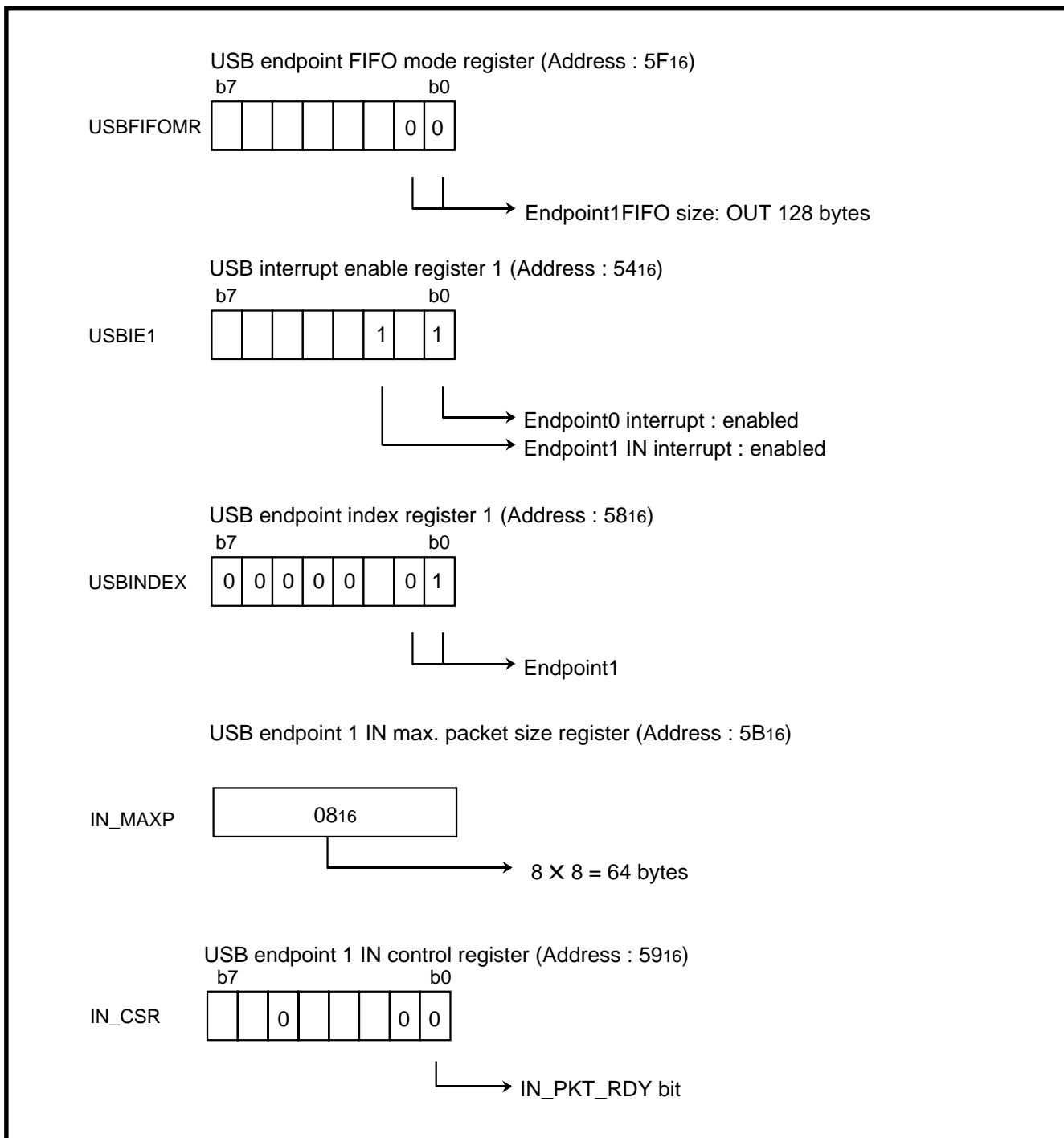


Fig. 2.4.34 Registers setting (4)

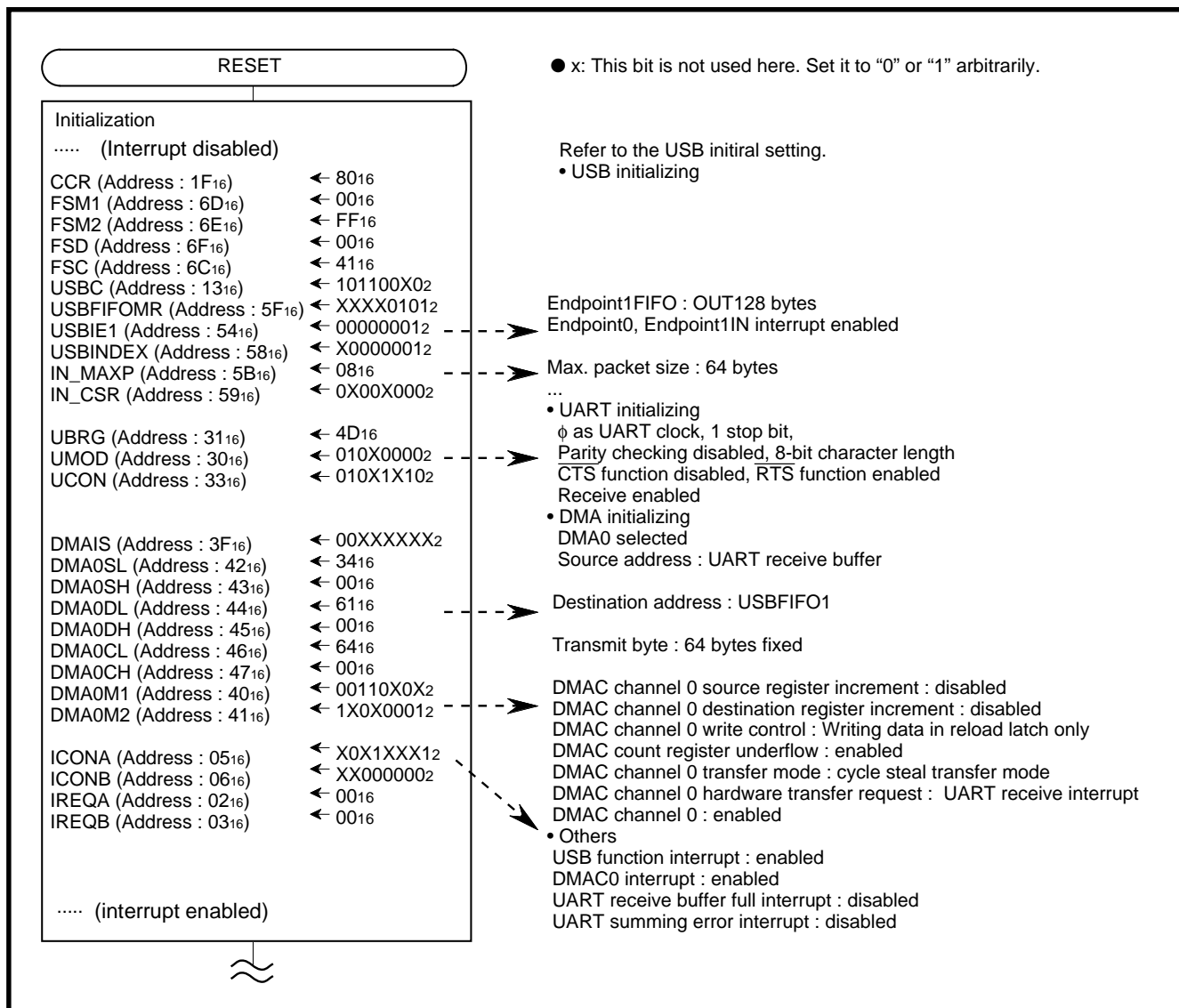


Fig. 2.4.35 Control procedure (1)



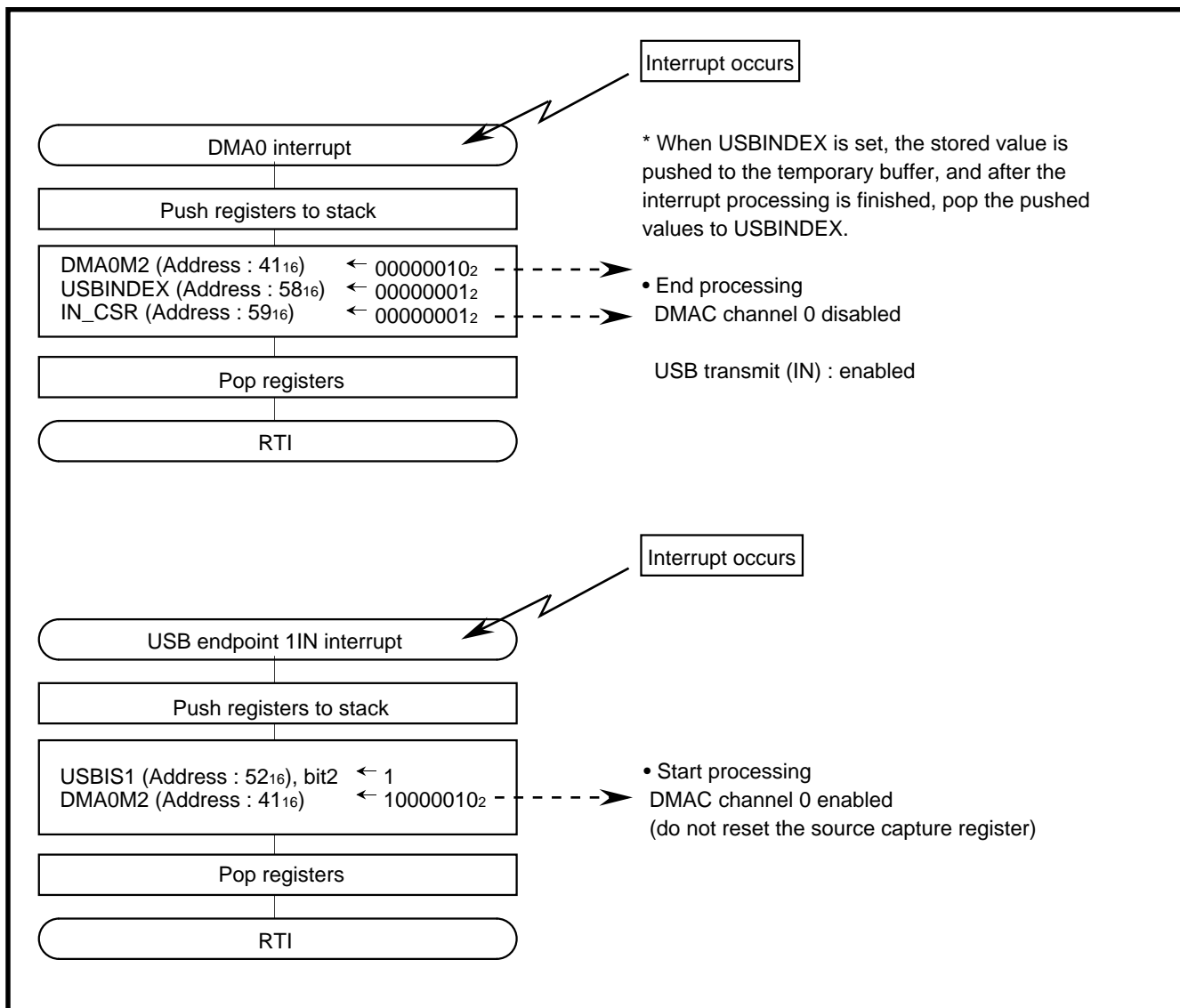


Fig. 2.4.36 Control procedure (2)

## 2.4.7 Notes on UART

### (1) Receive

- When any one of errors occurs, the summing error flag is set to “1” and the UART summing error interrupt request bit is also set to “1”. If a receive error occurs, the reception does not set the UART receive buffer full interrupt request bit to “1”.
- If the receive enable bit (REN) is set to “0” (disabled) while a data is being received, the receiving operation will stop after the data has been received.
- Setting the receive initialization bit (RIN) to “1” resets the UART RTS control register (URTS) to “80<sub>16</sub>”. After setting the RIN bit to “1”, set this URTS.

### (2) Transmit

- Once the transmission starts, it continues until the last bit has been transmitted even though clearing the transmit enable bit (TEN) to “0” (disabled) or inputting “H” to the  $\overline{\text{CTS}}$  pin. After completion of the current transmission, the transmission is disabled.
- The transmit complete flag (TCM) is changed from “1” to “0” later than 0.5 to 1.5 clocks of the shift clock. Accordingly, take it in consideration to transmit data confirming the TCM flag after the data is written into the transmit buffer register.

### (3) Register settings

- If updating a value of UART baud rate generator while the data is being transmitted or received, be sure to disable the transmission and reception before updating. If the former data remains in the UART transmit buffer registers 1 and 2 at retransmission, an undefined data might be output.
- The all error flags PER, FER, OER and SER are cleared to “0” when the UART status register is read, at the hardware reset or initialization by setting the Transmit Initialization Bit. These flags are also cleared to “0” by execution of bit test instructions such as **BBC** and **BCS**.
- The transmit buffer empty flag (TBE) is set to “0” when the low-order byte of transmitted data is written into the UART transmit buffer register 1. When using 9-bit character length, set the data into the UART transmit buffer register 2 (high-order byte) first before the UART transmit buffer register 1 (low-order byte).
- The receive buffer full flag (RBF) is set to “0” when the contents of UART receive buffer register 1 is read out. When using 9-bit character length, read the data from the UART receive buffer register 2 (high-order byte) first before the UART receive buffer register 1 (low-order byte).
- If a character bit length is 7 bits, bit 7 of the UART transmit/receive buffer register 1 and bits 0 to 7 of the UART transmit/receive buffer register 2 are ignored at transmitting; they are invalid at receiving.  
If a character bit length is 8 bits, bits 0 to 7 of the UART transmit/receive buffer register 2 are ignored at transmitting; they are invalid at receiving.  
If a character bit length is 9 bits, bits 1 to 7 of the UART transmit/receive buffer register 2 are ignored at transmitting; they are “0” at receiving.
- The reset cannot affect the contents of baud rate generator.

### (4) UART address mode

- When the MSB of the incoming data is “0” in the UART address mode, the receive buffer full flag (RBF) is set to “1”, but the receive buffer full interrupt request bit is not set to “1”.
- An overrun error cannot be detected after the first data has been received in UART address mode.
- The UART address mode can be used in either an 8-bit or 9-bit character length. 7-bit character length cannot be used.

**(5) Receive error flag**

The all error flags PER, FER, OER and SER are cleared to "0" when the UART status register is read, at the hardware reset or initialization by setting the Transmit Initialization Bit. Accordingly, note that these flags are also cleared to "0" by execution of bit test instructions such as BBC and BBS, not only LDA.

**(6)  $\overline{\text{CTS}}$  function**

When the CTS function is enabled, the transmitted data is not transferred to the transmit shift register until "L" is input to the  $\overline{\text{CTS}}$  pin ( $\text{P8}_6/\overline{\text{CTS}}$ ). As the result, do not set the following data to the transmit buffer register.

**(7)  $\overline{\text{RTS}}$  function**

- If the start bit is detected in the term of "H" assertion of  $\overline{\text{RTS}}$ , its assertion count is suspended and the RTS pin remains "H" output. After receiving the last stop bit, the count is resumed.
- Setting the receive initialization bit (RIN) to "1" resets the UART RTS control register (URTS) to "80<sub>16</sub>". After setting the RIN bit to "1", set this URTS.

**(8) Interrupt**

- When setting the transmit initialization bit (TIN) to "1", both the transmit buffer empty flag (TBE) and the transmit complete flag (TCM) are set to "1", so that the transmit interrupt request occurs independent of its interrupt source. After setting the transmit initialization bit (TIN) to "1", clear the transmit interrupt request bit to "0" before setting the transmit enable bit (TEN) to "1".
- The transmit interrupt request bit is set and the interrupt request is generated by setting the transmit enable bit (TIN) to "1" even when selecting timing that either of the following flags is set to "1" as timing where the transmission interrupt is generated:
  - (1) Transmit buffer empty flag is set to "1"
  - (2) Transmit complete flag is set to "1".

Therefore, when the transmit interrupt is used, set the transmit interrupt enable bit to transmit enabled as the following sequence:

- (1) Transmit enable bit is set to "1"
- (2) Transmit interrupt request bit is set to "0"
- (3) Transmit interrupt enable bit is set to "1".

## 2.5 DMAC

This paragraph explains the registers setting method and the notes related to the DMAC.

### 2.5.1 Memory map

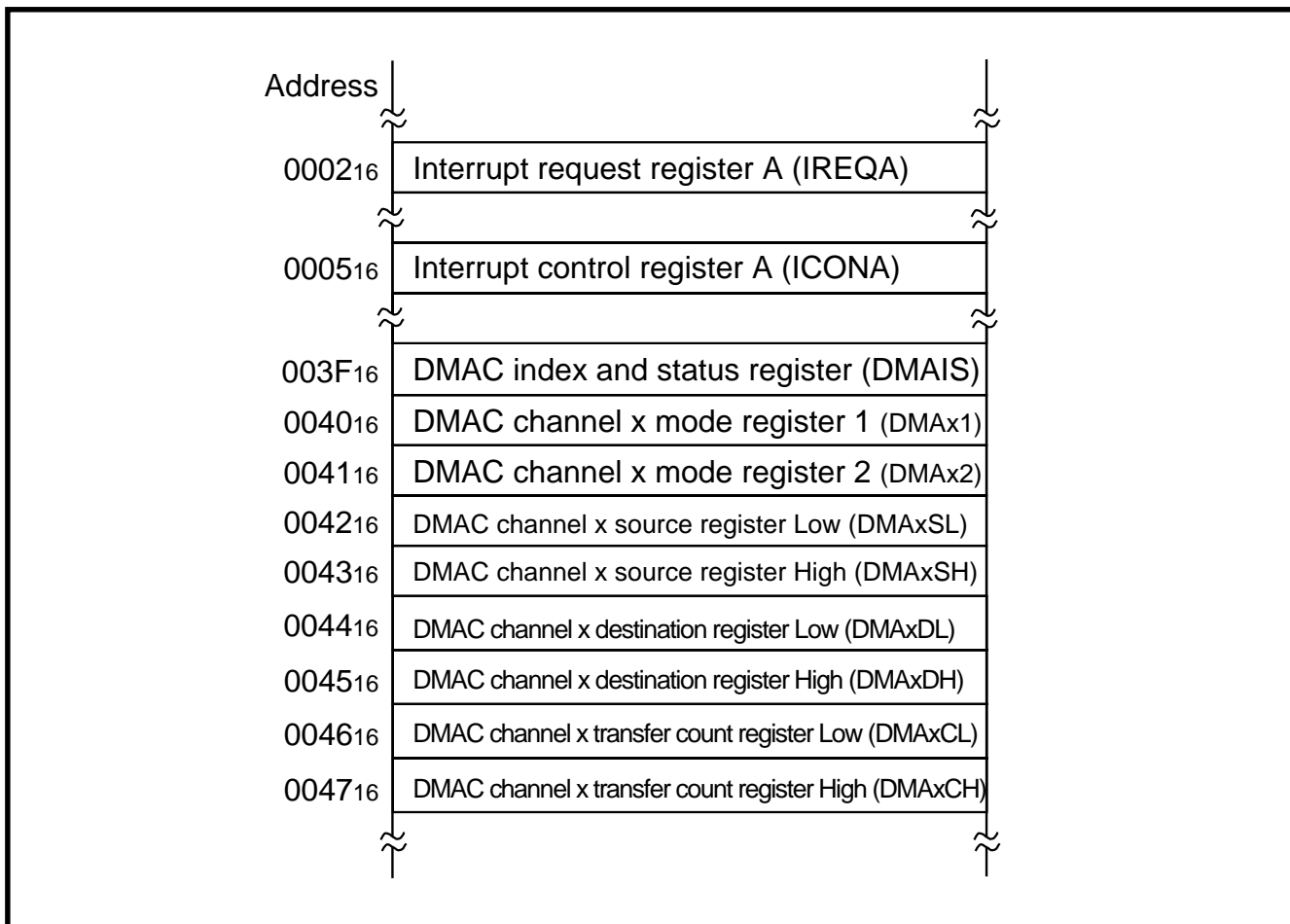


Fig. 2.5.1 Memory map of registers related to DMAC

## 2.5.2 Related registers

### (1) DMAC index and status register

#### •DMAC channel x (x = 0, 1) count register underflow flag (DxUF)

When the corresponding transfer count register Low (address 46<sub>16</sub>) underflows, this DxUF flag is set to "1". Writing "0" into this flag clears it.

#### •DMAC channel x (x = 0, 1) suspend flag (DxSFI)

When an interrupt routine is processed during any DMA operation, the transfer operation is suspended and the DMAC automatically sets the corresponding DxSFI flag to "1". As soon as the CPU completes the interrupt operation, the DMAC clears the DxSFI flag to "0" and resumes the original operation from the point where it was suspended.

#### •DMAC transfer suspend control bit (DTSC)

This bit specifies the transfer mode which can be suspended by an interrupt process.

#### •DMAC register reload disable bit (DRLDD)

If the DRLDD bit is "1", when the DMAC channel x transfer count register underflows, the DMAC channel x source registers and destination registers are disabled from being reloaded from their latches.

#### •Channel index bit (DCI)

The related registers of channels 1 and 2 are assigned on the same SFR addresses. This DCI bit specifies the accessible channel.

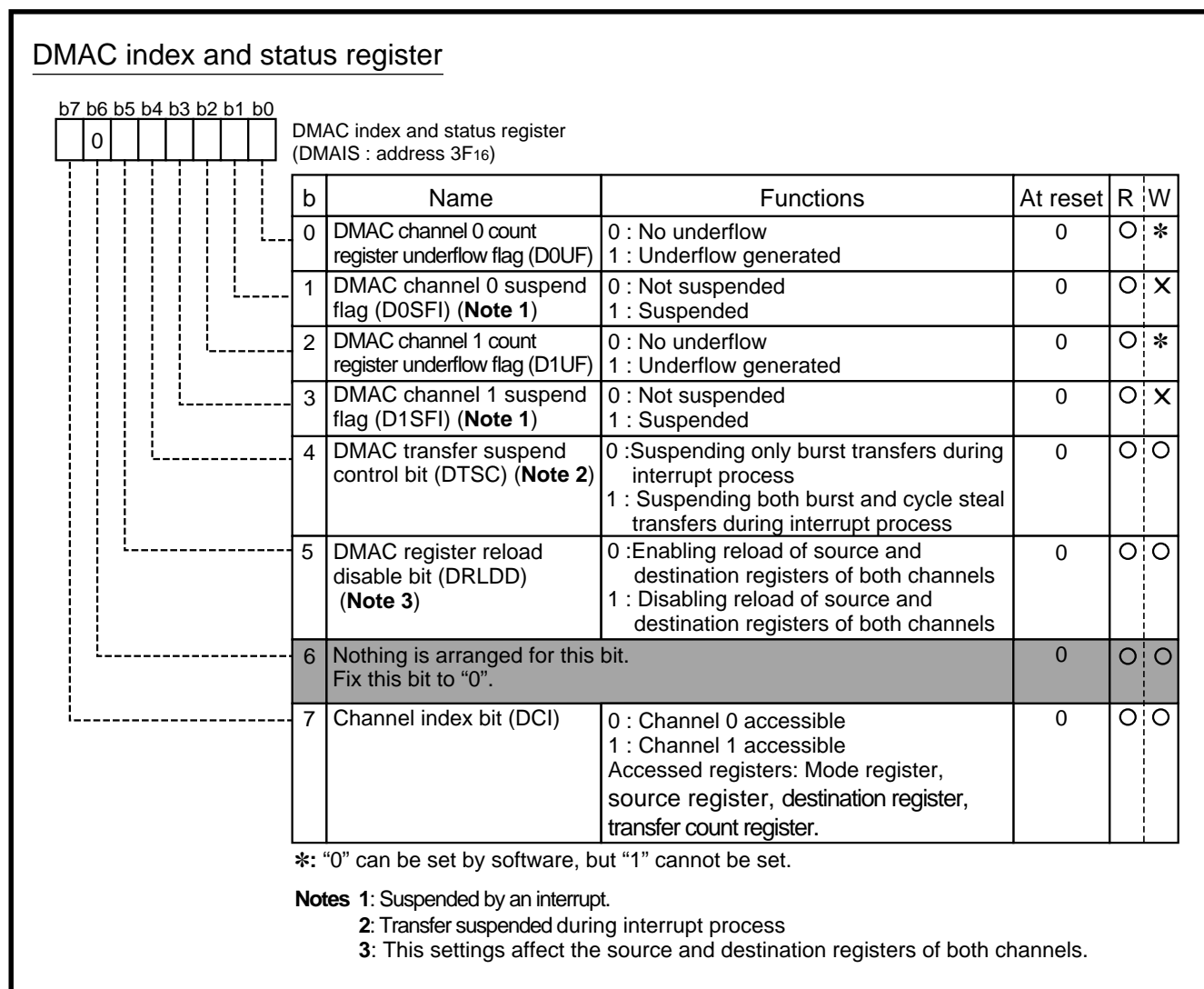


Fig. 2.5.2 Structure of DMAC index and status register

**(2) DMAC channel x (x = 0, 1) mode register 1**

- DMAC channel x source register increment/decrement selection bit (DxSRID)
- DMAC channel x source register increment/decrement enable bit (DxSRCE)
- DMAC channel x destination register increment/decrement selection bit (DxDRID)
- DMAC channel x destination register increment/decrement enable bit (DxDRCE)

These bits select that the DMAC channel X source registers and destination registers are either decreased or increased by 1 after transfer completion.

- DMAC channel x data write control bit (DxDWC)

The DxDWC bit controls write operation to the following registers and their latches: Low and High bytes of DMAC channel x source registers, destination registers and transfer count registers. When the DxDWC bit is "0", data is simultaneously written into each latch and register. When this bit is "1", data is written only into their latches.

- DMAC channel x disable after count register underflow enable bit (DxDAUE)

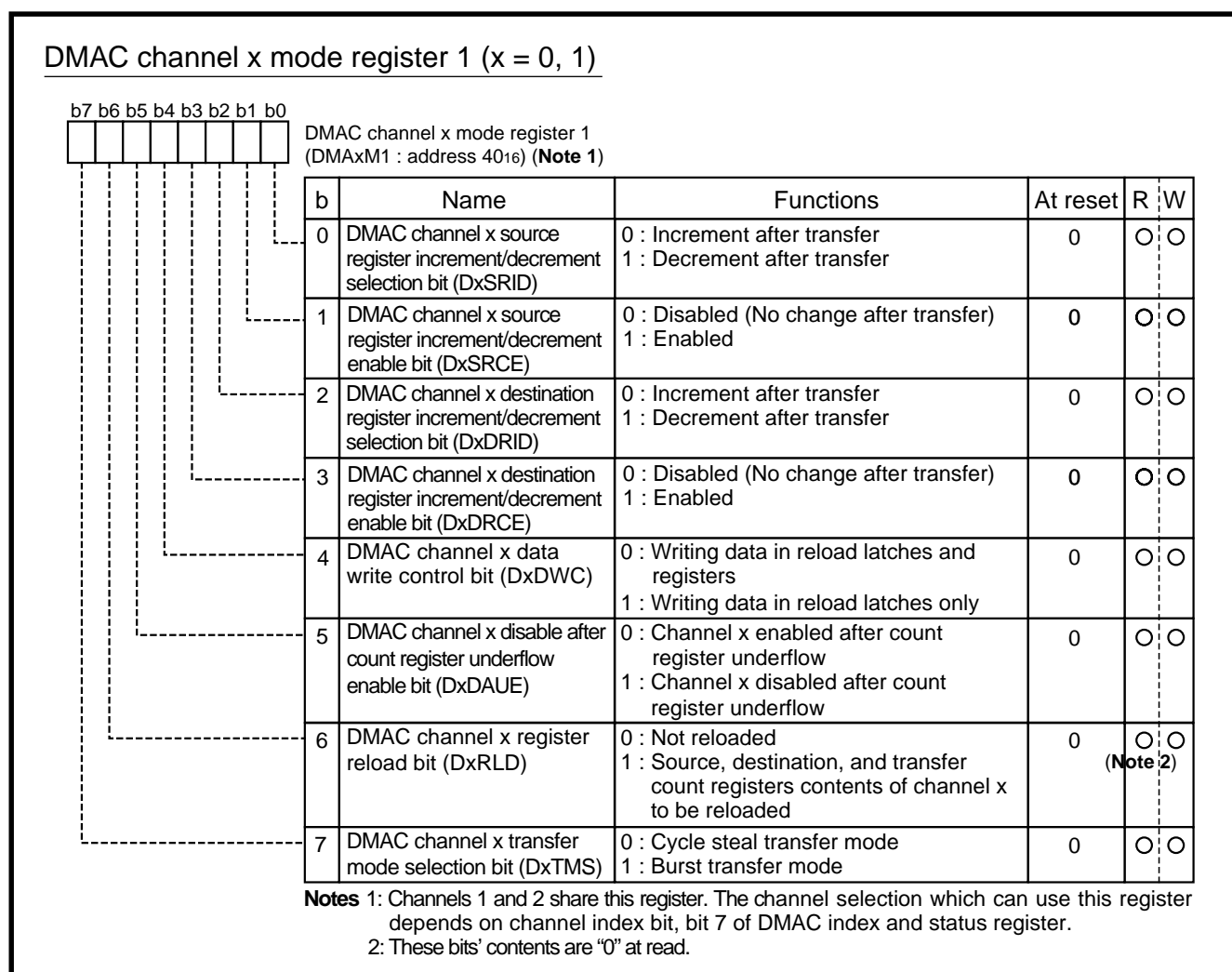
When the DxDAUE bit is "1", after the DMAC channel x transfer count register Low underflows the corresponding channel x is disabled. The DMAC channel x enable bit (DxCEN, bit 7 of DMAxM2) goes to "0" at the same time.

- DMAC channel x register reload bit (DxRLD)

Writing "1" to the DxRLD bit can update the DMAC channel x source registers, destination registers and transfer count registers with the values in their respective latches. It can be performed at anytime. This bit is fixed to "0" at read.

- DMAC channel x transfer mode selection bit (DxTMS)

The DxTMS bit selects the transfer mode.



**Fig. 2.5.3 Structure of DMAC channel x (x = 0, 1) mode register 1**

**(3) DMAC channel x (x = 0, 1) mode register 2****•DMAC channel x software transfer trigger bit (DxSWT)**

Writing "1" to the DxSWT bit can generate a transfer request as a software trigger. If all of DMACx hardware transfer request source bits (DxHR) are "0", the software trigger is only transfer request factor. This bit is fixed to "0" at read.

**•DMAC channel x transfer initiation source capture register reset bit (DxCRR)**

Writing "1" to the DxCRR bit can reset the transfer initiation source capture register. The request of the transfer initiation source is latched asynchronously and it is sampled into the transfer initiation source capture register at a rising edge of  $\phi$ . This bit is fixed to "0" at read.

**•DMAC channel x enable bit (DxCEN)**

The DMAC channel x is enabled by setting this bit to "1". When clearing this to "0", the DMA transfer is finished.

## DMAC channel 0 mode register 2

b	Name	Functions	At reset	R	W
0	DMAC channel 0 hardware transfer request source bits (D0HR)	b3b2b1b0 0 0 0 0 : Not used 0 0 0 1 : UART receive interrupt 0 0 1 0 : UART transmit interrupt 0 0 1 1 : Not used 0 1 0 0 : INT <sub>0</sub> interrupt 0 1 0 1 : USB endpoint 1 IN_PKT_RDY signal (falling edge active) 0 1 1 0 : USB endpoint 2 IN_PKT_RDY signal (falling edge active) 0 1 1 1 : Not used 1 0 0 0 : USB endpoint 1 OUT_PKT_RDY signal (rising edge active) 1 0 0 1 : USB endpoint 1 OUT_FIFO_NOT_EMPTY signal (rising edge active) 1 0 1 0 : USB endpoint 2 OUT_PKT_RDY signal (rising edge active) 1 0 1 1 : Not used 1 1 0 0 : Not used 1 1 0 1 : Not used 1 1 1 0 : Serial I/O transmit/receive interrupt 1 1 1 1 : Not used	0	○	○
1			0	○	○
2			0	○	○
3			0	○	○
4	DMAC channel 0 software transfer trigger (D0SWT)	0 : No action 1 : Request of channel 0 transfer by writing "1"	0	○	○ (Note 2)
5	Nothing is arranged for this bit. Fix this bit to "0".		0	○	○
6	DMAC channel 0 transfer initiation source capture register reset bit (D0CRR)	0 : No action 1 : Reset of channel 0 capture register by writing "1"	0	○	○ (Note 2)
7	DMAC channel 0 enable bit (D0CEN)	0 : Channel 0 disabled 1 : Channel 0 enabled (Note 3)	0	○	○

**Notes 1:** DMAC channel 0 mode register 2 and DMAC channel 1 mode register 2 are assigned at the same address 4116. The accessible register depends on channel index bit, bit 7 of DMAC index and status register.

**2:** These bits' contents are "0" at read. This bit is automatically cleared to "0" after writing "1".

**3:** When setting this bit to "1", simultaneously set the DMAC channel 0 transfer initiation source capture register reset bit (bit 6 of DMA0M2) to "1".

Fig. 2.5.4 Structure of DMAC channel 0 mode register 2



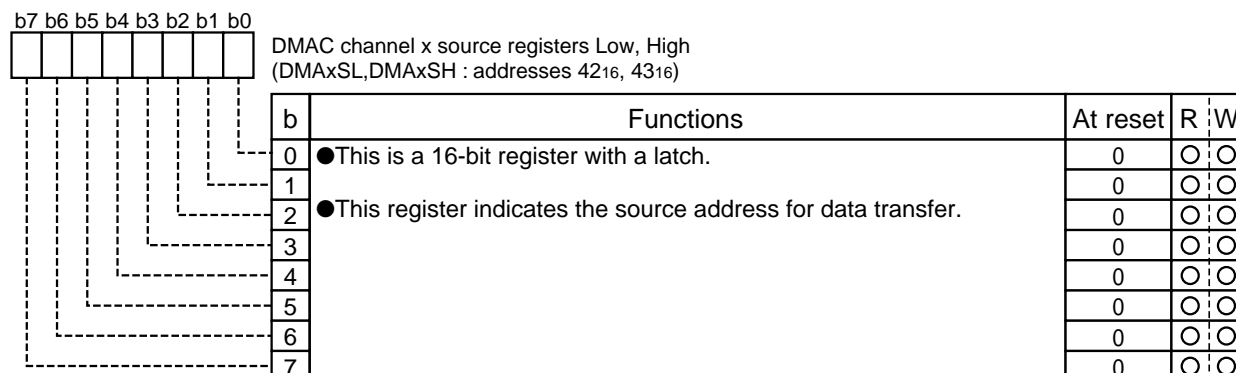
## DMAC channel 1 mode register 2

b	Name	Functions	At reset	R	W
0	DMAC channel 1 hardware transfer request source bits (D1HR)	b3b2b1b0 0 0 0 0 : Not used 0 0 0 1 : Not used 0 0 1 0 : Not used 0 0 1 1 : Not used 0 1 0 0 : INT <sub>1</sub> interrupt 0 1 0 1 : USB endpoint 1 IN_PKT_RDY signal (falling edge active) 0 1 1 0 : USB endpoint 2 IN_PKT_RDY signal (falling edge active) 0 1 1 1 : Not used 1 0 0 0 : USB endpoint 1 OUT_PKT_RDY signal (rising edge active) 1 0 0 1 : USB endpoint 1 OUT_FIFO_NOT_EMPTY signal (rising edge active) 1 0 1 0 : USB endpoint 2 OUT_PKT_RDY signal (rising edge active) 1 0 1 1 : Not used 1 1 0 0 : Not used 1 1 0 1 : Not used 1 1 1 0 : Timer 1 interrupt 1 1 1 1 : Not used	0	○	○
1			0	○	○
2			0	○	○
3			0	○	○
4	DMAC channel 1 software transfer trigger (D1SWT)	0 : No action 1 : Request of channel 1 transfer by writing "1"	0	○	○ (Note 2)
5	Nothing is arranged for this bit. Fix this bit to "0".		0	○	○
6	DMAC channel 1 transfer initiation source capture register reset bit (D1CRR)	0 : No action 1 : Reset of channel 1 capture register by writing "1"	0	○	○ (Note 2)
7	DMAC channel 1 enable bit (D1CEN)	0 : Channel 1 disabled 1 : Channel 1 enabled (Note 3)	0	○	○

- Notes**
- 1:** DMAC channel 0 mode register 2 and DMAC channel 1 mode register 2 are assigned at the same address 4116. The accessible register depends on channel index bit, bit 7 of DMAC index and status register.
- 2:** These bits' contents are "0" at read. This bit is automatically cleared to "0" after writing "1".
- 3:** When setting this bit to "1", simultaneously set the DMAC channel 1 transfer initiation source capture register reset bit (bit 6 of DMA1M2) to "1".

Fig. 2.5.5 Structure of DMAC channel 1 mode register 2

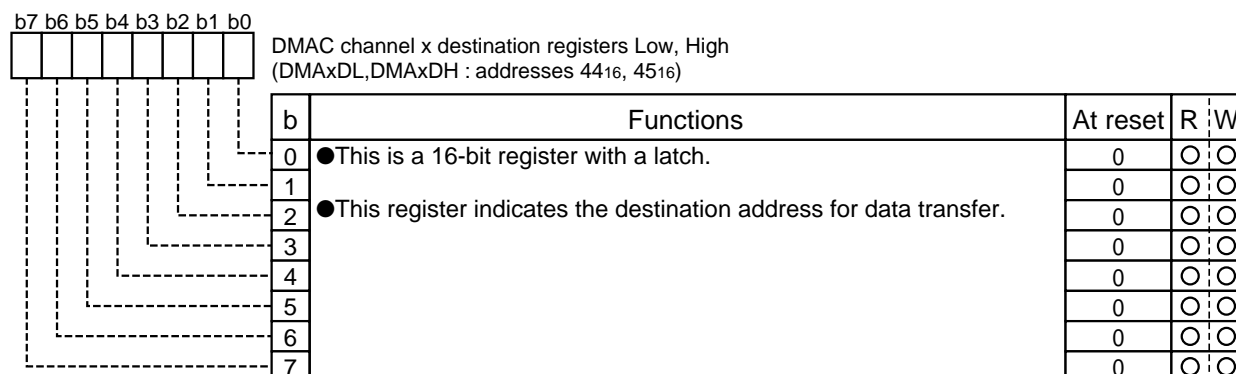
### DMAC channel x source registers Low, High



- Notes**
- 1: DMAC channel x source registers low, high of channels 0 and 1 are assigned at the same addresses. The accessible register depends on channel index bit, bit 7 of DMAC index and status register.
  - 2: Write data into the lower bytes first, and then the higher bytes.
  - 3: Read the contents from the higher bytes first, and then the lower bytes.

Fig. 2.5.6 Structure of DMAC channel x source registers Low, High

### DMAC channel x destination registers Low, High



- Notes**
- 1: DMAC channel x destination registers low, high of channels 0 and 1 are assigned at the same addresses. The accessible register depends on channel index bit, bit 7 of DMAC index and status register.
  - 2: Write data into the lower bytes first, and then the higher bytes.
  - 3: Read the contents from the higher bytes first, and then the lower bytes.

Fig. 2.5.7 Structure of DMAC channel x destination registers Low, High

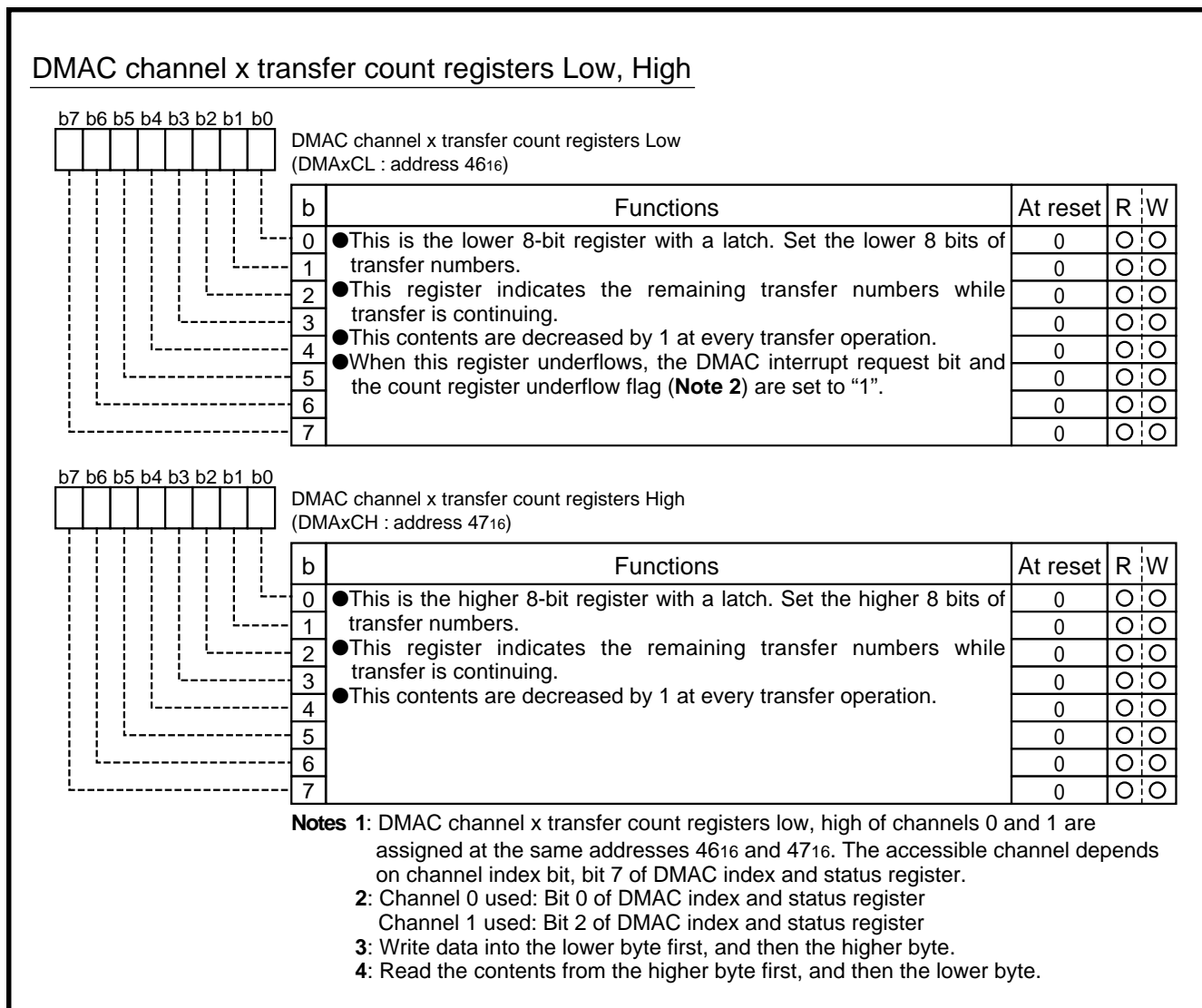


Fig. 2.5.8 Structure of DMAC channel x transfer count registers Low, High

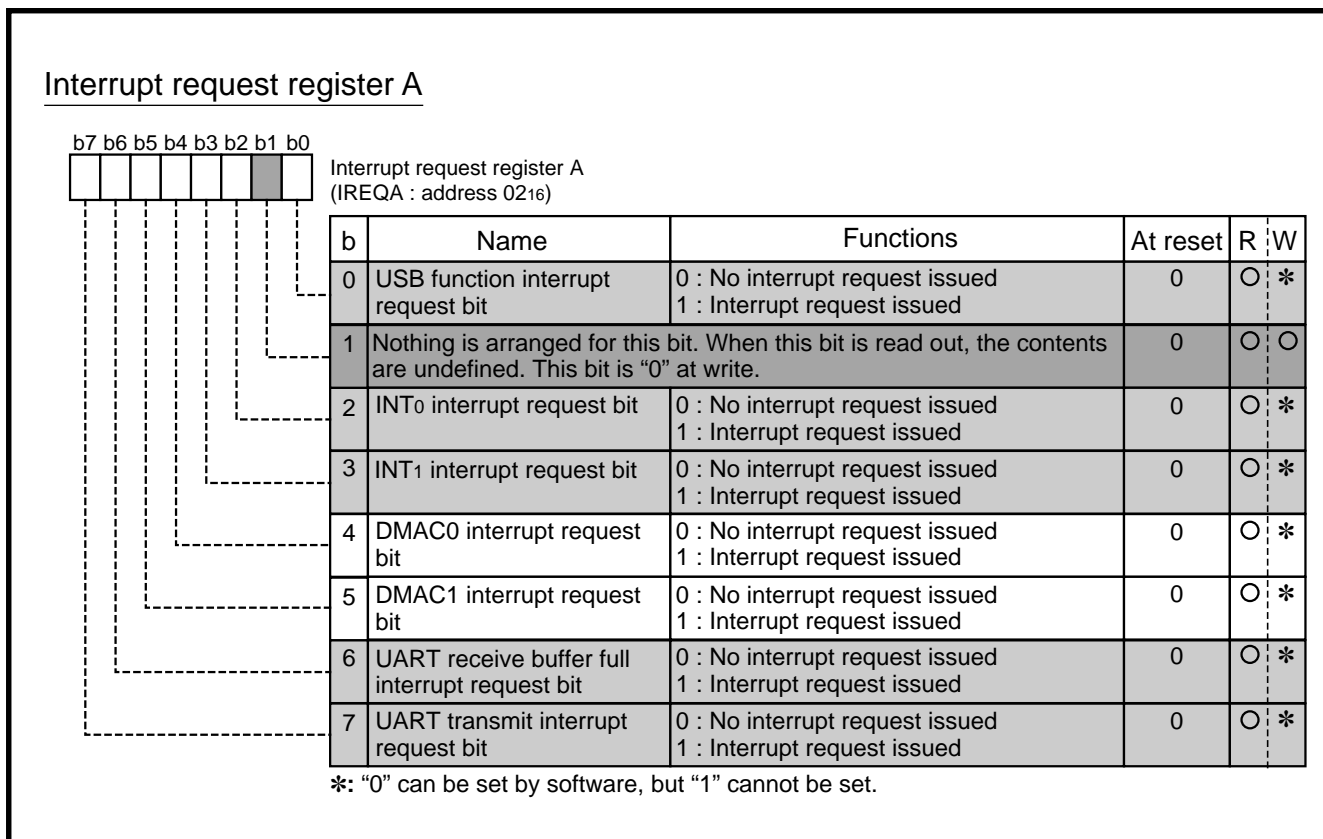


Fig. 2.5.9 Structure of Interrupt request register A

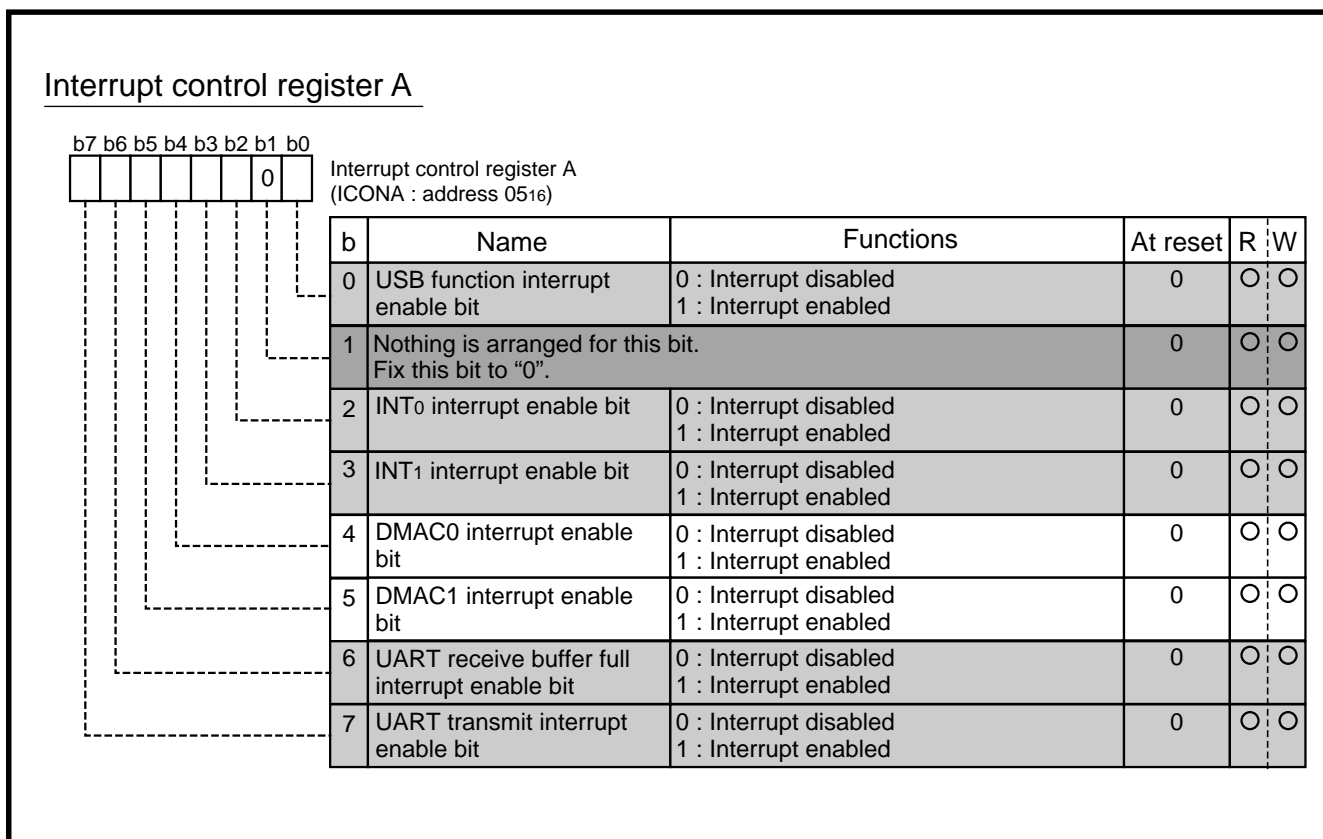


Fig. 2.5.10 Structure of Interrupt control register A

### 2.5.3 DMAC operation description

The DMAC transfers data using the bus without use of the CPU. The DMAC consists of DMAC0 and DMAC1, which have the same function each.

There are two transfer modes: Burst transfer mode or Cycle steal transfer mode.

#### •Burst transfer mode

Once a DMA transfer request is accepted, an entire batch of data is transferred. The right to use bus is not returned to the CPU until the transfer of all data has been completed.

The DMAC transfers the number of bytes data specified by the transfer count register for each request. The count register is a 16-bit counter; the maximum number of data is 65,536 bytes per one request.

#### •Cycle steal mode

The DMAC transfers one byte of data for each request. If one byte transfer has been completed and then a DMA transfer request is not generated, the right to use bus is returned to the CPU.

Figure 2.5.11 shows the transfer mode overview.

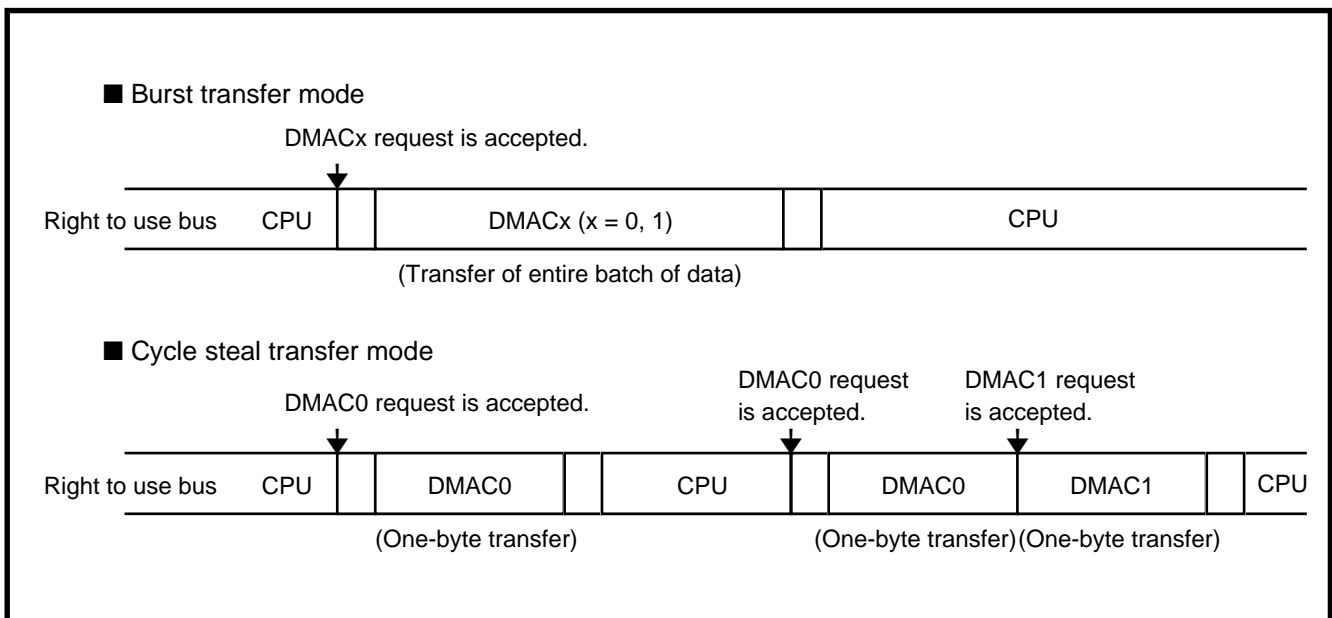


Fig. 2.5.11 Transfer mode overview

#### (1) Priority

The DMAC places a higher priority on Channel-0 transfer requests than on Channel-1 transfer requests.

If a channel-0 transfer request occurs during a channel-1 burst transfer operation, the DMAC completes the next transfer source and destination read/write operation first, and then starts the channel-0 transfer operation. As soon as the channel-0 transfer is completed, the DMAC resumes the channel-1 transfer operation.

#### (2) Transfer request acceptance

A transfer request is confirmed at every rising of  $\phi$ . After that a channel priority and a right to use the bus is judged.

A software trigger and/or a hardware factor can be selected as a transfer request source. The DMAC channel x hardware transfer request source bits (DxHR) selects a hardware factor.

Writing "1" to the DMAC channel x software transfer trigger bit (DxSWT) can generate a transfer request as a software trigger.

### (3) DMA execution

The selected channel transfers one byte of data from the address indicated by its source register (address 42<sub>16</sub> or 43<sub>16</sub>) into the address indicated by its destination register (address 44<sub>16</sub> or 45<sub>16</sub>) with at 2 cycles of  $\phi$ .

The operation of the source registers and destination registers after transfer completion can be selected between decreased/increased by 1 and no change with bits 0 to 3 in the DMAC channel x mode register 1.

When the transfer count register underflows, the source registers and destination registers are reloaded from their latches when the DMAC register reload disable bit (DRLDD) is "0". If the DRLDD bit is set to "1", a reload is disabled.

A read/write must be performed to the source registers, destination registers and transfer count registers as follows:

Read from each higher byte first, then the lower byte

Write to each lower byte first, then the higher byte.

Figure 2.5.12 shows the basic operation of registers transferring.

Tables 2.5.1 and 2.5.2 shows the address directions and examples of transfer result.

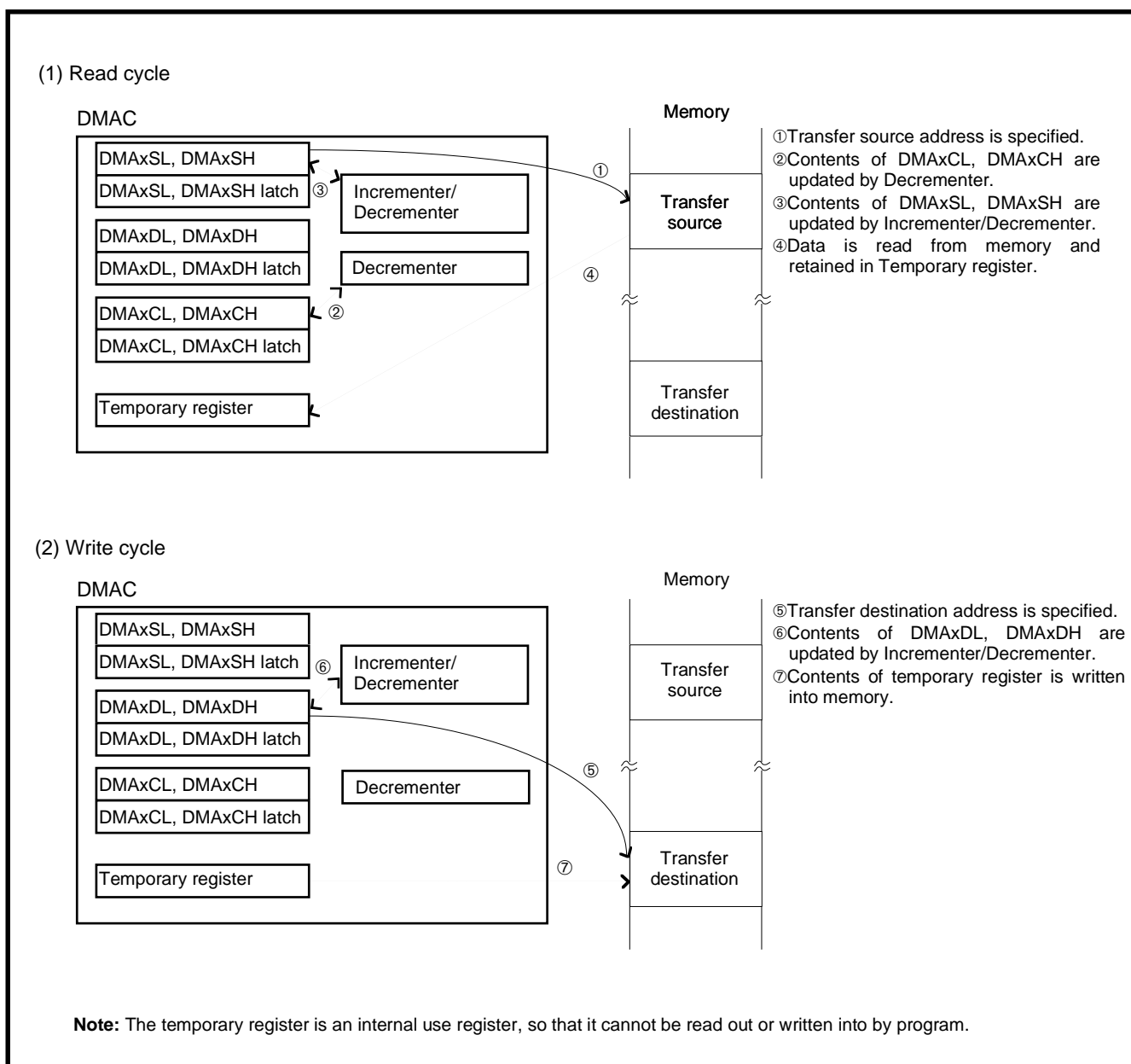


Fig. 2.5.12 Basic operation of registers transferring

**Table 2.5.1 Address directions and examples of transfer result (1)**

Address direction		Data arrangement on transfer source memory	Transfer sequence	Data arrangement on transfer destination memory (Transfer result)
Source	Destination			
Fixed	Fixed	* Data *		* Data *
Fixed	Forward	* Data 1 to 6 *		* Data 1 Data 2 Data 3 Data 4 Data 5 Data 6 *
Fixed	Backward	* Data 1 to 6 *		* Data 6 Data 5 Data 4 Data 3 Data 2 Data 1 *
Forward	Fixed	* Data 1 Data 2 Data 3 Data 4 Data 5 Data 6 *		* Data 1 to 6 *
Forward	Forward	* Data 1 Data 2 Data 3 Data 4 Data 5 Data 6 *		* Data 1 Data 2 Data 3 Data 4 Data 5 Data 6 *

**Table 2.5.2 Address directions and examples of transfer result (2)**

Address direction		Data arrangement on transfer source memory	Transfer sequence	Data arrangement on transfer destination memory (Transfer result)												
Source	Destination															
Forward	Backward	* <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>Data 1</td></tr> <tr><td>Data 2</td></tr> <tr><td>Data 3</td></tr> <tr><td>Data 4</td></tr> <tr><td>Data 5</td></tr> <tr><td>Data 6</td></tr> </table>	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6		<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>Data 6</td></tr> <tr><td>Data 5</td></tr> <tr><td>Data 4</td></tr> <tr><td>Data 3</td></tr> <tr><td>Data 2</td></tr> <tr><td>Data 1</td></tr> </table> *	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1
Data 1																
Data 2																
Data 3																
Data 4																
Data 5																
Data 6																
Data 6																
Data 5																
Data 4																
Data 3																
Data 2																
Data 1																
Backward	Fixed	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>Data 6</td></tr> <tr><td>Data 5</td></tr> <tr><td>Data 4</td></tr> <tr><td>Data 3</td></tr> <tr><td>Data 2</td></tr> <tr><td>Data 1</td></tr> </table> *	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1		<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>Data 1 to 6</td></tr> </table> *	Data 1 to 6					
Data 6																
Data 5																
Data 4																
Data 3																
Data 2																
Data 1																
Data 1 to 6																
Backward	Forward	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>Data 6</td></tr> <tr><td>Data 5</td></tr> <tr><td>Data 4</td></tr> <tr><td>Data 3</td></tr> <tr><td>Data 2</td></tr> <tr><td>Data 1</td></tr> </table> *	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1		<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>Data 1</td></tr> <tr><td>Data 2</td></tr> <tr><td>Data 3</td></tr> <tr><td>Data 4</td></tr> <tr><td>Data 5</td></tr> <tr><td>Data 6</td></tr> </table> *	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
Data 6																
Data 5																
Data 4																
Data 3																
Data 2																
Data 1																
Data 1																
Data 2																
Data 3																
Data 4																
Data 5																
Data 6																
Backward	Backward	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>Data 6</td></tr> <tr><td>Data 5</td></tr> <tr><td>Data 4</td></tr> <tr><td>Data 3</td></tr> <tr><td>Data 2</td></tr> <tr><td>Data 1</td></tr> </table> *	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1		<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>Data 6</td></tr> <tr><td>Data 5</td></tr> <tr><td>Data 4</td></tr> <tr><td>Data 3</td></tr> <tr><td>Data 2</td></tr> <tr><td>Data 1</td></tr> </table> *	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1
Data 6																
Data 5																
Data 4																
Data 3																
Data 2																
Data 1																
Data 6																
Data 5																
Data 4																
Data 3																
Data 2																
Data 1																

**(4) Transfer suspension**

Writing "0" to the DMAC channel x enable bit (DxCEN) can compulsorily suspend the transfer being executed. The suspended transfer can be resumed by writing "1" to the DxCEN bit.

When an interrupt request, which is enabled, occurs during any DMA operation, the transfer operation is suspended and the interrupt process routine is initiated. During the interrupt operation, the DMAC automatically sets the corresponding DMAC channel x suspend flag to "1". When the DMAC transfer suspend control bit (DTSC) is "1", the transfer is suspended in both burst transfer and cycle steal transfer modes during an interrupt process; when the DTSC bit is "0", it is suspended in only burst transfer mode.

The suspended transfer due to the interrupt can also be resumed during its interrupt process routine by writing "1" to the DxCEN bit.



#### 2.5.4 DMAC arbitration

The DMA transfer request is accepted at a rising of  $\phi$ . If the bus is not released 1 cycle of  $\phi$  later than the transfer request acceptance, the DMAC will wait for the bus released. When the bus is released, the DMAC has a right to use the bus and starts the DMA transfer unless a request of the right to use the bus having priority over the DMAC occurs.

Table 2.5.3 shows the priority to use the bus.

**Table 2.5.3 Priority to use bus**

Priority	Factor requesting right to use bus
1 (Higher)	Hold request via $\overline{\text{HOLD}}$ pin
2	DMAC
3	CPU data access
4 (Lower)	CPU instruction access

#### 2.5.5 Transfer time

One-byte transfer of the cycle steal transfer mode requires the time calculated by the following equation:

$$\text{Time (T)} = A + B + C$$

A: This means the time from the occurrence of DMA transfer source request to sampling it. It needs 1 cycle of  $\phi$  at the maximum. The sampling is asynchronously performed.

B: This means the delay time to sample the DMA transfer source request. It needs 1 cycle of  $\phi$  at the maximum.

C: This means the time to transfer data. It needs 2 cycles of  $\phi$ .

Figures 2.5.13 to 2.5.15 show the timing chart for DMA transfer.

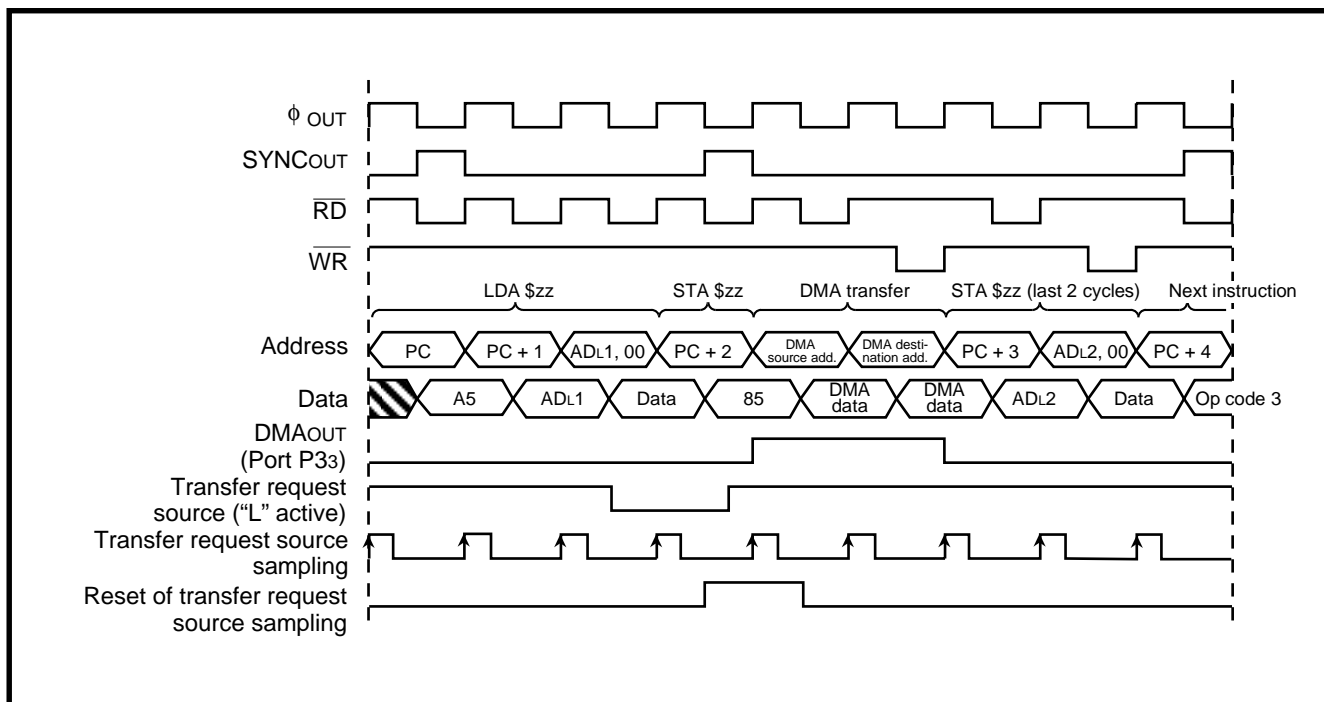


Fig. 2.5.13 Timing chart for cycle steal transfer caused by hardware-related transfer request

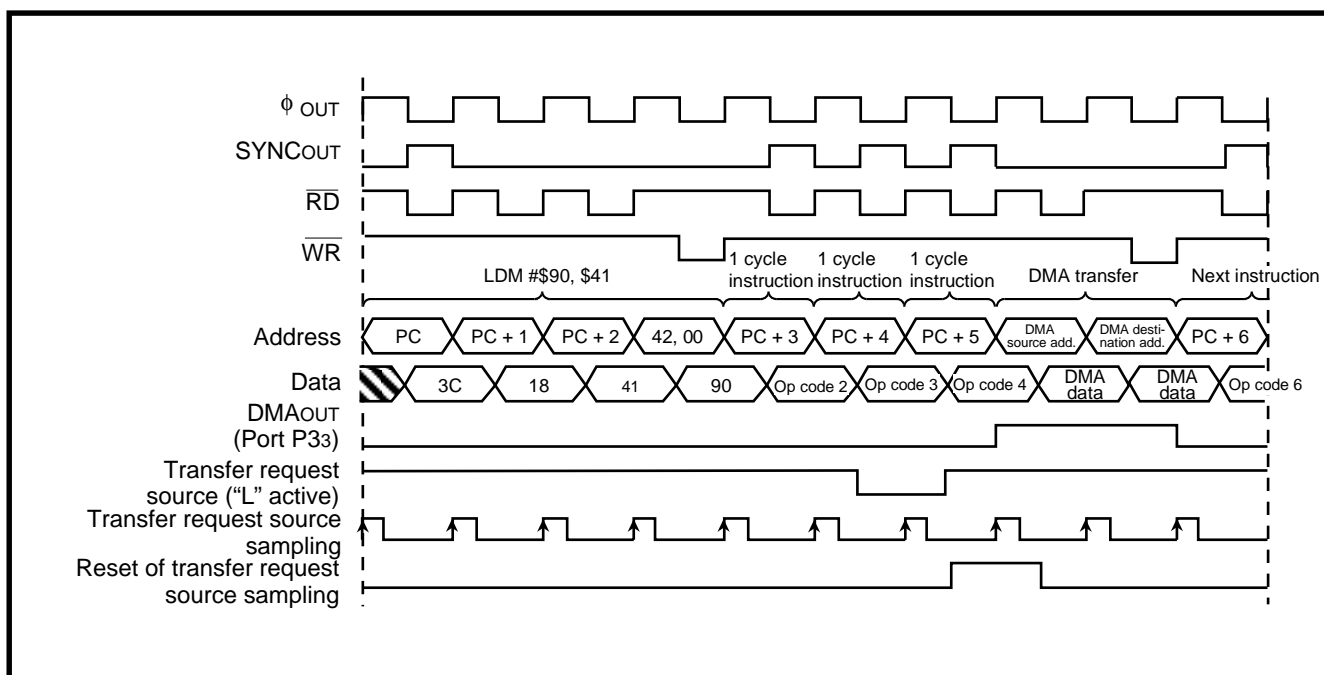


Fig. 2.5.14 Timing chart for cycle steal transfer caused by software trigger transfer request

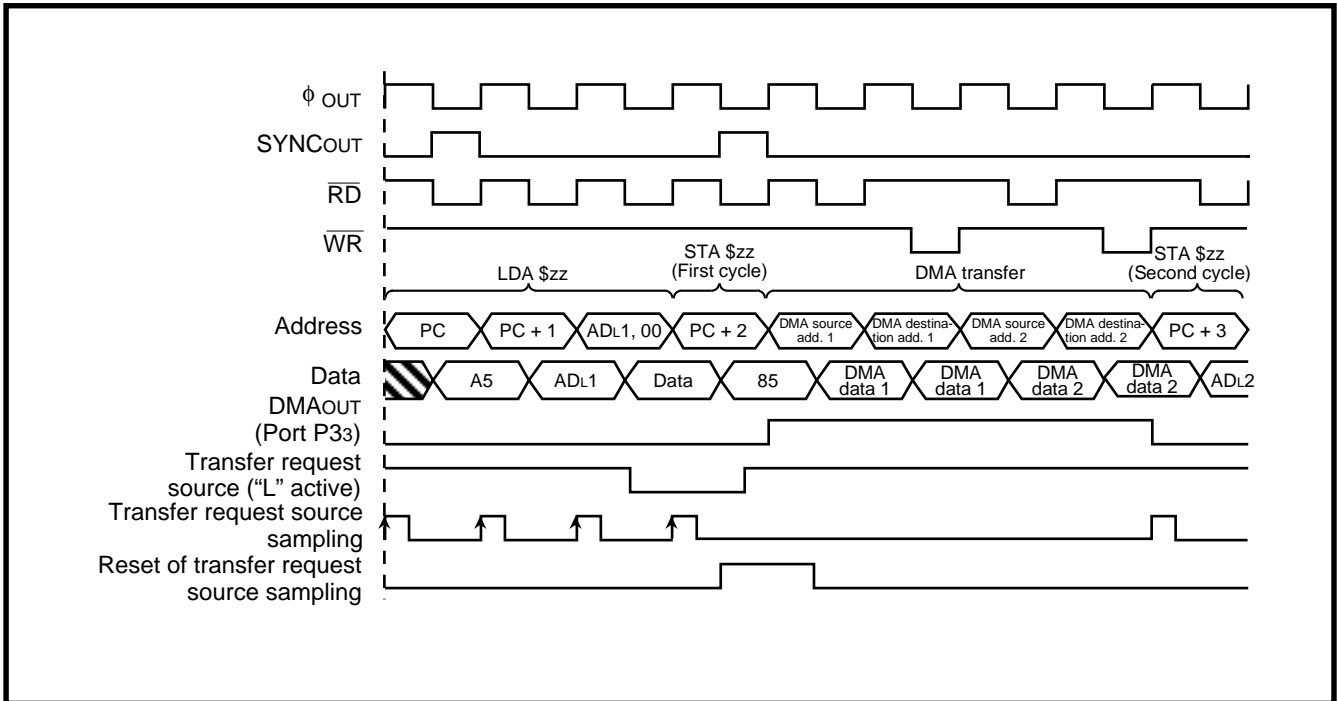


Fig. 2.5.15 Timing chart for burst transfer caused by hardware-related transfer request

### 2.5.6 DMAC application example

#### (1) Transfer from external FIFO to USB FIFO

**Outline:** Data are transferred from external FIFOs to USB FIFOs.

**Specifications:**

- A burst transfer is used and 128-byte (128 bytes/packet) continuous transfer is performed. The external FIFO size must be set as 128 bytes of an integral multiple.
- If the data is deficient in the external FIFO, the DMAC channel 0 transferring is stopped in the DMAC channel 0 interrupt routine process. The DMA transfer is resumed in the main routine process when the data is sufficient.
- To confirm the external FIFO state after completion of 128 bytes transferring, set the DMAC channel x disable after count register underflow enable bit (DxDAUE) to "1".
- USB endpoint 2 IN\_PKT\_RDY (falling edge) selected as hardware transfer request source.

Figures 2.5.16 and 2.5.17 show a setting of the related registers and Figure 2.5.18 shows a control procedure.

If data are transferred from USB FIFOs to external FIFOs as well as this, use USB endpoint 2 OUT\_PKT\_RDY (rising edge) selected as hardware transfer request source.

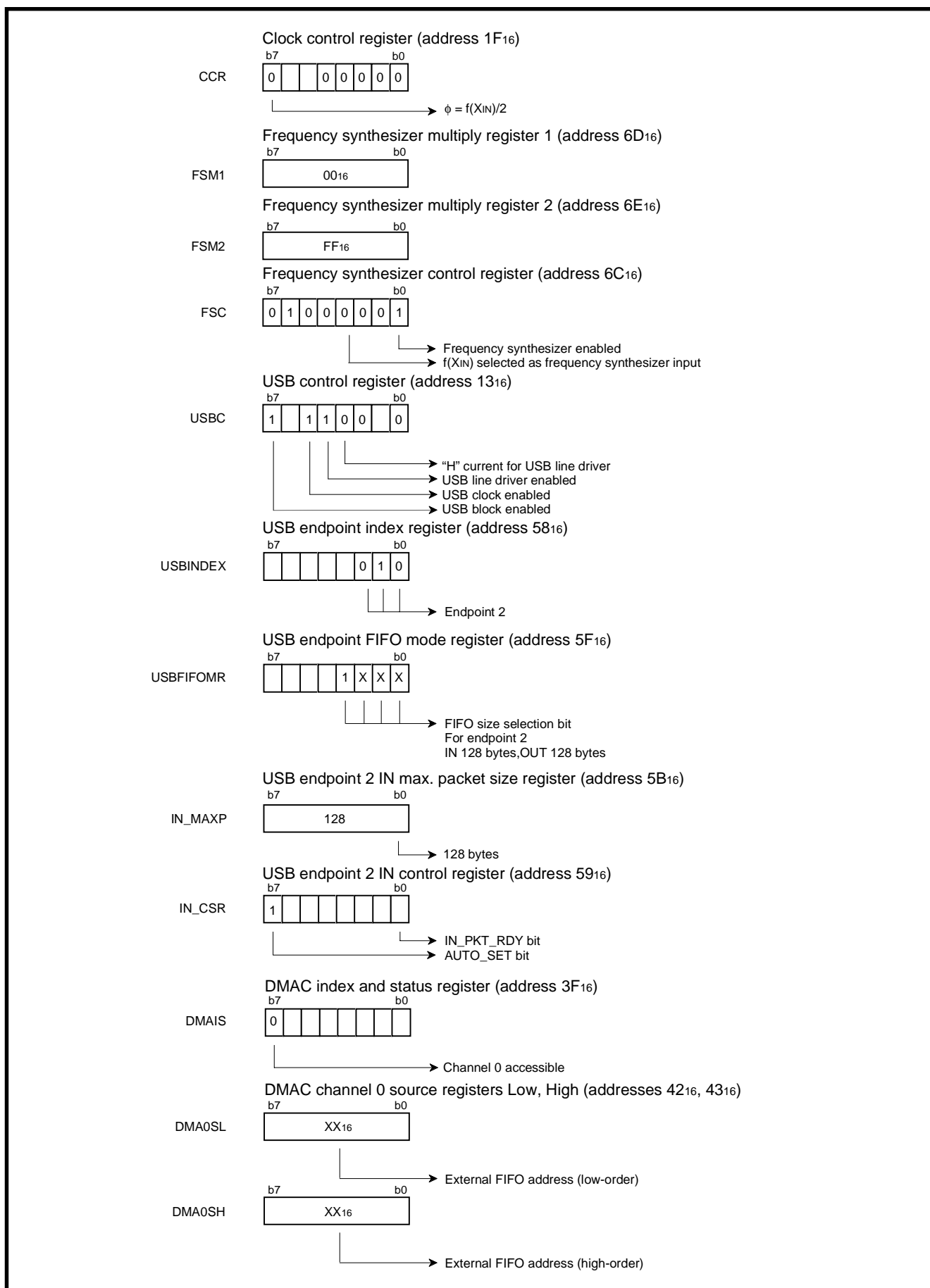


Fig. 2.5.16 Setting of relevant registers (1)

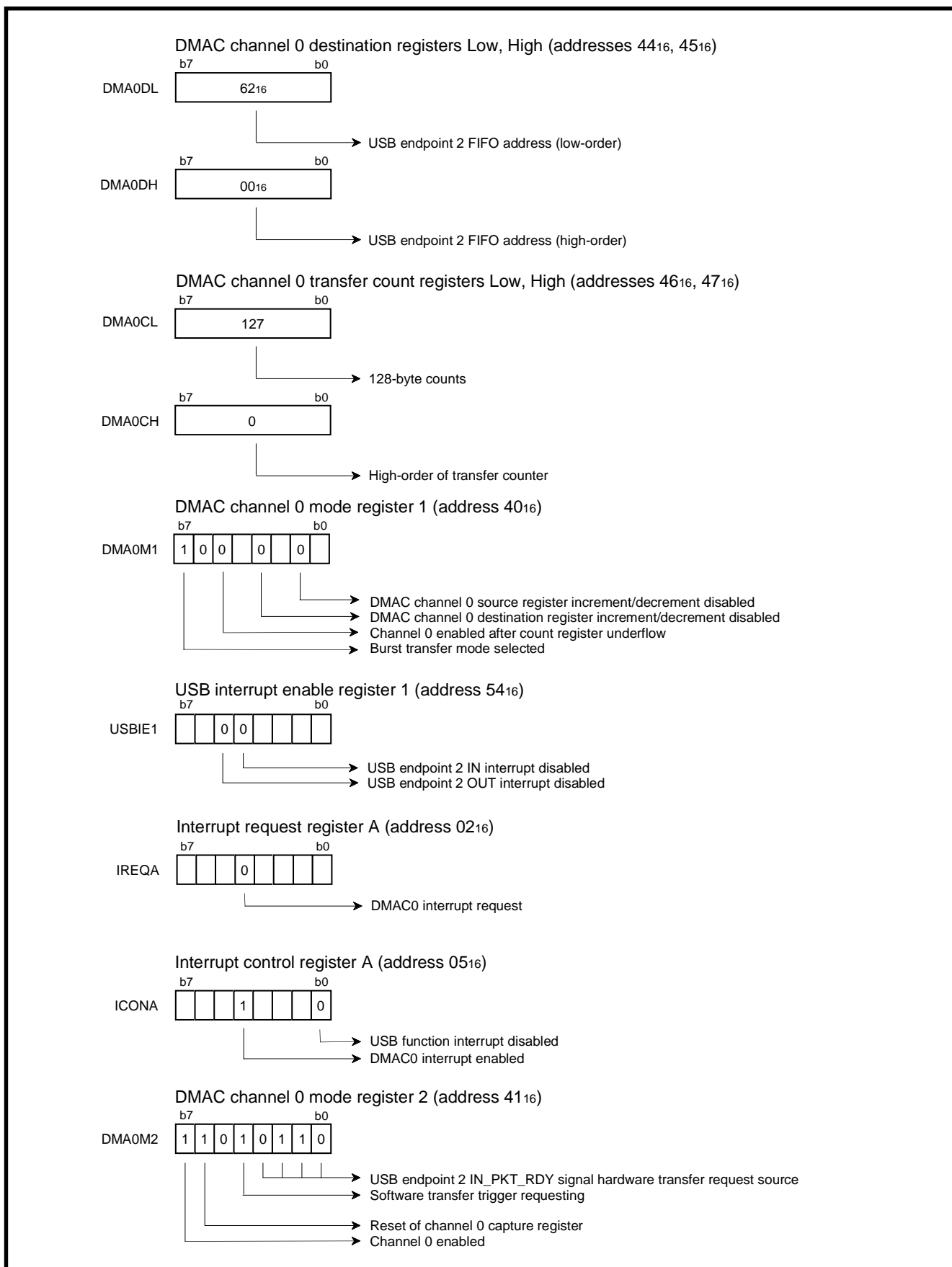


Fig. 2.5.17 Setting of relevant registers (2)

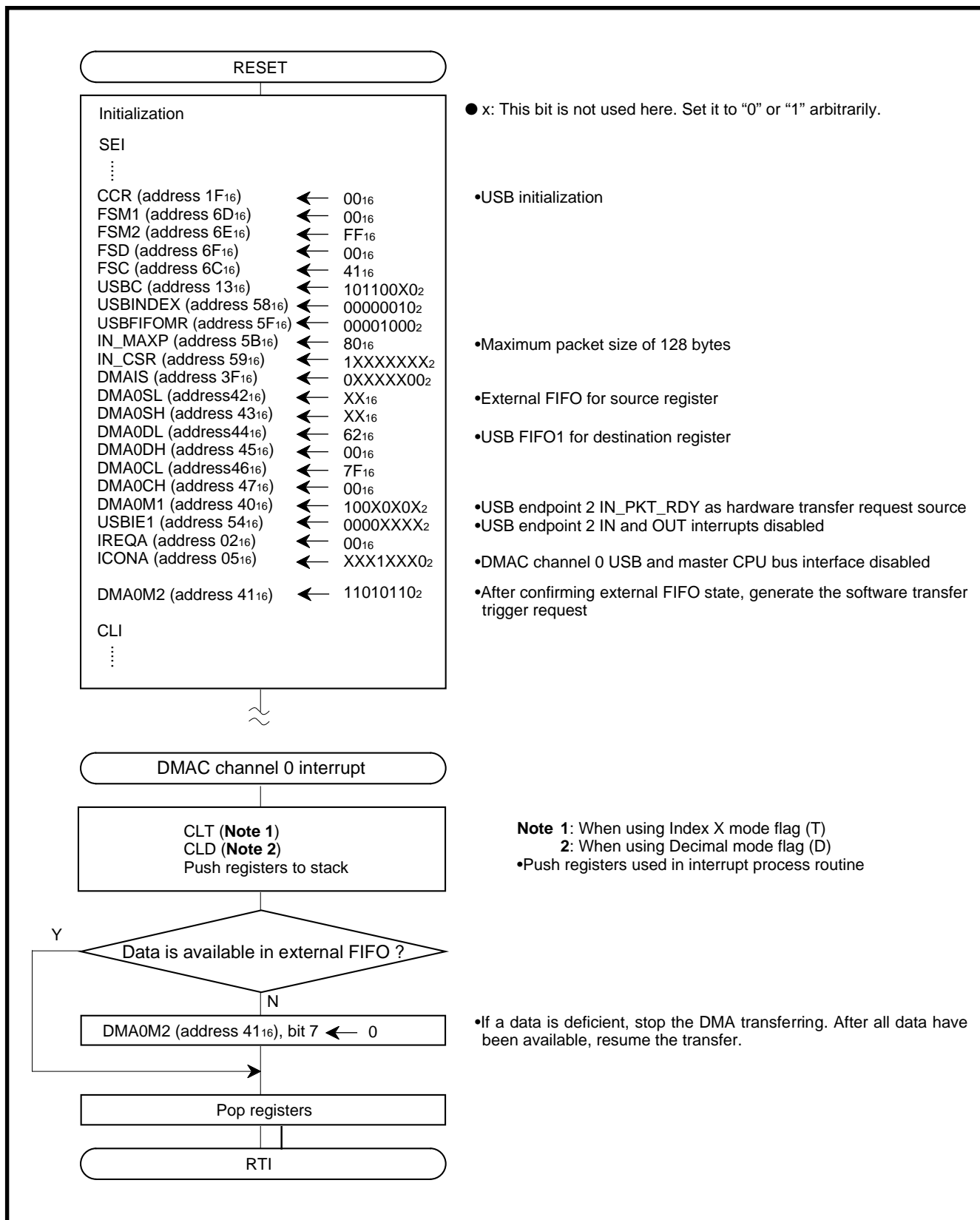


Fig. 2.5.18 Control procedure

### 2.5.7 Notes on DMAC

#### (1) Transfer time

- One-byte data transfer requires 2 cycles of  $\phi$  (read and write cycles).
- To perform DMAC transfer due to the different transfer requests on the same DMAC channel or DMAC transfer between both DMAC channels, 1 cycle of  $\phi$  or more is needed before transfer is started.

#### (2) Priority

- The DMAC places a higher priority on channel-0 transfer requests than on channel-1 transfer requests.

If a channel-0 transfer request occurs during a channel-1 burst transfer operation, the DMAC completes the next transfer source and destination read/write operation first, and then stops the channel-1 transfer operation.

The channel-1 transfer operation which has been suspended is automatically resumed from the point where it was suspended so that channel-1 transfer can complete its one-burst transfer unit. This will be performed even if another channel-0 transfer request occurs.

- The suspended transfer due to the interrupt can also be resumed during its interrupt process routine by writing "1" to the DMAC channel x enable bit (DxCEN).

#### (3) Related registers

- A read/write must be performed to the source registers, transfer destination registers and transfer count registers as follows:

Read from each higher byte first, then the lower byte

Write to each lower byte first, then the higher byte.

Note that if the lower byte is read out first, the values are the higher byte's.

- Do not access the DMAC-related registers by using a DMAC transfer. The destination address data and the source address data will collide in the DMAC internal bus.
- When setting the DMAC channel x enable bit (bit 7 of address 41<sub>16</sub>) to "1", be sure simultaneously to set the DMAC channel x transfer initiation source capture register reset bit (bit 6 of address 41<sub>16</sub>) to "1". If this is not performed, an incorrect data will be transferred at the same time when the DMAC is enabled.

#### (4) DMA<sub>OUT</sub> pin

In the memory expansion mode and microprocessor mode, the DMA<sub>OUT</sub> pin (P3<sub>3</sub>/DMA<sub>OUT</sub>) outputs "H" during a DMA transfer.



## 2.6 USB

The 7643 Group USB Function Control Unit (USB FCU) complies with the Full-Speed USB2.0 Specification, which defines the following four USB transfer types.

- Control transfer
- Interrupt transfer
- Bulk transfer
- Isochronous transfer

The 7643 Group default transfer mode is the bulk transfer. Interrupt (rate feedback) transfer is performed by endpoints 1 and 2, which need to be initialized accordingly. Control transfer is only performed by endpoint 0; the transfer format is the same as that of the bulk transfer.

To use a USB interrupt, set the USB enable bit (USBC7) to "1".

This paragraph explains the registers setting method and the notes related to USB.

### 2.6.1 USB outline

#### (1) Transfer types

The USB specification is generally divided into two sections: the host side (PC, Hub) which controls the peripherals, and the peripheral side (devices), indicating the peripherals that are connected to the host. In addition, the peripheral side has two communication (transfer) specifications which are applied according to the data to be transferred: the Full-Speed function supports high transfer speeds (12Mbps) for peripherals requiring large amounts of data transferred at one time (images, voice, etc.) and the Low-Speed function supports lower transfer speeds (1.5Mbps) for peripherals sending smaller amounts of data (keyboard, mouse, etc). The recent addition of the Hi-Speed function (480Mbps) specification allows for even faster data transfer.

The communication specification to be used is determined by the Device Class of the peripheral, and the transfer type is determined according to each peripheral device.

The 7643 Group is equipped with a Full-Speed Function Control Unit, which supports the following three transfer types.

#### **Control transfer**

Control transfer is a request-response, bi-directional transfer in the non-periodic and bursty communications. These are typically used during setup. As all devices must support standard device requests, all USB products support control transfers with no exception.

#### **Bulk transfer**

Bulk transfer is a non-periodic and bursty communication which is not adversely affected by delays. These are typically used for transfers of large amounts of data. To ensure that the data is transferred successfully, hardware is set to check for errors. If an error is detected, the data is resent. Examples of bulk transfer are word data to printers and image data from scanners.

#### **Interrupt transfer**

Interrupt transfer is used to inform the host of periodic, low-frequency data communication from the device. For example, interrupt transfer informs the host that the printer is out of paper or send data from the mouse or keyboard to the host.

## (2) Communication protocol

The host CPU does control all USB communications. Even when data is transmitted to the host from the device, the transmission is executed after the host gives the device the right to use the bus.

In order for the host to process multiple transfers simultaneously, each transfer is scheduled in packet units within 1ms frame. The diagram below shows an image of one frame.

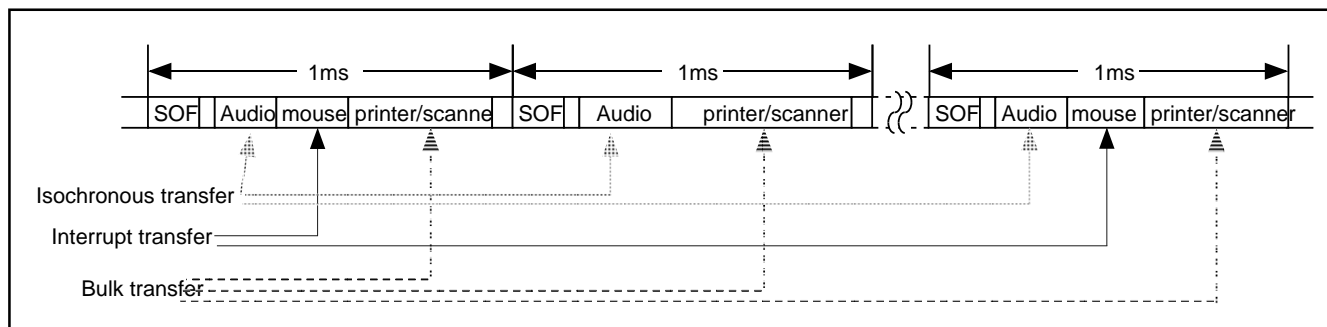
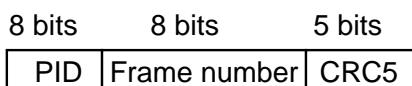


Fig. 2.6.1 1 frame image

### ● Packet

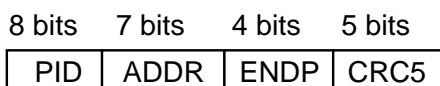
A packet is the unit used by the host CPU or the device to secure use of the bus. In USB communications, data is transmitted and received in packet units. A packet is a group of data string (field) of some bits. It starts from a SYNC (synchronous data) field and continues to a PID (Packet ID) field. Each type of packet can be identified by this PID field. The following is a list of packet types.

SOF packet : indicates the packet of frame start that Host sends every 1 ms.



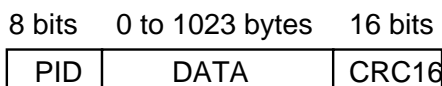
PID : SOF(0xA5)

Token packet : indicates the packet Host sends at transaction start.



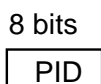
PID : OUT(0xE1),IN(0x69),SETUP(0x2D)

Data packet : indicates the packet to transfer data.



PID : DATA0(0xC3),DATA1(0x4B)

Handshake packet : indicates the packet in transaction to control the flow.



PID : ACK(0xD2),NAK(0xA5),STALL(0x1E)

The start of each packet is SOP.

The end of each packet is EOP.

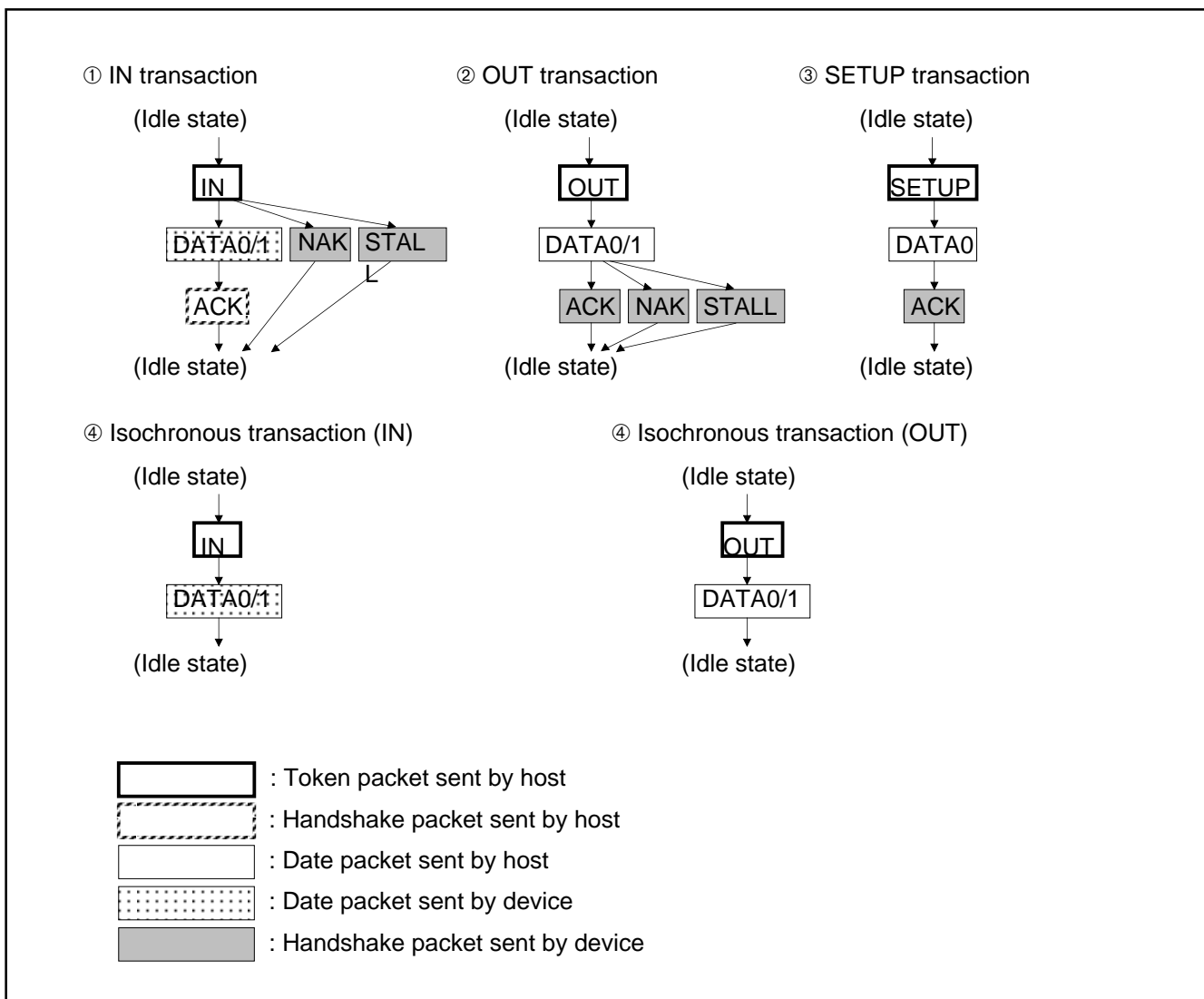
Fig. 2.6.2 Packet type

**Table 2.6.1 USB PID list**

PID type	PID name	bit structure (bit 3 to bit 0)	Outline of processing
Token	SETUP	1101 <sub>2</sub>	Host CPU informs device of operation process.
	IN	1001 <sub>2</sub>	Host CPU requests data transmit from device.
	OUT	0001 <sub>2</sub>	Host CPU requests data receive from device.
	SOF	0101 <sub>2</sub>	Host CPU informs start of frame to device.
Data	DATA0	0011 <sub>2</sub>	Indicates that the number of transfer data sequence bits is even.
	DATA1	1011 <sub>2</sub>	Indicates that the number of transfer data sequence bits is odd.
	ACK	0010 <sub>2</sub>	Reports that data transmit was successfully completed.
Handshake	NAK	1010 <sub>2</sub>	Reports that the device is in the transfer ready status.
	STALL	1110 <sub>2</sub>	Reports that transfer was completed successfully.

### ● Transaction

A transaction is the unit in which the host CPU schedules 1 frame. Each transaction is comprised of packets. The transaction format is shown below.

**Fig. 2.6.3 Transaction format**

### (3) Communication sequence

Each transfer type has a different transfer sequence, as described below.

#### Control transfer communication sequence

The control transfer is the common transfer type used at setup for all devices. Each transfer process consists of three different stages. The control transfer starts with the setup stage. Based on the contents of the Setup stage, the data stage (control Read transfer or control Write transfer) is executed. When the transfer is complete, the status stage is executed, completing one full process.

Endpoint 0 is always used for control transfer.

Figure 2.6.4 shows the communication sequence for a control transfer.

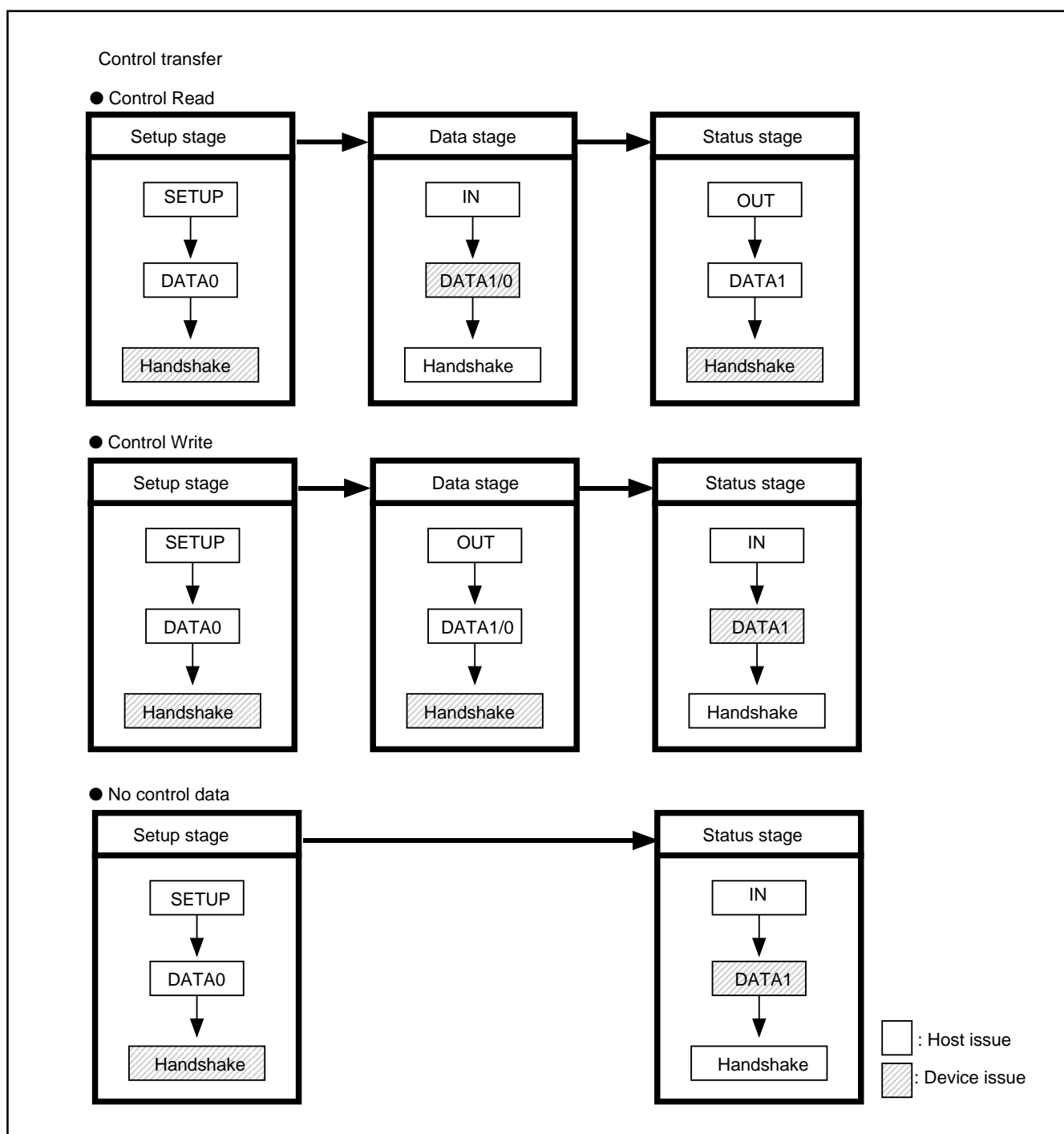


Fig. 2.6.4 Communication sequence of control transfer

**Control Read transfer**

After the SETUP transaction, the IN transaction is repeated to transmit data from the device to the host (data stage). Then, in order to wait for the host to finish processing the data, an IN transaction of data empty is executed (status stage).

**Control Write transfer**

After the SETUP transaction, the OUT transaction is repeated to transmit data from the host to the device (data stage). Then, in order that host waits the device to finish processing the data, an IN transaction of data empty is executed (status stage).

**No data**

After the SETUP transaction, if there is no request for data to be transmitted/received in the data stage, in order that host waits for the device to finish processing, an IN transaction of data empty is executed (status stage).

The status stage is used for reporting the results of the setup and data stage operations to the host CPU. The responses used in the status stage are as follows.

- Function completed normally
  - Control Write transfer...Data packet
  - Control Read transfer...ACK handshake
- Error detected in function
  - Control Write transfer...STALL handshake
  - Control Read transfer...STALL handshake
- Function in busy state
  - Control Write transfer...NAK handshake
  - Control Read transfer...NAK handshake

**● Device request**

The SETUP transaction in the setup stage of a control transfer is called a device request. The device request defines the format of its data phase.

When type is standard (0): Standard device request

This is the basic device request necessary for all USB devices.

When type is class (1): Class device request

Device class is defined by the USB Implementers Forum (USB IF). The USB IF also determines the requested configuration and class request.

Please refer to the Full-Speed USB2.0 specification for more details on each data format.

**● Bulk transfer****Bulk IN transfer**

The bulk IN transfer, which transmits data from the device to host CPU, continually repeats IN transactions. The 7643 MCU issues a data packet in response to an IN token. When the data is not transmitted successfully, the MCU sends one of the following responses.

- No response if the received IN token is broken.
- STALL handshake is returned if 7643 Group is stalled.
- NAK packet is returned if there is no transmit data in the IN FIFO.

**Bulk OUT Transfer**

The bulk OUT transfer, which transmits data from the host CPU to the device, continually repeats OUT transactions.

When the 7643 MCU receives a data packet successfully, an ACK packet is returned. A successful receive is the state in which there is no bit-stuffing error or CRC error, and the data PID was received correctly.

DATA0 and DATA1 of data packet of next data phase are toggled during the handshake phase of each transaction if the ACK packet sent by the receive side is received successfully by the transmit side.

The 7643 MCU transmits the following responses when data is not successfully received.

- No response if the received data is broken.
- STALL handshake is returned if 7643 Group is stalled.
- ACK packet is returned if inconsistency of sequence bit in received data.
- NAK packet is returned if the 7643 Group OUT FIFO is full.

For more details, refer to the Full-Speed USB2.0 specification.

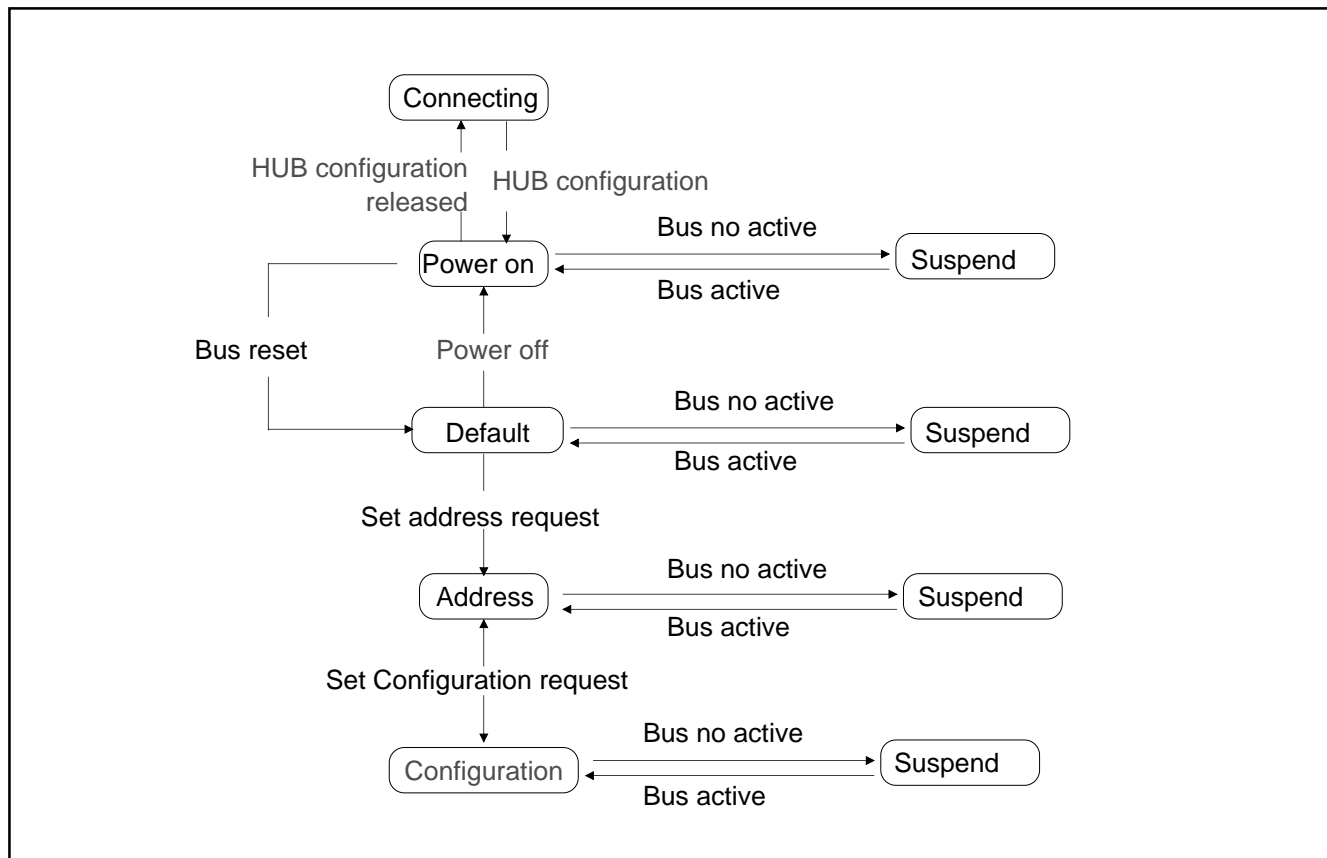
**● Interrupt transfer**

The interrupt transfer format is the same as that of the bulk transfer.

**(4) Device State**

The device has various states and can transit states. In the 7643 Group, control the state transition by software, not hardware.

Enumeration is the continuous process from bus connection to system configuration.



**Fig. 2.6.5 Device state transition**

1. Connection state: When the device is connected to the bus.
2. Power-on state: When the HUB has been configured and power is supplied to the bus.
3. Default state: When a reset signal is received from the host CPU. Default address (0) is configured.
4. Address assignment state: When the SET\_ADDRESS standard device request is received. This is the unconfigured state (configuration 0).
5. The device is configured when the SET\_CONFIGURATION standard device request is received.
6. Suspend state: When there is no activity on the bus for a 3ms period.

## 2.6.2 Memory map

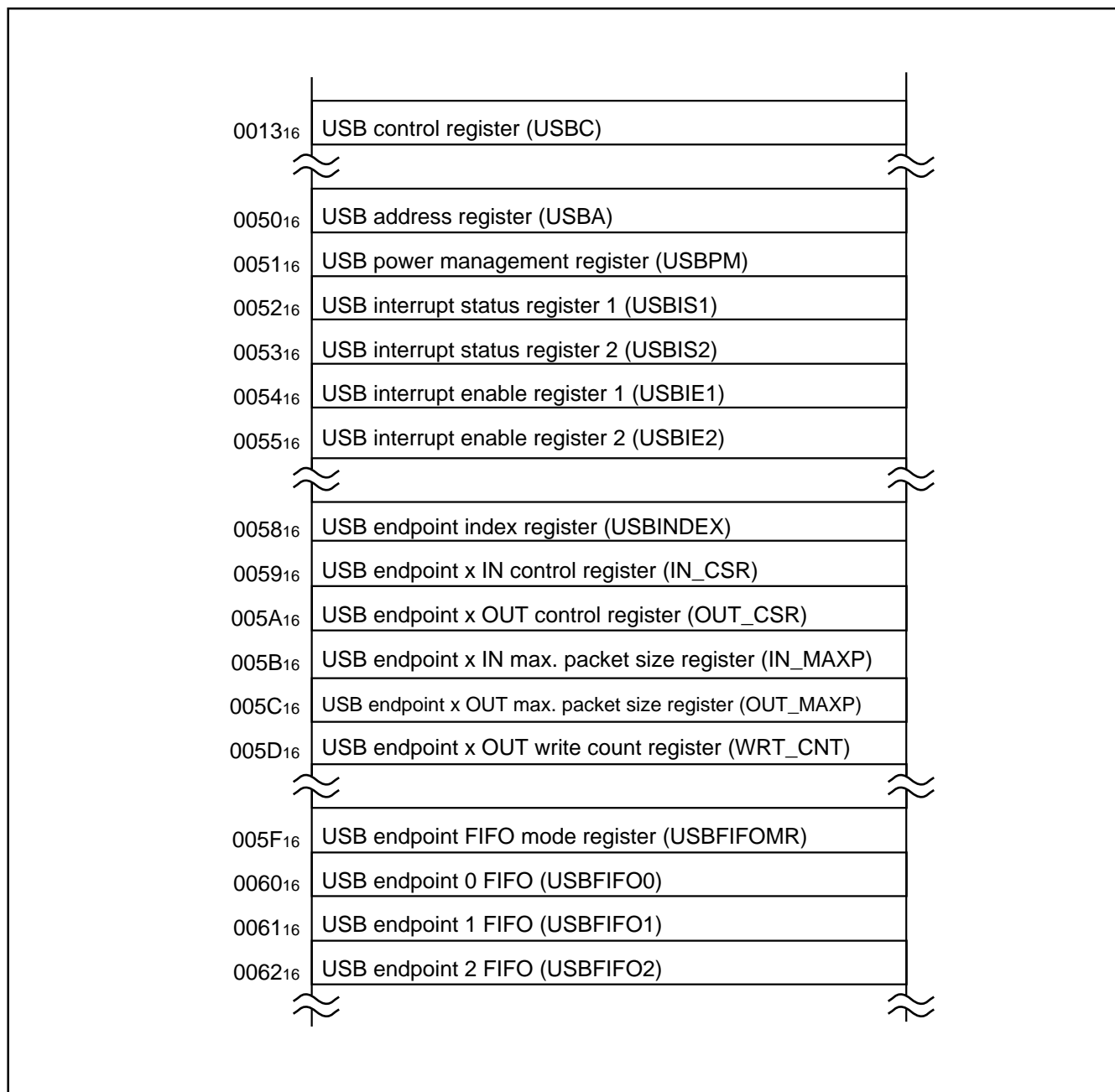


Fig. 2.6.6 Memory map of registers related to USB



### 2.6.3 Related registers

#### (1) USB control register

This register is used for all operation controls performed by the USB function control unit (USB FCU). USB reset does not affect this register.

##### ● USB default state selection bit

This bit selects the cases in which the USB core state machine 'state' (the section that controls transitions of the USB core state) will return to the default state. However, without regard to this bit, USB internal registers (addresses 0050<sub>16</sub> to 005F<sub>16</sub>) go into the states at reset (default state) at the same time when USB reset is detected.

When this bit is "0", the state returns to the default state at power-on/reset.

When this bit is "1", the state returns to the default state when the USB reset is received (recommended).

##### ● USB line driver current control bit

This bit is used to reduce the amount of USB dissipation power.

Set to "1" for the suspend state, but always set to "0" for USB control operations.

When using the MCU in Vcc = 3.3V, the value of this bit does not affect USB operations.

##### ● USB line driver supply enable bit

Set this bit to "1" to use the USB built-in DC-DC converter when using the MCU in Vcc = 5V. The built-in DC-DC converter is disabled when this bit is "0"; therefore, set to "0" when using the MCU in Vcc = 3.3V.

##### ● USB clock enable bit

Set this bit to "1" to enable the USB clock.

##### ● USB enable bit

Set to "1" to enable the USB function. Make sure you include a 250ns wait after setting the bit to "1" to read from/write to other USB-related registers.

Figure 2.6.7 shows the structure of USB control related register.

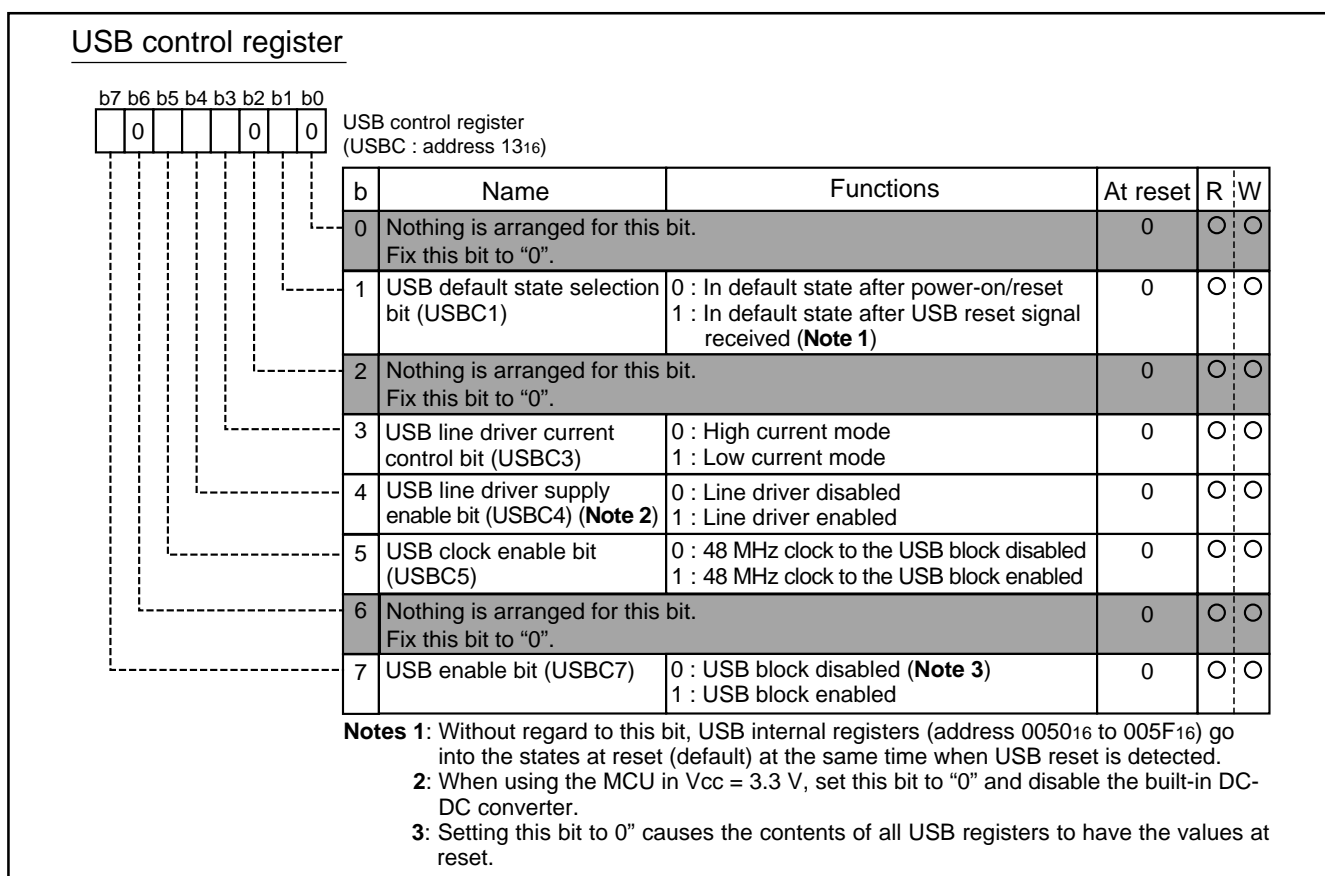


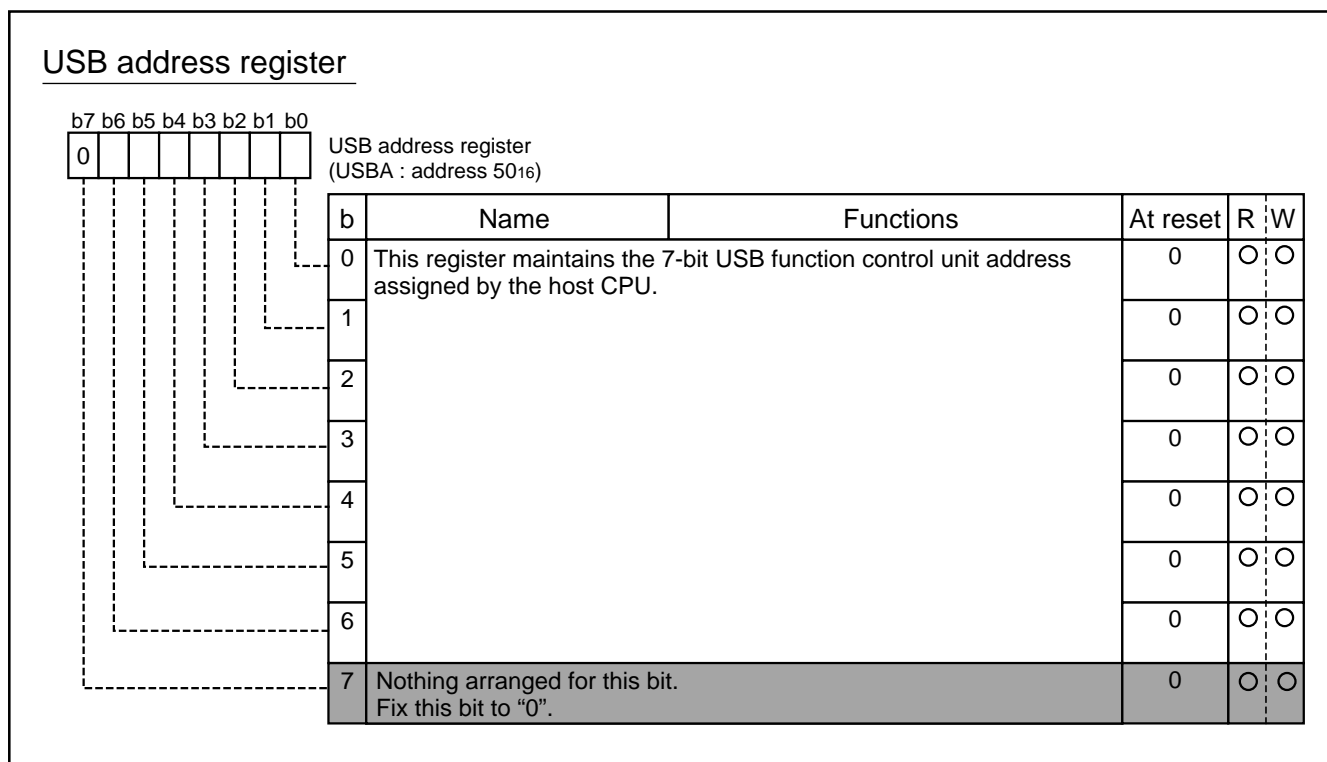
Fig. 2.6.7 Structure of USB control register

**(2) USB address register**

This register maintains the 7-bit USB function control unit (USB FCU) address assigned by the host CPU. The USB FCU stores addresses of USB token packets by using the value of this register.

The value of this register is "00<sub>16</sub>" when the device is not yet configured at reset.

Figure 2.6.8 shows the structure of the USB address register.



**Fig. 2.6.8 Structure of USB address register**

**(3) USB power management register**

This register is used for power management in the USB FCU.

- **USB suspend detection flag**

If no activity is detected on the D+/D- line for a period of 3ms, this flag is set to “1” and a USB suspend interrupt occurs.

When the USB resume detection flag is set to “1” after the MCU returns from the USB suspend state by a USB resume interrupt, the USB suspend detection flag is automatically cleared to “0”. Even if the USB remote wake-up bit is cleared to “0” after the MCU returns from the suspend state by a remote wake-up (USB remote wake-up bit set to “1”), the suspend detection flag is not automatically cleared to “0”. Clear this flag to “0” by software.

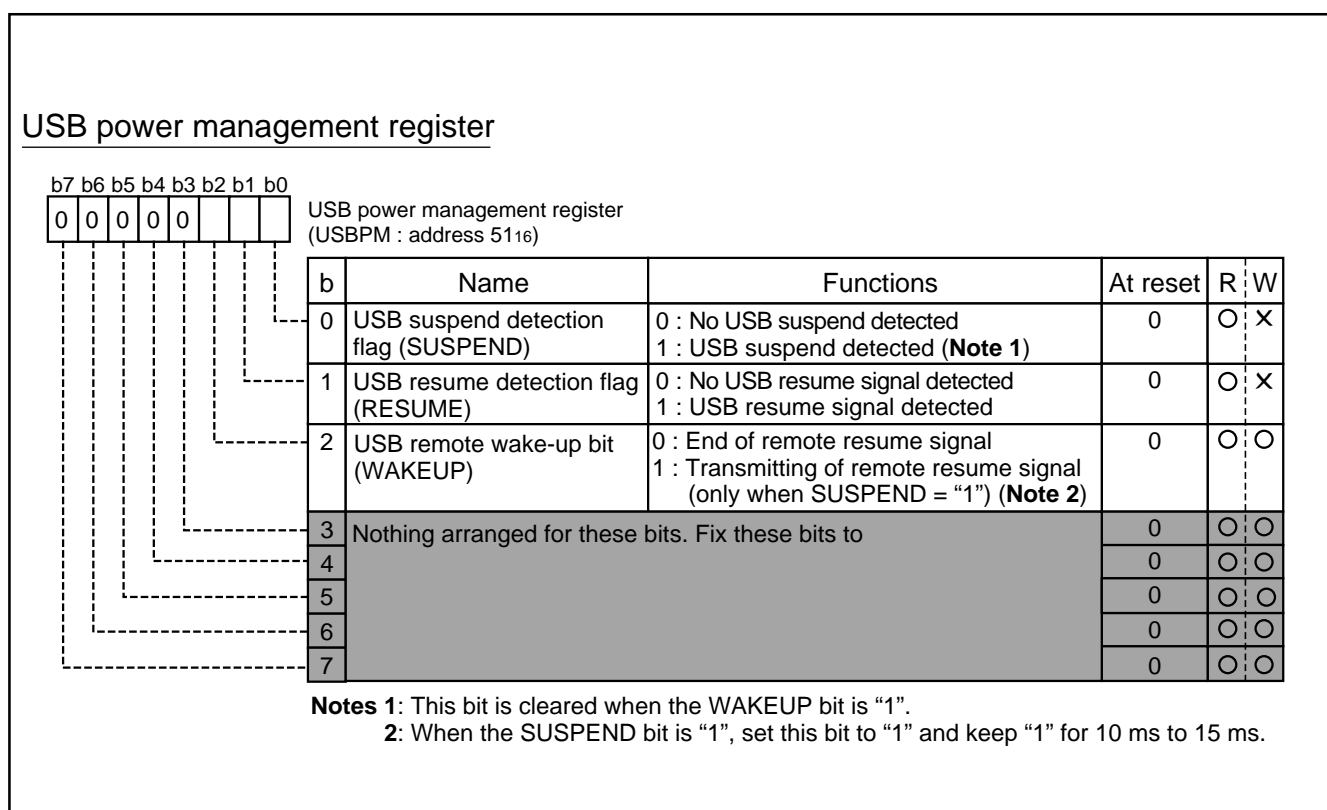
- **USB resume detection flag**

When a non-idle signal is detected on the D+/D- line in the suspend mode, this flag is set to “1” and a USB resume interrupt occurs. This flag is cleared to “0” when the USB resume signal interrupt status flag of USB interrupt status register 2 is cleared to “0”.

- **USB remote wake-up bit**

Set this bit to “1” after the MCU returns from the USB suspend state by a remote wake-up (INT interrupt, etc.). While this bit is “1”, the USB FCU sends a resume signal to the host CPU, informing it of the return from the suspend state. Clear the bit to “0” after holding it at “1” for 10ms to 15ms.

Figure 2.6.9 shows the structure of the USB power management register.



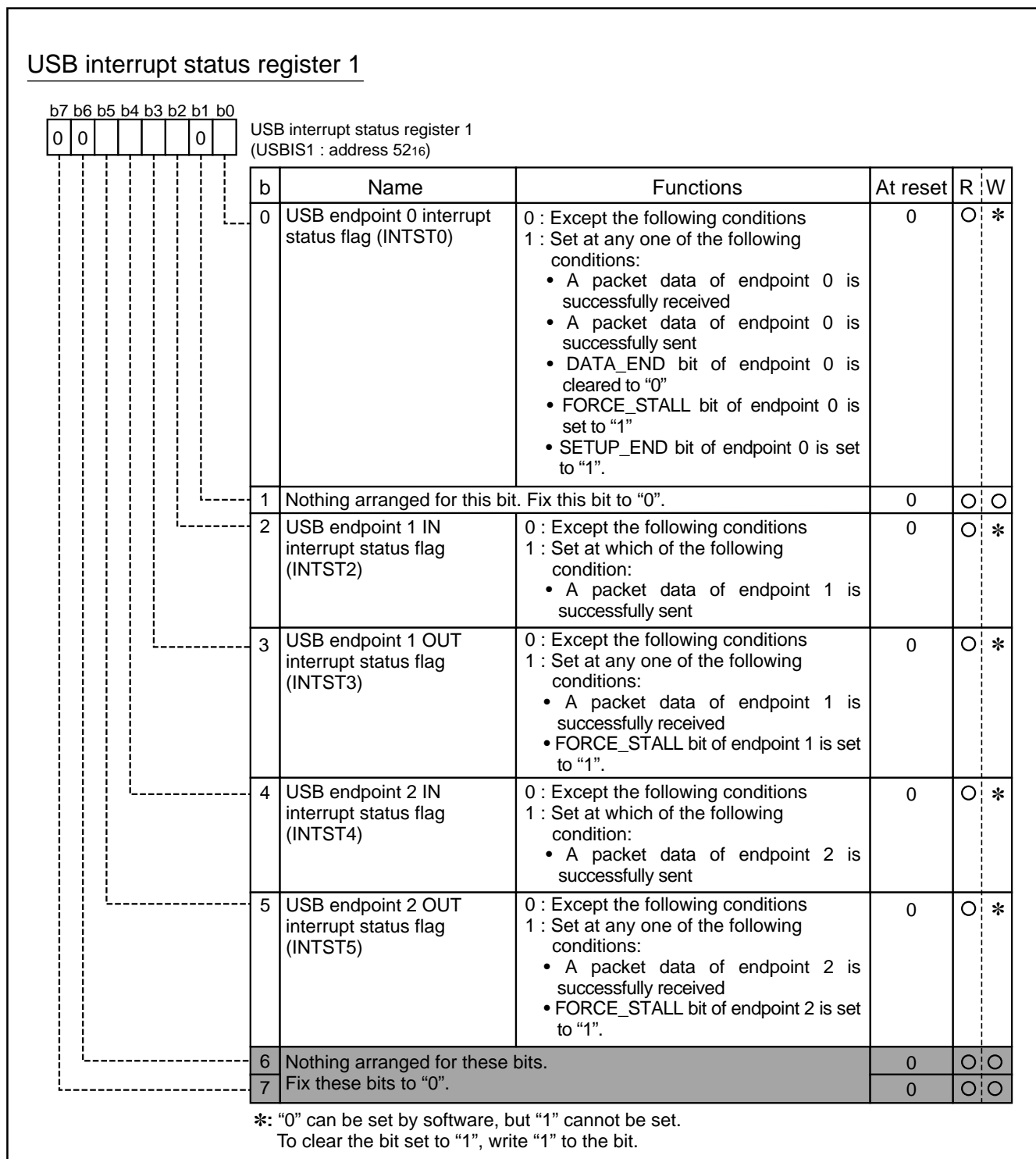
**Fig. 2.6.9 Structure of USB power management register**

**(4) USB interrupt status registers 1, 2**

These registers are used to determine the state of a USB function interrupt source. When an interrupt request occurs, the flag corresponding to the interrupt source is set to "1".

After "1" is read from each flag in this register, they are cleared to "0" when the next "1" is written. This flag is not cleared even if "0" is written. Make sure to write to/read from the USB interrupt status register 1 first and then USB interrupt status register 2.

Figures 2.6.10 and 2.6.11 show the structures of USB interrupt status registers 1 and 2, respectively.



**Fig. 2.6.10 Structure of USB interrupt status register 1**

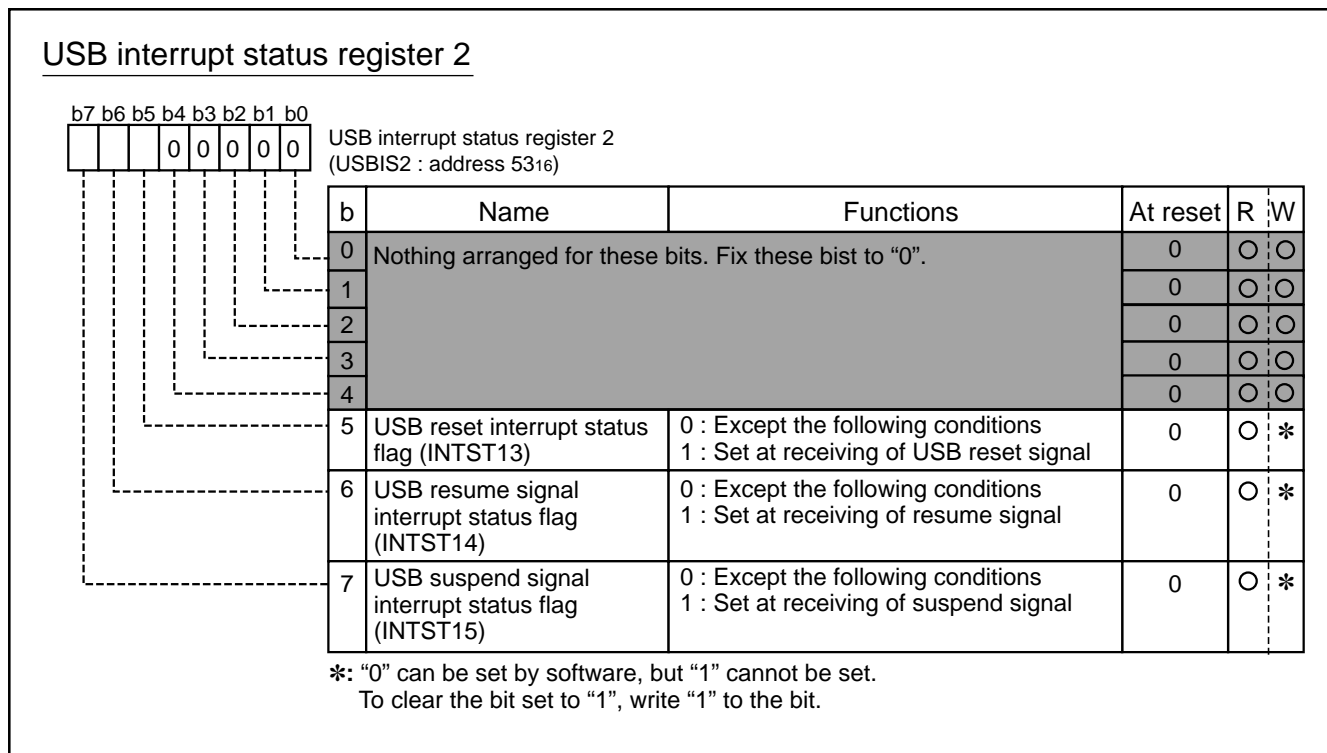


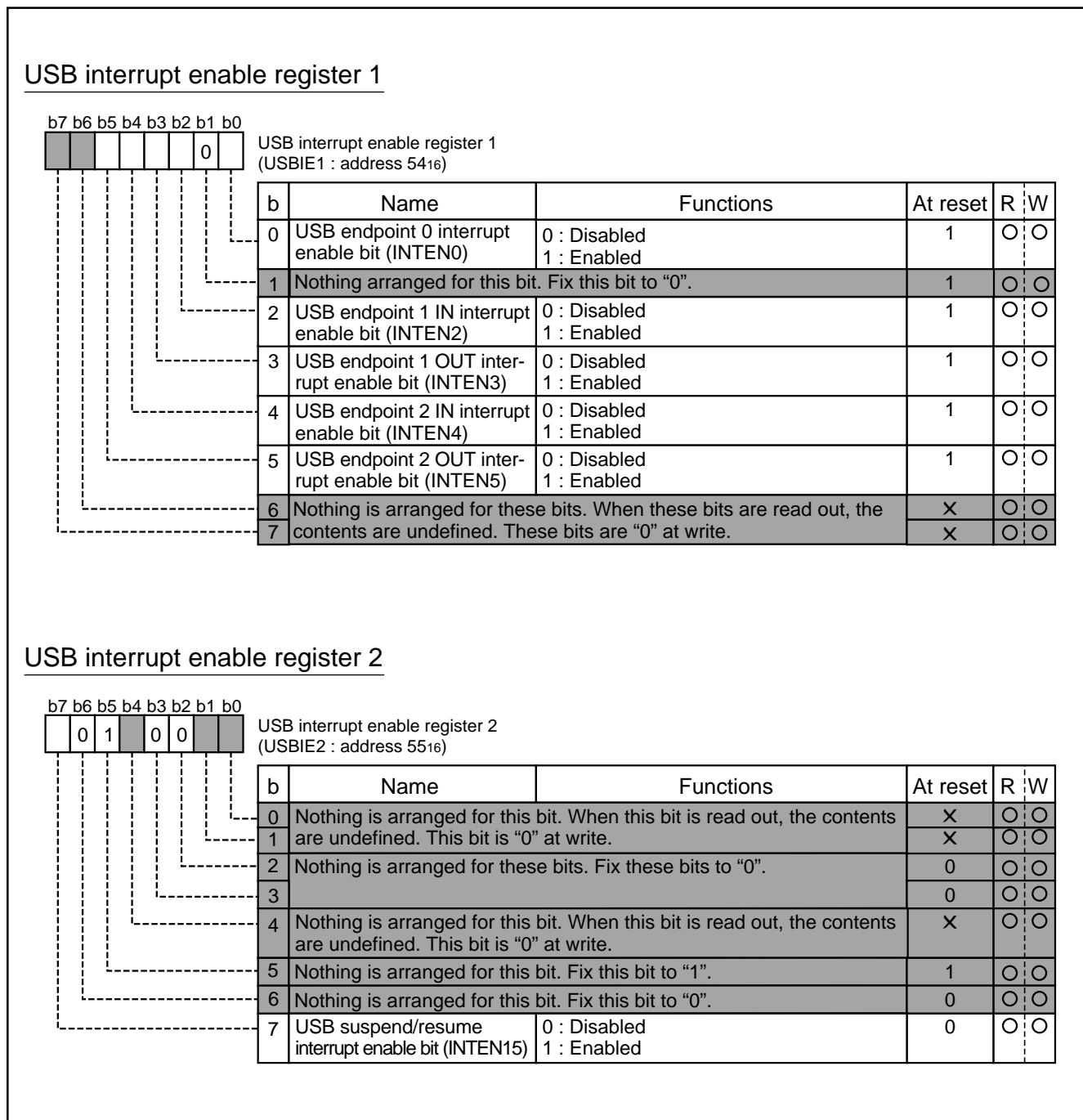
Fig. 2.6.11 Structure of USB interrupt status register 2

**(5) USB interrupt enable registers 1, 2**

The USB interrupt enable registers are used to enable the USB function interrupt. Upon reset, all USB interrupts except the USB suspend and USB resume interrupts are enabled.

The USB reset interrupt does not have an enable bit and is always in the enabled state.

Figures 2.6.12 shows the structures of USB interrupt enable registers 1 and 2.



**Fig. 2.6.12 Structure of USB interrupt enable register 1, USB interrupt enable register 2**

**(6) USB endpoint index register**

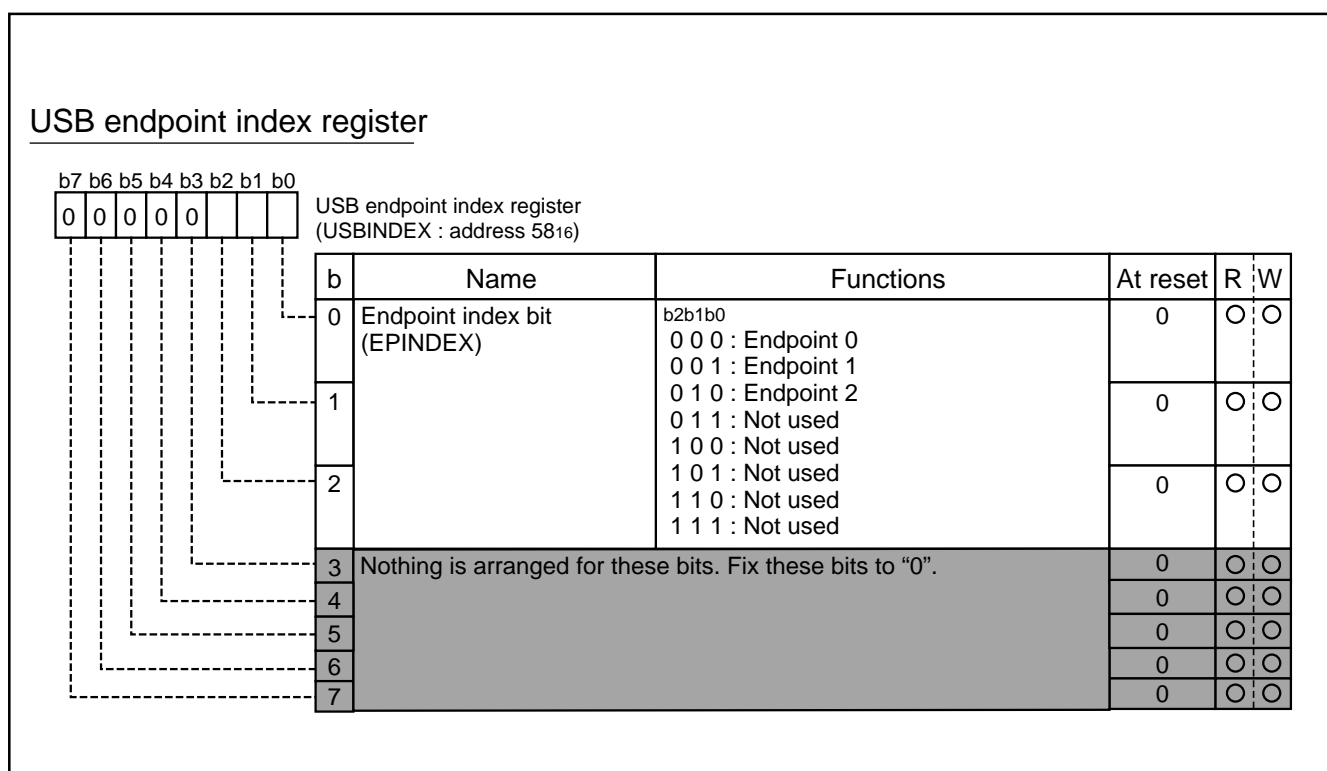
Endpoint index bits

These bits specify the accessible endpoint.

These bits serve as an index of the following registers;

- USB endpoint x (x=0 to 2) IN control register
- USB endpoint x (x=1 to 2) OUT control register
- USB endpoint x (x=0 to 2) IN max. packet size register
- USB endpoint x (x=0 to 2) OUT max. packet size register
- USB endpoint x (x=0 to 2) OUT write count register
- USB endpoint FIFO mode register (only endpoint 1 and endpoint 2)

Figure 2.6.13 shows the structure of the USB endpoint index register.



**Fig. 2.6.13 Structure of USB endpoint index register**

**(7) USB endpoint 0 IN control register**

This register comprises the bits related to endpoint 0 control and status information.

**● OUT\_PKT\_RDY flag**

This flag is set to "1" when a valid SETUP/OUT token is received from the host CPU. After the OUT FIFO is read, clear this flag to "0" by writing "1" to the SERVICED\_OUT\_PKT\_RDY bit. (However, do not clear this flag to "0" until the request from the host CPU has been completely decoded.)

Nothing is changed even when writing "1" to this flag is executed.

**● IN\_PKT\_RDY bit**

Set this bit to "1" after writing one packet of data to the endpoint 0 IN FIFO. The USB FCU clears this bit to "0" after an IN FIFO transmit or when the SETUP\_END flag is set to "1".

Nothing is changed even when writing "0" to this flag is executed.

**● SEND\_STALL bit**

Set this bit to "1" when an invalid standard request is received from the host CPU. When the OUT\_PKT\_RDY flag to receive the request is set to "0", the USB FCU sends a STALL handshake signal to the host CPU in response to all IN/OUT transactions.

Write "0" to this bit to clear it. If the OUT\_PKT\_RDY flag is "1" when the SEND\_STALL bit is set to "1", the OUT\_PKT\_RDY flag can be cleared to "0" by setting the SERVICED\_OUT\_PKT\_RDY bit to "1".

**● DATA\_END bit**

When the last data is written to the FIFO (IN data phase) or the last data is read from the FIFO (OUT data phase), set this bit to "1". This informs the USB FCU that all data set in the setup phase has been completely processed.

The USB FCU sets this bit to "0" automatically after the status phase process is completed. When this bit is "1", data receive and data transmit requests from the host are ignored. The USB FCU sends a STALL handshake signal to the host CPU and completes the current control operation.

**● FORCE\_STALL flag**

This flag is set to "1" for an error report if any of the following conditions occur.

- An IN/OUT token is received which does not have a setup stage.
- An incorrect data toggle is received (DATA0) in the status stage.
- An incorrect data toggle is received (DATA1) in the setup stage.
- A request for more data than requested in the setup stage (an IN token is received after the DATA\_END bit is set).
- A data receive for more data than specified in the setup stage (an OUT token is received after the DATA\_END bit is set).
- More data is received than the number of bytes in the maximum packet size of the corresponding endpoint.

Excluding the case in which an incorrect data toggle occurs in the setup stage, when one of the above conditions occurs, a STALL is sent in response to the problematic IN/OUT token. When there is an incorrect data toggle in the setup stage, an ACK is returned in response to the setup stage, and a STALL is returned in response to the next IN/OUT token. The generated STALL handshake is transmitted in response to only one transaction, and the control transfer in process is completed. The next packet received after the STALL handshake is recognized as the beginning of a new control transfer.

Write "0" to this flag to clear it.



- **SETUP\_END flag**

During a control transfer, if the process is terminated before the number of data bytes, as set in the data stage process, is completed, the SETUP\_END flag will go to “1”.

Clear the SETUP\_END flag to “0” by setting the SERVICED\_SETUP\_END bit to “1”.

When this flag is set to “1”, you need to disable the access to the FIFO and execute the setup process before it.

When this flag is set to “1”, the data in the IN FIFO is automatically flushed out.

If this flag and the OUT\_PKT\_RDY flag are set to “1” at the same time, it indicates that the previous setup process has been completed and there is a new SETUP token in the FIFO.

- **SERVICED\_OUT\_PKT\_RDY bit**

Setting this bit to “1” clears the OUT\_PKT\_RDY flag to “0”.

- **SERVICED\_SETUP\_END bit**

Setting this bit to “1” clears the SETUP\_END flag to “0”.

Figure 2.6.14 shows the structure of the USB endpoint x (x = 0 to 2) IN control register.

### (8) USB endpoint 1, 2 IN control register

These registers comprise bits related to endpoints 1 and 2 control and status information.

#### ● IN\_PKT\_RDY bit

##### Single-buffer mode:

Set this bit to "1" after writing one packet of data to the IN FIFO (this bit is automatically set to "1" when AUTO\_SET bit is "1").

The USB FCU clears this bit to "0" when the IN FIFO transmit is completed (after the ACK is received from the host).

##### Double-buffer mode:

Set this bit to "1" every writing one packet of data to the IN FIFO. After writing one packet of data, if there is one packet in the FIFO, the IN\_PKT\_RDY bit is not set to "1"; only the TX\_NOT\_EPT flag is set to "1". If the FIFO has two packets (FIFO full state) after the write, both the IN\_PKT\_RDY bit and TX\_NOT\_EPT flag are set to "1" (refer to Table 2.6.2).

The USB FCU clears this bit to "0" when the IN FIFO transmit is completed (after the ACK is received from the host).

Nothing is changed even when writing "0" to this flag is executed. When you need to access this bit in a case such as setting other bits in the endpoint x IN control register is performed, clear this bit to "0" by program.

#### ● SEND\_STALL bit

Set this bit to "1" in the STALL state. When this bit is "1", the USB FCU sends a STALL handshake signal to the host CPU.

#### ● TOGGLE\_INIT bit

This bit initializes the toggle sequence bit.

Setting the TOGGLE\_INIT bit to "1" assigns the PID of the next packet of data to be transmitted to the host as DATA0. To initialize the data toggle sequence bit for an endpoint (reset the next data packet as DATA0), set this bit to "1" and then clear it to "0".

#### ● INTPT bit

Set this bit to "1" in order to use an IN endpoint for rate feedback interrupt transfer, as described in detail below.

1. Set the USB endpoint x IN maximum packet size register to a value larger than 1/2 the USB endpoint x FIFO size.
2. Set the INTPT bit to "1".
3. Flush out the old data in the FIFO.
4. Store the transmit data in the IN FIFO and set IN\_PKT\_RDY bit to "1".
5. Repeat steps 3 and 4.

When using the endpoint for rate feedback interrupt transfer in an actual application, the function-side always must have transmit data to send to the host. Therefore, in a rate feedback interrupt transfer, the device never returns a NAK in response to the IN token from the host. The device transmits the data in the FIFO in response to an IN token regardless of the IN\_PKT\_RDY status. However, this function in the 7643 Group assumes that there is an ACK response from the host after data is transmitted in response to an IN token.

#### ● TX\_NOT\_EPT flag

This flag is set to "1" when there is data in the IN FIFO.

#### ● FLUSH bit

Set this bit to "1" in order to erase the data in the FIFO. If there are two or more packets in the IN FIFO, the old data will be erased. Setting this bit to "1" during a USB transfer may cause the transmit data to be erased.

**● AUTO\_SET bit**

When this bit is “1”, if the data written to the IN FIFO matches the max. IN packet size, the IN\_PKT\_RDY bit is automatically set to “1”. However, when transmitting a short packet (fewer data packet than the max. packet size), set the IN\_PKT\_RDY bit to “1” after writing the data to the IN FIFO.

Figure 2.6.14 shows the structure of USB endpoint x (0 to 2) IN control registers.

### USB endpoint 0 IN control register

b	Name	Functions	At reset	R	W
0	OUT_PKT_RDY flag (IN0CSR0)	0 : Except the following condition (Cleared to "0" by writing "1" into SERVICED_OUT_PKT_RDY bit) 1 : End of a data packet reception	0	○	*1
1	IN_PKT_RDY bit (IN0CSR1)	0 : End of a data packet transmission 1 : Write "1" at completion of writing a data packet into IN FIFO.	0	○	*2
2	SEND_STALL bit (IN0CSR2)	0 : Except the following condition 1 : Transmitting STALL handshake signal	0	○	○
3	DATA_END bit (IN0CSR3)	0 : Except the following condition (Cleared to "0" after completion of status phase) 1 : Write "1" at completion of writing or reading the last data packet to/from FIFO.	0	○	*2
4	FORCE_STALL flag (IN0CSR4)	0 : Except the following condition 1 : Protocol error detected	0	○	*1
5	SETUP_END flag (IN0CSR5)	0 : Except the following condition (Cleared to "0" by writing "1" into SERVICED_SETUP_END bit) 1 : Control transfer ends before the specific length of data is transferred during the data phase.	0	○	*1
6	SERVICED_OUT_PKT_RDY bit (IN0CSR6)	0 : Except the following condition 1 : Writing "1" to this bit clears OUT_PKT_RDY flag to "0".	0	○	○
7	SERVICED_SETUP_END bit (IN0CSR7)	0 : Except the following condition 1 : Writing "1" to this bit clears SETUP_END flag to "0".	0	○	○

### USB endpoint 1, 2 IN control register

b	Name	Functions	At reset	R	W
0	INT_PKT_RDY bit (INXCSR0) *1	0 : End of a data packet transmission 1 : Write "1" at completion of writing a data packet into IN FIFO. <b>(Note 2)</b>	0	○	*2
1	Nothing is arranged for this bit. Fix this bit to "0".		0	○	○
2	SEND_STALL bit (INXCSR2)	0 : Except the following condition 1 : Transmitting STALL handshake signal	0	○	○
3	TOGGLE_INIT bit (INXCSR3)	0 : Except the following condition 1 : Initializing the data toggle sequence bit	0	○	○
4	INTPT bit (INXCSR4)	0 : Except the following condition 1 : Initializing to endpoint used for interrupt transfer, rate feedback	0	○	○
5	TX_NOT_EPT flag (INXCSR5)	0 : Empty in IN FIFO 1 : Full in IN FIFO	0	○	×
6	FLUSH bit (INXCSR6)	0 : Except the following condition 1 : Flush FIFO	0	○	*1
7	AUTO_SET bit (INXCSR7)	0 : AUTO_SET disabled 1 : AUTO_SET enabled *3	0	○	○

\* 1: "1" can be set by software, but "0" cannot be set.

\* 2: When INXCSR7="1", this bit is automatically set to "1".  
When INXCSR7="0", writing data to FIFO, and then write "1" to this bit.

\* 3: To use the AUTO\_SET function for an IN transfer when the AUTO\_SET bit is set to "1", set the FIFO to single buffer mode.

Fig. 2.6.14 Structure of USB endpoint x IN control register

**(9) USB endpoint x (x = 1, 2) OUT control registers**

When using endpoint 0, these registers are the reserved registers and cannot be assigned to any function.

**● OUT\_PKT\_RDY flag**

This flag is set to "1" when a data receive from the host CPU is completed. Clear the flag to "0" after the data is read from the OUT FIFO. This flag is automatically cleared to "0" when the AUTO\_CLR bit is "1". To avoid internal Read pointer malfunction, do not clear the flag to "0" until reading out one packet of data from the host is completed. The value of the USB endpoint x OUT write count register is updated when this flag is cleared.

When two or more data packets are written to the OUT FIFO in the double-buffer mode, the flag is set to "0" if "0" is written to it after one packet of data is read, but returns to "1" after 83ns ( $V_{CC} = 5V$ ,  $f(X_{IN}) = 24MHz$ ).

USB operations are not affected by writing "1" to this flag. Write "1" to the OUT\_PKT\_RDY flag to access it.

**● SEND\_STALL bit**

Set this bit to "1" in the STALL state. When this bit is "1", the USB FCU sends a STALL handshake signal to the host CPU.

**● TOGGLE\_INIT bit**

This bit initializes the toggle sequence bit.

Setting the TOGGLE\_INIT bit to "1" assigns the PID of the next packet received from the host as DATA0. To initialize the data toggle sequence bit of an endpoint (reset the next data packet as DATA0), set this bit to "1" and then clear it to "0".

**● FORCE\_STALL flag**

This flag is set to "1" when data received from the host CPU is larger than the max. OUT packet size. The USB FCU then sends a STALL handshake signal to the host CPU. Write "0" to this flag to clear it.

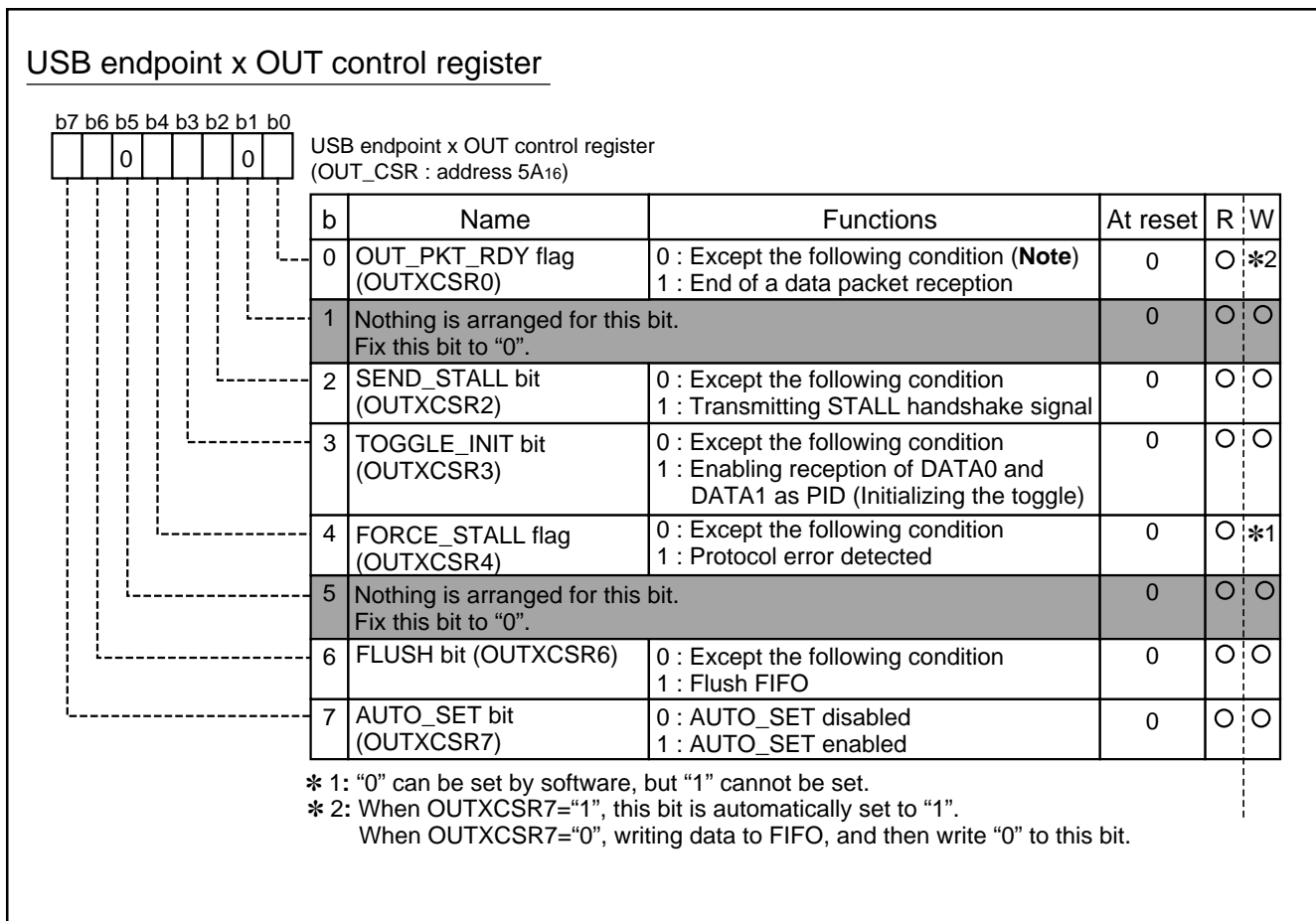
**● FLUSH bit**

Set this bit to "1" to erase the data in the OUT FIFO. When there are two or more packets of data in the OUT FIFO, the old data will be erased. Setting this bit to "1" during a transfer may cause the receive data to be erased. When setting the OUT\_PKT\_RDY flag to "1", also set this bit to "1".

**● AUTO\_CLR bit**

When this bit is "1" and the size of the data read from the OUT FIFO matches that of the received OUT packet, the OUT\_PKT\_RDY flag is automatically cleared to "0".

Figure 2.6.15 shows the structure of the USB endpoint x (x = 1, 2) OUT control register.



**Fig. 2.6.15 Structure of USB endpoint x (x=1 to 2) OUT control register**

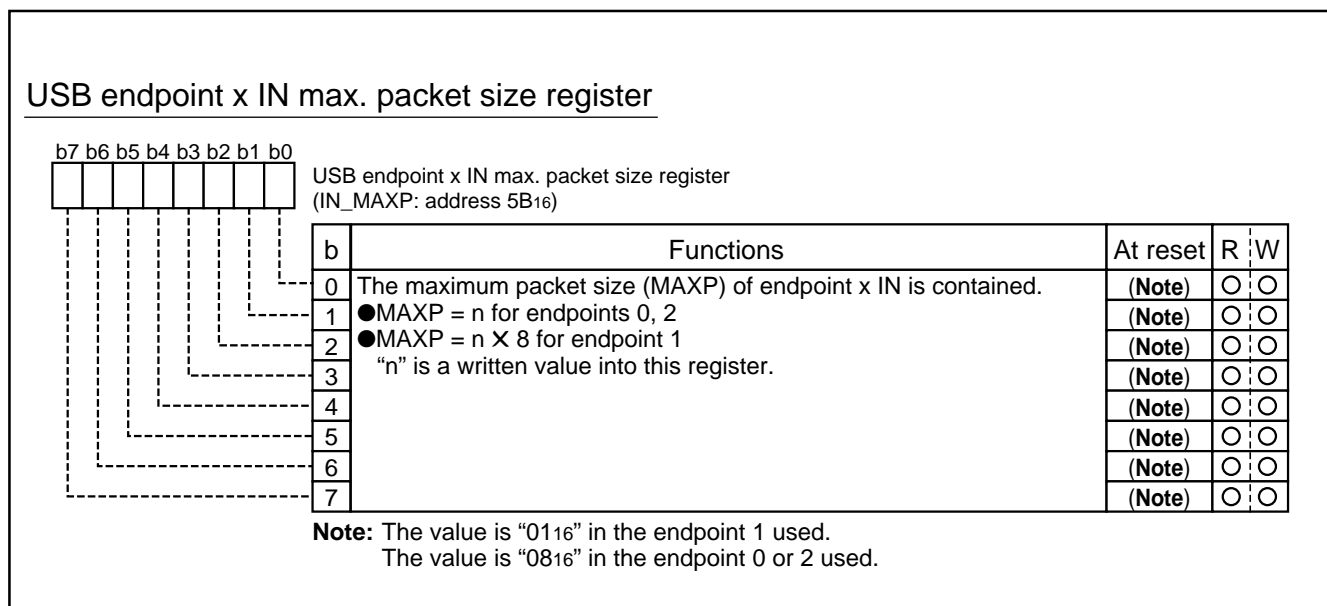
**(10) USB endpoint x (x = 0 to 2) IN max. packet size register**

This register specifies the maximum packet size (MAXP) of an endpoint x (x = 0 to 2) IN packet. Modify the contents of this register when the SET\_DESCRIPTOR command is received from the host CPU. The IN maximum packet sizes (MAXP) are as follows:

- When using endpoints 0, 2: MAXP = n
- When using endpoint 1: MAXP = n\*8 (n: the value set in this register)

When not using this register, clear "0" to this register.

Figure 2.6.16 shows the structure of the USB endpoint x (x = 0 to 2) IN max. packet size register.



**Fig. 2.6.16 Structure of USB endpoint x (x=0 to 2) IN max. packet size register**

**(11) USB endpoint x (x = 0 to 2) OUT max. packet size register**

This register specifies the maximum packet size (MAXP) of an endpoint x (x = 0 to 2) OUT packets. Modify the contents of this register when the SET\_DESCRIPTOR command is received from the host CPU. The OUT maximum packet sizes (MAXP) are as follows:

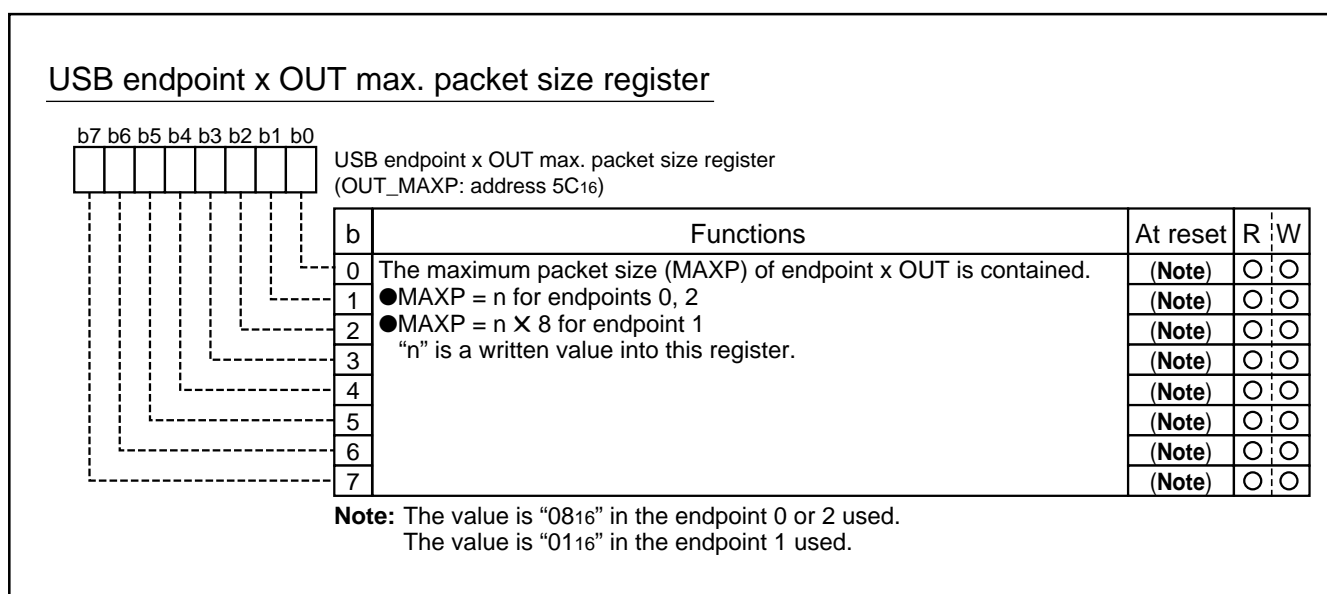
- When using endpoint 2: MAXP = n
- When using endpoint 1: MAXP = n\*8 (n: the value set in this register)

When not using this register, clear "0" to this register.

When using the endpoint 0, both USB endpoint x IN max. packet size register (IN\_MAXP) and USB endpoint x OUT max. packet size register (OUT\_MAXP) are set to the same value. Changing one register's value effectively changes the value of the other register as well.

If the maximum packet size is larger than 1/2 the FIFO size, one packet of data can be stored in the FIFO (single-buffer). If the maximum packet size is less than 1/2 the FIFO size, two packets can be stored (double-buffer).

Figure 2.6.17 shows the structure of the USB Endpinx (x = 0 to 2) OUT max. packet size register.



**Fig. 2.6.17 Structure of USB endpoint x (x=0 to 2) OUT max. packet size register**

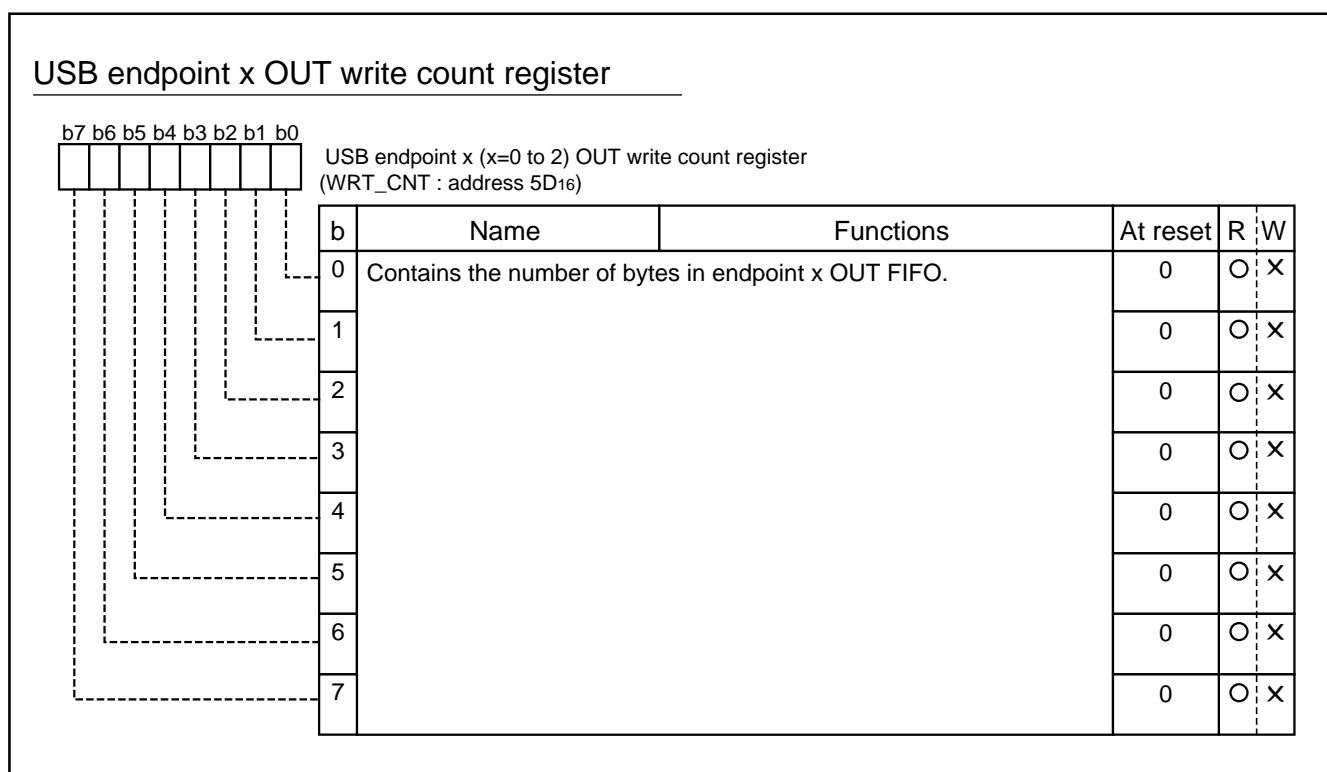


**(12) USB endpoint x (x = 0 to 2) OUT write count register**

This is an 8-bit register that contains the number of bytes in the endpoint x (x = 0 to 2) OUT FIFO. This register must be read after the USB FCU has received a packet of data from the host CPU. Refer to this register when reading the data from the OUT FIFO.

When there are two packets in the FIFO (double-buffer), the number of bytes of data in the first packet received will be read out from the register. After one packet of data is read from the FIFO and the OUT\_PKT\_RDY flag is cleared, the value of the register will be updated to the number of bytes of the packet received last.

Figure 2.6.18 shows the structure of the USB endpoint x (x = 0 to 2) OUT write count register.

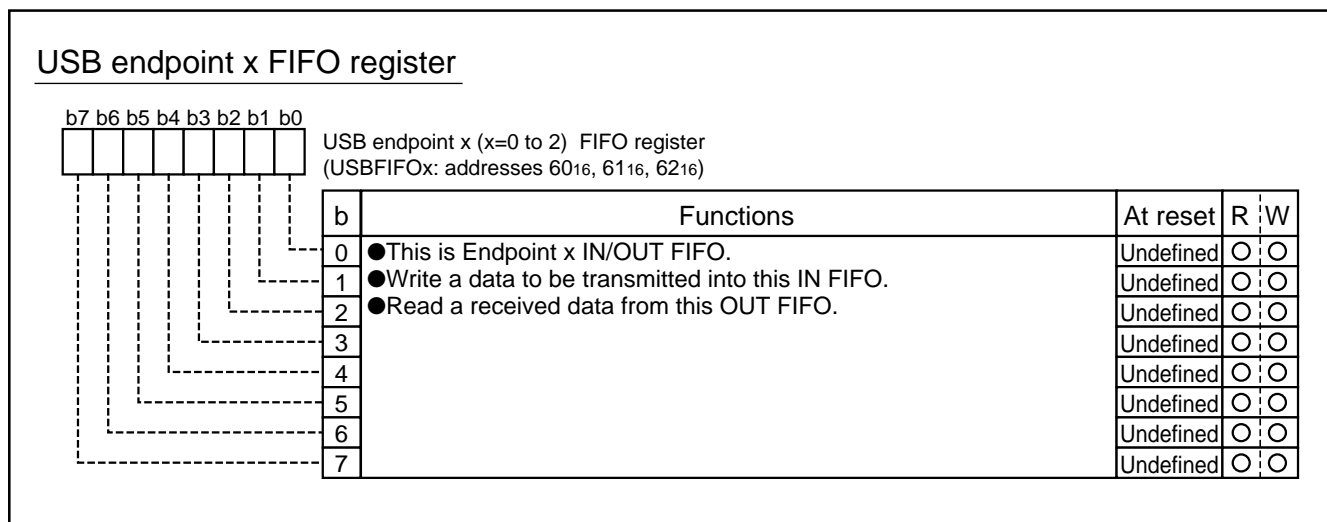


**Fig. 2.6.18 Structure of USB endpoint x (x=0 to 2) OUT write count register**

**(13) USB endpoint x (x = 0 to 2) FIFO register**

These registers are the USB IN (transmit) and OUT (receive). Write data to the corresponding register, and read data from the corresponding register. When the maximum packet size is equal to or less than 1/2 the FIFO size, these registers function in double buffer mode and can hold two packets of data.

Figure 2.6.19 shows the structure of the USB endpoint x (x = 0 to 2) FIFO register.

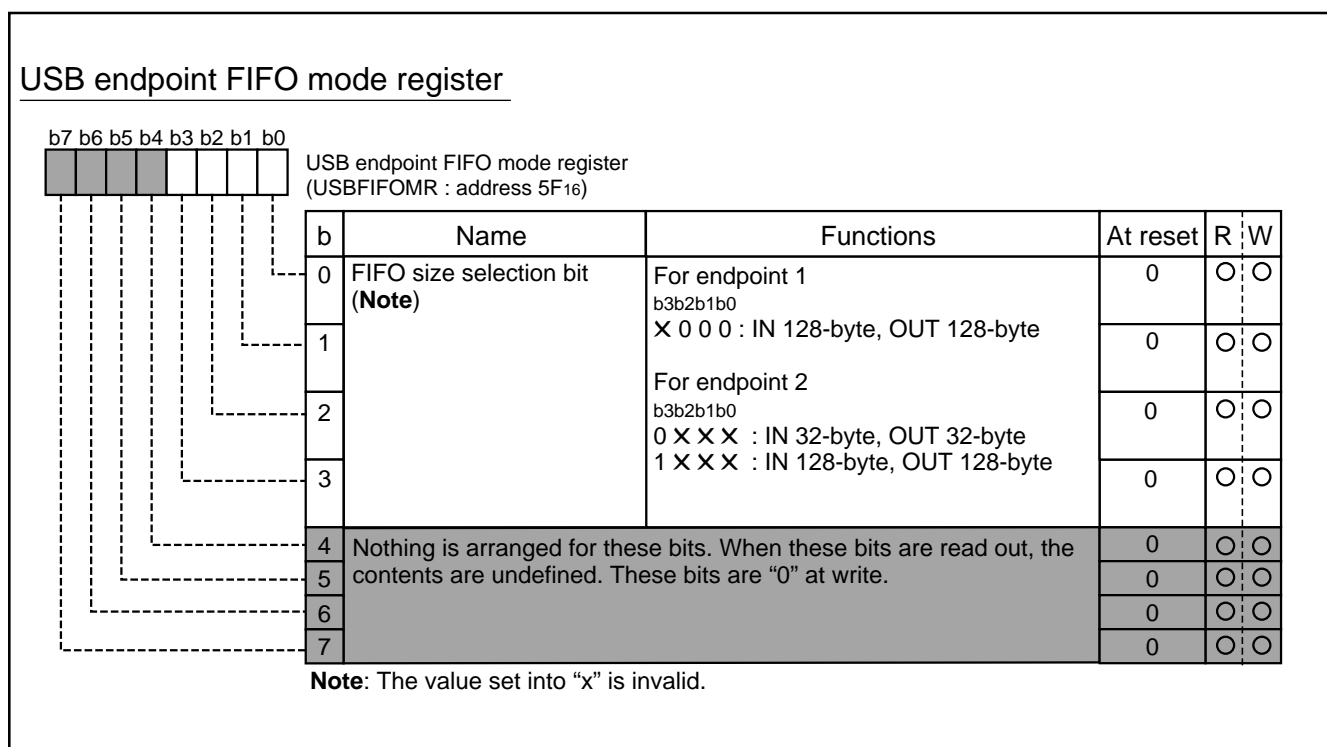


**Fig. 2.6.19 Structure of USB endpoint x (x=0 to 2) FIFO register**

**(14) USB endpoint FIFO mode register**

This register determines the IN/OUT FIFO size mode for endpoint 1 or endpoint 2.

Figure 2.6.20 shows the structure of the USB endpoint x FIFO mode register.



**Fig. 2.6.20 Structure of USB endpoint FIFO mode register**

**(15) Clock control register**

This register controls all oscillators.

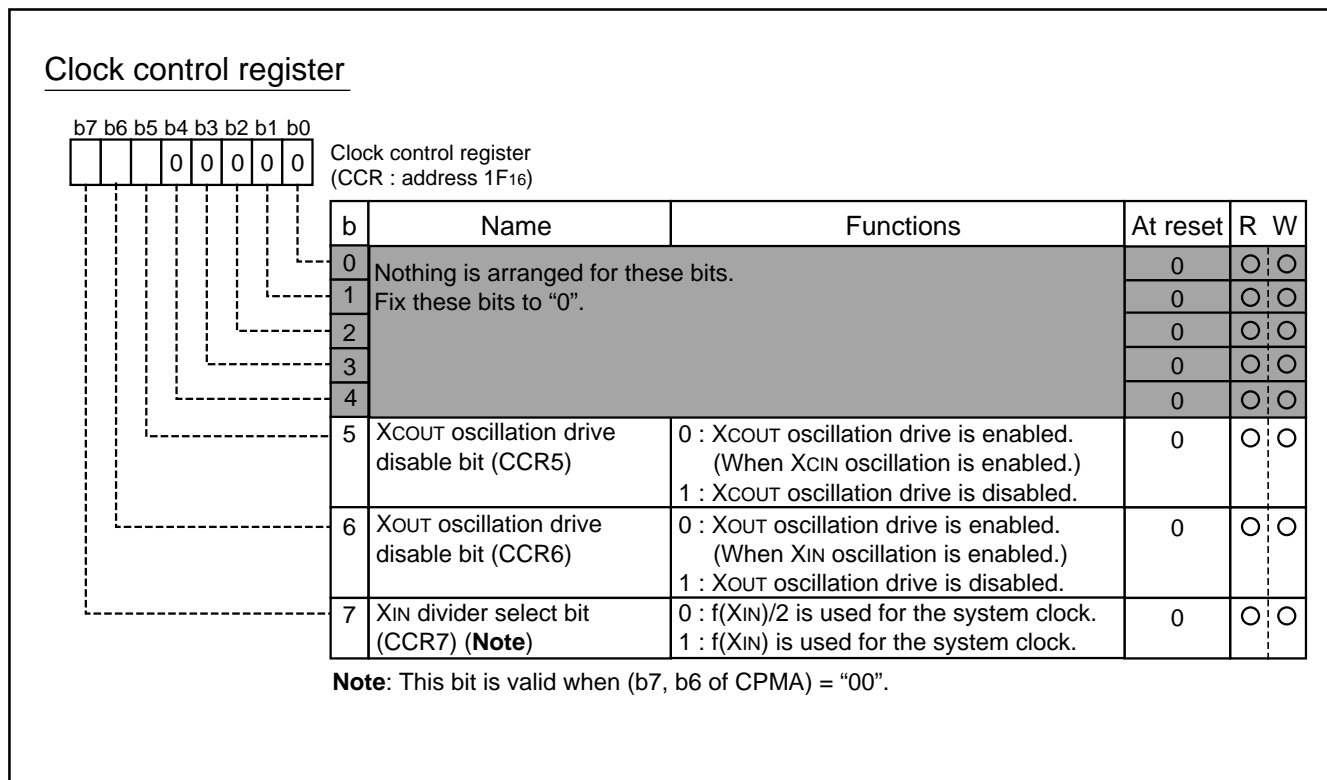
The X<sub>IN</sub> divider select bit is valid when CPMA6 and CPMA7 = "00"

When "0": f(X<sub>IN</sub>)/4 is the internal φ clock.

When "1": f(X<sub>IN</sub>)/2 is the internal φ clock.

Note that this bit is cleared to "0" by execution of the STP instruction.

Figure 2.6.21 shows the structure of the clock control register.



**Fig. 2.6.21 Structure of clock control register**

**(16) Frequency synthesizer control register**

This register performs all controls related to the frequency synthesizer, which generates  $f_{USB}$  and  $f_{SYN}$  from  $f(X_{IN})$ .

- **Frequency synthesizer enable bit**

Set this bit to “1” to use the frequency synthesizer.

- **Frequency synthesizer input bit**

When this bit is “0”,  $f(X_{IN})$  is used as the frequency synthesizer input.

When “1”,  $f(X_{CIN})$  is used.

- **LPF current control bit**

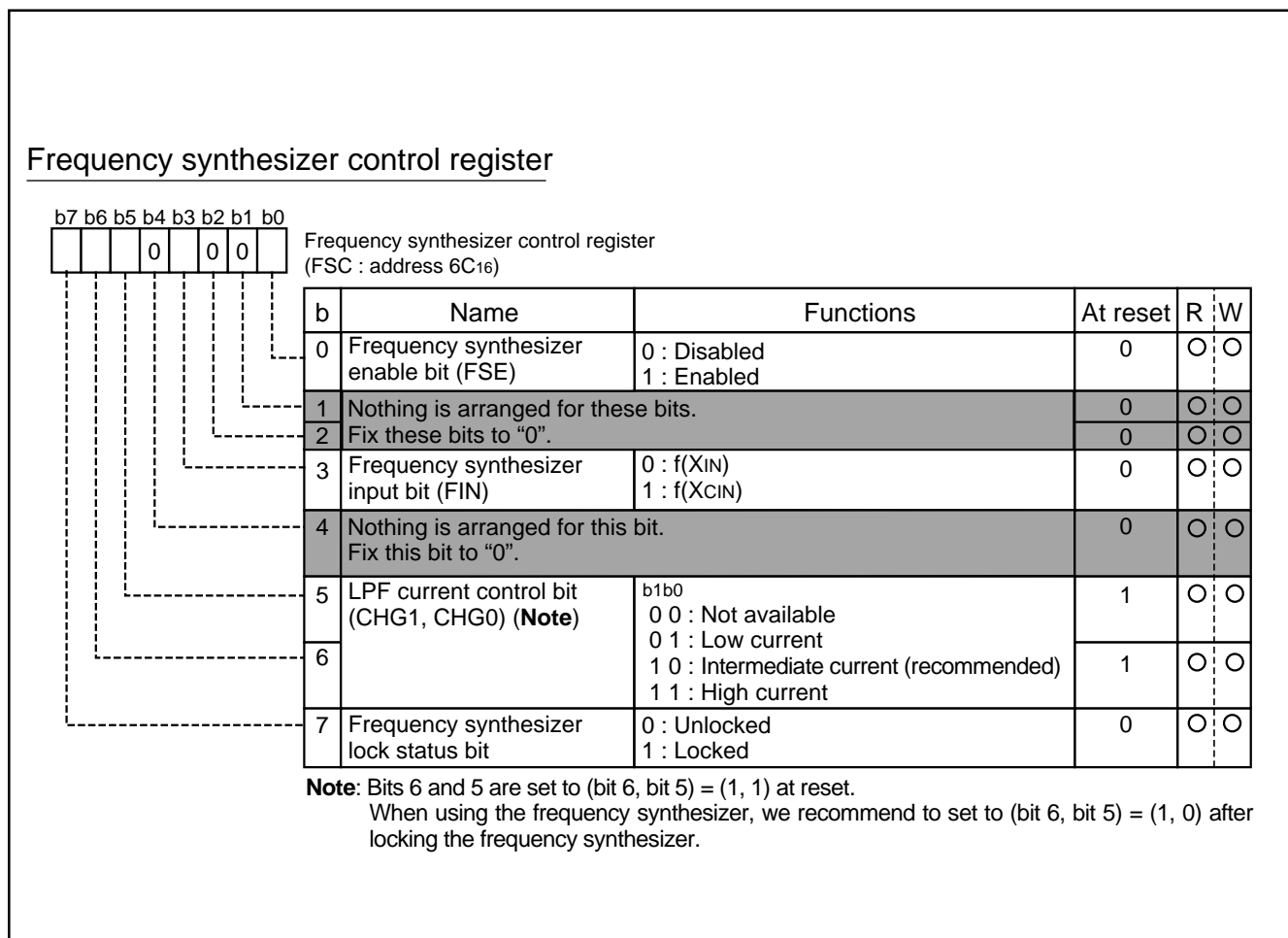
Bits 6 and 5 = “11” when reset is released.

To use the frequency synthesizer, make sure to specify bits 6 and 5 to “10” after the frequency synthesizer is locked.

- **Frequency synthesizer lock status bit**

When this bit is set to “1”, it indicates that both  $f_{SYN}$  and  $f_{VCO}$  have stabilized.

Figure 2.6.22 shows the structure of the frequency synthesizer control register.



**Fig. 2.6.22 Structure of frequency synthesizer control register**

### 2.6.4 USB transmit

Endpoint 0 to endpoint 2 have IN (transmit) FIFOs individually.

Each endpoint's FIFO is configured in following way:

Endpoint 0: 16-byte

Endpoint 1: 128-byte

Endpoint 2: Mode 0: 32-byte

Mode 1: 128-byte

When endpoint 2 is used, the IN FIFO size differs according to the mode.

To select the mode, use the USB endpoint FIFO mode register (address 005F<sub>16</sub>).

USB transmit data is written to the USB endpoint x (x = 0 to 2) FIFO (addresses 0060<sub>16</sub> to 0062<sub>16</sub>). When data is written to this register, the internal Write pointer is automatically incremented by 1. The contents of the internal Write pointer cannot be read out, and the contents of the FIFO are undefined at reset.

#### ● When using endpoint 0

Set the IN\_PKT\_RDY bit to "1" after writing transmit data to the IN FIFO. The IN\_PKT\_RDY bit is set to "0" automatically after one packet of data is transmitted or when the SETUP\_END flag is set to "1".

When using endpoint 0, set the DATA\_END bit to "1" when the last data packet has been written to the IN FIFO. When the DATA\_END bit is set to "1", the USB FCU will go on to the next phase process and clear the bit to "0".

#### ● When using endpoints 1, 2

##### When max. packet size is greater than 1/2 (single buffer)

- When AUTO\_SET bit = "1"

The IN\_PKT\_RDY bit (and TX\_NOT\_EPT flag) is automatically set to "1" when the number of bytes of data written to the IN FIFO equals the value set in the endpoint x IN max. packet size register (address 005B<sub>16</sub>).

- When AUTO\_SET bit = "0" or when a short packet (smaller than max. packet size) is transferred  
Set the IN\_PKT\_RDY bit to "1" by software.

##### When max. packet size is less than or equal to 1/2 (double buffer)

Set the IN\_PKT\_RDY bit to "1" by software after one packet has been written to the IN FIFO. When one packet data is set in the IN FIFO and there is only one packet stored in the IN FIFO, the IN\_PKT\_RDY bit is set to "1". However, it is cleared after 83ns (when  $f(X_{IN}) = 24\text{MHz}$ ,  $V_{CC} = 5\text{V}$ ), and the TX\_NOT\_EPT flag then is set to "1". If there are already two packets in the IN FIFO (IN FIFO full), both the IN\_PKT\_RDY bit and the TX\_NOT\_EPT flag are set to "1".

In the single-buffer mode, the IN\_PKT\_RDY bit and the TX\_NOT\_EPT flag will be automatically cleared to "0" after completion of the transmission.

In the double-buffer mode, if there is no data in the IN FIFO, the IN\_PKT\_RDY bit and the TX\_NOT\_EPT flag will be automatically cleared to "0" after completion of the transmission. If there is one packet data already in the IN FIFO, the IN\_PKT\_RDY bit will be set to "0" after completion of the transmission, but the TX\_NOT\_EPT flag will remain at "1".

The USB transmit default mode is the bulk transfer.

To use any other transfer mode, the user needs to initialize the corresponding endpoints.

**(1) Interrupt transfer**

Endpoints 1 to 2 can be used in interrupt transfer mode. During a regular interrupt transfer, an interrupt transaction is similar to the bulk transfer. Therefore, there is no special setting required. When IN-endpoint is used for a rate feedback interrupt transfer, INTPT bit of the IN\_CSR register must be set to "1". The following steps show how to configure the IN-endpoint for the rate feedback interrupt transfer.

1. Set a value which is larger than 1/2 of the USB endpoint-x FIFO size to the USB endpoint x IN max. package size register.
2. Set INTPT bit to "1".
3. Flush the old data in the FIFO.
4. Store transmission data to the IN FIFO and set the IN\_PKT\_RDY bit to "1".
5. Repeat steps 3 and 4.

In a real application, the function-side always has transfer data when the function sends an endpoint in a rate feedback interrupt. Accordingly, the USB FCU never returns a NAK against the host IN token for the rate feedback interrupt. The USB FCU always transmits data in the FIFO in response to an IN token, regardless of IN\_PKT\_RDY. However, this premises that there is always an ACK response from Host PC after the 7643 Group has transmitted data to IN token.

**(2) Usage notes concerning data transmit**

Usage notes concerning IN FIFO

Determine the number of data packets in the IN FIFO by the value of the IN\_PKT\_RDY bit and the TX\_NOT\_EPT flag. Erase the contents of the USB endpoint x IN FIFO by setting the FLUSH bit to "1". This will also modify the state of the IN\_PKT\_RDY bit and the TX\_NOT\_EPT flag.

If there are two packets in the FIFO, the older packet will be erased.

Table 2.6.2 shows the states of the IN FIFO.

**Table 2.6.2 IN FIFO States**

IN_PKT_RDY	TX_NOT_EMP	IN FIFO States
0	0	No data packet
0	1	1 data packet (max. packet size $\leq$ 1/2 IN FIFO size) In valid if max. packet size $>$ 1/2 size
1	0	Invalid
1	1	1 data packet (max. packet size $>$ 1/2 size) 2 data packets (max. packet size $\leq$ 1/2 IN FIFO size)

### 2.6.5 USB receive

Endpoint 0 to endpoint 2 have OUT (transmit) FIFOs individually.

Each endpoint's FIFO is configured in following way:

Endpoint 0: 16-byte

Endpoint 1: 128-byte

Endpoint 2: Mode 0: 32-byte  
Mode 1: 128-byte

When endpoint 2 is used, the OUT FIFO size differs according to the mode.

To select the mode, use the USB endpoint FIFO mode register (address 005F<sub>16</sub>).

USB receive data is read from the USB endpoint x (x = 0 to 2) OUT FIFO.

When data is read from this register, the internal Write pointer is automatically decremented by 1.

The contents of the internal Write pointer cannot be read out, and the contents of the FIFO are undefined at reset.

#### When using endpoint 0

When a receive is completed, the OUT\_PKT\_RDY flag is set to "1" and the number of receive data is set in the USB endpoint x (x = 0 to 2) OUT write count register (address 005D<sub>16</sub>). After data is read from the OUT FIFO, clear the OUT\_PKT\_RDY flag to "0" by writing "1" to the SERVICED\_OUT\_PKT\_RDY bit. When using endpoint 0, set the DATA\_END bit to "1" after the last data packet is read from the OUT FIFO. When the DATA\_END bit is set to "1", the USB FCU will go on to the next phase process and clear the bit to "0".

#### When using endpoints 1, 2

When a receive is completed, the OUT\_PKT\_RDY flag is set to "1" and the number of bytes of receive data is set in the USB endpoint x (x = 0 to 2) OUT write count register (address 005D<sub>16</sub>).

##### When max. packet size is greater than 1/2 (single-buffer)

- When AUTO\_CLR bit = "1"

When data (the size equals received OUT packet) is read from the OUT FIFO, the OUT\_PKT\_RDY flag is automatically cleared to "0".

- When AUTO\_CLR bit = "0"

The user needs to clear the OUT\_PKT\_RDY flag through software.

##### When max. packet size is less than or equal to 1/2 (double-buffer)

- When AUTO\_CLR bit = "1"

When data (the size equals received OUT packet) is read from the OUT FIFO but there is no data in the OUT FIFO, the OUT\_PKT\_RDY flag automatically is set to "0". If there is one packet of data in the OUT FIFO, OUT\_PKT\_RDY flag will be cleared to "0" but then set to "1" after 83ns (when  $f(X_{IN}) = 24\text{MHz}$ ,  $V_{CC} = 5\text{V}$ ).

- When AUTO\_CLR bit = "0"

Clear the OUT\_PKT\_RDY flag to "0" after reading data from the OUT FIFO (it will not be cleared automatically). When there is no data in the OUT FIFO, the OUT\_PKT\_RDY flag will be set to "0". If there is one packet of data in the OUT FIFO, OUT\_PKT\_RDY flag is cleared to "0" once, then set to "1" after 83ns (when  $f(X_{IN}) = 24\text{MHz}$ ,  $V_{CC} = 5\text{V}$ ).

#### (1) Usage notes concerning data receive

##### ● OUT\_PKT\_RDY flag

Read one packet data from the OUT FIFO before clearing the OUT\_PKT\_RDY flag. If the OUT\_PKT\_RDY flag is cleared while one packet data is being read, the internal Read pointer cannot operate normally.

##### ● OUT FIFO

For endpoints 1 or 2, when there are two packets of data in the OUT FIFO, after reading out the max OUT packet size of data, one packet of data will remain in the OUT FIFO. In this case, even if the OUT\_PKT\_RDY flag is set to "0", it will return to "1" after 83ns ( $V_{CC} = 5\text{V}$ ,  $f(X_{IN}) = 24\text{MHz}$ ).

## 2.6.6 USB interrupts

### (1) USB function interrupt

The USB function interrupt is used for data flow control as well as USB power management. In order to use a USB function interrupt, set the Interrupt disable flag (I) to "0" and USB function interrupt enable bit of interrupt control register A (address 0005<sub>16</sub>, bit 0) to "1". In addition, set all bits corresponding to USB interrupt enable register 1 (address 0054<sub>16</sub>) and USB interrupt enable register 2 (address 0055<sub>16</sub>).

Endpoint x IN interrupt, endpoint x OUT interrupt

An interrupt request occurs when the following flag is set to "1".

USB endpoint x IN/OUT interrupt status flag of USB interrupt status register 1 or 2 (addresses 0052<sub>16</sub>, 0053<sub>16</sub>)

Endpoints 1 and 2 have two interrupt status bits (IN and OUT) each and endpoint 0 has one interrupt status bit. Each interrupt status flag is set to "1" under any of the following conditions.

- USB endpoint 0 interrupt status flag (INTST0) (when using endpoint 0)
  - Endpoint 0: one packet successfully received, or
  - Endpoint 0: one packet successfully transmitted, or
  - DATA\_END bit is "0", or
  - FORCE\_STALL flag is "1", or
  - SETUP\_END flag is "1".
- USB endpoint x (x = 1, 2) IN interrupt status flag (INTST2, 4) (when using endpoints 1, 2):
  - Endpoint x (x = 1, 2): one packet data successfully transmitted
- USB endpoint x (x = 1, 2) OUT interrupt status flag (INTST3, 5) (when using endpoints 1, 2):
  - Endpoint x (x = 1, 2): one packet data successfully received, or
  - FORCE\_STALL flag is "1".

#### ● Reset interrupt

An interrupt request occurs when the USB reset interrupt status flag of USB interrupt status register 2 (0053<sub>16</sub>) is "1". The USB reset interrupt status flag is set to "1" when the USB FCU detects SE0 for a period of 2.5 ms on the D+/D- line by the USB FCU.

At USB reset, all USB internal registers (addresses 0050<sub>16</sub> to 005F<sub>16</sub>) bits, excluding this bit, are initialized. Initialize each endpoint when new data transfer is received from the host CPU.

#### ● Resume signal interrupt

An interrupt request occurs when the USB resume signal interrupt status flag of the USB interrupt status register 2 (address 0053<sub>16</sub>) is "1".

The USB resume signal interrupt status flag is set to "1" if a non-idle signal is detected on the D+/D- line when the USB FCU is in the suspend mode. Also, the USB resume signal interrupt status flag is set to "1" even when the MCU returns from the suspend mode by a remote wake-up (INT interrupt, etc.) and the USB remote wake-up bit is set to "1".

#### ● Suspend signal interrupt

An interrupt request occurs when the USB suspend signal interrupt status flag of the USB interrupt status register 2 (address 0053<sub>16</sub>) is "1".

This flag is set to "1" if no activity is detected on the D+/D- line for a period of 3ms.

The MCU returns from the suspend state with a USB resume interrupt or remote wake-up.



### 2.6.7 Application example

The following are several 7643 Group application examples showing the control procedure for performing USB controls.

#### (1) Application example 1: Initialization of USB function control unit

**Outline:** Initialization the USB function control unit (USB FCU) is performed.

**Specification:**

- To generate the 48MHz required for  $f_{USB}$ , the frequency synthesizer is used.
- USB FCU is enabled.
- Each endpoint is initialized.
  - Endpoint 0: Control transfer
  - Endpoint 1: Bulk transfer
  - Endpoint 2: Bulk transfer

This is an example when the MCU is operating at  $V_{CC} = 5V$ .

The following settings are required when using  $V_{CC} = 3V$ .

- Set the USB line driver supply enable bit (USB control register, bit 4) to "0" to disable the DC-DC converter.
- Set  $\phi$  to 6MHz or less.
  - When  $f(X_{IN}) = 24MHz$ , clear bit 7 of the clock control register ( $X_{IN}$  divider select bit) to "0".

Figure 2.6.23 shows the division settings of the frequency synthesizer, Figures 2.6.24 to 2.6.26 show related register settings, and Figures 2.6.27 to 2.6.29 show control procedure examples.

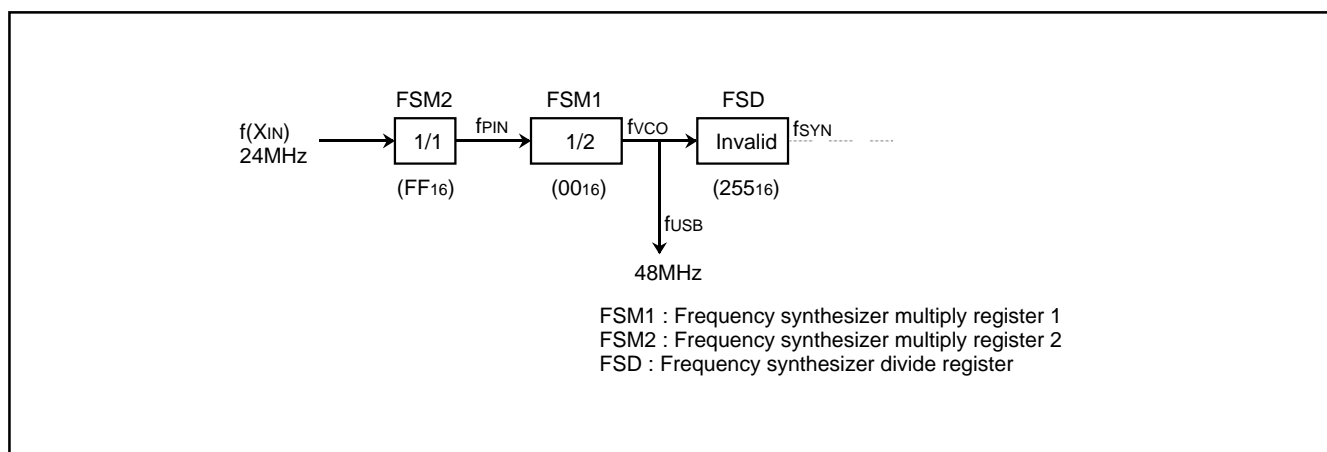


Fig. 2.6.23 Frequency synthesizer connection and setting of division ratios

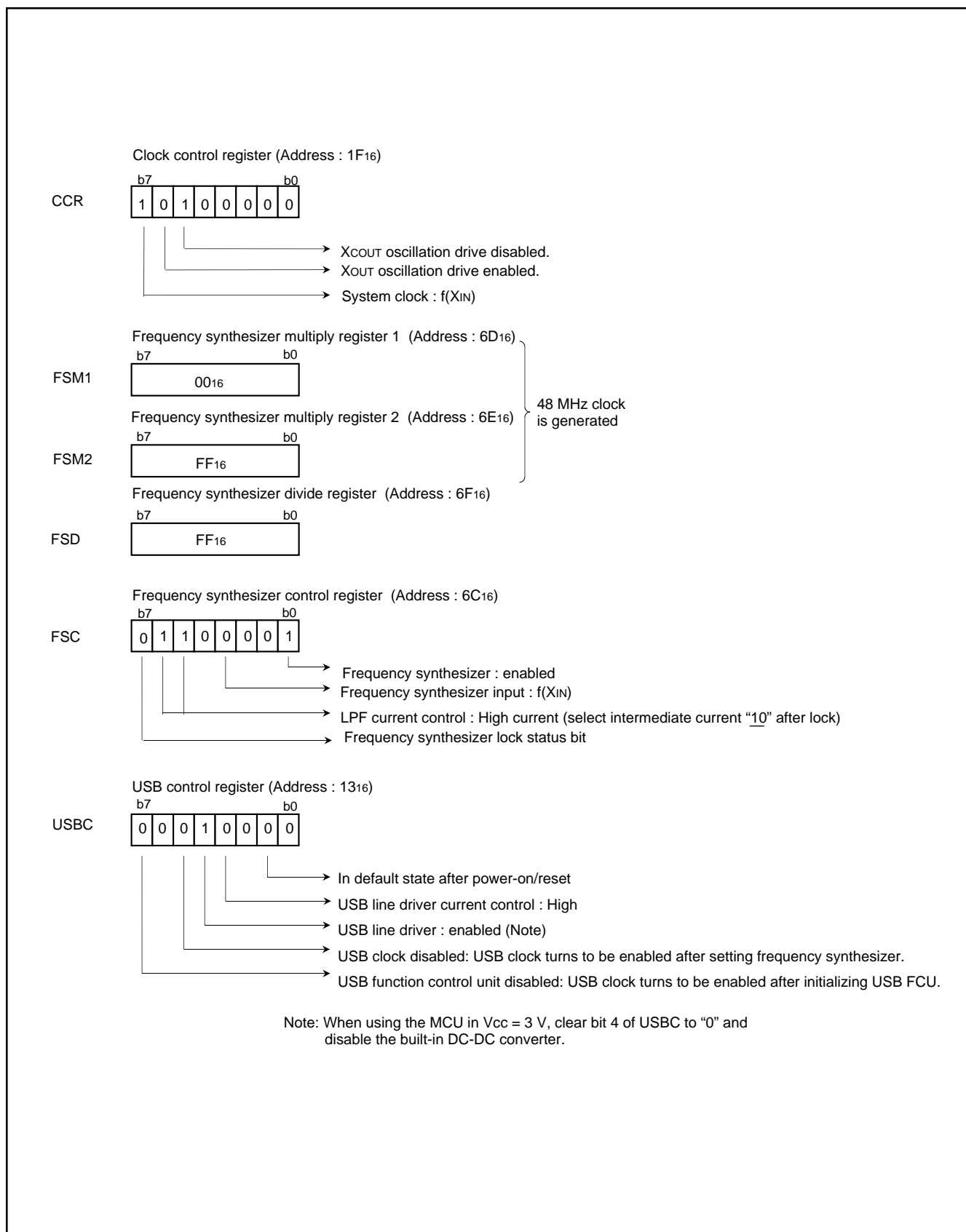


Fig. 2.6.24 Registers setting (1)

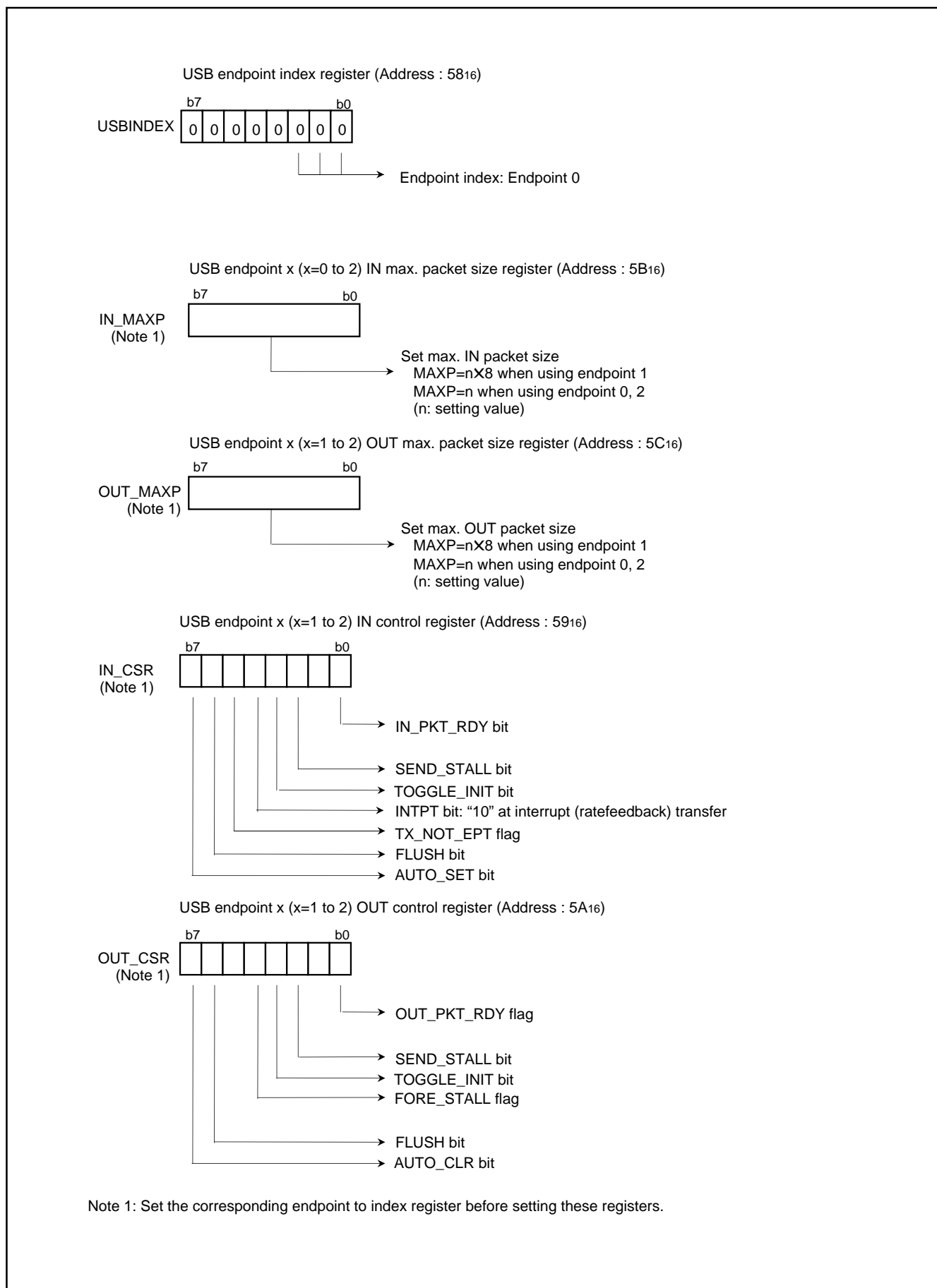


Fig. 2.6.25 Registers setting (2)

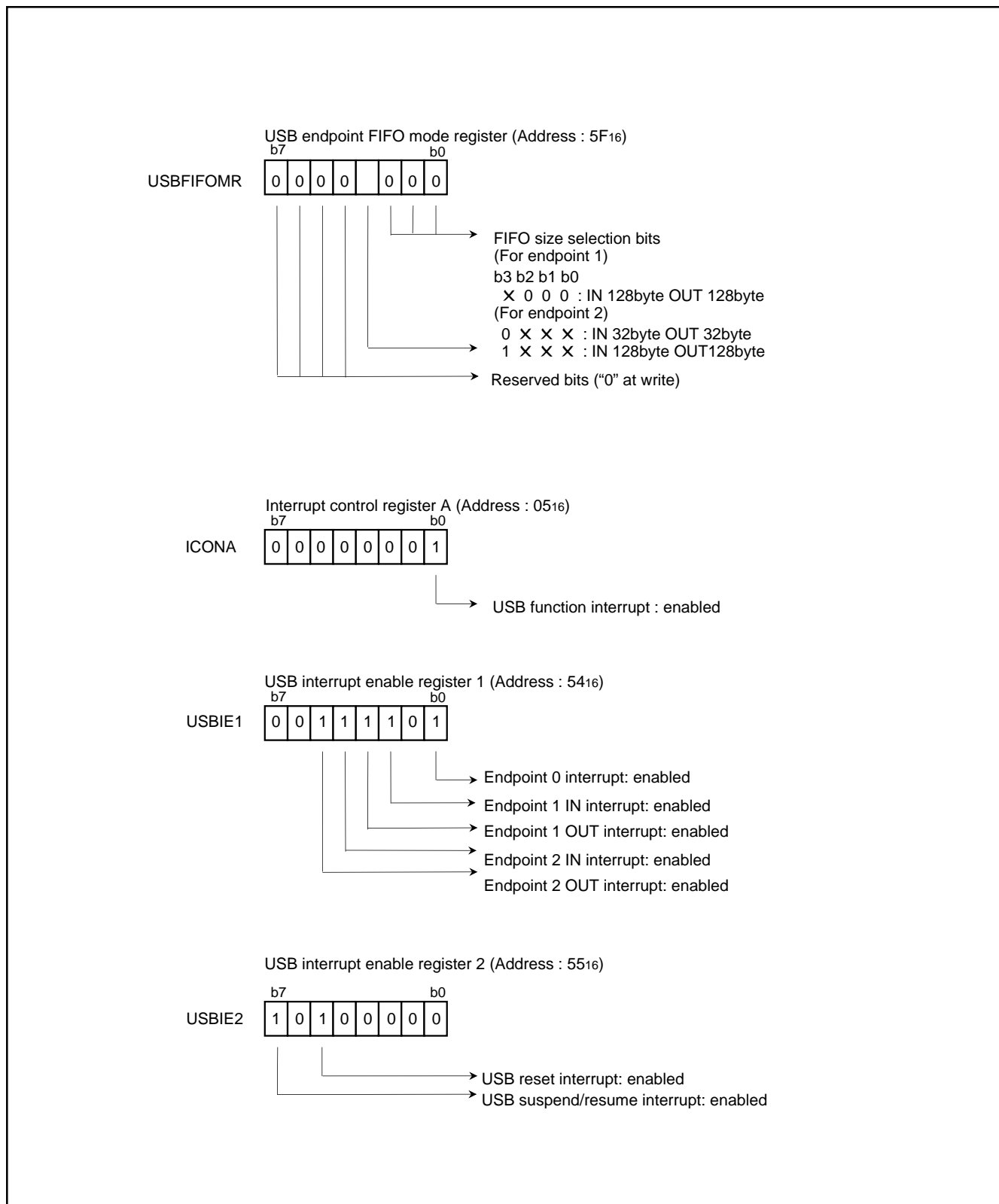
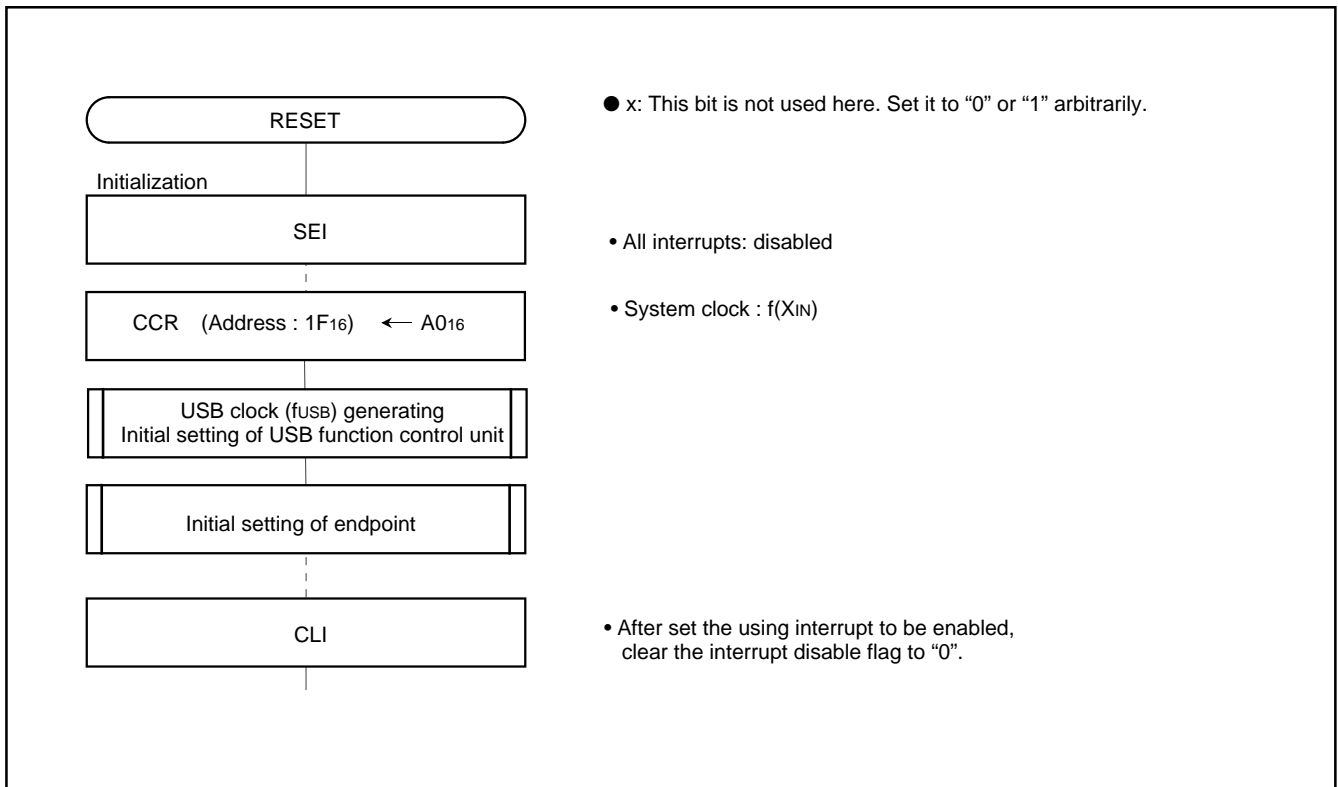


Fig. 2.6.26 Registers setting (3)



**Fig. 2.6.27 Control procedure (1) (USB block initial setting)**

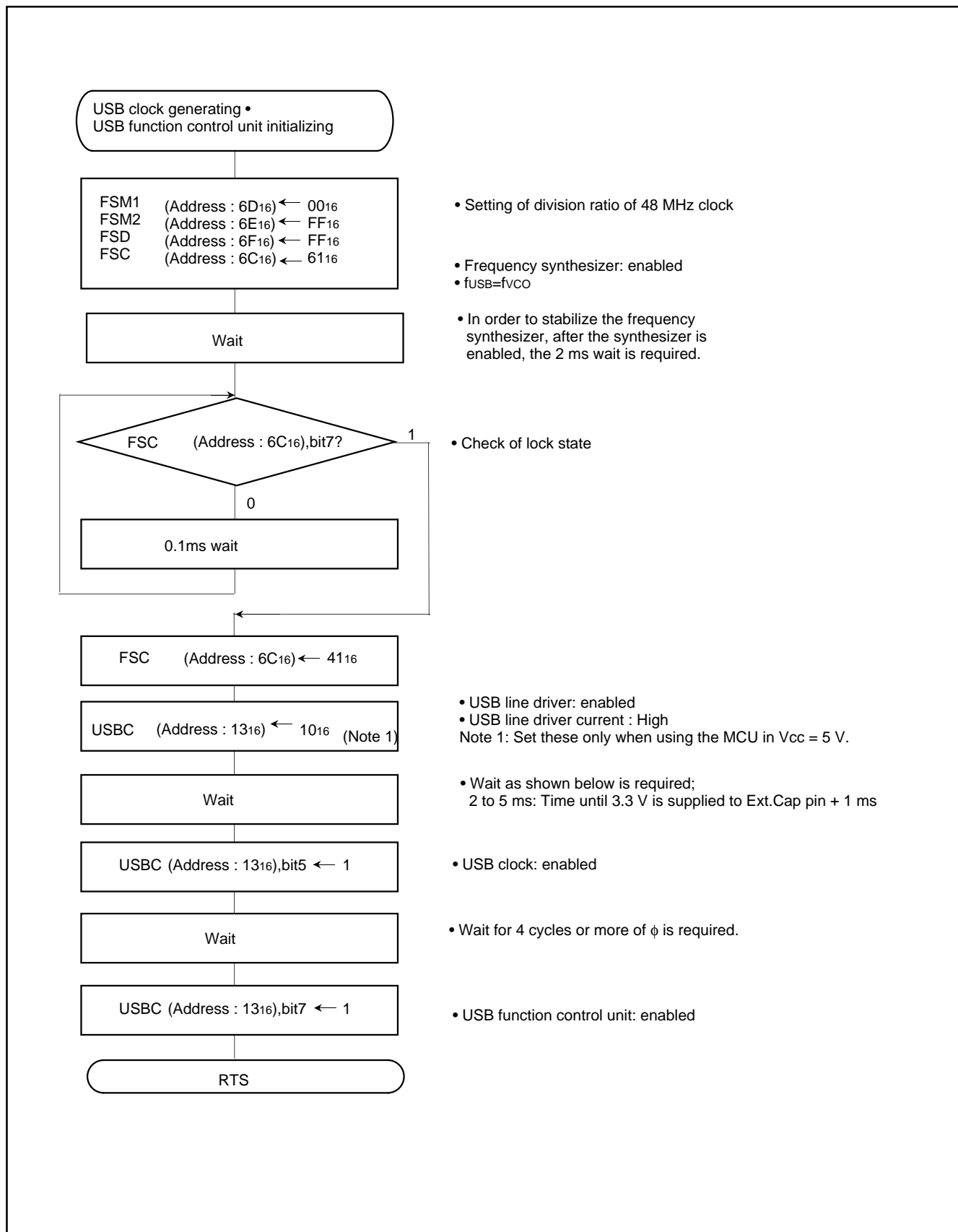


Fig. 2.6.28 Control procedure (2) (USB block generating)

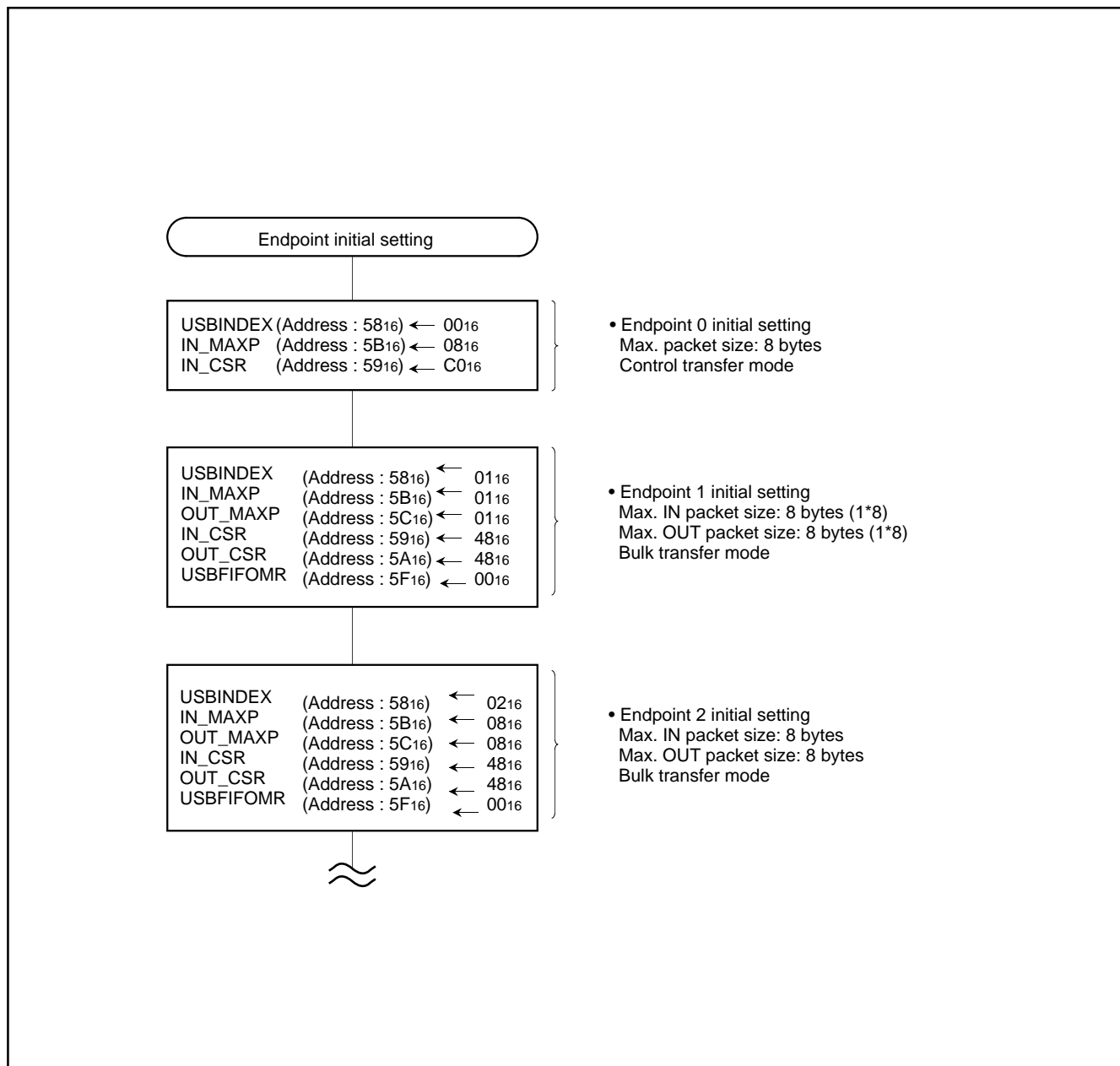


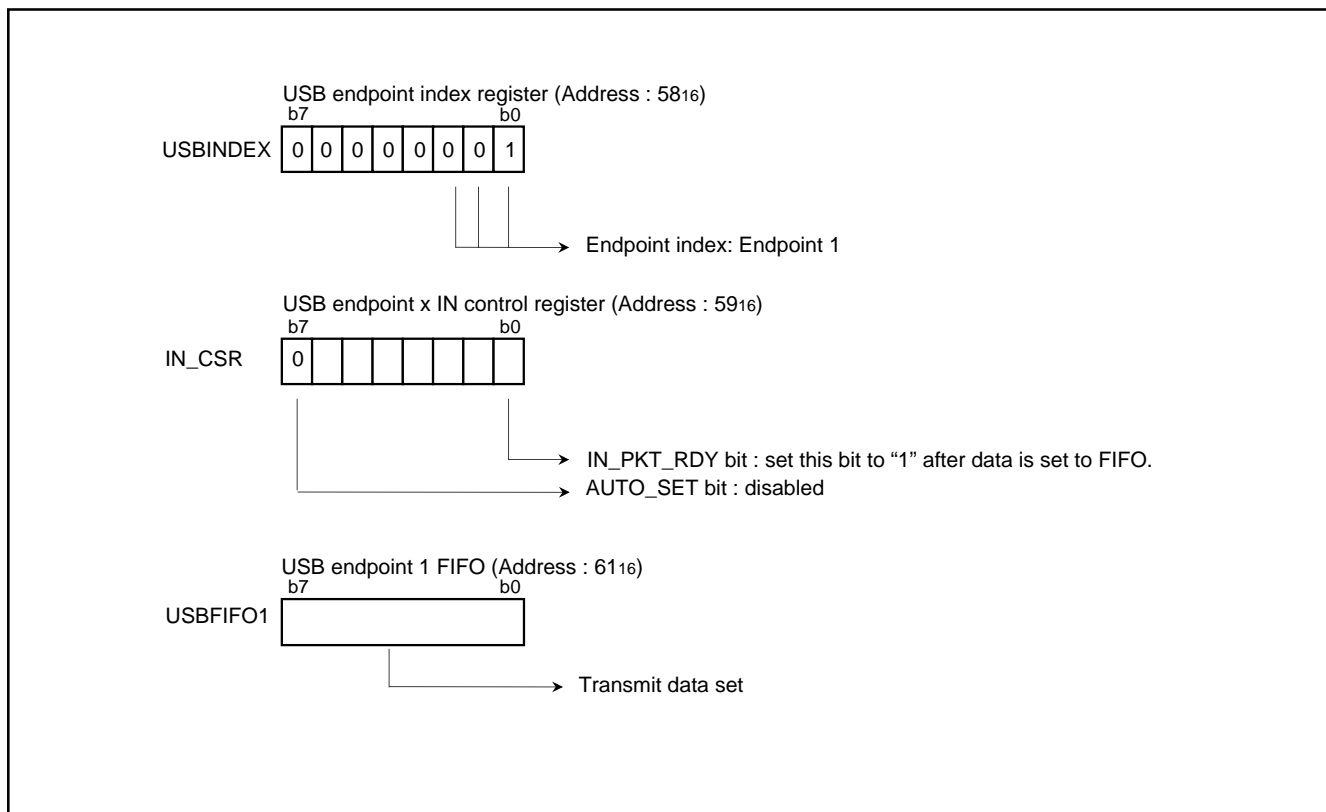
Fig. 2.6.29 Control procedure (3) (endpoint initial setting)

**(2) Application example 2: USB transmit**

**Outline:** Data is transmitted in a USB IN transmit.

**Specification:** - 8 bytes of data (data 0 to 7) is transmitted in the endpoint 1 IN interrupt routine.  
 - AUTO\_SET bit (address 0059<sub>16</sub>, bit 7) = "0" (disabled)  
 - USB endpoint 1 IN max. packet size : 8 bytes

Figure 2.6.30 shows the related register settings, Figure 2.6.31 shows a control procedure example.



**Fig. 2.6.30 Registers setting (1) (USB endpoint 1 transmit)**



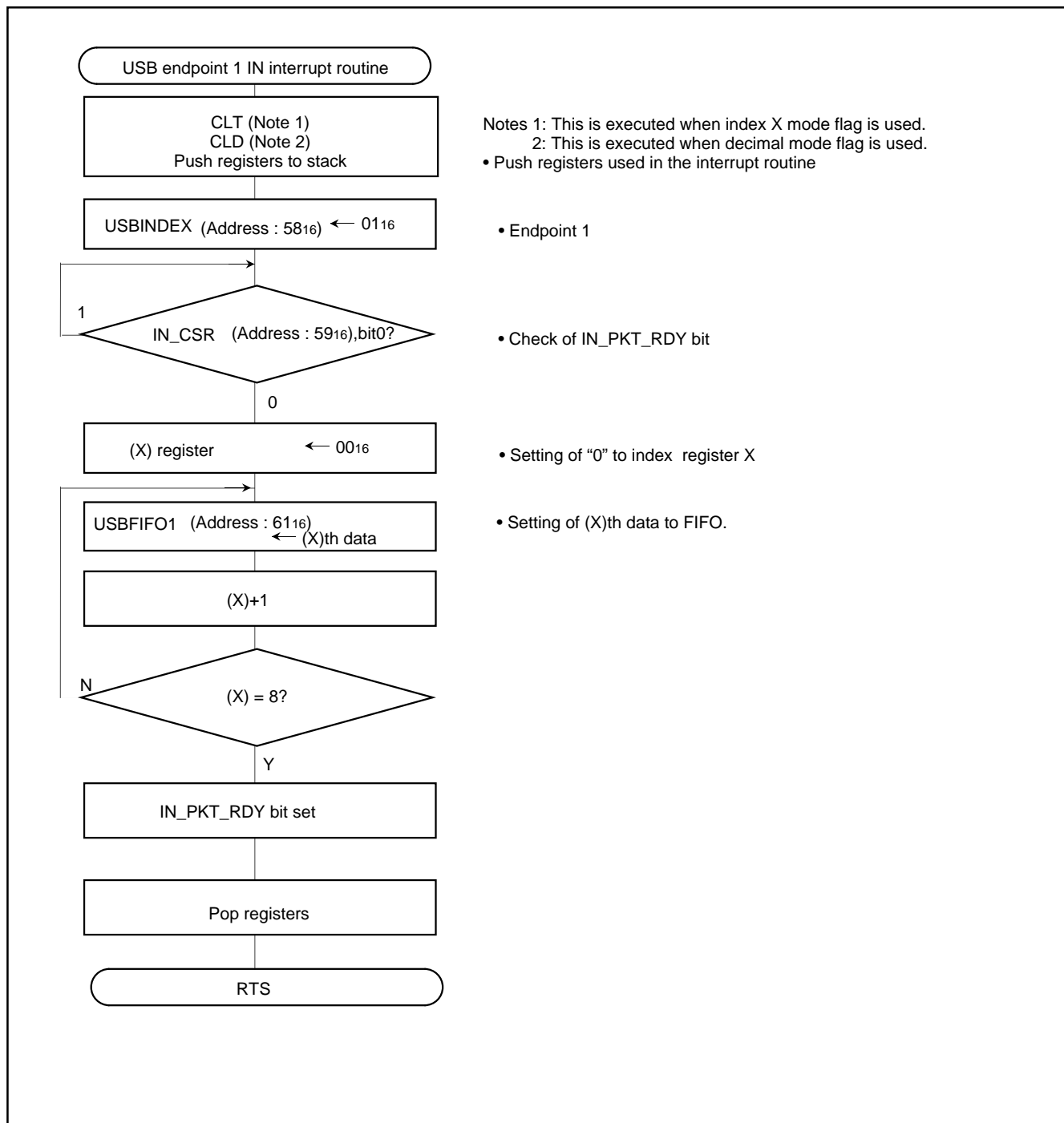
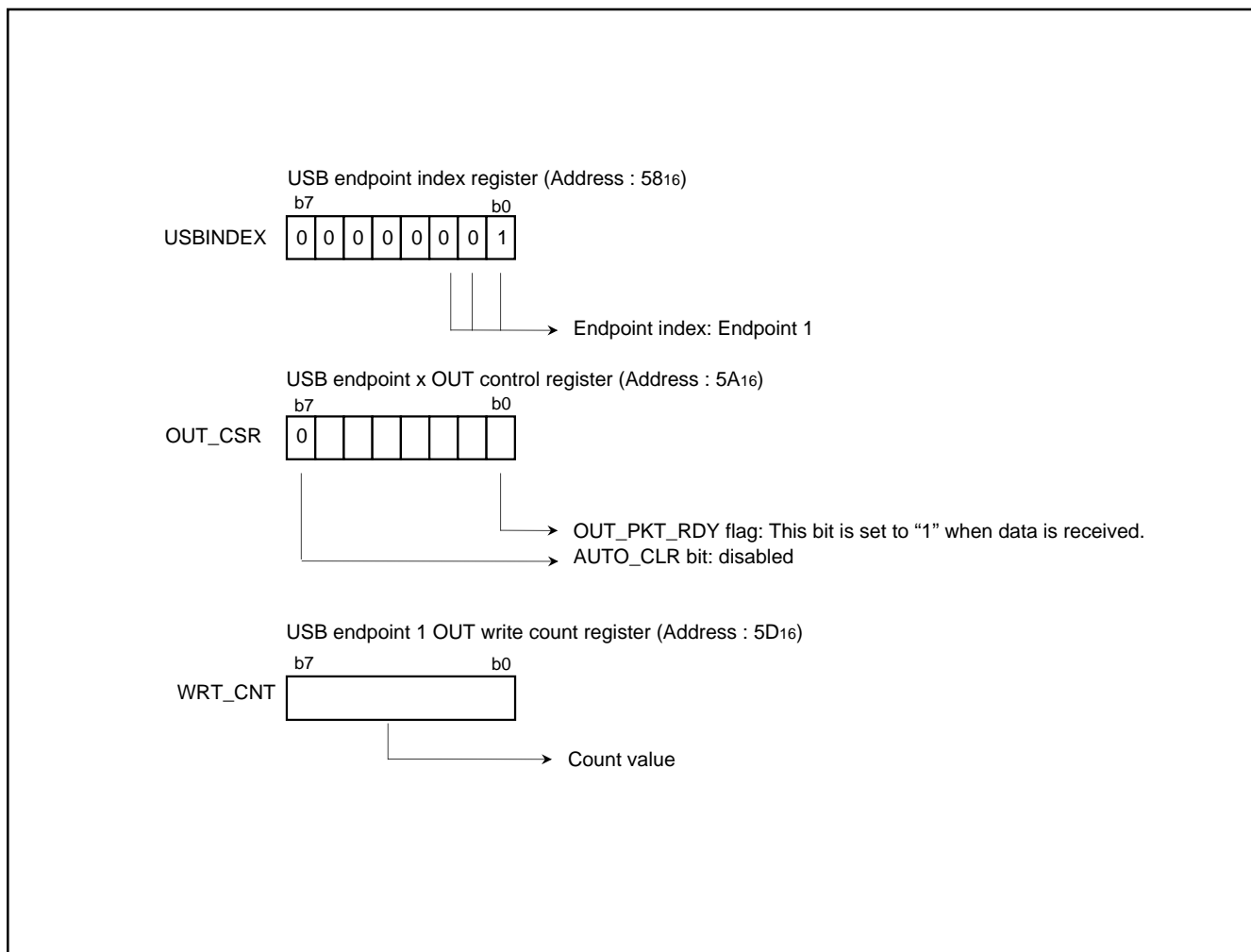


Fig. 2.6.31 Control procedure (USB endpoint 1 IN interrupt routine)

**(3) Application example 3: USB receive****Outline:** Data is received in a USB OUT receive**Specification:** - Data is received in the endpoint 1 OUT interrupt routine.- AUTO\_CLR bit (address 005A<sub>16</sub>, bit 7) = "0" (disabled).

Figure 2.6.32 shows the related register settings, Figure 2.6.33 shows a control procedure example.

**Fig. 2.6.32 Registers setting (USB endpoint 1 OUT receive)**

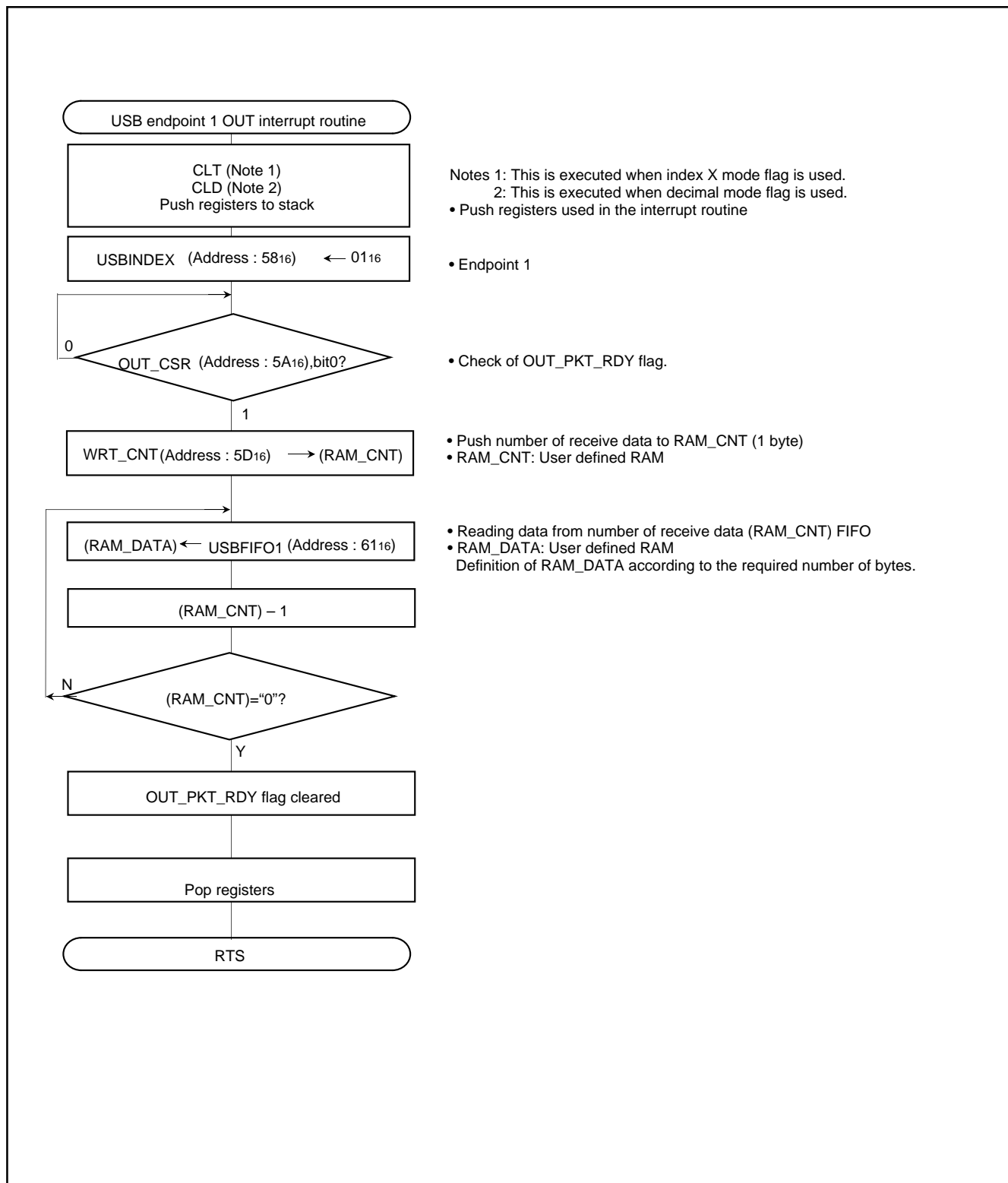


Fig. 2.6.33 Control procedure (USB endpoint 1 OUT interrupt routine)

**(4) Application example 4: Standard device • request (SET\_ADDRESS) receive**

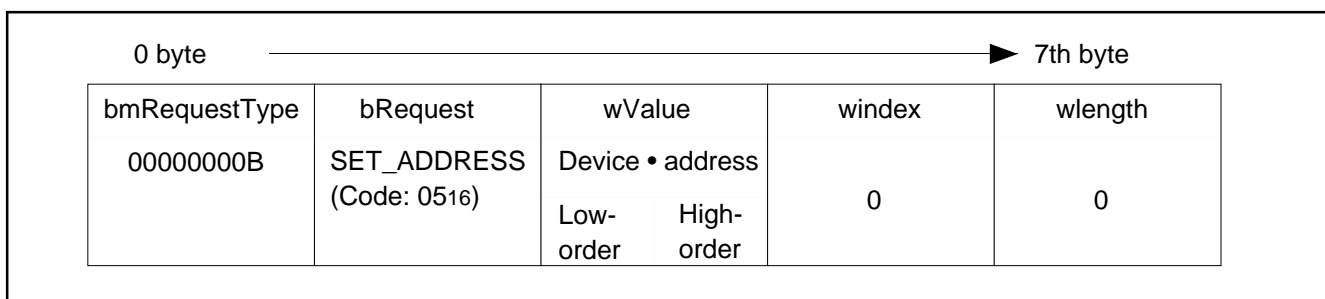
**Outline:** M37643 performs enumeration process after USB cable connection.

The USB endpoint 0 interrupt occurs when data is received successfully from the host CPU in an endpoint 0 control transfer. Whether the data was received in the setup stage or the data stage is not determined by hardware. The user needs to determine the received data by software and execute the process in an interrupt routine accordingly. For details on the data configuration of the standard device request, refer to the Full-Speed USB 2.0 specification.

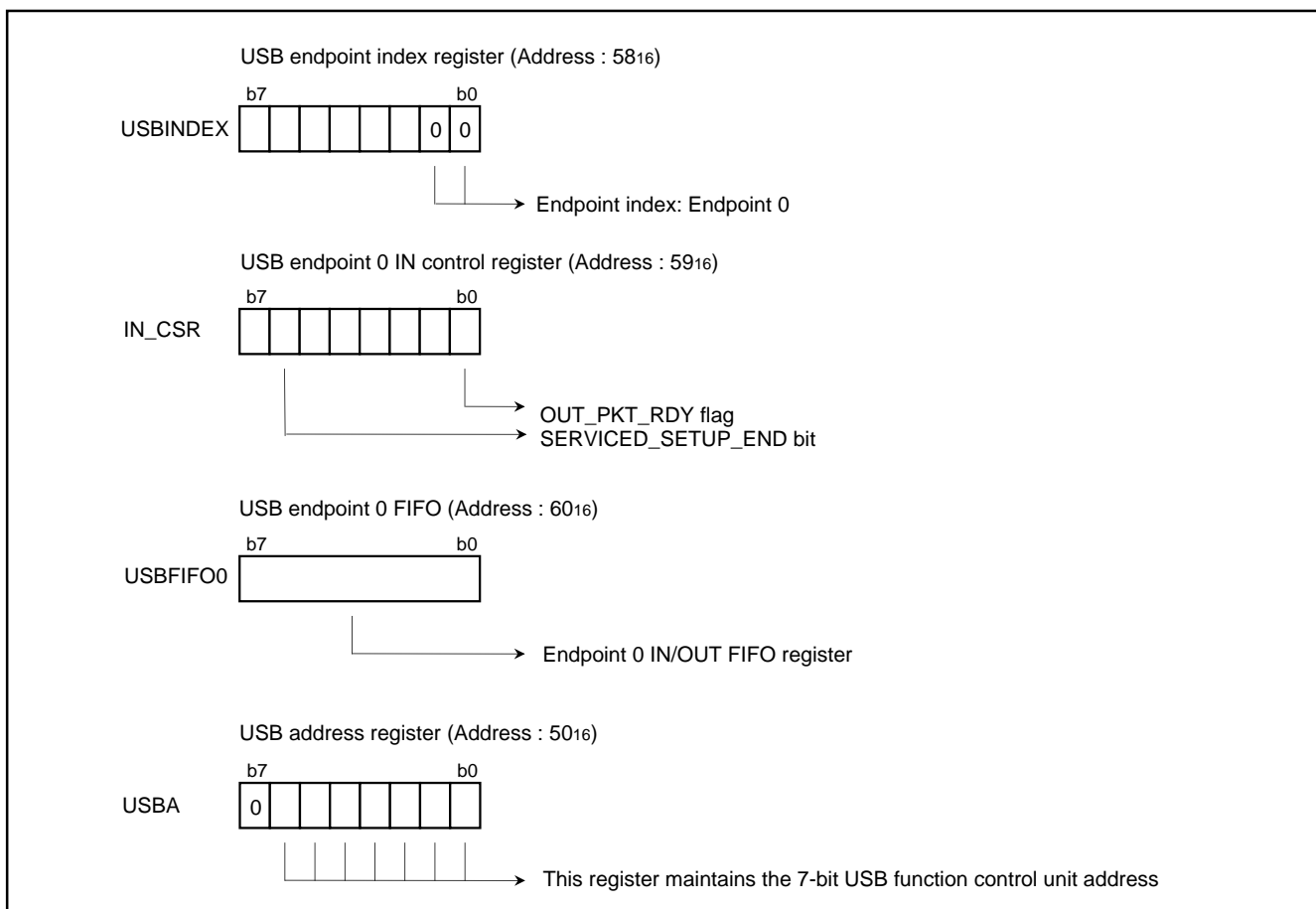
The process at a SET\_ADDRESS receive is shown as a control example.

The SET\_ADDRESS request for the address setting is received from the host CPU, and it is set in the USB address register (address 0050<sub>16</sub>). The USB FCU uses this address for all subsequent device accesses. A standard device request comprises 8 bytes.

Figure 2.6.34 shows the structure of the SET\_ADDRESS request, Figure 2.6.35 shows the related register settings, and Figure 2.6.36 shows a control procedure example.



**Fig. 2.6.34 Structure of SET\_ADDRESS request**



**Fig. 2.6.35 Register setting (processing when SET\_ADDRESS is received)**

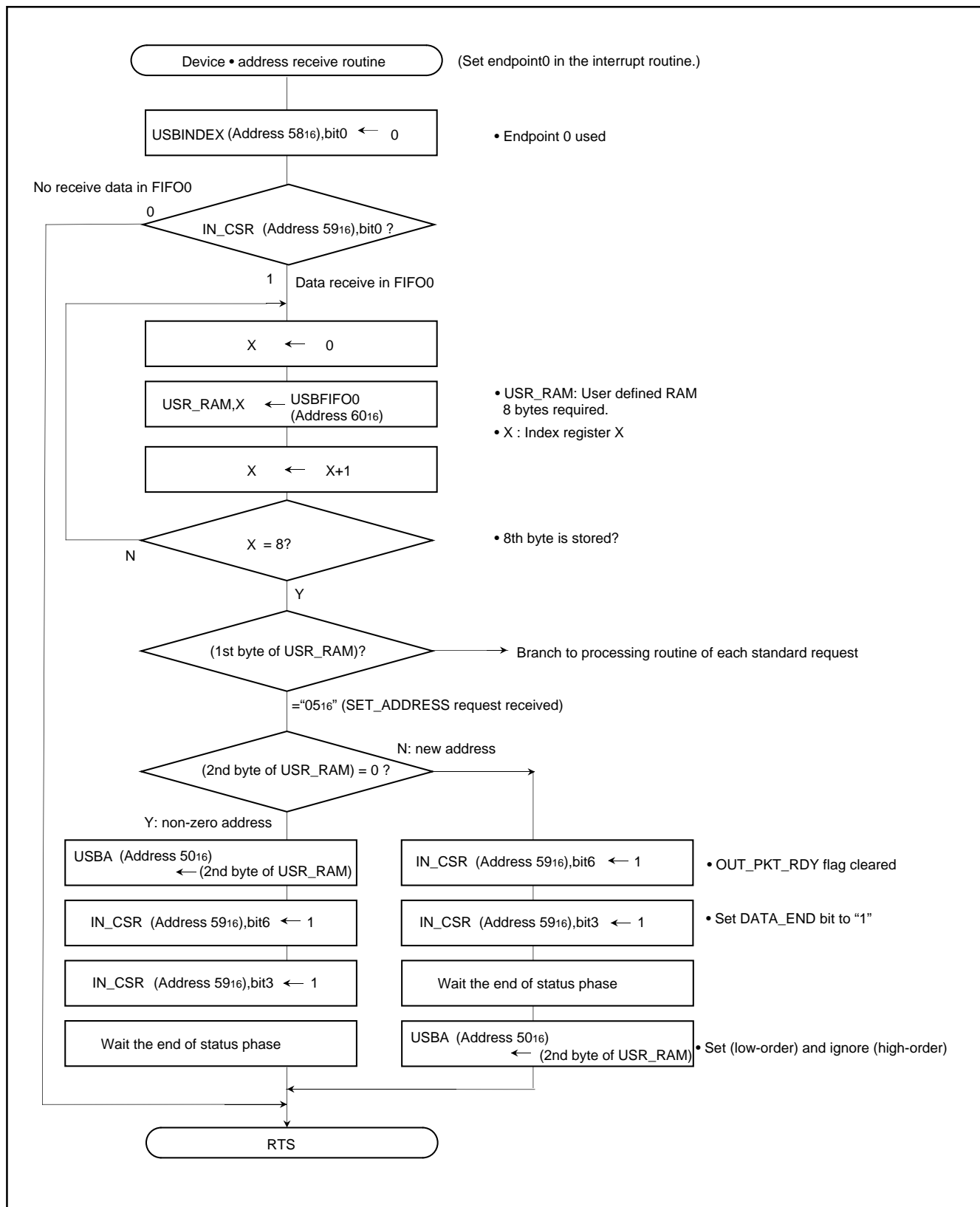


Fig. 2.6.36 Control procedure

**(5) Application example 5: USB function interrupt routine**

**Outline:** The following interrupts are assigned to the same vector.

- USB endpoint 0 interrupt
- USB endpoint x (x=1, 2) IN interrupt
- USB endpoint x (x=1, 2) OUT interrupt
- USB reset interrupt
- USB suspend interrupt
- USB resume interrupt

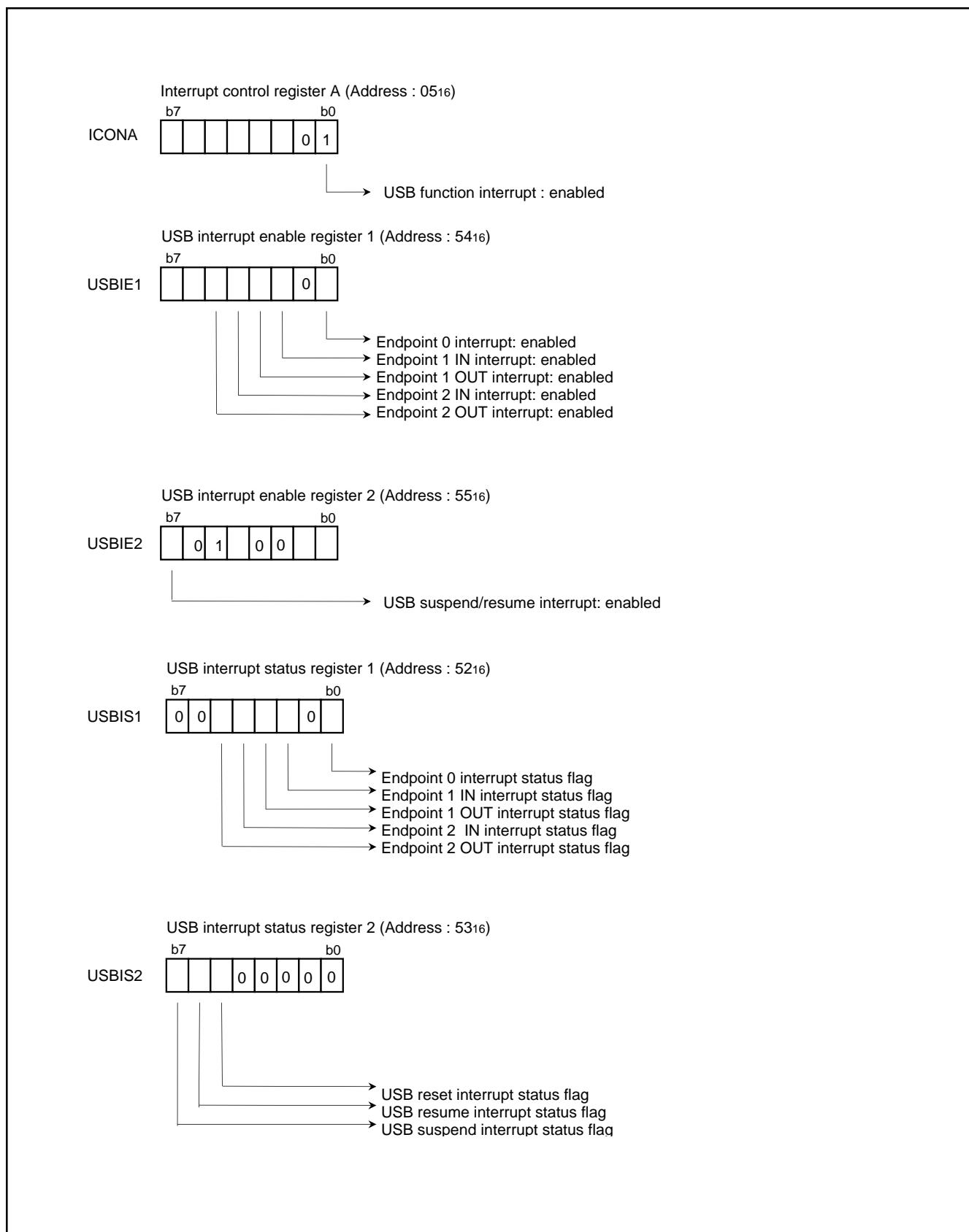
Make all of the following settings in order to enable the interrupts.

- Set USB function interrupt enable bit of interrupt control register A to "1".
- Set the bits corresponding to USB interrupt enable registers 1 and 2 to "1".
- Set the interrupt disable flag (I) to "0".

Each interrupt is determined according to the contents of USB interrupt status registers 1 and 2 in the USB function interrupt routine. When "1", the corresponding process routine is executed.

The USB reset interrupt does not have a specified enable bit and is always in the enabled state.

Figure 2.6.37 shows the related register settings and Figure 2.6.38 shows a control procedure example.



**Fig. 2.6.37 Register setting (USB function interrupt routine)**

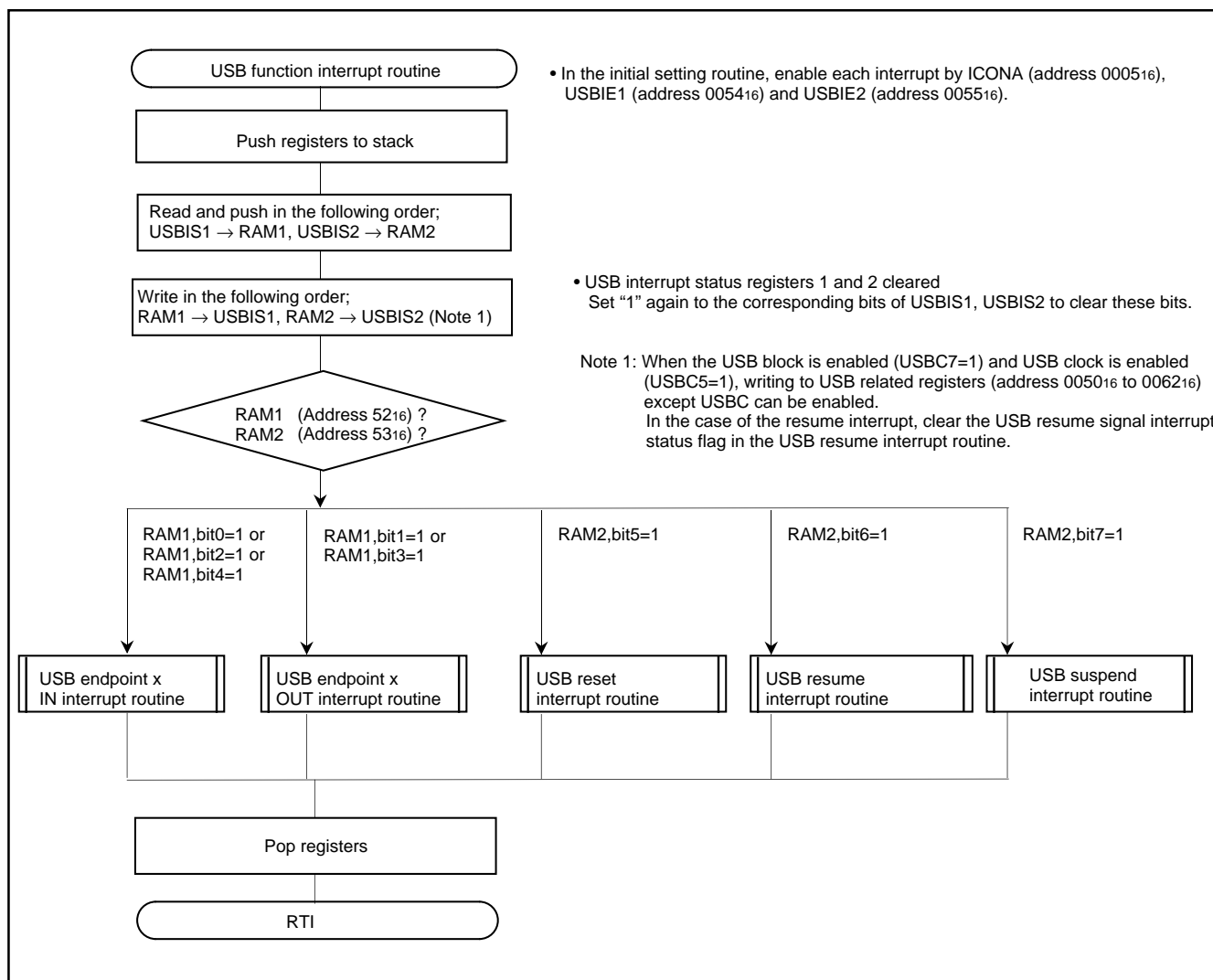


Fig. 2.6.38 Control procedure (USB function interrupt routine)



**(6) Application example 6: USB suspend interrupt**

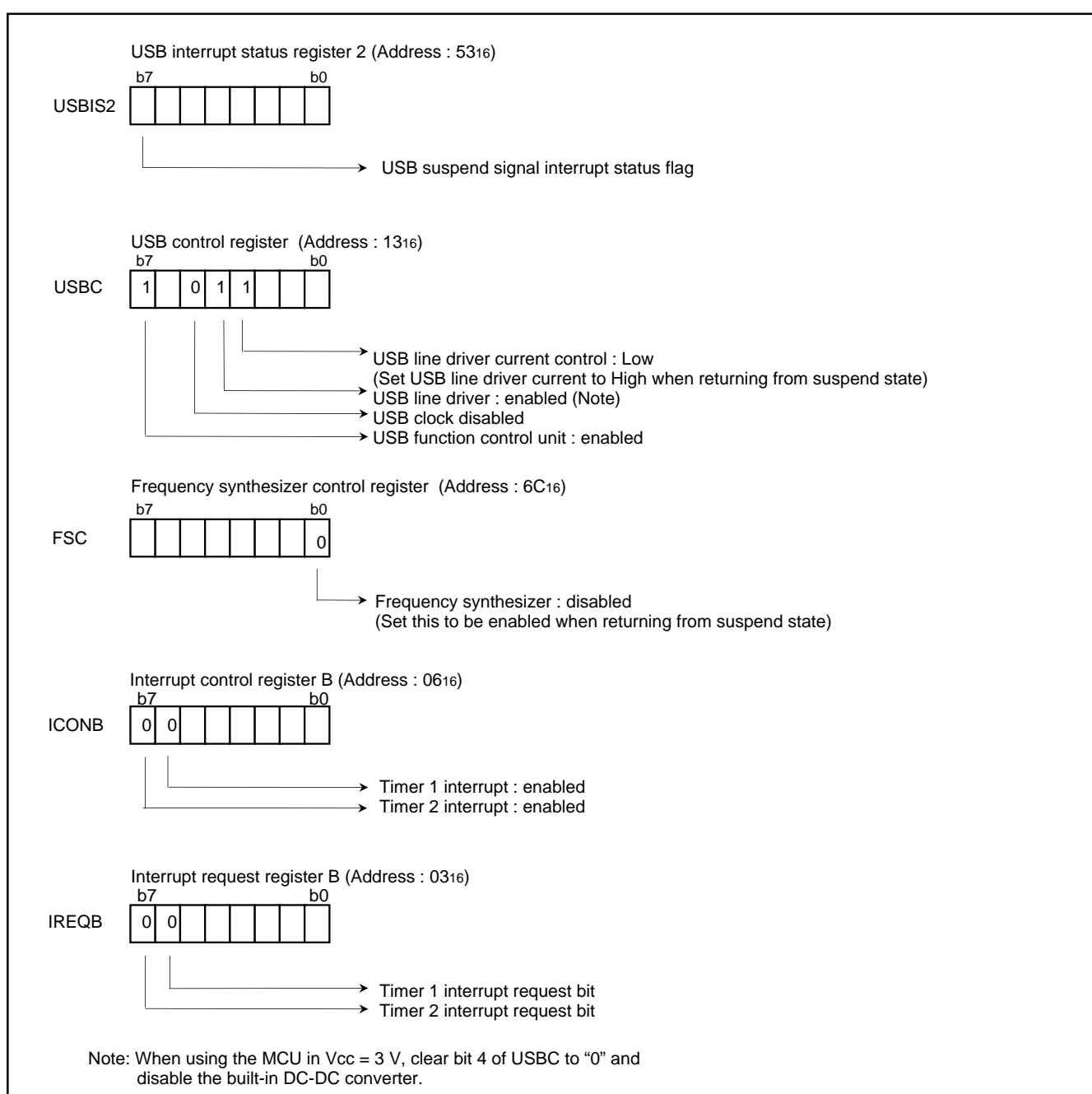
**Outline:** Execute the suspend mode process routine when the USB suspend signal interrupt status flag is “1” in the USB function interrupt routine.

Set the USB function interrupt enable bit and the USB suspend/resume interrupt enable bit to “1” (enabled).

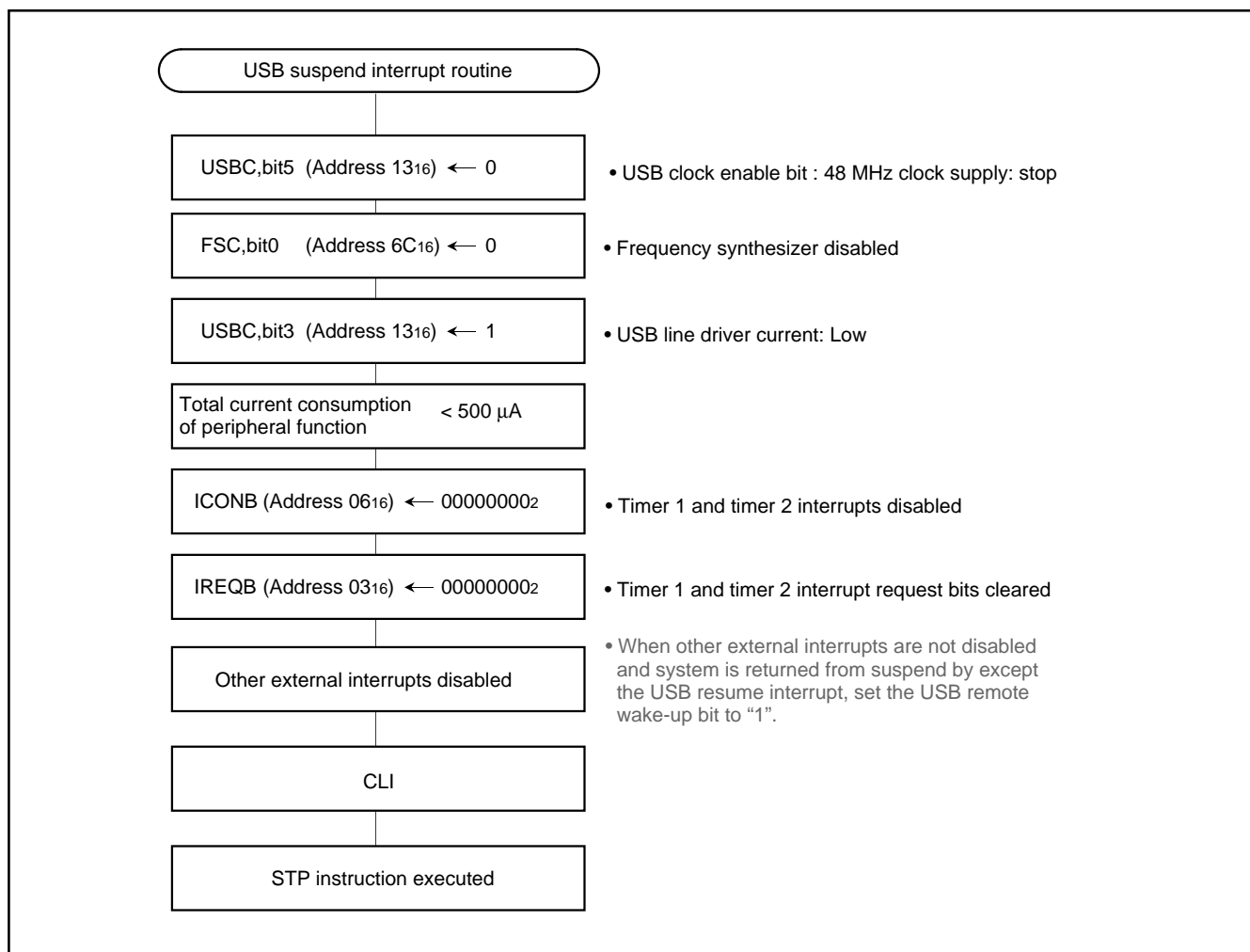
Before executing the STP instruction, clear all bits of USB interrupt status registers 1 and 2.

In addition, when an interrupt process is not used to transition to the suspend mode, be sure to set the USB function interrupt enable bit to “1” and clear the USB function interrupt request bit to “0” before executing the STP instruction.

Figure 2.6.39 shows the related register settings and Figure 2.6.40 shows a control procedure example.



**Fig. 2.6.39 Register setting (USB suspend interrupt)**



**Fig. 2.6.40 Control procedure (USB suspend interrupt routine)**

**(7) Application example 7: USB resume interrupt**

**Outline:** Perform the resume process when the USB resume signal interrupt status flag is “1” in the USB function interrupt routine.

Set the USB function interrupt enable bit to “1” (enabled).

This is an example when the MCU is operating at  $V_{CC} = 5V$ .

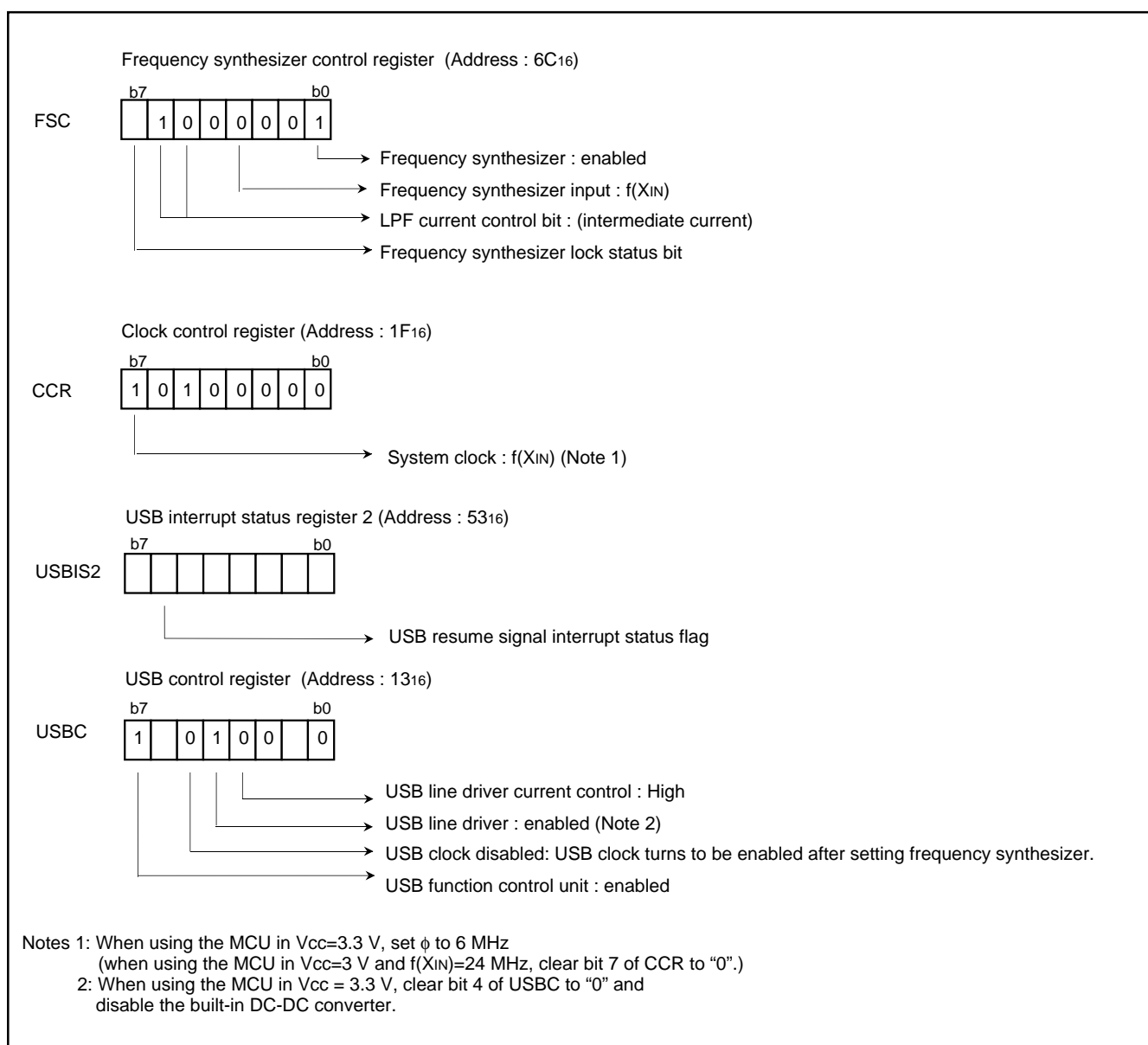
The following settings are required when using  $V_{CC} = 3V$ .

- Set USB line driver supply select bit (USB control register, bit 4) to “0” to disable the DC-DC converter.

- Set  $\phi$  to 6MHz or less.

When  $f(X_{IN}) = 24MHz$ , set bit 7 of the clock control register ( $X_{IN}$  divider select bit) to “0”.

Figure 2.6.41 shows the related register settings and Figure 2.6.42 shows a control procedure example.



**Fig. 2.6.41 Register setting (USB resume interrupt)**

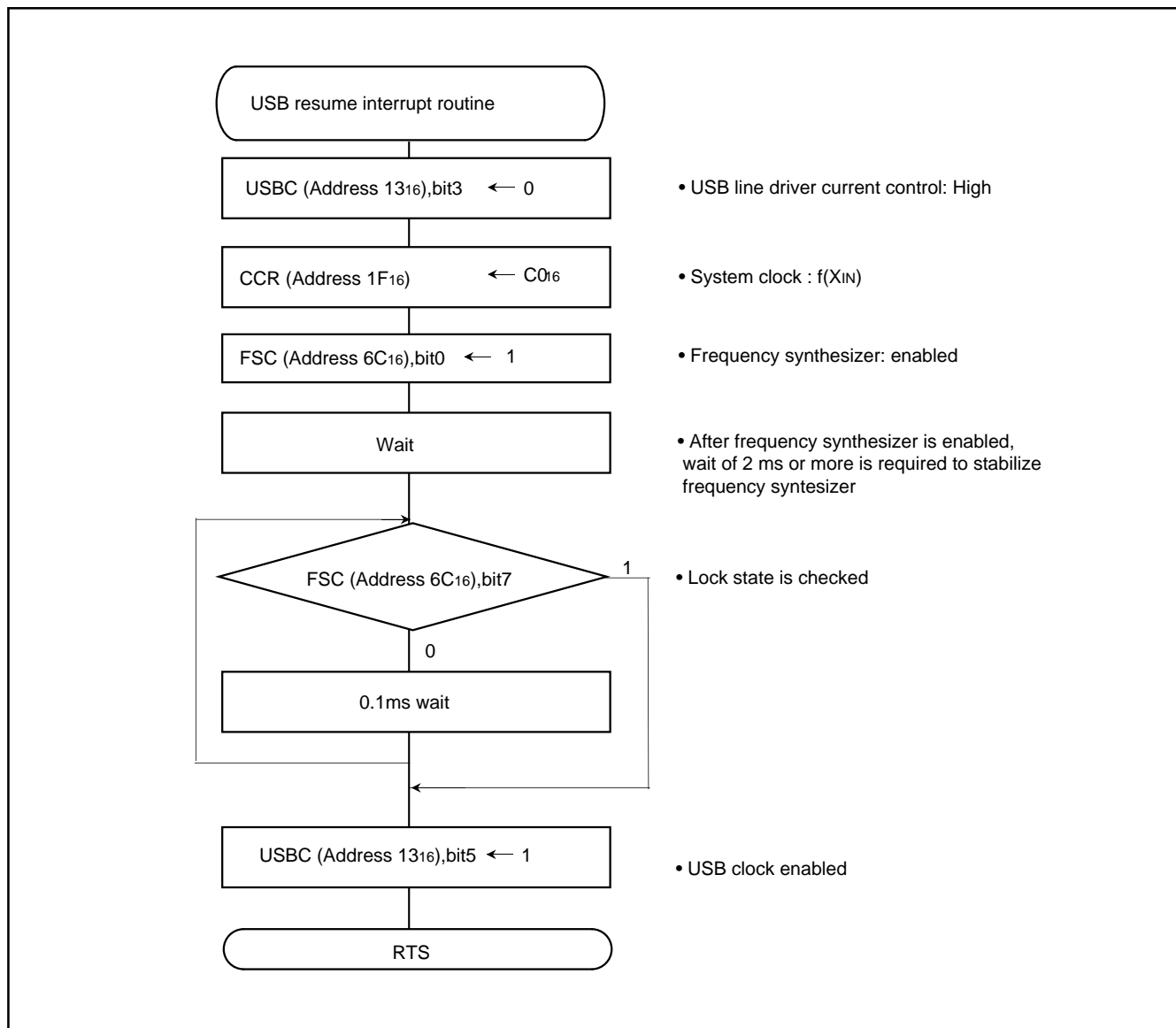


Fig. 2.6.42 Control procedure (USB resume interrupt routine)

### 2.6.8 Connection with other functions

The 7643 Group also has DMAC and UART connections for USB transfers, enabling high-volume data transfers.

#### (1) Application example 1: Data packet transfer from USB FIFO to UART (without DMA)

**Outline:** Data is transmitted from USB FIFO to host CPU using UART.

**Specification:** - USB endpoint 1 OUT bulk transfer is used.

- USB endpoint 1 OUT interrupt is used.
- USB endpoint 1 OUT packet size: 64 bytes
- UART transmit is used.
- UART transmit interrupt (when transmit shift operation is completed) is used.
- Transfer bit rate: 9600bps ( $\phi = 12\text{MHz}$  divided by 1248)
- Data format: 1ST-8DATA-1SP
- Parity bit is disabled
- Re-transmit is executed if transfer error occurs (re-transmit in byte unit)
- P4<sub>1</sub>/INT0 pin is used to detect the re-transmit request (host CPU sets P4<sub>1</sub>/INT0 pin to "L" when UART receive error occurs)
- INT0 interrupt request flag (falling edge) to detect the re-transmit request is used.
- P4<sub>0</sub> to inform host CPU of USB packet transmit is used.
- CTS function is used.

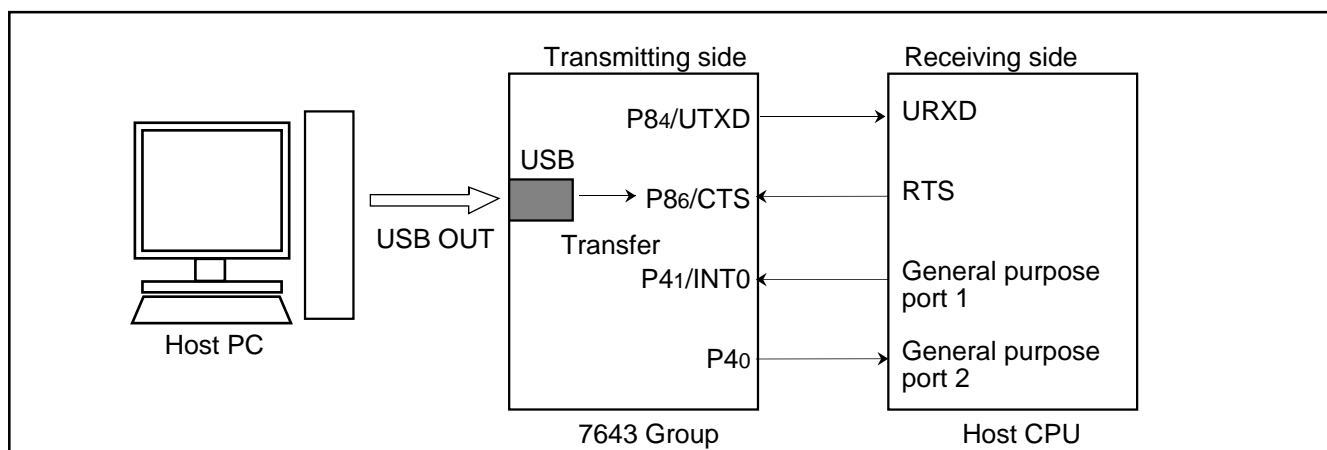
After USB data is received from the host PC, the 7643 MCU stores the value of the USB endpoint 1 OUT write count register (number of received bytes) in Storage RAM 1 by the USB endpoint 1 OUT interrupt. Then, sets P4<sub>0</sub> to "L", the UART transmit interrupt to be enabled, and starts the UART transmit.

As soon as the UART transmit is enabled, the first UART transmit interrupt occurs. Therefore, the 7643 MCU writes one byte of the received USB FIFO data to both Storage RAM2 and UART transfer buffer register 1, and returns from the interrupt. Next, the host CPU sets the CTS pin to "L" and reads the data in the UART transmit/receive shift register. At this time, since another UART transfer interrupt occurs, the same process is repeated until the USB FIFO is empty.

On the other hand, when the host CPU requests a re-transmit by a UART receive error etc., it is programmed to set the P4<sub>1</sub>/INT0 pin to "L". In the 7643 Group, the data in Re-transmit RAM 2 is written to the UART transmit/receive buffer register if the INT0 interrupt request flag is "1" in the UART transmit interrupt routine. By this action, the previous data is resent to the host CPU.

As described above, when all data in the USB FIFO are transmitted to the host CPU, P4<sub>0</sub> is set to "H", the UART transmit interrupt is disabled, and the OUT\_PKT\_RDY flag of USB endpoint 1 OUT is cleared. This makes USB endpoint 1 ready to receive the next USB data packet.

Figure 2.6.43 shows the connection diagram, Figures 2.6.44 to 2.6.46 show related register settings, and Figures 2.6.47 and 2.6.48 show control procedure examples.



**Fig. 2.6.43 Connection diagram**

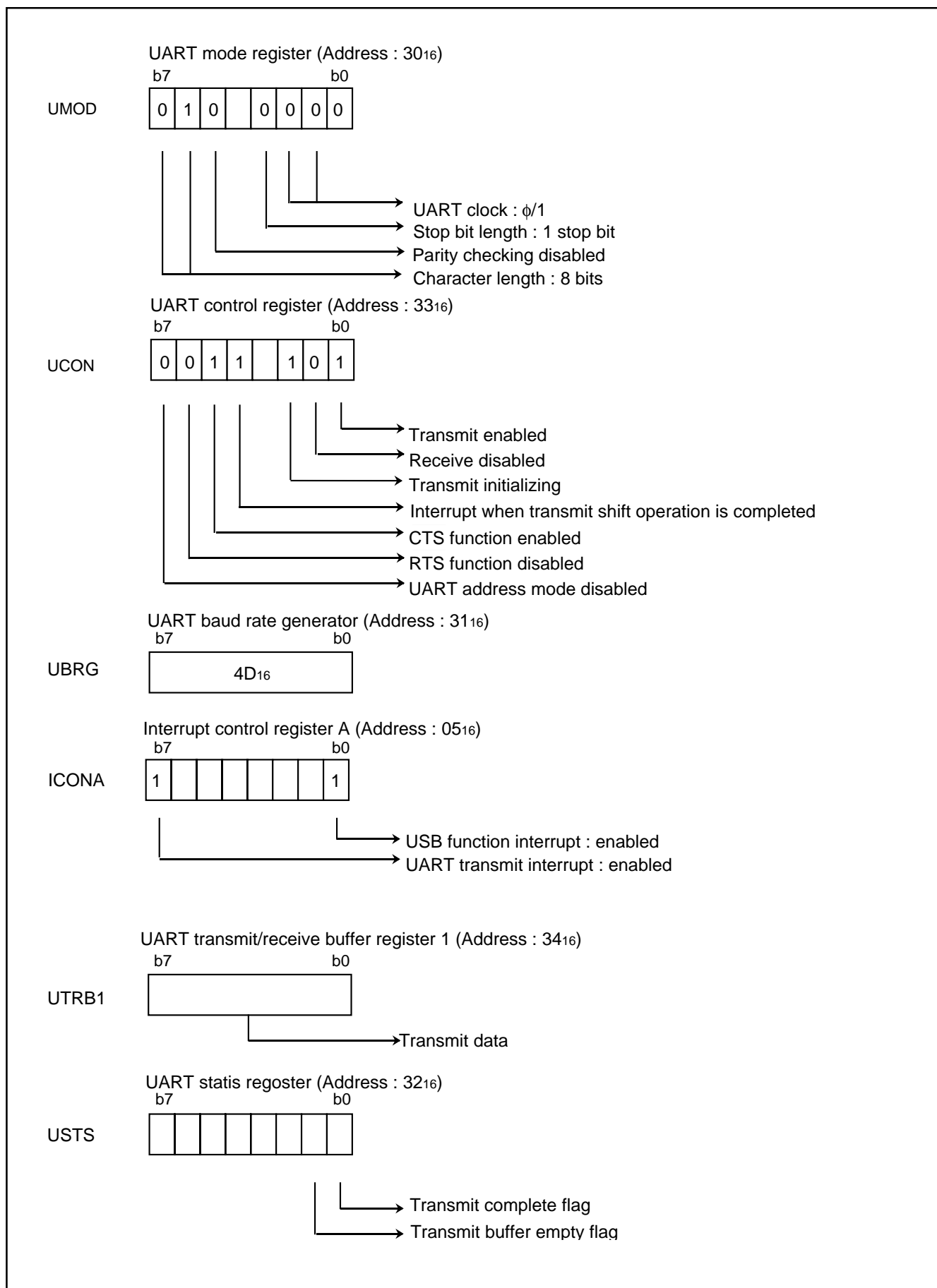


Fig. 2.6.44 Register setting (1)

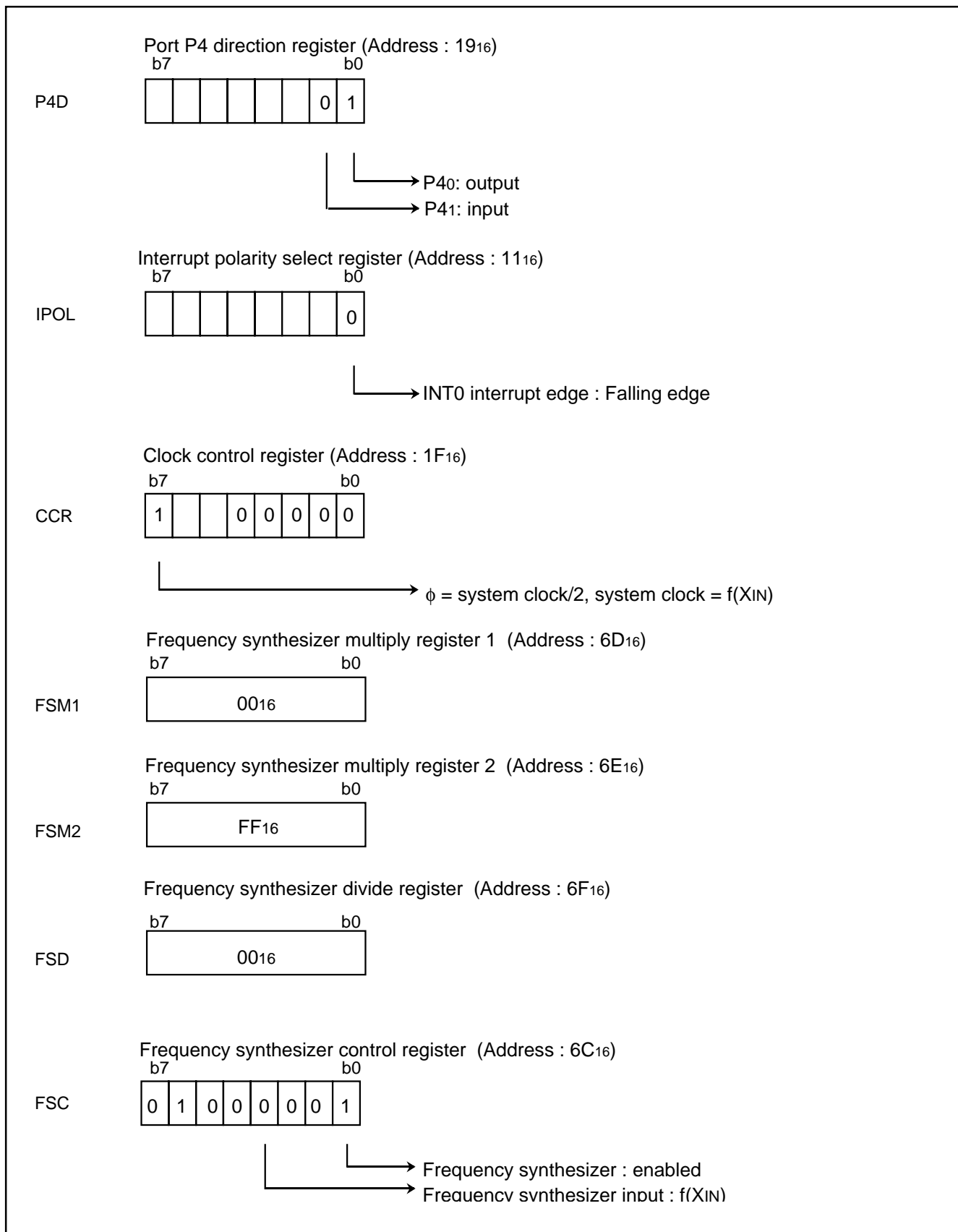


Fig. 2.6.45 Register setting (2)

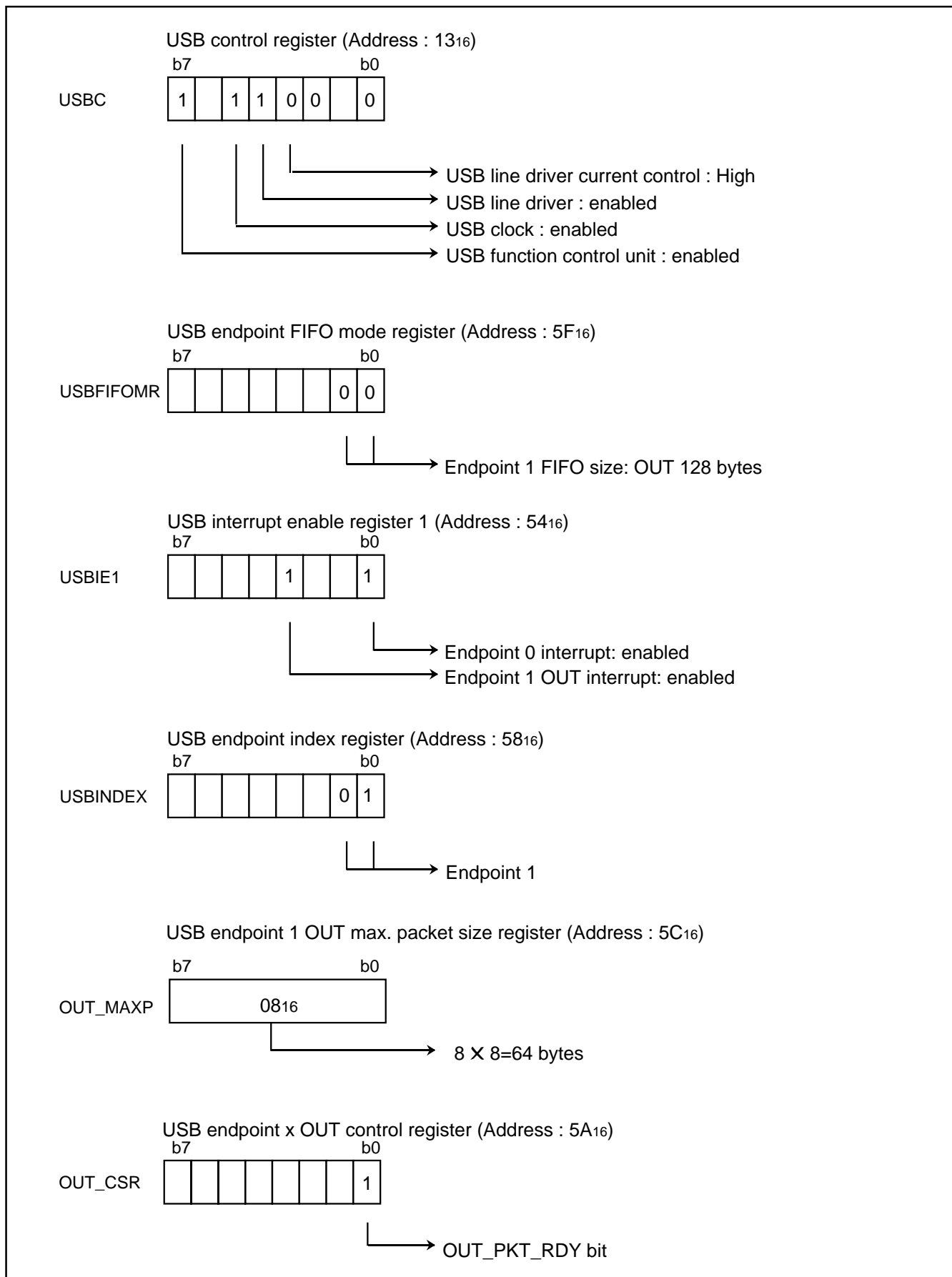


Fig. 2.6.46 Register setting (3)



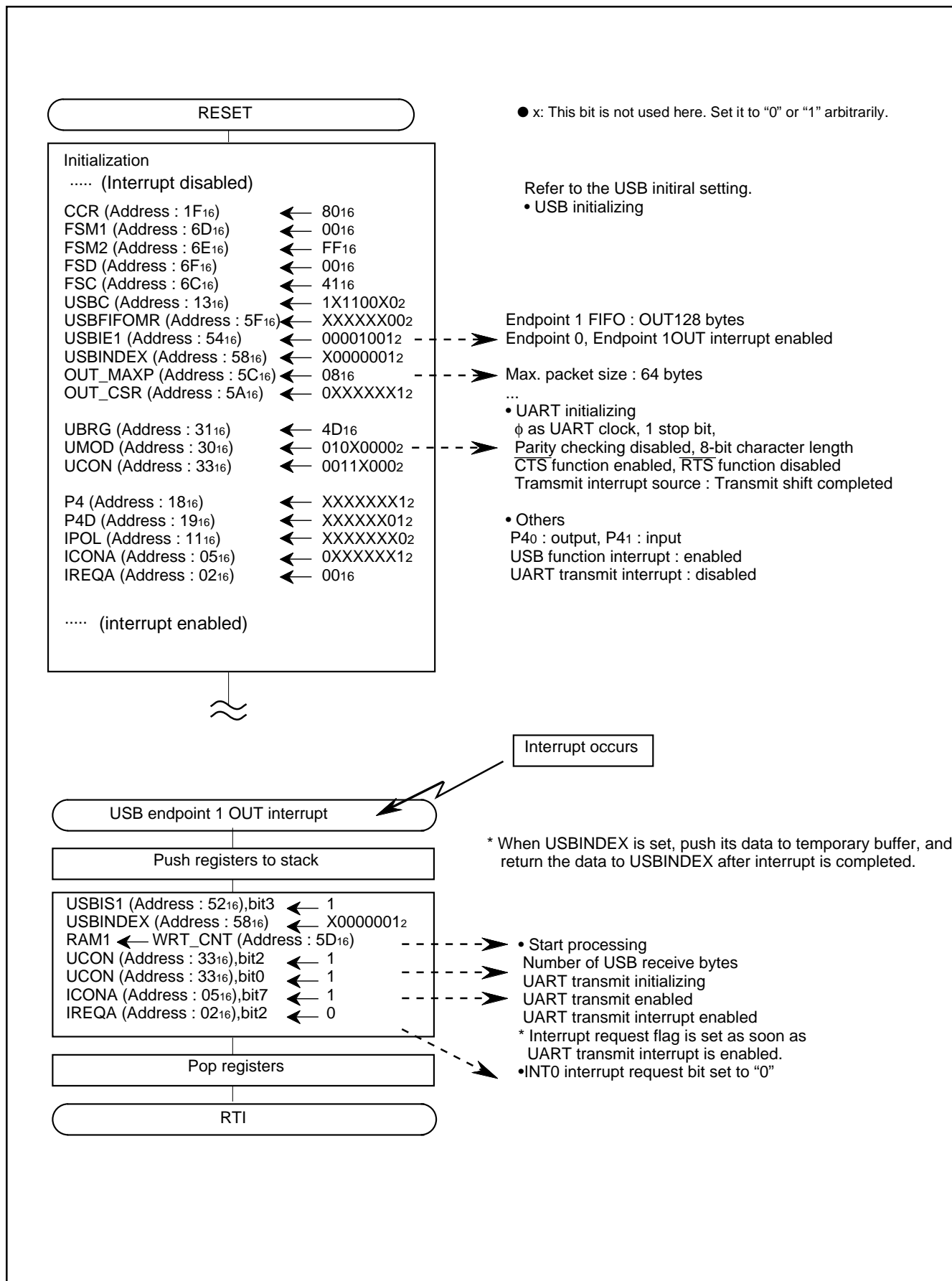


Fig. 2.6.47 Control procedure (1)

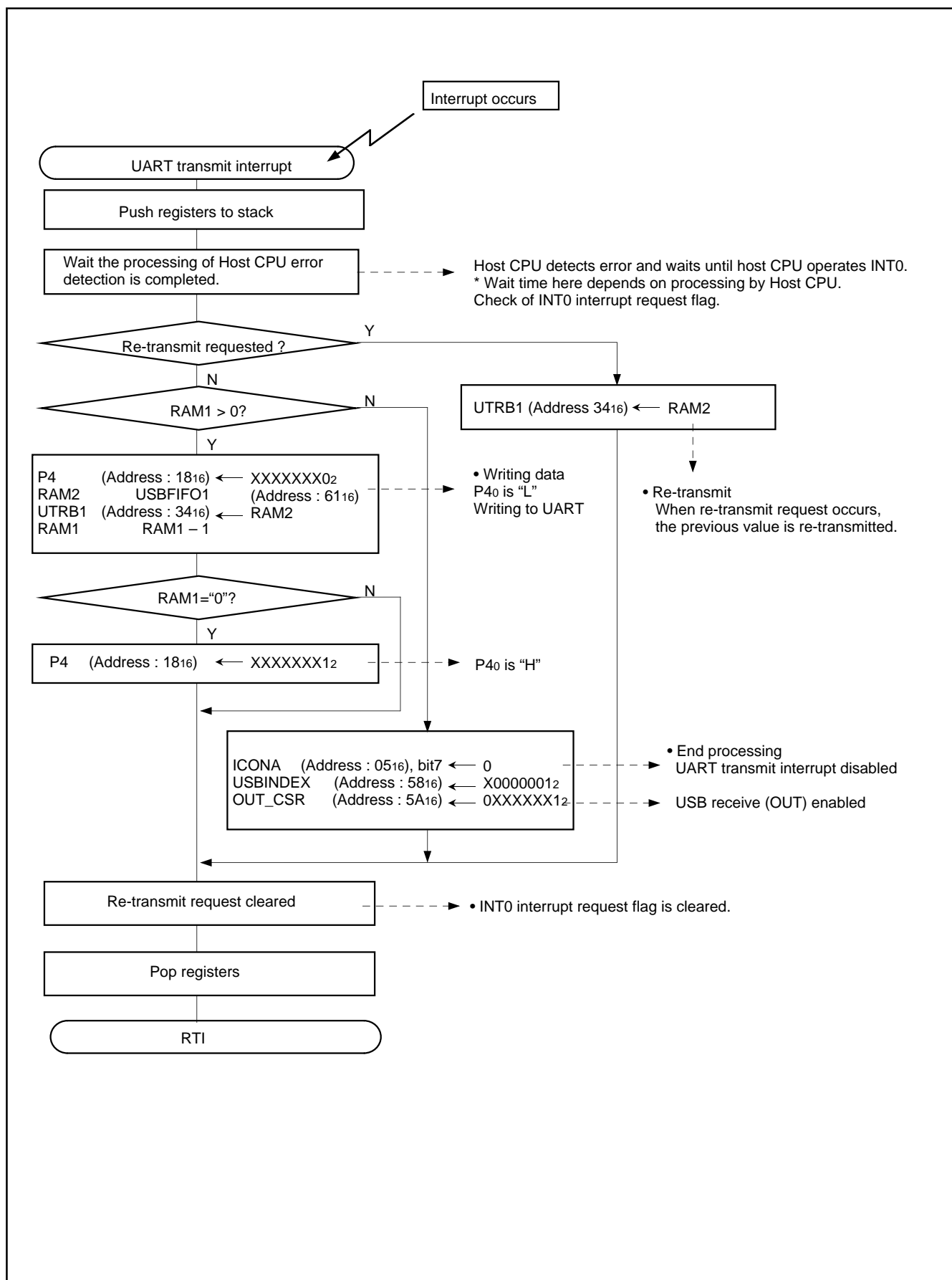


Fig. 2.6.48 Control procedure (2)

**(2) Application example 2: Data packet transfer from UART to USB FIFO (without DMA)**

**Outline:** The data received from host CPU through UART to the USB FIFO is written, and then it is transmitted to host PC.

**Specification:** - Use USB endpoint 1 IN bulk transfer is used.

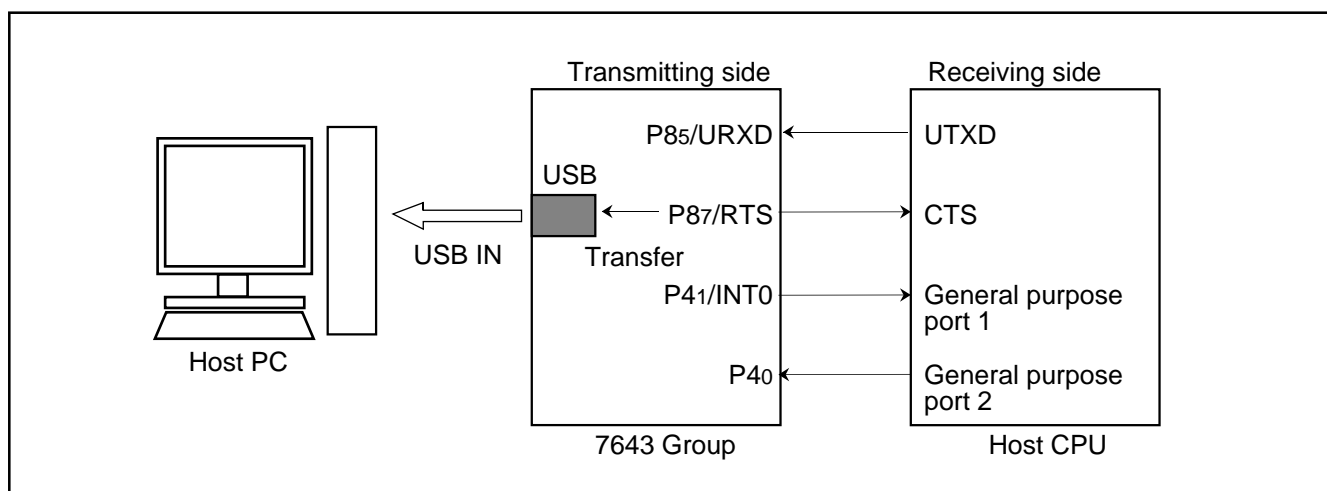
- USB endpoint 1 IN interrupt is used.
- USB endpoint one packet size: 64 bytes
- UART receive is used.
- UART receive buffer full interrupt is used.
- UART receive summing error interrupt is used.
- Transfer bit rate: 9600bps ( $\phi = 12\text{MHz}$  divided by 1248)
- Data format: 1ST-8DATA-1SP
- Parity bit is disabled.
- Re-transmit is executed if transfer error occurs (re-transmit in byte unit)
- P4<sub>0</sub> pin to detect packet transmit from host CPU is used. (the host CPU keeps P4<sub>0</sub> at "L" during a packet transfer, at "H" at all other times)
- RTS function is used.

When the USB FIFO data is transmitted to the host PC, in the 7643 Group, a USB endpoint 1 interrupt occurs. At this point, the 7643 MCU enables the UART receive (UART receive buffer full interrupt, UART receive summing error interrupt).

After the UART function is received from the host PC, in the 7643 Group, a UART receive buffer full interrupt occurs; the UART transmit/receive buffer register is read and the received data is written to the USB FIFO. The RTS pin is set to "L", indicating that the RTS pin is ready to receive the next data. If Port P4<sub>0</sub> is "H" at this point, that receive of one-packet data (64 bytes or more) has been completed is recognized, and sets the IN\_PKT\_RD flag USB endpoint 1 IN. This makes the data in the USB endpoint 1 FIFO ready for transmit.

On the other hand, if a UART receive summing error interrupt, not the UART receive buffer interrupt, occurs, both the UART transmit/receive buffer register and the UART status register is read (this clears the error flag), and an "L" pulse is generated to P4<sub>1</sub> (as a result from this operation, the re-transmit the previous data from the host CPU is realized). Note that the data received when an error occurs will be erased.

Figure 2.6.49 shows the connection diagram, Figures 2.6.50 to 2.6.53 show related register settings, and Figures 2.6.54 and 2.6.55 show control procedure examples.



**Fig. 2.6.49 Connection diagram**

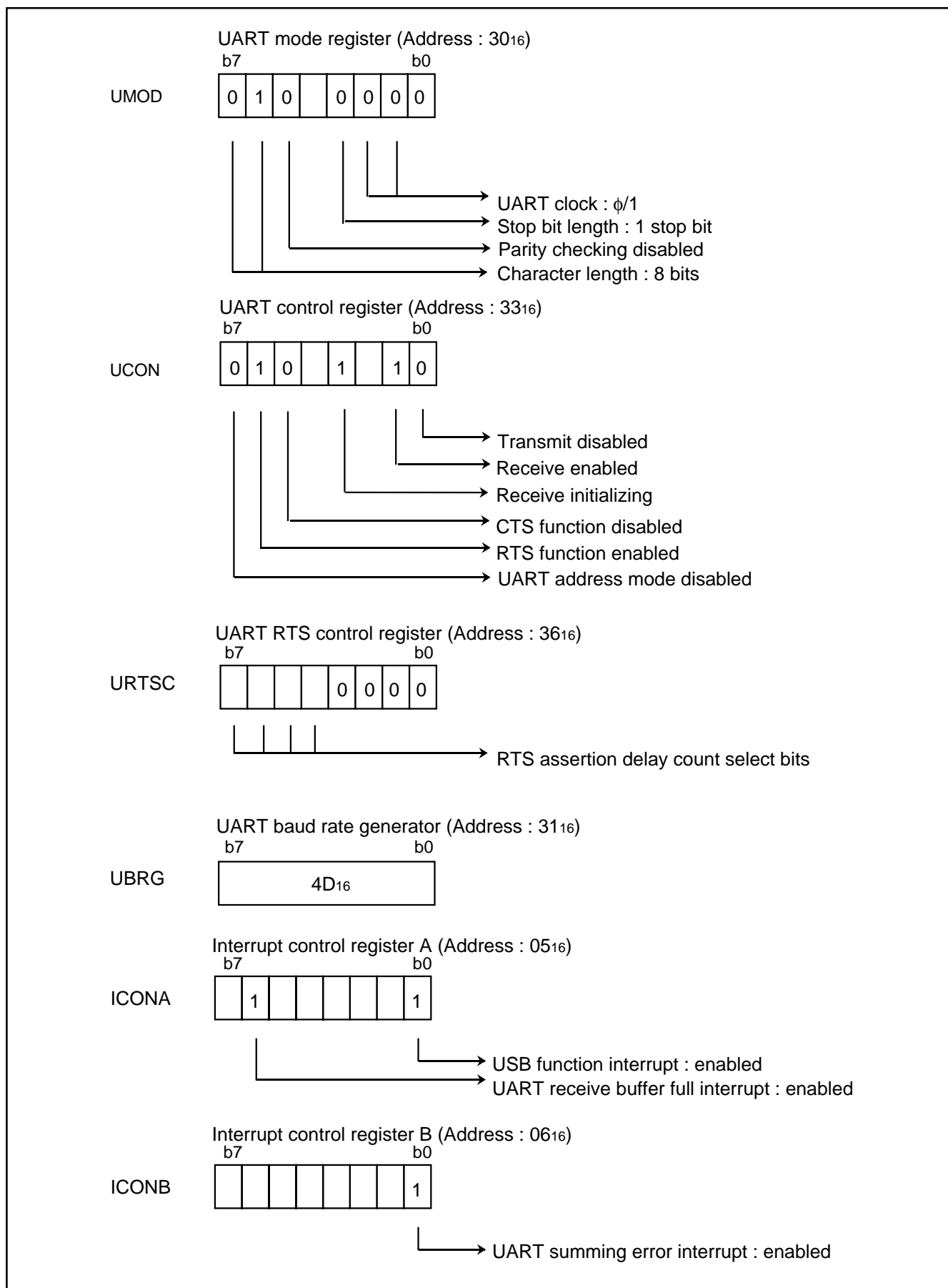


Fig. 2.6.50 Register setting (1)

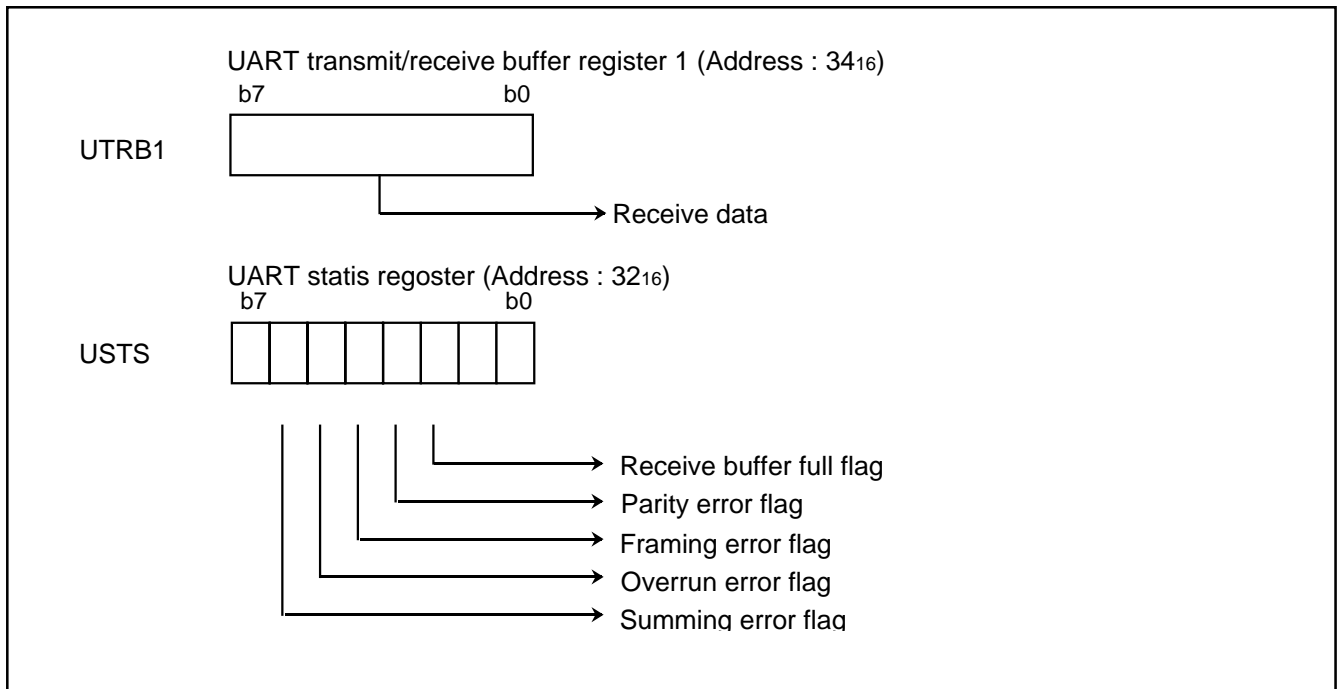


Fig. 2.6.51 Register setting (2)

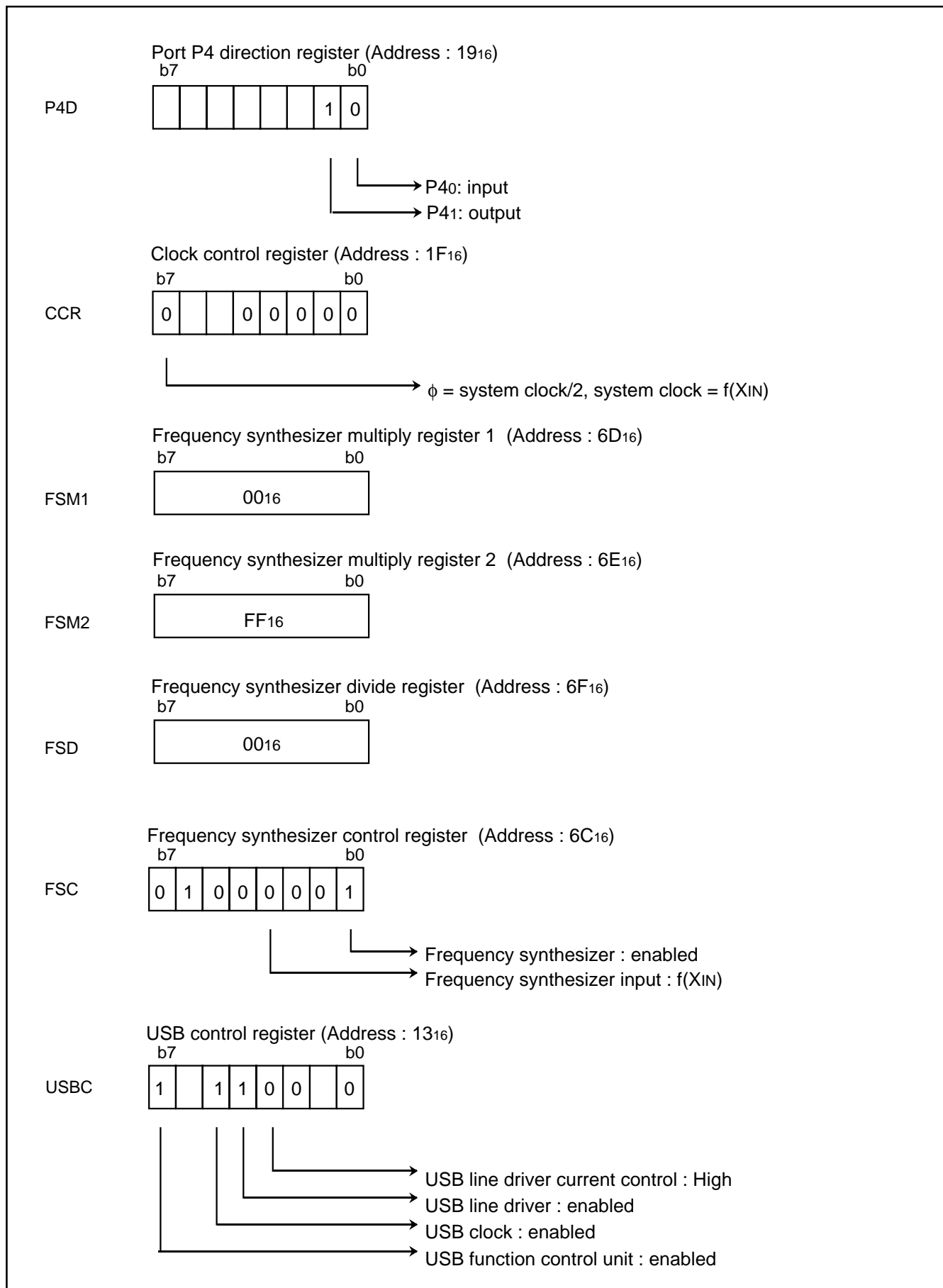


Fig. 2.6.52 Register setting (3)

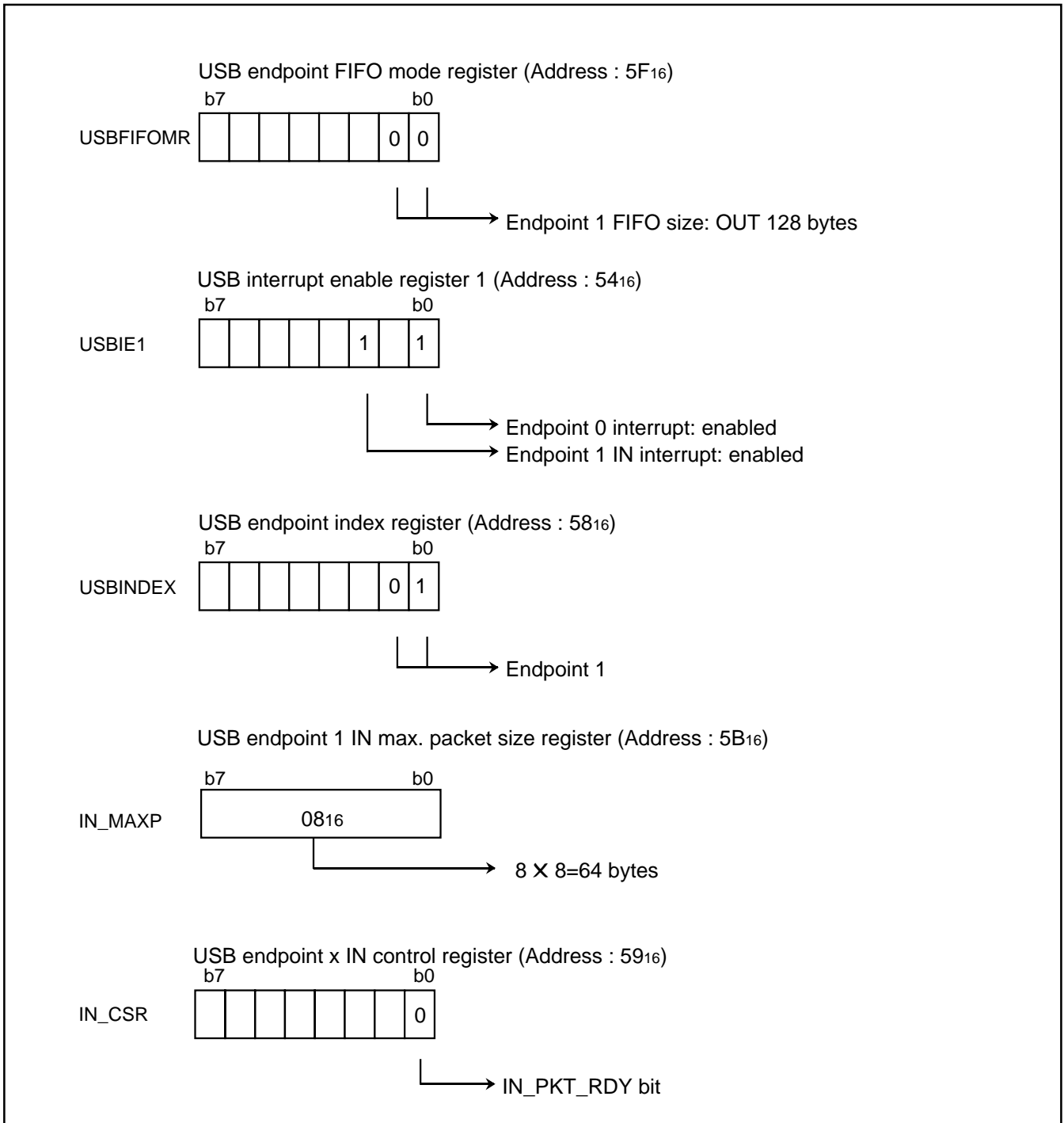


Fig. 2.6.53 Register setting (4)

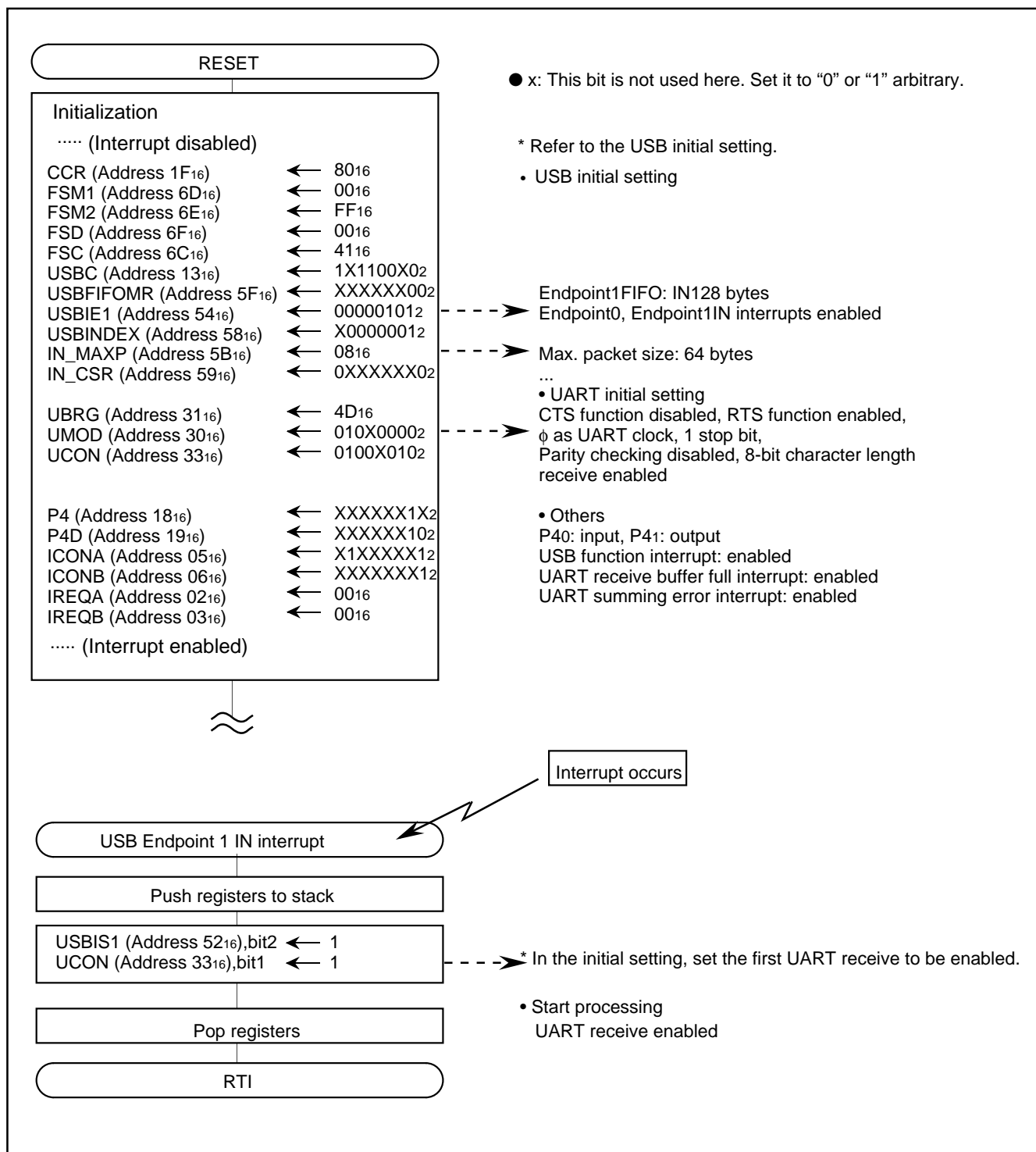


Fig. 2.6.54 Control procedure (1)



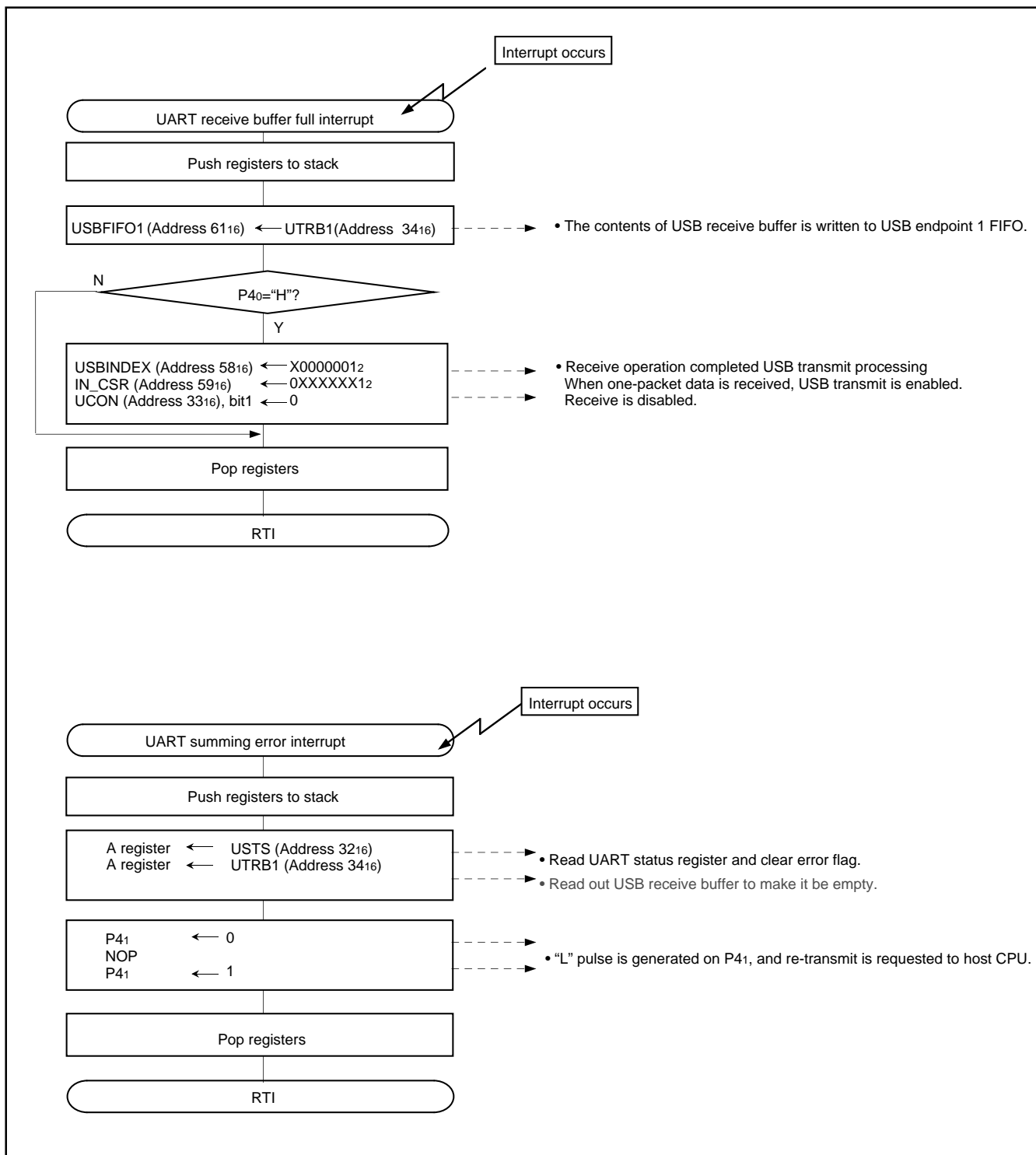
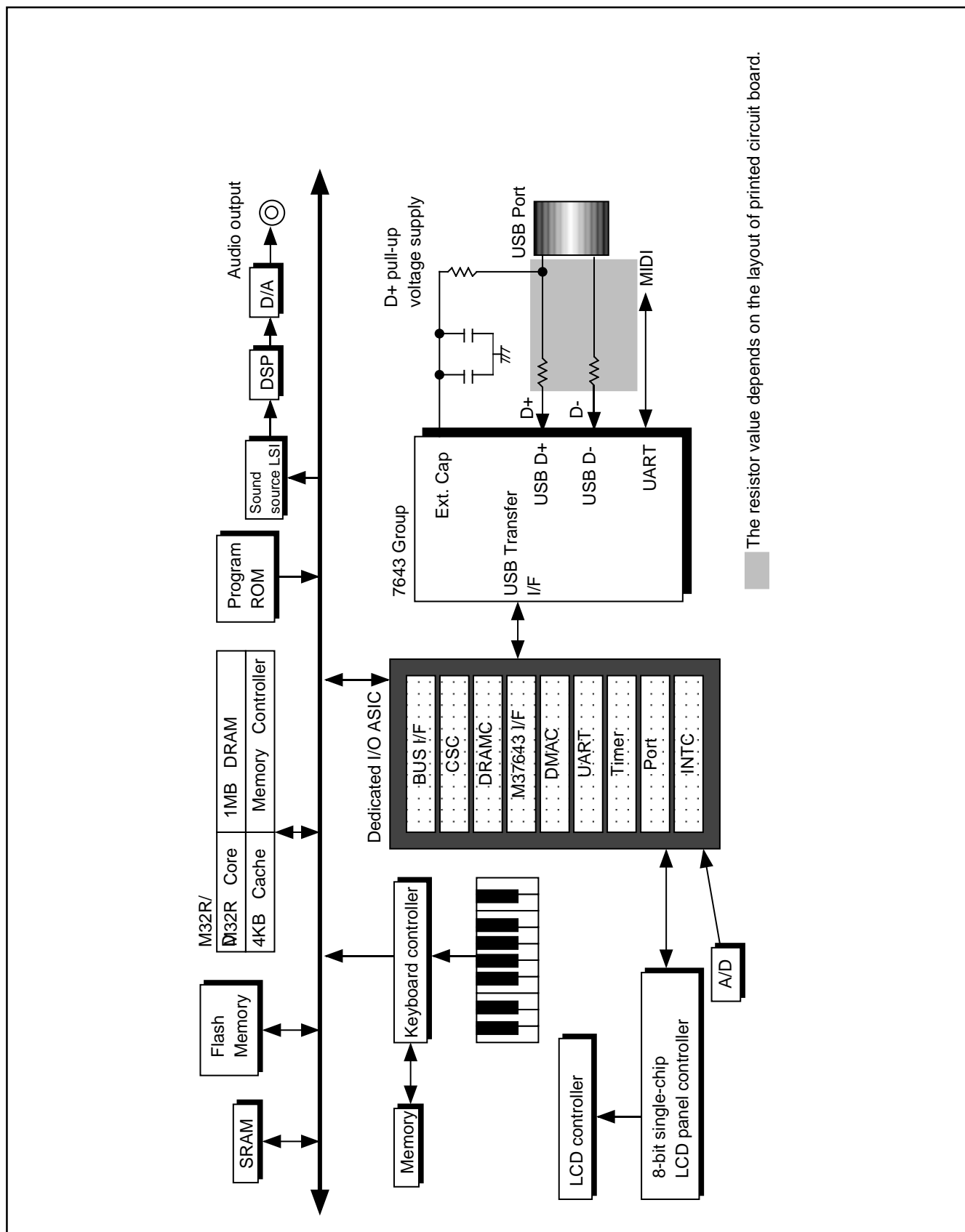


Fig. 2.6.55 Control procedure (2)

2.6.9 Application circuit example



The resistor value depends on the layout of printed circuit board.

Fig. 2.6.56 Electronic instrument application example

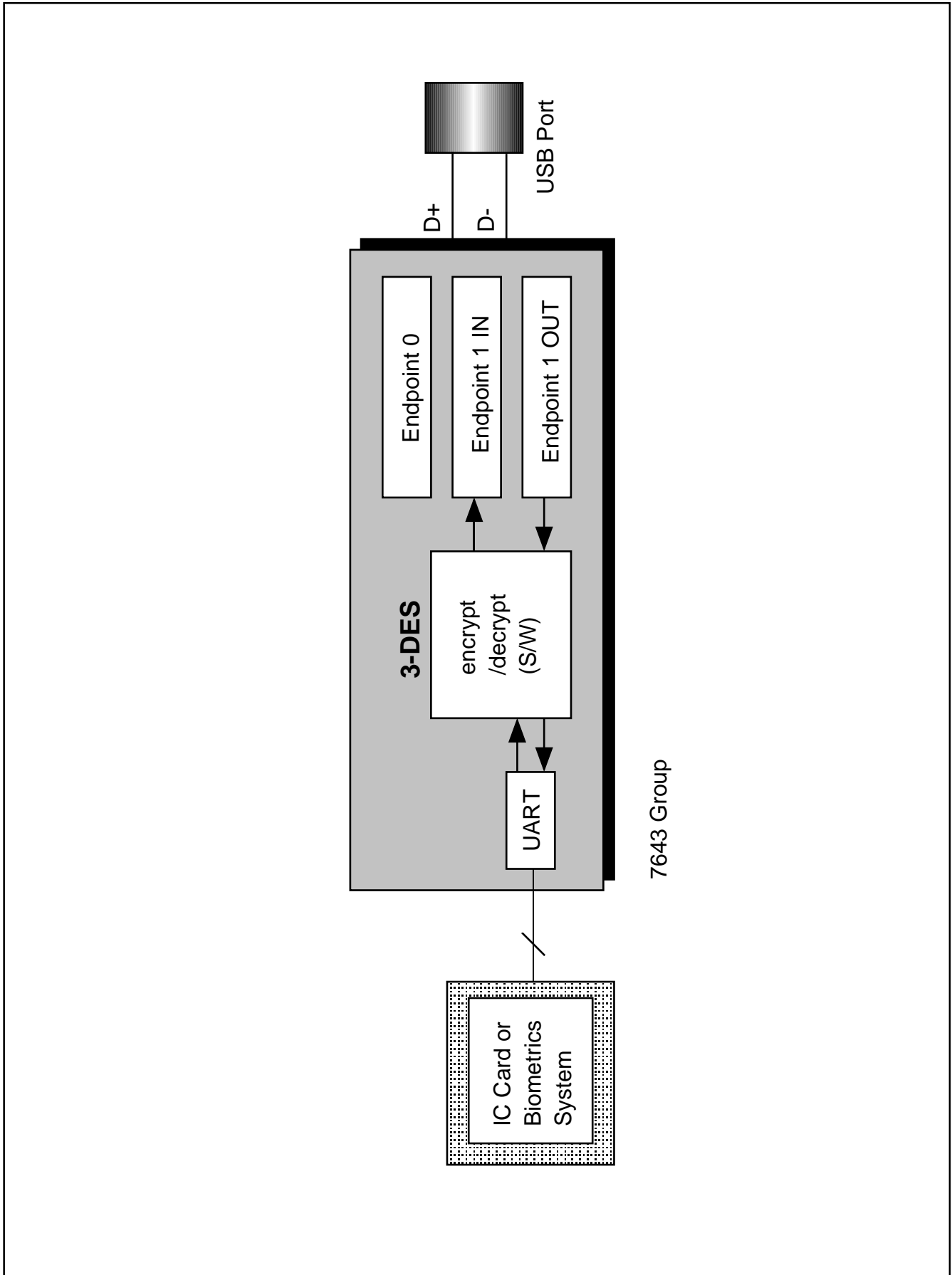


Fig. 2.6.57 Encryption system application example

### 2.6.10 Notes on USB function

#### (1) USB receive

- For endpoints 1 or 2, when there are two packets of data in the OUT FIFO, after reading out the max OUT packet size of data, one packet of data will remain in the OUT FIFO. In this case, even if the OUT\_PKT\_RDY flag is set to "0", it will return to "1" after 83ns ( $V_{cc} = 5V$ ,  $f(X_{IN}) = 24MHz$ ).
- Read one packet data from the OUT FIFO before clearing the OUT\_PKT\_RDY flag. If the OUT\_PKT\_RDY flag is cleared while one packet data is being read, the internal Read pointer cannot operate normally.

#### (2) USB transmit

- Determine the number of data packets in the IN FIFO by the value of the IN\_PKT\_RDY bit and the TX\_NOT\_EPT flag. For more details, refer to Table 2.6.2 in [Section 2.6.4 USB transmit].
- For endpoints 1 or 2, to transmit a NULL packet, set the IN\_PKT\_RDY bit to "1" without writing data to the FIFO.

#### (3) External circuits

- Figure 2.6.58 shows a peripheral circuit example of the external capacitor (Ext. Cap) pin. Connect a low-frequency 2.2  $\mu F$  Tantal capacitor and a 0.1  $\mu F$  ceramic capacitor (X7R type recommended) in parallel between the Ext. Cap pin and the Vss pin. In addition, connect a 1.5k $\Omega$  ( $\pm 5\%$ ) resistor between the Ext. Cap pin and the D+ pin.
- The Full-Speed USB2.0 specification requires a driver-impedance of 28 to 44  $\Omega$  (refer to Clause 7.1.1.1 Full-Speed (12Mb/s) Driver Characteristics). In order to meet USB specification impedance requirements, connect a resistor (27 to 33  $\Omega$  recommended) in series to the USB D+ and D- pins. In addition, in order to reduce the ringing and control the falling/rising timing of USB D+/D- and a crossover point, connect a capacitor (69pF recommended) between the USB D+/D- pins and the Vss pin if necessary. The values and structure of those peripheral elements depend on the impedance characteristics and the layout of the printed circuit board. Accordingly, evaluate your system and observe waveforms before actual use and decide use of elements and the values of resistors and capacitors.
- Make sure the wiring of the USB connector or USB cable is as short as possible.
- Make sure the USB D+/D- lines do not cross any other wires. Keep a large GND area to protect the USB lines. Also, make sure you use a USB specification compliant connector for the connection.
- All passive components must be located as close as possible to the LPF pin. For recommended values, see Figure 2.6.59.
- An insulation connector (Ferrite Beads) must be connected between AVss and Vss pins and between AVcc and Vcc pins.
- Connect 0.1  $\mu F$  and 4.7  $\mu F$  capacitors in parallel between the Vss and Vcc pins and between the AVss and AVcc pins.

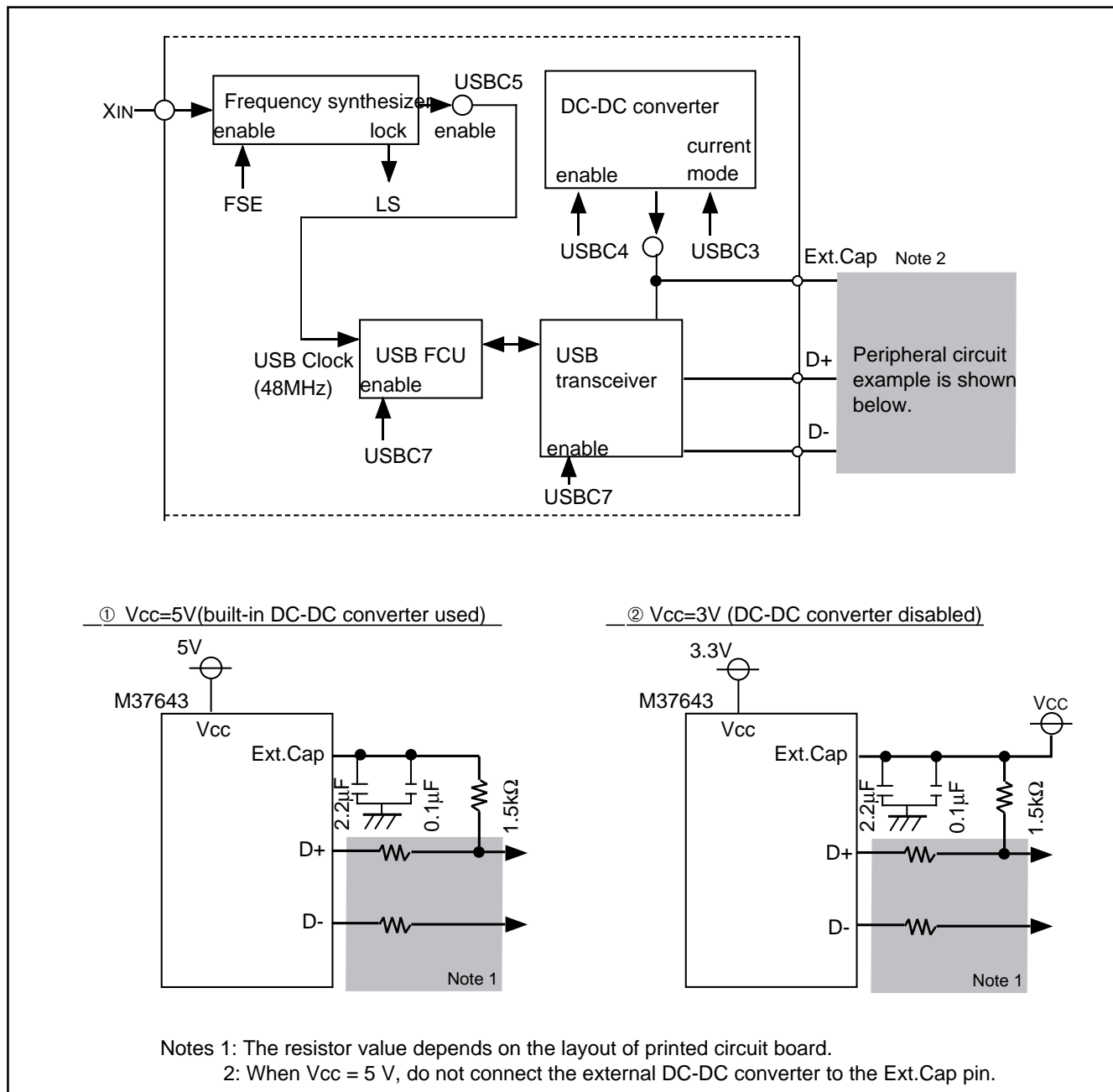


Fig. 2.6.58 Peripheral circuit example

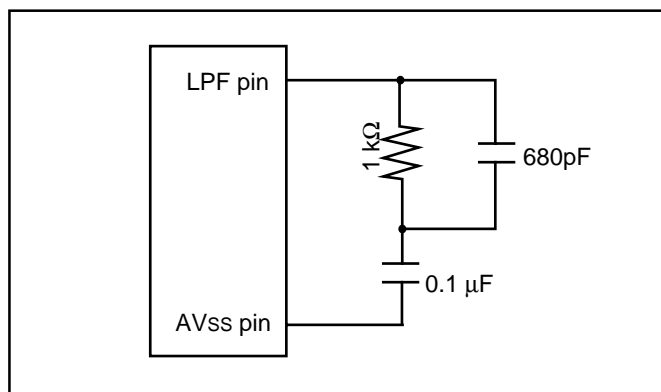


Fig. 2.6.59 LPF peripheral circuit

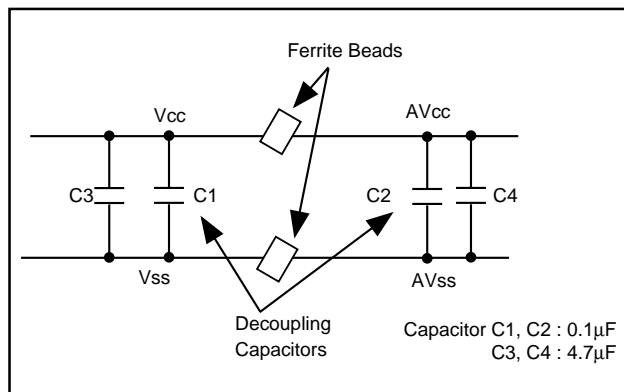


Fig. 2.6.60 Connection of insulation connector

#### (4) 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.

#### (5) Registers and bits

- When using endpoint 0, use USB endpoint 0 IN max. packet size register (IN\_MAXP: address 005B<sub>16</sub>) for data transfer to/from (IN packet size or OUT packet size).
- When not using USB endpoint x (x = 0 to 2) IN max. packet size register (IN\_MAXP: address 005B<sub>16</sub>), USB endpoint x (x = 0 to 2) OUT max. packet size register (OUT\_MAXP: address 005C<sub>16</sub>), clear "0" to this register.
- Make sure to write to/read from the USB interrupt status register 1 (USBIS1: address 0052<sub>16</sub>) first and then USB interrupt status register 2 (USBIS2: address 0053<sub>16</sub>).
- When accessing the following registers, always confirm that the index (bits 0 to 2 of USB endpoint index register (USBINDEX: address 0058<sub>16</sub>) is set properly.
  - USB endpoint x IN control register (IN\_CSR: address 0059<sub>16</sub>)
  - USB endpoint x OUT control register (OUT\_CSR: address 005A<sub>16</sub>)
  - USB endpoint x IN max. packet size register (IN\_MAXP: address 005B<sub>16</sub>)
  - USB endpoint x OUT max. packet size register (OUT\_MAXP: address 005C<sub>16</sub>)
  - USB endpoint x OUT write count register (WRT\_CNT: address 005D<sub>16</sub>)
  - USB endpoint FIFO mode register (USBFIFOMR: address 005F<sub>16</sub>)
- When the USB reset interrupt status flag is kept at "1", all other flags in the USB internal registers (addresses 0050<sub>16</sub> to 005F<sub>16</sub>) will return to their reset status. However, the following registers are not affected by the USB reset:
  - USB control register (USBC: address 0013<sub>16</sub>),
  - Frequency synthesizer control register (FSC: address 006C<sub>16</sub>),
  - Clock control register (CCR: address 001F<sub>16</sub>),
  - USB endpoint-x FIFO register (USB FIFOx: addresses 0060<sub>16</sub> to 0062<sub>16</sub>).
- When not using the USB function, set the USB line driver supply enable bit of the USB control register (USBC: address 0013<sub>16</sub>) to "1" for power supply to the internal circuits (at Vcc = 5V).
- The IN\_PKT\_RDY Bit can be set by software even when using the AUTO\_SET function.
- When the USB clock is disabled, do not write to any USB internal registers (addresses 0050<sub>16</sub> to 0062<sub>16</sub>) other than registers USBC, CCR, and FSC.
- In the USB suspend state, USB enable bit is fixed to "1" (USB clock enabled state). To write to a USB register other than registers USBC, CCR, and FSC (i.e. to addresses 0050<sub>16</sub> to 0062<sub>16</sub>) after the MCU recovers from the USB suspend state, set the USB clock enable bit (USBC, bit 5) to "1", and write to the register after a  $\phi = 4$  wait cycle.
- When using the MCU at Vcc = 3V, set the USB line driver supply selection bit (USBC, bit 4) to "0" (line driver disabled). Note that the USB line driver divider select bit (USBC, bit 3) does not affect the USB operation.

- Use the transfer instructions such as **LDA** and **STA** to set the following registers:
  - USB interrupt status registers 1, 2 (USBIS1, USBIS2: addresses 0052<sub>16</sub>, 0053<sub>16</sub>)
  - USB endpoint 0 IN control register (IN\_CSR: address 0059<sub>16</sub>)
  - USB endpoint x IN control register (IN\_CSR: address 0059<sub>16</sub>)
  - USB endpoint x OUT control register (OUT\_CSR: address 005A<sub>16</sub>)

Do not use the read-modify-write instructions such as the **SEB** or the **CLB** instruction.

When writing to bits shown by Table 2.6.3 using the transfer instruction such as **LDA** or **STA**, a value which never affects its bit state is required.

Take the following sequence to change these bits contents:

1. Store the register contents onto a variable or a data register.
  2. Change the target bit on the variable or the data register. Simultaneously mask the bit so that its bit state cannot be changed. (See to Table 2.6.3.)
  3. Write the value from the variable or the data register to the register using the transfer instruction such as **LDA** or **STA**.
- Erase the contents of the IN FIFO or OUT FIFO by setting the FLUSH bit to "1".  
If there are two packets in the FIFO, the older packet will be erased.  
This will also modify the state of the IN\_PKT\_RDY bit and the TX\_NOT\_EPT flag.
  - Transfer data may be erased if the FLUSH bit is set to "1" during a transfer.  
Receive: Set the FLUSH bit to "1" when the OUT\_PKT\_RDY flag is "1".
  - To use the AUTO\_SET function for an IN transfer when the AUTO\_SET bit is set to 1, set the FIFO to single buffer mode.

**Table 2.6.3 Bits of which state might be changed owing to software write**

Register name	Bit name	Value not affecting state ( <b>Note</b> )
USB endpoint 0 IN control register	IN_PKT_RDY (b1)	"0"
	DATA_END (b3)	"0"
	FORCE_STALL (b4)	"1"
USB endpoint x (x = 1, 2) IN control register	IN_PKT_RDY (b0)	"0"
USB endpoint x (x = 1, 2) OUT control register	OUT_PKT_RDY (b0)	"1"
	FORCE_STALL (b4)	"1"

**Note:** Writing this value will not change the bit state, because this value cannot be written to the bit by software.

Some application notes are available on the Web site:

<http://www.renesas.com/en/usb>

Please refer to them for more explanation and application of USB function.

## 2.7 Frequency synthesizer

This paragraph explains the registers setting method and the notes related to the frequency synthesizer.

### 2.7.1 Memory map

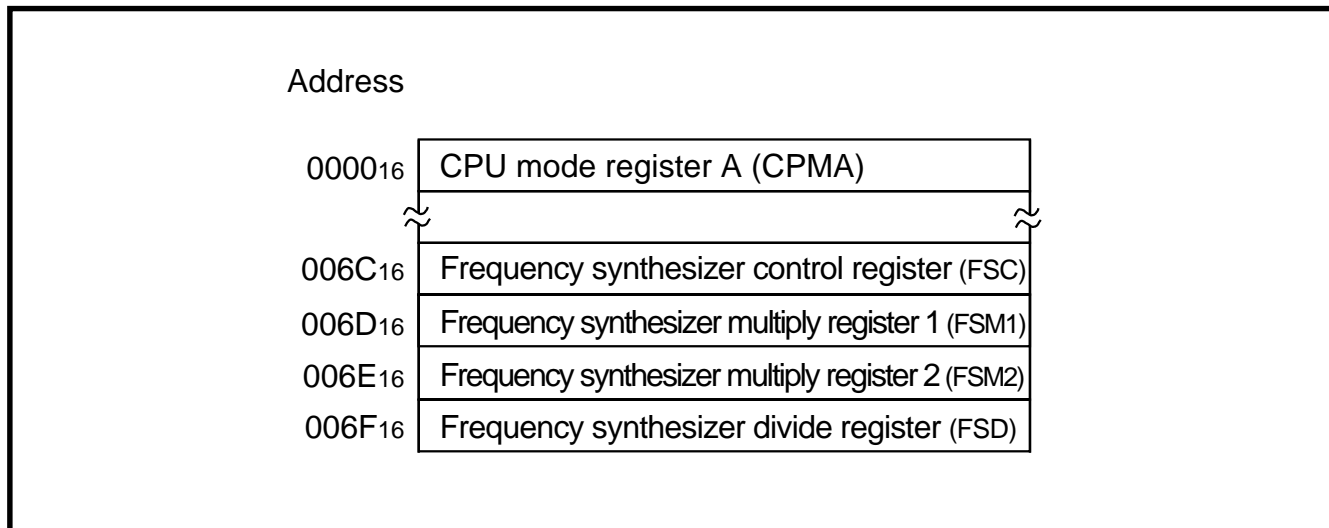


Fig. 2.7.1 Memory map of registers related to frequency synthesizer



## 2.7.2 Related registers

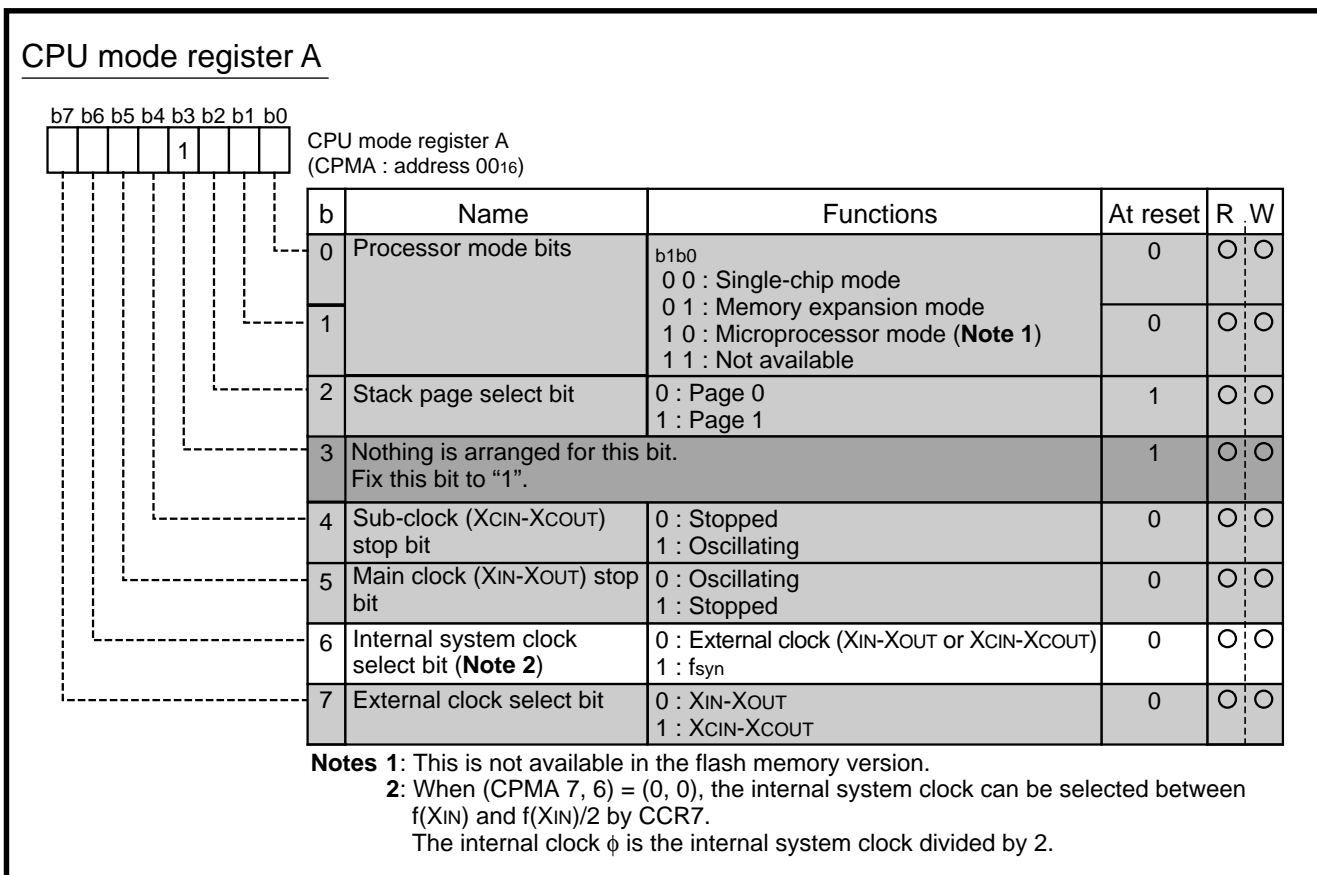


Fig. 2.7.2 Structure of CPU mode register A

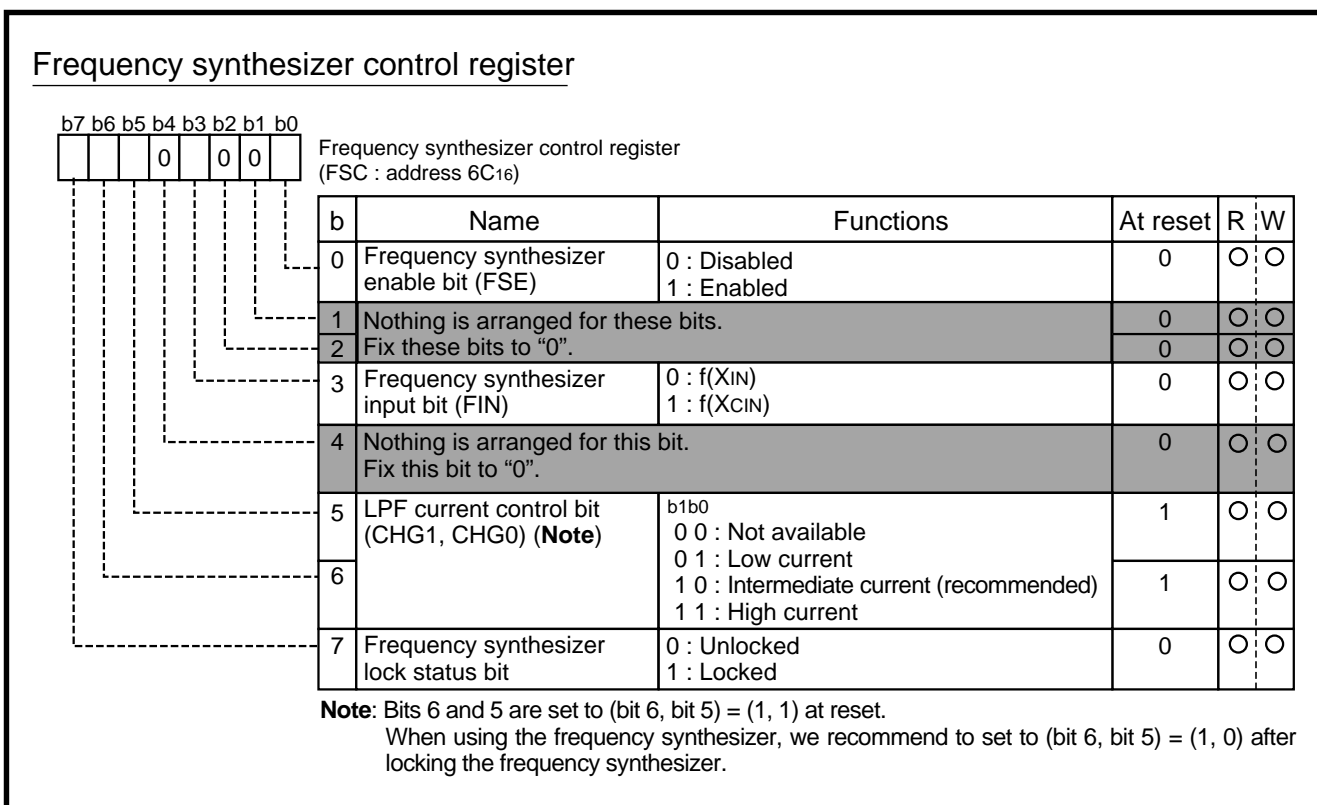


Fig. 2.7.3 Structure of Frequency synthesizer control register

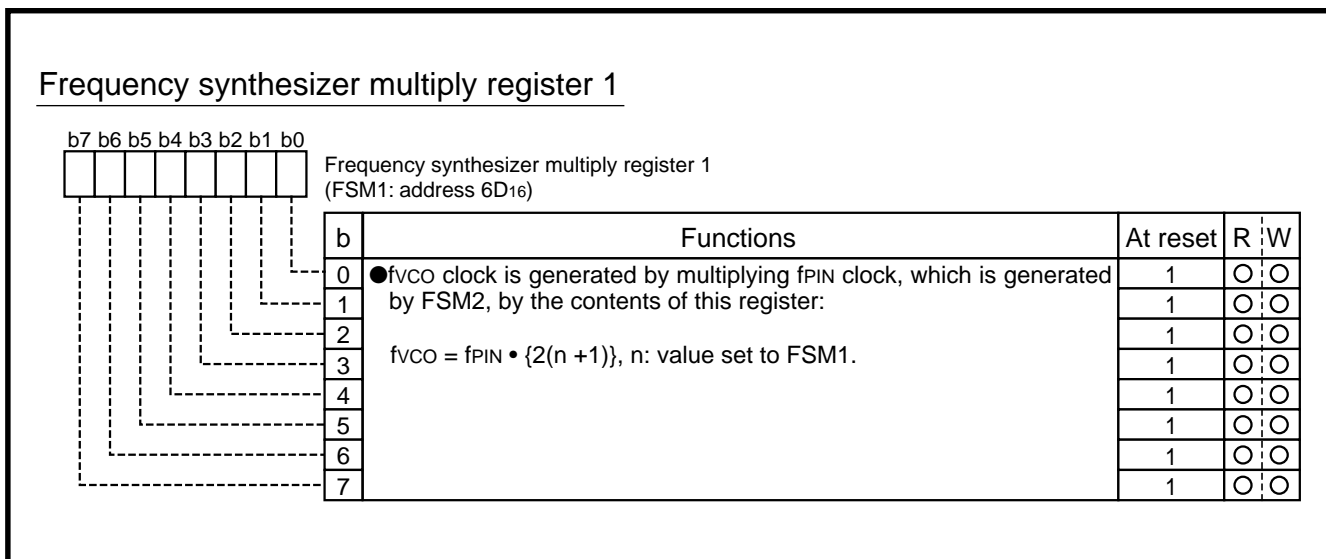


Fig. 2.7.4 Structure of Frequency synthesizer multiply register 1

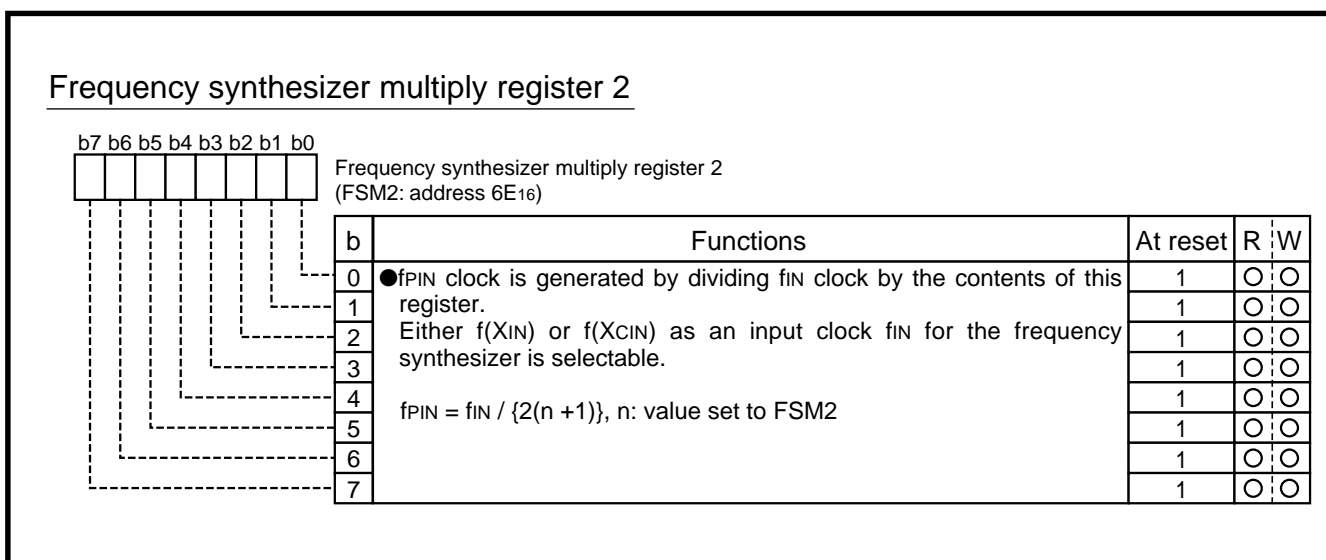


Fig. 2.7.5 Structure of Frequency synthesizer multiply register 2

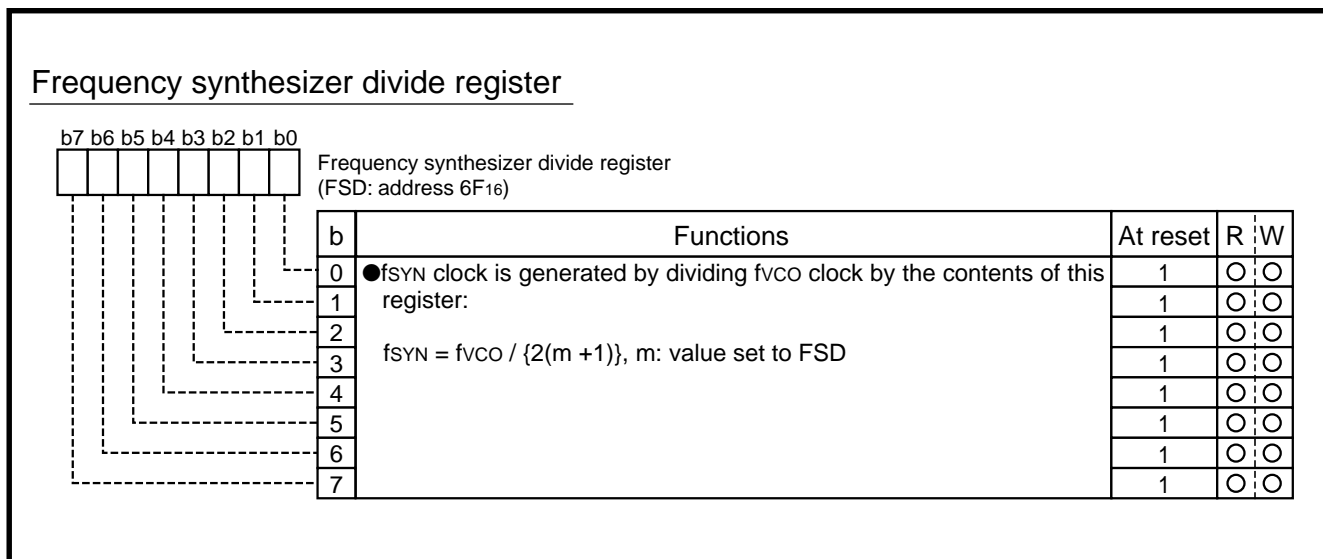


Fig. 2.7.6 Structure of Frequency synthesizer divide register

### 2.7.3 Functional description

The frequency synthesizer generates the 48 MHz clock required by  $f_{USB}$  and  $f_{SYN}$ , which are multiples of the external input reference  $f(X_{IN})$  or  $f(X_{CIN})$ . To use the frequency synthesizer, set the frequency synthesizer enable bit of frequency synthesizer control register (address  $6C_{16}$ ) to "1".

The frequency synthesizer input bit selects either  $f(X_{IN})$  or  $f(X_{CIN})$  as an input clock  $f_{IN}$  for the frequency synthesizer.

Figure 2.7.7 shows the block diagram for the frequency synthesizer circuit.

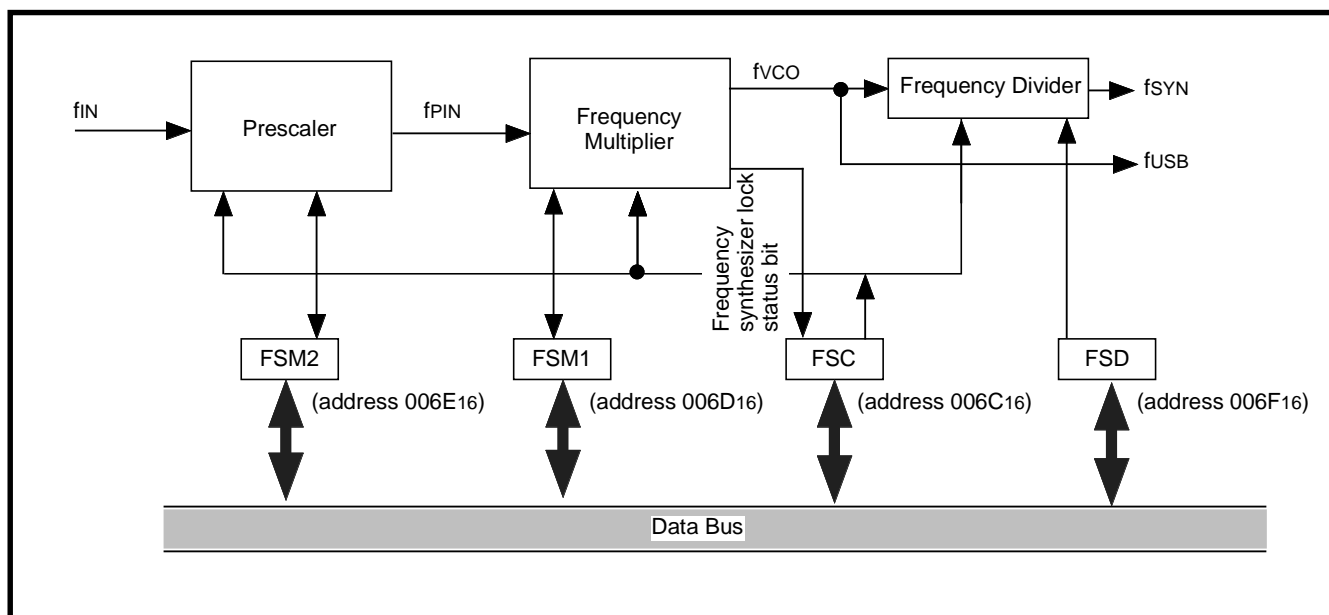


Fig. 2.7.7 Block diagram for frequency synthesizer circuit

#### (1) $f_{PIN}$

$f_{IN}$  is divided by the contents of frequency synthesizer multiply register 2 (FSM2: address  $6E_{16}$ ) to generate  $f_{PIN}$ , where

$$f_{PIN} = f_{IN} / 2(n + 1), \quad n: \text{value set to FSM2.}$$

When the value of FSM2 is set to 255, the division is not performed and  $f_{PIN}$  will equal  $f_{IN}$ .

Figure 2.7.8 shows the frequency synthesizer multiply register 2 setting example.

**Note:** Be sure to set  $f_{PIN}$  to 1 MHz or more.

$f_{PIN}$	FSM2 register set value		$f_{IN}$
	Decimal	Hexadecimal	
24 MHz	255	FF <sub>16</sub>	24.00 MHz
1 MHz	11	0B <sub>16</sub>	24.00 MHz
2 MHz	5	05 <sub>16</sub>	24.00 MHz
3 MHz	3	03 <sub>16</sub>	24.00 MHz
6 MHz	1	01 <sub>16</sub>	24.00 MHz
12 MHz	0	00 <sub>16</sub>	24.00 MHz

Fig. 2.7.8 Frequency synthesizer multiply register 2 setting example

**(2)  $f_{VCO}$** 

$f_{VCO}$  is generated by multiplying  $f_{PIN}$  by the contents of frequency synthesizer multiply register 1 (FSM1: address 6D<sub>16</sub>), where

$$f_{VCO} = f_{PIN} \times \{2(n + 1)\}, n: \text{value set to FSM1.}$$

Set the value of FSM1 so that  $f_{VCO}$  will be 48 MHz.

$f_{VCO}$  is optimized in the frequency synthesizer to be used as  $f_{USB}$  and it will be sent into the USB function control unit.

While the frequency synthesizer enable bit is "0" (disabled),  $f_{VCO}$  retains "H" or "L" level.

Figure 2.7.9 shows the frequency synthesizer multiply register 1 setting example.

f <sub>PIN</sub>	FSM1 register set value		f <sub>VCO</sub>
	Decimal	Hexadecimal	
2 MHz	11	0B <sub>16</sub>	48.00 MHz
4 MHz	5	05 <sub>16</sub>	48.00 MHz
6 MHz	3	03 <sub>16</sub>	48.00 MHz
12 MHz	1	01 <sub>16</sub>	48.00 MHz
24 MHz	0	00 <sub>16</sub>	48.00 MHz

**Fig. 2.7.9 Frequency synthesizer multiply register 1 setting example**

**(3)  $f_{SYN}$** 

$f_{VCO}$  is divided by the contents of frequency synthesizer divide register (FSD: address 6F<sub>16</sub>) to generate  $f_{SYN}$ , where

$$f_{SYN} = f_{VCO} / 2(m + 1), m: \text{value set to FSD.}$$

When the value of FSD is set to 255, the division is not performed and  $f_{SYN}$  becomes invalid.

$f_{SYN}$  can be used as the internal system clock by setting the internal system clock select bit of CPU mode register A.

Figure 2.7.10 shows the frequency synthesizer divide register setting example.

When the frequency synthesizer lock status bit is "1" in the frequency synthesizer enabled, this indicates that  $f_{SYN}$  and  $f_{VCO}$  have correct frequencies.

f <sub>VCO</sub>	FSD register set value		f <sub>SYN</sub>
	Decimal	Hexadecimal	
48.00 MHz	00	00 <sub>16</sub>	24.00 MHz
48.00 MHz	127	7F <sub>16</sub>	187.5 kHz

**Fig. 2.7.10 Frequency synthesizer divide register setting example**

**(4) Recovering from hardware reset**

The frequency synthesizer and DC-DC converter must be set up as follows when recovering from a Hardware Reset:

- ① Enable the frequency synthesizer after setting the frequency synthesizer related registers (addresses 6C<sub>16</sub> to 6F<sub>16</sub>). Then wait for 2 ms.
- ② Check the frequency synthesizer lock status bit. If “0”, wait for 0.1 ms and then recheck.
- ③ To use the intermediate current, set the LPF current control bits of frequency synthesizer control register (address 6C<sub>16</sub>) to (b6, b5) = “10”.
- ④ When using the USB built-in DC-DC converter, set the USB line driver supply enable bit of the USB control register (address 13<sub>16</sub>) to “1”. This setting must be done 2 ms or more later than the setup described in step ①. The USB line driver current control bit must be set to “0” at this time. (When V<sub>CC</sub> = 3.3V, the setting explained in this step is not necessary.)
- ⑤ After waiting for (C + 1) ms so that the external capacitance pin (Ext. Cap. pin) can reach approximately 3.3 V, set the USB clock enable bit to “1”. At this time, “C” equals the capacitance (μF) of the capacitor connected to the Ext. Cap. pin. For example, if 2.2 μF and 0.1 μF capacitors are connected to the Ext. Cap. in parallel, the required wait will be (2.3 + 1) ms.
- ⑥ After enabling the USB clock, wait for 4 or more φ cycles, and then set the USB enable bit to “1”.

Do not write to any of the USB internal registers (addresses 50<sub>16</sub> to 62<sub>16</sub>) until the USB clock enabled, except for the USB control register (address 13<sub>16</sub>), clock control register (address 1F<sub>16</sub>), and frequency synthesizer control register (address 6C<sub>16</sub>).

**2.7.4 Notes on frequency synthesizer**

- Bits 6 and 5 of the frequency synthesizer control register (address 6C<sub>16</sub>) are initialized to (b6, b5) = “11” after reset release. Make sure to set bits 6 and 5 to “10” after the frequency synthesizer lock status bit goes to “1”.
- Use the frequency synthesizer output clocks 2 ms to 5 ms later than setting the frequency synthesizer enable bit to “1” (enabled). After that do not change any register values because it might cause output clocks unstabilized temporarily.
- Make sure to connect a low-pulse filter to the LPF pin when using the frequency synthesizer.
- The frequency synthesizer divide register set value never affects f<sub>USB</sub> frequency.
- When using the f<sub>SYN</sub> as an internal system clock, set the frequency synthesizer divide register so that f<sub>SYN</sub> could be 24 MHz or less.
- When using the frequency synthesized clock function, we recommend using the fastest frequency possible of f(X<sub>IN</sub>) or f(X<sub>CIN</sub>) as an input clock for the PLL.
- Set the value of frequency synthesizer multiply register 2 (FSM2) so that the f<sub>PIN</sub> is 1 MHz or higher.

## 2.8 External devices connection

This paragraph explains the registers setting method and the notes related to the external devices connection.

### 2.8.1 Memory map

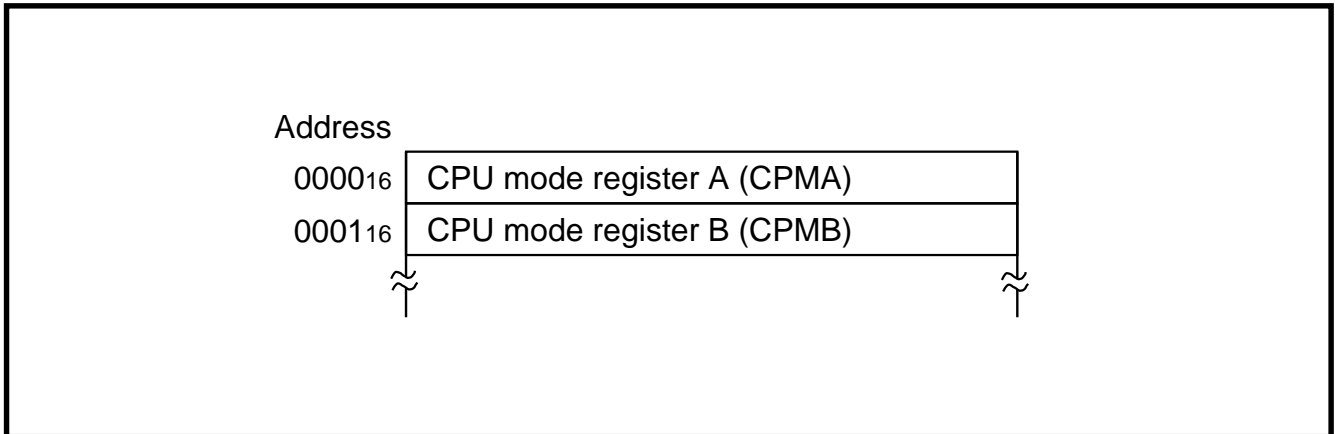


Fig. 2.8.1 Memory map of registers related to external devices connection

## 2.8.2 Related registers

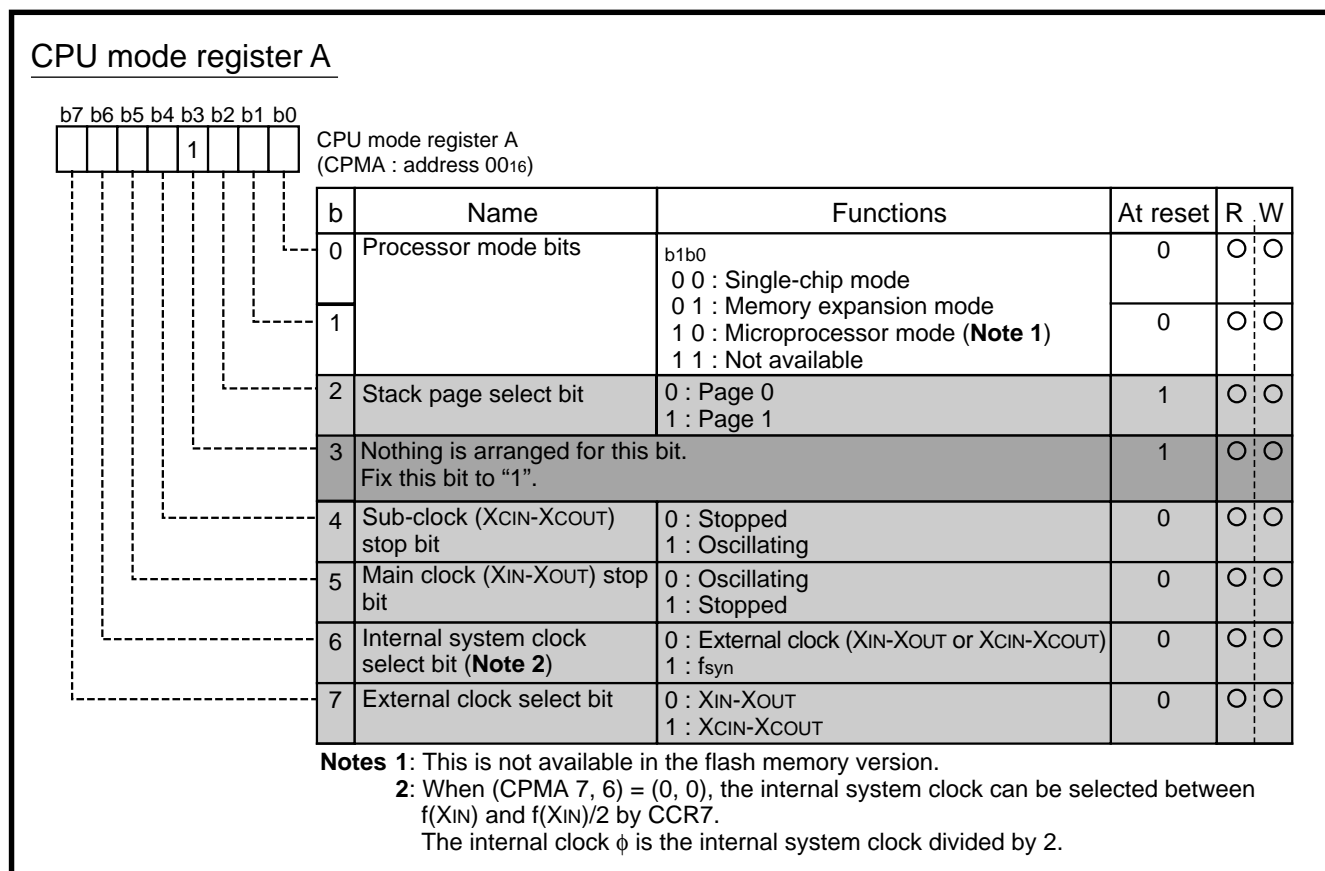


Fig. 2.8.2 Structure of CPU mode register A

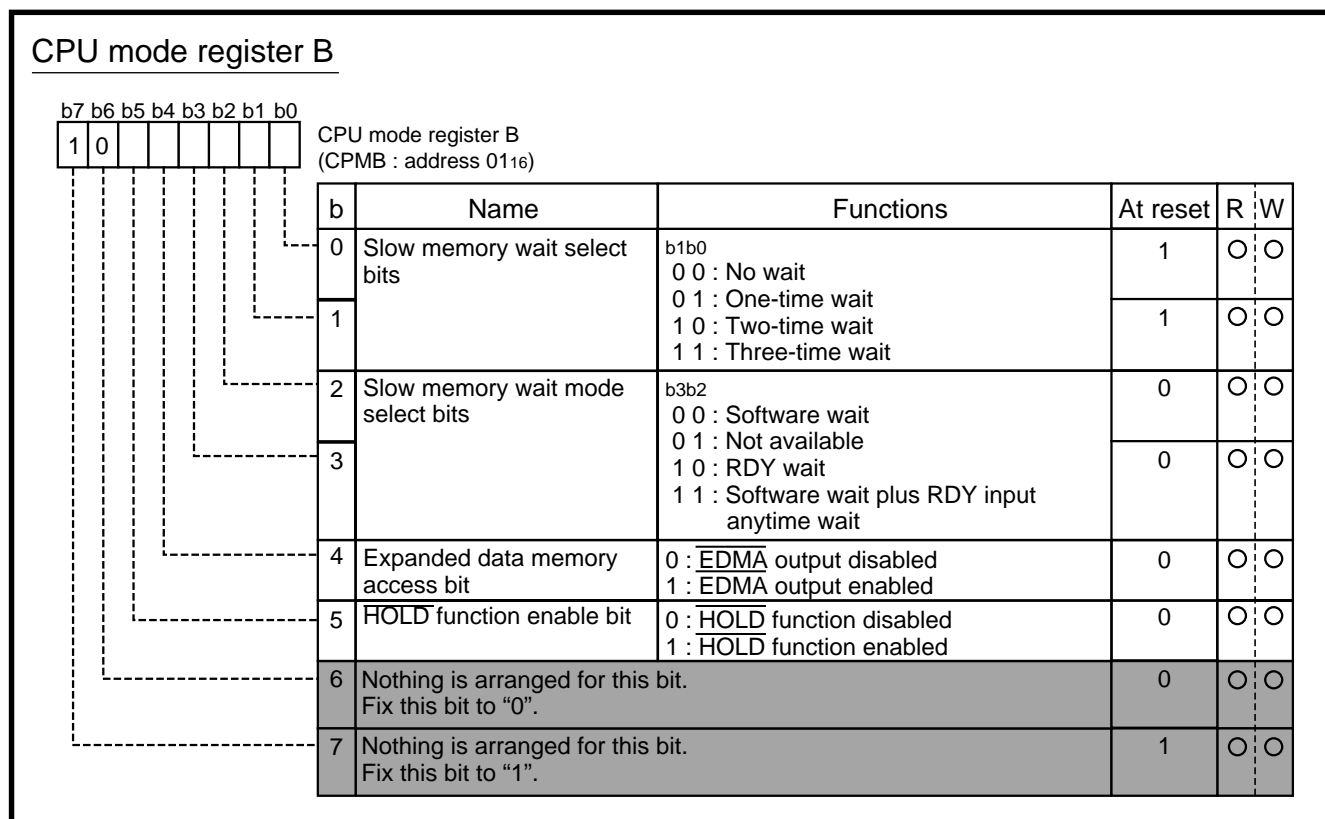


Fig. 2.8.3 Structure of CPU mode register B



### 2.8.3 Functional description

This MCU starts its operation in the single-chip mode just after reset.

#### (1) Memory expansion mode

This mode is selected by setting "01" to the processor mode bits (b1, b0 of CPMA) in software with CNV<sub>SS</sub> connected to V<sub>SS</sub>.

In this mode, the ports function as follows:

Ports P0 and P1 as address buses (AB<sub>0</sub> to AB<sub>15</sub>)

Port P2 as data buses (DB<sub>0</sub> to DB<sub>7</sub>)

Ports P3<sub>3</sub> to P3<sub>7</sub> as DMA<sub>OUT</sub>,  $\phi_{OUT}$ , SYNC<sub>OUT</sub>,  $\overline{WR}$ ,  $\overline{RD}$  pins respectively.

This mode enables external memory expansion while maintaining the validity of the internal ROM.

#### (2) Microprocessor mode

This mode is selected by resetting the MCU with CNV<sub>SS</sub> connected to V<sub>CC</sub>, or by setting "10" to the processor mode bits (b1, b0 of CPMA) in software with CNV<sub>SS</sub> connected to V<sub>SS</sub>. The function is the same as that of memory expansion mode.

In the microprocessor mode, the internal ROM is no longer valid and an external memory must be used.

Do not set this mode in the flash memory version.

### 2.8.4 Slow memory wait

The slow memory wait function is for easier interfacing with external devices that have long access times. This can be enabled in the memory expansion mode and microprocessor mode.

The wait is effective only to external areas. Access to internal area is always performed without wait.

#### (1) Software wait

The software wait is selected by setting "00" to the slow memory wait mode select bits (b3, b2 of CPMB).

Figure 2.8.4 shows the software wait timing example.

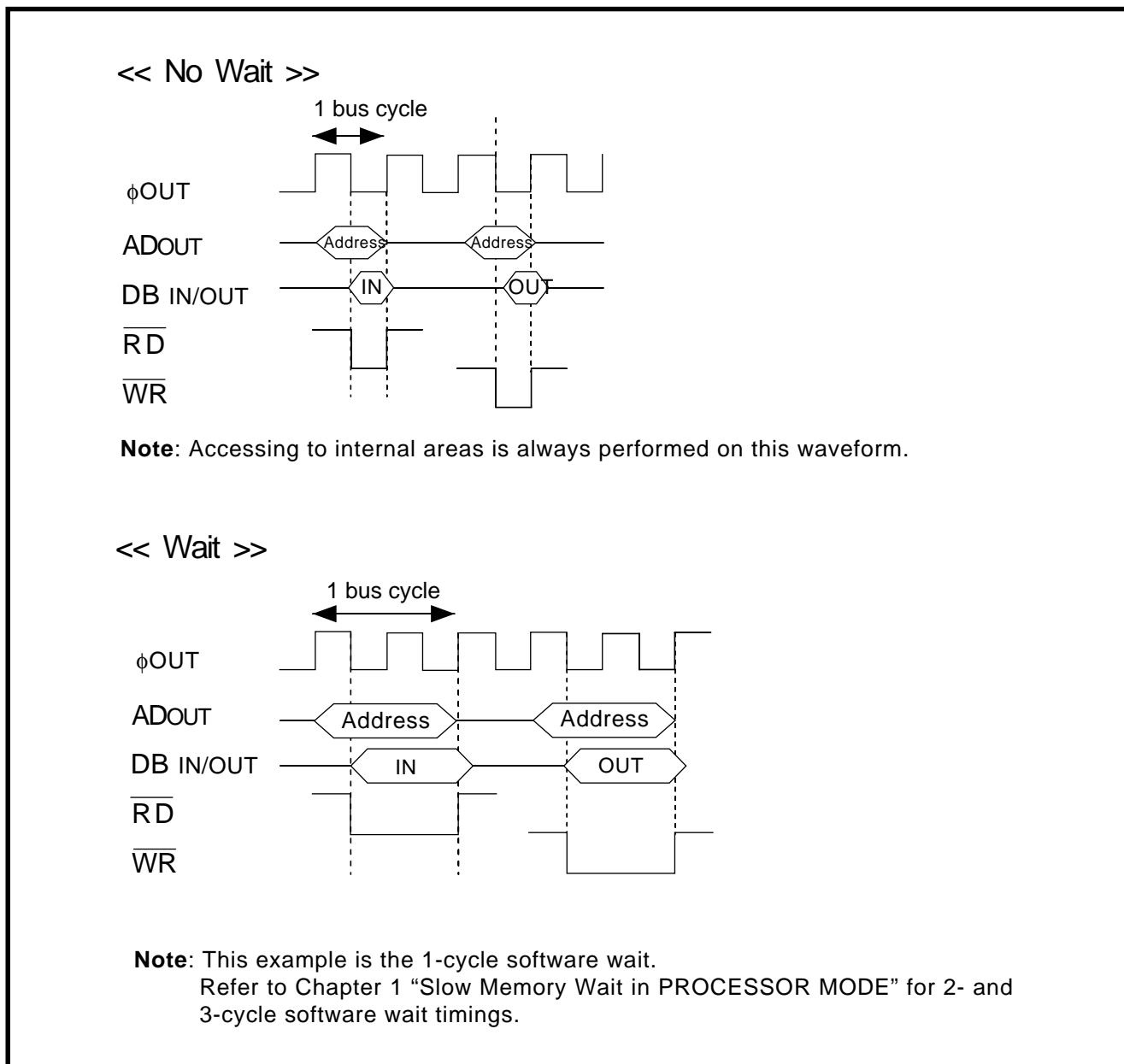


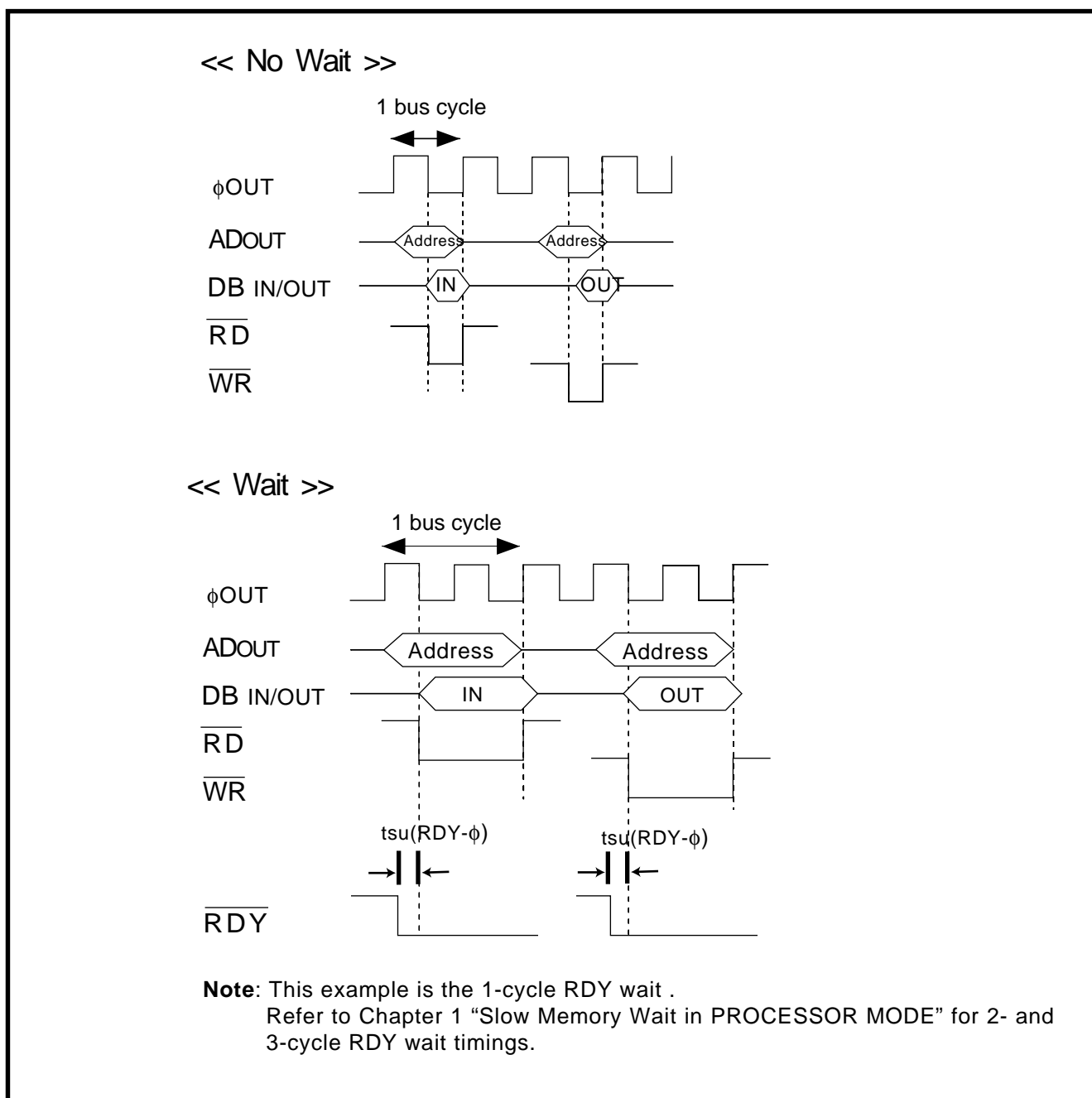
Fig. 2.8.4 Software wait timing example

**(2) RDY wait**

RDY wait is selected by setting "10" to the slow memory wait mode select bits (b3, b2 of CPMB). When a fixed time of "L" (tsu) is input to the RDY pin at the beginning of a read/write cycle (before  $\phi$  cycle falls), the MCU goes to the RDY state. Then the read/write cycle can be extended by one to three  $\phi$  cycles. The number of  $\phi$  cycles to be extended can be selected by the slow memory wait select bits (b1, b0 of CPMB).

Even if "L" is input to the RDY pin at the end of waited read/write cycle, the cycle is not extended. When a fixed time of "H" (tsu) is input to the RDY pin at the beginning of a read/write cycle (before  $\phi$  cycle falls), the MCU is released from the RDY state.

Figure 2.8.5 shows the RDY wait timing example.

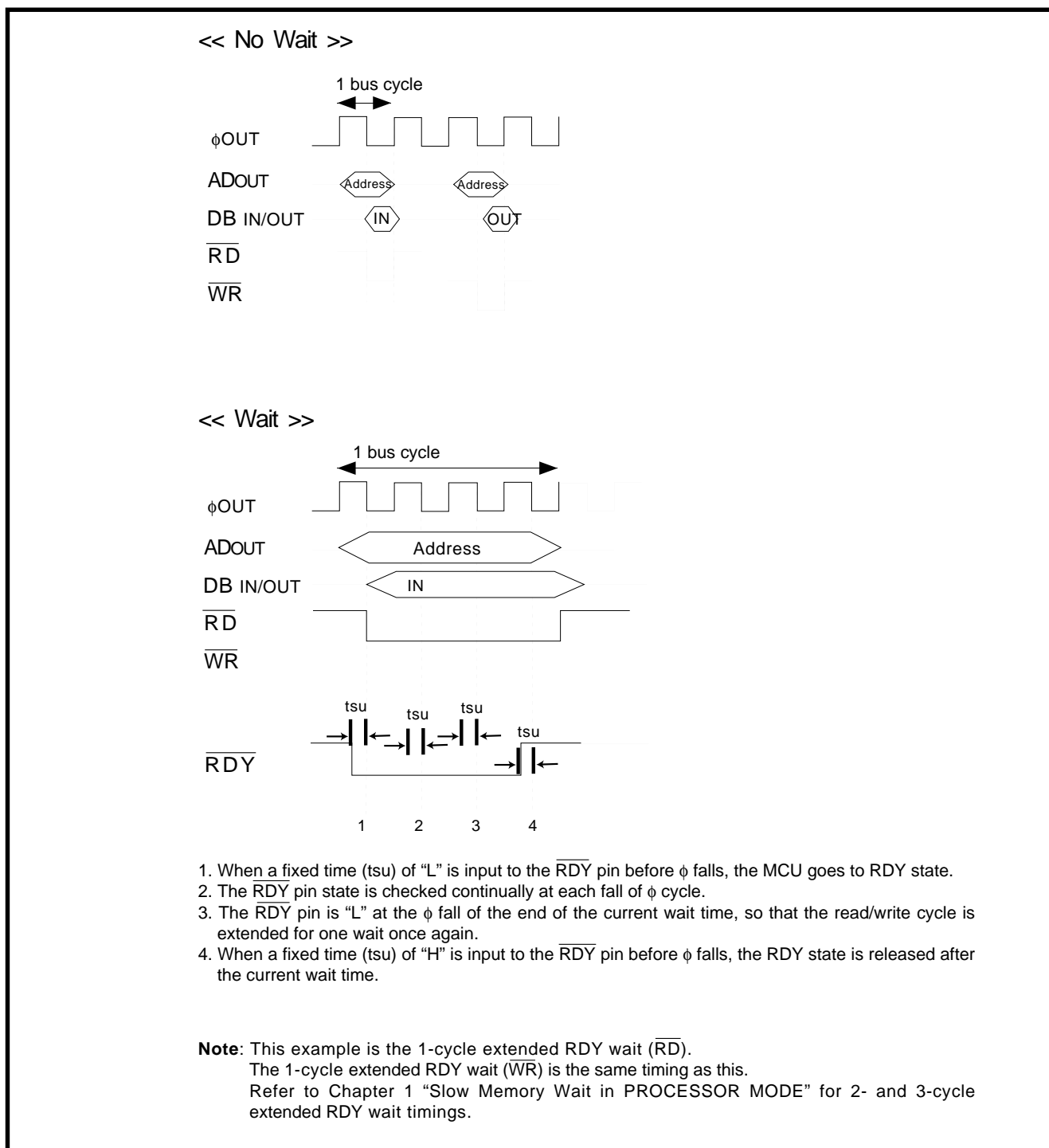


**Fig. 2.8.5 RDY wait timing example**

**(3) Software wait + Extended RDY wait**

Extended RDY wait (software wait plus RDY input anytime wait) is selected by setting "11" to the slow memory wait mode select bits (b3, b2 of CPMB). The read/write cycle can be extended when a fixed time ( $t_{su}$ ) of "L" is input to the RDY pin at the beginning of a read/write cycle (before  $\phi$  cycle falls). The RDY pin state is checked continually at each fall of  $\phi$  cycle until the RDY pin goes to "H" and the cycle keeps being extended. When a fixed time of "H" ( $t_{su}$ ) is input to the RDY pin at the beginning of a read/write cycle (before  $\phi$  cycle falls), the MCU is released from the wait within 1, 2, or 3  $\phi$  cycles as selected with the slow memory wait bits (b1, b0 of CPMB).

Figure 2.8.6 shows the extended RDY wait (software wait plus RDY input anytime wait) timing example



**Fig. 2.8.6 Extended RDY wait (software wait plus RDY input anytime wait) timing example**

### 2.8.5 HOLD function

The HOLD function is used for systems that consist of external circuits that access MCU buses without use of the CPU (Central Processing Unit). The HOLD function is used to generate the timing in which the MCU will relinquish the bus from the CPU to the external circuits. To use the HOLD function, set the HOLD function enable bit (b5 of CPMB) to "1". This can be enabled in the memory expansion mode and microprocessor mode.

When "L" level is input to the  $\overline{\text{HOLD}}$  pin, the MCU goes to the HOLD state and remains so while the pin is at "L". When the MCU relinquishes use of the bus, "L" level is output from the HLDA pin. The MCU puts ports P0 and P1 (address buses) and port P2 (data bus) to tri-state outputs and holds the  $\overline{\text{RD}}$  pin (P3<sub>7</sub>) and  $\overline{\text{WR}}$  pin (P3<sub>6</sub>) "H" level. This will prevent incorrect operations of external devices.

Though the clock supply to the CPU halts in HOLD state, the internal peripheral clocks and  $\phi_{\text{OUT}}$  (P3<sub>4</sub>) continues to be supplied.

When "H" level is input to the  $\overline{\text{HOLD}}$  pin, "H" level is output from the HLDA pin and the MCU can use address buses, data bus, signals  $\overline{\text{RD}}$  and  $\overline{\text{WR}}$  to access to external devices.

Figure 2.8.7 shows the Hold function timing diagram.

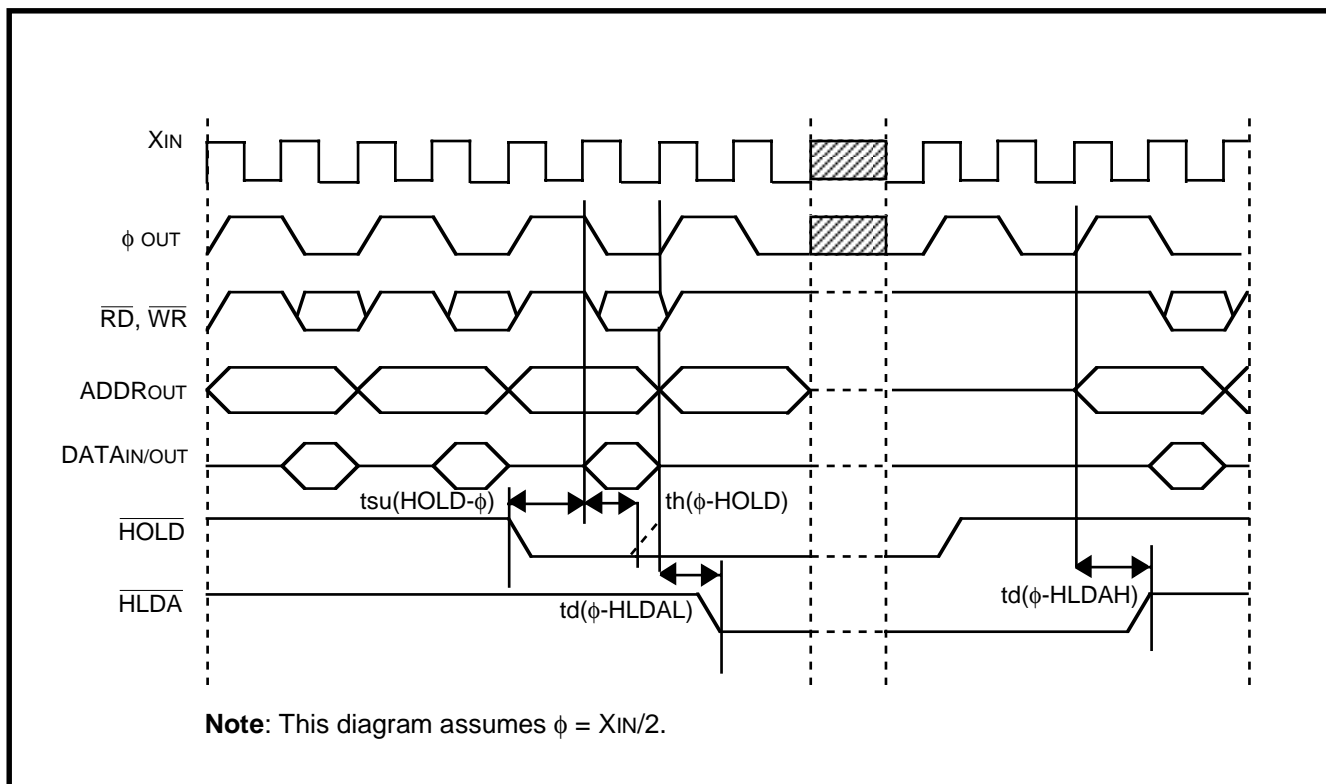


Fig. 2.8.7 Hold function timing diagram

### 2.8.6 Expanded data memory access

In Expanded Data Memory Access Mode (EDMA mode), the MCU can access a data area larger than 64 Kbytes with the LDA (\$zz), Y (indirect Y) instruction and the STA (\$zz), Y (indirect Y) instruction. It is only able to store and read data for the expanded data memory area. The access can be performed with T flag = "0" or "1". (T flag is Index X mode flag.)

To use this mode, set the expanded data memory access bit (b4 of CPMB) to "1". In this case,  $\overline{\text{EDMA}}$  pin (port P4<sub>0</sub>) goes "L" level during the read/write cycle of the LDA or STA instruction. The determination of which bank to access is done by using an I/O port to represent expanded addresses exceeding address bus (AB<sub>15</sub>)16. This signal and the port output signal are put together, and it becomes the chip select (selecting a bank, expanded memory).

In EDMA mode, the area from addresses 0000<sub>16</sub> to FFFF<sub>16</sub> can be accessed. When accessing a bank, follow this procedure (four banks are assumed):

- Bank specification (Data output to the port for a bank setup 16 (AB<sub>16</sub>), and 16 (AB<sub>17</sub>)). The user must setup 16 (AB<sub>16</sub>), and 16 (AB<sub>17</sub>).
- Enabling  $\overline{\text{EDMA}}$  output. (Set b4 of CPMB to "1".)
- Executing LDA (\$zz), Y (indirect Y) (In the case of read data of expanded data memory)

The data of the address (bank 0) + Y stored in the internal RAM address (\$zz) are loaded to the accumulator.

Figure 2.8.8 shows a connection example of memory access up to 256 Kbytes.

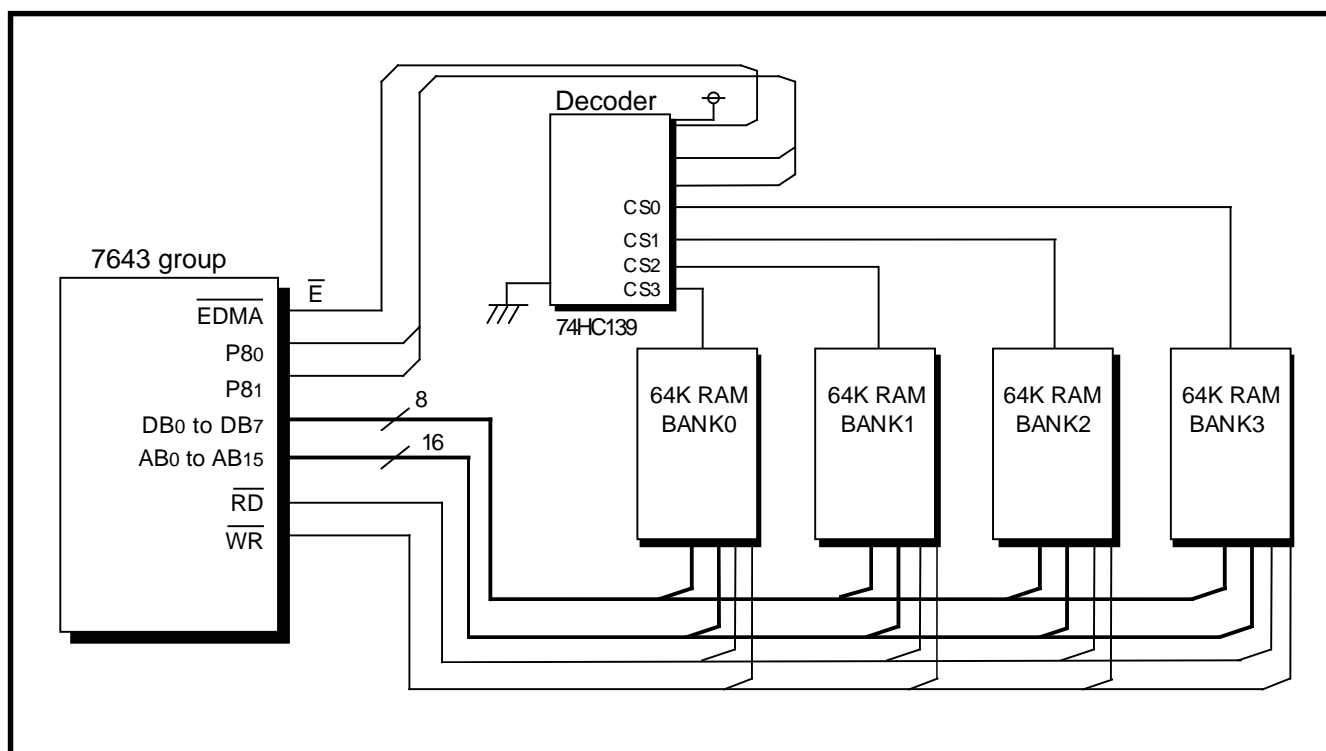


Fig. 2.8.8 Connection example of memory access up to 256 Kbytes

### 2.8.7 External devices connection example

Connection example for controlling external memory is shown below.

#### (1) External memory connection example : No Wait function

**Outline:** In microprocessor mode the external memory is accessed.

Figure 2.8.9 shows the external ROM and RAM example.

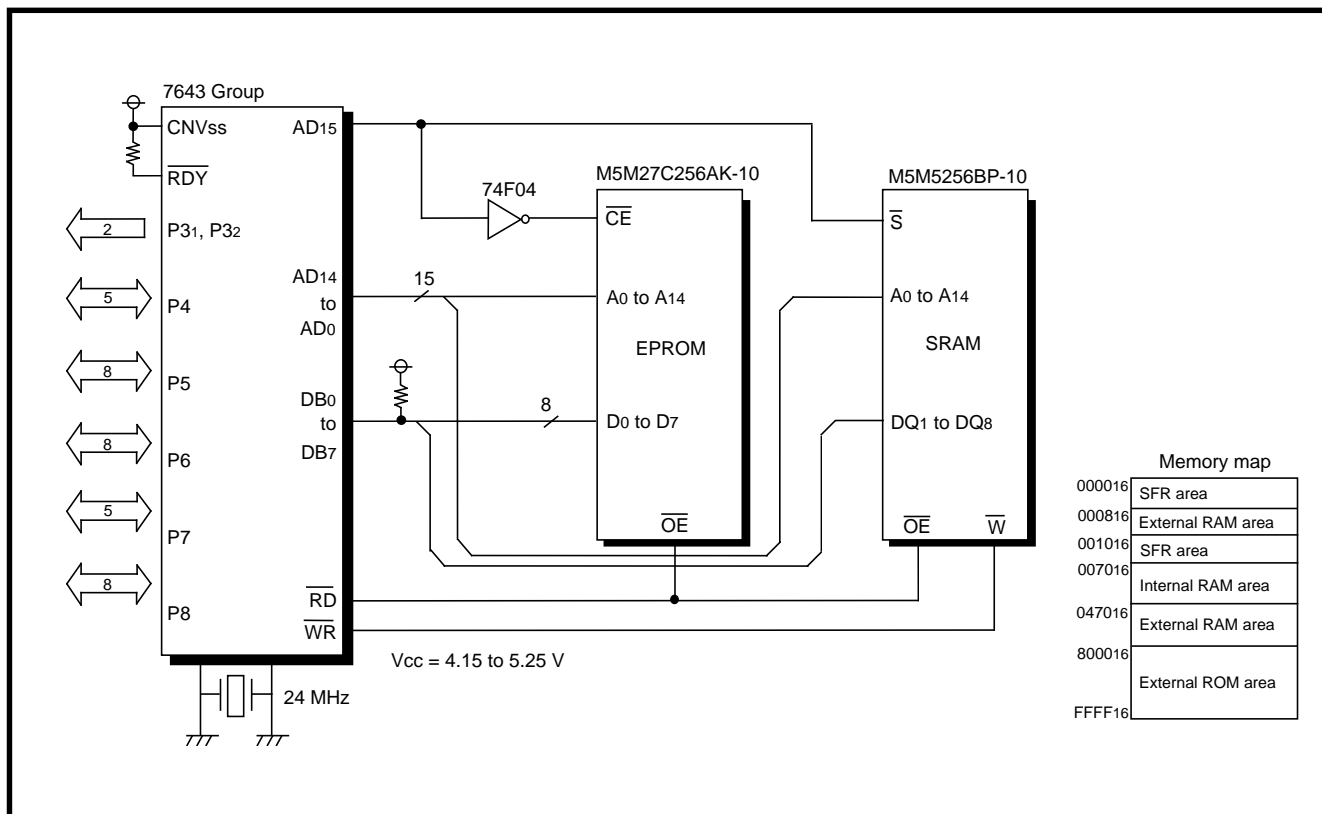


Fig. 2.8.9 External ROM and RAM example





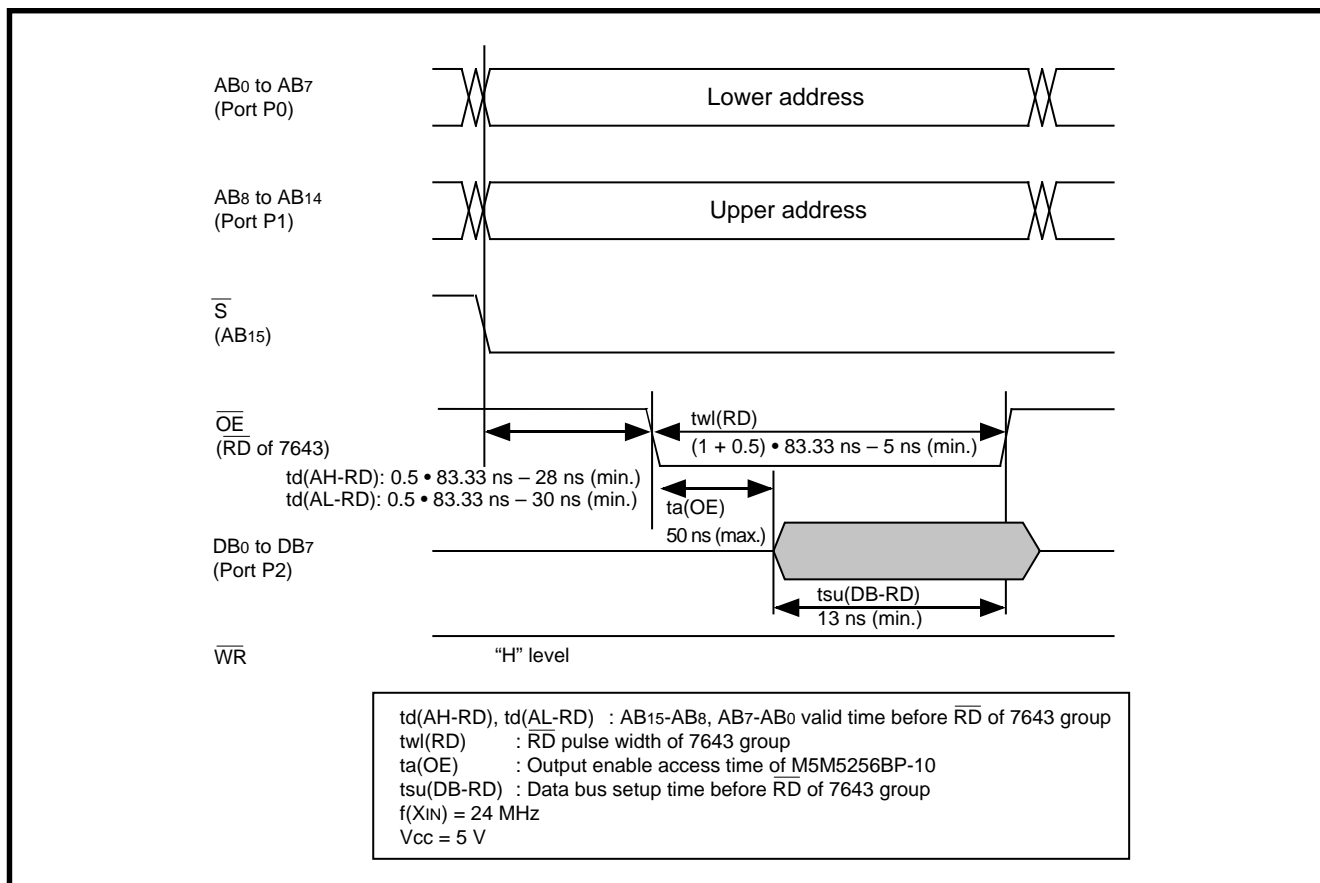


Fig. 2.8.11 Read cycle (OE access, SRAM)

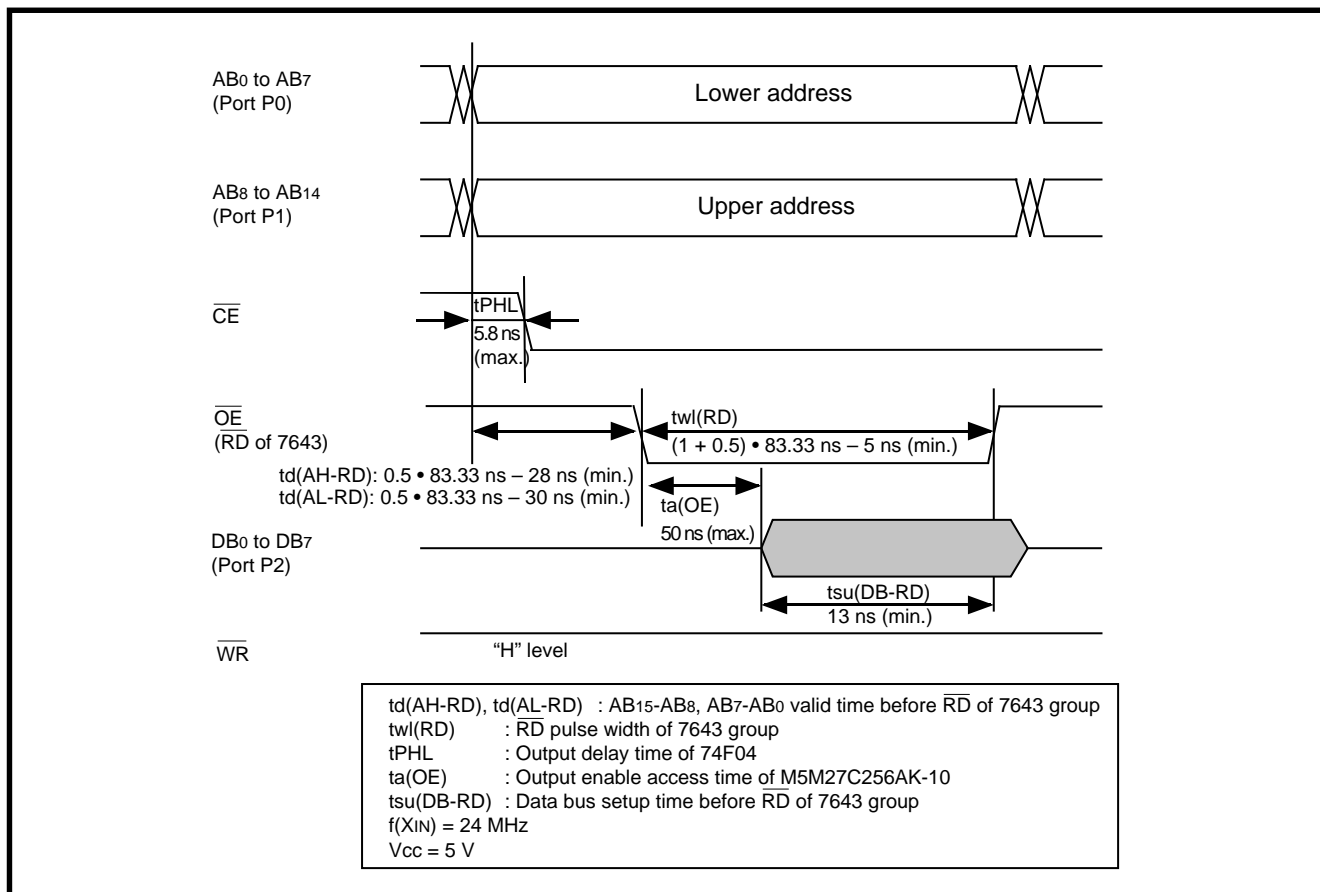


Fig. 2.8.12 Read cycle (OE access, EPROM)

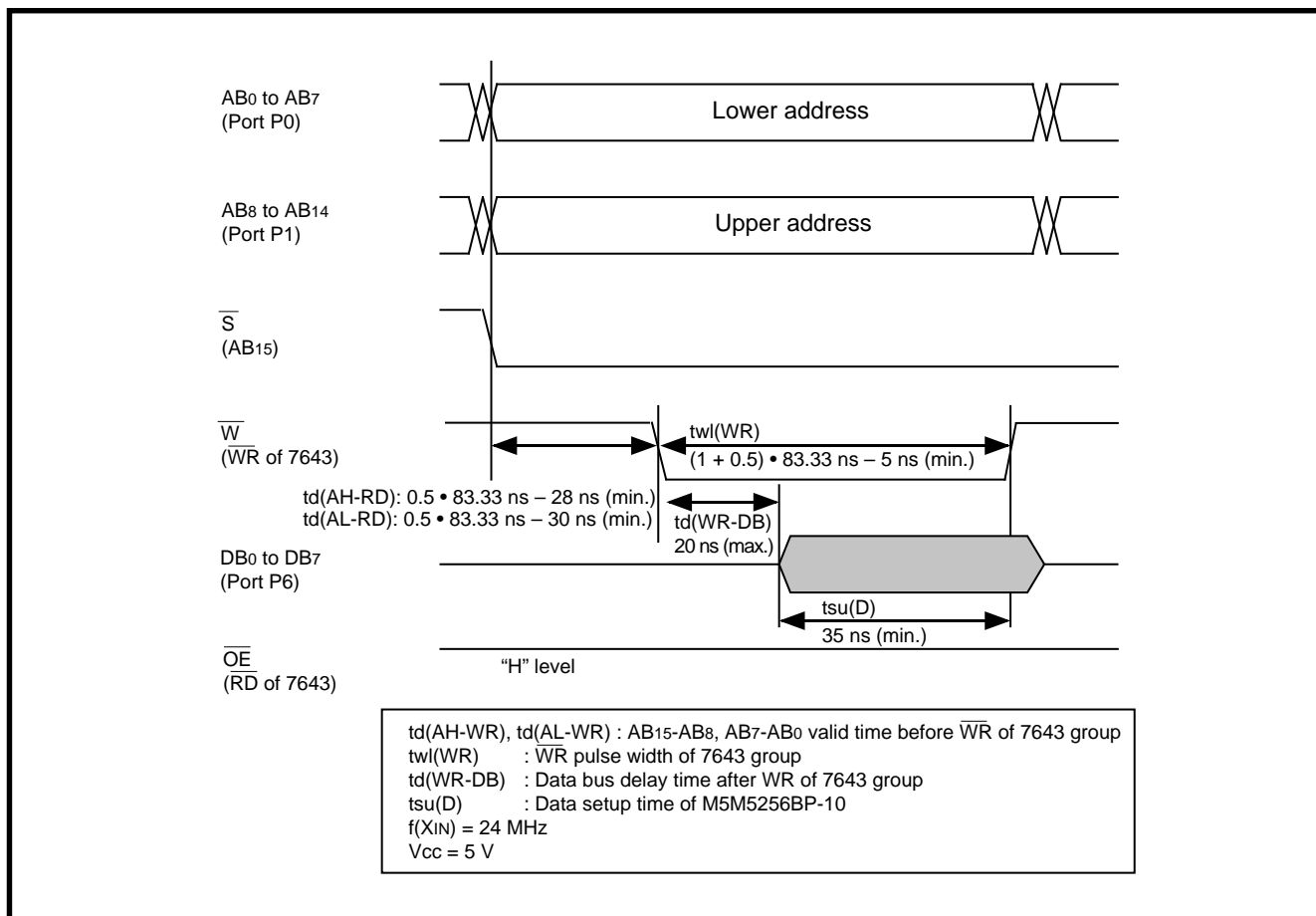


Fig. 2.8.13 Write cycle (W control, SRAM)

### 2.8.8 Notes on external devices connection

#### (1) Rewrite port P3 latch

In both memory expansion mode and microprocessor mode, ports P3<sub>1</sub> and P3<sub>2</sub> can be used as output ports. We recommend to use the **LDM** instruction or **STA** instruction to write to port P3 register (address 000E<sub>16</sub>). If using the Read-Modify-Write instruction (**SEB** instruction, **CLB** instruction) you will need to map a memory that the CPU can read from and write to.

#### [Reason]

The access to address 000E<sub>16</sub> is performed:

- Read from external memory
- Write to both port P3 latch and external memory.

It is because address 000E<sub>16</sub> is assigned on an external area In the memory expansion mode and microprocessor mode.

Accordingly, if a Read-Modify-Write instruction is executed to address 000E<sub>16</sub>, the external memory contents is read out and after its modification it will be written into both port P3 latch and an external memory. As a result, if an external memory is not allocated in address 000E<sub>16</sub> then, the MCU will read an undefined value and write its modified value into the port P3 latch. Therefore port P3 latch value will become undefined.

#### (2) overlap of internal and external memories

In the memory expansion mode, if the internal and external memory areas overlap, the internal memory becomes the valid memory for the overlapping area. When the CPU performs a read or a write operation on this overlapped area, the following things happen:

##### •Read

The CPU reads out the data in the internal memory instead of in the external memory. Note that, since the CPU will output a proper read signal, address signal, etc., the memory data at the respective address will appear on the external data bus.

##### •Write

The CPU writes data to both the internal and external memories.

#### (3) $\overline{RD}$ , $\overline{WR}$ pins

In the memory expansion mode or microprocessor mode, a read-out control signal is output from the  $\overline{RD}$  pin (P3<sub>6</sub>), and a write-in control signal is output from the  $\overline{WR}$  pin (P3<sub>7</sub>). "L" level is output from the  $\overline{RD}$  pin at CPU read-out and from the  $\overline{WR}$  pin at CPU write-in. These signals function for internal access and external access.

**(4) RDY function**

When using RDY function in usual connection, it does not operate at 12 MHz of  $\phi$  or faster.

**[Reason]**

$$td(\phi\text{-AH}) + tsu(\text{RDY-}\phi) = 31 \text{ ns (max.)} + 21 \text{ ns (min.)} = 52 \text{ ns.}$$

$$twh(\phi), twl(\phi) = 0.5 \times 83.33 - 5 = 36.665 \text{ ns}$$

Therefore, it becomes  $52 \text{ ns} > 36.665 \text{ ns}$ , so that the timing to enter RDY wait does not match.

However, if the timings can match owing to  $\overline{\text{RDY}}$  pin by "L" fixation and others, the RDY function can be used even at  $\phi = 12 \text{ MHz}$ . In this situation the slow memory wait always functions.

**(5) Wait function**

The Wait function is serviceable at accessing an external memory in the memory expansion mode and microprocessor mode. However, in these modes even if an external memory is assigned to addresses 0008<sub>16</sub> to 000F<sub>16</sub>, the Wait function cannot function to these areas.

**(6) Processor mode switch**

Note when the processor mode is switched by setting of the processor mode bits (b1, b0 of CPMA), that will immediately switch the accessible memory from external to internal or from internal to external. If this is done, the first cycle of the next instruction will be operated from the accidental memory.

To prevent this problem, follow the procedure below:

- (a) Duplicate the next instruction at the same address both in internal and external memories.
- (b) Switch from single-chip mode to memory expansion mode, jump to external memory, and then switch from memory expansion mode to microprocessor mode. (Because in general, the problem will not occur when switching the modes as long as the same memory is accessed after the switch.)
- (c) Load a simple program in RAM that switches the modes, jump to RAM and execute the program, then jump to the location of the code to run after the processor mode has switched.

## 2.9 Reset

### 2.9.1 Connection example of reset IC

Figure 2.9.1 shows the system example which switches to the RAM backup mode by detecting a drop of the system power source voltage with the INT interrupt.

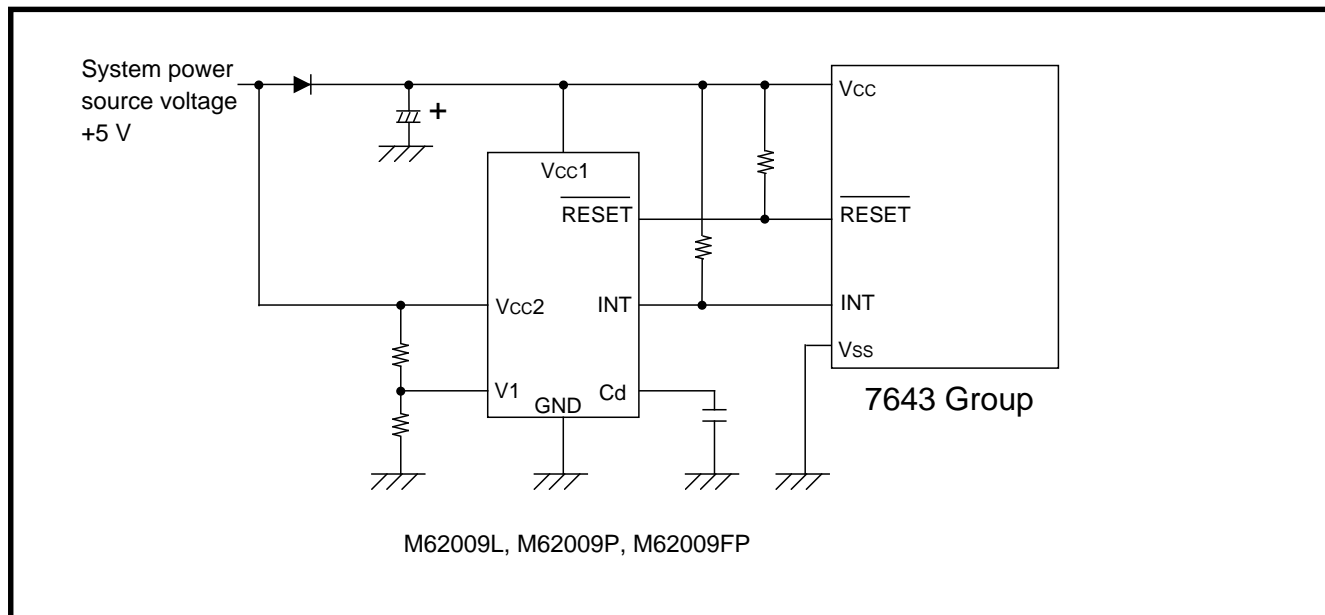


Fig. 2.9.1 RAM backup system

### 2.9.2 Notes on reset

#### (1) Connecting capacitor

In case where the  $\overline{\text{RESET}}$  signal rise time is long, connect a ceramic capacitor or others across the  $\overline{\text{RESET}}$  pin and the Vss pin. Use a 1000 pF or more capacitor for high frequency use. When connecting the capacitor, note the following :

- Make the length of the wiring which is connected to a capacitor as short as possible.
- Be sure to verify the operation of application products on the user side.

#### ● Reason

If the several nanosecond or several ten nanosecond impulse noise enters the  $\overline{\text{RESET}}$  pin, it may cause a microcomputer failure.

## 2.10 Clock generating circuit

This paragraph explains the registers setting method and the notes related to the clock generating circuit. Besides, two modes to realize less power dissipation due to the CPU and some peripherals halted are explained: Stop mode due to STP instruction, Wait mode due to WIT instruction.

### 2.10.1 Memory map

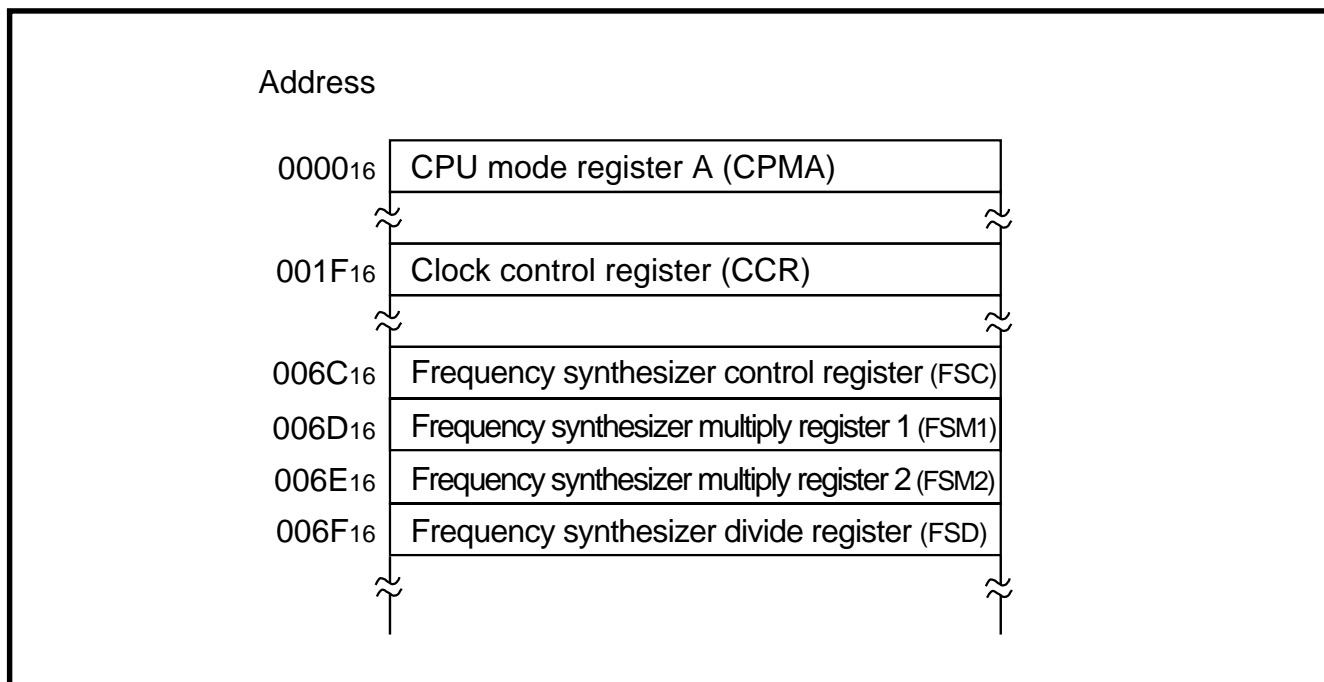


Fig. 2.10.1 Memory map of registers related to clock generating circuit

## 2.10.2 Related registers

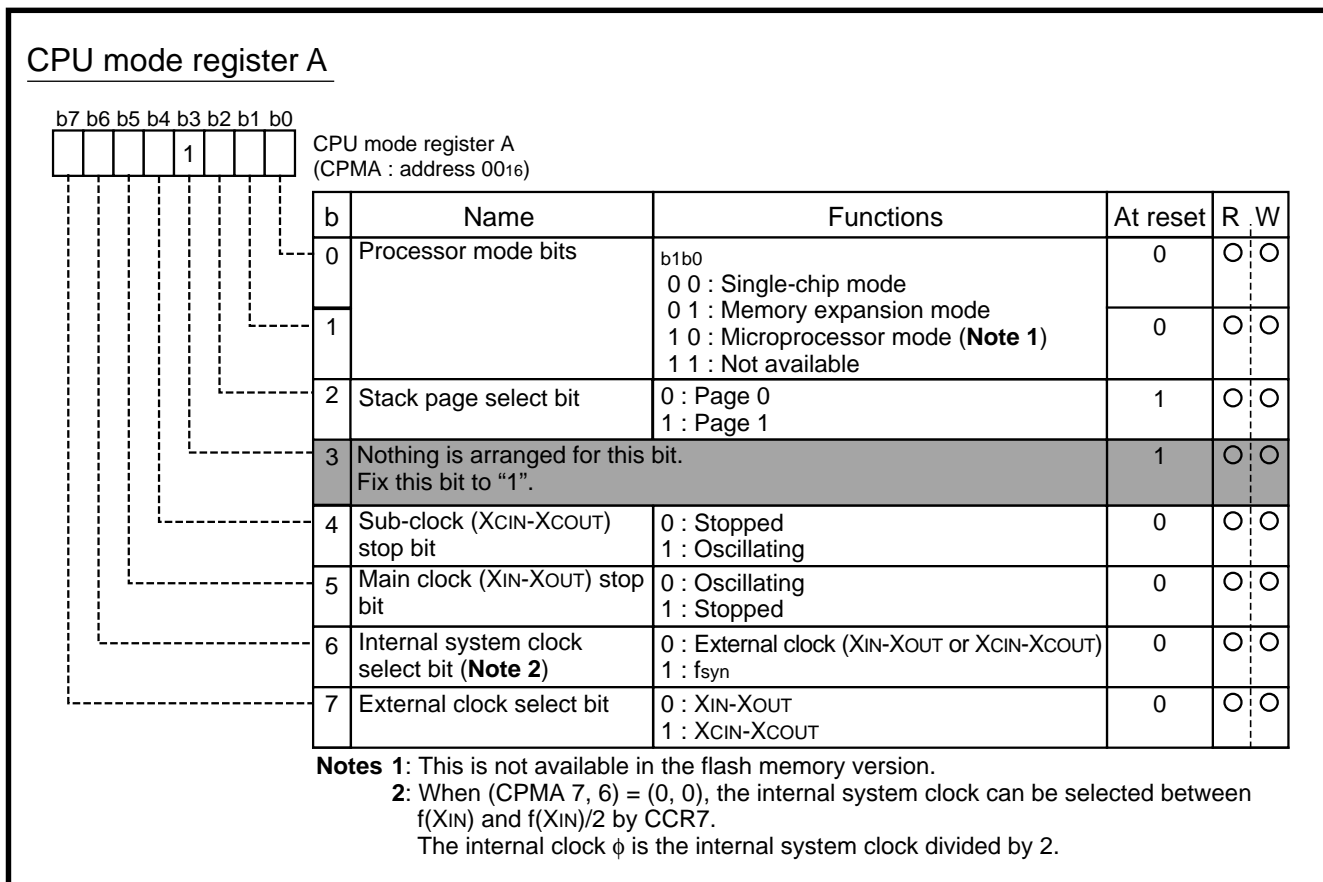


Fig. 2.10.2 Structure of CPU mode register A

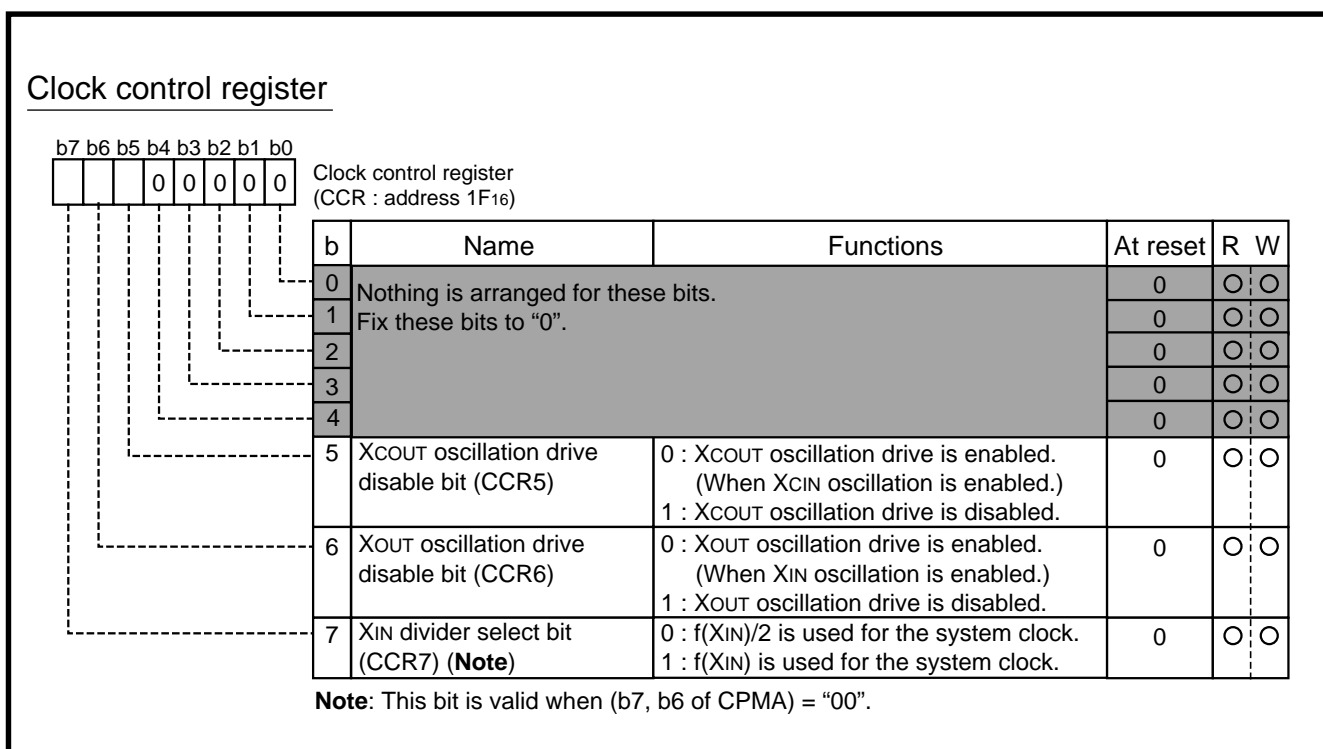


Fig. 2.10.3 Structure of Clock control register

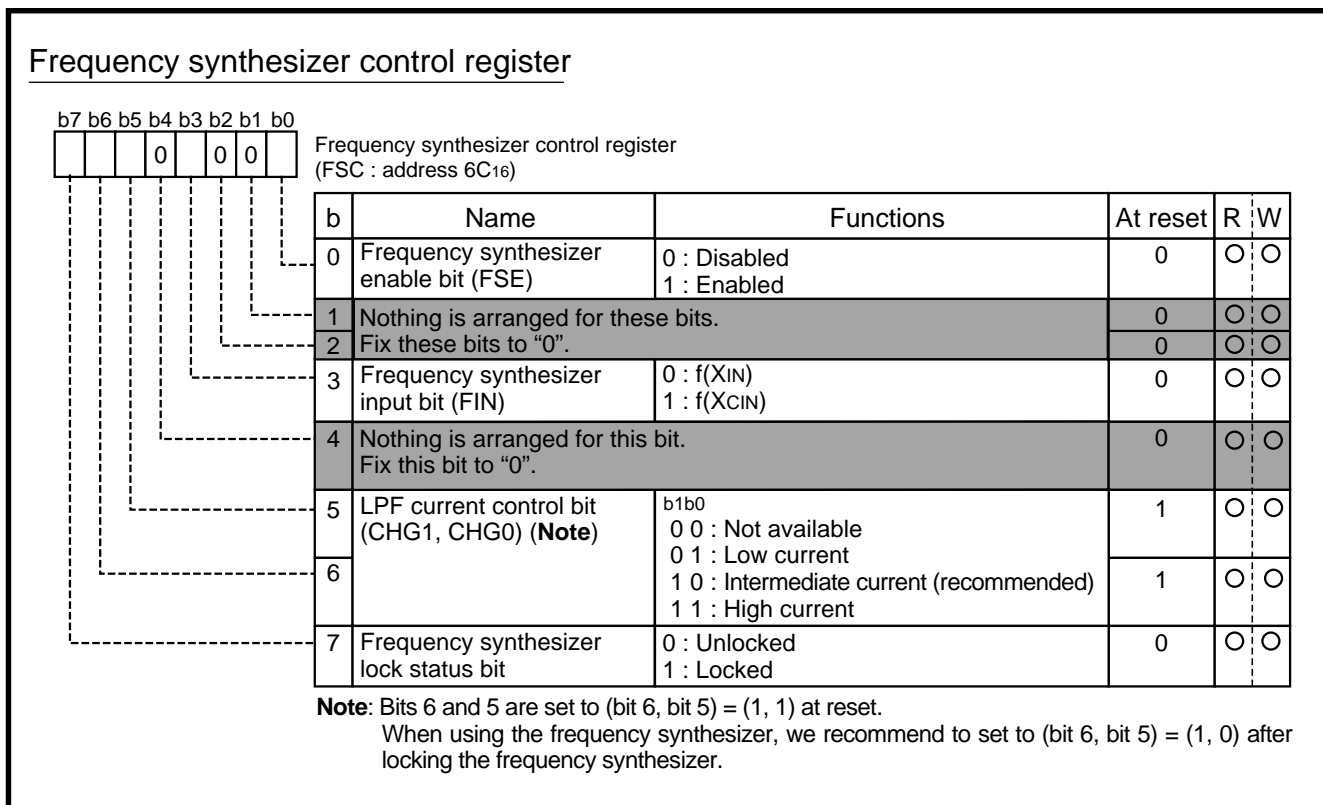


Fig. 2.10.4 Structure of Frequency synthesizer control register

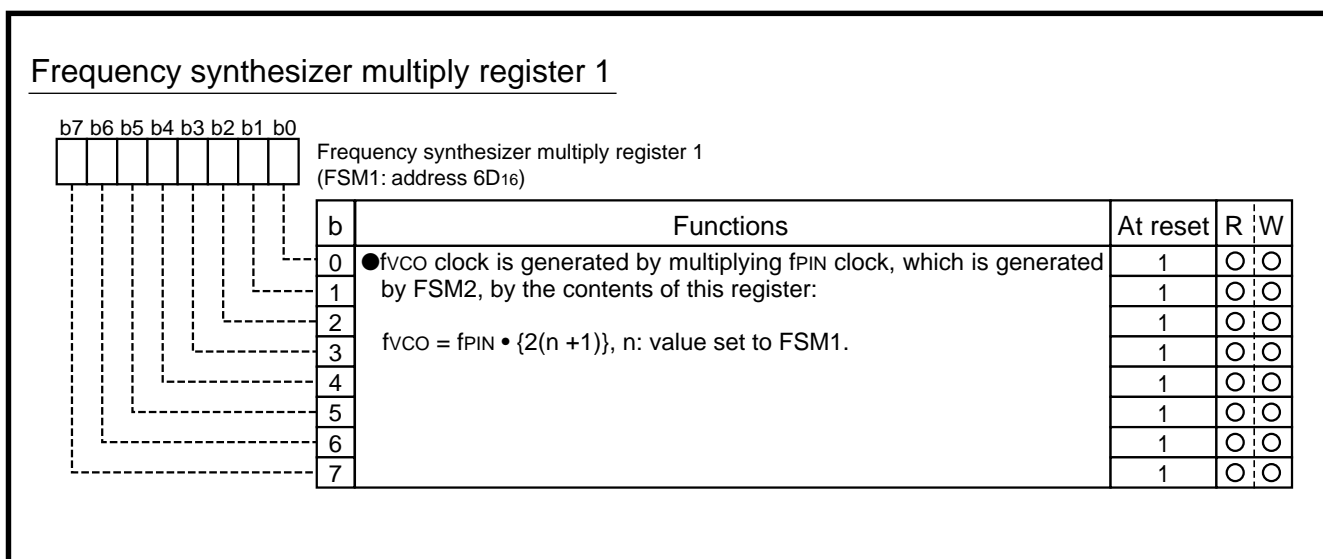


Fig. 2.10.5 Structure of Frequency synthesizer multiply register 1



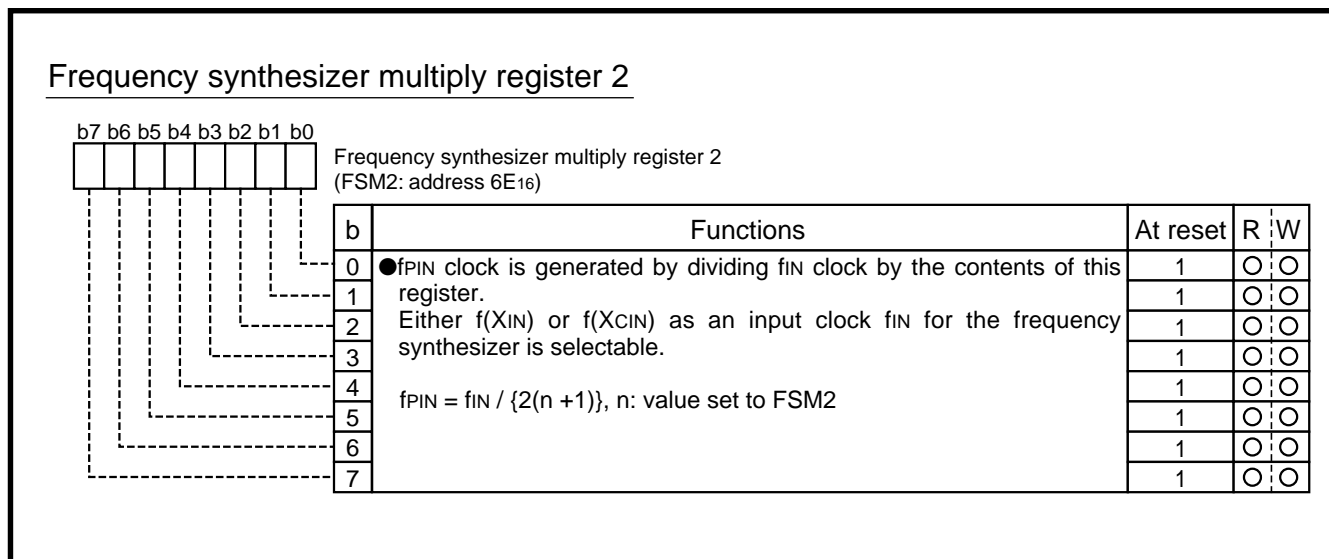


Fig. 2.10.6 Structure of Frequency synthesizer multiply register 2

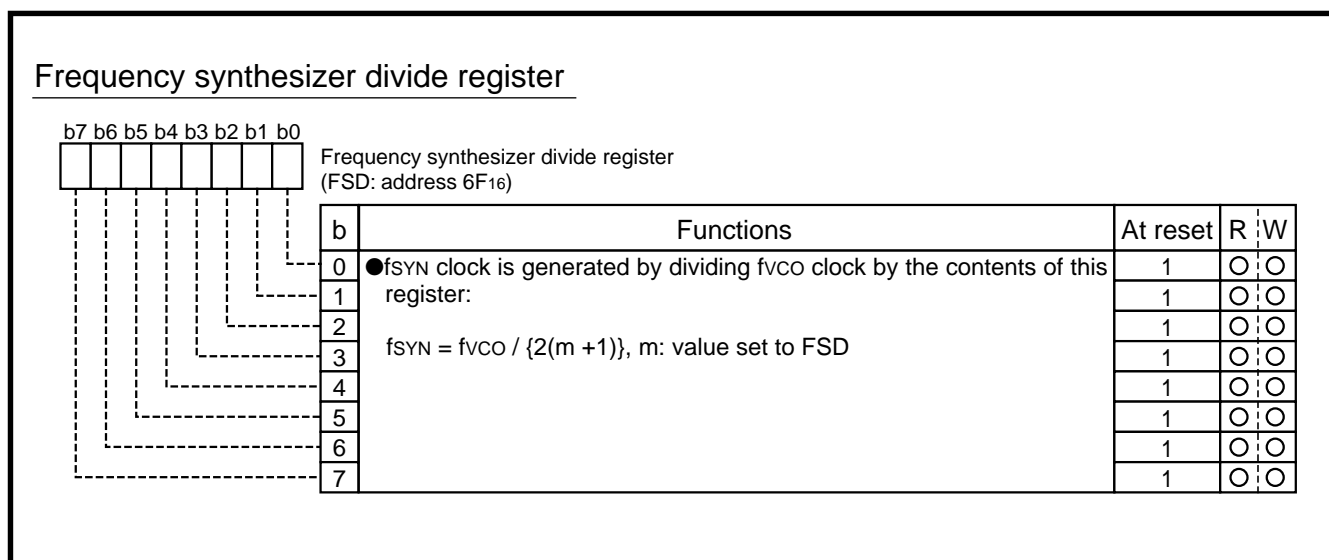


Fig. 2.10.7 Structure of Frequency synthesizer divide register

### 2.10.3 Stop mode

The Stop mode is set by executing the STP instruction. In Stop mode, the oscillation of both clocks ( $X_{IN}$ – $X_{OUT}$ ,  $X_{CIN}$ – $X_{COUT}$ ) stop and the internal clock  $\phi$  stops at the “H” level. The CPU stops and peripheral units stop operating. As a result, power dissipation is reduced.

#### (1) State in Stop mode

Table 2.10.1 shows the state in Stop mode.

**Table 2.10.1 State in Stop mode**

Item	State in Stop mode
Oscillation	Stopped.
CPU	Stopped.
Internal clock $\phi$	Stopped at “H” level.
I/O ports	Retains the state at the STP instruction execution.
Timer	Stopped.
UART	Stopped.
DMAC	Stopped.
Serial I/O	When using internal synchronous clock: Stopped. When using external synchronous clock: Operating.
USB	Stopped.
RAM	Retained.
SFR	Retained (except for Timer 1, Timer 2).
CPU registers	Retained: Accumulator, Index register X, Index register Y, Stack pointer, Program counter, Processor status register.

#### (2) Release of Stop mode

The Stop mode is released by a reset input or by the occurrence of an interrupt request.

These interrupt sources can be used for restoration:

- INT<sub>0</sub>, INT<sub>1</sub>
- Serial I/Os using an external clock
- Key-on wake-up
- USB function resume

However, when using any of these interrupt requests for restoration from Stop mode, in order to enable the selected interrupt, set the following conditions before execution of STP instruction.

[Necessary register setting]

- ① Timer 1 interrupt enable bit (b6 of ICONB) = “0” (interrupt disabled)
- ② Timer 2 interrupt enable bit (b7 of ICONB) = “0” (interrupt disabled)
- ③ Timer 1 interrupt request bit (b6 of IREQB) = “0” (no interrupt request issued)
- ④ Timer 2 interrupt request bit (b7 of IREQB) = “0” (no interrupt request issued)
- ⑤ Interrupt request bit of interrupt source to be used for restoration = “0” (no interrupt request issued)
- ⑥ Interrupt enable bit of interrupt source to be used for restoration = “1” (interrupts enabled)
- ⑦ Interrupt disable flag I = “0” (interrupt enabled)

#### (3) Notes on STP instruction

- Execution of STP instruction clears the timer 123 mode register (address 29<sub>16</sub>) except bit 4 to “0”.
- When using  $f_{SYN}$  as the internal system clock, switch to  $f(X_{IN})$  or  $f(X_{CIN})$  before execution of STP instruction.
- Execution of STP instruction clears bit 7 of clock control register to “0” ( $f(X_{IN})/2$ ).

### 2.10.4 Wait mode

The Wait mode is set by execution of the WIT instruction. In Wait mode, oscillation continues, but the internal clock  $\phi$  stops at the "H" level.

The CPU stops, but most of the peripheral units continue operating.

#### (1) State in Wait mode

Table 2.10.2 shows the state in Wait mode.

**Table 2.10.2 State in Wait mode**

Item	State in Wait mode
Oscillation	Operating.
CPU	Stopped.
Internal clock $\phi$	Stopped at "H" level.
I/O ports	Retains the state at the WIT instruction execution.
Timer	Operating.
UART	Operating.
DMAC	Stopped.
Serial I/O	Operating.
USB	Operating.
RAM	Retained.
SFR	Retained.
CPU registers	Retained: Accumulator, Index register X, Index register Y, Stack pointer, Program counter, Processor status register.

#### (2) Release of wait mode

The Wait mode is released by reset input or by the occurrence of an interrupt request.

In Wait mode oscillation is continued, so that an instruction can be executed immediately after the Wait mode is released.

These interrupt sources can be used for restoration:

- INT<sub>0</sub>, INT<sub>1</sub>
- Timers
- Serial I/Os
- UART
- DMAC
- Key-on wake-up
- USB function

However, when using any of these interrupt requests for restoration from Stop mode, in order to enable the selected interrupt, set the following conditions before execution of WIT instruction.

[Necessary register setting]

- ① Interrupt request bit of interrupt source to be used for restoration = "0" (no interrupt request issued)
- ② Interrupt enable bit of interrupt source to be used for restoration = "1" (interrupts enabled)
- ③ Interrupt disable flag I = "0" (interrupt enabled)

### 2.10.5 Clock generating circuit application examples

#### (1) Status transition during power failure

**Outline:** The clock is counted up every one second by using the timer interrupt during a power failure.

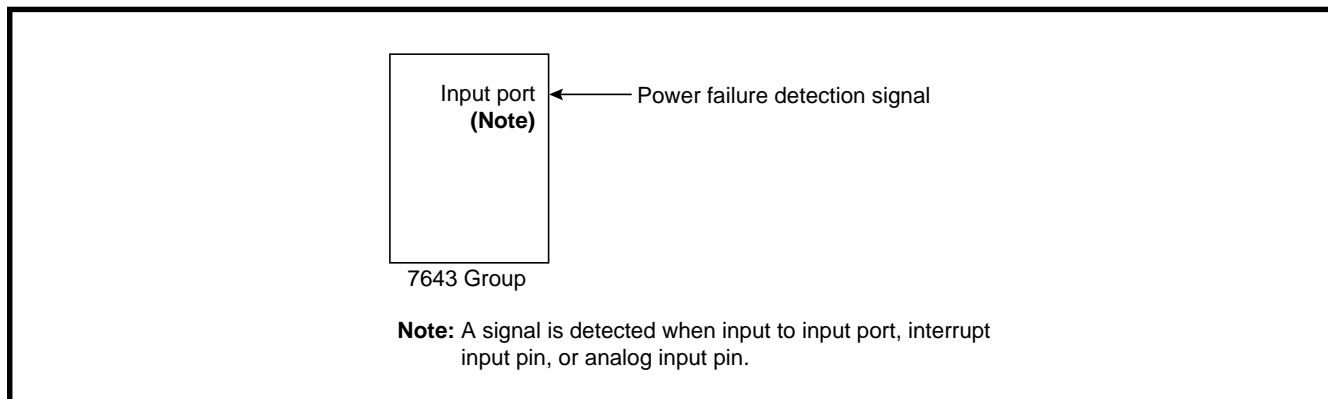


Fig. 2.10.8 Connection diagram

**Specifications:**

- Reducing power dissipation as low as possible while maintaining clock function
- Clock:  $f(X_{IN}) = 4.19 \text{ MHz}$ ,  $f(X_{CIN}) = 32.768 \text{ kHz}$
- Port processing

Input port: Fixed to "H" or "L" level on the external

Output port: Fixed to output level that does not cause current flow to the external

(Example) When a circuit turns on LED at "L" output level, fix the output level to "H".

I/O port: Input port → Fixed to "H" or "L" level on the external

Output port → Output of data that does not consume current

Figure 2.10.9 shows the status transition diagram during power failure and Figure 2.10.10 shows the setting of relevant registers.

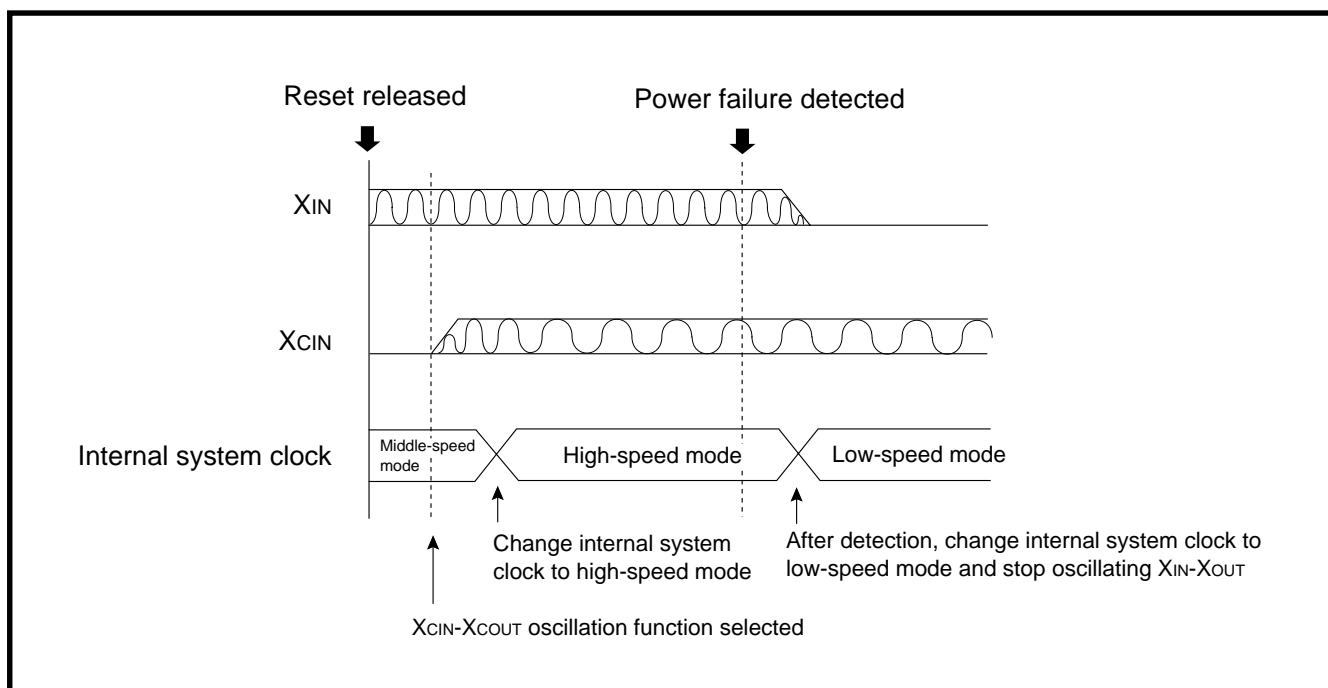


Fig. 2.10.9 Status transition diagram during power failure

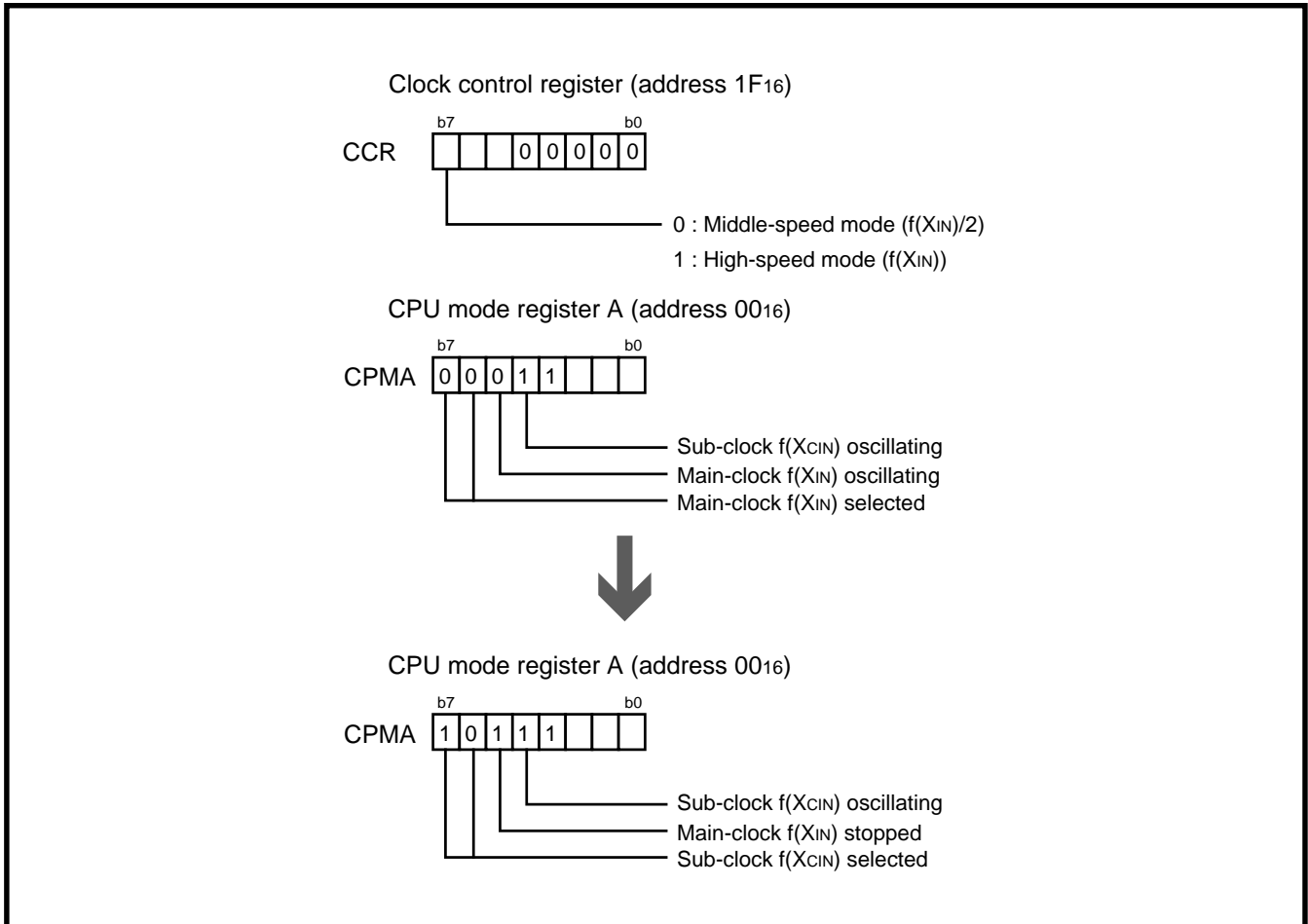


Fig. 2.10.10 Setting of relevant registers

**Control procedure:** Set the relevant registers in the order shown below to prepare for a power failure.

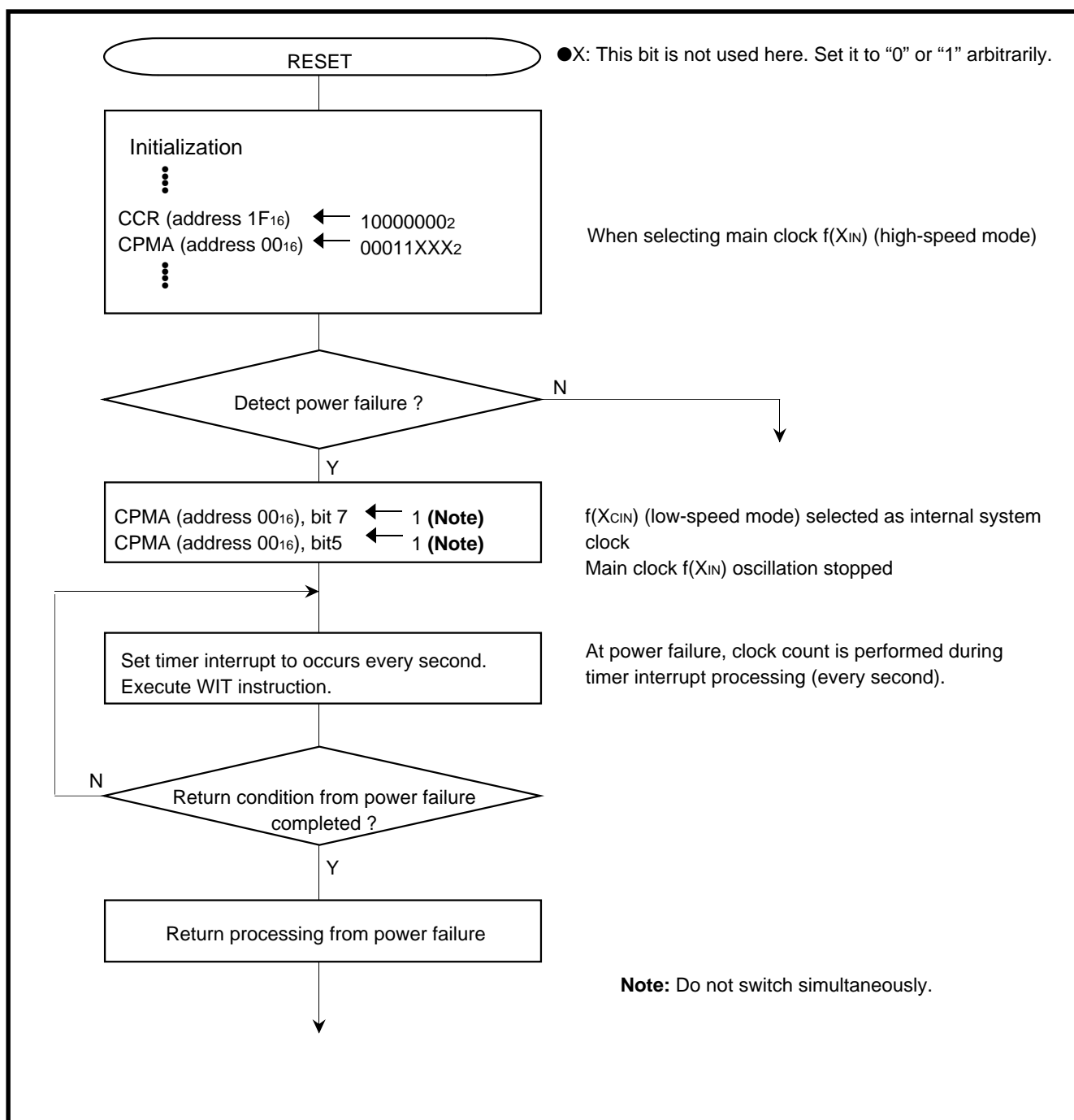


Fig. 2.10.11 Control procedure

**(2) Counting without clock error during power failure**

**Outline:** It keeps counting without clock error during a power failure.

**Specifications:** •Reducing power consumption as low as possible while maintaining clock function

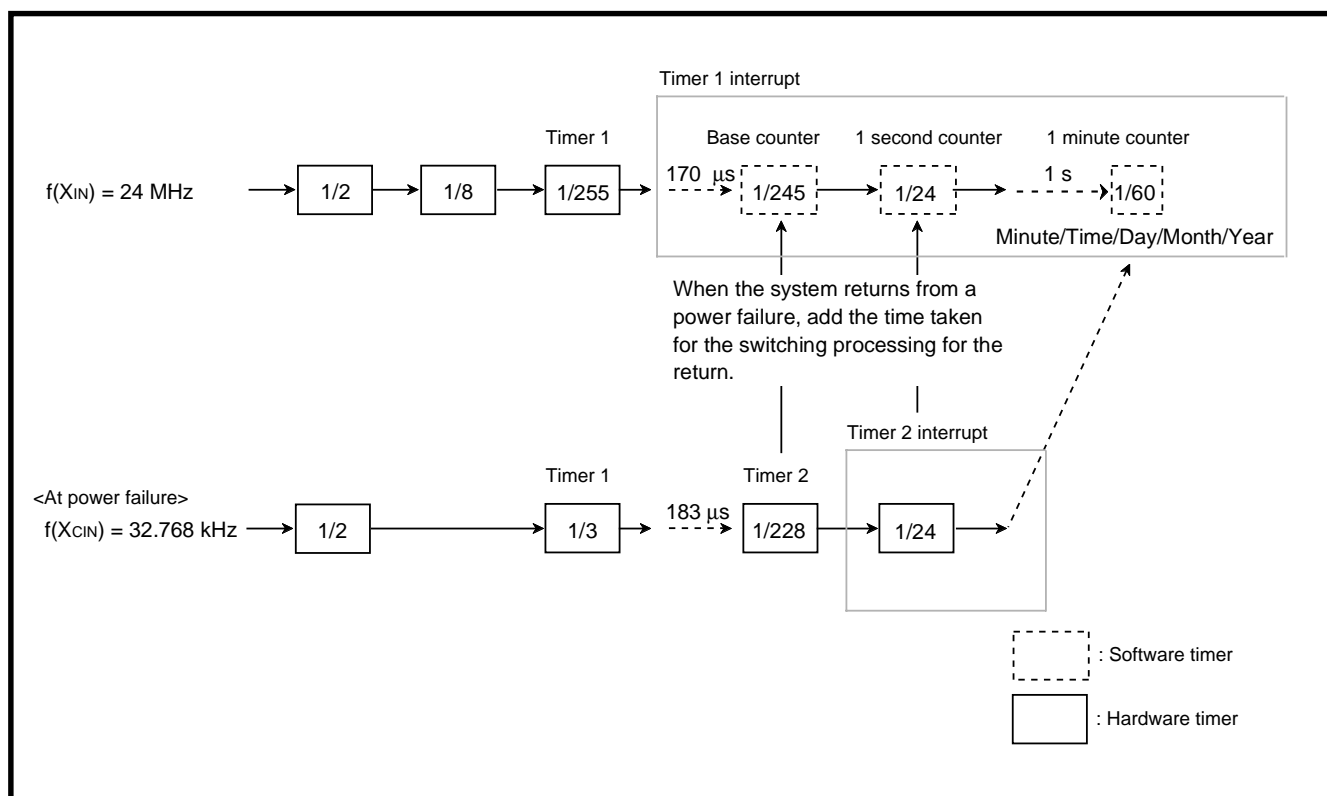
•Clock:  $f(X_{IN}) = 24 \text{ MHz}$

•Sub clock:  $f(X_{CIN}) = 32.768 \text{ kHz}$

•Use of Timer 2 interrupt

For the peripheral circuit and the status transition during a power failure, refer to Figures 2.10.8 and 2.10.9.

Figure 2.10.12 shows the structure of clock counter, Figures 2.10.13 and 2.10.14 show the setting of relevant registers.



**Fig. 2.10.12 Structure of clock counter**

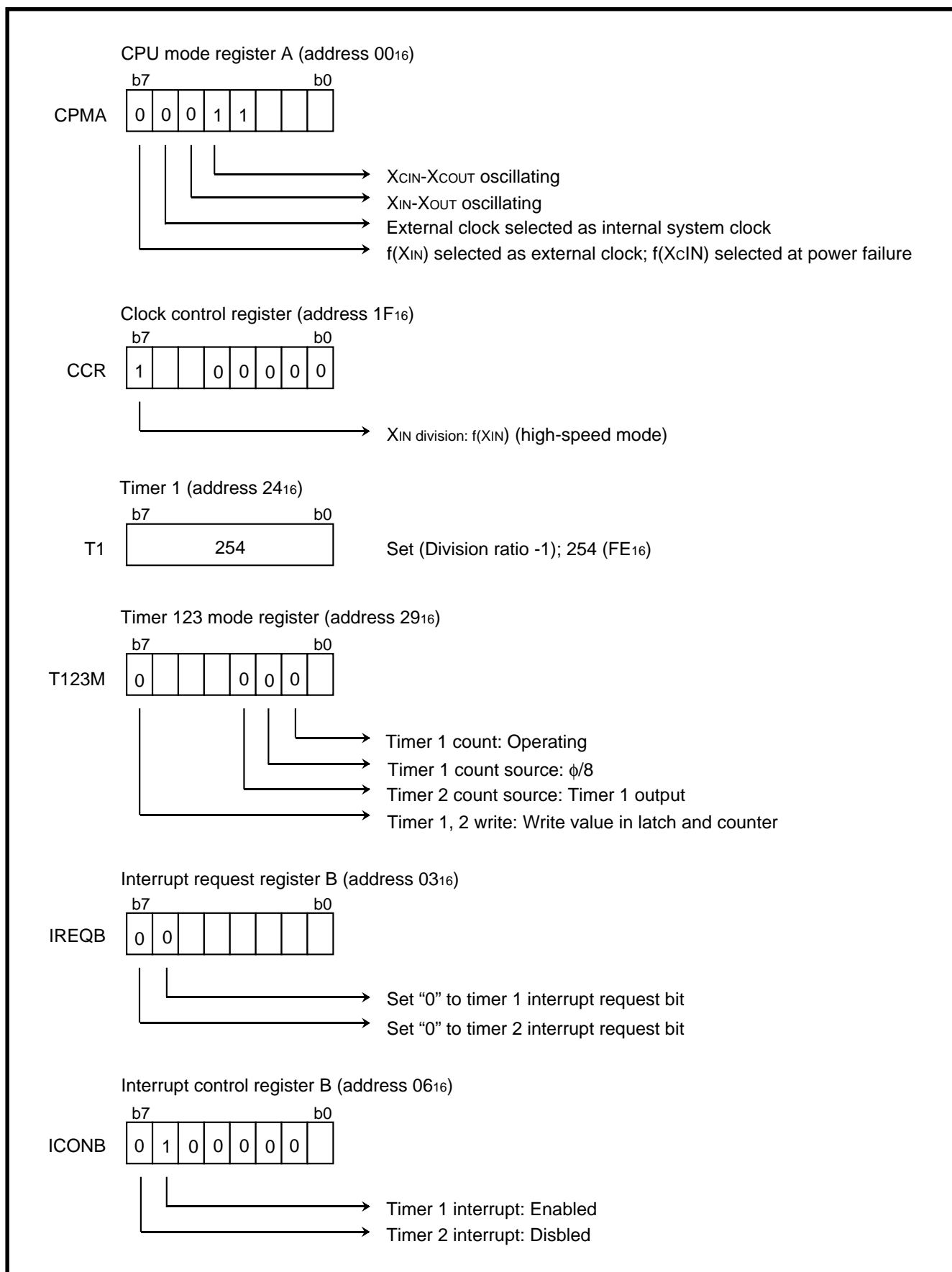


Fig. 2.10.13 Initial setting of relevant registers



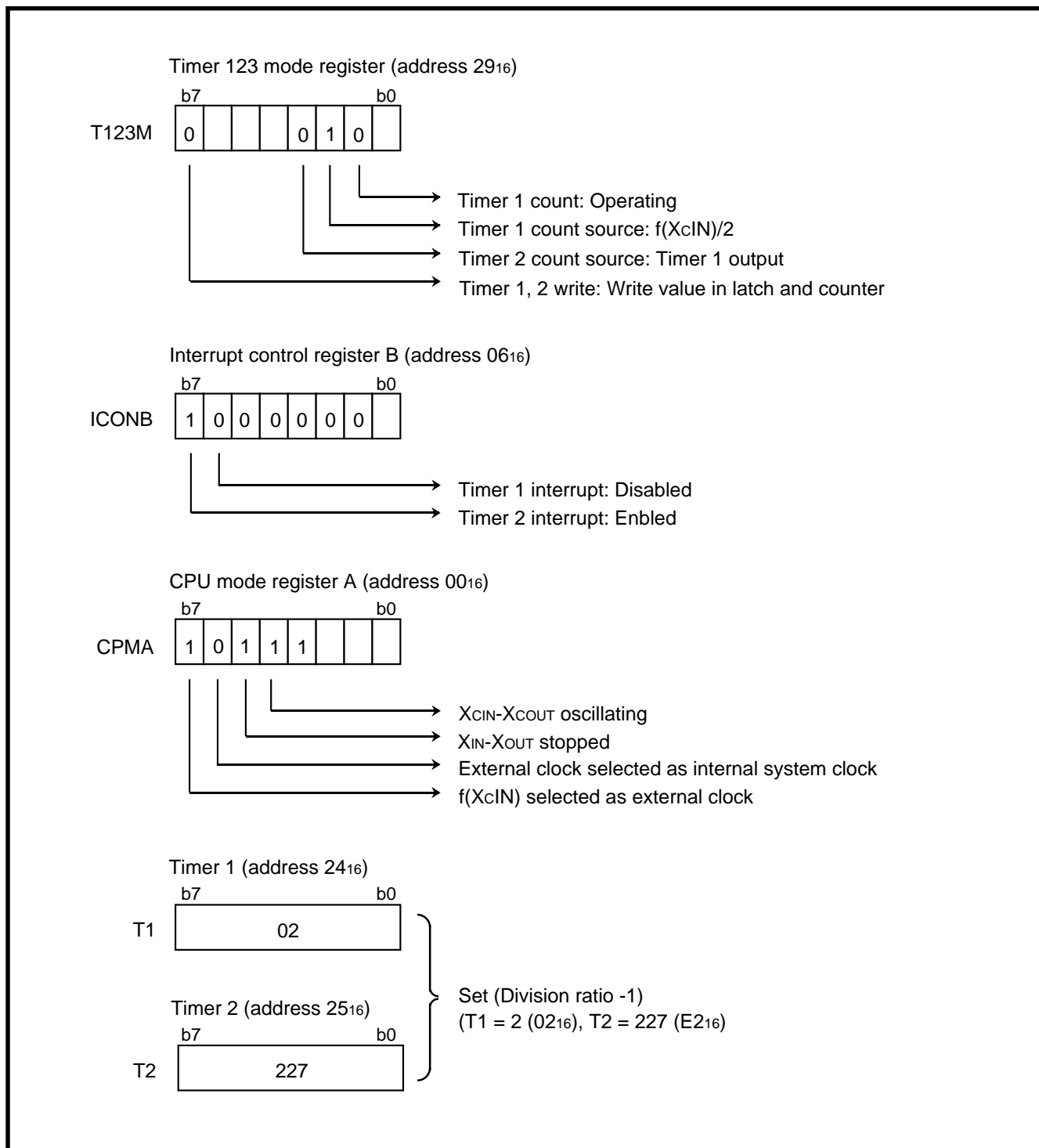


Fig. 2.10.14 Setting of relevant registers after detecting power failure

**Control procedure:** Set the relevant registers in the order shown below to prepare for a power failure.

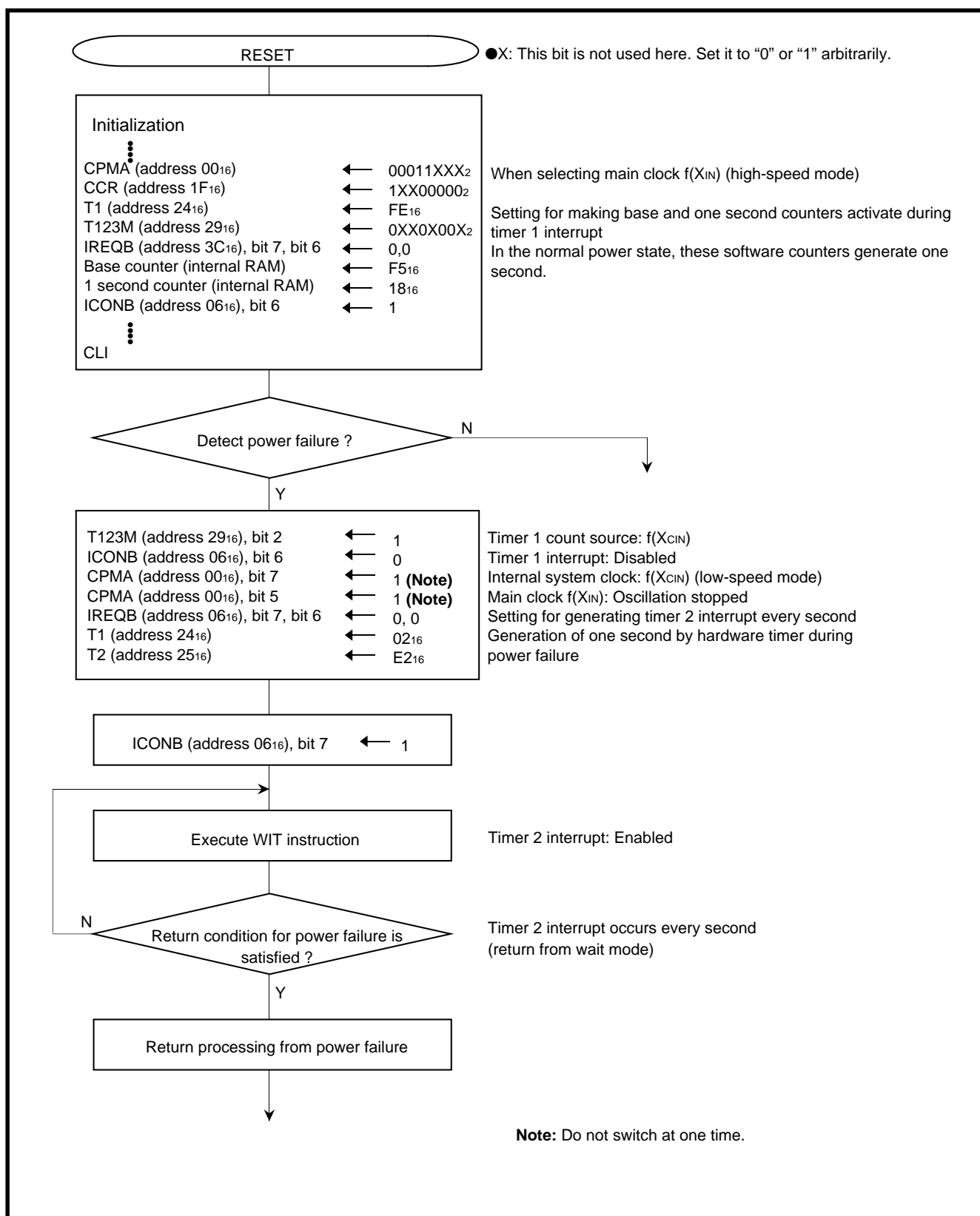


Fig. 2.10.15 Control procedure (1)

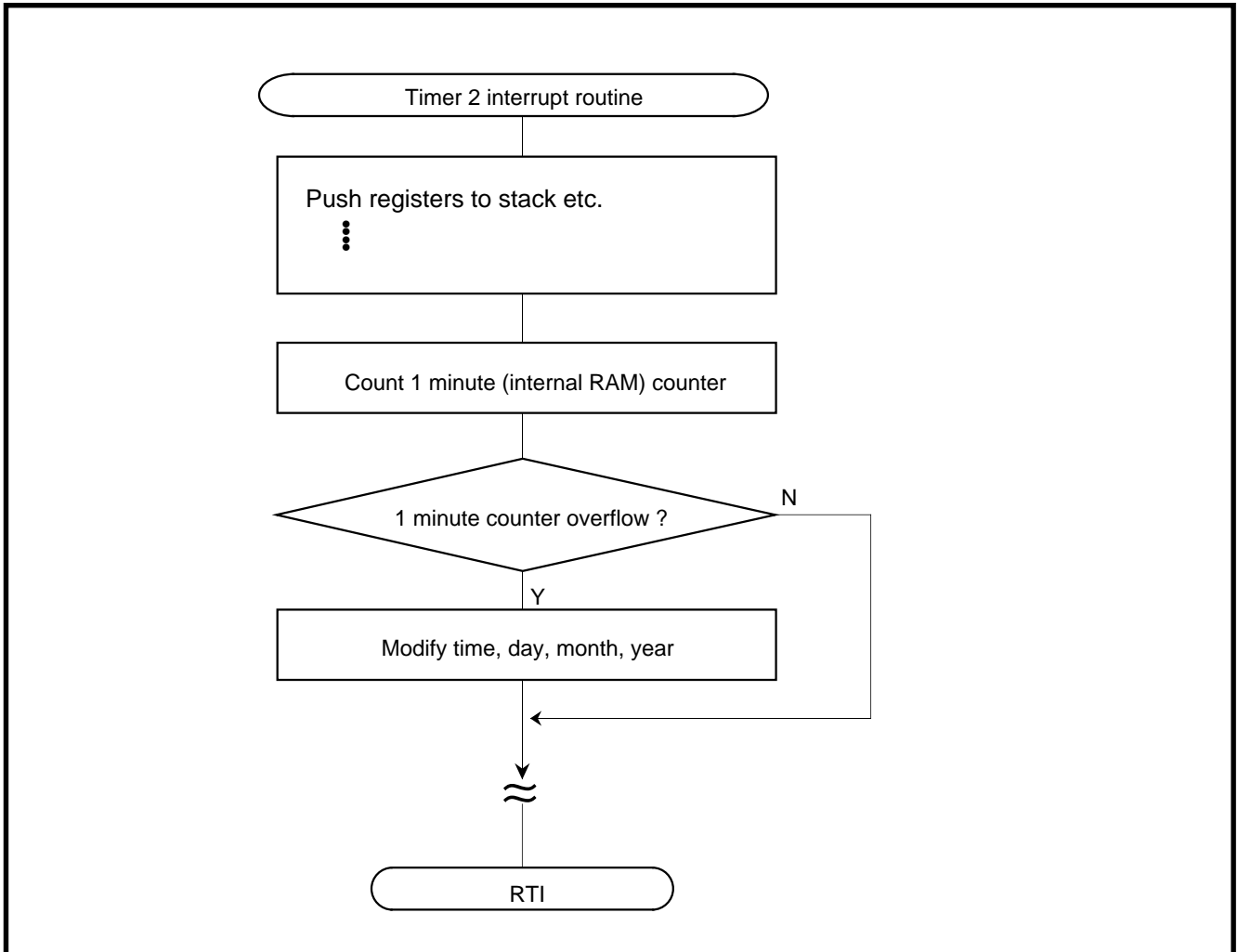


Fig. 2.10.16 Control procedure (2)

# **CHAPTER 3**

---

## **APPENDIX**

- 3.1 Electrical characteristics**
- 3.2 Standard characteristics**
- 3.3 Notes on use**
- 3.4 Countermeasures against noise**
- 3.5 Control registers**
- 3.6 Package outline**
- 3.7 Machine instructions**
- 3.8 List of instruction code**
- 3.9 SFR memory area**
- 3.10 Pin configuration**

## 3.1 Electrical characteristics

### 3.1.1 Absolute maximum ratings

**Table 3.1.1 Absolute maximum ratings**

Symbol	Parameter	Conditions	Ratings	Unit	
VCC	Power source voltage	All voltages are based on Vss. Output transistors are cut off.	-0.3 to 6.5	V	
AVCC	Analog power source voltage AVcc, Ext.Cap		-0.3 to VCC+0.3	V	
Vi	Input voltage P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87		-0.3 to VCC+0.3	V	
Vi	Input voltage RESET, XIN, XCIN		-0.3 to VCC+0.3	V	
Vi	Input voltage CNVss		Mask ROM version	-0.3 to VCC + 0.3	V
			Flash memory version	-0.3 to 6.5	V
Vi	Input voltage USB D+, USB D–		-0.5 to 3.8	V	
Vo	Output voltage P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87, XOUT, Xcout, LPF	-0.3 to VCC+0.3	V		
Vo	Output voltage USB D+, USB D–, Ext. Cap	-0.5 to 3.8	V		
Pd	Power dissipation ( <b>Note</b> )	Ta = 25°C	750	mW	
Topr	Operating temperature		-20 to 70	°C	
Tstg	Storage temperature		-40 to 125	°C	

**Note:** The maximum power dissipation depends on the MCU's power dissipation and the specific heat consumption of the package.

### 3.1.2 Recommended operating conditions (In Vcc = 5 V)

**Table 3.1.2 Recommended operating conditions**

(Vcc = 4.15 to 5.25 V, Vss = 0 V, Ta = -20 to 70°C, unless otherwise noted)

Symbol	Parameter	Limits			Unit
		Min.	Typ.	Max.	
VCC	Power source voltage	4.15	5.0	5.25	V
AVcc	Analog reference voltage	4.15	5.0	Vcc	V
VSS	Power source voltage		0		V
AVSS	Analog reference voltage		0		V
VIH	"H" input voltage P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87	0.8Vcc		Vcc	V
VIH	"H" input voltage (Selecting VIH level input) P20–P27	0.5Vcc		Vcc	V
VIH	"H" input voltage RESET, XIN, XCIN, CNVss	0.8Vcc		Vcc	V
VIH	"H" input voltage USB D+, USB D–	2.0		3.8	V
VIL	"L" input voltage P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87	0		0.2Vcc	V
VIL	"L" input voltage (Selecting VIH level input) P20–P27	0		0.16Vcc	V
VIL	"L" input voltage RESET, XIN, XCIN, CNVss	0		0.2Vcc	V
VIL	"L" input voltage USB D+, USB D–			0.8	V
ΣIOH(peak)	"H" total peak output current (Note 1) P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87			–80	mA
ΣIOL(peak)	"L" total peak output current (Note 1) P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87			80	mA
ΣIOH(avg)	"H" total average output current (Note 1) P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87			–40	mA
ΣIOL(avg)	"L" total average output current (Note 1) P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87			40	mA
IOH(peak)	"H" peak output current (Note 2) P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87			–10	mA
IOL(peak)	"L" peak output current (Note 2) P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87			10	mA
IOH(avg)	"H" average output current (Note 3) P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87			–5.0	mA
IOL(avg)	"L" average output current (Note 3) P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87			5.0	mA
f(XIN)	Main clock input frequency (Notes 4, 5)	1		24	MHz
f(XCIN)	Sub-clock input frequency (Notes 4, 6)		32.768	50/5.0	kHz/MHz

**Notes 1:** The total peak output current is the peak value of the peak currents flowing through all the applicable ports. The total average output current is the average value measured over 100 ms flowing through all the applicable ports.

**2:** The peak output current is the peak current flowing in each port.

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

**4:** The duty of oscillation frequency is 50 %.

**5:** Connect a ceramic resonator or a quartz-crystal oscillator between the XIN and XOUT pins. Its maximum oscillation frequency must be 24 MHz. However, make sure to set  $\phi$  to 12 MHz or slower. More faster clocks are required as the f(XIN) when using the frequency synthesizer as possible.

**6:** Connect a ceramic resonator or a quartz-crystal oscillator between the XCIN and XCOU pins. Its maximum oscillation frequency must be 50 kHz. Input an external clock having 5 MHz frequency (max.) from the XCIN pin.

## 3.1.3 Electrical characteristics (In Vcc = 5 V)

Table 3.1.3 Electrical characteristics (1)

(Vcc = 4.15 to 5.25 V, Vss = 0 V, Ta = -20 to 70°C, unless otherwise noted)

Symbol	Parameter	Test conditions	Limits			Unit
			Min.	Typ.	Max.	
VOH	"H" output voltage P00-P07, P10-P17, P20-P27, P30-P37, P40-P44, P50-P57, P60-P67, P70-P74, P80-P87	IOH = -10 mA	VCC-2.0			V
VOH	"H" output voltage USB D+, USB D-	USB+, and USB- pins pull-down via a resistor of 15 kΩ ± 5 %  USB+ pin pull-up to Ext. Cap. pin via a resistor of 1.5 kΩ ± 5 %	2.8		3.6	V
VOL	"L" output voltage P00-P07, P10-P17, P20-P27, P30-P37, P40-P44, P50-P57, P60-P67, P70-P74, P80-P87	IOL = 10 mA			2.0	V
VOL	"L" output voltage USB D+, USB D-	USB+, and USB- pins pull-down via a resistor of 15 kΩ ± 5 %  USB+ pin pull-up to Ext. Cap. pin via a resistor of 1.5 kΩ ± 5 %			0.3	V
VT+–VT-	Hysteresis INT0, INT1, RDY, $\overline{\text{HOLD}}$ , P20-P27 (Note 1)			0.5		V
VT+–VT-	Hysteresis URXD, SCLK, SRXD, SRDY, CTS			0.5		V
VT+–VT-	Hysteresis $\overline{\text{RESET}}$			0.5		V
IiH	"H" input current P00-P07, P10-P17, P20-P27, P30-P37, P40-P44, P50-P57, P60-P67, P70-P74, P80-P87	Vi = VCC			5.0	μA
IiH	"H" input current $\overline{\text{RESET}}$ , CNVss				5.0	μA
IiH	"H" input current XIN			9.0	20	μA
IiH	"H" input current XCIN				5.0	μA
IiL	"L" input current P00-P07, P10-P17, P30-P37, P40-P44, P50-P57, P60-P67, P70-P74, P80-P87	Vi = Vss			-5.0	μA
IiL	"L" input current $\overline{\text{RESET}}$				-5.0	μA
IiL	"L" input current CNVss				-20	μA
IiL	"L" input current XIN			-9.0	-20	μA
IiL	"L" input current XCIN				-5.0	μA
IiL	"L" input current P20-P27	Vi = Vss Pull-ups "off"			-5.0	μA
		VCC = 5.0 V, Vi = Vss Pull-ups "on"	-30	-65	-140	μA
VRAM	RAM hold voltage	When clock is stopped	2.0		5.25	V

Note 1: This spec is hysteresis of key input interrupt.

In  $V_{CC} = 5\text{ V}$ **Table 3.1.4 Electrical characteristics (2)**(V<sub>CC</sub> = 4.15 to 5.25 V, V<sub>SS</sub> = 0 V, T<sub>a</sub> = -20 to 70°C, unless otherwise noted)

Symbol	Parameter	Test conditions	Limits			Unit
			Min.	Typ.	Max.	
I <sub>CC</sub>	Power source current (Output transistor is isolated.)	Normal mode ( <b>Note 1</b> ) f(X <sub>IN</sub> ) = 24 MHz, φ = 12 MHz USB operating Frequency synthesizer ON		40	90	mA
		Wait mode ( <b>Note 2</b> ) f(X <sub>IN</sub> ) = 24 MHz, φ = 12 MHz USB block enabled, USB clock stopped, Frequency synthesizer ON		5.0	11	mA
		Wait mode ( <b>Note 3</b> ) f(X <sub>CIN</sub> ) = 32 kHz, φ = 16 kHz USB block disabled Frequency synthesizer OFF USB transceiver DC-DC converter OFF			10	μA
		Stop mode USB transceiver DC-DC converter ON Low current mode (USBC3 = "1")		100	250	μA
		Stop mode USB transceiver DC-DC converter OFF T <sub>a</sub> = 25 °C			1.0	μA
		Stop mode USB transceiver DC-DC converter OFF T <sub>a</sub> = 70 °C			10	μA

**<Test conditions>****Notes 1:** Operating in single-chip mode

Clock input from X<sub>IN</sub> pin (X<sub>OUT</sub> oscillator stopped)  
 USB operating with USB transceiver DC-DC converter enabled  
 Operating functions: Frequency synthesizer, CPU, UART, DMAC, Timers  
 Disabled functions: Serial I/O

**2:** Operating in single-chip mode with Wait mode

Clock input from X<sub>IN</sub> pin (X<sub>OUT</sub> oscillator stopped)  
 USB suspended due to USB clock stopped with USB transceiver DC-DC converter enabled  
 Operating functions: Frequency synthesizer, Timers  
 Disabled functions: CPU, UART, DMAC and Serial I/O

**3:** Operating in single-chip mode with Wait mode

X<sub>IN</sub> - X<sub>OUT</sub> oscillator stopped  
 Clock input from X<sub>CIN</sub> pin (X<sub>COUT</sub> oscillator stopped)  
 USB stopped, USB clock stopped and USB transceiver DC-DC converter disabled  
 Operating functions: Timers  
 Disabled functions: Frequency synthesizer, CPU, UART, DMAC and Serial I/O



### 3.1.4 Timing requirements (In $V_{CC} = 5\text{ V}$ )

**Table 3.1.5 Timing requirements**

 ( $V_{CC} = 4.15$  to  $5.25\text{ V}$ ,  $V_{SS} = 0\text{ V}$ ,  $T_a = -20$  to  $70^\circ\text{C}$ , unless otherwise noted)

Symbol	Parameter	Limits			Unit
		Min.	Typ.	Max.	
$t_w(\overline{\text{RESET}})$	Reset input "L" pulse width	2			$\mu\text{s}$
$t_c(\text{XIN})$	Main clock input cycle time ( <b>Note</b> )	41.66			ns
$t_{WH}(\text{XIN})$	Main clock input "H" pulse width	$0.4 \cdot t_c(\text{XIN})$			ns
$t_{WL}(\text{XIN})$	Main clock input "L" pulse width	$0.4 \cdot t_c(\text{XIN})$			ns
$t_c(\text{XCIN})$	Sub-clock input cycle time	200			ns
$t_{WH}(\text{XCIN})$	Sub-clock input "H" pulse width	$0.4 \cdot t_c(\text{XCIN})$			ns
$t_{WL}(\text{XCIN})$	Sub-clock input "L" pulse width	$0.4 \cdot t_c(\text{XCIN})$			ns
$t_c(\text{INT})$	INT0, INT1 input cycle time	200			ns
$t_{WH}(\text{INT})$	INT0, INT1 input "H" pulse width	90			ns
$t_{WL}(\text{INT})$	INT0, INT1 input "L" pulse width	90			ns
$t_d(\phi - \text{TOUT})$	Timer TOUT delay time			15	ns
$t_c(\text{SCLKE})$	Serial I/O external clock input cycle time	400			ns
$t_{WH}(\text{SCLKE})$	Serial I/O external clock input "H" pulse width	190			ns
$t_{WL}(\text{SCLKE})$	Serial I/O external clock input "L" pulse width	180			ns
$t_{su}(\text{SRXD-SCLKE})$	Serial I/O input setup time (external clock)	15			ns
$t_h(\text{SCLKE-SRXD})$	Serial I/O input hold time (external clock)	10			ns
$t_d(\text{SCLKE-STXD})$	Serial I/O output delay time (external clock)			25	ns
$t_v(\text{SCLKE-SRDY})$	Serial I/O SRDY valid time (external clock)			26	ns
$t_c(\text{SCLKI})$	Serial I/O internal clock output cycle time	166.66			ns
$t_{WH}(\text{SCLKI})$	Serial I/O internal clock output "H" pulse width	$0.5 \cdot t_c(\text{SCLKI}) - 5$			ns
$t_{WL}(\text{SCLKI})$	Serial I/O internal clock output "L" pulse width	$0.5 \cdot t_c(\text{SCLKI}) - 5$			ns
$t_{su}(\text{SRXD-SCLKI})$	Serial I/O input setup time (internal clock)	20			ns
$t_h(\text{SCLKI-SRXD})$	Serial I/O input hold time (internal clock)	5			ns
$t_d(\text{SCLKI-STXD})$	Serial I/O output delay time (internal clock)			5	ns

**Note:** Make sure not to exceed 12 MHz of  $\phi$ , in other words,  $t_c(\phi) \geq 83.33\text{ ns}$ . For example, set bit 7 of the clock control register (CCR) to "0" in the case of  $t_c(\text{XIN}) < 41.66\text{ ns}$ .

### 3.1.5 Timing requirements and switching characteristics in memory expansion and microprocessor modes (In $V_{CC} = 5\text{ V}$ )

**Table 3.1.6 Timing requirements and switching characteristics in memory expansion and microprocessor modes ( $V_{CC} = 4.15$  to  $5.25\text{ V}$ ,  $V_{SS} = 0\text{ V}$ ,  $T_a = -20$  to  $70^\circ\text{C}$ , unless otherwise noted)**

Symbol	Parameter	Limits			Unit
		Min.	Typ.	Max.	
$t_C(\phi)$	$\phi$ clock cycle time	83.33			ns
$t_{WH}(\phi)$	$\phi$ clock "H" pulse width	$0.5 \cdot t_C(\phi) - 5$			ns
$t_{WL}(\phi)$	$\phi$ clock "L" pulse width	$0.5 \cdot t_C(\phi) - 5$			ns
$t_d(\phi - AH)$	AB15–AB8 delay time			31	ns
$t_v(\phi - AH)$	AB15–AB8 valid time	0			ns
$t_d(\phi - AL)$	AB7–AB0 delay time			33	ns
$t_v(\phi - AL)$	AB7–AB0 valid time	0			ns
$t_d(\phi - WR)$	$\overline{WR}$ delay time			6	ns
$t_v(\phi - WR)$	$\overline{WR}$ valid time	0			ns
$t_d(\phi - RD)$	$\overline{RD}$ delay time			6	ns
$t_v(\phi - RD)$	$\overline{RD}$ valid time	0			ns
$t_d(\phi - SYNC)$	SYNCOUT delay time			6	ns
$t_v(\phi - SYNC)$	SYNCOUT valid time	0			ns
$t_d(\phi - DMA)$	DMAOUT delay time			25	ns
$t_v(\phi - DMA)$	DMAOUT valid time	0			ns
$t_{su}(RDY - \phi)$	RDY setup time	21			ns
$t_h(\phi - RDY)$	RDY hold time	0			ns
$t_{su}(HOLD - \phi)$	$\overline{HOLD}$ setup time	21			ns
$t_h(\phi - HOLD)$	$\overline{HOLD}$ hold time	0			ns
$t_d(\phi - HLDAL)$	$\overline{HOLD}$ "L" delay time			25	ns
$t_d(\phi - HLDALH)$	$\overline{HOLD}$ "H" delay time			25	ns
$t_{su}(DB - \phi)$	Data bus setup time	7			ns
$t_h(\phi - DB)$	Data bus hold time	0			ns
$t_d(\phi - DB)$	Data bus delay time			22	ns
$t_v(\phi - DB)$	Data bus valid time ( <b>Note 1</b> )	13			ns
$t_d(\phi - EDMA)$	$\overline{EDMA}$ delay time			12	ns
$t_v(\phi - EDMA)$	$\overline{EDMA}$ valid time	0			ns
$t_{WL}(WR)$ ( <b>Note 2</b> )	$\overline{WR}$ pulse width	$0.5 \cdot t_C(\phi) - 5$			ns
$t_{WL}(RD)$ ( <b>Note 2</b> )	$\overline{RD}$ pulse width	$0.5 \cdot t_C(\phi) - 5$			ns
$t_d(AH - WR)$	AB15–AB8 valid time before $\overline{WR}$	$0.5 \cdot t_C(\phi) - 28$			ns
$t_d(AL - WR)$	AB7–AB0 valid time before $\overline{WR}$	$0.5 \cdot t_C(\phi) - 30$			ns
$t_v(WR - AH)$	AB15–AB8 valid time after $\overline{WR}$	0			ns
$t_v(WR - AL)$	AB7–AB0 valid time after $\overline{WR}$	0			ns
$t_d(AH - RD)$	AB15–AB8 valid time before $\overline{RD}$	$0.5 \cdot t_C(\phi) - 28$			ns
$t_d(AL - RD)$	AB7–AB0 valid time before $\overline{RD}$	$0.5 \cdot t_C(\phi) - 30$			ns
$t_v(RD - AH)$	AB15–AB8 valid time after $\overline{RD}$	0			ns
$t_v(RD - AL)$	AB7–AB0 valid time after $\overline{RD}$	0			ns
$t_{su}(RDY - WR)$	RDY setup time before $\overline{WR}$	27			ns
$t_h(WR - RDY)$	RDY hold time after $\overline{WR}$	0			ns
$t_{su}(RDY - RD)$	RDY setup time before $\overline{RD}$	27			ns
$t_h(RD - RDY)$	RDY hold time after $\overline{RD}$	0			ns
$t_{su}(DB - RD)$	Data bus setup time before $\overline{RD}$	13			ns
$t_h(RD - DB)$	Data bus hold time after $\overline{RD}$	0			ns
$t_d(WR - DB)$	Data bus delay time before $\overline{WR}$			20	ns
$t_v(WR - DB)$	Data bus valid time after $\overline{WR}$ ( <b>Note 1</b> )	10			ns
$t_v(WR - EDMA)$	$\overline{EDMA}$ delay time after $\overline{WR}$	0			ns
$t_v(RD - EDMA)$	$\overline{EDMA}$ valid time after $\overline{RD}$	0			ns
$t_r(D+), t_r(D-)$	USB output rise time, $C_L = 50\text{ pF}$	4		20	ns
$t_f(D+), t_f(D-)$	USB output fall time, $C_L = 50\text{ pF}$	4		20	ns

**Notes 1:** Test conditions:  $I_{OHL} = \pm 5\text{ mA}$ ,  $C_L = 50\text{ pF}$

**2:**  $t_{WL}(RD) = ((n + 0.5) \cdot t_C(\text{PHI})) - 5\text{ ns}$  ( $n = \text{wait number}$ )

$t_{WL}(WR) = ((n + 0.5) \cdot t_C(\text{PHI})) - 5\text{ ns}$  ( $n = \text{wait number}$ )

For example, two software waits,  $\text{PHI} = 12\text{ MHz}$  operating:  $t_{WL}(RD) = 2.5 \cdot t_C(\text{PHI}) - 5\text{ ns} = 203.33\text{ ns}$

### 3.1.6 Recommended operating conditions

In  $V_{CC} = 3\text{ V}$

**Table 3.1.7 Recommended operating conditions**

( $V_{CC} = 3.0$  to  $3.6\text{ V}$ ,  $V_{SS} = 0\text{ V}$ ,  $T_a = -20$  to  $70^\circ\text{C}$ , unless otherwise noted)

Symbol	Parameter	Limits			Unit
		Min.	Typ.	Max.	
$V_{CC}$	Power source voltage	3.0	3.3	3.6	V
$AV_{CC}$	Analog reference voltage	3.0	3.3	$V_{CC}$	V
$V_{SS}$	Power source voltage		0		V
$AV_{SS}$	Analog reference voltage		0		V
Ext. Cap.	DC-DC converter voltage	3.0	3.3	3.6	V
$V_{IH}$	"H" input voltage P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87	0.8 $V_{CC}$		$V_{CC}$	V
$V_{IH}$	"H" input voltage (Selecting VIH level input) P20–P27	0.5 $V_{CC}$		$V_{CC}$	V
$V_{IH}$	"H" input voltage RESET, XIN, XCIN, CNVSS	0.8 $V_{CC}$		$V_{CC}$	V
$V_{IH}$	"H" input voltage USB D+, USB D–	2.0			V
$V_{IL}$	"L" input voltage P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87	0		0.2 $V_{CC}$	V
$V_{IL}$	"L" input voltage (Selecting VIH level input) P20–P27	0		0.16 $V_{CC}$	V
$V_{IL}$	"L" input voltage RESET, XIN, XCIN, CNVSS	0		0.2 $V_{CC}$	V
$V_{IL}$	"L" input voltage USB D+, USB D–			0.8	mA
$\Sigma I_{OH(peak)}$	"H" total peak output current (Note 1) P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87			–80	mA
$\Sigma I_{OL(peak)}$	"L" total peak output current (Note 1) P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87			80	mA
$\Sigma I_{OH(avg)}$	"H" total average output current (Note 1) P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87			–40	mA
$\Sigma I_{OL(avg)}$	"L" total average output current (Note 1) P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87			40	mA
$I_{OH(peak)}$	"H" peak output current (Note 2) P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87			–10	mA
$I_{OL(peak)}$	"L" peak output current (Note 2) P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87			10	mA
$I_{OH(avg)}$	"H" average output current (Note 3) P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87			–5.0	mA
$I_{OL(avg)}$	"L" average output current (Note 3) P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87			5.0	mA
$f(XIN)$	Main clock input frequency (Notes 4, 5)	1		24	MHz
$f(XCIN)$	Sub-clock input frequency (Notes 4, 6)		32.768	50/5.0	kHz/MHz

**Notes 1:** The total peak output current is the peak value of the peak currents flowing through all the applicable ports. The total average output current is the average value measured over 100 ms flowing through all the applicable ports.

**2:** The peak output current is the peak current flowing in each port.

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

**4:** The duty of oscillation frequency is 50 %.

**5:** Connect a ceramic resonator or a quartz-crystal oscillator between the XIN and XOUT pins. Its maximum oscillation frequency must be 24 MHz. However, make sure to set  $\phi$  to 6 MHz or slower. More faster clocks are required as the  $f(XIN)$  when using the frequency synthesizer as possible.

**6:** Connect a ceramic resonator or a quartz-crystal oscillator between the XCIN and XCOUT pins. Its maximum oscillation frequency must be 50 kHz. Input an external clock having 5 MHz (max.) frequency from the XCIN pin.

## 3.1.7 Electrical characteristics

In  $V_{CC} = 3\text{ V}$ 

Table 3.1.8 Electrical characteristics (1)

 $(V_{CC} = 3.0\text{ to }3.6\text{ V}, V_{SS} = 0\text{ V}, T_a = -20\text{ to }70^\circ\text{C}, \text{ unless otherwise noted})$ 

Symbol	Parameter	Test conditions	Limits			Unit
			Min.	Typ.	Max.	
VOH	"H" output voltage P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87	$I_{OH} = -1\text{ mA}$	$V_{CC}-1.0$			V
VOH	"H" output voltage USB D+, USB D-	USB+, and USB- pins pull-down via a resistor of $15\text{ k}\Omega \pm 5\%$  USB+ pin pull-up to Ext. Cap. pin via a resistor of $1.5\text{ k}\Omega \pm 5\%$	2.8		3.6	V
VOL	"L" output voltage P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87	$I_{OL} = 1\text{ mA}$			1.0	V
VOL	"L" output voltage USB D+, USB D-	USB+, and USB- pins pull-down via a resistor of $15\text{ k}\Omega \pm 5\%$  USB+ pin pull-up to Ext. Cap. pin via a resistor of $1.5\text{ k}\Omega \pm 5\%$	0		0.3	V
VT+–VT-	Hysteresis INT0, INT1, RDY, $\overline{\text{HOLD}}$ , P20–P27 (Note 1)			0.3		V
VT+–VT-	Hysteresis URXD, SCLK, SRXD, SRDY, CTS			0.3		V
VT+–VT-	Hysteresis $\overline{\text{RESET}}$			0.3		V
I <sub>IH</sub>	"H" input current P00–P07, P10–P17, P20–P27, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87	$V_I = V_{CC}$			5.0	$\mu\text{A}$
I <sub>IH</sub>	"H" input current $\overline{\text{RESET}}$ , CNVss				5.0	$\mu\text{A}$
I <sub>IH</sub>	"H" input current X <sub>IN</sub>			9.0	20	$\mu\text{A}$
I <sub>IH</sub>	"H" input current X <sub>CIN</sub>				5.0	$\mu\text{A}$
I <sub>IL</sub>	"L" input current P00–P07, P10–P17, P30–P37, P40–P44, P50–P57, P60–P67, P70–P74, P80–P87	$V_I = V_{SS}$			-5.0	$\mu\text{A}$
I <sub>IL</sub>	"L" input current $\overline{\text{RESET}}$				-5.0	$\mu\text{A}$
I <sub>IL</sub>	"L" input current CNVss				-20	$\mu\text{A}$
I <sub>IL</sub>	"L" input current X <sub>IN</sub>			-9.0	-20	$\mu\text{A}$
I <sub>IL</sub>	"L" input current X <sub>CIN</sub>				-5.0	$\mu\text{A}$
I <sub>IL</sub>	"L" input current P20–P27	$V_I = V_{SS}$ Pull-ups "off"			-5.0	$\mu\text{A}$
		$V_{CC} = 3.0\text{ V}, V_I = V_{SS}$ Pull-ups "on"	-10	-20	-50	$\mu\text{A}$
V <sub>RAM</sub>	RAM hold voltage	When clock is stopped	2.0			V

Note 1: This spec is hysteresis of key input interrupt.

In  $V_{CC} = 3\text{ V}$ **Table 3.1.9 Electrical characteristics (2)**(V<sub>CC</sub> = 3.0 to 3.6 V, V<sub>SS</sub> = 0 V, T<sub>a</sub> = -20 to 70°C, unless otherwise noted)

Symbol	Parameter	Test conditions	Limits			Unit
			Min.	Typ.	Max.	
I <sub>CC</sub>	Power source current (Output transistor is isolated.)	Normal mode ( <b>Note 1</b> ) f(X <sub>IN</sub> ) = 24 MHz, φ = 6 MHz USB operating Frequency synthesizer ON		25	45	mA
		Wait mode ( <b>Note 2</b> ) f(X <sub>IN</sub> ) = 24 MHz, φ = 6 MHz USB block enabled, USB clock stopped, Frequency synthesizer ON		2.5	6	mA
		Wait mode ( <b>Note 3</b> ) f(X <sub>CIN</sub> ) = 32 kHz, φ = 16 kHz USB block disabled Frequency synthesizer OFF USB transceiver DC-DC converter OFF			6	μA
		Stop mode USB transceiver DC-DC converter OFF T <sub>a</sub> = 25 °C			1.0	μA
		Stop mode USB transceiver DC-DC converter OFF T <sub>a</sub> = 70 °C			10	μA

**<Test conditions>****Notes 1:** Operating in single-chip mode

Clock input from X<sub>IN</sub> pin (X<sub>OUT</sub> oscillator stopped)  
 USB operating with USB transceiver DC-DC converter enabled  
 Operating functions: Frequency synthesizer, CPU, UART, DMAC, Timers  
 Disabled functions: Serial I/O

**2:** Operating in single-chip mode with Wait mode

Clock input from X<sub>IN</sub> pin (X<sub>OUT</sub> oscillator stopped)  
 USB suspended due to USB clock stopped with USB transceiver DC-DC converter enabled  
 Operating functions: Frequency synthesizer, Timers  
 Disabled functions: CPU, UART, DMAC and Serial I/O

**3:** Operating in single-chip mode with Wait mode

X<sub>IN</sub> - X<sub>OUT</sub> oscillator stopped  
 Clock input from X<sub>CIN</sub> pin (X<sub>COUT</sub> oscillator stopped)  
 USB stopped, USB clock stopped and USB transceiver DC-DC converter disabled  
 Operating functions: Timers  
 Disabled functions: Frequency synthesizer, CPU, UART, DMAC and Serial I/O

### 3.1.8 Timing requirements

In  $V_{CC} = 3\text{ V}$

**Table 3.1.10 Timing requirements**

( $V_{CC} = 3.0$  to  $3.6\text{ V}$ ,  $V_{SS} = 0\text{ V}$ ,  $T_a = -20$  to  $70^\circ\text{C}$ , unless otherwise noted)

Symbol	Parameter	Limits			Unit
		Min.	Typ.	Max.	
$t_w(\overline{\text{RESET}})$	Reset input "L" pulse width	2			$\mu\text{s}$
$t_c(X_{IN})$	Main clock input cycle time <b>(Note)</b>	41.66			ns
$t_{WH}(X_{IN})$	Main clock input "H" pulse width	$0.4 \cdot t_c(X_{IN})$			ns
$t_{WL}(X_{IN})$	Main clock input "L" pulse width	$0.4 \cdot t_c(X_{IN})$			ns
$t_c(X_{CIN})$	Sub-clock input cycle time	200			ns
$t_{WH}(X_{CIN})$	Sub-clock input "H" pulse width	$0.4 \cdot t_c(X_{CIN})$			ns
$t_{WL}(X_{CIN})$	Sub-clock input "L" pulse width	$0.4 \cdot t_c(X_{CIN})$			ns
$t_c(\text{INT})$	INT0, INT1 input cycle time	250			ns
$t_{WH}(\text{INT})$	INT0, INT1 input "H" pulse width	110			ns
$t_{WL}(\text{INT})$	INT0, INT1 input "L" pulse width	110			ns
$t_d(\phi - \text{TOUT})$	Timer TOUT delay time			17	ns
$t_c(\text{SCLKE})$	Serial I/O external clock input cycle time	450			ns
$t_{WH}(\text{SCLKE})$	Serial I/O external clock input "H" pulse width	220			ns
$t_{WL}(\text{SCLKE})$	Serial I/O external clock input "L" pulse width	190			ns
$t_{su}(\text{SRXD-SCLKE})$	Serial I/O input setup time (external clock)	20			ns
$t_h(\text{SCLKE-SRXD})$	Serial I/O input hold time (external clock)	15			ns
$t_d(\text{SCLKE-STXD})$	Serial I/O output delay time (external clock)			34	ns
$t_v(\text{SCLKE-SRDY})$	Serial I/O SRDY valid time (external clock)			35	ns
$t_c(\text{SCLKI})$	Serial I/O internal clock output cycle time	300			ns
$t_{WH}(\text{SCLKI})$	Serial I/O internal clock output "H" pulse width	$0.5 \cdot t_c(\text{SCLKI}) - 5$			ns
$t_{WL}(\text{SCLKI})$	Serial I/O internal clock output "L" pulse width	$0.5 \cdot t_c(\text{SCLKI}) - 5$			ns
$t_{su}(\text{SRXD-SCLKI})$	Serial I/O input setup time (internal clock)	20			ns
$t_h(\text{SCLKI-SRXD})$	Serial I/O input hold time (internal clock)	5			ns
$t_d(\text{SCLKI-STXD})$	Serial I/O output delay time (internal clock)			5	ns

**Note:** Make sure not to exceed 6 MHz of  $\phi$ , in other words,  $t_c(\phi) \geq 166.66\text{ ns}$ .

### 3.1.9 Timing requirements and switching characteristics in memory expansion and microprocessor modes (In $V_{CC} = 3\text{ V}$ )

**Table 3.1.11 Timing requirements and switching characteristics in memory expansion and microprocessor modes** ( $V_{CC} = 3.0$  to  $3.6\text{ V}$ ,  $V_{SS} = 0\text{ V}$ ,  $T_a = -20$  to  $70^\circ\text{C}$ , unless otherwise noted)

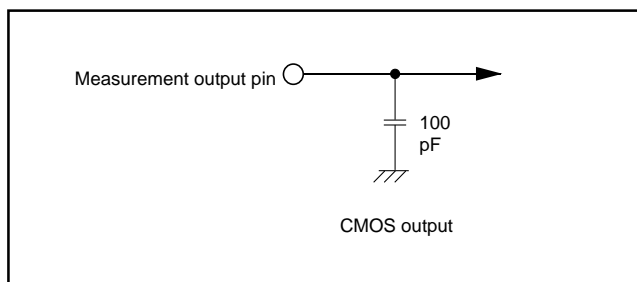
Symbol	Parameter	Limits			Unit
		Min.	Typ.	Max.	
$t_{C(\phi)}$	$\phi$ clock cycle time	166.66			$\mu\text{s}$
$t_{WH(\phi)}$	$\phi$ clock "H" pulse width	$0.5 \cdot t_{C(\phi)} - 5$			ns
$t_{WL(\phi)}$	$\phi$ clock "L" pulse width	$0.5 \cdot t_{C(\phi)} - 5$			ns
$t_{d(\phi - AH)}$	AB15–AB8 delay time			45	ns
$t_{v(\phi - AH)}$	AB15–AB8 valid time	0			ns
$t_{d(\phi - AL)}$	AB7–AB0 delay time			47	ns
$t_{v(\phi - AL)}$	AB7–AB0 valid time	0			ns
$t_{d(\phi - WR)}$	$\overline{WR}$ delay time			8	ns
$t_{v(\phi - WR)}$	$\overline{WR}$ valid time	0			ns
$t_{d(\phi - RD)}$	$\overline{RD}$ delay time			8	ns
$t_{v(\phi - RD)}$	$\overline{RD}$ valid time	0			ns
$t_{d(\phi - SYNC)}$	SYNCOUT delay time			11	ns
$t_{v(\phi - SYNC)}$	SYNCOUT valid time	0			ns
$t_{d(\phi - DMA)}$	DMAOUT delay time			26	ns
$t_{v(\phi - DMA)}$	DMAOUT valid time	0			ns
$t_{su}(RDY - \phi)$	RDY setup time	35			ns
$t_h(\phi - RDY)$	RDY hold time	0			ns
$t_{su}(HOLD - \phi)$	$\overline{HOLD}$ setup time	21			ns
$t_h(\phi - HOLD)$	$\overline{HOLD}$ hold time	0			ns
$t_{d(\phi - HLDAL)}$	$\overline{HOLD}$ "L" delay time			30	ns
$t_{d(\phi - HLDALH)}$	$\overline{HOLD}$ "H" delay time			30	ns
$t_{su}(DB - \phi)$	Data bus setup time	9			ns
$t_h(\phi - DB)$	Data bus hold time	0			ns
$t_{d(\phi - DB)}$	Data bus delay time			30	ns
$t_{v(\phi - DB)}$	Data bus valid time ( <b>Note 1</b> )	15			ns
$t_{d(\phi - EDMA)}$	$\overline{EDMA}$ delay time			15	ns
$t_{v(\phi - EDMA)}$	$\overline{EDMA}$ valid time	0			ns
$t_{WL}(WR)$ ( <b>Note 2</b> )	$\overline{WR}$ pulse width	$0.5 \cdot t_{C(\phi)} - 6$			ns
$t_{WL}(RD)$ ( <b>Note 2</b> )	$\overline{RD}$ pulse width	$0.5 \cdot t_{C(\phi)} - 6$			ns
$t_{d}(AH - WR)$	AB15–AB8 valid time before $\overline{WR}$	$0.5 \cdot t_{C(\phi)} - 33$			ns
$t_{d}(AL - WR)$	AB7–AB0 valid time before $\overline{WR}$	$0.5 \cdot t_{C(\phi)} - 35$			ns
$t_{v}(WR - AH)$	AB15–AB8 valid time after $\overline{WR}$	0			ns
$t_{v}(WR - AL)$	AB7–AB0 valid time after $\overline{WR}$	0			ns
$t_{d}(AH - RD)$	AB15–AB8 valid time before $\overline{RD}$	$0.5 \cdot t_{C(\phi)} - 33$			ns
$t_{d}(AL - RD)$	AB7–AB0 valid time before $\overline{RD}$	$0.5 \cdot t_{C(\phi)} - 35$			ns
$t_{v}(RD - AH)$	AB15–AB8 valid time after $\overline{RD}$	0			ns
$t_{v}(RD - AL)$	AB7–AB0 valid time after $\overline{RD}$	0			ns
$t_{su}(RDY - WR)$	RDY setup time before $\overline{WR}$	45			ns
$t_h(WR - RDY)$	RDY hold time after $\overline{WR}$	0			ns
$t_{su}(RDY - RD)$	RDY setup time before $\overline{RD}$	45			ns
$t_h(RD - RDY)$	RDY hold time after $\overline{RD}$	0			ns
$t_{su}(DB - RD)$	Data bus setup time before $\overline{RD}$	18			ns
$t_h(RD - DB)$	Data bus hold time after $\overline{RD}$	0			ns
$t_{d}(WR - DB)$	Data bus delay time after $\overline{WR}$			28	ns
$t_{v}(WR - DB)$	Data bus valid time after $\overline{WR}$ ( <b>Note 1</b> )	12			ns
$t_{v}(WR - EDMA)$	$\overline{EDMA}$ delay time after $\overline{WR}$	0			ns
$t_{v}(RD - EDMA)$	$\overline{EDMA}$ valid time after $\overline{RD}$	0			ns
$t_r(D+), t_r(D-)$	USB output rise time, $C_L = 50\text{ pF}$	4		20	ns
$t_f(D+), t_f(D-)$	USB output fall time, $C_L = 50\text{ pF}$	4		20	ns

**Notes 1:** Test conditions:  $I_{OHL} = \pm 5\text{ mA}$ ,  $C_L = 50\text{ pF}$

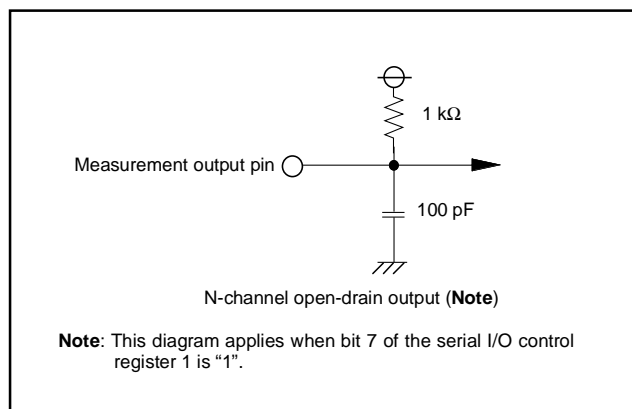
**2:**  $t_{WL}(RD) = ((n + 0.5) \cdot t_{C(Phi)}) - 5\text{ ns}$  ( $n =$  wait number)

$t_{WL}(WR) = ((n + 0.5) \cdot t_{C(Phi)}) - 5\text{ ns}$  ( $n =$  wait number)

For example, two software waits,  $Phi = 12\text{ MHz}$  operating:  $t_{WL}(RD) = 2.5 \cdot t_{C(Phi)} - 5\text{ ns} = 203.33\text{ ns}$



**Fig. 3.1.1 Circuit for measuring output switching characteristics (1)**

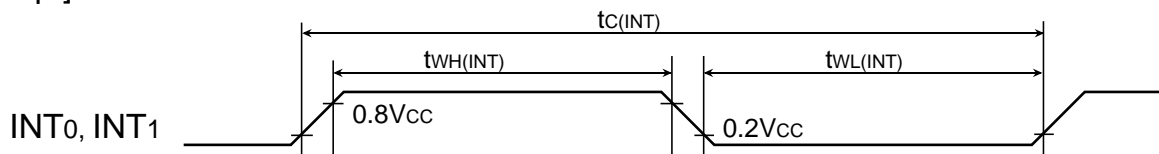


**Fig. 3.1.2 Circuit for measuring output switching characteristics (2)**

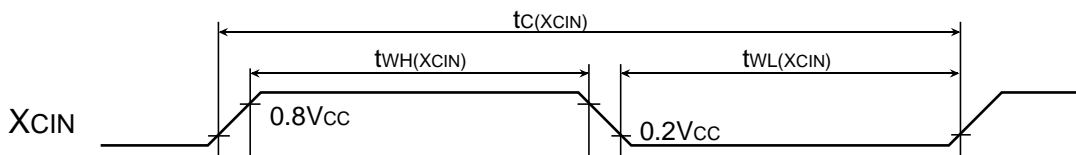
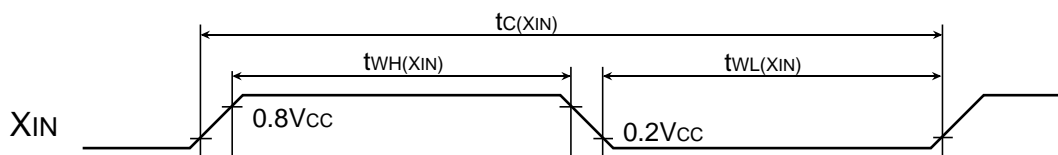
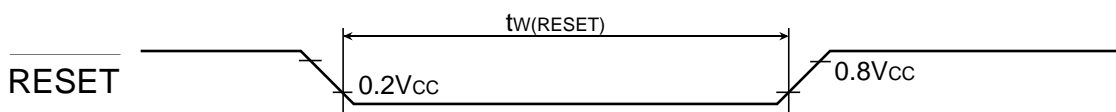


● Timing diagram

[Interrupt]



[Input]



[Timer]

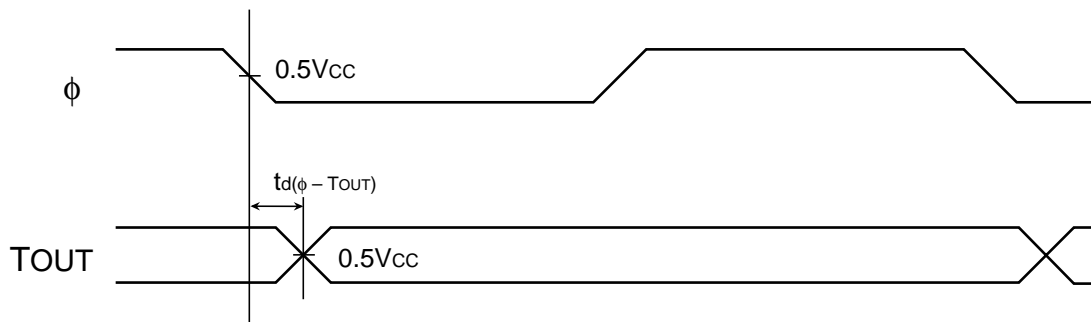


Fig. 3.1.3 Timing diagram (1)

● Timing diagram

[Serial I/O]

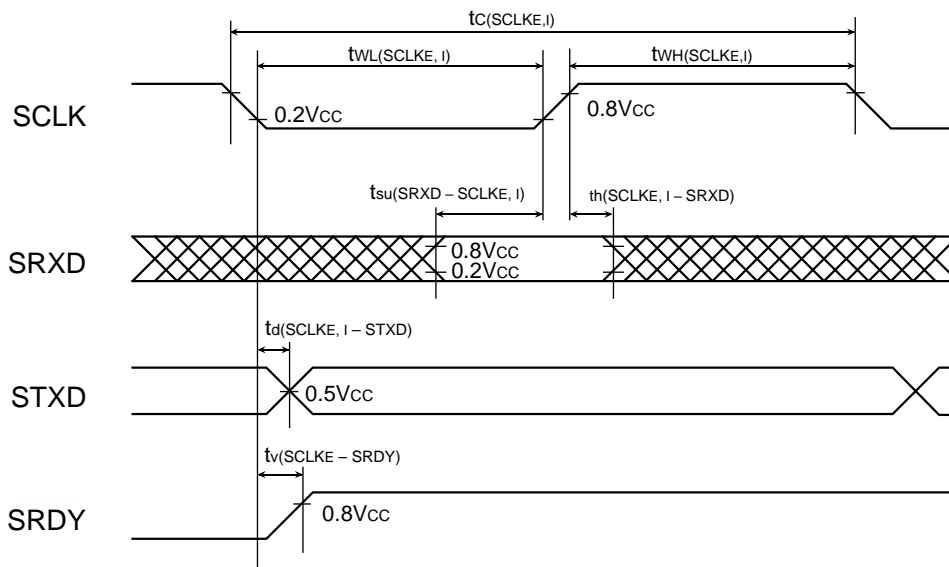


Fig. 3.1.4 Timing diagram (2)



Fig. 3.1.5 Timing diagram (3)

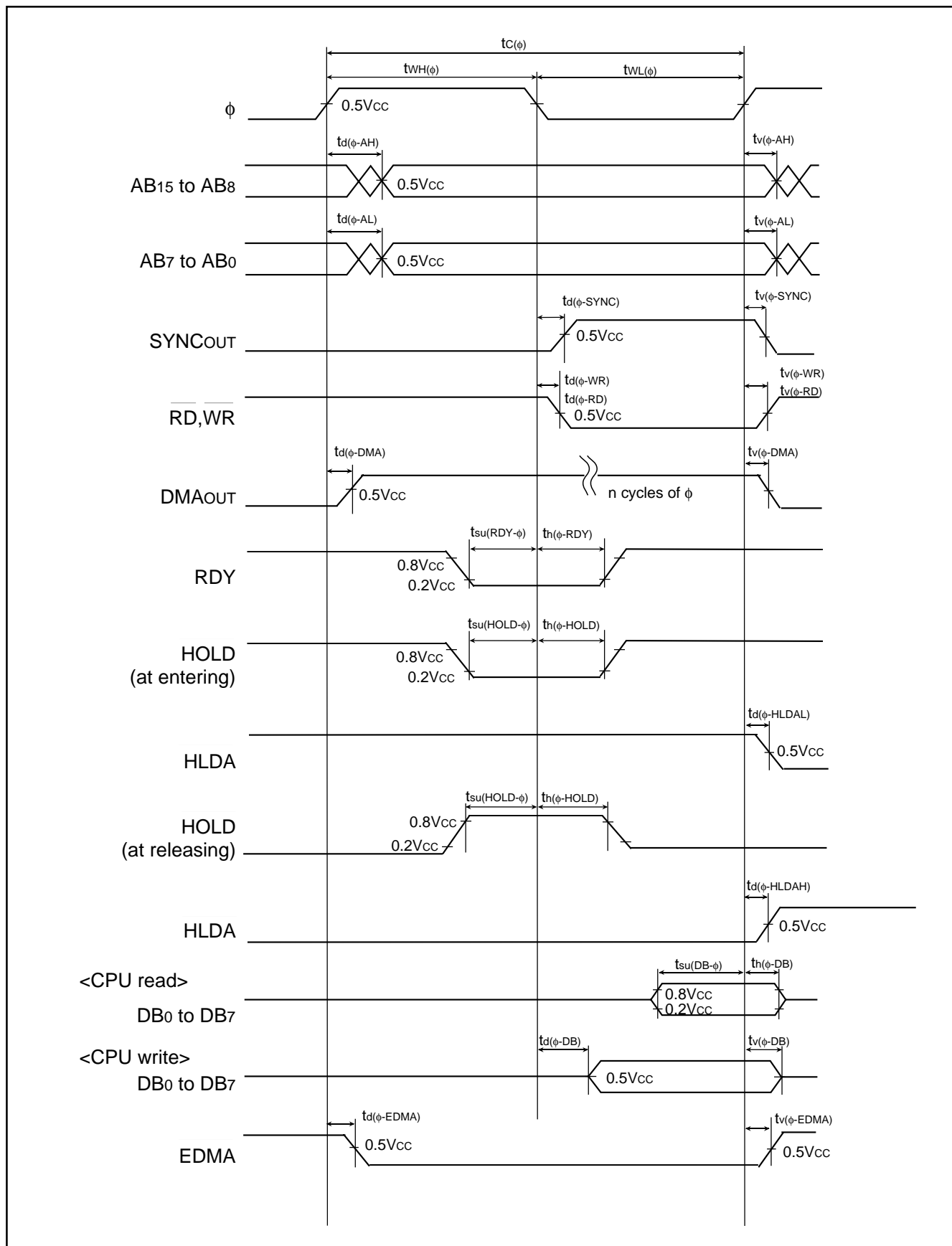


Fig. 3.1.6 Timing diagram (4); Memory expansion and microprocessor modes

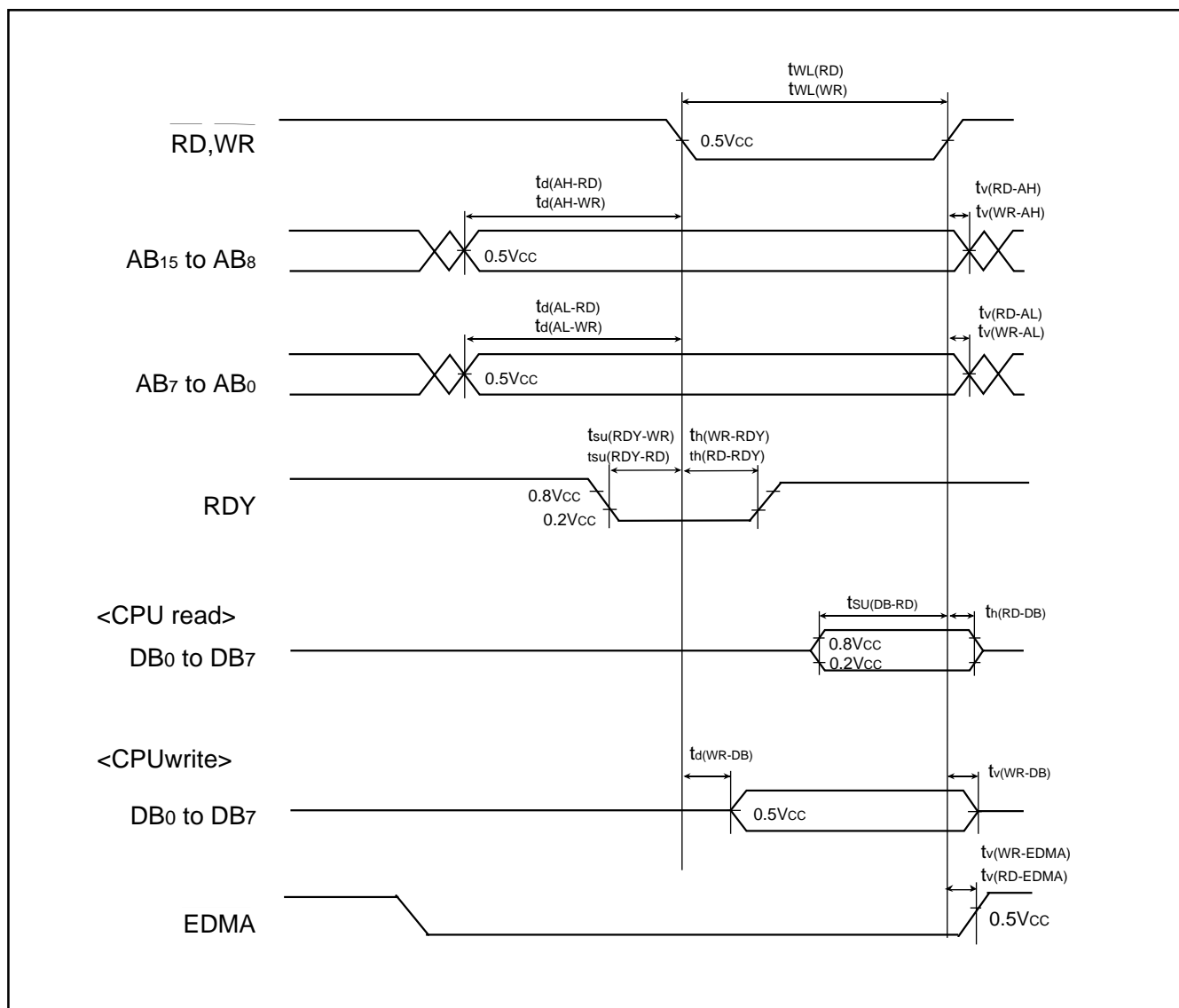


Fig. 3.1.7 Timing diagram (5); Memory expansion and microprocessor modes

## 3.2 Standard characteristics

Standard characteristics described below are just examples of the 7643 Group's characteristics and are not guaranteed. For rated values, refer to "3.1 Electrical characteristics".

### 3.2.1 Power source current standard characteristics

Figure 3.2.1 shows power source current standard characteristics.

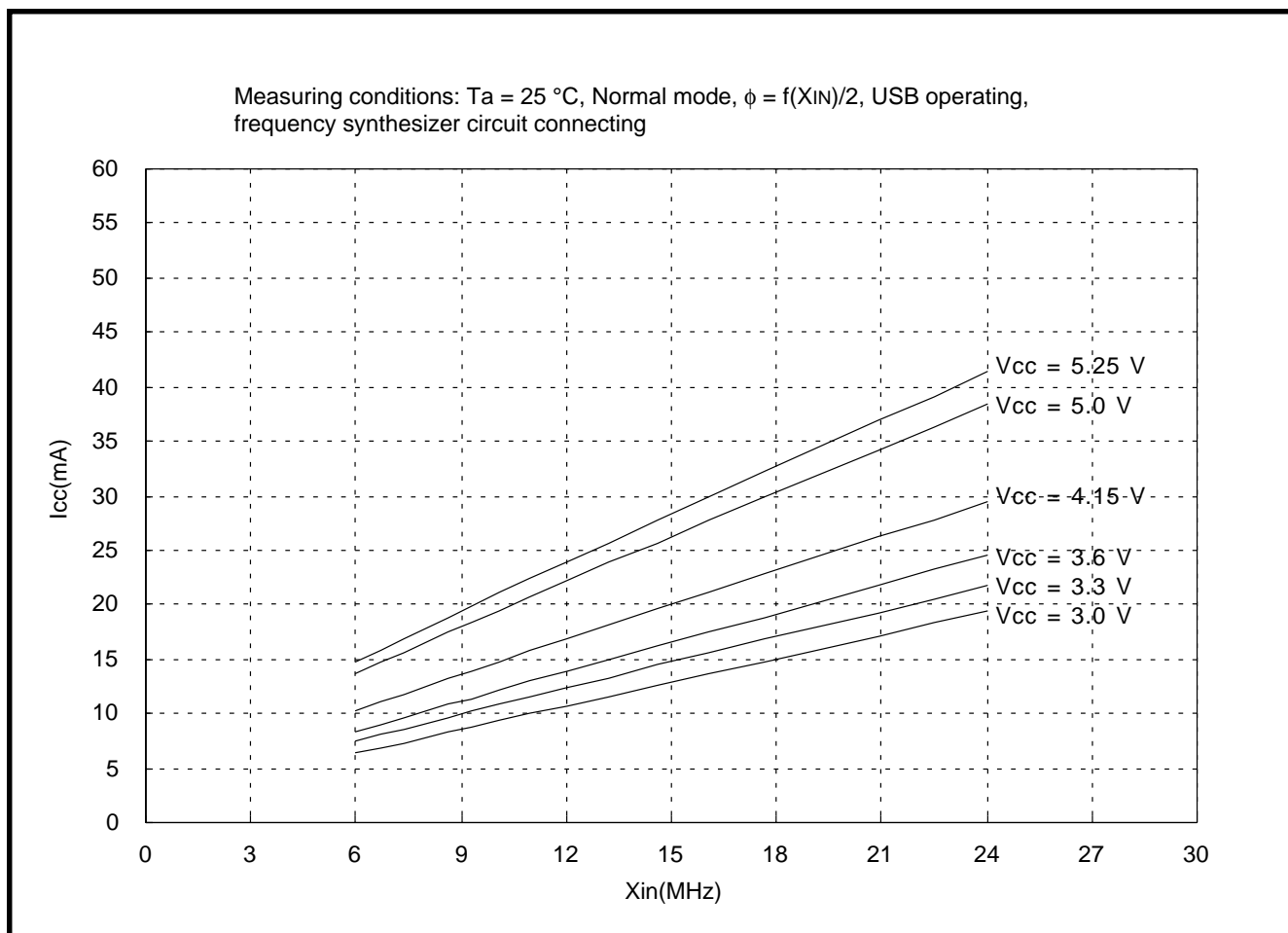


Fig. 3.2.1 Power source current standard characteristics ( $T_a = 25\text{ }^\circ\text{C}$ )

### 3.2.2 Port standard characteristics

Figure 3.2.2 to Figure 3.2.7 show port standard characteristics.

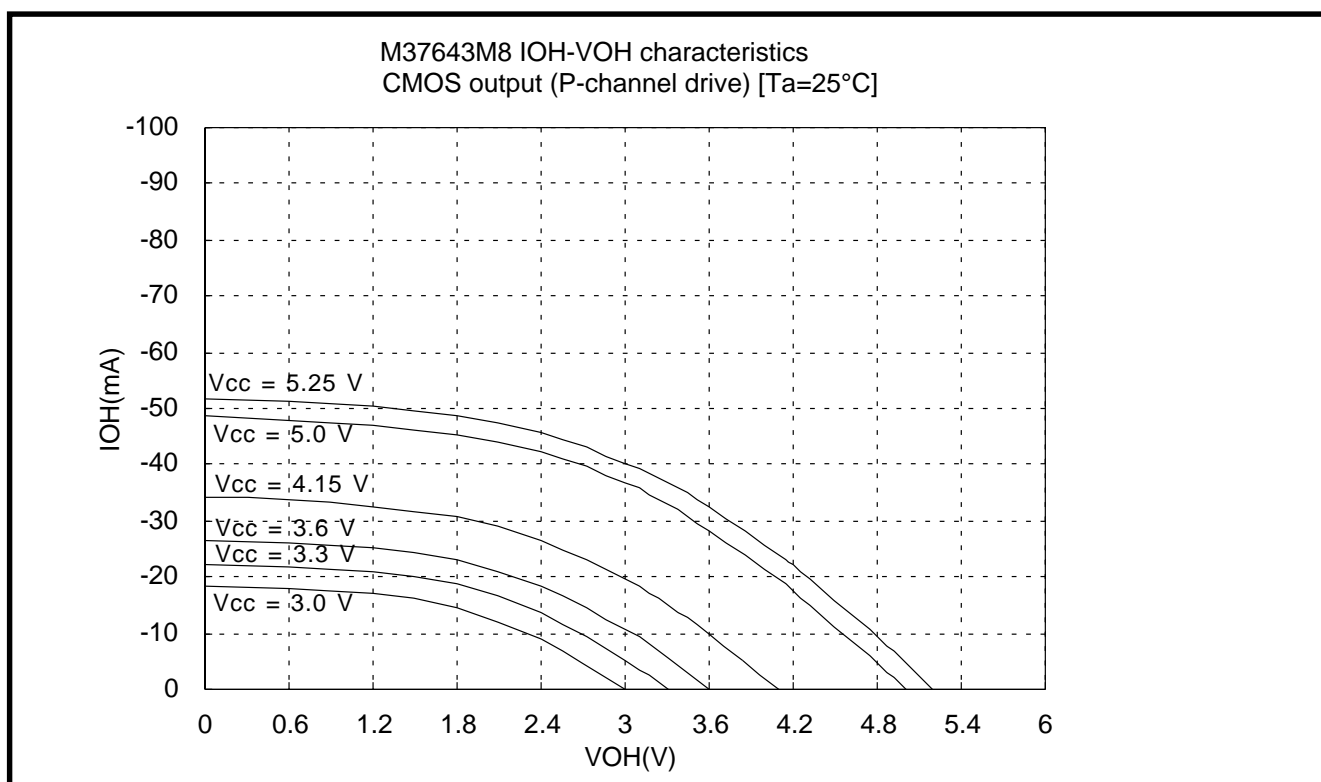


Fig. 3.2.2 CMOS output port P-channel side characteristics (Ta = 25 °C)

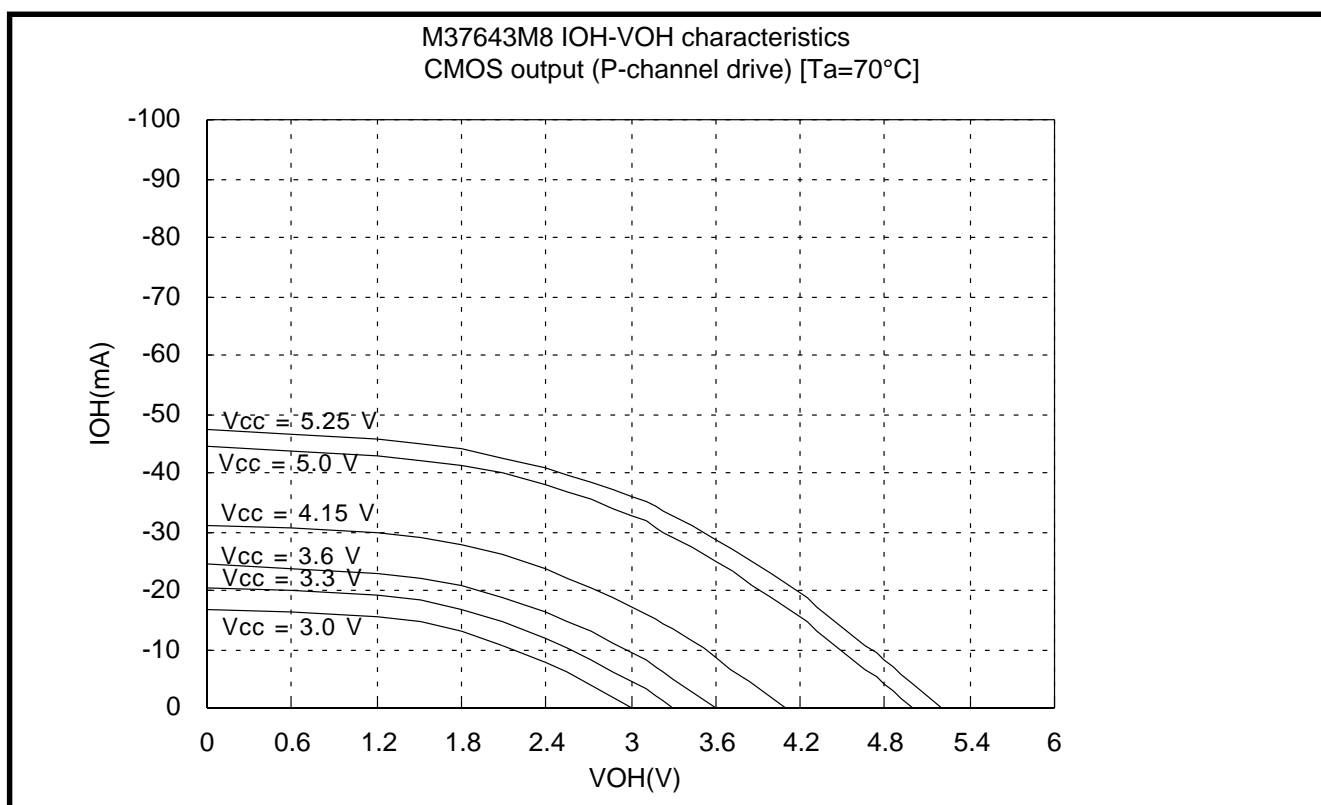


Fig. 3.2.3 CMOS output port P-channel side characteristics (Ta = 70 °C)

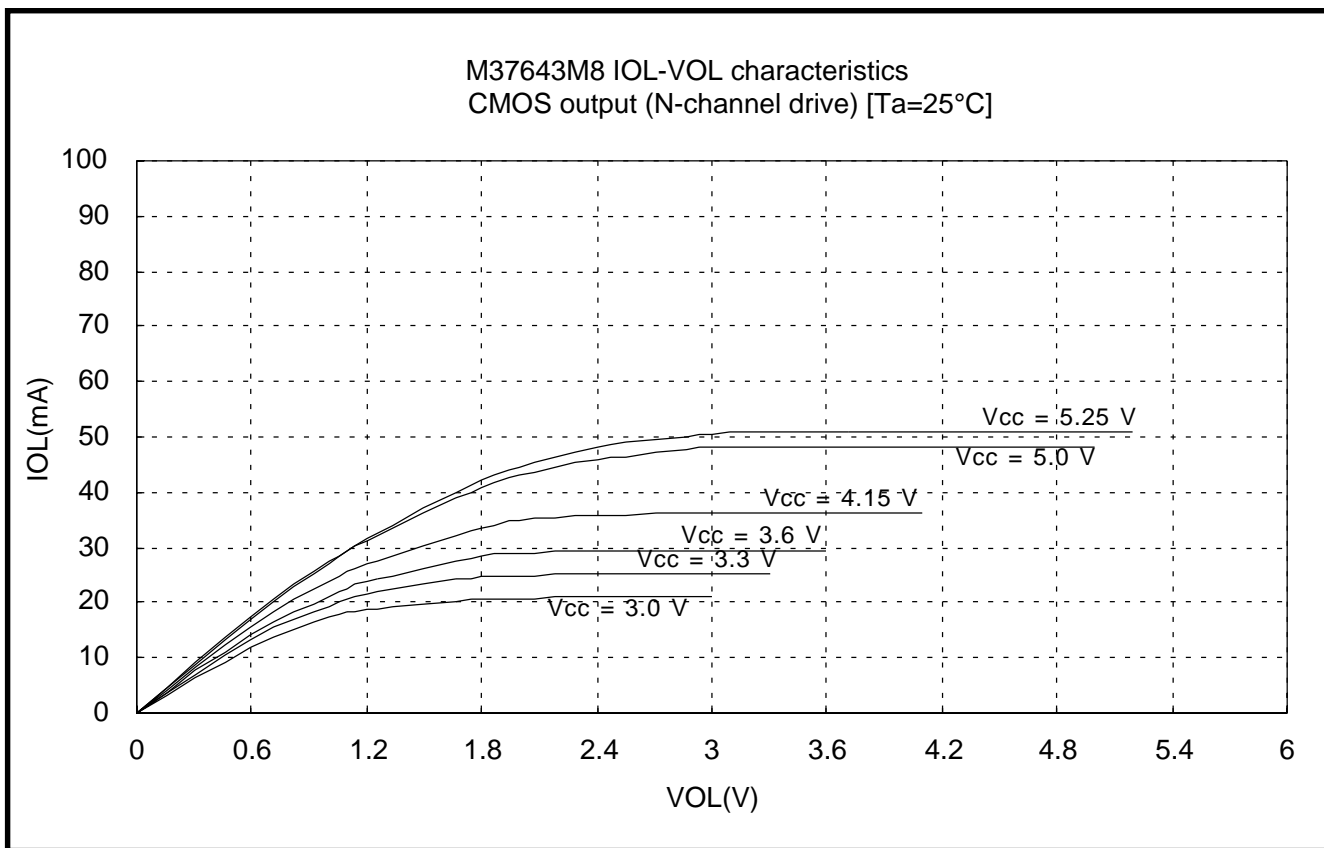


Fig. 3.2.4 CMOS output port N-channel side characteristics (Ta = 25 °C)

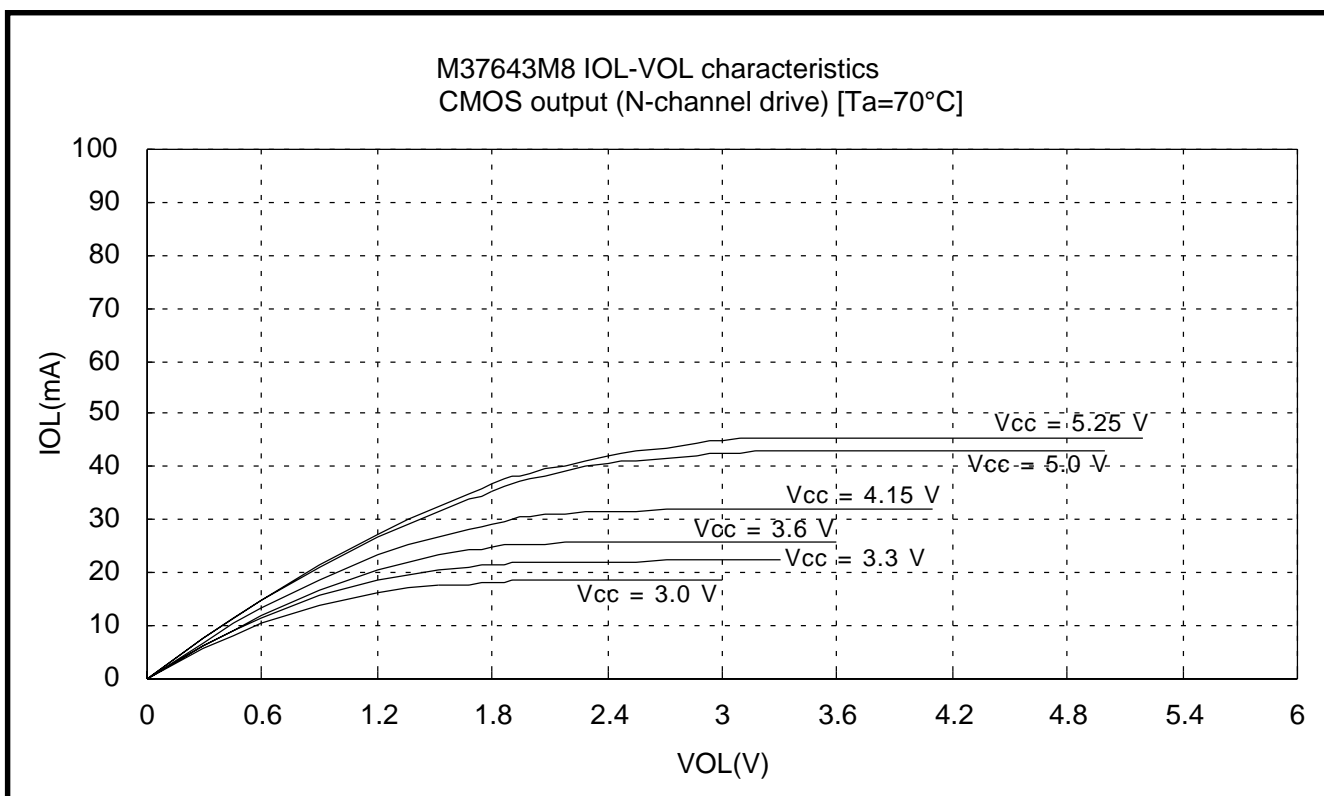


Fig. 3.2.5 CMOS output port N-channel side characteristics (Ta = 70 °C)

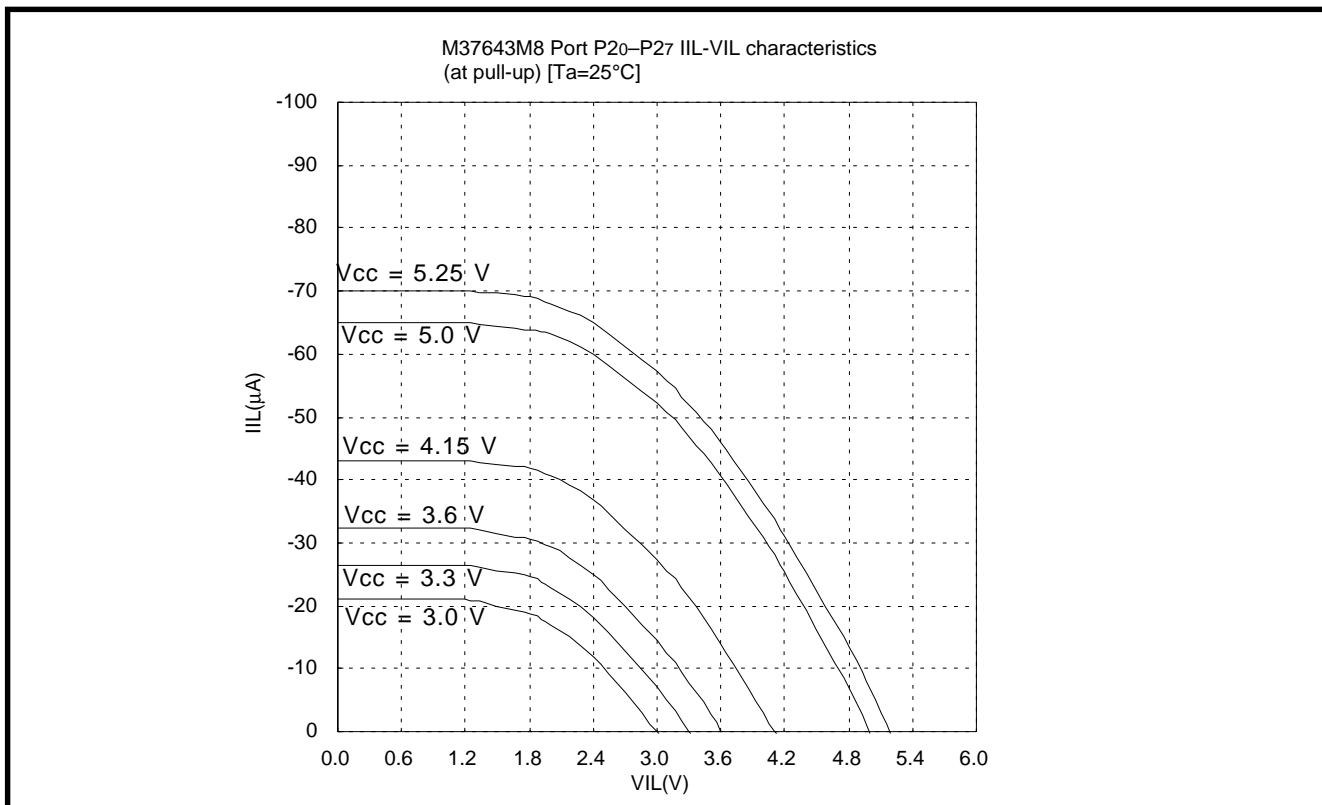


Fig. 3.2.6 Port P20–P27 at pull-up characteristics (Ta = 25 °C)

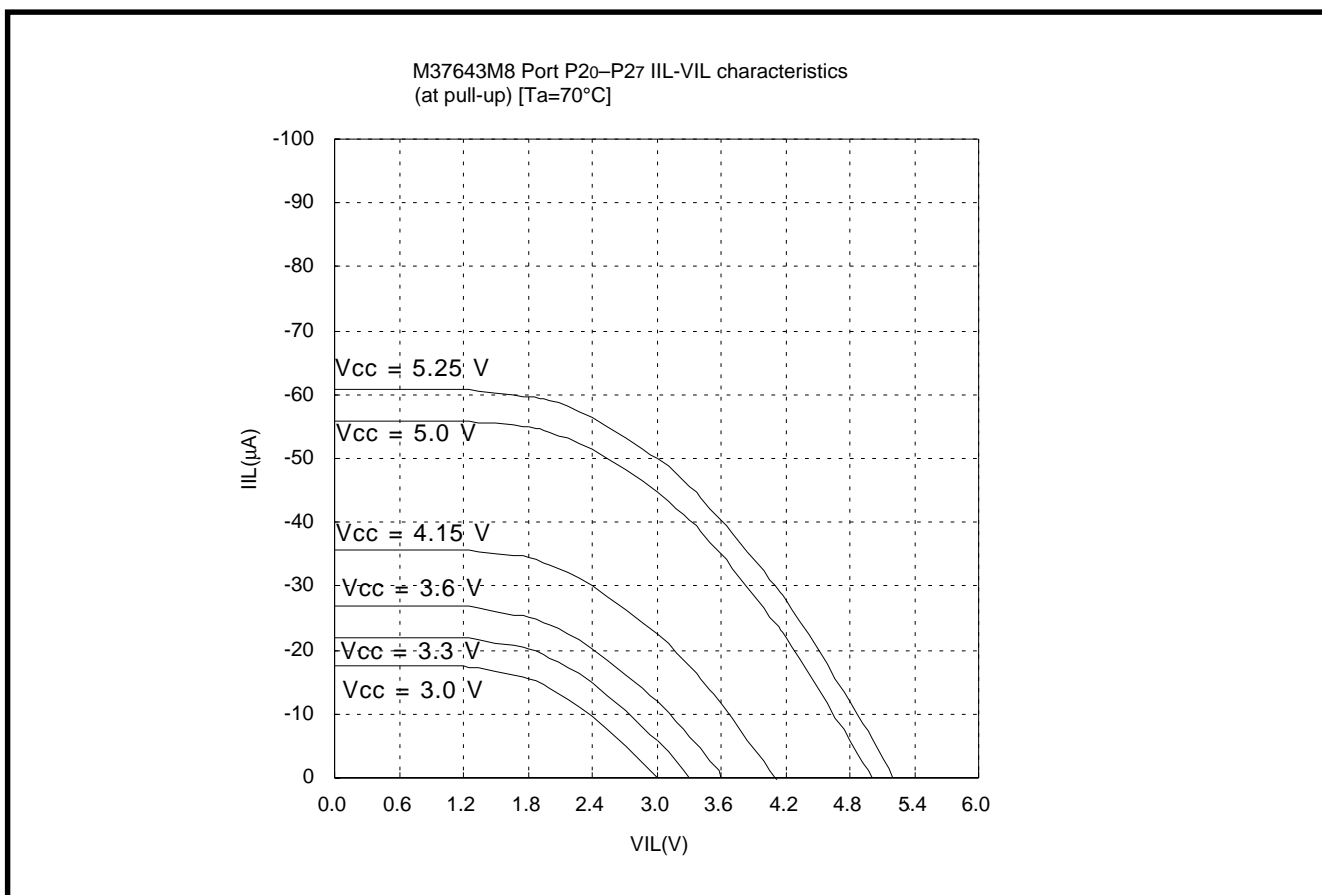


Fig. 3.2.7 Port P20–P27 at pull-up characteristics (Ta = 70 °C)



## 3.3 Notes on use

### 3.3.1 Notes on interrupts

#### (1) When setting external interrupt active edge

When setting the external interrupt active edge (INT<sub>0</sub>, INT<sub>1</sub>), the interrupt request bit may be set to "1". When not requiring the interrupt occurrence synchronized with these setting, take the following sequence.

- Interrupt polarity select register (address 0011<sub>16</sub>)

Set the above listed registers or bits as the following sequence.

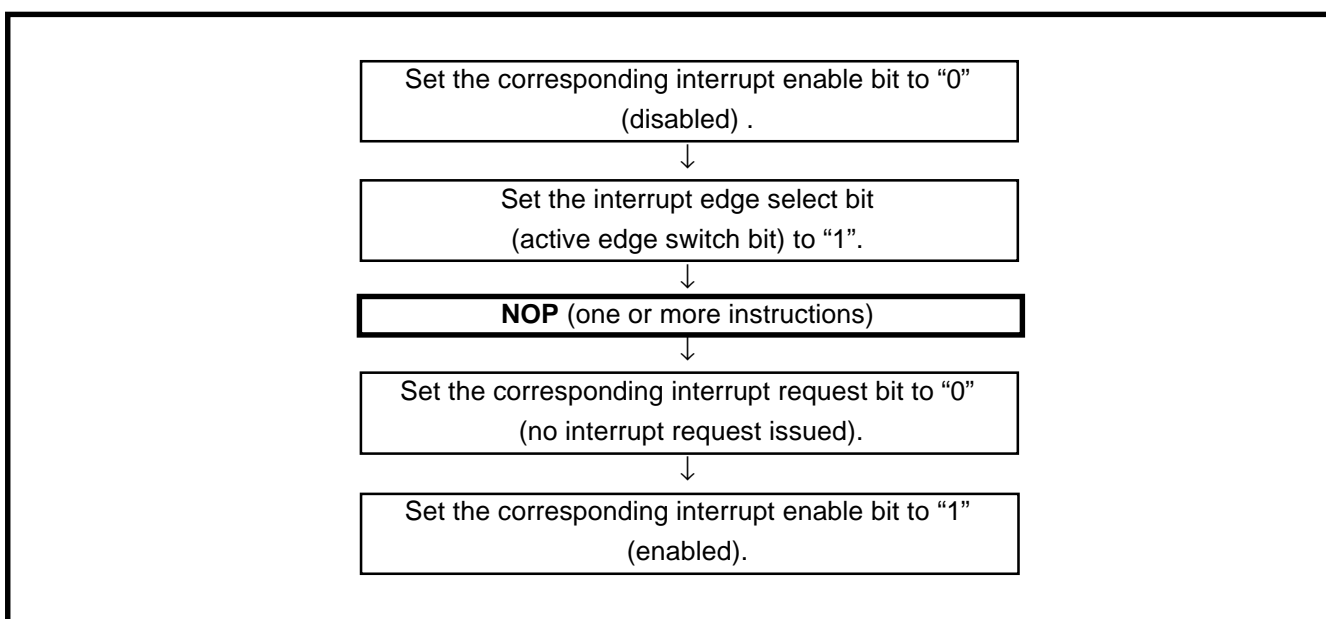


Fig. 3.3.1 Sequence of setting external interrupt active edge

### 3.3.2 Notes on serial I/O

#### (1) Clock

When the external clock (SCLK pin input) is selected as the transfer clock, its transfer clock needs to be controlled by the external source because the serial I/O shift register will keep being shifted while transfer clock is input even after transfer completion.

#### (2) Reception

When the external clock (SCLK pin input) is selected as the transfer clock for reception, the receiving operation will start owing to the shift clock input even if write operation to the serial I/O shift register (SIOSHT) is not performed. The serial I/O interrupt request also occurs at completion of receiving. However, we recommend to write dummy data in the serial I/O shift register. Because this will cause followings and improve transfer reliability.

- Write to SIOSHT puts the SRDY pin to "L". This enables shift clock output of an external device.
- Write to SIOSHT clears the internal serial I/O counter.

**Note:** Do not read the serial I/O shift register which is shifting. Because this will cause incorrect-data read.

#### (3) STXD output

- When the internal synchronous clock is selected as the transfer clock, the STXD pin goes a high-impedance state after transfer completion.
- When the external clock (SCLK pin input) is selected as the transfer clock, the STXD pin does not go a high-impedance state after transfer completion.

#### (4) SPI compatible mode

- When using the SPI compatible mode, set the  $\overline{\text{SRDY}}$  select bit to "1" ( $\overline{\text{SRDY}}$  signal output).
- When the external clock is selected in SPI compatible mode, the SRXD pin functions as a data output pin and the STXD pin functions as a data input pin.
- Do not write to the serial I/O shift register (SIOSHT) during a transfer as slave when in SPI compatible mode.
- Master operation of SPI compatible mode requires the timings:
  - From write operation to the SIOSHT to  $\overline{\text{SRDY}}$  pin put to "L"  
Requires 2 cycles of internal clock  $\phi$  + 2 cycles of serial I/O synchronous clock + 35 ns
  - From  $\overline{\text{SRDY}}$  pin put to "L" to SCLK switch  
Requires 35 ns
  - From the last pulse of SCLK to  $\overline{\text{SRDY}}$  pin put to "H"  
Requires 35 ns.

### 3.3.3 Notes on UART

#### (1) Receive

- When any one of errors occurs, the summing error flag is set to “1” and the UART summing error interrupt request bit is also set to “1”. If a receive error occurs, the reception does not set the UART receive buffer full interrupt request bit to “1”.
- If the receive enable bit (REN) is set to “0” (disabled) while a data is being received, the receiving operation will stop after the data has been received.
- Setting the receive initialization bit (RIN) to “1” resets the UART RTS control register (URTS) to “80<sub>16</sub>”. After setting the RIN bit to “1”, set this URTS.

#### (2) Transmit

- Once the transmission starts, it continues until the last bit has been transmitted even though clearing the transmit enable bit (TEN) to “0” (disabled) or inputting “H” to the CTS pin. After completion of the current transmission, the transmission is disabled.
- The transmit complete flag (TCM) is changed from “1” to “0” later than 0.5 to 1.5 clocks of the shift clock. Accordingly, take it in consideration to transmit data confirming the TCM flag after the data is written into the transmit buffer register.

#### (3) Register settings

- If updating a value of UART baud rate generator while the data is being transmitted or received, be sure to disable the transmission and reception before updating. If the former data remains in the UART transmit buffer registers 1 and 2 at retransmission, an undefined data might be output.
- The all error flags PER, FER, OER and SER are cleared to “0” when the UART status register is read, at the hardware reset or initialization by setting the Transmit Initialization Bit. These flags are also cleared to “0” by execution of bit test instructions such as **BBC** and **BCS**.
- The transmit buffer empty flag (TBE) is set to “0” when the low-order byte of transmitted data is written into the UART transmit buffer register 1. When using 9-bit character length, set the data into the UART transmit buffer register 2 (high-order byte) first before the UART transmit buffer register 1 (low-order byte).
- The receive buffer full flag (RBF) is set to “0” when the contents of UART receive buffer register 1 is read out. When using 9-bit character length, read the data from the UART receive buffer register 2 (high-order byte) first before the UART receive buffer register 1 (low-order byte).
- If a character bit length is 7 bits, bit 7 of the UART transmit/receive buffer register 1 and bits 0 to 7 of the UART transmit/receive buffer register 2 are ignored at transmitting; they are invalid at receiving.  
If a character bit length is 8 bits, bits 0 to 7 of the UART transmit/receive buffer register 2 are ignored at transmitting; they are invalid at receiving.  
If a character bit length is 9 bits, bits 1 to 7 of the UART transmit/receive buffer register 2 are ignored at transmitting; they are “0” at receiving.
- The reset cannot affect the contents of baud rate generator.

#### (4) UART address mode

- When the MSB of the incoming data is “0” in the UART address mode, the receive buffer full flag (RBF) is set to “1”, but the receive buffer full interrupt request bit is not set to “1”.
- An overrun error cannot be detected after the first data has been received in UART address mode.
- The UART address mode can be used in either an 8-bit or 9-bit character length. 7-bit character length cannot be used.

**(5) Receive error flag**

The all error flags PER, FER, OER and SER are cleared to "0" when the UART status register is read, at the hardware reset or initialization by setting the Transmit Initialization Bit. Accordingly, note that these flags are also cleared to "0" by execution of bit test instructions such as BBC and BBS, not only LDA.

**(6)  $\overline{\text{CTS}}$  function**

When the  $\overline{\text{CTS}}$  function is enabled, the transmitted data is not transferred to the transmit shift register until "L" is input to the  $\overline{\text{CTS}}$  pin (P8/ $\overline{\text{CTS}}$ ). As the result, do not set the following data to the transmit buffer register.

**(7)  $\overline{\text{RTS}}$  function**

- If the start bit is detected in the term of "H" assertion of  $\overline{\text{RTS}}$ , its assertion count is suspended and the  $\overline{\text{RTS}}$  pin remains "H" output. After receiving the last stop bit, the count is resumed.
- Setting the receive initialization bit (RIN) to "1" resets the UART RTS control register (URTS) to "80<sub>16</sub>". After setting the RIN bit to "1", set this URTS.

**(8) Interrupt**

- When setting the transmit initialization bit (TIN) to "1", both the transmit buffer empty flag (TBE) and the transmit complete flag (TCM) are set to "1", so that the transmit interrupt request occurs independent of its interrupt source. After setting the transmit initialization bit (TIN) to "1", clear the transmit interrupt request bit to "0" before setting the transmit enable bit (TEN) to "1".
- The transmit interrupt request bit is set and the interrupt request is generated by setting the transmit enable bit (TIN) to "1" even when selecting timing that either of the following flags is set to "1" as timing where the transmission interrupt is generated:
  - (1) Transmit buffer empty flag is set to "1"
  - (2) Transmit complete flag is set to "1".

Therefore, when the transmit interrupt is used, set the transmit interrupt enable bit to transmit enabled as the following sequence:

- (1) Transmit enable bit is set to "1"
- (2) Transmit interrupt request bit is set to "0"
- (3) Transmit interrupt enable bit is set to "1".

### 3.3.4 Notes on DMAC

#### (1) Transfer time

- One-byte data transfer requires 2 cycles of  $\phi$  (read and write cycles).
- To perform DMAC transfer due to the different transfer requests on the same DMAC channel or DMAC transfer between both DMAC channels, 1 cycle of  $\phi$  or more is needed before transfer is started.

#### (2) Priority

- The DMAC places a higher priority on channel-0 transfer requests than on channel-1 transfer requests.

If a channel-0 transfer request occurs during a channel-1 burst transfer operation, the DMAC completes the next transfer source and destination read/write operation first, and then stops the channel-1 transfer operation.

The channel-1 transfer operation which has been suspended is automatically resumed from the point where it was suspended so that channel-1 transfer can complete its one-burst transfer unit. This will be performed even if another channel-0 transfer request occurs.

- The suspended transfer due to the interrupt can also be resumed during its interrupt process routine by writing "1" to the DMAC channel x enable bit (DxCEN).

#### (3) Related registers

- A read/write must be performed to the source registers, transfer destination registers and transfer count registers as follows:

Read from each higher byte first, then the lower byte

Write to each lower byte first, then the higher byte.

Note that if the lower byte is read out first, the values are the higher byte's.

- Do not access the DMAC-related registers by using a DMAC transfer. The destination address data and the source address data will collide in the DMAC internal bus.
- When setting the DMAC channel x enable bit (bit 7 of address 41<sub>16</sub>) to "1", be sure simultaneously to set the DMAC channel x transfer initiation source capture register reset bit (bit 6 of address 41<sub>16</sub>) to "1". If this is not performed, an incorrect data will be transferred at the same time when the DMAC is enabled.

#### (4) DMA<sub>OUT</sub> pin

In the memory expansion mode and microprocessor mode, the DMA<sub>OUT</sub> pin (P3<sub>3</sub>/DMA<sub>OUT</sub>) outputs "H" during a DMA transfer.

### 3.3.5 Notes on USB

#### (1) USB reception

- When the OUT FIFO contains 2-data packets in the endpoints 1 to 2 used, one-data packet will still remain in the OUT FIFO even after the data of the OUT max. packet size has been read. In this case the OUT\_PKT\_RDY flag is not cleared even if it is set to "0". (The flag returns from "0" to "1" 83 ns later ( $V_{CC} = 5\text{ V}$ ,  $f(X_{IN}) = 24\text{ MHz}$ ) than the clearing.)
- Read one packet data from the OUT FIFO before clearing the OUT\_PKT\_RDY flag. If the OUT\_PKT\_RDY flag is cleared while one-data packet is being read, the internal read pointer cannot operate normally.

#### (2) USB transmission

- The IN FIFO status can be checked by monitoring the IN\_PKT\_RDY bit and the TX\_NOT\_EPT flag.
- When NULL packet transmission is required in the endpoints 1 to 2 used, perform it under the conditions of the IN\_PKT\_RDY bit set to "1" and no data in FIFO.

#### (3) External circuit

- Connect a capacitor between the Ext. Cap. pin and the Vss pin. The capacitor should have a 2.2  $\mu\text{F}$  capacitor (Tantalum capacitor) and a 0.1  $\mu\text{F}$  capacitor (ceramic capacitor) connected in parallel. Additionally, connect a 1.5  $\text{k}\Omega$  ( $\pm 5\%$ ) resistor between the Ext. Cap. pin and the D+ pin.
- The Full-Speed USB 2.0 requires a driver impedance 28 to 44  $\Omega$ . (Refer to Clause 7.1.1.1 Full-speed (12 Mb/s) Driver Characteristics in the USB specification.) In order to meet the USB specification impedance requirements, connect a resistor (27 to 33  $\Omega$  recommended) in series to the USB D+ pin and the USB D- pin.

In addition, in order to reduce the ringing and control the falling/rising timing of USB D+/D- and a crossover point, connect a capacitor between the USB D+/D- pins and the Vss pin if necessary. The values and structure of those peripheral elements depend on the impedance characteristics and the layout of the printed circuit board. Accordingly, evaluate your system and observe waveforms before actual use and decide use of elements and the values of resistors and capacitors.

Figure 3.3.2 shows the circuit example for the proper positions of the peripheral components.

- In  $V_{CC} = 3.3\text{ V}$  operation, connect the Ext. Cap. pin directly to the Vcc pin in order to supply power to the USB transceiver. In addition, you will need to disable the DC-DC converter in this operation (set bit 4 of the USB control register to "0".) If you are using the bus powered supply in  $V_{CC} = 3.3\text{ V}$  operation, the DC-DC converter must be placed outside the MCU.
- In  $V_{CC} = 5\text{ V}$  operation, do not connect the external DC-DC converter to the Ext. Cap. pin. Use the built-in DC-DC converter by enabling the USB line driver.
- Make sure the USB D+/D- lines do not cross any other wires. Keep a large GND area to protect the USB lines. Also, make sure you use a USB specification compliant connector for the connection.
- All passive components must be located as close as possible to the LPF pin. Figure 3.3.3 shows the passive components near LPF pin
- An insulation connector (Ferrite Beads) must be connected between AVss and Vss pins and between AVcc and Vcc pins. (See Figure 3.3.4.)

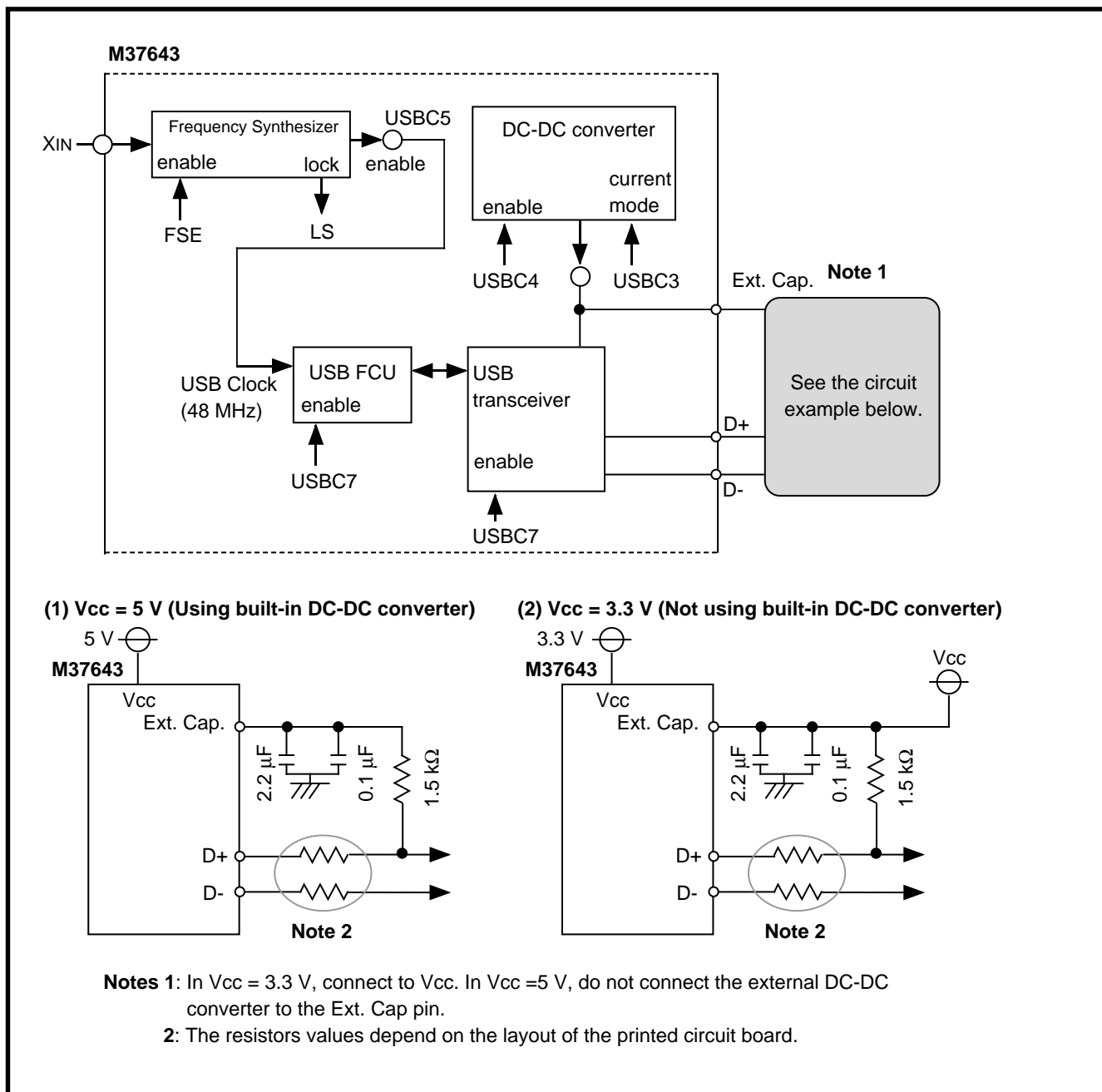


Fig. 3.3.2 Circuit example for the proper positions of the peripheral components

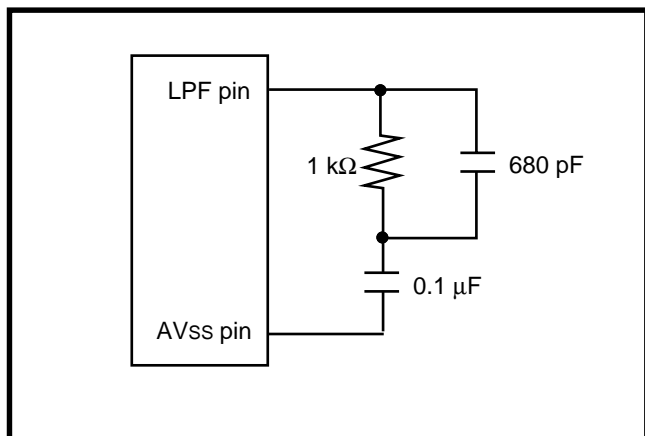


Fig. 3.3.3 Passive components near LPF pin

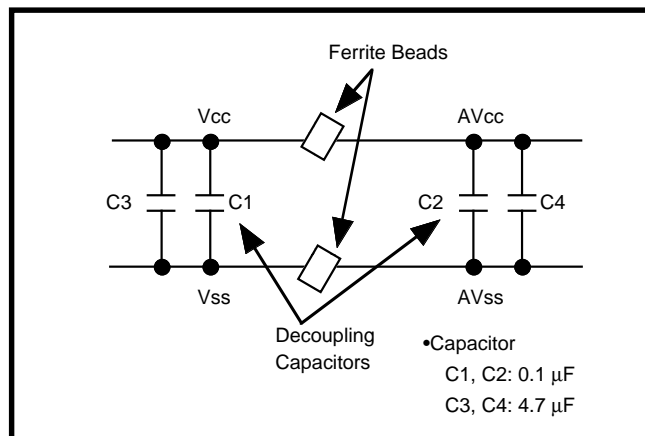


Fig. 3.3.4 Insulation connector connection

#### (4) 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.

#### (5) Registers and bits

- When using the endpoint 0, use the USB endpoint 0 IN max. packet size register for transmission and reception (IN packet size and OUT packet size).
- When not using the USB endpoint x (x = 0 to 2) IN max. packet size register and USB endpoint x OUT max. packet size register, set them to "0".
- To write to/read from the USB interrupt status registers 1 and 2, perform it for the USB interrupt status register 1 first and then the register 2.
- Make sure the index indicated by the USB endpoint index register is correct when accessing the registers: USB endpoint x (x = 0 to 2) IN control register, USB endpoint x OUT control register, USB endpoint x IN max. packet size register, USB endpoint x OUT max. packet size register, USB endpoint x (x = 0 to 2) OUT write count registers Low and High, USB endpoint FIFO mode register.
- When the USB reset interrupt status flag is kept at "1", all other flags in the USB internal registers (addresses 0050<sub>16</sub> to 005F<sub>16</sub>) will return to their reset status. However, the following registers are not affected by the USB reset: USB control register (address 0013<sub>16</sub>), Frequency synthesizer control register (address 006C<sub>16</sub>), Clock control register (address 001F<sub>16</sub>), and USB endpoint x FIFO register (addresses 0060<sub>16</sub> to 0062<sub>16</sub>).
- When not using the USB function, set the USB line driver supply enable bit of the USB control register (address 0013<sub>16</sub>) to "1" for power supply to the internal circuits (at Vcc = 5 V).
- The IN\_PKT\_RDY Bit can be set by software even when using the AUTO\_SET function.
- Do not write to USB-related registers (addresses 0050<sub>16</sub> to 0062<sub>16</sub>) except the USBC, CCR and FSC until the USB clock is enabled.
- When the MCU is in the USB-suspend state, the USB enable bit is kept "1"; the USB block is enabled. To write to USB-related registers (addresses 0050<sub>16</sub> to 0062<sub>16</sub>) except the USBC, CCR and FSC after returning from the USB-suspend state; after enabling the USB clock, wait for 4 or more φ cycles and then set those registers.
- When using the MCU at Vcc = 3.3V, set the USB line driver supply enable bit to "0" (line driver disable). Note that setting the USB line driver current control bit (USBC3) doesn't affect the USB operation.
- Setting the FLUSH bit to "1" eliminates the data in IN FIFO and OUT FIFO. If there are 2 or more data packets in them, the oldest data is eliminated. The FLUSH bit setting also affects the IN\_PKT\_RDY bit or the OUT\_PKT\_RDY flag.
- If the FLUSH bit is set to "1" while transmission/reception is being performed, the data might be corrupted. When receiving, setting the FLUSH bit must be done while the OUT\_PKT\_RDY flag is "1".



- Use the transfer instructions such as **LDA** and **STA** to set the registers: USB interrupt status registers 1, 2 (addresses 0052<sub>16</sub>, 0053<sub>16</sub>); USB endpoint 0 IN control register (address 0059<sub>16</sub>); USB endpoint x IN control register (address 0059<sub>16</sub>); USB endpoint x OUT control register (address 005A<sub>16</sub>). Do not use the read-modify-write instructions such as the **SEB** or the **CLB** instruction. When writing to bits shown by Table 32 using the transfer instruction such as **LDA** or **STA**, a value which never affect its bit state is required. Take the following sequence to change these bits contents:
  - (1) Store the register contents onto a variable or a data register.
  - (2) Change the target bit on the variable or the data register. Simultaneously mask the bit so that its bit state cannot be changed. (See to Table 3.3.1.)
  - (3) Write the value from the variable or the data register to the register using the transfer instruction such as **LDA** or **STA**.
- To use the AUTO\_SET function for an IN transfer when the AUTO\_SET bit is set to 1, set the FIFO to single buffer mode.

**Table 3.3.1 Bits of which state might be changed owing to software write**

Register name	Bit name	Value not affecting state ( <b>Note</b> )
USB endpoint 0 IN control register	IN_PKT_RDY (b1)	"0"
	DATA_END (b3)	"0"
	FORCE_STALL (b4)	"1"
USB endpoint x (x = 1 to 2) IN control register	IN_PKT_RDY (b0)	"0"
USB endpoint x (x = 1 to 2) OUT control register	OUT_PKT_RDY (b0)	"1"
	FORCE_STALL (b4)	"1"

**Note:** Writing this value will not change the bit state, because this value cannot be written to the bit by software.

### 3.3.6 Notes on frequency synthesizer

- Bits 6 and 5 of the frequency synthesizer control register (address 006C<sub>16</sub>) are initialized to (b6, b5) = "11" after reset release. Make sure to set bits 6 and 5 to "10" after the frequency synthesizer lock status bit goes to "1".
- Use the frequency synthesizer output clocks 2 ms to 5 ms later than setting the frequency synthesizer enable bit to "1" (enabled). After that do not change any register values because it might cause output clocks unstabilized temporarily.
- Make sure to connect a low-pulse filter to the LPF pin when using the frequency synthesizer.
- The frequency synthesizer divide register set value never affects f<sub>USB</sub> frequency.
- When using the f<sub>SYN</sub> as an internal system clock, set the frequency synthesizer divide register so that f<sub>SYN</sub> could be 24 MHz or less.
- When using the frequency synthesized clock function, we recommend using the fastest frequency possible of f(X<sub>IN</sub>) or f(X<sub>CIN</sub>) as an input clock for the PLL.
- Set the value of frequency synthesizer multiply register 2 (FSM2) so that the f<sub>PIN</sub> is 1 MHz or higher.

### 3.3.7 Notes on external devices connection

#### (1) Rewrite port P3 latch

In both memory expansion mode and microprocessor mode, ports P3<sub>1</sub> and P3<sub>2</sub> can be used as output ports. We recommend to use the **LDM** instruction or **STA** instruction to write to port P3 register (address 000E<sub>16</sub>). If using the Read-Modify-Write instruction (**SEB** instruction, **CLB** instruction) you will need to map a memory that the CPU can read from and write to.

#### [Reason]

The access to address 000E<sub>16</sub> is performed:

- Read from external memory
- Write to both port P3 latch and external memory.

It is because address 000E<sub>16</sub> is assigned on an external area In the memory expansion mode and microprocessor mode.

Accordingly, if a Read-Modify-Write instruction is executed to address 000E<sub>16</sub>, the external memory contents is read out and after its modification it will be written into both port P3 latch and an external memory. As a result, if an external memory is not allocated in address 000E<sub>16</sub> then, the MCU will read an undefined value and write its modified value into the port P3 latch. Therefore port P3 latch value will become undefined.

#### (2) overlap of internal and external memories

In the memory expansion mode, if the internal and external memory areas overlap, the internal memory becomes the valid memory for the overlapping area. When the CPU performs a read or a write operation on this overlapped area, the following things happen:

##### •Read

The CPU reads out the data in the internal memory instead of in the external memory. Note that, since the CPU will output a proper read signal, address signal, etc., the memory data at the respective address will appear on the external data bus.

##### •Write

The CPU writes data to both the internal and external memories.

#### (3) $\overline{RD}$ , $\overline{WR}$ pins

In the memory expansion mode or microprocessor mode, a read-out control signal is output from the  $\overline{RD}$  pin (P3<sub>6</sub>), and a write-in control signal is output from the  $\overline{WR}$  pin (P3<sub>7</sub>). "L" level is output from the  $\overline{RD}$  pin at CPU read-out and from the  $\overline{WR}$  pin at CPU write-in. These signals function for internal access and external access.

**(4) RDY function**

When using RDY function in usual connection, it does not operate at 12 MHz of  $\phi$  or faster.

**[Reason]**

$$td(\phi\text{-AH}) + tsu(\text{RDY-}\phi) = 31 \text{ ns (max.)} + 21 \text{ ns (min.)} = 52 \text{ ns.}$$

$$twh(\phi), twl(\phi) = 0.5 \times 83.33 - 5 = 36.665 \text{ ns}$$

Therefore, it becomes  $52 \text{ ns} > 36.665 \text{ ns}$ , so that the timing to enter RDY wait does not match.

However, if the timings can match owing to  $\overline{\text{RDY}}$  pin by "L" fixation and others, the RDY function can be used even at  $\phi = 12 \text{ MHz}$ . In this situation the slow memory wait always functions.

**(5) Wait function**

The Wait function is serviceable at accessing an external memory in the memory expansion mode and microprocessor mode. However, in these modes even if an external memory is assigned to addresses  $0008_{16}$  to  $000F_{16}$ , the Wait function cannot function to these areas.

**(6) Processor mode switch**

Note when the processor mode is switched by setting of the processor mode bits (b1, b0 of CPMA), that will immediately switch the accessible memory from external to internal or from internal to external. If this is done, the first cycle of the next instruction will be operated from the accidental memory.

To prevent this problem, follow the procedure below:

- (a) Duplicate the next instruction at the same address both in internal and external memories.
- (b) Switch from single-chip mode to memory expansion mode, jump to external memory, and then switch from memory expansion mode to microprocessor mode. (Because in general, the problem will not occur when switching the modes as long as the same memory is accessed after the switch.)
- (c) Load a simple program in RAM that switches the modes, jump to RAM and execute the program, then jump to the location of the code to run after the processor mode has switched.

### 3.3.8 Notes on timer

#### (1) Read/Write for timer

- The timer division ratio is :  $1 / (n + 1)$   
( $n = "0"$  to  $"255"$  written into the timer)
- When the value is loaded only in the latch, the value is loaded in the timer at the count pulse following the count where the timer reaches  $"00_{16}"$ .
- In the timers 1 to 3, switching of the count sources of timers 1 to 3 does not affect the values of reload latches. However, that may make count operation started. Therefore, write values again in the order of timers 1, 2 and then timer 3 after their count sources have been switched.
- The timer current count value can be read out by reading the timer.

#### (2) Pulse output

- When using the  $T_{OUT}$  output of timer 1 or timer 2, set bit 1 of port P5 direction register to  $"1"$  (output mode).
- The  $T_{OUT}$  output pin is shared with the  $X_{COUT}$  pin. Accordingly, when using  $f(X_{CIN})/2$  as the timer 1 count source (bit 2 of timer 123 mode register =  $"0"$ ),  $X_{COUT}$  oscillation drive must be disabled (bit 5 of clock control register =  $"1"$ ) to input clocks from the  $X_{CIN}$  pin.
- The  $P5_1/X_{COUT}/T_{OUT}$  pin cannot function as an ordinary I/O port while  $X_{CIN}-X_{COUT}$  is oscillating. When  $X_{CIN}-X_{COUT}$  oscillation is stopped or  $X_{COUT}$  oscillation drive is disabled, this can be used as the  $T_{OUT}$  output pin of timer 1 or 2.

#### (3) At STP instruction executed

When the STP instruction is executed or Reset occurs, the timer 1 is set to  $"FF_{16}"$  and the internal clock  $\phi$  divided by 8 is automatically selected as its count source. Additionally, the timer 2 is set to  $"01_{16}"$  and the timer 1's output is automatically selected as its count source. When the **STP** instruction is being executed, all bits except bit 4 of the timer 123 mode register (address  $0029_{16}$ ) are initialized to  $"0"$ . It is not necessary to set T123M1 (timer 1 count stop bit) to  $"0"$  before executing the **STP** instruction. After returning from Stop mode, reset the timer 1 (address  $0024_{16}$ ), timer 2 (address  $0025_{16}$ ), and the timer 123 mode register (address  $0029_{16}$ ).

### 3.3.9 Notes on Stop mode

- When the STP instruction is executed, bit 7 of the clock control register (address  $001F_{16}$ ) goes to  $"0"$ . To return from stop mode, reset CCR7 to  $"1"$ .
- When using  $f_{SYN}$  (set internal system clock select bit (CPMA6) to  $"1"$ ) as the internal system clock, switch CPMA6 to  $"0"$  before executing the **STP** instruction. Reset CPMA6 after the system returns from Stop Mode and the frequency synthesizer has stabilized.  
CPMA6 does not need to be switched to  $"0"$  when using the **WIT** instruction.
- When the STP instruction is executed or Reset occurs, the timer 1 is set to  $"FF_{16}"$  and the internal clock  $\phi$  divided by 8 is automatically selected as its count source. Additionally, the timer 2 is set to  $"01_{16}"$  and the timer 1's output is automatically selected as its count source. When the **STP** instruction is being executed, all bits except bit 4 of the timer 123 mode register (address  $0029_{16}$ ) are initialized to  $"0"$ . It is not necessary to set T123M1 (timer 1 count stop bit) to  $"0"$  before executing the **STP** instruction. After returning from Stop mode, reset the timer 1 (address  $0024_{16}$ ), timer 2 (address  $0025_{16}$ ), and the timer 123 mode register (address  $0029_{16}$ ).

### 3.3.10 Notes on reset

#### (1) Connecting capacitor

In case where the  $\overline{\text{RESET}}$  signal rise time is long, connect a ceramic capacitor or others across the  $\overline{\text{RESET}}$  pin and the Vss pin. Use a 1000 pF or more capacitor for high frequency use. When connecting the capacitor, note the following :

- Make the length of the wiring which is connected to a capacitor as short as possible.
- Be sure to verify the operation of application products on the user side.

#### ● Reason

If the several nanosecond or several ten nanosecond impulse noise enters the  $\overline{\text{RESET}}$  pin, it may cause a microcomputer failure.

### 3.3.11 Notes on I/O port

#### (1) Notes in standby state

In standby state\*1 for low-power dissipation, do not make input levels of an I/O port “undefined”. Pull-up (connect the port to VCC) or pull-down (connect the port to Vss) these ports through a resistor. When determining a resistance value, note the following points:

- External circuit
- Variation of output levels during the ordinary operation

When using built-in pull-up resistor, note on varied current values:

- When setting as an input port : Fix its input level
- When setting as an output port : Prevent current from flowing out to external

#### ● Reason

The potential which is input to the input buffer in a microcomputer is unstable in the state that input levels of an I/O port are “undefined”. This may cause power source current.

\*1 standby state: stop mode by executing **STP** instruction  
wait mode by executing **WIT** instruction

#### (2) Modifying output data with bit managing instruction

When the port latch of an I/O port is modified with the bit managing instruction\*2, the value of the unspecified bit may be changed.

#### ● Reason

The bit managing instructions are read-modify-write form instructions for reading and writing data by a byte unit. Accordingly, when these instructions are executed on a bit of the port latch of an I/O port, the following is executed to all bits of the port latch.

- As for bit which is set for input port:  
The pin state is read in the CPU, and is written to this bit after bit managing.
- As for bit which is set for output port:  
The bit value is read in the CPU, and is written to this bit after bit managing.

Note the following:

- Even when a port which is set as an output port is changed for an input port, its port latch holds the output data.
- As for a bit of which is set for an input port, its value may be changed even when not specified with a bit managing instruction in case where the pin state differs from its port latch contents.

\*2 Bit managing instructions: **SEB** and **CLB** instructions

**(3) Pull-up control**

When using port P2, which includes a pull-up resistor, as an output port, its port pull-up control is invalidated, that is, pull-up cannot be enabled.

**● Reason**

Pull-up/pull-down control is valid only when each direction register is set to the input mode.

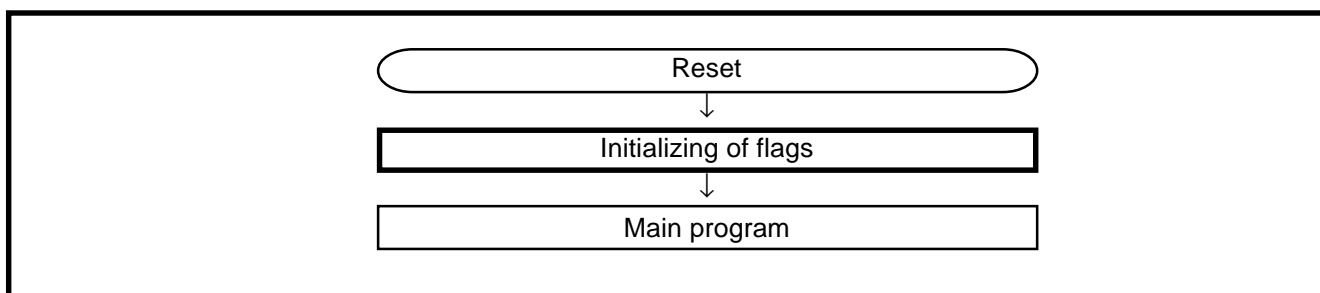
**3.3.12 Notes on programming****(1) Processor status register****① Initializing of processor status register**

Flags which affect program execution must be initialized after a reset.

In particular, it is essential to initialize the T and D flags because they have an important effect on calculations.

**● Reason**

After a reset, the contents of the processor status register (PS) are undefined except for the I flag which is "1".



**Fig. 3.3.5 Initialization of processor status register**

## ② How to reference the processor status register

To reference the contents of the processor status register (PS), execute the **PHP** instruction once then read the contents of (S+1). If necessary, execute the **PLP** instruction to return the PS to its original status. A **NOP** instruction should be executed after every **PLP** instruction.

Be sure to execute the **SEI** instruction before the **PLP** instruction. If executing the **CLI** instruction, do it after the **NOP** instruction

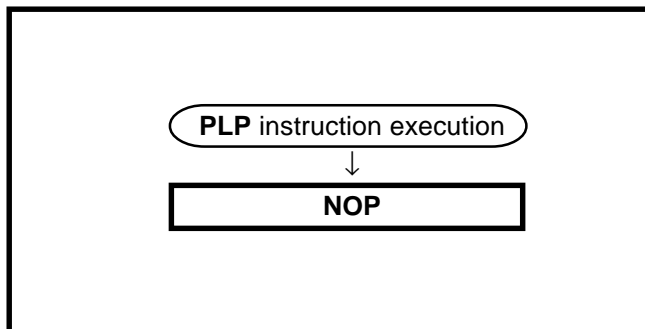


Fig. 3.3.6 Sequence of PLP instruction execution

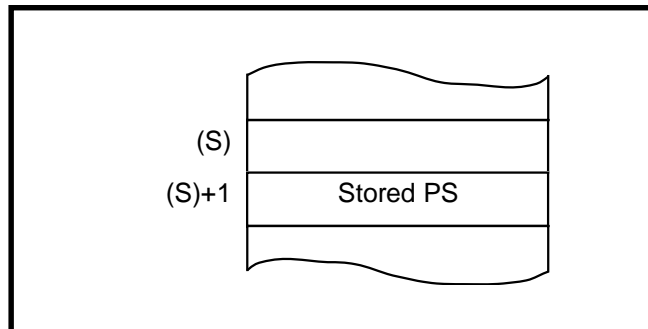


Fig. 3.3.7 Stack memory contents after PHP instruction execution

## (2) BRK Instruction

It can be detected that the **BRK** instruction interrupt event or the least priority interrupt event by referring the stored B flag state. Refer to the stored B flag state in the interrupt routine.

## (3) Decimal Calculations

When decimal mode is selected, the values of the V flags are invalid.

The carry flag (C) is set to "1" if a carry is generated as a result of the calculation, or is cleared to "0" if a borrow is generated. To determine whether a calculation has generated a carry, the C flag must be initialized to "0" before each calculation. To check for a borrow, the C flag must be initialized to "1" before each calculation.

## (4) Multiplication and Division Instructions

The index X mode (T) and the decimal mode (D) flags do not affect the MUL and DIV instruction.

## (5) Instruction Execution Time

The instruction execution time is obtained by multiplying the frequency of the internal clock  $\phi$  by the number of cycles needed to execute an instruction.

The number of cycles required to execute an instruction is shown in the list of machine instructions.

### 3.3.13 Termination of unused pins

#### (1) Terminate unused pins

① I/O ports :

- Set the I/O ports for the input mode and connect them to VCC or VSS through each resistor of 1 k $\Omega$  to 10 k $\Omega$ .  
Ports that permit the selecting of a built-in pull-up resistor can also use this resistor. Set the I/O ports for the output mode and open them at “L” or “H”.
- When opening them in the output mode, the input mode of the initial status remains until the mode of the ports is switched over to the output mode by the program after reset. Thus, the potential at these pins is undefined and the power source current may increase in the input mode. With regard to an effects on the system, thoroughly perform system evaluation on the user side.
- Since the direction register setup may be changed because of a program runaway or noise, set direction registers by program periodically to increase the reliability of program.

#### (2) Termination remarks

① I/O ports :

Do not open in the input mode.

● Reason

- The power source current may increase depending on the first-stage circuit.
- An effect due to noise may be easily produced as compared with proper termination ② and shown on the above.

② I/O ports :

When setting for the input mode, do not connect to VCC or VSS directly.

● Reason

If the direction register setup changes for the output mode because of a program runaway or noise, a short circuit may occur between a port and VCC (or VSS).

③ I/O ports :

When setting for the input mode, do not connect multiple ports in a lump to VCC or VSS through a resistor.

● Reason

If the direction register setup changes for the output mode because of a program runaway or noise, a short circuit may occur between ports.

- At the termination of unused pins, perform wiring at the shortest possible distance (20 mm or less) from microcomputer pins.



### 3.3.14 Notes on CPU rewrite mode for flash memory version

The below notes applies when rewriting the flash memory in CPU rewrite mode.

#### (1) Operation speed

During CPU rewrite mode, set the internal clock  $\phi$  to 6 MHz or less using the  $X_{IN}$  divider select bit (bit 7 of address 001F<sub>16</sub>).

#### (2) Instructions inhibited against use

The instructions which refer to the internal data of the flash memory cannot be used during CPU rewrite mode .

#### (3) Interrupts inhibited against use

The interrupts cannot be used during CPU rewrite mode because they refer to the internal data of the flash memory.

#### (4) Reset

Reset is always valid. When  $CNV_{SS}$  is "H" at reset release, the program starts from the address stored in addresses FFFA<sub>16</sub> and FFFB<sub>16</sub> of the boot ROM area in order that CPU may start in boot mode.

### 3.4 Countermeasures against noise

Countermeasures against noise are described below. The following countermeasures are effective against noise in theory, however, it is necessary not only to take measures as follows but to evaluate before actual use.

#### 3.4.1 Shortest wiring length

The wiring on a printed circuit board can function as an antenna which feeds noise into the microcomputer. The shorter the total wiring length (by mm unit), the less the possibility of noise insertion into a microcomputer.

##### (1) Wiring for $\overline{\text{RESET}}$ pin

Make the length of wiring which is connected to the  $\overline{\text{RESET}}$  pin as short as possible. Especially, connect a capacitor across the  $\overline{\text{RESET}}$  pin and the  $V_{\text{SS}}$  pin with the shortest possible wiring (within 20mm).

##### ● Reason

The width of a pulse input into the  $\overline{\text{RESET}}$  pin is determined by the timing necessary conditions. If noise having a shorter pulse width than the standard is input to the  $\overline{\text{RESET}}$  pin, the reset is released before the internal state of the microcomputer is completely initialized. This may cause a program runaway.

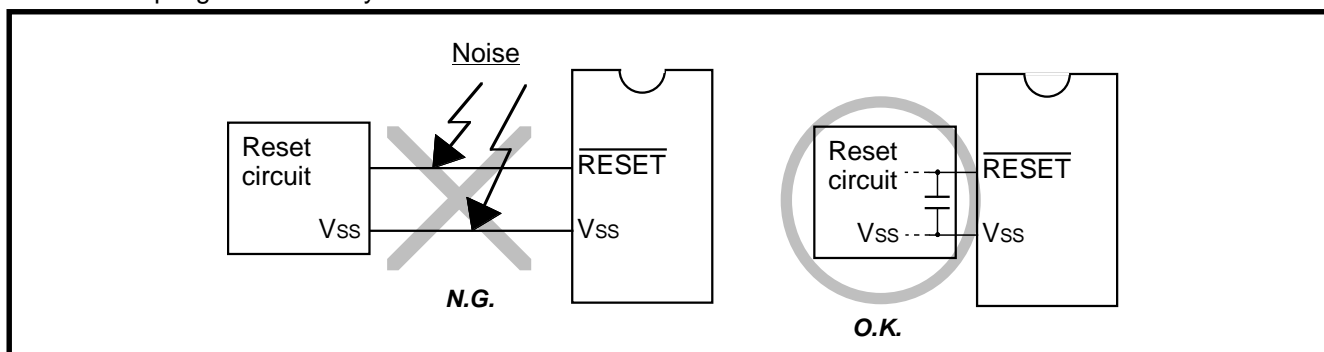


Fig. 3.4.1 Wiring for the  $\overline{\text{RESET}}$  pin

##### (2) Wiring for clock input/output pins

- Make the length of wiring which is connected to clock I/O pins as short as possible.
- Make the length of wiring (within 20 mm) across the grounding lead of a capacitor which is connected to an oscillator and the  $V_{\text{SS}}$  pin of a microcomputer as short as possible.
- Separate the  $V_{\text{SS}}$  pattern only for oscillation from other  $V_{\text{SS}}$  patterns.

##### ● Reason

If noise enters clock I/O pins, clock waveforms may be deformed. This may cause a program failure or program runaway. Also, if a potential difference is caused by the noise between the  $V_{\text{SS}}$  level of a microcomputer and the  $V_{\text{SS}}$  level of an oscillator, the correct clock will not be input in the microcomputer.

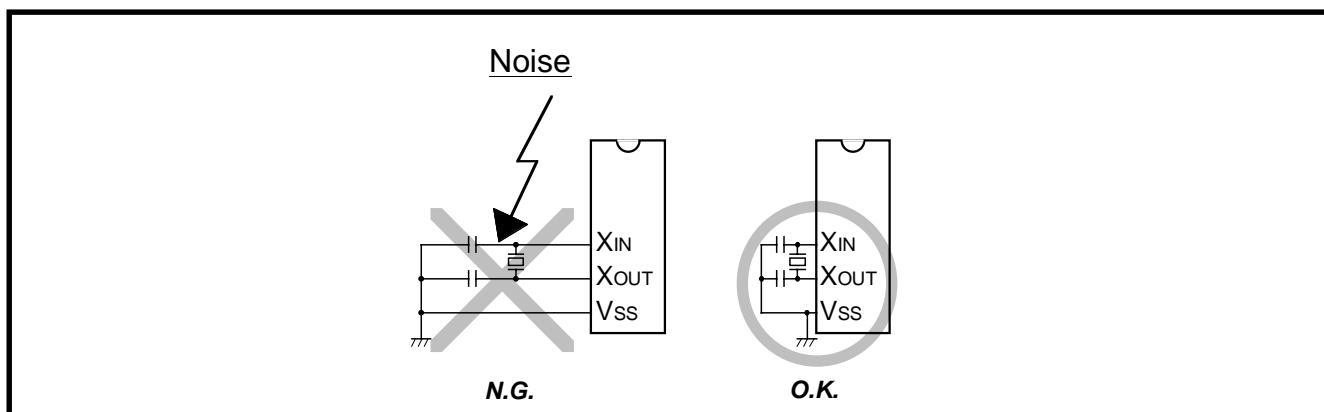


Fig. 3.4.2 Wiring for clock I/O pins

### 3.4.2 Connection of bypass capacitor across Vss line and Vcc line

Connect an approximately  $0.1 \mu\text{F}$  bypass capacitor across the Vss line and the Vcc line as follows:

- Connect a bypass capacitor across the Vss pin and the Vcc pin at equal length.
- Connect a bypass capacitor across the Vss pin and the Vcc pin with the shortest possible wiring.
- Use lines with a larger diameter than other signal lines for Vss line and Vcc line.
- Connect the power source wiring via a bypass capacitor to the Vss pin and the Vcc pin.

In use of the 7643 group it is recommended to connect  $0.1 \mu\text{F}$  and  $4.7 \mu\text{F}$  capacitors in parallel between pins Vcc and Vss, and pins AVss and AVcc. However, their capacitors must not be allocated near the LPF pin.

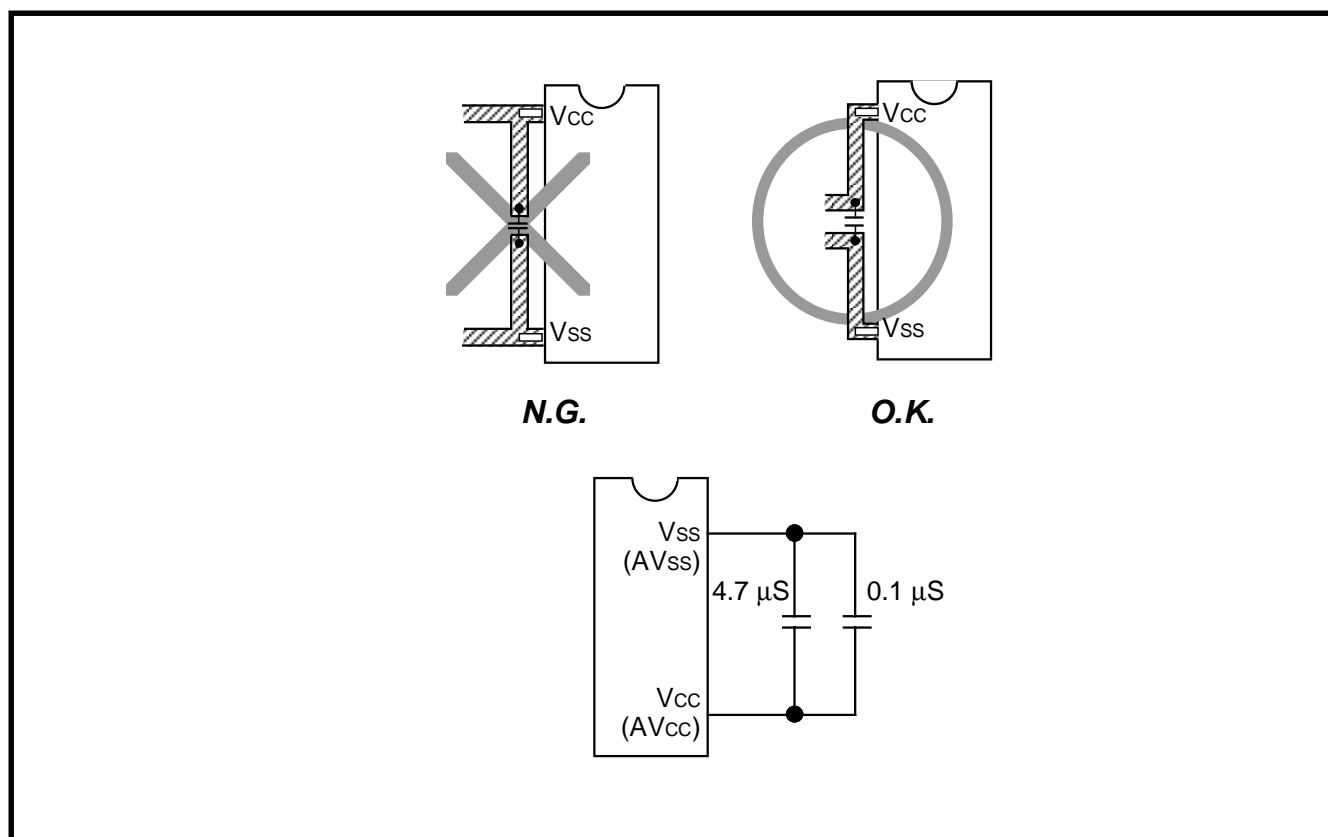


Fig. 3.4.3 Bypass capacitor across the Vss line and the Vcc line

### 3.4.3 Oscillator concerns

Take care to prevent an oscillator that generates clocks for a microcomputer operation from being affected by other signals.

#### (1) Keeping oscillator away from large current signal lines

Install a microcomputer (and especially an oscillator) as far as possible from signal lines where a current larger than the tolerance of current value flows.

##### ● Reason

In the system using a microcomputer, there are signal lines for controlling motors, LEDs, and thermal heads or others. When a large current flows through those signal lines, strong noise occurs because of mutual inductance.

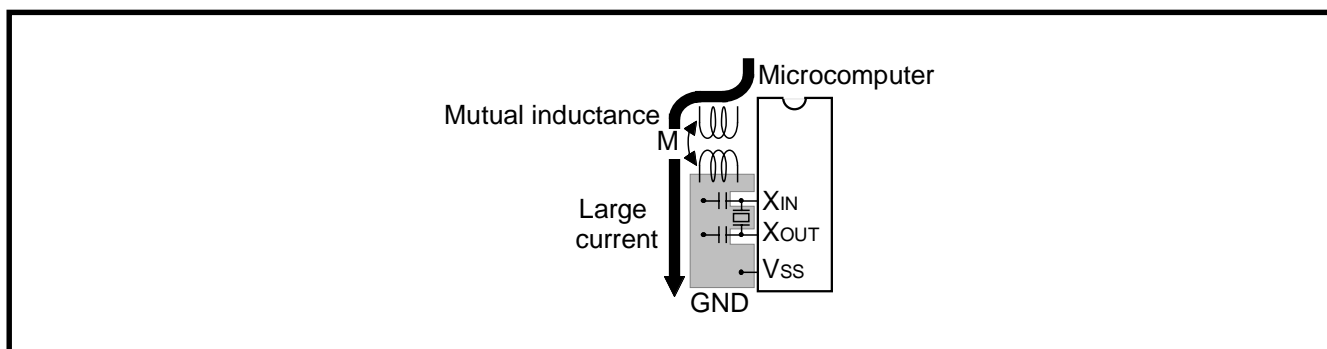


Fig. 3.4.4 Wiring for a large current signal line

#### (2) Installing oscillator away from signal lines where potential levels change frequently

Install an oscillator and a connecting pattern of an oscillator away from signal lines where potential levels change frequently. Also, do not cross such signal lines over the clock lines or the signal lines which are sensitive to noise.

##### ● Reason

Signal lines where potential levels change frequently (such as the T<sub>OUT</sub> pin signal line) may affect other lines at signal rising edge or falling edge. If such lines cross over a clock line, clock waveforms may be deformed, which causes a microcomputer failure or a program runaway.

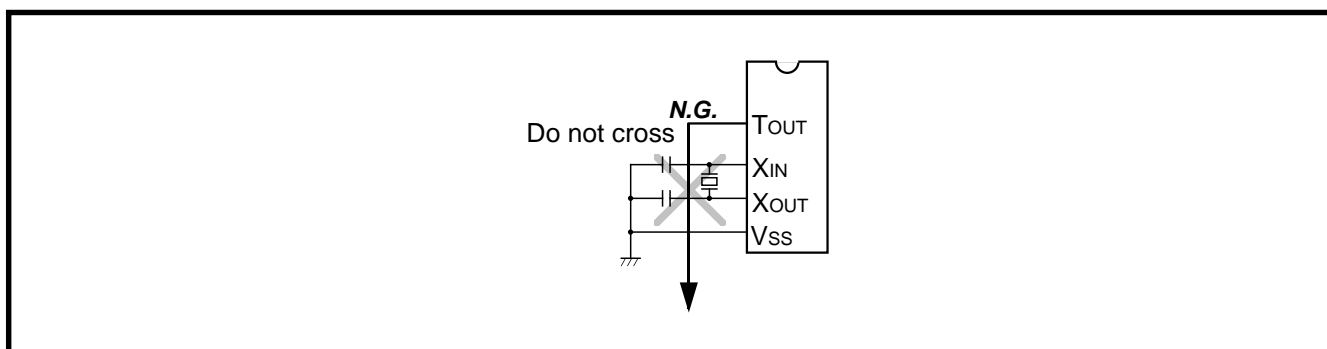


Fig. 3.4.5 Wiring for signal lines where potential levels change frequently

### (3) Oscillator protection using Vss pattern

As for a two-sided printed circuit board, print a Vss pattern on the underside (soldering side) of the position (on the component side) where an oscillator is mounted.

Connect the Vss pattern to the microcomputer Vss pin with the shortest possible wiring. Besides, separate this Vss pattern from other Vss patterns.

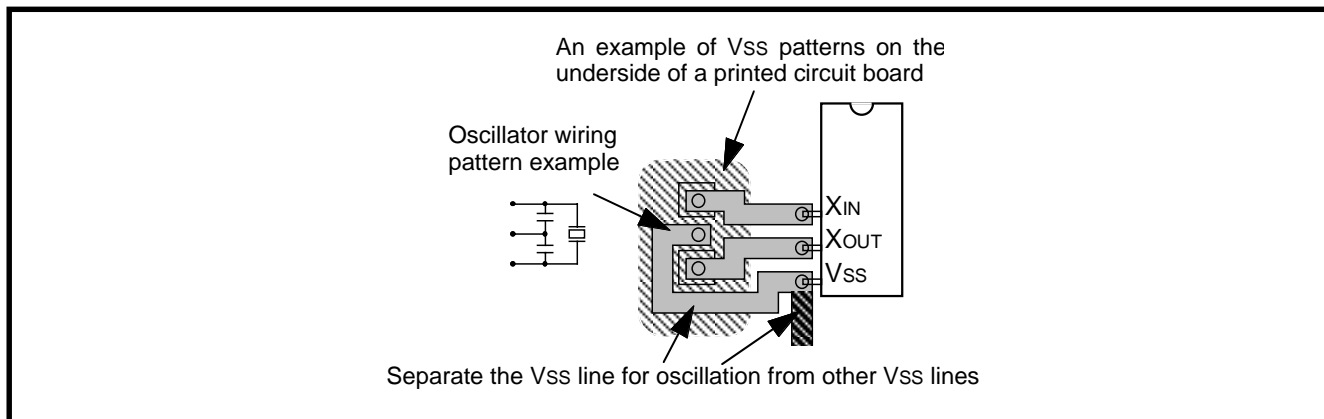


Fig. 3.4.6 Vss pattern on the underside of an oscillator

#### 3.4.4 Setup for I/O ports

Setup I/O ports using hardware and software as follows:

##### <Hardware>

- Connect a resistor of 100  $\Omega$  or more to an I/O port in series.

##### <Software>

- As for an input port, read data several times by a program for checking whether input levels are equal or not.
- As for an output port, since the output data may reverse because of noise, rewrite data to its port latch at fixed periods.
- Rewrite data to direction registers at fixed periods.

**Note:** When a direction register is set for input port again at fixed periods, a several-nanosecond short pulse may be output from this port. If this is undesirable, connect a capacitor to this port to remove the noise pulse.

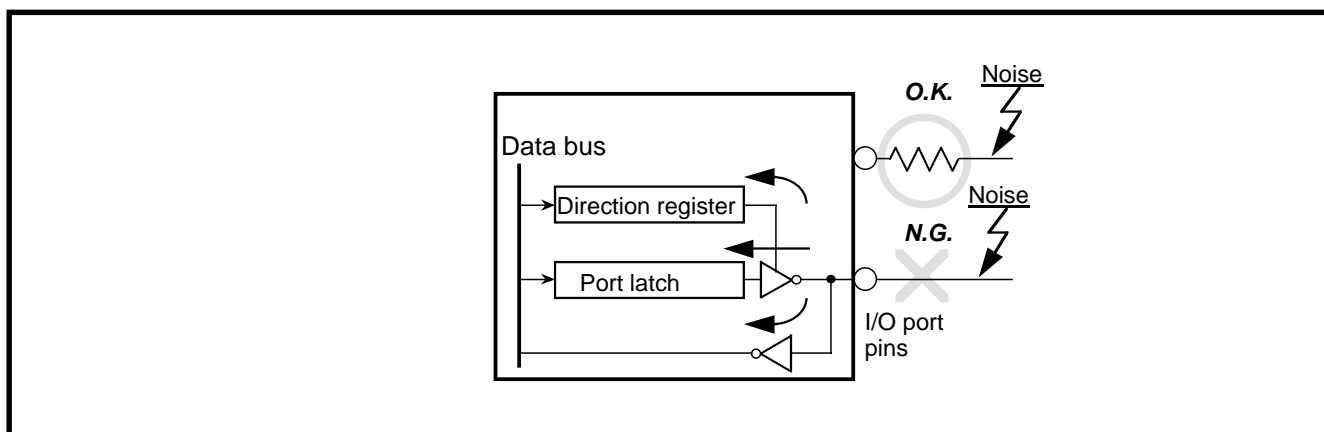


Fig. 3.4.7 Setup for I/O ports

### 3.4.5 Providing of watchdog timer function by software

If a microcomputer runs away because of noise or others, it can be detected by a software watchdog timer and the microcomputer can be reset to normal operation. This is equal to or more effective than program runaway detection by a hardware watchdog timer. The following shows an example of a watchdog timer provided by software.

In the following example, to reset a microcomputer to normal operation, the main routine detects errors of the interrupt processing routine and the interrupt processing routine detects errors of the main routine. This example assumes that interrupt processing is repeated multiple times in a single main routine processing.

#### <The main routine>

- Assigns a single byte of RAM to a software watchdog timer (SWDT) and writes the initial value N in the SWDT once at each execution of the main routine. The initial value N should satisfy the following condition:  

$$N+1 \geq (\text{Counts of interrupt processing executed in each main routine})$$
 As the main routine execution cycle may change because of an interrupt processing or others, the initial value N should have a margin.
- Watches the operation of the interrupt processing routine by comparing the SWDT contents with counts of interrupt processing after the initial value N has been set.
- Detects that the interrupt processing routine has failed and determines to branch to the program initialization routine for recovery processing in the following case:  
 If the SWDT contents do not change after interrupt processing.

#### <The interrupt processing routine>

- Decrements the SWDT contents by 1 at each interrupt processing.
- Determines that the main routine operates normally when the SWDT contents are reset to the initial value N at almost fixed cycles (at the fixed interrupt processing count).
- Detects that the main routine has failed and determines to branch to the program initialization routine for recovery processing in the following case:  
 If the SWDT contents are not initialized to the initial value N but continued to decrement and if they reach 0 or less.

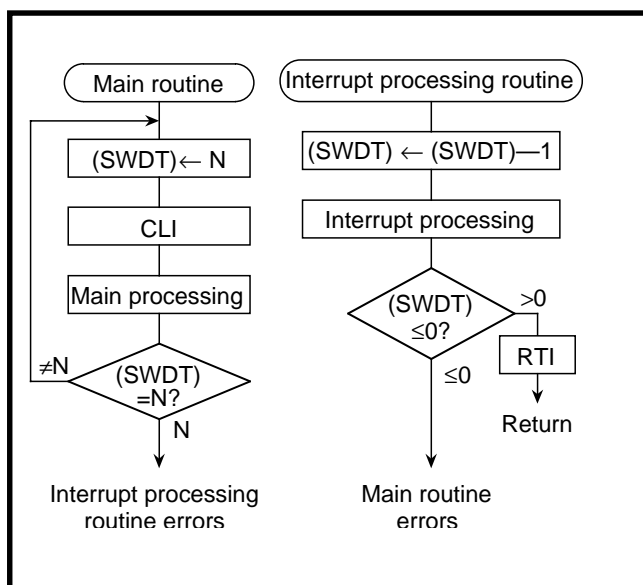


Fig. 3.4.8 Watchdog timer by software

## 3.5 Control registers

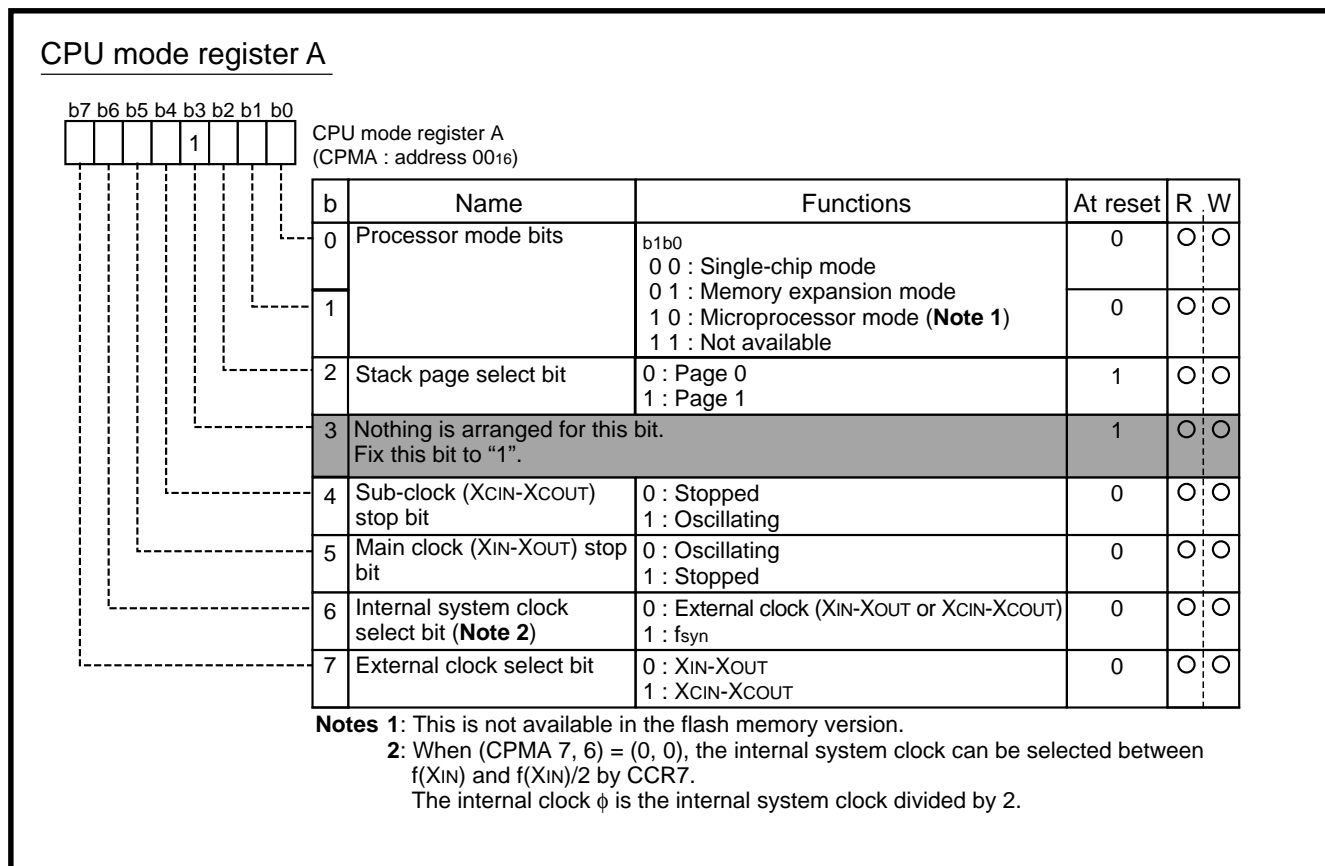


Fig. 3.5.1 Structure of CPU mode register A

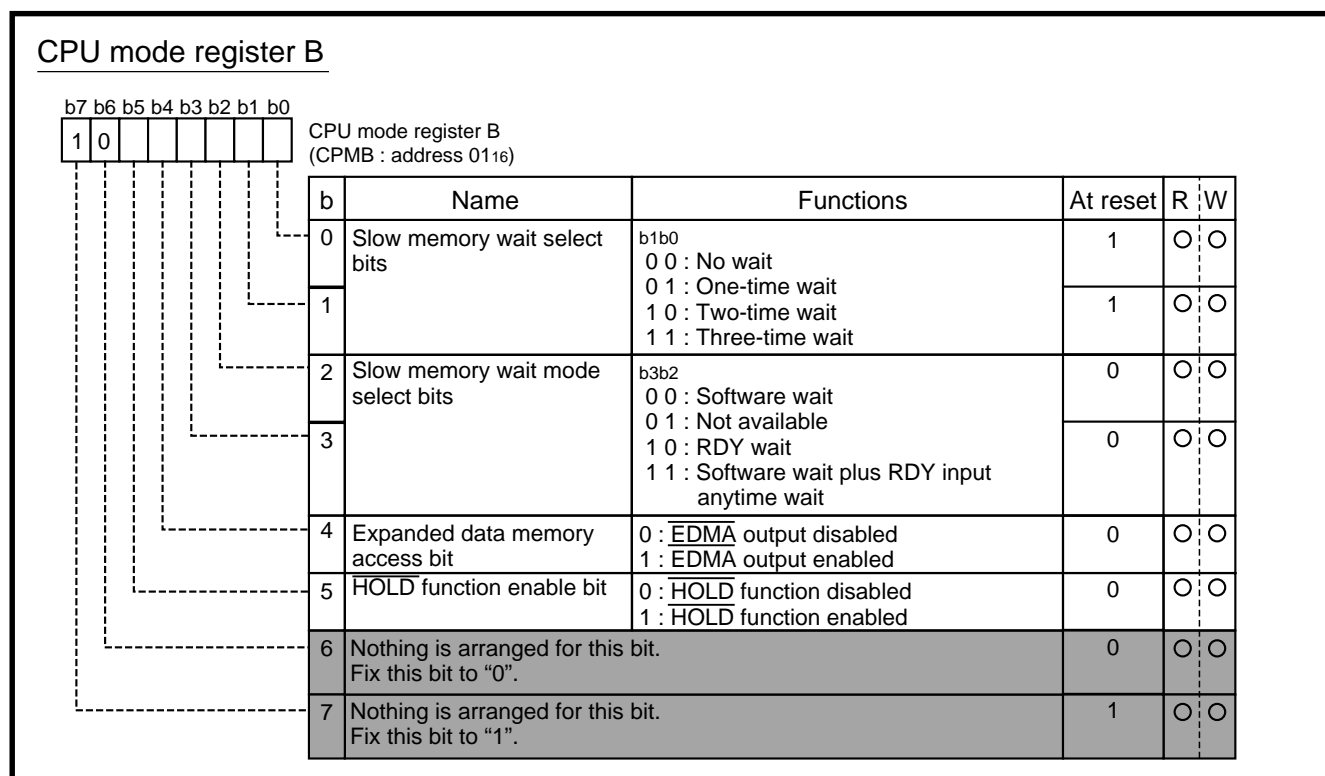


Fig. 3.5.2 Structure of CPU mode register B

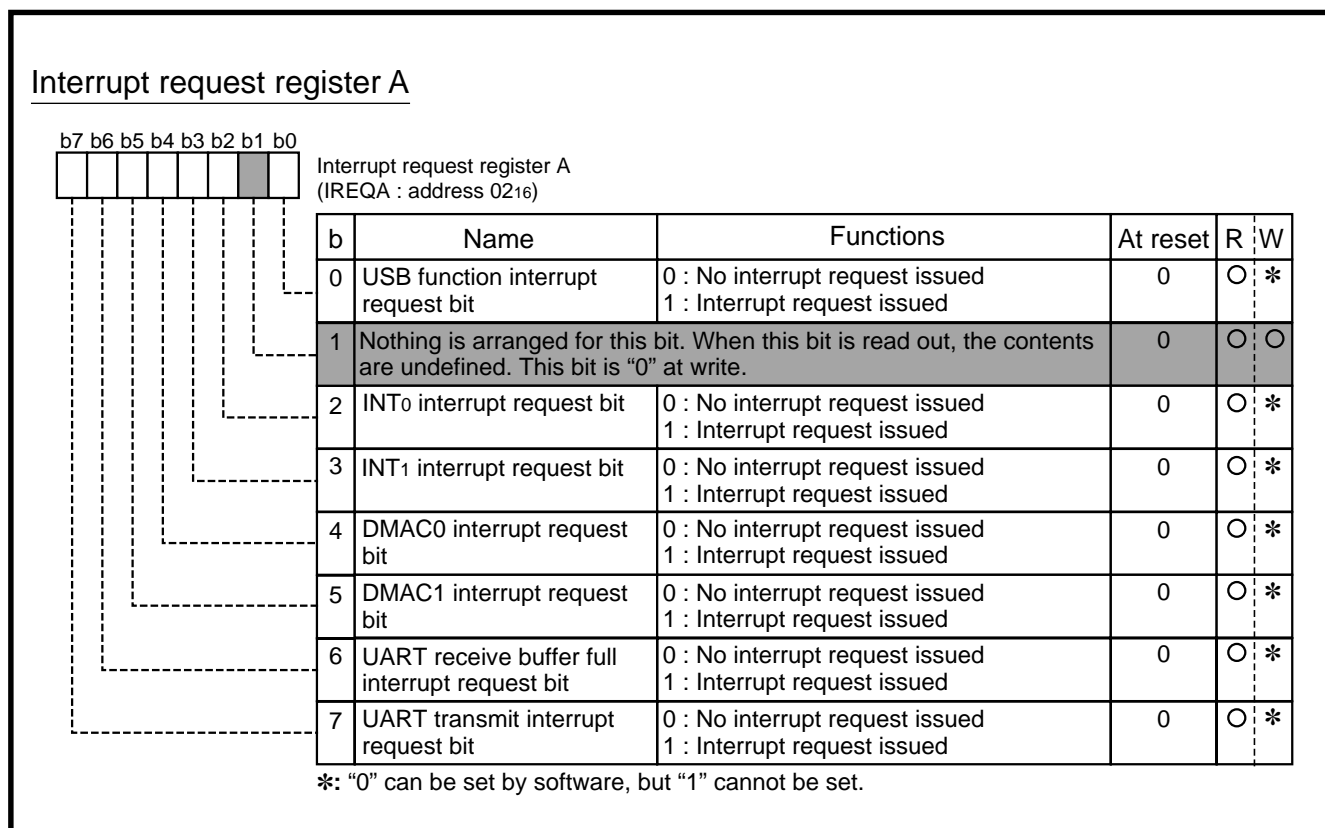


Fig. 3.5.3 Structure of Interrupt request register A

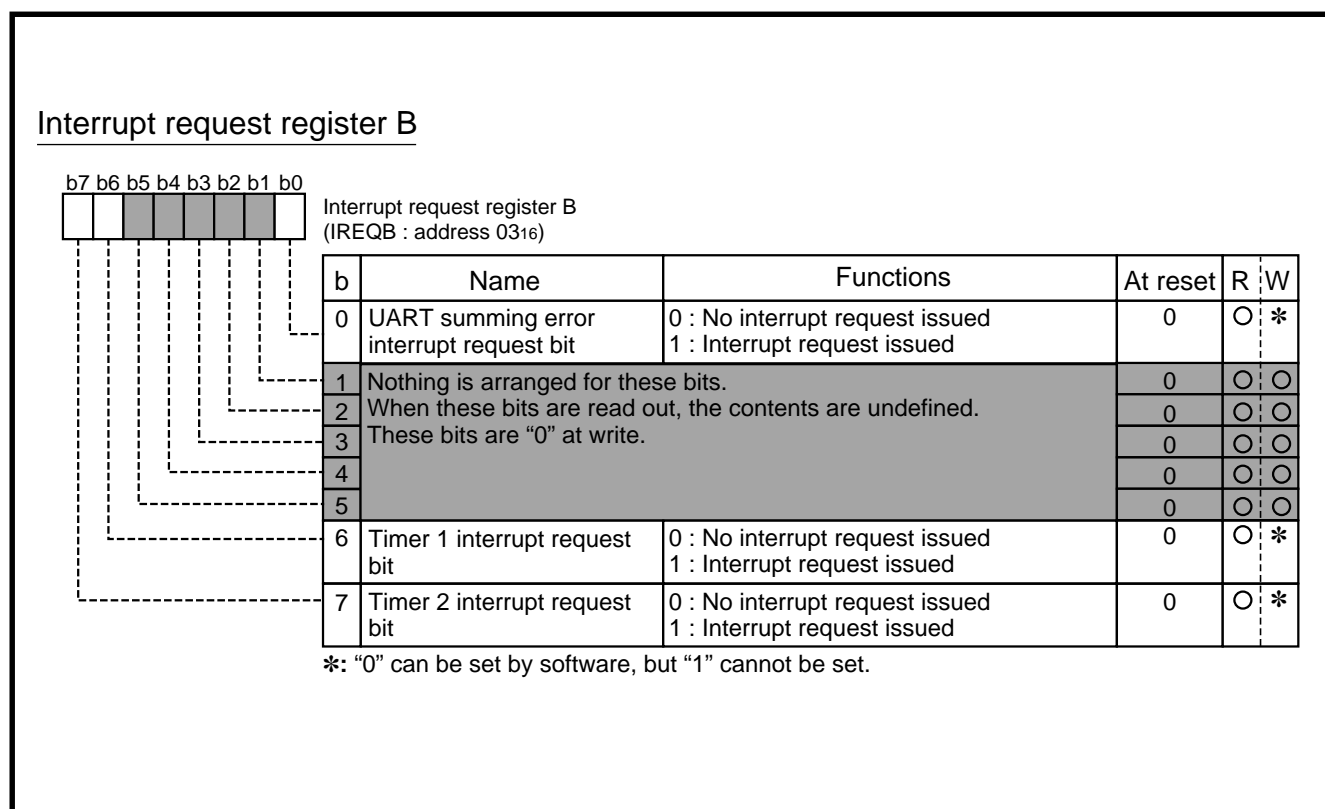


Fig. 3.5.4 Structure of Interrupt request register B



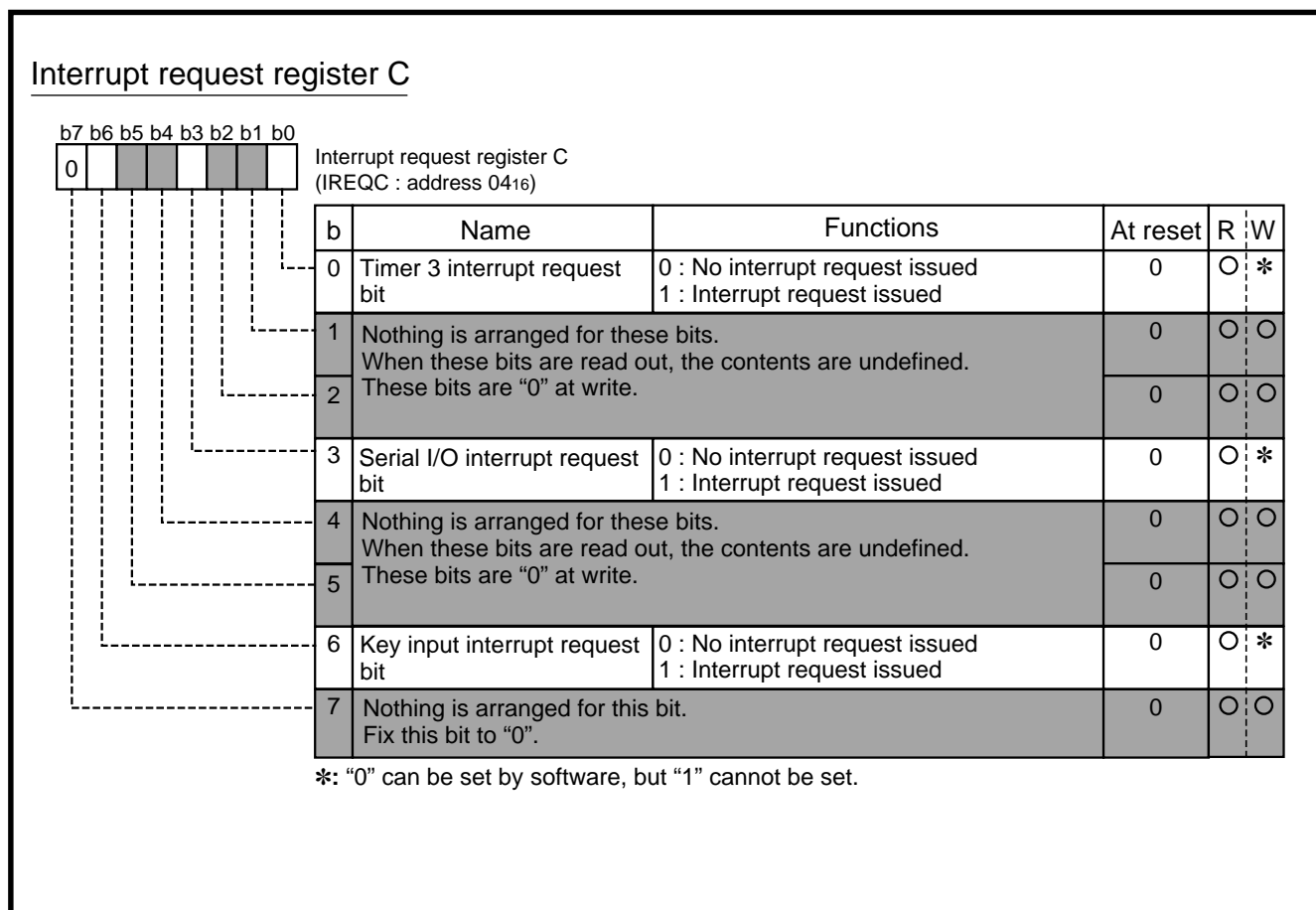


Fig. 3.5.5 Structure of Interrupt request register C

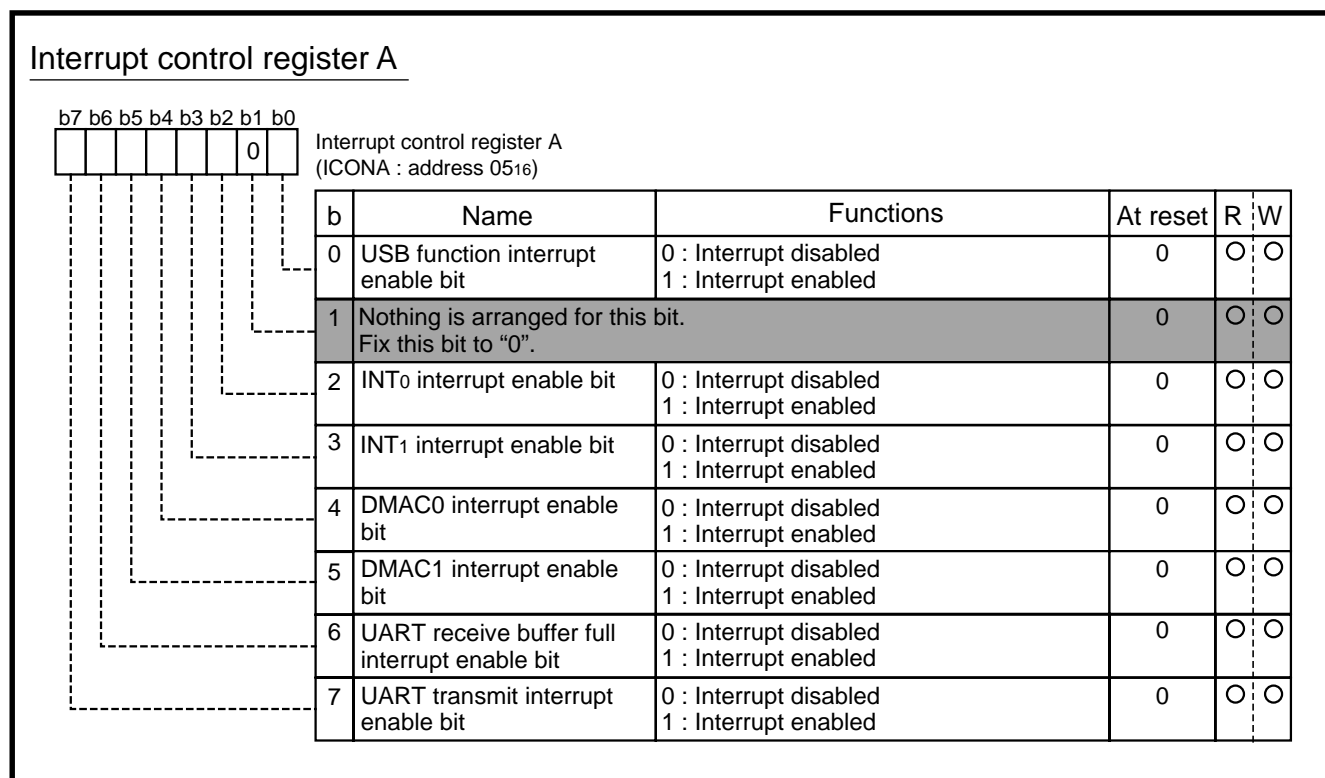


Fig. 3.5.6 Structure of Interrupt control register A

### Interrupt control register B

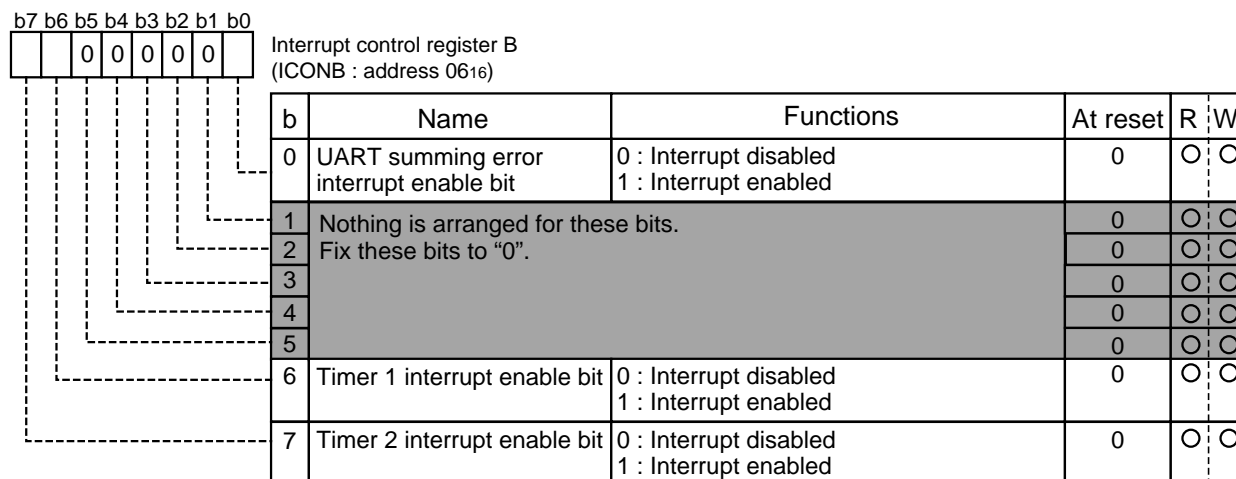


Fig. 3.5.7 Structure of Interrupt control register B

### Interrupt control register C

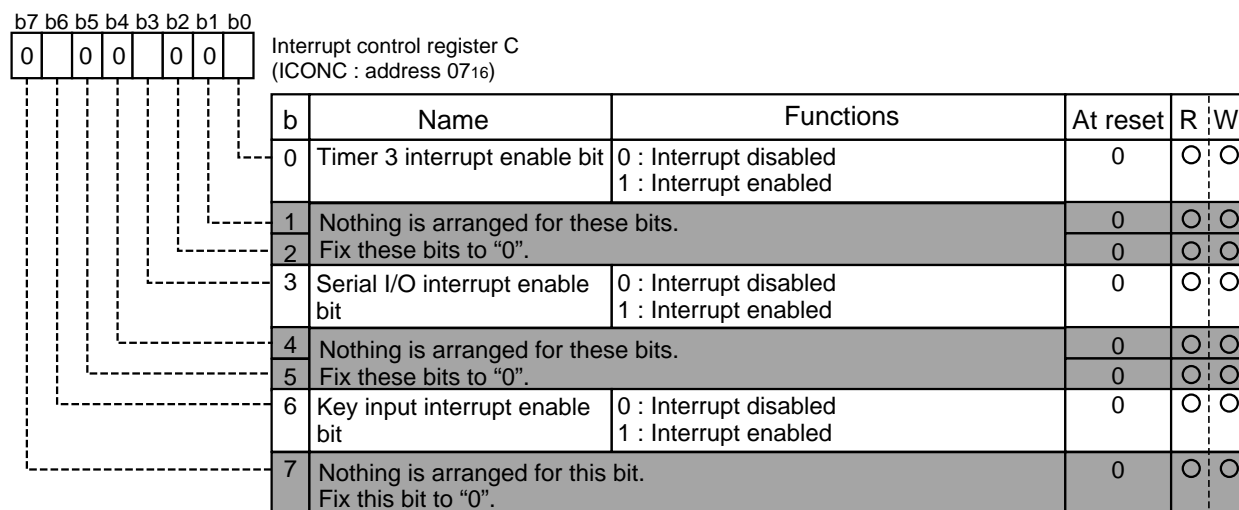


Fig. 3.5.8 Structure of Interrupt control register C

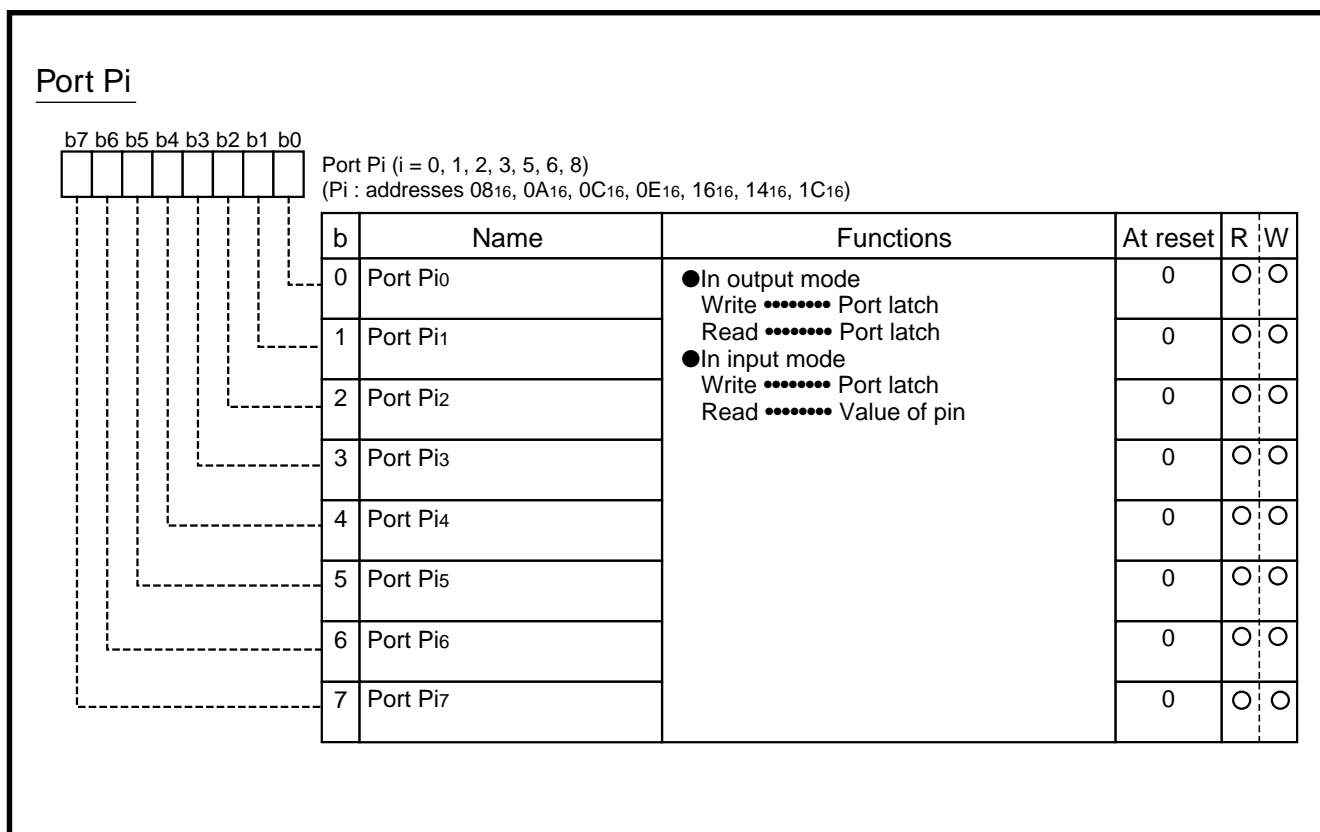


Fig. 3.5.9 Structure of Port Pi

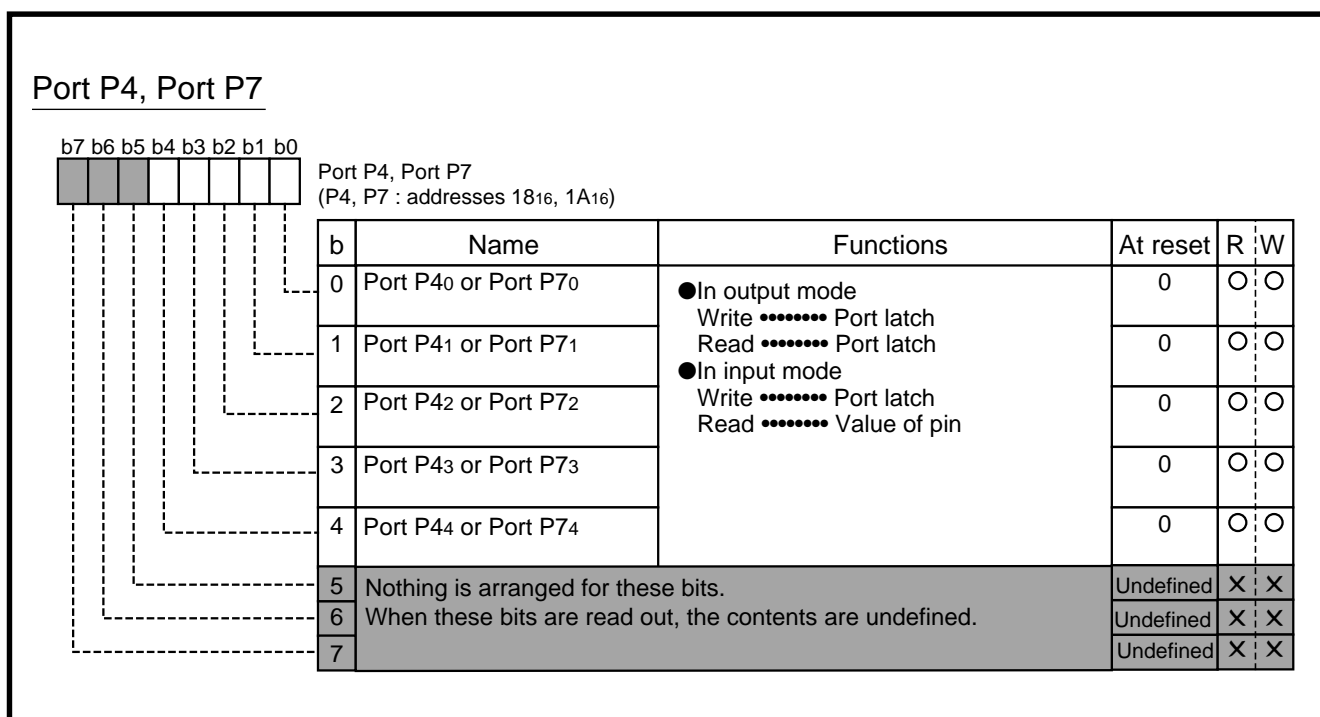


Fig. 3.5.10 Structure of Port P4, Port P7

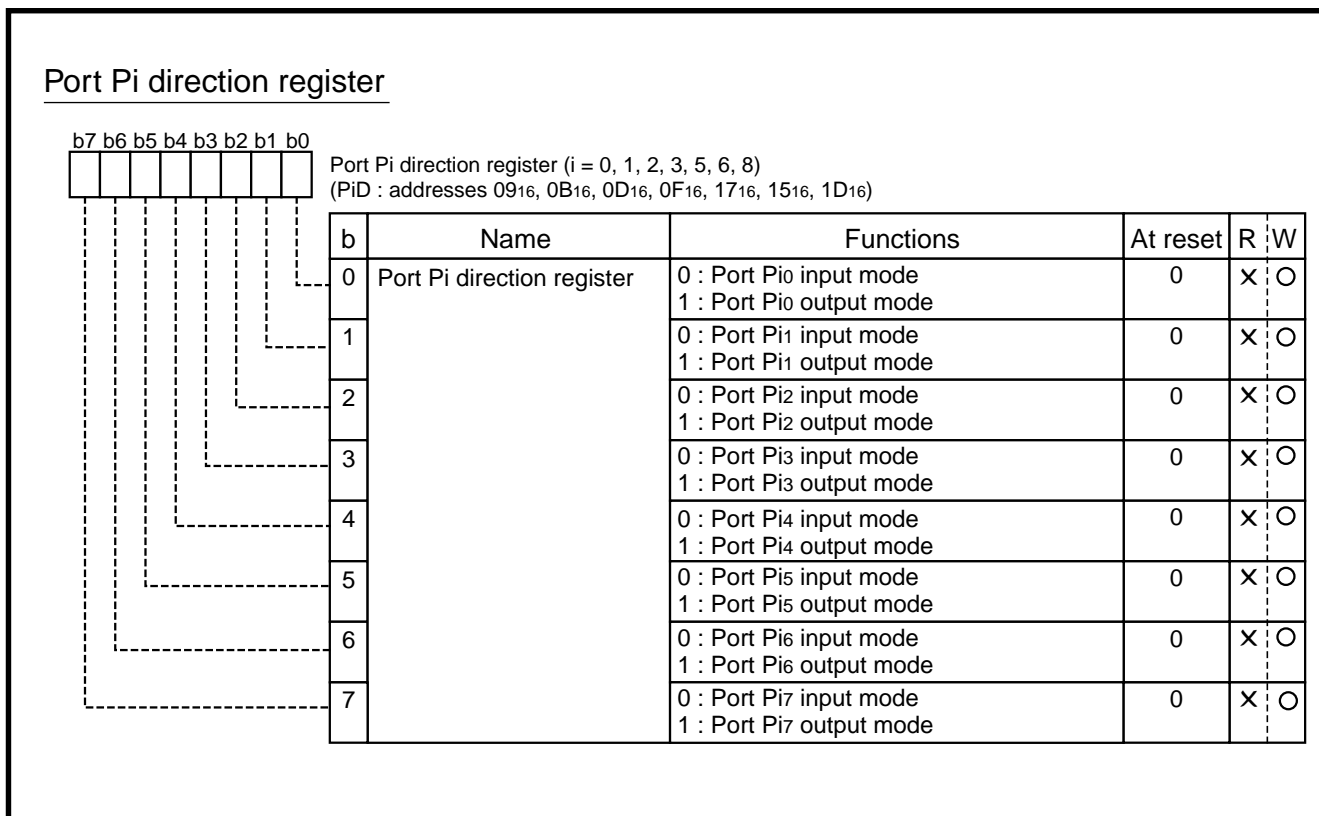


Fig. 3.5.11 Structure of Port Pi direction register

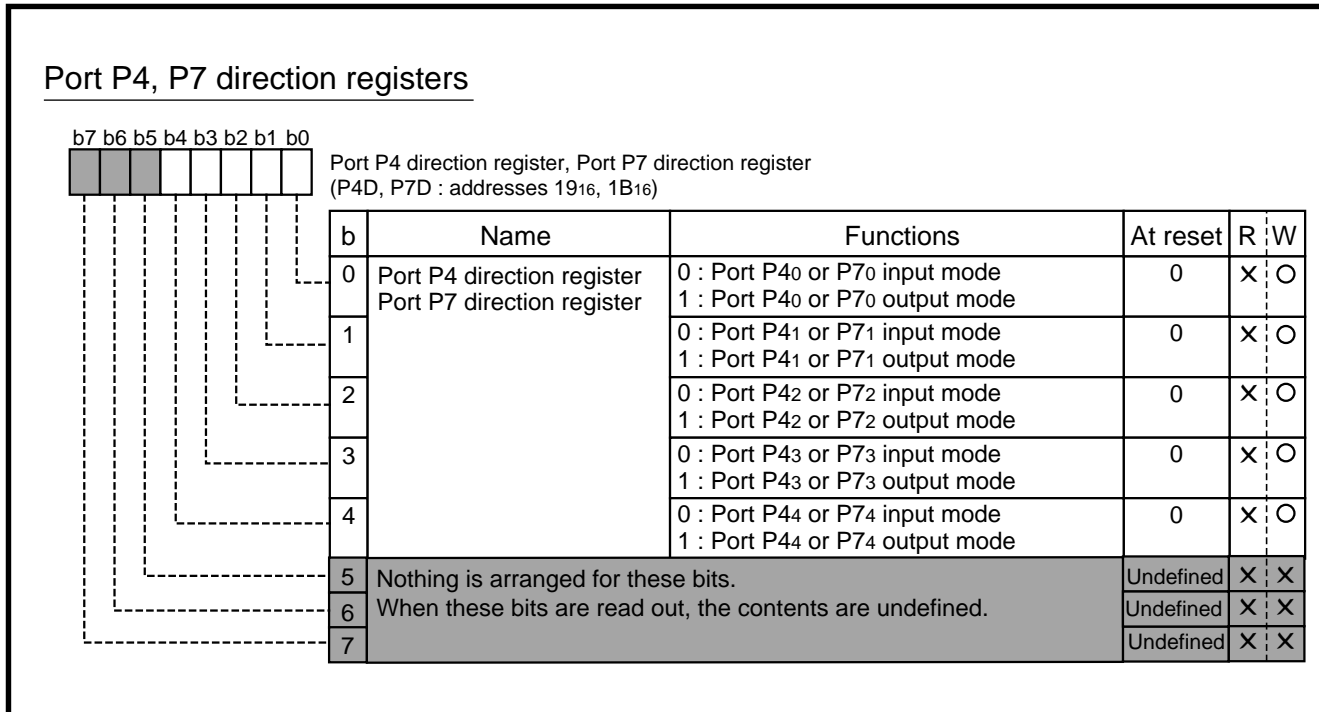


Fig. 3.5.12 Structure of Port P4, Port P7 direction registers

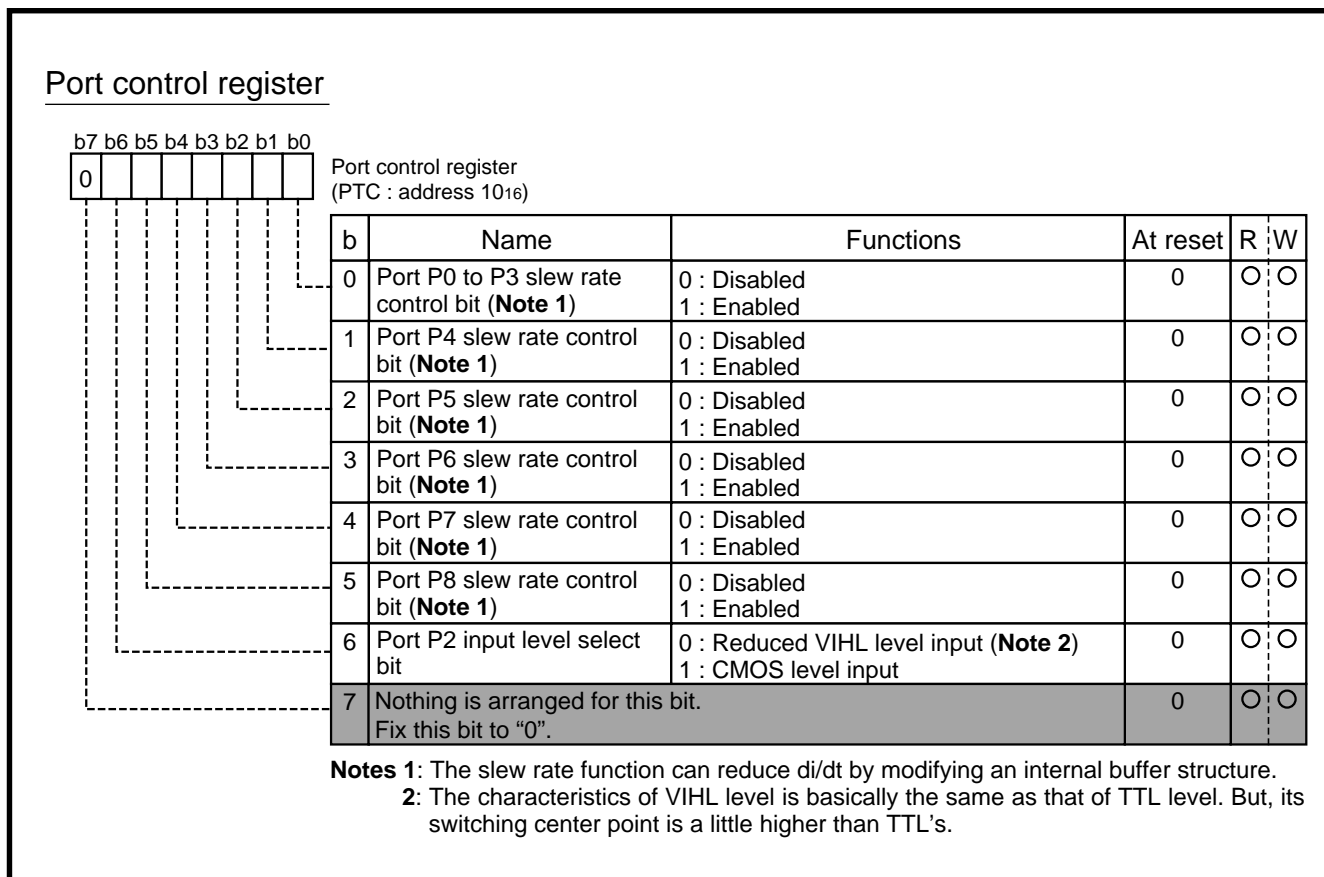


Fig. 3.5.13 Structure of Port control register

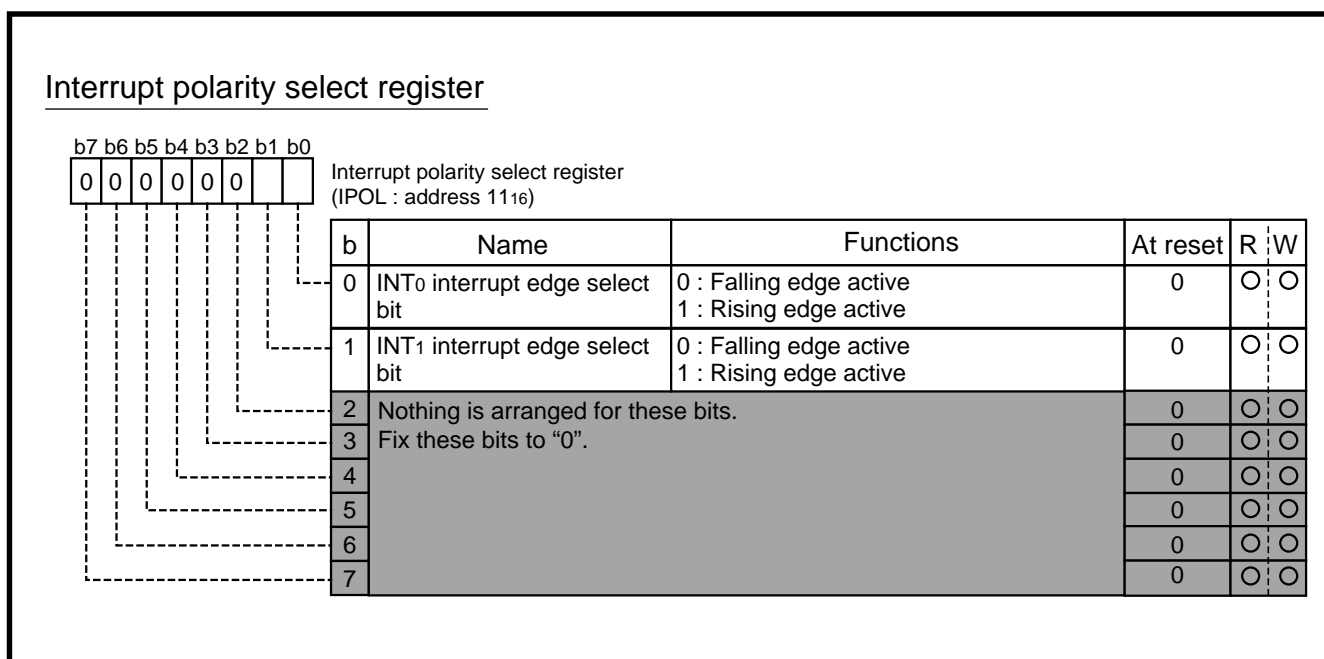


Fig. 3.5.14 Structure of Interrupt polarity select register

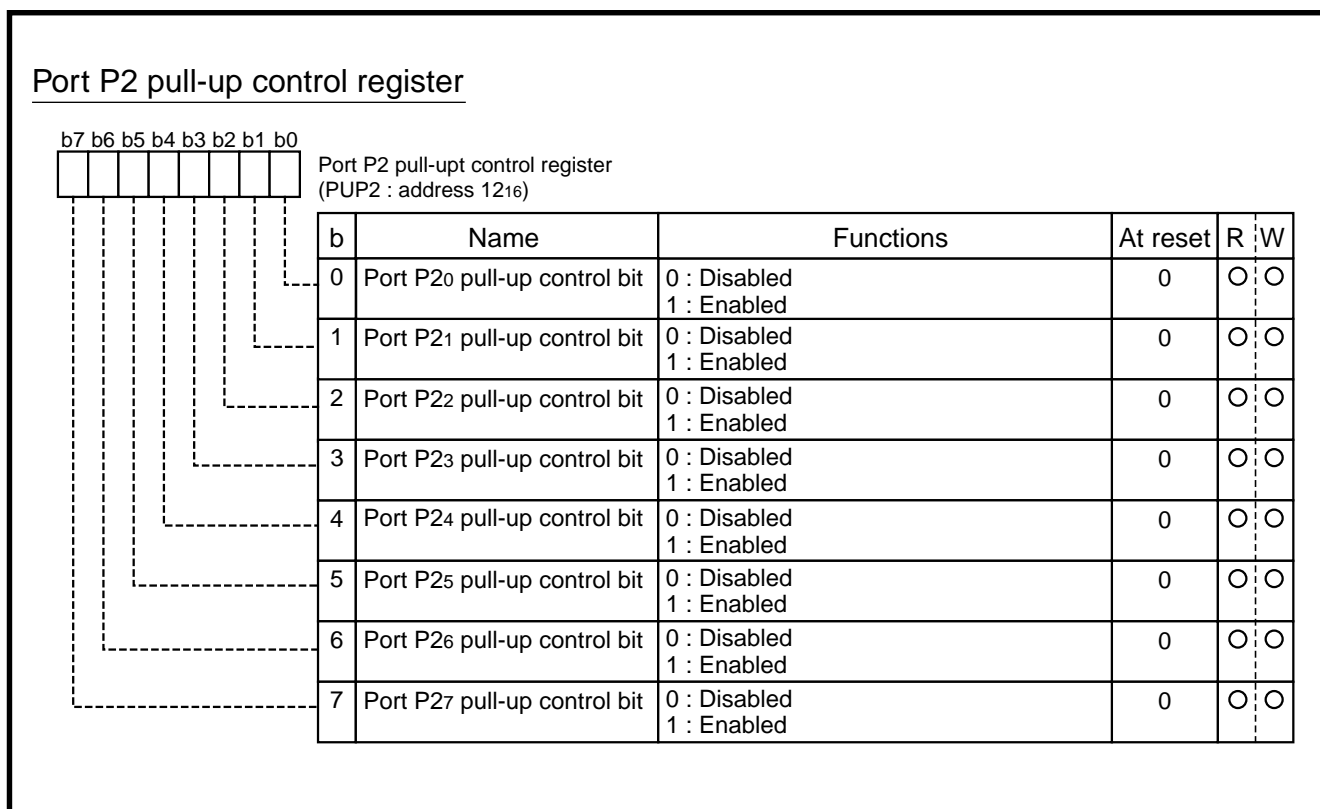


Fig. 3.5.15 Structure of Port P2 pull-up control register

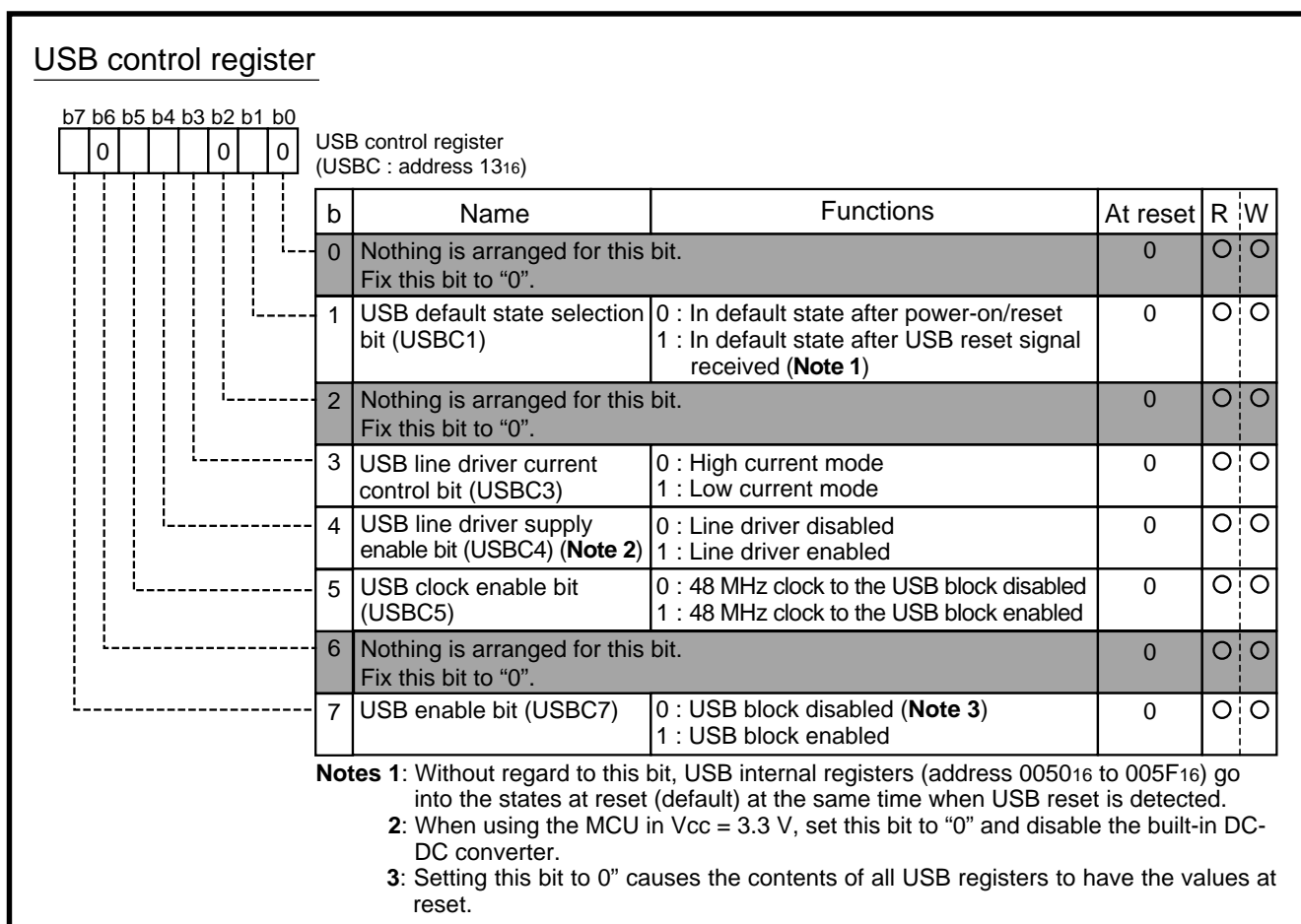


Fig. 3.5.16 Structure of USB control register

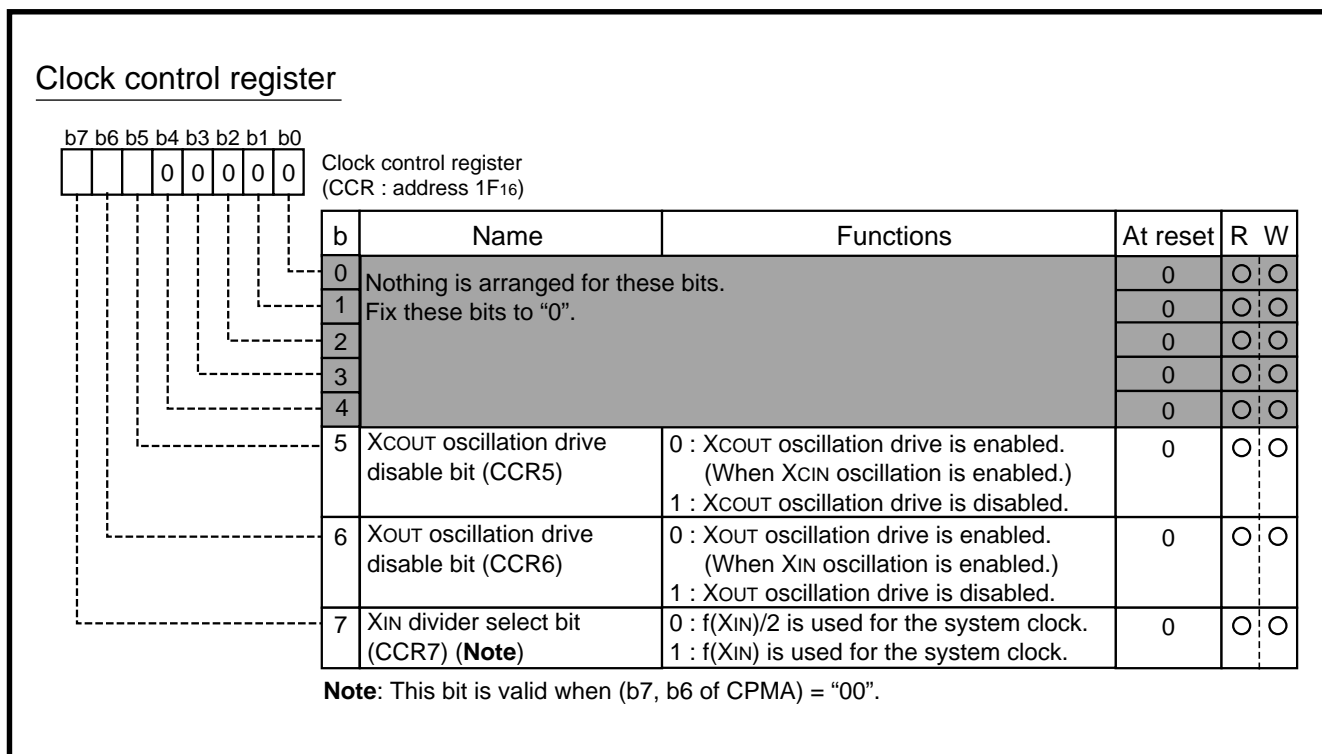


Fig. 3.5.17 Structure of Clock control register

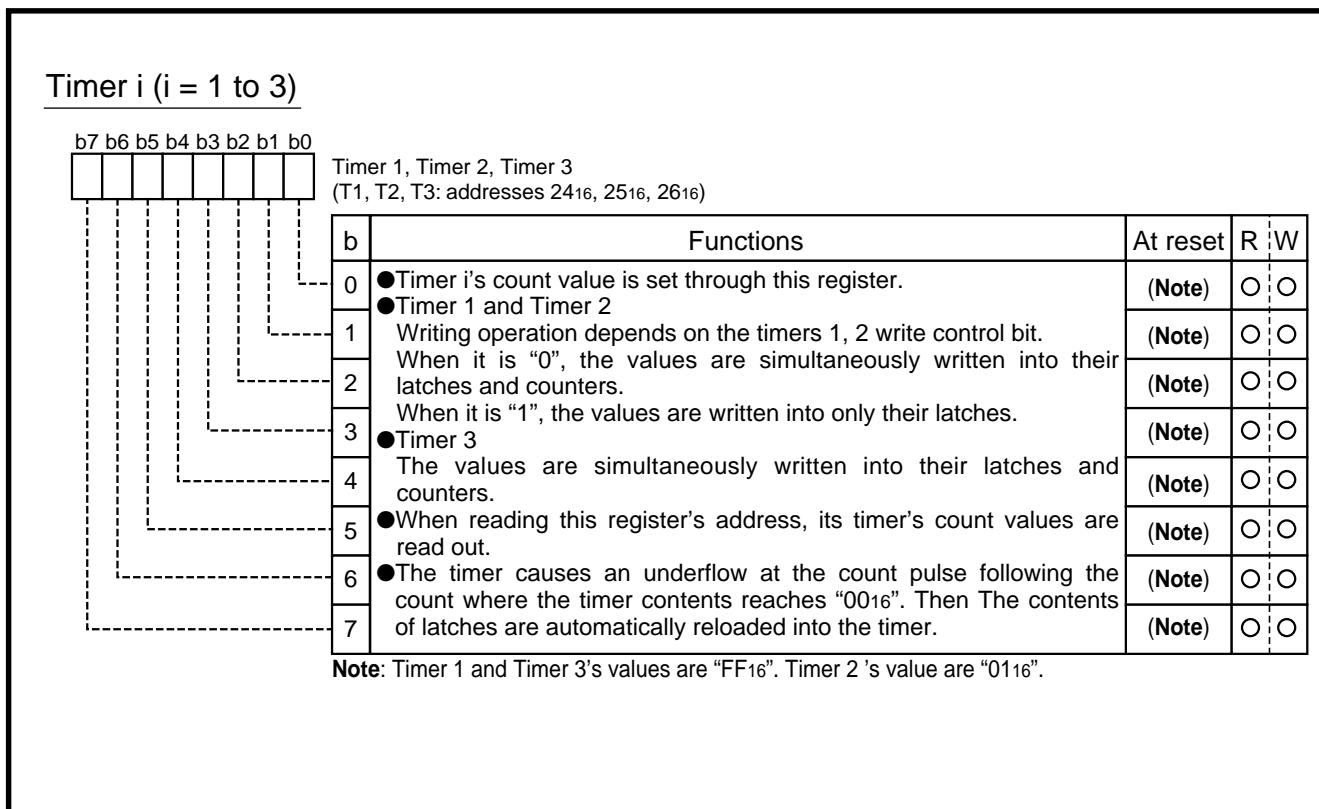


Fig. 3.5.18 Structure of Timer i

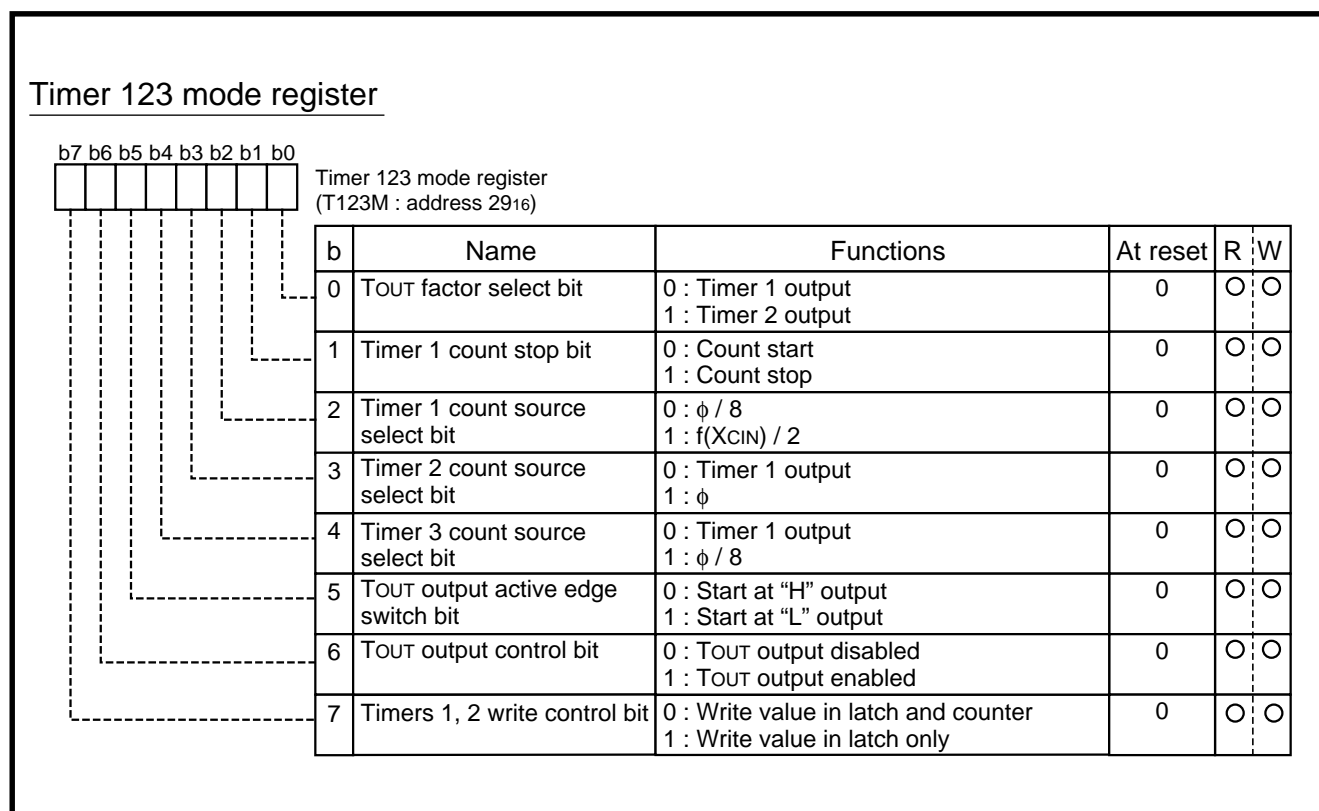


Fig. 3.5.19 Structure of Timer 123 mode register



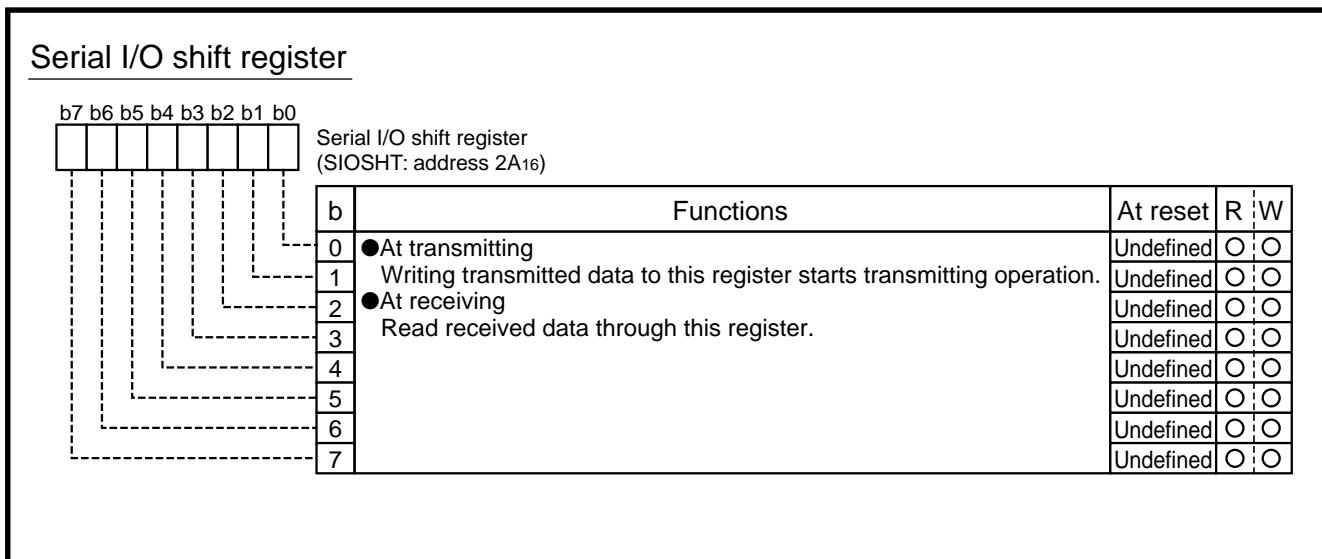


Fig. 3.5.20 Structure of Serial I/O shift register

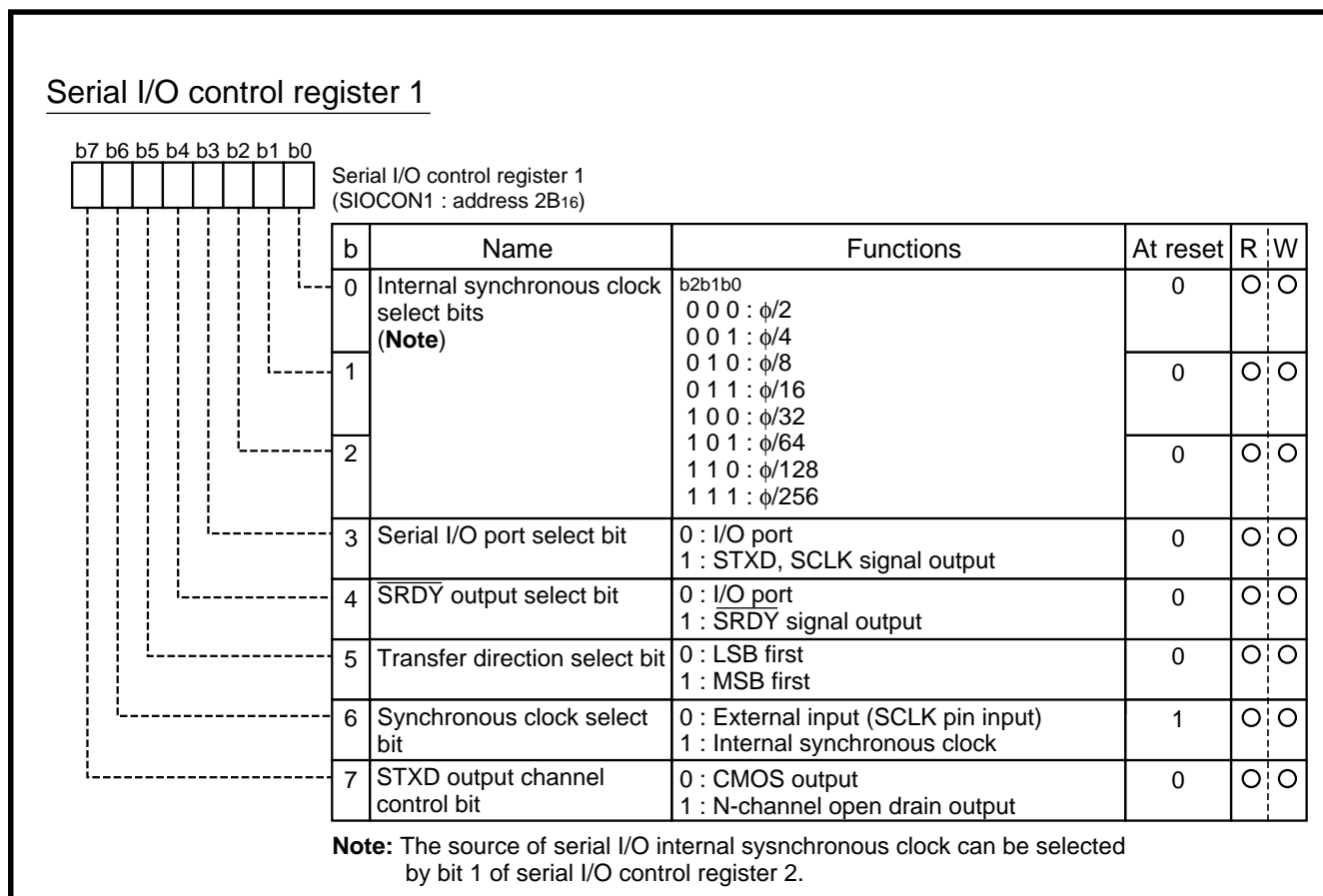


Fig. 3.5.21 Structure of Serial I/O control register 1

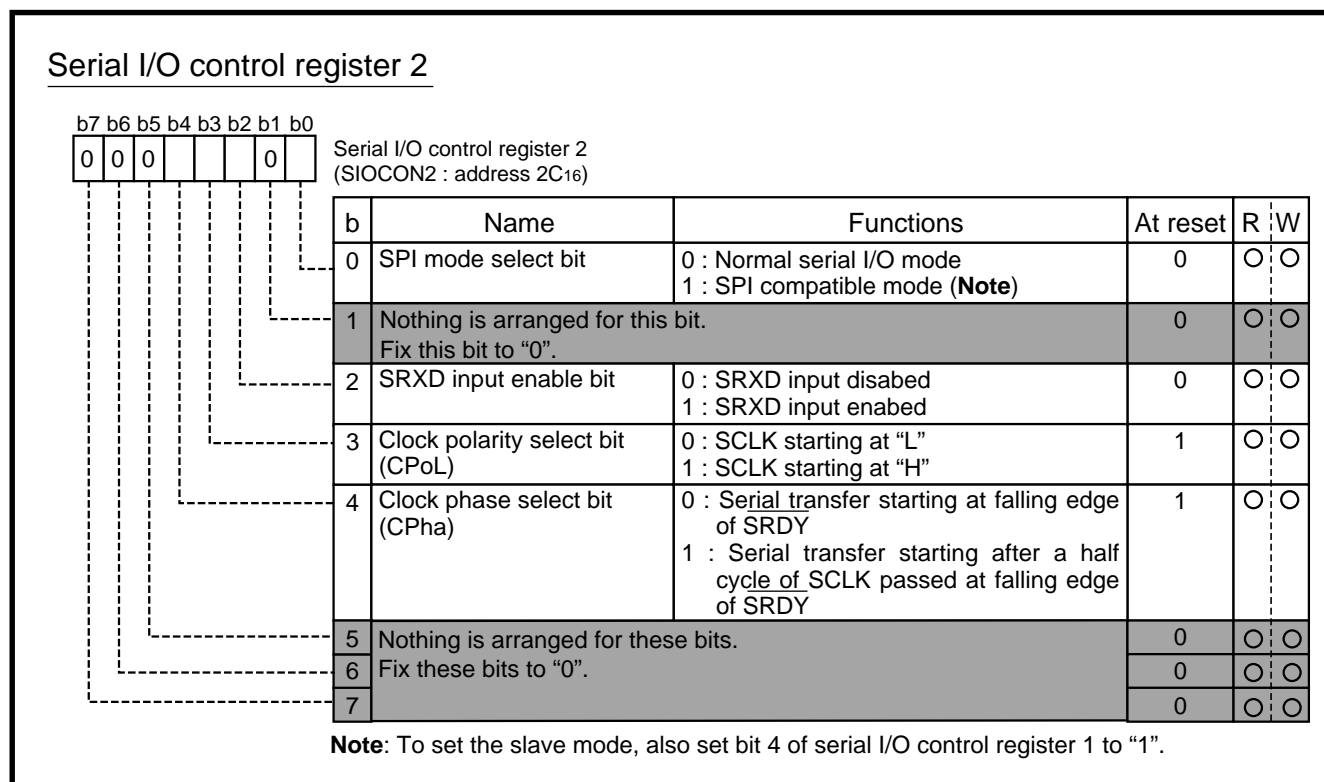


Fig. 3.5.22 Structure of Serial I/O control register 2

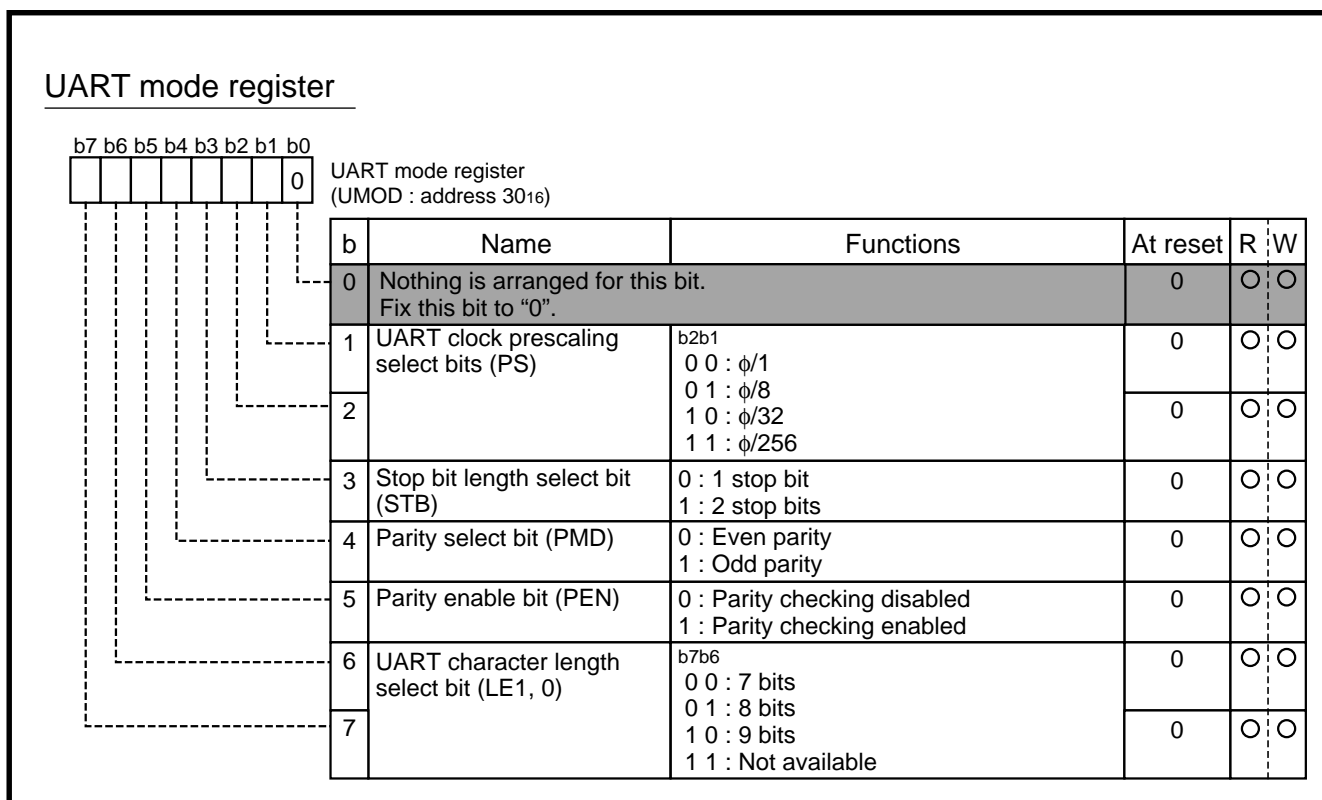


Fig. 3.5.23 Structure of Timer UART mode register

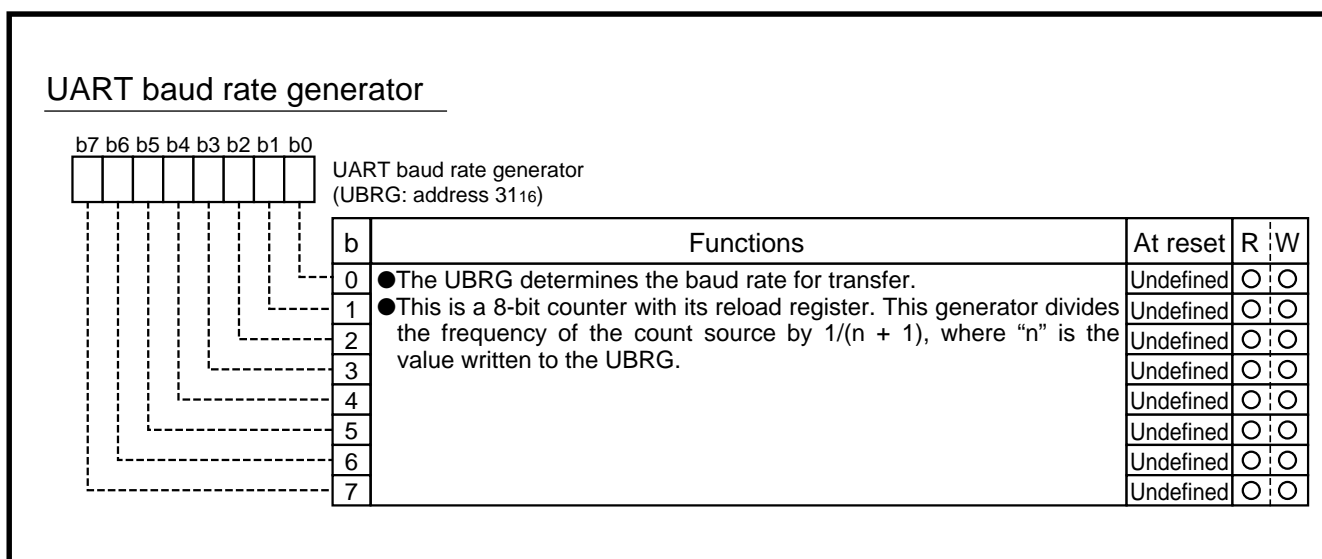


Fig. 3.5.24 Structure of UART baud rate generator

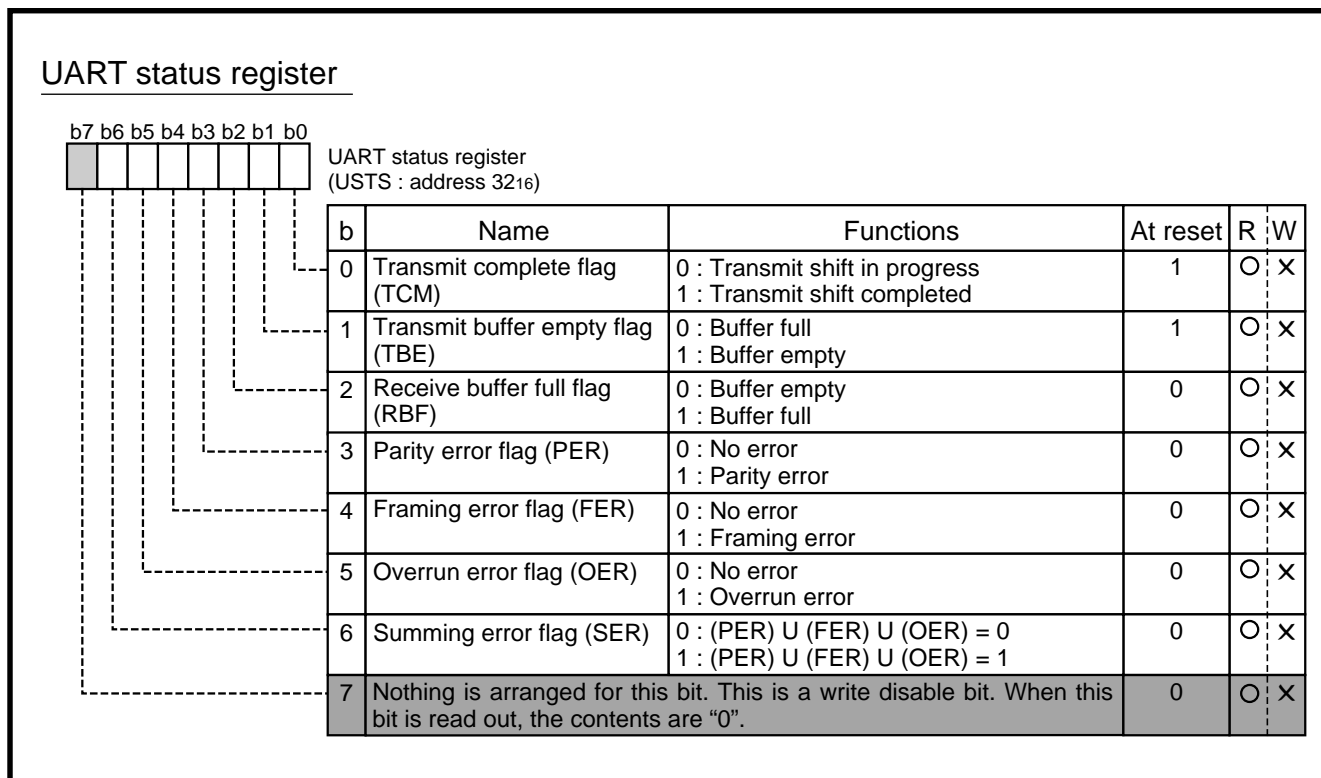


Fig. 3.5.25 Structure of UART status register

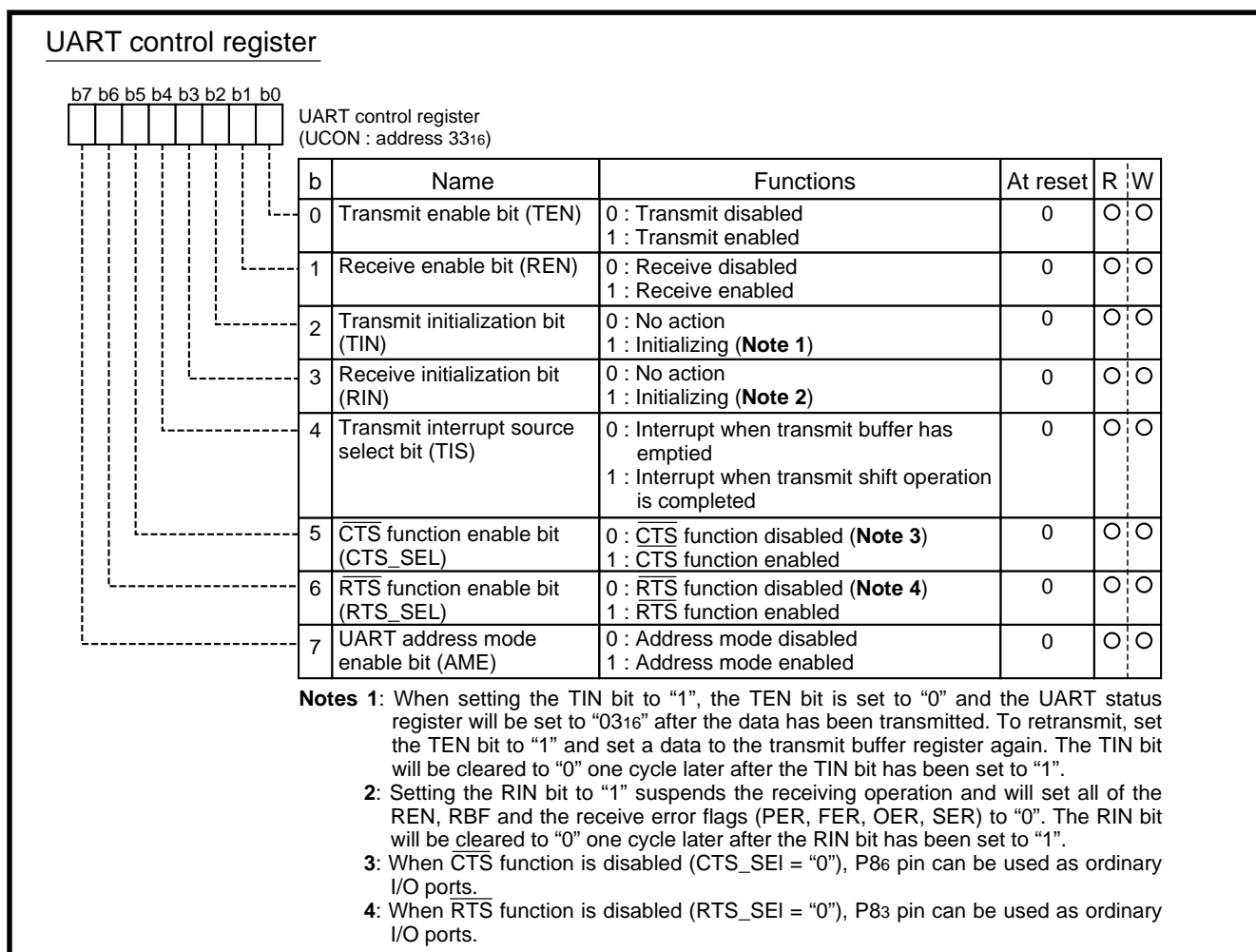


Fig. 3.5.26 Structure of UART control register

### UART transmit/receive buffer registers 1, 2

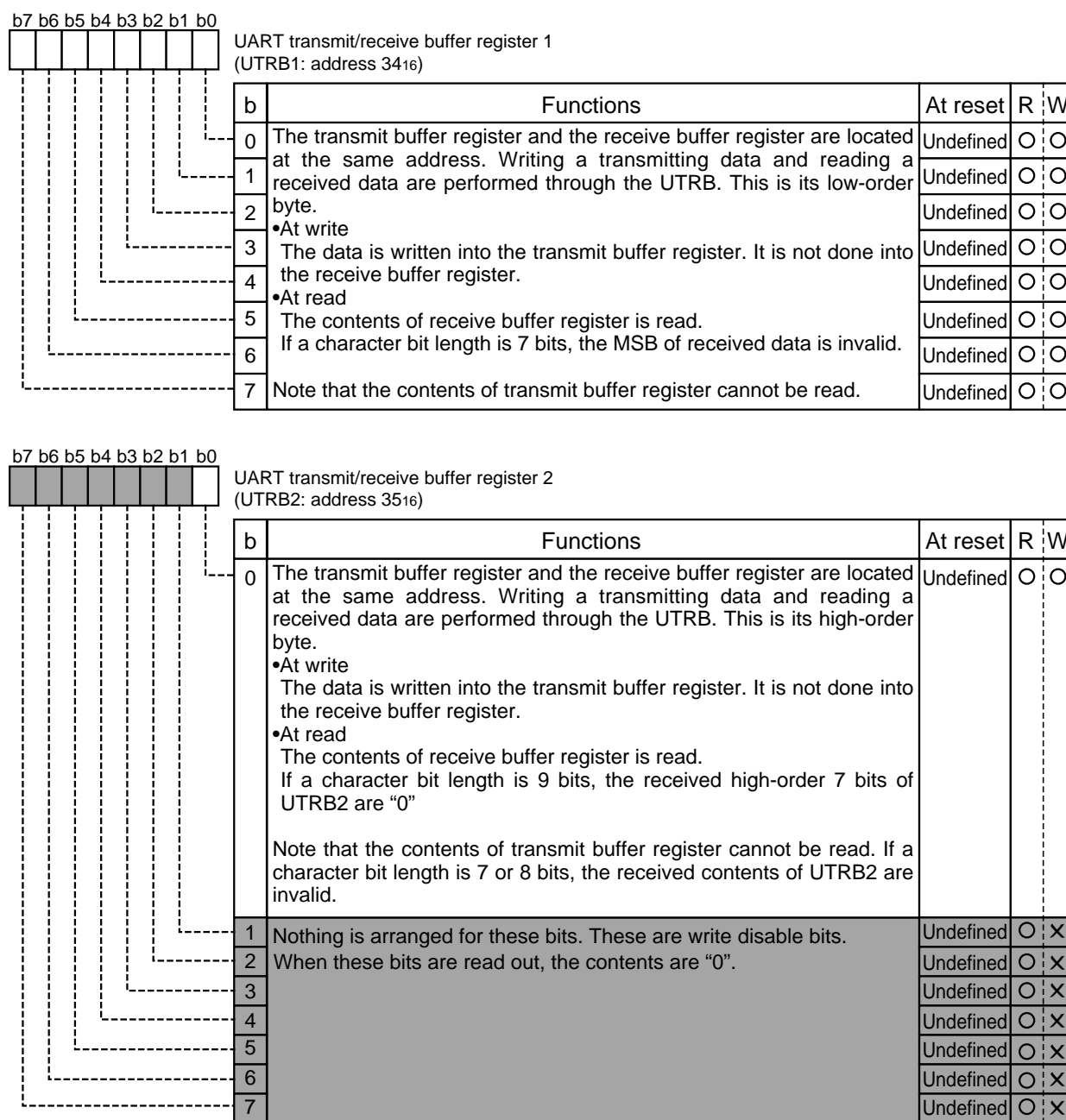


Fig. 3.5.27 Structure of UART transmit/receive buffer registers 1, 2

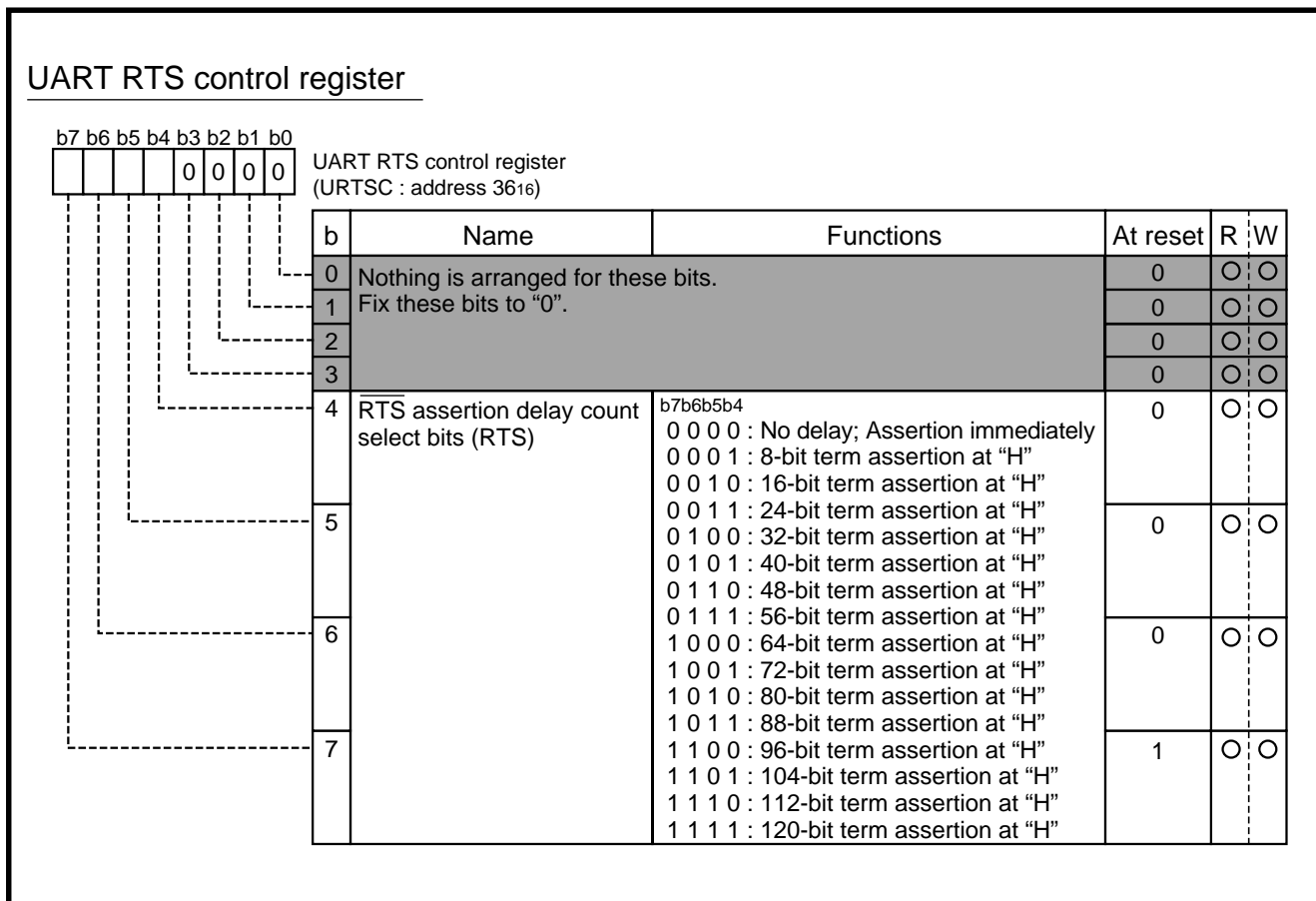


Fig. 3.5.28 Structure of UART RTS control register

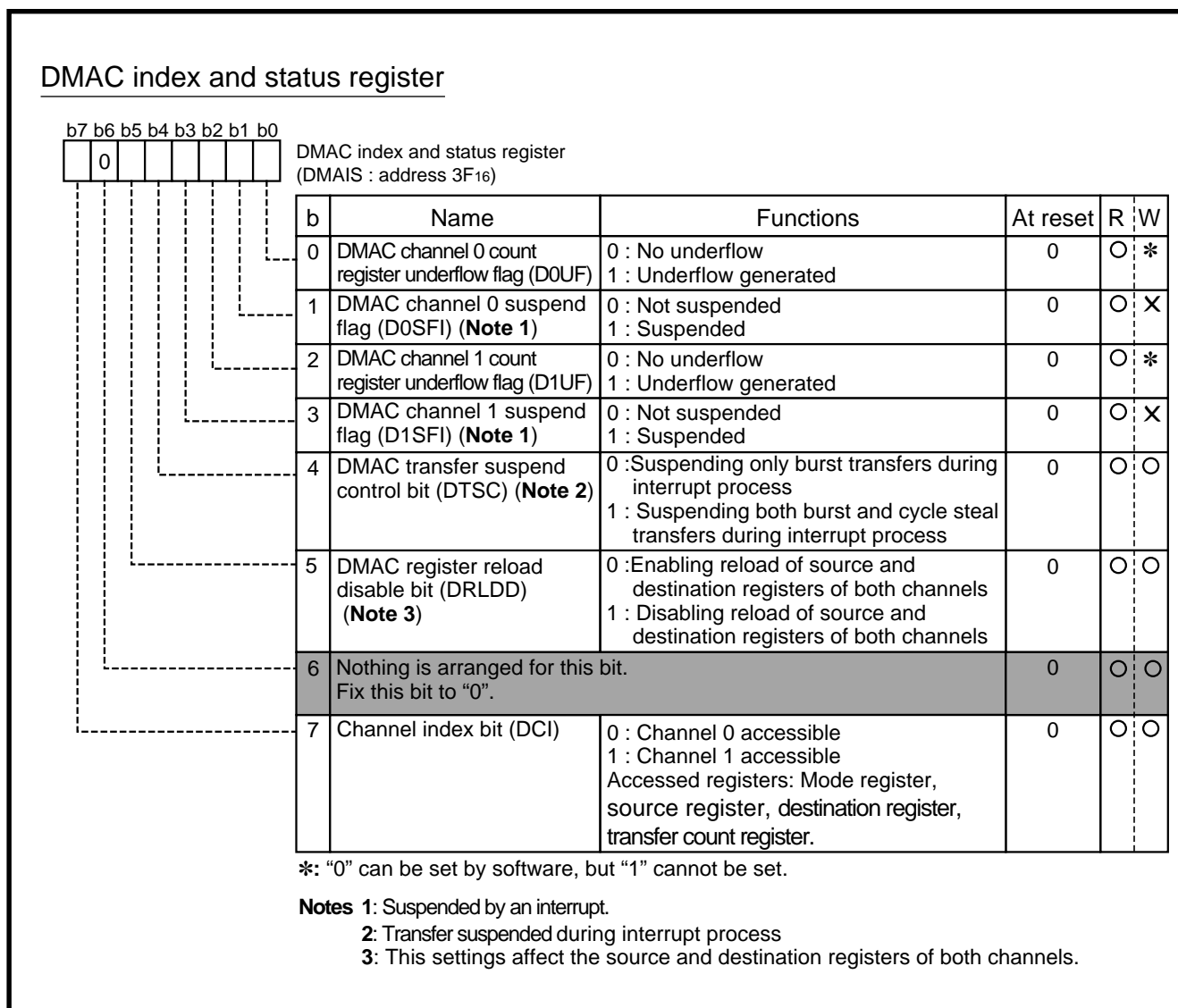


Fig. 3.5.29 Structure of DMAC index and status register

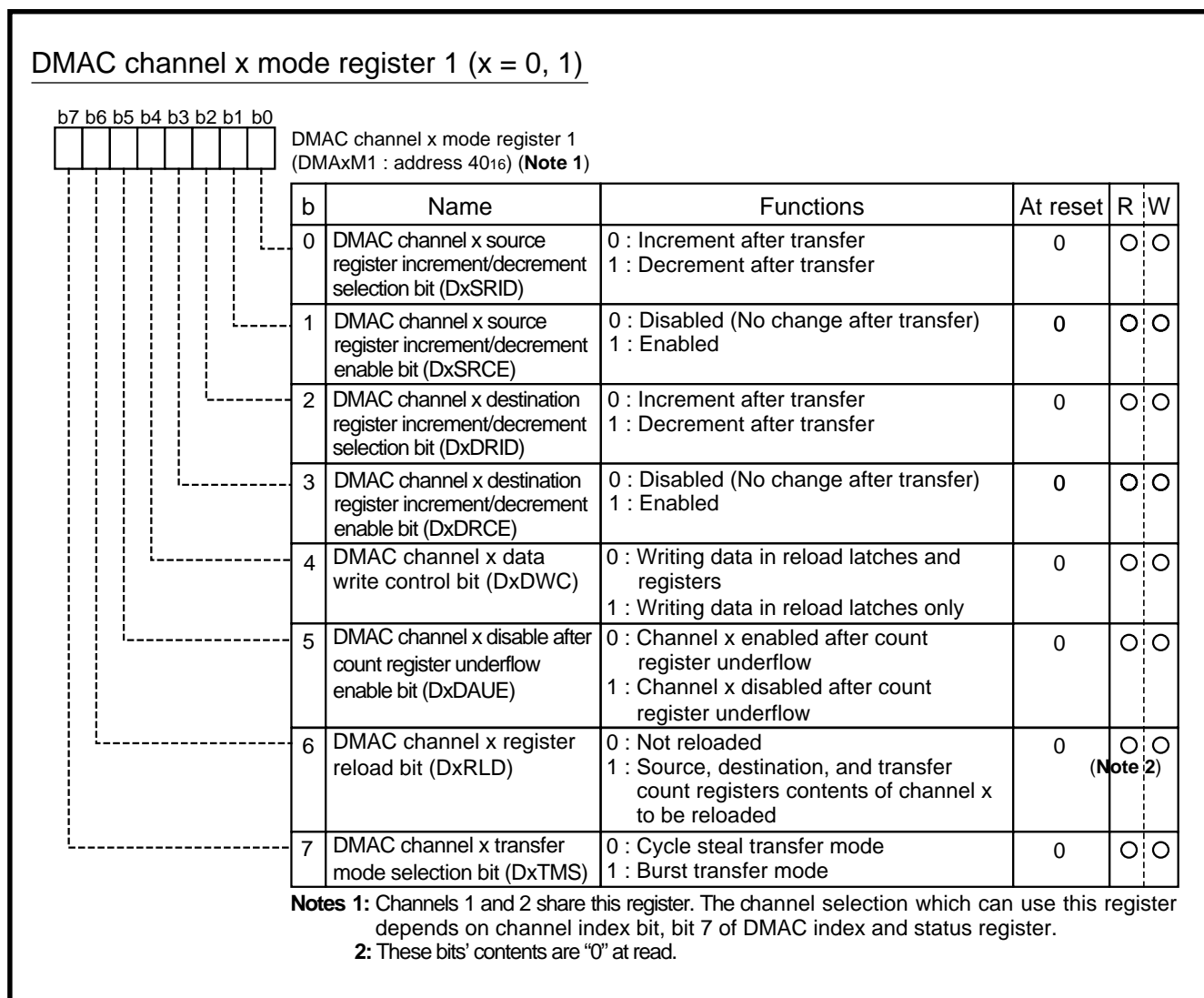


Fig. 3.5.30 Structure of DMAC channel x mode register 1 (x = 0, 1)



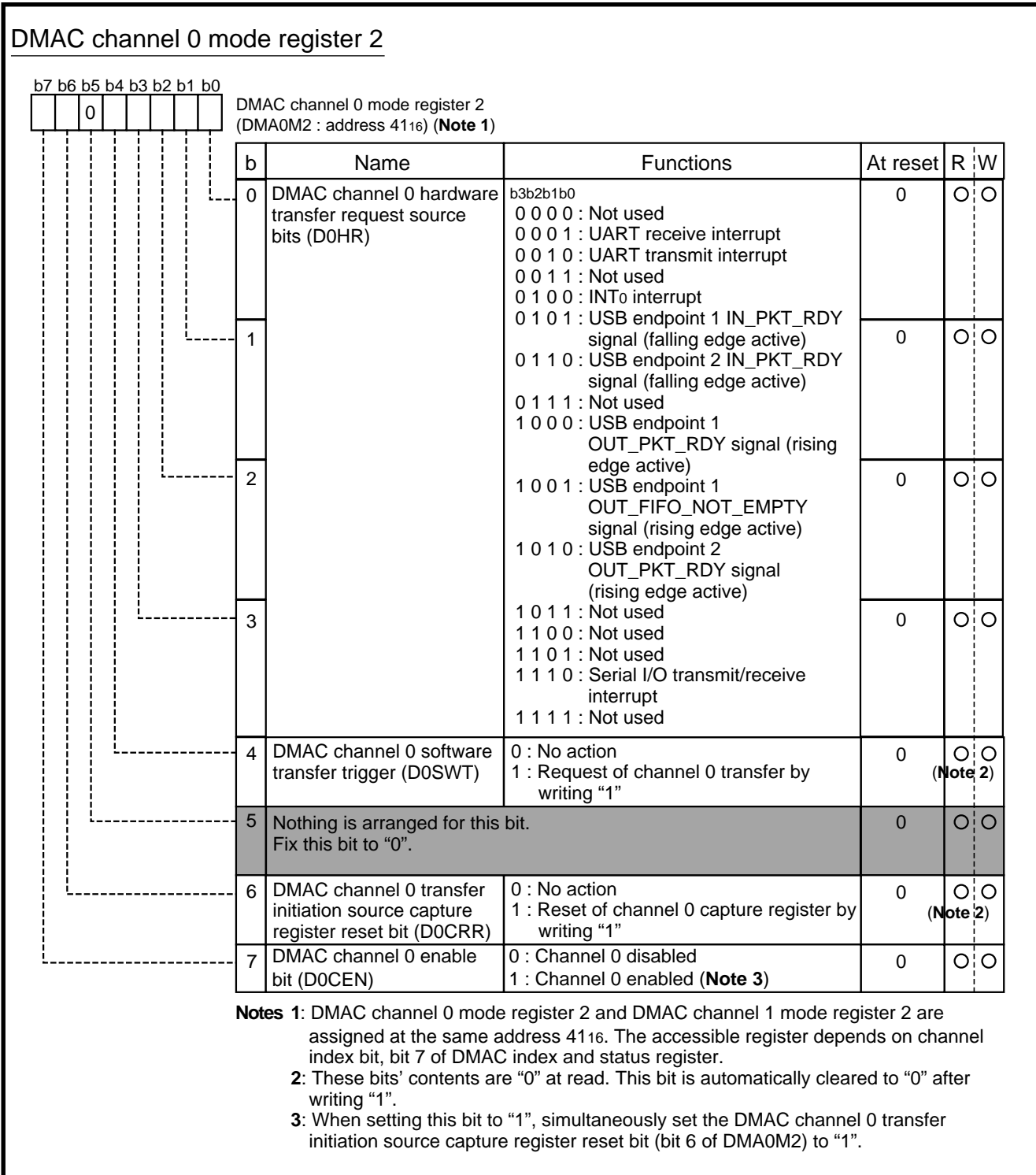


Fig. 3.5.31 Structure of DMAC channel 0 mode register 2

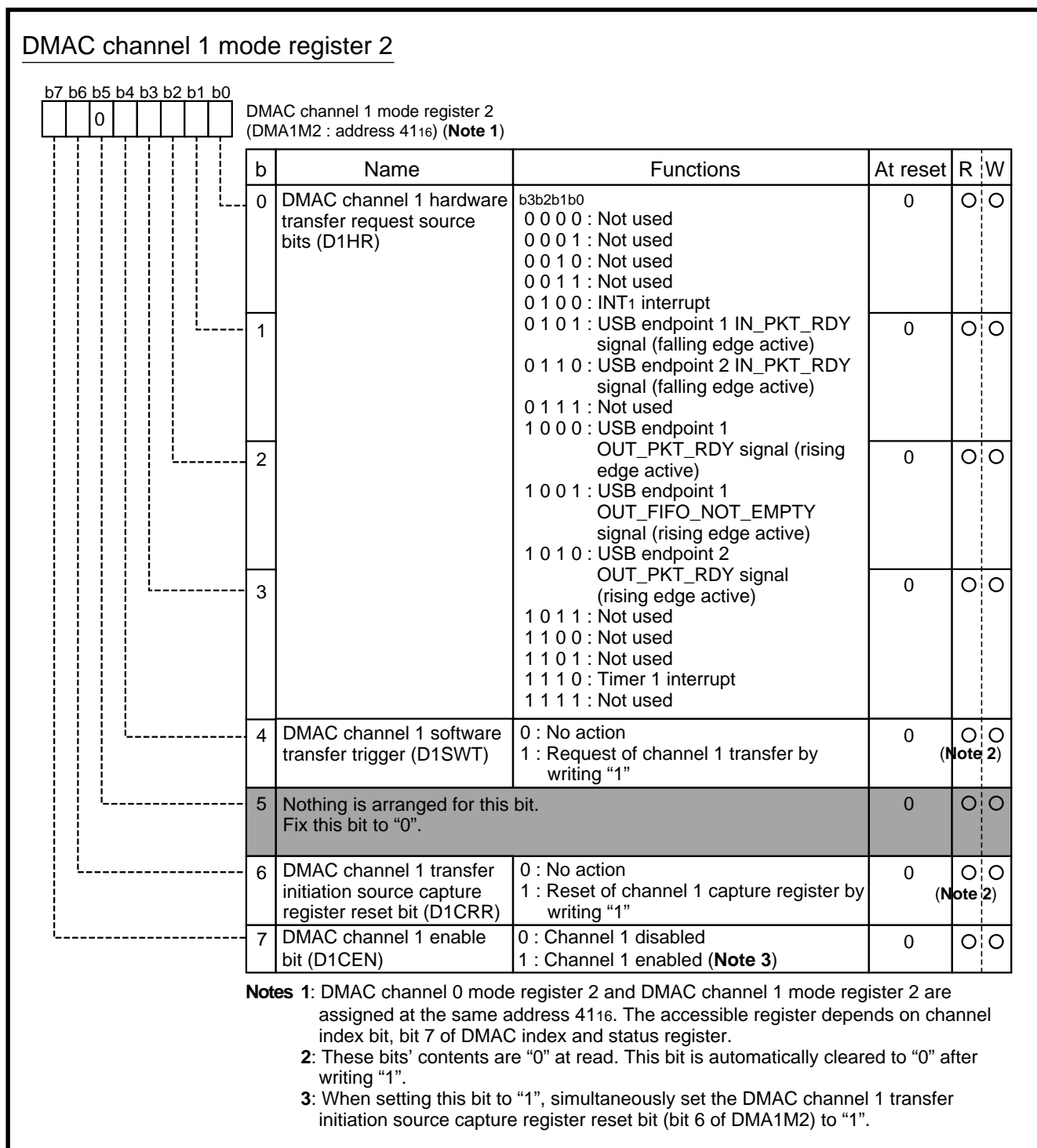


Fig. 3.5.32 Structure of DMAC channel 1 mode register 2

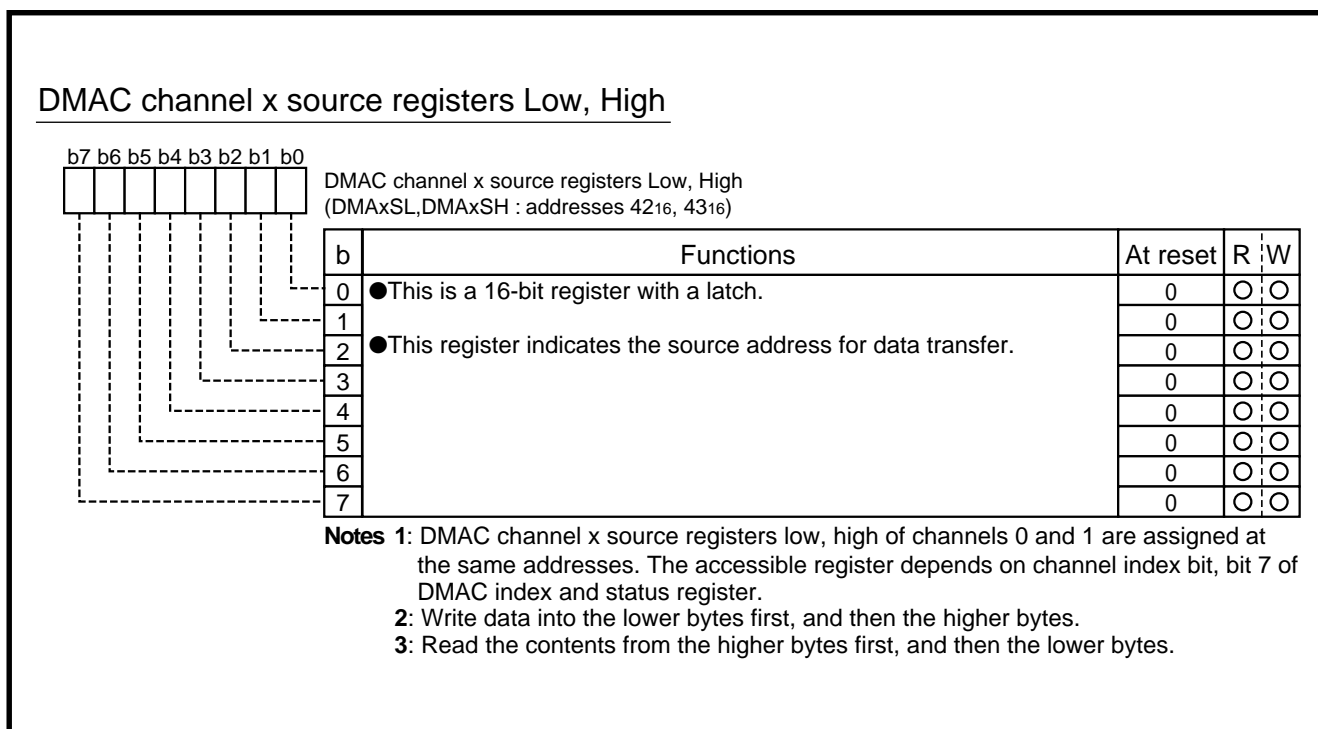


Fig. 3.5.33 Structure of DMAC channel x source registers Low, High

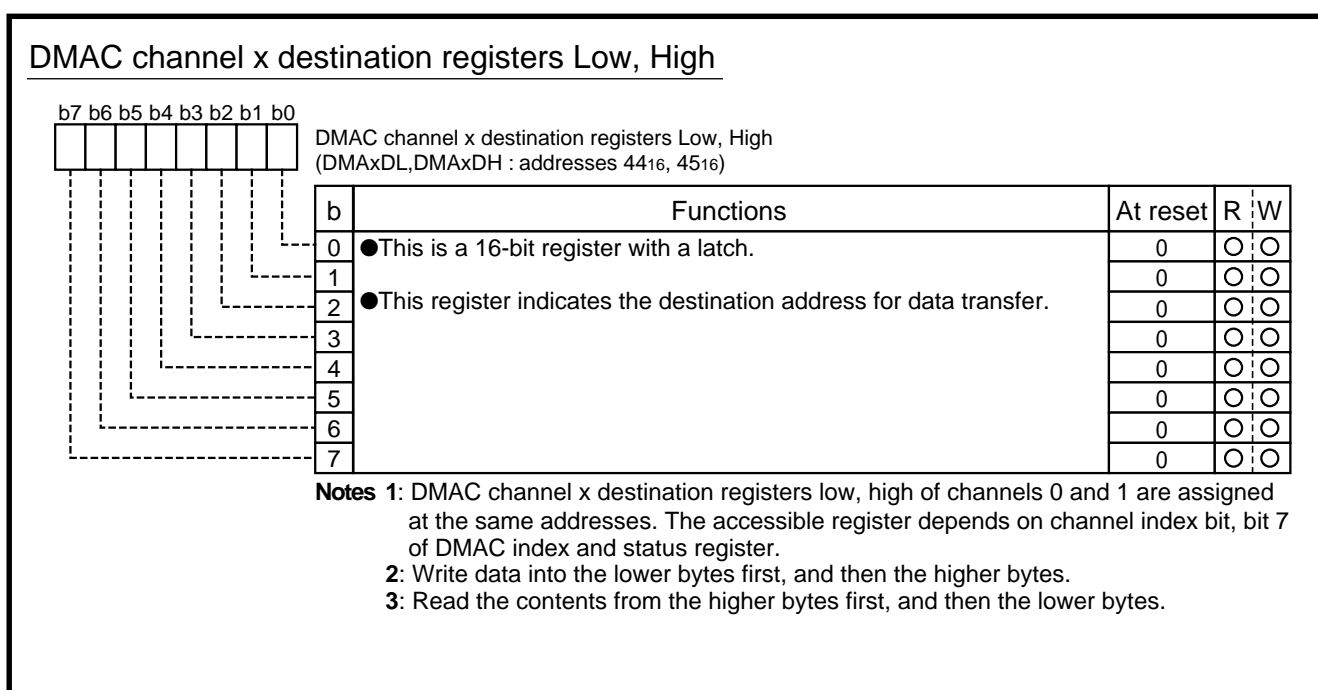
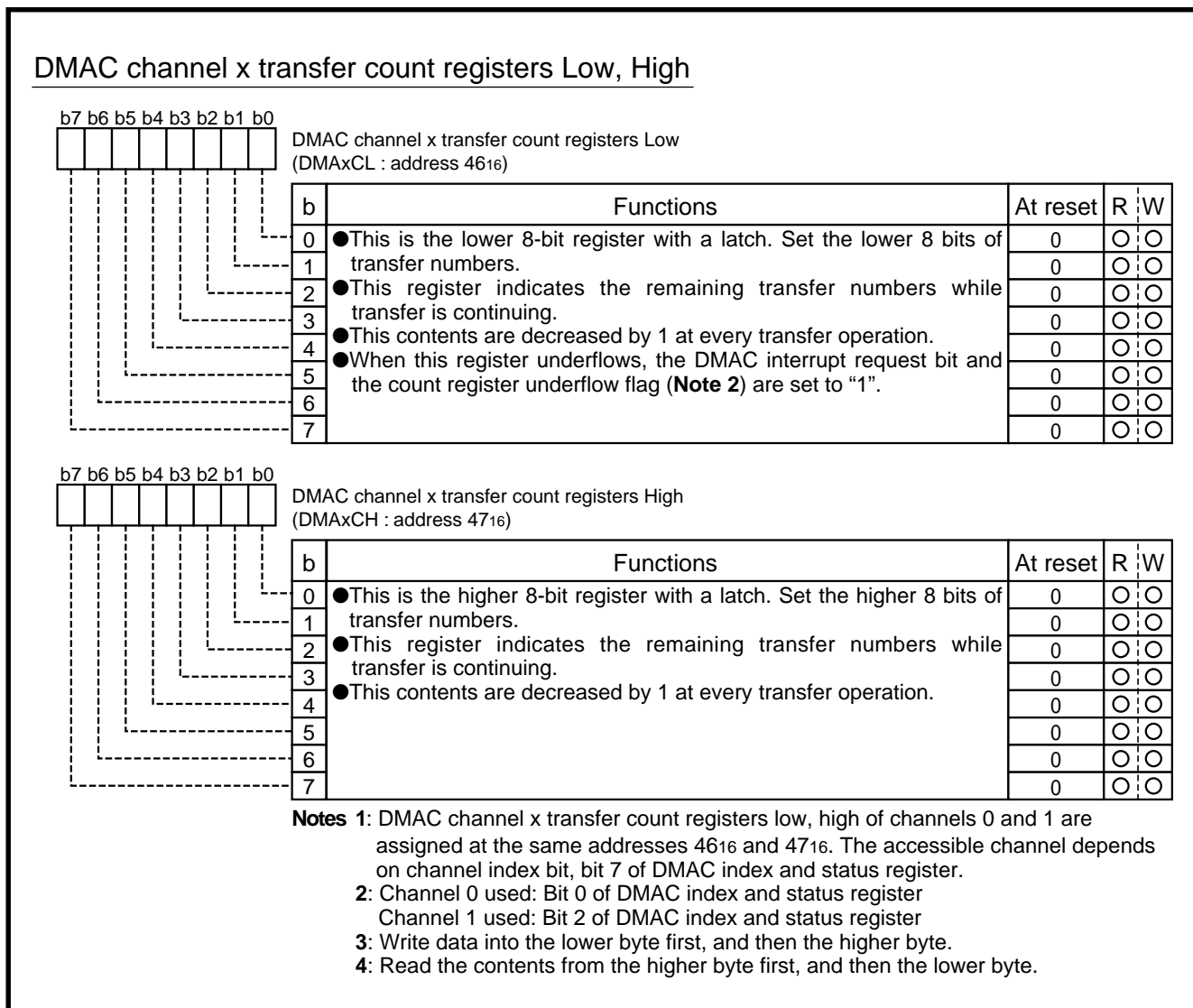


Fig. 3.5.34 Structure of DMAC channel x destination registers Low, High



**Fig. 3.5.35 Structure of DMAC channel x transfer count registers Low, High (x = 0, 1)**

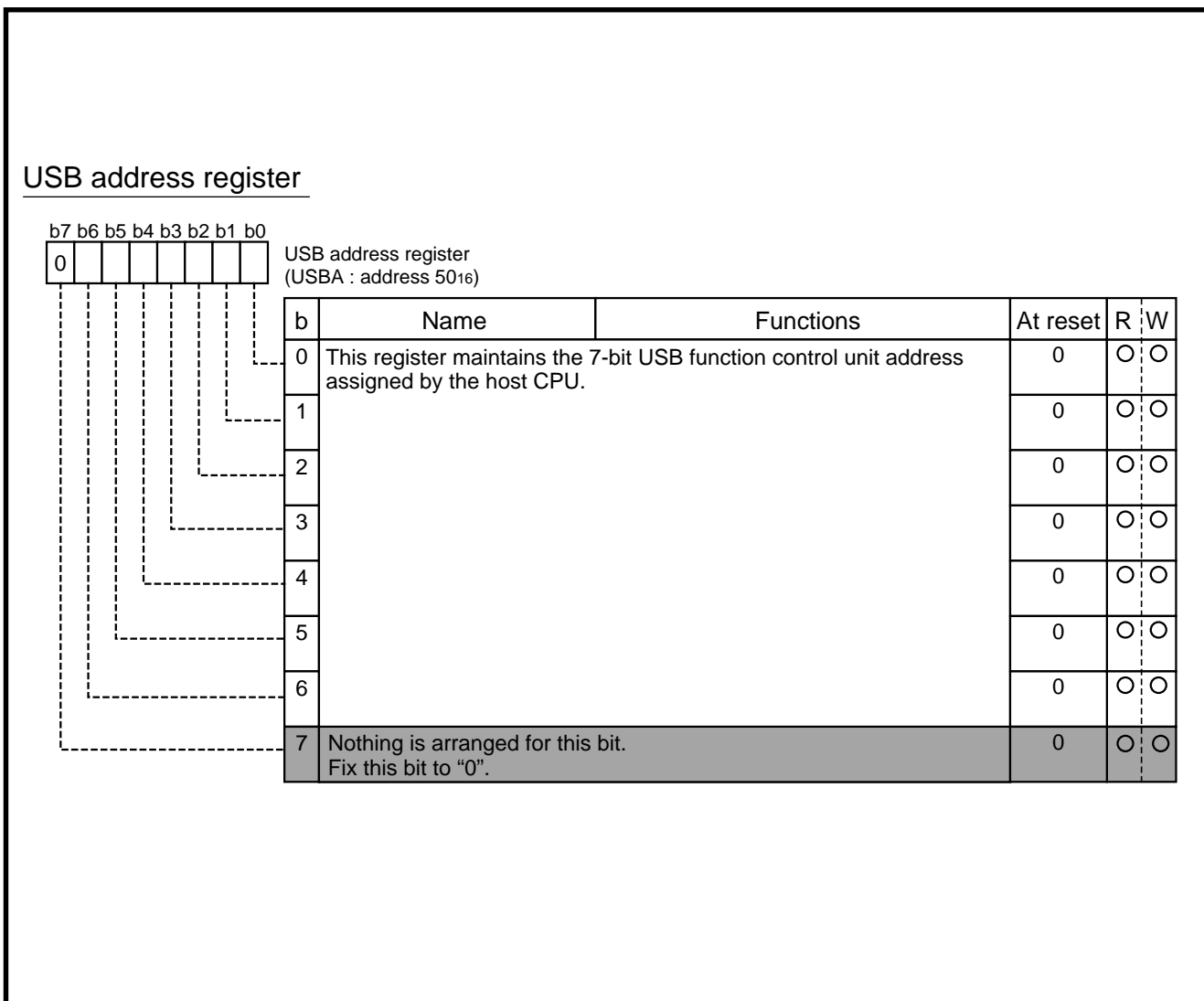


Fig. 3.5.36 Structure of USB address register

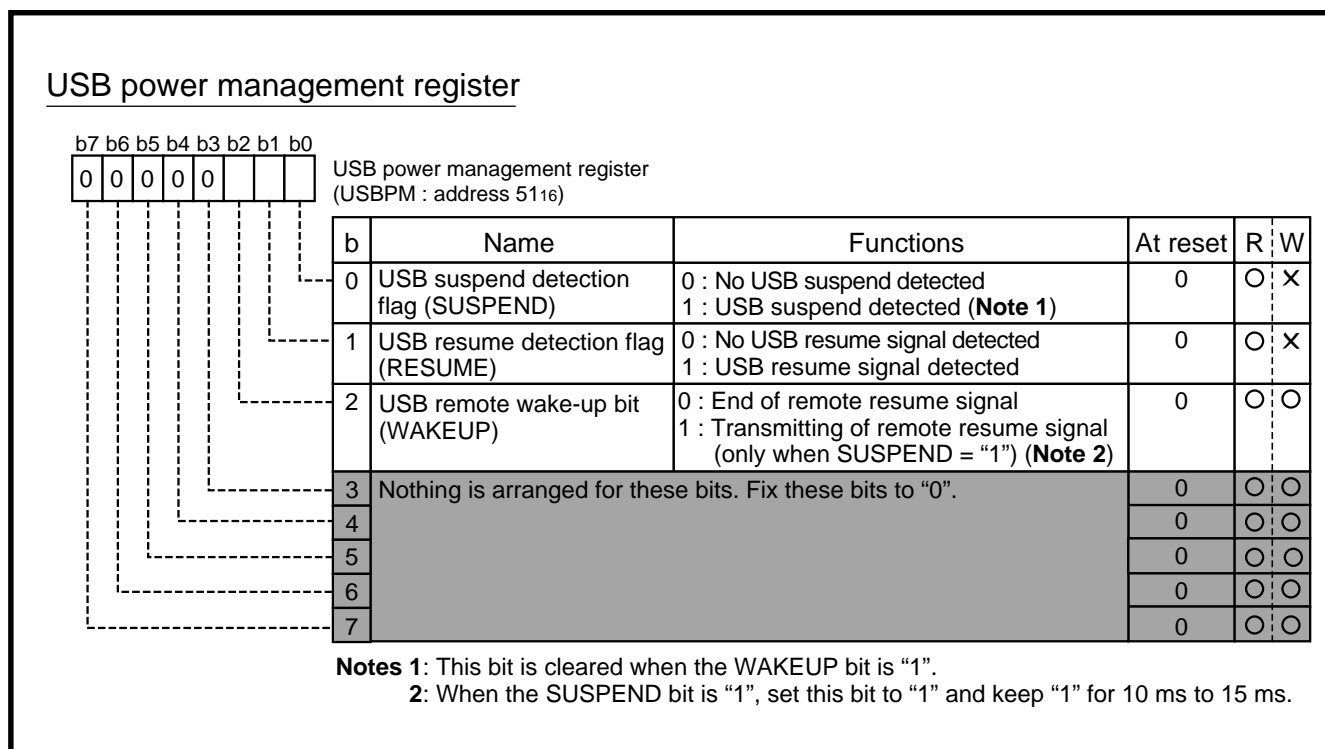


Fig. 3.5.37 Structure of USB power management register

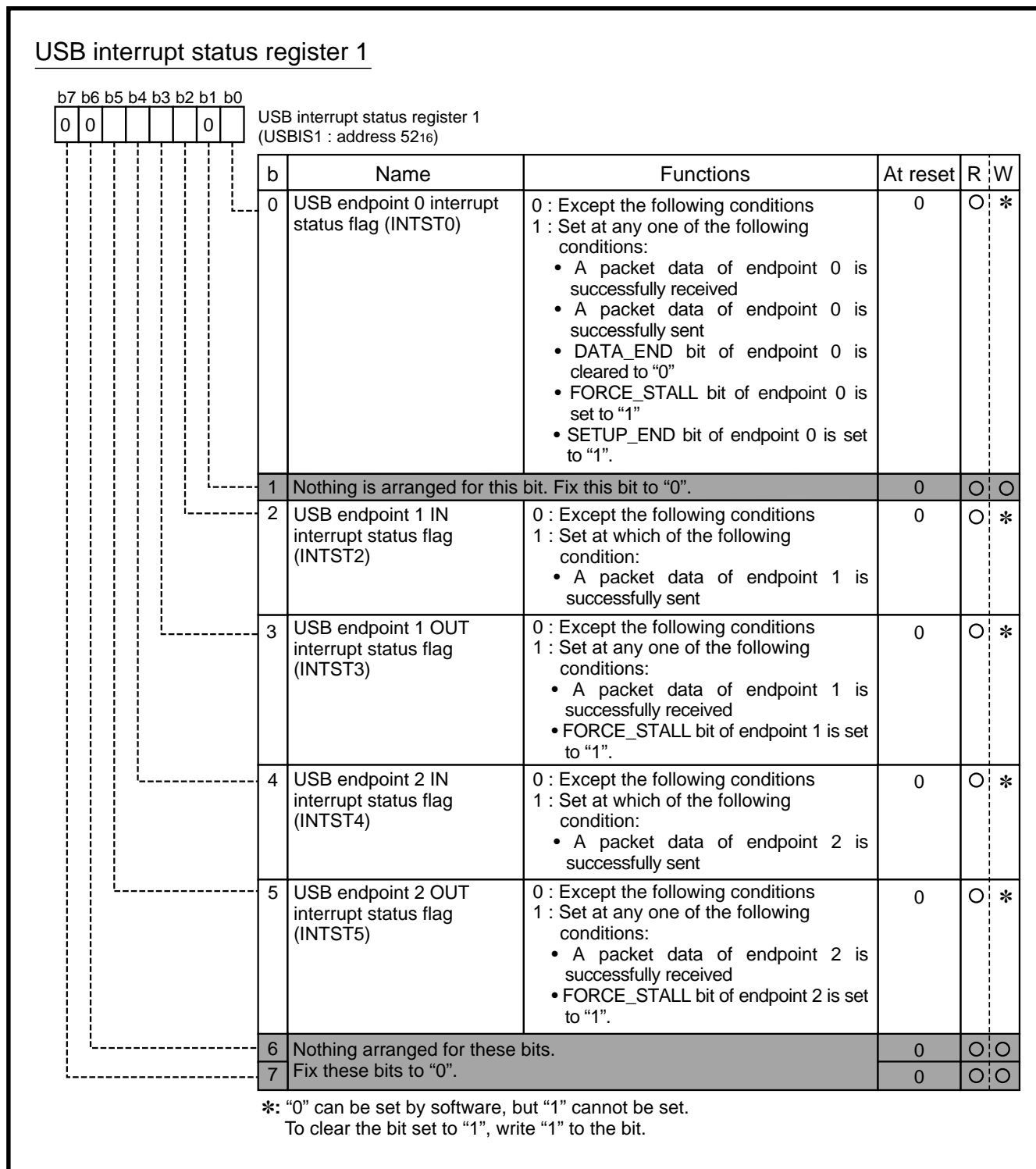


Fig. 3.5.38 Structure of USB interrupt status register 1

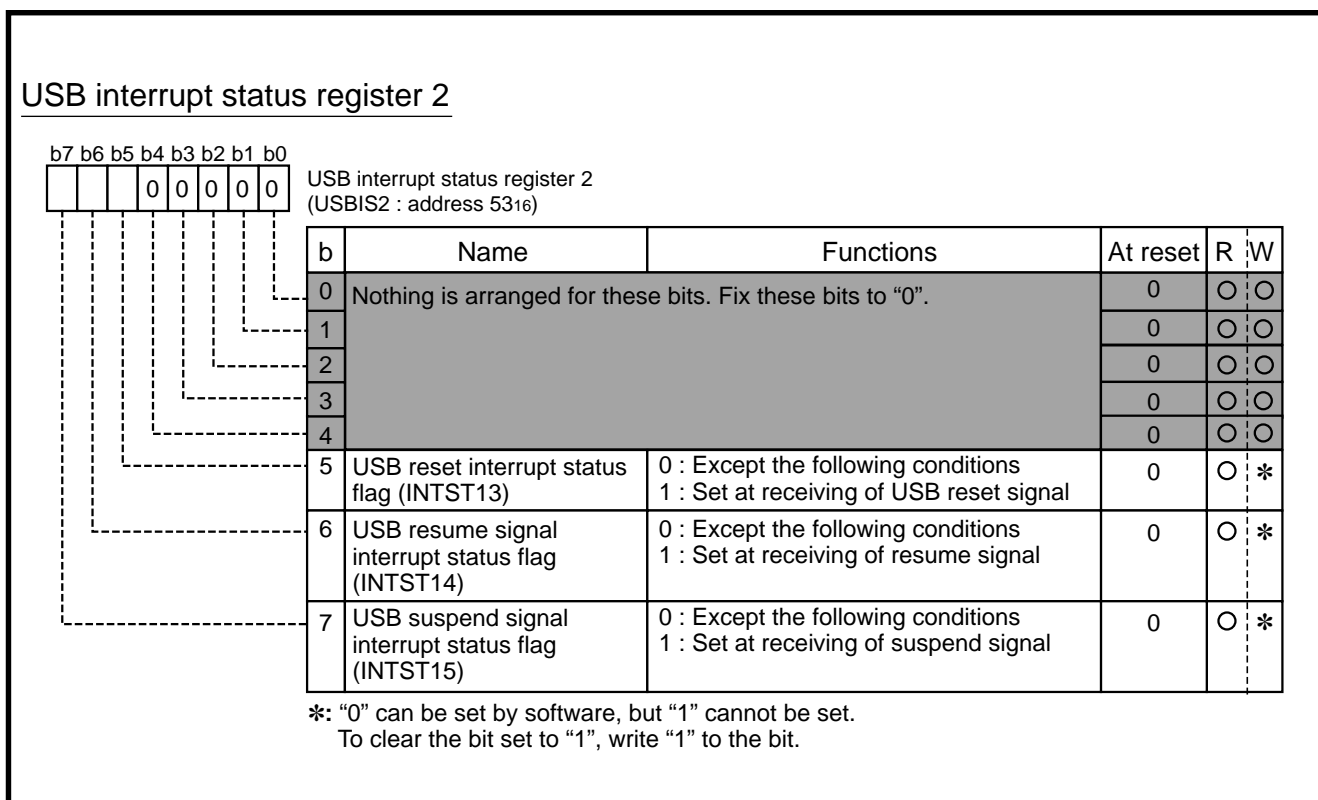


Fig. 3.5.39 Structure of USB interrupt status register 2



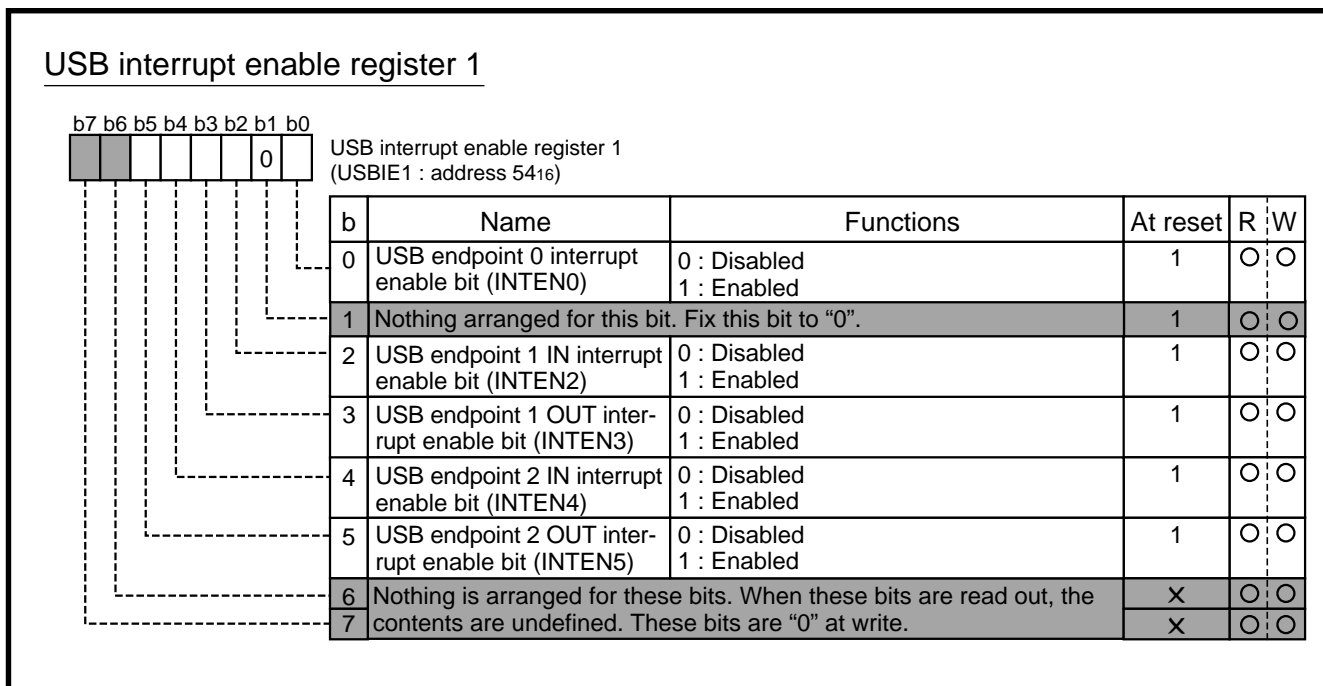


Fig. 3.5.40 Structure of USB interrupt enable register 1

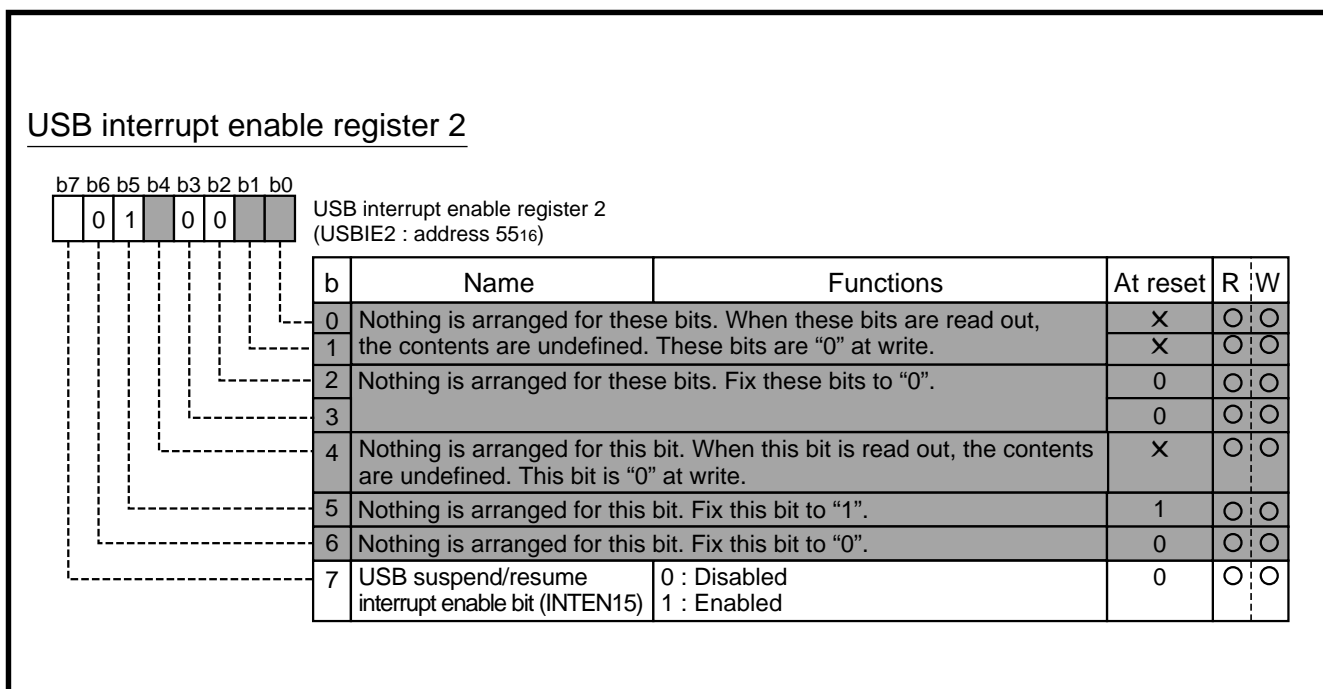


Fig. 3.5.41 Structure of USB interrupt enable register 2

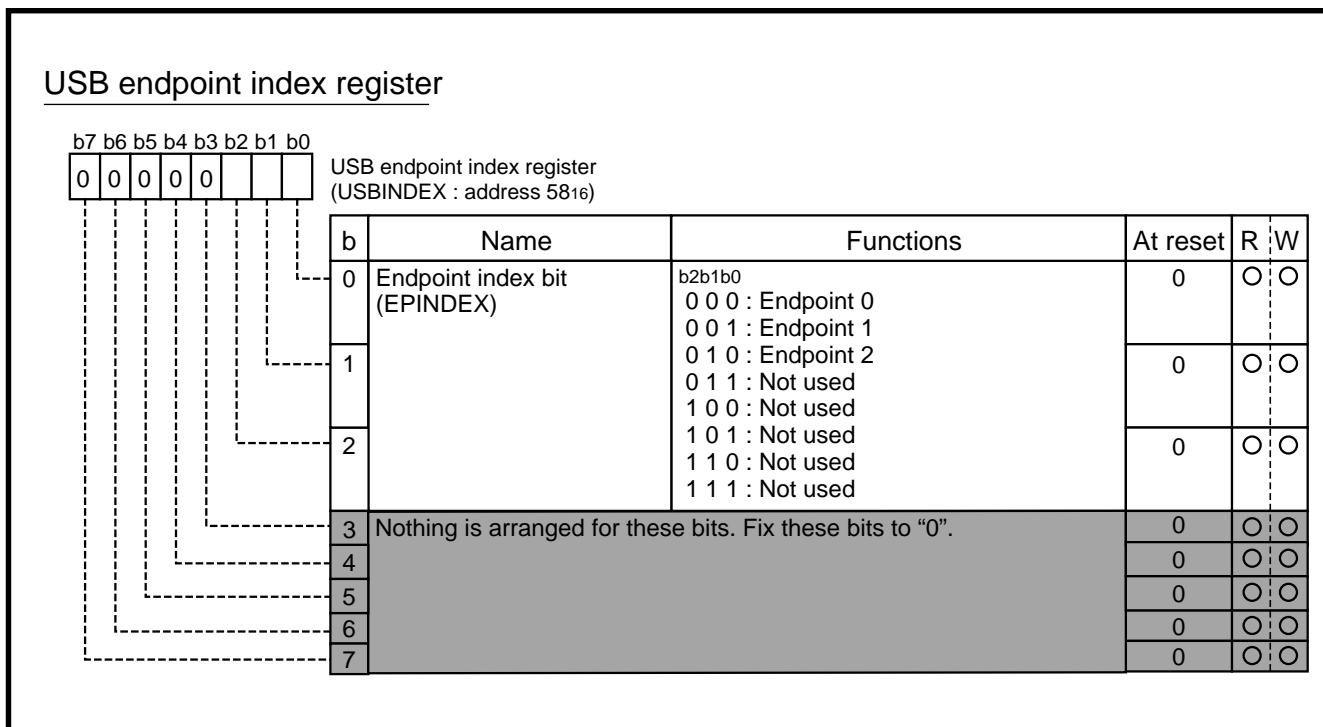


Fig. 3.5.42 Structure of USB endpoint index register

## USB endpoint 0 IN control register

b	Name	Functions	At reset	R	W
0	OUT_PKT_RDY flag (IN0CSR0)	0 : Except the following condition (Cleared to "0" by writing "1" into SERVICED_OUT_PKT_RDY bit) 1 : End of a data packet reception	0	○	*1
1	IN_PKT_RDY bit (IN0CSR1)	0 : End of a data packet transmission 1 : Write "1" at completion of writing a data packet into IN FIFO.	0	○	*2
2	SEND_STALL bit (IN0CSR2)	0 : Except the following condition 1 : Transmitting STALL handshake signal	0	○	○
3	DATA_END bit (IN0CSR3)	0 : Except the following condition (Cleared to "0" after completion of status phase) 1 : Write "1" at completion of writing or reading the last data packet to/from FIFO.	0	○	*2
4	FORCE_STALL flag (IN0CSR4)	0 : Except the following condition 1 : Protocol error detected	0	○	*1
5	SETUP_END flag (IN0CSR5)	0 : Except the following condition (Cleared to "0" by writing "1" into SERVICED_SETUP_END bit) 1 : Control transfer ends before the specific length of data is transferred during the data phase.	0	○	*1
6	SERVICED_OUT_PKT_RDY bit (IN0CSR6)	0 : Except the following condition 1 : Writing "1" to this bit clears OUT_PKT_RDY flag to "0".	0	○	○
7	SERVICED_SETUP_END bit (IN0CSR7)	0 : Except the following condition 1 : Writing "1" to this bit clears SETUP_END flag to "0".	0	○	○

## USB endpoint 1, 2 IN control register

b	Name	Functions	At reset	R	W
0	INT_PKT_RDY bit (INXCSR0)	0 : End of a data packet transmission 1 : Write "1" at completion of writing a data packet into IN FIFO. ( <b>Note 2</b> )	0	○	*2
1	Nothing is arranged for this bit. Fix this bit to "0".		0	○	○
2	SEND_STALL bit (INXCSR2)	0 : Except the following condition 1 : Transmitting STALL handshake signal	0	○	○
3	TOGGLE_INIT bit (INXCSR3)	0 : Except the following condition 1 : Initializing the data toggle sequence bit	0	○	○
4	INTPT bit (INXCSR4)	0 : Except the following condition 1 : Initializing to endpoint used for interrupt transfer, rate feedback	0	○	○
5	TX_NOT_EPT flag (INXCSR5)	0 : Empty in IN FIFO 1 : Full in IN FIFO	0	○	X
6	FLUSH bit (INXCSR6)	0 : Except the following condition 1 : Flush FIFO	0	○	*1
7	AUTO_SET bit (INXCSR7)	0 : AUTO_SET disabled 1 : AUTO_SET enabled *3	0	○	○

\* 1: "1" can be set by software, but "0" cannot be set.

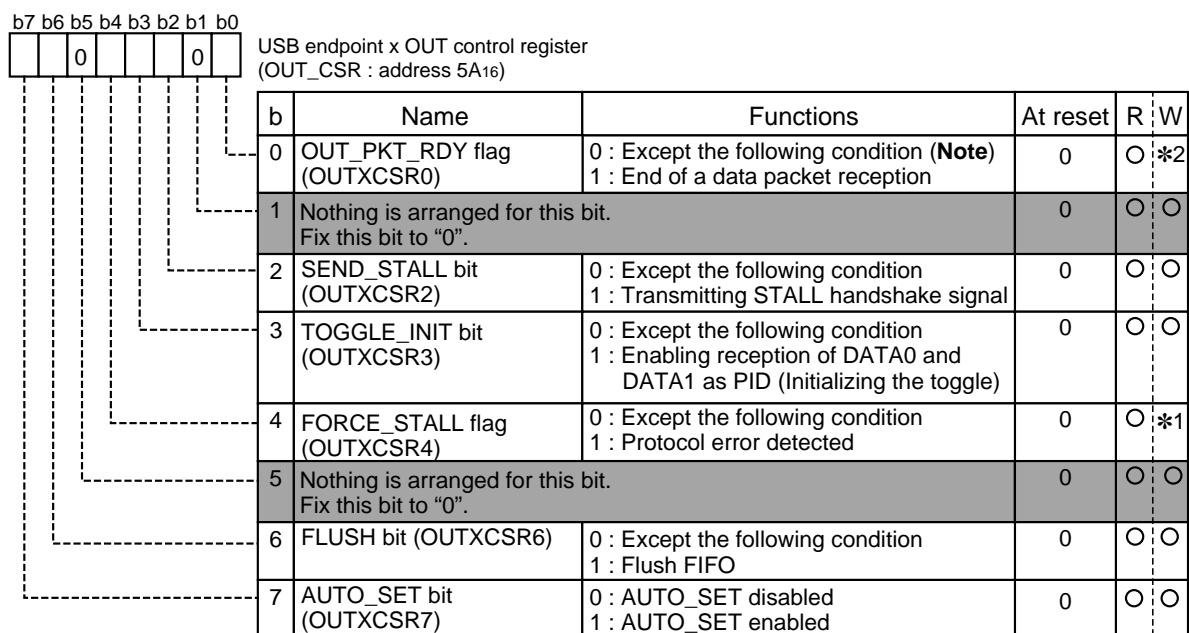
\* 2: When INXCSR7="1", this bit is automatically set to "1".

When INXCSR7="0", writing data to FIFO, and then write "1" to this bit.

\* 3: To use the AUTO\_SET function for an IN transfer when the AUTO\_SET bit is set to "1", set the FIFO to single buffer mode.

Fig. 3.5.43 Structure of USB endpoint x IN control register

### USB endpoint x OUT control register



\* 1: "0" can be set by software, but "1" cannot be set.

\* 2: When OUTXCSR7="1", this bit is automatically set to "1".

When OUTXCSR7="0", writing data to FIFO, and then write "0" to this bit.

Fig. 3.5.44 Structure of USB endpoint x OUT control register (x = 1, 2)

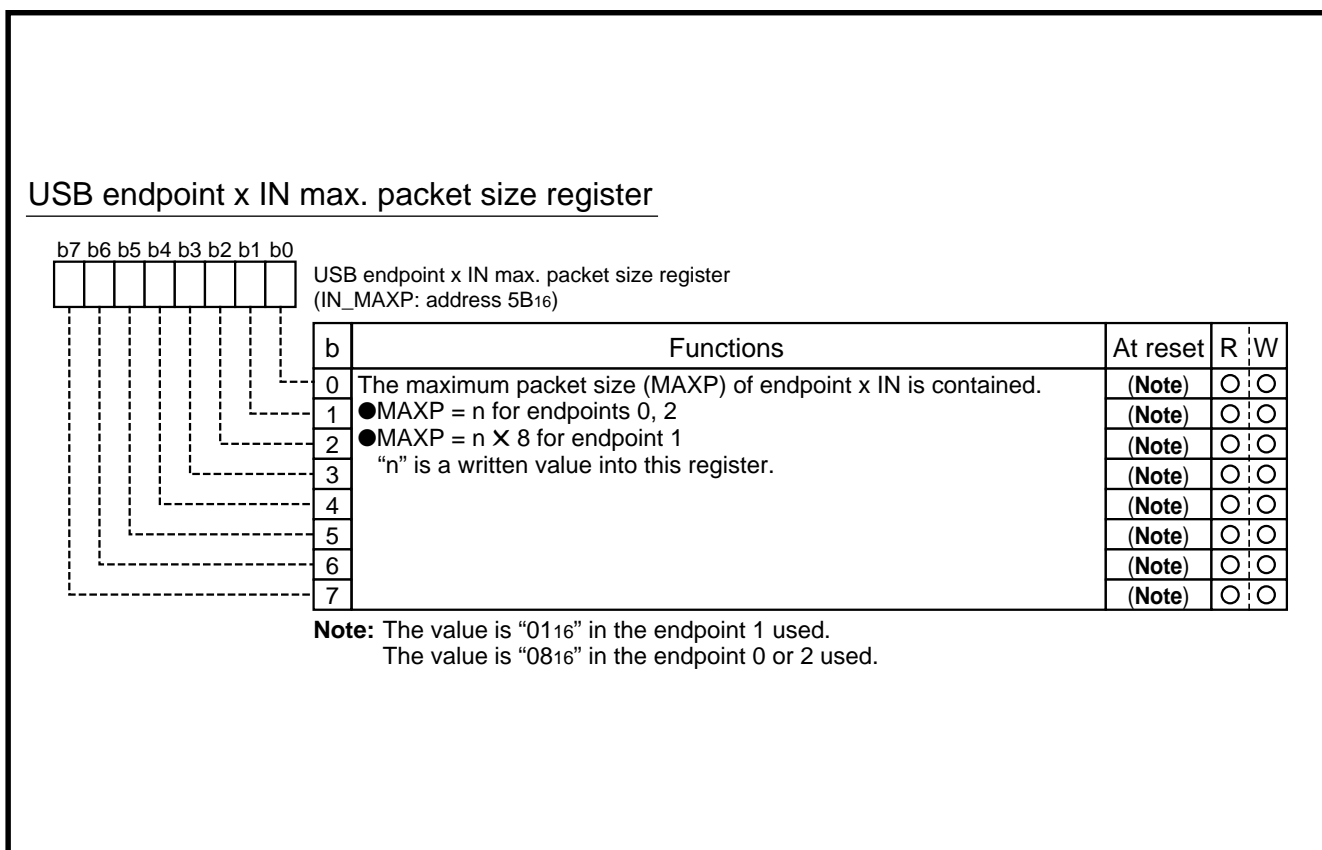


Fig. 3.5.45 Structure of USB endpoint x IN max. packet size register (x = 0 to 2)

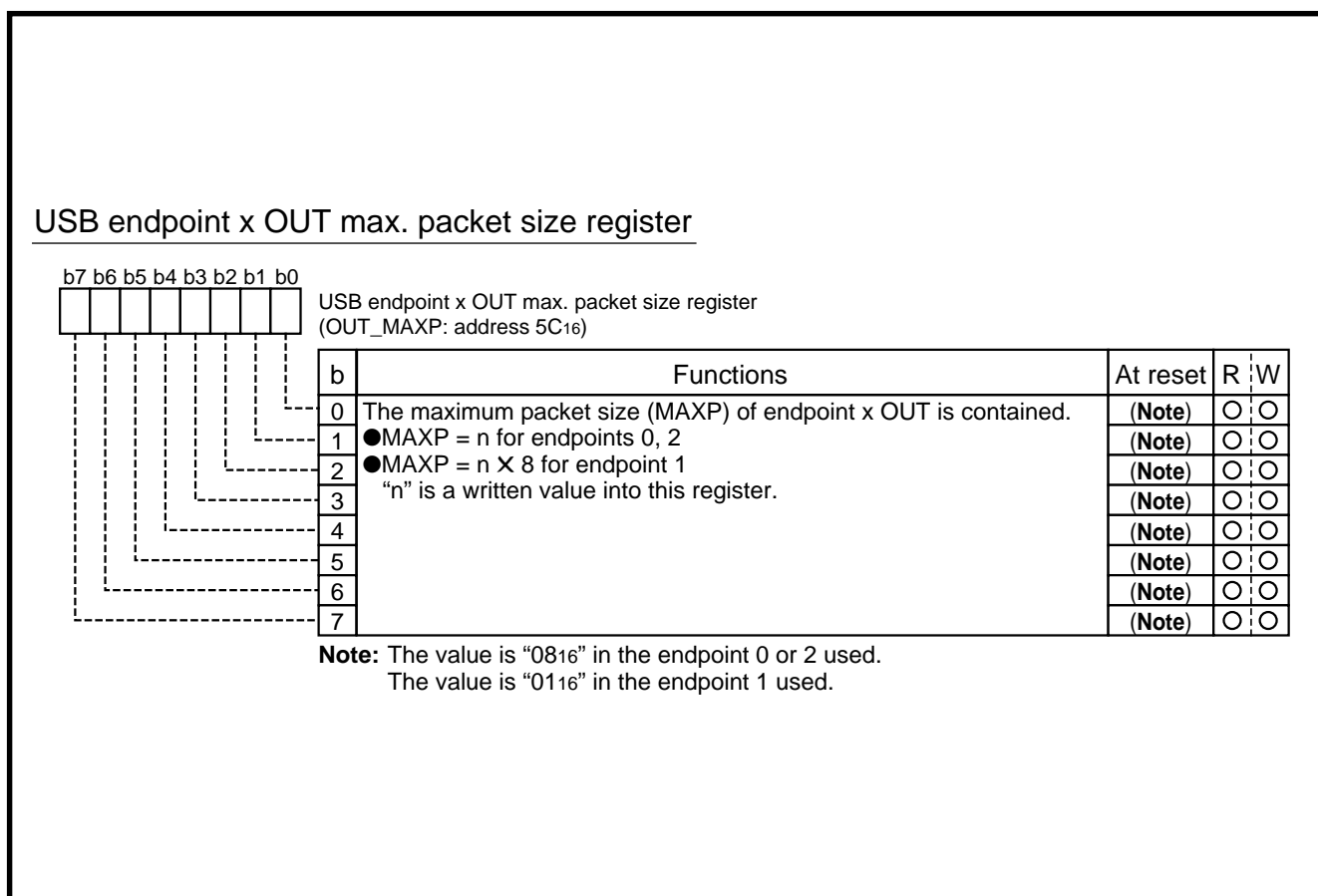


Fig. 3.5.46 Structure of USB endpoint x OUT max. packet size register (x = 0 to 2)

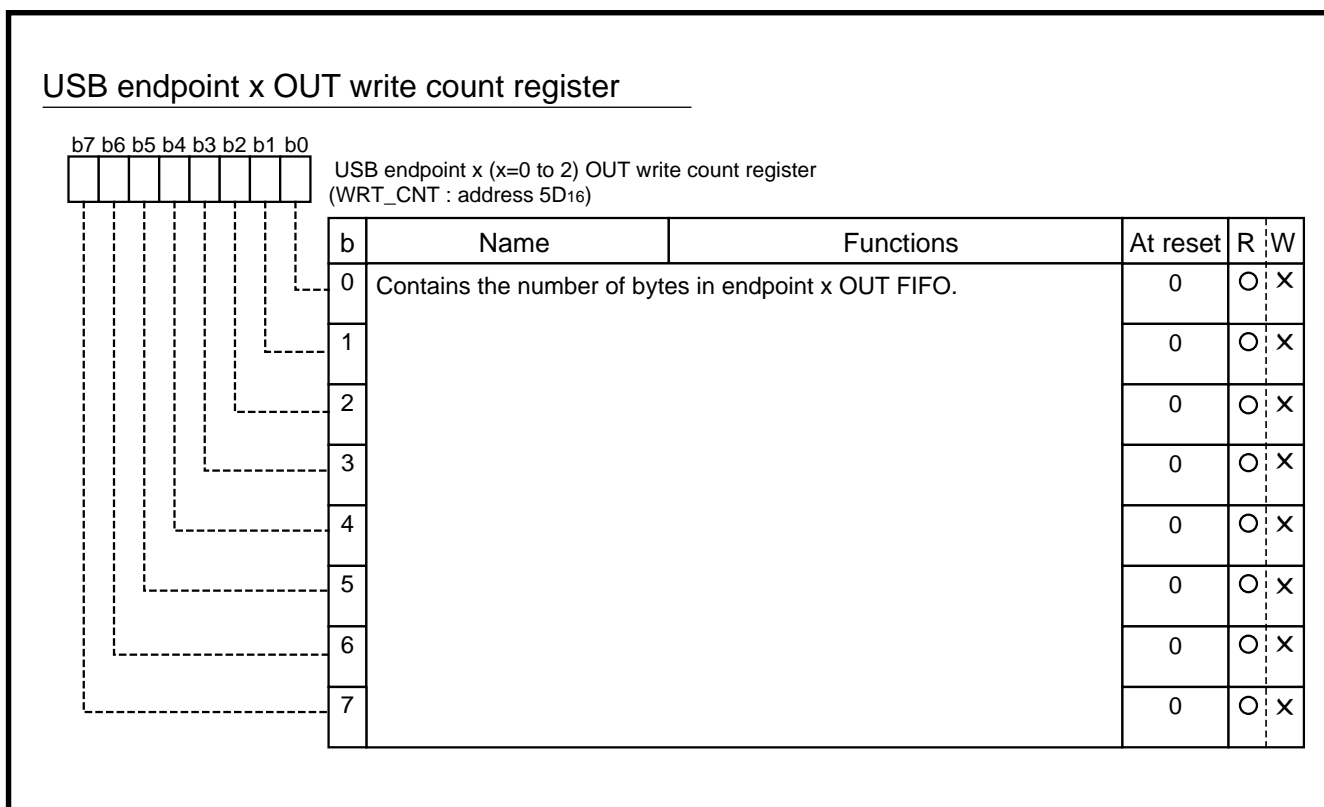


Fig. 3.5.47 Structure of USB endpoint x OUT write control register (x = 0 to 2)

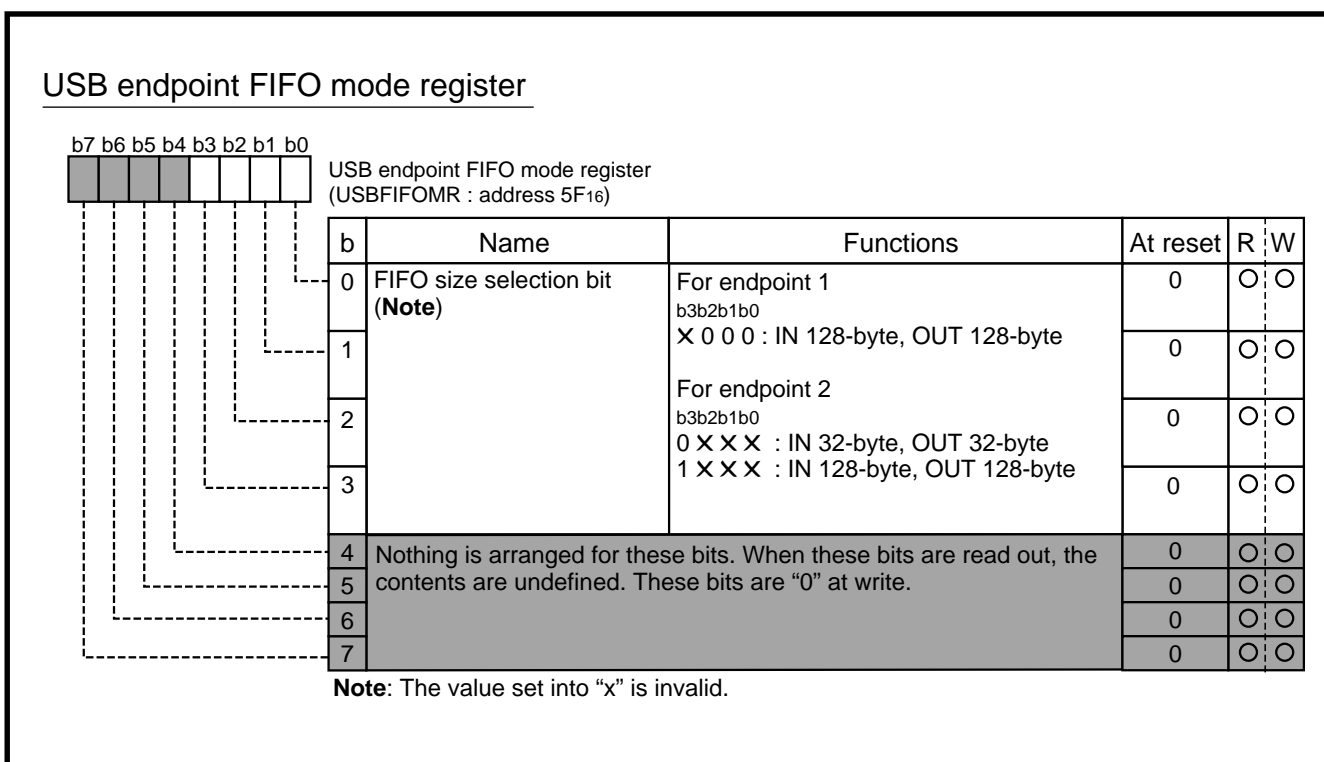


Fig. 3.5.48 Structure of USB endpoint FIFO mode register

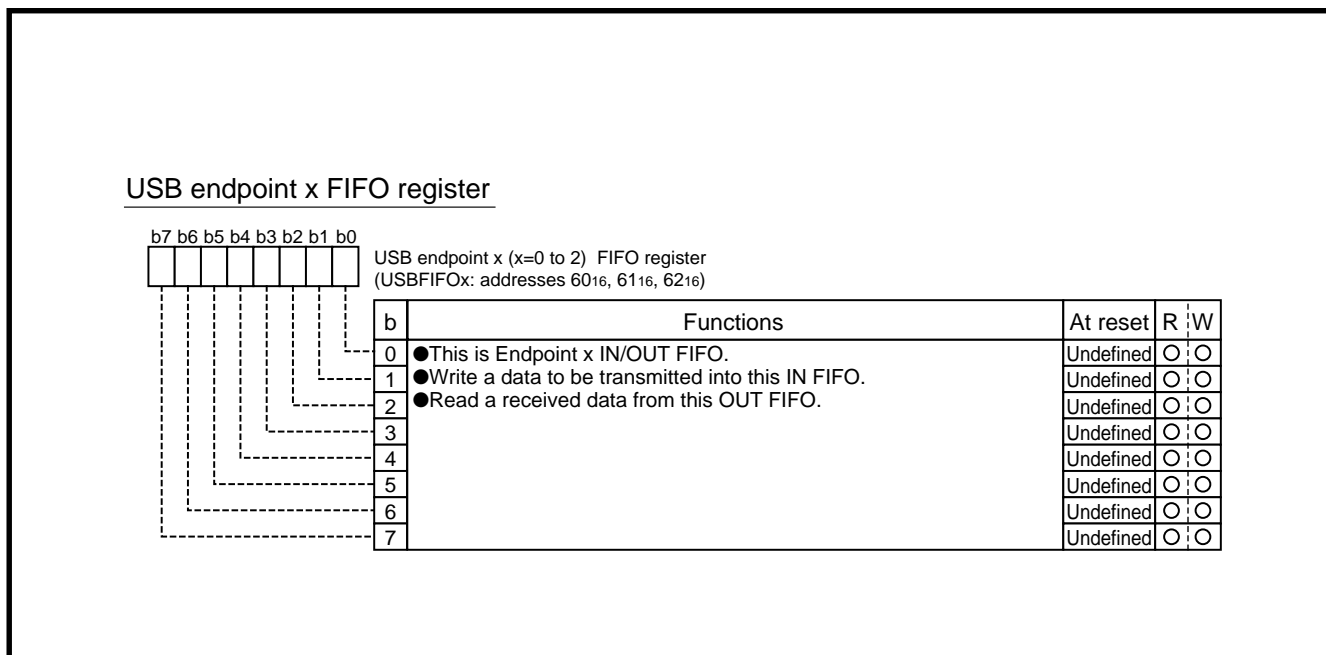


Fig. 3.5.49 Structure of USB endpoint x FIFO register (x = 0 to 2)

## Flash memory control register

b	Name	Functions	At reset	R	W
0	RY/BY status flag (FLCA0)	0 : Busy (being programmed or erased) 1 : Ready	1	○	○
1	CPU rewrite mode select bit (FLCA1) ( <b>Note 2</b> )	0 : Normal mode (Software commands invalid) 1 : CPU rewrite mode (Software commands acceptable)	0	○	○
2	CPU rewrite mode entry flag (FLCA2)	0 : Normal mode 1 : CPU rewrite mode	0	○	×
3	Flash memory reset bit (FLCA3) ( <b>Note 3</b> )	0 : Normal operation 1 : Reset	0	○	○
4	User ROM area / Boot ROM area select bit (FLCA4) ( <b>Note 4</b> )	0 : Interrupt disabled 1 : Interrupt enabled	0	○	○
5	Reserved bits	Indefinite at read. Write "0" at write.	0	○	○
6			0	○	○
7			0	○	○

**Notes** 1: The contents of flash memory control register are "XXX00001" just after reset release.

In the mask ROM version this area is reserved, so that do not write any data to this address.

**2:** For this bit to be set to "1", the user needs to write "0" and then "1" to it in succession. If it is not this procedure, this bit will not be set to "1". Additionally, it is required to ensure that no interrupt will be generated during that interval.

Use the control program in the area except the built-in flash memory for write to this bit.

**3:** This bit is valid when the CPU rewrite mode select bit is "1". Set this bit 3 to "0" subsequently after setting bit 3 to "1".

**4:** Use the control program in the area except the built-in flash memory for write to this bit.

Fig. 3.5.50 Structure of Flash memory control register



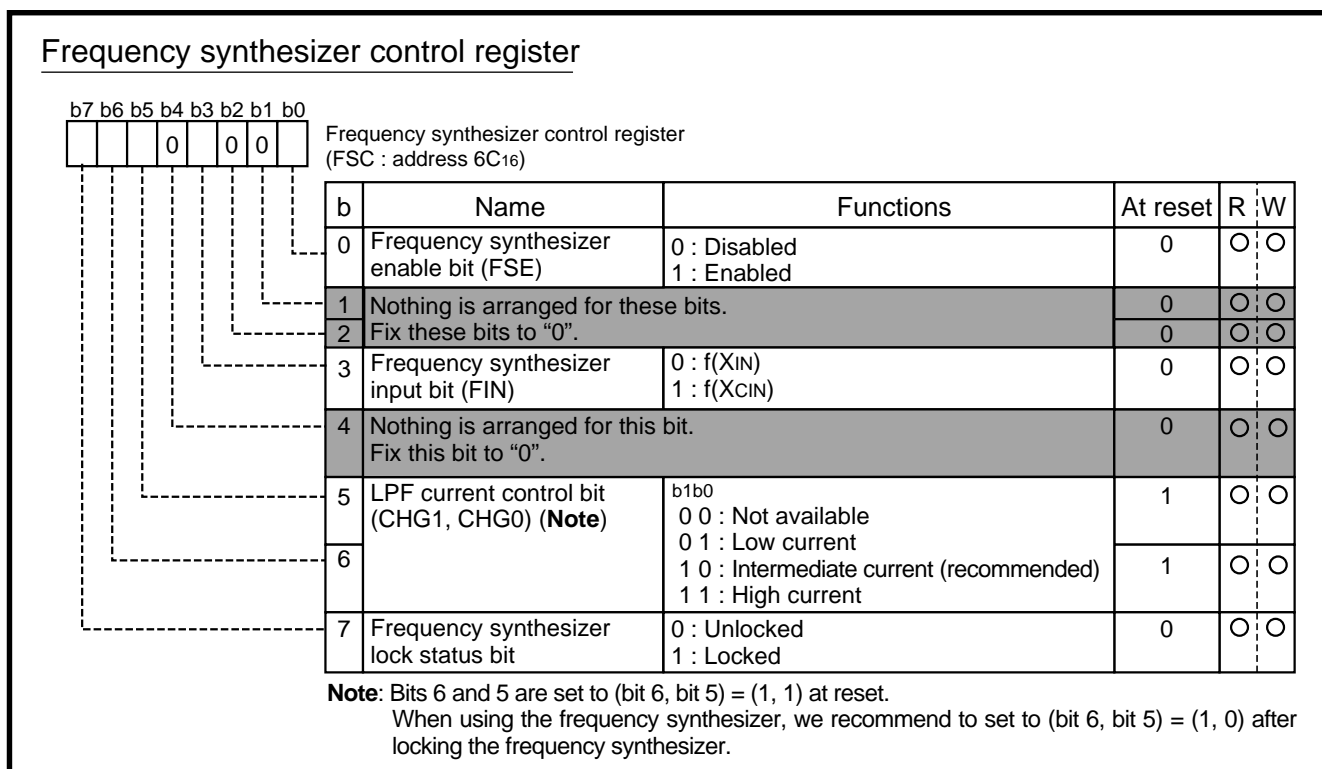


Fig. 3.5.51 Structure of Frequency synthesizer control register

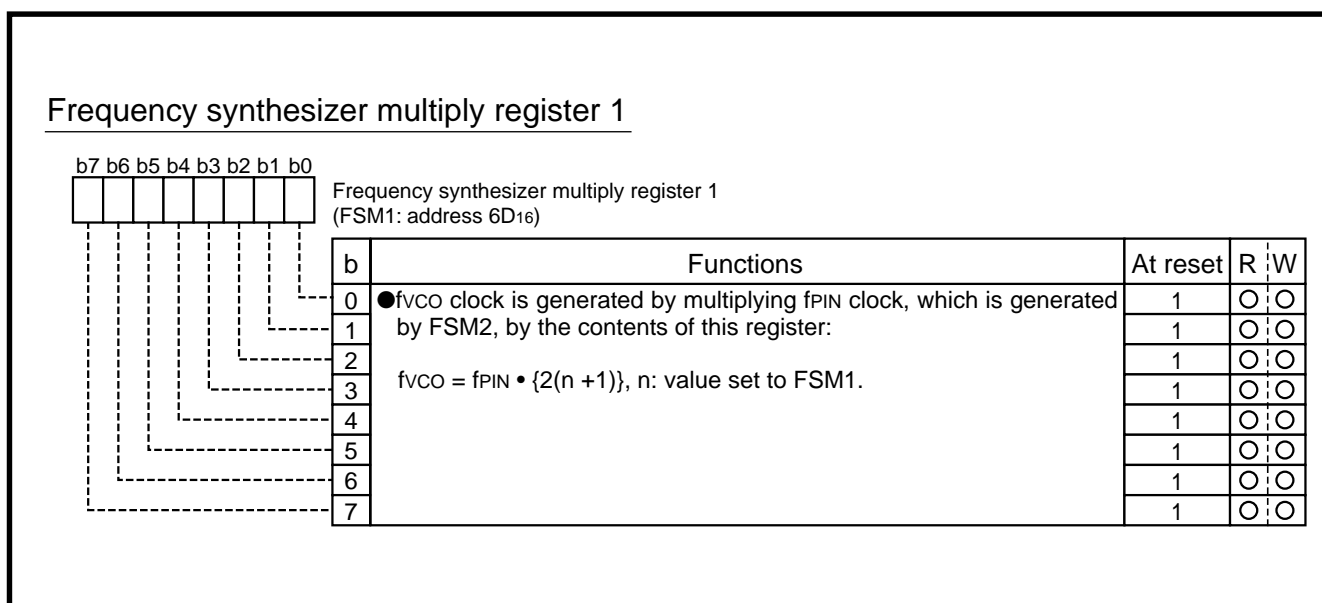


Fig. 3.5.52 Structure of Frequency synthesizer multiply register 1

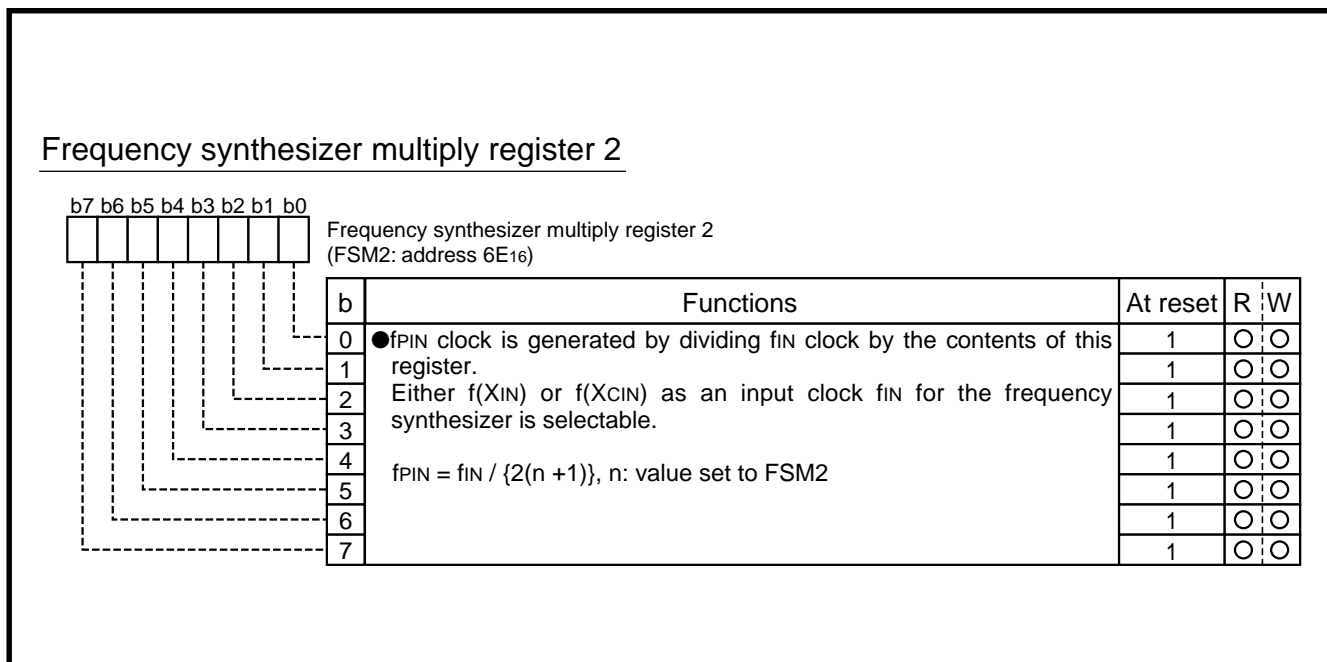


Fig. 3.5.53 Structure of Frequency synthesizer multiply register 2

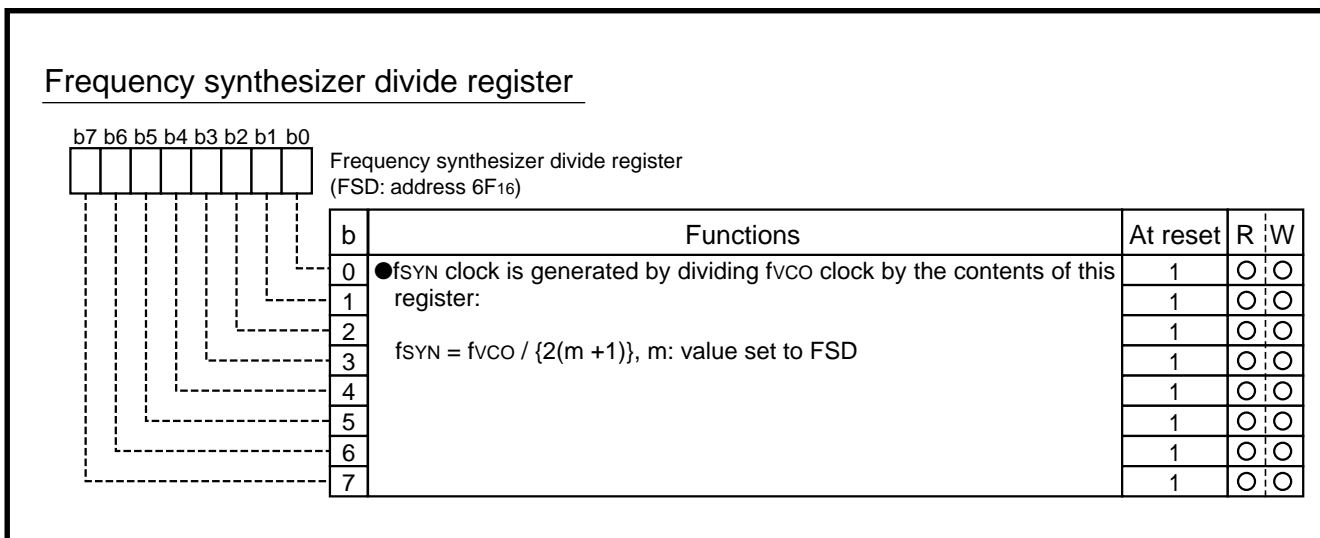


Fig. 3.5.54 Structure of Frequency synthesizer divide register

## ROM code protect control register

b	Name	Functions	At reset	R	W
0	Reserved bits.	Indefinite at read. Write "0" at write.	1	○	○
1			1	○	○
2	ROM code protect level 2 set bits (ROMCP2) (Notes 2, 3)	b3b2 0 0 : Protect enabled	1	○	○
3		0 1 : Protect enabled 1 0 : Protect enabled 1 1 : Protect disabled			
4	ROM code protect reset bits (ROMCR) (Note 4)	b5b4 0 0 : Protect removed	1	○	○
5		0 1 : Protect set bits effective 1 0 : Protect set bits effective 1 1 : Protect set bits effective			
6	ROM code protect level 1 set bits (ROMCP1) (Note 2)	b7b6 0 0 : Protect enabled	1	○	○
7		0 1 : Protect enabled 1 0 : Protect enabled 1 1 : Protect disabled			

**Notes 1:** This area is on the ROM in the mask ROM version.

**2:** When ROM code protect is turned on, the internal flash memory is protected against readout or modification in parallel I/O mode.

**3:** When ROM code protect level 2 is turned on, ROM code readout by a shipment inspection LSI tester, etc. also is inhibited.

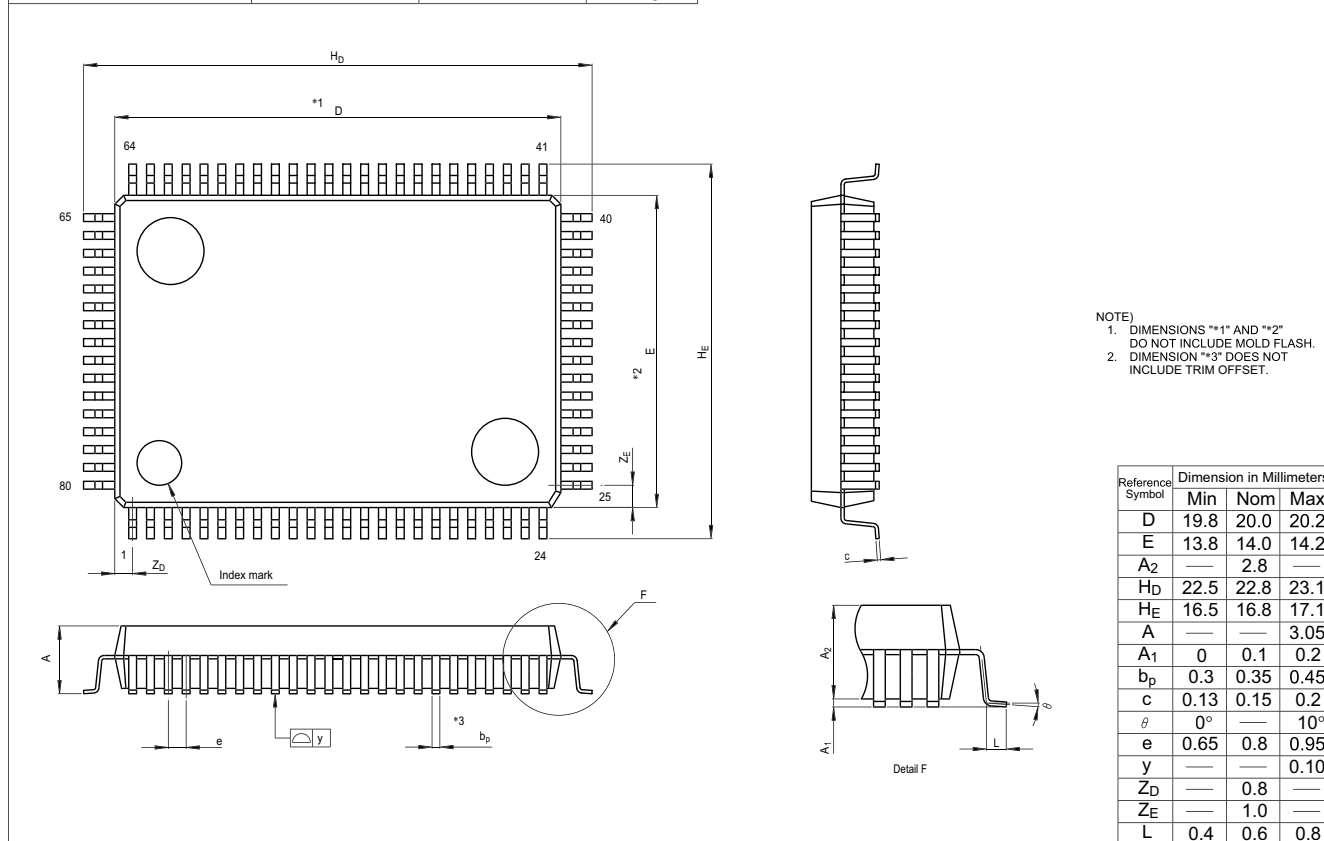
**4:** The ROM code protect reset bits can be used to turn off ROM code protect level 1 and ROM code protect level 2. However, since these bits cannot be modified in parallel I/O mode, they need to be rewritten in standard serial I/O mode or CPU rewrite mode.

Fig. 3.5.55 Structure of ROM code protect control register

### 3.6 Package outline

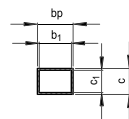
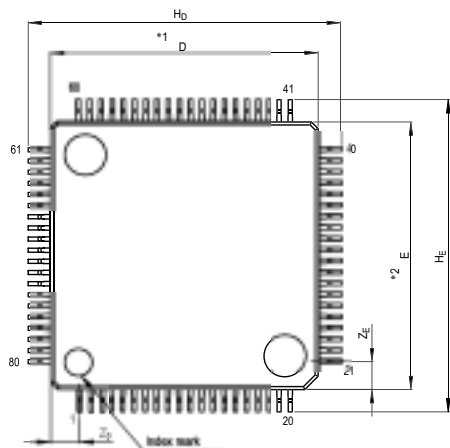
#### PRQP0080GB-A

JEITA Package Code	RENESAS Code	Previous Code	MASS[Typ.]
P-QFP80-14x20-0.80	PRQP0080GB-A	80P6N-A	1.6g

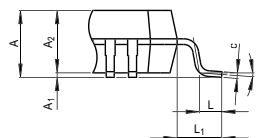
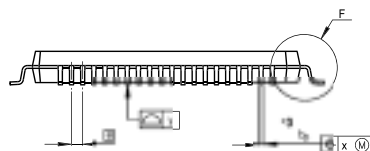


### PLQP0080KB-A

JEITA Package Code	RENESAS Code	Previous Code	MASS[Typ.]
P-LQFP80-12x12-0.50	PLQP0080KB-A	80P6Q-A	0.5g



Terminal cross section

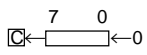


Detail F

NOTE)  
 1. DIMENSIONS \*\*1\* AND \*\*2\* DO NOT INCLUDE MOLD FLASH.  
 2. DIMENSION \*\*3\* DOES NOT INCLUDE TRIM OFFSET.

Reference Symbol	Dimension in Millimeters		
	Min	Nom	Max
D	11.9	12.0	12.1
E	11.9	12.0	12.1
A <sub>2</sub>	—	1.4	—
H <sub>D</sub>	13.8	14.0	14.2
H <sub>E</sub>	13.8	14.0	14.2
A	—	—	1.7
A <sub>1</sub>	0	0.1	0.2
b <sub>p</sub>	0.15	0.20	0.25
b <sub>1</sub>	—	0.18	—
c	0.09	0.145	0.20
c <sub>1</sub>	—	0.125	—
θ	0°	—	10°
ⓐ	—	0.5	—
x	—	—	0.08
y	—	—	0.08
Z <sub>D</sub>	—	1.25	—
Z <sub>E</sub>	—	1.25	—
L	0.3	0.5	0.7
L <sub>1</sub>	—	1.0	—

## 3.7 Machine instructions

Symbol	Function	Details	Addressing mode																				
			IMP			IMM			A			BIT, A, R			ZP			BIT, ZP, R					
			OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#			
ADC (Note 1) (Note 7)	When T = 0 $A \leftarrow A + M + C$  When T = 1 $M(X) \leftarrow M(X) + M + C$	When T = 0, this instruction adds the contents M, C, and A; and stores the results in A and C. When T = 1, this instruction adds the contents of M(X), M and C; and stores the results in M(X) and C. When T=1, the contents of A remain unchanged, but the contents of status flags are changed. M(X) represents the contents of memory where is indicated by X.				69	2	2										65	3	2			
AND (Note 1)	When T = 0 $A \leftarrow A \wedge M$  When T = 1 $M(X) \leftarrow M(X) \wedge M$	When T = 0, this instruction transfers the contents of A and M to the ALU which performs a bit-wise AND operation and stores the result back in A. When T = 1, this instruction transfers the contents M(X) and M to the ALU which performs a bit-wise AND operation and stores the results back in M(X). When T = 1, the contents of A remain unchanged, but status flags are changed. M(X) represents the contents of memory where is indicated by X.				29	2	2										25	3	2			
ASL		This instruction shifts the content of A or M by one bit to the left, with bit 0 always being set to 0 and bit 7 of A or M always being contained in C.							0A	1	1							06	5	2			
BBC	$A_i$ or $M_i = 0?$	This instruction tests the designated bit i of M or A and takes a branch if the bit is 0. The branch address is specified by a relative address. If the bit is 1, next instruction is executed.										$13$ $20_i$ (Note 4)	4	2				$17$ $20_i$ (Note 6)	5	3			
BBS	$A_i$ or $M_i = 1?$	This instruction tests the designated bit i of the M or A and takes a branch if the bit is 1. The branch address is specified by a relative address. If the bit is 0, next instruction is executed.										$03$ $20_i$ (Note 4)	4	2				$07$ $20_i$ (Note 6)	5	3			
BCC (Note 5) (Note 9)	$C = 0?$	This instruction takes a branch to the appointed address if C is 0. The branch address is specified by a relative address. If C is 1, the next instruction is executed.																					
BCS (Note 5) (Note 9)	$C = 1?$	This instruction takes a branch to the appointed address if C is 1. The branch address is specified by a relative address. If C is 0, the next instruction is executed.																					
BEQ (Note 5) (Note 8)	$Z = 1?$	This instruction takes a branch to the appointed address when Z is 1. The branch address is specified by a relative address. If Z is 0, the next instruction is executed.																					
BIT	$A \wedge M$	This instruction takes a bit-wise logical AND of A and M contents; however, the contents of A and M are not modified. The contents of N, V, Z are changed, but the contents of A, M remain unchanged.																24	3	2			
BMI (Note 5) (Note 8)	$N = 1?$	This instruction takes a branch to the appointed address when N is 1. The branch address is specified by a relative address. If N is 0, the next instruction is executed.																					
BNE (Note 5) (Note 8)	$Z = 0?$	This instruction takes a branch to the appointed address if Z is 0. The branch address is specified by a relative address. If Z is 1, the next instruction is executed.																					









Symbol	Function	Details	Addressing mode																	
			IMP			IMM			A			BIT, A			ZP			BIT, ZP		
			OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#
DEX	$X \leftarrow X - 1$	This instruction subtracts one from the current contents of X.	CA	1	1															
DEY	$Y \leftarrow Y - 1$	This instruction subtracts one from the current contents of Y.	88	1	1															
DIV	$A \leftarrow (M(\text{zz} + X + 1), M(\text{zz} + X)) / A$ $M(S) \leftarrow \text{one's complement of Remainder}$ $S \leftarrow S - 1$	This instruction divides the 16-bit data in $M(\text{zz} + X)$ (low-order byte) and $M(\text{zz} + X + 1)$ (high-order byte) by the contents of A. The quotient is stored in A and the one's complement of the remainder is pushed onto the stack.																		
EOR (Note 1)	When $T = 0$ $A \leftarrow A \vee M$  When $T = 1$ $M(X) \leftarrow M(X) \vee M$	When $T = 0$ , this instruction transfers the contents of the M and A to the ALU which performs a bit-wise Exclusive OR, and stores the result in A. When $T = 1$ , the contents of $M(X)$ and M are transferred to the ALU, which performs a bit-wise Exclusive OR and stores the results in $M(X)$ . The contents of A remain unchanged, but status flags are changed. $M(X)$ represents the contents of memory where is indicated by X.				49	2	2							45	3	2			
INC	$A \leftarrow A + 1$ or $M \leftarrow M + 1$	This instruction adds one to the contents of A or M.							3A	1	1				E6	5	2			
INX	$X \leftarrow X + 1$	This instruction adds one to the contents of X.	E8	1	1															
INY	$Y \leftarrow Y + 1$	This instruction adds one to the contents of Y.	C8	1	1															
JMP	If addressing mode is ABS $PCL \leftarrow ADL$ $PCH \leftarrow ADH$ If addressing mode is IND $PCL \leftarrow M(ADH, ADL)$ $PCH \leftarrow M(ADH, ADL + 1)$ If addressing mode is ZP, IND $PCL \leftarrow M(00, ADL)$ $PCH \leftarrow M(00, ADL + 1)$	This instruction jumps to the address designated by the following three addressing modes: Absolute Indirect Absolute Zero Page Indirect Absolute																		
JSR	$M(S) \leftarrow PCH$ $S \leftarrow S - 1$ $M(S) \leftarrow PCL$ $S \leftarrow S - 1$ After executing the above, if addressing mode is ABS, $PCL \leftarrow ADL$ $PCH \leftarrow ADH$ if addressing mode is SP, $PCL \leftarrow ADL$ $PCH \leftarrow FF$ If addressing mode is ZP, IND, $PCL \leftarrow M(00, ADL)$ $PCH \leftarrow M(00, ADL + 1)$	This instruction stores the contents of the PC in the stack, then jumps to the address designated by the following addressing modes: Absolute Special Page Zero Page Indirect Absolute																		
LDA (Note 2)	When $T = 0$ $A \leftarrow M$ When $T = 1$ $M(X) \leftarrow M$	When $T = 0$ , this instruction transfers the contents of M to A. When $T = 1$ , this instruction transfers the contents of M to $M(X)$ . The contents of A remain unchanged, but status flags are changed. $M(X)$ represents the contents of memory where is indicated by X.				A9	2	2							A5	3	2			
LDM	$M \leftarrow nn$	This instruction loads the immediate value in M.													3C	4	3			
LDX	$X \leftarrow M$	This instruction loads the contents of M in X.				A2	2	2							A6	3	2			
LDY	$Y \leftarrow M$	This instruction loads the contents of M in Y.				A0	2	2							A4	3	2			















- Notes**
- 1 : The number of cycles "n" is increased by 3 when T is 1.
  - 2 : The number of cycles "n" is increased by 2 when T is 1.
  - 3 : The number of cycles "n" is increased by 1 when T is 1.
  - 4 : The number of cycles "n" is increased by 2 when branching has occurred.
  - 5 : The number of cycles "n" is increased by 1 when branching to the same page has occurred. The number of cycles "n" is increased by 2 when branching to the other page has occurred.
  - 6 : The number of cycles "n" is increased by 1 when branching to the other page has occurred.
  - 7 : V flag is invalid in decimal operation mode.
  - 8 : When this instruction is executed immediately after executing DEX, DEY, INX, INY, TAX, TSX, TXA, TYA, DEC, INC, ASL, LSR, ROL, or ROR instructions, the number of cycles "n" becomes "3". Furthermore, the number of cycles "n" is increased by 1 (number of cycles "n" is "4") when branching to the same page has occurred. The number of cycles "n" is increased by 2 (number of cycles "n" is "5") when branching to the other page has occurred.
  - 9 : When this instruction is executed immediately after executing ASL, LSR, ROL, or ROR instructions, the number of cycles "n" becomes "3". Furthermore, the number of cycles "n" is increased by 1 (number of cycles "n" is "4") when branching to the same page has occurred. The number of cycles "n" is increased by 2 (number of cycles "n" is "5") when branching to the other page has occurred.


Symbol	Contents	Symbol	Contents
IMP	Implied addressing mode	+	Addition
IMM	Immediate addressing mode	-	Subtraction
A	Accumulator or Accumulator addressing mode	*	Multiplication
BIT, A	Accumulator bit addressing mode	/	Division
BIT, A, R	Accumulator bit relative addressing mode	∧	Logical OR
ZP	Zero page addressing mode	∨	Logical AND
BIT, ZP	Zero page bit addressing mode	⊕	Logical exclusive OR
BIT, ZP, R	Zero page bit relative addressing mode	—	Negation
ZP, X	Zero page X addressing mode	←	Shows direction of data flow
ZP, Y	Zero page Y addressing mode	X	Index register X
ABS	Absolute addressing mode	Y	Index register Y
ABS, X	Absolute X addressing mode	S	Stack pointer
ABS, Y	Absolute Y addressing mode	PC	Program counter
IND	Indirect absolute addressing mode	PS	Processor status register
ZP, IND	Zero page indirect absolute addressing mode	PCH	8 high-order bits of program counter
IND, X	Indirect X addressing mode	PCL	8 low-order bits of program counter
IND, Y	Indirect Y addressing mode	ADH	8 high-order bits of address
REL	Relative addressing mode	ADL	8 low-order bits of address
SP	Special page addressing mode	FF	FF in Hexadecimal notation
C	Carry flag	nn	Immediate value
Z	Zero flag	zz	Zero page address
I	Interrupt disable flag	M	Memory specified by address designation of any addressing mode
D	Decimal mode flag	M(X)	Memory of address indicated by contents of index register X
B	Break flag	M(S)	Memory of address indicated by contents of stack pointer
T	X-modified arithmetic mode flag	M(ADH, ADL)	Contents of memory at address indicated by ADH and ADL, in ADH is 8 high-order bits and ADL is 8 low-order bits.
V	Overflow flag	M(00, ADL)	Contents of address indicated by zero page ADL
N	Negative flag	Ai	Bit i (i = 0 to 7) of accumulator
		Mi	Bit i (i = 0 to 7) of memory
		OP	Opcode
		n	Number of cycles
		#	Number of bytes

## 3.8 List of instruction code

D7 – D4	D3 – D0 Hexadecimal notation	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	BRK	ORA IND, X	JSR ZP, IND	BBS 0, A	—	ORA ZP	ASL ZP	BBS 0, ZP	PHP	ORA IMM	ASL A	SEB 0, A	—	ORA ABS	ASL ABS	SEB 0, ZP
0001	1	BPL	ORA IND, Y	CLT	BBC 0, A	—	ORA ZP, X	ASL ZP, X	BBC 0, ZP	CLC	ORA ABS, Y	DEC A	CLB 0, A	—	ORA ABS, X	ASL ABS, X	CLB 0, ZP
0010	2	JSR ABS	AND IND, X	JSR SP	BBS 1, A	BIT ZP	AND ZP	ROL ZP	BBS 1, ZP	PLP	AND IMM	ROL A	SEB 1, A	BIT ABS	AND ABS	ROL ABS	SEB 1, ZP
0011	3	BMI	AND IND, Y	SET	BBC 1, A	—	AND ZP, X	ROL ZP, X	BBC 1, ZP	SEC	AND ABS, Y	INC A	CLB 1, A	LDM ZP	AND ABS, X	ROL ABS, X	CLB 1, ZP
0100	4	RTI	EOR IND, X	STP	BBS 2, A	COM ZP	EOR ZP	LSR ZP	BBS 2, ZP	PHA	EOR IMM	LSR A	SEB 2, A	JMP ABS	EOR ABS	LSR ABS	SEB 2, ZP
0101	5	BVC	EOR IND, Y	—	BBC 2, A	—	EOR ZP, X	LSR ZP, X	BBC 2, ZP	CLI	EOR ABS, Y	—	CLB 2, A	—	EOR ABS, X	LSR ABS, X	CLB 2, ZP
0110	6	RTS	ADC IND, X	MUL ZP, X	BBS 3, A	TST ZP	ADC ZP	ROR ZP	BBS 3, ZP	PLA	ADC IMM	ROR A	SEB 3, A	JMP IND	ADC ABS	ROR ABS	SEB 3, ZP
0111	7	BVS	ADC IND, Y	—	BBC 3, A	—	ADC ZP, X	ROR ZP, X	BBC 3, ZP	SEI	ADC ABS, Y	—	CLB 3, A	—	ADC ABS, X	ROR ABS, X	CLB 3, ZP
1000	8	BRA	STA IND, X	RRF ZP	BBS 4, A	STY ZP	STA ZP	STX ZP	BBS 4, ZP	DEY	—	TXA	SEB 4, A	STY ABS	STA ABS	STX ABS	SEB 4, ZP
1001	9	BCC	STA IND, Y	—	BBC 4, A	STY ZP, X	STA ZP, X	STX ZP, Y	BBC 4, ZP	TYA	STA ABS, Y	TXS	CLB 4, A	—	STA ABS, X	—	CLB 4, ZP
1010	A	LDY IMM	LDA IND, X	LDX IMM	BBS 5, A	LDY ZP	LDA ZP	LDX ZP	BBS 5, ZP	TAY	LDA IMM	TAX	SEB 5, A	LDY ABS	LDA ABS	LDX ABS	SEB 5, ZP
1011	B	BCS	LDA IND, Y	JMP ZP, IND	BBC 5, A	LDY ZP, X	LDA ZP, X	LDX ZP, Y	BBC 5, ZP	CLV	LDA ABS, Y	TSX	CLB 5, A	LDY ABS, X	LDA ABS, X	LDX ABS, Y	CLB 5, ZP
1100	C	CPY IMM	CMP IND, X	WIT	BBS 6, A	CPY ZP	CMP ZP	DEC ZP	BBS 6, ZP	INY	CMP IMM	DEX	SEB 6, A	CPY ABS	CMP ABS	DEC ABS	SEB 6, ZP
1101	D	BNE	CMP IND, Y	—	BBC 6, A	—	CMP ZP, X	DEC ZP, X	BBC 6, ZP	CLD	CMP ABS, Y	—	CLB 6, A	—	CMP ABS, X	DEC ABS, X	CLB 6, ZP
1110	E	CPX IMM	SBC IND, X	DIV ZP, X	BBS 7, A	CPX ZP	SBC ZP	INC ZP	BBS 7, ZP	INX	SBC IMM	NOP	SEB 7, A	CPX ABS	SBC ABS	INC ABS	SEB 7, ZP
1111	F	BEQ	SBC IND, Y	—	BBC 7, A	—	SBC ZP, X	INC ZP, X	BBC 7, ZP	SED	SBC ABS, Y	—	CLB 7, A	—	SBC ABS, X	INC ABS, X	CLB 7, ZP

 : 3-byte instruction

 : 2-byte instruction

 : 1-byte instruction

### 3.9 SFR memory map

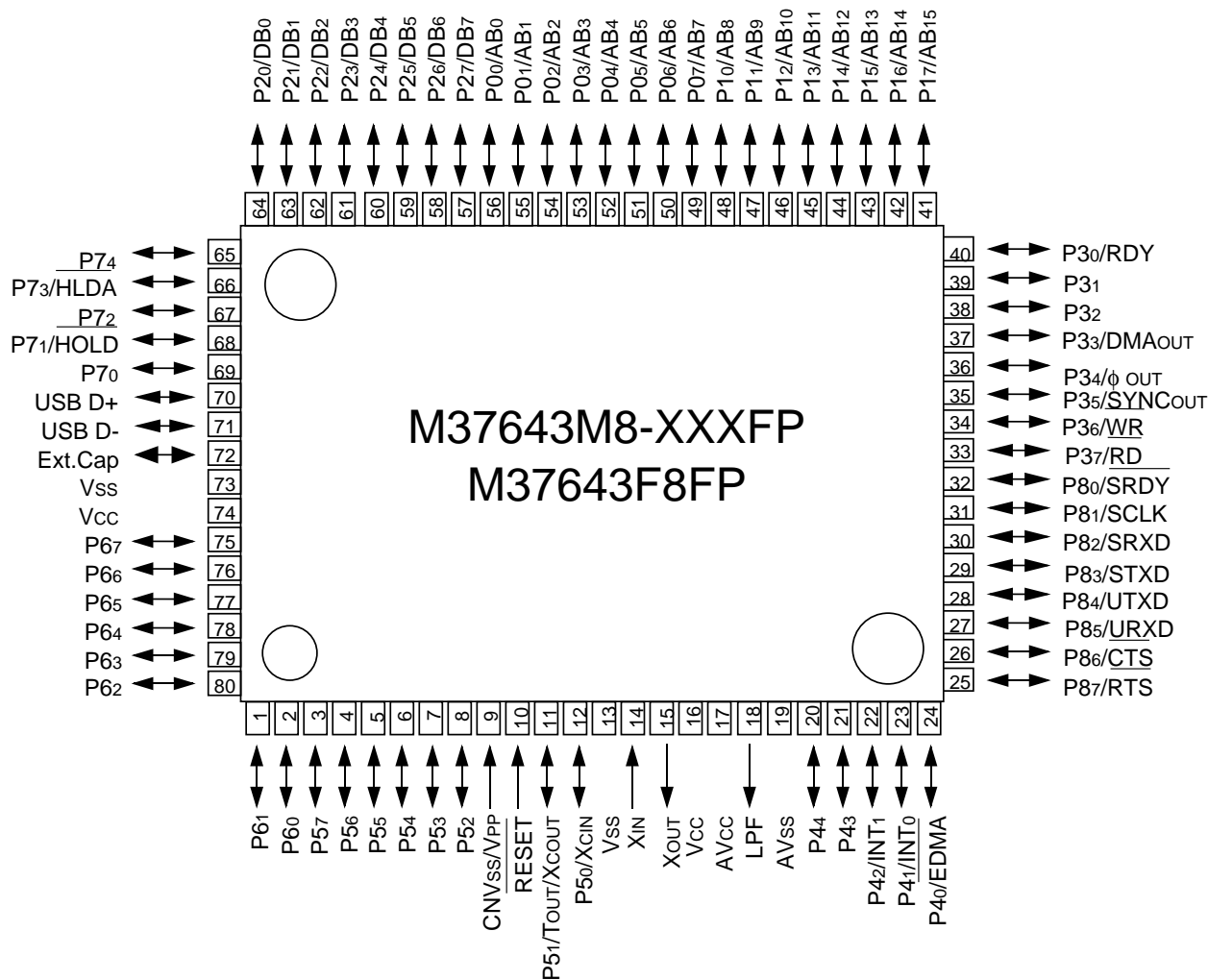
0000 <sub>16</sub>	CPU mode register A (CPUA)	0038 <sub>16</sub>	Reserved ( <b>Note 1</b> )
0001 <sub>16</sub>	CPU mode register B (CPUB)	0039 <sub>16</sub>	Reserved ( <b>Note 1</b> )
0002 <sub>16</sub>	Interrupt request register A (IREQA)	003A <sub>16</sub>	Reserved ( <b>Note 1</b> )
0003 <sub>16</sub>	Interrupt request register B (IREQB)	003B <sub>16</sub>	Reserved ( <b>Note 1</b> )
0004 <sub>16</sub>	Interrupt request register C (IREQC)	003C <sub>16</sub>	Reserved ( <b>Note 1</b> )
0005 <sub>16</sub>	Interrupt control register A (ICONA)	003D <sub>16</sub>	Reserved ( <b>Note 1</b> )
0006 <sub>16</sub>	Interrupt control register B (ICONB)	003E <sub>16</sub>	Reserved ( <b>Note 1</b> )
0007 <sub>16</sub>	Interrupt control register C (ICONC)	003F <sub>16</sub>	DMAC index and status register (DMAIS)
0008 <sub>16</sub>	Port P0 (P0)	0040 <sub>16</sub>	DMAC channel x mode register 1 (DMAx1)
0009 <sub>16</sub>	Port P0 direction register (P0D)	0041 <sub>16</sub>	DMAC channel x mode register 2 (DMAx2)
000A <sub>16</sub>	Port P1 (P1)	0042 <sub>16</sub>	DMAC channel x source register Low (DMAxSL)
000B <sub>16</sub>	Port P1 direction register (P1D)	0043 <sub>16</sub>	DMAC channel x source register High (DMAxSH)
000C <sub>16</sub>	Port P2 (P2)	0044 <sub>16</sub>	DMAC channel x destination register Low (DMAxDL)
000D <sub>16</sub>	Port P2 direction register (P2D)	0045 <sub>16</sub>	DMAC channel x destination register High (DMAxDH)
000E <sub>16</sub>	Port P3 (P3)	0046 <sub>16</sub>	DMAC channel x transfer count register Low (DMAxCL)
000F <sub>16</sub>	Port P3 direction register (P3D)	0047 <sub>16</sub>	DMAC channel x transfer count register High (DMAxCH)
0010 <sub>16</sub>	Port control register (PTC)	0048 <sub>16</sub>	Reserved ( <b>Note 1</b> )
0011 <sub>16</sub>	Interrupt polarity select register (IPOL)	0049 <sub>16</sub>	Reserved ( <b>Note 1</b> )
0012 <sub>16</sub>	Port P2 pull-up control register (PUP2)	004A <sub>16</sub>	Reserved ( <b>Note 1</b> )
0013 <sub>16</sub>	USB control register (USBC)	004B <sub>16</sub>	Reserved ( <b>Note 1</b> )
0014 <sub>16</sub>	Port P6 (P6)	004C <sub>16</sub>	Reserved ( <b>Note 1</b> )
0015 <sub>16</sub>	Port P6 direction register (P6D)	004D <sub>16</sub>	Reserved ( <b>Note 1</b> )
0016 <sub>16</sub>	Port P5 (P5)	004E <sub>16</sub>	Reserved ( <b>Note 1</b> )
0017 <sub>16</sub>	Port P5 direction register (P5D)	004F <sub>16</sub>	Reserved ( <b>Note 1</b> )
0018 <sub>16</sub>	Port P4 (P4)	0050 <sub>16</sub>	USB address register (USBA)
0019 <sub>16</sub>	Port P4 direction register (P4D)	0051 <sub>16</sub>	USB power management register (USBPM)
001A <sub>16</sub>	Port P7 (P7)	0052 <sub>16</sub>	USB interrupt status register 1 (USBIS1)
001B <sub>16</sub>	Port P7 direction register (P7D)	0053 <sub>16</sub>	USB interrupt status register 2 (USBIS2)
001C <sub>16</sub>	Port P8 (P8)	0054 <sub>16</sub>	USB interrupt enable register 1 (USBIE1)
001D <sub>16</sub>	Port P8 direction register (P8D)	0055 <sub>16</sub>	USB interrupt enable register 2 (USBIE2)
001E <sub>16</sub>	Reserved ( <b>Note 1</b> )	0056 <sub>16</sub>	Reserved ( <b>Note 1</b> )
001F <sub>16</sub>	Clock control register (CCR)	0057 <sub>16</sub>	Reserved ( <b>Note 1</b> )
0020 <sub>16</sub>	Reserved ( <b>Note 1</b> )	0058 <sub>16</sub>	USB endpoint index register (USBINDEX)
0021 <sub>16</sub>	Reserved ( <b>Note 1</b> )	0059 <sub>16</sub>	USB endpoint x IN control register (IN_CSR)
0022 <sub>16</sub>	Reserved ( <b>Note 1</b> )	005A <sub>16</sub>	USB endpoint x OUT control register (OUT_CSR)
0023 <sub>16</sub>	Reserved ( <b>Note 1</b> )	005B <sub>16</sub>	USB endpoint x IN max. packet size register (IN_MAXP)
0024 <sub>16</sub>	Timer 1 (T1)	005C <sub>16</sub>	USB endpoint x OUT max. packet size register (OUT_MAXP)
0025 <sub>16</sub>	Timer 2 (T2)	005D <sub>16</sub>	USB endpoint x OUT write count register (WRT_CNT)
0026 <sub>16</sub>	Timer 3 (T3)	005E <sub>16</sub>	Reserved ( <b>Note 1</b> )
0027 <sub>16</sub>	Reserved ( <b>Note 1</b> )	005F <sub>16</sub>	USB endpoint FIFO mode register (USBFIFOMR)
0028 <sub>16</sub>	Reserved ( <b>Note 1</b> )	0060 <sub>16</sub>	USB endpoint 0 FIFO (USBFIFO0)
0029 <sub>16</sub>	Timer 123 mode register (T123M)	0061 <sub>16</sub>	USB endpoint 1 FIFO (USBFIFO1)
002A <sub>16</sub>	Serial I/O shift register (SIOSHT)	0062 <sub>16</sub>	USB endpoint 2 FIFO (USBFIFO2)
002B <sub>16</sub>	Serial I/O control register 1 (SIOCON1)	0063 <sub>16</sub>	Reserved ( <b>Note 1</b> )
002C <sub>16</sub>	Serial I/O control register 2 (SIOCON2)	0064 <sub>16</sub>	Reserved ( <b>Note 1</b> )
002D <sub>16</sub>	Reserved ( <b>Note 1</b> )	0065 <sub>16</sub>	Reserved ( <b>Note 1</b> )
002E <sub>16</sub>	Reserved ( <b>Note 1</b> )	0066 <sub>16</sub>	Reserved ( <b>Note 1</b> )
002F <sub>16</sub>	Reserved ( <b>Note 1</b> )	0067 <sub>16</sub>	Reserved ( <b>Note 1</b> )
0030 <sub>16</sub>	UART mode register (UMOD)	0068 <sub>16</sub>	Reserved ( <b>Note 1</b> )
0031 <sub>16</sub>	UART baud rate generator (UBRG)	0069 <sub>16</sub>	Reserved ( <b>Note 1</b> )
0032 <sub>16</sub>	UART status register (USTS)	006A <sub>16</sub>	Flash memory control register (FMCR) ( <b>Note 2</b> )
0033 <sub>16</sub>	UART control register (UCON)	006B <sub>16</sub>	Reserved ( <b>Note 1</b> )
0034 <sub>16</sub>	UART transmit/receive buffer register 1 (UTRB1)	006C <sub>16</sub>	Frequency synthesizer control register (FSC)
0035 <sub>16</sub>	UART transmit/receive buffer register 2 (UTRB2)	006D <sub>16</sub>	Frequency synthesizer multiply register 1 (FSM1)
0036 <sub>16</sub>	UART RTS control register (URTSC)	006E <sub>16</sub>	Frequency synthesizer multiply register 2 (FSM2)
0037 <sub>16</sub>	Reserved ( <b>Note 1</b> )	006F <sub>16</sub>	Frequency synthesizer divide register (FSD)
		FFC9 <sub>16</sub>	ROM code protect control register (ROMCP) ( <b>Note 3</b> )

**Notes 1:** Do not write any data to this addresses, because these areas are reserved.

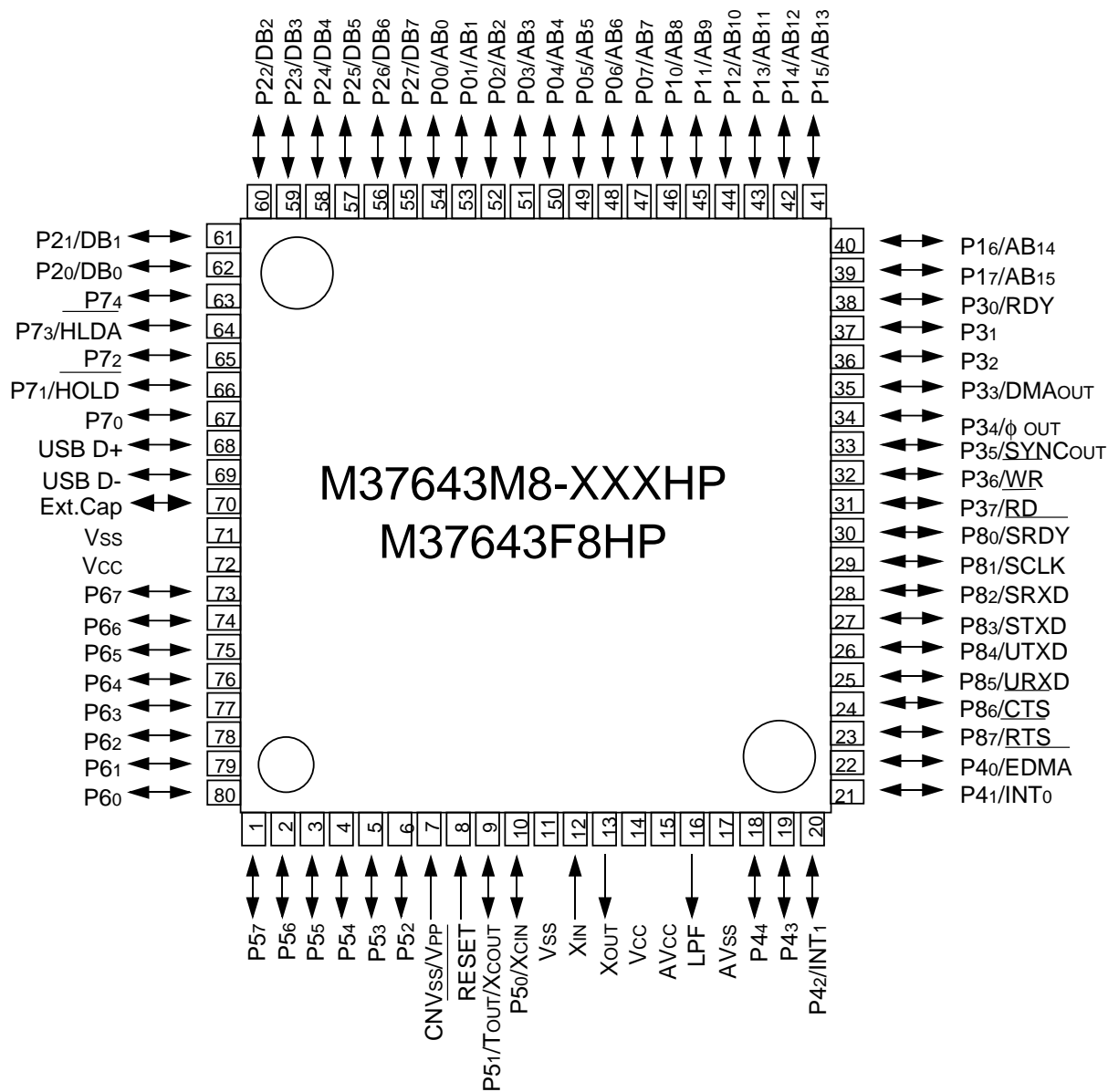
**2:** This area is reserved in the mask ROM version.

**3:** This area is on the ROM in the mask ROM version.

### 3.10 Pin configuration



Package type: PRQP0080GB-A (Top view)



Package type: PLQP0080KB-A (Top view)

REVISION HISTORY

7643 Group User's Manual

Rev.	Date	Description	
		Page	Summary
1.00	Jan 30, 2004	–	First edition issued
2.00	Aug 28, 2006	All pages	Package names “80P6N-A” → “PRQP0080GB-A” revised Package names “80P6Q-A” → “PLQP0080KB-A” revised
		Chapter 1	
		52	Fig. 44 “5: To use the AUTO_SET function .... to single buffer mode.” added
		61	CLOCK GENERATING CIRCUIT; “No external resistor is needed .... resistor exists on-chip.” → “No external resistor is needed .... depending on conditions.)
			Fig. 56; Pulled up added, NOTE added
		98	UART; “•Do not update .... an undefined data might be output.” added
		99	USB; “•To use the AUTO_SET function .... to single buffer mode.” added
		101	Power Source Voltage added
		102	USB Communication added
			“For the mask ROM confirmation .... <a href="http://www.infocom.maec.co.jp/indexe.htm">http://www.infocom.maec.co.jp/indexe.htm</a> ” → “For the mask ROM confirmation .... ( <a href="http://www.renesas.com">http://www.renesas.com</a> ).”
		Chapter 2	
		91	2.5.6 “64-byte”, “64 bytes” → “128-byte”, “128 bytes” “USB endpoint 1” → “USB endpoint 2”
		92-94	Fig. 2.5.16, Fig. 2.5.17, Fig. 2.5.18 revised
		113	(8) USB endpoint 1, 2 IN control register; “(this bit is automatically set to “1” when AUTO_SET bit is “1”)” deleted
		115	Fig. 2.6.14 “3: To use the AUTO_SET function .... to single buffer mode.” added
		124	2.6.4 “- When AUTO_SET bit = “1”.....packet size) is transferred” deleted
		165	2.6.10 (4) USB Communication added
		166	(5) Registers and bits; “• To use the AUTO_SET function .... to single buffer mode.” added
		Chapter 3	
		29	3.3.5 (4) USB Communication added
			(5) “•To read from the USB endpoint x .... then the higher byte.” deleted
		30	(5) Registers and bits; “• To use the AUTO_SET function .... to single buffer mode.” added
		72	Fig. 3.5.43 “3: To use the AUTO_SET function .... to single buffer mode.” added
		82, 83	3.6 Package outline revised

---

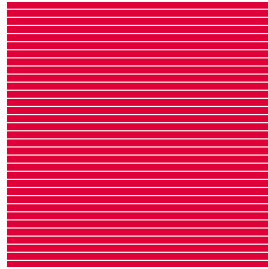
**RENESAS 8-BIT SINGLE-CHIP MICROCOMPUTER  
USER'S MANUAL  
7643 Group**

**Publication Data : Rev.1.00 Jan 30, 2004**

**Rev.2.00 Aug 28, 2006**

**Published by : Sales Strategic Planning Div.  
Renesas Technology Corp.**

# 7643 Group User's Manual



Renesas Technology Corp.  
2-6-2, Ote-machi, Chiyoda-ku, Tokyo, 100-0004, Japan