



---

The following document contains information on Cypress products. Although the document is marked with the name "Spansion" and "Fujitsu", the company that originally developed the specification, Cypress will continue to offer these products to new and existing customers.

**Continuity of Specifications**

There is no change to this document as a result of offering the device as a Cypress product. Any changes that have been made are the result of normal document improvements and are noted in the document history page, where supported. Future revisions will occur when appropriate, and changes will be noted in a document history page.

**Continuity of Ordering Part Numbers**

Cypress continues to support existing part numbers. To order these products, please use only the Ordering Part Numbers listed in this document.

**For More Information**

Please contact your local sales office for additional information about Cypress products and solutions.

**About Cypress**

Cypress (NASDAQ: CY) delivers high-performance, high-quality solutions at the heart of today's most advanced embedded systems, from automotive, industrial and networking platforms to highly interactive consumer and mobile devices. With a broad, differentiated product portfolio that includes NOR flash memories, F-RAM™ and SRAM, Traveo™ microcontrollers, the industry's only PSoC® programmable system-on-chip solutions, analog and PMIC Power Management ICs, CapSense® capacitive touch-sensing controllers, and Wireless BLE Bluetooth® Low-Energy and USB connectivity solutions, Cypress is committed to providing its customers worldwide with consistent innovation, best-in-class support and exceptional system value.

**F<sup>2</sup>MC-16LX**  
16-bit Microcontroller  
**MB90330A Series**  
**HARDWARE MANUAL**



# **F<sup>2</sup>MC-16LX**

## **16-bit Microcontroller**

# **MB90330A Series**

# **HARDWARE MANUAL**

For the information for microcontroller supports, see the following web site.

This web site includes the "**Customer Design Review Supplement**" which provides the latest cautions on system development and the minimal requirements to be checked to prevent problems before the system development.

<http://edevic.fujitsu.com/micom/en-support/>

**FUJITSU MICROELECTRONICS LIMITED**





# PREFACE

## ■ Purpose of this document and intended reader

We sincerely thank you for your continued use of Fujitsu semiconductor products.

MB90330A series is a 16-bit microcontroller designed for applications such as personal computer peripheral device requiring USB communication. The USB function is not only function operation of 12M bps but also simple HOST operation.

MB90330A series has functions which are suitable for controlling personal computer peripheral devices such as display and audio and mobile devices supporting the USB communication.

This manual describes the functions and operations of the MB90330A Series for engineers who develop products using the MB90330A Series. Please read through this manual.

For more information on various instructions, refer to "Instruction Manual".

Note: F<sup>2</sup>MC is the abbreviation of FUJITSU Flexible Microcontroller.

## ■ Trademarks

The company names and brand names herein are the trademarks or registered trademarks of their respective owners.

## ■ Sample program

We provide sample programs free of charge to operate peripheral functions of the F<sup>2</sup>MC-16LX family. The programs can be used to check the operational specification and usage of our microcontroller device.

MPU/MCU Support Information

<http://edevic.fujitsu.com/micom/en-support/>

Note: The sample programs are subject to change without notice. The software is designed to show the standard operation and usage of the product, therefore, it must be used after full evaluation before using it on your system. Moreover, we assume no liability for any damage resulting from or caused by the use of the programs.

## ■ Organization of this document

This manual contains the following 27 chapters and appendix.

### Chapter 1 OVERVIEW

This chapter describes basics to give the understanding of the MB90330A series as a whole such as the features, block diagrams, and overviews of the functions.

### Chapter 2 CPU

This chapter describes the basic knowledge on architecture, specifications, instructions, and others to provide better understanding of the CPU core functions of the MB90330A Series.

### **Chapter 3 INTERRUPT**

This chapter describes the overview of interruptions, interrupt vector and interrupt cause, register configuration/function, and interrupt processing operations.

### **Chapter 4 Reset**

This chapter describes the overview of reset, reset cause and oscillation stabilization wait time, and reset operation.

### **Chapter 5 Clock**

This chapter describes the overview of the clock, register configuration/function, clock mode, and oscillation stability wait time.

### **Chapter 6 Low-Power Consumption Mode**

This chapter describes the overview of the low-power consumption mode, register configuration/function, and operation of the low-power consumption mode.

### **Chapter 7 Mode setting**

This chapter describes the overview of mode settings, mode pin, mode data, and operation in each mode of mode setting.

### **Chapter 8 I/O port**

This chapter describes the overview of the I/O port and the register configuration/function used in the I/O port.

### **Chapter 9 Time-base timer**

This chapter describes the overview of the time-base time, register configuration/function, interrupt, and operation of the time-base timer.

### **Chapter 10 Watchdog timer**

This chapter describes the overview of the watchdog timer, register configuration/function, and operation of the watchdog timer.

### **Chapter 11 Watch timer**

This chapter describes a overview of the watch timer, functions and configurations of its registers, and its operation.

### **Chapter 12 16-bit I/O timer**

This chapter describes a overview of the 16-bit I/O timer, the functions and configurations of its registers, and its operation.

### **Chapter 13 USB function**

This chapter describes the overview, block diagram, register, and operation of the USB function.

### **Chapter 14 USB HOST**

This chapter describes the features of the USB HOST, difference from the USB HOST, block diagrams, registers and operations of USB HOST, and each token flowchart of USB HOST.

### **Chapter 15 PWC timer**

This chapter describes an overview of PWC timer, the configuration and function of register, and the PWC timer operation and precaution.

### **Chapter 16 16-bit reload timer**

This chapter describes an overview of 16-bit reload timer, the configuration and functions of register and the 16-bit reload timer operation.

#### **Chapter 17 8/16-bit PPG timer**

This chapter describes an overview of 8/16-bit PPG timer, the configuration and functions of register, and the 8/16-bit PPG timer operation.

#### **Chapter 18 DTP/External interrupt**

This chapter describes an overview of DTP/external interrupt, the configuration and functions of register, and the DTP/external interrupt operation.

#### **Chapter 19 8/10-bit A/D converter**

This chapter describes the overview of the 8/10-bit A/D converter, the configuration/functions of the register, and the operation of the 8/10-bit A/D converter.

#### **Chapter 20 Extended I/O serial interface**

This chapter describes an overview of the extended I/O serial interface, the configuration and function of registers, and operations of extended I/O serial interface.

#### **Chapter 21 UART**

This chapter describes the overview of the UART, the configuration/functions of the register, the operation of the UART, the usage note of the UART, and the example of UART program.

#### **Chapter 22 I<sup>2</sup>C interface**

This chapter gives an overview of I<sup>2</sup>C interface, the configuration and functions of registers, and operations of I<sup>2</sup>C interface.

#### **Chapter 23 ROM mirror function selection module**

This chapter describes the functions of the ROM mirror function selection module and the configuration/function of the register.

#### **Chapter 24 Address match detection function**

This chapter explains the address match detection function and its operation.

#### **Chapter 25 FLASH MEMORY**

This chapter describes the functions and operation of 3M/4M-bit flash memory, and the method for writing/deleting data to the flash memory.

#### **Chapter 26 Example of connecting serial writing**

**(FLASH MICROCONTROLLER PROGRAMMER MODE by YOKOGAWA  
DIGITAL COMPUTER CORPORATION)**

This chapter describes the serial on-board writing of the flash ROM (Fujitsu standard).

#### **Chapter 27 SERIAL PROGRAMMING CONNECTION (FUJITSU MICROELECTRONICSSERIAL PROGRAMMER)**

This chapter explains the basic configuration for serial write to flash memory by using the Fujitsu Microelectronics Serial Programmer.

#### **APPENDIX**

Appendix includes detailed information on I/O map, interrupt vector, and instruction list, which are not mentioned in this manual and information that is needed for programming.

- The contents of this document are subject to change without notice.  
Customers are advised to consult with sales representatives before ordering.
- The information, such as descriptions of function and application circuit examples, in this document are presented solely for the purpose of reference to show examples of operations and uses of FUJITSU MICROELECTRONICS device; FUJITSU MICROELECTRONICS does not warrant proper operation of the device with respect to use based on such information. When you develop equipment incorporating the device based on such information, you must assume any responsibility arising out of such use of the information. FUJITSU MICROELECTRONICS assumes no liability for any damages whatsoever arising out of the use of the information.
- Any information in this document, including descriptions of function and schematic diagrams, shall not be construed as license of the use or exercise of any intellectual property right, such as patent right or copyright, or any other right of FUJITSU MICROELECTRONICS or any third party or does FUJITSU MICROELECTRONICS warrant non-infringement of any third-party's intellectual property right or other right by using such information. FUJITSU MICROELECTRONICS assumes no liability for any infringement of the intellectual property rights or other rights of third parties which would result from the use of information contained herein.
- The products described in this document are designed, developed and manufactured as contemplated for general use, including without limitation, ordinary industrial use, general office use, personal use, and household use, but are not designed, developed and manufactured as contemplated (1) for use accompanying fatal risks or dangers that, unless extremely high safety is secured, could have a serious effect to the public, and could lead directly to death, personal injury, severe physical damage or other loss (i.e., nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system), or (2) for use requiring extremely high reliability (i.e., submersible repeater and artificial satellite).  
Please note that FUJITSU MICROELECTRONICS will not be liable against you and/or any third party for any claims or damages arising in connection with above-mentioned uses of the products.
- Any semiconductor devices have an inherent chance of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions.
- Exportation/release of any products described in this document may require necessary procedures in accordance with the regulations of the Foreign Exchange and Foreign Trade Control Law of Japan and/or US export control laws.
- The company names and brand names herein are the trademarks or registered trademarks of their respective owners.

# CONTENTS

<b>CHAPTER 1</b>	<b>OVERVIEW .....</b>	<b>1</b>
1.1	Feature of MB90330A Series .....	2
1.2	Block Diagram .....	7
1.3	Package Dimension .....	8
1.4	Pin Assignment .....	10
1.5	Pin Function .....	11
1.6	I/O Circuit Types .....	17
1.7	Precautions when Using Devices .....	20
<b>CHAPTER 2</b>	<b>CPU .....</b>	<b>23</b>
2.1	Overview of the CPU .....	24
2.2	Memory Space .....	25
2.3	Linear Addressing .....	28
2.4	Bank Addressing .....	29
2.5	Multibyte Data in Memory Space .....	31
2.6	Registers .....	32
2.6.1	Accumulator (A) .....	35
2.6.2	User Stack Pointer (USP) and System Stack Pointer (SSP) .....	36
2.6.3	Processor Status (PS) .....	37
2.6.4	Program Counter (PC) .....	40
2.6.5	Bank Registers (PCB, DTB, USB, SSB, ADB) .....	41
2.6.6	Direct Page Register (DPR) .....	42
2.7	Register Bank .....	43
2.8	Prefix Codes .....	44
2.9	Interrupt Disable Instructions .....	47
<b>CHAPTER 3</b>	<b>INTERRUPT .....</b>	<b>49</b>
3.1	Outline of Interrupt .....	50
3.2	Interrupt Cause and Interrupt Vector .....	53
3.3	Interrupt Control Register and Peripheral Function .....	56
3.3.1	Interrupt Control Registers (ICR00 to ICR15) .....	58
3.3.2	Interrupt Control Register Functions .....	60
3.4	Hardware Interrupt .....	63
3.4.1	Operation of Hardware Interrupt .....	66
3.4.2	Operation Flow of Hardware Interrupt .....	68
3.4.3	Procedure for Using a Hardware Interrupt .....	69
3.4.4	Multiple Interrupts .....	70
3.4.5	Hardware Interrupt Processing Time .....	72
3.5	Software Interrupt .....	74
3.6	Interrupts by Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	76
3.6.1	Extended Intelligent I/O Service (EI <sup>2</sup> OS) Descriptor (ISD) .....	78
3.6.2	Each Register of Extended Intelligent I/O Service (EI <sup>2</sup> OS) Descriptor (ISD) .....	80
3.6.3	Operation of Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	83

3.6.4	Procedure for use of Extended Intelligent I/O Service (EI <sup>2</sup> OS)	84
3.6.5	Extended Intelligent I/O Service (EI <sup>2</sup> OS) Processing Time	85
3.7	Exception Processing Interrupt	88
3.8	Interruption by $\mu$ DMAC	89
3.8.1	$\mu$ DMAC Function	90
3.8.2	Register of $\mu$ DMAC	91
3.8.2.1	DMA Descriptor Channel Specification Register (DCSR)	92
3.8.2.2	DMA Status Register (DSRH/DSRL)	94
3.8.2.3	DMA Stop Status Register (DSSR)	95
3.8.2.4	DMA Permission Register (DERH/DERL)	96
3.8.3	DMA Descriptor Window Register (DDWR)	97
3.8.3.1	DMA Data Counter (DDCTH/DDCTL)	98
3.8.3.2	DMA I/O Register Address Pointer (DIOAH/DIOAL)	99
3.8.3.3	DMA Control Register (DMACS)	100
3.8.3.4	DMA Buffer Address Pointer (DBAPH/DBAPM/DBAPL)	102
3.8.4	Explanation of Operation of $\mu$ DMAC	103
3.9	Exceptions	105
3.10	Stack Operation of Interrupt Processing	106
3.11	Program Example of Interrupt Processing	108
3.12	Delayed Interrupt Generation Module	112
3.12.1	Operation of Delayed Interrupt Generation Module	113
<b>CHAPTER 4</b>	<b>RESET</b>	<b>115</b>
4.1	Outline of Reset	116
4.2	Reset Factors and Oscillation Stabilization Wait Times	118
4.3	External Reset Pin	120
4.4	Reset Operation	121
4.5	Reset Factor Bit	123
4.6	State of Each Pin at Reset	125
<b>CHAPTER 5</b>	<b>CLOCK</b>	<b>127</b>
5.1	Outline of Clock	128
5.2	Block Diagram of Clock Generation Section	131
5.3	Clock Select Register (CKSCR)	134
5.4	Clock Mode	137
5.5	Oscillation Stabilization Wait Time	140
5.6	Connection of Oscillator and External Clock	141
<b>CHAPTER 6</b>	<b>LOW-POWER CONSUMPTION MODE</b>	<b>143</b>
6.1	Outline of Low-Power Consumption Mode	144
6.2	Block Diagram of Low-power Consumption Control Circuit	147
6.3	Low-power Consumption Mode Control Register (LPMCR)	149
6.4	CPU Intermittent Operation Mode	152
6.5	Standby Mode	153
6.5.1	Sleep Mode	154
6.5.2	Time-base Timer Mode	156
6.5.3	Watch Mode	158

6.5.4	Stop Mode .....	160
6.6	State Transition Diagram .....	162
6.7	State of the Pin during Standby Mode, Hold, and Reset .....	164
6.8	Precautions when Using Low-power Consumption Mode .....	174
<b>CHAPTER 7</b>	<b>MODE SETTING .....</b>	<b>177</b>
7.1	Mode Setting .....	178
7.2	Mode Pins (MD2 to MD0) .....	179
7.3	Mode Data .....	180
7.4	External Memory Access .....	184
7.4.1	Automatic Ready Function Selection Register (ARSR) .....	186
7.4.2	External Address Output Control Register (HACR) .....	187
7.4.3	Bus Control Signal Selection Register (EPCR) .....	188
7.5	Operation in Each Mode of Mode Setting .....	190
7.5.1	External Memory Access Control Signal .....	191
7.5.2	Ready Function .....	194
7.5.3	Holding Function .....	197
<b>CHAPTER 8</b>	<b>I/O PORT .....</b>	<b>199</b>
8.1	Functions of I/O Ports .....	200
8.2	I/O Port Register .....	201
8.2.1	Port Data Register (PDR0 to PDRB) .....	202
8.2.2	Port Direction Register (DDR0 to DDRB) .....	203
8.2.3	Other Registers .....	204
<b>CHAPTER 9</b>	<b>TIME-BASE TIMER .....</b>	<b>207</b>
9.1	Overview of Time-base Timer .....	208
9.2	Configuration of Time-base Timer .....	210
9.3	Time-base Timer Control Register (TBTC) .....	212
9.4	Interrupt of Time-base Timer .....	214
9.5	Operations of Time-base Timer .....	215
9.6	Precautions when Using Time-base Timer .....	217
9.7	Program Example of Time-base Timer .....	219
<b>CHAPTER 10</b>	<b>WATCHDOG TIMER .....</b>	<b>221</b>
10.1	Overview of Watchdog Timer .....	222
10.2	Watchdog Timer Control Register (WDTC) .....	224
10.3	Configuration of Watchdog Timer .....	226
10.4	Operations of Watchdog Timer .....	228
10.5	Precautions when Using Watchdog Timer .....	230
10.6	Program Examples of Watchdog Timer .....	231
<b>CHAPTER 11</b>	<b>WATCH TIMER .....</b>	<b>233</b>
11.1	Overview of Watch Timer .....	234
11.2	Configuration of Watch Timer .....	235
11.3	Watch Timer Control Register (WTC) .....	236
11.4	Operation of Watch Timer .....	238



<b>CHAPTER 12 16-BIT I/O TIMER .....</b>	<b>239</b>
12.1 Overview of 16-bit I/O Timer .....	240
12.2 Register of 16-bit I/O Timer .....	242
12.2.1 16-bit Free-run Timer .....	243
12.2.2 Output Compare .....	249
12.2.3 Input Capture .....	254
12.3 Operation of 16-bit I/O Timer .....	257
12.3.1 Operation of 16-bit Free-run Timer .....	258
12.3.2 Operation of 16-bit Output Compare .....	260
12.3.3 Operation of 16-bit Input Capture .....	262
12.3.4 Timing of 16-bit Free-run Timer .....	263
12.3.5 Output Compare Timing .....	264
12.3.6 Input Timing of Input Capture .....	265
 <b>CHAPTER 13 USB FUNCTION .....</b>	 <b>267</b>
13.1 Overview of USB Function .....	268
13.2 Block Diagram of USB Function .....	269
13.3 Registers of USB Function .....	270
13.3.1 UDC Control Register (UDCC) .....	273
13.3.2 EP0 Control Register (EP0C) .....	276
13.3.3 EP1 to EP5 Control Register (EP1C to EP5C) .....	278
13.3.4 Time Stamp Register (TMSP) .....	282
13.3.5 UDC Status Register (UDCS) .....	283
13.3.6 UDC Interruption Enable Register (UDCIE) .....	286
13.3.7 EP0I Status Register (EP0IS) .....	288
13.3.8 EP0O Status Register (EP0OS) .....	290
13.3.9 EP1 to EP5 Status Register (EP1S to EP5S) .....	293
13.3.10 EP0 to EP5 Data Register (EP0DT to EP5DT) .....	297
13.4 Operation Explanation of USB Function .....	298
13.4.1 Detecting Connection and Disconnection .....	301
13.4.2 Each Register Operation when Command Responds .....	303
13.4.3 STALL Response and Release .....	305
13.4.4 Suspend Function .....	309
13.4.5 Wake-up Function .....	310
13.4.6 DMA Transfer Function .....	311
13.4.7 NULL Transfer Function .....	315
 <b>CHAPTER 14 USB HOST .....</b>	 <b>317</b>
14.1 Feature of USB HOST .....	318
14.2 Restriction on USB HOST .....	319
14.3 Block Diagram of USB HOST .....	320
14.4 Register of USB HOST .....	321
14.4.1 Host Control Register 0,1(HCNT0/HCNT1) .....	324
14.4.2 Host Interruption Register (HIRQ) .....	328
14.4.3 Host Error Status Register (HERR) .....	331
14.4.4 Host State Status Register (HSTATE) .....	334
14.4.5 SOF Interruption FRAME Comparison Register (HFCOMP) .....	337

14.4.6	Retry Timer Setting Register (HRTIMER)	338
14.4.7	Host Address Register (HADR)	339
14.4.8	EOF Setting Register (HEOF)	340
14.4.9	FRAME Setting Register (HFRAME)	341
14.4.10	Host Token Endpoint Register (HTOKEN)	342
14.5	Operation of USB HOST	344
14.5.1	Connection of Device	345
14.5.2	Reset of USB Bus	347
14.5.3	Token Packet	348
14.5.4	Data Packet	350
14.5.5	Handshake Packet	351
14.5.6	Retry Function	352
14.5.7	SOF Interrupt	353
14.5.8	Error Status	355
14.5.9	Packet End	356
14.5.10	Suspend Resume	357
14.5.11	Cutting of Device	360
14.6	Each Token Flow Chart of USB HOST	361
<b>CHAPTER 15</b>	<b>PWC TIMER</b>	<b>363</b>
15.1	Overview of PWC Timer	364
15.2	Register of PWC Timer	366
15.2.1	PWC Control Status Register (PWCSR)	367
15.2.2	PWC Data Buffer Register (PWCR)	372
15.2.3	PWC Ratio of Dividing Frequency Control Register (DIVR)	373
15.3	Movement of PWC Timer	374
15.3.1	Operation of PWM Timer Functions	375
15.3.2	Operation of Pulse Width Measurement Function	376
15.3.3	Count Clock Selection and Operation Mode selection	377
15.3.4	Startup and Stop of Timer/Pulse Width Measurement	379
15.3.5	Operation of Timer Mode	381
15.3.6	Operation of Pulse Width Measurement Mode	384
15.4	Precautions when Using PWC Timer	389
<b>CHAPTER 16</b>	<b>16-BIT RELOAD TIMER</b>	<b>391</b>
16.1	Overview of 16-bit Reload Timer	392
16.1.1	Function of 16-bit Reload Timer	393
16.1.2	Block Diagram of 16-bit Reload Timer	395
16.2	Registers of 16-bit Reload Timer	396
16.2.1	Timer Control Status Register 0 to 2 (TMCSR0 to TMCSR2)	397
16.2.2	16-bit Timer Register 0 to 2 (TMR0 to TMR2)/16-bit Reload Register 0 to 2 (TMRLR0 to TMRLR2)	401
16.3	Movement of 16-bit Reload Timer	403
16.3.1	State Transition of Counter Operation	404
16.3.2	Operation of Internal Clock Mode (Reload Mode)	405
16.3.3	Operation of Internal Clock Mode (Single Shot Mode)	407
16.3.4	Event Count Mode	409

<b>CHAPTER 17 8/16-BIT PPG TIMER .....</b>	<b>411</b>
17.1 Overview of 8/16-bit PPG Timer .....	412
17.1.1 Block Diagram of 8/16-bit PPG Timer .....	413
17.2 Registers of 8/16-bit PPG Timer .....	415
17.2.1 PPG0/PPG2/PPG4 Operation Mode Control Register (PPGC0/PPGC2/PPGC4) .....	416
17.2.2 PPG1/PPG3/PPG5 Operation Mode Control Register (PPGC1/PPGC3/PPGC5) .....	418
17.2.3 PPG0 to PPG5 Output Control Register (PPG01/PPG23/PPG45) .....	421
17.2.4 PPG Reload Registers (PRL0 to PRL5, PRLH0 to PRLH5) .....	423
17.3 Operation of 8/16-bit PPG Timer .....	424
<b>CHAPTER 18 DTP/EXTERNAL INTERRUPT .....</b>	<b>429</b>
18.1 Overview of DTP/External Interrupt .....	430
18.2 Register of DTP/External Interrupt .....	431
18.3 Operation of DTP/External Interrupt .....	434
18.4 Precaution of Using DTP/External Interrupt .....	436
<b>CHAPTER 19 8/10-BIT A/D CONVERTER .....</b>	<b>439</b>
19.1 Overview of 8/10-bit A/D Converter .....	440
19.2 Configuration of 8/10-bit A/D Converter .....	441
19.3 Register of 8/10-bit A/D Converter .....	443
19.3.1 A/D Control Status Register (High) (ADCS1) .....	444
19.3.2 A/D Control Status Register (Low) (ADCS0) .....	446
19.3.3 A/D Conversion Channel Set Register (ADMR) .....	448
19.3.4 A/D Data Register (ADCR1/ADCR0) .....	450
19.4 Explanation of Operation of 8/10-bit A/D Converter .....	452
19.4.1 Conversion Operation Using $\mu$ DMAC or EI <sup>2</sup> OS .....	455
19.4.2 A/D-converted Data Protection Function .....	456
19.5 Precautions when Using 8/10-bit A/D Converter .....	458
19.6 Example of program-1 of 8/10-bit A/D Converter (Example of Starting the EI <sup>2</sup> OS in the Single Mode) .....	459
19.7 Example of Program-2 of 8/10-bit A/D Converter (Example of Starting the EI <sup>2</sup> OS in the Continuous Mode) .....	462
19.8 Example of Program-3 of 8/10-bit A/D Converter (Example of Starting the EI <sup>2</sup> OS in the Stop Mode) .....	465
<b>CHAPTER 20 EXTENDED I/O SERIAL INTERFACE .....</b>	<b>469</b>
20.1 Outline of Extended I/O Serial Interface .....	470
20.2 Register in Extended I/O Serial Interface .....	471
20.2.1 Serial Mode Control Status Register (SMCS) .....	472
20.2.2 Serial Data Register (SDR) .....	476
20.2.3 Communication Prescaler Control Register (SDCR) .....	477
20.3 Operation of Extended I/O Serial Interface .....	478
20.3.1 Shift Clock Mode .....	479
20.3.2 Operation State of Serial I/O .....	480
20.3.3 Start/stop Timing of Shift Operation and Timing of I/O .....	482
20.3.4 Interrupt Function .....	484

<b>CHAPTER 21</b>	<b>UART</b>	<b>485</b>
21.1	Overview of UART	486
21.2	UART Block Diagram	488
21.3	UART Pins	491
21.4	Register of UART	492
21.4.1	Serial Control Register 0 to 3 (SCR0 to SCR3)	493
21.4.2	Serial Mode Register 0 to 3 (SMR0 to SMR3)	495
21.4.3	Serial Status Register 0 to 3 (SSR0 to SSR3)	497
21.4.4	Serial Input Data Register 0 to 3 (SIDR0 to SIDR3) and Serial Output Data Register 0 to 3 (SODR0 to SODR3)	500
21.4.5	UART Prescaler Control Register 0 to 3 (UTCR0 to UTCR3) and UART Prescaler Reload Register 0 to 3 (UTRLR0 to UTRLR3)	502
21.5	UART Interrupt	504
21.5.1	Receive Interrupt Generation and Flag Set Timing	506
21.5.2	Transmit Interrupt Generation and Flag Set Timing	508
21.6	UART Baud Rate	510
21.6.1	Baud Rate of the UART Internal Clock Using the Dedicated Baud Rate Generator	511
21.6.2	Baud Rate of the External Clock Using the Dedicated Baud Rate Generator	512
21.6.3	Baud Rate of the External Clock (One-to-one Mode)	513
21.7	Explanation of Operation of UART	514
21.7.1	Operation in Asynchronous Mode (Operation Mode 0 or Operation Mode 1)	516
21.7.2	Operation in Synchronous Mode (Operation Mode 2)	519
21.7.3	Bi-directional Communication Function (Normal Mode)	522
21.7.4	Master/Slave Mode Communication Function (Multi-processor Mode)	524
21.8	Notes on Using UART	527
21.9	Example of UART Programming	528
<b>CHAPTER 22</b>	<b>I<sup>2</sup>C INTERFACE</b>	<b>531</b>
22.1	I <sup>2</sup> C Interface Outline	532
22.2	I <sup>2</sup> C Interface Register	534
22.2.1	I <sup>2</sup> C Bus Status Register 0 to 2 (IBSR0 to IBSR2)	535
22.2.2	I <sup>2</sup> C Bus Control Register 0 to 2 (IBCR0 to IBCR2)	537
22.2.3	I <sup>2</sup> C Bus Clock Control Register 0 to 2 (ICCR0 to ICCR2)	542
22.2.4	I <sup>2</sup> C Bus Address Register 0 to 2 (IADR0 to IADR2)	544
22.2.5	I <sup>2</sup> C Bus Data Register 0 to 2 (IDAR0 to IDAR2)	545
22.3	I <sup>2</sup> C Interface Operation	546
22.3.1	Transfer Flow of I <sup>2</sup> C Interface	548
22.3.2	Mode Flow of I <sup>2</sup> C Interface	550
22.3.3	Operation Flow of I <sup>2</sup> C Interface	551
<b>CHAPTER 23</b>	<b>ROM MIRROR FUNCTION SELECTION MODULE</b>	<b>553</b>
23.1	Overview of ROM Mirror Function Select Module	554
23.2	ROM Mirror Function Select Register (ROMM)	555
<b>CHAPTER 24</b>	<b>ADDRESS MATCH DETECTION FUNCTION</b>	<b>557</b>
24.1	Overview of Address Match Detection Function	558
24.2	Block Diagram of Address Match Detection Function	559

24.3	Configuration of Address Match Detection Function .....	560
24.3.1	Address Detection Control Register (PACSR) .....	561
24.3.2	Detect Address Setting Registers (PADR0, PADR1) .....	563
24.4	Explanation of Operation of Address Match Detection Function .....	565
24.4.1	Example of Using Address Match Detection Function .....	566
24.5	Program Example of Address Match Detection Function .....	571
<b>CHAPTER 25</b>	<b>FLASH MEMORY .....</b>	<b>573</b>
25.1	Overview of Flash Memory .....	574
25.2	Sector Configuration of Flash Memory .....	575
25.3	Flash Memory Control Status Register (FMCS) .....	577
25.4	Automatic Algorithm Initiation Method of Flash Memory .....	579
25.5	Check the Execution State of Automatic Algorithm .....	580
25.5.1	Data Polling Flag (DQ7) .....	582
25.5.2	Toggle Bit Flag (DQ6) .....	584
25.5.3	Timing Limit Over Flag (DQ5) .....	585
25.5.4	Sector Erasing Timer Flag (DQ3) .....	586
25.6	Write/Erase of Flash memory .....	587
25.6.1	Read/Reset State in Flash Memory .....	588
25.6.2	Writing Data to Flash Memory .....	589
25.6.3	Erasing All Data from Flash Memory (Chip Erase) .....	591
25.6.4	Erasing Any Data in Flash Memory (Sector Erasing) .....	592
25.6.5	Flash Memory Sector Erase Suspension .....	594
25.6.6	Flash Memory Sector Erase Resumption .....	595
<b>CHAPTER 26</b>	<b>EXAMPLE of CONNECTING SERIAL WRITING (FLASH MICROCONTROLLER PROGRAMMER MODE by YOKOGAWA DIGITAL COMPUTER CORPORATION) .....</b>	<b>597</b>
26.1	Basic Configuration .....	598
26.2	Oscillation Clock Frequency and Serial Clock Input Frequency .....	600
26.3	Flash Microcontroller Programmer System Configuration .....	601
26.4	Example of Connecting Serial Writing .....	602
26.4.1	Example Connection in Single-chip Mode (when Using User Power) .....	603
26.4.2	Example of Minimum Connection to Flash Microcontroller Programmer (when Using User Power) .....	605
<b>CHAPTER 27</b>	<b>SERIAL PROGRAMMING CONNECTION (FUJITSU MICROELECTRONICS SERIAL PROGRAMMER) .....</b>	<b>607</b>
27.1	Fujitsu Microelectronics Serial Programmer .....	608
27.1.1	Pins Used .....	612
<b>APPENDIX</b> .....		<b>613</b>
APPENDIX A	Memory Map .....	614
APPENDIX B	Instructions .....	630
B.1	Instruction Types .....	631
B.2	Addressing .....	632
B.3	Direct Addressing .....	634

B.4	Indirect Addressing .....	640
B.5	Execution Cycle Count .....	648
B.6	Effective address field .....	651
B.7	How to Read the Instruction List .....	652
B.8	F <sup>2</sup> MC-16LX Instruction List .....	655
B.9	Instruction Map .....	669
APPENDIX B .....		690

<b>INDEX.....</b>	<b>691</b>
-------------------	------------



# Main changes in this edition

Page	Changes (For details, refer to main body.)	
-	-	Changed USB. USB Mini-Host → USB HOST
2, 5	CHAPTER 1 OVERVIEW 1.1 Feature of MB90330A Series	Corrected the minimum instruction execution time. 41.6 ns → 41.7 ns
22	CHAPTER 1 OVERVIEW 1.7 Precautions when Using Devices	Added "● Serial communication".
24	CHAPTER 2 CPU 2.1 Overview of the CPU ■ Overview of the CPU	Corrected the "● Minimum execution time". <ul style="list-style-type: none"> <li>• 41.7 ns / 6 MHz original oscillation four-time multiplication (at 24 MHz/3.3 V ± 0.3 V internal operation)</li> <li>• PLL clock multiplication method</li> </ul> → <ul style="list-style-type: none"> <li>• 41.7 ns (at machine clock 24MHz)</li> <li>• The frequency of the machine clock varies depending on the model.</li> </ul>
		Changed the "● CPU-independent automatic transfer function". <ul style="list-style-type: none"> <li>• Extended intelligent I/O service function from maximum 16 channels</li> <li>• <math>\mu</math>DMAC from maximum 16 channels</li> </ul> → <ul style="list-style-type: none"> <li>• Up to 16 channels of the extended intelligent I/O service (EI<sup>2</sup>OS)</li> <li>• Up to 16 channels of the DMA transfer (<math>\mu</math>DMAC)</li> <li>• The DMA transfer (<math>\mu</math>DMAC) might not be built-in by the series.)</li> </ul>
39	CHAPTER 2 CPU 2.6 Registers	Corrected "Acceptable interrupt level" in Table 2.6-1.
41		Corrected title and content of section 2.6.5.
44	CHAPTER 2 CPU 2.8 Prefix Codes	Corrected "Bank select prefix" of PC space in Table 2.8-1. PCC → PCB
91	CHAPTER 3 INTERRUPT 3.8 Interruption by $\mu$ DMAC ■ $\mu$ DMAC Register List	Changed the Figure 3.8-1 $\mu$ DMA Register List. bit8 to bit15 → STP15 to STP8 bit0 to bit7 → STP7 to STP0
95	CHAPTER 3 INTERRUPT 3.8 Interruption by $\mu$ DMAC ■ DMA Stop Status Register (DSSR)	Changed the Notes. Bit8 to Bit15 → STP15 to STP8 Bit0 to Bit7 → STP7 to STP0
203	CHAPTER 8 I/O PORT 8.2.2 Port Direction Register (DDR0 to DDRB)	Corrected the explanation of "● When each terminal functions as a port".
319	CHAPTER 14 USB HOST 14.2 Restriction on USB HOST	Changed the description. Diversity with USB HOST → Restriction on USB HOST



Page	Changes (For details, refer to main body.)	
421	CHAPTER 17 8/16-BIT PPG TIMER 17.2 Registers of 8/16-bit PPG Timer	Corrected the operation mode column in the table "[bit 7 to bit 5] PCS2 to PCS0: ppg Count Select (count clock selection)". 41.6 ns → 41.7 ns
422	■ PPG0 to PPG5 Output Control Register (PPG01/PPG23/PPG45)	Corrected the operation mode column in the table "[bit 4 to bit 2] PCM2 to PCM0:ppg Count Mode (count clock selection)". 41.6 ns → 41.7 ns
580	CHAPTER 25 FLASH MEMORY	Deleted "Toggle Bit 2 Flag (DQ2)".
581	25.5 Check the Execution State of Automatic Algorithm	Deleted "Toggle Bit 2 Flag (DQ2)" in Table 25.5-2. Deleted "*" sentence under Table 25.5-2.
586		Deleted "25.5.5 Toggle Bit 2 Flag (DQ2)".
593	CHAPTER 25 FLASH MEMORY 25.6.4 Erasing Any Data in Flash Memory (Sector Erasing)	Corrected flowchart of figure 25.6-2.
636	B.3 Direct Addressing ● I/O direct addressing (io)	Changed Figure B.3-5. (MOVW A, i : 0C0H → MOVW A, I:0C0H)
		Added the note to Figure B.3-5.
637	B.3 Direct Addressing ● Abbreviated direct addressing (dir)	Added the note to Figure B.3-6.
638	B.3 Direct Addressing ● I/O direct bit addressing (io:bp)	Changed Figure B.3-8. (SETB i : 0C1H : 0 → SETB I:0C1H:0)
		Added the note to Figure B.3-8.
	B.3 Direct Addressing ● Abbreviated direct bit addressing (dir:bp)	Added the note to Figure B.3-9.
644	B.4 Indirect Addressing ● Program counter relative branch addressing (rel)	Changed Figure B.4-7. (BRA 10H → BRA 3C32H)
645	B.4 Indirect Addressing ● Register list (rlst)	Changed Figure B.4-9. (POPW, RW0, RW4 → POPW RW0, RW4)

Page	Changes (For details, refer to main body.)	
670	B.9 Instruction Map ■ Structure of Instruction Map	Changed column: instruction in Table B.9-1. (@RW2+d8, #8, rel → CBNE @RW2+d8, #8, rel)
671		Changed the operand at row: +0, column: E0 in Table B.9-2. (#4 → #vct4)
		Changed the mnemonic at row: +0, column: D0 in Table B.9-2. (MOV → MOVN)
		Changed the mnemonic at row: +0, column: B0 in Table B.9-2. (MOV → MOVX)
		Changed the mnemonic at row: +8, column: B0 in Table B.9-2. (MOV → MOVW)
		673
674		Changed Table B.9-5. · Moved "MUL A" and "MULW A" instruction from column:60 to column:70. · Changed mnemonic and moved the Instruction from column:60, row:+A to column:70, row:+A. (DIVU → DIV))
675		Changed the operand at row: +E and +F, column: F0 in Table B.9-6. (, #8, rel → #8, rel)
678		Changed the operand at row: +8 to +E, column: 50 in Table B.9-9. (@@ → @)
		Changed the operand at row: +0 to +7, column: 20 in Table B.9-9. (RWi → @RWi)
679		Changed the operand at column: E0 and F0 in Table B.9-10. (,r → ,rel)
680		Changed the operand at column: 70 in Table B.9-11. (NEG A, → NEG)
681		Changed the operand at column: E0 and F0 in Table B.9-12. (,r → ,rel)
689		Changed Table B.9-20 XCH Ri, ea Instruction (First Byte = 7EH). (Column "A" → Column "A0", Changed row "+A" ( W2+d16,A → @RW2+d16 ))

The vertical lines marked in the left side of the page show the changes.



# **CHAPTER 1**

---

# **OVERVIEW**

**This chapter describes basics to give the understanding of the MB90330A series as a whole such as the features, block diagrams, and overviews of the functions.**

- 1.1 Feature of MB90330A Series
- 1.2 Block Diagram
- 1.3 Package Dimension
- 1.4 Pin Assignment
- 1.5 Pin Function
- 1.6 I/O Circuit Types
- 1.7 Precautions when Using Devices

## 1.1 Feature of MB90330A Series

---

The MB90330A series are 16-bit microcontrollers designed for applications, such as personal computer peripheral devices, that require USB communications. The USB function enables not only 12-Mbps function operations but also simplified host operations. It is equipped with functions that are suitable for personal computer peripheral devices such as displays and audio devices, and control of mobile devices that support USB communications.

---

### ■ Feature of MB90330A Series

In the MB90330A series, there are the following features.

- Built-in PLL clock multiplying circuit
  - When the original oscillation is 6 MHz. Operating clock (PLL clock) of 3-24 MHz can be selected from: divided-by-two of the original oscillation or 1-, 2-, or 4-times multiplication of the original oscillation. A clock for USB is 48 MHz.
  - Minimum instruction execution time of 41.7 ns (at oscillation of 6 MHz, four multiplied PLL clock, operation at Vcc of 3.3 V)
- Maximum memory space: 16 Mbytes
- Instruction system optimized to control usage
  - Data type which can be handled: bit/byte/word/long word
  - Standard addressing mode: 23 types
  - High-precision operation enhanced by the employed 32-bit accumulator
  - Signed multiplication and division, and enhanced RETI instructions
- Instruction system that supports high-level language (C language) multitasking
  - Adoption of system stack point
  - Instruction set symmetry and barrel shift instructions
- For no multi-bus/multi-bus
- Higher execution speed: 4-byte queue
- Powerful interrupt function (priority is programmable and can be set to eight levels): 8 external interrupts
- Data transfer function
  - $\mu$ DMAC: maximum 16 channels
  - Extended intelligent I/O service function: maximum 16 channels

- Capacity of built-in ROM and ROM type
  - Mask ROM: 256 Kbytes, 384 Kbytes
  - Flash ROM: 384 Kbytes
- Built-in RAM
  - Mass production products: 16 Kbytes, 24 Kbytes
  - Flash products: 24 Kbytes, 30 Kbytes
  - Evaluation chip: 28 Kbytes
- Process: CMOS technology
- Low-power consumption (standby) mode
  - Sleep mode (mode by which the CPU operation clock is stopped)
  - Stop mode (mode by which original oscillation is stopped)
  - CPU intermittent operation mode
- Package
  - LQFP-120 (FPT-120P-M24: 0.4 mm pin pitch)
  - LQFP-120 (FPT-120P-M21: 0.5 mm pin pitch)
- Operation guaranteed temperature:
  - 40 °C to +85 °C (0 °C to +70 °C when USB is in use)
- General-purpose: maximum 94 ports
  - General-purpose I/O (CMOS): 56 ports
  - General-purpose I/O ports (input pull-up resistor settable): 16 ports
  - General-purpose I/O ports (output open drain/5 V tolerant I/O ports): 22 ports
- Timer: Time-base timer/watchdog timer/clock timer: One channel
- 16-bit I/O timer
  - 16-bit free-run timer: 1 channel
  - Input capture (ICU): 4 channels
  - Output compare (OCU): 4 channels
- 8/16-bit PPG timer: 8 bits x 16 channels or 16 bits x 3 channels
- 16-bit reload timer: 3 channels
- 16 bit PWC timer: 1 channel
- UART: 4 channels

- Extended I/O serial interface: 1 channel
- I<sup>2</sup>C interface: 3 channels
- 8/10-bit A/D converter (RC sequential comparator type):16 channels
- DTP/external interrupt: 8 channels
- USB
  - USB function (Correspond to USB Full Speed):1 channel
  - USB HOST:1 channel

## ■ Product Lineup

Table 1.1-1 MB90330A Series Product Lineup List (1 / 2)

Product name		MB90V330A *	MB90F334A	MB90F335A	MB90333A
Classification		Evaluation product	Flash memory product	Flash memory product	MASK ROM products
ROM size		None	384 Kbytes	512Kbytes	256 Kbytes
RAM size		28 Kbytes	24 Kbytes	30 Kbytes	16 Kbytes
Power supply for emulator		Provided	-	-	-
CPU function		Number of basic instructions: 351 Instruction bit length: 8 bits, 16 bits Minimum instruction execution time: 41.7 ns/24 MHz Addressing type: 23 types Maximum size of memory space: 16 Mbytes			
Port		Input/output port (CMOS) : 94			
16 bit Input/Output Timers	16-bit free-run timer	Channel count: 1 Overflow interrupt			
	Output Comparison (OCU)	Channel count: 4 Pin input factor: matching signal of the compare register			
	Input Capture (ICU)	Channel count: 4 Rewriting a register upon a pin input (rising edge, falling edge, or both edges)			
8/16-bit PPG timer		Number of channels: 8 bits x 6 channels, 16 bits x 3 channels with mode switching function PPG operations of byte or 16 bits Pulse waveform output at arbitrary cycle and duty			
16-bit reload timer		Channel count: 3 16-bit reload timer operation With Event Counter			
16-bit PWC timer		Channel count: 1 Timer function (selects one clock for a counter from three internal clocks) Pulse width measurement function (selects one clock for a counter from three internal clocks)			
UART		Channel count: 4 Clock synchronous/asynchronous selectable Dedicated baud rate generator Clock synchronizer LSB and MSB can be switched.			
I/O Extended serial interface		Channel count: 1 Clock synchronous transfer LSB first/MSB first			
I <sup>2</sup> C bus communication		Channel count: 3 Serial I/O by which Inter IC BUS is supported			
8/10-bit A/D converter		10-bit resolution analog input 16 channels			



**Table 1.1-1 MB90330A Series Product Lineup List (2 / 2)**

Product name	MB90V330A *	MB90F334A	MB90F335A	MB90333A
DTP/external interrupt	Input count: 8 Interrupt factor: rising edge/falling edge/"L" level/"H" level selectable			
USB	USB Function (Corresponding to USB Full Speed) Supports Full speed Endpoint are specifiable up to six. Transfer type: Control, Interrupt, Bulk, or Isochronous transfer possible Dual port RAM (The FIFO mode is supported). USB HOST Functions			
μDMAC	Corresponded			
External bus interface	It is (multi/no multi correspondence).			
The others	22 I/O pins with 5 V tolerant (including pins also used for I <sup>2</sup> C)			
Package	PGA299	LQFP120		
Operating voltage	3.3 V ± 0.3 V			

\*: It is setting of Jumper switch (TOOL VCC) when Emulator (MB2147-01) is used. Please refer to the MB2147-01 or MB2147-20 hardware manual (3.3 Emulator-dedicated Power Supply Switching) about details.

**Note:**

Writing to the Flash, be sure to perform  $V_{CC}=3.13\text{ V}$  to  $3.60\text{ V}$  more than.

## 1.2 Block Diagram

## ■ Block Diagram of the MB90330A Series

Block diagram of the F2MC-16LX core and peripheral components. The central F2MC-16LX BUS connects various modules.

**Left Side Components:**

- X0, X1, X0A, X1A, RST
- Clock control circuit
- CPU F<sup>2</sup>MC-16LX core
- RAM(28Kbyte)\*
- ROM(384Kbyte)\*
- Interrupt controller
- Port B
- 8/16-bit PPG timer (ch.2)
- I<sup>2</sup>C interface (ch.1,2)
- Port A
- 16-bit input capture (ch.0,1,2,3)
- 16-bit output compare (ch.0,1,2,3)
- 16-bit free-run timer
- UART(ch.2,3)
- Port 9
- Port 8
- A/D converter (16ch)
- Port 7
- USB HOST
- USB function
- uDMAC
- External bus interface

**Right Side Components:**

- Port 6
- External interrupt (ch.0 to 7)
- 16-bit PWC timer
- Extended I/O serial interface
- I<sup>2</sup>C interface (ch.0)
- Port 5
- Port 4
- UART(ch.0,1)
- 16-bit reload timer (ch.0)
- Port 3
- 16-bit reload timer (ch.1,2)
- Port 2
- 8/16-bit PPG (ch.0,1)
- Port 1
- Port 0
- Other pins

**Pin Lists (Right Side):**

- P67/INT7/SDA0, P66/INT6/SCL0, P65/INT5/PWC, P64/INT4/SCK, P63/INT3/SOT, P62/INT2/SIN, P61/INT1, P60/INT0
- P57/CLK, P56/RDY, P55/HAK, P54/HRO, P53/WRH, P52/WRL, P51/RD, P50/ALE
- P47/A15/SCK1, P46/A14/SOT1, P45/A13/SIN1, P44/A12/SCK0, P43/A11/SOT0, P42/A10/SIN0, P41/A09/TOT0, P40/A08/TIN0
- P37/A07, P36/A06, P35/A05, P34/A04, P33/A03/TOT2, P32/A02/TIN2, P31/A01/TOT1, P30/A00/TIN1
- P27/A23/PPG3, P26/A22/PPG2, P25/A21/PPG1, P24/A20/PPG0, P23/A19, P22/A18, P21/A17, P20/A16, P17 to P10/, AD15 to AD08/, D15 to D08, P07 to P00/, AD07 to AD00/, D07 to D00

**Power Pins (Bottom):** Vss, Vcc, MD0, MD1, MD2

\*: Maximum value

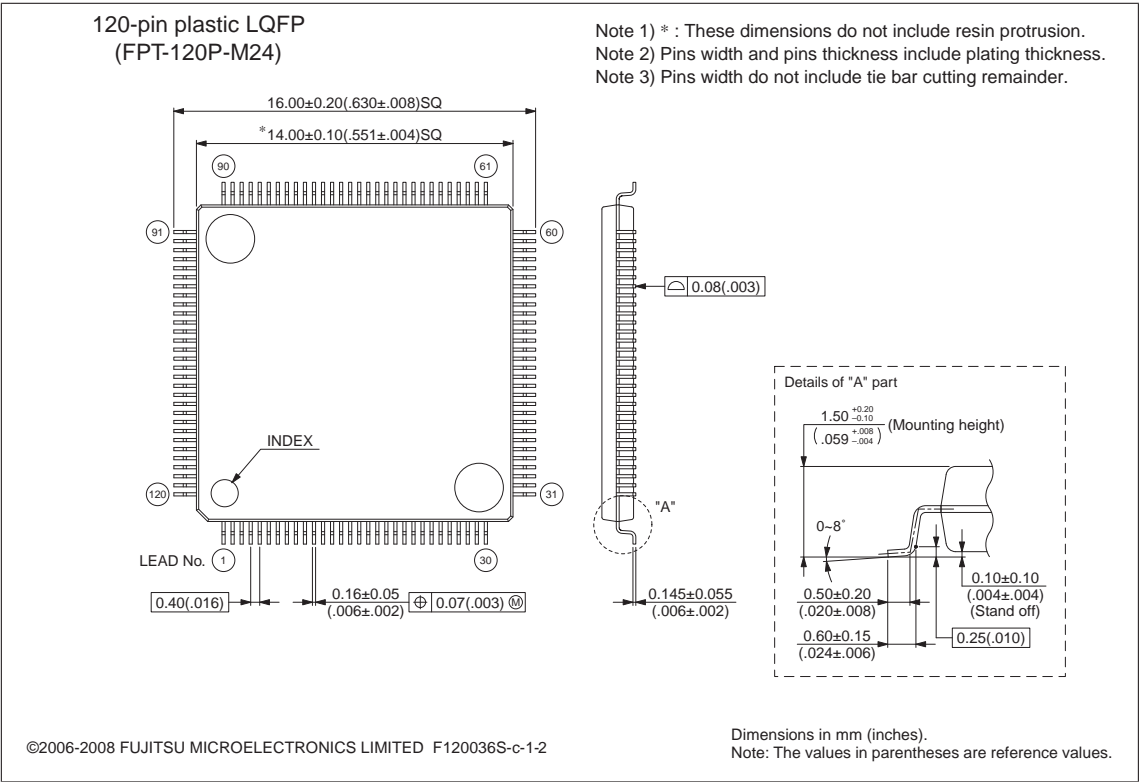
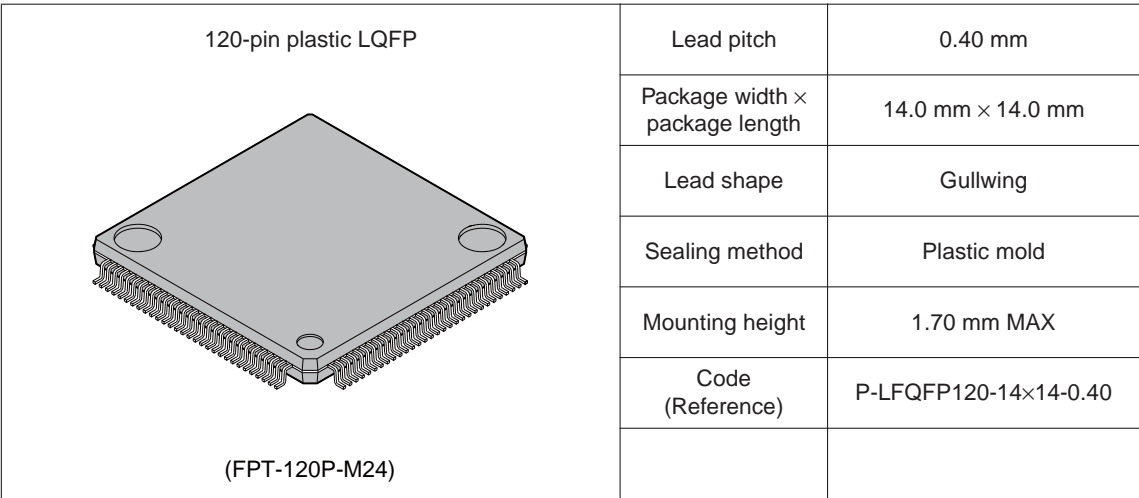
In Figure 1.2-1, I/O ports share pins with each of built-in functional blocks. Any port used for built-in module pin cannot be used as an I/O port.

1.3 Package Dimension

MB90330A series is available in two types of package.

■ Package Dimension (LQFP-120)

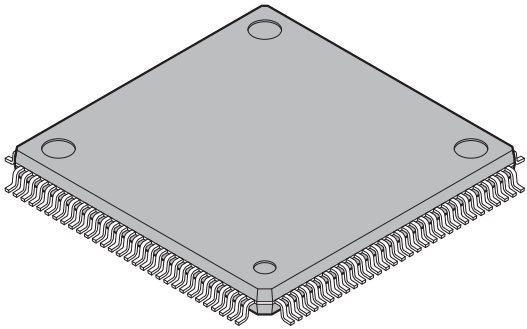
Figure 1.3-1 Package Dimension of LQFP-120 Type

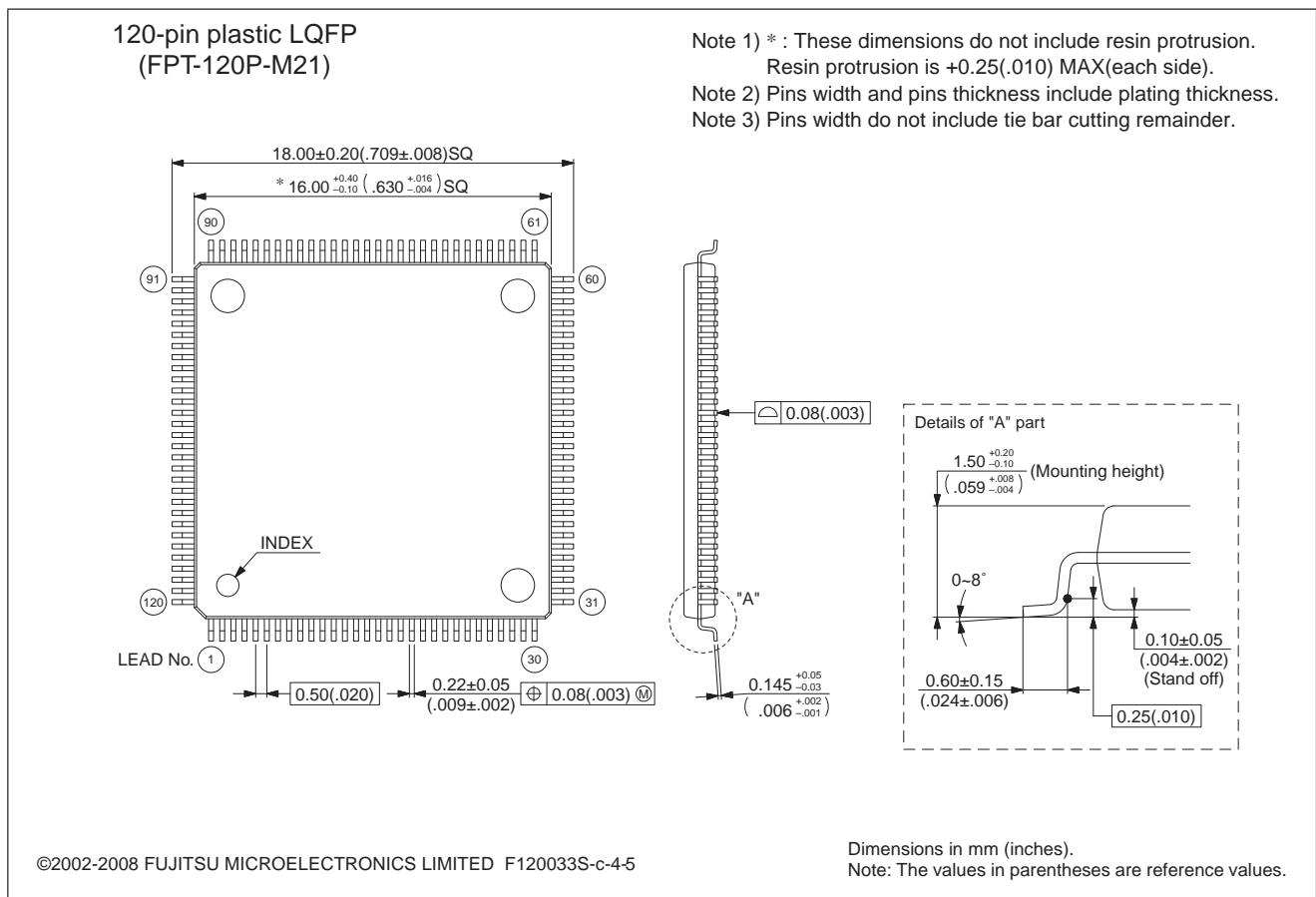


Please check the latest package dimension at the following URL.  
<http://edevic.fujitsu.com/package/en-search/>

#### ■Package Dimension (LQFP-120)

Figure 1.3-2 Package Dimension of LQFP-120 Type

 <p>120-pin plastic LQFP</p> <p>(FPT-120P-M21)</p>	Lead pitch	0.50 mm
	Package width × package length	16.0 × 16.0 mm
	Lead shape	Gullwing
	Sealing method	Plastic mold
	Mounting height	1.70 mm MAX
	Weight	0.88 g
	Code (Reference)	P-LFQFP120-16×16-0.50



Please check the latest package dimension at the following URL.

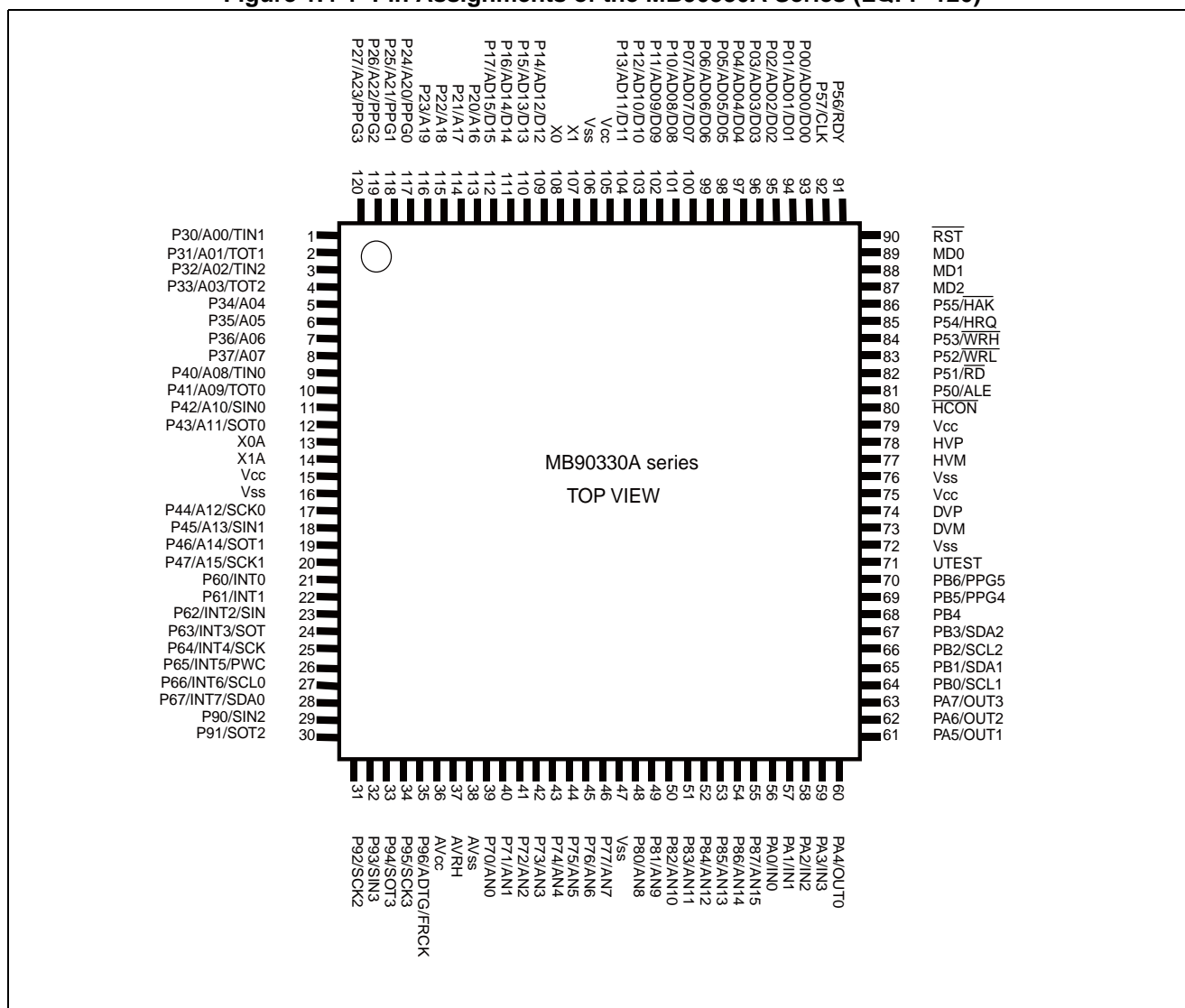
<http://edevic.fujitsu.com/package/en-search/>

## 1.4 Pin Assignment

Figure 1.4-1 shows the pin assignments of a MB90330A series.

### ■ Pin Assignment (LQFP-120)

Figure 1.4-1 Pin Assignments of the MB90330A Series (LQFP-120)



**MB90330A Series****1.5 Pin Function**

**Table 1.5-1 describes the MB90330A series pin functions.**

■ **Pin Function**

**Table 1.5-1 Pin Function (1 / 6)**

Pin No.	Pin Name	Circuit Type	Functional description
107	X1	A	It is oscillation pin.
108	X0	A	It is oscillation pin.
13	X0A	A	It is 32 kHz oscillation pin.
14	X1A	A	It is 32 kHz oscillation pin.
90	$\overline{\text{RST}}$	F	It is reset input.
93 to 100	P00 to P07	H	It is General-purpose I/O port. You can set a pull-up resistor ON (RD00 to RD07= 1) with the pull-up resistor setting register (RDR0) (When the power output is set, it is invalid).
	AD00 to AD07		Functions as an I/O pin for the low-order external address/data bus in multiplex mode.
	D00 to D07		Functions as an I/O pin for the low-order external data bus in non-multiplex mode.
101 to 104	P10 to P13	H	It is General-purpose I/O port. You can set a pull-up resistor ON (RD10 to RD13= 1) with the pull-up resistor setting register (RDR1) (When the power output is set, it is invalid).
	AD08 to AD11		Functions as an I/O pin for the high-order external address/data bus in multiplex mode.
	D08 to D11		Functions as an I/O pin for the high-order external data bus in non-multiplex mode.
109 to 112	P14 to P17	H	It is General-purpose I/O port. You can set a pull-up resistor ON (RD14 to RD17= 1) with the pull-up resistor setting register (RDR1) (When the power output is set, it is invalid)
	AD12 to AD15		Functions as an I/O pin for the high-order external address/data bus in multiplex mode.
	D12 to D15		Functions as an I/O pin for the high-order external data bus in non-multiplex mode.

**Table 1.5-1 Pin Function (2 / 6)**

Pin No.	Pin Name	Circuit Type	Functional description
113 to 116	P20 to P23	D	It is General-purpose I/O port. Functions as the general-purpose input/output port in the external bus mode if the bit corresponding to external address output control register (HACR) is set to "1".
	A16 to A19		Functions as the upper output pin of an address (A16 to A19) in the multiplex mode if the bit corresponding to external address output control register (HACR) is set to "0".
	A16 to A19		Functions as the upper output pin of an address (A16 to A19) in the non-multiplex mode if the bit corresponding to external address output control register (HACR) is set to "0".
117 to 120	P24 to P27	D	It is General-purpose I/O port. Functions as the general-purpose input/output port in the external bus mode if the bit corresponding to external address output control register (HACR) is set to "1".
	A20 to A23		Functions as the upper output pin of an address (A20 to A23) in the multiplex mode if the bit corresponding to external address output control register (HACR) is set to "0".
	A20 to A23		Functions as the upper output pin of an address (A20 to A23) in the non-multiplex mode if the bit corresponding to external address output control register (HACR) is set to "0".
	PPG0 to PPG3		Function as ch.0 to ch.3 output pins for the PPG timer.
1	P30	D	It is General-purpose I/O port.
	A00		Functions as the external address pin in non-multi-bus mode.
	TIN1		Functions as an event input pin for 16-bit reload timer ch.1.
2	P31	D	It is General-purpose I/O port.
	A01		Functions as the external address pin in non-multi-bus mode.
	TOT1		Functions as an output pin for 16-bit reload timer ch.1.
3	P32	D	It is General-purpose I/O port.
	A02		Functions as the external address pin in non-multi-bus mode.
	TIN2		Functions as an event input pin for 16-bit reload timer ch.2.
4	P33	D	It is General-purpose I/O port.
	A03		Functions as the external address pin in non-multi-bus mode.
	TOT2		Functions as an output pin for 16-bit reload timer ch.2.
5 to 8	P34 to P37	D	It is General-purpose I/O port.
	A04 to A07		Functions as the external address pin in non-multi-bus mode.

Table 1.5-1 Pin Function (3 / 6)

Pin No.	Pin Name	Circuit Type	Functional description
9	P40	G	It is General-purpose I/O port.
	A08		Functions as the external address pin in non-multi-bus mode.
	TIN0		Functions as an event input pin for 16-bit reload timer ch.0.
10	P41	G	It is General-purpose I/O port.
	A09		Functions as the external address pin in non-multi-bus mode.
	TOT0		Functions as an output pin for 16-bit reload timer ch.0.
11	P42	G	It is General-purpose I/O port.
	A10		Functions as the external address pin in non-multi-bus mode.
	SIN0		Functions as a data input pin for UART ch.0.
12	P43	G	It is General-purpose I/O port.
	A11		Functions as the external address pin in non-multi-bus mode.
	SOT0		Functions as a data output pin for UART ch.0.
17	P44	G	It is General-purpose I/O port.
	A12		Functions as the external address pin in non-multi-bus mode.
	SCK0		Functions as a clock I/O pin for UART ch.0.
18	P45	G	It is General-purpose I/O port.
	A13		Functions as the external address pin in non-multi-bus mode.
	SIN1		Functions as a data input pin for UART ch.1.
19	P46	G	It is General-purpose I/O port.
	A14		Functions as the external address pin in non-multi-bus mode.
	SOT1		Functions as a data output pin for UART ch.1.
20	P47	G	It is General-purpose I/O port.
	A15		Functions as the external address pin in non-multi-bus mode.
	SCK1		Functions as a clock I/O pin for UART ch.1.
81	P50	L	It is General-purpose I/O port.
	ALE		Functions as the address latch enable signal (ALE) pin in external bus mode.
82	P51	L	It is General-purpose I/O port.
	$\overline{RD}$		Functions as the read strobe output (RDX) pin in external bus mode.
83	P52	L	It is General-purpose I/O port.
	$\overline{WRL}$		Functions as the data write strobe output (WRLX) pin on the lower side in external bus mode. Functions as a general-purpose I/O port when the WRE bit in the EPCR register is "0".



Table 1.5-1 Pin Function (4 / 6)

Pin No.	Pin Name	Circuit Type	Functional description
84	P53	L	It is General-purpose I/O port.
	$\overline{\text{WRH}}$		Functions as the data write strobe output (WRHX) pin on the higher side in 16-bit external bus mode. Functions as a general-purpose I/O port when the WRE bit in the EPCR register is "0".
85	P54	L	It is General-purpose I/O port.
	HRQ		Functions as the hold request input (HRQ) pin in external bus mode. Functions as a general-purpose I/O port when the HDE bit in the EPCR register is "0".
86	P55	L	It is General-purpose I/O port.
	$\overline{\text{HAK}}$		Functions as the hold acknowledge output (HAKX) pin in external bus mode. Functions as a general-purpose I/O port when the HDE bit in the EPCR register is "0".
91	P56	L	It is General-purpose I/O port.
	RDY		Functions as the external ready input (RDY) pin in external bus mode. Functions as a general-purpose I/O port when the RYE bit in the EPCR register is "0".
92	P57	L	It is General-purpose I/O port.
	CLK		Functions as the machine cycle clock output (CLK) pin in external bus mode. Functions as a general-purpose I/O port when the CKE bit in the EPCR register is "0".
21, 22	P60, P61	C	It is General-purpose I/O port (Withstand voltage of 5 V).
	INT0, INT1		Function as input pins for external interrupt ch.0, ch.1.
23	P62	C	It is General-purpose I/O port (Withstand voltage of 5 V).
	INT2		Function as input pins for external interrupt ch.2.
	SIN		It is extended serial I/O data output pin.
24	P63	C	It is General-purpose I/O port (Withstand voltage of 5 V).
	INT3		Function as input pins for external interrupt ch.3.
	SOT		It is simple serial I/O data output pin.
25	P64	C	It is General-purpose I/O port (Withstand voltage of 5 V).
	INT4		Function as input pins for external interrupt ch.4.
	SCK		It is extended serial I/O clock input/output pin.
26	P65	C	It is General-purpose I/O port (Withstand voltage of 5 V).
	INT5		Function as input pins for external interrupt ch.5.
	PWC		Functions as the PWC input pin.

Table 1.5-1 Pin Function (5 / 6)

Pin No.	Pin Name	Circuit Type	Functional description
27	P66	C	It is General-purpose I/O port (Withstand voltage of 5 V).
	INT6		Function as input pins for external interrupt ch.6.
	SCL0		Functions as the clock I/O pin for the I <sup>2</sup> C interface ch.0. Set port output to Hi-Z during I <sup>2</sup> C interface operations.
28	P67	C	It is General-purpose I/O port (Withstand voltage of 5 V).
	INT7		Function as input pins for external interrupt ch.7.
	SDA0		Functions as the data I/O pin for the I <sup>2</sup> C interface ch.0. Set port output to Hi-Z during I <sup>2</sup> C interface operations.
39 to 46	P70 to P77	I	It is General-purpose I/O port.
	AN0 to AN7		Function as input pins for analog ch.0 to ch.7.
48 to 55	P80 to P87	I	It is General-purpose I/O port.
	AN8 to AN15		Function as input pins for analog ch.8 to ch.15.
29	P90	D	It is General-purpose I/O port.
	SIN2		Functions as a data input pin for UART ch.2.
30	P91	D	It is General-purpose I/O port.
	SOT2		Functions as a data output pin for UART ch.2.
31	P92	D	It is General-purpose I/O port.
	SCK2		Functions as a clock I/O pin for UART ch.2.
32	P93	D	It is General-purpose I/O port.
	SIN3		Functions as a data input pin for UART ch.3.
33	P94	D	It is General-purpose I/O port.
	SOT3		Functions as a data output pin for UART ch.3.
34	P95	D	It is General-purpose I/O port.
	SCK3		Functions as a clock I/O pin for UART ch.3.
35	P96	C	It is General-purpose I/O port (Withstand voltage of 5 V).
	ADTG		Functions as the external trigger input pin when the A/D converter is being used.
	FRCK		Functions as the external clock input pin when the free-run timer is being used.
56 to 59	PA0 to PA3	C	It is General-purpose I/O port (Withstand voltage of 5 V).
	IN0 to IN3		Captures as trigger input for ch.0 to ch.3 of the input capture.
60 to 63	PA4 to PA7	C	It is General-purpose I/O port (Withstand voltage of 5 V).
	OUT0 to OUT3		Functions as the event output pins for ch.0 to ch.3 of the output compare.

**Table 1.5-1 Pin Function (6 / 6)**

Pin No.	Pin Name	Circuit Type	Functional description
64	PB0	C	It is General-purpose I/O port (Withstand voltage of 5 V).
	SCL1		Functions as the clock I/O pin for the I <sup>2</sup> C interface ch.1. Set port output to Hi-Z during I <sup>2</sup> C interface operations.
65	PB1	C	It is General-purpose I/O port (Withstand voltage of 5 V).
	SDA1		Functions as the data I/O pin for the I <sup>2</sup> C interface ch.1. Set port output to Hi-Z during I <sup>2</sup> C interface operations.
66	PB2	C	It is General-purpose I/O port (Withstand voltage of 5 V).
	SCL2		Functions as the clock I/O pin for the I <sup>2</sup> C interface ch.2. Set port output to Hi-Z during I <sup>2</sup> C interface operations.
67	PB3	C	It is General-purpose I/O port (Withstand voltage of 5 V).
	SDA2		Functions as the data I/O pin for the I <sup>2</sup> C interface ch.2. Set port output to Hi-Z during I <sup>2</sup> C interface operations.
68	PB4	C	It is General-purpose I/O port (Withstand voltage of 5 V).
69	PB5	D	It is General-purpose I/O port.
	PPG4		Function as ch.4 output pins for the PPG timer.
70	PB6	D	It is General-purpose I/O port.
	PPG5		Function as ch.5 output pins for the PPG timer.
71	UTEST	C	It is USB test pin. Requires a pull-down connection in normal use.
73	DVM	K	It is USB Function D- pin.
74	DVP	K	It is USB Function D + pin.
77	HVM	K	It is USB HOST D- pin.
78	HVP	K	It is USB HOST D + pin.
80	$\overline{\text{HCON}}$	E	It is external pull-up resistor pin.
36	AVcc	-	It is A/D converter power supply pin.
37	AVRH	J	It is A/D converter external reference power supply pin.
38	AVss	-	It is A/D converter power supply pin.
87 to 89	MD2 to MD0	B	It is Operation mode select input pin.
15, 75, 79, 105	Vcc	-	It is power supply pin.
16, 47, 72, 76, 106	Vss	-	It is power supply pin (GND).

## 1.6 I/O Circuit Types

Table 1.6-1 shows I/O circuit types for pins of a MB90330A series.

## ■ I/O Circuit Types

Table 1.6-1 I/O Circuit Types (1 / 3)

Classification	Circuit	Remarks
A		<ul style="list-style-type: none"> <li>Oscillation return resistance: X1, X0 approximately 1 MΩ X1A, X0A 10 MΩ</li> <li>With standby control</li> </ul>
B		CMOS hysteresis input
C		<ul style="list-style-type: none"> <li>CMOS hysteresis input</li> <li>N-ch open drain output</li> </ul>
D		<ul style="list-style-type: none"> <li>CMOS output</li> <li>CMOS hysteresis input</li> <li>With standby control</li> </ul>
E		CMOS output

**Table 1.6-1 I/O Circuit Types (2 / 3)**

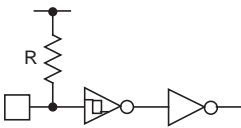
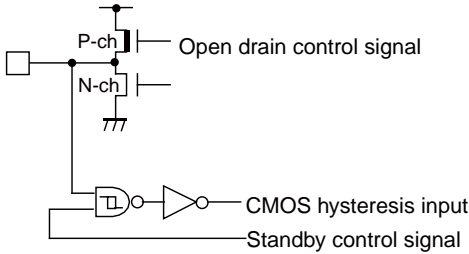
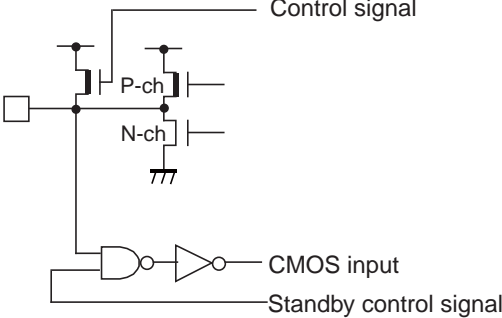
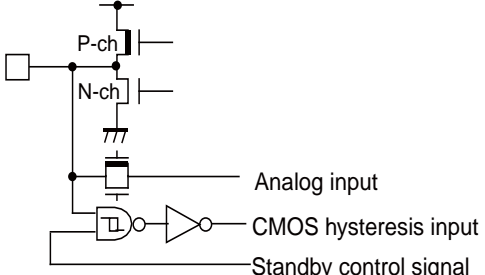
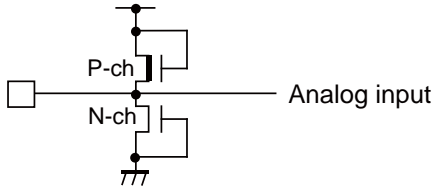
Classification	Circuit	Remarks
F	 <p>CMOS hysteresis input</p>	<ul style="list-style-type: none"> <li>• CMOS hysteresis input with pull-up</li> <li>• Resistance: Approximately 50 kΩ</li> </ul>
G	 <p>Open drain control signal</p> <p>CMOS hysteresis input</p> <p>Standby control signal</p>	<ul style="list-style-type: none"> <li>• CMOS output</li> <li>• CMOS hysteresis input</li> <li>• With open drain control</li> <li>• With standby control</li> </ul>
H	 <p>Control signal</p> <p>CMOS input</p> <p>Standby control signal</p>	<ul style="list-style-type: none"> <li>• CMOS output</li> <li>• CMOS input</li> <li>• With input pull-up resistor control</li> <li>• Resistance: Approximately 50 kΩ</li> <li>• With standby control</li> </ul>
I	 <p>Analog input</p> <p>CMOS hysteresis input</p> <p>Standby control signal</p>	<ul style="list-style-type: none"> <li>• CMOS output</li> <li>• CMOS hysteresis input</li> <li>• With standby control</li> <li>• Analog input</li> </ul>
J	 <p>Analog input</p>	A/D converter voltage input pin

Table 1.6-1 I/O Circuit Types (3 / 3)

Classification	Circuit	Remarks
K	<p>D<sup>+</sup> Input</p> <p>D<sup>-</sup> Input</p> <p>Differential input</p> <p>Full D<sup>+</sup> Output</p> <p>Full D<sup>-</sup> Output</p> <p>Low D<sup>+</sup> Output</p> <p>Low D<sup>-</sup> Output</p> <p>direction</p> <p>speed</p>	USB I/O pins
L	<p>P-ch</p> <p>N-ch</p> <p>Pout</p> <p>Nout</p> <p>CMOS input</p> <p>Standby control signal</p>	<ul style="list-style-type: none"> <li>• CMOS output</li> <li>• CMOS input (With input interception function at standby)</li> </ul>

## 1.7 Precautions when Using Devices

---

**This section describes the precautions when using devices.**

---

### ■ Precautions when Using Devices

#### ● Preventing Latch-up, Turning on Power Supply

Latch-up may occur on CMOS IC under the following conditions:

- If a voltage higher than  $V_{CC}$  or lower than  $V_{SS}$  is applied to input and output pins,
- If a voltage higher than the rated voltage is applied between  $V_{CC}$  pin to  $V_{SS}$  pin,
- If the  $AV_{CC}$  power supply is turned on before the  $V_{CC}$  voltage.

Ensure that you apply a voltage to the analog power supply at the same time as  $V_{CC}$  or after you turn on the digital power supply (when you perform power-off, turn off the analog power supply first or at the same time as  $V_{CC}$  and the digital power supply).

When latch-up occurs, power supply current increases rapidly and might thermally damage elements. When using CMOS IC, take great care to prevent the occurrence of latch-up.

#### ● Treatment of Unused Pins

Leaving unused input pins unconnected can cause abnormal operation or latch-up, leading to permanent damage.

Unused input pins should always be pulled up or down through resistance of at least 2 k $\Omega$ . Any unused input/output pins may be set to output mode and left open, or set to input mode and treated the same as unused input pins. If there is unused output pin, make it to open.

#### ● Treatment of Power Supply Pins on Models with A/D Converters

Even when the A/D converters are not in use, be sure to make the necessary connections  $AV_{CC} = AVR_H = V_{CC}$  and  $AV_{SS} = V_{SS}$ .

#### ● Treatment of Power Supply Pins ( $V_{CC}/V_{SS}$ )

In products with multiple  $V_{CC}$  or  $V_{SS}$  pins, the pins of the same potential are internally connected in the device to avoid abnormal operations including latch-up. However, you must connect the pins to external power supply and a ground line to lower the electro-magnetic emission level, to prevent abnormal operation of strobe signals caused by the rise in the ground level, and to conform to the total output current rating.

Moreover, connect the current supply source with the  $V_{CC}$  and  $V_{SS}$  pins of this device at the low impedance.

It is also advisable to connect a ceramic bypass capacitor of approximately 0.1  $\mu$ F between  $V_{CC}$  and  $V_{SS}$  near this device.

## ● About Crystal oscillator circuit

Noise near the X0/X1 pins and X0A/X1A pins may cause the device to malfunction. Design the printed circuit board so that X0/X1 pins and X0A/X1A pins, the crystal oscillator (or the ceramic oscillator) and the bypass capacitor to ground are located as close to the device as possible.

It is strongly recommended to design the PC board artwork with the X0/X1 pins and X0A/X1A pins surrounded by ground plane because stable operation can be expected with such a layout.

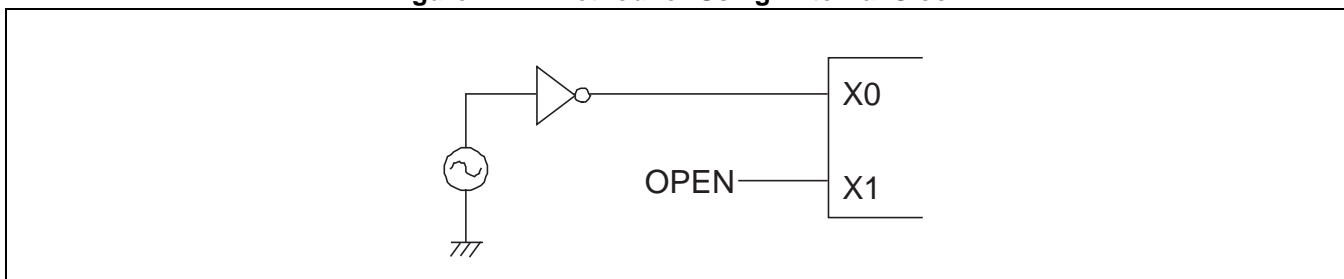
Please ask the crystal maker to evaluate the oscillational characteristics of the crystal and this device.

## ● Note on using external clock

If you are using the external clock, you must connect external pins as shown in the Figure 1.7-1.

Figure 1.7-1 illustrates an external clock usage. (under  $f=7$  MHz)

**Figure 1.7-1 Method for Using External Clock**



## ● Stabilization of Supply Voltage

A sudden change in the supply voltage may cause the device to malfunction even within the  $V_{CC}$  supply voltage operating range. For stabilization reference, the supply voltage should be stabilized so that  $V_{CC}$  ripple variations (peak-to-peak value) at commercial frequencies (50 Hz to 60 Hz) fall below 10% of the standard  $V_{CC}$  supply voltage and the transient regulation does not exceed 0.1 V/ms at temporary changes such as power supply switching.

## ● Crystal oscillator circuit of low voltage use

If you are using the device with voltages of 2.0 V or less, the external crystal oscillator may not oscillate at power-on. Therefore, It will recommend the use of the external clock.

## ● When the dual-supply is used as a single-supply device.

If you are using only a single-system of the MB90330A series that come in the dual-system product, use it with X0A= $V_{SS}$ : X1A=OPEN.

## ● Writing to flash memory

For serial writing to flash memory, always make sure that the operating voltage  $V_{CC}$  is between 3.13 V and 3.6 V.

For normal writing to flash memory, always make sure that the operating voltage  $V_{CC}$  is between 3.0 V and 3.6 V.



Ensure that you must write to Flash at the operation voltage  $V_{CC}$  of 3.0 V to 3.60 V.

● **Caution on Operations during PLL Clock Mode**

On this microcontroller, if in case the crystal oscillator breaks off or an external reference clock input stops while the PLL clock mode is selected, a self-oscillator circuit contained in the PLL may continue its operation at its self-running frequency. However, Fujitsu will not guarantee results of operations if such failure occurs.

● **Serial communication**

There is a possibility to receive wrong data due to noise or other causes on the serial communication. Therefore, design a printed circuit board so as to avoid noise.

Consider receiving of wrong data when designing the system. For example, apply a checksum to detect an error. If an error is detected, retransmit the data.

# CHAPTER 2

---

## CPU

**This chapter explains the setting and operation of the CPU.**

- 2.1 Overview of the CPU
- 2.2 Memory Space
- 2.3 Linear Addressing
- 2.4 Bank Addressing
- 2.5 Multibyte Data in Memory Space
- 2.6 Registers
- 2.7 Register Bank
- 2.8 Prefix Codes
- 2.9 Interrupt Disable Instructions

## 2.1 Overview of the CPU

The F<sup>2</sup>MC-16LX CPU core is a 16-bit CPU designed for applications that require high-speed real-time processing, such as consumer or vehicle-mounted equipments. The F<sup>2</sup>MC-16LX instruction set is designed for controller applications, and is capable of high-speed, highly efficient control processing.

### ■ Overview of the CPU

In addition to 16-bit data, the F<sup>2</sup>MC-16LX CPU core can process 32-bit data using an internal 32-bit accumulator. Up to 16Mbytes of memory space (expandable) can be used, which can be accessed by either the linear pointer or bank method. The instruction set, based on the F<sup>2</sup>MC-8L A-T architecture, has been reinforced by adding instructions compatible with high-level languages, expanding addressing modes, reinforcing multiplication and division instructions, and enhancing bit processing.

The features of the F<sup>2</sup>MC-16LX CPU are explained below:

- Minimum execution time

- 41.7 ns (at machine clock 24MHz)
- The frequency of the machine clock varies depending on the model.

- Maximum memory space

16Mbytes, accessed in linear or bank method

- Instruction set optimized for controller applications

- Rich data types: Bit, byte, word, long word
- Extended addressing modes: 23 types
- Reinforced high-precision operation (32-bit length) with 32-bit accumulator

- Powerful interrupt function

8 priority levels (programmable)

- CPU-independent automatic transfer function

- Up to 16 channels of the extended intelligent I/O service (EI<sup>2</sup>OS)
- Up to 16 channels of the DMA transfer (μDMAC)
- The DMA transfer (μDMAC) might not be built-in by the series.

- Instruction set for high-level language (C language)/multitask

System stack pointer/instruction set symmetry/barrel-shift instructions

- Higher execution speed

4-byte queue

# MB90330A Series

## 2.2 Memory Space

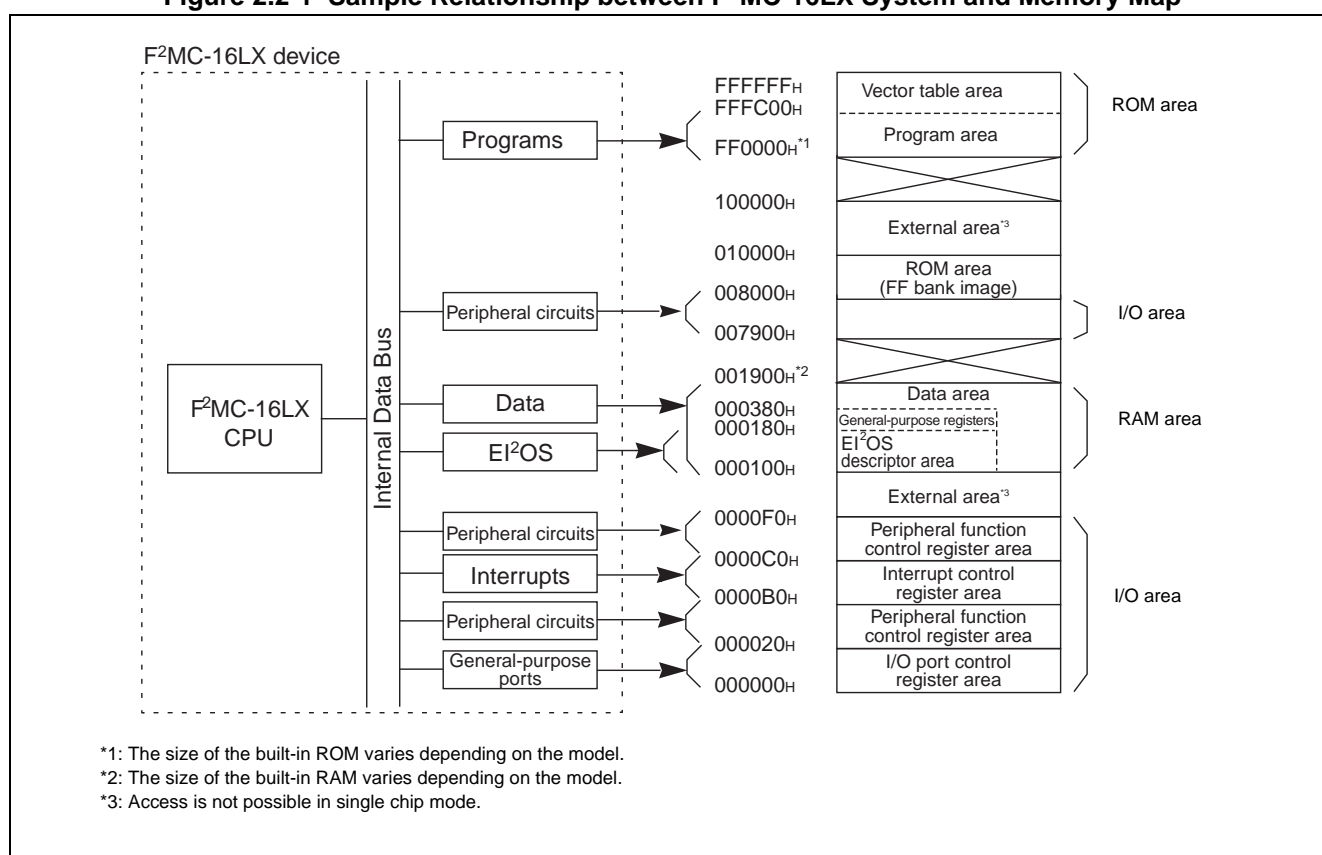
**F<sup>2</sup>MC-16LX CPU has the memory space of 16Mbytes.**

### ■ Overview of CPU Memory Space

An F<sup>2</sup>MC-16LX CPU has 16Mbytes of memory space where all data program I/Os managed by the F<sup>2</sup>MC-16LX CPU are located. The CPU accesses the resources by indicating their addresses using a 24-bit address bus.

Figure 2.2-1 shows a sample relationship between the F<sup>2</sup>MC-16LX system and memory map.

**Figure 2.2-1 Sample Relationship between F<sup>2</sup>MC-16LX System and Memory Map**



## ■ ROM Area

- Vector table area (address: FFFC00<sub>H</sub> to FFFFFFF<sub>H</sub>)
  - This area is used as a vector table for the reset, interrupt, and CALLV vectors.
  - This area is allocated at the highest addresses of the ROM area. The start address of the corresponding processing routine is set as data in each vector table address.
- Program area (address: Up to FFFBFF<sub>H</sub>)
  - ROM is built in as an internal program area.
  - The size of internal ROM varies depending on the model.

## ■ RAM Area

- Data area (address: 000100<sub>H</sub> to 0018FF<sub>H</sub> (for 6Kbytes))
  - The static RAM is built in as an internal data area.
  - The size of internal RAM varies depending on the model.
- General-purpose register area (address: 000180<sub>H</sub> to 00037F<sub>H</sub>)
  - Auxiliary registers, used for 8-bit, 16-bit, and 32-bit arithmetic operations and transfer, are allocated in this area.
  - Since this area is allocated to a part of the RAM area, it can be used as ordinary RAM.
  - When this area is used as a general-purpose register, general-purpose register addressing enables high speed access with short instructions.
- Extended intelligent I/O service (EI<sup>2</sup>OS) descriptor area (address: 0000100<sub>H</sub> to 00017F<sub>H</sub>)
  - This area retains the transfer modes, I/O addresses, transfer count, and buffer addresses.
  - Since this area is allocated to a part of the RAM area, it can be used as ordinary RAM.

## ■ I/O Area

- Interrupt control register area (address: 0000B0<sub>H</sub> to 0000BF<sub>H</sub>)

The interrupt control registers (ICR00 to ICR15) support all peripheral functions that have an interrupt function, and perform the interrupt levels setting and the control of the extended intelligent I/O service (EI<sup>2</sup>OS).
- Peripheral function control register area  
(address: 000020<sub>H</sub> to 0000AF<sub>H</sub>, 0000C0<sub>H</sub> to 0000EF<sub>H</sub>, and 007900<sub>H</sub> to 007FFF<sub>H</sub>)

This register controls the peripheral functions and inputs/outputs of data.
- I/O port control register area (address: 000000<sub>H</sub> to 00001F<sub>H</sub>)

This register controls I/O ports, and inputs/outputs of data.

## ■ Address Generation Methods

The F<sup>2</sup>MC-16LX has the following two addressing methods:

- Linear addressing

An 24-bit address is specified by an instruction.

- Bank addressing

Upper 8-bit of an address are specified by an appropriate bank register, and the remaining lower 16-bit of an address are specified by an instruction.

## 2.3 Linear Addressing

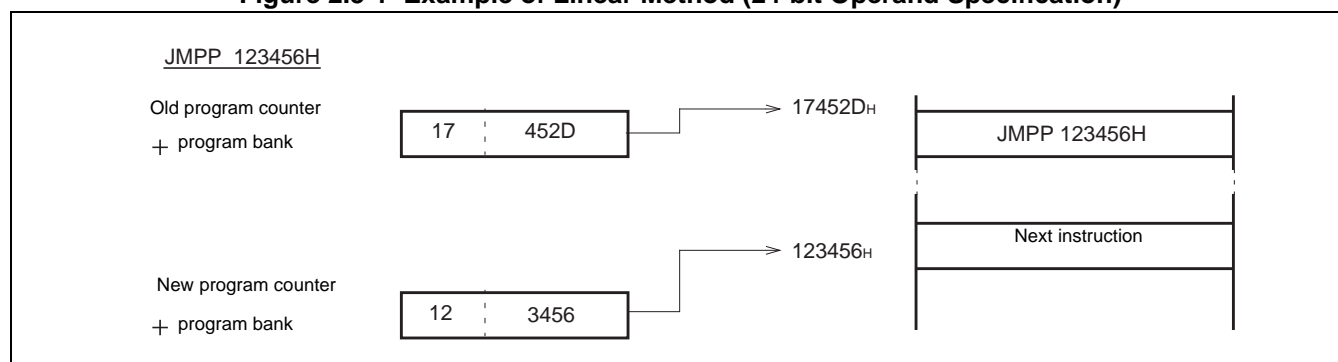
There are two types of linear addressing:

- **24-bit operand specification:** Directly specifies a 24-bit address using operands.
- **32-bit register indirect specification:** Uses the lower 24-bit of a 32-bit general-purpose register contents as the address.

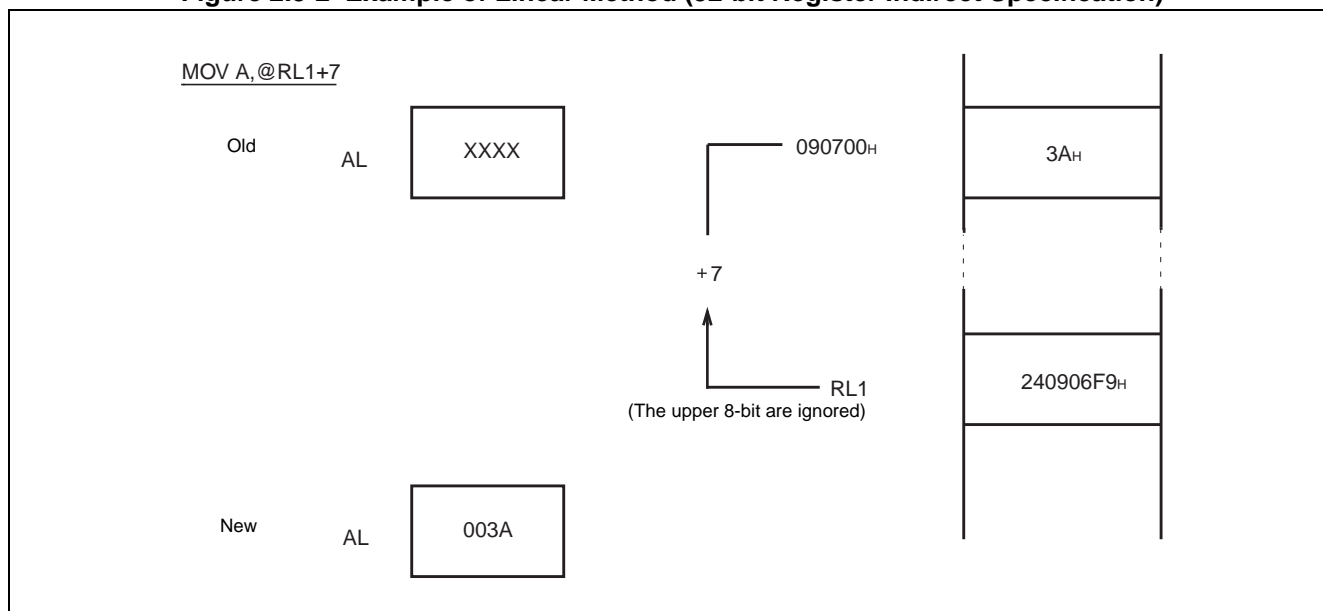
### ■ 24-bit Operand Specification

Figure 2.3-1 and Figure 2.3-2 show examples of 24-bit operand specification and 32-bit register indirect specification, respectively.

**Figure 2.3-1 Example of Linear Method (24-bit Operand Specification)**



**Figure 2.3-2 Example of Linear Method (32-bit Register Indirect Specification)**



## MB90330A Series

### 2.4 Bank Addressing

In the bank method, the 16 M byte space is divided into 256 of 64 K byte banks. The following five bank registers are used to specify the banks corresponding to each space:

- Program counter bank register (PCB)
- Data bank register (DTB)
- User stack bank register (USB)
- System stack bank register (SSB)
- Additional data bank register (ADB)

#### ■ Bank Addressing

##### ● Program counter bank register (PCB)

The 64Kbyte bank specified by the program counter bank register (PCB) is called a program (PC) space. The PC space typically contains instruction codes, vector tables, and immediate data.

##### ● Data bank register (DTB)

The 64Kbyte bank specified by the data bank register (DTB) is called a data (DT) space. The DT space typically contains readable/ writable data, and control/data registers for internal and external resources.

##### ● User stack bank register (USB) and System stack bank register (SSB)

The 64Kbyte bank specified by the user stack bank register (USB) or system stack bank register (SSB) is called a stack (SP) space. The SP space is accessed when a stack access occurs during a push/pop instruction or interrupt register saving. The S flag in the condition code register determines which stack space is to be accessed.

##### ● Additional data bank register (ADB)

The 64Kbyte bank specified by the Additional data bank register (ADB) is called an additional (AD) space. The AD space typically contains data that cannot fit into the DT space.

Table 2.4-1 lists the default spaces used in each addressing mode, which are pre-determined to improve instruction coding efficiency. To use a non-default space for an addressing mode, specify a prefix code corresponding to a bank before the instruction. This enables access to the bank space corresponding to the specified prefix code.

By resetting, the DTB, USB, SSB, and ADB are initialized to 00<sub>H</sub>. The PCB is initialized to a value specified by the reset vector. After reset, the DT, SP, and AD spaces are allocated in bank 00<sub>H</sub> (000000<sub>H</sub> to 00FFFF<sub>H</sub>), and the PC space is allocated in the bank specified by the reset vector.

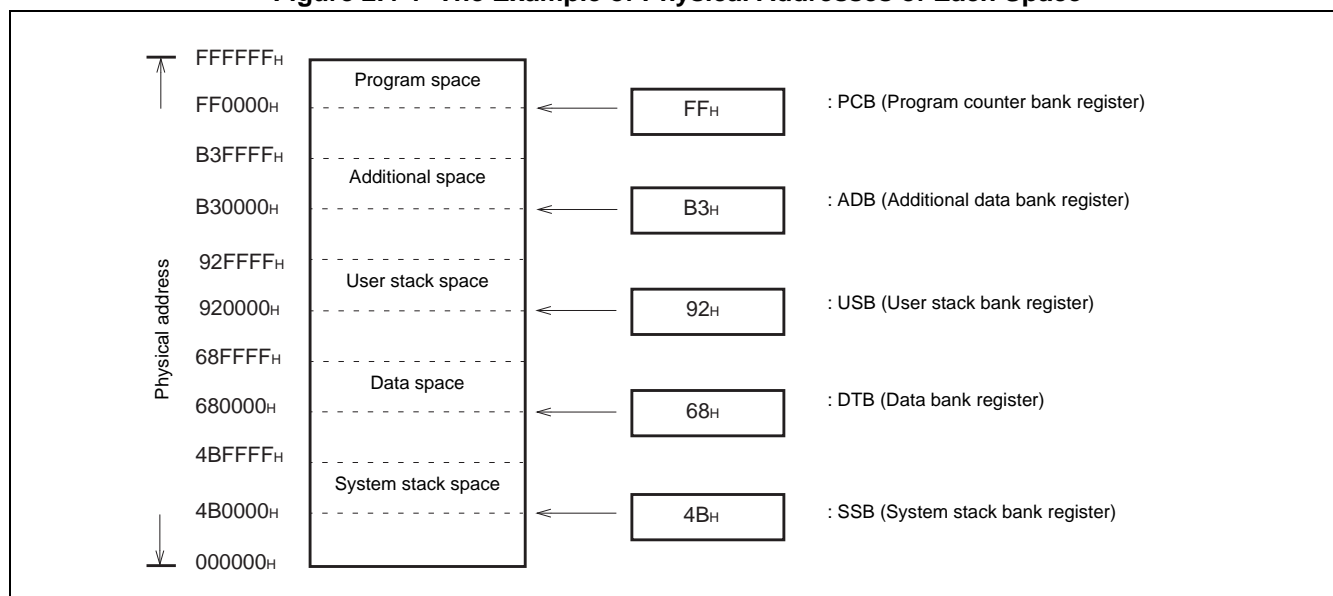


**Table 2.4-1 Default Space**

Default space	Addressing
Program space	PC indirect, program access, branch
Data space	@A, addr16, dir, and addressing using @RW0, @RW1, @RW4, or @RW5
Stack space	Addressing using PUSHW, POPW, @RW3, or @RW7
Additional space	Addressing using @RW2 or @RW6

The example of the physical address of the memory space divided into the register bank is shown in Figure 2.4-1.

**Figure 2.4-1 The Example of Physical Addresses of Each Space**



# MB90330A Series

## 2.5 Multibyte Data in Memory Space

Multibyte data is allocated from the low-order addresses to the high-order addresses in the memory space in the order from the byte in LSB to the byte in MSB.

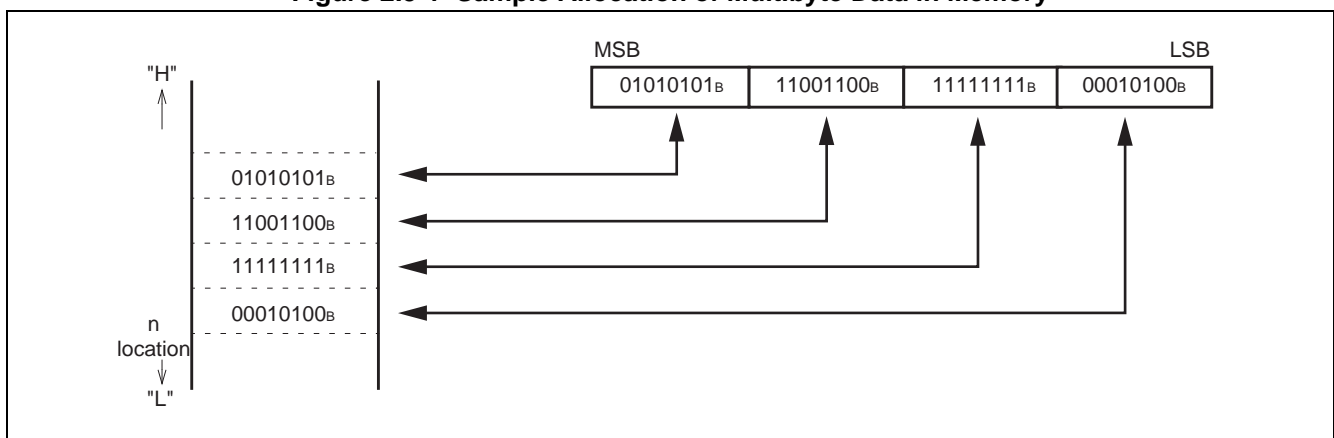
### ■ Multibyte Data Allocation in Memory Space

Data is written to memory from the low-order addresses. Therefore, for a 32-bit data item, the lower 16 bits are transferred before the upper 16 bits.

If a reset signal is input immediately after the lower bits are written, the upper bits might not be written.

Figure 2.5-1 shows a sample allocation of multibyte data in memory. The lower 8 bits of a data item are stored at address  $n$ , then address  $n+1$ , address  $n+2$ , address  $n+3$ , etc.

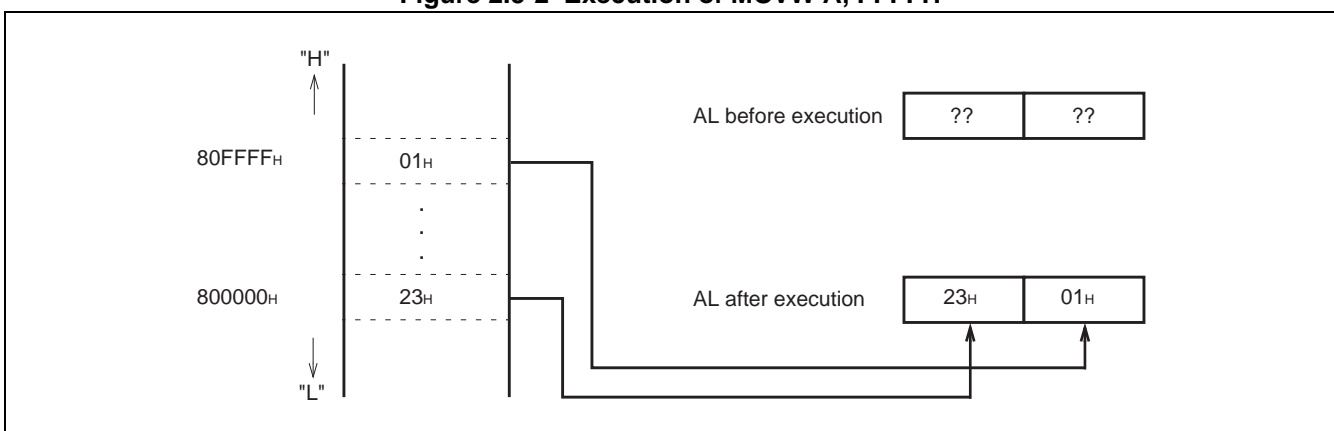
Figure 2.5-1 Sample Allocation of Multibyte Data in Memory



### ■ Accessing Multibyte Data

Basically, all accesses are made within a bank. For an instruction accessing a multibyte data item, the next address of  $\text{FFFF}_H$  location is  $\text{0000}_H$  location of the same bank. Figure 2.5-2 shows an example of an instruction accessing multibyte data.

Figure 2.5-2 Execution of MOVW A, FFFFH



## 2.6 Registers

---

The F<sup>2</sup>MC-16LX registers are largely classified into two types: dedicated registers and general-purpose registers.

The dedicated registers exist as dedicated internal hardware of the CPU, and they have specific use defined by the CPU architecture.

The applications of the general-purpose registers can be specified by the user, as is ordinary memory space. Sharing the CPU address space with RAM, the general-purpose registers are the same as the dedicated registers in that they can be accessed without using an address.

---

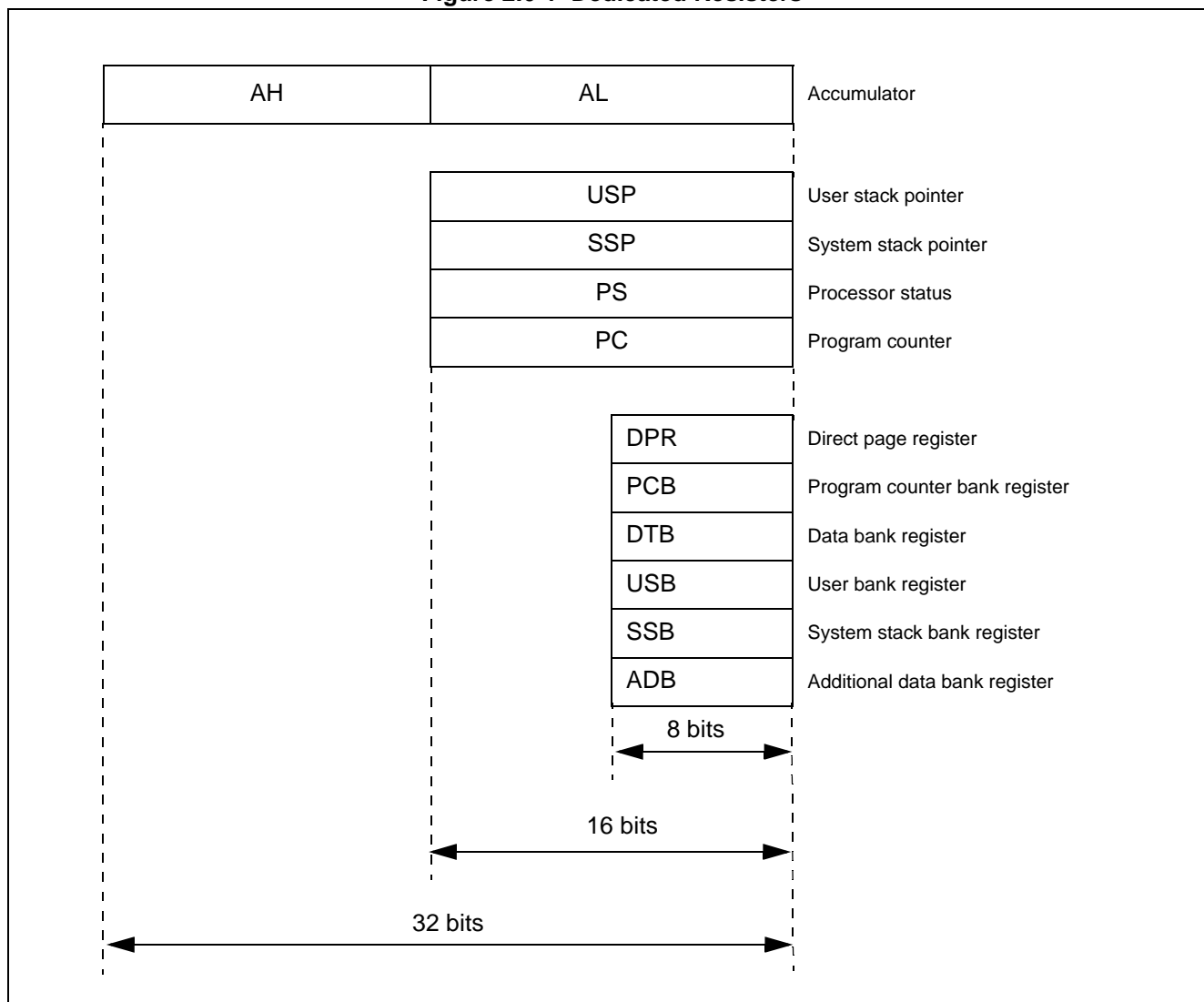
### ■ Dedicated Registers

The F<sup>2</sup>MC-16LX CPU core has the following 11 dedicated registers:

- Accumulator (A=AH: AL) : 2 × 16-bit accumulators  
(Can be used as a single 32-bit accumulator.)
- User stack pointer (USP) : 16-bit pointer indicating the user stack area
- System stack pointer (SSP) : 16-bit pointer indicating the system stack area
- Processor status (PS) : 16-bit register indicating the system status
- Program counter (PC) : 16-bit register containing the address where the program is stored
- Program counter bank register (PCB) : 8-bit register indicating the PC space
- Data bank register (DTB) : 8-bit register indicating the DT space
- User stack bank register (USB) : 8-bit register indicating the user stack space
- System stack bank register (SSB) : 8-bit register indicating the system stack space
- Additional data bank register (ADB) : 8-bit register indicating the AD space
- Direct page register (DPR) : 8-bit register indicating a direct page

Figure 2.6-1 shows the configuration of the dedicated registers.

Figure 2.6-1 Dedicated Registers

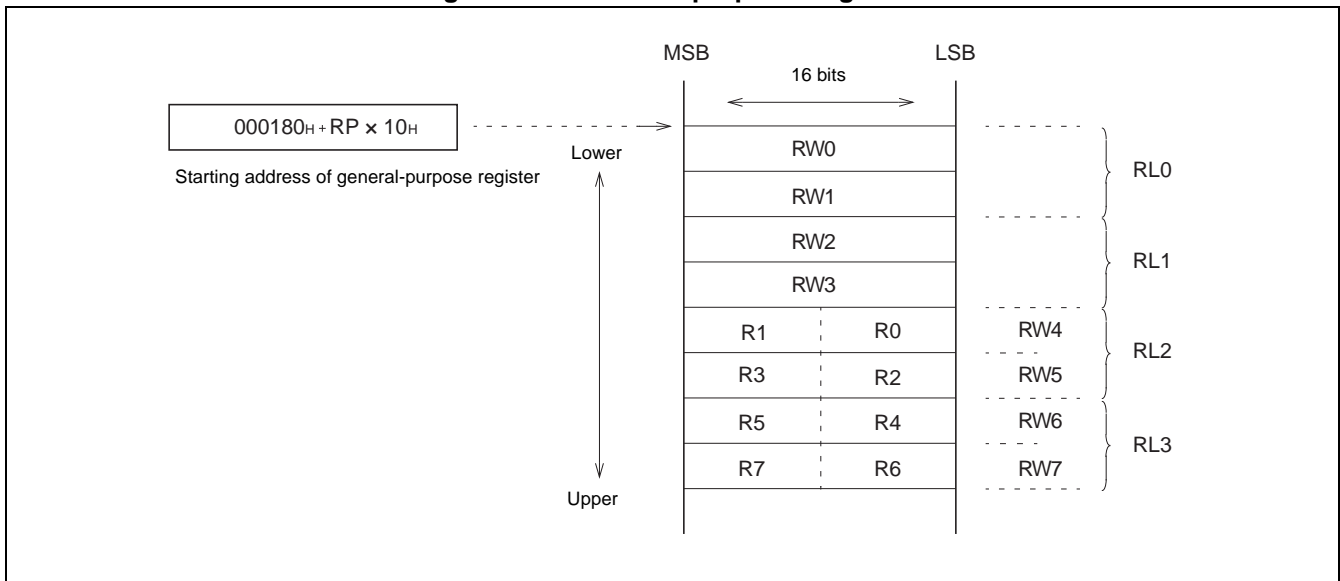


## ■ General-purpose Registers

As described in Figure 2.6-2, the F<sup>2</sup>MC-16LX general-purpose registers are located from 000180<sub>H</sub> to 00037F<sub>H</sub> (maximum configuration) of main storage. The register bank pointer (RP) indicates which of the above addresses is currently being used as a register bank. Each bank has the following three types of registers. These registers are mutually dependent and have a relationship as shown below:

- R0 to R7 : 8-bit general-purpose registers
- RW0 to RW7 : 16-bit general-purpose registers
- RL0 to RL3 : 32-bit general-purpose registers

**Figure 2.6-2 General-purpose Registers**



The relationship between the upper/lower bytes of a byte or word register is expressed as follows:

$$RW_{(i+4)} = R_{(i \times 2 + 1)} \times 256 + R_{(i \times 2)} \quad [i=0 \text{ to } 3]$$

The relationship between the upper/lower bytes of RL<sub>i</sub> and RW is expressed as follows:

$$RL_{(i)} = RW_{(i \times 2 + 1)} \times 65536 + RW_{(i \times 2)} \quad [i=0 \text{ to } 3]$$

# MB90330A Series

## 2.6.1 Accumulator (A)

The accumulator (A) register consists of  $2 \times 16$ -bit arithmetic operation registers (AH and AL), and is used as a temporary register for operation results and transfer data.

### ■ Accumulator (A)

During 32-bit data processing, AH and AL are used together (see Figure 2.6-3). Only AL is used for word processing in 16-bit data processing mode or for byte processing in 8-bit data processing mode (see Figure 2.6-4). The data stored in the accumulator (A) register can be operated upon with the data in memory or registers (Ri, RWi, and RLi). In the same manner as with the F<sup>2</sup>MC-8L, when a word or shorter data item is transferred to AL, the previous data item in AL is automatically sent to AH (data preservation function). The data preservation function and operation between AL and AH help improve processing efficiency.

When a byte or shorter data item is transferred to AL, the data is sign-extended or zero-extended and stored as a 16-bit data item in AL. The data in AL can be handled either as word or byte long. When a byte-processing arithmetic operation instruction is executed on AL, the upper 8 bits of AL before operation are ignored. The upper 8 bits of the operation result all become "0". The A register is not initialized by a reset and holds an undefined value right after the reset.

Figure 2.6-3 An Example of 32-bit Data Transfer

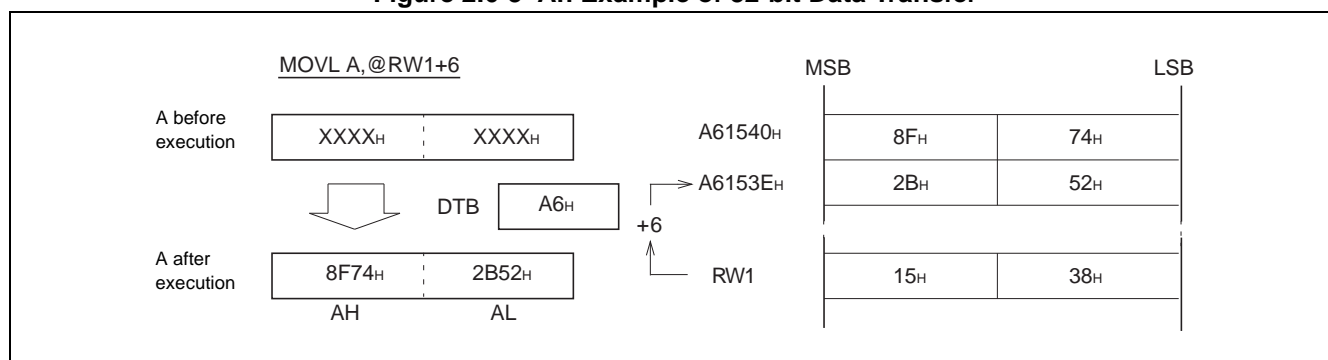
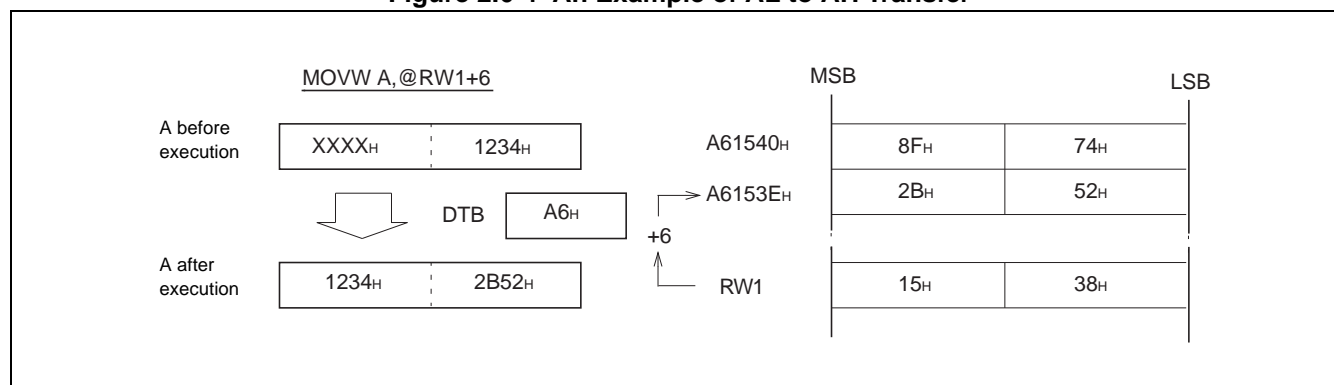


Figure 2.6-4 An Example of AL to AH Transfer



## 2.6.2 User Stack Pointer (USP) and System Stack Pointer (SSP)

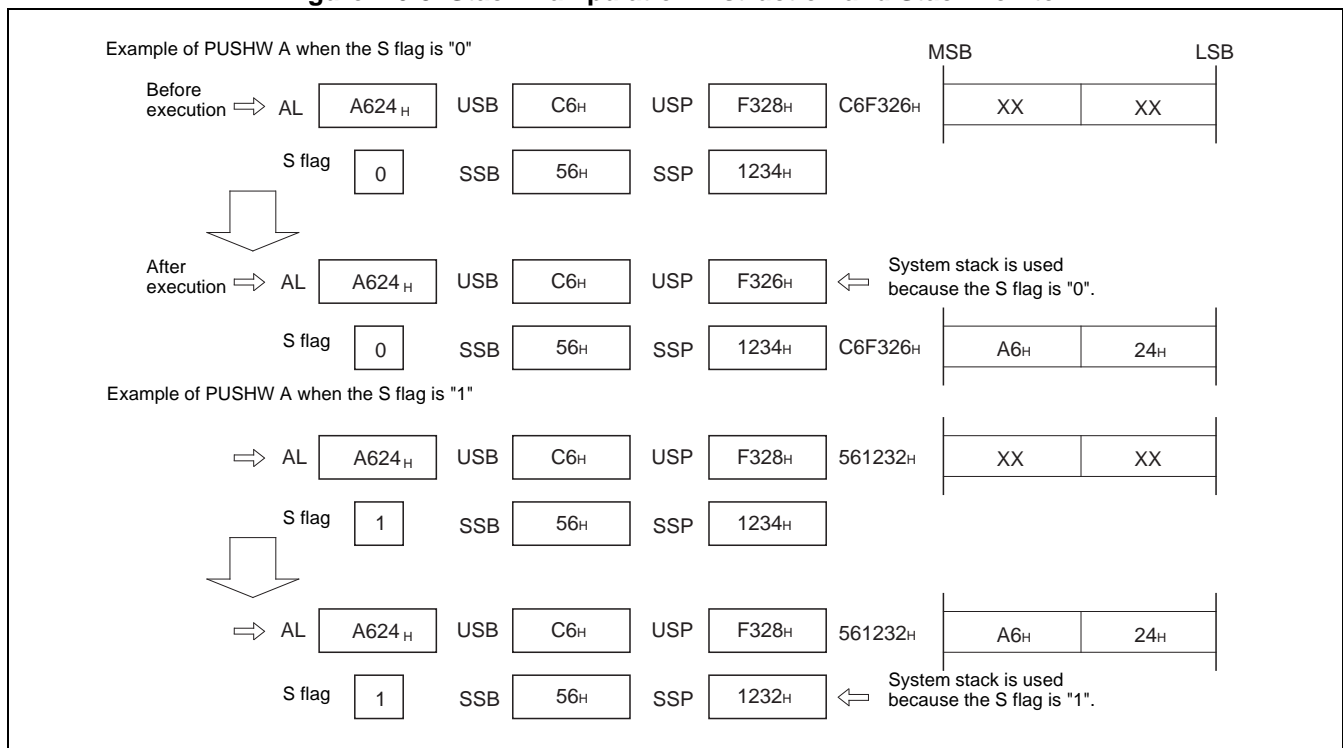
User stack pointer (USP) and system stack pointer (SSP) are 16-bit registers that indicate the memory addresses for saving/restoring data when a push/pop instruction or subroutine is executed.

### ■ User Stack Pointer (USP) and System Stack Pointer (SSP)

User stack pointer (USP) and system stack pointer (SSP) registers are used by the stack instructions. However, the USP register is enabled when the S flag in the processor status register is "0", and the SSP register is enabled when the S flag is "1" (see Figure 2.6-5). Since the S flag is set when an interrupt is accepted, register values are always saved in the memory area indicated by SSP during interrupt processing. SSP is used for stack processing in an interrupt routine, while USP is used for stack processing other than in an interrupt routine. If you do not need to divide the stack space, use only the SSP.

During stack processing, the upper 8 bits of an address are indicated by SSB (for SSP) or USB (for USP). USP and SSP are not initialized by a reset. Instead, they hold undefined values.

Figure 2.6-5 Stack Manipulation Instruction and Stack Pointer



Note:

When you specify a value to be set in the stack pointer, use an even-numbered address whenever possible.

## MB90330A Series

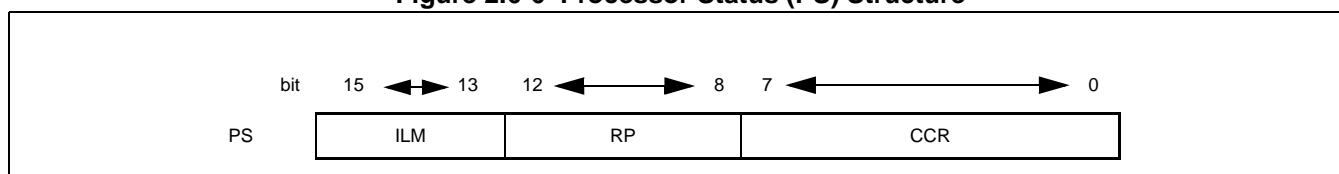
### 2.6.3 Processor Status (PS)

The processor status (PS) register consists of the bits controlling the CPU operation and the bits indicating the CPU status.

#### ■ Processor Status (PS)

As shown in Figure 2.6-6, the upper bytes of the PS register consist of the register bank pointers (RP) and the interrupt level mask register (ILM) that indicate the starting address of a register bank. The lower bytes of the PS register consist of the condition code register (CCR), containing the flags to be set or reset depending on the results of instruction execution or interrupt occurrences.

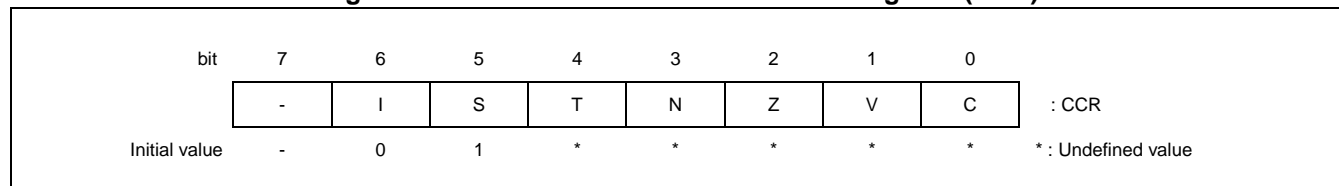
Figure 2.6-6 Processor Status (PS) Structure



#### ■ Condition Code Register (CCR)

Figure 2.6-7 shows the structure of the condition code register.

Figure 2.6-7 Structure of Condition Code Register (CCR)



#### ● Interrupt enable flag (I)

Interrupts other than software interrupts are enabled when the I flag is "1," and are disabled when the I flag is "0". The I flag is cleared to "0" by a reset.

#### ● Stack flag (S)

When the S flag is "0", USP is enabled as the stack manipulation pointer. When the S flag is "1", SSP is enabled as the stack manipulation pointer. The S flag is set to "1" by an interrupt reception or a reset.

#### ● Sticky bit flag (T)

"1" is set in the T flag when there is one or more "1" in the data shifted out from the carry after execution of a logical right/arithmetic right shift instruction. Otherwise, "0" is set in the T flag.

#### ● Negative flag (N)

The "1" is set in the N flag when the MSB of the operation result is "1". Otherwise, N flag is cleared to "0".



● Zero flag (Z)

The Z flag is set to "1" when the operation result is all "0". Otherwise, Z flag is cleared to "0".

● Overflow flag (V)

The V flag is set when an overflow of a signed value occurs as a result of operation execution. In other cases, V flag is cleared to "0".

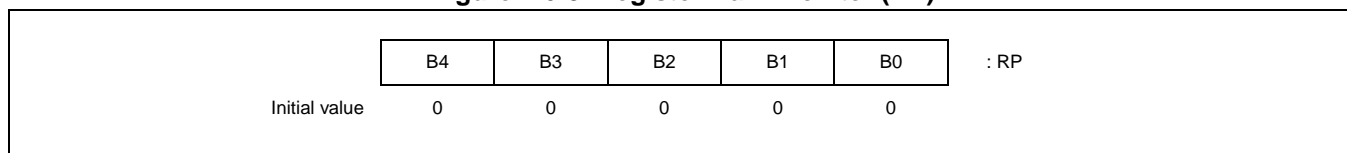
● Carry flag (C)

The C flag is set when a carry-up or carry-down from the MSB occurs as a result of operation execution. In other cases, C flag is cleared to "0".

## ■ Register Bank Pointer (RP)

As shown in Figure 2.6-8, the register bank pointer (RP) register indicates the relationship between the general-purpose registers of the F<sup>2</sup>MC-16LX and the internal RAM addresses where the general-purpose registers exist. Specifically, the RP register indicates the starting memory address of the currently used register bank in the following conversion expression:  $[00180_H + (RP) \times 10_H]$ . The RP register that consists of five bits can take a value between "00<sub>H</sub>" and "1F<sub>H</sub>" and allocate the register banks at addresses from 000180<sub>H</sub> to 00037F<sub>H</sub> in the memory. Even within that range, however, the register banks cannot be used as general-purpose registers if the banks are not in internal RAM. All RP registers are initialized to "0" by a reset. An instruction may transfer an 8-bit immediate value to the RP register but, only the lower 5 bits of that data are used.

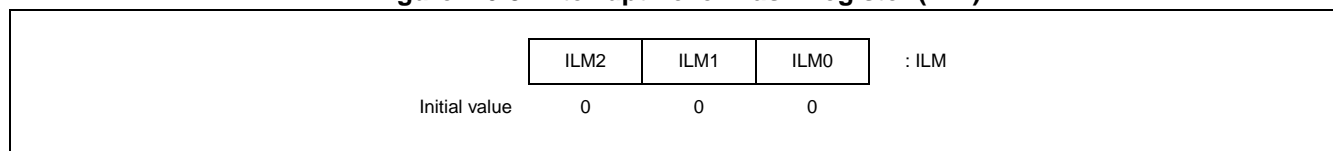
Figure 2.6-8 Register Bank Pointer (RP)



## ■ Interrupt Level Mask Register (ILM)

As described in Figure 2.6-9, the interrupt level mask register (ILM) consists of 3 bits, indicating the CPU interrupt masking level. Only an interrupt request of which interrupt level is higher than that indicated by these 3 bits will be accepted. Level 0 is the highest priority interrupt, and level 7 is the lowest priority interrupt (see Table 2.6-1). Therefore, for an interrupt to be accepted, its level value must be smaller than the current ILM value. When an interrupt is accepted, the level value of that interrupt is set in ILM. Thus, a subsequent interrupt of the same or lower level cannot be accepted. All ILMs are initialized to "0" by a reset. An instruction may transfer an 8-bit immediate value to the ILM register, but only the lower 3 bits of that data are used.

Figure 2.6-9 Interrupt Level Mask Register (ILM)



**Table 2.6-1 Levels Indicated by the Interrupt Level Mask Register (ILM)**

ILM2	ILM1	ILM0	Level value	Acceptable interrupt level
0	0	0	0	Interrupts disabled
0	0	1	1	Level value less than 1 (0 only)
0	1	0	2	Level value less than 2 (0 and 1)
0	1	1	3	Level value less than 3 (0, 1 and 2)
1	0	0	4	Level value less than 4 (0, 1, 2 and 3)
1	0	1	5	Level value less than 5 (0, 1, 2, 3 and 4)
1	1	0	6	Level value less than 6 (0, 1, 2, 3, 4 and 5)
1	1	1	7	Level value less than 7 (0, 1, 2, 3, 4, 5 and 6)

## 2.6.4 Program Counter (PC)

---

**Program counter (PC) shows lower 16-bit of the memory address of the instruction code that CPU should execute.**

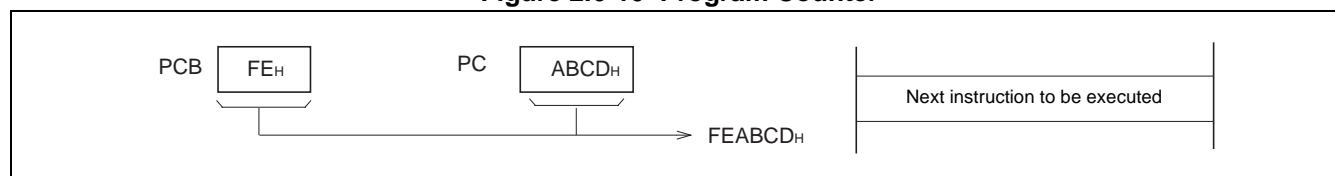
---

### ■ Program Counter (PC)

The program counter (PC) register is a 16-bit counter that indicates the lower 16 bits of the memory address of an instruction code to be executed by the CPU. The upper 8 bits of the address are indicated by the PCB. The PC register is updated by a conditional branch instruction, subroutine call instruction, interrupt, or reset. The PC register can also be used as a base pointer for operand access.

Figure 2.6-10 shows the program counter.

**Figure 2.6-10 Program Counter**



## 2.6.5 Bank Registers (PCB, DTB, USB, SSB, ADB)

---

The bank register shows the memory bank where the program space, the data space, the user stack space, the system stack space, and the Additional space are arranged.

---

### ■ Bank Registers (PCB, DTB, USB, SSB, ADB)

The bank registers includes the following five registers.

- Program Count Bank Register (PCB) <Initial Value: Value in Reset Vector>
- Data bank register (DTB) < Initial value: 00<sub>H</sub> >
- User stack bank register (USB) < Initial value: 00<sub>H</sub> >
- System stack bank register (SSB) < Initial value: 00<sub>H</sub> >
- Additional data bank register (ADB) < Initial value: 00<sub>H</sub> >

Each bank register indicates memory banks to which PC, DT, SP (user), SP (system), and AD space are allocated.

All bank registers has a length of 1 byte. They are initialized to "00<sub>H</sub>" by a reset. Bank registers other than PCB can be read. PCB can be read, but writing to PCB is not permitted.

PCB is updated either when the JMPP, CALLP, RETP, RETI, or RETF instruction that branches is executed, and it may then branch to an entire 16-M bytes space. PCB is also updated when an interrupt occurs. For information on the operation of each register, see Section "2.2 Memory Space".

## 2.6.6 Direct Page Register (DPR)

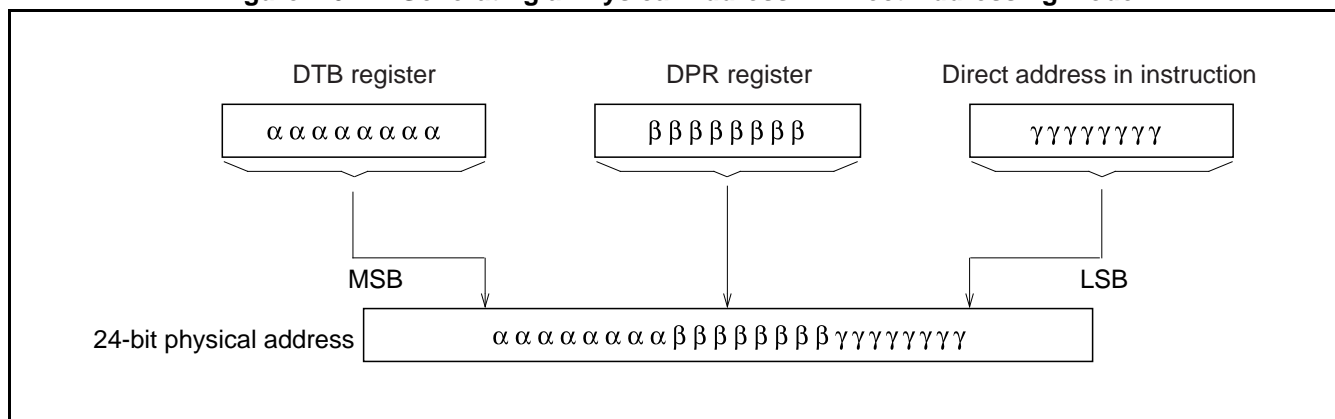
This section explains the direct page register (DPR) functions.

### ■ Direct Page Register (DPR) <Initial Value: 01<sub>H</sub>>

The direct page register (DPR) specifies, as shown in Figure 2.6-11, addresses 8 to 15 of an instruction operand in the direct addressing mode. DPR has a length of 8 bits, and is initialized to "01<sub>H</sub>" by a reset. It also allows reading and writing by instructions.

Figure 2.6-11 illustrates the generation of a physical address in the direct addressing mode.

**Figure 2.6-11 Generating a Physical Address in Direct Addressing Mode**



**MB90330A Series****2.7 Register Bank**

A register bank that consists of 8 words can be used as the general-purpose registers for the arithmetic operations or as the pointers for the instructions, such as byte registers (R0 to R7), word registers (RW0 to RW7), and long word registers (RL0 to RL3). In addition, RL0 to RL3 can also be used as the linear pointers to access directly to the entire space in the memory space.

**■ Register Bank**

Table 2.7-1 lists the register functions. Table 2.7-2 shows the relationship between each register.

In the same manner as for an ordinary RAM area, the register bank values are not initialized by a reset. The status before a reset is maintained. When the power is turned-on, however, the register bank will have an undefined value.

**Table 2.7-1 Register Functions**

R0 to R7	Used as operands of instructions. Note: R0 is also used as a counter for barrel shift or normalization instruction
RW0 to RW7	Used as pointers and operands of instructions. Note: RW0 is used as a counter for string instructions.
RL0 to RL3	Used as long pointers and operands of instructions.

**Table 2.7-2 Relationship between Registers**

Address	Byte register	Word register	Long word register	
$000180_{\text{H}} + \text{RP} \times 10_{\text{H}} + 0$		RW0	RL0	
$000180_{\text{H}} + \text{RP} \times 10_{\text{H}} + 1$				
$000180_{\text{H}} + \text{RP} \times 10_{\text{H}} + 2$		RW1		
$000180_{\text{H}} + \text{RP} \times 10_{\text{H}} + 3$				
$000180_{\text{H}} + \text{RP} \times 10_{\text{H}} + 4$		RW2	RL1	
$000180_{\text{H}} + \text{RP} \times 10_{\text{H}} + 5$				
$000180_{\text{H}} + \text{RP} \times 10_{\text{H}} + 6$		RW3		
$000180_{\text{H}} + \text{RP} \times 10_{\text{H}} + 7$				
$000180_{\text{H}} + \text{RP} \times 10_{\text{H}} + 8$	R0	RW4	RL2	
$000180_{\text{H}} + \text{RP} \times 10_{\text{H}} + 9$	R1			
$000180_{\text{H}} + \text{RP} \times 10_{\text{H}} + 10$	R2	RW5		
$000180_{\text{H}} + \text{RP} \times 10_{\text{H}} + 11$	R3			
$000180_{\text{H}} + \text{RP} \times 10_{\text{H}} + 12$	R4	RW6	RL3	
$000180_{\text{H}} + \text{RP} \times 10_{\text{H}} + 13$	R5			
$000180_{\text{H}} + \text{RP} \times 10_{\text{H}} + 14$	R6	RW7		
$000180_{\text{H}} + \text{RP} \times 10_{\text{H}} + 15$	R7			

## 2.8 Prefix Codes

Placing a prefix code before an instruction can partially change the operation of the instruction. 3 types of prefix codes can be used: bank select prefix, common register bank prefix, and flag change disable prefix.

### ■ Bank Select Prefix

The memory space used for accessing data depends on each addressing mode. When a bank select prefix is placed before an instruction, the memory space used for accessing data by that instruction can be selected regardless of the addressing mode.

Table 2.8-1 shows the bank select prefixes and selected memory spaces.

**Table 2.8-1 Bank Select Prefix**

Bank select prefix	Selected space
PCB	PC space
DTB	Data space
ADB	AD space
SPB	Either the SSP or USP space is used according to the stack flag value.

Use the following instructions with care:

- String instructions (MOVS / MOVSW / SCEQ / SCWEQ / FILS / FILSW)

The bank register specified by an operand is used regardless of the prefix.

- Stack manipulation instructions (PUSHW / POPW)

SSB or USB is used according to the S flag regardless of the prefix.

- I/O access instructions

MOV A,io	MOV io,A	MOVX A,io	MOVW A,io	MOVW io,A
MOV io,#imm8	MOVW io,#imm16	MOVB A,io:bp	MOVB io:bp,A	SETB io:bp
CLRB io:bp	BBC io:bp,rel	BBS io:bp,rel	WBTC	WBTS

The I/O space of the bank is used regardless of the prefix.

- Flag change instructions (AND CCR,#imm8 / OR CCR,#imm8)

The instruction is executed normally, but the prefix affects the next instruction.

- POPW PS

Either SSB or USB is used according to the S flag regardless of the prefix. The prefix affects the next instruction.

- MOV ILM,#imm8

The instruction is executed normally, but the prefix affects the next instruction.

- RETI

SSB is used regardless of the prefix.

## ■ Common Register Bank Prefix (CMR)

To simplify data exchange between multiple tasks, the same register bank must be accessed regardless of the RP value. When the common register bank prefix (CMR) is placed before an instruction that accesses the register bank, that instruction accesses the common bank (the register bank selected when RP=0) at addresses from 000180<sub>H</sub> to 00018F<sub>H</sub> regardless of the current RP value. Use the following instructions with care:

- String instructions (MOVS / MOVSW / SCEQ / SCWEQ / FILS / FILSW)

If an interrupt request occurs during execution of a string instruction with a prefix code, the string instruction is executed falsely because the prefix becomes invalid for the string instruction after the interrupt is returned. Do not attach CMR prefix to any of the above string instructions.

- Flag change instructions (AND CCR,#imm8 / OR CCR,#imm8 / POPW PS)

The instruction is executed normally, but the prefix affects the next instruction.

- MOV ILM,#imm8

The instruction is executed normally, but the prefix affects the next instruction.

## ■ Flag Change Disable Prefix (NCC)

To disable a flag change, use the flag change disable prefix code (NCC). Placing NCC before an instruction that disables an unwanted flag change can disable flag changes associated with that instruction. Use the following instructions with care:

- String instructions (MOVS / MOVSW / SCEQ / SCWEQ / FILS / FILSW)

If an interrupt request occurs during execution of a string instruction with a prefix code, the string instruction is executed falsely because the prefix becomes invalid for the string instruction after the interrupt is returned. Do not attach NCC prefix to any of the above string instructions.

- Flag change instructions (AND CCR,#imm8 / OR CCR,#imm8 / POPW PS)

The instruction is executed normally, but the prefix affects the next instruction.

- Interrupt instructions (INT #vct8 / INT9 / INT addr16 / INTP addr24 / RETI)

CCR changes according to the instruction specifications regardless of the prefix.



● JCTX @A

CCR changes according to the instruction specifications regardless of the prefix.

● MOV ILM,imm8

The instruction is executed normally, but the prefix affects the next instruction.

## 2.9 Interrupt Disable Instructions

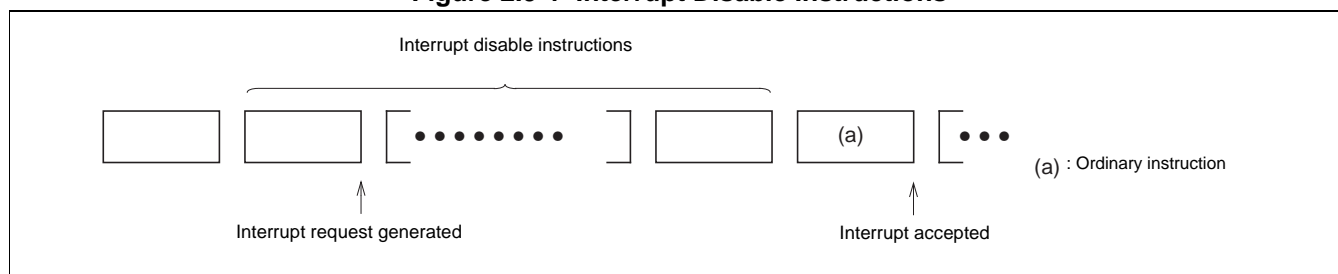
Interrupt requests are not accepted about following 10 instructions:

MOV ILM, #imm8	PCB	SPB	OR CCR, #imm8	NCC
AND CCR, #imm8	ADB	CMR	POPW PS	DTB

### ■ Interrupt Disable Instructions

As shown in Figure 2.9-1, if a valid hardware interrupt request occurs during execution of any of the above instructions, the interrupt can be processed only when an instruction other than the above is executed.

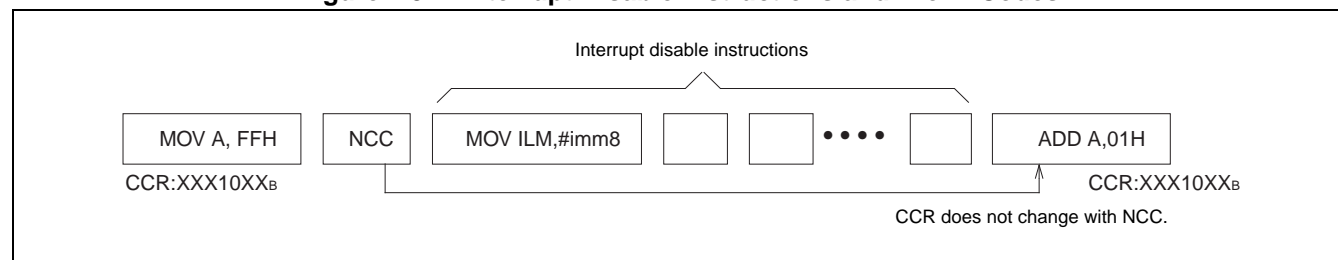
Figure 2.9-1 Interrupt Disable Instructions



### ■ Restrictions on Interrupt Disable Instructions and Prefix Instructions

As shown in Figure 2.9-2, when a prefix code is placed before an interrupt disable instruction, the prefix code affects the first instruction after the code other than the interrupt disable instruction.

Figure 2.9-2 Interrupt Disable Instructions and Prefix Codes

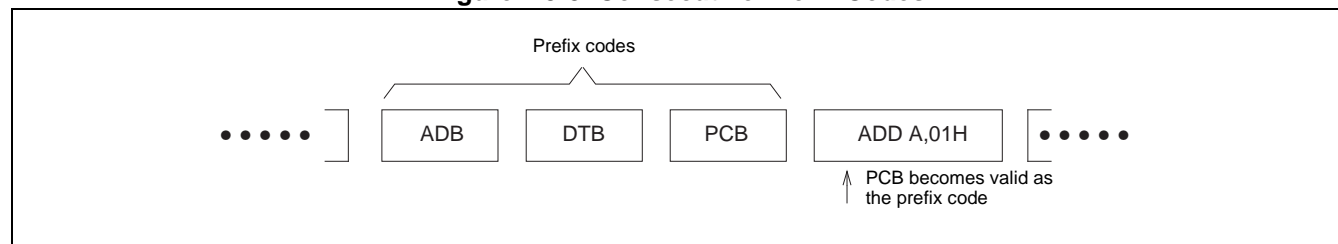


### ■ Consecutive Prefix Codes

As shown in Figure 2.9-3, when competitive prefix codes are placed consecutively, the latter one becomes valid.

Competitive prefix codes are PCB, ADB, DTB, and SPB.

Figure 2.9-3 Consecutive Prefix Codes





# CHAPTER 3

---

# INTERRUPT

**This chapter describes the interruption, extended intelligent I/O service (EI<sup>2</sup>OS), and direct memory access controller ( $\mu$ DMAC) of MB90330A Series.**

- 3.1 Outline of Interrupt
- 3.2 Interrupt Cause and Interrupt Vector
- 3.3 Interrupt Control Register and Peripheral Function
- 3.4 Hardware Interrupt
- 3.5 Software Interrupt
- 3.6 Interrupts by Extended Intelligent I/O Service (EI<sup>2</sup>OS)
- 3.7 Exception Processing Interrupt
- 3.8 Interruption by  $\mu$ DMAC
- 3.9 Exceptions
- 3.10 Stack Operation of Interrupt Processing
- 3.11 Program Example of Interrupt Processing
- 3.12 Delayed Interrupt Generation Module

## 3.1 Outline of Interrupt

F<sup>2</sup>MC-16LX has the following five interrupt functions, which suspend the current process when an event occurs and transfer control to a separately defined program.

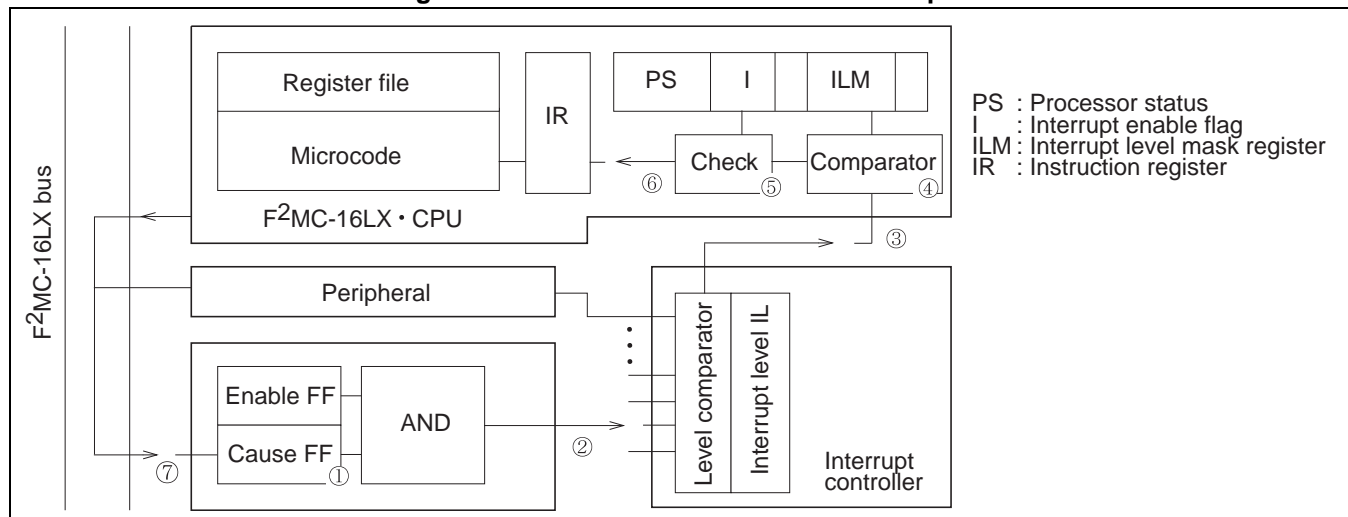
- Hardware Interrupt
- Software interrupt
- Interrupts by extended intelligent I/O service (EI<sup>2</sup>OS)
- Interruption by  $\mu$ DMAC
- Exception processing

### ■ Type and Function of Interrupt

#### ● Hardware Interrupt

Transfers control to the user-defined interrupt handling program in response to an interrupt request from a peripheral function.

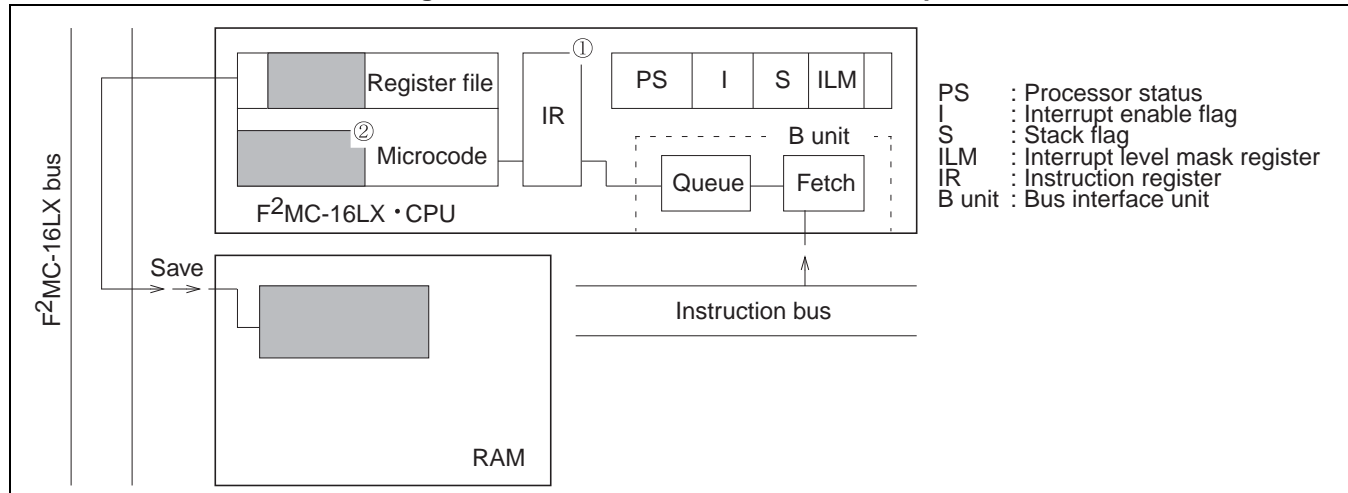
Figure 3.1-1 Overview of Hardware Interrupts



## ● Software interrupt

Transfers control to the user-defined interrupt handling program by executing the instruction dedicated to software interrupt (for example, INT instruction).

**Figure 3.1-2 Overview of Software Interrupts**

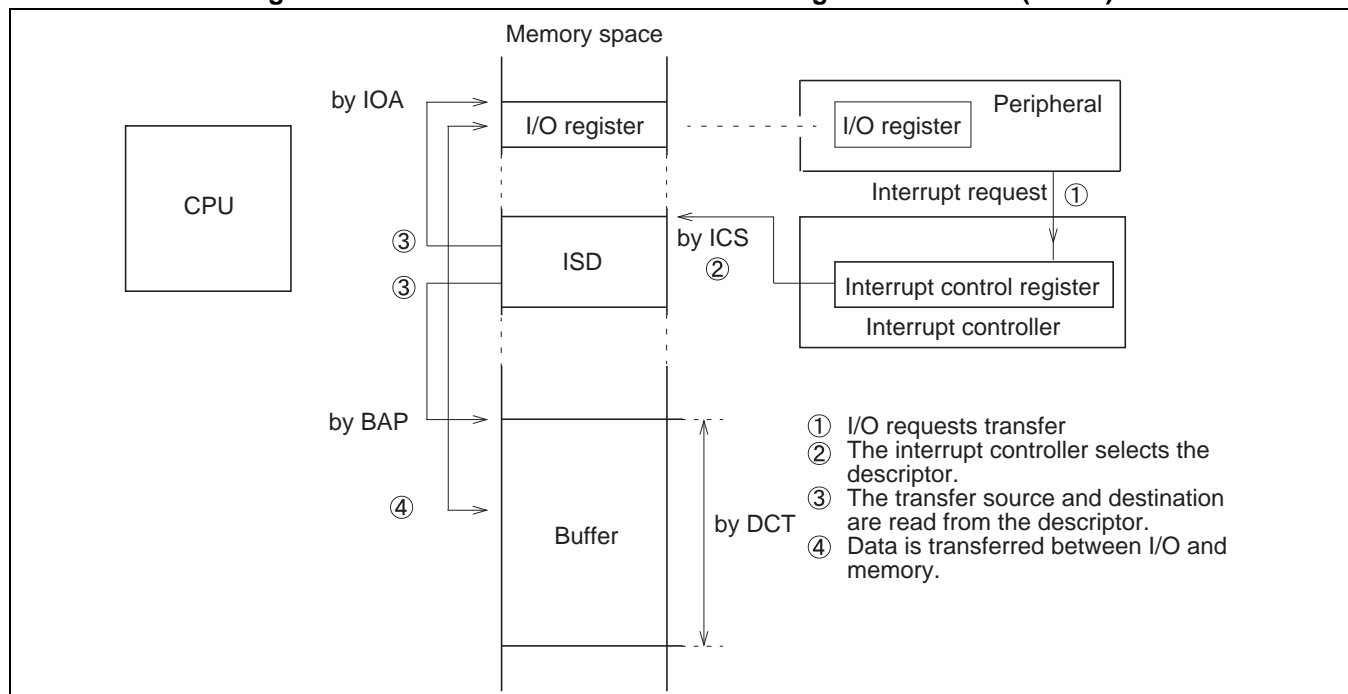


## ● Interrupts by extended intelligent I/O service (EI<sup>2</sup>OS)

EI<sup>2</sup>OS is involved in automatic data transfer between peripheral functions and memory. EI<sup>2</sup>OS is capable of performing data transfer like DMA (direct memory access) although it was previously performed by the interrupt handling program. Once the data transfer process has been performed the specified number of times, EI<sup>2</sup>OS automatically executes the interrupt handling program.

Interruption by EI<sup>2</sup>OS is a type of hardware interrupt.

**Figure 3.1-3 Overview of the Extended Intelligent I/O Service (EI<sup>2</sup>OS)**

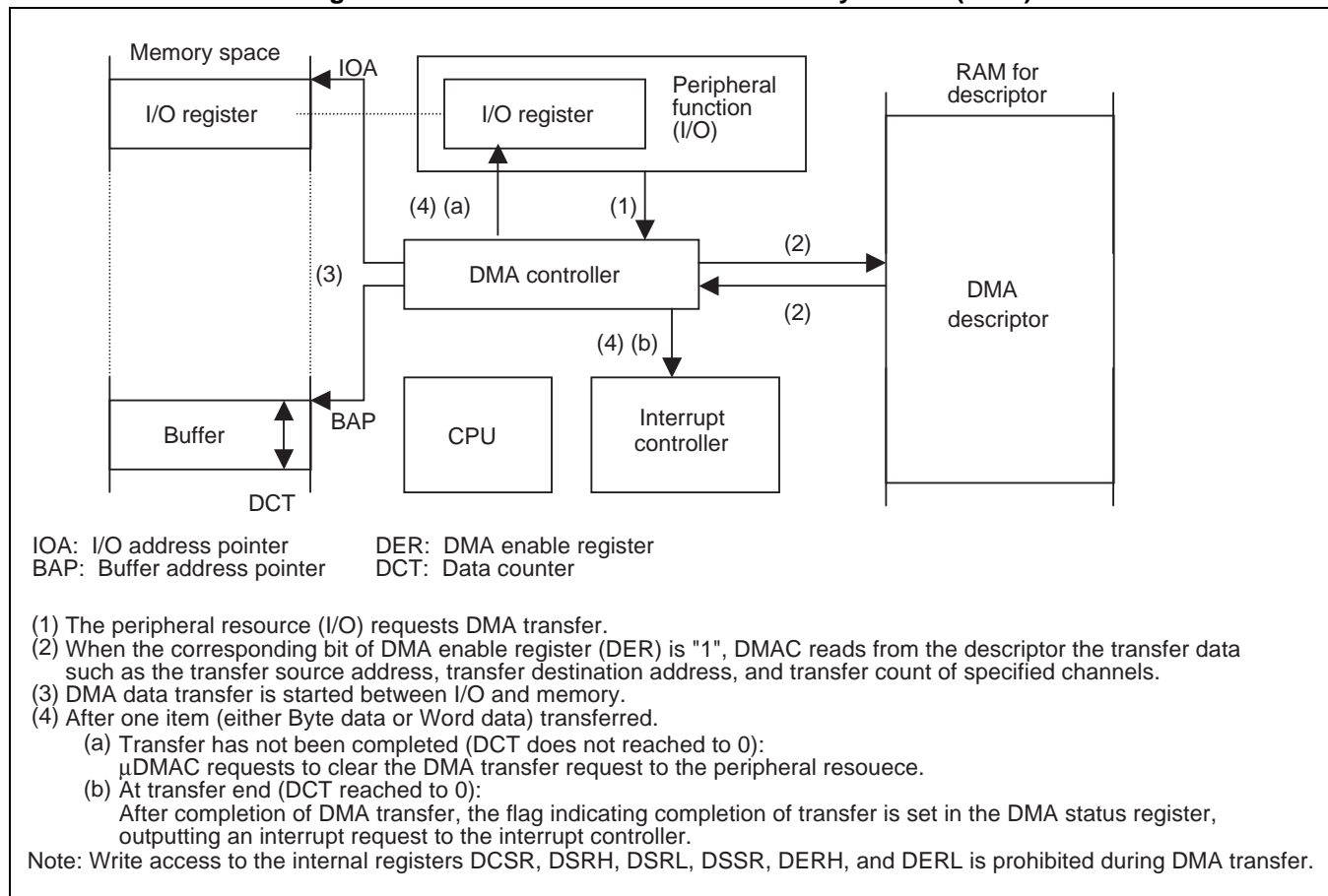


### ● Interruption by $\mu$ DMAC

$\mu$ DMAC is involved in automatic data transfer between peripheral functions and memory. EI<sup>2</sup>OS performs data transfer by DMA transfer although it was previously performed by the interrupt handling program. Once the data transfer process has been performed the specified number of times,  $\mu$ DMAC automatically executes the interrupt handling program.

Interruption by  $\mu$ DMAC is a type of hardware interrupt.

**Figure 3.1-4 Overview of the Direct Memory access (DMA)**



### ● Exception processing

Exception processing, basically the same as interrupt, is executed when an exception item (execution of an undefined instruction) is detected at an instruction-to-instruction boundary; the normal process is suspended for this purpose. Equivalent to software interrupt instruction "INT10".

## 3.2 Interrupt Cause and Interrupt Vector

F<sup>2</sup>MC-16LX has functions that are associated with 256 types of interrupt causes, and 256 interrupt vector tables are assigned to the most significant address area of memory. This interruption vector is shared by all the interruptions.

All of interrupt INT0 to INT255 are available for software interrupt, although some interrupt vectors are shared with hardware interrupt or exception processing interrupt. Furthermore, for hardware interrupt, the fixed interrupt vectors and interrupt control registers (ICR) are used for each of the peripheral functions.

### ■ Interrupt Vector

The interrupt vector table referenced during interrupt processing is assigned to the most significant of memory address area (FFFC00<sub>H</sub> to FFFFFFF<sub>H</sub>). Furthermore, the interrupt vectors share the same area with EI<sup>2</sup>OS,  $\mu$ DMAC, hardware interrupt, software interrupts, and exception processing. Table 3.2-1 shows the allocation of the interrupt numbers and interrupt vectors.

**Table 3.2-1 Interrupt Vector List**

Software interrupt instruction	Vector address L	Vector address M	Vector address H	Mode data	Interrupt number	Hardware interrupt
INT0	FFFFFC <sub>H</sub>	FFFFFD <sub>H</sub>	FFFFFE <sub>H</sub>	Unused	#0	None
:	:	:	:	:	:	:
INT7	FFFFE0 <sub>H</sub>	FFFFE1 <sub>H</sub>	FFFFE2 <sub>H</sub>	Unused	#7	None
INT8	FFFFDC <sub>H</sub>	FFFFDD <sub>H</sub>	FFFFDE <sub>H</sub>	FFFFDF <sub>H</sub>	#8	(RESET vector)
INT9	FFFFD8 <sub>H</sub>	FFFFD9 <sub>H</sub>	FFFFDA <sub>H</sub>	Unused	#9	None
INT10	FFFFD4 <sub>H</sub>	FFFFD5 <sub>H</sub>	FFFFD6 <sub>H</sub>	Unused	#10	<Exception processing>
INT11	FFFFD0 <sub>H</sub>	FFFFD1 <sub>H</sub>	FFFFD2 <sub>H</sub>	Unused	#11	Hardware Interrupt #0
INT12	FFFFCC <sub>H</sub>	FFFFCD <sub>H</sub>	FFFFCE <sub>H</sub>	Unused	#12	Hardware Interrupt #1
INT13	FFFFC8 <sub>H</sub>	FFFFC9 <sub>H</sub>	FFFFCA <sub>H</sub>	Unused	#13	Hardware Interrupt #2
INT14	FFFFC4 <sub>H</sub>	FFFFC5 <sub>H</sub>	FFFFC6 <sub>H</sub>	Unused	#14	Hardware Interrupt #3
:	:	:	:	:	:	:
INT254	FFFC04 <sub>H</sub>	FFFC05 <sub>H</sub>	FFFC06 <sub>H</sub>	Unused	#254	None
INT255	FFFC00 <sub>H</sub>	FFFC01 <sub>H</sub>	FFFC02 <sub>H</sub>	Unused	#255	None

Reference:

It is recommended that the unused interrupt vectors are set in the exception processing addresses, etc.



■ **Interrupt Factors, Interrupt Vectors, and Interrupt Control Registers**

Table 3.2-2 shows the relationship between the causes of interrupts except software interrupt, and the interrupt vectors and control registers.

**Table 3.2-2 Interrupt Factors, Interrupt Vectors, and Interrupt Control Registers**

Interrupt cause	EI <sup>2</sup> OS- corresponded	μDMAC- corresponded	Interrupt vector			Interrupt control registers		Priority *2
			Number		Address	ICR	Address	
Reset	×	×	#08	08 <sub>H</sub>	FFFFDC <sub>H</sub>	-	-	<div>High</div> <div>↑</div> <div>↓</div> <div>Low</div>
INT9 instruction	×	×	#09	09 <sub>H</sub>	FFFFD8 <sub>H</sub>	-	-	
Exception processing	×	×	#10	0A <sub>H</sub>	FFFFD4 <sub>H</sub>	-	-	
USB function 1	×	0,1	#11	0B <sub>H</sub>	FFFFD0 <sub>H</sub>	ICR00	0000B0 <sub>H</sub> <sup>*1</sup>	
USB function 2	×	2 to 6 <sup>*3</sup>	#12	0C <sub>H</sub>	FFFFCC <sub>H</sub>			
USB function 3	×	×	#13	0D <sub>H</sub>	FFFFC8 <sub>H</sub>	ICR01	0000B1 <sub>H</sub> <sup>*1</sup>	
USB function 4	×	×	#14	0E <sub>H</sub>	FFFFC4 <sub>H</sub>			
USB HOST 1	×	×	#15	0F <sub>H</sub>	FFFFC0 <sub>H</sub>	ICR02	0000B2 <sub>H</sub> <sup>*1</sup>	
USB HOST 2	×	×	#16	10 <sub>H</sub>	FFFFBC <sub>H</sub>			
I <sup>2</sup> C ch.0	×	×	#17	11 <sub>H</sub>	FFFFB8 <sub>H</sub>	ICR03	0000B3 <sub>H</sub> <sup>*1</sup>	
DTP/ External interruption ch.0/ch.1	○	×	#18	12 <sub>H</sub>	FFFFB4 <sub>H</sub>			
I <sup>2</sup> C ch.1	×	×	#19	13 <sub>H</sub>	FFFFB0 <sub>H</sub>	ICR04	0000B4 <sub>H</sub> <sup>*1</sup>	
DTP/ External interruption ch.2/ch.3	○	×	#20	14 <sub>H</sub>	FFFFAC <sub>H</sub>			
I <sup>2</sup> C ch.2	×	×	#21	15 <sub>H</sub>	FFFFA8 <sub>H</sub>	ICR05	0000B5 <sub>H</sub> <sup>*1</sup>	
DTP/ External interruption ch.4/ch.5	○	×	#22	16 <sub>H</sub>	FFFFA4 <sub>H</sub>			
PWC/reload timer ch.0	Δ	14	#23	17 <sub>H</sub>	FFFFA0 <sub>H</sub>	ICR06	0000B6 <sub>H</sub> <sup>*1</sup>	
DTP/ External interruption ch.6/ch.7	Δ	×	#24	18 <sub>H</sub>	FFFF9C <sub>H</sub>			
Input capture ch.0./ch.1	Δ	7	#25	19 <sub>H</sub>	FFFF98 <sub>H</sub>	ICR07	0000B7 <sub>H</sub> <sup>*1</sup>	
Reload timer ch.1	Δ	×	#26	1A <sub>H</sub>	FFFF94 <sub>H</sub>			
Input capture ch.2/ch.3	Δ	8	#27	1B <sub>H</sub>	FFFF90 <sub>H</sub>	ICR08	0000B8 <sub>H</sub> <sup>*1</sup>	
Reload timer ch.2	Δ	×	#28	1C <sub>H</sub>	FFFF8C <sub>H</sub>			
Output compare ch.0/ch.1	○	×	#29	1D <sub>H</sub>	FFFF88 <sub>H</sub>	ICR09	0000B9 <sub>H</sub> <sup>*1</sup>	
PPG ch.0/ch.1	×	×	#30	1E <sub>H</sub>	FFFF84 <sub>H</sub>			
Output compare ch.2/ch.3	○	×	#31	1F <sub>H</sub>	FFFF80 <sub>H</sub>	ICR10	0000BA <sub>H</sub> <sup>*1</sup>	
PPG ch.2/ch.3	×	×	#32	20 <sub>H</sub>	FFFF7C <sub>H</sub>			
UART transmission end ch.2/ch.3	○	11	#33	21 <sub>H</sub>	FFFF78 <sub>H</sub>	ICR11	0000BB <sub>H</sub> <sup>*1</sup>	
PPG ch.4/ch.5	×	×	#34	22 <sub>H</sub>	FFFF74 <sub>H</sub>			
UART1 reception ch.2/ch.3	⊙	10	#35	23 <sub>H</sub>	FFFF70 <sub>H</sub>	ICR12	0000BC <sub>H</sub> <sup>*1</sup>	
A/D conversion and free-run timer	Δ	15	#36	24 <sub>H</sub>	FFFF6C <sub>H</sub>			
UART1 transmission ch.0/ch.1	○	13	#37	25 <sub>H</sub>	FFFF68 <sub>H</sub>	ICR13	0000BD <sub>H</sub> <sup>*1</sup>	
Extended serial I/O	×	9	#38	26 <sub>H</sub>	FFFF64 <sub>H</sub>			
UART1 reception ch.0/ch.1	⊙	12	#39	27 <sub>H</sub>	FFFF60 <sub>H</sub>	ICR14	0000BE <sub>H</sub> <sup>*1</sup>	
Time-base timer/Watch timer	×	×	#40	28 <sub>H</sub>	FFFF5C <sub>H</sub>			
Writing and Erasing Flash Memory	×	×	#41	29 <sub>H</sub>	FFFF58 <sub>H</sub>	ICR15	0000BF <sub>H</sub> <sup>*1</sup>	
Delay interruption generation module	×	×	#42	2A <sub>H</sub>	FFFF54 <sub>H</sub>			

- ⊙: Available. With EI<sup>2</sup>OS stop function. (The interrupt request flag is cleared by the interrupt clear signal. With a stop request.)
- : Available. (The interrupt request flag is cleared by the interrupt clear signal.)
- Δ: Available when not using the interrupt factor shared with ICR
- ×: Not available
- \*1: The interrupt levels for the peripheral functions sharing the ICRs are identical.
- \*2: The priority is given when interrupts with the same level are generated simultaneously.
- \*3: Ch.2 and ch.3 can use even when USB HOST is operated.

## Notes:

- If use of EI<sup>2</sup>OS is permitted when a same interrupt control register (ICR) has two interrupt causes, EI<sup>2</sup>OS is activated when either of interrupt causes is detected. Any interrupt which is due to a non-activation cause is masked during activation of EI<sup>2</sup>OS. It is therefore recommended that either of the interrupt requests be masked while EI<sup>2</sup>OS is in use.
- In a resource for which two interrupt causes exist in a same interrupt control register (ICR), the interrupt flag is cleared by the EI<sup>2</sup>OS interrupt clear signal.
- If two interrupt causes were found at a single interrupt number, both the interrupt request flags in the resource are cleared by the μDMAC interrupt clear signal. Thus, if the μDMAC function is used for one of the two causes, the other interrupt function will not be available. Set the interrupt request permission bit to "0" in the appropriate resource, and take measures by software polling processing.

## ■ Type and Function of USB interrupt

USB interrupt factor	Details
USB function 1	End point0-IN End Point0-OUT
USB function 2	End Point1-5*
USB function 3	SUSP SOF BRST WKUP CONF
USB function 4	SPK
USB HOST 1	DIRQ CNNIRQ URIRQ RWKIRQ
USB HOST 2	SOFIRQ CMPIRQ

\*: End Point 1, 2 can use even when USB HOST is operated.

### 3.3 Interrupt Control Register and Peripheral Function

Interrupt control registers (ICR00 to ICR15) located in the interrupt controller, are associated with all the peripheral functions which have the interrupt function. This register controls the interrupt and the extended intelligent I/O service (EI<sup>2</sup>OS).

#### ■ Interrupt Control Register List

Table 3.3-1 shows the list of the peripheral functions associated with the interrupt control registers.

**Table 3.3-1 Interrupt Control Register List**

Address	Registers	Abbreviation	Corresponding peripheral function
0000B0 <sub>H</sub>	Interrupt control registers 00	ICR00	USB function 1 and USB function 2
0000B1 <sub>H</sub>	Interrupt control registers 01	ICR01	USB function 3 and USB function 4
0000B2 <sub>H</sub>	Interrupt control registers 02	ICR02	USB HOST 1 and USB HOST 2
0000B3 <sub>H</sub>	Interrupt control registers 03	ICR03	I <sup>2</sup> C ch.0, DTP External interruption ch.0/ch.1
0000B4 <sub>H</sub>	Interrupt control registers 04	ICR04	I <sup>2</sup> C ch.1, DTP External interruption ch.2/ch.3
0000B5 <sub>H</sub>	Interrupt control registers 05	ICR05	I <sup>2</sup> C ch.2, DTP External interruption ch.4/ch.5
0000B6 <sub>H</sub>	Interrupt control registers 06	ICR06	PWC, reload timer ch.0, DTP external interruption ch.6/ch.7
0000B7 <sub>H</sub>	Interrupt control registers 07	ICR07	Input capture ch.0/ch.1 and reload timer ch.1
0000B8 <sub>H</sub>	Interrupt control registers 08	ICR08	Input capture ch.2/ch.3, and reload timer ch.2
0000B9 <sub>H</sub>	Interrupt control registers 09	ICR09	Output compare ch.0/ch.1, PPG ch.0/ch.1
0000BA <sub>H</sub>	Interrupt control registers 10	ICR10	Output compare ch.2/ch.3, PPG ch.2/ch.3
0000BB <sub>H</sub>	Interrupt control registers 11	ICR11	UART Transmission ch.2/ch.3, PPG ch.4/ch.5
0000BC <sub>H</sub>	Interrupt control registers 12	ICR12	UART reception ch.2/ch.3, AD conversion, and free-run timer
0000BD <sub>H</sub>	Interrupt control registers 13	ICR13	UART transmission ch.0/ch.1 and extended serial I/O
0000BE <sub>H</sub>	Interrupt control registers 14	ICR14	UART reception ch.0/ch.1, timer base timer, and watch timer
0000BF <sub>H</sub>	Interrupt control registers 15	ICR15	Flash writing and delay interruption generation module

### ■ Interrupt Control Register Functions

Each of the interrupt control register (ICR) has the following four functions.

- Setting of interrupt level for peripheral function
- Selection of whether to perform normal interrupt or external intelligent for corresponding peripheral function (EI<sup>2</sup>OS)
- Select the channel of Extended Intelligent I/O service (EI<sup>2</sup>OS)
- Display of extended intelligent I/O service (EI<sup>2</sup>OS) status

For the interrupt control registers (ICRs), some functions are different between write and read. Details are shown in Figure 3.1-1 and Figure 3.1-2 in Section "3.3.1 Interrupt Control Registers (ICR00 to ICR15)".

---

#### Note:

Do not use any read modify write instruction to access the interrupt control register (ICR). An attempt of this may cause a malfunction.

---

### 3.3.1 Interrupt Control Registers (ICR00 to ICR15)

Interrupt control registers (ICR00 to ICR15) associated with all the peripheral functions provided with the interrupt function, controls the handling which takes place when an interrupt request is generated. Some functions of the registers are different between write and read.

#### ■ Interrupt Control Registers (ICR00 to ICR15)

Figure 3.3-1 Interrupt Control Register (ICR00 to ICR15) at Writing

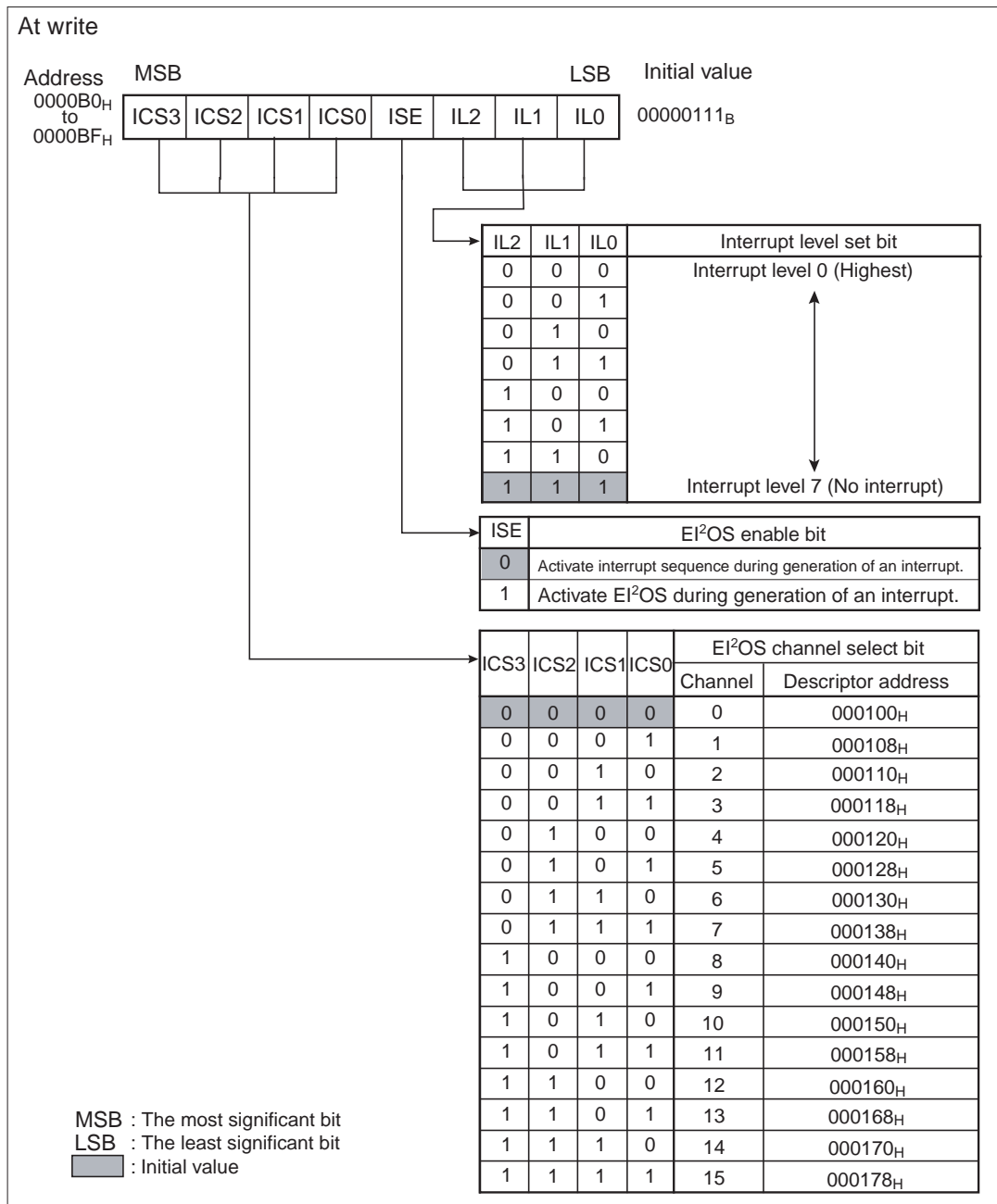
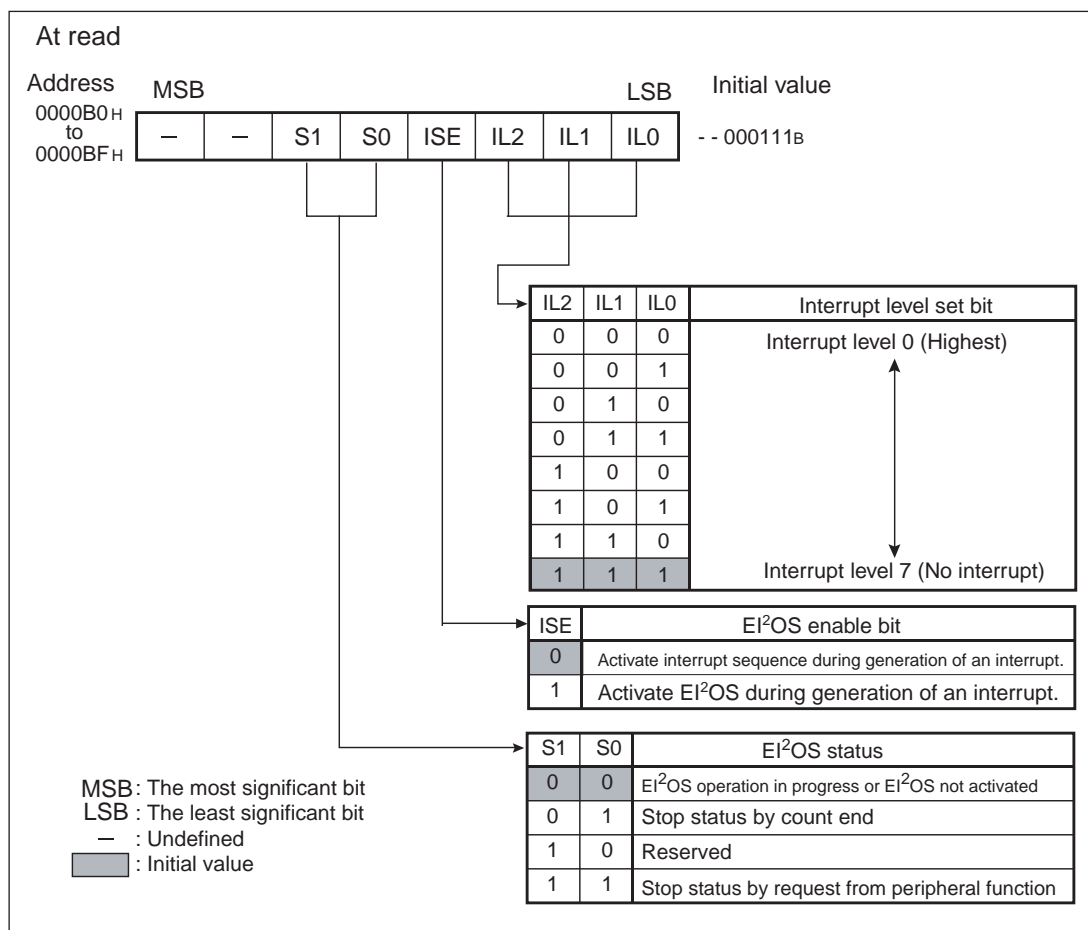


Figure 3.3-2 Interrupt Control Register (ICR00 to ICR15) at Read



### 3.3.2 Interrupt Control Register Functions

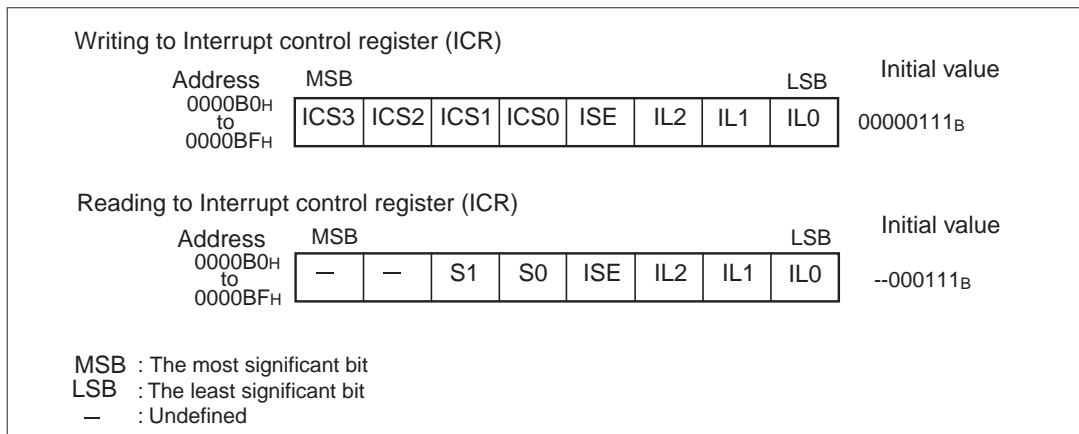
Each of the interrupt control register (ICR00 to ICR15) consists of the following bits, which have four functions.

- Interrupt level set bit (IL2 to IL0)
- EI<sup>2</sup>OS enable bit (ISE)
- EI<sup>2</sup>OS channel select bits (ICS3 to ICS0)
- EI<sup>2</sup>OS status bit (S1, S0)

#### ■ Configuration of Interrupt Control Register (ICR)

Figure 3.3-3 shows the bit configuration of the interrupt control register (ICR).

**Figure 3.3-3 Configuration of Interrupt Control Register (ICR)**



#### References:

- ICS3 to ICS0 bit are enabled only when starting the extended intelligent I/O service (EI<sup>2</sup>OS). If EI<sup>2</sup>OS is to be activated, set the ISE bit to "1". Otherwise, set it to "0". When you do not start EI<sup>2</sup>OS, you can not set ICS3 to ICS0.
- ICS1 and ICS0 are enabled only for write. S1 and S0 are enabled only for read.

#### Note:


Reading the upper rank two-bit is an irregular value.

## ■ Interrupt Control Register Functions

### ● Interrupt level set bit (IL2 to IL0)

Specifies the interrupt level for the associated peripheral function. Initialized to level 7 (no interrupts) by reset. Table 3.3-2 shows the relationship between the interrupt level set bits and each interrupt level.

**Table 3.3-2 Correspondence between Interrupt Level Set Bits and Interrupt Levels**

IL2	IL1	IL0	Interrupt level
0	0	0	<div style="text-align: center;">           0 (highest interrupt)              6 (lowest interrupt)         </div>
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	6 (lowest interrupt)
1	1	1	
1	1	1	7 (No interrupt)

### ● EI<sup>2</sup>OS enable bit (ISE)

If the ISE bit is "1" during generation of an interrupt request, EI<sup>2</sup>OS is activated; if it is "0", the interrupt sequence will be activated. Furthermore, when the end condition of EI<sup>2</sup>OS is satisfied, (both the S1 and S0 bits are other than "00<sub>B</sub>"), the ISE bit is cleared. If the associated peripheral function does not have the EI<sup>2</sup>OS function, the ISE bit must have been set to "0" by software. The ISE bit is initialized to "0" by reset by "0000<sub>B</sub>".

### ● EI<sup>2</sup>OS channel select bits (ICS3 to ICS0)

Specifies the channel of EI<sup>2</sup>OS with a write only bit. The address of the EI<sup>2</sup>OS descriptor is determined by the value set here. The ICS bit is initialized to "0000<sub>B</sub>" by reset by. Table 3.3-3 shows the correspondence between the EI<sup>2</sup>OS channel select bits and descriptor addresses.



**Table 3.3-3 Correspondence between EI<sup>2</sup>OS Channel Select Bits and Descriptor Addresses**

ICS3	ICS2	ICS1	ICS0	Channel to be selected	Descriptor address
0	0	0	0	0	000100 <sub>H</sub>
0	0	0	1	1	000108 <sub>H</sub>
0	0	1	0	2	000110 <sub>H</sub>
0	0	1	1	3	000118 <sub>H</sub>
0	1	0	0	4	000120 <sub>H</sub>
0	1	0	1	5	000128 <sub>H</sub>
0	1	1	0	6	000130 <sub>H</sub>
0	1	1	1	7	000138 <sub>H</sub>
1	0	0	0	8	000140 <sub>H</sub>
1	0	0	1	9	000148 <sub>H</sub>
1	0	1	0	10	000150 <sub>H</sub>
1	0	1	1	11	000158 <sub>H</sub>
1	1	0	0	12	000160 <sub>H</sub>
1	1	0	1	13	000168 <sub>H</sub>
1	1	1	0	14	000170 <sub>H</sub>
1	1	1	1	15	000178 <sub>H</sub>

● EI<sup>2</sup>OS status bits (S1 and S0)

It is a bit only for reading. By examining the value at the end of EI<sup>2</sup>OS operation, the operating state and/or termination status can be determined. The bit is initialized to "00<sub>B</sub>" at a reset. Table 3.3-4 shows the relationships between the S0/S1 bit and the EI<sup>2</sup>OS status.

**Table 3.3-4 Relation between EI<sup>2</sup>OS Status Bits and EI<sup>2</sup>OS Status**

S1	S0	EI <sup>2</sup> OS status
0	0	When EI <sup>2</sup> OS in operation or not started.
0	1	Stop state by end of counting
1	0	Reserved
1	1	Stop state by request from peripheral function

## 3.4 Hardware Interrupt

---

Hardware interrupt suspends the active program execution by the CPU in response to an interrupt request signal generated by a peripheral function, resulting in transfer of control to the user-defined interrupt handling program. Extended intelligent I/O service (EI<sup>2</sup>OS),  $\mu$ DMAC, external interrupts, and other similar processes are also executed as a type of hardware interrupt.

---

### ■ Function of Hardware Interrupt

#### ● Function of hardware interrupt

Hardware interrupt makes comparison between the interrupt level of the interrupt request signal which a peripheral function outputs and the interrupt level mask register (ILM) in the CPU's processor status (PS). This interrupt also refers the contents of the I flag in PS, by hardware, to determine whether the interrupt is acceptable.

Once the hardware interrupt is accepted, the contents of the registers and their related data in the CPU are automatically saved in the system stack. The level of the currently requested interrupt is stored in the ILM. Then, control branches to the associated interrupt vector.

#### ● Multiple interrupts

Multiple hardware interrupts can be activated.

#### ● EI<sup>2</sup>OS

At the completion of transfer, a hardware interrupt is activated although EI<sup>2</sup>OS is an automatic transfer function between EI<sup>2</sup>OS/memory and I/O. Further, EI<sup>2</sup>OS cannot be activated in a multiple manner. While an EI<sup>2</sup>OS process is in progress, all the other interrupt requests and  $\mu$ DMAC requests remain pending.

#### ● $\mu$ DMAC

At the completion of transfer, a hardware interrupt is activated although  $\mu$ DMAC is an automatic transfer function between memory and I/O. Further,  $\mu$ DMAC cannot be activated in a multiple manner. While an  $\mu$ DMAC process is in progress, all the other interrupt requests and EI<sup>2</sup>OS requests remain pending.

#### ● External interrupt

The external interrupt (including a wake-up interrupt) is accepted as a hardware interrupt through the peripheral function (its interrupt request detector circuit).

#### ● Interrupt vector

The interrupt vector table, referred during execution of the interrupt process, is assigned to the "FFFC00<sub>H</sub>" to "FFFFFF<sub>H</sub>" memory for shared with software interrupt. For allocations of interrupt numbers and interrupt vectors, see Section "3.2 Interrupt Cause and Interrupt Vector".

### ■ Construction of Hardware Interrupt

As shown in Table 3.4-1, there are four features related to hardware interrupt. These four must be programmed when hardware interrupt is used.

**Table 3.4-1 Mechanism Related to Hardware Interrupt**

	Mechanism related to hardware interrupt	Functions
Peripheral function	Interrupt enable bits, interrupt request bits	Controls interrupt request from peripheral function
Interrupt controller	Interrupt control registers (ICR)	Setting interrupt levels
CPU	Interrupt enable flag (I)	Identification of interrupt enable state
	Interrupt level mask register (ILM)	Compares request interrupt level and current interrupt level
	Microcode	Execution of interruption handling routine
FFFC00 <sub>H</sub> to FFFFFFF <sub>H</sub> in memory	Interrupt vector table	Stores the branch destination address at interrupt processing

### ■ Hardware Interrupt Suppression

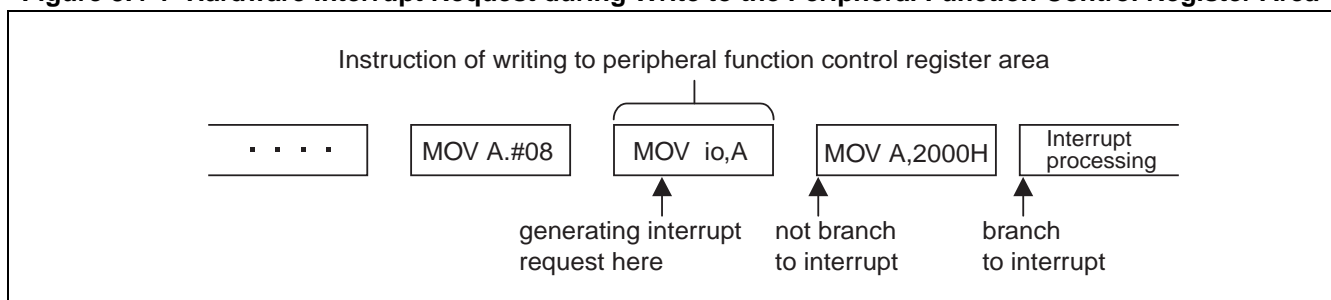
For hardware interrupt, acceptance of the interrupt request is suppressed in the following conditions:

- Suppressing a hardware interrupt which is generated during write to a peripheral function control register area

While data is being written to a peripheral function control register area, no hardware interrupt request is accepted. The purpose of this is to prevent the CPU from causing a malfunction due to some interrupt-related problem in response to an interrupt request which is generated while the interrupt control register relation for each resource is being rewritten. The peripheral function control register area is the area assigned to the control and data registers of the peripheral function control registers. Note that they are not the "000000<sub>H</sub>" to "0000FF<sub>H</sub>" I/O addressing area.

Figure 3.4-1 shows the hardware interrupt operation which takes place during write to the peripheral function control register area.

**Figure 3.4-1 Hardware Interrupt Request during Write to the Peripheral Function Control Register Area**



- Hardware interrupt suppression of interrupt suppression instruction

Table 3.4-2 shows the hardware interrupt suppression instruction. If the hardware interrupt request is generated during execution of hardware interrupt suppression instruction, an interrupt is processed after execution of hardware interrupt suppression instruction and then other instruction.

**Table 3.4-2 Hardware Interrupt Suppression Instruction**

	Prefix code	Interruption/holding control instruction (The command that delays the effect of prefix code)
The instructions which do not accept the interrupt and hold requests.	PCB DTB ADB SPB CMR NCC	MOV ILM,#imm8 OR CCR,#imm8 AND CCR,#imm8 POPW PS

- Suppressing a hardware interrupt during execution of a software interrupt

When a software interrupt is activated, no other interrupt requests are acceptable to clear the I flag to "0".

### **3.4.1 Operation of Hardware Interrupt**

---

**The following describes the operation sequence from generation of a hardware interrupt request to completion of interrupt handling.**

---

#### **■ Start of Hardware Interrupt**

- Operation of peripheral function (generation of interrupt request)

Any peripheral function provided with the hardware interrupt request function has "interrupt request flag" and "interrupt enable flag". The interrupt request flag indicates whether an interrupt request has been generated or not. The interrupt enable flag indicates whether an interrupt request is enabled or disabled. The interrupt request flag is set by occurrence of a specific event to the peripheral function. It causes an interrupt request to the interrupt controller when the interrupt enable flag indicates "enable".

- Operation of Interrupt controller (Control of interrupt request)

The interrupt controller makes a comparison between interrupt levels (ILs) of the simultaneously received interrupt requests. It adopts the highest-level request, (the request with the smallest IL value), and notifies it to the CPU. If two or more requests at the same level are found, the highest priority is given to the request with the smallest interrupt number.

- CPU operation (Interrupt request acceptance and interrupt processing)

The CPU compares the received interrupt level (IL2 to IL0 of ICR) with the contents of the interrupt level mask (ILM) register. If  $IL < ILM$  and the interrupt has been enabled (CCR I = 1), the currently active instruction terminates, the interrupt handling microcode is then activated to execute the interrupt handling.

First, the interrupt handling saves the contents of the dedicated registers (12 bytes of A, DPR, ADB, DTB, PCB, PC, and PS) in the system stack (system stack space indicated by SSB and SSP). Next, the interrupt handling loads to the program counters (PCB and PC) of the interrupt vector, and updates the ILM. It also sets the stack flag (S) (setting CCR:S = 1 to enable the system stack).

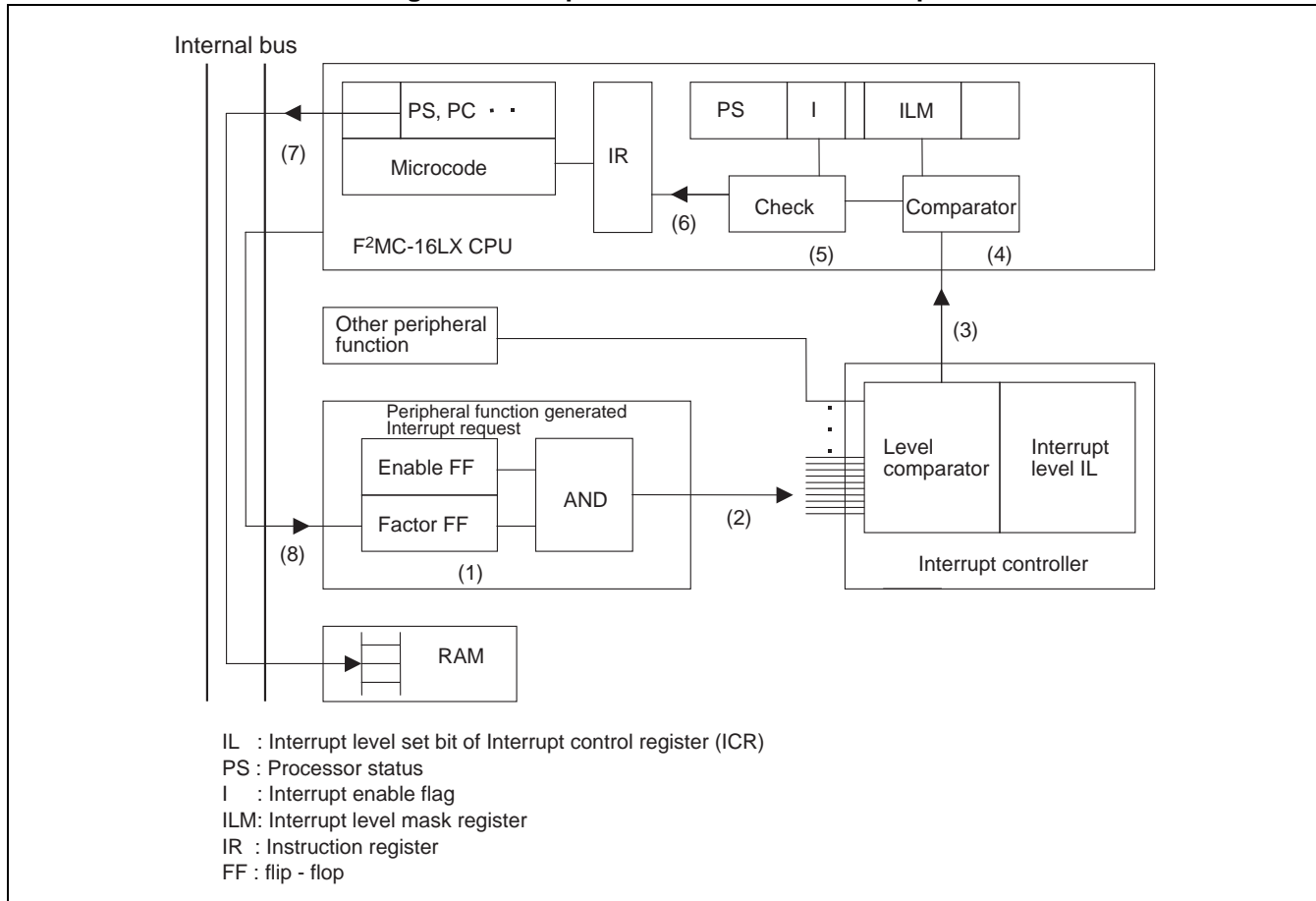
#### **■ Return from Hardware Interrupt**

When the interrupt request flag of the peripheral function causing the interrupt is cleared, and the RETI instruction is executed in the interrupt handling program, the 12-byte data saved in the system stack returns to the dedicated registers. Control then returns to the process which was being executed before the interrupt branch. By clearing the interrupt request flag, the interrupt request which the peripheral function has output to the interrupt controller is canceled automatically.

## ■ Operation of Hardware Interrupt

Figure 3.4-2 shows the operation sequence from generation of a hardware interrupt to completion of interrupt handling.

**Figure 3.4-2 Operation of Hardware Interrupt**



- (1) An interrupt cause occurs in a peripheral function.
- (2) The peripheral function interrupt enable bit is referred. If it indicates "enable", an interrupt request is output from the periphery to the interrupt controller.
- (3) After receiving the interrupt request, the interrupt controller determines the priorities of the simultaneously received interrupt requests. Then, the controller sends the ILs associated with the interrupt requests to the CPU.
- (4) The CPU compares the interrupt level requested from the interrupt controller with the interrupt level mask register (ILM).
- (5) If the comparison reveals that the priority is higher than the current interrupt handling level, the contents of the I flag of the condition code register (CCR) will be checked.
- (6) If the check in Item (5) reveals that the I flag indicates "enable", that is, I = 1, control waits for completion of the currently active instruction. The requested IL is set in the ILM on completion.
- (7) The contents of the register are saved before control branches to the interrupt handling routine.
- (8) The software in the interrupt handling routine clears the interrupt cause generated at 1 and executes the RETI instruction to complete the interrupt handling.

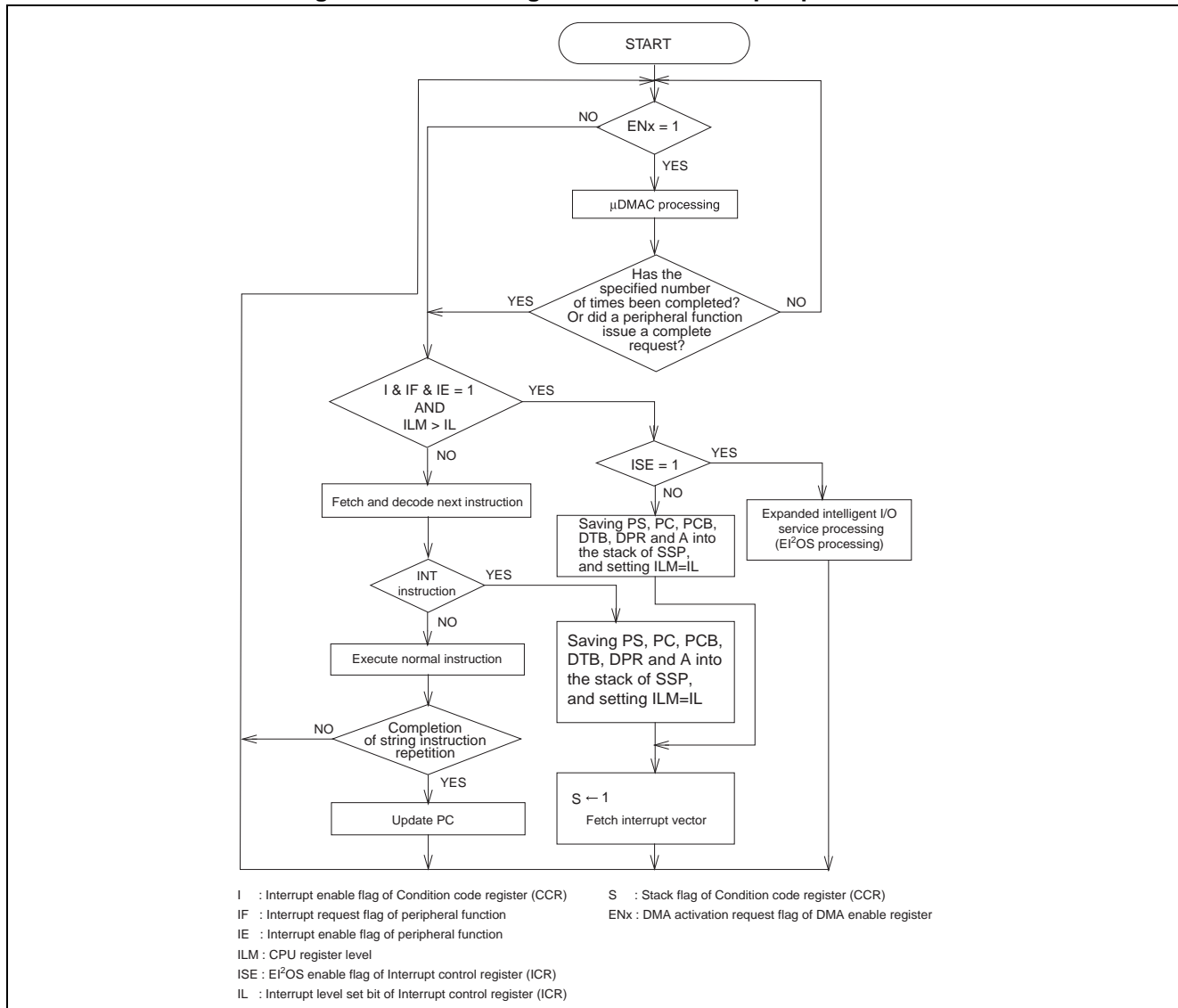
### 3.4.2 Operation Flow of Hardware Interrupt

When the peripheral function generates an interrupt request, the interrupt controller notifies the CPU of the interrupt level. If the CPU is ready to accept the interrupt, it suspends the currently active instruction; the CPU then executes the interrupt handling routine or activates Extended Intelligent I/O service (EI<sup>2</sup>OS)  $\mu$ DMAC. Furthermore, if a software interrupt is generated by the INT instruction, the interrupt handling routine is executed independently of the CPU state. At this time, the hardware interrupt is prohibited.

#### ■ Operation Flow of Hardware Interrupt

Figure 3.4-3 shows the handling flow which takes place during interrupt operation.

Figure 3.4-3 Handling Flow in the Interrupt Operation



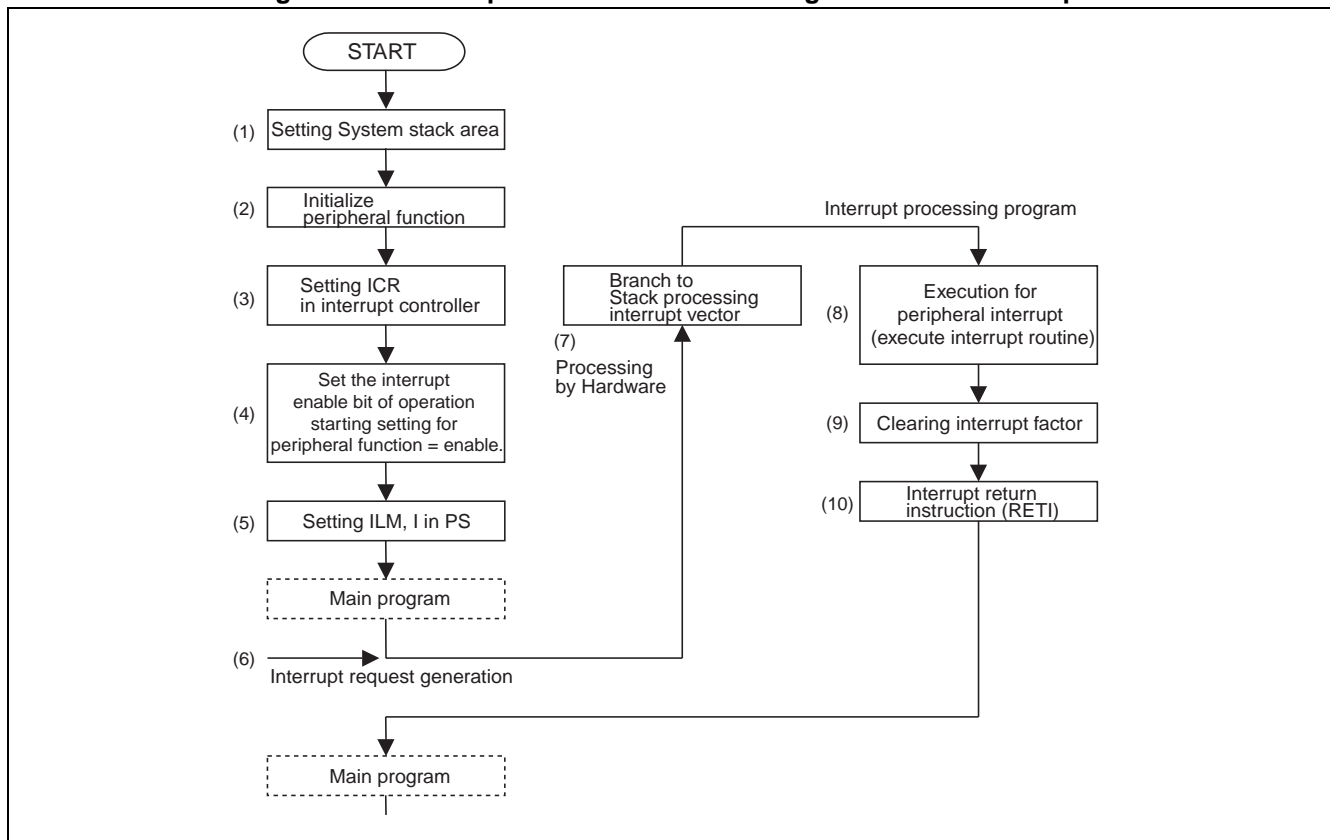
### 3.4.3 Procedure for Using a Hardware Interrupt

Use of hardware interrupt requires the system stack area, peripheral functions, and interrupt control registers (ICR) to be set up.

#### ■ Procedure for Using a Hardware Interrupt

Figure 3.4-4 shows an example of the procedure for using a hardware interrupt.

**Figure 3.4-4 Example of Procedure for Using a Hardware Interrupt**



- (1) The system stack area is set.
- (2) Initialize the setting of a peripheral function which can generate an interrupt request.
- (3) Set the interrupt control register (ICR) in the interrupt controller.
- (4) Make the peripheral function ready to start, and set the interrupt enable bit to "enable".
- (5) Sets the interrupt level mask register (ILM) and interrupt enable flag (I) to accept interrupts.
- (6) Generation of an interrupt in the peripheral function generates a hardware interrupt request.
- (7) The contents of the registers are saved by the interrupt handling hardware, and control branches to the interrupt handling program.
- (8) The interrupt handling program handles to the peripheral function in response to generation of the interrupt.
- (9) Clear the interrupt request from the peripheral function.
- (10) Execute the interrupt return instruction to return control to the program before branching.



### 3.4.4 Multiple Interrupts

---

**Multiple hardware interrupts can be implemented. To do so, set different interrupt levels for interrupt level set bits (IL0 to IL2) of the ICR in response to two or more interrupt requests from a peripheral function. However, multiple EI<sup>2</sup>OS and multiple  $\mu$ DMAC cannot be started.**

---

#### ■ Multiple Interrupts

If an interrupt request with a higher interrupt level is generated during execution of the interrupt handling routine, this higher interrupt level request is accepted with the current interrupt handling suspended. After completion of the higher interrupt level, control returns to the handling of the suspended interrupt. The interrupt level can be set to "0" to "7". If it is set to "7", the CPU will accept no interrupt request.

If a new interrupt at the same or a lower level is generated during execution of the current interrupt handling, the new one remains pending until the current one is completed, unless the I flag changes the ILM. Activation of the multiple interrupt function during the interrupt is temporarily disabled by setting the I flag in the condition code register (CCR) to "disable" (CCR:I = 0), or setting the interrupt level mask register (ILM) to "disable" (ILM = 000<sub>B</sub>) in the interrupt handling routine.

---

Note:

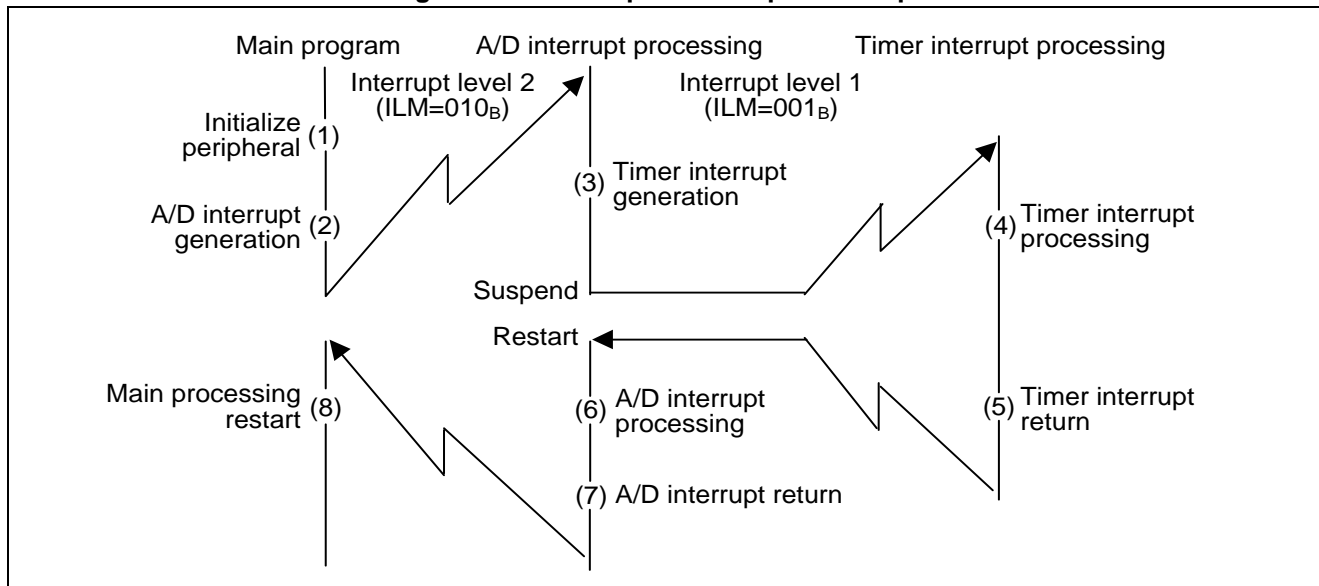
However, multiple EI<sup>2</sup>OS and multiple  $\mu$ DMAC cannot be started. If an extended intelligent I/O service (EI<sup>2</sup>OS) or  $\mu$ DMAC process is in progress, all the other interrupt requests and all the EI<sup>2</sup>OS and  $\mu$ DMAC requests remain pending.

---

## ■ Example of Multiple Interrupts

As an example of multiple interrupt processing, set the A/D converter interrupt level to 2 and the timer interrupt level to "1", considering a case when timer interrupts are to be given higher priority than A/D converter interrupts. At this time, the handling is executed as shown in Figure 3.4-5 if a timer interrupt is generated during the A/D converter interrupt handling.

**Figure 3.4-5 Example of Multiple Interrupts**



### ● A/D Interruption generation

At the start of the A/D converter interrupt handling, the ILM is automatically set to the same value as the A/D converter interrupt level (IL2 to IL0 in ICR). If a Level 1 or 0 interrupt request is generated, the interrupt handling makes it a higher priority in execution.

### ● End of interrupt processing

When the interrupt processing has completed and the return instruction (RETI) is executed, the values of the dedicated registers (A, DPR, ADB, DTB, PCB, PC, PS) saved in the stack are returned and the interrupt level mask register (ILM) has the values before the interrupt.

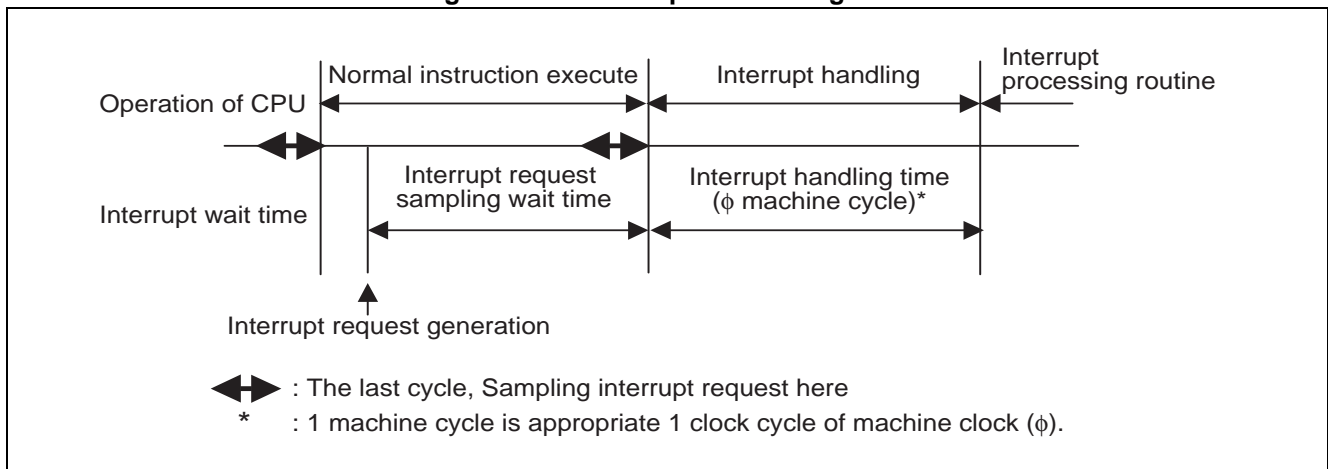
### 3.4.5 Hardware Interrupt Processing Time

Before the interrupt handling routine can be executed after a hardware interrupt request is generated, the time to complete of the currently active instruction and the interrupt handling time are required.

#### ■ Hardware Interrupt Processing Time

Before the interrupt handling routine can be executed after an interrupt request is generated and the interrupt is accepted, the waiting time for the interrupt request sample and the interrupt handling time (required for preparation for interrupt handling) are required. Figure 3.4-6 shows the interrupt handling time.

Figure 3.4-6 Interrupt Processing Time



#### ● Interrupt request sampling wait time

The wait time for an interrupt request sample refers to the time from generation of an interrupt request to the completion of the currently active instruction. Whether or not an interrupt request is present is determined by interrupt request sampling in the final cycle of each instruction. Because the CPU cannot recognize the interrupt request for the above reason, the wait time is produced.

The wait time for an interrupt request sample reaches the maximum if an interrupt request is generated immediately after the start of the PCPW, PW0,..., RW7 instructions with the longest cycle of execution (45 machine cycle).

● Interrupt handling time ( $\theta$  machine cycle)

The CPU must save the dedicated registers in the system stack, fetch the interrupt vectors, and execute other processes by acceptance of the interrupt request. To do so, it requires the interrupt handling time with the  $\theta$  machine cycles. The interrupt handling time can be calculated by the expression shown below.

- When an interrupt is activated:  $\theta = 24 + 6 \times Z$  machine cycles
- When an interrupt is returned:  $\theta = 11 + 6 \times Z$  machine cycles (RETI instructions)

The interrupt handling time depends on the address to which the stack pointer points. Table 3.4-3 shows the compensation values (Z) of the interrupt handling time.

One machine cycle is equal to one clock cycle of the machine clock ( $\phi$ ).

**Table 3.4-3 Compensation Value of Interrupt Handling Time (Z)**

Address which stack pointer indicates	Compensation value (Z)
For the external 8-bit	+4
For the external even address	+1
For the external odd address	+4
For internal even address	0
For internal odd address	+2

## 3.5 Software Interrupt

---

The software interrupt function transfers control from the currently active program by the CPU to the user-defined interrupt handling program in response to execution of the software interrupt instruction (INT instruction).

The hardware interrupt stops during execution of a software interrupt.

---

### ■ Start of Software Interrupt

#### ● Start of software interrupt

A software interrupt is started by using the INT instruction. The software interrupt request has neither the interrupt request flag nor enable flag, execution of the INT instruction always generates a software interrupt request.

#### ● Hardware interrupt inhibition

Because the INT instruction has no interrupt level, the interrupt level mask register (ILM) is not updated. During the execution of an INT instruction, "0" is set in the I flag of the condition code register (CCR) to mask hardware interrupts. To enable hardware interrupts also during software interrupt handling, set the I flag to "1" in the software interrupt handling routine.

#### ● Operation of software interrupt

Once the CPU fetches and executes the INT instruction, the software interrupt handling microcode is activated. With this microcode, the registers and their related data in the CPU are automatically saved in the system stack. After hardware interrupts are masked (CCR:I = 0), control branches to the associated interrupt vector.

For allocations of interrupt numbers and interrupt vectors, see Section "3.2 Interrupt Cause and Interrupt Vector".

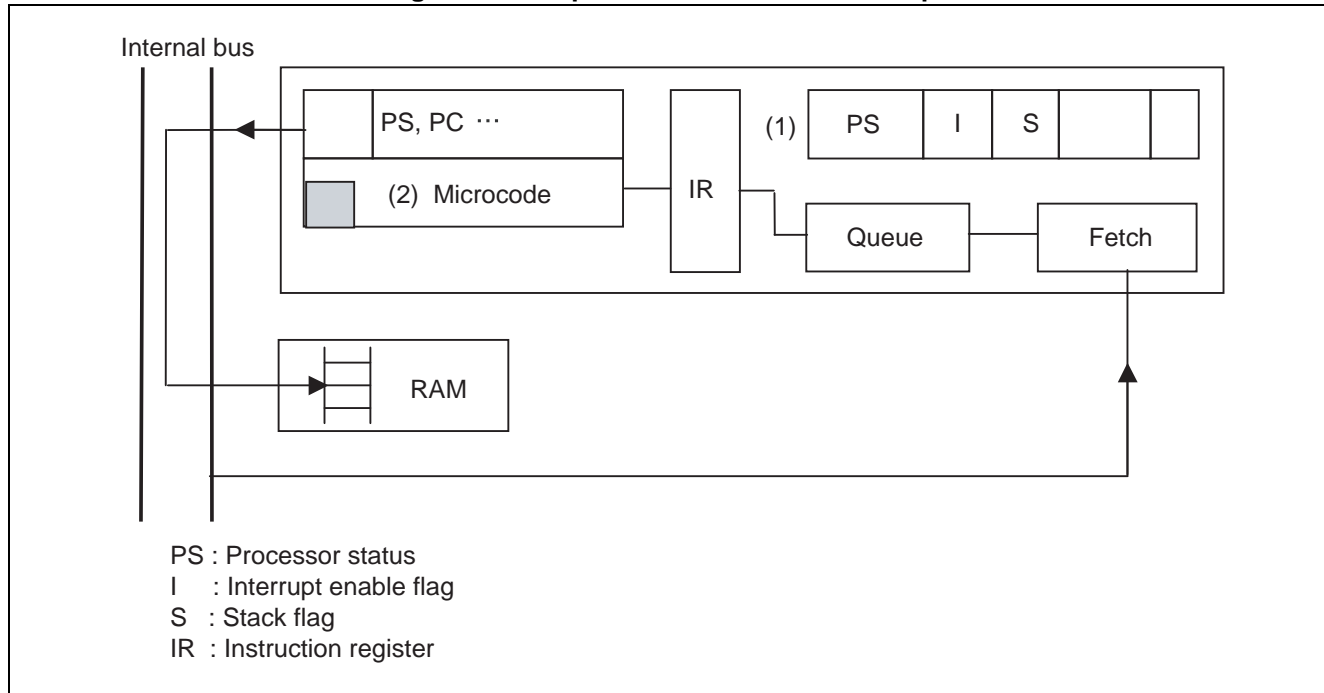
### ■ Return from Software Interrupt

When the return interrupt (RETI instruction) instruction is executed in the interrupt handling program, the 12-byte data saved in the system stack returns to the dedicated registers. Control then returns to the process which was being executed before the interrupt branch.

### ■ Operation of Software Interrupt

Figure 3.5-1 shows the operation sequence from generation of a software interrupt to completion of interrupt handling.

Figure 3.5-1 Operation of Software Interrupt



(1) Software interrupt instruction is executed.

(2) The required processes are performed; for example, the contents of the dedicated registers are saved according to the microcode associated with the software interrupt instruction. The branching process is then executed.

(3) The RETI instruction in the user-defined interrupt handling routine terminates the interrupt handling.

### ■ Precautions on Software Interrupt

If the program counter bank register (PCB) is "FF<sub>H</sub>", the vector area for the CALLV instruction overlaps with the table for the INT#vct8 instruction. When creating the software, pay attention to overlap of the CALLV instruction and INT#vct8 instruction addresses.

## 3.6 Interrupts by Extended Intelligent I/O Service (EI<sup>2</sup>OS)

---

**The extended intelligent I/O service (EI<sup>2</sup>OS) executes automatic data transfer between the peripheral function (I/O) and memory. A hardware interrupt is generated at the end of the data transfer.**

---

### ■ Extended Intelligent I/O Service (EI<sup>2</sup>OS)

The extended intelligent I/O service is a kind of hardware interrupts. EI<sup>2</sup>OS executes automatic data transfer between the peripheral function (I/O) and memory. EI<sup>2</sup>OS executes data exchange with the peripheral function (I/O), previously executed by the interrupt handling program, like direct memory access (DMA), at the end of transfer, and sets the end condition before control automatically branches to the interrupt handling routine. The user has to write the program only about the start and end of EI<sup>2</sup>OS without having to code the data transfer program between them.

#### ● Advantages of extended intelligent I/O service (EI<sup>2</sup>OS)

Compared with data transfer by the interrupt handling routine, EI<sup>2</sup>OS provides the user with the following advantages:

- Because no data transfer program needs to be coded, the program size can shrink.
- Because the transfer can be stopped depending the state of the peripheral function (I/O), no unnecessary data needs to be transferred.
- Enables the user to select whether the buffer address is incremented or not updated.
- Enables the user to select whether the I/O register address is incremented or not updated.

#### ● Interrupt by extended intelligent I/O service (EI<sup>2</sup>OS) termination

After completion of data transfer by EI<sup>2</sup>OS, this sets the end condition in the S1 and S0 bits of the interrupt control register (ICR) before automatic branch to the interrupt handling routine.

The cause for a stop of EI<sup>2</sup>OS can be determined by checking the EI<sup>2</sup>OS status (S1 and S0 of ICR) with the interrupt handling program.

The interrupt numbers, interrupt vectors, etc. are fixed for individual peripheral functions. For details, see Section "3.2 Interrupt Cause and Interrupt Vector".

#### ● Interrupt control registers (ICR)

Located in the interrupt controller. Activates EI<sup>2</sup>OS, specifies the channel, and displays the at end state of EI<sup>2</sup>OS.

#### ● Extended intelligent I/O service (EI<sup>2</sup>OS) descriptor (ISD)

Located in the 000100<sub>H</sub> to 00017F<sub>H</sub> area in RAM. Contains a set of 8-byte data used to store the transfer mode, I/O address and transfer count, and buffer address. Repeated for 16 channels. Specifies the channel with the interrupt control register (ICR).

---

Note:

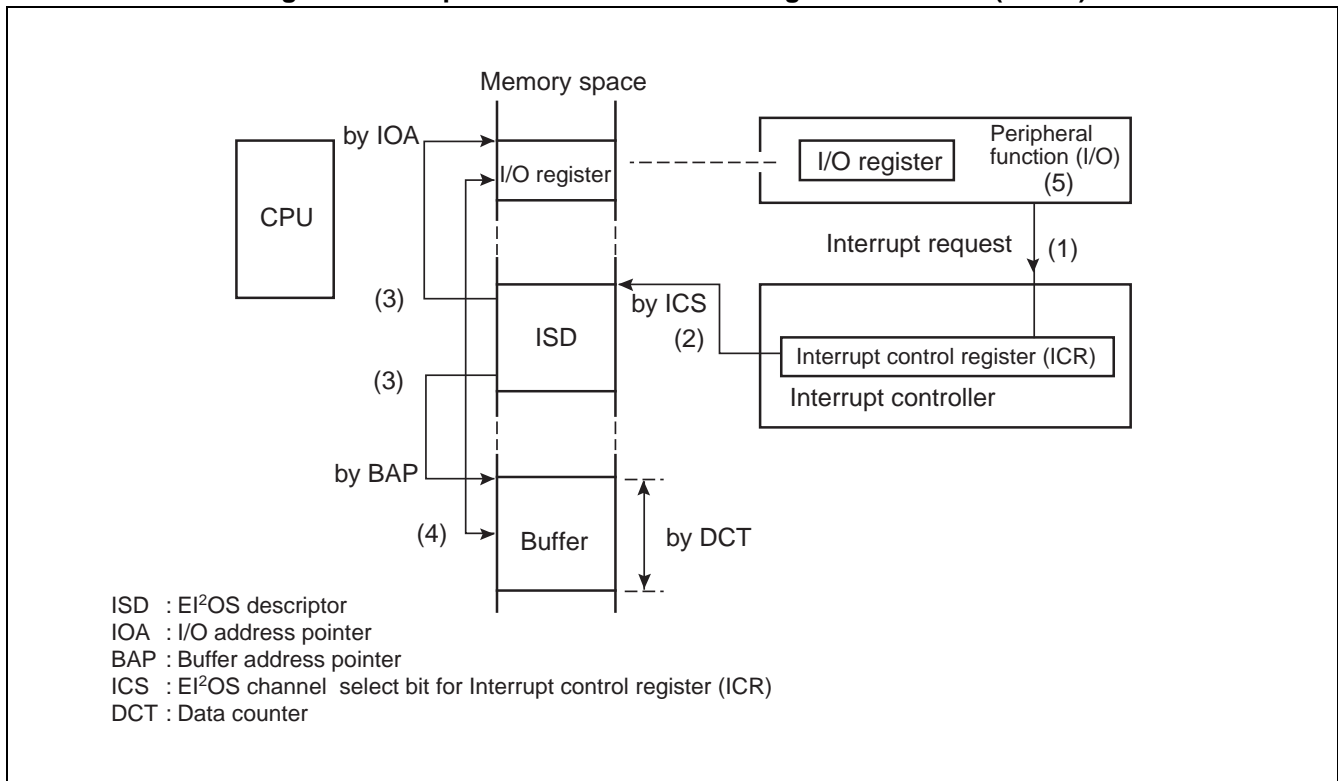
The CPU program execution stops while the extended intelligent I/O service (EI<sup>2</sup>OS).

---

### ■ Operation of Extended Intelligent I/O Service (EI<sup>2</sup>OS)

Figure 3.6-1 shows the operation of the EI<sup>2</sup>OS.

**Figure 3.6-1 Operation of Extended Intelligent I/O Service (EI<sup>2</sup>OS)**



- (1) I/O demands forwarding.
- (2) The interrupt controller selects the descriptor.
- (3) Reads the transfer-source or transfer-destination from the descriptor.
- (4) Forwarding between I/O and the memory is executed.
- (5) The interruption factor is automatically clear.



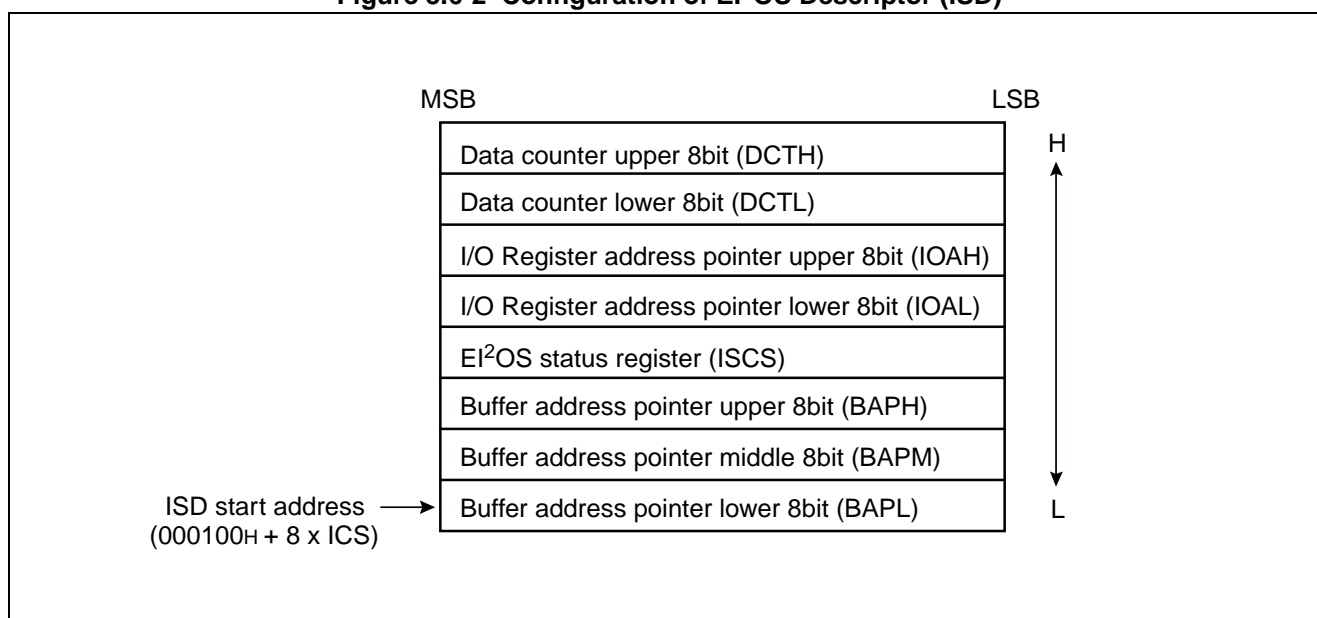
### 3.6.1 Extended Intelligent I/O Service (EI<sup>2</sup>OS) Descriptor (ISD)

Extended intelligent I/O service (EI<sup>2</sup>OS) descriptor (ISD) is existed to the addresses 000100<sub>H</sub> to 00017F<sub>H</sub> in the internal RAM and consists of 8 bytes × 16 channels.

#### ■ Configuration of Extended Intelligent I/O Service (EI<sup>2</sup>OS) Descriptor (ISD)

Configuration of ISD consists of 8 bytes × 16 channels, and each ISD has the configuration shown in Figure 3.6-2. The correspondence between the channel numbers and ISD addresses is as listed in Table 3.6-1.

Figure 3.6-2 Configuration of EI<sup>2</sup>OS Descriptor (ISD)



**Table 3.6-1 Correspondence between Channel Numbers and Descriptor Addresses**

Channel	Descriptor address
0	000100 <sub>H</sub>
1	000108 <sub>H</sub>
2	000110 <sub>H</sub>
3	000118 <sub>H</sub>
4	000120 <sub>H</sub>
5	000128 <sub>H</sub>
6	000130 <sub>H</sub>
7	000138 <sub>H</sub>
8	000140 <sub>H</sub>
9	000148 <sub>H</sub>
10	000150 <sub>H</sub>
11	000158 <sub>H</sub>
12	000160 <sub>H</sub>
13	000168 <sub>H</sub>
14	000170 <sub>H</sub>
15	000178 <sub>H</sub>

### 3.6.2 Each Register of Extended Intelligent I/O Service (EI<sup>2</sup>OS) Descriptor (ISD)

The extended intelligent I/O service (EI<sup>2</sup>OS) descriptor (ISD) consists of the following registers.

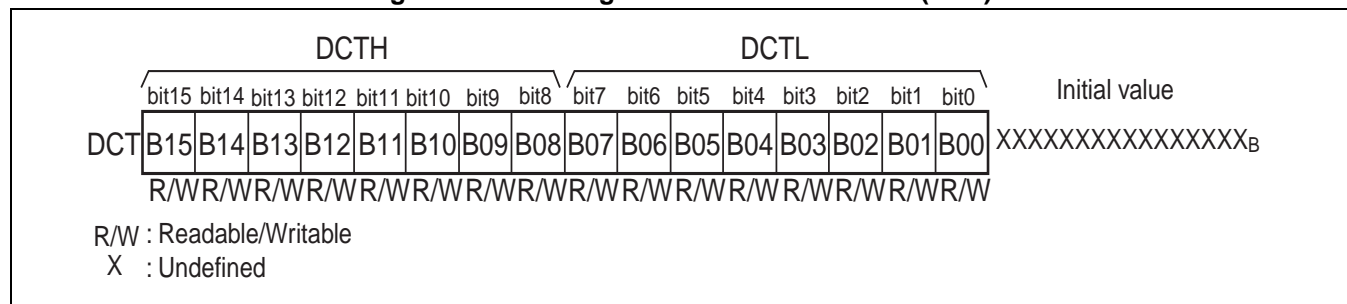
- Data counter (DCT)
- I/O register address pointer (IOA)
- EI<sup>2</sup>OS status register (ISCS)
- Buffer address pointer (BAP)

Note that resetting each register causes its initial value to be undefined.

#### ■ Data Counter (DCT)

Data counter (DCT), a 16-bit length register, indicates the corresponding to the transfer data count. After each of data has been transferred, the counter is decremented by 1 (reduced value). EI<sup>2</sup>OS ends when this counter reaches "0". Figure 3.6-3 shows the DCT configuration.

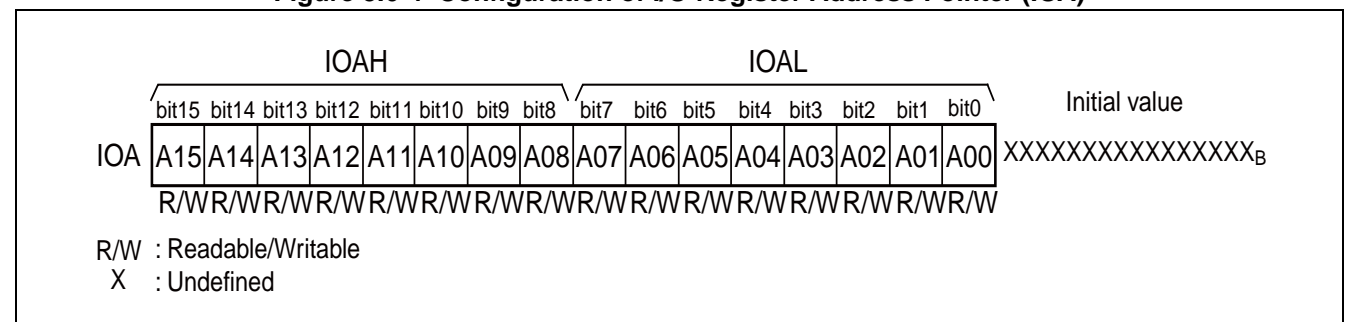
Figure 3.6-3 Configuration of Data Counter (DCT)



#### ■ I/O Register Address Pointer (IOA)

I/O register address pointer (IOA), a 16-bit length register, contains those low address (A15 to A00) of the I/O register which are used for data transfer to or from the buffer. The upper address (A23 to A16) containing all "0" data, can specify any I/O from 000000<sub>H</sub> to 00FFFF<sub>H</sub>. Figure 3.6-4 shows the IOA configuration.

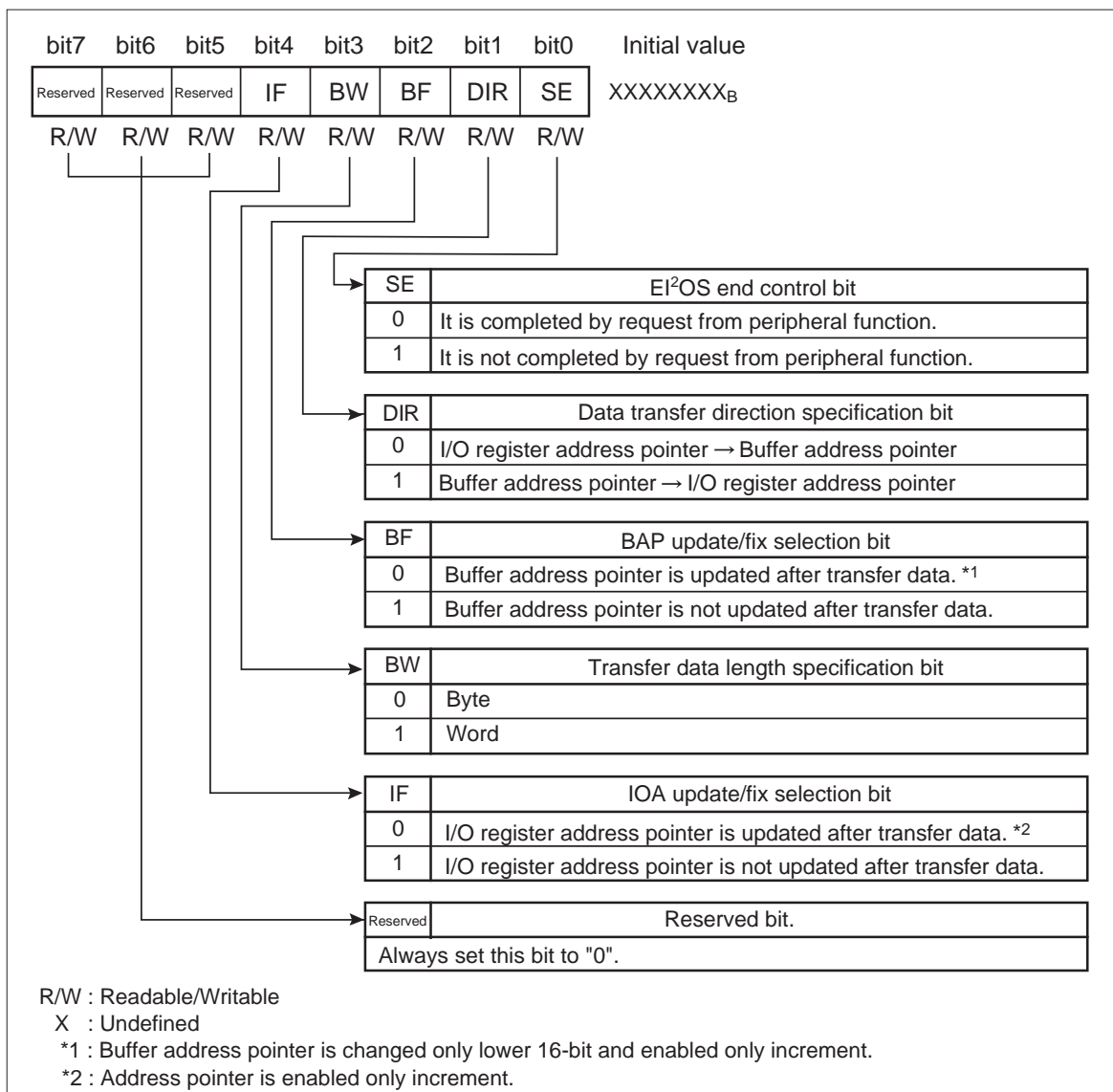
Figure 3.6-4 Configuration of I/O Register Address Pointer (IOA)



### Extended Intelligent I/O Service (EI<sup>2</sup>OS) Status Register (ISCS)

The extended intelligent I/O service status register (ISCS) updates or fixes the buffer address and I/O register address pointers by the 8-bit length register. It also indicates the transfer data format (byte or word) and the direction of transfer. Figure 3.6-5 shows the configuration of the ISCS.

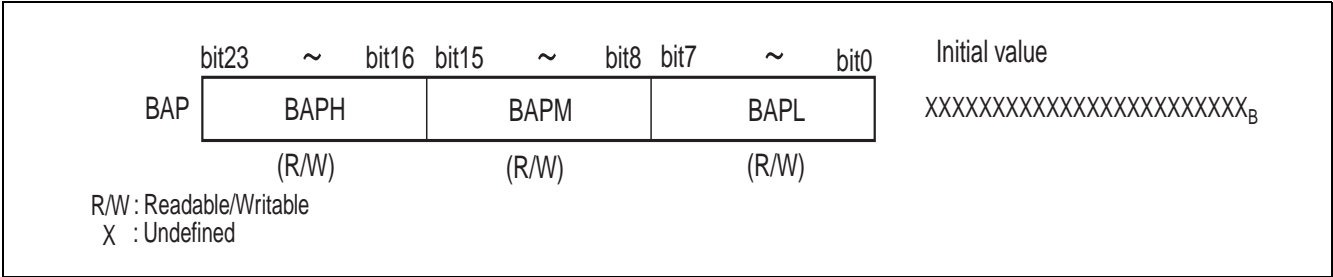
Figure 3.6-5 Configuration of EI<sup>2</sup>OS Status Register (ISCS)



■ Buffer address Pointer (BAP)

The buffer address pointer (BAP), a 24-bit register, contains the address used for the next attempt of transfer by EI<sup>2</sup>OS. The BAP is provided independently for the EI<sup>2</sup>OS channels to enable the EI<sup>2</sup>OS channels to transfer data between any address of 16 Mbytes and I/O. If the BF bit (BAP update/fix selection bit of EI<sup>2</sup>OS status register) of the EI<sup>2</sup>OS status register (ISCS) is set to "updated", only the low order 16-bit (BAPM, BAPL) will change in the BAP. The higher 8-bit (BAPH) will be unchanged in this case. Figure 3.6-6 shows the BAP configuration.

Figure 3.6-6 Configuration of Buffer address Pointer (BAP)



Note:

In the I/O address pointer (IOA), the 000000<sub>H</sub> to 00FFFF<sub>H</sub> area is available for specification.  
In the buffer address pointer (BAP), the 000000<sub>H</sub> to FFFFFF<sub>H</sub> area is available for specification.  
The maximum transfer count which may be specified in the data counter (DCT) is 65536 (64 Kbytes).

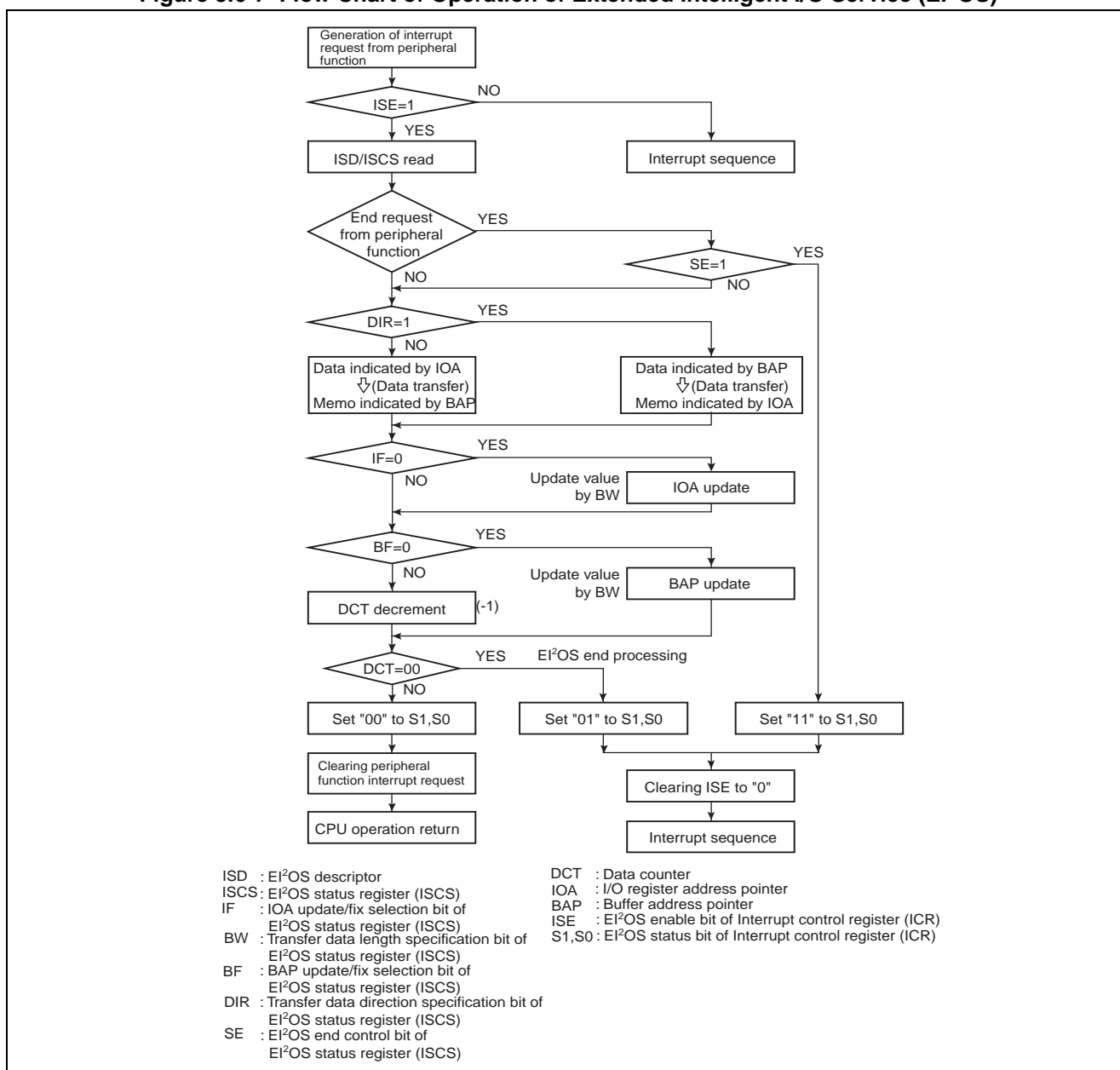
### 3.6.3 Operation of Extended Intelligent I/O Service (EI<sup>2</sup>OS)

If the peripheral function has generated an interrupt request and activation of EI<sup>2</sup>OS has been set in the associated interrupt control register (ICR), the CPU will execute data transfer using EI<sup>2</sup>OS. Once data transfer has been executed the specified number of times, the hardware interrupt handling is executed automatically.

#### ■ Operation of Extended Intelligent I/O Service (EI<sup>2</sup>OS)

Figure 3.6-7 shows the EI<sup>2</sup>OS operation flow chart by the microcode of the internal CPU.

Figure 3.6-7 Flow Chart of Operation of Extended Intelligent I/O Service (EI<sup>2</sup>OS)



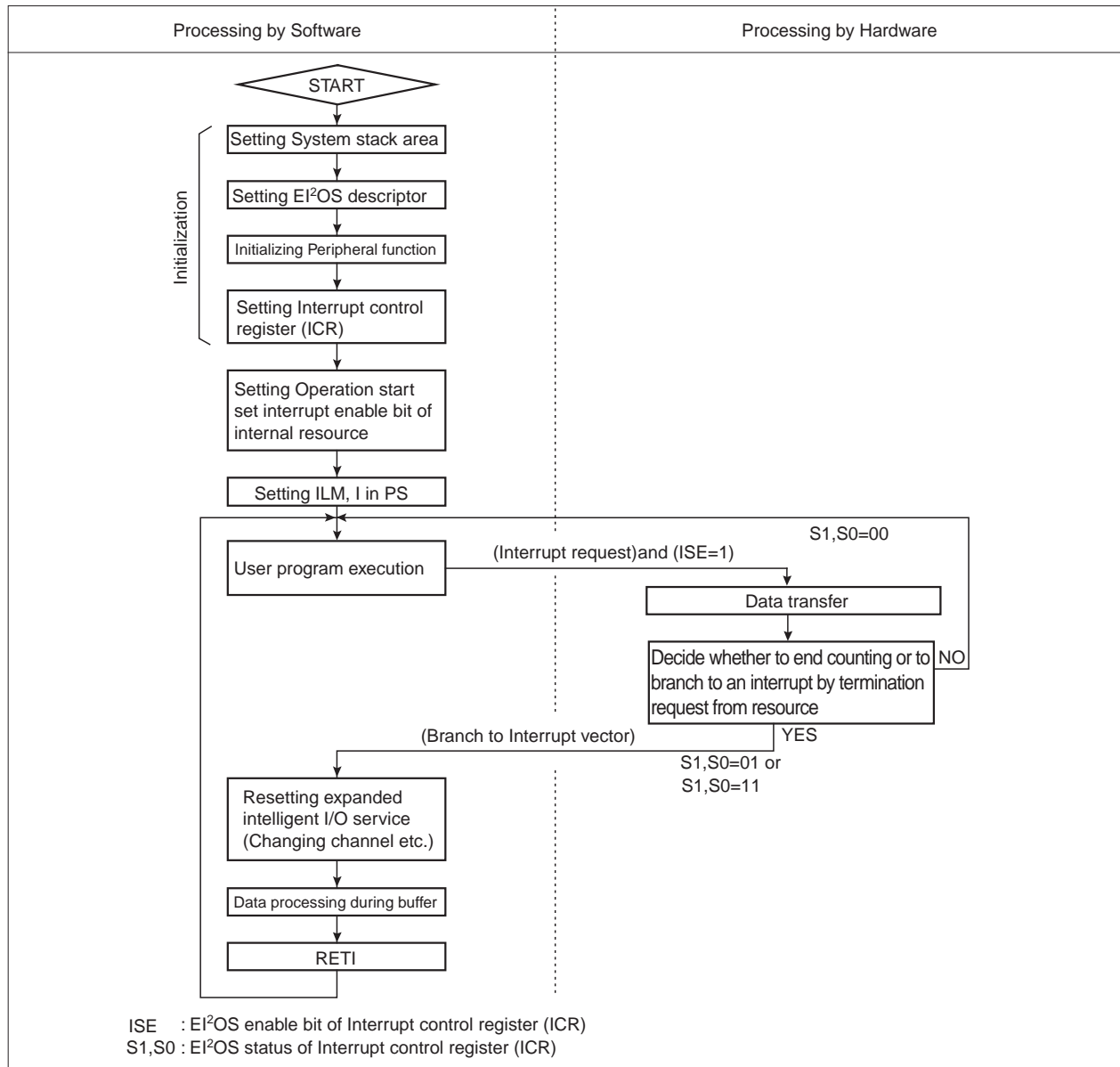
### 3.6.4 Procedure for use of Extended Intelligent I/O Service (EI<sup>2</sup>OS)

To use extended intelligent I/O service (EI<sup>2</sup>OS), the system stack area, EI<sup>2</sup>OS descriptor, peripheral function, interrupt control register (ICR), and other requirements must be set up.

#### ■ Procedure for Use of Extended Intelligent I/O Service (EI<sup>2</sup>OS)

Figure 3.6-8 shows the EI<sup>2</sup>OS software and the process by hardware.

Figure 3.6-8 Procedure for Use of Extended Intelligent I/O Service (EI<sup>2</sup>OS)



### 3.6.5 Extended Intelligent I/O Service (EI<sup>2</sup>OS) Processing Time

The time required for extended intelligent I/O service (EI<sup>2</sup>OS) processing depends on the following factors:

- Setting of EI<sup>2</sup>OS status register (ISCS)
- Address (area) indicating I/O register address pointer (IOA)
- Address (area) indicating buffer address pointer (BAP)
- Width of external data bus when external is accessed
- Data length of transfer data

Because a hardware interrupt is activated at the end of data transfer by EI<sup>2</sup>OS, the interrupt handling time will be added.

#### ■ Extended Intelligent I/O Service (EI<sup>2</sup>OS) Processing Time (Time for One Transfer)

- For continued data transfer

For continued data transfer, the EI<sup>2</sup>OS processing time varies with the setting in the EI<sup>2</sup>OS status register (ISCS), as listed in Table 3.6-2.

**Table 3.6-2 Extended Intelligent I/O Service Execution Time**

Setting of the EI <sup>2</sup> OS termination control bit (SE)		Terminated by an end request from periphery		Termination request from the surrounding is disregarded.	
Setting of the IOA updating/fixing select bit (IF)		Fixed	Update	Fixed	Update
BAP address update/fixation Setting of selection bits (BF)	Fixed	32	34	33	35
	Update	34	36	35	37

Unit: One machine cycle is equal to one clock cycle of the machine clock ( $\phi$ ).



Further, correction may be required, depending on the condition for executing EI<sup>2</sup>OS, as shown in Table 3.6-3.

**Table 3.6-3 Compensation Value for Data Transfer at EI<sup>2</sup>OS Processing Time**

I/O register address pointer			Internal Access		External access	
			B/Even	Odd	B/Even	8/Odd
Buffer address pointer	Internal Access	B/Even	0	+ 2	+ 1	+ 4
		Odd	+ 2	+ 4	+ 3	+ 6
	External access	B/Even	+ 1	+ 3	+ 2	+ 5
		8/Odd	+ 4	+ 6	+ 5	+ 8

B: Byte data transfer

8: External bus width 8-bit/word transfer

Even: Word transfer at even address

Odd: Word transfer at odd address

● At completion of counting by data counter (DCT) (for final data transfer)

Because a hardware interrupt is activated at the end of data transfer by EI<sup>2</sup>OS, the interrupt handling time is added. The EI<sup>2</sup>OS processing time at the end of counting is calculated by the following expression.

EI<sup>2</sup>OS processing time when count ends =

EI<sup>2</sup>OS processing time in data transfer +  $(21 + 6 \times Z)$  machine cycles

↑

Interrupt handling time

The interrupt handling time depends on the address to which the stack pointer points. Table 3.6-4 shows the compensation values (Z) of the interrupt handling time.

**Table 3.6-4 Compensation Value of Interrupt Handling Time (Z)**

Address which stack pointer indicates	Compensation Value (Z)
At the external 8 bits	+ 4
At the external even number address	+ 1
At the external odd number address	+ 4
At the internal even number address	0
At the internal odd number address	+ 2

- At the end caused by an end request from the peripheral function (I/O)

If data transfer by EI<sup>2</sup>OS is aborted due to an end request from the peripheral function (I/O) (ICR S1, ICR S0 = 11), the hardware interrupt is activated without performing data transfer. The EI<sup>2</sup>OS processing time is calculated using the expression below. Z in the expression represents the interrupt handling time compensation value (see Table 3.6-4).

EI<sup>2</sup>OS processing time for aborted =  $36 + 6 \times Z$  machine cycles

---

Reference:

One machine cycle is equal to one clock cycle of the machine clock ( $\phi$ ).

---

## 3.7 Exception Processing Interrupt

---

**F<sup>2</sup>MC-16LX executes exception handling by executing undefined instructions.**

**Exception handling, basically the same as interrupt, is executed when an exception item is detected during a period between instructions, the normal process is suspended for this purpose.**

**In general, exception handling takes place as a result of unpredicted operation; it is recommended that it be used only for debugging or for activating the recovery software in the event of an emergency.**

---

### ■ Exception Processing Interrupt

#### ● Operation of exception processing

F<sup>2</sup>MC-16LX regards as an undefined instruction any code not defined in the instruction map. When executing an undefined instruction, F<sup>2</sup>MC-16LX performs a process equivalent to software interrupt instruction "INT#10".

Before control branches to the interrupt routine, the exception handling performs the following processes:

- 1) Saving the registers of A, DPR, ADB, DTB, PCB, PC, and PS in the system stack.
- 2) Clearing the I flag of the condition code register (CCR) to "0" to mask the hardware interrupt.
- 3) Setting the S flag of the condition code register (CCR) to "1" to enable the system stack.

The program counter (PC) value saved in the system stack indicates the address storing the undefined instruction. For any instruction code of 2 or more bytes, the PC value indicates the address which stores the code by which the instruction has been identified as an undefined one. The PC value is useful to determine the type of the exception cause in the exception handling routine.

#### ● Return from exception processing

After returning from the exception handling according to the RETI instruction, control starts the exception handling again because the PC points to an undefined instruction. Some measurement such as performing a software reset should be taken.

## **3.8 Interruption by $\mu$ DMAC**

---

**$\mu$ DMAC is a simplified DMA with the same function as EI<sup>2</sup>OS.**

---

### 3.8.1 $\mu$ DMAC Function

### 3.8.2 Register of $\mu$ DMAC

#### 3.8.2.1 DMA Descriptor Channel Specification Register (DCSR)

#### 3.8.2.2 DMA Status Register (DSRH/DSRL)

#### 3.8.2.3 DMA Stop Status Register (DSSR)

#### 3.8.2.4 DMA Permission Register (DERH/DERL)

### 3.8.3 DMA Descriptor Window Register (DDWR)

#### 3.8.3.1 DMA Data Counter (DDCTH/DDCTL)

#### 3.8.3.2 DMA I/O Register Address Pointer (DIOAH/DIOAL)

#### 3.8.3.3 DMA Control Register (DMACS)

#### 3.8.3.4 DMA Buffer Address Pointer (DBAPH/DBAPM/DBAPL)

### 3.8.4 Explanation of Operation of $\mu$ DMAC

### **3.8.1 $\mu$ DMAC Function**

---

**$\mu$ DMAC is simple DMA with the function equal with EI<sup>2</sup>OS.**

---

#### **■ $\mu$ DMAC Function**

$\mu$ DMAC has the following functions.

- Performs automatic data transfer between the peripheral resource (I/O) and memory.
- The program execution of CPU stops in the DMA startup.
- The DMA transfer channel is 16 channels (The smaller the channel number, the higher the priority of DMA transfer).
- Can select either "incremented" or "not incremented" for the source or destination address.
- DMA transfer may be activated, depending on the interrupt cause by a peripheral resource (I/O).
- DMA transfer can be controlled with (a) DMA permission register (DERH/DERL), (b) DMA stop status register (DSSR), (c) DMA status register (DSRH/DSRL), (d) DMA descriptor channel specification register (DCSR), and (e) descriptor (DMACS).
- A STOP request is available for stopping DMA transfer from the resource.
- After completion of DMA transfer, the flag is set in the appropriate bit of the DMA status register (DSRH/DSRL), resulting in output of an interrupt to the interrupt controller.

3.8.2 Register of  $\mu$ DMAC

$\mu$ DMAC has four registers: DCSR, DSR, DSSR, and DER. The DMA descriptor used to set up DMA transfer is described in Section "3.8.3 DMA Descriptor Window Register (DDWR)".

■  $\mu$ DMAC Register ListFigure 3.8-1  $\mu$ DMA Register List

• DMA descriptor channel specification register (DCSR)									
bit	15	14	13	12	11	10	9	8	
00009B <sub>H</sub>	STP	Reserved	Reserved	Reserved	DCSR3	DCSR2	DCSR1	DCSR0	DCSR
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 00000000 <sub>B</sub>
• DMA status register (DSRH/DSRL)									
bit	15	14	13	12	11	10	9	8	
00009D <sub>H</sub>	DTE15	DTE14	DTE13	DTE12	DTE11	DTE10	DTE9	DTE8	DSRH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 00000000 <sub>B</sub>
bit	7	6	5	4	3	2	1	0	
00009C <sub>H</sub>	DTE7	DTE6	DTE5	DTE4	DTE3	DTE2	DTE1	DTE0	DSRL
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 00000000 <sub>B</sub>
• DMA stop status register (DSSR)									
bit	7	6	5	4	3	2	1	0	
0000A4 <sub>H</sub>	STP15	STP14	STP13	STP12	STP11	STP10	STP9	STP8	DSSR
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 00000000 <sub>B</sub>
bit	7	6	5	4	3	2	1	0	
0000A4 <sub>H</sub>	STP7	STP6	STP5	STP4	STP3	STP2	STP1	STP0	DSSR
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 00000000 <sub>B</sub>
When the STP bit of DCSR is "0", DSSR uses STP15 to STP8. When the STP bit of DCSR is "1", DSSR uses STP7 to STP0.									
• DMA permission register (DERH/DERL)									
bit	15	14	13	12	11	10	9	8	
0000AD <sub>H</sub>	EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	DERH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 00000000 <sub>B</sub>
bit	7	6	5	4	3	2	1	0	
0000AC <sub>H</sub>	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0	DERL
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 00000000 <sub>B</sub>

R/W : Readable/Writable

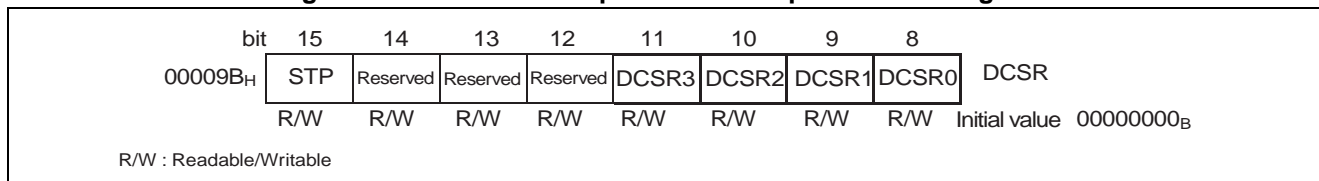
### 3.8.2.1 DMA Descriptor Channel Specification Register (DCSR)

DMA descriptor channel specification register (DCSR) switches the descriptor of each channel.

The descriptor is set after the channel is specified by this register.

#### ■ DMA Descriptor Channel Specification Register (DCSR)

**Figure 3.8-2 DMA Descriptor Channel Specification Register**



**[bit 11 to bit 8] DCSR<sub>x</sub>: Specifies the DMA descriptor channel.**

**Table 3.8-1 Relation between DCSR and Selector Channel**

DCSR3 to DCSR0	Selection channel	Resource interrupt request
0000 <sub>B</sub>	0	USB function 1 (End Point 0-IN)
0001 <sub>B</sub>	1	USB function 1 (End Point 0-OUT)
0010 <sub>B</sub>	2	USB function 2 (End Point 1)*
0011 <sub>B</sub>	3	USB function 2 (End Point 2)*
0100 <sub>B</sub>	4	USB function 2 (End Point 3)
0101 <sub>B</sub>	5	USB function 2 (End Point 4)
0110 <sub>B</sub>	6	USB function 2 (End Point 5)
0111 <sub>B</sub>	7	Input capture 0,1
1000 <sub>B</sub>	8	Input capture 2,3
1001 <sub>B</sub>	9	I/O extended serial
1010 <sub>B</sub>	10	UART2/3, Reception
1011 <sub>B</sub>	11	UART2/3, Transmission
1100 <sub>B</sub>	12	UART0/1, Reception
1101 <sub>B</sub>	13	UART0/1, Transmission
1110 <sub>B</sub>	14	PWC, reload timer 0
1111 <sub>B</sub>	15	A/D converter, conversion end, and free run timer

One descriptor channel of the 16 channels is selected by setting the DCSR. For details, see Section "3.8.3 DMA Descriptor Window Register (DDWR)".

\*: This function can use even when USB HOST is operated.

## [bit 15] STP:STP control bit

STP bit	Function
0 [Initial value]	STP8 to STP15 is selected as DSSR.
1	STP0 to STP7 is selected as DSSR.

## [bit 14 to bit 12] Reserved: (reserved bits)

These bits are a reserved bit.

These bits are always "0" at the beginning of reading.

Please write "0".

---

### Note:

Do not use any read modify write (RMW) instruction to access the DCSR register.

---

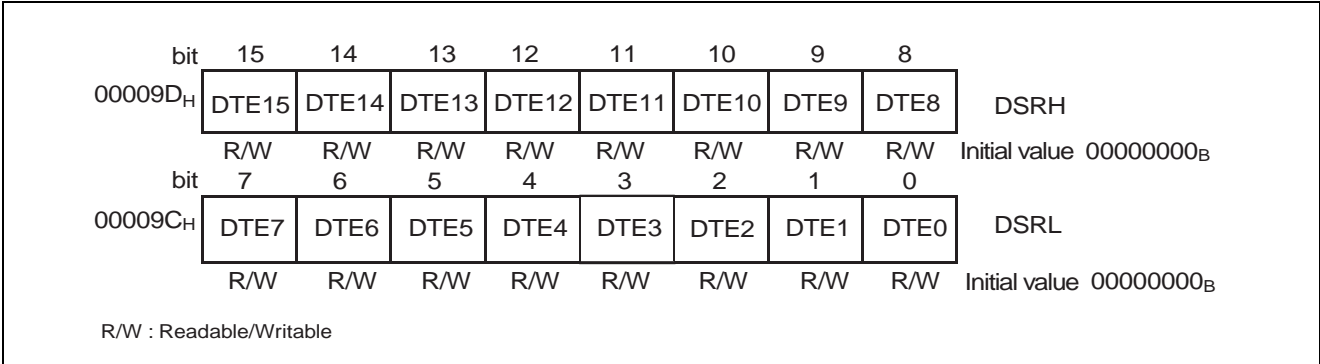


3.8.2.2 DMA Status Register (DSRH/DSRL)

DMA status register (DSRH/DSRL) indicates that the DMA transfer ended. When "1" is set to this register, the interrupt is generated at the same time.

■ Bit Configuration of DMA Status Register (DSRH/DSRL)

Figure 3.8-3 Bit Configuration of DMA Status Register (DSRH/DSRL)



[bit 15 to bit 0] DTE<sub>x</sub>: DMA Status

DTE <sub>x</sub> bit	Function
0 [Initial value]	The DMA transfer has not ended. Please write "0" when DTE <sub>x</sub> is "0".
1	Indicates that DMA transfer was completed and an interrupt request is being executing. The DMA transfer due to the STOP request except last transfer does not set 1 to this bit. When DTE <sub>x</sub> is "1", writing "0" clears it to "0" and writing "1" holds the previous data.

Note:

To write data to the DSRH/DSRL, use a read-modify-write (RMW) instruction.

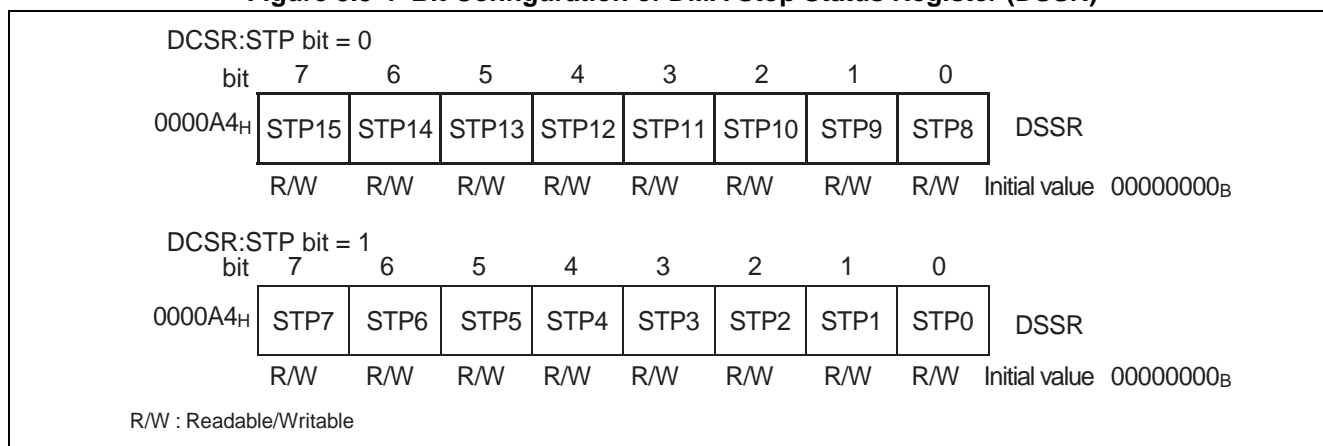
## 3.8.2.3 DMA Stop Status Register (DSSR)

DMA stop status register (DSSR) indicates that the DMA transfer stopped due to the STOP request.

The meaning of the bit in this register is different depending on the STP bit of the DMA descriptor channel specification register (DCSR).

## ■ DMA Stop Status Register (DSSR)

Figure 3.8-4 Bit Configuration of DMA Stop Status Register (DSSR)



## [bit15 to bit 0] STPx: DMA stop status

STPx bit	Function
0 [Initial value]	During DMA transfer, no STOP request is accepted from the resource. Please write "0" at STPx=0.
1	During DMA transfer, indicates that DMA transfer stopped in response to a STOP request from the resource. However, "1" is not set to STPx bit even though the STOP request is accepted at last transfer. If the SE bit of DMA control register is "1" and a STOP request is received by the associated channel, the corresponding bit of the DMA permission register is cleared to "0". When STPx = 1, writing "0" clears it to "0" and writing "1" holds the previous data.

The following two channels correspond to the STOP demand.

Channel	Corresponding STPx bit	Resource
ch.10	STP10	UART2/3, Reception
ch.12	STP12	UART0/1, Reception

Bits other than STP10 and STP12 do not have the meaning.

## Notes:

- DSSR is controlled by most significant bit (STP) of DCSR. If STP is "0", STP15 to STP8 will be selected as being used for the DSSR. If it is "1", STP7 to STP0 will be used for the DSSR. Because the initial value of STP is "0", STP15 to STP8 is initially selected.
- To write data to the DSSR, use a read modify write (RMW) instruction.

### 3.8.2.4 DMA Permission Register (DERH/DERL)

DMA permission register (DERH/DERL) enables the DMA transfer.

When "1" is set to this register, the interrupt request, which is the DMA transfer request, generates to the corresponding channel, and starts the DMA transfer.

#### ■ DMA Permission Register (DERH/DERL)

Figure 3.8-5 Bit Configuration of DMA Permission Register (DERH/DERL)

bit	15	14	13	12	11	10	9	8	
0000AD <sub>H</sub>	EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	DERH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 00000000 <sub>B</sub>
bit	7	6	5	4	3	2	1	0	
0000AC <sub>H</sub>	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0	DERL
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 00000000 <sub>B</sub>

R/W : Readable/Writable

[bit15 to bit0] ENx: DMA permission

ENx bit	Function
0 [Initial value]	This bit does not execute the DMA transfer.
1	The interrupt request from the resource is handled as a DMA activation request, and the interrupt request is output to the interrupt controller at the end of DMA transfer. When the number of DMA transfer bytes reaches 0, or a STOP request from the resource stops DMA transfer, this is cleared to "0".

#### Notes:

- To write data to the DERH/DERL, use a read modify write (RMW) instruction.
- Before changing the mode to the standby mode (sleep mode, stop mode, watch mode, and time-base timer mode) or the CPU intermittent operation mode (main clock intermittent operation mode, PLL clock intermittent mode, and sub clock intermittent mode), the DMA permission register (DERH/DERL) must be set to "0".

### 3.8.3 DMA Descriptor Window Register (DDWR)

The DMA descriptor, consisting of 8 bytes  $\times$  16 channels, is used to set up DMA transfer. One of the 16 channels is specified, and mapped to the DMA descriptor window register (DDWR) for being accessible. The address of DDWR is 007920<sub>H</sub> to 007927<sub>H</sub>.

#### ■ Configuration of DMA Descriptor Window Register (DDWR)

The DMA descriptor consists of 8 bytes  $\times$  16 channels. The configuration of each channel is shown in Figure 3.8-6. The descriptor of the channel selected by the DMA descriptor channel specification register (DCSR) or interrupt request channel number is mapped to the DMA descriptor window register (DDWR). See Table 3.8-1 for the relationship between the DMA descriptor channel specification register (DCSR) and the selected channel.

**Figure 3.8-6 Configuration of DMA Descriptor Window Register (DDWR)**

Address	
007927 <sub>H</sub>	DMA Data counter upper 8-bit (DDCTH)
007926 <sub>H</sub>	DMA Data counter lower 8-bit (DDCTL)
007925 <sub>H</sub>	DMA I/O register address pointer upper 8-bit (DIOAH)
007924 <sub>H</sub>	DMA I/O register address pointer lower 8-bit (DIOAL)
007923 <sub>H</sub>	DMA Control register (DMACS)
007922 <sub>H</sub>	DMA Buffer address pointer upper 8-bit (DBAPH)
007921 <sub>H</sub>	DMA Buffer address pointer middle 8-bit (DBAPM)
007920 <sub>H</sub>	DMA Buffer address pointer lower 8-bit (DBAPL)

#### ■ Each Register of DMA Descriptor

Each register configuring the DMA descriptor is described in the following pages. The initial value of each register is made undefined when a reset is generated. Thus, make sure that the initialization has finished before EN<sub>x</sub> is set to "1".

#### Note:

If the DCSR is used to switch the channel descriptor, access to DDWR is inhibited during the two subsequent machine cycles.

### 3.8.3.1 DMA Data Counter (DDCTH/DDCTL)

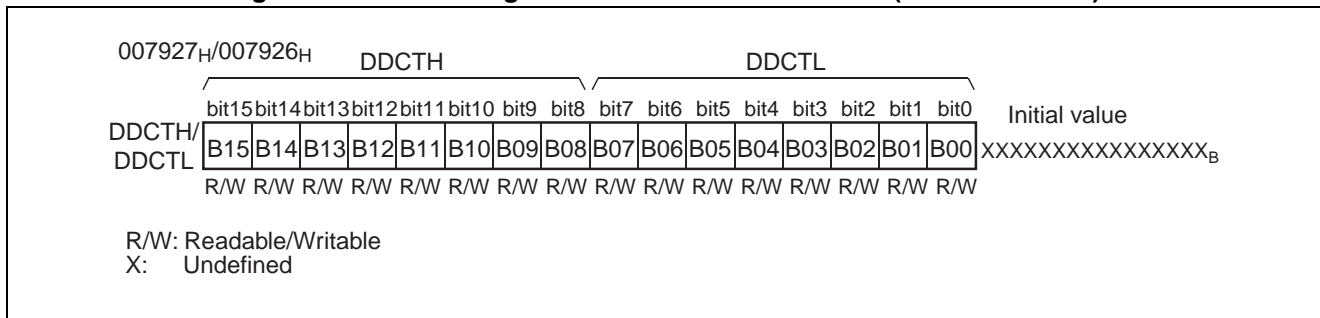
**DMA data counter (DDCTH/DDCTL) sets the data transfer.**  
**When DDCTH/DDCTL is 0, the DMA transfer ends.**

#### ■ DMA Data Counter (DDCTH/DDCTL)

DMA data counter (DDCTH/DDCTL), a 16-bit length register, indicates the counter associated with transferred number. After each data has been transferred, the counter is always decremented by 1 regardless of transferred data (word or byte). The DMA transfer ends when this counter reaches 0. Figure 3.8-7 shows the DDCT configuration.

If the DDCT is set to "0", the maximum data transfer count (65536) is set.

**Figure 3.8-7 Bit Configuration of DMA Data Counter (DDCTH/DDCTL)**



#### ■ About the Set Value of DMA Data Counter (DDCTH/DDCTL)

Table 3.8-2 shows the relationship between the number of transferred bytes and the DDCTH/DDCTL.

**Table 3.8-2 Set Value of DMA Data Counter (DDCTH/DDCTL)**

DMACS		DDCT
BW bit	BYTEL bit	
0	-	N
1	0	N/2
1	1	(N+1)/2

N: Number of transfer bytes

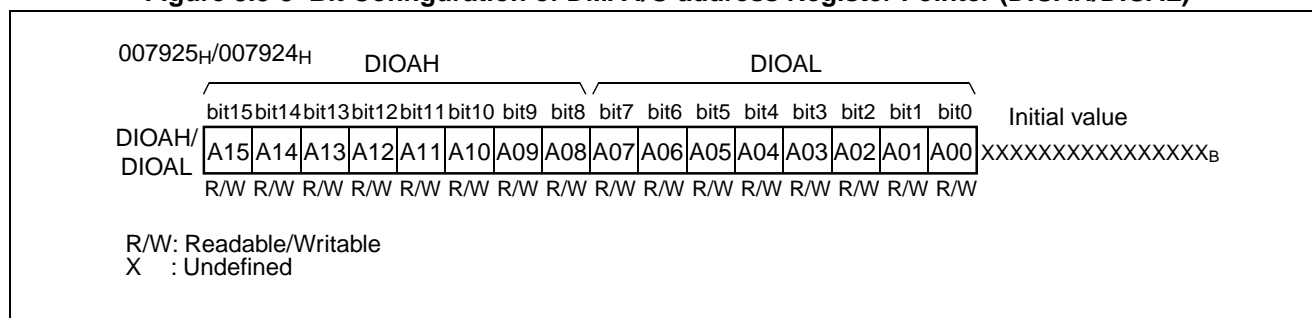
### 3.8.3.2 DMA I/O Register Address Pointer (DIOAH/DIOAL)

DMA I/O register address pointer (DIOAH/DIOAL) sets the I/O address pointer. The upper address (A23 to A16) is fixed at "00<sub>H</sub>".

#### ■ DMA I/O Register Address Pointer (DIOAH/DIOAL)

The DMA I/O register address pointer (DIOAH/DIOAL), a 16-bit length register, indicates the 16 low order bits (A15 to A00) of the DMA I/O register address. The upper-level address (A23 to A16), containing all "0" data, can specify any I/O address space from 000000<sub>H</sub> to 00FFFF<sub>H</sub>. If the IF bit (DIOA update/fix selection bit) of the DMA control register (DMACS) is set to "updated", the DIOA will be incremented by 1 during byte transfer or by 2 during word transfer. If this bit is set to "not updated", the DIOA will be fixed. Figure 3.8-8 shows the DIOA configuration.

**Figure 3.8-8 Bit Configuration of DMA I/O address Register Pointer (DIOAH/DIOAL)**



### 3.8.3.3 DMA Control Register (DMACS)

DMA control register (DMACS) controls the DMA transfer.

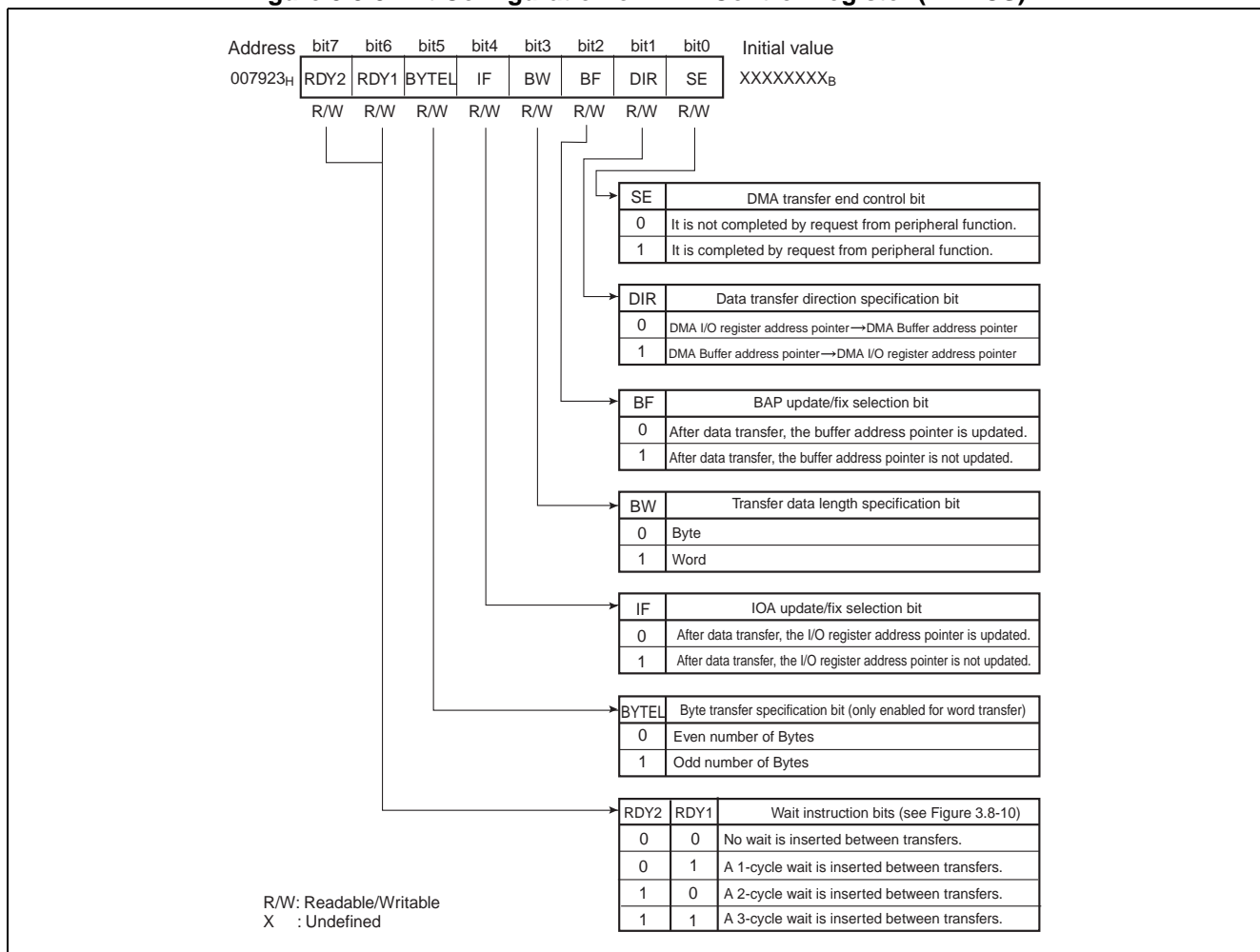
The following can be controlled by the DMACS.

- Direction control (IOA  $\rightarrow$  BAP and BAP  $\rightarrow$  IOA)
- Transfer bit length (Byte and word)
- Address update (provided or not provided)
- Transfer interval
- Odd-numbered byte control at word transfer

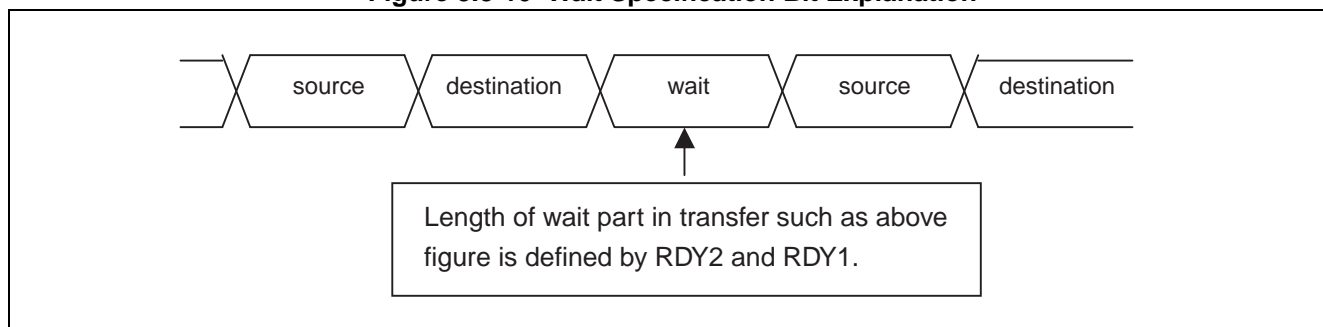
#### ■ DMA Control Register (DMACS)

The DMA control register (DMACS), an 8 bit length specifies update/fix, the transfer data format (byte or word), the direction of transfer, and byte transfer, and issues a wait instruction of DMA buffer address pointer and DMA I/O register address pointer. Figure 3.8-9 shows the DMACS configuration.

Figure 3.8-9 Bit Configuration of DMA Control Register (DMACS)



**Figure 3.8-10 Wait Specification Bit Explanation**



**Note:**

If writing transmission data to UART by using  $\mu$ DMAC, not setting RDY2 and RDY1 bit of DMACS register in (0, 0).



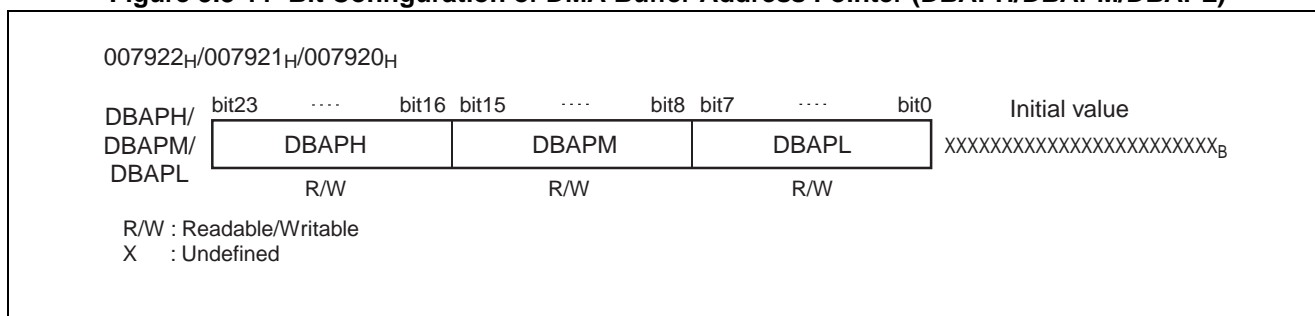
### 3.8.3.4 DMA Buffer Address Pointer (DBAPH/DBAPM/DBAPL)

**DMA buffer address pointer (DBAPH/DBAPM/DBAPL) sets the buffer address pointer. The DBAPH/DBAPM/DBAPL can be set A23 to A00.**

#### ■ DMA Buffer Address Pointer (DBAPH/DBAPM/DBAPL)

The DMA buffer address pointer (DBAPH/DBAPM/DBAPL), a 24-bit register, contains the address used for DMA transfer. DBAP are provided independently for the DMA channels to enable the DMA channels to transfer data between any address of 16 Mbytes and I/O. If the BF bit (DBAP update/fix selection bit) of the DMA control register (DMACS) is set to "updated", the 16 lower order bits (DBAPM, DBAPL) of the DBAP will be incremented by 1 during byte transfer or by 2 during word transfer. The 8 high order bits (DBAPH) will be unchanged in this case Figure 3.8-11 shows the DBAP configuration.

**Figure 3.8-11 Bit Configuration of DMA Buffer Address Pointer (DBAPH/DBAPM/DBAPL)**



#### Notes:

- In the DMA I/O register address pointer (DIOAH/DIOAL), the 000000<sub>H</sub> to 00FFFF<sub>H</sub> area is available for specification.
- In the DMA buffer address pointer (DBAPH/DBAPM/DBAPL), the 000000<sub>H</sub> to FFFFFFF<sub>H</sub> area is available for specification.
- $\mu$ DMA internal register DCSR, DSRH, DSRL, DSSR, DERH, or DERL, or an address of DMA descriptor window register (DDWR) may not be specified in the DIOA or DBAP.

### 3.8.4 Explanation of Operation of $\mu$ DMAC

This section describes the  $\mu$ DMAC operation.

#### ■ Operation of $\mu$ DMAC

Figure 3.8-12 shows the DMA operation.

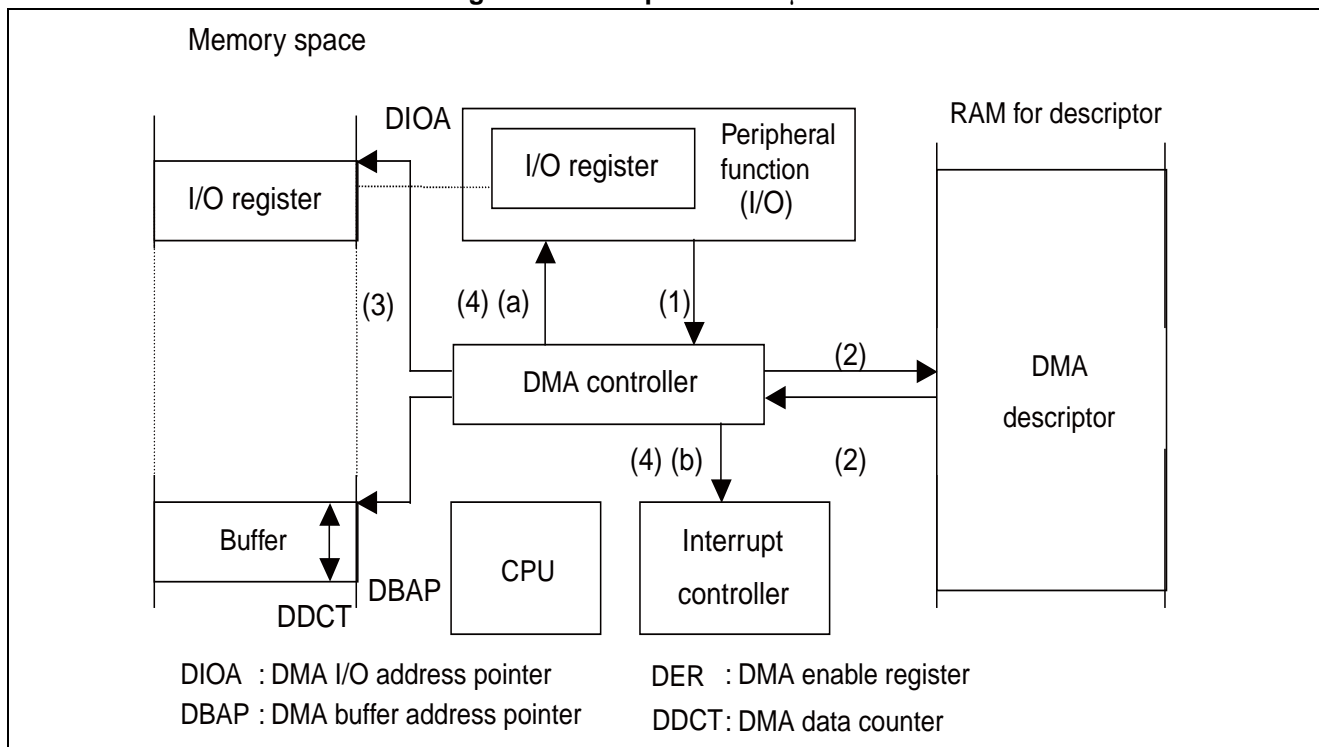
Data transfer using  $\mu$ DMAC performs the following steps in order:

1. The peripheral resource (I/O) makes a request for DMA transfer.
2. If the DMA permission register (DERH/DERL) is "1",  $\mu$ DMAC reads transfer-related data, such as the source and destination addresses for the specified channel and the transfer count, from the descriptor.
3. The DMA data transfer is begun between I/O and the memory.
4. After executing forwarding one byte or 1 word.
  - (a) If transfer is not yet completed, that is, the DMA data counter (DDCT) does not contain 0 yet,  
A request to clear the DMA transfer request is issued to the peripheral resource.
  - (b) When forwarding ends (DMA data counter DDCT=0)  
After completion of DMA transfer, the transfer end flag is set.

Note:

When writing to the internal register DSRH, DSRL, DSSR, DERH, and DERL, be sure to use the read modify write (RMW) instruction.

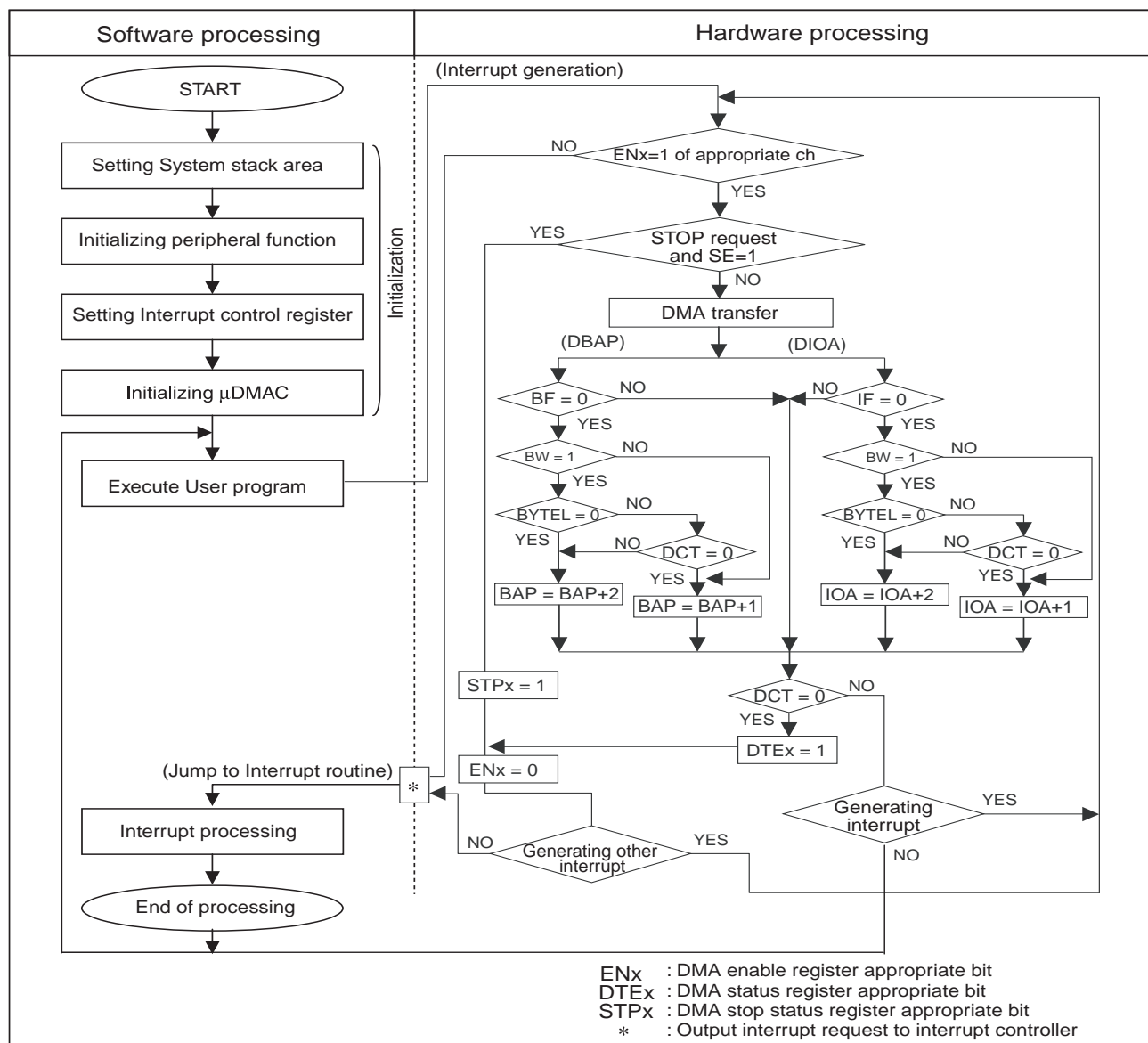
Figure 3.8-12 Operation of  $\mu$ DMAC



### ■ Use Procedure of $\mu$ DMAC

Figure 3.8-13 shows the procedure for using  $\mu$ DMAC.

Figure 3.8-13 Use Procedure of  $\mu$ DMAC



## 3.9 Exceptions

---

The F<sup>2</sup>MC-16LX performs exception processing when the following event occurs.

---

### ■ Execution of an Undefined Instruction

Exception processing is fundamentally the same as interrupt processing. When an exception is detected between instructions, exception processing is performed separately from ordinary processing. In general, exception processing is performed as a result of an unexpected operation. Fujitsu recommends using exception processing only for debugging or for activating emergency recovery software.

### ■ Exception due to Execution of an Undefined Instruction

The F<sup>2</sup>MC-16LX handles all codes that are not defined in the instruction map as undefined instructions. When an undefined instruction is executed, processing equivalent to the INT 10 software interrupt instruction is performed. Specifically, the AL, AH, DPR, DTB, ADB, PCB, PC, and PS values are saved into the system stack, and processing branches to the routine indicated by the interrupt number 10 vector. In the undefined instruction is stored. Processing can be restored by the RETI instruction, but is of no use, however, because the same exception occurs again.

## 3.10 Stack Operation of Interrupt Processing

Once an interrupt is accepted, the contents of the dedicated registers are automatically saved in the system stack before control branches to the interrupt handling. Return from the stack at the end of the interrupt handling also takes place automatically.

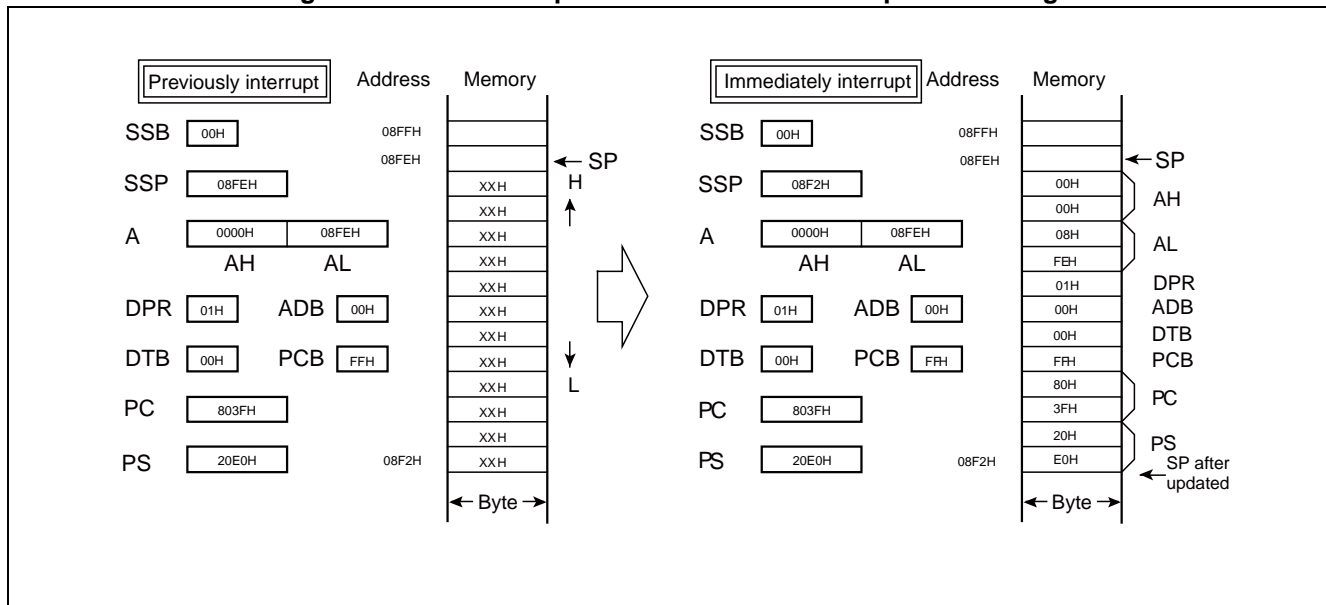
### ■ Stack Operation at the Start of Interrupt Processing

Once the interrupt is accepted, the CPU automatically saves the contents of the current dedicated registers and their related data in the system stack in the order below:

1. Accumulator (A)
2. Direct page register (DPR)
3. Additional data bank register (ADB)
4. Data bank register (DTB)
5. Program counter bank register (PCB)
6. Program counter (PC)
7. Processor status (PS)

Figure 3.10-1 shows the stack operation at the start of interrupt handling.

Figure 3.10-1 Stack Operation at Start of Interrupt Processing



### ■ Stack Operation when Interrupt Processing Returns

When the interrupt return instruction (RETI) is executed at the end of interrupt process, the values of PS, PC, PCB, DTB, ADB, DPR, and A return from the stack in the reverse order for the start of interrupt handling. The dedicated registers are restored to the initial state preceding the start of interrupt.

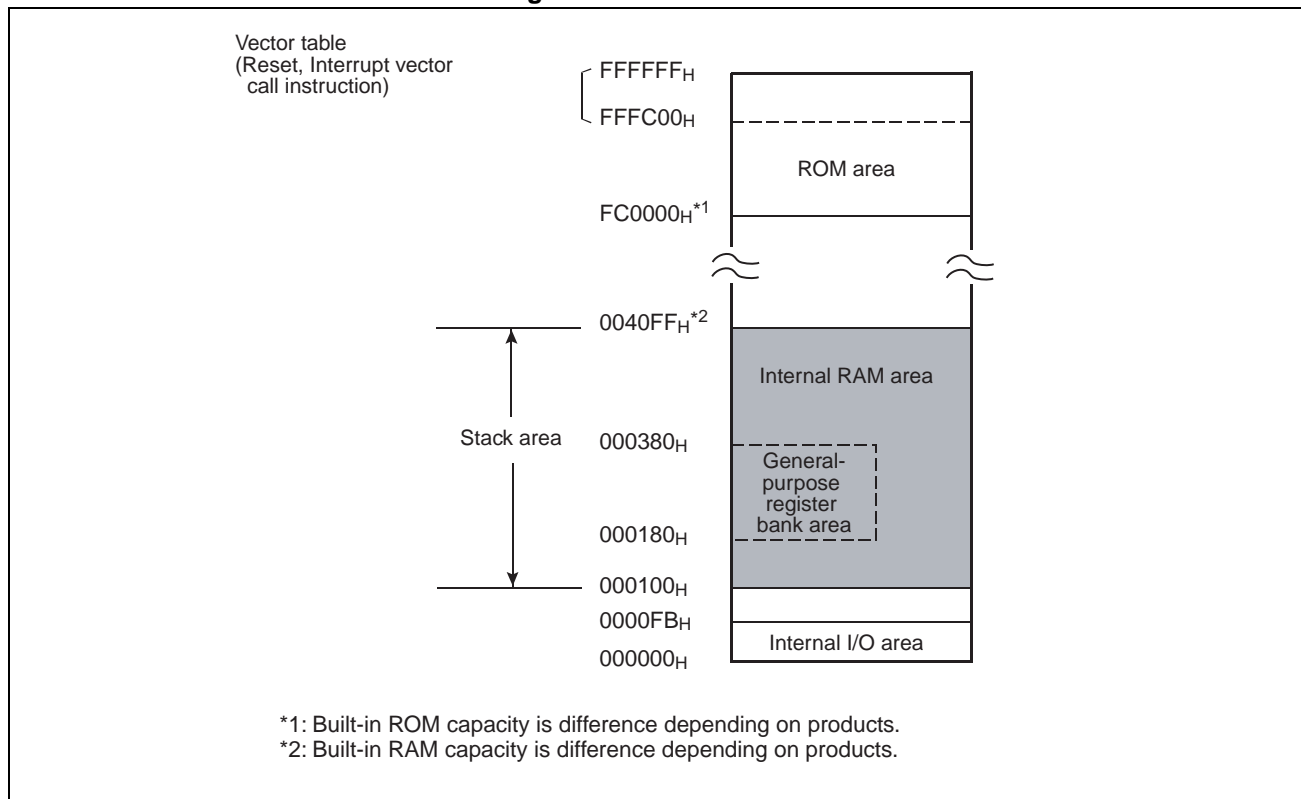
## ■ Stack Area

### ● Securing stack area

The stack area is used to save or return the program counter (PC) used to execute the subroutine call (CALL) or vector call (CALLV) instruction as well as execute the interrupt handling. This area is also used, by the PUSHW or POPW instruction, to save or return the contents of temporary registers and their related data. The stack area is located in RAM together with the data area.

Figure 3.10-2 shows the stack area.

**Figure 3.10-2 Stack Area**



### Notes:

- As a general rule, even addresses should be set in the stack pointers (SSP and USP).
- The system stack area, user stack area, and data area should not overlap.

### ● System stack area and user stack area

The system stack area is used for interrupt processing. Even if the user stack area is being used, generation of an interrupt causes forcible switching to the system stack. For this reason, the system stack area must have been set up properly even in a system where the user stack area is primarily used. In particular, only the system stack area should be used unless it is necessary to divide the stack space.

## 3.11 Program Example of Interrupt Processing

An example of interrupt processing program is shown below.

### ■ Program Example of Interrupt Processing

#### ● Processing specification

An example interruption program that uses external interruption 0 (INT0).

#### ● Coding example

```

DDR6      EQU      000016H          ; Port 6 direction register
ENIR      EQU      00003CH          ; Interruption/DTP permission register.
EIRR      EQU      00003DH          ; Interruption/DTP factor register
ELVR      EQU      00003EH          ; Request level setting register
ICR03     EQU      0000B3H          ; Interrupt control register 03
STACK     SSEG
RW        100
STACK_T   RW        1
STACK     ENDS
;-----Main Program-----
CODE      CSEG                      ;
START:
        MOV      RP, #0              ; Header bank used for general-purpose registers
        MOV      ILM, #07H           ; Sets ILM in PS to level 7
        MOV      A, #!STACK_T        ; Sets system stack
        MOV      SSB, A
        MOVW     A, #STACK_T          ; Setting of stack pointer, in this case,
        MOVW     SP, A                ; S flag = 1, so set to SSP
        MOV      DDR6, #00000000B    ; Sets the P60/INT0 pin for input
        OR       CCR, #40H           ; Sets I flag of CCR in PS to enable interrupts.
        MOV      I:ICR03, #00H        ; It is assumed interrupt levels 0(Max)
        MOV      I:ELVR, #00000001B  ; Make INT0 "H" level request
        MOV      I:EIRR, #00H         ; Clears interrupt cause for INT0.
        MOV      I:ENIR, #01H        ; INT0 input permitted
        :
LOOP:    NOP                          ; Dummy loop
        NOP
        NOP

```

```

NOP
BRA    LOOP                ; Unconditional jump
;-----Interrupt Program-----
ED_INT1:
MOV     I:EIRR, #00H        ; New acceptance of INT0 prohibited
NOP
NOP
NOP
NOP
NOP
NOP
RET     RETI                ; Returns from interrupt.
CODE    ENDS
;-----Vector Settings-----
VECT     CSEG    ABS=0FFH
ORG      0FFB4H          ; The vector is set in interruption #18(12H)
DSL      ED_INT1
ORG      0FFDCH          ; Reset vector setting
DSL      START
DB       00H             ; Single-chip mode setting
VECT     ENDS
END      START

```

● Processing specification for program example of extended intelligent I/O service (EI<sup>2</sup>OS)

1. The extended intelligent I/O service (EI<sup>2</sup>OS) is started upon detection of the "H" level signal which inputs to the INT0 pin.
2. When "H" level is input to INT0 pin, EI<sup>2</sup>OS is started, which transfers data at port 0 to memory address "3000<sub>H</sub>".
3. The number of transfer data bytes is 100. After the 100 bytes have been transferred, an interrupt is generated because of completion of EI<sup>2</sup>OS data transfer.



● Coding example

```

DDR6      EQU      000016H          ; Port 6 direction register
ENIR       EQU      00003CH          ; Interruption/DTP permission register
EIRR       EQU      00003DH          ; Interruption/DTP factor register
ELVR       EQU      00003EH          ; A register to specify the required level
ICR03      EQU      0000B3H          ; Interrupt control registers 03
BAPL       EQU      000100H          ; Buffer address pointer lower
BAPM       EQU      000101H          ; Buffer address pointer middle
BAPH       EQU      000102H          ; Buffer address pointer upper
ISCS       EQU      000103H          ; EI2OS Status
IOAL       EQU      000104H          ; Lower I/O address pointer
IOAH       EQU      000105H          ; Higher I/O address pointer
DCTL       EQU      000106H          ; Data counter lower
DCTH       EQU      000107H          ; Data counter upper
ER0        EQU      EIRR:0          ; Defines external interrupt request flag bit.
STACK      SSEG
RW         100
STACK_T    RW      1
STACK      ENDS
;-----Main Program-----
CODE       CSEG
START:
          AND      CCR,#0BFH          ; Clears the I flag of CCR in PS to disable interrupts.
          MOV      RP,#00              ; Sets register bank pointer.
          MOV      A,#!STACK_T        ; Sets system stack
          MOV      SSB,A
          MOVW     A,#STACK_T          ; Setting of stack pointer, in this case,
          MOVW     SP,A                ; S flag = 1, so set to SSP
          MOV      I:DDR6,#00000000B  ; Sets the P60/INT0 pin for input.
          MOV      BAPL,#00H          ; Sets buffer address (003000H)
          MOV      BAPM,#30H
          MOV      BAPH,#00H
          MOV      ISCS,#00010001B    ; No I/O address update, byte transfer
                                     ; Buffer address updated
                                     ; The peripheral function terminates I/O to buffer
                                     ; transfer.
                                     ; Yes
          MOV      IOAL,#00H          ; Sets transfer source address (port 0: 000000H)
          MOV      IOAH,#00H

```

```

MOV    DCTL,#64H          ; Sets transfer byte count (100 bytes)
MOV    DCTH,#00H
MOV    I:ICR00,#00001000B ; EI2OS channel 0, EI2OS enable, interrupt level 0
                                (highest)
MOV    I:ELVR,#00000001B  ; Make INT0 "H" level request
MOV    I:EIRR,#00H        ; Clears interrupt cause for INT0.
MOV    I:ENIR,#01H        ; Enables INT0 interrupt.
MOV    ILM,#07H           ; Sets ILM in PS to level 7
OR     CCR,#40H           ; Sets I flag of CCR in PS to enable interrupts.
:
LOOP:   BRA    LOOP        ; Infinite loop
;-----Interrupt Program-----
WARI    CLRB    ER0        ; Interrupt/DTP request flag clear
:
        user processing    ; confirmation of the cause of the EI2OS completion
:
                                ; Processes data in buffer, resetting EI2OS etc.
        RETI
CODE    ENDS
;-----Vector Settings-----
VECT    CSEG    ABS=0FFH
        ORG     0FFB4H      ; The vector is set in interruption #18(12H).
        DSL     WARI
        ORG     0FFDCH      ; Reset vector setting
        DSL     START
        DB      00H        ; Single-chip mode setting
VECT    ENDS
        END     START

```

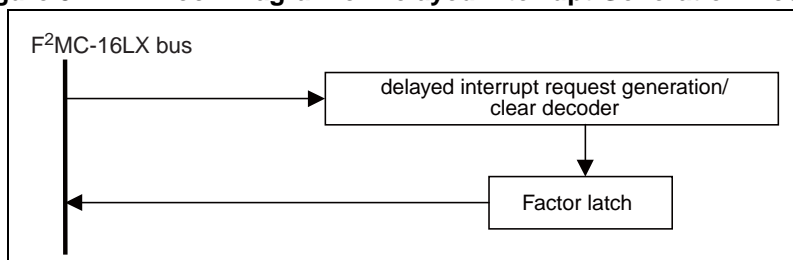
## 3.12 Delayed Interrupt Generation Module

The delayed interrupt generation module is used to generate a task switching interrupt. Use of this module enables F<sup>2</sup>MC-16LX CPU to generate or cancel an interrupt request.

### ■ Block Diagram of Delayed Interrupt Generation Module

Figure 3.12-1 shows the block diagram of the delayed interrupt generation module.

Figure 3.12-1 Block Diagram of Delayed Interrupt Generation Module



### ■ List of Register of Delay Interrupt Generation Module

The register configuration of the delayed interrupt generation module {delayed interrupt cause generation/clear register (delayed interrupt request register (DIRR))} is shown in the following figure.

Figure 3.12-2 Delayed Interrupt Cause Generation/clear Register (DIRR)

bit	15	14	13	12	11	10	9	8	Initial value
00009F <sub>H</sub>	—	—	—	—	—	—	—	R0	-----0 <sub>B</sub>
	—	—	—	—	—	—	—	R/W	

R/W : Readable/Writable

Delay interruption factor generation/release register (DIRR) controls the delay factor generation and release. Writing "1" to this register generates a delayed interrupt request. Writing "0" to it resets the request. A reset causes the state cause to remain cleared. Either "0" or "1" may be written to the reserved bit area. However, it is recommended that the set or clear bit instruction for accessing this register be used, taking the future extension into account.

### 3.12.1 Operation of Delayed Interrupt Generation Module

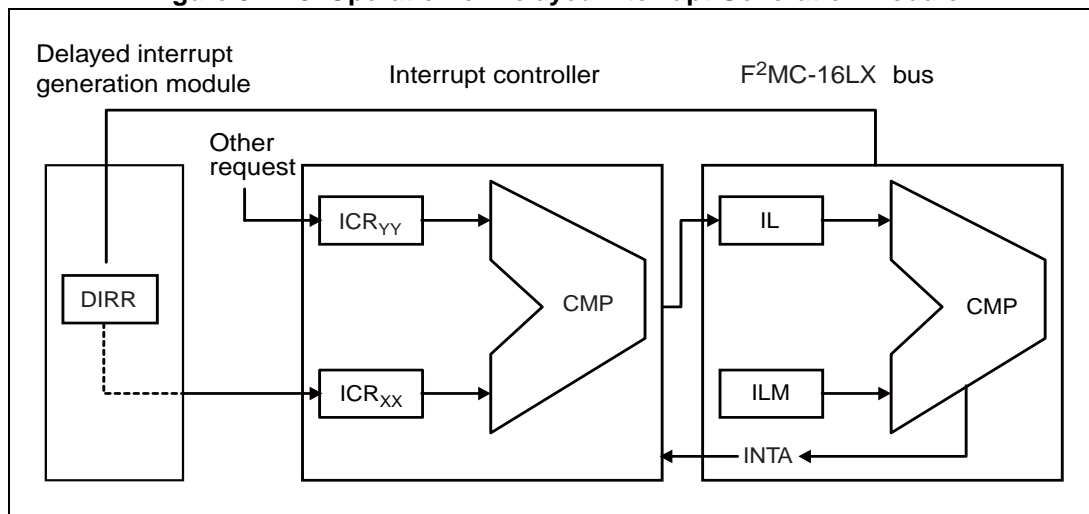
When the CPU writes "1" to the appropriate bit of the DIRR by software, the request latch in the delay interrupt generation module is set, resulting in generation of the interrupt request to the interrupt controller.

#### ■ Operation of Delayed Interrupt Generation Module

When the CPU writes "1" to the appropriate bit of the DIRR by software, the request latch in the delay interrupt generation module is set, resulting in generation of the interrupt request to the interrupt controller. If all the other interrupt requests have a lower priority than that of this one, or no other requests have been generated, the interrupt controller generates the interrupt to F<sup>2</sup>MC-16LX CPU. When F<sup>2</sup>MC-16LX CPU compares the ILM bit in the internal CCR register with interrupt request, if the request level is higher than the ILM bit, a hardware interrupt processing microprogram is activated as soon as the current execution instruction is completed. As a result, the interrupt routine is executed for this interrupt. The interrupt cause is cleared by writing 0 to the appropriate bit of the DIRR in the interrupt handling routine. This also causes task switching.

Figure 3.12-3 shows the above operation flow.

**Figure 3.12-3 Operation of Delayed Interrupt Generation Module**



#### ■ Notes on Use of Delay Interrupt Generation Module (Delay Interrupt Request Latch)

This latch is set by writing "1" to the appropriate bit of the DIRR, and cleared by writing "0" to this bit. Note that for this reason, the interrupt handling may be reactivated immediately when control returns from the interrupt cause handling, unless the software has been designed to clear the cause in the interrupt handling routine.



# **CHAPTER 4**

---

# ***RESET***

**This chapter explains reset of the MB90330A series.**

- 4.1 Outline of Reset
- 4.2 Reset Factors and Oscillation Stabilization Wait Times
- 4.3 External Reset Pin
- 4.4 Reset Operation
- 4.5 Reset Factor Bit
- 4.6 State of Each Pin at Reset

## 4.1 Outline of Reset

When the reset cause is generated, the CPU suspends the currently executed process immediately before entering the wait state for release of the reset. After the reset is cleared, processing starts from the address indicated in the reset vector.

There are the following four kinds of factors of resets.

- Generation power on reset
- Watchdog timer overflow
- Generation of external reset request from  $\overline{\text{RST}}$  pin
- Generation of software reset request

### ■ Reset Factor

Table 4.1-1 shows the causes of reset.

Table 4.1-1 Reset Factor

Reset	Generation factor	Machine clock	Watchdog timers	Oscillation stability waiting
Power on	At power on	Main clock (MCLK)	Stops	Yes
Watchdog timer	Watchdog timer overflow	Main clock (MCLK)	Stops	None
External pin	Input "L" level to $\overline{\text{RST}}$ pin	Main clock (MCLK)	Stops	None
Software	A "0" is written to the $\overline{\text{RST}}$ bit of low-power consumption mode control register (LPMCR)	Main clock (MCLK)	Stops	None

Main clock: Oscillation clock frequency divided by 2

#### ● Power on reset

Power on reset is reset generated at power on. For an evaluation or flash product, the oscillation stabilization wait time is  $2^{18}/\text{HCLK}$  (approx. 43.70 ms at an oscillation clock of 6 MHz). For a MASK product, this wait time is  $2^{17}/\text{HCLK}$  (approx. 21.85 ms at an oscillation clock of 6 MHz). Reset operation starts after elapsing the oscillation stabilization wait time.

#### ● Watchdog reset

Watchdog reset generates a reset in response to an overflow of the watchdog timer after start of the watchdog timer. This overflow occurs when a "0" is not written in the watchdog control bit (WTE) of the watchdog timer control register (WDTC) within the predetermined time. The oscillation stabilization wait time can be selected using the clock select register (CKSCR).

## ● External reset

An external reset is generated by inputting the "L" level signal to external reset pin ( $\overline{\text{RST}}$ ). The input time of the "L" level signal to the ( $\overline{\text{RST}}$ ) pin must be continued for 16 machine cycles ( $16/\phi$ ) or more.

The external reset, that is, the  $\overline{\text{RST}}$  pin input reset does not produce the oscillation stabilization wait time.

### Reference:

Only when a reset request via the  $\overline{\text{RST}}$  pin is generated, a reset cause generated in write operation (An example of such write operation is the MOV instruction which is issued during execution of a transfer instruction.) causes a wait state for release of the reset after completion of the instruction. Thus, the write process terminates normally even if a reset signal is input during write operation.

However, if a string instruction such as MOVS is used, transfer of all the data will not be guaranteed. This is because the instruction accepts a reset before the transfer data for the specified counter value is completed. A reset is accepted also if extension of the bus cycle via the RDY pin continues for 16 machine cycles or more during access to the external bus.

## ● Software reset

Software reset causes an internal reset by writing "0" to the internal reset signal generation bit (RST) of the low-power consumption mode control register (LPMCR). Software reset does not cause the oscillation stabilization wait time.

### Reference:

#### Clock definition

HCLK: Oscillation clock, which is provided via the high speed oscillation pin.

MCLK: Main clock (HCLK frequency divided by 4)

SCLK: Sub clock (a clock which is frequency divided by 4 on the clock provided via the low speed oscillation pin)

$\phi$ : Machine clock (CPU operation clock)

$1/\phi$ : machine cycle (CPU operating clock cycle)

Refer to the Section "5.1 Outline of Clock" for details of the machine clocks.

### Note:

A reset generated in stop or sub clock mode produces an  $2^{17}/\text{HCLK}$  oscillation stabilization wait time (approximately 21.85 ms at the oscillation clock is 6 MHz).

Refer to the Section "5.4 Clock Mode" for details of the clock mode.



## 4.2 Reset Factors and Oscillation Stabilization Wait Times

There are four kinds of reset factors of MB90330A series. The oscillation stabilization wait time varies with the reset cause.

### ■ Reset Factors and Oscillation Stabilization Wait Times

Table 4.2-1 shows the relationship between the reset causes and the oscillation stabilization wait time.

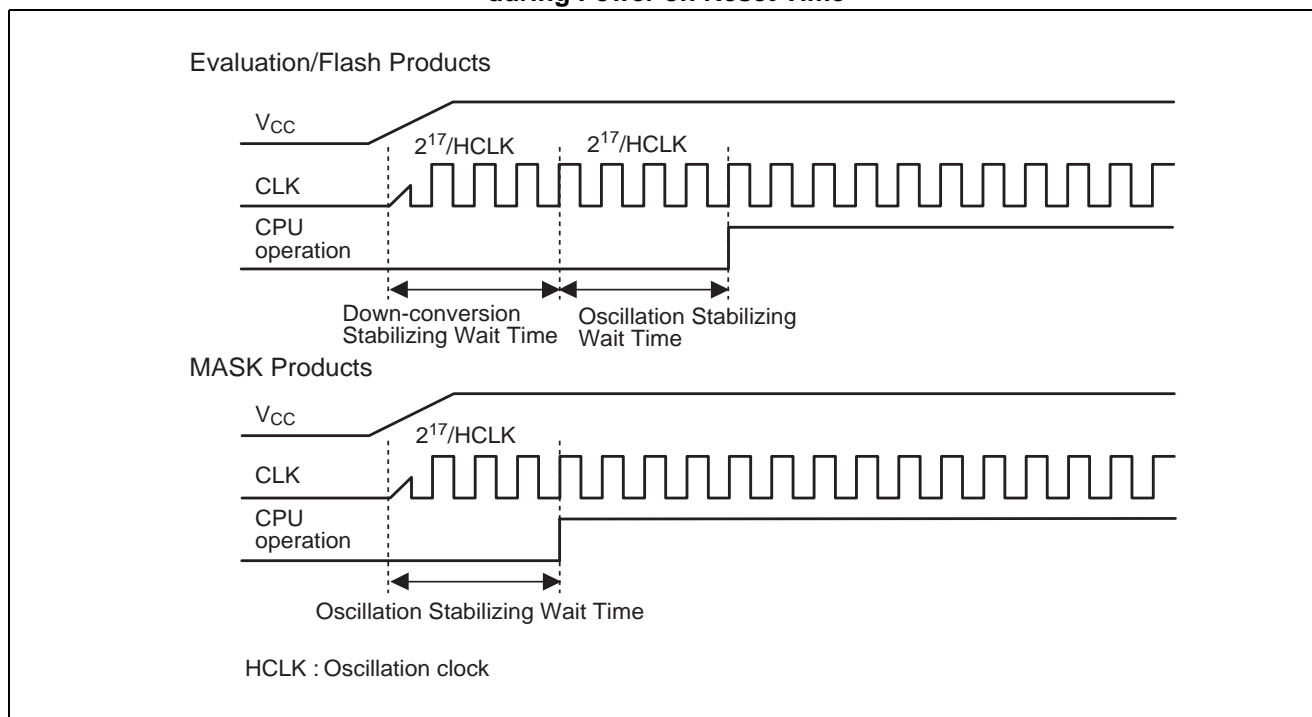
**Table 4.2-1 Reset Factors and Oscillation Stabilization Wait Times**

Reset factor	Oscillation stabilization wait time Each value in parentheses ( ) represents the period for the oscillation clock at 6 MHz.
Power on reset	Evaluation products/flash products: $2^{18}/\text{HCLK}$ (about 43.70 ms). MASK product: $2^{17}/\text{HCLK}$ (about 21.85 ms).
Watchdog timer	None: However, WS1 and WS0 bit are initialized by "11 <sub>B</sub> ".
External reset from $\overline{\text{RST}}$ pin	None: However, WS1 and WS0 bit are initialized by "11 <sub>B</sub> ".
Software reset	None: However, WS1 and WS0 bit are initialized by "11 <sub>B</sub> ".

HCLK: Oscillation clock

WS1, WS0: Bits used to select the oscillation stabilization wait time of the clock selection register (CKSCR).

Figure 4.2-1 shows the oscillation stabilization wait times for the evaluation/flash and MASK products during power on reset time.

**Figure 4.2-1 Oscillation Stabilization Wait Times for the Evaluation/flash and MASK Products during Power on Reset Time****Note:**

For ceramic or quartz oscillators, typically the oscillation stabilization wait time of several to some tens of milliseconds is required until the oscillation is stabilized at the natural oscillation. after it begins. For this reason, set the wait time value meeting the oscillator used.

For details, see Section "5.5 Oscillation Stabilization Wait Time".

**■ Oscillation Stabilization Waiting Reset State**

The reset operation in response to a reset which occurs during power on reset or in stop or sub clock mode begins after passing the oscillation stabilization wait time produced by the time-base timer expires. The reset operation is performed after the external reset is released.

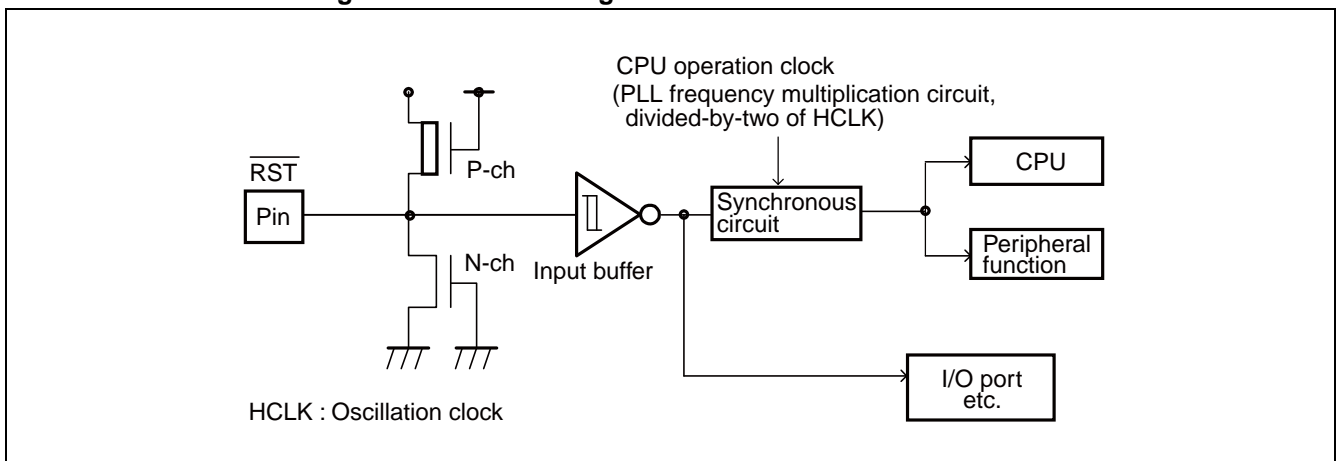
## 4.3 External Reset Pin

The external reset pin ( $\overline{\text{RST}}$  pin), dedicated to reset input pin, generates an internal reset in response to input of the "L" level signal. MB90330A series are reset in synch with the CPU operating clock, except for external pin in asynchronous (generated through ports and so on), which change to the reset state.

### ■ Block Diagram of External Reset Pin

Figure 4.3-1 shows the block diagram of the generation of the internal reset.

Figure 4.3-1 Block Diagram of Internal Reset Generation



#### Note:

To protect the memory from destruction by a reset during write operation, the  $\overline{\text{RST}}$  pin input is accepted with a cycle which does not cause the memory to be destroyed.

Also, the clock is required to initialize the internal circuits. In particular, when using an external clock, the clock must be input during input of the reset.

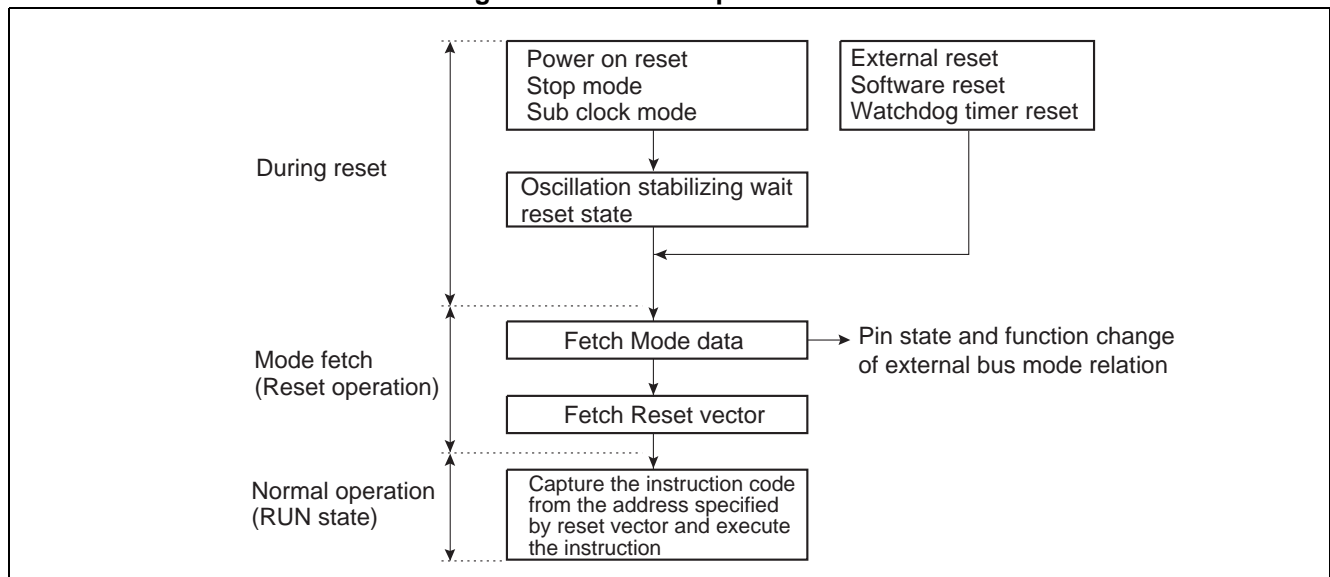
## 4.4 Reset Operation

Once the reset is released, the object from which to read the mode data and reset vector is selected by setting the mode pin, before the mode fetch is performed. This fetch determines the CPU operation mode and the execution activation address succeeding the reset operation. At power on or when recovering from stop mode via a reset, the mode fetch is performed after the oscillation stabilization delay time elapses.

### ■ Overview of Reset Operation

Figure 4.4-1 shows the reset operation flow.

Figure 4.4-1 Reset Operation Flow



### ■ Mode Pin

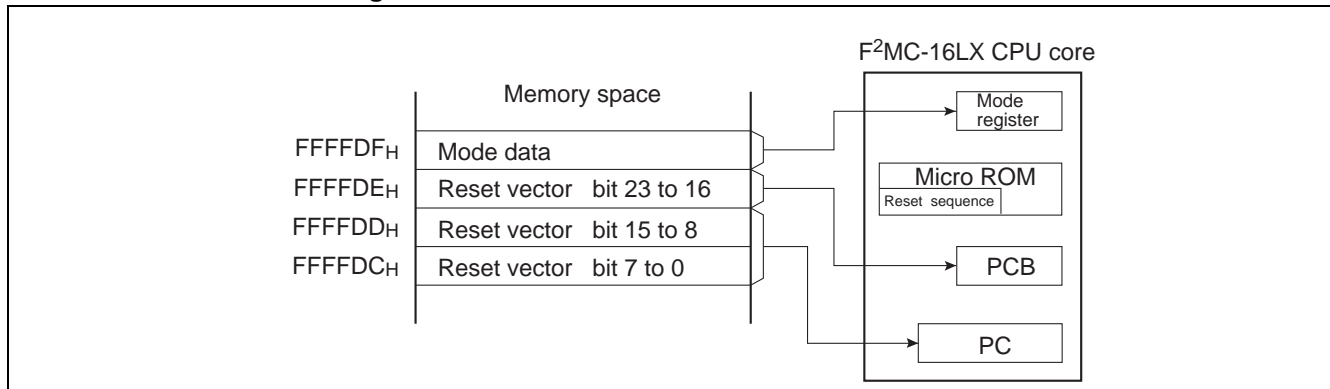
The mode pins (MD2 to MD0) specify the way to fetch the reset vector and mode data. This fetch is performed according to the reset sequence. See Section "7.2 Mode Pins (MD2 to MD0)" for details of the mode pins.

## ■ Mode Fetch

Once the reset is released, the CPU transfers the reset vector and mode data into the appropriate register in the CPU core. The reset vector and mode data are assigned to the four bytes of FFFFDC<sub>H</sub> to FFFFDF<sub>H</sub>. When the reset is released, the CPU immediately outputs these addresses to the bus before fetching the reset vector and mode data. The CPU starts the mode fetch process at the address pointed to by the reset vector.

Figure 4.4-2 shows how the reset vector and mode data are transferred.

**Figure 4.4-2 Reset Vector and Mode Data Transfers**



### Reference:

The object from which to read the reset vector and mode data (either internal ROM or external memory) can be specified using the mode pins. It is recommended that the internal vector mode be specified using the mode pins in single chip mode or internal ROM external bus mode. This is because specifying the external vector mode with the mode pins causes an attempt to read the reset vector and mode data from the external memory, instead of the internal memory.

### ● Mode data (address: FFFFDF<sub>H</sub>)

The contents of the mode register can be modified only by reset operation. The mode register settings will take effect after reset operation. Refer to the Section "7.3 Mode Data" for details of the mode data.

### ● Reset vector (address FFFFDC<sub>H</sub> to FFFFDE<sub>H</sub>)

Write the execution start address after the completion of the reset operation. Execution begins from the address of this content.

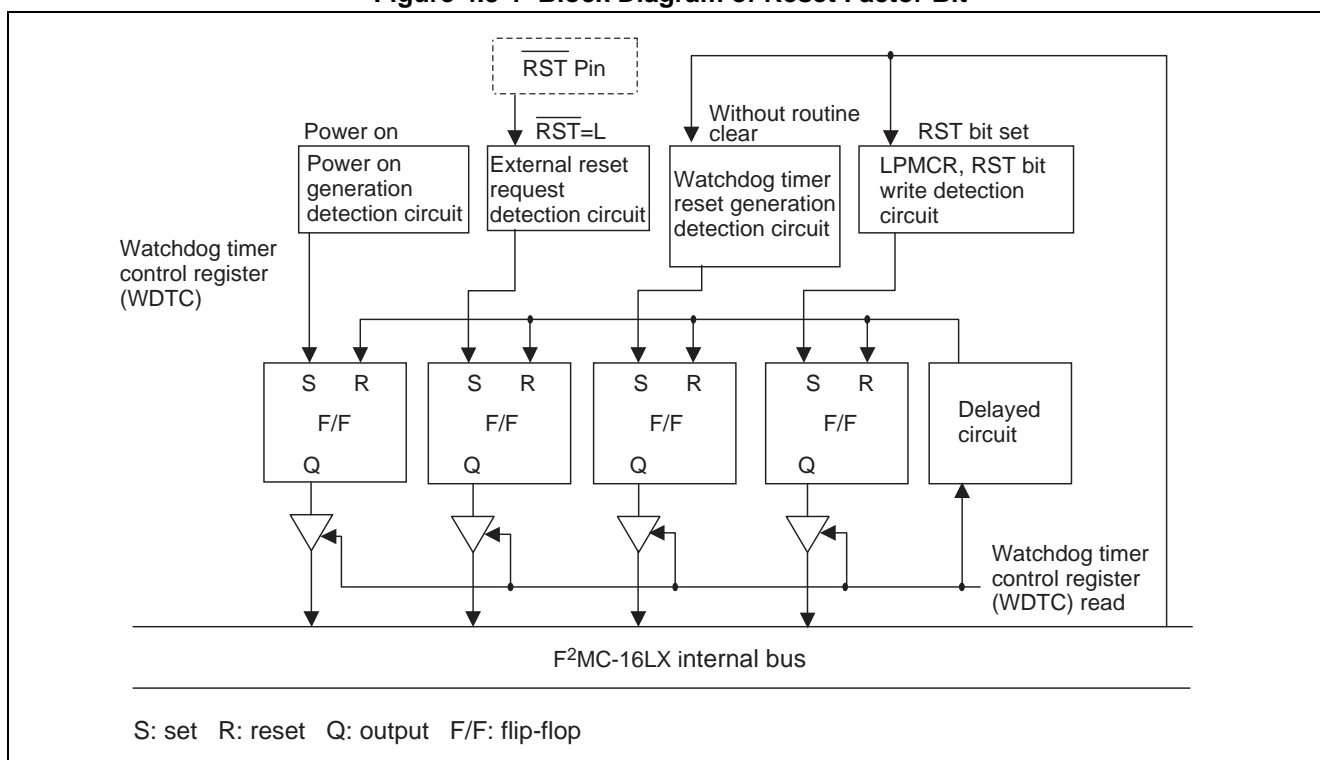
## 4.5 Reset Factor Bit

A reset factor can be identified by reading the watchdog timer control register (WDTC).

### Reset Factor Bit

There are the flip-flop registers associated with respective reset causes, as shown in Figure 4.5-1. The contents of the flip-flops are obtained by reading the watchdog timer control register (WDTC). If the reset cause needs to be identified after the reset has been released, the values read from the WDTC register should be processed by software before control branches to the program.

Figure 4.5-1 Block Diagram of Reset Factor Bit



## ■ Correspondence of Reset Factor Bit and Reset Factor

Figure 4.5-2 shows the configuration of the reset cause bits of the watchdog timer control register (WDTC). Contents of reset cause bits and associated reset causes are shown in the Table 4.5-1.

For details, see the Section "10.2 Watchdog Timer Control Register (WDTC)".

**Figure 4.5-2 Configuration of Reset Factor Bits (Watchdog Timer Control Register)**

Watchdog Timer Control Register (WDTC)									
bit	15 - 8	7	6	5	4	3	2	1	0
0000A8 <sub>H</sub>	(TBTC)	PONR	Reserved	WRST	ERST	SRST	WTE	WT1	WT0
		X	X	X	X	X	1	1	1
		R	-	R	R	R	W	W	W
		Initial value R/W							
		R: Read only W: Write only X: Undefined							

**Table 4.5-1 Correspondence of Reset Factor Bit and Reset Factor**

Reset Factor	PONR	WRST	ERST	SRST
Generating power-on reset	1	X	X	X
Generating watchdog timer overflow	*	1	*	*
Generating External reset request from $\overline{\text{RST}}$ pin	*	*	1	*
Generation of software reset request	*	*	*	1

\*: Previous state held

X: Undefined

## ■ Notes on Reset Factor Bit

### ● At two or more reset factors

If more than one reset cause is generated, the corresponding each reset cause bits of the WDTC register will be set to "1". For example, if an external reset request via the  $\overline{\text{RST}}$  pin and an overflow are generated simultaneously, ERST and WRST of reset cause bits are set to "1".

### ● At power on reset

When a power on reset is generated, the PONR of reset cause bit is set to "1" and all the other reset cause bits are made undefined. For this reason, the software must be designed so that all the reset cause bits other than PONR will be ignored if PONR is "1".

### ● Clearing of reset factor bit

Each of the reset cause bits is cleared only when the values are read from the WDTC register. Once a reset cause is generated, the bit corresponding to it is not cleared even when the reset is generated (a setting of "1" is retained).

### Note:

If power is turned on when a power on reset has not been generated, the WDTC register values may be unprotected.

## 4.6 State of Each Pin at Reset

---

**This section explains the state of each pin at reset.**

---

### ■ Pin Status during Reset

The state of each pin during reset is determined by the settings of the mode pins (MD2 to MD0).

- When internal vector mode has been set: (MD2 to MD0=011<sub>B</sub>)

All I/O, or resource, pins are placed at high impedance. The internal ROM is defined as the object form which to read the mode data.

See the Section "6.7 State of the Pin during Standby Mode, Hold, and Reset" for details of the state of each pin during reset."

### ■ State of Pins after Mode Data Read

The pin state succeeding read of the mode data is determined by the mode data (M1, M0).

- When single chip is selected a mode (M1,M0=00<sub>B</sub>)

All I/O, or resource, pins are placed at high impedance. The internal ROM is defined as the object form which to read the mode data.

---

Note:

For any pin to which place at high impedance when a reset cause is generated, give consideration so that the equipment connected to the pin will not malfunction.

---





# **CHAPTER 5**

---

# **CLOCK**

**This chapter explains the clock of the MB90330A series.**

- 5.1 Outline of Clock
- 5.2 Block Diagram of Clock Generation Section
- 5.3 Clock Select Register (CKSCR)
- 5.4 Clock Mode
- 5.5 Oscillation Stabilization Wait Time
- 5.6 Connection of Oscillator and External Clock

## 5.1 Outline of Clock

---

The clock generation section controls operation of the internal clock which is the operation clock for the CPU and peripheral functions. In this document, the clocks are called as follows according to clock type.

- **Machine clock:** Internal clock.
  - **Main clock:** Oscillation clock frequency divided by 2.
  - **Oscillation clock:** Clock provided via a high-speed oscillation pin.
  - **PLL clock:** Clock generated by internal PLL oscillation.
  - **Sub clock:** A clock frequency divided by 4 on the clock provided via the low speed oscillation pin.
- 

### ■ Overview of Clock

The clock generation section, containing an oscillator circuit, can generate the oscillation and sub clocks by being connected with an external oscillator. Clock generated externally can be input and used as the oscillation clock. The generator, also containing a PLL clock frequency multiplication circuit, can generate three frequency multiplication clocks of the oscillation clock. The clock generation section controls the oscillation stabilization wait time, controls the PLL clock multiplication and controls the internal clock operation by clock switching with the clock selector.

#### ● Oscillation clock (HCLK)

This clock is generated by connecting an oscillator or inputting an external clock to the high-speed oscillation pins.

#### ● Sub clock (SCLK)

It is a clock which operates the watch timer. Moreover, it is available as a low-speed machine clock.

A clock frequency divided by 4 on the clock which is generated by connecting an oscillator or inputting an external clock to the low-speed oscillation pin.

#### ● Main clock (MCLK)

It is an oscillation clock frequency divided by 2 and an input clock to the time-base timer and clock selector.

#### ● PLL clock (PCLK)

An oscillation clock which is obtained by frequency multiplication through the internal PLL clock frequency multiplication circuit (PLL oscillator circuit). Three kinds of clocks can be selected.

#### ● Machine clock ( $\phi$ )

Operation clock for CPU and peripheral functions. One cycle of this clock is used as machine cycle ( $1/\phi$ ). A desired clock can be selected from among the main clock, that is, an oscillation clock frequency divided by 2, the sub clock, and the three frequency multiplication clocks.

---

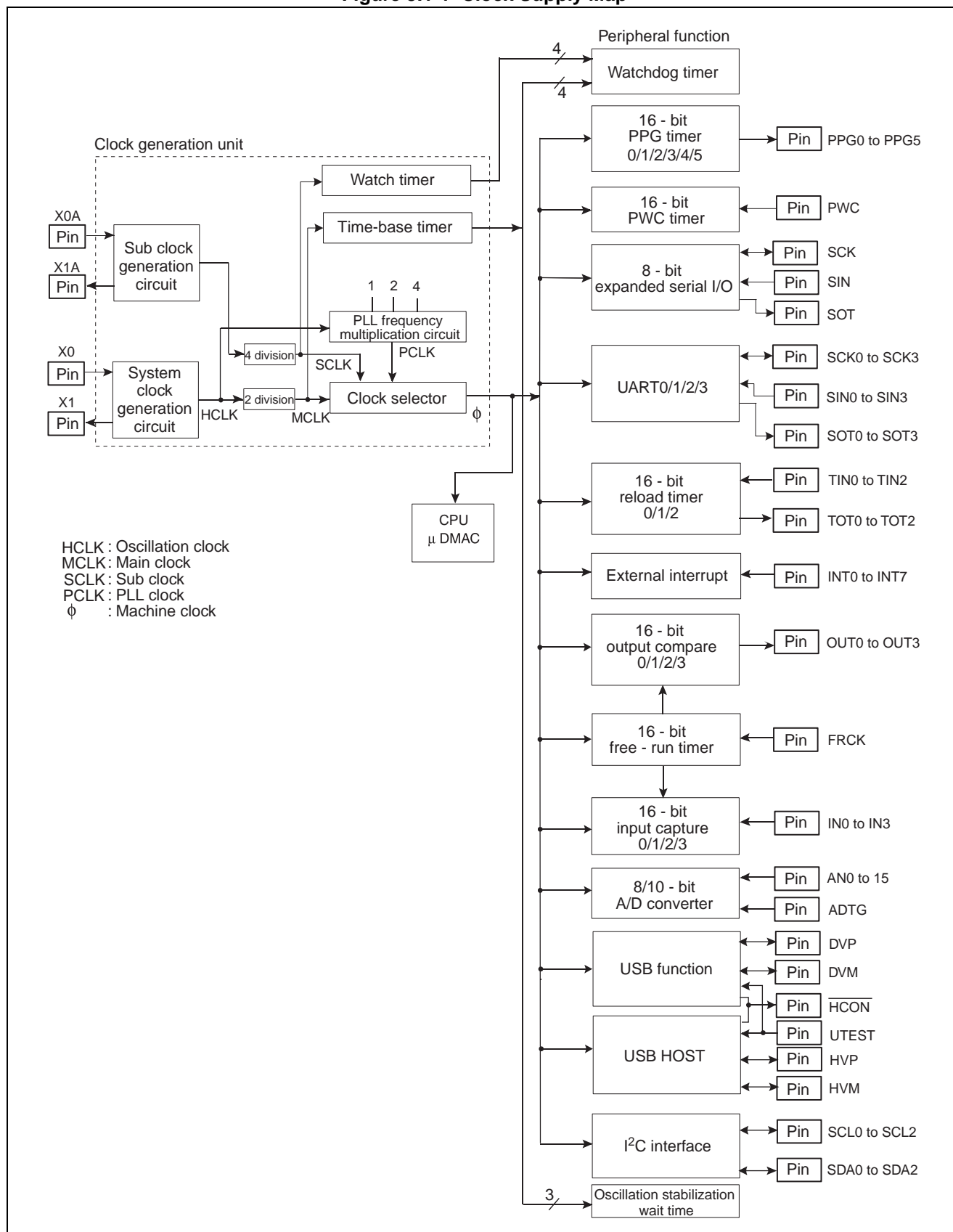
#### Notes:

- As for the oscillation clock, 1 MHz to 7 MHz can oscillate. The maximum operating frequency is 24 MHz for the CPU and peripheral functions. When multiplier exceeding the maximum operating frequency is specified, the device does not operate correctly. If the source oscillation at a frequency of 6 MHz, only 4-time frequency multiplication can be specified.
  - To use the USB HOST or USB function, the PLL clock mode must have been set.
- 

#### ■ Clock Supply Map

Machine clocks generated by the clock generation section are supplied as operating clocks of the CPU and peripheral function. For this reason, operation of the peripheral functions is influenced by switching clock mode between the main and PLL clock or switching the PLL clock frequency multiplication rate. The frequency divided output from the time-base timer is provided to some peripheral functions, thereby enabling each periphery to select its operating clock. Figure 5.1-1 contains the map showing how the clocks are provided.

**Figure 5.1-1 Clock Supply Map**



## 5.2 Block Diagram of Clock Generation Section

---

The clock generation section consists of the following six blocks:

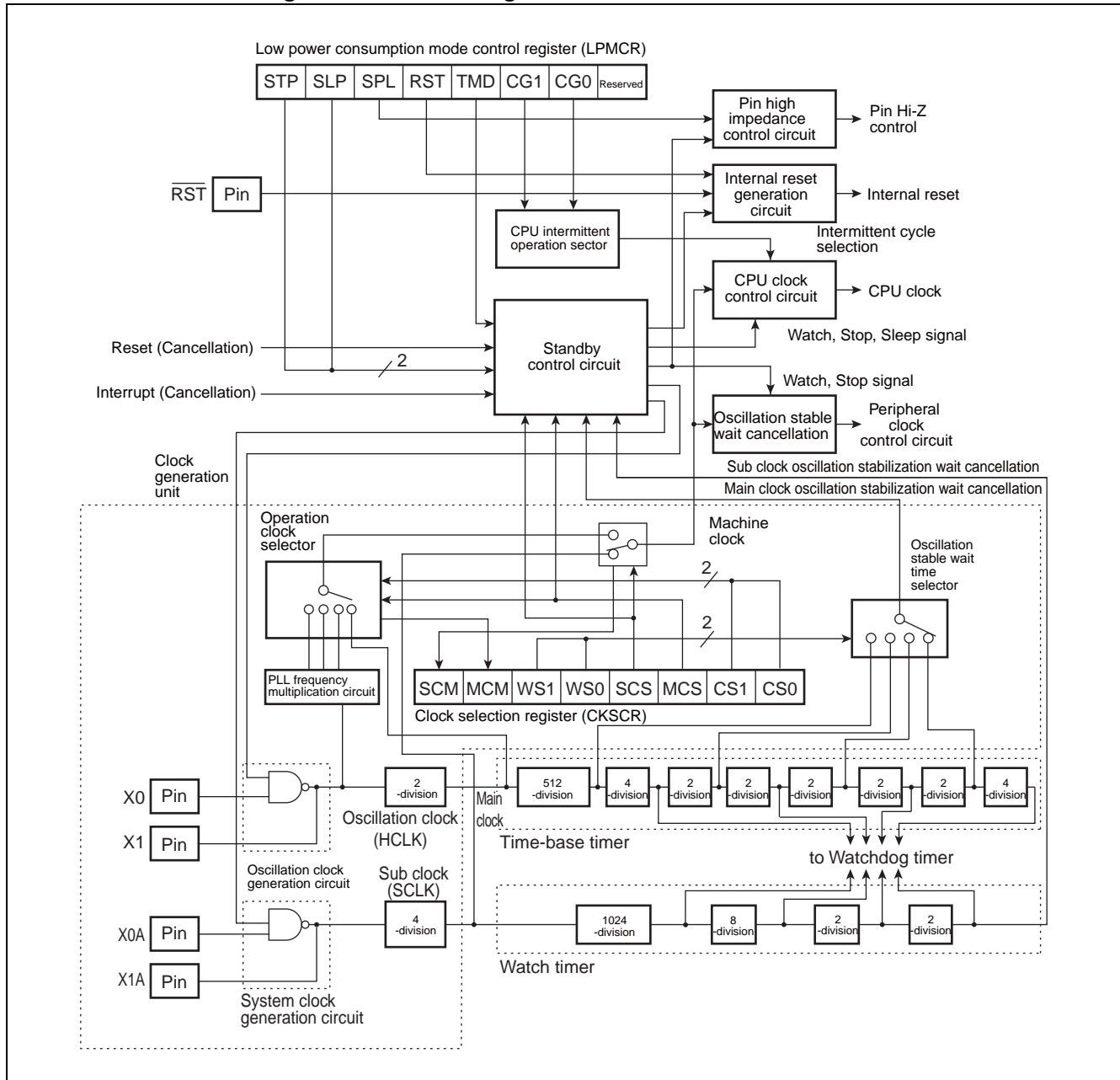
- System clock generation circuit
  - Sub clock generation circuit
  - PLL multiplying circuit
  - Clock selector
  - Clock select register (CKSCR)
  - Oscillation stabilization wait time selector
- 

### ■ Block Diagram of Clock Generation Section

Figure 5.2-1 shows the block diagram of the clock generation section.

Figure 5.2-1 contains the standby control and time-base timer circuits.

Figure 5.2-1 Block Diagram of Clock Generation Section



● System clock generation circuit

Generates an oscillation clock (HCLK) using the oscillator connected to the high-speed oscillation pin. It is also possible to input an external clock.

● Sub clock generator circuit

Generates the sub clock (SCLK) using the oscillator connected to the low-speed oscillation pin. It is also possible to input an external clock.

- PLL multiplying circuit

The oscillation clock is multiplied by PLL oscillation and supplied to the CPU clock selector.

- Clock selector

From the main and sub clocks, and the three PLL clocks, this selects the clock to be supplied to the CPU and periphery clock control circuits.

- Clock select register (CKSCR)

Switches between the oscillation and PLL clocks, selects the oscillation stabilization wait time, and selects the PLL clock multiplier.

- Oscillation stabilization wait time selector

A circuit which selects the oscillation stabilization wait time for the oscillation clock succeeding release of the stop mode or a watchdog reset. Four time-base timer outputs are selected.



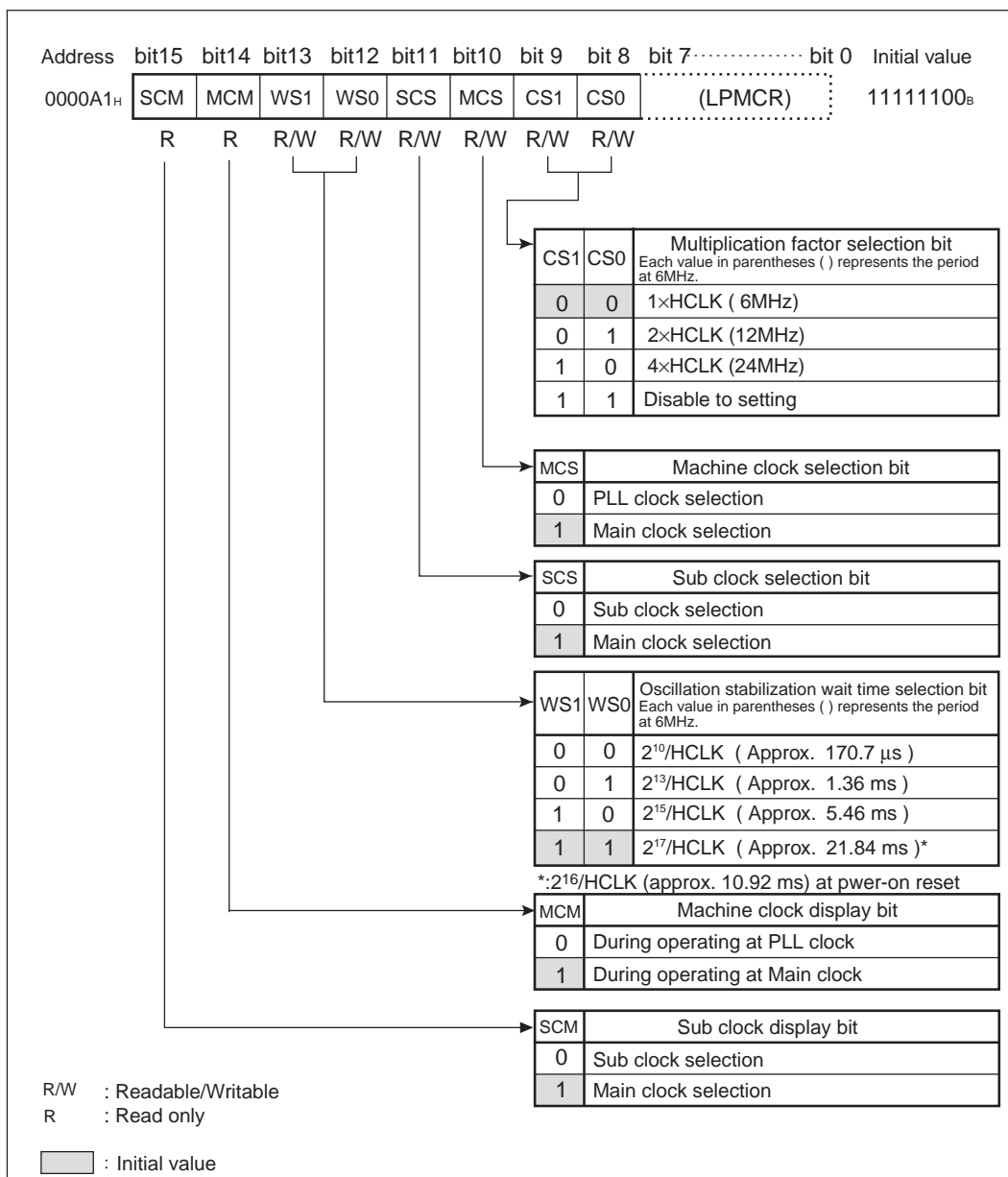
## 5.3 Clock Select Register (CKSCR)

The clock select register (CKSCR) switches the clock mode between the main, sub, and PLL clocks, and selects the oscillation stabilization wait time and the PLL clock frequency multiplier.

### ■ Configuration of Clock Select Register (CKSCR)

Figure 5.3-1 shows the clock selection register (CKSCR) configuration. Table 5.3-1 summarizes the functions of the clock selection register bits.

Figure 5.3-1 Configuration of Clock Select Register (CKSCR)



Note:

The machine clock selection bit (MCS) is initialized by reset to main clock selection.

**Table 5.3-1 Functions of Clock Select Register (CKSCR) Bits (1 / 2)**

Bit name		Functions
bit 15	SCM: Sub clock display bit	<ul style="list-style-type: none"> <li>• Bit indicating the main clock or sub clock, whichever selected as the machine clock.</li> <li>• A "0" in this bit indicates that the sub clock has been selected. A "1" in the bit indicates that the main or PLL clock has been selected.</li> <li>• If SCS = 1 and SCM = 0, now is the main clock oscillation stabilization wait time.</li> </ul>
bit 14	MCM: PLL clock display bit	<ul style="list-style-type: none"> <li>• Bit indicating the main or PLL clock, whichever selected as the machine clock.</li> <li>• A "0" in this bit indicates that the PLL clock has been selected. A "1" in the bit indicates that the main or sub clock has been selected.</li> <li>• If the PLL clock selection bit (MCS) = 0 and MCM = 1, now is the PLL clock oscillation stabilization wait time.</li> </ul>
bit 13, bit 12	WS1, WS0: Oscillation stabilization wait time selection bits	<ul style="list-style-type: none"> <li>• These bits are used to select an oscillation stabilization wait time required for the oscillation clock when the stop mode is canceled, when transition occurs from sub clock mode to main clock mode, or when transition occurs from sub clock mode to PLL clock.</li> <li>• Initialized to "11<sub>B</sub>" by every reset cause.</li> </ul> <p>Note: The oscillation stabilization wait time must be set to a suitable value for the oscillator to use. See Section "4.2 Reset Factors and Oscillation Stabilization Wait Times". Please set the setting of "00<sub>B</sub>" only at the main clock mode.</p> <p>Reference: The oscillation stabilization wait time for PLL clock is fixed to <math>2^{14}/\text{HCLK}</math>.</p>
bit 11	SCS: Sub clock selection bit	<ul style="list-style-type: none"> <li>• Bit for specifying main or sub clock as the machine clock.</li> <li>• A "0" in this bit selects the sub clock. A "1" in the bit selects the main clock.</li> <li>• If "1" is written to this bit when it is "0", the main clock oscillation stabilization wait time is produced, thereby clearing the time-base timer automatically.</li> <li>• When the sub clock is selected, it is used as the operating clock. (If the low-speed oscillation is 32 kHz, the machine clock is 8 kHz.)</li> <li>• When both SCS and MCS are "0", SCS has priority and sub clock is selected.</li> <li>• Initialized to "1" by every reset cause.</li> </ul>
bit 10	MCS: PLL clock selection bit	<ul style="list-style-type: none"> <li>• Bit for selecting main or PLL clock as the machine clock.</li> <li>• A "0" in this bit selects the PLL clock. A "1" in the bit selects the main clock.</li> <li>• If "0" is written to this bit when it is "1", the PLL clock oscillation stabilization wait time is produced, thereby clearing the time-base timer automatically. Also, the interrupt request flag bit (TBOF) of the time-base timer control register (TBTC) is cleared.</li> <li>• The PLL clock oscillation stabilization wait time is fixed to <math>2^{14}/\text{HCLK}</math>. (If the oscillation clock is 6 MHz, the oscillation stabilization wait time will be approximately 2.73 ms.)</li> <li>• If the main clock is selected, the operating clock frequency will be the oscillation clock frequency-divided by 2. (If the oscillation clock is 6 MHz, the operating clock will be 3 MHz.)</li> <li>• Initialized to "1" by every reset cause.</li> </ul> <p>Note: To write "0" if when MCS bit is "1", make sure that the time-base timer interrupt have been masked using the TBTC register interrupt request enable bit (TBIE) or interrupt level mask register (ILM).</p>

**Table 5.3-1 Functions of Clock Select Register (CKSCR) Bits (2 / 2)**

Bit name		Functions
bit 9, bit 8	CS1, CS0: Multiplier selection bit	<ul style="list-style-type: none"> <li>• Bit for selecting the multiply factor for PLL clock.</li> <li>• The multiplier can be selected from among three options.</li> <li>• Initialized to "00<sub>B</sub>" by every reset cause.</li> </ul> <p>Note: When the MCS or MCM is "0", writing to these bits is not allowed. Rewrite the CS1 and CS0 bits after setting the MCS bit to "1" once. "11<sub>B</sub>" is a set interdiction.</p>

HCLK: Oscillation clock

## 5.4 Clock Mode

---

The clock modes are the main clock mode, PLL clock mode and sub clock mode.

---

### ■ Main Clock Mode, PLL Clock Mode, Sub Clock Mode

- Main clock mode

The main clock mode stops the PLL clock by using an oscillation clock frequency divided by 2 as the operating clock of the CPU and peripheral resources.

- PLL clock mode

The PLL clock mode uses the PLL clock as operation clock for the CPU and peripheral functions. The PLL clock frequency multiplier factor can be selected using the clock selection register (CKSCR).

---

Note:

To use the USB HOST or USB function, the PLL clock mode must have been set.

---

- Sub clock mode

The sub clock mode stops the main and PLL clocks by using the sub clock as the operating clock of the CPU and peripheral resources.

### ■ Transition of Clock Mode

Writing the PLL clock selection bit (MCS) or sub clock selection bit (SCS) of the CKSCR changes the clock mode to main, PLL, or sub.

- Transition from main clock mode to PLL clock mode

If the CKSCR of MCS bit is rewritten from "1" to "0" while in the main clock mode, the mode will change from main to PLL after the PLL clock oscillation stabilization wait time ( $2^{14}/\text{HCLK}$ ).

- Transition from PLL clock mode to main clock mode

If the CKSCR of MCS bit is rewritten from "0" to "1" while in the PLL clock mode, the mode will change from PLL to main when the PLL and main clock edges will match (after 1 to 8 PLL clocks).

- Transition from main clock mode to sub clock mode

If the CKSCR of SCS bit is rewritten from "1" to "0" while in the main clock mode, the mode will change from main to sub.

- Transition from sub clock mode to main clock mode

If the CKSCR of SCS bit is rewritten from "0" to "1" while in the sub clock mode, the mode will change from sub to main after the main clock oscillation stabilization wait time. The oscillation stabilization wait time can be selected using the oscillation stabilization wait time selection bits (WS1, WS0) of the CKSCR.

- Transition from PLL clock mode to sub clock mode

If the sub clock selection bit (SCS) of the clock selection register (CKSCR) is rewritten from "1" to "0" while in the PLL clock mode, the mode changes from PLL clock to sub clock.

- Transition from sub clock mode to PLL clock mode

If the SCS bit of the CKSCR is rewritten from "0" to "1" while in the sub clock mode, the mode changes from sub clock to PLL clock after the main clock oscillation stabilization wait time. The oscillation stabilization wait time can be selected using the oscillation stabilization wait time selection bits (WS1, WS0) of the CKSCR.

---

Note:

Even if the PLL clock selection (MCS) or SCS bit of the CKSCR is rewritten, the machine clock is not switched immediately. If a resource which depends on the machine clock needs to be operated, refer the PLL clock indicator bit (MCM) and sub clock indicator bit (SCM) of the CKSCR to check that the machine clock has been switched, before beginning the operation.

If both the SCS and MCS bits are "0", the higher priority is given to SCS and the sub clock mode is entered.

If the clock mode is switched, the mode must not be switched to another clock mode or the low-power consumption mode until completion of the switching. Completion of the switching can be checked by referring the MCM and SCM bits of the CKSCR.

---

## ■ Selection of PLL Clock Multiplication Rate

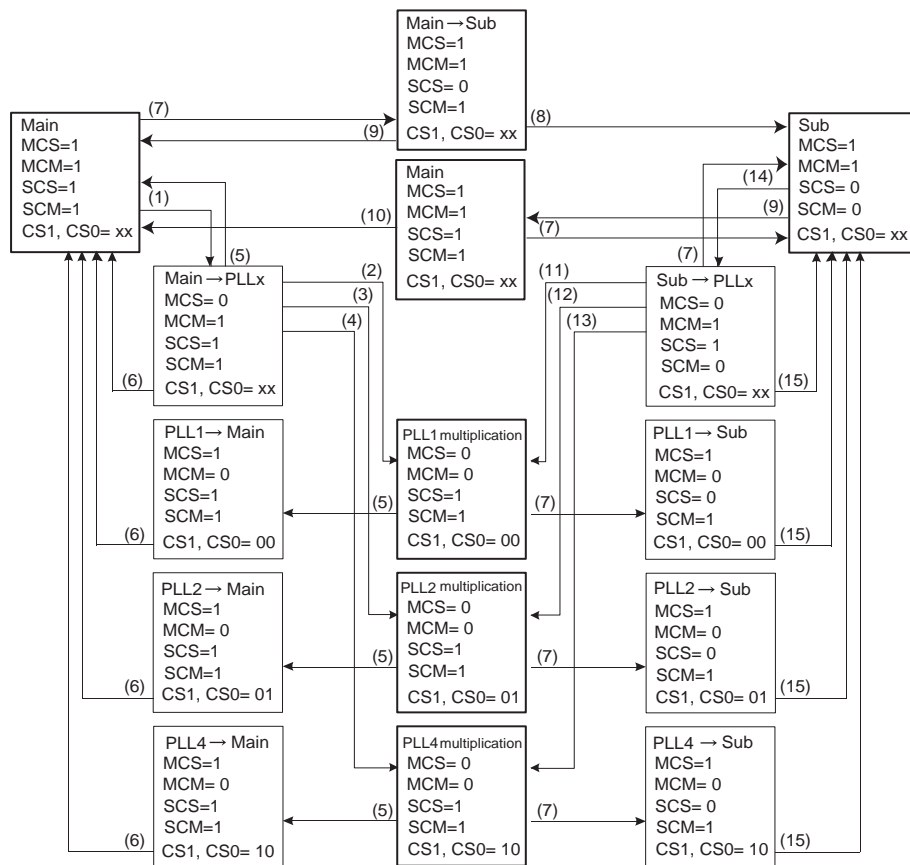
One of the three PLL clock frequency multiplier from 1 to 4 times can be selected by writing "00<sub>B</sub>" to "10<sub>B</sub>" to the CS1 and CS0 bits of the CKSCR. "11<sub>B</sub>" is a set interdiction.

## ■ Machine Clock

The PLL clock, main clock, and sub clock output from the PLL multiplying circuit are used as machine clocks. These machine are clocks supplied to the CPU or peripheral function. One of the main, PLL, and sub clock modes can be selected by writing the CKSCR of MCS and SCS bit.

Figure 5.4-1 shows the transition chart of the machine clock selection.

Figure 5.4-1 State Transition Diagram of Machine Clock Selection



- (1) MCS bit "0" write
- (2) PLL clock oscillation stabilization wait end & CS1, CS0=00
- (3) PLL clock oscillation stabilization wait end & CS1, CS0=01
- (4) PLL clock oscillation stabilization wait end & CS1, CS0=10
- (5) MCS bit "1" write (within Watchdog reset)
- (6) Synchronous timing of PLL clock and Main clock
- (7) SCS bit "0" write
- (8) Sub clock oscillation stabilization wait end (max.  $2^{15}/\text{SCLK}$ )
- (9) SCS bit "1" write
- (10) Main clock oscillation stabilization wait end
- (11) Main clock oscillation stabilization wait end & CS1, CS0=00
- (12) Main clock oscillation stabilization wait end & CS1, CS0=01
- (13) Main clock oscillation stabilization wait end & CS1, CS0=10
- (14) SCS bit "1" write, MCS bit "0" write
- (15) Synchronous timing of PLL clock and Sub clock

MCS : PLL clock selection bit of Clock selection register (CKSCR)  
 MCM : PLL clock display bit of Clock selection register (CKSCR)  
 SCS : Sub clock selection bit of Clock selection register (CKSCR)  
 SCM : Sub clock display bit of Clock selection register (CKSCR)  
 CS1, CS0 : Multiplication factor selection bit of Clock selection register (CKSCR)

**Note:**

The initial value for machine clock is the main clock (MCS = 1, SCS = 1).

If both the SCS and MCS bits are "0", the higher priority is given to SCS and the sub clock mode is entered.

When the mode needs to be switched from sub to PLL, the oscillation stabilization wait time selection bits (WS1, WS0) of CKSCR must be set to "01<sub>B</sub>", "10<sub>B</sub>", or "11<sub>B</sub>".

## 5.5 Oscillation Stabilization Wait Time

When power is turned on, the stop mode is quit, or the clock mode changes from the sub to main or PLL clock, the oscillation stabilization wait time is required after the oscillation begins. This is because the oscillation of the oscillation clock remains in stopped state. Also when the clock mode changes from main to PLL, the oscillation stabilization wait time is required after the PLL oscillation begins.

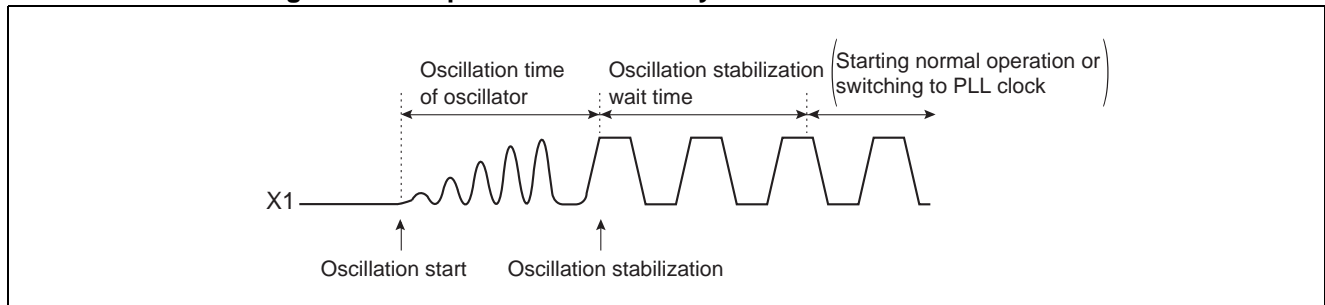
### ■ Oscillation Stabilization Wait Time

In general, ceramic and crystal oscillators require several milliseconds to some tens of milliseconds until they provide stable oscillation at their natural frequency (oscillation frequency). Accordingly, CPU operation is disabled immediately after the oscillation starts and the clock supply to the CPU is enabled until the oscillation stabilization wait time has elapsed. This gives the oscillation time to stabilize. It is necessary to select an oscillation stabilization wait time appropriate to an oscillator to be used because the oscillation stabilization time depends on the type of oscillator (crystal, ceramic, etc.). The oscillation stabilization wait time can be selected by setting in the clock selection register (CKSCR).

When changing the clock mode from main to PLL, the CPU operates with the main clock during the PLL oscillation stabilizing wait time before the mode actually changes to PLL.

Figure 5.5-1 shows the operation immediately after starting the oscillation.

Figure 5.5-1 Operation Immediately after the Start of Oscillation



## 5.6 Connection of Oscillator and External Clock

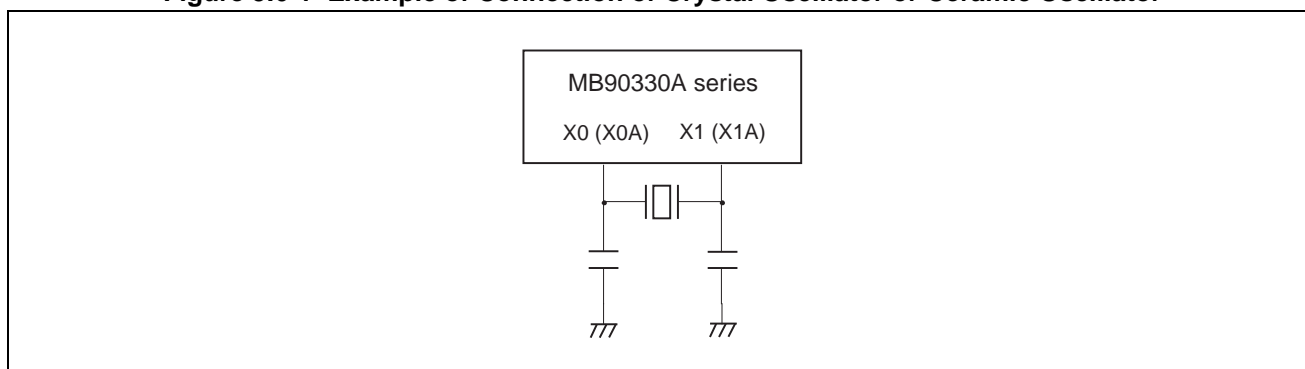
MB90330A series, containing a system clock generator circuit, generates the clock with the oscillator connected externally. It is also possible to input an externally generated clock.

### ■ Connection of Oscillator and External Clock

#### ● Example of connection of crystal oscillator or ceramic oscillator

Connect the crystal or ceramic oscillator as shown in Figure 5.6-1.

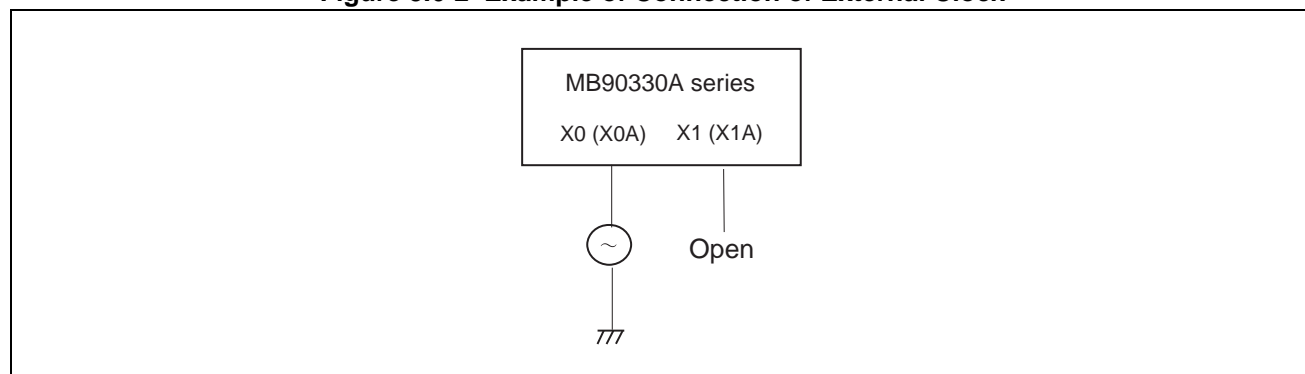
**Figure 5.6-1 Example of Connection of Crystal Oscillator or Ceramic Oscillator**



#### ● Example of connection of external clock

As shown in Figure 5.6-2, connect the external clock to the X0 pin and open the X1 pin.

**Figure 5.6-2 Example of Connection of External Clock**







# **CHAPTER 6**

---

# **LOW-POWER CONSUMPTION MODE**

**This chapter describes the low-power consumption mode of the MB90330A series.**

- 6.1 Outline of Low-Power Consumption Mode
- 6.2 Block Diagram of Low-power Consumption Control Circuit
- 6.3 Low-power Consumption Mode Control Register (LPMCR)
- 6.4 CPU Intermittent Operation Mode
- 6.5 Standby Mode
- 6.6 State Transition Diagram
- 6.7 State of the Pin during Standby Mode, Hold, and Reset
- 6.8 Precautions when Using Low-power Consumption Mode

## 6.1 Outline of Low-Power Consumption Mode

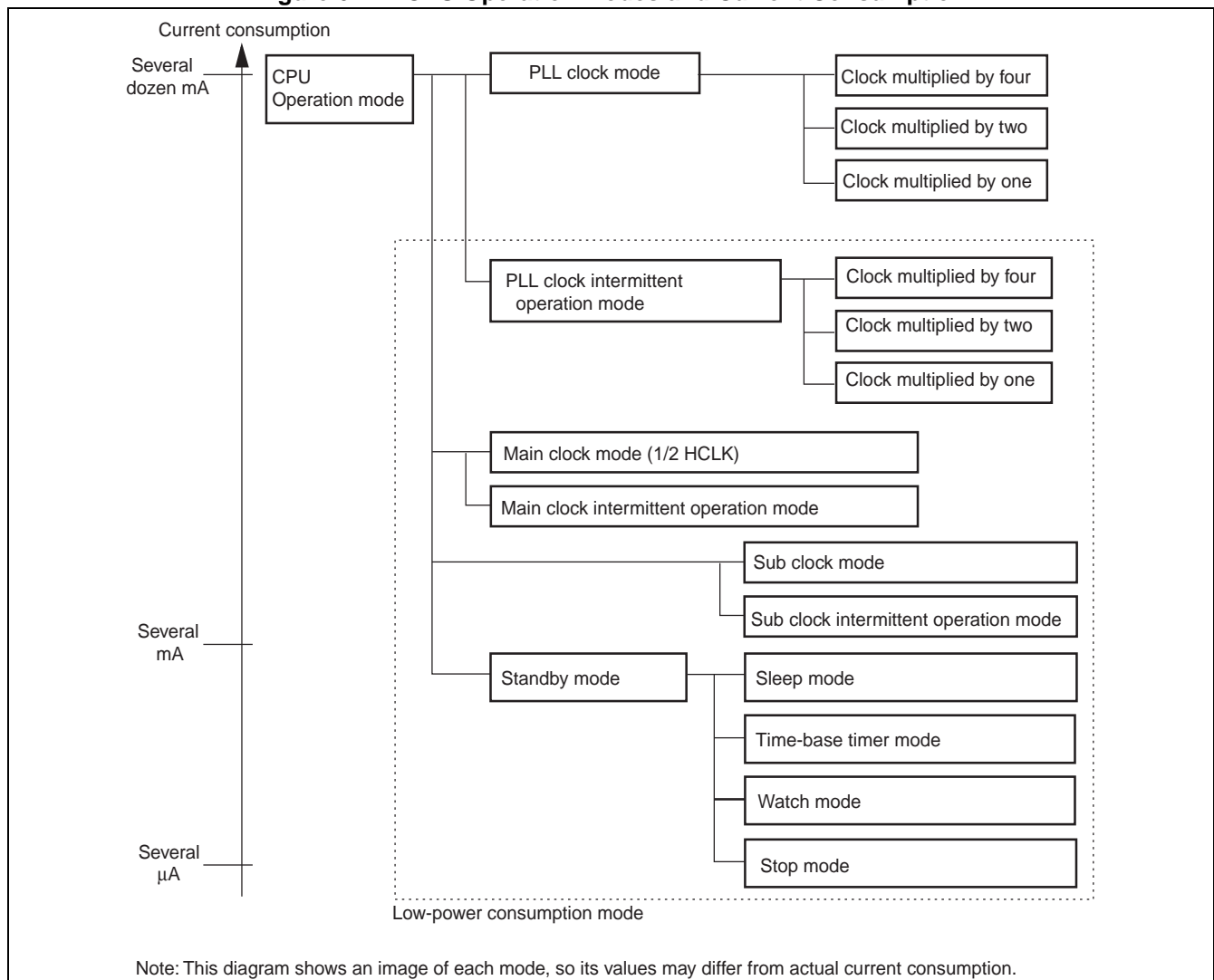
The MB90330A series have the following CPU operation modes by selecting the operation clock and operating the control of the clock.

- **Clock mode**  
(PLL clock mode, main clock mode, and sub clock mode)
- **CPU Intermittent operation mode**  
(PLL clock intermittent operation mode, main clock intermittent operation mode, and sub clock intermittent operation mode)
- **Standby mode**  
(sleep mode, time-base timer mode, and stop, watch mode)

### ■ CPU Operation Modes and Current Consumption

Figure 6.1-1 shows the relationships between the CPU operation modes and current consumption.

**Figure 6.1-1 CPU Operation Modes and Current Consumption**



#### ■ Clock Mode

##### ● PLL clock mode

In PLL clock mode, the CPU and peripheral function operate on a PLL multiplying clock of oscillation clock (HCLK).

---

#### Note:

When using USB HOST and the USB function, you need to set to the PLL clock mode.

---

##### ● Main clock mode

In main clock mode, the CPU and peripheral function operate on a clock with 2-frequency division of oscillation clock (HCLK). In this mode, the PLL multiplying circuit stops.

##### ● Sub clock mode

In sub clock mode, the CPU and peripheral function operate on a sub clock (SCLK). In this mode, the main clock and PLL multiplying circuit stop.

---

#### Reference:

For the clock mode, see Section "5.4 Clock Mode".

---

#### ■ CPU Intermittent Operation Mode

The CPU intermittent mode operates the CPU intermittently and lowers the power consumption while supplying the high-speed clock to the peripheral functions. The CPU intermittent operation mode is a mode for supplying intermittent clock only to the CPU when it makes access to the registers, internal memory, peripheral functions, or external devices.

#### ■ Standby Mode

Standby mode reduces the power consumption by the low-power consumption control circuit which stops clock supply to the CPU (sleep mode), stops clock supply to the CPU and peripheral functions (time-base timer mode), or stops the oscillation clock (stop mode).

##### ● PLL sleep mode

The PLL sleep mode terminates the CPU operation clock in the PLL clock mode and operates components except for CPU under the PLL clock.

##### ● Main sleep mode

The main sleep mode terminates the CPU operation clock in the main clock mode and operates components except for the CPU under the main clock.

- Sub sleep mode

The sub sleep mode terminates the CPU operation clock in the sub clock mode and operates components except for the CPU under the sub clock.

- Time-base timer mode

The time-base timer mode terminates all the operations other than the oscillation clock, the time-base timer, and the clock timer, terminating all the functions other than the time-base timer and the watch timer.

- Watch mode

It is a mode by which only the clock timer is operated. Only the sub clock operates and the main clock and PLL multiplication circuit stops.

- Stop mode

Stop mode is mode for stopping original oscillation and all functions are stopped.

---

Note:

In the stop mode, data is kept at the lowest power consumption since the oscillation clock is terminated.

When you change the clock mode, be sure not to shift to other clock modes or the low-power consumption mode until you complete the clock mode change. Completion of the switching can be checked by referring the MCM and SCM bits of the CKSCR.

The mode transits to the standby mode is also inhibited during the USB is transferred.

---

## 6.2 Block Diagram of Low-power Consumption Control Circuit

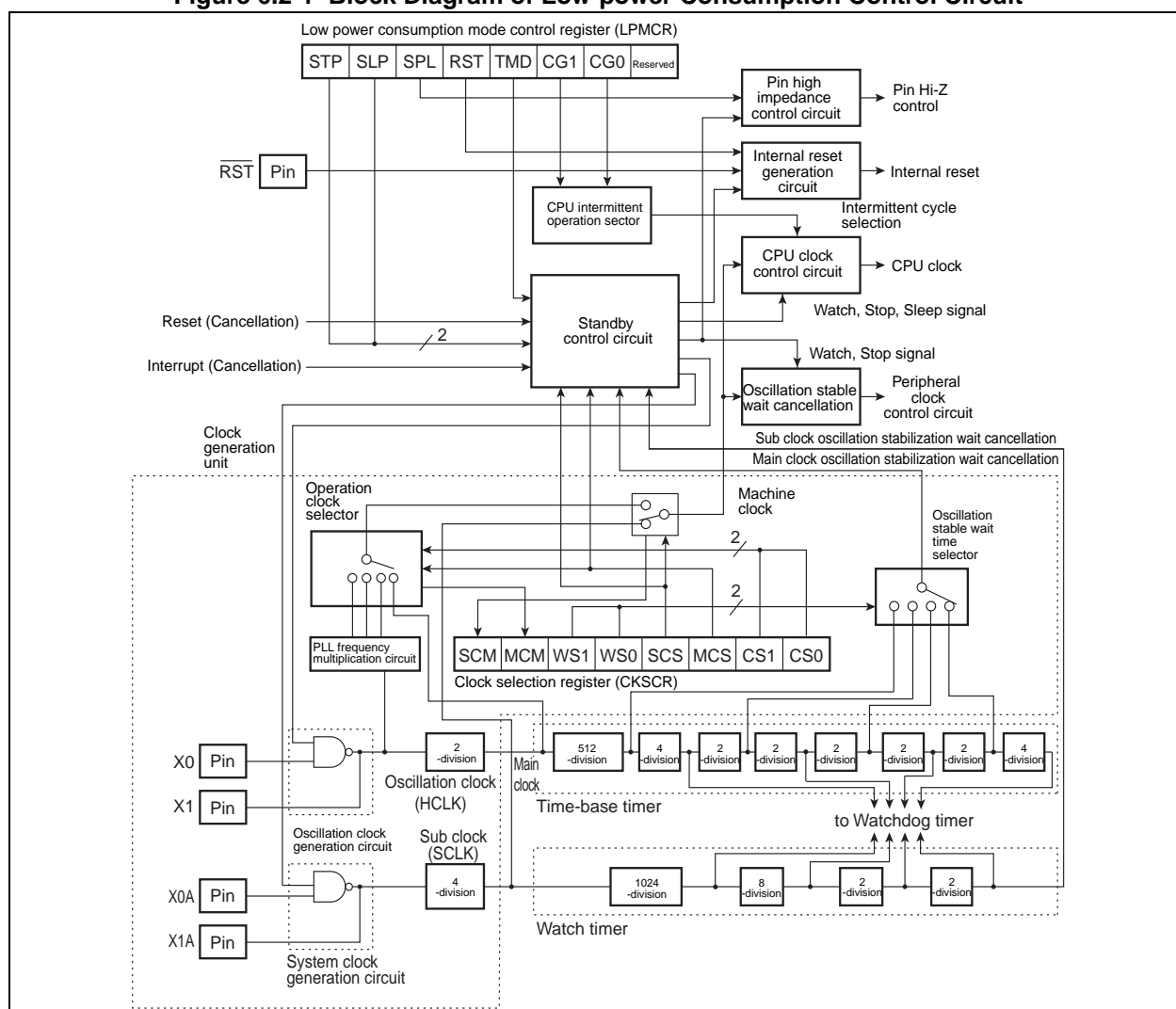
The low-power consumption control circuit is composed of the following seven blocks.

- CPU intermittent operation selector
- Standby controller circuit
- CPU clock controller circuit
- Peripheral clock controller circuit
- Pin high-impedance controller circuit
- Internal reset generator circuit
- Low-power consumption mode control register (LPMCR)

### ■ Block Diagram of Low-power Consumption Control Circuit

Figure 6.2-1 shows the block diagram of the low-power consumption control circuit.

Figure 6.2-1 Block Diagram of Low-power Consumption Control Circuit



● CPU intermittent operation selector

The CPU intermittent operation sector selects the number of the suspended clocks in the CPU intermittent operation mode.

● Standby controller circuit

The standby controller circuit controls the CPU clock control circuit and the peripheral clock control circuit, and then performs the transition to the low-power consumption mode or cancellation.

● CPU clock controller circuit

The CPU clock controller circuit controls the clock supplied to CPU.

● Peripheral clock controller circuit

The peripheral clock controller circuit controls the clock supplied to the peripheral functions.

● Pin high-impedance controller circuit

The pin high-impedance controller circuit makes the external pin to the high-impedance while the mode is in the time-base timer mode and the stop mode. For the pin to which pull-up option is selected, you need to cut the pull-up resistance during the stop mode.

● Internal reset generator circuit

This circuit generates internal reset signals.

● Low-power consumption mode control register (LPMCR)

The low-power consumption mode control register (LPMCR) shifts to/cancels the standby mode or sets the CPU intermittent operation function.

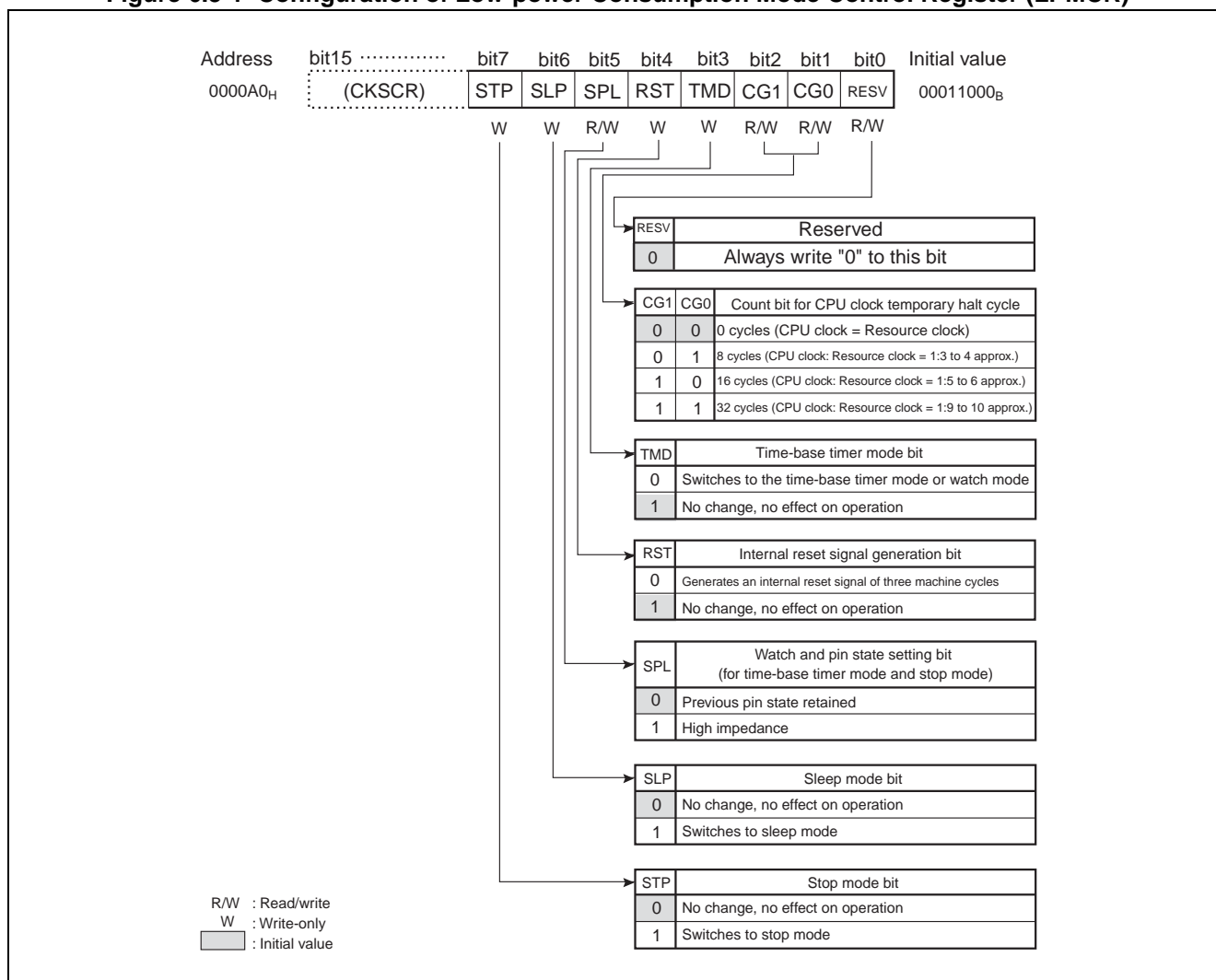
## 6.3 Low-power Consumption Mode Control Register (LPMCR)

The low-power consumption mode control register (LPMCR) performs transition to/cancellation of the low-power consumption mode or sets the number of the CPU clock suspend cycles in the CPU intermittent operation mode.

### ■ Low-power Consumption Mode Control Register (LPMCR)

Figure 6.3-1 shows the configuration of the low-power consumption mode control register (LPMCR).

**Figure 6.3-1 Configuration of Low-power Consumption Mode Control Register (LPMCR)**





**Table 6.3-1 Function Description of Each Bit of Low-power Consumption Mode Control Register (LPMCR)**

Bit name		Functions
bit 7	STP: Stop mode bit	<ul style="list-style-type: none"> <li>• This bit indicates the transition to the stop mode.</li> <li>• Writing "1" to this bit changes the stop mode.</li> <li>• Writing "0" to this bit has no influence on operation.</li> <li>• Cleared to "0" by generation of a reset or interrupt request.</li> <li>• The bit always returns "0" when read.</li> </ul>
bit 6	SLP: Sleep mode bit	<ul style="list-style-type: none"> <li>• This bit indicates the transition to the sleep mode.</li> <li>• Writing "1" to this bit changes the sleep mode.</li> <li>• Writing "0" to this bit has no influence on operation.</li> <li>• Cleared to "0" by generation of a reset or interrupt request.</li> <li>• The bit always returns "0" when read.</li> </ul>
bit 5	SPL: Pin state specification bit (At the stop mode of the watch and the time-base timer)	<ul style="list-style-type: none"> <li>• Valid only while in the watch mode, time-base timer, or stop mode.</li> <li>• A "0" in this bit causes the clock to remain at the level of the external pin.</li> <li>• An external pin is made high impedance in case of "1".</li> <li>• The bit is initialized to "0" at a reset.</li> </ul>
bit 4	RST: Internal reset signal generation bit	<ul style="list-style-type: none"> <li>• Writing "0" to this generator generates the internal reset signal three machine cycle.</li> <li>• Writing "1" to this bit has no influence on operation.</li> <li>• The bit always returns "1" when read.</li> </ul>
bit 3	TMD: Clock, time-base timer mode bit	<ul style="list-style-type: none"> <li>• This bit indicates the transition to the watch and time-base timer mode.</li> <li>• Writing "0" to this bit while in main or PLL clock mode changes the time-base timer mode.</li> <li>• Writing "0" to this bit while in sub clock mode changes the watch mode.</li> <li>• Initialized to "1" by generation of a reset or interrupt request.</li> <li>• The bit always returns "1" when read.</li> </ul>
bit 2, bit 1	CG1,CG0: Temporary CPU clock stop Cycle count selection bit	<ul style="list-style-type: none"> <li>• Bit used for setting the number of suspension cycles of the CPU clock for CPU intermittent operation function.</li> <li>• Stops the CPU clock supply for the specified number of cycles each time the instruction is executed once.</li> <li>• Selectable from four types of clocks counts.</li> <li>• The bit is initialized to "00<sub>B</sub>" at a reset.</li> </ul>
bit 0	Reserved: Reserved bit	Be sure to set this bit to "0".

## ■ Access to Low-power Consumption Mode Control Register

Transition to the low-power consumption modes (stop mode, sleep mode, time-base timer mode, and watch mode) by writing to the low-power consumption mode control register is made, in this case, be sure to use the instructions in Table 6.3-2.

The low-power consumption mode transition instruction in Table 6.3-2 must always be followed by an array of instructions highlighted by a line below.

MOV LPMCR, #H'XX ; the low-power consumption mode transition instruction in Table 6.3-2

NOP
NOP
JMP \$+3 ; jump to next instruction

MOV A, #H'10 ; any instruction

The device does not guarantee its operation after returning from the low-power consumption mode if you place an array of instructions other than the one enclosed in the line. To access the low-power consumption mode control register (LPMCR) with C language, refer to "■ Notes on Accessing the Low-Power Consumption Mode Control Register (LPMCR) to Enter the Standby Mode" in the Section "6.8 Precautions when Using Low-power Consumption Mode".

To write word length data into the low-power consumption mode control register (LPMCR), use even addresses. Transition to the low-power consumption mode is made by writing at odd addresses may cause an erroneous operation.

When you control other functions than the functions listed in Figure 6.3-1, you may use any instruction.

**Table 6.3-2 List of Instructions Used for Transition to Low-power Consumption Mode**

MOV io, #imm8	MOV dir, #imm8	MOV eam, #imm8	MOV eam, Ri
MOV io, A	MOV dir, A	MOV addr16, A	MOV eam, A
MOV @RLi+disp8, A			
MOVW io, #imm16	MOVW dir, #imm16	MOVW eam, #imm16	MOVW eam, RWi
MOVW io, A	MOVW dir, A	MOVW addr16, A	MOVW eam, A
MOVW @RLi+disp8, A			
SETB io:bp	SETB dir:bp	SETB addr16:bp	
CLRB io:bp	CLRB dir:bp	CLRB addr16:bp	

## ■ Priority Level of STP, SLP, and TMD Bit

When the stop mode, sleep mode, and watch mode • time-base timer mode requests are performed at the same time, requests are processed according to the following order.

Stop mode request > watch mode • time-base timer mode request > sleep mode request

## 6.4 CPU Intermittent Operation Mode

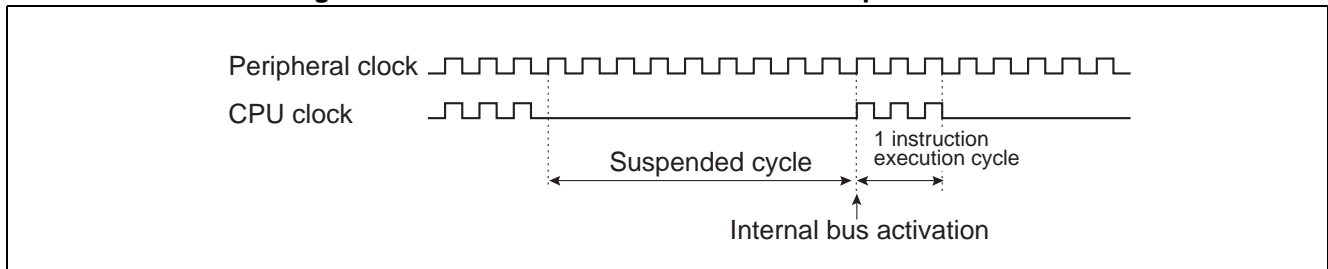
The CPU intermittent operation mode is a mode for reducing the power consumption by intermittently operating the CPU while operating the external bus and peripheral functions at high speed.

### ■ CPU Intermittent Operation Mode

The CPU intermittent operation mode is a mode for stopping the clock supplied to the CPU for a predetermined period of time for each instruction execution to delay the internal bus cycle start when accessing the registers, internal memory (ROM, RAM), I/O, peripheral functions, or external bus. If the CPU operation speed is decreased while supplying a high-speed peripheral clock to the peripheral functions, processing at low-power consumption becomes available.

- Selection of the number of the suspend cycles of the clock supplied to CPU is made in the CPU clock suspend cycle number selection bit (CG1 and CG0) of the low-power consumption mode control register (LPMCR).
- The external bus operation itself uses the same clock as the peripheral functions.
- The instruction execution time in CPU intermittent operation mode can be calculated by adding to the ordinary execution time the compensation value obtained by multiplying the instruction execution count for accessing the registers, internal memory, internal peripheral function, and external bus by the number of suspension cycles. Figure 6.4-1 shows the operation clock for CPU intermittent operation mode.

**Figure 6.4-1 Clock for the CPU Intermittent Operation Mode**



**MB90330A Series****6.5 Standby Mode**

**Standby modes are: sleep (PLL sleep, main sleep, sub sleep), watch, time-base timer, and stop mode.**

**■ Operation Status in Standby Mode**

Table 6.5-1 shows the operation state in the standby mode.

**Table 6.5-1 Operation Status in Standby Mode**

Standby Mode		Transition Condition	Main Clock	Sub Clock	Machine Clock	CPU	Peripheral	Pin	Release Method
Sleep mode	PLL sleep mode	SCS=1 MCS=0 SLP=1	Operation	Operation	Operation	Stops	Operation	Operation	Reset or interrupt
	Main sleep mode	SCS=1 MCS=1 SLP=1							
	Sub sleep mode	SCS=0 SLP=1	Stops						
Time-base timer mode	Time-base timer mode (SPL=0)	SCS=1 TMD=0	Operation	Stops	Stops	Stops *1	Hold		
	Time-base timer mode (SPL=1)	SCS=1 TMD=0					Hi-Z		
Watch mode	Watch mode (SPL=0)	SCS=0 TMD=0	Stops			Stops	Stops *2	Hold	
	Watch mode (SPL=1)	SCS=0 TMD=0						Hi-Z	
Stop mode	Stop mode (SPL=0)	STP=1		Stops	Stops	Stops	Hold		
	Stop mode (SPL=1)	STP=1	Hi-Z						

\*1: The time-base timer and the clock timer work.

\*2: The watch timer operates.

SPL: Pin state specification bit of low-power consumption mode control register (LPMCR)

SLP: Sleep mode bit of low-power consumption mode control register (LPMCR)

STP: Stop mode bit of low-power consumption mode control register (LPMCR)

TMD: Watch and time-base timer mode bit of low-power consumption mode control register (LPMCR)

MCS: Machine clock selection bit of clock selection register (CKSCR)

SCS: Machine clock selection bit (sub) of clock selection register (CKSCR)

Hi-Z: High impedance

RST: External reset pin

## 6.5.1 Sleep Mode

---

**Sleep mode is mode for stopping the CPU operation clock and the operation other than CPU continues. When you instruct the transition to the sleep mode with the low-power consumption mode control register (LPMCR), the transition to the PLL sleep mode is made if the PLL clock mode is set, the transition to the main sleep mode is made if the main clock mode is set, and the transition to the sub sleep mode is made if the sub clock mode is set.**

---

### ■ Transition to Sleep Mode

If you write "1" to the sleep mode bit (SLP), "1" to the watch/time-base timer mode bit (TMD), and "0" to the stop mode bit (STP) of the low-power consumption mode control register (LPMCR), the transition to the sleep mode is made. In this case, the transition to the PLL sleep mode is made if the clock selection register (CKSCR) is in the state of PLL clock selection bit (MCS)=0 and sub clock selection bit (SCS)=1, transition to the main sleep mode is made if MCS=1 and SCS=1, and transition to the sub sleep mode is made if SCS=0.

---

#### Note:

If you write "1" to the SLP bit and the STP bit of the LPMCR register at the same time, the STP bit is prioritized and the transition to the stop mode is made.

If you write "1" to the SLP bit and "0" to the TMD bit in the low-power consumption mode control register, the TMD bit is prioritized and the transition to the time-base timer mode or the watch mode is made.

---

#### ● Data retention function

In the sleep mode, the contents of the dedicated registers such as accumulators and the internal RAM are held unchanged.

#### ● Holding function

In the sleep mode, the external bus hold function operates and the state becomes the hold status upon a hold request.

#### ● Operation during an interrupt request

Writing "1" to the SLP bit in the LPMCR register does not make the transition to the sleep mode if there is an interrupt request. Therefore, the CPU executes the next instruction when it is in a state accepting no interrupts and branches immediately to the interrupt processing routine when it is in a state accepting interrupts.

#### ● Pin state

In the sleep mode, the preceding state is retained except for the pins used as the bus input/output or the bus control.

#### ■ Cancellation of Sleep Modes

The low-power consumption control circuit clears sleep mode by reset input or interrupt generation.

##### ● Return by reset

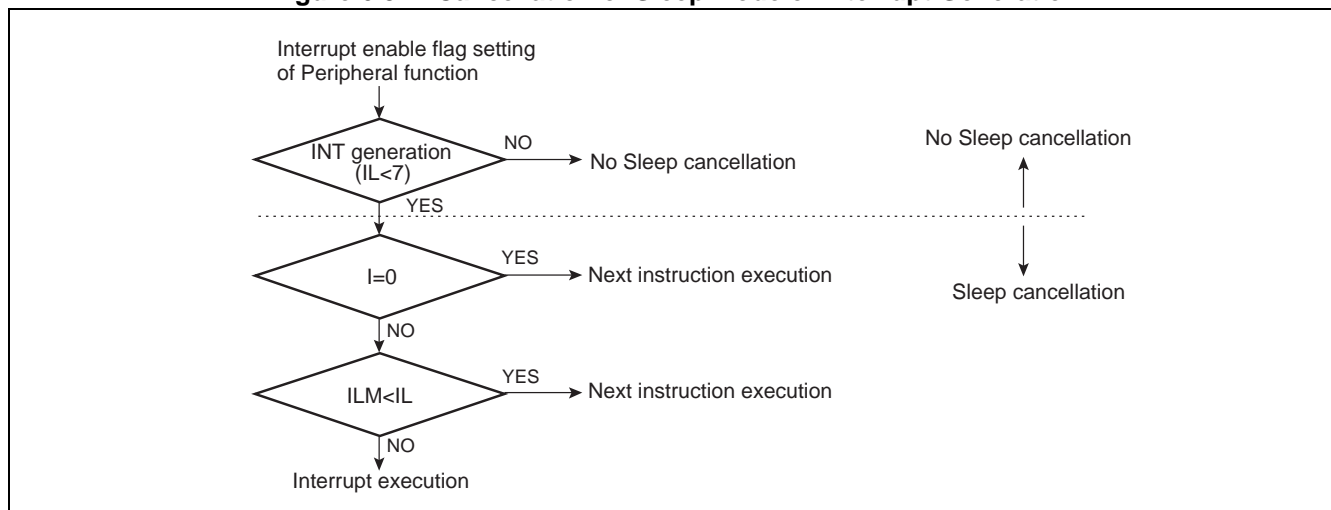
Initialization to the main clock mode is made by reset.

##### ● Return by interrupt

If there is an interrupt request higher than level 7 from the peripheral circuit and others in the sleep mode, the sleep mode is canceled. After clearing sleep mode, the action is the same as for ordinary interrupt processing. When an interrupt is acceptable by settings in the I flag of the condition code register (CCR), the interrupt level mask register (ILM), and the interrupt control register (ICR), the CPU performs interrupt processing. When an interrupt is not acceptable, processing from the instruction succeeding the one which has specified sleep mode continues.

Figure 6.5-1 shows clearing sleep mode by interrupt generation.

**Figure 6.5-1 Cancellation of Sleep Mode of Interrupt Generation**



#### Note:

When handling an interrupt, the CPU usually services the interrupt after executing the instruction that follows the one specifying the sleep mode. However, if the transition to the sleep mode and the receipt of the external bus hold request occur at the same time, the transition to the interruption procedure before executing the next instruction will be made.

## 6.5.2 Time-base Timer Mode

---

**The time-base timer mode terminates the original oscillation and all the operations other than the time-base timer and the watch timer, resulting in termination of all the functions other than the time-base timer and the watch timer.**

---

### ■ Transition to Time-base Timer Mode

In the PLL clock mode or the main clock mode (the sub clock display bit (SCM)=1 of the clock selection register (CKSCR)), writing "0" to the watch/time-base timer mode bit (TMD) of the low-power consumption mode control register (LPMCR) makes the transition to the time-base timer mode.

#### ● Data retention function

In the time-base timer mode, the contents of dedicated registers such as accumulators and the internal RAM are held unchanged.

#### ● Holding function

In the time-base timer mode, the external bus hold function is terminated, thereby neglecting a hold request even a request is input. If a hold request is input during the transition to the time-base timer mode, the condition of bus high-impedance is maintained and the HAK signal may not become "L".

#### ● Operation during an interrupt request

Writing "0" to the TMD bit of the low-power consumption mode control register (LPMCR) does not make the transition to the time-base timer mode if there is an interrupt request.

#### ● Pin state

You can set whether the external pin in the time-base timer mode should be retained in the preceding state or becomes to the high impedance state by controlling the pin state specification bit (SPL) in the LPMCR register.

## ■ Cancellation of Time-base Timer Modes

The low-power consumption circuit cancels the time-base timer mode by generating a reset input or an interrupt request.

- Return by external reset

The external reset initializes the mode to the main clock mode.

- Return by interrupt

If there is an interrupt request higher than level 7 from peripheral circuit and others in the time-base timer mode (except for IL2, IL1, IL0 of the interrupt control register (ICR) = 111<sub>B</sub>), the low-power consumption control circuit cancels the time-base timer mode. After cancellation of time-base timer modes, the action is the same as for ordinary interrupt processing. When an interrupt is acceptable according to the setting of the I flag of the condition code register (CCR), the interrupt level mask register (ILM), and the interrupt control register (ICR), interrupt processing is performed. When an interrupt is not acceptable, processing from the instruction succeeding the one before entering time-base timer mode continues.

---

### Note:

When handling an interrupt, the CPU usually services the interrupt after executing the instruction that follows the one specifying the time-base timer mode. When transition to time-base timer mode and acceptance of an external bus hold request have occurred at the same time, interrupt processing may transit before executing the next instruction.

---



### 6.5.3 Watch Mode

---

**The watch mode terminates all the operations other than the sub clock and the watch timer, where almost all the functions of the chip are terminated.**

---

#### ■ Transition to Watch Mode

In the sub clock mode (the sub clock display bit of the clock selection register (CKSCR) (SCS)=0), when you write "0" into the watch/time-base timer mode bit (TMD) of the low-power consumption mode control register (LPMCR the transition to the watch mode).

##### ● Data retention function

In the watch mode, contents of the dedicated registers such as accumulators and the internal RAM are held unchanged.

##### ● Holding function

During the watch mode, the external bus hold function halts, thereby neglecting a hold request if a hold request is input. If a hold request is input during the transition to the watch mode, the condition of bus high-impedance is maintained and the HAK signal may not become "L".

##### ● Operation during an interrupt request

When "0" is set to the TMD bit of the LPMCR register, transition to the watch mode is not made if there is an interrupt request.

##### ● Pin state setting

You can set whether the external pin in the watch mode should be retained in the preceding state or become to the high impedance state by controlling the pin state specification bit (SPL) in the LPMCR register.

#### ■ Cancellation of Watch Mode

The low-power consumption circuit cancels the watch mode by generating a reset input or an interrupt request.

##### ● Return by Reset

When clearing watch mode by a reset cause, transition occurs to oscillation stabilization wait reset state after clearing watch mode. The reset sequence is executed after the oscillation stabilization wait time.

##### ● Return by interrupt

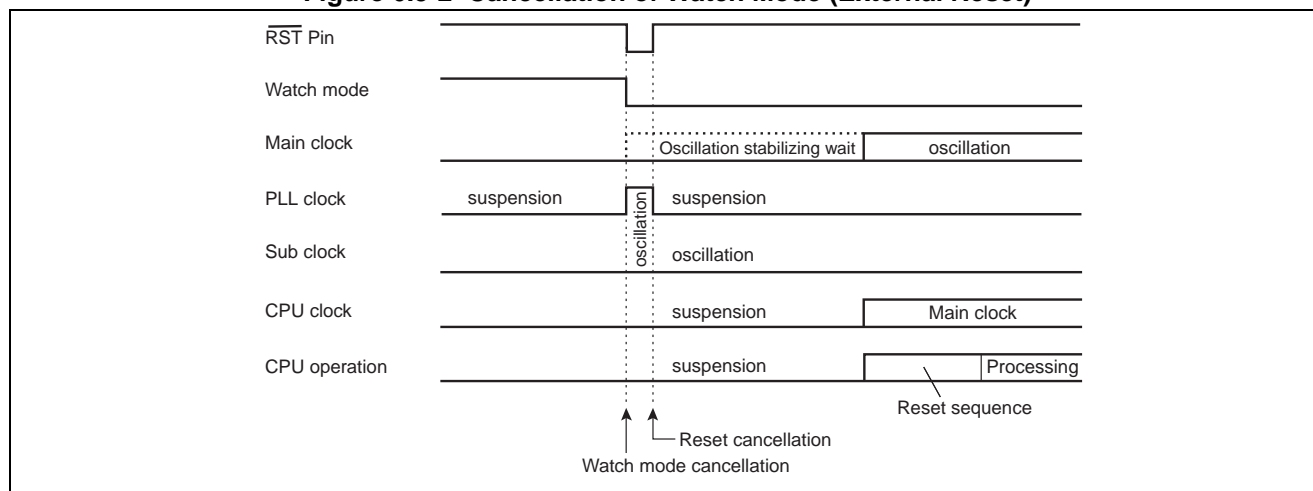
If there is an interrupt request higher than level 7 from peripheral circuits and others in the watch mode (except for IL2, IL1, IL0 of the interrupt control register (ICR) = 111<sub>B</sub>), the low-power consumption control circuit cancels the watch mode and immediately changes to the sub clock mode. After transiting to sub clock mode, the action is the same as for ordinary interrupt processing. When an interrupt is acceptable by settings in the I flag of the condition code register (CCR), the interrupt level mask register (ILM), and the interrupt control register (ICR), interrupt processing is carried out. When an interrupt is not acceptable, processing from the instruction succeeding the one caused to enter clock mode continues.

## Note:

When handling an interrupt, the CPU usually services the interrupt after executing the instruction that follows the one specifying the watch mode. However, if the transition to the watch mode and the receipt of the external bus hold request occur at the same time, transition to interruption procedure may be made before execution of the next instruction.

Figure 6.5-2 shows the cancellation operation of the watch mode.

**Figure 6.5-2 Cancellation of Watch Mode (External Reset)**



## 6.5.4 Stop Mode

---

**Stop mode is mode for stopping original oscillation and all functions are stopped. That means, data can be held with the lowest power consumption.**

---

### ■ Transition to Stop Mode

If you write "1" into the stop mode bit (STP) of the low-power consumption mode control register (LPMCR), the transition to the stop mode is made.

#### ● Data retention function

In the stop mode, the contents of the dedicated registers such as accumulators and the internal RAM are held unchanged.

#### ● Holding function

In the stop mode, the external bus hold function is terminated, thereby neglecting a hold request even a request is input. If you input a hold request while transiting to the stop mode, the HAK signal may not get to "L" with the bus being high impedance.

#### ● Operation during an interrupt request

When you write "1" into the SLP bit of the LPMCR register, the transition to the stop mode is not made if there is an interrupt request.

#### ● Pin state setting

You may determine by the pin state specification bit (SPL) of the LPMCR register whether the external pin in the stop mode should be retained in the preceding state or be shifted into the high impedance state.

### ■ Cancellation of Stop Modes

The low-power consumption control circuit clears stop mode when reset input or interrupt occurs. As oscillation of the operation clock is stopped when returning from stop mode, the low-power consumption control circuit first transits to the oscillation stabilization wait state and then clears stop mode.

#### ● Return by Reset

When clearing stop mode by a reset cause, transition occurs to oscillation stabilization wait reset state after clearing stop mode. The reset sequence is executed after the oscillation stabilization wait time.

#### ● Return by interrupt

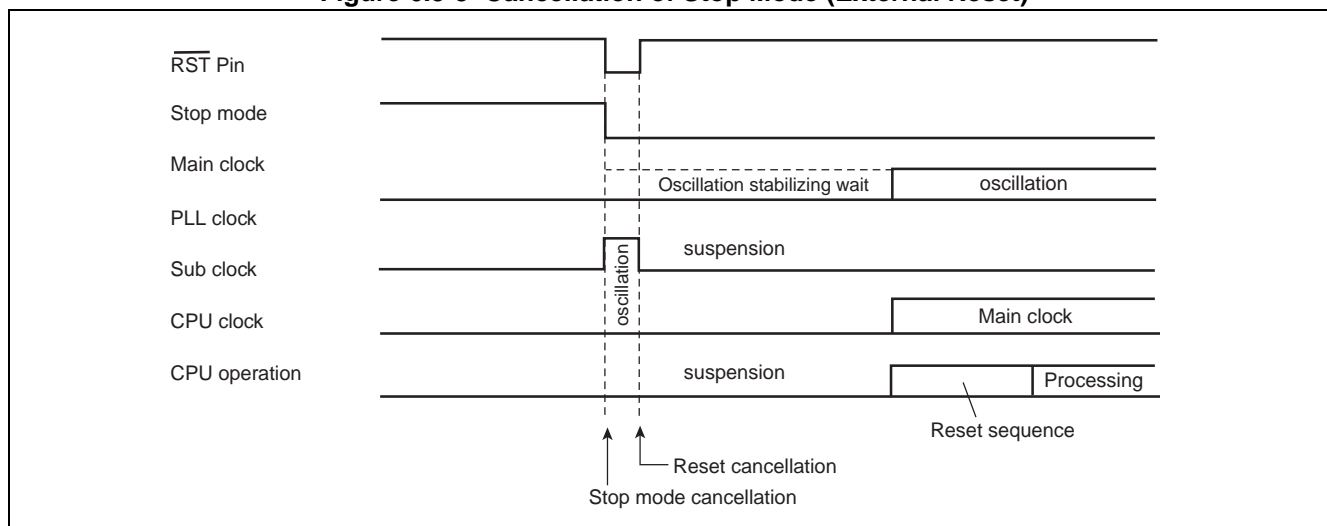
If there is an interrupt request higher than level 7 from the peripheral circuit and others in the stop mode (except for IL2, IL1, IL0 of the interrupt control register (ICR) = 111<sub>B</sub>), the low-power consumption control circuit cancels the stop mode. After canceling the stop mode, a normal interrupt procedure is executed when elapsing the main clock oscillation stable wait time specified by the oscillation stable wait time selection bit (WS1, WS0) of the clock selection register (CKSCR). When an interrupt is acceptable by settings in the I flag of the condition code register (CCR), the interrupt level mask register (ILM), and the interrupt control register (ICR), interrupt processing is carried out. When an interrupt is not acceptable, processing from the instruction succeeding the one caused to enter stop mode continues.

## Notes:

- When handling an interrupt, the CPU usually services the interrupt after executing the instruction that follows the one specifying the stop mode. When transition to stop mode and acceptance of an external bus hold request have occurred at the same time, interrupt processing may transit before executing the next instruction.
- In PLL stop mode, the main clock and PLL multiplier circuit remain stopped. When the CPU returns from PLL stop mode, therefore, it is necessary to allow for the main clock oscillation stabilization wait time and PLL clock oscillation stabilization wait time. In this case, the oscillation stabilization wait time concurrently counts the main clock oscillation stabilization wait time and the PLL clock oscillation stabilization wait time, according to the value specified in the oscillation stabilization wait time selection bit (CKSCR; WS1, WS0) of the clock selection register, thus the CKSCR; WS1, WS0 bit must be set in accordance with the longer oscillation stabilization wait time. However, as the PLL clock stabilization wait time requires at least  $2^{14}/\text{HCLK}$ , be sure to set the CKSCR; WS1, WS0 bit to "10<sub>B</sub>" or "11<sub>B</sub>".

Figure 6.5-3 shows the cancellation operation of the stop mode.

**Figure 6.5-3 Cancellation of Stop Mode (External Reset)**



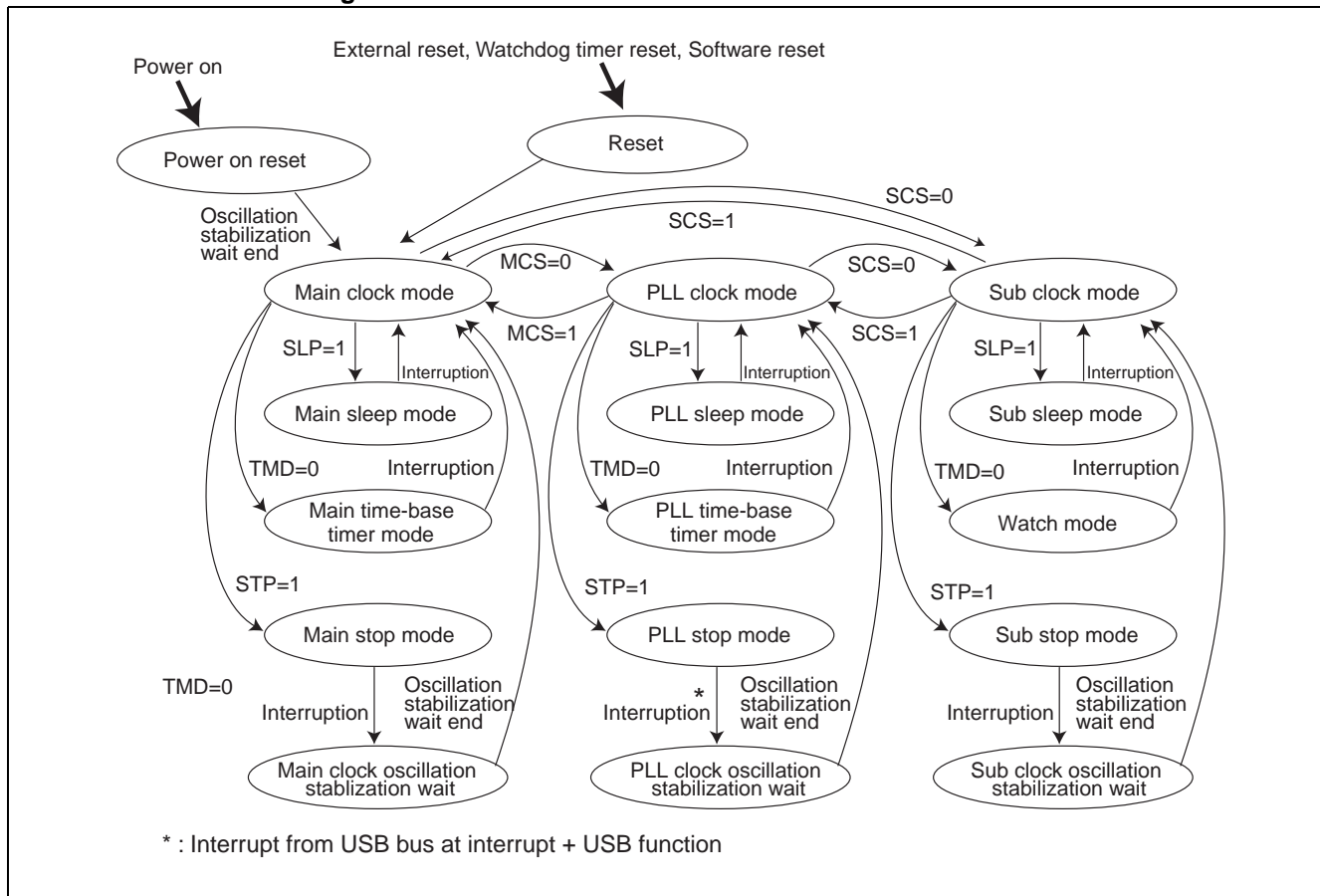
## 6.6 State Transition Diagram

The transition of the operation state and the transition conditions of MB90330A series are shown.

### ■ State Transition Diagram

Figure 6.6-1 shows the transition of the operation state and the transition conditions of MB90330A series.

Figure 6.6-1 State Transition and Transition Conditions



#### ■ Operation Status in Low-power Consumption Mode

Table 6.6-1 lists the operation states in low-power consumption mode.

**Table 6.6-1 Operation States in Low-power Consumption Mode**

Operating State	Main Clock	Sub Clock	PLL Clock	CPU	Peripheral	Clock	Time-base Timers	Clock Source
PLL	Operation	Operation	Operation	Operation	Operation	Operation	Operation	PLL Clock
PLL sleep				Stops				
PLL time-base timer								
PLL stop					Stops			
PLL oscillation stabilization waiting	Operation	Operation	Operation	Stops	Operation	Operation		
Main	Operation	Operation	Stops		Operation	Operation	Operation	Main Clock
Main sleep					Stops			
Main time-base timer								
Main stop	Stops	Stops		Stops		Stops		
Main oscillation stabilization waiting	Operation	Operation		Stops	Stops	Operation	Operation	
Sub	Stops	Operation	Operation			Operation	Stops	Sub Clock
Sub sleep			Stops					
Clock								
Sub-stop		Stops			Stops			
Sub oscillation stabilization waiting		Operation	Operation					
Power on reset	Operation	Operation	Stops	Stops	Stops	Operation	Operation	Main Clock
Reset			Operation					

## 6.7 State of the Pin during Standby Mode, Hold, and Reset

The state of the pin at the time of the stand by mode, the hold, or the reset is shown for each memory access code.

### ■ Pin State in Single-chip Mode

Table 6.7-1 shows the state of the pin in the single-chip mode.

**Table 6.7-1 Pin State in Single-chip Mode**

Pin Name	At sleep	In stop mode		At holding	At a reset
		SPL=0	SPL=1		
P07 to P00	The state immediately before hold*2	Input cutoff/ The state immediately before hold *2,*3	Input cutoff/ output Hi-Z *3	This state does not exist.	Impossible to input/ output Hi-Z
P17 to P10					
P27 to P20					
P37 to P30					
P47 to P40					
P57 to P50					
P77 to P70					
P87 to P80					
P96 to P90					
PA7 to PA0					
PB6 to PB0					
P67 to P60	Possible to input*1	Possible to input*1			Impossible to input
DVP	USB port input	USB port input *4		*5	Hi-Z
DVM					
HVP					
HVM					
UTEST	Pull-down connect	Pull-down connect			
HCON	The state immediately before hold	Hold state preserved			"H" output

<sup>\*1</sup>: Same as the other ports if this is being used for the output state. "Input enabled" means that the input function is currently enabled; Pull-up/Pull-down or an input from the external is required.

<sup>\*2</sup>: Outputs the initial output preceding this mode. Alternatively, this means "Input disabled" if the signal is input. "Input disabled" means that the contents of a pin are not accepted internally, because the internal circuit is not in operation although operation of the input gate nearest to the pin is currently enabled.

- \*3: In the input shutoff state, the input is masked and the "L" level is passed to the internal.
- \*4: When operation stops due to a cause in the USB, the signal is input via the USB port. When it stops due to another cause, the preceding state remains unchanged.
- \*5: There is no influence in the pin USB operation.

---

Note:

The mode transits to the standby mode is also inhibited during the USB is transferred.

---



### ■ Pin State in External Bus 16-bit Data Bus and Multiplex 16-bit External Bus Mode

Table 6.7-2 shows the state of the pins of the external bus 16-bit data bus mode and the multiplex 16-bit external bus mode.

**Table 6.7-2 Pin State in External Bus 16-bit Data Bus and Multiplex 16-bit External Bus Mode**

Pin Name	At sleep	In stop mode		At holding	At a reset
		SPL=0	SPL=1		
P07 to P00 (AD07 to AD00)	Impossible to input/ output Hi-Z	Input cutoff/ output Hi-Z	Input cutoff/ output Hi-Z <sup>*6</sup>	Impossible to input/ output Hi-Z	Impossible to input/ output Hi-Z
P17 to P10 (AD15 to AD08)					
P27 to P20 (A23 to A16)	Output state <sup>*1,*4</sup>	Output state <sup>*1,*4</sup>		Impossible to input/ output Hi-Z <sup>*4</sup>	Output state <sup>*1</sup>
P57(CLK)	Impossible to input/ possible to output <sup>*2,*4</sup>	Impossible to input/ output state <sup>*1,*4</sup>		Impossible to input/ possible to output <sup>*2,*4</sup>	Impossible to input/ possible to output <sup>*2</sup>
P56(RDY)	The state immediately before hold <sup>*5</sup>	Input cutoff/ The state immediately before hold <sup>*5,*6</sup>		Impossible to input <sup>*4</sup>	Impossible to input/ output Hi-Z
P55( $\overline{\text{HAK}}$ )				"L" output	
P54(HRQ)				"1" input	
P53( $\overline{\text{WRH}}$ )	"H" output <sup>*4</sup>	"H" output <sup>*4</sup>		Impossible to input/ output Hi-Z <sup>*4</sup>	"H" output
P52( $\overline{\text{WRL}}$ )					
P51( $\overline{\text{RD}}$ )	"H" output	"H" output		Impossible to input/ output Hi-Z	Possible to output <sup>*2</sup>
P50(ALE)	"L" output	"L" output			
P37 to P30	The state immediately before hold <sup>*5</sup>	Input cutoff/ The state immediately before hold <sup>*5</sup>		Hold state preserved <sup>*5</sup>	Impossible to input/ output Hi-Z
P47 to P40					
P77 to P70					
P87 to P80					
P96 to P90					
PA7 to PA0					
PB6 to PB0					
P67 to P60	Possible to input <sup>*3</sup>	Possible to input <sup>*3</sup>	Possible to input <sup>*3</sup>	Impossible to input	
DVP	USB port input	USB port input <sup>*7</sup>	<sup>*8</sup>	Hi-Z	
DVM					
HVP					
HVM					
UTEST	Pull-down connect	Pull-down connect		Pull-down connect	
$\overline{\text{HCON}}$	Hold state preserved	Hold state preserved		"H" output	

\*1: "Output status" means that although the pin driving transistor is left drive enabled, fixed value "H" or "L" is output, because operation of the internal circuit remains in stopped state. If the internal peripheral circuit is in operation and the output function is in use, the output varies except when a reset is generated. There is no output change in reset.

- \*2: "Output enabled" means that the contents of operation appear via the pin because the pin driving transistor is in driven state and operation of the internal circuit remains enabled.
- \*3: Same as the other ports if this is being used for the output state. "Input enabled" means that the input function is currently enabled; Pull-up/Pull-down or an input from the external is required.
- \*4: If this is being used as an output port, it holds the preceding value.
- \*5: Outputs the initial output preceding this mode. Alternatively, this means "input disabled" if the signal is input. "Input disabled" means that the contents of a pin are not accepted internally, because the internal circuit is not in operation although operation of the input gate nearest to the pin is currently enabled.
- \*6: In the input shutoff state, the input is masked and the "L" level is passed to the internal.
- \*7: When operation stops due to a cause in the USB, the signal is input via the USB port. When it stops due to another cause, the preceding state remains unchanged.
- \*8: There is no influence in the pin USB operation.

---

**Note:**

The mode transits to the standby mode is also inhibited during the USB is transferred.

---

■ **Pin State in External Bus 8-bit Data Bus and Multiplex 8-bit External Bus Mode**

Table 6.7-3 shows the state of the pins of the external bus 8-bit data bus mode and the multiplex 8-bit external bus mode.

**Table 6.7-3 Pin Signal State in External Bus 8-bit Data Bus and Multiplex 8-bit External Bus Mode**

Pin Name	At sleep	In stop mode		At holding	At a reset
		SPL=0	SPL=1		
P07 to P00 (AD07 to AD00)	Impossible to input/ output Hi-Z	Input cutoff/ output Hi-Z	Input cutoff/ output Hi-Z *6	Impossible to input/ output Hi-Z	Impossible to input/ output Hi-Z
P17 to P10 (A15 to A08)	Output state *1	Output state *1		Impossible to input/ output Hi-Z	Output state *1
P27 to P20 (A23 to A16)	Output state *1,*4	Output state *1,*4			
P57(CLK)	Impossible to input/ possible to output *2,*4	Impossible to input/ output state *1,*4		Impossible to input/ possible to output *2,*4	Impossible to input/ possible to output *2
P56(RDY)	Hold state preserved*5	Input cutoff/ The state immediately before hold*5		Impossible to input*4	Impossible to input/ output Hi-Z
P55(HAK)				"L" output	
P54(HRQ)				"1" input	
P53(WRH)				The state immediately before hold*5	
P52(WRL)	"H" output *5	"H" output *5		Impossible to input/ output Hi-Z *4	"H" output
P51(RD)	"H" output	"H" output		Impossible to input/ output Hi-Z	"L" output
P50(ALE)	"L" output	"L" output			
P37 to P30	Hold state preserved*5	Input cutoff/ The state immediately before hold*5		The state immediately before hold*5	Impossible to input/ output Hi-Z
P47 to P40					
P77 to P70					
P87 to P80					
P96 to P90					
PA7 to PA0					
PB6 to PB0					
P67 to P60	Possible to input *3	Possible to input *3	Possible to input*3	Impossible to input	
DVP	USB port input	USB port input *7	*8	Hi-Z	
DVM					
HVP					
HVM					
UTEST	Pull-down connect	Pull-down connect		Pull-down connect	
HCON	The state immediately before hold	Hold state preserved		"H" output	

\*1: "Output status" means that although the pin driving transistor is left drive enabled, fixed value "H" or "L" is output, because operation of the internal circuit remains in stopped state. If the internal peripheral circuit is in operation and the output function is in use, the output varies except when a reset is generated. There is no output change in reset.

- \*2: "Output enabled" means that the contents of operation appear via the pin because the pin driving transistor is in driven state and operation of the internal circuit remains enabled.
- \*3: Same as the other ports if this is being used for the output state. "Input enabled" means that the input function is currently enabled; Pull-up/Pull-down or an input from the external is required.
- \*4: If this is being used as an output port, it holds the preceding value.
- \*5: Outputs the initial output preceding this mode. Alternatively, this means "Input disabled" if the signal is input. "Input disabled" means that the contents of a pin are not accepted internally, because the internal circuit is not in operation although operation of the input gate nearest to the pin is currently enabled.
- \*6: In the input shutoff state, the input is masked and the "L" level is passed to the internal.
- \*7: When operation stops due to a cause in the USB, the signal is input via the USB port. When it stops due to another cause, the preceding state remains unchanged.
- \*8: There is no influence in the pin USB operation.

---

**Note:**

The mode transits to the standby mode is also inhibited during the USB is transferred.

---

■ **Pin State in External Bus 16-bit Data Bus and Non-multiplex 16-bit External Bus Mode**

Table 6.7-4 shows the state of the pins of the external bus 16-bit data bus mode and the non-multiplex 16-bit external bus mode.

**Table 6.7-4 Pin State in External Bus 16-bit Data Bus and Non-multiplex 16-bit External Bus Mode**

Pin Name	At sleep	In stop mode		At holding	At a reset
		SPL=0	SPL=1		
P07 to P00 (D07 to D00)	Possible to input/ output Hi-Z	Input cutoff/ output Hi-Z	Input cutoff/ output Hi-Z <sup>*6</sup>	Impossible to input/ output Hi-Z	Impossible to input/ output Hi-Z
P17 to P10 (D15 to D08)					Output state <sup>*1</sup>
P37 to P30 (A07 to A00)	Output state <sup>*1</sup>	Output state <sup>*1</sup>			
P47 to P40 (A15 to A08)					
P27 to P20 (A23 to A16)	Output state <sup>*1,*4</sup>	Output state <sup>*1,*4</sup>		Impossible to input/ output Hi-Z <sup>*4</sup>	Impossible to input/ possible to output <sup>*2</sup>
P57(CLK)	Impossible to input/ possible to output <sup>*2,*4</sup>	Impossible to input/ possible to output <sup>*1,*4</sup>		Impossible to input/ possible to output <sup>*2,*4</sup>	
P56(RDY)	The state immediately before hold <sup>*5</sup>	Input cutoff/ The state immediately before hold <sup>*5</sup>		Impossible to input <sup>*4</sup>	Impossible to input/ output Hi-Z
P55(HAK)				"L" output	
P54(HRQ)				"1" input	
P53(WRH)	"H" output <sup>*5</sup>	"H" output <sup>*5</sup>		Impossible to input/ output Hi-Z <sup>*4</sup>	"H" output
P52(WRL)				Impossible to input/ output Hi-Z	
P51(RD)	"H" output	"H" output			
P50(ALE)	"L" output	"L" output			
P77 to P70	The state immediately before hold <sup>*5</sup>	Input cutoff/ The state immediately before hold <sup>*5</sup>		The state immediately before hold <sup>*5</sup>	Impossible to input/ output Hi-Z
P87 to P80					
P96 to P90					
PA7 to PA0					
PB6 to PB0					
P67 to P60	Possible to input <sup>*3</sup>	Possible to input <sup>*3</sup>	Possible to input <sup>*3</sup>	Impossible to input	
DVP	USB port input	USB port input <sup>*7</sup>	<sup>*8</sup>	Hi-Z	
DVM					
HVP					
HVM					
UTEST	Pull-down connect	Pull-down connect			Pull-down connect
HCON	The state immediately before hold	Hold state preserved		"H" output	

\*1: "Output status" means that although the pin driving transistor is left drive enabled, fixed value "H" or "L" is output, because operation of the internal circuit remains in stopped state. If the internal peripheral circuit is in operation and the output function is in use, the output varies except when a reset is generated. There is no output change in reset.

\*2: "Output enabled" means that the contents of operation appear via the pin because the pin driving transistor is in driven state and operation of the internal circuit remains enabled.

- \*3: Same as the other ports if this is being used for the output state. "Input enabled" means that the input function is currently enabled; Pull-up/Pull-down or an input from the external is required.
- \*4: If this is being used as an output port, it holds the preceding value.
- \*5: Outputs the initial output preceding this mode. Alternatively, this means "Input disabled" if the signal is input. "Input disabled" means that the contents of a pin are not accepted internally, because the internal circuit is not in operation although operation of the input gate nearest to the pin is currently enabled.
- \*6: In the input shutoff state, the input is masked and the "L" level is passed to the internal.
- \*7: When operation stops due to a cause in the USB, the signal is input via the USB port. When it stops due to another cause, the preceding state remains unchanged.
- \*8: There is no influence in the pin USB operation.

---

**Note:**

The mode transits to the standby mode is also inhibited during the USB is transferred.

---

### ■ Pin State in External Bus 8-bit Data Bus and Non-multiplex 8-bit External Bus Mode

Table 6.7-5 shows the state of the pins of the external bus 8-bit data bus mode and the non-multiplex 8-bit external bus mode.

**Table 6.7-5 Pin State in External Bus 8-bit Data Bus and Non-multiplex 8-bit External Bus Mode**

Pin Name	At sleep	In stop mode		At holding	At a reset	
		SPL=0	SPL=1			
P07 to P00 (D07 to D00)	Impossible to input/ output Hi-Z	Input cutoff/ output Hi-Z	Input cutoff/ output Hi-Z <sup>*6</sup>	Impossible to input/ output Hi-Z	Impossible to input/ output Hi-Z	
P37 to P30 (A07 to A00)	Output state <sup>*1</sup>	Output state <sup>*1</sup>			Impossible to input/ output Hi-Z <sup>*4</sup>	Output state <sup>*1</sup>
P47 to P40 (A15 to A08)						
P27 to P20 (A23 to A16)	Output state <sup>*1,*4</sup>	Output state <sup>*1,*4</sup>		Impossible to input/ output Hi-Z <sup>*4</sup>	Impossible to input/ possible to output <sup>*2</sup>	
P57(CLK)	Impossible to input/ possible to output <sup>*2,*4</sup>	Impossible to input/ possible to output <sup>*1,*4</sup>		Impossible to input/ possible to output <sup>*2,*4</sup>		
P56(RDY)	The state immediately before hold <sup>*5</sup>	Input cutoff/ The state immediately before hold <sup>*5</sup>		Impossible to input/ <sup>*4</sup>	Impossible to input/ output Hi-Z	
P55( $\overline{\text{HAK}}$ )				"L" output		
P54(HRQ)				"1" input		
P53( $\overline{\text{WRH}}$ )				Hold state preserved <sup>*5</sup>	"H" output	
P52( $\overline{\text{WRL}}$ )	"H" output <sup>*4</sup>	"H" output <sup>*4</sup>		Impossible to input/ output Hi-Z <sup>*4</sup>		
P51( $\overline{\text{RD}}$ )	"H" output	"H" output		Impossible to input/ output Hi-Z		
P50(ALE)	"L" output	"L" output		Hold state preserved <sup>*5</sup>	Possible to output <sup>*2</sup>	
P17 to P10	The state immediately before hold <sup>*5</sup>	Input cutoff/ The state immediately before hold <sup>*5</sup>			Impossible to input/ output Hi-Z	Impossible to input/ output Hi-Z
P77 to P70						
P87 to P80						
P96 to P90						
PA7 to PA0						
PB6 to PB0						
P67 to P60	Possible to input <sup>*3</sup>	Possible to input <sup>*3</sup>	Possible to input <sup>*3</sup>	Impossible to input		
DVP	USB port input	USB port input <sup>*7</sup>	<sup>*8</sup>	Hi-Z		
DVM						
HVP						
HVM						
UTEST	Pull-down connect	Pull-down connect		Pull-down connect		
$\overline{\text{HCON}}$	The state immediately before hold	Hold state preserved	"H" output			

<sup>\*1</sup>: "Output status" means that although the pin driving transistor is left drive enabled, fixed value "H" or "L" is output, because operation of the internal circuit remains in stopped state. If the internal peripheral circuit is in operation and the output function is in use, the output varies except when a reset is generated. There is no output change in reset.

- \*2: "Output enabled" means that the contents of operation appear via the pin because the pin driving transistor is in driven state and operation of the internal circuit remains enabled.
- \*3: Same as the other ports if this is being used for the output state. "Input enabled" means that the input function is currently enabled; Pull-up/Pull-down or an input from the external is required.
- \*4: If this is being used as an output port, it holds the preceding value.
- \*5: Outputs the initial output preceding this mode. Alternatively, this means "input disabled" if the signal is input. "Input disabled" means that the contents of a pin are not accepted internally, because the internal circuit is not in operation although operation of the input gate nearest to the pin is currently enabled.
- \*6: In the input shutoff state, the input is masked and the "L" level is passed to the internal.
- \*7: When operation stops due to a cause in the USB, the signal is input via the USB port. When it stops due to another cause, the preceding state remains unchanged.
- \*8: There is no influence in the pin USB operation.

---

**Note:**

The mode transits to the standby mode is also inhibited during the USB is transferred.

---



## **6.8 Precautions when Using Low-power Consumption Mode**

---

**Special attention for the following is needed when using the low-power consumption mode.**

- **Transition to standby mode and interrupt**
  - **Cancellation of standby mode by interrupt**
  - **Oscillation stabilization wait time**
  - **Switching clock mode**
  - **Notes on Accessing the Low-Power Consumption Mode Control Register (LPMCR) to Enter the Standby Mode**
- 

### **■ Transition to Standby Mode and Interrupt**

When there is the generation of an interrupt request to the CPU from the peripheral functions, the transition to each standby mode is not made (the transition to the standby mode is not made even after the interrupt processing) because even if you set "1" to the stop mode bit (STP) and the sleep mode bit (SLP) of the low-power consumption mode control register (LPMCR), or "0" to the watch/time-base timer mode bit (TMD), it is ignored. In this case, if the interrupt level is higher than 7, whether or not the interrupt request is accepted by the CPU is not related to this operation.

Even when the CPU is currently performing interrupt processing, the device can go to the standby mode if no other interrupt request is present with the interrupt request flag bit cleared.

### **■ Cancellation of Standby Mode by Interrupt**

When an interrupt request with an interrupt level higher than 7 has been generated from a peripheral circuit, etc. in sleep, time-base timer, or stop mode, standby mode is cleared. Whether or not the interrupt request is accepted by the CPU is not related to this operation.

After the standby mode is canceled by the interruption, the operation is branched off to the interrupt processing routine, if the priority of the interruption level set bit (IL2, IL1, IL0 bits of ICR register) to the interrupt request is over the interrupt level mask register (ILM), and the interrupt is permitted by the I flag of the condition code register (CCR) (I=1). When an interrupt is not acceptable, processing from the instruction succeeding the one which has specified standby mode continues.

When performing interrupt processing, interrupt processing normally transits after executing the instruction succeeding the one specifying standby mode.

Depending on the conditions for transiting to standby mode, interrupt processing may start before executing the next instruction.

---

**Note:**

To prohibit a branch to the interrupt processing routine immediately after return, interrupts must be prohibited before standby mode is set.

---

## ■ Oscillation Stabilization Wait Time

### ● Oscillation Stabilization Wait Time of oscillation clock

Because the oscillator for original oscillation is stopped in stop mode, the oscillation stabilization wait time must be required. For the oscillation stabilization wait time, you take the time selected in the oscillation stabilization wait time selection bit (WA1, WA0) of the clock selection register (CKSCR).

---

#### Note:

Be sure to set "00<sub>B</sub>" for the oscillation stabilization wait time selection bit (WS1, WS0) of the CKSCR register ONLY at the time of the main clock.

---

### ● Oscillation stabilization wait time of PLL clock

When the transition is made from a state where the CPU is operated by the main clock and the PLL clock halts, to a mode where CPU and the peripherals are operated by the PLL clock, transition to the PLL clock oscillation stabilization wait state is made and it is operated by the main clock during the oscillation stabilization wait state.

The PLL clock oscillation stabilization wait time is fixed to  $2^{14}/\text{HCLK}$  (HCLK: oscillation clock).

In sub clock mode, the main clock and PLL multiplication circuit stop. When changing to PLL clock mode, it is necessary to reserve the main clock oscillation stabilization wait time and PLL clock oscillation stabilization wait time. The oscillation stabilization wait times for main clock and PLL clock are counted simultaneously according to the value specified in the oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register. The oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register must be selected accordingly to account for the longer of the main clock and PLL clock oscillation stabilization wait times. The PLL clock oscillation stabilization wait time, however, requires  $2^{14}/\text{HCLK}$  or more. Set the oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register to "10<sub>B</sub>" or "11<sub>B</sub>".

In PLL stop mode, the main clock and PLL multiplication circuit stop. During recovery from PLL stop mode, it is necessary to allot the main clock oscillation stabilization wait time and PLL clock oscillation stabilization wait time. The oscillation stabilization wait times for main clock and PLL clock are counted simultaneously according to the value specified in the oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register. The oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register must be selected accordingly to account for the longer of the main clock and PLL clock oscillation stabilization wait time. The PLL clock oscillation stabilization wait time, however, requires  $2^{14}/\text{HCLK}$  or more. Set the oscillation stabilization wait time selection bits (CKSCR: WS1, WS0) in the clock selection register to "10<sub>B</sub>" or "11<sub>B</sub>".

## ■ Switching Clock Mode

When switching of the clock mode is made, be sure not to change to other clock modes or the low-power consumption mode until the completion of the switching. Completion of the switching can be checked by referencing the MCM and SCM bits of the CKSCR. If the mode is switched to another clock mode or low-power-consumption mode before completion of switching, the mode may not be switched.

## ■ Notes on Accessing the Low-Power Consumption Mode Control Register (LPMCR) to Enter the Standby Mode

- To access the low-power consumption mode control register (LPMCR) with assembler language
  - To set the low-power consumption mode control register (LPMCR) to enter the standby mode, use the instruction listed in Table 6.3-2.
  - The low-power consumption mode transition instruction in Table 6.3-2 must always be followed by an array of instructions highlighted by a line below.

MOV LPMCR,#H'XX	; the low-power consumption mode transition instruction in Table 6.3-2
NOP	
NOP	
JMP \$+3	; jump to next instruction
MOV A,#H'10	; any instruction

The device does not guarantee its operation after returning from the low-power consumption mode if you place an array of instructions other than the one enclosed in the line.

- To access the low-power consumption mode (LPMCR) with C language

To enter the standby mode using the low-power consumption mode control register (LPMCR), use one of the following methods (1) to (3) to access the register:

- (1) Specify the standby mode transition instruction as a function and insert two `_wait_nop()` built-in functions after that instruction. If any interrupt other than the interrupt to return from the standby mode can occur within the function, optimize the function during compilation to suppress the LINK and UNLINK instructions from occurring.

Example: Watch mode or time-base timer mode transition function

```
Void enter_watch(){
    IO_LPMCR_byte = 0x10          /* Set LPMCR TMD bit to 0 */
    _wait_nop();
    _wait_nop();
}
```

- (2) Define the standby mode transition instruction using `_asm` statements and insert two NOP and JMP instructions after that instruction.

Example: Transition to sleep mode

```
_asm(" MOV I: _IO_LPMCR,#H'58"); /* Set LPMCR SLP bit to 1 */
_asm(" NOP");
_asm(" NOP");
_asm(" JMP $+3");                /* Jump to next instruction */
```

- (3) Define the standby mode transition instruction between `#pragma asm` and `#pragma endasm` and insert two NOP and JMP instructions after that instruction.

Example: Transition to stop mode

```
#pragma asm
MOV I: _IO_LPMCR,#H'98          /* Set LPMCR STP bit to 1 */
NOP
NOP
JMP $+3                          /* Jump to next instruction */
#pragma endasm
```

# ***CHAPTER 7***

---

# ***MODE SETTING***

**This chapter describes the mode setting and the external memory access.**

- 7.1 Mode Setting
- 7.2 Mode Pins (MD2 to MD0)
- 7.3 Mode Data
- 7.4 External Memory Access
- 7.5 Operation in Each Mode of Mode Setting

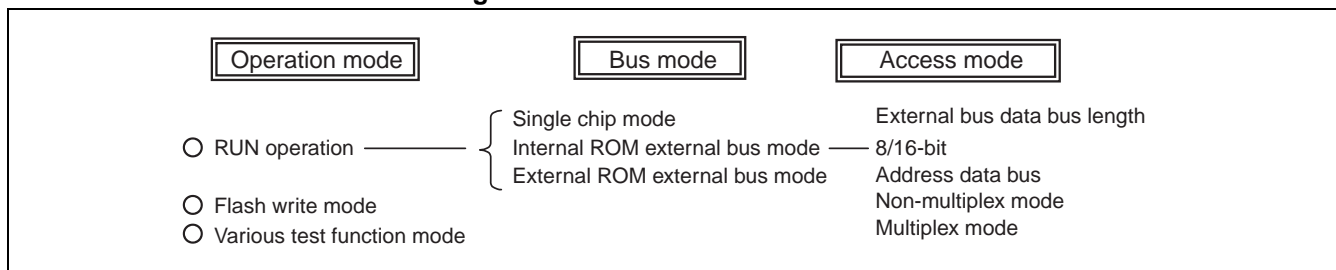
## 7.1 Mode Setting

**F<sup>2</sup>MC-16LX has respective modes in the access method, access area, and test. Respective mode is set according to the mode pin at the time of reset and the mode-fetched mode data.**

### ■ Mode Setting

F<sup>2</sup>MC-16LX has respective modes in the access method, the access area, and the test, and Figure 7.1-1 shows the classification.

**Figure 7.1-1 Classification of Mode**



### ■ Operating Mode

The operation mode controls the operation state of the device and is specified by the mode setting pin (MDx) and the contents of the Mx bit in the mode data. Selecting the operation mode, you may perform the normal operation/the start of the internal test program/the start of the special test function.

### ■ Bus Modes

The bus mode controls the internal ROM operation and the external access function operation, and is specified by the mode setting pin (MDx) and the contents of the Mx bit in the mode data. The mode setting pin (MDx) specifies the bus mode to read the reset vector and the mode data. The Mx bit in the mode data specifies the bus mode at the normal operation.

### ■ Access Mode

The access mode controls the external data bus width and is specified by the mode setting pin (MDx) and the Sx bit in the mode data. Selecting the access mode specifies either the 8-bit length or the 16-bit length for the external data bus. Also, this mode specifies either the non-multiplex mode or the multiplex mode for the address data bus.

## 7.2 Mode Pins (MD2 to MD0)

The mode pin is the three external pins (MD2 to MD0) and specifies the load methods for the reset vector and the mode data.

### ■ Setting of Mode Pins (MD2 to MD0)

With the mode pin (MD2 to MD0), you may select either the external data bus or the internal data bus for the reset vector read and select the bus width while selecting the external data bus. For the internal Flash ROM products, the mode pin also specifies the Flash ROM write mode to write the internal ROM program and so forth.

Table 7.2-1 shows the content of the mode pin setting.

**Table 7.2-1 Content of Setting of Mode Pins**

P61	P60	MD2	MD1	MD0	Mode Name	Reset vectors access area	External Data Bus width	Remark
-	-	0	0	0	External vector mode 0	External	Multiplex mode	Reset vector bus width access
-	-	0	0	1	External vector mode 1	External	Multiplex mode	Reset vector bus width access
-	-	0	1	0	External vector mode 2	External	Non-multiplex mode	Reset vector bus width access
-	-	0	1	1	External vector mode	Internal	Mode Data	Operation after reset sequence is controlled with mode data
-	-	1	0	0	Setting disabled			
-	-	1	0	1				
1	0	1	1	0	Flash serial writing			
-	-	1	1	1	Flash writer write mode	-	-	-

**Note:**

Set MD2 to MD0:0= $V_{SS}$  or 1= $V_{CC}$ . For the external vector mode 2, the data bus width also has the initial value of 8-bit. When setting 16-bit as the data bus width, specify the mode data for the non-multiplex external data bus 16-bit mode, and then the LMBS areas are set to the 16-bit size access. To set the HMBS area for the 16-bit size access change the EPCR:HMBS bit setting.

## 7.3 Mode Data

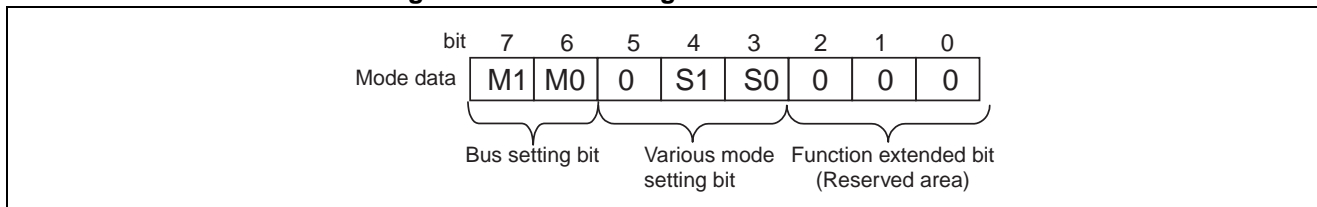
The mode data is located on the memory of the FFFFDF<sub>H</sub> address and specifies the operation after the reset sequence. The mode data is automatically taken into the CPU by a mode fetch.

### ■ Mode Data

While executing the reset sequence, the mode data in the FFFFDF<sub>H</sub> address is captured into the mode register in the CPU core. CPU sets the memory access mode by this mode data. The values of the mode register can be changed only in the reset sequence. The setting of the mode data is enabled after the reset sequence.

Figure 7.3-1 shows the configuration of the mode data.

**Figure 7.3-1 Bit Configuration of Mode Data**



### ■ Set Bit of Various Modes (S1, S0)

The S1 and S0 bits specify the bus mode and access mode after the reset sequence.

Table 7.3-1 shows the content of the S1, S0 bit setting.

**Table 7.3-1 Content of S1 and S0 Bit Setting**

S1	S0	Functions	
0	0	External data bus 8-bit mode	Address data bus multiplex
0	1	External data bus 16-bit mode	
1	0	External data bus 8-bit mode	Address data bus non-multiplex
1	1	External data bus 16-bit mode	

## ■ Set Bit of Bus Mode (M1, M0)

The M1 and M0 bits specify the operation mode after the reset sequence.

Table 7.3-2 shows the content of the M1, M0 bit setting.

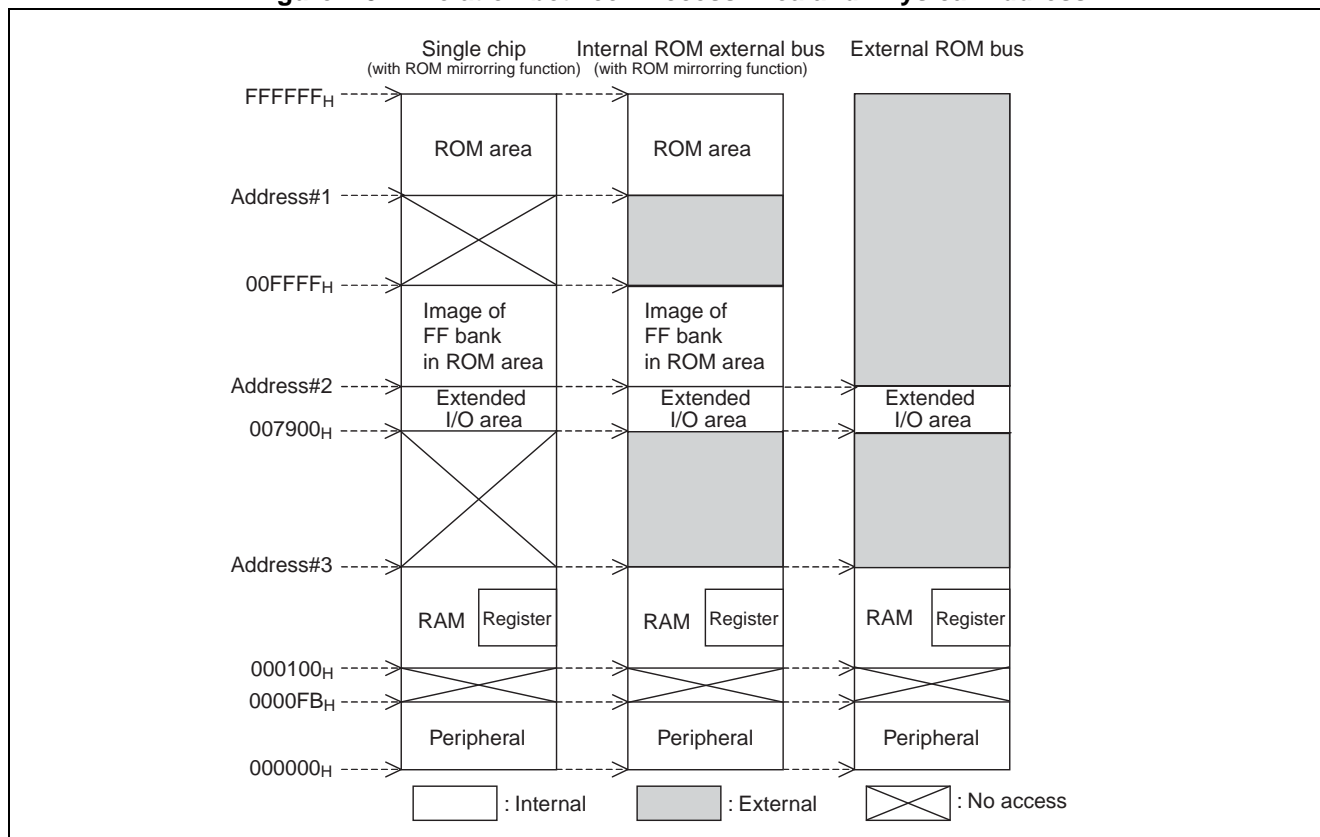
**Table 7.3-2 Content of M1 and M0 Bit Setting**

M1	M0	Function
0	0	Single-chip mode
0	1	Internal ROM external bus modes
1	0	External ROM external bus modes
1	1	(Setting prohibited)

## ■ Relation between Access Area and Physical Address

Figure 7.3-2 shows the relation between the access area and the physical address.

**Figure 7.3-2 Relation between Access Area and Physical Address**



Note:

The "address #X" is determined by each family. Please refer to "Appendix A memory map" for details.



■ **Relation between Mode Pin and Mode Data (Recommended Example)**

Table 7.3-3 shows the relation between the mode pin and the mode data.

**Table 7.3-3 Relation between Mode Pin and Mode Data**

Mode	MD2	MD1	MD0	M1	M0	S1	S0
Single Chip	0	1	1	0	0	X	X
Internal ROM external bus mode 8 bits (Address data multiplex)	0	1	1	0	1	0	0
Internal ROM external bus Mode 16 bits (Address data multiplex)	0	1	1	0	1	0	1
Internal ROM external bus Mode 8 bits (Address data non-multiplex)	0	1	1	0	1	1	0
Internal ROM external bus Mode 16 bits (Address data non-multiplex)	0	1	1	0	1	1	1
External ROM external bus mode 16 bits bus vector 16 bits width (Address data multiplex)	0	0	1	1	0	0	1
External ROM external bus mode 8 bits (Address data bus multiplex)	0	0	0	1	0	0	0
External ROM external bus mode 8 bits (Address data bus non-multiplex)	0	1	0	1	0	1	0

**Note:**

The maximum accessible capacity is 64 Kbytes when you control the output of the upper addresses A23 to A16.

#### ■ Operation of External Pin in Each Mode

Table 7.3-4 shows the operation relation of each external pin in the non-multiplex mode and the multiplex mode.

**Table 7.3-4 Operation Relation of External Pin in Each Mode**

	Functions							
	Non- multiplex mode				Multiplex mode			
	External Address control				External Address control			
	Permission (Address)		Prohibit (port)		Permission (Address)		Prohibit (port)	
	External bus expansion		External bus expansion		External bus expansion		External bus expansion	
	8 bits	16 bits	8 bits	16 bits	8 bits	16 bits	8 bits	16 bits
P07 to P00/ D07 to D00/ AD07 to AD00	D07 to D00				AD07 to AD00			
P17 to P10/ D15 to D08/ AD15 to AD08	Port	D15 to D08	Port	D15 to D08	A15 to A08	AD15 to AD08	A15 to A08	AD15 to AD08
P27 to P20	A23 to A16		Port		A23 to A16		Port	
P37 to P30	A07 to A00		A07 to A00		Port			
P47 to P40	A15 to A08		A15 to A08					
ALE	ALE				ALE			
$\overline{RD}$	$\overline{RD}$				$\overline{RD}$			
P52/ $\overline{WRL}$	$\overline{WRL}$				$\overline{WRL}$			
P53/ $\overline{WRH}$	Port	$\overline{WRH}$	Port	$\overline{WRH}$	Port	$\overline{WRH}$	Port	$\overline{WRH}$
P54/HRQ	HRQ				HRQ			
P55/ $\overline{HAK}$	$\overline{HAK}$				$\overline{HAK}$			
P56/ $\overline{RDY}$	RDY				RDY			
P57/CLK	CLK				CLK			

Notes:

- The single chip mode is all available as the port.
- The upper addresses, ( $\overline{WRL}$ ), ( $\overline{WRH}$ ), ( $\overline{HAK}$ ), HRQ, RDY, and CLK are used as a port according to the function selection.
- While using the multiplex, you can use the upper addresses A23 to A20 as a port or PPG3 to PPG0.
- While using the non-multiplex, you may not use the reload time ch.0/ch.1/ch.2, UART ch.0/ch.1. It works as an address.

## 7.4 External Memory Access

The block diagram of the external memory access, the configuration/function of the register, and the operation of the external memory access are described.

### ■ I/O Signal Pin of External Memory Access

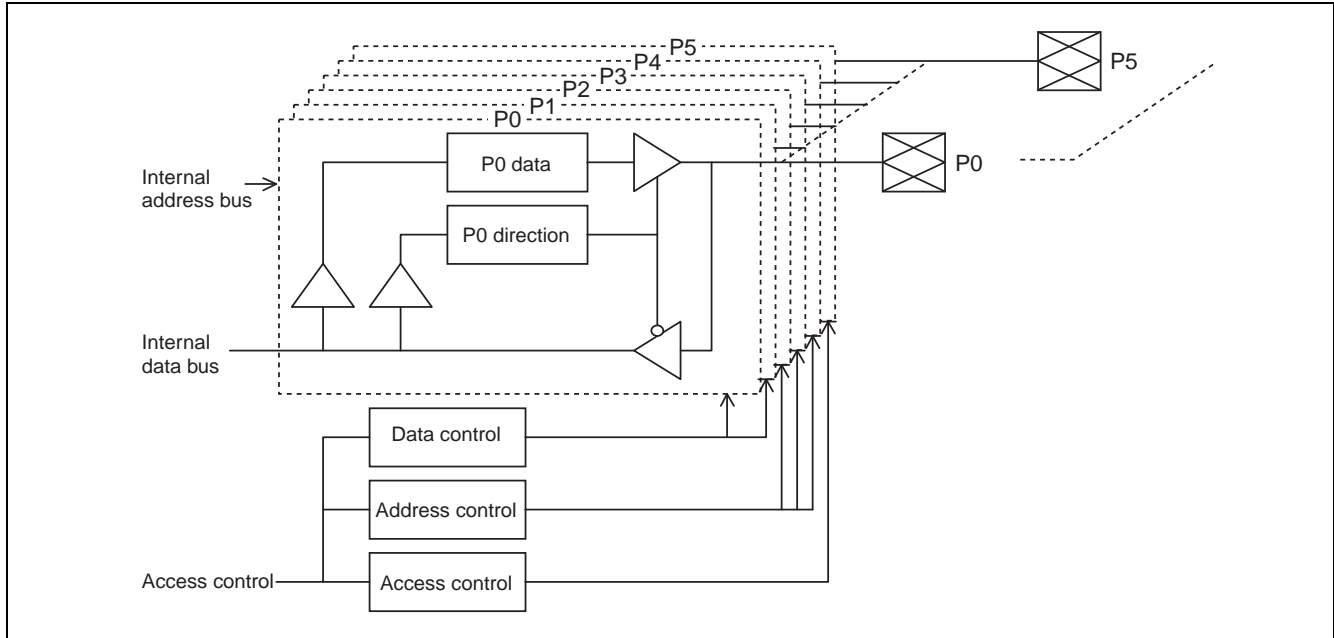
F<sup>2</sup>MC-16LX provides the following address/data/control signal to access the external memory/peripheral.

- CLK(P57): Machine cycle clock (KBP) is output.
- RDY(P56): It is an external ready input pin.
- ( $\overline{\text{HAK}}$ )(P55): It is a hold acknowledge output pin.
- HRQ(P54): It is a holding request input pin.
- ( $\overline{\text{WRH}}$ )(P53): It is a writing signal of a data bus upper byte.
- ( $\overline{\text{WRL}}$ )(P52): It is a data bus lower 8 bit write signal.
- ( $\overline{\text{RD}}$ )(P51): It is a reading signal.
- ALE(P50): It is an address latch permission signal (enabled in the multiplex mode).

## ■ Block Diagram

Figure 7.4-1 shows the block diagram of the external bus pin control circuit.

### Figure 7.4-1 Block Diagram of External Bus Pin Control Circuit



## ■ Register List

Figure 7.4-2 shows the list of the registers of the external bus pin control circuit.

**Figure 7.4-2 List of the Registers of External Bus Pin Control Circuit**

bit	15	14	13	12	11	10	9	8	
0000A5 <sub>H</sub>	Reserved	Reserved	HMR1	HMR0	—	—	LMR1	LMR0	Automatic Ready Function Selection Register (ARSR)
	(W)	(W)	(W)	(W)	(-)	(-)	(W)	(W)	Read/write
	(0)	(0)	(1)	(1)	(-)	(-)	(0)	(0)	Initial value
bit	7	6	5	4	3	2	1	0	
0000A6 <sub>H</sub>	E23	E22	E21	E20	E19	E18	E17	E16	External Address Output Control Register (HACR)
	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	Read/write
	(*)	(*)	(*)	(*)	(*)	(*)	(*)	(*)	Initial value
bit	15	14	13	12	11	10	9	8	
0000A7 <sub>H</sub>	CKE	RYE	HDE	Reserved	HMBS	WRE	LMBS	—	Bus Control Signal Selection Register (EPCR)
	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(-)	Read/write
	(1)	(0)	(0)	(0)	(*)	(1)	(0)	(-)	Initial value

W: Write only

## 7.4.1 Automatic Ready Function Selection Register (ARSR)

The configuration and functions of the automatic ready function selection register (ARSR) are described.

### ■ Automatic Ready Function Selection Register (ARSR)

Figure 7.4-3 shows the bit configuration of the automatic ready function selection register (ARSR).

**Figure 7.4-3 Bit Configuration of automatic Ready Function Selection Register (ARSR)**

bit	15	14	13	12	11	10	9	8	Automatic Ready Function Selection Register (ARSR)
0000A5 <sub>H</sub>	Reserved	Reserved	HMR1	HMR0	—	—	LMR1	LMR0	Read/write Initial value
	(W)	(W)	(W)	(W)	(-)	(-)	(W)	(W)	
	(0)	(0)	(1)	(1)	(-)	(-)	(0)	(0)	

W: Write only

Function of each bit for the automatic ready function selection register (ARSR) shows as follows.

#### [bit 15, bit 14] Reserved

It is Reserved bit. Please write "00<sub>B</sub>".

#### [bit 13, bit 12] HMR1, HMR0

The automatic wait function is selected when executing the external access to the area 800000<sub>H</sub> to FFFFFFF<sub>H</sub>. A set content is a table below.

HMR1	HMR0	Setting
0	0	Automatic wait interdiction
0	1	At the external access, an automatic wait of 1 machine cycle
1	0	At the external access, an automatic wait of 2 machine cycle
1	1	At the external access, an automatic wait of 3 machine cycles [Initial value]

#### [bit 11, bit 10] Undefined bit

Nothing is affected when it is written.

#### [bit 9, bit 8] LMR1, LMR0

The automatic wait function is selected when executing the external access to the area 007100<sub>H</sub> to 7FFFFFF<sub>H</sub> (however, the extended I/O area 007900<sub>H</sub> to 007FFF<sub>H</sub> is disable). A set content is a table below.

LMR1	LMR0	Setting
0	0	Automatic wait interdiction [Initial value]
0	1	At the external access, an automatic wait of 1 machine cycle
1	0	At the external access, an automatic wait of 2 machine cycle
1	1	At the external access, an automatic wait of 3 machine cycle

## 7.4.2 External Address Output Control Register (HACR)

The configuration and functions of the external address output control register are described.

### ■ External Address Output Control Register (HACR)

Figure 7.4-4 shows the bit configuration of the external address output control register.

**Figure 7.4-4 Bit Configuration of External Address Output Control Register (HACR)**

bit	7	6	5	4	3	2	1	0	
0000A6H	E23	E22	E21	E20	E19	E18	E17	E16	External Address Output Control Register (HACR)
	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	Read/write
	(*)	(*)	(*)	(*)	(*)	(*)	(*)	(*)	Initial value
W: Write only									

The external address output control register controls the output of the address (A23 to A16) to the outside. Each bit corresponds to the addresses A23 to A16 respectively, and controls each address output pin as shown in the following table.

0	The corresponding pin is address output (AXX). [Initial value]
1	The corresponding pin is I/O port (PXX).

The HACR register is not accessible when the device is in the single chip mode. In that case, all the ports function as the I/O ports regardless of the value of the HACR register. All the bits of the HACR register are write-only and the read-out value is "1". Furthermore, when expecting an output of an address by selecting the address output, be sure to use DDR with "0".

The initial value is "1" only when starting in the internal vector mode. Otherwise, it is "0".

#### Note:

When using PPG, be sure to set to "1" (I/O port setting).

### 7.4.3 Bus Control Signal Selection Register (EPCR)

The configuration and functions of the bus control signal selection register are described.

#### ■ Bus Control Signal Selection Register (EPCR)

The bus control signal selection register sets the control functions of the bus operation in the external bus mode.

Figure 7.4-5 shows the bit configuration of the bus control signal selection register.

Figure 7.4-5 Bit Configuration of Bus Control Signal Selection Register (EPCR)

bit	15	14	13	12	11	10	9	8	
0000A7 <sub>H</sub>	CKE	RYE	HDE	Reserved	HMBS	WRE	LMBS	—	Bus Control Signal Selection Register (EPCR)
	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(-)	Read/write
	(1)	(0)	(0)	(0)	(*)	(1)	(0)	(-)	Initial value
W: Write only									

The EPCR register is not accessible when the device is in the single chip mode. In the single chip mode, all pins function as the I/O ports regardless of the register value. All the bits of the EPCR register are write-only and the read-out value is "1".

The functions of each bit of the bus control signal selection register are described below.

#### [bit 15] CKE

The output of external clock (CLK) is controlled.

0	I/O port (P57) operation (clock interdiction)
1	Clock signal (CLK) output permission [Initial value]

#### [bit 14] RYE

The input of external ready (RDY) is controlled.

0	I/O port (P56) operation (external RDY input prohibited) [Initial value]
1	External ready (RDY) input permission

#### [bit 13] HDE

This is the bit that specifies the I/O permission of hold related pin. The setting controls the hold request input (HRQ) and the hold acknowledge output ( $\overline{\text{HAK}}$ ).

0	I/O port (P55, P54) operation (hold function input/output prohibited) [Initial value]
1	Hold Request (HRQ) Input/Hold acknowledge ( $\overline{\text{HAK}}$ ) output permission

#### [bit 12] Reserved

Reserved bit. Please write "0".

## [bit 11] HMBS

Specify the bus size when accessing the external bus to the area 800000<sub>H</sub> to FFFFFFF<sub>H</sub> in the external data bus 16-bit mode.

0	Bus size access of 16 bits (Initial Value in the external vector mode 1)
1	Bus size access of 8 bits (Initial Value in the external vector mode 0.2)

## [bit 10] WRE

Control the output of the external write signal ( $\overline{\text{WRH}}/\overline{\text{WRL}}$ ) pins in the external data bus 16-bit mode, and ( $\overline{\text{WRL}}$ ) pin in the external data bus 8-bit mode).

0	I/O Port (P53, P52) operation (Write signal output permission)
1	Write strobe signal ( $\overline{\text{WRH}}/\overline{\text{WRL}}$ or $\overline{\text{WRL}}$ only) output permission. [Initial value]

## [bit 9] LMBS

Specify the bus size when accessing the external bus to the area 007100<sub>H</sub> to 7FFFFFF<sub>H</sub> (however, the extended I/O area 007900<sub>H</sub> to 007FFF<sub>H</sub> is disable) in the external data bus 16-bit mode.

0	16-bit bus size access [Initial value]
1	8-bit bus size access

### Note:

In the external data bus 16-bit mode, when you permit the  $\overline{\text{WRH}}/\overline{\text{WRL}}$  function by the WRE bit, be sure to place P53/P52 in the input mode (set the bit 3 and 2 of DDR5 to "0").

In addition, the I/O function of the port is enabled even when the RDY and HRQ inputs by RYE and HDE bits are permitted. Therefore, be sure to set to "0" (input mode) for the bit corresponding to the port in DDR5.



## 7.5 Operation in Each Mode of Mode Setting

---

Operation in each mode of mode setting is described in the timing chart.

---

### ■ Mode Type

Operations of the following items are described for each function.

- External memory access control signal
  - External data bus 8-bit mode (non-multiplex mode)
  - External data bus 8-bit mode (multiplex mode)
  - External data bus 16-bit mode (non-multiplex mode)
  - External data bus 16-bit mode (multiplex mode)
- Ready Function
  - Non-multiplex mode
  - Multiplex mode
- Holding function
  - Non-multiplex mode
  - Multiplex mode

## 7.5.1 External Memory Access Control Signal

The access to the external memory is performed at 3 cycles unless the ready function is not used.

### External Memory Access Control Signal

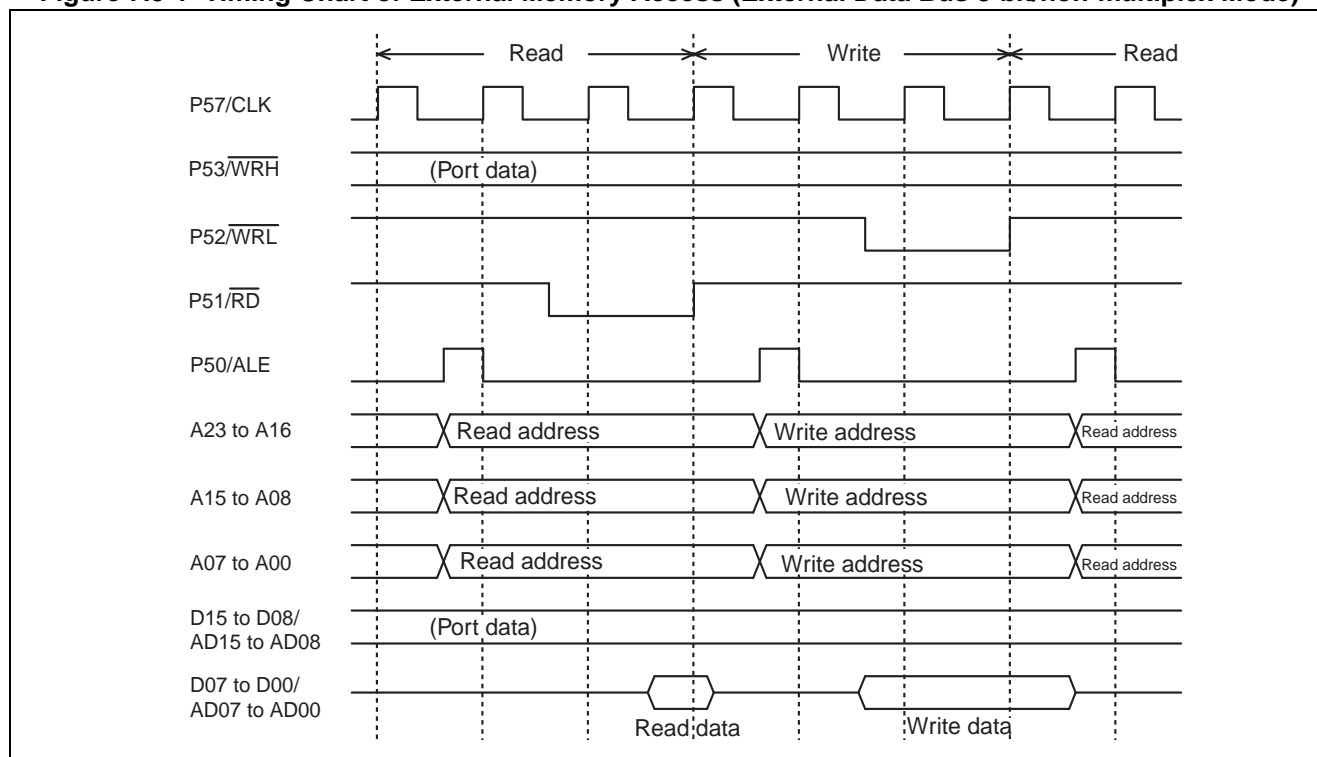
Figure 7.5-1 to Figure 7.5-4 show the timing charts of the external access in each mode. The 8-bit bus width access in the external data bus 16-bit mode reads or writes the 8-bit width peripheral chip when you connect to the external bus by mixing the 8-bit width peripheral chip and the 16-bit width peripheral chip. Be sure to connect the 8-bit width peripheral chip to the lower 8-bit of data because the 8-bit bus width access is executed by the lower 8-bit of the data bus. Whether performing the 16-bit bus width access or the 8-bit bus width access in the external data bus 16-bit mode is determined by the specification of the HMBS/LMBS bit of EPCR. Furthermore, in the multiplex mode, any actual bus operation may not be performed by executing only the address output and the assert output of ALE and not asserting  $\overline{RD}/\overline{WRL}/\overline{WRH}$ .

Note:

Be sure not to execute the access to the peripheral chip only by using the ALE signal.

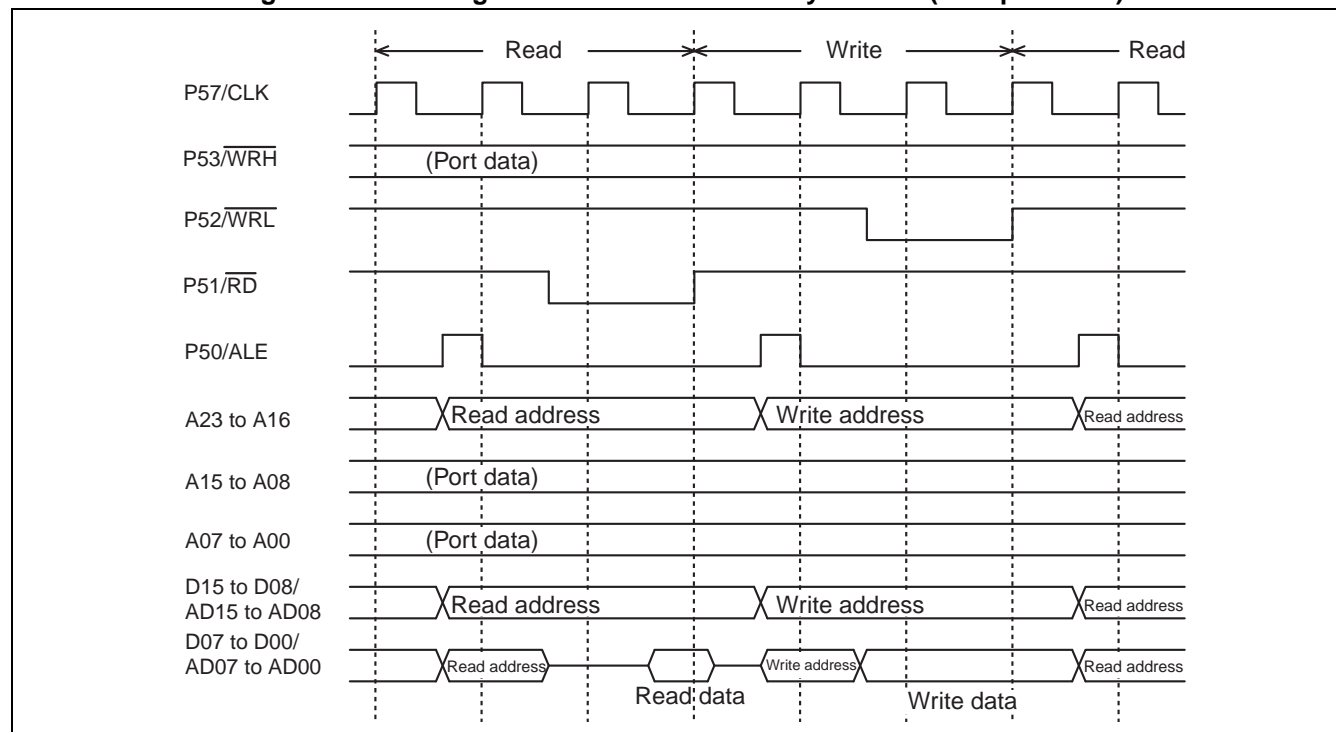
#### ● External data bus 8-bit mode (non-multiplex mode)

**Figure 7.5-1 Timing Chart of External Memory Access (External Data Bus 8-bit/non-multiplex Mode)**



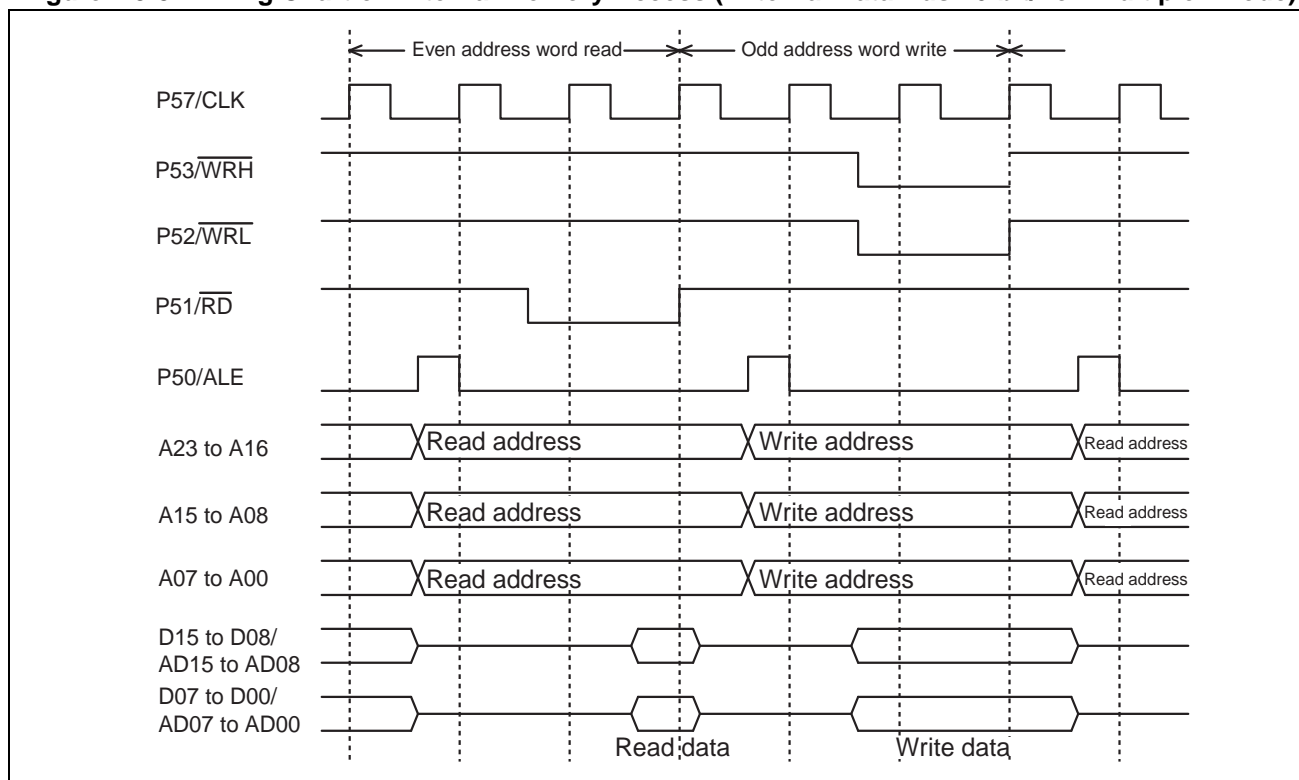
● External Data Bus 8-Bit Mode (External Data Bus 8-Bit/Multiplex Mode)

**Figure 7.5-2 Timing Chart of External Memory Access (Multiple Mode)**



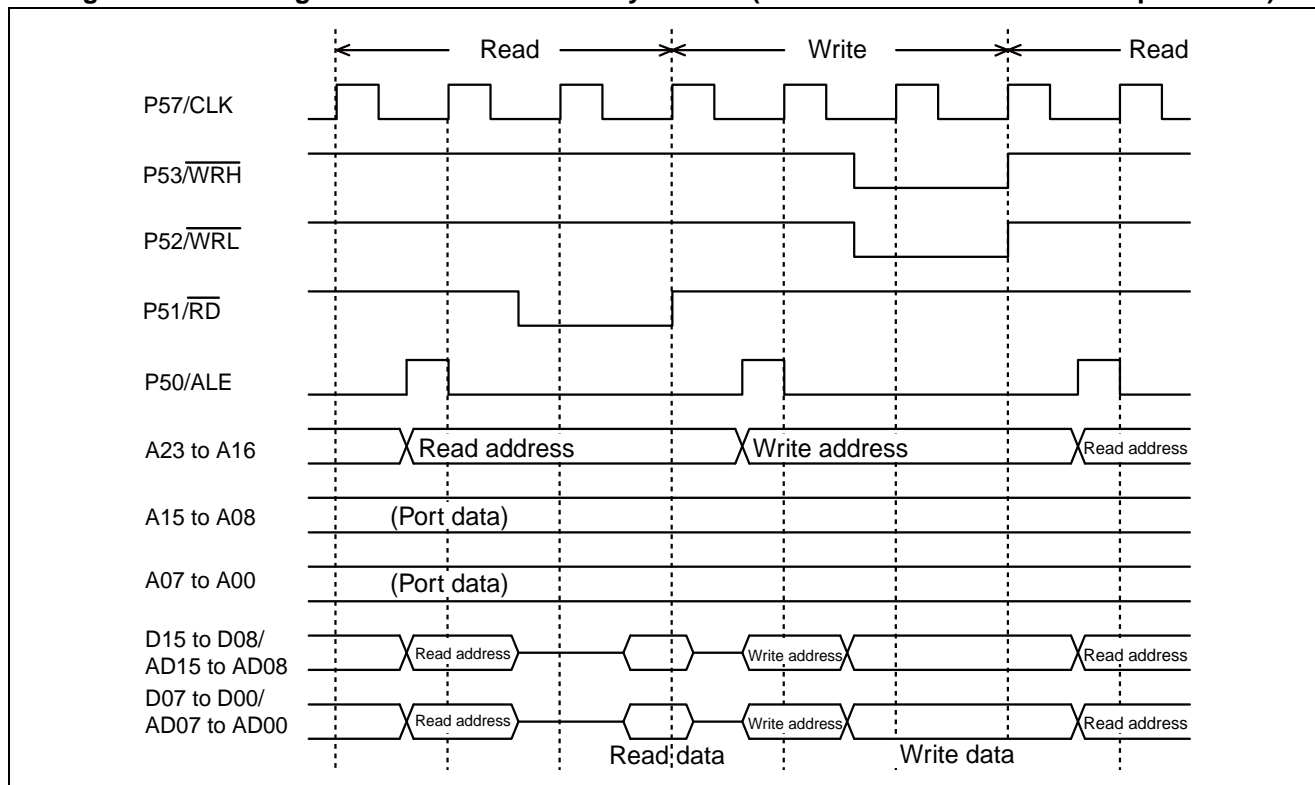
● External Data Bus 16-bit Mode (External Data Bus 16-bit/Non-Multiplex Mode)

**Figure 7.5-3 Timing Chart of External Memory Access (External Data Bus 16-bit/non-multiplex Mode)**



## ■ External 16-bit Bus Mode (External Data Bus 16-bit/Multiplex Mode)

Figure 7.5-4 Timing Chart of External Memory Access (External Data Bus 16-bit/multiplex Mode)



## 7.5.2 Ready Function

---

The setting of the P56/RDY pin or the automatic ready function selection register (ARSR) enables the access to the low-speed memory and the peripheral circuits. When the RYE bit in the bus control signal selection register (EPCR) is set to "1", the condition is in the wait cycle while the period "L" level is being input to the P56/RDY pin at the time of accessing to the external area, and then it is possible to extend the access cycle.

---

### ■ Ready Function

The F<sup>2</sup>MC-16LX has two built-in auto-ready functions for the external memory. The auto-ready function inserts the wait cycle of 1 to 3 cycles without the external circuit and extends the access cycle, when an access to the lower address external area located between the addresses 007100<sub>H</sub> to 7FFFFF<sub>H</sub> (however, the extend I/O area 007900<sub>H</sub> to 007FFF<sub>H</sub> is disable) is generated, and when an access to the upper address external area located between the addresses 800000<sub>H</sub> to FFFFFFFF<sub>H</sub> is generated. The auto-ready function is activated by the setting of the LMR1/LMR0 bit (lower address external area) in the ARSR and the HMR1/HMR0 bit (upper address external area) in the ARSR.

In the auto-ready functions for the external memory or the external I/O, when the RYE bit in the EPCR is set to "1", the wait cycle is maintained if the "L" level is input into the P56/RDY pin after the wait cycle ends by the above auto-ready function.

The timing charts of the ready functions in the non-multiplex mode and the multiplex mode are illustrated below. In both modes, the above chart does not set the ready function; the following chart sets this function.

---

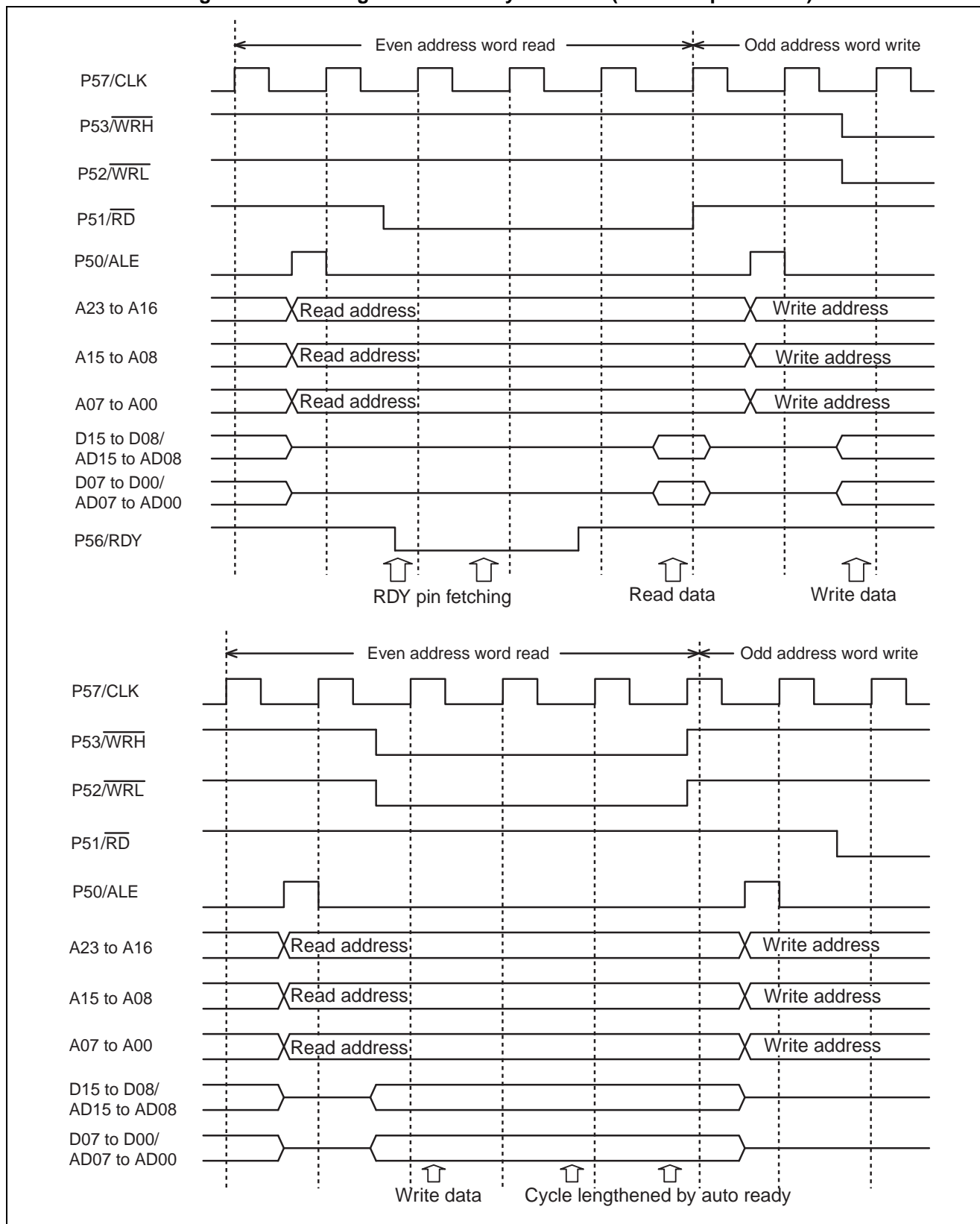
#### Note:

When inputting from the RDY pin, note that the device may run away if the AC standard is not satisfied.

---

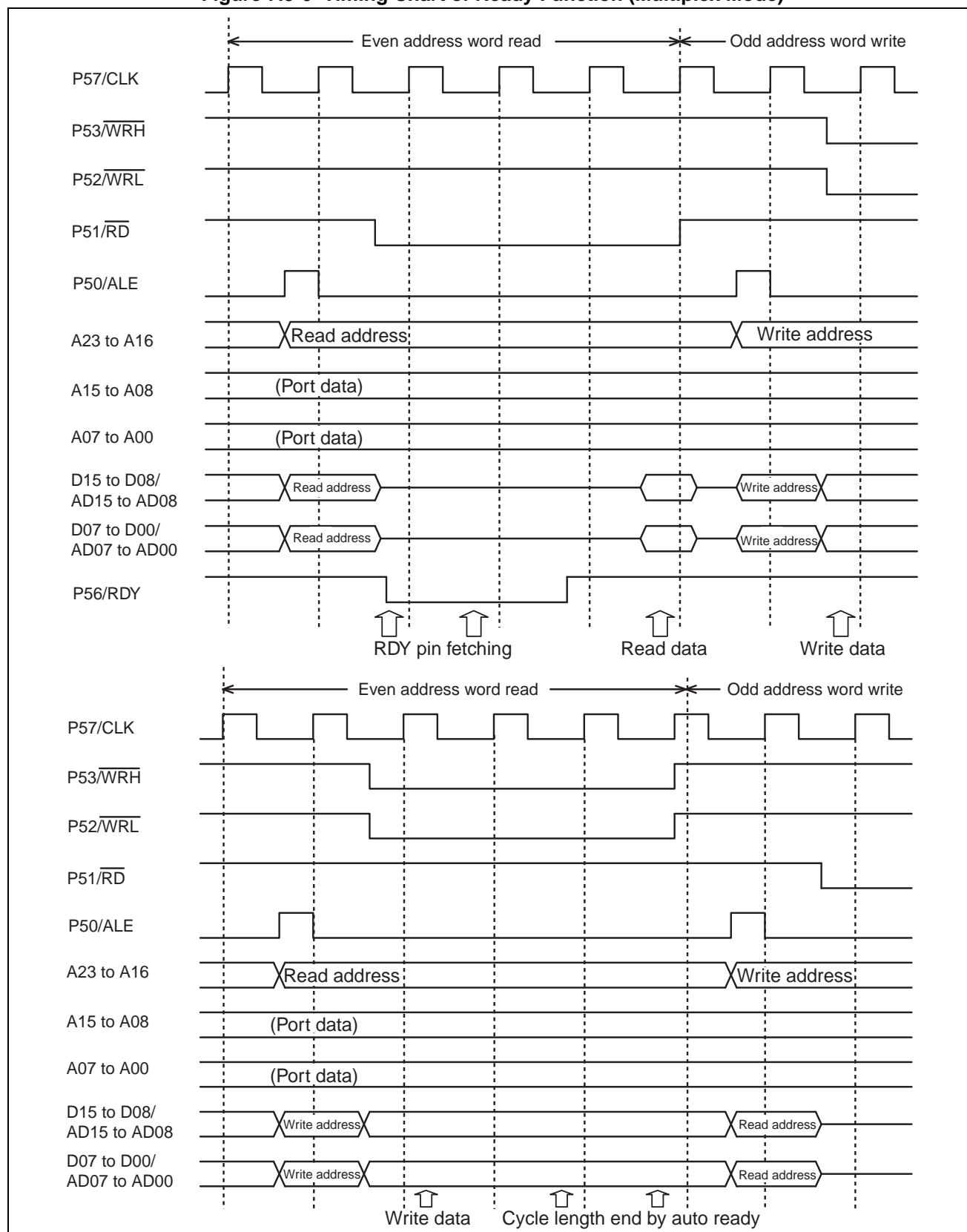
● Non-multiplex mode

**Figure 7.5-5 Timing Chart of Ready Function (Non-multiplex Mode)**



● Multiplex mode

Figure 7.5-6 Timing Chart of Ready Function (Multiplex Mode)



### 7.5.3 Holding Function

---

The operation of the hold function is described in the timing chart.

---

#### ■ Operation of Holding Function

If the HDE bit in the EPCR is set to "1", the hold function of the external bus by both the P54/HRQ and the P55/ $\overline{\text{HAK}}$  pins are enabled. When the "H" level is input to the P54/HRQ pin, the pin enters in the hold state when CPU's instruction ends (for the string instruction, when 1 element data procedure ends), and outputs the "L" level from the P55/ $\overline{\text{HAK}}$  and the following pin enters into the high-impedance state.

##### ● Non-multiplex mode

- Address output: A23 to A00
- Data I/O: D15/AD15 to D00/AD00
- Bus control signal: P51/ $\overline{\text{RD}}$ , P52/ $\overline{\text{WRL}}$ , P53/ $\overline{\text{WRH}}$

##### ● Multiplex mode

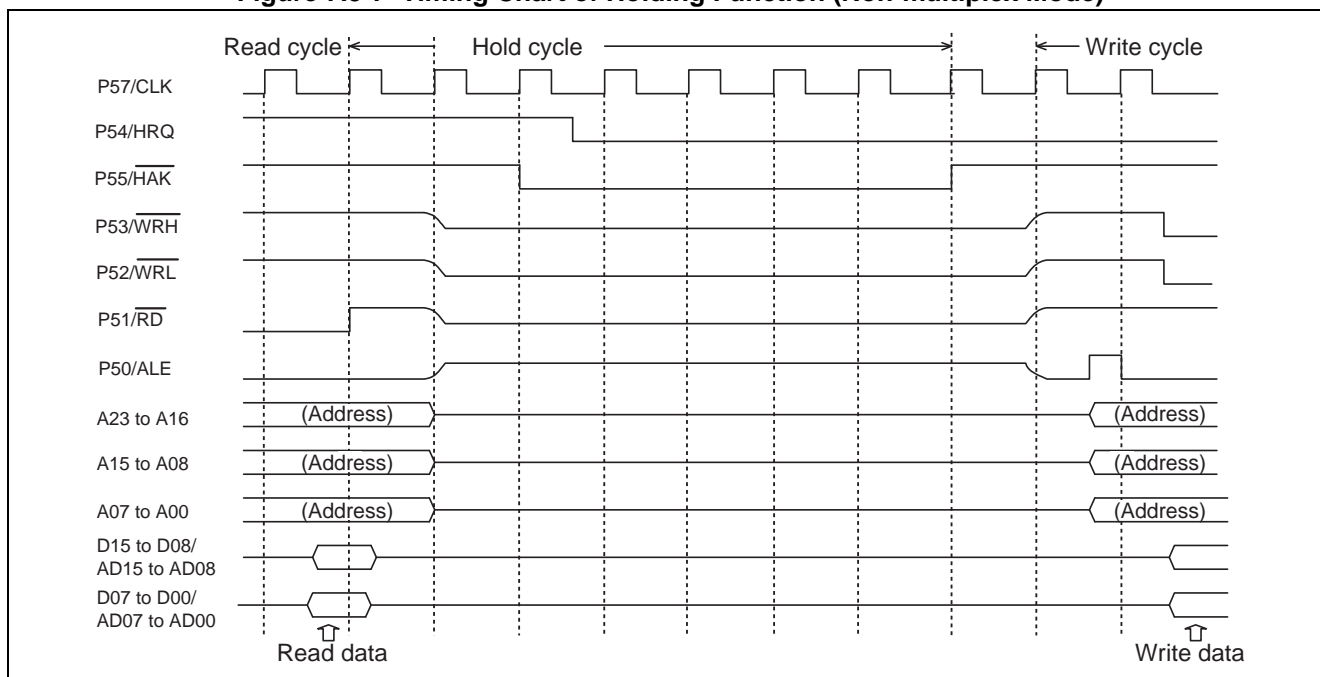
- Address output: A23 to A16
- Data I/O: D15/AD15 to D00/AD00
- Bus control signal: P51/ $\overline{\text{RD}}$ , P52/ $\overline{\text{WRL}}$ , P53/ $\overline{\text{WRH}}$

Thus, the device external circuit enables the use of the external bus. If the "L" level is input to the P54/HRQ pin, the P55/ $\overline{\text{HAK}}$  pin becomes the "H" level output and the external pin state is restored, and then the CPU restarts the operation. In the state of STOP, the holding demand is not accepted.

#### ■ Non-multiplex Mode

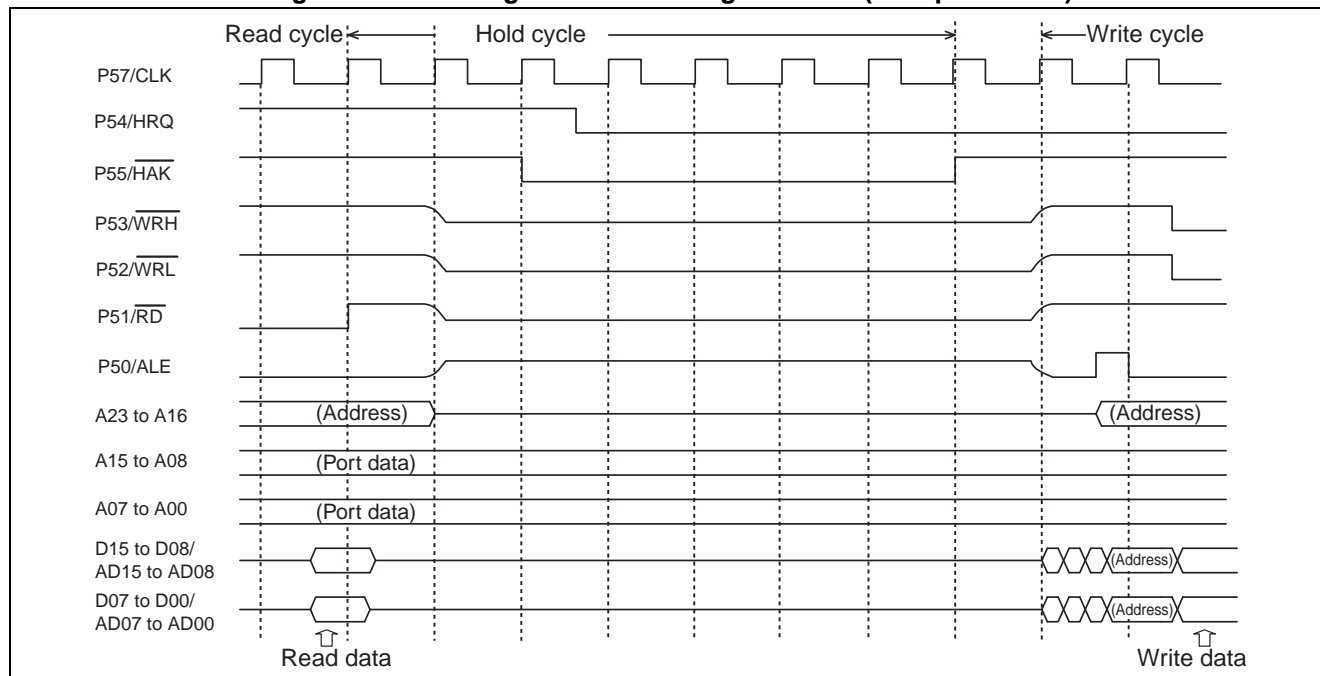
Figure 7.5-7 shows the timing chart of the hold function of the non-multiplex mode in the external data bus 16-bit mode.



**Figure 7.5-7 Timing Chart of Holding Function (Non-multiplex Mode)**

### ■ Multiple Mode

Figure 7.5-8 shows the timing chart of the hold function of the multiplex mode in the external data bus 16-bit mode.

**Figure 7.5-8 Timing Chart of Holding Function (Multiplex Mode)**

Note:

When "H" level is inputted to the P54/HRQ pin, it is retained until P55 pin becomes "L" level.

# **CHAPTER 8**

---

## ***I/O PORT***

**This chapter describes the configuration and functions of the register used in the I/O port.**

8.1 Functions of I/O Ports

8.2 I/O Port Register

## **8.1 Functions of I/O Ports**

---

**The overview of functions of the I/O ports is shown.**

---

### **■ Functions of I/O Ports**

The I/O port outputs data from the CPU to the I/O pin and loads the signal input in the I/O pin in the CPU by using the port data register (PDR). In addition, the port can randomly set the direction of the input/output of the I/O pin in bit unit by the port direction register (DDR).

The MB90330A Series has 72 input/output pins and 22 open drain output pins.

P07 to P00, 17 to P10, P27 to P20, P37 to P30, P47 to P40, P57 to P50, P77 to P70, P87 to P80, P95 to P90, PB6, and PB5 are I/O ports; P67 to P60, P96, and PA7 to PA0, PB4 to PB0 are open drain pins.

## 8.2 I/O Port Register

---

The configuration and functions of the register used in the I/O port are described.

---

### ■ I/O Port Registers

There are the following registers in I/O port.

- Port data register (PDR0 to PDRB)
- Port direction register (DDR0 to DDRB)
- Input resistance register (RDR0,RDR1)
- Output terminal register (ODR4)
- Analog input enable register (ADER0,ADER1)

## 8.2.1 Port Data Register (PDR0 to PDRB)

The configuration and functions of the port data register (PDR0 to PDRB) are described.

### ■ Port Data Register (PDR0 to PDRB)

Figure 8.2-1 shows the list of the port data register (PDR0 to PDRB).

**Figure 8.2-1 List of Port Data Register (PDR0 to PDRB)**

PDR0	bit	7	6	5	4	3	2	1	0	Initial value	Access
Address : 000000 <sub>H</sub>		P07	P06	P05	P04	P03	P02	P01	P00	XXXXXXXX <sub>B</sub>	R/W *
PDR1	bit	15	14	13	12	11	10	9	8		
Address : 000001 <sub>H</sub>		P17	P16	P15	P14	P13	P12	P11	P10	XXXXXXXX <sub>B</sub>	R/W *
PDR2	bit	7	6	5	4	3	2	1	0		
Address : 000002 <sub>H</sub>		P27	P26	P25	P24	P23	P22	P21	P20	XXXXXXXX <sub>B</sub>	R/W *
PDR3	bit	15	14	13	12	11	10	9	8		
Address : 000003 <sub>H</sub>		P37	P36	P35	P34	P33	P32	P31	P30	XXXXXXXX <sub>B</sub>	R/W *
PDR4	bit	7	6	5	4	3	2	1	0		
Address : 000004 <sub>H</sub>		P47	P46	P45	P44	P43	P42	P41	P40	XXXXXXXX <sub>B</sub>	R/W *
PDR5	bit	15	14	13	12	11	10	9	8		
Address : 000005 <sub>H</sub>		P57	P56	P55	P54	P53	P52	P51	P50	XXXXXXXX <sub>B</sub>	R/W *
PDR6	bit	7	6	5	4	3	2	1	0		
Address : 000006 <sub>H</sub>		P67	P66	P65	P64	P63	P62	P61	P60	XXXXXXXX <sub>B</sub>	R/W *
PDR7	bit	15	14	13	12	11	10	9	8		
Address : 000007 <sub>H</sub>		P77	P76	P75	P74	P73	P72	P71	P70	XXXXXXXX <sub>B</sub>	R/W *
PDR8	bit	7	6	5	4	3	2	1	0		
Address : 000008 <sub>H</sub>		P87	P86	P85	P84	P83	P82	P81	P80	XXXXXXXX <sub>B</sub>	R/W *
PDR9	bit	15	14	13	12	11	10	9	8		
Address : 000009 <sub>H</sub>		—	P96	P95	P94	P93	P92	P91	P90	-XXXXXXXX <sub>B</sub>	R/W *
PDR A	bit	7	6	5	4	3	2	1	0		
Address : 00000A <sub>H</sub>		PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	XXXXXXXX <sub>B</sub>	R/W *
PDR B	bit	7	6	5	4	3	2	1	0		
Address : 00000C <sub>H</sub>		—	PB6	PB5	PB4	PB3	PB2	PB1	PB0	-XXXXXXXX <sub>B</sub>	R/W *

\*: The R/W access to the input/output port operates slightly different from the R/W access to the memory. Please note that the following operations are done.

- Input mode
  - When reading: Read the level of the corresponding pin.
  - When writing: Write into the latch for the input/output.
- Output mode
  - When reading: Read the value of the data register latch.
  - When writing: Output into the corresponding pin.

## MB90330A Series

### 8.2.2 Port Direction Register (DDR0 to DDRB)

The configuration and functions of the port direction register are described.

#### ■ Port Direction Register (DDR0 to DDRB)

Figure 8.2-2 shows the list of the port direction register (DDR0 to DDRB).

**Figure 8.2-2 List of Port Direction Register (DDR0 to DDRB)**

DDR0	bit	7	6	5	4	3	2	1	0	Initial value	Access
Address : 000010 <sub>H</sub>		D07	D06	D05	D04	D03	D02	D01	D00	00000000 <sub>B</sub>	R/W
DDR1	bit	15	14	13	12	11	10	9	8		
Address : 000011 <sub>H</sub>		D17	D16	D15	D14	D13	D12	D11	D10	00000000 <sub>B</sub>	R/W
DDR2	bit	7	6	5	4	3	2	1	0		
Address : 000012 <sub>H</sub>		D27	D26	D25	D24	D23	D22	D21	D20	00000000 <sub>B</sub>	R/W
DDR3	bit	15	14	13	12	11	10	9	8		
Address : 000013 <sub>H</sub>		D37	D36	D35	D34	D33	D32	D31	D30	00000000 <sub>B</sub>	R/W
DDR4	bit	7	6	5	4	3	2	1	0		
Address : 000014 <sub>H</sub>		D47	D46	D45	D44	D43	D42	D41	D40	00000000 <sub>B</sub>	R/W
DDR5	bit	15	14	13	12	11	10	9	8		
Address : 000015 <sub>H</sub>		D57	D56	D55	D54	D53	D52	D51	D50	00000000 <sub>B</sub>	R/W
DDR6	bit	7	6	5	4	3	2	1	0		
Address : 000016 <sub>H</sub>		D67	D66	D65	D64	D63	D62	D61	D60	00000000 <sub>B</sub>	R/W
DDR7	bit	15	14	13	12	11	10	9	8		
Address : 000017 <sub>H</sub>		D77	D76	D75	D74	D73	D72	D71	D70	00000000 <sub>B</sub>	R/W
DDR8	bit	7	6	5	4	3	2	1	0		
Address : 000018 <sub>H</sub>		D87	D86	D85	D84	D83	D82	D81	D80	00000000 <sub>B</sub>	R/W
DDR9	bit	15	14	13	12	11	10	9	8		
Address : 000019 <sub>H</sub>		—	D96	D95	D94	D93	D92	D91	D90	-00000000 <sub>B</sub>	R/W
DDRA	bit	7	6	5	4	3	2	1	0		
Address : 00001A <sub>H</sub>		DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	00000000 <sub>B</sub>	R/W
DDRB	bit	15	14	13	12	11	10	9	8		
Address : 00000D <sub>H</sub>		—	DB6	DB5	DB4	DB3	DB2	DB1	DB0	-00000000 <sub>B</sub>	R/W

#### ● When each terminal functions as a port

When each terminal functions as a port, controls the corresponding each terminal as follows.

- 0: Input mode.
- 1: Output mode

Reset sets to "0".

#### Note:

When accessing the DDR0 to DDRB registers by using the instruction of the read modify write system (instructions such as bit set) is made, the bit targeted by an instruction becomes the defined value, while the content of the output register set with the other bit changes to the input value to the pin. Therefore, be sure to write an expected value into PDR firstly, and then set DDR, and finally change to the output when changing the input pin to the output pin is made.

## 8.2.3 Other Registers

The configuration and functions of the register other than the port data register (PDR0 to PDRB) and the port direction register (DDR0 to DDRB) are described.

### ■ Port 0, 1 Pull-up Resistance Register (RDR0, RDR1)

Figure 8.2-3 shows the bit configuration of the pull-up resistance register (RDR0, RDR1).

Figure 8.2-3 Bit Configuration of Pull-up Resistance Register (RDR0,RDR1)

RDR0	bit	7	6	5	4	3	2	1	0	Initial value	Access
Address : 00001C <sub>H</sub>		RD07	RD06	RD05	RD04	RD03	RD02	RD01	RD00	00000000 <sub>B</sub>	R/W
RDR1	bit	15	14	13	12	11	10	9	8		
Address : 00001D <sub>H</sub>		RD17	RD16	RD15	RD14	RD13	RD12	RD11	RD10	00000000 <sub>B</sub>	R/W

The pull-up resistance register (RDR0, RDR1) determines whether the pull-up resistor is enabled or not in the input mode.

- 0: There is not a pull-up resistor none at input mode
- 1: There is a pull-up resistor at the input mode.

The RDR0 and RDR1 registers do not have any function in the output mode (no pull-up resistor).

The input/output register is decided by the setting of the direction register (DDR).

There is no pull-up resistor at the time of stop (SPL=1) (high impedance).

This function is disable for the external bus. Be sure not to write into the RDR0 and RDR1 registers.

### ■ Port 4 Output Terminal Register (ODR4)

Figure 8.2-4 shows the bit configuration of the output pin register (ODR4).

Figure 8.2-4 Bit Configuration of Output Terminal Register (ODR4)

ODR4	bit	15	14	13	12	11	10	9	8	Initial value	Access
Address : 00001B <sub>H</sub>		OD47	OD46	OD45	OD44	OD43	OD42	OD41	OD40	00000000 <sub>B</sub>	R/W

The output terminal register (ODR4) executes open drain control in the output mode.

- 0: Entering in the standard output port in the output mode
- 1: Entering in the open drain output port in the output mode

The output terminal register (ODR4) does not have it function in the input mode (Output of Hi-Z).

The input/output mode is decided by the setting of the direction register (DDR).

This function is disable for the external bus. Do not write to the output terminal register (ODR4).

## ■ Analog Input Enable Register (ADER0, ADER1)

Figure 8.2-5 shows the bit configuration of the analog input permission registers (ADER0, ADER1).

**Figure 8.2-5 Bit Configuration of analog Input Enable Register (ADER0, ADER1)**

ADER0	bit	7	6	5	4	3	2	1	0	Initial value	Access
Address : 00001E <sub>H</sub>		ADE07	ADE06	ADE05	ADE04	ADE03	ADE02	ADE01	ADE00	11111111 <sub>B</sub>	R/W
ADER1	bit	15	14	13	12	11	10	9	8		
Address : 00001F <sub>H</sub>		ADE15	ADE14	ADE13	ADE12	ADE11	ADE10	ADE09	ADE08	11111111 <sub>B</sub>	R/W

The analog input enable registers (ADER0, ADER1) control each pin of port 7, 8 as follows;

- 0: Entering in the port I/O mode
- 1: Entering in the analog input mode. This bit becomes 0 after a reset.





# CHAPTER 9

---

## ***TIME-BASE TIMER***

**This chapter describes the function and operation of the time-base timer.**

- 9.1 Overview of Time-base Timer
- 9.2 Configuration of Time-base Timer
- 9.3 Time-base Timer Control Register (TBTC)
- 9.4 Interrupt of Time-base Timer
- 9.5 Operations of Time-base Timer
- 9.6 Precautions when Using Time-base Timer
- 9.7 Program Example of Time-base Timer

## 9.1 Overview of Time-base Timer

The time-base timer has an interval timer function that enables a selection of four interval times using 18-bit free-run counter (time-base counter) count-up with synchronizing to the internal count clock (2 division of original oscillation). Furthermore, the function of timer output of oscillation stabilization wait time or function supplying operation clocks for watchdog timer are provided.

### ■ Interval Timer Function

The interval timer function generates interrupt requests at regular intervals.

- An overflow of the bit for the interval timer in the time-base counter generates an interrupt request.
- You can select one of four bits (interval time) for the interval timer.

Table 9.1-1 shows the interval time of the time-base timer.

**Table 9.1-1 Interval Time of Time-base Timer**

Internal count clock cycle	Time of interval
2/HCLK(0.33 μs)	$2^{12}/\text{HCLK}$ (Approx. 0.68 ms)
	$2^{14}/\text{HCLK}$ (Approx. 2.7 ms)
	$2^{16}/\text{HCLK}$ (Approx. 10.9 ms)
	$2^{19}/\text{HCLK}$ (Approx. 87.4 ms)

HCLK: Oscillation clock

The value in parentheses is applicable when the oscillation clock operates at 6 MHz.

## ■ Function of Clock Supply

The clock supply function supplies the operating clock to the timer for oscillation stabilization wait time and some peripheral functions. Table 9.1-2 shows the clock cycle supplied from the time-base timer to each peripheral.

**Table 9.1-2 Clock Cycles Supplied from Time-base Timer**

Where to Supply Clock	Clock Cycle	Remark
Oscillation Stabilization Wait Time	$2^{13}/\text{HCLK}$ (Approx. 1.4 ms)	Oscillation Stabilization Wait Time for ceramic oscillator
	$2^{15}/\text{HCLK}$ (Approx. 5.5 ms)	Oscillation Stabilization Wait Time for crystal oscillator
	$2^{17}/\text{HCLK}$ (Approx. 21.8 ms)	
Watchdog timer	$2^{12}/\text{HCLK}$ (Approx. 0.68 ms)	Watchdog timer count up clock
	$2^{14}/\text{HCLK}$ (Approx. 2.7 ms)	
	$2^{16}/\text{HCLK}$ (Approx. 10.9 ms)	
	$2^{19}/\text{HCLK}$ (Approx. 87.4 ms)	

HCLK: Oscillation clock

The value in parentheses is applicable when the oscillation clock operates at 6 MHz.

### Reference:

Because oscillation cycles is unstabled after oscillation starts, the oscillation stabilization wait time is listed for reference.

## 9.2 Configuration of Time-base Timer

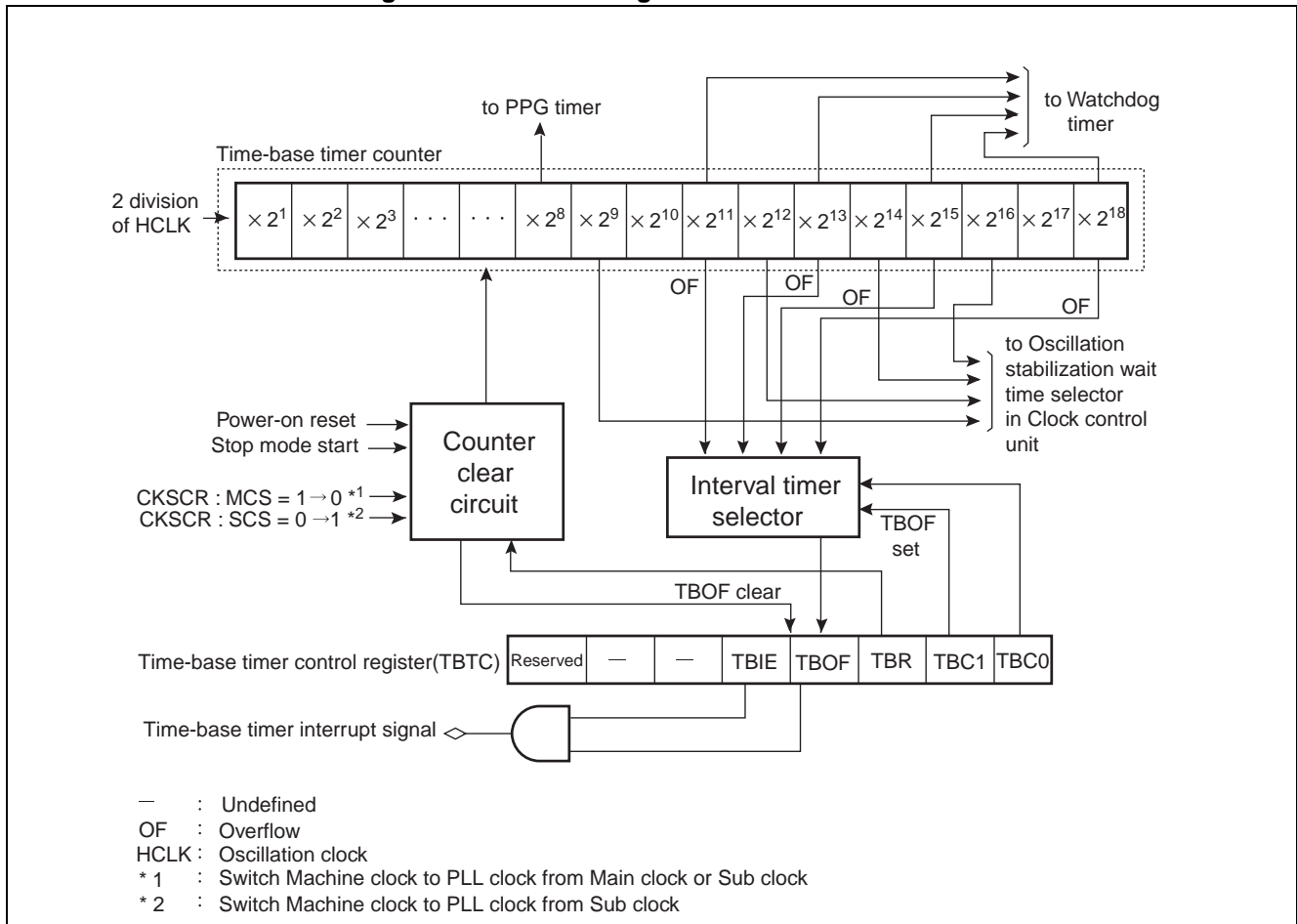
The time-base timer consists of the following four blocks.

- Time-base timer counter
- Counter clear circuit
- Interval timer selector
- Time-base timer control register (TBTC)

### ■ Block Diagram of Time-base Timer

Figure 9.2-1 shows the block diagram of the time-base timer.

Figure 9.2-1 Block Diagram of Time-base Timer



#### ● Time-base timer counter

This is an 18-bit up-counter whose count clock is two-division clock of oscillation clock (HCLK).

- Counter clear circuit

This circuit clears the counter by writing "0" to time-base timer initialization bit (TBR) of time-base timer control register (TBTC), power-on reset, transition to the stop mode, switching to PLL clock mode from the main clock mode or sub clock, or switching to the main clock mode from sub clock.

- Interval timer selector

This selects the output of the time-base timer counter from one of four types. The overflow of selected bit will be the interrupt cause.

- Time-base timer control register (TBTC)

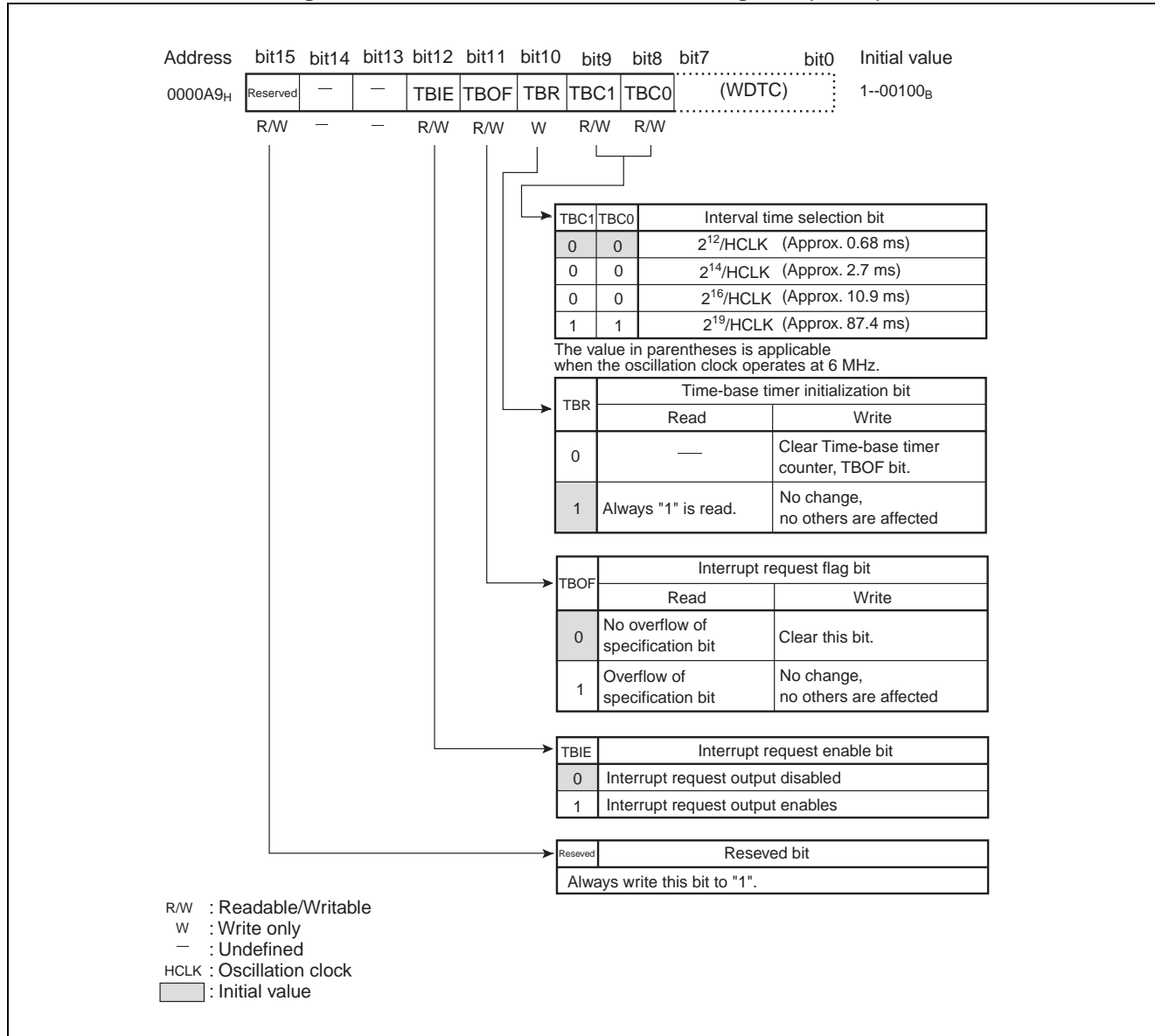
Interval time selection, counter clearance, and interrupt request control and status check are executed.

## 9.3 Time-base Timer Control Register (TBTC)

The time-base timer control register (TBTC) executes interval time selection, time-base timer counter clearance, and interrupt control and status check.

### ■ Time-base Timer Control Register (TBTC)

Figure 9.3-1 Time-base Timer Control Register (TBTC)



**Table 9.3-1 Time-base Timer Control Register (TBTC)**

Bit name		Functions
bit 15	Reserved: Reserved bit	Note: Be sure to write "1".
bit 14, bit 13	Unused bits	<ul style="list-style-type: none"> <li>• The value at the time of reading is irregular.</li> <li>• No effect on writing.</li> </ul>
bit 12	TBIE: Interrupt request enable bit	<ul style="list-style-type: none"> <li>• This bit permits or prohibits an interrupt request output to the CPU.</li> <li>• If the TBIE bit and interrupt request flag bit (TBOF) is set to "1", an interrupt request is output.</li> </ul>
bit 11	TBOF: Interrupt request flag bit	<ul style="list-style-type: none"> <li>• An overflow of time-base timer counter specification bit sets the status to "1".</li> <li>• If the TBOF bit and interrupt request permission bit (TBIE) is set to "1", an interrupt request is output.</li> </ul> <p>Setting to "0" executes clearance during writing and no changes are made at "1", making no influences on others.</p> <p>Note:</p> <ul style="list-style-type: none"> <li>• When clearing the TBOF bit, set to the condition that prohibits time-base timer interrupt with the interrupt request permission bit (TBIE) or by specifying interrupt level mask register (ILM) in processor status (PS).</li> <li>• The status is cleared to "0" by "0" writing, transition to the stop mode, transition from the sub clock mode to main clock mode, transition from the main clock mode to the PLL clock mode, "0" writing to time-base timer initialization bit (TBR), or resetting.</li> </ul>
bit 10	TBR: Time-base timer initialization bit	<ul style="list-style-type: none"> <li>• This bit clears the time-base timer counter.</li> <li>• Writing "0" clears the counter, immediately after that, clears the TBOF bit. No changes are made at "1", making no influences on others.</li> </ul> <p>Reference: Always read value is "1".</p>
bit 9, bit 8	TBC1, TBC0: Interval time select bits	<ul style="list-style-type: none"> <li>• These bits specify a cycle for the interval timer.</li> <li>• The bit for the interval timer of time-base timer counter is specified.</li> <li>• One of four interval time can be selected.</li> </ul>



## 9.4 Interrupt of Time-base Timer

The time-base timer can generate an interrupt request by the overflow of the specified bit of the time-base timer counter (interval timer function).

### ■ Interrupt of Time-base Timer

After the time-base counter undergoes count-up with the internal count clock and the bit for the selected interval timer overflows, the interrupt request flag bit (TBOF) of time-base timer control register (TBTC) is set to "1". In this case, an interrupt request to CPU is generated if interrupt request is permitted by setting the interrupt request permission bit (TBIE) is set to "1". Clear the interrupt request by writing "0" to the TBOF bit in the interrupt processing routine. The TBOF bit is set when the specified bit overflows regardless of the value of interrupt request permission bit (TBIE).

#### Note:

When clearing the interrupt request flag bit (TBOF) in the time-base timer control register (TBTC), perform while the time-base timer interrupt is prohibited by the setting of the interrupt level mask register (IML) of the interrupt request permission bit (TBIE) or the processor status (PS).

#### References:

- When the TBOF bit is "1", an interrupt request is immediately generated upon the transition of the TBIE bit from prohibition to permission ("0" to "1").
- EI<sup>2</sup>OS and  $\mu$ DMAC cannot be used.

### ■ Interrupt of Time-base Timer and EI<sup>2</sup>OS, $\mu$ DMAC

Table 9.4-1 shows the time-base timer interrupt and EI<sup>2</sup>OS,  $\mu$ DMAC.

**Table 9.4-1 Interrupt of Time-base Timer and EI<sup>2</sup>OS,  $\mu$ DMAC**

Interrupt number	Interrupt level setting register		Address in Vector Table			EI <sup>2</sup> OS	$\mu$ DMAC
	Register Name	Address	Low	High	Bank		
#40	ICR14	0000BE <sub>H</sub>	FFFF5C <sub>H</sub>	FFFF5D <sub>H</sub>	FFFF5E <sub>H</sub>	×	×

×: Not available

#### Note:

ICR14 can be used for three interrupts, time-base timer interrupt, watch timer interrupt, and UART reception end ch.0/1 interrupt, but the interrupt level is the same.

## 9.5 Operations of Time-base Timer

The time-base timer has functions of interval timer and clock supply to peripheral functions.

### ■ Operation of Interval Timer Function (Time-base Timer)

Interval timer function generates interrupt requests at regular intervals. In order to function as an interval timer, setup in Figure 9.5-1 is needed.

**Figure 9.5-1 Setting of Time-base Timer**

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit0
0000A9 <sub>H</sub> TBTC	Reserved	—	—	TBIE	TBOF	TBR	TBC1	TBC0	(WDTC)	
	1	—	—	⊙	0	0	⊙	⊙		

⊙ : Used bit  
 — : Unused bit  
 0 : Set to "0".  
 1 : Set to "1".

- The time-base timer counter continues count-up synchronizing to the internal count clock (two division of oscillation clock) as long as the clock oscillate.
- Count-up from "0" is made when the counter is cleared; the interrupt request flag bit (TBOF) is set to "1" when the bit for the interval timer overflows. In this case, interrupts are generated at selected intervals based on the cleared if the interrupt request output is permitted (TBIE = 1).
- The interval time may be longer than the set period by the clearance operation of the time-base timer.

### ■ Oscillation Stabilization Wait Time Function

The time-base timer can also be used for the oscillation clock or timer for PLL clock oscillation stabilization wait time. The oscillation stabilization wait time is the duration elapsed by counting up from "0" (count clear) to the moment at which the bit for oscillation stabilization wait time overflows. When the mode recovers from the time-base timer mode to the PLL clock mode or main clock mode, the oscillation stabilization wait time is the duration from the middle of counting because the time-base timer counter has not cleared. Table 9.5-1 shows the clearance operation of the time-base timer counter and oscillation stabilization wait time.

**Table 9.5-1 Time-base Timer Counter Clearance Operation and Oscillation Stabilization Wait Time**

Operation	Counter Clear	TBOF	Oscillation Stabilization Wait Time
Writing "0" to time-base timer initialization bit (TBR) of time-base timer control register (TBTC)	○	○	×
Power-on reset	○	○	Main clock oscillation stabilization wait time
Watchdog reset	×	○	
Release of main stop mode	○	○	
Release of PLL stop mode	○	○	
Release of Sub-stop mode	×	×	Sub clock oscillation stabilization wait time
Transition from main clock mode to PLL clock mode (MCS=1 → 0)	○	○	PLL clock oscillation stabilization wait time
Transition from sub clock mode to main clock mode (MCS=1 → 1)	○	○	Main clock oscillation stabilization wait time
Release of time-base timer mode	×	×	×
Release of sleep mode	×	×	×

○: Yes

×: None

## ■ Function of Clock Supply

The time-base timer supplies clock to the watchdog timer. The clearance of time-base counter effects the operation of watchdog timer.

## 9.6 Precautions when Using Time-base Timer

---

**Cautions about influences on peripheral functions due to interrupt request and time-base timer clearances.**

---

### ■ Precautions when Using Time-base Timer

#### ● Clearing Interrupt request

When clearing the interrupt request flag bit (TBOF) in the time-base timer control register (TBTC), perform while the time-base timer interrupt is masked by the setting of the interrupt level mask register (IML) of the interrupt request permission bit (TBIE) or the processor status (PS).

#### ● Effect from time-base timer clear

By clearing the time-base timer counter, the following operations are affected.

- When using the interval timer function (interval interrupt) in the time-base timer
- When using the watchdog timer

#### ● Using time-base timer as oscillation stabilization wait time

After power on, since the oscillation clock halts in the main stop mode, oscillation stabilization wait time of the oscillation clock will be needed using the operation clock supplied from the time-base timer after the oscillator starts operating. Selection of suitable oscillation stabilization wait time is necessary depending on the types of resonators connected to the high-speed oscillation pin. For details, see Section "5.5 Oscillation Stabilization Wait Time".

#### ● Notes on peripheral functions the time-base timer supplies to the clock

The mode that stops the main clock clears the counter, and then the time-base timer stops operation. The clock supplied from the time-base timer may shorten the "H" level or lengthen the "L" level for up to 1/2 cycle because the clock is supplied from the initial state when the counter of the time-base timer is cleared. The clock for the watchdog timer is also supplied from the initial status, however, the watchdog timer operates in the normal cycles because the counter of the watchdog timer is cleared at the same time.

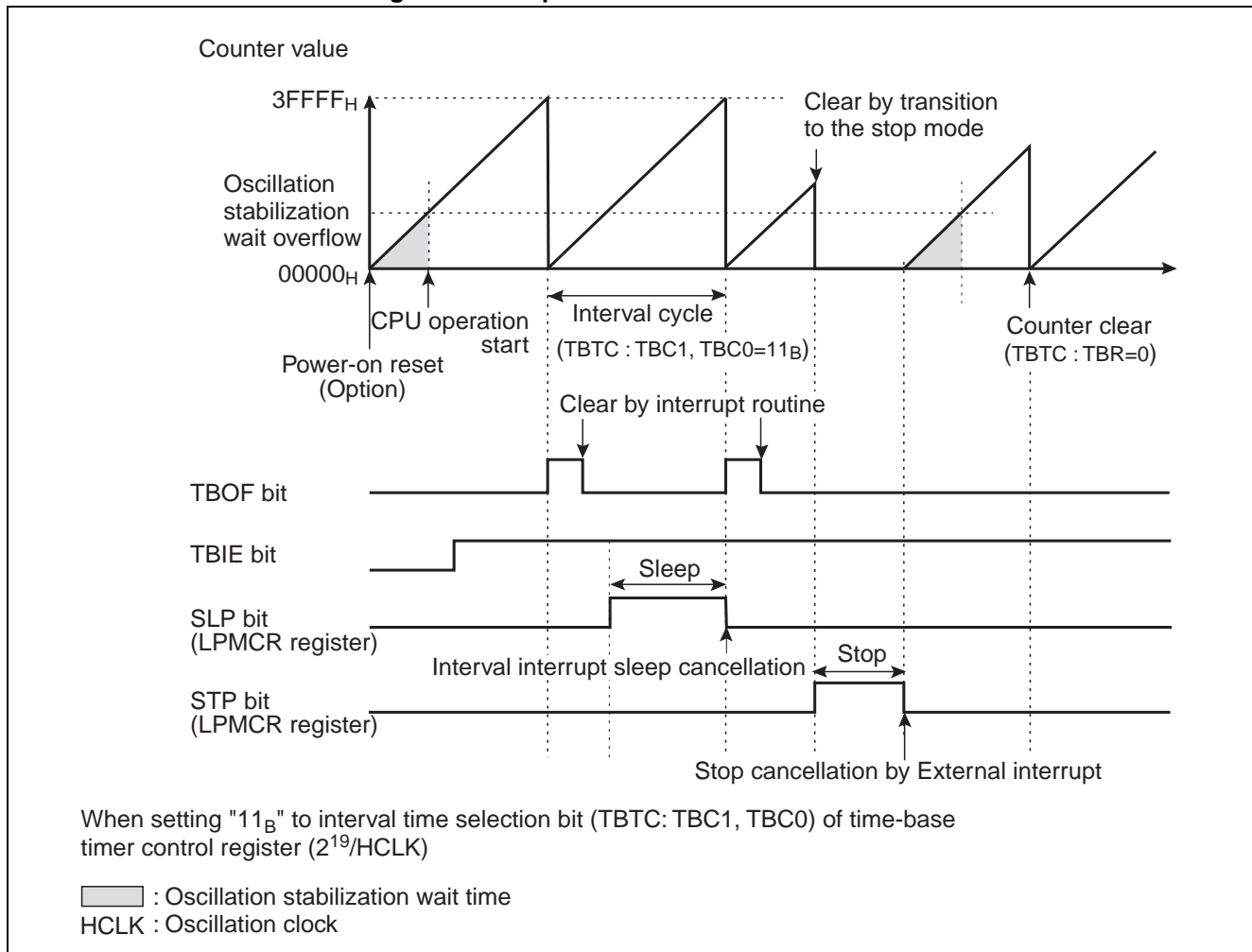
## ■ Operations of Time-base Timer

Operations in the following situations are shown in Figure 9.6-1.

- At a power-on reset occurs.
- At a transition to sleep mode during the operation of interval timer function
- At a transition to stop mode.
- At a counter clear request occurs.

The transition to the stop mode clears the time-base timer, terminating operations. Upon the recovery from the stop mode, oscillation stabilization wait time is counted with the time-base timer.

Figure 9.6-1 Operations of Time-base Timer



## 9.7 Program Example of Time-base Timer

Programming examples for the time-base timer are shown below.

### ■ Program Example of Time-base Timer

#### ● Processing specification

Interval interruptions of  $2^{12}/\text{HCLK}$  (oscillation clock) are repeatedly generated. In this case, the interval time is about 0.68 ms (at 6-MHz operation).

#### ● Coding example

```

ICR14    EQU    0000BEH            ; Interrupt control register for time-base timer
TBTC     EQU    0000A9H            ; Time-base timer control register
TBOF     EQU    TBTC:3             ; Interrupt request flag bit
;-----Main Program-----
CODE     CSEG
START:
;          :                      ; Initialize such as a stack pointer (SP)
;          AND    CCR, #0BFH        ; Disables the interrupt
;          MOV    I:ICR14, #00H     ; Interrupt level 0 (highest)
;          MOV    I:TBTC, #10010000B ; Upper 3 bits fixed
;                                     ; Interrupt enabled, TBOF Clear
;                                     ; Counter Clear
;                                     ; Interval Time Selection  $2^{12}/\text{HCLK}$  selection
;          MOV    ILM, #07H         ; Sets ILM in PS to level 7
;          OR     CCR, #40H         ; Interruption permission
LOOP:    MOV     A, #00H            ; infinite loop
;          MOV     A, #01H
;          BRA    LOOP
;-----Interrupt Program-----
WARI:
;          CLR    bit BOF           ; The interrupt request flag is clear
;          :
;          User processing
;          :
;          RETI                     ; Returns from interrupt
CODE     ENDS
;-----Vector Settings-----
VECT     CSEG    ABS=0FFH

```

```
                                ORG    0FF6CH                ; The interruption vector is set
                                DSL     WARI
                                ORG    0FFDCH                ; Reset vector setting
                                DSL     START
                                DB      00H                  ; Single-chip mode
VECT    ENDS
                                END     START
```

# ***CHAPTER 10***

---

# ***WATCHDOG TIMER***

**This chapter describes the function and operation of the watchdog timer.**

- 10.1 Overview of Watchdog Timer
- 10.2 Watchdog Timer Control Register (WDTC)
- 10.3 Configuration of Watchdog Timer
- 10.4 Operations of Watchdog Timer
- 10.5 Precautions when Using Watchdog Timer
- 10.6 Program Examples of Watchdog Timer



## 10.1 Overview of Watchdog Timer

The watchdog timer is a 2-bit counter operating with an output of the time-base timer or clock timer as the count clock and resets the CPU when the counter is not cleared for a preset period of time.

### ■ Functions of Watchdog Timer

The watchdog timer is a counter for preventing programs from hanging up. The timer must be cleared at specified intervals after being activated. If the watchdog timer is not cleared within a certain time due to an infinite loop of the program, etc., a watchdog reset is generated to the CPU. The interval time of the watchdog timer can be set by the watchdog timer control register (WDTC), as shown in Table 10.1-1. When the watchdog timer is not cleared, a watchdog reset occurs following the time between the minimum time interval and the maximum time interval. The counter must be cleared within the time of the minimum time interval.

**Table 10.1-1 Interval Time of Watchdog Timer**

WT1	WT0	WDCS & SCM	Interval Time		Clock cycle number
			Min. *	Max. *	
0	0	1	Approx. 2.39 ms	Approx. 3.07 ms	$(2^{14} \pm 2^{11})/\text{HCLK}$
0	1	1	Approx. 9.56 ms	Approx. 12.29 ms	$(2^{16} \pm 2^{13})/\text{HCLK}$
1	0	1	Approx. 38.23 ms	Approx. 49.15 ms	$(2^{18} \pm 2^{15})/\text{HCLK}$
1	1	1	Approx. 305.83 ms	Approx. 393.22 ms	$(2^{21} \pm 2^{18})/\text{HCLK}$
0	0	0	Approx. 0.448 s	Approx. 0.576 s	$(2^{12} \pm 2^9)/\text{SCLK}$
0	1	0	Approx. 3.584 s	Approx. 4.608 s	$(2^{15} \pm 2^{12})/\text{SCLK}$
1	0	0	Approx. 7.168 s	Approx. 9.216 s	$(2^{16} \pm 2^{13})/\text{SCLK}$
1	1	0	Approx. 14.336 s	Approx. 18.432 s	$(2^{17} \pm 2^{14})/\text{SCLK}$

\*: Value for when operating at oscillator clock (HCLK) of 6 MHz, sub clock (SCLK) of 32 kHz divided by 4 (= 8 kHz).

The maximum and minimum watchdog timer interval time and the number of oscillation clock cycles are determined by the timing of clear operation. The interval time will be 3.5 to 4.5 times of the count clock (supplied clock of time-base timer) cycle. For the watchdog timer interval time, see Section "10.4 Operations of Watchdog Timer".

#### Note:

The watchdog counter is a 2-bit counter that counts carry-up signals from the time-base timer. Therefore, when the time-base timer is cleared, the time period until the occurrence of a watchdog timer reset may be longer than the preset period of time.

---

Reference:

When the watchdog timer is activated, it is initialized and set to the stopped state by a reset upon power-on or by a reset by the watchdog. Also, the watchdog counter is cleared by writing to the reset by the external pin, the software reset, and the watchdog control bit (WTE) of the watchdog timer control register and by changing to sleep, stop, and watch mode, but the watchdog timer is still activated.

---

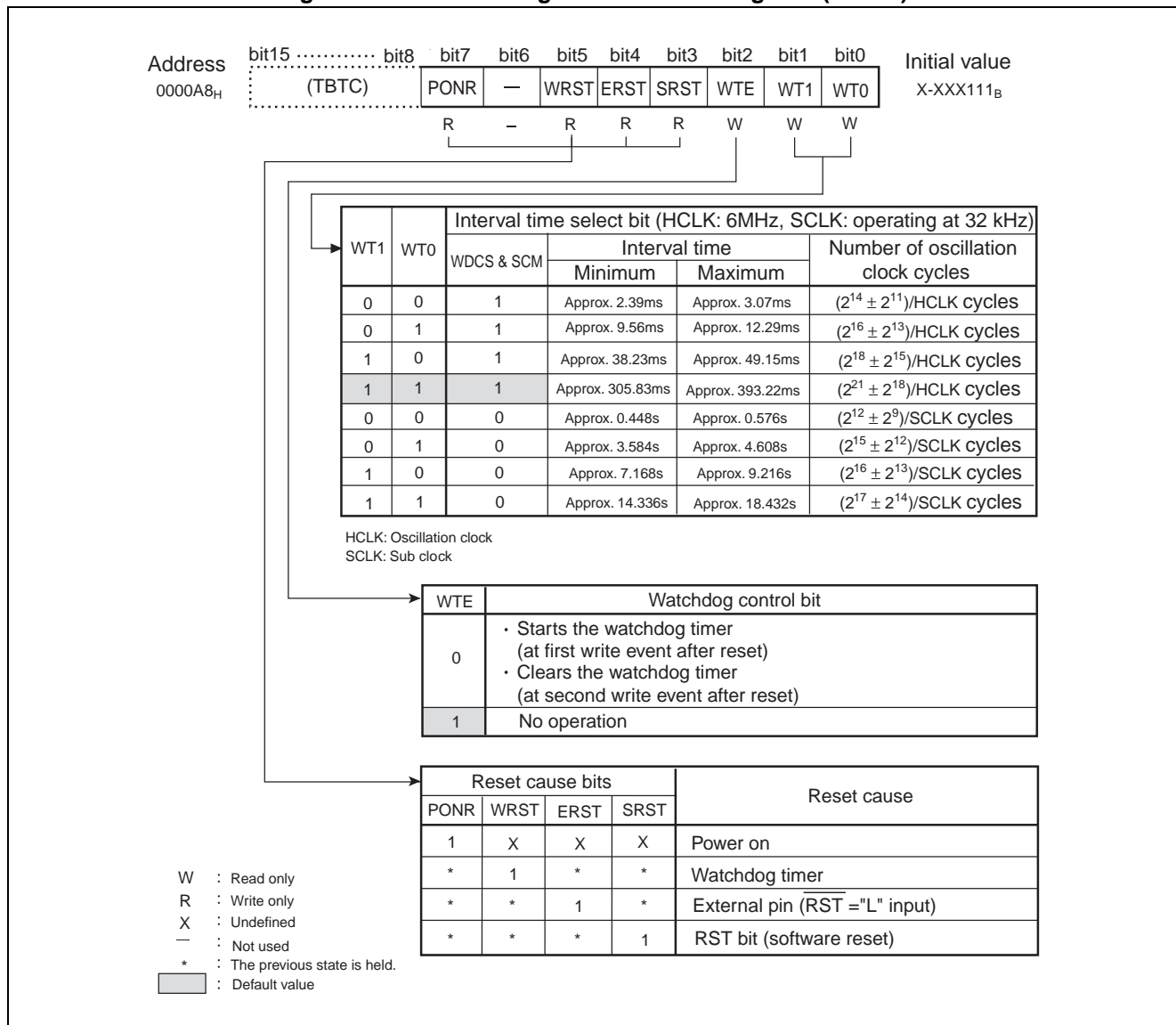
## 10.2 Watchdog Timer Control Register (WDTC)

The watchdog timer control register (WDTC) displays the activation, clearance, and reset factor of the watchdog timer.

### ■ Watchdog Timer Control Register (WDTC)

Figure 10.2-1 shows the watchdog timer control register (WDTC). Table 10.2-1 describes the function of each bit of the WDTC register.

Figure 10.2-1 Watchdog Timer Control Register (WDTC)



The interval time will be 3.5 to 4.5 times of the count clock (output value from the time-base timer) cycle. For details, see Section "10.4 Operations of Watchdog Timer".

**Table 10.2-1 Function of Each Bit of Watchdog Timer Control Register (WDTC)**

Bit name			Functions
bit 7, bit 5 to bit 3	PONR WRST ERST SRST	Reset factor bit	<ul style="list-style-type: none"> <li>Read-only bits that indicate reset factors. When a reset factor occurs, the relevant bit is set to "1".</li> <li>The PONR, WRST, ERST and SRST bits are all cleared to "0" after the WDTC register is read.</li> <li>The contents of the bits other than the PONR bit are not assured at power-on. Therefore, when the PONR bit is "1", ignore the contents of the bits other than the PONR bit.</li> </ul>
bit 6	Reserved	Reserved bit	<ul style="list-style-type: none"> <li>The reading value is irregular.</li> <li>Writing does not have the influence in the operation.</li> </ul>
bit 2	WTE	Watchdog control bit	<ul style="list-style-type: none"> <li>Writing "0", activates the watchdog timer (at the first write after reset) or clears the 2-bit counter (at the second write after reset).</li> <li>There is no influence in the operation in writing "1".</li> </ul>
bit 1, bit 0	WT1 WT0	Interval time select bit	<ul style="list-style-type: none"> <li>This is a bit to select the interval time of the watchdog timer.</li> <li>The interval timer when the sub clock mode is selected as the clock mode (the sub clock display bit (SCM) of clock select register (CKSCR) sets to "0") or when the watch timer is used as the clock source of watchdog timer (watchdog timer clock source select bit (WDSCS) sets to "0") is different from when the main clock mode or PLL clock mode is selected as the clock mode and the WDSCS bit of the WTC is set to "1" as shown in Figure 10.2-1 according to the setting of the watch timer control register (WTC).</li> <li>The data at the activation of the watchdog timer is valid. Data written after the activation of the watchdog timer is ignored.</li> <li>The WT1 and WT0 bit are only for writing.</li> </ul>

## 10.3 Configuration of Watchdog Timer

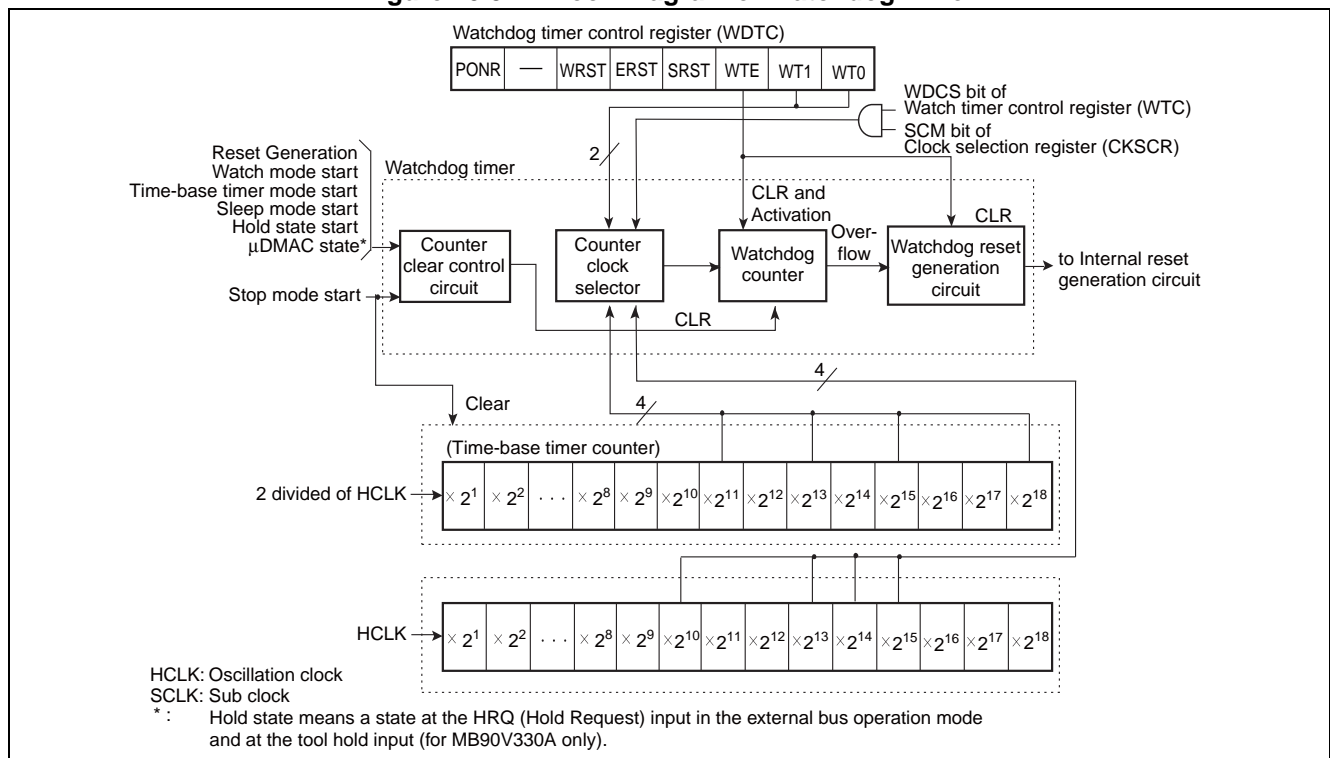
The watchdog timer consists of following five blocks.

- Count clock selector
- Watchdog counter (two bits counter)
- Watchdog reset generator circuit
- Counter clear control circuit
- Watchdog timer control register (WDTC)

### ■ Block Diagram of Watchdog Timer

Figure 10.3-1 shows a watchdog timer block diagram.

Figure 10.3-1 Block Diagram of Watchdog Timer



#### ● Count clock selector

Circuit that selects the count clock of the watchdog timer from four types of time-base timer output and four types of watch timer output. This determines the watchdog reset generation time.

#### ● Watchdog counter (two bits counter)

2-bit up-counter that uses time-base timer output as the count clock.

#### ● Watchdog reset generator circuit

Generates a reset signal by an overflow of the watchdog counter.

## ● Counter clear control circuit

Clears the watchdog counter and controls operation/stop of the counter.

**Table 10.3-1 Watchdog Timer Clearing Conditions**

Operation mode	Reset	WDTC register WTE=0	Stop mode	Sleep mode	Time-base timer mode	Watch mode	Hold	μDMAC
Clear	In transition	In writing	In transition	In transition	In transition	In transition	In transition	In starting
Watchdog timer state in modes	Disabled	-	Stopped (Keeps cleared)	Operating (Starts counting after clearing)	Operating (Starts counting after clearing)	Operating (Starts counting after clearing)	Stopped (Keeps cleared)	Stopped (Keeps cleared)
Watchdog reset in modes	Not occur	-	Not occur	Occur	Occur	Occur	Not occur	Not occur
Watchdog timer state after canceling and returning modes	Disabled	Operating	Operating (Restarts counting after clearing)	Operating (Continues counting)	Operating (Continues counting)	Operating (Continues counting)	Operating (Restarts counting after clearing)	Operating (Restarts counting after clearing)

## ● Watchdog timer control register (WDTC)

Activates/clears the watchdog timer and holds the reset occurrence factor.

## 10.4 Operations of Watchdog Timer

The watchdog timer generates a watchdog reset upon an overflow of the watchdog counter.

### ■ Operations of Watchdog Timer

Figure 10.4-1 shows the setting required to operate the watchdog timer.

**Figure 10.4-1 Setting of Watchdog Timer**

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0000A8 <sub>H</sub>	WDTC (TBTC)							PONR	—	WRST	ERST	SRST	WTE	WT1	WT0	
													0	⊙	⊙	

⊙ : Used bit  
 — : Unused bit  
 0 : Set to "0".

#### ● Activating watchdog timer

- The watchdog timer is activated at the first write of "0" to the watchdog control bit (WTE) of the watchdog timer control register (WDTC) after reset. In this case, specify the interval time at the same time by the interval time selection bits (WT1, WT0) of the WDTC register.
- Once the watchdog time is activated, it cannot stop until power-on or a watchdog reset occurs.

#### ● Clearing watchdog timer

- The 2-bit counter of the watchdog timer is cleared by the second write of "0" to the WTE bit. If the counter is not cleared within the interval time, the counter overflows and the watchdog is reset.
- The watchdog counter is cleared when a reset operation occurs or when a change to sleep mode, stop mode or time-base timer mode is made.
- When a change to time-base timer mode or watch mode is made, the watchdog counter is once cleared. However, be careful that the watchdog counter does not stop after being cleared.
- When watch mode is used (sub clock), do not use the watchdog timer.

#### ● Interval Time of Watchdog Timer

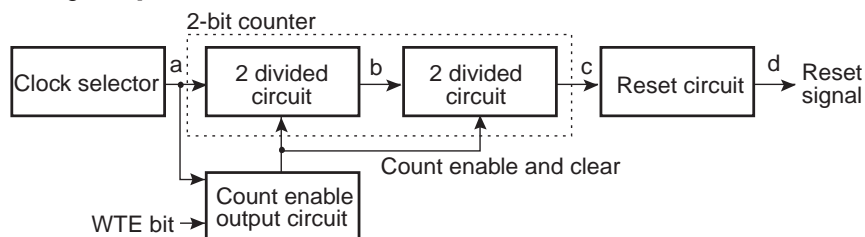
Figure 10.4-2 shows the relationship between watchdog timer clear timing and interval time. The interval time varies depending on the timing of clearing the watchdog timer and takes 3.5 to 4.5 times of the count clock cycle.

#### ● Checking reset factors

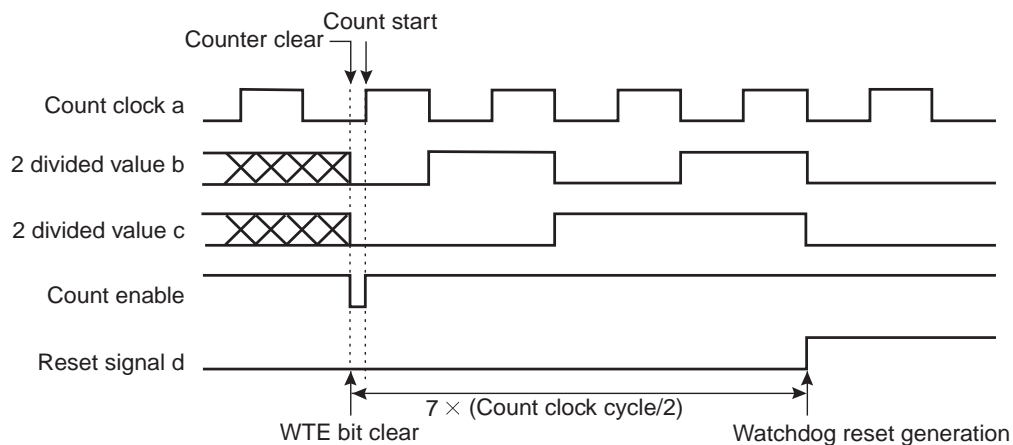
By checking the reset factor bit (PONR, WRST, ERST, SRST) of the WDTC register after reset, the factor of the reset can be identified.

Figure 10.4-2 Relationship between Clear Timing and Interval Time of Watchdog Timer

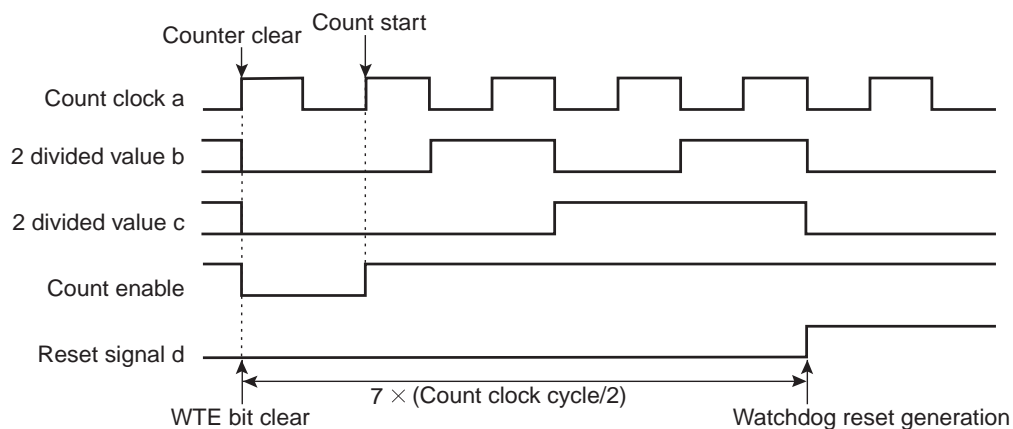
[Block diagram of Watchdog timer]



[Minimum interval time] When clear WTE bit immediately before rising of count clock



[Maximum interval time] When clear WTE bit immediately after rising of count clock





## **10.5 Precautions when Using Watchdog Timer**

---

**This section explains precautions when using watchdog timer.**

---

### **■ Precautions when Using Watchdog Timer**

- **Stopping watchdog timer**

Once the watchdog time is activated, it cannot stop until power-on or a watchdog-external reset occurs.

- **Interval Time**

Because the interval time uses carry-up signals from the time-base timer, as the count clock, clearing the time-base timer may make the interval time of the watch dog timer longer than the preset period of time.

- **Selecting Interval Time**

The interval time can be set when activating the watchdog timer. Data written after the activation of the watchdog timer is ignored.

- **Precautions when creating program**

When creating a program that clears the watchdog timer repeatedly in the main loop, the main loop processing time including the interrupt processing must not exceed the minimum interval time of the watchdog timer.

## 10.6 Program Examples of Watchdog Timer

Program example of watchdog timer is given below.

### ■ Program Examples of Watchdog Timer

#### ● Processing specification

- The watchdog timer is cleared each time in loop of the main program.
- The processing of the main loop must go round within the minimum interval time.

#### ● Coding example

```

WDTC      EQU      0000A8H          ; Watchdog timer control register
WTE       EQU      WDTC:2          ; Watchdog timer control bit
;-----Main program-----
CODE      CSEG
START:
;          :                      ; Initialize such as a stack pointer (SP).

WDG_START:
          MOV      WDTC, #00000011B ; Activating watchdog timer
                                           ;  $2^{21} \pm 2^{18}$  cycles in time of the interval are selected.
;-----Main loop-----
MAIN:     CLRB     I:WTE            ; Clearing watchdog timer
;          :                      ; It is regularly clearness of two bits.
;          User processing
;          :
          JMP      MAIN            ; Interval Time of Watchdog Timer
                                           ; Loop in the shortest possible time

CODE      ENDS
;-----Vector Settings-----
VECT      CSEG      ABS=0FFH
          ORG      0FFDCH          ; Reset vector setting
          DSL      START
          DB        00H            ; Single-chip mode setting
VECT      ENDS
          END      START

```



# ***CHAPTER 11***

---

# ***WATCH TIMER***

**This chapter describes a overview of the watch timer, functions and configurations of its registers, and its operation.**

- 11.1 Overview of Watch Timer
- 11.2 Configuration of Watch Timer
- 11.3 Watch Timer Control Register (WTC)
- 11.4 Operation of Watch Timer

## 11.1 Overview of Watch Timer

The watch timer is a 15-bit timer using the sub clock. It can generate interval interrupts. The watch timer can also be used as the clock source of the watchdog timer by setting so.

### ■ Functions of Watch Timer

The watch timer consists of a 15-bit timer and a circuit that controls interval interrupt.

The watch timer uses the sub clock, regardless of the values of the PLL clock selection bit (MCS) and sub clock selection bit (SCS) of the clock selection register (CKSCR).

The interval time of the watch timer are shown in Table 11.1-1.

**Table 11.1-1 Interval Times of Watch Timer**

WTC2	WTC1	WTC0	Interval Time*
0	0	0	31.25 ms
0	0	1	62.5 ms
0	1	0	125 ms
0	1	1	250 ms
1	0	0	500 ms
1	0	1	1.000 s
1	1	0	2.000 s
1	1	1	Setting disabled

\*: Four dividing of sub clock 32 kHz (=8 kHz)

## 11.2 Configuration of Watch Timer

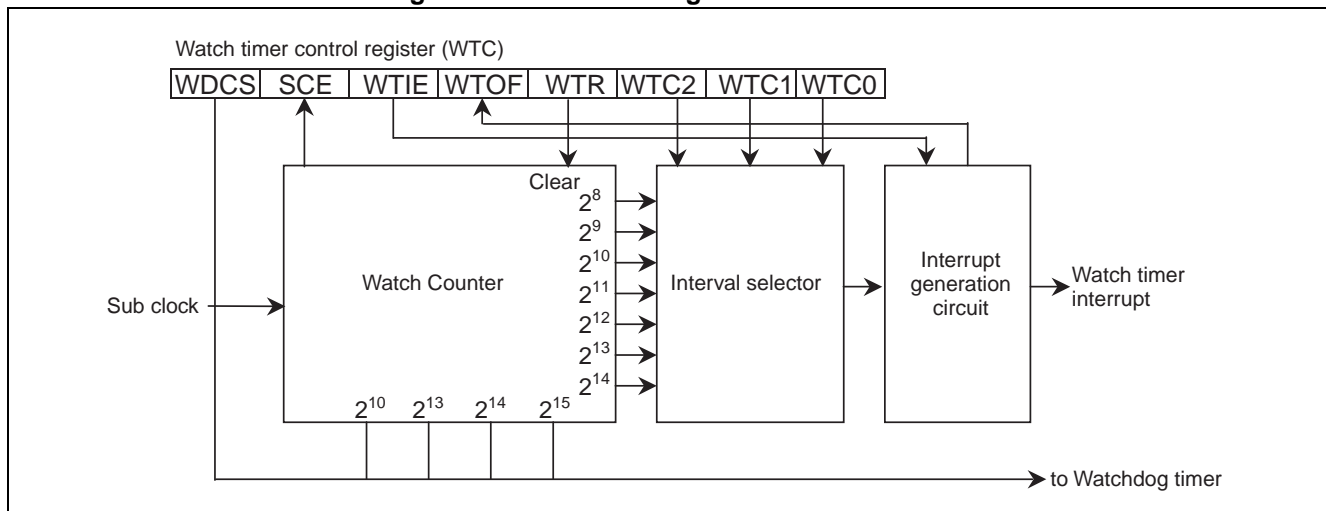
The watch timer is composed of the following four blocks.

- Interval selector
- Watch Counter
- Watch timer interruption generation circuit
- Watch timer control register (WTC)

### ■ Block Diagram of Watch Timer

Figure 11.2-1 shows the watch timer block diagram.

Figure 11.2-1 Block Diagram of Watch Timer



#### ● Clock Counter

A 15-bit up-counter using the sub clock as the clock source.

#### ● Interval selector

A selector that selects one from seven types of interval of the watch timer interrupt.

#### ● Interrupt generation circuit

Generates an interval interrupt of the watch timer.

#### ● Watch timer control register (WTC)

It specifies the operation of the watch timer, the control of the watch timer interrupt, and the clock source of the watchdog timer.

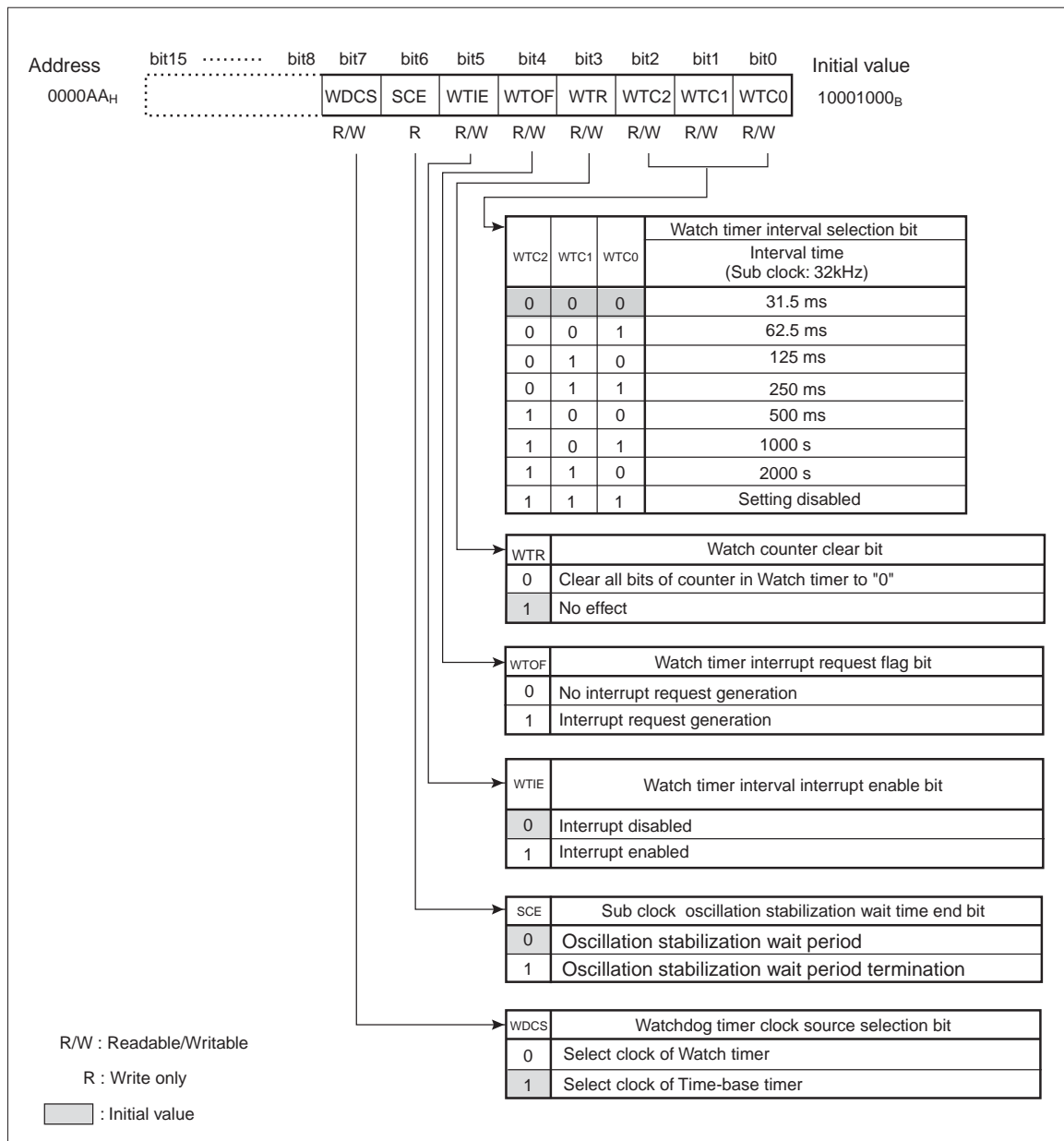
## 11.3 Watch Timer Control Register (WTC)

Watch timer control register (WTC) controls the operation of watch timer. It also controls the interval interrupt time.

### ■ Configuration of Watch Timer Control Register (WTC)

Figure 11.3-1 shows the watch timer control register (WTC) configuration. Table 11.3-1 summarizes the functions of each bit functions of watch timer control register (WTC).

Figure 11.3-1 Configuration of Watch Timer Control Register (WTC)



**Table 11.3-1 Each Bit Functions of Watch Timer Control Register (WTC)**

Bit name		Functions
bit 7	WDCS: Watchdog timer clock source selection bit	<ul style="list-style-type: none"> <li>• This bit selects the clock source of the watchdog timer.</li> <li>• Watch timer clock is selected if this bit is "0", and otherwise time-base timer clock is selected if this bit is "1".</li> <li>• Initialized to "1" by a reset.</li> </ul>
bit 6	SCE: Sub clock oscillation stabilization wait time end bit	<ul style="list-style-type: none"> <li>• It is a bit that indicates the oscillation stabilization wait time for the sub clock ends.</li> <li>• When this bit is "0", it indicates that the oscillation stabilization wait time is elapsing.</li> <li>• The oscillation stabilization wait time for the sub clock is fixed to <math>2^{14}</math> sub clock cycles.</li> <li>• Initialized to "0" upon power-on reset and upon stop.</li> </ul>
bit 5	WTIE: Watch timer interval interrupt enable bit	<ul style="list-style-type: none"> <li>• This bit enables an interval interrupt by the watch timer.</li> <li>• Interrupt is enabled if this bit is "1", and otherwise it is disabled if this bit is "1".</li> <li>• Initialized to "0" by a reset.</li> </ul>
bit 4	WTOF: Watch timer interrupt request flag bit	<ul style="list-style-type: none"> <li>• This bit indicates that an interrupt request of the watch timer has occurred.</li> <li>• When the WTIE bit is 1 and this bit is set to "1", an interrupt request is generated.</li> <li>• It is set to "1" at every interval set with the WTC2 to WTC0 bits.</li> <li>• It is cleared with "0" by writing "0" into it, shifting to stop mode, and reset operation.</li> <li>• Writing "1" does not have the meaning.</li> </ul>
bit 3	WTR: Watch counter clear bit	<ul style="list-style-type: none"> <li>• This bit clears all bits of the watch timer counter.</li> <li>• The watch timer counter is cleared to "0" by writing "0".</li> <li>• Writing "1" does not have the meaning.</li> <li>• The bit always returns "1" when read.</li> </ul>
bit 2 to bit 0	WTC2,WTC1,WTC0: Watch timer interval selection bit	<ul style="list-style-type: none"> <li>• It is a bit by which the interval of the watch timer is set.</li> <li>• Initialized to "000<sub>B</sub>" by reset.</li> <li>• To change the WTC2, WTC1, and WTC0 bits, clear the WTOF bit at the same time.</li> </ul>



## 11.4 Operation of Watch Timer

---

The watch timer functions as a clock source for the watchdog timer, a timer for sub clock oscillation stabilization delay time and an interval timer that generates an interrupt at regular intervals.

---

### ■ Watch Counter

The watch counter is a 15-bit counter that counts sub clock and continues counting while sub clock is input.

#### ● Clearing Watch Counter

The watch counter is cleared upon a power-on reset, change to stop mode and writing "0" to the watch counter clear bit (WTR) of the watch timer control register (WTC).

---

Notes:

- The watchdog timer and interval interrupts that use the output of the watch timer effect on the operation by clearing the watch counter.
  - To clear the watch timer by writing "0" to the WTR bit in the watch timer control register (WTC), set the WTIE bit to "0" and set the watch timer to interrupt inhibited state. Before permitting an interrupt, clear the interrupt request issued by writing "0" to the WTOF flag.
- 

### ■ Interruption Function of Interval of Watch Timer

A carry-up signal from the watch counter causes an interrupt at regular intervals.

#### ● Specification of Interval Time

The (WTC2, WTC1 and WTC0) bits of the WTC register specifies interval time.

#### ● Generation of Watch timer interrupt

The watch timer interrupt request flag bit (WTOF) is set for each interval time set by the WTC2, WTC1 and WTC0 bits. In this case, if interrupt is enabled by setting "1" in the watch timer interval interrupt enable bit (WTIE), a watch timer interrupt is generated.

The WTOF bit is set, based on the time the watch timer was cleared last.

Because, in stop mode, the watch timer functions as a timer for sub clock oscillation stabilization delay time, the WTOF bit is cleared immediately when mode is changed to stop mode.

### ■ Specification Function of Clock Source of Watchdog Timer

The clock source of the watchdog timer can be specified by the watchdog timer clock source selection bit (WDCS) of the WTC register. However, when clock mode is sub clock mode, the counter value of the watch timer is used, regardless of the value of the WDCS bit.

### ■ Sub Clock Oscillation Stabilization Delay Time Function

When recovering from power-on reset or stop mode, the watch timer functions as a timer for sub clock oscillation stabilization delay time. The sub clock oscillation stabilization delay time is fixed to  $2^{14}$  sub clock cycles.

# **CHAPTER 12**

---

## **16-BIT I/O TIMER**

**This chapter describes a overview of the 16-bit I/O timer, the functions and configurations of its registers, and its operation.**

- 12.1 Overview of 16-bit I/O Timer
- 12.2 Register of 16-bit I/O Timer
- 12.3 Operation of 16-bit I/O Timer

## 12.1 Overview of 16-bit I/O Timer

---

**The 16-bit I/O timer consists of one 16-bit free-run timer, and four output compare and four input capture modules. This function enables four independent waveforms to be output based on the 16-bit free-run timer, and input pulse widths and external clock cycle to be measured.**

---

### ■ Configuration Function of 16-bit I/O Timer

The functions of a 16-bit free-run timer, output compare, and input captures that make up the 16-bit I/O timer are as follows:

#### ● 16-bit free-run timer (x1)

The 16-bit free-run timer consists of a 16-bit up counter, control register, and prescaler. A value output from the timer counter is used as a basic time by the input capture and output compare. Clock for the counter operation can be selected from eight types ( $\phi$ : Machine clock).

- Eight types of internal clocks ( $\phi, \phi/2, \phi/4, \phi/8, \phi/16, \phi/32, \phi/64, \phi/128$ )
- Either the internal or external clock (FRCK) for the basic machine clock can be selected.

An interrupt can be generated through the overflow of the counter value and comparing match with the compare clear register (Mode setting is required for comparing match).

The count value can be initialized to "0000<sub>H</sub>" with reset, software clear, and compare match operations with compare clear register.

#### ● Output compare (x4)

The output compare consists of four 16-bit compare registers, compare output latch, and control registers. It can reverse the output level and at the same time generate an interrupt when the 16-bit free-run timer value matches that of the compare registers.

- Four compare registers are operated independently of each other. There are an output pin and interrupt flag that correspond to each compare register.
- The output pin can be controlled by pairing two compare registers.
- The initial value of the output pin can be set.
- An interrupt can be generated by a comparing match.

#### ● Input capture (x4)

The input capture consists of four independent external input pins and associated capture and control registers. It can detect an arbitrary edge of the signal input from external input pin to generate an interrupt while holding the 16-bit free-run timer value in the capture register.

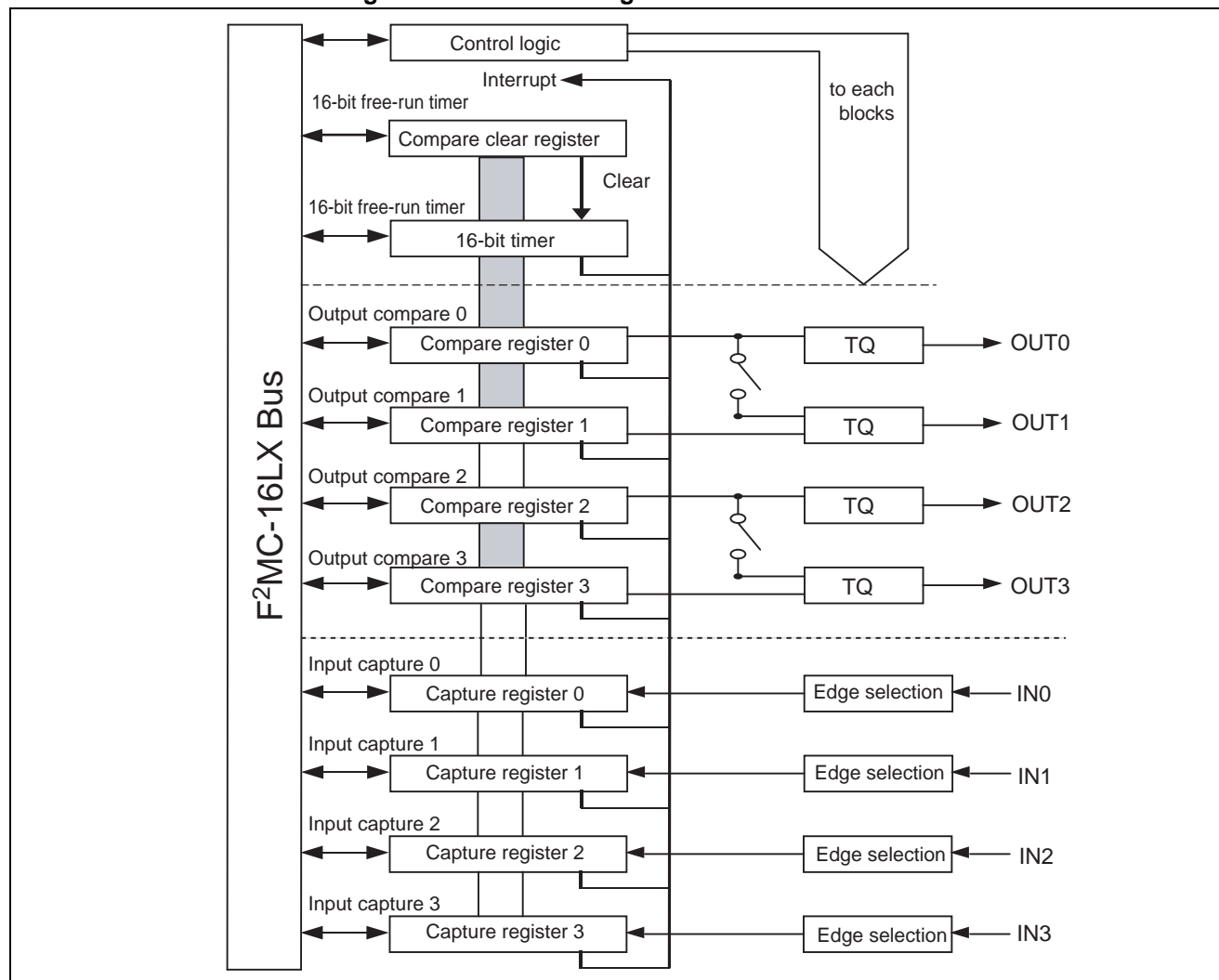
- The edge of an external input signal is optional. The external input signal edge can be selected from a rising edge, falling edge, or both edges.
- Four input capture can operate independently.

Interrupts can be generated through valid edges of external input signals. The input capture can activate  $\mu$ DMAC via interrupt.

## ■ Block Diagram

Figure 12.1-1 shows the 16-bit I/O timer block diagram.

**Figure 12.1-1 Block Diagram of 16-bit I/O Timer**



## 12.2 Register of 16-bit I/O Timer

Registers for the 16-bit I/O timer are classified into the following general categories:

- 16-bit free-run timer
- 16-bit output compare
- 16 bit input capture

The configuration and functions of register are described.

### ■ Register Configuration of 16-bit I/O Timer

The register configuration of the 16-bit I/O timer is shown in the following.

#### ● 16-bit free-run timer

Figure 12.2-1 16-bit Free-run Timer Configuration

00008BH,00008AH	bit 15 CPCLR 0	Compare clear register
000087H,000086H	TCDT	Timer counter data register
000089H,000088H	TCCS	Timer control status register

#### ● 16-bit Output compare

Figure 12.2-2 16-bit Output Compare Configuration

007919H,007918H	bit 15 OCCP0 0	Output compare register
00791BH,00791AH	OCCP1	
00791DH,00791CH	OCCP2	
00791FH,00791EH	OCCP3	
000055H,000054H	OCS1	Output compare control register
	OCS0	
000057H,000056H	OCS3	
	OCS2	

#### ● 16 bit input capture

Figure 12.2-3 16 Bit Input Capture Configuration

007911H,007910H	bit 15 IPCP0 0	Input capture data register
007913H,007912H	IPCP1	
007915H,007914H	IPCP2	
007917H,007916H	IPCP3	
000053H,000052H	ICS23	Input capture control status register
	ICS01	

## MB90330A Series

### 12.2.1 16-bit Free-run Timer

The 16-bit free-run timer consists of a 16-bit up-down counter and control status register.

A value of the timer counter is used as a basic time of the input capture and output compare (base timer).

- Clock for the counter operation can be selected from eight types.
- The counter overflow interruption can be generated.
- Setting the mode enables initialization of the counter through match operation with the value of the compare clear register in the output compare.

#### ■ Register List of 16-bit Free-run Timer

Figure 12.2-4 lists the 16-bit free-run timer registers.

Figure 12.2-4 Register List of 16-bit Free-run Timer

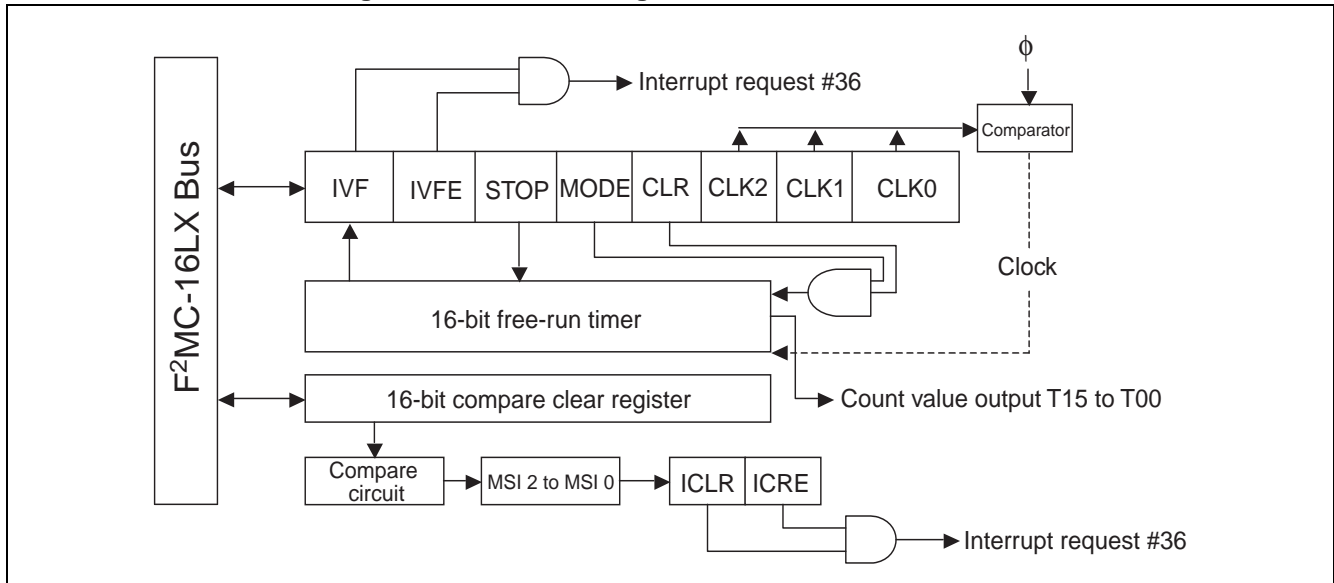
bit	15	14	13	12	11	10	9	8	CPCLR
00008B <sub>H</sub>	CL15	CL14	CL13	CL12	CL11	CL10	CL09	CL08	Compare clear register upper
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value XXXXXXXX <sub>B</sub>
bit	7	6	5	4	3	2	1	0	CPCLR
00008A <sub>H</sub>	CL07	CL06	CL05	CL04	CL03	CL02	CL01	CL00	Compare clear register lower
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value XXXXXXXX <sub>B</sub>
bit	15	14	13	12	11	10	9	8	TCDT
000087 <sub>H</sub>	T15	T14	T13	T12	T11	T10	T09	T08	Timer counter data register upper
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 00000000 <sub>B</sub>
bit	7	6	5	4	3	2	1	0	TCDT
000086 <sub>H</sub>	T07	T06	T05	T04	T03	T02	T01	T00	Timer counter data register lower
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 00000000 <sub>B</sub>
bit	15	14	13	12	11	10	9	8	TCCS
000089 <sub>H</sub>	ECKE	—	—	MSI2	MSI1	MSI0	ICLR	ICRE	Timer control status register upper
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 0XX00000 <sub>B</sub>
bit	7	6	5	4	3	2	1	0	TCCS
000088 <sub>H</sub>	IVF	IVFE	STOP	MODE	CLR	CLK2	CLK1	CLK0	Timer control status register lower
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 00000000 <sub>B</sub>

R/W :Readable/Writable

## ■ Block Diagram of 16-bit Free-run Timer

Figure 12.2-5 shows the 16-bit free-run timer block diagram.

Figure 12.2-5 Block Diagram of 16-bit Free-run Timer



## ■ Compare Clear Register (CPCLR)

Figure 12.2-6 shows the bit configuration of the compare clear register (CPCLR).

Figure 12.2-6 Bit Configuration of Compare Clear Register (CPCLR)

bit	15	14	13	12	11	10	9	8	CPCLR
00008B <sub>H</sub>	CL15	CL14	CL13	CL12	CL11	CL10	CL09	CL08	Compare clear register upper
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value XXXXXXXX <sub>B</sub>
bit	7	6	5	4	3	2	1	0	CPCLR
00008A <sub>H</sub>	CL07	CL06	CL05	CL04	CL03	CL02	CL01	CL00	Compare clear register lower
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value XXXXXXXX <sub>B</sub>

R/W :Readable/Writable

The compare clear register (CPCLR) is a 16-bit length compare register that is compared with the 16-bit free-run timer. Since the CPCLR register has an indefinite, initial value, you must set the value before enabling the operation. Moreover, please access the CPCLR register the word.

When the MODE bit of the timer control status register (TCCS) is set to "1", 16-bit free-run timer value is cleared to "0000<sub>H</sub>" at matching between the register value and 16-bit free-run timer value. And when the register value is matched with 16-bit free-run timer value, 16-bit free-run timer value is initialized to "0000<sub>H</sub>", and the compare clear interrupt flag is set. When the interrupt is enabled on the compare clear interrupt flag = 1, the interrupt request is generated to CPU.

## ■ Timer Counter Data Register (TCDT)

Figure 12.2-7 shows the bit configuration of the timer counter data register (TCDT).

**Figure 12.2-7 Bit Configuration of Timer Counter Data Register (TCDT)**

bit	15	14	13	12	11	10	9	8	TCDT
000087 <sub>H</sub>	T15	T14	T13	T12	T11	T10	T09	T08	Timer counter data register upper
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 00000000 <sub>B</sub>
bit	7	6	5	4	3	2	1	0	TCDT
000086 <sub>H</sub>	T07	T06	T05	T04	T03	T02	T01	T00	Timer counter data register lower
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 00000000 <sub>B</sub>

R/W :Readable/Writable

The timer counter data register (TCDT) is a register that can read a count value of the 16-bit free-run timer. A reset operation clears the count value with "0000<sub>H</sub>". A timer value can be set by writing the value into the TCDT register, but ensure to perform at the stop state (STOP=1) before doing so.

Please access the TCDC register the word. The 16-bit free-run timer is initialized with the following causes:

- Initialization by Reset
- Initialization by clear bit (CLR) of control status
- Initialization by match operation between the compare clear register and the timer counter value (setting the mode is required)

## ■ Timer Control Status Register (TCCS)

Figure 12.2-8 shows the bit configuration of the timer control status register (TCCS).

**Figure 12.2-8 Bit Configuration of Timer Control Status Register (TCCS)**

bit	15	14	13	12	11	10	9	8	TCCS
000089 <sub>H</sub>	ECKE	—	—	MSI2	MSI1	MSI0	ICLR	ICRE	Timer control status register upper
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 0XX00000 <sub>B</sub>
bit	7	6	5	4	3	2	1	0	TCCS
000088 <sub>H</sub>	IVF	IVFE	STOP	MODE	CLR	CLK2	CLK1	CLK0	Timer control status register lower
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 00000000 <sub>B</sub>

R/W :Readable/Writable

The function of each bit in the timer control status register (TCCS) is described in the following.

### [bit 15] ECKE (External clock input enable bit)

This bit enables selection of the count clock source of the 16-bit free-run timer from the internal or external source. Since the clock is changed immediately after writing into the ECKE bit, you must change it when the output compare and input capture are in stopping state.

0	The internal clock source is selected	[Initial value]
1	The clock is input from external pin (FRCK).	

### [bit 14, bit 13] Undefined bit

The reading value is irregular. No effect on writing.



**[bit 12 to bit 10] MSI2, MSI1, MSI0 (Interrupt mask selection bit)**

Bit to set the number of times the compare clear interrupt be masked. It consists of a 3-bit reload counter and the count value is reloaded every time the counter value is "000<sub>B</sub>". In addition, when writing into the MSI2, MSI1, and MSI0 registers, the count value is also loaded. The number of masks is the number of settings. (Example: Set "010<sub>B</sub>" if masked twice, interrupted at the third time) However, setting "000<sub>B</sub>" cannot mask interrupt causes.

**[bit 9] ICLR (Compare clear interrupt flag bit)**

It is the interrupt request flag of compare clear. When the compare clear register value matches with the 16-bit free-run timer value by compare operation, this bit is set to "1". When the interrupt request enable bit (ICRE of bit 8) is set, an interrupt is generated. The ICLR bit is cleared by writing "0". Writing "1" does not have the meaning. By the read-modify-write instruction, this bit is always read "1".

0	No Interrupt request	[Initial value]
1	Interrupt request	

**[bit 8] ICRE (Compare clear interrupt request enable bit)**

It is the interrupt enable bits of compare clear. When the bit is "1" and the interrupt flag (ICLR of bit 9) is set to "1", an interrupt is generated.

0	Interrupt disabled	[Initial value]
1	Interruption permission	

**[bit 7] IVF (Time overflow generation flag bit)**

It is the interrupt request flag of 16-bit free-run timer.

If the 16-bit free-run timer overflows, or the counter is cleared when it matches the compare clear register through compare operation as a result the mode setting, the IVF bit is set to "1". When the interrupt request enable bit (IVFE of bit 5) is set, an interrupt is generated. Writing "0" into it clears it. Writing "1" does not have the meaning. The read-modify-write instructions always read "1" from this bit.

0	No Interrupt request [Initial value]
1	Interrupt request

**[bit 6] IVFE (Time overflow interrupt enable bit)**

It is Interrupt enable bit of 16-bit free-run timer. When the bit is "1" and the write flag (IVF of bit 5) is set to "1", an interrupt is generated.

0	Interrupt disabled	[Initial value]
1	Interruption permission	

**[bit 5] STOP (Timer operation stop bit)**

It is bit for stopping count of 16-bit free-run timer. Writing "1" into this bit stops counting the 16-bit free-run timer and writing "0" starts counting it.

0	Count permission (operation) [Initial value]
1	Count disabled (Stop)

Note that the output compare operation will stop when the 16-bit free-run timer stops counting.

**[bit 4] MODE (Clear condition selection bit)**

This bit specifies the initialization conditions of the 16-bit free-run timer.

If this bit is "0", the value of the count can be initialized with reset operation and clear bit ((CLR) of bit 2).

If this bit is "1", the value of the count can be initialized with reset operation and clear bit ((CLR) of bit 2), and through matching the value of the compare clear register.

0	Initialization by reset and clear bit [Initial value]
1	Initialization by a reset, clear bit or the compare clear register.

Further, note that the value of the counter is initialized on the point of change in the count value.

**[bit 3] CLR (Timer clear bit)**

This bit initializes the value of the 16-bit free-run timer to "0000<sub>H</sub>" while it is operating.

Writing "1" into the bit initializes the value of the counter to "0000<sub>H</sub>". Writing "0" into the bit has no meaning. This bit is always "0" at the beginning of reading. The value of the counter is initialized on the point of change in the counter value.

0	No meaning. [Initial value]
1	The counter value is initialized to "0000 <sub>H</sub> "

Note that you must write "0000<sub>H</sub>" into the data register if you want to initialize the counter value while the timer is stopped.

---

**Note:**

If "0" is written to this bit after writing "1" and until next count clock, counter value is not initialized.

---

**[bit 2 to bit 0] CLK2,CLK1,CLK0 (Count clock cycle selection bit)**

Select count clock of 16-bit free-run timer. Since the clock is changed immediately after writing into the CLK2, CLK1, and CLK0 bits you must change it when the output compare and input capture are in stopping state.

CLK2	CLK1	CLK0	Count Clock	$\phi=24$ MHz	$\phi=12$ MHz	$\phi=6$ MHz	$\phi=3$ MHz
0	0	0	$\phi$	41.7 ns	83.3 ns	0.17 $\mu$ s	0.33 $\mu$ s
0	0	1	$\phi/2$	83.3 ns	0.17 $\mu$ s	0.33 $\mu$ s	0.67 $\mu$ s
0	1	0	$\phi/4$	0.17 $\mu$ s	0.33 $\mu$ s	0.67 $\mu$ s	1.33 $\mu$ s
0	1	1	$\phi/8$	0.33 $\mu$ s	0.67 $\mu$ s	1.33 $\mu$ s	2.67 $\mu$ s
1	0	0	$\phi/16$	0.67 $\mu$ s	1.33 $\mu$ s	2.67 $\mu$ s	5.33 $\mu$ s
1	0	1	$\phi/32$	1.33 $\mu$ s	2.67 $\mu$ s	5.33 $\mu$ s	10.7 $\mu$ s
1	1	0	$\phi/64$	2.67 $\mu$ s	5.33 $\mu$ s	10.7 $\mu$ s	21.3 $\mu$ s
1	1	1	$\phi/128$	5.33 $\mu$ s	10.7 $\mu$ s	21.3 $\mu$ s	42.7 $\mu$ s

## MB90330A Series

### 12.2.2 Output Compare

The output compare consists of 16-bit compare registers, compare output pin part, and a control register. It can reverse the output level for the pin and, at the same time, generate an interrupt when the 16-bit free-run timer value matches a value set in the 16-bit compare registers.

- It has a total of four compare registers that can operate independently.  
In addition, the output can be set to be controlled by using two compare registers.
- Interrupt can be set through compare-match operation.

#### ■ List of Output Compare Registers

Figure 12.2-9 lists the output compare registers.

Figure 12.2-9 List of Output Compare Registers

		bit 15 14 13 12 11 10 9 8								OCCP0 to OCCP3
ch.0:007919H		C15	C14	C13	C12	C11	C10	C09	C08	Output compare register upper
ch.1:00791BH		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value XXXXXXXXB
ch.2:00791DH										
ch.3:00791FH										
		bit 7 6 5 4 3 2 1 0								OCCP0 to OCCP3
ch.0:007918H		C07	C06	C05	C04	C03	C02	C01	C00	Output compare register lower
ch.1:00791AH		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value XXXXXXXXB
ch.2:00791CH										
ch.3:00791EH										
		bit 15 14 13 12 11 10 9 8								OCS1/OCS3
ch.1:000055H		—	—	—	CMOD	OTE1	OTE0	OTD1	OTD0	Output compare control register upper
ch.3:000057H		-	-	-	R/W	R/W	R/W	R/W	R/W	Initial value ---00000B
		bit 7 6 5 4 3 2 1 0								OCS0/OCS2
ch.0:000054H		ICP1	ICP0	ICE1	ICE0	—	—	CST1	CST0	Output compare control register lower
ch.2:000056H		R/W	R/W	R/W	R/W	-	-	R/W	R/W	Initial value 0000--00B

R/W :Readable/Writable

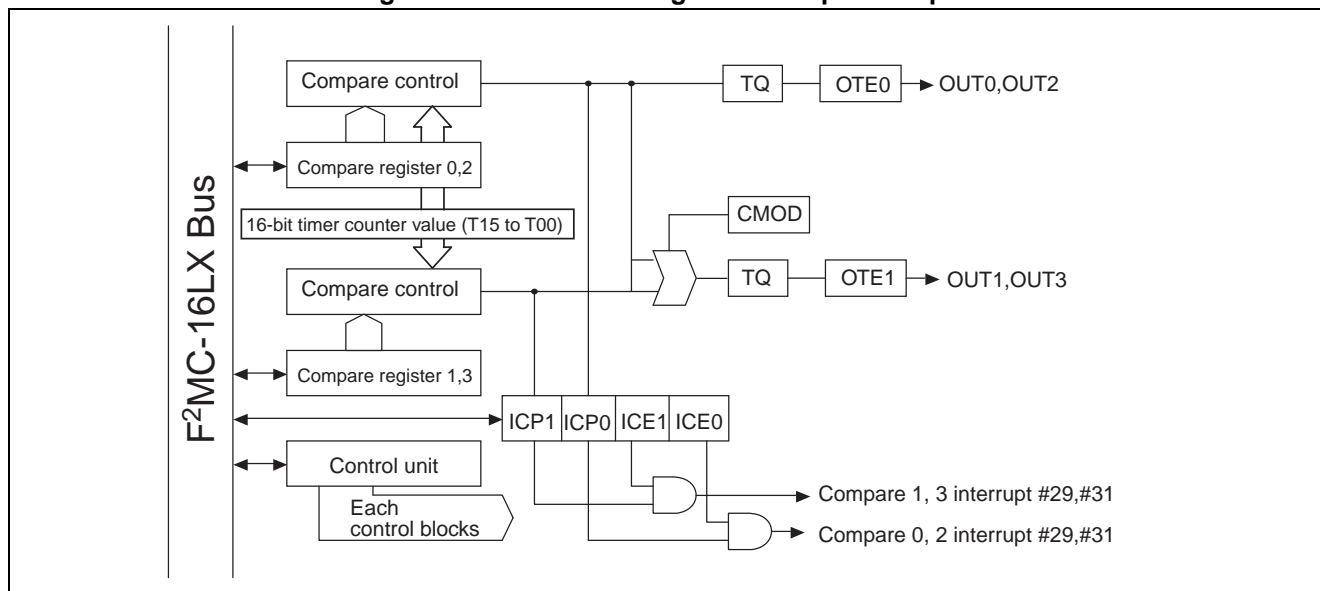
#### Note:

If you rewrite compare registers, you must rewrite them in a compare interrupt routine or in the state of disabled compare operation to ensure that compare-match operation and write operation never occur at the same time.

## ■ Block Diagram of Output Compare

Figure 12.2-10 shows the output compare block diagram.

Figure 12.2-10 Block Diagram of Output Compare



## ■ Output Compare Registers (OCCP0 to OCCP3)

Figure 12.2-11 shows the bit configurations of the output compare registers (OCCP0 to OCCP3).

Figure 12.2-11 Bit Configurations of Output Compare Registers (OCCP0 to OCCP3)

bit	15	14	13	12	11	10	9	8	OCCP0 to OCCP3
ch.0:007919H	C15	C14	C13	C12	C11	C10	C09	C08	Output compare register upper
ch.1:00791BH	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value XXXXXXXXB
ch.2:00791DH									
ch.3:00791FH									
bit	7	6	5	4	3	2	1	0	OCCP0 to OCCP3
ch.0:007918H	C07	C06	C05	C04	C03	C02	C01	C00	Output compare register lower
ch.1:00791AH	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value XXXXXXXXB
ch.2:00791CH									
ch.3:00791EH									

R/W :Readable/Writable

The output compare registers (OCCP0 to OCCP3) are 16-bit length compare registers any of which is compared with the 16-bit free-run timer. Since the registers have indefinite, initial values, set the initial values before enabling them. Please access the OCCP0 to OCCP3 register the word. When the values of the OCCP0 to OCCP3 registers match that of the 16-bit free-run timer, the compare signal is generated and the output compare interrupt flag is set. In addition, if output is enabled, the output level corresponding to each of the compare registers is reversed.

## ■ Output Compare Control Registers (OCS0 to OCS3)

Figure 12.2-12 shows the bit configurations of the output compare control registers (OCS0 to OCS3).

**Figure 12.2-12 Bit Configurations of Output Compare Control Registers (OCS0 to OCS3)**

								OCS1/OCS3	
								Output compare	
								control register upper	
ch.1:000055H	bit 15	14	13	12	11	10	9	8	
ch.3:000057H	—	—	—	CMOD	OTE1	OTE0	OTD1	OTD0	
	-	-	-	R/W	R/W	R/W	R/W	R/W	Initial value ---00000B
								OCS0/OCS2	
								Output compare	
								control register lower	
ch.0:000054H	bit 7	6	5	4	3	2	1	0	
ch.2:000056H	ICP1	ICP0	ICE1	ICE0	—	—	CST1	CST0	
	R/W	R/W	R/W	R/W	-	-	R/W	R/W	Initial value 0000--00B
R/W :Readable/Writable									

The function of each bit in the output compare control registers (OCS0 to OCS3) is described in the following.

### [bit 15 to bit 13] Undefined bit

The reading value is irregular. No effect on writing.

### [bit 12] CMOD (Output level inverse mode bit)

If pin output is enabled (OTE1=1 or OTE0=1), pin output level inverse operation mode is switched when there is a comparing match.

- When CMOD=0 (initial value), the pin output level that correspond to the compare register is reversed.
  - OUT0/OUT2: Reverses the level at a match with the compare register 0/2.
  - OUT1/OUT3: Reverses the level at a match with the compare register 1/3.
- When CMOD=1, compare register 0/2 reverses the output level the same way as when CMOD=0, and the output level of the pin (OUT1/OUT3) that corresponds to compare register 1/3 is reversed when there are comparing matches with both compare register 0/2 and compare register 1/3. If compare register 0/2 and compare register 1/3 have the same value, the operation would be the same as if they were one compare register.
  - OUT0/OUT2: Reverses the level at a match with the compare register 0/2.
  - OUT1/OUT3: Reverses the level at a match with the compare register 1/3 or 0/2.

### [bit 11, bit 10] OTE1,OTE0 (Output enable bit)

Bit to enable the pin output of the output compare. The initial value is "0".

0	It operates as a general-purpose port. [Initial value]
1	Becomes the output compare pin output.

- OTE1: Corresponding to output compare 1/3
- OTE0: Corresponding to output compare 0/2

**[bit 9, bit 8] OTD1,OTD0 (Output level bit)**

It is used when you want to change the pin output level if the pin output of the output compare is enabled. The initial value of the compare pin output is "0". To perform a write operation, first, stop the compare operation. When you perform a read operation, the output value of the output compare pin is read out.

0	Sets the compare pin output to "0". [Initial value]
1	Sets the compare pin output to "1".

- OTD1: Corresponding to output compare 1/3
- OTD0: Corresponding to output compare 0/2

**[bit 7, bit 6] ICP1,ICP0 (Compare match interrupt flag bit)**

It is output compare interrupt flag. When the values of the compare registers match that of the 16-bit free-run timer, it is set to "1". If the interrupt request bits (ICE1 and ICE0) are enabled, an output compare interrupt is generated when ICP1 and ICP0 bits are set. It is cleared by writing "0" and writing "1" has no meaning. In the read modify system, "1" is read.

0	No Compare match[Initial value]
1	Compare match

- ICP1: Corresponding to output compare 1/3
- ICP0: Corresponding to output compare 0/2

**[bit 5, bit 4] ICE1,ICE0 (Compare match interrupt enable bit)**

It is the interrupt enable bit for the output compare. If the ICE1 and ICE0 bits are "1", an output compare interrupt is generated when the interrupt flags (ICP1 and ICP0) are set.

0	Output compare interruption interdiction [Initial value]
1	Output compare interruption permission.

- ICE1: Corresponding to output compare 1/3
- ICE0: Corresponding to output compare 0/2

**[bit 3, bit 2] Undefined bit**

The reading value is irregular. No effect on writing.

**[bit 1, bit 0] CST1,CST0 (Compare operation enable bit)**

These bits permit a matching operation with the 16-bit free-run timer.

0	Compare operation disabled [Initial value]
1	Compare Operation enabled

- CST1: Corresponding to output compare 1/3
- CST0: Corresponding to output compare 0/2

Set the compare register values before enabling a compare operation.

---

Note:

Since the output compare is synchronous with the clock of the 16-bit free-run timer, stopping the timer means the stop of the compare operation.

---



## 12.2.3 Input Capture

This module has a function that detects a rising edge, falling edge, or both edges of externally input signal and holds a value of the 16-bit free-run timer in a register at the time of detection. It can also generate an interrupt when detecting an edge.

### ■ Input Capture

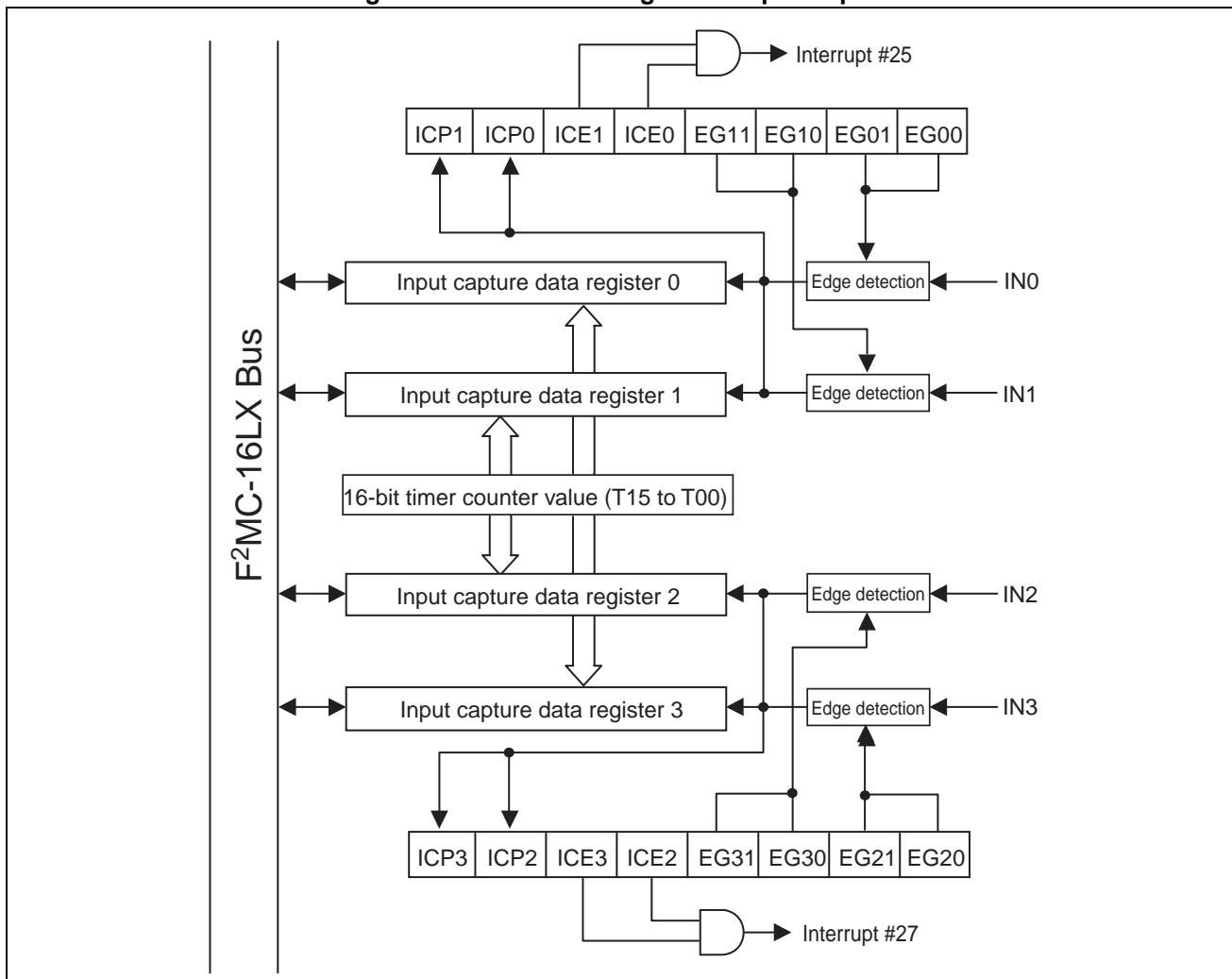
The input capture consist of input capture and control registers. Each input capture has its corresponding external input pin.

- The valid edge of the external input can be selected from three types: Rising edge/falling edge/both edges.
- It can generate an interrupt when it detects the valid edge of the external input.

### ■ Block Diagram of Input Capture

Figure 12.2-13 shows the input capture block diagram.

Figure 12.2-13 Block Diagram of Input Capture



## ■ List of Register of Input Capture

Figure 12.2-14 lists input capture registers.

**Figure 12.2-14 List of Register of Input Capture**

ch.0:007911H	bit 15	14	13	12	11	10	9	8	IPCP0 to IPCP3
ch.1:007913H	CP15	CP14	CP13	CP12	CP11	CP10	CP09	CP08	Input capture data register upper
ch.2:007915H	R	R	R	R	R	R	R	R	Initial value XXXXXXXXB
ch.3:007917H									
ch.0:007910H	bit 7	6	5	4	3	2	1	0	IPCP0 to IPCP3
ch.1:007912H	CP07	CP06	CP05	CP04	CP03	CP02	CP01	CP00	Input capture data register lower
ch.2:007914H	R	R	R	R	R	R	R	R	Initial value XXXXXXXXB
ch.3:007916H									
000053H	bit 15	14	13	12	11	10	9	8	ICS23
	ICP3	ICP2	ICE3	ICE2	EG31	EG30	EG21	EG20	Input capture control status register 23
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 00000000B
000052H	bit 7	6	5	4	3	2	1	0	ICS01
	ICP1	ICP0	ICE1	ICE0	EG11	EG10	EG01	EG00	Input capture control status register 01
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value 00000000B

R/W :Readable/Writable  
R: Read Only

## ■ Input Capture Data Registers (IPCP0 to IPCP3)

Figure 12.2-15 shows the bit configurations of the input capture data registers (IPCP0 to IPCP3).

**Figure 12.2-15 Bit Configurations of Input Capture Data Register (IPCP0 to IPCP3)**

ch.0:007911H	bit 15	14	13	12	11	10	9	8	IPCP0 to IPCP3
ch.1:007913H	CP15	CP14	CP13	CP12	CP11	CP10	CP09	CP08	Input capture data register upper
ch.2:007915H	R	R	R	R	R	R	R	R	Initial value XXXXXXXXB
ch.3:007917H									
ch.0:007910H	bit 7	6	5	4	3	2	1	0	IPCP0 to IPCP3
ch.1:007912H	CP07	CP06	CP05	CP04	CP03	CP02	CP01	CP00	Input capture data register lower
ch.2:007914H	R	R	R	R	R	R	R	R	Initial value XXXXXXXXB
ch.3:007916H									

R :Read Only

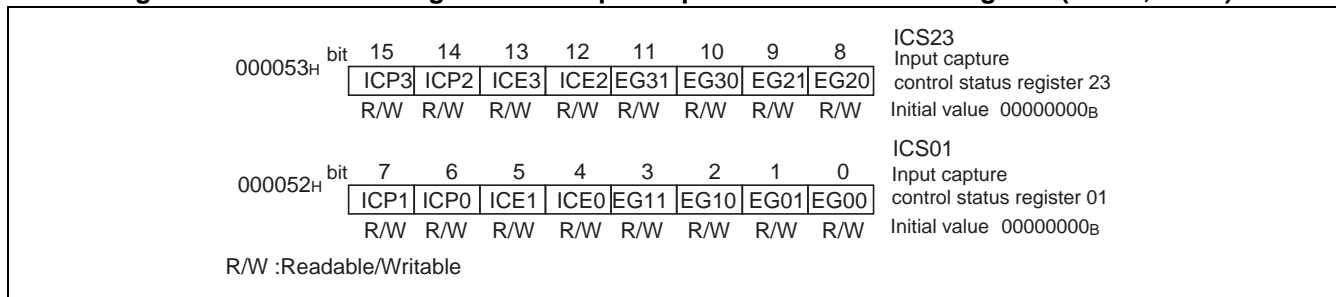
The input capture data registers (IPCP0 to IPCP3) are registers that hold a value of the 16-bit free-run timer when the valid edges of waveforms input from their corresponding external pins are detected.

Please access the IPCP0 to IPCP3 register the word. The IPCP0 to IPCP3 registers are not permitted to write.

## ■ Control Status Registers (ICS01, ICS23)

Figure 12.2-16 shows the bit configurations of the control status registers (ICS01 and ICS23).

**Figure 12.2-16 The Configuration of Input Capture Control Status Register (ICS01, ICS23)**



The function of each bit in the input capture control status registers (ICS01, ICS23) is described in the following.

### [bit 15, bit 14, bit 7, bit 6] ICP3, ICP2, ICP1, ICP0 (Valid edge detection flag bit)

It is input capture interrupt flag. The input capture sets the ICP3 to ICP0 bits to "1" when it detects valid edges of external input pins. It can generate an interrupt by detecting a valid edge when any of the interrupt enable bits (ICE3, ICE2, ICE1, and ICE0) is set.

Writing "0" into it clears it. Writing "1" does not have the meaning. The read-modify-write instructions always read "1" from this bit.

0	There is no effective edge detection	[Initial value]
1	There is effective edge detection.	

ICP<sub>n</sub>: a number denoted by n corresponds to the channel number of the input capture.

### [bit 13, bit 12, bit 5, bit 4] ICE3, ICE2, ICE1, ICE0 (Capture interrupt enable bit)

It is input capture interrupt enable bit. An input capture interrupt is generated when any of ICE3 to ICE0 bits is "1" and its corresponding interrupt flag (ICP3 to ICP0) is also set.

0	Interrupt disabled	[Initial value]
1	Interruption permission	

ICE<sub>n</sub>: a number denoted by n corresponds to the channel number of the input capture.

### [bit 11 to bit 8, bit 3 to bit 0] EG31, EG30, EG21, EG20, EG11, EG10, EG01, EG00 (Edge selection bit)

It specifies the polarity of a valid edge of the external input. The input capture operation permission is used combinedly.

EG <sub>n</sub> 1	EG <sub>n</sub> 0	Edge detection polarity
0	0	No edge detection (stopped state) [Initial value]
0	1	Rising edge detection
1	0	Falling edge detection
1	1	Both edges detection

EG<sub>n</sub>1/EG<sub>n</sub>0: a number denoted by n corresponds to the channel number of the input capture.

## 12.3 Operation of 16-bit I/O Timer

---

The operation and timing of the 16-bit I/O timer are described.

---

### ■ Operation and Timing of 16-bit I/O Timer

The following items related to the operation and timing of the 16-bit I/O timer are described.

- Operation of 16-bit free-run timer
- Operation of 16-bit output compare
- Operation of 16-bit input capture
- Timing of 16-bit free-run timer
  - Count timing
  - Clear timing
- Output compare timing
  - Timing of compare operation
  - Interrupt timing
  - Changing timing of the output pin
- Input timing of input capture
  - Capture timing to input signal

## 12.3.1 Operation of 16-bit Free-run Timer

The operation and timing of the 16-bit free-run timer are described.

### ■ Operation of 16-bit Free-run Timer

The 16-bit free-run timer starts counting at the counter value of "0000<sub>H</sub>" when a reset has been released. This counter value is a reference time for the 16-bit output compare and 16-bit input capture.

The count value is cleared by the following condition.

- When an overflow occurs
- When there is a comparing match with the value of compare clear register (setting mode is required).
- When "1" is written into the CLR bit in the TCCS register while it is operating.
- When you write "0000<sub>H</sub>" in the TCDC register while operating.
- At a reset

An interrupt is raised when an overflow occurs, or there is a comparing match with the value of compare clear register and the counter value of 16-bit free-run timer (the compare-match interrupt requires mode setting).

Figure 12.3-1 shows the timing chart for clearing the counter due to overflow, and Figure 12.3-2 shows the timing chart for clearing the counter due to compare-match operation.

Figure 12.3-1 Timing Chart of Counter Clear by Overflow

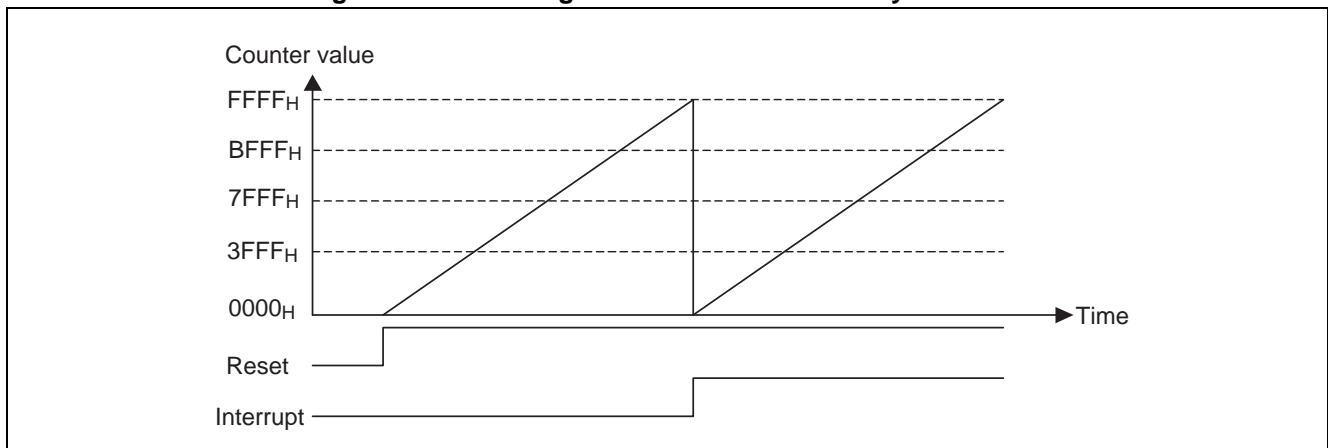
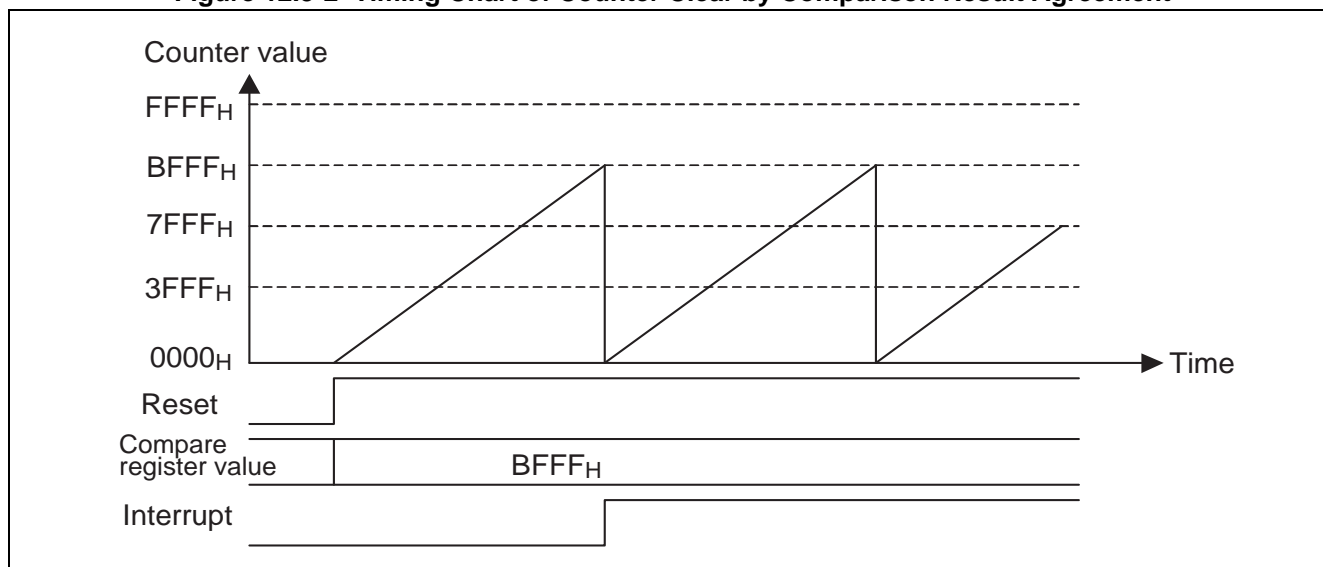


Figure 12.3-2 Timing Chart of Counter Clear by Comparison Result Agreement



## 12.3.2 Operation of 16-bit Output Compare

The 16-bit output compare compares the value of the 16-bit free-run timer with that set in one of the compare registers, sets the interrupt request flag and, at the same time, reverses the output level when a match is detected.

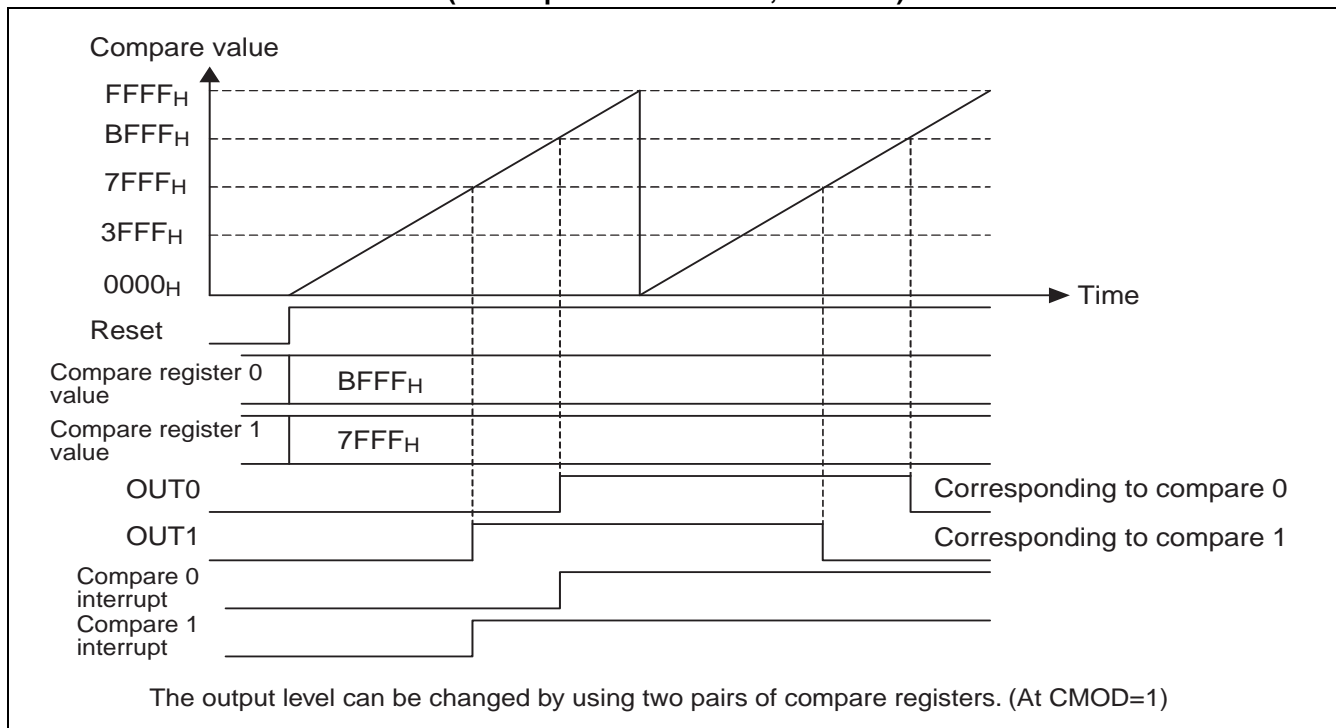
### ■ Example of Output Waveform

Examples of the output waveform are illustrated in the following.

- Example of the output waveform when using compare register 0 and 1:

Figure 12.3-3 shows an example of the output waveform when the output initial value is "0".

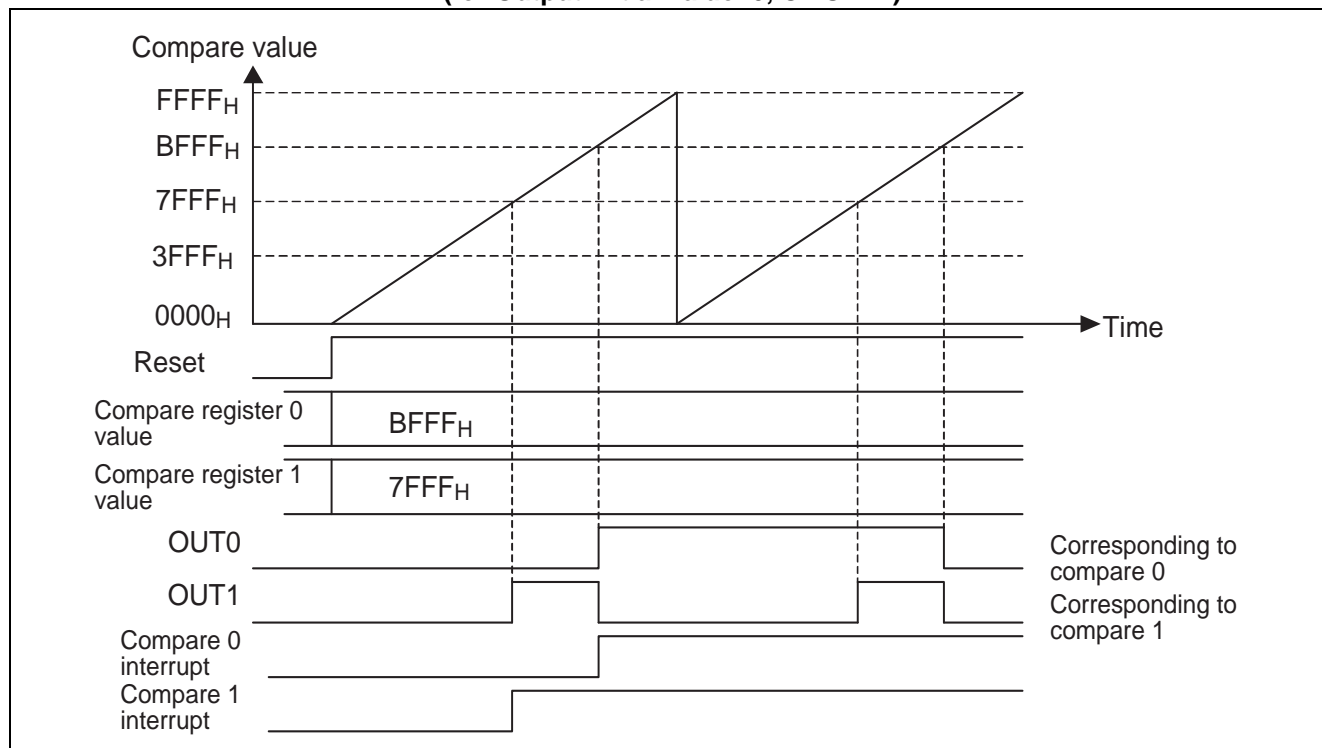
**Figure 12.3-3 Example of Output Waveform when Using Compare Register 0 and 1  
(for Output Initial Value=0, CMOD=0)**



● Example of the output waveform by two pairs of compare registers

Figure 12.3-4 shows an example of the output waveform when the output initial value is "0".

**Figure 12.3-4 Example of Output Waveform by Two Pairs of Compare Registers  
(for Output Initial Value=0, CMOD=1)**



Note:

To rewriting the compare register, within the compare interrupt routine or compare operation is disabled. Be sure not to occur simultaneously a compare match and writing the compare register.



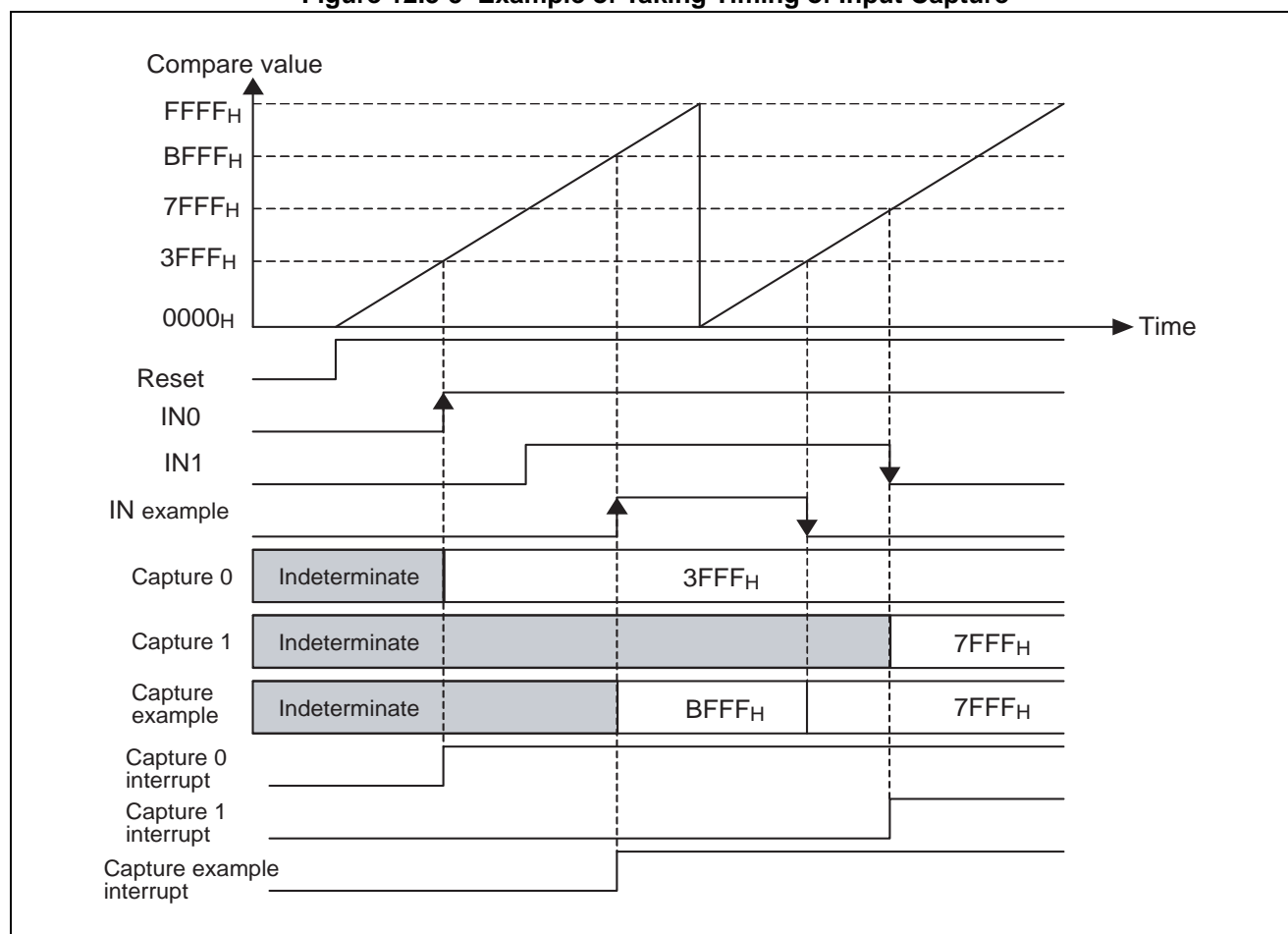
### 12.3.3 Operation of 16-bit Input Capture

The 16-bit input capture can capture the value of the 16-bit free-run timer into the capture register and generate an interrupt when it detects a predefined valid edge.

#### ■ Example of Taking Timing of Input Capture

Figure 12.3-5 shows the input capture capturing timing example.

Figure 12.3-5 Example of Taking Timing of Input Capture



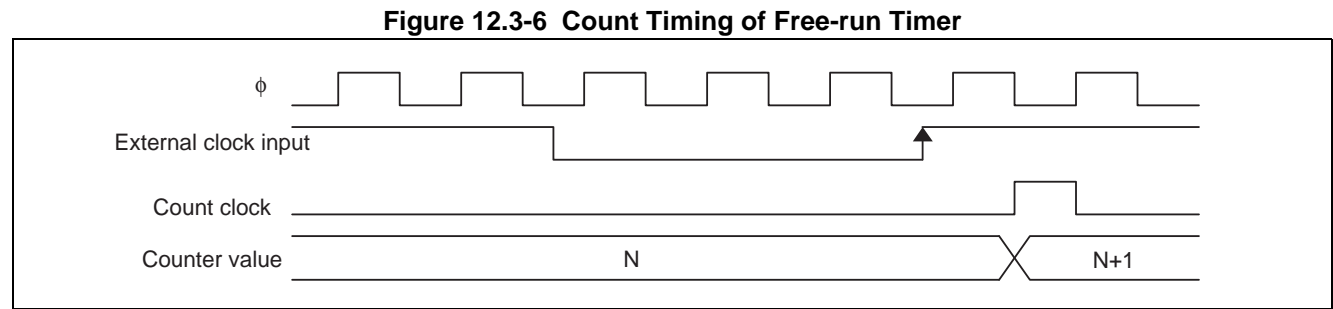
MB90330A Series

12.3.4 Timing of 16-bit Free-run Timer

The 16-bit free-run timer is counted up by the input clock (internal or external clock). When an external clock is selected, this timer is counted up at a rising edge.

■ Count Timing of Free-run Timer

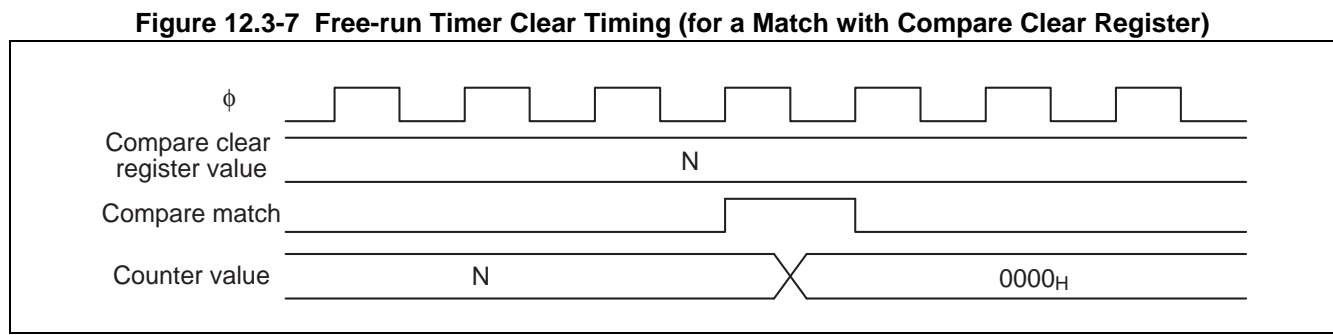
Figure 12.3-6 shows the count timing for the free-run timer.



The counter can be cleared by a reset operation, software, and match operation with compare clear register. Clearing the counter with a reset operation and software is performed as a clear is generated. Clearing the counter with a match with compare clear register is performed in sync with the count timing.

■ Free-run Timer Clear Timing (for a Match with Compare Clear Register)

Figure 12.3-7 shows the free-run timer clear timing for a match with compare clear register.



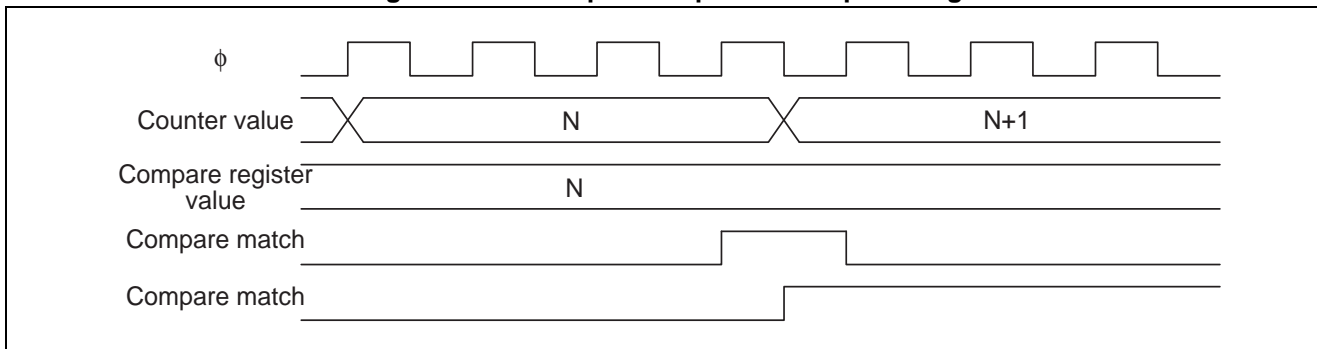
## 12.3.5 Output Compare Timing

The output compare indicates that a compare match signal that is generated when the free-run timer match the value set in the compare registers can reverse the output value and raise an interrupt. The output inverse timing when a comparing match is detected is in sync with the counter timing.

### ■ Interrupt Timing

Figure 12.3-8 shows the output compare interrupt timing.

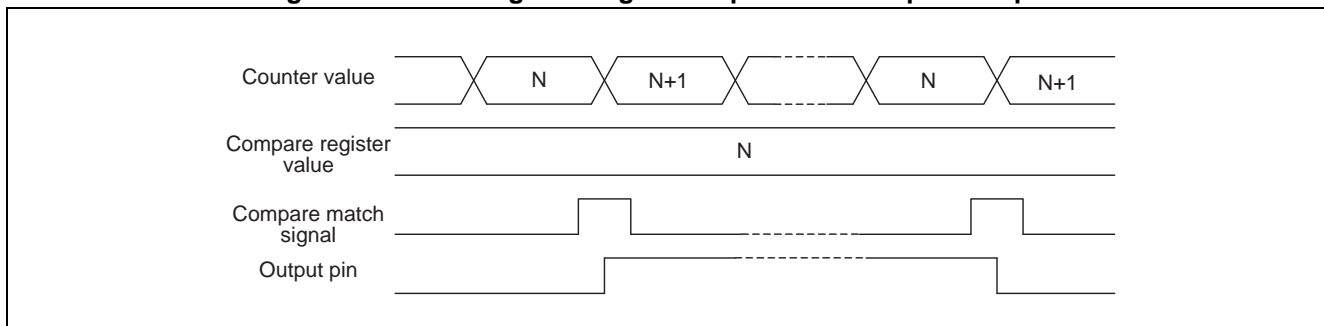
Figure 12.3-8 Output Compare Interrupt Timing



### ■ Change Timing for Output Pin

Figure 12.3-9 shows the change timing for the output pin of the output compare.

Figure 12.3-9 Change Timing for Output Pin of Output Compare



Note:

If you rewrite compare registers, you must rewrite them in a compare interrupt routine or in the state of disabled compare operation to ensure that compare-match operation and write operation never occur at the same time.

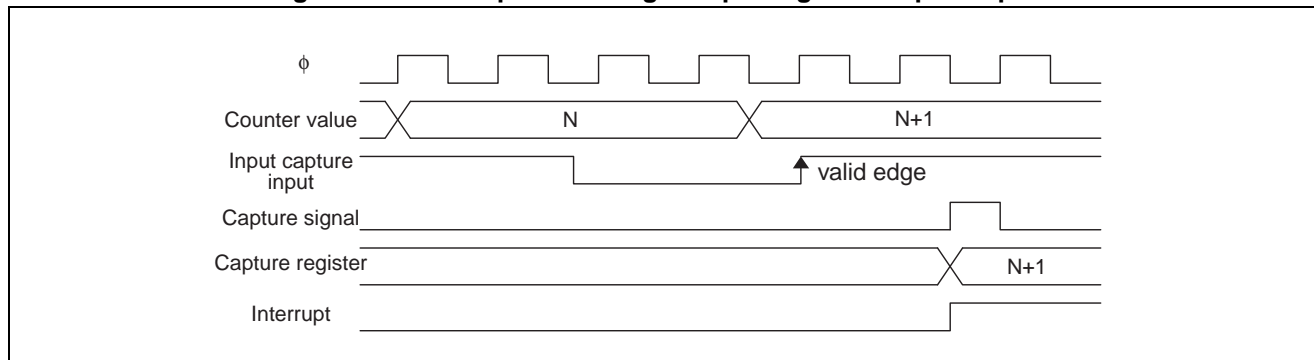
## 12.3.6 Input Timing of Input Capture

Capture timing to input signal of input capture is described.

### ■ Capture Timing to Input Signal

Figure 12.3-10 shows the capture timing to the input signal of the input capture.

**Figure 12.3-10 Capture Timing to Input Signal of Input Capture**





# ***CHAPTER 13***

---

# ***USB FUNCTION***

**This chapter describes the functions and overview of the USB Function.**

- 13.1 Overview of USB Function
- 13.2 Block Diagram of USB Function
- 13.3 Registers of USB Function
- 13.4 Operation Explanation of USB Function

## **13.1 Overview of USB Function**

---

**The USB Function is an interface that supports the USB (Universal Serial Bus) communication protocol. It operates supporting the transfer speed of FULL (12 Mbps) and has the following characteristics.**

---

### **■ Features of USB Function**

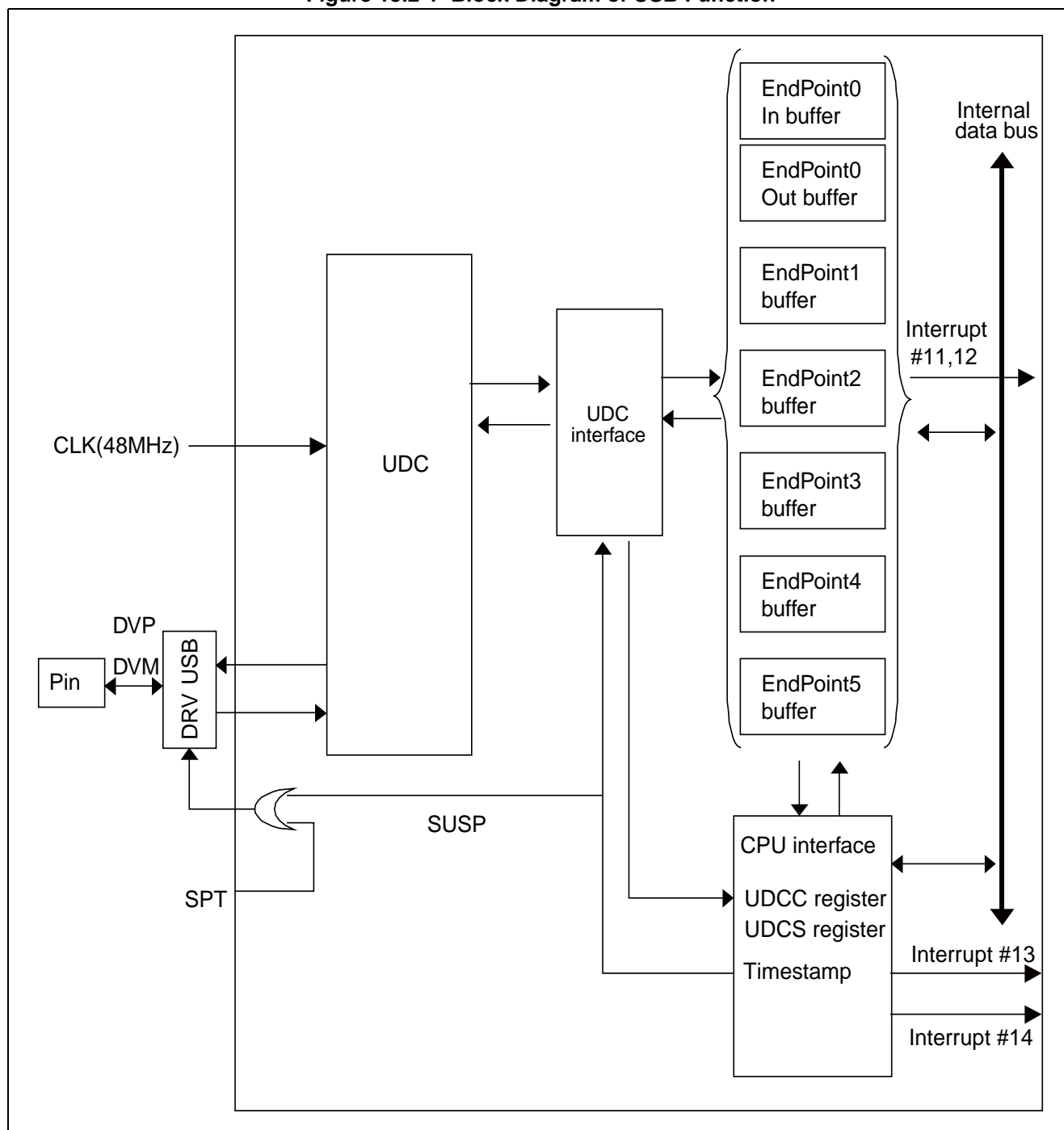
- FULL speed (12 Mbps) is supported. (Correspond to USB Full Speed)
- The device status is auto-answer.
- Bit string, bit stuffing, and automatic generation and check of CRC5 and CRC16
- Toggle check by data synchronization bit
- Automatic response to all standard commands except Get/SetDescriptor and SynchFrame commands (these three commands can be processed the same way as the class vendor commands.)
- The class vendor commands can be received as data and responded via firmware.
- Support up to six EndPoints (EndPoint0 is fixed to control transfer.)
- 2 built-in transfer data buffers for each EndPoint (each of two built-in buffers dedicated to IN and OUT, respectively, for EndPoint0)
- Support automatic transfer mode for transfer data via DMA (except buffers for EndPoint0)

## 13.2 Block Diagram of USB Function

Figure 13.2-1 shows the USB Function block diagram.

### ■ Block Diagram of USB Function

Figure 13.2-1 Block Diagram of USB Function





## 13.3 Registers of USB Function

The configuration and functions of registers used in the USB Function are described.

### ■ Register List of USB Function

Figure 13.3-1 Register List of USB Function

bit	15	8	7	0
	UDCC			(R/W)
	EP0C			(R/W)
	EP1C			(R/W)
	EP2C			(R/W)
	EP3C			(R/W)
	EP4C			(R/W)
	EP5C			(R/W)
	TMSP			(R)
	UDCIE		UDCS	(R/W)
	EP0IS			(R/W)
	EP0OS			(R/W)
	EP1S			(R/W)
	EP2S			(R/W)
	EP3S			(R/W)
	EP4S			(R/W)
	EP5S			(R/W)
	EP0DT			(R/W)
	EP1DT			(R/W)
	EP2DT			(R/W)
	EP3DT			(R/W)
	EP4DT			(R/W)
	EP5DT			(R/W)
	<div> <div>← 8 bits</div> <div>→</div> <div>← 8 bits →</div> </div>			

Figure 13.3-2 Registers of USB Function

bit	7	6	5	4	3	2	1	0	
Address:0000D0 <sub>H</sub>	RST	RESUM	HCON	USTP	Reserved	Reserved	RFBK	PWC	UDC control register (UDCC)
bit	15	14	13	12	11	10	9	8	
Address:0000D1 <sub>H</sub>	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	
bit	7	6	5	4	3	2	1	0	
Address:0000D2 <sub>H</sub>	Reserved	PKS0							EP0 control register (EP0C)
bit	15	14	13	12	11	10	9	8	
Address:0000D3 <sub>H</sub>	-	-	-	-	Reserved	Reserved	STAL	Reserved	
bit	7	6	5	4	3	2	1	0	
Address:0000D4 <sub>H</sub>	PKS1								EP1 control register (EP1C)
bit	15	14	13	12	11	10	9	8	
Address:0000D5 <sub>H</sub>	EPEN	TYPE		DIR	DMAE	NULE	STAL	PKS1	
bit	7	6	5	4	3	2	1	0	
Address:0000D6 <sub>H</sub>	Reserved	PKS2							EP2 control register (EP2C)
bit	15	14	13	12	11	10	9	8	
Address:0000D7 <sub>H</sub>	EPEN	TYPE		DIR	DMAE	NULE	STAL	Reserved	
bit	7	6	5	4	3	2	1	0	
Address:0000D8 <sub>H</sub>	Reserved	PKS3							EP3 control register (EP3C)
bit	15	14	13	12	11	10	9	8	
Address:0000D9 <sub>H</sub>	EPEN	TYPE		DIR	DMAE	NULE	STAL	Reserved	
bit	7	6	5	4	3	2	1	0	
Address:0000DA <sub>H</sub>	Reserved	PKS4							EP4 control register (EP4C)
bit	15	14	13	12	11	10	9	8	
Address:0000DB <sub>H</sub>	EPEN	TYPE		DIR	DMAE	NULE	STAL	Reserved	
bit	7	6	5	4	3	2	1	0	
Address:0000DC <sub>H</sub>	Reserved	PKS5							EP5 control register (EP5C)
bit	15	14	13	12	11	10	9	8	
Address:0000DD <sub>H</sub>	EPEN	TYPE		DIR	DMAE	NULE	STAL	Reserved	
bit	7	6	5	4	3	2	1	0	
Address:0000DE <sub>H</sub>	TMSP								Time stamp register (TMSP)
bit	15	14	13	12	11	10	9	8	
Address:0000DF <sub>H</sub>	-	-	-	-	-	TMSP			
bit	7	6	5	4	3	2	1	0	
Address:0000E0 <sub>H</sub>	-	-	SUSP	SOF	BRST	WKUP	SETP	CONF	UDC status register (UDCS)
bit	15	14	13	12	11	10	9	8	
Address:0000E1 <sub>H</sub>	Reserved	Reserved	SUSPIE	SOFIE	BRSTIE	WKUPIE	CONFN	CONFIE	UDC interruption permission register (UDCIE)
bit	7	6	5	4	3	2	1	0	
Address:0000E2 <sub>H</sub>	-	-	-	-	-	-	-	-	EP0I status register (EP0IS)
bit	15	14	13	12	11	10	9	8	
Address:0000E3 <sub>H</sub>	BFINI	DRQIE	-	-	-	DRQI	-	-	
bit	7	6	5	4	3	2	1	0	
Address:0000E4 <sub>H</sub>	Reserved	SIZE							EP0O status register (EP0OS)
bit	15	14	13	12	11	10	9	8	
Address:0000E5 <sub>H</sub>	BFINI	DRQOIE	SPKIE	-	-	DRQO	SPK	Reserved	

(Continued)

(Continued)

Address:0000E6 <sub>H</sub>	7	6	5	4	3	2	1	0	EP1 status register (EP1S)
	SIZE								
Address:0000E7 <sub>H</sub>	bit 15	14	13	12	11	10	9	8	
	BFINI	DRQIE	SPKIE	Reserved	BUSY	DRQ	SPK	SIZE	
Address:0000E8 <sub>H</sub>	bit 7	6	5	4	3	2	1	0	EP2 status register (EP2S)
	Reserved	SIZE							
Address:0000E9 <sub>H</sub>	bit 15	14	13	12	11	10	9	8	
	BFINI	DRQIE	SPKIE	Reserved	BUSY	DRQ	SPK	Reserved	
Address:0000EA <sub>H</sub>	bit 7	6	5	4	3	2	1	0	EP3 status register (EP3S)
	Reserved	SIZE							
Address:0000EB <sub>H</sub>	bit 15	14	13	12	11	10	9	8	
	BFINI	DRQIE	SPKIE	Reserved	BUSY	DRQ	SPK	Reserved	
Address:0000EC <sub>H</sub>	bit 7	6	5	4	3	2	1	0	EP4 status register (EP4S)
	Reserved	SIZE							
Address:0000ED <sub>H</sub>	bit 15	14	13	12	11	10	9	8	
	BFINI	DRQIE	SPKIE	Reserved	BUSY	DRQ	SPK	Reserved	
Address:0000EE <sub>H</sub>	bit 7	6	5	4	3	2	1	0	EP5 status register (EP5S)
	Reserved	SIZE							
Address:0000EF <sub>H</sub>	bit 15	14	13	12	11	10	9	8	
	BFINI	DRQIE	SPKIE	Reserved	BUSY	DRQ	SPK	Reserved	
Address:0000F0 <sub>H</sub>	bit 7	6	5	4	3	2	1	0	EP0 data register (EP0DT)
	BFDT								
Address:0000F1 <sub>H</sub>	bit 15	14	13	12	11	10	9	8	
	BFDT								
Address:0000F2 <sub>H</sub>	bit 7	6	5	4	3	2	1	0	EP 1 data register (EP1DT)
	BFDT								
Address:0000F3 <sub>H</sub>	bit 15	14	13	12	11	10	9	8	
	BFDT								
Address:0000F4 <sub>H</sub>	bit 7	6	5	4	3	2	1	0	EP 2 data register (EP2DT)
	BFDT								
Address:0000F5 <sub>H</sub>	bit 15	14	13	12	11	10	9	8	
	BFDT								
Address:0000F6 <sub>H</sub>	bit 7	6	5	4	3	2	1	0	EP 3 data register (EP3DT)
	BFDT								
Address:0000F7 <sub>H</sub>	bit 15	14	13	12	11	10	9	8	
	BFDT								
Address:0000F8 <sub>H</sub>	bit 7	6	5	4	3	2	1	0	EP 4 data register (EP4DT)
	BFDT								
Address:0000F9 <sub>H</sub>	bit 15	14	13	12	11	10	9	8	
	BFDT								
Address:0000FA <sub>H</sub>	bit 7	6	5	4	3	2	1	0	EP 5 data register (EP5DT)
	BFDT								
Address:0000FB <sub>H</sub>	bit 15	14	13	12	11	10	9	8	
	BFDT								

### 13.3.1 UDC Control Register (UDCC)

UDC control register (UDCC) controls the UDC core circuit.

#### ■ UDC Control Register (UDCC)

Figure 13.3-3 shows the bit configuration of the UDC control register (UDCC).

**Figure 13.3-3 UDC Control Register (UDCC)**

Address	bit	7	6	5	4	3	2	1	0	
0000D0 <sub>H</sub>		RST	RESUM	HCON	USTP	Reserved	Reserved	RFBK	PWC	UDC control register
		1	0	1	0	0	0	0	0	← Initial value
		R/W	R/W	R/W	R/W	-	-	R/W	R/W	← Access
Address	bit	15	14	13	12	11	10	9	8	
0000D1 <sub>H</sub>		Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	
		0	0	0	0	0	0	0	0	← Initial value
		-	-	-	-	-	-	-	-	← Access

R/W : Readable/Writable

Note:

The UDC control register (UDCC) should be set when bit7:RST=1 and not be rewritten when USB is in operating. However, bit 6 of RESUM and bit 4 of USTP are exclusive. RESUM of bit6 should be set or reset in suspend status of USB only by the remote wake-up enable status due to the following commands.

Set USTP of bit4 to "1" before entering the stop mode state.

To deselect the stop mode, set the order of SUSP in the UDCS and USTP in the UDCC to "0".

The following describes the function of each bit in the UDC control register (UDCC).

#### [bit15 to bit8] Reserved bits

These bits are reserved bits. Please write "0". The bit always reads "0".

#### [bit7] RST: Function reset bit

Apply an individual reset that is equivalent to the system reset to the chip to the USB Function. Apply a reset to the USB Function with the RST bit when connecting a cable to the HOST PC. Since the RST bit has the initial value of "1" and the USB Function is in reset status, release the Function by writing "0" to the bit.

RST	Operating mode
0	Reset of USB Function release
1	Reset USB Function

**Note:**

The RST bit initializes the corresponding bits of the timestamp register, UDC status register, and interrupt enable register at once. In addition, since it sets EP0I, EP0O, and BFINIs of EP1 to EP5 status registers at the same time after initialization, clear the RST bit (which does not clear the BFINI bits), and clear the BFINI of an endpoint to be used in this sequence.

**[bit6] RESUM: Resume setting bit**

When it is in remote Wake-up enabled status (or DEVICE\_REMOTE\_WAKEUP bit is set with the SET\_FEATURE command by the host) and in suspend status, the resume operation is started by writing "1" to the RESUM bit. Clear the resume direction by writing "0" to the RESUM bit that was set to "1".

RESUM	Operating mode
0	USB resume start dictating bit release
1	USB resume start dictate

**[bit5]  $\overline{\text{HCON}}$ : Host connection bit**

Control a switch between an external pull-up resistor and the USB data line and allows the USB Function to recognize connections to HOST PC or HUB.

$\overline{\text{HCON}}$	Operating mode
0	HOST PC or HUB and connection
1	HOST PC or HUB and state of cutting

**Note:**

Even if the external pull-up resistor is HOST at the state of ON or the connection is recognized from the HUB, the bus reset and the command of the USB bus ignores when the  $\overline{\text{HCON}}$  bit is set to "1".

**[bit4] USTP:USB operation clock stop bit**

Stop the clock of the USB operation part by setting the USTP bit. Setting the USTP bit can reduce power consumption when keeping the USB out of service.

USTP	Operating mode
0	Normal mode
1	Clock stop of USB operation part

---

Note:

If the USTP bit is not used in the stop mode, wait for 3 cycles or 43 cycles to elapse in FULL speed or in LOW speed (that is supported only in HOST mode) so that you can ensure that the reset operation will function when you have set RST=1. You may clear the USTP bit and the RST bit at once.

---

**[bit3, bit2] Reserved bits**

It is reserved bit. Please write "0". The bit always reads "0".

**[bit1] RFBK: Data toggle mode selection bit (rate feedback mode)**

It is a bit that selects data toggle mode for USB Interrupt transfer.

RFBK	Operating mode
0	Selection of alternation data toggle mode It toggles a data PID when a transfer has been successfully completed.
1	Selection of data toggle mode It is a toggle the data PID in unconditional.

The data toggle mode selection can be used for rate feedback information in ISO transfer.

**[bit0] PWC: Power supply control bit**

It specifies the operating power supply mode (self power or bus power) for the USB Function.

(The setting of the PWC bit is reflected in the standard command GetStatus.)

PWC	Operating mode
0	Bus power
1	Self power

## 13.3.2 EP0 Control Register (EP0C)

EP0 control register (EP0C) controls concerning end point 0.

### ■ EP0 Control Register (EP0C)

Figure 13.3-4 shows the bit configuration of the EP0 control register (EP0C).

Figure 13.3-4 EP0 Control Register (EP0C)

Address	bit	7	6	5	4	3	2	1	0	
0000D2 <sub>H</sub>		Reserved	PKS0							EP0 control register
		0	1	0	0	0	0	0	0	← Initial value
		-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	← Access
Address	bit	15	14	13	12	11	10	9	8	
0000D3 <sub>H</sub>		-	-	-	-	Reserved	Reserved	STAL	Reserved	
		X	X	X	X	0	0	0	0	← Initial value
		-	-	-	-	-	-	R/W	-	← Access

R/W : Readable/Writable

Note:

Ensure that you must set the EP0 control register (EP0C), except bit9 STAL, when both bit7 RST of the UDC control register (UDCC) and bit15 BFINI of the EP0I/EP0O status register (EP0IS/EP0OS) are "1" and must not rewrite it while the USB is operating.

The following describes the function of each bit in the EP0 control register (EP0C).

#### [bit15 to bit12] Undefined bits

Writing has no effect on the operation. Reading is undefined.

#### [bit11, bit10] Reserved bits

It is reserved bit. Please write "00<sub>B</sub>".

These bits always reads "00<sub>B</sub>" when read.

#### [bit9] STAL:STALL EndPoint0 set bit

Setting the STAL bit can put EndPoint0 in STALL status (STALL response).

STAL	Operating mode
0	Release of state of STALL
1	Set of state of STALL (STALL response)

---

Note:

The STALL response is continued to the host while the STAL bit is set. The USB Function returns from STALL status when it receives a normal SETUP packet after the STAL bit is deselected.

---

**[bit8, bit7] Reserved bits**

It is reserved bit. Please write "00<sub>B</sub>".

These bits always reads "00<sub>B</sub>" when read.

**[bit6 to bit0] PKS0:Packet size end point 0 set bits**

It specifies the maximum number of transfer bytes per packet. The maximum number of transfer bytes per packet that EndPoint0 can specify is 64 bytes, which is a setting common to IN and OUT.

<Example> "08<sub>H</sub>" → 8 Byte, "40<sub>H</sub>" → 64 Byte (maximum specified value)

---

Note:

Setting a value not less than the maximum number of transfer bytes (40<sub>H</sub>) and "00<sub>H</sub>" is forbidden.

---



### 13.3.3 EP1 to EP5 Control Register (EP1C to EP5C)

EP1 to EP5 control register (EP1C to EP5C) controls concerning end point 1 to end point 5.

#### ■ EP1 to EP5 Control Register (EP1C to EP5C)

Figure 13.3-5 shows the bit configuration of the EP1 to EP5 control register (EP1C to EP5C).

Figure 13.3-5 EP1 to EP5 Control Register (EP1C to EP5C)

Address	bit	7	6	5	4	3	2	1	0	
EP1C 0000D4 <sub>H</sub>		PKS1								
		0	0	0	0	0	0	0	0	← Initial value
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	← Access
Address	bit	7	6	5	4	3	2	1	0	
EP2C 0000D6 <sub>H</sub>		Reserved	PKS5 to PKS2							
EP3C 0000D8 <sub>H</sub>		0	1	0	0	0	0	0	0	← Initial value
EP4C 0000DA <sub>H</sub>		-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	← Access
EP5C 0000DC <sub>H</sub>										
Address	bit	15	14	13	12	11	10	9	8	
EP1C 0000D5 <sub>H</sub>		EPEN	TYPE		DIR	DMAE	NULE	STAL	PKS1	
		0	1	1	0	0	0	0	1	← Initial value
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	← Access
Address	bit	15	14	13	12	11	10	9	8	
EP2C 0000D7 <sub>H</sub>		EPEN	TYPE		DIR	DMAE	NULE	STAL	Reserved	
EP3C 0000D9 <sub>H</sub>		0	1	1	0	0	0	0	0	← Initial value
EP4C 0000DB <sub>H</sub>		R/W	R/W	R/W	R/W	R/W	R/W	R/W	-	← Access
EP5C 0000DD <sub>H</sub>										

R/W : Readable/Writable

Note:

Ensure that you must set the EP1 to EP5 control registers (EP1C to EP5C), except DMAE, NULE, and STAL, when both bit7 RST of the UDC control register (UDCC) and bit15 BFINI of the EP0 to EP5 status registers (EP1S to EP5S) are "1" and must not rewrite them while the USB is operating.

The following describes the function of each bit in the EP1 to EP5 control register (EP1C to EP5C).

#### [bit15] EPEN: End Point 1 to End Point 5 permission bit

The end point is made effective. Setting the EPEN bit allows it to be configured by the host as an end point for use in the USB Function. TYPE, DIR, and PKS of the EP1 to EP5 control registers become valid for configuration information.

EPEN	Operating mode
0	End Point is invalid.
1	End Point is assumed to be effective.

**[bit14, bit13] TYPE: End point forwarding type selection bits**

The forwarding type supported by the end point is specified.

TYPE	Operating mode
00	Specification prohibited
01	Isochronous transfer
10	Bulk transfer
11	Interrupt transfer

**Note:**

In Isochronous transfer of USB 2.0, the following item was added to the specification. Initial setting: Alternate value 0 and Maximum number of packets 0. For USB function of this series, if the Alternate value is set to the number other than 0, STALL response is given to Host automatically and the Alternate value cannot be changed. Therefore, maximum number of packets cannot be set by changing the Alternate value.

**[bit12] DIR: End point forwarding direction selection bit**

It specifies a transfer direction that the end point supports.

DIR	Function operating mode	HOST operating mode (for EP1 and EP2 only)
0	OUT end point	IN end point
1	IN end point	OUT end point

**[bit11] DMAE: DMA automatic transfer enable bit**

It is a mode setting that uses DMA for read and write operations to transmit and receive buffers for transfer data and automatically transfers transmit and receive data in sync with IN and OUT data requests from the HOST until data whose pieces are as many as the number of transfer data set in DMA is transferred.

DMAE	Operating mode
0	Release of the automatic buffer TRANSFER mode
1	Set of the automatic buffer TRANSFER mode

**Note:**

Access to transmit and receive buffers by CPU is forbidden while the DMAE bit is set. Set the number of DMA transfer data to a multiple of the value of the PKS set in direction the EP1 to EP5 control registers (EP1C to EP5C) when transferring data to OUT.

**[bit10] NULE: NULL automatic transfer enable bit**

This bit sets up a mode where the last packet transfer will be detected and 0-byte data transfer will be automatically sent when IN- direction data transfer request arrives if the automatic buffer transfer mode is set (DMAE=1).

NULE	Operating mode
0	Release of the NULL automatic TRANSFER mode
1	Set of the NULL automatic TRANSFER mode

Note:

Setting the NULE bit has no effect on communications when transferring data to OUT direction or the automatic buffer transfer mode is not set.

**[bit9] STAL:STALL set bit**

Setting the STAL bit can put EndPoint in STALL status (STALL response).

STAL	Operating mode
0	Release of state of STALL
1	Set of state of STALL (STALL response)

Note:

STALL keeps responding for HOST while setting the STAL bit. The USB Function can return from STALL status with the ClearFutcher command from the host after the STAL bit was deselected.

**EP2 to EP5: [bit8, bit7] Reserved bits**

These bits are reserved bit. Please write "00<sub>B</sub>". These bits always read "00<sub>B</sub>".

**EP1:[bit8 to bit0] PKS: Packet size set bits**

**EP2 to EP5:[bit6 to bit0] PKS: Packet size set bits**

The number of maximum forwarding by one packet is specified. The following lists a maximum number of transfer packets that can be specified in each of EndPoint1 to EndPoint5.

EndPoint	Max. number of transfer	Range which can be set
1	256 bytes (The odd can be set).	001 <sub>H</sub> to 100 <sub>H</sub>
2 to 5	64 bytes (The odd can be set).	01 <sub>H</sub> to 40 <sub>H</sub>

---

Note:

Setting any number not less than a maximum number of transfer (100<sub>H</sub> or 40<sub>H</sub>) and "00<sub>H</sub>" are prohibited. Please write "00" in bit8, bit7 about EndPoint2 to EndPoint5. In addition, when using the automatic buffer transfer mode (DMAE=1), setting bit0 to bit2 in the corresponding EndPoint is forbidden.

---

13.3.4 Time Stamp Register (TMSP)

The time stamp register (TMSP) displays a frame number when an SOF packet is received.

■ Time Stamp Register (TMSP)

Figure 13.3-6 shows the bit configuration of the timestamp register (TMSP).

Figure 13.3-6 Time Stamp Register (TMSP)

Address bit	7	6	5	4	3	2	1	0	
0000DE <sub>H</sub>	TMSP								Time stamp register
	0	0	0	0	0	0	0	0	← Initial value
	0	0	0	0	0	0	0	0	← RST Reset
	R	R	R	R	R	R	R	R	← Access
Address bit	15	14	13	12	11	10	9	8	
0000DF <sub>H</sub>	-	-	-	-	-	TMSP			
	X	X	X	X	X	0	0	0	← Initial value
	X	X	X	X	X	0	0	0	← RST Reset
	-	-	-	-	-	R	R	R	← Access

R : Read only

The functions of each bit of the time stamp register (TMSP) are described below.

[bit15 to bit11] Undefined bits

Writing has no effect on the operation. Reading is undefined.

[bit10 to bit0] TMSP: Time stamp bits

Frame number by the reception of the SOF packet is shown. When the SOF packet is received, frame number is updated.

### 13.3.5 UDC Status Register (UDCS)

The UDC status register (UDCS) is a register that indicates the status of a bus on USB communications and a particular command received. Each bit in the register except SETP indicates an interrupt factor and raises an interrupt to CPU if its corresponding interrupt enable bit is specified and valid.

#### ■ UDC Status Register (UDCS)

Figure 13.3-7 shows the bit configurations of the UDC status register (UDCS).

Figure 13.3-7 UDC Status Register (UDCS)

Address	bit	7	6	5	4	3	2	1	0	
0000E0 <sub>H</sub>		-	-	SUSP	SOF	BRST	WKUP	SETP	CONF	UDC status register
		X	X	0	0	0	0	0	0	← Initial value
		X	X	0	0	0	0	0	0	← RST Reset
		-	-	R/W	R/W	R/W	R/W	R/W	R/W	← Access
R/W : Readable/Writable										

The function of each bit in the UDC status register (UDCS) is described in the following.

#### [bit7, bit6] Undefined bits

Writing has no effect on the operation. Reading is undefined.

#### [bit5] SUSP: Suspend detection bit

It displays the fact that the USB Function shifts to suspend status. The SUSP bit is an interrupt factor and writing "1" is ignored. Please clear by writing "0". "1" is read at the read modification write.

SUSP	Operating mode
0	Suspend undetection and interruption clear factor
1	Suspend detection

#### [bit4] SOF:SOF reception detection bit

It indicates that an SOF packet has been received, and the value of the time stamp register is updated. The SOF bit is an interrupt factor and writing "1" is ignored. Please clear by writing "0". "1" is read at the read modification write.

SOF	Operating mode
0	SOF unreception and interruption clear factor
1	The SOF packet is received.

**[bit3] BRST: Bus reset detection bit**

The detection of USB bus reset is displayed. The BRST bit is an interrupt factor and writing "1" is ignored. Please clear by writing "0". "1" is read at the read modification write.

BRST	Operating mode
0	USB bus reset undetected and interrupt factors clear
1	USB bus reset is detected.

---

**Note:**

Set registers again by initializing the USB Function with RST in the UDCC register when the BRST bit is detected.

---

**[bit2] WKUP: Wake-Up detection bit**

It displays the fact that the USB Function has returned from suspend status. What causes the USB Function to return from suspend status are a remote wake-up by setting the RESUM bit and a wake-up from the host request, and the WKUP bit is automatically set only by a return request from the host. The WKUP bit is an interrupt factor and writing "1" is ignored. Please clear by writing "0". "1" is read at the read modification write.

WKUP	Operating mode
0	HOST factor RESUME undetected and interrupt factors clear
1	HOST factor RESUME is detected.

---

**Note:**

Even if a wake up is caused by a host request, the WKUP bit is not set when the RESUM bit in the UDCC register is set.

---

**[bit1] SETP:SETUP stage detection bit**

It indicates that received data belongs to the Setup stage of USB control transfer. "1" Writing is disregarded. Please clear by writing "0". "1" is read at the read modification write.

SETP	Operating mode
0	SETUP unreception and clear factor
1	Control forwarding SETUP stage is received.

---

**Note:**

It is not set when automatically responding to any standard command. The SETP bit is not an interruption factor.

---

## [bit0] CONF: Configuration detection bit

It displays the fact that the USB Function has been configured. The CONF bit is set when a SetConfig, a USB command, has been successfully received. The CONF bit is an interrupt factor and writing "1" is ignored. Please clear by writing "0". "1" is read at the read modification write.

CONF	Operating mode
0	SetConfig undetection and interruption clear factor
1	SetConfig is detected.



### 13.3.6 UDC Interrupt Enable Register (UDCIE)

The UDC interrupt enable register (UDCIE) is a register that allows each interrupt factor in the UDC status register to be raised as an interrupt bit wisely except CONFN.

#### ■ UDC Interrupt Enable Register (UDCIE)

Figure 13.3-8 shows the bit configuration of the UDC interrupt enable register (UDCIE).

Figure 13.3-8 UDC Interrupt Enable Register (UDCIE)

Address	bit	15	14	13	12	11	10	9	8	
0000E1 <sub>H</sub>		Reserved	Reserved	SUSPIE	SOFIE	BRSTIE	WKUPIE	CONFN	CONFIE	UDC interrupt enable register
		0	0	0	0	0	0	0	0	← Initial value
		0	0	0	0	0	0	0	0	← RST Reset
		-	-	R/W	R/W	R/W	R/W	R	R/W	← Access
R/W : Readable/Writable										
R : Read Only										

The function of each bit in the UDC interrupt enable register (UDCIE) is described in the following.

#### [bit15, bit14] Reserved bits

It is reserved bit. Please write "00<sub>B</sub>". These bits always read "00<sub>B</sub>".

#### [bit13] SUSPIE: Suspend interrupt enable bit

It allows an interrupt due to the interrupt factor for the UDC status register "SUSP" to be generated.

SUSPIE	Operating mode
0	Interrupt disabled by SUSP factor
1	Interruption permission by SUSP factor

#### [bit12] SOFIE: SOF reception interruption permission bit

It allows an interrupt due to the interrupt factor for the UDC status register "SOF" to be generated.

SOFIE	Operating mode
0	Interrupt disabled by SOF factor
1	Interruption permission by SOF factor

## [bit11] BRSTIE: Bus reset interruption permission bit

It allows an interrupt due to the interrupt factor for the UDC status register "BRST" to be generated.

BRSTIE	Operating mode
0	Interrupt disabled by BRST factor
1	Interruption permission by BRST factor

## [bit10] WKUPIE: Wake-Up interruption permission bit

It allows an interrupt due to the interrupt factor for the UDC status register "WKUP" to be generated.

WKUPIE	Operating mode
0	Interrupt disabled by WKUP factor
1	Interruption permission by WKUP factor

## [bit9] CONFN: Configuration number display bit

It displays a configuration number. It is updated when setting the interrupt cause for the UDC status register "CONF".

CONFN	Operating mode
0	CONFIG Number 0
1	CONFIG Number 1

## [bit8] CONFIE: Configuration interrupt enable bit

It allows an interrupt due to the interrupt factor for the UDC status register "CONF" to be generated.

CONFIE	Operating mode
0	Interrupt disabled by CONF factor
1	Interruption permission by CONF factor

### 13.3.7 EP0I Status Register (EP0IS)

The EP0I status register (EP0IS) displays status related to transfer toward In for EndPoint0.

#### ■ EP0I Status Register (EP0IS)

Figure 13.3-9 shows the bit configuration of the EP0I Status Register (EP0IS).

Figure 13.3-9 EP0I Status Register (EP0IS)

Address	bit	7	6	5	4	3	2	1	0	
0000E2 <sub>H</sub>		-	-	-	-	-	-	-	-	EP0I status register
		X	X	X	X	X	X	X	X	← Initial value
		X	X	X	X	X	X	X	X	← BFINI Reset
		-	-	-	-	-	-	-	-	← Access
Address	bit	15	14	13	12	11	10	9	8	
0000E3 <sub>H</sub>		BFINI	DRQIIE	-	-	-	DRQI	-	-	
		1	0	X	X	X	1	X	X	← Initial value
		1	Irrelevance	X	X	X	1	X	X	← BFINI Reset
		R/W	R/W	-	-	-	R/W	-	-	← Access

R/W : Readable/Writable

The function of each bit in the EP0I status register (EP0IS) is described in the following.

#### [bit15] BFINI: Transmission buffer initialization bit

The forwarding data transmission buffer is initialized. The BFINI bit is automatically set by setting the RST bit in the UDC control register (UDCC). Consequently, when the reset operation has been performed with the RST bit, clear the RST bit before clearing the BFINI bit.

BFINI	Operating mode
0	Cancelling Initialization
1	Initialization of transmission buffer

Note:

The initialization of the BFINI bit initializes a buffer and the DRQI bit. You must initialize the buffer when you have set the STAL bit if necessary after you ensure that the DRQI or DRQO bit is set and there is no access from the HOST.

## [bit14] DRQIIE: Transmit data interrupt enable bits

It allows an interrupt due to the interrupt factor for the EP0I status register "DRQI" to be generated.

DRQIIE	Operating mode
0	Interrupt disabled by DRQI factor
1	Interruption permission by DRQI factor

## [bit13 to bit11] Undefined bits

Writing has no effect on the operation. Reading is undefined.

## [bit10] DRQI: Transmission data interrupt request bit

It indicates that IN packet has been successfully transferred from the EP0 host, data has been read from the transmission buffer, and the next transmit data can be written into the buffer. The DRQI bit is a interrupt factor and writing "1" is ignored. Please clear by writing "0". "1" is read at the read modification write.

DRQI	Operating mode
0	Clearing interrupt cause
1	Writing transmit data enable state

### Note:

After the data write of the transmission buffer is processed, the DRQI must be cleared. Also, when the DRQI is not set, writing "0" is prohibited. When the DRQI is set to "1", writing data to the transmission buffer is enabled. Furthermore, it indicates the data is set to the transmission buffer at the time of clearing. Therefore, when IN packet request is performed with DRQI set "1", the NAK is responded to the HOST automatically.

## [bit9 to bit0] Undefined bits

Writing has no effect on the operation. Reading is undefined.

## 13.3.8 EP0O Status Register (EP0OS)

The EP0O status register (EP0OS) displays status related to transfer toward out for EndPoint0.

### ■ EP0O Status Register (EP0OS)

Figure 13.3-10 shows the bit configuration of the EP0O Status Register (EP0OS).

Figure 13.3-10 EP0O Status Register (EP0OS)

Address	bit	7	6	5	4	3	2	1	0	
0000E4 <sub>H</sub>		SIZE								EP0O status register
		0	X	X	X	X	X	X	X	← Initial value
		0	X	X	X	X	X	X	X	← BFINI Reset
		-	R	R	R	R	R	R	R	← Access
Address	bit	15	14	13	12	11	10	9	8	
0000E5 <sub>H</sub>		BFINI	DRQOIE	SPKIE	-	-	DRQO	SPK	Reserved	
		1	0	0	X	X	0	0	0	← Initial value
		1	Irrelevance	Irrelevance	X	X	0	0	0	← BFINI Reset
		R/W	R/W	R/W	-	-	R/W	R/W	-	← Access

R/W : Readable/Writable  
R : Read Only

The function of each bit in the EP0O status register (EP0OS) is described in the following.

#### [bit15] BFINI: Reception buffer initialization bit

The forwarding data reception buffer is initialized. The BFINI bit is automatically set by setting the RST bit in the UDC control register (UDCC). Consequently, when the reset operation has been performed with the RST bit, clear the RST bit before clearing the BFINI bit.

BFINI	Operating mode
0	Cancelling Initialization
1	Initialization of reception buffer

#### Note:

The initialization of the BFINI bit initializes a DRQO and the SPK bit. You must initialize the buffer when you have set the STAL bit if necessary after you ensure that the DRQI or DRQO bit is set and there is no access from the HOST.

#### [bit14] DRQOIE: Received data interruption permission bit

It allows an interrupt due to the interrupt factor for the EP0O status register "DRQO" to be generated.

DRQOIE	Operating mode
0	Interrupt disabled by DRQO factor
1	Interruption permission by DRQO factor

#### [bit13] SPKIE: Short packet interruption permission bit

It allows an interrupt due to the interrupt factor for the EP0O status register "SPK" to be generated.

SPKIE	Operating mode
0	Interrupt disabled by SPK factor
1	Interruption permission by SPK factor

#### [bit12, bit11] Undefined bits

These bits are undefined at read. No effect on writing.

#### [bit10] DRQO: Received data interrupt request bit

It indicates that OUT packet has been successfully transferred from the EP0 host, data has been written into the receive buffer, and the receive data can be read from the buffer. The DRQO bit is an interrupt factor and writing "1" is ignored. Please clear by writing "0". "1" is read at the read modification write.

DRQO	Operating mode
0	Clearing interrupt cause
1	Reading receive data enable state

#### Note:

After the data read of reception buffer is processed, the DRQO must be cleared. Also, writing "0" when the DRQO is not set is disabled. When the DRQO is set to "1", the reception buffer is not updated. The buffer is enabled to update when it is cleared. When the OUT packet request is executed with DRQO set "1", the NAK is responded to the HOST automatically.

#### [bit9] SPK: Short packet interrupt request bit

It indicates that the number of pieces of transfer data that has been successfully received from the host is less than a maximum number of packets set in PKS in the EP0 control register (EP0C) (including 0 byte). The SPK bit is a interrupt factor and writing "1" is ignored. Please clear by writing "0". "1" is read at the read modification write.

SPK	Operating mode
0	Maximum number of transfer packets received
1	Data less than the maximum number of transfer packets received

**[bit8, bit7] Reserved bits**

These bits are reserved bits. Writing has no effect on the operation. Reading is undefined.

**[bit6 to bit0] SIZE: Packet size display bit**

When OUT packets have been transferred from EP0, the number of data bytes that has been written into the receive buffer is displayed. The SIZE bit is updated to a valid value when the interrupt factor for DRQO in the EP0O status register (EP0OS) has been set.

<Example> 8 bytes → "08<sub>H</sub>", 64 bytes → "40<sub>H</sub>" (maximum value)

### 13.3.9 EP1 to EP5 Status Register (EP1S to EP5S)

The EP1 to EP5 status registers (EP1S to EP5S) displays status related to EndPoint1 to EndPoint5.

#### ■ EP1 to EP5 Status Register (EP1S to EP5S)

Figure 13.3-11 shows the bit configurations of the EP1 to EP5 status registers (EP1S to EP5S).

**Figure 13.3-11 EP1 to EP5 Status Register (EP1S to EP5S)**

Address	bit	7	6	5	4	3	2	1	0	
EP1S 0000E6 <sub>H</sub>		SIZE								
		X	X	X	X	X	X	X	X	← Initial value
		X	X	X	X	X	X	X	X	← BFINI Reset
		R	R	R	R	R	R	R	R	← Access
Address	bit	7	6	5	4	3	2	1	0	
EP2S 0000E8 <sub>H</sub>		Reserved	SIZE							
EP3S 0000EA <sub>H</sub>	0	X	X	X	X	X	X	X	X	← Initial value
EP4S 0000EC <sub>H</sub>	0	X	X	X	X	X	X	X	X	← BFINI Reset
EP5S 0000EE <sub>H</sub>	-	R	R	R	R	R	R	R	R	← Access
Address	bit	15	14	13	12	11	10	9	8	
EP1S 0000E7 <sub>H</sub>		BFINI	DRQIE	SPKIE	Reserved	BUSY	DRQ	SPK	SIZE	
		1	0	0	-	0	0	0	X	← Initial value
		1	Irrelevance	Irrelevance	-	Irrelevance	0	0	X	← BFINI Reset
		R/W	R/W	R/W	-	R	R/W	R/W	R	← Access
Address	bit	15	14	13	12	11	10	9	8	
EP2S 0000E9 <sub>H</sub>		BFINI	DRQIE	SPKIE	Reserved	BUSY	DRQ	SPK	Reserved	
EP3S 0000EB <sub>H</sub>	1	0	0	-	0	0	0	0	0	← Initial value
EP4S 0000ED <sub>H</sub>	1	Irrelevance	Irrelevance	-	Irrelevance	0	0	0	0	← BFINI Reset
EP5S 0000EF <sub>H</sub>		R/W	R/W	R/W	-	R	R/W	R/W	-	← Access

R/W : Readable/Writable  
R : Read Only

The function of each bit in the EP1 to EP5 status register (EP1S to EP5S) is described in the following.



**[bit15] BFINI: Transmission/receive buffer initialization bit**

The transmission and reception buffer of the forwarding data is initialized. The BFINI bit is automatically set by setting the RST bit in the UDC control register (UDCC). Consequently, when the reset operation has been performed with the RST bit, clear the RST bit before clearing the BFINI bit.

BFINI	Operating mode
0	Cancelling Initialization
1	Initialization of transmitting and receiving buffer

---

Note:

The transmission/receive buffer for EP1 to EP5 has a configuration of double buffers and initialization by the BFINI bit initializes the double buffers at once and also initializes the DRQ and SPK bits. You must initialize the buffers when you have set the STAL bit after you ensure that the DRQ bit is set and the BUSY bit shows no access from the HOST.

---

**[bit14] DRQIE: Packet forwarding interruption permission bit**

It allows an interrupt due to the interrupt factor for the EP1 to EP5 status register "DRQ" to be generated.

DRQIE	Operating mode
0	Interrupt disabled by DRQ factor
1	Interruption permission by DRQ factor

---

Note:

If you use the automatic buffer transfer mode (DMAE=1), you must enable the settings of DMA and transfer before enabling the DRQIE bit.

---

**[bit13] SPKIE: Short packet interruption permission bit**

It allows an interrupt due to the interrupt factor for the EP1 to EP5 status register "SPK" to be generated.

SPKIE	Operating mode
0	Interrupt disabled by SPK factor
1	Interruption permission by SPK factor

**[bit12] Reserved bit**

This bit is reserved bit. Writing has no effect on the operation. Reading is undefined.

## [bit11] BUSY: Busy flag bit

It indicates that writing into the transmission/receive buffer or accessing it for read from the HOST is under way. The BUSY bit is set by the automatic operation, and reset.

BUSY	Operating mode
0	There is no access by HOST.
1	During writing/reading from HOST

### Note:

It indicates that the HOST is accessing a buffer that is different from either of the double buffer accessed from CPU or DMA when the DRQ bit is set and the BUSY bit is set. Normally, you do not need to control via the BUSY bit, but if you initialize the buffer with BFINI set, you must initialize the buffer by setting the STAL bit after you ensure that the DRQ bit is set and the BUSY bit shows no access from the HOST.

## [bit10] DRQ: Packet forwarding interrupt request bit

It indicates that the packet transfer for EP1 to EP5 has been successfully completed and data processing is needed. The DRQ bit is a interrupt factor and writing "1" is ignored. Please clear by writing "0". "1" is read at the read modification write.

DRQ	Operating mode
0	Clearing Interrupt cause
1	The packet forwarding ends normally.

### Note:

When the automatic buffer transfer mode (DMAE=1) is not used after the data read or write of transmission and reception buffers is processed, "0" must be write to DRQ bit. When the DRQ bit is cleared, access buffer is switched. When the transfer direction is set to IN direction if the DRQ bit is "1" and the buffer is cleared without writing data, 0-byte data is set to it. In the initial setting, when the DIR of the EP1 to EP5 control register (EP1C to EP5C) is set to "1", the DRQ bit of the corresponding end point is set at the same time. Furthermore, writing 0 is prohibited when the DRQ bit is not set.

**[bit9] SPK: Short packet interrupt request bit**

It indicates that the number of pieces of transfer data that has been successfully received from the host is less than a maximum number of packets set the PKS in the EP1 to EP5 control register (EP1C to EP5C) (including 0 packet). The SPK bit is a interrupt factor and writing "1" is ignored. Please clear by writing "0". "1" is read at the read modification write.

SPK	Operating mode
0	Max. number of transfer packets received
1	Data less than the Max. number of transfer packets received

Note:

The SPK bit is not set when data toward IN is transferred.

**EP2 to EP5:[bit8, bit7] Reserved bits**

In EP2 to EP5, these bits are reserved bits.  
"0" is read out from the bit. No effect on writing.

**EP1:[bit8 to bit0] PKS: Packet size display bits**

**EP2 to EP5:[bit6 to bit0] PKS: Packet size display bits**

It displays the number of data bytes that have been written into the receive buffer when OUT packet transfer for EP1 to EP5 has been completed. The SIZE bit is updated to a valid value when the interrupt factor for DRQ in the EP1 to EP5 status register (EP1S to EP5S) has been set.

The following lists each maximum number of transfer data for EndPoint1 to 5.

EndPoint	Max. number of transfer	Range of display
1	256 Bytes	000 <sub>H</sub> to 100 <sub>H</sub>
2 to 5	64 Bytes	00 <sub>H</sub> to 40 <sub>H</sub>

Note:

Since the SIZE bit is set to the number of pieces of data that are written into the buffer from the HOST for the OUT direction transfer, any value read from the SIZE bit when IN direction is under way has no meaning.

## MB90330A Series

## 13.3.10 EP0 to EP5 Data Register (EP0DT to EP5DT)

The EP0 to EP5 data registers (EP0DT to EP5DT) are access registers used to read or write into the transmission/receive buffer for transfer data related to EndPoint0 to EndPoint5.

### ■ EP0 to EP5 Data Register (EP0DT to EP5DT)

Figure 13.3-12 shows the bit configurations of the EP0 to EP5 data registers (EP0DT to EP5DT).

**Figure 13.3-12 EP0 to EP5 Data Register (EP0DT to EP5DT)**

Address	bit	7	6	5	4	3	2	1	0	
		BFD								
EP0DT 0000F0 <sub>H</sub>		X	X	X	X	X	X	X	X	← Initial value
EP1DT 0000F2 <sub>H</sub>		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	← Access
EP2DT 0000F4 <sub>H</sub>										
EP3DT 0000F6 <sub>H</sub>										
EP4DT 0000F8 <sub>H</sub>										
EP5DT 0000FA <sub>H</sub>										

Address	bit	15	14	13	12	11	10	9	8	
		BFD								
EP0DT 0000F1 <sub>H</sub>		X	X	X	X	X	X	X	X	← Initial value
EP1DT 0000F3 <sub>H</sub>		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	← Access
EP2DT 0000F5 <sub>H</sub>										
EP3DT 0000F7 <sub>H</sub>										
EP4DT 0000F9 <sub>H</sub>										
EP5DT 0000FB <sub>H</sub>										

R/W : Readable/Writable

The following describes the function of each bit in the EP0 to EP5 data registers (EP0DT to EP5DT).

#### [bit15 to bit0] BFD: EndPoint transmission/receive buffer data bits

It is a data read and data write register for the transmission/receive buffer for each EndPoint. Access to the BFD register via DMA transfer is supported on a word access only. If you transfer the odd number of pieces of data through DMA transfer, you can do so by setting a byte transfer for the last data transfer. If you perform word transfer via CPU access, the last transfer must be byte transfer the same way as in DMA transfer.

#### Note:

CPU access to the EP0DT to EP5DT registers are possible both on a per-byte basis and on a per-word basis, and if you byte access any of the registers, first, access bit7 to bit0, then access bit15 to bit8, and subsequently access the high-order and low-order alternately. Accessing the EP0DT to EP5DT registers via bit instruction is prohibited.

## **13.4 Operation Explanation of USB Function**

---

**The USB Function conforms to the USB (Universal Serial Bus) communication protocol and supports basic protocol operations (handshake) by hardware. Consequently, only processing communication data can provide the USB communication.**

---

### **■ Operation of USB Function**

The USB Function performs two-way packet transfer with a host controller that supports the USB protocol. A host PC and its devices are connected and configured through enumeration. Then, communications based on various types of transfers using device drivers are performed.

This section describes the operation of the USB communications between a host PC and its devices by using enumeration as an example.

It illustrates the operations of registers and USB packets to understand the overview of USB communication processing.

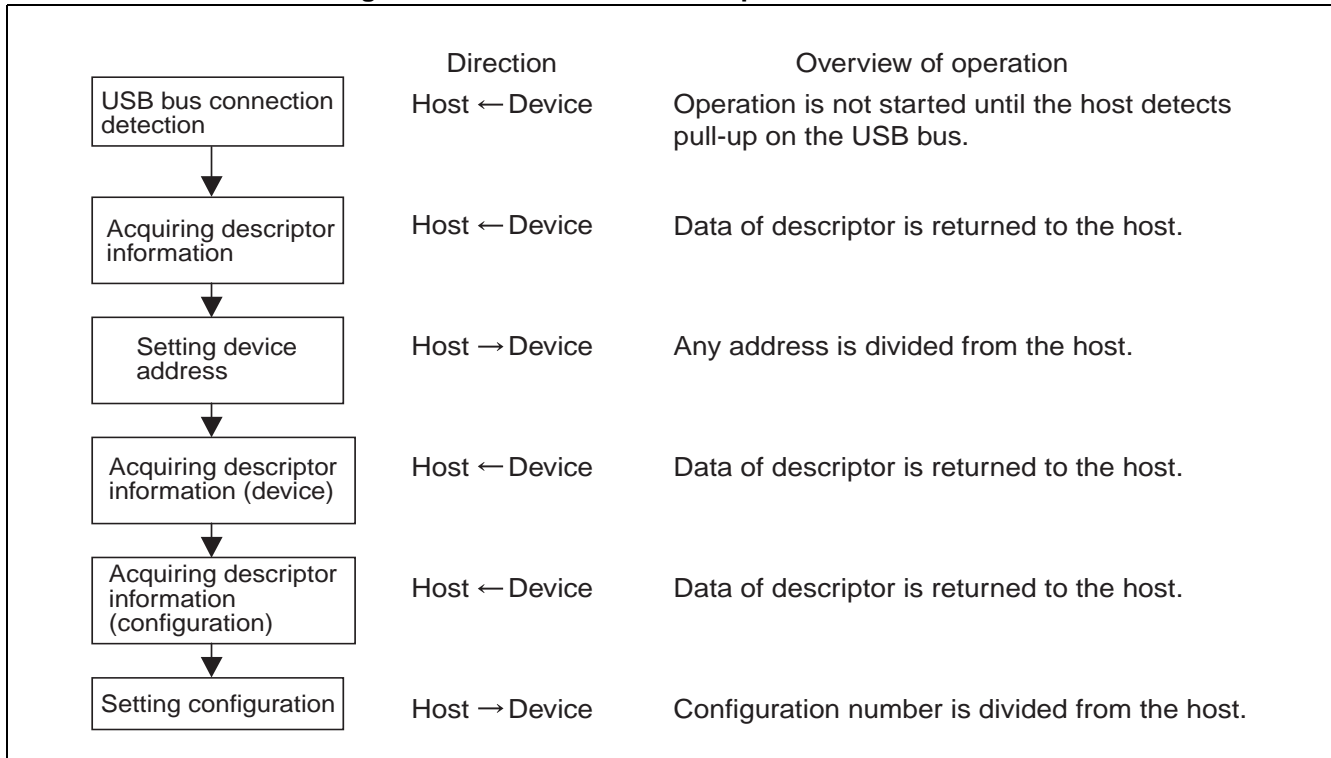
#### **● Enumeration process**

It is the first process that establishes the connection between a host PC and its device before the USB can operate. The host PC examines which devices are connected to the USB bus by using USB control transfer (USB transfer type). This uses EP0 (EndPoint0) out of six endpoints (as defined in the USB specifications) (USB Specification).

When EP1 to EP5 are used, the following must be received on the USB bus.

1. USB bus reset
2. Address set by SET\_Address
3. Configuration set by SET\_Config

Figure 13.4-1 Connection Example of USB Cable Pin



#### ● Detecting a connection

The device notifies the host PC.

The host monitors the two signals (D+ and D-) on the USB bus and detects a device connection if either signal goes to the "H" level.

For the detailed procedure in the case of a self-powered device, see "13.4.1 Detecting Connection and Disconnection". In the case of a bus-powered device, perform the operation described in "● Example Register Initialization and Operation Startup Procedure".

#### ● Example Register Initialization and Operation Startup Procedure

The following is an example of how to initialize the registers and start operation.

1. Set EP0 in the EP0C register (packet size, etc.)
2. Set the EPEN, DIR, and TYPE settings for each EP in the EP1C to EP5C registers.
3. Clear the RST bit in the UDCC register
4. Clear BFINI in the EP0IS, EP0OS, and EP1S to EP5S registers
5. Clear the  $\overline{\text{HCON}}$  bit in the UDCC register

#### ● USB bus reset

A bus reset is applied from the host PC to the device and the USB device core is initialized.

A device must perform processing in the following steps. (The first bus reset after USB has been connected does not need any processing.)

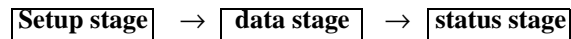
1. Initializes the USB Function with RST in the UDCC register.
2. Set transmission/receive buffers and related control registers again it will use.
3. Return firmware control to the state prior to enumeration.

● Getting descriptor

The device receives a request from the host PC and sends data to the host.

In more detail, communications are performed in the following three stages:

**Figure 13.4-2 Communication Stage**



The setup stage ensures that the device receives normal packets from the host PC and identify the command by decoding it. In addition, the next data stage prepares information on a descriptor to be sent back in a transmission buffer. The data stage simply confirms that normal data is sent from the host PC. The status stage performs end processing when the host PC sends a packet without data.





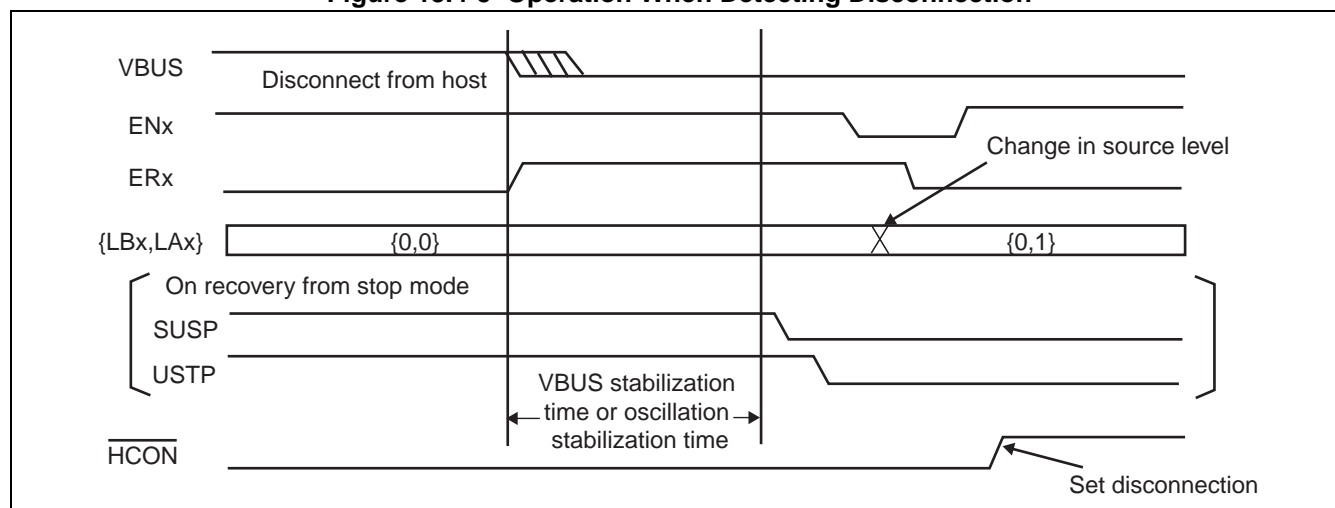
3. Temporarily disable the external interrupt. Change the interrupt setting to detect "L" level inputs to the external interrupt pin, and then clear and re-enable the external interrupt.
4. Perform initialization (complete initialization including the USB Function Register). See "Example Register Initialization and Operation Startup Procedure".
5. Clear the  $\overline{\text{HCON}}$  bit in the UDCC register and connect the D+ pull-up resistor. (Clear the  $\overline{\text{HCON}}$  bit even if not performing control of the pull-up resistor.)

**Note:**

You do not need to allow for the above VBUS stabilization time in your program if using an external noise filter on the external interrupt pin.

● Detecting disconnection

**Figure 13.4-5 Operation When Detecting Disconnection**



The device uses the following sequence to detect a disconnection from the host PC.

1. Detection of an "L" level on the external interrupt pin connected to VBUS indicates disconnection from the USB host.
2. On recovery from stop mode:  
 After waiting for the oscillation stabilization time, clear SUSP in the UDCCS register followed by USTP in the UDCC register.  
 When not recovering from stop mode:  
 Wait for the VBUS stabilization time.
3. Temporarily disable the external interrupt. Change the interrupt setting to detect "H" level inputs to the external interrupt pin, and then clear and re-enable the external interrupt.
4. Set the  $\overline{\text{HCON}}$  bit in the UDCC register and disconnect the D+ pull-up resistor. (Set the  $\overline{\text{HCON}}$  bit even if not performing control of the pull-up resistor.)

**Note:**

You do not need to allow for the above VBUS stabilization time in your program if using an external noise filter on the external interrupt pin.

## MB90330A Series

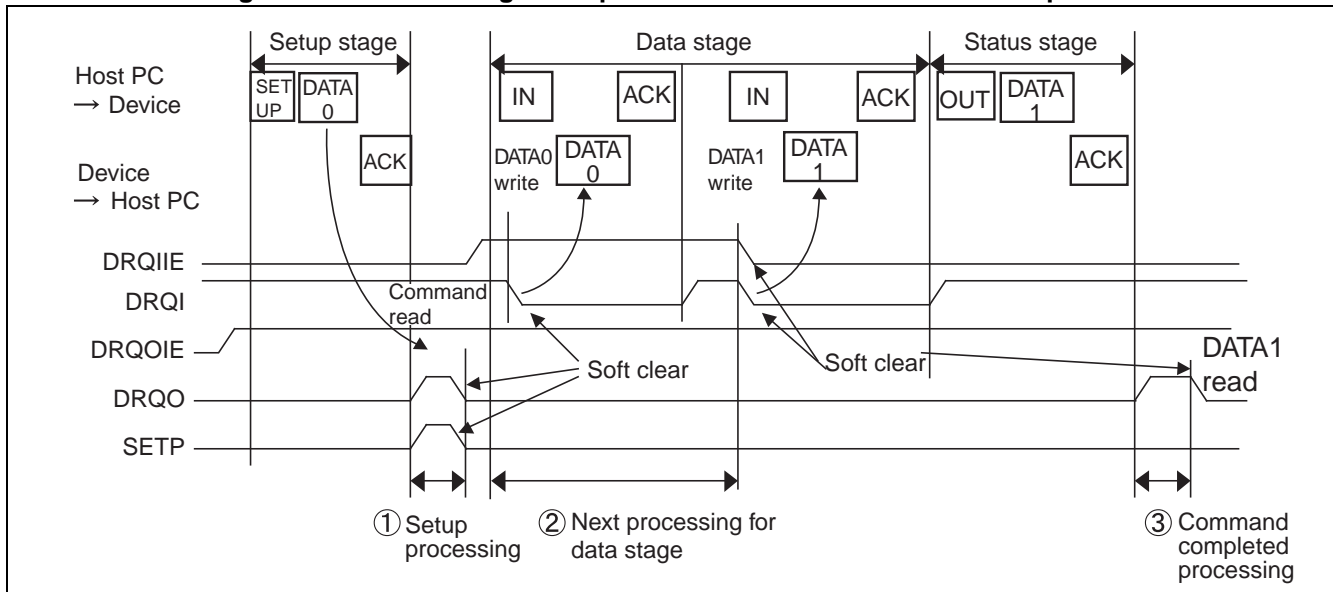
### 13.4.2 Each Register Operation when Command Responds

This section describes basic operations and control of registers and then how to process USB packets (architecture). Firmware tasks triggered via CPU interrupt are processed for each handshake operation. This is equivalent to processing each packet on a per-stage basis.

#### ■ Each Register Operation when Read Command Responds

For GetDescriptor, SynchFrame, and the class vendor command

**Figure 13.4-6 Each Register Operation when Read Command Responds**



#### ● Set-up processing

When the setup stage is received, DRQO is set. If the DRQO is set, CPU interrupt is raised and the SETP flag is confirmed. It reads as many commands as necessary in the receive buffer if the DRQO is set (which does not mean all eight bytes need to be read), decode the commands, performs setting tasks, and returns to a point where a process was interrupted after it clears the SETP flag and DRQO interrupt cause.

#### ● Data stage processing

If the data stage indicates IN direction as a result of a decoded command, it enables the DRQIIE (as the interrupt cause DRQI has the initial value of "1", it only sets an interrupt to be enabled), and transfers transmission data to the transmission buffer triggered by an CPU interrupt. When transfer has been completed, it clears the interrupt cause DRQI before returning to the interrupted point.

The DRQI is set when the data packet toward IN has been completed. The CPU interrupt is entered when the DRQI is set, the transfer data is transferred to the transmission buffer to prepare for the next data packet. When transfer has been completed, it clears the interrupt cause DRQI before returning to the interrupted point.

#### Note:

This USB function is not compatible with the newly added commands for USB 2.0; for GetDescriptor command, respond with the USB 1.1.

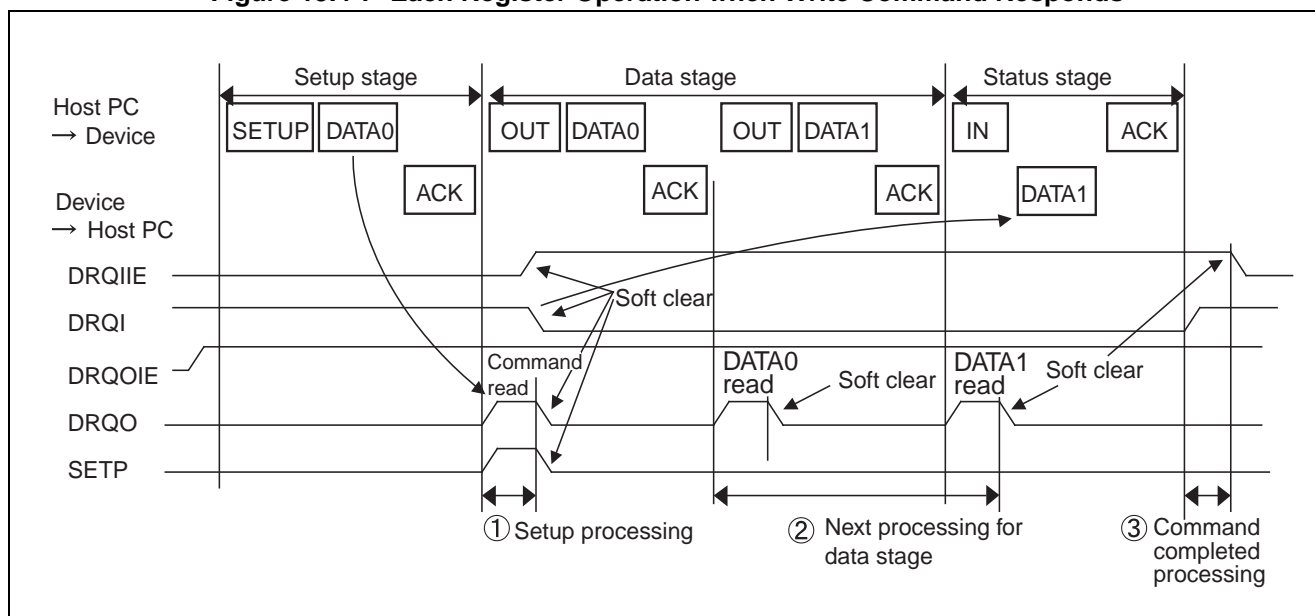
● Command completion processing

The DRQI is set when the status stage toward OUT has been completed. It enters a CPU interrupt process when the DRQO is set, confirms that the number of received data is 0, and clears the interrupt cause DRQO and returns to the interrupted point to prepare for the next setup stage.

■ Each Register Operation when Write Command Responds

For GetDescriptor and the class vendor command

Figure 13.4-7 Each Register Operation when Write Command Responds



● Set-up processing

When the setup stage is received, DRQO is set. If the DRQO is set, CPU interrupt is raised and the SETP flag is confirmed. It reads as many commands as necessary in the receive buffer if the DRQO is set (which does not mean all eight bytes need to be read), decode the commands, performs setting tasks. It clears the DRQI (the interrupt cause DRQI due to the initial value of "1") without writing data to the transmission buffer to prepare for a 0-byte response in the status stage, and sets the DRQIE to confirm the successful completion of the status stage. It also clears the SETP flag and the DRQO interrupt cause before returning from the interrupt to the interrupted point.

● Data stage processing

The DRQO is set when the data stage toward OUT has been completed. It enters a CPU interrupt process when the DRQO is set, and first, confirms SIZE of the EP0 status register, and then activates DMA as many times as required for the received number of pieces of data or reads data from the receive buffer via CPU read. Then, it clears the DRQO interrupt cause before returning from the interrupt to the interrupted point.

● Command completion processing

The DRQI is set when the status stage toward IN has been completed. It enters a CPU interrupt process when the DRQI is set, and can confirm that the status stage has been successfully completed. Then, it clears the DRQIE interrupt enable before returning to the interrupted point.

### 13.4.3 STALL Response and Release

For Endpoint0 and For Endpoints 1 to 5, this section explains STALL response and release procedures.

#### ■ STALL response and release procedures for Endpoint0

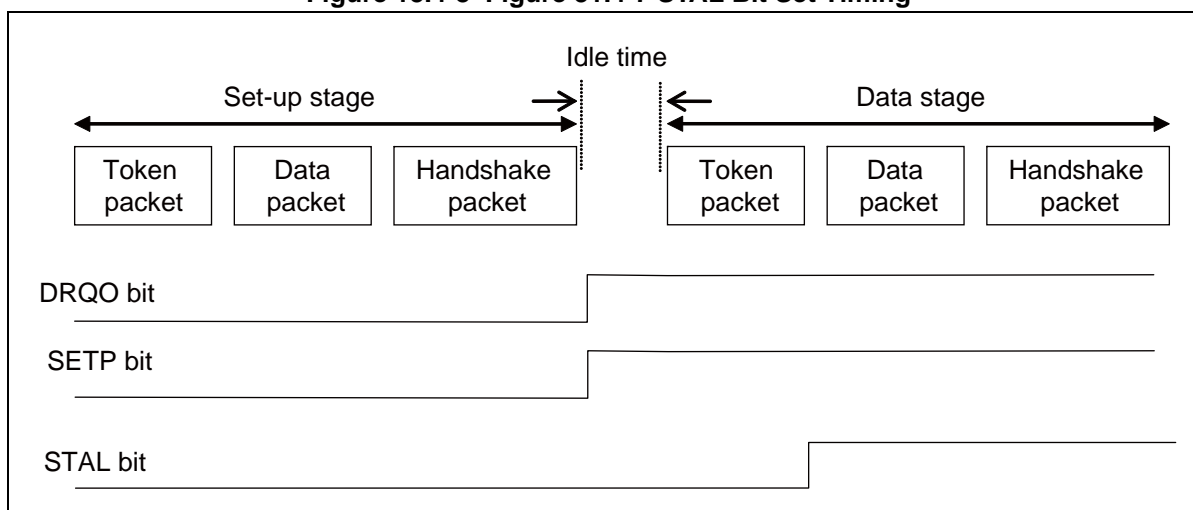
STALL response and release procedures for Endpoint0 are executed with STAL bit of EP0 Control Register (EP0C).

- Set timing of STAL bit

For STALL response, interprets the command at detecting SETP bit of "1" (DRQO bit = 1 for interrupt) that indicates the set-up stage of control transfer. (See Figure 13.4-8.)

After setting STAL bit, clear interrupt cause (DRQO bit).

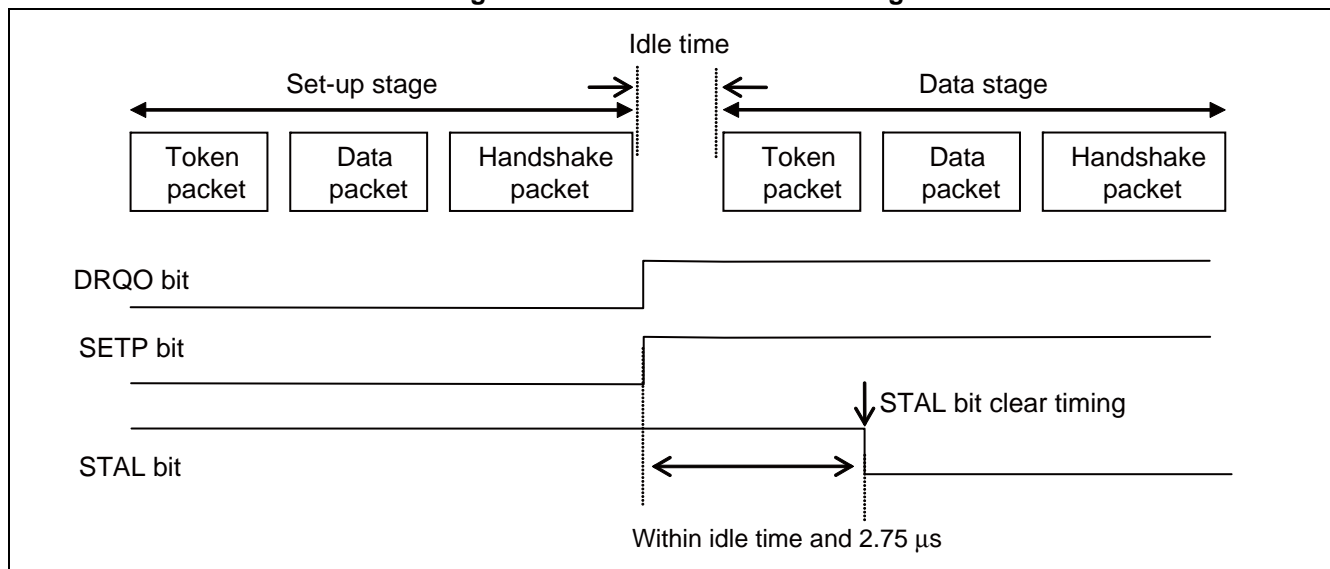
Figure 13.4-8 Figure 31.4-7 STAL Bit Set Timing



- STALL Bit Clear Timing

For STALL release, clears STAL bit at detecting SETP bit of "1" (DRQO bit = 1 for interrupt) that indicates the set-up stage of control transfer, and sets STAL bit if the STALL response is required. (See Figure 13.4-9.)

**Figure 13.4-9 STAL Bit Clear Timing**



For STALL response release (STAL bit clear), clear STAL bit the period between the time when SETP bit of "1" (DRQO bit= 1 for interrupt) is detected and the time when the data packet transmission/reception of the next data stage is started. The period between the time when DRQO becomes "1" and the time when STAL bit is cleared is as follows: (Transfer speed: at Full speed of 12Mbps) When STAL bit is not cleared in the following period, execute STAL response with the handshake of data stage.

The period between the time when DRQO bit of "1" is detected and the time when STAL bit is cleared: within idle time + 2.75 μs

\* When idle time is the shortest period of 2-bit transfer time, the above period is within about 2.9 μs.

If the STAL bit clear cannot executed within the above period, take an appropriate countermeasures such as lengthening of the idle time with a driver of USB host.

## ■ STALL response /release of Endpoints 1 to 5

STALL response /release of Endpoints 1 to 5 are controlled with Control registers of EP1 to EP5 and internal condition bit

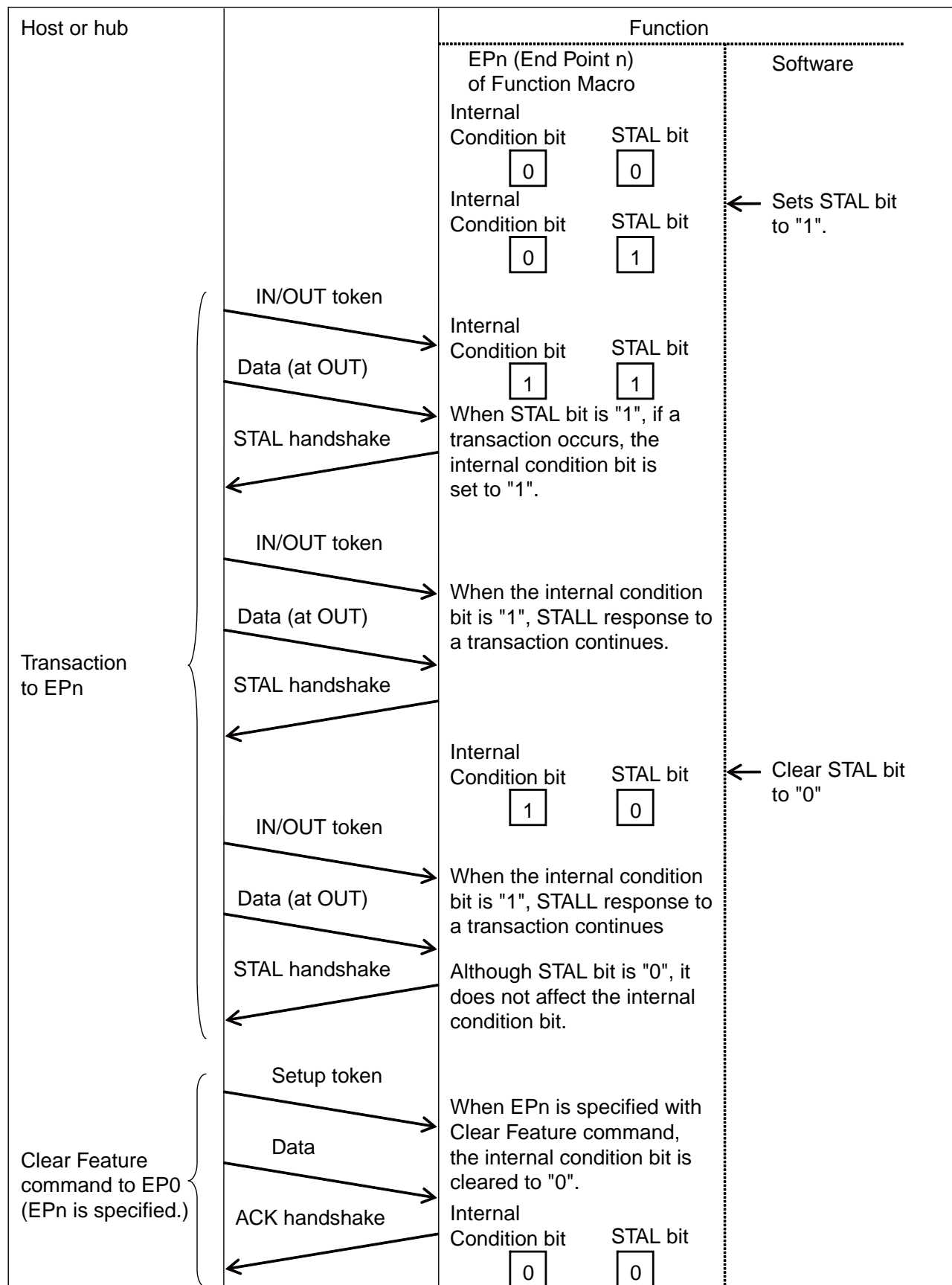
- To execute STAL response with software

The procedures to execute STAL response with software are shown in Figure 13.4-10. To execute STAL response, set STAL bit of the relevant endpoint with software. In this time, the internal condition bit does not change. Furthermore, when a host generates a transaction to the endpoint where STAL bit is set, hardware would automatically set the internal condition bit of the relevant endpoint and gives STALL response to the host.

Once the internal condition bit is set, the internal condition bit has been set and continues STAL response until Clear Feature command is issued from the host despite of the clearing of STAL bit.

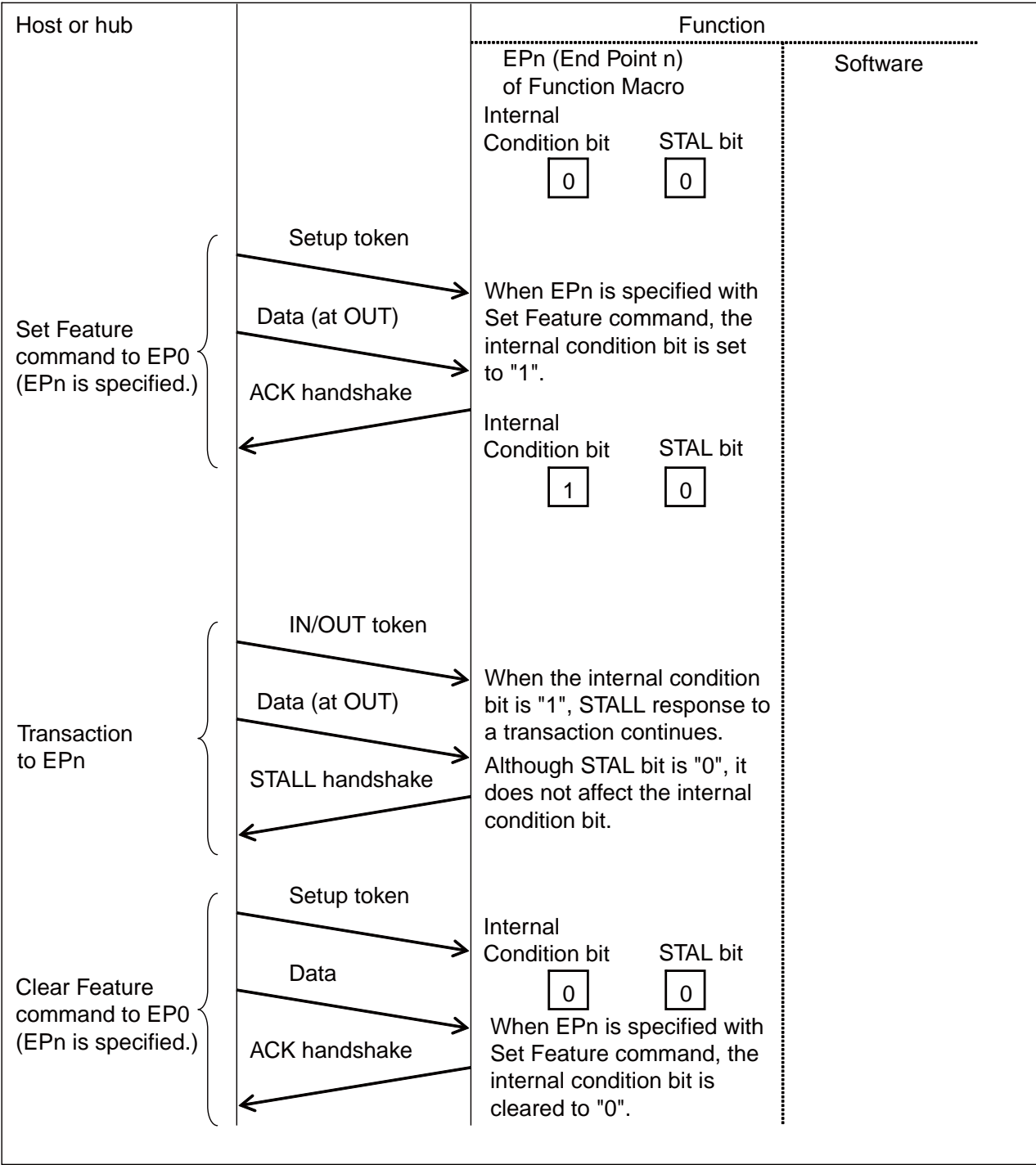
As long as the STAL bit is set, STAL bit response continues even if the internal condition bit is cleared with Clear Feature command because the internal condition bit is set every time a transaction to the relevant endpoint occurs. Therefore, to release the STAL response, be sure to clear STAL bit and the internal condition bit with the Clear Feature command.

Figure 13.4-10 Case where STALL response is executed with software processing



- To automatically execute STALL response with hardware.
- The procedures to execute STALL response with hardware are shown in Figure 13.4-11. When STALL response is set with Set Feature command, the hardware would automatically set the internal condition bit of the relevant endpoint and gives the STALL response regardless of the STAL bit. Once the internal condition bit is set, the internal condition bit has been held until Clear Feature command is issued from the host to clear the internal condition bit regardless of STAL bit. After the relevant bit is cleared with Clear Feature command, STAL bit is referenced. Therefore, to clear STALL response, be sure to clear the internal condition bit with Clear Feature command.

Figure 13.4-11 Case where STALL response is executed with hardware automatically



## MB90330A Series

### 13.4.4 Suspend Function

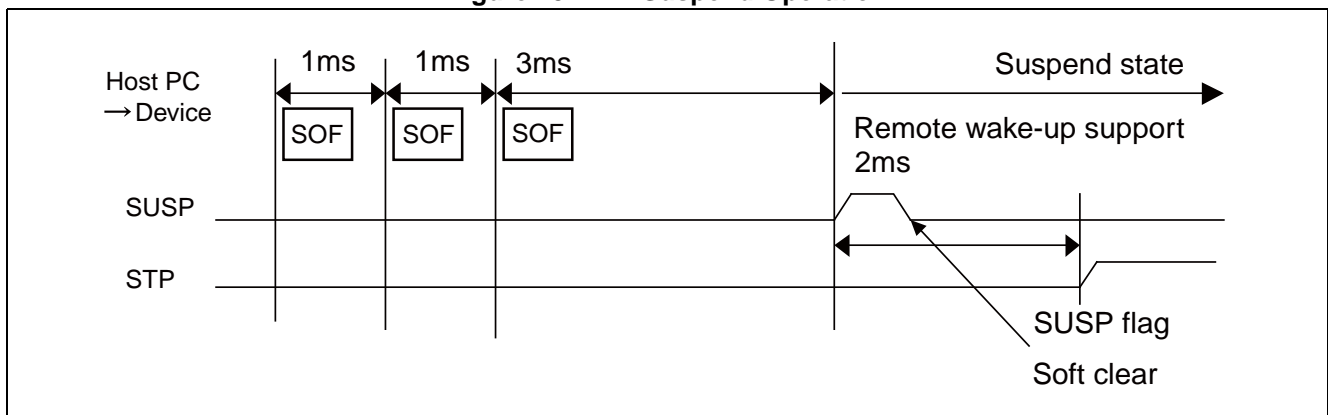
A USB device must have a configuration of bus power supply where power consumption is 500 $\mu$ A or less in suspend status. The section covers a USB device from its transmitting to suspend status to its entering STOP mode.

#### ■ Suspend Processing

When the USB device core detects suspend status, SUSP of the UDCS register is set to be enabled.

The following shows an example of suspend operation:

Figure 13.4-12 Suspend Operation



#### ● Suspend processing

The USB Function determines that it detects suspend status when there is no operation for not less than 3 ms on the USB bus, and the SUSP interrupt cause in the UDCS register is set. If a USB device supports remote wake-up, the USB Function waits another 2 ms (which blocks remote wake-up during this time period), and set the device to stop mode.



## 13.4.5 Wake-up Function

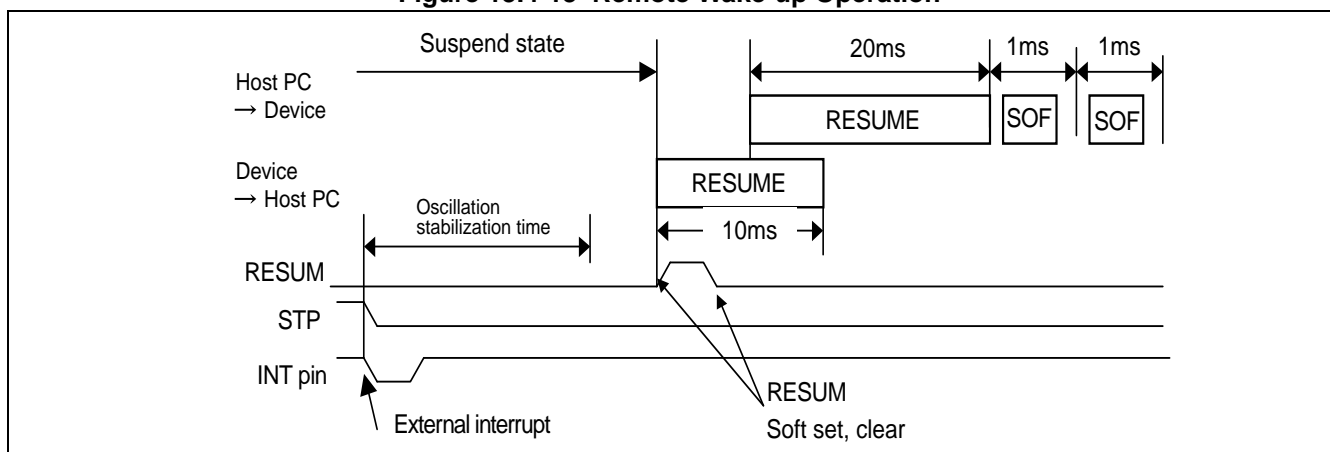
To shift a USB device from suspend status to wake-up status, the USB protocol provides the following two ways:

- Remote wake-up from device
- Wake-up from host PC

The above is explained.

### ■ Remote Wake-up

Figure 13.4-13 Remote Wake-up Operation

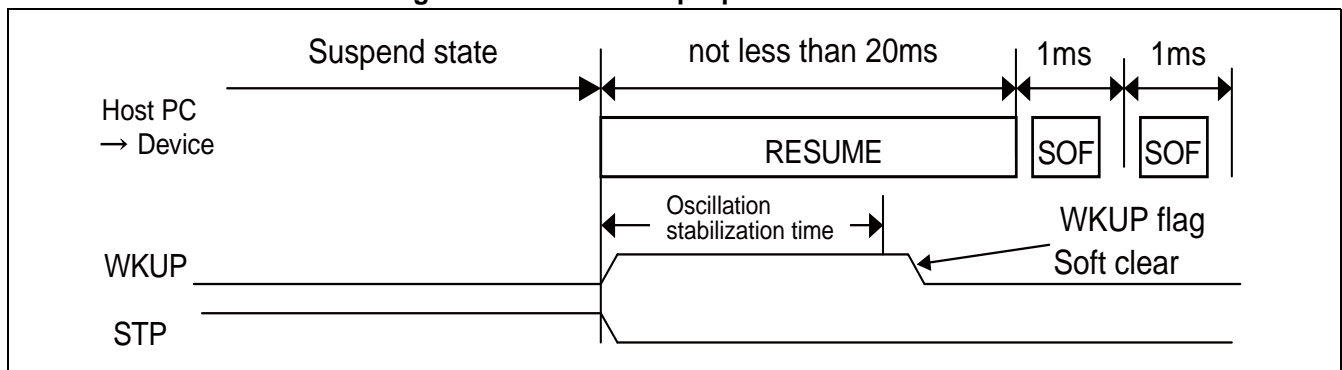


A device must perform processing in the following steps:

1. Recover the device from stop mode through external interrupt.
2. RESUM of the UDCC register is set.
3. RESUM of the UDCC register is cleared.

### ■ Wake-up from Host

Figure 13.4-14 Wake-up Operation from Host



For the devices, the following processing is required:

1. Set oscillation stabilizing time to be not more than 10 ms.
2. Enter an interrupt due to the WKUP factor and clear WKUP of UDCC that is the interrupt cause and return from the interrupt.

## MB90330A Series

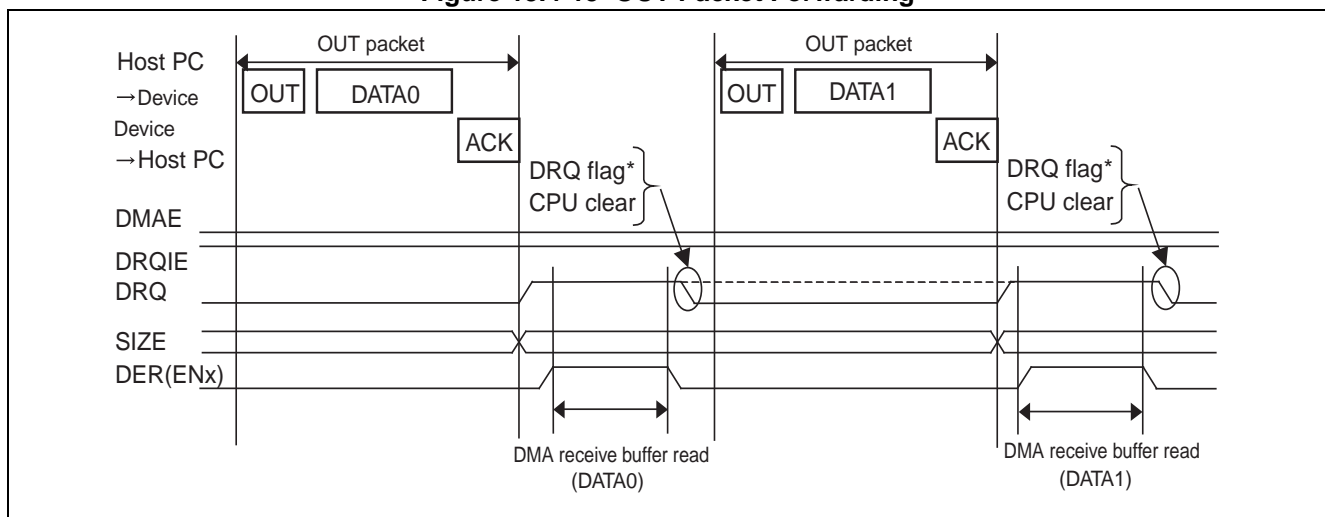
## 13.4.6 DMA Transfer Function

It is possible to transfer data between transmission/receive buffer and internal RAM that the USB Function communicates. You can select the following two modes in DMA transfer: one is packet transfer mode where data is transferred based on the number of pieces of transfer set on a per-packet basis and another is data number automatic transfer mode where all data is transferred based on the number of pieces of data specified once. Each DMA transfer mode is explained.

### ■ Packet Transfer Mode

- The packet transfer mode performs transfer by setting the number of pieces of transfer per packet in DMA and clearing the interrupt cause when transfer has been completed. The transfer mode can access any buffer in each endpoint.
- Timing by which the buffer is accessed in OUT direction and IN direction is shown as follows.
- OUT direction (host PC → device) forwarding

Figure 13.4-15 OUT Packet Forwarding



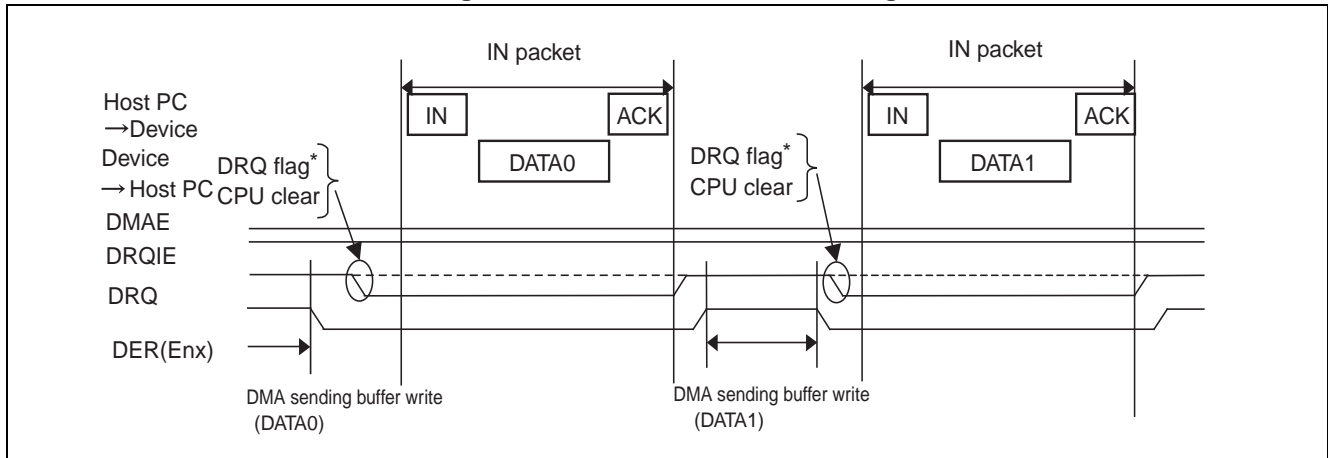
In OUT- direction transfer, a USB device must perform processes in the follows steps:

1. It confirms the number of pieces of transfer data when the DRQ flag is set and the interrupt process is entered.
2. It sets the number of pieces of transfer data in the data counter register DDCT of DMA, enables DMA with the DER register, and start transfer.
3. Once transfer has been completed, it clears the corresponding DRQ flag in the EP1S to EP5S registers and the corresponding interrupt factor flag in the DSR register of  $\mu$ DMAC and returns from the interrupt process.

\*: EP1 to EP5 consists of the double buffers, it can be cleared only when one buffer that is not being accessed is empty and data is read from another buffer being accessed and cannot be cleared even though "0" is written to it if one buffer that is not being accessed has data left to be read (Dotted line status). It continuously enters the DRQ interrupt process.

● IN direction (host PC → device) forwarding

Figure 13.4-16 IN Packet Forwarding



In IN- direction transfer, a USB device must perform processes in the follows steps:

1. When the DRQ flag is set and enters the interrupt process, it sets the number of pieces of data to be transferred in an IN packet in the data counter register DDCT of DMA, enables DMA with the DER register, and start transfer.
  2. Once DMA transfer has been completed, it clears the corresponding DRQ flag in the EP1S to EP5S registers and the corresponding interrupt factor flag in the DSR register of  $\mu$ DMAC and returns from the interrupt process.
- \*: EP1 to EP5 consists of the double buffers, it can be cleared only when one buffer that is not being accessed has already data written into it and data is written into another buffer that is being accessed and cannot be cleared even though "0" is written to it when one buffer that is not being accessed is empty (Dotted line status). It continuously enters the DRQ interrupt process.

It sets the total number of pieces of data to be transferred in DMA and sets the transfer enable bit in advance. When DMAE is enabled and the DRQ is set after transfer from the HOST has been completed, the interrupt cause is automatically cleared when data whose number of pieces is equal to the PKS in the EP1 to EP5 control registers (EPxC) has been transferred (Whether the DRQ flag is actually cleared depends on the fact that both buffers in a double buffer are empty or full). Subsequently, when transfer from the HOST has been completed, repeat the similar process until data equivalent to the number of pieces of transfer data predefined in DMA. has been transferred. Meanwhile, any intervention from CPU is not required, and transfer will be completed with only one setting, which is the automatic transfer mode. If the device performs the next transfer, it sets  $\mu$ DMAC again and enables DMA when a CPU interrupt is raised when the last data has been transferred, and returns from the CPU interrupt. Since the data number automatic transfer mode is used for DMAE=1, only buffer access to endpoint 1 to endpoint5 is enabled. Timing by which the buffer is accessed in OUT direction and IN direction is shown as follows.

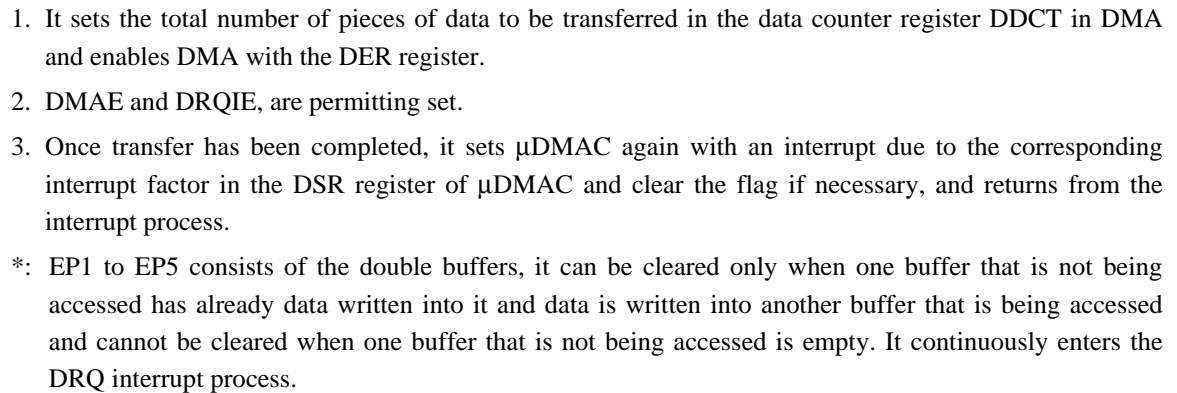
The diagram illustrates the timing of a DMA receive buffer read operation. It shows two packets, DATA0 and DATA1, being read from the buffer. The signals involved are Host PC, Device, DMAE, DRQ, SIZE, and DER(Enx).

- Host PC → Device:** OUT, DATA0, ACK, OUT, DATA1, ACK.
- Device → Host PC:** DRQ flag\* Automatic clear (indicated by a bracket).
- DMAE:** Signal transitions from high to low at the start of each packet read.
- DRQ:** Signal transitions from low to high at the start of each packet read.
- SIZE:** Signal transitions from high to low at the start of each packet read.
- DER(Enx):** Signal transitions from high to low at the start of each packet read.

The diagram is divided into two sections: "Read PKS part of DMA receive buffer" and "Read the rest of DMA receive buffer".

1. It sets the total number of pieces of data to be transferred in the data counter register DDCT in DMA and enables DMA with the DER register.
2. DMAE and DRQIE, are permitting set.
3. Once transfer has been completed, it sets  $\mu$ DMAC again with an interrupt due to the corresponding interrupt factor in the DSR register of  $\mu$ DMAC and clear the flag if necessary, and returns from the interrupt process.

313



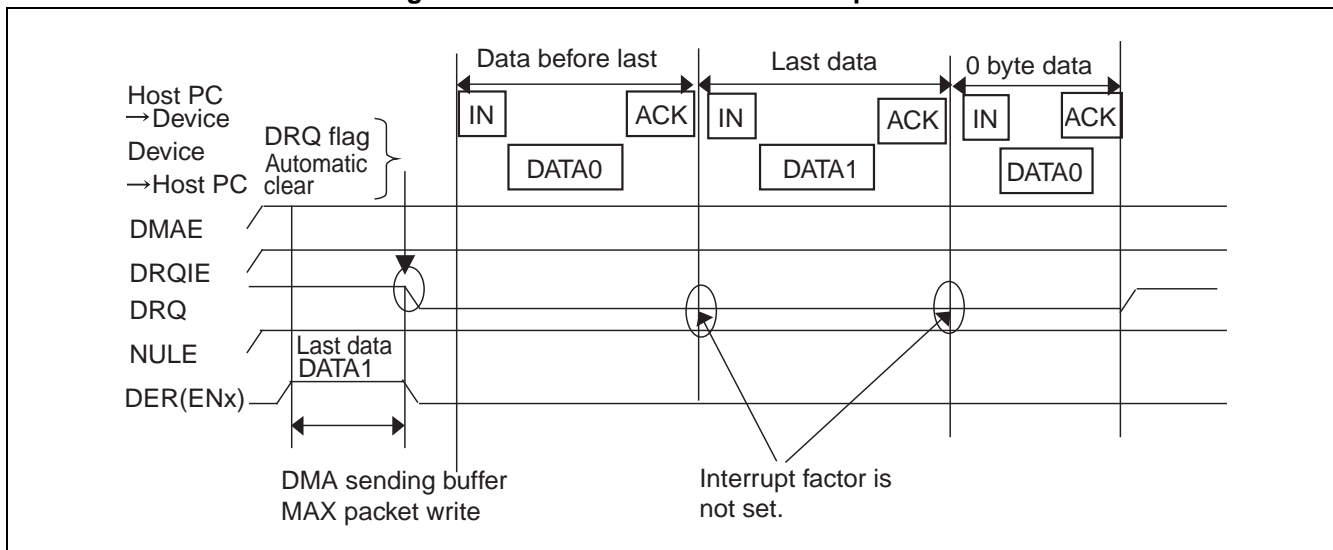
### 13.4.7 NULL Transfer Function

If data sent from the USB Function is the last packet and a maximum number of packets, it is possible to automatically transfer 0-byte data in the next packet transfer. The NULL transfer function requires that DMAE is enabled and is a function that is only valid for IN transfer.

#### ■ NULL Transfer Mode

- This is the mode where if the automatic buffer transfer mode is set (DMAE=1) and IN-direction data transfer request arrives, and a maximum number of packets are written via DMA and the last data write decrements the number of DMA count data to "0", it automatically sets 0-byte data transfer and will send 0-byte data for the next IN-direction data transfer request when the last IN-direction data transfer request from the HOST has been received. The DRQ interrupt flag is not set until 0-byte data is read from the HOST after the last data was written into a buffer via DMA. Timing by which the buffer is accessed is shown as follows.
- Only IN direction (device → host PC) forwarding

Figure 13.4-19 NULL Data Transfer Operation



For the devices, the following processing is required:

DMAE, DRQIE, and NULE are permitting set.



# ***CHAPTER 14***

---

## ***USB HOST***

**This chapter describes the functions and operation of USB HOST.**

- 14.1 Feature of USB HOST
- 14.2 Restriction on USB HOST
- 14.3 Block Diagram of USB HOST
- 14.4 Register of USB HOST
- 14.5 Operation of USB HOST
- 14.6 Each Token Flow Chart of USB HOST



## **14.1 Feature of USB HOST**

---

**USB HOST provides minimum host operations required and is a function that enables data to be transferred to and from Device without PC intervention.**

---

### **■ Feature of USB HOST**

USB HOST has not only original function as the USB host but also the USB function by switching operation mode.

USB HOST has the following features.

- Automatic detection of Low Speed/Full Speed forwarding
- Low Speed/Full Speed forwarding support
- Automatic detection of connection and cutting device
- Reset sending function support to USB bus
- Support of IN/OUT/SETUP/SOF token
- Automatic transmission of handshake packet for IN token (excluding STALL)
- Handshake packet automatic detection at OUT token
- Supports a maximum packet length of 256 bytes.
- Various error (CRC error/toggle error/time-out) supports
- Wake Up function support

## 14.2 Restriction on USB HOST

This section indicates Restriction on USB HOST.

### ■ Restriction on USB HOST

		HOST
Support Hub		○ *
Transfer	Bulk transfer	○
	Control transfer	○
	Interrupt transfer	○
	Isochronous transfer	×
Transfer speed	Low Speed	○
	Full Speed	○
PRE packet support		×
SOF packet support		○
Error	CRC error	○
	Toggle error	○
	Time-out	○
	Max. packet < Receive Data	○
Detection of connection and cutting of device		○
Transfer speed detection		○

○: Supported

×: Not supported

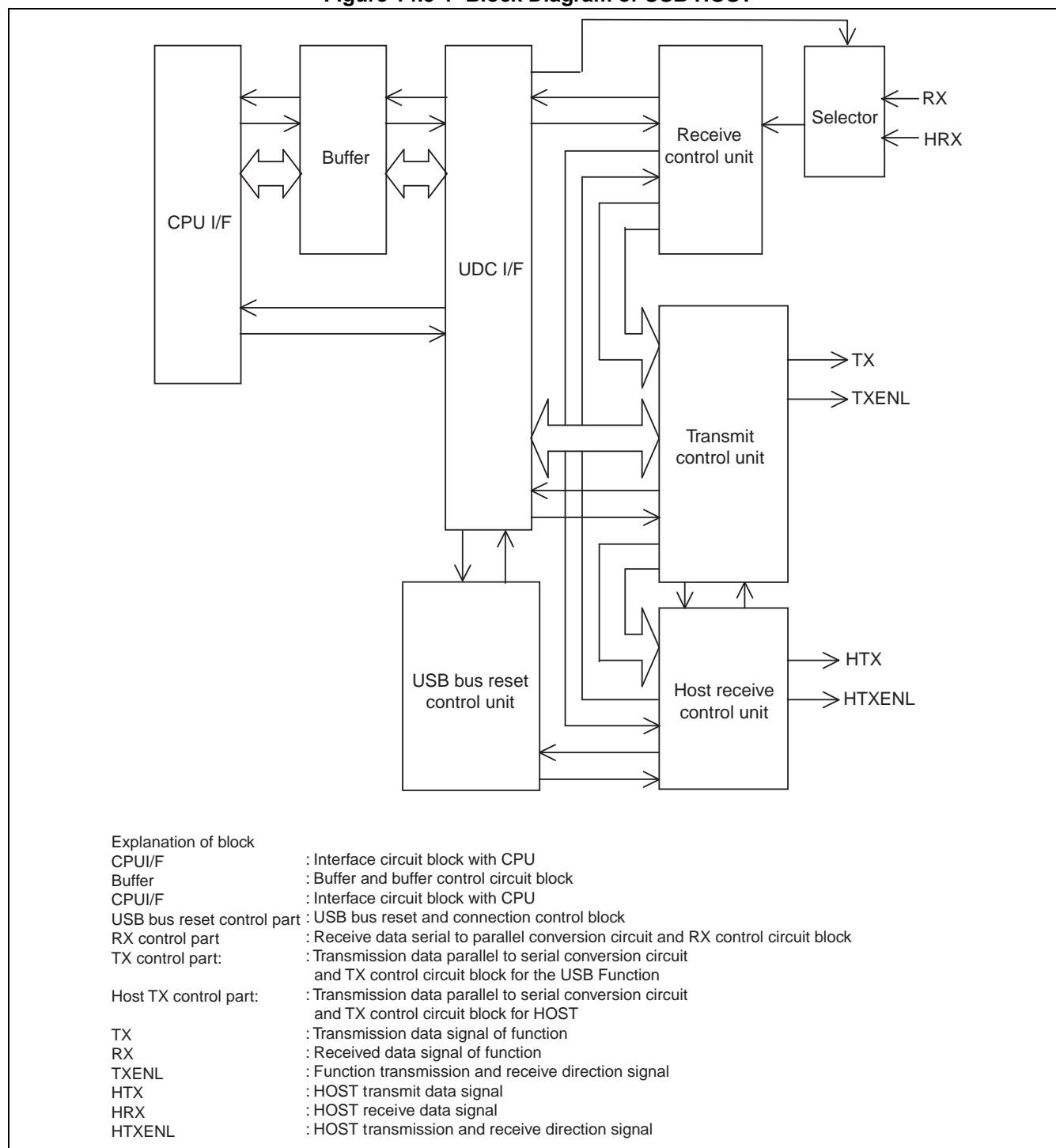
\* : Only Full Speed corresponds, and HUB is a support up to one step.

## 14.3 Block Diagram of USB HOST

Figure 14.3-1 shows the block diagram of USB HOST.

### ■ UART Block Diagram of USB HOST

Figure 14.3-1 Block Diagram of USB HOST



## 14.4 Register of USB HOST

In USB HOST, there are the following ten types of registers:

- Host control register 0,1(HCNT0/HCNT1)
- Host interruption register (HIRQ)
- Host error status register (HERR)
- Host state status register (HSTATE)
- SOF interruption FRAME comparison register (HFCOMP)
- Retry timer setting register (HRTIMER)
- HOST address register (HADR)
- EOF setting register (HEOF)
- FRAME setting register (HFRAME)
- Host token end point register (HTOKEN)

### ■ Register of USB HOST

• Host control register 0								
bit	7	6	5	4	3	2	1	0
Address: 0000C0 <sub>H</sub>	RWKIRE	URIRES	CMPIRES	CNNIRES	DIRE	SOFIRE	URST	HOST
Read/Write →	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value →	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
• Host control register 1								
bit	15	14	13	12	11	10	9	8
Address: 0000C1 <sub>H</sub>	Reserved	Reserved	Reserved	Reserved	Reserved	SOFSTEP	CANCEL	RETRY
Read/Write →	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value →	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(1)
• Host interruption register								
bit	7	6	5	4	3	2	1	0
Address: 0000C2 <sub>H</sub>	TCAN	Reserved	RWKIRQ	URIRQ	CMPIRQ	CNNIRQ	DIRQ	SOFIRQ
Read/Write →	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value →	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
• Host error status register								
bit	15	14	13	12	11	10	9	8
Address: 0000C3 <sub>H</sub>	LSTSOF	RERR	TOUT	CRC	TGERR	STUFF	HS	
Read/Write →	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value →	(0)	(0)	(0)	(0)	(0)	(0)	(11 <sub>B</sub> )	

(Continued)





## 14.4.1 Host Control Register 0,1(HCNT0/HCNT1)

Host control registers 0,1(HCNT0/HCNT1) specify the USB operation mode and the settings of an interrupt.

### ■ Host Control Register 0, 1(HCNT0/HCNT1)

Figure 14.4-1 Bit Configuration of Host Control Register 0, 1 (HCNT0/HCNT1)

Host control register 0								
bit	7	6	5	4	3	2	1	0
Address: 0000C0 <sub>H</sub>	RWKIRE	URIRES	CMPIRES	CNNIRES	DIRE	SOFIRE	URST	HOST
Read/Write →	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value →	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
Reset On/Off at UDCC RST bit →	(X)	(X)	(X)	(X)	(X)	(X)	(○)	(X)
Host control register 1								
bit	15	14	13	12	11	10	9	8
Address: 0000C1 <sub>H</sub>	Reserved	Reserved	Reserved	Reserved	Reserved	SOFSTEP	CANCEL	RETRY
Read/Write →	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value →	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(1)
Reset On/Off at UDCC RST bit →	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

#### [bit 15 to bit 11] Reserved

It is reserved bit.

Be sure to set this bit to "0".

#### [bit 10] SOFSTEP: SOF interrupt condition selection

It sets whether an interrupt due to SOF is generated every time SOF is executed. The interrupt is enabled when the SOFIRE bit in host control register 0 (HCNT0) is "1".

When it is "0", the interrupt is generated via the setting of the SOF interrupt FRAME comparison register (HFCOMP), and when it is "1", the interrupt is unconditionally generated every time SOF is executed. However, the interruption is not generated at the first SOF token. It is not initialized with the RST bit in the UDC control register (UDCC).

SOFSTEP	Operation mode
0	Interrupt is generated due to the setting of HFCOMP.
1	Interruption generation

**[bit 9] CANCEL: Token cancellation permission**

This bit sets whether a token is to be cancelled when the token (which is issued in an EOF area) has never been executed and is in waiting status if the SOFIRQ bit in the host interrupt register (HIRQ) is "1". It is not initialized with the RST bit in the UDC control register (UDCC).

CANCEL	Operation mode
0	Token continuance
1	Token discontinuance

**[bit 8] RETRY: Retry permission**

This bit sets whether a retry is attempted when a NAK and CRC errors happen. It is not initialized with the RST bit in the UDC control register (UDCC).

RETRY	Operation mode
0	No retry
1	Retry

**[bit 7] RWKIRE: Reactivation interrupt request permission**

It is a bit to set whether an interrupt is to be generated when the HOST function can be operational after resume operation has been completed. In host mode, to enter suspend status, write "1" to the SUSP bit in the host state status register. Only the host mode is effective. It is not initialized with the RST bit in the UDC control register (UDCC).

RWKIRE	Operation mode
0	Reactivation interrupt disable
1	Reactivation interrupt enable

**[bit 6] URIRE: USB bus reset interrupt request enable**

This bit sets whether an interrupt is to be generated when reset operation to the USB bus has been completed. Only the host mode is effective. It is not initialized with the RST bit in the UDC control register (UDCC).

URIRE	Operation mode
0	Interrupt disabled after USB bus is reset
1	Interruption enable after USB bus is reset



**[bit 5] CMPIRE: Completion interrupt request enable**

It sets whether an interrupt is generated when a token has been completed. Only the host mode is effective. It is not initialized with the RST bit in the UDC control register (UDCC).

CMPIRE	Operation mode
0	Completion interrupt disabled
1	Completion interrupt enabled

**[bit 4] CNNIRE: Connection interrupt request enable**

It is a bit used to set whether an interrupt is generated when a connection is made to a device. Only the host mode is effective. It is not initialized with the RST bit in the UDC control register (UDCC).

CNNIRE	Operation mode
0	Interrupt disabled when device is connected
1	Interruption enable when device is connected

**[bit 3] DIRE: Disconnection interrupt request enable**

It is a bit used to set whether an interrupt is generated when a disconnection is made to a device. Only the host mode is effective. It is not initialized with the RST bit in the UDC control register (UDCC).

DIRE	Operation mode
0	Interrupt disabled when device is cut
1	Interruption enable when device is cut

**[bit 2] SOFIRE:SOF interrupt request enable**

It is a bit used to set whether an interrupt is generated when sending an SOF. Only the host mode is effective. It is not initialized with the RST bit in the UDC control register (UDCC).

SOFIRE	Operation mode
0	Interrupt disabled when SOF is transmitted
1	Interrupt enabled when SOF is transmitted

**[bit 1] URST: USB bus reset**

It is set to USB bus whether reset is generated. It indicates "1" while the USB bus is being reset and turns "0" when it has been completed. It is forbidden to set it to "1" when the SUSP bit in the host state status register (HSTATE) is "1" or while a token is being executed. It is also forbidden to update the host control registers (HCNT0, HCNT1) while it is set to "1". Only the host mode is effective. To update them, you must set the RST bit in the UDC control register (UDCC) to "0".

URST	Operation mode
0	USB bus state maintenance
1	USB bus reset

**[bit 0] HOST: Host mode**

The LSI is set whether it is a function or HOST. Since it is not initialized due to the RST bit in the UDC control register (UDCC), change it when the RST bit is "1".

And, if you change the mode from function mode to host mode, make a disconnection to the HOST PC or HUB by setting the  $\overline{\text{HCON}}$  bit in the UDC control register (UDCC) to "1". If you change the mode from host mode to function mode, ensure that the SOFBUSY bit in the host status register (HSTATE) is set to "0" and the TKNEN bit in the host token endpoint register (HTOKEN) is 000<sub>B</sub>.

HOST	Operation mode
0	Function mode
1	Host mode

## 14.4.2 Host Interruption Register (HIRQ)

The host interrupt register (HIRQ) indicates for the interrupt request flag for USB HOST. It can allow an interrupt to be generated by setting the interrupt enable bit in the host control registers (HCNT0/HCNT1) except the TCAN bit.

### ■ Host Interruption Register (HIRQ)

Figure 14.4-2 Bit Configuration of Host Interruption Register (HIRQ)

Host interruption register								
bit	7	6	5	4	3	2	1	0
Address: 0000C2 <sub>H</sub>	TCAN	Reserved	RWKIRQ	URIRQ	CMPIRQ	CNNIRQ	DIRQ	SOFIRQ
Read/Write →	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value →	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
Reset On/Off at UDCC RST bit →	(○)	(○)	(○)	(○)	(○)	(×)	(×)	(○)

#### [bit 7] TCAN: Token cancellation flag

When the SOFIRQ bit in the host interrupt register (HIRQ) becomes "1", it indicates that a token is cancelled without being executed once. Any interrupt is not raised because the register is combined with interrupt due to SOF. To set it to "0", write "0" to it.

To update them, you must set the RST bit in the UDC control register (UDCC) to "0".

TCAN	Operation mode
0	There is no token discontinuance.
1	There is token discontinuance.

#### [bit 6] Reserved

It is reserved bit.

Be sure to set this bit to "0".

#### [bit 5] RWKIRQ: Reactivation interrupt request

It indicates that resume operation has been completed. When it becomes "1", it gets back to "0" by writing "0" to it. When you write "1" to it, the current state will be preserved. If the RWKIRE bit in the host control register 0 (HCNT0) is "1", an interrupt is generated when it is "1". The interrupt signal is cleared when it is cleared with "0".

To update them, you must set the RST bit in the UDC control register (UDCC) to "0".

RWKIRQ	Operation mode
0	There is no interrupt request by reactivation.
1	There is an interrupt request by reactivation.

**[bit 4] URIRQ: USB bus interrupt request**

It is shown that reset in USB bus ended. When it becomes "1", it gets back to "0" by writing "0" to it. When you write "1" to it, the current state will be preserved. If the URIRE bit in the host control register 0 (HCNT0) is "1", an interrupt is generated when it is "1". The interrupt signal is cleared when it is cleared with "0".

To update them, you must set the RST bit in the UDC control register (UDCC) to "0".

URIRQ	Operation mode
0	There is no interrupt request by USB bus reset.
1	There is an interrupt request by USB bus reset.

**[bit 3] CMPIRQ: Completion interrupt request**

It is shown to have completed the token. It is not set to "1" when the TCAN bit in the host interrupt register (HIRQ) is "1". When it becomes "1", it gets back to "0" by writing "0" to it. When you write "1" to it, the current state will be preserved. If the CMPIRE bit in the host control register 0 (HCNT0) is "1", an interrupt is generated when it is "1". The interrupt signal is cleared when it is cleared with "0".

To update them, you must set the RST bit in the UDC control register (UDCC) to "0".

CMPIRQ	Operation mode
0	There is no interrupt request by token completion.
1	There is an interrupt request by the token completion.

**[bit 2] CNNIRQ: Connected interrupt request**

It is shown to have detected the connection of the device. When it becomes "1", it gets back to "0" by writing "0" to it. When you write "1" to it, the current state will be preserved. If the CNNIRE bit in the host control register 0 (HCNT0) is "1", an interrupt is generated when it is "1". The interrupt signal is cleared when it is cleared with "0".

It is not initialized with the RST bit in the UDC control register (UDCC).

CNNIRQ	Operation mode
0	Interrupt request none by device connection detection.
1	Indicates interrupt request due to device connection detected.

**[bit 1] DIRQ: Cutting interrupt request**

It is shown to have detected cutting the device. When it becomes "1", it gets back to "0" by writing "0" to it. When you write "1" to it, the current state will be preserved. If the DIRE bit in the host control register 0 (HCNT0) is "1", an interrupt is generated when it is "1". The interrupt signal is cleared when it is cleared with "0".

It is not initialized with the RST bit in the UDC control register (UDCC).

DIRQ	Operation mode
0	There is no interrupt request by device cutting detection.
1	There is interrupt request by device cutting detection.

**[bit 0] SOFIRQ: SOF interruption requests**

Whether the SOF token was begun is shown. When it becomes "1", it gets back to "0" by writing "0" to it. When you write "1" to it, the current state will be preserved. If the SOFIRE bit in the host control register 0 (HCNT0) is "1", an interrupt is generated when it is "1". The interrupt signal is cleared when it is cleared with "0".

To update them, you must set the RST bit in the UDC control register (UDCC) to "0".

SOFIRQ	Operation mode
0	There is no interrupt request by SOF token beginning.
1	There is an interrupt request by the SOF token beginning.

### 14.4.3 Host Error Status Register (HERR)

The host error status register (HERR) is a register that indicates whether an error occurs or not when sending or receiving data in host mode.

#### ■ Host Error Status Register (HERR)

Figure 14.4-3 Bit Configuration of Host Error Status Register (HERR)

Host error status register								
bit	15	14	13	12	11	10	9	8
Address: 0000C3 <sub>H</sub>	LSTSOF	RERR	TOUT	CRC	TGERR	STUFF	HS	
Read/Write →	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value →	(0)	(0)	(0)	(0)	(0)	(0)	(11 <sub>B</sub> )	
Reset On/Off at UDCC RST bit →	(○)	(○)	(○)	(○)	(○)	(○)	(○)	

#### [bit 15] LSTSOF: SOF execution error

It indicates that an SOF token could not be executed because another token was running when you tried to execute it in host mode. Please do "0" in the writing to clear "1". To update them, you must set the RST bit in the UDC control register (UDCC) to "0".

LSTSOF	Operation mode
0	SOF execution
1	SOF execution error

#### [bit 14] RERR: Receive error

It indicates whether data more than a maximum number of packets set was received in host mode. When the reception error occurs, TOUT is set in "1".

Please do "0" in the writing to clear "1". To update them, you must set the RST bit in the UDC control register (UDCC) to "0".

RERR	Operation mode
0	No receive error
1	The maximum packet reception error

**[bit 13] TOUT: Time-out**

It indicates whether time-out was generated. If "1" is cleared, write "0" to this bit. The bit is updated after the RST bit of the UDC control register (UDCC) is set to "0".

TOUT	Operation mode
0	There is no time-out.
1	There is a time-out.

**[bit 12] CRC: CRC error**

It is shown whether the CRC error occurred at the host mode. When the CRC error occurs, TOUT is set in "1".

Please do "0" in the writing to clear "1".

To update them, you must set the RST bit in the UDC control register (UDCC) to "0".

CRC	Operation mode
0	There is no CRC error.
1	There is a CRC error.

**[bit 11] TGERR: Toggle error**

It indicates whether a toggle error occurs in host mode. Please do "0" in the writing to clear "1". To update them, you must set the RST bit in the UDC control register (UDCC) to "0".

TGERR	Operation mode
0	There is no toggle error.
1	There is a toggle error.

**[bit 10] STUFF: Stuffing error**

It indicates whether a stuffing error happens in host mode. TOUT is also set to "1" when a stuffing error occurs.

Please do "0" in the writing to clear "1". To update it, you must set the RST bit in the UDC control register (UDCC) to "0".

## [bit 9, bit 8] HS: Handshake status

It indicates the status of handshake operations between transmission and reception in host mode. It indicates NULL when handshake operation is not performed due to any reasons such as an error and the SOF token is completed. It is updated when transmission or reception has been completed. To update them, you must set the RST bit in the UDC control register (UDCC) to "0".

**Table 14.4-1 Handshake**

HS		Handshake
bit9	bit8	
0	0	ACK
0	1	NAK
1	0	STALL
1	1	NULL



## 14.4.4 Host State Status Register (HSTATE)

The host state status register (HSTATE) is a register that indicates the status of the USB circuit such as connections to devices and transfer mode. Note that the CLKSEL bit is also enabled in the function mode.

### ■ Host State Status Register (HSTATE)

Figure 14.4-4 Bit Configuration of Host State Status Register (HSTATE)

Host state status register										
	bit	7	6	5	4	3	2	1	0	
Address: 0000C4 <sub>H</sub>		Reserved	Reserved	ALIVE	CLKSEL	SOFBUSY	SUSP	TMODE	CSTAT	HSTATE
Read/Write →		(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	(R)	(R)	
Initial value →		(x)	(x)	(0)	(1)	(0)	(0)	(1)	(0)	
Reset On/Off at UDCC RST bit →		(-)	(-)	(×)	(×)	(○)	(○)	(×)	(×)	

#### [bit 7, bit 6] Reserved

It is reserved bits. The reading is undefined. The writing does not influence the operation.

#### [bit 5] ALIVE: Keep-Alive function setting

It is a bit that sets the Keep-Alive function in Low Speed. If you set it to "1" when the CLKSEL bit in the host state status register (HSTATE) is "0", SE0 is will be output instead of an SOF. If it is enabled when the CLKSEL bit in the host state status register (HSTATE) is "0", and the CLKSEL becomes "1", an SOF will be output regardless of the setting of the ALIVE bit.

ALIVE	Operation mode
0	SOF
1	SE0 output (Keep-Alive)

#### [bit 4] CLKSEL: Clock selection

The operation clock of USB is selected. You must set it to "1" for Full Speed, and to "0" for Low Speed.

You must switch clock when the RST bit in the UDC control register (UDCC) is "1" and both the function mode and host mode are enabled. "0" setting at the function mode is a interdiction.

CLKSEL	Operation mode
0	Clock for Low Speed
1	Clock for Full Speed

**[bit 3] SOFBUSY: SOF timer operation**

It indicates whether the SOF timer is operating in host mode. Sending SOF stops when "0" is done in the writing. To update them, you must set the RST bit in the UDC control register (UDCC) to "0".

SOFBUSY	Operation mode
0	SOF timer is stop.
1	SOF timer is working.

**[bit 2] SUSP: Suspend**

It is a bit used to set suspend status in host mode. Writing "1" to the bit enables suspend status. When you write "0" to the bit that holds "1" or the USB bus changes to k-state status, the suspend status is deselected, the RWKIRQ bit in the host interrupt register (HIRQ) is "1". When connection or disconnection is detected if the SUSP bit is "1", you must write "0" to the bit. Then, when disconnection or connection is detected and the SUSP bit is cleared, the RWKIRQ bit in the host interrupt register (HIRQ) is set to "1" twice, and overwrite the RWKIRQ bit with "1" when it is set to "1". It is forbidden to set the bit to "1" while the USB is operating (such as resetting the USB bus or sending/receiving data).

In host mode, it is forbidden to stop the USB clock even in suspend status. To update it, you must set the RST bit in the UDC control register (UDCC) to "0" and are prohibited to set it to "1" in function mode. In addition, if it is "1" when you change the mode from host mode to function mode, you must get out of suspend status by writing "0" into it before changing the mode.

**Table 14.4-2 Suspend Setting**

SUSP	Operation
"1" Write	Suspend
They are "0" writing at "1" state.	Resume
The others	State maintenance

**[bit 1] TMODE: Transfer mode**

The transfer mode at the host mode is shown. Write "1" for the write operation. It is not initialized with the RST bit in the UDC control register (UDCC). Please use the CPU clock at 24 MHz.

TMODE	Operation mode
0	Low Speed
1	Full Speed

**[bit 0] CSTAT: Connected state**

It is whether the device is connected is shown. The pin for HOST becomes an object. It is not initialized with the RST bit in the UDC control register (UDCC).

CSTAT	Operation mode
0	Device cut off
1	Device connect

### 14.4.5 SOF Interruption FRAME Comparison Register (HFCOMP)

The SOF interrupt FRAME comparison register (HFCOMP) is a register used to set data that is compared with the lower 8 bits of FRAME Number for SOF token. If the lower 8 bits of FRAME Number is compared with the HFCOMP register and a match is detected with the SOFIRE bit in host control register 0 (HCNT0) set to "1", an interrupt will be generated by setting the SOFIRQ bit in the host interrupt register (HIRQ) to "1" when starting SOF transmission.

#### ■ SOF Interruption FRAME Comparison Register (HFCOMP)

Figure 14.4-5 Bit Configuration of SOF Interruption FRAME Comparison Register (HFCOMP)

SOF interruption FRAME comparison register								
bit	15	14	13	12	11	10	9	8
Address: 0000C5 <sub>H</sub>	FRAMECOMP							HFCOMP
Read/Write →	(R/W)							
Initial value →	(00000000 <sub>B</sub> )							
Reset On/Off at UDCC RST bit →	(×)							

#### [bit 15 to bit 8] FRAMECOMP

It sets data that is to be compared with the lower 8 bits of Frame Number. It is not initialized with the RST bit in the UDC control register (UDCC). To update them, you must set the RST bit in the UDC control register (UDCC) to "0".

14.4.6 Retry Timer Setting Register (HRTIMER)

The retry timer setting register (HRTIMER) is a register used to set a retry time period for a token.

■ Retry Timer Setting Register (HRTIMER)

Figure 14.4-6 Bit Configuration of Retry Timer Setting Register (HRTIMER)

Retry timer setting register								
bit	7	6	5	4	3	2	1	0
Address: 0000C6 <sub>H</sub>	RTIMER0							HRTIMER
Read/Write →	(R/W)							
Initial value →	(00000000 <sub>B</sub> )							
Reset On/Off at UDCC RST bit →	(×)							
bit	15	14	13	12	11	10	9	8
Address: 0000C7 <sub>H</sub>	RTIMER1							HRTIMER
Read/Write →	(R/W)							
Initial value →	(00000000 <sub>B</sub> )							
Reset On/Off at UDCC RST bit →	(×)							
bit	7(23)	6(22)	5(21)	4(20)	3(19)	2(18)	1(17)	0(16)
Address: 0000C8 <sub>H</sub>	Reserved						RTIMER2	HRTIMER
Read/Write →	(-)						(R/W)	
Initial value →	(x)						(00 <sub>B</sub> )	
Reset On/Off at UDCC RST bit →	(-)						(×)	

[bit 23 to bit 18] Reserved

These are reserved bits.

The reading is undefined. The writing does not influence the operation.

[bit 17 to bit 0] HRTIMER0, HRTIMER1, HRTIMER2

These bits set a time to retry a token. When the RETRY bit of the host control register (HCNT1) is "1", a retry timer is activated after the token is started, and the timer is decremented by "1" due to 1-bit transfer clock (12 MHz at Full Speed). When the retry timer becomes "0", token retry dose not execute.

When the token retry occurs in an EOF area, the retry timer stops until the SOF is completed. The timer value that was stopped is decremented by 1 after the SOF is executed. It is not initialized with the RST bit in the UDC control register (UDCC). To update them, you must set the RST bit in the UDC control register (UDCC) to "0".

14.4.7 Host Address Register (HADR)

The host address register (HADR) is a register used for an address field when a token is sent.

■ Host Address Register (HADR)

Figure 14.4-7 Bit Configuration of Host address Register (HADR)

Host address register									
	bit	15	14	13	12	11	10	9	8
Address: 0000C9 <sub>H</sub>		Reserved	Address						HADR
Read/Write →		(-)	(R/W)						
Initial value →		(x)	(0000000 <sub>B</sub> )						
Reset On/Off at UDCC RST bit →		(-)	(X)						

[bit 15] Reserved

It is reserved bit. The reading is undefined. The writing does not influence the operation.

[bit 14 to bit 8] Address: Address

The address of the token is set. It is not initialized with the RST bit in the UDC control register (UDCC). To update them, you must set the RST bit in the UDC control register (UDCC) to "0".



## 14.4.9 FRAME Setting Register (HFRAME)

The FRAME setting register (HFRAME) is a register that sets a FRAME Number in handling SOF tokens. When you set the TKNEN bits of the host token endpoint register (HTOKEN) to SOF activation, the SOF timer starts and, afterwards, an SOF is automatically sent out every 1 ms. The FRAME setting register is automatically incremented by 1 every time an SOF is completed.

### ■ FRAME Setting Register (HFRAME)

Figure 14.4-9 Bit Configuration of FRAME Setting Register (HFRAME)

FRAME setting register

bit	7	6	5	4	3	2	1	0
Address: 0000CC <sub>H</sub>	FRAME0							HFRAME
Read/Write →	(R/W)							
Initial value →	(00000000 <sub>B</sub> )							
Reset On/Off at UDCC RST bit →	(○)							

bit	15	14	13	12	11	10	9	8	
Address: 0000CD <sub>H</sub>	Reserved					FRAME1			HFRAME
Read/Write →	(-)					(R/W)			
Initial value →	(x)					(000 <sub>B</sub> )			
Reset On/Off at UDCC RST bit →	(-)					(○)			

#### [bit 15 to bit 11] Reserved

These are reserved bits. The reading is undefined. The writing does not influence the operation.

#### [bit 10 to bit 0] FRAME1, FRAME0

Frame Number is set. Before setting the TKNEN bits of the host token endpoint register (HTOKEN) to SOF, set Frame Number. Furthermore, when the SOFBUSY bit of the host status register (HSTATE) is "1" and an SOF token is being executed, write operation is inhibited. To update them, you must set the RST bit in the UDC control register (UDCC) to "0".



14.4.10 Host Token Endpoint Register (HTOKEN)

The host token endpoint register (HTOKEN) is a register that sets a toggle, endpoint, and token.

■ Host Token Endpoint Register (HTOKEN)

Figure 14.4-10 Bit Configuration of Host Token Endpoint Register (HTOKEN)

Host token end point register											
		bit	7	6	5	4	3	2	1	0	
Address: 0000CE <sub>H</sub>			TGGL	TKNEN			ENDPT				HTOKEN
Read/Write →			(R/W)	(R/W)			(R/W)				
Initial value →			(0)	(000 <sub>B</sub> )			(0000 <sub>B</sub> )				
Reset On/Off at UDCC RST bit →			(○)	(○)			(○)				

[bit 7] TGGL: Toggle

This bit sets toggle data. At transmission, the toggle data is sent according to the bit. At reception, the received toggle data is compared to the toggle data which the bit shows and use at the error detection. The bit is updated after the RST bit of the UDC control register (UDCC) is set to "0" and the TKNEN bit to 000<sub>B</sub>.

TGGL	Operation Mode
0	Data 0
1	Data 1

[bit 6 to bit 4] TKNEN: Token permission

These bits send a token corresponding to the setting. After the operation is completed TKNEN = 000<sub>B</sub>, the CMPIRQ bit of the host interrupt register (HIRQ) is set to "1". At that time, if the CMPIRE bit of the host control register 0 (HCNT0) is "1", an interrupt generates.

The TGGL and ENDPT bits are ignored during the SOF token. To write to the TKNEN bits, you must set the RST bit in the UDC control register (UDCC) to "0" and turn the mode to host mode. In addition, if you issue a token again because an interrupt due to a token is generated, you must wait three cycles or longer in terms of USB transfer clock (12 MHz for Full Speed and 1.5 MHz for Low Speed) before writing to the TKNEN bits. Note that writing to the TKNEN bits will not run a token in disconnection status (HSTATE CSTAT= 0).

**Table 14.4-3 Token Setting**

bit6	bit5	bit4	Operation
0	0	0	No send out
0	0	1	SETUP is sent
0	1	0	IN is sent.
0	1	1	OUT is sent.
1	0	0	SOF is sent.
1	0	1	Reserved (Set prohibition)
1	1	0	Reserved (Set prohibition)
1	1	1	Reserved (Set prohibition)

**Note:**

The PRE packet is not supported.

When the SOFBUSY bit in the host state status register (HSTATE) is "1", setting TKNEN = 100<sub>B</sub> respectively, is forbidden.

**[bit 3 to bit 0] ENDPT: end point**

The transmitted and received endpoint to the device is set. To update them, you must set the RST bit in the UDC control register (UDCC) to "0".

## **14.5 Operation of USB HOST**

---

**The operation of USB HOST is explained.**

---

### **■ Operation of USB HOST**

- **Connection of device**

The software detects that the external USB device was connected.

- **Reset of USB bus**

USB bus is reset.

- **Token packet**

Three kinds of tokens can be selected at the host mode.

- **Data packet**

The data packet is transmitted and received.

- **Handshake packet**

It informs send/receive partners of status via handshake packet.

- **Retry function**

When an error etc. occur at the packet termination, the retry operation is continued.

- **SOF INTERRUPT**

The interruption is generated.

- **Error status**

Various errors are displayed.

- **Packet end**

When the packet ends, the interruption is generated.

- **Suspend-resume**

It puts the USB circuit in suspend-resume status.

- **Cutting of device**

The detection of the status of HOST pins determines the disconnection.

## 14.5.1 Connection of Device

---

The method for detecting the connection of the external USB device by software is described.

---

### ■ Setting of HOST Function

To make it operate as a host of the USB device, set the HOST bit of the host control register 0 (HCNT0) to "1".

### ■ Disconnection Status, Connection Status of the External USB Device

When the external USB device is disconnected, both HOST pins, D + and D-, are "L" by the pull-down resistor. Then, the CSTAT bit in the host state status register (HSTATE) is "0", and the TMODE bit is indefinite. The CSTAT bit of the host state status register (HSTATE) becomes "1" when the external USB device is connected.

### ■ Connected Detection of External USB Device

To detect the connection of the external USB device, set the CNNIRE bit of the host control register 0 (HCNT0) to "1". The CNNIRQ bit of the host interrupt register (HIRQ) becomes "1", and a device connection interrupt is generated. To clear this interrupt, write "0" to the CNNIRQ bit of the host interrupt register (HIRQ). To detect the connection of the external USB device not through interrupt but through polling, create a program so that it ensures that the CNNIRE bit of the host control register 0 (HCNT0) is set to "0" and the CNNIRQ bit of the host interrupt register (HIRQ) is "1".

### ■ Acquiring Transfer Speed of Destination USB Device and Selecting Clock

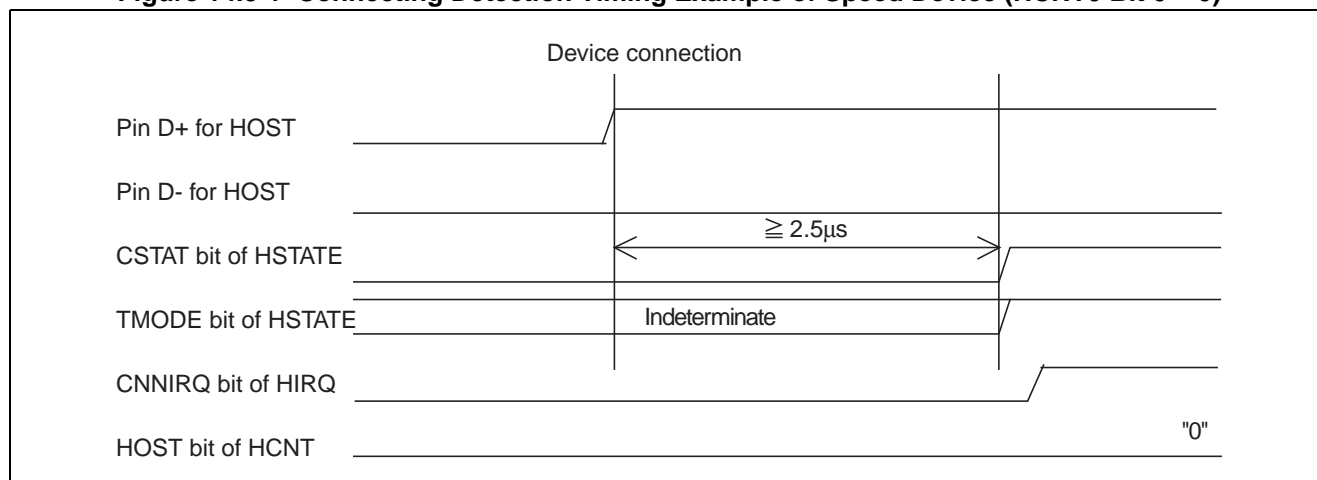
To acquire the transfer speed of a destination USB after detecting the connection, make reference to the value of the TMODE of the host state status register (HSTATE). The relation between the transfer speed and the TMODE bit of the host state status register (HSTATE) is as follows:

- When a destination device is a Full Speed-enabled device → TMODE= 1
- When a destination device is a Low Speed-enabled device → TMODE= 0

After the transfer speed of the external USB device is acquired, when the RST bit of the UDC control register (UDCC) is "1", update the CLKSEL bit of the host state status register (HSTATE) based on the acquired transfer speed. The relation between the TMODE and CLKSEL bits of the host status register (HSTATE) is as follows:

- TMODE = 1 → "1" is set to CLKSEL bit.
- TMODE = 0 → "0" is set to CLKSEL bit.

**Figure 14.5-1 Connecting Detection Timing Example of Speed Device (HCNT0 Bit 0 = 0)**



**Note:**

The CSTAT bit of the host state status register (HSTATE) becomes "1" in 2.5  $\mu\text{s}$  after the connection of the external USB device.

The TMODE and CSTAT bits of the host state status register (HSTATE) are updated regardless of the setting of the HOST bit in the host control register 0 (HCNT0).

## 14.5.2 Reset of USB Bus

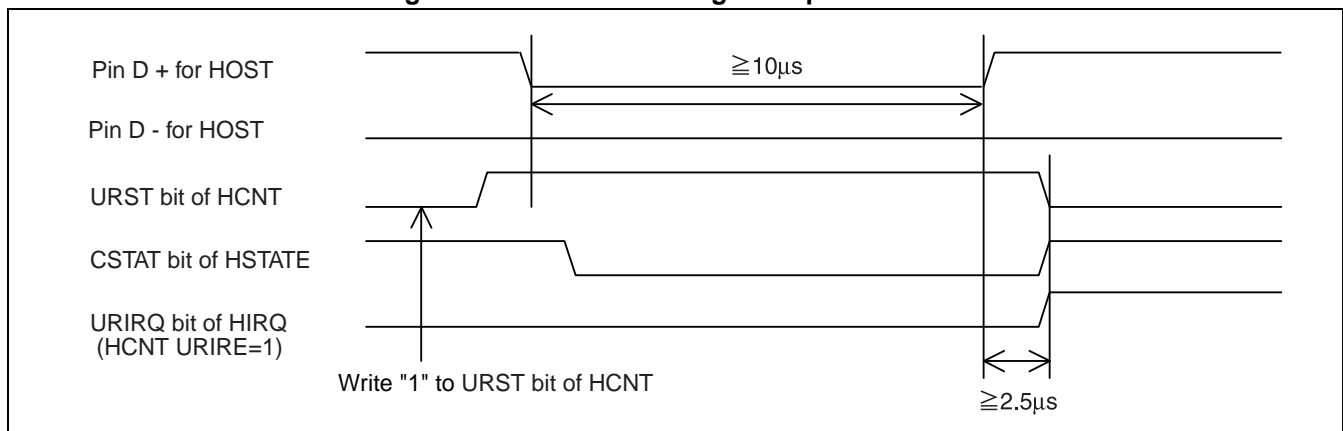
When you set the URST bit of the host control register 0 (HCNT0) to "1" in the host mode, it sends out SE0 for not less than 10 ms and resets the USB bus. When the USB bus has been reset, it sets back the URST bit of the host control register to "0" and generates an interrupt and the URIRQ bit of the host interrupt register (HIRQ) is set to "1" when the URIRE bit of the host control register 0 (HCNT0) is "1." If you clear the interrupt, write "0" to the URIRQ bit of the host interrupt register (HIRQ).

### ■ Notes before and after Reset of USB Bus

Please note the following points about reset of USB bus.

1. Before the USB bus is reset, confirm that the device is connected to and the CSTAT bit of the host state status register (HSTATE) is set to "1".
2. When you reset the USB bus, the CSTAT bit of the host state status register (HSTATE) turns to "0" and the USB device is put into disconnection status. Then, the DIRQ bit of the host interrupt control register (HIRQ) does not become "1".
3. After the USB bus is reset, you must update the CLKSEL bit of the host state status register (HSTATE) to match the TMODE bit of the same register if you compare them and they do not match. Before you update the CLKSEL bit, ensure that the RST bit of the UDC control register (UDCC) is "1".

Figure 14.5-2 Reset Timing Example to Device



### 14.5.3 Token Packet

If you execute any of an IN token, OUT token, and SETUP token in the host mode, a token packet is started when you set necessary data in the host token register (HTOKEN) after you set the PKS bit of the EP1 control register (EP1C) or EP2 control register (EP2C) based on the host address register (HADR) and the DIR bit in EP1C. In handling an SOF token, you must set necessary data in the host token register (HTOKEN) after configuring the FRAME setting register (HFRAME) and EOF setting register (HEOF). If registers (HADR, EP1C, EP2C, HFRAME, and HEOF) have not been changed, setting them is not required.

#### ■ Setting of Token Packet

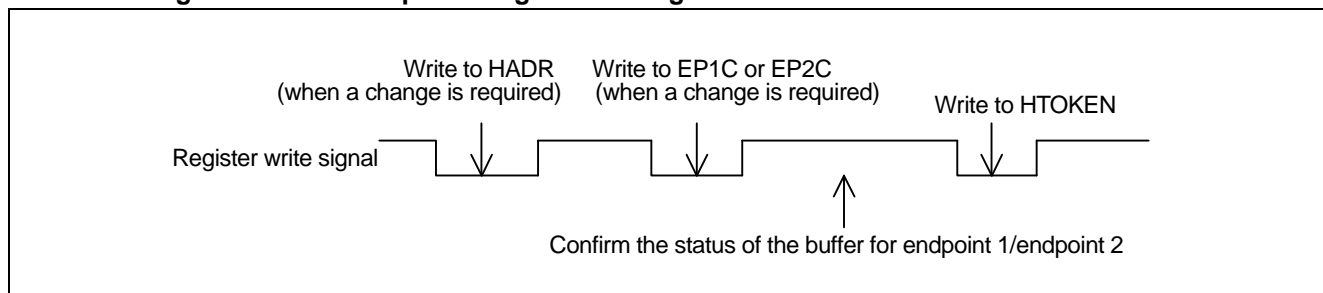
For the IN token, OUT token, and SETUP token, set the destination address to the host address register (HADR), and set the maximum bytes of one packet to the PKS bit of the EP1 control register (EP1C) or the EP2 control register based on the token to be executed and the DIR bit of the EP1 control register.

If the DIR bit of the EP1 control register (EP1C) is "1", the buffer for endpoint 1 is used as an OUT- direction buffer and the one for endpoint 2 is used as an IN- direction buffer. Then, set the DIR bit of the EP2 control register (EP2C) to "0".

If the DIR bit of the EP1 control register (EP1C) is "0", the buffer for endpoint 1 is used as an IN- direction buffer and the one for endpoint 2 is used as an OUT- direction buffer. Then, set the DIR bit of the EP2 control register (EP2C) to "1".

If you want to use the buffer for endpoint 1, you must ensure that the DRQ bit of EP1 status register (EP1S) is set to "0" and if you want to use the buffer for endpoint 2, you must ensure that the DRQ bit of EP2 status register (EP2S) is set to "1," before setting the target endpoint, token, and toggle data in the host token endpoint register (HTOKEN). The USB circuit sends out a token packet in the order of a Sync, token, address, endpoint, CRC5, and EOP based on the specified token (a Sync, CRC5, and EOP are automatically sent). After one packet is ended, the CMPIRQ bit of the host interrupt register (HIRQ) becomes "1", and the TKNEN bit of the host token endpoint register (HTOKEN) is set to 000<sub>B</sub> (See Section "14.5.7 SOF Interrupt"). At that time, if the CMPIRE bit of the host control register 0 (HCNT0) is "1", an interrupt occurs. To clear the interrupt, write "0" to the CMPIRQ bit of the host interrupt register (HIRQ).

**Figure 14.5-3 Example of Register Setting until Execution of IN/OUT/SETUP Token**



In the case of an SOF token, when you set an EOF time and FRAME number in the EOF setting register (HEOF) and FRAME setting register (HFRAME), and write the code of the SOF token to the TKNEN bits of the host token endpoint register (HTOKEN), a Sync, SOF token, FRAME number, CRC5, and EOP will be sent out and the SOFBUSY bit of the host state status register (HSTATE) is set to "1" and HFRAME is

incremented by 1. In this case, the CMPIRQ of the host interrupt register (HIRQ) is also set to "1", and the TKNEN bit of the host token endpoint register (HTOKEN) is cleared to 000<sub>B</sub>. When the CMPIRE bit of host control register (HCNTO) is "1", an interrupt occurs. Then, when the SOF that generates automatically is used, the interrupt by the CMPIRQ does not occur. To clear the interrupt of the token completion, write "0" to the CMPIRQ of the HIRQ.

SOF is automatically sent out every 1 ms while the SOFBUSY bit of the host state status register (HSTATE) is "1". The conditions (SOF stop conditions) that make the SOFBUSY bit of the host state status register (HSTATE) "0" are as follows:

"0" write to SOFBUSY bit of host state register (HSTATE)

Reset ("1" write to URST bit of HCNT) in USB bus

"1" write to SUSP bit of host state status register (HSTATE)

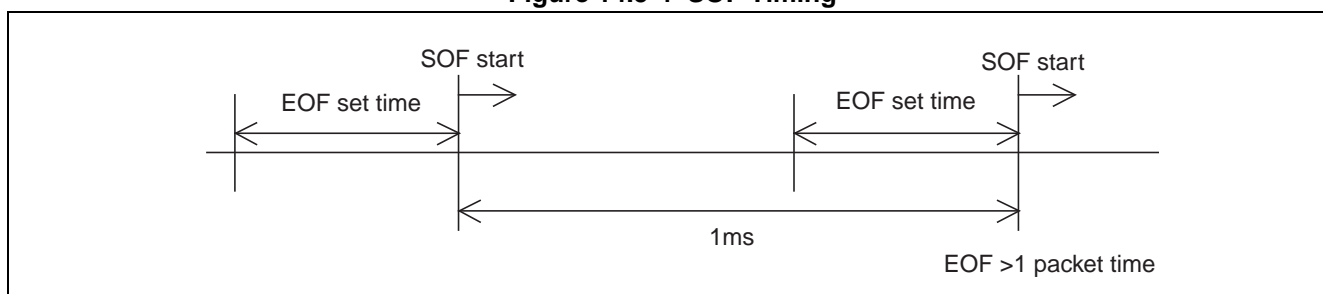
Cutting of the device (For "0" the CSTAT bit of HSTATE).

To switch from host mode to function mode, first, ensure that the SOFBUSY bit of the host state status register (HSTATE) is "0" after writing "0" to it.

If you want to set back the SOFBUSY bit of the host state status register (HSTATE) to "1" again, you need to run an SOF token once again.

To prevent the simultaneous executions of an SOF token and other tokens, the EOF setting register is used to run a token you have set after making it wait for the end of SOF execution if you write to the TKNEN bits of the host token endpoint register (HTOKEN) in a time period from the EOF setting time to SOF start time. The unit of time for the EOF setting register is one-bit time. For example, when setting 10<sub>H</sub> to the EOF setting register, the following time is required:  $16 \times 1 / 12 \text{ MHz} = 13333.3 \text{ ns}$  in Full speed mode and  $16 \times 1 / 1.5 \text{ MHz} = 10666.6 \text{ ns}$  in Low speed mode. If the EOF set time is shorter than one packet time, the SOF execution max overlap other token execution. In this case, the LSTOF bit of host error status register (HERR) is set to "1", the SOF is not executed. When the LSTOF bit of the host error status register (HERR) is set to "1", you must increase data of the EOF setting register. (See Section "14.4.8 EOF Setting Register (HEOF)")

Figure 14.5-4 SOF Timing





## **14.5.4 Data Packet**

---

If a data packet is transmitted after a token packet has been sent, toggle data will be transmitted based on the TGGL bit of the host token endpoint register (HTOKEN), and the buffer data for endpoint 1 or endpoint 2 according to the DIR bit of the EP1 control register (EP1C), CRC16 data, and EOP is sent. In the case of receiving a data packet, the TGGL bit of the host token endpoint register (HTOKEN) and received toggle data are compared, and, if they match, the received data is written to the buffer for endpoint 1 or endpoint 2 based on the DIR bit of the EP1 control register (EP1C) and the CRC16 is checked for an error.

---

### **■ Data Packet**

After sending a token packet, the data packet is executed in the following procedure:

#### ● At Transmission

- Automatic sending of Sync
- DATA0 is transmitted when the TGGL bit of the host token endpoint register (HTOKEN) is "0" and DATA1 is transmitted when the TGGL bit is "1".
- The buffer for endpoint 1 is selected when the DIR bit of the EP1 control register (EP1C) is "1" and otherwise the one for endpoint 2 is selected when the DIR bit is "0" and all transmit data is sent.
- The CRC16-bit is sent.
- The EOP 2-bit is sent.
- The J State 1-bit is sent.

#### ● At Reception

- Reception of Sync
- The toggle data is received, and is compared with the TGGL bit of the host token endpoint register (HTOKEN).
- If they match as a result of the comparison, the buffer for endpoint 2 is selected when the DIR bit of the EP1 control register (EP1C) is "1" and the one for endpoint 0 is selected when the DIR bit is "0" and the received data is written into it.
- When the EOF is received, the CRC16 bit is inspected.

You must set inversion data, respectively, in the DIR bits of the EP1 control register (EP1C) and EP2 control register (EP2C) when the HOST bit of the host control register 0 (HCNT0) is "1". For example, when the DIR bit of the EP1 control register (EP1C) is "0", the DIR bit of the EP2 control register (EP2C) is set to "1".

## 14.5.5 Handshake Packet

---

**Transmission/reception partner must be informed of your own status via handshake packet.**

---

### ■ Handshake Packet

The reception side transmits one of ACK, NAK, and STALL when it determines through handshake packet whether it can receive data properly or the endpoint supports it. Then, when the USB circuit receives a handshake packet, the received handshake packet is set to the HS bit of the host error status register (HERR). When the handshake packet is transmitted, the transmitted handshake packet is set to the HS bit of the host error status register (HERR).

## 14.5.6 Retry Function

At the termination of the packet, when NAK or an error such as CRC error occurs, and the RETRY bit of the host control register 1 (HCNT1) is "1", it continues to retry during a time period set in the retry timer register (HRTIMER).

### ■ Retry Function

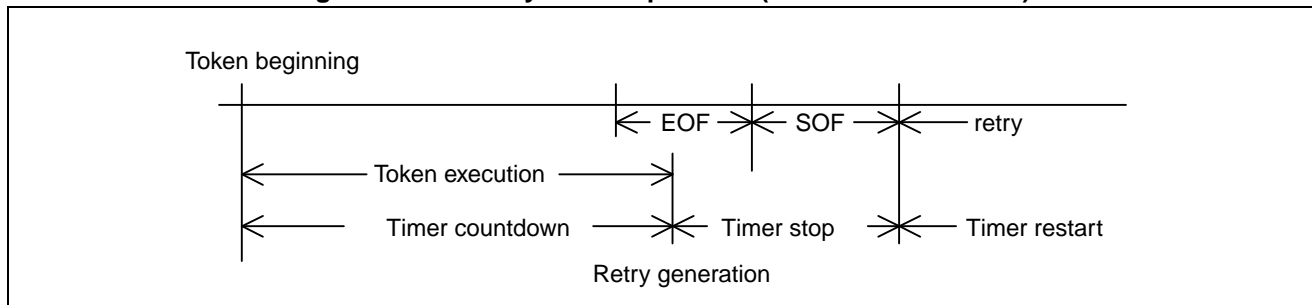
If an error except STALL and disconnected device happens, it retries to process the token when the RETRY bit of the host control register 1 (HCNT1). The end condition of retry

"0" setting of RETRY bit of host control register 1(HCNT1)

- Detecting 0 in the retry timer
- Occurrence of an interrupt due to SOF (SOFIRE= 1 of HCNT0 and SOFIRQ= 1 of HIRQ)
- Detection of ACK
- Detection of cutting device

The retry timer is activated when the process of a token is started, counts down with one-bit transfer clock, and stops counting when a retry happens in an EOF area. The retry timer restarts at the value when the timer stopped if the SOFIRQ bit of HIRQ was "0". The SOF token was completed, and when the timer counts down to "0" and a packet ends, any retry request it receives is cancelled and the packet will be terminated.

**Figure 14.5-5 Retry Timer Operation (SOFIRQ of HIRQ = 0)**



When the retry operation is completed, end information on the completed packet is set in related registers.

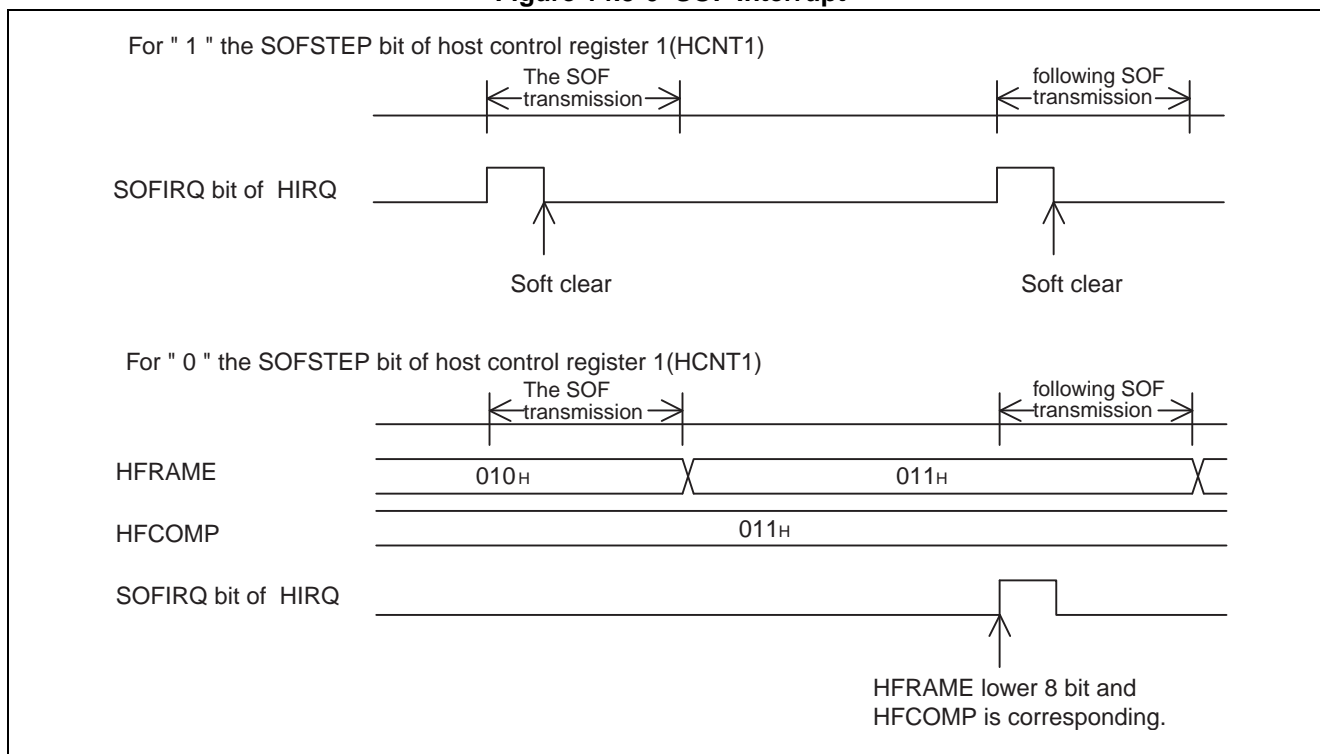
## 14.5.7 SOF Interrupt

Once you have set the **SOFIRE** bit of the host control register 0 (**HCNT0**) to "1", it sets the **SOFIRQ** bit of the host interrupt register (**HIRQ**) to "1" and will generate an interrupt when starting an SOF with the **SOFSTEP** bit of the host control register 1 (**HCNT1**) and the SOF interrupt **FRAME** comparison register (**HFCOMP**). The SOF execution with the host token endpoint register (**HTOKEN**) does not set the **SOFIRQ** bit of the host interrupt register (**HIRQ**) to "1".

### ■ SOF Interrupt

Once you have set the **SOFIRE** bit in host control register 0 (**HCNT0**) to "1", an interrupt is generated when sending an SOF, and the **SOFIRQ** bit in the host interrupt register (**HIRQ**) is set to "1". You can select one of an SOF interrupt conditions: One is generation of an SOF interrupt every time an SOF is sent out with the setting of the **SOFSTEP** bit of the host control register 1 (**HCNT1**) and another generation is due to the Frame number in the lower 8 bits indicated by the SOF interrupt **FRAME** comparison register (**HFCOMP**).

Figure 14.5-6 SOF Interrupt

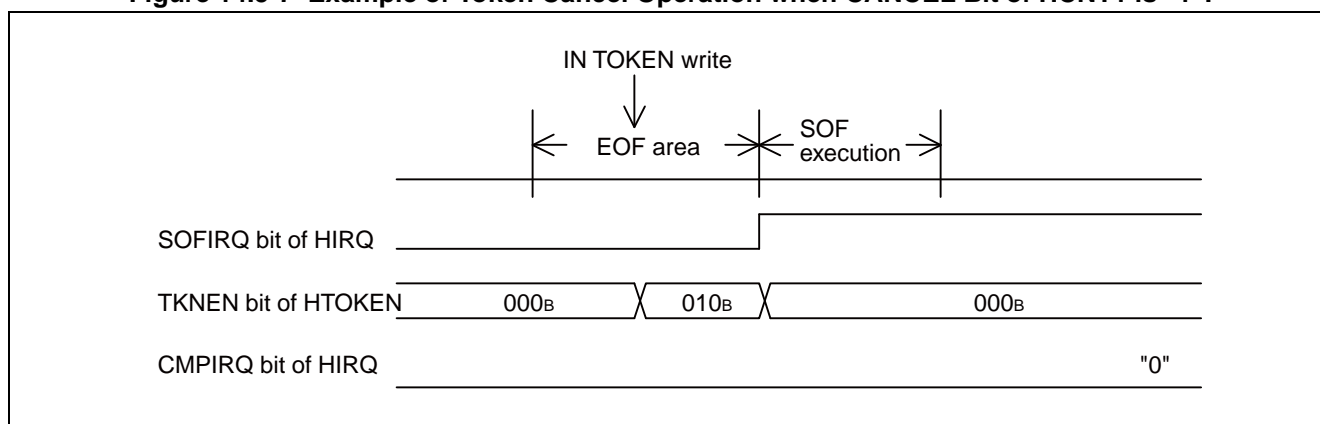


If you set the **CANCEL** bit of the host control register 1 (**HCNT1**) to "1" and set a token other than an SOF token in the host token endpoint register (**HTOKEN**) in the EOF area, and the **SOFIRQ** bit of the host interrupt register (**HIRQ**) is set to "1" in the next SOF, the token is not executed and the **TKNEN** bits of the **HTOKEN** are set to  $000_B$ . Then, the **CMPIRQ** bit of the host interrupt register (**HIRQ**) does not become "1".

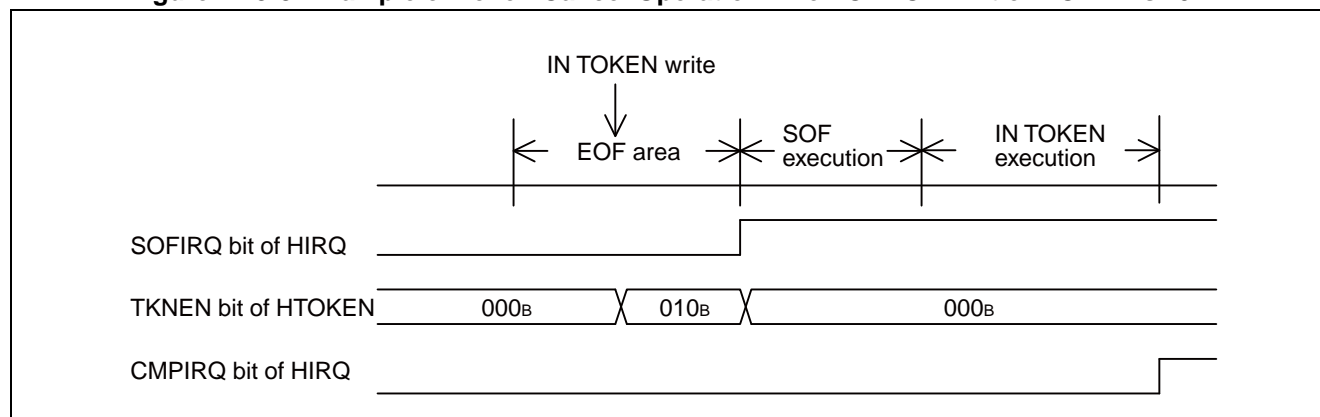
Canceling the token can be known by the **TCAN** bit of the **HIRQ** when the **SOFIRQ** bit becomes "1". If you execute the token again, write "0" to the **TCAN** bit of the **HIRQ**, and write the token to be executed to the **TKNEN** bit of **HTOKEN**.

If you set the CANCEL bit of host control register 1 (HCNT1) to "0", the token set in the host token endpoint register (HTOKEN) is executed after the SOF is sent.

**Figure 14.5-7 Example of Token Cancel Operation when CANCEL Bit of HCNT1 is "1".**



**Figure 14.5-8 Example of Token Cancel Operation when CANCEL Bit of HCNT1 is "0".**



## 14.5.8 Error Status

---

**USB HOST supports various error information.**

---

### ■ Error Status

#### ● Stuffing error

If continuous 6 bits happen to be "1", one bit of "0" should be inserted in somewhere in the sequence, but the STUFF bit of the host error status register (HERR) is set to "1" as a stuffing error if continuous 7 bits of "1" are detected. Please do "0" to the STUFF bit in the writing to clear this. If the next token is executed without clearing the STUFF bit, it is updated at the termination of the next token.

#### ● Toggle error

When receiving an IN token, the TGERR bit in the host error register (HERR) is set to "1" if the toggle data for the data packet and the TGGL bit of the host token endpoint register (HTOKEN) are compared and a match is not detected. To clear the TGERR bit, write "0" to it of the host error register (HERR). If the next token is executed without clearing the TGERR bit, it is updated at the termination of the next token.

#### ● CRC error

When receiving an IN token, data in the received data packet and CRC are calculated with the CRC polynomial  $G(X)=X^{16}+X^{15}+X^2+1$ , and if the remainder is not 800D<sub>H</sub>, then a CRC error is assumed to happen and the CRC bit of the host error register (HERR) is set to "1". To clear the CRC bit, write "0" to it of the host error register (HERR). If the next token is executed without clearing the CRC bit, it is updated at the termination of the next token.

#### ● Time-out error

The TOUT bit of the host error status register (HERR) is set to "1" if a data packet or handshake is not received in a given time, SE0 is detected in received data, or a stuffing error is detected. To clear the TOUT bit, write "0" to it of the host error register (HERR). If the next token is executed without clearing the TOUT bit, it is updated at the termination of the next token.

#### ● Receive error

If EP1 or EP2 is used as the reception buffer, the PKS bit of the EP1 control register (EP1C) or EP2 control register (EP2C), respectively. If the received data is greater than that of the received packet size, the RERR bit of the host error status register (HERR) is set to "1". To clear the RERR bit, write "0" to it of the host error register (HERR). If the next token is executed without clearing the RERR bit, it is updated at the termination of the next token.

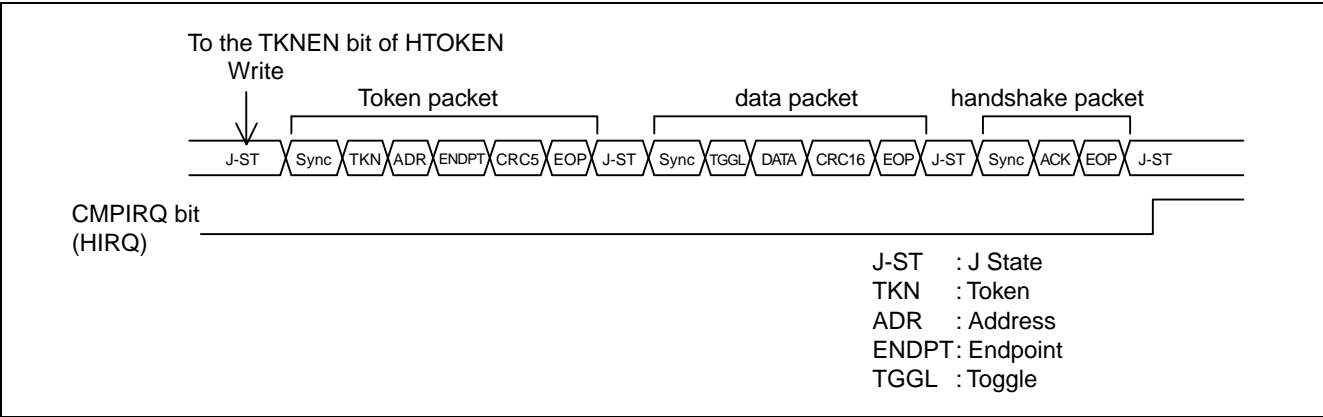
14.5.9 Packet End

When one packet terminates in USB HOST, if the CMPIRE bit of the host control register 0 (HCNT0) is "1", an interrupt is generated to set the CMPIRQ bit of the host interrupt register (HIRQ) to "1".

■ Packet End Timing

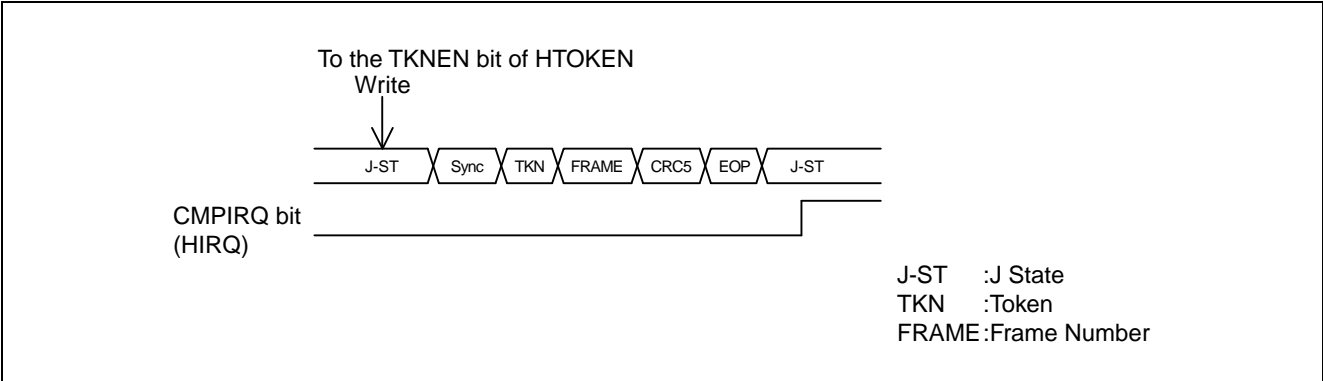
When one packet terminates, an interrupt is generated in the following timing:  
When the TKNEN bits of the host token end point register (HTOKEN) are 001<sub>B</sub>, 010<sub>B</sub> or 011<sub>B</sub> (SETUP token, IN token, and OUT token)

Figure 14.5-9 CMPIRQ Bit Set Timing Example 1 of HOST Interrupt Register (HIRQ)



When the TKNEN bit of the host token end point register (HTOKEN) is 100<sub>B</sub> (SOF token)

Figure 14.5-10 CMPIRQ Bit Set Timing Example 2 of HOST Interrupt Register (HIRQ) (SOF TOKEN)



## 14.5.10 Suspend Resume

USB HOST supports suspend and resume operations.

### ■ Suspend Operation

When writing "1" to the SUSP bit of the host state status register (HSTATE),

- USB bus high impedance state
- Stop of circuit block where clock is not necessary

USB HOST follows the steps above, and puts the USB circuit in suspend status. When the USB circuit is put in suspend status, it sets the SUSP bit of the host state status register (HSTATE) to "1".

It is inhibited for USB HOST to set the USB circuit to suspend status or stop clock supplied to the circuit when the USB bus is being reset or the SOFBUSY of HSTATE is "1" or data is being sent or received.

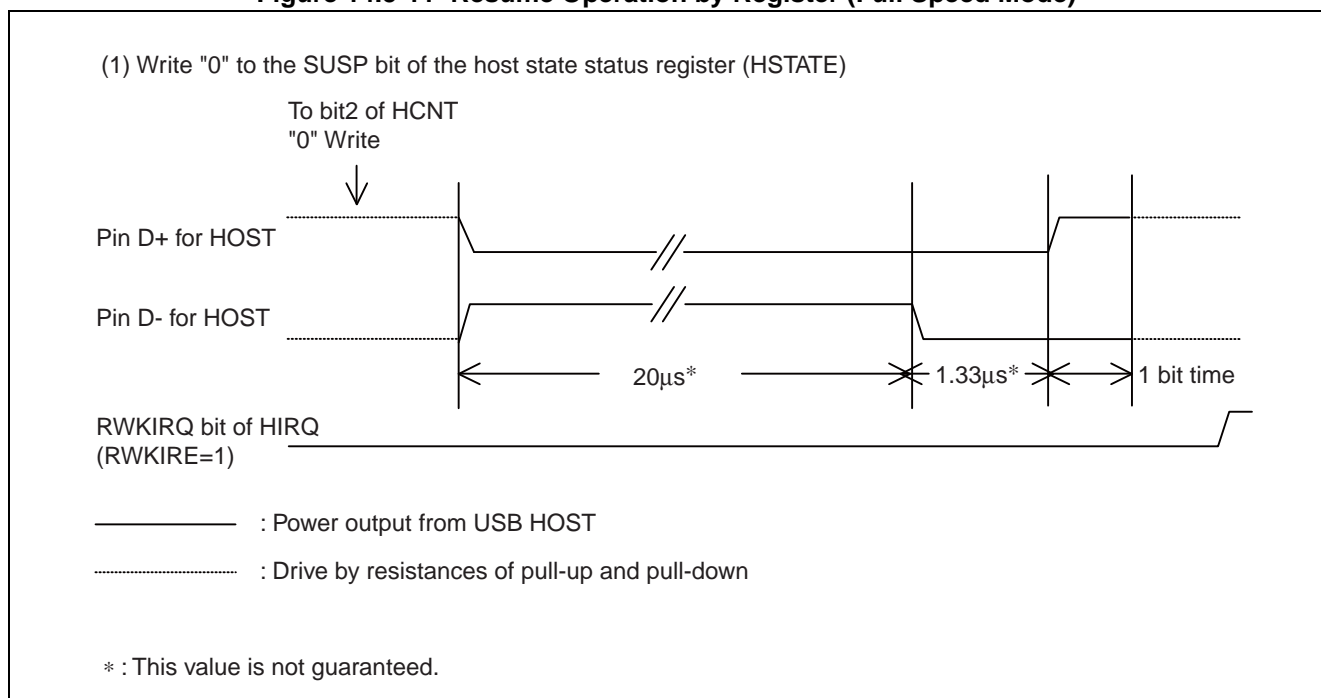
### ■ Resume Operation

Before it can start resume operation in suspend status, one of the following conditions must be true:

- (1) Write "0" to the SUSP bit of the host state status register (HSTATE).
- (2) The pins D+ and D- for HOST are detected to be k-state.
- (3) The device is detected being cut.
- (4) The device is detected being connected.

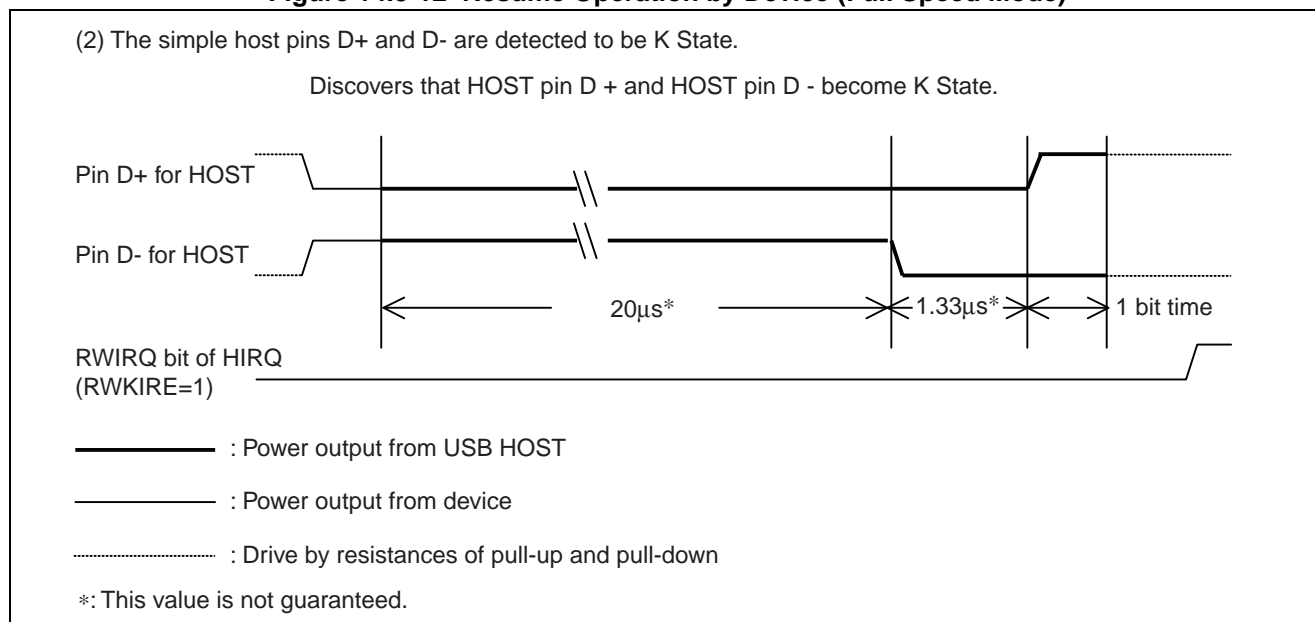
After the RWKIRQ bit of the host interrupt register (HIRQ) is set to "1", an issuance of the token is allowed. The followings show the operation timing for each condition.

**Figure 14.5-11 Resume Operation by Register (Full Speed Mode)**

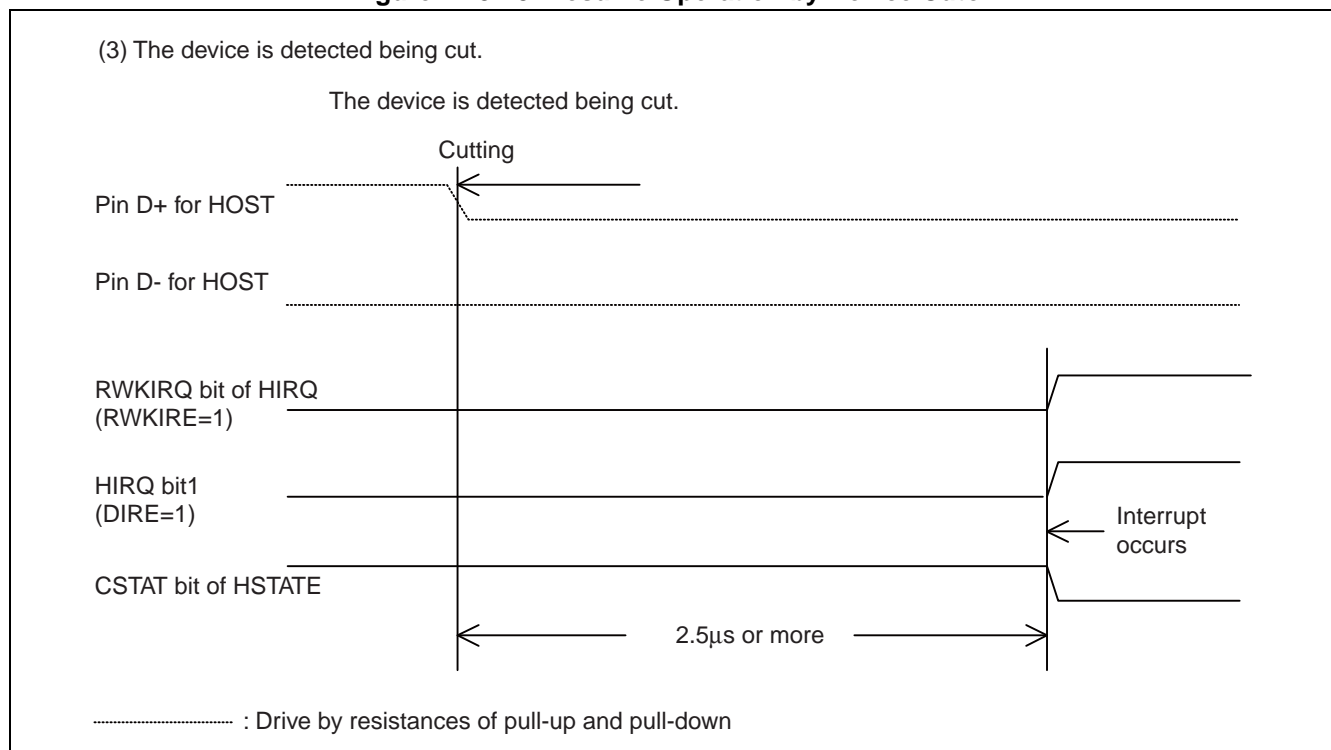




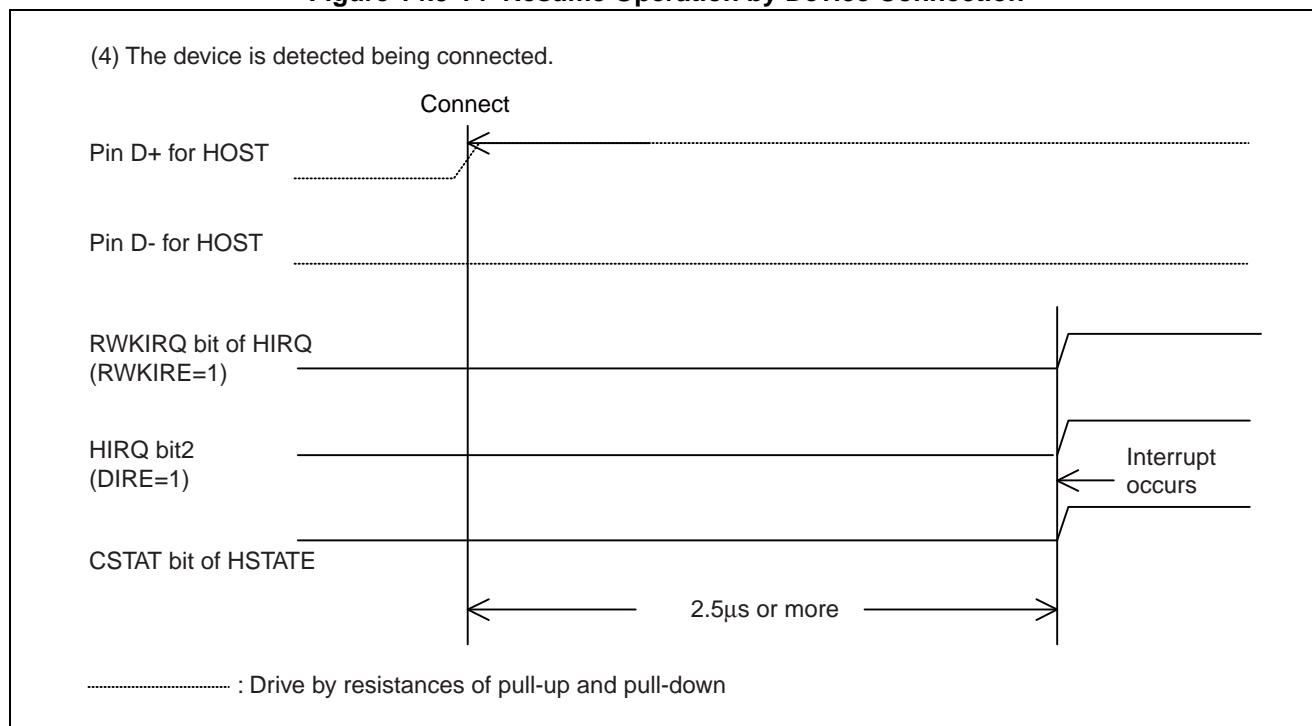
**Figure 14.5-12 Resume Operation by Device (Full Speed Mode)**



**Figure 14.5-13 Resume Operation by Device Cutoff**



**Figure 14.5-14 Resume Operation by Device Connection**



## **14.5.11 Cutting of Device**

---

**Once both HOST pins D + and D- become "L", the disconnection timer starts, and sets the CSTAT bit of the host state status register (HSTATE) to "0" when both pins detect "L" for 2.5  $\mu$ s or longer.**

---

### **■ Cutting of Device**

Regardless of Host mode and function mode, when both Host pins D+ and D- detect "L" for 2.5  $\mu$ s or longer, it determines that the device is cut. Therefore, the CSTAT bit of the host state status register (HSTATE) becomes "0", and the DIRQ bit of the host interrupt register (HIRQ) is set to "1". If the DIRE bit of the host control register 0 (HCNT0) is "1", an interrupt generates. To clear the interrupt, write "0" to the DIRQ bit of the HIRQ.

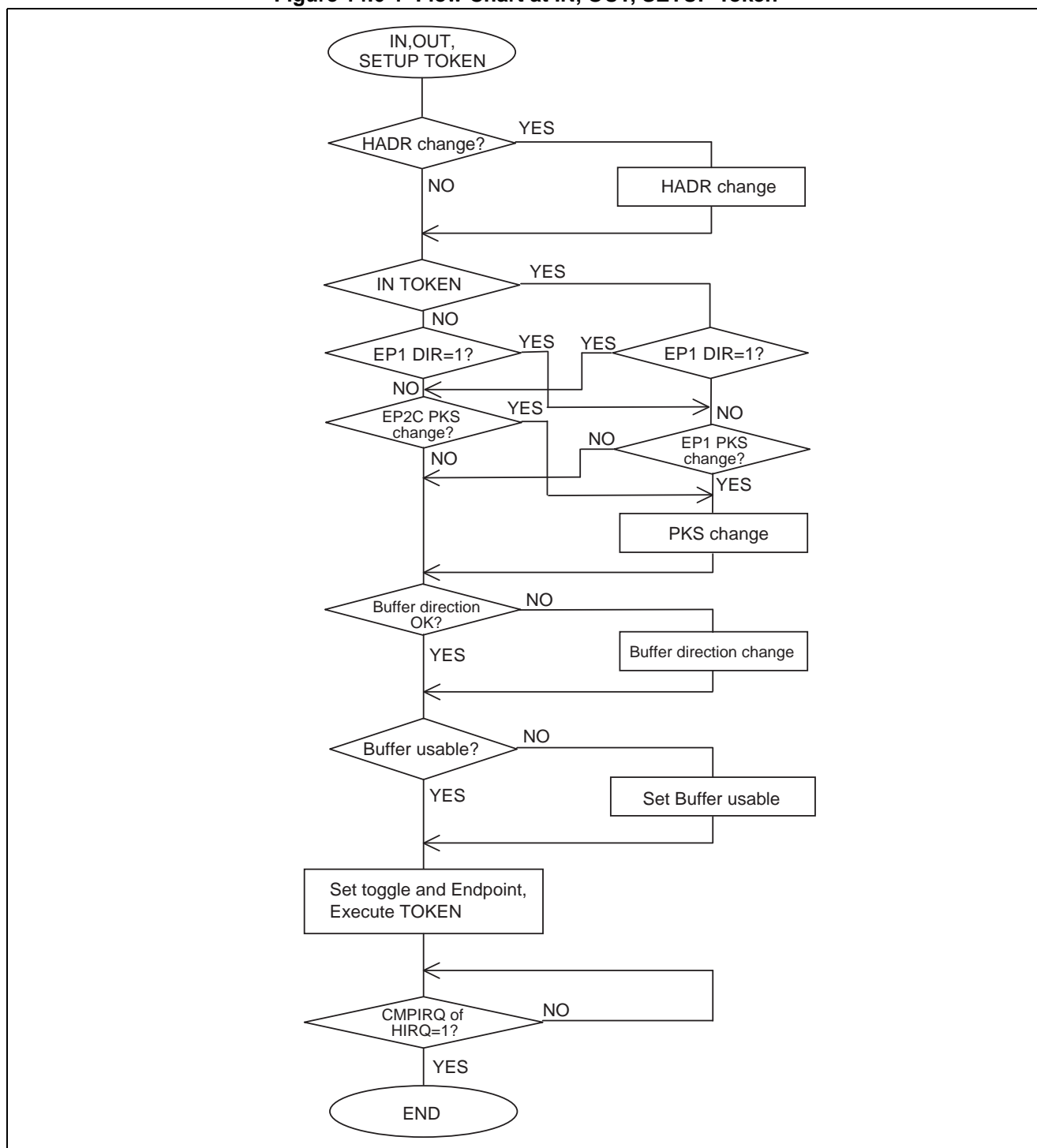
When a reset is performed on the USB bus, it determines that the device is disconnected and sets the CSTAT bit of the host state status register 0 (HSTATE) to "0", but the DIRQ bit of the host interrupt register (HIRQ) does not become "1".

## 14.6 Each Token Flow Chart of USB HOST

The flow chart of each token of USB HOST is as follows.

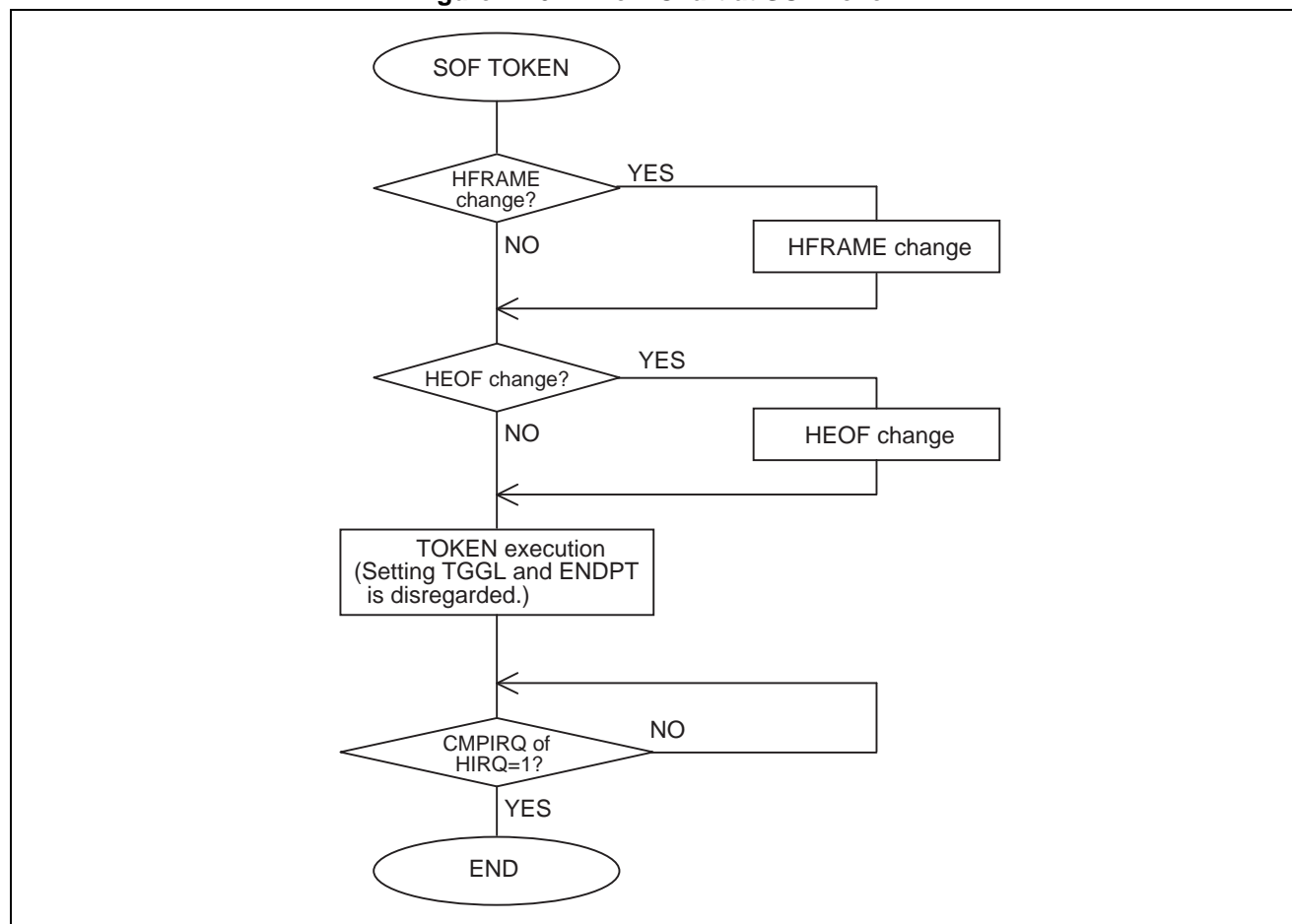
### ■ IN, OUT, SETUP Token

Figure 14.6-1 Flow Chart at IN, OUT, SETUP Token



■ SOF Token

Figure 14.6-2 Flow Chart at SOF Token



# ***CHAPTER 15***

---

## ***PWC TIMER***

**This chapter describes an overview of PWC timer, the configuration and function of register, and the PWC timer operation and precaution.**

- 15.1 Overview of PWC Timer
- 15.2 Register of PWC Timer
- 15.3 Movement of PWC Timer
- 15.4 Precautions when Using PWC Timer

## 15.1 Overview of PWC Timer

---

**The PWC timer is the multi-functional 16-bit up count timer that has the function to measure the pulse width of input signal.**

**PWC: Pulse Width Count (pulse width measurement)**

---

### ■ Function of PWC Timer

Following functions are implemented by hardware of a single channel including a 16-bit up count timer, a register to control input pulse divider and division ratio, a measurement input pin, and a 16-bit control register:

#### ● Timer Functions

- The set interval interrupt request can be generated.
- Standard internal clock can be selected from three types.

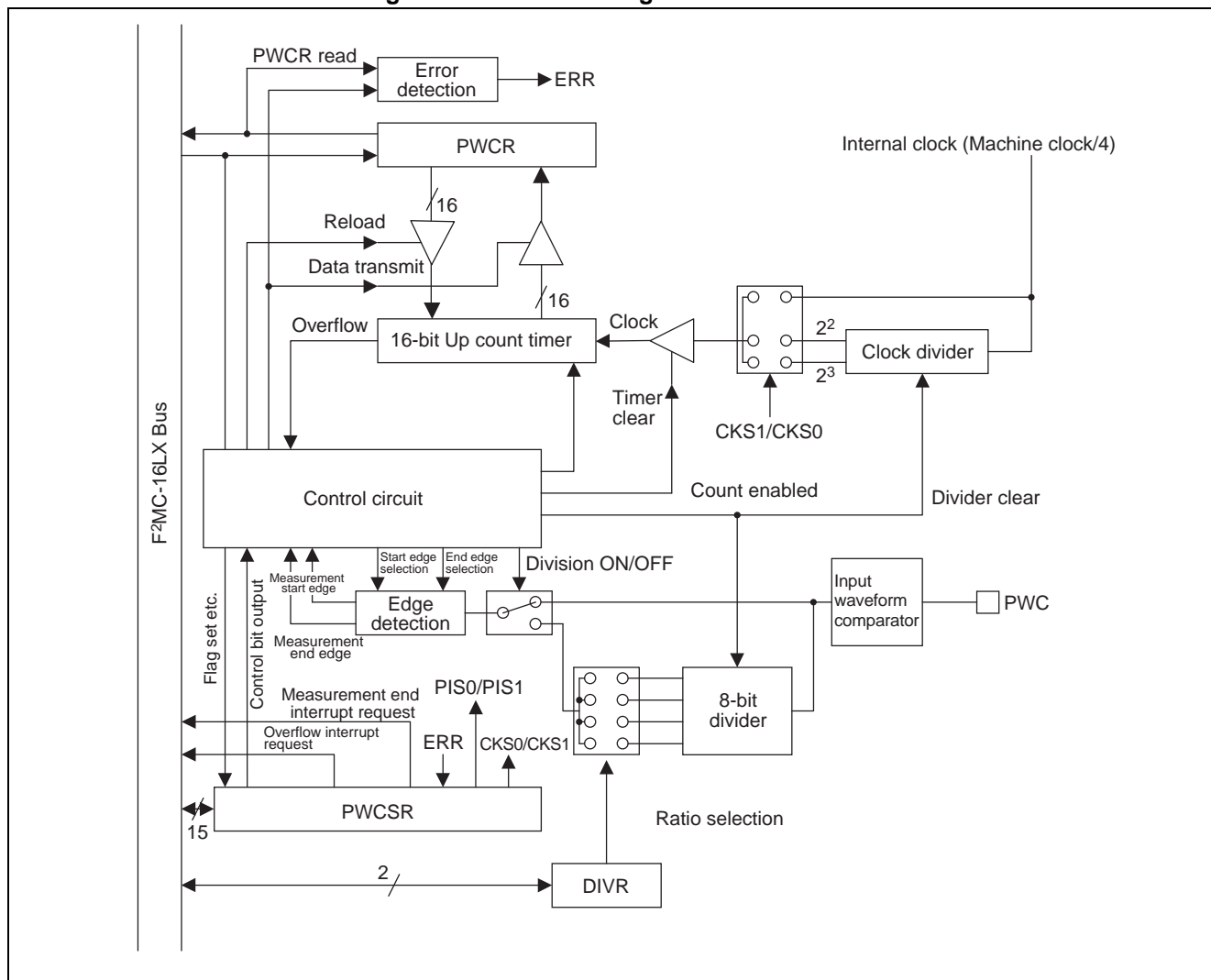
#### ● Pulse width measurement function

- The time interval between arbiter events of external pulse input is measured.
- Standard internal clock can be selected from three types.
- Each kind of measurement mode
  - "H" pulse width( $\uparrow - \downarrow$ )/"L" pulse width( $\uparrow - \downarrow$ )
  - Rising cycle( $\uparrow - \uparrow$ )/falling cycle( $\downarrow - \downarrow$ )
  - Measurement between edges( $\uparrow$  or  $\downarrow$  to  $\downarrow$  or  $\uparrow$ )
- The 8-bit input divider enables the division measurement by dividing the input pulses with a denominator of  $22n$  ( $n = 1, 2, 3, \text{ or } 4$ ).
- You can make an interrupt occur at the end of measurement.
- You can select either a single measurement or continuous measurements.

## ■ Block Diagram of PWC Timer

Figure 15.1-1 shows the PWC timer block diagram.

**Figure 15.1-1 Block Diagram of PWC Timer**





## 15.2 Register of PWC Timer

Configuration and function of the register used for PWC timer are described.

### ■ Register List of PWC Timer

Figure 15.2-1 shows the PWC timer register list.

Figure 15.2-1 Register List of PWC Timer

bit 15	8 7						0	
PWCSR								(R/W)
PWCR								(R/W)
DIVR								(R/W)

bit 15	14	13	12	11	10	9	8	PWCSR
00005DH	STRT	STOP	EDIR	EDIE	OVIR	OVIE	ERR	Reserved
	(R/W)	(R/W)	(R)	(R/W)	(R/W)	(R/W)	(R)	(R/W)
								PWC control status register
								Initial value 0000000Xb

bit 7	6	5	4	3	2	1	0	PWCSR
00005CH	CKS1	CKS0	PIS1	PIS0	S/C	MOD2	MOD1	MOD0
	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
								PWC control status register
								Initial value 00000000b

bit 15	14	13	12	11	10	9	8	PWCR
00005FH	D15	D14	D13	D12	D11	D10	D9	D8
	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
								PWC data buffer register
								Initial value 00000000b

bit 7	6	5	4	3	2	1	0	PWCR
00005EH	D7	D6	D5	D4	D3	D2	D1	D0
	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
								PWC data buffer register
								Initial value 00000000b

bit 7	6	5	4	3	2	1	0	DIVR
000060H	—	—	—	—	—	—	DIV1	DIV0
	(—)	(—)	(—)	(—)	(—)	(—)	(R/W)	(R/W)
								PWC ratio of dividing frequency control register
								Initial value -----00b

R/W : Readable/Writable  
R : Read only  
— : Undefined

## 15.2.1 PWC Control Status Register (PWCSR)

Configuration and function of PWC control status register (PWCSR) are described.

### ■ PWC Control Status Register (PWCSR)

Figure 15.2-2 shows the bit configuration of PWC control status register (PWCSR).

**Figure 15.2-2 Bit Configuration of PWC Control Status Register (PWCSR)**

bit	15	14	13	12	11	10	9	8	PWCSR
00005DH	STRT	STOP	EDIR	EDIE	OVIR	OVIE	ERR	Reserved	PWC control status register
	(R/W)	(R/W)	(R)	(R/W)	(R/W)	(R/W)	(R)	(R/W)	Initial value 0000000Xb
bit	7	6	5	4	3	2	1	0	PWCSR
00005CH	CKS1	CKS0	PIS1	PIS0	S/C	MOD2	MOD1	MOD0	PWC control status register
	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	Initial value 00000000b
R/W : Readable/Writable									
R : Read only									
— : Undefined									

The function of each bit in the PWC control status register (PWCSR) is described in the following.

#### [bit 15, bit 14] STRT, STOP (Timer Start Bit, Timer Stop Bit)

This bit controls to start, restart, and stop the 16-bit up count timer. Timer operation status is displayed when reading.

The table below shows the function of STRT and STOP bit.

**Table 15.2-1 Functions for Write Operation (Controlling the 16-bit Up Count Timer Operation)**

STRT	STOP	Operation control functions
0	0	The influence is not in the function none/the operation.
0	1	Timer start/restart (count enabled).*
1	0	Timer operation forced stop (count disabled).*
1	1	The influence is not in the function none/the operation.

\*: Enable use for clear bit instruction.

**Table 15.2-2 Functions for Read Operation (Displaying the 16-bit Up-count Timer Operation Status)**

STRT	STOP	Operation control functions
0	0	Timer under suspension (unstarted or measurement completed). [initial value]
1	1	During timer count operating (during measurement).

- Initialized to "0" at reset.
- Read and write are enabled. However, meanings are different between when writing and reading as shown in Table 15.2-1 and Table 15.2-2.
- The value read by read-modify-write instructions is always "1<sub>B</sub>" regardless of the bit value.
- Note that no bit instruction can be used for reading in the operating status (reading always produces the operating status) although the bit instruction (bit clear) can be used for either STRT or STOP bit to start or stop the timer.

**[bit 13] EDIR (measurement end interrupt request flag)**

This flag indicates the measurement termination in the pulse width measurement. When the measurement termination interrupt factor is permitted (bit 12: EDIE = 1) and this bit is set, the measurement termination interrupt request occurs.

EDIR	Operation mode
Set factor	Set when the pulse width measurement terminates (measured results are stored in PWCR).
Clear factor	Because PWCR (measurement result) is read, it is clear.

- Initialized to "0" at reset.
- Only reading is allowed.
- Bit values cannot be changed by writing.

**[bit 12] EDIE (measurement end interruption permission)**

Measurement termination interrupt request during the pulse width measurement is controlled as shown in the table below:

EDIE	Operation mode
0	Measurement end interrupt request output disabled (interrupt is not generated even if EDIR is set). [Initial value]
1	Measurement end interrupt request output enabled (interrupt is generated if EDIR is set).

- Initialized to "0" at reset.
- Reading or writing is allowed.

**[bit 11] OVIR (timer overflow interrupt request flag)**

This flag indicates the 16-bit up count timer overflowed from FFFF<sub>H</sub> to 0000<sub>H</sub>. When the timer overflow interrupt factor is permitted (bit 10: OVIE = 1) and this bit is set, the timer overflow interrupt request occurs.

OVIR	Operation mode
Set factor	Set when the overflow generates (FFFF <sub>H</sub> to 0000 <sub>H</sub> ).
Clear factor	It is clear because of "0" writing or $\mu$ DMAC.

- Initialized to "0" at reset.
- Reading and writing are allowed; however, only writing "0" is effective, while writing "1" causes no changes in the bit value.
- The read value is "1" in the read-modify write instructions regardless of the bit value.

## [bit 10] OVIE (timer overflow interrupt request permission)

Measurement termination interrupt request during the pulse width measurement is controlled as shown in the table below:

OVIE	Operation mode
0	Overflow interrupt request output disabled (interrupt is not generated even if OVIR is set). [Initial value]
1	Overflow interrupt request output enabled (interrupt is generated when OVIR is set).

- Initialized to "0" at reset.
- Reading and writing are allowed.

## [bit 9] ERR (error flag)

This flag indicates the next measurement termination before reading the measured result in the PWCR in the pulse width measurement in the continuous measurement mode. In this case, the PWCR value is updated with the new measurement result and the previous measurement result is lost. The measurement is continued regardless of the bit value.

ERR	Operation mode
Set factor	Set when measurement results that are not read yet are erased due to the following results:
Clear factor	Because PWCR (measurement result) is read, it is clear.

- Initialized to "0" at reset.
- Only reading is allowed. Write operations have no effect.

## [bit 8] Reserved bit

It is Reserved bit. Be sure to write "0".

## [bit 7, bit 6] CKS1 and CKS0 (clock selection)

Internal count clock can be selected from three types shown in Table 15.2-3.

**Table 15.2-3 Count Clock of 16 Bit Up Count Timer**

CKS1	CKS0	Operation mode
0	0	Clock of machine clock divided by 4 (0.17 $\mu$ s for 24 MHz machine clock) [Initial value]
0	1	Clock of machine clock divided by 16 (0.67 $\mu$ s for 24 MHz machine clock)
1	0	Clock of machine clock divided by 32 (1.337 $\mu$ s for 24 MHz machine clock)
1	1	Setting disabled (Undefined)

- Initialized to 00<sub>B</sub> at reset.
- Read and write are enabled. However, setting "11<sub>B</sub>" is a interdiction.

Note:

Rewriting after the startup is an interdiction. Always perform the write operation before start or after stop.

**[bit 5, bit 4] PIS1, PIS0 (pulse width measurement input pin selection)**

The pulse width measurement input pin is selected.

**Table 15.2-4 Selection of Pulse Width Measurement Input Pin**

PIS1	PIS0	Operation mode
0	0	(The pin PWC is selected). [Initial value]
0	1	Setting disabled
1	0	Setting disabled
1	1	Setting disabled (Undefined)

- Initialized to 00<sub>B</sub> at reset.
- Read and write are enabled. However, do not set "01<sub>B</sub>", "10<sub>B</sub>", and "11<sub>B</sub>".

Note:

Rewriting after the startup is an interdiction. Always perform the write operation before start or after stop.

**[bit 3] S/C (Measurement mode (single/continuous) selection)**

The measurement mode is selected.

**Table 15.2-5 Selection of Measurement Mode of 16-bit Up-count Timer**

S/C	Measurement mode selection	At timer mode	Pulse width
0	Single measurement mode [Initial value]	Reload none (single shot)	Stop after one measurement
1	Continuous measurement mode	There is reload (reload timer).	Continuous measurement: buffer register enabled

- Initialized to "0" at reset.
- Reading and writing are allowed.

Note:

Rewriting after the startup is an interdiction. Always perform the write operation before start or after stop.

**[bit 2 to bit 0] MOD2, MOD1, MOD0 (operation mode/measurement edge selection)**

Operation mode and width measurement edge are selected.

**Table 15.2-6 Selection of Operating Mode/measurement Edge of 16-bit Up-count Timer**

MOD2	MOD1	MOD0	Operation mode/measurement edge selection
0	0	0	Timer mode [Initial value]
0	0	1	Timer mode (only Reload mode)
0	1	0	Pulse width measurement mode between all edges (↑ or ↓ - ↓ or ↑)
0	1	1	Divided cycle measurement mode (input divisor enabled)
1	0	0	Cycle between rising edges measurement mode (↑ - ↑)
1	0	1	"H" pulse width measurement mode (↑ - ↓)
1	1	0	"L" pulse width measurement mode (↓ - ↑)
1	1	1	Cycle between falling edges measurement mode (↓ - ↓)

- Initialized to "00<sub>B</sub>" at reset.
- Reading and writing are allowed.

**Note:**

Rewriting after the startup is an interdiction. Always perform the write operation before start or after stop.

## 15.2.2 PWC Data Buffer Register (PWCR)

Configuration and function of PWC data buffer register (PWCR) are described.

### ■ PWC Data Buffer Register (PWCR)

Figure 15.2-3 shows the bit configuration of PWC data buffer register (PWCR).

**Figure 15.2-3 Bit Configuration of PWC Data Buffer Register (PWCR)**

bit	15	14	13	12	11	10	9	8	PWCR
00005FH	D15	D14	D13	D12	D11	D10	D9	D8	PWC data buffer register
	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	Initial value 00000000b
bit	7	6	5	4	3	2	1	0	PWCR
00005EH	D7	D6	D5	D4	D3	D2	D1	D0	PWC data buffer register
	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	Initial value 00000000b

R/W : Readable/Writable

The function of PWC data buffer register (PWCR) varies in between the timer mode and the pulse width setting mode that are set by the PWCSR register bit 2 to "0" (MOD2 to MOD0).

#### ● In the timer mode (read/write enabled)

Becomes the reload register that holds the reload data when the reload timer operates (PWCSR bit 3: S/C = 1). It is reading/writing in this case, it is possible in both writing.

Becomes a direct access to the up count timer in the one-shot timer operation mode (PWCSR bit 3: S/C = 0). Perform the write operation when the timer stops although either read or write is possible also in this case. Reading is always enabled to read the timer value in the count operation.

#### ● In the pulse width measurement mode (read only enabled)

Becomes the buffer register that holds the previous measurement result in the continuous measurement mode (PWCSR bit 3: S/C = 1).

In this case, the read only is enabled and no register value will be changed even when writing.

Becomes a direct access to the up count timer in the single mode (PWCSR bit 3: S/C = 0).

Also in this case, the read only is enabled and no register value will be changed even when writing. Reading is anytime enabled to obtain the timer value during the count operation. When the measurement is finished, the measured value is held.

Note:

Please access the PWCR register by word move operation. Be initialized to "00<sub>B</sub>" when resetting.

### 15.2.3 PWC Ratio of Dividing Frequency Control Register (DIVR)

Configuration and function of PWC Ratio of dividing frequency control register (DIVR) are described.

#### ■ PWC Ratio of Dividing Frequency Control Register (DIVR)

Figure 15.2-4 shows the bit configuration of a PWC ratio of dividing frequency control register (DIVR).

**Figure 15.2-4 Bit Configuration of PWC Ratio of Dividing Frequency Control Register (DIVR)**

bit	7	6	5	4	3	2	1	0	DIVR
000060H	—	—	—	—	—	—	DIV1	DIV0	PWC ratio of dividing frequency control register
	(—)	(—)	(—)	(—)	(—)	(—)	(R/W)	(R/W)	Initial value -----00 <sub>B</sub>

R/W : Readable/Writable  
— : Undefined

#### [bit 7 to bit 2] Undefined bit

The reading value is irregular. No effect on writing.

#### [bit 1, bit 0] DIV1, DIV0 (division ratio selection)

This register is used in the division cycle measurement mode (PWCSR bit 2.1.0: MOD2.MOD1.MOD0 = 001<sub>B</sub>) and has no meaning in the other mode else.

In the division cycle measurement mode, pulses input to the measurement pin are divided by the division ratio set in the DIVR register and a single cycle width is measured after dividing.

**Table 15.2-7 Division Ratio Selection**

DIV1	DIV0	Count clock selection
0	0	4-dividing frequency [Initial value]
0	1	16-frequency division
1	0	64-frequency division
1	1	256-frequency division

- Initialized to "00<sub>B</sub>" at reset.
- Reading and writing are allowed.

Note:

Rewriting after the startup is an interdiction. Always perform the write operation before start or after stop.



## **15.3 Movement of PWC Timer**

---

**The movement of the PPG timer is explained.**

---

### **■ Outline of PWC Timer Operation**

The PWC timer, a multi-functional timer based on the 16-bit up count timer has built-in measurement input pin, 8-bit input division, etc. PWC timer has the following two main functions:

- Timer Functions
- Pulse width count function

Either function can select three kinds of count clocks. Basic performance and operation of each function are described as follows:



## 15.3.2 Operation of Pulse Width Measurement Function

Time cycle between arbiter events of input pulse can be measured by the time.

### ■ Operation of Pulse Width Measurement Function

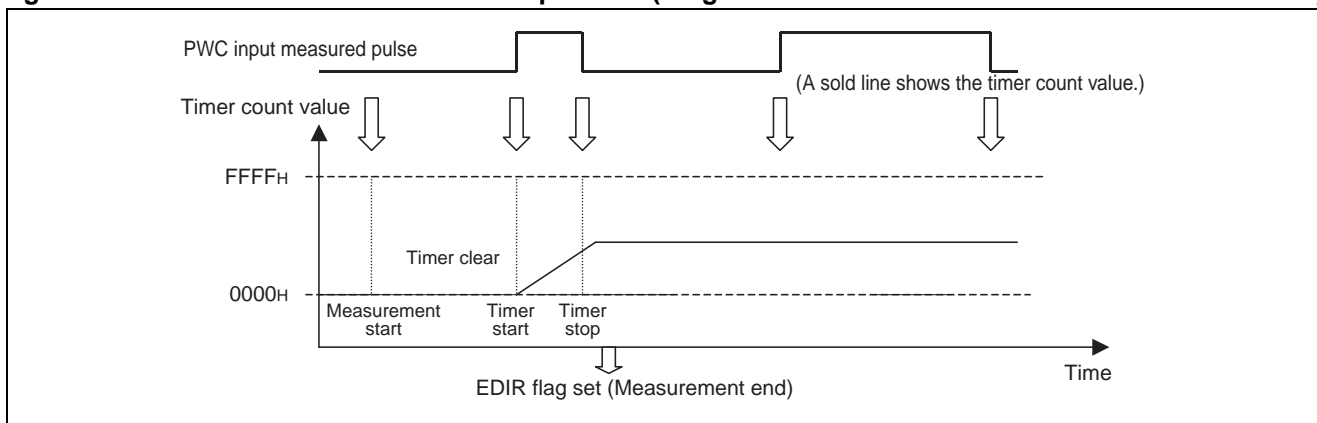
The pulse width measurement function does not start the count until the set measurement start edge is input after it is started. Starts the count up after clearing the timer to be 0000<sub>H</sub> when detecting the start edge and stops the count when detecting the stop edge. The value counted in this period is registered as the pulse width. When the measurement ends, the interruption is detected.

Operates as follows depending on the measurement mode after the measurement.

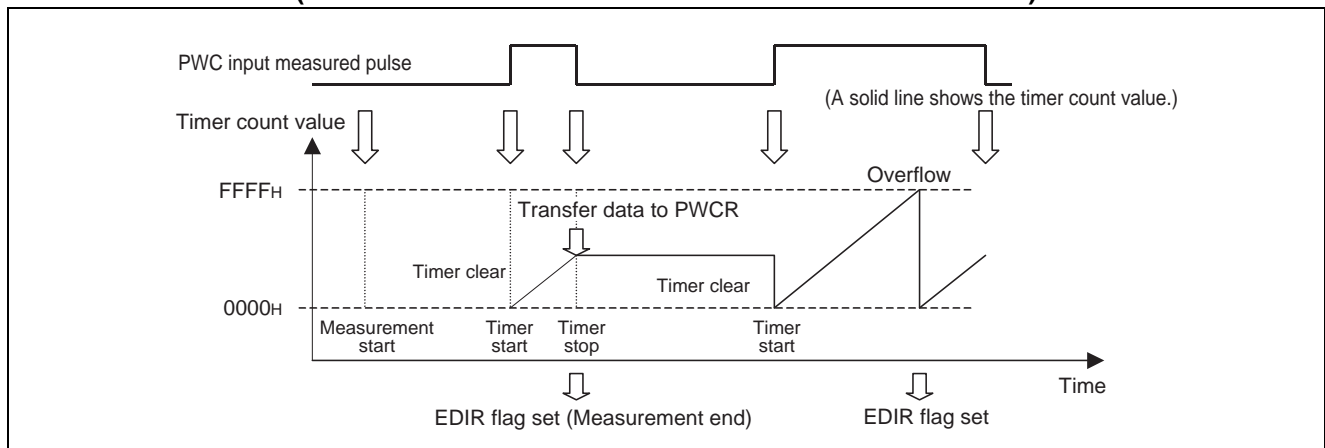
- Single measurement mode: Suspends the operation.
- Continuous measurement mode: The timer value is transferred to the buffer register and the measurement is suspended until the next start edge is input.

Figure 15.3-2 and Figure 15.3-3 show the single measurement mode operation and the continuous measurement mode operation, respectively.

**Figure 15.3-2 Pulse Width Measurement Operation (Single Measurement Mode/ "H" Width Measurement)**



**Figure 15.3-3 Pulse Width Measurement Operation (Continuous Measurement Mode/ "H" Width Measurement)**



### 15.3.3 Count Clock Selection and Operation Mode selection

---

Count clock selection and operation mode selection are described.

---

#### ■ Count Clock Selection

Timer count clock can be selected from three types of internal clock sources by setting the RWCSR bit 7 (CKS1) and bit 6 (CKS0).

Table 15.3-1 shows the count clock selection contents.

**Table 15.3-1 Count Clock Selection Contents**

PWCSR/ bit7,bit6:CKS1,CKS0	Selected internal count clock
00 <sub>B</sub>	4-division of Machine clock (0.17 μs in case of machine clock of 24 MHz) [Initial value]
01 <sub>B</sub>	16-division of Machine clock (0.67 μs in case of machine clock of 24 MHz)
10 <sub>B</sub>	32-division of Machine clock (1.33 μs in case of machine clock of 24 MHz)

- 4-division clock of machine clock is selected for the initial value after the reset.

---

Note:

Select the count clock always before starting the timer.

---

#### ■ Selects Operation Mode

Select each of operation and measurement modes by setting the PWCSR bit.

- Operation mode selection: PWCSR bit 2, 1, 0 (MOD2, MOD1, and MOD0 bits) timer mode/pulse width measurement mode selection, measurement edge decision, etc.
- Measurement mode setting: PWCSR bit 3 (S/C bit) single measurement/continuous measurement or reload/one-shot selection.

Table 15.3-2 shows the setting contents of operation mode/measurement mode.

**Table 15.3-2 Setting Contents of Operation Mode/measurement Mode Content**

Operating mode			S/C	MOD2	MOD1	MOD0
Timer		Single shot timer	0	0	0	0
		Reload timer	1	0	0	0
Pulse width measurement	↑ or ↓ - ↑ or ↓ Measurement between all edges	Single measurement: Buffer invalidity	0	0	1	0
		Continuous measurement: Buffer effective	1	0	1	0
	Measurement at cycle of dividing frequency 1-256 frequency division	Single measurement: Buffer invalidity	0	0	1	1
		Continuous measurement: Buffer effective	1	0	1	1
	↑ - ↑ Measurement of cycle between rising	Single measurement: Buffer invalidity	0	1	0	0
		Continuous measurement: Buffer effective	1	1	0	0
	↑ - ↓ "H" pulse width measurement	Single measurement: Buffer invalidity	0	1	0	1
		Continuous measurement: Buffer effective	1	1	0	1
	↓ - ↑ "L" pulse width measurement	Single measurement: Buffer invalidity	0	1	1	0
		Continuous measurement: Buffer effective	1	1	1	0
	↓ - ↓ Measurement of cycle between falling	Single measurement: Buffer invalidity	0	1	1	1
		Continuous measurement: Buffer effective	1	1	1	1

- One-shot timer is selected in the initial setting after the reset.

**Note**

Select the operation mode always before starting the timer.

### 15.3.4 Startup and Stop of Timer/Pulse Width Measurement

Start/restart/stop/forced stop of each operation are performed by using the PWCSR bit 15 and PWCSR bit 14 (STRT and STOP bits).

#### ■ Startup and Stop of Timer/Pulse Width Measurement

Functions are separated so that the STRT bit starts and restarts the timer/pulse width measurement and the STOP bit forcibly stops the measurement when "0" is written in either one of bits. However, no function is effective unless the values written in both bits are exclusive each other. When writing an instruction other than bit manipulation instructions (one-byte instruction or more), the bits combination is always limited as shown in Table 15.3-3.

**Table 15.3-3 Function of STRT Bit and STOP Bit**

STRT	STOP	Function
0	1	Startup/reactivation of timer/pulse width measurement
1	0	Compulsion stop of timer/pulse width measurement

When using a bit manipulation instruction (clear bit instruction), the hardware automatically writes it in the combination as shown in Table 15.3-3 and no care is necessary.

#### ■ Operation after Startup

Operations are as follow after starting the timer mode or the pulse width measurement mode.

##### ● Timer mode

The count operation begins at once.

##### ● Pulse width measurement mode

No count is performed until the measurement start edge is input. After the measurement start edge is detected, the 16-bit up count timer is cleared to be 0000<sub>H</sub> and the count starts.

#### ■ Reactivation

Starting (writing "0" in the STRT bit) during the operation after starting the timer/pulse width measurement mode is called as restarting.

Restarting performs the following operations depending on the mode:

##### ● Single shot timer mode

There is no influence in the operation.

##### ● Reload timer mode

Reload operates, and the operation is continued. Restarting at the same timing when an overflow occurs sets the overflow flag (OVIR).

● Pulse width measurement mode

In the state of measurement starting edge waiting there is no influence in the operation. During the measurement, the count is stopped and the measurement start edge is again waited. In this case, if the measurement termination edge detection and the restart occur at the same time, the measurement termination flag (EDIR) is set and the result is transferred to PWCR in the continuous measurement mode.

■ Stops

In the one-shot timer mode and the single measurement mode, the timer overflow or the measurement termination automatically stops the count and no intentional stop is necessary. In the other modes, however, the timer must be forcibly stopped. Similarly, if you want to stop the timer before the automatic stop, you must forcibly stop it.

■ Check Operating State

When the above-mentioned STRT and STOP bits are read, they are functional as the operating state indication bits.

Table 15.3-4 shows the function of operating state indication bit.

**Table 15.3-4 Function of Operating State Indication Bit**

STRT	STOP	Operating State
0	0	Timer stop (excluding status of waiting for measurement start edge): It is not active or shows that the measurement ended.
1	1	During timer counting or waiting for measurement start edge

---

Note:

Reading either the STRT or STOP bit provides a same value. Do not use, however, the read-modify-write instructions (bit process instructions, etc.) to read the STRT and STOP bits because "11<sub>B</sub>" always results.

---

### 15.3.5 Operation of Timer Mode

---

The operation of the timer mode is explained.

---

#### ■ Clearing Timer

The 16-bit up count timer is cleared to be 0000<sub>H</sub> in the following case:

- At a reset
- In the pulse width measurement mode, when the measurement start edge is detected and the count is started.

#### ■ One-shot Operation Modes.

In the one-shot operation mode, the count up is performed at every count clock after the timer started and an overflow occurrence at counting from FFFF<sub>H</sub> to 0000<sub>H</sub> automatically stops the count. When a value is set in the PWCR before starting the timer, the count is started from the value. In this case, the set value is not held and the PWCR value indicates the current count value.

#### ■ Reload Operation Mode

In the reload operation mode, the count up is performed at every count clock after the reload value in the PWCR is set in the timer after starting the timer. Overflow occurrence at counting from FFFF<sub>H</sub> to 0000<sub>H</sub> again sets the reload value in the PWCR in the timer (reload operation) to repeat the count operation. The timer does not stop until it is forcibly stopped by writing into the PWCSR STOP bit or it is reset. The value set in the PWCR before starting the timer is held as the reload value during the count operation and always set in the timer at starting/restarting and an overflow occurrence. Use a newly changed reload value to change the set value during the count operation at the next overflow occurrence or the timer restarting.

#### ■ Timer Value and Reload Value

The PWCR in the one-shot operation mode has a direct access to the up count timer. The value written in the PWCR is written in the timer as it is so that you can acquire the timer value in the count operation by reading the PWCR value during the timer operation. If you write an arbiter value in the PWCR before starting the timer, the count operation is started from this specified value. In the reload operation mode, accessing the up count timer is impossible. The PWCR functions as the reload register to hold the reload value. When the timer start, restart, or overflow occurs, the value written in the PWCR always set to the timer. When PWCR is read, the stored reload value is read.

The PWCR value and the timer value are undefined when the timer is set to the one-shot operation mode after forcibly suspending the reload operation mode. Therefore, always write the value before using the timer.

#### ■ Interrupt Generation Request

In timer mode operation, interrupt request by the timer overflow can be generated. An interrupt request is generated when the overflow flag is set at an overflow occurrence caused by the count up and the overflow interrupt request is permitted.



## ■ Timer Cycle

If 0000<sub>H</sub> is set to the PWCR in the one-shot operation mode and the timer is started, an overflow occurs after 65536 times counted up to stop the count. The time period from the start to the stop is calculated by the following expression:

$$T_1 = (65536 - n_1) \times t$$

$T_1$ : Time from startup to stop (μs)

$n_1$ : Timer value written in the PWCR at the start

$t$ : Count Clock cycle (μs)

If 0000<sub>H</sub> is set to the PWCR in the reload operation mode and the timer is started, an overflow occurs at every 65536-time counted up. Reload cycle time is calculated by the following expression:

$$T_R = (65536 - n_R) \times t$$

$T_R$ : Reload cycle (overflow cycle) (μs)

$n_R$ : Reload value held in the PWCR

$t$ : Count Clock cycle (μs)

## ■ Count Clock and Maximum Cycle

Maximum cycle is provided when 0000<sub>H</sub> is set for the PWCR value in the timer mode.

The count clock cycle and the maximum cycle of the timer are shown in Table 15.3-5 when the machine clock frequency (called F hereafter) is 24 MHz.

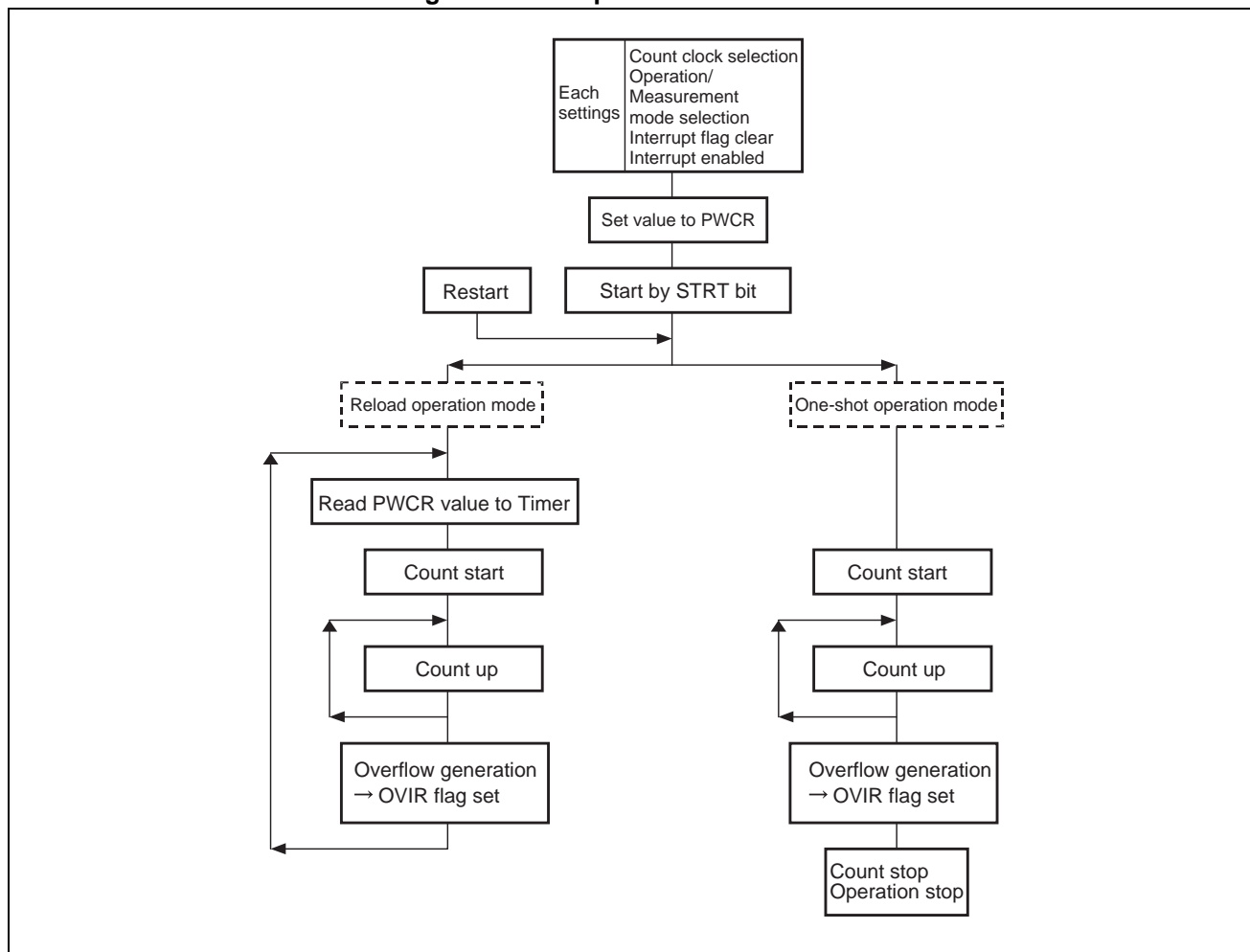
**Table 15.3-5 Count Clock and Cycle**

Count clock selection	in CSK1,CSK0=00:(φ/4)	in CSK1,CSK0=00:(φ/16)	in CSK1,CSK0=00:(φ/32)
Count Clock Cycle	0.17 μs	0.67 μs	1.33 μs
Timer maximum cycle	10.92 ms	43.7 ms	87.4 ms

## ■ Operation Flow of Timer

Figure 15.3-4 shows the timer operation flow.

**Figure 15.3-4 Operation Flow of Timer**



## **15.3.6 Operation of Pulse Width Measurement Mode**

---

**Operation of pulse width measurement mode is described.**

---

### **■ Single Measurement and Continuous Measurement**

Pulse width measurement modes include a mode to perform only one-time measurement and a mode to perform continuous measurements. Each mode is selected by the PWCSR S/C bits (see Section "15.3.3 Count Clock Selection and Operation Mode selection").

#### ● One-time measurement mode

When the first measurement termination edge is input, the timer counter stops and the measurement termination flag (EDIR) in the PWCSR is set not to perform measurement anymore (however, when the restart occurs at the same time, the measurement start is waited).

#### ● Continuous measurement mode

When the measurement termination edge is input, the timer counter stops and the measurement termination flag (EDIR) in the PWCSR is set to stop the count until the measurement start edge is again input. When the measurement start edge is again input, the measurement is started after the timer is cleared to 0000<sub>H</sub>. At the measurement termination, the measured result in the timer is transferred to the PWCR.

---

Note:

Always change the measurement mode selection when the timer stops.

---

### **■ Data of Measurement Result**

Measured results, the timer value handling, and the PWCR function are different between the single measurement mode and the continuous measurement mode. The measurement result in both modes is as follows:

#### ● One-time measurement mode

- When the PWCR is read during the operation, the timer value in the measurement is acquired.
- When the PWCR is read after the measurement termination, the measured result data is acquired.

#### ● Continuous measurement mode

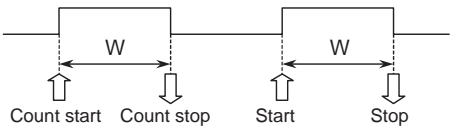
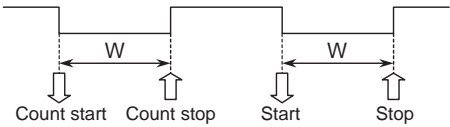
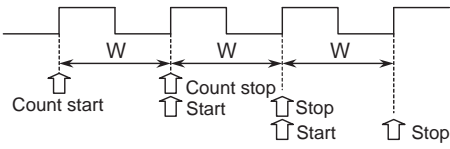
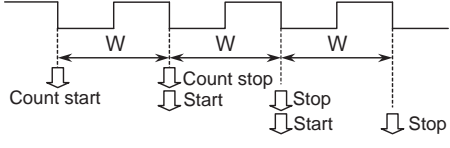
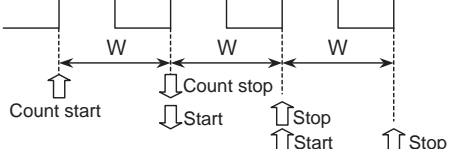
- At the measurement termination, the measured result in the timer is transferred to the PWCR.
- When the PWCR is read, the last measurement result is acquired and the previous measurement result is held even in the measurement operation. The timer value under the measurement cannot be read.

When the next measurement is terminated before reading the measured result in the continuous measurement mode, the previous measurement result is erased by a new measurement result. In this case, the gill in PWCSR-Flag (ERR) is set. Error flag (ERR) is automatically cleared when reading the PWCR.

## ■ Measurement Mode and Counter Operation

The measurement mode can be selected from six types depending on the place where the input pulse is measured. The cycle measurement mode is also prepared to arbitrarily divide the input pulse for high-precision measurement of higher frequency pulse width. Table 15.3-6 shows the measurement mode list.

**Table 15.3-6 The Measurement Mode List (1 / 2)**

Measurement mode	MOD2	MOD1	MOD0	Measurement mode measured target (W: width of a measured pulse)
"H" pulse width measurement	1	0	1	 <p>The width at "H" period is measured. Count (measurement) start: when detecting a rising edge Count (measurement) end: when detecting a falling edge</p>
"L" pulse width measurement	1	1	0	 <p>The width at "L" period is measured. Count (measurement) start: when detecting a falling edge Count (measurement) end: when detecting a rising edge</p>
Rising Between edges Measurement at cycle	1	0	0	 <p>A cycle between rising edges is measured. Count (measurement) start: when detecting a rising edge Count (measurement) end: when detecting a rising edge</p>
Falling Between edges Measurement at cycle	1	1	1	 <p>A cycle between falling edges is measured. Count (measurement) start: when detecting a falling edge Count (measurement) end: when detecting a falling edge</p>
Between all edges pulse width measurement	0	1	0	 <p>The width between the edges continuously input is measured. Count (measurement) beginning: When edge is detected Count (measurement) end: When edge is detected</p>

**Table 15.3-6 The Measurement Mode List (2 / 2)**

Measurement mode	MOD2	MOD1	MOD0	Measurement mode measured target (W: width of a measured pulse)
Measurement at cycle of dividing frequency	0	1	1	<p>For only dividing ratio selected by division setting register DIVR The input pulse is divided and the cycle is measured. Count (measurement) start: when detecting a rising edge immediately after counting starts Count (measurement) end: when one cycle ends after it is divided</p>

No timer count is performed from start of the count to input of the count start edge in any modes. The timer is cleared to 0000<sub>H</sub> after inputting the count start edge and the timer count is performed at every count clock until the count termination edge is input. When the count termination edge is input, the next operation is executed.

1. Measurement ending flag (EDIR) in PWCSR is set.
2. The timer count operation stops (excluding when it is at the same time of restarting).
3. Continuous measurement mode: The timer value (measured result) is transferred to the PWCR and the count is suspended until the next measurement start edge is input.
4. Single measurement mode: The measurement is terminated (excluding when it is at the same time of restarting).

When the pulse width measurement between all edges or the division measurement is performed in the continuous measurement mode, the termination edge is the next measurement edge.

### ■ Minimum Input Pulse Width

There are following limitations for pulses that can be input to width measurement input pins (PWC).

Pulse width must be 4-machine cycle (0.17 μs for the 24-MHz machine clock) or more.

### ■ Calculation Method of Pulse Width/cycle

Pulse width/cycle to be measured can be calculated by the following expression:

$$T_W = n \times t / D_{IV} \text{ (}\mu\text{s)}$$

$T_W$ : pulse width to be measured/cycle (μs)

$n$ : Measurement result data in PWCR

$t$ : Count Clock cycle (μs)

$D_{IV}$ : Division ratio selected by the division ratio register DIVR

(Frequency of dividing frequency measurement mode)

## ■ Range of Count at Pulse Width/cycle

Measurable ranges of pulse width/cycle vary depending on the selected combination of division ratios of count clock and input divider.

Table 15.3-7 shows the measurement range list of machine clock when the clock frequency (called  $\phi$  hereafter) is 24 MHz.

**Table 15.3-7 Pulse Width Measurement Range List**

Divide ratio	DIV1	DIV0	in CKS1,CKS0=00( $\phi/4$ )	in CKS1,CKS0=01( $\phi/16$ )	in CKS1,CKS0=10( $\phi/32$ )
None	-	-	0.17 $\mu$ s to 10.92 ms (0.25 $\mu$ s)	0.17 $\mu$ s to 43.7 ms (1.6 $\mu$ s)	0.17 $\mu$ s to 87.4 ms (3.2 $\mu$ s)
4-frequency division	0	0	0.17 $\mu$ s to 2.73 ms (6.25 $\mu$ s)	0.17 $\mu$ s to 10.92 ms (0.4 $\mu$ s)	0.17 $\mu$ s to 21.85 ms (0.8 $\mu$ s)
16-frequency division	0	1	0.17 $\mu$ s to 683 $\mu$ s (15.6 ns)	0.17 $\mu$ s to 2.73 ms (1.6 $\mu$ s)	0.17 $\mu$ s to 5.46 ms (0.2 $\mu$ s)
64-frequency division	1	0	0.17 $\mu$ s to 170 $\mu$ s (3.91 $\mu$ s)	0.17 $\mu$ s to 683 $\mu$ s (25 $\mu$ s)	0.17 $\mu$ s to 1.36 ms (50 ns)
256-frequency division	1	1	0.17 $\mu$ s to 43 $\mu$ s (0.98 $\mu$ s)	0.17 $\mu$ s to 171 $\mu$ s (6.25 $\mu$ s)	0.17 $\mu$ s to 341 $\mu$ s (12.5 ns)

Note: These values enclosed in brackets ( ) indicate the granularity per bit

## ■ Interrupt Request Generation

Two following interrupt requests can be generated in the pulse width measurement mode:

### ● Interrupt request by overflow of timer

When an overflow is generated by the count up during the measurement, the overflow flag is set. When the overflow interrupt request is permitted, an interrupt request occurs.

### ● Interrupt request by measurement termination

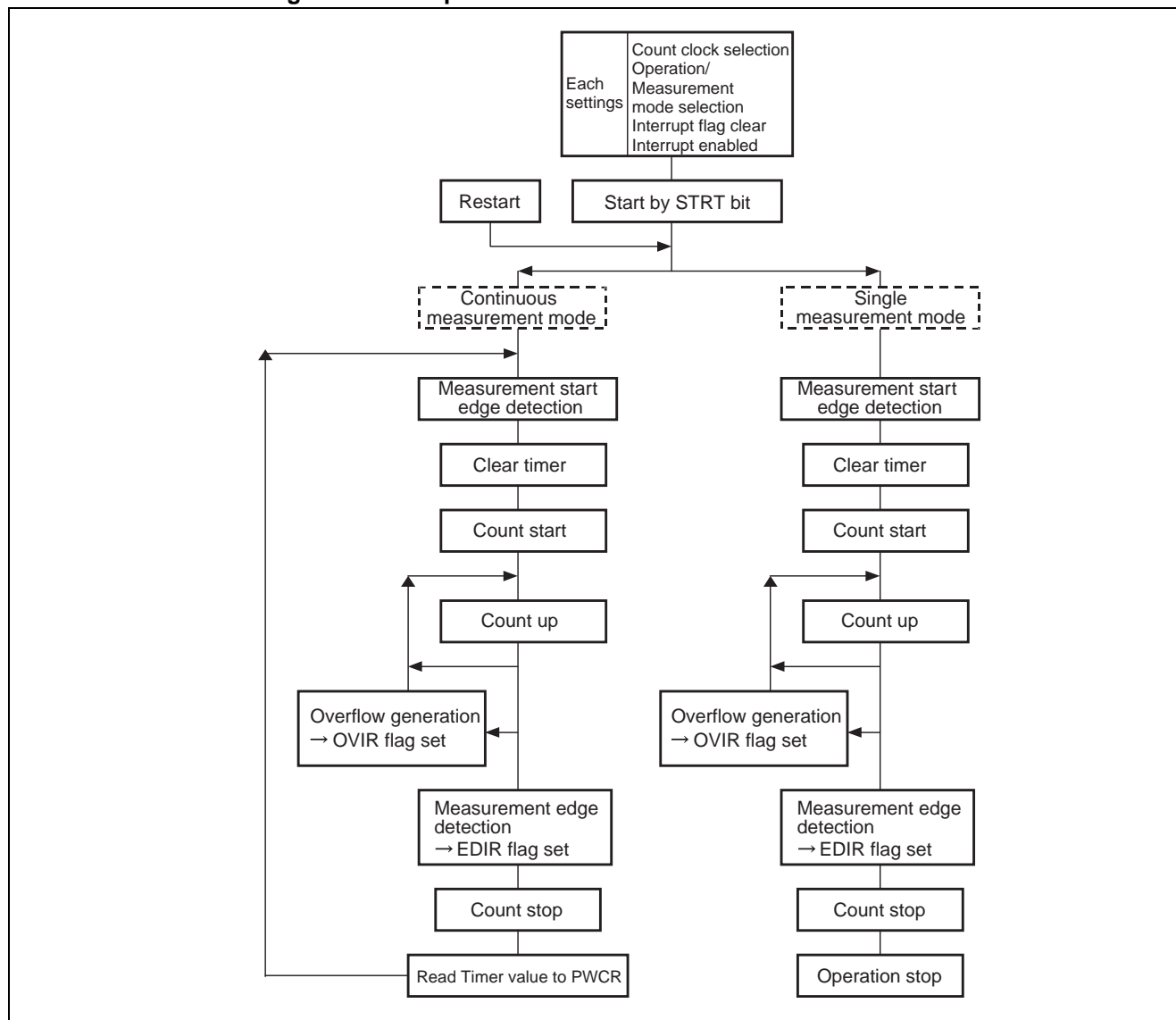
When the measurement termination edge is detected, the measurement termination flag (EDIR) in the PWCSR is set. If the measurement termination interrupt request is permitted, the interrupt request occurs.

The measurement termination flag (EDIR) is automatically cleared as soon as the measured result PWCR is read.

## ■ Operation Flow of Pulse Width Measurement

Figure 15.3-5 shows the pulse width measurement operation flow.

Figure 15.3-5 Operation Flow of Pulse Width Measurement



## 15.4 Precautions when Using PWC Timer

---

Precautions when using the PWC timer are described.

---

### ■ Precautions when Using PWC Timer

#### ● Notes concerning rewriting register

Following bits among PWCSRs are inhibited to be updated during the operation. Always update the bits before starting or after stopping the operation.

- (Bit 7, bit 6) CKS1, CKS0 (clock selection)
- (Bit 5, bit 4) PIS1, PIS0 (input signal selection)
- (Bit 3) S/C (Measurement mode (single/continuous) selection)
- (Bit 2 to bit 0) MOD2, MOD1, MOD0 (operation mode/measurement edge selection)

It is inhibited to update the DIVR during the operation. Always update the bits before starting or after stopping the operation.

#### ● Handling of measurement ending flag of timer mode

Measurement termination interrupt request flag (EDIR) value in the PWCSR has no meaning in the timer mode. Therefore, always set "0" to the measurement termination interrupt request bit (EDIE) in the PWCSR, when using in the timer mode.

#### ● Treatment of STRT and STOP bit in PWCSR

Note that both bits have different meaning between when writing and reading (see Section "15.2.1 PWC Control Status Register (PWCSR)"). In addition, the read value is "11<sub>B</sub>" when using a read-modify-write instruction regardless of the bit value. Therefore, note that no bit process instruction can be used to read the operating status (reading always produces the operating status). When writing to the STRT and STOP bits to start/stop the timer, bit process instruction (bit clear instruction, etc.) can be used for the respective bits.

#### ● Clearing Timer

Since the timer is cleared by the measurement start edge in the pulse width measurement mode, the data present in the timer before starting become invalid.

#### ● Value of PWCR and timer when mode is changed

- Always write the value before using the timer because the held value in the PWCR and the timer value become undefined when they are used in the reload timer mode before forcibly stopping the timer to switch to the one-shot timer mode.
- Always write the value before the use because the PWCR value becomes undefined if it is used in the one-shot timer mode.
- Always reset the value to the PWCR before starting the operation when switching the pulse width measurement mode to the timer mode.



● Minimum pulse width

There are following limitations for pulses that can be input to width measurement input pins:

- Minimum pulse width: 2 divisions of machine clock (0.25 $\mu$ s or more for 16-MHz machine clock)
- Minimum input frequency: 4 divisions of machine clock (4 MHz or less for 16-MHz machine clock)

When the input pulse width is smaller or the input pulse frequency is higher than the above description, the operation cannot be guaranteed. If there is a possibility of such noise in the input signal, use, for example, a filter outside of the chip to eliminate the noise before the signal is input.

● Frequency of dividing frequency measurement mode

Note that the pulse width obtained from the measured result calculation is the average value because the input pulses are divided in the division cycle measurement mode among the pulse width measurement modes.

● Handling of clock select bit

Do not set "11<sub>B</sub>" to (Bit 7, bit 6) CKS1, CKS0 (clock selection) in the PWCSR.

● Reactivation under operation

When restarting after starting the count operation, following cases may occur depending on the timing:

- When restarting at the same time of an overflow occurrence in the reload timer mode, the overflow flag (OVIR) is set.
- When restarting at the same time of measurement termination edge input in the pulse width single measurement mode, the measurement start edge is waited and the measurement termination edge (EDIR) is set.
- When restarting at the same time of measurement termination edge input in the pulse width single measurement mode, the measurement start edge is waited and the measurement termination edge (EDIR) is set and the current measured result is transferred to the PWCR.

● When the PWC timer is used in "the "H" pulse-width or "L" pulse-width measurement mode in the continuous measurement mode":

The pulse width measurement is completed, the next pulse width measurement start is waited, the timer operation is unexpectedly continued, and the overflow flag (OVIR) of timer may be sometime set before starting the next pulse width measurement. In this case, even when no overflow occurs at the next pulse width measurement termination, the overflow flag has been set. Therefore, do not use the overflow flag when using the PWC timer in the "H" pulse width or "L" pulse width measurement mode in the continuous measurement mode.

# **CHAPTER 16**

---

## ***16-BIT RELOAD TIMER***

**This chapter describes an overview of 16-bit reload timer, the configuration and functions of register and the 16-bit reload timer operation.**

16.1 Overview of 16-bit Reload Timer

16.2 Registers of 16-bit Reload Timer

16.3 Movement of 16-bit Reload Timer

## **16.1 Overview of 16-bit Reload Timer**

---

**The 16-bit reload timer provides two functions either one of which can be selected, the internal clock that performs the count down by synchronizing with 3-type internal clocks and the event count mode that performs the count down by detecting the arbiter edge of pulses input to the external pin.**

---

### **■ Overview of 16-bit Reload Timer**

Underflow of the 16-bit reload timer is defined as a case when the count value becomes from 0000<sub>H</sub> to FFFF<sub>H</sub>. Therefore, when the equation (reload register setting value + 1) holds, an underflow occurs. Either mode can be selected for the count operation from the reload mode which repeats the count by reloading the count setting value at the underflow occurrence or the one-shot mode which stops the count at the underflow occurrence. The interrupt can be generated at the counter underflow occurrence so as to correspond to the DTC.

## 16.1.1 Function of 16-bit Reload Timer

This section describes overview and Function of 16-bit reload timer.

### ■ Operation Modes of 16-bit Reload Timer

Clock Mode	Counter operation	16-bit reload timer operation
Internal clock	Reload mode	Software trigger operation
	One-shot mode	External trigger operation External gate input operation
Event count mode (External clock Mode)	Reload mode	Software trigger operation
	One-shot mode	

### ■ Internal Clock Mode

One type can be selected for the count clock from 3 types of internal clocks.

#### ● Software trigger operation

When 1 is written to TGR bit of timer control status register (TMCSR0 to TMCSR2), counter operation is started. Trigger input using the TRG bit is valid for the external trigger input and the external gate input.

#### ● External trigger operation

When the selected edges (rising edge, falling edge, or both edges) are input to the TIN0/TIN1/TIN2 pin, the count operation is started.

#### ● External gate input operation

While the selected signal ("L" or "H") is being input to TIN0/TIN1/TIN2, the count operation is continued.

### ■ Event Count Mode (External Clock Mode)

This function count downs at the edge when the selected edges (rising edge, falling edge, or both edges) are input to the TIN0/TIN1/TIN2 pin.

External clock of a constant cycle can be also used for the interval timer.

## ■ Counter Operation Mode

### ● Reload mode

When an underflow ( $0000_H \rightarrow FFFF_H$ ) occurs during the count down, the count setting value is reloaded to continue the count operation. Interrupt request that can be generated at the underflow occurrence can be also used as an interval timer. The toggle waveform which reverses at every underflow occurrence can be also output from the TOT0/TOT1/TOT2 pin.

Count Clock	Count Clock Cycle	Time of interval
Internal clock	$2^1 / \phi$ (0.083 $\mu$ s)	0.083 $\mu$ s to 5.461 ms
	$2^3 / \phi$ (0.33 $\mu$ s)	0.33 $\mu$ s to 21.845 ms
	$2^5 / \phi$ (1.33 $\mu$ s)	1.33 $\mu$ s to 87.38 ms
External clock	$2^3 / \phi$ (0.5 $\mu$ s)	0.5 $\mu$ s or more

$\phi$ : Machine clock values in parentheses ( ) are when machine clock is in 24 MHz operating.

### ● One-shot mode

When an underflow ( $0000_H \rightarrow FFFF_H$ ) occurs during the count down, the count operation is stopped.

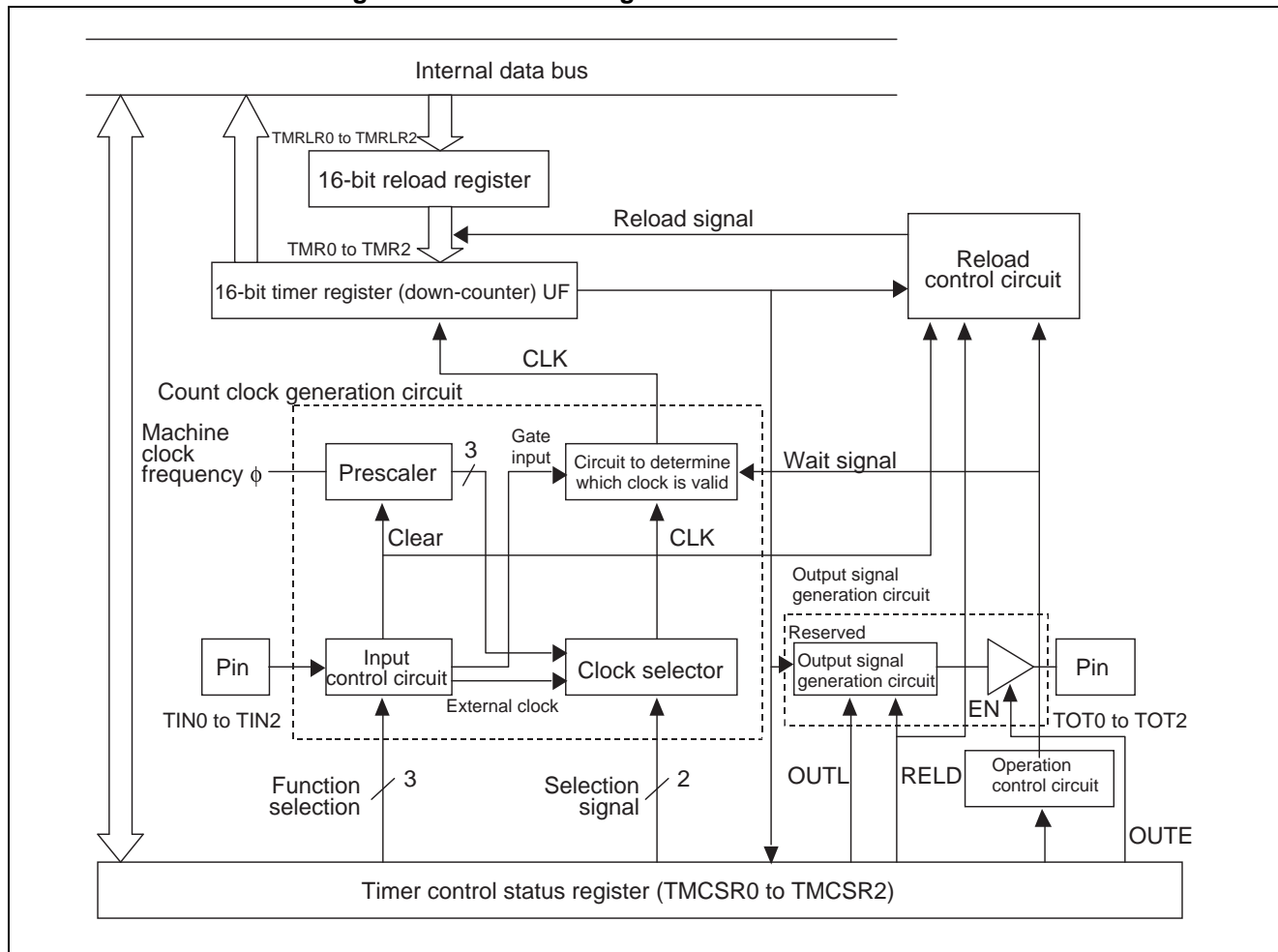
The interruption can be generated by the underflow. Short waveform which indicates the count operation can be also output from the TOT0/TOT1/TOT2 pin during the count operation.

**MB90330A Series****16.1.2 Block Diagram of 16-bit Reload Timer**

Block Diagram of 16-bit Reload Timer is shown.

■ Block Diagram of 16-bit Reload Timer

Figure 16.1-1 Block Diagram of 16-bit Reload Timer



## 16.2 Registers of 16-bit Reload Timer

Configuration and functions of register used for the 16-bit reload timer are described.

### ■ Register List of 16-bit Reload Timer

Figure 16.2-1 is shown the list of the register of 16-bit reload timer.

**Figure 16.2-1 List of Register of 16-bit Reload Timer**

	bit	15	14	13	12	11	10	9	8	TMCSR0 to TMCSR2
ch.0 : 000063 <sub>H</sub>		—	—	—	—	CSL1	CSL0	MOD2	MOD1	Timer control status register (Upper)
ch.1 : 000067 <sub>H</sub>		(—)	(—)	(—)	(—)	(R/W)	(R/W)	(R/W)	(R/W)	Read/Write
ch.2 : 00006B <sub>H</sub>		(X)	(X)	(X)	(X)	(0)	(0)	(0)	(0)	Initial value
	bit	7	6	5	4	3	2	1	0	TMCSR0 to TMCSR2
ch.0 : 000062 <sub>H</sub>		MOD0	OUTE	OUTL	RELD	INTE	UF	CNTE	TRG	Timer control status register (Lower)
ch.1 : 000066 <sub>H</sub>		(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	Read/Write
ch.2 : 00006A <sub>H</sub>		(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	Initial value
	bit	15	14	13	12	11	10	9	8	TMR0 to TMR2/TMRLR0 to TMRLR2
ch.0 : 000065 <sub>H</sub>		D15	D14	D13	D12	D11	D10	D09	D08	16-bit timer register/ 16-bit reload register (Upper)*
ch.1 : 000069 <sub>H</sub>		(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	Read (TMR0 to TMR2)
ch.2 : 00006D <sub>H</sub>		(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	Write (TMRLR0 to TMRLR2)
		(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	Initial value
	bit	7	6	5	4	3	2	1	0	TMR0 to TMR2/TMRLR0 to TMRLR2
ch.0 : 000064 <sub>H</sub>		D07	D06	D05	D04	D03	D02	D01	D00	16-bit timer register/ 16-bit reload register (Lower)*
ch.1 : 000068 <sub>H</sub>		(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	Read (TMR0 to TMR2)
ch.2 : 00006C <sub>H</sub>		(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	Write (TMRLR0 to TMRLR2)
		(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	Initial value

\* : This functions as 16-bit timer register (TMR0 to TMR2) at reading,  
This functions as 16-bit timer register (TMRLR0 to TMRLR2) at writing.

**MB90330A Series****16.2.1 Timer Control Status Register 0 to 2 (TMCSR0 to TMCSR2)**

Configuration and functions of timer control status registers 0 to 2 (TMCSR0 to TMCSR2) are described.

**■ Timer Control Status Register 0 to 2 (TMCSR0 to TMCSR2)**

The timer control status registers 0 to 2 (TMCSR0 to TMCSR2) control the operation mode and interrupt of 16-bit reload timer. Bits other than UF/CNTE/TRG are modified at CNTE=0.

Figure 16.2-2 shows the bit configuration of timer control status registers 0 to 2 (TMCSR0 to TMCSR2).

**Figure 16.2-2 Bit Configuration of Timer Control Status Registers 0 to 2 (TMCSR0 to TMCSR2)**

Address: bit	15	14	13	12	11	10	9	8	
ch.0 : 000063 <sub>H</sub>	—	—	—	—	CSL1	CSL0	MOD2	MOD1	TMCSR0 to TMCSR2 (upper)
ch.1 : 000067 <sub>H</sub>	(X)	(X)	(X)	(X)	(R/W)	(R/W)	(R/W)	(R/W)	Initial value
ch.2 : 00006B <sub>H</sub>									XXXX0000 <sub>B</sub>
Address: bit	7	6	5	4	3	2	1	0	
ch.0 : 000062 <sub>H</sub>	MOD0	OUTE	OUTL	RELD	INTE	UF	CNTE	TRG	TMCSR0 to TMCSR2 (lower)
ch.1 : 000066 <sub>H</sub>	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	Initial value
ch.2 : 00006A <sub>H</sub>									00000000 <sub>B</sub>
R/W : Readable/Writable									
X : Undefined value									
— : Undefined									

The functions of each bit of timer control status registers 0 to 2 (TMCSR0 to TMCSR2) are described in the following:

**[bit 15 to bit 12] Undefined bit**

The reading value is irregular. No effect on writing.

**[bit 11, bit 10] CSL1, CSL0 (clock selection)**

Clock source is selected by count clock selection.

CSL1	CSL0	Clock source (machine clock $\phi$ =24 MHz time)
0	0	$\phi/2^1$ (0.083 $\mu$ s) [Initial value]
0	1	$\phi/2^3$ (0.33 $\mu$ s)
1	0	$\phi/2^5$ (1.33 $\mu$ s)
1	1	Event count mode



**[bit 9 to bit 7] MOD2, MOD1, MOD0**

This bit is used to set the operation mode and the I/O pin functions. The input pin functions as a trigger at MOD2=0. When the active edge is input to the input pin and the count operation proceeds, the content of the reload register is loaded to the counter. When the MOD2 value is "1", the timer operates in the gate counter mode and the input pin functions as the gate input. In this mode, the counter operates only while the active level is being input to the input pin.

The internal clock mode and the event counter mode are selected from the modes shown in Table 16.2-1 and Table 16.2-2 by combining the bits from MOD2 to MOD0.

**Table 16.2-1 Internal Clock Mode (CSL1, CSL0 = 00, 01 or 10)**

MOD2	MOD1	MOD0	Input Pin function	Active edge or level
0	0	0	Trigger invalidity	- [Initial value]
0	0	1	Trigger input	Rising edge
0	1	0		Falling edge
0	1	1		Both edges
1	X	0	Gate input	"L" level
1	X	1		"H" level

**Table 16.2-2 Event Counter Mode (CSL1, CSL0 = 11)**

MOD2	MOD1	MOD0	Input Pin function	Active edge or level
X	0	0	Trigger invalidity	- [Initial value]
X	0	1	Trigger input	Rising edge
X	1	0		Falling edge
X	1	1		Both edges

**[bit 6] OUTE (power output permission)**

The power output permission is controlled.

The TOT0 to TOT2 pin functions as the general-purpose port and the timer output pin when "0" and "1" are provided, respectively. The output waveform becomes a toggle waveform in the reload mode. In the one-shot mode, the TOT0 to TOT2 pin outputs a short-shape wave which indicates the proceeding count operation.

OUTE	Function
0	General-purpose port [Initial value]
1	Timer output

**[bit 5] OUTL (setting of output level)**

This bit is used to set the TOT0 to TOT2 pin output level. OUTL and the output pin level reverses in 0/1.

OUTL	At One-shot mode (RELD=0)	At Reload mode (RELD=1)
0	Short-shape wave during counting "H"	0 [Initial value]
1	Short-shape wave during counting "L"	1
X	1	0
X	1	1

**[bit 4] RELD (reload operation permission)**

It is a bit by which the reload operation is permitted. When the RELD is "1", the timer operates in the reload mode operation. In this mode, the timer loads the reload register content to the counter and continues the count operation even when an underflow occurs (when the counter value changes from 0000<sub>H</sub> to FFFF<sub>H</sub>). The timer works in the single shot mode when RELD is "0". In this mode, the counter operation stops when the counter value changes from 0000<sub>H</sub> to FFFF<sub>H</sub> and the underflow occurs.

RELD	Function
0	One-shot mode [Initial value]
1	Reload mode

**[bit 3] INTE (timer interruption demand permission)**

This bit permits the timer interrupt request. If the INTE value is 0, no interrupt request is generated even when the UF value becomes "1".

INTE	Function
0	Interrupt request power output interdiction [Initial value]
1	Interrupt request output enabled

**[bit 2] UF (timer interruption demand flag)**

It is Interrupt request flag. When the underflow is generated, UF is set in "1". Cleared by writing "0" or by  $\mu$ DMAC. The read value is "1" when using a read-modify-write instruction.

UF	At write	At programming
0	Underflow none of counter [Initial value]	Clearness of this bit [Initial value]
1	There is an underflow of the counter.	There is no change (There is no influence on another).

**[bit 1] CNTE (timer counter permission)**

It is a bit by which the timer counter is permitted.

CNTE	Function
0	Counter stop [Initial value]
1	Counter permission (startup trigger waiting)

**[bit 0] TRG (Software trigger)**

It is Software trigger bit. When 1 is written on the TRG, the software trigger is applied so that the reload register contents of timer is loaded to the counter to start the count operation. Writing "0" has no meaning. Reading is always "0". Always enabled only when the CNTE value is 1 regardless of the operation mode.

TRG	Function
0	There is no change (There is no influence on another).
1	Count operation start

**MB90330A Series****16.2.2 16-bit Timer Register 0 to 2 (TMR0 to TMR2)/  
16-bit Reload Register 0 to 2 (TMRLR0 to TMRLR2)**

Configuration and functions of 16-bit timer registers 0 to 2 (TMR0 to TMR2)/16-bit reload registers 0 to 2 (TMRLR0 to TMRLR2) are described.

**■ 16-bit Timer Register 0 to 2 (TMR0 to TMR2)/16-bit Reload Register 0 to 2 (TMRLR0 to TMRLR2)**

Figure 16.2-3 shows the bit configuration of 16-bit timer registers 0 to 2 (TMR0 to TMR2)/16-bit reload registers (TMRLR0 to TMRLR2).

**Figure 16.2-3 Bit Configuration of 16-bit Timer Registers 0 to 2 (TMR0 to TMR2)/16-bit Reload Registers 0 to 2 (TMRLR0 to TMRLR2)**

	bit	15	14	13	12	11	10	9	8	TMR0 to TMR2/TMRLR0 to TMRLR2
ch.0 : 000065 <sub>H</sub>		D15	D14	D13	D12	D11	D10	D09	D08	16-bit timer register/
ch.1 : 000069 <sub>H</sub>		(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	16-bit reload register (Upper)*
ch.2 : 00006D <sub>H</sub>		(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	Read (TMR0 to TMR2)
		(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	Write (TMRLR0 to TMRLR2)
										Initial value

	bit	7	6	5	4	3	2	1	0	TMR0 to TMR2/TMRLR0 to TMRLR2
ch.0 : 000064 <sub>H</sub>		D07	D06	D05	D04	D03	D02	D01	D00	16-bit timer register/
ch.1 : 000068 <sub>H</sub>		(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	16-bit reload register (Lower)*
ch.2 : 00006C <sub>H</sub>		(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	Read (TMR0 to TMR2)
		(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	Write (TMRLR0 to TMRLR2)
										Initial value

\* : This functions as 16-bit timer register (TMR0 to TMR2) at reading.  
This functions as 16-bit timer register (TMRLR0 to TMRLR2) at writing.

**■ 16-bit Timer Register 0 to 2 (TMR0 to TMR2)**

This register can read the counter value of 16-bit down counter. When the counter operation is permitted (CNTE = 1 for TMCSR0 to TMCSR2) and the count operation is started, the value written in the 16-bit reload register is loaded to the registers TMR0 to TMR2 and the count down is started. In the count stop status (CNTE = 0 for TMCSR0 to TMCSR2), the TMR0 to TMR2 register value is held.

**Note:**

Always use the word transfer instruction (MOVW A 003AH, etc.) when reading the TMR0 to TMR2 register that is enabled during the counter operation.

The 16-bit timer register (TMR0 to TMR2) has a read-only function despite its location at the same address as the write-only 16-bit reload register (TMRLR0 to TMRLR2). Therefore, the write process does not affect the TMR0 to TMR2 value but the TMRLR0 to TMRLR2 is written.

### ■ 16-bit Reload Register 0 to 2 (TMRLR0 to TMRLR2)

Set the initial counter value to the registers TMRLR0 to TMRLR2 in the status the counter operation is inhibited (CNTE = 0 for TMCSR0 to TMCSR2) regardless of the 16-bit reload timer operation. When the counter operation is permitted (CNTE = 1 for TMCSR0 to TMCSR2) and the counter is started, the count down is started from the value written in the registers TMRLR0 to TMCSR2. Values set in the registers TMRLR0 to TMCSR2 are reloaded to the counter to continue the count down at the underflow occurrence in the reload mode. In the one-shot mode, the counter stops at  $FFFF_H$  when an underflow occurs.

---

**Note:**

Write the TMRLR0 to TMRLR2 register in the status the counter stops (CNTE = 0 for TMCSR0 to TMCSR2).

In addition, always use the word transfer instruction (MOVW A 003AH, etc.) when writing.

The 16-bit timer register (TMRLR0 to TMRLR2) has a write-only function despite its location at the same address as the read-only 16-bit reload register (TMR0 to TMR2). Therefore, since the read value is the TMR0 to TMR2 value, instructions such as INC, DEC, etc. cannot be used to perform the read-modify-write (RMW) operations.

---

## MB90330A Series

### 16.3 Movement of 16-bit Reload Timer

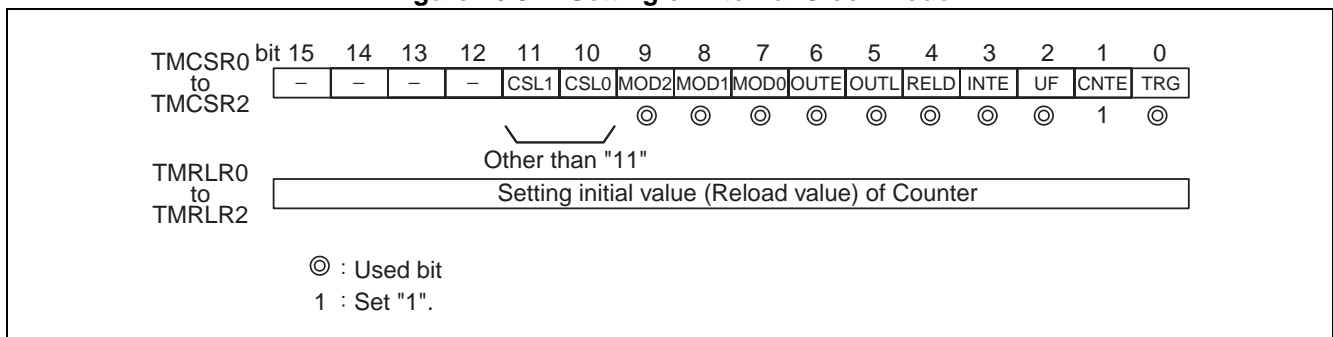
The 16-bit reload timer setting and the counter operation status transition are described.

#### ■ Setting of 16-bit Reload Timer

##### ● Setting of internal clock mode

To operate it as an interval timer, the setting shown in Figure 16.3-1 is necessary.

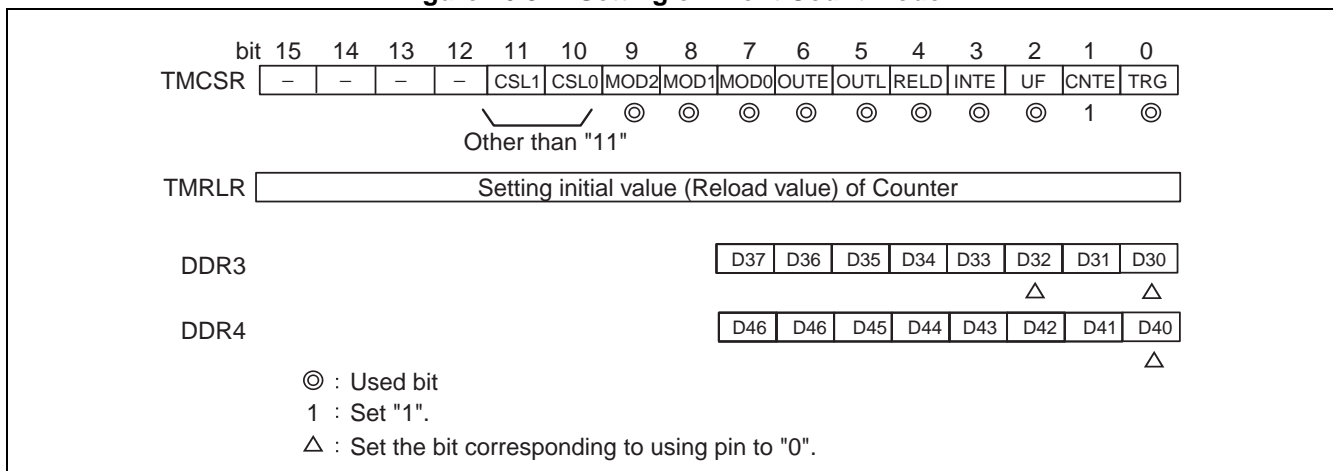
Figure 16.3-1 Setting of Internal Clock Mode



##### ● Setting of Event Count Mode

To operate it in the event counter mode, the setting shown in Figure 16.3-2 is necessary.

Figure 16.3-2 Setting of Event Count Mode

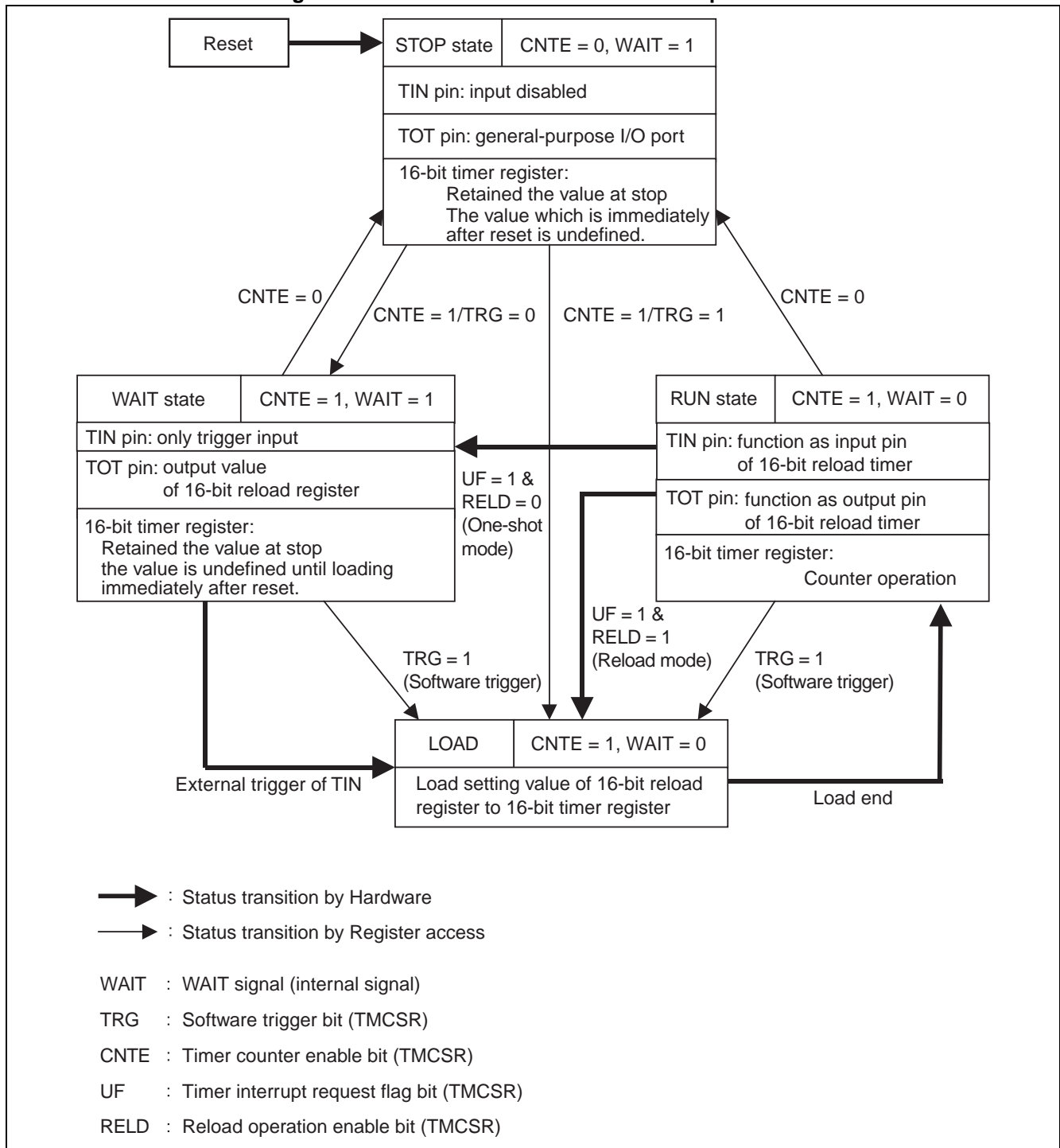


## 16.3.1 State Transition of Counter Operation

The state transition of the counter operation is shown.

### ■ State Transition of Counter Operation

Figure 16.3-3 State Transition of Counter Operation



### 16.3.2 Operation of Internal Clock Mode (Reload Mode)

It is synchronized with the internal count clock, the 16-bit counter performs the count down, and the counter underflow generates the CPU interrupt request. Also can output toggle waveforms from the timer output pin.

#### ■ Operation of Internal Clock Mode (Reload Mode)

When the count operation is permitted (CNTE = 1 for TMCSR) and the timer is started by the software trigger bit (TRG of TMCSR) or an external trigger, the reload register (TMRLR) value is loaded to the counter and the counter operation is started.

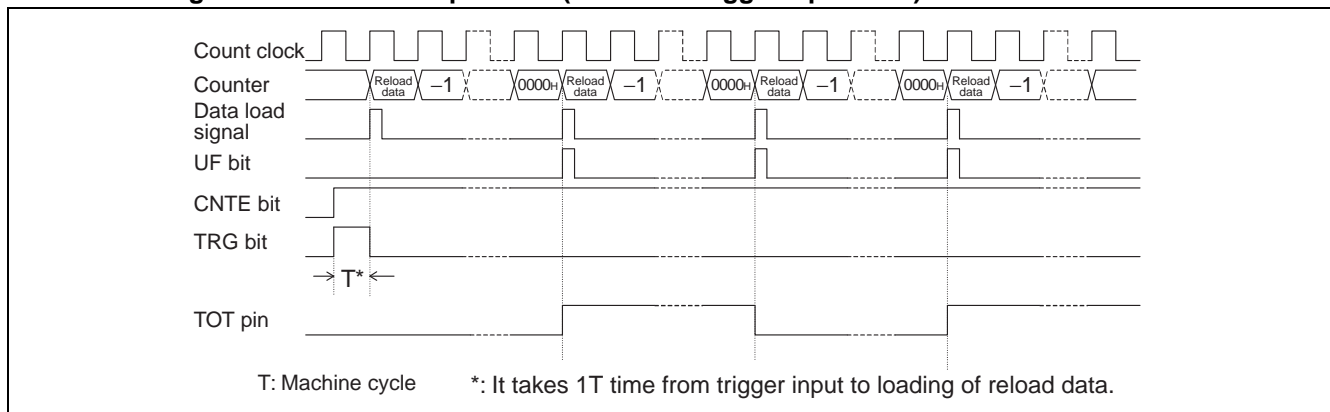
When the counter permission bit and the software trigger bit are simultaneously set to "1", the count operation starts at the same time of the counter permission. When the counter value underflows ("0000<sub>H</sub>" → "FFFF<sub>H</sub>"), the 16-bit reload register (TMRLR) value is loaded to the counter and the count operation is continued. At this moment, if the underflow interrupt request flag bit (UF) is set to "1" and the interrupt request permission bit (INTE) is set to "1", the interrupt request occurs. In addition, the toggle waveform which reverses at every underflow can be output from the TOT pin.

#### ● Operation of Software trigger

When "1" is written to TRG bit of timer control status register (TMCSR), the counter is started.

Figure 16.3-4 shows the software trigger operation at reloading.

**Figure 16.3-4 Count Operation (Software Trigger Operation) in the Reload Mode**



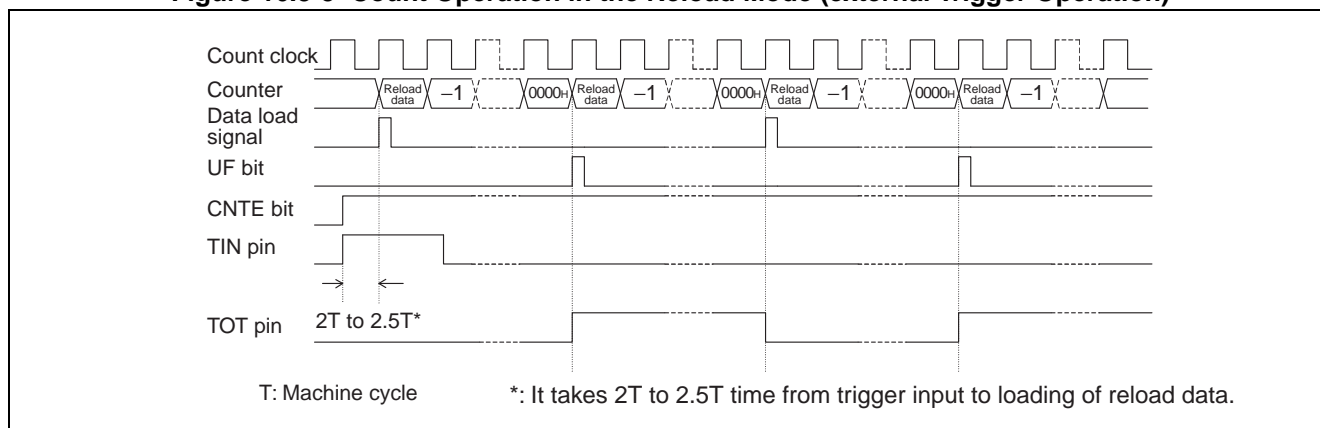
#### ● Operation of External trigger

When a valid edge (rising edge, falling edge, or both edges can be selected) is input to the TIN pin, the counter is started.

Figure 16.3-5 shows the external trigger operation in the reload mode.



**Figure 16.3-5 Count Operation in the Reload Mode (external Trigger Operation)**



Note:

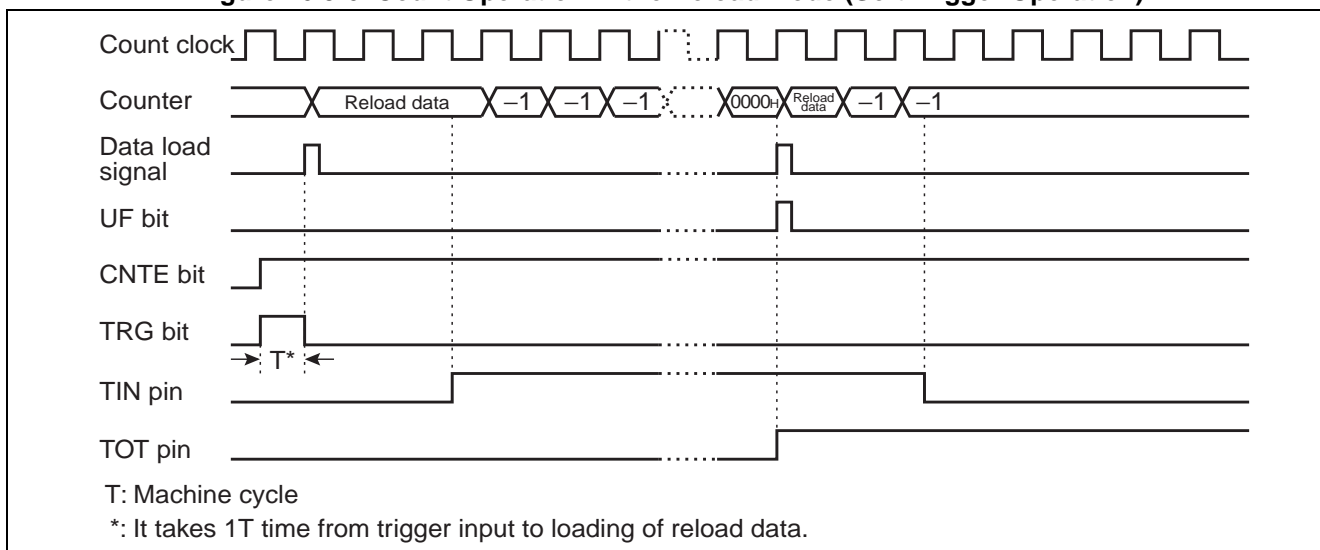
Input a trigger pulse of which width is  $2/\phi$  or more to the TIN pin.

#### ● Operation of gate input

When "1" is set to the count permission bit (CNTE) of timer control status register (TMCSR) and "1" is set to the software trigger bit (TRG), the count operation is started.

While a valid level ("L" or "H" can be set) of gate input, which is, set in the operation mode-setting bit (MOD2, MOD1, and MOD0) is being input to the TIN pin, the count operation is performed.

**Figure 16.3-6 Count Operation in the Reload Mode (Soft Trigger Operation)**



Note:

Input a trigger pulse of which width is  $2/\phi$  or more to the TIN pin.

### 16.3.3 Operation of Internal Clock Mode (Single Shot Mode)

It is synchronized with the internal count clock, the 16-bit counter performs the count down, and the counter underflow generates the CPU interrupt request. Also the TOT pin can output rectangular waveforms indicating that counting is going on.

#### ■ Internal Clock Mode (Single Shot Mode)

When the count operation is permitted (CNTE = 1 for TMCSR) and the timer is started by the software trigger bit (TRG of TMCSR) or an external trigger, the count operation is started. When the count permission bit and the software trigger bit are simultaneously set to "1", the count is started at the same time of the count permission. When the counter value underflows ("0000<sub>H</sub>" → "FFFF<sub>H</sub>"), the counter stops in the state of "FFFF<sub>H</sub>". At this moment, if the underflow interrupt request flag bit (UF) is set to "1" and the interrupt request permission bit (INTE) is set to "1", the interrupt request occurs.

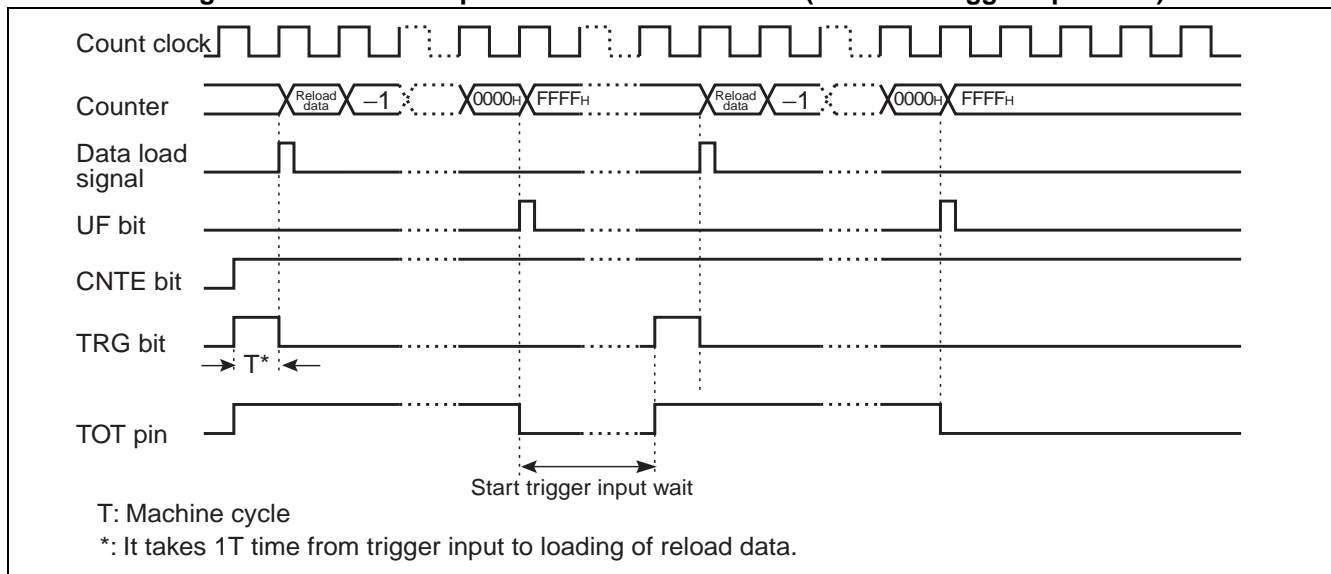
In addition, the rectangular waveform indicating the count operation can be output from the TOT pin.

#### ● Operation of Software trigger

When "1" is written to TRG bit of timer control status register (TMCSR), the counter is started.

Figure 16.3-7 shows the software trigger operation in the one-shot mode.

**Figure 16.3-7 Count Operation in One-shot Mode (Software Trigger Operation)**

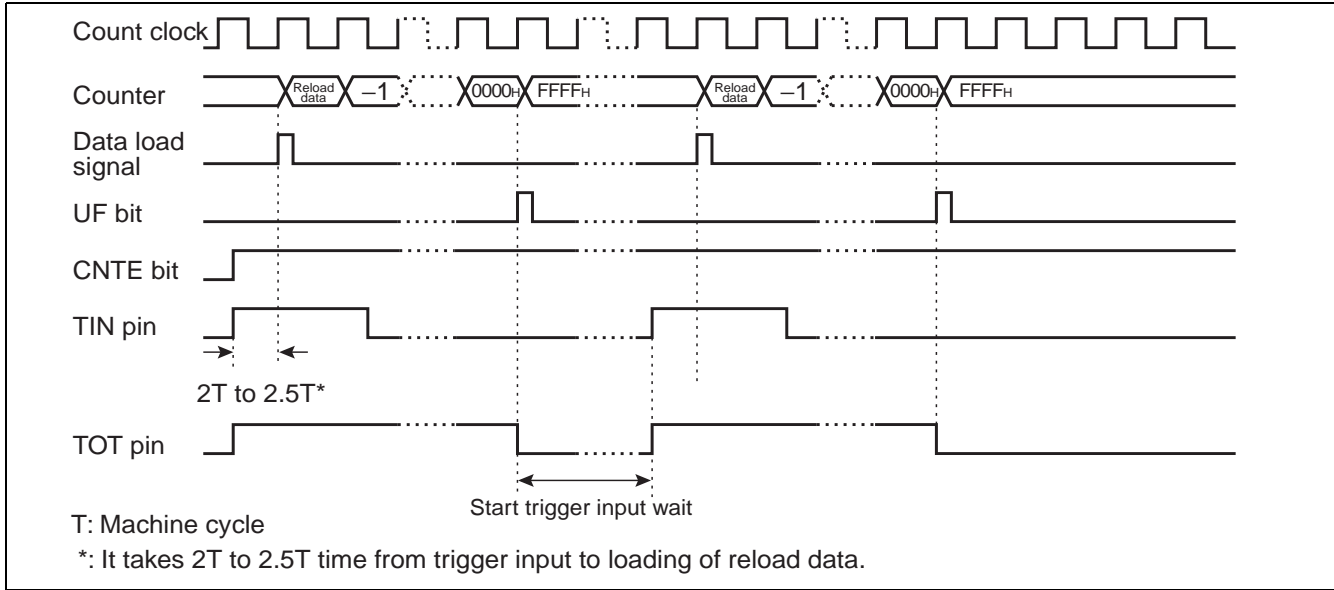


#### ● Operation of External trigger

When a valid edge (rising edge, falling edge, or both edges can be selected) is input to the TIN pin, the counter is started.

Figure 16.3-8 shows the external trigger operation in the one-shot mode.

Figure 16.3-8 Count Operation in One-shot Mode (external Trigger Operation)

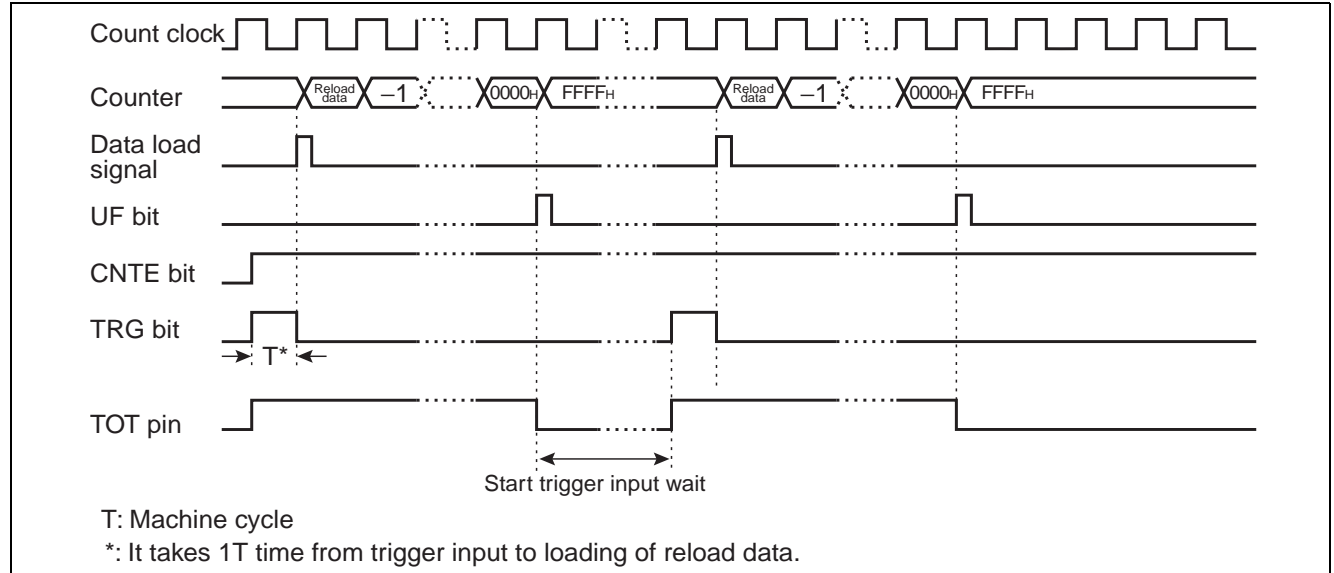


Note:  
Input a trigger pulse of which width is  $2/\phi$  or more to the TIN pin.

● Operation of gate input

While the valid level ("H" level or "L" level can be selected) is being input to the TIN pin, the count operation is performed. Figure 16.3-9 shows the gate input operation in the one-shot mode.

Figure 16.3-9 Count Operation in One-shot Mode (Soft Trigger Gate Input Operation)



Note:  
Input a gate input of which pulse width is  $2/\phi$  or more to the TIN pin.

### 16.3.4 Event Count Mode

When the input edge from the TIN pin is counted, the 16-bit counter is counted down and the counter underflow occurs, the CPU interrupt request is generated. In addition, the toggle waveform or the rectangular waveform can be output from the TOT pin.

#### ■ Event Count Mode

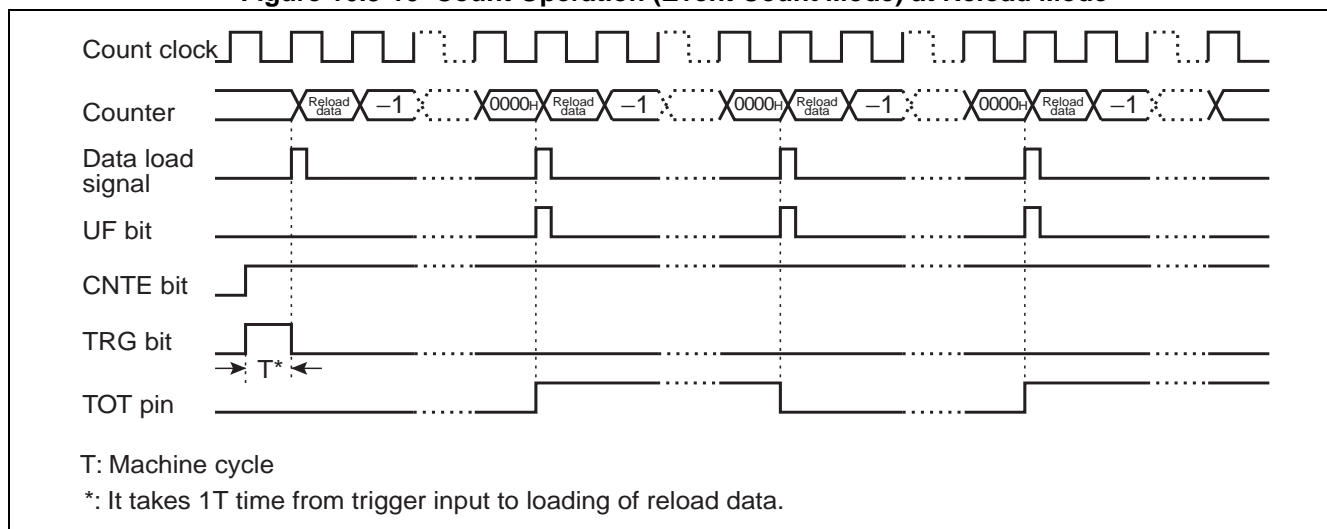
When the count operation is permitted (CNTE = 1 for TMCSR) and the counter is started (TRG = 1 for TMCSR), the 16-bit reload register (TMRLR) value is loaded to the counter. Every time when the valid edge (rising edge, falling edge, or both edge can be selected) of pulses (external count clock) input to the TIN pin is detected, the count down is performed. When the count permission bit and the software trigger bit are simultaneously set to "1", the count is started at the same time of the count permission.

#### ● Operation of Reload mode

When the counter value underflows ("0000<sub>H</sub>" → "FFFF<sub>H</sub>"), the 16-bit reload register (TMRLR) value is loaded to the counter and the count operation is continued. At this moment, if the underflow interrupt request flag bit (UF) is set to "1" and the interrupt request permission bit (INTE of TMCSR) is "1", the interrupt request is generated. In addition, the toggle waveform which reverses at every underflow can be output from the TOT pin.

Figure 16.3-10 shows Count operation (event count mode) at reload mode.

**Figure 16.3-10 Count Operation (Event Count Mode) at Reload Mode**



Note:

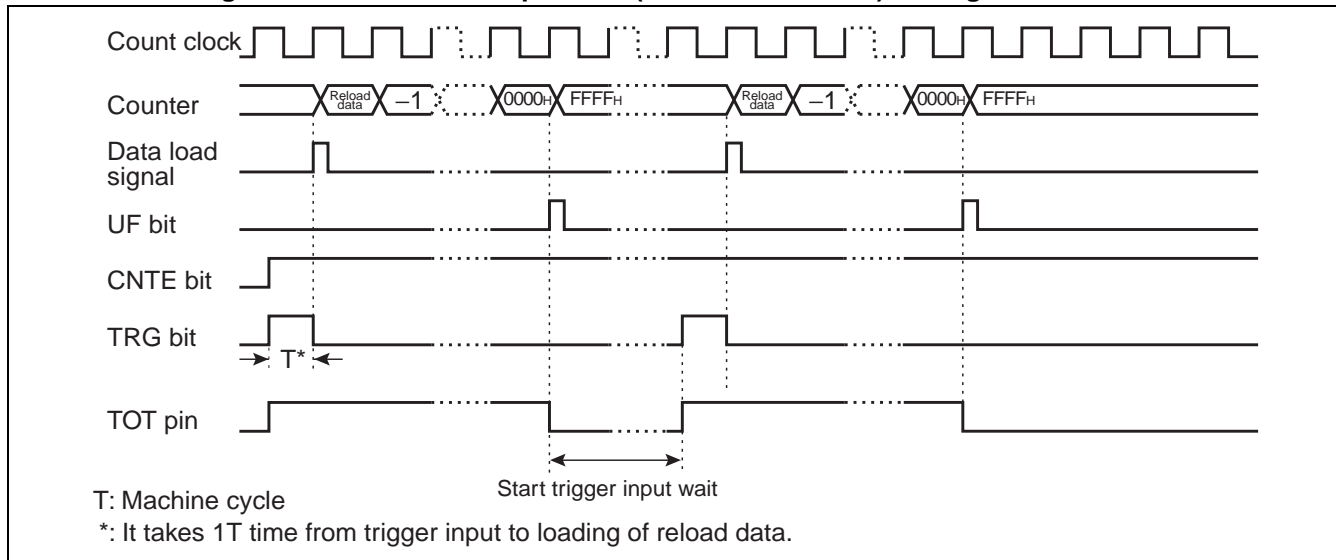
Use the 4/φ or more "H" width and "L" width of the clock to be input to the TIN pin.

● Operation of One-shot mode

When the counter value underflows ("0000<sub>H</sub>" → "FFFF<sub>H</sub>"), the counter stops in the state of "FFFF<sub>H</sub>". At this moment, if the underflow request flag bit (UF) is set to "1" and the interrupt request output permission bit (INTE) is "1", the interrupt request is generated. In addition, the rectangular waveform indicating the count operation can be output from the TOT pin.

Figure 16.3-11 shows Counter operation (event count mode) at single shot mode.

**Figure 16.3-11 Counter Operation (Event Count Mode) at Single Shot Mode**



Note:

Use the  $4/\phi$  or more "H" width and "L" width of the clock to be input to the TIN pin.

# **CHAPTER 17**

---

## **8/16-BIT PPG TIMER**

**This chapter describes an overview of 8/16-bit PPG timer, the configuration and functions of register, and the 8/16-bit PPG timer operation.**

- 17.1 Overview of 8/16-bit PPG Timer
- 17.2 Registers of 8/16-bit PPG Timer
- 17.3 Operation of 8/16-bit PPG Timer

## 17.1 Overview of 8/16-bit PPG Timer

---

The 8/16-bit PPG timer provides the PPG output via the pulse output according to the timer operation with the 8-bit reload timer module.

It has the following as a hardware.

- Six 8-bit down counters
- Twelve 8-bit reload timers
- Three 16-bit control registers
- Six external pulse output pins
- Six interrupt outputs.

Further, the MB90330A series provides 6 channels as the 8-bit PPG that also operate as the 16-bit PPG (3 channels) in the combination of PPG0 + PPG1/PPG2 + PPG3/PPG4 + PPG5.

---

### ■ Overview of 8/16-bit Timer PPG Timer

Overview of 8/16-bit PPG timer function is shown below.

- 8-PPG power output six channel independent operation mode

Independent 6-channel PPG output operation is enabled.

- 16-bit PPG output operation mode

The 3-channel and 16-bit PPG output operation is enabled.

Combination of PPG0 + PPG1, PPG2 + PPG3, and PPG4 + PPG5 are used.

- 8 + 8-bit PPG output operation mode

When the PPG0 (PPG2/PPG4) output is clock-input of PPG1 (PPG3/PPG5), the 8-bit PPG output in an arbiter cycle is enabled.

- PPG output operation

Pulse waves in an arbiter cycle and of duty ratio are output.

---

Reference:

Only the PPG1 can be used for the interrupt trigger of A/D converter.

---

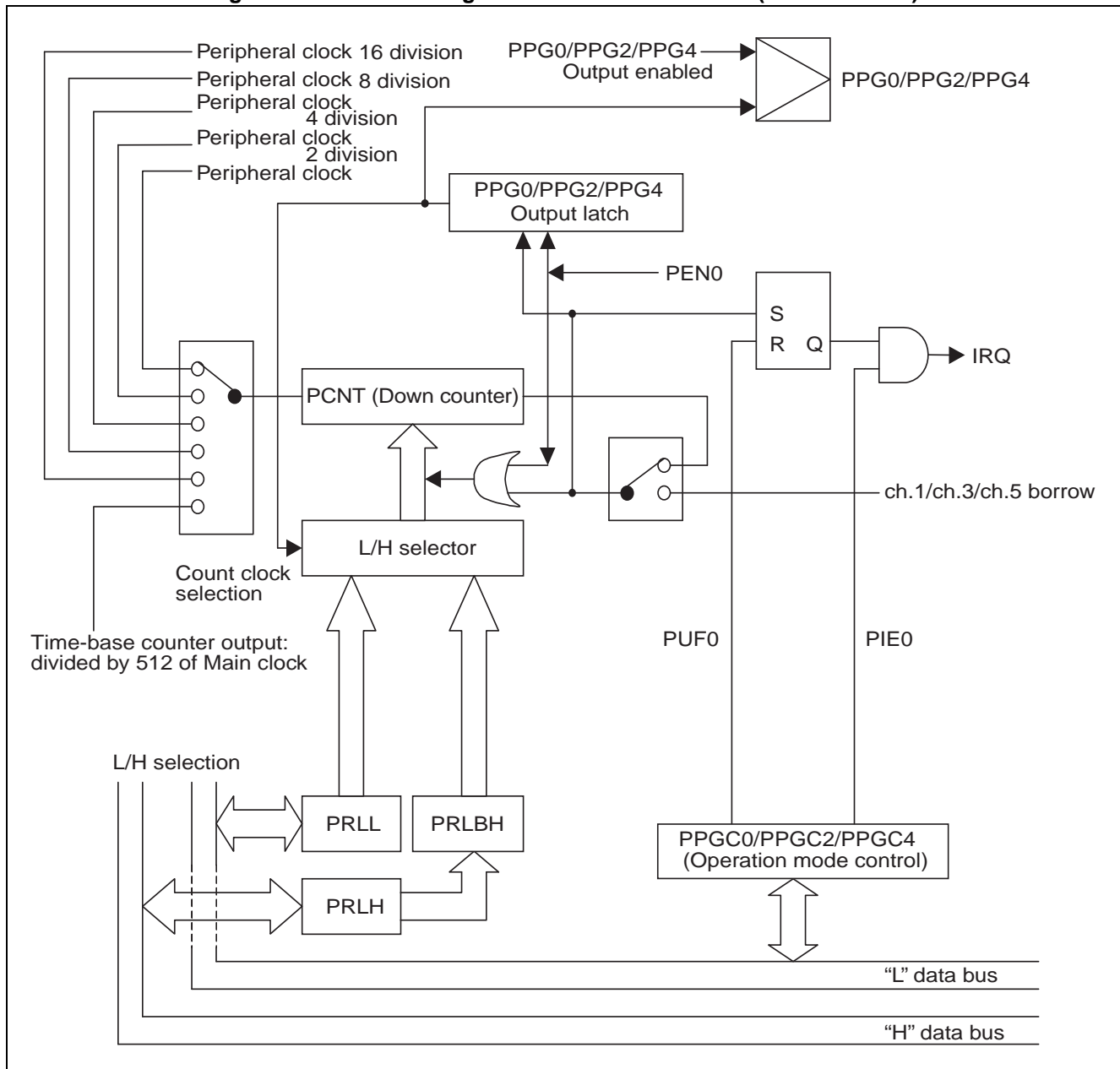
## 17.1.1 Block Diagram of 8/16-bit PPG Timer

Block diagram of ch.0/ch.2/ch.4 and ch.1/ch.3/ch.5 of 8/16-bit PPG timer is shown.

## ■ Block Diagram of 8/16-bit PPG Timer

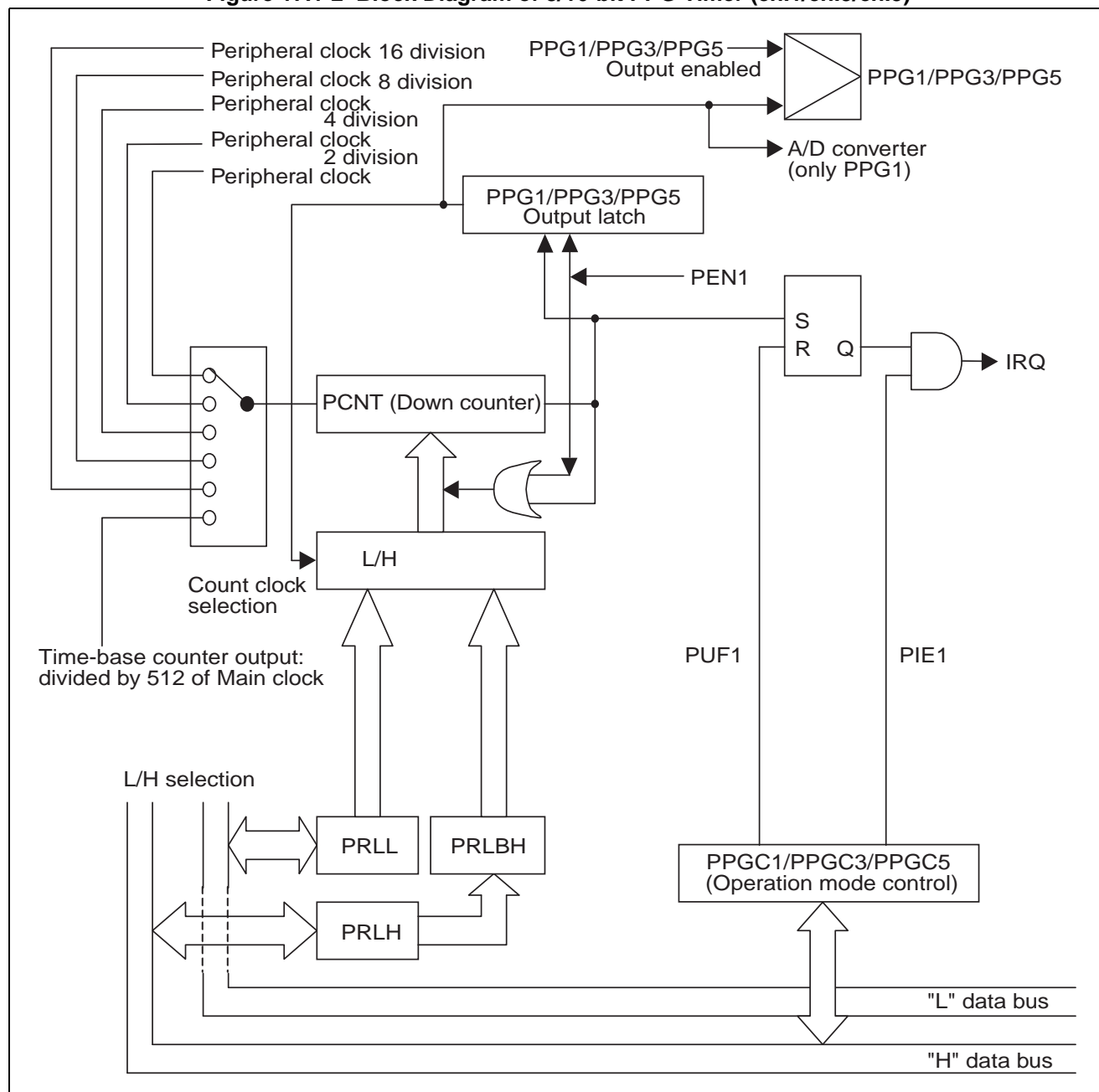
Figure 17.1-1 shows the block diagram of ch.0/ch.2/ch.4. Figure 17.1-2 shows the block diagram of channels 1/3/5.

Figure 17.1-1 Block Diagram of 8/16-bit PPG Timer (ch.0/ch.2/ch.4)





**Figure 17.1-2 Block Diagram of 8/16-bit PPG Timer (ch.1/ch.3/ch.5)**



## 17.2 Registers of 8/16-bit PPG Timer

Configuration and functions of register used for the 8/16-bit PPG timer are described.

### ■ Register List of 8/16-bit PPG Timer

Figure 17.2-1 shows the register list of 8/16-bit PPG timer.

Figure 17.2-1 Register List of 8/16-bit PPG Timer

ch.0 : 000046H ch.2 : 000048H ch.4 : 00004AH	bit 7 6 5 4 3 2 1 0	PPGC0/PPGC2/PPGC4
	PEN0 – PE00 PIE0 PUF0 – – Reserved	PPG Operation mode control register
	(R/W) (–) (R/W) (R/W) (R/W) (–) (–) (R/W)	Read/Write
	(0) (X) (0) (0) (0) (X) (X) (1)	Initial value
ch.1 : 000047H ch.3 : 000049H ch.5 : 00004BH	bit 15 14 13 12 11 10 9 8	PPGC1/PPGC3/PPGC5
	PEN1 – PE10 PIE1 PUF1 MD1 MD0 Reserved	PPG Operation mode control register
	(R/W) (–) (R/W) (R/W) (R/W) (R/W) (R/W) (R/W)	Read/Write
	(0) (X) (0) (0) (0) (0) (0) (1)	Initial value
ch.0, 1 : 00004CH ch.2, 3 : 00004EH ch.4, 5 : 000050H	bit 7 6 5 4 3 2 1 0	PPG01/PPG23/PPG45
	PCS2 PCS1 PCS0 PCM2 PCM1 PCM0 Reserved Reserved	PPG Output control register
	(R/W) (R/W) (R/W) (R/W) (R/W) (R/W) (R/W) (R/W)	Read/Write
	(0) (0) (0) (0) (0) (0) (X) (X)	Initial value
ch.0 : 007900H ch.1 : 007902H ch.2 : 007904H ch.3 : 007906H ch.4 : 007908H ch.5 : 00790AH	bit 7 6 5 4 3 2 1 0	PRL0 to PRL5
	D07 D06 D05 D04 D03 D02 D01 D00	PPG Reload register lower
	(R/W) (R/W) (R/W) (R/W) (R/W) (R/W) (R/W) (R/W)	Read/Write
	(X) (X) (X) (X) (X) (X) (X) (X)	Initial value
ch.0 : 007901H ch.1 : 007903H ch.2 : 007905H ch.3 : 007907H ch.4 : 007909H ch.5 : 00790BH	bit 15 14 13 12 11 10 9 8	PRLH0 to PRLH5
	D15 D14 D13 D12 D11 D10 D09 D08	PPG Reload register upper
	(R/W) (R/W) (R/W) (R/W) (R/W) (R/W) (R/W) (R/W)	Read/Write
	(X) (X) (X) (X) (X) (X) (X) (X)	Initial value

## 17.2.1 PPG0/PPG2/PPG4 Operation Mode Control Register (PPGC0/PPGC2/PPGC4)

Configuration and functions of PPG0/PPG2/PPG4 operation mode control register (PPGC0/PPGC2/PPGC4) are described.

### ■ PPG0/2/4 Operation Mode Control Register (PPGC0/PPGC2/PPGC4)

The PPG0/PPG2/PPG4 operation mode control register (PPGC0/PPGC2/PPGC4) selects the operation mode of ch.0/ch.2/ch.4, controls the pin output, selects the count clock, and controls the trigger.

Figure 17.2-2 shows the bit configuration of PPG0/PPG2/v4 operation mode control registers (PPGC0/PPGC2/PPGC4).

**Figure 17.2-2 Bit Configuration of PPG0/PPG2/PPG4 Operation Mode Control Registers (PPGC0/PPGC2/PPGC4)**

	bit	7	6	5	4	3	2	1	0	PPGC0/PPGC2/PPGC4
ch.0 : 000046H		PEN0	—	PE00	PIE0	PUF0	—	—	Reserved	PPG Operation mode control register
ch.2 : 000048H		(R/W)	(—)	(R/W)	(R/W)	(R/W)	(—)	(—)	(R/W)	Read/Write
ch.4 : 00004AH		(0)	(X)	(0)	(0)	(0)	(X)	(X)	(1)	Initial value

The following describes functions of each bit of PPG0/PPG2/PPG4 operation mode control registers (PPGC0/PPGC2/PPGC4):

#### [bit 7] PEN0: ppg Enable (operation permission)

The operation start and operation mode of PPG0/PPG2/PPG4 are selected.

PEN0	Operating State
0	Operation stop ("L" level output maintenance)
1	PPG Enabling Operations

- With this bit is set to "1", the PPG starts counting.
- This bit is initialized to "0" at reset.
- Reading and writing are allowed.

#### [bit 6] Undefined bit

The read value is irregular. Nothing is affected when it is written.

#### [bit 5] PE00: ppg output Enable 00 (PPG0/PPG2/PPG4 output pin enabled)

Inhibition and permission of pulse output to the external pulse output pin PPG0/PPG2/PPG4 are controlled.

PE00	Operating State
0	General-purpose port pin (pulse output interdiction).
1	PPG0/PPG2/PPG4 pulse output (pulse output permission).

- This bit is initialized to "0" at reset.
- Reading and writing are allowed.

**[bit 4] PIE0: ppg Interrupt Enable (interrupt to PPG0/PPG2/PPG4 enabled)**

PPG0/PPG2/PPG4 interrupt inhibition and permission are controlled.

PIE0	Operating State
0	Disables the interrupt
1	Interruption permission

- If PUF0 is changed to "1" while this bit is "1", an interrupt request is generated. If this bit is "0", no interrupts are generated.
- This bit is initialized to "0" at reset.
- Reading and writing are allowed.

**[bit 3] PUF0: ppg Underflow Flag (PPG0/PPG2/PPG4 counter underflow)**

Detected result of counter underflow of the PPG0/PPG2/PPG4 is shown.

PUF0	Operating State
0	The PPG counter underflow is not detected.
1	The PPG counter underflow was detected.

In the 8-bit PPG6 channel mode (PPG0, PPG1/PPG2, PPG3/PPG4, PPG5) and the 8-bit prescaler + 8-bit PPG mode, the counter values of ch.0, ch.2, ch.4 are set to "1" when they underflow from 00<sub>H</sub> to FF<sub>H</sub>. In the 16-bit PPG3 channel mode (PPG0, PPG1/PPG2, PPG3/PPG4, PPG5), the counter values of ch.1, ch.3, ch.5/ch.0, ch.2, ch.4 are set to "1" when they underflow from 0000<sub>H</sub> to FFFF<sub>H</sub>. Becomes "0" by written "0". "1" writing in the PUF0 bit is not significant. "1" is read with a read-modify-write instruction.

- This bit is initialized to "0" at reset.
- Reading and writing are allowed.

**[bit 2, bit 1] Undefined bit**

The read value is irregular. Nothing is affected when it is written.

**[bit 0] Reserved bit**

It is Reserved bit. Always set this bit to "1".

## 17.2.2 PPG1/PPG3/PPG5 Operation Mode Control Register (PPGC1/PPGC3/PPGC5)

Configuration and functions of PPG1/PPG3/PPG5 operation mode control register (PPGC1/PPGC3/PPGC5) are described.

### ■ PPG1/PPG3/PPG5 Operation Mode Control Register (PPGC1/PPGC3/PPGC5)

The PPG1/PPG3/PPG5 operation mode control register (PPGC1/PPGC3/PPGC5) selects the operation mode of ch.1/ch.3/ch.5, controls the pin output, selects the count clock, and controls the trigger.

Figure 17.2-3 shows the bit configuration of PPG1/PPG3/PPG5 operation mode control registers (PPGC1/PPGC3/PPGC5).

**Figure 17.2-3 Bit Configuration of PPG1/PPG3/PPG5 Operation Mode Control Registers (PPGC1/PPGC3/PPGC5)**

ch.1 : 000047H	bit 15	14	13	12	11	10	9	8	PPGC1/PPGC3/PPGC5
ch.3 : 000049H	PEN1	–	PE10	PIE1	PUF1	MD1	MD0	Reserved	PPG Operation mode control register
ch.5 : 00004BH	(R/W)	(–)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	Read/Write
	(0)	(X)	(0)	(0)	(0)	(0)	(0)	(1)	Initial value

The following describes functions of each bit of PPG1/PPG3/PPG5 operation mode control registers (PPGC1/PPGC3/PPGC5):

#### [bit 15] PEN1: ppg Enable (operation permission)

The operation start and operation mode of PPG1/PPG3/PPG5 are selected.

PEN1	Operating State
0	Operation stop ("L" level power output maintenance)
1	PPG Enabling Operations

- When this bit is set to "1", PPG count starts.
- This bit is initialized to "0" at reset.
- Reading and writing are allowed.

#### [bit 14] Undefined bit

The lead value is irregular. Nothing is affected when it is written.

## [bit 13] PE10: ppg output Enable10 (PPG1/PPG3/PPG5 output pin enabled)

Inhibition and permission of pulse output to the external pulse output pin PPG1/PPG3/PPG5 are controlled.

PE10	Operating State
0	General-purpose port pin (pulse output interdiction)
1	PPG1/PPG3/PPG5 pulse output (pulse output permission)

- This bit is initialized to "0" at reset.
- Reading and writing are allowed.

## [bit 12] PIE1: ppg Interrupt Enable (interrupt to PPG1/PPG3/PPG5 enabled)

PPG1/PPG3/PPG5 interrupt inhibition and permission are controlled.

PIE1	Operating State
0	Disables the interrupt.
1	Interruption permission.

- If PUF1 is set to "1" when this bit is "1", an interrupt request is generated. When this bit is "0", no interrupts are generated.
- This bit is initialized to "0" at reset.
- Reading and writing are allowed.

## [bit 11] PUF1:Ppg Underflow Flag (PPG1/PPG3/PPG5 counter underflow)

Detected result of counter underflow of the PPG1/PPG3/PPG5 is shown.

PUF1	Operating State
0	The PPG counter underflow has not been detected.
1	The PPG counter underflow was detected.

In the 8-bit PPG6 channel mode (PPG0, PPG1/PPG2, PPG3/PPG4, PPG5) and the 8-bit prescaler + 8-bit PPG mode, the counter values of ch.1, ch.3, ch.5 are set to "1" when they underflow from 00<sub>H</sub> to FF<sub>H</sub>. In the 16-bit PPG3 channel mode (PPG0, PPG1/PPG2, PPG3/PPG4, PPG5), the counter values of ch.1, ch.3, ch.5/ch.0, ch.2, ch.4 are set to "1" when they underflow from 0000<sub>H</sub> to FFFF<sub>H</sub>. Becomes "0" by written "0". "1" writing in the PUF0 bit is not significant. "1" is read with a read-modify-write instruction.

- This bit is initialized to "0" at reset.
- Reading and writing are allowed.

**[bit 10, bit 9] MD1, MD0: ppg count Mode (operation mode selection)**

The operation mode of the PPG timer is selected.

MD1	MD0	Operating mode
0	0	Byte PPG2 channel independent mode (The case multiplied by 3 is enabled).
0	1	8-bit prescaler + 8-bit PPG 1channel.
1	0	Reserved (Setting prohibited).
1	1	16-bit PPG1 channel mode (The case multiplied by 3 is enabled).

- This bit is initialized to "0" at reset.
- Reading and writing are allowed.

---

Notes:

- Please set neither MD1 nor MD0 bits in "10<sub>B</sub>".
  - Do not set the PPGC0 PEN0 bit and the PPGC1 PEN1 bit to "01<sub>B</sub>" when setting the MD1 and MD0 bits to "01<sub>B</sub>". In addition, the PEN0 and PEN1 bits are recommended to simultaneously set to "11<sub>B</sub>" or "00<sub>B</sub>".
  - Set the PEN0/PEN1 to "11<sub>B</sub>" or "00<sub>B</sub>" simultaneously, when setting the MD1 and MD0 bits to "11<sub>B</sub>" by updating the PPGC0/PPGC1 via the word transfer.
- 

**[bit 8] Reserved bit**

It is Reserved bit. Always set this bit to "1".

### 17.2.3 PPG0 to PPG5 Output Control Register (PPG01/PPG23/PPG45)

Configuration and functions of PPG0 to PPG5 output control register (PPG01/PPG23/PPG45) are described.

#### ■ PPG0 to PPG5 Output Control Register (PPG01/PPG23/PPG45)

Figure 17.2-4 shows the bit configuration of the PPG0 to PPG5 output control registers (PPG01/PPG23/PPG45).

**Figure 17.2-4 PPG0 to PPG5 Output Control Register (PPG01/PPG23/PPG45)**

	bit	7	6	5	4	3	2	1	0	
ch.0, 1 : 00004C <sub>H</sub>		PCS2	PCS1	PCS0	PCM2	PCM1	PCM0	Reserved	Reserved	PPG01/PPG23/PPG45
ch.2, 3 : 00004E <sub>H</sub>		(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	PPG Output control register
ch.4, 5 : 000050 <sub>H</sub>		(0)	(0)	(0)	(0)	(0)	(0)	(X)	(X)	Read/Write
										Initial value

PPG0 to PPG5 output control register (PPG01/PPG23/PPG45)

#### [bit 7 to bit 5] PCS2 to PCS0: ppg Count Select (count clock selection)

Selects the operation clock of the ch.1, ch.3, and ch.5 down counters.

PCS2	PCS1	PCS0	Operating mode
0	0	0	Peripheral Clock (41.7 ns machine clock 24 MHz time) in surrounding
0	0	1	Peripheral Clock /2 (83.3 ns machine clock 24 MHz time) in surrounding
0	1	0	Peripheral Clock /4 (167 ns machine clock 24 MHz time) in surrounding
0	1	1	Peripheral Clock /8 (333 ns machine clock 24 MHz time) in surrounding
1	0	0	Peripheral Clock /16 (667 ns machine clock 24 MHz time) in surrounding
1	1	1	Input clock from time-base counter ( $2^9 \times 167 \text{ ns} = 85 \mu\text{s}$ field oscillation 6 MHz time)

- These bits are initialized to "000<sub>B</sub>" at reset.
- Reading and writing are allowed.

#### Note:

Since the PPG of ch.1, ch.3, and ch.5 operates by receiving the count clock from the ch.0, ch.2, and ch.4 in the 8-bit prescaler + 8-bit PPG mode and the 16-bit PPG mode, the specified PCS2 to PCS0 bits become invalid.



**[bit 4 to bit 2] PCM2 to PCM0:ppg Count Mode (count clock selection)**

These bits select the down counter operation clock of ch.0, ch.2, and ch.4.

PCM2	PCM1	PCM0	Operating mode
0	0	0	Peripheral Clock (41.7 ns machine clock 24 MHz time) in surrounding
0	0	1	Peripheral Clock /2 (83.3 ns machine clock 24 MHz time) in surrounding
0	1	0	Peripheral Clock /4 (167 ns machine clock 24 MHz time) in surrounding
0	1	1	Peripheral Clock /8 (333 ns machine clock 24 MHz time) in surrounding
1	0	0	Peripheral Clock /16 (667 ns machine clock 24 MHz time) in surrounding
1	1	1	Input clock from time-base counter ( $2^9 \times 167 \text{ ns} = 85 \text{ ms}$ field oscillation 6 MHz time)

- These bits are initialized to "000<sub>B</sub>" at reset.
- Reading and writing are allowed.

**[bit 1, bit 0] Reserved bit**

It is Reserved bit. Always set this bit to "00<sub>B</sub>".

## 17.2.4 PPG Reload Registers (PRL0 to PRL5, PRLH0 to PRLH5)

Configuration and functions of PPG reload registers (PRL0 to PRL5, PRLH0 to PRLH5) are described.

### ■ PPG Reload Registers (PRL0 to PRL5, PRLH0 to PRLH5)

Figure 17.2-5 shows the bit configuration of PPG reload registers (PRL0 to PRL5, PRLH0 to PRLH5).

**Figure 17.2-5 PPG Reload Registers ((PRL0 to PRL5, PRLH0 to PRLH5)**

ch.0 : 007900H	bit	7	6	5	4	3	2	1	0	PRL0 to PRL5
ch.1 : 007902H										
ch.2 : 007904H										PPG Reload register lower
ch.3 : 007906H										
ch.4 : 007908H	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	Read/Write
ch.5 : 00790AH	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	Initial value
ch.0 : 007901H	bit	15	14	13	12	11	10	9	8	PRLH0 to PRLH5
ch.1 : 007903H										
ch.2 : 007905H										PPG Reload register upper
ch.3 : 007907H										
ch.4 : 007909H	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	Read/Write
ch.5 : 00790BH	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	Initial value

The PPG reload registers (PRL0 to PRL5, PRLH0 to PRLH5) are the 8-bit registers that hold the reload values for the down counter (PCNT). Each of them has roles as shown in the following table:

Register Name	Function
PRL	Hold the reload value of L side.
PRLH	Hold the reload value of H side.

- Both registers can be read and written.

#### Note:

When different values are set for the PRL and PRLH of the ch.0, ch.2, and ch.4 using the 8-bit prescaler + 8-bit PPG mode, the PPG waveforms of ch.1, ch.3, and ch.5 may be different each other according to the cycle. Therefore, the same value is recommended to set for the PRL and PRLH of the ch.0, ch.2, and ch.4.

## 17.3 Operation of 8/16-bit PPG Timer

The 8/16-bit PPG timer has the 6 channels (PPG0, PPG1/PPG2, PPG3/PPG4, PPG5) of 8-bit length PPG unit. Each of them can operate 3-type operations in total, the 8-bit prescaler + 8-bit PPG modes and the 16-bit PPG mode by performing the direct-coupled (PPG0 + PPG1/PPG2 + PPG3/PPG4 + PPG5) operations in addition to the independent mode.

### ■ Outline of Operation of 8/16-bit PPG Timer

Each of 8-bit length PPG units has two 8-bit-length reload registers for the "L" and "H" sides (PRL, PRLH).

Values written in the PRL and PRLH registers are reloaded to the 8-bit down counter (PCNT) for the "L" and "H" sides alternately and down-counted at every count clock so that the output pin value is reversed when the registers are reloaded at the count borrow occurrence. This operation causes the pin output to be the "L"-width/"H"-width pulse output corresponding to the reload register value.

Operation is started/restarted by the written register bit.

The relation between the reload operation and the pulse output is shown in the following table.

Reload operation	Pin output change
PRLH → PCNT	PPG0/PPG1 (0 → 1) rising

In addition, when the bit 4 (PIE0) in PPGC0/PPGC2/PPGC4 is "1" and the bit 2 (PIE1) in PPGC1/PPGC3/PPGC5 is "1", the interrupt request is output at a borrow occurrence from 00<sub>H</sub> to FF<sub>H</sub> (borrow from 0000<sub>H</sub> to FFFF<sub>H</sub> in the 16-bit PPG mode) for respective counter.

### ■ Operating Mode

The 8/16-bit PPG timer has 3-type operation modes in total, the 2-channel independent mode, the 8-bit prescaler + 8-bit PPG mode, and the 16-bit PPG mode (MB90330A series have 3 channels for each mode).

The 2-channel independent mode is a mode to allow an independent 2-channel operation as the 8-bit PPG. The PPG outputs of ch.0 and ch.1 are connected to the PPG0 and PPG1 pins, respectively (PPG2 to PPG5 correspond to ch.2 to ch.5).

The 8-bit prescaler + 8-bit PPG mode is the operation mode to operate the ch.0 (ch.2/ch.4) as the 8-bit prescaler and to enable the 8-bit PPG waveform output in an arbiter cycle, by counting the ch.1 (ch.3/ch.5) at a ch.0 (ch.2/ch.4) borrow output. The PPG0 (PPG2/PPG4) pin is connected to the bit prescaler output of ch.0 (ch.2/ch.4) and the PPG1 pin is connected to the PPG output of ch.1 (ch.3/ch.5).

The 16-bit PPG channel mode (MB90330A series provide 3 channels) is the operation mode to allow the direct-coupling of ch.0 and ch.1 (direct-coupled ch.2, ch.3/ch.4, ch.5) to be operated as the 16-bit PPG. Both PPG0 and PPG1 16-bit PPG power outputs are both connected.

## ■ PPG Output Operation

The 8/16-bit PPG timer is started to begin the count when both the bit 7 (PEN0) of PPGC0 register for 0 (ch.2/ch.4) PPG and the bit 15 (PEN1) of PPGC1 register for 1 (ch.3/ch.5) PPG are set to "1". After the operation started, when "0" is written to the bit 7 (PEN0) of PPGC0 or the bit 15 (PEN1) of PPGC1, the count operation is stopped and the pulse output is held at the "L" level thereafter.

Do not set the ch.1 (ch.3/ch.5) to the operating status in the 8-bit prescaler + 8-bit PPG mode when the ch.0 (ch.2/4) is in the stop state.

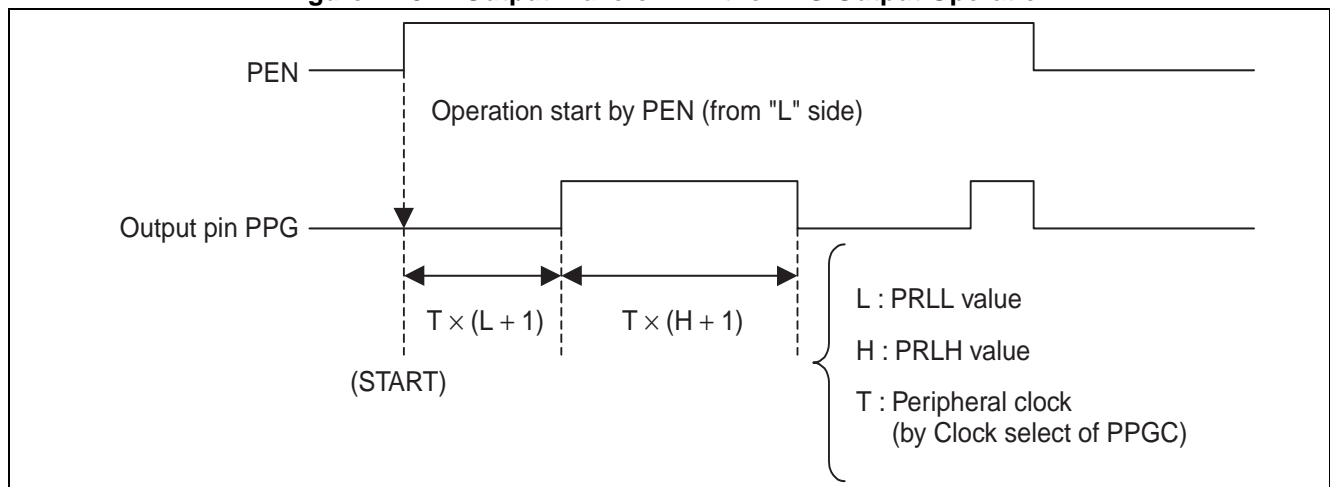
Control the bit 7 (PEN0) in the PPGC0 register and the bit 15 (PEN1) in the PPGC1 register to simultaneously start and stop in the 16-bit PPG mode.

PPG output operation is explained as follows.

When the PPG operates, the pulse wave is continuously output in an arbiter cycle and arbiter duty ratio (ratio of "H" level period and "L" level period of pulse wave). The PPG starts the pulse wave output and does not stop until the operation stop is set.

Figure 17.3-1 shows the output waveform in the PPG output operation.

**Figure 17.3-1 Output Waveform in the PPG Output Operation**



## ■ Relation between Reload Value and Pulse Width

The written value in the reload register added to "1" and multiplied by the count clock cycle produces the output pulse width. Note, therefore, that the pulse width is one count clock cycle when the reload register value is 00<sub>H</sub> during the 8-bit PPG operation or when the reload register value is 0000<sub>H</sub> during the 16-bit PPG operation. Note also that the pulse width is the 256 count clock cycles when the reload register value is FF<sub>H</sub> during the 8-bit PPG operation and the pulse width is the 65536 divisions of count clock when the reload register value is FFFF<sub>H</sub> during the 16-bit PPG operation. The equations for calculating the pulse width are shown below:

$$P_L = T \times (L + 1)$$

$$P_H = T \times (H + 1)$$

$P_L$ : Width of "L" pulse

$P_H$ : Width of "H" pulse

T: Clock Cycle

L: PRL value

H: PRLH value

### ■ Count Clock Selection

The count clock used for 8/16-bit PPG timer operation uses the peripheral clock and the time-base counter input to allow 6 types of count clock input selection.

The bit4 to bit2 (PCM2 to PCM0) of PPG01/PPG23/PPG45 register selects the ch.0 (ch.2/ch.4) clock and the bit7 to bit5 (PCS2 to PCS0) of PPG01/PPG23/PPG45 register selects the ch.1 (ch.3/ch.5) clock.

For the clock selection, 1/16 to 1 times of machine clock and the time-base counter input are selected.

#### Notes:

- The bit 14 (PCS1) value of PPGC1 register is invalid in the 8-bit prescaler + 8-bit PPG mode and 16-bit PPG mode.
- When the time-base timer input is used, the trigger or the first count cycle after stopping may deviate. In addition, if the time-base counter is initialized in the 8/16-bit PPG timer operation, the cycle may deviate.
- When the ch.0/ch.2/ch.4 is in the operating status, the ch.1/ch.3/ch.5 is in the stopped status, and the ch.1 (ch.3/ch.5) is started in the 8-bit prescaler + 8-bit PPG mode, the first count cycle may deviate.

### ■ Pin Output Control of Pulse

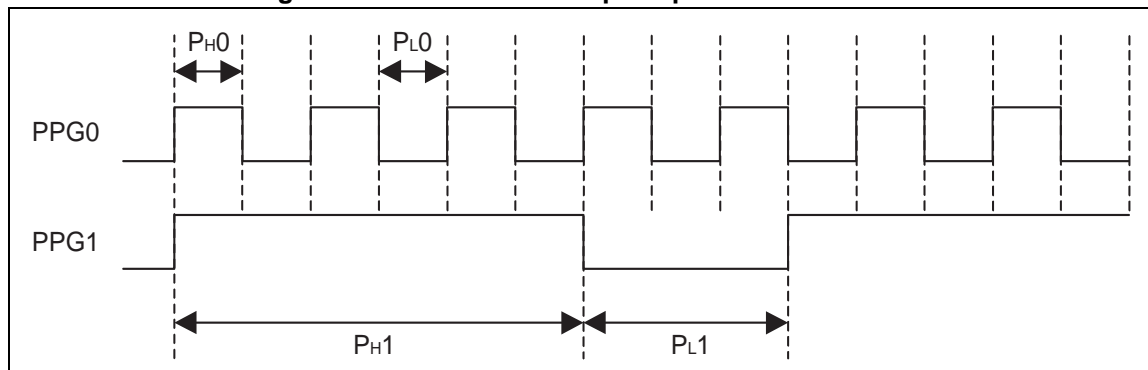
The pulse generated by the 8/16-bit PPG timer can be output from the external pins (PPG0 to PPG5). To output pulses from the external pin, "1" is written in the bit corresponding to each pin. The PPGC0 bit 5 (PE0) and the PPGC1 bit 3 (PE1) are used for the PPG0/PPG2/PPG4 and PPG1/PPG3/PPG5 pins, respectively. When "0" is written in this pin (initial value), no pulse is output from the external pin that then functions as a general-purpose port.

Since the same waveform is output from the PPG0 to PPG5 in the 16-bit PPG mode, either of external pins can be enabled to obtain the same output.

The 8-bit prescaler toggle waveform and the 8-bit PPG waveform are output from the PPG0/PPG2/PPG4 and the PPG1/PPG3/PPG5, respectively, in the 8-bit prescaler + 8-bit PPG mode.

The output waveform in this mode is shown in Figure 17.3-2.

**Figure 17.3-2 8 + 8PPG Output Operation Waveform**



The pulse width shown in Figure 17.3-2 can be calculated by the following expression.

$$P_{L0} = T \times (L0 + 1)$$

$$P_{H0} = T \times (L0 + 1)$$

$$P_{L1} = T \times (L0 + 1) \times (L1 + 1)$$

$$P_{H1} = T \times (L0 + 1) \times (H1 + 1)$$

L0: Value of PRLl of ch.0 and value of PRLH of ch.1

L1: Value of PRLl of ch.1

H1: Value of PRLH of ch.1

T: input clock cycle

P<sub>H0</sub>: Width of "H" pulse of PPG0

P<sub>L0</sub>: Width of "L" pulse of PPG0

P<sub>H1</sub>: Width of "H" pulse of PPG1

P<sub>L1</sub>: Width of "L" pulse of PPG1

---

Note:

PRLl of ch.0 and PRLH of ch.1 set the same value.

---

#### ■ Interrupts of 8/16-bit PPG Timer

The 8/16-bit PPG timer interrupt becomes active when the reload value is counted out and a borrow occurs. In the 8-bit PPG2 channel mode or the 8-bit prescaler + 9 bit PPG mode (each of MB90330A series has 3 channels), each interrupt request is generated by each borrow. In the 16-bit PPG mode, however, the PUF0 and the PUF1 are simultaneously set by the 16-bit counter borrow. Permit, therefore, either only one of PIE0 or PIE1 to unify interrupt factors. Also clear interrupt factors for the PUF0 and the PUF1 at the same time.

#### ■ Initial Value of Hardware Component

The 8/16-bit PPG timer hardware component is initialized to the following value at the reset:

<REGISTERS> PPG0 → 0X000001<sub>B</sub>

PPG1 → 00000001<sub>B</sub>

PPG01 → XXXXXX00<sub>B</sub>

<pulse output> PPG0 → "L"

PPG1 → "L"

PE0 → PPG0 power output interdiction

PE1 → PPG1 power output interdiction

<interruption requests> IRQ0 → "L"

IRQ1 → "L"

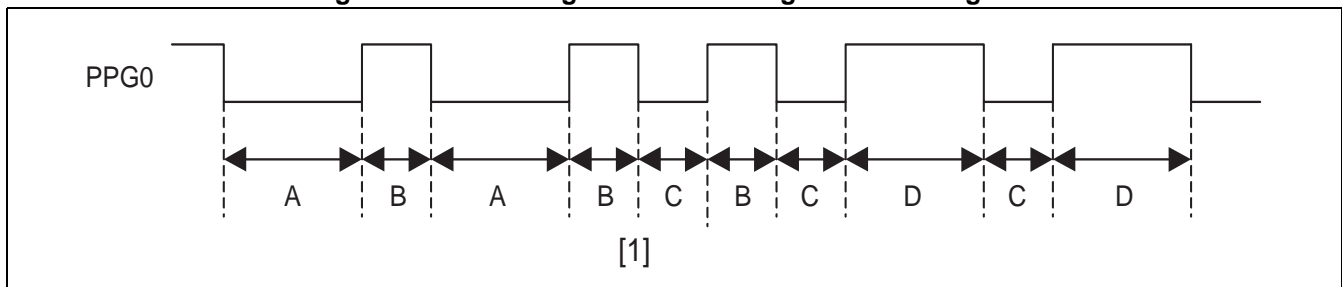
Hardware components not listed above are not initialized.

### ■ Writing Timing to Reload Register

In any modes other than the 16-bit PPG mode, the word transfer instruction is recommended to write data into the reload registers PRLH and PRLH. When the data item is written in the register by using the byte transfer instructions for two times, an unexpected pulse width output may be generated depending on the timing.

Figure 17.3-3 shows the timing for writing in reload register.

**Figure 17.3-3 Timing Chart for Writing in Reload Register**

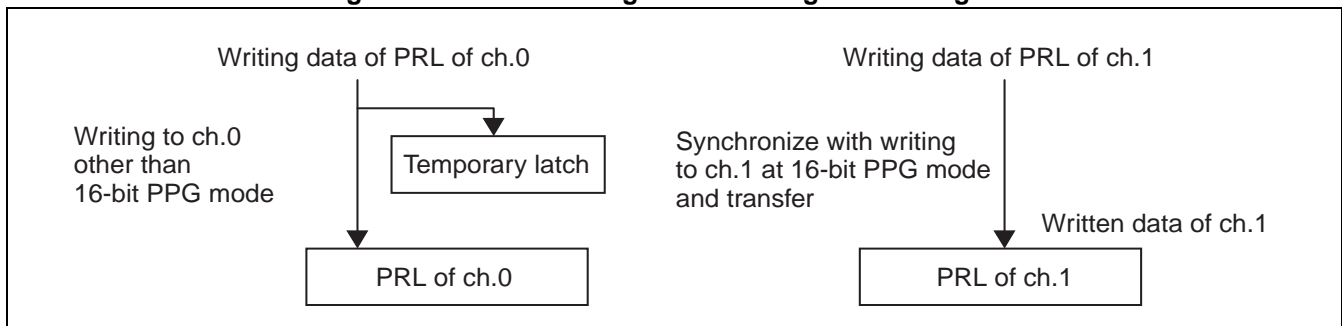


In Figure 17.3-3, when the PRLH is updated from A to C before the [1] timing and the PRLH value is updated from B to D after the [1] timing, the PRL values at the [1] timing are represented as PRLH = C and PRLH = B to generate pulses of the count number C for the L side and the count number B for H side only once. Similarly, in order to write data in the PRL of ch.0/ch.2/ch.4 and ch.1/ch.3/ch.5 in the 16-bit PPG mode, the long-word transfer instruction is used or the word transfer instruction is sequentially used for the PRL of the ch.0 → ch.1 (ch.2 → ch.3/ch.4 → ch.5). In this mode, the data are temporally written from the ch.0/ch.2/ch.4 to the PRL. Then, when they are written in the PRL of ch.1/ch.3/ch.5, they actually written in the PRL of ch.0.

In modes other than the 16-bit PPG mode, data can be written independently to the ch.0/ch.2/ch.4 and ch.1/ch.3/ch.5.

Figure 17.3-4 shows the block diagram of writing to PRL register.

**Figure 17.3-4 Block Diagram of Writing to PRL Register**



# CHAPTER 18

---

## ***DTP/EXTERNAL INTERRUPT***

**This chapter describes an overview of DTP/external interrupt, the configuration and functions of register, and the DTP/external interrupt operation.**

- 18.1 Overview of DTP/External Interrupt
- 18.2 Register of DTP/External Interrupt
- 18.3 Operation of DTP/External Interrupt
- 18.4 Precaution of Using DTP/External Interrupt



## 18.1 Overview of DTP/External Interrupt

The DTP (Data Transfer Peripheral) is located between peripherals existing out of the device and the F<sup>2</sup>MC-16LX CPU. It is the peripheral control section that receives a DMA request or an interrupt request generated by the external peripheral, reports it to the F<sup>2</sup>MC-16LX CPU, and starts the  $\mu$ DMAC or the interrupt processing.

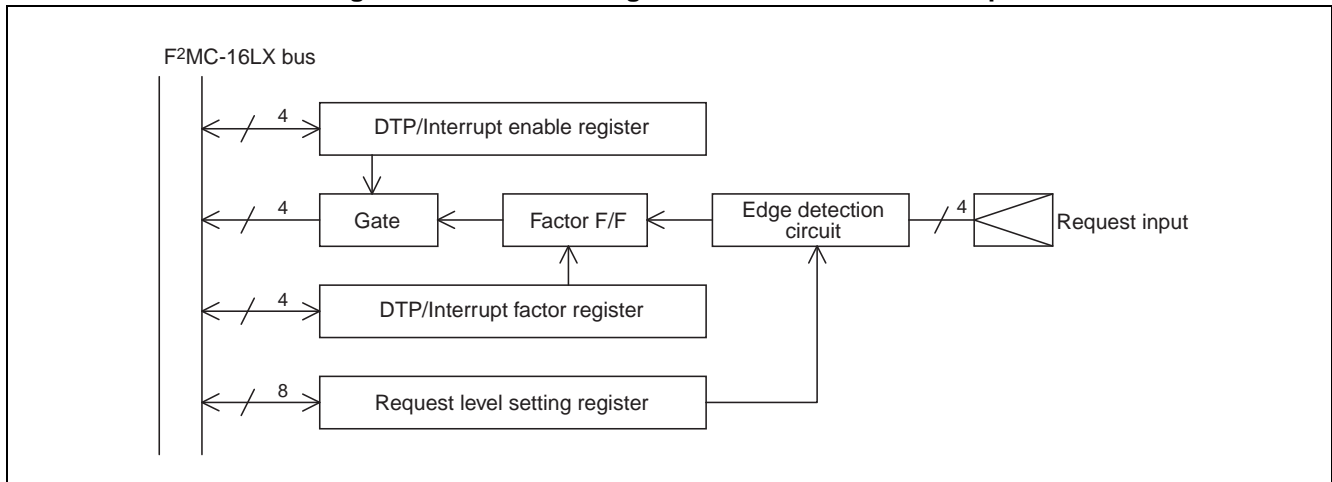
### ■ Overview of DTP/External Interrupt

Two types, "H" and "L" can be selected as the request level for  $\mu$ DMAC. Four types in total, the rising edge and falling edge in addition to "H" and "L" can be selected for an external interrupt request.

### ■ Block Diagram of DTP/External Interrupt

Figure 18.1-1 shows the block diagram of DTP/external interrupt.

Figure 18.1-1 Block Diagram of DTP/External Interrupt



## MB90330A Series

### 18.2 Register of DTP/External Interrupt

This section describes the configuration and functions of registers used for the DTP and external interrupts.

#### ■ Register List of DTP/External Interrupt

Figure 18.2-1 shows the register list of the DTP/external interrupts.

**Figure 18.2-1 Register List of DTP/External Interrupt**

bit	7	6	5	4	3	2	1	0	
Address : 00003C <sub>H</sub>	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0	DTP/Interrupt register (ENIR)
bit	15	14	13	12	11	10	9	8	
Address : 00003D <sub>H</sub>	ER7	ER6	ER5	ER4	ER3	ER2	ER1	ER0	DTP/Interrupt register (EIRR)
bit	7	6	5	4	3	2	1	0	
Address : 00003E <sub>H</sub>	LB3	LA3	LB2	LA2	LB1	LA1	LB0	LA0	Request level setting register (ELVR)
bit	15	14	13	12	11	10	9	8	
Address : 00003F <sub>H</sub>	LB7	LA7	LB6	LA6	LB5	LA5	LB4	LA4	Request level setting register (ELVR)

#### ■ DTP/Interruption Permission Register (ENIR: Enable Interrupt Request Register)

Figure 18.2-2 shows the bit configuration of the DTP and enable interrupt register (ENIR).

**Figure 18.2-2 Bit Configuration of the DTP and the Enable Interrupt Request Register (ENIR)**

ENIR	bit	7	6	5	4	3	2	1	0	Initial value
Address : 00003C <sub>H</sub>		EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0	00000000 <sub>B</sub>
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
R/W: Readable/Writable										

The DTP and the enable interrupt register (ENIR) determine to issue the request to the interrupt controller by using the device pin as the external interrupt and the DTP request input. The pin corresponding to the bits set to "1" in the ENIR register is used to input an external interrupt or DTP request to issue the requests to the interrupt controller. The pin corresponding to the bits set to "0" holds the external interrupt or DTP request input factor but issues no request to the interrupt controller.

### ■ DTP/Interrupt Request Register (EIRR: External Interrupt Request Register)

Figure 18.2-3 shows the bit configuration of DTP/interruption request register (EIRR).

**Figure 18.2-3 Bit Configuration of DTP/interruption Request Register (EIRR)**

EIRR	bit	15	14	13	12	11	10	9	8	Initial value
Address : 00003D <sub>H</sub>		ER7	ER6	ER5	ER4	ER3	ER2	ER1	ER0	00000000 <sub>B</sub>
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	----
R/W: Readable/Writable										(However, the object is different between both of them.)

The DTP/interruption request register (EIRR) indicates the presence of corresponding external DTP/interruption request when reading and clears the flip-flop contents that indicates this request when writing. When "1" is read from the EIRR register it indicates the external DTP/interruption request presence in a pin corresponding to the ER<sub>x</sub> bit. In addition, when "0" is written in the EIRR register, the request flip-flop of the corresponding bit is cleared. Writing "1" causes no operation. "1" is read with a read-modify-write instruction.

#### Notes:

- The initial value is "00<sub>H</sub>" while the value is changed after the reset depending on the status of pin in the common-use with the external interrupt.
- Clear only the bits that the CPU accepted the interrupt (those bits that ER7 to ER0 are set to "1") to "0" when plural external interrupt request outputs are enabled (ENIR: EN7 to EN0 = 1). No other bits must be cleared unconditionally.
- The value of the DTP/external interrupt request bit (EIRR:ER) is available only when the corresponding DTP/external internal enable bit (ENIR:EN) is set to "1". When the DTP/external interrupt is not enabled (ENIR:EN = 0), the DTP/external interrupt factor may be set regardless of whether the DTP/external interrupt request exists or not.
- Clear the corresponding DTP/external interrupt request bit (EIRR:ER) just before enabling the DTP/external interrupts (ENIR:EN = 1).

### ■ Request Level Setting Register (ELVR: External Level Register)

Figure 18.2-4 shows the bit configuration of the request level setting register (ELVR).

**Figure 18.2-4 Request Level Setting Register (ELVR)**

ELVR	bit	7	6	5	4	3	2	1	0	Initial value
Address : 00003E <sub>H</sub>		LB3	LA3	LB2	LA2	LB1	LA1	LB0	LA0	00000000 <sub>B</sub>
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	bit	15	14	13	12	11	10	9	8	Initial value
Address : 00003F <sub>H</sub>		LB7	LA7	LB6	LA6	LB5	LA5	LB4	LA4	00000000 <sub>B</sub>
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
R/W: Readable/Writable										

The request level setting register (ELVR) selects the request detection level. Two bits are allocated per pin as shown in Table 18.2-1. When the request input is in the level mode and the input is active, it is again set even if it is cleared.

**Table 18.2-1 ELVR allocation (LA0-LA7, LB0-LB7)**

LBx	LAx	Operation
0	0	There is a demand at "L" level.
0	1	There is a demand at "H" level.
1	0	Request present at the rising edge
1	1	Request present at the falling edge

## 18.3 Operation of DTP/External Interrupt

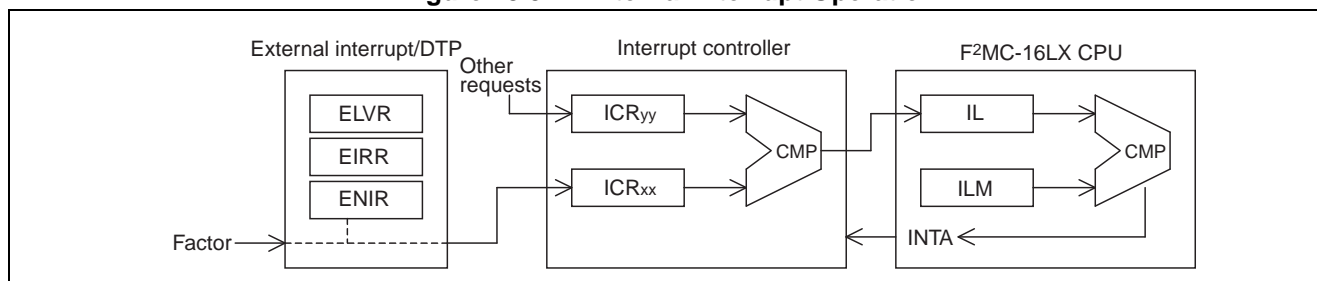
This section describes the Operation of DTP/External Interrupt.

### ■ External Interrupt Operation

If a request set by the ELVR register at the corresponding pin is input after setting an external interrupt request, this resource issues an interrupt request signal for the interrupt controller. When the interrupt from this resource had the highest priority as a result of the priority identification of the interrupts simultaneously occurred in the interrupt controller, the interrupt controller issues an interrupt request to the F<sup>2</sup>MC-16LX CPU. The F<sup>2</sup>MC-16LX CPU compares the interrupt level mask register (ILM) in the processor status (PS) with the interrupt request. When the request level is higher than the ILM bit, the hardware interrupt process microprogram is started as soon as the current instruction execution is terminated.

Figure 18.3-1 shows the external interrupt operation flow.

Figure 18.3-1 External Interrupt Operation



The interrupt process microprogram reads the interrupt vector area, issues the interrupt acknowledge to the interrupt controller, transfers the macro instruction jump address generated by the vector to the program counter, and executes the user interrupt process program.

### ■ Operation of DTP

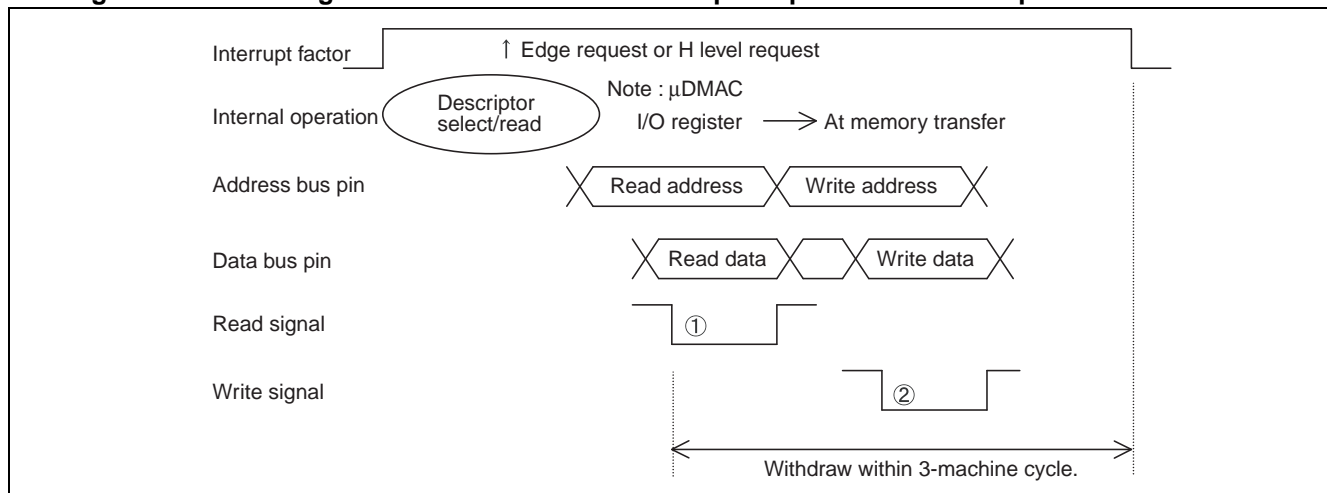
As an initialization in the user program before starting the  $\mu$ DMAC, the register addresses allocated in from 000000<sub>H</sub> to 0000FF<sub>H</sub> are set to the I/O address pointer in the  $\mu$ DMAC descriptor and the memory buffer start address is set to the buffer address pointer.

The DTP operation sequence is almost same with those of external interrupt and are quite identical until the CPU starts the hardware interrupt process microprogram. When the  $\mu$ DMAC is started, the read or write signal is sent to the addressed external peripheral device for the transfer operation with this chip. The external peripheral device must cancel the interrupt request to this chip within three machine cycles after the transfer operation. When the transfer is terminated, the descriptor, etc. are updated and the interrupt controller generates the signal to clear the transfer factor. When this resource receives the signal to clear the transfer factor, it clears the flip-flop that holds the factor and prepares for the next pin request.

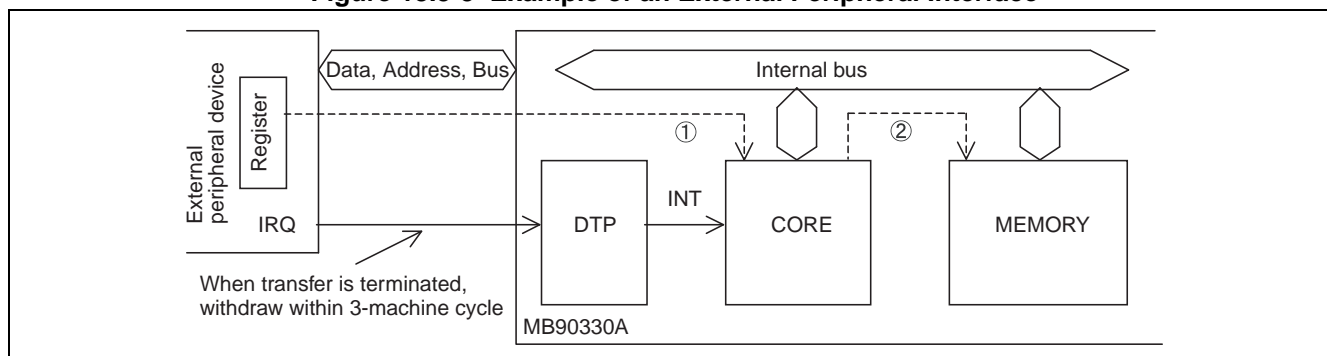
Figure 18.3-2 shows the timing to cancel the external interrupt request at the DTP operation termination.

In addition, Figure 18.3-3 shows the example of interfaces with external peripheral devices.

**Figure 18.3-2 Timing to Cancel the External Interrupt Request at the DTP Operation Termination**



**Figure 18.3-3 Example of an External Peripheral Interface**



## 18.4 Precaution of Using DTP/External Interrupt

Notes DTP/an external interruption is used are explained.

### ■ Condition of Peripheral Equipment Connected Outside

The external peripheral device that the DTP can support must automatically clear the request at the transfer execution. In addition, if the transfer request is not canceled within three-machine-cycle (an interim value) after the transfer operation started, this resource handles it as if the next transfer request occurs.

### ■ Operation Process of DTP/External Interrupt

Set the registers within the DTP/external interrupt by using the following procedures:

1. Set the general purpose I/O port that is shared with the external interrupt as input port.
2. Disable the bits for the enable register.
3. The target bit for request level set register is set.
4. The target bit for the factor register is cleared.
5. Enable the bits for the enable register.

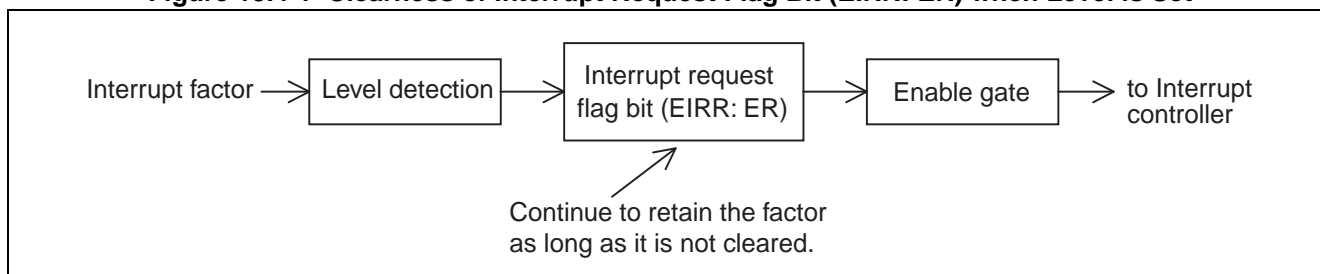
Simultaneous writing is possible for 4 and 5 with the word specification.

Before setting registers in this resource, the enable register must be disabled. In addition, before enabling the enable register, the factor register must be cleared. These actions prevent erroneous interrupt factor occurrence at register setting or in the interrupt enable state.

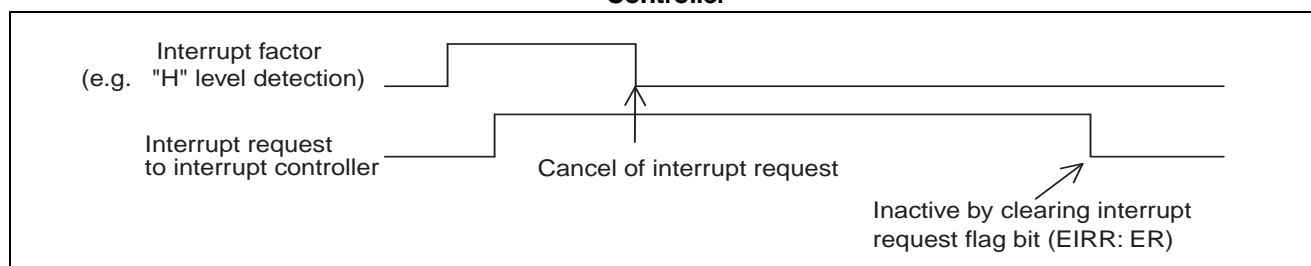
### ■ External Interrupt Request Level

- Minimum 3 machine cycles are necessary for the pulse width to detect the edge presence when the request level is set to the edge request.
- When the request input level is set to the level setting, the request to the interrupt controller remains active even if an external request is input and canceled afterward, in case of the interrupt request enable state (ENIR:EN=1). To cancel the request for the interrupt controller, the interrupt request flag bit (EIRR: ER) must be cleared.

Figure 18.4-1 Clearness of Interrupt Request Flag Bit (EIRR: ER) when Level is Set



**Figure 18.4-2 Interrupt Factor when Enabling the Interrupt and the Interrupt Request for the Interrupt Controller**



Note:

Returning from the clock mode is impossible at the edge detection.





# **CHAPTER 19**

---

## **8/10-BIT A/D CONVERTER**

**This chapter explains the functions and operation of 8/10-bit A/D converter.**

- 19.1 Overview of 8/10-bit A/D Converter
- 19.2 Configuration of 8/10-bit A/D Converter
- 19.3 Register of 8/10-bit A/D Converter
- 19.4 Explanation of Operation of 8/10-bit A/D Converter
- 19.5 Precautions when Using 8/10-bit A/D Converter
- 19.6 Example of program-1 of 8/10-bit A/D Converter  
(Example of Starting the EI<sup>2</sup>OS in the Single Mode)
- 19.7 Example of Program-2 of 8/10-bit A/D Converter  
(Example of Starting the EI<sup>2</sup>OS in the Continuous Mode)
- 19.8 Example of Program-3 of 8/10-bit A/D Converter  
(Example of Starting the EI<sup>2</sup>OS in the Stop Mode)

## 19.1 Overview of 8/10-bit A/D Converter

The 8/10-bit A/D converter has a function to convert the analog input voltage to the 8-bit or 10-bit digital value by the RC sequential comparison conversion method. The input signal can be selected from the 16-channel analog input pin. The conversion start can be selected from three types of the software, the PPG timer 1 output, and the trigger input start from external pin.

### ■ Function of 8/10-bit A/D Converter

There is a function having following features to perform the A/D conversion of the analog voltage input from the analog input pin (input voltage) to the digital value:

- The conversion time is a minimum of 7.16  $\mu$ s (including the sampling time using the 24 MHz machine clock).
- Sampling time is 3.33  $\mu$ s or more (machine clock 6 MHz time).
- The used conversion method is the RC sequential compare conversion method with sample hold circuit.
- The analog input pin can be selected from the 16 channels using the program.
- The interrupt request can be generated at the A/D conversion termination and the EI<sup>2</sup>OS can be also started.

No data are lost in the interrupt enable status since the conversion data protection function works even in the continuous conversion.

- The conversion start factor can be selected from the software, the PPG timer 1 output (rising edge), and the external trigger input (falling edge).

Conversion mode has three types shown in Table 19.1-1.

**Table 19.1-1 Conversion Modes of 8/10-bit A/D Converter**

Conversion Mode	Single conversion operation	Scanning conversion operation
Single-shot conversion mode	The specified channel (one channel only) is converted for one time and terminated.	Continuous plural channels (maximum 16 channels can be specified) are converted for one time and terminated.
Continuous conversion mode	The specified channel (one channel only) is repetitively converted.	Continuous plural channels (maximum 16 channels can be specified) are repetitively converted.
Pause-conversion mode	The specified channel (one channel only) is converted for one time and suspended until the next start.	Continuous plural channels (maximum 16 channels can be specified) are converted. However, one channel is converted and suspended until the next start.

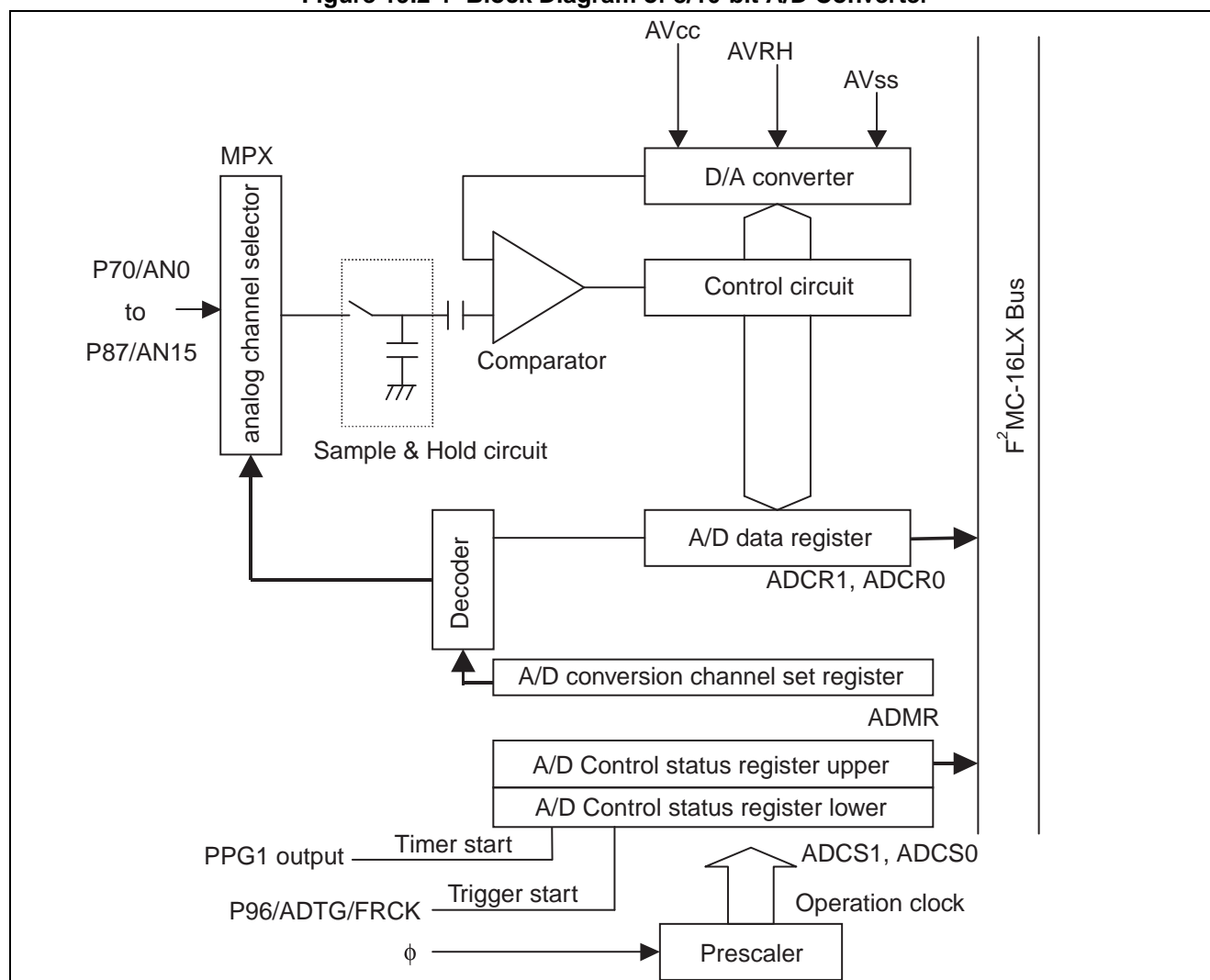
## 19.2 Configuration of 8/10-bit A/D Converter

8/10-bit A/D converter is composed of the following 9 blocks.

- A/D control status register (ADCS)
- A/D data register (ADCR)
- A/D conversion channel set register (ADMR)
- Decoder
- Analog channel selector
- Sample hold circuit
- D/A converter
- Comparator
- Control circuit (Sequential comparison register)

### ■ Block Diagram of 8/10-bit A/D Converter

Figure 19.2-1 Block Diagram of 8/10-bit A/D Converter



● A/D control status registers (ADCS)

Displayed are the start and start trigger selection by software, the conversion mode selection, the A/D conversion channel selection, the enabled and disabled interrupt requests, the interrupt request status confirmation, the suspension status and the conversion status.

● A/D data registers (ADCR)

This register stores the A/D conversion result and also has a function to select the A/D conversion resolution.

● A/D conversion channel set register (ADMR)

This register selects the A/D conversion channel.

● Decoder

This circuit selects the analog input pin to be used from the setting of bits from ANE0 to ANE3 and from ANS0 to ANS3 of A/D conversion channel set register (ADMR).

● Analog channel selector

This circuit selects the pin to be used from the 16 analog input pins.

● Sample hold circuit

This circuit holds the input voltage selected by the analog channel selector. The input voltage can be converted without affected by the input voltage fluctuation in the A/D conversion (in the comparison) by holding the sample of input voltage immediately after starting the A/D conversion.

● D/A converter

The reference voltage is generated to compare the held sample of input voltage.

● Comparator

This compares the input voltage for which sample hold is performed, with the output voltage of the D/A converter to determine which is the greater of the two.

● Control circuit (Sequential comparison register)

The signal from the comparator (higher or lower) determines the A/D conversion value. When the A/D conversion is terminated, the conversion result is stored in the A/D data register (ADCR) and the interrupt request is generated.

## 19.3 Register of 8/10-bit A/D Converter

Register list of 8/10-bit A/D converter is shown.

■ Register List of 8/10-bit A/D Converter

Figure 19.3-1 Register of 8/10-bit A/D Converter

Address: 000040 <sub>H</sub>	bit 7	6	5	4	3	2	1	0	A/D Control status register (Lower) (ADCS0)
	MD1	MD0	—	—	—	—	—	Reserved	
	(R/W)	(R/W)	(—)	(—)	(—)	(—)	(—)	(R/W)	Initial value 00-----0 <sub>B</sub>
Address: 000041 <sub>H</sub>	bit 15	14	13	12	11	10	9	8	A/D Control status register (Upper) (ADCS1)
	BUSY	INT	INTE	PAUS	STS1	STS0	STRT	Reserved	
	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(W)	(R/W)	Initial value 00000000 <sub>B</sub>
Address: 000042 <sub>H</sub>	bit 7	6	5	4	3	2	1	0	A/D Data register (Lower) (ADCR0)
	D7	D6	D5	D4	D3	D2	D1	D0	
	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	Initial value XXXXXXXX <sub>B</sub>
Address: 000043 <sub>H</sub>	bit 15	14	13	12	11	10	9	8	A/D Data register (Upper) (ADCR1)
	S10	ST1	ST0	CT1	CT0	—	D9	D8	
	(R/W)	(W)	(W)	(W)	(W)	(—)	(R)	(R)	Initial value 00101XXX <sub>B</sub>
Address: 000045 <sub>H</sub>	bit 15	14	13	12	11	10	9	8	A/D Conversion Channel Set register (ADMR)
	ANS3	ANS2	ANS1	ANS0	ANE3	ANE2	ANE1	ANE0	
	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	Initial value 00000000 <sub>B</sub>

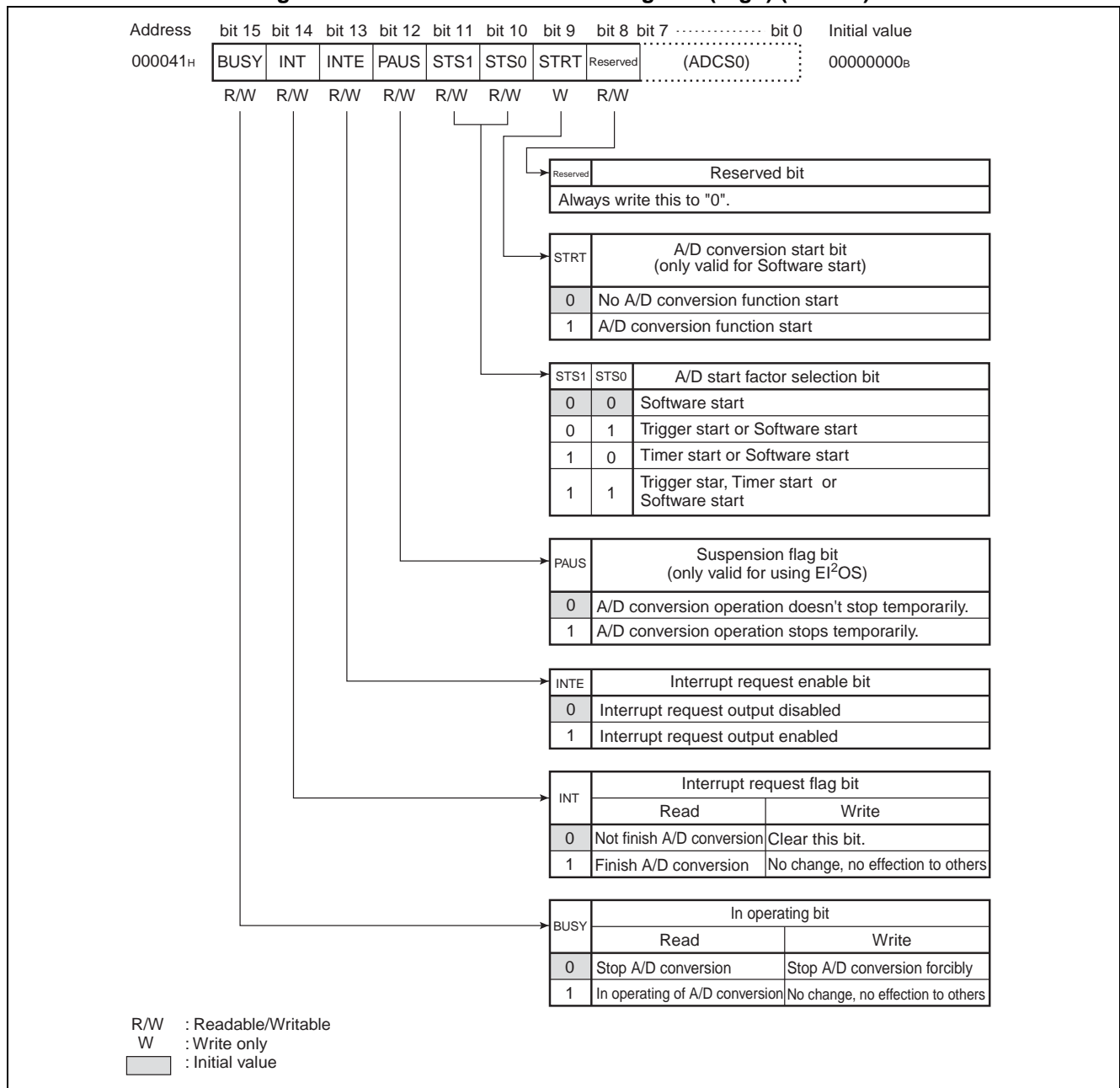
R/W: Readable/Writable

## 19.3.1 A/D Control Status Register (High) (ADCS1)

The A/D control status register "H" level (ADCS1) has functions to start the software, to select the start trigger, to enable and disable the interrupt request, and to confirm the interrupt request, suspension and conversion statuses.

### ■ A/D Control Status Register (High) (ADCS1)

Figure 19.3-2 A/D Control Status Register (High) (ADCS1)



**Table 19.3-1 Description and Functions of Each Bit of A/D Control Status Register (High) (ADCS1)**

Bit name		Functions
bit 15	BUSY: Under the conversion bit	<ul style="list-style-type: none"> <li>• The operation display bit of A/D converter.</li> <li>• When the BUSY bit is "0" and "1" in the read process, it indicates the stopped A/D conversion and the operated A/D conversion, respectively.</li> <li>• When "0" is written in the BUSY bit in the write process, the A/D conversion operation is forcibly stopped. When "1" is written, the conversion is not changed and no others are affected.</li> </ul> <p>Note: Do not perform the forced stop and the software start (BUSY = 0, STRT = 1) at the same time.</p>
bit 14	INT: Interrupt request flag bit	<ul style="list-style-type: none"> <li>• When the data are set to the A/D data register by the A/D conversion, the INT bit is set to "1".</li> <li>• When the INT bit and the interrupt request enabled bit (ADCS1: INTE) are set to "1", the interrupt request is generated. If EI<sup>2</sup>OS has been permitted, EI<sup>2</sup>OS is started.</li> <li>• When "0" is set in the write process, the INT bit is cleared. When "1" is set, it is not changed and no others are affected.</li> <li>• The INT bit is cleared by the startup of EI<sup>2</sup>OS.</li> </ul> <p>Note: The A/D must be stopped when writing "0" in the INT bit for clearing.</p>
bit 13	INTE: Interrupt demand permission bit	<ul style="list-style-type: none"> <li>• This bit is used to enable and disable the interrupt output to CPU.</li> <li>• When the INTE bit and the interrupt request flag bit (ADCS1: INT) are set to "1", the interrupt request is generated.</li> <li>• Set it to "1" when using the EI<sup>2</sup>OS.</li> </ul>
bit 12	PAUS: Temporary stop flag bit	<ul style="list-style-type: none"> <li>• It is set to "1" when the A/D conversion operation is suspended.</li> <li>• Since this A/D converter has only one A/D data register, when the CPU has not yet completed to read the old conversion result by using the continuous conversion mode, the old conversion data are lost by a new conversion result. Therefore, to use the continuous conversion mode, the automatic transfer of conversion result to the memory must be basically set by using the EI<sup>2</sup>OS at every conversion termination. However, such case can be assumed that the conversion data transfer is not completed before the next conversion in the multiple interrupts, etc. The PAUS bit is the function to prepare for the such case. When the PAUS bit is set to "1", the A/D conversion is stopped so that the next conversion data are not stored during the period from the conversion termination to the data register contents transfer with the EI<sup>2</sup>OS. After that, the A/D converter automatically restarts the conversion at the transfer termination with the EI<sup>2</sup>OS.</li> </ul> <p>Note: Only when EI<sup>2</sup>OS is used, the PAUS bit is effective.</p>
bit 11, bit 10	STS1, STS0: A/D startup factor selection bits	<ul style="list-style-type: none"> <li>• The start factor of A/D conversion is selected.</li> <li>• When the start factor is in the common use, the first start factor generation starts the operation.</li> </ul> <p>Note: Switch the start factor when the target start factor is not present to update it during the A/D conversion operation because it is changed at the same time of updating.</p>
bit 9	STRT: Analog to digitalizing A/D converter start up bit	<ul style="list-style-type: none"> <li>• This bit is used to start the A/D conversion operation by the software.</li> <li>• When "1" is written in the STRT bit, the A/D conversion is started.</li> <li>• During the suspend conversion mode, the restart operation by STRT bit is not performed.</li> </ul> <p>Note: Forcibly stop and software start up must not be done at the same time.</p>
bit 8	Reserved: reserved bit	<p>Note: Be sure to write "0".</p>

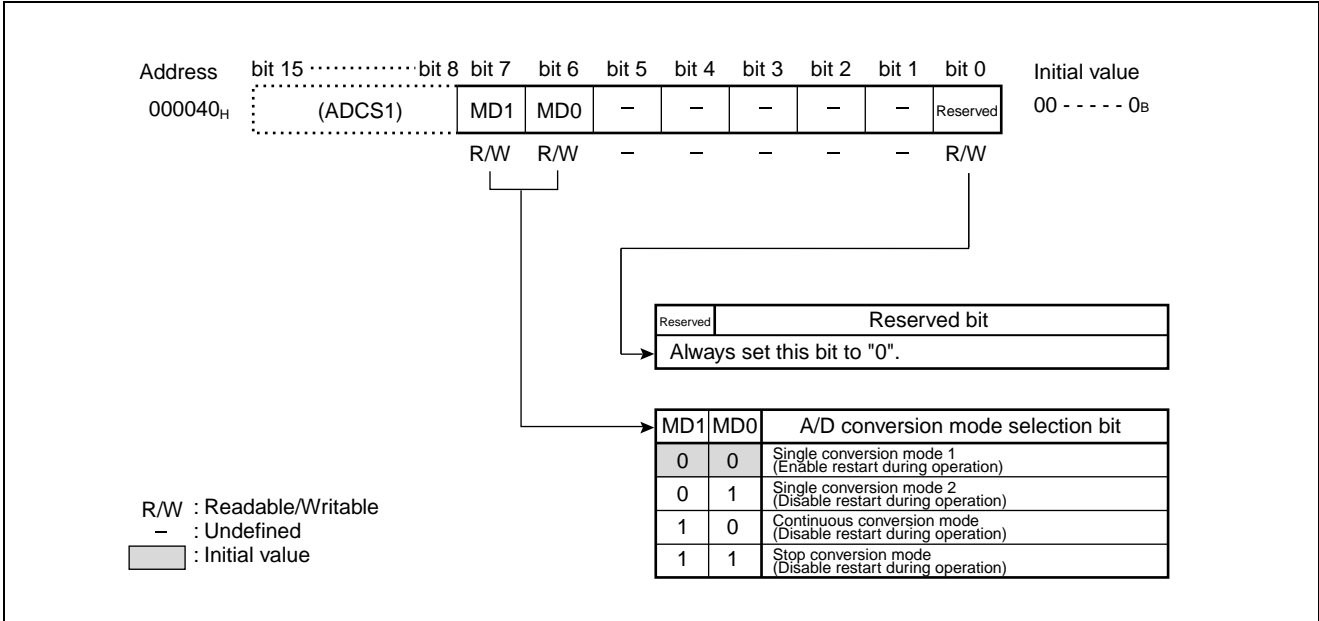


19.3.2 A/D Control Status Register (Low) (ADCS0)

The A/D control status register low-level (ADCS0) has a function to select the conversion mode.

■ A/D Control Status Register (Low) (ADCS0)

Figure 19.3-3 A/D Control Status Register (Low) (ADCS0)



**Table 19.3-2 Description and Functions of Each Bit of A/D Control Status Register (Low) (ADCS0)**

Bit name		Functions
bit 7, bit 6	MD1, MD0: A/D conversion mode selection bits	<ul style="list-style-type: none"> <li>• This bit is used to select the conversion mode during the A/D conversion operation.</li> <li>• Two bit values of MD1 and MD0 allow the selection of either the single conversion mode 1, the single conversion mode 2, the continuous conversion mode, or the stop conversion mode.</li> <li>• The meaning of each mode is as follows.                             <p>Single conversion mode 1: The continuous A/D conversion from the setting channels of ANS3 to ANS0 to the setting channels of ANE3 to ANE0 is performed only once. It is possible to reactivate while operating.</p> <p>Single conversion mode 2: The continuous A/D conversion from the setting channels of ANS3 to ANS0 to the setting channels of ANE3 to ANE0 is performed only once. It is not possible to reactivate while operating.</p> <p>Continuous conversion mode: The continuous A/D conversion from the setting channels of ANS3 to ANS0 to the setting channels of ANE3 to ANE0 is repetitively performed until it is forcibly stopped with the BUSY bit. It is not possible to reactivate while operating.</p> <p>Stop conversion mode: The A/D conversion from the setting channels of ANS3 to ANS0 to the setting channel of ANE3 to ANE0 is repeated and suspended at every channel until it is forcibly stopped with the BUSY bit. It is not possible to reactivate while operating. The suspended conversion is restarted by the start factor occurrence selected by the STS1 and STS0 bits.</p> </li> </ul> <p>Note:</p> <p>The disabled restart in each of the single, continuous, and stop conversion modes is applied to the external trigger and software start.</p>
bit 5 to bit 1	Undefined bits	The read value is irregular. Nothing is affected when it is written.
bit 0	Reserved bit	Be sure to set this bit to "0".

### 19.3.3 A/D Conversion Channel Set Register (ADMR)

A/D conversion channel set register (ADMR) has the function to select the A/D conversion.

#### ■ A/D Conversion Channel Set Register (ADMR)

Figure 19.3-4 A/D Conversion Channel Set Register (ADMR)

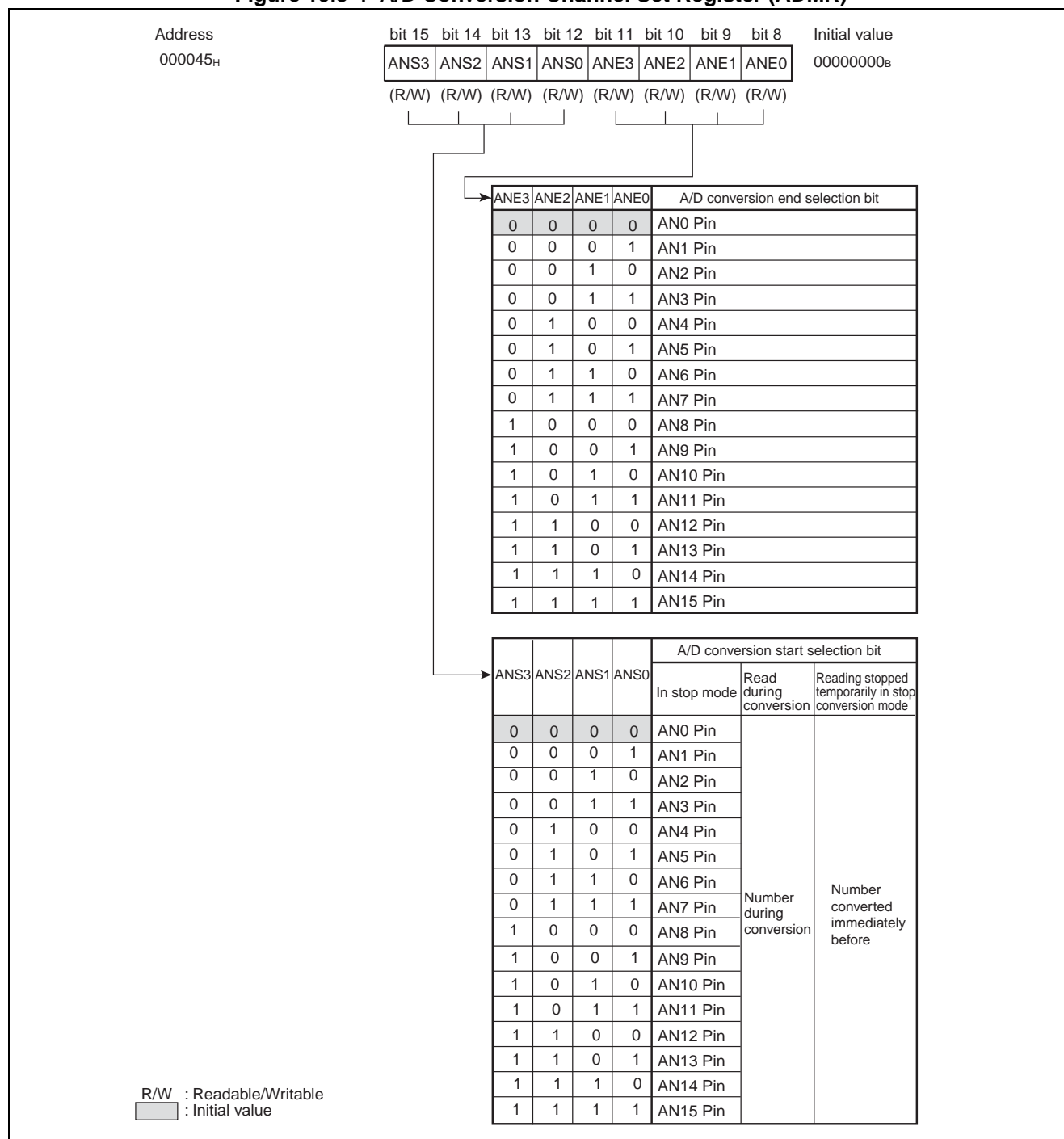


Table 19.3-3 Description and Functions of Each Bit of A/D Conversion Channel Set Register (ADMR)

Bit name		Functions
bit 15 to bit 12	ANS3, ANS2, ANS1, ANS0: A/D conversion beginning channel selection bits	<ul style="list-style-type: none"> <li>• This bit set the start channel of the A/D conversion and indicates the channel numbers under A/D conversion during conversion operation.</li> <li>• When the A/D conversion is started, it is begun from the channels written in these bits.</li> <li>• Channel numbers under the conversion can be read during the A/D conversion. The channel number converted immediately before can be read during the suspension in the stop conversion mode.</li> </ul>
bit 11 to bit 8	ANE3, ANE2, ANE1, ANE0: A/D conversion end channel selection bits	<ul style="list-style-type: none"> <li>• This bit sets the end channel of the A/D conversion.</li> <li>• A/D conversion is performed up to the specified channel.</li> <li>• When the same channels with ANS3 to ANS0 are set, only those channels are converted. In addition, when the continuous conversion mode or the stop conversion mode is set and the conversion is completed for the channels set by those bits, the start channel set by the ANS3 to ANS0 is resumed.</li> </ul> <p>Note: Do not set the A/D conversion mode selection bits (MD1 and MD0) and A/D conversion end channel selection bits (ANE3, ANE2, ANE1 and ANE0) through read-modify-write commands after the start channel is set in the A/D conversion start channel selection bits (ANS3, ANS2, ANS1 and ANS0). The ANS3, ANS2, ANS1 and ANS0 bits will read the last conversion channel until the A/D conversion operation starts. Accordingly when the MD1 and MD0 bits and the ANE3, ANE2, ANE1 and ANE0 bits are set through read-modify-write commands after the start channel is set in the ANS3, ANS2, ANS1 and ANS0 bits, the values of the ANE3, ANE2, ANE1 and ANE0 bits may be rewritten.</p>

---

Note:

Setting that the A/D conversion end channel is smaller than the A/D conversion start channel (ANS > ANE) is disabled. Analog to digital conversion channel set register (ADMR) should be set by byte access.

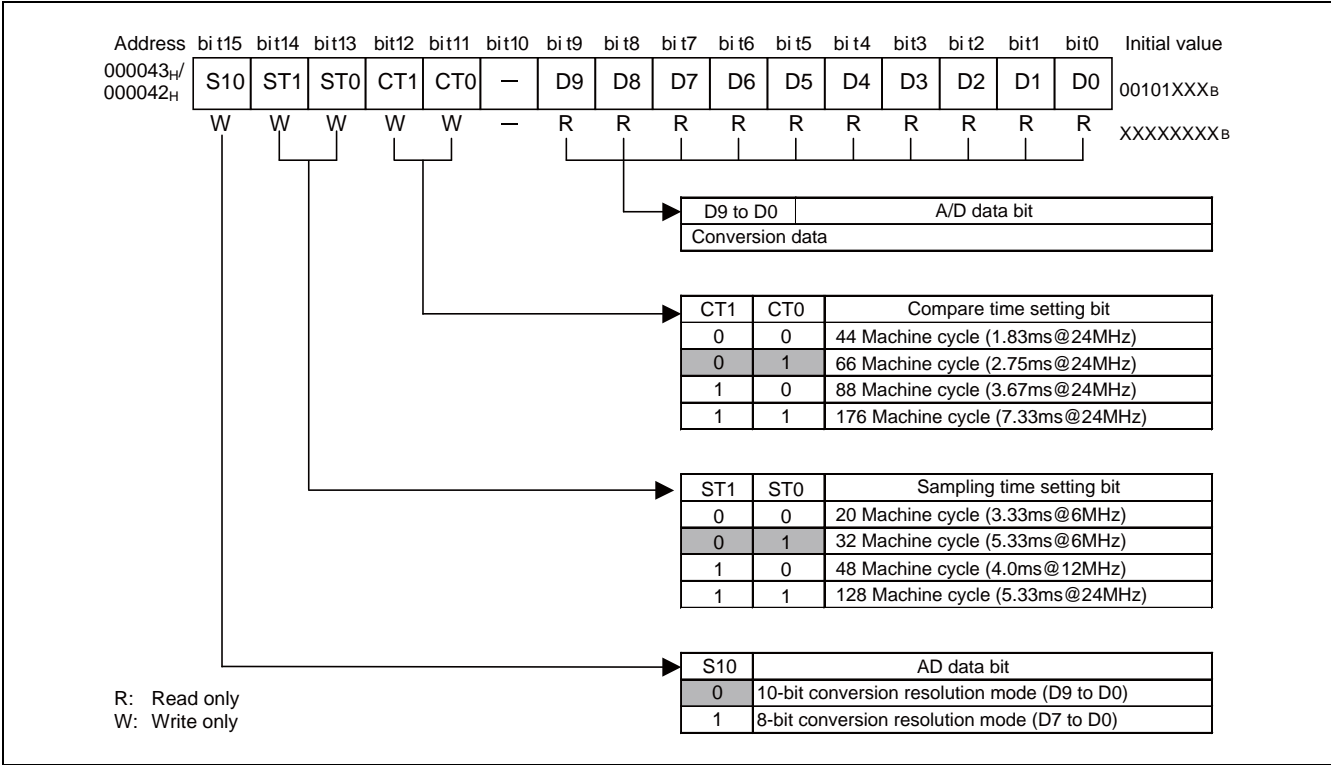
---

19.3.4 A/D Data Register (ADCR1/ADCR0)

The A/D data register (ADCR1/ADCR0) stores the A/D conversion result and also has a function to select the A/D conversion resolution.

■ A/D Data Register (ADCR1/ADCR0)

Figure 19.3-5 A/D Data Register (ADCR1/ADCR0)



**Table 19.3-4 Functional Explanation of Each Bit in the A/D Data Register (ADCR0/ADCR1)**

Bit name		Functions
bit 15	S10: A/D conversion resolution selection bit	<ul style="list-style-type: none"> <li>• This bit is used to select the A/D conversion resolution.</li> <li>• When "0" and "1" are written in the S10 bit, 10-bit and 8-bit resolutions are selected, respectively.</li> </ul> Note: The A/D data bits used are different depends on the resolution.
bit 14, bit 13	ST1,ST0: Sampling time setting bit	<ul style="list-style-type: none"> <li>• This bit is used to select the sampling time at the A/D conversion.</li> <li>• When the A/D is started, the set time and the analog input are fetched into the ST1 and ST0 bits.</li> </ul> Note: When the setting for "00" 6 MHz is provided during the 24 MHz operation, the normal analog voltage sometime cannot be fetched.
bit 12, bit 11	CT1,CT0: Comparison time setting bit	<ul style="list-style-type: none"> <li>• This bit is used to select the comparison time at the A/D conversion.</li> <li>• The analog input is fetched (sampling time elapsed), the CT1 and CT0 bits are set, the conversion result data is determined, and then they are stored in the bit 9 to bit 0 of this register.</li> </ul> Note: When the setting for "00" 6 MHz is provided during the 24 MHz operation, the normal analog conversion value sometime cannot be fetched.
bit 10	Undefined bit	The read value is irregular. Nothing is affected when it is written.
bit 9 to bit 0	D9 to D0	<ul style="list-style-type: none"> <li>• The A/D conversion result is stored and the register is updated at every conversion termination.</li> <li>• The final conversion value is stored usually.</li> <li>• The initial value of the ADCR register is irregular.</li> </ul> Note: The conversion data protection function is provided. (Please refer to Section "19.4 Explanation of Operation of 8/10-bit A/D Converter".) Do not write data in the D9 to D0 bits during the A/D conversion.

**Notes:**

- Rewrite the S10 bit always when the A/D operation before the conversion is stopped. When updating after the conversion, the ADCR contents become undefined.
- Always use the word transfer instruction (MOVW A, 0042H, etc.) to read the ADCR register by specifying the 10 bit mode.

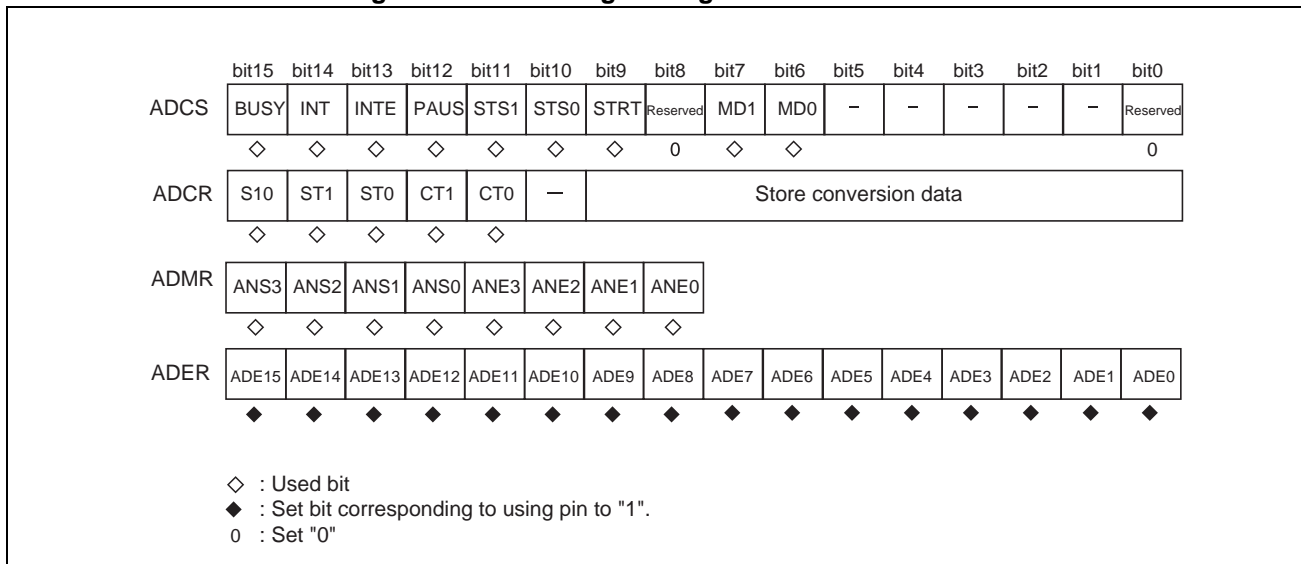
## 19.4 Explanation of Operation of 8/10-bit A/D Converter

Three mode types, the single conversion, continuous conversion, and stop conversion modes are available for the 8/10-bit A/D converter. The operation explanation in each mode is done.

### ■ Operation of Single-shot Conversion Mode

In single conversion mode, it sequentially converts the analog input on the channels which have been set by the ANS bit and ANE bit, and when it reaches to the end channel set in ANE bit, it stops the A/D conversion. When the start channel and the termination channel are same (ANS = ANE), only one channel specified by the ANS bit is converted. For the operation in the single conversion mode, the setting as shown in Figure 19.4-1 is necessary.

Figure 19.4-1 Setting of Single Conversion Mode



Example of Conversion Order in Single Conversion Mode.

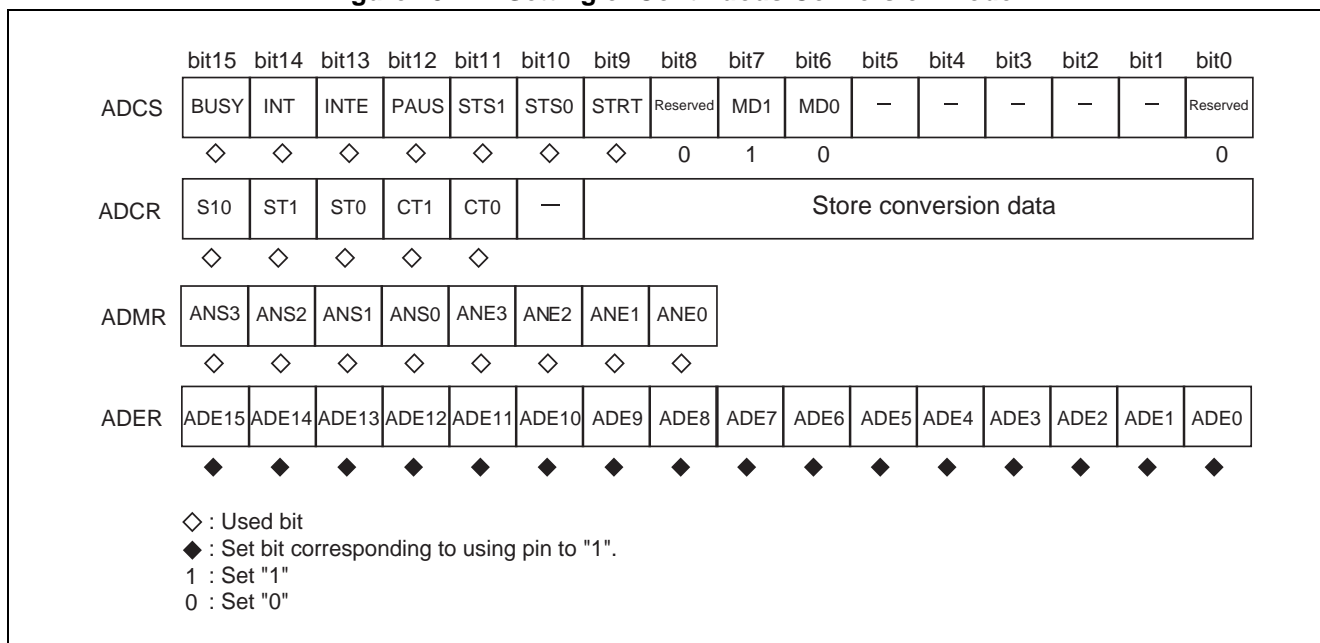
In the case of ANS = 000<sub>B</sub> and ANE = 011<sub>B</sub>: AN0 → AN1 → AN2 → AN3 → termination

In the case of ANS = 011<sub>B</sub> and ANE = 011<sub>B</sub>: AN3 → termination

## ■ Operation of Continuous Conversion Mode

In the continuous conversion mode, the analog inputs set by the ANS and ANE bits are sequentially converted, the analog input set by the ANE bit is resumed at the end of conversion of the termination channel set by the ANS bit, and the A/D conversion operation is continued. When the start channel and the termination channel are same (ANS = ANE), the conversion of only channels specified by the ANS is repeated. For the operation in the continuous conversion mode, the setting as shown in Figure 19.4-2 is necessary.

Figure 19.4-2 Setting of Continuous Conversion Mode



Example of Conversion Order in Continuous Conversion Mode is shown below.

In the case of ANS = 000<sub>B</sub> and ANE = 011<sub>B</sub>: AN0 → AN1 → AN2 → AN3 → AN0 → repeat

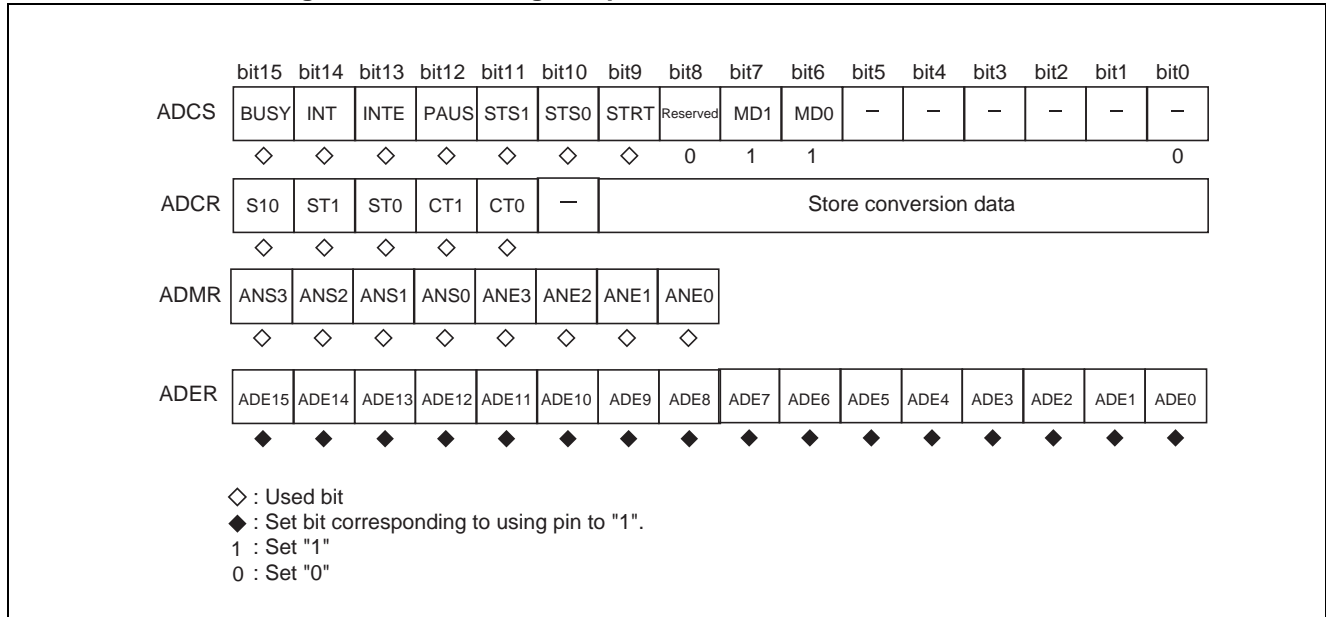
In the case of ANS = 011<sub>B</sub> and ANE = 011<sub>B</sub>: AN3 → AN3 → repeat



### ■ Operation of Pause-conversion Mode

In the stop conversion mode, the analog input set by the ANS and ANE bits is converted by being suspended for every channel, the analog input set by the ANE bit is resumed at the end of conversion of the termination channel set by the ANS bit, and the operation of A/D conversion and suspension is continued. When the start channel and the termination channel are same (ANS = ANE), the conversion of only channels specified by the ANS is repeated. When the conversion is restarted during the suspension, the start factor specified by the STS1 and STS0 bits is generated. For the operation in the stop conversion mode, the setting as shown in Figure 19.4-3 is necessary.

**Figure 19.4-3 Setting of Operation of Pause Conversion Mode**



Example of Conversion Order in Pause-conversion Mode is shown below.

In the case of ANS=000<sub>B</sub>, ANE=011<sub>B</sub>: AN0 → Suspension → AN1 → Suspension → AN2 →  
 Suspension → AN0 → Repetitive

In the case of ANS = 011<sub>B</sub> and ANE = 011<sub>B</sub>: AN3 → Suspension → AN3 → Suspension → Repetitive

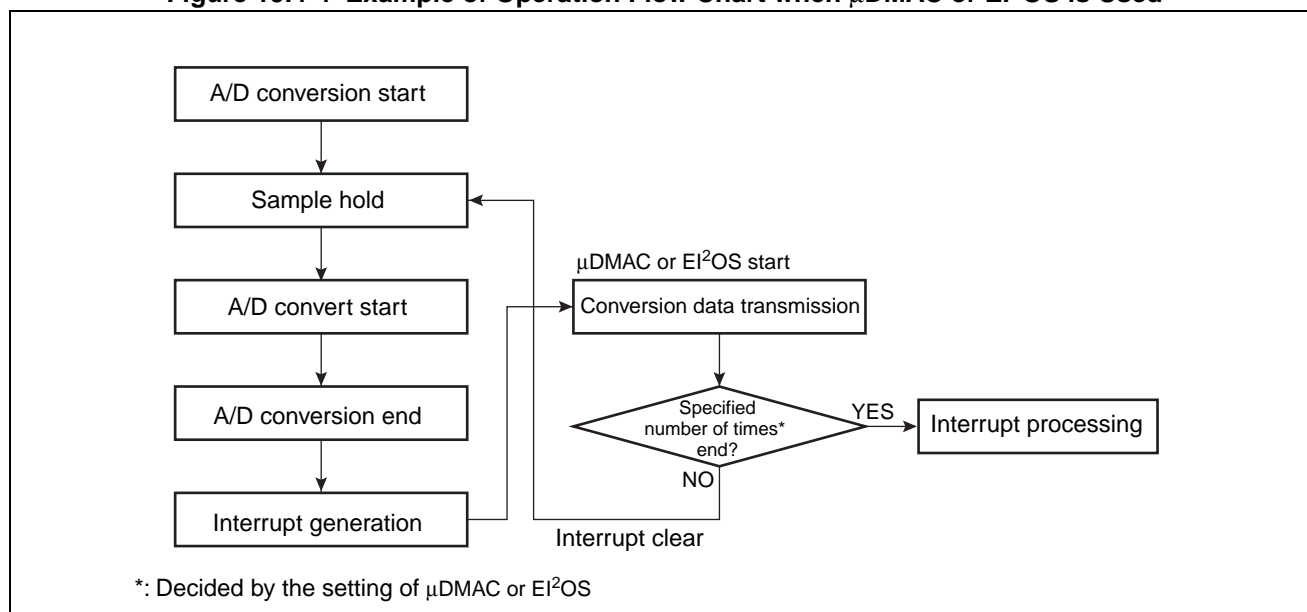
### 19.4.1 Conversion Operation Using $\mu$ DMAC or EI<sup>2</sup>OS

The 10-bit A/D converter allows the transfer of A/D conversion result to memory using the  $\mu$ DMAC or EI<sup>2</sup>OS.

#### ■ Conversion Operation Using $\mu$ DMAC or EI<sup>2</sup>OS

Figure 19.4-4 shows the operation flow when using the  $\mu$ DMAC or EI<sup>2</sup>OS.

**Figure 19.4-4 Example of Operation Flow Chart when  $\mu$ DMAC or EI<sup>2</sup>OS is Used**



By using the  $\mu$ DMAC or EI<sup>2</sup>OS, the conversion data protection function allows the assured data transfer to the memory without losing data even in the continuous conversion.

## 19.4.2 A/D-converted Data Protection Function

---

**When the A/D conversion is executed in the interrupt enable status, the conversion data protection function works.**

---

### ■ A/D-converted Data Protection Function

Since this A/D converter has only one data register to store the converted data, the A/D conversion operation rewrites the data stored in data register at the conversion termination. Therefore, when the converted data transfer to the memory is delayed, a part of previous data may be lost. To cope with this problem, when the interrupt is enabled (INTE = 1), the data protection function operates as follows:

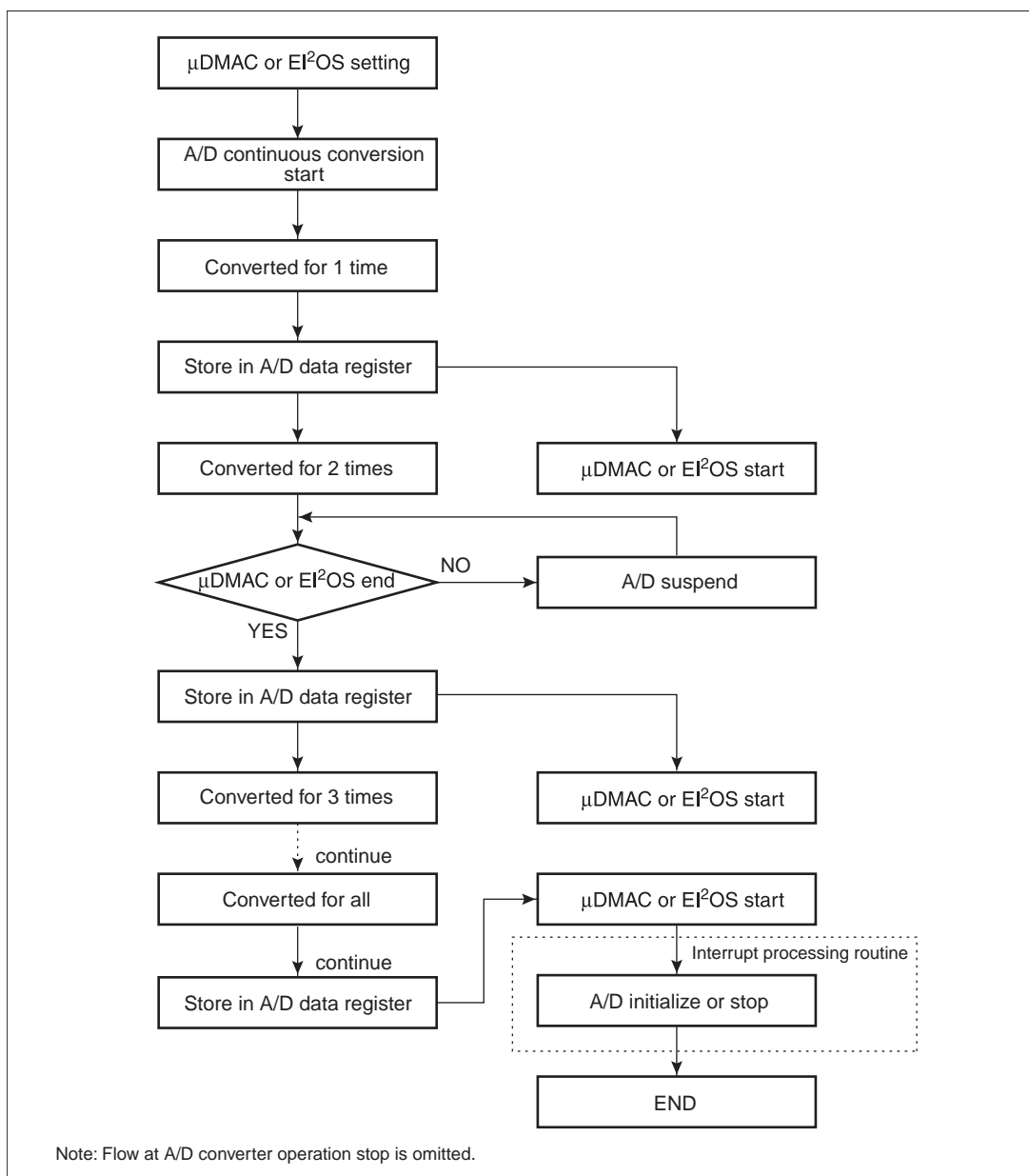
- Data protection function when  $\mu$ DMAC or EI<sup>2</sup>OS is not used

When the converted data is stored in the A/D data register (ADCR), the INT bit of the A/D control status register "H" level (ADCS1) is set to "1". While this INT bit is set to "1", the A/D conversion is in the temporally stopped state. When the INT bit is cleared after the A/D data register (ADCR) transferred to the memory, etc. in the interrupt routine, the stopped state is released.

- Data protection function when  $\mu$ DMAC or EI<sup>2</sup>OS is used

When the  $\mu$ DMAC or EI<sup>2</sup>OS is used to specify the continuous conversion, the PAUS bit of A/D control status register "H" level (ADCS1) is set to "1" in the period from the conversion termination to the completion of converted data transfer from the data register to the memory with the  $\mu$ DMAC or EI<sup>2</sup>OS, the A/D conversion operation is stopped, and the next conversion data is not stored. When the data transfer to the memory is completed, the PAUS bit is cleared to "0" and the conversion operation is restarted. Figure 19.4-5 shows the flow of the data protection function when using the  $\mu$ DMAC or EI<sup>2</sup>OS.

Figure 19.4-5 Data Protection Function Flow when  $\mu$ DMAC or EI<sup>2</sup>OS is Used



Notes:

- The converted data protection function operates only in the interrupt enabled (ADCS1: INTE=1) status.
- When the  $\mu$ DMAC or EI<sup>2</sup>OS operates, the A/D conversion is suspended, and the interrupt is disabled, the A/D conversion may operate to write new data before the old data transfer. Also when the conversion is restarted during the suspension, the old data are destroyed.
- When the conversion is restarted during the suspension, the waiting data are destroyed.

## **19.5 Precautions when Using 8/10-bit A/D Converter**

---

**Precautions when using 8/10-bit A/D converter is indicated.**

---

### **■ Precautions when Using 8/10-bit A/D Converter**

#### ● Analog input pin

The A/D input pin is in the common use with the input-output pins of port 7 and port 8 to be used by switching between the port 7 and port 8 direction register (DDR7 and DDR8) and the analog input enable register (ADER). Write "0" to the corresponding bits of DDR7 and DDR8 to input the port setting and set the analog input mode (ADEx = 1) to the ADER register to fix the port side input gate for pins to be used as analog input. When the intermediate level signal is input in the port input mode (ADEx = 0), the input leak current flows through the gate.

#### ● Notes on using in Internal timer (16-bit reload timer 1)

Set the input value of internal timer to the inactive side ("L" for the internal timer) when setting the STS1 and STS0 bits of A/D control status register "H" level (ADCS1) to start the A/D converter by the internal timer. When it is set to the active side, the operation may be started at the same time of writing on the ADCS1 register.

Set the STS1 and STS0 in the status of the "1" input to the ADTG and the "0" output from the internal timer (PPG1).

#### ● Turning-on sequence of power supply to A/D converter and analog inputs

Apply the A/D converter power supply (AVcc and AVRH) and the analog input (AN0 to AN15) always after or simultaneously turning on the digital power supply (Vcc). In addition, turn off the A/D converter power supply and the analog input after or simultaneously turn off the digital power supply (Vcc).

#### ● Power supply voltage of the A/D converter

Keep the A/D converter power supply voltage (AVcc) not to exceed the digital power supply voltage (Vcc) to prevent the latch up.

## 19.6 Example of program-1 of 8/10-bit A/D Converter (Example of Starting the EI<sup>2</sup>OS in the Single Mode)

This section shows the A/D conversion program started in single mode EI<sup>2</sup>OS

### ■ Example of EI<sup>2</sup>OS Start Program in Single Mode

#### ● Processing specification

Conversion is performed up to analog inputs AN1 to AN3.

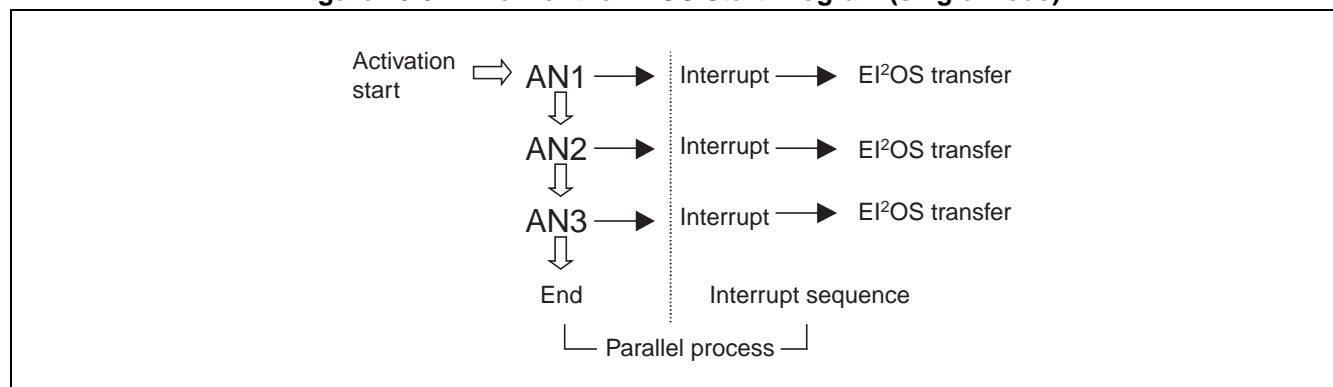
The converted data is transferred to addresses 200<sub>H</sub> to 205<sub>H</sub>.

Resolution is assumed to be ten bits.

Startup is performed by software.

Figure 19.6-1 shows flow of the EI<sup>2</sup>OS start program (single mode).

**Figure 19.6-1 Flow of the EI<sup>2</sup>OS Start Program (single Mode)**



## ● Coding example

```

BAPL    EQU    000100H    ; Buffer address pointer lower
BAPM    EQU    000101H    ; Buffer address pointer middle
BAPH    EQU    000102H    ; Buffer address pointer upper
ISCS    EQU    000103H    ; EI2OS status register
IOAL    EQU    000104H    ; I/O address register lower
IOAH    EQU    000105H    ; I/O address register upper
DCTL    EQU    000106H    ; Data counter lower
DCTH    EQU    000107H    ; Data counter upper
DDR7    EQU    000017H    ; Port 7 direction register
ADER0   EQU    00001EH    ; Analog input enable register 0
ICR12   EQU    0000BCH    ; Interrupt control register for A/DC
ADMR    EQU    000045H    ; A/D conversion channel set register
ADCS0   EQU    000040H    ; A/D Control status register
ADCS1   EQU    000041H    ;
ADCR0   EQU    000042H    ; A/D data register
ADCR1   EQU    000043H    ;

;-----Main Program-----
CODE     CSEG
START:                                       ; Stack pointer (SP), etc. shall be initialized.

        AND     CCR, #0BFH                ; Disables the interrupt.
        MOV     ICR12, #00H                ; Interrupt levels (0 strength)
        MOV     BAPL, #00H                ; Setting the converted data storage address
        MOV     BAPM, #02H                ; (uses 200H to 205H)
        MOV     BAPH, #00H                ;
        MOV     ISCS, #18H                ; Transferring the word data, Transferred address + 1,
                                           ; I/O → Transfer to the memory

        MOV     IOAL, #42H                ; As forwarding former address pointer
        MOV     IOAH, #00H                ; Setting analog data register address
        MOV     DCTL, #03H                ; Transferring the EI2OS for three times, Performing
                                           ; the conversion for the same number of times.

        MOV     DDR7, #11110001B          ; P71 to P73 are set to input.
        MOV     DCTH, #00H                ;
        MOV     ADER0, #00001110B         ; P71/AN1 to P73/AN3 are set in the analog input.
        MOV     ADMR, #013H                ; AN1 to AN3 CH are converted
        MOV     ADCSL, #000H              ; Single startup
        MOV     ADCSH, #0A2H              ; Starting the software, Starting the A/D conversion,
                                           ; Enabling the interrupt

        MOV     ILM, #07H                ; Sets ILM in PS to level 7

```

```

                                OR      CCR, #40H          ; Interruption permission
LOOP:    MOV      A, #00H          ; Infinite loop
                                MOV      A, #01H
                                BRA      LOOP

;-----Interrupt Program-----
ED_INT1:
                                MOV      I:ADCS1, #00H      ; Stopping the A/D, Disabling the interrupt and the
                                                                ; flag clearance
                                RETI          ; Returns from interrupt.
CODE      ENDS

;-----Vector Settings-----
VECT      CSEG      ABS=0FFH
                                ORG      0FF6CH          ; The vector is set in interruption #36(24H).
                                DSL      ED_INT1
                                ORG      0FFDCH          ; Reset vector setting
                                DSL      START
                                DB      00H          ; Set Single-chip mode
VECT      ENDS
                                END      START
    
```



## 19.7 Example of Program-2 of 8/10-bit A/D Converter (Example of Starting the EI<sup>2</sup>OS in the Continuous Mode)

This section shows the A/D conversion program started the EI<sup>2</sup>OS in continuous mode.

### ■ Example of EI<sup>2</sup>OS Start Program in Continuous Mode

#### ● Processing specification

Convert the analog inputs AN3 to AN5 twice to acquire two converted data for each of channels.

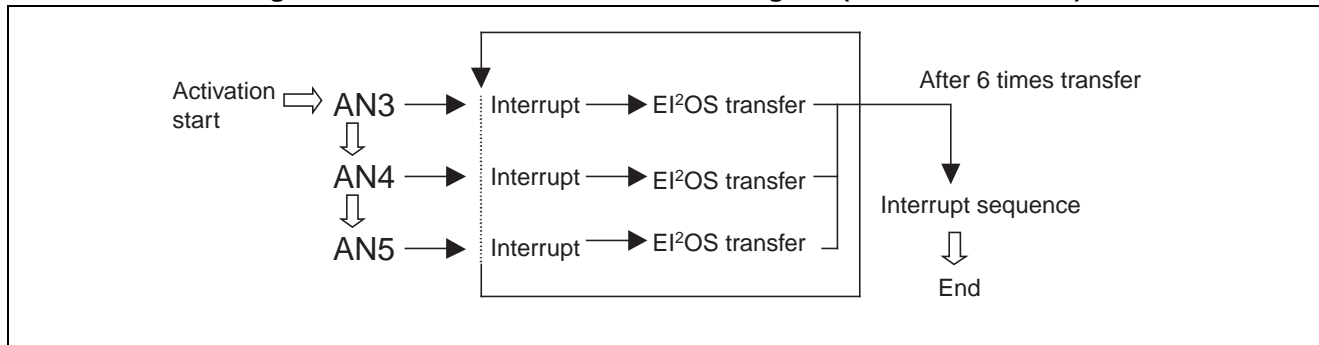
The converted data is transferred to addresses 600<sub>H</sub> to 60B<sub>H</sub>.

Resolution is assumed to be ten bits.

Start up is performed by 16-bit reload timer 0

Figure 19.7-1 shows flow of the EI<sup>2</sup>OS start program (continuous mode).

**Figure 19.7-1 Flow of the EI<sup>2</sup>OS Start Program (Continuous Mode)**



**● Coding example**

```

BAPL    EQU    000100H    ; Buffer address pointer lower
BAPM    EQU    000101H    ; Buffer address pointer middle
BAPH    EQU    000102H    ; Buffer address pointer upper
ISCS    EQU    000103H    ; EI2OS status register
IOAL    EQU    000104H    ; I/O address register lower
IOAH    EQU    000105H    ; I/O address register upper
DCTL    EQU    000106H    ; Data counter lower
DCTH    EQU    000107H    ; Data counter upper
DDR7    EQU    000017H    ; Port 7 direction register
ADER0   EQU    00001EH    ; Analog input enable register 0
ICR12   EQU    0000BCH    ; Interrupt control register for A/DC
ADMR    EQU    000045H    ; A/D conversion channel set register
ADCS0   EQU    000040H    ; A/D Control status register
ADCS1   EQU    000041H    ;
ADCR0   EQU    000042H    ; A/D data register
ADCR1   EQU    000043H    ;
TMCSR1L EQU    000062H    ; Timer control status register 0 Low
TMCSR0H EQU    000063H    ;
TMRLR0L EQU    000064H    ; Reload Register 0
TMRLR0H EQU    000065H    ;
;-----Main Program-----
CODE     CSEG
START:
; Stack pointer (SP), etc. shall be initialized.
AND      CCR,#0BFH        ; Disables the interrupt.
MOV      ICR12, #08H       ; Interrupt level 0 (the strongest), Interrupt enabled
MOV      BAPL, #00H        ; Setting the converted data storage address
MOV      BAPM, #06H        ; (uses 600H to 60BH)
MOV      BAPH, #00H        ;
MOV      ISCS, #18H        ; Transferring the word data, Transferred address +,
; I/O → Transfer to the memory
MOV      IOAL, #42H        ; As forwarding former address pointer
MOV      IOAH, #00H        ; Setting analog data register address
MOV      DCTL, #06H        ; EI2OS transfer for 6 times, Transfer for 3ch x 2
; times
MOV      DCTH, #00H
MOV      DDR7, #00000000B   ; P70 to P77 are set in the input.
MOV      ADER0, #00111000B  ; P73/AN3 to P75/AN5 are set in the analog input.
MOV      ADMR, #035H        ; AN3 to AN5 CH are converted.

```

```

MOV    ADCS0, #080H      ; Continuous mode
MOV    ADCS1, #0A8H      ; Starting the 16-bit timer, Starting the A/D
                          ; conversion, Interrupting permission

MOVW   TMRLR0L, #0320H    ; Setting the timer value 800 (320H) 66 ms
MOV    TMCSR0H, #00H      ; Setting the clock source to 83 ns, External trigger
                          ; Interdiction

MOV    TMCSR0L, #12H      ; Disabling the timer output, Disabling the interrupt,
                          ; Enabling the reload

MOV    TMCSR0L, #13H      ; 16-bit timer startup
MOV    ILM, #07H          ; Sets ILM in PS to level 7
OR     CCR, #40H          ; Interruption permission

LOOP:  MOV    A, #00H      ; Infinite loop
      MOV    A, #01H
      BRA    LOOP

;-----Interrupt Program-----
ED_INT1:
      MOV    I:ADCS1, #80H      ; The A/D does not stop, The A/D conversion process
                          ; program started by the EI2OS with the interrupt flag
                          ; clearance in the stop mode is shown.
                          ; Interdiction

      RETI                    ; Returns from interrupt.

CODE   ENDS

;-----Vector Setting-----
VECT   CSEG    ABS=0FFH
      ORG     0FF7CH            ; The vector is set in interruption #36(24H).
      DSL     ED_INT1
      ORG     0FFDCH            ; Reset vector setting
      DSL     START
      DB      00H              ; Setting Single-chip mode

VECT   ENDS
      END     START

```

## 19.8 Example of Program-3 of 8/10-bit A/D Converter (Example of Starting the EI<sup>2</sup>OS in the Stop Mode)

This section shows the A/D conversion program started the EI<sup>2</sup>OS in stop mode.

### ■ Example of EI<sup>2</sup>OS Start Program in Continuous Mode

#### ● Processing specification

Analog input AN3 is converted for 12 times in a constant period.

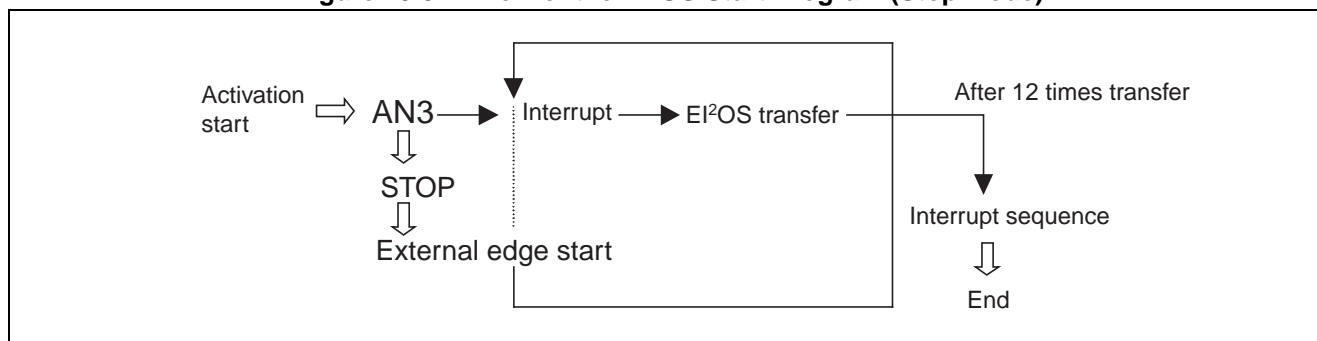
The converted data is transferred to addresses 600<sub>H</sub> to 617<sub>H</sub>.

Resolution is assumed to be ten bits.

Start up is performed by 16-bit reload timer 0.

Figure 19.8-1 shows flow of the EI<sup>2</sup>OS start program (stop mode).

Figure 19.8-1 Flow of the EI<sup>2</sup>OS Start Program (Stop Mode)



## ● Coding example

```

BAPL    EQU    000100H    ; Buffer address pointer lower
BAPM    EQU    000101H    ; Buffer address pointer middle
BAPH    EQU    000102H    ; Buffer address pointer upper
ISCS    EQU    000103H    ; EI2OS status register
IOAL    EQU    000104H    ; I/O address register lower
IOAH    EQU    000105H    ; I/O address register upper
DCTL    EQU    000106H    ; Data counter lower
DCTH    EQU    000107H    ; Data counter upper
DDR7    EQU    000017H    ; Port 7 direction register
ADER0   EQU    00001EH    ; Analog input enable register 0
ICR12   EQU    0000BCH    ; Interrupt control register for A/DC
ADMR    EQU    000045H    ; A/D conversion channel set register
ADCS0   EQU    000040H    ; A/D Control status register
ADCS1   EQU    000041H    ;
ADCR0   EQU    000042H    ; A/D data register.
ADCR1   EQU    000043H    ;
TMCSR0L EQU    000062H    ; Timer control status register 0 Low
TMCSR0H EQU    000063H    ;
TMRLR0L EQU    000064H    ; Reload Register 0
TMRLR0H EQU    000065H    ;
;-----Main Program-----
CODE     CSEG
START:
; Stack pointer (SP), etc. shall be initialized.
AND      CCR,#0BFH        ; Disables the interrupt.
MOV      ICR12, #08H       ; Interrupt levels (0 strength)
MOV      BAPL, #00H        ; Setting the converted data storage address
MOV      BAPM, #06H        ; (uses 600H to 617H).
MOV      BAPH, #00H        ;
MOV      ISCS, #19H        ; Transferring the word data, Transferred address +,
; I/O → Transfer to the memory, Termination at the
; resource request
MOV      IOAL, #42H        ; As forwarding former address pointer
MOV      IOAH, #00H        ; Setting analog data register address
MOV      DCTL, #0CH        ; EI2OS transfer for 12 times, 3ch only
MOV      DCTH, #00H        ;
MOV      DDR7, #00000000B   ; P70 to P77 are set in the input
MOV      ADER, #00001000B   ; P73/AN3 are set in the analog input
MOV      ADMR, #033H        ; AN3 CH is converted

```

```

MOV    ADCS0, #0C0H      ; Stop mode
MOV    ADCS1, #0A8H      ; Starting the 16-bit timer, Starting the A/D
                                ; conversion, Interrupting permission

MOVW   TMRLR0L, #0320H    ; Setting the timer value 800 (320H) 66 ms
MOV    TMCSR0H, #00H      ; Setting the clock source to 83 ns, External trigger
                                ; Interdiction

MOV    TMCSR0L, #12H      ; Disabling the timer output, Disabling the interrupt,
                                ; Enabling the reload

MOV    TMCSR0L, #13H      ; 16-bit timer startup
MOV    ILM, #07H          ; Sets ILM in PS to level 7
OR     CCR, #40H          ; Interruption permission

LOOP:  MOV    A, #00H      ; Infinite loop
      MOV    A, #01H
      BRA    LOOP

;-----Interrupt Program-----
ED_INT1:
      MOV    I:ADCS1, #80H      ; The A/D does not stop, The A/D conversion process
                                ; program started by the EI2OS with the interrupt flag
                                ; clearance in the stop mode is shown.
                                ; Interdiction

      RETI                    ; Returns from interrupt.

CODE   ENDS

;-----Vector Settings-----
VECT    CSEG    ABS=0FFH
      ORG    0FF6CH      ; The vector is set in interruption #36(24H).
      DSL    ED_INT1
      ORG    0FFDCH      ; Reset vector setting
      DSL    START
      DB     00H          ; Setting Single-chip mode
VECT    ENDS
      END    START

```



# **CHAPTER 20**

---

## ***EXTENDED I/O SERIAL INTERFACE***

**This chapter describes an overview of the extended I/O serial interface, the configuration and function of registers, and operations of extended I/O serial interface.**

20.1 Outline of Extended I/O Serial Interface

20.2 Register in Extended I/O Serial Interface

20.3 Operation of Extended I/O Serial Interface



## 20.1 Outline of Extended I/O Serial Interface

The extended I/O serial interface is a serial I/O interface that can transfer data through the adoption of 8-bit  $\times$  1 channel configured clock synchronization scheme. LSB first/MSB first can be selected in data transfer.

### ■ Outline of Extended I/O Serial Interface

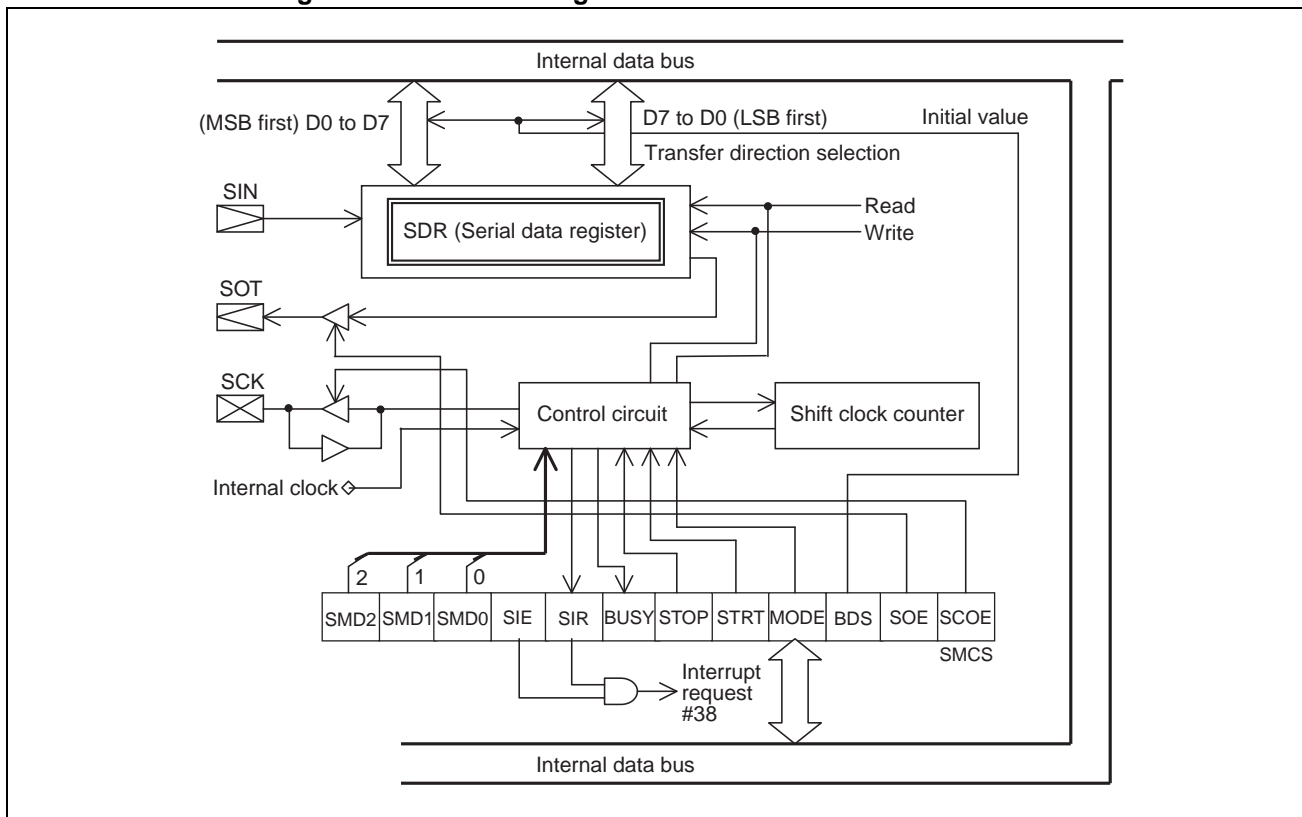
The following two operation modes are available for the extended I/O serial interface.

- Internal shift clock mode: Transfers data in sync with the internal clock.
- External shift clock mode: Transfers data in sync with the clock input through an external pin (SCK). In this mode, transfer operation performed by the CPU instruction is also available by operating the general-use port sharing an external pin (SCK).

### ■ Block Diagram in Extended I/O Serial Interface

Figure 20.1-1 shows the block diagram of extended I/O serial interface.

Figure 20.1-1 Block Diagram of Extended I/O Serial Interface



**MB90330A Series****20.2 Register in Extended I/O Serial Interface**

The configuration and functions of registers used in the extended I/O serial interface are described.

### ■ List of Register in Extended I/O Serial Interface

Figure 20.2-1 shows the list of register in extended I/O serial interface.

**Figure 20.2-1 List of Register in Extended I/O Serial Interface**

SMCS	bit	15	14	13	12	11	10	9	8	
Address : 000059H		SMD2	SMD1	SMD0	SIE	SIR	BUSY	STOP	STRT	Serial mode control status register (SMCS)
		(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R)	(R/W)	(R/W)	Initial value 00000010 <sub>B</sub>
SMCS	bit	7	6	5	4	3	2	1	0	
Address : 000058H		—	—	—	—	MODE	BDS	SOE	SCOE	Serial mode control status register (SMCS)
		(—)	(—)	(—)	(—)	(R/W)	(R/W)	(R/W)	(R/W)	Initial value XXXX0000 <sub>B</sub>
SDR	bit	7	6	5	4	3	2	1	0	
Address : 00005AH		D7	D6	D5	D4	D3	D2	D1	D0	Serial data register (SDR)
		(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	Initial value XXXXXXXX <sub>B</sub>
SDCR	bit	15	14	13	12	11	10	9	8	
Address : 00005BH		MD	—	—	—	DIV3	DIV2	DIV1	DIV0	Communication prescaler control register (SDCR)
		(R/W)	(—)	(—)	(—)	(R/W)	(R/W)	(R/W)	(R/W)	Initial value 0XXX0000 <sub>B</sub>
R/W : Readable/Writable										
— : Undefined										
X : Undefined										

## 20.2.1 Serial Mode Control Status Register (SMCS)

The configuration and functions of Serial mode control status register (SMCS) is described.

### ■ Serial Mode Control Status Register (SMCS)

Serial mode control status register (SMCS) is a register which controls the transfer operating mode of serial I/O.

Figure 20.2-2 shows the bit configuration of serial mode control status register (SMCS).

**Figure 20.2-2 Bit Configuration of Serial Mode Control Status Register (SMCS)**

SMCS	bit	15	14	13	12	11	10	9	8	Serial mode control status register
Address : 000059 <sub>H</sub>		SMD2	SMD1	SMD0	SIE	SIR	BUSY	STOP	STRT	Initial value 00000010 <sub>B</sub>
		(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R)	(R/W)	(R/W)	
SMCS	bit	7	6	5	4	3	2	1	0	Serial mode control status register
Address : 000058 <sub>H</sub>		—	—	—	—	MODE	BDS	SOE	SCOE	Initial value XXXX0000 <sub>B</sub>
		(—)	(—)	(—)	(—)	(R/W)	(R/W)	(R/W)	(R/W)	
R/W : Readable/Writable										
— : Undefined										

Function of each bit of serial mode control status register (SMCS) is described as follows:

#### [bit 15 to bit 13] SMD2,SMD1,SMD0:Serial Shift Clock Mode (shift clock selection)

The serial shift clock mode is selected.

Table 20.2-1 and Table 20.2-2 show the settings of the serial shift clock mode.

**Table 20.2-1 Serial Shift Clock Mode Selection**

SMD2	SMD1	SMD0	$\phi=24$ MHz div=8	$\phi=12$ MHz div=4	$\phi=6$ MHz div=6	Value of dividing frequency
0	0	0	1.5 MHz	1.5 MHz	500 kHz	2
0	0	1	750 kHz	750 kHz	250 kHz	4
0	1	0	188 kHz	188 kHz	62.5 kHz	16
0	1	1	93.4 kHz	93.4 kHz	31.2 kHz	32
1	0	0	46.9 kHz	46.9 kHz	15.6 kHz	64
1	0	1	External shift clock mode			
1	1	0	Setting disabled			
1	1	1	Setting disabled			

Initialized to "000<sub>B</sub>" by reset. Rewriting under forwarding is a interdiction.

Shift clock includes five alternatives of internal shift clock and one alternative of external shift clock. The external shifts. Please set neither "110<sub>B</sub>" nor "111<sub>B</sub>" to SMD2, SMD1, and SMD0.

**Table 20.2-2 Example of Settings of the Communication Prescaler (SDCR)**

Div	(Machine clock)					Machine cycle (Recommended Setting)
	MD	DIV3	DIV2	DIV1	DIV0	
1	1	0	0	0	0	3 MHz
2	1	0	0	0	1	6 MHz
4	1	0	0	1	1	12 MHz
8	1	0	1	1	1	24 MHz

Providing shift operation for each instruction is also possible by defining SCOE=0 on clock selection and operating the port sharing SCK pin.

**[bit 12] SIE: Serial I/O Interrupt Enable (enabling serial I/O interrupt)**

Serial I/O interrupt request is controlled as shown in the table below.

SIE	Operation
0	Interrupt disabled [Initial value]
1	Serial I/O interrupt enabled

- This bit is initialized to "0" at reset.
- This bit can be read and written.

**[bit 11] SIR: Serial I/O Interrupt Request (serial I/O interrupt request)**

On completion of the serial data transfer, this bit is set to "1". And when this bit turns to "1" on interrupt enabled (SIE= 1), an interrupt request to the CPU occurs. A clear condition is different according to the MODE bit.

- When MODE bit is "0", it is cleared by writing "0" to SIR bit.
- When MODE bit is "1", it is cleared by writing to or reading SDR bit.
- Regardless of values of MODE bit, it is cleared by resetting or writing "1" to STOP bit.
- It is not significant to write "1".
- On reading read/modify/write instructions, "1" is read in all cases.

**[bit 10] BUSY (forwarding status display)**

This bit indicates whether serial transfer is running or not.

BUSY	Operation
0	Stop or serial data register R/W standby [Initial value]
1	State of serial transfer

- This bit is initialized to "0" at reset.
- This bit can only be read.

**[bit 9] STOP (stop bit)**

This bit forcibly suspends serial transfer. When this bit is "1", the state changes into HALT based on STOP=1.

STOP	Operation
0	Normal Operation
1	Forwarding stop [Initial value] by STOP=1

- This bit is initialized to "0" at reset.
- This bit can be read and written.

**[bit 8] STRT: Start (start bit)**

It is a bit by which the serial transfer is started. Forwarding is begun by writing "1" in the stopped state. Lines specified as "1" are ignored, where serial transfer operation is on-going and the state is in serial shift register R/W WAIT.

- Writing "0" is not significant.
- The bit always returns "0" when read.

**[bit 7 to bit 4] Undefined bit**

The read value is irregular. Nothing is affected when it is written.

**[bit 3] MODE (serial mode selection)**

The startup condition from the stopped state is selected. However, rewriting under the operation is a interdiction.

MODE	Operation
0	Started when STRT = 1. [Initial value]
1	Started by reading or writing the serial data register.

- This bit is initialized to "0" at reset.
- This bit can be read and written.
- This bit set to "1" at  $\mu$ DMAC start.

**[bit 2] BDS: Bit Direction Selection (transfer direction selection)**

On input and output of the serial data, Select either of the following alternatives: transfer in ascending order from the least significant bit (LSB first); transfer in descending order from the most significant bit (MSB first).

BDS	Operation
0	LSB first [Initial value]
1	MSB first

- This bit is initialized to "0" at reset.
- This bit can be read and written.

---

Note:

Select transfer direction before writing data to SDR.

---

**[bit 1] SOE: Serial Out Enable (serial output permission)**

Controls the external pin (SOT) for serial I/O.

SOE	Operation
0	General-purpose port [Initial value]
1	Serial data output

- This bit is initialized to "0" at reset.
- This bit can be read and written.

**[bit 0] SCOE: SCK1 Output Enable (shift clock output enable)**

Controls output of the input/output external pins (SCK1,SCK2) for shift clock.

SCOE	Operation
0	At the transfer by general-purpose port pin and instruction [Initial value]
1	Shift clock output pin

Set this bit to "0", when you transfer data in external shift clock mode for each individual instruction.

- Initialized to "0" at reset.
- This bit can be read and written.

20.2.2 Serial Data Register (SDR)

The configuration and functions of Serial data register (SDR) are described.

Serial Data Register (SDR)

Figure 20.2-3 shows the bit configuration of the serial data register (SDR).

Figure 20.2-3 Bit Configuration of Serial Data Register (SDR)

SDR	bit	7	6	5	4	3	2	1	0	
Address : 00005A <sub>H</sub>		D7	D6	D5	D4	D3	D2	D1	D0	Serial data register
		(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	Initial value XXXXXXXX <sub>B</sub>
R/W : Readable/Writable										
X : Undefined										

The serial data register (SDR) is a serial data register to hold transfer data of the serial I/O.  
Both writing and reading to and from SDR during transfer are disabled.

## 20.2.3 Communication Prescaler Control Register (SDCR)

The configuration and functions of communication prescaler control register (SDCR) are described.

### ■ Communication Prescaler Control Register (SDCR)

Figure 20.2-4 shows the bit configuration of communication prescaler control register (SDCR).

**Figure 20.2-4 Communication Prescaler Control Register (SDCR)**

SDCR	bit	15	14	13	12	11	10	9	8	Communication prescaler control register
Address : 00005B <sub>H</sub>		MD	—	—	—	DIV3	DIV2	DIV1	DIV0	Initial value 0XXX0000 <sub>B</sub>
		(R/W)	(—)	(—)	(—)	(R/W)	(R/W)	(R/W)	(R/W)	
R/W : Readable/Writable										
— : Undefined										

The functions of each bit of the communication prescaler control register (SDCR) are described below.

#### [bit 15] MD: Machine clock divide moDe select

Bit to enable the operation of the communication prescaler

MD	Operation
0	Communication Prescaler stops.
1	Communication Prescaler is operating.

#### [bit 11 to bit 8] DIV3,DIV2,DIV1,DIV0:DIVide3 to DIVide0

These bits determine the machine clock division ratio.

DIV3 to DIV0	Rate of division
0000 <sub>B</sub>	1-frequency division
0001 <sub>B</sub>	2-frequency division
0010 <sub>B</sub>	3-frequency division
0011 <sub>B</sub>	4-frequency division
0100 <sub>B</sub>	5-frequency division
0101 <sub>B</sub>	6-frequency division
0110 <sub>B</sub>	7-frequency division
0111 <sub>B</sub>	8-frequency division

Note:

In the case of making changes to the rate of division, allow for two divisions of intervals as the duration of stabilization of the clock before communication.



## 20.3 Operation of Extended I/O Serial Interface

---

**Extended I/O interface consists of a serial mode control status register (SMCS) and a serial data register (SDR) and is used to input and output 8-bit serial data.**

**Operation of extended I/O serial interface is described.**

---

### ■ Outline of Operation of Extended I/O Serial Interface

When serial data is input or output, each of input and output operations is performed as described below.

#### ● Input of Serial data

The content of the serial data register is output in a bit serial fashion to the serial output pin (SOT pin) in synchronization with the falling edge of the serial shift clock (external clock, internal clock).

#### ● Output of Serial data

Input from the serial input pin (SIN pin) into SDR (serial data register) in a bit serial fashion in synchronization with the rising edge of the serial shift clock (external clock, internal clock).

The direction of shift (transfer from MSB or LSB) can be specified by the bit of direction specification (BDS) of SMCS (serial mode control status register).

Once the transfer completes, the state goes into STOP or into data register R/W wait (or SDR R/W WAIT) by MODE bit of the serial mode control status register (SMCS). To change state into TRANSFER from each of the states, you should do each of the following steps.

- To return from HALT, write "0" in STOP bit, and "1" in STRT bit (STOP and STRT can be set concurrently).
- To return from serial data register R/W wait (SDR R/W WAIT), read or write the data register.

## 20.3.1 Shift Clock Mode

---

Shift clock includes two types of modes; one is Internal Shift Clock Mode, the other is External Shift Clock Mode, both of which are specified by settings of SMCS. Please switch the mode with serial I/O stopped. Reading BUSY bit allows the checking of the state of HALT.

---

### ■ Internal Shift Clock Mode

Operation is driven by the internal clock, and the shift clock whose duty ratio is 50% is output from SCK pin as an output of the timing of synchronization.

Data is forwarded by one bit per a clock.

Transfer rate can be calculated using the following formula.

$$\text{transfer rate (S)} = \frac{A}{\text{Internal clock machine cycle (A)}}$$

A indicates the rate of division represented by SMD bit of SMCS.

$(\phi/\text{div})/2$ ,  $(\phi/\text{div})/2^2$ ,  $(\phi/\text{div})/2^4$ ,  $(\phi/\text{div})/2^5$ ,  $(\phi/\text{div})/2^6$

### ■ External Shift Clock Mode

In synchronization with the external shift clock input through SCK pin, 1 bit of data is transferred for each individual clock.

Allowed transfer rate varies from DC to 1/(8 machine cycles). When 1 machine cycle = 62.5 ns for example, a maximum of 2 MHz is allowable.

Transfer by instruction by instruction can be accomplished by setting as described below.

- Select the external shift clock mode, and set SCOE bit of SMCS to "0".
- Write "1" in the direction register of which the port shares SCK pin, and place the port in the output mode.

Once the settings complete as indicated above, write "1" or "0" in the data register (PDR) of the port, then the value to be delivered to SCK pin is captured as the external clock and transfer operation is accomplished. Have the shift clock start at "H".

---

Note:

Writing to SMCS.SDR during serial I/O operation is disabled.

---

## 20.3.2 Operation State of Serial I/O

---

The states of serial I/O operation includes the following 4 types of states; STOP, HALT, R/W WAIT of SDR, and TRANSFER.

---

### ■ Operation State of Serial I/O

#### ● STOP State

On RESET or in the state of writing "1" in STOP bit of SMCS, the shift counter is initialized, resulting in SIR=0.

Returning from the stop state is performed by setting STOP = 0 and STRT = 1 (both can be set at the same time). Even though STRT=1 is provided when STOP=1, transfer operation is not executed, since STOP bit is upper than STRT bit.

#### ● HALT State

When MODE bit is "0", termination of transfer provides SMCS with BUSY=0 and SIR=1, resulting in the initialization of the counter to go into HALT state. To return from the HALT state, set STRT to "1", then transfer operation restarts.

#### ● Serial data register R/W standby

When MODE bit of SMCS is "1", termination of serial transfer provides SMCS with BUSY=0 and SIR=1, resulting in the serial data register to go into R/W WAIT state. If the interrupt enable register is enabled, this block issues interrupt signals.

To return from the R/W WAIT state, when the serial data register is read or written, then BUSY is set to "1", which allows to transfer operation to restart.

#### ● Transfer State

It is a state to do the serial transfer by BUSY = 1. MODE bit triggers the transition to the state of HALT and R/W WAIT respectively.

Figure 20.3-1 shows the transition diagram of each state, and Figure 20.3-2 shows the conceptual diagram of read/write of the serial data register.

Figure 20.3-1 Transition Diagram of Operation in Extended I/O Serial Interface

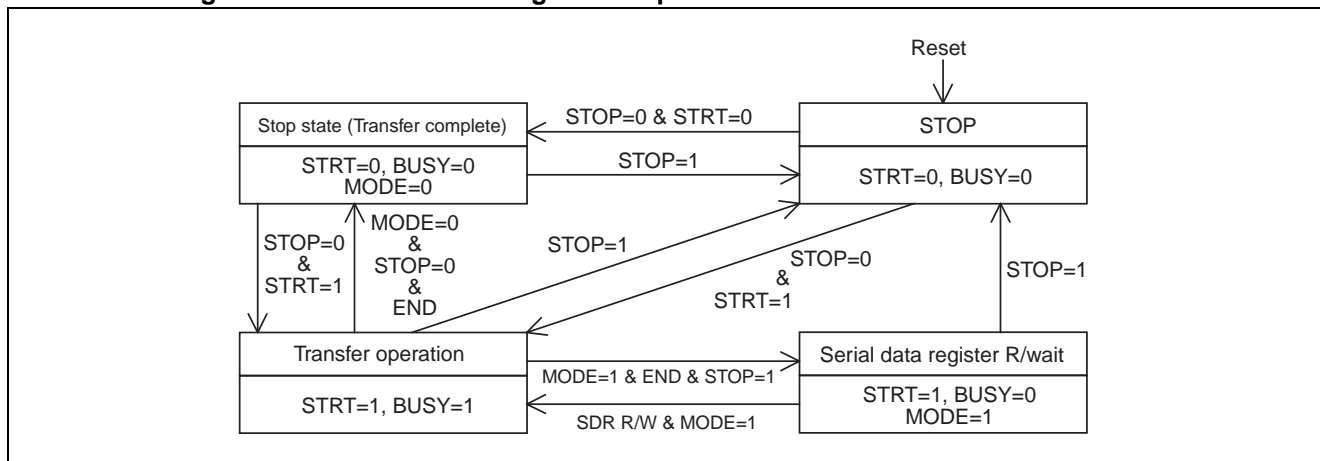
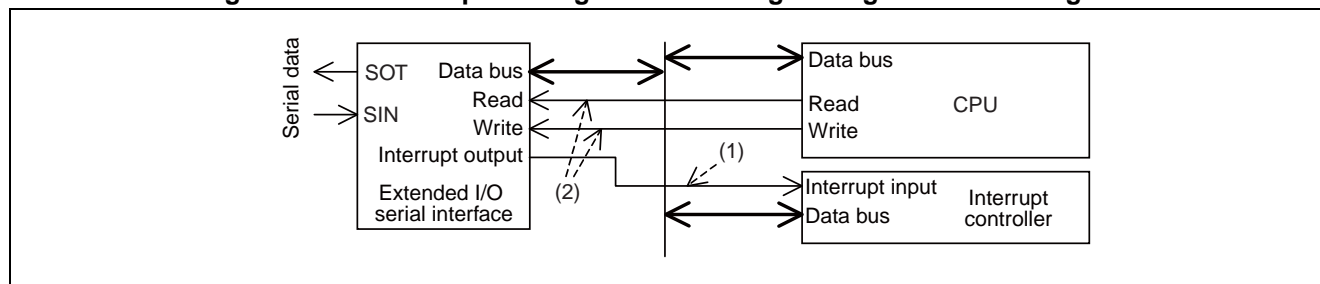


Figure 20.3-2 Conceptual Diagram of Reading/writing Serial Data Register



- (1) When MODE=1, transfer is terminated by the shift clock counter. This allows SIR=1 to go into read/write state. If the SIE bit is "1", the interruption signal is generated. However, when SIE is inactive, or when writing "1" in STOP causes the suspend of transfer, interrupt signals are not generated.
- (2) Once the serial data register is read or written, interrupt requests are cleared, and serial transfer starts.

### 20.3.3 Start/stop Timing of Shift Operation and Timing of I/O

Start/stop timing of shift operation and timing of I/O is described.

#### ■ Start/stop Timing of Shift Operation and Timing of I/O

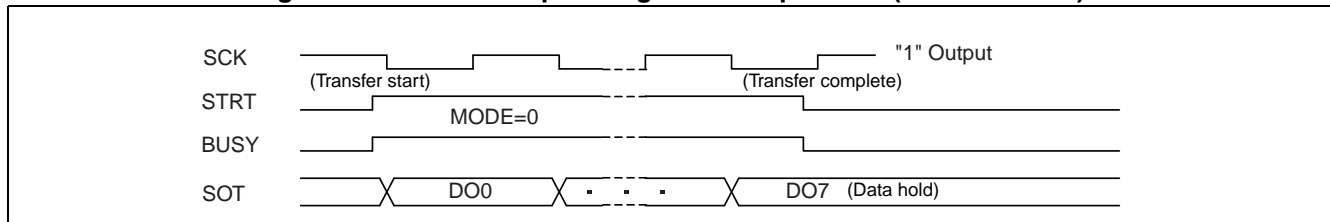
- Start  
STOP bit of Start SMCS is set to "0", while STRT bit to "1".
- Stop  
One halt is triggered from the termination of transfer; the other from STOP=1.  
  - Halt from STOP=1: Halts staying with SIR=0, regardless of MODE bit.
  - Halt from transfer termination: Halts with SIR=1, regardless of MODE bit.

Regardless of MODE bit when BUSY bit is in the state of serial transfer, it is set to "1", and when in HALT or R/W WAIT state, it is set to "0". Please read this bit to confirm forwarding.

The following chart shows the operation for each mode and the timing of halt operation. DO7 to DO0 in figure shows output data.

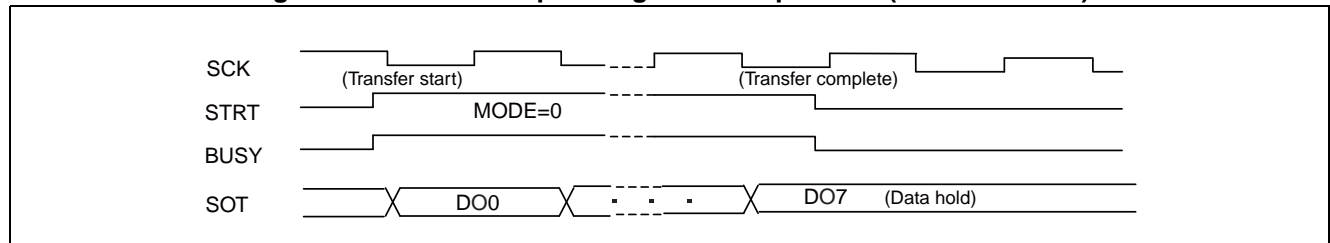
#### ● Internal shift clock mode (LSB first)

Figure 20.3-3 Start/stop Timing of Shift Operation (Internal Clock)



#### ● External shift clock mode (LSB first)

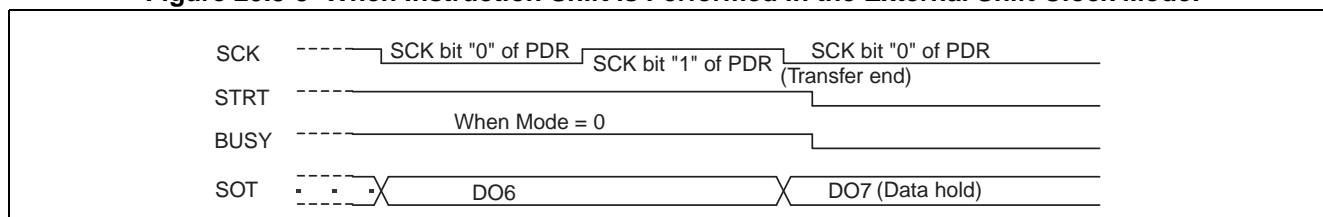
Figure 20.3-4 Start/stop Timing of Shift Operation (External Clock)



#### ● For instruction shift in external shift clock mode (LSB first)

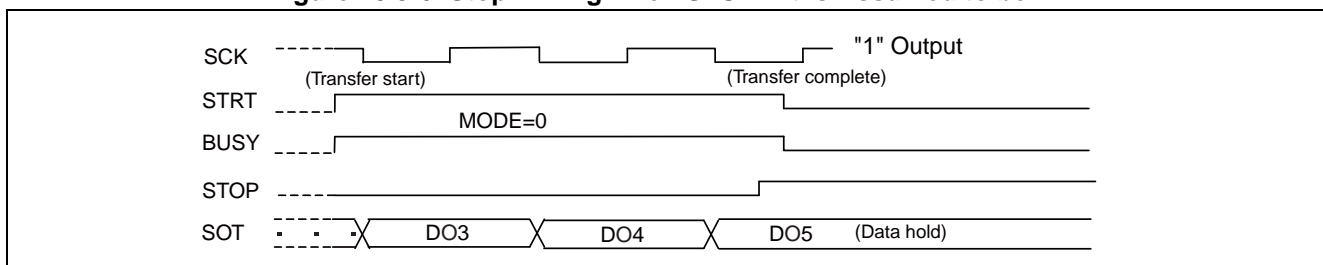
In instruction shift, when "1" is written to the bit corresponding to SCK of PDR, "H" is output, and when "0" is written, "L" is output (where external shift clock mode is selected, and SCOE=0).

Figure 20.3-5 When Instruction Shift is Performed in the External Shift Clock Mode.



- Stop by STOP=1 (LSB first, At internal clock)

Figure 20.3-6 Stop Timing when STOP Bit is Assumed to be "1"

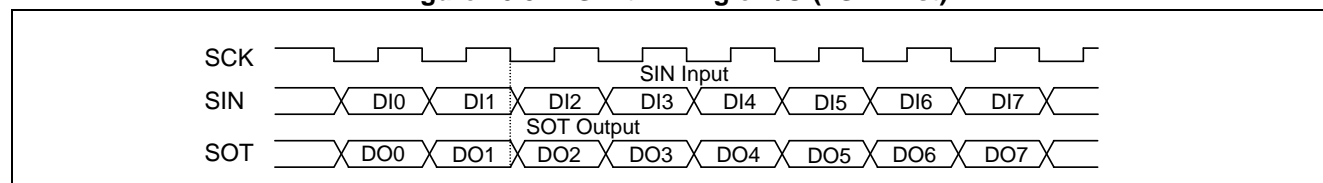


## ■ Operation during Transfer Serial Data

During transferring serial data, data from the serial output pin (SOT) is output on the falling edge of the shift clock, data of the serial input pin (SIN) is input on the rising edge of the shift clock.

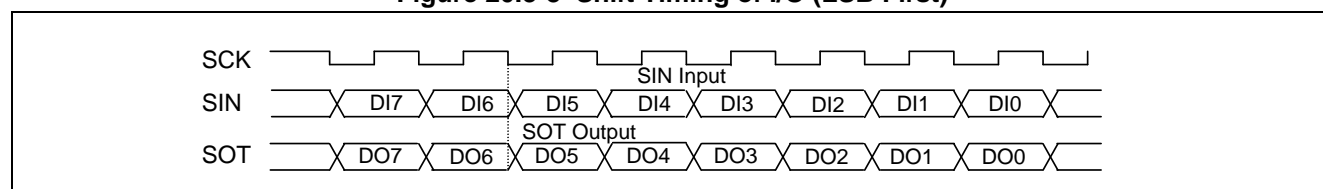
- LSB first (when BDS bit is set to "0")

Figure 20.3-7 Shift Timing of I/O (LSB First)



- MSB first (when BDS bit is set to "1")

Figure 20.3-8 Shift Timing of I/O (MSB First)



20.3.4 Interrupt Function

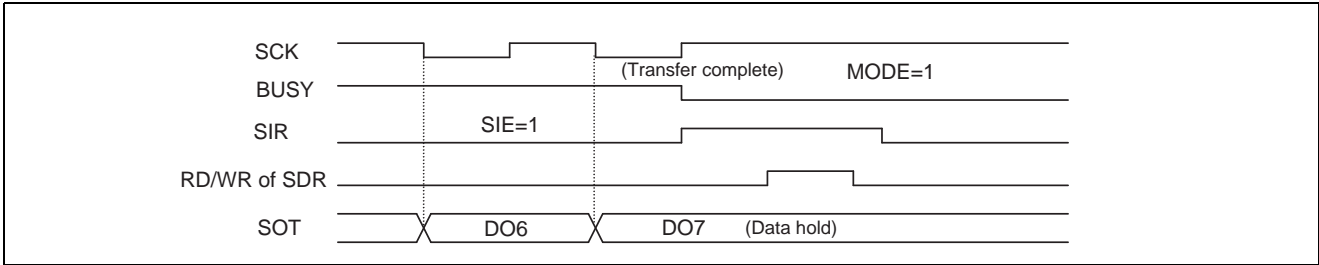
Extended I/O serial interface can generate the interrupt request to the CPU.

■ Interruption Function of Extended I/O Serial Interface

Upon completion of data transfer, SIR bit indicating an interrupt flag is set, and when SIE bit of the SMCS enabling interrupts is "1", the interrupt request is output to the CPU.

Figure 20.3-9 shows the output timing of interrupt signal.

Figure 20.3-9 Output Timing of Interruption Signal



# **CHAPTER 21**

---

## **UART**

**This chapter explains the function and operation of the UART.**

- 21.1 Overview of UART
- 21.2 UART Block Diagram
- 21.3 UART Pins
- 21.4 Register of UART
- 21.5 UART Interrupt
- 21.6 UART Baud Rate
- 21.7 Explanation of Operation of UART
- 21.8 Notes on Using UART
- 21.9 Example of UART Programming



## 21.1 Overview of UART

UART is a general purpose serial data communication interface for synchronous or asynchronous (start-stop synchronization) communications with external devices. Not only the typical function of bi-directional communication (normal mode), but also the function of master/slave communication (multiprocessor mode: only supported master side) are supported.

### ■ UART Function

#### ● UART function

UART, or a general serial data communication interface that sends and receives serial data to and from other CPU and peripherals, has the functions listed in Table 21.1-1.

Table 21.1-1 UART Function

	Functions
Data buffer	Full-duplicate double-buffer
Transfer mode	<ul style="list-style-type: none"><li>• Clock synchronous (No start/stop bit)</li><li>• Clock asynchronous (start-stop synchronization to clock)</li></ul>
Baud rate	<ul style="list-style-type: none"><li>• Dedicate baud-rate generator</li><li>• External clock input enabled.</li></ul>
Data length	<ul style="list-style-type: none"><li>• 8 bits or 7 bits (in the asynchronous normal mode only)</li><li>• 1 to 8 bit (s) (synchronous mode only)</li></ul>
Signal type	NRZ (Non Return to Zero) type
Detection of receive error	<ul style="list-style-type: none"><li>• Framing error</li><li>• Overrun error</li><li>• Parity error (Not supported in operation mode 1)</li></ul>
Interrupt request	<ul style="list-style-type: none"><li>• Receive interrupt (receive, detection of receive error)</li><li>• Transmission interrupt (transmission complete)</li><li>• Extended intelligent I/O service for both sending and receiving</li></ul> Support of (EI <sup>2</sup> OS) $\mu$ DMA
Master/slave type communication function (multi processor mode)	Capable of 1 (master) to n (slaves) communication (available just as master)

Note:

At clock synchronous transfer, data is transferred alone with neither start nor stop bit added.

**Table 21.1-2 UART Operation Modes**

Operating mode		Data length		Synchronous type	Length of Stop Bit
		Without Parity	With Parity		
0	Normal mode	7 bits or 8 bits		Asynchronous	1 bit or 2 bits <sup>*2</sup>
1	Multiprocessor mode	8 bits + 1 <sup>*1</sup>	-	Asynchronous	
2	Normal mode	1 to 8 bits	-	Synchronous	None

-: Unavailable

\*1: "+1" indicates the address/data select bit (A/D) used for communication control.

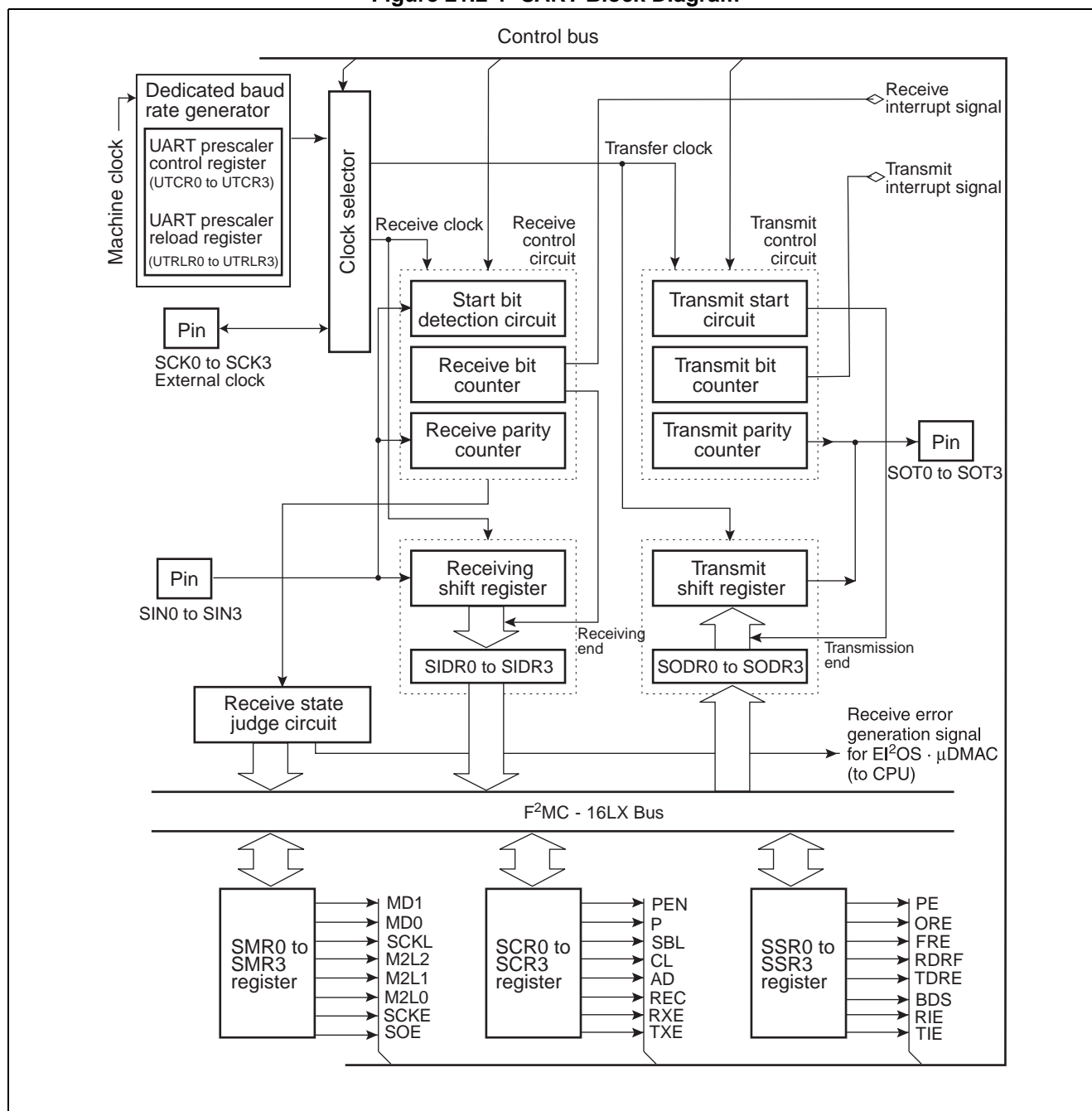
\*2: During reception, only one bit is detected as the stop bit.

## 21.2 UART Block Diagram

UART is composed of the following block.

### ■ UART Block Diagram

Figure 21.2-1 UART Block Diagram



- **Clock selector**

Dedicated baud rate generator, selecting the send and receive clock from external input clocks.

- **Reception Control Circuit**

The reception control circuit is configured with the reception bit counter, start bit detecting circuit, and reception parity counter. The receive bit counter counts receiving data. Once this counter completes receiving a piece of data based on the specified data length, then a receiving interrupt request is generated. The start bit detection circuit is a circuit that detects the start bit from serial input signals. When detecting the start bit, this circuit writes the data in the serial input data registers 0 to 3 (SIDR0 to SIDR3) with shifting based on specified transfer rates. The receive parity counter calculates parity in received data.

- **Transmission Control Circuit**

The transmission control circuit is configured with the transmission bit counter, transmission start circuit, and transmission parity counter. The send bit counter counts sending data. Once this counter completes sending a piece of data based on the specified data length, then a sending interrupt request is generated. The sending start circuit starts sending operation by writing to the serial output data registers 0 to 3 (SODR0 to SODR3). The transmit parity counter generates a parity bit for data to be transmitted if the data is parity-checked.

- **Receive shift register**

This circuit captures the receiving data, shifting bit by bit, that is input from SIN0 to SIN3 pins. On completion of reception, this circuit transfers the receiving data to the serial input data registers 0 to 3 (SIDR0 to SIDR3).

- **Transmit shift register**

The data that is written to the serial output data registers 0 to 3 (SODR0 to SODR3) is transferred to the sending shift register, which outputs it to the SOT pin with shifting bit by bit.

- **Serial mode register 0 to 3 (SMR0 to SMR3)**

Specifies selecting operation modes, enabling/disabling output of serial data to the pin, setting to enable/disable output of the clock to the pin, setting the arbitrary number of 1 bit to 8 bits to be transferred in the synchronous communication mode, and setting levels of serial clock output (fixed on "L", fixed on "H") when not operating.

- **Serial control register 0 to 3 (SCR0 to SCR3)**

Specifies setting the presence or the absence of parity, selection of parity, setting stop bit length, setting data length, selecting frame data format in the mode 1, clearance of flags, and enabling/disabling of sending and receiving.

- **Serial Status Register 0 to 3 (SSR0 to SSR3)**

Checking sending and receiving, or the states of errors, and specifies enabling/disabling sending and receiving interrupt requests.

- Serial input data register 0 to 3 (SIDR0 to SIDR3)

The register retains the receive data. The serial input is converted and then stored in this register.

- Serial output data register 0 to 3 (SODR0 to SODR3)

The register sets the transmit data. Data written to this register is serially converted to be output.

- UART prescaler control register 0 to 3 (UTCR0 to UTCR3)

Specifies start-up/halt of the communication prescaler, forced reset of UART, selecting of clock sources, and the rate of division of the machine clock.

- UART prescaler reload register 0 to 3 (UTRLR0 to UTCR3)

Specifies the division rate of the machine clock.

## 21.3 UART Pins

The pin of UART is shown.

### ■ UART Pins

The UART pins also serve as general-purpose ports. Table 21.3-1 shows the functions of pins, input-output formats, and settings in using UART, etc.

**Table 21.3-1 UART Pins**

Pin Name	Pin Function	I/O Type	Pull-up Selection	Stand-by control	Setting for the use of the pin
P42/A10/SIN0 P45/A13/SIN1 P90/SIN2 P93/SIN3	Port 4,9 I/O / serial data input	CMOS output/ CMOS hysteresis input	None	Yes	Set to the input port (DDR4: bit2=0) (DDR4: bit5=0) (DDR9: bit0=0) (DDR9: bit3=0)
P43/A11/SOT0 P46/A14/SOT1 P91/SOT2 P94/SOT3	Port 4,9 I/O / serial data output				Set to output enable. (SMR0 to SMR3: SOE=1)
P44/A12/SCK0 P47/A15/SCK1 P92/SCK2 P95/SCK3	Port 4,9 I/O / serial clock input/output				Set to the input port at clock input. (DDR4: bit4=0) (DDR4: bit7=0) (DDR9: bit2=0) (DDR9: bit5=0)
					Set to output enable at clock output. (SMR0 to SMR3: SCKE=1)

## 21.4 Register of UART

The list of the register of UART is shown.

### ■ List of UART Register

Figure 21.4-1 List of UART Register

Address	bit15 .....	bit8	bit7 .....	bit0
ch.0 : 000021H, 20H ch.1 : 000027H, 26H ch.2 : 00002DH, 2CH ch.3 : 000033H, 32H	SCR0 to SCR3 (Serial control register 0 to 3)		SMR0 to SMR3 (Serial mode register 0 to 3)	
ch.0 : 000023H, 22H ch.1 : 000029H, 28H ch.2 : 00002FH, 2EH ch.3 : 000035H, 34H	SSR0 to SSR3 (Serial status register 0 to 3)		SIDR0 to SIDR3 · SODR0 to SODR3 (Serial input · Output data register 0 to 3)	
ch.0 : 000025H, 24H ch.1 : 00002BH, 2AH ch.2 : 000031H, 30H ch.3 : 000037H, 36H	UTCR0 to UTCR3 (UART prescaler control register 0 to 3)		UTRLR0 to UTRLR3 (UART prescaler reload register 0 to 3)	

Note:

When you set the communication mode, do this while operation is halted. Any data sent or received while setting modes is not guaranteed.

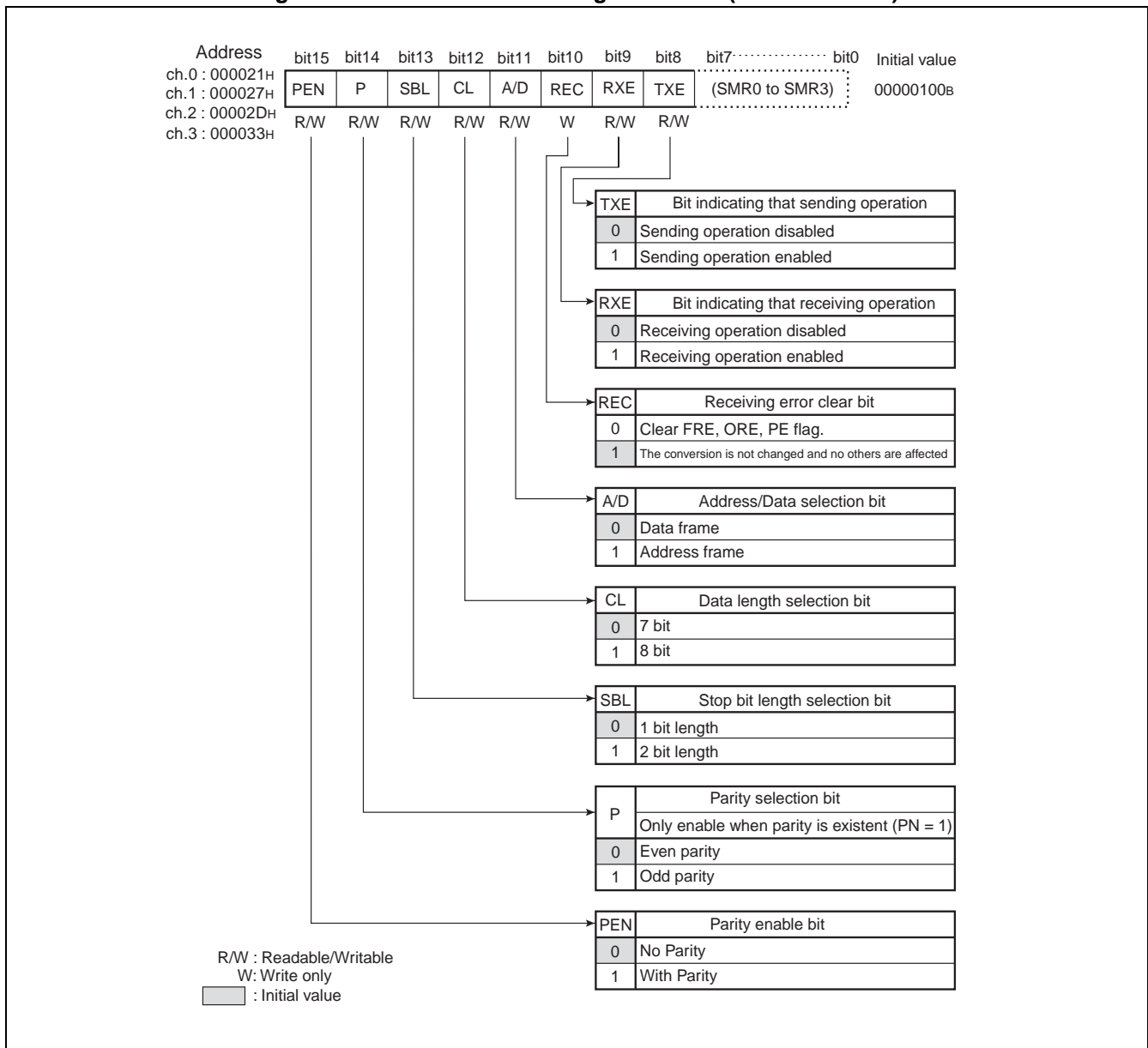
## MB90330A Series

### 21.4.1 Serial Control Register 0 to 3 (SCR0 to SCR3)

Serial control registers 0 to 3 (SCR0 to SCR3) are responsible for setting parity, selecting the stop bit length and data length, selecting the frame data format in mode 1, clearing receiving error flags, and enabling/disabling send and receive operations.

#### Serial Control Register 0 to 3 (SCR0 to SCR3)

Figure 21.4-2 Serial Control Register 0 to 3 (SCR0 to SCR3)





**Table 21.4-1 Functional Description of Each Bit in Serial Control Register 0 to 3 (SCR0 to SCR3)**

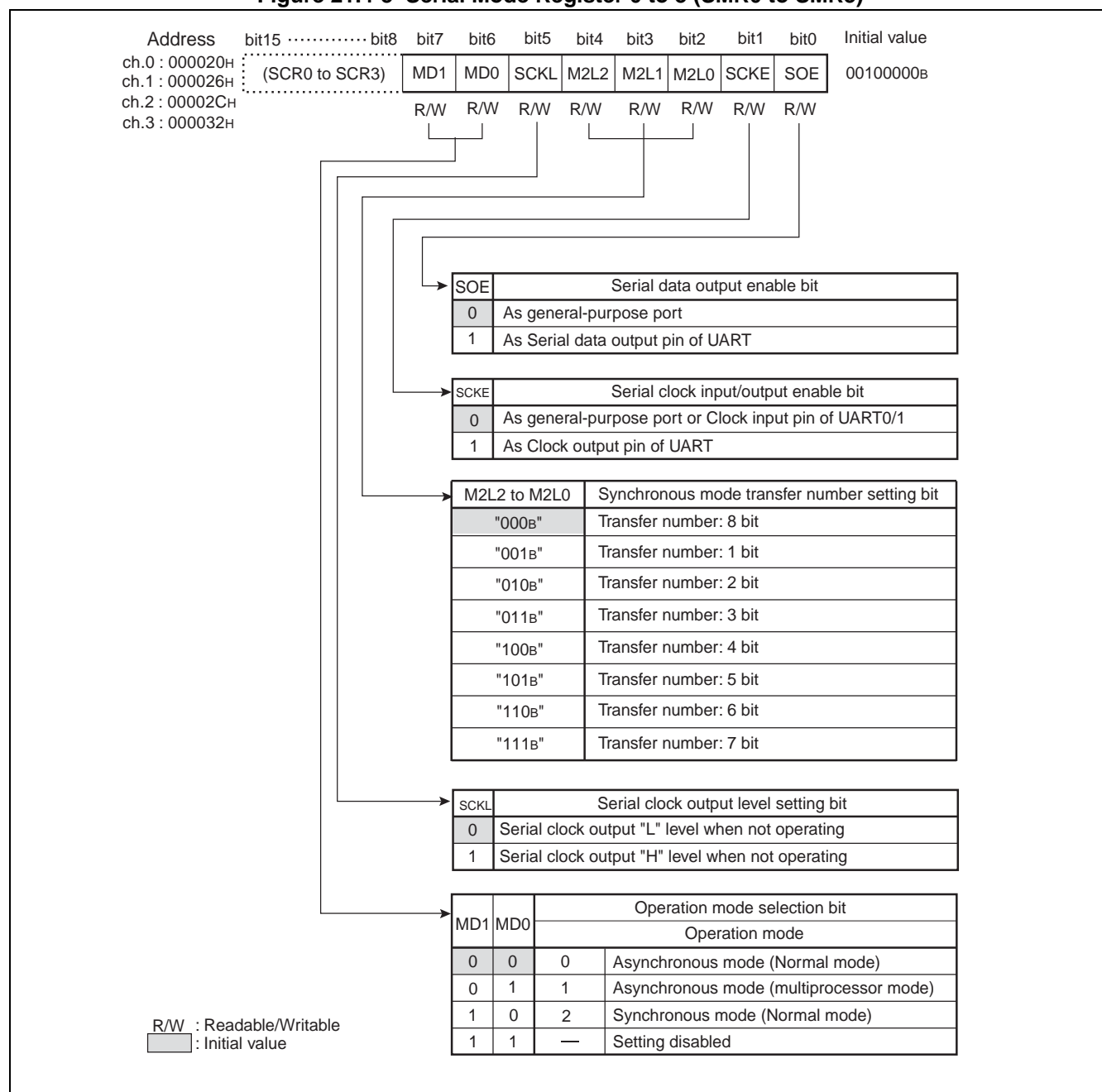
Bit name		Functions
bit 15	PEN: Parity enable bit	Specify whether to add (at sending) or detect (at receiving) a parity bit. Note: In operation mode1 and operation mode 2, parity bit cannot be appended. Always set this bit to "0".
bit 14	P: Parity selection bit	Select either odd or even parity when the use of the parity bit has been selected (SCR0 to SCR3: PEN = 1).
bit 13	SBL: Stop bit length selection bit	Set the length of the stop bits (transmit data's frame end mark) in operation mode 0 and operation mode 1 (asynchronous). Note: During receiving data, only the first bit of the stop bit is detected in all cases.
bit 12	CL: Data length selection bit	Specify the data length of data to be transmitted and received. It is only operation mode 0 to be able to select seven bits. In operation mode 1 and operation mode 2, be sure to set a data length of 8 bits.
bit 11	A/D: Address/Data selection bit	<ul style="list-style-type: none"> <li>• In operation mode 1, set the data format of frames to be transmitted/received.</li> <li>• When the bit is set to "0": The frame format is set to data frame.</li> <li>• When the bit is set to "1": The frame format is set to address data frame.</li> </ul>
bit 10	REC: Receive error flag clear bit	<ul style="list-style-type: none"> <li>• Clear the reception error flags (SSR0 to SSR3: FRE, ORE, PE) in the serial status register to "0".</li> <li>• When the bit is set to "0": The FRE, ORE, and PE flags are cleared.</li> <li>• When the bit is set to "1": No effect.</li> <li>• When read: "1" is always read.</li> </ul> Note: When the receiving interrupt is set to be enabled (SSR0 to SSR3:RIE=1), REC bit may be set to "0" as long as each of FRE, ORE, PE flag is set to "1".
bit 9	RXE: Reception operation enable bit	<ul style="list-style-type: none"> <li>• The bit enables or disables the UART for reception.</li> <li>• When the bit is set to "0": Reception is disabled.</li> <li>• When the bit is set to "1": Reception is enabled.</li> </ul> Note: During receiving data, if the receiving operation is set to be disabled, the receiving operation is halted after in-coming data is stored in the serial input data register.
bit 8	TXE: Transmission operation enable bit	<ul style="list-style-type: none"> <li>• The bit enables or disables the UART for transmission.</li> <li>• When the bit is set to "0": Transmission is disabled.</li> <li>• When the bit is set to "1": Transmission is enabled.</li> </ul> Note: If transfer operation is set to be disabled during data transfer, the transfer operation is stopped on completion of data transfer of the serial output data register.

## 21.4.2 Serial Mode Register 0 to 3 (SMR0 to SMR3)

The serial mode registers 0 to 3 (SMR0 to SMR3) are responsible for selecting operation modes, setting pins related to serial data and clock to be enabled or disabled, and setting how many bit to transfer ranging from 1 to 8 bits, setting the serial clock output level in the inactive operation (fixed to "L" and "H").

## ■ Serial Mode Register 0 to 3 (SMR0 to SMR3)

Figure 21.4-3 Serial Mode Register 0 to 3 (SMR0 to SMR3)



**Table 21.4-2 Functional Description of Each Bit in Serial Mode Register 0 to 3 (SMR0 to SMR3)**

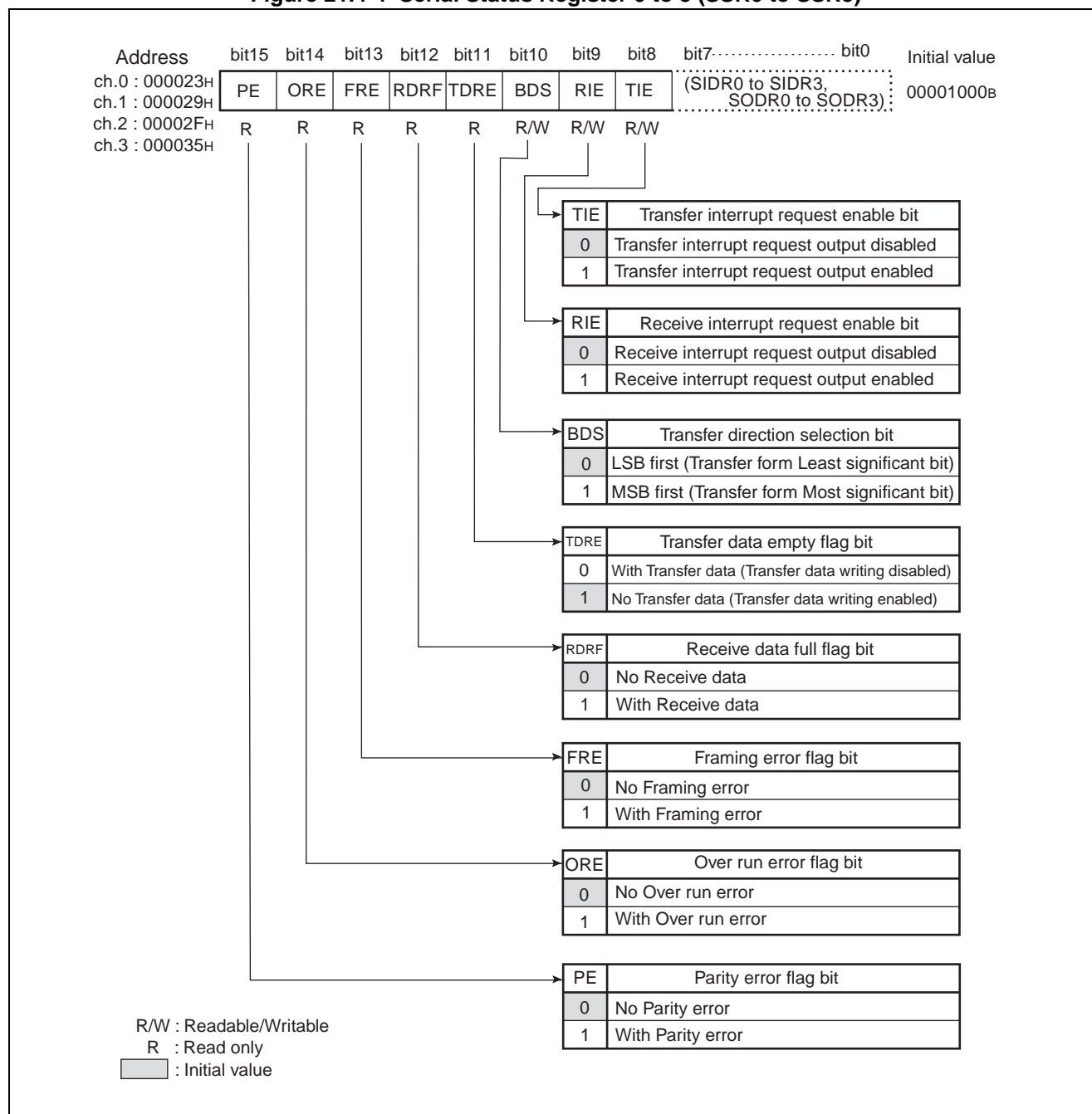
Bit name		Functions
bit 7, bit 6	MD1, MD0: Operating mode selection bits	<p>Set Operating mode</p> <p>Notes:</p> <ul style="list-style-type: none"> <li>- In operation mode 1, the device can be used only as the master for master/slave communication. In operation mode 1, the address/data bit as bit 9 cannot be received, so the device cannot be used as the slave.</li> <li>- In operation mode 1, set the parity addition enable bit to no parity (SCR0 to SCR3: PEN = 0) as the parity check function cannot be used.</li> </ul>
bit 5	SCKL: Serial clock output level set bit	<ul style="list-style-type: none"> <li>• When communication is performed in the clock synchronous mode, serial clock output level at non-operating is set.</li> <li>• When set to "0": Output of serial clock is "L" level.</li> <li>• When set to "1": Output of serial clock is "H" level.</li> </ul>
bit 4 to bit 2	M2L2, M2L1, M2L0: Synchronous mode transfer numerical set bits	<ul style="list-style-type: none"> <li>• Specify the number of bits transferred in the synchronous communication mode.</li> <li>• It is invalid at the asynchronous communication mode.</li> <li>• When set to "000<sub>B</sub>": Specify as 8-bit transmission.</li> <li>• When set to "001<sub>B</sub>": Specify as 1-bit transmission.</li> <li>• When set to "010<sub>B</sub>": Specify as 2-bit transmission.</li> <li>• When set to "011<sub>B</sub>": Specify as 3-bit transmission.</li> <li>• When set to "100<sub>B</sub>": Specify as 4-bit transmission.</li> <li>• When set to "101<sub>B</sub>": Specify as 5-bit transmission.</li> <li>• When set to "110<sub>B</sub>": Specify as 6-bit transmission.</li> <li>• When set to "111<sub>B</sub>": Specify as 7-bit transmission.</li> </ul>
bit 1	SCKE: Serial clock input/output enable bit	<ul style="list-style-type: none"> <li>• Switch between input and output of the serial clock.</li> <li>• When the bit is set to "0": The pin is set as a general-purpose I/O port or serial clock input pin.</li> <li>• When the bit is set to "1": The pin is set as a serial clock output pin.</li> </ul> <p>Notes:</p> <ul style="list-style-type: none"> <li>- When using the SCK0 to SCK3 pin as the serial clock input (SCK=0), set the pin to the input port using the port direction register (DDR). Also, use the clock input source select bit to select the external clock (UTCR0 to UTCR3: CKS=1).</li> <li>- When using this as serial clock output, set the clock input source selection bit to the dedicated baud rate generator (UTCR0 to UTCR3: CKS=0).</li> <li>- When SCK pins (0 to 3) are set to serial clock output, they function as serial clock output pins, regardless of the settings of the generic input-output port.</li> </ul>
bit 0	SOE: Serial data output enable bit	<ul style="list-style-type: none"> <li>• Enable or disable output of serial data.</li> <li>• When the bit is set to "0": The pin is set as a general-purpose I/O port.</li> <li>• When the bit is set to "1": The pin is set as a serial data output pin.</li> <li>• When SOT pins (0 to 3) are set to serial data output, they function as serial clock output pins, regardless of the settings of the generic input-output port.</li> </ul>

## 21.4.3 Serial Status Register 0 to 3 (SSR0 to SSR3)

The serial status registers 0 to 3 (SSR0 to SSR3) are responsible for checking sending and receiving and the states of errors, and setting interrupts to be enabled or disabled.

## ■ Serial Status Register 0 to 3 (SSR0 to SSR3)

Figure 21.4-4 Serial Status Register 0 to 3 (SSR0 to SSR3)



**Table 21.4-3 Description of Each Bit of the Serial Status Registers 0 to 3 (SSR0 to SSR3) (1 / 2)**

Bit name		Functions
bit 15	PE: Parity error flag bit	<ul style="list-style-type: none"> <li>• Detect a parity error of receiving data.</li> <li>• This bit is set to "1" when a parity error occurs.</li> <li>• This is cleared by writing "0" in the receiving error clear bit (SCR0 to SCR3: REC).</li> <li>• When receiving interrupts are enabled (SSR0 to SSR3: RIE=1), a receiving interrupt request is generated if a parity error occurs</li> <li>• When the parity error flag bit is set (SSR0 to SSR3: PE = 1), data in serial input data register is invalid.</li> </ul>
bit 14	ORE: Overrun error flag bit	<ul style="list-style-type: none"> <li>• Detect an overrun error in receiving.</li> <li>• This bit is set to "1" when an overrun error occurs.</li> <li>• This is cleared by writing "0" in the receiving error flag clear bit (SCR0 to SCR3: REC).</li> <li>• When receiving interrupts are enabled (SSR0 to SSR3: RIE=1), a receiving interrupt request is generated if a overrun error occurs.</li> <li>• When the overrun error flag bit is set (SSR0 to SSR3: ORE = 1), data in serial input data register is invalid.</li> </ul>
bit 13	FRE: Framing error flag bit	<ul style="list-style-type: none"> <li>• Detect a framing error of receive data.</li> <li>• This bit is set to "1" when a framing error occurs.</li> <li>• This is cleared by writing "0" in the receiving error clear bit (SCR0 to SCR3: REC).</li> <li>• When receiving interrupts are enabled (SSR0 to SSR3: RIE=1), a receiving interrupt request is generated if a framing error occurs.</li> <li>• When the framing error flag bit is set (SSR0 to SSR3: FRE = 1), data in serial input data register is invalid.</li> </ul>
bit 12	RDRF: Receive data full flag bit	<ul style="list-style-type: none"> <li>• Show the status of the serial input data register.</li> <li>• When received data is loaded to serial input data register 0 to 3 (SIDR0 to SIDR3), "1" is set.</li> <li>• This bit is cleared to "0" when data is read from the serial input data register 0 to 3 (SIDR0 to SIDR3).</li> <li>• When receiving interrupts are enabled (SSR0 to SSR3: RIE=1), a receiving interrupt request are generated if receiving data is loaded to the serial input data registers (SIDR0 to SIDR3).</li> </ul>
bit 11	TDRE: Transmission data empty flag bit	<ul style="list-style-type: none"> <li>• Show the status of the serial output data register 0 to 3 (SODR0 to SODR3).</li> <li>• The bit is cleared to "0" by writing sending data to the serial output data registers 0 to 3 (SODR0 to SODR3).</li> <li>• This bit is set to "1" when data is loaded to the send shift register and transmission starts.</li> <li>• When sending interrupts are enabled (SSR0 to SSR3: TIE=1) and if the data that has written to the serial output data registers 0 to 3 (SODR0 to SODR3) is transferred to the sending shift register, then a sending interrupt request is generated.</li> </ul>
bit 10	BDS: Transfer direction selection bit	<ul style="list-style-type: none"> <li>• This bit sets the direction of serial data transfer.</li> <li>• When set to "0": Serial data is transferred from the LSB bit first (LSB first).</li> <li>• When set to "1": Serial data is transferred from the MSB bit first (MSB first).</li> </ul> <p>Note:</p> <p>If BDS bit is rewritten after the completion of the access to the register, the rewritten data will be invalid, since in reading to the serial input data register and in writing to the serial output data register, the LSB data and the MSB data are turned upside down.</p>

**Table 21.4-3 Description of Each Bit of the Serial Status Registers 0 to 3 (SSR0 to SSR3) (2 / 2)**

Bit name		Functions
bit 9	RIE: Reception interrupt request enable bit	<ul style="list-style-type: none"> <li>• Enable or disable receive data.</li> <li>• When set to "1": If receiving data is loaded to the serial input data registers 0 to 3 (SSR0 to SSR3: RDRF=1). Or if an receiving error occurs (SSR0 to SSR3: PE=1, or ORE=1, or FRE=1), then a receiving interrupt request is generated.</li> </ul>
bit 8	TIE: Transmission interrupt request enable bit	<ul style="list-style-type: none"> <li>• Enable or disable send interrupt.</li> <li>• When set to "1": If the data written to the serial output data registers 0 to 3 is sent to the sending shift register (SSR0 to SSR3: TDRE=1), then a sending interrupt request is generated.</li> </ul>

## 21.4.4 Serial Input Data Register 0 to 3 (SIDR0 to SIDR3) and Serial Output Data Register 0 to 3 (SODR0 to SODR3)

Serial input data and serial output data registers are located in the same address. They function as a serial input data register in reading, while in writing as a serial output data register.

### ■ Serial Input Data Register 0 to 3 (SIDR0 to SIDR3)

Figure 21.4-5 shows the bit configuration of the serial input data register.

**Figure 21.4-5 Serial Input Data Register 0 to 3 (SIDR0 to SIDR3)**

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch.0 : 000022H	.....								XXXXXXXXB
ch.1 : 000028H	.....								
ch.2 : 00002EH	.....								
ch.3 : 000034H	.....								
	D7	D6	D5	D4	D3	D2	D1	D0	
	R	R	R	R	R	R	R	R	

R : Read only  
X : Undefined

Serial input data register 0 to 3 (SIDR0 to SIDR3) is a data buffer register for receiving serial data.

- The serial data signals sent to the serial input pins (SIDR0 to SIDR3) are converted in the shift register, and stored in the serial input data registers 0 to 3 (SIDR0 to SIDR3).
- When the data length is 7 bits, the upper one bit (SIDR0 to SIDR3:D7) becomes invalid.
- When the receiving data is stored in the serial input data registers 0 to 3 (SIDR0 to SIDR3), the receiving data full flag bit (SSR0 to SSR3: RDRF) is set to "1". When receiving interrupts are enabled (SSR0 to SSR3: RIE=1), receiving interrupt requests are generated.
- Read the serial input data registers 0 to 3 (SIDR0 to SIDR3) in the state that the receiving data full flag bit (SSR0 to SSR3: RDRF) is set to "1". When the receiving data full flag bit (SSR0 to SSR3: RDRF) reads the serial input data registers 0 to 3 (SIDR0 to SIDR3), it is automatically cleared to "0".
- If any receiving error occurs (SSR0 to SSR3: PE, ORE, or FRE is "1"), the data of the serial input data registers 0 to 3 (SIDR0 to SIDR3) is invalid.

## ■ Serial Output Data Register 0 to 3 (SODR0 to SODR3)

Figure 21.4-6 shows the bit configuration of serial output data register.

**Figure 21.4-6 Serial Output Data Register 0 to 3 (SODR0 to SODR3)**

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch.0 : 000022H	.....								XXXXXXXXB
ch.1 : 000028H	.....	D7	D6	D5	D4	D3	D2	D1	
ch.2 : 00002EH		W	W	W	W	W	W	W	
ch.3 : 000034H		W	W	W	W	W	W	W	

W : Write only  
X : Undefined

The serial output data register 0 to 3 (SODR0 to SODR3) is a data buffer register for transmitting serial data.

- When sending operation is allowed (SCR0 to SCR3: TXE=1), and if the sending data is written to the serial output data registers 0 to 3 (SODR0 to SODR3), the sending data is transferred to the sending shift register to be converted into the serial data and is sent out from the serial data output pins (SOT0 to SOT3 pins).
- When the data length is seven bits, the data in upper one bit (SODR0 to SODR3:D7) is invalid.
- The sending data empty flag (SSR0 to SSR3: TDRE) is cleared to "0" as soon as sending data is written into the serial output data registers 0 to 3 (SODR0 to SODR3).
- The sending data empty flag (SSR0 to SSR3: TDRE) is set to "1" as soon as transmission to the sending shift register completes.
- If the sending data empty flag (SSR0 to SSR3: TDRE) is set to "1", the next sending data can be written in. When sending interrupts are enabled, a sending interrupt is generated. Write the subsequent sending data in the state that the sending data empty flag (SSR0 to SSR3: TDRE) is "1".

---

Note:

Serial output data registers are write-only, while serial input data registers are read-only. Both of the registers are located in the same address, so the writing values and reading values are different. Instructions, such as the INC/DEC instruction, which provide the read modify write (RMW) operation, cannot be used.

---



## 21.4.5 UART Prescaler Control Register 0 to 3 (UTCR0 to UTCR3) and UART Prescaler Reload Register 0 to 3 (UTRLR0 to UTRLR3)

UART prescaler control registers 0 to 3 (UTCR0 to UTCR3) are responsible of setting start-up/halt of the prescaler, forced reset, and selecting clock sources. The UART prescaler control registers 0 to 3 (UTCR0 to UTCR3) are also responsible of setting the division rate of the machine clock by combining their lower 3 bits with UART prescaler reload registers 0 to 3 (UTCR0 to UTCR3).

### ■ UART Prescaler Control Register 0 to 3 (UTCR0 to UTCR3) and UART Prescaler Reload Register 0 to 3 (UTRLR0 to UTRLR3)

The UART operation clock is obtained by dividing the machine clock. Is designed to obtain a certain level of baud rate for a wide variety of machine cycles using this prescaler. Figure 21.4-7 shows the configuration of UTCR0 to UTCR3.

**Figure 21.4-7 UART Prescaler Control Register 0 to 3 (UTCR0 to UTCR3) and UART Prescaler Reload Register 0 to 3 (UTRLR0 to UTRLR3)**

UART prescaler control register 0 to 3 (UTCR0 to UTCR3)									
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
ch.0 : 000025H	MD	SRST	CKS	Reserved	—	D10	D9	D8	0000-000B
ch.1 : 00002BH									
ch.2 : 000031H	R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	
ch.3 : 000037H									
UART prescaler reload register 0 to 3 (UTRLR0 to UTRLR3)									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch.0 : 000024H	D7	D6	D5	D4	D3	D2	D1	D0	00000000B
ch.1 : 00002AH									
ch.2 : 000030H	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
ch.3 : 000036H									
R/W : Readable/Writable									
— : Undefined									

#### [bit 15] MD: Prescaler permission bit

It is an operation permission bit of the prescaler.

0: Prescaler is stop.

1: Prescaler is in operation.

#### [bit 14] SRST: UART compulsion reset bit

It is an internal reset bit for UART. Forcibly resets UART to initialize. Once initialized, turns to "0" automatically.

0: State maintenance

1: Forced reset

## [bit 13] CKS: clock source selection bit

The clock source is selected.

0: Dedicated baud rate generator

1: External clock

## [bit 12] Reserved: reserved bit

It is Reserved bit.

Be sure to set this bit to "0".

## [bit 11] Undefined bit

This bit is undefined when it is read. Nothing is affected when it is written.

## [bit 10 to bit 0] D10 to D0: Fundamental period set bit

<Asynchronous mode>

Two cycles of the serial clock worth of a cycle is decided.

UART is responsible for dividing the serial clock into 8 pieces, and baud rate is as follows.

BAUD RATES =  $\phi/4n$ (bps)

$\phi$ : Machine clock n: D10 to D0 (fundamental period set bit)

However, please set neither n=1 or 2 nor 3.

<Clock synchronous mode>

BAUD RATES =  $2\phi/n$  (bps)

$\phi$  : Machine clock n : D10 to D0 (fundamental period set bit)

However, please set n to "00<sub>B</sub>", in case of  $n \geq 16$  (D1, D0).

---

### Notes:

- Setting "01<sub>B</sub>", "10<sub>B</sub>", and "11<sub>B</sub>" to D1 and D0 (bit 1 and 0 of UTRLR0 to UTRLR3) of the basic cycle set bit generates the serial clock whose duty ratio is different. So set D1 and D0 (bit 1 and 0 of UTRLR0 to UTRLR3) to "00<sub>B</sub>" when in the clock synchronization mode.
  - Please access UTCR0 to UTCR3 and UTRLR0 to UTRLR3 by word move operation.
  - Halt the operation of the prescaler (MD=0 and CKS=0) before you switch the clock source.
  - When the external selection mode is selected and if you need to modify the reload value, halt the operation of the prescaler (MD=0 and CKS=0) before you change to your reload value.
  - Please set neither reload value n=1, 2 nor 3.
-

## 21.5 UART Interrupt

The UART support reception and transmission interrupts, capable of generating an interrupt request in the following conditions:

- Where the receiving data is set to the serial input data registers 0 to 3 (SIDR0 to SIDR3), or an receiving error has occurred.
- When data to transmit is transferred from serial output data register 0 to 3 (SODR0 to SODR3) to the transmission shift register.

Also, support for each extended intelligent I/O service (EI<sup>2</sup>OS),  $\mu$ DMAC.

### ■ UART Interrupt

Table 21.5-1 shows the interrupt control bit and interrupt factor of UART.

**Table 21.5-1 Interrupt Control Bit of UART and Interruption Factor**

Transmission/ Reception	Serial Status Register 0 to 3 (SSR0 to SSR3)						
	Interrupt flag bit	Operating mode			Interrupt cause	Interruption permission bit	Clear Interrupt flag
		0	1	2			
Reception	RDRF	○	○	○	Load receive data to the buffer (SIDR0 to SIDR3)	RIE	• Reading receive data • Reset
	ORE	○	○	○	Generating overrun error		• Writing "0" to the reception error flag clear bit (SSR0 to SSR3: REC) • Reset
	FRE	○	○	×	Generating framing error		
	PE	○	×	×	Generating parity error		
Transmission	TDRE	○	○	○	Transmission buffer (SODR0 to SODR3) is empty.	TIE	• Writing transmit data • Reset

○: Using bit

×: Unused bit

#### ● Reception Interrupt

When receiving interrupts are enabled (SSR0 to SSR3: RIE=1) and if either completion of data reception (SSR0 to SSR3: RDRF=1), an overrun error (SSR0 to SSR3: ORE=1), a framing error (SSR0 to SSR3: FRE=1), or a parity error (SSR0 to SSR3: PE=1) occurs, then an receiving interrupt request is generated.

When the receiving data full flag (SSR0 to SSR3: RDRF) reads the serial input data registers 0 to 3 (SIDR0 to SIDR3: RDRF), it is automatically cleared to "0". Each reception error flag (SSR0 to SSR3: PE, ORE, FRE) is cleared to "0" when "0" is written to the reception error flag clear bit (SCR0 to SCR3: REC).

In the case that any receiving error (a parity error, an overrun error, a framing error) occurs, handle such error where necessary, and then write "0" in the receiving error flag clear bit (SCR0 to SCR3: REC) to clear each of the receiving error flags.

## ● Transmission Interrupt

When sending data is sent from the serial output data registers 0 to 3 (SODR0 to SODR3) to the sending shift register, then the sending data empty flag bit (SSR0 to SSR3: TDRE) is set to "1". When sending interrupts are enabled (SSR0 to SSR3: TIE=1), a sending interrupt request is generated.

## ■ Interruption of UART, EI<sup>2</sup>OS, and $\mu$ DMAC

**Table 21.5-2 Interruption of UART, EI<sup>2</sup>OS, and  $\mu$ DMAC**

Interrupt cause	Interrupt number	Interrupt control register		Vector table address			EI <sup>2</sup> OS	$\mu$ DMAC Channel number
		Register Name	Address	Low	High	Bank		
UART3 Reception Interrupt	#35(23 <sub>H</sub> )	ICR12	0000BC <sub>H</sub>	FFFF70 <sub>H</sub>	FFFF71 <sub>H</sub>	FFFF72 <sub>H</sub>	⊙	10
UART3 Transmission Interrupt	#33(21 <sub>H</sub> )	ICR11	0000BB <sub>H</sub>	FFFF78 <sub>H</sub>	FFFF79 <sub>H</sub>	FFFF7A <sub>H</sub>	○	11
UART2 Reception Interrupt	#35(23 <sub>H</sub> )	ICR12	0000BC <sub>H</sub>	FFFF70 <sub>H</sub>	FFFF71 <sub>H</sub>	FFFF72 <sub>H</sub>	⊙	10
UART2 Transmission Interrupt	#33(21 <sub>H</sub> )	ICR11	0000BB <sub>H</sub>	FFFF78 <sub>H</sub>	FFFF79 <sub>H</sub>	FFFF7A <sub>H</sub>	○	11
UART1 Reception Interrupt	#39(27 <sub>H</sub> )	ICR14	0000BE <sub>H</sub>	FFFF60 <sub>H</sub>	FFFF61 <sub>H</sub>	FFFF62 <sub>H</sub>	⊙	12
UART1 Transmission Interrupt	#37(25 <sub>H</sub> )	ICR13	0000BD <sub>H</sub>	FFFF68 <sub>H</sub>	FFFF69 <sub>H</sub>	FFFF6A <sub>H</sub>	○	13
UART0 Reception Interrupt	#39(27 <sub>H</sub> )	ICR14	0000BE <sub>H</sub>	FFFF60 <sub>H</sub>	FFFF61 <sub>H</sub>	FFFF62 <sub>H</sub>	⊙	12
UART0 Transmission Interrupt	#37(25 <sub>H</sub> )	ICR13	0000BD <sub>H</sub>	FFFF68 <sub>H</sub>	FFFF69 <sub>H</sub>	FFFF6A <sub>H</sub>	○	13

⊙: Available; with a function that stops EI<sup>2</sup>OS by detecting a UART receive error

○: Available

## ■ UART EI<sup>2</sup>OS Function

UART has the circuit of the EI<sup>2</sup>OS correspondence. This allows EI<sup>2</sup>OS to start up separately on the occasion of each of the send interrupt and the receive interrupt.

## ● At Transmission/Reception

At the transmission / reception, EI<sup>2</sup>OS is available regardless of the states of any other peripherals.

## 21.5.1 Receive Interrupt Generation and Flag Set Timing

---

Interrupts during reception are one generated upon completion of reception (SSR0 to SSR3: RDRF) and one generated upon occurrence of a reception error (SSR0 to SSR3: PE, ORE, FRE).

---

### ■ Receive Interrupt Generation and Flag Set Timing

When data is received, it is stored in serial input data register 0 to 3 (SIDR0 to SIDR3) upon detection of the stop bit (in operation mode 0 or 1) or of the data's last bit (SIDR0 to SIDR3: D7) (in operation mode 2). When a receiving error occurs, an error flag (SSR0 to SSR3: PE, ORE, FRE) is set, a receiving data full flag (SSR0 to SSR3: RDRF) is set. If any of the flags is set to the each operation mode, the serial input data registers 0 to 3 (SIDR0 to SIDR3) that have received are invalid.

#### ● Operation mode 0 (Asynchronous normal mode)

The receiving data full flag (SSR0 to SSR3: RDRF) is set on detection of the stop bit. When a reception occurs, the error flag (SSR0 to SSR3: PE, ORE, FRE) is set.

#### ● Operating mode 1 (asynchronous multiprocessor mode)

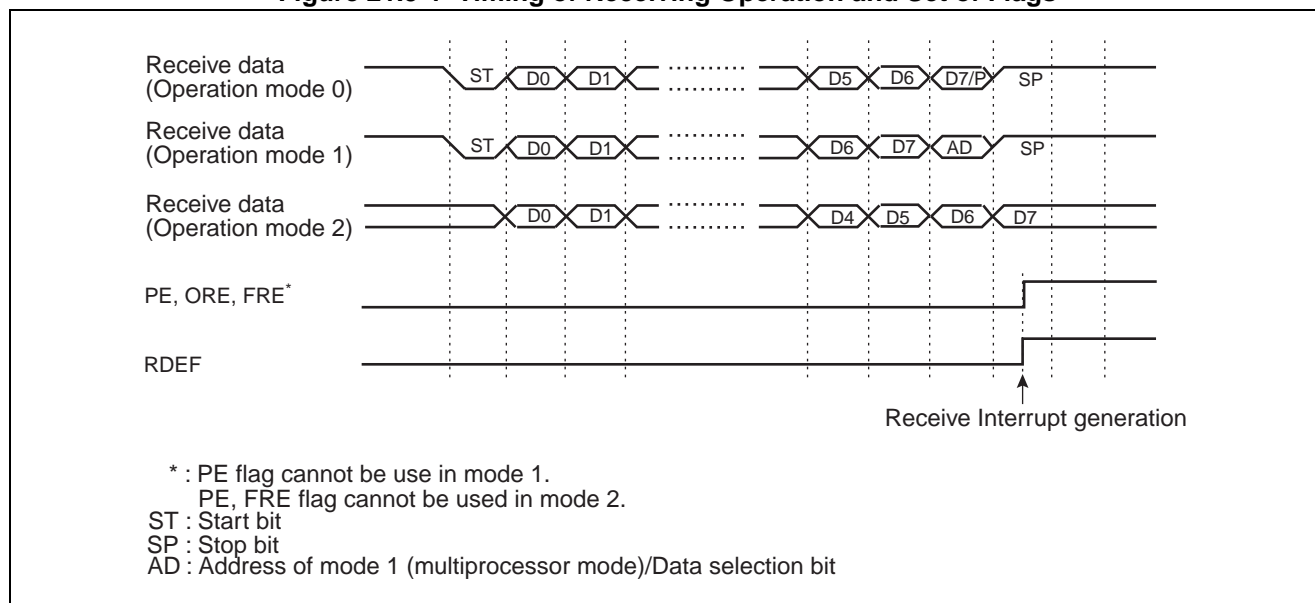
The receiving data full flag bit (SSR0 to SSR3: RDRF) is set when the stop bit is detected. When a reception error occurs, the error flag (SSR0 to SSR3: ORE, FRE) is set. But parity errors (SSR0 to SSR3:PE) cannot be detected.

#### ● Operation mode 2 (clock synchronizer normal mode)

When the last bit (SIDR0 to SIDR3: D7) of the receiving data is detected, the receiving data full flag bit (SSR0 to SSR3: RDRF) is set. When a reception error occurs, the error flag (SSR0 to SSR3:ORE) is set. Neither a parity error (SSR0 to SSR3:PE) nor a framing error (SSR0 to SSR3:FRE) can be detected.

Figure 21.5-1 shows the timing of receiving operation and the set of flags.

Figure 21.5-1 Timing of Receiving Operation and Set of Flags



### ● Timing of receiving interrupt generation

When receiving interrupts are enabled (SSR0 to SSR3: RIE=1), any of a receiving data full flag (SSR0 to SSR3: RDRF), a parity error flag (SSR0 to SSR3: PE), an overrun error flag (SSR0 to SSR3: ORE), and a framing error flag (SSR0 to SSR3: FRE) is set to "1", then a receiving interrupt request is generated.

## 21.5.2 Transmit Interrupt Generation and Flag Set Timing

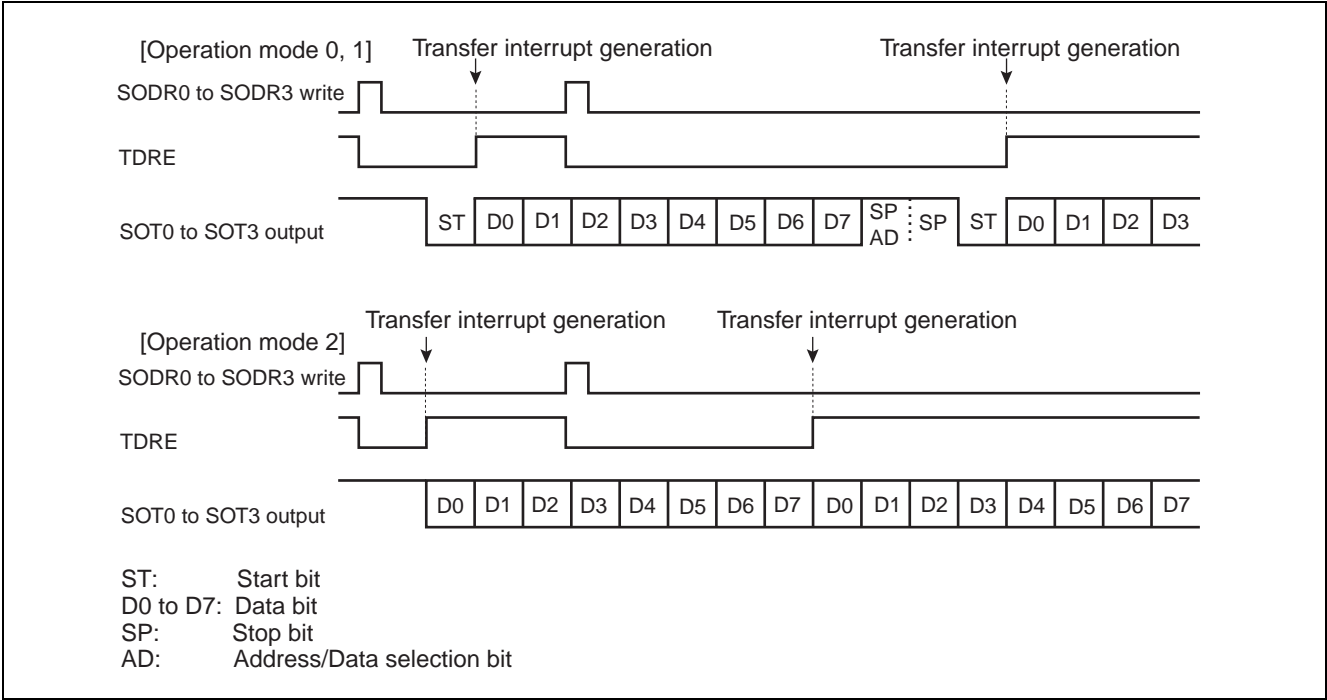
An interrupt during transmission is generated when serial output data register 0 to 3 (SODR0 to SODR3) becomes empty, or ready to accommodate the next data to transmit.

### ■ Transmit Interrupt Generation and Flag Set Timing

The sending data empty flag bit (SSR0 to SSR3: TDRE) is set to "1" in the state that the sending data that is written to the serial output data registers 0 to 3 (SODR0 to SODR3) is transferred to the sending shift register, the subsequent data goes into the readable state. When the subsequent data is written to the serial output data registers 0 to 3 (SODR0 to SODR3), the sending data empty flag bit (SSR0 to SSR3: TDRE) is cleared to "0".

Figure 21.5-2 shows the timing of sending operation and the set of flags.

Figure 21.5-2 Timing of Sending Operation and the Set of Flags



- Timing transmission interrupt request generation

When sending interrupts are enabled (SSR0 to SSR3: TIE=1) and if the sending data empty flag bit (SSR0 to SSR3: TDRE) is set to "1", a sending interrupt request is generated.

---

Note:

If the sending operation is set to be disabled (SCR0 to SCR3: TXE=0, in the operation mode 1, also including receiving operation disabled RXE) in the middle of sending operation, the sending data empty flag bit is set (SSR0 to SSR3: TDRE=1), the shift operation of the sending shift register is halted and then UART communication operation is disabled. The send data written to the serial output data register 1 before the transmission stops 0 to 3 (SODR0 to SODR3) is sent.

By default TDRE bit is "1". So, as soon as sending interrupts are enabled (TIE=1), the interrupt indicating completion of transmission is generated. TDRE bit is a read-only bit and has no other way to clear than by writing new data in the serial output data registers 0 to 3 (SODR0 to SODR3) to clear. So be careful when to enable a sending interrupt.

---



## 21.6 UART Baud Rate

---

The sending and receiving clocks of UART has the following alternatives.

- Internal clock (reload counter)
  - External clock (reload counter)
  - External clock (clock input to SCK pin)
- 

### ■ UART Baud Rate Selection

You can select one of the following three different baud rates:

- Baud rate resulting from a dedicated baud rate generator of the internal clock.

The baud rate can be selected by setting the 11-bit reload value on settings of the UART prescaler control registers 0 to 3 (UTCR0 to UTCR3), and the UART prescaler reload registers 0 to 3 (UTRLR0 to UTRLR3). The reload counter divides the machine clock by the specified value. It is used in the synchronous and asynchronous modes.

- Baud rate resulting from a dedicated baud rate generator of the external clock.

The external clock is used as the clock source for the reload counter. The baud rate can be selected by setting the 11-bit reload value on settings of the UART prescaler control registers 0 to 3 (UTCR0 to UTCR3), and the UART prescaler reload registers 0 to 3 (UTRLR0 to UTRLR3). The reload counter divides the external clock frequency by the set value. It is used in the asynchronous modes.

- Baud rate of the external clock (one-to-one mode)

The clock that is entered from the UART clock input pins (SCK0 to SCK3) is used as the baud rate as it is. It is used in the synchronous mode.

## 21.6.1 Baud Rate of the UART Internal Clock Using the Dedicated Baud Rate Generator

---

Indicates baud rates possible to set when selecting a dedicated baud rate generator, as an UART transfer clock.

---

### ■ Baud Rate of the Internal Clock Using the Dedicated Baud Rate Generator

Divides the machine clock by setting the 11-bit reload value at the UART prescaler control registers 0 to 3 (UTCR0 to UTCR3), and the UART prescaler reload registers 0 to 3 (UTRLR0 to UTRLR3).

<Asynchronous mode>

BAUD RATES =  $\phi/4n$  (bps)

$\phi$ : Frequency of the machine clock, n: Values of D10 to D0 (UTCR0 to UTCR3, UTRLR0 to UTRLR3)

However, please set neither n=1, 2 nor 3.

<Synchronous mode>

BAUD RATES =  $2\phi/n$  (bps)

$\phi$ : Frequency of the machine clock, n: Values of D10 to D0 (UTCR0 to UTCR3, UTRLR0 to UTRLR3)

However, please set n to "00<sub>B</sub>", in case of  $n \geq 16$  (D1, D0).

---

Note:

Setting "01<sub>B</sub>", "10<sub>B</sub>", and "11<sub>B</sub>" to D1 and D0 (bit1 and bit0 of UTRLR0 to UTRLR3) of the basic cycle set bit generates the serial clock whose duty ratio is different. So set D1 and D0 (bit1 and bit0 of UTRLR0 to UTRLR3) to "00<sub>B</sub>" when in the clock synchronization mode.

---

## 21.6.2 Baud Rate of the External Clock Using the Dedicated Baud Rate Generator

---

Indicates baud rates possible to set when a dedicated baud rate generator of an external clock is selected, as UART transfer clock. It is used in the clock asynchronous modes.

---

### ■ Baud Rate of the External Clock Using the Dedicated Baud Rate Generator

Divides the clock by setting the 11-bit reload value at the UART prescaler control registers 0 to 3 (UTCR0 to UTCR3), and the UART prescaler reload registers 0 to 3 (UTRLR0 to UTRLR3).

Provided  $f$  for the frequency of the external clock, baud rates are described as follows:

- UTCR0 to UTCR3: If MD is set to "1".

$$b = f/4n \text{ (bps)}, f \leq \phi \times n/4$$

$f$ : Frequency of the external clock,  $n$ : Values of D10 to D0 (UTCR0 to UTCR3, UTRLR0 to UTRLR3),

$b$ : Baud rate

However, please set neither  $n=1, 2$  nor  $3$ .

The maximum frequency of  $f$  is 24 MHz.

### 21.6.3 Baud Rate of the External Clock (One-to-one Mode)

---

Indicates the formula for the settings and the baud rates for selecting the external clock, as UART transfer clock.

---

#### ■ Baud Rate of the External Clock (One-to-one Mode)

Selecting the baud rate of external clock (one-to-one mode) requires the following three settings.

Write "0" in MD bit of UART prescaler control registers 0 to 3 (UTCR0 to UTCR3), and "1" to CKS bit, and select the baud rate resulting from the external clock (one-to-one mode).

Set SCK pins 0 to 3 to Enter (DDR4: bit4=0, bit7=0, DDR9: bit2=0, bit5=0).

Write "0" in SCKE bit of the serial mode registers 0 to 3 (SMR0 to SMR3) to define the pin as external clock input pin.

It is used in the synchronous modes.

#### ● Calculation expression for baud rate

When the baud rate resulting from the external clock (one-to-one mode) is selected (UTCR0 to UTCR3: MD=0, CKS=1), resulted baud rate is as follows (f for the frequency of the external clock).

Synchronous =  $f'$  (bps)

$f'$ : External clock frequency

However, the maximum of  $f'$  is up to 1/8 of the machine clocks.

## 21.7 Explanation of Operation of UART

UART supports the master/slave connection based communication function (operation mode 1) as well as the typical bi-directional serial communication function (operation mode 0, operation mode 2).

### ■ Operation of UART

#### ● Operating mode

UART has 3 types of operation modes (i.e. mode 0 to 3), which can be selected based on the appropriate connection system or data transfer system between CPUs as shown in Table 21.7-1.

**Table 21.7-1 UART Operation Modes**

Operating mode		Data length		Synchronous type	Length of Stop Bit
		Without Parity	With Parity		
0	Normal mode	7 bits or 8 bits		Asynchronous	1 bit or 2 bits*2
1	Multiprocessor mode	8+1 *1 bits	-	Asynchronous	
2	Normal mode	8 bits	-	Synchronous	None

- : Unavailable

\*1: "+1" indicates the address/data select bit (AD) used for communication control.

\*2: Only one bit can be detected as a stop bit at reception.

Note:

The UART operation mode 1 is used only for the master connection on the master/slave connection.

#### ● Inter-CPU connection method

Two alternatives are provided; one-to-one connection and master/slave connection. In both cases, the data length, parity, synchronous mode, must be the same for all CPUs. The operation modes are selected as follows.

One-to-one connection: 2 CPUs should adopt the same method either in the mode 0 or in the mode 2. Select the mode 0 in the asynchronous method and the mode 2 in the synchronous method.

Master/slave connection: Set the operation mode 1. When selecting the operation mode 1, use this as master. For this connection, select no parity and a data length of 8 bits.

- Synchronous type

Either asynchronous method or (start-stop synchronization) clock synchronous method can be selected.

- Signal type

Data in NRZ (Non Return to Zero) format is only supported.

- Start of transmission/reception

When the bit indicating that sending operation is enabled (SCR0 to SCR3: TXE) is set to "1", sending operation starts.

Receiving operation starts when the reception enable bit of the serial control register (SCR0 to SCR3:RXE) is set to "1".

- Stop of transmission/reception

Transmission stops when the transmission enable bit (SCR0 to SCR3:TXE) in the serial control register is set to "0".

Reception stops when the reception enable bit (SCR0 to SCR3:RXE) in the serial control register is set to "0".

- Stop during transmission/reception

When receiving operation (SCR0 to SCR3: RXE=0) is disabled in the middle of reception (while data is being input to the receiving shift register), the receiving operation is halted after the frame that is being received has completely received and the received data has been stored into the serial input data registers 0 to 3 (SIDR0 to SIDR3).

When sending operation (SCR0 to SCR3: TXE=0) is disabled in the middle of transmission (while data is being output from the sending shift register), the sending operation is halted after one frame has been completely sent.

## 21.7.1 Operation in Asynchronous Mode (Operation Mode 0 or Operation Mode 1)

The UART uses asynchronous transfer when used in operation mode 0 (normal mode) or operation mode 1 (multiprocessor mode).

### ■ Operation in Asynchronous Mode

#### ● Format of transmit/receive data

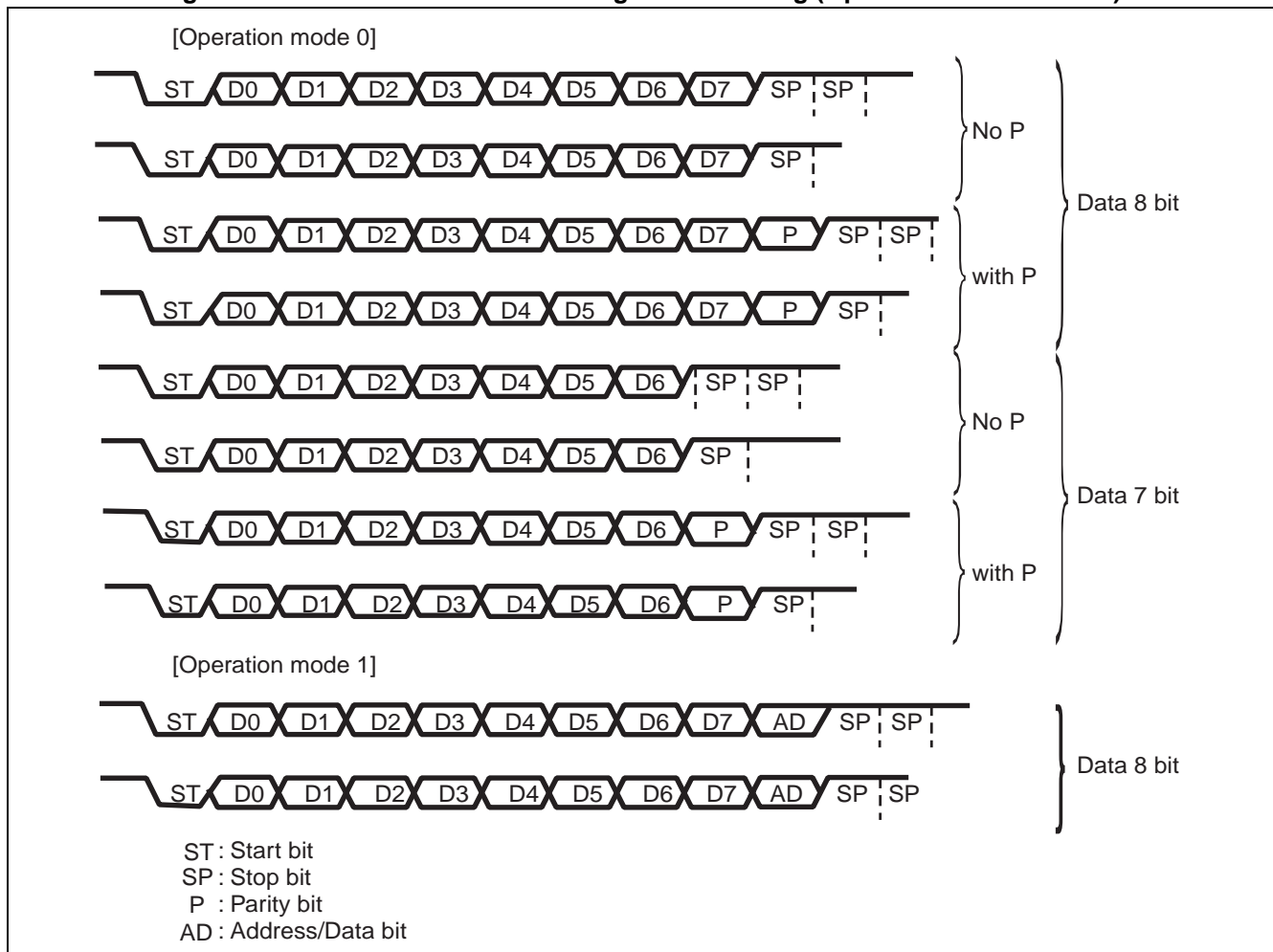
Sending and receiving always start with the start bit ("L" Level), specified data bit length is sent and received, and end with the stop bit ("H" Level).

For use in operation mode 0, select a data length of seven or eight bits. You can select whether to use the parity bit.

In operation mode 1, the data length is fixed at 8 bits. There is no parity bit. The address/data bit (SCR0 to SCR3:AD) is added as bit 9.

Figure 21.7-1 shows the data formats for sending and receiving in the asynchronous mode.

**Figure 21.7-1 Data Format for Sending and Receiving (Operation Mode 0 and 1)**



### ● Transmission Operation

- Sending data is written in the serial output data registers 0 to 3 (SODR0 to SODR3) in the state of "1" being set to the sending data empty flag bit (SSR0 to SSR3: TDRE).
- When the sending data is written, and the bit of the serial control register indicating that sending operation is enabled (SCR0 to SCR3: TXE) is set to "1", then sending starts.
- When the sending data is written to the serial output data register, the sending data empty flag bit (SSR0 to SSR3: TDRE) is once cleared to "0".
- When the sending data is transferred from the serial output data registers 0 to 3 (SODR0 to SODR3) to the sending shift register, the sending data empty flag bit (SSR0 to SSR3: TDRE) is set to "1" again.
- When the sending data is transferred from the serial output data registers 0 to 3 (SODR0 to SODR3) to the sending shift register, the sending data empty flag bit (SSR0 to SSR3: TDRE) is set to "1" again.
- When the bit indicating that the sending interrupts (SSR0 to SSR3: TIE) is enabled has already been set to "1", a sending interrupt request is generated if the sending data empty flag bit (SSR0 to SSR3: TDRE) is set to "1". In interrupt processing, the following data can be written to the serial output data registers 0 to 3 (SODR0 to SODR3).

### ● Reception Operation

- Receiving operation are always performed when the receiving operations are set to be enabled (SCR0 to SCR3: RXE=1).
- When the start bit of the receiving data is detected, one frame of data is received in the serial input data registers 0 to 3 (SIDR0 to SIDR3) based on the data format that is set in the serial input control register 0 to 3 (SCR0 to SCR3).
- One frame of data reception is completed, the receiving data full flag bit (SSR0 to SSR3: RDRF) is set to "1".
- When reading receiving data, check the state of error flags of the serial status registers 0 to 3 (SSR0 to SSR3). If receiving is successful, then read the receiving data from the serial input data register. When a reception error occurs, perform error handling.
- When the receiving data has been read, the receiving data full flag bit (SSR0 to SSR3: RDRF) is cleared to "0".

### ● Stop Bit

One bit or two bits length can be selected. However, the receive side always detects only the first bit.

### ● Error detection

In mode 0, parity, overrun, and framing error can be detected.

In the operation mode 1, overrun and framing errors can be detected. But parity errors cannot be detected.

### ● Parity bit

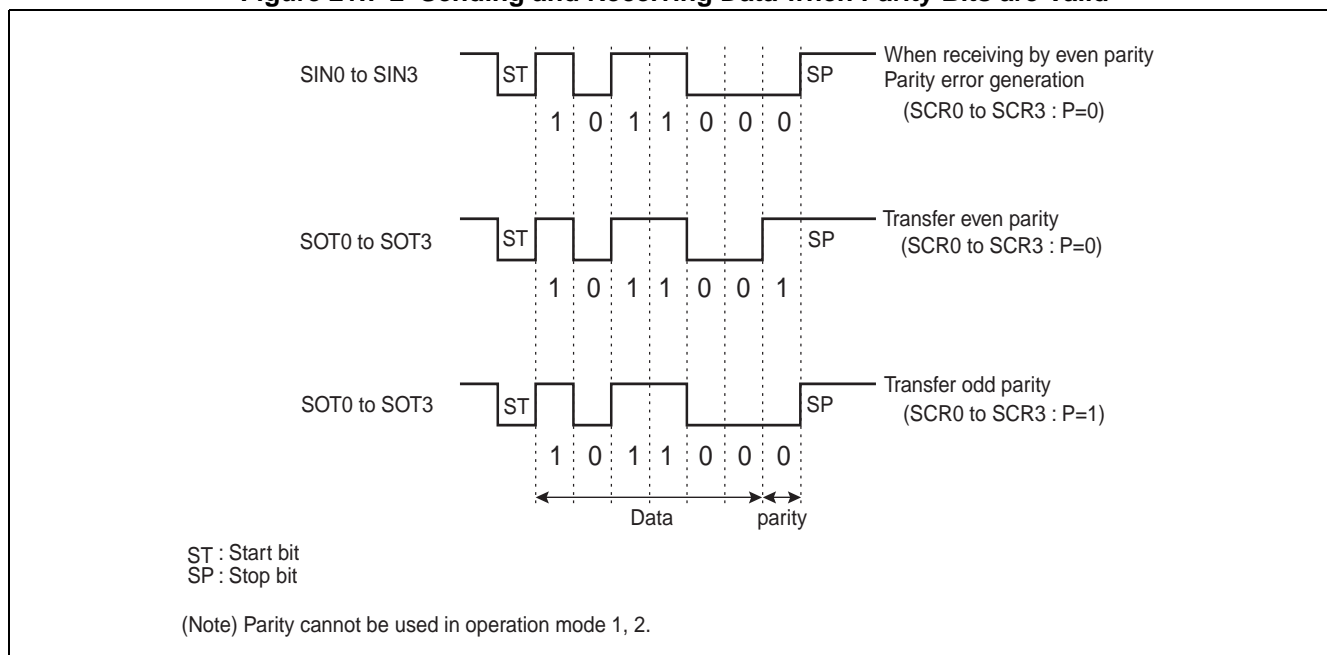
The addition of a parity bit can be set only in operation mode 0. The parity addition enable bit (SCR0 to SCR3: PEN) and parity select bit (SCR0 to SCR3:P) can be used to select whether to use parity and to set the even or odd parity, respectively.

In the operation mode 1 and 2, parity cannot be appended.

Figure 21.7-2 shows the sending and receiving data when parity bits are valid.



**Figure 21.7-2 Sending and Receiving Data when Parity Bits are Valid**



## 21.7.2 Operation in Synchronous Mode (Operation Mode 2)

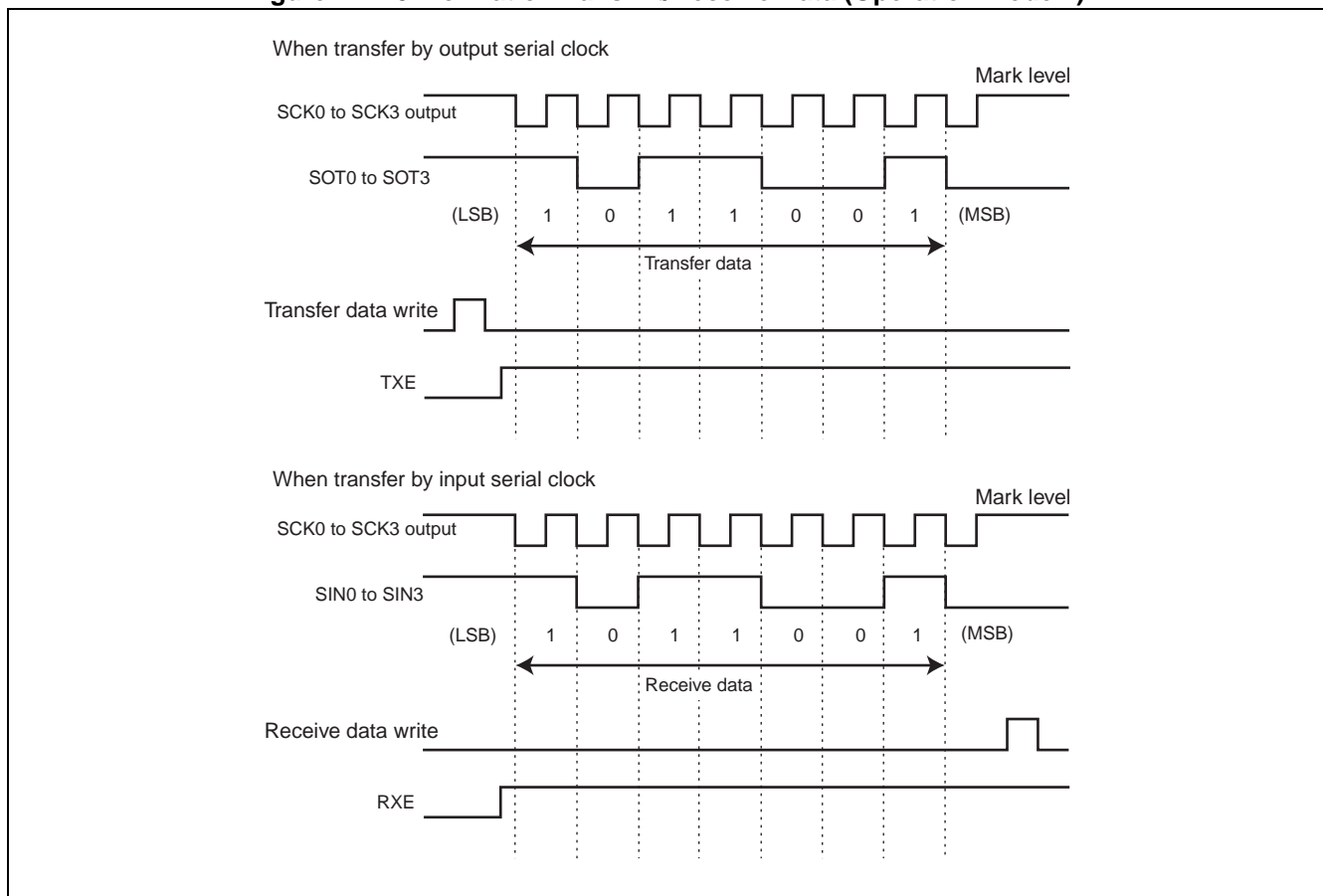
The UART uses clock-synchronous transfer when used in operation mode 2 (normal mode).

### ■ Operation in Synchronous Mode (Operation Mode 2)

#### ● Format of transmit/receive data

In the synchronous mode, 1 to 8 bits of data is transferred and the stop bit is not appended. Figure 21.7-3 shows the formats of send and receive data in the clock synchronization mode.

**Figure 21.7-3 Format of Transmit/Receive Data (Operation Mode 2)**



#### ● Clock Supply

In the clock synchronous mode, count of clocks equal to the transmit and receive bits count must be supplied.

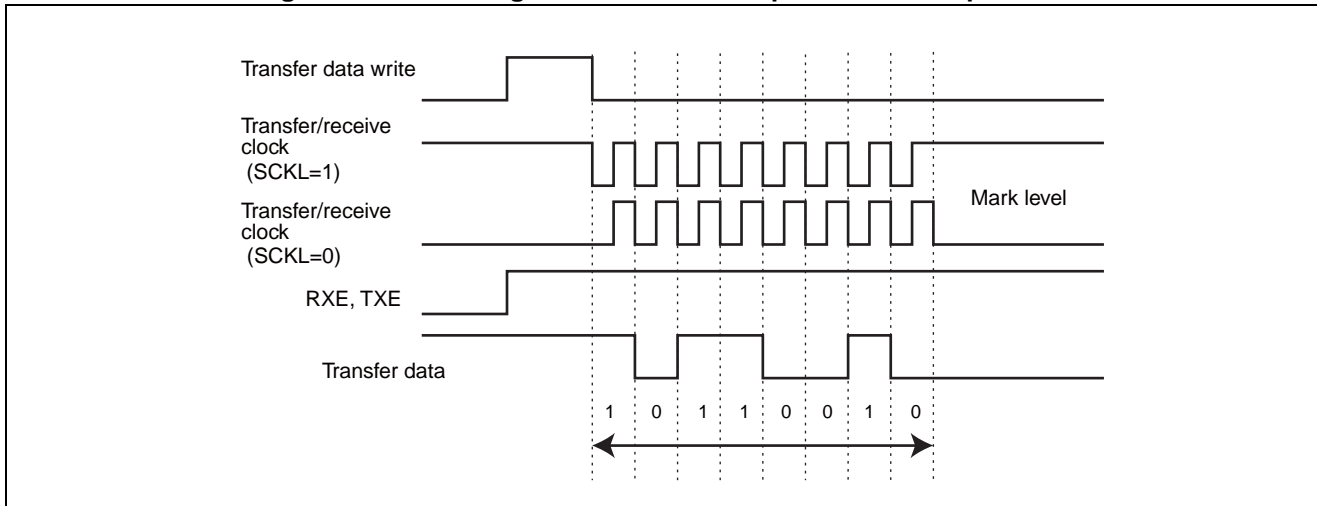
When the internal clock (a dedicated baud rate generator) has been selected, the synchronous clock for receiving data is automatically generated by sending data.

When the external clock has been selected, exactly one byte of clock should be provided externally after making sure that the serial output data registers 0 to 3 (SODR0 to SODR3) contains sending data (SSR0 to SSR3: TDRE=0). Also, both before and after sending, ensure that the level is marked with "H".

● Specification of serial clock output level at inoperative

Serial clock output level (SMR: SCL) at the clock synchronous mode and inoperative can be set.

**Figure 21.7-4 Setting of Serial Clock Output Level at Inoperative**



● Error detection

Only overrun errors can be detected. Parity and framing errors cannot be detected.

● Initialization

The setting value of each control register is described when using the synchronous mode.

[Serial mode register 0 to 3 (SMR0 to SMR3)]

MD1, MD0 : "10<sub>B</sub>"

SCKL : If the level of the serial clock output in the inactive operating state is "L", set to "1"  
: If the level of the serial clock output in the inactive operating state is "L", set to "0".

M2L2 to M2L0 : If 8-bit transmission is specified, set to "000<sub>B</sub>".

: If 7-bit transmission is specified, set to "111<sub>B</sub>".

: If 6-bit transmission is specified, set to "110<sub>B</sub>".

: If 5-bit transmission is specified, set to "101<sub>B</sub>".

: If 4-bit transmission is specified, set to "100<sub>B</sub>".

: If 3-bit transmission is specified, set to "011<sub>B</sub>".

: If 2-bit transmission is specified, set to "010<sub>B</sub>".

: If 1-bit transmission is specified, set to "001<sub>B</sub>".

SCKE : "1" for the dedicated baud rate generator, "0" for the clock output and the external clock (clock input).

SOE : "1" for transmitting, "0" only in case of reception

[Serial control register 0 to 3 (SCR0 to SCR3)]

PEN : "0"

P, SBL, and AD : These bits do not have the meaning.

CL : "1"(8-bit data)

REC : "0" (for initialization, error flags cleared).

RXE and TXE : At least, it is "1" as for either

[Serial Status Register 0 to 3 (SSR0 to SSR3)]

RIE : "1" to use interrupts, "0" not to use interrupts.

TIE : "1" to use interrupts, "0" not to use interrupts.

## ● Starting communications

When the sending data is written in the serial output data registers 0 to 3 (SODR0 to SODR3), communication starts. To start communication, temporal sending data must be written to the serial output data registers 0 to 3 (SODR0 to SODR3) even when only receiving operation is necessary.

## ● Terminating communications

As soon as one frame of data is sent and received, the receiving data full flag bit (SSR0 to SSR3: RDRF) is set to "1". Having received the data, check the overrun error flag bit (SSR0 to SSR3: ORE) to ensure that the communication was successfully done.

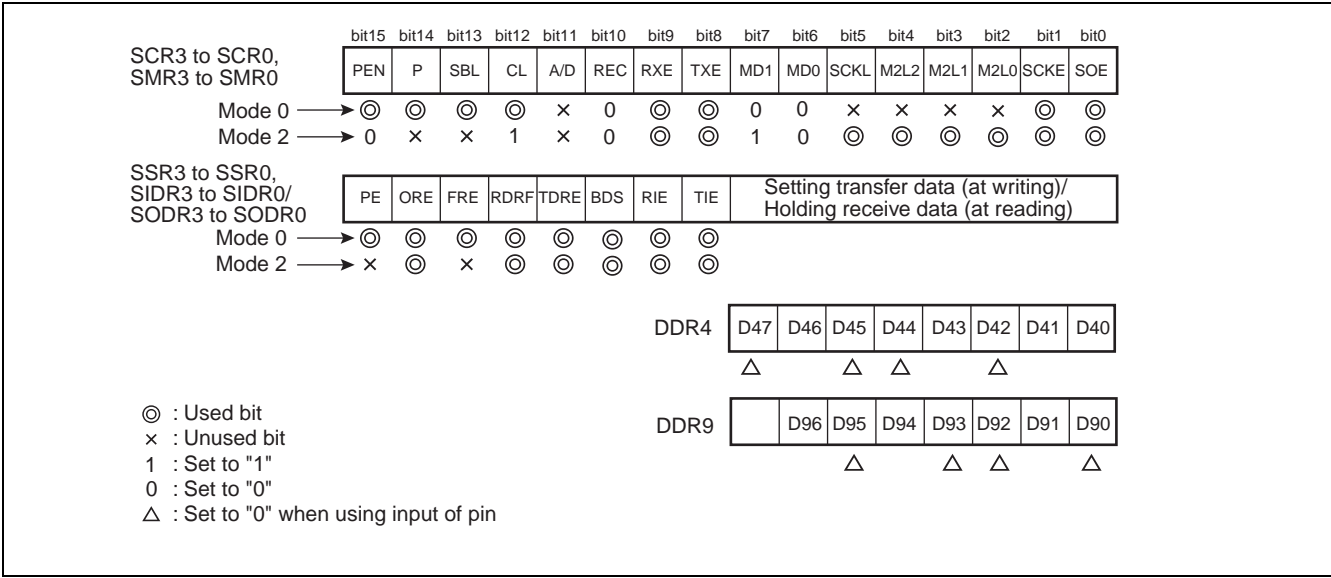
21.7.3 Bi-directional Communication Function (Normal Mode)

In the mode 0 and 2, typical serial bi-directional communication on the one-to-one connection is available. The communication clock mode becomes asynchronous for the operation mode 0, synchronous for the operation mode 2.

■ Bi-directional Communication Function

To operate UART in the normal mode (operation mode 0, operation mode 2), the settings described in Figure 21.7-5 must be executed.

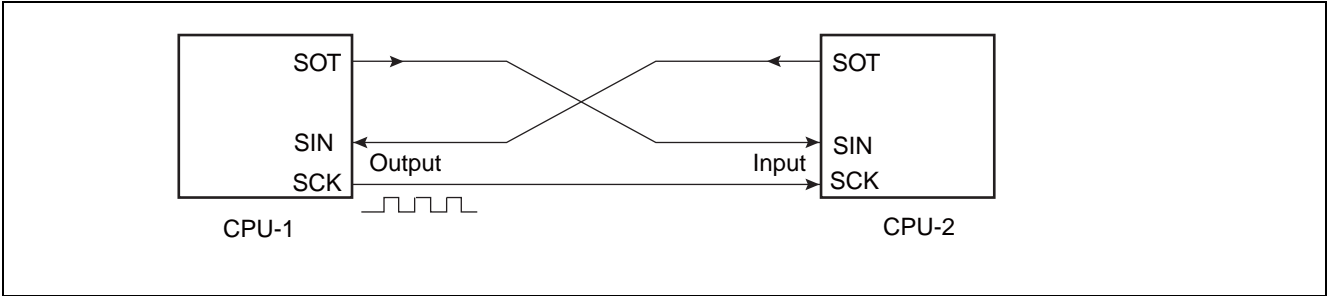
Figure 21.7-5 Setting of Operation Mode 0 in UART



● Inter-CPU connection

As shown in Figure 21.7-6, connect 2 CPUs each other.

Figure 21.7-6 Example of Bi-directional Communication Connect for UART

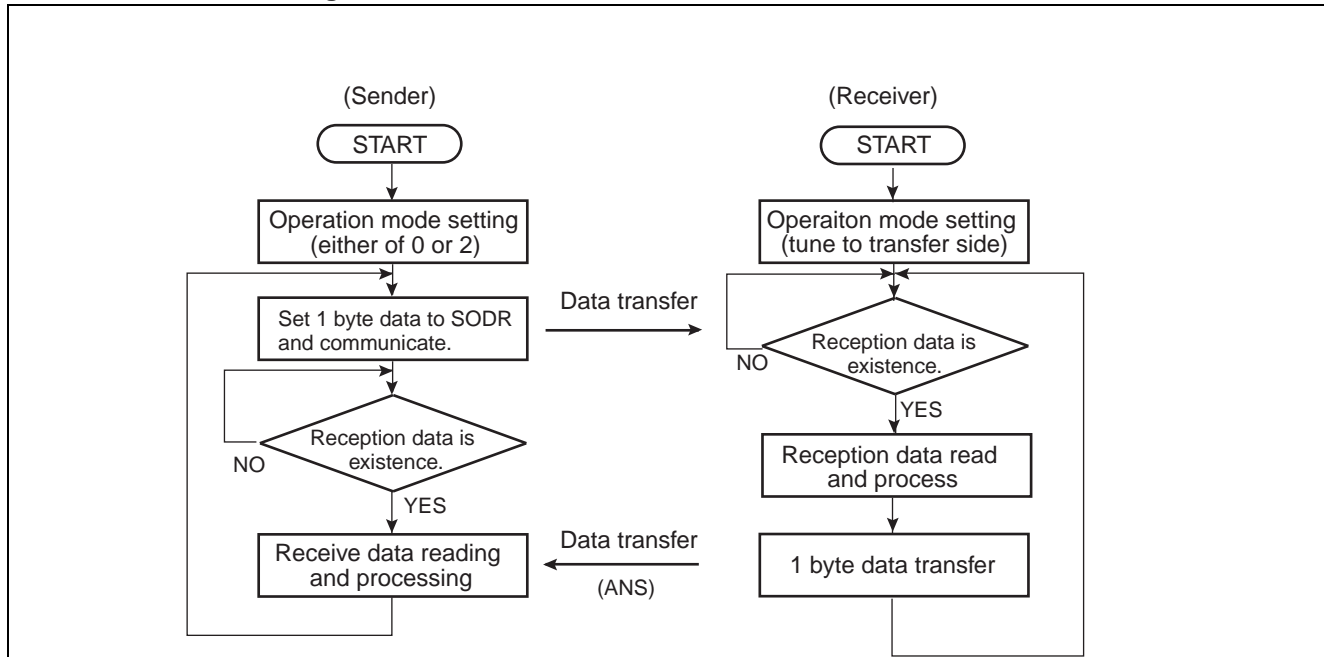


## ● Communication procedure

Communications start at any timing from the transmitting side when transmit data is provided. On the transmission side, load transmit data into the serial output data register 0 to 3 (SODR0 to SODR3) and set the transmission enable bit (SCR0 to SCR3:TXE) in the serial control register to "1" to start transmission.

Figure 21.7-7 shows the example of transferring the receiving data to the sender to indicate that the data was successfully sent. Upon the receipt of the sending data, the receiver periodically returns ANS (1 byte in this example).

**Figure 21.7-7 Flowchart for Bi-directional Communication**



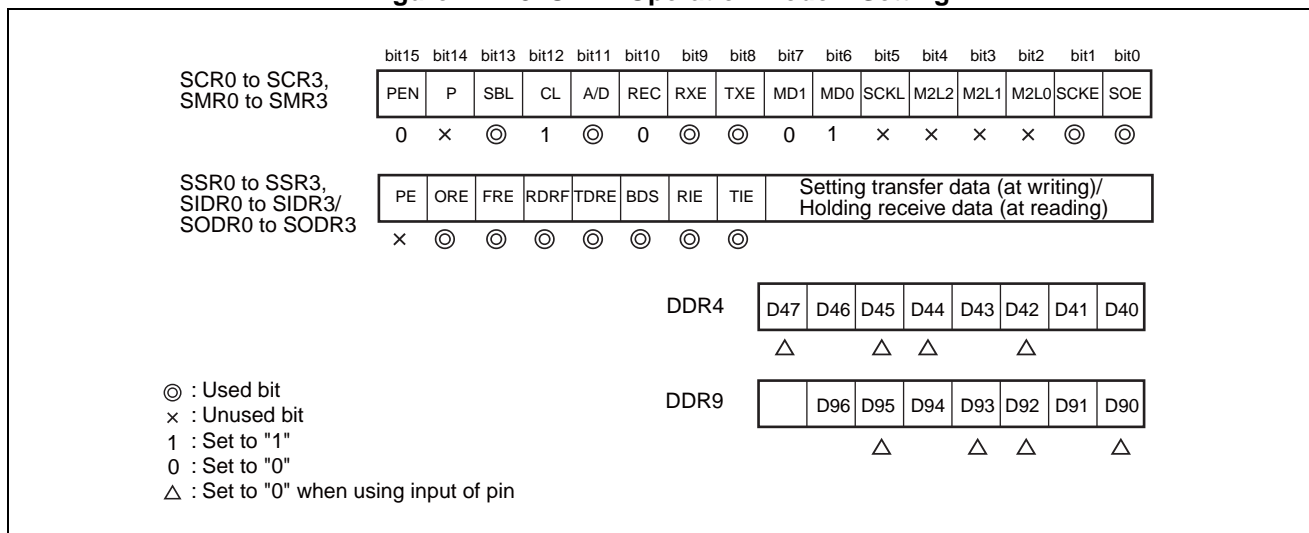
## 21.7.4 Master/Slave Mode Communication Function (Multi-processor Mode)

Operation mode 1 allows communication between multiple CPUs connected in a master/slave configuration. However, this is available only as master.

### ■ Master/Slave Mode Communication Function

To operate UART in the multi-processor mode (operation mode 1), the settings described in Figure 21.7-8 must be executed.

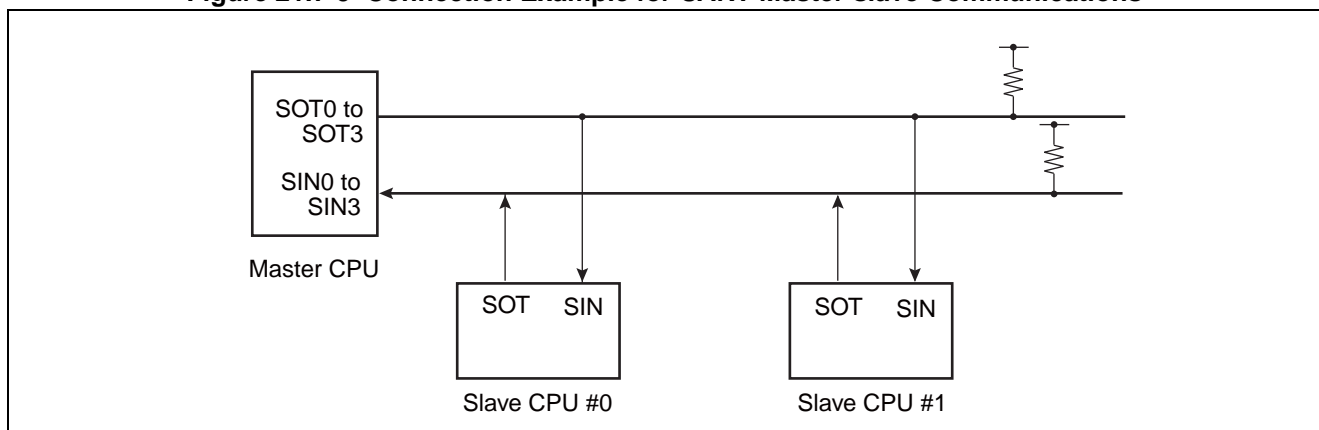
Figure 21.7-8 UART Operation Mode 1 Setting



### ● Inter-CPU connection

One master CPU and two or more slave CPUs are connected to a pair of common communication lines to make up the master/slave communication system as shown in Figure 21.7-9. The UART1 can be used only as the master CPU.

Figure 21.7-9 Connection Example for UART Master-slave Communications



### ● Function Selection

When it comes to master/slave communication, select the operation mode and data transfer direction, as shown in Table 21.7-2. Since the parity check function cannot be used in operation mode 1, set the parity enable bit (SCR0 to SCR3:PEN) to "0".

**Table 21.7-2 Select of Master/Slave Communication Function**

	Operating mode		Data	Parity	Synchronous type	Stop bit
	Master CPU	Slave CPU				
Address Trans-mission/ Reception	Mode 1	-	AD= 1 + 8 bits Address	Not provided	Asynchronous	1 bit or 2 bits
Data Trans-mission/ Reception			AD= 0 + 8 bits Data			

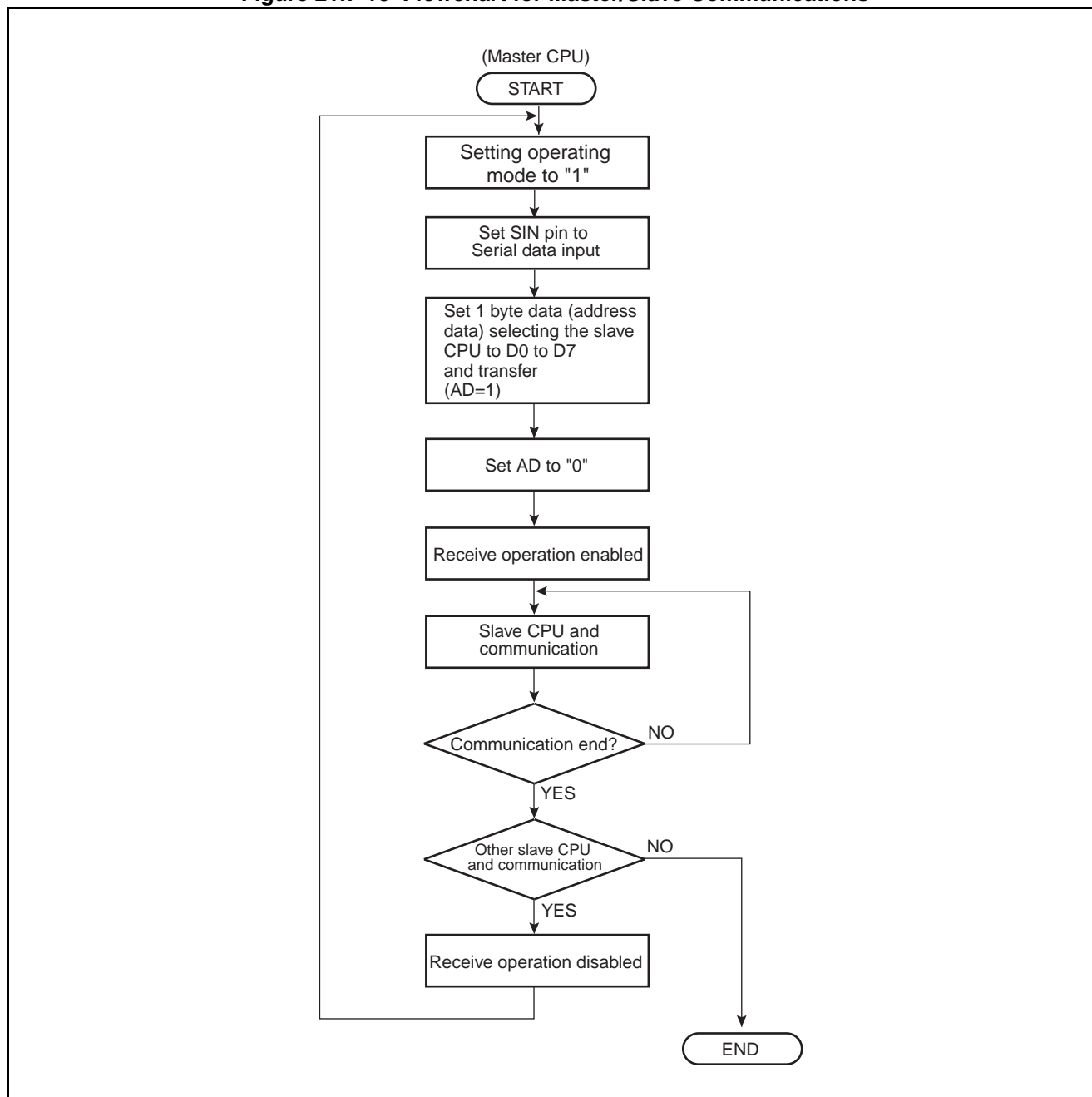
### ● Communication procedure

The communication starts when the master CPU transmits the address data. The address data, having an AD bit as 1, locates the slave CPU as the destination. Each slave CPU identifies the address data based on the program, and if there is a match with the address that is already assigned, the slave CPU communicates (typical data) with the master CPU.

Figure 21.7-10 shows the flowchart of the master/slave communication (multiprocessor mode).



Figure 21.7-10 Flowchart for Master/Slave Communications



## 21.8 Notes on Using UART

---

Use of the UART requires the following cautions.

---

### ■ Notes on Using UART

#### ● Enabling sending and receiving

- The bits indicating that sending operation is enabled (SCR0 to SCR3: TXE) and that receiving operation is enabled (SCR0 to SCR3: RXE) is provided for sending and receiving respectively.
- In the initial state after a reset, transmission and reception are both disabled (SCR0 to SCR3:TXE=0,RXE=0). Transmission and reception must therefore be enabled in advance.
- The device can stop transmission and reception by disabling them (SCR0 to SCR3:TXE=0,RXE=0).

#### ● Setting operation mode

Set the operation mode after disabling transmission and reception (SCR0 to SCR3:TXE=0,RXE=0). When the operation mode is changed during transmission or reception, the transmitted/received data is not guaranteed.

#### ● About clock synchronous mode

Operation mode 2 is set as a clock synchronizer method. Transmit/receive data is associated with no start and stop bits.

#### ● Transmission interrupt enable timing

The sending data empty flag bit (SSR0 to SSR3: TDRE) is set to "1" (no sending data, enabled to write sending data) by default after resetting. So, as soon as a sending interrupt is enabled (SSR0 to SSR3: TIE=1), the sending interrupt request is generated. Be sure to prepare data to transmit before enabling transmission (SSR0 to SSR3:TIE=1).

#### ● Receiving Multiprocessor mode

In the multiprocessor mode, receiving operation based on 9-bit receipt is not allowed.

## 21.9 Example of UART Programming

This section provides program example for UART.

### ■ Example of UART Programming

#### ● Processing specification

Perform serial transmission/reception using the bi-directional communication function (normal mode) of the UART.

Defined as: Operation mode 0, asynchronous, 8 bits of data length, 2 bits of stop bit length, without parity.

Use the P42/A10/SIN0, P43/A11/SOT0 pins for communication.

The baud rate is set to about 9600 bps using the dedicated baud rate generator.

Send character "13<sub>H</sub>" from the SOT0 pin to receive using an interrupt.

Set the machine clock ( $\phi$ ) to 16 MHz.

#### ● Coding example

```
ICR14    EQU    0000BEH        ; UART send/receive interrupt control register.
DDR4     EQU    000014H        ; Port 4 direction register
SMR0     EQU    000020H        ; Serial mode register 0
SCR0     EQU    000021H        ; Serial control register 0
SIDR0    EQU    000022H        ; Serial input data register 0
SODR0    EQU    000022H        ; Serial output data register 0
SSR0     EQU    000023H        ; Serial Status Register 0
UTCR0    EQU    000025H        ; UART prescaler control register 0
UTRLR0   EQU    000024H        ; UART prescaler reload register 0
REC      EQU    SCR:2          ; Receive error flag clear bit
```

;-----Main Program-----

```
CODE     CSEG    ABS = 0FFH
```

```
START:
```

```
;          :          ; Initialize such as a stack pointer (SP).
```

```
; Defined as done.
```

```
AND      CCR,#0BFH          ; Disables the interrupt.
```

```
MOV      I:ICR14, #00H      ; Interrupt levels 0 (strength)
```

```
MOV      I:DDR4, #00000000B ; Sets the SIN0 pin for input.
```

```
MOV      I:SMR0, #00000001B ; operation mode 0 (asynchronous)
```

```
; Disables the clock output and enables the data  
; output.
```

```
MOVW     I:UTRL0, #81A0H    ; uses Dedicated baud rate generator
```

```
; (9615bps Selection)
```

```

MOV    I:SCR0, #00010011B    ; Parity none and stop bit 2 bits
                                ; Data length 8 bits and reception clear error flag
                                ; Enables the transmission/reception operation
MOV    I:SSR0, #00000010B    ; sending interrupt disabled, receiving interrupt
                                ; enabled
MOV    I:SODR0, #13H        ; Transmission data writing
MOV    ILM, #07H            ; Sets ILM in PS to level 7
OR     CCR, #40H            ; Interruption permission
LOOP:  MOV    A, #00H        ; infinite loop
      MOV    A, #01H
      BRA    LOOP
;-----Interrupt Program-----
WARI:
      MOV    A, SIDR0        ; Reading receive data
      CLRB   I:REC          ; The reception interrupt request flag is clear.
;      :
;      User processing
;      :
      RETI                  ; Returns from interrupt.
CODE   ENDS
;-----Vector Settings-----
VECT    CSEG    ABS=0FFH
      ORG    0FF60H        ; The vector is set in interruption #39(27H).
      DSL    WARI
      ORG    0FFDCH        ; Reset vector setting
      DSL    START
      DB     00H          ; Set to Single-chip mode
      VECT    ENDS

```



# **CHAPTER 22**

---

## ***I<sup>2</sup>C INTERFACE***

**This chapter gives an overview of I<sup>2</sup>C interface, the configuration and functions of registers, and operations of I<sup>2</sup>C interface.**

22.1 I<sup>2</sup>C Interface Outline

22.2 I<sup>2</sup>C Interface Register

22.3 I<sup>2</sup>C Interface Operation

## **22.1 I<sup>2</sup>C Interface Outline**

---

**I<sup>2</sup>C interface is Serial I/O by which Inter IC BUS is supported. Operates as the master/slave devices on I<sup>2</sup>C bus.**

---

### **■ I<sup>2</sup>C Interface Function**

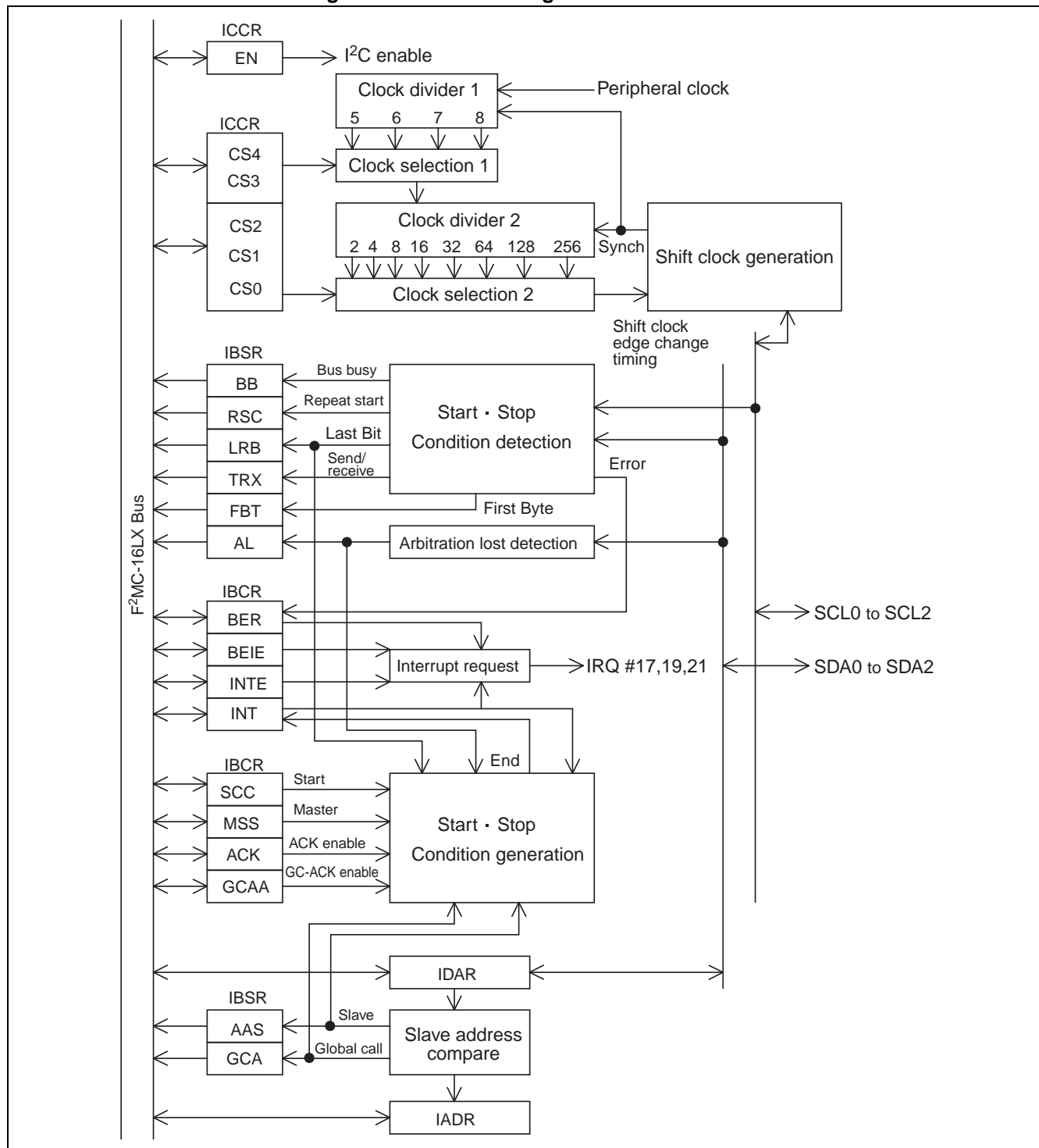
The I<sup>2</sup>C interface has the following functions:

- Master/slave sending and receiving
- Arbitration function
- Clock synchronous function
- Slave address/General call address detection function
- Forwarding direction detection function
- Repetitive generation and detection function of the start condition
- Bus error detection function

## ■ Block Diagram of I<sup>2</sup>C Interface

Figure 22.1-1 shows the block diagram of I<sup>2</sup>C interface.

Figure 22.1-1 Block Diagram of I<sup>2</sup>C Interface





## 22.2 I<sup>2</sup>C Interface Register

The configuration and functions of registers used in the I<sup>2</sup>C interface are described.

### ■ Register List of I<sup>2</sup>C Interface

Figure 22.2-1 Register List of I<sup>2</sup>C Interface

bit 7 6 5 4 3 2 1 0								IBSR0 to IBSR2
ch.0:000070 <sub>H</sub>								I <sup>2</sup> C Bus status register
ch.1:000076 <sub>H</sub>	BB	RSC	AL	LRB	TRX	AAS	GCA	Initial value 00000000 <sub>B</sub>
ch.2:00007C <sub>H</sub>	R	R	R	R	R	R	R	Read/Write
bit 15 14 13 12 11 10 9 8								IBCR0 to IBCR2
ch.0:000071 <sub>H</sub>								I <sup>2</sup> C Bus status register
ch.1:000077 <sub>H</sub>	BER	BEIE	SCC	MSS	ACK	GCAA	INTE	Initial value 00000000 <sub>B</sub>
ch.2:00007D <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Read/Write
bit 7 6 5 4 3 2 1 0								ICCR0 to ICCR2
ch.0:000072 <sub>H</sub>	—	—	EN	CS4	CS3	CS2	CS1	I <sup>2</sup> C Clock control register
ch.1:000078 <sub>H</sub>	—	—	EN	CS4	CS3	CS2	CS1	Initial value --0XXXXXB
ch.2:00007E <sub>H</sub>	—	—	R/W	R/W	R/W	R/W	R/W	Read/Write
bit 15 14 13 12 11 10 9 8								IADR0 to IADR2
ch.0:000073 <sub>H</sub>	—	A6	A5	A4	A3	A2	A1	I <sup>2</sup> C Bus address register
ch.1:000079 <sub>H</sub>	—	A6	A5	A4	A3	A2	A1	Initial value -XXXXXXXX <sub>B</sub>
ch.2:00007F <sub>H</sub>	—	R/W	R/W	R/W	R/W	R/W	R/W	Read/Write
bit 7 6 5 4 3 2 1 0								IDAR0 to IDAR2
ch.0:000074 <sub>H</sub>	D7	D6	D5	D4	D3	D2	D1	I <sup>2</sup> C Bus data register
ch.1:00007A <sub>H</sub>	D7	D6	D5	D4	D3	D2	D1	Initial value XXXXXXXXB
ch.2:000080 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Read/Write

R/W : Readable/Writable  
R : Read only

## 22.2.1 I<sup>2</sup>C Bus Status Register 0 to 2 (IBSR0 to IBSR2)

The configuration and functions of I<sup>2</sup>C bus status register 0 to 2 (IBSR0 to IBSR2) are described.

### ■ I<sup>2</sup>C Bus Status Register 0 to 2 (IBSR0 to IBSR2)

Figure 22.2-2 shows the bit configuration of I<sup>2</sup>C bus status registers 0 to 2 (IBSR0 to IBSR2).

**Figure 22.2-2 Bit Configuration of I<sup>2</sup>C Bus Status Register 0 to 2 (IBSR0 to IBSR2)**

ch.0:000070 <sub>H</sub>	bit	7	6	5	4	3	2	1	0	IBSR0 to IBSR2
ch.1:000076 <sub>H</sub>		BB	RSC	AL	LRB	TRX	AAS	GCA	FBT	I <sup>2</sup> C Bus status register
ch.2:00007C <sub>H</sub>		R	R	R	R	R	R	R	R	Initial value
										00000000B

R : Read only

The function of each bit of the I<sup>2</sup>C bus status registers 0 to 2 (IBSR0 to IBSR2) is described as follows:

#### [bit 7] BB: Bus Busy

It is a bit shown the state of the I<sup>2</sup>C bus.

0	Stop condition is detected.
1	Start condition is detected. (The bus is used.)

#### [bit 6] RSC: Repeated Start Condition

It is a start condition detection repeatedly bit.

0	The start condition is not repeatedly detected.
1	The start condition was detected in the bus Occupied again.

Is cleared either by writing "0" in INT bit, with no addressing on the slave connection, or by detecting the start condition during the halted bus, or by detecting the stop condition.

#### [bit 5] AL: Arbitration Lost

It is an arbitration lost detection bit.

0	The arbitration lost is not detected.
1	When arbitration lost has occurred on master transmission, or when other systems are using the bus, "1" was written in MSS bit.

It is cleared by writing "0" in INT bit.

**[bit 4] LRB: Last Received bit**

It is the acknowledgement storage bit. Stores the acknowledge from the receiver.

0	Reception is acknowledged.
1	Reception is not acknowledged.

It is cleared by detecting the start condition or the stop condition.

**[bit 3] TRX: Transfer/Receive**

It is the bit indicating sending and receiving of data transfer.

0	Receiving state
1	Transmission state

**[bit 2] AAS: Addressed As Slave**

It is Addressing detection bit.

0	Addressing is not done at the slave.
1	Addressing was done at the slave.

It is cleared by detecting of the start condition or stop condition.

**[bit 1] GCA: General Call Address**

It is General call address (00<sub>H</sub>) detection bit.

0	The General call address is not received at the slave.
1	The General call address was received at the slave.

It is cleared by detecting of the start condition or stop condition.

**[bit 0] FBT: First Byte Transfer**

It is a first byte detection bit.

0	Received data is things except the first byte.
1	Received data is a first byte (address data).

It is cleared either if "0" is written in INT bit or if no addressing is provided on the slave connection, even though detection of the start condition has set "1".

## 22.2.2 I<sup>2</sup>C Bus Control Register 0 to 2 (IBCR0 to IBCR2)

The configuration and functions of I<sup>2</sup>C bus control register 0 to 2 (IBCR0 to IBCR2) are described.

### ■ I<sup>2</sup>C Bus Control Register 0 to 2 (IBCR0 to IBCR2)

Figure 22.2-3 shows the bit configuration of bus control register 0 to 2 (IBCR0 to IBCR2).

**Figure 22.2-3 Bit Configuration of I<sup>2</sup>C Bus Control Register 0 to 2 (IBCR0 to IBCR2)**

bit 15 14 13 12 11 10 9 8								IBCR0 to IBCR2	
ch.0:000071 <sub>H</sub>	BER	BEIE	SCC	MSS	ACK	GCAA	INTE	INT	I <sup>2</sup> C Bus status register
ch.1:000077 <sub>H</sub>									Initial value 00000000 <sub>B</sub>
ch.2:00007D <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Read/Write
R/W : Readable/Writable									

The function of each bit of the bus control registers 0 to 2 (IBCR0 to IBCR2) is described as follows:

#### [bit 15] BER: Bus Error

It is Bus error Interrupt request flag. Functions in writing phase differ from those as follows.

(at writing)

0	Clear bus error interrupt request flag.
1	No effect on operation

(at reading)

0	The bus error is not detected.
1	Illegal start and stop condition was detected during data transfer.

If BER bit is set, EN bit of ICCR register is cleared, the I<sup>2</sup>C interface goes into a halted state, and data transfer is terminated.

#### [bit 14] BEIE: Bus Error Interrupt Enable

It is bus error Interrupt enable bit.

0	Bus error interrupt disabled
1	Bus error interrupt enabled

The interruption is generated if the BER bit is "1" when this bit is "1".

**[bit 13] SCC: Start Condition Continue**

It is a start condition generation bit.

(at writing)

0	No effect on operation
1	A start condition is regenerated on master transmission.

This bit is always "0" at the beginning of reading.

**[bit 12] MSS: Master Slave Select**

It is Master/slave selection bit.

0	Generates the stop condition, and goes into the slave mode after the completion of transmission.
1	Goes into the master mode, generates the start condition, and starts transferring.

If the arbitration lost has occurred during the master sending, it will be cleared and will switch to the slave mode.

---

**Note:**

The transmission of the general call address is prohibited, because the receiving as the slave cannot be operation on the following condition.

- When (1) other LSI that is a master mode without this LSI exists on the bus, and (2) this LSI transmits the general call address as master, and (3) the arbitration lost is generated since the second byte.

---

**[bit 11] ACK: ACKnowledge**

It is the acknowledge generation enable bit when the data is received.

0	Acknowledgement is disabled to be generated.
1	Acknowledgement is enabled to be generated.

Is invalid during receiving address data in the slave.

**[bit 10] GCAA: General Call Address Acknowledge**

It is an acknowledge generating enable bit, when you receive the General call address.

0	Acknowledgement is disabled to be generated
1	Acknowledgement is enabled to be generated.

**[bit 9] INTE: INTerrupt Enable**

It is Interrupt enable bits.

0	Disables the interrupt.
1	Enables the interrupt.

The interruption is generated if the INT bit is "1" when this bit is "1".

**[bit 8] INT: INTerrupt**

It is transfer stop interrupt request flag bit.

(at writing)

0	Clear Transfer stop interrupt request flag.
1	No effect on operation

(at reading)

0	Transfer has not been finished yet.
1	<p>This is set when 1 byte of data including the ACK bit has already been transferred and if the following condition is applied.</p> <ul style="list-style-type: none"> <li>• It is a bus master.</li> <li>• It is a slave that the address is done.</li> <li>• The General call address was received.</li> <li>• The arbitration lost happened.</li> <li>• Other systems tried to generate a start condition while the bus was in use.</li> </ul>

The SCL line is kept at "L" level at "1". Is cleared by writing "0", releases SCL line, and sends the subsequent bite. And is reset to "0" by occurrence of the start condition or the stop condition on the master connection.

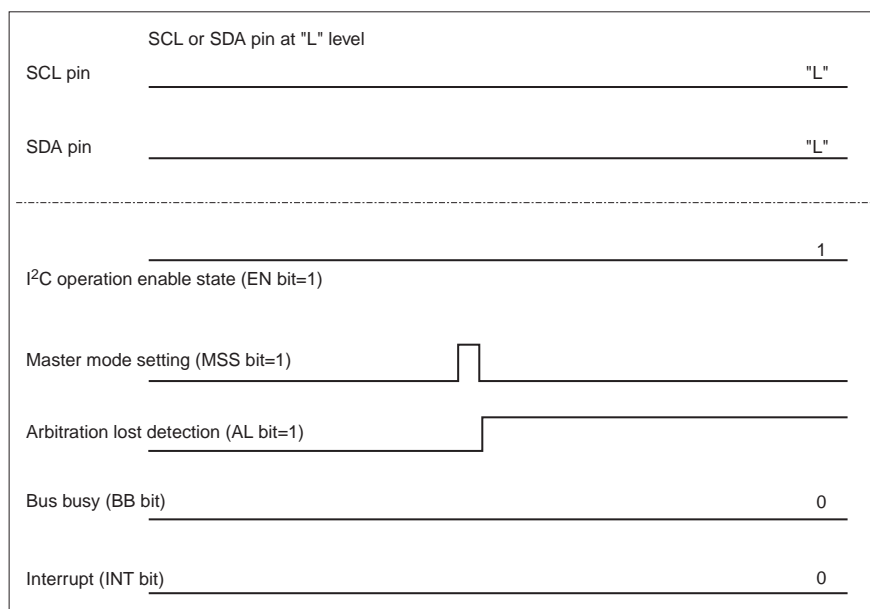
**Notes:**

When an instruction which generates a start condition is executed (the MSS bit is set to "1") at the timing shown in Figure 22.2-4 and Figure 22.2-5, arbitration lost detection (AL bit=1) prevents an interrupt (INT bit=1) from being generated.

- Condition 1 in which an interrupt (INT bit=1) upon detection of "AL bit=1" does not occur.

When an instruction which generates a start condition is executed (setting the MSS bit in the IBCR register to "1") with no start condition detected (BB bit=0) and with the SDA or SCL pin at the "L" level.

**Figure 22.2-4 Diagram of Timing at which an Interrupt upon Detection of "AL bit=1" does not Occur**

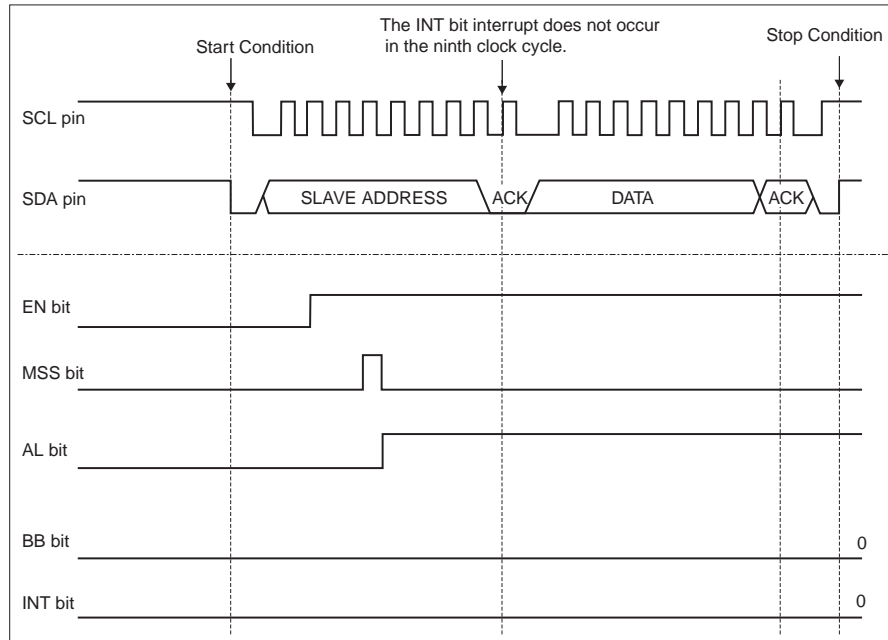


- Condition 2 in which an interrupt (INT bit=1) upon detection of "AL bit=1" does not occur

When an instruction which generates a start condition by enabling I<sup>2</sup>C operation (EN bit=1) is executed (setting the MSS bit in the IBCR register to "1") with the I<sup>2</sup>C bus occupied by another master.

This is because, as shown in Figure 22.2-5, when the other master on the I<sup>2</sup>C bus starts communication with I<sup>2</sup>C disabled (EN bit=0), the I<sup>2</sup>C bus enters the occupied state with no start condition detected (BB bit=0).

**Figure 22.2-5 Diagram of Timing at which an Interrupt upon Detection of "AL bit=1" does not Occur**



If a symptom as described above can occur, follow the procedure below for software processing.

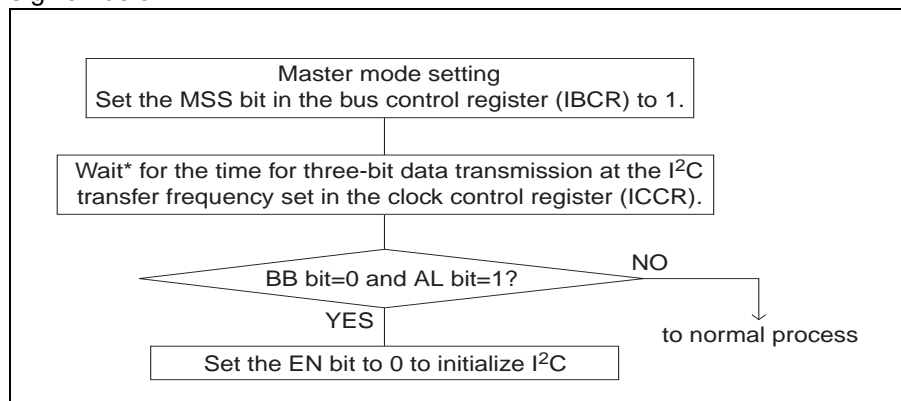
1. Execute the instruction that generates a start condition (set the MSS bit to "1").
2. Use, for example, the timer function to wait for the time\* for three-bit data transmission at the I<sup>2</sup>C transfer frequency set in the ICCR register.

Example: Time for three-bit data transmission at an I<sup>2</sup>C transfer frequency of 100 kHz

$$\{1/(100 \times 10^3)\} \times 3 = 30 \mu\text{s}$$

3. Check the AL and BB bits in the IBSR register and, if the AL and BB bits are "1" and "0", respectively, set the EN bit in the ICCR register to "0" to initialize I<sup>2</sup>C. When the AL and BB bits are not so, perform normal processing.

A sample flow is given below.

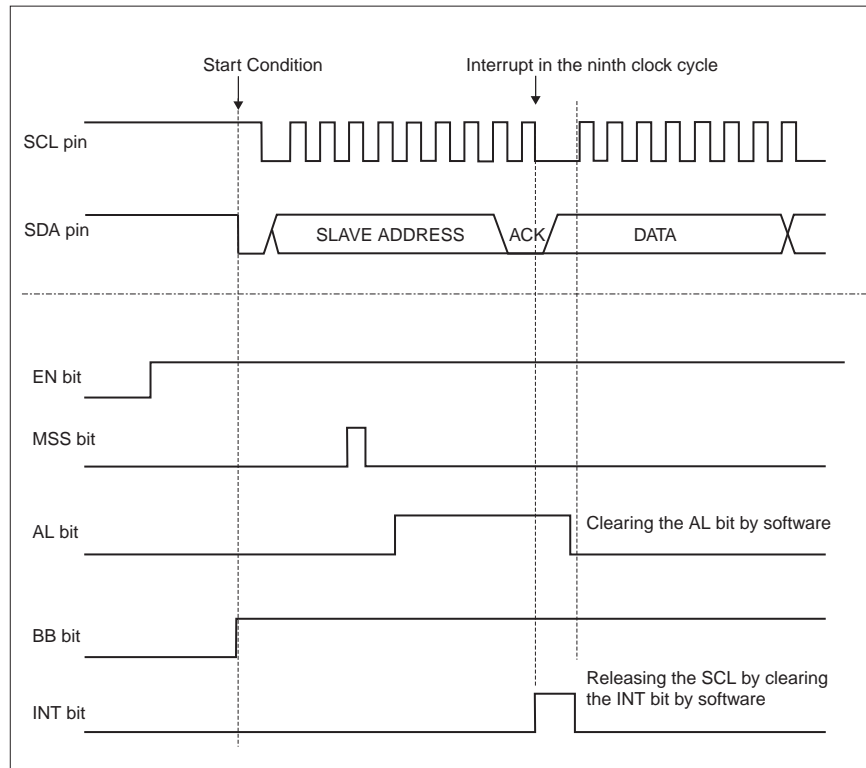


\*: When "arbitration lost" is detected, the MSS bit is set to "1" and then the AL bit is set to "1" without failure after the time for three-bit data transmission at the I<sup>2</sup>C transfer frequency.

- Example of occurrence of an interrupt (INT bit=1) upon detection of "AL bit=1"

When an instruction which generates a start condition is executed (setting the MSS bit to "1") with "bus busy" detected (BB bit=1) and arbitration is lost, the INT bit interrupt occurs upon detection of "AL bit=1".

**Figure 22.2-6 Diagram of Timing at which an Interrupt upon Detection of "AL bit=1" Occurs**



## ■ Notes on Use of I<sup>2</sup>C Bus Control Register 0 to 2 (IBCR0 to IBCR2)

The following care should be taken to conflicts among SCC bit, MSS bit, and INT bit.

Writing to SCC bit, MSS bit, INT bit simultaneously causes conflicts of transferring the next bite, occurrence of the start condition, and occurrence of the stop condition. The priority in this case is as follows.

- The following byte forwarding and stop condition generation

If "0" is written in INT bit and "0" in MSS bit, writing "0" in MSS bit is preferred and the stop condition occurs.

- The following byte forwarding and start condition generation

If "0" is written in INT bit and "1" in SCC bit, writing "1" in SCC bit is preferred and the start condition occurs.

- Start condition generation and stop condition generation

Simultaneous writing is disabled that not only "1" is written in SCC bit but also "0" is written in MSS bit.



### 22.2.3 I<sup>2</sup>C Bus Clock Control Register 0 to 2 (ICCR0 to ICCR2)

The configuration and functions of I<sup>2</sup>C bus clock control register 0 to 2 (ICCR0 to ICCR2) are described.

#### ■ I<sup>2</sup>C Bus Clock Control Register 0 to 2 (ICCR0 to ICCR2)

Figure 22.2-7 shows the bit configuration of I<sup>2</sup>C bus clock control registers 0 to 2 (ICCR0 to ICCR2).

Figure 22.2-7 Bit Configuration of I<sup>2</sup>C Bus Clock Control Register 0 to 2 (ICCR0 to ICCR2)

ch.0:000072 <sub>H</sub>	bit	7	6	5	4	3	2	1	0	ICCR0 to ICCR2
ch.1:000078 <sub>H</sub>		—	—	EN	CS4	CS3	CS2	CS1	CS0	I <sup>2</sup> C Clock control register
ch.2:00007E <sub>H</sub>		—	—	R/W	R/W	R/W	R/W	R/W	R/W	Initial value XX0XXXXXB
R/W : Readable/Writable										

The functions of I<sup>2</sup>C bus clock control registers 0 to 2 (ICCR0 to ICCR2) are described below.

#### [bit 7, bit 6] undefinition bit

The read value is irregular. Nothing is affected when it is written.

#### [bit 5] EN: ENable

It is an operation permission bit in the I<sup>2</sup>C interface.

0	Operation disabled
1	Operation enabled

- When "0", each bit of IBSR register and IBCR register (except BER and BEIE bits) is cleared.
- Is cleared when BER bit is set.

#### Notes:

- If the I<sup>2</sup>C interface operation is inhibited, transmit/receive operation is immediately stopped.
- If you wish to inhibit the I<sup>2</sup>C interface operation after generating the stop condition by writing "0" to the MSS bit, confirm the stop condition is generated (BB=0 for IBSR) before inhibiting the operation (EN=0 for ICCR).

[bit 4 to bit 0] CS4 to CS0:Clock Period Select 4-0

It is the bit which sets the serial clock frequency. The frequency f<sub>sck</sub> in the shift clock is set as shown in the next formula.

$$f_{sck} = \frac{\phi}{m \times n + 4} \quad \phi: \text{Machine clock}$$

Note:

The cycle + 4 is minimum overhead for checking that the output level of SCL pin has changed. If delay is longer on the rising edge of SCL pin, or a slave device delays a clock, it exceeds this value. Note that the frequency of the serial clock must be set to 100 kHz or less.

m and n for CS4 to CS0 is as shown in Table 22.2-1.

**Table 22.2-1 Setting of Serial Clock Frequency**

m	CS4	CS3	n	CS2	CS1	CS0
5	0	0	4	0	0	0
6	0	1	8	0	0	1
7	1	0	16	0	1	0
8	1	1	32	0	1	1
			64	1	0	0
			128	1	0	1
			256	1	1	0
			512	1	1	1

22.2.4 I<sup>2</sup>C Bus Address Register 0 to 2 (IADR0 to IADR2)

The configuration and functions of I<sup>2</sup>C bus address register 0 to 2 (IADR0 to IADR2) are described.

■ I<sup>2</sup>C Bus Address Register 0 to 2 (IADR0 to IADR2)

Figure 22.2-8 shows the bit configuration of the I<sup>2</sup>C bus address registers 0 to 2 (IADR0 to IADR2).

Figure 22.2-8 I<sup>2</sup>C Bus address Register 0 to 2 (IADR0 to IADR2)

ch.0:000073 <sub>H</sub>	bit	15	14	13	12	11	10	9	8	
ch.1:000079 <sub>H</sub>		—	A6	A5	A4	A3	A2	A1	A0	IADR0 to IADR2
ch.2:00007F <sub>H</sub>		—	R/W	R/W	R/W	R/W	R/W	R/W	R/W	I <sup>2</sup> C Bus address register
										Initial value XXXXXXXXB

R/W : Readable/Writable

[bit 14 to bit 8] A6 to A0

It is a slave address bit. It is a register which specifies the slave address. In the slave, the address data is compared to the IDAR register upon reception of the address, and if they match, then it will send the acknowledge to the master.

## 22.2.5 I<sup>2</sup>C Bus Data Register 0 to 2 (IDAR0 to IDAR2)

The configuration and functions of I<sup>2</sup>C bus data register 0 to 2 (IDAR0 to IDAR2) are described.

### ■ I<sup>2</sup>C Bus Data Register 0 to 2 (IDAR0 to IDAR2)

Figure 22.2-9 shows the bit configuration of the I<sup>2</sup>C bus data registers 0 to 2 (IDAR0 to IDAR2).

**Figure 22.2-9 Bit Configuration of I<sup>2</sup>C Bus Data Register 0 to 2 (IDAR0 to IDAR2)**

ch.0:000074 <sub>H</sub>	bit	7	6	5	4	3	2	1	0	IDAR0 to IDAR2
ch.1:00007A <sub>H</sub>		D7	D6	D5	D4	D3	D2	D1	D0	I <sup>2</sup> C Bus data register
ch.2:000080 <sub>H</sub>		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Initial value XXXXXXXXB

R/W : Readable/Writable

#### [bit 7 to bit 0] D7 to D0

It is a data bit.

It is the data register used for the serial transfer, and transferred from MSB. During the data receiving (TRX= 0), the data output value becomes "1".

The writing side of IDAR register is double buffering. If the bus is in use (BB=1), the data to be written to is loaded to the register for serial transferring when each byte is transferred. In reading the register for serial transferring is directly read, so the receiving data is valid only when INT bit is set.

## 22.3 I<sup>2</sup>C Interface Operation

---

For I<sup>2</sup>C bus, 1 serial data line (SDA), 1 serial clock line (SCL), and 2 bi-directional bus lines are responsible for communication. I<sup>2</sup>C interface, which has 2 open drain input-output pins (SDA and SCL) to them, allows wired logic.

---

### ■ Start Condition

If "1" is written to MSS bit in the state of the bus being released (BB=0 and MSS=0), I<sup>2</sup>C interface generates the start condition as soon as it changes to the master mode. In the master mode the start condition can be regenerated by writing "1" to SCC bit even though the bus is in use (BB=1). To generate start conditions, 2 methods are provided as follows.

- Writing "1" to MSS bit in the state that the bus is not in use (MSS=0 \* BB=0 \* INT=0 \* AL=0).
- Writing "1" to SEC bit in the state of interrupting in bus master (MSS=1 \* BB=1 \* INT=1 \* AL=0).

When the bus (in the idle state) is used by other systems and if "1" is written to MSS bit, then AL bit is set to "1". Writing to MSS bit and SCC bit in any other state than being mentioned above is ignored.

### ■ Stop Condition

When "0" is written to MSS bit in the state of the master mode (MSS=1), the stop condition is generated, resulting in the slave mode. The condition to generate the stop condition is as follows.

Writing "0" to MSS bit in the state of interrupting in bus master (MSS=1 \* BB=1 \* INT=1 \* AL=0).

Writing "0" excluding this in the MSS bit is disregarded.

### ■ Addressing

In the master mode, BB= 1 and TRX= 1 will be set after the start condition generation, and it outputs the IDAR register contents from the MSB. After sending the address data, receives the acknowledgement from the slave, reverses bit 0 (bit 0 of IDAR register that is already sent) of the sending data, and then stores it into TRX bit.

In the slave mode, BB is set to "1" and TRX is set to "0" after the start condition is generated, and the sending data from the master is received in IDAR register. After the address data is received, IDAR register is compared with IADR register. If they matches, AAS is set to "1" and the acknowledgement is sent to the master. Then, bit 0 of the receiving data (bit 0 of IDAR register that is already received) is stored in TRX bit.

### ■ Arbitration

If other master are sending the data simultaneously in the master sending mode, the arbitration will occur. When the sending data of its own is "1", and the data on SDA line is "L" level, it is considered that its arbitration is lost and AL is set to "1". When an attempt is made that the start condition is generated in the state the bus is in use as mentioned above, Al is also set to "1". When AL is set to "1", MSS is "0" and TRX is "0", resulting in the slave receiving mode.

## ■ Acknowledge

The receiver send the acknowledge to the sender. During data reception, ACK bit can specify acknowledgement is necessary or not. During the data sending, the acknowledge from the receiver is stored in the LRB bit.

If the sender as slave does not receive any acknowledgement from the receiver as master, TRX becomes "0", resulting in the slave receiving mode. This allows the master to generate the stop condition when the slave releases the SCL line.

## ■ Bus Error

If the following conditions exist, it will be considered as bus error, and the I<sup>2</sup>C interface will be in the stopped state.

- Detecting of violation of the basic rules on I<sup>2</sup>C bus during data transfer (including ACK bit)
- Stop condition detection at master
- Detecting of violation of the basic rules on I<sup>2</sup>C bus in bus idle.

## ■ The Others

### ● Processing after arbitration lost is detected

After arbitration lost occurs, must decide if it is addressed or not by software.

Once arbitration lost occurs, it becomes slave from the viewpoint of hardware. After one byte of data transfer both CLK line and DATA line are pulled to "L". For this reason, if addressing is done, CLK line and DATA line should be released after slave transmission or slave reception is ready. if no addressing is done, then CLK line and DATA line should be immediately released. (Software is responsible for this all.)

### ● Interruption factor when arbitration lost is detected

When arbitration lost is detected, causes of interrupts are not issued immediately but after one byte of data has been transferred.

When arbitration lost is detected, it becomes slave from the viewpoint of hardware. Even if so, it outputs 9 clocks in all in order to issue causes of interrupts. For this reason, causes and interrupts are not immediately issued, so no processing is allowed after arbitration lost.

### ● Interrupt condition

It is specified that I<sup>2</sup>C bus has one interrupt and causes of interrupts are issued once one byte of transfer completes or if the conditions of interrupts are satisfied.

Each flag should be checked within the interrupt routine, since multiple conditions of interrupts is identified based on one interrupt. Multiple conditions of interrupts on completion of one byte transfer is as follows.

- When it is a bus master
- When it is a slave that the address is done
- When you receive the General call address
- When arbitration lost has occurred.

### ● Transfer rate

Note that I<sup>2</sup>C bus can support up to 100 kHz of the serial clock frequency of transmission.

22.3.1 Transfer Flow of I<sup>2</sup>C Interface

Figure 22.3-1 shows the 1-byte transfer flow from master to slave, and Figure 22.3-2 shows the 1-byte transfer flow from slave to master.

■ Transfer Flow of I<sup>2</sup>C Interface

Figure 22.3-1 1-byte Transfer Flow from the Master to Slave

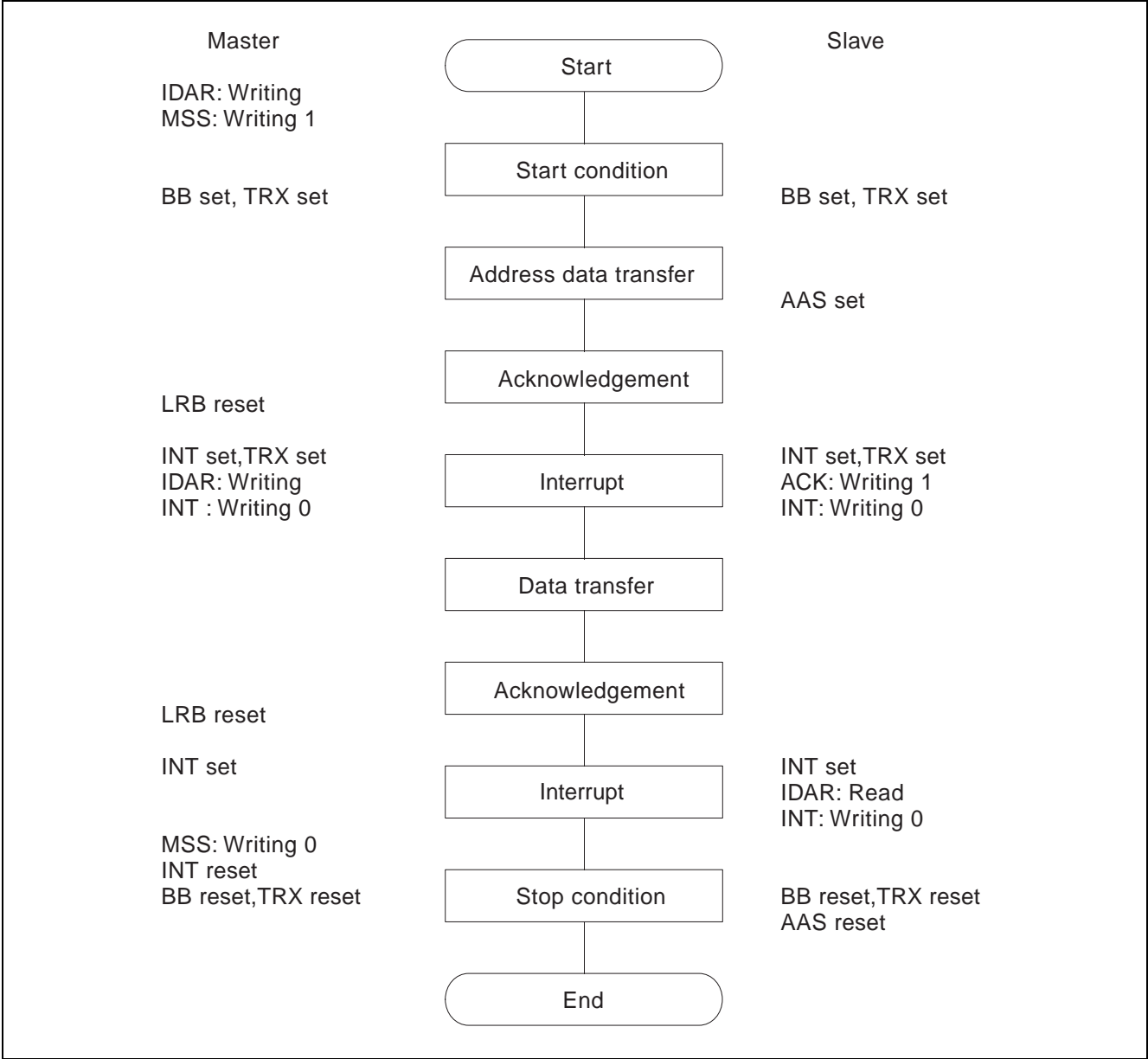
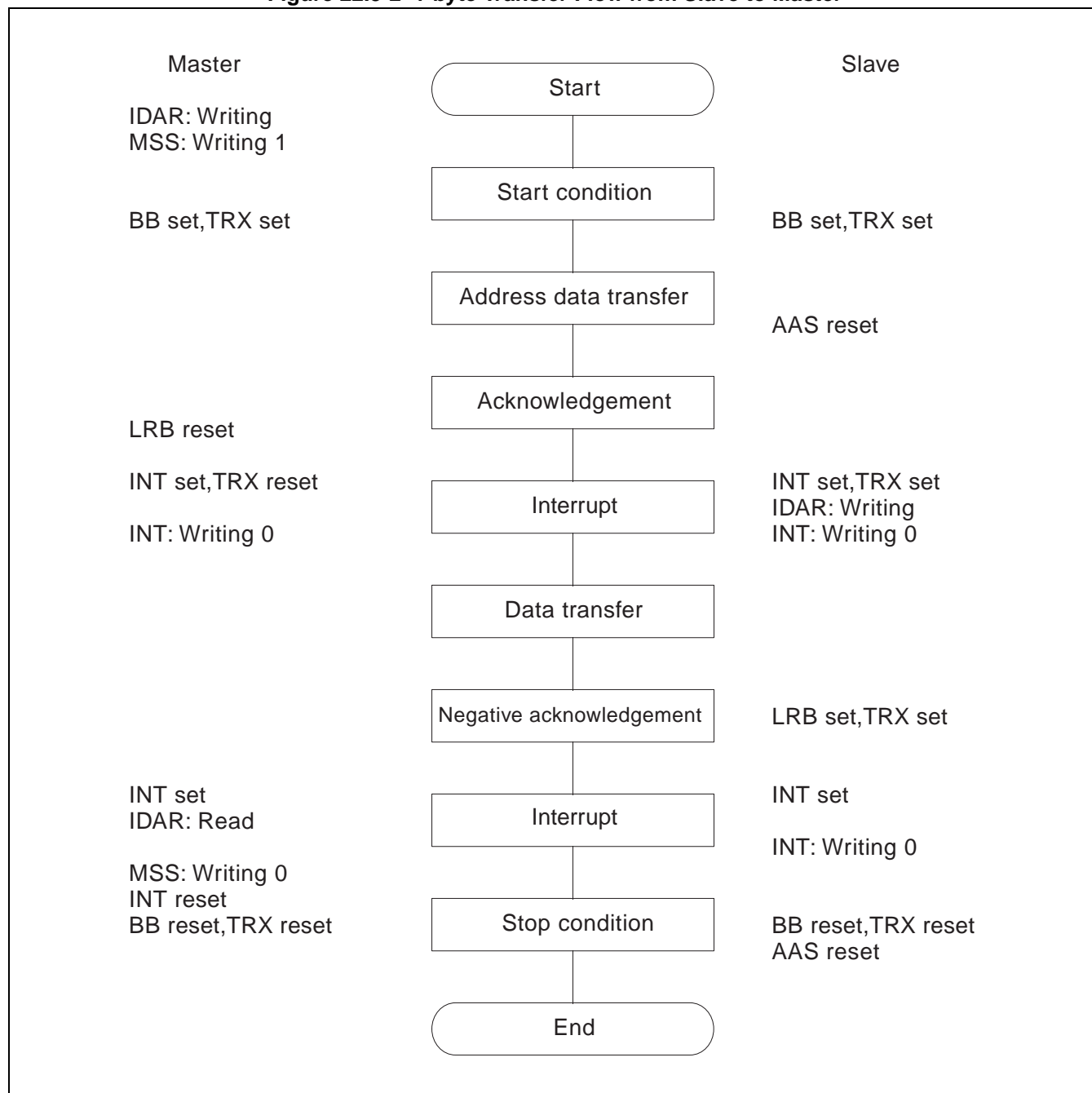


Figure 22.3-2 1-byte Transfer Flow from Slave to Master



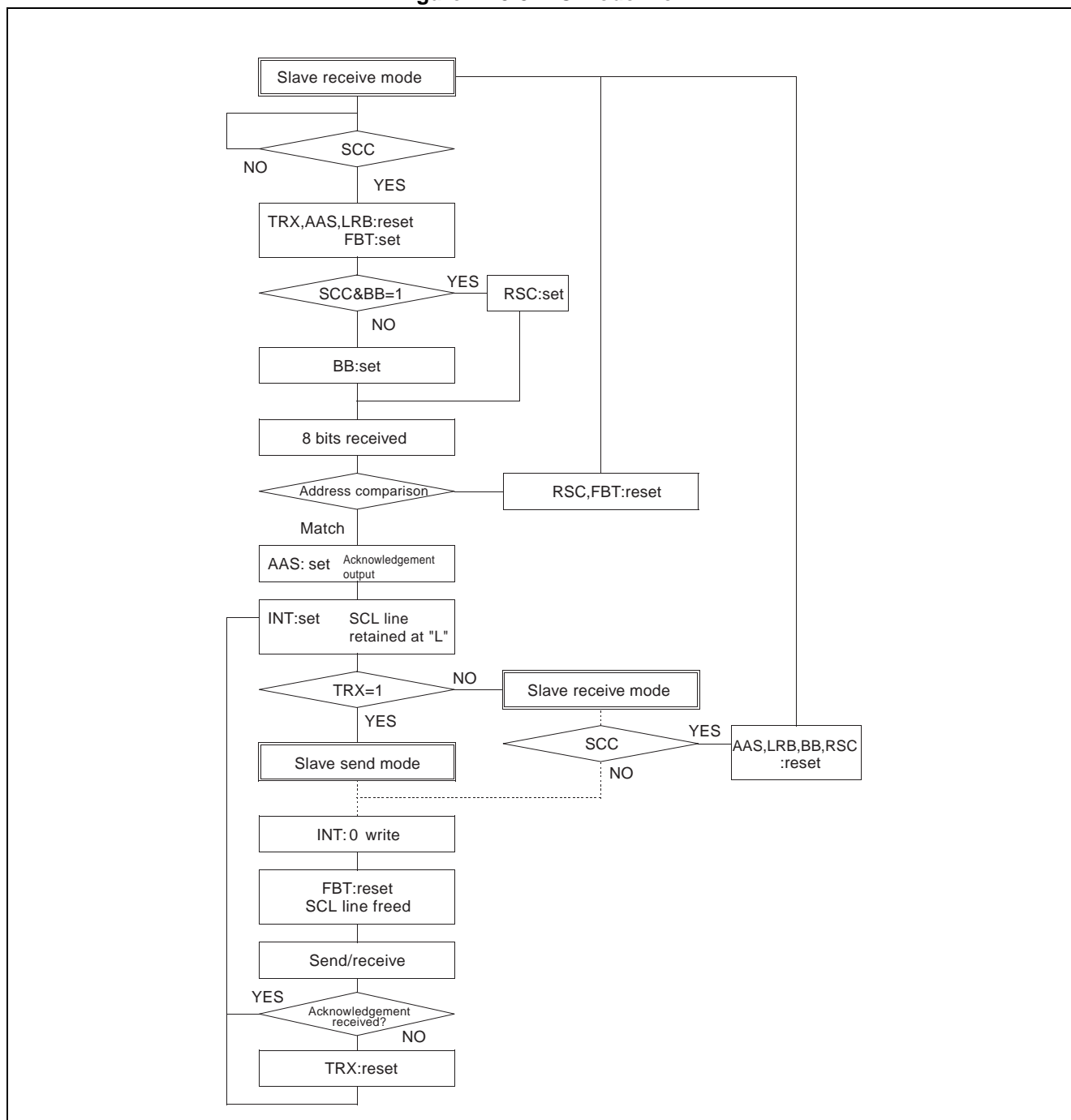


## 22.3.2 Mode Flow of I<sup>2</sup>C Interface

Figure 22.3-3 shows the flow of mode transitions for the I<sup>2</sup>C interface.

### ■ Flow of I<sup>2</sup>C Interface Mode Transitions

Figure 22.3-3 I<sup>2</sup>C Mode Flow

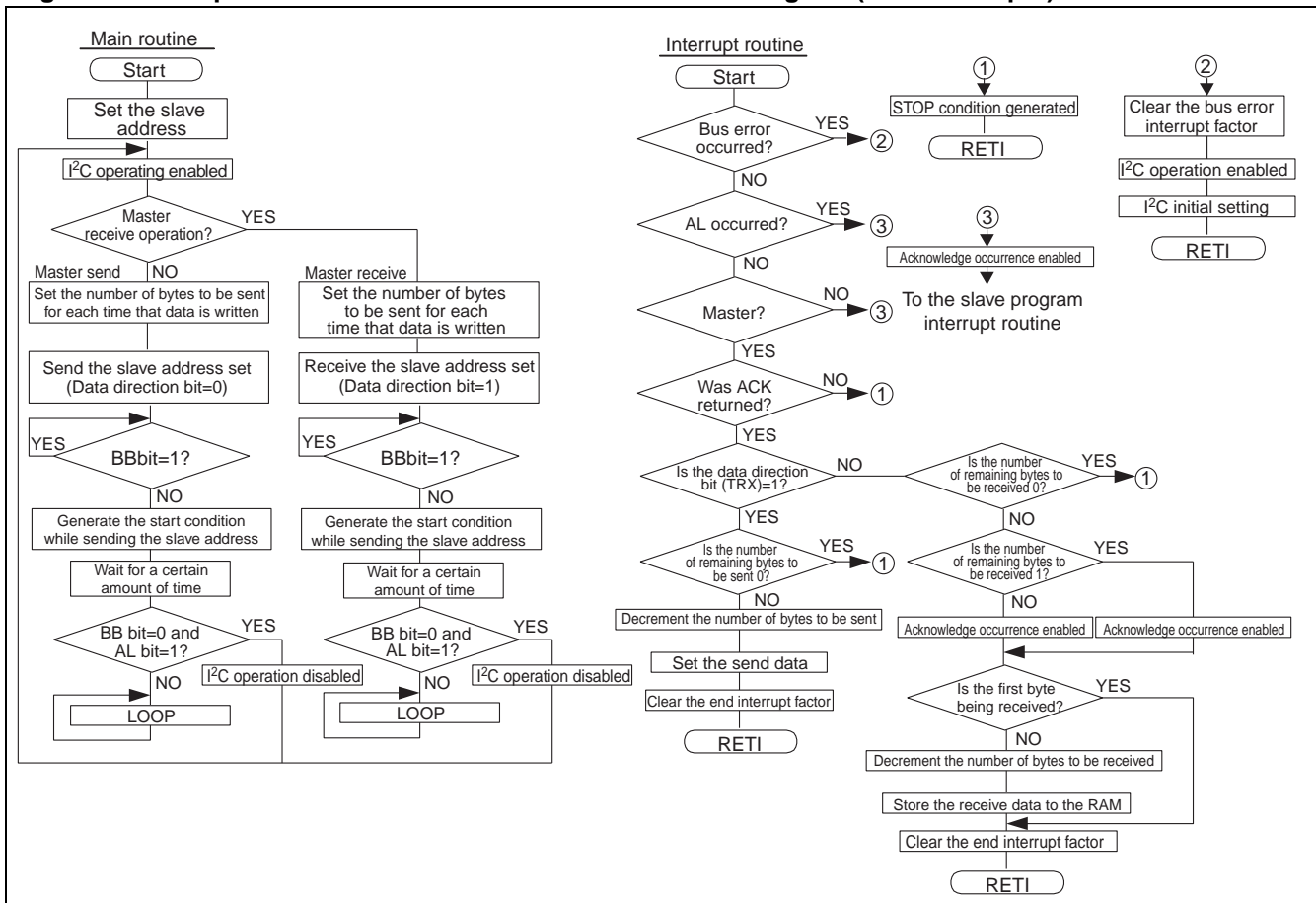


### 22.3.3 Operation Flow of I<sup>2</sup>C Interface

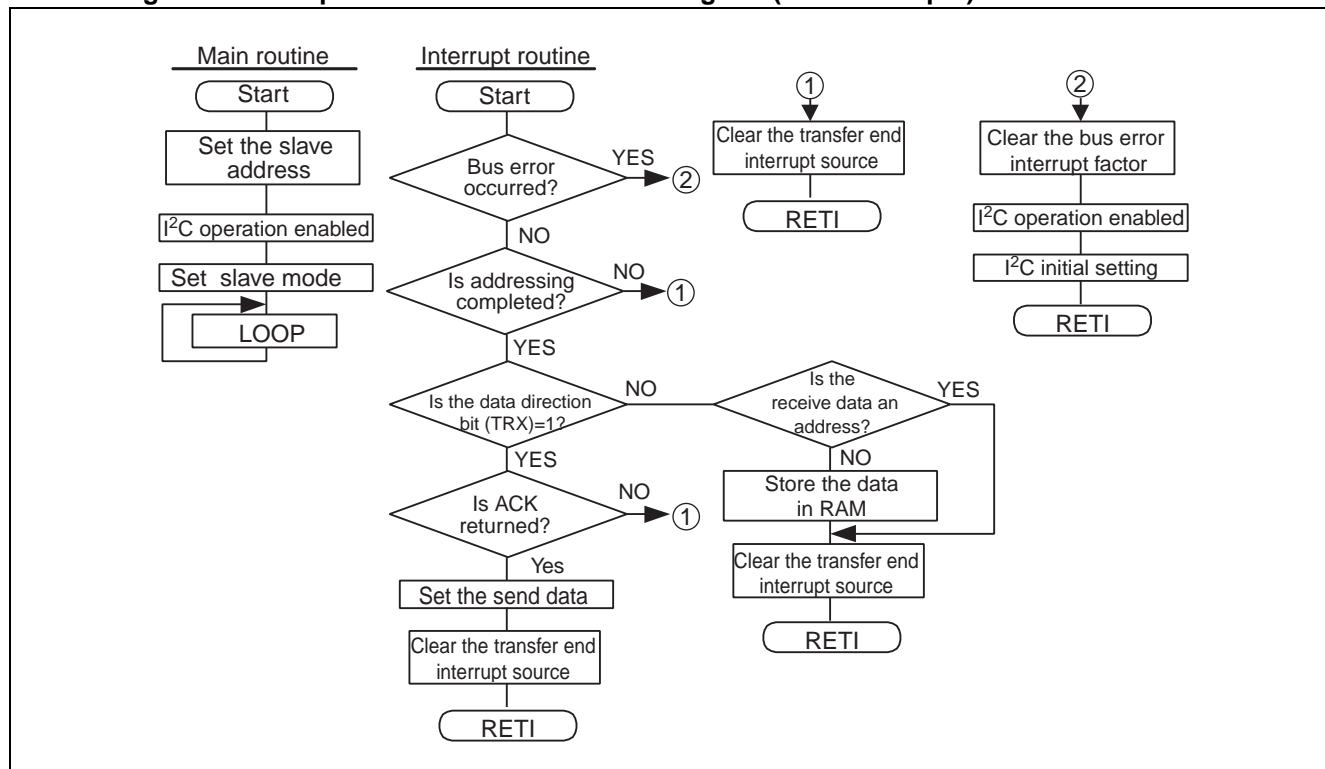
Figure 22.3-4 shows the operation flow of a master send/receive program (with interrupts) for the I<sup>2</sup>C interface. Figure 22.3-5 shows the operation flow of the slave program (with interrupts) for the I<sup>2</sup>C interface.

#### ■ Operation Flow of I<sup>2</sup>C Interface

Figure 22.3-4 Operation Flow of the Master Send/Receive Program (with Interrupts) for the I<sup>2</sup>C Interface



**Figure 22.3-5 Operation Flow of the Slave Program (with Interrupts) for the I<sup>2</sup>C Interface**



# **CHAPTER 23**

---

## ***ROM MIRROR FUNCTION SELECTION MODULE***

**This chapter describes the ROM mirror function selection module.**

23.1 Overview of ROM Mirror Function Select Module

23.2 ROM Mirror Function Select Register (ROMM)

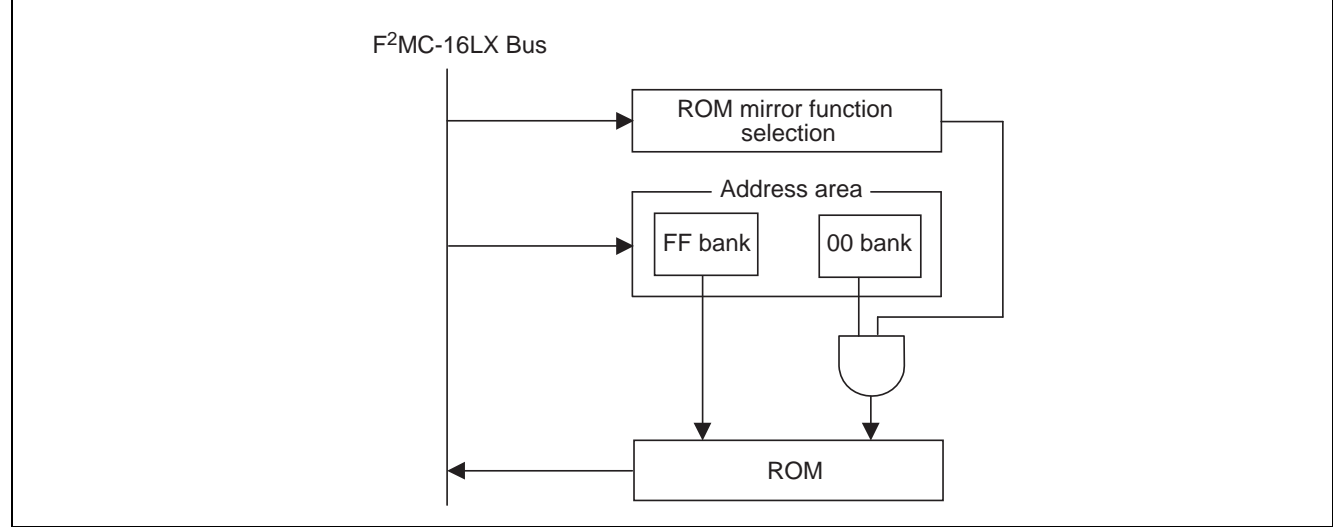
23.1 Overview of ROM Mirror Function Select Module

The ROM mirror function selection module is used to select via register settings an FF bank in ROM, whose contents can be viewed via 00 bank.

■ Block Diagram of ROM Mirror Function Select Module

Figure 23.1-1 shows the block diagram of ROM mirror function selection module.

Figure 23.1-1 Block Diagram of ROM Mirror Function Select Module



■ Register of ROM Mirror Function Selection Module

Figure 23.1-2 shows the configuration of ROM mirror function select module.

Figure 23.1-2 ROM Mirror Function Selection Module Configuration

	bit	15	14	13	12	11	10	9	8	initial value
ROMM Address : 00006FH		—	—	—	—	—	—	Reserved	MI	----- 11 <sub>B</sub>
								R/W	W	

R/W : Readable/Writable  
W : Write only

## MB90330A Series

### 23.2 ROM Mirror Function Select Register (ROMM)

The configuration and functions of ROM Mirror Function Select Register (ROMM) are described.

#### ■ ROM Mirror Function Select Register (ROMM)

Figure 23.2-1 shows the bit configuration of ROM mirror function select register (ROMM).

**Figure 23.2-1 Bit Configuration of ROM Mirror Function Select Register (ROMM)**

	bit	15	14	13	12	11	10	9	8	initial value
ROMM Address : 00006F <sub>H</sub>		—	—	—	—	—	—	Reserved	MI	----- 11 <sub>B</sub>
								R/W	W	
R/W : Readable/Writable										
W : Write only										

#### [bit 9] Reserved bit

It is Reserved bit. Please write "1".

#### [bit 8] MI

- When "1" is written, ROM data in FF bank can be also read in 00 bank.
- When "0" is written, this function is not available to 00 bank.
- Only writing is enabled.

#### Notes:

- Do not access to ROMM register in the middle of the operation of the address 008000<sub>H</sub> to 00FFFF<sub>H</sub>.
- When ROM mirror function is activated, the addresses from FF8000<sub>H</sub> to FFFFFFF<sub>H</sub> are mirrored to the addresses from 008000<sub>H</sub> to 00FFFF<sub>H</sub> of 00 bank. So ROM addresses under the address FF7FFF<sub>H</sub> are not mirrored even though ROM mirror function is set.



# ***CHAPTER 24***

---

# ***ADDRESS MATCH DETECTION FUNCTION***

**This chapter explains the address match detection function and its operation.**

- 24.1 Overview of Address Match Detection Function
- 24.2 Block Diagram of Address Match Detection Function
- 24.3 Configuration of Address Match Detection Function
- 24.4 Explanation of Operation of Address Match Detection Function
- 24.5 Program Example of Address Match Detection Function



## 24.1 Overview of Address Match Detection Function

---

If the address of the instruction to be processed next to the instruction currently processed by the program matches the address set in the detect address setting registers, the address match detection function forcibly replaces the next instruction to be processed by the program with the INT9 instruction to branch to the interrupt processing program. Since the address match detection function can use the INT9 interrupt for instruction processing, the program can be corrected by patch processing.

---

### ■ Overview of Address Match Detection Function

- The address of the instruction to be processed next to the instruction currently processed by the program is always held in the address latch through the internal data bus. The address match detection function always compares the value of the address held in the address latch with that of the address set in the detect address setting registers. When these compared values match, the next instruction to be processed by the CPU is forcibly replaced by the INT9 instruction, and the interrupt processing program is executed.
- There are two detect address setting registers (PADR0 and PADR1), each of which has an interrupt enable bit. The generation of an interrupt due to a match between the address held in the address latch and the address set in the detect address setting registers can be enabled and disabled for each register.

**MB90330A Series****24.2 Block Diagram of Address Match Detection Function**

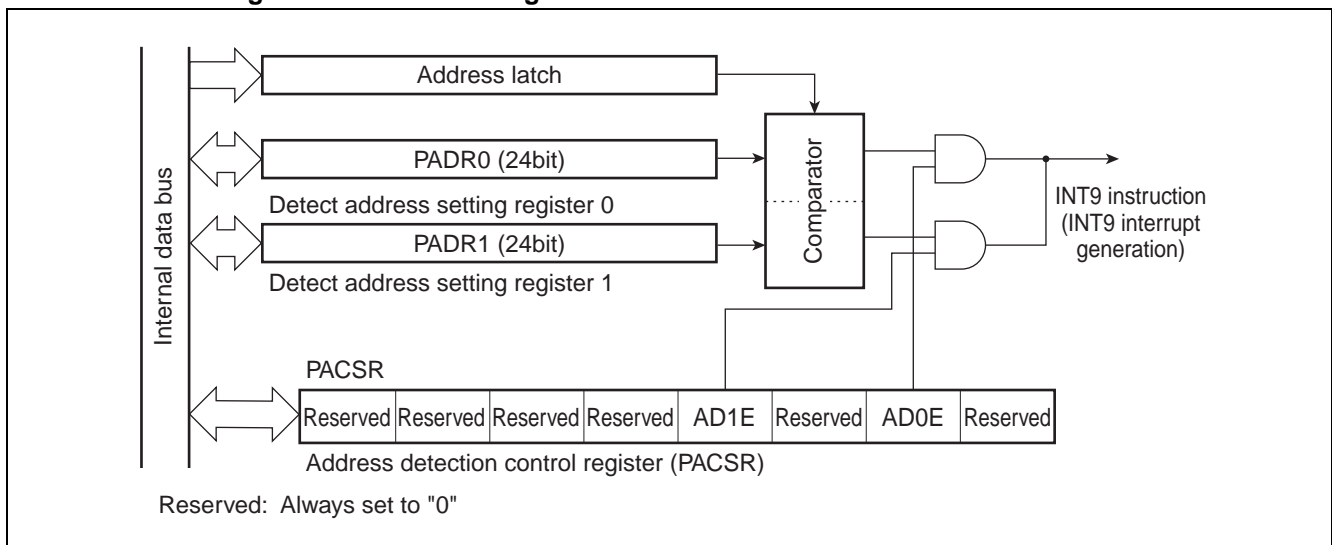
The address match detection module consists of the following blocks:

- Address latch
- Address detection control register (PACSR)
- Detect address setting registers (RADR0, RADR1)

### ■ Block Diagram of Address Match Detection Function

Figure 24.2-1 shows the block diagram of the address match detection function.

**Figure 24.2-1 Block Diagram of the Address Match Detection Function**



#### ● Address latch

The address latch stores the value of the address output to the internal data bus.

#### ● Address detection control register (PACSR)

The address detection control register enables or disables output of an interrupt at an address match.

#### ● Detect address setting registers (PADR0, PADR1)

The detect address setting registers set the address that is compared with the value of the address latch.

#### Note:

The addresses of the detect address setting register are 1FF0<sub>H</sub> to 1FF5<sub>H</sub> and are included in the RAM area. Therefore, the access to the RAM area should not be performed during the use of this function.

## 24.3 Configuration of Address Match Detection Function

This section details the registers used by the address match detection function.

### ■ List of Registers and Reset Values of Address Match Detection Function

Figure 24.3-1 List of Registers and Reset Values of Address Match Detection Function

		bit	7	6	5	4	3	2	1	0
Address detection control register (PACSR)			0	0	0	0	0	0	0	0
Address : 009E <sub>H</sub>		bit	7	6	5	4	3	2	1	0
Detect address setting register 0 (PADR0): High			×	×	×	×	×	×	×	×
Address : 1FF2 <sub>H</sub>		bit	15	14	13	12	11	10	9	8
Detect address setting register 0 (PADR0): Middle			×	×	×	×	×	×	×	×
Address : 1FF1 <sub>H</sub>		bit	7	6	5	4	3	2	1	0
Detect address setting register 0 (PADR0): Low			×	×	×	×	×	×	×	×
Address : 1FF0 <sub>H</sub>		bit	7	6	5	4	3	2	1	0
Detect address setting register 1 (PADR1): High			×	×	×	×	×	×	×	×
Address : 1FF5 <sub>H</sub>		bit	15	14	13	12	11	10	9	8
Detect address setting register 1 (PADR1): Middle			×	×	×	×	×	×	×	×
Address : 1FF4 <sub>H</sub>		bit	7	6	5	4	3	2	1	0
Detect address setting register 1 (PADR1): Low			×	×	×	×	×	×	×	×
Address : 1FF3 <sub>H</sub>										
×		Undefined								

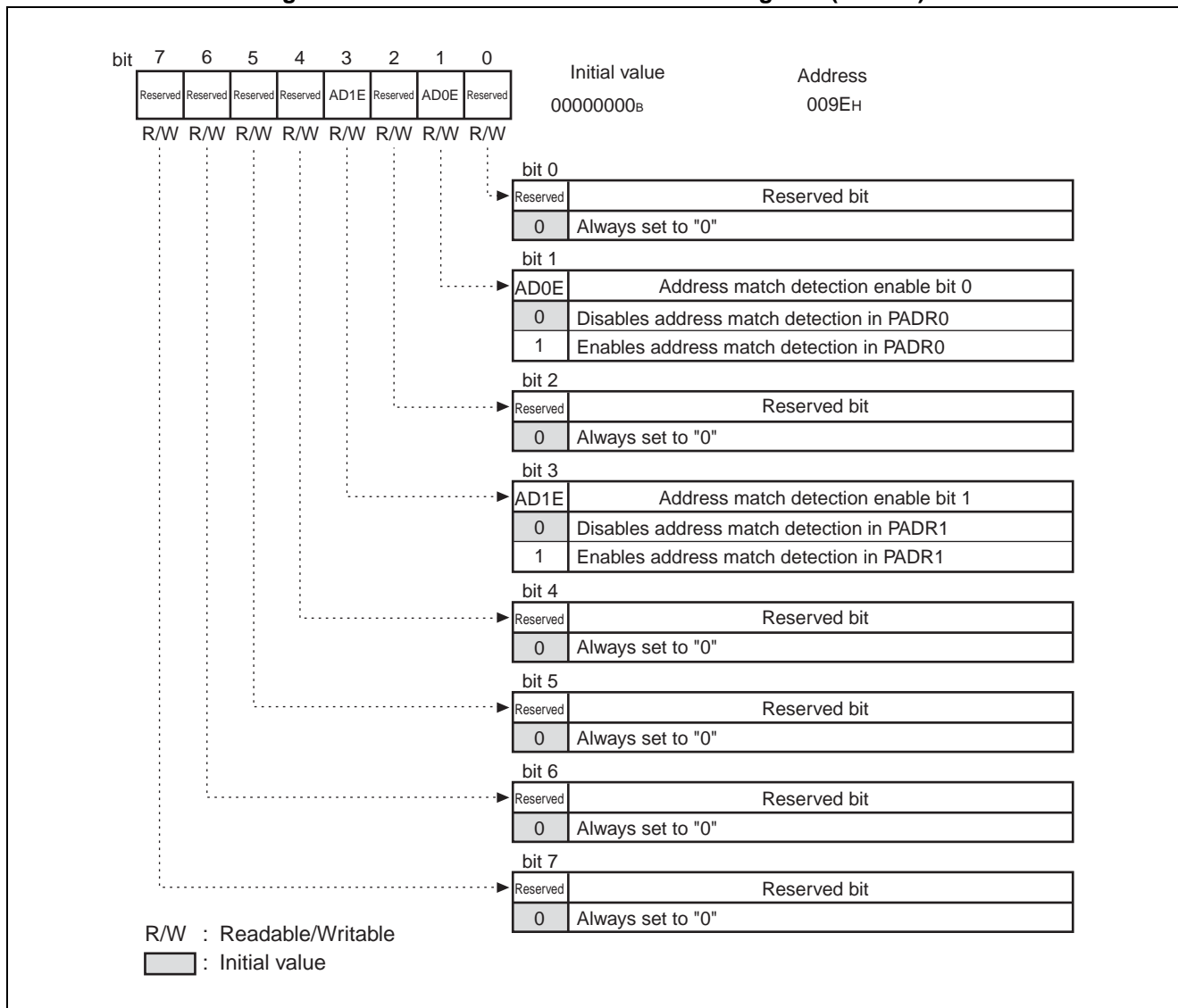
## MB90330A Series

### 24.3.1 Address Detection Control Register (PACSR)

The address detection control register (PACSR) enables or disables output of an interrupt at an address match. When an address match is detected when output of an interrupt at an address match is enabled, the INT9 interrupt is generated.

#### ■ Address Detection Control Register (PACSR)

Figure 24.3-2 Address Detection Control Register (PACSR)



**Table 24.3-1 Functions of Address Detection Control Register (PACSR)**

Bit Name		Function
bit 7 to bit 4	Reserved bit	Always set to "0".
bit 3	AD1E: Address match detection enable bit 1	<p>The address match detection operation with the detect address setting register 1 (PADR1) is enabled or disabled.</p> <p><b>When set to "0":</b> Disables the address match detection operation.</p> <p><b>When set to "1":</b> Enables the address match detection operation.</p> <ul style="list-style-type: none"> <li>When the value of detect address setting registers 1 (PADR1) matches with the value of address latch at enabling the address match detection operation (AD1E = 1), the INT9 instruction is immediately executed.</li> </ul>
bit 2	Reserved bit	Always set to "0".
bit 1	AD0E: Address match detection enable bit 0	<p>The address match detection operation with the detect address setting register 0 (PADR0) is enabled or disabled.</p> <p><b>When set to "0":</b> Disables the address match detection operation.</p> <p><b>When set to "1":</b> Enables the address match detection operation.</p> <ul style="list-style-type: none"> <li>When the value of detect address setting register 0 (PADR0) matches with the value of address latch at enabling the address match detect operation (AD0E = 1), the INT9 instruction is immediately executed.</li> </ul>
bit 0	Reserved bit	Always set to "0".

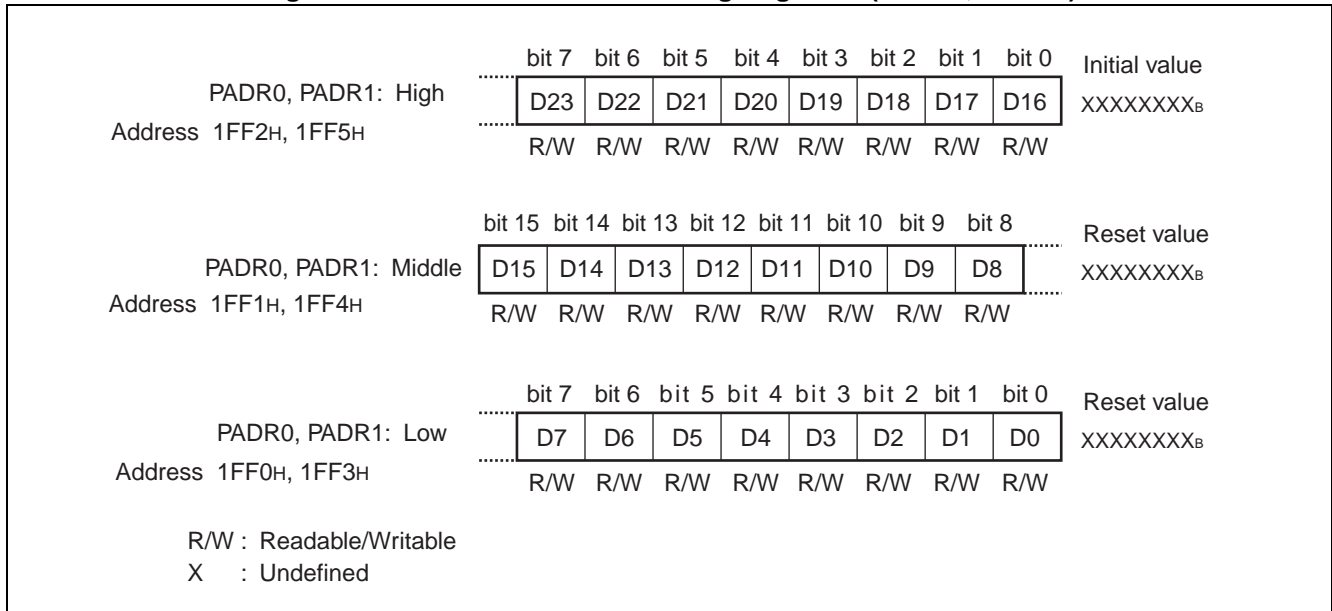
## MB90330A Series

### 24.3.2 Detect Address Setting Registers (PADR0, PADR1)

The value of an address to be detected is set in the detect address setting registers. When the address of the instruction processed by the program matches the address set in the detect address setting registers, the next instruction is forcibly replaced by the INT9 instruction, and the interrupt processing program is executed.

#### ■ Detect Address Setting Registers (PADR0, PADR1)

Figure 24.3-3 Detect Address Setting Registers (PADR0, PADR1)



## ■ Functions of Detect Address Setting Registers

- There are two detect address setting registers (PADR0, PADR1) that consist of a high byte (bank), middle byte, and low byte, totaling 24 bits.

**Table 24.3-2 Address Setting of Detect Address Setting Registers**

Register Name	Interrupt Output Enable	Address Setting	
Detect address setting register 0 (PADR0)	PACSR: AD0E	High	Set the upper 8 bits of detect address 0 (bank).
		Middle	Set the middle 8 bits of detect address 0.
		Low	Set the lower 8 bits of detect address 0.
Detect address setting register 1 (PADR1)	PACSR: AD1E	High	Set the upper 8 bits of detect address 1 (bank).
		Middle	Set the middle 8 bits of detect address 1.
		Low	Set the lower 8 bits of detect address 1.

- In the detect address setting registers (PADR0, PADR1), starting address (first byte) of instruction to be replaced by INT9 instruction should be set.

**Figure 24.3-4 Setting of Starting Address of Instruction Code to be Replaced by INT9**

Figure 2-10-4 Setting of Starting Address of Instruction Code to be Replaced by MOVW

Address	Instruction code	Mnemonic
FF001C :	A8 00 00	MOVW RW0, #0000
FF001F :	4A 00 00	MOVW A, #0000
FF0022 :	4A 80 08	MOVW A, #0880

### Notes:

- When an address of other than the first byte is set to the detect address setting register (PADR0, PADR1), the instruction code is not replaced by INT9 instruction and a program of an interrupt processing is not performed. When the address is set to the second byte or subsequent, the address set by the instruction code is replaced by "01<sub>H</sub>" (INT9 instruction code) and, which may cause malfunction.
- The detect address setting registers (PADR0, PADR1) should be set after disabling the address match detection (PACSR: AD0E = 0 or AD1E = 0) of corresponding address match control registers. If the detect address setting registers are changed without disabling the address match detection, the address match detection function will work immediately after an address match occurs during writing address, which may cause malfunction.
- The address match detection function can be used only for addresses of the internal ROM area. If addresses of the external memory area are set, the address match detection function will not work and the INT9 instruction will not be executed.

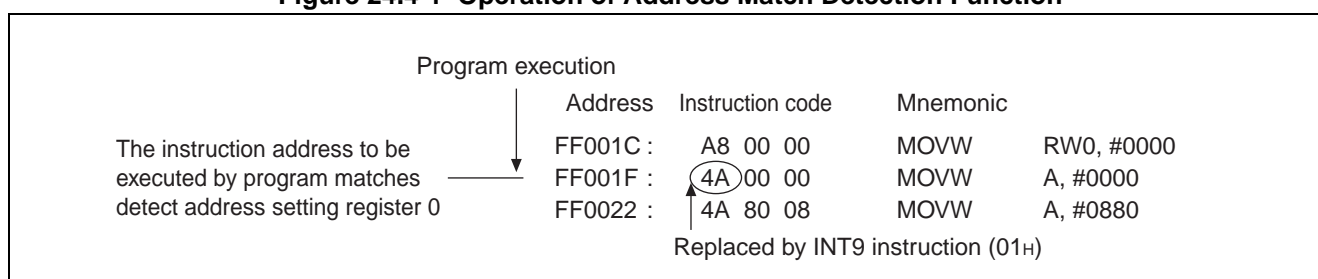
## 24.4 Explanation of Operation of Address Match Detection Function

If the addresses of the instructions executed in the program match those set in the detection address setting registers (PADR0, PADR1), the address match detection function will replace the first instruction with the INT9 instruction (01<sub>H</sub>) to branch to the interrupt processing program.

### ■ Operation of Address Match Detection Function

Figure 24.4-1 shows the operation of the address match detection function when the detect addresses are set and an address match is detected.

**Figure 24.4-1 Operation of Address Match Detection Function**



### ■ Setting Detect Address

1. Disable the detection address setting register 0 (PADR0) where the detect address is set for address match detection (PACSR: AD0E = 0).
2. Set the detect address in the detection address setting register 0 (PADR0). Set "FF<sub>H</sub>" at the higher bits of the detection address setting register 0 (PADR0), "00<sub>H</sub>" at the middle bits, and "1F<sub>H</sub>" at the lower bits.
3. Enable the detect address setting register 0 (PADR0) where the detect address is set for address match detection (PACSR: AD0E = 1).

### ■ Program Execution

1. If the address of the instruction to be executed in the program matches the set detect address, the first instruction code at the matched address is replaced by the INT9 instruction code (01<sub>H</sub>).
2. INT9 instruction is executed. INT9 interrupt is generated and then interrupt processing program is executed.

Table 25.5-2 shows the function list of the hardware sequence flag.



## 24.4.1 Example of Using Address Match Detection Function

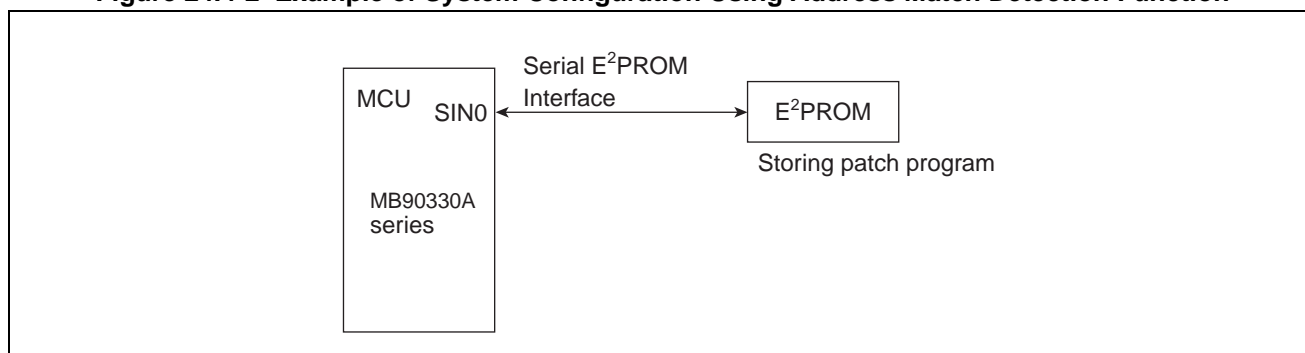
This section gives an example of patch processing for program correction using the address match detection function.

### ■ System Configuration and E<sup>2</sup>PROM Memory Map

#### ● System configuration

Figure 24.4-2 gives an example of the system configuration using the address match detection function.

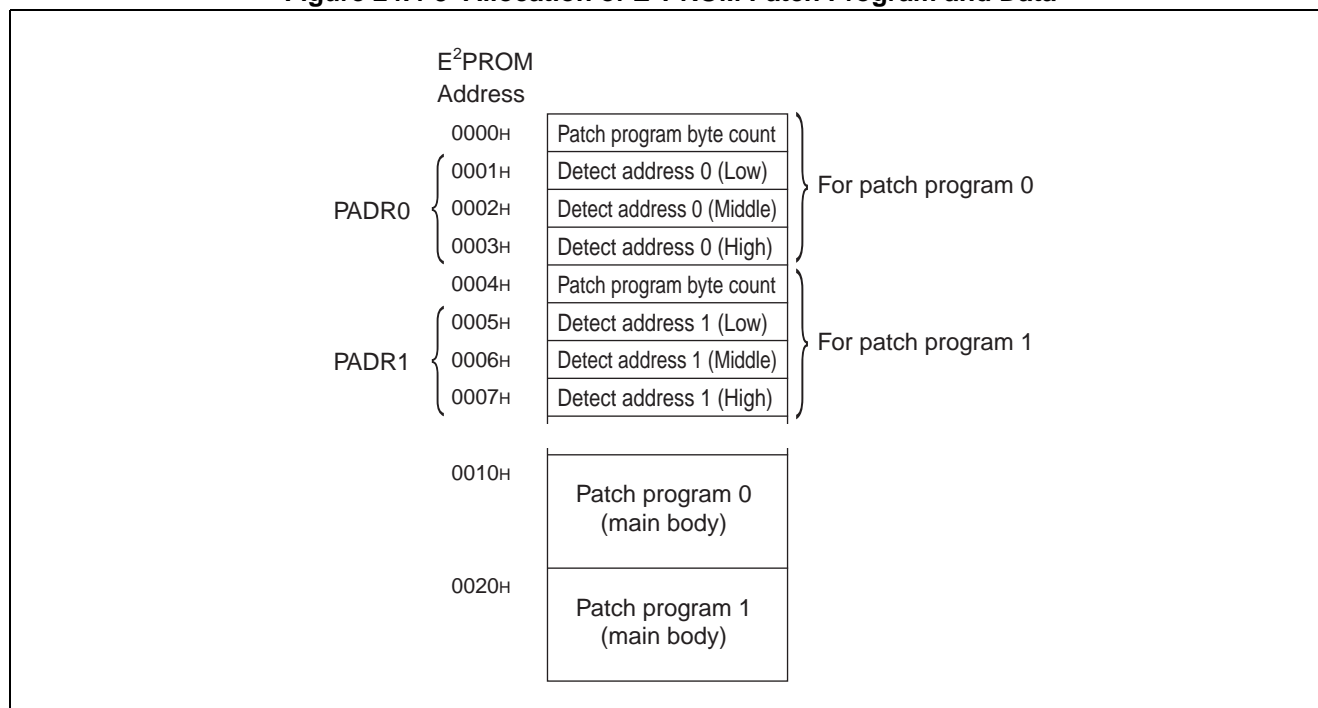
**Figure 24.4-2 Example of System Configuration Using Address Match Detection Function**



### ■ E<sup>2</sup>PROM Memory Map

Figure 24.4-3 shows the allocation of the patch program and data at storing the patch program in E<sup>2</sup>PROM.

Figure 24.4-3 Allocation of E<sup>2</sup>PROM Patch Program and Data



● Patch program byte count

The total byte count of the patch program (main body) is stored. If the byte count is "00H", it indicates that no patch program is provided.

● Detect address (24 bits)

The address where the instruction code is replaced by the INT9 instruction code due to program error is stored. This address is set in the detection address setting registers (PADR0, PADR1).

● Patch program (main body)

The program executed by the INT9 interrupt processing when the program address matches the detect address is stored. Patch program 0 is allocated from any predetermined address. Patch program 1 is allocated from the address indicating <starting address of patch program 0 + total byte count of patch program 0>.

## ■ Setting and Operating State

### ● Initialization

- E<sup>2</sup>PROM data are all cleared to "00<sub>H</sub>".

### ● Occurrence of program error

- By using the connector (UART), information about the patch program is transmitted to the MCU (MB90330A series) from the outside according to the allocation of the E<sup>2</sup>PROM patch program and data.
- The MCU (MB90330A series) stores the information received from outside in the E<sup>2</sup>PROM.

### ● Reset sequence

- After reset, the MCU (MB90330A series) reads the byte count of the E<sup>2</sup>PROM patch program to check the presence or absence of the correction program.
- If the byte count of the patch program is not "00<sub>H</sub>", the higher, middle and lower bits at detect addresses 0 and 1 are read and set in the detection address setting registers 0 and 1 (PADR0, PADR1). The patch program (main body) is read according to the byte count of the patch program and written to RAM in the MCU (MB90330A series).
- The patch program (main body) is allocated to the address where the patch program is executed in the INT9 interrupt processing by the address match detection function.
- Address match detection is enabled (PACSR: AD0E = 1, AD1E = 1)

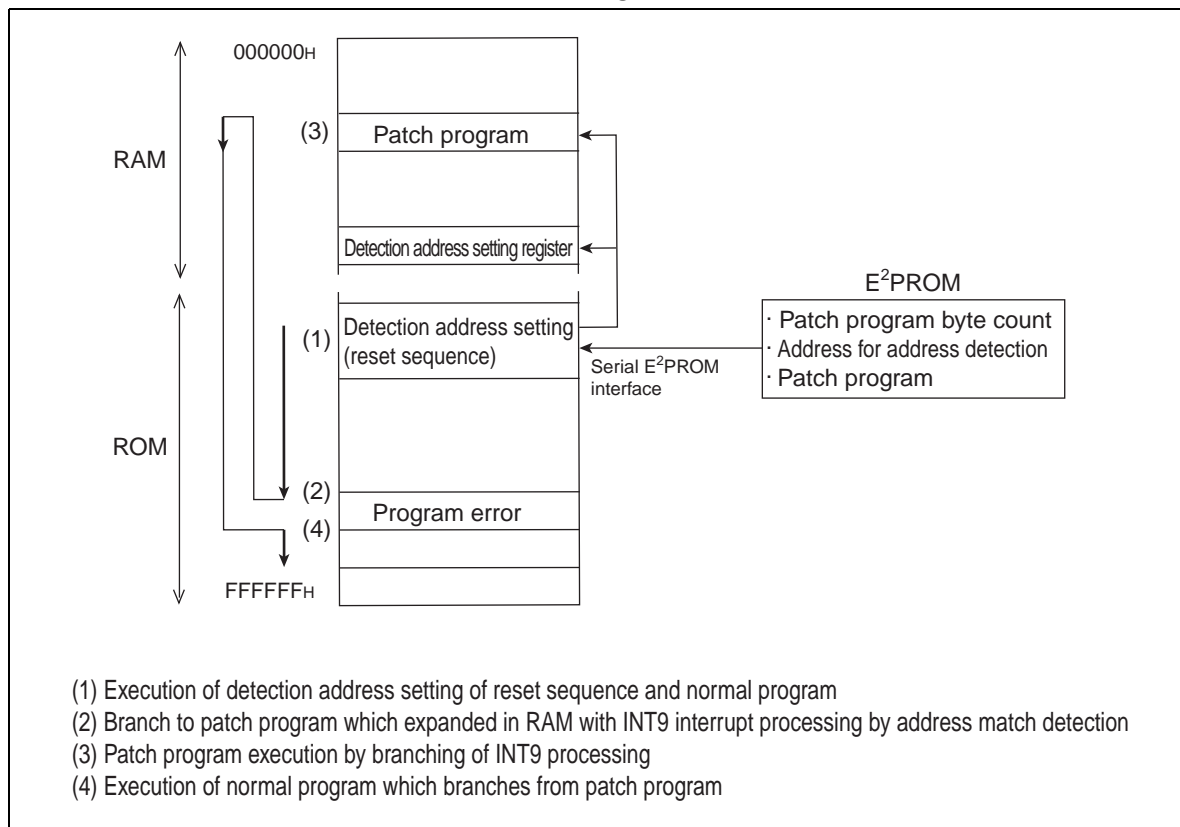
### ● INT9 Interrupt processing

- Interrupt processing is performed by the INT9 instruction. The MB90330A series has no interrupt request flag by address match detection. Therefore, if the stack information in the program counter is discarded, the detect address cannot be checked. When checking the detect address, check the value of program counter stacked in the interrupt processing routine.
- The patch program is executed, branching to the normal program.

## ■ Operation of Address Match Detection Function at Storing Patch Program in E<sup>2</sup>PROM

Figure 24.4-4 shows the operation of the address match detection function at storing the patch program in E<sup>2</sup>PROM.

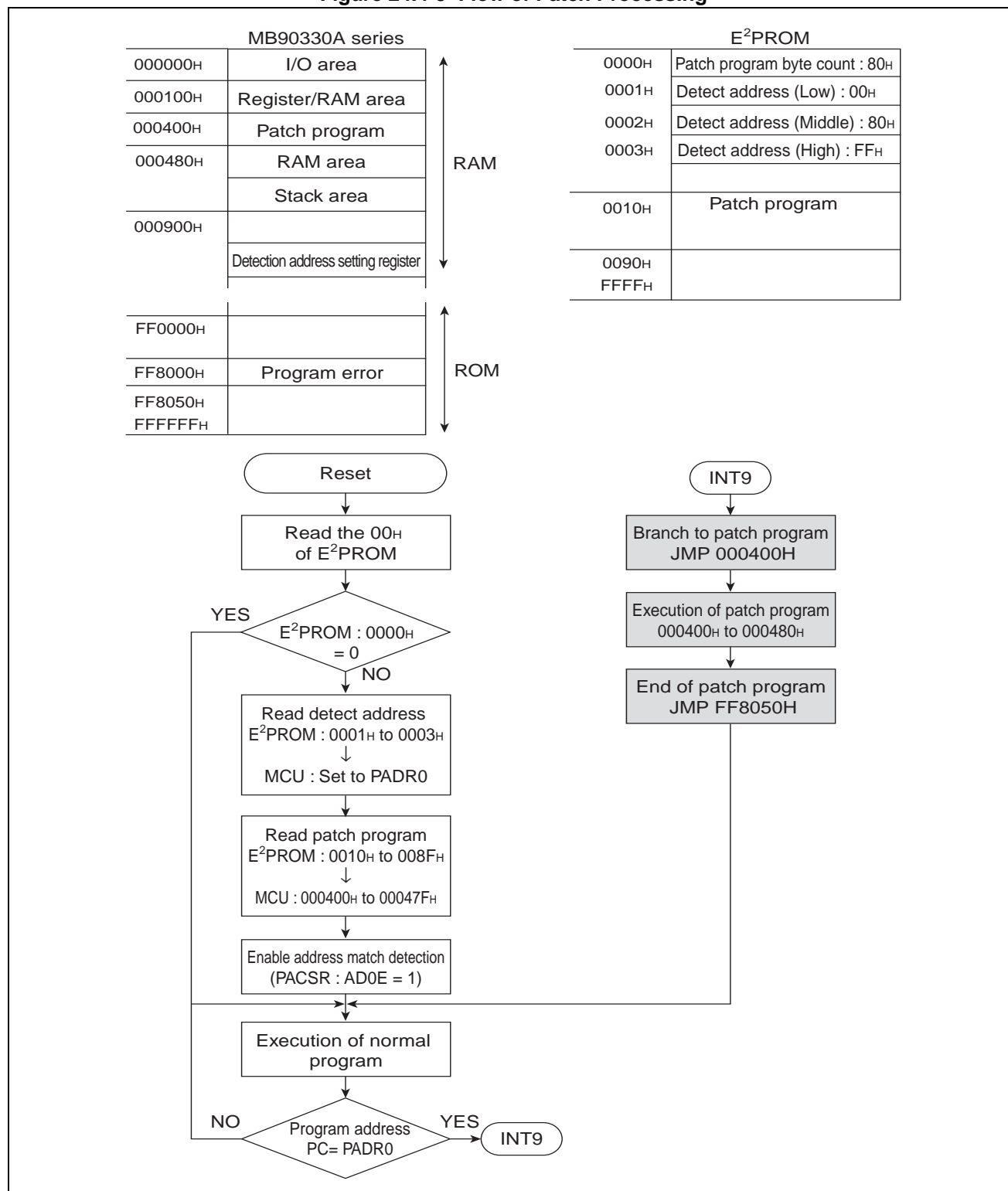
**Figure 24.4-4 Operation of Address Match Detection Function at Storing Patch Program in E<sup>2</sup>PROM**



## ■ Flow of Patch Processing

Figure 24.4-5 shows the flow of patch processing using the address match detection function.

Figure 24.4-5 Flow of Patch Processing



## 24.5 Program Example of Address Match Detection Function

---

This section gives a program example for the address match detection function.

---

### ■ Program Example for Address Match Detection Function

#### ● Processing specifications

If the address of the instruction to be executed by the program matches the address set in the detection address setting register (PADR0), the INT9 instruction is executed.

#### ● Coding example

```

PACSR    EQU    00009EH            ; Address detection control register
PADRL    EQU    001FF0H            ; Detection address setting register 0 (Low)
PADRM    EQU    001FF1H            ; Detection address setting register 0 (Middle)
PADRH    EQU    001FF2H            ; Detection address setting register 0 (High)
;
;-----Main program-----
CODE      CSEG
START:
    ; Stack pointer (SP), etc.,
    ; already reset
    MOV    PADRL,#00H              ; Set address detection register 0 (Low)
    MOV    PADM, #00H              ; Set address detection register 0 (Middle)
    MOV    PADRH,#00H              ; Set address detection register 0 (High)
;
    MOV    I:PACSR,#00000010B      ; Enable address match
    :
    processing by user
    :
LOOP:
    :
    processing by user
    :
    BAR    LOOP
;-----Interrupt program-----
WARI:
    :
    processing by user
    :
```

```

                                BETI                                ; Return from interrupt processing
CODE    ENDS
;-----Vector setting-----
VECT    CSEG    ABS=0FFH
        ORG     00FFDCH
        DSL     WARI
        ORG     00FFDCH                                ; Set reset vector
        DSL     START
        DB      00H                                    ; Set to single-chip mode
VECT    ENDS
        END     START
```

# **CHAPTER 25**

---

## **FLASH MEMORY**

**This chapter describes the functions and operations of 3M/4M-bit flash memory.**

**There are three methods for writing and erasing data on flash memory as described in the following.**

- **Programming and erasing by executing program**
- **Writing by serial writer**
- **Flash memory writer**

**In this chapter, we describe "Write/Erase by Program Execution".**

25.1 Overview of Flash Memory

25.2 Sector Configuration of Flash Memory

25.3 Flash Memory Control Status Register (FMCS)

25.4 Automatic Algorithm Initiation Method of Flash Memory

25.5 Check the Execution State of Automatic Algorithm

25.6 Write/Erase of Flash memory



## 25.1 Overview of Flash Memory

---

The flash memory is allocated from F9 to FF banks on the CPU memory map. The flash memory can be read-accessed and program-accessed from the CPU using the function of the flash memory interface circuit. Writing/erasing on flash memory can be executed by the instructions from the CPU through the flash memory interface circuit. Additionally, rewriting can be executed in the mounted condition by the control of the built-in CPU, the effective improvement of the program and data are realized.

---

### ■ Features of Flash Memory

Flash memory has the following features.

- Uses automatic program algorithm (Embedded Algorithm: the same manner as MBM29LV400TC.)
- Erase pause/restart function
- Detection of completion of writing/erasing by data polling and toggle bit.
- Detection of completion of writing/erasing by CPU interrupts.
- Erase per sector enabled (random sector combination)
- Programming/erase count 10000 (min.)
- Sector protect function (Possibility that sets up with a recommendation parallel writer)

### ■ Flash Memory Size and Products

Either 3M-bit or 4M-bit flash memory is mounted, depending on your model type.

#### ● 3M-bit flash memory

- Product: MB90F334A
- Size: 384K Bytes / 192K words
- Sector configuration:  $64K \times 5 + 32K + 8K \times 2 + 16K$

#### ● 4M-bit flash memory

- Product: MB90F335A
- Size: 512K Bytes / 256K words
- Sector configuration:  $64K \times 6 + 32K \times 2 + 8K \times 4 + 16K \times 2$

### ■ Method for Writing/deleting Flash Memory

Programming and erasing flash memory cannot be performed at one time. In other words, to perform writing/erasing data on flash memory, it is possible to execute writing operation by copying the program on the flash memory on RAM and executing from RAM, without program-accessing from the flash memory.

## 25.2 Sector Configuration of Flash Memory

Sector configuration of the flash memory is shown.

### ■ Sector Configuration

Figure 25.2-1 and Figure 25.2-2 show the sector configuration of the 3M-bit and 4M-bit flash memory respectively. The upper and lower addresses of each sector are shown in the figure.

**Figure 25.2-1 Sector Configuration of 3M-bit Flash Memory**

Flash memory	CPU address	Writer address*
SA8 (16K bytes)	FFFFFF <sub>H</sub>	7FFFF <sub>H</sub>
	FFC000 <sub>H</sub>	7C000 <sub>H</sub>
SA7 ( 8K bytes)	FFBFFF <sub>H</sub>	7BFFF <sub>H</sub>
	FFA000 <sub>H</sub>	7A000 <sub>H</sub>
SA6 ( 8K bytes)	FF9FFF <sub>H</sub>	79FFF <sub>H</sub>
	FF8000 <sub>H</sub>	78000 <sub>H</sub>
SA5 (32K bytes)	FF7FFF <sub>H</sub>	77FFF <sub>H</sub>
	FF0000 <sub>H</sub>	70000 <sub>H</sub>
SA4 (64K bytes)	FEFFFF <sub>H</sub>	6FFFF <sub>H</sub>
	FE0000 <sub>H</sub>	60000 <sub>H</sub>
SA3 (64K bytes)	FDFFFF <sub>H</sub>	5FFFF <sub>H</sub>
	FD0000 <sub>H</sub>	50000 <sub>H</sub>
Using disabled	FCFFFF <sub>H</sub>	4FFFF <sub>H</sub>
	FC0000 <sub>H</sub>	40000 <sub>H</sub>
SA2 (64K bytes)	FBFFFF <sub>H</sub>	3FFFF <sub>H</sub>
	FB0000 <sub>H</sub>	30000 <sub>H</sub>
SA1 (64K bytes)	FAFFFF <sub>H</sub>	2FFFF <sub>H</sub>
	FA0000 <sub>H</sub>	20000 <sub>H</sub>
SA0 (64K bytes)	F9FFFF <sub>H</sub>	1FFFF <sub>H</sub>
	F90000 <sub>H</sub>	10000 <sub>H</sub>
Using disabled	F8FFFF <sub>H</sub>	0FFFF <sub>H</sub>
	F80000 <sub>H</sub>	00000 <sub>H</sub>

In the case of access from CPU, SA0 is allocated on the F9 bank register, SA1 on the FA bank register, SA2 on the FB bank register, SA3 on the FD bank register, SA4 on the FE bank register, and SA5 to SA8 on the FF bank register.

**Figure 25.2-2 Sector Configuration of 4M-bit Flash Memory**

Flash memory	CPU address	Writer address*
SA13 (16K bytes)	FFFFFF <sub>H</sub>	7FFFF <sub>H</sub>
	FFC000 <sub>H</sub>	7C000 <sub>H</sub>
SA12 (8K bytes)	FFBFFF <sub>H</sub>	7BFFF <sub>H</sub>
	FFA000 <sub>H</sub>	7A000 <sub>H</sub>
SA11 (8K bytes)	FF9FFF <sub>H</sub>	79FFF <sub>H</sub>
	FF8000 <sub>H</sub>	78000 <sub>H</sub>
SA10 (32K bytes)	FF7FFF <sub>H</sub>	77FFF <sub>H</sub>
	FF0000 <sub>H</sub>	70000 <sub>H</sub>
SA9 (64K bytes)	FEFFFF <sub>H</sub>	6FFFF <sub>H</sub>
	FE0000 <sub>H</sub>	60000 <sub>H</sub>
SA8 (64K bytes)	FDFFFF <sub>H</sub>	5FFFF <sub>H</sub>
	FD0000 <sub>H</sub>	50000 <sub>H</sub>
SA7 (64K bytes)	FCFFFF <sub>H</sub>	4FFFF <sub>H</sub>
	FC0000 <sub>H</sub>	40000 <sub>H</sub>
SA6 (16K bytes)	FBFFFF <sub>H</sub>	3FFFF <sub>H</sub>
	FBC000 <sub>H</sub>	3C000 <sub>H</sub>
SA5 (8K bytes)	FBBFFF <sub>H</sub>	3BFFF <sub>H</sub>
	FBA000 <sub>H</sub>	3A000 <sub>H</sub>
SA4 (8K bytes)	FB9FFF <sub>H</sub>	39FFF <sub>H</sub>
	FB8000 <sub>H</sub>	38000 <sub>H</sub>
SA3 (32K bytes)	FB7FFF <sub>H</sub>	37FFF <sub>H</sub>
	FB0000 <sub>H</sub>	30000 <sub>H</sub>
SA2 (64K bytes)	FAFFFF <sub>H</sub>	2FFFF <sub>H</sub>
	FA0000 <sub>H</sub>	20000 <sub>H</sub>
SA1 (64K bytes)	F9FFF <sub>H</sub>	1FFFF <sub>H</sub>
	F90000 <sub>H</sub>	10000 <sub>H</sub>
SA0 (64K bytes)	F8FFF <sub>H</sub>	0FFFF <sub>H</sub>
	F80000 <sub>H</sub>	00000 <sub>H</sub>

In the case of access from CPU, SA0 is allocated on the F8 bank register, SA1 on the F9 bank register, SA2 on the FA bank register, SA3 to SA6 on the FB bank register, SA7 on the FC bank register, and SA8 on the FD bank register, SA9 on the FE bank register, SA10 to SA13 on the FF bank register.

\*: The writer address

The writer address is the address corresponding to the CPU address when data is written on the flash memory with the parallel writer. When writing/erasing is executed using a general-purpose writer, writing/erasing is executed in this address.

**MB90330A Series****25.3 Flash Memory Control Status Register (FMCS)**

The flash memory control status register (FMCS), which is allocated in the flash memory interface circuit, is used for writing/erasing on the flash memory.

**Flash Memory Control Status Register (FMCS)**

Figure 25.3-1 shows the bit configuration of the flash memory control status register (FMCS).

**Figure 25.3-1 Bit Configuration of Flash Memory Control Status Register (FMCS)**

Flash memory control status register (FMCS)								Initial value 000X0000 <sub>B</sub>
Address: 0000AE <sub>H</sub>	bit 7	6	5	4	3	2	1	0
	INTE	RDYINT	WE	RDY	Reserved	Reserved	Reserved	Reserved
	R/W	R/W	R/W	R	W	W	W	W
R/W : Readable/Writable								
R : Read only								
W : Write only								

The function of each bit in the flash memory control status register (FMCS) is described in the following.

**[bit 7] INTE: INTerrupt Enable**

Finishing writing/erasing on the flash memory allows the CPU to generate an interrupt.

If the INTE bit is "1" and the RDYINT bit is "1", an interrupt to the CPU is generated. If the INTE bit is "0", the interrupt is not generated.

0	Interrupt disabled in writing/erasing end
1	Interruption permission by writing/erase end

**[bit 6] RDYINT: ReaDY INTerrupt**

This is the bit representing the operation state of flash memory.

Completion of writing/erasing on the flash memory allows this bit to set to "1". After writing/erasing on flash memory, it is not possible to write/erase to flash memory while the RDYINT bit is "0". After writing/erasing is completed and "1" is set, writing/erasing to flash memory is possible.

Writing "0" clears to "0" and writing "1" is ignored. The completion timing of the flash memory automatic algorithm (see Section "25.4 Automatic Algorithm Initiation Method of Flash Memory") sets to "1". When the read modify write (RMW) instruction is used, "1" is always read.

0	During writing/the deletion operation execution
1	Completion of writing/erasing operation (occurrence of interrupt requests)

**[bit 5] WE: Write Enable**

It is a write enable bit to the flash memory area.

When the setting is "1", writing after issuing command sequences for F9 to FB, FD to FF banks (see Section "25.4 Automatic Algorithm Initiation Method of Flash Memory") is executed on the area in the flash memory. The signal of writing/ deletion is not generated at "0". It is used for activating the write/ erase command of flash memory.

In order to avoid erroneous data writing to the flash memory, it is recommended to set "0" permanently when the writing/erasing command is not used.

0	Writing Erasing on Flash Memory disabled
1	Writing Erasing on Flash Memory enabled

**[bit 4] RDY: Ready**

This is the writing/erasing permission bit of flash memory.

While this is "0", the writing/erasing flash memory are disabled. However, the read/reset command or sector erase suspend command is enabled in this state.

0	During writing/ deletion operation execution
1	Completion of writing/erasing operation (write/erase the next data enabled)

**[bit 3 to bit 0] Reserved bit**

These are reserved bits. Be sure to set to "0".

Note:

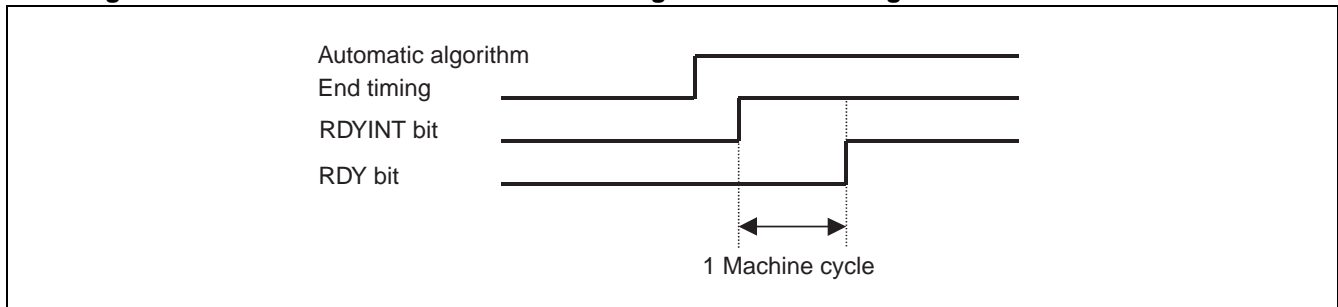
As for the FMCS register, only byte access is possible.

**■ Automatic Algorithm End Timing**

Figure 25.3-2 shows the relation between automatic algorithm end timing and RDYINT bit and RDY bit.

RDYINT and RDY bits do not change at the same time. Please make the program to judge either by one of bits.

**Figure 25.3-2 Relation between Automatic Algorithm End Timing and RDYINT Bit and RDY Bit**



## 25.4 Automatic Algorithm Initiation Method of Flash Memory

There are four types of commands to start the automatic algorithm of flash memory: read/reset, write, sector erase, and chip erase. The control of the suspend and the restart is enabled for the sector erase.

### ■ Command Sequence Table

Table 25.4-1 lists the commands used for flash memory write/erase. All data written on the command register is in the units of byte, but be sure that the data can be written in word access. In this case, high-order byte data is ignored.

**Table 25.4-1 Command Sequence Table**

Command Sequence	Bus Programming Cycle	the 1th bus Programming Cycle		the 2th bus Programming Cycle		the 3th bus Programming Cycle		the 4th bus Programming Cycle		the 5th bus Programming Cycle		the 6th bus Programming Cycle	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Read/Reset *	1	FxXXXX	XXF0	-	-			-	-	-	-	-	-
Read/Reset *	4	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XXF0	RA	RD	-	-	-	-
Programming	4	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XXA0	PA (even)	PD (word)	-	-	-	-
Chip Erasing	6	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX80	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX10
Sector Erasing	6	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX80	FxAAAA	XXAA	Fx5554	XX55	SA (even)	XX30
Sector Erasing being Suspended		Entry of Address "FxXXXX" Data(0xB0 <sub>H</sub> ) temporally suspends erasing a sector in the middle of erasing the sector.											
Sector Erasing being restarted		Entry of Address "FxXXXX" Data(0x30 <sub>H</sub> ) restarts erasing after the temporal suspension of erasing of the sector.											
Auto-Select	3	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX90	-	-	-	-	-	-

RA: Read address

PA: Program address. Only even addresses can be specified.

SA: See sector addresses (see Section "25.2 Sector Configuration of Flash Memory").

RD: Read data

PD: Program data. Only word data can be specified.

\*: Two kinds of read/reset commands can reset flash memory to the read mode.

Notes: • The address Fx in the table means FF, FE, FD, FB, FA, and F9. In each operation, specify this as the value of the bank to be accessed.

• Addresses in the table are the values in the CPU memory map. All addresses and data are expressed as hexadecimals. However, "X" is any value.

The Auto-Select in Table 25.4-1 is a command that is used to recognize the state of the sector protection.

Actually, an address is needed to be set along with the above command as follows:

**Table 25.4-2 Address Setting at Auto-Select**

	AQ13 to AQ17	AQ7	AQ2	AQ1	AQ0	DQ7 to DQ0
Sector protection	Sector address	L	H	L	L	CODE *

\*: "01<sub>H</sub>" for the output in protected sector addresses, "00<sub>H</sub>" for the output in non-protected sector addresses.

## 25.5 Check the Execution State of Automatic Algorithm

Because write/erase flow are performed through automatic algorithm in flash memory, the algorithm must wait signals from hardware that notify the operation conditions or operation end. This automatic algorithm can check the operation state of the internal flash memory by the following hardware sequence.

### ■ Hardware Sequence Flag

The hardware sequence flag consists of 4-bit outputs: DQ7, DQ6, DQ5, and DQ3. Each bit has the functions of the data polling flag (DQ7), toggle bit flag (DQ6), timing limit excess flag (DQ5), and sector erase timer flag (DQ3). These flags can be used to confirm whether write/chip sector erase end and erase code write are effective or not.

Reference to the hardware sequence flag, read-access to the address in the target sector in the flash memory after setting the command sequence (See Table 25.4-1). Table 25.5-1 shows the bit allocation of the hardware sequence flag.

**Table 25.5-1 Bit Allocation of Hardware Sequence Flag**

Bit No.	7	6	5	4	3	2	1	0
Hardware sequence flag	DQ7	DQ6	DQ5	-	DQ3	-	-	-

To check if the automatic write/chip sector erase is operating or ended, verify the hardware sequence flag or the RDY bit of the flash memory control register (FMCS). After the write/erase commands end, it returns to the read/reset state. When actually creating a program, perform the following procedures including data read after verifying the automatic write/erase end with one of these flags. Also, you can check if the sector erase code write after the second command is enabled by using the hardware sequence flag. The next section describes each hardware sequence flag.

Table 25.5-2 shows the list of the hardware sequence flag functions.

**Table 25.5-2 Function List of Hardware Sequence Flag**

State		DQ7	DQ6	DQ5	DQ3
State change in normal operation	Writing operation → writing completion (when program address specified)	$\overline{DQ7} \rightarrow \text{DATA:7}$	Toggle → DATA:6	0 → DATA:5	0 → DATA:3
	Chip sector deletion operation → deletion completion	0 → 1	Toggle → Stop	1	0 → 1
	Wait for Sector Erasing → erasing start	0	Toggle	0 → 1	0
	Sector Erasing being Suspended (Sector being erased)	0 → 1	Toggle → 1	1 → 0	0
	Sector erasing suspended → Resumed (Sector being erased)	1 → 0	1 → Toggle	0 → 1	0
	Sector erasing being suspended (Sector not being deleted)	DATA:7	DATA:6	DATA:5	DATA:3
Abnormal operation	Programming	$\overline{DQ7}$	Toggle	0	1
	Chip sector deletion operation	0	Toggle	1	1



## 25.5.1 Data Polling Flag (DQ7)

The data polling flag (DQ7) is a hardware sequence flag that notifies the state that the automatic algorithm execution is proceeding or ended by using the data polling function.

### ■ State Transition of Data Polling Flag (DQ7)

Table 25.5-3 and Table 25.5-4 shows the state transition of the data polling flag.

**Table 25.5-3 State Transition of Data Polling Flag (State Change in Normal Operation)**

Operation State	Programming → Completion	Chip and sector erasing → Completion	Sector erasing wait → start	Sector Erasing → Temporary deletion stop Sector being erased	Sector Erasing being Suspended → Restart Sector being erased	Sector Erasing Temporary Sector not being deleted
DQ7	$\overline{DQ7} \rightarrow \text{DATA:7}$	$0 \rightarrow 1$	0	$0 \rightarrow 1$	$1 \rightarrow 0$	DATA:7

**Table 25.5-4 State Transition of Data Polling Flag (State Change in Abnormal Operation)**

Operation State	Programming operation	Chip sector erase operation
DQ7	$\overline{DQ7}$	0

### ■ At Programming Operation

If you execute read-access while executing the automatic write algorithm, the flash memory outputs the inversion data of the last update of bit 7 regardless of address specifications. If you execute read-access when the automatic write algorithm ends, the flash memory outputs the bit 7 of the read value of the specified address.

### ■ When Chip/sector Deletion Operates

If you execute read-access from the sector which is currently being erased, while executing the automatic sector-erase algorithm, the flash memory outputs "0". If you execute read-access while executing the automatic chip-erase algorithm, the flash memory outputs "0", regardless of the specified address. Once the chip/sector-erase operation ends, the flash memory outputs "1".

### ■ Sector Erasing being Suspended

If you execute read-access when the sector erase suspends, the flash memory outputs "1" if the specified address is the erased sector, and it outputs bit 7 (DATA: 7) of the read value of the specified address if it is not the erased sector. You can determine whether the current state is the sector suspend or not and which sector is being erased by referencing with the toggle bit flag (DQ6).

---

**Note:**

At the start-up of the automatic algorithm, the read-access to the specified address is disregarded. As the data polling flag (DQ7) ends, other bit outputs are enabled for data read. Therefore, be sure to read data after the automatic algorithm ends, following read-access whose end of the data polling has confirmed.

---

## 25.5.2 Toggle Bit Flag (DQ6)

Like the data polling flag (DQ7), the toggle bit flag (DQ6) is a hardware sequence flag that indicates that the automatic algorithm execution is proceeding or ended by the toggle bit function.

### ■ Transition State of Toggle Bit Flag (DQ6)

Table 25.5-5 and Table 25.5-6 shows the transition state of the toggle bit flag.

**Table 25.5-5 Transition State of Toggle Bit Flag (State Change in Normal Operation)**

Operation State	Programming → Completion	Chip and sector erasing → Completion	Sector erasing wait → start	Sector Erasing → Temporary deletion stop Sector being erased	Sector Erasing being Suspended → Restart Sector being erased	Sector Erasing Temporary Sector not being deleted
DQ6	Toggle → DATA:6	Toggle → Stop	Toggle	Toggle → 1	1 → Toggle	DATA:6

**Table 25.5-6 Transition State of Toggle Bit Flag (State Change in Abnormal Operation)**

Operation State	Programming operation	Chip sector deletion operation
DQ6	Toggle	Toggle

### ■ When Writing/chip Sector is Deleted

If you execute read-access sequentially while executing the automatic write algorithm and the automatic algorithm of chip/sector erase, the flash memory outputs the toggle state generating "1" and "0" alternately per read, regardless of the specified address. If you execute read-access sequentially after the automatic write algorithm and the automatic chip/sector-erase algorithm end, the flash memory stops the toggle operation of bit 6 and outputs the bit 6 (DATA: 6) of the read value of the specified address.

### ■ Sector Erasing being Suspended

If you execute read-access when the sector erase suspends, the flash memory outputs "1" if the specified address belongs to the sector being erased. If the specified address does not belong to the sector being erased, it outputs the bit 6 (DATA: 6) of the read value of the specified address.

#### Reference:

During writing, when the sector being written is rewrite-protected, the toggle operation is terminated without rewriting the data after toggle-operating for about 2μs.

During erasing, if all the selected sectors are rewrite-protected, the toggle bit performs toggle-operation for about 100 μs, and then it goes back to the read/reset state without rewriting the data.

**MB90330A Series****25.5.3 Timing Limit Over Flag (DQ5)**

The timing limit excess flag (DQ5) is a hardware sequence flag that indicates that the automatic algorithm execution is over the predefined time (internal pulse frequency) in the flash memory.

**■ Transition State of Timing Limit Over Flag (DQ5)**

Table 25.5-7 and Table 25.5-8 shows the transition state of the timing limit over flag.

**Table 25.5-7 Transition State of Timing Limit Over Flag (State Change in Normal Operation)**

Operation State	Programming → Completion	Chip and sector erasing → Completion	Sector erasing wait → start	Sector Erasing → Temporary deletion stop Sector being erased	Sector Erasing being Suspended → Restart Sector being erased	Sector Erasing Temporary Sector not being deleted
DQ5	0 → DATA:5	0 → 1	0	0	0	DATA:5

**Table 25.5-8 Transition State of Timing Limit Over Flag (State Change in Abnormal Operation)**

Operation State	Programming operation	Chip sector deletion operation
DQ5	1	1

**■ When Writing/chip Sector is Deleted**

If you execute read-access after starting the automatic algorithms of the write or the chip erase/sector erase, the flash memory outputs "0" if it is within the predefined time (time necessary for write/erase) and "1" if it is over the predefined time. As this flag is irrelevant to the state of the automatic algorithm, you can judge whether the write/erase algorithm was successfully executed or not. In other words, the automatic algorithm is still operating by the data polling or toggle bit functions when this flag outputs "1", you can recognize that writing was failed.

For example, if you write "1" into the flash memory address to which "0" is already written, the operation fails. In this case, the flash memory is locked and the automatic algorithm does not end. In rare cases, it ends normally as if "1" was written to the address. Therefore, an effective data is not output from the data polling flag (DQ7). Also, the toggle bit flag (DQ6) does not terminate the toggle operation, and the timing limit excess flag (DQ5) outputs "1" overrunning the time limit. This state means that the flash memory does not have any defect, but it was not used properly. Be sure to execute the reset command if this state appears.

## 25.5.4 Sector Erasing Timer Flag (DQ3)

The sector erase timer flag (DQ3) is a hardware sequence flag that indicates whether the status is in the sector erase wait period or not after starting the sector erase command.

### ■ Transition State of Sector Erasing Timer Flag (DQ3)

Table 25.5-9 and Table 25.5-10 shows the transition state of the sector erasing timer flag.

**Table 25.5-9 Transition State of Sector Erasing Timer Flag (State Change in Normal Operation)**

Operation State	Programming → Completion	Chip and sector erasing → Completion	Sector erasing wait → start	Sector Erasing → Temporary deletion stop Sector being erased	Sector Erasing being Suspended → Restart Sector being erased	Sector Erasing Temporary Sector not being deleted
DQ3	0 → DATA:3	1	0 → 1	1 → 0	0 → 1	DATA:3

**Table 25.5-10 Transition State of Sector Erasing Timer Flag (State Change in Abnormal Operation)**

Operation State	Programming operation	Chip sector deletion operation
DQ3	0	1

### ■ When Sector Deletion Operate

If you execute read-access after starting the sector erase command, the flash memory outputs "0" if this access is in the sector erase wait period and "1" if this access is over the sector erase wait period, regardless of the specified address of the sector.

When the erase algorithm is in execution with the data polling and toggle bit functions, the internal erasing is beginning if this flag is "1". Any subsequent commands are ignored except for the write command or erase suspend command for the sector erase code until erasing finishes.

When this flag is "0", the flash memory accepts writing of an additional sector erase code. To verify that, it is recommended to check this flag state before writing of subsequent erase codes. If it is "1" at the second state checking, the additional sector erase code may not be accepted.

### ■ Sector Erasing being Suspended

If you execute read-access when the sector erase suspends, the flash memory outputs "1" if the specified address is the erased sector, and it outputs bit 3 (DATA: 3) of the read value of the specified address if it is not the erased sector.

## 25.6 Write/Erase of Flash memory

---

**This section describes the procedure to issue a command to start the automatic algorithm and how to perform the operations of the read/reset, write, chip erase, sector erase, sector erase suspend, and sector erase restart.**

---

### ■ Write/Erase of Flash Memory

The flash memory can execute the automatic algorithm by performing the read/reset, write, chip erase, sector erase, sector erase suspend, and sector erase restart operations to write in cycle into the bus of the command sequence (See Table 25.4-1). The write cycle into each bus needs to be performed consecutively. Besides, the automatic algorithm can detect the end time with data polling function or other functions. After normal termination, it returns to the read/reset state.

In the following section, the following items regarding the write/erase on the flash memory are described.

- Read/reset state
- Writing Data
- All data erasing (chip all erase)
- Any data erasing (sector erase)
- Suspend Sector erasing
- Restart Sector erasing

## **25.6.1 Read/Reset State in Flash Memory**

---

**This section describes the procedure to issue the read/reset commands and to set the read/reset state in flash memory.**

---

### **■ Read/Reset State in Flash Memory**

To set the flash memory to the read/reset state, it is possible to execute by sending the read/reset command sequentially in the command sequence table (See Table 25.4-1) to the target sector in the flash memory.

The read/reset command has two kinds of command sequences of the one-time bus operation and the three-times bus operation, while in fact, there is no fundamental differences between them.

The read/reset state is the initial state of the flash memory, and this system is always the read/reset state at the time of the power-on or the normal command completion. Reading/reset is an input waiting state of other commands.

In the read/reset state, you can read-access normally to read data. As with mask ROM, program access from the CPU is enabled. This command is not needed to read data in the normal data read. When you need to initialize the automatic algorithm as in the case of an abnormal command completion for some causes, this command is mainly used.

## 25.6.2 Writing Data to Flash Memory

---

The procedure to issue the write command and write data into the flash memory is described.

---

### ■ Writing Data to Flash Memory

To start the data write automatic algorithm, you may send the write command sequentially in the command sequence table (See Table 25.4-1) to the target sector in the flash memory. When data writing to the target address ends at the forth cycle, the automatic algorithm is activated and then the automatic writing starts.

#### ● How to specify address

Only an even address is acceptable for the write address specified in the data write cycle. An odd address does not allow you to write properly. In other words, writing into an even address per word data is required. However, execution of one programming command, permits programming of only one word for data.

#### ● Notes on data programming

Data "0" cannot be returned to data "1" by writing. If you write data "1" into data "0", the data polling algorithm (DQ7) or the toggle operation (DQ5) do not finish and the flash memory device is determined to have a defect, and as a result writing exceeds the predefined time, and the timing limit excess. However, when data is read while read/reset, data is "0". You can change data from "0" to "1" only in the erase operation.

All commands are disregarded while the automatic write command is being executed. Please note that if the hardware reset is started while writing, the address data currently being written is not secured.

### ■ Writing Procedure of Flash Memory

Figure 25.6-1 shows an Procedure example for writing to flash memory. You may determine the state of the automatic algorithm in the flash memory by using the hardware sequence flag (See Section "25.5 Check the Execution State of Automatic Algorithm"). In this section data polling flag (DQ7) is used to confirm an end of writing.

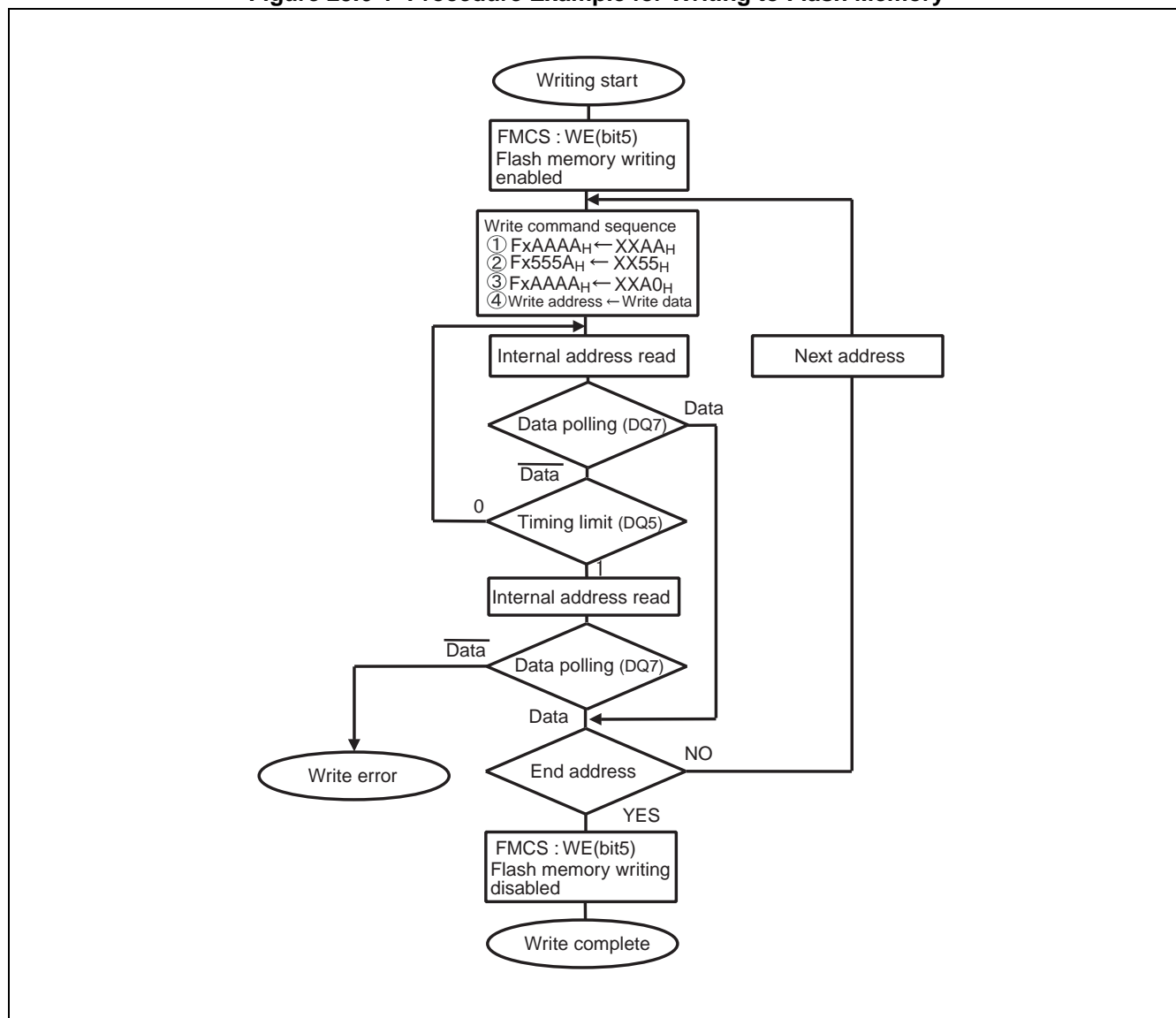
Flag check data should be read from the address where data was last written.

As the data polling flag (DQ7) is changed along with the timing limit excess flag (DQ5), you need to recheck the data polling flag bit (DQ7) even if the timing limit excess flag (DQ5) is "1".

Likewise, the toggle bit flag (DQ6) terminates the toggle operation just when the timing limit excess flag (DQ5) changes into "1", you need to recheck the toggle bit flag (DQ6).



**Figure 25.6-1 Procedure Example for Writing to Flash Memory**



### 25.6.3 Erasing All Data from Flash Memory (Chip Erase)

---

**This section describes the procedure for issuing the chip erase command and erasing all the data from flash memory.**

---

#### ■ Erasing All Data from Flash Memory (Chip Erase)

To erase all the data from the flash memory, you can erase by sending the chip erase command in the command sequence table (See Table 25.4-1) to the target sector in the flash memory.

Chip deletion command is done by six bus operations. When data writing into the target address ends at the sixth cycle, the chip erase operation is started. Before chip erasing, the user need not perform programming to flash memory. While the automatic algorithm is executed, the flash memory writes "0" to check conditions before erasing all the cells automatically.

## **25.6.4 Erasing Any Data in Flash Memory (Sector Erasing)**

---

**The procedure to issue the sector erase command and erase any data in the flash memory is described. Sector-by-sector erasing is enabled and multiple sectors can be specified at a time.**

---

### **■ Erasing Any Data in Flash Memory (Sector Erasing)**

To erase any data in the flash memory, you may send the sector erase command sequentially in the command sequence table (See Table 25.4-1) to the target sector in the flash memory.

#### **● How to specify sector**

The sector erase command is executed in six bus operations. In the sixth cycle, the minimum 50  $\mu$ s sector erase wait is started by writing the sector erase code (30<sub>H</sub>) into any even one address accessible in the target sector. To erase multiple sectors, write the erase code (30<sub>H</sub>) into the address in the target sector to be erased, following the above procedure.

#### **● Notes on specifying multiple sectors**

Sector erasing is started after a minimum 50  $\mu$ s period waiting for sector erasing is completed after the last sector erase code has been programmed. In other words, when multiple sectors are erased at the same time, the next erase sector address and the erase code (in the command sequence sixth cycle) need to be entered within 50  $\mu$ s, after which they may not be accepted. You may check if writing the subsequent sector erase code is enabled or not by the sector erase timer (hardware sequence flag: DQ3). In this case, make sure that the address to read the sector erase timer indicates the sector to be erased.

### **■ Procedure of Sector Deletion**

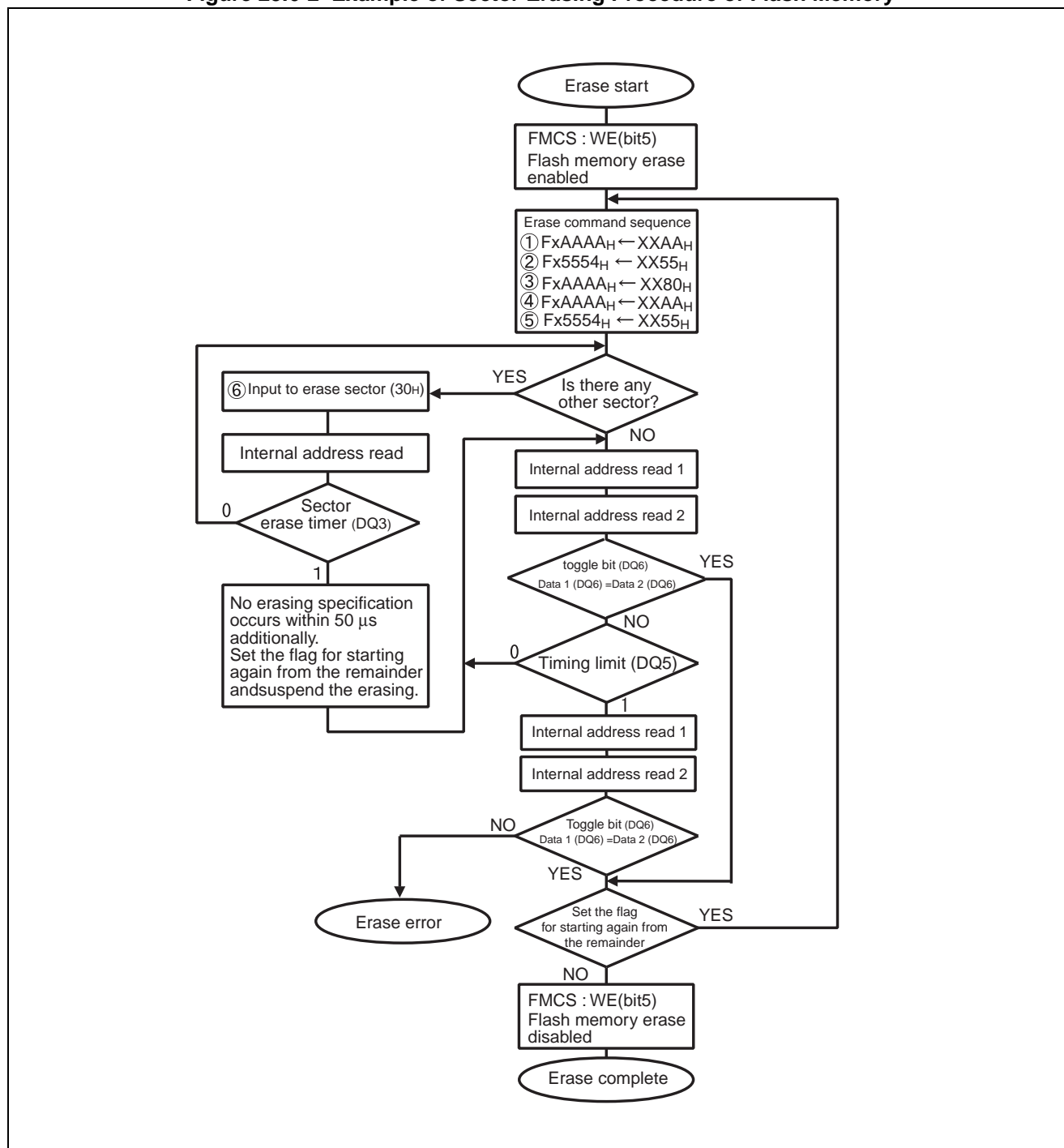
You may determine the state of the automatic algorithm in the flash memory by using the hardware sequence flag (See Section "25.5 Check the Execution State of Automatic Algorithm"). Figure 25.6-2 shows an example of the sector erase procedure of the flash memory. In this example, the toggle bit flag (DQ6) is used to check that erase ends.

Please note that data to be read for flag check is read from the sector to be erased.

The toggle bit flag (DQ6) terminates the toggle operation at the moment at which the timing limit excess flag (DQ5) changes into "1", you need to recheck the toggle bit flag (DQ6) even if the timing limit excess flag (DQ5) is "1".

Likewise, the data polling flag (DQ7) changes at the moment at which the timing limit excess flag (DQ5) changes into "1", you need to recheck the data polling flag (DQ7).

Figure 25.6-2 Example of Sector Erasing Procedure of Flash Memory



## **25.6.5 Flash Memory Sector Erase Suspension**

---

**This section describes the procedure to issue the sector erase suspend command and suspend the sector erase of flash memory. Data can be read from the sector not being deleted.**

---

### **■ Flash Memory Sector Erase Suspension**

To suspend the sector erase of the flash memory, you can perform the suspension by sending the sector erase suspend command in the command sequence table (See Table 25.4-1) to the flash memory.

The sector erase suspend command enables you to suspend the erase while erasing a sector and read data from the sector that is not being erased. In this state, you can only read but cannot write. This command is enabled only when a sector is being erased including the erase wait time and ignored while erasing or writing to a chip.

This command can be executed by writing the erase suspend code (B0<sub>H</sub>), for which the address should indicate any address in the flash memory. The sector erase suspend command is ignored at the second time for erase suspend.

If the sector erase suspend command is entered during a sector erase wait period, sector erase wait is finished immediately and erase operation is stopped, and the state enters in erase-terminated condition. During sector deletion following the waiting period for sector deletion, if a deletion suspension command is input, deletion suspension status starts after a maximum of 15  $\mu$ s. The sector erase suspend command is performed after 20  $\mu$ s following the issuance of the sector erase command or sector erase resume command.

## 25.6.6 Flash Memory Sector Erase Resumption

---

**The procedure to issue the sector erase restart command and restart the flash memory sector erase that is suspended is described.**

---

### ■ Flash Memory Sector Erase Resumption

To restart the suspended sector erase, you can restart by sending the sector erase restart command sequentially in the command sequence table (See Table 25.4-1) to the flash memory.

The sector erase resume command resumes sector erasing suspended by the sector erase suspend command. This command can be executed by writing the erase restart code (30<sub>H</sub>), for which the address should indicate arbitrary address in the flash memory area.

Further, issuing the sector erase restart command is ignored while erasing a sector.



# **CHAPTER 26**

---

## ***EXAMPLE of CONNECTING SERIAL WRITING***

***(FLASH MICROCONTROLLER PROGRAMMER MODE  
by YOKOGAWA DIGITAL COMPUTER CORPORATION)***

**This chapter describes examples of serial write connection when using flash microcontroller programmer mode by Yokogawa Digital Computer Corporation.**

- 26.1 Basic Configuration
- 26.2 Oscillation Clock Frequency and Serial Clock Input Frequency
- 26.3 Flash Microcontroller Programmer System Configuration
- 26.4 Example of Connecting Serial Writing



## 26.1 Basic Configuration

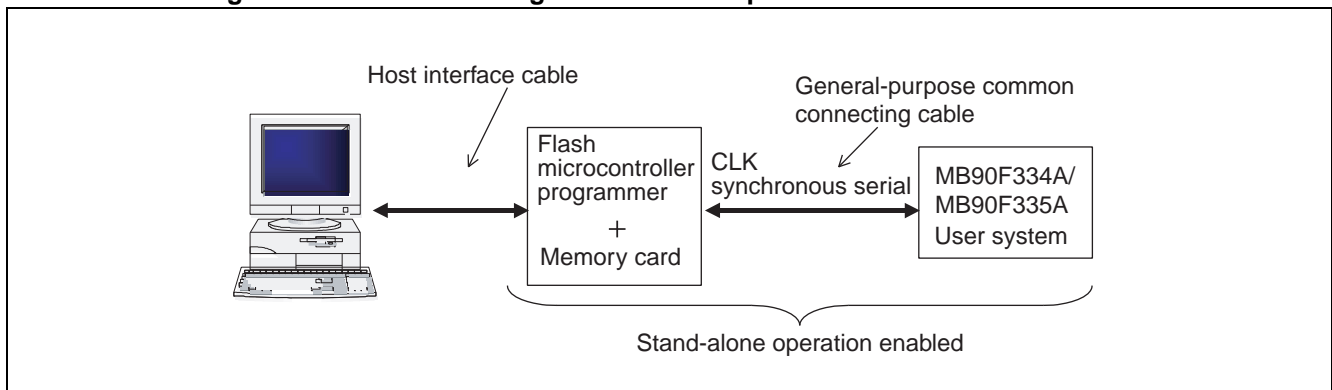
The MB90F334A and MB90F335A support the serial on-board programming of flash ROM (Fujitsu Microelectronics standard). The specification for serial on-board programming are explained below.

### ■ The Serial On-board Writing Basic Component

The flash microcontroller programmer made by Yokogawa Digital Computer Corporation is used for Fujitsu standard serial on-board programming. It is possible to choose between the program operated in single-chip mode and the program operated in internal ROM external bus mode and to write.

Figure 26.1-1 shows the basic configuration of serial write connection examples.

**Figure 26.1-1 Basic Configuration of Example for Serial Write Connection**



Note:

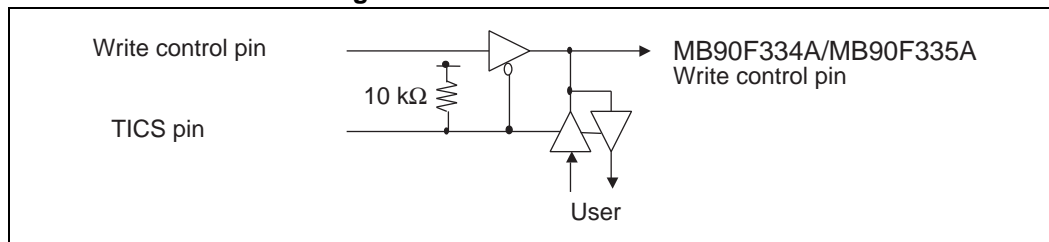
Contact Yokogawa Digital Computer Corporation for details of the functions, operations, general-purpose common connecting cable and applicable connectors of the flash microcontroller programmer.

**MB90330A Series****■ Pins Used for Fujitsu Standard Serial On-board Programming**

Table 26.1-1 shows the function of pins used for Fujitsu standard serial on-board programming.

**Table 26.1-1 Function of Used Pins**

Pin	Function	Supplementary Information
MD2,MD1,MD0	Mode Pin	Setting MD2=1, MD1=1, and MD0=0 allows the mode to be in the serial write mode.
X0,X1	Oscillation pins	Because the internal CPU operation clock is set to be the 1 multiplication PLL clock in the serial write mode, the internal operation clock frequency is the same as the oscillation clock frequency. When you perform the serial write, the frequency you can input into the high-speed oscillation input pin is fixed to 6 MHz.
P60,P61	Writing program activating pins	Input a "L" level to P60 and a "H" level to P61.
$\overline{\text{RST}}$	Reset	-
SIN0	Serial data input	UART0 is used as CLK synchronous mode.
SOT0	Serial data output	
SCK0	Serial clock input	
V <sub>CC</sub>	Supply voltage	The write voltage (V <sub>CC</sub> =3.3 V ± 0.3 V)
V <sub>SS</sub>	GND	GND pin is common to the ground of the flash microcontroller programmer.

**Figure 26.1-2 Pin Control Circuit****Notes:**

- When the P60, P61, SIN0, SOT0, or SCK0 pin is used also in a user system, the control circuit shown in Figure 26.1-2 is required.
- During serial write, the user circuit can be disconnected using the /TICS signal from the flash microcontroller programmer. Please refer to Section "26.4 Example of Connecting Serial Writing".

## 26.2 Oscillation Clock Frequency and Serial Clock Input Frequency

---

The MB90F334A and MB90F335A serial clock frequencies that can be input are determined by the following expression:

Thus, set up the flash microcontroller programmer to change the serial clock input frequency corresponding to the using oscillation clock frequency.

---

### ■ Oscillation Clock Frequency and Serial Clock Input Frequency

Serial clock frequencies that can be input are determined using the expression below.

Imputable serial clock frequency =  $0.125 \times$  oscillation clock frequency.

Table 26.2-1 shows the serial clock frequencies that can be input.

**Table 26.2-1 Serial Clock Frequency that can be Input**

Oscillation clock frequency	Maximum serial clock frequency that can be input to the microcontroller	Maximum serial clock frequency that can be set
in 6 MHz	750 kHz	500 kHz

## 26.3 Flash Microcontroller Programmer System Configuration

Table 26.3-1 shows the system configuration of the flash microcontroller programmer.

### ■ Flash Microcontroller Programmer System

Table 26.3-1 System Configuration of Flash Microcontroller Programme

Product name	Part number	Description
Flash Microcontroller Programmer Main Unit	AF420/AC4P	FULL KEY model 100BASE-TX host interface
	AF620/AC4P	FULL KEY model CAN interface 100BASE-TX host interface
	AF320/AC4P	Single KEY model 100BASE-TX host interface
	AF520/AC4P	Single KEY model CAN interface 100BASE-TX host interface
Standard Target Probe	AZ410	Standard Target Probe (a): 1 m
Compact Modules	FF801	Control modules for Fujitsu Microelectronics microcontroller
Remote Controller	AZ490	Remote Controller
Memory card	-	PC Card

Contact: Yokogawa Digital Computer Corporation Tel: + 81-42-333-6224

#### Note:

Sales of the AF2xx/AF1xx series products have been terminated since March 31, 2007. The maintenance (repair) service of the products will be provided for five years from the date of the sales termination. The customer who hopes for the new or additional purchase of the product might consider the purchase of the succession product (AF400/300 and AF600/500 series) in the future. For details, contact Yokokawa Digital Computer Ltd. We will continue the support for Fujitsu microcomputer successors.

## **26.4 Example of Connecting Serial Writing**

---

**The examples of serial write connection is shown.**

---

### **■ Example of Connecting Serial Writing**

Example of connecting serial writing has following two types.

- Connection example in Single-chip mode (when using user power)
- Example of minimum connection to flash microcontroller programmer (when using user power)

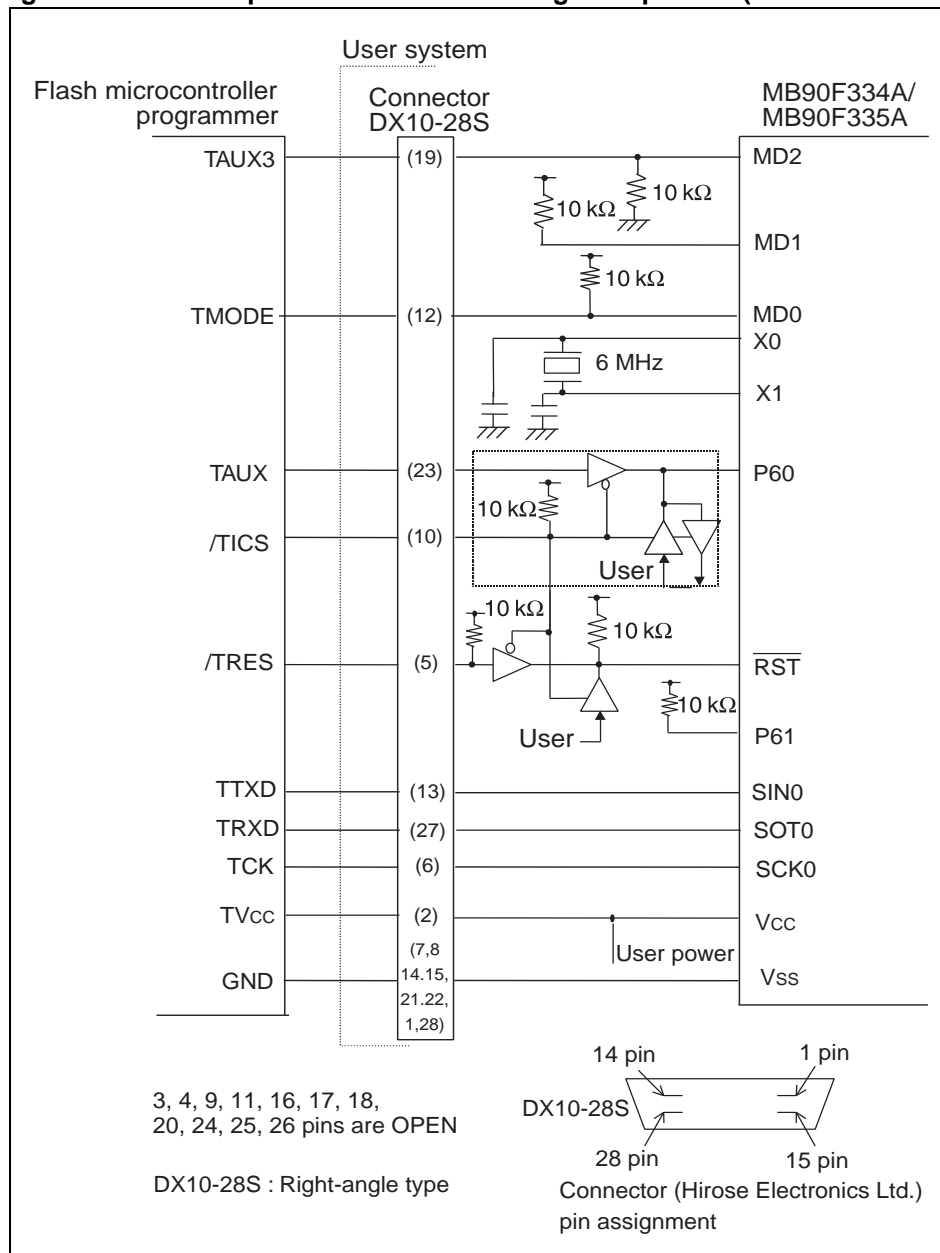
## MB90330A Series

### 26.4.1 Example Connection in Single-chip Mode (when Using User Power)

In a user system, from TAUX3 and TMODE of flash microcontroller programmer, "1" for MD1 and "0" for MD0 are input to the MD2 and MD0 mode pins, which have been set to the single chip mode; this changes the mode to serial write mode. (Serial write mode:MD2, MD1, and MD0=110)

#### ■ Connection Example in Single-chip Mode (when Using User Power)

Figure 26.4-1 Example of Connection in Single Chip Mode (with User Power)



---

Notes:

- When the SIN0, SOT0, or SCK0 pin is used also in a user system, the control circuit shown in Figure 26.1-2 is required like P60. (During serial write, the user circuit can be disconnected using the /TICS signal from the flash microcontroller programmer.)
  - Connect the flash microcontroller programmer while the user power is off.
-

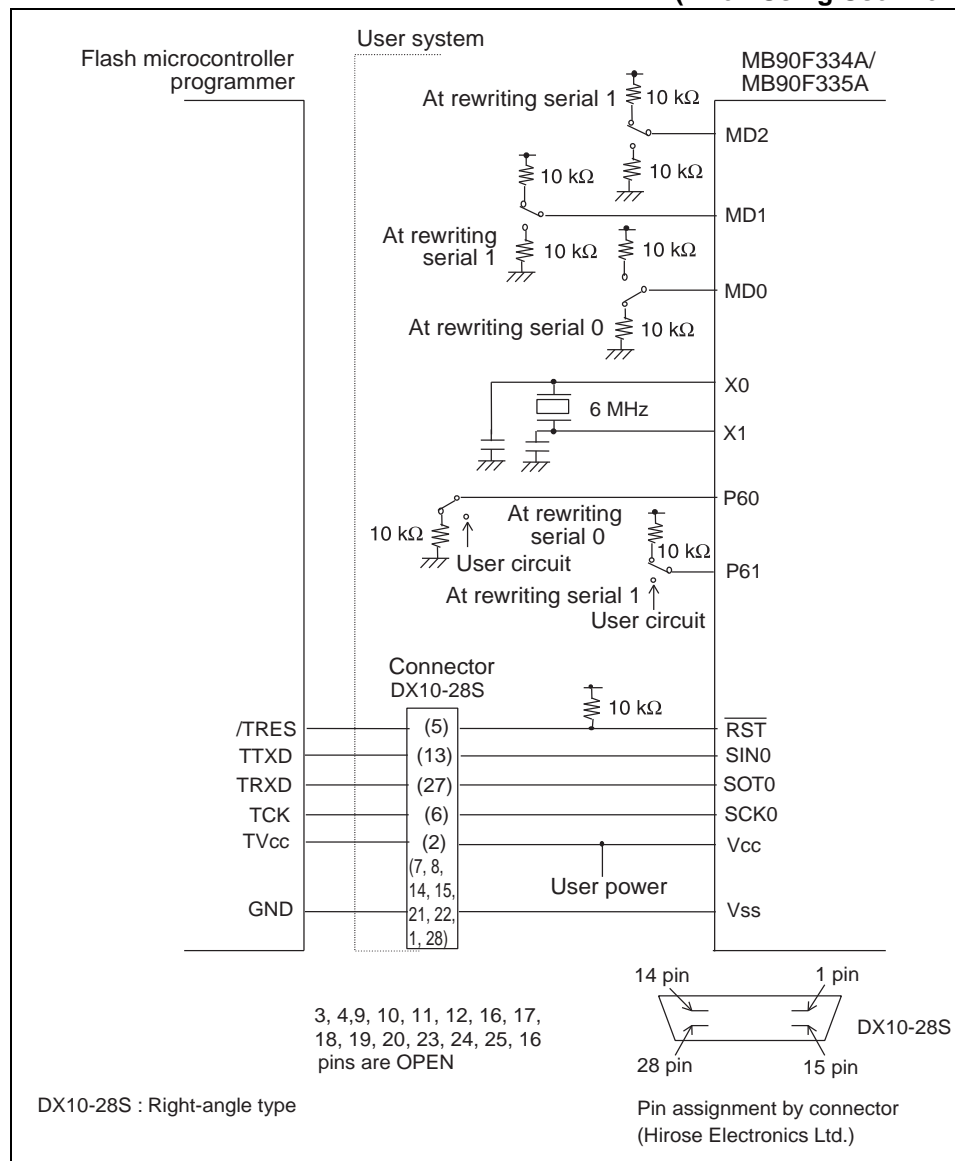
## MB90330A Series

### 26.4.2 Example of Minimum Connection to Flash Microcontroller Programmer (when Using User Power)

During serial write, when MD2, MD0, and P60 pins are set as shown in Figure 26.4-2, it is unnecessary to connect these pins with the flash microcontroller programmer.

#### ■ Example of Minimum Connection to Flash Microcontroller Programmer (when Using User Power)

Figure 26.4-2 Example of Minimum Connection to Flash Microcontroller Programmer (when Using User Power)





---

Notes:

- When the SIN0, SOT0, or SCK0 pin is used also in a user system, the control circuit shown in Figure 26.1-2 is required. (During serial write, the user circuit can be disconnected using the /TICS signal from the flash microcontroller programmer.)
  - Connect the flash microcontroller programmer while the user power is off.
-

# ***CHAPTER 27***

---

## ***SERIAL PROGRAMMING CONNECTION***

### ***(FUJITSU MICROELECTRONICS SERIAL PROGRAMMER)***

**MB90F334A/335A supports serial onboard write (Fujitsu Microelectronics standard) to flash memory.**

**This chapter explains the basic configuration for serial write to flash memory by using the Fujitsu Microelectronics Serial Programmer.**

## 27.1 Fujitsu Microelectronics Serial Programmer

**Fujitsu Microelectronics Serial Programmer (software) is an onboard programming tool for all Fujitsu Microelectronics-made controllers with built-in flash memory.**

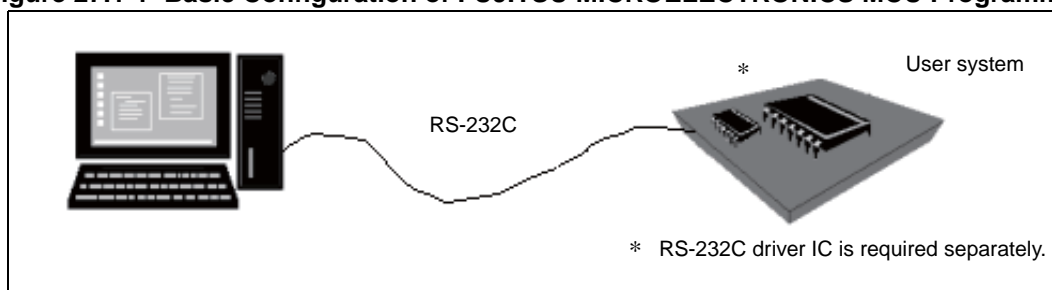
**Two types of Serial Programmer are available according to the PC interface (RS-232C or USB) used. Choose the type according to your environment.**

### ■ Basic Configuration of FUJITSU MICROELECTRONICS MCU Programmer (Clock Asynchronous Serial Write)

FUJITSU MICROELECTRONICS MCU Programmer is used when the PC and microcontroller are connected through an RS-232C cable. MCU Programmer writes data, through clock asynchronous serial communication, to built-in flash memory of a microcontroller installed in the user system.

Figure 27.1-1 shows the basic configuration of FUJITSU MICROELECTRONICS MCU Programmer, and Table 27.1-1 lists the system configuration.

**Figure 27.1-1 Basic Configuration of FUJITSU MICROELECTRONICS MCU Programmer**



**Table 27.1-1 System Configuration of FUJITSU MICROELECTRONICS MCU Programmer**

Name	Type	Specifications
FUJITSU MICROELECTRONICS MCU Programmer	-	Software (can be downloaded from Web (registration system))*

\* For registration, contact your sales representatives.

Figure 27.1-2 shows a connection example.

Figure 27.1-2 Connection Example using FUJITSU MICROELECTRONICS MCU Programmer

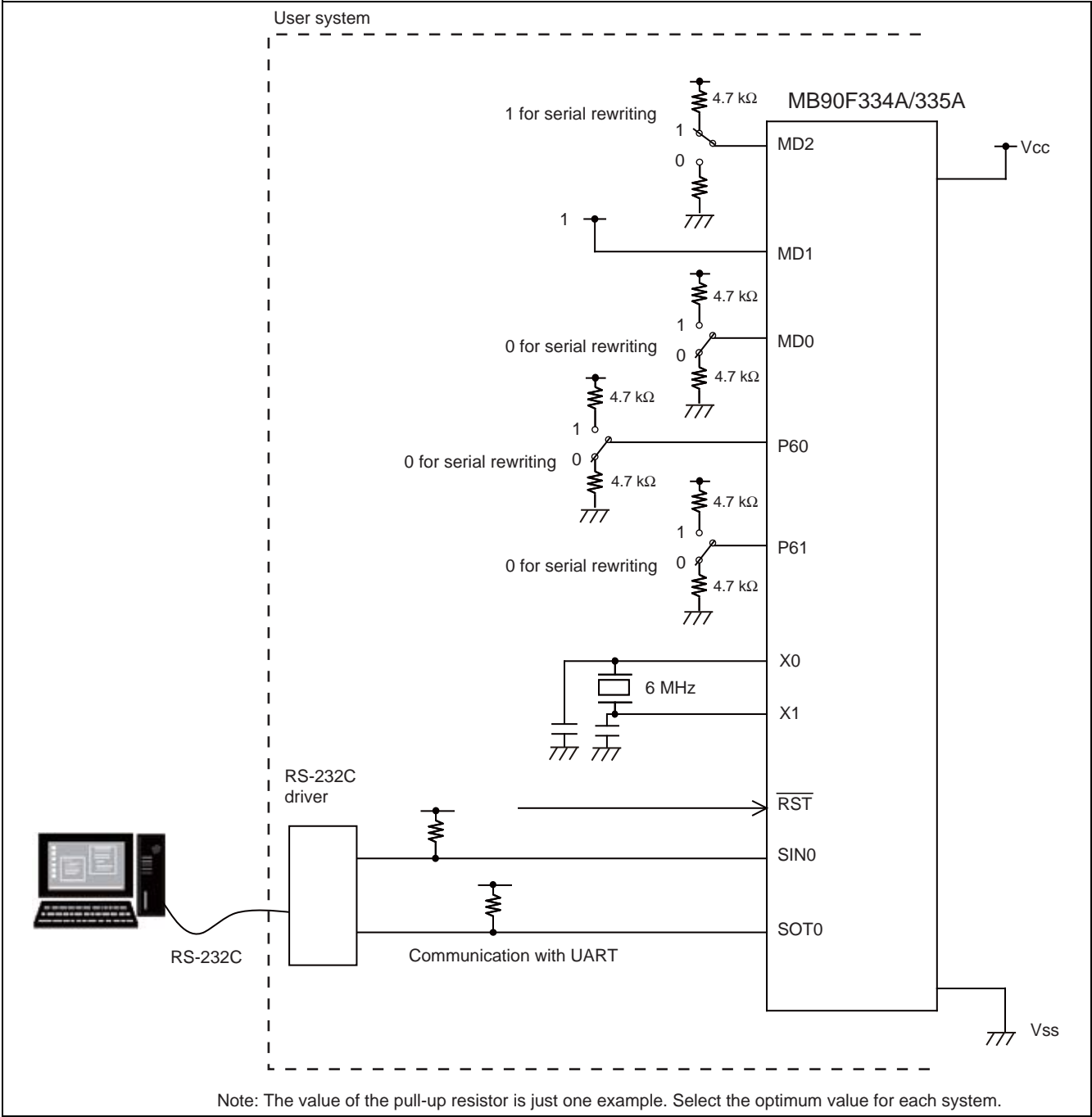


Table 27.1-2 Oscillating frequency and communication baud rate available for clock asynchronous serial communication

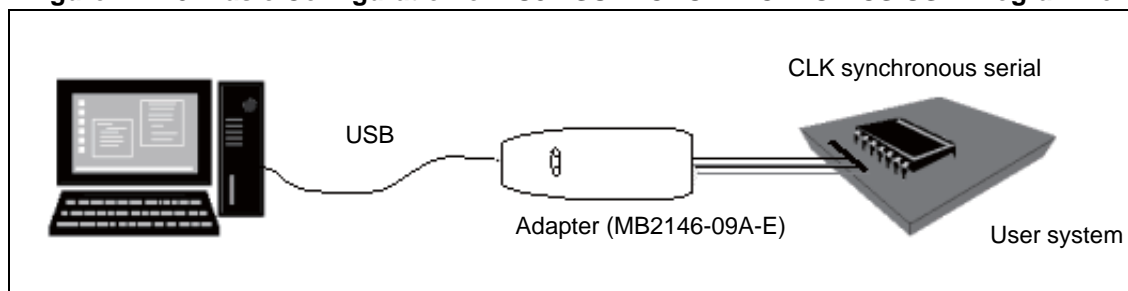
Master Oscillating Frequency	Communication Baud Rate
6 MHz	19200 bps

## ■ Basic Configuration of FUJITSU MICROELECTRONICS USB Programmer (Clock Synchronous Serial Write)

FUJITSU MICROELECTRONICS USB Programmer is used when the PC and microcontroller are connected through an adapter (MB2146-09A-E). USB Programmer writes data, through clock synchronous serial communication, to built-in flash memory of a microcontroller.

Figure 27.1-3 shows the basic configuration of FUJITSU MICROELECTRONICS USB Programmer, and Table 27.1-3 lists the system configuration.

**Figure 27.1-3 Basic Configuration of FUJITSU MICROELECTRONICS USB Programmer**



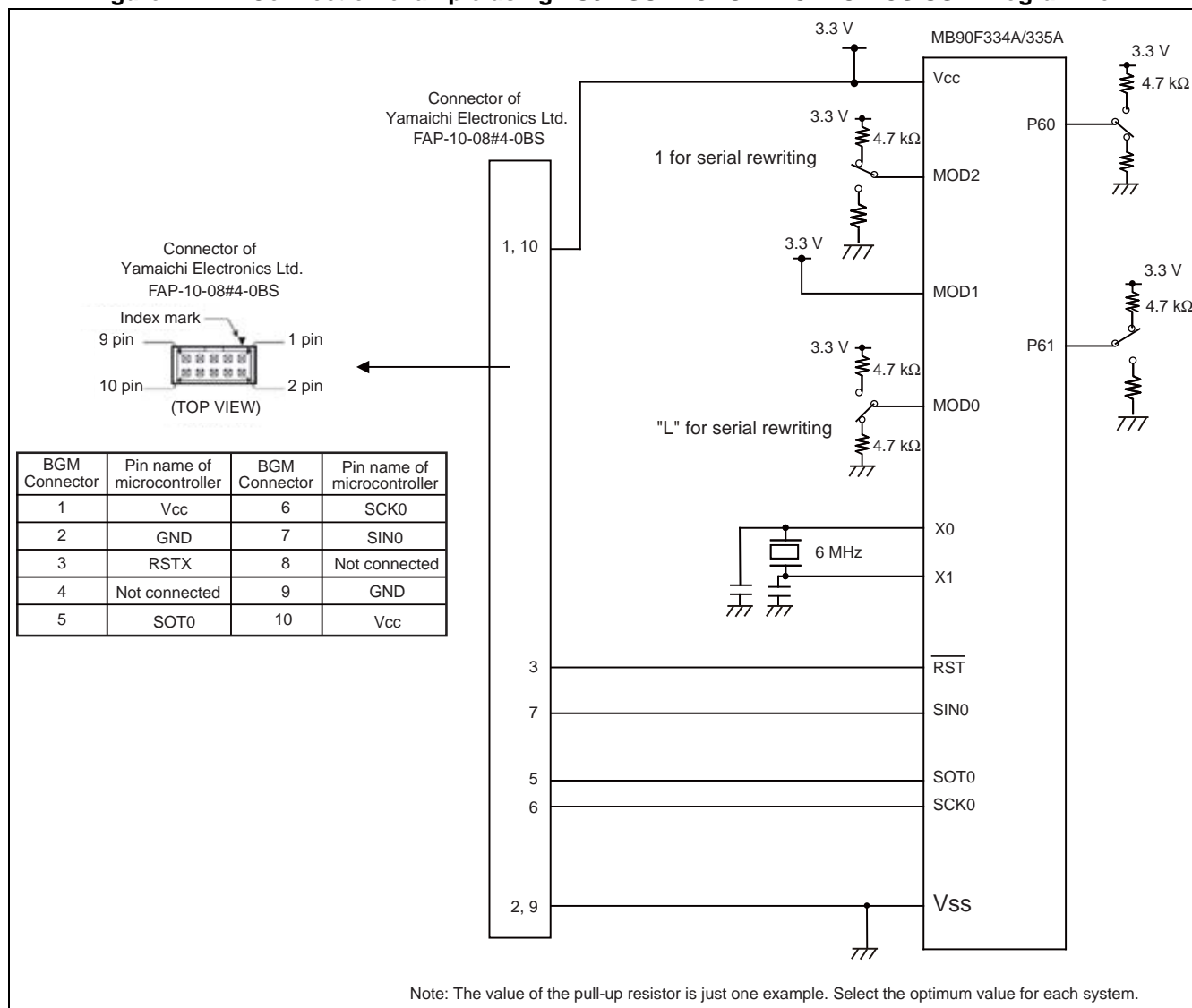
**Table 27.1-3 System Configuration of FUJITSU MICROELECTRONICS USB Programmer**

Name	Type	Specifications
FUJITSU MICROELECTRONICS USB Programmer	-	Software (can be downloaded from Web (registration system))*
Adapter	MB2146-09A-E	F <sup>2</sup> MC family BGM adapter (Accessory: USB cable)

\* For registration, contact your sales representatives.

Figure 27.1-4 shows a connection example.

**Figure 27.1-4 Connection example using FUJITSU MICROELECTRONICS USB Programmer**



## 27.1.1 Pins Used

**Table 27.1-4 Function of Used Pins**

Pin	Function	Supplementary Information
MD2,MD1,MD0	Mode Pin	Setting MD2=1, MD1=1, and MD0=0 allows the mode to be in the serial write mode.
X0,X1	Oscillation pins	Because the internal CPU operation clock is set to be the 1 multiplication PLL clock in the serial write mode, the internal operation clock frequency is the same as the oscillation clock frequency. When you perform the serial write, the frequency you can input into the high-speed oscillation input pin is fixed to 6 MHz.
P60,P61	Writing program activating pins	For CLK synchronous serial writing, input "L" level to P60 and "H" level to P61. For CLK asynchronous serial writing, input "L" level to P60 and P61.
$\overline{\text{RST}}$	Reset	-
SIN0	Serial data input	UART0 is used in CLK synchronous mode/CLK asynchronous mode. Note: SCK0 is not used in CLK asynchronous mode.
SOT0	Serial data output	
SCK0	Serial clock input	
V <sub>CC</sub>	Supply voltage	The write voltage (V <sub>CC</sub> =3.3 V ± 0.3 V)
V <sub>SS</sub>	GND	GND pin is common to the ground of the flash microcontroller programmer.

# ***APPENDIX***

---

**The appendix describes the memory map and the instructions used in the F<sup>2</sup>MC-16LX.**

APPENDIX A Memory Map

APPENDIX B Instructions

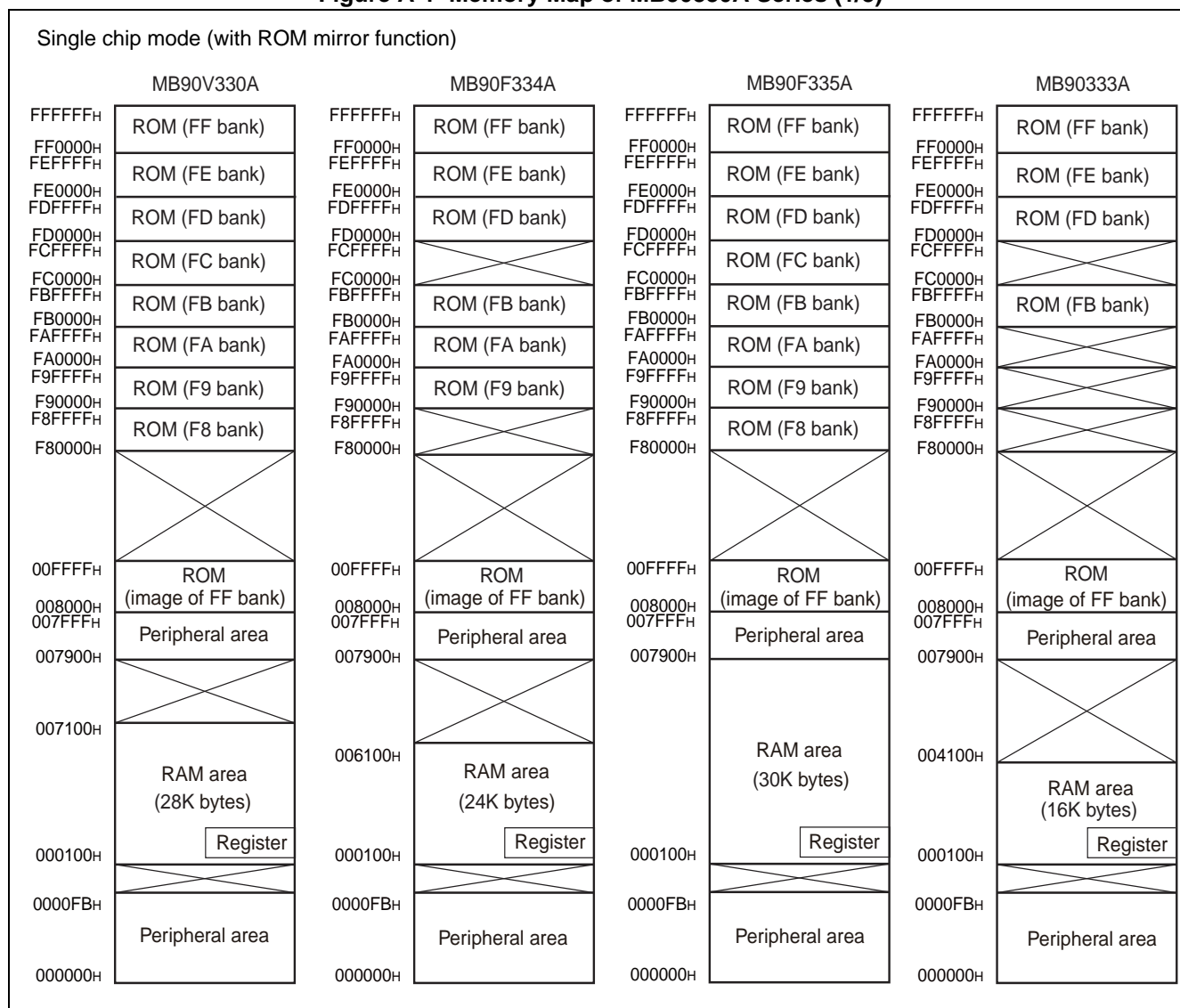


## APPENDIX A Memory Map

The memory space divides into three modes.

### ■ Memory Map

Figure A-1 Memory Map of MB90330A Series (1/3)

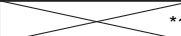

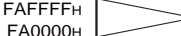







---

Notes:

- For setting the ROM mirror function, see "23. ROM mirror function selection module".
  - When the ROM mirror function register has been set, the mirror image data at higher addresses ("FF8000<sub>H</sub> to FFFFFFF<sub>H</sub>") of bank FF is visible from the higher addresses ("008000<sub>H</sub> to 00FFFF<sub>H</sub>") of bank 00.
  - The ROM mirror function is effective for using the C compiler small model.
  - The lower 16-bit addresses of bank FF are equivalent to those of bank 00. Since the ROM area in bank FF exceeds 48 Kbytes, however, the mirror image of all the data in the ROM area cannot be reproduced in bank 00.
  - When the C compiler small model is used, the data table mirror image can be shown at "008000<sub>H</sub> to 00FFFF<sub>H</sub>" by storing the data table at "FF8000<sub>H</sub> to FFFFFFF<sub>H</sub>". Therefore, data tables in the ROM area can be referred without declaring the far addressing with the pointer.
  - MB90F335A has the larger size of RAM area than MB90V330A, so that the emulation memory area needs to be set in the tools for a larger size of emulation area than 007100<sub>H</sub>. For details of setting, please refer to "Notes on Debug Environment Setting for MB90330A Series" by clicking "Application note" at the following URL.  
<http://edevic.fujitsu.com/micom/en-support/>
  - Access to the emulation memory area (007100<sub>H</sub> to 0078FF<sub>H</sub>) is 3 cycles, which is greater by one cycle than access to the mounted RAM area.
-

**Figure A-2 Memory Map of MB90330A Series (2/3)**

Internal ROM external bus mode (with ROM mirror function)			
MB90V330A	MB90F334A	MB90F335A	MB90333A
FFFFF <sub>H</sub> FF000 <sub>H</sub> FEFFF <sub>H</sub> FE000 <sub>H</sub> FDFFF <sub>H</sub> FD000 <sub>H</sub> FCFFF <sub>H</sub> FC000 <sub>H</sub> FBFFF <sub>H</sub> FB000 <sub>H</sub> FAFFF <sub>H</sub> FA000 <sub>H</sub> F9FFF <sub>H</sub> F9000 <sub>H</sub> F8FFF <sub>H</sub> F8000 <sub>H</sub>	FFFFF <sub>H</sub> FF000 <sub>H</sub> FEFFF <sub>H</sub> FE000 <sub>H</sub> FDFFF <sub>H</sub> FD000 <sub>H</sub> FCFFF <sub>H</sub> FC000 <sub>H</sub> FBFFF <sub>H</sub> FB000 <sub>H</sub> FAFFF <sub>H</sub> FA000 <sub>H</sub> F9FFF <sub>H</sub> F9000 <sub>H</sub> F8FFF <sub>H</sub> F8000 <sub>H</sub>	FFFFF <sub>H</sub> FF000 <sub>H</sub> FEFFF <sub>H</sub> FE000 <sub>H</sub> FDFFF <sub>H</sub> FD000 <sub>H</sub> FCFFF <sub>H</sub> FC000 <sub>H</sub> FBFFF <sub>H</sub> FB000 <sub>H</sub> FAFFF <sub>H</sub> FA000 <sub>H</sub> F9FFF <sub>H</sub> F9000 <sub>H</sub> F8FFF <sub>H</sub> F8000 <sub>H</sub>	FFFFF <sub>H</sub> FF000 <sub>H</sub> FEFFF <sub>H</sub> FE000 <sub>H</sub> FDFFF <sub>H</sub> FD000 <sub>H</sub> FCFFF <sub>H</sub> FC000 <sub>H</sub> FBFFF <sub>H</sub> FB000 <sub>H</sub> FAFFF <sub>H</sub> FA000 <sub>H</sub> F9FFF <sub>H</sub> F9000 <sub>H</sub> F8FFF <sub>H</sub> F8000 <sub>H</sub>
ROM (FF bank)	ROM (FF bank)	ROM (FF bank)	ROM (FF bank)
ROM (FE bank)	ROM (FE bank)	ROM (FE bank)	ROM (FE bank)
ROM (FD bank)	ROM (FD bank)	ROM (FD bank)	ROM (FD bank)
ROM (FC bank)	 *1	ROM (FC bank)	 *2
ROM (FB bank)	ROM (FB bank)	ROM (FB bank)	ROM (FB bank)
ROM (FA bank)	ROM (FA bank)	ROM (FA bank)	 *2
ROM (F9 bank)	ROM (F9 bank)	ROM (F9 bank)	External area
ROM (F8 bank)	 *1	ROM (F8 bank)	External area
External area	External area	External area	External area
00FFF <sub>H</sub> 00800 <sub>H</sub> 007FF <sub>H</sub> 00790 <sub>H</sub>	00FFF <sub>H</sub> 00800 <sub>H</sub> 007FF <sub>H</sub> 00790 <sub>H</sub>	00FFF <sub>H</sub> 00800 <sub>H</sub> 007FF <sub>H</sub> 00790 <sub>H</sub>	00FFF <sub>H</sub> 00800 <sub>H</sub> 007FF <sub>H</sub> 00790 <sub>H</sub>
ROM (image of FF bank)	ROM (image of FF bank)	ROM (image of FF bank)	ROM (image of FF bank)
Peripheral area	Peripheral area	Peripheral area	Peripheral area
External area	External area		External area
00710 <sub>H</sub>	00610 <sub>H</sub>		00410 <sub>H</sub>
RAM (28K bytes)	RAM (24K bytes)	RAM (30K bytes)	RAM (16K bytes)
00010 <sub>H</sub>	00010 <sub>H</sub>	00010 <sub>H</sub>	00010 <sub>H</sub>
Register	Register	Register	Register
			
0000FB <sub>H</sub>	0000FB <sub>H</sub>	0000FB <sub>H</sub>	0000FB <sub>H</sub>
Peripheral area	Peripheral area	Peripheral area	Peripheral area
00000 <sub>H</sub>	00000 <sub>H</sub>	00000 <sub>H</sub>	00000 <sub>H</sub>

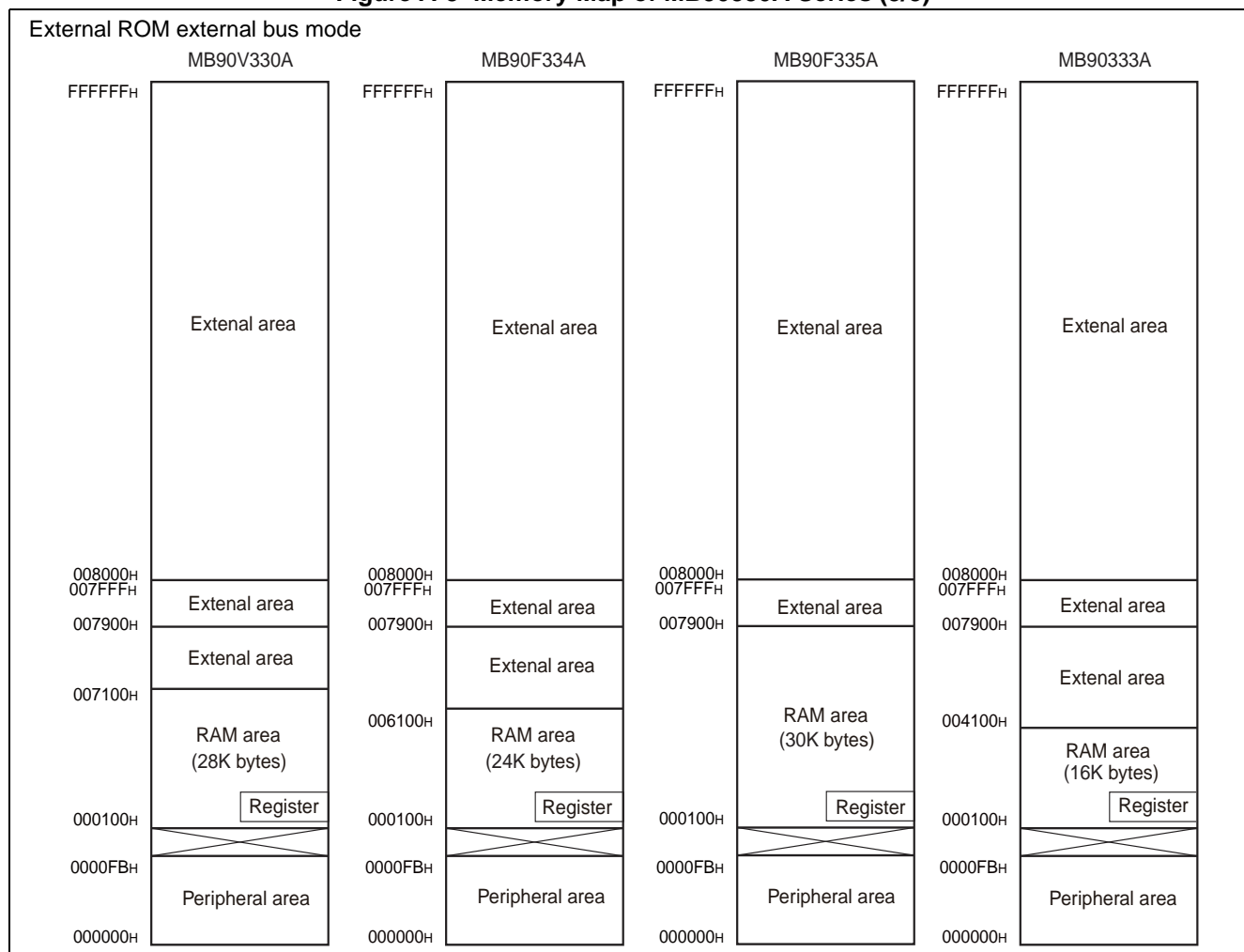
\*1: In the area of F8000<sub>H</sub> to F8FFF<sub>H</sub> and FC000<sub>H</sub> to FCFFF<sub>H</sub> at MB90F334 a value of "1" is read at read operating.

\*2: In the area of FA000<sub>H</sub> to FAFFF<sub>H</sub> and FC000<sub>H</sub> to FCFFF<sub>H</sub> at MB90333, a value of "1" is read at read operating.

**Notes:**

- For setting the ROM mirror function, see "23. ROM mirror function selection module" in "■ PERIPHERAL RESOURCES".
- When the ROM mirror function register has been set, the mirror image data at higher addresses ("FF8000<sub>H</sub> to FFFFF<sub>H</sub>") of bank FF is visible from the higher addresses ("008000<sub>H</sub> to 00FFF<sub>H</sub>") of bank 00.
- The ROM mirror function is effective for using the C compiler small model.
- The lower 16-bit addresses of bank FF are equivalent to those of bank 00. Since the ROM area in bank FF exceeds 48 Kbytes, however, the mirror image of all the data in the ROM area cannot be reproduced in bank 00.
- When the C compiler small model is used, the data table mirror image can be shown at "008000<sub>H</sub> to 00FFF<sub>H</sub>" by storing the data table at "FF8000<sub>H</sub> to FFFFF<sub>H</sub>". Therefore, data tables in the ROM area can be referred without declaring the far addressing with the pointer.
- MB90F335A has the larger size of RAM area than MB90V330A, so that the emulation memory area needs to be set in the tools for a larger size of emulation area than 007100<sub>H</sub>. For details of setting, please refer to "Notes on Debug Environment Setting for MB90330A Series" by clicking "Application note" at the following URL.  
<http://edevic.fujitsu.com/micom/en-support/>
- Access to the emulation memory area (007100<sub>H</sub> to 0078FF<sub>H</sub>) is 3 cycles, which is greater by one cycle than access to the mounted RAM area.

Figure A-3 Memory Map of MB90330A Series (3/3)



Notes:

- For setting the ROM mirror function, see "23. ROM mirror function selection module" in "■ PERIPHERAL RESOURCES".
- When the ROM mirror function register has been set, the mirror image data at higher addresses ("FF8000<sub>H</sub> to FFFFFFF<sub>H</sub>") of bank FF is visible from the higher addresses ("008000<sub>H</sub> to 00FFFF<sub>H</sub>") of bank 00.
- The ROM mirror function is effective for using the C compiler small model.
- The lower 16-bit addresses of bank FF are equivalent to those of bank 00. Since the ROM area in bank FF exceeds 48 Kbytes, however, the mirror image of all the data in the ROM area cannot be reproduced in bank 00.
- When the C compiler small model is used, the data table mirror image can be shown at "008000<sub>H</sub> to 00FFFF<sub>H</sub>" by storing the data table at "FF8000<sub>H</sub> to FFFFFFF<sub>H</sub>". Therefore, data tables in the ROM area can be referred without declaring the far addressing with the pointer.
- MB90F335A has the larger size of RAM area than MB90V330A, so that the emulation memory area needs to be set in the tools for a larger size of emulation area than 007100<sub>H</sub>. For details of setting, please refer to "Notes on Debug Environment Setting for MB90330A Series" by clicking "Application note" at the following URL.  
<http://edevic.fujitsu.com/micom/en-support/>
- Access to the emulation memory area (007100<sub>H</sub> to 0078FF<sub>H</sub>) is 3 cycles, which is greater by one cycle than access to the mounted RAM area.

■ I/O Map

Table A-1 lists addresses assigned to registers of each peripheral function.

**Table A-1 I/O Map (1 / 9)**

Address	Registers	Abbreviation	Access	Release	Initial value
000000 <sub>H</sub>	Port 0 data register	PDR0	R/W	Port 0	XXXXXXXX
000001 <sub>H</sub>	Port 1 data register	PDR1	R/W	Port 1	XXXXXXXX
000002 <sub>H</sub>	Port 2 data register	PDR2	R/W	Port 2	XXXXXXXX
000003 <sub>H</sub>	Port 3 data register	PDR3	R/W	Port 3	XXXXXXXX
000004 <sub>H</sub>	Port 4 data register	PDR4	R/W	Port 4	XXXXXXXX
000005 <sub>H</sub>	Port 5 data register	PDR5	R/W	Port 5	XXXXXXXX
000006 <sub>H</sub>	Port 6 data register	PDR6	R/W	Port 6	XXXXXXXX
000007 <sub>H</sub>	Port 7 data register	PDR7	R/W	Port 7	XXXXXXXX
000008 <sub>H</sub>	Port 8 data register	PDR8	R/W	Port 8	XXXXXXXX
000009 <sub>H</sub>	Port 9 data register	PDR9	R/W	Port 9	-XXXXXXXX
00000A <sub>H</sub>	Port A data register	PDRA	R/W	Port A	XXXXXXXX
00000B <sub>H</sub>	Use prohibited				
00000C <sub>H</sub>	Port B data register	PDRB	R/W	Port B	-XXXXXXXX
00000D <sub>H</sub>	Port B direction register	DDRB	R/W	Port B	-0000000
00000E <sub>H</sub>	Use prohibited				
00000F <sub>H</sub>					
000010 <sub>H</sub>	Port 0 data direction register	DDR0	R/W	Port 0	00000000
000011 <sub>H</sub>	Port 1 direction register	DDR1	R/W	Port 1	00000000
000012 <sub>H</sub>	Port 2 direction register	DDR2	R/W	Port 2	00000000
000013 <sub>H</sub>	Port 3 direction register	DDR3	R/W	Port 3	00000000
000014 <sub>H</sub>	Port 4 direction register	DDR4	R/W	Port 4	00000000
000015 <sub>H</sub>	Port 5 direction register	DDR5	R/W	Port 5	00000000
000016 <sub>H</sub>	Port 6 direction register	DDR6	R/W	Port 6	00000000
000017 <sub>H</sub>	Port 7 direction register	DDR7	R/W	Port 7	00000000
000018 <sub>H</sub>	Port 8 direction register	DDR8	R/W	Port 8	00000000
000019 <sub>H</sub>	Port 9 direction register	DDR9	R/W	Port 9	-0000000
00001A <sub>H</sub>	Port A direction register	DDRA	R/W	Port A	00000000
00001B <sub>H</sub>	Port 4 output terminal register	ODR4	R/W	Port 4 (OD control)	00000000
00001C <sub>H</sub>	Port 0 Pull-up resistance register	RDR0	R/W	Port 0(PULLUP)	00000000
00001D <sub>H</sub>	Port 1 Pull-up resistance register	RDR1	R/W	Port 1(PULLUP)	00000000
00001E <sub>H</sub>	Analog input enable register 0	ADER0	R/W	Port 7,A/D	11111111
00001F <sub>H</sub>	Analog input enable register 1	ADER1	R/W	Port 8,A/D	11111111

**Table A-1 I/O Map (2 / 9)**

Address	Registers	Abbreviation	Access	Release	Initial value
000020 <sub>H</sub>	Serial mode register 1	SMR0	R/W	UART0	00100000
000021 <sub>H</sub>	Serial control register 0	SCR0	R/W,W		00000100
000022 <sub>H</sub>	Serial input data register 0/serial output data register 0	SIDR0/ SODR0	R/W		XXXXXXXX
000023 <sub>H</sub>	Serial Status Register 0	SSR0	R/W,R		00001000
000024 <sub>H</sub>	UART prescaler reload register 0	UTRLR0	R/W	Communication Prescaler (UART0)	00000000
000025 <sub>H</sub>	UART prescaler control register 0	UTCR0	R/W		0000-000
000026 <sub>H</sub>	Serial mode register 1	SMR1	R/W	UART1	00100000
000027 <sub>H</sub>	Serial control register 1	SCR1	R/W,W		00000100
000028 <sub>H</sub>	Serial input data register 1/serial output data register 1	SIDR1/ SODR1	R/W		XXXXXXXX
000029 <sub>H</sub>	Serial status register 1	SSR1	R/W,R		00001000
00002A <sub>H</sub>	UART prescaler reload register 1	UTRLR1	R/W	Communication Prescaler (UART1)	00000000
00002B <sub>H</sub>	UART prescaler control register 1	UTCR1	R/W		0000-000
00002C <sub>H</sub>	Serial mode register 2	SMR2	R/W	UART2	00100000
00002D <sub>H</sub>	Serial control register 2	SCR2	R/W,W		00000100
00002E <sub>H</sub>	Serial input data register 2/serial output data register 2	SIDR2/ SODR2	R/W		XXXXXXXX
00002F <sub>H</sub>	Serial Status Register 2	SSR2	R/W,R		00001000
000030 <sub>H</sub>	UART prescaler reload register 2	UTRLR2	R/W	Communication Prescaler (UART2)	00000000
000031 <sub>H</sub>	UART prescaler control register 2	UTCR2	R/W		0000-000
000032 <sub>H</sub>	Serial mode register 3	SMR3	R/W	UART3	00100000
000033 <sub>H</sub>	Serial control register 3	SCR3	R/W,W		00000100
000034 <sub>H</sub>	Serial input data register 3/serial output data register 3	SIDR3/ SODR3	R/W		XXXXXXXX
000035 <sub>H</sub>	Serial Status Register 3	SSR3	R/W,R		00001000
000036 <sub>H</sub>	UART prescaler reload register 3	UTRLR3	R/W	Communication Prescaler (UART3)	00000000
000037 <sub>H</sub>	UART prescaler control register 3	UTCR3	R/W		0000-000
000038 <sub>H</sub> to 00003B <sub>H</sub>	Use prohibited				
00003C <sub>H</sub>	DTP/interruption permission register	ENIR	R/W	DTP/external interrupt	00000000
00003D <sub>H</sub>	DTP/interruption request register	EIRR	R/W		00000000
00003E <sub>H</sub>	Request level set register subordinate position	ELVR	R/W		00000000
00003F <sub>H</sub>	High rank of request level set register		R/W		00000000

**Table A-1 I/O Map (3 / 9)**

Address	Registers	Abbreviation	Access	Release	Initial value
000040 <sub>H</sub>	A/D Control Status Register (Low)	ADCS0	R/W	8/10-bit A/D converter	00-----0
000041 <sub>H</sub>	A/D Control Status Register (High)	ADCS1	R/W,W		00000000
000042 <sub>H</sub>	A/D Data Register (Low)	ADCR0	R		XXXXXXXXXX
000043 <sub>H</sub>	A/D Data Register (High)	ADCR1	W,R		00101XXX
000044 <sub>H</sub>	Use prohibited				
000045 <sub>H</sub>	Analog to digital conversion channel selection register	ADMR	R/W	8/10-bit A/D converter	00000000
000046 <sub>H</sub>	PPG0 operation mode control register	PPGC0	R/W	PPG ch.0	0X000XX1
000047 <sub>H</sub>	PPG1 operation mode control register	PPGC1	R/W	PPG ch.1	0X000001
000048 <sub>H</sub>	PPG2 operation mode control register	PPGC2	R/W	PPG ch.2	0X000XX1
000049 <sub>H</sub>	PPG3 operation mode control register	PPGC3	R/W	PPG ch.3	0X000001
00004A <sub>H</sub>	PPG4 operation mode control register	PPGC5	R/W	PPG ch.4	0X000XX1
00004B <sub>H</sub>	PPG5 operation mode control register	PPGC6	R/W	PPG ch.5	0X000001
00004C <sub>H</sub>	PPG0,1 output control register	PPG01	R/W	PPG ch.0/1	000000XX
00004D <sub>H</sub>	Use prohibited				
00004E <sub>H</sub>	PPG2,3 output control register	PPG23	R/W	PPG ch.2/3	000000XX
00004F <sub>H</sub>	Use prohibited				
000050 <sub>H</sub>	PPG4,5 output control register	PPG45	R/W	PPG ch.4/5	000000XX
000051 <sub>H</sub>	Use prohibited				
000052 <sub>H</sub>	Input capture control status register 01	ICS01	R/W	Input capture 0/1	00000000
000053 <sub>H</sub>	Input capture control status register 23	ICS23	R/W	Input capture 2/3	00000000
000054 <sub>H</sub>	Low order of output compare control register ch.0	OCS0	R/W	Output compare ch.0/1	0000--00
000055 <sub>H</sub>	High order of output compare control register ch.1	OCS1	R/W		---00000
000056 <sub>H</sub>	Low order of output compare control register ch.2	OCS2	R/W	Output compare ch.2/3	0000--00
000057 <sub>H</sub>	High order of output compare control register ch.3	OCS3	R/W		---00000
000058 <sub>H</sub>	Serial mode control status register	SMCS	R/W	I/O Extended serial	XXXX0000
000059 <sub>H</sub>			R/W		00000010
00005A <sub>H</sub>	Serial data register	SDR	R/W		XXXXXXXXXX
00005B <sub>H</sub>	Communication Prescaler Control Register	SDCR	R/W	Communication Prescaler	0XXX0000
00005C <sub>H</sub>	PWC Control Status Registers	PWCSR	R/W,R	16-bit PWC timer	00000000
00005D <sub>H</sub>					
00005E <sub>H</sub>	PWC data buffer register	PWCR	R/W		00000000
00005F <sub>H</sub>					
000060 <sub>H</sub>	PWC ratio of dividing frequency control register	DIVR	R/W		

**Table A-1 I/O Map (4 / 9)**

Address	Registers	Abbreviation	Access	Release	Initial value
000061 <sub>H</sub>	Use prohibited				
000062 <sub>H</sub>	Timer control status register 0	TMCSR0	R/W	16bit reload timer ch.0	00000000
000063 <sub>H</sub>					XXXX0000
000064 <sub>H</sub>	Low order of 16-bit timer register 0	TMR0	R		XXXXXXXXXX
	Low order of 16-bit reloading 0	TMRLR0	W		XXXXXXXXXX
000065 <sub>H</sub>	High order of 16-bit timer register 0	TMR0	R		XXXXXXXXXX
	High order of 16-bit reloading 0	TMRLR0	W		XXXXXXXXXX
000066 <sub>H</sub>	Timer control status register 1	TMCSR1	R/W	16bit reload timer ch.1	00000000
000067 <sub>H</sub>					XXXX0000
000068 <sub>H</sub>	Low order of 16-bit timer register 1	TMR1	R		XXXXXXXXXX
	Low order of 16-bit reloading 1	TMRLR1	W		XXXXXXXXXX
000069 <sub>H</sub>	High order of 16-bit timer register 1	TMR1	R		XXXXXXXXXX
	High order of 16-bit reloading 1	TMRLR1	W		XXXXXXXXXX
00006A <sub>H</sub>	Timer control status register 2	TMCSR2	R/W	16bit reload timer ch.2	00000000
00006B <sub>H</sub>					XXXX0000
00006C <sub>H</sub>	Low order of 16-bit timer register 2	TMR2	R		XXXXXXXXXX
	Low order of 16-bit reloading 2	TMRLR2	W		XXXXXXXXXX
00006D <sub>H</sub>	High order of 16-bit timer register 2	TMR2	R		XXXXXXXXXX
	High order of 16-bit reloading 2	TMRLR2	W		XXXXXXXXXX
00006E <sub>H</sub>	Use prohibited				
00006F <sub>H</sub>	ROM Mirroring Function Select Register	ROMM	R/W,W	ROM mirror function	-----11
000070 <sub>H</sub>	I <sup>2</sup> C bus status register 0	IBSR0	R	I <sup>2</sup> C interface ch.0	00000000
000071 <sub>H</sub>	I <sup>2</sup> C bus control register 0	IBCR0	R/W		00000000
000072 <sub>H</sub>	I <sup>2</sup> C bus clock control register 0	ICCR0	R/W		XX0XXXXX
000073 <sub>H</sub>	I <sup>2</sup> C bus address register 0	IADR0	R/W		XXXXXXXXXX
000074 <sub>H</sub>	I <sup>2</sup> C bus data register 0	IDAR0	R/W		XXXXXXXXXX
000075 <sub>H</sub>	Use prohibited				
000076 <sub>H</sub>	I <sup>2</sup> C bus status register 1	IBSR1	R	I <sup>2</sup> C interface ch.1	00000000
000077 <sub>H</sub>	I <sup>2</sup> C bus control register 1	IBCR1	R/W		00000000
000078 <sub>H</sub>	I <sup>2</sup> C bus clock control register 1	ICCR1	R/W		XX0XXXXX
000079 <sub>H</sub>	I <sup>2</sup> C bus address register 1	IADR1	R/W		XXXXXXXXXX
00007A <sub>H</sub>	I <sup>2</sup> C bus data register 1	IDAR1	R/W		XXXXXXXXXX
00007B <sub>H</sub>	Use prohibited				



**Table A-1 I/O Map (5 / 9)**

Address	Registers	Abbreviation	Access	Release	Initial value
00007C <sub>H</sub>	I <sup>2</sup> C bus status register 2	IBSR2	R	I <sup>2</sup> C interface ch.2	00000000
00007D <sub>H</sub>	I <sup>2</sup> C bus control register 2	IBCR2	R/W		00000000
00007E <sub>H</sub>	I <sup>2</sup> C clock control register 2	ICCR2	R/W		XX0XXXXX
00007F <sub>H</sub>	I <sup>2</sup> C bus address register 2	IADR2	R/W		XXXXXXXX
000080 <sub>H</sub>	I <sup>2</sup> C bus data register 2	IDAR2	R/W		XXXXXXXX
000081 <sub>H</sub> to 000085 <sub>H</sub>	Use prohibited				
000086 <sub>H</sub>	Low order of timer counter data register	TCDT	R/W	16-bit free-run timer	00000000
000087 <sub>H</sub>	High order of timer counter data register		R/W		00000000
000088 <sub>H</sub>	Timer control status register, lower	TCCS	R/W		00000000
000089 <sub>H</sub>	Timer control status register, higher		R/W		0XX00000
00008A <sub>H</sub>	Low order of compare clear register	CPCLR	R/W		XXXXXXXX
00008B <sub>H</sub>	High order of compare clear register		R/W		XXXXXXXX
00008C <sub>H</sub> to 00009A <sub>H</sub>	Use prohibited				
00009B <sub>H</sub>	DMA descriptor channel specification register	DCSR	R/W	μDMAC	00000000
00009C <sub>H</sub>	Low order of DMA status register	DSRL	R/W		00000000
00009D <sub>H</sub>	Low order of DMA status register	DSRH	R/W		00000000
00009E <sub>H</sub>	Program address detection control status register	PACSR	R/W	Address compare detection	00000000
00009F <sub>H</sub>	Delay interruption factor generation/release register	DIRR	R/W	Delayed interrupt	-----0
0000A0 <sub>H</sub>	Low-power consumption mode control register	LPMCR	R/W,W	Low power consumption	00011000
0000A1 <sub>H</sub>	Clock select register	CKSCR	R/W,R	Clock	11111100
0000A2 <sub>H</sub> 0000A3 <sub>H</sub>	Use prohibited				
0000A4 <sub>H</sub>	DMA stop status register	DSSR	R/W	μDMAC	00000000
0000A5 <sub>H</sub>	Automatic ready function selection register	ARSR	W	External pin	0011--00
0000A6 <sub>H</sub>	External address output control register	HACR	W		*****
0000A7 <sub>H</sub>	Bus control signal selection register	EPCR	W		1000*10-
0000A8 <sub>H</sub>	Watchdog timer control register	WDTC	R,W	Watchdog Timers	X-XXX111
0000A9 <sub>H</sub>	Time-base timer control register	TBTC	R/W,W	Time-base Timers	1--00100
0000AA <sub>H</sub>	Watch timer control register	WTC	R/W,R	Watch timer	10001000
0000AB <sub>H</sub>	Use prohibited				

**Table A-1 I/O Map (6 / 9)**

Address	Registers	Abbreviation	Access	Release	Initial value
0000AC <sub>H</sub>	Low order of DMA permission register	DERL	R/W	μDMAC	00000000
0000AD <sub>H</sub>	High order of DMA permission register	DERH	R/W		00000000
0000AE <sub>H</sub>	Flash memory control status register	FMCS	R/W,R,W	Flash memory I/F	000X0000
0000AF <sub>H</sub>	Use prohibited				
0000B0 <sub>H</sub>	Interrupt control registers 00	ICR00	R/W	Interrupt controller	00000111
0000B1 <sub>H</sub>	Interrupt control registers 01	ICR01	R/W		00000111
0000B2 <sub>H</sub>	Interrupt control registers 02	ICR02	R/W		00000111
0000B3 <sub>H</sub>	Interrupt control registers 03	ICR03	R/W		00000111
0000B4 <sub>H</sub>	Interrupt control registers 04	ICR04	R/W		00000111
0000B5 <sub>H</sub>	Interrupt control registers 05	ICR05	R/W	Interrupt controller	00000111
0000B6 <sub>H</sub>	Interrupt control registers 06	ICR06	R/W		00000111
0000B7 <sub>H</sub>	Interrupt control registers 07	ICR07	R/W		00000111
0000B8 <sub>H</sub>	Interrupt control registers 08	ICR08	R/W		00000111
0000B9 <sub>H</sub>	Interrupt control registers 09	ICR09	R/W		00000111
0000BA <sub>H</sub>	Interrupt control registers 10	ICR10	R/W		00000111
0000BB <sub>H</sub>	Interrupt control registers 11	ICR11	R/W		00000111
0000BC <sub>H</sub>	Interrupt control registers 12	ICR12	R/W		00000111
0000BD <sub>H</sub>	Interrupt control registers 13	ICR13	R/W		00000111
0000BE <sub>H</sub>	Interrupt control registers 14	ICR14	R/W		00000111
0000BF <sub>H</sub>	Interrupt control registers 15	ICR15	R/W		00000111
0000C0 <sub>H</sub>	Host control register 0	HCNT0	R/W	USB HOST	00000000
0000C1 <sub>H</sub>	Host control register 1	HCNT1	R/W		00000001
0000C2 <sub>H</sub>	Host interruption register	HIRQ	R/W		00000000
0000C3 <sub>H</sub>	Host error status register	HERR	R/W		00000011
0000C4 <sub>H</sub>	Host state status register	HSTATE	R/W,R		XX010010
0000C5 <sub>H</sub>	SOF interruption FRAME comparison register	HFCOMP	R/W		00000000
0000C6 <sub>H</sub>	Retry timer setting register	HRTIMER	R/W		00000000
0000C7 <sub>H</sub>					00000000
0000C8 <sub>H</sub>					XXXXXX00
0000C9 <sub>H</sub>	Host address register	HADR	R/W		X0000000
0000CA <sub>H</sub>	EOF setting register	HEOF	R/W		00000000
0000CB <sub>H</sub>					XX000000
0000CC <sub>H</sub>	FRAME setting register	HFRAME	R/W		00000000
0000CD <sub>H</sub>					XXXXXX00
0000CE <sub>H</sub>	Host token end point register	HTOKEN	R/W		00000000
0000CF <sub>H</sub>	Use prohibited				

**Table A-1 I/O Map (7 / 9)**

Address	Registers	Abbreviation	Access	Release	Initial value
0000D0 <sub>H</sub>	UDC control register	UDCC	R/W	USB function	10100000
0000D1 <sub>H</sub>					00000000
0000D2 <sub>H</sub>	EP0 control register	EP0C	R/W		01000000
0000D3 <sub>H</sub>			R/W		XXXXX0000
0000D4 <sub>H</sub>	EP1 control register	EP1C	R/W		00000000
0000D5 <sub>H</sub>			R/W		01100001
0000D6 <sub>H</sub>	EP2 control register	EP2C	R/W		01000000
0000D7 <sub>H</sub>			R/W		01100000
0000D8 <sub>H</sub>	EP3 control register	EP3C	R/W		01000000
0000D9 <sub>H</sub>			R/W		01100000
0000DA <sub>H</sub>	EP4 control register	EP4C	R/W		01000000
0000DB <sub>H</sub>			R/W		01100000
0000DC <sub>H</sub>	EP5 control register	EP5C	R/W		01000000
0000DD <sub>H</sub>			R/W		01100000
0000DE <sub>H</sub>	Time stamp register	TMSP	R		00000000
0000DF <sub>H</sub>			R		XXXXXX000
0000E0 <sub>H</sub>	UDC status register	UDCS	R/W		XX000000
0000E1 <sub>H</sub>	UDC interruption permission register	UDCIE	R/W,R		00000000
0000E2 <sub>H</sub>	EP0I status register	EP0IS	R/W		XXXXXXXXXX
0000E3 <sub>H</sub>			R/W		10XXX1XX
0000E4 <sub>H</sub>	EP0O status register	EP0OS	R/W,R		0XXXXXXXXX
0000E5 <sub>H</sub>			R/W		100XX000
0000E6 <sub>H</sub>	EP1 status register	EP1S	R		XXXXXXXXXX
0000E7 <sub>H</sub>			R/W,R		1000000X
0000E8 <sub>H</sub>	EP2 status register	EP2S	R		0XXXXXXXXX
0000E9 <sub>H</sub>			R/W,R		10000000
0000EA <sub>H</sub>	EP3 status register	EP3S	R		0XXXXXXXXX
0000EB <sub>H</sub>			R/W,R		10000000
0000EC <sub>H</sub>	EP4 status register	EP4S	R		0XXXXXXXXX
0000ED <sub>H</sub>			R/W,R		10000000
0000EE <sub>H</sub>	EP5 status register	EP5S	R		0XXXXXXXXX
0000EF <sub>H</sub>			R/W,R		10000000
0000F0 <sub>H</sub>	EP0 data register	EP0DT	R/W		XXXXXXXXXX
0000F1 <sub>H</sub>			R/W		XXXXXXXXXX
0000F2 <sub>H</sub>	EP1 data register	EP1DT	R/W		XXXXXXXXXX
0000F3 <sub>H</sub>			R/W		XXXXXXXXXX

**Table A-1 I/O Map (8 / 9)**

Address	Registers	Abbreviation	Access	Release	Initial value
0000F4 <sub>H</sub>	EP2 data register	EP2DT	R/W	USB function	XXXXXXXX
0000F5 <sub>H</sub>			R/W		XXXXXXXX
0000F6 <sub>H</sub>	EP3 data register	EP3DT	R/W		XXXXXXXX
0000F7 <sub>H</sub>			R/W		XXXXXXXX
0000F8 <sub>H</sub>	EP4 data register	EP4DT	R/W		XXXXXXXX
0000F9 <sub>H</sub>			R/W		XXXXXXXX
0000FA <sub>H</sub>	EP5 data register	EP5DT	R/W		XXXXXXXX
0000FB <sub>H</sub>			R/W		XXXXXXXX
0000FC <sub>H</sub> to 0000FF <sub>H</sub>	Use prohibited				
000100 <sub>H</sub> to # <sub>H</sub>	RAM Area				
001FF0 <sub>H</sub>	Low order of program address detection register ch.0	PADR0	R/W	Address Match detection	XXXXXXXX
001FF1 <sub>H</sub>	Middle order of program address detection register ch.0		R/W		XXXXXXXX
001FF2 <sub>H</sub>	High order of program address detection register ch.0		R/W		XXXXXXXX
001FF3 <sub>H</sub>	Low order of program address detection register ch.1	PADR1	R/W		XXXXXXXX
001FF4 <sub>H</sub>	Middle order of program address detection register ch.1		R/W		XXXXXXXX
001FF5 <sub>H</sub>	High order of program address detection register ch.1		R/W		XXXXXXXX
#H to 0078FF <sub>H</sub>	Unused area				
007900 <sub>H</sub>	Lower ch.0 of PPG reload register	PRL0	R/W	PPG ch.0	XXXXXXXX
007901 <sub>H</sub>	Higher ch.0 of PPG reload register	PRLH0	R/W		XXXXXXXX
007902 <sub>H</sub>	Lower ch.1 of PPG reload register	PRL1	R/W	PPG ch.1	XXXXXXXX
007903 <sub>H</sub>	Higher ch.1 of PPG reload register	PRLH1	R/W		XXXXXXXX
007904 <sub>H</sub>	Lower ch.2 of PPG reload register	PRL2	R/W	PPG ch.2	XXXXXXXX
007905 <sub>H</sub>	Higher ch.2 of PPG reload register	PRLH2	R/W		XXXXXXXX
007906 <sub>H</sub>	Lower ch.3 of PPG reload register	PRL3	R/W	PPG ch.3	XXXXXXXX
007907 <sub>H</sub>	Higher ch.3 of PPG reload register	PRLH3	R/W		XXXXXXXX
007908 <sub>H</sub>	Lower ch.4 of PPG reload register	PRL4	R/W	PPG ch.4	XXXXXXXX
007909 <sub>H</sub>	Higher ch.4 of PPG reload register	PRLH4	R/W		XXXXXXXX
00790A <sub>H</sub>	Lower ch.5 of PPG reload register	PRL5	R/W	PPG ch.5	XXXXXXXX
00790B <sub>H</sub>	Higher ch.5 of PPG reload register	PRLH5	R/W		XXXXXXXX

**Table A-1 I/O Map (9 / 9)**

Address	Registers	Abbreviation	Access	Release	Initial value
00790C <sub>H</sub> to 00790F <sub>H</sub>	Use prohibited				
007910 <sub>H</sub>	Input Capture Data Register lower ch.0	IPCP0	R	Input capture ch.0/1	XXXXXXXX
007911 <sub>H</sub>	Input Capture Data Register upper ch.0		R		XXXXXXXX
007912 <sub>H</sub>	Input Capture Data Register lower ch.1	IPCP1	R		XXXXXXXX
007913 <sub>H</sub>	Input Capture Data Register upper ch.1		R		XXXXXXXX
007914 <sub>H</sub>	Input Capture Data Register lower ch.2	IPCP2	R	Input capture ch.2/3	XXXXXXXX
007915 <sub>H</sub>	Input Capture Data Register upper ch.2		R		XXXXXXXX
007916 <sub>H</sub>	Input Capture Data Register lower ch.3	IPCP3	R		XXXXXXXX
007917 <sub>H</sub>	Input Capture Data Register upper ch.3		R		XXXXXXXX
007918 <sub>H</sub>	Lower ch.0 of output compare register	OCCP0	R/W	Output compare ch.0/1	XXXXXXXX
007919 <sub>H</sub>	Higher ch.0 of output compare register		R/W		XXXXXXXX
00791A <sub>H</sub>	Lower ch.1 of output compare register	OCCP1	R/W		XXXXXXXX
00791B <sub>H</sub>	Higher ch.1 of output compare register		R/W		XXXXXXXX
00791C <sub>H</sub>	Lower ch.2 of output compare register	OCCP2	R/W	Output compare ch.2/3	XXXXXXXX
00791D <sub>H</sub>	Higher ch.2 of output compare register		R/W		XXXXXXXX
00791E <sub>H</sub>	Lower ch.3 of output compare register	OCCP3	R/W		XXXXXXXX
00791F <sub>H</sub>	Higher ch.3 of output compare register		R/W		XXXXXXXX
007920 <sub>H</sub>	DMA Buffer address pointer lower 8 bit	DBAPL	R/W	μDMAC	XXXXXXXX
007921 <sub>H</sub>	DMA Buffer address pointer middle 8 bit	DBAPM	R/W		XXXXXXXX
007922 <sub>H</sub>	DMA Buffer address pointer upper 8 bit	DBAPH	R/W		XXXXXXXX
007923 <sub>H</sub>	DMA control register	DMACS	R/W		XXXXXXXX
007924 <sub>H</sub>	DMA I/O Register Address Pointer lower 8 bit	DIOAL	R/W		XXXXXXXX
007925 <sub>H</sub>	DMA I/O Register Address Pointer upper 8 bit	DIOAH	R/W		XXXXXXXX
007926 <sub>H</sub>	Lower 8 bits of DMA data counter	DDCTL	R/W		XXXXXXXX
007927 <sub>H</sub>	Higher 8 bits of DMA data counter	DDCTH	R/W		XXXXXXXX
007928 <sub>H</sub> to 007FFF <sub>H</sub>	Use prohibited				

- Explanation of reading/writing  
R/W: Readable and Writable  
R: Read only  
W: Write only
- Explanation of initial value  
0: The initial value is "0".

- 1: The initial value is "1".
- X: The initial value is irregular.
- : It is Undefined bit. The initial value is indefinite.
- \*: The initial value is "1" or "0".

---

Note:

You cannot use any I/O-related command to registers that are placed from 7900<sub>H</sub> to 7FFF<sub>H</sub>.

---

■ **Interrupt Factors, Interrupt Vectors, and Interrupt Control Registers**

Table A-2 lists the correspondence between interrupt factors, and interrupt vectors and interrupt control registers.

**Table A-2 Correspondence between Interrupt Factors, and Interrupt Vectors and Interrupt Control Registers**

Interrupt cause	EI <sup>2</sup> OS Clear	μDMAC Channel	Interrupt vector		Interrupt control register	
			Number	Address	ICR	Address
Reset	×	×	#08	FFFFDC <sub>H</sub>	-	-
INT9 instruction	×	×	#09	FFFFD8 <sub>H</sub>	-	-
Exception	×	×	#10	FFFFD4 <sub>H</sub>	-	-
USB function 1	×	0, 1	#11	FFFFD0 <sub>H</sub>	ICR00	0000B0 <sub>H</sub>
USB function 2	×	2 to 6 *	#12	FFFFCC <sub>H</sub>		
USB function 3	×	×	#13	FFFFC8 <sub>H</sub>	ICR01	0000B1 <sub>H</sub>
USB function 4	×	×	#14	FFFFC4 <sub>H</sub>		
USB HOST 1	×	×	#15	FFFFC0 <sub>H</sub>	ICR02	0000B2 <sub>H</sub>
USB HOST 2	×	×	#16	FFFFBC <sub>H</sub>		
I <sup>2</sup> C ch.0	×	×	#17	FFFFB8 <sub>H</sub>	ICR03	0000B3 <sub>H</sub>
DTP/External interrupt ch.0/ch.1	○	×	#18	FFFFB4 <sub>H</sub>		
I <sup>2</sup> C ch.1	×	×	#19	FFFFB0 <sub>H</sub>	ICR04	0000B4 <sub>H</sub>
DTP/External interrupt ch.2/ch.3	○	×	#20	FFFFAC <sub>H</sub>		
I <sup>2</sup> C ch.2	×	×	#21	FFFFA8 <sub>H</sub>	ICR05	0000B5 <sub>H</sub>
DTP/External interrupt ch.4/ch.5	○	×	#22	FFFFA4 <sub>H</sub>		
PWC/reload timer ch.0	Δ	14	#23	FFFFA0 <sub>H</sub>	ICR06	0000B6 <sub>H</sub>
DTP/External interrupt ch.6/ch.7	Δ	×	#24	FFFF9C <sub>H</sub>		
Input capture ch.0/ch.1	Δ	7	#25	FFFF98 <sub>H</sub>	ICR07	0000B7 <sub>H</sub>
Reload timer ch.1	Δ	×	#26	FFFF94 <sub>H</sub>		
Input capture ch.2/ch.3	Δ	8	#27	FFFF90 <sub>H</sub>	ICR08	0000B8 <sub>H</sub>
Reload timer ch.2	Δ	×	#28	FFFF8C <sub>H</sub>		
Output compare ch.0/ch.1	○	×	#29	FFFF88 <sub>H</sub>	ICR09	0000B9 <sub>H</sub>
PPG ch.0/ch.1	×	×	#30	FFFF84 <sub>H</sub>		
Output compare ch.2/ch.3	○	×	#31	FFFF80 <sub>H</sub>	ICR10	0000BA <sub>H</sub>
PPG ch.2/ch.3	×	×	#32	FFFF7C <sub>H</sub>		
UART transmit end ch.2/ch.3	○	11	#33	FFFF78 <sub>H</sub>	ICR11	0000BB <sub>H</sub>
PPG ch.4/ch.5	×	×	#34	FFFF74 <sub>H</sub>		
UART receive end ch.2/ch.3	⊙	10	#35	FFFF70 <sub>H</sub>	ICR12	0000BC <sub>H</sub>
Analog to digital conversion / free-run timer	Δ	15	#36	FFFF6C <sub>H</sub>		
UART transmit end ch.0/ch.1	○	13	#37	FFFF68 <sub>H</sub>	ICR13	0000BD <sub>H</sub>
Extended serial I/O	×	9	#38	FFFF64 <sub>H</sub>		
UART receive end ch.0/ch.1	⊙	12	#39	FFFF60 <sub>H</sub>	ICR14	0000BE <sub>H</sub>
Time-base timer /Watch timer	×	×	#40	FFFF5C <sub>H</sub>		
Flash writing/deletion	×	×	#41	FFFF58 <sub>H</sub>	ICR15	0000BF <sub>H</sub>
Delay interrupt generation module	×	×	#42	FFFF54 <sub>H</sub>		

⊙: Available. With the EI<sup>2</sup>OS stop function (the interrupt request flag is cleared with the interrupt clear signal. With a stop request.)

○: Available (the interrupt request flag is cleared with the interrupt clear signal.)

Δ: Usable at non-using interrupt source of sharing

×: Use disabled

\*: Channels 2 and 3 can use even when USB HOST is operated.

## Notes:

- If the same interrupt control register (ICR) has two interrupt factors and the use of the EI<sup>2</sup>OS is permitted, the EI<sup>2</sup>OS is activated when either of the factors is detected. As any interrupt other than the activation factor is masked while the EI<sup>2</sup>OS is running, it is recommended that you should mask either of the interrupt requests when using the EI<sup>2</sup>OS.
- The interrupt flag is cleared by the EI<sup>2</sup>OS interrupt clear signal for the resource that has two interrupt factors in the same interrupt control register (ICR).
- When the same interrupt number has two interrupt factors, both interrupt request flags for a resource are cleared with the μDMAC interrupt clear signal. Therefore, when you use either of two interrupt factors for the DMAC function, another interrupt function is disabled. Set the interrupt request permission bit to "0" in the appropriate resource, and take measures by software polling.

## ■ Type and Function of USB Interrupt

USB interrupt factor	Details
USB function1	End Point0-IN End Point0-OUT
USB function 2	End Point1 to 5 *
USB function 3	SUSP SOF BRST WKUP CONF
USB function 4	SPK
USB HOST 1	DIRQ CNNIRQ URIRQ RWKIRQ
USB HOST 2	SOFIRQ CMPIRQ

\*: End Point 1 and 2 can use even when USB HOST is operated.



## **APPENDIX B Instructions**

---

**APPENDIX B describes the instructions used by the F<sup>2</sup>MC-16LX.**

---

- B.1 Instruction Types
- B.2 Addressing
- B.3 Direct Addressing
- B.4 Indirect Addressing
- B.5 Execution Cycle Count
- B.6 Effective address field
- B.7 How to Read the Instruction List
- B.8 F<sup>2</sup>MC-16LX Instruction List
- B.9 Instruction Map

## B.1 Instruction Types

---

The F<sup>2</sup>MC-16LX supports 351 types of instructions. Addressing is enabled by using an effective address field of each instruction or using the instruction code itself.

---

### ■ Instruction Types

The F<sup>2</sup>MC-16LX supports the following 351 types of instructions:

- 41 transfer instructions (byte)
- 38 transfer instructions (word or long word)
- 42 addition/subtraction instructions (byte, word, or long word)
- 12 increment/decrement instructions (byte, word, or long word)
- 11 comparison instructions (byte, word, or long word)
- 11 unsigned multiplication/division instructions (word or long word)
- 11 signed multiplication/division instructions (word or long word)
- 39 logic instructions (byte or word)
- 6 logic instructions (long word)
- 6 sign inversion instructions (byte or word)
- 1 normalization instruction (long word)
- 18 shift instructions (byte, word, or long word)
- 50 branch instructions
- 6 accumulator operation instructions (byte or word)
- 28 other control instructions (byte, word, or long word)
- 21 bit operation instructions
- 10 string instructions

## B.2 Addressing

---

With the F<sup>2</sup>MC-16LX, the address format is determined by the instruction effective address field or the instruction code itself (implied). When the address format is determined by the instruction code itself, specify an address in accordance with the instruction code used. Some instructions permit the user to select several types of addressing.

---

### ■ Addressing

The F<sup>2</sup>MC-16LX supports the following 23 types of addressing:

- Immediate (#imm)
- Register direct
- Direct branch address (addr16)
- Physical direct branch address (addr24)
- I/O direct (io)
- Abbreviated direct address (dir)
- Direct address (addr16)
- I/O direct bit address (io:bp)
- Abbreviated direct bit address (dir:bp)
- Direct bit address (addr16:bp)
- Vector address (#vct)
- Register indirect (@RWj j = 0 to 3)
- Register indirect with post increment (@RWj+ j = 0 to 3)
- Register indirect with displacement (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)
- Long register indirect with displacement (@RLi + disp8 i = 0 to 3)
- Program counter indirect with displacement (@PC + disp16)
- Register indirect with base index (@RW0 + RW7, @RW1 + RW7)
- Program counter relative branch address (rel)
- Register list (rlst)
- Accumulator indirect (@A)
- Accumulator indirect branch address (@A)
- Indirectly-specified branch address (@ear)
- Indirectly-specified branch address (@eam)

## ■ Effective Address Field

Table B.2-1 lists the address formats specified by the effective address field.

**Table B.2-1 Effective Address Field**

Code	Representation			Address format	Default bank
00	R0	RW0	RL0	Register direct: Individual parts correspond to the byte, word, and long word types in order from the left.	None
01	R1	RW1	(RL0)		
02	R2	RW2	RL1		
03	R3	RW3	(RL1)		
04	R4	RW4	RL2		
05	R5	RW5	(RL2)		
06	R6	RW6	RL3		
07	R7	RW7	(RL3)		
08	@RW0			Register indirect	DTB
09	@RW1				DTB
0A	@RW2				ADB
0B	@RW3				SPB
0C	@RW0+			Register indirect with post increment	DTB
0D	@RW1+				DTB
0E	@RW2+				ADB
0F	@RW3+				SPB
10	@RW0+disp8			Register indirect with 8-bit displacement	DTB
11	@RW1+disp8				DTB
12	@RW2+disp8				ADB
13	@RW3+disp8				SPB
14	@RW4+disp8			Register indirect with 8-bit displacement	DTB
15	@RW5+disp8				DTB
16	@RW6+disp8				ADB
17	@RW7+disp8				SPB
18	@RW0+disp16			Register indirect with 16-bit displacement	DTB
19	@RW1+disp16				DTB
1A	@RW2+disp16				ADB
1B	@RW3+disp16				SPB
1C	@RW0+RW7			Register indirect with index	DTB
1D	@RW1+RW7			Register indirect with index	DTB
1E	@PC+disp16			PC indirect with 16-bit displacement	PCB
1F	addr16			Direct address	DTB

## B.3 Direct Addressing

An operand value, register, or address is specified explicitly in direct addressing mode.

### ■ Direct Addressing

#### ● Immediate addressing (#imm)

Specify an operand value explicitly (#imm4/ #imm8/ #imm16/ #imm32).

**Figure B.3-1 Example of Immediate Addressing (#imm)**

MOVW A, #01212H (This instruction stores the operand value in A.)		
Before execution	A	2 2 3 3 : 4 4 5 5
After execution	A	4 4 5 5 : 1 2 1 2 (Some instructions transfer AL to AH.)

#### ● Register direct addressing

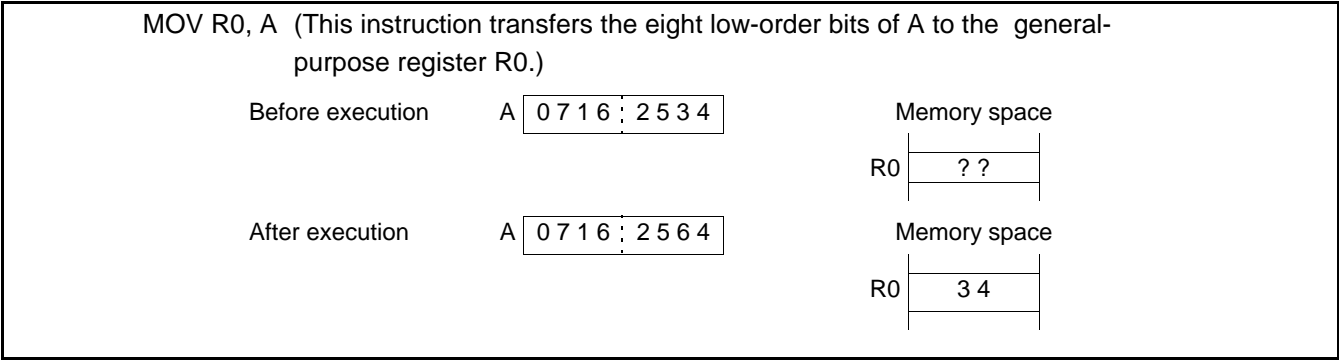
Specify a register explicitly as an operand. Table B.3-1 lists the registers that can be specified. Figure B.3-2 shows an example of register direct addressing.

**Table B.3-1 Direct Addressing Registers**

General-purpose register	Byte	R0, R1, R2, R3, R4, R5, R6, R7
	Word	RW0, RW1, RW2, RW3, RW4, RW5, RW6, RW7
	Long word	RL0, RL1, RL2, RL3
Special-purpose register	Accumulator	A, AL
	Pointer	SP *
	Bank	PCB, DTB, USB, SSB, ADB
	Page	DPR
	Control	PS, CCR, RP, ILM

\*: One of the user stack pointer (USP) and system stack pointer (SSP) is selected and used depending on the value of the S flag bit in the condition code register (CCR). For branch instructions, the program counter (PC) is not specified in an instruction operand but is specified implicitly.

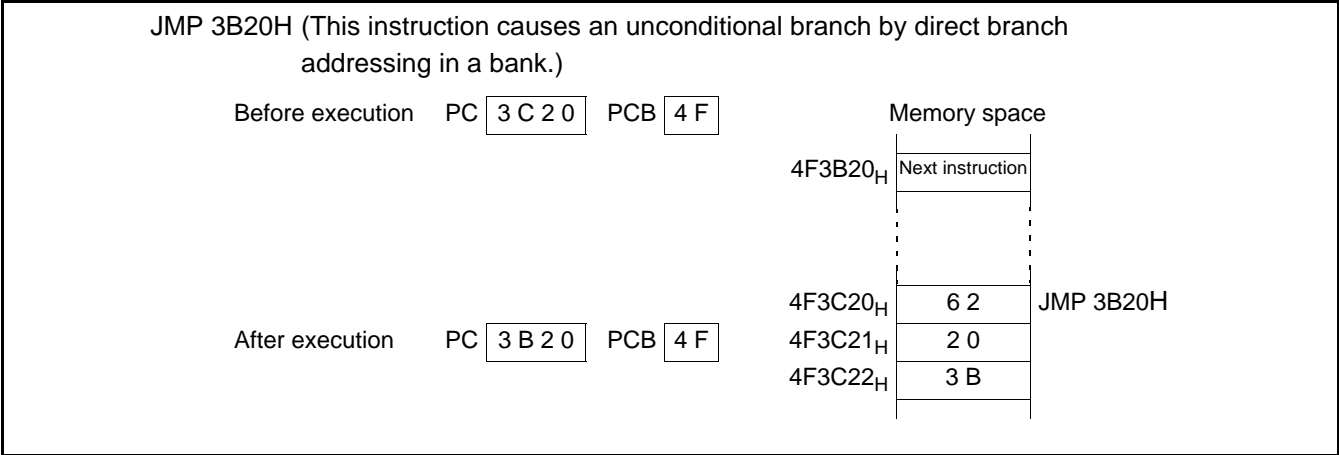
Figure B.3-2 Example of Register Direct Addressing



● Direct branch addressing (addr16)

Specify an offset explicitly for the branch destination address. The size of the offset is 16 bits, which indicates the branch destination in the logical address space. Direct branch addressing is used for an unconditional branch, subroutine call, or software interrupt instruction. Bit23 to bit16 of the address are specified by the program counter bank register (PCB).

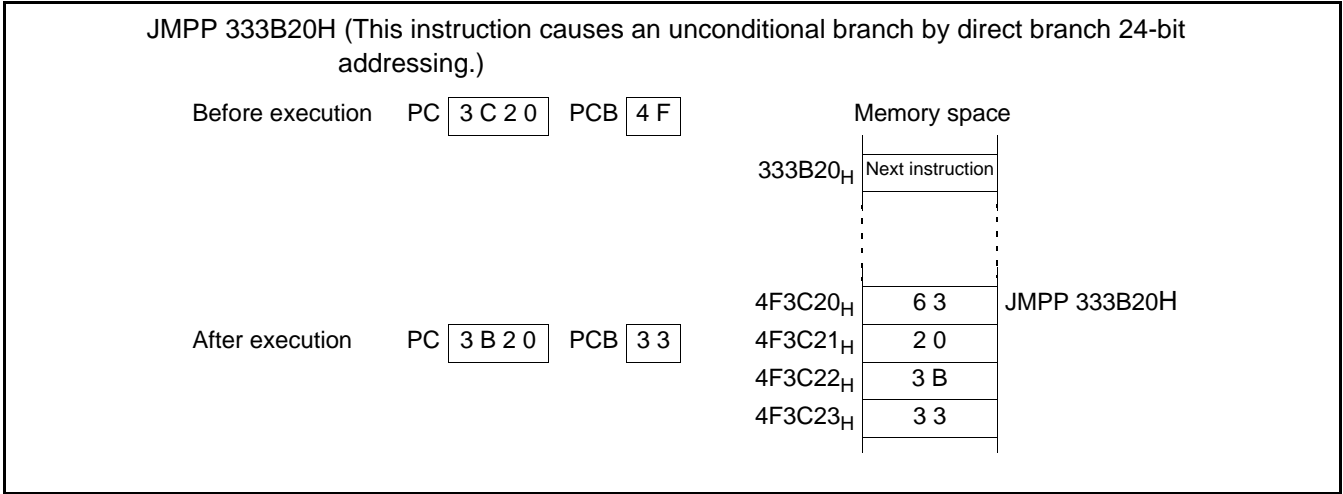
Figure B.3-3 Example of Direct Branch Addressing (addr16)



● Physical direct branch addressing (addr24)

Specify an offset explicitly for the branch destination address. The size of the offset is 24 bits. Physical direct branch addressing is used for unconditional branch, subroutine call, or software interrupt instruction.

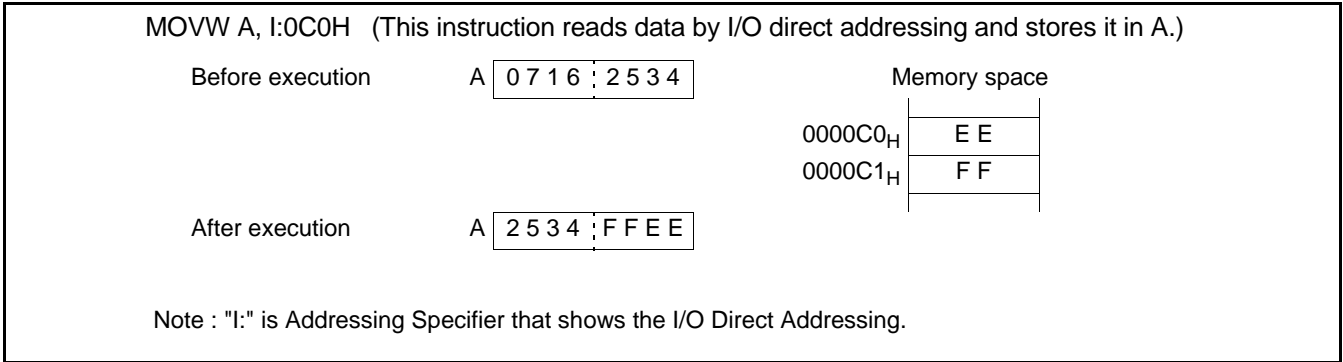
Figure B.3-4 Example of Direct Branch Addressing (addr24)



● I/O direct addressing (io)

Specify an 8-bit offset explicitly for the memory address in an operand. The I/O address space in the physical address space from 000000<sub>H</sub> to 0000FF<sub>H</sub> is accessed regardless of the data bank register (DTB) and direct page register (DPR). A bank select prefix for bank addressing is invalid if specified before an instruction using I/O direct addressing.

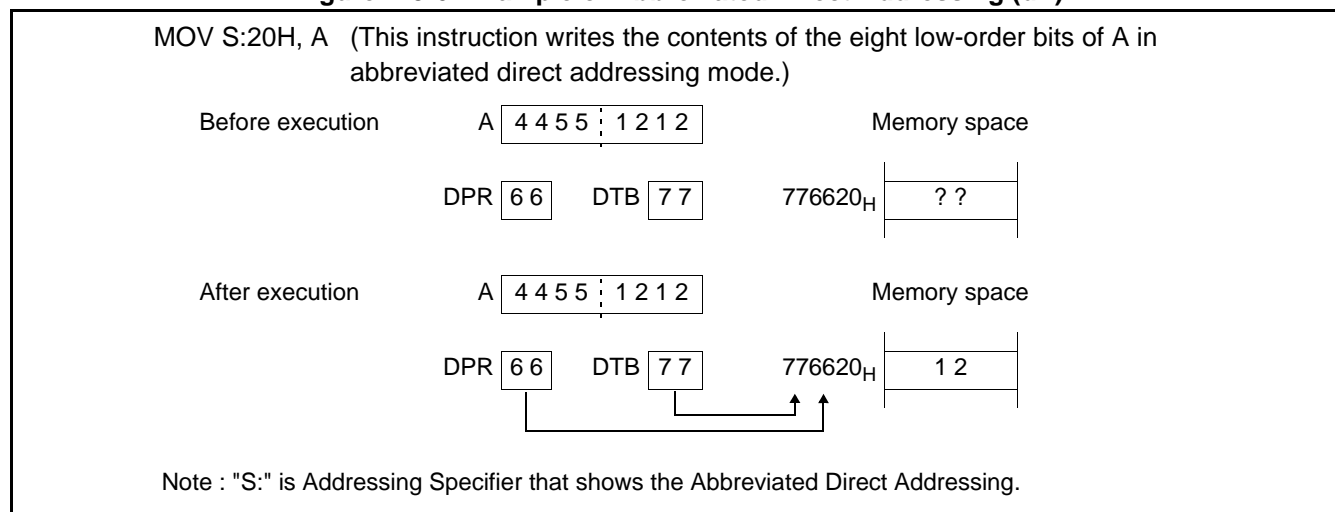
Figure B.3-5 Example of I/O Direct Addressing (io)



## ● Abbreviated direct addressing (dir)

Specify the eight low-order bits of a memory address explicitly in an operand. Address bits 8 to 15 are specified by the direct page register (DPR). Address bits 16 to 23 are specified by the data bank register (DTB).

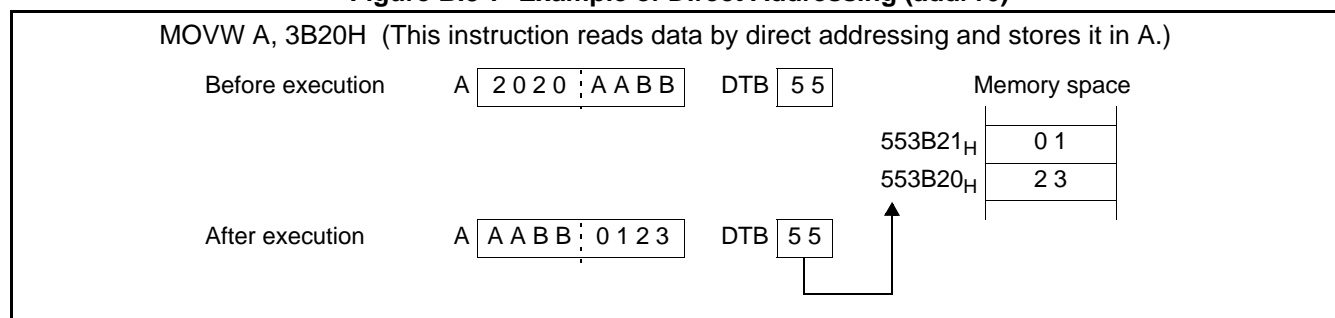
**Figure B.3-6 Example of Abbreviated Direct Addressing (dir)**



## ● Direct addressing (addr16)

Specify the 16 low-order bits of a memory address explicitly in an operand. Address bits 16 to 23 are specified by the data bank register (DTB). A prefix instruction for access space addressing is invalid for this mode of addressing.

**Figure B.3-7 Example of Direct Addressing (addr16)**



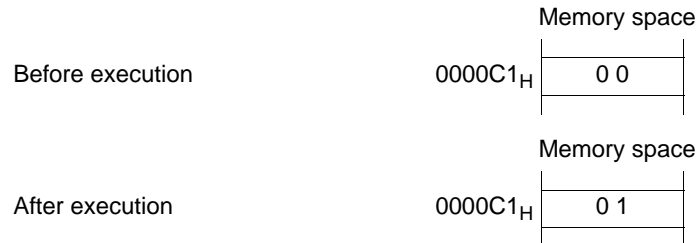


● I/O direct bit addressing (io:bp)

Specify bits in physical addresses 000000<sub>H</sub> to 0000FF<sub>H</sub> explicitly. Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

**Figure B.3-8 Example of I/O Direct Bit Addressing (io:bp)**

SETB I:0C1H:0 (This instruction sets bits by I/O direct bit addressing.)



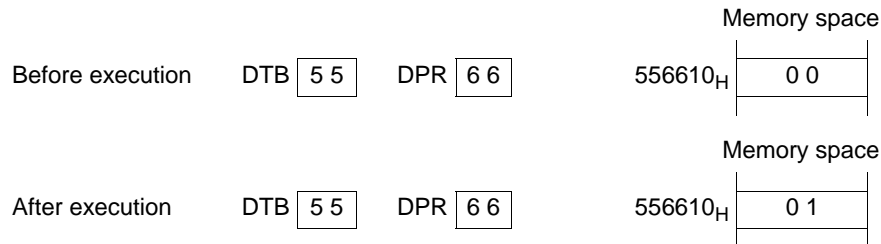
Note : "I:" is Addressing Specifier that shows the I/O Direct Addressing.

● Abbreviated direct bit addressing (dir:bp)

Specify the eight low-order bits of a memory address explicitly in an operand. Address bits 8 to 15 are specified by the direct page register (DPR). Address bits 16 to 23 are specified by the data bank register (DTB). Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

**Figure B.3-9 Example of Abbreviated Direct Bit Addressing (dir:bp)**

SETB S:10H:0 (This instruction sets bits by abbreviated direct bit addressing.)



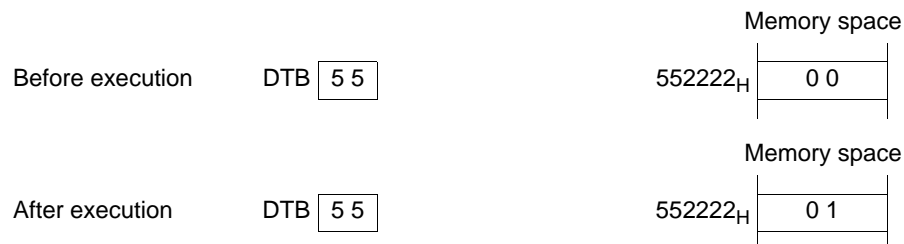
Note : "S:" is Addressing Specifier that shows the Abbreviated Direct Addressing.

● Direct bit addressing (addr16:bp)

Specify arbitrary bits in 64 kilobytes explicitly. Address bits 16 to 23 are specified by the data bank register (DTB). Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

**Figure B.3-10 Example of Direct Bit Addressing (addr16:bp)**

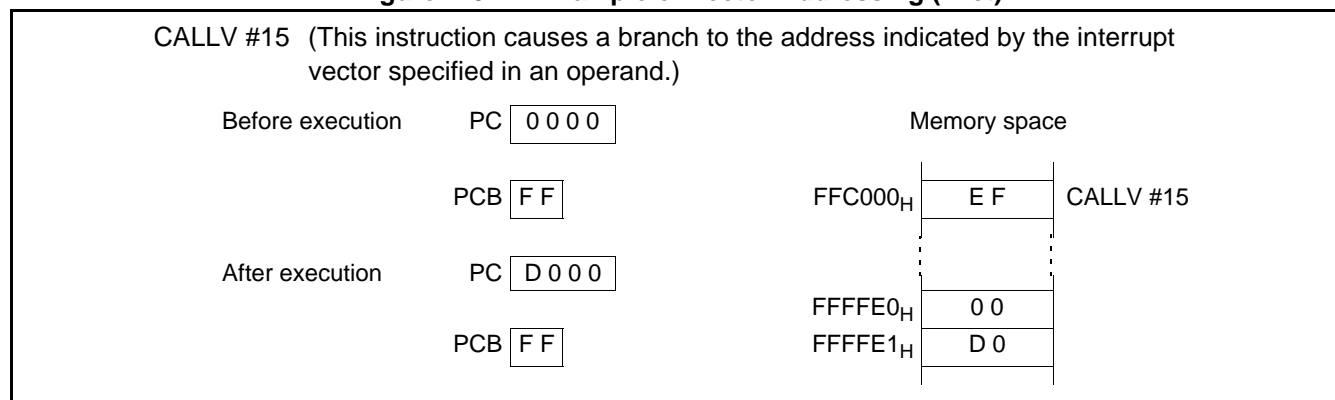
SETB 2222H : 0 (This instruction sets bits by direct bit addressing.)



## ● Vector Addressing (#vct)

Specify vector data in an operand to indicate the branch destination address. There are two sizes for vector numbers: 4 bits and 8 bits. Vector addressing is used for a subroutine call or software interrupt instruction.

**Figure B.3-11 Example of Vector Addressing (#vct)**



**Table B.3-2 CALLV Vector List**

Instruction	Vector address L	Vector address H
CALLV #0	XXFFFE <sub>H</sub>	XXFFFF <sub>H</sub>
CALLV #1	XXFFFC <sub>H</sub>	XXFFFD <sub>H</sub>
CALLV #2	XXFFFA <sub>H</sub>	XXFFFB <sub>H</sub>
CALLV #3	XXFFF8 <sub>H</sub>	XXFFF9 <sub>H</sub>
CALLV #4	XXFFF6 <sub>H</sub>	XXFFF7 <sub>H</sub>
CALLV #5	XXFFF4 <sub>H</sub>	XXFFF5 <sub>H</sub>
CALLV #6	XXFFF2 <sub>H</sub>	XXFFF3 <sub>H</sub>
CALLV #7	XXFFF0 <sub>H</sub>	XXFFF1 <sub>H</sub>
CALLV #8	XXFFEE <sub>H</sub>	XXFFEF <sub>H</sub>
CALLV #9	XXFFEC <sub>H</sub>	XXFFED <sub>H</sub>
CALLV #10	XXFFEA <sub>H</sub>	XXFFEB <sub>H</sub>
CALLV #11	XXFFE8 <sub>H</sub>	XXFFE9 <sub>H</sub>
CALLV #12	XXFFE6 <sub>H</sub>	XXFFE7 <sub>H</sub>
CALLV #13	XXFFE4 <sub>H</sub>	XXFFE5 <sub>H</sub>
CALLV #14	XXFFE2 <sub>H</sub>	XXFFE3 <sub>H</sub>
CALLV #15	XXFFE0 <sub>H</sub>	XXFFE1 <sub>H</sub>

Note: A PCB register value is set in XX.

### Note:

When the program counter bank register (PCB) is FF<sub>H</sub>, the vector area overlaps the vector area of INT #vct8 (#0 to #7). Use vector addressing carefully (see Table B.3-2).

## B.4 Indirect Addressing

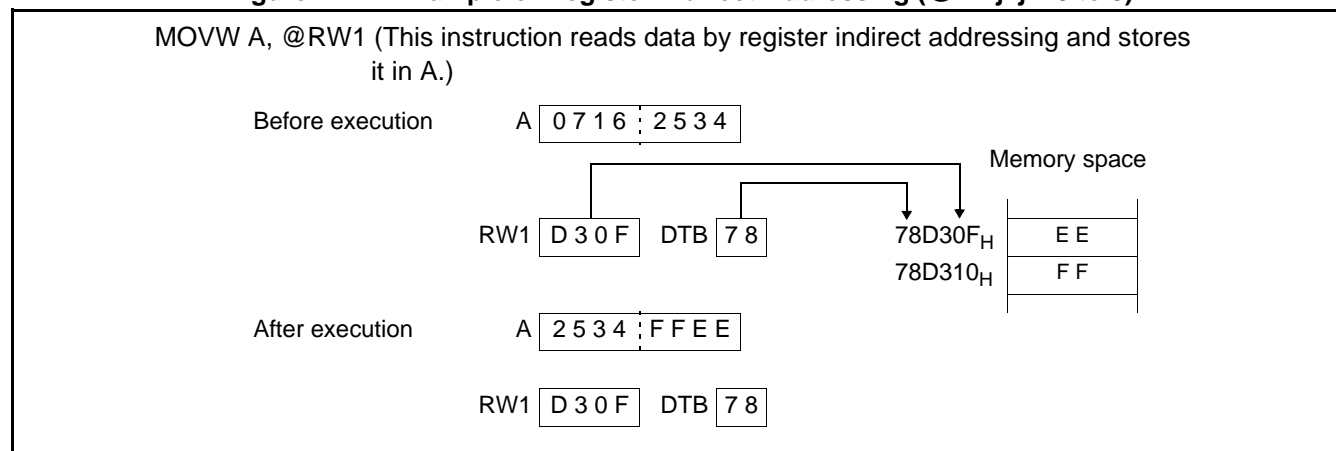
In indirect addressing mode, an address is specified indirectly by the address data of an operand.

### ■ Indirect Addressing

#### ● Register indirect addressing (@RWj j = 0 to 3)

Memory is accessed using the contents of general-purpose register RWj as an address. Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0 or RW1 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 is used, or additional data bank register (ADB) when RW2 is used.

**Figure B.4-1 Example of Register Indirect Addressing (@RWj j = 0 to 3)**

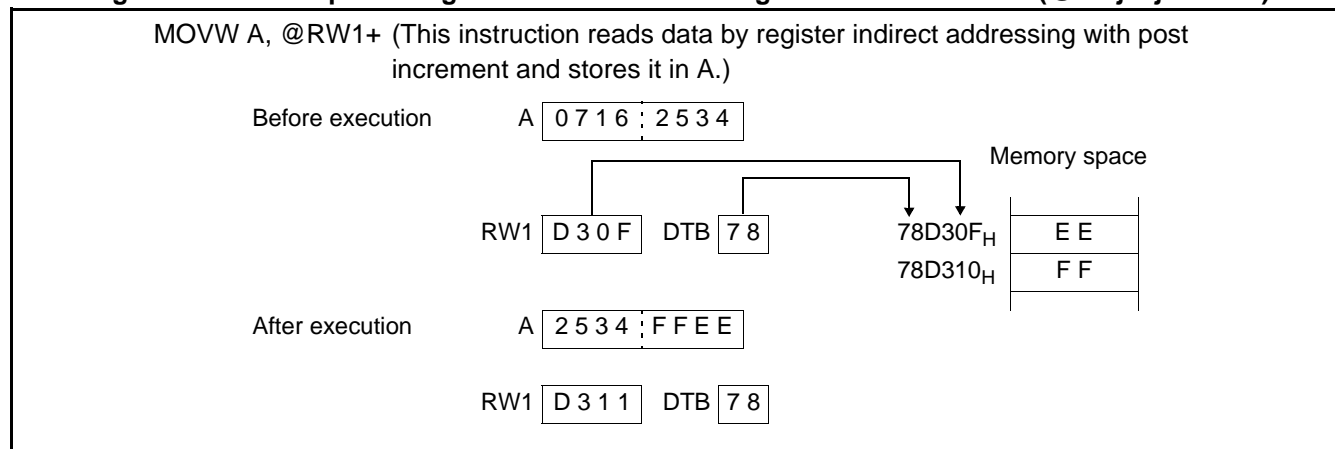


#### ● Register indirect addressing with post increment (@RWj+ j = 0 to 3)

Memory is accessed using the contents of general-purpose register RWj as an address. After operand operation, RWj is incremented by the operand size (1 for a byte, 2 for a word, or 4 for a long word). Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0 or RW1 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 is used, or additional data bank register (ADB) when RW2 is used.

If the post increment results in the address of the register that specifies the increment, the incremented value is referenced after that. In this case, if the next instruction is a write instruction, priority is given to writing by an instruction and, therefore, the register that would be incremented becomes write data.

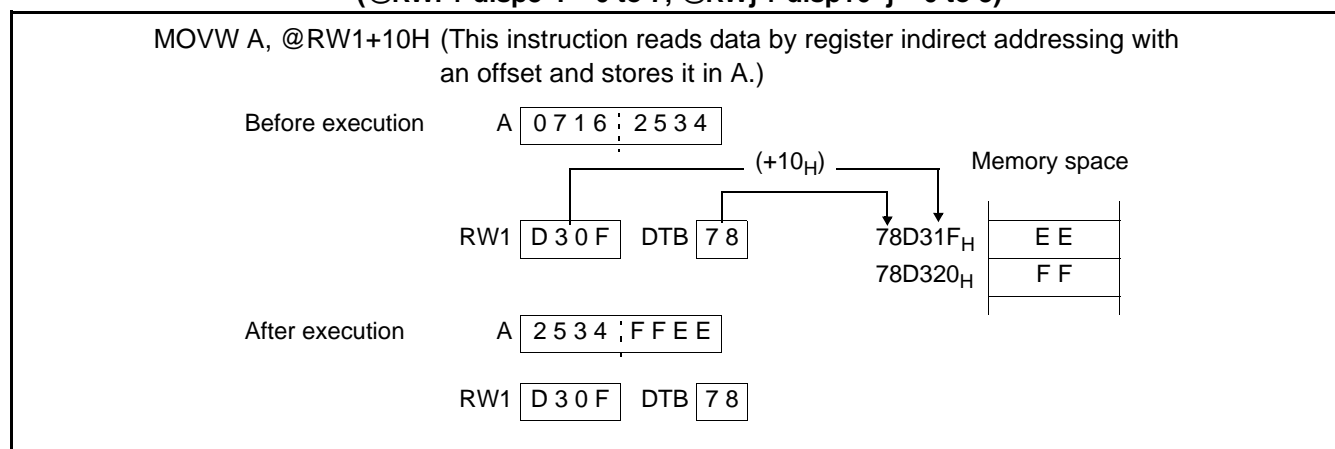
**Figure B.4-2 Example of Register Indirect Addressing with Post Increment (@RWj+ j = 0 to 3)**



● Register indirect addressing with offset (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)

Memory is accessed using the address obtained by adding an offset to the contents of general-purpose register RWj. Two types of offset, byte and word offsets, are used. They are added as signed numeric values. Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0, RW1, RW4, or RW5 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 or RW7 is used, or additional data bank register (ADB) when RW2 or RW6 is used.

**Figure B.4-3 Example of Register Indirect Addressing with Offset (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)**

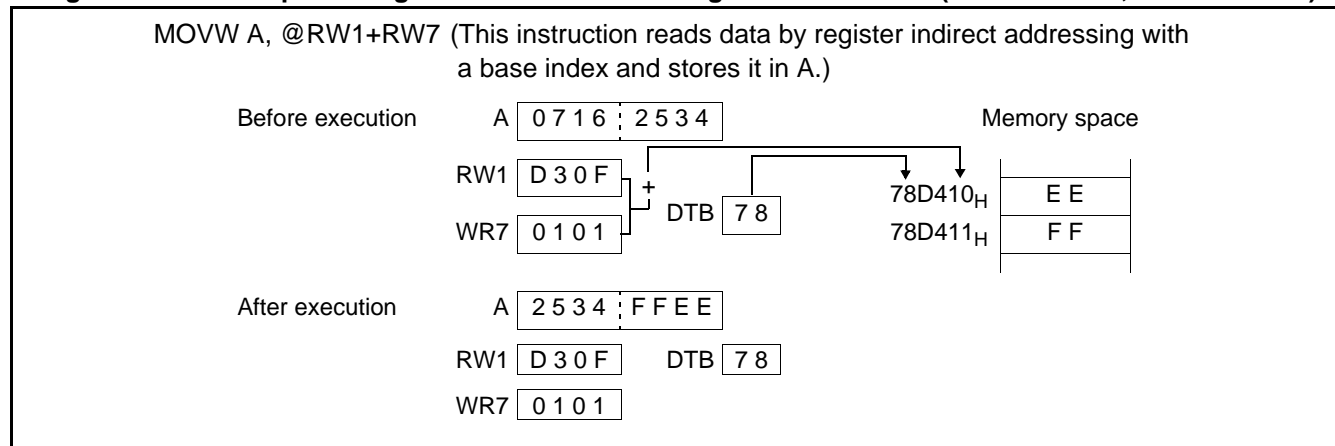




● Register indirect addressing with base index (@RW0 + RW7, @RW1 + RW7)

Memory is accessed using the address determined by adding RW0 or RW1 to the contents of general-purpose register RW7. Address bits 16 to 23 are indicated by the data bank register (DTB).

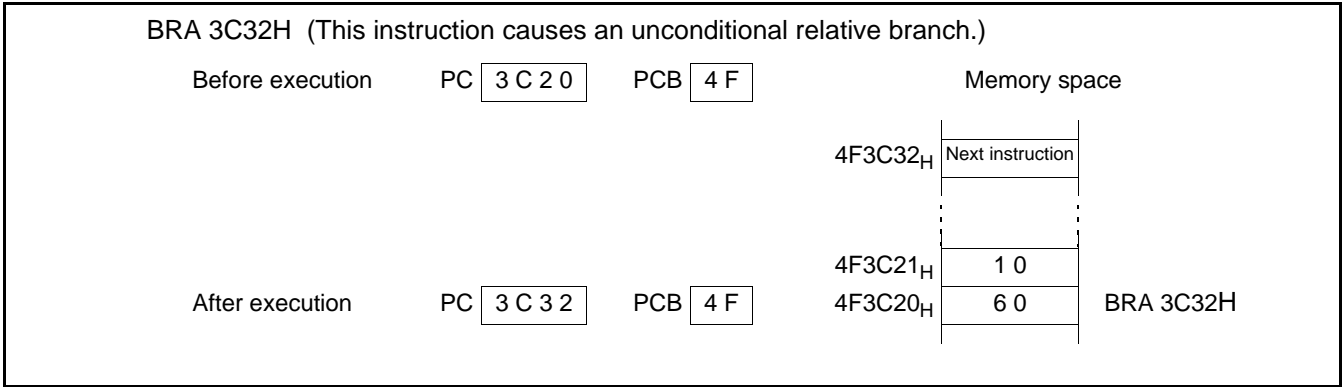
**Figure B.4-6 Example of Register Indirect Addressing with Base Index (@RW0 + RW7, @RW1 + RW7)**



● Program counter relative branch addressing (rel)

The address of the branch destination is a value determined by adding an 8-bit offset to the program counter (PC) value. If the result of addition exceeds 16 bits, bank register incrementing or decrementing is not performed and the excess part is ignored, and therefore the address is contained within a 64-kilobyte bank. This addressing is used for both conditional and unconditional branch instructions. Address bits 16 to 23 are indicated by the program counter bank register (PCB).

Figure B.4-7 Example of Program Counter Relative Branch Addressing (rel)



● Register list (rlst)

Specify a register to be pushed onto or popped from a stack.

Figure B.4-8 Configuration of the Register List

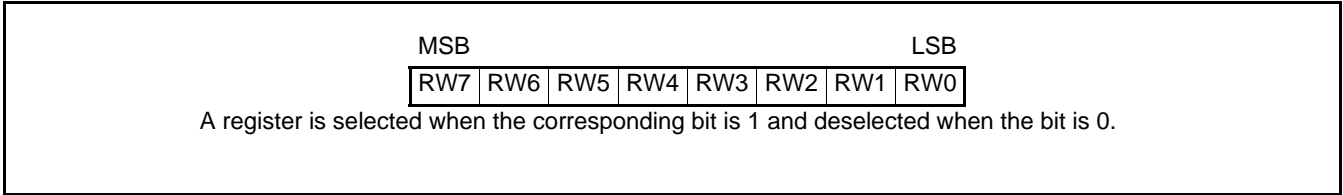
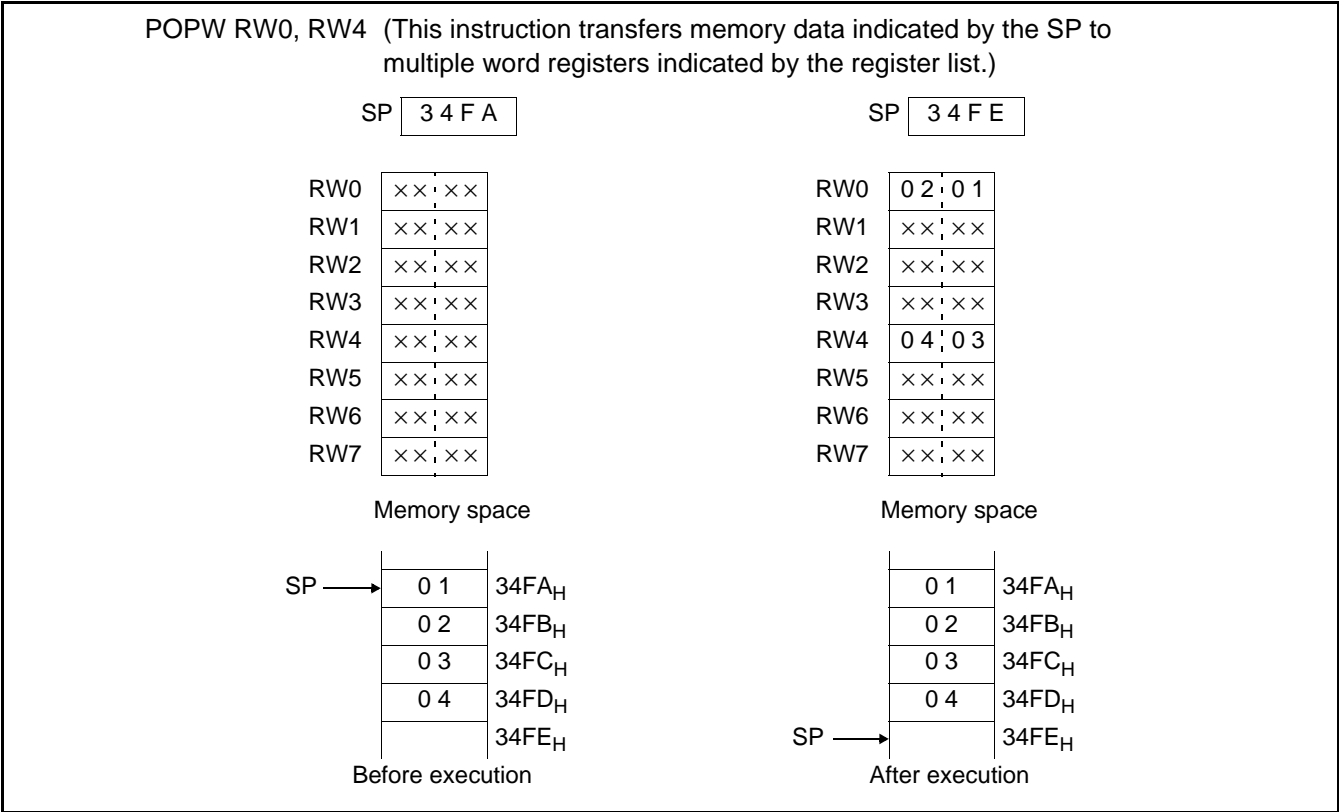


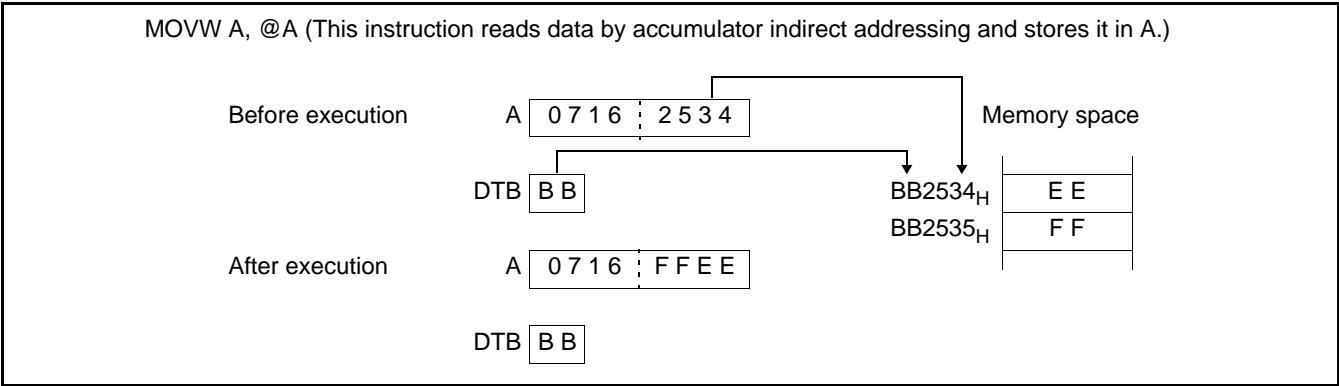
Figure B.4-9 Example of Register List (rlist)



● Accumulator indirect addressing (@A)

Memory is accessed using the address indicated by the contents of the low-order bytes (16 bits) of the accumulator (AL). Address bits 16 to 23 are specified by a mnemonic in the data bank register (DTB).

Figure B.4-10 Example of Accumulator Indirect Addressing (@A)

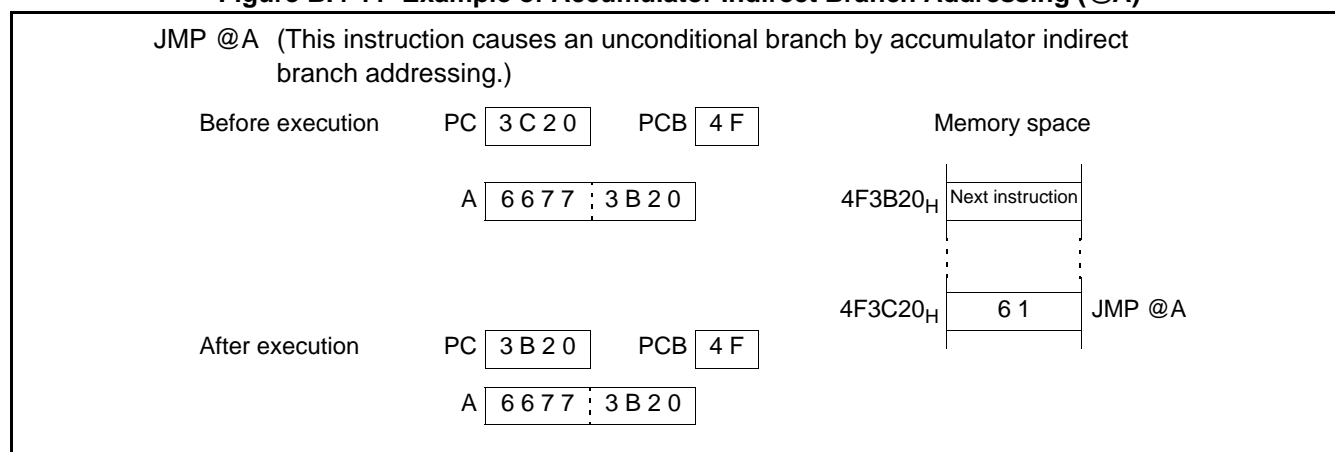




● Accumulator indirect branch addressing (@A)

The address of the branch destination is the content (16 bits) of the low-order bytes (AL) of the accumulator. It indicates the branch destination in the bank address space. Address bits 16 to 23 are specified by the program counter bank register (PCB). For the Jump Context (JCTX) instruction, however, address bits 16 to 23 are specified by the data bank register (DTB). This addressing is used for unconditional branch instructions.

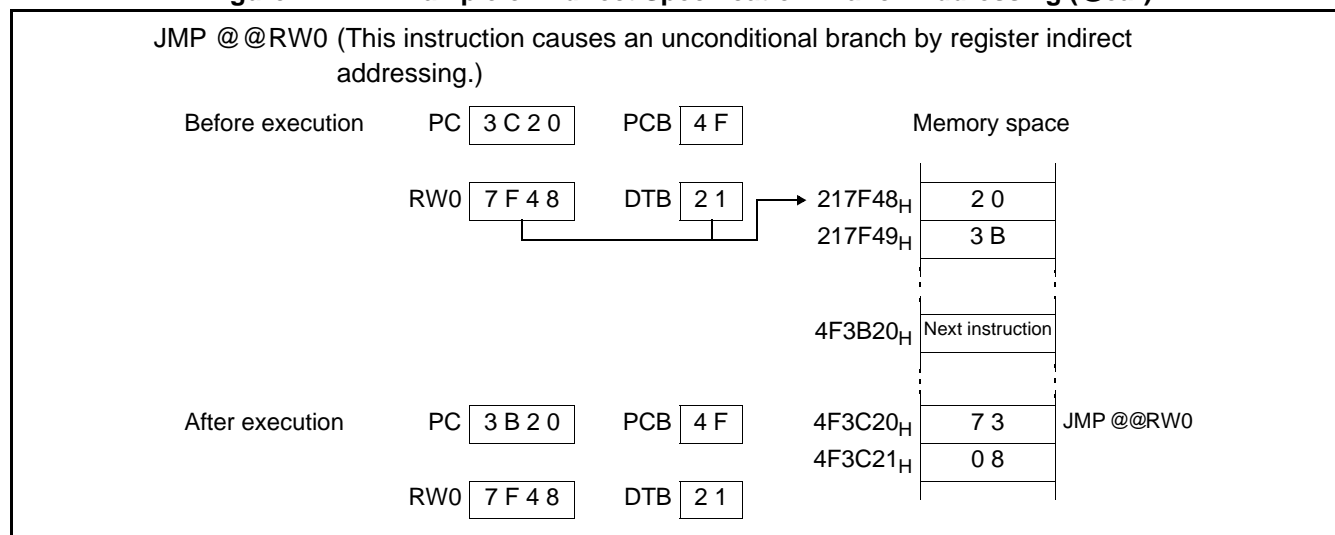
**Figure B.4-11 Example of Accumulator Indirect Branch Addressing (@A)**



● Indirect specification branch addressing (@ear)

The address of the branch destination is the word data at the address indicated by ear.

**Figure B.4-12 Example of Indirect Specification Branch Addressing (@ear)**





## **B.5 Execution Cycle Count**

---

**The number of cycles required for instruction execution (execution cycle count) is obtained by adding the number of cycles required for each instruction, "correction value" determined by the condition, and the number of cycles for instruction fetch.**

---

### **■ Execution Cycle Count**

The number of cycles required for instruction execution (execution cycle count) is obtained by adding the number of cycles required for each instruction, "correction value" determined by the condition, and the number of cycles for instruction fetch. In the mode of fetching an instruction from memory such as internal ROM connected to a 16-bit bus, the program fetches the instruction being executed in word increments. Therefore, intervening in data access increases the execution cycle count.

Similarly, in the mode of fetching an instruction from memory connected to an 8-bit external bus, the program fetches every byte of an instruction being executed. Therefore, intervening in data access increases the execution cycle count. In CPU intermittent operation mode, access to a general-purpose register, internal ROM, internal RAM, internal I/O, or external data bus causes the clock to the CPU to halt for the cycle count specified by the CG0 and CG1 bits of the low power consumption mode control register. Therefore, for the cycle count required for instruction execution in CPU intermittent operation mode, add the "access count x cycle count for the halt" as a correction value to the normal execution count.

## ■ Calculating the Execution Cycle Count

Table B.5-1 lists execution cycle counts and Table B.5-2 and Table B.5-3 summarize correction value data.

**Table B.5-1 Execution Cycle Counts in Each Addressing Mode**

Code	Operand	(a) *	Register access count in each addressing mode
		Execution cycle count in each addressing mode	
00   07	Ri Rwi RLi	See the instruction list.	See the instruction list.
08   0B	@RWj	2	1
0C   0F	@RWj+	4	2
10   17	@RWi+disp8	2	1
18   1B	@RWi+disp16	2	1
1C 1D 1E 1F	@RW0+RW7 @RW1+RW7 @PC+disp16 addr16	4 4 2 1	2 2 0 0

\*: (a) is used for ~ (cycle count) and B (correction value) in "B.8 F2MC-16LX Instruction List".

**Table B.5-2 Cycle Count Correction Values for Counting Execution Cycles**

Operand	(b) byte *		(c) word *		(d) long *	
	Cycle count	Access count	Cycle count	Access count	Cycle count	Access count
Internal register	+0	1	+0	1	+0	2
Internal memory Even address	+0	1	+0	1	+0	2
Internal memory Odd address	+0	1	+2	2	+4	4
External data bus 16-bit even address	+1	1	+1	1	+2	2
External data bus 16-bit odd address	+1	1	+4	2	+8	4
External data bus 8-bits	+1	1	+4	2	+8	4

\*: (b), (c), and (d) are used for ~ (cycle count) and B (correction value) in "B.8 F2MC-16LX Instruction List".

**Note:**

When an external data bus is used, the cycle counts during which an instruction is made to wait by ready input or automatic ready must also be added.

**Table B.5-3 Cycle Count Correction Values for Counting Instruction Fetch Cycles**

Instruction	Byte boundary	Word boundary
Internal memory	-	+2
External data bus 16-bits	-	+3
External data bus 8-bits	+3	-

**Notes:**

- When an external data bus is used, the cycle counts during which an instruction is made to wait by ready input or automatic ready must also be added.
- Actually, instruction execution is not delayed by every instruction fetch. Therefore, use the correction values to calculate the worst case.

## B.6 Effective address field

Table B.6-1 shows the effective address field.

■ Effective Address Field

Table B.6-1 Effective Address Field

Code	Representation			Address format	Byte count of extended address part *
00	R0	RW0	RL0	Register direct: Individual parts correspond to the byte, word, and long word types in order from the left.	-
01	R1	RW1	(RL0)		
02	R2	RW2	RL1		
03	R3	RW3	(RL1)		
04	R4	RW4	RL2		
05	R5	RW5	(RL2)		
06	R6	RW6	RL3		
07	R7	RW7	(RL3)		
08	@RW0			Register indirect	0
09	@RW1				
0A	@RW2				
0B	@RW3				
0C	@RW0+			Register indirect with post increment	0
0D	@RW1+				
0E	@RW2+				
0F	@RW3+				
10	@RW0+disp8			Register indirect with 8-bit displacement	1
11	@RW1+disp8				
12	@RW2+disp8				
13	@RW3+disp8				
14	@RW4+disp8				
15	@RW5+disp8				
16	@RW6+disp8				
17	@RW7+disp8				
18	@RW0+disp16			Register indirect with 16-bit displacement	2
19	@RW1+disp16				
1A	@RW2+disp16				
1B	@RW3+disp16				
1C	@RW0+RW7			Register indirect with index	0
1D	@RW1+RW7			Register indirect with index	0
1E	@PC+disp16			PC indirect with 16-bit displacement	2
1F	addr16			Direct address	2

\*1: Each byte count of the extended address part applies to + in the # (byte count) column in "B.8 F2MC-16LX Instruction List".

## B.7 How to Read the Instruction List

Table B.7-1 describes the items used in "B.8 F2MC-16LX Instruction List", and Table B.7-2 describes the symbols used in the same list.

### ■ Description of Instruction Presentation Items and Symbols

**Table B.7-1 Description of Items in the Instruction List (1/2)**

Item	Description
Mnemonic	Uppercase, symbol: Represented as is in the assembler. Lowercase: Rewritten in the assembler. Number of following lowercase: Indicates bit length in the instruction.
#	Indicates the number of bytes.
~	Indicates the number of cycles. See Table B.2-1 for the alphabetical letters in items.
RG	Indicates the number of times a register access is performed during instruction execution. The number is used to calculate the correction value for CPU intermittent operation.
B	Indicates the correction value used to calculate the actual number of cycles during instruction execution. The actual number of cycles during instruction execution can be determined by adding the value in the ~ column to this value.
Operation	Indicates the instruction operation.
LH	Indicates the special operation for bit15 to bit08 of the accumulator. Z: Transfers 0. X: Transfers after sign extension. -: No transfer
AH	Indicates the special operation for the 16 high-order bits of the accumulator. *: Transfers from AL to AH. -: No transfer Z: Transfers 00 to AH. X: Transfers 00 <sub>H</sub> or FF <sub>H</sub> to AH after AL sign extension.

**Table B.7-1 Description of Items in the Instruction List (1/2)**

Item	Description
I	Each indicates the state of each flag: I (interrupt enable), S (stack), T (sticky bit), N (negative), Z (zero), V (overflow), C (carry). *: Changes upon instruction execution. -: No change S: Set upon instruction execution. R: Reset upon instruction execution.
S	
T	
N	
Z	
V	
C	
RMW	Indicates whether the instruction is a Read Modify Write instruction (reading data from memory by the I instruction and writing the result to memory). *: Read Modify Write instruction -: Not Read Modify Write instruction <b>Note:</b> Cannot be used for an address that has different meanings between read and write operations.

**Table B.7-2 Explanation on Symbols in the Instruction List (1/2)**

Symbol	Explanation
A	The bit length used varies depending on the 32-bit accumulator instruction. Byte: Low-order 8 bits of byte AL Word: 16 bits of word AL Long word: 32 bits of AL and AH
AH	16 high-order bits of A
AL	16 low-order bits of A
SP	Stack pointer (USP or SSP)
PC	Program counter
PCB	program counter bank register
DTB	Data bank register
ADB	Additional data bank register
SSB	System stack bank register
USB	User stack bank register
SPB	Current stack bank register (SSB or USB)
DPR	Direct page register
brg1	DTB, ADB, SSB, USB, DPR, PCB, SPB
brg2	DTB, ADB, SSB, USB, DPR, SPB



**Table B.7-2 Explanation on Symbols in the Instruction List (1/2)**

Symbol	Explanation
Ri	R0, R1, R2, R3, R4, R5, R6, R7
RWi	RW0, RW1, RW2, RW3, RW4, RW5, RW6, RW7
RWj	RW0, RW1, RW2, RW3
RLi	RL0, RL1, RL2, RL3
dir	Abbreviated direct addressing
addr16	Direct addressing
addr24	Physical direct addressing
ad24 0-15	Bit0 to bit15 of addr24
ad24 16-23	Bit16 to bit23 of addr24
io	I/O area (000000 <sub>H</sub> to 0000FF <sub>H</sub> )
#imm4	4-bit immediate data
#imm8	8-bit immediate data
#imm16	16-bit immediate data
#imm32	32-bit immediate data
ext (imm8)	16-bit data obtained by sign extension of 8-bit immediate data
disp8	8-bit displacement
disp16	16-bit displacement
bp	Bit offset
vct4	Vector number (0 to 15)
vct8	Vector number (0 to 255)
( ) b	Bit address
rel	PC relative branch
ear	Effective addressing (code 00 <sub>H</sub> to 07 <sub>H</sub> )
eam	Effective addressing (code 08 <sub>H</sub> to 1F <sub>H</sub> )
rlst	Register list

B.8 F<sup>2</sup>MC-16LX Instruction List

Table B.8-1 to Table B.8-18 list the instructions used by the F<sup>2</sup>MC-16LX.

■ F<sup>2</sup>MC-16LX Instruction List

Table B.8-1 41 Transfer Instructions (Byte)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOV A,dir	2	3	0	(b)	byte (A) ← (dir)	Z	*	-	-	-	*	*	-	-	-
MOV A,addr16	3	4	0	(b)	byte (A) ← (addr16)	Z	*	-	-	-	*	*	-	-	-
MOV A,Ri	1	2	1	0	byte (A) ← (Ri)	Z	*	-	-	-	*	*	-	-	-
MOV A,ear	2	2	1	0	byte (A) ← (ear)	Z	*	-	-	-	*	*	-	-	-
MOV A,eam	2+	3 + (a)	0	(b)	byte (A) ← (eam)	Z	*	-	-	-	*	*	-	-	-
MOV A,io	2	3	0	(b)	byte (A) ← (io)	Z	*	-	-	-	*	*	-	-	-
MOV A,#imm8	2	2	0	0	byte (A) ← imm8	Z	*	-	-	-	*	*	-	-	-
MOV A,@A	2	3	0	(b)	byte (A) ← ((A))	Z	-	-	-	-	*	*	-	-	-
MOV A,@RLi+disp8	3	10	2	(b)	byte (A) ← ((RLi)+disp8)	Z	*	-	-	-	*	*	-	-	-
MOVN A,#imm4	1	1	0	0	byte (A) ← imm4	Z	*	-	-	-	R	*	-	-	-
MOVX A,dir	2	3	0	(b)	byte (A) ← (dir)	X	*	-	-	-	*	*	-	-	-
MOVX A,addr16	3	4	0	(b)	byte (A) ← (addr16)	X	*	-	-	-	*	*	-	-	-
MOVX A,Ri	2	2	1	0	byte (A) ← (Ri)	X	*	-	-	-	*	*	-	-	-
MOVX A,ear	2	2	1	0	byte (A) ← (ear)	X	*	-	-	-	*	*	-	-	-
MOVX A,eam	2+	3 + (a)	0	(b)	byte (A) ← (eam)	X	*	-	-	-	*	*	-	-	-
MOVX A,io	2	3	0	(b)	byte (A) ← (io)	X	*	-	-	-	*	*	-	-	-
MOVX A,#imm8	2	2	0	0	byte (A) ← imm8	X	*	-	-	-	*	*	-	-	-
MOVX A,@A	2	3	0	(b)	byte (A) ← ((A))	X	-	-	-	-	*	*	-	-	-
MOVX A,@RWi+disp8	2	5	1	(b)	byte (A) ← ((RWi)+disp8)	X	*	-	-	-	*	*	-	-	-
MOVX A,@RLi+disp8	3	10	2	(b)	byte (A) ← ((RLi)+disp8)	X	*	-	-	-	*	*	-	-	-
MOV dir,A	2	3	0	(b)	byte (dir) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV addr16,A	3	4	0	(b)	byte (addr16) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV Ri,A	1	2	1	0	byte (Ri) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV ear,A	2	2	1	0	byte (ear) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV eam,A	2+	3 + (a)	0	(b)	byte (eam) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV io,A	2	3	0	(b)	byte (io) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV @RLi+disp8,A	3	10	2	(b)	byte ((RLi)+disp8) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV Ri,ear	2	3	2	0	byte (Ri) ← (ear)	-	-	-	-	-	*	*	-	-	-
MOV Ri,eam	2+	4 + (a)	1	(b)	byte (Ri) ← (eam)	-	-	-	-	-	*	*	-	-	-
MOV ear,Ri	2	4	2	0	byte (ear) ← (Ri)	-	-	-	-	-	*	*	-	-	-
MOV eam,Ri	2+	5 + (a)	1	(b)	byte (eam) ← (Ri)	-	-	-	-	-	*	*	-	-	-
MOV Ri,#imm8	2	2	1	0	byte (Ri) ← imm8	-	-	-	-	-	*	*	-	-	-
MOV io,#imm8	3	5	0	(b)	byte (io) ← imm8	-	-	-	-	-	-	-	-	-	-
MOV dir,#imm8	3	5	0	(b)	byte (dir) ← imm8	-	-	-	-	-	-	-	-	-	-
MOV ear,#imm8	3	2	1	0	byte (ear) ← imm8	-	-	-	-	-	*	*	-	-	-
MOV eam,#imm8	3+	4 + (a)	0	(b)	byte (eam) ← imm8	-	-	-	-	-	-	-	-	-	-
MOV @AL,AH	2	3	0	(b)	byte ((A)) ← (AH)	-	-	-	-	-	*	*	-	-	-
XCH A,ear	2	4	2	0	byte (A) ↔ (ear)	Z	-	-	-	-	-	-	-	-	-
XCH A,eam	2+	5 + (a)	0	2 × (b)	byte (A) ↔ (eam)	Z	-	-	-	-	-	-	-	-	-
XCH Ri,ear	2	7	4	0	byte (Ri) ↔ (ear)	-	-	-	-	-	-	-	-	-	-
XCH Ri,eam	2+	9 + (a)	2	2 × (b)	byte (Ri) ↔ (eam)	-	-	-	-	-	-	-	-	-	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) and (b) in the table.

Table B.8-2 38 Transfer Instructions (Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVW A,dir	2	3	0	(c)	word (A) ← (dir)	-	*	-	-	-	*	*	-	-	-
MOVW A,addr16	3	4	0	(c)	word (A) ← (addr16)	-	*	-	-	-	*	*	-	-	-
MOVW A,SP	1	1	0	0	word (A) ← (SP)	-	*	-	-	-	*	*	-	-	-
MOVW A,RWi	1	2	1	0	word (A) ← (RWi)	-	*	-	-	-	*	*	-	-	-
MOVW A,ear	2	2	1	0	word (A) ← (ear)	-	*	-	-	-	*	*	-	-	-
MOVW A,eam	2+	3 + (a)	0	(c)	word (A) ← (eam)	-	*	-	-	-	*	*	-	-	-
MOVW A,io	2	3	0	(c)	word (A) ← (io)	-	*	-	-	-	*	*	-	-	-
MOVW A,@A	2	3	0	(c)	word (A) ← ((A))	-	-	-	-	-	*	*	-	-	-
MOVW A,#imm16	3	2	0	0	word (A) ← imm16	-	*	-	-	-	*	*	-	-	-
MOVW A,@RWi+disp8	2	5	1	(c)	word (A) ← ((RWi)+disp8)	-	*	-	-	-	*	*	-	-	-
MOVW A,@RLi+disp8	3	10	2	(c)	word (A) ← ((RLi)+disp8)	-	*	-	-	-	*	*	-	-	-
MOVW dir,A	2	3	0	(c)	word (dir) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW addr16,A	3	4	0	(c)	word (addr16) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW SPA	1	1	0	0	word (SP) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,A	1	2	1	0	word (RWi) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW ear,A	2	2	1	0	word (ear) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW eam,A	2+	3 + (a)	0	(c)	word (eam) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW io,A	2	3	0	(c)	word (io) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW @RWi+disp8,A	2	5	1	(c)	word ((RWi)+disp8) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW @RLi+disp8,A	3	10	2	(c)	word ((RLi)+disp8) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,ear	2	3	2	0	word (RWi) ← (ear)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,eam	2+	4 + (a)	1	(c)	word (RWi) ← (eam)	-	-	-	-	-	*	*	-	-	-
MOVW ear,RWi	2	4	2	0	word (ear) ← (RWi)	-	-	-	-	-	*	*	-	-	-
MOVW eam,RWi	2+	5 + (a)	1	(c)	word (eam) ← (RWi)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,#imm16	3	2	1	0	word (RWi) ← imm16	-	-	-	-	-	*	*	-	-	-
MOVW io,#imm16	4	5	0	(c)	word (io) ← imm16	-	-	-	-	-	-	-	-	-	-
MOVW ear,#imm16	4	2	1	0	word (ear) ← imm16	-	-	-	-	-	*	*	-	-	-
MOVW eam,#imm16	4+	4 + (a)	0	(c)	word (eam) ← imm16	-	-	-	-	-	-	-	-	-	-
MOVW @AL,AH	2	3	0	(c)	word ((A)) ← (AH)	-	-	-	-	-	*	*	-	-	-
XCHW A,ear	2	4	2	0	word (A) ↔ (ear)	-	-	-	-	-	-	-	-	-	-
XCHW A,eam	2+	5 + (a)	0	2 × (c)	word (A) ↔ (eam)	-	-	-	-	-	-	-	-	-	-
XCHW RWi, ear	2	7	4	0	word (RWi) ↔ (ear)	-	-	-	-	-	-	-	-	-	-
XCHW RWi, eam	2+	9 + (a)	2	2 × (c)	word (RWi) ↔ (eam)	-	-	-	-	-	-	-	-	-	-
MOVL A,ear	2	4	2	0	long (A) ← (ear)	-	-	-	-	-	*	*	-	-	-
MOVL A,eam	2+	5 + (a)	0	(d)	long (A) ← (eam)	-	-	-	-	-	*	*	-	-	-
MOVL A,#imm32	5	3	0	0	long (A) ← imm32	-	-	-	-	-	*	*	-	-	-
MOVL ear,A	2	4	2	0	long (ear) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVL eam,A	2+	5 + (a)	0	(d)	long(eam) ← (A)	-	-	-	-	-	*	*	-	-	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a), (c), and (d) in the table.

Table B.8-3 42 Addition/Subtraction Instructions (Byte, Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ADD A,#imm8	2	2	0	0	byte (A) ← (A) + imm8	Z	-	-	-	-	*	*	*	*	-
ADD A,dir	2	5	0	(b)	byte (A) ← (A) + (dir)	Z	-	-	-	-	*	*	*	*	-
ADD A,ear	2	3	1	0	byte (A) ← (A) + (ear)	Z	-	-	-	-	*	*	*	*	-
ADD A,eam	2+	4 + (a)	0	(b)	byte (A) ← (A) + (eam)	Z	-	-	-	-	*	*	*	*	-
ADD ear,A	2	3	2	0	byte (ear) ← (ear) + (A)	-	-	-	-	-	*	*	*	*	-
ADD eam,A	2+	5 + (a)	0	2 × (b)	byte (eam) ← (eam) + (A)	Z	-	-	-	-	*	*	*	*	*
ADDC A	1	2	0	0	byte (A) ← (AH) + (AL) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC A,ear	2	3	1	0	byte (A) ← (A) + (ear) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC A,eam	2+	4 + (a)	0	(b)	byte (A) ← (A) + (eam) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC A	1	3	0	0	byte (A) ← (AH) + (AL) + (C) (decimal)	Z	-	-	-	-	*	*	*	*	-
SUB A,#imm8	2	2	0	0	byte (A) ← (A) - imm8	Z	-	-	-	-	*	*	*	*	-
SUB A,dir	2	5	0	(b)	byte (A) ← (A) - (dir)	Z	-	-	-	-	*	*	*	*	-
SUB A,ear	2	3	1	0	byte (A) ← (A) - (ear)	Z	-	-	-	-	*	*	*	*	-
SUB A,eam	2+	4 + (a)	0	(b)	byte (A) ← (A) - (eam)	Z	-	-	-	-	*	*	*	*	-
SUB ear,A	2	3	2	0	byte (ear) ← (ear) - (A)	-	-	-	-	-	*	*	*	*	-
SUB eam,A	2+	5 + (a)	0	2 × (b)	byte (eam) ← (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBC A	1	2	0	0	byte (A) ← (AH) - (AL) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC A,ear	2	3	1	0	byte (A) ← (A) - (ear) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC A,eam	2+	4 + (a)	0	(b)	byte (A) ← (A) - (eam) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC A	1	3	0	0	byte (A) ← (AH) - (AL) - (C) (decimal)	Z	-	-	-	-	*	*	*	*	-
ADDW A	1	2	0	0	word (A) ← (AH) + (AL)	-	-	-	-	-	*	*	*	*	-
ADDW A,ear	2	3	1	0	word (A) ← (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDW A,#imm16	3	2	0	0	word (A) ← (A) + imm16	-	-	-	-	-	*	*	*	*	-
ADDW ear,A	2	3	2	0	word (ear) ← (ear) + (A)	-	-	-	-	-	*	*	*	*	-
ADDW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) + (A)	-	-	-	-	-	*	*	*	*	*
ADDCW A,ear	2	3	1	0	word (A) ← (A) + (ear) + (C)	-	-	-	-	-	*	*	*	*	-
ADDCW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) + (eam) + (C)	-	-	-	-	-	*	*	*	*	-
SUBW A	1	2	0	0	word (A) ← (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
SUBW A,ear	2	3	1	0	word (A) ← (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBW A,#imm16	3	2	0	0	word (A) ← (A) - imm16	-	-	-	-	-	*	*	*	*	-
SUBW ear,A	2	3	2	0	word (ear) ← (ear) - (A)	-	-	-	-	-	*	*	*	*	-
SUBW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBCW A,ear	2	3	1	0	word (A) ← (A) - (ear) - (C)	-	-	-	-	-	*	*	*	*	-
SUBCW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) - (eam) - (C)	-	-	-	-	-	*	*	*	*	-
ADDL A,ear	2	6	2	0	long (A) ← (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDL A,#imm32	5	4	0	0	long (A) ← (A) + imm32	-	-	-	-	-	*	*	*	*	-
SUBL A,ear	2	6	2	0	long (A) ← (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBL A,#imm32	5	4	0	0	long (A) ← (A) - imm32	-	-	-	-	-	*	*	*	*	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

**Table B.8-4 12 Increment/decrement Instructions (Byte, Word, Long Word)**

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
INC	ear	2	3	2	0	byte (ear) $\leftarrow$ (ear) + 1	-	-	-	-	-	*	*	*	-	-
INC	eam	2+	5+(a)	0	2 $\times$ (b)	byte (eam) $\leftarrow$ (eam) + 1	-	-	-	-	-	*	*	*	-	*
DEC	ear	2	3	2	0	byte (ear) $\leftarrow$ (ear) - 1	-	-	-	-	-	*	*	*	-	-
DEC	eam	2+	5+(a)	0	2 $\times$ (b)	byte (eam) $\leftarrow$ (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCW	ear	2	3	2	0	word (ear) $\leftarrow$ (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCW	eam	2+	5+(a)	0	2 $\times$ (c)	word (eam) $\leftarrow$ (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECW	ear	2	3	2	0	word (ear) $\leftarrow$ (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECW	eam	2+	5+(a)	0	2 $\times$ (c)	word (eam) $\leftarrow$ (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCL	ear	2	7	4	0	long (ear) $\leftarrow$ (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCL	eam	2+	9+(a)	0	2 $\times$ (d)	long (eam) $\leftarrow$ (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECL	ear	2	7	4	0	long (ear) $\leftarrow$ (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECL	eam	2+	9+(a)	0	2 $\times$ (d)	long (eam) $\leftarrow$ (eam) - 1	-	-	-	-	-	*	*	*	-	*

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

**Table B.8-5 11 Compare Instructions (Byte, Word, Long Word)**

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CMP	A	1	1	0	0	byte (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMP	A,ear	2	2	1	0	byte (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMP	A,eam	2+	3+(a)	0	(b)	byte (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMP	A,#imm8	2	2	0	0	byte (A) - imm8	-	-	-	-	-	*	*	*	*	-
CMPW	A	1	1	0	0	word (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMPW	A,ear	2	2	1	0	word (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPW	A,eam	2+	3+(a)	0	(c)	word (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPW	A,#imm16	3	2	0	0	word (A) - imm16	-	-	-	-	-	*	*	*	*	-
CMPL	A,ear	2	6	2	0	long (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPL	A,eam	2+	7+(a)	0	(d)	long (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPL	A,#imm32	5	3	0	0	long (A) - imm32	-	-	-	-	-	*	*	*	*	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-6 11 Unsigned Multiplication/Division Instructions (Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIVU A	1	*1	0	0	word (AH) / byte (AL) quotient → byte (AL) remainder → byte (AH)	-	-	-	-	-	-	-	*	*	-
DIVU A,ear	2	*2	1	0	word (A) / byte (ear) quotient → byte (A) remainder → byte (ear)	-	-	-	-	-	-	-	*	*	-
DIVU A,eam	2+	*3	0	*6	word (A) / byte (eam) quotient → byte (A) remainder → byte (eam)	-	-	-	-	-	-	-	*	*	-
DIVUW A,ear	2	*4	1	0	long (A) / word (ear) quotient → word (A) remainder → word (ear)	-	-	-	-	-	-	-	*	*	-
DIVUW A,eam	2+	*5	0	*7	long (A) / word (eam) quotient → word (A) remainder → word (eam)	-	-	-	-	-	-	-	*	*	-
MULU A	1	*8	0	0	byte (AH) * byte (AL) → word (A)	-	-	-	-	-	-	-	-	-	-
MULU A,ear	2	*9	1	0	byte (A) * byte (ear) → word (A)	-	-	-	-	-	-	-	-	-	-
MULU A,eam	2+	*10	0	(b)	byte (A) * byte (eam) → word (A)	-	-	-	-	-	-	-	-	-	-
MULUW A	1	*11	0	0	word (AH) * word (AL) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW A,ear	2	*12	1	0	word (A) * word (ear) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW A,eam	2+	*13	0	(c)	word (A) * word (eam) → Long (A)	-	-	-	-	-	-	-	-	-	-

\*1: 3: Division by 0 7: Overflow 15: Normal  
 \*2: 4: Division by 0 8: Overflow 16: Normal  
 \*3: 6+(a): Division by 0 9+(a): Overflow 19+(a): Normal  
 \*4: 4: Division by 0 7: Overflow 22: Normal  
 \*5: 6+(a): Division by 0 8+(a): Overflow 26+(a): Normal  
 \*6: (b): Division by 0 or overflow 2 × (b): Normal  
 \*7: (c): Division by 0 or overflow 2 × (c): Normal  
 \*8: 3: Byte (AH) is 0. 7: Byte (AH) is not 0.  
 \*9: 4: Byte (ear) is 0. 8: Byte (ear) is not 0.  
 \*10: 5+(a): Byte (eam) is 0, 9+(a): Byte (eam) is not 0.  
 \*11: 3: Word (AH) is 0. 11: Word (AH) is not 0.  
 \*12: 4: Word (ear) is 0. 12: Word (ear) is not 0.  
 \*13: 5+(a): Word (eam) is 0. 13+(a): Word (eam) is not 0.

## Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.

Table B.8-7 11 Signed Multiplication/Division Instructions (Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIV A	2	*1	0	0	word (AH) / byte (AL) quotient → byte (AL) remainder → byte (AH)	Z	-	-	-	-	-	-	*	*	-
DIV A,ear	2	*2	1	0	word (A) / byte (ear) quotient → byte (A) remainder → byte (ear)	Z	-	-	-	-	-	-	*	*	-
DIV A,eam	2+	*3	0	*6	word (A) / byte (eam) quotient → byte (A) remainder → byte (eam)	Z	-	-	-	-	-	-	*	*	-
DIVW A,ear	2	*4	1	0	long (A) / word (ear) quotient → word (A) remainder → word (ear)	-	-	-	-	-	-	-	*	*	-
DIVW A,eam	2+	*5	0	*7	long (A) / word (eam) quotient → word (A) remainder → word (eam)	-	-	-	-	-	-	-	*	*	-
MUL A	2	*8	0	0	byte (AH) * byte (AL) → word (A)	-	-	-	-	-	-	-	-	-	-
MUL A,ear	2	*9	1	0	byte (A) * byte (ear) → word (A)	-	-	-	-	-	-	-	-	-	-
MUL A,eam	2+	*10	0	(b)	byte (A) * byte (eam) → word (A)	-	-	-	-	-	-	-	-	-	-
MULW A	2	*11	0	0	word (AH) * word (AL) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULW A,ear	2	*12	1	0	word (A) * word (ear) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULW A,eam	2+	*13	0	(c)	word (A) * word (eam) → Long (A)	-	-	-	-	-	-	-	-	-	-

\*1: 3: Division by 0, 8 or 18: Overflow, 18: Normal  
\*2: 4: Division by 0, 11 or 22: Overflow, 23: Normal  
\*3: 5+(a): Division by 0, 12+(a) or 23+(a): Overflow, 24+(a): Normal  
\*4: When dividend is positive; 4: Division by 0, 12 or 30: Overflow, 31: Normal  
When dividend is negative; 4: Division by 0, 12 or 31: Overflow, 32: Normal  
\*5: When dividend is positive; 5+(a): Division by 0, 12+(a) or 31+(a): Overflow, 32+(a): Normal  
When dividend is negative; 5+(a): Division by 0, 12+(a) or 32+(a): Overflow, 33+(a): Normal  
\*6: (b): Division by 0 or overflow, 2 × (b): Normal  
\*7: (c): Division by 0 or overflow, 2 × (c): Normal  
\*8: 3: Byte (AH) is 0, 12: result is positive, 13: result is negative  
\*9: 4: Byte (ear) is 0, 13: result is positive, 14: result is negative  
\*10: 5+(a): Byte (eam) is 0, 14+(a): result is positive, 15+(a): result is negative  
\*11: 3: Word (AH) is 0, 16: result is positive, 19: result is negative  
\*12: 4: Word (ear) is 0, 17: result is positive, 20: result is negative  
\*13: 5+(a): Word (eam) is 0, 18+(a): result is positive, 21+(a): result is negative

Notes:

- The execution cycle count found when an overflow occurs in a DIV or DIVW instruction may be a pre-operation count or a post-operation count depending on the detection timing.
- When an overflow occurs with DIV or DIVW instruction, the contents of the AL are destroyed.
- See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.

Table B.8-8 39 Logic 1 Instructions (Byte, Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
AND A,#imm8	2	2	0	0	byte (A) ← (A) and imm8	-	-	-	-	-	*	*	R	-	-
AND A,ear	2	3	1	0	byte (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
AND A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
AND ear,A	2	3	2	0	byte (ear) ← (ear) and (A)	-	-	-	-	-	*	*	R	-	-
AND eam,A	2+	5+(a)	0	2 × (b)	byte (eam) ← (eam) and (A)	-	-	-	-	-	*	*	R	-	*
OR A,#imm8	2	2	0	0	byte (A) ← (A) or imm8	-	-	-	-	-	*	*	R	-	-
OR A,ear	2	3	1	0	byte (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
OR A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
OR ear,A	2	3	2	0	byte (ear) ← (ear) or (A)	-	-	-	-	-	*	*	R	-	-
OR eam,A	2+	5+(a)	0	2 × (b)	byte (eam) ← (eam) or (A)	-	-	-	-	-	*	*	R	-	*
XOR A,#imm8	2	2	0	0	byte (A) ← (A) xor imm8	-	-	-	-	-	*	*	R	-	-
XOR A,ear	2	3	1	0	byte (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XOR A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XOR ear,A	2	3	2	0	byte (ear) ← (ear) xor (A)	-	-	-	-	-	*	*	R	-	-
XOR eam,A	2+	5+(a)	0	2 × (b)	byte (eam) ← (eam) xor (A)	-	-	-	-	-	*	*	R	-	*
NOT A	1	2	0	0	byte (A) ← not (A)	-	-	-	-	-	*	*	R	-	-
NOT ear	2	3	2	0	byte (ear) ← not (ear)	-	-	-	-	-	*	*	R	-	-
NOT eam	2+	5+(a)	0	2 × (b)	byte (eam) ← not (eam)	-	-	-	-	-	*	*	R	-	*
ANDW A	1	2	0	0	word (A) ← (AH) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW A,#imm16	3	2	0	0	word (A) ← (A) and imm16	-	-	-	-	-	*	*	R	-	-
ANDW A,ear	2	3	1	0	word (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ANDW ear,A	2	3	2	0	word (ear) ← (ear) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) and (A)	-	-	-	-	-	*	*	R	-	*
ORW A	1	2	0	0	word (A) ← (AH) or (A)	-	-	-	-	-	*	*	R	-	-
ORW A,#imm16	3	2	0	0	word (A) ← (A) or imm16	-	-	-	-	-	*	*	R	-	-
ORW A,ear	2	3	1	0	word (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
ORW ear,A	2	3	2	0	word (ear) ← (ear) or (A)	-	-	-	-	-	*	*	R	-	-
ORW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) or (A)	-	-	-	-	-	*	*	R	-	*
XORW A	1	2	0	0	word (A) ← (AH) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW A,#imm16	3	2	0	0	word (A) ← (A) xor imm16	-	-	-	-	-	*	*	R	-	-
XORW A,ear	2	3	1	0	word (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XORW ear,A	2	3	2	0	word (ear) ← (ear) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW eam,A	2+	5+(a)	0	2 × (c)	word (eam) ← (eam) xor (A)	-	-	-	-	-	*	*	R	-	*
NOTW A	1	2	0	0	word (A) ← not (A)	-	-	-	-	-	*	*	R	-	-
NOTW ear	2	3	2	0	word (ear) ← not (ear)	-	-	-	-	-	*	*	R	-	-
NOTW eam	2+	5+(a)	0	2 × (c)	word (eam) ← not (eam)	-	-	-	-	-	*	*	R	-	*

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.



**Table B.8-9 6 Logic 2 Instructions (Long Word)**

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ANDL A,ear	2	6	2	0	long (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ORL A,ear	2	6	2	0	long (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
XORL A,ear	2	6	2	0	long (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) and (d) in the table.

**Table B.8-10 6 Sign Inversion Instructions (Byte, Word)**

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NEG A	1	2	0	0	byte (A) ← 0 - (A)	X	-	-	-	-	*	*	*	*	-
NEG ear	2	3	2	0	byte (ear) ← 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEG eam	2+	5+(a)	0	2 × (b)	byte (eam) ← 0 - (eam)	-	-	-	-	-	*	*	*	*	*
NEGW A	1	2	0	0	word (A) ← 0 - (A)	-	-	-	-	-	*	*	*	*	-
NEGW ear	2	3	2	0	word (ear) ← 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEGW eam	2+	5+(a)	0	2 × (c)	word (eam) ← 0 - (eam)	-	-	-	-	-	*	*	*	*	*

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (c) in the table.

**Table B.8-11 1 Normalization Instruction (Long Word)**

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NRML A,R0	2	*1	1	0	long (A) ← Shift left to the position where '1' is set for the first time. byte (R0) ← Shift count at that time	-	-	-	-	-	-	*	-	-	-

\*1: 4 when all accumulators have a value of 0; otherwise, 6+(R0)

Table B.8-12 18 Shift Instructions (Byte, Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
RORC A	2	2	0	0	byte (A) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	-
ROLC A	2	2	0	0	byte (A) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	-
RORC ear	2	3	2	0	byte (ear) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	-
RORC eam	2+	5+(a)	0	2 × (b)	byte (eam) ← Right rotation with carry	-	-	-	-	-	*	*	-	*	*
ROLC ear	2	3	2	0	byte (ear) ← Left rotation with carry	-	-	-	-	-	*	*	-	*	-
ROLC eam	2+	5+(a)	0	2 × (b)	byte (eam) ← Left rotation with carry	-	-	-	-	-	*	*	-	*	*
ASR A,R0	2	*1	1	0	byte (A) ← Arithmetic right shift (A, 1 bit)	-	-	-	-	*	*	*	-	*	-
LSR A,R0	2	*1	1	0	byte (A) ← Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSL A,R0	2	*1	1	0	byte (A) ← Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-
ASRW A	1	2	0	0	word (A) ← Arithmetic right shift (A, 1 bit)	-	-	-	-	*	*	*	-	*	-
LSRW A/SHRW A	1	2	0	0	word (A) ← Logical right shift (A, 1 bit)	-	-	-	-	*	R	*	-	*	-
LSLW A/SHLW A	1	2	0	0	word (A) ← Logical left shift (A, 1 bit)	-	-	-	-	-	*	*	-	*	-
ASRW A,R0	2	*1	1	0	word (A) ← Arithmetic right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSRW A,R0	2	*1	1	0	word (A) ← Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSLW A,R0	2	*1	1	0	word (A) ← Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-
ASRL A,R0	2	*2	1	0	long (A) ← Arithmetic right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSRL A,R0	2	*2	1	0	long (A) ← Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSLL A,R0	2	*2	1	0	long (A) ← Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-

\*1: 6 when R0 is 0; otherwise, 5 + (R0)

\*2: 6 when R0 is 0; otherwise, 6 + (R0)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) and (b) in the table.

Table B.8-13 31 Branch 1 Instructions

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
BZ/BEQ rel	2	*1	0	0	Branch on (Z) = 1	-	-	-	-	-	-	-	-	-	-
BNZ/ rel BNE	2	*1	0	0	Branch on (Z) = 0	-	-	-	-	-	-	-	-	-	-
BC/BLO rel	2	*1	0	0	Branch on (C) = 1	-	-	-	-	-	-	-	-	-	-
BNC/ rel BHS	2	*1	0	0	Branch on (C) = 0	-	-	-	-	-	-	-	-	-	-
BN rel	2	*1	0	0	Branch on (N) = 1	-	-	-	-	-	-	-	-	-	-
BP rel	2	*1	0	0	Branch on (N) = 0	-	-	-	-	-	-	-	-	-	-
BV rel	2	*1	0	0	Branch on (V) = 1	-	-	-	-	-	-	-	-	-	-
BNV rel	2	*1	0	0	Branch on (V) = 0	-	-	-	-	-	-	-	-	-	-
BT rel	2	*1	0	0	Branch on (T) = 1	-	-	-	-	-	-	-	-	-	-
BNT rel	2	*1	0	0	Branch on (T) = 0	-	-	-	-	-	-	-	-	-	-
BLT rel	2	*1	0	0	Branch on (V) xor (N) = 1	-	-	-	-	-	-	-	-	-	-
BGE rel	2	*1	0	0	Branch on (V) xor (N) = 0	-	-	-	-	-	-	-	-	-	-
BLE rel	2	*1	0	0	Branch on ((V) xor (N)) or (Z) = 1	-	-	-	-	-	-	-	-	-	-
BGT rel	2	*1	0	0	Branch on ((V) xor (N)) or (Z) = 0	-	-	-	-	-	-	-	-	-	-
BLS rel	2	*1	0	0	Branch on (C) or (Z) = 1	-	-	-	-	-	-	-	-	-	-
BHI rel	2	*1	0	0	Branch on (C) or (Z) = 0	-	-	-	-	-	-	-	-	-	-
BRA rel	2	*1	0	0	Unconditional branch	-	-	-	-	-	-	-	-	-	-
JMP @A	1	2	0	0	word (PC) ← (A)	-	-	-	-	-	-	-	-	-	-
JMP addr16	3	3	0	0	word (PC) ← addr16	-	-	-	-	-	-	-	-	-	-
JMP @ear	2	3	1	0	word (PC) ← (ear)	-	-	-	-	-	-	-	-	-	-
JMP @eam	2+	4+(a)	0	(c)	word (PC) ← (eam)	-	-	-	-	-	-	-	-	-	-
JMPP @ear *3	2	5	2	0	word (PC) ← (ear), (PCB) ← (ear+2)	-	-	-	-	-	-	-	-	-	-
JMPP @eam *3	2+	6+(a)	0	(d)	word (PC) ← (eam), (PCB) ← (eam+2)	-	-	-	-	-	-	-	-	-	-
JMPP addr24	4	4	0	0	word (PC) ← ad24 0-15, (PCB) ← ad24 16-23	-	-	-	-	-	-	-	-	-	-
CALL @ear *4	2	6	1	(c)	word (PC) ← (ear)	-	-	-	-	-	-	-	-	-	-
CALL @eam *4	2+	7+(a)	0	2 × (c)	word (PC) ← (eam)	-	-	-	-	-	-	-	-	-	-
CALL addr16 *5	3	6	0	(c)	word (PC) ← addr16	-	-	-	-	-	-	-	-	-	-
CALLV #vct4 *5	1	7	0	2 × (c)	Vector call instruction	-	-	-	-	-	-	-	-	-	-
CALLP @ear *6	2	10	2	2 × (c)	word (PC) ← (ear), (PCB) ← (ear+2)	-	-	-	-	-	-	-	-	-	-
CALLP @eam *6	2+	11+(a)	0	*2	word (PC) ← (eam), (PCB) ← (eam+2)	-	-	-	-	-	-	-	-	-	-
CALLP addr24 *7	4	10	0	2 × (c)	word (PC) ← ad24 0-15, (PCB) ← ad24 16-23	-	-	-	-	-	-	-	-	-	-

\*1: 4 when a branch is made; otherwise, 3

\*2:  $3 \times (c) + (b)$

\*3: Read (word) of branch destination address

\*4: W: Save to stack (word) R: Read (word) of branch destination address

\*5: Save to stack (word)

\*6: W: Save to stack (long word), R: Read (long word) of branch destination address

\*7: Save to stack (long word)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-14 19 Branch 2 Instructions

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CBNE A,#imm8,rel	3	*1	0	0	Branch on byte (A) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CWBNE A,#imm16,rel	4	*1	0	0	Branch on word (A) not equal to imm16	-	-	-	-	-	*	*	*	*	-
CBNE ear,#imm8,rel	4	*2	1	0	Branch on byte (ear) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CBNE eam,#imm8,rel *9	4+	*3	0	(b)	Branch on byte (eam) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CWBNE ear,#imm16,rel	5	*4	1	0	Branch on word (ear) not equal to imm16	-	-	-	-	-	*	*	*	*	-
CWBNE eam,#imm16,rel*9	5+	*3	0	(c)	Branch on word (eam) not equal to imm16	-	-	-	-	-	*	*	*	*	-
DBNZ ear,rel	3	*5	2	0	byte (ear) ← (ear) - 1, Branch on (ear) not equal to 0	-	-	-	-	-	*	*	*	-	-
DBNZ eam,rel	3+	*6	2	2 × (b)	byte (eam) ← (eam) - 1, Branch on (eam) not equal to 0	-	-	-	-	-	*	*	*	-	*
DWBNZ ear,rel	3	*5	2	0	word (ear) ← (ear) - 1, Branch on (ear) not equal to 0	-	-	-	-	-	*	*	*	-	-
DWBNZ eam,rel	3+	*6	2	2 × (c)	word (eam) ← (eam) - 1, Branch on (eam) not equal to 0	-	-	-	-	-	*	*	*	-	*
INT #vct8	2	20	0	8 × (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT addr16	3	16	0	6 × (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INTP addr24	4	17	0	6 × (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT9	1	20	0	8 × (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
RETI	1	*8	0	*7	Return from interrupt	-	-	*	*	*	*	*	*	*	-
LINK #imm8	2	6	0	(c)	Saves the old frame pointer in the stack upon entering the function, then sets the new frame pointer and reserves the local pointer area.	-	-	-	-	-	-	-	-	-	-
UNLINK	1	5	0	(c)	Recovers the old frame pointer from the stack upon exiting the function.	-	-	-	-	-	-	-	-	-	-
RET *10	1	4	0	(c)	Return from subroutine	-	-	-	-	-	-	-	-	-	-
RETP *11	1	6	0	(d)	Return from subroutine	-	-	-	-	-	-	-	-	-	-

\*1: 5 when a branch is made; otherwise, 4

\*2: 13 when a branch is made; otherwise, 12

\*3: 7+(a) when a branch is made; otherwise, 6+(a)

\*4: 8 when a branch is made; otherwise, 7

\*5: 7 when a branch is made; otherwise, 6

\*6: 8+(a) when a branch is made; otherwise, 7+(a)

\*7: 3 × (b) + 2 × (c) when jumping to the next interruption request; 6 × (c) when returning from the current interruption

\*8: 15 when jumping to the next interruption request; 17 when returning from the current interruption

\*9: Do not use RWj+ addressing mode with a CBNE or CWBNE instruction.

\*10: Return from stack (word)

\*11: Return from stack (long word)

## Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-15 28 Other Control Instructions (Byte, Word, Long Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
PUSHW A	1	4	0	(c)	word (SP) $\leftarrow$ (SP) - 2, ((SP)) $\leftarrow$ (A)	-	-	-	-	-	-	-	-	-	-
PUSHW AH	1	4	0	(c)	word (SP) $\leftarrow$ (SP) - 2, ((SP)) $\leftarrow$ (AH)	-	-	-	-	-	-	-	-	-	-
PUSHW PS	1	4	0	(c)	word (SP) $\leftarrow$ (SP) - 2, ((SP)) $\leftarrow$ (PS)	-	-	-	-	-	-	-	-	-	-
PUSHW rlst	2	*3	*5	*4	(SP) $\leftarrow$ (SP) - 2n, ((SP)) $\leftarrow$ (rlst)	-	-	-	-	-	-	-	-	-	-
POPW A	1	3	0	(c)	word (A) $\leftarrow$ ((SP)), (SP) $\leftarrow$ (SP) + 2	-	*	-	-	-	-	-	-	-	-
POPW AH	1	3	0	(c)	word (AH) $\leftarrow$ ((SP)), (SP) $\leftarrow$ (SP) + 2	-	-	-	-	-	-	-	-	-	-
POPW PS	1	4	0	(c)	word (PS) $\leftarrow$ ((SP)), (SP) $\leftarrow$ (SP) + 2	-	-	*	*	*	*	*	*	*	-
POPW rlst	2	*2	*5	*4	(rlst) $\leftarrow$ ((SP)), (SP) $\leftarrow$ (SP) + 2n	-	-	-	-	-	-	-	-	-	-
JCTX @A	1	14	0	6 $\times$ (c)	Context switch instruction	-	-	*	*	*	*	*	*	*	-
AND CCR,#imm8	2	3	0	0	byte (CCR) $\leftarrow$ (CCR) and imm8	-	-	*	*	*	*	*	*	*	-
OR CCR,#imm8	2	3	0	0	byte (CCR) $\leftarrow$ (CCR) or imm8	-	-	*	*	*	*	*	*	*	-
MOV RP,#imm8	2	2	0	0	byte (RP) $\leftarrow$ imm8	-	-	-	-	-	-	-	-	-	-
MOV ILM,#imm8	2	2	0	0	byte (ILM) $\leftarrow$ imm8	-	-	-	-	-	-	-	-	-	-
MOVEA RWi,ear	2	3	1	0	word (RWi) $\leftarrow$ ear	-	-	-	-	-	-	-	-	-	-
MOVEA RWi,eam	2+	2+(a)	1	0	word (RWi) $\leftarrow$ eam	-	-	-	-	-	-	-	-	-	-
MOVEA A,ear	2	1	0	0	word (A) $\leftarrow$ ear	-	*	-	-	-	-	-	-	-	-
MOVEA A,eam	2+	1+(a)	0	0	word (A) $\leftarrow$ eam	-	*	-	-	-	-	-	-	-	-
ADDSP #imm8	2	3	0	0	word (SP) $\leftarrow$ (SP) + ext(imm8)	-	-	-	-	-	-	-	-	-	-
ADDSP #imm16	3	3	0	0	word (SP) $\leftarrow$ (SP) + imm16	-	-	-	-	-	-	-	-	-	-
MOV A,brg1	2	*1	0	0	byte (A) $\leftarrow$ (brg1)	Z	*	-	-	-	*	*	-	-	-
MOV brg2,A	2	1	0	0	byte (brg2) $\leftarrow$ (A)	-	-	-	-	-	*	*	-	-	-
NOP	1	1	0	0	No operation	-	-	-	-	-	-	-	-	-	-
ADB	1	1	0	0	Prefix code for AD space access	-	-	-	-	-	-	-	-	-	-
DTB	1	1	0	0	Prefix code for DT space access	-	-	-	-	-	-	-	-	-	-
PCB	1	1	0	0	Prefix code for PC space access	-	-	-	-	-	-	-	-	-	-
SPB	1	1	0	0	Prefix code for SP space access	-	-	-	-	-	-	-	-	-	-
NCC	1	1	0	0	Prefix code for flag no-change	-	-	-	-	-	-	-	-	-	-
CMR	1	1	0	0	Prefix code for common register bank	-	-	-	-	-	-	-	-	-	-

\*1: PCB, ADB, SSB, USB, SPB: 1, DTB, DPR: 2

\*2:  $7 + 3 \times (\text{POP count}) + 2 \times (\text{POP last register number})$ , 7 when RLST = 0 (no transfer register)

\*3:  $29 + 3 \times (\text{PUSH count}) - 3 \times (\text{PUSH last register number})$ , 8 when RLST = 0 (no transfer register)

\*4: (POP count)  $\times$  (c) or (PUSH count)  $\times$  (c)

\*5: (POP count) or (PUSH count)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) and (c) in the table.

Table B.8-16 21 Bit Operand Instructions

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVB A,dir:bp	3	5	0	(b)	byte (A) $\leftarrow$ (dir:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB A,addr16:bp	4	5	0	(b)	byte (A) $\leftarrow$ (addr16:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB A,io:bp	3	4	0	(b)	byte (A) $\leftarrow$ (io:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB dir:bp,A	3	7	0	$2 \times (b)$	bit (dir:bp)b $\leftarrow$ (A)	-	-	-	-	-	*	*	-	-	*
MOVB addr16:bp,A	4	7	0	$2 \times (b)$	bit (addr16:bp)b $\leftarrow$ (A)	-	-	-	-	-	*	*	-	-	*
MOVB io:bp,A	3	6	0	$2 \times (b)$	bit (io:bp)b $\leftarrow$ (A)	-	-	-	-	-	*	*	-	-	*
SETB dir:bp	3	7	0	$2 \times (b)$	bit (dir:bp)b $\leftarrow$ 1	-	-	-	-	-	-	-	-	-	*
SETB addr16:bp	4	7	0	$2 \times (b)$	bit (addr16:bp)b $\leftarrow$ 1	-	-	-	-	-	-	-	-	-	*
SETB io:bp	3	7	0	$2 \times (b)$	bit (io:bp)b $\leftarrow$ 1	-	-	-	-	-	-	-	-	-	*
CLRB dir:bp	3	7	0	$2 \times (b)$	bit (dir:bp)b $\leftarrow$ 0	-	-	-	-	-	-	-	-	-	*
CLRB addr16:bp	4	7	0	$2 \times (b)$	bit (addr16:bp)b $\leftarrow$ 0	-	-	-	-	-	-	-	-	-	*
CLRB io:bp	3	7	0	$2 \times (b)$	bit (io:bp)b $\leftarrow$ 0	-	-	-	-	-	-	-	-	-	*
BBC dir:bp,rel	4	*1	0	(b)	Branch on (dir:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBC addr16:bp,rel	5	*1	0	(b)	Branch on (addr16:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBC io:bp,rel	4	*2	0	(b)	Branch on (io:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBS dir:bp,rel	4	*1	0	(b)	Branch on (dir:bp) b = 1	-	-	-	-	-	-	*	-	-	-
BBS addr16:bp,rel	5	*1	0	(b)	Branch on (addr16:bp) b = 1	-	-	-	-	-	-	*	-	-	-
BBS io:bp,rel	4	*2	0	(b)	Branch on (io:bp) b = 1	-	-	-	-	-	-	*	-	-	-
SBBS addr16:bp,rel	5	*3	0	$2 \times (b)$	Branch on (addr16:bp) b = 1, bit (addr16:bp) b $\leftarrow$ 1	-	-	-	-	-	-	*	-	-	*
WBTS io:bp	3	*4	0	*5	Waits until (io:bp) b = 1	-	-	-	-	-	-	-	-	-	-
WBTC io:bp	3	*4	0	*5	Waits until (io:bp) b = 0	-	-	-	-	-	-	-	-	-	-

\*1: 8 when a branch is made; otherwise, 7

\*2: 7 when a branch is made; otherwise, 6

\*3: 10 when the condition is met; otherwise, 9

\*4: Undefined count

\*5: Until the condition is met

Note:

See Table B.5-1 and Table B.5-2 for information on (b) in the table.

Table B.8-17 6 Accumulator Operation Instructions (Byte, Word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
SWAP	1	3	0	0	byte (A)0-7 $\leftrightarrow$ (A)8-15	-	-	-	-	-	-	-	-	-	-
SWAPW	1	2	0	0	word (AH) $\leftrightarrow$ (AL)	-	*	-	-	-	-	-	-	-	-
EXT	1	1	0	0	Byte sign extension	X	-	-	-	-	*	*	-	-	-
EXTW	1	2	0	0	Word sign extension	-	X	-	-	-	*	*	-	-	-
ZEXT	1	1	0	0	Byte zero extension	Z	-	-	-	-	R	*	-	-	-
ZEXTW	1	1	0	0	Word zero extension	-	Z	-	-	-	R	*	-	-	-

Table B.8-18 10 String Instructions

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVS / MOVSI	2	*2	*5	*3	byte transfer @AH+ ← @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSD	2	*2	*5	*3	byte transfer @AH- ← @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCEQ / SCEQI	2	*1	*8	*4	byte search @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
SCEQD	2	*1	*8	*4	byte search @AH- ← AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
FILS / FILSI	2	6m+6	*8	*3	byte fill @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	-	-	-
MOVSW / MOVSWI	2	*2	*5	*6	word transfer @AH+ ← @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSWD	2	*2	*5	*6	word transfer @AH- ← @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCWEQ / SCWEQI	2	*1	*8	*7	word search @AH+ - AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
SCWEQD	2	*1	*8	*7	word search @AH- - AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
FILSW / FILSWI	2	6m+6	*8	*6	word fill @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	-	-	-

\*1: 5 when RW0 is 0,  $4 + 7 \times (RW0)$  when the counter expires, or  $7n + 5$  when a match occurs

\*2: 5 when RW0 is 0; otherwise,  $4 + 8 \times (RW0)$

\*3:  $(b) \times (RW0) + (b) \times (RW0)$  When the source and destination access different areas, calculate the (b) item individually.

\*4:  $(b) \times n$

\*5:  $2 \times (b) \times (RW0)$

\*6:  $(c) \times (RW0) + (c) \times (RW0)$  When the source and destination access different areas, calculate the (c) item individually.

\*7:  $(c) \times n$

\*8:  $(b) \times (RW0)$

Note:

m: RW0 value (counter value), n: Loop count

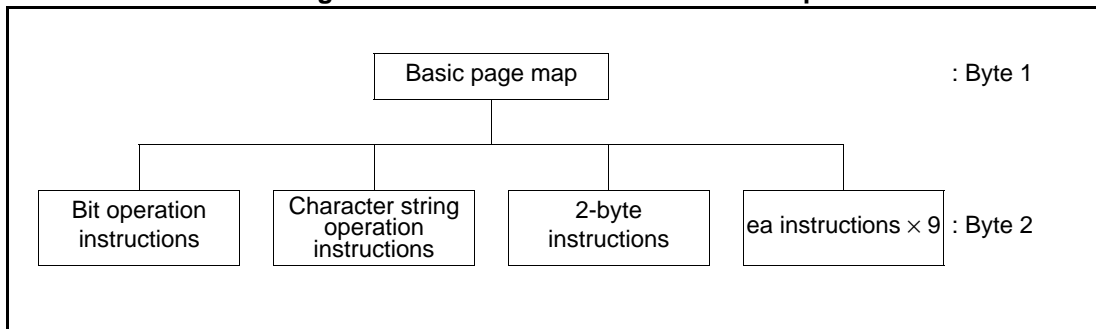
See Table B.5-1 and Table B.5-2 for information on (b) and (c) in the table.

## B.9 Instruction Map

Each F<sup>2</sup>MC-16LX instruction code consists of 1 or 2 bytes. Therefore, the instruction map consists of multiple pages. Table B.9-2 to Table B.9-21 summarize the F<sup>2</sup>MC-16LX instruction map.

### ■ Structure of Instruction Map

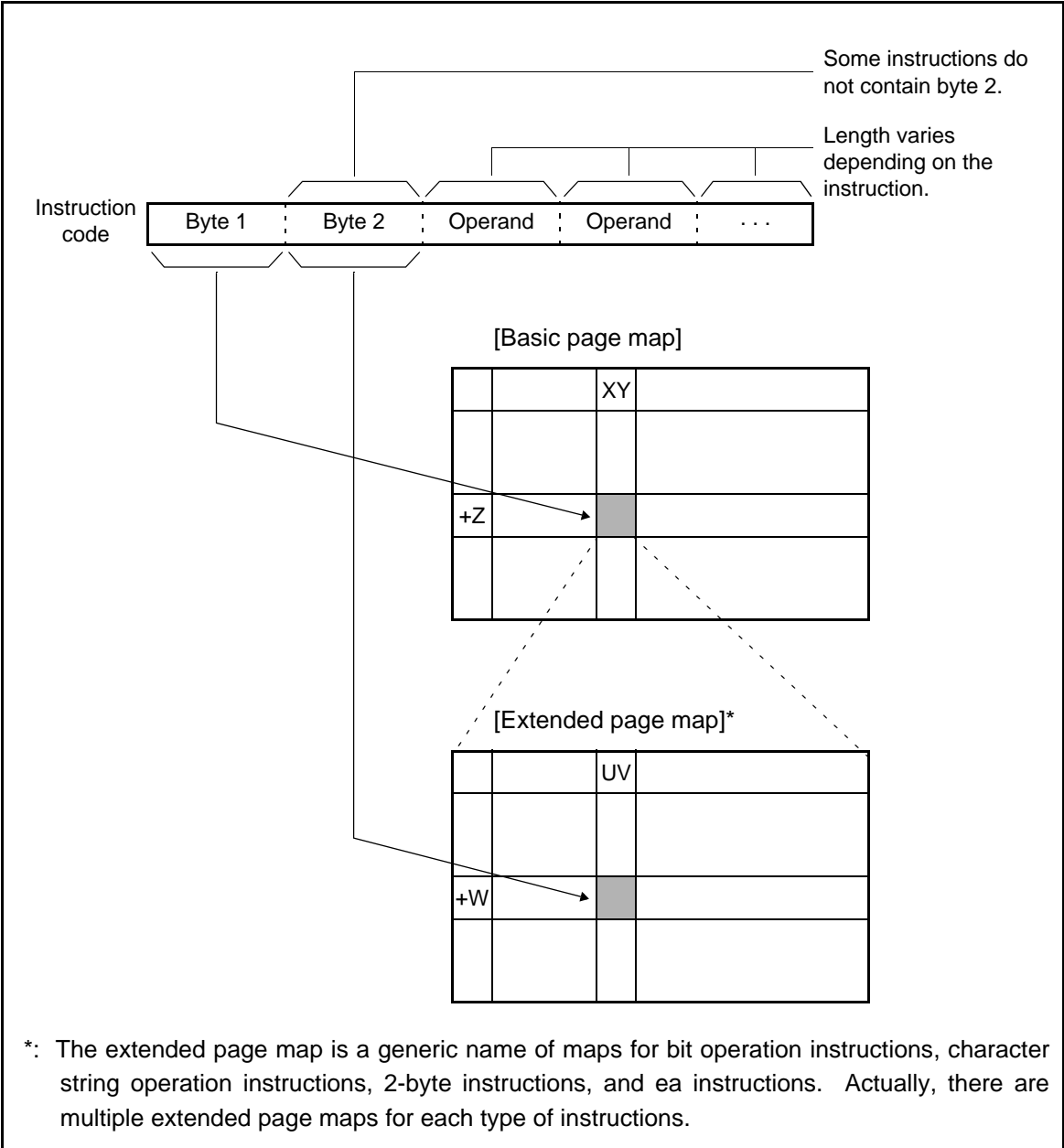
Figure B.9-1 Structure of Instruction Map



An instruction such as the NOP instruction that ends in one byte is completed within the basic page. An instruction such as the MOVS instruction that requires two bytes recognizes the existence of byte 2 when it references byte 1, and can check the following one byte by referencing the map for byte 2. Figure B.9-2 shows the correspondence between an actual instruction code and instruction map.



Figure B.9-2 Correspondence between Actual Instruction Code and Instruction Map



An example of an instruction code is shown in Table B.9-1.

Table B.9-1 Example of an Instruction Code

Instruction	Byte 1 (from basic page map)	Byte 2 (from extended page map)
NOP	00 +0=00	-
AND A, #8	30 +4=34	-
MOV A, ADB	60 +F=6F	00 +0=00
CBNE @RW2+d8, #8, rel	70 +0=70	F0 +2=F2

Table B.9-2 Basic Page Map

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	NOP	CMR	ADD A, dir	ADD A, #8	MOV A, dir	MOV A, io	BRA rel	ea instruction 1	MOV A, Ri	MOV Ri, A	MOV Ri, #8	MOVX A, Ri	MOVX A, @RiHd8	MOVN A, #4	CALL #vct4	BZ/BEQ rel
+1	INT9	NCC	SUB A, dir	SUB A, #8	MOV dir, A	MOV io, A	JMP @A	ea instruction 2								BNZ/BNE rel
+2	ADDDC A	SUBDC A	ADDC A	SUBC A	MOV A, #8	MOV A, addr16	JMP addr16	ea instruction 3								BC/BLO rel
+3	NEG A	JCTX @A	CMP A	CMP A, #8	MOVX A, #8	MOV addr16, A	JMPP addr24	ea instruction 4								BNC/BHS rel
+4	PCB	EXT	AND CCR, #8	AND A, #8	MOV dir, #8	MOV io, #8	CALL addr16	ea instruction 5								BN rel
+5	DTB	ZEXT	OR CCR, #8	OR A, #8	MOVX A, dir	MOVX A, io	CALLP addr24	ea instruction 6								BP rel
+6	ADB	SWAP	DIVU A	XOR A, #8	MOVW A, SP	MOVW io, #16	RETP	ea instruction 7								BV rel
+7	SPB	ADDSP #8	MULU A	NOT A	MOVW SP, A	MOVX A, addr16	RET	ea instruction 8								BNV rel
+8	LINK #imm8	ADDL A, #32	ADDW A	ADDW A, #16	MOVW A, dir	MOVW A, io	INT #vct8	ea instruction 9	MOVW A, RWi	MOVW RWi, A	MOVW RWi, #16	MOVW A, @RWiHd8	MOVW @RWiHd8, A			BT rel
+9	UNLINK	SUBL A, #32	SUBW A	SUBW A, #16	MOVW dir, A	MOVW io, A	INT addr16	MOVEA RWi, ea								BNT rel
+A	MOV RP, #8	MOV ILM, #8	CBNE A, #8, rel	CWBNE A, #16, rel	MOVW A, #16	MOVW A, addr16	INTP addr24	MOV Ri, ea								BLT rel
+B	NEGW A	CMPL A, #32	CMPW A	CMPW A, #16	MOVL A, #32	MOVW addr16, A	RETI	MOVW RWi, ea								BGE rel
+C	LSLW A	EXTW	ANDW A	ANDW A, #16	PUSHW A	POPW A	Bit operation instruction	MOV ea, Ri								BLE rel
+D		ZEXTW	ORW A	ORW A, #16	PUSHW AH	POPW AH		MOVW ea, RWi								BGT rel
+E	ASRW A	SWAPW A	XORW A	XORW A, #16	PUSHW PS	POPW PS	Character string operation instruction	XCH Ri, ea								BLS rel
+F	LSRW A	ADDSP #16	MULW A	NOTW A	PUSHW r1st	POPW r1st	2-byte instruction	XCHW RWi, ea								BHI rel

**Table B.9-3 Bit Operation Instruction Map (First Byte = 6C<sub>H</sub>)**

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOVB A, io:bp		MOVB io:bp, A		CLRB io:bp		SETB io:bp		BBC io:bp, rel		BBS io:bp, rel		WBTS io:bp		WBTC io:bp	
+1																
+2																
+3																
+4																
+5																
+6																
+7																
+8	MOVB A, dir:bp	MOVB A, addr16:bp	MOVB dir:bp, A	MOVB addr16:bp, A	CLRB dir:bp	CLRB addr16:bp	SETB dir:bp	SETB addr16:bp	BBC dir:bp, rel	BBC addr16:bp, rel	BBS dir:bp, rel	BBS addr16:bp, rel				SBBS addr16:bp
+9																
+A																
+B																
+C																
+D																
+E																
+F																

Table B.9-4 Character String Operation Instruction Map (First Byte = 6E<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOVSI PCB, PCB	MOVSD PCB, PCB	MOVSWI PCB, PCB	MOVSWD PCB, PCB					SCEQI PCB	SCEQD PCB	SCWEQI PCB	SCWEQD PCB	FILSI PCB		FILSWI PCB	
+1	PCB, DTB								DTB	DTB	DTB	DTB	DTB		DTB	
+2	PCB, ADB								ADB	ADB	ADB	ADB	ADB		ADB	
+3	PCB, SPB								SPB	SPB	SPB	SPB	SPB		SPB	
+4	DTB, PCB															
+5	DTB, DTB															
+6	DTB, ADB															
+7	DTB, SPB															
+8	ADB, PCB															
+9	ADB, DTB															
+A	ADB, ADB															
+B	ADB, SPB															
+C	SPB, PCB															
+D	SPB, DTB															
+E	SPB, ADB															
+F	SPB, SPB															

**Table B.9-5 2-byte Instruction Map (First Byte = 6FH)**

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV A, DTB	MOV DTB, A	MOVX A, @RL0+d8	MOV @RL0+d8, A	MOV A, @RL0+d8											
+1	MOV A, ADB	MOV ADB, A														
+2	MOV A, SSB	MOV SSB, A	MOVX A, @RL1+d8	MOV @RL1+d8, A	MOV A, @RL1+d8											
+3	MOV A, USB	MOV USB, A														
+4	MOV A, DPR	MOV DPR, A	MOVX A, @RL2+d8	MOV @RL2+d8, A	MOV A, @RL2+d8											
+5	MOV A, @A	MOV @AL, AH														
+6	MOV A, PCB	MOV A, @A	MOVX A, @RL3+d8	MOV @RL3+d8, A	MOV A, @RL3+d8											
+7	ROL A	ROL A														
+8				MOVW @RL0+d8, A	MOVW A, @RL0+d8			MUL A								
+9								MULW A								
+A				MOVW @RL1+d8, A	MOVW A, @RL1+d8			DIV A								
+B																
+C	LSLW A, R0	LSLL A, R0	LSL A, R0	MOVW @RL2+d8, A	MOVW A, @RL2+d8											
+D	MOVW A, @A	MOVW @AL, AH	NRML A, R0													
+E	ASRW A, R0	ASRL A, R0	ASR A, R0	MOVW @RL3+d8, A	MOVW A, @RL3+d8											
+F	LSRW A, R0	LSRL A, R0	LSR A, R0													

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
					CWbNE ↓, CWbNE ↓										CBNE ↓, CBNE ↓	
+0	ADDL A, A, RL0' @RW0+d8	ADDL A, A, RL0' @RW0+d8	SUBL A, A, RL0' @RW0+d8	SUBL A, A, RL0' @RW0+d8	RW0, @RW0+d8 #16, rel'	CMPL A, A, RL0' @RW0+d8	CMPL A, A, RL0' @RW0+d8	CMPL A, A, RL0' @RW0+d8	ANDL A, A, RL0' @RW0+d8	ANDL A, A, RL0' @RW0+d8	ORL A, A, RL0' @RW0+d8	ORL A, A, RL0' @RW0+d8	XORL A, A, RL0' @RW0+d8	XORL A, A, RL0' @RW0+d8	R0, @RW0+d8 #8, rel'	R0, @RW0+d8 #8, rel'
+1	ADDL A, A, RL0' @RW1+d8	ADDL A, A, RL0' @RW1+d8	SUBL A, A, RL0' @RW1+d8	SUBL A, A, RL0' @RW1+d8	RW1, @RW1+d8 #16, rel'	CMPL A, A, RL0' @RW1+d8	CMPL A, A, RL0' @RW1+d8	CMPL A, A, RL0' @RW1+d8	ANDL A, A, RL0' @RW1+d8	ANDL A, A, RL0' @RW1+d8	ORL A, A, RL0' @RW1+d8	ORL A, A, RL0' @RW1+d8	XORL A, A, RL0' @RW1+d8	XORL A, A, RL0' @RW1+d8	R1, @RW1+d8 #8, rel'	R1, @RW1+d8 #8, rel'
+2	ADDL A, A, RL1' @RW2+d8	ADDL A, A, RL1' @RW2+d8	SUBL A, A, RL1' @RW2+d8	SUBL A, A, RL1' @RW2+d8	RW2, @RW2+d8 #16, rel'	CMPL A, A, RL1' @RW2+d8	CMPL A, A, RL1' @RW2+d8	CMPL A, A, RL1' @RW2+d8	ANDL A, A, RL1' @RW2+d8	ANDL A, A, RL1' @RW2+d8	ORL A, A, RL1' @RW2+d8	ORL A, A, RL1' @RW2+d8	XORL A, A, RL1' @RW2+d8	XORL A, A, RL1' @RW2+d8	R2, @RW2+d8 #8, rel'	R2, @RW2+d8 #8, rel'
+3	ADDL A, A, RL1' @RW3+d8	ADDL A, A, RL1' @RW3+d8	SUBL A, A, RL1' @RW3+d8	SUBL A, A, RL1' @RW3+d8	RW3, @RW3+d8 #16, rel'	CMPL A, A, RL1' @RW3+d8	CMPL A, A, RL1' @RW3+d8	CMPL A, A, RL1' @RW3+d8	ANDL A, A, RL1' @RW3+d8	ANDL A, A, RL1' @RW3+d8	ORL A, A, RL1' @RW3+d8	ORL A, A, RL1' @RW3+d8	XORL A, A, RL1' @RW3+d8	XORL A, A, RL1' @RW3+d8	R3, @RW3+d8 #8, rel'	R3, @RW3+d8 #8, rel'
+4	ADDL A, A, RL2' @RW4+d8	ADDL A, A, RL2' @RW4+d8	SUBL A, A, RL2' @RW4+d8	SUBL A, A, RL2' @RW4+d8	RW4, @RW4+d8 #16, rel'	CMPL A, A, RL2' @RW4+d8	CMPL A, A, RL2' @RW4+d8	CMPL A, A, RL2' @RW4+d8	ANDL A, A, RL2' @RW4+d8	ANDL A, A, RL2' @RW4+d8	ORL A, A, RL2' @RW4+d8	ORL A, A, RL2' @RW4+d8	XORL A, A, RL2' @RW4+d8	XORL A, A, RL2' @RW4+d8	R4, @RW4+d8 #8, rel'	R4, @RW4+d8 #8, rel'
+5	ADDL A, A, RL2' @RW5+d8	ADDL A, A, RL2' @RW5+d8	SUBL A, A, RL2' @RW5+d8	SUBL A, A, RL2' @RW5+d8	RW5, @RW5+d8 #16, rel'	CMPL A, A, RL2' @RW5+d8	CMPL A, A, RL2' @RW5+d8	CMPL A, A, RL2' @RW5+d8	ANDL A, A, RL2' @RW5+d8	ANDL A, A, RL2' @RW5+d8	ORL A, A, RL2' @RW5+d8	ORL A, A, RL2' @RW5+d8	XORL A, A, RL2' @RW5+d8	XORL A, A, RL2' @RW5+d8	R5, @RW5+d8 #8, rel'	R5, @RW5+d8 #8, rel'
+6	ADDL A, A, RL3' @RW6+d8	ADDL A, A, RL3' @RW6+d8	SUBL A, A, RL3' @RW6+d8	SUBL A, A, RL3' @RW6+d8	RW6, @RW6+d8 #16, rel'	CMPL A, A, RL3' @RW6+d8	CMPL A, A, RL3' @RW6+d8	CMPL A, A, RL3' @RW6+d8	ANDL A, A, RL3' @RW6+d8	ANDL A, A, RL3' @RW6+d8	ORL A, A, RL3' @RW6+d8	ORL A, A, RL3' @RW6+d8	XORL A, A, RL3' @RW6+d8	XORL A, A, RL3' @RW6+d8	R6, @RW6+d8 #8, rel'	R6, @RW6+d8 #8, rel'
+7	ADDL A, A, RL3' @RW7+d8	ADDL A, A, RL3' @RW7+d8	SUBL A, A, RL3' @RW7+d8	SUBL A, A, RL3' @RW7+d8	RW7, @RW7+d8 #16, rel'	CMPL A, A, RL3' @RW7+d8	CMPL A, A, RL3' @RW7+d8	CMPL A, A, RL3' @RW7+d8	ANDL A, A, RL3' @RW7+d8	ANDL A, A, RL3' @RW7+d8	ORL A, A, RL3' @RW7+d8	ORL A, A, RL3' @RW7+d8	XORL A, A, RL3' @RW7+d8	XORL A, A, RL3' @RW7+d8	R7, @RW7+d8 #8, rel'	R7, @RW7+d8 #8, rel'
+8	ADDL A, A, @RW0, @RW0+d16	ADDL A, A, @RW0, @RW0+d16	SUBL A, A, @RW0, @RW0+d16	SUBL A, A, @RW0, @RW0+d16	@RW0, @RW0+d16 #16, rel'	CMPL A, A, @RW0, @RW0+d16	CMPL A, A, @RW0, @RW0+d16	CMPL A, A, @RW0, @RW0+d16	ANDL A, A, @RW0, @RW0+d16	ANDL A, A, @RW0, @RW0+d16	ORL A, A, @RW0, @RW0+d16	ORL A, A, @RW0, @RW0+d16	XORL A, A, @RW0, @RW0+d16	XORL A, A, @RW0, @RW0+d16	@RW0, @RW0+d16 #8, rel'	@RW0, @RW0+d16 #8, rel'
+9	ADDL A, A, @RW1, @RW1+d16	ADDL A, A, @RW1, @RW1+d16	SUBL A, A, @RW1, @RW1+d16	SUBL A, A, @RW1, @RW1+d16	@RW1, @RW1+d16 #16, rel'	CMPL A, A, @RW1, @RW1+d16	CMPL A, A, @RW1, @RW1+d16	CMPL A, A, @RW1, @RW1+d16	ANDL A, A, @RW1, @RW1+d16	ANDL A, A, @RW1, @RW1+d16	ORL A, A, @RW1, @RW1+d16	ORL A, A, @RW1, @RW1+d16	XORL A, A, @RW1, @RW1+d16	XORL A, A, @RW1, @RW1+d16	@RW1, @RW1+d16 #8, rel'	@RW1, @RW1+d16 #8, rel'
+A	ADDL A, A, @RW2, @RW2+d16	ADDL A, A, @RW2, @RW2+d16	SUBL A, A, @RW2, @RW2+d16	SUBL A, A, @RW2, @RW2+d16	@RW2, @RW2+d16 #16, rel'	CMPL A, A, @RW2, @RW2+d16	CMPL A, A, @RW2, @RW2+d16	CMPL A, A, @RW2, @RW2+d16	ANDL A, A, @RW2, @RW2+d16	ANDL A, A, @RW2, @RW2+d16	ORL A, A, @RW2, @RW2+d16	ORL A, A, @RW2, @RW2+d16	XORL A, A, @RW2, @RW2+d16	XORL A, A, @RW2, @RW2+d16	@RW2, @RW2+d16 #8, rel'	@RW2, @RW2+d16 #8, rel'
+B	ADDL A, A, @RW3, @RW3+d16	ADDL A, A, @RW3, @RW3+d16	SUBL A, A, @RW3, @RW3+d16	SUBL A, A, @RW3, @RW3+d16	@RW3, @RW3+d16 #16, rel'	CMPL A, A, @RW3, @RW3+d16	CMPL A, A, @RW3, @RW3+d16	CMPL A, A, @RW3, @RW3+d16	ANDL A, A, @RW3, @RW3+d16	ANDL A, A, @RW3, @RW3+d16	ORL A, A, @RW3, @RW3+d16	ORL A, A, @RW3, @RW3+d16	XORL A, A, @RW3, @RW3+d16	XORL A, A, @RW3, @RW3+d16	@RW3, @RW3+d16 #8, rel'	@RW3, @RW3+d16 #8, rel'
+C	ADDL A, A, @RW0+, @RW0+RW7	ADDL A, A, @RW0+, @RW0+RW7	SUBL A, A, @RW0+, @RW0+RW7	SUBL A, A, @RW0+, @RW0+RW7	Use prohibited	CMPL A, A, @										

Table B.9-7 ea Instruction 2 (First Byte = 71<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	JMPP @RL0, @RW0+d8	JMPP @ @RL0, @RW0+d8	CALLP @RL0, @RW0+d8	CALLP @ @RL0, @RW0+d8	INCL RL0, @RW0+d8	INCL RL0, @RW0+d8	DECL RL0, @RW0+d8	DECL RL0, @RW0+d8	MOVL A, RL0, @RW0+d8	MOVL A, RL0, @RW0+d8	MOVL RL0, A, @RW0+d8	MOVL RL0, A, @RW0+d8	MOV R0, #8, @RW0+d8	MOV R0, #8, @RW0+d8	MOVEA A, RL0, @RW0+d8	MOVEA A, RL0, @RW0+d8
+1	JMPP @RL0, @RW1+d8	JMPP @ @RL0, @RW1+d8	CALLP @RL0, @RW1+d8	CALLP @ @RL0, @RW1+d8	INCL RL0, @RW1+d8	INCL RL0, @RW1+d8	DECL RL0, @RW1+d8	DECL RL0, @RW1+d8	MOVL A, RL0, @RW1+d8	MOVL A, RL0, @RW1+d8	MOVL RL0, A, @RW1+d8	MOVL RL0, A, @RW1+d8	MOV R1, #8, @RW1+d8	MOV R1, #8, @RW1+d8	MOVEA A, RL1, @RW1+d8	MOVEA A, RL1, @RW1+d8
+2	JMPP @RL1, @RW2+d8	JMPP @ @RL1, @RW2+d8	CALLP @RL1, @RW2+d8	CALLP @ @RL1, @RW2+d8	INCL RL1, @RW2+d8	INCL RL1, @RW2+d8	DECL RL1, @RW2+d8	DECL RL1, @RW2+d8	MOVL A, RL1, @RW2+d8	MOVL A, RL1, @RW2+d8	MOVL RL1, A, @RW2+d8	MOVL RL1, A, @RW2+d8	MOV R2, #8, @RW2+d8	MOV R2, #8, @RW2+d8	MOVEA A, RL2, @RW2+d8	MOVEA A, RL2, @RW2+d8
+3	JMPP @RL1, @RW3+d8	JMPP @ @RL1, @RW3+d8	CALLP @RL1, @RW3+d8	CALLP @ @RL1, @RW3+d8	INCL RL1, @RW3+d8	INCL RL1, @RW3+d8	DECL RL1, @RW3+d8	DECL RL1, @RW3+d8	MOVL A, RL1, @RW3+d8	MOVL A, RL1, @RW3+d8	MOVL RL1, A, @RW3+d8	MOVL RL1, A, @RW3+d8	MOV R3, #8, @RW3+d8	MOV R3, #8, @RW3+d8	MOVEA A, RL3, @RW3+d8	MOVEA A, RL3, @RW3+d8
+4	JMPP @RL2, @RW4+d8	JMPP @ @RL2, @RW4+d8	CALLP @RL2, @RW4+d8	CALLP @ @RL2, @RW4+d8	INCL RL2, @RW4+d8	INCL RL2, @RW4+d8	DECL RL2, @RW4+d8	DECL RL2, @RW4+d8	MOVL A, RL2, @RW4+d8	MOVL A, RL2, @RW4+d8	MOVL RL2, A, @RW4+d8	MOVL RL2, A, @RW4+d8	MOV R4, #8, @RW4+d8	MOV R4, #8, @RW4+d8	MOVEA A, RL4, @RW4+d8	MOVEA A, RL4, @RW4+d8
+5	JMPP @RL2, @RW5+d8	JMPP @ @RL2, @RW5+d8	CALLP @RL2, @RW5+d8	CALLP @ @RL2, @RW5+d8	INCL RL2, @RW5+d8	INCL RL2, @RW5+d8	DECL RL2, @RW5+d8	DECL RL2, @RW5+d8	MOVL A, RL2, @RW5+d8	MOVL A, RL2, @RW5+d8	MOVL RL2, A, @RW5+d8	MOVL RL2, A, @RW5+d8	MOV R5, #8, @RW5+d8	MOV R5, #8, @RW5+d8	MOVEA A, RL5, @RW5+d8	MOVEA A, RL5, @RW5+d8
+6	JMPP @RL3, @RW6+d8	JMPP @ @RL3, @RW6+d8	CALLP @RL3, @RW6+d8	CALLP @ @RL3, @RW6+d8	INCL RL3, @RW6+d8	INCL RL3, @RW6+d8	DECL RL3, @RW6+d8	DECL RL3, @RW6+d8	MOVL A, RL3, @RW6+d8	MOVL A, RL3, @RW6+d8	MOVL RL3, A, @RW6+d8	MOVL RL3, A, @RW6+d8	MOV R6, #8, @RW6+d8	MOV R6, #8, @RW6+d8	MOVEA A, RL6, @RW6+d8	MOVEA A, RL6, @RW6+d8
+7	JMPP @RL3, @RW7+d8	JMPP @ @RL3, @RW7+d8	CALLP @RL3, @RW7+d8	CALLP @ @RL3, @RW7+d8	INCL RL3, @RW7+d8	INCL RL3, @RW7+d8	DECL RL3, @RW7+d8	DECL RL3, @RW7+d8	MOVL A, RL3, @RW7+d8	MOVL A, RL3, @RW7+d8	MOVL RL3, A, @RW7+d8	MOVL RL3, A, @RW7+d8	MOV R7, #8, @RW7+d8	MOV R7, #8, @RW7+d8	MOVEA A, RL7, @RW7+d8	MOVEA A, RL7, @RW7+d8
+8	JMPP @RW0, @RW0+d16	JMPP @ @RW0, @RW0+d16	CALLP @RW0, @RW0+d16	CALLP @ @RW0, @RW0+d16	INCL @RW0, @RW0+d16	INCL @RW0, @RW0+d16	DECL @RW0, @RW0+d16	DECL @RW0, @RW0+d16	MOVL A, @RW0, @RW0+d16	MOVL A, @RW0, @RW0+d16	MOVL @RW0, A, @RW0+d16	MOVL @RW0, A, @RW0+d16	MOV @RW0, #8, @RW0+d16	MOV @RW0, #8, @RW0+d16	MOVEA A, @RW0, @RW0+d16	MOVEA A, @RW0, @RW0+d16
+9	JMPP @RW1, @RW1+d16	JMPP @ @RW1, @RW1+d16	CALLP @RW1, @RW1+d16	CALLP @ @RW1, @RW1+d16	INCL @RW1, @RW1+d16	INCL @RW1, @RW1+d16	DECL @RW1, @RW1+d16	DECL @RW1, @RW1+d16	MOVL A, @RW1, @RW1+d16	MOVL A, @RW1, @RW1+d16	MOVL @RW1, A, @RW1+d16	MOVL @RW1, A, @RW1+d16	MOV @RW1, #8, @RW1+d16	MOV @RW1, #8, @RW1+d16	MOVEA A, @RW1, @RW1+d16	MOVEA A, @RW1, @RW1+d16
+A	JMPP @RW2, @RW2+d16	JMPP @ @RW2, @RW2+d16	CALLP @RW2, @RW2+d16	CALLP @ @RW2, @RW2+d16	INCL @RW2, @RW2+d16	INCL @RW2, @RW2+d16	DECL @RW2, @RW2+d16	DECL @RW2, @RW2+d16	MOVL A, @RW2, @RW2+d16	MOVL A, @RW2, @RW2+d16	MOVL @RW2, A, @RW2+d16	MOVL @RW2, A, @RW2+d16	MOV @RW2, #8, @RW2+d16	MOV @RW2, #8, @RW2+d16	MOVEA A, @RW2, @RW2+d16	MOVEA A, @RW2, @RW2+d16
+B	JMPP @RW3, @RW3+d16	JMPP @ @RW3, @RW3+d16	CALLP @RW3, @RW3+d16	CALLP @ @RW3, @RW3+d16	INCL @RW3, @RW3+d16	INCL @RW3, @RW3+d16	DECL @RW3, @RW3+d16	DECL @RW3, @RW3+d16	MOVL A, @RW3, @RW3+d16	MOVL A, @RW3, @RW3+d16	MOVL @RW3, A, @RW3+d16	MOVL @RW3, A, @RW3+d16	MOV @RW3, #8, @RW3+d16	MOV @RW3, #8, @RW3+d16	MOVEA A, @RW3, @RW3+d16	MOVEA A, @RW3, @RW3+d16
+C	JMPP @RW0+, @RW0+RW7	JMPP @ @RW0+, @RW0+RW7	CALLP @RW0+, @RW0+RW7	CALLP @ @RW0+, @RW0+RW7	INCL @RW0+, @RW0+RW7	INCL @RW0+, @RW0+RW7	DECL @RW0+, @RW0+RW7	DECL @RW0+, @RW0+RW7	MOVL A, @RW0+, @RW0+RW7	MOVL A, @RW0+, @RW0+RW7	MOVL @RW0+, A, @RW0+RW7	MOVL @RW0+, A, @RW0+RW7	MOV @RW0+, #8, @RW0+RW7	MOV @RW0+, #8, @RW0+RW7	MOVEA A, @RW0+, @RW0+RW7	MOVEA A, @RW0+, @RW0+RW7
+D	JMPP @RW1+, @RW1+RW7	JMPP @ @RW1+, @RW1+RW7	CALLP @RW1+, @RW1+RW7	CALLP @ @RW1+, @RW1+RW7	INCL @RW1+, @RW1+RW7	INCL @RW1+, @RW1+RW7	DECL @RW1+, @RW1+RW7	DECL @RW1+, @RW1+RW7	MOVL A, @RW1+, @RW1+RW7	MOVL A, @RW1+, @RW1+RW7	MOVL @RW1+, A, @RW1+RW7	MOVL @RW1+, A, @RW1+RW7	MOV @RW1+, #8, @RW1+RW7	MOV @RW1+, #8, @RW1+RW7	MOVEA A, @RW1+, @RW1+RW7	MOVEA A, @RW1+, @RW1+RW7
+E	JMPP @RW2+, @PC+d16	JMPP @ @RW2+, @PC+d16	CALLP @RW2+, @PC+d16	CALLP @ @RW2+, @PC+d16	INCL @RW2+, @PC+d16	INCL @RW2+, @PC+d16	DECL @RW2+, @PC+d16	DECL @RW2+, @PC+d16	MOVL A, @RW2+, @PC+d16	MOVL A, @RW2+, @PC+d16	MOVL @RW2+, A, @PC+d16	MOVL @RW2+, A, @PC+d16	MOV @RW2+, #8, @PC+d16	MOV @RW2+, #8, @PC+d16	MOVEA A, @RW2+, @PC+d16	MOVEA A, @RW2+, @PC+d16
+F	JMPP @RW3+, @addr16	JMPP @ @RW3+, @addr16	CALLP @RW3+, @addr16	CALLP @ @RW3+, @addr16	INCL @RW3+, @addr16	INCL @RW3+, @addr16	DECL @RW3+, @addr16	DECL @RW3+, @addr16	MOVL A, @RW3+, @addr16	MOVL A, @RW3+, @addr16	MOVL @RW3+, A, @addr16	MOVL @RW3+, A, @addr16	MOV @RW3+, #8, @addr16	MOV @RW3+, #8, @addr16	MOVEA A, @RW3+, @addr16	MOVEA A, @RW3+, @addr16

Table B.9-8 ea Instruction 3 (First Byte = 72<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ROL R0, @RW0+d8	ROL R0, @RW0+d8	ROR R0, @RW0+d8	ROR R0, @RW0+d8	INC R0, @RW0+d8	INC R0, @RW0+d8	DEC R0, @RW0+d8	DEC R0, @RW0+d8	MOV A, R0, @RW0+d8	MOV A, R0, @RW0+d8	MOV R0, A, @RW0+d8	MOV R0, A, @RW0+d8	MOVX A, R0, @RW0+d8	MOVX A, R0, @RW0+d8	XCH A, R0, @RW0+d8	XCH A, R0, @RW0+d8
+1	ROL R1, @RW1+d8	ROL R1, @RW1+d8	ROR R1, @RW1+d8	ROR R1, @RW1+d8	INC R1, @RW1+d8	INC R1, @RW1+d8	DEC R1, @RW1+d8	DEC R1, @RW1+d8	MOV A, R1, @RW1+d8	MOV A, R1, @RW1+d8	MOV R1, A, @RW1+d8	MOV R1, A, @RW1+d8	MOVX A, R1, @RW1+d8	MOVX A, R1, @RW1+d8	XCH A, R1, @RW1+d8	XCH A, R1, @RW1+d8
+2	ROL R2, @RW2+d8	ROL R2, @RW2+d8	ROR R2, @RW2+d8	ROR R2, @RW2+d8	INC R2, @RW2+d8	INC R2, @RW2+d8	DEC R2, @RW2+d8	DEC R2, @RW2+d8	MOV A, R2, @RW2+d8	MOV A, R2, @RW2+d8	MOV R2, A, @RW2+d8	MOV R2, A, @RW2+d8	MOVX A, R2, @RW2+d8	MOVX A, R2, @RW2+d8	XCH A, R2, @RW2+d8	XCH A, R2, @RW2+d8
+3	ROL R3, @RW3+d8	ROL R3, @RW3+d8	ROR R3, @RW3+d8	ROR R3, @RW3+d8	INC R3, @RW3+d8	INC R3, @RW3+d8	DEC R3, @RW3+d8	DEC R3, @RW3+d8	MOV A, R3, @RW3+d8	MOV A, R3, @RW3+d8	MOV R3, A, @RW3+d8	MOV R3, A, @RW3+d8	MOVX A, R3, @RW3+d8	MOVX A, R3, @RW3+d8	XCH A, R3, @RW3+d8	XCH A, R3, @RW3+d8
+4	ROL R4, @RW4+d8	ROL R4, @RW4+d8	ROR R4, @RW4+d8	ROR R4, @RW4+d8	INC R4, @RW4+d8	INC R4, @RW4+d8	DEC R4, @RW4+d8	DEC R4, @RW4+d8	MOV A, R4, @RW4+d8	MOV A, R4, @RW4+d8	MOV R4, A, @RW4+d8	MOV R4, A, @RW4+d8	MOVX A, R4, @RW4+d8	MOVX A, R4, @RW4+d8	XCH A, R4, @RW4+d8	XCH A, R4, @RW4+d8
+5	ROL R5, @RW5+d8	ROL R5, @RW5+d8	ROR R5, @RW5+d8	ROR R5, @RW5+d8	INC R5, @RW5+d8	INC R5, @RW5+d8	DEC R5, @RW5+d8	DEC R5, @RW5+d8	MOV A, R5, @RW5+d8	MOV A, R5, @RW5+d8	MOV R5, A, @RW5+d8	MOV R5, A, @RW5+d8	MOVX A, R5, @RW5+d8	MOVX A, R5, @RW5+d8	XCH A, R5, @RW5+d8	XCH A, R5, @RW5+d8
+6	ROL R6, @RW6+d8	ROL R6, @RW6+d8	ROR R6, @RW6+d8	ROR R6, @RW6+d8	INC R6, @RW6+d8	INC R6, @RW6+d8	DEC R6, @RW6+d8	DEC R6, @RW6+d8	MOV A, R6, @RW6+d8	MOV A, R6, @RW6+d8	MOV R6, A, @RW6+d8	MOV R6, A, @RW6+d8	MOVX A, R6, @RW6+d8	MOVX A, R6, @RW6+d8	XCH A, R6, @RW6+d8	XCH A, R6, @RW6+d8
+7	ROL R7, @RW7+d8	ROL R7, @RW7+d8	ROR R7, @RW7+d8	ROR R7, @RW7+d8	INC R7, @RW7+d8	INC R7, @RW7+d8	DEC R7, @RW7+d8	DEC R7, @RW7+d8	MOV A, R7, @RW7+d8	MOV A, R7, @RW7+d8	MOV R7, A, @RW7+d8	MOV R7, A, @RW7+d8	MOVX A, R7, @RW7+d8	MOVX A, R7, @RW7+d8	XCH A, R7, @RW7+d8	XCH A, R7, @RW7+d8
+8	ROL @RW0, @RW0+d16	ROL @RW0, @RW0+d16	ROR @RW0, @RW0+d16	ROR @RW0, @RW0+d16	INC @RW0, @RW0+d16	INC @RW0, @RW0+d16	DEC @RW0, @RW0+d16	DEC @RW0, @RW0+d16	MOV A, @RW0, @RW0+d16	MOV A, @RW0, @RW0+d16	MOV @RW0, A, @RW0+d16	MOV @RW0, A, @RW0+d16	MOVX A, @RW0, @RW0+d16	MOVX A, @RW0, @RW0+d16	XCH A, @RW0, @RW0+d16	XCH A, @RW0, @RW0+d16
+9	ROL @RW1, @RW1+d16	ROL @RW1, @RW1+d16	ROR @RW1, @RW1+d16	ROR @RW1, @RW1+d16	INC @RW1, @RW1+d16	INC @RW1, @RW1+d16	DEC @RW1, @RW1+d16	DEC @RW1, @RW1+d16	MOV A, @RW1, @RW1+d16	MOV A, @RW1, @RW1+d16	MOV @RW1, A, @RW1+d16	MOV @RW1, A, @RW1+d16	MOVX A, @RW1, @RW1+d16	MOVX A, @RW1, @RW1+d16	XCH A, @RW1, @RW1+d16	XCH A, @RW1, @RW1+d16
+A	ROL @RW2, @RW2+d16	ROL @RW2, @RW2+d16	ROR @RW2, @RW2+d16	ROR @RW2, @RW2+d16	INC @RW2, @RW2+d16	INC @RW2, @RW2+d16	DEC @RW2, @RW2+d16	DEC @RW2, @RW2+d16	MOV A, @RW2, @RW2+d16	MOV A, @RW2, @RW2+d16	MOV @RW2, A, @RW2+d16	MOV @RW2, A, @RW2+d16	MOVX A, @RW2, @RW2+d16	MOVX A, @RW2, @RW2+d16	XCH A, @RW2, @RW2+d16	XCH A, @RW2, @RW2+d16
+B	ROL @RW3, @RW3+d16	ROL @RW3, @RW3+d16	ROR @RW3, @RW3+d16	ROR @RW3, @RW3+d16	INC @RW3, @RW3+d16	INC @RW3, @RW3+d16	DEC @RW3, @RW3+d16	DEC @RW3, @RW3+d16	MOV A, @RW3, @RW3+d16	MOV A, @RW3, @RW3+d16	MOV @RW3, A, @RW3+d16	MOV @RW3, A, @RW3+d16	MOVX A, @RW3, @RW3+d16	MOVX A, @RW3, @RW3+d16	XCH A, @RW3, @RW3+d16	XCH A, @RW3, @RW3+d16
+C	ROL @RW0+, @RW0+RW7	ROL @RW0+, @RW0+RW7	ROR @RW0+, @RW0+RW7	ROR @RW0+, @RW0+RW7	INC @RW0+, @RW0+RW7	INC @RW0+, @RW0+RW7	DEC @RW0+, @RW0+RW7	DEC @RW0+, @RW0+RW7	MOV A, @RW0+, @RW0+RW7	MOV A, @RW0+, @RW0+RW7	MOV @RW0+, A, @RW0+RW7	MOV @RW0+, A, @RW0+RW7	MOVX A, @RW0+, @RW0+RW7	MOVX A, @RW0+, @RW0+RW7	XCH A, @RW0+, @RW0+RW7	XCH A, @RW0+, @RW0+RW7
+D	ROL @RW1+, @RW1+RW7	ROL @RW1+, @RW1+RW7	ROR @RW1+, @RW1+RW7	ROR @RW1+, @RW1+RW7	INC @RW1+, @RW1+RW7	INC @RW1+, @RW1+RW7	DEC @RW1+, @RW1+RW7	DEC @RW1+, @RW1+RW7	MOV A, @RW1+, @RW1+RW7	MOV A, @RW1+, @RW1+RW7	MOV @RW1+, A, @RW1+RW7	MOV @RW1+, A, @RW1+RW7	MOVX A, @RW1+, @RW1+RW7	MOVX A, @RW1+, @RW1+RW7	XCH A, @RW1+, @RW1+RW7	XCH A, @RW1+, @RW1+RW7
+E	ROL @RW2+, @RW2+PC+d16	ROL @RW2+, @RW2+PC+d16	ROR @RW2+, @RW2+PC+d16	ROR @RW2+, @RW2+PC+d16	INC @RW2+, @RW2+PC+d16	INC @RW2+, @RW2+PC+d16	DEC @RW2+, @RW2+PC+d16	DEC @RW2+, @RW2+PC+d16	MOV A, @RW2+, @RW2+PC+d16	MOV A, @RW2+, @RW2+PC+d16	MOV @RW2+, A, @RW2+PC+d16	MOV @RW2+, A, @RW2+PC+d16	MOVX A, @RW2+, @RW2+PC+d16	MOVX A, @RW2+, @RW2+PC+d16	XCH A, @RW2+, @RW2+PC+d16	XCH A, @RW2+, @RW2+PC+d16
+F	ROL @RW3+, @RW3+addr16	ROL @RW3+, @RW3+addr16	ROR @RW3+, @RW3+addr16	ROR @RW3+, @RW3+addr16	INC @RW3+, @RW3+addr16	INC @RW3+, @RW3+addr16	DEC @RW3+, @RW3+addr16	DEC @RW3+, @RW3+addr16	MOV A, @RW3+, @RW3+addr16	MOV A, @RW3+, @RW3+addr16	MOV @RW3+, A, @RW3+addr16	MOV @RW3+, A, @RW3+addr16	MOVX A, @RW3+, @RW3+addr16	MOVX A, @RW3+, @RW3+addr16	XCH A, @RW3+, @RW3+addr16	XCH A, @RW3+, @RW3+addr16



Table B.9-9 ea Instruction 4 (First Byte = 73<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	JMP @RW0, @RW0+d8	JMP @RW0, @RW0+d8	CALL @RW0, @RW0+d8	CALL @RW0, @RW0+d8	INCW RW0, @RW0+d8	INCW RW0, @RW0+d8	DECW RW0, @RW0+d8	DECW RW0, @RW0+d8	MOVW A, RW0, @RW0+d8	MOVW A, RW0, @RW0+d8	MOVW RW0, #16, @RW0+d8	MOVW RW0, #16, @RW0+d8	MOVW RW0, #16, @RW0+d8	MOVW RW0, #16, @RW0+d8	XCHW A, RW0, @RW0+d8	XCHW A, RW0, @RW0+d8
+1	JMP @RW1, @RW1+d8	JMP @RW1, @RW1+d8	CALL @RW1, @RW1+d8	CALL @RW1, @RW1+d8	INCW RW1, @RW1+d8	INCW RW1, @RW1+d8	DECW RW1, @RW1+d8	DECW RW1, @RW1+d8	MOVW A, RW1, @RW1+d8	MOVW A, RW1, @RW1+d8	MOVW RW1, #16, @RW1+d8	MOVW RW1, #16, @RW1+d8	MOVW RW1, #16, @RW1+d8	MOVW RW1, #16, @RW1+d8	XCHW A, RW1, @RW1+d8	XCHW A, RW1, @RW1+d8
+2	JMP @RW2, @RW2+d8	JMP @RW2, @RW2+d8	CALL @RW2, @RW2+d8	CALL @RW2, @RW2+d8	INCW RW2, @RW2+d8	INCW RW2, @RW2+d8	DECW RW2, @RW2+d8	DECW RW2, @RW2+d8	MOVW A, RW2, @RW2+d8	MOVW A, RW2, @RW2+d8	MOVW RW2, #16, @RW2+d8	MOVW RW2, #16, @RW2+d8	MOVW RW2, #16, @RW2+d8	MOVW RW2, #16, @RW2+d8	XCHW A, RW2, @RW2+d8	XCHW A, RW2, @RW2+d8
+3	JMP @RW3, @RW3+d8	JMP @RW3, @RW3+d8	CALL @RW3, @RW3+d8	CALL @RW3, @RW3+d8	INCW RW3, @RW3+d8	INCW RW3, @RW3+d8	DECW RW3, @RW3+d8	DECW RW3, @RW3+d8	MOVW A, RW3, @RW3+d8	MOVW A, RW3, @RW3+d8	MOVW RW3, #16, @RW3+d8	MOVW RW3, #16, @RW3+d8	MOVW RW3, #16, @RW3+d8	MOVW RW3, #16, @RW3+d8	XCHW A, RW3, @RW3+d8	XCHW A, RW3, @RW3+d8
+4	JMP @RW4, @RW4+d8	JMP @RW4, @RW4+d8	CALL @RW4, @RW4+d8	CALL @RW4, @RW4+d8	INCW RW4, @RW4+d8	INCW RW4, @RW4+d8	DECW RW4, @RW4+d8	DECW RW4, @RW4+d8	MOVW A, RW4, @RW4+d8	MOVW A, RW4, @RW4+d8	MOVW RW4, #16, @RW4+d8	MOVW RW4, #16, @RW4+d8	MOVW RW4, #16, @RW4+d8	MOVW RW4, #16, @RW4+d8	XCHW A, RW4, @RW4+d8	XCHW A, RW4, @RW4+d8
+5	JMP @RW5, @RW5+d8	JMP @RW5, @RW5+d8	CALL @RW5, @RW5+d8	CALL @RW5, @RW5+d8	INCW RW5, @RW5+d8	INCW RW5, @RW5+d8	DECW RW5, @RW5+d8	DECW RW5, @RW5+d8	MOVW A, RW5, @RW5+d8	MOVW A, RW5, @RW5+d8	MOVW RW5, #16, @RW5+d8	MOVW RW5, #16, @RW5+d8	MOVW RW5, #16, @RW5+d8	MOVW RW5, #16, @RW5+d8	XCHW A, RW5, @RW5+d8	XCHW A, RW5, @RW5+d8
+6	JMP @RW6, @RW6+d8	JMP @RW6, @RW6+d8	CALL @RW6, @RW6+d8	CALL @RW6, @RW6+d8	INCW RW6, @RW6+d8	INCW RW6, @RW6+d8	DECW RW6, @RW6+d8	DECW RW6, @RW6+d8	MOVW A, RW6, @RW6+d8	MOVW A, RW6, @RW6+d8	MOVW RW6, #16, @RW6+d8	MOVW RW6, #16, @RW6+d8	MOVW RW6, #16, @RW6+d8	MOVW RW6, #16, @RW6+d8	XCHW A, RW6, @RW6+d8	XCHW A, RW6, @RW6+d8
+7	JMP @RW7, @RW7+d8	JMP @RW7, @RW7+d8	CALL @RW7, @RW7+d8	CALL @RW7, @RW7+d8	INCW RW7, @RW7+d8	INCW RW7, @RW7+d8	DECW RW7, @RW7+d8	DECW RW7, @RW7+d8	MOVW A, RW7, @RW7+d8	MOVW A, RW7, @RW7+d8	MOVW RW7, #16, @RW7+d8	MOVW RW7, #16, @RW7+d8	MOVW RW7, #16, @RW7+d8	MOVW RW7, #16, @RW7+d8	XCHW A, RW7, @RW7+d8	XCHW A, RW7, @RW7+d8
+8	JMP @RW0, @RW0+d16	JMP @RW0, @RW0+d16	CALL @RW0, @RW0+d16	CALL @RW0, @RW0+d16	INCW @RW0, @RW0+d16	INCW @RW0, @RW0+d16	DECW @RW0, @RW0+d16	DECW @RW0, @RW0+d16	MOVW A, @RW0, @RW0+d16	MOVW A, @RW0, @RW0+d16	MOVW @RW0, #16, @RW0+d16	MOVW @RW0, #16, @RW0+d16	MOVW @RW0, #16, @RW0+d16	MOVW @RW0, #16, @RW0+d16	XCHW A, @RW0, @RW0+d16	XCHW A, @RW0, @RW0+d16
+9	JMP @RW1, @RW1+d16	JMP @RW1, @RW1+d16	CALL @RW1, @RW1+d16	CALL @RW1, @RW1+d16	INCW @RW1, @RW1+d16	INCW @RW1, @RW1+d16	DECW @RW1, @RW1+d16	DECW @RW1, @RW1+d16	MOVW A, @RW1, @RW1+d16	MOVW A, @RW1, @RW1+d16	MOVW @RW1, #16, @RW1+d16	MOVW @RW1, #16, @RW1+d16	MOVW @RW1, #16, @RW1+d16	MOVW @RW1, #16, @RW1+d16	XCHW A, @RW1, @RW1+d16	XCHW A, @RW1, @RW1+d16
+A	JMP @RW2, @RW2+d16	JMP @RW2, @RW2+d16	CALL @RW2, @RW2+d16	CALL @RW2, @RW2+d16	INCW @RW2, @RW2+d16	INCW @RW2, @RW2+d16	DECW @RW2, @RW2+d16	DECW @RW2, @RW2+d16	MOVW A, @RW2, @RW2+d16	MOVW A, @RW2, @RW2+d16	MOVW @RW2, #16, @RW2+d16	MOVW @RW2, #16, @RW2+d16	MOVW @RW2, #16, @RW2+d16	MOVW @RW2, #16, @RW2+d16	XCHW A, @RW2, @RW2+d16	XCHW A, @RW2, @RW2+d16
+B	JMP @RW3, @RW3+d16	JMP @RW3, @RW3+d16	CALL @RW3, @RW3+d16	CALL @RW3, @RW3+d16	INCW @RW3, @RW3+d16	INCW @RW3, @RW3+d16	DECW @RW3, @RW3+d16	DECW @RW3, @RW3+d16	MOVW A, @RW3, @RW3+d16	MOVW A, @RW3, @RW3+d16	MOVW @RW3, #16, @RW3+d16	MOVW @RW3, #16, @RW3+d16	MOVW @RW3, #16, @RW3+d16	MOVW @RW3, #16, @RW3+d16	XCHW A, @RW3, @RW3+d16	XCHW A, @RW3, @RW3+d16
+C	JMP @RW0+, @RW0+RW7	JMP @RW0+, @RW0+RW7	CALL @RW0+, @RW0+RW7	CALL @RW0+, @RW0+RW7	INCW @RW0+, @RW0+RW7	INCW @RW0+, @RW0+RW7	DECW @RW0+, @RW0+RW7	DECW @RW0+, @RW0+RW7	MOVW A, @RW0+, @RW0+RW7	MOVW A, @RW0+, @RW0+RW7	MOVW @RW0+, #16, @RW0+RW7	MOVW @RW0+, #16, @RW0+RW7	MOVW @RW0+, #16, @RW0+RW7	MOVW @RW0+, #16, @RW0+RW7	XCHW A, @RW0+, @RW0+RW7	XCHW A, @RW0+, @RW0+RW7
+D	JMP @RW1+, @RW1+RW7	JMP @RW1+, @RW1+RW7	CALL @RW1+, @RW1+RW7	CALL @RW1+, @RW1+RW7	INCW @RW1+, @RW1+RW7	INCW @RW1+, @RW1+RW7	DECW @RW1+, @RW1+RW7	DECW @RW1+, @RW1+RW7	MOVW A, @RW1+, @RW1+RW7	MOVW A, @RW1+, @RW1+RW7	MOVW @RW1+, #16, @RW1+RW7	MOVW @RW1+, #16, @RW1+RW7	MOVW @RW1+, #16, @RW1+RW7	MOVW @RW1+, #16, @RW1+RW7	XCHW A, @RW1+, @RW1+RW7	XCHW A, @RW1+, @RW1+RW7
+E	JMP @RW2+, @RW2+PC+d16	JMP @RW2+, @RW2+PC+d16	CALL @RW2+, @RW2+PC+d16	CALL @RW2+, @RW2+PC+d16	INCW @RW2+, @RW2+PC+d16	INCW @RW2+, @RW2+PC+d16	DECW @RW2+, @RW2+PC+d16	DECW @RW2+, @RW2+PC+d16	MOVW A, @RW2+, @RW2+PC+d16	MOVW A, @RW2+, @RW2+PC+d16	MOVW @RW2+, #16, @RW2+PC+d16	MOVW @RW2+, #16, @RW2+PC+d16	MOVW @RW2+, #16, @RW2+PC+d16	MOVW @RW2+, #16, @RW2+PC+d16	XCHW A, @RW2+, @RW2+PC+d16	XCHW A, @RW2+, @RW2+PC+d16
+F	JMP @RW3+, @RW3+addr16	JMP @RW3+, @RW3+addr16	CALL @RW3+, @RW3+addr16	CALL @RW3+, @RW3+addr16	INCW @RW3+, @RW3+addr16	INCW @RW3+, @RW3+addr16	DECW @RW3+, @RW3+addr16	DECW @RW3+, @RW3+addr16	MOVW A, @RW3+, @RW3+addr16	MOVW A, @RW3+, @RW3+addr16	MOVW @RW3+, #16, @RW3+addr16	MOVW @RW3+, #16, @RW3+addr16	MOVW @RW3+, #16, @RW3+addr16	MOVW @RW3+, #16, @RW3+addr16	XCHW A, @RW3+, @RW3+addr16	XCHW A, @RW3+, @RW3+addr16

Table B.9-10 ea Instruction 5 (First Byte = 74<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADD A, R0', @RW0+d8	SUB A, R0', @RW0+d8	SUB A, R0', @RW0+d8	SUB A, R0', @RW0+d8	ADDC A, R0', @RW0+d8	ADDC A, R0', @RW0+d8	CMP A, R0', @RW0+d8	CMP A, R0', @RW0+d8	AND A, R0', @RW0+d8	AND A, R0', @RW0+d8	OR A, R0', @RW0+d8	OR A, R0', @RW0+d8	XOR A, R0', @RW0+d8	XOR A, R0', @RW0+d8	DBNZ R0, rel' +d8, rel	DBNZ R0, rel' +d8, rel
+1	ADD A, R1', @RW1+d8	SUB A, R1', @RW1+d8	SUB A, R1', @RW1+d8	SUB A, R1', @RW1+d8	ADDC A, R1', @RW1+d8	ADDC A, R1', @RW1+d8	CMP A, R1', @RW1+d8	CMP A, R1', @RW1+d8	AND A, R1', @RW1+d8	AND A, R1', @RW1+d8	OR A, R1', @RW1+d8	OR A, R1', @RW1+d8	XOR A, R1', @RW1+d8	XOR A, R1', @RW1+d8	DBNZ R1, rel' +d8, rel	DBNZ R1, rel' +d8, rel
+2	ADD A, R2', @RW2+d8	SUB A, R2', @RW2+d8	SUB A, R2', @RW2+d8	SUB A, R2', @RW2+d8	ADDC A, R2', @RW2+d8	ADDC A, R2', @RW2+d8	CMP A, R2', @RW2+d8	CMP A, R2', @RW2+d8	AND A, R2', @RW2+d8	AND A, R2', @RW2+d8	OR A, R2', @RW2+d8	OR A, R2', @RW2+d8	XOR A, R2', @RW2+d8	XOR A, R2', @RW2+d8	DBNZ R2, rel' +d8, rel	DBNZ R2, rel' +d8, rel
+3	ADD A, R3', @RW3+d8	SUB A, R3', @RW3+d8	SUB A, R3', @RW3+d8	SUB A, R3', @RW3+d8	ADDC A, R3', @RW3+d8	ADDC A, R3', @RW3+d8	CMP A, R3', @RW3+d8	CMP A, R3', @RW3+d8	AND A, R3', @RW3+d8	AND A, R3', @RW3+d8	OR A, R3', @RW3+d8	OR A, R3', @RW3+d8	XOR A, R3', @RW3+d8	XOR A, R3', @RW3+d8	DBNZ R3, rel' +d8, rel	DBNZ R3, rel' +d8, rel
+4	ADD A, R4', @RW4+d8	SUB A, R4', @RW4+d8	SUB A, R4', @RW4+d8	SUB A, R4', @RW4+d8	ADDC A, R4', @RW4+d8	ADDC A, R4', @RW4+d8	CMP A, R4', @RW4+d8	CMP A, R4', @RW4+d8	AND A, R4', @RW4+d8	AND A, R4', @RW4+d8	OR A, R4', @RW4+d8	OR A, R4', @RW4+d8	XOR A, R4', @RW4+d8	XOR A, R4', @RW4+d8	DBNZ R4, rel' +d8, rel	DBNZ R4, rel' +d8, rel
+5	ADD A, R5', @RW5+d8	SUB A, R5', @RW5+d8	SUB A, R5', @RW5+d8	SUB A, R5', @RW5+d8	ADDC A, R5', @RW5+d8	ADDC A, R5', @RW5+d8	CMP A, R5', @RW5+d8	CMP A, R5', @RW5+d8	AND A, R5', @RW5+d8	AND A, R5', @RW5+d8	OR A, R5', @RW5+d8	OR A, R5', @RW5+d8	XOR A, R5', @RW5+d8	XOR A, R5', @RW5+d8	DBNZ R5, rel' +d8, rel	DBNZ R5, rel' +d8, rel
+6	ADD A, R6', @RW6+d8	SUB A, R6', @RW6+d8	SUB A, R6', @RW6+d8	SUB A, R6', @RW6+d8	ADDC A, R6', @RW6+d8	ADDC A, R6', @RW6+d8	CMP A, R6', @RW6+d8	CMP A, R6', @RW6+d8	AND A, R6', @RW6+d8	AND A, R6', @RW6+d8	OR A, R6', @RW6+d8	OR A, R6', @RW6+d8	XOR A, R6', @RW6+d8	XOR A, R6', @RW6+d8	DBNZ R6, rel' +d8, rel	DBNZ R6, rel' +d8, rel
+7	ADD A, R7', @RW7+d8	SUB A, R7', @RW7+d8	SUB A, R7', @RW7+d8	SUB A, R7', @RW7+d8	ADDC A, R7', @RW7+d8	ADDC A, R7', @RW7+d8	CMP A, R7', @RW7+d8	CMP A, R7', @RW7+d8	AND A, R7', @RW7+d8	AND A, R7', @RW7+d8	OR A, R7', @RW7+d8	OR A, R7', @RW7+d8	XOR A, R7', @RW7+d8	XOR A, R7', @RW7+d8	DBNZ R7, rel' +d8, rel	DBNZ R7, rel' +d8, rel
+8	ADD A, @RW0, @RW0+d16	SUB A, @RW0, @RW0+d16	SUB A, @RW0, @RW0+d16	SUB A, @RW0, @RW0+d16	ADDC A, @RW0, @RW0+d16	ADDC A, @RW0, @RW0+d16	CMP A, @RW0, @RW0+d16	CMP A, @RW0, @RW0+d16	AND A, @RW0, @RW0+d16	AND A, @RW0, @RW0+d16	OR A, @RW0, @RW0+d16	OR A, @RW0, @RW0+d16	XOR A, @RW0, @RW0+d16	XOR A, @RW0, @RW0+d16	DBNZ @RW0, rel' +d16, rel	DBNZ @RW0, rel' +d16, rel
+9	ADD A, @RW1, @RW1+d16	SUB A, @RW1, @RW1+d16	SUB A, @RW1, @RW1+d16	SUB A, @RW1, @RW1+d16	ADDC A, @RW1, @RW1+d16	ADDC A, @RW1, @RW1+d16	CMP A, @RW1, @RW1+d16	CMP A, @RW1, @RW1+d16	AND A, @RW1, @RW1+d16	AND A, @RW1, @RW1+d16	OR A, @RW1, @RW1+d16	OR A, @RW1, @RW1+d16	XOR A, @RW1, @RW1+d16	XOR A, @RW1, @RW1+d16	DBNZ @RW1, rel' +d16, rel	DBNZ @RW1, rel' +d16, rel
+A	ADD A, @RW2, @RW2+d16	SUB A, @RW2, @RW2+d16	SUB A, @RW2, @RW2+d16	SUB A, @RW2, @RW2+d16	ADDC A, @RW2, @RW2+d16	ADDC A, @RW2, @RW2+d16	CMP A, @RW2, @RW2+d16	CMP A, @RW2, @RW2+d16	AND A, @RW2, @RW2+d16	AND A, @RW2, @RW2+d16	OR A, @RW2, @RW2+d16	OR A, @RW2, @RW2+d16	XOR A, @RW2, @RW2+d16	XOR A, @RW2, @RW2+d16	DBNZ @RW2, rel' +d16, rel	DBNZ @RW2, rel' +d16, rel
+B	ADD A, @RW3, @RW3+d16	SUB A, @RW3, @RW3+d16	SUB A, @RW3, @RW3+d16	SUB A, @RW3, @RW3+d16	ADDC A, @RW3, @RW3+d16	ADDC A, @RW3, @RW3+d16	CMP A, @RW3, @RW3+d16	CMP A, @RW3, @RW3+d16	AND A, @RW3, @RW3+d16	AND A, @RW3, @RW3+d16	OR A, @RW3, @RW3+d16	OR A, @RW3, @RW3+d16	XOR A, @RW3, @RW3+d16	XOR A, @RW3, @RW3+d16	DBNZ @RW3, rel' +d16, rel	DBNZ @RW3, rel' +d16, rel
+C	ADD A, @RW0+, @RW0+RW7	SUB A, @RW0+, @RW0+RW7	SUB A, @RW0+, @RW0+RW7	SUB A, @RW0+, @RW0+RW7	ADDC A, @RW0+, @RW0+RW7	ADDC A, @RW0+, @RW0+RW7	CMP A, @RW0+, @RW0+RW7	CMP A, @RW0+, @RW0+RW7	AND A, @RW0+, @RW0+RW7	AND A, @RW0+, @RW0+RW7	OR A, @RW0+, @RW0+RW7	OR A, @RW0+, @RW0+RW7	XOR A, @RW0+, @RW0+RW7	XOR A, @RW0+, @RW0+RW7	DBNZ @RW0+, rel' +RW7, rel	DBNZ @RW0+, rel' +RW7, rel
+D	ADD A, @RW1+, @RW1+RW7	SUB A, @RW1+, @RW1+RW7	SUB A, @RW1+, @RW1+RW7	SUB A, @RW1+, @RW1+RW7	ADDC A, @RW1+, @RW1+RW7	ADDC A, @RW1+, @RW1+RW7	CMP A, @RW1+, @RW1+RW7	CMP A, @RW1+, @RW1+RW7	AND A, @RW1+, @RW1+RW7	AND A, @RW1+, @RW1+RW7	OR A, @RW1+, @RW1+RW7	OR A, @RW1+, @RW1+RW7	XOR A, @RW1+, @RW1+RW7	XOR A, @RW1+, @RW1+RW7	DBNZ @RW1+, rel' +RW7, rel	DBNZ @RW1+, rel' +RW7, rel
+E	ADD A, @RW2+, @PC+d16	SUB A, @RW2+, @PC+d16	SUB A, @RW2+, @PC+d16	SUB A, @RW2+, @PC+d16	ADDC A, @RW2+, @PC+d16	ADDC A, @RW2+, @PC+d16	CMP A, @RW2+, @PC+d16	CMP A, @RW2+, @PC+d16	AND A, @RW2+, @PC+d16	AND A, @RW2+, @PC+d16	OR A, @RW2+, @PC+d16	OR A, @RW2+, @PC+d16	XOR A, @RW2+, @PC+d16	XOR A, @RW2+, @PC+d16	DBNZ @RW2+, rel' +d16, rel	DBNZ @RW2+, rel' +d16, rel
+F	ADD A, @RW3+, A, addr16	SUB A, @RW3+, A, addr16	SUB A, @RW3+, A, addr16	SUB A, @RW3+, A, addr16	ADDC A, @RW3+, A, addr16	ADDC A, @RW3+, A, addr16	CMP A, @RW3+, A, addr16	CMP A, @RW3+, A, addr16	AND A, @RW3+, A, addr16	AND A, @RW3+, A, addr16	OR A, @RW3+, A, addr16	OR A, @RW3+, A, addr16	XOR A, @RW3+, A, addr16	XOR A, @RW3+, A, addr16	DBNZ @RW3+, rel' +addr16, rel	DBNZ @RW3+, rel' +addr16, rel

Table B.9-11 ea Instruction 6 (First Byte = 75<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADD R0, A, @RW0+d8, A	SUB R0, A, @RW0+d8, A	SUB R0, A, @RW0+d8, A	SUB R0, A, @RW0+d8, A	SUBC A, R0, @RW0+d8	SUBC A, R0, @RW0+d8	NEG R0, @RW0+d8	NEG R0, @RW0+d8	AND R0, A, @RW0+d8, A	AND R0, A, @RW0+d8, A	OR R0, A, @RW0+d8, A	OR R0, A, @RW0+d8, A	XOR R0, A, @RW0+d8, A	XOR R0, A, @RW0+d8, A	NOT R0, @RW0+d8	NOT R0, @RW0+d8
+1	ADD R1, A, @RW1+d8, A	SUB R1, A, @RW1+d8, A	SUB R1, A, @RW1+d8, A	SUB R1, A, @RW1+d8, A	SUBC A, R1, @RW1+d8	SUBC A, R1, @RW1+d8	NEG R1, @RW1+d8	NEG R1, @RW1+d8	AND R1, A, @RW1+d8, A	AND R1, A, @RW1+d8, A	OR R1, A, @RW1+d8, A	OR R1, A, @RW1+d8, A	XOR R1, A, @RW1+d8, A	XOR R1, A, @RW1+d8, A	NOT R1, @RW1+d8	NOT R1, @RW1+d8
+2	ADD R2, A, @RW2+d8, A	SUB R2, A, @RW2+d8, A	SUB R2, A, @RW2+d8, A	SUB R2, A, @RW2+d8, A	SUBC A, R2, @RW2+d8	SUBC A, R2, @RW2+d8	NEG R2, @RW2+d8	NEG R2, @RW2+d8	AND R2, A, @RW2+d8, A	AND R2, A, @RW2+d8, A	OR R2, A, @RW2+d8, A	OR R2, A, @RW2+d8, A	XOR R2, A, @RW2+d8, A	XOR R2, A, @RW2+d8, A	NOT R2, @RW2+d8	NOT R2, @RW2+d8
+3	ADD R3, A, @RW3+d8, A	SUB R3, A, @RW3+d8, A	SUB R3, A, @RW3+d8, A	SUB R3, A, @RW3+d8, A	SUBC A, R3, @RW3+d8	SUBC A, R3, @RW3+d8	NEG R3, @RW3+d8	NEG R3, @RW3+d8	AND R3, A, @RW3+d8, A	AND R3, A, @RW3+d8, A	OR R3, A, @RW3+d8, A	OR R3, A, @RW3+d8, A	XOR R3, A, @RW3+d8, A	XOR R3, A, @RW3+d8, A	NOT R3, @RW3+d8	NOT R3, @RW3+d8
+4	ADD R4, A, @RW4+d8, A	SUB R4, A, @RW4+d8, A	SUB R4, A, @RW4+d8, A	SUB R4, A, @RW4+d8, A	SUBC A, R4, @RW4+d8	SUBC A, R4, @RW4+d8	NEG R4, @RW4+d8	NEG R4, @RW4+d8	AND R4, A, @RW4+d8, A	AND R4, A, @RW4+d8, A	OR R4, A, @RW4+d8, A	OR R4, A, @RW4+d8, A	XOR R4, A, @RW4+d8, A	XOR R4, A, @RW4+d8, A	NOT R4, @RW4+d8	NOT R4, @RW4+d8
+5	ADD R5, A, @RW5+d8, A	SUB R5, A, @RW5+d8, A	SUB R5, A, @RW5+d8, A	SUB R5, A, @RW5+d8, A	SUBC A, R5, @RW5+d8	SUBC A, R5, @RW5+d8	NEG R5, @RW5+d8	NEG R5, @RW5+d8	AND R5, A, @RW5+d8, A	AND R5, A, @RW5+d8, A	OR R5, A, @RW5+d8, A	OR R5, A, @RW5+d8, A	XOR R5, A, @RW5+d8, A	XOR R5, A, @RW5+d8, A	NOT R5, @RW5+d8	NOT R5, @RW5+d8
+6	ADD R6, A, @RW6+d8, A	SUB R6, A, @RW6+d8, A	SUB R6, A, @RW6+d8, A	SUB R6, A, @RW6+d8, A	SUBC A, R6, @RW6+d8	SUBC A, R6, @RW6+d8	NEG R6, @RW6+d8	NEG R6, @RW6+d8	AND R6, A, @RW6+d8, A	AND R6, A, @RW6+d8, A	OR R6, A, @RW6+d8, A	OR R6, A, @RW6+d8, A	XOR R6, A, @RW6+d8, A	XOR R6, A, @RW6+d8, A	NOT R6, @RW6+d8	NOT R6, @RW6+d8
+7	ADD R7, A, @RW7+d8, A	SUB R7, A, @RW7+d8, A	SUB R7, A, @RW7+d8, A	SUB R7, A, @RW7+d8, A	SUBC A, R7, @RW7+d8	SUBC A, R7, @RW7+d8	NEG R7, @RW7+d8	NEG R7, @RW7+d8	AND R7, A, @RW7+d8, A	AND R7, A, @RW7+d8, A	OR R7, A, @RW7+d8, A	OR R7, A, @RW7+d8, A	XOR R7, A, @RW7+d8, A	XOR R7, A, @RW7+d8, A	NOT R7, @RW7+d8	NOT R7, @RW7+d8
+8	ADD @RW0, A, @RW0+d16, A	SUB @RW0, A, @RW0+d16, A	SUB @RW0, A, @RW0+d16, A	SUB @RW0, A, @RW0+d16, A	SUBC A, @RW0, @RW0+d16	SUBC A, @RW0, @RW0+d16	NEG @RW0, @RW0+d16	NEG @RW0, @RW0+d16	AND @RW0, A, @RW0+d16, A	AND @RW0, A, @RW0+d16, A	OR @RW0, A, @RW0+d16, A	OR @RW0, A, @RW0+d16, A	XOR @RW0, A, @RW0+d16, A	XOR @RW0, A, @RW0+d16, A	NOT @RW0, @RW0+d16	NOT @RW0, @RW0+d16
+9	ADD @RW1, A, @RW1+d16, A	SUB @RW1, A, @RW1+d16, A	SUB @RW1, A, @RW1+d16, A	SUB @RW1, A, @RW1+d16, A	SUBC A, @RW1, @RW1+d16	SUBC A, @RW1, @RW1+d16	NEG @RW1, @RW1+d16	NEG @RW1, @RW1+d16	AND @RW1, A, @RW1+d16, A	AND @RW1, A, @RW1+d16, A	OR @RW1, A, @RW1+d16, A	OR @RW1, A, @RW1+d16, A	XOR @RW1, A, @RW1+d16, A	XOR @RW1, A, @RW1+d16, A	NOT @RW1, @RW1+d16	NOT @RW1, @RW1+d16
+A	ADD @RW2, A, @RW2+d16, A	SUB @RW2, A, @RW2+d16, A	SUB @RW2, A, @RW2+d16, A	SUB @RW2, A, @RW2+d16, A	SUBC A, @RW2, @RW2+d16	SUBC A, @RW2, @RW2+d16	NEG @RW2, @RW2+d16	NEG @RW2, @RW2+d16	AND @RW2, A, @RW2+d16, A	AND @RW2, A, @RW2+d16, A	OR @RW2, A, @RW2+d16, A	OR @RW2, A, @RW2+d16, A	XOR @RW2, A, @RW2+d16, A	XOR @RW2, A, @RW2+d16, A	NOT @RW2, @RW2+d16	NOT @RW2, @RW2+d16
+B	ADD @RW3, A, @RW3+d16, A	SUB @RW3, A, @RW3+d16, A	SUB @RW3, A, @RW3+d16, A	SUB @RW3, A, @RW3+d16, A	SUBC A, @RW3, @RW3+d16	SUBC A, @RW3, @RW3+d16	NEG @RW3, @RW3+d16	NEG @RW3, @RW3+d16	AND @RW3, A, @RW3+d16, A	AND @RW3, A, @RW3+d16, A	OR @RW3, A, @RW3+d16, A	OR @RW3, A, @RW3+d16, A	XOR @RW3, A, @RW3+d16, A	XOR @RW3, A, @RW3+d16, A	NOT @RW3, @RW3+d16	NOT @RW3, @RW3+d16
+C	ADD @RW0+, A, @RW0+RW7, A	SUB @RW0+, A, @RW0+RW7, A	SUB @RW0+, A, @RW0+RW7, A	SUB @RW0+, A, @RW0+RW7, A	SUBC A, @RW0+, @RW0+RW7	SUBC A, @RW0+, @RW0+RW7	NEG @RW0+, @RW0+RW7	NEG @RW0+, @RW0+RW7	AND @RW0+, A, @RW0+RW7, A	AND @RW0+, A, @RW0+RW7, A	OR @RW0+, A, @RW0+RW7, A	OR @RW0+, A, @RW0+RW7, A	XOR @RW0+, A, @RW0+RW7, A	XOR @RW0+, A, @RW0+RW7, A	NOT @RW0+, @RW0+RW7	NOT @RW0+, @RW0+RW7
+D	ADD @RW1+, A, @RW1+RW7, A	SUB @RW1+, A, @RW1+RW7, A	SUB @RW1+, A, @RW1+RW7, A	SUB @RW1+, A, @RW1+RW7, A	SUBC A, @RW1+, @RW1+RW7	SUBC A, @RW1+, @RW1+RW7	NEG @RW1+, @RW1+RW7	NEG @RW1+, @RW1+RW7	AND @RW1+, A, @RW1+RW7, A	AND @RW1+, A, @RW1+RW7, A	OR @RW1+, A, @RW1+RW7, A	OR @RW1+, A, @RW1+RW7, A	XOR @RW1+, A, @RW1+RW7, A	XOR @RW1+, A, @RW1+RW7, A	NOT @RW1+, @RW1+RW7	NOT @RW1+, @RW1+RW7
+E	ADD @RW2+, A, @PC+d16, A	SUB @RW2+, A, @PC+d16, A	SUB @RW2+, A, @PC+d16, A	SUB @RW2+, A, @PC+d16, A	SUBC A, @RW2+, @PC+d16	SUBC A, @RW2+, @PC+d16	NEG @RW2+, @PC+d16	NEG @RW2+, @PC+d16	AND @RW2+, A, @PC+d16, A	AND @RW2+, A, @PC+d16, A	OR @RW2+, A, @PC+d16, A	OR @RW2+, A, @PC+d16, A	XOR @RW2+, A, @PC+d16, A	XOR @RW2+, A, @PC+d16, A	NOT @RW2+, @PC+d16	NOT @RW2+, @PC+d16
+F	ADD @RW3+, A, addr16, A	SUB @RW3+, A, addr16, A	SUB @RW3+, A, addr16, A	SUB @RW3+, A, addr16, A	SUBC A, @RW3+, addr16	SUBC A, @RW3+, addr16	NEG @RW3+, addr16	NEG @RW3+, addr16	AND @RW3+, A, addr16, A	AND @RW3+, A, addr16, A	OR @RW3+, A, addr16, A	OR @RW3+, A, addr16, A	XOR @RW3+, A, addr16, A	XOR @RW3+, A, addr16, A	NOT @RW3+, addr16	NOT @RW3+, addr16

Table B.9-12 ea Instruction 7 (First Byte = 76<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADDW A, RW0, @RW0+d8	ADDW A, RW0, @RW0+d8	SUBW A, RW0, @RW0+d8	SUBW A, RW0, @RW0+d8	ADDCW A, RW0, @RW0+d8	ADDCW A, RW0, @RW0+d8	CMPW A, RW0, @RW0+d8	CMPW A, RW0, @RW0+d8	ANDW A, RW0, @RW0+d8	ANDW A, RW0, @RW0+d8	ORW A, RW0, @RW0+d8	ORW A, RW0, @RW0+d8	XORW A, RW0, @RW0+d8	XORW A, RW0, @RW0+d8	DWBNZ @RW0, rel	DWBNZ @RW0, rel
+1	ADDW A, RW1, @RW1+d8	ADDW A, RW1, @RW1+d8	SUBW A, RW1, @RW1+d8	SUBW A, RW1, @RW1+d8	ADDCW A, RW1, @RW1+d8	ADDCW A, RW1, @RW1+d8	CMPW A, RW1, @RW1+d8	CMPW A, RW1, @RW1+d8	ANDW A, RW1, @RW1+d8	ANDW A, RW1, @RW1+d8	ORW A, RW1, @RW1+d8	ORW A, RW1, @RW1+d8	XORW A, RW1, @RW1+d8	XORW A, RW1, @RW1+d8	DWBNZ @RW1, rel	DWBNZ @RW1, rel
+2	ADDW A, RW2, @RW2+d8	ADDW A, RW2, @RW2+d8	SUBW A, RW2, @RW2+d8	SUBW A, RW2, @RW2+d8	ADDCW A, RW2, @RW2+d8	ADDCW A, RW2, @RW2+d8	CMPW A, RW2, @RW2+d8	CMPW A, RW2, @RW2+d8	ANDW A, RW2, @RW2+d8	ANDW A, RW2, @RW2+d8	ORW A, RW2, @RW2+d8	ORW A, RW2, @RW2+d8	XORW A, RW2, @RW2+d8	XORW A, RW2, @RW2+d8	DWBNZ @RW2, rel	DWBNZ @RW2, rel
+3	ADDW A, RW3, @RW3+d8	ADDW A, RW3, @RW3+d8	SUBW A, RW3, @RW3+d8	SUBW A, RW3, @RW3+d8	ADDCW A, RW3, @RW3+d8	ADDCW A, RW3, @RW3+d8	CMPW A, RW3, @RW3+d8	CMPW A, RW3, @RW3+d8	ANDW A, RW3, @RW3+d8	ANDW A, RW3, @RW3+d8	ORW A, RW3, @RW3+d8	ORW A, RW3, @RW3+d8	XORW A, RW3, @RW3+d8	XORW A, RW3, @RW3+d8	DWBNZ @RW3, rel	DWBNZ @RW3, rel
+4	ADDW A, RW4, @RW4+d8	ADDW A, RW4, @RW4+d8	SUBW A, RW4, @RW4+d8	SUBW A, RW4, @RW4+d8	ADDCW A, RW4, @RW4+d8	ADDCW A, RW4, @RW4+d8	CMPW A, RW4, @RW4+d8	CMPW A, RW4, @RW4+d8	ANDW A, RW4, @RW4+d8	ANDW A, RW4, @RW4+d8	ORW A, RW4, @RW4+d8	ORW A, RW4, @RW4+d8	XORW A, RW4, @RW4+d8	XORW A, RW4, @RW4+d8	DWBNZ @RW4, rel	DWBNZ @RW4, rel
+5	ADDW A, RW5, @RW5+d8	ADDW A, RW5, @RW5+d8	SUBW A, RW5, @RW5+d8	SUBW A, RW5, @RW5+d8	ADDCW A, RW5, @RW5+d8	ADDCW A, RW5, @RW5+d8	CMPW A, RW5, @RW5+d8	CMPW A, RW5, @RW5+d8	ANDW A, RW5, @RW5+d8	ANDW A, RW5, @RW5+d8	ORW A, RW5, @RW5+d8	ORW A, RW5, @RW5+d8	XORW A, RW5, @RW5+d8	XORW A, RW5, @RW5+d8	DWBNZ @RW5, rel	DWBNZ @RW5, rel
+6	ADDW A, RW6, @RW6+d8	ADDW A, RW6, @RW6+d8	SUBW A, RW6, @RW6+d8	SUBW A, RW6, @RW6+d8	ADDCW A, RW6, @RW6+d8	ADDCW A, RW6, @RW6+d8	CMPW A, RW6, @RW6+d8	CMPW A, RW6, @RW6+d8	ANDW A, RW6, @RW6+d8	ANDW A, RW6, @RW6+d8	ORW A, RW6, @RW6+d8	ORW A, RW6, @RW6+d8	XORW A, RW6, @RW6+d8	XORW A, RW6, @RW6+d8	DWBNZ @RW6, rel	DWBNZ @RW6, rel
+7	ADDW A, RW7, @RW7+d8	ADDW A, RW7, @RW7+d8	SUBW A, RW7, @RW7+d8	SUBW A, RW7, @RW7+d8	ADDCW A, RW7, @RW7+d8	ADDCW A, RW7, @RW7+d8	CMPW A, RW7, @RW7+d8	CMPW A, RW7, @RW7+d8	ANDW A, RW7, @RW7+d8	ANDW A, RW7, @RW7+d8	ORW A, RW7, @RW7+d8	ORW A, RW7, @RW7+d8	XORW A, RW7, @RW7+d8	XORW A, RW7, @RW7+d8	DWBNZ @RW7, rel	DWBNZ @RW7, rel
+8	ADDW A, @RW0, @RW0+d16	ADDW A, @RW0, @RW0+d16	SUBW A, @RW0, @RW0+d16	SUBW A, @RW0, @RW0+d16	ADDCW A, @RW0, @RW0+d16	ADDCW A, @RW0, @RW0+d16	CMPW A, @RW0, @RW0+d16	CMPW A, @RW0, @RW0+d16	ANDW A, @RW0, @RW0+d16	ANDW A, @RW0, @RW0+d16	ORW A, @RW0, @RW0+d16	ORW A, @RW0, @RW0+d16	XORW A, @RW0, @RW0+d16	XORW A, @RW0, @RW0+d16	DWBNZ @RW0, rel	DWBNZ @RW0, rel
+9	ADDW A, @RW1, @RW1+d16	ADDW A, @RW1, @RW1+d16	SUBW A, @RW1, @RW1+d16	SUBW A, @RW1, @RW1+d16	ADDCW A, @RW1, @RW1+d16	ADDCW A, @RW1, @RW1+d16	CMPW A, @RW1, @RW1+d16	CMPW A, @RW1, @RW1+d16	ANDW A, @RW1, @RW1+d16	ANDW A, @RW1, @RW1+d16	ORW A, @RW1, @RW1+d16	ORW A, @RW1, @RW1+d16	XORW A, @RW1, @RW1+d16	XORW A, @RW1, @RW1+d16	DWBNZ @RW1, rel	DWBNZ @RW1, rel
+A	ADDW A, @RW2, @RW2+d16	ADDW A, @RW2, @RW2+d16	SUBW A, @RW2, @RW2+d16	SUBW A, @RW2, @RW2+d16	ADDCW A, @RW2, @RW2+d16	ADDCW A, @RW2, @RW2+d16	CMPW A, @RW2, @RW2+d16	CMPW A, @RW2, @RW2+d16	ANDW A, @RW2, @RW2+d16	ANDW A, @RW2, @RW2+d16	ORW A, @RW2, @RW2+d16	ORW A, @RW2, @RW2+d16	XORW A, @RW2, @RW2+d16	XORW A, @RW2, @RW2+d16	DWBNZ @RW2, rel	DWBNZ @RW2, rel
+B	ADDW A, @RW3, @RW3+d16	ADDW A, @RW3, @RW3+d16	SUBW A, @RW3, @RW3+d16	SUBW A, @RW3, @RW3+d16	ADDCW A, @RW3, @RW3+d16	ADDCW A, @RW3, @RW3+d16	CMPW A, @RW3, @RW3+d16	CMPW A, @RW3, @RW3+d16	ANDW A, @RW3, @RW3+d16	ANDW A, @RW3, @RW3+d16	ORW A, @RW3, @RW3+d16	ORW A, @RW3, @RW3+d16	XORW A, @RW3, @RW3+d16	XORW A, @RW3, @RW3+d16	DWBNZ @RW3, rel	DWBNZ @RW3, rel
+C	ADDW A, @RW0+, @RW0+RW7	ADDW A, @RW0+, @RW0+RW7	SUBW A, @RW0+, @RW0+RW7	SUBW A, @RW0+, @RW0+RW7	ADDCW A, @RW0+, @RW0+RW7	ADDCW A, @RW0+, @RW0+RW7	CMPW A, @RW0+, @RW0+RW7	CMPW A, @RW0+, @RW0+RW7	ANDW A, @RW0+, @RW0+RW7	ANDW A, @RW0+, @RW0+RW7	ORW A, @RW0+, @RW0+RW7	ORW A, @RW0+, @RW0+RW7	XORW A, @RW0+, @RW0+RW7	XORW A, @RW0+, @RW0+RW7	DWBNZ @RW0+, rel	DWBNZ @RW0+, rel
+D	ADDW A, @RW1+, @RW1+RW7	ADDW A, @RW1+, @RW1+RW7	SUBW A, @RW1+, @RW1+RW7	SUBW A, @RW1+, @RW1+RW7	ADDCW A, @RW1+, @RW1+RW7	ADDCW A, @RW1+, @RW1+RW7	CMPW A, @RW1+, @RW1+RW7	CMPW A, @RW1+, @RW1+RW7	ANDW A, @RW1+, @RW1+RW7	ANDW A, @RW1+, @RW1+RW7	ORW A, @RW1+, @RW1+RW7	ORW A, @RW1+, @RW1+RW7	XORW A, @RW1+, @RW1+RW7	XORW A, @RW1+, @RW1+RW7	DWBNZ @RW1+, rel	DWBNZ @RW1+, rel
+E	ADDW A, @RW2+, @PC+d16	ADDW A, @RW2+, @PC+d16	SUBW A, @RW2+, @PC+d16	SUBW A, @RW2+, @PC+d16	ADDCW A, @RW2+, @PC+d16	ADDCW A, @RW2+, @PC+d16	CMPW A, @RW2+, @PC+d16	CMPW A, @RW2+, @PC+d16	ANDW A, @RW2+, @PC+d16	ANDW A, @RW2+, @PC+d16	ORW A, @RW2+, @PC+d16	ORW A, @RW2+, @PC+d16	XORW A, @RW2+, @PC+d16	XORW A, @RW2+, @PC+d16	DWBNZ @RW2+, rel	DWBNZ @RW2+, rel
+F	ADDW A, @RW3+, addr16	ADDW A, @RW3+, addr16	SUBW A, @RW3+, addr16	SUBW A, @RW3+, addr16	ADDCW A, @RW3+, addr16	ADDCW A, @RW3+, addr16	CMPW A, @RW3+, addr16	CMPW A, @RW3+, addr16	ANDW A, @RW3+, addr16	ANDW A, @RW3+, addr16	ORW A, @RW3+, addr16	ORW A, @RW3+, addr16	XORW A, @RW3+, addr16	XORW A, @RW3+, addr16	DWBNZ @RW3+, rel	DWBNZ @RW3+, rel

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADDW RW0, A' @RW0+d8, A	SUBW RW0, A' @RW0+d8, A	SUBW RW0, A' @RW0+d8, A	SUBCW SUBCW A, A, RW0' @RW0+d8	SUBCW SUBCW A, A, RW0' @RW0+d8	NEGW RW0' @RW0+d8	NEGW RW0' @RW0+d8	NEGW RW0' @RW0+d8	ANDW RW0, A' @RW0+d8, A	ANDW RW0, A' @RW0+d8, A	ORW RW0, A' @RW0+d8, A	ORW RW0, A' @RW0+d8, A	XORW RW0, A' @RW0+d8, A	XORW RW0, A' @RW0+d8, A	NOTW RW0' @RW0+d8	NOTW RW0' @RW0+d8
+1	ADDW RW1, A' @RW1+d8, A	SUBW RW1, A' @RW1+d8, A	SUBW RW1, A' @RW1+d8, A	SUBCW SUBCW A, A, RW1' @RW1+d8	SUBCW SUBCW A, A, RW1' @RW1+d8	NEGW RW1' @RW1+d8	NEGW RW1' @RW1+d8	NEGW RW1' @RW1+d8	ANDW RW1, A' @RW1+d8, A	ANDW RW1, A' @RW1+d8, A	ORW RW1, A' @RW1+d8, A	ORW RW1, A' @RW1+d8, A	XORW RW1, A' @RW1+d8, A	XORW RW1, A' @RW1+d8, A	NOTW RW1' @RW1+d8	NOTW RW1' @RW1+d8
+2	ADDW RW2, A' @RW2+d8, A	SUBW RW2, A' @RW2+d8, A	SUBW RW2, A' @RW2+d8, A	SUBCW SUBCW A, A, RW2' @RW2+d8	SUBCW SUBCW A, A, RW2' @RW2+d8	NEGW RW2' @RW2+d8	NEGW RW2' @RW2+d8	NEGW RW2' @RW2+d8	ANDW RW2, A' @RW2+d8, A	ANDW RW2, A' @RW2+d8, A	ORW RW2, A' @RW2+d8, A	ORW RW2, A' @RW2+d8, A	XORW RW2, A' @RW2+d8, A	XORW RW2, A' @RW2+d8, A	NOTW RW2' @RW2+d8	NOTW RW2' @RW2+d8
+3	ADDW RW3, A' @RW3+d8, A	SUBW RW3, A' @RW3+d8, A	SUBW RW3, A' @RW3+d8, A	SUBCW SUBCW A, A, RW3' @RW3+d8	SUBCW SUBCW A, A, RW3' @RW3+d8	NEGW RW3' @RW3+d8	NEGW RW3' @RW3+d8	NEGW RW3' @RW3+d8	ANDW RW3, A' @RW3+d8, A	ANDW RW3, A' @RW3+d8, A	ORW RW3, A' @RW3+d8, A	ORW RW3, A' @RW3+d8, A	XORW RW3, A' @RW3+d8, A	XORW RW3, A' @RW3+d8, A	NOTW RW3' @RW3+d8	NOTW RW3' @RW3+d8
+4	ADDW RW4, A' @RW4+d8, A	SUBW RW4, A' @RW4+d8, A	SUBW RW4, A' @RW4+d8, A	SUBCW SUBCW A, A, RW4' @RW4+d8	SUBCW SUBCW A, A, RW4' @RW4+d8	NEGW RW4' @RW4+d8	NEGW RW4' @RW4+d8	NEGW RW4' @RW4+d8	ANDW RW4, A' @RW4+d8, A	ANDW RW4, A' @RW4+d8, A	ORW RW4, A' @RW4+d8, A	ORW RW4, A' @RW4+d8, A	XORW RW4, A' @RW4+d8, A	XORW RW4, A' @RW4+d8, A	NOTW RW4' @RW4+d8	NOTW RW4' @RW4+d8
+5	ADDW RW5, A' @RW5+d8, A	SUBW RW5, A' @RW5+d8, A	SUBW RW5, A' @RW5+d8, A	SUBCW SUBCW A, A, RW5' @RW5+d8	SUBCW SUBCW A, A, RW5' @RW5+d8	NEGW RW5' @RW5+d8	NEGW RW5' @RW5+d8	NEGW RW5' @RW5+d8	ANDW RW5, A' @RW5+d8, A	ANDW RW5, A' @RW5+d8, A	ORW RW5, A' @RW5+d8, A	ORW RW5, A' @RW5+d8, A	XORW RW5, A' @RW5+d8, A	XORW RW5, A' @RW5+d8, A	NOTW RW5' @RW5+d8	NOTW RW5' @RW5+d8
+6	ADDW RW6, A' @RW6+d8, A	SUBW RW6, A' @RW6+d8, A	SUBW RW6, A' @RW6+d8, A	SUBCW SUBCW A, A, RW6' @RW6+d8	SUBCW SUBCW A, A, RW6' @RW6+d8	NEGW RW6' @RW6+d8	NEGW RW6' @RW6+d8	NEGW RW6' @RW6+d8	ANDW RW6, A' @RW6+d8, A	ANDW RW6, A' @RW6+d8, A	ORW RW6, A' @RW6+d8, A	ORW RW6, A' @RW6+d8, A	XORW RW6, A' @RW6+d8, A	XORW RW6, A' @RW6+d8, A	NOTW RW6' @RW6+d8	NOTW RW6' @RW6+d8
+7	ADDW RW7, A' @RW7+d8, A	SUBW RW7, A' @RW7+d8, A	SUBW RW7, A' @RW7+d8, A	SUBCW SUBCW A, A, RW7' @RW7+d8	SUBCW SUBCW A, A, RW7' @RW7+d8	NEGW RW7' @RW7+d8	NEGW RW7' @RW7+d8	NEGW RW7' @RW7+d8	ANDW RW7, A' @RW7+d8, A	ANDW RW7, A' @RW7+d8, A	ORW RW7, A' @RW7+d8, A	ORW RW7, A' @RW7+d8, A	XORW RW7, A' @RW7+d8, A	XORW RW7, A' @RW7+d8, A	NOTW RW7' @RW7+d8	NOTW RW7' @RW7+d8
+8	ADDW @RW0, A' @RW0+d16, A	SUBW @RW0, A' @RW0+d16, A	SUBW @RW0, A' @RW0+d16, A	SUBCW SUBCW A, A, @RW0' @RW0+d16	SUBCW SUBCW A, A, @RW0' @RW0+d16	NEGW @RW0, @RW0+d16	NEGW @RW0, @RW0+d16	NEGW @RW0, @RW0+d16	ANDW @RW0, A' @RW0+d16, A	ANDW @RW0, A' @RW0+d16, A	ORW @RW0, A' @RW0+d16, A	ORW @RW0, A' @RW0+d16, A	XORW @RW0, A' @RW0+d16, A	XORW @RW0, A' @RW0+d16, A	NOTW @RW0' @RW0+d16	NOTW @RW0' @RW0+d16
+9	ADDW @RW1, A' @RW1+d16, A	SUBW @RW1, A' @RW1+d16, A	SUBW @RW1, A' @RW1+d16, A	SUBCW SUBCW A, A, @RW1' @RW1+d16	SUBCW SUBCW A, A, @RW1' @RW1+d16	NEGW @RW1, @RW1+d16	NEGW @RW1, @RW1+d16	NEGW @RW1, @RW1+d16	ANDW @RW1, A' @RW1+d16, A	ANDW @RW1, A' @RW1+d16, A	ORW @RW1, A' @RW1+d16, A	ORW @RW1, A' @RW1+d16, A	XORW @RW1, A' @RW1+d16, A	XORW @RW1, A' @RW1+d16, A	NOTW @RW1' @RW1+d16	NOTW @RW1' @RW1+d16
+A	ADDW @RW2, A' @RW2+d16, A	SUBW @RW2, A' @RW2+d16, A	SUBW @RW2, A' @RW2+d16, A	SUBCW SUBCW A, A, @RW2' @RW2+d16	SUBCW SUBCW A, A, @RW2' @RW2+d16	NEGW @RW2, @RW2+d16	NEGW @RW2, @RW2+d16	NEGW @RW2, @RW2+d16	ANDW @RW2, A' @RW2+d16, A	ANDW @RW2, A' @RW2+d16, A	ORW @RW2, A' @RW2+d16, A	ORW @RW2, A' @RW2+d16, A	XORW @RW2, A' @RW2+d16, A	XORW @RW2, A' @RW2+d16, A	NOTW @RW2' @RW2+d16	NOTW @RW2' @RW2+d16
+B	ADDW @RW3, A' @RW3+d16, A	SUBW @RW3, A' @RW3+d16, A	SUBW @RW3, A' @RW3+d16, A	SUBCW SUBCW A, A, @RW3' @RW3+d16	SUBCW SUBCW A, A, @RW3' @RW3+d16	NEGW @RW3, @RW3+d16	NEGW @RW3, @RW3+d16	NEGW @RW3, @RW3+d16	ANDW @RW3, A' @RW3+d16, A							

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MULU A, R0' @RW0+d8	MULU A, R0' @RW0+d8	MULUW A, RW0' @RW0+d8	MULUW A, RW0' @RW0+d8	MUL A, R0' @RW0+d8	MULW A, RW0' @RW0+d8	MULW A, RW0' @RW0+d8	MULW A, RW0' @RW0+d8	DIVU A, R0' @RW0+d8	DIVU A, R0' @RW0+d8	DIVUW A, RW0' @RW0+d8	DIVUW A, RW0' @RW0+d8	DIV A, R0' @RW0+d8	DIV A, R0' @RW0+d8	DIVW A, RW0' @RW0+d8	DIVW A, RW0' @RW0+d8
+1	MULU A, R1' @RW1+d8	MULU A, R1' @RW1+d8	MULUW A, RW1' @RW1+d8	MULUW A, RW1' @RW1+d8	MUL A, R1' @RW1+d8	MULW A, RW1' @RW1+d8	MULW A, RW1' @RW1+d8	MULW A, RW1' @RW1+d8	DIVU A, R1' @RW1+d8	DIVU A, R1' @RW1+d8	DIVUW A, RW1' @RW1+d8	DIVUW A, RW1' @RW1+d8	DIV A, R1' @RW1+d8	DIV A, R1' @RW1+d8	DIVW A, RW1' @RW1+d8	DIVW A, RW1' @RW1+d8
+2	MULU A, R2' @RW2+d8	MULU A, R2' @RW2+d8	MULUW A, RW2' @RW2+d8	MULUW A, RW2' @RW2+d8	MUL A, R2' @RW2+d8	MULW A, RW2' @RW2+d8	MULW A, RW2' @RW2+d8	MULW A, RW2' @RW2+d8	DIVU A, R2' @RW2+d8	DIVU A, R2' @RW2+d8	DIVUW A, RW2' @RW2+d8	DIVUW A, RW2' @RW2+d8	DIV A, R2' @RW2+d8	DIV A, R2' @RW2+d8	DIVW A, RW2' @RW2+d8	DIVW A, RW2' @RW2+d8
+3	MULU A, R3' @RW3+d8	MULU A, R3' @RW3+d8	MULUW A, RW3' @RW3+d8	MULUW A, RW3' @RW3+d8	MUL A, R3' @RW3+d8	MULW A, RW3' @RW3+d8	MULW A, RW3' @RW3+d8	MULW A, RW3' @RW3+d8	DIVU A, R3' @RW3+d8	DIVU A, R3' @RW3+d8	DIVUW A, RW3' @RW3+d8	DIVUW A, RW3' @RW3+d8	DIV A, R3' @RW3+d8	DIV A, R3' @RW3+d8	DIVW A, RW3' @RW3+d8	DIVW A, RW3' @RW3+d8
+4	MULU A, R4' @RW4+d8	MULU A, R4' @RW4+d8	MULUW A, RW4' @RW4+d8	MULUW A, RW4' @RW4+d8	MUL A, R4' @RW4+d8	MULW A, RW4' @RW4+d8	MULW A, RW4' @RW4+d8	MULW A, RW4' @RW4+d8	DIVU A, R4' @RW4+d8	DIVU A, R4' @RW4+d8	DIVUW A, RW4' @RW4+d8	DIVUW A, RW4' @RW4+d8	DIV A, R4' @RW4+d8	DIV A, R4' @RW4+d8	DIVW A, RW4' @RW4+d8	DIVW A, RW4' @RW4+d8
+5	MULU A, R5' @RW5+d8	MULU A, R5' @RW5+d8	MULUW A, RW5' @RW5+d8	MULUW A, RW5' @RW5+d8	MUL A, R5' @RW5+d8	MULW A, RW5' @RW5+d8	MULW A, RW5' @RW5+d8	MULW A, RW5' @RW5+d8	DIVU A, R5' @RW5+d8	DIVU A, R5' @RW5+d8	DIVUW A, RW5' @RW5+d8	DIVUW A, RW5' @RW5+d8	DIV A, R5' @RW5+d8	DIV A, R5' @RW5+d8	DIVW A, RW5' @RW5+d8	DIVW A, RW5' @RW5+d8
+6	MULU A, R6' @RW6+d8	MULU A, R6' @RW6+d8	MULUW A, RW6' @RW6+d8	MULUW A, RW6' @RW6+d8	MUL A, R6' @RW6+d8	MULW A, RW6' @RW6+d8	MULW A, RW6' @RW6+d8	MULW A, RW6' @RW6+d8	DIVU A, R6' @RW6+d8	DIVU A, R6' @RW6+d8	DIVUW A, RW6' @RW6+d8	DIVUW A, RW6' @RW6+d8	DIV A, R6' @RW6+d8	DIV A, R6' @RW6+d8	DIVW A, RW6' @RW6+d8	DIVW A, RW6' @RW6+d8
+7	MULU A, R7' @RW7+d8	MULU A, R7' @RW7+d8	MULUW A, RW7' @RW7+d8	MULUW A, RW7' @RW7+d8	MUL A, R7' @RW7+d8	MULW A, RW7' @RW7+d8	MULW A, RW7' @RW7+d8	MULW A, RW7' @RW7+d8	DIVU A, R7' @RW7+d8	DIVU A, R7' @RW7+d8	DIVUW A, RW7' @RW7+d8	DIVUW A, RW7' @RW7+d8	DIV A, R7' @RW7+d8	DIV A, R7' @RW7+d8	DIVW A, RW7' @RW7+d8	DIVW A, RW7' @RW7+d8
+8	MULU A, @RW0, @RW0+d16	MULU A, @RW0, @RW0+d16	MULUW A, @RW0, @RW0+d16	MULUW A, @RW0, @RW0+d16	MUL A, @RW0' @RW0+d16	MULW A, @RW0, @RW0+d16	MULW A, @RW0, @RW0+d16	MULW A, @RW0, @RW0+d16	DIVU A, @RW0, @RW0+d16	DIVU A, @RW0, @RW0+d16	DIVUW A, @RW0, @RW0+d16	DIVUW A, @RW0, @RW0+d16	DIV A, @RW0, @RW0+d16	DIV A, @RW0, @RW0+d16	DIVW A, @RW0, @RW0+d16	DIVW A, @RW0, @RW0+d16
+9	MULU A, @RW1, @RW1+d16	MULU A, @RW1, @RW1+d16	MULUW A, @RW1, @RW1+d16	MULUW A, @RW1, @RW1+d16	MUL A, @RW1' @RW1+d16	MULW A, @RW1, @RW1+d16	MULW A, @RW1, @RW1+d16	MULW A, @RW1, @RW1+d16	DIVU A, @RW1, @RW1+d16	DIVU A, @RW1, @RW1+d16	DIVUW A, @RW1, @RW1+d16	DIVUW A, @RW1, @RW1+d16	DIV A, @RW1, @RW1+d16	DIV A, @RW1, @RW1+d16	DIVW A, @RW1, @RW1+d16	DIVW A, @RW1, @RW1+d16
+A	MULU A, @RW2, @RW2+d16	MULU A, @RW2, @RW2+d16	MULUW A, @RW2, @RW2+d16	MULUW A, @RW2, @RW2+d16	MUL A, @RW2' @RW2+d16	MULW A, @RW2, @RW2+d16	MULW A, @RW2, @RW2+d16	MULW A, @RW2, @RW2+d16	DIVU A, @RW2, @RW2+d16	DIVU A, @RW2, @RW2+d16	DIVUW A, @RW2, @RW2+d16	DIVUW A, @RW2, @RW2+d16	DIV A, @RW2, @RW2+d16	DIV A, @RW2, @RW2+d16	DIVW A, @RW2, @RW2+d16	DIVW A, @RW2, @RW2+d16
+B	MULU A, @RW3' @RW3+d16	MULU A, @RW3' @RW3+d16	MULUW A, @RW3' @RW3+d16	MULUW A, @RW3' @RW3+d16	MUL A, @RW3' @RW3+d16	MULW A, @RW3' @RW3+d16	MULW A, @RW3' @RW3+d16	MULW A, @RW3' @RW3+d16	DIVU A, @RW3' @RW3+d16	DIVU A, @RW3' @RW3+d16	DIVUW A, @RW3' @RW3+d16	DIVUW A, @RW3' @RW3+d16	DIV A, @RW3' @RW3+d16	DIV A, @RW3' @RW3+d16	DIVW A, @RW3' @RW3+d16	DIVW A, @RW3' @RW3+d16
+C	MULU A, @RW0+ @RW0+RW7	MULU A, @RW0+ @RW0+RW7	MULUW A, @RW0+ @RW0+RW7													

[illegible]

[illegible]



Table B.9-17 MOVW RWi, ea Instruction (First Byte = 7BH)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOVW RW0, RW0, @RW0+d8	MOVW RW1, RW1, @RW1+d8	MOVW RW2, RW2, @RW2+d8	MOVW RW3, RW3, @RW3+d8	MOVW RW4, RW4, @RW4+d8	MOVW RW5, RW5, @RW5+d8	MOVW RW6, RW6, @RW6+d8	MOVW RW7, RW7, @RW7+d8	MOVW RW0, RW0, @RW0+d8	MOVW RW1, RW1, @RW1+d8	MOVW RW2, RW2, @RW2+d8	MOVW RW3, RW3, @RW3+d8	MOVW RW4, RW4, @RW4+d8	MOVW RW5, RW5, @RW5+d8	MOVW RW6, RW6, @RW6+d8	MOVW RW7, RW7, @RW7+d8
+1	MOVW RW0, RW1, @RW1+d8	MOVW RW1, RW2, @RW2+d8	MOVW RW2, RW3, @RW3+d8	MOVW RW3, RW4, @RW4+d8	MOVW RW4, RW5, @RW5+d8	MOVW RW5, RW6, @RW6+d8	MOVW RW6, RW7, @RW7+d8	MOVW RW7, RW0, @RW0+d8	MOVW RW0, RW1, @RW1+d8	MOVW RW1, RW2, @RW2+d8	MOVW RW2, RW3, @RW3+d8	MOVW RW3, RW4, @RW4+d8	MOVW RW4, RW5, @RW5+d8	MOVW RW5, RW6, @RW6+d8	MOVW RW6, RW7, @RW7+d8	MOVW RW7, RW0, @RW0+d8
+2	MOVW RW0, RW2, @RW2+d8	MOVW RW1, RW3, @RW3+d8	MOVW RW2, RW4, @RW4+d8	MOVW RW3, RW5, @RW5+d8	MOVW RW4, RW6, @RW6+d8	MOVW RW5, RW7, @RW7+d8	MOVW RW6, RW0, @RW0+d8	MOVW RW7, RW1, @RW1+d8	MOVW RW0, RW2, @RW2+d8	MOVW RW1, RW3, @RW3+d8	MOVW RW2, RW4, @RW4+d8	MOVW RW3, RW5, @RW5+d8	MOVW RW4, RW6, @RW6+d8	MOVW RW5, RW7, @RW7+d8	MOVW RW6, RW0, @RW0+d8	MOVW RW7, RW1, @RW1+d8
+3	MOVW RW0, RW3, @RW3+d8	MOVW RW1, RW4, @RW4+d8	MOVW RW2, RW5, @RW5+d8	MOVW RW3, RW6, @RW6+d8	MOVW RW4, RW7, @RW7+d8	MOVW RW5, RW0, @RW0+d8	MOVW RW6, RW1, @RW1+d8	MOVW RW7, RW2, @RW2+d8	MOVW RW0, RW3, @RW3+d8	MOVW RW1, RW4, @RW4+d8	MOVW RW2, RW5, @RW5+d8	MOVW RW3, RW6, @RW6+d8	MOVW RW4, RW7, @RW7+d8	MOVW RW5, RW0, @RW0+d8	MOVW RW6, RW1, @RW1+d8	MOVW RW7, RW2, @RW2+d8
+4	MOVW RW0, RW4, @RW4+d8	MOVW RW1, RW5, @RW5+d8	MOVW RW2, RW6, @RW6+d8	MOVW RW3, RW7, @RW7+d8	MOVW RW4, RW0, @RW0+d8	MOVW RW5, RW1, @RW1+d8	MOVW RW6, RW2, @RW2+d8	MOVW RW7, RW3, @RW3+d8	MOVW RW0, RW4, @RW4+d8	MOVW RW1, RW5, @RW5+d8	MOVW RW2, RW6, @RW6+d8	MOVW RW3, RW7, @RW7+d8	MOVW RW4, RW0, @RW0+d8	MOVW RW5, RW1, @RW1+d8	MOVW RW6, RW2, @RW2+d8	MOVW RW7, RW3, @RW3+d8
+5	MOVW RW0, RW5, @RW5+d8	MOVW RW1, RW6, @RW6+d8	MOVW RW2, RW7, @RW7+d8	MOVW RW3, RW0, @RW0+d8	MOVW RW4, RW1, @RW1+d8	MOVW RW5, RW2, @RW2+d8	MOVW RW6, RW3, @RW3+d8	MOVW RW7, RW4, @RW4+d8	MOVW RW0, RW5, @RW5+d8	MOVW RW1, RW6, @RW6+d8	MOVW RW2, RW7, @RW7+d8	MOVW RW3, RW0, @RW0+d8	MOVW RW4, RW1, @RW1+d8	MOVW RW5, RW2, @RW2+d8	MOVW RW6, RW3, @RW3+d8	MOVW RW7, RW4, @RW4+d8
+6	MOVW RW0, RW6, @RW6+d8	MOVW RW1, RW7, @RW7+d8	MOVW RW2, RW0, @RW0+d8	MOVW RW3, RW1, @RW1+d8	MOVW RW4, RW2, @RW2+d8	MOVW RW5, RW3, @RW3+d8	MOVW RW6, RW4, @RW4+d8	MOVW RW7, RW5, @RW5+d8	MOVW RW0, RW6, @RW6+d8	MOVW RW1, RW7, @RW7+d8	MOVW RW2, RW0, @RW0+d8	MOVW RW3, RW1, @RW1+d8	MOVW RW4, RW2, @RW2+d8	MOVW RW5, RW3, @RW3+d8	MOVW RW6, RW4, @RW4+d8	MOVW RW7, RW5, @RW5+d8
+7	MOVW RW0, RW7, @RW7+d8	MOVW RW1, RW0, @RW0+d8	MOVW RW2, RW1, @RW1+d8	MOVW RW3, RW2, @RW2+d8	MOVW RW4, RW3, @RW3+d8	MOVW RW5, RW4, @RW4+d8	MOVW RW6, RW5, @RW5+d8	MOVW RW7, RW6, @RW6+d8	MOVW RW0, RW7, @RW7+d8	MOVW RW1, RW0, @RW0+d8	MOVW RW2, RW1, @RW1+d8	MOVW RW3, RW2, @RW2+d8	MOVW RW4, RW3, @RW3+d8	MOVW RW5, RW4, @RW4+d8	MOVW RW6, RW5, @RW5+d8	MOVW RW7, RW6, @RW6+d8
+8	MOVW RW0, @RW0	MOVW RW1, @RW1	MOVW RW2, @RW2	MOVW RW3, @RW3	MOVW RW4, @RW4	MOVW RW5, @RW5	MOVW RW6, @RW6	MOVW RW7, @RW7	MOVW RW0, @RW0	MOVW RW1, @RW1	MOVW RW2, @RW2	MOVW RW3, @RW3	MOVW RW4, @RW4	MOVW RW5, @RW5	MOVW RW6, @RW6	MOVW RW7, @RW7
+9	MOVW RW0, @RW1	MOVW RW1, @RW2	MOVW RW2, @RW3	MOVW RW3, @RW4	MOVW RW4, @RW5	MOVW RW5, @RW6	MOVW RW6, @RW7	MOVW RW7, @RW0	MOVW RW0, @RW1	MOVW RW1, @RW2	MOVW RW2, @RW3	MOVW RW3, @RW4	MOVW RW4, @RW5	MOVW RW5, @RW6	MOVW RW6, @RW7	MOVW RW7, @RW0
+A	MOVW RW0, @RW2	MOVW RW1, @RW3	MOVW RW2, @RW4	MOVW RW3, @RW5	MOVW RW4, @RW6	MOVW RW5, @RW7	MOVW RW6, @RW0	MOVW RW7, @RW1	MOVW RW0, @RW2	MOVW RW1, @RW3	MOVW RW2, @RW4	MOVW RW3, @RW5	MOVW RW4, @RW6	MOVW RW5, @RW7	MOVW RW6, @RW0	MOVW RW7, @RW1
+B	MOVW RW0, @RW3	MOVW RW1, @RW4	MOVW RW2, @RW5	MOVW RW3, @RW6	MOVW RW4, @RW7	MOVW RW5, @RW0	MOVW RW6, @RW1	MOVW RW7, @RW2	MOVW RW0, @RW3	MOVW RW1, @RW4	MOVW RW2, @RW5	MOVW RW3, @RW6	MOVW RW4, @RW7	MOVW RW5, @RW0	MOVW RW6, @RW1	MOVW RW7, @RW2
+C	MOVW RW0, @RW0+	MOVW RW1, @RW1+	MOVW RW2, @RW2+	MOVW RW3, @RW3+	MOVW RW4, @RW4+	MOVW RW5, @RW5+	MOVW RW6, @RW6+	MOVW RW7, @RW7+	MOVW RW0, @RW0+	MOVW RW1, @RW1+	MOVW RW2, @RW2+	MOVW RW3, @RW3+	MOVW RW4, @RW4+	MOVW RW5, @RW5+	MOVW RW6, @RW6+	MOVW RW7, @RW7+
+D	MOVW RW0, @RW1+	MOVW RW1, @RW2+	MOVW RW2, @RW3+	MOVW RW3, @RW4+	MOVW RW4, @RW5+	MOVW RW5, @RW6+	MOVW RW6, @RW7+	MOVW RW7, @RW0+	MOVW RW0, @RW1+	MOVW RW1, @RW2+	MOVW RW2, @RW3+	MOVW RW3, @RW4+	MOVW RW4, @RW5+	MOVW RW5, @RW6+	MOVW RW6, @RW7+	MOVW RW7, @RW0+
+E	MOVW RW0, @RW2+	MOVW RW1, @RW3+	MOVW RW2, @RW4+	MOVW RW3, @RW5+	MOVW RW4, @RW6+	MOVW RW5, @RW7+	MOVW RW6, @RW0+	MOVW RW7, @RW1+	MOVW RW0, @RW2+	MOVW RW1, @RW3+	MOVW RW2, @RW4+	MOVW RW3, @RW5+	MOVW RW4, @RW6+	MOVW RW5, @RW7+	MOVW RW6, @RW0+	MOVW RW7, @RW1+
+F	MOVW RW0, @RW3+	MOVW RW1, @RW4+	MOVW RW2, @RW5+	MOVW RW3, @RW6+	MOVW RW4, @RW7+	MOVW RW5, @RW0+	MOVW RW6, @RW1+	MOVW RW7, @RW2+	MOVW RW0, @RW3+	MOVW RW1, @RW4+	MOVW RW2, @RW5+	MOVW RW3, @RW6+	MOVW RW4, @RW7+	MOVW RW5, @RW0+	MOVW RW6, @RW1+	MOVW RW7, @RW2+

Table B.9-18 MOV ea, Ri Instruction (First Byte = 7C<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV R0, R0; @RW0+d8, R0	MOV R0, R1; @RW0+d8, R1	MOV R0, R2; @RW0+d8, R2	MOV R0, R3; @RW0+d8, R3	MOV R0, R4; @RW0+d8, R4	MOV R0, R5; @RW0+d8, R5	MOV R0, R6; @RW0+d8, R6	MOV R0, R7; @RW0+d8, R7	MOV R0, R8; @RW0+d8, R8	MOV R0, R9; @RW0+d8, R9	MOV R0, R10; @RW0+d8, R10	MOV R0, R11; @RW0+d8, R11	MOV R0, R12; @RW0+d8, R12	MOV R0, R13; @RW0+d8, R13	MOV R0, R14; @RW0+d8, R14	MOV R0, R15; @RW0+d8, R15
+1	MOV R1, R0; @RW1+d8, R0	MOV R1, R1; @RW1+d8, R1	MOV R1, R2; @RW1+d8, R2	MOV R1, R3; @RW1+d8, R3	MOV R1, R4; @RW1+d8, R4	MOV R1, R5; @RW1+d8, R5	MOV R1, R6; @RW1+d8, R6	MOV R1, R7; @RW1+d8, R7	MOV R1, R8; @RW1+d8, R8	MOV R1, R9; @RW1+d8, R9	MOV R1, R10; @RW1+d8, R10	MOV R1, R11; @RW1+d8, R11	MOV R1, R12; @RW1+d8, R12	MOV R1, R13; @RW1+d8, R13	MOV R1, R14; @RW1+d8, R14	MOV R1, R15; @RW1+d8, R15
+2	MOV R2, R0; @RW2+d8, R0	MOV R2, R1; @RW2+d8, R1	MOV R2, R2; @RW2+d8, R2	MOV R2, R3; @RW2+d8, R3	MOV R2, R4; @RW2+d8, R4	MOV R2, R5; @RW2+d8, R5	MOV R2, R6; @RW2+d8, R6	MOV R2, R7; @RW2+d8, R7	MOV R2, R8; @RW2+d8, R8	MOV R2, R9; @RW2+d8, R9	MOV R2, R10; @RW2+d8, R10	MOV R2, R11; @RW2+d8, R11	MOV R2, R12; @RW2+d8, R12	MOV R2, R13; @RW2+d8, R13	MOV R2, R14; @RW2+d8, R14	MOV R2, R15; @RW2+d8, R15
+3	MOV R3, R0; @RW3+d8, R0	MOV R3, R1; @RW3+d8, R1	MOV R3, R2; @RW3+d8, R2	MOV R3, R3; @RW3+d8, R3	MOV R3, R4; @RW3+d8, R4	MOV R3, R5; @RW3+d8, R5	MOV R3, R6; @RW3+d8, R6	MOV R3, R7; @RW3+d8, R7	MOV R3, R8; @RW3+d8, R8	MOV R3, R9; @RW3+d8, R9	MOV R3, R10; @RW3+d8, R10	MOV R3, R11; @RW3+d8, R11	MOV R3, R12; @RW3+d8, R12	MOV R3, R13; @RW3+d8, R13	MOV R3, R14; @RW3+d8, R14	MOV R3, R15; @RW3+d8, R15
+4	MOV R4, R0; @RW4+d8, R0	MOV R4, R1; @RW4+d8, R1	MOV R4, R2; @RW4+d8, R2	MOV R4, R3; @RW4+d8, R3	MOV R4, R4; @RW4+d8, R4	MOV R4, R5; @RW4+d8, R5	MOV R4, R6; @RW4+d8, R6	MOV R4, R7; @RW4+d8, R7	MOV R4, R8; @RW4+d8, R8	MOV R4, R9; @RW4+d8, R9	MOV R4, R10; @RW4+d8, R10	MOV R4, R11; @RW4+d8, R11	MOV R4, R12; @RW4+d8, R12	MOV R4, R13; @RW4+d8, R13	MOV R4, R14; @RW4+d8, R14	MOV R4, R15; @RW4+d8, R15
+5	MOV R5, R0; @RW5+d8, R0	MOV R5, R1; @RW5+d8, R1	MOV R5, R2; @RW5+d8, R2	MOV R5, R3; @RW5+d8, R3	MOV R5, R4; @RW5+d8, R4	MOV R5, R5; @RW5+d8, R5	MOV R5, R6; @RW5+d8, R6	MOV R5, R7; @RW5+d8, R7	MOV R5, R8; @RW5+d8, R8	MOV R5, R9; @RW5+d8, R9	MOV R5, R10; @RW5+d8, R10	MOV R5, R11; @RW5+d8, R11	MOV R5, R12; @RW5+d8, R12	MOV R5, R13; @RW5+d8, R13	MOV R5, R14; @RW5+d8, R14	MOV R5, R15; @RW5+d8, R15
+6	MOV R6, R0; @RW6+d8, R0	MOV R6, R1; @RW6+d8, R1	MOV R6, R2; @RW6+d8, R2	MOV R6, R3; @RW6+d8, R3	MOV R6, R4; @RW6+d8, R4	MOV R6, R5; @RW6+d8, R5	MOV R6, R6; @RW6+d8, R6	MOV R6, R7; @RW6+d8, R7	MOV R6, R8; @RW6+d8, R8	MOV R6, R9; @RW6+d8, R9	MOV R6, R10; @RW6+d8, R10	MOV R6, R11; @RW6+d8, R11	MOV R6, R12; @RW6+d8, R12	MOV R6, R13; @RW6+d8, R13	MOV R6, R14; @RW6+d8, R14	MOV R6, R15; @RW6+d8, R15
+7	MOV R7, R0; @RW7+d8, R0	MOV R7, R1; @RW7+d8, R1	MOV R7, R2; @RW7+d8, R2	MOV R7, R3; @RW7+d8, R3	MOV R7, R4; @RW7+d8, R4	MOV R7, R5; @RW7+d8, R5	MOV R7, R6; @RW7+d8, R6	MOV R7, R7; @RW7+d8, R7	MOV R7, R8; @RW7+d8, R8	MOV R7, R9; @RW7+d8, R9	MOV R7, R10; @RW7+d8, R10	MOV R7, R11; @RW7+d8, R11	MOV R7, R12; @RW7+d8, R12	MOV R7, R13; @RW7+d8, R13	MOV R7, R14; @RW7+d8, R14	MOV R7, R15; @RW7+d8, R15
+8	MOV @RW0, R0; @RW0+d16, R0	MOV @RW0, R1; @RW0+d16, R1	MOV @RW0, R2; @RW0+d16, R2	MOV @RW0, R3; @RW0+d16, R3	MOV @RW0, R4; @RW0+d16, R4	MOV @RW0, R5; @RW0+d16, R5	MOV @RW0, R6; @RW0+d16, R6	MOV @RW0, R7; @RW0+d16, R7	MOV @RW0, R8; @RW0+d16, R8	MOV @RW0, R9; @RW0+d16, R9	MOV @RW0, R10; @RW0+d16, R10	MOV @RW0, R11; @RW0+d16, R11	MOV @RW0, R12; @RW0+d16, R12	MOV @RW0, R13; @RW0+d16, R13	MOV @RW0, R14; @RW0+d16, R14	MOV @RW0, R15; @RW0+d16, R15
+9	MOV @RW1, R0; @RW1+d16, R0	MOV @RW1, R1; @RW1+d16, R1	MOV @RW1, R2; @RW1+d16, R2	MOV @RW1, R3; @RW1+d16, R3	MOV @RW1, R4; @RW1+d16, R4	MOV @RW1, R5; @RW1+d16, R5	MOV @RW1, R6; @RW1+d16, R6	MOV @RW1, R7; @RW1+d16, R7	MOV @RW1, R8; @RW1+d16, R8	MOV @RW1, R9; @RW1+d16, R9	MOV @RW1, R10; @RW1+d16, R10	MOV @RW1, R11; @RW1+d16, R11	MOV @RW1, R12; @RW1+d16, R12	MOV @RW1, R13; @RW1+d16, R13	MOV @RW1, R14; @RW1+d16, R14	MOV @RW1, R15; @RW1+d16, R15
+A	MOV @RW2, R0; @RW2+d16, R0	MOV @RW2, R1; @RW2+d16, R1	MOV @RW2, R2; @RW2+d16, R2	MOV @RW2, R3; @RW2+d16, R3	MOV @RW2, R4; @RW2+d16, R4	MOV @RW2, R5; @RW2+d16, R5	MOV @RW2, R6; @RW2+d16, R6	MOV @RW2, R7; @RW2+d16, R7	MOV @RW2, R8; @RW2+d16, R8	MOV @RW2, R9; @RW2+d16, R9	MOV @RW2, R10; @RW2+d16, R10	MOV @RW2, R11; @RW2+d16, R11	MOV @RW2, R12; @RW2+d16, R12	MOV @RW2, R13; @RW2+d16, R13	MOV @RW2, R14; @RW2+d16, R14	MOV @RW2, R15; @RW2+d16, R15
+B	MOV @RW3, R0; @RW3+d16, R0	MOV @RW3, R1; @RW3+d16, R1	MOV @RW3, R2; @RW3+d16, R2	MOV @RW3, R3; @RW3+d16, R3	MOV @RW3, R4; @RW3+d16, R4	MOV @RW3, R5; @RW3+d16, R5	MOV @RW3, R6; @RW3+d16, R6	MOV @RW3, R7; @RW3+d16, R7	MOV @RW3, R8; @RW3+d16, R8	MOV @RW3, R9; @RW3+d16, R9	MOV @RW3, R10; @RW3+d16, R10	MOV @RW3, R11; @RW3+d16, R11	MOV @RW3, R12; @RW3+d16, R12	MOV @RW3, R13; @RW3+d16, R13	MOV @RW3, R14; @RW3+d16, R14	MOV @RW3, R15; @RW3+d16, R15
+C	MOV @RW0+, R0; @RW0+RW7, R0	MOV @RW0+, R1; @RW0+RW7, R1	MOV @RW0+, R2; @RW0+RW7, R2	MOV @RW0+, R3; @RW0+RW7, R3	MOV @RW0+, R4; @RW0+RW7, R4	MOV @RW0+, R5; @RW0+RW7, R5	MOV @RW0+, R6; @RW0+RW7, R6	MOV @RW0+, R7; @RW0+RW7, R7	MOV @RW0+, R8; @RW0+RW7, R8	MOV @RW0+, R9; @RW0+RW7, R9	MOV @RW0+, R10; @RW0+RW7, R10	MOV @RW0+, R11; @RW0+RW7, R11	MOV @RW0+, R12; @RW0+RW7, R12	MOV @RW0+, R13; @RW0+RW7, R13	MOV @RW0+, R14; @RW0+RW7, R14	MOV @RW0+, R15; @RW0+RW7, R15
+D	MOV @RW1+, R0; @RW1+RW7, R0	MOV @RW1+, R1; @RW1+RW7, R1	MOV @RW1+, R2; @RW1+RW7, R2	MOV @RW1+, R3; @RW1+RW7, R3	MOV @RW1+, R4; @RW1+RW7, R4	MOV @RW1+, R5; @RW1+RW7, R5	MOV @RW1+, R6; @RW1+RW7, R6	MOV @RW1+, R7; @RW1+RW7, R7	MOV @RW1+, R8; @RW1+RW7, R8	MOV @RW1+, R9; @RW1+RW7, R9	MOV @RW1+, R10; @RW1+RW7, R10	MOV @RW1+, R11; @RW1+RW7, R11	MOV @RW1+, R12; @RW1+RW7, R12	MOV @RW1+, R13; @RW1+RW7, R13	MOV @RW1+, R14; @RW1+RW7, R14	MOV @RW1+, R15; @RW1+RW7, R15
+E	MOV @RW2+, R0; @PC+d16, R0	MOV @RW2+, R1; @PC+d16, R1	MOV @RW2+, R2; @PC+d16, R2	MOV @RW2+, R3; @PC+d16, R3	MOV @RW2+, R4; @PC+d16, R4	MOV @RW2+, R5; @PC+d16, R5	MOV @RW2+, R6; @PC+d16, R6	MOV @RW2+, R7; @PC+d16, R7	MOV @RW2+, R8; @PC+d16, R8	MOV @RW2+, R9; @PC+d16, R9	MOV @RW2+, R10; @PC+d16, R10	MOV @RW2+, R11; @PC+d16, R11	MOV @RW2+, R12; @PC+d16, R12	MOV @RW2+, R13; @PC+d16, R13	MOV @RW2+, R14; @PC+d16, R14	MOV @RW2+, R15; @PC+d16, R15
+F	MOV @RW3+, R0; addr16, R0	MOV @RW3+, R1; addr16, R1	MOV @RW3+, R2; addr16, R2	MOV @RW3+, R3; addr16, R3	MOV @RW3+, R4; addr16, R4	MOV @RW3+, R5; addr16, R5	MOV @RW3+, R6; addr16, R6	MOV @RW3+, R7; addr16, R7	MOV @RW3+, R8; addr16, R8	MOV @RW3+, R9; addr16, R9	MOV @RW3+, R10; addr16, R10	MOV @RW3+, R11; addr16, R11	MOV @RW3+, R12; addr16, R12	MOV @RW3+, R13; addr16, R13	MOV @RW3+, R14; addr16, R14	MOV @RW3+, R15; addr16, R15

Table B.9-19 MOVW ea, Rwi Instruction (First Byte = 7D<sub>H</sub>)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOVW RW0, RW0, @RW0+8, RW0	MOVW RW0, RW1, @RW0+8, RW1	MOVW RW0, RW1, @RW1+8, RW1	MOVW RW0, RW2, @RW0+8, RW2	MOVW RW0, RW2, @RW1+8, RW2	MOVW RW0, RW2, @RW2+8, RW2	MOVW RW0, RW3, @RW0+8, RW3	MOVW RW0, RW3, @RW1+8, RW3	MOVW RW0, RW4, @RW0+8, RW4	MOVW RW0, RW4, @RW1+8, RW4	MOVW RW0, RW5, @RW0+8, RW5	MOVW RW0, RW5, @RW1+8, RW5	MOVW RW0, RW6, @RW0+8, RW6	MOVW RW0, RW6, @RW1+8, RW6	MOVW RW0, RW7, @RW0+8, RW7	MOVW RW0, RW7, @RW1+8, RW7
+1	MOVW RW1, RW0, @RW1+8, RW0	MOVW RW1, RW1, @RW1+8, RW1	MOVW RW1, RW2, @RW1+8, RW2	MOVW RW1, RW2, @RW2+8, RW2	MOVW RW1, RW3, @RW1+8, RW3	MOVW RW1, RW3, @RW2+8, RW3	MOVW RW1, RW4, @RW1+8, RW4	MOVW RW1, RW4, @RW2+8, RW4	MOVW RW1, RW5, @RW1+8, RW5	MOVW RW1, RW5, @RW2+8, RW5	MOVW RW1, RW6, @RW1+8, RW6	MOVW RW1, RW6, @RW2+8, RW6	MOVW RW1, RW7, @RW1+8, RW7	MOVW RW1, RW7, @RW2+8, RW7	MOVW RW1, RW7, @RW3+8, RW7	MOVW RW1, RW7, @RW4+8, RW7
+2	MOVW RW2, RW0, @RW2+8, RW0	MOVW RW2, RW1, @RW2+8, RW1	MOVW RW2, RW2, @RW2+8, RW2	MOVW RW2, RW3, @RW2+8, RW3	MOVW RW2, RW3, @RW3+8, RW3	MOVW RW2, RW4, @RW2+8, RW4	MOVW RW2, RW4, @RW3+8, RW4	MOVW RW2, RW5, @RW2+8, RW5	MOVW RW2, RW5, @RW3+8, RW5	MOVW RW2, RW6, @RW2+8, RW6	MOVW RW2, RW6, @RW3+8, RW6	MOVW RW2, RW7, @RW2+8, RW7	MOVW RW2, RW7, @RW3+8, RW7	MOVW RW2, RW7, @RW4+8, RW7	MOVW RW2, RW7, @RW5+8, RW7	MOVW RW2, RW7, @RW6+8, RW7
+3	MOVW RW3, RW0, @RW3+8, RW0	MOVW RW3, RW1, @RW3+8, RW1	MOVW RW3, RW2, @RW3+8, RW2	MOVW RW3, RW3, @RW3+8, RW3	MOVW RW3, RW4, @RW3+8, RW4	MOVW RW3, RW4, @RW4+8, RW4	MOVW RW3, RW5, @RW3+8, RW5	MOVW RW3, RW5, @RW4+8, RW5	MOVW RW3, RW6, @RW3+8, RW6	MOVW RW3, RW6, @RW4+8, RW6	MOVW RW3, RW7, @RW3+8, RW7	MOVW RW3, RW7, @RW4+8, RW7	MOVW RW3, RW7, @RW5+8, RW7	MOVW RW3, RW7, @RW6+8, RW7	MOVW RW3, RW7, @RW7+8, RW7	MOVW RW3, RW7, @RW8+8, RW7
+4	MOVW RW4, RW0, @RW4+8, RW0	MOVW RW4, RW1, @RW4+8, RW1	MOVW RW4, RW2, @RW4+8, RW2	MOVW RW4, RW3, @RW4+8, RW3	MOVW RW4, RW4, @RW4+8, RW4	MOVW RW4, RW5, @RW4+8, RW5	MOVW RW4, RW5, @RW5+8, RW5	MOVW RW4, RW6, @RW4+8, RW6	MOVW RW4, RW6, @RW5+8, RW6	MOVW RW4, RW7, @RW4+8, RW7	MOVW RW4, RW7, @RW5+8, RW7	MOVW RW4, RW7, @RW6+8, RW7	MOVW RW4, RW7, @RW7+8, RW7	MOVW RW4, RW7, @RW8+8, RW7	MOVW RW4, RW7, @RW9+8, RW7	MOVW RW4, RW7, @RW10+8, RW7
+5	MOVW RW5, RW0, @RW5+8, RW0	MOVW RW5, RW1, @RW5+8, RW1	MOVW RW5, RW2, @RW5+8, RW2	MOVW RW5, RW3, @RW5+8, RW3	MOVW RW5, RW4, @RW5+8, RW4	MOVW RW5, RW5, @RW5+8, RW5	MOVW RW5, RW6, @RW5+8, RW6	MOVW RW5, RW7, @RW5+8, RW7	MOVW RW5, RW7, @RW6+8, RW7	MOVW RW5, RW7, @RW7+8, RW7	MOVW RW5, RW7, @RW8+8, RW7	MOVW RW5, RW7, @RW9+8, RW7	MOVW RW5, RW7, @RW10+8, RW7	MOVW RW5, RW7, @RW11+8, RW7	MOVW RW5, RW7, @RW12+8, RW7	MOVW RW5, RW7, @RW13+8, RW7
+6	MOVW RW6, RW0, @RW6+8, RW0	MOVW RW6, RW1, @RW6+8, RW1	MOVW RW6, RW2, @RW6+8, RW2	MOVW RW6, RW3, @RW6+8, RW3	MOVW RW6, RW4, @RW6+8, RW4	MOVW RW6, RW5, @RW6+8, RW5	MOVW RW6, RW6, @RW6+8, RW6	MOVW RW6, RW7, @RW6+8, RW7	MOVW RW6, RW7, @RW7+8, RW7	MOVW RW6, RW7, @RW8+8, RW7	MOVW RW6, RW7, @RW9+8, RW7	MOVW RW6, RW7, @RW10+8, RW7	MOVW RW6, RW7, @RW11+8, RW7	MOVW RW6, RW7, @RW12+8, RW7	MOVW RW6, RW7, @RW13+8, RW7	MOVW RW6, RW7, @RW14+8, RW7
+7	MOVW RW7, RW0, @RW7+8, RW0	MOVW RW7, RW1, @RW7+8, RW1	MOVW RW7, RW2, @RW7+8, RW2	MOVW RW7, RW3, @RW7+8, RW3	MOVW RW7, RW4, @RW7+8, RW4	MOVW RW7, RW5, @RW7+8, RW5	MOVW RW7, RW6, @RW7+8, RW6	MOVW RW7, RW7, @RW7+8, RW7	MOVW RW7, RW7, @RW8+8, RW7	MOVW RW7, RW7, @RW9+8, RW7	MOVW RW7, RW7, @RW10+8, RW7	MOVW RW7, RW7, @RW11+8, RW7	MOVW RW7, RW7, @RW12+8, RW7	MOVW RW7, RW7, @RW13+8, RW7	MOVW RW7, RW7, @RW14+8, RW7	MOVW RW7, RW7, @RW15+8, RW7
+8	MOVW @RW0, RW0, +d16, RW0	MOVW @RW0, RW1, +d16, RW1	MOVW @RW0, RW2, +d16, RW2	MOVW @RW0, RW3, +d16, RW3	MOVW @RW0, RW4, +d16, RW4	MOVW @RW0, RW5, +d16, RW5	MOVW @RW0, RW6, +d16, RW6	MOVW @RW0, RW7, +d16, RW7	MOVW @RW0, RW7, +d16, RW7	MOVW @RW0, RW7, +d16, RW7	MOVW @RW0, RW7, +d16, RW7	MOVW @RW0, RW7, +d16, RW7	MOVW @RW0, RW7, +d16, RW7	MOVW @RW0, RW7, +d16, RW7	MOVW @RW0, RW7, +d16, RW7	MOVW @RW0, RW7, +d16, RW7
+9	MOVW @RW1, RW0, +d16, RW0	MOVW @RW1, RW1, +d16, RW1	MOVW @RW1, RW2, +d16, RW2	MOVW @RW1, RW3, +d16, RW3	MOVW @RW1, RW4, +d16, RW4	MOVW @RW1, RW5, +d16, RW5	MOVW @RW1, RW6, +d16, RW6	MOVW @RW1, RW7, +d16, RW7	MOVW @RW1, RW7, +d16, RW7	MOVW @RW1, RW7, +d16, RW7	MOVW @RW1, RW7, +d16, RW7	MOVW @RW1, RW7, +d16, RW7	MOVW @RW1, RW7, +d16, RW7	MOVW @RW1, RW7, +d16, RW7	MOVW @RW1, RW7, +d16, RW7	MOVW @RW1, RW7, +d16, RW7
+A	MOVW @RW2, RW0, +d16, RW0	MOVW @RW2, RW1, +d16, RW1	MOVW @RW2, RW2, +d16, RW2	MOVW @RW2, RW3, +d16, RW3	MOVW @RW2, RW4, +d16, RW4	MOVW @RW2, RW5, +d16, RW5	MOVW @RW2, RW6, +d16, RW6	MOVW @RW2, RW7, +d16, RW7	MOVW @RW2, RW7, +d16, RW7	MOVW @RW2, RW7, +d16, RW7	MOVW @RW2, RW7, +d16, RW7	MOVW @RW2, RW7, +d16, RW7	MOVW @RW2, RW7, +d16, RW7	MOVW @RW2, RW7, +d16, RW7	MOVW @RW2, RW7, +d16, RW7	MOVW @RW2, RW7, +d16, RW7
+B	MOVW @RW3, RW0, +d16, RW0	MOVW @RW3, RW1, +d16, RW1	MOVW @RW3, RW2, +d16, RW2	MOVW @RW3, RW3, +d16, RW3	MOVW @RW3, RW4, +d16, RW4	MOVW @RW3, RW5, +d16, RW5	MOVW @RW3, RW6, +d16, RW6	MOVW @RW3, RW7, +d16, RW7	MOVW @RW3, RW7, +d16, RW7	MOVW @RW3, RW7, +d16, RW7	MOVW @RW3, RW7, +d16, RW7	MOVW @RW3, RW7, +d16, RW7	MOVW @RW3, RW7, +d16, RW7	MOVW @RW3, RW7, +d16, RW7	MOVW @RW3, RW7, +d16, RW7	MOVW @RW3, RW7, +d16, RW7
+C	MOVW @RW0+, RW0, +RW7, RW0	MOVW @RW0+, RW1, +RW7, RW1	MOVW @RW0+, RW2, +RW7, RW2	MOVW @RW0+, RW3, +RW7, RW3	MOVW @RW0+, RW4, +RW7, RW4	MOVW @RW0+, RW5, +RW7, RW5	MOVW @RW0+, RW6, +RW7, RW6	MOVW @RW0+, RW7, +RW7, RW7	MOVW @RW0+, RW7, +RW7, RW7	MOVW @RW0+, RW7, +RW7, RW7	MOVW @RW0+, RW7, +RW7, RW7	MOVW @RW0+, RW7, +RW7, RW7	MOVW @RW0+, RW7, +RW7, RW7	MOVW @RW0+, RW7, +RW7, RW7	MOVW @RW0+, RW7, +RW7, RW7	MOVW @RW0+, RW7, +RW7, RW7
+D	MOVW @RW1+, RW0, +RW7, RW0	MOVW @RW1+, RW1, +RW7, RW1	MOVW @RW1+, RW2, +RW7, RW2	MOVW @RW1+, RW3, +RW7, RW3	MOVW @RW1+, RW4, +RW7, RW4	MOVW @RW1+, RW5, +RW7, RW5	MOVW @RW1+, RW6, +RW7, RW6	MOVW @RW1+, RW7, +RW7, RW7	MOVW @RW1+, RW7, +RW7, RW7	MOVW @RW1+, RW7, +RW7, RW7	MOVW @RW1+, RW7, +RW7, RW7	MOVW @RW1+, RW7, +RW7, RW7	MOVW @RW1+, RW7, +RW7, RW7	MOVW @RW1+, RW7, +RW7, RW7	MOVW @RW1+, RW7, +RW7, RW7	MOVW @RW1+, RW7, +RW7, RW7
+E	MOVW @RW2+, RW0, +d16, RW0	MOVW @RW2+, RW1, +d16, RW1	MOVW @RW2+, RW2, +d16, RW2	MOVW @RW2+, RW3, +d16, RW3	MOVW @RW2+, RW4, +d16, RW4	MOVW @RW2+, RW5, +d16, RW5	MOVW @RW2+, RW6, +d16, RW6	MOVW @RW2+, RW7, +d16, RW7	MOVW @RW2+, RW7, +d16, RW7	MOVW @RW2+, RW7, +d16, RW7	MOVW @RW2+, RW7, +d16, RW7	MOVW @RW2+, RW7, +d16, RW7	MOVW @RW2+, RW7, +d16, RW7	MOVW @RW2+, RW7, +d16, RW7	MOVW @RW2+, RW7, +d16, RW7	MOVW @RW2+, RW7, +d16, RW7
+F	MOVW @RW3+, RW0, +addr16, RW0	MOVW @RW3+, RW1, +addr16, RW1	MOVW @RW3+, RW2, +addr16, RW2	MOVW @RW3+, RW3, +addr16, RW3	MOVW @RW3+, RW4, +addr16, RW4	MOVW @RW3+, RW5, +addr16, RW5	MOVW @RW3+, RW6, +addr16, RW6	MOVW @RW3+, RW7, +addr16, RW7	MOVW @RW3+, RW7, +addr16, RW7	MOVW @RW3+, RW7, +addr16, RW7	MOVW @RW3+, RW7, +addr16, RW7	MOVW @RW3+, RW7, +addr16, RW7	MOVW @RW3+, RW7, +addr16, RW7	MOVW @RW3+, RW7, +addr16, RW7	MOVW @RW3+, RW7, +addr16, RW7	MOVW @RW3+, RW7, +addr16, RW7

[illegible]

**Table B.9-21 XCHW RWi, ea Instruction (First Byte = 7F<sub>H</sub>)**

[illegible]

# INDEX

---

The index follows on the next page.  
This is listed in alphabetic order.

---

# Index

## Numerics

### 16-bit Bus

External 16-bit Bus Mode (External Data Bus 16-bit/ Multiplex Mode).....	193
---	-----

### 16-bit Data Bus

Pin State in External Bus 16-bit Data Bus and Multiplex 16-bit External Bus Mode .....	166
Pin State in External Bus 16-bit Data Bus and Non-multiplex 16-bit External Bus Mode .....	170

### 16-bit External Bus

Pin State in External Bus 16-bit Data Bus and Multiplex 16-bit External Bus Mode .....	166
Pin State in External Bus 16-bit Data Bus and Non-multiplex 16-bit External Bus Mode .....	170

### 16-bit Free-run Timer

Block Diagram of 16-bit Free-run Timer .....	244
Operation of 16-bit Free-run Timer .....	258
Register List of 16-bit Free-run Timer .....	243

### 16-bit I/O Timer

Configuration Function of 16-bit I/O Timer .....	240
Operation and Timing of 16-bit I/O Timer .....	257
Register Configuration of 16-bit I/O Timer .....	242

### 16-bit Reload Register

16-bit Reload Register 0 to 2 (TMRLR0 to TMRLR2) .....	402
16-bit Timer Register 0 to 2 (TMR0 to TMR2)/ 16-bit Reload Register 0 to 2 (TMRLR0 to TMRLR2) .....	401

### 16-bit Reload Timer

Block Diagram of 16-bit Reload Timer .....	395
Operation Modes of 16-bit Reload Timer.....	393
Overview of 16-bit Reload Timer .....	392
Register List of 16-bit Reload Timer .....	396
Setting of 16-bit Reload Timer .....	403

### 16-bit Timer Register

16-bit Timer Register 0 to 2 (TMR0 to TMR2) .....	401
16-bit Timer Register 0 to 2 (TMR0 to TMR2)/ 16-bit Reload Register 0 to 2 (TMRLR0 to TMRLR2) .....	401

### 16-bit/Multiplex Mode

External 16-bit Bus Mode (External Data Bus 16-bit/ Multiplex Mode).....	193
---	-----

### 8/10-bit A/D Converter

Block Diagram of 8/10-bit A/D Converter .....	441
Function of 8/10-bit A/D Converter.....	440
Precautions when Using 8/10-bit A/D Converter .....	458
Register List of 8/10-bit A/D Converter .....	443

### 8/16-bit PPG Timer

Block Diagram of 8/16-bit PPG Timer .....	413
Interrupts of 8/16-bit PPG Timer .....	427
Outline of Operation of 8/16-bit PPG Timer .....	424
Register List of 8/16-bit PPG Timer .....	415

### 8/16-bit Timer

Overview of 8/16-bit Timer PPG Timer .....	412
--	-----

### 8-bit Data Bus

Pin State in External Bus 8-bit Data Bus and Multiplex 8-bit External Bus Mode .....	168
Pin State in External Bus 8-bit Data Bus and Non-multiplex 8-bit External Bus Mode .....	172

### 8-bit External Bus

Pin State in External Bus 8-bit Data Bus and Multiplex 8-bit External Bus Mode .....	168
Pin State in External Bus 8-bit Data Bus and Non-multiplex 8-bit External Bus Mode .....	172

<b>A</b>	
<b>A</b>	
Accumulator (A) .....	35
<b>A/D Control Status Register</b>	
A/D Control Status Register (High) (ADCS1) .....	444
A/D Control Status Register (Low) (ADCS0) .....	446
<b>A/D Conversion Channel Set Register</b>	
A/D Conversion Channel Set Register (ADMR) .....	448
<b>A/D Converter</b>	
Block Diagram of 8/10-bit A/D Converter .....	441
Function of 8/10-bit A/D Converter .....	440
Precautions when Using 8/10-bit A/D Converter .....	458
Register List of 8/10-bit A/D Converter .....	443
<b>A/D Data Register</b>	
A/D Data Register (ADCR1/ADCR0) .....	450
<b>A/D-converted Data</b>	
A/D-converted Data Protection Function .....	456
<b>Access Mode</b>	
Access Mode .....	178
<b>Accumulator</b>	
Accumulator (A) .....	35
<b>Acknowledge</b>	
Acknowledge .....	547
<b>ADB</b>	
Bank Registers (PCB, DTB, USB, SSB, ADB) .....	41
<b>ADCR</b>	
A/D Data Register (ADCR1/ADCR0) .....	450
<b>ADCS</b>	
A/D Control Status Register (High) (ADCS1) .....	444
A/D Control Status Register (Low) (ADCS0) .....	446
<b>Address Detection Control Register</b>	
Address Detection Control Register (PACSR) .....	561
<b>Address Generation Method</b>	
Address Generation Methods .....	27
<b>Address Match Detection Function</b>	
Block Diagram of Address Match Detection Function .....	559
List of Registers and Reset Values of Address Match Detection Function .....	560
Operation of Address Match Detection Function .....	565
Operation of Address Match Detection Function at Storing Patch Program in E <sup>2</sup> PROM .....	569
Overview of Address Match Detection Function .....	558
Program Example for Address Match Detection Function .....	571
<b>Addressing</b>	
Addressing .....	546, 632
Direct Addressing .....	634
Indirect Addressing .....	640
<b>ADER</b>	
Analog Input Enable Register (ADER0, ADER1) .....	205
<b>ADMR</b>	
A/D Conversion Channel Set Register (ADMR) .....	448
<b>Analog Input Enable Register</b>	
Analog Input Enable Register (ADER0, ADER1) .....	205
<b>Arbitration</b>	
Arbitration .....	546
<b>ARSR</b>	
Automatic Ready Function Selection Register (ARSR) .....	186
<b>Asynchronous Mode</b>	
Operation in Asynchronous Mode .....	516
<b>Automatic Algorithm</b>	
Automatic Algorithm End Timing .....	578
<b>Automatic Ready Function Selection Register</b>	
Automatic Ready Function Selection Register (ARSR) .....	186
<b>B</b>	
<b>Bank Addressing</b>	
Bank Addressing .....	29
<b>Bank Registers</b>	
Bank Registers (PCB, DTB, USB, SSB, ADB) .....	41
<b>Bank Select Prefix</b>	
Bank Select Prefix .....	44
<b>BAP</b>	
Buffer address Pointer (BAP) .....	82
<b>Basic Component</b>	
The Serial On-board Writing Basic Component .....	598
<b>Baud Rate</b>	
Baud Rate of the External Clock (One-to-one Mode) .....	513
Baud Rate of the External Clock Using the Dedicated Baud Rate Generator .....	512
Baud Rate of the Internal Clock Using the Dedicated Baud Rate Generator .....	511
UART Baud Rate Selection .....	510
<b>Bi-directional Communication</b>	
Bi-directional Communication Function .....	522
<b>Bit</b>	
Priority Level of STP, SLP, and TMD Bit .....	151
<b>Block Diagram</b>	
Block Diagram .....	184, 241
Block Diagram in Extended I/O Serial Interface .....	470
Block Diagram of 16-bit Free-run Timer .....	244
Block Diagram of 16-bit Reload Timer .....	395
Block Diagram of 8/10-bit A/D Converter .....	441
Block Diagram of 8/16-bit PPG Timer .....	413
Block Diagram of Address Match Detection Function .....	559
Block Diagram of Clock Generation Section .....	131
Block Diagram of Delayed Interrupt Generation Module .....	112
Block Diagram of DTP/External Interrupt .....	430
Block Diagram of External Reset Pin .....	120
Block Diagram of Input Capture .....	254
Block Diagram of Low-power Consumption Control Circuit .....	147
Block Diagram of Output Compare .....	250
Block Diagram of PWC Timer .....	365
Block Diagram of ROM Mirror Function Select Module .....	554
Block Diagram of the MB90330A Series .....	7



## INDEX

Block Diagram of Time-base Timer .....	210
Block Diagram of USB Function .....	269
Block Diagram of Watch Timer .....	235
Block Diagram of Watchdog Timer.....	226
UART Block Diagram .....	488
UART Block Diagram of USB HOST .....	320
<b>Buffer address Pointer</b>	
Buffer address Pointer (BAP) .....	82
<b>Bus Control Signal Selection Register</b>	
Bus Control Signal Selection Register (EPCR).....	188
<b>Bus Error</b>	
Bus Error .....	547
<b>Bus Mode</b>	
Bus Modes .....	178
Set Bit of Bus Mode (M1, M0).....	181
<b>C</b>	
<b>Calculating</b>	
Calculating the Execution Cycle Count .....	649
<b>Capture Timing</b>	
Capture Timing to Input Signal .....	265
<b>CCR</b>	
Condition Code Register (CCR) .....	37
<b>Change Timing</b>	
Change Timing for Output Pin .....	264
<b>Chip</b>	
When Chip/sector Deletion Operates .....	582
<b>Chip Erase</b>	
Erasing All Data from Flash Memory (Chip Erase) .....	591
<b>Chip Sector</b>	
When Writing/chip Sector is Deleted .....	584, 585
<b>CKSCR</b>	
Configuration of Clock Select Register (CKSCR).....	134
<b>Clear Timing</b>	
Free-run Timer Clear Timing (for a Match with Compare Clear Register) .....	263
<b>Clearing Timer</b>	
Clearing Timer .....	381
<b>Clock</b>	
Acquiring Transfer Speed of Destination USB Device and Selecting Clock .....	345
Baud Rate of the External Clock (One-to-one Mode).....	513
Baud Rate of the External Clock Using the Dedicated Baud Rate Generator.....	512
Baud Rate of the Internal Clock Using the Dedicated Baud Rate Generator.....	511
Clock Supply Map .....	129
Connection of Oscillator and External Clock .....	141
Count Clock and Maximum Cycle .....	382
Count Clock Selection.....	377, 426
Event Count Mode (External Clock Mode) .....	393
Function of Clock Supply .....	209, 216
Internal Clock Mode .....	393
Internal Clock Mode (Single Shot Mode).....	407
Operation of Internal Clock Mode (Reload Mode) .....	405
Overview of Clock.....	128
<b>Clock Generation Section</b>	
Block Diagram of Clock Generation Section .....	131
<b>Clock Mode</b>	
Clock Mode .....	145
External Shift Clock Mode .....	479
Internal Shift Clock Mode .....	479
Switching Clock Mode .....	175
Transition of Clock Mode.....	137
<b>Clock Select Register</b>	
Configuration of Clock Select Register (CKSCR) .....	134
<b>Clock Source</b>	
Specification Function of Clock Source of Watchdog Timer.....	238
<b>CMR</b>	
Common Register Bank Prefix (CMR) .....	45
<b>Command Sequence</b>	
Command Sequence Table.....	579
<b>Common Register Bank Prefix</b>	
Common Register Bank Prefix (CMR) .....	45
<b>Communication</b>	
Bi-directional Communication Function .....	522
Master/Slave Mode Communication Function .....	524
<b>Communication Prescaler Control Register</b>	
Communication Prescaler Control Register (SDCR).....	477
<b>Compare Clear Register</b>	
Compare Clear Register (CPCLR) .....	244
<b>Compare Register</b>	
Free-run Timer Clear Timing (for a Match with Compare Clear Register) .....	263
<b>Condition Code Register</b>	
Condition Code Register (CCR) .....	37
<b>Connected Detection</b>	
Connected Detection of External USB Device .....	345
<b>Connecting Serial Writing</b>	
Example of Connecting Serial Writing .....	602
<b>Connection Example</b>	
Connection Example in Single-chip Mode (when Using User Power).....	603
<b>Connection Status</b>	
Disconnection Status, Connection Status of the External USB Device .....	345
<b>Continuous Conversion</b>	
Operation of Continuous Conversion Mode .....	453
<b>Continuous Measurement</b>	
Single Measurement and Continuous Measurement ....	384
<b>Continuous Mode</b>	
Example of EI <sup>2</sup> OS Start Program in Continuous Mode .....	462, 465
<b>Control Status Registers</b>	
Control Status Registers (ICS01, ICS23).....	256
<b>Conversion Mode</b>	
Operation of Continuous Conversion Mode .....	453
Operation of Pause-conversion Mode .....	454

Operation of Single-shot Conversion Mode .....	452
<b>Conversion Operation</b>	
Conversion Operation Using $\mu$ DMAC or EI <sup>2</sup> OS .....	455
<b>Count Clock</b>	
Count Clock and Maximum Cycle .....	382
Count Clock Selection .....	377, 426
<b>Count Timing</b>	
Count Timing of Free-run Timer .....	263
<b>Counter</b>	
Counter Operation Mode .....	394
State Transition of Counter Operation .....	404
<b>Counter Operation</b>	
Measurement Mode and Counter Operation .....	385
<b>CPCLR</b>	
Compare Clear Register (CPCLR) .....	244
<b>CPU</b>	
Overview of CPU Memory Space .....	25
Overview of the CPU .....	24
<b>CPU Intermittent Operation</b>	
CPU Intermittent Operation Mode .....	145, 152
<b>CPU Operation Mode</b>	
CPU Operation Modes and Current Consumption .....	144
<b>Current Consumption</b>	
CPU Operation Modes and Current Consumption .....	144
<b>Cycle</b>	
Calculation Method of Pulse Width/cycle .....	386
Range of Count at Pulse Width/cycle .....	387
<b>D</b>	
<b>Data Bus</b>	
External 16-bit Bus Mode (External Data Bus 16-bit/ Multiplex Mode) .....	193
Pin State in External Bus 16-bit Data Bus and Multiplex 16-bit External Bus Mode .....	166
Pin State in External Bus 16-bit Data Bus and Non-multiplex 16-bit External Bus Mode .....	170
Pin State in External Bus 8-bit Data Bus and Multiplex 8-bit External Bus Mode .....	168
Pin State in External Bus 8-bit Data Bus and Non-multiplex 8-bit External Bus Mode .....	172
<b>Data Counter</b>	
Data Counter (DCT) .....	80
<b>Data Number Automatic Transfer</b>	
Data Number Automatic Transfer Mode .....	313
<b>Data Packet</b>	
Data Packet .....	350
<b>Data Polling Flag</b>	
State Transition of Data Polling Flag (DQ7) .....	582
<b>DBAPH</b>	
DMA Buffer Address Pointer (DBAPH/DBAPM/DBAPL) .....	102
<b>DBAPL</b>	
DMA Buffer Address Pointer (DBAPH/DBAPM/DBAPL) .....	102
<b>DBAPM</b>	
DMA Buffer Address Pointer (DBAPH/DBAPM/DBAPL) .....	102
<b>DCSR</b>	
DMA Descriptor Channel Specification Register (DCSR) .....	92
<b>DCT</b>	
Data Counter (DCT) .....	80
<b>DDCTH</b>	
About the Set Value of DMA Data Counter (DDCTH/DDCTL) .....	98
DMA Data Counter (DDCTH/DDCTL) .....	98
<b>DDCTL</b>	
About the Set Value of DMA Data Counter (DDCTH/DDCTL) .....	98
DMA Data Counter (DDCTH/DDCTL) .....	98
<b>DDR</b>	
Port Direction Register (DDR0 to DDRB) .....	203
<b>DDWR</b>	
Configuration of DMA Descriptor Window Register (DDWR) .....	97
<b>Dedicated Baud Rate Generator</b>	
Baud Rate of the External Clock Using the Dedicated Baud Rate Generator .....	512
Baud Rate of the Internal Clock Using the Dedicated Baud Rate Generator .....	511
<b>Dedicated Register</b>	
Dedicated Registers .....	32
<b>Delay Interruption</b>	
Notes on Use of Delay Interruption Generation Module (Delay Interruption Request Latch) .....	113
<b>Delay Interruption Generation Module</b>	
List of Register of Delay Interruption Generation Module .....	112
Notes on Use of Delay Interruption Generation Module (Delay Interruption Request Latch) .....	113
<b>Delay Time</b>	
Sub Clock Oscillation Stabilization Delay Time Function .....	238
<b>Delayed Interrupt Generation Module</b>	
Block Diagram of Delayed Interrupt Generation Module .....	112
Operation of Delayed Interrupt Generation Module .....	113
<b>Deleting Flash Memory</b>	
Method for Writing/deleting Flash Memory .....	574
<b>Deletion</b>	
When Chip/sector Deletion Operates .....	582
<b>DERH</b>	
DMA Permission Register (DERH/DERL) .....	96
<b>DERL</b>	
DMA Permission Register (DERH/DERL) .....	96
<b>Description</b>	
Description of Instruction Presentation Items and Symbols .....	652
<b>Descriptor</b>	
Configuration of Extended Intelligent I/O Service (EI <sup>2</sup> OS) Descriptor (ISD) .....	78

## INDEX

Each Register of DMA Descriptor .....	97
Detect Address	
Setting Detect Address .....	565
Detect Address Setting Registers	
Detect Address Setting Registers (PADR0, PADR1) .....	563
Device	
Cutting of Device.....	360
Precautions when Using Devices .....	20
DIOAH	
DMA I/O Register Address Pointer (DIOAH/DIOAL) .....	99
DIOAL	
DMA I/O Register Address Pointer (DIOAH/DIOAL) .....	99
Direct Addressing	
Direct Addressing .....	634
Direct Page Register	
Direct Page Register (DPR) <Initial Value: 01 <sub>H</sub> > .....	42
Disconnection Status	
Disconnection Status, Connection Status of the External USB Device.....	345
DIVR	
PWC Ratio of Dividing Frequency Control Register (DIVR).....	373
DMA	
Each Register of DMA Descriptor .....	97
DMA Buffer Address Pointer	
DMA Buffer Address Pointer (DBAPH/DBAPM/DBAPL).....	102
DMA Control Register	
DMA Control Register (DMACS).....	100
DMA Data Counter	
About the Set Value of DMA Data Counter (DDCTH/DDCTL) .....	98
DMA Data Counter (DDCTH/DDCTL) .....	98
DMA Descriptor Channel Specification Register	
DMA Descriptor Channel Specification Register (DCSR) .....	92
DMA Descriptor Window Register	
Configuration of DMA Descriptor Window Register (DDWR).....	97
DMA I/O Register Address Pointer	
DMA I/O Register Address Pointer (DIOAH/DIOAL) .....	99
DMA Permission Register	
DMA Permission Register (DERH/DERL) .....	96
DMA Status Register	
Bit Configuration of DMA Status Register (DSRH/DSRL).....	94
DMA Stop Status Register	
DMA Stop Status Register (DSSR) .....	95
DMACS	
DMA Control Register (DMACS).....	100
DPR	
Direct Page Register (DPR) <Initial Value: 01 <sub>H</sub> > .....	42
DQ3	
Transition State of Sector Erasing Timer Flag (DQ3) .....	586
DQ5	
Transition State of Timing Limit Over Flag (DQ5) .....	585
DQ6	
Transition State of Toggle Bit Flag (DQ6) .....	584
DQ7	
State Transition of Data Polling Flag (DQ7) .....	582
DSRH	
Bit Configuration of DMA Status Register (DSRH/DSRL).....	94
DSRL	
Bit Configuration of DMA Status Register (DSRH/DSRL).....	94
DSSR	
DMA Stop Status Register (DSSR) .....	95
DTB	
Bank Registers (PCB, DTB, USB, SSB, ADB) .....	41
DTP	
Block Diagram of DTP/External Interrupt .....	430
Operation of DTP .....	434
Operation Process of DTP/External Interrupt .....	436
Overview of DTP/External Interrupt .....	430
Register List of DTP/External Interrupt .....	431
DTP/Interrupt Permission Register	
DTP/Interrupt Permission Register (ENIR: Enable Interrupt Request Register) .....	431
DTP/Interrupt Request Register	
DTP/Interrupt Request Register (EIRR: External Interrupt Request Register) .....	432
E	
E <sup>2</sup> PROM	
E <sup>2</sup> PROM Memory Map .....	566
Operation of Address Match Detection Function at Storing Patch Program in E <sup>2</sup> PROM .....	569
System Configuration and E <sup>2</sup> PROM Memory Map.....	566
Each Register Operation	
Each Register Operation when Read Command Responds .....	303
Each Register Operation when Write Command Responds .....	304
Effective Address Field	
Effective Address Field.....	633, 651
EI <sup>2</sup> OS	
Configuration of Extended Intelligent I/O Service (EI <sup>2</sup> OS) Descriptor (ISD) .....	78
Conversion Operation Using $\mu$ DMAC or EI <sup>2</sup> OS .....	455
Example of EI <sup>2</sup> OS Start Program in Continuous Mode .....	462, 465
Example of EI <sup>2</sup> OS Start Program in Single Mode .....	459
Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	76
Extended Intelligent I/O Service (EI <sup>2</sup> OS) Processing Time (Time for One Transfer) .....	85

Extended Intelligent I/O Service (EI <sup>2</sup> OS) Status Register (ISCS) .....	81
Interrupt of Time-base Timer and EI <sup>2</sup> OS, $\mu$ DMAC .....	214
Interrupt of UART, EI <sup>2</sup> OS, and $\mu$ DMAC .....	505
Operation of Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	77, 83
Procedure for Use of Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	84
UART EI <sup>2</sup> OS Function .....	505
<b>EIRR</b>	
DTP/Interrupt Request Register (EIRR: External Interrupt Request Register) .....	432
<b>ELVR</b>	
Request Level Setting Register (ELVR: External Level Register) .....	432
<b>End Timing</b>	
Automatic Algorithm End Timing .....	578
Packet End Timing .....	356
<b>ENIR</b>	
DTP/Interrupt Permission Register (ENIR: Enable Interrupt Request Register) .....	431
<b>EOF Setting Register</b>	
EOF Setting Register (HEOF) .....	340
<b>EP0 Control Register</b>	
EP0 Control Register (EP0C) .....	276
<b>EP0 to EP5 Data Register</b>	
EP0 to EP5 Data Register (EP0DT to EP5DT) .....	297
<b>EP0C</b>	
EP0 Control Register (EP0C) .....	276
<b>EP0DT to EP5DT</b>	
EP0 to EP5 Data Register (EP0DT to EP5DT) .....	297
<b>EP0I Status Register</b>	
EP0I Status Register (EP0IS) .....	288
<b>EP0IS</b>	
EP0I Status Register (EP0IS) .....	288
<b>EP0O Status Register</b>	
EP0O Status Register (EP0OS) .....	290
<b>EP0OS</b>	
EP0O Status Register (EP0OS) .....	290
<b>EP1 to EP5 Control Register</b>	
EP1 to EP5 Control Register (EP1C to EP5C) .....	278
<b>EP1 to EP5 Status Register</b>	
EP1 to EP5 Status Register (EP1S to EP5S) .....	293
<b>EP1C to EP5C</b>	
EP1 to EP5 Control Register (EP1C to EP5C) .....	278
<b>EP1S to EP5S</b>	
EP1 to EP5 Status Register (EP1S to EP5S) .....	293
<b>EPCR</b>	
Bus Control Signal Selection Register (EPCR) .....	188
<b>Equipment</b>	
Condition of Peripheral Equipment Connected Outside .....	436
<b>Erase</b>	
Erasing All Data from Flash Memory (Chip Erase) .....	591
<b>Erase of Flash Memory</b>	
Write/Erase of Flash Memory .....	587
<b>Erasing</b>	
Erasing Any Data in Flash Memory (Sector Erasing) .....	592
<b>Erasing All Data</b>	
Erasing All Data from Flash Memory (Chip Erase) .....	591
<b>Erasing Any Data</b>	
Erasing Any Data in Flash Memory (Sector Erasing) .....	592
<b>Error Status</b>	
Error Status .....	355
<b>Event Count</b>	
Event Count Mode .....	409
Event Count Mode (External Clock Mode) .....	393
<b>Example</b>	
Example USB System Connection .....	301
<b>Execution Cycle Count</b>	
Calculating the Execution Cycle Count .....	649
Execution Cycle Count .....	648
<b>Extended I/O Serial Interface</b>	
Block Diagram in Extended I/O Serial Interface .....	470
Interrupt Function of Extended I/O Serial Interface .....	484
List of Register in Extended I/O Serial Interface .....	471
Outline of Extended I/O Serial Interface .....	470
Outline of Operation of Extended I/O Serial Interface .....	478
<b>Extended Intelligent I/O Service</b>	
Configuration of Extended Intelligent I/O Service (EI <sup>2</sup> OS) Descriptor (ISD) .....	78
Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	76
Extended Intelligent I/O Service (EI <sup>2</sup> OS) Processing Time (Time for One Transfer) .....	85
Extended Intelligent I/O Service (EI <sup>2</sup> OS) Status Register (ISCS) .....	81
Operation of Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	77, 83
Procedure for Use of Extended Intelligent I/O Service (EI <sup>2</sup> OS) .....	84
<b>External 16-bit Bus</b>	
External 16-bit Bus Mode (External Data Bus 16-bit/Multiplex Mode) .....	193
<b>External Address Output Control Register</b>	
External Address Output Control Register (HACR) .....	187
<b>External Bus</b>	
Pin State in External Bus 16-bit Data Bus and Multiplex 16-bit External Bus Mode .....	166
Pin State in External Bus 16-bit Data Bus and Non-multiplex 16-bit External Bus Mode .....	170
Pin State in External Bus 8-bit Data Bus and Multiplex 8-bit External Bus Mode .....	168
Pin State in External Bus 8-bit Data Bus and Non-multiplex 8-bit External Bus Mode .....	172
<b>External Bus 16-bit Data Bus</b>	
Pin State in External Bus 16-bit Data Bus and Non-multiplex 16-bit External Bus Mode .....	170

## INDEX

External Clock	
Baud Rate of the External Clock	
(One-to-one Mode).....	513
Baud Rate of the External Clock Using the Dedicated	
Baud Rate Generator.....	512
Connection of Oscillator and External Clock .....	141
Event Count Mode (External Clock Mode) .....	393
External Data Bus	
External 16-bit Bus Mode (External Data Bus 16-bit/	
Multiplex Mode).....	193
External Interrupt	
Block Diagram of DTP/External Interrupt.....	430
External Interrupt Operation .....	434
External Interrupt Request Level .....	436
Operation Process of DTP/External Interrupt.....	436
Overview of DTP/External Interrupt.....	430
Register List of DTP/External Interrupt.....	431
External Memory	
I/O Signal Pin of External Memory Access .....	184
External Memory Access	
External Memory Access Control Signal .....	191
External Pin	
Operation of External Pin in Each Mode.....	183
External Reset Pin	
Block Diagram of External Reset Pin.....	120
External Shift Clock	
External Shift Clock Mode .....	479
External USB	
Connected Detection of External USB Device .....	345
Disconnection Status, Connection Status of the External	
USB Device.....	345
<b>F</b>	
F <sup>2</sup> MC-16LX Instruction List	
F <sup>2</sup> MC-16LX Instruction List .....	655
Features	
Features of USB Function .....	268
Fetch	
Mode Fetch .....	122
Flag	
Hardware Sequence Flag .....	580
Receive Interrupt Generation and Flag Set Timing .....	506
State Transition of Data Polling Flag (DQ7).....	582
Transition State of Sector Erasing Timer Flag	
(DQ3) .....	586
Transition State of Timing Limit Over Flag (DQ5) .....	585
Transition State of Toggle Bit Flag (DQ6) .....	584
Transmit Interrupt Generation and Flag Set Timing ....	508
Flag Change Disable Prefix	
Flag Change Disable Prefix (NCC) .....	45
Flash Memory	
Erasing All Data from Flash Memory (Chip Erase) .....	591
Erasing Any Data in Flash Memory	
(Sector Erasing).....	592
Features of Flash Memory.....	574
Flash Memory Sector Erase Resumption .....	595
Flash Memory Sector Erase Suspension .....	594
Flash Memory Size and Products .....	574
Method for Writing/deleting Flash Memory.....	574
Read/Reset State in Flash Memory .....	588
Write/Erase of Flash Memory.....	587
Writing Data to Flash Memory .....	589
Writing Procedure of Flash Memory .....	589
Flash Memory Control Status Register	
Flash Memory Control Status Register	
(FMCS).....	577
Flash Microcontroller Programmer	
Example of Minimum Connection to Flash Microcontroller	
Programmer (when Using User Power) .....	605
Flash Microcontroller Programmer System	
Flash Microcontroller Programmer System .....	601
FMCS	
Flash Memory Control Status Register	
(FMCS).....	577
FRAME Setting Register	
FRAME Setting Register (HFRAME) .....	341
Free-run Timer	
Block Diagram of 16-bit Free-run Timer.....	244
Count Timing of Free-run Timer .....	263
Free-run Timer Clear Timing	
(for a Match with Compare Clear Register)	
.....	263
Operation of 16-bit Free-run Timer.....	258
Register List of 16-bit Free-run Timer .....	243
Fujitsu Standard	
Pins Used for Fujitsu Standard Serial On-board	
Programming.....	599
<b>G</b>	
General-purpose Register	
General-purpose Registers .....	34
<b>H</b>	
HACR	
External Address Output Control Register	
(HACR) .....	187
HADR	
Host Address Register (HADR) .....	339
Handshake Packet	
Handshake Packet.....	351
Hardware Component	
Initial Value of Hardware Component .....	427
Hardware Interrupt	
Construction of Hardware Interrupt.....	64
Function of Hardware Interrupt.....	63
Hardware Interrupt Processing Time .....	72
Hardware Interrupt Suppression .....	64
Operation Flow of Hardware Interrupt.....	68
Operation of Hardware Interrupt .....	67
Procedure for Using a Hardware Interrupt .....	69
Return from Hardware Interrupt.....	66
Start of Hardware Interrupt .....	66
Hardware Sequence	
Hardware Sequence Flag .....	580

<b>HCNT</b>		
Host Control Register 0, 1(HCNT0/HCNT1).....	324	
<b>HEOF</b>		
EOF Setting Register (HEOF) .....	340	
<b>HERR</b>		
Host Error Status Register (HERR) .....	331	
<b>HFCOMP</b>		
SOF Interruption FRAME Comparison Register (HFCOMP) .....	337	
<b>HFRAME</b>		
FRAME Setting Register (HFRAME) .....	341	
<b>HIRQ</b>		
Host Interruption Register (HIRQ).....	328	
<b>Holding Function</b>		
Operation of Holding Function .....	197	
<b>HOST</b>		
Feature of USB HOST .....	318	
Register of USB HOST .....	321	
Restriction on USB HOST .....	319	
Setting of HOST Function .....	345	
UART Block Diagram of USB HOST .....	320	
<b>Host</b>		
Wake-up from Host.....	310	
<b>Host Address Register</b>		
Host Address Register (HADR) .....	339	
<b>Host Control Register</b>		
Host Control Register 0, 1(HCNT0/HCNT1).....	324	
<b>Host Error Status Register</b>		
Host Error Status Register (HERR) .....	331	
<b>Host Interruption Register</b>		
Host Interruption Register (HIRQ).....	328	
<b>Host State Status Register</b>		
Host State Status Register (HSTATE) .....	334	
<b>Host Token Endpoint Register</b>		
Host Token Endpoint Register (HTOKEN).....	342	
<b>HRTIMER</b>		
Retry Timer Setting Register (HRTIMER).....	338	
<b>HSTATE</b>		
Host State Status Register (HSTATE) .....	334	
<b>HTOKEN</b>		
Host Token Endpoint Register (HTOKEN).....	342	
<b>I</b>		
<b>I/O Area</b>		
I/O Area .....	26	
<b>I/O Circuit</b>		
I/O Circuit Types.....	17	
<b>I/O Map</b>		
I/O Map .....	618	
<b>I/O Port</b>		
Functions of I/O Ports.....	200	
I/O Port Registers .....	201	
<b>I/O Register Address Pointer</b>		
I/O Register Address Pointer (IOA) .....	80	
<b>I/O Signal Pin</b>		
I/O Signal Pin of External Memory Access .....	184	
<b>I/O Timer</b>		
Configuration Function of 16-bit I/O Timer.....	240	
Operation and Timing of 16-bit I/O Timer.....	257	
Register Configuration of 16-bit I/O Timer.....	242	
<b>I<sup>2</sup>C Bus Address Register</b>		
I <sup>2</sup> C Bus Address Register 0 to 2 (IADR0 to IADR2) .....	544	
<b>I<sup>2</sup>C Bus Clock Control Register</b>		
I <sup>2</sup> C Bus Clock Control Register 0 to 2 (ICCR0 to ICCR2).....	542	
<b>I<sup>2</sup>C Bus Control Register</b>		
I <sup>2</sup> C Bus Control Register 0 to 2 (IBCR0 to IBCR2).....	537	
Notes on Use of I <sup>2</sup> C Bus Control Register 0 to 2 (IBCR0 to IBCR2).....	541	
<b>I<sup>2</sup>C Bus Data Register</b>		
I <sup>2</sup> C Bus Data Register 0 to 2 (IDAR0 to IDAR2) .....	545	
<b>I<sup>2</sup>C Bus Status Register</b>		
I <sup>2</sup> C Bus Status Register 0 to 2 (IBSR0 to IBSR2).....	535	
<b>I<sup>2</sup>C Interface</b>		
Flow of I <sup>2</sup> C Interface Mode Transitions.....	550	
I <sup>2</sup> C Interface Function .....	532	
Operation Flow of I <sup>2</sup> C Interface.....	551	
Register List of I <sup>2</sup> C Interface .....	534	
Transfer Flow of I <sup>2</sup> C Interface.....	548	
<b>IADR</b>		
I <sup>2</sup> C Bus Address Register 0 to 2 (IADR0 to IADR2) .....	544	
<b>IBCR</b>		
I <sup>2</sup> C Bus Control Register 0 to 2 (IBCR0 to IBCR2).....	537	
Notes on Use of I <sup>2</sup> C Bus Control Register 0 to 2 (IBCR0 to IBCR2).....	541	
<b>IBSR</b>		
I <sup>2</sup> C Bus Status Register 0 to 2 (IBSR0 to IBSR2).....	535	
<b>ICCR</b>		
I <sup>2</sup> C Bus Clock Control Register 0 to 2 (ICCR0 to ICCR2).....	542	
<b>ICR</b>		
Configuration of Interrupt Control Register (ICR).....	60	
Interrupt Control Registers (ICR00 to ICR15) .....	58	
<b>ICS</b>		
Control Status Registers (ICS01, ICS23).....	256	
<b>IDAR</b>		
I <sup>2</sup> C Bus Data Register 0 to 2 (IDAR0 to IDAR2) .....	545	
<b>ILM</b>		
Interrupt Level Mask Register (ILM) .....	38	
<b>IN</b>		
IN, OUT, SETUP Token .....	361	
<b>Indirect Addressing</b>		
Indirect Addressing.....	640	
<b>Initial Value</b>		
Direct Page Register (DPR) <Initial Value: 01 <sub>H</sub> >.....	42	
Initial Value of Hardware Component .....	427	

## INDEX

Input Capture	
Block Diagram of Input Capture .....	254
Example of Taking Timing of Input Capture .....	262
Input Capture .....	254
List of Register of Input Capture .....	255
Input Capture Data Register	
Input Capture Data Registers (IPCP0 to IPCP3) .....	255
Input Pulse Width	
Minimum Input Pulse Width.....	386
Instruction	
Description of Instruction Presentation Items and Symbols .....	652
Exception due to Execution of an Undefined Instruction .....	105
Execution of an Undefined Instruction .....	105
F2MC-16LX Instruction List .....	655
Instruction Types .....	631
Structure of Instruction Map .....	669
Instruction Presentation Items and Symbols	
Description of Instruction Presentation Items and Symbols .....	652
Internal Clock	
Baud Rate of the Internal Clock Using the Dedicated Baud Rate Generator.....	511
Internal Clock Mode .....	393
Internal Clock Mode (Single Shot Mode).....	407
Internal Clock Mode	
Operation of Internal Clock Mode (Reload Mode) .....	405
Internal Shift Clock	
Internal Shift Clock Mode.....	479
Interrupt	
Block Diagram of DTP/External Interrupt.....	430
Cancellation of Standby Mode by Interrupt .....	174
Construction of Hardware Interrupt .....	64
Example of Multiple Interrupts .....	71
Exception Processing Interrupt.....	88
External Interrupt Operation .....	434
External Interrupt Request Level .....	436
Function of Hardware Interrupt .....	63
Hardware Interrupt Processing Time .....	72
Hardware Interrupt Suppression .....	64
Interrupt Control Register Functions.....	57
Interrupt Control Register List .....	56
Interrupt Factors, Interrupt Vectors, and Interrupt Control Registers .....	628
Interrupt Generation Request .....	381
Interrupt of Time-base Timer .....	214
Interrupt of Time-base Timer and EI <sup>2</sup> OS, $\mu$ DMAC .....	214
Interrupt Request Generation .....	387
Interrupt Timing .....	264
Interrupts of 8/16-bit PPG Timer .....	427
Multiple Interrupts .....	70
Operation Flow of Hardware Interrupt .....	68
Operation of Hardware Interrupt.....	67
Operation of Software Interrupt.....	75
Operation Process of DTP/External Interrupt.....	436
Overview of DTP/External Interrupt.....	430
Precautions on Software Interrupt .....	75
Procedure for Using a Hardware Interrupt .....	69
Program Example of Interrupt Processing.....	108
Receive Interrupt Generation and Flag Set Timing .....	506
Register List of DTP/External Interrupt .....	431
Return from Hardware Interrupt .....	66
Return from Software Interrupt .....	74
SOF Interrupt .....	353
Stack Operation at the Start of Interrupt Processing .....	106
Stack Operation when Interrupt Processing Returns .....	106
Start of Hardware Interrupt .....	66
Start of Software Interrupt .....	74
Transition to Standby Mode and Interrupt .....	174
Transmit Interrupt Generation and Flag Set Timing .....	508
Type and Function of Interrupt .....	50
Type and Function of USB Interrupt .....	629
UART Interrupt .....	504
Interrupt Control Register	
Configuration of Interrupt Control Register (ICR) .....	60
Interrupt Control Register Functions .....	57, 61
Interrupt Control Register List .....	56
Interrupt Control Registers (ICR00 to ICR15) .....	58
Interrupt Factors, Interrupt Vectors, and Interrupt Control Registers .....	628
Interrupt Disable Instruction	
Interrupt Disable Instructions .....	47
Restrictions on Interrupt Disable Instructions and Prefix Instructions .....	47
Interrupt Factors	
Interrupt Factors, Interrupt Vectors, and Interrupt Control Registers .....	628
Interrupt Level Mask Register	
Interrupt Level Mask Register (ILM) .....	38
Interrupt Timing	
Interrupt Timing .....	264
Interrupt Vector	
Interrupt Factors, Interrupt Vectors, and Interrupt Control Registers .....	628
Interrupt Vector.....	53
Interruption	
Interruption Function of Interval of Watch Timer .....	238
Interruption of UART, EI <sup>2</sup> OS, and $\mu$ DMAC .....	505
Interval Timer	
Interval Timer Function .....	208
Operation of Interval Timer Function (Time-base Timer).....	215
IOA	
I/O Register Address Pointer (IOA) .....	80
IPCP	
Input Capture Data Registers (IPCP0 to IPCP3).....	255
ISCS	
Extended Intelligent I/O Service (EI <sup>2</sup> OS) Status Register (ISCS) .....	81
ISD	
Configuration of Extended Intelligent I/O Service (EI <sup>2</sup> OS) Descriptor (ISD) .....	78

**L****Latch**

- Notes on Use of Delay Interruption Generation Module (Delay Interruption Request Latch) ..... 113

**Low-power Consumption**

- Block Diagram of Low-power Consumption Control Circuit..... 147
- Operation Status in Low-power Consumption Mode ..... 163

**Low-power Consumption Mode Control Register**

- Access to Low-power Consumption Mode Control Register ..... 151
- Low-power Consumption Mode Control Register (LPMCR) ..... 149

**LPMCR**

- Low-power Consumption Mode Control Register (LPMCR) ..... 149

**LQFP-120**

- Package Dimension (LQFP-120) ..... 8
- Pin Assignment (LQFP-120) ..... 10

**M****M0**

- Set Bit of Bus Mode (M1, M0) ..... 181

**M1**

- Set Bit of Bus Mode (M1, M0) ..... 181

**Machine Clock**

- Machine Clock..... 138

**Main Clock Mode**

- Main Clock Mode, PLL Clock Mode, Sub Clock Mode..... 137

**Master/Slave Mode**

- Master/Slave Mode Communication Function ..... 524

**Maximum Cycle**

- Count Clock and Maximum Cycle ..... 382

**MB90330A Series**

- Block Diagram of the MB90330A Series ..... 7
- Feature of MB90330A Series ..... 2

**MD**

- Setting of Mode Pins (MD2 to MD0) ..... 179

**Measurement**

- Data of Measurement Result ..... 384
- Measurement Mode and Counter Operation ..... 385
- Operation Flow of Pulse Width Measurement ..... 388

**Memory Map**

- E<sup>2</sup>PROM Memory Map ..... 566
- Memory Map..... 614
- System Configuration and E<sup>2</sup>PROM Memory Map..... 566

**Memory Space**

- Multibyte Data Allocation in Memory Space..... 31
- Overview of CPU Memory Space..... 25

**μDMAC**

- Interrupt of Time-base Timer and EI<sup>2</sup>OS, μDMAC ..... 214

- μDMAC Function .....90
- μDMAC Register List .....91
- Use Procedure of μDMAC .....104

**Minimum Connection**

- Example of Minimum Connection to Flash Microcontroller Programmer (when Using User Power).....605

**Mode Data**

- Mode Data.....180
- Relation between Mode Pin and Mode Data (Recommended Example) .....182
- State of Pins after Mode Data Read .....125

**Mode Fetch**

- Mode Fetch.....122

**Mode Pin**

- Mode Pin .....121
- Relation between Mode Pin and Mode Data (Recommended Example) .....182
- Setting of Mode Pins (MD2 to MD0) .....179

**Mode Setting**

- Mode Setting.....178

**Mode Type**

- Mode Type .....190

**Multibyte Data**

- Accessing Multibyte Data .....31

**Multibyte Data Allocation**

- Multibyte Data Allocation in Memory Space .....31

**Multiple Interrupts**

- Example of Multiple Interrupts .....71
- Multiple Interrupts.....70

**Multiple Mode**

- Multiple Mode.....198

**Multiplex 16-bit External Bus**

- Pin State in External Bus 16-bit Data Bus and Multiplex 16-bit External Bus Mode .....166

**Multiplex 8-bit External Bus**

- Pin State in External Bus 8-bit Data Bus and Multiplex 8-bit External Bus Mode .....168

**Multiplex Mode**

- External 16-bit Bus Mode (External Data Bus 16-bit/Multiplex Mode) .....193

**Multiplication Rate**

- Selection of PLL Clock Multiplication Rate .....138

**N****NCC**

- Flag Change Disable Prefix (NCC).....45

**Non-multiplex 16-bit External Bus**

- Pin State in External Bus 16-bit Data Bus and Non-multiplex 16-bit External Bus Mode .....170

**Non-multiplex 8-bit External Bus**

- Pin State in External Bus 8-bit Data Bus and Non-multiplex 8-bit External Bus Mode .....172

**Non-multiplex Mode**

- Non-multiplex Mode.....197



## INDEX

### Notes

- Notes on Accessing the Low-Power Consumption Mode Control Register (LPMCR) to Enter the Standby Mode ..... 176

### NULL Transfer

- NULL Transfer Mode ..... 315

## O

### OCCP

- Output Compare Registers (OCCP0 to OCCP3) ..... 250

### OCS

- Output Compare Control Registers (OCS0 to OCS3) ..... 251

### ODR

- Port 4 Output Terminal Register (ODR4) ..... 204

### One-shot

- One-shot Operation Modes ..... 381

### One-to-one Mode

- Baud Rate of the External Clock (One-to-one Mode) ..... 513

### Operand

- 24-bit Operand Specification ..... 28

### Operating Mode

- Operating Mode ..... 178, 424

### Operating State

- Check Operating State ..... 380

### Operation

- Each Register Operation when Write Command Responds ..... 304
- Operation of USB Function ..... 298

### Operation Mode

- Counter Operation Mode ..... 394
- One-shot Operation Modes ..... 381
- Operation in Synchronous Mode (Operation Mode 2) ..... 519
- Reload Operation Mode ..... 381
- Selects Operation Mode ..... 377

### Operation Mode Control Register

- PPG0/2/4 Operation Mode Control Register (PPGC0/PPGC2/PPGC4) ..... 416
- PPG1/PPG3/PPG5 Operation Mode Control Register (PPGC1/PPGC3/PPGC5) ..... 418

### Operation State

- Operation State of Serial I/O ..... 480

### Operation Status

- Operation Status in Low-power Consumption Mode ..... 163
- Operation Status in Standby Mode ..... 153

### Oscillation Clock

- Oscillation Clock Frequency and Serial Clock Input Frequency ..... 600

### Oscillation Stabilization Wait Time

- Oscillation Stabilization Wait Time ..... 140, 175
- Oscillation Stabilization Wait Time Function ..... 215
- Reset Factors and Oscillation Stabilization Wait Times ..... 118

### Oscillation Stabilization Waiting

- Oscillation Stabilization Waiting Reset State ..... 119

### Oscillator

- Connection of Oscillator and External Clock ..... 141

### OUT

- IN, OUT, SETUP Token ..... 361

### Output Compare

- Block Diagram of Output Compare ..... 250
- List of Output Compare Registers ..... 249

### Output Compare Control Registers

- Output Compare Control Registers (OCS0 to OCS3) ..... 251

### Output Compare Registers

- Output Compare Registers (OCCP0 to OCCP3) ..... 250

### Output Control Register

- PPG0 to PPG5 Output Control Register (PPG01/PPG23/PPG45) ..... 421

### Output Waveform

- Example of Output Waveform ..... 260

### Overview

- Overview of CPU Memory Space ..... 25

## P

### Package Dimension

- Package Dimension (LQFP-120) ..... 8

### Packet

- Data Packet ..... 350
- Handshake Packet ..... 351
- Setting of Token Packet ..... 348

### Packet End

- Packet End Timing ..... 356

### Packet Transfer

- Packet Transfer Mode ..... 311

### PACSR

- Address Detection Control Register (PACSR) ..... 561

### PADR

- Detect Address Setting Registers (PADR0, PADR1) ..... 563

### Patch Program

- Operation of Address Match Detection Function at Storing Patch Program in E<sup>2</sup>PROM ..... 569

### Pause-conversion

- Operation of Pause-conversion Mode ..... 454

### PC

- Program Counter (PC) ..... 40

### PCB

- Bank Registers (PCB, DTB, USB, SSB, ADB) ..... 41

### PDR

- Port Data Register (PDR0 to PDRB) ..... 202

### Peripheral Equipment

- Condition of Peripheral Equipment Connected Outside ..... 436

### Physical Address

- Relation between Access Area and Physical Address ..... 181

<b>Pin</b>	
Pin State in External Bus 16-bit Data Bus and Non-multiplex 16-bit External Bus Mode .....	170
Pin State in External Bus 8-bit Data Bus and Non-multiplex 8-bit External Bus Mode .....	172
State of Pins after Mode Data Read .....	125
<b>Pin Assignment</b>	
Pin Assignment (LQFP-120) .....	10
<b>Pin Function</b>	
Pin Function .....	11
<b>PLL Clock</b>	
Selection of PLL Clock Multiplication Rate .....	138
<b>PLL Clock Mode</b>	
Main Clock Mode, PLL Clock Mode, Sub Clock Mode .....	137
<b>Port 0</b>	
Port 0, 1 Pull-up Resistance Register (RDR0, RDR1) .....	204
<b>Port 0 Pull-up Resistance Register</b>	
Port 0, 1 Pull-up Resistance Register (RDR0, RDR1) .....	204
<b>Port 1</b>	
Port 0, 1 Pull-up Resistance Register (RDR0, RDR1) .....	204
<b>Port 1 Pull-up Resistance Register</b>	
Port 0, 1 Pull-up Resistance Register (RDR0, RDR1) .....	204
<b>Port 4</b>	
Port 4 Output Terminal Register (ODR4) .....	204
<b>Port 4 Output Terminal Register</b>	
Port 4 Output Terminal Register (ODR4) .....	204
<b>Port Data Register</b>	
Port Data Register (PDR0 to PDRB) .....	202
<b>Port Direction Register</b>	
Port Direction Register (DDR0 to DDRB) .....	203
<b>PPG</b>	
PPG Output Operation .....	425
PPG0 to PPG5 Output Control Register (PPG01/PPG23/PPG45) .....	421
PPG0/2/4 Operation Mode Control Register (PPGC0/PPGC2/PPGC4) .....	416
PPG1/PPG3/PPG5 Operation Mode Control Register (PPGC1/PPGC3/PPGC5) .....	418
<b>PPG Reload Registers</b>	
PPG Reload Registers (PRL0 to PRL5, PRLH0 to PRLH5) .....	423
<b>PPG Timer</b>	
Block Diagram of 8/16-bit PPG Timer .....	413
Interrupts of 8/16-bit PPG Timer .....	427
Outline of Operation of 8/16-bit PPG Timer .....	424
Overview of 8/16-bit Timer PPG Timer .....	412
Register List of 8/16-bit PPG Timer .....	415
<b>PPGC</b>	
PPG0/2/4 Operation Mode Control Register (PPGC0/PPGC2/PPGC4) .....	416
PPG1/PPG3/PPG5 Operation Mode Control Register (PPGC1/PPGC3/PPGC5) .....	418
<b>Prefix Code</b>	
Consecutive Prefix Codes .....	47
<b>Prefix Instruction</b>	
Restrictions on Interrupt Disable Instructions and Prefix Instructions .....	47
<b>Priority</b>	
Priority Level of STP, SLP, and TMD Bit .....	151
<b>PRLH</b>	
PPG Reload Registers (PRL0 to PRL5, PRLH0 to PRLH5) .....	423
<b>PRL</b>	
PPG Reload Registers (PRL0 to PRL5, PRLH0 to PRLH5) .....	423
<b>Processing Time</b>	
Extended Intelligent I/O Service (EI <sup>2</sup> OS) Processing Time (Time for One Transfer) .....	85
Hardware Interrupt Processing Time .....	72
<b>Processor Status</b>	
Processor Status (PS) .....	37
<b>Product</b>	
Product Lineup .....	5
<b>Program</b>	
Program Execution .....	565
<b>Program Counter</b>	
Program Counter (PC) .....	40
<b>Program Example</b>	
Program Example for Address Match Detection Function .....	571
Program Example of Interrupt Processing .....	108
Program Example of Time-base Timer .....	219
Program Examples of Watchdog Timer .....	231
<b>Programming</b>	
Example of UART Programming .....	528
Pins Used for Fujitsu Standard Serial On-board Programming .....	599
<b>Programming Operation</b>	
At Programming Operation .....	582
<b>Protection Function</b>	
A/D-converted Data Protection Function .....	456
<b>PS</b>	
Processor Status (PS) .....	37
<b>Pulse</b>	
Calculation Method of Pulse Width/cycle .....	386
Pin Output Control of Pulse .....	426
Range of Count at Pulse Width/cycle .....	387
<b>Pulse Width</b>	
Calculation Method of Pulse Width/cycle .....	386
Minimum Input Pulse Width .....	386
Operation Flow of Pulse Width Measurement .....	388
Range of Count at Pulse Width/cycle .....	387
Relation between Reload Value and Pulse Width .....	425
<b>Pulse Width Measurement</b>	
Operation of Pulse Width Measurement Function .....	376
Startup and Stop of Timer/Pulse Width Measurement .....	379

## INDEX

PWC	
PWC Ratio of Dividing Frequency Control Register (DIVR) .....	373
PWC Control Status Register	
PWC Control Status Register (PWCSR) .....	367
PWC Data Buffer Register	
PWC Data Buffer Register (PWCR) .....	372
PWC Ratio of Dividing Frequency Control Register	
PWC Ratio of Dividing Frequency Control Register (DIVR) .....	373
PWC Timer	
Block Diagram of PWC Timer .....	365
Function of PWC Timer .....	364
Outline of PWC Timer Operation .....	374
Precautions when Using PWC Timer .....	389
Register List of PWC Timer .....	366
PWCR	
PWC Data Buffer Register (PWCR) .....	372
PWCSR	
PWC Control Status Register (PWCSR) .....	367
PWM Timer	
Operation of PWM Timer Functions .....	375
R	
RAM Area	
RAM Area .....	26
RDR	
Port 0, 1 Pull-up Resistance Register (RDR0, RDR1) .....	204
Reactivation	
Reactivation .....	379
Read	
Read/Reset State in Flash Memory .....	588
Read Command Responds	
Each Register Operation when Read Command Responds .....	303
Ready Function	
Ready Function .....	194
Recommended Example	
Relation between Mode Pin and Mode Data (Recommended Example) .....	182
Register	
Direct Page Register (DPR) <Initial Value: 01 <sub>H</sub> > .....	42
Each Register Operation when Read Command Responds .....	303
EP0 Control Register (EP0C) .....	276
EP0 to EP5 Data Register (EP0DT to EP5DT) .....	297
EP0I Status Register (EP0IS) .....	288
EP0O Status Register (EP0OS) .....	290
EP1 to EP5 Control Register (EP1C to EP5C) .....	278
EP1 to EP5 Status Register (EP1S to EP5S) .....	293
Register List of USB Function .....	270
Time Stamp Register (TMSP) .....	282
UDC Control Register (UDCC) .....	273
UDC Interruption Enable Register (UDCIE) .....	286
UDC Status Register (UDCS) .....	283
Register Bank	
Register Bank .....	43
Register Bank Pointer	
Register Bank Pointer (RP) .....	38
Reload	
Reload Operation Mode .....	381
Reload Mode	
Operation of Internal Clock Mode (Reload Mode) .....	405
Reload Register	
16-bit Reload Register 0 to 2 (TMRLR0 to TMRLR2) .....	402
16-bit Timer Register 0 to 2 (TMR0 to TMR2)/16-bit Reload Register 0 to 2 (TMRLR0 to TMRLR2) .....	401
Writing Timing to Reload Register .....	428
Reload Timer	
Block Diagram of 16-bit Reload Timer .....	395
Operation Modes of 16-bit Reload Timer .....	393
Overview of 16-bit Reload Timer .....	392
Register List of 16-bit Reload Timer .....	396
Setting of 16-bit Reload Timer .....	403
Reload Value	
Relation between Reload Value and Pulse Width .....	425
Timer Value and Reload Value .....	381
Remote	
Remote Wake-up .....	310
Request	
Interrupt Generation Request .....	381
Request Level	
External Interrupt Request Level .....	436
Request Level Setting Register	
Request Level Setting Register (ELVR: External Level Register) .....	432
Reset	
Block Diagram of External Reset Pin .....	120
Notes before and after Reset of USB Bus .....	347
Oscillation Stabilization Waiting Reset State .....	119
Overview of Reset Operation .....	121
Pin Status during Reset .....	125
Reset Factor	
Correspondence of Reset Factor Bit and Reset Factor .....	124
Notes on Reset Factor Bit .....	124
Reset Factor .....	116
Reset Factor Bit .....	123
Reset Factors and Oscillation Stabilization Wait Times .....	118
Reset State	
Read/Reset State in Flash Memory .....	588
Reset Values	
List of Registers and Reset Values of Address Match Detection Function .....	560
Resume Operation	
Resume Operation .....	357
Resumption	

Retry	
Retry Function .....	352
Retry Timer Setting Register	
Retry Timer Setting Register (HRTIMER).....	338
Return	
Return from Software Interrupt .....	74
ROM Area	
ROM Area .....	26
ROM Mirror Function Select Module	
Block Diagram of ROM Mirror Function Select Module .....	554
ROM Mirror Function Select Register	
ROM Mirror Function Select Register (ROMM) .....	555
ROM Mirror Function Selection Module	
Register of ROM Mirror Function Selection Module .....	554
ROMM	
ROM Mirror Function Select Register (ROMM) .....	555
RP	
Register Bank Pointer (RP) .....	38
<b>S</b>	
S0	
Set Bit of Various Modes (S1, S0) .....	180
S1	
Set Bit of Various Modes (S1, S0) .....	180
SCR	
Serial Control Register 0 to 3 (SCR0 to SCR3) .....	493
SDCR	
Communication Prescaler Control Register (SDCR) .....	477
SDR	
Serial Data Register (SDR) .....	476
Sector	
Sector Configuration .....	575
When Writing/chip Sector is Deleted.....	584, 585
Sector Deletion	
Procedure of Sector Deletion .....	592
When Sector Deletion Operate .....	586
When Chip/sector Deletion Operates .....	582
Sector Erase	
Flash Memory Sector Erase Resumption .....	595
Flash Memory Sector Erase Suspension .....	594
Sector Erasing	
Erasing Any Data in Flash Memory (Sector Erasing) .....	592
Sector Erasing being Suspended .....	582, 584, 586
Sector Erasing Timer Flag	
Transition State of Sector Erasing Timer Flag (DQ3) .....	586
Serial Clock	
Oscillation Clock Frequency and Serial Clock Input Frequency .....	600
Serial Control Register	
Serial Control Register 0 to 3 (SCR0 to SCR3) .....	493
Serial Data	
Operation during Transfer Serial Data .....	483
Serial Data Register	
Serial Data Register (SDR) .....	476
Serial I/O	
Operation State of Serial I/O .....	480
Serial Input Data Register	
Serial Input Data Register 0 to 3 (SIDR0 to SIDR3) .....	500
Serial Interface	
Block Diagram in Extended I/O Serial Interface .....	470
Interruption Function of Extended I/O Serial Interface .....	484
List of Register in Extended I/O Serial Interface .....	471
Outline of Extended I/O Serial Interface .....	470
Outline of Operation of Extended I/O Serial Interface .....	478
Serial Mode Control Status Register	
Serial Mode Control Status Register (SMCS).....	472
Serial Mode Register	
Serial Mode Register 0 to 3 (SMR0 to SMR3) .....	495
Serial On-board	
Pins Used for Fujitsu Standard Serial On-board Programming .....	599
The Serial On-board Writing Basic Component.....	598
Serial Output Data Register	
Serial Output Data Register 0 to 3 (SODR0 to SODR3) .....	501
serial programming connection	
Fujitsu Serial Programmer .....	608
Serial Status Register	
Serial Status Register 0 to 3 (SSR0 to SSR3) .....	497
SETUP Token	
IN, OUT, SETUP Token .....	361
Shift Operation	
Start/stop Timing of Shift Operation and Timing of I/O .....	482
SIDR	
Serial Input Data Register 0 to 3 (SIDR0 to SIDR3) .....	500
Single Measurement	
Single Measurement and Continuous Measurement .....	384
Single Mode	
Example of EI <sup>2</sup> OS Start Program in Single Mode.....	459
Single Shot	
Internal Clock Mode (Single Shot Mode) .....	407
Single-chip Mode	
Connection Example in Single-chip Mode (when Using User Power).....	603
Pin State in Single-chip Mode .....	164
Single-shot Conversion	
Operation of Single-shot Conversion Mode .....	452
Sleep Mode	
Cancellation of Sleep Modes.....	155
Transition to Sleep Mode .....	154

## INDEX

SLP	
Priority Level of STP, SLP, and TMD Bit.....	151
SMCS	
Serial Mode Control Status Register (SMCS) .....	472
SMR	
Serial Mode Register 0 to 3 (SMR0 to SMR3) .....	495
SODR	
Serial Output Data Register 0 to 3 (SODR0 to SODR3) .....	501
SOF	
SOF Interrupt.....	353
SOF Interruption FRAME Comparison Register	
SOF Interruption FRAME Comparison Register (HFCOMP) .....	337
SOF Token	
SOF Token.....	362
Software Interrupt	
Operation of Software Interrupt.....	75
Precautions on Software Interrupt .....	75
Return from Software Interrupt .....	74
Start of Software Interrupt.....	74
SSB	
Bank Registers (PCB, DTB, USB, SSB, ADB) .....	41
SSP	
User Stack Pointer (USP) and System Stack Pointer (SSP) .....	36
SSR	
Serial Status Register 0 to 3 (SSR0 to SSR3) .....	497
Stabilization	
Sub Clock Oscillation Stabilization Delay Time Function .....	238
Stack Area	
Stack Area.....	107
Stack Operation	
Stack Operation at the Start of Interrupt Processing .....	106
Stack Operation when Interrupt Processing Returns .....	106
Standby Mode	
Cancellation of Standby Mode by Interrupt .....	174
Operation Status in Standby Mode .....	153
Standby Mode .....	145
Transition to Standby Mode and Interrupt.....	174
Start Condition	
Start Condition .....	546
Startup	
Operation after Startup .....	379
Startup and Stop of Timer/Pulse Width Measurement .....	379
State	
Read/Reset State in Flash Memory .....	588
State Transition	
State Transition of Data Polling Flag (DQ7).....	582
State Transition Diagram	
State Transition Diagram .....	162
Status Register	
Extended Intelligent I/O Service (EI <sup>2</sup> OS) Status Register (ISCS) .....	81
Stop	
Startup and Stop of Timer/Pulse Width Measurement .....	379
Stops .....	380
Stop Condition	
Stop Condition .....	546
Stop Mode	
Cancellation of Stop Modes .....	160
Transition to Stop Mode .....	160
STP	
Priority Level of STP, SLP, and TMD Bit .....	151
Structure	
Structure of Instruction Map.....	669
Sub Clock Mode	
Main Clock Mode, PLL Clock Mode, Sub Clock Mode.....	137
Sub Clock Oscillation	
Sub Clock Oscillation Stabilization Delay Time Function .....	238
Suppression	
Hardware Interrupt Suppression .....	64
Suspend	
Suspend Processing.....	309
Suspend Operation	
Suspend Operation.....	357
Suspension	
Flash Memory Sector Erase Suspension .....	594
Synchronous Mode	
Operation in Synchronous Mode (Operation Mode 2).....	519
System Configuration	
System Configuration and E <sup>2</sup> PROM Memory Map.....	566
System Stack Pointer	
User Stack Pointer (USP) and System Stack Pointer (SSP) .....	36
<b>T</b>	
Taking Timing	
Example of Taking Timing of Input Capture .....	262
TBTC	
Time-base Timer Control Register (TBTC).....	212
TCCS	
Timer Control Status Register (TCCS) .....	245
TCDT	
Timer Counter Data Register (TCDT) .....	245
Time Stamp Register	
Time Stamp Register (TMSP).....	282
Time-base Timer	
Block Diagram of Time-base Timer .....	210
Interrupt of Time-base Timer .....	214
Interrupt of Time-base Timer and EI <sup>2</sup> OS, $\mu$ DMAC .....	214

Operation of Interval Timer Function (Time-base Timer) .....	215
Operations of Time-base Timer .....	218
Precautions when Using Time-base Timer .....	217
Program Example of Time-base Timer .....	219
<b>Time-base Timer Control Register</b>	
Time-base Timer Control Register (TBTC) .....	212
<b>Time-base Timer Mode</b>	
Cancellation of Time-base Timer Modes .....	157
Transition to Time-base Timer Mode .....	156
<b>Timer</b>	
Operation Flow of Timer .....	383
Startup and Stop of Timer/Pulse Width Measurement .....	379
<b>Timer Control Status Register</b>	
Timer Control Status Register (TCCS) .....	245
Timer Control Status Register 0 to 2 (TMCSR0 to TMCSR2) .....	397
<b>Timer Counter Data Register</b>	
Timer Counter Data Register (TCDDT) .....	245
<b>Timer Cycle</b>	
Timer Cycle .....	382
<b>Timer Register</b>	
16-bit Timer Register 0 to 2 (TMR0 to TMR2) .....	401
16-bit Timer Register 0 to 2 (TMR0 to TMR2)/ 16-bit Reload Register 0 to 2 (TMRLR0 to TMRLR2) .....	401
<b>Timer Value</b>	
Timer Value and Reload Value .....	381
<b>Timing</b>	
Writing Timing to Reload Register .....	428
<b>Timing Limit Over Flag</b>	
Transition State of Timing Limit Over Flag (DQ5) .....	585
<b>TMCSR</b>	
Timer Control Status Register 0 to 2 (TMCSR0 to TMCSR2) .....	397
<b>TMD Bit</b>	
Priority Level of STP, SLP, and TMD Bit .....	151
<b>TMR</b>	
16-bit Timer Register 0 to 2 (TMR0 to TMR2) .....	401
16-bit Timer Register 0 to 2 (TMR0 to TMR2)/ 16-bit Reload Register 0 to 2 (TMRLR0 to TMRLR2) .....	401
<b>TMRLR</b>	
16-bit Reload Register 0 to 2 (TMRLR0 to TMRLR2) .....	402
16-bit Timer Register 0 to 2 (TMR0 to TMR2)/ 16-bit Reload Register 0 to 2 (TMRLR0 to TMRLR2) .....	401
<b>TMSP</b>	
Time Stamp Register (TMSP) .....	282
<b>Toggle Bit Flag</b>	
Transition State of Toggle Bit Flag (DQ6) .....	584
<b>Token</b>	
IN, OUT, SETUP Token .....	361
SOF Token .....	362
<b>Token Packet</b>	
Setting of Token Packet .....	348
<b>Transfer</b>	
Data Number Automatic Transfer Mode .....	313
NULL Transfer Mode .....	315
Packet Transfer Mode .....	311
<b>Transfer Speed</b>	
Acquiring Transfer Speed of Destination USB Device and Selecting Clock .....	345
<b>Transition</b>	
State Transition of Counter Operation .....	404
Transition to Sleep Mode .....	154
<b>Transition State</b>	
Transition State of Sector Erasing Timer Flag (DQ3) .....	586
Transition State of Timing Limit Over Flag (DQ5) .....	585
<b>U</b>	
<b>UART</b>	
Example of UART Programming .....	528
Interruption of UART, EI <sup>2</sup> OS, and $\mu$ DMAC .....	505
List of UART Register .....	492
Notes on Using UART .....	527
Operation of UART .....	514
UART Baud Rate Selection .....	510
UART Block Diagram .....	488
UART Block Diagram of USB HOST .....	320
UART EI <sup>2</sup> OS Function .....	505
UART Function .....	486
UART Interrupt .....	504
UART Pins .....	491
<b>UART Prescaler Control Register</b>	
UART Prescaler Control Register 0 to 3 (UTCRO to UTCRC3) and UART Prescaler Reload Register 0 to 3 (UTRLR0 to UTRLR3) .....	502
<b>UDC Control Register</b>	
UDC Control Register (UDCC) .....	273
<b>UDC Interruption Enable Register</b>	
UDC Interruption Enable Register (UDCIE) .....	286
<b>UDC Status Register</b>	
UDC Status Register (UDCS) .....	283
<b>UDCC</b>	
UDC Control Register (UDCC) .....	273
<b>UDCIE</b>	
UDC Interruption Enable Register (UDCIE) .....	286
<b>UDCS</b>	
UDC Status Register (UDCS) .....	283
<b>Undefined Instruction</b>	
Exception due to Execution of an Undefined Instruction .....	105
Execution of an Undefined Instruction .....	105
<b>USB</b>	
Acquiring Transfer Speed of Destination USB Device and Selecting Clock .....	345
Bank Registers (PCB, DTB, USB, SSB, ADB) .....	41
Block Diagram of USB Function .....	269
Connected Detection of External USB Device .....	345

## INDEX

Disconnection Status, Connection Status of the External USB Device.....	345
Feature of USB HOST .....	318
Features of USB Function .....	268
Notes before and after Reset of USB Bus.....	347
Operation of USB Function .....	298
Register List of USB Function .....	270
Register of USB HOST .....	321
Restriction on USB HOST .....	319
UART Block Diagram of USB HOST .....	320
<b>USB HOST</b>	
Operation of USB HOST .....	344
<b>USB Interrupt</b>	
Type and Function of USB Interrupt.....	629
<b>USB System Connection</b>	
Example USB System Connection .....	301
<b>User Power</b>	
Connection Example in Single-chip Mode (when Using User Power) .....	603
Exempl of Minimum Connection to Flash Microcontroller Programmer (when Using User Power) .....	605
<b>User Stack Pointer</b>	
User Stack Pointer (USP) and System Stack Pointer (SSP) .....	36
<b>USP</b>	
User Stack Pointer (USP) and System Stack Pointer (SSP) .....	36
<b>UTCR</b>	
UART Prescaler Control Register 0 to 3 (UTCR0 to UTCR3) and UART Prescaler Reload Register 0 to 3 (UTRLR0 to UTRLR3) .....	502
<b>UTRLR</b>	
UART Prescaler Control Register 0 to 3 (UTCR0 to UTCR3) and UART Prescaler Reload Register 0 to 3 (UTRLR0 to UTRLR3) .....	502
<b>V</b>	
<b>Various Modes</b>	
Set Bit of Various Modes (S1, S0) .....	180
<b>W</b>	
<b>Wake-up</b>	
Remote Wake-up .....	310
Wake-up from Host .....	310
<b>Watch Counter</b>	
Watch Counter .....	238
<b>Watch Mode</b>	
Cancellation of Watch Mode .....	158
Transition to Watch Mode .....	158
<b>Watch Timer</b>	
Block Diagram of Watch Timer .....	235
Functions of Watch Timer .....	234
Interruption Function of Interval of Watch Timer .....	238
<b>Watch Timer Control Register</b>	
Configuration of Watch Timer Control Register (WTC) .....	236
<b>Watchdog Timer</b>	
Block Diagram of Watchdog Timer .....	226
Functions of Watchdog Timer .....	222
Operations of Watchdog Timer .....	228
Precautions when Using Watchdog Timer .....	230
Program Examples of Watchdog Timer .....	231
Specification Function of Clock Source of Watchdog Timer .....	238
<b>Watchdog Timer Control Register</b>	
Watchdog Timer Control Register (WDTC).....	224
<b>Waveform</b>	
Example of Output Waveform .....	260
<b>WDTC</b>	
Watchdog Timer Control Register (WDTC).....	224
<b>Write</b>	
Write/Erase of Flash Memory .....	587
<b>Write Command Responds</b>	
Each Register Operation when Write Command Responds .....	304
<b>Writing</b>	
Method for Writing/deleting Flash Memory .....	574
When Writing/chip Sector is Deleted .....	584, 585
<b>Writing Data</b>	
Writing Data to Flash Memory .....	589
<b>Writing Procedure</b>	
Writing Procedure of Flash Memory .....	589
<b>Writing Timing</b>	
Writing Timing to Reload Register .....	428
<b>WTC</b>	
Configuration of Watch Timer Control Register (WTC) .....	236

CM44-10129-6E

---

**FUJITSU MICROELECTRONICS • CONTROLLER MANUAL**

F<sup>2</sup>MC-16LX

16-bit Microcontroller

MB90330A Series

HARDWARE MANUAL

---

October 2009 the sixth edition

Published **FUJITSU MICROELECTRONICS LIMITED**

Edited Sales Promotion Dept.

---



